HAW HAMBURG

BACHELOR THESIS
Dennis Sentler

Detektion von Sprachbefehlen auf Edge-Geräten unterstützt durch automatisierte Trainingsdaten-Synthese für eine Not-Halt-Anwendung

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Dennis Sentler

Detektion von Sprachbefehlen auf Edge-Geräten unterstützt durch automatisierte Trainingsdaten-Synthese für eine Not-Halt-Anwendung

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Technische Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stephan Pareigis
Zweitgutachter: Prof. Dr. Peer Stelldinger

Eingereicht am: 22. August 2022

Dennis Sentler

Thema der Arbeit

Detektion von Sprachbefehlen auf Edge-Geräten unterstützt durch automatisierte Trainingsdaten-Synthese für eine Not-Halt-Anwendung

Stichworte

Edge Computing, Künstliche Intelligenz, AI on Edge, Automatisiertes Training, Robotik, Smart City, Not-Halt, Sprachsteuerung, Mensch-Maschinen-Interaktion

Kurzzusammenfassung

Ziel dieser Arbeit ist es, einen durch Sprachbefehle gesteuerten Not-Halt-Kontroller zu entwickeln, der auf einem Edge-Gerät mit geringem Leistungsvermögen betrieben werden kann. Sekundäres Ziel ist eine Trainingsanwendung, die den aufwendigen Trainingsprozess mithilfe von synthetischen Trainingsdaten automatisieren kann. Dabei stellen sich zusätzliche Fragen: Welche neuronale Netzwerkarchitektur eignet sich für eine Herunterskalierung auf Edge-Geräte-Niveau? Kann State of the Art Sprachsynthese den manuellen Arbeitsaufwand im vorliegenden Fall ersetzen?

Es ist möglich, dass für neuronale Netze ein synthetischer Datensatz noch kein voller Ersatz ist. Andererseits hat die Sprachsynthese in den letzten Jahren große Fortschritte gezeigt.

Für die Umsetzung wird eine aus dem MobileNet [1] bekannte DS-CNN-Architektur verwendet und herunterskaliert. Das Training wird auf dem Google Speech Command Dataset [2] sowie auf einem synthetischen Datensatz umgesetzt. Die Audiosynthese erfolgt über eine eigene GUI-Anwendung, die mit führenden Online-TTS-Anbietern integriert ist.

Es stellt sich heraus, dass die DS-CNN-Architektur grundsätzlich für diesen Kontext geeignet ist, die Trainingsdaten aber über den Erfolg bestimmen. Im Vergleich können die allein synthetisch-trainierten Netzwerke bei der Detektion nicht mithalten. Allerdings konnte mit dem Google Speech Command Dataset [2] der Not-Halt-Kontroller erfolgreich umgesetzt werden.

Dennis Sentler

Title of Thesis

Detection of voice commands on edge devices supported by automated training data synthesis for an emergency stop application

Keywords

Edge Computing, Artificial Intelligence, AI on Edge, Automated Training, Robotics, Smart City, Emergency Stop, Voice Control, Human–Machine Interface

Abstract

The goal of this work is to develop an emergency stop controller operated by voice commands that can be deployed on an edge device with a small footprint. The secondary goal is a training application that automates the time-consuming training process using synthetic training data. Several questions arise: Which neural network architecture is suitable for downscaling to edge device capabilities? Can state-of-the-art speech synthesis replace the need to manually collect and categorize speech recordings in our case?
It is possible that for neural networks, a synthetic dataset is still not a full substitute. On the other hand, speech synthesis has shown growing potential in recent years.
For the implementation, a downscaled DS-CNN architecture known from MobileNet [1] is used. Training is performed on the Google Speech Command Dataset [2] and on a synthetic dataset. Audio synthesis is accomplished using a custom GUI application integrated with leading online TTS providers.
It is found that the DS-CNN architecture is fundamentally suitable for this context, but the training data dictates its success; the networks trained on only synthetic data cannot keep up with those trained on real audio data. However, by using the Google Speech Command Dataset [2], the emergency stop controller has been successfully implemented.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

Acc. Accuracy *(ger.: Genauigkeit)*.

AMS Autonomous Mobile System.

CLI Command Line Interface.

CNN Convolutional Neural Network.

DS-CNN Depthwise Separable Convolutional Neural Network.

GUI Graphical User Interface.

HAW Hochschule für Angewandte Wissenschaften.

KI Künstliche Intelligenz *(eng.: artificial intelligence)*.

KWS Keyword Spotting *(ger.: Erkennung von Schlüsselbegriffen)*.

LED Lichtemittierende Diode.

LSTM Long short-term memory.

MCU Microcontroller Unit *(ger.: Mikrokontroller)*.

MFCC Mel Frequency Cepstral Coefficient.

MSS Multi-Sensor-System.

NN Neuronales Netzwerk.

REQ Requirement.

RNN Recurrent Neural Network.

SDK Software Development Kit.

TDNN Time Delay Neural Network.

TIQ Test Area Intelligent Quartier Mobility.

TPU Tensor Processing Unit.

TTS Text-to-Speech.

UC Use-Case.

YAML YAML Ain't Markup Language [4].

Glossar

Data Augmentation Bei Data Augmentation handelt es sich um eine Reihe von Techniken zur künstlichen Vergrößerung der Datenmenge durch Generierung neuer Datenpunkte aus vorhandenen Daten [5].

harte Echtzeitanforderungen Das System muss im definierten Zeitfenster das Ergebnis vollständig präsentieren.

HAW Hamburg Die Hochschule für Angewandte Wissenschaften (HAW) Hamburg ist die vormalige Fachhochschule am Berliner Tor.

Husky Ein konkretes Autonomous Mobile System (AMS) bestehend aus einer Fahrplattform und einem montiertem Greifarm. Genutzt innerhalb der HAW.

kwsCoach Im Kontext dieser Bachelorarbeit entwickelte Anwendung zur automatisierten Synthese von Trainingsdaten und anschließendem Training eines neuronalen Modells.

Mel Frequency Cepstral Coefficient Eine Singalverarbeitungsmethode, die dafür ausgelegt ist, aus Audiosignalen Charakteristika von menschlicher Sprache hervorzuheben. Dabei wird auch das Informationsvolumen stark komprimiert [6].

Not-Aus Eine Handlung im Notfall, die dazu bestimmt ist, die Versorgung mit elektrischer Energie ganz oder teilweise eines technischen Gerätes abzuschalten, bei der ein Risiko für einen elektrischen Schlag oder ein anderes Risiko elektrischen Ursprungs besteht [7].

Not-Halt Eine Handlung im Notfall, die dazu bestimmt ist, einen gefahrbringenden Prozess oder eine gefahrbringende Bewegung anzuhalten. Die Energieversorgung wird dadurch nicht zwingend unterbrochen [7].

Requirement Ein Requirement *(engl.)* ist eine Anforderung, durch die Ziele während der Software-Konzeptionsphase formuliert werden. Es wird unterschieden zwischen funktionalen (Funktionen) und nichtfunktionalen (Qualität) Anforderungen.

Text-to-Speech Eine Technologie, die geschriebenen Text in gesprochene Sprache umwandelt.

Use-Case Ein Use-Case *(engl.)* ist ein Anwendungsfall für einen potentiellen Nutzer einer Anwendung.

1 Einleitung

1.1 Kontext

Dem Projekt Test Area Intelligent Quartier Mobility (TIQ)[8] an der HAW Hamburg liegt die Idee zugrunde, einen repräsentativen Ausschnitt aus der städtischen Umgebung mit einer intelligenten Infrastruktur und autonomen, mobilen Systemen neu zu gestalten. Ziel ist es, die wachsenden Mobilitätsanforderungen der Gesellschaft unter Berücksichtigung moderner sozialethischer und umweltlicher Aspekte zu erfüllen.

Als Teil des Projektes wird ein konkretes Autonomous Mobile System (AMS) namens Husky entwickelt. Dieses ist repräsentativ für eine AMS Plattform mittlerer Größe, die sich sowohl innerhalb von Gebäuden als auch außerhalb, in einem weitläufigeren Gebiet, fortbewegen kann. Es ist zu erwarten, dass diese Plattform innerhalb von Gebäuden mit Passanten, weiteren AMS und Hindernissen in Kontakt kommt. Außerhalb der Gebäude kommen noch Interaktionen zwischen der intelligenten Multi-Sensor-System (MSS)-Infrastruktur, Fahrzeugen und sonstigen Verkehrsbeteiligten hinzu. Eine Priorität hierbei ist, dass die Freiheit, Gesundheit und Rechte der anderen Verkehrsteilnehmer nicht verletzt werden.

1.2 Problem

Wenn autonome, intelligente Systeme sich in der Gesellschaft bewegen, müssen sie sich so verhalten, dass Unfälle mit Menschen oder andere Gefährdung um jeden Preis verhindert werden. Allerdings reicht es nicht aus, sich auf die korrekten Aktionen und Reaktionen der AMS zu verlassen.

Aus diversen Gründen muss mit einer Notsituation gerechnet werden:

- **Fehler**: Design- und Programmierfehler, Schäden oder Ausfälle in einem komplexen, verteilten System können eine autonome, korrekte Reaktion des AMS verhindern und dadurch Menschenleben gefährden.
- **Unberechenbares menschliches Verhalten**: Unerfahrene, außenstehende Personen können Aktionen von AMS nicht einschätzen und dadurch sich selbst in eine gefährdende Situation bringen.
- **Design-Lücke**: Bei der hohen Vielfalt an menschlichen Interaktionen kann ein AMS sehr wahrscheinlich nicht alle Situationen korrekt deuten und angemessen reagieren - ein zusätzliches Risiko für Fehlhandlungen.

Im Falle einer Notsituation zwischen Passant und einem AMSmuss eine Kontrollübergabe von letzterem an den Passanten stattfinden. Ein Not-Halt-System ist dafür bestens geeignet.

1.2.1 Notschalter

An konventionellen Maschinen im industriellen Umfeld wird dafür ein Notschalter bereitgestellt, mit dem geschultes Personal einen Not-Aus oder Not-Halt einleiten und auch wieder beenden.

Lässt sich der Notschalter auf diese Weise auch im neuen Kontext der AMS verwenden? Für mich hat diese Variante zwei Kernprobleme:

1. **Personen müssen geschult sein.** Auch wenn dies im technischen Bereich oft für selbstverständlich gehalten wird, ist der klassische Notschalter zwar bekannt, der größte Teil der Bevölkerung hat einen solchen bisher aber noch nicht bedient.
2. **Notschalter ist am beweglichen Körper montiert.** Üblicherweise wird ein Notschalter an einem Steuergerät montiert, das der Kontrolle des Personals unterliegt. Im vorliegenden Fall ist das nicht möglich. Der Notschalter könnte nur direkt am Husky montiert werden. Das hat zur Folge, dass:
    a) ein Passant sich möglicherweise in einen Gefahrenbereich begeben müsste, um den Schalter zu betätigen

b) Passanten mit Bewegungseinschränkungen keine Möglichkeit haben, den Schalter zu erreichen

c) der Husky sich wegbewegen könnte und damit unabsichtlich den Abschaltversuch eines Passanten sabotiert

Dennoch ist der Notschalter eine absolute Notwendigkeit und hat auch eine Daseinsberechtigung: Entwickler, Wartungspersonal oder weiteres geschultes Servicepersonal (z. B. Logistiker, Paketshop-Betreiber u. v. m.) können diese Möglichkeit nutzen.

Für den öffentlichen Raum ist der Notschalter allein unzureichend und muss durch eine zusätzliche Schnittstelle ergänzt werden.

1.2.2 Sprachgesteuerter Not-Halt

Stetig steigende Verkaufszahlen von intelligenten Lautsprechern von Google, Amazon oder Apple sind ein Beispiel für zunehmende Beliebtheit und Akzeptanz der integrierten Sprachassistenten [9]. Auch in modernen Autos werden heutzutage eigene Sprachassistenten verwendet. Dies fördert bei den Nutzern das Vertrauen und die Anwendungssicherheit bei der Sprachsteuerung.

Die Ergebnisse aus meiner Vorstudie *Fragebogen: Brauchen die Roboter von morgen ein neues Not-Halt-System?* bestätigen, dass die Umfrageteilnehmer in einer sich anbahnenden Notsituation einen Sprachbefehl anderen Methoden vorziehen würden. Deswegen halte ich die Sprachsteuerung auch bei der Not-Halt-Anwendung für eine geeignete Methode. Der sprachgesteuerte Not-Halt sollte ausschließlich als Ergänzung zum klassischen Notschalter eingesetzt werden, nicht als Ersatz dafür.

1.3 Anforderungen

Intuitiv
Die sprachgesteuerte Not-Halt-Lösung muss sowohl für fachkundige Personen als auch für fachfremde Passanten intuitiv anwendbar sein. Nur dann gibt es einen Mehrwert gegenüber der klassischen Notschalterlösung.

Konkret bedeutet dies, dass die möglichen Sprachbefehle sowie die Tatsache, dass die Sprachbefehle vom AMS verstanden werden, dem nicht fachkundigen Passanten klar sein sollten, ohne dass dieser darüber aufgeklärt wurde.

Echtzeitfähigkeit
Bei einem Not-Halt handelt es sich um eine Funktionalität, die eine direkte Auswirkung darauf haben kann, ob eine Person zu Schaden kommt oder ob auch das jeweilige AMS einen mechanischen Schaden erleidet. Deswegen unterliegt diese Funktion den harten Echtzeitanforderungen [10]. Die benötigte Zeit zur Berechnung des Ergebnisses, ob ein Not-Halt-Sprachbefehl gegeben wurde oder nicht, darf eine zuvor definierte Antwortzeit nicht überschreiten.

Hardware Entkopplung
Die Not-Halt-Anwendung muss auf gesonderter, dedizierter Hardware betrieben werden. Dadurch wird unterbunden, dass sich Fehlerzustände aus dem restlichen System direkt auf die Auswertung von Sprachbefehlen auswirken können. Zusätzlich ist dies notwendig, um die harten Echtzeitanforderungen zu erfüllen.

Erweiterbarkeit
Eine weitere Anforderung ist die leichte Erweiterbarkeit der Not-Halt-Anwendung. Wenn die Erkennung der Sprachbefehle mithilfe von maschinellem Lernen erfolgt, ist das daraus resultierende KI-Modell von einem eingeschränkten Datensatz abhängig.

Es sollte jedoch möglich sein, dem Modell zu einem späteren Zeitpunkt mit einfachen Mitteln neue Sprachbefehle hinzuzufügen. Nur so kann in Zukunft ein Rollout auf andere AMS (mit ggf. anderen Namen) ermöglicht werden.

2 Analyse

2.1 Auswertung Fragebogen

Als Vorstudie für die Orientierung der Arbeit wurde ein Fragebogen konzipiert und als Online-Umfrage verteilt. In diesem Kapitel wird auf ursprüngliche Absichten und Ziele, Teilnehmerkreis, Umfrageergebnisse und zuletzt auf die davon abgeleiteten Schlussfolgerungen eingegangen.

2.1.1 Ziele

Diese Vorstudie hat das Ziel, mehrere Kernfragen für die Implementation eines Not-Halt-Systems zu beantworten. Zusätzlich sollen Informationen darüber gesammelt werden, wie Menschen über die Robotik in der Öffentlichkeit und damit verbundene Notsituationen denken.

