Konzeption und Entwicklung von Lernbausteinen im Themengebiet der Künstlichen Intelligenz mit dem Fokus auf die Anwendungsfelder Servicerobotik und Smart Home

MASTERARBEIT

zur Erlangung des akademischen Grades
Master of Engineering "M.Eng"

an der Technischen Hochschule Wildau

Technische Hochschule Wildau
Fachbereich Ingenieurwesen / Wirtschaftsingenieurwesen
Studiengang Ingenieurwesen
Studienrichtung Telematik

eingereicht am:	18. April 2017
Name:	Janine Breßler
geboren am:	01. Februar 1983
Betreuer:	Prof. Dr. rer. nat. Janett Mohnke Prof. Dipl.-Inf. Birgit Wilkes
Themenstellender Betrieb:	Technische Hochschule Wildau

Bibliografische Beschreibung und Referat

Breßler, Janine

Konzeption und Entwicklung von Lernbausteinen im Themengebiet der Künstlichen Intelligenz mit dem Fokus auf die Anwendungsfelder Servicerobotik und Smart Home.

Masterarbeit, Technische Hochschule Wildau 2017, 168 Seiten, 82 Abbildungen, 11 Tabellen, 2 Anlagen.

Ziel

Das Ziel dieser Masterarbeit ist die Entwicklung von Lernbausteinen, die für den praktischen Einsatz in Schülerlaboren, Seminararbeiten sowie Abschlussarbeiten im Studiengang Telematik eingesetzt, angewandt und weiterentwickelt werden können.

Die Lernbausteine sollen das Thema *Künstliche Intelligenz* (KI) behandeln. Hierfür soll eine Anwendung entwickelt werden, die mit Hilfe von Werkzeugen der KI dazu in der Lage ist, die beiden Bereiche *Servicerobotik* und *Smart Home* miteinander zu verbinden, um damit die Verhaltensabläufe einer Person in ihren eigenen vier Wänden zu erlernen und zu überwachen.

Inhalt

Die Masterarbeit beschreibt die Evaluation, die Konzeption und die Entwicklung einer Anwendung sowie eines Lernbaukastens, durch die einerseits das Verbinden der Bereiche *Smart Home* und *Servicerobotik* mit Hilfe von künstlichen neuronalen Netzen (KNN) ermöglicht und andererseits die Arbeitsweise von KNN an Schüler der Sekundarstufe 2 sowie Studierende des Studiengangs Telematik vermittelt werden sollen.

Hierfür wird zunächst die grundlegende Idee der Arbeit näher erläutert und anschließend eine kurze Einführung in die drei Bereiche *Künstliche Intelligenz*, *Servicerobotik* und *Smart Home* gegeben.

Daran anschließend werden die Anforderungen an die jeweilige Lösung sorgsam geplant und konzeptioniert. So werden bspw. verwandte Lösungen untersucht, um daraus wichtige Erkenntnisse für die eigene Lösung nutzen oder übernehmen zu können. Bei der Auswahl eines geeigneten Frameworks für die Arbeit mit KNN werden verschiedene Lösungen auf relevante Kriterien hin untersucht und darauf aufbauend eine Auswahl getroffen.

Bei der Planung des Lernbaukastens wird die bewährte Lehrstrategie MOMBI (Model Of Model-Based Instruction) vorgestellt und im Folgenden auf die individuellen Bedürfnisse hin adaptiert. Zusätzlich wird die erfolgreiche Philosophie der bereits bestehenden Schülerlabore erläutert und deren Grundsätze in den entwickelten Lernbausteinen berücksichtigt.

Abschließend wird eine Testphase durchgeführt und dokumentiert, in der unter Einsatz der entwickelten Anwendung und des Prototyps ein erstes übersichtliches Szenario getestet wird, das den Schlafrhythmus in Abhängigkeit zur Zeit überwacht.

Im Anhang dieser Masterarbeit ist eine CD-ROM beigelegt, die den Lernbaukasten sowie den Quellcode und eine funktionsfähige Version der entwickelten Anwendung enthält.

Abstract

In dieser Arbeit wurde eine Anwendung entwickelt, die das Verbinden der beiden technischen Bereiche *Smart Home* und *Servicerobotik* mit Hilfe von künstlichen neuronalen Netzen (KNN) ermöglicht, um damit Verhaltensabläufe einer Person in ihren eigenen vier Wänden zu erlernen und dabei auf grobe Abweichungen hin zu überwachen. Zusätzlich wurde ein Lernbaukasten für Schülergruppen der Sekundarstufe 2 entwickelt, mit dem die Funktionsweise von KNN vermittelt wird.

Es wurden unterschiedliche Werkzeuge der Künstlichen Intelligenz untersucht sowie Projekte mit ähnlichen Zielen, um darauf aufbauend eine geeignete Anwendung zu entwickeln. Der entwickelte Lernbaukasten basiert auf der Lehrstrategie MOMBI (Model Of Model-Based Instruction).

In der Testphase konnte erfolgreich ein erstes Anwendungsszenario getestet werden, dass das Überwachen des Schlafrhythmus einer Person auf Unregelmäßigkeiten hin ermöglicht.

Ein erster Test hat gezeigt, dass neuronale Netze gut dafür geeignet sind, um Tätigkeiten mit einem regelmäßigen Verhalten zu überwachen. Im nächsten Schritt sollten die zu überwachenden Tätigkeitsvielfalt ausgebaut werden und auch die Verwendung von weiteren Werkzeugen der Künstlichen Intelligenz in Betracht gezogen werden.

Over the course of this thesis, an application was developed which combined the fields of smart home and service robotics utilizing artificial neural networks (ANN), in order to learn the behavioral patterns of a person in their home and monitor them for significant deviations. Additionally, a learning tool comprising of educational building sets was developed for groups of students in second stage secondary education, to convey the functional principles of ANN.

Various AI tools, as well as other projects with similar goals, were analyzed in order to build an application on their basis. The learning tool developed for this thesis is based on the teaching strategy MOMBI (Model Of Model-Based Instruction).

During the testing phase, a first application scenario which enables the monitoring of a person's sleep patterns for irregularities was successfully completed.

A pilot has shown that neural networks are a good fit for monitoring regular behavior. The next steps should include extending the range of activities to be monitored, as well as considering the utilization of additional AI tools.

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Abschlussarbeit selbstständig angefertigt und nur die angegebenen Hilfsmittel und Quellen verwendet habe.

Wildau, den 13. April 2017

.....

Janine Breßler

Inhaltsverzeichnis

Danksagung	5
Hinweise zum Lesen der Arbeit	7
1 Einleitung	8
1.1 Motivation	8
1.2 Zielsetzung	10
1.3 Abgrenzung	11
1.4 Gliederung der Arbeit	11
2 Künstliche Intelligenz	13
2.1 Was ist Künstliche Intelligenz - Eine Definition	13
2.2 Die Geschichte der KI - Eine Einführung	14
2.3 KI heute - Der Stand der Technik	17
2.3.1 Aussagenlogik	18
2.3.2 Prädikatenlogik erster Stufe	20
2.3.3 Schließen mit Unsicherheit	20
2.3.4 Suchalgorithmen zur Problemlösung	21
2.3.5 Konnektionismus - Künstliche neuronale Netze	24
2.4 Eine Auswahl an Anwendungsbereichen der KI	29
3 Smart Home und Servicerobotik	32
3.1 Smart Home	32
3.1.1 Was ist Smart Home - Eine Definition	32
3.1.2 Eine Auswahl an Anwendungsbereichen im eigenen Zuhause	33
3.2 Serviceroboter	34
3.2.1 Was ist ein Serviceroboter - Eine Definition	34
3.2.2 Eine Auswahl an Anwendungsgebieten der Servicerobotik	35
4 Analyse der Anforderungen	37
4.1 Das große Ganze - Was soll erreicht werden?	37
4.1.1 Das Szenario	37
4.1.2 Intelligente Verarbeitungseinheit	38
4.1.3 Lernbausteine	40
4.1.4 Entwicklungsbausteine	40
4.1.5 Wer gehört zur Zielgruppe?	40
4.2 Anwendungsfälle für die intelligente Verarbeitungseinheit	41
4.2.1 Sensoreinheit	41
4.2.2 Anwender	42

4.2.3	Serviceroboter	45
4.3	Definition der Anforderungen	46
4.3.1	Anforderungen an die Lernbausteine	46
4.3.2	Anforderungen an die intelligente Verarbeitungseinheit	47
4.3.3	Anforderungen an die Entwicklungsbausteine	47
5	Lösungskonzeption der intelligenten Verarbeitungseinheit	48
5.1	Related works - Analyse einer Auswahl wichtiger Systeme	48
5.1.1	Das Monitoringsystem SAMDY	48
5.1.2	CASAS: A Smart Home in a Box	50
5.1.3	Power Selekt Management	52
5.1.4	Schlussfolgerung	53
5.2	Anwendungsmodi für den Benutzer	54
5.2.1	Discover-Modus	54
5.2.2	Trainings-Modus	57
5.2.3	Go Live-Modus	58
5.3	Schnittstellenbereitstellung Sensorik	60
5.4	Analyse und Auswahl einer intelligenten Software	61
5.4.1	Neuroph	61
5.4.2	Encog Machine Learning Framework	62
5.4.3	JOONE	63
5.4.4	Schlussfolgerung	63
5.5	Das Prinzip „Out of the box“	65
5.5.1	Gradle - Build-Management-System	65
5.5.2	Persistenz	66
5.5.3	Jersey - RESTful Web Services in Java	67
5.6	Graphische Benutzeroberfläche	69
5.6.1	JavaFX	69
5.6.2	Responsive Design	69
5.6.3	Aufbau der Benutzeroberfläche	70
5.7	Zusammenfassung der Systemarchitektur	73
6	Lösungskonzeption des Lernbaukastens und der Entwicklungsbausteine	75
6.1	Der Lernprozess	75
6.2	MOMBI - Model-of-Model-Based Instruction	76
6.3	Der Lernbaukasten	78
6.3.1	Lernbaustein 1: Theorie	79
6.3.2	Lernbaustein 2: Praxis	81
6.3.3	Lernbaustein 3: Festigung	86
6.4	Geplante Entwicklungsbausteine	87
6.4.1	Entwicklungsbaustein <i>5. Prüfungskomponente</i>	87
6.4.2	Entwicklungsbaustein <i>Embedded C++</i>	88
6.4.3	Entwicklungsbaustein <i>Abschlussarbeit</i>	89

7	Technische Umsetzung	91
	<i>Smarter Home</i>	
7.1	Grundlegende Anwendungsarchitektur	91
7.2	Grafische Benutzeroberfläche	92
7.2.1	Bedienelemente	92
7.2.2	Benutzerführung	98
7.2.3	Komponenten der Benutzeroberfläche	99
7.3	Serielle Schnittstelle	100
7.3.1	Nachrichtenformate	101
7.3.2	RXTX Communication Service	103
7.4	Sensor Manager Service	104
7.5	Deep Learning Service	105
7.5.1	Eingangs- und Ausgangsneuronen	105
7.5.2	Neuronales Netzwerk	108
7.6	Encog Framework	109
7.6.1	Erstellung des neuronalen Netzes	109
7.6.2	Trainieren des Neuronalen Netzes	110
7.6.3	Einsatz des neuronalen Netzes	111
7.7	Speichern und Laden von KNN	111
7.7.1	Relevante Informationen	111
7.7.2	Speichervorgang	112
7.8	Embedded REST Server	113
7.8.1	Serverarchitektur	113
7.8.2	Aufbau REST API	113
7.9	Abweichungen vom Konzept	118
7.9.1	Simulieren und Überschreiben von Eingangsneuronen	118
7.9.2	Importieren von Trainingsdaten	119
7.10	Gesamtübersicht	122
8	Umsetzung Lernbaukasten	123
8.1	Lernbaustein Theorie	123
8.1.1	Interaktive Mitarbeit	123
8.1.2	Begleitende Arbeitsblätter	124
8.1.3	Provozieren	125
8.1.4	Aktivieren	125
8.1.5	Informieren	126
8.2	Lernbaustein Praxis	127
8.2.1	Prototyp	127
8.2.2	Sensorik	129
8.2.3	Tutorial	130
8.3	Lernbaustein Festigung	131
9	Testphase	132
9.1	Schlafrhythmus in Abhängigkeit zur Zeit	132
9.1.1	Trainingsdaten	133
9.1.2	Multilayer Perzeptron 1x4	134
9.1.3	Multilayer Perzeptron 2x8	138
9.1.4	Zusammenfassung	141

10 Abschlussbetrachtung	143
10.1 Ausblick	143
10.1.1 Weitere Konfigurationsmöglichkeiten des Netzwerkes	143
10.1.2 Zusätzliche Aufnahmefunktion für Trainingsdaten	144
10.1.3 Ausweitung der Testfälle für die Anwendung <i>Smarter Home</i>	144
10.1.4 Hybride Lösung	144
10.1.5 Erweiterung des Lernbausteins Festigung	144
10.2 Zusammenfassung und Fazit	145
A Anforderungsdefinition	147
B Die beigelegte CD-ROM	150
B.1 Inhalt der CD-ROM	150
Abkürzungsverzeichnis	151
Glossar	153
Abbildungsverzeichnis	161
Tabellenverzeichnis	162
Literaturverzeichnis	163

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Masterarbeit unterstützt und motiviert haben.

Mein ganz besonderer Dank gilt daher zunächst meinen Betreuern der Abschlussarbeit, Prof. Dr. rer. nat. Janett Mohnke und Prof. Dipl.-Inf. Birgit Wilkes, die mich nicht nur während der Abschlussarbeit, sondern auch während des Studiums oft zurück auf den richtigen Pfad gebracht haben und mich immer wieder ermutigt und daran erinnert haben, das Wesentliche in vielerlei Hinsicht nicht aus den Augen zu verlieren, bis zum Schluss. Ich bin froh, dass sie eingewilligt haben, mich auch bei dem letzten Schritt meines Studiums zu begleiten.

Des Weiteren danke ich Frau Dr. Anke Renger, die mich bei der Konzeption der Lernbausteine für Schüler mit wertvollen Tipps und Ideen begeistert und unterstützt hat.

Ganz besonders danke ich auch meinen Eltern, Cornelia und Jörg-Werner Bliß, für ihren Glauben an mein Können und ihren Stolz, der mir immer ein großer Ansporn war.

Meinen Schwiegereltern, Birgit und Jörg Breßler, danke ich ebenfalls vielmals dafür, dass sie während des Studiums ständig in „Rufbereitschaft“ waren und stets eingesprungen sind, wenn es erforderlich wurde.

Zu guter Letzt möchte ich mich bei meiner Familie bedanken,
bei meinem Ehemann Tobias Breßler, der mich mit seiner Begeisterung für die Telematik
angesteckt und dazu motiviert hat, ebenfalls zu studieren und der für mich immer ein un-
verzichtbarer Rückhalt war und ist.

Und natürlich bei meinen wunderbaren Kindern Amélie und Julien,
denen ich staunend beim Aufwachsen zusehe und die mir mit ihrer Unbekümmertheit die
Leichtigkeit gegeben haben, diese Masterarbeit zu beenden.

Ich widme diese Arbeit meiner geliebten Oma, Helga Gertraud Bliß.

Wildau, den 13. April 2017

Janine Breßler

*Those who can imagine anything,
can create the impossible.*

(Alan Turing)

Hinweise zum Lesen der Arbeit

Nachfolgend werden die verwendeten Konventionen genannt, die innerhalb der Arbeit verwendet werden.

Die Angabe von Quellen in Textabschnitten erfolgt immer nach einem vollendeten Absatz in Form von zwei eckigen Klammern, die mit Hilfe einer Nummer auf eine Quelle im Quellenverzeichnis referenzieren.

Bei Abbildungen wird die Quellenangabe immer im Anschluss an die Bildbeschreibung eingefügt. Handelt es sich bei der Abbildung um eine selbst entworfene und erstellte Grafik, so wird keine Angabe zur Quelle im Beschreibungstext gegeben.

Zitate werden innerhalb der Arbeit durch „*Anführungszeichen und kursive Schrift*“ gekennzeichnet. Dabei markieren die Anführungszeichen jeweils den Anfang und das Ende eines Zitats. Auslassungen innerhalb eines Zitats werden durch drei Punkte in eckigen Klammern [...] dem Leser kenntlich gemacht. Nach jedem Zitat erfolgt eine Quellenangabe.

Eigennamen, Firmennamen und Fachtermini werden einheitlich *kursiv* geschrieben und dadurch ausgezeichnet.

Abkürzungen, die laut Duden anerkannt sind, werden innerhalb der Masterarbeit nicht ausgeschrieben und ebenfalls nicht im Abkürzungsverzeichnis genannt.

Fachspezifische Abkürzungen hingegen werden in der Arbeit bei erstmaliger Nennung einmalig ausgeschrieben und dahinter erfolgt in Klammern die dazugehörige Abkürzung. Im weiteren Verlauf der Arbeit wird lediglich die genannte Abkürzung weiterverwendet.

Bei sehr langen Bedeutungen abzukürzender Begriffe, die für den Lesefluss hinderlich sind, werden von Anfang an lediglich die Abkürzungen verwendet. Bei erstmaliger Nennung der Abkürzung wird jedoch die Bedeutung zu dieser Abkürzung in Form einer Fußnote gegeben. Fachspezifische Abkürzungen sind zusätzlich im Abkürzungsverzeichnis aufgeführt.

Neben dem Abkürzungsverzeichnis gibt es darüber hinaus ein Glossar, in dem erklärungsbedürftige Wörter mit Erläuterungen verzeichnet sind. Wörter, die sich im Glossar befinden, werden in der Arbeit mit dem hochgestellten Buchstaben g^g gekennzeichnet.

Kapitel 1

Einleitung

1.1 Motivation

„Softbank und Honda wollen sprechendes Auto bauen“ [24]

„Algorithmus sagt menschliche Verhaltensweisen voraus“ [24]

„Google will mit KI Musik machen“ [24]

„Warum Watson in die Sesamstraße zieht“ [24]

„Roboter sollen anhand von Bildern physikalische Gesetze lernen“ [28]

Die Schlagzeilen über immer neue Errungenschaften im Gebiet der *Künstlichen Intelligenz* (KI) reißen nicht ab. Die Frequenz solcher Meldungen lässt erahnen, welche Aktualität und welchen Stellenwert die KI als Teilgebiet der Informatik durch immer neue Weiterentwicklungen und Forschungsprojekte in den letzten Jahren und Monaten eingenommen hat.

Die KI ist nicht nur in der Forschung ein allgegenwärtiges Thema, einst als Schmutzkind der Informatik verschrien, hat sich die KI längst in unserem Alltag verstetigt und ist aus diesem nicht mehr wegzudenken. Wir finden sie in unseren Smartphones, nutzen ihre Vorteile bei Recherchen mit Hilfe von Suchmaschinen oder auch im Auto, indem wir uns von einem Ort zum anderen navigieren lassen oder auch das Auto bis zu einem gewissen Grad am Straßenverkehr autonom teilnehmen lassen.

Doch was ist genau Künstliche Intelligenz - Welche Faktoren genau die KI ausmachen und wodurch sie sich definiert, darüber kann nur selten eine eindeutige und einheitliche Aussage getroffen werden.

Als Königsdisziplin der KI wird die Fähigkeit verstanden, dass eine Maschine denken kann (Abbildung 1.1). Der Mathematiker *Alan Turing* hatte bereits 1950 einen Test entwickelt, mit dem es nach seiner Auffassung möglich ist zu überprüfen, ob eine Maschine hierzu in der Lage ist. Hierbei gilt der Test als bestanden, wenn ein Computer dazu fähig ist in einer Unterhaltung in Form eines Chats dem Menschen vorzutäuschen, ebenfalls ein Mensch zu sein. Im Juni 2014 gelang es erstmals in der Geschichte der KI mit Hilfe eines Supercomputers und einem entwickelten Algorithmus von *Vladimir Veselov* und *Eugene Demchenko* diesen Test zu bestehen. In diesem Fall unterhielten sich 33% der Testpersonen, und damit 3% mehr als gefordert, fünf Minuten lang über einen Textchat mit einem Computer ohne dies zu erkennen. [52]

Ein Meilenstein in der Forschung der KI. Bereits 2011 hatte der Computer *Watson* der Firma

Abbildung 1.1: Die Königsdisziplin der *Künstlichen Intelligenz* ist die Fähigkeit, eine Maschine denken zu lassen. [2]

IBM¹ die zwei erfolgreichsten Teilnehmer der *Jeopardy*²-Geschichte bezwungen [51] und der jüngste Erfolg konnte im März 2016 erzielt werden, als die von *Google* entwickelte Software *AlphaGo*³ in dem strategischen Brettspiel *Go* den aktuellen *Go*-Meister *Lee Sedol* in einem Turnier mit 4 zu 1 deutlich bezwang. Das Strategie-Spiel *Go* galt bis dato als zu komplex für Computer.

Auch in dieser Arbeit wird die KI einen hohen Stellenwert einnehmen. Neben der Frage wie die KI hinreichend definiert werden kann, wird sie einen entscheidenden Baustein in der Vernetzung der beiden Bereiche *Servicerobotik* und *Smart Home* einnehmen.

Die beiden Gebiete *Servicerobotik* und *Smart Home* und die Vernetzung dieser stellt einen wichtigen Forschungszweig im Studiengang der Telematik dar. Dabei sollen vor allem die Vorteile beider Welten miteinander vereint werden und einen wichtigen Mehrwert zur Unterstützung von älteren Menschen in den eigenen vier Wänden erzielen.

Als jüngster Erfolg in diesem Bereich kann die Bachelorarbeit mit dem Titel „*Humanoide Roboter als integrierte Assistenzsysteme in einer altersgerechten SmartHome Umgebung - Eruiierung von Möglichkeiten und Risiken*“ von *Tilman Bock* genannt werden, die mit dem *SmartHome Deutschland Award 2016* für die beste studentische Leistung ausgezeichnet wurde⁴.

Hierbei werden Sensordaten, die in einer smarten Küche integriert sind, an einen *Serviceroboter* gesendet oder von diesem abgerufen. In diesem Fall der Roboter *NAO* von der Firma *Aldebaran*⁵. Dieser wertet diese nach festgelegten Vorschriften aus und informiert die Umwelt bei Verletzung einer oder mehrerer Regeln. Einer dieser integrierten Sensoren stellt beispielsweise einen Durchflussmesser am Wasserhahn dar. Bei Erkennung eines besonders hohen bzw. langen Wasserverbrauchs wertet der *Serviceroboter* dies als eine mögliche unbeabsichtigte Aktion und informiert seinen Benutzer darüber, dass der Wasserhahn womöglich versehentlich angelassen wurde. Ähnliches erfolgt mit Hilfe eines Temperatursensors, der an

1 IBM Watson: <http://www-05.ibm.com/de/watson/>

2 Jeopardy Official Site: <https://www.jeopardy.com/>

3 AlphaGo: <https://deepmind.com/alpha-go>

4 SmartHome Deutschland, Die Sieger und Platzierten 2016:
<http://www.smarthome-deutschland.de/award/sieger-2016.html>

5 Aldebaran: <https://www.aldebaranrobotics.com>

einer Herdplatte montiert ist.

Eine weitere Abschlussarbeit mit dem Titel „*Konzeption und Evaluation von Szenarien zum Einsatz humanoider Roboter für das betreute Wohnen im Alter*“ von der Autorin *Janine Breßler*, die im Sommersemester 2014 entstanden ist, verfolgt die Idee, das Verhaltensmuster eines Menschen in seinen eigenen vier Wänden mit Hilfe der beiden Techniken *Servicerobotik* und *Smart Home* zu erlernen, um dabei Regelmäßigkeiten definieren zu können. Bei zunehmenden Abweichungen vom täglichen Ablauf des Menschen kann dies als ein Indikator für eine beginnende altersbedingte Krankheit, wie beispielsweise Demenz, genommen werden und entsprechend interveniert werden. Für solch ein Szenario ist es nötig, eine intelligente Anwendung zu entwickeln, die dazu in der Lage ist Verhaltensmuster zu erlernen und diese zu erkennen bzw. Abweichungen von diesen. [9]

Neben diesem Forschungszweig stellen die sogenannten *RoboticLabs* einen grundlegenden Baustein dar, welcher auf die Zukunftsarbeit des Studiengangs Telematik abzielt. Gerade Schüler der Sekundarstufe 2 (SEK 2) können hier in Form von Laboren und Praktika erste Eindrücke in die Welt der Telematik erhalten und ihr Geschick im Umgang mit Robotern und der Programmierung dieser unter Beweis stellen.

Das Feld der KI bildet noch ein eher wenig behandeltes Gebiet im Studiengang Telematik. In Verbindung mit den Techniken der *Servicerobotik* und des *Smart Home* bietet es jedoch viel Potential, um zu einem wichtigen Forschungszweig im Studiengang Telematik heranreifen zu können.

1.2 Zielsetzung

Das Ziel dieser Masterarbeit ist die Entwicklung von Lernbausteinen, die für den praktischen Einsatz in Schülerlaboren, Seminararbeiten sowie Abschlussarbeiten im Studiengang Telematik eingesetzt, angewandt und weiterentwickelt werden können.

Dabei soll eine Infrastruktur konzeptioniert und implementiert werden, die es erlaubt, Komponenten aus dem Bereich *Smart Home* und *Servicerobotik* über eine intelligente Verarbeitungseinheit zu vernetzen. Hierfür wird eine Applikation in Form einer dezentralen Speicher- und Verarbeitungseinheit entwickelt, die mit Hilfe von Segmenten der KI und erhaltenen Sensordaten dazu in der Lage ist, Verhaltensmuster zu lernen. Ein Beispiel stellt die tägliche Nutzung eines elektronischen Haushaltsgerätes dar. Neben dem Lernen von Verhaltensmustern soll die Anwendung darüber hinaus dazu in der Lage sein, Unregelmäßigkeiten anhand von Sensordaten zu erkennen und diese generierte Information über einen definierten Kanal der Umwelt zur Verfügung zu stellen. Dieser Informationskanal bietet die Voraussetzungen, um in weiteren Entwicklungsschritten im Zuge von zukünftigen Abschlussarbeiten o.Ä. die Informationen mit Hilfe von *Servicerobotern* nutzen zu können. Das Generieren von Sensordaten hingegen eignet sich gut für den praktischen Einsatz in Schülerlaboren, bei denen das Erzeugen der Daten und die Übertragung dieser in einem passenden Format an die intelligente Verarbeitungseinheit mögliche Bausteine darstellen.

Letztlich soll mit der Umsetzung dieser Masterarbeit eine Vernetzung der zwei in Abschnitt 1.1 genannten Teilbereiche *Servicerobotik* und *Smart Home* erfolgen mit der Intention weitere Forschungsprojekte und -arbeiten in Form von Lernbausteinen generieren zu können. Diese sollen auf der einen Seite eine Integration der Schülerlabore in die Telematikprojekte

ermöglichen sowie auf der anderen Seite die Möglichkeit für die Erlangung und Festigung von zusätzlichem Fachwissen im Bereich der KI sowie der *Servicerobotik* und *Smart Home* bieten.

1.3 Abgrenzung

Die Masterarbeit beschäftigt sich mit der Konzeption und der Umsetzung von Lernbausteinen in Verbindung mit einer dezentralen intelligenten Verarbeitungseinheit. Dabei umfasst die Implementierung das Grundgerüst zur Vernetzung der beiden Techniken *Servicerobotik* und *Smart Home* in Form einer intelligenten Verarbeitungseinheit oder auch Middleware. Die Umsetzung von Komponenten aus den beiden Bereichen *Servicerobotik* und *Smart Home* hingegen ist nicht Bestandteil dieser Ausarbeitung.

1.4 Gliederung der Arbeit

Die Gliederung der Masterarbeit spiegelt die Herangehensweise an das gesamte Projekt wider und soll im Folgenden erläutert werden.

Das Thema der Masterarbeit vereint die drei großen Bereiche *Servicerobotik*, *Smart Home* und *Künstliche Intelligenz*.

Aufgrund dessen setzt sich der erste Teil der Ausarbeitung zunächst thematisch mit den Grundlagen der jeweiligen Technik auseinander. Dabei liegt der Hauptfokus auf der KI, da in diesem Bereich auch die spätere Umsetzung erfolgen soll. Im Verlauf der Arbeit soll dafür ein geeignetes Werkzeug der KI ausgewählt werden. Aufgrund dessen wird in diesem Abschnitt auf eine Auswahl solcher Werkzeuge eingegangen, um so einen ersten Überblick über die Vielfalt dieses Bereiches zu vermitteln.

Nach der Abhandlung der Grundlagen wird in Form einer Anforderungsanalyse zunächst, durch die Definition der Zielgruppe, des Szenarios und der zu entwickelnden Komponenten, ein Überblick gegeben, was im Verlauf der Arbeit erreicht werden soll. Anschließend wird mit Hilfe von Use-Case-Diagrammen ermittelt, über welche Funktionen die intelligente Verarbeitungseinheit verfügen soll und im Folgenden die Anforderungen daraus abgeleitet und spezifiziert.

Im nächsten Schritt erfolgt die Konzeption der intelligenten Verarbeitungseinheit. Dabei werden zunächst verwandte Arbeiten untersucht, um dabei interessante Ansätze und Erkenntnisse für die eigene Entwicklung adaptieren und nutzen zu können.

Im Anschluss daran werden die einzelnen Anwendungsmodi der intelligenten Verarbeitungseinheit konzipiert und die dafür geeigneten Lösungskomponenten evaluiert, verglichen sowie ausgewählt.

Bei der anschließenden Konzeption des Lernbaukastens und der Entwicklungsbausteine wird zunächst der Lernprozess und eine geeignete Lehrstrategie untersucht, bevor im Anschluss darauf aufbauend die einzelnen Bausteine des Lernbaukastens und die Entwicklungsbausteine definiert werden.

Im weiteren Verlauf erfolgt die Beschreibung der Umsetzung der konzipierten Komponenten, dabei wird bei der intelligenten Verarbeitungseinheit hauptsächlich auf die Schwerpunkte bei

der Umsetzung, die wichtigsten Services sowie den Abweichungen vom Konzept eingegangen. Bei der Umsetzung des Lernbaukastens hingegen stehen vor allem der erstellte Prototyp, die verwendeten Techniken sowie die erstellten Präsentationen und Tutorials im Vordergrund.

Nach der Beschreibung der Umsetzungsphase werden im nachfolgenden Kapitel in Form einer Testphase die Ergebnisse der ersten Tests der Gesamtanwendung vorgestellt und ausgewertet.

Schließlich erfolgt im letzten Kapitel eine Abschlussbetrachtung, in der die Ergebnisse der Arbeit noch einmal zusammengefasst werden und ein Ausblick für zukünftige Weiterentwicklungen gegeben wird.

Kapitel 2

Künstliche Intelligenz

In diesem Kapitel wird zunächst eine Definition für die KI gegeben, die als Richtlinie für die nachfolgenden Ausarbeitungen dient. Anschließend erfolgt in Abschnitt 2.2 eine Einführung in die KI in Form eines historischen Abrisses, der den Werdegang der KI mit seinen Errungenschaften und Niederlagen erzählt. Damit soll ein guter Einstieg in diese komplexe Thematik ermöglicht werden.

Darauf aufbauend wird in Abschnitt 2.3 ein kurzer und grundsätzlicher Überblick zu dem heutigen Repertoire an Werkzeugen der KI gegeben, die sich in die zwei Kategorien der *Symbolverarbeitung*^g und des *Konnektionismus*^g klassifizieren lassen. Abschließend erfolgt, ohne Anspruch auf Vollständigkeit, ein Einblick in die Anwendungsbereiche der KI.

2.1 Was ist Künstliche Intelligenz - Eine Definition

Das Gebiet der *Künstlichen Intelligenz* versteht sich als ein Teilbereich der Informatik. Selten gab es ein anderes Gebiet in der Informatik, welches die Kreise der Wissenschaft aber auch die der Gesellschaft derart polarisiert hat.

In ihrem Buch „*Künstliche Intelligenz*“ [12] begründen *Uwe Lämmel* und *Jürgen Cleve* [12] dies einerseits durch die englische Wortwahl des Ausdrucks (Artificial Intelligence) und andererseits mit der Übersetzung ins Deutsche, denn der Begriff *Künstliche Intelligenz* wirft unweigerlich die Frage auf, ob Maschinen dazu in der Lage sind zu denken. Eine Übersetzung wie *Gekünstelte Intelligenz* oder auch *Synthetische Intelligenz* hätte möglicher Weise zu weniger emotionsgeladenen Debatten geführt, ob dies jedoch als positiv oder negativ einzuschätzen ist, vermag jeder für sich selbst zu entscheiden.

Es gibt eine ganze Reihe von Ansätzen und Standpunkten, unter denen die KI definiert werden kann. In dieser Arbeit wird sich jedoch auf die ingenieurtechnische Betrachtungsweise beschränkt. In ihr wird unter der KI nicht das Messen mit dem Menschen sondern vielmehr das Nachbilden von menschlichen Fähigkeiten verstanden.

Hierunter fallen die Fertigkeiten zu denken, anhand von Beispielen zu lernen oder auch gegebene Problemstellungen eigenständig lösen zu können und damit die Leistung eines Rechners zu erhöhen.

Abschließend soll für das Feld der KI die folgende Definition als Einstimmung auf die Thematik und als Richtlinie für die nachfolgenden Ausarbeitungen dienen.

Demnach ist die *Künstliche Intelligenz* ein:

„Teilgebiet der Informatik, welches versucht,
menschliche Vorgehensweisen der Problemlösung auf Computern nachzubilden,
um auf diesem Wege neue oder effizientere Aufgabenlösungen zu erreichen.“ [12, S. 13]

2.2 Die Geschichte der KI - Eine Einführung

Die nachfolgenden Ausführungen beschreiben die geschichtlichen Anfänge und Meilensteine in der KI, ohne dabei einen Anspruch auf Vollständigkeit zu haben.

Die KI hat bis zu dem heutigen Tag einen langen Werdegang hinter sich. Bereits in den dreißiger Jahren des 20. Jahrhunderts legten die Mathematiker *Kurt Gödel* und *Alonzo Church* sowie der Logiker *Alan Turing* den Grundstein für die Logik und die theoretische Informatik. [18, S. 6ff]

1931 erschütterte der Österreicher *Kurt Gödel* die mathematische Welt mit seinem *Unvollständigkeitssatz*. Dieser Satz sagt aus, dass jedes widerspruchsfreie System, welches dazu in der Lage ist einfache arithmetische Aussagen zu treffen, immer auch wahre Aussagen enthält, die sich weder beweisen noch widerlegen lassen. Damit zeigte er die Grenzen der Mathematik und damit auch die der Computer auf.

Für die Computerwelt bedeutete dies, dass es nie eine Software geben kann, die dazu in der Lage ist, die Fehlerlosigkeit beliebiger Programme zu beweisen. [18, S. 6ff]

Bezugnehmend auf die heutige KI hatte dies vor allem Auswirkungen auf den Bereich der Symbol verarbeitenden KI, deren Ansatz darin besteht, Wissen in Form von Symbolen und logischen Konzepten darzustellen, um daraus Schlussfolgerungen ziehen zu können. *Prädikatenlogiken* höherer Stufen, wie zum Beispiel die *Prädikatenlogik zweiter Stufe*^g, gelten seitdem als unvollständig. Aber *Gödel* konnte auch beweisen, dass die *Aussagenlogik* (Abschnitt 2.3.1) und die *Prädikatenlogik* (Abschnitt 2.3.2) vollständig sind und damit jede darstellbare Aussage in diesen Systemen auch beweisbar bzw. widerlegbar ist. [18, S. 6ff]

Ebenfalls in diese Epoche fällt der Beweis von *Alan Turing* über die *Unentscheidbarkeit des Halteproblems*. Diese Thematik beschäftigt sich mit der Informatik und der Fragestellung, ob Algorithmen bei beliebiger Eingabe terminieren. Dabei bewies Turing mit Hilfe eines *Turing-Automaten*^g, dass es Probleme gibt, die sich nicht mittels Algorithmus automatisiert lösen lassen. Das heißt, es kann keinen Algorithmus geben, der bestimmen kann, ob andere beliebige Algorithmen mit beliebigen Eingaben terminieren oder nicht. [18, S. 6ff]

In den vierziger Jahren wurden erstmals durch *McCulloch*, *Pitts* und *Hebb* sogenannte *künstliche neuronale Netze* (KNN) konzipiert. Zu diesem Zeitpunkt stellte jedoch das Fehlen von leistungsfähigen Rechnersystemen den Flaschenhals zur Simulation von einfachen Gehirnen mittels KNN dar.

Die KI hat als eigenständige Wissenschaft erst seit Mitte des zwanzigsten Jahrhundert Bestand. Als die Geburtsstunde der KI gilt dabei die legendäre Dartmouth-Konferenz aus dem Jahr 1956. Auf dieser Konferenz stellte *McCarthy* erstmals eine Programmiersprache vor,

mit der es möglich war, Symbol verarbeitende KI-Systeme zu entwickeln. Dabei handelte es sich um die Programmiersprache *LISP*¹. Ebenfalls auf dieser Konferenz präsentierten die Computerwissenschaftler *Allen Newell* und *Herbert Simon* der Welt den ersten automatischen Theorembeweiser² *Logic Theorist*.

Es folgten die *Resolution*³ für die *Prädikatenlogik* und seit den siebziger Jahren gibt es *PROLOG*⁴, eine Programmiersprache aus Europa. [18, S. 6ff]

Bis Mitte der achtziger Jahre schwamm die Logik auf einer Welle der Euphorie. Befeuert durch immer neue Errungenschaften in der *Symbolverarbeitung*, wie beispielsweise die *Fifth Generation Computer Systems*⁵ aus Japan oder das ESPRIT⁶-Programm aus Europa. [18, S. 6ff]

Ende der achtziger Jahre ebte die Welle der Euphorie ab, da die erwarteten Erfolge in der Wirtschaft ausblieben. Die Folge waren massive Einschränkungen der Mittel in den Forschungsbereichen der Symbol verarbeitenden KI. [18, S. 6ff]

Aber die Stagnation währte nicht lang, denn mit Hilfe der neuen leistungsstarken Rechner lebte wiederum der konnektionistische Ansatz auf und konnte stark auftrumpfen. Informatikern, Physikern und Kognitionswissenschaftlern gelang es zu zeigen, dass ihre mathematisch modellierten KNN mit Hilfe von Trainingsdaten dazu in der Lage waren Aufgaben zu erlernen, die vorher nur mit sehr großem Aufwand zu programmieren waren. Gerade in der Bildverarbeitung, wie etwa der Erkennung von Personen auf Porträtfotos oder die Handschrifterkennung und das Vorlesen von Texten durch das System *Nettalk* (1996), konnten enorme Erfolge gefeiert werden. Der *Konnektionismus* stellte fortan ein neues starkes Teilgebiet der *Künstlichen Intelligenz* dar.

Aber auf jeden Anstieg folgt meist auch ein Tal und nach einigen erfolgreichen Jahren mussten auch die Verfechter des *Konnektionismus* sich eingestehen, dass die KNN ihre Grenzen haben. So war es bis dato nicht möglich, die Netze mit anderen Werkzeugen der KI zu verknüpfen, wie beispielsweise mit logischen Regeln oder mit menschlichem Expertenwissen. [18, S. 6ff]

Die Expertenwelt suchte nach neuen Wegen mit denen man die beiden Teildisziplinen der KNN und die der Logik und deren jeweiligen Vorteile miteinander verknüpfen könnte.

Eine Möglichkeit dafür bot das sogenannte *prohabilitische Schließen*, bei dem mit Hilfe von bedingten Wahrscheinlichkeiten aussagenlogische Formeln verwendet werden können (Ab-

1 LISP: List Processing

2 Theorembeweiser:

Führen aufeinander aufbauende Theoreme automatisch oder auch interaktiv aus bzw. arbeiten diese ab. [68, S. 12]

3 Resolution:

Bei der Resolution wird die Gültigkeit einer Formel γ mit Hilfe eines Widerspruchsbe-
weises bewiesen, indem $!\gamma$ zu einem Widerspruch geführt wird. [16]

4 PROLOG: Programmieren in Logik

5 Fifth Generation Computer Systems:

Ein japanisches Projekt, das in den späten 1980ern fertiggestellt wurde. Ziel dieses Pro-
jektes war es, leistungsstarke Supercomputersysteme zu entwickeln, die den zukünftigen
Entwicklungen und Anforderungen der Künstlichen Intelligenz gewappnet sind. [48]

6 European Strategic Programme for Information Technology:

Programm der Europäischen Forschungs- und Technologiepolitik, das in den 1980er
Jahren begann und ein Forschungsnetzwerk in der Informationstechnologie aufbaute
mit der Beteiligung von 12 große europäischen Elektronikkonzernen. [11]

schnitt 2.3.3). Mit Hilfe dieses Ansatzes können heutzutage viele erfolgreiche Diagnose- und Expertensysteme^g konstruiert und eingesetzt werden. [18, S. 6ff]

Eine andere erfolgreiche Idee zur Zusammenführung von Logik und *Konnektionismus* bilden sogenannte hybride Systeme, bei denen beispielsweise KNN eingesetzt werden, um Heuristiken zu erlernen, die wiederum zur Eingrenzung der Suchräume bei der Beweissuche eingesetzt werden können.

Auch das Erlernen von Entscheidungsbäumen, die wiederum aussagenlogische Konzepte repräsentieren, bieten eine gute Kombination von *Konnektionismus* und Logik. Sie gelten heutzutage als die beliebtesten maschinellen Lernverfahren und werden für Expertensysteme genutzt, ein Beispiel hierfür ist CART^g. [18, S. 6ff]

Das *Data Mining* darf sich seit 1990 als Teil der KI-Familie glücklich schätzen. Es wird vor allem für das statistische Auswerten von Daten innerhalb von Datenbanken und das Extrahieren von explizitem Wissen aus einer großen Menge an Daten genutzt und ist zu einem unverzichtbaren Werkzeug in der KI geworden. [18, S. 6ff]

Eine weitere Königsdisziplin der KI bilden sogenannte verteilte, lernende und autonom agierende Agenten. Dabei sollen Softwareagenten oder auch Roboter dazu in der Lage sein, in Abstimmung und Kooperation mit anderen Agenten in einem Team zu agieren, um damit Probleme zu lösen, die durch einen einzelnen Agenten nicht lösbar wären. Dieser Ansatz greift auf ein Phänomen der Natur zurück, nämlich auf das der Ameise und ihrem Ameisenvolk. Als Gruppengemeinschaft sind die Ameisen in der Lage architektonisch komplexe Gebäude zu errichten. Ein einziges Ameisenindividuum hingegen verfügt jedoch über keinerlei globales Verständnis über die Errichtung von komplexen Bauten. [18, S. 6ff]

In den letzten Jahren hat sich das verstärkte Lernen, auch unter dem Begriff *Deep Learning*^g bekannt, als spezielle Form der KNN einen Namen gemacht. Der jüngste Erfolg ist das System *AlphaGo* von *Google DeepMind*, das in dem strategischen Brettspiel *Go* Anfang 2016 zunächst den Europameister mit 5:0 und anschließend im März den bis dato weltbesten Go-Spieler *Lee Sedol* mit 4:1 bezwang. [18, S. 6ff]

In der nachfolgenden Abbildung 2.1 sind der zeitliche Verlauf der Geschichte und die Meilensteine der KI nochmals graphisch zusammengefasst.

Die Y-Achse unterteilt in numerische¹ und symbolische Verfahren zur Repräsentation der KI. Die Dicke der Balken stellen die Verbreitung der jeweiligen Techniken in den Anwendungen zu dem jeweiligen Zeitpunkt dar. Gut ist auch zu erkennen, dass ab den siebziger Jahren zusätzlich sogenannte hybride Systeme entstanden, bei denen Elemente einer Technik in anderen Systemen verwendet wurden. Gekennzeichnet ist dies durch die Überführung des Balkens von einer Technik in den einer anderen, gekennzeichnet durch die grüne Verbindungslinie. [18, S. 6ff]

¹ Zu den numerischen Verfahren zählen die *Vorhersage*, die *Klassifizierung* und das *maschinelle Lernen*^g

Abbildung 2.1: Die Geschichte und Meilensteine der Künstlichen Intelligenz auf einen Blick. [18, Abb. 1.3]

2.3 KI heute - Der Stand der Technik

In der Gegenwart angekommen ist die *Künstliche Intelligenz* längst aus ihren Kinderschuhen herausgewachsen und hat sich als ein hochgradig interdisziplinäres Teilgebiet der Informatik etabliert.

Es gibt eine Palette an verschiedenen Werkzeugen, die für unterschiedliche Aufgabenbereiche eingesetzt werden können. In Form von Software-Bibliotheken sind die meisten dieser bereits für die große Masse an Entwicklern zugänglich. Die Schwierigkeit besteht jedoch darin, das passende Werkzeug zur vorliegenden Aufgabenstellung zu finden.

Dabei gibt es zwei Leitphilosophien in der KI, zwischen denen grundlegend unterschieden wird: die *Symbolverarbeitung* und der *Konnektionismus*.

Die Techniken der Symbol verarbeitenden KI, die auch unter dem Begriff der *klassischen KI* bekannt sind, eignen sich vor allem für die schnelle Verarbeitung großer Datenmengen oder der Speicherung und Auswertung einer riesigen Anzahl an Informationshäppchen mit Hilfe von Superrechnern.

Ein berühmtes Beispiel für den Einsatz von symbolverarbeitenden Algorithmen ist der Schachcomputer *Deep Blue*¹ von der Firma *IBM*, der im Jahr 1997 den damaligen Schachweltmeister *Garri Kasparow* erstmalig in einem Schachturnier unter Wettbewerbsbedingungen bezwang. In sogenannten Expertensystemen[§] finden Symbol verarbeitende Algorithmen ihre größte

1 Deep Blue: <http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/>

Anwendung. In den nachfolgenden Abschnitten (2.3.1, 2.3.2, 2.3.3 und 2.3.4) wird eine kurze Übersicht zu den wichtigsten Bestandteilen eines wissensbasierten Systems gegeben. [12, S. 28ff]

Im letzten Abschnitt (2.3.5) wird eine Einführung zu dem konnektionistischen Ansatz gegeben. Dieser Ansatz setzt auf dem Ursprung der menschlichen Intelligenz an. Die Intelligenz des Menschen wird auf die physikalischen Bausteine des Gehirns, den Nervenzellen oder auch Neuronen, und deren komplexen Verschaltung zurückgeführt. Der *Konnektionismus* bildet diese KNN in Form von mathematischen Modellen auf dem Computer ab. [12]

Wie in vielen anderen Bereichen, besteht auch hier die Möglichkeit der Kombination beider Verfahren. Dies spielt bspw. in der Robotik eine große Rolle. Auf die Kombination beider Verfahren soll jedoch in den folgenden Abschnitten nicht näher eingegangen werden, denn dies wäre an dieser Stelle zu umfangreich. [12]

2.3.1 Aussagenlogik

Alle Menschen sind sterblich.

Sokrates ist ein Mensch.

Also ist Sokrates sterblich.

Unter der Logik wird grundsätzlich die Fähigkeit verstanden, aufgrund von schlüssigem und folgerichtigem Denken, wahre Schlussfolgerungen ziehen zu können.

Die *Aussagenlogik* versteht sich als ein Teilgebiet der Logik und beschäftigt sich mit der Repräsentation von Wissen und *Inferenzmechanismen*¹, mit denen es in einem wissensbasierten System möglich ist, aus dem vorhandenen Wissen richtige Schlüsse zu ziehen. Zu wissensbasierten Systemen zählen beispielsweise Expertensysteme. [12, S. 33ff]

Oft wird die *Aussagenlogik* auch als eine grundlegende Logik beschrieben, die als Teillo-
gik in anderen Logiken enthalten ist.

Hierfür verfügt die *Aussagenlogik* über eine gekünstelte Sprache, mit der es möglich ist, die natürliche Sprache in Form von elementaren bzw. atomaren Aussagen und logischen Operatoren abzubilden und anhand daran Wissen aufzubauen und Schlussfolgerungen ziehen zu können. [12, S. 33ff]

Die einzelnen Elemente dieser künstlichen Sprache werden in Form von Symbolen repräsentiert, die das Alphabet darstellen. Dabei stehen große Buchstaben *A* für Teilaussagen und beispielsweise ein logisches Und $\wedge$ für einen Junktoren in der Sprache.

Bei den Teilaussagen in der *Aussagenlogik* handelt es sich um zweiwertige Aussagen, die nur den Wert *wahr* **oder** *falsch* annehmen können.

Für die Verknüpfung von Aussagen stehen die in Tabelle 2.1 aufgezeigten logischen Operatoren zur Verfügung. Neben der Auflistung der einzelnen Junktoren sind in der rechten Spalte zusätzlich Satzmuster gegeben, die zeigen, welches Symbol für die Abbildung der Problemstellung genutzt werden kann. [12, S. 33ff]

¹ Inferenzmechanismus: Bestandteil eines wissensbasierten Systems. Dieser hat die Aufgabe Schlussfolgerungen aus der Wissensbasis abzuleiten, um somit zur Lösung des zu bearbeitenden Problems beizutragen.[61]

Symbol	Bezeichnung	Beispiel	Aussage ist wahr, falls ...	Satzmuster
$\wedge$	Und	$X \wedge Y$	X und Y wahr sind	... und ...; sowohl ... als auch ...
$\vee$	Oder	$X \vee Y$	X oder Y wahr sind	... oder ...
$\neg$	Negation	$\neg X$	X falsch ist	es stimmt nicht, dass ...
$\rightarrow$	Implikation	$X \rightarrow Y$	Y wahr ist, falls X wahr ist	wenn X dann Y; aus X folgt Y
$\leftrightarrow$	Äquivalenz	$X \leftrightarrow Y$	X genau dann wahr ist, wenn Y wahr ist	X genau dann, wenn Y

Tabelle 2.1: Aussagenlogische Verknüpfungen in Kombination mit der Alltagssprache.

Mit Hilfe der Symbolik in der *Aussagenlogik* lassen sich atomare Aussagen wie *die Straße ist nass* oder *es regnet* zu neuen Aussagen verknüpfen und als Regeln im System hinterlegen:

Wenn es regnet ist die Straße nass.

