technische universität dortmund

IT in Produktion und Logistik

Bachelorarbeit

Untersuchung eines Open-Source-Laufroboters in Bezug auf Programmierbarkeit und Einsatz in der Lehre

Technische Universität Dortmund

Fakultät Maschinenbau

Fachgebiet IT in Produktion und Logistik

Gutachter/in:	Dr.-Ing. Dipl. Inform. Anne Antonia Scheidler
	M. Sc. Nils Heidemann
Vorgelegt von:	Leon Ertural
Matrikel-Nummer:	175840
Datum der Abgabe:	02.03.2020

Inhaltsverzeichnis

Inhaltsverzeichnis I

Abbildungsverzeichnis III

Tabellenverzeichnis III

Formelverzeichnis VI

Abkürzungsverzeichnis VII

1. Einleitung 1

2. Grundlagen und Gebrauch der vorliegenden Mikroelektronik und Robotik 3

2.1 Grundbestandteile des Roboters 3

2.1.1 Aufbau des Lokomotion-Controllers 3

2.1.2 Funktionsweise der Servomotoren 5

2.1.3 Beschreibung der kabellosen Schnittstellen und Verbindungen 7

2.1.4 Energieversorgung des Roboters 8

2.2 Grundriss der Softwarearchitektur 9

2.2.1 Informative Inhalte des Downloadpakets 10

2.2.2 Umsetzung relevanter Kommunikationsprotokolle 12

2.3 Bewegungssteuerung des Laufroboters 17

2.4 Erweiterungsmöglichkeiten des Roboters mit zur Verfügung stehender Ausrüstung 20

2.4.1 Eigenschaften eines Arduino UNO-Mikrocontroller-Boards 20

2.4.2 Eigenschaften eines NodeMCU-Mikrocontroller-Boards 21

2.4.3 Eigenschaften eines Raspberry Pi-Computers 22

2.4.4 Aufbau und Funktionsweise des Ultraschallsensors 23

2.4.5 Beschreibung der Tiefenkamera 24

2.4.6 Erweiterungen mittels additiver Fertigungsverfahren 25

3. Vorstellung des Fachlabors und vergleichbarer Lehrkonzepte 26

4. Implementierungsmöglichkeiten des Laufroboters in das Lehrkonzept 28

4.1 Konstruktion eines Überwachungsroboters 28

4.1.1 Formulierung der Problemstellung 29

4.1.2 Vorstellung einer möglichen Lösung 29
4.1.3 Reflektion des Beispiels 34
4.2 Konstruktion einer bildverarbeitenden Laufmaschine 38
4.2.1 Formulierung der Problemstellung 38
4.2.2 Vorstellung einer möglichen Lösung 39
4.2.3 Reflektion des Beispiels 41
4.3 Konstruktion eines autonomen Geländeroboters 42
4.3.1 Formulierung der Problemstellung 43
4.3.2 Vorstellung einer möglichen Lösung 44
4.3.3 Umsetzung der Aufgabenstellung 50
4.3.4 Reflektion des Beispiels 57
5. Zusammenfassung und Ausblick 60
Literaturverzeichnis 62
Anhang 67

Abbildungsverzeichnis

Abbildung 1: Lokomotion-Controller-Board 4

Abbildung 2: Regelkreis eines Modellbauservos 6

Abbildung 3: PWM-Signal 7

Abbildung 4: Verbindung zweier UART-Bausteine 13

Abbildung 5: Aufbau einer einfachen seriellen Verbindung 14

Abbildung 6: Aufbau des Daten-Arrays der UART-Verbindung (Conrad Electronics SE 2020) 15

Abbildung 7: Grundschaltung eines I2C-Busses (Brühlmann 2019, S. 136) 16

Abbildung 8: I2C-Implementierung bei Master- und Slave-Modul (nach Brühlmann 2019, S. 138f.) 16

Abbildung 9: Gelenke und Glieder des Laufroboters (Conrad Electronic SE 2020) 17

Abbildung 10: Tripod-Laufmuster 19

Abbildung 11: Wave-Laufmuster 19

Abbildung 12: Ripple-Laufmuster 19

Abbildung 13: Aufbau des Mikrocontrollers (nach Wüst 2011, S. 119) 21

Abbildung 14: Erweiterungen auf Lokomotion-Controller montiert (Makerfactory 2020b) 23

Abbildung 15: Anschluss des Raspberry Pi an der Universalanschlussleiste des Lokomotion-Controller-Boards (Fritzing-Darstellung nach Conrad Electronic SE 2020) 31

Abbildung 16: Elemente der Weboberfläche 34

Abbildung 17: Einsehbares Umfeld der drei Ultraschallsensoren des Roboters 46

Abbildung 18: Grundriss eines beispielhaften Hindernislaufs 49

Abbildung 19: Initialisieren der seriellen Schnittstellen 51

Abbildung 20: Montage der Halterungen und Ultraschallsensoren 52

Abbildung 21: Start und Stopp durch Tasterabfragen 54

Abbildung 22: Kletterfunktion 55

Abbildung 23: Hindernisparcours aus Kartonage 57

Tabellenverzeichnis

Tabelle 1: Übersicht der Board-Anschlüsse 5

Tabelle 2: Beispielprogramme der mitgelieferten Betriebssoftware 11

Formelverzeichnis

Formel 1: Formel zur Abstandsberechnung (nach Brühlmann 2019, S. 234) 24

Abkürzungsverzeichnis

API	Programmierschnittstelle
CAD	Computer Aided Design
DSI	Display-Schnittstellensteckverbinder
FPS	Bildrate
FSR	Kraftempfindliche Widerstände
GPIO	General-Purpose-Input/Output-Pins
I2C	Inter-Integrated Circuit
ICSP	In-Circuit Serial Programming
IDE	Integrierte Entwicklungsumgebung
ITPL	Fachgebiet IT in Produktion und Logistik
PC	Persönlicher Computer
PWM	Pulsweitenmoduliertes Signal
SBC	Einplatinencomputer
SCL	Taktleitung
SDA	Datenleitung
SDK	Softwareentwicklungspaket
SLAM	Simultaneous Localization and Mapping
SoC	Ein-Chip-System
SPI	Serial Peripheral Interface
SuS	Schülerinnen und Schüler
TU	Technische Universität Dortmund
UART	Universal Asynchronous Receiver Transmitter

1. Einleitung

Aus der heutigen Produktion und Logistik ist die Automatisierung und Robotik nicht mehr wegzudenken. Als logische Konsequenz müssen auch Hochschulen ihr Lehrangebot dahingehend anpassen, dass ihre Studierenden nach Beendigung ihrer universitären, beziehungsweise fachhochschulischen, Ausbildung über ein grundlegendes Wissen, speziell im Bereich der Automatisierungstechnik, verfügen. Gerade in diesen Bereichen werden häufig Einplatinencomputer mit verbauten Mikrocontrollern und -prozessoren eingesetzt (Abel und Bollig 2006). Um den Studierenden ein ausgeprägtes, fundiertes Verständnis von Kompetenzen auf diesem Gebiet vermitteln zu können, eignet sich ein auf praktische Anwendungen ausgerichtetes Lehrangebot (Jara u. a. 2011). Hiermit konforme Lehrveranstaltungen setzen einen gewissen Bestand an technischer Ausstattung voraus. Als Beispiele hierfür sind Industrieroboter oder andere Automatisierungssysteme mit verbauten Mikrocontrollern oder -prozessoren zu nennen, mit denen dann die Studierenden, neben dem Erlernen von grundlegenden Fertigkeiten in Programmierung und Robotik, auch eigene Projekte, beispielsweise im Rahmen von Fachlaboren oder Projektarbeiten, umsetzen können. Aus diesem Grund hat das Fachgebiet IT in Produktion und Logistik (ITPL) der Technischen Universität Dortmund (TU) einen Laufroboter mit sechs Beinen angeschafft. Diese Arbeit befasst sich zudem mit den möglichen Problemen und Grenzen, auf die Studierende im Rahmen einer praktisch orientierten Lehrveranstaltung stoßen können. Die sich durch die Verknüpfung von Theorie und Praxis und aus der Kombination der Forschungsfelder Mechanik, Elektrotechnik und Informatik ergebende Interdisziplinarität dieser und zukünftiger Arbeiten mit dem Laufroboter, ist ein weiterer motivierender Aspekt, der das Arbeiten mit diesem Roboter ausgeprägt interessant gestalten lässt.