Primäre Fragen sind:

- Ist ein sprachgesteuerter Not-Halt-Auslöser intuitiv?
- Wie kann ein Roboter bei einem Sprachbefehl adressiert werden?

Sekundäre Fragen sind:

- Welche Sprachbefehle sind geeignet?
- Gibt es Alternativen zur Sprachsteuerung?

2.1.2 Teilnehmerkreis

Diese Umfrage wurde größtenteils im Bekanntenkreis und unter Kommilitonen verteilt. Insgesamt haben 54 verschiedene Personen daran teilgenommen. Um untersuchen zu können, ob die Umfrageergebnisse und daraus abgeleiteten Rückschlüsse repräsentativ für die Gesellschaft sind, wurden auch persönliche Angaben erfasst. Diese waren allerdings optional und nicht jeder Teilnehmer hat eine Angabe dazu gemacht.

Alter	**Anzahl**	**Verhältnis**
15–19	4	7,4 %
20–29	30	55,6 %
30–39	13	24,1 %
40–49	2	3,7 %
50–59	3	5,6 %
> 60	1	1,9 %
Gesamt	53	98,1%

Tabelle 2.1: Umfrageteilnehmer nach Alter

Im Teilnehmerkreis sind zwei Altersklassen überrepräsentiert. Die Altersklasse 20–29 ist am stärksten vertreten mit 55,6 %. Die nächststärkste Altersklasse 30–39 hat einen Anteil von 24,1 % (Tabelle 2.1). Diese Klassen dominieren den Teilnehmerkreis mit ca. 80 %. Die restlichen vier Altersklassen haben insgesamt nur einen Anteil von 20 %. Diese Verteilung ist suboptimal und lässt nicht zu, dass die Ergebnisse altersunabhängig interpretiert werden können. Alle aus diesen Umfrageergebnissen abgeleiteten Rückschlüsse sind ebenfalls nur auf die überrepräsentierte Altersklasse 20–39 anzuwenden.

Neben dem Alter wurde der technische Hintergrund der jeweiligen Teilnehmer erfasst. Vier verschiedene Kategorien stufen das potenzielle Interesse und Kenntnisse über technische Systeme ab (Tabelle 2.2).

Technischer Hintergrund	**Anzahl**	**Verhältnis**
Beruf oder Ausbildung mit rein technischem Bezug	13	24,1 %
Beruf oder Ausbildung mit technischem Bezug	11	20,4 %
Technisch interessiert	17	31,5 %
Technisch wenig interessiert	11	20,4 %
Gesamt	52	96,3 %

Tabelle 2.2: Umfrageteilnehmer nach technischem Hintergrund

Hat eine befragte Person eine Ausbildung oder einen Job in einer rein technischen Berufsgruppe, so wird angenommen, dass auch technische Kenntnisse ausgeprägt vorhanden sind. Wenn andererseits weder eine technische Ausbildung noch technisches Interesse vorhanden sind, wird angenommen, dass auch Kenntnisse in diesem Umfeld nur spärlich vorliegen. Diese Pauschalisierung hilft, eine objektive Einschätzung der technischen Kenntnisse der Teilnehmer zu ermöglichen. In diesem Fall besteht eine verhältnismäßi-

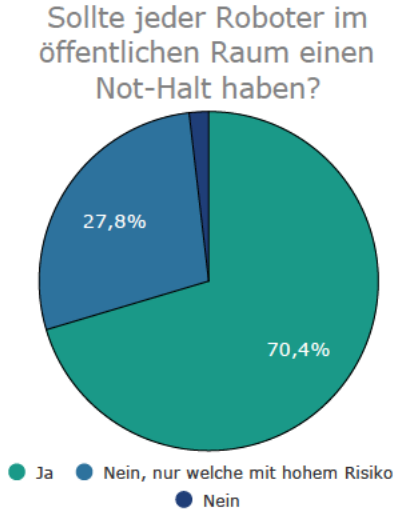
ge gute Verteilung. Hierdurch lässt sich untersuchen, ob weniger erfahrende Passanten anders reagieren als Passanten mit mehr technischer Erfahrung.

Außerdem wussten die Teilnehmer zum Zeitpunkt der Umfrage nicht, dass Sprachsteuerung im speziellen Fokus der Umfrage stand. Auch in den vorgegebenen Antwortmöglichkeiten wurden neben Sprachsteuerung stets andere Mittel und auch Freitextantworten angeboten.

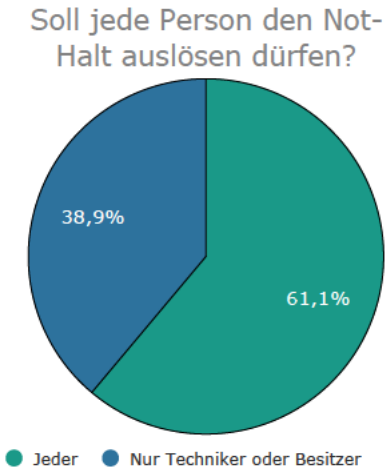
2.1.3 Ergebnisse

Notwendigkeit des Not-Halts

Um die Erwartungshaltung der Passanten gegenüber AMS zu prüfen, wurde abgefragt, ob sie den Not-Halt als essenziellen Bestandteil eines solchen ansehen. Dabei haben sich 98,2 % dafür ausgesprochen, sobald das AMS ein potenzielles Sicherheitsrisiko hat. Ein Großteil davon, 70,4 %, erwartet sogar bei jedem AMS eine Not-Halt-Funktionalität (ungeachtet des Auslösers) (Abbildung 2.1a).
Entgegen den Erwartungen sind allerdings nur 61,1 % der Befragten der Auffassung, dass jede Person das Recht auf die Auslösung des Not-Halts haben sollte (Abbildung 2.1b).



(a) Notwendigkeit einer Not-Halt-Funktionalität



(b) Nutzer des Not-Halts

Abb. 2.1: Erwartungen der Passanten an das AMS in Bezug auf Not-Halt

Sinnvolle Auslöser für einen Not-Halt

Bei der Frage nach intuitiven und sinnvollen Auslösern für die Not-Halt-Funktion konnten die Befragten mehrere Antworten geben. Dabei halten nur knapp über die Hälfte der Befragten die Sprachsteuerung für ein sinnvolles Werkzeug für diese Aufgabe (Blauer Balken, Abbildung 2.2).

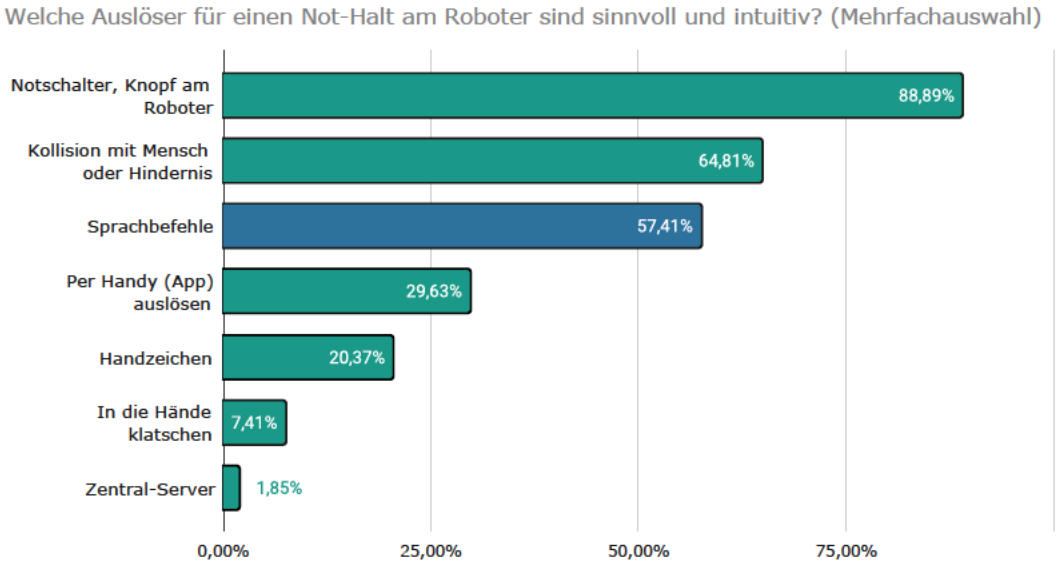


Abb. 2.2: Verschiedene Not-Halt-Auslöser, Frage aus Unterabschnitt A.4.3, Mehrfachauswahl war möglich.

Überzeugend war vor allem der klassische Notschalter, dieser ist für die allermeisten Passanten am naheliegendsten. Das ist nicht überraschend, denn dieser ist stark reguliert und genauso weit verbreitet.
Über 60 % sehen eine Kollision mit dem AMS für einen sinnvollen Auslöser eines Not-Halts. Dass das AMS bei einer Kollisionserkennung reagiert, anhält oder auch zurückfährt, ist nur sinnvoll. Allerdings ist dies nicht für die Steuerung eines Not-Aus oder Not-Halt geeignet, deswegen wird darauf in dieser Arbeit nicht mehr eingegangen. Die Begründung dafür ist, dass die Steuerung des Not-Halts in der Hand des Bedieners liegen muss, bei der Kollisionserkennung ist das nicht der Fall.
Die weiteren Alternativen: Handzeichen, App-Steuerung oder akustische Signale wie in-die-Hände-klatschen fallen im Vergleich allerdings weit zurück und werden infolgedessen auch nicht weiter betrachtet.

Reaktionsfragen

Ein weiterer wesentlicher Bestandteil der Umfrage waren die Reaktionsfragen zu anschaulichen Beispielsituationen (Abbildung A.1, Abbildung A.2). Diese sind von der vorherigen Frage über die Sinnhaftigkeit unterschiedlicher Auslöser abzugrenzen, denn hier geht es in den konkreten Beispielszenarien um die Praxistauglichkeit. Eine andere Verteilung war zu erwarten. In diesem Fall war keine Mehrfachauswahl möglich und alle Anteile werden zu 100 % zusammengerechnet. Außerdem wurden die Reaktionsergebnisse aus beiden Reaktionsfragen kombiniert, um eine größere Antwortmenge zu erzielen.

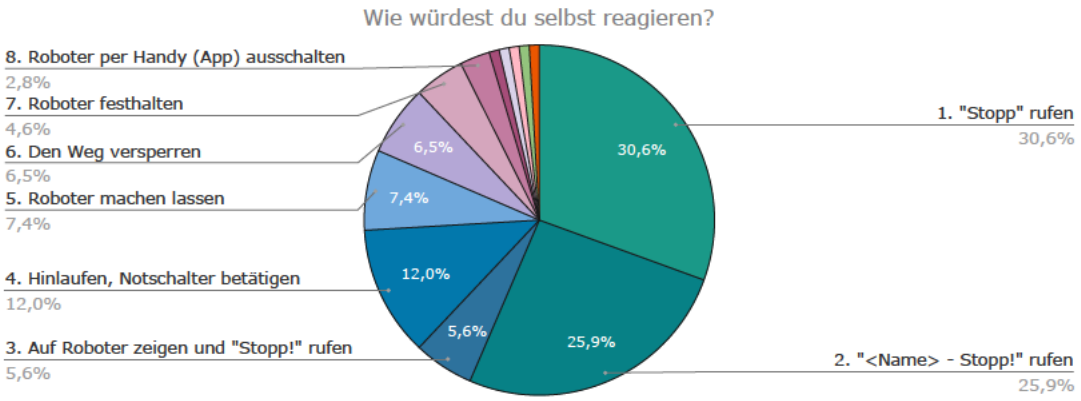


Abb. 2.3: Reaktionen der Befragten

Es ist zu erkennen, dass hier wiederum die sprachbasierten Optionen zusammengefasst mit über 60 % dominieren.
Den Notschalter am aktiven AMS zu betätigen trauen sich nur 12 % zu. Als Ursache dafür ist die Distanz zum AMS denkbar, aber auch die Tatsache, dass das AMS sich zusätzlich durchgehend bewegt.

Unter den unterschiedlichen sprachbasierten Optionen wurde der einfache Stopp-Befehl am häufigsten verwendet. Diese Option hat allerdings das Problem, dass keine Adressierung eines konkreten AMS möglich ist, denn alle sich in der Umgebung befindlichen AMS würden gleich agieren. Die beiden weiteren Sprachbefehle nutzen eine direkte Adressierung, in einem Fall wurde optisch auf das AMS gezeigt, im zweiten Fall wurde beim Sprachbefehl der Name des AMS vom Nummernschild abgelesen. Die Adressierung über den Namen wurde deutlich bevorzugt.

Die Sprachbefehle wurden in diesen Situationen deutlich häufiger verwendet als die möglichen Alternativen, das bedeutet allerdings nicht, dass Sprachbefehle intuitiv sind. Um dies zu prüfen, wird eine Teilmenge der Befragten isoliert betrachtet. Die Befragten mit dem Hintergrund "Beruf oder Ausbildung mit rein technischem Bezug" und "Beruf oder Ausbildung mit technischem Bezug" (Tabelle 2.2) werden aus den Reaktionsfragen herausgefiltert. Übrig bleiben Reaktionen von Befragten ohne technischen Hintergrund. Diese Personen haben erwartungsgemäß weniger technische Kenntnisse und sind deswegen repräsentativer für eine intuitive Handlung.

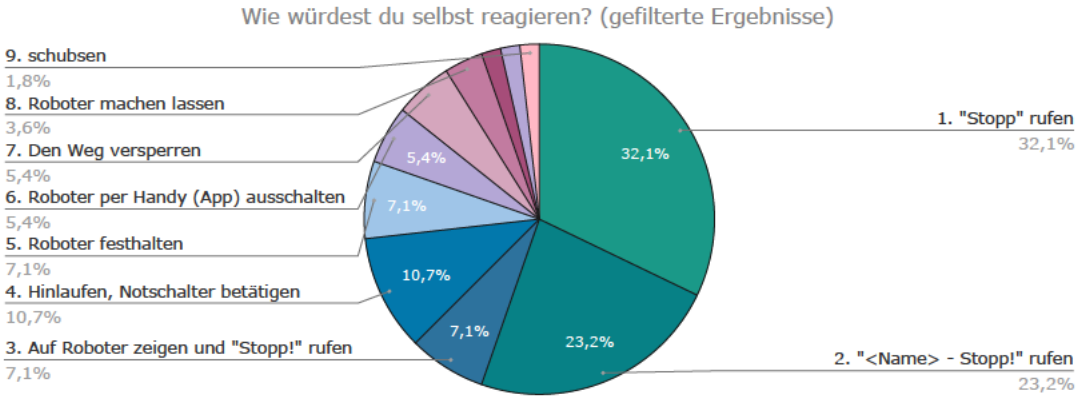


Abb. 2.4: Reaktionen der Befragten, gefiltert: nur fachfremde Teilnehmer

Die Teilmenge der Befragten besteht nur noch aus 28 Personen. Die Reaktionsverteilung ist allerdings ähnlich, die Sprachbefehle dominieren in dieser Teilmenge ebenfalls mit über 60 %. Dies zeigt, dass die Sprachsteuerung ebenfalls intuitiv ist für die Not-Halt-Anwendung.