Oder symbolisch in der *Aussagenlogik* ausgedrückt:

$es\ regnet \rightarrow die\ Straße\ ist\ nass$

Der Wahrheitswert einer Aussage, die aus einzelnen atomaren Teilaussagen besteht, kann immer durch die Wahrheitswerte und anhand der jeweiligen Verknüpfungen der einzelnen Teilaussagen bestimmt werden. Dies kann beispielsweise mit Hilfe von Wahrheitstabellen erfolgen. Für das Beweisen von Hypothesen anhand von bestehendem Wissen in der *Aussagenlogik* wird u.a. die sogenannte *Resolution* angewandt. Dabei handelt es sich um ein Beweisverfahren, das das automatische Beweisen auf einem Rechner ermöglicht. [12, S. 33ff]

Anwendung findet die *Aussagenlogik* generell in der Digitaltechnik, z.B. in der Verifikation von digitalen Schaltungen. Speziell in der *Künstlichen Intelligenz* wird die *Aussagenlogik* bei einfachen Expertensystemen⁸ eingesetzt, wobei hier die Einschränkung besteht, dass ausschließlich diskrete Variablenwerte verwendet werden dürfen, die wenig unterschiedliche Werte annehmen und darüber hinaus über keine Querbeziehungen zwischen den Variablen verfügen. Für komplexere Zusammenhänge wird auf die *Prädikatenlogik* zurückgegriffen. [18]

Aussagenlogik ist deklarativ, das heißt die Syntaxelemente entsprechen Fakten. Zusätzlich erlaubt die *Aussagenlogik* partielle, disjunkte und negative Aussagen, im Gegensatz zu vielen anderen Datenstrukturen und Datenbanken. [12, S. 33ff]

Aufgrund der Einschränkungen bei den Wahrheitswerten und den logischen Operatoren, ist die *Aussagenlogik* jedoch nur begrenzt ausdrucksstark. Das Abbilden der Aussage *Jede natürliche Zahl ist entweder gerade oder ungerade* würde beispielsweise eine Formel für *jede* Zahl erfordern und ist somit mit der *Aussagenlogik* nicht praktikabel umsetzbar. Trotz alledem bildet die *Aussagenlogik* das Fundament vieler anderer Logiksprachen und ist daher unverzichtbar. [12, S. 33ff]

2.3.2 Prädikatenlogik erster Stufe

Die *Prädikatenlogik* der ersten Stufe setzt auf der *Aussagenlogik* auf und erweitert diese um zusätzliche Elemente:

1. Für Elementaraussagen ist die Zuordnung von Objekten möglich, für diese Objekte gelten dann die Eigenschaft aus der Elementaraussage oder auch nicht.
2. An Stelle von Objekten können ebenfalls Variablen verwendet werden, diese müssen dann existenz- oder all-quantifiziert sein.

Durch die Erweiterungen ist die *Prädikatenlogik* ausdrucksstärker und bietet die Möglichkeit den Eigenschaften feste Objekte oder Variablen zuzuweisen. In Tabelle 2.2 sind die aussagenlogischen Verknüpfungsinstrumente der Aussagen- und der *Prädikatenlogik* dargestellt. [12, S. 52ff]

Bezeichnung	Aussagenlogik	Prädikatenlogik
Elementare Aussagen	$zahl_4$	$zahl(4)$
$\wedge$ Und	$x \wedge y$	$x \wedge y$
$\vee$ Oder	$x \vee y$	$x \vee y$
$\neg$ Negation	$\neg x$	$\neg x$
$\rightarrow$ Implikation	$x \rightarrow y$	$x \rightarrow y$
$\leftrightarrow$ Äquivalenz	$x \leftrightarrow y$	$x \leftrightarrow y$
Existenzquantifizierung	-	$\exists z zahl(z)$
Allquantifizierung	-	$\forall z zahl(z)$

Tabelle 2.2: Eine Übersicht zu den möglichen Verknüpfungen in der Aussagen- und *Prädikatenlogik*. [12, Tab. 2.11]

Somit ist die Aussage aus Abschnitt 2.3.1 *Jede natürliche Zahl ist entweder gerade oder ungerade* mit der *Prädikatenlogik* sehr komfortabel darstellbar:

$$\forall x \in \mathbb{N} ((gerade(x) \vee ungerade(x)))$$

Die Instrumente der Aussagen- und *Prädikatenlogik* ermöglichen das Aufbauen einer Wissensbasis, in der Fakten, Relationen aber auch Regeln (Beispiel Implikation) abgebildet werden können. [12, S. 33ff]

2.3.3 Schließen mit Unsicherheit

„Insofern sich die Gesetze der Mathematik auf die Wirklichkeit beziehen, sind sie nicht sicher.

Und insofern sie sich sicher sind, beziehen sie sich nicht auf die Wirklichkeit.“

(Albert Einstein)

Viele tägliche Situationen sind von Unsicherheit und Ungenauigkeit geprägt. Darüber hinaus sind Worte und Ausdrücke als Elemente der natürlichen Sprache meist vage.

Die bisher vorgestellte zweiwertige Logik erlaubt lediglich das Abbilden und Modellieren von Sachverhalten, die entweder *wahr* oder *falsch* sein können. Es ist jedoch nicht möglich auszudrücken, dass man eine Aussage für wahrscheinlich hält.

Für viele Problemstellungen und Fragen im Alltag ist die zweiwertige Logik damit nicht ausdrucksstark genug, um diese geeignet darzustellen zu können. [12, S. 91ff]

Zwei gängige Verfahren in dem Bereich für das Schlussfolgern mit Unsicherheiten sind das *prohabilitische Schließen* und die *Fuzzy-Logik*.

Bei dem *prohabilitischen Schließen* bildet die Basis das Verwenden von Wahrscheinlichkeiten. Diese Wahrscheinlichkeiten repräsentieren dabei den Grad der Unsicherheit für eine Aussage. Die Aussagen

es schneit morgen
oder *der Kurs des US-Dollars wird steigen*

sind unsichere Aussagen, die für eine adäquate Repräsentation mit Wahrscheinlichkeiten versehen werden müssen. [12, S. 91ff]

Abhilfe für die Modellierung von vagen Aussagen bietet die sogenannte Fuzzy-Logik. Das Wort *Fuzzy* kommt aus dem Englischen und bedeutet übersetzt etwa *flaumig* oder *weich* und mit Blickrichtung auf die Logik passt wohl das Adjektiv *unscharf* am besten. Unscharfe Aussagen sind Aussagen, wie beispielsweise

der Mann ist groß
oder *das Essen ist heiß*

Mit der Fuzzy-Logik können diese vagen oder auch unscharfen Aussagen semantisch interpretiert und mathematisch dargestellt werden, um dadurch Schlussfolgerungen über diese Aussagen treffen zu können. Dies wird ermöglicht, indem die diskreten Wahrheitswerte 0 und 1 aus der herkömmlichen Logik durch einen stetigen Bereich in Form von Intervallen ersetzt werden. [12, S. 91ff]

2.3.4 Suchalgorithmen zur Problemlösung

*„Die Lösung ist immer einfach,
man muss sie nur finden.“*

(Alexander Solschenizyn)

Eine ganze Reihe an alltäglichen Problemen der Informatik lassen sich auch als Suchprobleme darstellen und mit dem Einsatz von passenden Suchalgorithmen lösen.

Unter den Suchalgorithmen in der Informatik werden Verfahren verstanden, die dazu in der Lage sind, innerhalb eines Suchraums Muster oder Objekte zu finden, die eine bestimmte Eigenschaft aufweisen. [12, S. 114ff]

Gerade für KI-spezifische Problemstellungen stellen die Suchverfahren ein essentielles Werkzeug dar. Beispielsweise muss ein *Inferenzsystem*^g anhand der erhaltenen Eingangsinforma-

tionen möglichst effizient eine Schlussfolgerung ziehen können, sodass das Expertensystem anschließend damit weiterarbeiten kann.

Hierfür muss das *Inferenzsystem* dazu in der Lage sein, die enthaltenen Fakten und Regeln in einer annehmbaren Zeit in der Wissensbasis zu finden. Um dies erfüllen zu können, werden Suchalgorithmen verwendet. [12, S. 114ff]

Eine Problemlösung mittels Suche ist immer dann möglich, wenn der Zustandsraum eines Problems bekannt ist und sich mit Hilfe eines Graphen darstellen lässt.

Dabei definiert sich der Zustandsraum anhand

- des Ausgangszustandes
- der möglichen Zielzustände oder eines Zielzustandes
- und wohldefinierter Zustandsübergänge (Aktionen)

Ein Graph G besteht aus einer Menge N von Knoten und einer Menge E von Kanten:

$$G = (N, E)$$

Die Menge E besteht aus Paaren von Knoten:

$$E \subseteq N \times N$$

In einem Graphen repräsentieren die Knoten die möglichen Zustände eines Problems, inklusive dem Ausgangs- und dem Zielzustand. Wenn zwei Knoten durch eine Kante verbunden sind, dann bedeutet dies, dass durch das Ausführen einer entsprechenden Aktion von dem einen Zustand (Knoten A) in den anderen Zustand (Knoten B) übergegangen wird. An der Kante ist entsprechend die Aktion vermerkt, die den Zustandswechsel herbeiführt. Handelt es sich um gerichtete Kanten (mit einem Pfeil gekennzeichnet) ist der Zustandswechsel nur in diese Richtung möglich. Anhand des Graphen kann ein Suchbaum gebildet werden. Dieser wird ausgehend von dem Startzustand durch das rekursive Anwenden der möglichen Aktionen in einem Zustand gebildet. [12, S. 114ff]

Das nachfolgende Beispiel ist wohl jedem aus dem alltäglichen Leben aber spätestens seit dem Film „Stirb langsam - Jetzt erst recht“¹ in dieser oder ähnlicher Form bekannt und zeigt, wie ein Suchproblem mit Hilfe eines Graphen und eines Suchbaumes dargestellt werden kann, sodass ein Suchalgorithmus darauf angewandt werden kann. Dieses Szenario spielt sich jedoch nicht an einem Brunnen in New York ab und bei Nichtgelingen droht auch nicht die Explosion einer Bombe.

Vielmehr spielt sich diese Situation in einer heimischen Küche auf einem Campingplatz irgendwo an der schönen Ostsee ab. Dies impliziert jedoch nicht, dass die Situation weniger brisant ist:

Um die Mittagszeit soll das Mittagessen zubereitet werden. Dabei steht Buchstabensuppe auf dem Programm und die Kochanleitung verlangt eine Zubereitung mit exakt zwei Litern Wasser. Zur Verfügung stehen dem Koch oder der Köchin jedoch nur zwei Kanister mit jeweils 3 und 4 Litern. Wie ist es nun durch Auffüllen, Ausschütten und Umfüllen mit

¹ *Zeus* und *McClane* bekommen von *Simon* an einem Brunnen die Aufgabe mit einem 3- und einem 5-Gallonen-Kanister genau 4 Gallonen abzumessen.

den zwei Kanistern trotzdem möglich eine genießbare Buchstabensuppe (nicht zu dickflüssig und nicht zu verwässert) zuzubereiten?

Der Ausgangszustand und der Zielzustand sind wie folgt definiert:

Ausgangszustand: Beide Kanister sind leer

Zielzustand: Mindestens ein Kanister enthält genau zwei Liter Wasser

Darüber hinaus gibt es die folgenden möglichen drei *Aktionen*:

1. Auffüllen eines Kanisters (AF)
2. Ausschütten eines Kanisters (AS)
3. Umfüllen des Inhalts eines Kanisters in einen anderen (UF)

In Abbildung 2.2 ist der dazugehörige Graph zu dem Problem ausschnittsweise dargestellt. Die Knotenbezeichnungen repräsentieren dabei den aktuellen Zustand von dem Kanister mit 4 Litern Fassungsvermögen (x) und dem mit 3 Litern (y). Die Kantenbeschriftungen zeigen die nötige Aktion (AF = Auffüllen, AS = Ausschütten, US = Umschütten) für die jeweiligen Zustandswechsel.

Abbildung 2.2: Das Kanister-Problem ausschnittsweise als Suchraum mit Hilfe eines gerichteten Graphen dargestellt. [12, Anlehnung an Bild 3.3]

Anhand des Graphen kann ausgehend vom Startzustand der Suchbaum gebildet werden, dargestellt in Abbildung 2.3. Auf diesen Suchbaum kann nun ein passender Suchalgorithmus angewandt werden.

Bei den Suchalgorithmen wird grundsätzlich zwischen den *uninformierten* und den *informierten* oder auch *heuristischen* Algorithmen unterschieden. [12, S. 114ff]

Bei der *informierten* oder auch *heuristischen* Suche fließen Zusatzinformationen in den Suchalgorithmus mit ein, um diesen zu optimieren, dies können zum Beispiel Erfahrungswerte sein. Ein Erfahrungswert kann beim Suchen eines möglichen Zuges beim Schach die Information sein, dass der König nie isoliert von anderen Figuren auf dem Brett positioniert

Abbildung 2.3: Anhand des Graphen aus Abbildung 2.2 ergibt sich der folgende Suchbaum (Ausschnitt), ausgehend von dem Startzustand, in dem beide Kanister leer sind. [12, Anlehnung an Bild 3.3]

werden sollte. Aber auch bei der Berechnung einer optimalen Route zwischen zwei Orten können die Angabe von Kosten, die die Distanz zwischen zwei Orten widerspiegeln, zusätzliche Informationen darstellen. Dies kann mit Hilfe von Heuristiken entsprechend in dem Suchalgorithmus berücksichtigt werden. Suchalgorithmen, die solche Zusatzinformationen verwenden sind z.B. die *Nächster-Nachbar-Suche*^g, die *Bestensuche*^g oder der *A*-Algorithmus*^g. [12, S. 114ff]

Die *uninformierte* Suchstrategie agiert ohne zusätzliche Informationen „blind“ durch ihren Suchbaum, Beispiele hierfür sind die sogenannte *Breitensuche*^g und die *Tiefensuche*^g. [12, S. 114ff]

In diesem hier vorgestellten Beispiel mit dem Kanister-Problem, liegen für die Suche nach der Lösung keine zusätzlichen Information, die die Suche beschleunigen könnten, vor. Deshalb würde hier ein uninformerter Suchalgorithmus verwendet werden, speziell in diesem Fall führt die Breitensuche am schnellsten zum Ziel. [12, S. 114ff]

2.3.5 Konnektionismus - Künstliche neuronale Netze

Der Mensch verdankt seine Intelligenz, die Fähigkeit neue motorische und intellektuelle Fertigkeiten zu erlernen und sich an neue unbekannte Umweltbedingungen anpassen zu können, einem hochkomplexen und adaptiven Netzwerk aus Nervenzellen in seinem Gehirn, dem sogenannten neuronalen Netz. [12, S. 189ff]

Der Leitfaden der sogenannten *konnektionistischen* KI setzt genau auf diesem Ansatz an und versucht ein biologisches neuronales Netz durch ein KNN auf einem Rechner zu simu-

lieren, um dabei die Arbeitsweise des menschlichen Gehirns nachzubilden. [12, S. 189ff]

KNN werden hierfür auf spezielle Verhalten trainiert, sodass sie anschließend dazu in der Lage sind auf bekannte Trainingssituationen aber auch neue Situationen entsprechend zu reagieren. [12, S. 189ff]

Ein biologisches Neuron besteht grundsätzlich aus Dendriten, einem Zellkern, einem Axon und Synapsen. Die Dendriten nehmen Erregungen von anderen Neuronen auf und leiten diese an den Zellkern weiter. Dieser wiederum leitet die Erregungen an das Axon und dieses an die anliegenden Synapsen weiter, die mit anderen Neuronen verbunden sind. Ein künstliches Neuron bildet diese Bestandteile und die Arbeitsweise in Softwarebausteinen nach. In Abbildung 2.4 ist oben ein biologisches und unten ein künstliches Neuron abgebildet. [12, S. 189ff]

Abbildung 2.4: Das biologische (oben) und das künstliche Neuron (unten) im Vergleich. [12, Bild 5.2 und 5.3]

Die nachfolgende Auflistung beschreibt zunächst die einzelnen Bestandteile und Funktionen eines künstlichen Neurons:

- **Eingabeinformationen**, bestehend aus direkten Eingabewerten o_k und jeweiligen Wichtungen w_{kj}
- **Propagierungsfunktion** net_j , verknüpft die Eingabeinformationen mit den Gewichten zu einer Netzeingabe. Die Netzeingabe fasst quasi die eingehenden Informationen zusammen.

- **Aktivierungsfunktion** f_{act} , bestimmt anhand der Netzeingabe und unter Berücksichtigung des Schwellwertes Θ_j die neue Aktivität a_j bzw. neuen Zustand des Neurons: $a_j = f_{act}(net_j, \Theta_j)$
- **Ausgabefunktion** f_{out} , ermittelt anhand der Aktivität den Ausgabewert o_j , welcher wiederum an andere Neuronen weitergegeben wird um da als Eingabeinformation zu dienen: $o_j = f_{out}(a_j)$
- **Lokaler Speicher**, enthält den aktuellen Zustand bzw. die aktuelle Aktivierung a_j und den Schwellwert Θ_j des Neurons [12, S. 189ff]

Ein Neuron in einem KNN ist also eine Verarbeitungseinheit, die gewichtete Eingangswerte mit Hilfe einer Propagierungsfunktion zusammenfasst. Darauf aufbauend wird, mit Hilfe einer Aktivierungsfunktion und unter Beachtung eines Schwellwertes, ein Aktivierungszustand bestimmt. Anhand dieser Aktivierung ermittelt die Ausgabefunktion die Ausgabe des Neurons.

Die Komplexität der jeweiligen Funktionen von Neuronen sollen dabei einfach gehalten sein, denn die Leistungsfähigkeit eines KNN wird durch das Zusammenschalten vieler einfacher Elemente mit einfachen Verarbeitungsschritten erreicht. [12, S. 189ff]

Ein kleines Beispiel ([12, Beispiel 5.1]) soll die Arbeitsweise eines Neurons in einem KNN verdeutlichen. Hierfür wird eine logische UND-Schaltung mit Hilfe eines KNN nachgebildet:

Das kleinste KNN besteht aus genau einem Neuron und wird *Linear Threshold Unit* (LTU) genannt. Mit diesem einzelnen Neuron ist es möglich, eine logische UND-Schaltung zu emulieren. Hierfür müssen geeignete Werte für die Gewichte und den Schwellwert des Neurons verwendet werden. In Abbildung 2.5 ist das Neuron für die UND-Schaltung dargestellt.

Abbildung 2.5: Der Aufbau des KNN zur Abbildung einer UND-Schaltung [12, Bild 5.6]

Normalerweise erfolgt das Finden von geeigneten Gewichten und Schwellwerten in Form eines Trainingsprozesses. In diesem einfachen Beispiel werden jedoch die Werte vorab geeignet festgelegt. Die Gewichte haben die Werte $w_1=1$ und $w_2=1$ und der Schwellwert wird auf $\Theta=1,5$ gesetzt. Durch die Festlegung des Schwellwertes auf 1,5 kann das gewünschte Verhalten des Netzes erzielt werden, da nur das Aufsummieren von zwei Eingaben, mit einem jeweiligen Wert von 1, zur Überschreitung des Schwellwertes führen.

Das Verhalten der Schaltung des Neurons kann in Tabelle 2.3 abgelesen werden, in den Spalten o_1 und o_2 sind die Eingangswerte dargestellt. Die Spalte net stellt das Ergebnis der Propagierungsfunktion dar, die mit Hilfe der jeweiligen Wichtung, die Eingangswerte aufsummiert. Die letzte Spalte repräsentiert die Ausgabe o und die Aktivität act , die das Neuron aufgrund der erhaltenen Eingaben annimmt. [12, S. 189ff]

o_1	o_2	net	$o=act$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

Tabelle 2.3: Die Ein- und Ausgabemuster des UND-Schalters. [12, Tab. 5.1]

Ein KNN besteht entweder aus einem Neuron (LTU) oder aus einer Menge an Neuronen, die durch gerichtete Verbindungen und Wichtungen miteinander verknüpft sind. Durch die Anzahl der Verbindungen werden die verschiedenen Architekturen festgelegt. [12, S. 189ff]

Ein KNN und deren Struktur kann gut mit Hilfe eines Graphen dargestellt werden. In Abbildung 2.6 ist die Architektur eines *vorwärts gerichteten* Netzes dargestellt. Ein KNN wird in Schichten unterteilt, wobei es drei Arten von Schichten gibt:

- **Eingabeschicht**, grün umrandet, bestehend aus einer Menge an Eingabe-Neuronen
- **Ausgabeschicht**, rot umrandet, bestehend aus Ausgabe-Neuronen und daran erkennbar, dass von diesen Neuronen keine weiterführenden Verbindungen ausgehen
- **verdeckte Schicht(en)**, die sich zwischen der Eingabe- und Ausgabeschicht befinden und sogenannte *verdeckte* Neuronen enthalten

Abbildung 2.6: Das vorwärts gerichtete KNN ist in drei Schichten aufgeteilt, eine Eingabeschicht (grün), eine verdeckte Schicht (blau) und eine Ausgabeschicht (rot). [12, S. 197]

In Abbildung 2.7 sind zusätzliche mögliche Architekturen von KNN dargestellt, dazu gehören ein vorwärts gerichtetes Netz mit mehreren verdeckten Schichten (links) oder auch der Möglichkeit, eine Schicht zu überspringen (mittig) und ein KNN mit einer sogenannten Rückkopplung (rechts). [12, S. 189ff]

Die Wahl einer geeigneten Architektur hängt von dem zu lösenden Problem und dem gewünschten Einsatzbereich ab. Darauf soll jedoch hier nicht weiter eingegangen werden, dies kann aber in [12, S. 202ff] genauer nachgelesen werden.

Neben einer geeigneten Architektur ist eine ausreichende Anzahl an Trainingsdaten und

Abbildung 2.7: Es gibt verschiedene Architekturen für KNN. Die Wahl der Architektur hängt von der Problemstellung ab. [12, S. 197f]

Testdaten erforderlich, mit denen zunächst das Netz trainiert und anschließend mit bis dato unbekannten Daten getestet wird. Bei nicht zufriedenstellendem Ergebnis anhand der Testdaten, muss das Netz neu ausgerichtet werden. Dies kann mit Hilfe von veränderten Lernparametern, Lernalgorithmen oder gar der Architektur erfolgen. Der beschriebene Kreislauf der Entwicklung eines KNN ist auch nochmals in Abbildung 2.8 verdeutlicht. [12, S. 189ff]

Abbildung 2.8: Die Entwicklung eines KNN erfolgt in einem iterativen Prozess mit Hilfe von Trainings- und Testdaten. [12, Bild 5.7]

Für die Art des Antrainierens eines KNN unterscheidet man zwischen drei Verfahren, dem *überwachten*, dem *bestärkenden* und dem *nicht überwachten* Lernen.

Beim überwachten Lernen liegen die Eingabewerte und die dafür erwarteten Ausgabewerte vor und damit wird das Netz trainiert. Dieses Lernverfahren gilt in der Anwendung als sehr effizient.

Bestärkendes Lernen wird eingesetzt, wenn für die Eingabedaten vorher nicht bekannt ist, ob diese richtig oder falsch sind. Es kann nur festgestellt werden, ob die Ausgabe richtig

ist. Anhand daran wird selbstständig ermittelt welche Aktionen zum Erfolg oder Misserfolg geführt haben und eine Strategie entwickelt. Diese Methode des Lernens kommt vor allem in der Robotik zum Einsatz, über verschiedene Sensoren besteht hier die Möglichkeit die Umwelt wahr zu nehmen. Durch Motoren sind die Roboter in der Lage sich in dieser zu bewegen bzw. mit ihr zu interagieren.

Das unüberwachte Lernen findet Anwendung in den Bereichen, in denen das Ergebnis nicht vorher klar ist, wie beispielsweise Clusterungen in Form von Klassifizierungen bei denen die Klassengröße im Voraus nicht bekannt ist. Ein Anwendungsfall hierfür wäre beispielsweise das Einordnen von Fotos in bestimmte Klassen wie Landschaftsfotos, Portraits, Tieraufnahmen etc. [12, S. 189ff]

Abschließend lässt sich zusammenfassen, dass alle KNN-Modelle über die Eigenschaft verfügen, mit Hilfe einer Vielzahl an einfachen Einheiten, Neuronen oder Teilchen, zusammen eine Aufgabe zu lösen. Das Verknüpfen dieser Einheiten kann in Form von gewichteten Verbindungen oder Verarbeitungsfunktionen erfolgen.

Ziel dieses Zusammenwirkens der einzelnen Einheiten ist das Lösen von Problemstellungen, die nicht in Form eines Algorithmus explizit umschrieben werden können, deren Lösung jedoch durch Beispiele beschreibbar ist.

Anhand dieser Beispiele wird ein Netzverhalten angelernet, das anschließend dazu in der Lage ist, auch bis dahin unbekannte Eingaben angemessen zu verarbeiten. [12, S. 189ff]

2.4 Eine Auswahl an Anwendungsbereichen der KI

Die Einsatzgebiete und Anwendungsbereiche der KI sind vielfältig. Angefangen mit der maschinellen Beweisführung und dem Einsatz in Spielanwendungen, wie beispielsweise *DeepBlue* oder *AlphaGo*, reicht die Einflussnahme von KI-Techniken von der Verarbeitung natürlicher Sprache, Beispiel *Siri* oder das Android-Pendant *Google Now*, bis hin zur Bildverarbeitung und dem Einsatz in Form von Expertensystemen. Diese sind gerade in der medizinischen Diagnostik weit verbreitet.

Auch in autonomen Autos wie dem *Google self-driving car* (Abbildung 2.9) werden Techniken der KI angewendet. [18, S. 6f]

Abbildung 2.9:

Das *Google self-driving car* benutzt Techniken der *Künstlichen Intelligenz*, um autonom am Straßenverkehr teilnehmen zu können. [70]

Einen großen und sehr anspruchsvollen Teilbereich der KI stellt ebenfalls die Robotik dar.

Hier sind Fähigkeiten wie die Bildverarbeitung, natürliche Sprachverarbeitung, Selbstständigkeit und das Erlernen von neuen Fähigkeiten gefordert. In diesen Bereichen werden sich in den nächsten Jahren neue Meilensteine der KI versprochen.

Ein Kräftemessen der Robotik findet jährlich in Form des sogenannten RoboCups¹ statt, bei dem Roboter in verschiedenen Ligen antreten. Dazu gehört beispielsweise die Disziplin *RoboCup@Home*, in der autonome mobile Roboter Aufgaben des täglichen Lebens in einer Modell-Wohnung bewältigen müssen oder auch die Liga *RoboCupRescue*. Dieses Teilgebiet befasst sich mit dem Einsatz von Robotern zur Suche von Überlebenden während einer Katastrophe wie bspw. einem Erdbeben oder Ähnliches.

Besonders populär ist die Liga *RoboCupSoccer*, in der u.a. die NAO-Roboter der Firma Aldebaran gegeneinander im Fußball antreten, ein Ausschnitt dazu ist in Abbildung 2.10 dargestellt. [18, S. 6f]

Abbildung 2.10: Jährlich findet in Form des RoboCups ein weltweites Kräftemessen im Fußball zwischen den NAO-Robotern der Firma Aldebaran statt. [55]

Ein weiterer Anwendungsbereich von KI sind kognitive Systeme. Zu den bekanntesten kognitiven Systemen zählt *Watson* von IBM. *Watson* hat das Ziel selbstständig Informationen aus Daten zu generieren, um daraus Schlüsse ziehen zu können.

Hierfür ist *Watson* in der Lage, die natürliche menschliche Sprache zu verstehen, deren Wörter und Kontext zu analysieren und die Informationen zu verarbeiten, um so präzise Antworten auf Fragen in natürlicher Sprache ausgeben zu können.

Damit ist das System nicht nur dafür prädestiniert, in berühmten Fernseh-Quizshows Quizshow-Könige zu besiegen, sondern *Watson* soll zukünftig auch im Gesundheitswesen und der Finanzbranche zum Einsatz kommen. [34]

Hierfür soll er als Entscheidungshelfer bei Diagnosen die Ärzte unterstützen, indem er alle für die Analyse verfügbaren Datenquellen, wie z.B. elektronische Krankenakten, Notizen von Ärzten und Pflegepersonal oder Forschungsergebnisse, einbezieht und anhand dieser Daten

1 RoboCup: <http://www.robocup.org/>

eine gewichtete Diagnoseauswahl vorschlägt. [34]

In der Finanzbranche soll *Watson* als Unterstützer für Berater in Banken fungieren, um dadurch bessere Empfehlungen zu Finanzprodukten für Kunden abgeben zu können.

Dabei könnte *Watson*, basierend auf einer umfassenden Analyse von Marktbedingungen, bisherigen Entscheidungen und jüngsten Ereignissen im Leben des Kunden sowie verfügbaren Angeboten, gezielte und auf den Kunden abgestimmte Empfehlungen für die Finanzgeschäfte abgeben und damit die Beratung für Finanzprodukte qualitativ auf eine neuen Ebene heben. [34]

Kapitel 3

Smart Home und Servicerobotik

Dieses Kapitel dient als Vorbereitung auf das große Thema dieser Arbeit, indem es die beiden Techniken des *Smart Home* und der *Servicerobotik* kurz vorstellt. Zusätzlich soll durch die Einführung in beide Bereiche noch einmal deutlich werden, warum eine Verbindung der beiden Gebiete so interessant ist.

3.1 Smart Home

Die Vorstellung des Anwendungsfelds der Technik *Smart Home* erfolgt in Form einer kurzen Beschreibung und Definition und einer anschließenden Beschreibung typischer Anwendungsfelder.

3.1.1 Was ist Smart Home - Eine Definition

Die Technik *Smart Home*, welche auch unter den Begriffen *Intelligentes Wohnen* oder *Smart Living* verbreitet ist, kann in die Bereiche der Gebäude- und Hausautomatisierung eingeordnet werden.

Unter einem *Smart Home* wird ein intelligent vernetztes Gebäude verstanden, in dem verschiedene Sensoren und Aktoren miteinander so verknüpft sind, sodass viele routinemäßige Arbeiten automatisiert, zuverlässig und exakt ablaufen können. [62]

Einsatzfelder von *Smart Home* sind beispielsweise die automatisierte Betätigung von Rollläden zum Schutz vor Überhitzung oder Auskühlung von Räumen oder auch Zugangssysteme, die dafür sorgen, dass bei Ankunft nicht nur die Haustür geöffnet wird sondern auch zusätzliche Aktionen angestoßen werden, wie beispielsweise das Einschalten des Lichts oder das Abspielen von Musik etc.

Aber auch das Erkennen von Gefahren- und Notsituationen von Bewohnern sind mögliche Einsatzfelder der Technik und sie leistet somit auch einen Beitrag in Richtung Sicherheit. [62]

Oft fallen im Zusammenhang mit *Smart Home* auch die Begriffe *Smart Meter* und *Smart Grid*. Unter einem *Smart Meter* werden intelligente und vernetzte Zähler im Bezug auf Ressourcen und Energien verstanden wie Wasser, Gas oder Strom, die im Haushalt verwendet werden. Der intelligente Stromzähler ist Teil des *Smart Grids*. [4] Ein *Smart Grid*, also ein intelligentes Stromnetz, soll wiederum ermöglichen, Netzzustände in Echtzeit zu erfassen, um dadurch das Netz steuern und regeln zu können und damit die Netzkapazitäten voll auszuschöpfen. [10, S. 11f]

Zusammengefasst ist das Ziel des Einsatzes von *Smart Home*-Techniken die Erhöhung der Wohn- und Lebensqualität, der Sicherheit und eine effizientere Energienutzung. Demnach ist ein Zuhause *smart*, wenn:

„[...] verschiedene Sensoren und Aktoren intelligent miteinander verknüpft werden, um den darin lebenden Menschen das Leben zu erleichtern und sicherer zu machen und gleichzeitig den Energieverbrauch des Heims zu reduzieren.“ [62]

3.1.2 Eine Auswahl an Anwendungsbereichen im eigenen Zuhause

Der intelligente Kühlschrank verkörpert für viele Menschen die Vision eines intelligenten Zuhauses. Der smarte Kühlschrank, der seinem Besitzer den Einkauf abnimmt, lässt zwar noch auf sich warten, doch die Technik um *Smart Home* hat sich in den letzten Jahren immer weiter entwickelt und bietet heutzutage bereits eine große Vielfalt an Möglichkeiten. In Abbildung 3.1 werden die einzelnen Bausteine der *Smart Home*-Technik dargestellt.

Abbildung 3.1: Die Bausteine der Technik Smart Home.

Durch das Interagieren zwischen den einzelnen Komponenten des *Smart Home* können verschiedenste Anwendungsszenarien im wohnlichen Umfeld abgedeckt werden. Grob lassen sich diese in die drei großen Anwendungsbereiche *Komfort*, *Energie* und *Sicherheit* aufteilen. [62]

Ein Beispiel hierfür ist die gezielte Unterstützung älterer Menschen in ihrem Alltag, dazu zählen neben dem Komfort- und Energieaspekt in Form von bedarfsgerechter und optimaler Wärme- und Lichtversorgung beispielsweise auch das Erkennen von Notsituationen mit Hilfe von Sicherheitsanwendungen.

Aber nicht nur für das Erkennen von Notfallsituationen können Sicherheitsanwendungen ein-

gesetzt werden, sondern auch in Form des Einbruchsschutzes, der z.B. die Anwesenheit der Bewohner simuliert oder die Überwachung mit Hilfe von Kameras und Bewegungsmeldern, um dadurch ungewöhnliche Aktivitäten erkennen und melden zu können. [62]

Im Hinblick auf das *Smart Grid* bildet das *Smart Home* die Voraussetzung in einem möglichen Zukunftsszenario, in dem sich intelligente Haushaltsgeräte, wie zum Beispiel die Waschmaschine, automatisch an- und ausschalten in Abhängigkeit von dem aktuellen Stromtarif. [73]

3.2 Serviceroboter

Die nachfolgenden Ausführungen geben einen kurzen Überblick zu der Thematik *Serviceroboter*. Hierfür wird wieder zunächst eine Beschreibung und Definition der Technik gegeben und abschließend eine Auswahl an typischen Anwendungsfeldern in diesem Bereich erläutert.

3.2.1 Was ist ein Serviceroboter - Eine Definition

Der Begriff *Serviceroboter* setzt sich aus den beiden Wörtern *Service* und *Roboter* zusammen.

Das Wort *Service* hat seinen Ursprung aus dem Englischen und bedeutet ins Deutsche übersetzt *Dienst* oder *Bedienung*. [7] Der Begriff *Roboter* definiert sich durch eine

„(der menschlichen Gestalt nachgebildete) Apparatur, die bestimmte Funktionen eines Menschen ausführen kann; Maschinenmensch“. [6]

Die International Federation of Robotic (IFR) definiert *Serviceroboter* wie folgt:

„A service robot is a robot that performs useful tasks for humans or equipment excluding industrial automation application.“ [35]

Ganz allgemein gehalten kann unter einem *Serviceroboter* also ein *Roboter* verstanden werden, der dafür entwickelt wurde, für den Menschen Dienstleistungen zu erbringen. Um ihren Anforderungen gerecht zu werden und den Menschen zu unterstützen, müssen *Serviceroboter* eine Reihe an Fähigkeiten aufweisen.

Dazu gehören mobiles Agieren, die Umgebung zu verändern, mit Menschen zu interagieren oder auch eigenständig Aufgaben ausführen zu können etc. [54, S. 1]

Abschließend soll die Definition des Fraunhofer Institutes als Beschreibung eines *Service-roboters* dienen. Ein *Serviceroboter* ist demnach:

„[...] ein Roboter, der teil- oder vollautonom Dienstleistungen zum Nutzen menschlichen Wohlbefindens und für Einrichtungen ausführt. Serviceroboter werden unterschieden nach Serviceroboter für gewerbliche Anwendungen (üblicherweise bedient durch eine eingewiesene Person) und Serviceroboter für persönliche und domestische Anwendungen (bedient durch Laien, nicht eingewiesene Personen). [33]

3.2.2 Eine Auswahl an Anwendungsgebieten der Servicerobotik

Zu der Kategorie der *Servicerobotik* zählen eine Vielzahl von Robotertypen, wie beispielsweise menschenähnliche humanoide Roboter oder auch einfache mobile Plattformen mit Rädern. Sie kommen in verschiedensten Umgebungen, wie auf Land, im Wasser oder in der Luft, zum Einsatz. Ihre Aufgaben sind vielfältig, dazu gehören z.B. das Retten von Lebewesen in Notsituationen, die soziale Interaktion mit Menschen oder auch die Unterstützung bei Haushaltstätigkeiten.

In diesem Abschnitt stehen speziell Roboter mit Anwendungsgebieten im privaten Haushalt im Fokus. Hierfür gibt es bereits staubsaugende, Rasen mähende und Fensterscheiben putzende Roboter, es gibt jedoch auch spezielle *Pflegeroboter* wie den *Care-O-bot* vom Fraunhofer Institut oder *Romeo* [63] (Abbildung 3.2) von der Firma *Aldebaran*.

Deren Intention ist es, Menschen in Form eines Begleiters in den eigenen vier Wänden zu unterstützen, gerade auch wenn die geistigen und körperlichen Fähigkeiten im Alter anfangen nachzulassen. Hierfür gibt es eine ganze Reihe an Anwendungsszenarien, in denen der Roboter sich als nützlicher Alltagshelfer erweisen kann.

Abbildung 3.2: Der Roboter *Romeo* von der Firma *Aldebaran* unterstützt ältere Menschen in ihren eigenen vier Wänden. [63]

Darunter fallen Transportdienste wie zum Beispiel das selbstständige Finden, Holen und Bereitstellen von Medikamenten und Nahrungsmitteln. Das Erkennen von Notfallsituationen mit entsprechender Alarmierung, das Heben und Tragen von Menschen oder die Unterstützung bei der täglichen Körperpflege oder Essenaufnahme oder auch die Durchführung von Reinigungsaufgaben zählen ebenfalls zu unterstützenden Tätigkeiten.

Aber auch hinsichtlich der sozialen Komponente gibt es Einsatzfelder, wie beispielsweise das Einsetzen des *Serviceroboters* als Medium zur Kommunikation zwischen Besitzer und anderen Menschen oder auch als Kommunikationspartner selbst. [29]

Abschließend lässt sich festhalten, dass die Anwendungsszenarien der *Service-* bzw. speziell der *Pflegerobotik* die Intention verfolgen den Komfort und die Sicherheit für Personen in den eigenen vier Wänden zu erhöhen.

Folglich weisen die Ziele der *Servicerobotik* und die der *Smart Home*-Technik eine große gemeinsame Schnittmenge an Zielen auf. Eine Verbindung beider Forschungs- bzw. Anwendungsgebiete ist daher naheliegend.

Kapitel 4

Analyse der Anforderungen

In diesem Kapitel werden die Anforderungen an das Gesamtsystem und die einzelnen Komponenten analysiert und definiert.

Hierfür wird zunächst in Abschnitt 4.1 das große Ziel der Arbeit noch einmal genannt, indem ein Ausgangsszenario festgelegt und anhand daran die einzelnen zu betrachtenden Komponenten des Gesamtprojektes bestimmt werden. Anschließend wird in Form von Anwendungsfällen (Use-Cases) eine Evaluation der funktionalen Anforderungen an die zu entwickelnde intelligente Verarbeitungseinheit (IVE) durchgeführt. Im letzten Abschnitt erfolgt die Definition der Anforderungen an das Gesamtsystem, wozu die IVE, die Lernbausteine und die Entwicklungsbausteine zählen.

4.1 Das große Ganze - Was soll erreicht werden?

In diesem Abschnitt sollen die in Abschnitt 1.2 genannten Ziele noch einmal präzisiert werden, um daraus die benötigten Anforderungen für die einzelnen Teilelemente des Projekts ableiten zu können. Hierfür wird zunächst ein Ausgangsszenario beschrieben, in dem die zu entwickelnde Anwendung angewandt werden kann. Nachfolgend werden die Rahmenbedingungen für die einzelnen zu betrachtenden Elemente wie beispielsweise die Zielgruppe etc. festgelegt.

4.1.1 Das Szenario

Die nachfolgende Beschreibung soll als exemplarisches Ausgangsszenario für die zu entwickelnde Anwendung dienen.

Im Kontext der Anwendungsfelder *Smart Home* und *Servicerobotik* und unter Berücksichtigung der in Abschnitt 1.1 ausgeführten Motivation zur Vernetzung beider Techniken, wurde ein entsprechendes Szenario entworfen.

Bei der Wahl der agierenden Person wurde versucht eine statistisch durchschnittliche alleinlebende Person der Generation 65+ in Deutschland zu repräsentieren.

Laut dem statistischen Bundesamt lebten am Ende des Jahres 2013 rund 81 Millionen Menschen in Deutschland. Hiervon waren ca. 17 Millionen Menschen 65 Jahre oder älter. Mit 57% waren Ende 2013 mehr Frauen in der Generation 65+ vertreten als Männer (43%).

Im Jahr 2014 lebten ca. ein Drittel aller 65-jährigen und älteren Menschen allein in einem Einpersonenhaushalt. Dabei ist die Verteilung der alleinlebenden Menschen in Deutschland

ungleich. Mit einem Anteil von 45% leben Frauen in diesem Alter mehr als doppelt so häufig allein, als Männer in diesem Lebensabschnitt (19%). [66]

In einer von Statista veröffentlichten Umfrage¹ sprachen sich insgesamt zwei Drittel (67%) der befragten über 50-Jährigen dafür aus, auch mit 70 Jahren noch komplett ohne Hilfe zuhause wohnen zu wollen. Etwas mehr als die Hälfte der Befragten (57%) konnten sich ebenfalls vorstellen, Hilfe beim Wohnen im eigenen Zuhause in Anspruch zu nehmen. [65]

Anlehnend an diese Daten wird die zu überwachende Person im Folgenden exemplarisch Karin² genannt und soll den überwiegenden Teil der über 64-jährigen alleinlebenden Menschen in Deutschland repräsentieren.

Folgende Annahmen werden über Karin getroffen:

1. Karin ist weiblich
2. Karin ist 73 Jahre alt
3. Karin lebt allein zuhause
4. Karins Kinder leben im Ausland
5. Karins Mann ist bereits verstorben
6. Karin legt sehr viel Wert darauf, weiterhin selbstbestimmt in ihrer eigenen Wohnung zu leben

Karin konnte bisher ihren Lebensalltag gut und selbstständig bestreiten.

In letzter Zeit setzen bei Karin jedoch erste mögliche Anzeichen für eine Demenz ein:

Vorletzte Woche hat sie ihre wöchentliche Bibelstunde, zu der sie schon seit Jahren regelmäßig geht, einfach vergessen. Zur Zeit fällt ihr in der dunklen Jahreszeit auch das morgendliche Aufstehen immer schwerer und sie möchte an manchen Tagen am liebsten ganz liegen bleiben. Aber nicht nur ihr geregelter Tagesablauf leidet, ihre Vergesslichkeit birgt auch Gefahren. So hat sich Karina vor ein paar Tagen auf den Weg zum Supermarkt gemacht und dabei den Herd in ihrer Wohnung angelassen. Auf halber Strecke fiel ihr jedoch ihr Missgeschick auf und so konnte Schlimmeres verhindert werden.

Karins Kinder sind besorgt und halten dies für mögliche Anzeichen einer beginnenden Demenz. Sie wissen, dass für ihre Mutter nun ein geregelter Tagesablauf und die gewohnte Umgebung, das heißt das häusliche Umfeld, wichtig sind, denn dies gibt an Demenz erkrankten Menschen Sicherheit und kann gar einen möglichen Krankheitsverlauf verlangsamen. [22]

4.1.2 Intelligente Verarbeitungseinheit

Ziel ist der Entwurf und die Umsetzung einer geeigneten IVE zur Verhaltensermittlung von Personen in ihrem häuslichen Umfeld. Dabei handelt es sich mehrheitlich um ältere Menschen, bei denen mit Hilfe von geeigneten Datenquellen in Form von Sensorik im Wohnbereich, verwertbare Aussagen hinsichtlich deren Alltagskompetenz getroffen werden kann.

¹ Deutschland; 08.12.2010 bis 03.11.2011;

² 1100 Befragte ab 50 Jahre; 500 Mieter, 600 Wohnungs-/Hauseigentümer

² Beliebtester weiblicher Vorname im Jahr 1943 in Deutschland.

Quelle: <http://www.beliebte-vornamen.de/jahrgang/j1943>

Hierfür sollen signifikante Abweichungen vom Tagesablauf, der über einen längeren Zeitraum beobachtet wurde, mittels IVE detektiert und als Information bereitgestellt werden.

Die IVE soll in dem angestrebten Szenario als Bindeglied zwischen den beiden Techniken *Smart Home* und *Servicerobotik* fungieren.

Hierfür werden die Sensoren aus dem *Smart Home*-Bereich dafür verwendet, um Anhaltspunkte über den Tagesablauf der zu überwachenden Person, also Karin, mittels Sensordaten zu sammeln. Diese Daten werden an die IVE übertragen. Diese wiederum hat die Aufgabe, die Daten entsprechend zu verwalten, zu verarbeiten und auszuwerten, sodass der Tagesablauf von Karin auf etwaige Verhaltensauffälligkeiten überprüft werden kann, die nicht dem normalen bekannten Verhalten entsprechen.

Neben dem Erkennen von Unregelmäßigkeiten bietet die IVE zusätzlich eine Schnittstelle an, die darüber informiert, ob ein auffälliges Verhaltensmuster eingetreten ist.

Übertragen auf das Ursprungsszenario, könnte diese Schnittstelle einem *Serviceroboter* dienen, dem dadurch alle nötigen Informationen zur Verfügung gestellt werden, um entsprechend auf das veränderte oder auffällige Verhalten von Karin reagieren zu können.

Für die Umsetzung der IVE sollen geeignete Segmente der KI verwendet werden.

In Abbildung 4.1 ist der grundsätzliche Aufbau der intelligenten Infrastruktur schematisch skizziert. Dabei steht die Sensorik stellvertretend für den *Smart Home*-Bereich und die Aktorik symbolisiert den Bereich, in dem in diesem Szenario die *Servicerobotik* agiert.

Abbildung 4.1: Schematische Darstellung der Infrastruktur und der Komponenten für das Ausgangsszenario.

4.1.3 Lernbausteine

Neben der Umsetzung und Bereitstellung einer geeigneten intelligenten Infrastruktur zielt die Arbeit zusätzlich auf die Evaluation und Entwicklung eines Lernbaukastens für Schüler in Form von Lernbausteinen ab.

Diese Lernbausteine sollen das Thema *Künstliche Intelligenz* thematisieren und es ermöglichen, die Arbeitsweise der verwendeten KI-Infrastruktur zu verstehen und darüber hinaus praktisch anzuwenden.

Hierfür stellt die intelligente Infrastruktur die benötigten Werkzeuge als Basis zur Verfügung, mit denen es für die Anwender möglich wird, die Arbeitsweise von KI zu verstehen und anzuwenden.

4.1.4 Entwicklungsbausteine

Die intelligente Infrastruktur soll die Grundlage für weitere Entwicklungen und Implementierungen darstellen. Hierfür sollen Ideen in Form von Entwicklungsbausteinen konzipiert werden, die als Leitfaden und Anregung für spätere Entwicklungen z.B. im Rahmen von Seminararbeiten und Abschlussarbeiten des Bachelorstudienganges Telematik dienen.

4.1.5 Wer gehört zur Zielgruppe?

Für die zu entwickelnden Lern- und Entwicklungsbausteine muss die Zielgruppe klar definiert sein. Nur mit Hilfe einer klar definierten Zielgruppe ist es auch möglich, die Bausteine zielgerichtet entwickeln zu können.

Die Zielgruppe für die Lernbausteine kann aus der Zielgruppe des Schülerlabors Robotic abgeleitet werden. Hierbei handelt es sich um Schüler der SEK 2 aus Berlin und Brandenburg.

Diese Stufe des deutschen Bildungssystems umfasst gewöhnlich zwei bis drei Jahre und hat bei erfolgreichem Abschluss den Erwerb der allgemeinen Hochschulreife zur Folge. Diese berechtigt zum Studium an einer Hochschule oder Universität in Deutschland.

Hierfür wird eine gymnasiale Oberstufe oder eine berufsbildende Schule besucht und der Abschluss in Form einer Abiturprüfung abgelegt. Je nach Schulausführung handelt es sich dabei um Schüler der Klassenstufen 11, 12 und 13. Ausgehend von dem normalen Einschulungsalter von 6 Jahren in Berlin und Brandenburg und einem normalen Schulverlauf ohne das Wiederholen einer Klasse, kann eine Zielgruppe im Alter von ca. 17 bis 19 Jahren angenommen werden. Zusätzlich wird davon ausgegangen, dass die Zielgruppe zumindest an einem Informatik-Grundkurs in Form eines Schulfaches teilgenommen hat.

An die zweite Zielgruppe richtet sich vor allem die Konzeption der Entwicklungsbausteine. Dabei handelt es sich in erster Linie um Studierende des Bachelorstudiengangs Telematik. Dieser Studiengang setzt die Fachhochschulreife voraus und umfasst eine Regelstudienzeit von 6 Fachsemestern, wobei im letzten Semester eine Bachelorarbeit anzufertigen ist.

Das Alter der Zielgruppe beginnt bei ca. 18 Jahren¹ und ist nach oben hin offen, da hier keine Zulassungsbeschränkungen gegeben sind.

4.2 Anwendungsfälle für die intelligente Verarbeitungseinheit

Für das Ableiten der Anforderungen an die intelligente Verarbeitungskomponente werden in diesem Abschnitt zunächst Anwendungsfälle in Form von Use-Case-Diagrammen erstellt. Hierbei werden die Ziele an das System aus Sicht der einzelnen Akteure beschrieben. Die Akteure dieses Szenarios setzen sich aus *Sensoreinheit*, *Anwender* und *Serviceroboter* zusammen.

Mit Hilfe dieser Anwendungsfälle soll es ermöglicht werden, eine grobe Übersicht und Ordnung der funktionalen Anforderungen an das System zu erhalten, um anschließend in Abschnitt 4.3 die funktionalen Anforderungen eindeutig definieren zu können.

4.2.1 Sensoreinheit

Die Sensoreinheit hat in dem Ausgangsszenario die Aufgabe, die IVE mit aktuellen Sensorwerten zu versorgen.

Hierfür muss die Sensoreinheit über die Möglichkeit verfügen, sich an das System anmelden zu können. Darüber hinaus besteht die Notwendigkeit, dass die jeweiligen Sensorwerte an die IVE gesendet werden können. Diese zwei Funktionalitäten sind nochmals in Abbildung 4.2 in Form eines Use-Case-Diagrammes verdeutlicht.

Abbildung 4.2: Ziel der Sensoreinheit ist es, sich an die Verarbeitungskomponente anmelden zu können und aktuelle Sensordaten an diese zu senden.

¹ Bei der Annahme von einem Einschulungsalter von 6 Jahren und dem Erlangen des Abiturs nach 12 Jahren Schullaufbahn.

4.2.2 Anwender

Der Anwender hat mittels einer Benutzeroberfläche die Möglichkeit, mit der Verarbeitungskomponente zu interagieren. Für den Benutzer teilen sich die insgesamt drei großen Anwendungsbereiche in *Discover*, *Training*, *Go Live* auf.