Dieser Roboter soll im Rahmen der vorliegenden Arbeit untersucht werden. Dabei liegt der Fokus der Untersuchungen auf der Einsetzbarkeit des Laufroboters im Lehrbetrieb oder genauer: inwieweit sich der Gebrauch dessen eignet, um Studierenden aktuelle Themen der Automatisierungstechnik näher zu bringen. Der Bausatz des Laufroboters, der bereits vor Beginn der Untersuchungen montiert worden ist, stammt von der Firma Makerfactory und wird exklusiv von der Conrad Electronic SE vertrieben. Zusätzlich zu dem Roboter hat das Fachgebiet einen neuen 3D-Drucker angeschafft, der auch Veränderungen der Erscheinungsform des Roboters durch das Drucken zusätzlicher Bauteile zulassen soll. Einige Konstruktionsideen zur Erweiterung befinden sich bereits in der mitgelieferten Dokumentation des Roboters (Conrad Electronic SE 2020). Die dadurch entstehenden Freiheiten ermöglichen den Studierenden bei der Arbeit mit dem Laufroboter diesen an die individuellen

Einsatzzwecke anzupassen. Durch diese Freiheiten soll der Laufroboter im späteren Einsatz die Kreativität der Studierenden in der Planung zusätzlicher Erweiterungen fördern. Neben Kenntnissen in den Bereichen Programmierung, Sensorik und Aktorik lassen sich Kenntnisse in der Konstruktion von Bauteilen erlernen und praktisch anwenden. Die grundlegenden Untersuchungen am Roboter sollen Erkenntnisse darüber liefern, inwieweit sich der Laufroboter in seiner Soft- und Hardware modifizieren lässt, um so seine Verwendbarkeit im Lehrbetrieb fundiert einschätzen zu können.

Um dieses Ziel zu erreichen, muss zunächst die Dokumentation begutachtet, die Ansteuerbarkeit der Aktoren geprüft und die Kompatibilität des Roboters zu Mikrocontrollern und anderen Einplatinencomputern (SBC) eingeschätzt werden. Um deren Programmierungsmöglichkeiten nachvollziehen zu können, sollen die relevanten Sensoren und Aktoren hinsichtlich ihres Aufbaus und ihrer Funktionsweisen beschrieben werden. Mit den Erkenntnissen dieser Vorbereitungen werden dann drei Konzepte ausgearbeitet, die eine wissenschaftlich sinnvolle Modifikation des Roboters in Soft- als auch Hardware vorsehen und die Funktionen des Roboters erweitern. Die ausgearbeiteten Konzepte sollen in der Lage sein, dem Bearbeitenden gezielt Wissen über möglichst unterschiedliche Themen der Automatisierungstechnik, beispielsweise über Sensorik und Aktorik, praktisch zu vermitteln. Eines dieser erstellten Konzepte soll im Anschluss auch praktisch umgesetzt werden. Das ausgewählte Konzept soll in seinem Umfang dem Anspruch einer Abschlussarbeit entsprechen und möglichst komplexe Funktionsweisen realisieren. Die drei Ausarbeitungen werden dann in Hinblick auf ihre Umsetzbarkeit, ihre Einsetzbarkeit in der Praxis und die Komplexität und Diversität des durch sie vermittelnden Wissens bewertet. Diese Arbeit soll damit abschließen, in Bezug auf die Ergebnisse der Untersuchungen und ausgearbeiteten, beziehungsweise umgesetzten Konzepte zur Erweiterung des Roboters eine begründete Aussage über die Verwendbarkeit des Laufroboters im Lehrbetrieb, speziell in Fachlaboren, zu treffen und gegebenenfalls ein genaues Vorgehenskonzept zu empfehlen.

2. Grundlagen und Gebrauch der vorliegenden Mikroelektronik und Robotik

Um den Laufroboter hinsichtlich seiner Verwendbarkeit im Lehrbetrieb begründet bewerten zu können, muss dieser zunächst genauer untersucht werden. Grundlage dieser Untersuchungen ist unter anderem die Dokumentation des Herstellers. Ein entsprechendes Downloadpaket mit der Bedienungsanleitung und zahlreichen zusätzlichen Inhalten kann online im Conrad-Downloadcenter kostenlos heruntergeladen werden (Conrad Electronic SE 2020). Darüber hinaus wird hier ein Überblick der, mit der Programmierung von Mikrocontrollern-basierenden Laufrobotern benötigten Technik und einhergehenden Kompetenzen geboten. Im Folgenden wird, bevor die Funktionsweisen und mögliche Erweiterungen des Roboters näher erläutert werden, der strukturelle Aufbau des Laufroboters analysiert. Dazu gehören neben den physischen Komponenten (der Hardware) auch alle nicht-physischen, programmierbaren Bestandteile des Roboters. Bei Letzteren wird insbesondere auf die Struktur der vom Hersteller mitgelieferten Betriebssoftware des Roboters eingegangen.

2.1 Grundbestandteile des Roboters

Hier werden die Grundbestandteile der Hardware des Roboters und deren Funktionsweisen beschrieben. Zu den Grundbestandteilen zählen alle Bauteile, die für die Basisfunktionen des Roboters zwingend benötigt werden und auch Bestandteil des angeschafften Bausatzes sind. Der Lieferumfang beinhaltet neben dem Roboter und dessen Dokumentation noch ein *Gamepad* mit Funkschnittstelle für eine kabellose Steuerung (Makerfactory 2020b). Die beigelegte Dokumentation beinhaltet eine Aufbauanleitung, Schaltpläne sowie einige Programmbeispiele, durch welche der Roboter sofort betriebsbereit wird. Bei dem Roboter handelt es sich um einen sogenannten *Open-Source-Roboter*. Unter *Open Source* wird grundsätzlich Software verstanden, deren Quelltext vom Autor veröffentlicht worden ist und gebührenfrei verwendet werden kann (DiBona und Ockman 1999). In dem Fall des Roboters bietet der Hersteller diesen Quelltext im Internet zur freien Verfügung an (Conrad Electronic SE 2020). Diese Quelloffenheit ermöglicht dem Endverbraucher diverse Möglichkeiten, das Produkt an seine spezifischen Bedürfnisse anzupassen.

2.1.1 Aufbau des Lokomotion-Controllers

Als Lokomotion-Controller wird grundsätzlich die Bewegungssteuerung der Fortbewegung eines Laufroboters bezeichnet. Dieser stellt somit das Kernelement eines laufenden Roboters dar. In diesem konkreten Fall handelt es sich bei dem Lokomotion-Controller um ein Mikrocontroller-Board.