2.1.4 Implementationsrückschlüsse

Aus den zuvor beschriebenen Ergebnissen lassen sich Rückschlüsse für die Implementation ableiten:

- Sprachbefehle sind sinnvoll und intuitiv im Einsatz für eine Not-Halt-Anwendung
- Adressierung eines AMS erfolgt über den Namen
- Jedes AMS erhält einen individuellen Namen
- Name des AMS ist lesbar in Form eines Kennzeichens vorne und hinten montiert

- Ein Piktogramm auf dem Kennzeichen deutet die Sprachsteuerung an (z. B.: Abbildung 2.5)

Abb. 2.5: Beispielhaftes Piktogramm Sprachsteuerung von Website PNGWING[11]

Sprachbefehle

Aus der Umfrage selektierte Not-Halt-Aktivierungswörter sind:

Stop (*engl.*) – Stopp – Halt – Aus – Abbrechen – Achtung

Kommandos zum Lösen des Not-Halt-Modus könnten folgende sein:

Go (*engl.*) – Weiter – Freigeben

Diese Sprachkommandos adressieren in Kombination mit dem Namen des AMS ein konkretes System wie folgt:

«Name» «Sprachkommando»

2.2 Spracherkennung

Die Kernaufgabe wird sein, in einem begrenzten Umfeld des AMS das Audio auszuwerten und auf die definierten Wörter (Unterunterabschnitt 2.1.4) zu reagieren. Um das Erkennen der Signalwörter in einer Notsituation zu ermöglichen, muss das Audio kontinuierlich und in Echtzeit ausgewertet werden. Es wird also eine schnelle und zuverlässige Spracherkennung benötigt.

Für das Erkennen von gesprochener Sprache und Wörtern gibt es unterschiedlichste Methoden. Eine aktuelle und führende Variante funktioniert mithilfe von Künstlicher Intelligenz (KI). Dabei wird ein neuronales Netz auf großen, gelabelten Datensätzen trainiert, um genau diese trainierten Wörter im neuen Kontext zu detektieren. Diese Methode steht im Fokus der vorliegenden Arbeit.

2.2.1 KI Modellarchitekturen

Da es sich bei gesprochener Sprache um eine Sequenz von Silben handelt, die aufgezeichneten Daten in Form von Datenreihen vorliegen und diese im zeitlichen Zusammenhang zueinander stehen, bieten sich einige Architekturen an. Sowohl Recurrent Neural Networks (RNNs), Long short-term memory (LSTM) und Time Delay Neural Networks (TDNNs) arbeiten grundlegend auf Datenreihen. Beim Training und der Inferenz wird nicht nur ein Zeitausschnitt isoliert betrachtet, sondern es werden zusätzlich die bereits vergangenen Daten bei der Berechnung eingebunden. Somit können Zusammenhänge auf der Zeitebene erkannt werden. Beliebte Anwendungsfälle für diese Architekturen sind Texte, Videos oder auch Audio.

Auch die Allzweck-Architektur der Convolutional Neural Networks (CNNs) lässt sich im vorliegenden Fall anwenden. Bei einem CNN werden stets nur isolierte Datenausschnitte betrachtet, im Gegensatz zu RNNs kann kein Bezug zu vorangegangenen Daten hergestellt werden. Bei der Anforderung, ein spezifisches Wort zu erkennen, bedarf es allerdings nicht unbedingt eines ‘Gedächtnisses’. Passt das gesprochene Wort in die definierte Größe des Datenausschnittes, so kann dieses ebenfalls erkannt werden, ohne den Kontext – das zuvor oder danach Gesprochene – einzubeziehen. Das Betrachtungsfenster des Audios muss also gezwungenermaßen größer sein als das längste gesprochene Wort, das das Model zu erkennen versucht. Dies muss beim Entwurf der Architektur berücksichtigt werden.

Eine Alternative zum CNN ist die aus dem MobileNet [1] bekannt gewordene Architektur Depthwise Separable Convolutional Neural Network (DS-CNN). Im Vergleich zum CNN werden Operationen der Matrizenmultiplikation um einen Faktor von bis zu 23 eingespart und dennoch wird eine vergleichbare Genauigkeit erreicht [12].

2.2.2 Eignung für Edge-Geräte

Die Ausarbeitung von ARM Limited und der Stanford University unter dem Titel *Hello Edge: Keyword Spotting on Microcontrollers* [3] aus dem Jahr 2018 dokumentiert Untersuchungen zur Genauigkeit von unterschiedlichen Architekturen auf ARM MCUs unterschiedlicher Größe. Dabei sind die Bedingungen stets limitierter Arbeitsspeicher und limitierte Rechenleistung. Tests werden auf MCUs der Größe S, M und L durchgeführt (Tabelle B.1). Der Umfang des KI-Modells wurde dabei durch Reduktion der Modellparameter an die Größe der jeweiligen MCU angepasst.

Ziel dieser Untersuchung war es, die Architektur zu finden, die im Anwendungsfall des Keyword Spotting *(ger.: Erkennung von Schlüsselbegriffen)* (KWS) mit reduzierter Hardware Echtzeitanforderungen am besten erfüllen kann. Dabei wurde die Genauigkeit der Interpretation am *Speech Commands Dataset* [2] von Google bemessen.

Das Team belegte, dass ein DS-CNN gut skaliert und unter stark reduzierter Hardware die besten Ergebnisse bot. Diese Architektur ließ sich um den Faktor 10 auf 40 KB Arbeitsspeicher und 5,4 Mil. Operationen reduzieren mit einem Verlust von nur 1 % an Genauigkeit (Tabelle B.2, Abbildung B.1). Die alternativen Architekturen aus dem Bericht schnitten unter vergleichbaren Hardwarebedingungen schlechter ab.

Die Resultate aus diesem Bericht untermauern die Tauglichkeit des DS-CNN für eine echtzeitfähige Not-Halt-Detektion-Anwendung auf einem Edge-Gerät. Für die Umsetzung der Not-Halt-Detektion in diesem Projekt wird die von ARM Limited auf GitHub [13] bereitgestellte DS-CNN-Architektur verwendet.

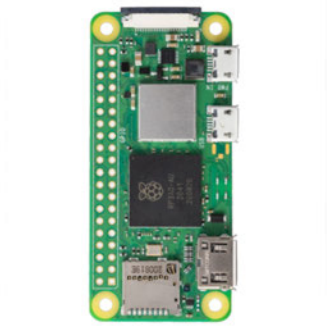
3 Umsetzung

3.1 Hardware

Raspberry Pi Zero 2W: Als Hauptcomputer des Not-Halt-Erkennungssystems wurde der *Raspberry Pi Zero 2W* ausgewählt (Abbildung 3.1a), folgend nur Pi genannt. Die kompakte Bauweise ist das entscheidende Argument für diesen Mikrocomputer. Das verbreitete Format (65 mm x 30 mm) des Boards und die Lochungen zur Arretierung ermöglichen eine kompakte Bauweise des Gesamtsystems.
Im Gegensatz zum Vormodell *Pi Zero W* ist der *Pi Zero 2W* mit einem ARM Cortex-A53 ausgestattet. Dieser Prozessor implementiert die Armv8-A-Architektur mit 64-Bit-Unterstützung und ist deshalb für ein großes Spektrum an Softwareanwendungen geeignet [14]. Die vergleichsweise hohe Leistung des Prozessors (4×1 GHz) eröffnet viel Spielraum bei den laufenden Entwicklungen, da die Leistungsanforderungen zum Zeitpunkt der Hardwareauswahl nicht komplett abschätzbar sind.

Coral USB Accelerator: Um eine echtzeitfähige Inferenz zu ermöglichen, wird eine TPU (Abbildung 3.1b) in das System eingebunden. Die Rechenoperationen des neuronalen Netzwerks werden von der CPU des Pi auf die architektonisch spezialisierte Recheneinheit des Coral Sticks ausgelagert. Dies stellt sicher, dass auch bei größeren Netzwerken die Inferenzgeschwindigkeit stabil bleibt. Der Coral Stick bietet eine USB-3.0-Schnittstelle und kann über ein USB-Kabel mit dem Pi verbunden werden.
Eine von Google bereitgestellte *Edge TPU Runtime* implementiert die Kommunikation zwischen der Anwendung auf dem Host-Gerät und der TPU. Mindestanforderung für die Runtime ist eine Armv7-Systemarchitektur [15], mit dem ausgewählten Modell *Pi Zero 2W* ist diese kompatibel, mit dem Vormodell *Pi Zero W* von Raspberry allerdings nicht mehr.

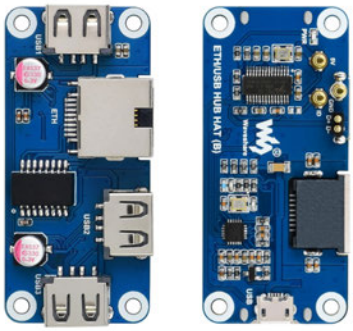
ReSpeaker Mic Array: Für die Erfassung des Audios aus der Umgebung wird das Mikrofon-Array *Respeaker USB Mic Array* (Abbildung 3.1c) von Seeed Studio verwendet. Das Mikrofon-Array ist ein integriertes System mit vier Fernfeldmikrofonen und einem Mikrocontroller. Die Fernfeldmikrofone haben eine Reichweite bis zu fünf Metern in alle Richtungen. Der integrierte Chip implementiert eine Audiovorverarbeitung mit dem Fokus auf der menschlichen Stimme, Hintergrundgeräusche werden herausgefiltert. Ein LED-Ring auf der Oberseite lässt sich programmieren, um dem Nutzer einen Systemstatus zu vermitteln. Folgend wird das ReSpeaker Mic Array nur Mikrofoneinheit genannt.



(a) Raspberry Pi Zero 2W [16]

(b) Coral USB Accelerator, Edge Tensor Processing Unit (TPU) [17]

(c) ReSpeaker Mic Array [18]



(d) Waveshare ETH/USB HUB HAT (B) [19]

Abb. 3.1: Genutzte Hardware für das Not-Halt Erkennungssystem

Waveshare ETH/USB HUB HAT (B): Um alle Peripheriegeräte mit dem Pi zu verbinden, wird ein Extension Hub (*Waveshare ETH/USB HUB HAT (B)*) verwendet (Abbildung 3.1d).

3.1.1 Layout

Der Extension Hub wird ohne Löten unter den Pi geschraubt. Stromversorgung und USB-Verbindung werden über die Pogo-Pins des Hubs und die Kontaktflächen des Pi's realisiert. Das Extension Board hat drei USB 2.0 Anschlüsse und einen Ethernet Anschluss. Die Mikrofoneinheit und der Coral USB Accelerator werden über USB angeschlossen. Der Coral USB Accelerator kann zwar für schnelle Übertragung über USB 3.0 angebunden werden, der Pi und auch das Extension Board unterstützen allerdings nur USB 2.0.

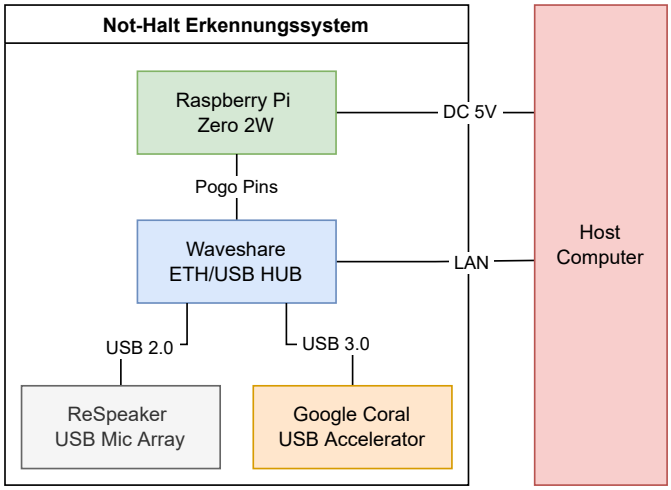


Abb. 3.2: Hardwarelayout und -schnittstellen der genutzten Komponenten

Es gibt zwei Verbindungen zwischen dem Not-Halt-Erkennungssystem und dem Host-Computer. Eine USB-Verbindung für die 5V-Gleichstromversorgung und eine Ethernet-Verbindung, um Steuerbefehle vom Erkennungssystem zu empfangen.

3.2 Softwareimplementation

3.2.1 Machbarkeitsstudie: *Google Speech Command Dataset*

Allgemeine Informationen

Das Speech Command Dataset [2] von Google ist ein weit verbreiteter Datensatz mit ausgewählten Sprachbefehlen. Dieser besteht aus sortierten und gelabelten Audiodateien für 35 verschiedene Kommandos, die von natürlichen, englischsprachigen Personen aufgezeichnet worden sind. Alle Audiodateien liegen im 16000 Hz WAV-Format vor und sind auf 1000 ms Gesamtlänge zugeschnitten. Ein Befehl wird jeweils von ca. 2000 bis 4000 verschiedenen Audiodateien repräsentiert.

Der Datensatz beinhaltet folgende Kernbefehle:

Yes – No – Up – Down – Left – Right – On – Off – Stop – Go – [Zero to Nine]

Zusätzlich zu den Kernbefehlen sind auch noch einige weniger repräsentative Befehle vorhanden.

Der Datensatz wird vor allem in Forschungsprojekten verwendet, da er sich im Umfeld des maschinellen Lernens für KWS als De-facto-Standard etabliert hat und als Benchmark zum Vergleich verschiedener Architekturen hinsichtlich ihrer Leistungsfähigkeit dient.

DS-CNN im Benchmark

Auf der Plattform *PapersWithCode* werden für den Datensatz laufend aktuelle Forschungsergebnisse zu neuen Architekturen gelistet und nach Treffsicherheit gewertet [20].

Das DS-CNN von ARM Limited [3] hat mit 94,4 % gut abgeschnitten und liegt auf Platz 13. Zur Veranschaulichung: Die aktuell beste veröffentliche und bemessene Architektur ist die *TrippleLoss-res15* [21] mit 98,5 % Treffsicherheit (Stand: 11.06.2022) [20].

Aufgrund der bereits erläuterten Skalierbarkeit des DS-CNN (Unterabschnitt 2.2.2) wird die schlechtere Treffsicherheit als Kompromiss akzeptiert.

Durchführung

Um evaluieren zu können, ob die ausgewählte Hardware für die Softwarekomponenten geeignet ist, wurde eine experimentelle Machbarkeitsstudie durchgeführt. Der Hardwareaufbau erfolgte nach dem geplanten Layout (Abbildung 3.2).

Modell-Vorverarbeitungskette:

Abb. 3.3: Dreischrittige Datenverarbeitungspipeline für das Generieren des gewünschten Modells mit Ein- und Ausgängen der jeweiligen Daten

1. Das Training wird durch das `/train.py` Script umgesetzt
   - Ausgabe: trainiertes Model liegt vor (Abbildung C.1)
2. Konvertierung und Quantisierung werden in dem Script `/convert.py` umgesetzt
   - Ausgabe: kompaktes *.tflite*-Modell
3. Kompilierung für die Coral Edge TPU (Abbildung 3.1b) werden von der bereitgestellten Anwendung `edgetpu-compiler` umgesetzt
   - Ausgabe: TPU-fähiges Modellkompilat, 16 von 16 Operationen können auf die TPU ausgelagert werden

Das Training des DS-CNN in Größenvariante L (Tabelle B.1) wurde durchgeführt. Nach dem Training liegt das Modell in einem Checkpoint-Format [22] vor. Anschließende Tests belegen eine Genauigkeit von 90,14 % (Abbildung C.2). Das Modell hat in diesem Zustand eine Größe von ca. 420.000 Parametern und ist 5 MB groß.