Discover

Der Anwendungsbereich *Discover* bietet dem Benutzer vor allem die Möglichkeit, die Sensorik aus dem *Smart Home*-Bereich in der Anwendung einzusehen, zu überwachen und zu bearbeiten. So können bspw. alle angemeldeten Sensoren sowie deren Sensorwerte eingesehen werden. Die Sensoren haben in der IVE die Aufgabe, Aufschluss über das Verhalten von Karin zu erhalten und zusätzlich werden deren Werte für die Durchführung eines Trainings verwendet.

Neben realer Sensorik, die in diesem Szenario in der Wohnung von Karin angebracht ist, gibt es für den Anwender auch die Möglichkeit, Trainingsdaten in Form einer *Simulation* zu erzeugen. Hierfür können Sensoren über die Benutzeroberfläche im System hinzugefügt, d.h. angemeldet, und deren Werte simuliert werden. Zusätzlich ist es möglich, die Sensoren wieder zu entfernen.

Neben der Simulation von Sensoren hat der Nutzer außerdem die Möglichkeit, im Trainings- oder Überwachungsmodus die aktuelle Uhrzeit und den Wochentag hinzuzufügen oder auch einzustellen. Dies ermöglicht eine komfortable Durchführung von Trainingseinheiten und die anschließende Überwachung der alltäglichen Verhaltensmuster von Karin in Abhängigkeit von der Zeit. In Abbildung 4.3 sind die Funktionen des *Discover*-Bereichs zusammengefasst.

Abbildung 4.3: Der Anwender hat im Anwendungsbereich *Discover* bspw. die Möglichkeit, angemeldete Sensoren einzusehen.

Training

Das *Training* dient zum Abbilden von Alltagsmustern oder in diesem Szenario speziell zum Abbilden des Alltags von Karin in ihren eigenen vier Wänden. Mit Hilfe dieser Muster kann später im Überwachungsmodus das tatsächliche Verhalten von Karin in ihrem Alltag überprüft bzw. abgeglichen werden. In Form eines *Trainings*-Modus kann der Nutzer der intelligenten Verarbeitungskomponente das Normalverhalten von Karin anhand der Sensorik bekannt machen. Hierfür hat er die Option, den *Trainings*-Modus zu starten, ein Training durchzuführen und bei Bedarf nach einer geeigneten Trainingsdauer diesen wieder zu beenden (Abbildung 4.4). Zusätzlich besteht die Funktionsmöglichkeit, einen antrainierten Zustand abzuspeichern und zu einem späteren Zeitpunkt auch wieder zu laden. So kann der Anwender beispielsweise unterschiedliche Trainingseinheiten miteinander vergleichen, testen etc.

Abbildung 4.4: Dem Anwender steht über eine Benutzeroberfläche im Bereich *Training* die Möglichkeit zur Verfügung, ein Verhaltensmuster anzutrainieren.

Go Live

Der Anwendungsbereich *Go Live* dient in der IVE quasi als Überwachungsmodus.

Der *Go Live*-Bereich hat die Aufgabe, den Ist-Zustand der aktuell am System angemeldeten Sensoren und deren Werte mit dem antrainierten Soll-Zustand abzugleichen und grobe Abweichungen zu erkennen.

Um in diesen Überwachungsmodus zu gelangen, hat der Benutzer die Möglichkeit, diesen zu starten und bei Bedarf auch wieder zu beenden. Während des Überwachungsmodus können durch den Anwender auch Sensorwerte simuliert werden, um beispielsweise ein Verhalten von Karin vorzutäuschen oder die Qualität der trainierten Verarbeitungskomponente zu überprüfen.

Befindet sich die IVE im Überwachungsmodus, wird dem Benutzer der aktuelle Status und eventuelle Verhaltensabweichungen angezeigt. In Abbildung 4.5 sind die einzelnen Funktionen für den Anwender nochmals dargestellt.

Der letzte Bereich bildet die Aktorik ab. Hier hat der Anwender die Möglichkeit angemeldete Aktoren einzusehen. Speziell in diesem Fall handelt es sich dabei um *Serviceroboter*.

Abbildung 4.5: Der Anwender kann im *Go Live*-Modus u.a. den Überwachungsstatus einsehen.

4.2.3 Serviceroboter

Die Hauptaufgabe des *Serviceroboters* besteht darin, mit Hilfe der IVE, Informationen über das Verhalten von Karin zu erhalten und entsprechend geeignet darauf zu reagieren.

Hierfür muss der *Serviceroboter* die Möglichkeit besitzen, sich an der Verarbeitungskomponente an- und abzumelden. Sobald der *Serviceroboter* am System angemeldet ist, hat er die Möglichkeit, Informationen über den aktuellen Stand zum Verhalten von Karin abzurufen.

Das entsprechende Reagieren und Agieren des *Serviceroboters*, bei Erkennung von Abweichungen im Tagesablauf von Karin, gehört nicht mehr zum Bestandteil dieser Arbeit. Die Funktionalitäten, die die IVE für den *Serviceroboter* zur Verfügung stellen muss, sind in Abbildung 4.6 nochmals aufgezeigt.

Abbildung 4.6: Dem *Serviceroboter* müssen über die intelligente Verarbeitungskomponente die Funktionalitäten *an-* und *abmelden* sowie das *Abrufen von Informationen* bereitgestellt werden.

4.3 Definition der Anforderungen

In diesem Abschnitt werden abschließend für dieses Kapitel die benötigten Anforderungen an die Elemente *Lernbausteine*, *intelligente Verarbeitungseinheit* und *Entwicklungsbausteine* zusammengestellt. Dies erfolgt in Form einer Beschreibung und Begründung ausgewählter Anforderungen, zusätzlich sind die vollständigen Anforderungen in Anhang A nochmals tabellarisch diesem Dokument angefügt und liegen als Excel-Datei „*Anforderungen.xlsx*“ in dem Verzeichnis „*Dokumente/Anforderungsanalyse*“ zusätzlich auf der mitgelieferten CD-ROM (siehe Anhang B) bei.

4.3.1 Anforderungen an die Lernbausteine

Grundsätzlich haben Lernbausteine in der Lehre die Aufgabe Wissensnetze zu bestimmten Fachinhalten aufzubauen. Hierfür sollen Schüler mit Hilfe dieser Bausteine in der Lage sein, durch selbstständiges Arbeiten sich entsprechendes Fachwissen zu erarbeiten und optimaler Weise in praktischer Form anzuwenden und zu festigen.

Durch die bisherige Arbeit in den Schülerlaboren der Robotik kann hier bereits auf eigene gesammelte Erfahrungen zurückgegriffen werden. Die Philosophie der Robotiklabore im Studiengang Telematik beruht auf dem Ansatz des eigenständigen Arbeitens, dem Entwickeln von eigenen Ideen und einem hohen praktischen Anteil. Die Schüler haben hier nach einer kurzen Einführung in die Thematik die Möglichkeit, ihre eigenen Ideen praktisch umzusetzen und die Ergebnisse am „lebenden Objekt“ mit Hilfe eines Roboters auszuprobieren und zu testen.

Hierbei lässt sich immer wieder beobachten, dass gerade der hohe Praxisanteil oft einen großen Ansporn bei den Schülern hervorruft und ein hohes Maß an Kreativität und Ehrgeiz bei der Arbeit mit den Robotern freisetzt.

Diese Erkenntnisse und Erfahrungen sollen als Leitphilosophie für die Entwicklung und Ge-

staltung der Lernbausteine dienen.

Zusätzlich bilden Fähigkeiten, wie das selbstständige Erarbeiten von Wissen und dem Anwenden des Erlernten in der Praxis sowie das Hineindenken in fremde Thematiken, eine wichtige Voraussetzung zur Vorbereitung auf ein Studium oder die spätere Arbeitswelt. Das einfache Vermitteln von Tatsachen reicht hierfür oft nicht aus, vielmehr muss der Wissenserwerb ein aktiver, selbstgesteuerter, konstruktiver und sozialer Prozess sein.

Die Lernbausteine sollen darüber hinaus die Möglichkeit bieten, in den Unterricht der Schulen integriert zu werden. Sie haben dabei die Aufgabe den Unterricht in den Schulen zu ergänzen aber keinesfalls zu ersetzen.

Hierfür müssen die Lernbausteine einerseits auf die Zielgruppe, das heißt Schüler der SEK 2, zugeschnitten sein und andererseits Schnittstellen zu den Lehrplänen des Faches Informatik in Berlin und Brandenburg aufweisen.

4.3.2 Anforderungen an die intelligente Verarbeitungseinheit

Neben den funktionalen Anforderungen, die sich anhand der Anwendungsfälle aus Abschnitt 4.2 ableiten lassen, sollen in diesem Abschnitt zusätzliche Anforderungen an die intelligente Verarbeitungskomponente kurz erläutert werden.

Mit Hilfe der zu entwickelnden IVE soll das Thema und die Arbeitsweise der *Künstlichen Intelligenz* in Form von Lern- und Entwicklungsbausteinen verständlich und praxisnah erklärt werden. In Folge dessen ist das Verwenden von Werkzeugen der KI eine grundlegende Anforderung an die Verarbeitungskomponente.

Für die Bedienung muss dem Anwender eine grafische Benutzeroberfläche zur Verfügung stehen. Mit dieser soll es ihm möglich sein, die einzelnen Funktionen für die einzelnen Modi (*Discover*, *Training*, *Go Live*) durchzuführen.

Für das An- und Abmelden und das Senden von Daten einer Sensoreinheit, muss die IVE eine geeignete Schnittstelle zur Verfügung stellen. Das Schnittstellenprotokoll muss dabei so gestaltet sein, dass es, unabhängig von der Sensorart, die Kommunikation über die Schnittstelle ermöglicht.

Das Ausgangsszenario bildet das häusliche Umfeld und deren Bewohnerin Karin ab. Hierfür muss die Kommunikation zwischen Sensoreinheit und Schnittstelle drahtlos erfolgen, um etwaige Stolperfallen in Form von Kabeln für die Bewohnerin zu vermeiden.

4.3.3 Anforderungen an die Entwicklungsbausteine

Jeder Entwicklungsbaustein enthält eine Zielstellung, das heißt eine konkrete Aufgabenstellung. In Abhängigkeit des Schwierigkeitsgrades, ist jeder konzipierte Entwicklungsbaustein an eine bestimmte Zielgruppe gerichtet. Dies können bspw. Studierende des Bachelorstudiengangs Telematik sein.

Zusätzlich muss der jeweilige Entwicklungsbaustein die Information enthalten, in welchem Rahmen dieser durchgeführt werden soll. Dazu können bspw. Abschlussarbeiten, Belegarbeiten etc. zählen.

Darüber hinaus muss der Umfang des Entwicklungsbausteins auf den jeweiligen Rahmen entsprechend abgestimmt sein.

Kapitel 5

Lösungskonzeption der intelligenten Verarbeitungseinheit

In diesem Kapitel werden die einzelnen Schritte für die Konzeption der IVE beschrieben. Die IVE wird nachfolgend auch unter dem Begriff *Smarter Home* beschrieben.

Im ersten Teil werden bereits existierende verwandte Anwendungen und Lösungen vorgestellt. Dabei wird selektiv insbesondere auf die Teile des jeweiligen Projektes eingegangen, die im Folgenden für die konkrete Konzeption der IVE besonders relevant sind.

Anschließend werden die einzelnen Funktionen des Systems mit Hilfe von Anwendungsmodi beschrieben, wodurch der grundsätzliche Aufbau der Anwendung klar wird.

Darauf aufbauend werden danach die Werkzeuge für die Realisierung der einzelnen Teilaufgaben evaluiert, erläutert und ausgewählt. Im letzten Abschnitt erfolgt noch einmal eine Zusammenfassung der Systemarchitektur in Form einer Gesamtübersicht.

Die in Kapitel 4 definierten Anforderungen für die IVE *Smarter Home* werden im Laufe dieses Abschnittes durch entsprechende Funktionen abgebildet. Gekennzeichnet wird dies, indem nach der Beschreibung einer Funktion die betreffenden Anforderungen in Form der ID¹ in Klammern genannt werden. Beispiel: Die ID *R001* steht für die erste Anforderung aus der tabellarischen Anforderungsanalyse, einzusehen in Anhang A.

5.1 Related works - Analyse einer Auswahl wichtiger Systeme

In diesem Abschnitt werden bereits bestehende Systeme und Objekte vorgestellt, die für die Umsetzung der IVE interessant sein können. Dabei handelt es sich um Systeme, die ähnliche Aufgaben erfüllen, wie bspw. das Monitoringsystem *SAMDY*, oder um vielversprechende Lösungsansätze, die für die Problemstellung dieser Arbeit interessant sind.

5.1.1 Das Monitoringsystem SAMDY

Bei dem Projekt *SAMDY*², handelt es sich um ein sensorbasiertes adaptives Monitoringsystem für die Verhaltensanalyse von Senioren.

¹ ID: Identifikationsnummer

² Das Projekt SAMDY:

<https://www.erfahrung-ist-zukunft.de/SharedDocs/Artikel/Engagement/Betreuung-Pflege-von-Angehörigen/20121220-assistenzsysteme-samdy.html>

Dieses Projekt wurde im Kontext des nationalen AAL¹-Programms des Bundesministeriums für Bildung und Forschung gefördert und hatte eine Laufzeit von gut 3 Jahren (1.12.2009 bis 31.01.2013). Geleitet wurde das Projekt von Marco Munstermann am Fraunhofer IMS² in Duisburg.

Ziel dieses Projektes ist die Unterstützung von Pflegekräften bei der Betreuung von hilfsbedürftigen Personen. Zusätzlich soll eine Entlastung für das Pflegepersonal durch das Abnehmen von Dokumentations- und Verwaltungsaufgaben erzielt werden.

Hierfür werden angebrachte Sensoren im Wohnraum, wie bspw. Bewegungsmelder, Kontakte an Wohnungs-, Kühlschrank- und Ofentüren, via Funk mit einer sogenannten Homestation verbunden. Diese befindet sich ebenfalls vor Ort in der Wohnung und erfasst die Daten, bereitet sie auf und gibt diese über eine Kommunikationsverbindung an das Pflegepersonal weiter. Die gewonnenen Daten werden direkt für die Dokumentation der Pflegeleistungen genutzt und für die Abrechnung automatisch erfasst.

Zusätzlich ist es möglich, mit Hilfe dieser erhaltenen Sensordaten, Abweichungen im Verhalten der zu pflegenden Person zu erkennen und das Pflegepersonal darüber in Kenntnis zu setzen. Dafür wird ein normales Verhalten in Form einer Wochenstruktur des Tagesablaufes der zu pflegenden Person erhoben und in die Homestation eingegeben. Anschließend werden die Live-Daten mit hinterlegten Regeln abgeglichen und das Pflegepersonal informiert, sobald etwas von der Norm abweicht.

Hierfür wird zunächst ein Abbild des aktuellen Verhaltens (AV) anhand der Sensorik bestimmt. Zusätzlich wird ein mittleres Verhalten (MV), das sich aus einem Zeitfenster an bisher aufgezeichneten Verhalten zusammensetzt, als Referenz berechnet. Anschließend wird das Delta, also die Differenz, zwischen AV und MV ermittelt, dem sogenannten Differenzverhaltensabbild (DV). Anhand dessen können Abweichungen im Verhalten der beobachteten Klienten erkannt werden, insbesondere Notsituationen. Liegt eine Person bspw. morgens länger als üblich in ihrem Bett, kann dies von der Anwendung erkannt werden und nach einer individuell abzustimmenden Zeit, ein Kontrollanruf erfolgen, um zu hören, ob alles in Ordnung ist. Reagiert die Person jedoch auf keine Kontaktaufnahme, kann sehr schnell die Hilfskette in Gang gesetzt werden.

Die Realisierung des Systems erfolgte mit Hilfe von sogenannten *Heuristics Miner* (HM) und *Petrinetzen*^g. Der HM zählt zur Familie der *Process-Miner*. *Process-Miner* gehören zu einer speziellen Form des *Data-Minings* und werden zur Generierung und Modellierung von Wissen anhand von definierten Ereignisprotokollen (auch Event-Logs) eingesetzt. [49, S. 72] Der HM zeichnet sich durch einen speziellen Umgang mit Fehlern im Event-Logging aus, da hier durch Schwellwerte gewisse Toleranzen möglich werden. Diese Toleranzen sind gerade bei der Beobachtung von Tagesabläufen sinnvoll. [49, S. 78]

Zusätzlich werden für die weitere Analyse *Petrinetze* eingesetzt, in denen die Ergebnisse der HM zusätzlich ausgewertet werden.

Wichtig für die Funktionsweise des Systems ist ein relativ regelmäßiger Tagesablauf der zu pflegenden Person. [20]

¹ AAL: Ambient Assisted Living

² Fraunhofer IMS: Fraunhofer-Institut für Mikroelektronische Schaltungen und Systeme

5.1.2 CASAS: A Smart Home in a Box

Bei dem *CASAS* Smart-Home-Projekt handelt es sich um ein multidisziplinäres Forschungsprojekt der *Washington State University* für die Entwicklung von intelligenten Wohnumgebungen. Das Projekt ist 2007 gestartet, über das Projektende sowie den aktuellen Status des Projektes konnten keine weiteren Informationen gefunden werden.

Dabei wird der Ansatz verfolgt, die Wohnung als intelligenten Agenten zu betrachten. Dieser Agent ist dazu in der Lage, das Verhalten der Bewohner anhand von Sensordaten zu erkennen, zu beurteilen und entsprechend darauf zu reagieren.

Ein Anwendungsfall hierfür ist die Beobachtung von älteren Menschen in ihrer Wohnung, um etwaige Unfälle zu erkennen und Hilfe herbeizurufen. [13]

Die Beobachtung von Personen innerhalb einer Wohnung erfolgt in Form einer Aktivitätserkennung. Dafür senden angebrachte Sensoren ihre Daten an eine Verarbeitungseinheit, welche wiederum versucht, mit Hilfe einer Sequenz an Sensordaten, eine Aktivität zu erkennen. Hierfür gibt es bestimmte vordefinierte Klassen für mögliche Aktivitäten.

Dabei verfügt jeder Sensor über eine Kennung, aus der ersichtlich wird, um was es sich für einen Sensor handelt und wo dieser eingesetzt wird.

In diesem Fall handelt es sich um einen Bewegungssensor, der an dem Bett der beobachteten Person angebracht ist. Neben der Sensorkennung wird zusätzlich das Datum und die Uhrzeit mitgesandt und natürlich der Sensorwert selbst:

```
2011-06-15 03:38:23.271939 BedMotionSensor ON
2011-06-15 03:38:28.212060 BedMotionSensor ON
2011-06-15 03:38:29.213955 BedMotionSensor ON
```

Diese Sequenz an Sensorwerten könnte beispielsweise der Aktivität *Schlafen* zugeordnet werden.

Für die Klassifizierung oder das Mapping von Sensordaten auf eine bestimmte Aktivität wurden bei der Entwicklung verschiedene Werkzeuge des maschinellen Lernens^g untersucht, wie die *Naiven Bayes-Klassifikatoren*^g, die *Hidden Markov Models*^g, die *Conditional Random Fields*^g oder die *Support Vector Machines* (SVM). [13]

Dabei stellte sich heraus, dass SVMs im Vergleich die beste Performanz aufweisen und darüber hinaus als einzige Lösung die Möglichkeit bieten, den Grad der Passgenauigkeit zwischen den Daten und den Aktivitäten zu bestimmen. Dies ist gerade für das Erkennen von Anomalien in der Verhaltensweise der beobachteten Personen hilfreich. [13]

SVMs sind mathematische Modelle, die eine automatische Klassifizierung von Objekten sowie die Entwicklung von Klassengrenzen ermöglichen. Diese Grenzen sind möglichst weit voneinander entfernt und die Objekte können anhand ihrer Eigenschaften den Klassen zugeordnet werden. Sie fallen in die Kategorie des überwachten Lernens. [18]

In Abbildung 5.1 ist die Architektur des *CASAS*-Systems dargestellt. Diese setzt sich aus den drei Schichten physikalische Schicht (blau), Middleware (rot) und der Anwendungsschicht (grün) zusammen. [13]

Während der Beobachtungsphase fließt der Steuerungsfluss, ausgehend von der physischen Schicht, über die Middleware zur Anwendungsschicht. Sobald eine Aktion von der intelli-

Abbildung 5.1: CASAS zeichnet sich durch eine leichtgewichtige 3-Schichten-Architektur aus.[13, Fig. 1]

genten Wohnung in Form eines Aktors initiiert wird, dreht sich der Steuerungsfluss um, von der Anwendungsschicht, über die Middleware zur physischen Schicht. Ziel der 3-Schichten-Architektur ist es, dass jede Schicht leichtgewichtig und erweiterbar ist. [13]

Die physikalische Schicht enthält die Hardware, wozu die Sensoren und Aktoren zählen. Die Architektur verwendet ein vermaschtes ZigBee-Netz, das direkt mit den Hardware-Komponenten kommuniziert.

Ein Publish/Subscriber-Manager regelt die mittlere Schicht (Middleware), hierfür bietet der Manager einen Broadcastchannel an, über die die Komponenten des Systems Nachrichten veröffentlichen und empfangen können.

Zusätzlich stellt die Middleware wertvolle Dienste zur Verfügung, worunter das Hinzufügen von Zeitstempeln zu Ereignissen oder die Zuweisung von universellen eindeutigen Identifikatoren gehört. [13]

In der Anwendungsschicht erfolgt die Feststellung von Aktivitäten sowie die Zuordnung zu einer bestimmten Aktivität.

Alle Komponenten von CASAS passen in eine kleine Box. Zu den Komponenten gehören Sensoren, die mit einem vorgesehenen Ort in der Wohnung markiert sind sowie die Middleware, die Datenbank und die Anwendungskomponenten, welche auf einem energiesparsamen Computer ausgeliefert werden.

Das System kann sowohl unabhängig und lokal laufen als auch extern mit Hilfe einer Datenbank in einer Cloud. [13]

5.1.3 Power Selekt Management

Power Selekt Management (PSM) ist ein Projekt, das sich im Bereich „Innovative Energiesysteme“ ansiedelt.

Das Projekt hat eine Laufzeit von einem Jahr und neun Monaten, gestartet ist es am 01.09.2015 und läuft noch bis zum 31.05.2017. Dabei zielt PSM auf einen neuartigen Energieverbund von Energiesystemen ab, bei dem mit Hilfe einer intelligenten Steuerungseinheit eine optimierte Erzeugung und Verteilung von verschiedenen Energieformen möglich wird.

In Anbetracht dieses Ziels soll eine Zentraleinheit entwickelt werden, die in Kombination von selbstlernender Software und innovativer Mess- und Regelungstechnik dazu in der Lage ist, den Eigenverbrauchsanteil effizient zu gestalten.

Das Ergebnis des Projektes soll im optimalen Falle die Möglichkeit bieten, mit Hilfe der eigenerzeugten Energie, den Gesamtenergiebedarf des Energieverbundes decken zu können, um so möglichst wenig Energie aus öffentlichen Netzen beziehen zu müssen.

Hierfür teilt sich das Projekt in zwei Teilprojekte auf:

Für die Umsetzung ist auf der einen Seite die *Technische Hochschule Wildau*, unter Leitung von Frau Prof. Dipl.-Inf. Birgit Wilkes und Herrn Prof. Dr. Rolle, verantwortlich und auf der anderen Seite die Firma *Conviva GmbH*¹ aus Berlin.

Der *Technischen Hochschule Wildau* obliegt dabei die Entwicklung von Algorithmen, mit denen es möglich ist, selbstlernende Lastprofilanalysen zu erstellen, die als Basis für langfristige Verbrauchsprognosen dienen.

Die *Conviva GmbH* hingegen übernimmt die Entwicklung der Hard- und Softwarekomponenten für das Mess-, Steuerungs- und Regelungssystem.

In Anbetracht der Konzeption der Anwendung *Smarter Home*, ist vor allem der Teil zur Entwicklung eines selbstlernenden Algorithmus interessant, um anhand von vorhandenen Lastprofilen dazu in der Lage zu sein, Prognosen für den zukünftigen Verbrauch treffen zu können.

Dabei hat die intelligente Software die Aufgabe, unter Einbeziehung von meteorologischen Daten und mit Hilfe von analysierten Verbrauchsprofilen, die regenerativ erzeugte Energie vorherzusagen und die Verteilung an die Verbraucher, unter Berücksichtigung des bisherigen Bedarfs, zu bestimmen.

Für die Prognostizierung mit Hilfe von intelligenten Algorithmen werden zwei grundsätzliche Verfahren angewandt. Zum einen werden statistische Analyseverfahren in Form einer Regressionsanalyse verwendet und zum anderen dienen angelernete KNN für das Prognostizieren der erzeugten Energie und des Verbrauchs.

Im Moment befindet sich das Projekt noch in der Entwicklungsphase, es gibt jedoch schon erste Erfolge. So hat sich der Einsatz von KNN bisher als sehr geeignet zur Bestimmung von kurzfristigen Prognosen erwiesen. Aktuell konnte für kurzfristige Prognosen, d.h. 24 Stunden im Voraus, eine Fehlerquote von 2% erzielt werden.² Für das Trainieren des KNN wird dabei der sogenannte Levenberg-Marquart-Algorithmus⁵ verwendet.

1 Conviva GmbH: <http://www.conviva-gmbh.de/>

2 Interview mit Marcel Block, Mitarbeiter des Projekts PSM, vom 19.01.2017

5.1.4 Schlussfolgerung

Für die in Abschnitt 5.4 geplante Analyse und Auswahl einer geeigneten intelligenten Software muss zunächst grundsätzlich entschieden werden, ob bei der Umsetzung der klassische oder der konnektionistische Ansatz an Werkzeugen der KI verwendet werden soll.

Bei den in Abschnitt 5.1.1 und 5.1.2 vorgestellten Lösungen werden, für die Umsetzung von sehr ähnlichen Projekten, regelbasierte Ansätze (*SAMDY*) oder auch Klassifizierungen (*CASAS*) zur Umsetzung verwendet.

Das in Abschnitt 5.1.3 vorgestellte Projekt setzt für die Erstellung einer Bedarfsprognose des Energieverbrauchs KNN ein, also ein Werkzeug der konnektionistischen KI. Hierbei dient das KNN zur Bestimmung von Prognosen anhand von bisherigen Verbrauchsprofilen und Wetterdaten.

Auf den ersten Blick scheinen die beiden Anwendungsfälle der zu entwickelnden IVE und des Projektes PSM nicht viel gemein zu haben. Bei genauerer Betrachtung bestehen jedoch Parallelen zwischen den beiden Anwendungsszenarien.

Auch die IVE hat grundsätzlich die Aufgabe, mit Hilfe von Trainingsdaten, die das Alltagsverhalten der zu beobachtenden Person darstellen, eine Prognose dahingehend zu stellen, wie das Verhalten zum aktuellen Zeitpunkt im Normalfall aussehen sollte. Diese Prognose wird mit den aktuell erhaltenen Sensordaten abgeglichen und bewertet.

Dabei können grobe Abweichung von der Norm erkannt und als Information entsprechend über eine Schnittstelle bereitgestellt werden.

Bei den Untersuchungen in diesem Abschnitt wurden mit *SAMDY* und *CASAS* zwei Projekte vorgestellt, die einen sehr ähnlichen Anwendungsfall wie die IVE verfolgen. Zusätzlich konnte ein Projekt vorgestellt werden, das vom Anwendungsfeld eine andere Sparte bedient, jedoch trotzdem Parallelen zur IVE aufweist und erfolgreich KNN für die Umsetzung einsetzt.

Bezüglich der IVE soll aufgrund der erhaltenen Erkenntnisse aus den vorgestellten Projekten die Implementierung mit Hilfe von KNN erfolgen. (R010) Hierbei gilt es herauszufinden, was in diesem Bereich mit KNN möglich ist und eventuelle Grenzen ausgelotet werden. Zusätzlich birgt der Umstand, dass bei den Recherchen in diesem Anwendungsfall keine ähnlichen Lösungen mit KNN gefunden werden konnten, außerdem ein gewisses Entwicklungspotential für zukünftige Beleg- oder Abschlussarbeiten in diesem Bereich.

Ebenfalls soll das *Out-of-the-box*-Prinzip aus dem *CASAS*-Projekt in ähnlicher Form übernommen bzw. adaptiert werden. Das heißt, es soll ein System entwickelt werden, das alle nötigen Ressourcen, wie etwaige Serverkomponenten etc., bereits enthält und keine zusätzliche Anbindung an ein externes oder öffentliches Netzwerk nötig macht. Alle aufgezeichneten und generierten Daten bleiben zum Schutz der Privatsphäre in den eigenen vier Wänden. Eine unkomplizierte Installation und Inbetriebnahme soll ebenso ermöglicht werden. Mehr dazu in Abschnitt 5.5.

In dem vorgestellten Projekt *CASAS* ist anhand der Sensordaten eine Aktivitätserkennung möglich, sodass nicht nur Aussagen über bestimmte Sensoren und deren Werte getroffen werden können, sondern vielmehr bezüglich bestimmter Tätigkeiten.

Dieses Mapping von Sensordaten auf Tätigkeiten soll ebenfalls über die IVE *Smarter Home* realisierbar sein. Dadurch können bereitgestellte Information für *Serviceroboter* oder andere Akteure aussagekräftiger gestaltet werden, sodass hierdurch ein besseres Agieren möglich

wird.

Mit der Entwicklung der IVE soll zunächst die prinzipielle Durchführbarkeit getestet werden (*Proof of Concept*), das heißt, ob es grundsätzlich möglich ist, mit Hilfe von KNN das Alltagsverhalten von älteren Menschen zu erlernen und im Anschluss daran zu überwachen. Als Voraussetzung hierfür wird im Rahmen dieser Arbeit zunächst von einem regelmäßigen Tagesablauf der zu beobachtenden Person ausgegangen. Dies ist das Szenario, das auch in dem Projekt *SAMDY* abgedeckt wird. Es soll als erster Meilenstein genügen und für weitere zukünftige Entwicklungen dienen.

5.2 Anwendungsmodi für den Benutzer

In diesem Abschnitt wird das Lösungskonzept der IVE anhand der verschiedenen Anwendungsmodi für den Benutzer vorgestellt und begründet. Mit der Vorstellung der Anwendungsmodi kann ein guter Einstieg und Überblick in den grundsätzlichen Aufbau der Anwendung gegeben werden. So kann in den darauffolgenden Abschnitten anschaulicher auf die Konzeption der enthaltenen Funktionen und Komponenten eingegangen werden.

Insgesamt setzt sich die IVE *Smarter Home* aus drei Anwendungsmodi zusammen: dem *Discover-Modus*, dem *Trainings-Modus* und dem *Go Live-Modus*.

Diese drei Modi bilden die einzelnen Teilschritte ab, die grundsätzlich für die Erstellung und den Einsatz eines KNN nötig sind.

In den folgenden Unterabschnitten werden die drei Modi, analog zu der Reihenfolge ihrer Ausführung in der Anwendung, vorgestellt.

5.2.1 Discover-Modus

Der erste Modus ist der *Discover-Modus*, hier hat der Benutzer die Möglichkeit, die grundsätzliche Architektur des KNN festzulegen. Dazu gehören die zu verwendenden Sensoren und die Definition von Tätigkeiten zur Abdeckung des Szenarios.

Es erfolgt quasi das Mapping der Sensorik und Tätigkeiten auf das KNN.

Zusätzlich können hier bspw. die Anzahl der verdeckten Schichten oder die anzuwendenden Algorithmen bestimmt werden.

Für das Anmelden und Anlegen von Sensoren stehen dem Benutzer zwei Möglichkeiten zur Verfügung, er kann reale **und** simulierte Sensoren hinzufügen.

Dass der Benutzer auch simulierte Sensoren anlegen kann, soll eine komfortable Möglichkeit zum Testen von verschiedenen Szenarien mit unterschiedlichen Sensoren und Sensorwerten ermöglichen.

Zusätzlich bietet die Simulation eine gute Möglichkeit, um Randbedingungen zu testen und Unregelmäßigkeiten zu provozieren. Auch für das deterministische Testen, z.B. anhand von Benchmarks, bietet die Simulation eine komfortable Bedienungsmöglichkeit.

Jeder Sensor verfügt über eine eindeutige ID, einen Namen, einen Typ und einen Wert. Der Typ legt dabei fest, um was es sich für einen Sensor handelt (Temperatur, Luftfeuchtigkeit, an/aus etc.).

Reale Sensoren melden sich über eine Schnittstelle (siehe Abschnitt 5.3) an das System an und können nach erfolgreicher Anmeldung ihre Sensorwerte an das System schicken. Die si-

mulierten Sensoren hingegen werden von dem Benutzer selbst hinzugefügt. Jeder Sensor, der am System angemeldet ist, kann durch den Benutzer modifiziert und auch wieder entfernt werden.

Die Sensoren bilden, Bezug nehmend auf das KNN, die sogenannte Eingangsschicht. Jeder Sensor stellt dabei ein einzelnes Eingabe- bzw. Input-Neuron dar.

Neben der Sensorik selbst ist für das Erlernen von Verhaltensmustern und Tagesrhythmen die Zeit ein wichtiger Bestandteil. Viele Tätigkeiten, wie z.B. das Schlafen, müssen im Zusammenhang mit der Tageszeit betrachtet werden. Ebenfalls ist der aktuelle Wochentag in Abhängigkeit von Gewohnheiten ein weiterer wichtiger zu betrachtender Faktor. Aufgrund dessen kann neben den Sensoren ebenfalls die Uhrzeit (R031) und der Wochentag (R032) als Eingabe-Neuronen hinzugefügt werden. Auch hier hat der Benutzer die Möglichkeit, die Werte zu modifizieren oder echte Werte zu übernehmen. (R033) (R034)

Im *Discover*-Modus können zusätzlich Tätigkeiten definiert werden. Wie in Absatz 5.1.4 beschrieben, soll das KNN nicht in der Lage sein, Aussagen hinsichtlich einzelner Sensoren treffen zu können, sondern vielmehr bezüglich bestimmter Tätigkeiten, nämlich ob hierbei eine Unregelmäßigkeit vorliegt oder nicht.

Hierfür verfügt der Benutzer über die Option, diese Tätigkeiten anzulegen. Eine Tätigkeit setzt sich zusammen aus einer ID, einem Namen und einem Zustand (true / false).

Die Tätigkeiten repräsentieren im KNN die Ausgabeschicht, bzw. jede Tätigkeit stellt darin ein Ausgabe- oder Output-Neuron dar.

Neben der Konfiguration der Ein- und Ausgabeschicht ist es dem Benutzer ebenfalls möglich, die Netzwerkarchitektur des KNN zu gestalten.

Dazu gehört zum einen die Bestimmung der Anzahl an verdeckten Schichten (hidden layer) in dem KNN und zum anderen die jeweilig gewünschte Anzahl an Neuronen für die jeweilige verdeckte Schicht.

Für die Gestaltung der Architektur eines KNN zur Bewältigung einer bestimmten Aufgabe gibt es bisher kein universelles Erfolgsrezept. Aufgrund dessen ist die Entwicklung eines KNN ein iterativer Prozess, bei dem optimaler Weise verschiedene Architekturen ausgetestet werden, um ein gutes Ergebnis zu erzielen. [12, S. 198].

Genau diese Vorgehensweise, die ein Experimentieren mit verschiedenen Architekturen des KNN zulässt, soll mit dieser Funktion in der Anwendung ermöglicht werden.

Zur besseren Verständlichkeit soll das nachfolgende Beispiel einer logischen AND- und OR-Schaltung noch einmal den Prozess zur Erstellung des grundlegenden KNN im *Discover*-Modus verdeutlichen:

Für die Abbildung der logischen AND- und OR-Schaltung werden zwei Eingänge bzw. Sensoren erstellt, die den Wert 1 und 0 annehmen können (Abbildung 5.2 grün umrandet).

In der Ausgangsschicht werden zwei Output-Neuronen benötigt, um bezüglich einer AND- und einer OR-Schaltung jeweilig eine Aussage treffen zu können. Diese beiden Ausgänge sind in Abbildung 5.2 rot markiert.

Der Bereich zwischen der Eingangs- und Ausgangsschicht in dem KNN enthält in diesem Beispiel genau eine verdeckte Schicht (blau umrahmt in Abbildung 5.2) mit insgesamt vier verdeckten Neuronen. Die verdeckte Schicht ist mit den Eingangs- und Ausgangsneuronen

Abbildung 5.2: Im *Discover*-Modus wird das KNN vom Benutzer erstellt.

voll vermascht. Damit ist die grundsätzliche Struktur zur Abbildung einer logischen AND- und OR-Schaltung fertig erstellt. Im nächsten Schritt muss dieses erstellte KNN trainiert werden, sodass die Gewichtungen der einzelnen Neuronen entsprechend angepasst werden können (Abschnitt 5.2.2).

Abschließend für diesen Abschnitt sind in Abbildung 5.3 noch einmal die Funktionen des *Discover*-Modus zusammengefasst.

Die farblichen Umrandungen ordnen dabei die Funktionen den entsprechenden Prozessen in der Erstellung des KNN zu. Bei den grün umrandeten Funktionen handelt es sich um die Konfiguration der Eingabeschicht, bei den rot umrandeten um Einstellungen zur Ausgabeschicht und die blau umrahmte Funktion ermöglicht die Konfiguration der Netzwerkarchitektur.

Abbildung 5.3: Die einzelnen Funktionen des *Discover*-Modus ermöglichen das Abbilden eines zu beobachtenden Szenarios in ein KNN.

5.2.2 Trainings-Modus

Der *Trainings*-Modus bildet den zweiten Anwendungsschritt in der IVE. Hier wird das im *Discover*-Modus erstellte KNN mit Hilfe von Testdaten trainiert. An der Eingangs- und Ausgangsschicht sowie der Architektur des KNN kann in diesem Modus nichts mehr verändert werden. Lediglich die Feinjustierung der Gewichte und Schwellwerte der einzelnen Neuronen werden in diesem Abschnitt durch Test- bzw. Trainingsdaten angepasst.

Hierfür werden im optimalen Fall positive und negative Trainingsdaten verwendet. Dadurch wird dem Netz beigebracht, welche eingehenden Datenwerte zu einem positiven Zustand des Ausgangsneurons führen sollen und welche hingegen zu einem negativen.

Der Benutzer hat hier die Möglichkeit über die Anwendung reale oder simulierte Daten für das Training zu verwenden. Gerade für Daten, die in Abhängigkeit von Wochentag und Uhrzeit trainiert werden sollen, bietet die Möglichkeit zur Simulation einen großen (zeitlichen) Vorteil.

Da das Generieren von geeigneten Trainingsdaten sehr aufwendig werden kann, ermöglicht die Anwendung zusätzlich das Abspeichern und Laden des Zustandes eines trainierten Netzes (mehr dazu in Abschnitt 5.5.2).

Für die Trainingsdurchführung hat der Benutzer die Möglichkeit, zwei Abbruchbedingungen zu definieren. Zum einen kann er eine maximale Anzahl an Wiederholungen der Trainingseinheiten bestimmen und zum anderen eine Fehlerquote definieren. Sobald einer der beiden gesetzten Werte während der Trainingsphase erreicht wird, terminiert das Training. Wurde das Training aufgrund der Trainingsdurchläufe terminiert, bedeutet dies, dass die gewünschte Fehlerquote während des Trainings nicht erzielt werden konnte. Um dies zu ändern kann an mehreren „Rädchen“ gedreht werden. So kann bspw. die Qualität der Trainingsdaten verbessert, die Trainingsdurchläufe erhöht oder die Architektur und Algorithmen für das KNN abgeändert werden. Hier soll noch einmal auf den iterativen Prozess bei der Entwicklung eines geeigneten KNN hingewiesen werden.

In Abbildung 5.4 sind die Funktionen des *Trainings*-Modus noch einmal zusammengefasst.

Abbildung 5.4: Die einzelnen Funktionen des *Trainings*-Modus ermöglichen das Trainieren eines KNN.

Um das Beispiel der binären AND- und OR-Schaltung fortzuführen, hat der Benutzer nun die Möglichkeit, das erstellte KNN zu trainieren. Hierfür können die zwei Eingangsneuronen

insgesamt vier Konstellationen an Sensorwerten ('0' und '1') annehmen, ersichtlich in der Tabelle in Abbildung 5.5 in den Spalten *Input 1* und *Input 2*. Das heißt, es sind vier Trainingsdatensätze notwendig, um alle möglichen Eingangszustände abbilden zu können.

Das Ziel des Trainings besteht darin, dass das KNN dazu in der Lage ist zu erkennen, bei welchen anliegenden Eingangssignalen die AND- bzw. OR-Schaltung eine '1' bzw. eine '0' als Ausgangssignal zurückgibt. Mit Hilfe von negativen und positiven Trainingsdaten kann dem KNN beigebracht werden, welche Konstellationen zu welchen Ergebnissen bei den Schaltungen führen.

Hierbei wird für dieses Training die Konvention festgelegt, dass das Ergebnis '1' einer Schaltung durch einen positiven Trainingsdatensatz angelernt wird. Ein positiver Trainingsdatensatz wird in der Anwendung durch ein *true* gekennzeichnet. Die Konstellationen der AND- und OR-Schaltungen, die hingegen zu einer '0' als Ausgangswert führen, werden dem KNN durch negative Trainingsdatensätze (*false*) antrainiert.

Der erste Trainingsdatensatz, Zeile 1 in der Tabelle in Abbildung 5.5, hat für Input 1 und 2 die Eingangswerte '0' und '0'. Bei einer AND- sowie OR-Verknüpfung führen diese Werte zu dem Ausgangssignal '0'. In der Tabelle symbolisiert durch ein *false*.

Der zweite Datensatz enthält für Input 1 den Wert '0' und für Input 2 den Wert '1'. Bezüglich der AND-Verknüpfung hat dies einen Ausgangswert von '0' (*false*) zur Folge, bei der OR-Verknüpfung hingegen den Wert '1' (*true*).

Analog dazu werden die restlichen zwei Trainingsdatensätze definiert und anschließend kann der Benutzer das Training mit Hilfe der angelegten Datensätze durchführen.

Input 1	Input 2	AND-Verknüpfung	OR-Verknüpfung
0	0	false	false
0	1	false	true
1	0	false	true
1	1	true	true

Abbildung 5.5: Mit Hilfe von positiven (*true*) und negativen (*false*) Trainingsdatensätzen wird das KNN trainiert.

Nach der Durchführung des Trainings kann das KNN im *Go Live*-Modus getestet und eingesetzt werden.

5.2.3 Go Live-Modus

Der *Go Live*-Modus bietet für den Anwender die Möglichkeit, das erstellte und trainierte KNN in Betrieb zu nehmen und unter echten oder auch simulierten Bedingungen zu testen. Hierfür muss zwingend ein trainiertes KNN in der Anwendung zur Verfügung stehen, das für diesen Modus genutzt werden kann. (R043) Andernfalls kann der Modus nicht aufgerufen werden.

Im *Go Live*-Modus steht dem Benutzer eine Übersicht zur Verfügung, in der die einzelnen Ausgangsneuronen (Tätigkeiten) des KNN dargestellt sind mit der jeweiligen ID, dem Na-

men und dem **aktuellen** Wert. Dieser aktuelle Wert des Neurons repräsentiert hierbei den gegenwärtig detektierten Zustand des Ausgangsneurons anhand der anliegenden Eingangswerte. Dieser Wert liegt zwischen '0' und '1'. Der Wert '1' oder ein Wert in der Nähe von '1' bedeutet, dass das KNN einen positiven Zustand berechnet hat. Tendiert der Wert hingegen gegen '0' oder ist '0', dann symbolisiert dies hingegen, dass das KNN einen negativen Zustand detektiert hat.

Entsprechend dieses Wertes und der Tatsache, dass ein Ausgangsneuron für eine bestimmte Funktion oder Tätigkeit steht, können Rückschlüsse darüber gezogen werden, ob und wenn ja, für welches Verhalten eine Unregelmäßigkeit zur Zeit vorliegt.

Die aktuellen Zustände der einzelnen Ausgangsneuronen werden ebenfalls über eine Schnittstelle bereitgestellt. Hierüber können Clients, wie z.B. *Serviceroboter*, die einzelnen Werte abrufen (mehr dazu in Abschnitt 5.5.3). Dafür melden sie sich an der Anwendung *Smarter Home* an und haben anschließend Zugang zu den Daten. Jeder angemeldete Client wird in der Anwendung als Listener angezeigt und kann in einer entsprechenden Auflistung eingesehen werden. In Abbildung 5.6 werden die Funktionen des *Go Live*-Modus noch einmal zusammengefasst.

Abbildung 5.6: Während des *Go Live*-Modus hat der Anwender die Möglichkeit, den aktuellen Status des Netzes und die angemeldeten Listener einzusehen sowie die Sensorwerte der Eingangsneuronen zu modifizieren.

Angewandt auf das Beispiel mit der AND- und OR-Schaltung aus den vorangegangenen Abschnitten bedeutet dies, dass das KNN im *Go Live*-Modus Aussagen darüber treffen kann, bei welchen Eingangswerten das Ausgangssignal der Schaltung eine '0' oder eine '1' zur Folge hat. Abbildung 5.7 zeigt dies anhand des Beispiels für die Eingangswerte '1' und '0'. Das Ausgangsneuron, das die UND-Verknüpfung repräsentiert, hat den Wert '0'. Dies ist nicht als der direkte Wert des Ausgangssignals der AND-Verknüpfung zu verstehen, sondern dass das KNN für diese Eingangskonstellation, aufgrund der Unterscheidung zwischen positiven und negativen Trainingsdaten, einen negativen Zustand detektiert hat. Für die ODER-Schaltung hingegen wurde ein positiver Zustand erkannt, da die Eingangssignale '1' und '0' das Ausgangssignal '1' ergeben.

Abbildung 5.7: Das KNN kann im *Go Live*-Modus Aussagen darüber treffen, ob ein Zustand *true* oder *false* ist.

5.3 Schnittstellenbereitstellung Sensorik

Damit Sensoren sich mit der IVE verbinden und deren Sensorwerte verwendet werden können, wird eine drahtlose Schnittstelle mit einem entsprechenden Protokoll seitens der Anwendung *Smarter Home* bereitgestellt.

Der zu verwendende Funkstandard sollte hierfür einen niedrigen Energieverbrauch aufweisen, um so einen möglichst langen Batteriebetrieb zu ermöglichen. Hierfür wurden die beiden stromsparenden Funktechniken Bluetooth Low Energy (BLE) und ZigBee als mögliche Übertragungstechniken miteinander verglichen. Tabelle 5.1 zeigt die Gegenüberstellung. [38] [39]

Grundsätzlich sind beide Techniken gut geeignet als Übertragungsprotokoll für die Schnittstelle. BLE ist bei der Datenübertragung viermal so schnell im Vergleich zu ZigBee, dafür bietet ZigBee wiederum Vorteile in der Übertragungreichweite.

Die Übertragungsgeschwindigkeit ist für das angestrebte Szenario bei ZigBee völlig ausreichend, da lediglich wenige Sensordatenwerte in bestimmten Zeitintervallen übertragen werden müssen. Eine hohe Datenrate, wie diese bei einer Übertragung von Fotos oder Videos nötig ist, ist hier nicht erforderlich.

Größere Relevanz hingegen hat die Reichweite, denn die einzelnen Sensoreinheiten müssen optimaler Weise in den verschiedensten Ecken und Räumen der Wohnung verteilt werden und trotzdem in der Lage sein, die Sensorwerte an die zentrale IVE zu senden. Hinsichtlich dieses Merkmals schneidet ZigBee mit einer Reichweite von 30 bis 100 Metern besser ab, als BLE mit ca. 10 Metern. [38] [39]

Merkmal	BLE	ZigBee
Frequenz	2.4 GHz	868 MHz, 2.4 GHz
Datenrate	1 MBit/s	250 KBit/s
Reichweite	ca. 10 m	30 - 100 m
Energieverbrauch	max. 10mA	ca. 0,01 mA
Verbreitete Einsatzgebiete	Wearables	Smart Home

Tabelle 5.1: Die beiden Funkstandards *Bluetooth Low Energy* und *ZigBee* im Vergleich.

BLE als Funktechnik wird eher für den Einsatz bei Wearables¹, das heißt bei Technologien, die nur eine sehr geringe Reichweite verlangen, empfohlen. ZigBee hingegen wurde u.a. für den Einsatz im *Smart Home*-Bereich konzipiert und entwickelt. Aufgrund dessen ist auch die anteilige Verteilung von ZigBee in diesem Bereich im Vergleich höher. [23]

Für die Umsetzung der Schnittstelle wird daher zu einer Implementierung mit dem ZigBee-Standard tendiert. (R020) (R021) Hierüber haben die Sensoreinheiten die Möglichkeit, sich über einen seriellen Datenaustausch an die IVE anzumelden und anschließend Sensordaten zu verschicken. (R022) (R023)

5.4 Analyse und Auswahl einer intelligenten Software

Auf dem Markt gibt es bereits eine große Anzahl an Softwarelösungen für die Arbeit mit KNN.

Ähnlich groß wie die Vielfalt an Softwarelösungen ist auch die Möglichkeit der zu verwendenden Programmiersprachen.

Für die Umsetzung der intelligenten Software wird auf eine Implementierung in der Programmiersprache *Java* gesetzt.

In Anbetracht der zukünftig angestrebten Weiterentwicklungen des Projektes durch Studenten des Studiengangs Telematik, auch außerhalb des Rahmens dieser Arbeit, soll so ein möglichst geringer Einarbeitungsaufwand ermöglicht werden.

Java wird quasi als Grundlage in den Bachelor- und Masterstudiengängen der Telematik gelehrt und somit kann davon ausgegangen werden, dass die Beherrschung dieser Programmiersprache vergleichsweise hoch ist.

Bei der Auswahl einer geeigneten Software für die IVE werden zunächst drei verschiedene Lösungen vorgestellt, die eine Programmierung in Java unterstützen, und anschließend auf unterschiedliche Anforderungen hin verglichen, sodass die jeweiligen Vor- und Nachteile abgewägt werden können. Darauf aufbauend wird das geeignetste Tool für die Implementierung ausgewählt.

5.4.1 Neuroph

*Neuroph*² ist ein leichtgewichtiges Framework zur Erstellung und dem Trainieren von KNN in Java.

Entwickler der Software *Neuroph* ist *Zoran Sevarac*. *Sevarac* ist Assistenzprofessor in der Abteilung Software Engineering an der Fakultät für Organisationswissenschaften in der Universität in Belgrad. [60]

Bei *Neuroph* handelt es sich um eine Open-Source-Software, die unter der Apache 2.0 Lizenz³ steht. (R009) Diese Lizenz beinhaltet bspw. die wichtige Eigenschaft, dass die Software in jedem Umfeld frei genutzt werden darf. Dies ist für die Verwendung in dieser Arbeit ein wichtiges Kriterium. [60] Darüber hinaus handelt es sich um keine Copyleft-Lizenz, d.h. bei

1 Wearables: Computertechnologien, die am Körper getragen werden. Beispiel hierfür ist eine SmartWatch.

2 Neuroph: <http://neuroph.sourceforge.net/>

3 Apache Lizenz 2.0: <https://www.apache.org/licenses/LICENSE-2.0>

Benutzung von *Neuroph* in der eigenen Software-Anwendung ist es nicht nötig die Lizenz von *Neuroph* zu übernehmen.