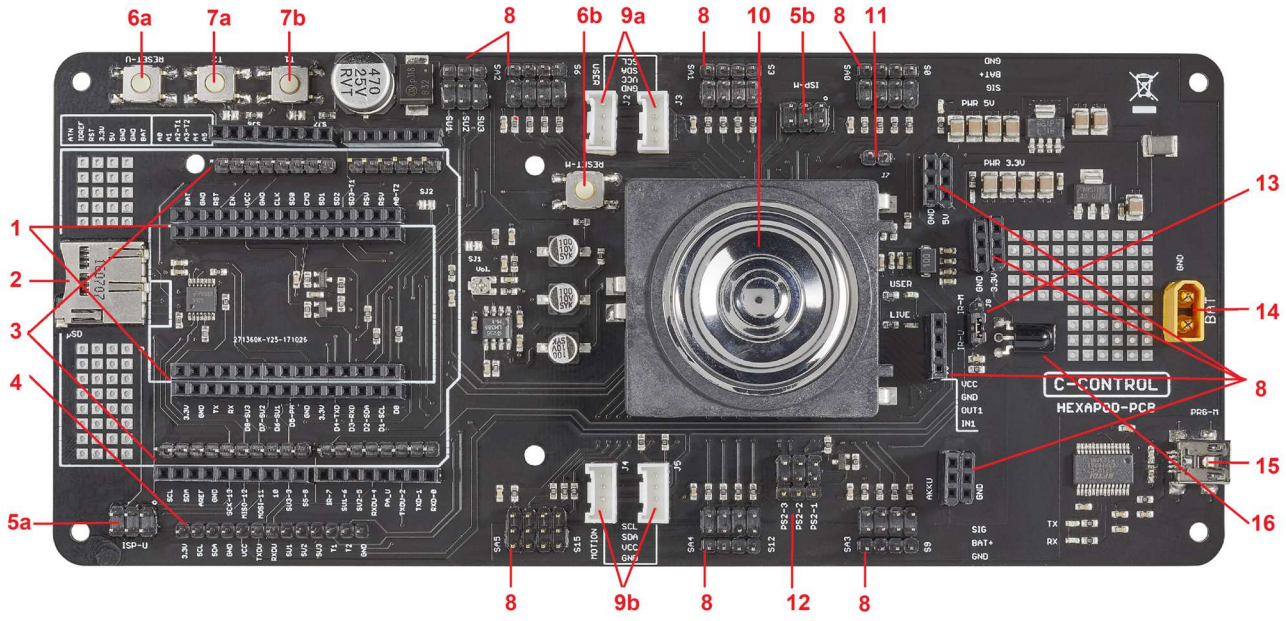

Abbildung 1: Lokomotion-Controller-Board

Dieses Board ist in Abbildung 1 dargestellt. Der dem Board zu Grunde liegende 8-Bit Mikrocontroller ist ein Atmel ATmega2560 mit 86 Input/Output-Anschlüssen (Atmel 2014) und befindet sich direkt unter dem Lautsprecher. Dieser Controller ist dafür bekannt, auch im Arduino MEGA 2560 verbaut zu sein, sodass der Lokomotion-Controller und seine Betriebssoftware, deren Bestandteile in Kapitel 2.2.1 näher beschrieben sind, mit diesem Board kompatibel ist (Conrad Electronic SE 2020). Des Weiteren besitzt der Lokomotion-Controller zahlreiche Anschlüsse und Schnittstellen, die in Tabelle 1 mit einer jeweiligen Kurzbeschreibung und mit Verweis auf Abbildung 1 aufgeführt sind. Die Leiterplatte ist so ausgelegt, dass an ihr eigene Erweiterungen in Form von zusätzlichen Servomotoren und Sensoren an den Controller angeschlossen werden können. Die zahlreichen freien Steckerleisten ermöglichen außerdem die Verbindung diverser Mikrocontroller und anderer Einplatinencomputer. Mit welchen, vom Benutzer ausgewählten Boards (den Benutzer-Boards) solche Verbindungen realisiert werden können, wird in Kapitel 2.4 näher erläutert.

Elementnummer	Beschreibung
1	Anschlussleiste zum Verbinden eines NodeMCU ESP8266 CP2102 Mikrocontroller-Boards.
2	microSD-Speicherkartenlesegerät, das nur in Verbindung mit einem Arduino UNO verwendbar ist. Hiermit können zum Beispiel Sensordaten von einer Speicherkarte gelesen oder auf diese geschrieben werden.
3	Anschlussleiste zum Verbinden eines Arduino UNO Mikrocontroller-Boards.
4	Universalanschlussleiste zum Verbinden von Erweiterungen.
5a 5b	In-System-Programmierung-Schnittstelle des angeschlossenen Benutzer-Boards. In-System-Programmierung-Schnittstelle des Lokomotion-Controllers.
6a 6b	Reset-Taster des Lokomotion-Controllers startet das Programm des Controllers neu. Reset-Taster des angeschlossenen Benutzer-Boards startet das Programm des Boards neu.
7a 7b	Frei programmierbarer Taster T2 bei angeschlossenen Benutzer-Board. Frei programmierbarer Taster T1 bei angeschlossenen Benutzer-Board.
8	Frei belegbare Anschlüsse für zusätzliche Elektronik, beispielsweise Sensoren oder Aktoren
9a 9b	Inter-Integrated-Circuit-Schnittstelle des Benutzer-Boards ermöglicht serielle Kommunikation mit mehreren externen Geräten gleichzeitig. Inter-Integrated-Circuit-Schnittstelle des Lokomotion-Controllers ermöglicht serielle Kommunikation mit mehreren externen Geräten gleichzeitig.
10	Lautsprecher mit Verstärkerschaltung zur Sprach- und Tonausgabe.
11	Jumper J7 (falls verbunden) deaktiviert die Steuerung des Lokomotion-Controllers über das Gamepad. Gamepad-Steuerbefehle sind dann lediglich durch ein angeschlossenes Benutzer-Board auslesbar.
12	Anschlüsse für die Steuerung mit Hilfe eines Playstation-2-kompatiblen Gamepads, beziehungsweise dessen Empfänger.
13	Jumper J8 verbindet den Infrarotempfänger des Boards entweder mit dem Lokomotion-Controller oder dem angeschlossenen Benutzer-Board.
14	XT30-Stecker zum Anschluss der Stromversorgung.
15	USB-Anschluss zum Übertragen der Betriebssoftware des Lokomotion-Controllers oder zur eigenen Programmierung dessen.
16	38 kHz Infrarotempfänger, der mit oder ohne angeschlossenen Arduino UNO verwendbar ist (siehe Nummer 13).

Tabelle 1: Übersicht der Board-Anschlüsse

2.1.2 Funktionsweise der Servomotoren

Im Allgemeinen gelten alle Schaltelemente einer Schaltung, die eine Eingangsgröße in eine Ausgangsgröße umwandeln, als Aktoren und sind somit in Regelsystemen häufig die Stellglieder, die

eine Aktion erzeugen (Brühlmann 2019). Diese Aktionen sind meist Schaltvorgänge, die Motoren, Relais oder Ventile schalten und begegnen uns oft im Alltag. Servomotoren gelten somit als Aktoren (Brühlmann 2019). Als Servo wird ein Bauteil mit verbautem Elektromotor (meist Gleichstrom) samt Steuerungs- oder Regelungselektronik bezeichnet (Brühlmann 2019). Im Zusammenhang mit Arduino-Projekten finden, wie auch bei diesem Roboter der Fall, häufig Modellbauservos Verwendung. Diese Servos wandeln eine ihnen übermittelte elektrische Information in eine mechanische Bewegung um und bestehen nach Orton (1990) aus mehreren Komponenten:

- Zahnradgetriebe
- Elektromotor
- Leistungselektronik zur Ansteuerung des Motors
- Elektronik für Pulsauswertung und -generation
- Potentiometer

Mit Hilfe dieser Komponenten wird ein einfacher Regelkreis konstruiert (Abbildung 2). Angeschlossen an einen Mikrocontroller, wird einem Servo ein Puls mit Information über die zu stellende Position, in der Regel in Form eines pulsweitenmodulierten Signals (PWM), übertragen. Im Falle der Modellbauservos ist dies ein Steuersignal mit einer Periodenlänge von 20 ms, bei dem Pulsbreiten von 0,5 bis 2,5 ms auftreten können (Abbildung 3). Mit dem Senden der mittleren Pulsbreite (1,5 ms) wird die Mittelstellung des Servomotors erreicht (oft die 0-°-Position). Ist die übertragene Pulsbreite größer als 1,5 ms dreht der Servo in die eine Richtung, ist sie kleiner, dreht der Servo in die entgegengesetzte Richtung (Conrad Electronic SE 2020).