Die Konvertierung ins *.tflite*-Format ist obligatorisch für die Weiterverarbeitung. Durch die Quantisierung ist eine Reduktion der Genauigkeit naturgemäß. Nach der Konvertierung werden die Tests wiederholt, es ergibt sich eine minimal schlechtere Inferenzgenauigkeit von 90,04 % (-0,1 %) (Abbildung C.3). Das Einsparungspotenzial ist hingegen enorm, der Speicherbedarf des Modells wurde auf 492 KB (-90 %) reduziert.

Die Kompilierung für die TPU hat keinen Einfluss auf die Qualitäten des Modells.

3.2.2 Trainingsanwendung: *kwsCoach*

Vision

Die Grundvoraussetzung für das Training von neuronalen Netzen sind Daten. Im vorliegenden Fall werden Sprachaufnahmen von den spezifischen Steuerbefehlen (Unterunterabschnitt 2.1.4) benötigt. Letztere müssen manuell aufbereitet, sortiert und gelabelt werden. Dies erfordert einen hohen Aufwand, der nicht nur initial betrieben werden muss, sondern auch bei jeder weiteren Ergänzung des neuronalen Modells. Fertige Datensätze wie das Google Speech Command Dataset [2] sind nicht nur eine Seltenheit, sondern auch ausschließlich in englischer Sprache erhältlich. Für die bestehenden Anforderungen sind solche Datensätze keine Option.

Die hier bevorzugte Option ist die Synthese von Audiodateien durch Text-to-Speech (TTS). Wird die Datenbasis des Modells auf rein synthetische Daten reduziert, so lässt sich dadurch eine Automatisierung umsetzen, die eine hohe Zeitersparnis bietet. Die Daten ließen sich auf Abruf synthetisieren, labeln und variieren und auch das Training wäre automatisiert möglich.

Ein weiterer Vorteil, der sich aus der Automatisierung ergibt, ist, dass Komplexität und notwendige Fachkenntnisse reduziert werden, wodurch diese Technologie einer größeren Personengruppe zugänglich gemacht wird.

Zusammengefasst besteht die Vision darin, dass zukünftige neue Sprachbefehle oder auch Namen (der jeweiligen AMS) auf Knopfdruck in einer Anwendung trainiert werden können. Am Ende des Prozesses entsteht ein neues neuronales Modell, das ausgerollt werden kann.

Funktionale Anforderungen

Die Anwendung *kwsCoach* bietet mit einer grafischen Benutzeroberfläche eine intuitive Übersicht über alle Funktionsmöglichkeiten. Der Funktionsumfang lässt sich in drei Kerngruppen unterteilen: Synthese, Data Augmentation und Training.

Synthese von Audiodateien

- Integration von mehreren TTS Lösungen
- Gebündelte Übersicht von möglichen Stimmen und Sprachen
- Ausprobieren und Anhören von den TTS-Stimmen
- Eingabe von Wörtern, die für den Datensatz synthetisiert werden sollen
- Synthese von zufälligen Wörtern mittels eines Wörterbuches
- Ausgabeformat: WAV-Datei, 16 kHz, 1-Channel
- Datensatzstruktur mithilfe von Ordnern
  Wortklasse: `directory/word/*.wav`
  Klasse für unbekannte Wörter: `directory/_unknown_/*.wav`
- Übersicht über den Datensatz (Anzahl, Länge des Audios)

Data Augmentation

- Trimmen von Stille aus Audiodateien
- Vereinheitlichung der Audiolänge
- Vervielfachung von synthetisierten Dateien nach zufälligen Parametern:
  - Veränderung der Tonhöhe
  - Verschiebung auf Zeitachse
  - Lautstärke
  - Sprechgeschwindigkeit
- Hinzufügen von Hintergrundgeräuschen

Training des Modells

- Modellarchitektur Auswahl
- Auswahl unterschiedlicher Modellgrößen
- Automatisiertes Training
- Ausgabeformat: TensorFlow Model Checkpoint [22]

TTS-Lösungen

Text-to-Speech (TTS) Technologie ist neu und es gibt viele Lösungen auf dem Markt. Damit diese in kwsCoach integriert werden können, ist eine technische Schnittstelle erforderlich. Das bedeutet, dass Lösungen mit einem User-Interface (z.B.: web-basierte Anwendungen) nicht infrage kommen.

Open-Source CLI Anwendungen wurden als Erstes geprüft. Diese sind kostenfrei und vor allem offline ohne Abrechnungskonto nutzbar. Geprüft wurden folgende Anwendungen: **eSpeak** [23], **Gespeaker** [24], **Festival** [25] und **pico** (nicht Open-Source) [26]. Teile der synthetisierten Resultate waren gut, andere wiederum schlecht. Das größte gemeinsame Problem dieser Anwendungen war allerdings, der stark limitierte Umfang der Sprachen und Stimmen. Für die Integration waren diese Anwendungen nicht lohnenswert oder wegen der schlechten Sprachqualität sogar ungeeignet.

Weiterhin wurden TTS Online Services geprüft. Vorrangig lag der Fokus auf den bekanntesten Online Service Providern: **Google Cloud TTS** [27], **Microsoft's Azure TTS** aus den Cognitive Services [28] und **IBM's Watson TTS** [29]. Alle drei Provider bieten ihre TTS Services in einem kostenlosen Abrechnungsmodell an, auch wenn ein Zahlungsmittel hinterlegt werden muss. Das kostenlose Abrechnungsmodell deckt die zuvor angeführten Zwecke ab. Die angebotene Vielfalt an Sprachen und Stimmen ist sehr groß (Tabelle 3.1).

Google und IBM lassen sich über ein SDK einfach integrieren. Dieses abstrahiert von der Netzwerkkommunikation mit den Online Services. Microsoft hat ebenfalls ein SDK, das allerdings nicht alle notwendigen Funktionen anbietet, aber auch über eine REST-API integriert werden kann.

Provider	Anzahl	Männlich	Weiblich	Deutsch[a]	Englisch[b]
Google Cloud	314 (100 %)	138 (44 %)	176 (56 %)	12 (4 %)	46 (15 %)
Microsoft Azure	340 (100 %)	157 (46 %)	183 (54 %)	19 (6 %)	60 (18 %)
IBM Watson	60 (100 %)	20 (33 %)	40 (66 %)	7 (12 %)	20 (33 %)
Gesamt	714 (100 %)	315 (44 %)	399 (56 %)	38 (5 %)	126 (18 %)

Tabelle 3.1: Stimm- und Sprachvielfalt der ausgewählten TTS Provider

[a]Inklusive: de-DE, de-AT und de-CH
[b]Inklusive: en-AU, en-CA, en-GB, en-HK, en-IE, en-IN, en-KE, en-NG, en-NZ, en-PH, en-SG, en-TZ, en-US und en-ZA

3.2.3 Embedded Not-Halt-Kontroller

Vision

Diese Echtzeitanwendung wird auf dem Raspberry Pi betrieben. Unter Zuhilfenahme des trainierten Modells sollen die Sprachbefehle erkannt und folglich die Stopp- und Startprozesse initiiert werden. Idealerweise gibt es auch ein optisches Statusfeedback für den Benutzer.

Anforderungen

Mit Unterstützung des Use-Cases Diagramms (Abbildung 3.4) werden die Funktionen bildhaft dargestellt. Dabei wird unterschieden zwischen den Anwendungsfällen für normale Passanten und Techniker des jeweiligen AMS, wobei der Techniker auch alle Anwendungsfälle des Passanten nutzen kann.

Aus diesem Diagramm werden Use-Cases (Tabelle 3.2) formuliert und durchnummeriert. Das Schema der Nummerierung ist “UC-#”, z. B. “UC-1”.

Mithilfe der Use-Cases werden funktionale und nichtfunktionale Anforderungen (Tabelle 3.3) für den Not-Halt-Kontroller formuliert und nach dem Schema “REQ-#” beziffert.

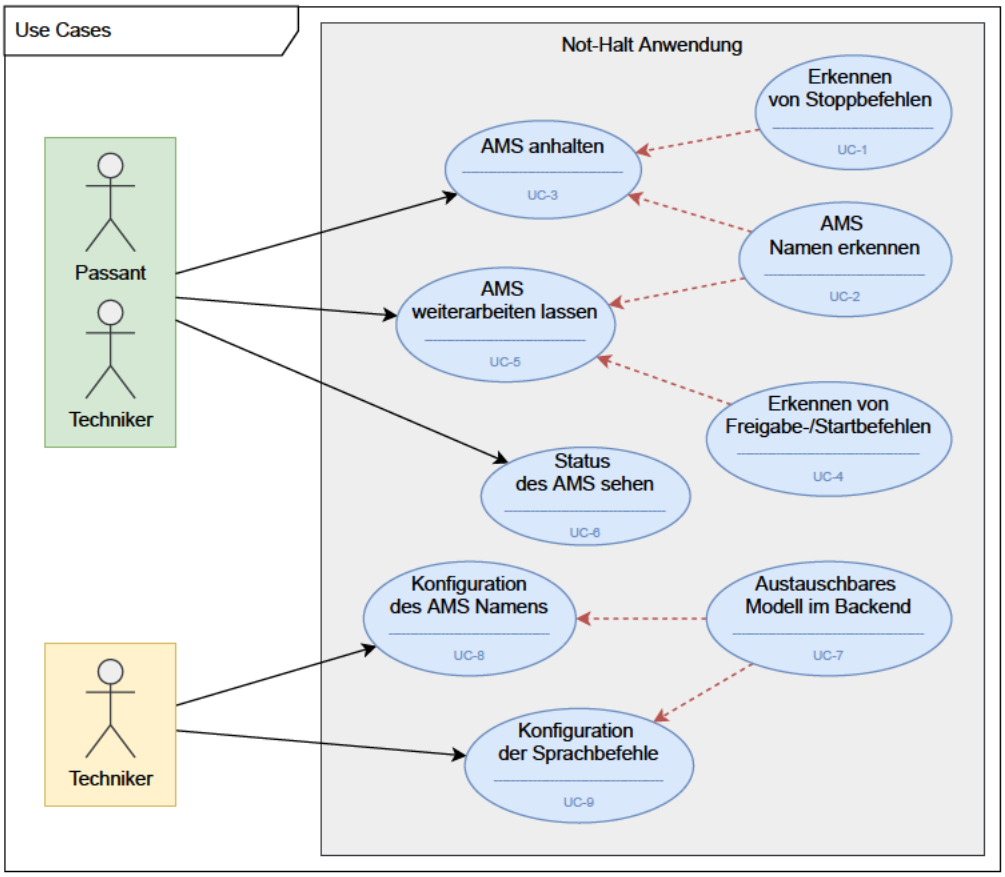


Abb. 3.4: Use-Cases-Diagramm des Not-Halt-Kontroller

Tabelle 3.2: Use-Cases der Not-Halt-Kontroller, UC-1 bis UC-9 (oben)

UC-1	**Erkennen von Stoppbefehlen**
Abhängigkeiten	-
Beschreibung	Der Nutzer spricht einen Sprachbefehl der Kategorie «Stopp» (Unterunterabschnitt 2.1.4). Das AMS erkennt den Befehl.
Akteur	Passant oder Techniker
Vorbedingung	Der Nutzer befindet sich in Hörreichweite des AMS.
Nachbedingung	Das AMS hat den Sprachbefehl der Kategorie «Stopp» (Unterunterabschnitt 2.1.4) erkannt.

UC-2	**AMS Namen erkennen**
Abhängigkeiten	-
Beschreibung	Der Nutzer spricht des spezifizierten Namen des AMS. Der Name wird vom AMS erkannt.
Akteur	Passant oder Techniker
Vorbedingung	Der Nutzer befindet sich in Hörreichweite des AMS.
Nachbedingung	Das AMS hat den Namen erkannt. Das AMS ist bereit für einen Folgebefehl.

UC-3	**AMS anhalten**
Abhängigkeiten	UC-1, UC-2
Beschreibung	Der Nutzer spricht einen Sprachbefehl der Kategorie «Stopp» (Unterunterabschnitt 2.1.4). Das AMS erkennt den Befehl und reagiert.
Akteur	Passant oder Techniker
Vorbedingung	Der Nutzer befindet sich in Hörreichweite des AMS. Der Nutzer hat zuvor das AMS namentlich angesprochen.
Nachbedingung	Das AMS signalisiert optisch, dass der Not-Halt Modus eingeleitet wurde. Der Not-Halt wurde umgesetzt.

UC-4	**Erkennen von Freigabe-/Startbefehlen**
Abhängigkeiten	-
Beschreibung	Der Nutzer spricht einen Sprachbefehl der Kategorie «Weiter» (Unterunterabschnitt 2.1.4) aus. Das AMS erkennt den Befehl.
Akteur	Passant oder Techniker
Vorbedingung	Der Nutzer befindet sich in Hörreichweite des AMS.
Nachbedingung	Das AMS hat den Sprachbefehl der Kategorie «Weiter» (Unterunterabschnitt 2.1.4) erkannt.

UC-5	**AMS weiterarbeiten lassen**
Abhängigkeiten	UC-2, UC-4
Beschreibung	Der Nutzer spricht einen Sprachbefehl der Kategorie «Weiter» (Unterunterabschnitt 2.1.4) aus. Das AMS erkennt den Befehl und verlässt den Not-Halt Modus.
Akteur	Passant oder Techniker
Vorbedingung	Der Nutzer befindet sich in Hörreichweite des AMS. Der Nutzer hat zuvor das AMS namentlich angesprochen.
Nachbedingung	Das AMS signalisiert optisch, dass der Not-Halt Modus verlassen wurde. Der Not-Halt wurde beendet.

UC-6	**Status des AMS sehen**
Abhängigkeiten	UC-1, UC-2, UC-4
Beschreibung	Ein optisches Signal gibt den aktuellen Not-Halt Status des AMS wieder. Dieser ist für den Nutzer sichtbar und interpretierbar.
Akteur	Passant oder Techniker
Vorbedingung	-
Nachbedingung	Hat das AMS seinen Namen erkannt, so leuchtet eine LED kurz auf. Befindet sich das AMS im Not-Halt-Modus, so leuchtet eine LED rot-pulsierend. Befindet sich das AMS **nicht** im Not-Halt-Modus, so leuchtet eine LED statisch-grün.

UC-7	**Austauschbares Modell im Backend**
Abhängigkeiten	-
Beschreibung	Während der Inbetriebnahme des Not-Halt-Kontrollers kann ein neuronales Netz/Modell konfiguriert werden, das ggf. andere oder zusätzliche Wörter erkennen kann.
Akteur	Techniker
Vorbedingung	-
Nachbedingung	Der Not-Halt-Kontroller integriert dynamisch ein neues Modell, wenn dieses korrekt konfiguriert wurde, und arbeitet entsprechend mit den neuen Sprachbefehlen.