Das aktuellste Release von *Neuroph* ist die Version 2.92.

Neuroph bietet für die Arbeit mit KNN eine Multicore- und GPU¹-Unterstützung an, [60] die gerade für das Arbeiten mit größeren Netzen eine positive Auswirkung auf die Performance erzielen kann.

Bei den Netzwerkarchitekturen werden viele Arten unterstützt, wie unter anderem das bekannte *Hopfield-Netz*^g oder das *Multilayer Perzeptron*^g mit *Backpropagation*^g.

Auch für die Lernmethoden bietet *Neuroph* eine komfortable Auswahl an Algorithmen, so wird der überwachte und unüberwachte Lernmodus unterstützt (siehe Abschnitt 2.3.5). Zusätzlich bietet *Neuroph* ein einfaches Framework, um Benchmark Tests durchzuführen. [60]

Neuroph verfügt über eine sehr gute Dokumentation in Form einer API² und zahlreichen Tutorials und Hello-World-Programmen auf dem Webportal *Sourceforge*. [60]

5.4.2 Encog Machine Learning Framework

*Encog*³ ist ein fortgeschrittenes Framework für maschinelles Lernen und wurde von *Jeff Heaton* ins Leben gerufen, einem indischen Datenwissenschaftler und PhD⁴-Studenten von der Universität in Washington.

Heute läuft das Projekt unter dem Dach der *Heaton Research, Inc and contributors*. [27]

Die Software *Encog* ist Open Source und steht ebenfalls, wie *Neuroph*, unter der Apache Lizenz 2.0 und ist damit in der Nutzung frei verfügbar. (R009) Die aktuellste Version von *Encog* ist 3.3. [27]

Für die Arbeit mit KNN bietet das Framework sowohl paralleles Rechnen mit mehreren Kernen (Multicore) als auch mit Grafikprozessoren (GPU) an. [27]

Bezüglich der Netzwerkarchitekturen für KNN bietet *Encog* neben den *Multilayer*- und *Hopfield*-Netzen im Vergleich zu *Neuroph* mehr Varianten an. Auch bezüglich des Trainings verfügt *Encog* im Vergleich über eine größere Auswahl an Methodiken, wie bspw. den *Backpropagation*-Algorithmus oder die *Resilient Backpropagation*^g. Im Gegensatz zu *Neuroph* wird zusätzlich der *Levenberg-Marquart-Algorithmus* unterstützt, der Algorithmus, der bereits erfolgreich in dem Projekt PSM aus Abschnitt 5.1 verwendet wird.

Neben der Arbeit mit KNN unterstützt das Framework zusätzlich SVM, also einen Lösungsansatz, der in dem Projekt *CASAS* aus Abschnitt 5.1.2 umgesetzt wurde. [27]

Dies bietet Potential hinsichtlich späterer Entwicklungsmöglichkeiten.

Vergleichbar gut ausgeprägt wie bei *Neuroph* ist die Dokumentation. Für die Benutzer von *Encog* werden ebenfalls eine API und eine große Anzahl an Tutorials und Hello-World-Programmen zur Verfügung gestellt. [27]

1 GPU: Graphics Processing Unit

2 API: Application Programming Interface

3 Encog Machine Learning Framework: <http://www.heatonresearch.com/encog/>

4 Phd: Doctor of Philosophy

5.4.3 JOONE

Bei dem letzten in diesem Abschnitt vorgestellten Framework für KNN handelt es sich um die Software *JOONE*¹.

JOONE wurde von *Paolo Marrone und dem Joone Team* entwickelt. [50]

Der Quelltext ist, wie auch bei den bereits vorgestellten Frameworks, Open Source.

JOONE steht jedoch nicht unter der Apache 2.0 Lizenz sondern unter der GNU LGPL², einer abgeschwächten Form der GNU GPL³, und ist damit ebenfalls frei. (R009)

Bezüglich des Copylefts besteht hier eine abgeschwächte Form: Die verwendete Software, die unter der LGPL steht, und eventuelle Anpassungen darin, müssen weiterhin die Kriterien der LGPL erfüllen. Die eigene entwickelte Softwarelösung hingegen nicht. Aktuell ist *JOONE* in der Version 2.0 RC⁴ verfügbar. [50]

Hinsichtlich der parallelen Verarbeitung wird das Rechnen mit mehreren Kernen (Multicore) von *JOONE* unterstützt. Das Arbeiten mit Grafikprozessoren hingegen ist nicht möglich. [50]

JOONE bietet ebenfalls für den Einstieg eine gute Dokumentation in Form von Tutorials und Beispielprogrammen. [50]

Bei den Netztypen oder -arten für das Arbeiten mit den KNN bietet *JOONE* vorwärts gerichtete Netze an, wie das *Perzeptron*^g, oder rekursive Netze, wie bspw. das *Elman-Netz*^g. Bei den Trainingsmethoden werden nur drei Methoden unterstützt, wozu z.B. die *Backpropagation* zählt. Der *Levenberg-Marquart-Algorithmus* wird auch hier nicht unterstützt.

Im Gegensatz zu den beiden anderen vorgestellten Frameworks *Neuroph* und *Encog*, wird *JOONE* zum aktuellen Zeitpunkt nicht mehr weiterentwickelt. Es ist jedoch weiterhin in der aktuellen Version verfügbar und kann verwendet werden. [50]

5.4.4 Schlussfolgerung

Abschließend für diesen Abschnitt werden die drei vorgestellten Frameworks für KNN noch einmal in Bezug auf relevante Kriterien verglichen. Darauf aufbauend wird eine Entscheidung für die am besten geeignetste Software getroffen und begründet.

In Tabelle 5.2 sind nachfolgend noch einmal die Funktionen und Eigenschaften der drei Frameworks gegenübergestellt. Ein „ $\ominus$ “ kennzeichnet in der Tabelle, dass diese Funktion nicht vorhanden ist, ein „ $\oplus$ “ symbolisiert hingegen eine unterstützte Funktion. Ein nicht umkreistes Pluszeichen „+“ zeigt anhand der Anzahl an, wie stark ein bestimmtes Merkmal ausgeprägt ist.

Wichtig für die Auswahl der Software war, dass deren Lizenzbestimmung die freie Nutzung erlaubt, ohne dass dabei Kosten oder zusätzliche Bedingungen entstehen. Dieses Kriterium haben alle drei Frameworks erfüllt.

Hinsichtlich einer guten Performance, gerade bei den Trainingseinheiten der KNN, ist die

1 JOONE: Java Object Oriented Neural Engine

2 LGPL: Lesser General Public License

3 GPL: General Public License

4 RC: Release Candidate

Möglichkeit für das parallele Rechnen mit mehreren Kernen oder mit Hilfe von GPUs ein wichtiges Kriterium. *Neuroph* und *Encog* unterstützen beide Varianten, mit *JOONE* dagegen ist das Rechnen mit Grafikprozessoren nicht möglich.

Hinsichtlich der Performance haben die Entwickler von *Neuroph* einen Benchmark Test¹ mit den drei vorgestellten Frameworks veröffentlicht. Dabei wurde für jedes Framework das gleiche Netz und der gleiche Trainingsalgorithmus verwendet, zusätzlich wurde bei keinem Framework das parallele Rechnen eingesetzt.

Als Gewinner des Benchmarks ging das Framework *Encog* hervor, welches 2-3 Mal so schnell wie *Neuroph* in den Tests abschnitt. *Neuroph* landete auf dem zweiten Platz und war dabei 5-6 Mal schneller als *JOONE*, dem letztplatzierten Framework.

Zwar wurden die Tests mit *Encog* 2.4² und *Neuroph* 2.5b³ durchgeführt, also nicht mit den aktuellen Versionen, trotzdem sind sie ein guter Indikator für eine Performancebewertung der drei Kontrahenten. *JOONE* wurde in der Version 2 RC1 getestet, also der aktuellsten Version, hier ist demnach sicher keine Weiterentwicklung hinsichtlich der Performance erfolgt. Die kompletten Daten zu dem Benchmarktest können unter [59] eingesehen werden.

Die Dokumentation ist bei allen drei Software-Lösungen gleich gut ausgeprägt, sodass die Möglichkeit einer einfachen und schnellen Einarbeitung bei allen drei Frameworks gegeben ist.

Bei den Features, wozu bei der Bewertung die unterstützten Netzwerkarchitekturen und Trainingsalgorithmen zählten, verfügt *Encog* über das vielfältigste Angebot. Gerade für das Forschen mit KNN ist es wichtig eine große Anzahl an Möglichkeiten zur Verfügung zu haben. Dabei unterstützt *Encog* alle Architekturen und Algorithmen, die von *Neuroph* und *JOONE* zur Verfügung gestellt werden und weitere Features darüber hinaus.

Als zusätzlicher negativer Aspekt im Bezug auf das Framework *JOONE* ist die Tatsache, dass dieses aktuell nicht mehr weiterentwickelt wird.

Software	Version	Lizenz	CPU	GPU	Dokumentation	Benchmark	Feature
Neuroph	2.92	Apache 2.0	⊕	⊕	+++	++	++
Encog	3.3	Apache 2.0	⊕	⊕	+++	+++	+++
JOONE	v2 RC1	LGPL	⊕	⊖	+++	+	+

Tabelle 5.2: Die drei vorgestellten Frameworks *Neuroph*, *Encog* und *JOONE* zur Erstellung und Verarbeitung von KNN noch einmal im Vergleich.

Für die Entwicklung der IVE wird das Framework *Encog* gewählt. Es hat hinsichtlich der bestehenden Anforderungen insgesamt am besten abgeschnitten und zusätzlich wird es bereits erfolgreich in dem Projekt *Power Selekt Management* (siehe Abschnitt 5.1.3) an der TH Wildau eingesetzt, sodass hier bei eventuellen Problemen auf bereits vorhandenes Fachwissen zurückgegriffen werden kann.

¹ Benchmark Test aus dem Jahr 2010.

² Encog 2.4, Release date: Juni 2010

³ Neuroph 2.5b, Release date: 29.12.2010

5.5 Das Prinzip „Out of the box“

Für die Umsetzung der Anwendung *Smarter Home* wird das Paradigma einer *Out-of-the-box-Lösung* angestrebt. Diese Lösung verfolgt grundsätzlich den Ansatz, dass das fertige Produkt jegliche erforderlichen Mittel zur Verfügung stellt, die zur Lösung eines bestimmten Problems nötig sind. [5]

In Bezug auf die zu entwickelnde Software bedeutet dies, dass diese nach der Installation ohne weitere Anpassungen sofort einsatzbereit ist und keine externen oder zusätzlichen Werkzeuge, Installationen oder Anpassungen benötigt werden.

Die nachfolgenden Abschnitte zeigen auf, wie dieser Ansatz für die Umsetzung der IVE berücksichtigt werden soll.

5.5.1 Gradle - Build-Management-System

„Building a software artifact is a complex process involving various activities such as compiling source code, running automated tests, packaging distributable files, and so on.“ [15, S. 1]

Die Erstellung einer Software (Build) ist ein komplexer Prozess, der sich aus unterschiedlichen Aktivitäten zusammensetzt, die abhängig von der Reihenfolge der Ausführung sind und eine ganze Menge an Abhängigkeiten beachten müssen. Diesen Vorgang bei jedem Build-Prozess manuell durchzuführen ist nicht nur umständlich sondern auch fehleranfällig.

Mit einem Build-Management-Systems ist es möglich, automatisiert Software aus Quelltexten, Bibliotheken und sonstigen Dateien zu erzeugen. Durch den Einsatz solch eines Systems kann dem Entwickler der aufwendige Prozess zur Erstellung einer Software abgenommen werden.

Darüber hinaus bieten heutige Build-Systeme die Möglichkeit, das Kompilieren, das Warten, das Testen und die Installation von Softwareprogrammen möglichst komfortabel und einfach zu gestalten.

Unter dem Gesichtspunkt ein System zu entwickeln, das den Gedanken einer *Out-of-the-box-Lösung* verinnerlicht, ist es sinnvoll den Entwicklern der Software genau solch ein Tool an die Hand zu geben, um die eben genannten Ziele bedienen zu können. [15]

Als das erste richtungsweisende Build-Tool in der Java-Umgebung gilt das im Jahr 2000 erschienene Werkzeug *Apache Ant*¹². Dieses Tool ist in der Programmiersprache Java implementiert und plattformunabhängig. Gesteuert wird *Ant* über eine Build-Datei in der Auszeichnungssprache XML³. [3]

Ant verfügt über keine Abhängigkeits- und Versionsverwaltung. Diese Funktionalität stellt hingegen das im Jahr 2002 erstmalig veröffentlichte Tool *Apache Maven*⁴ bereit, ein in Java implementiertes Build-System, das einen deklarativen Ansatz verfolgt:

1 Apache Ant: <http://ant.apache.org/>

2 Apache Ant: Apache Another Net tool

3 XML: Extensible Markup Language

4 Apache Maven: <https://maven.apache.org/>

Mit *Maven* ist die Verwaltung und Versionierung von Abhängigkeiten über ein Maven-Repository gelöst. Über dieses Repository können Java Archive Files (JARs) automatisch über das Internet geladen werden. Die Konfiguration erfolgt wie bei *Ant* über eine XML-Datei mit dem Dateinamen „*pom.xml*“.

Ein großer Nachteil von *Maven* besteht darin, dass der Prozess sowie die Konventionen sehr strikt gehalten sind. Dies hat zur Folge, dass jedes mit *Maven* aufgesetzte Programm den gleichen Aufbau hat und dies zu Kollisionen mit der Infrastruktur des Projektes selbst führen kann. [3]

Das Build-System *Gradle*¹ gibt es seit dem Jahr 2007 und wurde von dem Entwickler *Hans Dockter* ins Leben gerufen. Dieses Tool integriert die beiden Systeme *Ant* und *Maven* mit der Intention, deren jeweilige Vorteile zu nutzen. Zusätzlich soll die Konfiguration für den Benutzer so einfach wie möglich gestaltet werden ohne die Funktionalitäten dabei einzuschränken. [3]

Hierfür wurden bspw. bei der Entwicklung die beiden Programmierparadigmen „*Convention over Configuration*“^g und „*Don't Repeat Yourself*“^g berücksichtigt und umgesetzt.

Ersteres durch das Verwenden von Standardkonventionen im Build-Skript, die nur bei speziellen Abweichungen von der Norm eine Änderung bzw. Anpassung im Skript nötig werden lassen. Und Letzteres durch die Vermeidung von redundantem Code und der damit verbundenen Vermeidung von potentiellen Fehlerquellen. Eine zusätzliche Folge daraus ist, dass auf diese Weise das Build-Skript so kurz und übersichtlich wie möglich gehalten werden kann.

Durch die Einbeziehung beider Tools können die Funktionalitäten sowie der deklarative Ansatz von *Maven* mit der nötigen Freiheit von *Ant* erzielt werden. So werden bspw. in *Gradle* für das automatische Herunterladen und Konfigurieren von Abhängigkeiten die direkten Repositories von *Maven* weiter genutzt. [3]

Für die Beschreibung der zu bauenden Projekte verwendet *Gradle* die Programmier- und Skriptsprache *Groovy*. Mit dieser ist es möglich, Klassen, Methoden und Kontrollstrukturen für die Erstellung des Builds zu programmieren. Dies ist ein wesentlicher Vorteil gegenüber den Systemen *Ant* und *Maven*, bei denen die Konfiguration über XML erfolgt und Kontrollstrukturen bspw. nur sehr aufwendig zu realisieren sind. [3]

Für die Umsetzung der Anwendung *Smarter Home* wird das Build-Management-System *Gradle* verwendet.

5.5.2 Persistenz

Mit der IVE muss der Benutzer die Möglichkeit haben, ein KNN zu trainieren und zu testen. Das Trainieren bzw. das Erzeugen von Testdaten stellt, je nach Qualitätsanspruch an das KNN, einen mittleren bis großen Aufwand für den Anwender dar. Aufgrund dieser Tatsache soll die Anwendung die Möglichkeit besitzen, die Konfiguration des KNN, das KNN selbst und die Trainingsdaten abzuspeichern und diese bei Bedarf auch wieder zu laden.

Diese Funktion kann mit Hilfe eines nicht flüchtigem Speichermediums wie bspw. einer Datenbank oder auch einem Dateisystem realisiert werden.

¹ Gradle: <https://gradle.org/>

Bei der Lösung mit Hilfe einer Datenbank wäre der Datenbankserver *Derby*¹ von der Apache Software Foundation² eine geeignete Variante.

Apache Derby ist eine relationale Datenbank, die in Java implementiert und unter der Apache Lizenz Version 2.0 steht. Als Datenbanksprache verwendet *Derby* die weit verbreitete Structured Query Language (SQL) und liefert einen bereits integrierten JDBC^{3g}-Treiber mit, der die Einbettung der Datenbank in jede Java-basierte Anwendung ohne einen zusätzlichen externen Server ermöglicht.

Im Gegensatz zu anderen Datenbanklösungen, wie *PostgreSQL*⁴ oder *MySQL*⁵, kann die Datenbank *Derby* als interne Instanz direkt in der Anwendung gestartet werden und unterstützt damit den Leitgedanken einer angestrebten *Out-of-the-box-Lösung*.

Darüber hinaus ist *Derby* in seiner Basisvariante sehr schlank gehalten und benötigt wenig Speicherplatz. [69]

Die andere Lösungsalternative wäre für das einfache Abspeichern und Auslesen der Daten das jeweilige Dateisystem zu benutzen und die entsprechenden Informationen in einer Datei abzulegen.

Datenbanksysteme (DBS) sind für das Abspeichern und Auslesen von Daten grundsätzlich eine geeignete Lösungsmöglichkeit, denn das Abspeichern und Verwalten von Daten ist die zentrale Aufgabe eines DBS, hierfür bietet es eine ganze Menge an Funktionen an, wie bspw. den Mehrbenutzerzugriff, die Transaktionssicherheit bei der Änderungen bestehender Daten, die Datensicherheit etc.

In diesem speziellen Fall wird jedoch nur ein kleiner Bruchteil dieser Funktionen benötigt, sodass die bestehenden Anforderungen, das einfache Abspeichern und Auslesen von Daten, auch über ein einfaches Dateisystem zufriedenstellend gelöst werden können.

Die Verwendung einer Datenbank wäre in diesem Fall überdimensioniert. Zusätzlich kann ein KNN mit seiner variablen Anzahl an Eingangsneuronen, verdeckten Schichten und dazugehörigen verdeckten Neuronen sowie Ausgangsneuronen verschiedenste Gestalten annehmen. Die Abbildung dieser Variabilität an Daten ist mit einer Datenbank nur sehr viel aufwendiger zu realisieren, als im Vergleich zur Speicherung über ein Dateisystem. Darüber hinaus kann durch das persistente Abspeichern über ein Dateisystem ein einfacher und komfortabler Dateiaustausch erfolgen. Gerade in einem Schülerlabor, bei dem die Schüler z.B. unterschiedliche KNN testen und vergleichen sollen, ist diese Variante besser geeignet. So wäre es bspw. möglich, vortrainierte KNN den Schülern in Form einer Datei zur Verfügung zu stellen, die diese dann weiter optimieren sollen oder Ähnliches.

Der Austausch eines DBS oder deren Inhalt in eine andere Anwendung auf einem anderen Rechner wäre in diesem Fall sehr viel aufwendiger.

5.5.3 Jersey - RESTful Web Services in Java

Der in Abschnitt 4.2.3 beschriebene Anwendungsfall macht es erforderlich, dass die *Service-roboter* oder andere Aktoren über die Möglichkeit verfügen, sich einerseits mit der IVE zu verbinden und andererseits von ihr Informationen abzurufen. Diese Informationen müssen eine Aussage darüber enthalten, ob aktuell Auffälligkeiten in dem Verhalten der beobachte-

1 Apache Derby: <https://db.apache.org/derby/>

2 Apache Software Foundation: <https://www.apache.org/>

3 JDBC: Java Database Connectivity

4 PostgreSQL: <https://www.postgresql.org/>

5 MySQL: <https://www.mysql.de/>

ten Person festgestellt wurden.

Mit Hilfe dieser Informationen soll der *Serviceroboter* anschließend in der Lage sein, auf die Situation geeignet zu reagieren.

Die Funktion zur Bereitstellung dieses Servicedienstes wird mit Hilfe eines Webservices realisiert. Bei einem Webservice handelt es sich um eine plattformunabhängige Softwarekomponente, mit der es möglich ist, verteilte Anwendungen umzusetzen. Hierfür stellt der Webservice die Schnittstelle für den *Serviceroboter* bereit, über die dieser die benötigten Daten abrufen kann. (R015) (R016)

Die Verwendung eines Webservices hat den Vorteil, dass die Nutzung des Dienstes nicht auf eine bestimmte Anwendergruppe beschränkt wird. Durch die Nutzung des öffentlichen und weit verbreiteten plattformunabhängigen Standards, dem Übertragungsprotokoll HTTP¹, wird eine offene und flexible Architektur bereitgestellt, die in weiteren Entwicklungsschritten auch von jedem anderen HTTP-fähigen Gerät genutzt werden kann. Für einen Produktiveinsatz der Anwendung ist das Protokoll HTTP sicherlich nicht ausreichend, vielmehr sollte in diesem Fall eine Übertragung über das gesicherte bzw. verschlüsselte Kommunikationsprotokoll HTTPS² erfolgen.

Für die konkrete Implementierung des Webservices wird auf das Architekturmodell *REST*³ zurückgegriffen. Das REST-Modell wurde bereits erfolgreich auf den beiden Plattformen des *NAO*- und *Pepper*-Roboters umgesetzt. Aufgrund dessen kann davon ausgegangen werden, dass eine Integration des REST-Services auf beiden Robotern für zukünftige Entwicklungen ebenfalls möglich ist.

Das REST-Modell ist in der Lage einen Webservice mit Hilfe von HTTP-Befehlen zu beschreiben. Dabei wird bei *REST* jedes Objekt und jede Datei, die über einen Uniform Resource Identifier (URI) adressierbar ist, als eine Ressource auf dem Webserver betrachtet. [71, S. 1165ff]

Der Client kann solch eine Ressource mit Hilfe der HTTP-Befehle GET, PUT, POST und DELETE abrufen, abändern, hinzufügen und löschen.

Der Web- bzw. REST-Server kann wiederum die jeweilig angeforderte Ressource in den verschiedenen Formaten JSON⁴, XML und HTTP an den Client ausliefern. [71, S. 1165ff]

Java stellt für die Umsetzung von RESTful Web Services die Spezifikation JAX-RS⁵ zur Verfügung. Diese Programmierschnittstelle ermöglicht die architektonische und einheitliche Umsetzung des REST-Modells.

Für die Umsetzung der RESTful Web Services wird das Framework *Jersey* verwendet. Dieses wurde eigens zur Implementierung von RESTful Webservices entwickelt, hierfür bedient das Framework die Programmierschnittstelle JAX-RS und dient ebenfalls als dessen Referenzimplementierung⁸. [71, S. 1165ff]

Ein *Serviceroboter* oder auch jedes andere HTTP-fähige Gerät hat dann die Möglichkeit, sich an der IVE anzumelden (R017) und bei Bedarf die nötigen Informationen abzurufen.

1 HTTP: Hypertext Transfer Protocol

2 HTTPS: Hypertext Transfer Protocol Secure

3 REST: Representational State Transfer

4 JSON: JavaScript Object Notation

5 JAX-RS: Java API for RESTful Web Services

Dazu zählen die Eingangs- und Ausgangsneuronen und deren jeweiligen Eigenschaften, die aktuell im System hinterlegt sind. (R018) (R046) Nach Bedarf bietet die Schnittstelle auch die Möglichkeit das Gerät von der Anwendung wieder abzumelden. (R019)

5.6 Graphische Benutzeroberfläche

Zur Bedienung der Anwendung *Smarter Home* steht dem Anwender eine Benutzeroberfläche zur Verfügung, in der alle bereitgestellten Funktionen der Applikation genutzt werden können. (R011) Nachfolgend wird auf das zu verwendende Framework, das angestrebte gestalterische Design und den geplanten Aufbau für die grafische Benutzeroberfläche eingegangen.

5.6.1 JavaFX

Für die Umsetzung der graphischen Oberfläche wird das Framework *JavaFX* verwendet. Mit Hilfe dieses Frameworks ist es möglich, moderne grafische Oberflächen zu entwerfen. *JavaFX* wird mit Java 8 als empfohlenes Tool für die Erstellung von Benutzeroberflächen ausgeliefert und löst damit die beiden Vorgänger AWT¹ und Swing ab. Neben der Erstellung der Benutzeroberfläche über die API von *JavaFX* selbst, wird zusätzlich ein deklarativer Ansatz über die XML-basierte Auszeichnungssprache FXML angeboten.

Für das Design der Benutzeroberfläche wird das von Oracle bereitgestellte grafische Tool *JavaFX Scene Builder*² verwendet. Dieses ermöglicht die komfortable Erstellung einer grafischen Oberfläche und generiert eine FXML-Datei, die in jeden Java-Quellcode eingebunden werden kann. [71, S. 1031ff]

Eine Besonderheit von *JavaFX*, im Gegensatz zu seinen Vorgängern, ist die hierarchische Datenstruktur in Form eines sogenannten Szenegraphen.

Dieser nimmt alle grafischen Objekte, wie Linien, Flächen, Textfelder, 3D-Objekte etc. auf und verwaltet diese. All diese Elemente sind während einer Anwendung mit *JavaFX* im Speicher präsent. Gerade bei großen Datenmengen muss dieser Umstand bei der Umsetzung und Programmierung beachtet werden. [71, S. 1031ff]

5.6.2 Responsive Design

Das Responsive Design ist ursprünglich aus der Webprogrammierung bekannt und wird in diesem Bereich unter dem Begriff *Responsive Web Design* gehandelt.

Dabei verfolgt dieses Design-Paradigma die Intention, dass die Webseite an die jeweilige Größe des Ausgabemediums dynamisch angepasst wird. Erforderlich wurde diese Art der Gestaltung von Webseiten vor allem durch die immer größere Vielfalt in Art und Displaygröße der internetfähigen Endgeräte:

„A web page should look good, and be easy to use, regardless of the device!“ [74]

Hierfür passt sich die Webseite durch ein Verschieben, Vergrößern und Verkleinern des Inhalts an das jeweilige Endgerät an und ermöglicht stets eine akkurate Darstellung unabhängig von

1 AWT: Abstract Window Toolkit

2 JavaFX Scene Builder: <http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html>

der Bildschirmgröße. [74]

Für die Umsetzung der graphischen Benutzeroberfläche soll quasi ein ähnliches Konzept umgesetzt werden. Hierfür werden die Elemente in der Bedienoberfläche in *JavaFX* so gestaltet, dass diese sich an die jeweilige Fenstergröße dynamisch anpassen. Damit soll die Anwendung eine gute Übersicht und Lesbarkeit für den Benutzer ermöglichen.

5.6.3 Aufbau der Benutzeroberfläche

Der Aufbau der Benutzeroberfläche orientiert sich bei der Gestaltung an den in Abschnitt 5.2 beschriebenen Anwendungsmodi und deren Funktionen.

Hierfür hat der Benutzer über die drei Schaltflächen (in Abbildung 5.8 oben dargestellt) *Discover*, *Training* und *Go Live* (R012) (R013) (R014) die Möglichkeit, die jeweiligen Anwendungsbereiche zu aktivieren bzw. zu deaktivieren. (R024) (R025) (R036) (R037) (R041) (R042) Zu einem Zeitpunkt ist immer genau ein Anwendungsmodus aktiv. Je nach Modus werden lediglich die dafür benötigten Bedienelemente auf der Oberfläche freigeschaltet und alle nicht dazugehörigen entsprechend deaktiviert.

Dies ermöglicht einerseits eine bessere Übersicht und Orientierung in der Anwendung selbst und schützt andererseits den Benutzer vor möglichen Fehleingaben bei der Bedienung.

In Abbildung 5.8 ist eine Gesamtübersicht der Benutzeroberfläche dargestellt, in ihr sind alle Bedienelemente aufgezeigt. Nachfolgend wird auf die Darstellung der einzelnen Modi in der Benutzeroberfläche eingegangen. Dabei werden nur die Elemente angezeigt, die auch in dem jeweiligen Modus aktiv sind.

Die Anwendung wird in dem Modus *Discover* gestartet.

Abbildung 5.8: Die grafische Oberfläche führt den Benutzer durch die drei Anwendungsbereiche *Discover*, *Training* und *Go Live*.

Die aktiven Bedienelemente des *Discover*-Modus sind in Abbildung 5.9 aufgezeigt.

Dieser Modus ist für den Aufbau des KNN zuständig.

Alle aktiven Eingangsneuronen des KNN werden in der Tabelle *Input Neurons* mit ID, Name, Typ und Wert im *Discover*-Bereich dargestellt. (R026) (R028) Über den Toggle-Button *Live data* werden die aktuellen Werte der realen Sensoren dargestellt und aktualisiert. Der Toggle-Button *Override* ermöglicht bei Aktivierung dem Benutzer das Verändern der Werte in der Tabelle (Simulation). (R029) (R030)

Über die beiden Buttons *Add* und *Remove* lassen sich Eingangsneuronen entfernen oder neue hinzufügen. (R027) (R035)

Die zweite Tabelle *Output neurons* listet die aktuellen Ausgangsneuronen mit ID, Namen und Value auf. Sie können über die darunter befindlichen Buttons *Add* und *Remove* hinzugefügt bzw. entfernt werden können, solange der *Discover*-Bereich aktiv ist.

Der Button *Configure Network* öffnet einen modalen Dialog zur Einstellung der verdeckten Schichten im KNN.

Abbildung 5.9: Der Bereich *Discover* ermöglicht dem Benutzer das Erstellen der grundlegenden Architektur des KNN.

Die Benutzeroberfläche, die den *Trainings*-Modus abbildet, ist in Abbildung 5.10 ersichtlich. Im *Trainings*-Modus hat der Anwender in der Benutzeroberfläche die Möglichkeit, Trainingsdatensätze für ein Training anzulegen. Hierfür stehen ihm die Tabellen aus dem *Discover*-Modus zur Verfügung, bei dem die realen oder simulierten Sensorwerte verwendet werden können. In der Tabelle *Output neurons* wird über den Value-Wert festgelegt, ob diese Konstellation der Sensorwerte als ein normales (true) oder nicht normales (false) Verhalten bezüglich der jeweiligen Tätigkeit gewertet werden soll. Der Button *Configure* ermöglicht das Festlegen der maximalen Anzahl an Wiederholungen der Trainingseinheiten sowie die angestrebte Fehlerquote. Über den Button *Do Training* wird das Training gestartet. (R040)

Das Laden und Abspeichern von Trainingsdatensätzen, KNN und der Konfiguration stehen dem Benutzer über die Buttons *Load* und *Save* zur Verfügung. (R038) (R039)

Abbildung 5.10: Der Bereich *Training* ermöglicht dem Benutzer bspw. das Anlegen von Trainingsdatensätzen und die Durchführung eines Trainings.

Den letzten Anwendungsbereich bildet der *Go Live*-Modus, die Benutzeroberfläche hierfür ist in Abbildung 5.11 ersichtlich. Hier sind alle angemeldeten Listener in der Tabelle für den Benutzer einsehbar. (R047) Die Tabelle *Output neurons* listet nochmals alle Ausgangsneuronen des KNN auf, die die erstellten Tätigkeiten repräsentieren. Die Spalte *Value* hat jedoch eine andere Funktion als im Modus *Training*: Sie gibt darüber Auskunft, welchen Zustand das KNN bezüglich der Tätigkeit aktuell festgestellt hat. Ist dieser Wert nahe '1', dann kann dies als Unregelmäßigkeit bezüglich der Tätigkeit interpretiert werden. Tendiert der Wert hingegen gegen '0', so ist eine Diskrepanz zwischen normalem (antrainiertem) Verhalten und dem aktuellen Verhalten durch das KNN entdeckt worden. (R045) Noch einmal beziehend auf Abbildung 5.11 ist hinsichtlich der Tätigkeit *Kochen* also alles in Ordnung, beim Schlafverhalten ist dies jedoch offensichtlich nicht der Fall. (R044)

Über die Tabelle *Input neurons* hat der Benutzer auch während des *Go Live*-Modus die Möglichkeit, die Sensorwerte zu manipulieren. Dies ermöglicht bspw. das Testen der Qualität des trainierten KNN durch das Forcieren eines nicht normalen Verhaltens mit Hilfe der Werte der Eingangsneuronen.

Abbildung 5.11: Im *Go Live*-Bereich geht die Anwendung in den Live-Modus und der Benutzer kann den aktuell detektierten Zustand der Anwendung einsehen.

5.7 Zusammenfassung der Systemarchitektur

Abschließend für die Konzeption soll noch einmal in Form einer kurzen Zusammenfassung ein Gesamtüberblick über die Systemarchitektur der Anwendung *Smarter Home* gegeben werden. Abbildung 5.12 dient als zusätzliche Übersicht.

Für die Sensorik, die in der Anwendung als Datenquellen dienen, stellt die IVE *Smarter Home* eine serielle Verbindung über den Funkstandard ZigBee zur Verfügung. Im Bereich Aktorik können sich beispielsweise *Serviceroboter* via REST-Schnittstelle die relevanten Informationen von der IVE abrufen und weiterverarbeiten.

Smarter Home wird in der Programmiersprache Java Version 8 umgesetzt. Das Prinzip „*Out of the box*“ soll eine möglichst unkomplizierte Inbetriebnahme und Weiterentwicklung der Anwendung ermöglichen. Hierfür wird das Dateisystem für die Persistenz von KNN und Trainingsdaten benutzt sowie eine eingebettete Serveranwendung für die Bereitstellung des Web Services verwendet. Zusätzlich wird das Build-Management-System *Gradle* eingesetzt. Zur Erstellung der Benutzeroberfläche wird das seit Java 8 empfohlene Framework *JavaFX* verwendet. Für die Arbeit mit KNN wird das Framework *Encog* eingesetzt, das sich im Vergleich zu den anderen untersuchten Java-basierten Frameworks, hinsichtlich der Performance und dem Umfang an Features, ab- bzw. durchsetzen konnte.

Die Anwendung selbst setzt sich aus den drei Anwendungsmodi *Discover*, *Training* und *Go Live* zusammen, die in dieser Reihenfolge auch den Ablauf zur Erstellung des KNN abbilden. Dabei stehen dem Benutzer, je nach aktuellem Modus, die unterschiedlichen Bedienungsfunktionen der Anwendung zur Verfügung.

Abbildung 5.12: Das Konzept zur Umsetzung der intelligenten Verarbeitungseinheit *Smarter Home* im Überblick.

Kapitel 6

Lösungskonzeption des Lernbaukastens und der Entwicklungsbausteine

In diesem Abschnitt stehen der geplante Lernbaukasten und die Entwicklungsbausteine im Fokus. Für die Konzeption des Lernbaukastens wird zunächst der Lernprozess an sich untersucht und nachfolgend eine Lehrmethode präsentiert, die sich an diesem Lernprozess orientiert. Im Anschluss daran wird das Konzept des Lernbaukastens und dessen Lernbausteine erläutert.

Abschließend werden Ideen für mögliche Entwicklungsbausteine vorgestellt, die aufbauend auf die Ergebnisse dieser Masterarbeit auf zukünftige Weiterentwicklungen durch Schüler und Studierende abzielen.

Auch in diesem Abschnitt werden auf die betreffenden Anforderungen im Anschluss an die beschriebenen Funktionen oder Komponenten mit Hilfe der jeweiligen ID in Klammern referenziert.

6.1 Der Lernprozess

Ganz allgemein formuliert besteht das große Ziel des Lernbaukastens darin, dass die Schüler mit Hilfe der darin enthaltenen Lernbausteine etwas über das Thema KI lernen.

Damit dieses Ziel erreicht werden kann, muss zunächst untersucht werden, wie der Lernprozess aufgebaut ist und anschließend eine passende Lernstrategie angewandt werden.

Das Lernen ist ein aktiver Prozess, bei dem die Lernenden neues Wissen aufbauen bzw. Handlungsmuster erwerben.

Bestehendes Wissen und bestehende Handlungsmuster werden in Form einer kognitiven Struktur vom Menschen repräsentiert. Beim Lernprozess werden das Wissen und die Handlungsmuster angepasst, indem eine aktive und konstruktive Restrukturierung der kognitiven Struktur durchgeführt wird.

Das Lernen erfolgt nicht nur dann, wenn man bewusst vorgibt zu lernen, sondern auch nebenbei:

Wird eine Person mit Aufgaben, Informationen oder Situationen, die diese nicht lösen kann, konfrontiert, dann kann ein inneres Ungleichgewicht entstehen. Dieses Ungleichgewicht ent-

steht genau dann, wenn die Person eine persönliche Relevanz oder ein persönliches Interesse für sich selbst erkennt. Die Folge ist der Versuch, das Ungleichgewicht durch einen entsprechenden Wissenserwerb wieder auszugleichen.

Dieser Prozess erfolgt in Form einer sogenannten Modellbildung, an dessen Ende ein mentales Modell steht, welches die neue Information für die Person verständlich macht. [25]

Diese mentale Modellbildung des Lernens lässt sich in fünf Teilprozesse unterteilen:

Im ersten Teilprozess wird ein mentales Ungleichgewicht ausgelöst (Teilprozess 1: Ungleichgewicht wird ausgelöst).

Anschließend wird bereits vorhandenes Wissen aktiviert. Mit diesem Vorwissen wird zunächst versucht das Ungleichgewicht aufzulösen. (Teilprozess 2: Vorwissen wird aktiviert).

Da dieses Vorwissen jedoch in der Regel nicht ausreicht, wird zusätzlich die Umgebung nach sachdienlichen Informationen durchsucht, um die Reizinformation verständlich zu machen (Teilprozess 3: Suchen).

Letztlich wird das Vorwissen, die Reizinformation selbst und die zusätzliche Information aus der Umgebung in Form eines mentalen Modells integriert (Teilprozess 4: Integrieren).

Damit ist die mentale Modellbildung abgeschlossen. Im anschließenden Prozess müssen wiederholt ähnliche mentale Modelle aufgebaut werden, damit sie schließlich in Form eines Schemas abgespeichert werden können. Dieser abschließende Teilprozess (Teilprozess 5: Elaborieren) umfasst folglich ein wiederholtes Durchlaufen der Teilprozesse 1 bis 4, wobei Teilprozess 3 bei jedem Durchlauf einen kleineren Stellenwert einnimmt. Das Elaborieren kann in Form von Wiederholen, Darübersprechen, Durchdenken, Hinterfragen, Aufschreiben, Üben etc. erfolgen. [25]

In Abbildung 6.1 sind die Teilprozesse des Lernens, der Reihenfolge nach, dargestellt.

Abbildung 6.1: Der Lernprozess setzt sich aus insgesamt fünf Teilprozessen zusammen, die nacheinander durchlaufen werden.

6.2 MOMBI - Model-of-Model-Based Instruction

Model-of-Model-Based Instruction (MOMBI) ist eine Lehrstrategie, die auf Basis der Erkenntnisse über das modellbegründete Lernen in der allgemeinen Erziehungswissenschaft entwickelt wurde. Dabei richtet sich MOMBI nach dem in Abschnitt 6.1 beschriebenen Prozess des Lernens und verfolgt dabei die Intention, durch entsprechende Lehrschritte, den Lernprozess bzw. dessen einzelne Teilprozesse gezielt hervorzurufen.

Hierfür wird für jeden Teilprozess des Lernens ein entsprechender Lehrschritt vorgeschlagen. Dazu zählen das *Provozieren*, das *Aktivieren*, das *Informieren*, das *Unterstützen* und das *Festigen*. [25] In Abbildung 6.2 sind diese Schritte im linken Kasten des Bildes dargestellt. Der große Pfeil im linken Kasten weist auf die Reihenfolge für die Abarbeitung der Lehrschritte hin. Die Pfeile zwischen den Kästen zeigen auf, auf welchen Teilprozess des Lernens (rechter Kasten) der jeweilige Lehrschritt von MOMBI abzielt.

Abbildung 6.2: Durch das zielgerichtete Abarbeiten der Lehrschritte nach MOMBI, werden die einzelnen Teilprozesse des Lernens angesteuert.

Der Lehrschritt *Provozieren* hat die Aufgabe, bei den Lernenden einen mentalen Zustand des Ungleichgewichts auszulösen und damit zum Lernen anzuregen. Dabei ist es essentiell wichtig, dass die Fragestellung oder das Problem für die Lernenden relevant ist. Nur so werden die Voraussetzungen dafür geschaffen, dass die Lernenden die Bereitschaft und das Interesse verspüren sich mit der Thematik zu beschäftigen. [25]

Der Lehrschritt des *Aktivierens* dient dazu, die bestehenden kognitiven Strukturen beim Lernenden anzuregen, um bereits vorhandene Ansatzpunkte für das Verarbeiten der neuen Reizinformation und Anknüpfungspunkte für das neue Wissen zu finden. Da es von dem Lernenden personalisiert verarbeitet wird, kann das neue Wissen besser in die kognitiven Strukturen integriert werden. Die Folge ist ein einfacheres Aufnehmen und Abrufen der neuen Informationen.

Praktisch umgesetzt kann der Lehrprozess *Aktivieren* in Form von Denkphasen erfolgen, wie z.B. Gruppenarbeiten, Brainstormings oder Mind Mappings. [25]

Da in der Regel das aktivierte Vorwissen noch nicht dafür ausreicht, um das mentale Ungleichgewicht wieder auszugleichen, dient dieser Lehrschritt zum *Informieren* der Lernenden über den neuen Lernstoff.

Das heißt in diesem Schritt wird das neue Wissen erarbeitet oder präsentiert. Die Art und Weise der Präsentation des Wissens ist dafür verantwortlich, wie schnell der Lernprozess erfolgen kann. Mögliche Methodiken hierfür sind beispielsweise die direkte Instruktion, bei dem der Lehrende die Informationen strukturiert präsentiert. Dabei ist es wichtig, dass die Inhalte hierarchisch gegliedert sind. Solch ein Aufbau hat sich für Lernende als sehr zu-träglich erwiesen, was damit begründet werden kann, dass das Wissen ebenfalls hierarchisch aufgebaut und vernetzt ist.

Neben einer Präsentation ist es ebenfalls möglich, den Lernenden geeignetes Material in Form von Texten oder Links im Internet zur Verfügung zu stellen. [25]

Während des Lehrschrittes *Informieren* sind die Lernenden eher passiv. Für das Lernen ist jedoch eine konstruktive Restrukturierung notwendig, wie in Abschnitt 6.1 beschrieben. Aufgrund dessen folgt auf das *Informieren* der Lehrschritt *Unterstützen*.

Hier hat der Lehrende die Aufgabe, die Lernenden bei der Integration der neuen Informationen in ihre bestehende kognitive Struktur zu unterstützen. Dies kann er erzielen, indem

er ihnen Hilfestellungen gibt und darauf achtet, dass das mentale Modell wissenschaftlich korrekt ist, das heißt das *richtige* Wissen aufgebaut wird.

Praktisch umgesetzt kann dieser Schritt durch sämtliche Aufgaben, wie bspw. das Erstellen von Zusammenfassungen oder das praktische Arbeiten an einem bestimmten Problem, durchgeführt werden, sofern möglichst alle Lernenden dabei selbst aktiv gefordert sind. [25]

Der letzte Schritt, die Festigung, dient dazu, längerfristig die kognitiven Strukturen zu restrukturieren. Eine einmalige Konstruktion des mentalen Modells reicht hierfür nicht aus, vielmehr muss das Modell von den Lernenden schematisiert werden, um das neu erlernte Wissen längerfristig verfügbar zu machen. Konkret erfolgt dies, indem der Lehrende in Form von Wiederholungen und Übungen den Lernenden die Möglichkeit gibt, das Wissen zu festigen. [25]

6.3 Der Lernbaukasten

Der geplante Lernbaukasten orientiert sich an dem Ansatz eines bereits erfolgreich erprobten Schülerlabors im Rahmen des NaWiTex-Projekts¹. (R004)

Dieses Schülerlabor aus dem Bereich *EcoLab* mit dem Titel „*Aspirin & Co - Medikamenten-Rückstände im Wasser*“ [53] wurde von Frau Dr. Anke Renger, Herrn Andreas Kapp und Herrn Prof. Dr. Fred Lisdat entworfen und ausgearbeitet. Das Labor setzt sich aus den drei Lerneinheiten *Vorlesung*, *Experimentieren* und *Auswertung* zusammen.

Die erste Lerneinheit wird durch einen Wissenschaftler des *EcoLabs* in Form einer Vorlesung direkt in der betreffenden Schule abgehalten. Als Vorbereitung auf die zweite Lerneinheit, behandelt diese Vorlesung alle relevanten Inhalte, die für das spätere Experimentieren in der zweiten Lerneinheit nötig sind.

Neben der Vorlesung erhalten die Schüler in Form eines Arbeitsblattes die Aufgabe, während oder nach der Vorlesung, Fragestellungen zu den vorgestellten Thematiken zu beantworten.

In der zweiten Einheit (Experimentieren) besuchen die Schüler die Technische Hochschule in Wildau. Hier wird den Schülern ein fiktives Szenario präsentiert, in dem sie in die Rolle des Wissenschaftlers schlüpfen. Dabei haben sie die Aufgabe im Analytik-Labor verschiedene Wasserproben auf Unreinheiten hin zu untersuchen und verschiedene Reinigungsverfahren anzuwenden und auszuprobieren. Neben dem Experimentieren haben die Schüler die Aufgabe, alle Versuche und Ergebnisse entsprechend zu dokumentieren.

Die dritte und letzte Lerneinheit findet als Auswertung im Unterricht in der Schule statt. Hier sollen die aufgenommenen Daten aus den durchgeführten Experimenten der Wasseranalytik analysiert und ausgewertet werden. Zusätzlich dient ein zweites Arbeitsblatt dazu, die durchgeführten Verfahren zu wiederholen sowie die Ergebnisse der Experimente zu reflektieren.

Ähnlich wie das vorgestellte Schülerlabor, wird ebenfalls der Lernbaukasten für das Thema KI gestaltet sein. Der Lernbaukasten setzt sich aus den insgesamt drei Lernbausteinen *Theorie*, *Praxis* und *Festigung* zusammen. (R001) Diese drei Lernbausteine verinnerlichen

¹ NaWiTex-Projekt: Das naturwissenschaftlich technische Projekt für Schülerlabore setzt sich aus den vier Bereichen *RoboticLab*, *EcoLab*, *Biologie trifft Technik* und *PhyTecLab* zusammen.

den Ansatz der Lehrstrategie MOMBI, die in Abschnitt 6.2 vorgestellt wurde.

Der erste Lernbaustein beschäftigt sich mit der Theorie der KI und hat die Aufgabe, die ersten drei Lehrschritte nach der Lehrstrategie MOMBI durch einen Mitarbeiter des Schülerlabors *Robotic* umzusetzen. Darunter fallen das Provozieren, das Aktivieren und das Informieren. (R004)

Im zweiten Lernbaustein, der Praxisphase, sollen die Schüler aufbauend auf dem ersten Lernbaustein nun selbst aktiv werden. Indem sie ein bestimmtes Problem der KI praktisch bearbeiten, soll die Integration der neuen Informationen in ihre kognitive Struktur unterstützt werden.

Im letzten Baustein wird den Schülern die Möglichkeit gegeben, ihr neu erlerntes Wissen zu festigen. An dieser Stelle erfolgt die Schematisierung des mentalen Modells nach MOMBI, um das neu erworbene Wissen längerfristig verfügbar zu machen.

In Abbildung 6.3 sind zusammenfassend noch einmal die Lernprozesse und die jeweiligen Lehrschritte nach MOMBI dargestellt. Darunter sind die drei aufeinander aufbauenden Lernbausteine der Länge nach so angeordnet, dass ersichtlich wird, welcher Baustein die jeweiligen Lehrschritte und Lernprozesse abdeckt. (R006)

Abbildung 6.3: Der Lernbaukasten setzt sich insgesamt aus den drei Lernbausteinen *Theorie*, *Praxis* und *Festigung* zusammen. Diese drei Bausteine orientieren sich an den fünf Lehrschritten nach MOMBI.

6.3.1 Lernbaustein 1: Theorie

Der Lernbaustein wird in Form einer Vorlesung vor Ort in der Schule durch einen Mitarbeiter des Schülerlabors *Robotic* durchgeführt.

Dieser Lernbaustein hat die Aufgabe, die Schüler auf den praktischen Abschnitt in der Technischen Hochschule Wildau vorzubereiten und die dazu benötigten Informationen den Schülern zur Verfügung zu stellen.

Insgesamt besteht der Baustein *Theorie* aus fünf Elementen. Das erste Element enthält eine *Fragestellung* bzw. eine *Diskussion*. Im Anschluss darauf erfolgt in Form eines *Brainstormings* die Aktivierung von vorhandenem Vorwissen bei den Schülern und die letzten beiden Elemente beinhalten das Informieren der Schüler über die *Grundlagen der KI* und die *Vorbereitung auf die Praxisphase* in Lernbaustein 2. Das fünfte Element enthält ein *Arbeitsblatt* mit Fragestellungen zu der Thematik. In Abbildung 6.4 sind die fünf Elemente aufgeführt und nachfolgend wird auf diese noch einmal näher eingegangen. (R002)

Theorie

Abbildung 6.4: Der erste Lernbaustein *Theorie* hat die Intention, den Schülern die nötigen Informationen für den praktischen Einsatz in der Hochschule zur Verfügung zu stellen.

Jeder Schüler erhält zu Beginn der Vorlesung ein Arbeitsblatt mit Aufgabenstellungen. Dieses Arbeitsblatt dient zur Vorbereitung auf den praktischen Abschnitt im zweiten Lernbaustein. Den Schülern wird die Möglichkeit gegeben, die Fragestellungen während der Vorlesung zu bearbeiten. Im ausgefüllten Zustand dient das Arbeitsblatt für die Schüler als Abrundung und Zusammenfassung des Lernbausteins *Theorie* und als Vorbereitung auf den zweiten Lernbaustein.

Die Vorlesung wird mit einer Diskussion gestartet. Diese Diskussion zielt auf das Thema KI und dessen Auswirkungen auf das Individuum und die Gesellschaft ab. Dabei werden die Schüler durch Fragestellungen oder Thesen dazu aufgefordert, mögliche Auswirkungen der KI auf unsere Gesellschaft zu reflektieren und zu diskutieren.