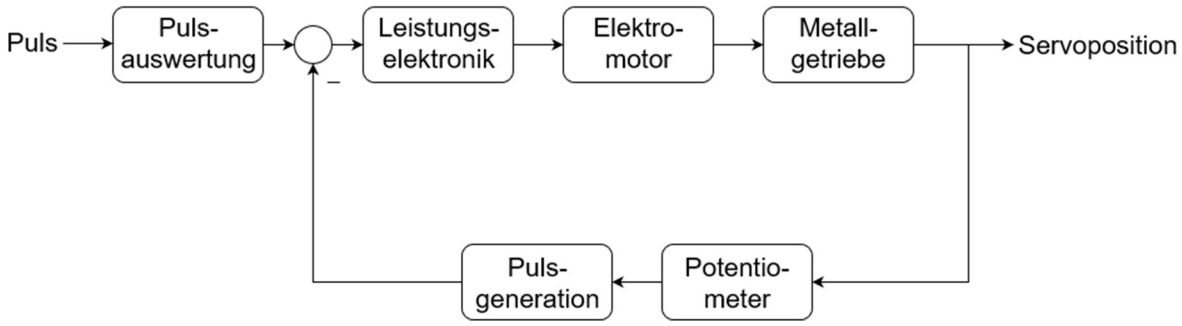

Abbildung 2: Regelkreis eines Modellbauservos

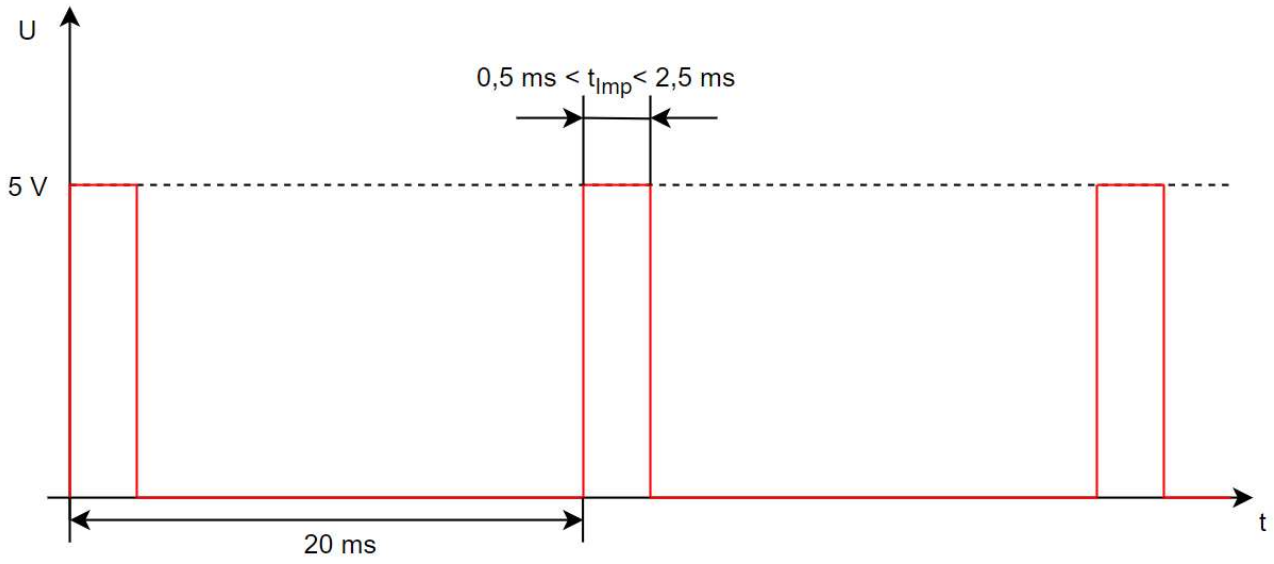

Abbildung 3: PWM-Signal

Der Drehwinkel ist also proportional zur Pulsbreite und beträgt minimal -90 ° und maximal +90 ° zur Mittelstellung, da Modellbauservos typischerweise eine maximale Gesamtrotation von 180 ° zulassen. Um zu garantieren, dass die gewünschte Position des Servos tatsächlich erreicht wird, ist, wie in Abbildung 2 dargestellt, die Servoposition über ein Potentiometer rückgekoppelt. Das Potentiometer, welches als Winkelgeber dient, liest die Ist-Position des Servos und die Elektronik des Servos vergleicht die generierte Pulsbreite mit der vom Mikrocontroller vorgegebenen Soll-Position und justiert gegebenenfalls den Elektromotor nach (Conrad Electronic SE 2020).

2.1.3 Beschreibung der kabellosen Schnittstellen und Verbindungen

Der Roboter kann auf viele verschiedene Arten gesteuert werden. Neben der bereits erwähnten Möglichkeiten mittels Gamepad oder Benutzer-Boards den Roboter zu kontrollieren, können auch Steuerbefehle durch Nutzung der im Lokomotion-Controller-Board verbauten Infrarotschnittstelle empfangen werden. Soll mit einem passendem Infrarottransmitter eine drahtlose Verbindung zu dem Lokomotions-Controller geschaffen werden, so muss der Jumper J8 in Richtung „IR-M“ gesteckt werden, damit der Infrarotempfänger mit dem Lokomotion-Controller verbunden ist. Für den Fall, dass ein passender Transmitter mit einem angeschlossenen Arduino UNO kommunizieren soll, so muss derselbe Jumper in die Richtung mit der Aufschrift „IR-U“ gesteckt werden, sodass der Empfänger mit dem Benutzer-Board verbunden wäre (Conrad Electronic SE 2020). Das Einbinden der

Bibliothek „IRremote“ in den Programmcode des jeweiligen Controllers ermöglicht es dann, mit den in dieser Bibliothek definierten Methoden in Abhängigkeit der gesendeten Daten Funktionen, beispielsweise Bewegungsbefehle, aufzurufen. Die Daten werden nach dem RC-5-Protokoll übertragen, welches auch Verwendung in Fernseherfernbedienungen findet (Conrad Electronic SE 2020). Diese Infrarotschnittstelle ist nicht die einzige Möglichkeit den Roboter über eine kabellose Verbindung zu steuern. Das im Lieferumfang enthaltene Gamepad sendet ebenso Daten kabellos an den unter dem Lokomotion-Controller-Board befestigten Empfänger. Das Gamepad sendet Daten über die Zustände der Hebel (X-Y-Position) und Taster (gedrückt/nicht gedrückt) über eine 2,4 GHz Funkverbindung an seinen Empfänger. Dieser gibt dann die Daten über die Anschlüsse PS2-1, PS2-2 und PS3-3 an den Lokomotion-Controller weiter. Die für die Auswertung der Daten benötigte Bibliothek „PS2X_lib“ befindet sich ebenfalls im Downloadpaket (Conrad Electronic SE 2020). In der Bibliothek „Data_Input“ werden dann, abhängig von den betätigten Hebeln und Tastern, die einzelnen Bewegungsbefehle aufgerufen. Für den Fall, dass Jumper J7 angeschlossen ist, wird die Verbindung von Lokomotion-Controller und Gamepad-Empfänger getrennt (Conrad Electronic SE 2020). Der Gamepad-Empfänger gibt die Steuerdaten dann an das Benutzer-Board weiter, sofern eins angeschlossen ist. Somit lassen sich auch eigene Programme auf dem Benutzer-Board mit dem Gamepad steuern. Um die Programmierung einer solchen Funktion zu erleichtern, sollte das in Kapitel 2.2.1 beschriebene Beispielprogramm des Downloadpakets betrachtet werden.