UC-8	**Konfiguration des AMS-Namens**
Abhängigkeiten	UC-2, UC-7
Beschreibung	Während der Inbetriebnahme des Not-Halt-Kontrollers kann der Nutzer festlegen, welcher Sprachbefehl des bereitgestellten Modells als Name für das AMS fungiert.
Akteur	Techniker
Vorbedingung	-
Nachbedingung	Der Not-Halt-Kontroller reagiert auf den neu-konfigurierten Namen für das AMS.

UC-9	**Konfiguration der Sprachbefehle**
Abhängigkeiten	UC-1, UC-4, UC-7
Beschreibung	Während der Inbetriebnahme des Not-Halt-Kontrollers kann der Nutzer festlegen, welche Sprachbefehle des Modells zur Kategorie «Stopp» und welche Sprachbefehle der Kategorie «Weiter» zugehören.
Akteur	Techniker
Vorbedingung	-
Nachbedingung	Der Not-Halt-Kontroller reagiert auf die neukonfigurierten «Stopp»-, «Weiter»-Sprachbefehle. Wie bereits in UC-1 und UC-4 beschrieben.

Tabelle 3.2: Use-Cases des Not-Halt-Kontrollers, UC-1 bis UC-9 (unten)

Tabelle 3.3: Anforderungen des Not-Halt-Kontrollers, REQ-1 bis REQ-9 (oben)

REQ-1	**Not-Halt-Kontroller erkennt Stopp-Sprachbefehle**
Typ	Funktional
Use-Cases	UC-1, UC-7, UC-9
Beschreibung	In Hörweite des AMS erkennt der Not-Halt-Kontroller gesprochene, spezifizierte Wörter der Klasse «Stopp».
Voraussetzung	Die spezifizierten Wörter wurden beim Training des Modells berücksichtigt. Die spezifizierten Wörter wurden während der Inbetriebnahme des Not-Halt-Kontrollers als Wörter der Klasse «Stopp» konfiguriert.

REQ-2	**Not-Halt-Kontroller erkennt Namen des AMS**
Typ	Funktional
Use-Cases	UC-2, UC-7, UC-8
Beschreibung	In Hörweite des AMS erkennt der Not-Halt-Kontroller den gesprochenen, spezifizierten Namen des jeweiligen AMS.
Voraussetzung	Der spezifizierte Name wurde beim Training des Modells berücksichtigt. Der spezifizierte Name wurde während der Inbetriebnahme des Not-Halt-Kontrollers als Name des AMS konfiguriert.

REQ-3	**Not-Halt-Kontroller erkennt Freigabe-Sprachbefehle**
Typ	Funktional
Use-Cases	UC-4, UC-7, UC-9
Beschreibung	In Hörweite des AMS erkennt der Not-Halt-Kontroller gesprochene, spezifizierte Wörter der Klasse «Weiter».
Voraussetzung	Die spezifizierten Wörter wurden beim Training des Modells berücksichtigt. Die spezifizierten Wörter wurden während der Inbetriebnahme des Not-Halt-Kontrollers als Wörter der Klasse «Weiter» konfiguriert.

REQ-4	**Not-Halt-Kontroller initiiert den Not-Halt Modus**
Typ	Funktional
Use-Cases	UC-3
Beschreibung	• Werden der Name des AMS und darauffolgend ein Stopp-Befehl gesprochen, so wird der Not-Halt-Modus initiiert. • Ein elektronisches Signal wird an den Host-Computer des AMS übermittelt, um den Not-Halt des AMS umzusetzen.
Voraussetzung	Not-Halt-Kontroller **befindet sich nicht** bereits im Not-Halt-Modus.

REQ-5	**Not-Halt-Kontroller beendet den Not-Halt Modus**
Typ	Funktional
Use-Cases	UC-5
Beschreibung	• Werden der Name des AMS und darauffolgend ein Weiter-Befehl gesprochen, so wird der Not-Halt-Modus verlassen. • Ein elektronisches Signal wird an den Host-Computer des AMS übermittelt, um den Not-Halt des AMS zu beenden.
Voraussetzung	Not-Halt-Kontroller **befindet sich** im Not-Halt-Modus.

REQ-6	**Not-Halt-Kontroller gibt dem Nutzer optisches Feedback über LED**
Typ	Funktional
Use-Cases	UC-6
Beschreibung	• Wird der Name des AMS verstanden, so leuchtet eine LED für eine konfigurierte Zeit auf. • Befindet sich der Not-Halt-Kontroller im Not-Halt-Modus, so blinkt eine LED rot auf. • Befindet sich der Not-Halt-Kontroller **nicht** im Not-Halt-Modus, so leuchtet eine LED grün.
Voraussetzung	-

REQ-7	**NN-Modell und die enthaltenen Stopp- und Freigabe-Befehle lassen sich im Not-Halt-Kontroller konfigurieren**
Typ	Funktional
Use-Cases	UC-7, UC-8, UC-9
Beschreibung	• Ein NN-Modell im Format **.tflite*, kompiliert für die Coral Edge TPU, lässt sich in einer Konfigurationsdatei austauschen. • Zuordnung der im Modell enthaltenen Wörter zur Befehlsklasse «Stopp». • Zuordnung der im Modell enthaltenen Wörter zur Befehlsklasse «Weiter». • Zuordnung eines der im Modell enthaltenen Wörter als Name des AMS.
Voraussetzung	Nutzer hat Schreibzugriff auf die SD-Karte des Raspberry Pi's. Nutzer kann die YAML-Datei: `~/e-stop-on-coral/config.yaml` lesen und bearbeiten.

REQ-8	**Datenverarbeitungspipeline entspricht harten Echtzeitanforderungen**
Typ	Nichtfunktional
Use-Cases	UC-3, UC-4
Beschreibung	Die gesamte Datenverarbeitungspipeline vom Eingang des Audios, bis zur Ausgabe des Steuersignals darf nie länger als 1000 Millisekunden dauern.
Voraussetzung	Hardware funktioniert ordnungsgemäß.

REQ-9	**Erkennungsreichweite**
Typ	Nichtfunktional
Use-Cases	UC-1, UC-2, UC-4
Beschreibung	• In Innenräumen kann der Not-Halt-Kontroller Befehle erkennen, die aus einer Entfernung von bis zu fünf Metern gesprochen wurden. • Außerhalb von Innenräumen kann der Not-Halt-Kontroller Befehle erkennen, die aus einer Entfernung von bis zu drei Metern gesprochen wurden.
Voraussetzung	Hardware funktioniert ordnungsgemäß.

Tabelle 3.3: Requirements der Not-Halt-Kontroller, REQ-1 bis REQ-9 (unten)

Softwarearchitektur

Das UML-Klassendiagramm (Abbildung 3.5) verdeutlicht den internen Aufbau des Not-Halt-Kontrollers.

Alle notwendigen Parameter werden aus der YAML-Konfigurationsdatei `~/e-stop-on-coral/config.yaml` vom zentralen Modul `config` ausgelesen und bereitgestellt. Dieses sollte vor der ersten Inbetriebnahme überprüft und gegebenenfalls angepasst werden, da entscheidende Funktionsweisen damit gesteuert werden.

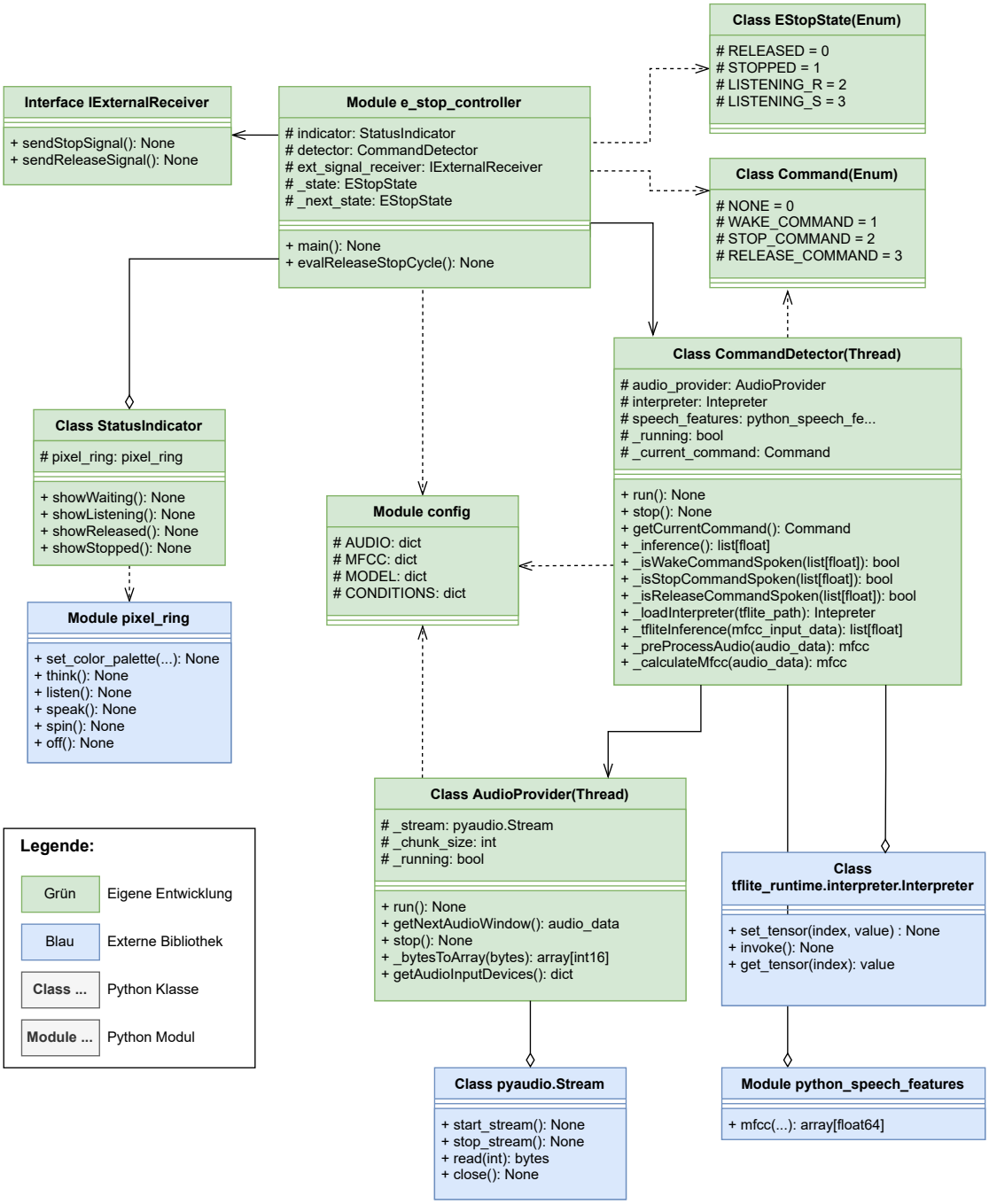


Abb. 3.5: Klassendiagramm des Embedded Not-Halt-Kontrollers

Der Programmeinstieg ist das Modul `e_stop_controller`. Beim Starten initialisiert dieses alle notwendigen Klassen und Module. Der Kontroller implementiert die Entscheidungslogik der Not-Halt-Steuerung. Technisch wird dies mit einem geradlinigen und transparenten endlichen Automaten umgesetzt (Abbildung 3.6).

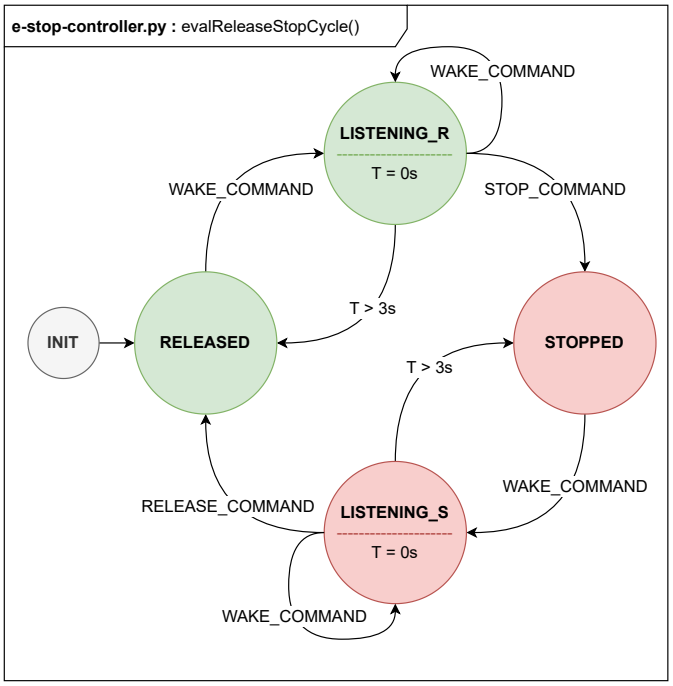


Abb. 3.6: Endlicher Automat als Abbild der Entscheidungslogik

Zwei weitere Kernkomponenten, die für die Verarbeitung des Datenflusses zuständig sind, sind die Klassen `AudioProvider` und `CommandDetector`. Diese sind als Threads implementiert, was ein unabhängiges Prozessieren ermöglicht. Die öffentlichen Methoden sind durch Synchronisation threadsafe.

Der `StatusIndicator` implementiert Zugriffe auf den LED-Ring des ReSpeaker Mikrofonarrays (Abbildung 3.1c) und kann damit die unterschiedlichen Systemstatus dem Nutzer signalisieren.

Eine weitere Signalisierung erfolgt an das Interface `IExternalReceiver`. Hier kann ein externer Signalempfänger integriert werden. Dies ist eingeplant für die Integration mit der AMS-Plattform Husky.

Datenfluss Die Audiodaten werden vom `AudioProvider` Thread aus dem `pyaudio.Stream` im konfigurierbaren Intervall als `bytes` ausgelesen, konvertiert und über die Methode `.getNextAudioWindow()` als `array[int16]` rausgegeben.

Der `CommandDetector` Thread holt sich die Audioblöcke vom `AudioProvider` ab und verarbeitet diese über die Bibliothek `python_speech_features` zu Mel Frequency Cepstral Coefficients (MFCCs). Die MFCCs liegen als zweidimensionales `array[float64]` vor. Dieses wird quantisiert, bevor die Inferenz über die `tflite_runtime.interpreter` Bibliothek an die externe TPU ausgelagert wird. Der Interpreter liefert ein eindimensionales `array[int8]`, das auf `array[float32]` interpoliert wird. Dieses Array beinhaltet die Wahrscheinlichkeiten für die jeweiligen Wortklassen. Die Wahrscheinlichkeiten werden ausgewertet und einer `Command`-Klasse zugeordnet.

Das Hauptmodul `e_stop_controller` holt den aktuellen `Command` vom `CommandDetector` über die Methode `.getCurrentCommand()` ab und reevaluiert den Zustandsautomaten (Abbildung 3.6).

Kommt es beim Reevaluieren des Automaten zu einem neuen Zustand, wird das über den `StatusIndicator` und gegebenenfalls `IExternalReceiver` signalisiert.

4 Evaluation

4.1 Not-Halt Tests

Um prüfen zu können ob der Not-Halt-Kontroller die spezifizierten Kernfunktionen erfüllt, wurden mehrere Tests definiert und durchgeführt.