Dieser Einstieg stellt den ersten Lehrschrift nach MOMBI dar und soll ein mentales Ungleichgewicht bei den Schülern provozieren, sodass diese zur mentalen Modellbildung angeregt werden. Zusätzlich schafft die Diskussion einen guten Einstieg in die Thematik der KI und hilft dabei, Schnittstellen zu dem Lehrplan des Informatikunterrichtes zu schaffen. Laut Rahmenplan sieht dieser für Berlin und Brandenburg in Abschnitt 4.5 das Thema *Informatik, Mensch und Gesellschaft* vor. (R005) [56, S. 23] [46, S. 21]

Im nächsten Schritt wird in Form eines Brainstormings das aktuelle Wissen der Schüler abgerufen bzw. nach MOMBI aktiviert. Hierfür sollen die Schüler zusammentragen, was sie über das Thema KI bereits wissen und dieses in die Kategorien Anwendungsbereiche, Funktionsprinzipien, Wissenschaft, Gefühle und Science Fiction einordnen.

Das Ziel dieses Abschnittes besteht darin, anhand der gesammelten Informationen eine zufriedenstellende Definition der KI gemeinsam erarbeiten zu können.

Im Anschluss daran erfolgt eine *erste* gemeinsame Wissenserarbeitung zu den Grundlagen und Arbeitsweisen der verwendeten KI-Werkzeuge in der Anwendung. Dabei handelt es sich lediglich um eine Einführung in die Thematik, um den zeitlichen Rahmen eines Vortrags einhalten zu können. Die konkrete Erarbeitung der Thematik soll vielmehr durch den unterrichtenden Lehrer erfolgen, bspw. in Form eines Seminarkurses, der in Abstimmung mit dem RoboticLab ausgearbeitet wird, oder Ähnliches.

Da hier ein gewisser Abstimmungsaufwand zwischen dem Personal des Schülerlabors und dem Lehrer nötig ist, ist es sinnvoll dieses Labor lediglich für Schulen anzubieten, mit denen eine Kooperation besteht und mit denen ohnehin eine enge Zusammenarbeit bereits erfolgt.

Darauf aufbauend wird im Vortrag das Szenario und die Anwendung als Vorbereitung auf die Praxis vorgestellt und erläutert.

Als zeitlicher Rahmen für den Vortrag wird ein normaler Unterrichtsblock mit einer Zeiteinheit von 90 Minuten angestrebt. Für Schüler der SEK 2 ist dieser zeitliche Rahmen aus dem schulischen Alltag bekannt und zumutbar. (R008)

6.3.2 Lernbaustein 2: Praxis

*„Ich höre und vergesse.
Ich sehe und erinnere.
Ich tue und verstehe.“*

(Konfuzius)

Der zweite Lernbaustein stellt den praktischen Abschnitt des Lernbaukastens dar. Hier haben die Schüler nach einer kurzen Einführungsphase die Möglichkeit, das erlernte Wissen aus dem ersten Lernbaustein praktisch anzuwenden, zu testen und zu festigen. (R007)

Die Intention dieser Praxisphase verfolgt die Idee, dass die Schüler mit Hilfe der IVE, einem bereitgestellten Prototyp sowie Sensorik ein bestimmtes Szenario abbilden. Hierfür dient das vorgestellte Szenario aus Abschnitt 4.1.1, bei dem das Alltagsverhalten der älteren Dame Karin in ihren eigenen vier Wänden überwacht werden soll.

Den Schülern wird dafür ein Prototyp zur Verfügung gestellt, der einen Teil des vorgestellten Szenarios repräsentiert und mit dem das Experimentieren und Testen mit KNN über die IVE möglich wird.

Der Prototyp stellt einen speziellen Raum von Karins Wohnung dar, dabei handelt es sich um das Schlafzimmer. Neben dem Prototyp selbst, gibt es zusätzlich die benötigte Hardware, wie z.B. die Sensorik.

Die Schüler sollen mit Hilfe des Prototyps und der Anwendung *Smarter Home* den Schlafrythmus der älteren Dame überwachen, um bei etwaigen Unregelmäßigkeiten im Schlafverhalten dies über die IVE feststellen zu können.

Hierfür müssen sie zunächst die Sensorik in dem Prototyp geschickt anbringen, sodass die Erzeugung von verwertbaren Daten möglich ist. Hiernach haben sie die Aufgabe, die Kommunikation zwischen Sensorik und IVE herzustellen. Und im letzten Schritt durchlaufen sie mit Hilfe der Anwendung, dem Prototyp und der darin angebrachten Sensorik alle drei Phasen von der Erstellung, dem Training bis hin zum Einsatz des KNN. Dabei müssen sie zunächst eine geeignete Netzarchitektur erstellen (*Discover-Modus*), Trainingsdaten erzeugen und das Netz damit trainieren (*Trainings-Modus*), um anschließend durch den Einsatz des KNN überprüfen (*Go Live-Modus*) zu können, ob das trainierte Netz dazu in der Lage ist, Unregelmäßigkeiten zu erkennen.

Da es hier kein „Kochrezept“ für die geeignetste Architektur des KNN gibt, sollen die Schüler in Form eines iterativen Prozesses verschiedene Netzarchitekturen sowie Trainingsdaten und -konfigurationen austesten und dokumentieren, um am Ende die beste Variante davon vorstellen zu können (siehe Abschnitt 6.3.3). Dafür bekommen die Schüler ein entsprechendes

Protokoll, in dem sie ihre Versuche und Ergebnisse dokumentieren können. Abbildung 6.5 zeigt die einzelnen Phasen des Lernbausteins sowie deren chronologischen Ablauf.

Abbildung 6.5: Der Lernbaustein für die Praxis soll die Planung und Konstruktion, die Programmierung und das Arbeiten mit KNN für die Schüler ermöglichen.

Neben der praktischen Anwendung von Wissen über KNN sollen hierbei auch die programmiertechnischen Fähigkeiten, die Teamarbeit und das Experimentieren sowie Dokumentieren gefordert und gefördert werden.

Hierfür arbeiten die Schüler in Teams mit einer Gruppenstärke von mindestens 2 und maximal 3 Personen zusammen.

Der Prototyp, der das Schlafzimmer von Karin darstellen soll, wird mit Hilfe von Lego-Bausteinen umgesetzt. Die ältere Dame Karin wird in Form einer entsprechenden Legofigur repräsentiert.

Durch die Verwendung von Bausteinen des Spielzeugherstellers *LEGO*¹ besteht einerseits die Möglichkeit, unproblematisch Erweiterungen und Modifizierungen an der Wohnung vornehmen zu können und andererseits wird den Schülern eine vertraute Technik an die Hand gegeben, die sie aller Wahrscheinlichkeit nach bereits kennen.

So sollen Berührungsängste seitens der Schüler genommen werden und die Wohnung zum Anfassen zusätzlich einen haptischen und spielerischen Part darstellen, der eine emotionale Bindung entstehen lässt. In Abbildung 6.6 ist der geplante Grundriss des Prototyps, der das Schlafzimmer und deren Komponenten sowie Karin selbst enthält, dargestellt. Das Simulieren des Schlafes von Karin soll über RFID²-Technik realisiert werden. Hierfür muss ein RFID-Empfänger an das Bett angebracht und der RFID-Sender von Karin am Körper getragen werden. Sobald sie sich in der Nähe des Betts befindet, erhält der RFID-Empfänger ein Signal von Karins Sender. Dies wird in dem Szenario dahingehend interpretiert, dass Karin sich im Bett befindet und schläft bzw. vorhat zu schlafen.

¹ The LEGO Group: <https://www.lego.com/de-de>

² RFID: Radio-Frequency Identification

Abbildung 6.6: Der geplante Prototyp für den praktischen Lernbaustein repräsentiert das Schlafzimmer von Karin. Darin soll mit Hilfe der IVE das Schlafverhalten von Karin trainiert und überwacht werden können.

Für die Erzeugung und Übertragung von Sensordaten wird das *Funduino Lernset Nr. 8 mit Arduino UNO R3 Controller*¹ verwendet. Dabei handelt es sich um ein Set, bei dem neben dem Mikrocontroller von *Arduino* noch diverse Sensoren, Aktoren, Kabelverbindungen und weitere nützliche und vor allem kompatible Zubehörartikel mitgeliefert werden.

Neben den einzelnen Zubehörteilen bietet *Funduino* darüber hinaus über die eigene Webseite ein großes Repertoire an Anleitungen und Tutorials für die jeweilig mitgelieferten Zubehörartikel an.

Der *Arduino UNO R3* (Abbildung 6.7) selbst basiert auf dem Mikrocontroller ATmega328P. Eine ausführliche Übersicht zu den technischen Daten des Mikrocontrollers wird in der Tabelle 6.1 gegeben. [1]

¹ Funduino Lernset Nr. 8:
http://www.funduinoshop.com/epages/78096195.sf/de_DE/?ObjectPath=/Shops/78096195/Products/01-U8

Abbildung 6.7: Der *Arduino UNO* wird für die Erzeugung sowie Übertragung der Sensordaten an die IVE eingesetzt. [1]

Der beschriebene Mikrocontroller wird als fester Bestandteil schon erfolgreich in anderen Schülerlaboren im Rahmen des RoboticLabs von NaWiTex eingesetzt. Die Verwendung dieses Mikrocontrollers für die Realisierung des Prototyps hat den Vorteil, dass für spätere Durchführungen mit Schülerklassen bereits die Hardware als Klassensatz im Bestand vorhanden ist und dafür genutzt werden kann. Zusätzlich sind die Laborrechner bereits mit der Software zur Programmierung der *Arduinos*¹ bespielt, sodass kein zusätzlicher Installationsaufwand entsteht.

Eigenschaft	Ausführung
Mikrocontroller	ATmega328
Betriebsspannung	5 V
Eingangsspannung	7-12 V
Ausgangsspannung	6-20 V
Digitale I/O-Pins	14
Analoge Input Pins	6
DC-Stromstärke pro I/O-Pin	40 mA
DC-Stromstärke für 3.3 V Pin	50 mA
Flash-Speicher	32 KB
SRAM	2 KB
EEPROM	1 KB
Taktfrequenz	16 MHz

Tabelle 6.1: Die technischen Daten des *Arduino UNO* in der Übersicht. [1]

Die Sensorik im Prototyp wird durch das RFID-Kit aus dem Set von *Funduino* realisiert. Hierbei handelt es sich um einen RFID-Empfänger vom Typ RFID-RC522. Zusätzlich enthalten sind zwei Mifare Transponder in Form einer Scheckkarte und eines Schlüsselanhängers. [21] In Abbildung 6.8 ist das Set mit seinen Bestandteilen dargestellt. Für den Einbau in den Prototyp werden lediglich der RFID-Empfänger und der Transponder in Form des Schlüsselanhängers verwendet.

¹ Arduino IDE: <https://www.arduino.cc/en/main/software>

Abbildung 6.8: Das RFID-Kit von *Funduino* besteht aus einem RFID-Empfänger und zwei RFID-Sendern in Form einer Karte im Scheckkartenformat und eines Schlüsselanhängers. [21]

In Tabelle 6.2 sind die technischen Daten des RFID-Kits zusammengefasst.

Eigenschaft	Ausführung
Chip	MFRC522
Protokoll	Serial Peripheral Interface
Betriebsspannung	3.3Volt
Frequenz	13,56 MHz
Stromaufnahme	13-26mA
Stromaufnahme Standby	10-13mA
Betriebstemperatur	-20 bis 80°C

Tabelle 6.2: Die technischen Daten des RFID-Kits von *Funduino* in der Übersicht. [21]

Für die Kommunikation zwischen dem Mikrocontroller von *Arduino* und der IVE werden die XBee-Module *XBee PRO S1* von der Firma *Digi International*¹ verwendet.

Diese Module ermöglichen einen Datenaustausch über den Funkstandard ZigBee. Für den *Arduino UNO* wird zusätzlich ein sogenanntes XBee Shield von der Firma *SainSmart*² verwendet, das eine einfache Montage am *Arduino* sowie die Datenübertragung über das XBee-Modul ermöglicht. Abbildung 6.9 zeigt das XBee Shield mit angebrachtem XBee-Modul zum Aufstecken auf den *Arduino UNO*.

¹ Digi International XBee: <https://www.digi.com/lp/xbee>

² SainSmart XBee Schild Modul für *Arduino UNO*:
<https://www.sainsmart.com/sainsmart-xbee-shield-module-for-zigbee-arduino-uno-duemilanove-mega-1280-2560.html>

Abbildung 6.9: Die Kommunikation zwischen *Arduino* und IVE soll über sogenannte XBee-Module erfolgen, die eine Kommunikation via ZigBee ermöglichen. Für den *Arduino UNO* wird ein entsprechendes XBee Shield verwendet. [14]

Der zeitliche Rahmen für diesen Lernbaustein wird an einen normalen Schultag angelegt mit einer Dauer von 7 Zeitstunden inklusive der Pausen. Die Praxisphase besteht aus mehreren Abschnitten, in denen die Schüler experimentieren und kreativ sein sollen, aufgrund dessen kann der zeitliche Rahmen begründet werden. (R008)

6.3.3 Lernbaustein 3: Festigung

Der letzte Lernbaustein wird in der Schule von der Lehrkraft selbst durchgeführt und dient zur Festigung des Erlernten aus den vorangegangenen Lernbausteinen.

Hierfür werden der Lehrkraft Anregungen und Material für Vorträge und Präsentationen mitgegeben, die die Schüler ausarbeiten können. Darin können sie einerseits ihre Ergebnisse, Erkenntnisse und Erfolge aus der Arbeit in der Praxisphase auswerten und präsentieren und zusätzlich weitere interessante Forschungsprojekte im Bereich der KI vorstellen und dabei den Nutzen und die eventuellen Gefahren gesellschaftskritisch beleuchten und diskutieren. (R003)

Der Lehrkraft wird somit die Möglichkeit gegeben, sich ein Bild von der Qualität der Arbeit der Schüler aus der Praxisphase zu machen.

Außerdem wird durch die Beleuchtung der Thematik KI unter einem gesellschaftskritischen Aspekt eine Schnittstelle zu den Lehrplänen an Berliner und Brandenburger Schulen der SEK 2 geschaffen. Dieser sieht bei den inhaltlichen Schwerpunkten für die Sekundarstufe 2 in Abschnitt 4.5 das Themenfeld *Informatik, Mensch und Gesellschaft* vor. In diesem Abschnitt sollen die Schüler sich mit den Auswirkungen der sich immer schneller entwickelnden und ausbreitenden Informatiksysteme auf das Individuum und die Gesellschaft beschäftigen. [57, S. 23, Abschn. 4.5] [44, S. 21, Abschn. 4.5]

Somit hat die Lehrkraft die Möglichkeit, am Ende dieses Lernbausteines eine Benotung vorzunehmen, die im Einklang mit den vorgesehen Rahmenlehrplänen von Berlin und Brandenburg steht.

6.4 Geplante Entwicklungsbausteine

Die geplanten Entwicklungsbausteine sollen Ideen, Ansätze und Anreize zur Weiterentwicklung der entstandenen Ergebnisse und Produkte in dieser Masterarbeit schaffen. Die Entwicklungssteine stellen keine konkret ausgearbeiteten Aufgabenstellungen dar, sondern lassen einen gewissen Spielraum für die Konkretisierung der jeweiligen Aufgaben zu.

Dabei orientieren sich an Schüler adressierte Entwicklungsbausteine an deren schulischen Alltag und versuchen damit in Einklang gebracht zu werden.

Bei den Entwicklungsbausteinen für Studierende werden hingegen die Studienziele des Bachelorstudiengangs Telematik berücksichtigt. Dazu zählen die Beherrschung methodischer, technischer und fächerübergreifender Grundlagen, die Fähigkeit zur Planung und Entwicklung von Telematikprodukten und -systemen sowie die Fähigkeit sich weiterzubilden, gerade in Bezug auf zukünftige technologische Entwicklungen.

6.4.1 Entwicklungsbaustein 5. *Prüfungs-komponente*

Dieser Entwicklungsbaustein bezieht sich auf das Anfertigen einer *besonderen Lernleistung*. Diese Leistung kann im Rahmen der 5. Prüfungskomponente von Schülern der SEK 2 an Berliner Gymnasien oder Gesamtschulen mit gymnasialer Oberstufe abgelegt werden.

Im Land Berlin wurde im Jahr 2006 die sogenannte 5. Prüfungskomponente im Abitur eingeführt und hat sich seitdem als innovatives und zukunftsweisendes Prüfungsformat an den Berliner Schulen etabliert. [58]

So erzielen bspw. die Schüler in diesem Prüfungsteil vergleichsweise bessere Noten als in den anderen Prüfungsfächern. Zusätzlich haben sie in der 5. Prüfungskomponenten die Möglichkeit, ihre Fähigkeiten und Fertigkeiten ganzheitlich einzubringen und Arbeitstechniken und Arbeitsweisen zu trainieren, die sie für die spätere Arbeitswelt oder das Studium vorbereiten. Hierfür wählen die Schüler ihr Thema und die Darstellungsform selbstständig aus und erhalten seitens der Schule eine entsprechende Vorbereitung auf die für die Prüfungsform erforderlichen Fähigkeiten, wie bspw. das Kommunizieren mit anderen oder die strukturierte und anschauliche Darstellung von Sachverhalten.

Für die Wahl und Genehmigung des Themas steht jedem Schüler mindestens eine betreuende Lehrkraft der Schule zur Verfügung. [58]

Grundsätzlich gliedert sich die 5. Prüfungskomponente in zwei Prüfungsformen auf: die besondere Lernleistung und die Präsentationsprüfung. [58]

Die besondere Lernleistung eignet sich vor allem für Schüler, deren Stärken im schriftlichen Prüfungsbereich liegen.

In einem längerfristigen Prozess schreiben die Schüler hierfür eine Arbeit, die sie auf ein mögliches Studium und das wissenschaftliche Arbeiten heranzuführen und vorbereiten soll. Hierfür ist ein großes Maß an Selbstständigkeit, Kontinuität und Zielgerichtetheit aufseiten der Schüler erforderlich. [58]

Die Präsentationsprüfung eignet sich wiederum vor allem für Schüler, deren Stärken im mündlichen Prüfungsbereich anzusiedeln sind. Die Prüfung setzt sich aus einer schriftlichen Ausarbeitung, einer Präsentation sowie einem anschließenden Prüfungsgespräch zusammen. Sie kann in Form einer Einzel- oder auch Gruppenprüfung erfolgen, mit einer Gruppenstärke von bis zu vier Prüflingen. Die Präsentation erfolgt mediengestützt und hat als Vorgabe, dass auch fachübergreifende Aspekte der Thematik behandelt werden. [58]

Im Rahmen der Schülerlabore ist es nicht ungewöhnlich, dass Schüler nach dem Durchlaufen von praktischen Schülerlaboren an der Hochschule auch ihre 5. Prüfungskomponente unter der Betreuung von wissenschaftlichen Mitarbeitern des Schülerlabors schreiben. Im Jahr 2016 waren dies allein 5 Schüler in dem Bereich RoboticLab.

Aufgrund dessen visiert der Entwicklungsbaustein in diesem Abschnitt die Ausarbeitung einer solchen Arbeit im Rahmen der 5. Prüfungskomponente an. (R050)

Dabei erhalten die Schüler die Möglichkeit, ihre 5. Prüfungskomponente im Bereich KI abzulegen. Hierfür bekommen die Schüler den Prototyp und die Anwendung *Smarter Home*, die im Rahmen dieser Masterarbeit entstanden sind, zur Verfügung gestellt. (R048)

Die Idee besteht darin, neue Tätigkeiten im Szenario der Wohnung der älteren Dame Karin zu entwickeln. Hierfür sollen zusätzliche sinnvolle Sensoren ausgewählt und verwendet werden, um dadurch eine oder mehrere neue Tätigkeiten anzulegen und zu testen, um anschließend deren Eignung bewerten zu können. (R049)

Die dafür benötigte Hardware in Form von Sensorik wird seitens des Schülerlabors für den Schüler zur Verfügung gestellt.

Eine Zuordnung dieses Entwicklungsbausteins, hinsichtlich der beiden Prüfungsformen, kann und soll in diesem Abschnitt nicht getroffen werden. Optimaler Weise wird die Zuordnung zusammen mit der betreuenden Lehrkraft entschieden, wodurch auch leichte Anpassungen in der Aufgabenstellung möglich werden.

Die Voraussetzung zur Durchführung dieses Entwicklungsbausteins besteht darin, dass der Schüler den in Abschnitt 6.3 konzipierten Lernbaukasten in Form eines Schülerlabors erfolgreich absolviert hat. Dadurch soll sichergestellt werden, dass für die Ausarbeitung kein zu großer Einarbeitungsaufwand in die Thematik und die Praxisanwendung für den Schüler entsteht und der zeitliche Rahmen eingehalten werden kann. (R051)

6.4.2 Entwicklungsbaustein *Embedded C++*

Dieser Entwicklungsbaustein ist als Projektaufgabe für das Wahlpflichtfach Embedded C++ im 6. Semester des Bachelorstudienganges Telematik angedacht. (R050)

In diesem Wahlpflichtfach haben die Studierenden, nach einer kurzen Einführung in die Programmiersprache C++, die Aufgabe, ein Projekt mit dem humanoiden Roboter *NAO* von der Firma *Aldebaran* zu entwerfen und praktisch umzusetzen. Die Ausarbeitung erfolgt in Form einer Gruppenarbeit mit 2 - 3 Studierenden.

Die Idee dieses Entwicklungsbausteins verfolgt die Intention, den Prototyp aus dem Lernbaukasten aus Abschnitt 6.3 durch Studierende im Rahmen dieser Projektaufgabe weiterzu-

entwickeln. (R048)

Dabei soll das Szenario aus dem Lernbaukasten um die Komponente der *Servicerobotik* ergänzt werden.

Die Studierenden haben hierfür zunächst die Aufgabe, die Verbindung zwischen IVE und Roboter über die REST-Schnittstelle zu realisieren.

Anschließend sollen, anhand der erhaltenen Informationen über den Webservice, Kriterien getroffen und begründet werden, wann der *NAO* in das Geschehen eingreifen soll.

Hierfür muss zusätzlich eine geeignete Form der Interaktion mit der betroffenen Person evaluiert und umgesetzt werden.

Im letzten Schritt wäre das Entwerfen und gegebenenfalls auch die Umsetzung einer Eskalationskette denkbar. Diese soll aufeinanderfolgende Eskalationsstufen enthalten, die bei Feststellung einer Notsituation von dem *NAO* durchlaufen werden und in der Alarmierung einer außenstehenden Person oder Einheit enden. (R049)

Für die Umsetzung des Entwicklungsbausteins wird eine *Smart Home*-Komponente benötigt, dessen konkrete Umsetzung den zeitlichen Rahmen des Bausteins sprengen würde. In diesem Fall ermöglicht die Anwendung *Smarter Home* jedoch auch die Simulation von Sensoren und macht damit die Umsetzung dieser Komponente nicht erforderlich. (R051)

6.4.3 Entwicklungsbaustein *Abschlussarbeit*

Im Verlauf des 6. Semesters im Bachelorstudiengang Telematik müssen die Studierenden neben den Lehrveranstaltungen eine Bachelorarbeit anfertigen, um erfolgreich den Abschluss *Bachelor of Engineering* erlangen zu können.

Hierfür haben die Studierenden die Möglichkeit, die Bachelorarbeit innerhalb eines Unternehmens anzufertigen oder direkt an der Hochschule. Dabei stellen gerade die Vorarbeiten für große Forschungsprojekte oder die Weiterarbeit an bereits bestehenden Projekten wichtige und interessante Themen für eine Bachelorarbeit dar. (R048)

Der Entwicklungsbaustein in diesem Abschnitt stellt solch ein mögliches Thema für die Anfertigung einer Abschlussarbeit im Bachelorstudiengang Telematik dar. (R050)

Dafür soll der *NAO*-Roboter von der Firma *Aldebaran* verwendet werden. Dieser wird für den Zeitraum der Abschlussarbeit dem Studierenden zur Verfügung gestellt.

Die Aufgabe besteht darin, die zwei Bereiche *Smart Home* und *Servicerobotik* durch den Einsatz der IVE zu verbinden. Dabei soll ein sinnvolles Szenario evaluiert, getestet und prototypisch umgesetzt werden, mit dem es möglich ist eine alltägliche Tätigkeit im Haushalt überwachen zu können. (R049)

Für diese Aufgabe muss der Student verschiedene Fähigkeiten unter Beweis stellen. Einerseits muss er sich in das bestehende Framework (IVE) einarbeiten, dieses verstehen und anwenden können und andererseits muss er ein sinnvolles Szenario aus dem Bereich *Smart Home* evaluieren und mit Hilfe von Sensoren umsetzen und mit der IVE verbinden. Zusätzlich muss der *NAO*-Roboter über die bereitgestellte Schnittstelle der IVE in der Lage sein, die nötigen Informationen abzurufen, um anhand daran entsprechend agieren und reagieren zu können.

Für die Umsetzung dieser Bachelorarbeit wäre es sinnvoll, wenn der Student die Lehrveran-

staltung Embedded C++ besucht hat und somit schon Vorkenntnisse in der Arbeit mit dem *NAO*-Roboter besitzt. (R051)

Neben dem Anwendungsbereich *Smart Home* wäre ebenfalls ein Szenario aus dem Bereich *Smart Library* denkbar, in dem ein Szenario mit den vorgestellten Komponenten entworfen wird, das zur Umsetzung einer 24/7-Bibliothek beiträgt.

Kapitel 7

Technische Umsetzung *Smarter Home*

Im vorangegangenen Kapitel 5 wurden die verschiedenen Lösungskonzepte zur Entwicklung der Intelligenten Verarbeitungseinheit *Smarter Home* vorgestellt.

In diesem nun folgenden Kapitel wird die Umsetzung der einzelnen Komponenten der Anwendung *Smarter Home* beschrieben.

Hierbei wird zunächst auf die grundlegende Anwendungsarchitektur eingegangen und ein erster Überblick zu den Komponenten gegeben, bevor nachfolgend auf die wichtigsten Komponenten noch einmal im Detail eingegangen wird. Abschließend für dieses Kapitel wird im letzten Abschnitt eine Gesamtübersicht gegeben, die das grundlegende Zusammenspiel der Komponenten in der Anwendung noch einmal verdeutlicht und zusammenfasst.

7.1 Grundlegende Anwendungsarchitektur

Die grundlegende Architektur der Anwendung ist nach dem sogenannten *Drei-Schichten-Modell* ausgerichtet.

Dabei handelt es sich um eine Software-Architektur, die sich aus den drei Schichten *Präsentation*, *Fachkonzept* und *Datenhaltung* zusammensetzt. Die Präsentationsschicht übernimmt dabei die Funktionalität zur Darstellung der Benutzeroberfläche. Die Schicht des Fachkonzepts ist für die eigentliche Umsetzung der Funktionen bzw. der fachlichen Anforderungen an die Anwendung zuständig. Die letzte Schicht, die Datenhaltungsschicht, dient dem Speichern und Laden von Daten innerhalb der Applikation.

In Abbildung 7.1 ist der grundsätzliche Aufbau der Architektur dargestellt, die Pfeile symbolisieren dabei die jeweilig möglichen Zugriffsrichtungen zwischen den Schichten.

Dabei gibt es zwei Ansätze, der erste erlaubt nur die strikte Kommunikation zwischen zwei benachbarten Schichten, gekennzeichnet in der Abbildung durch die durchgezogene Pfeillinie. Der zweite Ansatz lässt auch das Überspringen von Schichten in der Kommunikation zu (gestrichelte Pfeillinien). [36]

In der Anwendung *Smarter Home* wurde bei der Kommunikation der erste Ansatz, die strikte Drei-Schichten-Architektur, realisiert.

Durch die Abbildung wird deutlich, dass weder die Fachkonzept- noch die Datenhaltungsschicht eine Zugriffsmöglichkeit auf die Präsentationsschicht besitzen. Somit hat keine der beiden Schichten Kenntnis darüber, wie die Benutzeroberfläche explizit aufgebaut ist.

Abbildung 7.1: Die Drei-Schichten-Architektur teilt sich in die Schichten GUI, Fachkonzept und Datenhaltung auf. [36]

Dies ermöglicht eine unabhängige Entwicklung, Modifizierung und Wartung der Schichten. Die konsequente Durchsetzung einer solchen Architektur verfügt darüber hinaus über den Vorteil der Skalierbarkeit hinsichtlich der Anwendung. [36]

Die Kommunikation zwischen den Schichten wird in der Anwendung mit Hilfe des *Observer*-Patterns realisiert. Dabei informiert die Fachkonzeptsschicht indirekt die Komponenten der Präsentationsschicht, sobald Änderungen aufgetreten sind. Die Präsentationsschicht ruft daraufhin alle relevanten und aktualisierten Daten aus der Fachkonzeptsschicht ab.

Für das Abrufen der jeweiligen Daten stehen der Präsentationsschicht in der Fachkonzeptsschicht verschiedene *Services* zur Verfügung, die die jeweilig benötigten Dienste anbieten. Auf die konkrete Umsetzung der jeweiligen Komponenten und deren *Services* wird in den nachfolgenden Abschnitten eingegangen.

7.2 Grafische Benutzeroberfläche

Die grafische Benutzeroberfläche der Anwendung *Smarter Home* dient zur Interaktion mit dem Anwender. Sie ermöglicht diesem über die Bereiche *Discover*, *Training* und *Go Live* alle nötigen Schritte, um ein KNN zu erstellen, zu trainieren und einzusetzen.

In den nachfolgenden Abschnitten werden zunächst ein paar Bedienelemente beschrieben und anschließend die wichtigsten Komponenten erklärt, die für die Umsetzung der Benutzeroberfläche maßgeblich sind.

7.2.1 Bedienelemente

In diesem Abschnitt werden die wichtigsten Bedienelemente der grafischen Benutzeroberfläche von der Anwendung *Smarter Home* vorgestellt.

Verbinden mit dem XBee-Empfänger

Die Herstellung einer Verbindung mit dem XBee-Empfänger ist für das Empfangen von Sensordaten essentiell. Hierfür muss zunächst in der Anwendung der entsprechende USB¹-Port auf dem Rechner ausgewählt werden, auf dem die Anwendung läuft. Dafür bietet die Anwendung in dem *Discover*-Modus über den Button *Connect* die Möglichkeit diesen auszuwählen. Nach Betätigung des Buttons kann der gewünschte USB-Port ausgewählt werden,

1 USB: Universal Serial Bus

wie dargestellt in Abbildung 7.2. Nachdem die Anwendung die Verbindung mit dem Empfänger hergestellt hat, können Sensordaten von einem Sensor empfangen werden. Wichtig hierfür ist, dass die Anwendung sich im Live Data-Modus befindet.

Abbildung 7.2: Über den Button *Connect* kann die Verbindung zu dem XBee-Empfänger hergestellt werden.

Verwaltung der Eingangsneuronen

Die aktuell vorhandenen Eingangsneuronen des KNN sind in der Anwendung in der Tabelle *Input neurons* mit den Attributen *ID*, *Name*, *Type* und *Value* zeilenweise hinterlegt, wie in Abbildung 7.3 dargestellt.

Connect

Live Data

Override

Input neurons:

ID	Name	Type	Value
1	Kühlschrank	BINARY	1.0
2	Wohnzimmer	TEMPER...	22.5
3	RFID Bett	BINARY	0.0

Add

Remove

Abbildung 7.3: Die einzelnen Eingangsneuronen sind in der Tabelle der Anwendung zeilenweise hinterlegt.

Über den *Add*-Button unter den *Input neurons* können manuell neue Eingangsneuronen hinzugefügt werden. Dabei werden die Attribute für das Eingangsneuron über einen zusätzlichen Dialog eingegeben. Das doppelte Vergeben einer ID für ein neues Eingangsneuron wird durch die Anwendung abgefangen (Abbildung 7.4).

Über den *Remove*-Button kann ein Eingangsneuron entfernt werden. Hierfür muss das zu entfernende Neuron in der Tabelle vorher durch Anklicken ausgewählt werden.

Befindet sich die Anwendung im Override-Modus ist die Tabelle der Eingangsneuronen editierbar. Dadurch ist es möglich, einzelne Werte der Eingangsneuronen, außer der ID, durch

Anklicken zu verändern. Im Live Data-Modus ist das Editieren deaktiviert und die aktuellen Sensorwerte der Sensoren werden angezeigt, simulierte Eingangsneuronen bleiben unverändert.

Abbildung 7.4: Beim Anlegen eines neuen Eingangsneurons vergibt der Benutzer die ID, den gewünschten Namen und den Sensortyp.

Verwaltung der Ausgangsneuronen

Die Verwaltung und Darstellung der Ausgangsneuronen erfolgt in der Benutzeroberfläche nahezu äquivalent zu den Eingangsneuronen. Die bereits angelegten Ausgangsneuronen werden ebenfalls zeilenweise mit deren Attributen (ID, Name, Value) in einer eigens dafür vorhandenen Tabelle mit dem Namen *Output neurons* angezeigt (Abbildung 7.5). Dabei können ebenfalls Neuronen über die entsprechenden Buttons hinzugefügt und entfernt werden. Der Wert für den *Value* ist in der Tabelle als Button hinterlegt. Über das Betätigen des Buttons kann sein Wert (true und false) geändert werden, zusätzlich werden die Werte true und false durch eine grüne bzw. rote Farbe des Buttons visualisiert. Damit wird das Anlegen von positiven und negativen Trainingsdatensätzen möglich.

Output neurons:

ID	Name	Value
1	Schlafen	true
2	Kochen	false

Add Remove

Abbildung 7.5: Die angelegten Ausgangsneuronen sind in der Tabelle *Output neurons* einzusehen. Auch hier besteht die Möglichkeit, neue Ausgangsneuronen hinzuzufügen oder bestehende zu löschen.

Konfiguration des Netzwerkes

Für die Konfiguration des Netzwerkes stehen dem Benutzer verschiedene Konfigurationsmöglichkeiten zur Verfügung. Über den Button *Add Layer* werden dem KNN verdeckte Schichten hinzugefügt und über den Button *Remove Layer* entsprechend entfernt.

Jede verdeckte Schicht ist mit ihren Eigenschaften tabellarisch in dem Konfigurationsdialog dargestellt, siehe Abbildung 7.6. Eine Zeile repräsentiert dabei eine Schicht im KNN und die Spalten die jeweils definierten Einstellungen.

Dabei kann der Benutzer für jede Schicht die Anzahl der verwendeten Neuronen festlegen, die Aktivierungsfunktion konfigurieren und definieren, ob ein zusätzlicher Schwellwert in Form eines Bias-Neurons für die Schicht verwendet werden soll.

Configure network...

Neurons	Activation Fun...	Bias
8	SIGMOID	true
8	SIGMOID	true
8	SIGMOID	false

Add Layer Remove Layer

Cancel OK

Abbildung 7.6: Über den Button *Configure Network* kann der Benutzer in einem Dialog die verdeckten Schichten des KNN konfigurieren.

Die Aktivierungsfunktion ist in einem KNN dafür zuständig, anhand der Netzeingabe und unter der Berücksichtigung eines Schwellwertes, den neuen Zustand eines Neurons zu bestimmen.

Zu den bekanntesten Aktivierungsfunktionen gehören die binäre Schwellenwertfunktion, die lineare Funktion und die Sigmoidfunktion. In Abbildung 7.7 ist deren jeweiliges grundsätzliches Verhalten dargestellt.

Abbildung 7.7: Eine Übersicht zu den grundlegenden Aktivierungsfunktionen für ein KNN.

Bei dem Bias-Neuron handelt es sich um einen Schwellwert, der einer Schicht hinzugefügt werden kann. Dieses Neuron besitzt den konstanten Wert 1, verfügt ausschließlich über ausgehende Verbindungen und dient als zusätzlicher Parameter zur Bestimmung der Aktivierung eines Neurons. In Abbildung 7.8 ist das grundsätzliche Hinzuschalten eines Bias-Neurons zu einer Schicht dargestellt.

Abbildung 7.8: Das sogenannte Bias-Neuron kann zu einer Schicht hinzugefügt werden. Dieses spezielle Neuron stellt einen Schwellwert dar.

Als Lernverfahren wird der *Backpropagation-Algorithmus* verwendet, das Lernverfahren kann in dieser Version der Anwendung nicht verändert werden. Der *Backpropagation-Algorithmus* ist ein sehr bekanntes überwachtes Lernverfahren. Die schichtweise Gewichts Anpassung während des Trainings erfolgt anhand der Abweichung von tatsächlichem und gewünschtem Ergebnis (siehe Glossareintrag).

Über die Konfiguration des KNN ist der Benutzer also in der Lage, ein Perzeptron mit einer gewünschten Anzahl an verdeckten Schichten und Neuronen zu erstellen sowie eine ge-

wünschte Aktivierungsfunktion für jede Schicht zu bestimmen. Bei Bedarf hat der Anwender zusätzlich die Möglichkeit, ein Bias-Neuron den gewünschten Schichten hinzufügen.

Training vorbereiten und durchführen

Im *Trainings*-Modus, in den man über den Toggle-Button *Training* gelangt, wird das Training vorbereitet und durchgeführt. Die Trainingsdatensätze können durch den Button *Add Training Data* angelegt werden. Die Anzahl der hinzugefügten Trainingsdatensätze wird dem Benutzer im *Trainings*-Modus zur Orientierung angezeigt. Zusätzlich wird auch signalisiert, ob bereits ein Training durchgeführt wurde. Dieses wird über den Button *Do Training* angestoßen. Die maximalen Trainingsdurchläufe und die angestrebte Fehlerrate kann über *Configure* gesetzt werden. In Abbildung 7.9 sind die einzelnen Bedienelemente des *Trainings*-Modus noch einmal aufgezeigt.

Das Abspeichern und Laden von Trainingszuständen ist über die Buttons *Load* und *Save* möglich.

Abbildung 7.9: Die einzelnen Bedienelemente des *Trainings*-Modus.

Einsatz des neuronalen Netzes

Durch Betätigung des Toggle-Buttons *Go Live* wird das aktuell in der Anwendung verwendete KNN in den Live-Modus überführt. Das heißt die Werte der Eingangsneuronen dienen nicht mehr als Trainingsdaten, sondern werden direkt für die Berechnung der Ausgangsneuronen im KNN verwendet.

In Abbildung 7.10 sind die einzelnen Elemente des Modus aufgezeigt.

Die aktuell errechneten Ergebnisse der einzelnen Ausgangsneuronen stehen dem Benutzer im *Go Live*-Bereich in der Tabelle *Output neurons* zur Verfügung. Zusätzlich hat dieser in der Tabelle *Actuators* eine Übersicht über alle am System angemeldeten Aktoren bzw. Listener.

Go Live

Output neurons:

ID	Name	Value
1	Schlafen	1.0
2	Kochen	0.0

Actuators:

ID	Name
1	Listener 1
2	Listener 2

Abbildung 7.10: Die einzelnen Bedienelemente des *Go Live*-Modus.

7.2.2 Benutzerführung

Die grafische Benutzeroberfläche enthält bestimmte Funktionen, um den Benutzer vor gewissen Eingabefehlern zu bewahren. Das Erstellen und Trainieren von KNN stellt eine komplizierte und unter Umständen auch langwierige Angelegenheit dar. Hierbei soll der Anwender unterstützt werden und durch die Benutzerführung mögliche Fehlerquellen von vornherein unterbunden werden.

Eine erste solche Fehlerquelle stellt die Erstellung des KNN im *Discover*-Modus dar. Hierfür ist es nötig zunächst mindestens ein Eingangsneuron zu erstellen, anschließend die gewünschten verdeckten Schichten zu definieren und letztlich mindestens ein Ausgangsneuron in der Anwendung anzulegen. Erst dann sind die Bedingungen an die Architektur des Netzes erfüllt, um für ein Training verwendet werden zu können und der Benutzer kann den *Trainings*-Modus aufrufen.

Im *Trainings*-Modus besitzt der Benutzer keine Möglichkeit mehr, neue Eingangs- oder Ausgangsneuronen zu erstellen oder bestehende zu entfernen, denn dies könnte zu Inkonsistenzen bei den Trainingsdaten führen. Aufgrund dessen sind die entsprechenden Schaltflächen in diesem Modus deaktiviert.

Möchte der Nutzer trotz alledem Modifikationen in der Anzahl der Neuronen im Netz vornehmen, so muss dieser zurück in den *Discover*-Modus wechseln. Sämtliche Trainingsdaten und Trainingszustände gehen dann jedoch verloren. Der Nutzer wird entsprechend darüber informiert und hat nach Aufruf des *Discover*-Modus noch einmal die Möglichkeit, das Wechseln in den Modus zu bestätigen oder abubrechen, wie dargestellt in Abbildung 7.11.

Abbildung 7.11: Über einen Dialog wird der Benutzer bei Verlassen des *Trainings*-Modus über einen möglichen Datenverlust informiert.

Im *Trainings*-Modus müssen zwei Schritte durchgeführt werden, bevor das KNN im *Go Live*-Modus eingesetzt werden kann: Zunächst muss mindestens ein Trainingsdatensatz erstellt werden, bevor im Anschluss das Training durchgeführt werden kann. Erst nach erfolgreicher Durchführung des Trainings ist es dem Benutzer möglich, in den *Go Live*-Modus zu wechseln.

7.2.3 Komponenten der Benutzeroberfläche

In diesem Abschnitt wird auf die einzelnen Komponenten der Benutzeroberfläche eingegangen. Diese setzt sich aus den drei Bestandteilen der FXML-Datei für das Layout, der eigentlichen Anwendung und dem sogenannten Controller zusammen.

FXML

FXML ist ein auf XML basierendes Deklarationsformat für *JavaFX*. Mit ihm ist es möglich, eine hierarchische Baumstruktur aller grafischen Objekte für eine Benutzeroberfläche zu definieren.

Für das Parsen der einzelnen FXML-Dateien bietet *JavaFX* die Klasse `FXMLLoader` an. Der Loader erstellt anhand der Datei den Szenegraphen und ordnet die einzelnen Grafikobjekte in diesen ein:

```
FXMLLoader loader =
    new FXMLLoader(getClass().
        getResource("/ui/window_smarterhome.fxml"));
```

Mit Hilfe des in Abschnitt 5.6.1 beschriebenen *JavaFX Scene-Builders* wurden die benötigten Benutzeroberflächen der Anwendung erstellt und das daraus erhaltene FXML-Dokument an den entsprechenden Stellen mit Hilfe des *FXML Loaders* eingebunden.

UI Application

Jede *JavaFX*-Anwendung stellt eine Erweiterung der Basisklasse `javafx.application.Application` dar. Dafür wird von dieser Klasse geerbt und deren Lebenszyklusmethoden `init()`, `destroy()` und `stop()` können überschrieben werden. Zwingend ist jedoch eine Implementierung der Methode `start()` nötig.

In der Applikation *SmarterHome* ist diese Erweiterung in Form der Klasse `UIApplication`

realisiert. Sie gilt quasi als Einstiegspunkt in die *JavaFX*-Anwendung. In ihr werden die Grundeinstellungen zum Start definiert, dazu gehören bspw. die initiale Fensterdimension, das Laden des Layouts durch den *FXML Loader* und die Verbindung der Anwendung mit einem sogenannten Controller (mehr dazu siehe Abschnitt 7.2.3).

Zusätzlich werden in der *UI¹Application* die Komponenten und Dienste der Anwendung initialisiert und dem Hauptcontroller übergeben, der Klasse `MainWindowController`.

Controller

Der Controller in der *JavaFX*-Umgebung dient in der Anwendung als Verknüpfung zwischen Benutzeroberfläche und Datenmodell. Dafür wird jedem FXML-Layout ein Controller zugewiesen.

Damit der Controller auf die einzelnen Objekte aus der jeweiligen Benutzeroberfläche zugreifen kann, muss ein Mapping zwischen FXML-Datei und Controller erfolgen.

Dies erfolgt in zwei Schritten, zunächst muss in der FXML-Datei der gewünschte Controller gesetzt werden:

```
fx:controller=
"de.thwildau.smarterhome.ui.controller.MainWindowController"
```

Anschließend kann im Controller selbst, in Form von sogenannten `@FXML`-Annotationen, der Zugriff auf die einzelnen Elemente ermöglicht werden. Dieses Mapping zwischen FXML-Datei und dem Controller erfolgt über die ID des FXML-Elements. Dabei ist zwingend notwendig, dass die zugeordneten Variablen im Controller über die gleiche Bezeichnung verfügen, wie ihr FXML-Äquivalent. Neben den Annotationen für Elemente ist es ebenfalls möglich, auf Event-Handler zuzugreifen, die äquivalente Bezeichnung ist hierfür ebenfalls zwingend:

```
window_smarterhome.fxml:
<Button fx:id="buttonConnect" onAction="#handleConnect"/>
```

```
MainWindowController.java:
@FXML
private Button buttonConnect;

...

@FXML
public void handleConnect(){
    ...
}
```

7.3 Serielle Schnittstelle

Die Bereitstellung der seriellen Schnittstelle in der Anwendung *Smarter Home* dient als Zugangspunkt für Sensoren. Diese haben darüber die Möglichkeit, sich am System anzumelden

1 UI: User Interface

und Sensorwerte zu übertragen. Dabei wird mit Hilfe des Standards ZigBee und einem selbst entwickelten Protokoll ermöglicht, sich drahtlos mit der Schnittstelle zu verbinden und zu kommunizieren. In den nachfolgenden Unterabschnitten wird zunächst auf das entwickelte Protokoll bzw. Nachrichtenformat näher eingegangen und anschließend der allgemeine Aufbau und die Implementierung innerhalb der Anwendung beschrieben.

7.3.1 Nachrichtenformate

Für die sensorseitige Übertragung von Informationen an die Anwendung wurde ein Protokoll entwickelt, das einen einfachen Datenaustausch ermöglicht.

Insgesamt sind zwei Nachrichtentypen für die Kommunikation erforderlich: ein Nachrichtenformat, das das Anmelden an die Anwendung ermöglicht und ein zweites Nachrichtenformat, das die Übertragung von Sensordaten unterstützt.

Nachfolgend wird zunächst das allgemeine Nachrichtenformat erläutert, um anschließend auf die zwei spezifischen Formen einzugehen.

Allgemeines Nachrichtenformat

Bei den Nachrichten, die von einem Sensor verschickt werden, handelt es sich um sogenannte Frames. Solch ein Frame beinhaltet immer genau drei Elemente, in denen Daten verschickt werden können. Das Ende eines Elementes wird mit einem Semikolon als Trennzeichen gekennzeichnet. Handelt es sich um das dritte bzw. letzte Element im Frame, dient ein sogenannter Zeilenumbruch `\n` als Trennzeichen für das Element und den Frame (Frame Delimiter).

In Abbildung 7.12 ist der grundsätzliche Aufbau des Nachrichtenformats dargestellt.

Nachrichtentyp		ID		Beliebige Nutzdaten	Frame-Delimiter
1 Char	","	INT AS CHARS "12"	","	beliebige Anzahl an CHARS	1 CHAR "\n"

Abbildung 7.12: Der grundsätzliche Aufbau eines Frames enthält drei Elemente, die durch ein Semikolon getrennt sind.

Das erste Element besteht aus einem Zeichen vom Datentyp CHAR und dient der Bestimmung des Nachrichtentyps. Da es verschiedene Arten von Nachrichten gibt, müssen diese unterschiedlich verarbeitet werden. Die Angabe des Nachrichtentyps macht eine gesonderte Verarbeitung in Abhängigkeit des Nachrichtentyps möglich.

Bei dem zweiten Element wird die ID übertragen, sodass bspw. ein Zuweisen von Daten zu einem bestimmten Sensor möglich wird.

Das letzte Element enthält die eigentlichen Nutzdaten. Dies können eine beliebige Anzahl an Zeichen des Datentyps CHAR sein.

Für die Nutzung des Protokolls muss auf Sensorseite darauf geachtet werden, dass nicht irrtümlich die im Protokoll definierten Trennzeichen, nämlich das Semikolon und der Zeilenumbruch, innerhalb der Nutzdaten verwendet werden.

Nachrichtenformat *Bonjour*

Das Nachrichtenformat *Bonjour* ermöglicht das Anmelden eines Sensors an die Anwendung. Spezifiziert wird dies im Frame durch den Nachrichtentypen „B“. Anschließend folgt die ID, die für den Sensor in der Anwendung *Smarter Home* angelegt werden soll. Hierbei liegt die Verantwortung auf Seiten der Sensorik, dass die ID in der Anwendung noch nicht vergeben ist. In Abbildung 7.13 ist der Aufbau des *Bonjour*-Frames dargestellt.

Nachrichtentyp		ID		Beliebige Nutzdaten	Frame-Delimiter
"B"	","	z.B. "13"	","	Sensortyp (1 CHAR) "T" "H" ...	"\n"

Abbildung 7.13: Der *Bonjour*-Frame ermöglicht das Anmelden eines Sensors an die Anwendung *Smarter Home*.

Das dritte Element, das die Nutzdaten enthält, definiert den Sensortypen durch die Angabe eines Zeichens in Form eines einzelnen CHAR-Wertes.

Aktuell werden hierfür die Sensortypen *Temperature* T, *Humidity* H und *Binary* B unterstützt.

Ein Beispiel für eine *Bonjour*-Nachricht könnte wie nachfolgend dargestellt aussehen:

B ; 13 ; T \n

Hierbei wird ein neuer Sensor mit der ID '13' am System als Temperatursensor angemeldet.

Nach erfolgreichem Anmelden eines Sensors mit Hilfe der *Bonjour*-Nachricht wird der Sensor anschließend in der Benutzeroberfläche mit seinen definierten Eigenschaften angezeigt.

Nachrichtenformat *Talk*

Das Nachrichtenformat *Talk* ermöglicht das „Sprechen“ eines angemeldeten Sensors mit der Anwendung, d.h. mit diesem Nachrichtenformat ist es möglich, einen Sensorwert an die Anwendung zu übertragen.

Angekündigt wird dieses im Frame durch den Nachrichtentypen 'T'. Die ID im zweiten Element kennzeichnet, welchem angemeldeten Sensor der Sensorwert zu geordnet werden soll. Die Nutzdaten enthalten den aktuellen Sensorwert und werden nach Übermittlung in der Anwendung für den entsprechenden Sensor angezeigt.

Eine Beispielnachricht für eine *Talk*-Nachricht könnte also wie folgt aussehen:

T ; 13 ; 12.25 \n

Dabei handelt es sich um den Sensorwert '12.25', der für den angemeldeten Sensor mit der ID '13' bestimmt ist.

Talk-Nachrichten, deren ID nicht zu einem angemeldeten Sensor zugeordnet werden können, werden von der Anwendung *Smarter Home* ignoriert. In Abbildung 7.14 ist der Aufbau des *Talk*-Frames aufgezeigt.

Nachrichtentyp		ID		Beliebige Nutzdaten	Frame-Delimiter
"T"	","	z.B. "13"	","	Sensorwert (FLOAT AS CHARS) "12.25"	"\n"

Abbildung 7.14: Mit Hilfe des *Talk*-Frames können angemeldete Sensoren ihre Daten an die Anwendung *Smarter Home* senden.

7.3.2 RXTX Communication Service

Der generelle Aufbau zur Kommunikation via ZigBee zwischen Sensoreinheit und Anwendung sieht vor, dass beide Einheiten mit einem ZigBee-Sender bzw. Empfänger ausgestattet werden.