2.1.4 Energieversorgung des Roboters

Die korrekte Stromversorgung des Roboters ist für die Sicherheit und einen störungsfreien Betrieb von fundamentaler Bedeutung (Conrad Electronic SE 2020). Grundsätzlich ist der Batterieanschluss des Lokomotion-Controller-Boards für eine Versorgungsspannung zwischen 4,5 und 10 V Gleichstrom ausgelegt. Wie genau der Roboter idealerweise betrieben wird, bleibt eine Einzelfallentscheidung. Entsprechende Modifikationen am Roboter, beispielsweise der Einbau von Hochleistungsservos aus dem Modellbau, führen dazu, dass sich die ideale Versorgungsspannung und Art der Batterie verändert. Sind aber an dem Board hauptsächlich klassische Modellbauservos mit einer Stellkraft von unter 8 kg angeschlossen (das Gewicht, das mit einem Hebel von 1 cm durch den Servo angehoben werden kann), so liegt die optimale Versorgungsspannung bei 6 V (Conrad Electronic SE 2020). Daher wird im Datenblatt des Lokomotion-Controller-Boards für den Standardbetrieb ein 5-zelliges NiMH-Akkupack mit mindestens 2000 mAh als primäre Energiequelle empfohlen. Bei Betrieb mit Akkupacks mit geringerer Kapazität besteht die Gefahr von zu kurzen Laufzeiten des Roboters oder Ausfällen

der Stromversorgung und damit einem Reset des Lokomotion-Controllers und den eventuell angeschlossenen Benutzer-Boards (Conrad Electronic SE 2020). Bei der Erweiterung des Roboters sollte der Zuwachs an Gewicht und die damit einhergehende höhere benötigte Leistung der Servomotoren bedacht werden. Angeschlossen wird das Akkupack über einen Stecker des Typs XT30. Sollte die Laufzeit als zu gering erscheinen, kann ein angeschlossenes Benutzer-Board, bei dem eine hinreichende Stromversorgung kritisch ist, auch über eine zweite Energiequelle, beispielsweise einer 9-Volt-Blockbatterie oder *Powerbank*, betrieben werden. Im Besitz des Fachgebiets befindet sich ein NiMH-Akkupack mit einer Spannung von 6 V und einer Kapazität von 3700 mAh. Ein dazu passendes Ladegerät ist ebenfalls vorhanden. Es wird dringend geraten, den Roboter mit diesem modellbautypischen Akkupack zu betreiben, um die oben genannten Risiken zu vermeiden (Conrad Electronic SE 2020). Der Akku wird aus Sicherheitsgründen unter dem Lokomotion-Controller-Board, also im Grundgerüst des Roboters, mit Klettverschlüssen fixiert, damit es zu keinen Behinderungen im Bewegungsablauf kommt.

2.2 Grundriss der Softwarearchitektur

Während weiter oben auf die technischen Bestandteile eingegangen wird, liegt der Fokus hier nun auf dem Aufbau der programmierbaren Bestandteile des Roboters, also der Software. Aufgrund des mikrocontrollerbasierten Aufbaus des Laufroboters, ist die Programmierung des Roboters mit der zur freien Verfügung stehenden, auf der Programmiersprache *Processing* basierenden Arduino Software (Banzi und Shiloh 2014) möglich und aufgrund der Simplizität ratsam. Innerhalb dieser integrierten Entwicklungsumgebung (IDE) wird mit einer vereinfachten Form der Programmiersprache *C* und *C++* programmiert, was das Erlernen der benötigten Fertigkeiten erleichtert (Brühlmann 2019). Innerhalb dieser IDE lassen sich Programmcodes schreiben, verändern und kompilieren. Nach dem erfolgreichen Kompilieren des Codes, lässt sich dieser mühelos über eine USB-Schnittstelle des Computers auf den Mikrocontroller des Roboters oder einem an diesen angeschlossenen Benutzer-Board laden. Die Arduino IDE ermöglicht den Zugriff auf eine Vielzahl an Bibliotheken, die selektiv in das geschriebene Programm (auch *Sketch* genannt) eingebunden werden können. Diese Bibliotheken erweitern die Funktionen des Programmes und ermöglichen beispielsweise das Arbeiten mit bestimmten Hardwarekomponenten oder das manipulieren von Daten (Arduino 2020a) und vereinfachen die Programmierung dadurch erheblich. Um dem Endverbraucher die Programmierung des Roboters zu erleichtern, bietet dessen Hersteller grundlegende Softwarebibliotheken im Sinne

der Definition von *Open Source* durch Bruce Perens (1999) zur freien Verfügung an (Conrad Electronic SE 2020).

2.2.1 Informative Inhalte des Downloadpakets

In den oben erwähnten Bibliotheken befinden sich kleinere Beispielprogramme für den Betrieb des Roboters mit oder ohne einem angeschlossenen Benutzer-Board sowie die Lokomotion-Betriebssoftware des Lokomotion-Controllers, welche, nachdem sie auf den Controller übertragen wurde, den Roboter bereits bewegungsfähig und mit dem ebenfalls mitgelieferten Gamepad steuerbar macht. Da sich das Lokomotion-Controller-Board in seinen Bauelementen jedoch nur minimal von denen des Arduino MEGA unterscheidet, kann es auch wie ein solcher frei programmiert werden, ohne die vom Hersteller veröffentlichte Betriebssoftware zu verwenden. Um die eigene freie Programmierung des Roboters zu vereinfachen, liegt dem Downloadpaket eine Übersicht der Pinbelegungen der Leiterplatte und wie sie in der Arduino IDE anzusteuern sind bei. In Tabelle 2 sind die Funktionen ausgewählter Beispielprogramme aufgeführt, die das Nutzen von Benutzer-Boards veranschaulichen und dem Benutzer ein Einbinden ähnlicher Funktionen in sein eigenes Programm erleichtern sollen (Makerfactory 2020a). Die hier getroffene Auswahl der insgesamt 19 mitgelieferten Demoprogramme beschränkt sich auf die Programme, die Kompetenzen von essentieller Bedeutung für die Programmierung der Erweiterungen mit Benutzer-Boards vermitteln. Während auf dem Lokomotion-Controller lediglich die Betriebssoftware aufgespielt wird, können die möglichen Erweiterungen (zusätzliche Mikrocontroller oder andere Einplatinencomputer) frei programmiert werden und, sofern korrekt angeschlossen, mit dem Lokomotion-Controller kommunizieren und Befehle der Betriebssoftware aufrufen. Um das Aufrufen verschiedener Funktionen der Betriebssoftware über einen Mikrocontroller zu vereinfachen, beinhaltet das Downloadpaket des Herstellers eine zusätzliche Bibliothek („Hexapod_Lib“). In dieser Bibliothek sind Steuerungsbefehle des Roboters bestimmte Werte zugeordnet, die dann mit der dort ebenfalls definierten Funktion „SendData()“ über eine durch *Universal Asynchronous Receiver Transmitter* (UART) realisierte, digitale Schnittstelle dem Lokomotion-Controller zugesendet werden und dort dann die, dem Wert entsprechenden Befehle aufruft. Über die UART-Schnittstelle, die in jedem geläufigen Mikrocontroller verbaut ist, kann das angeschlossene Benutzer-Board in der Motion-Betriebssoftware definierte Bewegungsbefehle an den Lokomotion-Controller weitergeben oder Statusabfragen über den Zustand des Roboters einholen.

Beispielprogramm	verwendete Erweiterung	Funktion
EspMoveWIFI	NodeMCU	Das Programm verbindet den Roboter über eine WiFi-Verbindung mit einem lokalen Netzwerk. Nach der Verbindung und der Eingabe einer ausgewiesenen IP-Adresse in einen Webbrowser lässt sich der Roboter über diesen Browser innerhalb des Netzwerkes steuern.
ReadGamepad	Arduino UNO oder NodeMCU	Dieses simple Programm gibt die Steuerdaten aus, die das Gamepad an den Lokomotion-Controller sendet. Ist der Jumper „J7“ aufgesteckt, so werden die Daten vom Funkempfänger nur vom Benutzer-Board empfangen und die Verbindung zum Lokomotion-Controller deaktiviert.
Moving_01	Arduino UNO oder NodeMCU	Das Programm zeigt, wie die Bewegungsbefehle auch ohne die Verwendung der mitgelieferten „Hexapod_Lib"-Bibliothek an den Lokomotion-Controller weitergegeben werden können. Es wird ein näheres Analysieren der hier erwähnten „SentData()"-Funktion empfohlen. Nach einem ähnlichen Muster können auch Einplatinencomputer programmiert werden, um den Lokomotion-Controller zu steuern.
Moving_02	Arduino UNO oder NodeMCU	Dieses Programm verwendet zur Ausführung der Standardbewegungen des Roboters die mitgelieferte „Hexapod_Lib"-Bibliothek. Ein näheres Betrachten dieses Programmes erleichtert die Programmierung eines angeschlossenen Benutzer-Boards.
ReadValues_01	Arduino UNO oder NodeMCU	Das Programm zeigt, wie Daten ohne Gebrauch der „Hexapod_Lib"-Bibliothek zwischen Lokomotion-Controller und Benutzer-Board über die UART-Schnittstelle gesendet werden können. Es wird ein näheres Analysieren der Dokumentation der hier erwähnten „SoftwareSerial“-Bibliothek empfohlen. Nach einem ähnlichen Muster können auch andere Einplatinencomputer programmiert werden, um mit dem Lokomotion-Controller zu kommunizieren.