Für aussagekräftige Testergebnisse wurden alle Testszenarien (Tabelle 4.2) insgesamt 20 Mal wiederholt. Bei den Positivtests wurden je 10 Wiederholungen von einer männlichen Testperson und weitere 10 Wiederholungen von einer weiblichen Testperson absolviert.

Zwei verschiedene Setups werden vorbereitet. Für jedes Setup wird ein anderer Datensatz für das Training des Detektion-Modells verwendet.

Setup 1: Das Speech Command Dataset [2] von Google

- Name des AMS: «Marvin» *(engl.)*
- Stopp-Befehle: «Stop» *(engl.)*
- Weiter-Befehle: «Go» *(engl.)*

Setup 2: Mit kwsCoach synthetisierter Datensatz

- Name des AMS: «Husky» *(engl.)* – «Husky» *(ger.)*
- Stopp-Befehle:
  «Stop» *(engl.)* – «Stopp» – «Halt» – «Aus» – «Abbrechen» – «Achtung»
- Weiter-Befehle: «Go» *(engl.)* – «Weiter» – «Freigeben»

Stehen mehrere alternative Wörter einer Befehlskategorie zur Verfügung, so werden diese während der Wiederholungen abgewechselt verwendet.

Schwellwerte: Nach der Inferenz wird für jede Wortklasse im Modell eine Wahrscheinlichkeit prognostiziert. Für jede Befehlskategorie (Name, Stopp, Weiter) wurde ein Schwellwert für diese Wahrscheinlichkeit festgesetzt (Tabelle 4.1). Übertrifft die prognostizierte Wahrscheinlichkeit den definierten Schwellwert, so wird dieses Wort erfolgreich detektiert.
Für beide Testsetups wurde dieselben Schwellwerte konfiguriert.

Kategorie	**Schwellwert**
Name	80 %
Stopp	90 %
Weiter	98 %

Tabelle 4.1: Erkennungsschwellwerte für die Befehlskategorien

4.1.1 Testdefinition

Tabelle 4.2: Tests des Not-Halt-Kontrollers, TEST-1 bis TEST-8 (oben)

TEST-1	**Benutzer spricht den Namen des AMS (indoor)**
Typ	Positivtest
Beschreibung	Der Benutzer spricht den Namen des AMS.
Voraussetzung	Der Benutzer und der Not-Halt-Kontroller befindet sich beide in einem Raum. Die Distanz zwischen Benutzer und dem Not-Halt-Kontroller ist zwei Meter. Es gibt keine Störgeräusche.
Ziel	Der LED-Ring leuchtet kurz auf.

TEST-2	**Benutzer spricht den Namen des AMS und ein Stopp-Befehl (indoor)**
Typ	Positivtest
Beschreibung	Der Benutzer spricht den Namen des AMS und eine Sekunde darauffolgend einen Stopp-Befehl.
Voraussetzung	Der Benutzer und der Not-Halt-Kontroller befindet sich beide in einem Raum. Die Distanz zwischen Benutzer und dem Not-Halt-Kontroller ist zwei Meter. Es gibt keine Störgeräusche. Der Not-Halt-Kontroller befindet sich im RELEASED Zustand.
Ziel	Der LED-Ring leuchtet kurz auf bei Namenserkennung. Der LED-Ring leuchtet **dauerhaft rot** nach Erkennung des Stopp-Befehls.

TEST-3	**Benutzer spricht den Namen des AMS und ein Weiter-Befehl (indoor)**
Typ	Positivtest
Beschreibung	Der Benutzer spricht den Namen des AMS und eine Sekunde darauffolgend einen Weiter-Befehl.
Voraussetzung	Der Benutzer und der Not-Halt-Kontroller befindet sich beide in einem Raum. Die Distanz zwischen Benutzer und dem Not-Halt-Kontroller ist zwei Meter. Es gibt keine Störgeräusche. Der Not-Halt-Kontroller befindet sich im STOPPED Zustand.
Ziel	Der LED-Ring leuchtet kurz auf bei Namenserkennung. Der LED-Ring leuchtet **dauerhaft grün** nach Erkennung des Weiter-Befehls.

TEST-4	**Benutzer spricht den Namen des AMS (outdoor)**
Typ	Positivtest
Beschreibung	Der Benutzer spricht den Namen des AMS.
Voraussetzung	Der Benutzer und der Not-Halt-Kontroller befindet sich beide im Außenbereich. Die Distanz zwischen Benutzer und dem Not-Halt-Kontroller ist zwei Meter. Es gibt stadttypische Störgeräusche (spielende Kinder, Autoverkehr u. v. m.) in weiterer Entfernung (>10 Meter).
Ziel	Der LED-Ring leuchtet kurz auf bei Namenserkennung.

TEST-5	**Benutzer spricht den Namen des AMS und ein Stopp-Befehl (outdoor)**
Typ	Positivtest
Beschreibung	Der Benutzer spricht den Namen des AMS und eine Sekunde darauffolgend einen Stopp-Befehl.
Voraussetzung	Der Benutzer und der Not-Halt-Kontroller befindet sich beide im Außenbereich. Die Distanz zwischen Benutzer und dem Not-Halt-Kontroller ist zwei Meter. Es gibt stadttypische Störgeräusche (spielende Kinder, Autoverkehr u. v. m.) in weiterer Entfernung (>10 Meter). Der Not-Halt-Kontroller befindet sich im RELEASED Zustand.
Ziel	Der LED-Ring leuchtet kurz auf bei Namenserkennung. Der LED-Ring leuchtet **dauerhaft rot** nach Erkennung des Stopp-Befehls.

TEST-6	**Benutzer spricht den Namen des AMS und ein Weiter-Befehl (outdoor)**
Typ	Positivtest
Beschreibung	Der Benutzer spricht den Namen des AMS und eine Sekunde darauffolgend einen Weiter-Befehl.
Voraussetzung	Der Benutzer und der Not-Halt-Kontroller befindet sich beide im Außenbereich. Die Distanz zwischen Benutzer und dem Not-Halt-Kontroller ist zwei Meter. Es gibt stadttypische Störgeräusche (spielende Kinder, Autoverkehr u. v. m.) in weiterer Entfernung (>10 Meter). Der Not-Halt-Kontroller befindet sich im STOPPED Zustand.
Ziel	Der LED-Ring leuchtet kurz auf bei Namenserkennung. Der LED-Ring leuchtet **dauerhaft grün** nach Erkennung des Weiter-Befehls.

TEST-7	**Benutzer spricht nicht (outdoor)**
Typ	Negativtest
Beschreibung	Der Benutzer spricht eine Minute nicht. Es gibt nur die Umgebungsgeräusche zu hören.
Voraussetzung	Der Benutzer und der Not-Halt-Kontroller befindet sich beide im Außenbereich. Die Distanz zwischen Benutzer und dem Not-Halt-Kontroller ist zwei Meter. Es gibt stadttypische Störgeräusche (spielende Kinder, Autoverkehr u. v. m.) in weiterer Entfernung (>10 Meter). Der Not-Halt-Kontroller befindet sich im RELEASED Zustand.
Ziel	Es gibt keine Detektion. Es gibt keinen Zustandswechsel.

TEST-8	**Benutzer spricht andere Wörter, außer die spezifizierten Sprachbefehle (outdoor)**
Typ	Negativtest
Beschreibung	Der Benutzer spricht eine Minute lang andere Wörter, außer die spezifizierten Sprachbefehle.
Voraussetzung	Der Benutzer und der Not-Halt-Kontroller befindet sich beide im Außenbereich. Die Distanz zwischen Benutzer und dem Not-Halt-Kontroller ist zwei Meter. Es gibt stadttypische Störgeräusche (spielende Kinder, Autoverkehr u. v. m.) in weiterer Entfernung (>10 Meter). Der Not-Halt-Kontroller befindet sich im RELEASED Zustand.
Ziel	Es gibt keine Detektion. Es gibt keinen Zustandswechsel.

Tabelle 4.2: Tests des Not-Halt-Kontrollers, TEST-1 bis TEST-8 (unten)

4.1.2 Testergebnisse

Testfall	**Setup 1**	**Setup 2**
TEST-1 *(Name, Indoor)*	20/20 (100 %)	15/20 (75 %)
TEST-2 (Name, Stopp, Indoor)	19/20 (95 %)	13/20 (65 %)
TEST-3 (Name, Weiter, Indoor)	17/20 (85 %)	4/20 (20 %)
TEST-4 (Name, Outdoor)	14/20 (70 %)	10/20 (50 %)
TEST-5 (Name, Stopp, Outdoor)	7/20 (35 %)	5/20 (25 %)
TEST-6 (Name, Weiter, Outdoor)	6/20 (30 %)	2/20 (10 %)
TEST-7 (Nichts sagen, Outdoor)	17/20 (85 %)	4/20 (20 %)
TEST-8 (Andere Wörter, Outdoor)	15/20 (75 %)	2/20 (10 %)
Durchschnitt	115/160 (71,875 %)	55/160 (34,375 %)

Tabelle 4.3: Testergebnisse der beides Setups (Absatz 4.1) für alle Testfälle (Tabelle 4.2). Pro Testfall wurden die erfolgreichen Wiederholungen aufsummiert.

4.2 Bewertung

4.2.1 KwsCoach

Die im Unterunterabschnitt 3.2.2 spezifizierten funktionalen Anforderungen für eine automatisierte Trainingsanwendung wurden von der GUI-Anwendung "kwsCoach" **erfolgreich** umgesetzt.

4.2.2 Not-Halt-Kontroller

Prüfen der Anforderungen

REQ-1: Das Erkennen von Stopp-Sprachbefehlen wurde **erfolgreich** implementiert.
Bestätigt durch TEST-2.

REQ-2: Das Erkennen von Namen wurde **erfolgreich** implementiert.
Bestätigt durch TEST-1.

REQ-3: Das Erkennen von Freigabe-Sprachbefehlen wurde **erfolgreich** implementiert.
Bestätigt durch TEST-3.

REQ-4: Das Anhalten des AMS wurde **nicht vollständig** implementiert.
Der Not-Halt-Kontroller wurde in kein konkretes AMS (z. B. der Husky) integriert. Die Signalisierung eines Not-Halt-Status wurde über eine LED implementiert. Funktion der Statussignalisierung über LED wurden im TEST-2 bestätigt.

REQ-5: Das AMS weiterarbeiten lassen wurde **nicht vollständig** implementiert.
Der Not-Halt-Kontroller wurde in kein konkretes AMS (z. B. der Husky) integriert. Die Signalisierung eines Not-Halt-Status wurde über eine LED implementiert. Funktion der Statussignalisierung über LED wurden im TEST-3 bestätigt.

REQ-6: Status des AMS visuell signalisieren über LED wurde **erfolgreich** implementiert. Bestätigt durch TEST-2 und TEST-3.

REQ-7: Das Modell kann im Backend allein durch Konfiguration der `config.yaml`-Datei ausgetauscht werden.
Durch Konfiguration der `config.yaml`-Datei kann der AMS Name geändert werden.
Durch Konfiguration der `config.yaml`-Datei können alle im Modell enthaltenen Sprachbefehle einer Befehlskategorie zugeordnet werden.
Anforderung wurde **erfolgreich** implementiert.

REQ-8: Die Erfüllung der harten Echtzeitanforderungen für die Verarbeitungspipeline wurde **nicht erfolgreich** ausgewertet. Ein Ergebnis der Gesamtverarbeitungsdauer wurde nicht ermittelt.

REQ-9: Die maximale Erkennungsreichweite beträgt in Innenräumen bis zu sechs Meter. Außerhalb von Innenräumen beträgt die maximale Erkennungsreichweite bis zu fünf Meter. Die definierte Erkennungsreichweite vom REQ-9 wurde damit überschritten. Die Reproduzierbarkeit lässt bei der maximalen Entfernung stark nach. Dennoch wurde die spezifizierte Anforderung damit **erfolgreich** erfüllt.

4.2.3 Fazit

Das System des Not-Halt-Kontrollers läuft AMS-unabhängig auf effizienter und gesonderter Hardware und kann diverse Sprachbefehle erkennen und umsetzen. Die gemessene Genauigkeit (Tabelle 4.2) des Systems macht es aus meiner Sicht interessant für den Einsatz als echte Not-Halt-Lösung, allerdings nicht für die breite Gesellschaft. Denn die besseren Ergebnisse konnten nur mit einem englischsprachigen Speech Command Dataset [2] von Google umgesetzt werden. Dieses hat den Nachteil, dass nur ein eingeschränkter Wortschatz enthalten ist, zusätzlich nur auf Englisch.

Das Setup (Absatz 4.1) mit einem Modell auf Basis von synthetischen, deutschen Audiodaten liefert deutlich schlechtere Ergebnisse und somit absolut nicht tauglich für diesen Einsatz.

Der Einsatz als Not-Halt-System in einem Forschungsumfeld ist die englischsprachige Lösung mit dem Speech Command Dataset [2] geeignet.

Literaturverzeichnis

[1] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications." https://arxiv.org/pdf/1704.04861.pdf, 2017. Zugriff: 31.03.2022.

[2] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition." https://arxiv.org/abs/1804.03209, 2018. Zugriff: 12.03.2022.

[3] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers." https://arxiv.org/abs/1711.07128, 2017. Zugriff: 12.03.2022.

[4] Red Hat, Inc, "Was ist yaml?." https://www.redhat.com/de/topics/automation/what-is-yaml, 2021. Zugriff: 14.08.2022.

[5] A. Takimoglu, "What is data augmentation? techniques and examples in 2022." https://research.aimultiple.com/data-augmentation, 2022. Zugriff: 05.05.2022.

[6] J. Lyons, "Mel frequency cepstral coefficient (mfcc) tutorial." http://www.practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/, 2013. Zugriff: 09.07.2022.

[7] Schalk GmbH, "Unterschied zwischen den funktionen not-halt und not-aus." https://schalk-elektrotechnik.de/enzyklopaedie/not-halt-vs-not-aus, 2020. Zugriff: 13.03.2022.

[8] S. Pareigis, T. Tiedemann, and M. A. D. Muirier, "Test area intelligent quartier mobility (tiq)." https://autosys.informatik.haw-hamburg.de/project/smartmobility, 2021. Zugriff: 12.03.2022.

[9] F. Tenzer, "Absatz von intelligenten lautsprechern weltweit vom 3. quartal 2016 bis zum 3. quartal 2021." https://de.statista.com/statistik/daten/studie/818982/umfrage/absatz-von-intelligenten-lautsprechern-weltweit-pro-quartal, 2022. Zugriff: 13.03.2022.

[10] Academic Dictionaries and Encyclopedias, "Echtzeitfähigkeit." https://de-academic.com/dic.nsf/dewiki/369201. Zugriff: 14.03.2022.

[11] PNGWING, "Computer icons speech, speaking, text, logo, monochrome png." https://www.pngwing.com/en/free-png-nrzpw. Zugriff: 27.03.2022.

[12] A. Sarkar, "Understanding depthwise separable convolutions and the efficiency of mobilenets." https://towardsdatascience.com/understanding-depthwise-separable-convolutions-and-the-efficiency-of-mobilenets-6de3d6b62503, 2021. Zugriff: 31.03.2022.