Aufgrund der in Abschnitt 6.3.2 gewählten Hardware wird für den ZigBee-Empfänger ebenfalls ein sogenanntes XBee-Modul von der Firma *Digi International* verwendet. Dieses Modul kann über die serielle USB-Schnittstelle mit einer Kabelverbindung an die Rechneinheit angeschlossen werden, auf dem die Anwendung *Smarter Home* läuft. In Abbildung 7.15 ist der prinzipielle Aufbau inklusive der beteiligten Komponenten noch einmal aufgezeigt.

Abbildung 7.15: Die beteiligten Komponenten für die Kommunikation zwischen einem Sensor und der Anwendung *Smarter Home* im Überblick.

Die Kommunikation über einen seriellen Port in Java ist grundsätzlich möglich, dafür wird jedoch eine zusätzliche Bibliothek benötigt. Für die Implementierung der seriellen Kommunikation auf Anwendungsseite wurde hierfür die Bibliothek *RXTX* verwendet. Neben dieser Bibliothek wird zusätzlich das Framework *Netty* eingesetzt. Traditionell dient dieses asynchrone Client-Server-Framework zur Kommunikation zwischen Java-Anwendungen innerhalb eines Netzwerkes. Dafür bietet es eine Menge an nützlichen Werkzeugen, die jedoch auch für die serielle Datenverarbeitung genutzt werden können. So bietet bspw. das *Netty*-Framework einen Decoder an, mit dem ein komfortables Detektieren einzelner Frames aus dem Bytestrom heraus möglich ist. Mit Hilfe der *ByteBuffer* API von *Netty* können z.B. auch einzelne Nachrichten in die jeweiligen Bestandteile zerlegt werden.

Für die Kommunikation mit dem ZigBee-Empfänger und der Datenverarbeitung in der Applikation ist der *RXTX Communication Service* zuständig. Dieser Service ist hierfür in verschiedene Schichten unterteilt, die die verschiedenen Aufgaben der Datenverarbeitung abhandeln. In Abbildung 7.16 ist der Aufbau der Schichten und der Datenfluss innerhalb der Abarbeitungskette für das Empfangen und Verschicken einer Nachricht dargestellt.

Die ankommenden Nachrichten am ZigBee-Empfänger haben, je nach Nachrichtentyp, eine unterschiedliche Frame-Länge. Aufgrund dessen wird ein Decoder bzw. Encoder benötigt. Der Decoder `RxtxMessageFrameDecoder` hat dabei die Aufgabe, den ankommenden Bytestrom in einzelne Frames zu extrahieren. Der Encoder `RxtxMessageFrameEncoder` agiert entsprechend entgegengesetzt zum Verschicken von Nachrichten. Als sogenannter Frame Delimiter oder auch Trennzeichen zwischen den Frames wird der Zeilenumbruch '\n' verwendet.

Abbildung 7.16: Der *RXTX Communication Service* besteht aus mehreren Schichten, die bei Empfang oder Versenden einer Nachricht durchlaufen werden.

Anstatt direkt auf den Byte-Arrays zu arbeiten wird für die Verarbeitung der Datenfelder einer Nachricht auf die ByteBuffer API von *Netty* zurückgegriffen. Dies bietet mehrere Vorteile:

Die Rohdaten einer Nachricht können einheitlich gekapselt werden.

Zusätzlich ist ein komfortabler Methodenzugriff auf die jeweiligen Datenfelder der Nachricht möglich, im Vergleich zur direkten Verarbeitung auf Byte-Arrays.

Der *XbeeMessageParser* hat die Aufgabe, die einzelnen Nachrichten zu kodieren bzw. zu dekodieren, je nachdem, ob die Nachricht verschickt werden soll oder empfangen wurde. Dabei agiert der Parser in Abhängigkeit des Nachrichtentyps. Handelt es sich um eine Nachricht vom Typ *Bonjour*, so wird der *BonjourMessageInterpreter* verwendet. Bei Nachrichten vom Typ *Talk* entsprechend der *TalkMessageInterpreter*.

7.4 Sensor Manager Service

Für eine ordentliche Verwaltung der Sensoren müssen verschiedene Aufgaben abgedeckt werden, so müssen bspw. neue Sensoren erkannt und der Anwendung hinzugefügt werden. Bei Eintreffen neuer Sensorwerte müssen diese bestehenden Sensoren zugeordnet und entsprechend aktualisiert werden etc.

Für die Verwaltung dieser Aufgaben ist die Klasse **SensorManagerService** zuständig. Diese nutzt den *RXTX Communication Service* (Abschnitt 7.3.2) indem an diesem sogenannte *Listener* oder auch *Observer* registriert werden. Sobald eine *Bonjour*- oder auch *Talk*-Nachricht an diesem Service eintrifft, wird der *Sensor Manager Service* darüber informiert. Dieser überträgt anschließend, in Abhängigkeit vom Nachrichtentyp, die Daten aus den Nachrichten in das entsprechende Datenmodell aus der Anwendung. Dabei handelt es sich um die Modell-Klasse **Sensor**, die einen einzelnen Sensor repräsentiert.

Die Klasse **Sensor** beinhaltet dabei alle benötigten Eigenschaften, um die erhaltenen Daten aus einer Nachricht abbilden zu können:

- ID

- Sensortyp
- Aktueller Sensorwert

Jedem **Sensor**-Objekt kann ebenfalls ein Listener hinzugefügt werden, der wiederum darüber informiert, sobald eine Änderung aufgetreten ist.

Erhält nun der *Sensor Manager Service* eine Notifikation über das Eintreffen einer *Bonjour*-Nachricht über die serielle Schnittstelle, so wird zunächst anhand der übermittelten ID überprüft, ob es sich um einen bereits bekannten Sensor handelt. Ist dies der Fall, wird die Nachricht ignoriert. Andernfalls, also bei einem bisher unbekannten Sensor, wird er hinzugefügt und die entsprechenden Listener über das Hinzufügen informiert.

Bei Eintreffen einer *Talk*-Nachricht wird, ebenfalls mit Hilfe der ID, der passende aus den bereits vorhandenen Sensoren herausgesucht. Adressiert die *Talk*-Nachricht einen nicht bekannten Sensor, so wird die Nachricht verworfen. Im anderen Fall wird der entsprechende Sensor aktualisiert und die registrierten Listener über die Veränderung informiert.

In Abbildung 7.17 sind die für die Verarbeitung der Sensoren beteiligten Komponenten der Anwendung nochmals dargestellt.

Abbildung 7.17: Der *Sensor Manager Service* ist für die Verwaltung der Sensoren zuständig. Dafür nutzt dieser den *RXTX Communication Service*, um über Änderungen und neue Sensoren informiert zu werden.

7.5 Deep Learning Service

Der *Deep Learning Service* nimmt in der Anwendung *Smarter Home* die zentrale Komponente ein. Der Service ist für die Verwaltung der einzelnen Elemente des KNN zuständig, dazu zählen die Eingangs- und Ausgangsneuronen, die Architektur und Konfiguration des Netzes sowie deren jeweiligen bereitgestellten Funktionen. In den nachfolgenden Ausführungen wird auf die Umsetzung der einzelnen Komponenten mit Hilfe des *Deep Learning Service* eingegangen.

7.5.1 Eingangs- und Ausgangsneuronen

Die Eingangs- und Ausgangsneuronen stellen in einem KNN zwei besondere Neuronentypen dar. Über die Eingangsneuronen in der Eingangsschicht gelangen Eingangs-

werte in das KNN, die Ausgangsneuronen wiederum enthalten die berechneten Ausgangswerte, die als Ergebniswerte dienen.

Die Eingangs- und Ausgangsneuronen sowie deren Eigenschaften werden in der Anwendung *Smarter Home* durch spezielle Modellklassen realisiert und durch den *Deep Learning Service* verwaltet. Nachfolgend wird auf diese Realisierung näher eingegangen.

In Abbildung 7.18 ist der Aufbau der Klassenhierarchie zur Abbildung der einzelnen Neuronen in der Anwendung dargestellt.

Abbildung 7.18: Der hierarchische Aufbau der einzelnen Klassen zur Abbildung der jeweiligen Neuronentypen.

Einfache Eingangsneuronen

Der *Sensor Manager Service* (siehe Abschnitt 7.4) ist für die Verwaltung der Sensoren im System zuständig, deren Werte sollen als Eingangsdaten für das KNN dienen. Hierfür müssen die Sensoren mit den Eingangsneuronen des Netzes verbunden werden, dies ist eine Aufgabe des *Deep Learning Service*, dieser dient quasi als Verbindungselement zwischen Sensorik und KNN. Zusätzlich hat der Service die Aufgabe, die Daten aus dem Override-Modus der Benutzeroberfläche entgegen zu nehmen, um damit das KNN zu beeinflussen.

Der *Deep Learning Service* verwendet für die Erstellung der Eingangsneuronen die Modell-Klasse **InNeuron**. Sie dient der Entkopplung der „echten“ Sensoren, indem sie diese kapselt. Dadurch ist es beispielsweise möglich, die Werte eines „echten“ Sensors über die Override-Funktion zu ändern. Zusätzlich ist auch das Anlegen, Manipulieren und Entfernen von Sensoren möglich, ohne dass diese physikalisch vorhanden sind.

Befindet sich die Anwendung im sogenannten *Live*-Modus überwacht der *Deep Learning Service* die vom *Sensor Manager Service* verwalteten Sensoren und je nach Änderung wird entweder ein neues Objekt vom Typ **InNeuron** erstellt oder der Wert eines bestehenden entsprechend angeglichen.

Sobald sich die Anwendung im Override-Modus befindet, werden keine Werte der physikalischen Sensoren über den *Sensor Manager Service* mehr übernommen, sondern der Wert eines **InNeuron** kann dann ausschließlich über die Benutzeroberfläche verändert werden. Dadurch wird die Simulation von Sensorwerten realisiert.

Spezielle Eingangsneuronen

Neben den „normalen“ Eingangsneuronen für das KNN gibt es noch spezielle Eingangsneuronen, die keinen „physikalischen“ oder simulierten Sensoren repräsentieren. Dabei handelt es sich um die Eingangsneuronen, die für das Einfließen der Zeitinformation in das KNN zuständig sind und von der Anwendung in regelmäßigen Abständen und nach definierten Regeln aktualisiert werden.

So gibt es einen Neuronentyp für die Zeit, der die aktuelle Uhrzeit als Eingangswert für das KNN liefert. Hierfür wird die Uhrzeit in einen Float-Wert umgewandelt, sodass das KNN dazu in der Lage ist damit zu arbeiten. Die Uhrzeit 12:30 wird entsprechend in den Float-Wert 12.5 umgerechnet. In der Anwendung *Smarter Home* wird diese Art von Neuron durch den Typ `TIME_OF_DAY` angelegt. Zusätzlich ist es möglich, den Wochentag als Zeitinformation zu nutzen. Dafür kann über die Benutzeroberfläche ein Neuron vom Typ `DAY_OF_WEEK` erstellt werden, wie in Abbildung 7.19 dargestellt. Der Wochentag wird ebenfalls als Float-Wert bereitgestellt, in dem Zahlenbereich von 1.0 - 7.0.

Der Wochentag Montag wird dabei in Form einer '1.0' und Mittwoch durch eine '3.0' repräsentiert.

Im Live-Modus ist der *Deep Learning Service* für das Berechnen und Aktualisieren der Zeitinformationen zuständig. Im Override-Modus werden die von der Anwendung generierten Werte ignoriert und lediglich Änderungen durch den Anwender über die Benutzeroberfläche berücksichtigt.

Abbildung 7.19: Bei der Erstellung eines Eingangsneurons kann der Typ festgelegt werden, wie z.B. `TIME_OF_DAY` oder `DAY_OF_WEEK`.

Ausgangsneuronen

Die Ausgangsneuronen werden in der Anwendung durch die Modell-Klasse `OutNeuron` abgebildet. Sobald das neuronale Netzwerk die einzelnen Ausgänge berechnet hat, hat der *Deep Learning Service* die Aufgabe, diese in dem entsprechenden `OutNeuron`-Objekt zu setzen.

Die Werte der Ausgangsneuronen sind vor allem für die Aktoren bedeutsam. Die einzelnen Ausgangsneuronen werden über die Klasse `ActuatorManagerService` zur Verfügung gestellt. Dieser Service ermöglicht auch das spätere Abrufen der Informationen über die REST API (siehe Abschnitt 7.8.2).

7.5.2 Neuronales Netzwerk

Die Klasse `NeuralNetwork` bildet das Verbindungselement zwischen dem *Deep Learning Service* und dem Framework *Encog*. Dabei bildet die Klasse eine Abstraktionsschicht zwischen den beiden Schichten (Abbildung 7.20).

Für den *Deep Learning Service* bleibt die eigentliche Struktur und Funktionsweise des *Encog*-Frameworks verborgen, der Service kommuniziert ausschließlich mit der Abstraktionsschicht, dem `NeuralNetwork`.

Diese Entkopplung ermöglicht ein einfaches Austauschen des Frameworks *Encog*, ohne dass dabei ein Anpassen der oberen Programmschichten nötig wird.

Praktisch gesehen erstellt die Klasse `NeuralNetwork` mit Hilfe der Eingangs- und Ausgangsneuronen, die über den *Deep Learning Service* bereitgestellt werden, ein KNN, indem es diese auf die äquivalenten Klassen des *Encog*-Frameworks abstrahiert.

Neben der Erstellung des Netzwerkes ist die Abstraktionsklasse zusätzlich für die Verwaltung, Konfiguration und Durchführung des Trainings zuständig.

Ein Training wird mit mindestens einem oder mehreren Trainingsdatensätzen durchgeführt. Ein einzelner Trainingsdatensatz besteht aus den aktuellen Werten der Eingangsneuronen, repräsentiert durch ein jeweiliges `InNeuron` sowie den jeweils gewünschten bzw. gesetzten Werten der Ausgangsneuronen (`OutNeuron`).

Unter Verwendung dieser Daten wird dem KNN beigebracht, mit welchen Eingangswerten welche Ausgangswerte zu erzeugen sind. Mit der Durchführung des Trainings werden dem Netz genau diese Werte bzw. dieses Verhalten angelernt. Das Training setzt sich aus mehreren Trainingsiterationen zusammen. Nach jeder Iteration wird das bisherige Trainingsergebnis auf die aktuelle Fehlerrate und die aktuelle Durchlaufanzahl hin überprüft und mit der 'Minimum error rate' bzw. den 'Maximum iterations' abgeglichen, denn diese stellen die Abbruchbedingung dar.

Die beiden Bedingungen zum Abbruch des Trainings werden im *Deep Learning Service* gesetzt und können im `NeuralNetwork` verwendet werden.

Die Ausführung des trainierten Netzes im *Go Live*-Modus wird ebenfalls über die Abstraktionsklasse angestoßen.

Abbildung 7.20: Die Abstraktionsklasse `NeuralNetwork` dient als Vermittler zwischen dem *Deep Learning Service* und dem Framework *Encog*

7.6 Encog Framework

Zur Veranschaulichung der Arbeitsweise des *Encog*-Frameworks wird in diesem Abschnitt anhand eines Beispiels erklärt, wie das Framework funktioniert und wie es in dem Projekt umgesetzt wurde.

7.6.1 Erstellung des neuronalen Netzes

Für die Erstellung eines KNN in *Encog* stellt das Framework die Klasse `BasicNetwork` bereit. Über diese Klasse kann zunächst ein *rudimentäres* KNN erstellt werden, das dann um die gewünschten Eigenschaften und Netzschichten ergänzt wird:

```
BasicNetwork network = new BasicNetwork();
```

Über die Methode `addLayer()` erfolgt das Hinzufügen der einzelnen Layer, dabei repräsentiert die Reihenfolge der Erstellung der Schichten auch den Aufbau des Netzes.

Die erste hinzugefügte Schicht stellt demnach immer die Eingangsschicht mit den enthaltenen Eingängen bzw. Eingangsneuronen dar:

```
network.addLayer(new BasicLayer(null, true, 5));
```

In diesem Fall handelt es sich um eine Eingangsschicht, die 5 Neuronen enthält. Über den ersten Parameter kann die Aktivierungsfunktion übergeben werden, für die Eingangsschicht wird jedoch diese nicht benötigt. Der zweite Parameter ermöglicht das Verwenden eines Bias-Neurons. [26]

Jeder weitere hinzugefügte Layer stellt eine verdeckte Schicht im KNN dar:

```
network.addLayer(new BasicLayer(new ActivationSigmoid(), true, 8));
```

Der als letztes hinzugefügte Layer bildet die Ausgangsschicht und deren Ausgänge, in diesem Fall sind dies genau zwei Ausgänge:

```
network.addLayer(new BasicLayer(new ActivationSigmoid(), false, 2));
```

In diesem Beispiel wurde ein KNN erstellt, das insgesamt über 3 Schichten verfügt. Neben der Anzahl der Neuronen können bei der Erstellung einer Schicht auch die Verwendung eines *Bias-Neurons* (Schicht 1 und 2) als Schwellwert sowie eine Aktivierungsfunktion (Schicht 2 und 3) übergeben werden. [26]

In Abbildung 7.21 ist das erstellte Beispiel des KNN dargestellt.

Sobald die Struktur des Netzes fertig konstruiert ist, muss diese finalisiert werden. Mit Hilfe des Befehls `reset()` können und sollten die Gewichte der Neuronen vor jedem Training zurückgesetzt werden:

```
network.getStructure().finalizeStructure();  
network.reset();
```

Das KNN ist nun erstellt und kann im nächsten Schritt mit Hilfe von Trainingsdaten antrainiert werden. [26]

Abbildung 7.21: Das in *Encog* erstellte KNN verfügt über 3 Schichten und 2 Bias-Neuronen für die erste und zweite KNN-Schicht.

7.6.2 Trainieren des Neuronalen Netzes

In diesem Abschnitt sollen nun insgesamt 100 Trainingsdatensätze für ein KNN mit 5 Eingangs- und 2 Ausgangsneuronen erstellt werden. Hierfür ist es zunächst nötig, die passenden Matrizen anzulegen, die die Trainingsdaten beinhalten sollen:

```
double[][] input = new double[100][5];
double[][] ideal = new double[100][2];
```

Anschließend müssen die Matrizen mit den gewünschten Daten gefüllt werden. Nachdem dies erfolgt ist, können die Daten dem Datenset von *Encog* für das Training übergeben werden:

```
MLDataSet trainingSet = new BasicMLDataSet(input, ideal);
```

Nun kann das Training vorbereitet werden und im Anschluss durchgeführt werden. Die Trainingsdurchläufe werden solange wiederholt, bis die minimale Fehlerrate (`MIN_ERROR_RATE`) oder die maximale Anzahl an Durchläufen (`MAX_ITERATIONS`) erreicht wurde:

```
MLTrain train = new ResilientPropagation(network, trainingSet);

int iteration = 0;
do {
    train.iteration();
    iteration++;
} while ((train.getError() > MIN_ERROR_RATE)
        || (iteration <= MAX_ITERATIONS));
```

Nach Durchlaufen der Schleife ist das Netzwerk trainiert und das Wissen des KNN steckt nun in den Gewichten der einzelnen Neuronen. [26]

7.6.3 Einsatz des neuronalen Netzes

Das antrainierte Netz kann anschließend eingesetzt werden, z.B. für den *Go Live*-Modus der Anwendung *Smarter Home*. Hierfür müssen die Eingangsdaten in einem Datenfeld vom Typ `MLData` hinterlegt werden:

```
double[] input = new double[5];  
MLData inputData = new BasicMLData(input);
```

Nachfolgend können mit Hilfe des KNN die Ausgangsneuronen berechnet werden. Hierfür dient der Befehl `compute()`, diesem werden die Eingangsdaten `inputData` übergeben:

```
final MLData outputData = network.compute(inputData);  
double[] output = outputData.getData();
```

In dem Array `output` sind nun die Werte der Ausgangsneuronen enthalten. [26]

7.7 Speichern und Laden von KNN

Das Persistieren und Laden eines trainierten KNN über die Anwendung wird in diesem Abschnitt erläutert. Dabei wird zunächst auf die benötigten Informationen zur erfolgreichen Abspeicherung und anschließendem Laden eines KNN eingegangen und anschließend auf die konkrete Umsetzung der Funktionen.

7.7.1 Relevante Informationen

Die relevanten abzuspeichernden Informationen setzen sich aus mehreren Bereichen zusammen. Zunächst muss das trainierte KNN selbst gesichert werden, darunter fallen die Netzwerkarchitektur und deren Konfiguration (einzelne Schichten, Anzahl der Neuronen, Kantenverbindungen etc.) sowie der aktuelle Zustand der einzelnen Gewichtungen der Kantenverbindungen zwischen den Neuronen.

Neben den grundlegenden Informationen gibt es darüber hinaus Zusatzinformationen, die ebenfalls für das Abspeichern relevant sind, jedoch nicht in dem KNN selbst hinterlegt sind. Bei den Eingangsneuronen sind dies bspw. Eigenschaften wie Typ, ID und Name. Gleiches gilt für die Ausgangsneuronen, die neben der ID auch über einen Namen verfügen.

Das Erstellen von geeigneten und qualitativ hochwertigen Trainingsdaten kann in dem Trainingsprozess einen aufwändigen und zeitlich intensiven Faktor darstellen. Aufgrund dessen werden, neben dem eigentlichen KNN und den nötigen Zusatzinformationen, die generierten Trainingsdatenätze zu dem KNN abgespeichert. Beim Laden des gewünschten KNN steht dem Nutzer damit nicht nur das trainierte KNN zur Verfügung, sondern auch die bereits angelegten Trainingsdaten. In der nachfolgenden Aufzählung sind die zu speichernden Elemente noch einmal aufgezählt.

1. Architektur und Konfiguration des KNN
2. Zustand des KNN
3. Zusatzinformationen zu den Eingangs- und Ausgangsneuronen
4. Trainingsdaten

7.7.2 Speichervorgang

Das *Encog*-Framework bringt bereits eine Funktion für das Abspeichern des Zustands eines KNN mit. Für das Persistieren wurde auf die entsprechenden Funktionen der Klasse *EncogDirectoryPersistence* von *Encog* zurückgegriffen, welches das serialisierte Abspeichern ermöglicht sowie ein späteres Laden.

Neben dem Zustand des Netzes müssen jedoch auch die Architektur und Konfiguration des KNN, die Eingangs- und Ausgangsneuronen sowie die Trainingsdaten hinterlegt werden. Aufgrund dessen wurde sich dafür entschieden, die zu speichernden Daten in die drei folgenden Dateien aufzuteilen:

1. Netzwerkaufbau und -konfiguration (Datei *shnn.cfg*)
2. Zustand des Netzwerkes (Datei *shnn.network*)
3. Trainingsdaten (Datei *shnn.training*)

Im Speichervorgang werden diese drei Dateien einzeln persistiert und anschließend in einer einzelnen Datei zusammengeführt, um anschließend im ZIP-Dateiformat komprimiert abgelegt zu werden.

Diese Handhabung soll vor allem für eine leichtere Nutzung auf Seiten des Benutzers sorgen. Denn mit dieser Variante muss dieser beim Laden nicht drei Dateien handhaben, sondern lediglich eine.

Die nicht von *Encog* persistierten Informationen bzw. Dateien werden im JSON-Format abgespeichert. Die über *Encog* gespeicherten Informationen legt das Framework in einem proprietären Format ab.

In der Benutzeroberfläche steht dem Anwender im *Trainings*-Modus die entsprechende Speicher- und Ladefunktion über die Buttons *Load* und *Save* zur Verfügung.

In Abbildung 7.22 ist der Dialog zum Abspeichern eines KNN dargestellt.

Abbildung 7.22: Das KNN und die Trainingsdaten werden als SHNN-Datei gespeichert und können über die Anwendung auch wieder geladen werden.

7.8 Embedded REST Server

Der `EmbeddedRestServer` dient in der Anwendung *Smarter Home* als Schnittstelle für Aktoren, z.B. in Form des humanoiden Roboters *NAO*. Die Aktoren haben über die REST-Schnittstelle die Möglichkeit, mit der Anwendung zu kommunizieren und bspw. den Zustand der Ausgangsneuronen des KNN abzufragen.

7.8.1 Serverarchitektur

Für die Implementierung des REST Servers wurden die beiden Bibliotheken *Jetty* (Eclipse) und *Jersey* (Glassfish) verwendet.

Der *Jetty* Server wird in der Anwendung als lokale Instanz in Form eines eingebetteten Servers gestartet. Nach erfolgreichem Start können die einzelnen sogenannten Endpoints an dem Servlet Container (siehe Abbildung 7.23) registriert werden.

Ein Endpoint ist in der REST-Umgebung ein Verweis auf einen URI¹, über den ein Objekt des REST-Servers verfügbar ist.

Während der Laufzeit greift der Webserver bei einer Anfrage auf den Servlet Container zu. Über die darin registrierten Endpoints können die angefragten Informationen abgerufen werden.

Abbildung 7.23: Nach Start des REST-Servers registrieren sich die Endpoints an dem Servlet-Container.

Die Endpoints wiederum greifen auf die Klassen `DeepLearningService` und `ActuatorManager` der Verarbeitungseinheit zu, um an die angefragten Informationen zu gelangen. Die Antwort an den Aktor erfolgt in Form eines HTTP Response über das Datenformat JSON.

7.8.2 Aufbau REST API

Über einen HTTP Request kann ein Client bzw. Aktor eine Anfrage an den REST Server schicken. Hierfür muss der Server eine Programmierschnittstelle in Form einer API bereitstellen, über die die einzelnen Aktionen von Seiten des Clients auf dem Server ausgelöst

¹ URI: Uniform Resource Identifier

werden können. Die API für diesen REST Server wird in diesem Abschnitt vorgestellt.

Aktuell ist der REST Server in der Lage die folgend aufgeführten Anfragen zu bedienen:

- API Version abrufen
- Aktor am REST Server bekannt machen
- Aktor am REST Server abmelden
- Abrufen aller Eingangsneuronen
- Abrufen einzelner Eingangsneuronen
- Abrufen aller Ausgangsneuronen
- Abrufen einzelner Ausgangsneuronen

Neben dem Abrufen der Ausgangsneuronen ist es dem Client ebenfalls möglich, die Eingangsneuronen und deren Zustand abzurufen. Dadurch soll dem Client zusätzliche Optionen an die Hand gegeben werden, um eine Situation bewerten zu können

Nachfolgend wird auf die einzelnen Anfragemöglichkeiten nochmals näher eingegangen.

API Version abrufen

Diese Anfrage dient der Möglichkeit zur Überprüfung für den Aktor, ob dieser sich mit dem richtigen Server verbunden hat und welche Protokoll-Version seitens des Servers unterstützt wird.

Gerade auch um eine spätere Abwärtskompatibilität gewährleisten zu können, kann solch eine Funktion hilfreich sein.

Request:

```
GET http://localhost:<port>/api/info
```

Bei dieser Anfrage genügt ein einfacher GET-Befehl, um an die Information zur API-Version zu gelangen.

Response:

```
{
  "name" : "Smarter Home API",
  "apiVersion" : "1.0"
}
```

Aktor am REST Server bekannt machen

Dieser Endpoint ist vergleichbar mit der Bonjour-Nachricht für die Sensoren der seriellen Schnittstelle. Über diesen Endpoint hat der Aktor die Möglichkeit, sich an dem System *Smarter Home* anzumelden. Jeder am Server bekannt gemachte Aktor wird in der Benutzeroberfläche der Anwendung angezeigt.

Request:

```
POST http://localhost:<port>/api/discovery/bonjour
```

Die POST-Methode benötigt für das Anmelden des Listeners deren ID und Namen. Diese Informationen müssen entsprechend in dem Request enthalten sein:

```
{
  "id" : "13",
  "name" : "NAO Amelie"
}
```

Response:

Bei erfolgreicher Anmeldung erhält der Client den HTTP-Status 200 (OK), der eine erfolgreiche Bearbeitung der Anfrage bestätigt.

Aktor am REST Server abmelden

Dieser Endpoint bildet das Pendant zum voran vorgestellten Endpoint und meldet den Listener von der Anwendung *Smarter Home* wieder ab. Der Listener verschwindet nach erfolgreicher Abmeldung ebenfalls in der Anzeige der Benutzeroberfläche.

Request:

```
POST http://localhost:<port>/api/discovery/aurevoir
```

Die POST-Methode benötigt für das Anmelden des Listeners deren ID und Namen. Diese Informationen müssen entsprechend in dem Request enthalten sein.

```
{
  "id" : "13",
  "name" : "NAO Amelie"
}
```

Response:

Bei erfolgreicher Abmeldung erhält der Client ebenfalls den HTTP-Status 200 (OK), der das erfolgreiche Abmelden des Listeners bestätigt.

Abrufen aller Eingangsneuronen

Mit Hilfe dieser Anfrage ist es möglich, alle Eingangsneuronen, die aktuell in der Anwendung angemeldet sind, abzurufen. Die Antwort enthält entsprechend die einzelnen Eingangsneuronen und deren Eigenschaften.

Request:

```
GET http://localhost:<port>/api/service/input
```

Die GET-Methode benötigt keine zusätzlichen Daten zum Abruf der gesamten Eingangsneuronen.

Response:

```
[
  {
    "id" : 1,
    "value" : 1.0,
    "name" : "Neuron 1",
    "linkedSensor" : null,
    "sensorType" : "BINARY",
    "inOverrideMode" : false
  },
  {
    "id" : 2,
    "value" : 22.5,
    "name" : "Neuron 2",
    "linkedSensor" : null,
    "sensorType" : "TEMPERATURE",
    "inOverrideMode" : false
  }
]
```

Abrufen einzelner Eingangsneuronen

Diese Anfrage ermöglicht das Abrufen eines einzelnen Eingangsneurons.

Request:

```
GET http://localhost:<port>/api/service/input/<id>
```

Die GET-Methode benötigt keine zusätzlichen Daten zum Abruf eines einzelnen Eingangsneurons. Über den URI muss jedoch das gewünschte Eingangsneuron durch Angabe der <id> übergeben werden.

Response:

```
{
  "id" : 1,
  "value" : 1.0,
  "name" : "InNeuron 1",
  "linkedSensor" : null,
  "sensorType" : "BINARY",
  "inOverrideMode" : false
}
```

Abrufen aller Ausgangsneuronen

Durch das Abrufen der Ausgangsneuronen hat der Client die Möglichkeit, den Zustand bezüglich bestimmter Tätigkeiten zu erhalten.

Über diesen Endpoint ist es möglich, alle Ausgangsneuronen und deren aktuellen Wert zu erhalten.

Request:

```
GET http://localhost:<port>/api/service/output
```

Die GET-Methode benötigt keine zusätzlichen Daten zum Abruf der gesamten Ausgangsneuronen.

Response:

```
[
  {
    "id" : 1,
    "value" : 0.0,
    "name" : "OutNeuron 1"
  },
  {
    "id" : 2,
    "value" : 1.0,
    "name" : "OutNeuron 2"
  }
]
```

Abrufen einzelner Ausgangsneuronen

Wie bereits bei den Eingangsneuronen gibt es auch hier die Option, ein einzelnes Ausgangsneuron abzurufen. Die Bestimmung des Neurons erfolgt auch hier wieder über die Angabe der ID.

Request:

```
GET http://localhost:<port>/api/service/output/<id>
```

Die GET-Methode benötigt keine zusätzlichen Daten, über den URI muss jedoch das gewünschte Ausgangsneuron über die <id> angegeben werden.

Response:

```
{
  "id" : 1,
  "value" : 0.0,
  "name" : "OutNeuron 1"
}
```

7.9 Abweichungen vom Konzept

In diesem Abschnitt werden nachfolgend auf die Abweichungen vom Konzept eingegangen und entsprechend begründet.

7.9.1 Simulieren und Überschreiben von Eingangsneuronen

Die Anwendung *Smarter Home* ermöglicht es dem Benutzer, neben der Benutzung von realen Sensoren und realer Sensorwerte, die Simulation von Sensoren und das Überschreiben der Werte von realen oder simulierten Sensoren.

Ursprünglich sollte die Funktion zum Überschreiben dieser Werte in der Anwendung mit Hilfe der beiden Toggle-Buttons *Live Data* und *Override* realisiert werden. Dabei sollte bei Aktivieren des Toggle-Buttons *Override* der Benutzer die Möglichkeit verfügen, durch Anklicken der gewünschten Zelle, den Wert darin zu verändern. Reale über die XBee-Verbindung empfangene Sensorwerte werden in diesem Fall ignoriert. Bei Aktivierung des *Live Data*-Toggle-Buttons hat der Benutzer hingegen die Möglichkeit, reale Sensorwerte zu verwenden. Bei Eingangsneuronen mit Zeitinformationen werden im *Live Data*-Modus die aktuelle Uhrzeit oder der aktuelle Wochentag verwendet und angezeigt. In Abbildung 7.24 ist die geplante Realisierung durch die Toggle-Buttons *Live Data* und *Override* dargestellt.

Abbildung 7.24: Ursprünglich wurde die Anwendung so geplant, dass der Live-Modus und die Simulation der Werte für alle Eingangsneuronen ausschließlich über die Buttons *Live Data* und *Override* gesteuert werden.

Das Problem dieser Lösung besteht darin, dass alle Eingangsneuronen sich entweder im Override-Modus befinden, in dem die Daten simuliert werden können, oder alle mit realen Live-Daten arbeiten. Es ist jedoch nicht möglich, dass zur gleichen Zeit einzelne Eingangsneuronen mit realen Sensorwerten arbeiten und andere mit simulierten Werten.

Gerade für den Einsatz im praktischen Abschnitt des Lernbaukastens ist diese Option jedoch wünschenswert. Wenn die Schüler mit Hilfe des Legoprototyps und der RFID-Sensorik die Überwachung des Schlafrhythmus von Karin mit ihrem trainierten Netz testen wollen, soll-

te das Eingangsneuron, das mit dem RFID-Sensor verknüpft ist, die Live-Daten des realen Sensors verarbeiten, und die Uhrzeit über die Anwendung simulierbar sein.

Aufgrund dessen wurde zusätzlich zu den beiden Toggle-Buttons *Live Data* und *Override* eine weitere Funktion eingebaut, die ein explizites Freischalten von einzelnen Eingangsneuronen für den einen oder anderen Modus ermöglicht, ungeachtet des aktuell ausgewählten Toggle-Buttons.

Dafür wurde in der Tabelle *Input Neurons* die zusätzliche Spalte *Override* hinzugefügt. Diese ermöglicht das Einstellen des Modus durch Anklicken des darin hinterlegten Buttons. In Abbildung 7.25 ist diese Funktion dargestellt, dabei befindet sich die Anwendung im *Discover*-Modus und die Eingangsneuronen 1 und 3 sind im *Override*-Modus und deren Werte können jeweils simuliert werden. Das Eingangsneuron mit der ID 2 ist jedoch auch auf *live* gesetzt und berechnet die reale aktuelle Uhrzeit (22:30 Uhr) und diese wird als Eingangswert für das KNN verwendet.

Abbildung 7.25: Durch das Einfügen einer zusätzlichen Spalte in der Tabelle *Input neurons*, können einzelne Eingangsneuronen in den *Live*- oder *Override*-Modus überführt werden.

Durch diese zusätzlich eingefügte Funktion konnte das bei der Umsetzung festgestellte Problem erfolgreich gelöst und der geplante Praxis-Lernbaustein wie ursprünglich beabsichtigt umgesetzt werden. Die Toggle-Buttons *Live Data* und *Override* wurden als einfache Buttons umfunktioniert, die durch Betätigung die gesamten Eingangsneuronen entweder in den *Live-Data*-Modus oder den *Override*-Modus versetzen.

7.9.2 Importieren von Trainingsdaten

Bei der Erstellung der Trainingsdaten wurde in dem Konzept festgelegt, dass der Benutzer über genau eine Möglichkeit verfügt, um Trainingsdaten anzulegen. Durch Betätigung des Buttons *Add Training Data* im *Trainings*-Modus wird der aktuelle Zustand der Eingangsneuronen als einzelner Trainingsdatensatz angelegt. Jeder Trainingsdatensatz muss mit dieser Lösungsvariante einzeln konfiguriert und hinzugefügt werden. Gerade bei einer großen Menge an Trainingsdatensätzen erweist sich dieser Erstellungsprozess recht schnell als sehr aufwändig.

Die Schüler haben in ihrer Praxisphase die Aufgabe, verschiedene Varianten und Konfigurationen mit Hilfe von Trainingsdaten zu testen, um dabei ein geeignetes KNN zur Überwa-

chung von Karins Schlafrhythmus zu erstellen. Dabei soll das Erzeugen der Trainingsdaten zwar einen wichtigen aber nicht den zentralen Vorgang bei der Erstellung eines geeigneten KNN einnehmen.

Aufgrund dessen wurde eine zusätzliche Option für den Benutzer implementiert, mit der es möglich ist, kurzerhand eine geeignete Menge an Trainingsdatensätzen zu generieren und diese im *Trainings*-Modus zu verwenden. Hierfür bietet die Anwendung *Smarter Home* das Importieren von Trainingsdatensätzen über eine sogenannte CSV¹-Datei an. Dabei wird eine Zeile in der Datei als ein Trainingsdatensatz interpretiert. Die einzelnen durch Semikolon getrennten Datenwerte eines Trainingsdatensatzes repräsentieren dabei die einzelnen Eingangs- und Ausgangsneuronen in der Anwendung. Dabei muss demnach die Anzahl der erstellten Eingangs- und Ausgangsneuronen mit der Anzahl an Datenwerten eines Datensatzes übereinstimmen.

Discover

Connect ... Live Data Override Configure Network ...

Input neurons:

ID	Name	Type	Value	Override
1	RFID	BINARY	1.0	live
2	Uhrzeit	TIME_OF_DAY	21.0	live

Output neurons:

ID	Name	Value
1	Schlafen	false

Abbildung 7.26: In der Anwendung sind zwei Eingangsneuronen angelegt, mit dessen Werten das Training durchgeführt werden kann. Über das Ausgangsneuron wird im Training festgelegt, ob es sich um einen positiven (true) oder negativen (false) Trainingsdatensatz handelt.

Abbildung 7.26 zeigt eine Konfiguration in der Anwendung *Smarter Home* mit den zwei Eingangsneuronen *RFID* und *Uhrzeit* sowie dem Ausgangsneuron *Schlafen*. Abbildung 7.27 zeigt die passende CSV-Datei zu der Konfiguration mit insgesamt vier Trainingsdatensätzen. Die erste Spalte enthält jeweils den Wert für den RFID-Sensor, die zweite die jeweilige Uhrzeit und die letzte Spalte, ob es sich bei der Konstellation der beiden Eingangswerten um einen positiven oder negativen Trainingsdatensatz bezüglich des Ausgangsneurons *Schlafen* handelt. Bei dem ersten Trainingsdatensatz handelt es sich also um einen positiven Trainingsdatensatz, der den normalen Zustand beschreibt, dass um 7:00 Uhr morgens der RFID-Sensor den Wert '1' sendet bzw. Karin um die Uhrzeit in ihrem Bett liegt und schläft. Nicht normal ist es hingegen, wenn Karin um 23:00 Uhr abends noch nicht in ihrem Bett liegt, wie beschrieben im dritten Trainingsdatensatz.

¹ CSV: Comma-Separated Values

Abbildung 7.27: In der CSV-Datei repräsentiert eine Zeile die durch Semikolon getrennten Daten für einen Trainingsdatensatz.

Beim Einlesen der CSV-Datei durch die Anwendung wird zum jetzigen Zeitpunkt nicht überprüft, ob die einzelnen Datenwerte eines Datensatzes mit der Anzahl an konfigurierten Eingangs- und Ausgangsneuronen übereinstimmen. Die Sorgfaltspflicht obliegt in diesem Fall auf Seiten des Erstellers der Trainingsdatensätze.

In der Anwendung *Smarter Home* wurde die zusätzliche Funktion zum Einlesen von Trainingsdatensätzen aus einer CSV-Datei über den Button *Import training from CSV* realisiert.

7.10 Gesamtübersicht

In diesem Abschnitt werden die einzelnen Komponenten der Anwendung *Smarter Home* noch einmal zusammengefasst sowie deren Zusammenspiel aufgezeigt.

In Abbildung 7.28 wird eine Übersicht zu den einzelnen Komponenten der Anwendung und deren Interagieren untereinander aufgezeigt. Das User Interface ist dabei für die Darstellung der einzelnen Anwendungsfunktionen verantwortlich. Dafür greift es auf die verschiedenen Dienste und Komponenten der Anwendung zu, indem es sich bei den verschiedenen Services registriert. Bei Änderungen, wie beispielsweise einem neuen Sensor, wird das User Interface von dem Service entsprechend darüber informiert.

Die serielle Schnittstelle für das Einbringen von Sensoren ist über den *RXTX Communication Service* realisiert. Dieser empfängt die einzelnen Sensoren und Sensordaten. Der *RXTX Communication Service* arbeitet mit dem *Sensor Manager Service* zusammen, dieser ist wiederum für das Halten und Bereitstellen der einzelnen Sensoren für die anderen Komponenten zuständig.

Der *Deep Learning Service* repräsentiert das Herzstück der Anwendung. Der Service abstrahiert das KNN, sorgt dafür, dass die simulierten und originalen Sensorwerte in Form von Eingangsneuronen in das Netzwerk einfließen und das Ergebnis über die Ausgangsneuronen zur Verfügung steht.

Der *Embedded REST Server* dient als Zugangspunkt für die Aktoren, die über dessen Endpoints die gewünschten Informationen abrufen können. Der *Actuator Manager Service* enthält alle an der Anwendung angemeldeten Aktoren.

Eine lauffähige Version der Anwendung *Smarter Home* mit dem Dateinamen „*smarterhome-1.0.3.jar*“ sowie eine Konfigurationsdatei „*smarterhome.config*“ befinden sich auf der mitgelieferten CD-ROM in dem Verzeichnis „*Programme/Smarter Home/Application*“. Der dazugehörige Quellcode befindet sich auf der CD-ROM in dem Verzeichnis „*Quelltexte/Smarter Home/Source*“.

Abbildung 7.28: Die Komponenten der Anwendung *Smarter Home* in der Gesamtübersicht.

Kapitel 8

Umsetzung Lernbaukasten

In dem Kapitel 6 wurden die einzelnen Lösungskonzepte für die Entwicklung eines Lernbaukastens für Schüler der SEK 2 präsentiert.

In diesem Kapitel folgt nun die Beschreibung der einzelnen Umsetzungsdetails, indem auf die wichtigsten Elemente eingegangen wird, die für das Verständnis nötig sind.

8.1 Lernbaustein Theorie

Der Lernbaustein *Theorie* wird vor Ort als 90-minütige Vorlesung bzw. Vortrag von einem Mitarbeiter des Schülelabors *RoboticLab* in Zusammenarbeit mit den Schülern durchgeführt. Der Vortrag ist nach den Lehrsritten MOMBIs aufgebaut und bedient dabei die drei Elemente *Provozieren*, *Aktivieren* und *Informieren*. Nachfolgend werden die einzelnen umgesetzten Schritte näher beleuchtet. Zuvor wird in den Abschnitten 8.1.1 und 8.1.2 kurz auf die interaktive Mitarbeit seitens der Schüler und die begleitenden Arbeitsblätter eingegangen.

8.1.1 Interaktive Mitarbeit

Damit die Schüler während des Vortrags nicht nur passiv das Geschehene und Gelehrte verfolgen, sind Elemente in den Vortrag eingebunden, bei denen die aktive Mitarbeit der Schüler erforderlich wird.

Dafür gibt es drei Kategorien der Mitarbeit, zu der die Schüler während des Vortrags durch entsprechende Symbole aufgefordert werden. Diese werden am Anfang des Vortrages den Schülern vorgestellt und erläutert.

Das erste Symbol (Abbildung 8.1 links) erfordert eine aktive Mitarbeit und Zuarbeit der Schüler, diese Art der Mitarbeit ist bei der Diskussion und dem Brainstorming erforderlich.

Das zweite Symbol (Abbildung 8.1 mittig) signalisiert den Schülern, dass eine Aufgabe in Eigenarbeit auf dem Arbeitsblatt bearbeitet werden muss, z.B. das Erstellen einer Oder-Schaltung mit Hilfe eines künstlichen Neurons.

Neben der Abarbeitung von Aufgaben müssen zusätzlich Informationen auf dem Arbeitsblatt ergänzt werden. Dazu gehören bspw. die einzelnen Bestandteile eines künstlichen Neurons oder auch verschiedene Definitionen bezüglich der Thematik. Das Symbol für das Ergänzen von Informationen auf dem Arbeitsblatt ist in Abbildung 8.1 rechts dargestellt.

Abbildung 8.1: Zu den Aufgaben der Schüler gehört die aktive Mitarbeit, das Lösen von Aufgaben und das Ergänzen von Informationen auf dem Arbeitsblatt.

8.1.2 Begleitende Arbeitsblätter

Die Arbeitsblätter, die vor Beginn der Vorlesung den Schülern ausgehändigt werden, sollen einerseits begleitend durch die Vorlesung führen und andererseits enthalten sie Aufgaben und Lücken, die von den Schülern bearbeitet bzw. ausgefüllt werden müssen. Die Arbeitsblätter sind grundsätzlich wie der Vortrag bzw. die Präsentation aufgebaut. Sie enthalten den gleichen Aufbau und Inhalt, sodass eine paralleles Abarbeiten der Aufgaben möglich ist. Die verwendete Symbolik (siehe Abschnitt 8.1.1), die in dem Vortrag auf eine entsprechende Mitarbeit seitens der Schüler verweist, ist äquivalent dazu auf den Arbeitsblättern an den entsprechenden Stellen zu finden, wie dargestellt in Abbildung 8.2.

Im ausgefüllten Zustand dienen die Arbeitsblätter den Schülern als Zusammenfassung des Vortrages. Die Arbeitsblätter befinden sich als Datei mit dem Namen „*Theorie-Arbeitsblatt.pdf*“ auf der CD-ROM in dem Verzeichnis „*Lernbaukasten/Theorie*“

Abbildung 8.2: Entsprechende Symboliken auf den Arbeitsblättern signalisieren den Schülern, welche Art von Mitarbeit von Ihnen für die jeweilige Aufgabe gefordert ist.

8.1.3 Provozieren

Die erste inhaltliche Thematik im Vortrag erfolgt in Form einer Fragestellung, mit der Intention eine Diskussion über die Thematik KI mit den Schülern zu führen.

Watson, wir haben ein Problem.

Kann die Künstliche Intelligenz den Menschen ersetzen?

Dafür wurde eine Fragestellung gewählt, die vielleicht auf den ersten Blick nicht gerade sehr originell wirkt, aber dafür andere Vorteile mit sich bringt. Denn das Ziel der Fragestellung ist nach MOMBI, die Schüler durch die Fragestellung zu provozieren. Hierfür muss die Fragestellung für jeden Einzelnen relevant sein und entsprechend eine Meinung dazu vorhanden sein, um einen Anreiz zu erhalten an der Diskussion teilzunehmen.

Die Fragestellung ist so gewählt, dass sie die Zielgruppe möglichst polarisiert, um dadurch verschiedene Meinungsäußerungen zu ermöglichen mit einer entsprechend lebhaften Diskussion im Ergebnis. Die Fragestellung ermöglicht es darüber hinaus, dass die Schüler, unabhängig von ihren fachlichen Vorkenntnissen zu dem Thema KI, an der Diskussion teilnehmen können.

Neben der polarisierenden Eigenschaft besitzt die Fragestellung zusätzlich einen persönlichen Charakter, d.h. die Schüler sind von dem Ausgang der Thematik direkt betroffen. Niemand möchte, sei es nun im Berufsleben oder in einem anderen Bereich, ersetzt werden, sodass dadurch zusätzlicher „Zündstoff“ in die Diskussion mit einfließt. Durch diese Eigenschaft der Fragestellung wird außerdem eine Schnittstelle zu der Thematik *Informatik, Mensch und Gesellschaft* geschaffen, die im Rahmenplan der SEK 2 für Berliner und Brandenburger Schulen vorgesehen ist.

Um einen Einstiegspunkt in die Diskussion zu finden, soll nicht nur die Fragestellung an die Schüler weitergegeben werden, sondern eine Hilfestellung angeboten werden. Durch die Beschreibung eines Berufs, der bereits durch KI ersetzt wurde, soll eine mögliche Gefahr aufgezeigt und eine Diskussion eingeleitet werden.

Sollte dies nicht den gewünschten Erfolg zur Folge haben, wird eine weitere Hilfestellung gegeben, indem einzelne Schüler direkt nach ihren Berufswünschen befragt werden und mit ihrer Hilfe mögliche Übernahmeszenarien durch die KI skizziert werden.

8.1.4 Aktivieren

Im Anschluss an die Diskussion sollen die Schüler auf dem Stand sein, dass sie sich gedanklich mit der Thematik KI schon ein wenig auseinandergesetzt haben, sodass nun der Part zur Aktivierung des Vorwissens im Anschluss folgen kann.

Dieses Aktivieren erfolgt in Form eines sogenannten Brainstormings, mit dem Hintergrund, im Anschluss eine geeignete Definition der KI gemeinsam definieren zu können.

Dafür werden den Schülern einzelne Kategorien aus der KI vorgeschlagen, in welche die Beiträge eingeordnet werden können. Diese Vorschläge sollen die Schüler nicht einschränken, sondern bieten auch die Möglichkeit, um weitere Kategorien ergänzt oder auch abgeändert zu werden. Hierbei wird die Richtung des Brainstormings vollständig den Schülern überlassen, um so keine Ideen zu unterdrücken, sondern der Kreativität freien Lauf zu lassen.

Dabei fungiert der Vortrag haltende Mitarbeiter als Moderator und achtet darauf, dass die folgenden Regeln des Brainstormings eingehalten werden:

- Jeder Vorschlag, jeder Gedanke und jede Idee sind erwünscht.
- Die Ideen sollten schnell vorgebracht werden, um die anderen Schüler zu motivieren.
- Vorhandene Ideen sollen nicht im Kopf abgewägt werden, die Teilnehmer sprechen sie einfach aus.
- Quantität geht vor Qualität. Die Anzahl der Ideen sind entscheidend.
- Bereits vorgebrachte Ideen dürfen aufgegriffen und weiterentwickelt werden.
- Kritik an Ideen in jeglicher Form ist nicht erlaubt. [67]

Die Regeln des Brainstormings sind auf dem Arbeitsblatt für die Schüler ebenfalls kurz erklärt.

Im Anschluss an das Brainstorming wird gemeinsam mit Hilfe der gesammelten Ideen zum Thema KI eine passende Definition erarbeitet, die die Schüler an der entsprechenden Stelle auf ihrem Arbeitsblatt notieren.

Hierbei wird seitens des Vortrag haltenden Mitarbeiters darauf geachtet, dass die entwickelte Definition im Einklang mit den Definitionen der KI aus literarischen Quellen steht.