Tabelle 2: Beispielprogramme der mitgelieferten Betriebssoftware

Dadurch, dass die Benutzer-Boards über die Kommunikationsschnittstelle so nur die Laufparameter, wie zum Beispiel Schritthöhe und -weite, Laufmuster und -geschwindigkeit oder Körperhöhe, an den

Lokomotion-Controller übermitteln müssen, kann sich das angeschlossene Benutzer-Board mit seiner begrenzten Rechenleistung beispielsweise auf das Auswerten von Sensordaten konzentrieren, ohne simultan die Berechnung der Laufalgorithmen durchführen zu müssen (Conrad Electronic SE 2020). Differenzierter betrachtet besteht die Lokomotion-Betriebssoftware aus mehreren Arduino-Programmen. Diese Programme wurden wiederum der Betriebssoftware des Laufroboters „Phoenix“ der Firma Lynxmotion entnommen, aus der Programmiersprache *Basic* übersetzt und modifiziert, um Unterschiede im Aufbau der Roboter anzugleichen. In der modifizierten Betriebssoftware sind neben einiger Standard-Arduino-Bibliotheken eine weiterentwickelte Version der „Servo“-Bibliothek enthalten, die es den Servomotoren ermöglicht, zeitlich festgelegte Bewegungen auszuführen und zu terminieren. Zudem kommen noch die Programme, in denen die mathematischen Kinematikmodelle des Roboters und seine Bewegungsalgorithmen definiert sind, sowie zwei Programme, die für die Steuerung des Roboters mit Hilfe des Gamepads benötigt werden (Conrad Electronic SE 2020).

Neben den Bibliotheken und Beispielprogrammen enthält das Downloadpaket des Herstellers zusätzlich einige Datenblätter über den Aufbau der Lokomotion-Controller-Leiterplatte samt Anschlussplänen für Erweiterungen und deren Pinbelegungen. Zudem kommen die Definitionen der Lagebeziehungen der Roboterbeine und diverse Treiber hinzu, unter anderem für die USB-Schnittstelle zwischen Computer und Lokomotion-Controller. Um die korrekte Kalibrierung des Roboters zu erleichtern, wurde dem Downloadpaket zusätzlich die Windows-Anwendung „Terminal.exe“ hinzugefügt. Mit Hilfe dieser Anwendung lassen sich die Servomotoren des Roboters einzeln ansteuern und kalibrieren, um Positionsungenauigkeiten, die aufgrund von Fertigungstoleranzen oder während des Einbaus der Motoren entstanden sind, auszugleichen.

2.2.2 Umsetzung relevanter Kommunikationsprotokolle

Im Folgenden wird beispielhaft das Verbinden eines Arduino-Mikrocontrollers mit einem geeigneten Peripheriegerät, also einem Gerät außerhalb des Einplatinencomputers, beispielsweise einem weiteren Mikrocontroller oder in diesem speziellen Fall dem Lokomotion-Controller des Roboters, beschrieben. Ist ein Einplatinencomputer dank verbauter Schnittstellenbausteine in der Lage mit anderen Peripheriegeräten kommunizieren zu können, so müssen die bestimmten seriellen Kommunikationsprotokolle in dem Programmcode des Einplatinencomputer initialisiert und eine entsprechende Verdrahtung realisiert werden. Hier wird lediglich auf die Verbindung über eine UART-

Schaltung und den *Inter-Integrated-Circuit*-Bus (I2C-Bus oder auch nur I2C genannt) eingegangen, da diese in der Mikrocontrollertechnik zu den gängigsten gehören (Brühlmann 2019).

Bei Arduino-Boards liegt die serielle Schnittstelle zum Schaffen einer UART-Verbindung nach dem üblichen RS232-Standard auf den digitalen Pins 0 (Empfängerleitung RX) und 1 (Sendeleitung TX) (Brühlmann 2019). Außer dieser seriellen Schnittstelle können aber mit Gebrauch der „SoftwareSerial"-Bibliothek, einer Arduino-Standardbibliothek, auch alle weiteren digitalen Pins des Arduino-Boards zu weiteren RX- und TX-Datenleitungen umprogrammiert werden (Brühlmann 2019). Die Verdrahtung erfolgt bei einer solchen Verbindung, wie Abbildung 4 zeigt, gekreuzt. Bei der Verbindung eines Mikrocontroller-Boards entsprechend der Bedienungsanleitung des Roboters an den Steckerleisten des Lokomotion-Controllers ist diese Verdrahtung bereits in der Leiterplatte des Roboter-Boards implementiert und muss nicht vom Benutzer realisiert werden (Conrad Electronic SE 2020). Die genaue Pinbelegung dieser Verbindung kann der Datei „Roboboard-Pintable.pdf" des Downloadpakets im elektronischen Anhang (F) entnommen werden. Nach der physischen Umsetzung muss die Verbindung bei den beiden Komponenten noch mit einem Sketch initiiert werden. Wie diese Verbindung in einfacher Form aufgebaut wird, zeigt das Beispielprogramm in Abbildung 5. Das UART-typische Daten-Array dieser Verbindung ist in Abbildung 6 dargestellt.

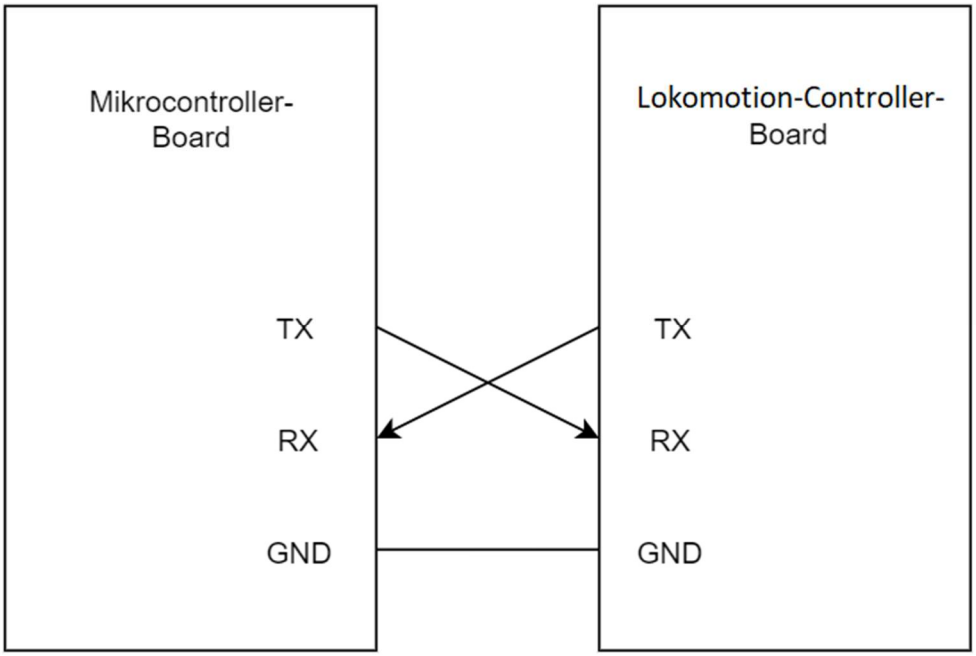

Abbildung 4: Verbindung zweier UART-Bausteine

```
//Einbetten der benötigten Bibliothek

#include <SoftwareSerial.h>

//Initiieren und Verbinden eines SoftwareSerial-Objekts mit
//den Pins
//RX und TX sind wie folgt definiert: Objekt(RX,TX)

SoftwareSerial Bauteil(0,1);

//Im Setup wird die Übertragungsgeschwindigkeit (Bits pro
//Sekunde = Baudrate) des Objektes eingestellt
void setup()
{
  Bauteil.begin(9600);
}

//Im sich wiederholenden Hauptprogramm können nun seriell
//Daten gesendet oder empfangen werden
void loop()
{

//Code für das Senden von Integer-Daten (hier 1909)
int GesendenteDaten = 1909;
  Bauteil.write(GesendenteDaten);

//Code für das Empfangen von Integer-Daten
 while (Bauteil.available())
  {
    int EmpfangeneDaten = Bauteil.read();
  }
}
```

Abbildung 5: Aufbau einer einfachen seriellen Verbindung

Während „CRC“, „CMD“ und „DATA1“ bis „DATA4“ beziehungsweise „DATA6“ die zu übertragenden Inhalte darstellen, sind die Variablen „SYNC0“, „SYNC1“ und „TERMINATION_BYTE“ Bestandteile des Übertragungsprotokolls und stellen die Kommunikation zwischen den beiden Schnittstellen her. Die Betriebssoftware des Lokomotion-Controllers initiiert in der Bibliothek „Data_Input“ bereits diese Verbindung, sodass vom Benutzer nur der erweiternde Einplatinencomputer mit dem passenden Kommunikationsprotokoll programmiert werden muss, wobei die Beispielprogramme unterstützen sollen (Conrad Electronic SE 2020).