[13] A. Limited, "New keyword spotting for microcontrollers." https://github.com/ARM-software/ML-examples/tree/master/tflu-kws-cortex-m, 2021. Zugriff: 03.04.2022.

[14] ARM Limited, "Cortex-a53." https://developer.arm.com/Processors/Cortex-A53. Zugriff: 26.05.2022.

[15] Google LLC, "Get started with the usb accelerator." https://coral.ai/docs/accelerator/get-started/#requirements. Zugriff: 26.05.2022.

[16] Raspberry Pi Foundation, "Raspberry pi zero 2 w specification." https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/, 2021. Zugriff: 12.03.2022.

[17] Google LLC, "Coral usb accelerator datasheet." https://coral.ai/static/files/Coral-USB-Accelerator-datasheet.pdf, 2020. Zugriff: 22.05.2022.

[18] Seeed Technology Co.,Ltd., "Respeaker usb mic array." https://wiki.seeedstudio.com/ReSpeaker-USB-Mic-Array, 2021. Zugriff: 26.05.2022.

[19] Waveshare Electronics, "Ethernet / usb hub hat (b) for raspberry pi series, 1x rj45, 3x usb 2.0." https://www.waveshare.com/eth-usb-hub-hat-b.htm, 2020. Zugriff: 26.05.2022.

[20] Papers With Code, "Keyword spotting on google speech commands." https://paperswithcode.com/sota/keyword-spotting-on-google-speech-commands. Zugriff: 11.06.2022.

[21] R. Vygon and N. Mikhaylovskiy, "Learning efficient representations for keyword spotting with triplet loss." https://paperswithcode.com/paper/learning-efficient-representations-for-1, 2021. Zugriff: 11.06.2022.

[22] TensorFlow, "Training checkpoints." https://www.tensorflow.org/guide/checkpoint. Zugriff: 25.06.2022.

[23] eSpeak, "espeak text to speech." http://espeak.sourceforge.net/. Zugriff: 07.05.2022.

[24] Gespeaker, "Gespeaker." https://www.muflone.com/gespeaker/english. Zugriff: 07.05.2022.

[25] R. Clark and A. Black, "The festival speech synthesis system." https://www.cstr.ed.ac.uk/projects/festival. Zugriff: 07.05.2022.

[26] SVOX, "libttspico-utils." https://packages.debian.org/de/stretch/libttspico-utils. Zugriff: 07.05.2022.

[27] Google, "Text-to-speech, a speech service feature that converts text to lifelike speech." https://azure.microsoft.com/de-de/services/cognitive-services/text-to-speech. Zugriff: 07.05.2022.

[28] Microsoft, "Azure text-to-speech, a speech service feature that converts text to lifelike speech." https://azure.microsoft.com/de-de/services/cognitive-services/text-to-speech. Zugriff: 07.05.2022.

[29] IBM, "Watson text to speech." https://www.ibm.com/de-de/cloud/watson-text-to-speech. Zugriff: 07.05.2022.

A Fragebogen: Brauchen die Roboter von morgen ein neues Not-Halt-System?

A.1 Willkommen

Hi! - und danke, dass du dir 5 Minuten Zeit nimmst für diese Umfrage. In unserem Forschungsprojekt verfolgen wir die Vision den städtischen, öffentlichen Raum umzugestalten und mit moderner, autonomer Technik auf ein neues Komfort- und Effizienz-Level zu bringen.

Aufgaben für autonome Robotersysteme könnten sein:

- Öffentlicher Personentransport
- Paketzustellung, Anlieferung für Geschäfte
- Sammeln von Abfall, Laub

und vieles mehr.

Menschliche Gesundheit und Privatsphäre haben dabei die höchste Priorität. Deswegen setzen wir unseren Fokus auf ein neues Not-Halt-System, welches jeder bedienen kann.

A.2 Kurz zu deiner Person

Diese Angaben sind freiwillig, aber sie helfen uns, die Ergebnisse einzuschätzen.

A.2.1 Wie alt bist du?

Auswahl:

- 15 - 19
- 20 - 29
- 30 - 39
- 40 - 49
- 50 - 59
- $>$ 60

A.2.2 Hast du einen technischen Hintergrund?

Auswahl:

- Ich habe einen Beruf oder Ausbildung mit rein technischem Bezug
- Ich habe einen Beruf oder Ausbildung mit technischem Bezug
- Ich bin technisch interessiert
- Ich bin technisch wenig interessiert

A.3 Wie stehst du zur Technik?

Bitte bewerte die folgend kommenden Aussagen. Was denkst du über diese Aussagen? Schätze die Aussagen deiner Meinung nach ab.

Falls du die Aussage nicht eindeutig einstufen kannst, kannst du sie überspringen.

Aussage	trifft nicht zu	trifft weniger zu	trifft eher zu	trifft zu
Neue Technologien interessieren mich				
In Zukunft möchte ich weniger Autos/LKWs in der Stadt				
Autonome Roboter werden eine Bereicherung sein				
Technik im öffentlichen Raum macht mir Sorgen				
Ich habe Angst vor autonomen Robotern				
Autonome Roboter werden sicherer als menschengeführte Autos sein				

A.4 Fragen zum Not-Halt

A.4.1 Bist du der Meinung, dass jeder Roboter im öffentlichen Raum einen Not-Halt haben sollte?

Auswahl:

- Ja
- Nein
- Nein, nur welche mit hohem Risiko

A.4.2 Soll jede Person den Not-Halt auslösen dürfen?

Auswahl:

- Ja
- Nein, nur Techniker oder Besitzer des Robots

A.4.3 Welche Auslöser für einen Not-Halt am Roboter sind sinnvoll und intuitiv?

Mehrfachauswahl:

- Handzeichen
- Sprachbefehle
- Kollision mit Mensch oder Hindernis
- Per Handy (App) auslösen
- Notschalter, Knopf am Roboter
- In die Hände klatschen
- *eigene Eingabe ...*

A.5 Wie würdest du selbst reagieren?

In den folgenden zwei Situationen würden wir gerne wissen, wie du dich in einer Notsituation verhalten wollen würdest, um den Not-Halt auszulösen und damit die Gefahr zu entschärfen.

Versuch dich in die folgenden Szenen hineinzuversetzen.

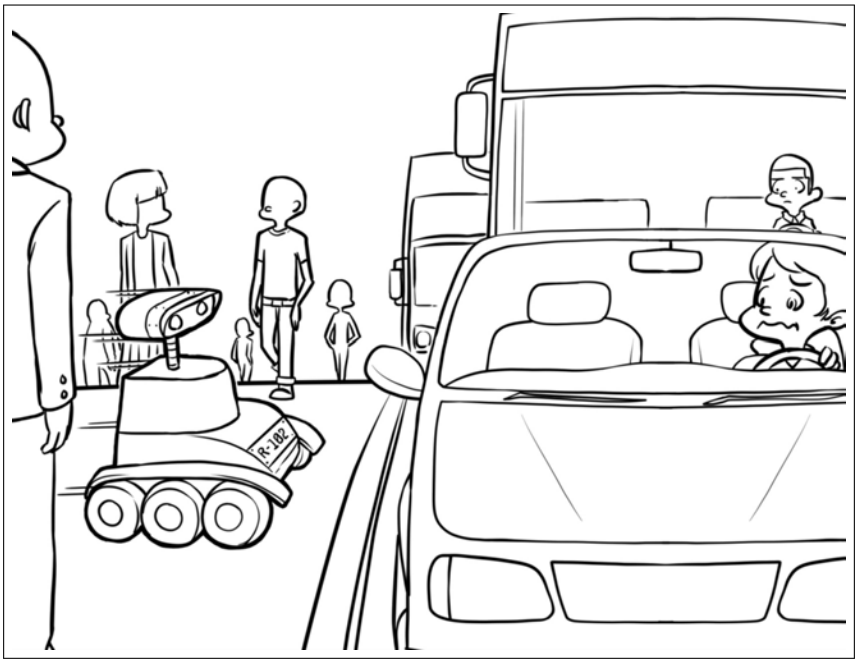


Abb. A.1: Als Beobachter dieser Situation siehst du einen Roboter mit der Aufschrift "R-102" auf eine stark befahrene Straße rasen

Auswahl:

- Roboter per Handy (App) ausschalten
- "Stopp" rufen
- Auf Roboter zeigen und "Stopp!" rufen
- Den Weg versperren
- Hinlaufen, Notschalter betätigen
- Roboter machen lassen
- Roboter festhalten
- Durch Winken sich aufmerksam machen
- "R-102 Stopp!" rufen
- *eigene Eingabe ...*

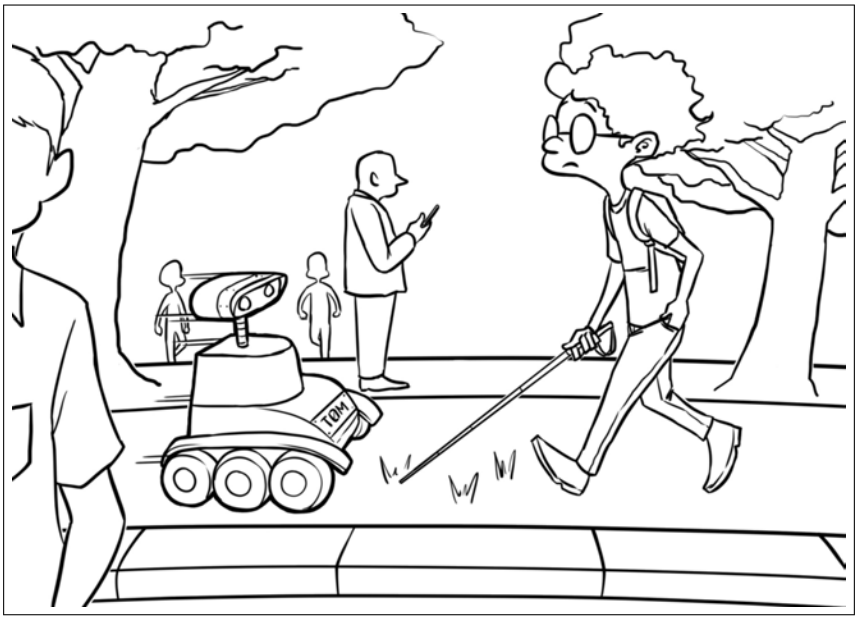


Abb. A.2: Als Beobachter dieser Situation siehst du einen Roboter mit der Aufschrift "TOM" auf eine blinde Person zufahren

Auswahl:

- Roboter per Handy (App) ausschalten
- "Stopp" rufen
- "Tom - Stopp!" rufen - **neue Antwort**
- Auf Roboter zeigen und "Stopp!" rufen
- Den Weg versperren
- Hinlaufen, Notschalter betätigen
- Roboter machen lassen
- Roboter festhalten
- Durch Winken sich aufmerksam machen
- *eigene Eingabe ...*

B Hello Edge: Keyword Spotting on Microcontrollers [3] (Anhänge)

Diese Anhänge sind unverändert aus dem technischen Bericht *Hello Edge: Keyword Spotting on Microcontrollers* [3] übernommen worden.

NN size	**NN memory limit**	**Ops/inference limit**
Small (S)	80 KB	6 MOps
Medium (M)	200 KB	20 MOps
Large (L)	500 KB	80 MOps

Tabelle B.1: Neural network (NN) classes for KWS models considered in this work, assuming 10 inferences per second and 8-bit weights/activations. Quelle: [3]

NN model	S(80KB, 6MOps)			M(200KB, 20MOps)			L(500KB, 80MOps)		
	Acc.	Mem.	Ops	Acc.	Mem.	Ops	Acc.	Mem.	Ops
DNN	84.6%	80.0KB	158.8K	86.4%	199.4KB	397.0K	86.7%	496.6KB	990.2K
CNN	91.6%	79.0KB	5.0M	92.2%	199.4KB	17.3M	92.7%	497.8KB	25.3M
Basic LSTM	92.0%	63.3KB	5.9M	93.0%	196.5KB	18.9M	93.4%	494.5KB	47.9M
LSTM	92.9%	79.5KB	3.9M	93.9%	198.6KB	19.2M	94.8%	498.8KB	48.4M
GRU	93.5%	78.8KB	3.8M	94.2%	200.0KB	19.2M	94.7%	499.7KB	48.4M
CRNN	94.0%	79.7KB	3.0M	94.4%	199.8KB	7.6M	95.0%	499.5KB	19.3M
DS-CNN	94.4%	38.6KB	5.4M	94.9%	189.2KB	19.8M	95.4%	497.6KB	56.9M

Tabelle B.2: Summary of best neural networks from the hyperparameter search. The memory required for storing the 8-bit weights and activations is shown in the table. Quelle: [3]

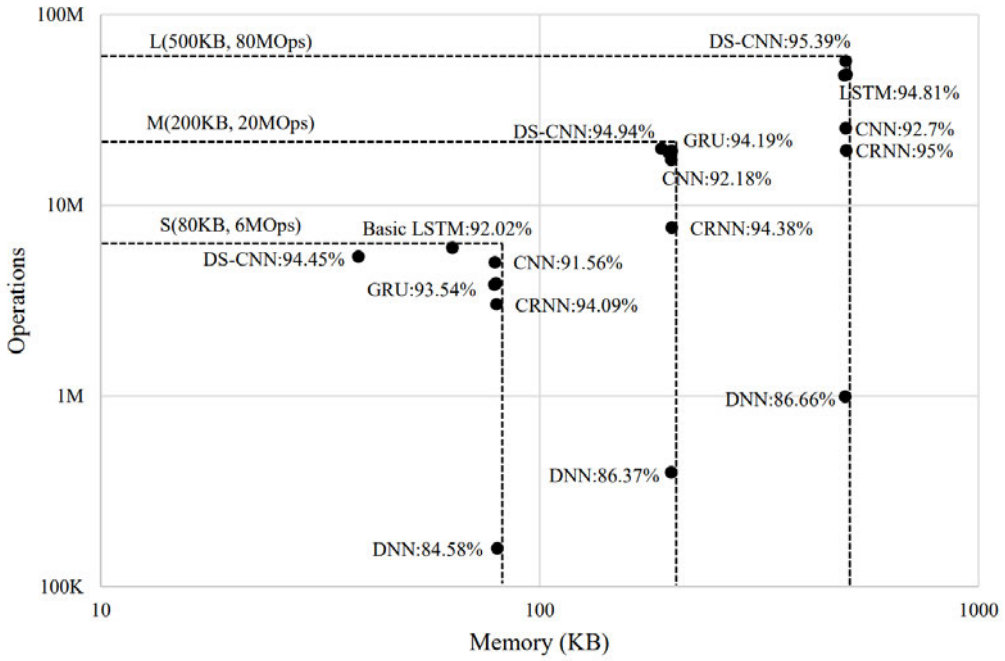


Abb. B.1: Memory vs. Ops/inference of the best models described in "Tabelle B.2". Quelle: [3]