Anschließend sind die Schüler ausreichend darauf vorbereitet, um genauer über das Thema Künstliche Intelligenz *informiert* zu werden.

8.1.5 Informieren

Im letzten Abschnitt erfolgt die eigentliche Wissenserarbeitung zu den Grundlagen und Arbeitsweisen der KI sowie einer Vorbereitung auf den praktischen Abschnitt in der Hochschule. Das Element *Informieren* setzt sich dabei aus zwei Teilelementen zusammen: dem Vortrag und zusätzlich nachfolgenden Veranstaltungen durch den unterrichtenden Lehrer.

Im Vortrag in der Schule wird zunächst der geschichtliche Werdegang der KI aufgegriffen, beginnend mit dem wohl bekanntesten Test für das Prüfen von KI: dem Turing-Test.

Dabei soll den Schülern aufgezeigt werden, in welcher bedeutenden Epoche der KI wir uns momentan befinden, denn erst vor zwei Jahren wurde erstmalig der Turing-Test durch den Chatbot *Eugene Goostman* bestanden. Bis dato galt der Turing-Test 64 Jahre lang als unbezwingbar für Maschinen.

Im Anschluss daran wird auf den geschichtlichen Werdegang der KI eingegangen, bei dem zusätzlich ein Überblick über die Vielzahl an Werkzeugen der KI gegeben wird. Hier werden auch wichtige geschichtliche Fakten erläutert, wie z.B. warum die KNN seit 2005 einen solch großen Aufschwung in Form von *Deep Learning*-Netzwerken erleben.

Nach dem geschichtlichen Überblick wird der Fokus auf die KNN gerichtet.

Hierbei werden schrittweise die einzelnen Elemente eines KNN vorgestellt, angefangen mit einem einzelnen Neuron oder dem Entwicklungszyklus eines KNN, bis hin zu den unterschiedlichen Architekturen, die es dafür gibt.

Die konkrete und detaillierte Erarbeitung erfolgt nicht in diesem Vortrag, da dies den zeitlichen Rahmen von 90 Minuten sprengen würde. Die Erarbeitung erfolgt durch den unterrichtenden Lehrer in Form eines Seminarkurses o.Ä. zwischen dem Vortrag und dem Praxis-Abschnitt in der Technischen Hochschule Wildau. So hat der Lehrer die Möglichkeit, den Fokus und die Richtung in dieser Thematik mitzubestimmen, dies ist gerade für eine spätere Notenvergabe wichtig. Zwingend notwendig ist hierbei eine Abstimmung zwischen Mitarbeiter des Schülerlabors und Lehrer, damit dieser intensive Abschnitt nicht konträr zu dem praktischen Element in der Hochschule erfolgt.

Im letzten großen Abschnitt der Vorlesung erfolgt die Vorstellung des Szenarios und der Anwendung *Smarter Home*.

Dafür wird zunächst Karin und ihr soziales Umfeld den Schülern vorgestellt. Anschließend wird das Verhalten von Karin beschrieben, das sich in jüngster Vergangenheit immer mehr verändert hat und auf eine beginnende Demenz deutet.

Darauf aufbauend wird erläutert, welche geeigneten Maßnahmen in den eigenen vier Wänden (eine Auswahl) bei Einsetzen der Krankheit Demenz helfen können und welche Rolle dabei die Anwendung *Smarter Home* einnehmen kann und soll.

Abschließend wird die Anwendung und der zu verwendende Prototyp und Sensorik für den praktischen Abschnitt vorgestellt.

Die Präsentation zum Vortrag mit dem Dateinamen „*Theorie-Präsentation.pdf*“ befindet sich auf der CD-ROM in dem Verzeichnis „*Lernbaukasten/Theorie*“.

8.2 Lernbaustein Praxis

Der Praxis-Lernbaustein findet in einem Telematik-Labor der Technischen Hochschule Wildau statt. Dabei haben die Schüler die Aufgabe, das in Abschnitt 6.3.2 beschriebene Szenario mit der älteren Dame Karin mit Hilfe eines Prototyps, geeigneten Sensoren und der Anwendung *Smarter Home* in die Praxis umzusetzen. Am Ende sollen die Schüler dazu in der Lage sein, den Schlafrhythmus der älteren Dame zu überwachen und etwaige Unregelmäßigkeiten im Schlafverhalten über die IVE *Smarter Home* festzustellen.

Im Folgenden wird zunächst auf die entwickelten Komponenten dieses Lernbausteins eingegangen, dazu gehören bspw. der Prototyp aus Lego, der das Schlafzimmer repräsentiert, oder die eingesetzte Sensorik und deren Programmierung. Abschließend wird das erstellte Tutorial vorgestellt, das die Aufgabe hat, die Schüler durch die Praxisphase zu führen.

8.2.1 Prototyp

Der Prototyp für die Praxisphase stellt das Schlafzimmer von Karin dar. Dieses wurde mit Hilfe von Legobausteinen umgesetzt und verfügt über ein Bett, einen Kleiderschrank, einen Nachtschrank, eine Sitzgelegenheit, zwei Fenster, die sich öffnen lassen, sowie eine Tür. Das Bett ist so konzipiert, dass der verwendete RFID-Empfänger ohne Probleme unter die Liegefläche eingebaut werden kann. In der Wand des Raumes befindet sich eine Auslassung, durch die der RFID-Empfänger mit dem *Arduino* verkabelt werden kann. Die zusätzlichen Komponenten im Schlafzimmer, wie die Fenster oder der Kleiderschrank, bieten Potential für das zukünftige Verbauen von weiterer Sensorik oder Ähnlichem. In Abbildung 8.3 ist der fertige Prototyp für das Schlafzimmer von Karin dargestellt.

Abbildung 8.3: Der Prototyp aus Lego bietet die Möglichkeit, verschiedene Sensoren zu verbauen. In dem Bett angebracht, befindet sich der RFID-Empfänger.

Das Schlafzimmer selbst ist auf einer sogenannten Grundplatte von Lego angebracht, zusätzlich befindet sich darauf eine Halterung für den *Arduino UNO*, wodurch ein einfaches Fixieren des Zimmers und des Mikrocontrollers möglich wird.

Karin wird in dem Szenario durch eine Legofigur repräsentiert und ist mit dem RFID-Sender in Schlüsselanhängerform ausgestattet. Durch die Anbringung eines entsprechenden Schlüsselrings am Sender, kann dieser bequem am Hals von Karin fixiert werden. Abbildung 8.4 zeigt Karin mit angebrachtem RFID-Sender.

Abbildung 8.4: Über einen Schlüsselring und einen zusätzlichen Draht ist der RFID-Schlüsselanhänger an Karin befestigt.

Sobald Karin in das Bett gelegt wird, wird dies durch den Empfänger erfolgreich erkannt. Befindet sich Karin hingegen lediglich im Raum oder auch außerhalb des Raumes, so erhält der Empfänger kein Signal von Seiten des RFID-Senders.

8.2.2 Sensorik

Damit die Schüler in der Praxisphase die Anwendung *Smarter Home* mit den RFID-Sensoren nutzen können, müssen sie diese mit dem Framework über die bereitgestellte Schnittstelle bekannt machen und anschließend kommunizieren lassen.

Hierfür wurde eine Library für die *Arduino*-Plattform entwickelt, mit denen die Schüler durch bereitgestellte Funktionen die Möglichkeit haben, komfortabel die Anmeldung des Sensors selbst und das Übertragen der Sensordaten durchzuführen.

Die Library für die Sensorik setzt sich aus der Header-Datei „*Sensor.h*“ sowie der Implementierungsdatei „*Sensor.cpp*“ zusammen.

Durch die Einbindung der Bibliothek in ein *Arduino*-Programm können mehrere Sensor-Objekte erstellt und über einen Mikrocontroller administriert werden. Somit ist es auch möglich, unterschiedliche Sensoren über eine einzelne Mikrocontrollereinheit anzusteuern und deren Sensordaten separiert an die IVE zu verschicken.

Bei der Erstellung eines Sensorobjektes werden über den Konstruktor die Sensoreigenschaften übergeben, dazu gehören der serielle Outputstream des *Arduinos*, die ID des Sensors und um was für einen Sensortypen es sich handelt ('B' = Binary, 'T' = Temperature, 'H' = Humidity).

Für die Kommunikation mit der IVE bietet die Bibliothek die beiden Methoden `bonjour()` und `talk()` an. Erstere meldet die Sensoreinheit am *Smarter Home*-System (IVE) an. An die Methode `talk()` kann ein Sensorwert übergeben werden, der an die IVE nach erfolgreicher Anmeldung übertragen wird.

Für die eigentliche Programmierung im Schülerlabor steht den Schülern das Programm „*HelloRFID.ino*“ zur Verfügung. Dieses enthält ein HelloWorld-Programm, in der das einfache Auslesen eines RFID-Sensors demonstriert wird. Dabei zeigt das Programm über den seriellen Monitor an, ob der verbundene RFID-Empfänger aktuell einen RFID-Sensor in seiner Nähe erkannt hat oder nicht.

Die serielle Kommunikation zwischen *Arduino* und der Anwendung *Smarter Home* ist jedoch noch nicht enthalten und soll von den Schülern mit Hilfe der Bibliothek implementiert werden.

Dafür haben sie zunächst die Aufgabe den Programmcode nachzuvollziehen und dann entsprechend zu erweitern. Das fertige Programm soll dann dazu in der Lage sein, dass der *Arduino* den Sensorwert '1' an die IVE schickt, wann immer vom Empfänger ein RFID-Sensor erkannt wurde, und den Wert '0', wenn kein Sensor detektiert wurde. Als Hilfestellung sind hierfür entsprechend im Programmcode die zu bearbeitenden Passagen durch ein `TODO` gekennzeichnet, gefolgt von einer zusätzlichen Beschreibung, was an dieser Stelle einzufügen ist.

Für die Erstellung der Datei „*HelloRFID.ino*“ wurde die von der Arduino IDE mitgelieferte Datei „*Addicore_RFID_Example.ino*“ als Grundlage verwendet und entsprechend abgeändert. Neben dem Programm „*HelloRFID.ino*“ wird im Zuge dieser Arbeit ebenfalls die Datei „*HelloRFID_complete.ino*“ ausgeliefert, die das komplette Programm zur Übertragung der RFID-Sensorwerte an die Anwendung *Smarter Home* enthält.

Die beiden Programme „*HelloRFID.ino*“ und „*HelloRFID_complete.ino*“ befinden sich auf der mitgelieferten CD-ROM in dem Verzeichnis „*Lernbaukasten/Praxis/Arduino/src*“.

Die Sensor-Bibliothek mit dem Namen „*sensorkit*“ befindet sich in dem Verzeichnis „*Lernbaukasten/Praxis/Arduino/libraries*“.

8.2.3 Tutorial

Für den praktischen Abschnitt in der Hochschule wurde ein Tutorial entworfen, das die einzelnen zu erledigenden Aufgaben für die Praxisphase beschreibt und von den Schülern durchgearbeitet werden soll.

Bevor die Schüler das Tutorial durcharbeiten, wird ihnen jedoch zunächst noch einmal die Zielstellung für diesen Lernbaustein erklärt und eine kurze Einführung in die zu bearbeitenden Aufgaben und die einzelnen Komponenten gegeben, mit denen sie es in der Praxisphase zu tun bekommen. Dabei werden in der Anwendung *Smarter Home* bspw. die einzelnen Funktionen der Anwendung durch einen Mitarbeiter live demonstriert und erklärt. Dadurch soll den Schülern ein schnellerer und besserer Einstieg in das Tutorial ermöglicht werden.

Das Tutorial selbst enthält 6 Arbeitsschritte, die aufeinander aufbauen und im Ergebnis das lauffähige Gesamtsystem enthalten. Bei der Abarbeitung müssen sich die Schüler verschiedenen Herausforderungen stellen, wie beispielsweise das Verkabeln und Programmieren von Sensoren und einem Mikrocontroller, das geeignete Anbringen der Sensorik im Prototypen oder die Vernetzung zweier Komponenten, um so eine Kommunikation zwischen diesen zu ermöglichen.

Jeder Arbeitsschritt enthält dabei eine Liste der benötigten Komponenten mit einer anschließenden Kurzbeschreibung des Ziels und einer detaillierten Anleitung.

Die folgenden Arbeitsschritte enthält das Tutorial:

1. Verbinden des Arduino Uno R3 mit dem RFID-Kit
2. Auslesen des RFID-Empfängers
3. Einbau des RFID-Empfängers in den Prototypen
4. Arduino mit der Anwendung Smarter Home verbinden
5. Smarter Home für Karins Schlafrhythmus vorbereiten
6. Karins Schlafrhythmus antrainieren und überwachen

Zu manchen Arbeitsschritten werden zusätzlich Hilfsmittel den Schülern zur Verfügung gestellt. In Arbeitsschritt 6 bspw. erhalten die Schüler neben der Beschreibung und Anleitung eine vorgefertigte Excel-Arbeitsmappe, in der sie ihre Ergebnisse der Tests protokollieren können. Diese Protokollierung ist gerade für eine spätere Auswertung wichtig, das Protokoll soll dazu beitragen, dass die Schüler nicht vergessen, die wichtigsten Informationen zu einem Test zu dokumentieren.

Das in diesem Abschnitt beschriebene Tutorial mit dem Dateinamen „*Praxis-Tutorial.pdf*“ befindet sich auf der mitgelieferten CD-ROM in dem Verzeichnis „*Lernbaukasten/Praxis*“.

Die Arbeitsmappe mit dem Dateinamen „*Praxis-Protokoll-Training.xlsx*“ für die Protokollierung der Testergebnisse sowie eine Beispieldatei mit Trainingsdaten „*Praxis-Beispiel-RFID-Zeit-Trainingsdaten.csv*“ befinden sich ebenfalls in diesem Verzeichnis.

8.3 Lernbaustein Festigung

Der letzte Lernbaustein des Lernbaukastens wird wieder in der jeweiligen Schule in Eigenregie durch den unterrichtenden Lehrer durchgeführt. Darin sollen die Schüler einerseits ihre gesammelten Ergebnisse auswerten und vorstellen und zusätzlich über interessante Forschungsprojekte im Bereich der KI recherchieren, gesellschaftskritisch beleuchten und anschließend darüber berichten.

Als zusätzliches Angebot wird von Seiten des RoboticLabs den Schülern die Option gegeben, die Recherchemöglichkeiten über das Datenbank-Infosystem¹ (DBIS) der Bibliothek in der TH Wildau zu nutzen. Hier haben die Schüler freien Zugriff auf verschiedene Literaturdatenbanken, wie beispielsweise Statista, SpringerLink und IEEE Xplore. Diese Datenbanken sind nur über das Campusnetz kostenlos nutzbar und ermöglichen die Recherche und den Zugriff auf eine Vielzahl von aktuellen wissenschaftlichen Dokumenten.

Darüber hinaus wäre es zukünftig denkbar diesen Lernbaustein ebenfalls mit einem praktischen Abschnitt in der TH Wildau zu kombinieren. Dabei könnten die einzelnen Recherchemöglichkeiten der Bibliothek den Schülern erklärt werden, sodass diese die im Anschluss für die eigenen Recherchen nutzen können. Mehr dazu kann in Abschnitt 10.1.5 nachgelesen werden.

¹ Datenbank-Infosystem der TH Wildau:
<http://rzblx10.uni-regensburg.de/dbinfo/fachliste.php?lett=1>

Kapitel 9

Testphase

In diesem Kapitel werden mit Hilfe des erstellten Prototypen, der Anwendung *Smarter Home* sowie einer vorgefertigten CSV-Datei, die Trainingsdaten enthält, Testfälle durchgeführt und deren Ergebnisse beschrieben. Diese Tests und Ergebnisse sollen keinesfalls eine hinreichende Abdeckung der nötigen Tests für die Überwachung einer Person in ihren eigenen vier Wänden darstellen, vielmehr soll es als ein erster *Proof of Concept* dienen, indem ein erstes übersichtliches Szenario getestet wird.

In den nachfolgenden Tests wird das Szenario aus dem Lernbaukasten (Abschnitt 6.3.2) aufgegriffen, indem der Schlafrhythmus von Karin antrainiert und anschließend überwacht bzw. getestet wird. Dabei werden verschiedene Architekturen und Konfigurationen des KNN verwendet, um dabei Aussagen über die Eignung dieser treffen zu können.

Im letzten Abschnitt erfolgt eine Zusammenfassung der Testergebnisse mit ersten Ideen für weitere Testszenarien.

9.1 Schlafrhythmus in Abhängigkeit zur Zeit

In diesem Abschnitt wird der Schlafrhythmus von Karin in Abhängigkeit zur Zeit getestet, d.h. es wird ein KNN verwendet, das insgesamt über zwei Eingangsneuronen und einem Ausgangsneuron verfügt. Die Eingangsneuronen setzen sich zusammen aus dem binären RFID-Sensor und einem Eingangsneuron für die Uhrzeit. Das Ausgangsneuron repräsentiert in diesem Szenario den Zustand bezüglich der Tätigkeit *Schlafen*.

9.1.1 Trainingsdaten

Für das Training selbst wird in den nachfolgenden Abschnitten immer die gleiche unveränderte CSV-Datei mit Trainingsdaten verwendet. Diese Trainingsdaten dokumentieren ein Schlafverhalten von insgesamt 8 Tagen. In Abbildung 9.1 und 9.2 sind diese 8 Tage in Form eines Diagramms dargestellt. Abbildung 9.1 beinhaltet die positiven Trainingsdaten für die ersten vier Tage, diese Daten beschreiben also das normale alltägliche Schlafverhalten von Karin. Abbildung 9.2 enthält die Datensätze für vier weitere Tage, diese repräsentieren jedoch nicht normale Verhaltensweisen hinsichtlich des Schlafrhythmus von Karin, sondern das gegenteilige Verhalten.

Bewusst wurden bei den Aufsteh- und Zubettgehzeiten leichte Diskrepanzen zwischen den Tagen eingebaut, um so auch das natürliche nicht immer konstante Verhalten im Alltag widerzuspiegeln und um zusätzlich zu testen, wie das KNN darauf reagiert.

So schwankt die Aufstehzeit von Karin zwischen 8:00 und 9:00 Uhr und die Zubettgehzeit zwischen 21:00 und 23:00 Uhr.

Abbildung 9.1: Die positiven Trainingsdaten enthalten vier Tage, bei denen es leichte Diskrepanzen zwischen den Aufsteh- und Zubettgehzeiten von Karin gibt.

Bei den Werten für das Eingangsneuron, das für die Zeitangabe dient, wurden immer volle Stunden verwendet, wie bspw. 17:00 Uhr, 18:00 Uhr etc. Nicht eingesetzt wurden hingegen Zeiten wie 8:30 Uhr oder 0:15 Uhr. Diese werden jedoch bei den anschließenden Tests abgefragt, damit soll überprüft werden, ob das Netz nicht nur einfach die Trainingsdaten „auswendig lernt“ (Überanpassung), sondern ob es auch bei unbekannten Datensätzen richtig reagiert.

Die detaillierten Trainingsdaten können in der Datei „*RFID-Zeit-Trainingsdaten.csv*“ nachgelesen werden, die sich in dem Verzeichnis „*Dokumente/Testphase/RFID-Zeit*“ auf der CD-ROM (siehe Anhang B) befindet.

Abbildung 9.2: Die negativen Trainingsdaten enthalten das gegenteilige Verhalten im Bezug auf das Schlafverhalten von Karin.

9.1.2 Multilayer Perzeptron 1x4

Für das erste KNN wird eine Architektur mit einer verdeckten neuronalen Schicht verwendet. Diese verdeckte Schicht enthält vier Neuronen und jedes Neuron ist mit dem Ausgangsneuron verbunden. Es handelt sich um ein vorwärts gerichtetes KNN, das heißt die Verbindungen sind gerichtet, die Richtung der gewichteten Verbindungen ist in Abbildung 9.3 für die erste durchgehende Verbindung beispielhaft durch zwei Pfeile symbolisiert. Die Bias-Neuronen sind zur besseren Übersicht in dieser Grafik nicht abgebildet.

Abbildung 9.3: Das Multilayer Perzeptron im ersten Testversuch ist vorwärts gerichtet und enthält eine verdeckte Schicht mit 4 Neuronen.

Als Aktivierungsfunktion wird für jedes Neuron die Sigmoidfunktion verwendet, dabei handelt es sich um eine S-förmige stetige Kurvenfunktion, die die Eigenschaft der Differenzierbarkeit aufweist.

Diese Aktivierungsfunktion gilt gerade für Netzwerke, mit denen etwas angelernet werden soll, als geeignet, da durch die Differenzierbarkeit eine Optimierung der Gewichte der einzelnen Neuronen möglich wird. Mehr dazu kann unter [72] nachgelesen werden. In Abbildung 9.4

ist ein Beispiel für den Kurvenverlauf einer sigmoiden Funktion dargestellt.

Abbildung 9.4: Die sigmoide Funktion wird als Aktivierungsfunktion in den Tests verwendet. [30]

Ein Wert gegen '0' signalisiert bei einem Ausgangsneuron, dass die aktuellen Sensorwerte einen Zustand beschreiben, der als keine normale Situation antrainiert wurde. Ein Wert gegen '1' gibt hingegen an, dass das KNN keine Unregelmäßigkeit oder ungewöhnlichen Zustand erkannt hat.

In Tabelle 9.1 ist die Konfiguration des KNN für den Testdurchlauf noch einmal zusammengefasst. Die Fehlerrate bei den einzelnen Testdurchführungen wurde bewusst sehr niedrig gehalten, um damit die jeweilig eingestellten Iterationen auch erreichen zu können. Insgesamt wurden drei Versuchsdurchläufe mit unterschiedlicher Anzahl an Trainingsiterationen durchgeführt und dokumentiert. Das erste Training erfolgte mit einer Anzahl von 100 Iterationen, das zweite mit 1.000 und das letzte mit 10.000 Iterationen. Alle drei Ergebnisse können in der Datei „RFID-Zeit-Versuchsdaten.xlsx“ eingesehen werden, in den nachfolgenden Ausführungen wird jedoch lediglich auf das beste der drei Trainingsresultate eingegangen.

Eigenschaft	Ausführung
Verdeckte Schichten	1 Schicht á 4 Neuronen
Bias	aktiviert
Aktivierungsfunktion	Sigmoid
Fehlerrate	0.0001
Iterationen	100, 1000 und 10000

Tabelle 9.1: Die Konfiguration des Netzwerkes für den ersten Testdurchlauf in der Übersicht.

In den ersten drei Testdurchläufen konnte das beste Ergebnis mit dem KNN erzielt werden, das insgesamt 1000 Trainingswiederholungen durchlaufen hat. Dabei wurde am Ende des Trainings eine Fehlerrate von 0,059795 erzielt. Die Fehlerquote ist das entscheidende Qualitätsmerkmal für das Training eines KNN. Bei der Durchführung des Trainings wurde nach jeder Iteration die aktuelle Fehlerrate ermittelt und im Verlauf dokumentiert und ist in Abbildung 9.5 in Form eines Diagramms dargestellt.

Abbildung 9.5: Der Verlauf der Fehlerrate bei 1000 Trainingswiederholungen.

Gut erkennbar im Kurvenverlauf ist, dass die Fehlerrate mit zunehmenden Trainingswiederholungen nicht zwingend absinkt. Immer wieder sind zwischendurch Ausreißer zu erkennen, bei denen die Fehlerrate plötzlich stark nach oben springt, wie bspw. in Abbildung 9.5 bei den Iterationen 170 bis 180 und 900 bis 910.

Die Ergebnisse des anschließenden Tests sind in Tabelle 9.2 einzusehen. Hierfür wurden in der Anwendung *Smarter Home* im *Go Live*-Modus die Eingangsneuronen entsprechend so verändert, wie in der Tabelle in den Spalten *Uhrzeit* und *RFID-Sensor* dargestellt und der daraus berechnete Ausgangswert dokumentiert. Dieser ist in der jeweiligen Zeile in der Spalte *Eingetroffenes Ergebnis* hinterlegt.

Immer wenn das erwartete Ergebnis eine '0' ist, dann wurde mit Hilfe der Eingangsneuronen eine nicht normale Situation simuliert, die von dem KNN erkannt werden soll. Dies ist bei den Versuchsnummern (VNR) 1, 2, 14 und 15 der Fall.

Grundsätzlich kann zu diesem Trainingsdurchlauf gesagt werden, dass die eingetroffenen Ergebnisse in keiner großen Diskrepanz zu den erwarteten Ergebnissen stehen und bereits ordentliche Resultate liefern.

Gerade bei den eindeutigen Tages- oder Nachtzeiten, bei denen in den Trainingsdatensätzen einheitliche widerspruchsfreie Werte benutzt wurden, konnten sehr gute Ergebnisse erzielt werden.

In den VNR 25, 26 und 27 konnte gezeigt werden, dass das KNN nicht einfach die trainierten Datensätze auswendig gelernt hat, sondern auch dazu in der Lage ist, auf vorher unbekannte Konstellationen hinsichtlich der Sensorwerte, richtig zu reagieren.

Für die Aufsteh- und Zubettgehzeiten wurden bei den Trainingsdatensätzen unterschiedliche Werte bei den positiven und negativen Daten verwendet. Dies wird auch bei den Messergebnissen sichtbar. Die Aufstehzeit von Karin ist zwischen 8:00 und 9:00 Uhr, diese werden in VNR 8 und 9 abgefragt. Die Zubettgehzeit von Karin schwankt zwischen 21:00 und 23:00 Uhr, repräsentiert in VNR 21, 22 und 23. Gerade die vier Messreihen 8, 9, 22 und 23 haben mit Abweichungen von 0,1219 bis hin zu 0,6141 die vergleichsweise größte Diskrepanz zwischen erwartetem und eingetroffenem Ergebnis.

Versuchsnr.	Uhrzeit	RFID-Sensor	Erwartetes Ergebnis	Eingetroffenes Ergebnis
1	1	0	0	0,000000832
2	2	0	0	0,000000027
3	3	1	1	1
4	4	1	1	1
5	5	1	1	1
6	6	1	1	1
7	7	1	1	1
8	8	0	1	0,3858495
9	9	0	1	0,7017181
10	10	0	1	0,85535824
11	11	0	1	0,9189036
12	12	0	1	0,9475529
13	13	0	1	0,962119
14	14	1	0	0,00000028
15	15	1	0	0,00000108
16	16	0	1	0,9785594
17	17	0	1	0,9807461
18	18	0	1	0,9822746
19	19	0	1	0,9833702
20	20	0	1	0,9841705
21	21	0	1	0,98476356
22	22	1	1	0,64867026
23	23	1	1	0,8780591
24	0	1	1	0,9999974
25	8,5	0	1	0,5633403
26	12,5	0	1	0,95595074
27	0,25	1	1	0,9999364

Tabelle 9.2: Die Versuchsergebnisse des ersten Tests in Form eines Soll-Ist-Vergleichs gegenübergestellt.

Hier wird der Vorteil der Verwendung einer sigmoiden Aktivierungsfunktion bspw. gegenüber einer binären Schwellenwertfunktion klar, denn diese sorgt dafür, dass die Ausgangswerte zwischen „0“ und „1“ fließend sind. Aufgrund dessen wäre es möglich, eine Art Schwellwert auf Seiten des Aktors, der die Daten aus der Anwendung abrufen, festzulegen, sodass erst ab einem gewissen Grenzwert die Situation als kritisch beurteilt wird. Dadurch entsteht ein gewisser Freiraum bei der Bewertung der Situation und es können Phasen, die leichte Unregelmäßigkeiten enthalten, wie die Aufsteh- und Zubettgehzeit von Karin, abgefangen werden.

In Abbildung 9.6 sind die Verlaufskurven der Soll- und Ist-Ausgangswerte der Versuchsreihe aus Tabelle 9.2 gegenübergestellt. Exemplarisch ist auch ein möglicher Schwellwert (orange Kurve) im Diagramm enthalten, der es in diesem Beispiel ermöglicht, zwischen leichten und groben Unregelmäßigkeiten zu unterscheiden.

Abbildung 9.6: Durch den Einbau eines Schwellwertes auf Seiten des Aktors, können leichte Abweichungen im Schlafverhalten abgefangen werden.

9.1.3 Multilayer Perzeptron 2x8

Der zweite Testversuch wurde mit einem KNN durchgeführt, das über zwei verdeckte neuronale Schichten verfügt. Die Schichten bestehen jeweils aus 8 Neuronen und jedes Neuron ist mit den Neuronen der nachfolgenden Schicht voll vermascht. Das KNN ist vorwärts gerichtet, d.h. die Verbindungen zwischen den Neuronen verlaufen ebenfalls gerichtet von einer niedrigen zur nächsthöheren Schicht.

In Abbildung 9.7 ist dieser Aufbau dargestellt. Die gerichteten Verbindungen verlaufen darin von links nach rechts, wie bei der ersten durchgehenden Verbindung vom Eingangs- zum Ausgangsneuron beispielhaft in der Grafik veranschaulicht. Die jeweiligen Bias-Neuronen der einzelnen Schichten sind zur besseren Übersicht nicht in der Grafik hinterlegt.

Abbildung 9.7: Das Multilayer Perzeptron im zweiten Testversuch ist ebenfalls vorwärts gerichtet und enthält zwei verdeckte Schichten mit jeweils 8 Neuronen.

In Tabelle 9.3 ist das KNN und dessen Konfiguration noch einmal zusammengefasst.

Eigenschaft	Ausführung
Verdeckte Schichten	2 Schichten á 8 Neuronen
Bias	aktiviert
Aktivierungsfunktion	Sigmoid
Fehlerrate	0.0001
Durchläufe	100, 1000 und 10000

Tabelle 9.3: Die Konfiguration des Netzwerkes für den Test in der Übersicht.

Auch in dieser Konstellation schnitt das KNN mit 1000 Trainingswiederholungen am besten ab, mit einer Fehlerrate von 0,03274.

Der Verlauf der Fehlerrate während des Trainings ist in Abbildung 9.8 einzusehen. Auch hier sind sogenannte Ausreißer bezüglich der Fehlerrate während des Trainings zu erkennen.

Abbildung 9.8: Im Verlauf des Trainings sind immer wieder Ausreißer in der Fehlerrate zu erkennen.

Bei den anschließenden Tests im *Go Live*-Modus, die Ergebnisse daraus sind in Tabelle 9.4 hinterlegt, ist das trainierte Netz dazu in der Lage, den antrainierten Schlafrythmus gut zu überwachen. Auch die kritischen Situationen in Versuch 1, 2, 14 und 15 werden erkannt. Wie bei dem vorherigen Test ist auch in VNR 22 die größte Abweichung zwischen erwartetem und eingetroffenem Ergebnis zu verzeichnen. Allerdings kann dieses KNN besser als sein Vorgänger mit den Schwankungen bei den Aufsteh- und Zubettgehzeiten der Trainingsdaten umgehen, so sind lediglich um 8 Uhr (VNR 8) und um 22 Uhr (VNR 22) größere Diskrepanzen zu erkennen. Das KNN mit einer verdeckten Schicht aus Abschnitt 9.1.2 hatte auch bei den Uhrzeiten 9 Uhr und 23 Uhr leichte Diskrepanzen, hier ist dies jedoch nicht der Fall.

Versuchsnr.	Uhrzeit	RFID-Sensor	Erwartetes Ergebnis	Eingetroffenes Ergebnis
1	1	0	0	0,029491365
2	2	0	0	0,029491365
3	3	1	1	0,99999636
4	4	1	1	0,99999636
5	5	1	1	0,9999964
6	6	1	1	0,9999964
7	7	1	1	0,9999964
8	8	0	1	0,5020566
9	9	0	1	0,94514436
10	10	0	1	0,94664437
11	11	0	1	0,95567614
12	12	0	1	0,9556774
13	13	0	1	0,95568246
14	14	1	0	0,029491365
15	15	1	0	0,029491365
16	16	0	1	0,9559044
17	17	0	1	0,9559536
18	18	0	1	0,95588446
19	19	0	1	0,9557496
20	20	0	1	0,9556828
21	21	0	1	0,9556531
22	22	1	1	0,43184766
23	23	1	1	0,94568557
24	0	1	1	0,99966556
25	8,5	0	1	0,92692703
26	12,5	0	1	0,9556785
27	0,25	1	1	0,9996672

Tabelle 9.4: Die Versuchsergebnisse des zweiten Tests in Form eines Soll-Ist-Vergleichs gegenübergestellt.

Offensichtlich hat das KNN sich auch in diesem Testdurchlauf die Daten nicht einfach „gemerkt“, sondern reagiert auf die ungewohnten Konstellationen der Sensorwerte in den VNR 25, 26 und 27 in gleicher Weise wie bei bekannten Mustern.

Für dieses Netz könnte eine Schwelle bei dem Ausgangswert 0,25 festgelegt werden. Abbildung 9.9 zeigt noch einmal den Soll-Ist-Vergleich der Ergebnisse dieses Versuches inklusive dem möglichen Schwellwert.

Die gesamten Versuchsdaten und -ergebnisse zu dieser Testphase sind auf der CD-ROM in dem Verzeichnis „Dokumente/Testphase/RFID-Zeit“ in der Datei „RFID-Zeit-Versuchsdaten.xlsx“ protokolliert.

Abbildung 9.9: Die Festlegung eines Schwellwertes bei dem Ausgangswert von 0,25 macht es möglich, dass die leichten Unregelmäßigkeiten bei den Aufsteh- und Zubettgehzeiten von Karin abgefangen werden können.

9.1.4 Zusammenfassung

Im Zuge der durchgeführten Testphase konnten mit Hilfe der Anwendung *Smarter Home* erfolgreich zwei KNN erstellt und mit den angefertigten Trainingsdaten in einer CSV-Datei angelernet werden, um damit den Schlafrhythmus von Karin in Abhängigkeit von der Uhrzeit zu überwachen und auf grobe Unregelmäßigkeiten hin zu überprüfen. Durch die Festlegung von Schwellwerten war es in den Testfällen möglich, Toleranzen bei den Aufsteh- und Zubettgehzeiten von Karin einzufügen.

Im nächsten Schritt der Testphase sollte getestet werden, ob die Überwachung des Schlafrhythmus von Karin auch in Abhängigkeit der Zeit und dem Wochentag möglich ist.

Neben der erfolgreichen Konfiguration der KNN konnten zusätzlich erste Erkenntnisse über den Trainings- bzw. Lernprozess eines KNN an sich gewonnen werden. So hat eine hohe Anzahl an Trainingsiterationen nicht zwingend das beste Endergebnis zur Folge, da die Fehlerrate während des Trainings Schwankungen und Ausreißern unterliegen kann.

Als Folge dessen wäre es denkbar, während eines Trainings, durch ein Abgleichen nach jeder Iteration, den stets aktuell besten Trainingszustand abzuspeichern, um diesen anschließend verwenden zu können.

Im Moment wird das trainierte KNN nach der letzten Iteration verwendet, ungeachtet dessen, ob zwischendurch Zustände mit qualitativ besserer Fehlerrate vorgelegen haben. Dies war in der Testphase im Training mit 10.000 Trainingsiterationen bei dem KNN aus Abschnitt 9.1.2 der Fall, dabei lag die Fehlerrate zwischenzeitlich bei ca. 0,035 und am Ende bei nur noch bei 0,117¹.

Diese Option müsste angesichts der damit verbundenen Performanceeinbußen jedoch abgewägt werden.

Das Bedienkonzept und die Tauglichkeit der Anwendung *Smarter Home* wurde während der Testphase ebenfalls einer ersten Bewährungsprobe unterzogen. Dabei hat sich vor allem

¹ Einzusehen in der beigelegten CD-ROM (siehe Anhang B) in der Datei „RFID-Zeit-Versuchsdaten.xlsx“, auf dem Datenblatt „1x4 10000 Iterationen“.

das Einlesen und Hinzufügen von Trainingsdaten in Form einer externen CSV-Datei als sehr hilfreich und komfortabel erwiesen, gerade bei einer großen Anzahl an Trainingsdaten. Die Führung durch die einzelnen Anwendungsmodi über die Applikation *Smarter Home* hat sich während der Anwendungsphase ebenfalls als sehr dienlich und unterstützend erwiesen. Durch die Kontrollstrukturen der Benutzeroberfläche sowie das Deaktivieren der nicht benötigten Bedienelemente während eines Modus, konnten des Öfteren Fehler und ungewollte Änderungen auf Anwenderseite während eines Tests vermieden werden.

Kapitel 10

Abschlussbetrachtung

In diesem letzten abschließenden Kapitel soll ein Überblick über die Ergebnisse dieser Masterarbeit in Form eines Fazits gegeben werden. Zuvor wird in einem kurzen Ausblick über zukünftige Aufgabenfelder informiert.

10.1 Ausblick

Die Ergebnisse dieser Masterarbeit sollen als Grundlage für zukünftige Forschungsprojekte und -arbeiten dienen. Aufgrund dessen werden im folgenden Abschnitt Ideen und Handlungsempfehlungen für Weiterentwicklungen gegeben.

10.1.1 Weitere Konfigurationsmöglichkeiten des Netzwerkes

Im Fokus der nächsten Weiterentwicklungen der Anwendung *Smarter Home* sollte zunächst die Erweiterung der Konfigurationsmöglichkeiten des KNN stehen. Darunter fallen bspw. die Option zum Einbau von Rückkopplungen innerhalb eines Netzes, um so die Architektur flexibler gestalten zu können, oder das Verwenden von verschiedenen Algorithmen zum Erlernen der Verhaltensmuster anhand von Trainingsdaten.

Mit den vorgeschlagenen zusätzlichen Konfigurationsmöglichkeiten wäre es bspw. möglich, die Architektur eines sogenannten *Elman-Netzes*, auch als *Simple Recurrent Network* bezeichnet, zu erstellen und dieses zu testen. Das *Elman-Netz* ist nach seinem Erfinder Jeffrey L. Elman benannt und wird in der Literatur als eine Netzwerkarchitektur beschrieben, die sich besonders dafür eignet, Eingaben in zeitlicher Abhängigkeit zu verarbeiten. [17] Ein interessanter Lösungsansatz für die weiteren Entwicklungen.

Encog unterstützt neben dem Perzeptron und dem Multilayer-Perzeptron auch die *Elman*-Architektur, sodass auf Seiten des verwendeten Frameworks die nötigen Grundvoraussetzungen gegeben sind.

10.1.2 Zusätzliche Aufnahmefunktion für Trainingsdaten

Smarter Home bietet in der aktuellen Version die Möglichkeit, Trainingsdaten in Form von simulierten oder Echtzeitdaten zu erstellen oder die Trainingsdaten über eine CSV-Datei zu importieren. Gerade hinsichtlich des Aspekts der Weiterforschung und einem möglichen praktischen Einsatz der Anwendung in einer Wohnung o.Ä. wäre es sinnvoll, die Anwendung um eine sogenannte Aufnahmefunktion zu erweitern. Diese Funktion ermöglicht es, über einen gewünschten Zeitraum Sensordaten aufzunehmen und diese in Form von Trainingsdatensätzen der Anwendung hinzuzufügen, sodass diese für ein anschließendes Training genutzt werden können.

10.1.3 Ausweitung der Testfälle für die Anwendung *Smarter Home*

Wie bereits in Abschnitt 9.1.4 angedeutet, wird es nötig sein, zukünftig zusätzliche Tests mit der Anwendung *Smarter Home* durchzuführen.

Dabei sollte im ersten Schritt, neben der Überwachung des Schlafrhythmus in Abhängigkeit zur Zeit, als zusätzliche Komponente der Wochentag hinzugenommen werden.

Bei erfolgreichem Testausgang diesbezüglich könnte in weiteren aufbauenden Testszenarien die zu überwachenden Tätigkeiten ausgeweitet und untersucht werden, inwiefern sich diese für die Überwachung durch KNN eignen.

10.1.4 Hybride Lösung

Die im Rahmen dieser Masterarbeit entwickelte Anwendung *Smarter Home* verwendet aktuell ausschließlich KNN, um das Verhalten in einer Umgebung bzw. Wohnung anhand von Sensorik zu bewerten. Für den Einsatz in den entwickelten Lernbausteinen ist dies völlig ausreichend. Als Grundlage für zukünftige Forschungsprojekte wäre es jedoch wünschenswert, die Anwendung um weitere Werkzeuge der KI in Form einer hybriden Lösung zu erweitern. So könnte bspw. mit Hilfe der logischen Programmiersprache *Prolog* die Anwendung um Elemente eines Expertensystems erweitert werden. Dadurch wäre es komfortabel möglich, Regeln zu definieren und zu überwachen. Solch eine Regel könnte bspw. dafür Sorge tragen, dass die Heizung in einem Raum nicht eingeschaltet ist, sobald ein Fenster in dem gleichen Raum geöffnet ist.

Zusätzlich wurden mit dem sorgfältig ausgewählten Framework *Encog* die nötigen Voraussetzungen geschaffen, um neben dem Einsatz von KNN auch weitere Werkzeuge der KI nutzen zu können.

10.1.5 Erweiterung des Lernbausteins Festigung

Der Lernbaustein *Festigung* wird nach jetzigem Stand in Eigenregie durch den unterrichtenden Lehrer in der jeweiligen Schule durchgeführt. Neben der Auswertung der gesammelten Ergebnisse aus der Praxisphase, haben die Schüler zusätzlich die Aufgabe, nach interessanten Forschungsprojekten aus dem Bereich KI zu recherchieren. An diesem Punkt könnte zukünftig die Ausweitung des Lernbausteins ansetzen. *Benjamin Körner* hat sich in seiner Masterarbeit mit dem Titel „*Konzeption und Umsetzung eines IT-gestützten interaktiven Rundgangs in der Bibliothek der TH Wildau für SchülerInnen zur Schulung des bewussten Einsatzes angewandter Informations- und Medienkompetenz*“⁹ mit der Vermittlung von Medien-, Informations- und Recherchekompetenz für Schüler der SEK 1 und 2 sowie Studienanfänger beschäftigt. Dabei entstand unter anderem ein Konzept, zugeschnitten auf Schüler der SEK 2, bei dem die Schüler mit Hilfe einer Lernplattform mit dem Namen „*Chamilo*“

das selbstständige Arbeiten mit Katalogen und Fachdatenbanken trainieren können. Hier könnte zukünftig überlegt werden, wie eine Integration dieses Konzepts in den Lernbaustein aussehen kann.

10.2 Zusammenfassung und Fazit

Im Rahmen dieser Masterarbeit wurde die Anwendung *Smarter Home* entwickelt, mit der es möglich ist, die beiden technischen Bereiche *Smart Home* und *Servicerobotik* unter dem Einsatz von Werkzeugen der KI zu verbinden. Ziel dabei war es, mit Hilfe der Anwendung die Verhaltensabläufe einer Person in ihren eigenen vier Wänden zu erlernen, um sie anschließend überwachen zu können. In Abbildung 10.1 ist diese Anwendung dargestellt. Im Zuge dessen wurde darüber hinaus ein Lernbalkkasten für Schülergruppen der Sekundarstufe 2 entworfen und entwickelt, in dem die Funktionsweise der verwendeten KI-Werkzeuge theoretisch erklärt und praktisch angewandt werden.

Abbildung 10.1: Screenshot von der intelligenten Verarbeitungseinheit *Smarter Home*.

Vor der Konzeptionierung der Anwendung und des Lernbalkkastens wurde zunächst eine kurze Einführung in die drei Thematiken gegeben, auf denen der Schwerpunkt dieser Arbeit liegt: das *Smart Home*, die *Servicerobotik* sowie die *Künstliche Intelligenz*. Im Anschluss daran wurde eine Anforderungsanalyse durchgeführt, bei der zunächst das grundsätzliche Ziel und Anwendungsszenario definiert wurde, um im Anschluss daran die Zielgruppen und Anwendungsfälle in Form von Use-Case-Diagrammen definieren zu können. Darauf aufbauend konnten die nötigen Anforderungen an die einzelnen zu entwickelnden Komponenten abgeleitet und spezifiziert werden.

Im Folgenden wurden die Anwendung *Smarter Home* sowie der Lernbalkkasten und mögliche Entwicklungsbausteine sorgfältig geplant und konzeptioniert.

Bezüglich der Anwendung *Smarter Home* wurden dafür zunächst bestehende ähnliche Systeme untersucht, um dabei erhaltene Erkenntnisse für die Konzeption der eigenen Entwicklung zu berücksichtigen.

Anschließend wurden die verschiedenen Anwendungsmodi der Anwendung konzeptioniert, die sich an dem Erstellungsprozess eines KNN orientieren. In der Folge wurden mögliche Lösungsoptionen untersucht, diskutiert und ausgewählt, wie bspw. die Wahl eines geeigneten Frameworks zur Arbeit mit KNN oder eines passenden Funkprotokolls für die drahtlose Kommunikation zwischen Sensorik und Anwendung.

Bei der Implementierung der Anwendung *Smarter Home* wurde bei der Systemarchitektur darauf geachtet, eine modulare und erweiterbare Anwendung zu entwickeln, indem das Prinzip einer 3-Schichten-Architektur eingehalten wurde. Dadurch ist es zukünftig möglich, Weiterentwicklungen ohne größere Anpassungen zu realisieren.

Bei der Umsetzung der Benutzeroberfläche wurde besonderer Wert auf eine gute Bedienbarkeit und eine übersichtliche Benutzerführung gelegt. Mit Hilfe der Benutzerführung konnte zusätzlich dafür gesorgt werden, dass Eingabefehler auf Seiten des Anwenders vermieden werden.

Für die Konzeption des Lernbaukastens wurde sich einerseits an dem Lernprozess nach MOMBI orientiert und andererseits an einem bereits bestehendem und etabliertem Schülerlabor aus dem Bereich *EcoLab*. In Folge dessen wurden unter der Berücksichtigung der Zielgruppe drei aufeinander aufbauende Lernbausteine konzipiert, bestehend aus einer Theorie-, einer Praxis- und einer Festigungsphase.

Zusätzlich wurden drei Entwicklungsbausteine entworfen, die jeweils an eine Zielgruppe adressiert sind und mögliche Weiterentwicklungen der Gesamtanwendung vorschlagen.

Hinsichtlich der Umsetzung des Lernbaukastens wurde großen Wert darauf gelegt, den Lernprozess interaktiv zu gestalten und die Schüler nicht als passives sondern aktives Element sowohl bei der Theorie-, als auch bei der Praxis- und Festigungsphase einzubinden.

Bei der Realisierung des Prototyps für das Szenario zur Überwachung des Schlafrhythmus von Karin wurden bewusst vertraute Elemente wie die Bausteine des Spielzeugherstellers *LEGO* verwendet um einerseits Berührungsängste zu nehmen und andererseits auch eine haptischen und spielerische Komponente den Schülern an die Hand zu geben. Bei der verwendeten Hardware für die Sensorik und die Verarbeitung und Übertragung der Daten an die Anwendung *Smarter Home* wurde sich an dem bereits vorhandenen Hardwarerepertoire des Schülerlabors für Robotic orientiert und auf den *Arduino UNO R3* und kompatible Zusatzhardware gesetzt.

Die Konzepte und Vorüberlegungen aus dem theoretischen Abschnitt dieser Abschlussarbeit konnten in der Praxis erfolgreich umgesetzt werden. Unerwartet aufgetretene Probleme während der Umsetzung konnten durch entsprechend angepasste Lösungskonzepte gelöst werden. Die entwickelte Anwendung sowie der entstandene Prototyp für den Lernbaukasten wurden bereits erfolgreich in ersten Tests eingesetzt und einer ersten Prüfung hinsichtlich der Tauglichkeit und Handhabung unterzogen.

Sämtliche Ausarbeitungen dieser Masterarbeit befinden sich auf der beigelegten CD-ROM (siehe Anhang B).

Anhang A

Anforderungsdefinition

In diesem Kapitel werden die einzelnen Anforderungen an die zu entwickelnde IVE, den Lernbaukasten sowie die Entwicklungsbausteine näher aufgeschlüsselt. Die vollständige Anforderungsdefinition liegt als Excel-Datei mit dem Namen „Anforderungen.xlsx“ auf der CD-ROM in dem Verzeichnis „Dokumente/Anforderungsanalyse“ bei (siehe Anhang B).