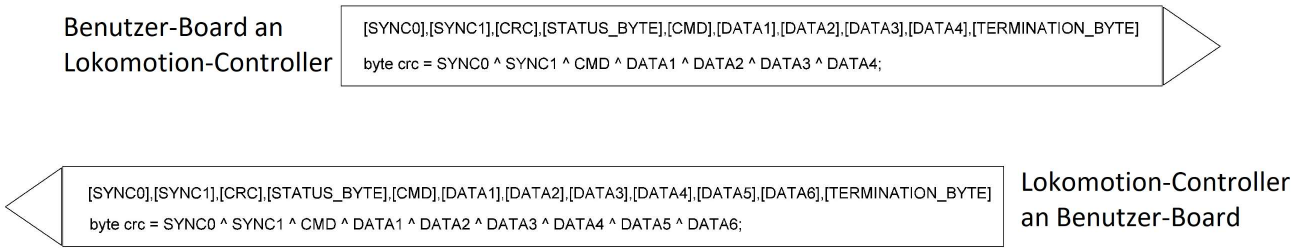

Abbildung 6: Aufbau des Daten-Arrays der UART-Verbindung (Conrad Electronics SE 2020)

Neben einer Verbindung über die UART-Schnittstelle, ist das I2C-Protokoll eine weitere Art die Kommunikation eines Einplatinencomputers mit externen Komponenten zu ermöglichen (Brühlmann 2019). Wie bei serieller Kommunikation üblich, benötigt auch diese Verbindung nur zwei Leitungen. Ein Vorteil gegenüber einer UART-Verbindung ist, dass hier im Datenbus ein so genannter *Master* über dieselben zwei Leitungen mit mehreren *Slaves* kommunizieren kann. Für die Kommunikation benötigt der I2C-Bus eine Datenleitung (SDA), eine Taktleitung (SCL) und eine Definition der Master/Slave-Beziehungen (Brühlmann 2019). Hierfür wird die Arduino-Bibliothek „Wire“ benötigt. In der Praxis ist der Master meist ein Mikrocontroller-Board, während externe Module wie Anzeigen, Sensoren oder weitere Mikrocontroller Slave-Geräte darstellen, denen jeweils eine eigene Adresse im Bus zugeordnet wird (Brühlmann 2019). Die Umsetzung einer solchen Verbindung ist in Abbildung 7 dargestellt. Die hier zu sehenden Widerstände sind in den SDA- und SCL-Anschlüssen von Mikrocontroller-Boards bereits integriert. Alle am Bus angeschlossenen Module müssen in ihrer Rolle klar definiert sein. Abbildung 8 (A) zeigt das benötigte Programm des Masters, während in Abbildung 8 (B) die Definition eines Slaves zu sehen ist, der Daten empfangen soll. Der entscheidendste Nachteil gegenüber einer UART-Verbindung liegt darin, dass ein am Bus angeschlossenes Gerät nicht zeitgleich Master und Slave sein kann, das heißt ein sendendes Mikrocontroller-Board kann nicht parallel Daten empfangen und vice versa.

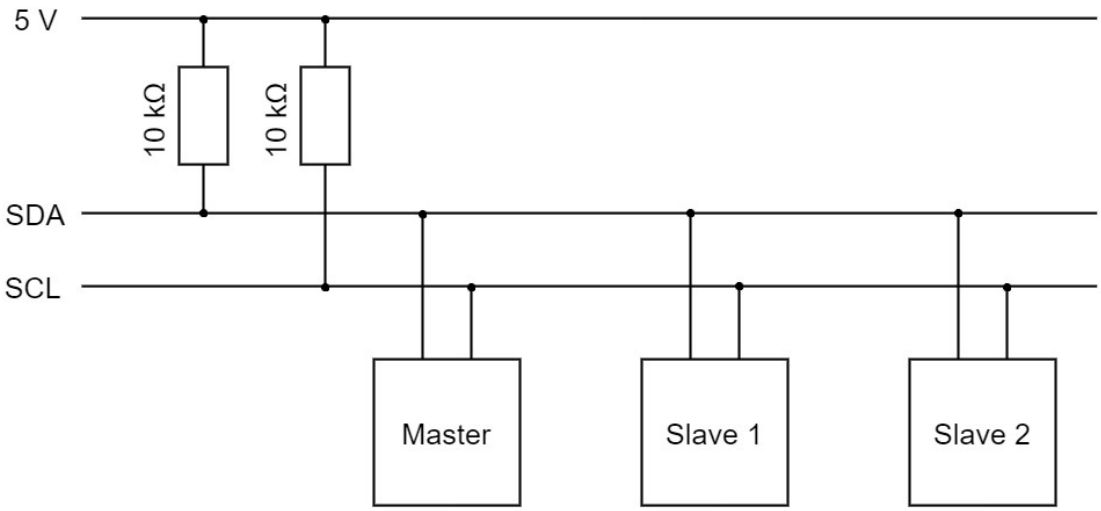

Abbildung 7: Grundschaltung eines I2C-Busses (Brühlmann 2019, S. 136)

```
// Einbetten der benötigten Bibliothek
#include <Wire.h>
// Im Setup wird sich mit dem I2C-Bus verbunden
void setup()
{
// Rolle:
// Master, da keine Adresse festgelegt wird
  Wire.begin();
}
// Im Hauptprogramm können nun Daten an Slaves
// im Bus gesendet werden
void loop(){

// Sende an Slave mit Adresse 3
  Wire.beginTransmission(3);

// Integer-Wert zum Senden (hier 1909);
  int SendeDaten = 1909;

// Daten senden
  Wire.write(SendeDaten);

// Datenübertragung beenden
  Wire.endTransmission();
}
```

A: Master sendet Daten

```
// Einbetten der benötigten Bibliothek
#include <Wire.h>

// Im Setup wird sich mit dem I2C-Bus verbunden
void setup()
{

// Rolle:
//Slave, da eine Adresse 3 zugeordnet wurde
  Wire.begin(3);

// Definiert, welche Funktion aufgerufen wird,
// falls Slave vom Master aufgerufen wird

  Wire.onReceive(DatenEmpfangen);
}
void loop(){

  delay(100);
}
// "DatenEmpfangen" Funktion soll empfangene
// Daten abspeichern
void DatenEmpfangen(int AnzahlBytes)
{
// Daten empfangen und in Variable abspeichern
  int Speichervariable = Wire.read();
}
```

B: Slave empfängt Daten

Abbildung 8: I2C-Implementierung bei Master- und Slave-Modul (nach Brühlmann 2019, S. 138f.)