C Machine Learning Pipeline Logs

C.1 POC-1

C.1.1 Training

```
Epoch 1/75
400/400 [==============================] -
185s 452ms/step - loss: 0.8760 - accuracy: 0.7192
val_loss: 2.9305 - val_accuracy: 0.0835
[...]
Epoch 74/75
400/400 [==============================] -
257s 642ms/step - loss: 2.6475e-06 - accuracy: 1.0000
val_loss: 0.6039 - val_accuracy: 0.9069
Epoch 75/75
400/400 [==============================] -
261s 653ms/step - loss: 2.4049e-06 - accuracy: 1.0000
val_loss: 0.6065 - val_accuracy: 0.9075

49/49 [==============================]
6s 131ms/step - loss: 0.6419 - accuracy: 0.9029
Final test accuracy: 90.29%
```

Modell:

`ds-cnn-for-kws/work/DS_CNN/DS_CNN3_v3/training/best/ds_cnn_0.910_ckpt`

Modellübersicht

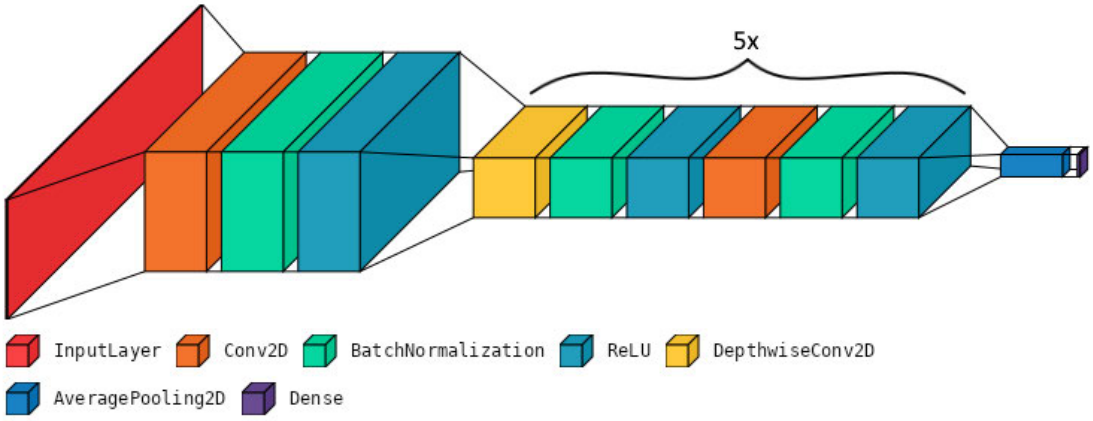


Abb. C.1: Vereinfachte Darstellung der DS-CNN Modellarchitektur in Größenvariante L. Gekennzeichneter Mittelteil ist periodisch wiederholend.

Detaillierte Modellzusammenfassung

```
model.summary()
Model: "DS-CNN (L)"
```

Layer (type)	Output Shape	Param #
input (InputLayer)	[(None, 49, 10)]	0
tf.reshape (TFOpLambda)	(None, 49, 10, 1)	0
conv2d (Conv2D)	(None, 25, 10, 276)	11316
batch_normalization (BatchNo	(None, 25, 10, 276)	1104
re_lu (ReLU)	(None, 25, 10, 276)	0
depthwise_conv2d (DepthwiseC	(None, 13, 5, 276)	2760
batch_normalization_1 (Batch	(None, 13, 5, 276)	1104
re_lu_1 (ReLU)	(None, 13, 5, 276)	0

conv2d_1 (Conv2D)	(None, 13, 5, 276)	76452
batch_normalization_2 (Batch	(None, 13, 5, 276)	1104
re_lu_2 (ReLU)	(None, 13, 5, 276)	0
depthwise_conv2d_1 (Depthwis	(None, 13, 5, 276)	2760
batch_normalization_3 (Batch	(None, 13, 5, 276)	1104
re_lu_3 (ReLU)	(None, 13, 5, 276)	0
conv2d_2 (Conv2D)	(None, 13, 5, 276)	76452
batch_normalization_4 (Batch	(None, 13, 5, 276)	1104
re_lu_4 (ReLU)	(None, 13, 5, 276)	0
depthwise_conv2d_2 (Depthwis	(None, 13, 5, 276)	2760
batch_normalization_5 (Batch	(None, 13, 5, 276)	1104
re_lu_5 (ReLU)	(None, 13, 5, 276)	0
conv2d_3 (Conv2D)	(None, 13, 5, 276)	76452
batch_normalization_6 (Batch	(None, 13, 5, 276)	1104
re_lu_6 (ReLU)	(None, 13, 5, 276)	0
depthwise_conv2d_3 (Depthwis	(None, 13, 5, 276)	2760
batch_normalization_7 (Batch	(None, 13, 5, 276)	1104
re_lu_7 (ReLU)	(None, 13, 5, 276)	0
conv2d_4 (Conv2D)	(None, 13, 5, 276)	76452
batch_normalization_8 (Batch	(None, 13, 5, 276)	1104

```
_________________________________________________________________
re_lu_8 (ReLU)               (None, 13, 5, 276)        0
_________________________________________________________________
depthwise_conv2d_4 (Depthwis (None, 13, 5, 276)        2760
_________________________________________________________________
batch_normalization_9 (Batch (None, 13, 5, 276)        1104
_________________________________________________________________
re_lu_9 (ReLU)               (None, 13, 5, 276)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 13, 5, 276)        76452
_________________________________________________________________
batch_normalization_10 (Batc (None, 13, 5, 276)        1104
_________________________________________________________________
re_lu_10 (ReLU)              (None, 13, 5, 276)        0
_________________________________________________________________
average_pooling2d (AveragePo (None, 1, 1, 276)         0
_________________________________________________________________
tf.reshape_1 (TFOpLambda)    (None, 276)               0
_________________________________________________________________
dense (Dense)                (None, 12)                3324
=================================================================
Total params: 422,844
Trainable params: 416,772
Non-trainable params: 6,072
_________________________________________________________________
```

Test

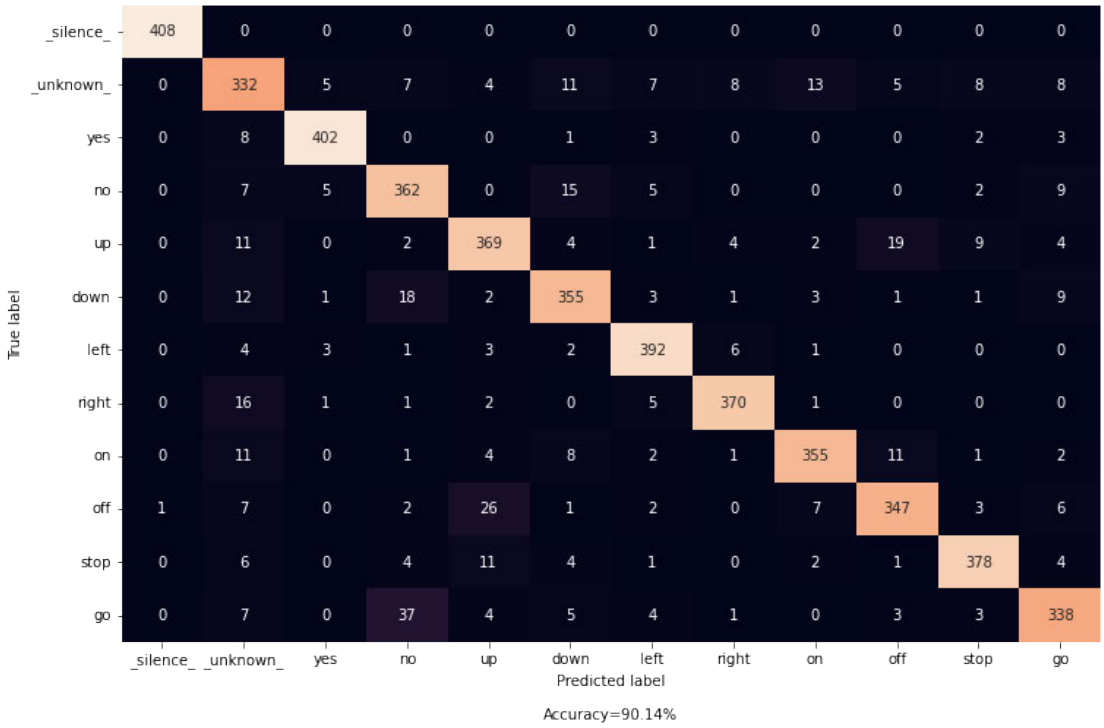


Abb. C.2: Confusion Matrix: Test nach dem POC-1 Training

C.1.2 Konvertierung

Launch Configuration:

```
{
    "name": "Convert(ckpt␣->␣tflite)",
    "type": "python",
    "request": "launch",
    "program": "convert.py",
    "console": "integratedTerminal",
    "args": ["--model_architecture","ds_cnn",
             "--model_size_info", "6", "276", "10", "4", "2", "1",
                 ↪ "276", "3", "3", "2", "2", "276", "3", "3", "1",
                 ↪ "1", "276", "3", "3", "1", "1", "276", "3", "3",
                 ↪ "1", "1", "276", "3", "3", "1", "1",
             "--checkpoint",
             "work/DS_CNN/DS_CNN3_v3/training/best/ds_cnn_0.910_ckpt",
             "--quantize"]
```

```
12 }
```

Size: 4.931 KB → 492 KB
Accuracy: 90,14% → 90,04% (Abbildung C.3)

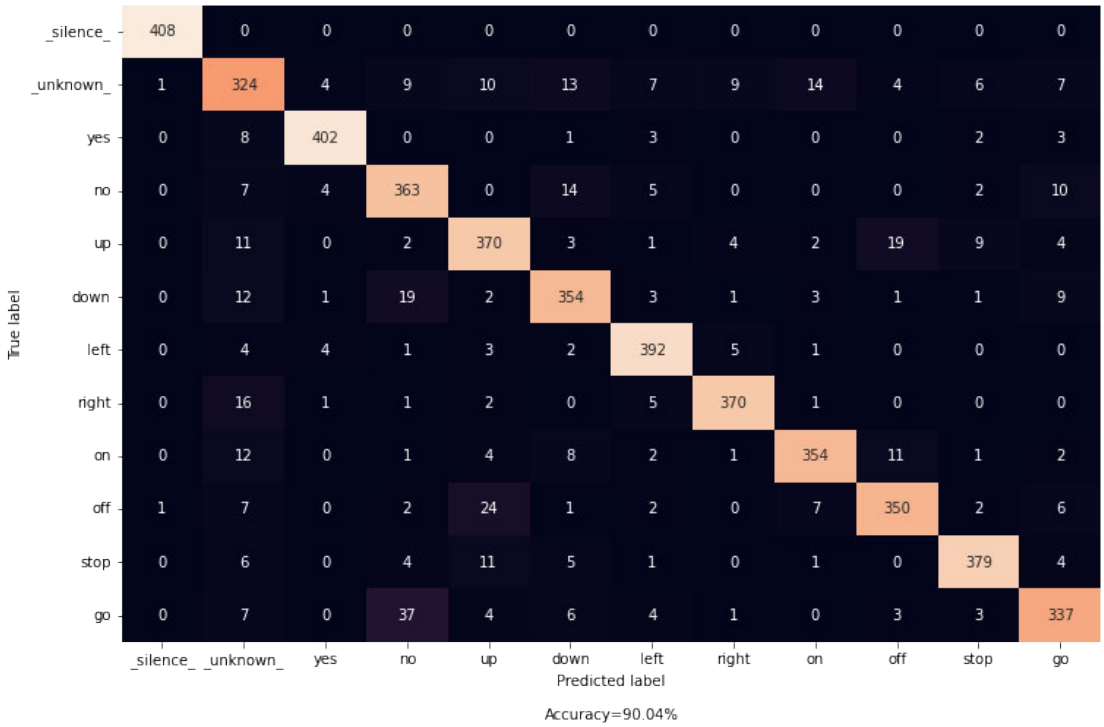


Abb. C.3: Confusion Matrix: Test nach Konvertierung des POC-1 Modells

C.1.3 Kompilierung

Command:

```
> edgetpu_compiler \
    ds_cnn_quantized_v3.tflite \
    ds_cnn_quantized_v3_edgetpu.tflite
```

Output:

```
Input model: ds_cnn_quantized_v3.tflite
Input size: 492.21KiB
```

```
Output model: ds_cnn_quantized_v3_edgetpu.tflite
Output size: 660.84KiB
On-chip memory used for caching model parameters: 586.25KiB
On-chip memory remaining for caching model parameters: 7.09MiB
Off-chip memory used for streaming uncached model parameters: 0.00B
Number of Edge TPU subgraphs: 1
Total number of operations: 18
Operation log: ds_cnn_quantized_v3_edgetpu.log
```

Log:

```
Edge TPU Compiler version 16.0.384591198
Input: ds_cnn_quantized_v3.tflite
Output: ds_cnn_quantized_v3_edgetpu.tflite

Operator            Count   Status

DEPTHWISE_CONV_2D   5       Mapped to Edge TPU
QUANTIZE            1       Operation is otherwise supported
                            but not mapped due to some
                            unspecified limitation
AVERAGE_POOL_2D     1       Mapped to Edge TPU
RESHAPE             2       Mapped to Edge TPU
FULLY_CONNECTED     1       Mapped to Edge TPU
DEQUANTIZE          1       Operation is working on an
                            unsupported data type
SOFTMAX             1       Mapped to Edge TPU
CONV_2D             6       Mapped to Edge TPU
```

D Not-Halt-Kontroller Externe Abhängigkeiten

Eine kleine Zusammenfassung über die externen, genutzten Open-Source oder Drittanbieter Softwarepakete.

Tabelle D.1: Detailinformationen zu den Externen Bibliotheken des Not-Halt-Kontrollers

Name	**python_speech_features**
Genutzte Version	v. 0.6.1
Letztes Update	14 Januar 2020 (v. 0.6.1)
Autor	James Lyons
Lizenz	MIT license
Webseite	https://python-speech-features.readthedocs.io
Zweck	Einsatz ist in der Audio-Vorverarbeitungskette. Es wird die MFCC Signalverarbeitungsmethode verwendet um Sprachfeatures aus dem Audiosignal zu extrahieren.

Name	**PyAudio**
Genutzte Version	v. 0.2.11
Letztes Update	18 März 2017 (v. 0.2.11)
Autor	Hubert Pham
Lizenz	MIT license
Webseite	http://people.csail.mit.edu/hubert/pyaudio
Zweck	PyAudio wird verwendet um Audiostreams von Audioaufnahmegeräten in den Python Programmcode einbinden können.

Name	**tflite_runtime**
Genutzte Version	v. 2.5.0.post1
Letztes Update	15 Juni 2022 (v. 2.9.1)
Autor	Google, LLC
Lizenz	Apache Software License (Apache 2.0)
Webseite	https://pypi.org/project/tflite-runtime
Zweck	Wird verwendet als Python Schnittstelle zu der Edge-TPU-Runtime (libedgetpu1-std).

Name	**libedgetpu1-std**
Genutzte Version	TODO
Letztes Update	08 März 2022 (v. 20220308)
Autor	Google, LLC - Coral.ai
Lizenz	Apache Software License (Apache 2.0)
Webseite	https://github.com/google-coral/libedgetpu
Zweck	Die Edge-TPU-Runtime sind Treiber auf Betriebssysteme-Ebene, zur Kommunikation mit der Coral USB Accelerator Hardware.

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