ID	Komponente	Kriterium	Anforderung	Art
R001	Der Lernbaukasten	muss	sich aus mehreren Lernbausteinen zusammensetzen.	F
R002	Der Lernbaukasten	soll	ein Wissensnetz zu dem Thema KI aufbauen.	F
R003	Der Lernbaukasten	soll	die Möglichkeit zum selbstständigen Erarbeiten von einzelnen Inhalten zur KI geben.	F
R004	Der Lernbaukasten	muss	auf die Zielgruppe zugeschnitten sein.	F
R005	Der Lernbaukasten	muss	Schnittstellen zu den Lehrplänen des Fachs Informatik der Zielgruppe aufweisen.	F
R006	Der Lernbaukasten	soll	aus aufeinander aufbauenden Lernbausteinen bestehen.	F
R007	Der Lernbaukasten	muss	mindestens einen Lernbaustein mit praktischem Inhalt enthalten.	Q
R008	Ein Lernbaustein	muss	über einen angesetzten zeitlichen Rahmen verfügen.	F
R009	Die verwendete Softwarelösung	muss	in jedem Umfeld frei nutzbar sein.	Q
R010	Die intelligente Verarbeitungseinheit	muss	zumindest ein Werkzeug der KI verwenden.	Q
R011	Die intelligente Verarbeitungseinheit	muss	über eine grafische Benutzeroberfläche verfügen.	F
R012	Die grafische Benutzeroberfläche	muss	über einen Bereich <i>Discover</i> verfügen.	F

F = Funktionale Anforderung, Q = Qualitative Anforderung

ID	Komponente	Kriterium	Anforderung	Art
R013	Die grafische Benutzeroberfläche	muss	über einen Bereich <i>Training</i> verfügen.	F
R014	Die grafische Benutzeroberfläche	muss	über einen Bereich <i>Go Live</i> verfügen.	F
R015	Die intelligente Verarbeitungseinheit	muss	eine Schnittstelle für Serviceroboter zur Verfügung stellen.	F
R016	Die Schnittstelle für Serviceroboter	muss	eine drahtlose Kommunikation ermöglichen.	Q
R017	Die Schnittstelle für Serviceroboter	muss	das Anmelden für Serviceroboter ermöglichen.	F
R018	Die Schnittstelle für Serviceroboter	muss	das Abrufen der Informationen über das Verhalten der überwachten Person ermöglichen.	F
R019	Die Schnittstelle für Serviceroboter	muss	das Abmelden für Serviceroboter ermöglichen.	F
R020	Die intelligente Verarbeitungseinheit	muss	eine Schnittstelle für Sensoreinheiten zur Verfügung stellen.	F
R021	Die Schnittstelle für Sensoreinheiten	muss	eine drahtlose Kommunikation ermöglichen.	Q
R022	Die Schnittstelle für Sensoreinheiten	muss	das Anmelden für Sensoren ermöglichen.	F
R023	Die Schnittstelle für Sensoreinheiten	muss	das Empfangen von Sensorwerten für angemeldete Sensoren ermöglichen.	F
R024	Der Bereich Discover	muss	das Starten des Discovermodus ermöglichen.	F
R025	Der Bereich Discover	muss	das Beenden des Discovermodus ermöglichen.	F
R026	Der Bereich Discover	muss	jeden angemeldeten Sensor darstellen.	F
R027	Der Bereich Discover	muss	das Anlegen bzw. Anmelden von einem simulierten Sensor ermöglichen.	F
R028	Der Bereich Discover	muss	jeden angemeldeten Sensor mit ID, Name, Sensorart (z.B. Temperatur, Feuchtigkeit etc.) und aktuellem Sensorwert anzeigen.	F
R029	Der Bereich Discover	muss	das Einstellen von Sensorwerten bei realen Sensoren ermöglichen.	F
R030	Der Bereich Discover	muss	das Einstellen von Sensorwerten bei simulierten Sensoren ermöglichen.	F
R031	Der Bereich Discover	muss	das Verwenden der aktuellen Uhrzeit ermöglichen.	F
R032	Der Bereich Discover	muss	das Verwenden des aktuellen Wochentags ermöglichen.	F
R033	Der Bereich Discover	muss	das Simulieren der Uhrzeit ermöglichen.	F
R034	Der Bereich Discover	muss	das Simulieren des Wochentags ermöglichen.	F

F = Funktionale Anforderung, Q = Qualitative Anforderung

ID	Komponente	Kriterium	Anforderung	Art
R035	Der Bereich Discover	muss	das Entfernen eines angemeldeten Sensors ermöglichen.	F
R036	Der Bereich Training	muss	das Starten eines Trainingsmodus ermöglichen.	F
R037	Der Bereich Training	muss	das Beenden eines Trainingsmodus ermöglichen.	F
R038	Der Bereich Training	muss	das Abspeichern eines Trainingszustandes ermöglichen.	F
R039	Der Bereich Training	muss	das Laden eines abgespeicherten Trainingszustandes ermöglichen.	F
R040	Der Bereich Training	muss	die Durchführung eines Trainings ermöglichen.	F
R041	Der Bereich Go Live	muss	das Starten des Überwachungsmodus ermöglichen.	F
R042	Der Bereich Go Live	muss	das Beenden des Überwachungsmodus ermöglichen.	F
R043	Der Bereich Go Live	muss	für die Überwachung einen Trainingszustand verwenden.	F
R044	Der Bereich Go Live	soll	in der Lage sein, grobe Abweichungen zwischen aktuellen und angelernten Sensorwerten zu erkennen.	F
R045	Der Bereich Go Live	muss	die Informationen über die Abweichung dem Anwender anzeigen.	F
R046	Der Bereich Go Live	muss	die Informationen über die Abweichung den angemeldeten Servicerobotern über die Schnittstelle bereitstellen.	F
R047	Der Bereich Go Live	muss	jeden angemeldeten Serviceroboter darstellen.	F
R048	Ein Entwicklungsbaustein	muss	an eine Zielgruppe gerichtet sein.	F
R049	Ein Entwicklungsbaustein	muss	eine Zielstellung enthalten.	F
R050	Ein Entwicklungsbaustein	muss	enthalten, für welchen Rahmen er konzeptioniert ist.	F
R051	Ein Entwicklungsbaustein	muss	vom Umfang her auf den festgelegten Rahmen abgestimmt sein.	Q

Anhang B

Die beigelegte CD-ROM

B.1 Inhalt der CD-ROM

Die beigelegte CD-ROM hat den folgenden Inhalt:

Dokumente - Das Verzeichnis enthält die zusammengestellten Anforderungen sowie die Versuchsdaten und -ergebnisse aus der Testphase (Abschnitt 9) als Excel-Dokumente. Zusätzlich enthält es die vorliegende Masterarbeit als PDF-Dokument.

Programme - Das Verzeichnis enthält eine funktionsfähige Version der Anwendung *Smarter Home*.

Quelltexte - Dieses Verzeichnis enthält den Quelltext zu der realisierten Anwendung *Smarter Home*. Sie können in eine beliebige IntelliJ IDEA importiert werden. Es wird empfohlen die *IntelliJ IDEA CE* in der Version *2016.2.5* zu verwenden.

Lernbaukasten - In diesem Verzeichnis befinden sich die Ausarbeitungen zu den Lernbausteinen *Theorie* und *Praxis*. Dazu gehören ebenfalls die *Arduino*-Programme, die über die Arduino IDE¹ geöffnet werden können. Empfohlen wird hierfür die Arduino IDE in der Version 1.6.9.

¹ Arduino IDE: <https://www.arduino.cc/en/main/software>

Abkürzungsverzeichnis

AAL	Ambient Assisted Living
AF	Auffüllen
AI	Artificial Intelligence
ANN	Artificial Neural network
API	Application Programming Interface
AS	Ausschütten
AV	Aktuelles Verhalten
AWT	Abstract Window Toolkit
BLE	Bluetooth Low Energy
CART	Classification and Regression Trees
CRF	Conditional Random Fields
CSV	Comma-Seperated Values
DBIS	Datenbank-Infosystem
DBS	Datenbanksystem
DV	Differenzverhaltensabbild
ESPRIT	European Strategic Programme for Research and Development in Information Technology
Fraunhofer IMS	Fraunhofer-Institut für Mikroelektronische Schaltungen und Systeme
GPL	General Public License
GPU	Graphics Processing Unit
HM	Heuristics Miner
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identifikationsnummer
IVE	Intelligente Verarbeitungseinheit
JAR	Java Archive File
JAX-RS	Java API for RESTful Web Services

JDBC	Java Database Connectivity
JOONE	Java Object Oriented Neural Engine
JSON	JavaScript Object Notation
KI	Künstliche Intelligenz
KNN	Künstliche neuronale Netze
LGPL	Lesser General Public License
LISP	List Processing
LTU	Linear Treshold Unit
MOMBI	Model-of-Model-Based Instruction
MV	Mittleres Verhalten
PROLOG	Programmieren in Logik
PSM	Power Selekt Management
PhD	Doctor of Philosophy
RC	Release Candidate
REST	Representational State Transfer
RFID	Radio-Frequency Identification
RX	Receiver
RXTX	Receiver Transmitter
SAMDY	Sensorbasiertes adaptives Monitoring System
SEK	Sekundarstufe
SVM	Support Vector Machine
SQL	Structed Query Language
TH Wildau	Technische Hochschule Wildau
TX	Transmitter
URI	Uniform Resource Identifier
US	Umschütten
USB	Universal Serial Bus
VNR	Versuchsnummer
XML	Extensible Markup Language

Glossar

A*-Algorithmus

Der A*-Algorithmus dient zur Bestimmung des kostengünstigsten Pfades zwischen zwei Knoten (Start und Ziel) innerhalb eines Suchbaumes. Es handelt sich um einen informierten Algorithmus, d.h. er verwendet Heuristiken, die das Finden der Lösung beschleunigen sollen. Der A*-Algorithmus benutzt hierbei zur Pfadbestimmung die bisherigen sowie die geschätzten noch folgenden Kosten bis zum Ziel. [12, S.136ff]

Backpropagation

Das Backpropagation-Lernverfahren zählt zu den Verfahren des überwachten Lernens zum Anpassen der Verbindungsgewichte. Dabei erfolgt anhand der Abweichung von tatsächlichem und gewünschtem Ergebnis eine Gewichtsänderung. Diese Gewichts Anpassung erfolgt schichtweise, beginnend mit der letzten verdeckten Schicht endend an der ersten verdeckten Schicht, die an die erste neuronale Schicht mit den Eingangsneuronen angrenzt. [12, S. 212]

Bestensuche

Die Bestensuche dient zur Bestimmung des kostengünstigsten Pfades zwischen zwei Knoten (Start und Ziel) innerhalb eines Suchbaumes. Es handelt sich um einen informierten Algorithmus, d.h. er verwendet Heuristiken, die das Finden der Lösung beschleunigen sollen. Die Bestensuche benutzt hierbei zur Pfadbestimmung die geschätzten Kosten bis zum Ziel. [12, S.135f]

Bichromatisch

Bichromatisch ist ein Begriff aus der Graphentheorie und bedeutet im Zusammenhang mit Petrinetzen, dass in zwei Arten von Knoten unterschieden wird: die Zustände eines Systems (Vorbedingungen) und die Übergänge für Ereignisse (Aktivitäten). [40]

Breitensuche	Die Breitensuche dient zur Bestimmung des kürzesten Pfades zwischen zwei Knoten (Start und Ziel) innerhalb eines Suchbaumes. Die Breitensuche zählt zu den uninformierten Suchalgorithmen und untersucht, vom Ausgangsknoten ausgehend, zunächst alle direkt angrenzenden Knoten, bevor Folgeknoten untersucht werden. Angewandt auf den Suchbaum, wird dieser bei der Breitensuche ebenenweise durchsucht. Der Einsatz der Breitensuche ist dann sinnvoll, wenn keine Zusatzinformationen zu dem vorliegenden Problem vorhanden sind. [12, S.119ff]
CART	Der CART ist ein Klassifizierungs-Algorithmus und gehört zu den Entscheidungsbaum-Verfahren. [18, S. 10f]
Conditional Random Fields	Conditional Random Fields (CRF) sind ungerichtete prohabilitische Modelle, mit denen es möglich ist, sequenzielle Daten zu klassifizieren bzw. zu segmentieren. Ein CRF kann zu jeder Zeit auf alle Teile der eingegeben Sequenz zugreifen. [75]
Convention over Configuration	Darunter wird ein Paradigma für das Software-Design, das die Komplexität zur Konfiguration einer Anwendung minimieren soll. Dafür werden Konventionen festgelegt, die, bei Einhaltung auf Seiten des Programmierers, keine weiteren Konfigurationen erforderlich machen. Solch eine Konvention kann bspw. die Verwendung von einheitlichen Bezeichnern darstellen. [3]
Deep Learning	Deep Learning ist ein Teilbereich des Machine Learnings und steht quasi für die Wiederbelebung der bereits seit Jahrzehnten bestehenden künstlichen neuronalen Netze (KNN). Dieses Wiederaufleben wurde vor allem durch die verbesserten Rechenleistung seit Ende des 20. und Anfang des 21. Jahrhunderts möglich. Denn im Gegensatz zu früheren Architekturen, besitzen KNN, die im Rahmen des Deep Learnings eingesetzt werden, eine sehr viel komplexere Struktur, die sich vor allem in der Anzahl der Tiefe der Schichten widerspiegeln. Hierfür bestehen selbstverständlich auch entsprechende Anforderungen an die Rechenleistung. Zusätzlich kann heutzutage ein KNN mit einer Vielzahl an digitalen Datenbeständen trainiert werden, sodass dies hinsichtlich der Qualität der Netze einen weiteren Meilenstein beim Deep Learning ermöglichte. [64]
Don't Repeat Yourself	Ein Prinzip, das Redundanzen minimieren und wenn möglich gar komplett vermeiden soll. Redundanzen stellen potentielle Fehlerquellen dar, da sie an verschiedenen Stellen eine einheitliche Pflege erfordern. [3]

Elman-Netz	Das Elman-Netz, auch <i>Simple Recurrent Network</i> (SRN) genannt, ist nach dem Erfinder Jeffrey L. Elman benannt. Dabei handelt es sich um ein einfaches KNN, das sogenannte Rückkopplungen enthält. Es wird in der Literatur als eine Netzwerkarchitektur beschrieben, die sich besonders dafür eignet, Eingaben in zeitlicher Abhängigkeit zu verarbeiten. [17]
Expertensystem	<i>„in der Künstlichen Intelligenz (KI) wird ein Programm oder ein Softwaresystem als Expertensystem bezeichnet, wenn es in der Lage ist, Lösungen für Probleme aus einem begrenzten Fachgebiet (Wissensdomäne) zu liefern, die von der Qualität her denen eines menschlichen Experten vergleichbar sind oder diese sogar übertreffen (Expertenwissen).“</i> [42]
Hidden Markov Model	Hidden Markov Models (HMM) werden bei der Erkennung von Mustern in sequentiellen Daten, sie sind benannt nach dem russischen Mathematiker A. Markov. Bei dem HMM handelt es sich um ein stochastisches Modell, gekennzeichnet durch einen zweistufigen Prozess. In der ersten Stufe wird mit Hilfe einer Markow-Kette ein System angefertigt, deren Zustände von außen nicht einsehbar sind (hidden). In der zweiten Stufe wird ein stochastischer Prozess mit sogenannten Observationen erstellt. Darunter werden jederzeit einsehbare Ausgangssymbole verstanden, die mit einer Wahrscheinlichkeitsverteilung ausgestattet sind. Anhand der Sequenz von Ausgangssymbolen wird nun versucht auf die nicht sichtbaren Zustände zu schließen. [37]
Hopfield-Netz	Man spricht von einem Hopfield-Netz, wenn alle verwendeten Neuronen vollständig untereinander verknüpft sind (Rückkopplung), außer mit sich selbst.
JDBC	JDBC ist die Abkürzung für <i>Java Database Connectivity</i> . Darunter wird eine Sammlung von Schnittstellen verstanden, mit denen es möglich ist, relationale Datenbanksystem von Java zu verwenden. [71, S. 1343]
Konnektionismus	Forschungsrichtung der Informatik, die sich mit der Entwicklung neuronaler Netze zur Simulierung von künstlicher Intelligenz befasst [8]
Levenberg-Marquart-Algorithmus	Bei dem Levenberg-Marquart-Algorithmus handelt es sich um einen numerischen Optimierungsalgorithmus, der das Minimum einer Funktion lokalisiert mit Hilfe der Methode der kleinsten Quadrate. Mehr dazu kann unter [47] nachgelesen werden.

Maschinelles Lernen	<i>„Anwendung und Erforschung von Verfahren, durch die Computersysteme befähigt werden, selbstständig Wissen aufzunehmen und zu erweitern, um ein gegebenes Problem besser lösen zu können als vorher (Learning).“ [43]</i>
Medienkompetenz	Unter der Medienkompetenz wird der souveräne Umgang mit den Medien der aktuellen wesentlichen Kulturtechniken verstanden. [76, S. 4]
Multilayer Perzeptron	Man spricht von einem Multilayer Perzeptron, sobald es neben der Eingangs- und Ausgangsschicht noch mindestens eine weitere Schicht gibt. Dabei handelt es sich um sogenannte versteckte Schichten (hidden layer).
Nächster-Nachbar-Suche	Die Nächste-Nachbar-Suche dient zur Bestimmung des kostengünstigsten Pfades zwischen zwei Knoten (Start und Ziel) innerhalb eines Suchbaumes. Es handelt sich um einen informierten Algorithmus, d.h. er verwendet Heuristiken, die das Finden der Lösung beschleunigen sollen. Die Nächste-Nachbar-Suche benutzt hierbei zur Pfadbestimmung die Kosten zum direkt angrenzenden noch nicht besuchten Nachbar. [12, S.132f]
Naives Bayes-Netz	Bei einem naiven Bayes-Netz handelt es sich um eine Funktion, die dazu in der Lage ist jedem Punkt in einem Merkmalsraum einer Klasse zuzuordnen. Dadurch kann das Netz die Zugehörigkeit eines bestimmten Objektes zu einer Klasse zu bestimmen. Der Naive Bayes-Klassifikator zeichnet sich vor allem durch seine schnellen Berechenbarkeit und einer guten Erkennungsrate aus. S. 237ff [18]
Perzeptron	Unter einem Perzeptron versteht man ein einfaches vorwärts gerichtetes künstliches neuronales Netz, das über keine verdeckten Schichten verfügt und über das Backpropagation-Verfahren trainiert wird. [12, S. 202]
Petrinetze	Petrinetze sind mathematische Strukturen in Form eines gerichteten , markierten bichromatischen ⁸ Graphen. Sie ermöglichen die Analyse, die Modellbildung und die Simulation von dynamischen Systemen mit nebenläufigen oder auch nicht-deterministischen Vorgängen. Die Petrinetze sind nach ihrem Erfinder, dem deutschen Mathematiker Carl Adam Petri, benannt. [40]
Prädikatenlogik zweiter Stufe	Die Prädikatenlogik zweiter Stufe (PL2) ist eine Erweiterung der Prädikatenlogik erster Stufe (PL1) und besitzt eine höhere Ausdrucksstärke. Die Erweiterung besteht darin, dass die Quantoren $\forall$ und $\exists$ nun auch über die Prädikate und Funktionen quantifiziert werden können. Mit der PL2 ist, im Gegensatz zur PL1, damit auch eine Aussage wie <i>Es gibt einen Studierenden und dieser weist alle guten Eigenschaften auf</i> möglich. [31]

Referenzimplementierung	In der Informatik wird unter einer Referenzimplementierung eine Spezifikationsprogramm verstanden, das die eigentliche Spezifikation ersetzt. Eine Referenzimplementierung gilt de facto als korrekt. So gilt eine Implementierung als fehlerfrei, sobald sie für alle Eingaben die gleichen Ausgaben wie die Referenzimplementierung erzeugt. [32, S. 146ff]
Resilient Backpropagation	Resilient Backpropagation (RP) ist ein iterativer Algorithmus, der für das Trainieren von KNN eingesetzt werden kann. Er ist dem Backpropagation-Algorithmus sehr ähnlich, besitzt aber den Vorteil, dass das Training mit ihm vergleichsweise schneller ist. Dafür gilt der RP-Algorithmus als komplexer in der Implementierung im Vergleich zum Backpropagation-Algorithmus. Mehr kann unter [45] nachgelesen werden,
Symbolverarbeitung	Unter der Symbolverarbeitung wird das Rechnen mit Symbolen verstanden. Zu solchen Symbolen zählen Variablen, Konstanten und Funktionssymbole. [19, S. 2]
Tiefensuche	Die Tiefensuche dient zur Bestimmung des kürzesten Pfades zwischen zwei Knoten (Start und Ziel) innerhalb eines Suchbaumes. Die Tiefensuche zählt zu den uninformierten Suchalgorithmen und untersucht, vom Ausgangsknoten ausgehend, zunächst einen Pfad vollständig in die Tiefe, bevor im Anschluss auch abgehende Pfade durchsucht werden. Der Einsatz der Tiefensuche ist dann sinnvoll, wenn keine Zusatzinformationen zu dem vorliegenden Problem vorhanden sind. [12, S.119ff]
Turingautomat	Mathematisch fundiertes Modell eines Computers, mit dem es möglich ist, maschinelle Berechnungen zu formalisieren und darüber strukturiert zu argumentieren. Benannt ist der Turingautomat nach Alan M. Turing, auf den dieses Modell zurückgeht. [41]

Abbildungsverzeichnis

1.1	Die Königsdisziplin der <i>Künstlichen Intelligenz</i> ist die Fähigkeit, eine Maschine denken zu lassen.	9
2.1	Die Geschichte und Meilensteine der Künstlichen Intelligenz auf einen Blick. .	17
2.2	Das Kanister-Problem ausschnittsweise als Suchraum dargestellt mit Hilfe eines gerichteten Graphen.	23
2.3	Anhand des Graphen aus Abbildung 2.2 ergibt sich der folgende Suchbaum (Ausschnitt), ausgehend von dem Startzustand, in dem beide Kanister leer sind.	24
2.4	Das biologische (oben) und das künstliche Neuron (unten) im Vergleich. . . .	25
2.5	Der Aufbau des KNN zur Abbildung einer UND-Schaltung.	26
2.6	Das vorwärts gerichtete KNN ist in drei Schichten aufgeteilt, eine Eingabeschicht (grün), eine verdeckte Schicht (blau) und eine Ausgabeschicht (rot). .	27
2.7	Es gibt verschiedene Architekturen für KNN. Die Wahl der Architektur hängt von der Problemstellung ab.	28
2.8	Die Entwicklung eines KNN erfolgt in einem iterativen Prozess mit Hilfe von Trainings- und Testdaten. [12]	28
2.9	Das <i>Google self-driving car</i> benutzt Techniken der <i>Künstlichen Intelligenz</i> , um autonom am Straßenverkehr teilnehmen zu können.	29
2.10	Jährlich findet in Form des RoboCups ein weltweites Kräfteressen statt, wie u.a. im Fußball zwischen den <i>NAO</i> -Robotern der Firma <i>Aldebaran</i>	30
3.1	Die Anwendungs- und Einsatzbereiche der Technik Smart Home.	33
3.2	Der Roboter <i>Romeo</i> von der Firma <i>Aldebaran</i> unterstützt ältere Menschen in ihren eigenen vier Wänden. [63]	35
4.1	Schematische Darstellung der Infrastruktur und der Komponenten für das Ausgangsszenario.	39
4.2	Ziel der Sensoreinheit ist es, sich an die Verarbeitungskomponente an- und abmelden zu können und aktuelle Sensordaten an diese zu senden.	41
4.3	Der Anwender hat im Anwendungsbereich <i>Discover</i> bspw. die Möglichkeit, angemeldete Sensoren einzusehen.	43
4.4	Dem Anwender steht über eine Benutzeroberfläche im Bereich <i>Training</i> die Möglichkeit zur Verfügung, ein Verhaltensmuster anzutrainieren.	44
4.5	Der Anwender kann im <i>Go Live</i> -Modus u.a. den Überwachungsstatus einsehen.	45
4.6	Dem <i>Serviceroboter</i> müssen über die intelligente Verarbeitungskomponente die Funktionalitäten <i>an-</i> und <i>abmelden</i> sowie das <i>Abrufen von Informationen</i> bereitgestellt werden.	46
5.1	<i>CASAS</i> zeichnet sich durch eine leichtgewichtige 3-Schichten-Architektur aus.	51

5.2	Im <i>Discover</i> -Modus wird das KNN vom Benutzer erstellt.	56
5.3	Die einzelnen Funktionen des <i>Discover</i> -Modus ermöglichen das Abbilden eines zu beobachtenden Szenarios in ein KNN.	56
5.4	Die einzelnen Funktionen des <i>Trainings</i> -Modus ermöglichen das Trainieren eines KNN.	57
5.5	Mit Hilfe von positiven (<i>true</i>) und negativen (<i>false</i>) Trainingsdatensätzen wird das KNN trainiert.	58
5.6	Während des <i>Go Live</i> -Modus hat der Anwender die Möglichkeit, den aktuellen Status des Netzes und die angemeldeten Listener einzusehen sowie die Sensorwerte der Eingangsneuronen zu modifizieren.	59
5.7	Das KNN kann im <i>Go Live</i> -Modus Aussagen darüber treffen, ob ein Zustand <i>true</i> oder <i>false</i> ist.	60
5.8	Die grafische Oberfläche führt den Benutzer durch die drei Anwendungsbereiche <i>Discover</i> , <i>Training</i> und <i>Go Live</i>	70
5.9	Der Bereich <i>Discover</i> ermöglicht dem Benutzer das Erstellen der grundlegenden Architektur des KNN.	71
5.10	Der Bereich <i>Training</i> ermöglicht dem Benutzer bspw. das Anlegen von Trainingsdatensätzen und die Durchführung eines Trainings.	72
5.11	Im <i>Go Live</i> -Bereich geht die Anwendung in den Live-Modus und der Benutzer kann den aktuell detektierten Zustand der Anwendung einsehen.	73
5.12	Das Konzept zur Umsetzung der intelligenten Verarbeitungseinheit <i>Smarter Home</i> im Überblick.	74
6.1	Der Lernprozess setzt sich aus insgesamt fünf Teilprozessen zusammen, die nacheinander durchlaufen werden.	76
6.2	Durch das zielgerichtete Abarbeiten der Lehrschritte nach MOMBI, werden die einzelnen Teilprozesse des Lernens angesteuert.	77
6.3	Der Lernbaukasten setzt sich insgesamt aus den drei Lernbausteinen <i>Theorie</i> , <i>Praxis</i> und <i>Festigung</i> zusammen. Diese drei Bausteine orientieren sich an den fünf Lehrschritten nach MOMBI.	79
6.4	Der erste Lernbaustein <i>Theorie</i> hat die Intention, den Schülern die nötigen Informationen für den praktischen Einsatz in der Hochschule zur Verfügung zu stellen.	80
6.5	Der Lernbaustein für die Praxis soll die Planung und Konstruktion, die Programmierung und das Arbeiten mit KNN für die Schüler ermöglichen.	82
6.6	Der geplante Prototyp für den praktischen Lernbaustein repräsentiert das Schlafzimmer von Karin. Darin soll mit Hilfe der IVE das Schlafverhalten von Karin trainiert und überwacht werden können.	83
6.7	Der <i>Arduino UNO</i> wird für die Erzeugung sowie Übertragung der Sensordaten an die IVE eingesetzt. [1]	84
6.8	Das RFID-Kit von <i>Funduino</i> besteht aus einem RFID-Empfänger und zwei RFID-Sendern in Form einer Chipkarte und eines Schlüsselanhängers. [21]	85
6.9	Die Kommunikation zwischen <i>Arduino</i> und IVE soll über sogenannte XBee-Module erfolgen, die eine Kommunikation via ZigBee ermöglichen. Für den <i>Arduino UNO</i> wird ein entsprechendes XBee Shield verwendet. [14]	86
7.1	Die Drei-Schichten-Architektur teilt sich in die Schichten GUI, Fachkonzept und Datenhaltung auf.	92

7.2	Über den Button <i>Connect</i> kann die Verbindung zu dem XBee-Empfänger hergestellt werden.	93
7.3	Die einzelnen Eingangsneuronen sind in der Tabelle der Anwendung zeilenweise hinterlegt.	93
7.4	Beim Anlegen eines neuen Eingangsneurons vergibt der Benutzer die ID, den gewünschten Namen und den Sensortyp.	94
7.5	Die angelegten Ausgangsneuronen sind in der Tabelle <i>Output neurons</i> einzusehen. Auch hier besteht die Möglichkeit, neue Ausgangsneuronen hinzuzufügen oder bestehende zu löschen.	95
7.6	Über den Button <i>Configure Network</i> kann der Benutzer in einem Dialog die verdeckten Schichten des KNN konfigurieren.	95
7.7	Eine Übersicht zu den grundlegenden Aktivierungsfunktionen für ein KNN. .	96
7.8	Das sogenannte Bias-Neuron kann zu einer Schicht hinzugefügt werden. Dieses spezielle Neuron stellt einen Schwellwert dar.	96
7.9	Die einzelnen Bedienelemente des <i>Trainings</i> -Modus.	97
7.10	Die einzelnen Bedienelemente des <i>Go Live</i> -Modus.	98
7.11	Über einen Dialog wird der Benutzer bei Verlassen des <i>Trainings</i> -Modus über einen möglichen Datenverlust informiert.	99
7.12	Der grundsätzliche Aufbau eines Frames enthält drei Elemente, die durch ein Semikolon getrennt sind.	101
7.13	Der <i>Bonjour</i> -Frame ermöglicht das Anmelden eines Sensors an die Anwendung <i>Smarter Home</i>	102
7.14	Mit Hilfe des <i>Talk</i> -Frames können angemeldete Sensoren ihre Daten an die Anwendung <i>Smarter Home</i> senden.	103
7.15	Die beteiligten Komponenten für die Kommunikation zwischen einem Sensor und der Anwendung <i>Smarter Home</i> im Überblick.. . . .	103
7.16	Der <i>RXTX Communication Service</i> besteht aus mehreren Schichten, die bei Empfang oder Versenden einer Nachricht durchlaufen werden	104
7.17	Der <i>Sensor Manager Service</i> ist für die Verwaltung der Sensoren zuständig. Dafür nutzt dieser den <i>RXTX Communication Service</i> , um über Änderungen und neue Sensoren informiert zu werden.	105
7.18	Der hierarchische Aufbau der einzelnen Klassen zur Abbildung der jeweiligen Neuronentypen.	106
7.19	Bei der Erstellung eines Eingangsneurons kann der Typ festgelegt werden, wie z.B. <i>TIME_OF_DAY</i> oder <i>DAY_OF_WEEK</i>	107
7.20	Die Abstraktionsklasse <i>NeuralNetwork</i> dient als Vermittler zwischen dem <i>Deep Learning Service</i> und dem Framework <i>Encog</i>	108
7.21	Das in <i>Encog</i> erstellte KNN verfügt über 3 Schichten und 2 Bias-Neuronen für die erste und zweite KNN-Schicht.	110
7.22	Das KNN und die Trainingsdaten werden als SHNN-Datei gespeichert und können über die Anwendung auch wieder geladen werden.	112
7.23	Nach Start des REST-Servers registrieren sich die Endpoints an dem Servlet-Container.	113
7.24	Ursprünglich wurde die Anwendung so geplant, dass der Live-Modus und die Simulation der Werte für alle Eingangsneuronen ausschließlich über die Buttons <i>Live Data</i> und <i>Override</i> gesteuert werden.	118

7.25	Durch das Einfügen einer zusätzlichen Spalte in der Tabelle <i>Input neurons</i> , können einzelne Eingangsneuronen in den <i>Live</i> - oder <i>Override</i> -Modus überführt werden.	119
7.26	In der Anwendung sind zwei Eingangsneuronen angelegt, mit dessen Werten das Training durchgeführt werden kann. Über das Ausgangsneuron wird im Training festgelegt, ob es sich um einen positiven (true) oder negativen (false) Trainingsdatensatz handelt.	120
7.27	In der CSV-Datei repräsentiert eine Zeile die durch Semikolon getrennten Daten für einen Trainingsdatensatz.	121
7.28	Die Komponenten der Anwendung <i>Smarter Home</i> in der Gesamtübersicht. . .	122
8.1	Zu den Aufgaben der Schüler gehört die aktive Mitarbeit, das Lösen von Aufgaben und das Ergänzen von Informationen auf dem Arbeitsblatt.	124
8.2	Entsprechende Symboliken auf den Arbeitsblättern signalisieren den Schülern, welche Art von Mitarbeit von Ihnen für die jeweilige Aufgabe gefordert ist. . .	124
8.3	Der Prototyp aus Lego bietet die Möglichkeit, verschiedene Sensoren zu verbauen. In dem Bett angebracht, befindet sich der RFID-Empfänger.	128
8.4	Über einen Schlüsselring und einen zusätzlichen Draht ist der RFID-Schlüsselanhänger an Karin befestigt.	128
9.1	Die positiven Trainingsdaten enthalten vier Tage, bei denen es leichte Diskrepanzen zwischen den Aufsteh- und Zubettgehzeiten von Karin gibt.	133
9.2	Die negativen Trainingsdaten enthalten das gegenteilige Verhalten im Bezug auf das Schlafverhalten von Karin.	134
9.3	Das Multilayer Perzeptron im ersten Testversuch ist vorwärts gerichtet und enthält eine verdeckte Schicht mit 4 Neuronen.	134
9.4	Die sigmoide Funktion wird als Aktivierungsfunktion in den Tests verwendet. [30]	135
9.5	Der Verlauf der Fehlerrate bei 1000 Trainingswiederholungen.	136
9.6	Durch den Einbau eines Schwellwertes auf Seiten des Aktors, können leichte Abweichungen im Schlafverhalten abgefangen werden.	138
9.7	Das Multilayer Perzeptron im zweiten Testversuch ist ebenfalls vorwärts gerichtet und enthält zwei verdeckte Schichten mit jeweils 8 Neuronen.	138
9.8	Im Verlauf des Trainings sind immer wieder Ausreißer in der Fehlerrate zu erkennen.	139
9.9	Die Festlegung eines Schwellwertes bei dem Ausgangswert von 0,25 macht es möglich, dass die leichten Unregelmäßigkeiten bei den Aufsteh- und Zubettgehzeiten von Karin abgefangen werden können.	141
10.1	Screenshot von der intelligenten Verarbeitungseinheit <i>Smarter Home</i>	145

Tabellenverzeichnis

2.1	Aussagenlogische Verknüpfungen in Kombination mit der Alltagssprache. . . .	19
2.2	Eine Übersicht zu den möglichen Verknüpfungen in der Aussagen- und <i>Prädikatenlogik</i>	20
2.3	Die Ein- und Ausgabemuster des UND-Schalters.	27
5.1	Die beiden Funkstandards <i>Bluetooth Low Energy</i> und <i>ZigBee</i> im Vergleich. . .	60
5.2	Die drei vorgestellten Frameworks <i>Neuroph</i> , <i>Encog</i> und <i>JOONE</i> zur Erstellung und Verarbeitung von KNN noch einmal im Vergleich.	64
6.1	Die technischen Daten des <i>Arduino UNO</i> in der Übersicht. [1]	84
6.2	Die technischen Daten des RFID-Kits von <i>Funduino</i> in der Übersicht. [21] . .	85
9.1	Die Konfiguration des Netzwerkes für den ersten Testdurchlauf in der Übersicht.	135
9.2	Die Versuchsergebnisse des ersten Tests in Form eines Soll-Ist-Vergleichs gegenübergestellt.	137
9.3	Die Konfiguration des Netzwerkes für den Test in der Übersicht.	139
9.4	Die Versuchsergebnisse des zweiten Tests in Form eines Soll-Ist-Vergleichs gegenübergestellt.	140

Literaturverzeichnis

- [1] ARDUINO AG SWITZERLAND: *Arduino UNO*. http://www.arduino.org/products/boards/arduino-uno#board_programming, Zuletzt geprüft am 31.01.2017.
- [2] BAMBUSCH, F.: *Wie Roboter unser Leben prägen*. <http://www.pc-magazin.de/ratgeber/wie-roboter-unser-leben-praegen-1330067.html>, Zuletzt geprüft am 5.09.2016.
- [3] BAUMANN, JOACHIM: *Gradle - Ein kompakter Einstieg in das Build-Management-System*. dPunkt.verlag GmbH, Wiebinger Weg 17, 69115 Heidelberg, Deutschland, 1. Auflage, 2013.
- [4] BENDEL, PROF. DR. OLIVER: *Smart Metering*. <http://wirtschaftslexikon.gabler.de/Definition/smart-metering.html>, Zuletzt geprüft am 29.10.2016.
- [5] BIBLIOGRAPHISCHES INSTITUT GMBH: *Out-of-the-box-Lösung*. http://www.duden.de/rechtschreibung/Out_of_the_box_Loesung, Zuletzt geprüft am 22.02.2017.
- [6] BIBLIOGRAPHISCHES INSTITUT GMBH: *Roboter*. <http://www.duden.de/rechtschreibung/Roboter>, Zuletzt geprüft am 29.10.2016.
- [7] BIBLIOGRAPHISCHES INSTITUT GMBH: *Service*. http://www.duden.de/rechtschreibung/Service_Dienstleistung_Betreuung, Zuletzt geprüft am 29.10.2016.
- [8] BIBLIOGRAPHISCHES INSTITUT GMBH: *Konnektionismus*. <http://www.duden.de/rechtschreibung/Konnektionismus>, Zuletzt geprüft am 6.09.2016.
- [9] BRESSLER, JANINE: *Konzeption und Evaluation von Szenarien zum Einsatz humanoider Roboter für das betreute Wohnen im Alter*. Bachelorarbeit, Technische Hochschule Wildau, Studiengang Telematik, 2014.
- [10] BUNDESNETZAGENTUR: „*Smart Grid*“ und „*Smart Market*“. http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Energie/Unternehmen_Institutionen/NetzzugangUndMesswesen/SmartGridEckpunktepapier/SmartGridPapierpdf.pdf?__blob=publicationFile, Zuletzt geprüft am 28.10.2016.
- [11] BUNDESZENTRALE FÜR POLITISCHE BILDUNG: *ESPRIT*. <http://www.bpb.de/nachschlagen/lexika/176826/esprit>, Zuletzt geprüft am 9.09.2016.
- [12] CLEVE, JÜRGEN, LÄMMEL, UWE UND: *Künstliche Intelligenz*. Carl Hanser Verlag München, Kolbergerstraße 22, D-81679 München, 4. Auflage, 2012.
- [13] COOK, DIANE J., CRANDALL, AARON S., THOMAS, BRIAN L. und KRISHNAN, NARAYANAN C.: *CASAS: A Smart Home in a Box*. <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6313586>, Zuletzt geprüft am 19.12.2016.

- [14] COOKING HACKS: *XBee for Arduino and Raspberry Pi*. https://www.cooking-hacks.com/media/cooking/images/documentation/tutorial_arduino_XBee_shield/XBee_big.png, Zuletzt geprüft am 31.01.2017.
- [15] DABIR, KUNAL und ABHINANDAN: *Gradle Essentials*. Packt Publishing Ltd., Livery Place, 35 Livery Street, Birmingham B3 2PB, UK, 2015.
- [16] EIBEGGER, MANDANA: *Die Resolution*. <http://www.logic-rulez.net/aussagenlogik/resolution.php>, Zuletzt geprüft am 10.10.2016.
- [17] ELMAN, JEFFREY L.: *Finding Structure in Time*. Cognitive Science, Seiten 179–211.
- [18] ERTEL, WOLFGANG: *Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung*. Springer Verlag, 4. Auflage, 2016.
- [19] ERTEL, WOLFGANG: *Mathematik für Informatiker*. <http://www.hs-weingarten.de/~ertel/vorlesungen/mathi/mathi-skript.pdf>, Zuletzt geprüft am 21.11.2016.
- [20] FRAUNHOFER IMS: *Sensorbasiertes adaptives Monitoringsystem für die Verhaltensanalyse von Senioren: SAMDY; Abschlussbericht;*. <https://www.tib.eu/suchen/id/TIBKAT:791122328/>, Zuletzt geprüft am 12.12.2016.
- [21] FUNDUINO GMBH: *Inferenzmaschine*. <http://www.funduinoshop.com/RFID-KIT>, Zuletzt geprüft am 31.01.2017.
- [22] GENSTHALER, BRIGITTE M.: *Demenzpatienten: Bloß nicht ins Krankenhaus*. <http://www.pharmazeutische-zeitung.de/index.php?id=42759>, Zuletzt geprüft am 18.11.2016.
- [23] GOLEM.DE IT-NEWS FÜR PROFIS: *ZigBee und Z-Wave - So funkt's im Smarthome*. <http://www.golem.de/news/zigbee-und-z-wave-so-funkts-im-smarthome-1502-111858.html>, Zuletzt geprüft am 17.01.2017.
- [24] GOLEM.DE IT-NEWS FÜR PROFIS: *Künstliche Intelligenz*. <http://www.golem.de/specials/ki/>, Zuletzt geprüft am 5.09.2016.
- [25] HANKE, ULRIKE und WINANDY, SAMANTHA: *Lehrveranstaltungen lernförderlich gestalten*. Tübinger Beiträge zur Hochschuldidaktik; (2014) 10,2.
- [26] HEATON, JEFF: *An Introduction to Encog Neural Networks for Java*. <https://www.codeproject.com/Articles/52847/An-Introduction-to-Encog-Neural-Networks-for-Java>, Zuletzt geprüft am 8.04.2017.
- [27] HEATON RESEARCH: *Encog Machine Learning Framework*. <http://www.heatonresearch.com/encog/>, Zuletzt geprüft am 6.04.2017.
- [28] HEISE ONLINE: *Künstliche Intelligenz*. <http://www.heise.de/thema/K%C3%BCnstliche-Intelligenz>, Zuletzt geprüft am 5.09.2016.
- [29] HELMHOLTZ GEMEINSCHAFT: *Die Roboter kommen - Hilfe in der Pflege und im Alltag*. <https://www.helmholtz.de/gesundheitswissenschaften/die-roboter-kommen-hilfe-in-der-pflege-und-im-alltag-2900/>, Zuletzt geprüft am 29.10.2016.
- [30] HOBBIZINE: *a neural network for arduino*. <http://robotics.hobbizine.com/arduinoann.html>, Zuletzt geprüft am 24.03.2017.

- [31] HOFFMANN, DIRK: *Theoretische Informatik*. Carl Hanser Verlag GmbH & Co. KG, D-81631 München, 3. Auflage, 2015.
- [32] HOFFMANN, DIRK W.: *Software-Qualität*. Springer Vieweg, Springer-Verlag Berlin Heidle, 2. Auflage, 2013.
- [33] HÄGELE, MARTIN, BLÜMLEIN, NIKOLAUS UND KLEINE, OLIVER: *Wirtschaftlichkeitsanalysen neuartiger Servicerobotik-Anwendungen und ihre Bedeutung für die Robotik-Entwicklung*. http://www.ipa.fraunhofer.de/fileadmin/user_upload/Publikationen/Studien/Studientexte/Studie_EFFIROB.pdf, Zuletzt geprüft am 29.10.2016.
- [34] IBM DEUTSCHLAND: *Watson geht an die Arbeit*. <http://www-05.ibm.com/de/watson/praxis.html>, Zuletzt geprüft am 23.11.2016.
- [35] INTERNATIONAL FEDERATION OF ROBOTICS: *Service Robots*. <http://www.ifr.org/service-robots/>, Zuletzt geprüft am 29.10.2016.
- [36] IT WISSEN: *Drei-Schichten-Architektur*. <http://www.itwissen.info/definition/lexikon/Drei-Schichten-Architektur-three-tier-architecture.html>, Zuletzt geprüft am 14.02.2017.
- [37] IT WISSEN: *HMM (Hidden Markov Modell)*. <http://www.itwissen.info/definition/lexikon/Hidden-Markov-Modell-HMM.html>, Zuletzt geprüft am 16.12.2016.
- [38] IT WISSEN: *BLE (Bluetooth Low Energy)*. <http://www.itwissen.info/definition/lexikon/ZigBee-ZigBee.html>, Zuletzt geprüft am 17.01.2017.
- [39] IT WISSEN: *ZigBee*. <http://www.itwissen.info/definition/lexikon/ZigBee-ZigBee.html>, Zuletzt geprüft am 17.01.2017.
- [40] IT WISSEN: *Petrinetz*. <http://www.itwissen.info/Petrinetz-petri-net.html>, Zuletzt geprüft am 4.04.2017.
- [41] KROHMER, ANTON: *Turingmaschinen*. <https://hpi.de/friedrich/teaching/units/turing-machines.html>, Zuletzt geprüft am 21.11.2016.
- [42] LACKES, PROF. DR. RICHARD UND SIEPERMANN, DR. MARKUS: *Expertensystem*. <http://wirtschaftslexikon.gabler.de/Archiv/57388/expertensystem-v9.html>, Zuletzt geprüft am 10.10.2016.
- [43] LACKES, PROF. DR. RICHARD UND SIEPERMANN, DR. MARKUS: *Maschinelles Lernen*. <http://wirtschaftslexikon.gabler.de/Definition/maschinelles-lernen.html>, Zuletzt geprüft am 8.04.2017.
- [44] LANDESINSTITUT FÜR SCHULE UND MEDIEN BERLIN-BRANDENBURG (LISUM): *Vorläufiger Rahmenlehrplan für den Unterricht in der gymnasialen Oberstufe im Land Brandenburg - Informatik*. http://bildungsserver.berlin-brandenburg.de/fileadmin/bbb/unterricht/rahmenlehrplaene/gymnasiale_oberstufe/curricula/2011/Informatik-VRLP_GOST_2011_Brandenburg.pdf, Zuletzt geprüft am 31.01.2017.
- [45] MCCAFFREY, JAMES: *How To Use Resilient Back Propagation To Train Neural Networks*. <https://visualstudiomagazine.com/articles/2015/03/01/resilient-back-propagation.aspx>, Zuletzt geprüft am 24.03.2017.

- [46] MINISTERIUM FÜR BILDUNG, JUGEND UND SPORT DES LANDES BRANDENBURG: *Vorläufiger Rahmenlehrplan für den Unterricht in der gymnasialen Oberstufe im Land Brandenburg*. http://bildungsserver.berlin-brandenburg.de/fileadmin/bbb/unterricht/rahmenlehrplaene/gymnasiale_oberstufe/curricula/2011/Informatik-VRLP_GOST_2011_Brandenburg.pdf, Zuletzt geprüft am 30.11.2016.
- [47] MORÉ, JORGE J: *The Levenberg-Marquardt algorithm: implementation and theory*, 1978.
- [48] MOTO-OKA, TOHRU und STONE, HAROLD S.: *Fifth-Generation Computer Systems - A Japanese Project*. <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1659076>, Zuletzt geprüft am 5.04.2017.
- [49] MUNSTERMANN, MARCO: *Technisch unterstützte Pflege von morgen - Innovative Aktivitätserkennung und Verhaltensermittlung durch ambiente Sensorik*. Springer Vieweg, Wiesbaden, 2014.
- [50] PAOLO MARRONE AND THE JOONE TEAM: *Joone - Java Object Oriented Neural Engine*. <http://www.jooneworld.com/>, Zuletzt geprüft am 6.04.2017.
- [51] RAUNER, M., & SCHRÖDER-T.: *Künstliche Intelligenz - Die Cogs kommen*. <http://www.zeit.de/zeit-wissen/2015/02/kuenstliche-intelligenz-cognitive-computing-cogs>, Aktualisiert am 25.2.2015, zuletzt geprüft am 5.09.2016.
- [52] RAV: *Computerprogramm gaukelt erfolgreich Menschsein vor*. <http://www.zeit.de/wissen/2014-06/kuenstliche-intelligenz-turing-test>, Zuletzt geprüft am 5.09.2016.
- [53] RENGER, DR. ANKE, KAPP, ANDREAS UND LISDAT, PROF. DR. FRED: *Aspirin & Co - Medikamenten-Rückstände im Wasser*. Schülerlabor EcoLab, Projekt NaWiTex, Technische Hochschule Wildau.
- [54] RÖFER, DR.-ING. DIPL.-INFORM. THOMAS: *Kognitive Robotik*. <http://www.informatik.uni-bremen.de/~roeper/kr02/02s.pdf>, Zuletzt geprüft am 7.11.2016.
- [55] SCHULZ, PETER: *RoboCup2016*. https://upload.wikimedia.org/wikipedia/commons/1/18/SPL_Team_B-Human%2C_RoboCup_2016.jpg, Zuletzt geprüft am 5.10.2016.
- [56] SENATSVERWALTUNG FÜR BILDUNG, JUGEND UND SPORT BERLIN: *Rahmenlehrplan für die gymnasiale Oberstufe*. https://www.berlin.de/sen/bildung/unterricht/faecher-rahmenlehrplaene/rahmenlehrplaene/mdb-sen-bildung-unterricht-lehrplaene-sek2_informatik.pdf, Zuletzt geprüft am 30.11.2016.
- [57] SENATSVERWALTUNG FÜR BILDUNG, JUGEND UND SPORT BERLIN: *Rahmenlehrplan für die gymnasiale Oberstufe - Informatik*. http://www.berlin.de/sen/bildung/unterricht/faecher-rahmenlehrplaene/rahmenlehrplaene/mdb-sen-bildung-unterricht-lehrplaene-sek2_informatik.pdf, Zuletzt geprüft am 31.01.2017.
- [58] SENATSVERWALTUNG FÜR BILDUNG, JUGEND UND WISSENSCHAFT: *Die fünfte Prüfungskomponente im Abitur - Eine Handreichung*. https://www.berlin.de/sen/bildung/schule/pruefungen-und-abschluesse/abitur/handreichung_5pk.pdf, Zuletzt geprüft am 12.04.2017.

- [59] SEVARAC, ZORAN: *Improving Neuroph Performance*. http://neuroph.sourceforge.net/improving_performance.html, Zuletzt geprüft am 10.12.2016.
- [60] SEVARAC, ZORAN: *Neuroph - Java Neural Network Framework*. <http://neuroph.sourceforge.net/index.html>, Zuletzt geprüft am 10.12.2016.
- [61] SIEPERMANN, DR. MARKUS: *Inferenzmaschine*. <http://wirtschaftslexikon.gabler.de/Definition/inferenzmaschine.html>, Zuletzt geprüft am 2.11.2016.
- [62] SMARTHOME INITIATIVE DEUTSCHLAND E.V.: *Was ist ein SmartHome?* <http://www.smarthome-deutschland.de/en/initiative/ziele-und-hintergruende.html>, Zuletzt geprüft am 28.10.2016.
- [63] SOFTBANK ROBOTICS: *Projet Romeo*. <http://projetromeo.com/>, Zuletzt geprüft am 29.10.2016.
- [64] STANGL, WERNER: *Deep Learning*. <http://lexikon.stangl.eu/12367/deep-learning/>, Zuletzt geprüft am 09.03.2017.
- [65] STATISTA: *Wie möchten Sie im Alter von 70 Jahren wohnen?* <https://de.statista.com/statistik/daten/studie/170431/umfrage/gewuenschte-wohnformen-der-generation-50-plus-mit-70-jahren/>, Zuletzt geprüft am 10.11.2016.
- [66] STATISTISCHES BUNDESAMT WIESBADEN: *Die Generation 65+ in Deutschland*. http://ibk-kubia.de/IBK-Dateien/PDFs/Nachrichten/Pressebrochure_generation65.pdf, Zuletzt geprüft am 10.11.2016.
- [67] STUDIENKREIS GMBH: *Brainstorming - wenn die Gedanken wirbeln*. <https://www.studienkreis.de/infothek/lerntipps/brainstorming/>, Zuletzt geprüft am 4.04.2017.
- [68] STUDYLIB: *E-Assessment mathematischer Beweise*. <http://studylibde.com/doc/2376292/>, Zuletzt geprüft am 22.10.2016.
- [69] THE APACHE SOFTWARE FOUNDATION: *Apache Derby*. <https://db.apache.org/derby/index.html>, Zuletzt geprüft am 6.03.2017.
- [70] THE USB PORT: *Google Self Driving Car*. <http://theusbport.com/wp-content/uploads/2016/03/google-self-driving-car-repaint-1200x615.jpg>, Zuletzt geprüft am 5.10.2016.
- [71] ULLENBOOM, CHRISTIAN: *Java SE 8 Standard-Bibliothek: Das Handbuch für Entwickler*. Galileo Press, Bonn 2014, Bonn, 2. Auflage, 2014.
- [72] UNI FREIBURG: *Lernstrategien für Neuronale Netze - Der Backpropagation-Algorithmus*. http://lmb.informatik.uni-freiburg.de/lectures/mustererkennung/WS0506/08_b_ME.pdf, Zuletzt geprüft am 24.03.2017.
- [73] VON SEGBUSCH, DR.-ING. KLAUS und STRUWE, CHRISTIAN: *Smart-Home als Basis für Smart-Grids*. <http://www.etz.de/935-0-Smart+Home+als+Basis+fuer+Smart+Grids.html>, Zuletzt geprüft am 28.10.2016.
- [74] W3SCHOOLS.COM: *HTML Responsive Design*. http://www.w3schools.com/html/html_responsive.asp, Zuletzt geprüft am 18.01.2017.

- [75] WALLACH, HANNA M.: *Conditional Random Fields: An Introduction*). <http://dirichlet.net/pdf/wallach04conditional.pdf>, Zuletzt geprüft am 16.12.2016.
- [76] WETTERICH, FRANK, BURGHART, MARTIN und RAVE, NORBERT: *Medienbildung an deutschen Schulen*. http://www.initiaved21.de/wp-content/uploads/2014/11/141106_Medienbildung_Onlinefassung_komprimiert.pdf, Zuletzt geprüft am 21.11.2016.