2.3 Bewegungssteuerung des Laufroboters

Wie in Kapitel 2.2.1 bereits beschrieben, sind die Bewegungsalgorithmen und die mathematischen Modelle dahinter Bestandteil der vom Hersteller bereitgestellten Betriebssoftware des Roboters. Im Allgemeinen lassen sich Bewegungsalgorithmen der Robotik nach zwei verschiedenen Prinzipien kategorisieren. So gibt es die Möglichkeit die Bewegungsabläufe durch indirekte oder direkte Kinematik zu realisieren, wobei letzteres grundsätzlich die einfachere Variante darstellt (Spong u. a. 2006). Während bei direkter Kinematik allgemein mit bekannten Gelenkvariablen, also Gelenkwinkel und -längen, die Position und Orientierung (auch Pose genannt) des Endeffektors in Relation zu den Ursprungskoordinaten bestimmt wird, wird bei der Methodik der inversen Kinematik versucht, aus einer gewünschten Pose des Endeffektors die benötigten Gelenkwinkel des Roboterarms abzuleiten, um diese zu erreichen (Spong u. a. 2006). In diesem konkreten Fall mit drei Freiheitsgraden aufgrund dreier Gelenke mit nur einer zugelassenen Bewegungsrichtung (Siciliano und Khatib 2016), ist der Endeffektor als Fuß eines Roboterbeins und die Glieder des Beins mit Coxa, Femur und Tibia (Abbildung 9) zu bezeichnen. Die direkte Kinematik ist deterministischer Natur, das heißt, es gibt immer nur eine mögliche Lösung, beziehungsweise Endposition und -orientierung des Fußes. Während mit der alternativen Vorgehensweise das Problem einhergeht, dass Berechnungen nach inverser Kinematik oft nicht eindeutig sind (Goldenberg u. a. 1985).

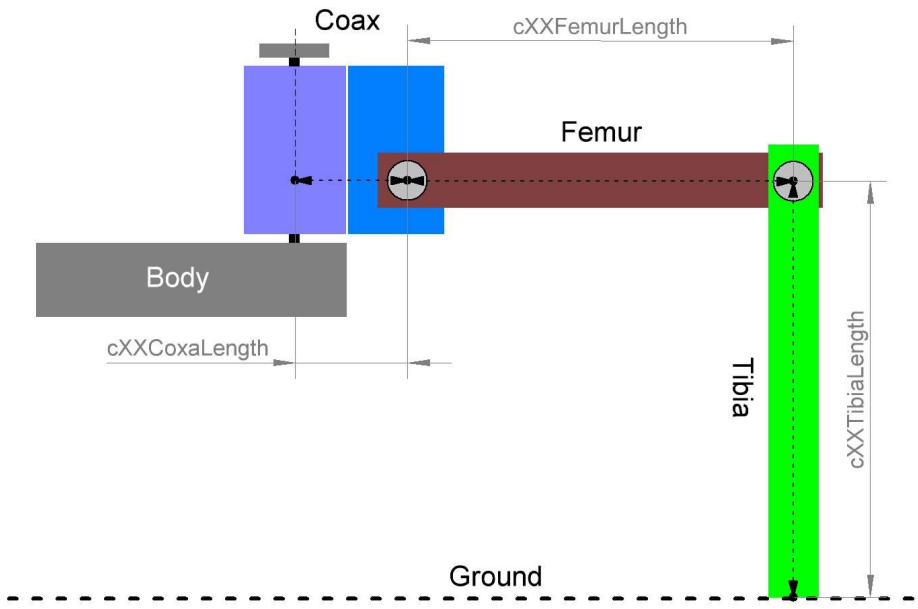

Abbildung 9: Gelenke und Glieder des Laufroboters (Conrad Electronic SE 2020)

So gibt es meist mehrere Möglichkeiten der Gelenkwinkelkombinationen, die den Endeffektor in dieselbe Pose bewegen. Zusätzlich zu der Kinematik der Beine, muss die des Korpus berechnet werden, sodass der Lokomotion-Controller weiß, wie er die Servomotoren der Beine stellen muss, um den Körper in eine bestimmte Position zu bringen. Ebenso müssen, damit ein Fuß des Roboters eine gegebene Pose im Raum erreichen kann, die anderen Beine einen Befehl erhalten, wie sie sich zu positionieren haben. Im Fall der Bewegungssteuerung einer Laufmaschine ist das Vorgehen mit direkter Kinematik daher eher unpassend und es wird nach den Prinzipien der inversen Kinematik vorgegangen, was die Bewegungsabläufe zudem natürlich erscheinen lässt (Yamane und Nakamura 2003). Wie schon zuvor erwähnt, befindet sich das kinematische Modell des Roboters bereits in der Lokomotion-Betriebssoftware. In der Bibliothek „HEX_CNC_3DOF_V2“ sind hierzu die Längen der Beine (Abbildung 9) und des Körpers sowie die Versätze der Coxa-Achsen von der Körpermitte definiert. Um das Problem der Mehrdeutigkeit in der inversen Kinematik zu lösen, werden dort für den Roboter, beziehungsweise seine Gelenke, zusätzliche Bewegungsrestriktionen definiert, sodass sich die Gelenke nur noch innerhalb eines bestimmten Rahmens bewegen lassen und es lediglich eine mögliche Zusammensetzung von Gelenkwinkelkombinationen geben kann.

Neben dem eigentlichen mathematischen Modell samt der benötigten Transformationsmatrizen für Körper und Beine des Roboters befindet sich in der Bibliothek „Phoenix_Code“ außerdem die Definition der Laufsequenzen. Die hier bereitgestellten Gangarten sind typisch für die einer sechsbeinigen Laufmaschine (Ferrell 1995) und inspiriert durch die, schon 1966 durch Wilson erforschten, Laufmuster von Insekten. Im Folgenden werden die drei, in der Natur vorkommenden und auch bei diesem Laufroboter Verwendung findenden, grundlegenden Schrittmuster nach Wilson näher beschrieben. Ein Roboterbein befindet sich demnach entweder in der Stand- oder der Schwungphase und nie in beiden. Insgesamt befinden sich in der Bibliothek sechs verschiedene vorprogrammierte Gangarten, wobei die übrigen drei, hier nicht aufgeführten Muster lediglich Abarten der Hauptgangarten mit geringen Unterschieden, beispielsweise in der Geschwindigkeit, darstellen.

Tripod_6 Bei dieser Gangart stehen immer drei Beine des Roboters fest auf dem Boden (Tripod), während sich die drei verbleibenden Beine in der Schwungphase befinden. Ein Tripod besteht aus dem Vorder- und Hinterbein der einen und dem mittleren Bein der anderen Seite. Sie ist die schnellste der üblichen Gangarten (Abbildung 10).

Wave_24 In dieser Gangart befindet sich immer nur ein Bein allein in der Schwungphase, während die übrigen fünf Beine fest den Boden berühren. Die Bezeichnung rührt daher, dass zuerst alle Beine der einen Seite nacheinander in die Schwungphase wechseln, bevor dann die andere Seite nachzieht. Sie ist die langsamste, aber auch stabilste der üblichen Gangarten (Abbildung 11).

Ripple_12 Diese Gangart kombiniert die Geschwindigkeit des Tripod-Schrittmusters mit der Standfestigkeit des Wave-Musters. Hier besitzt jede der Roboterseiten eine eigene Welle, die zueinander jedoch asynchron verlaufen (Abbildung 12).

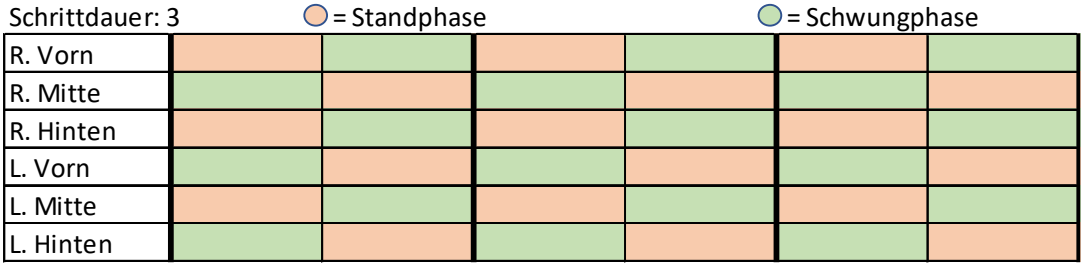

Abbildung 10: Tripod-Laufmuster

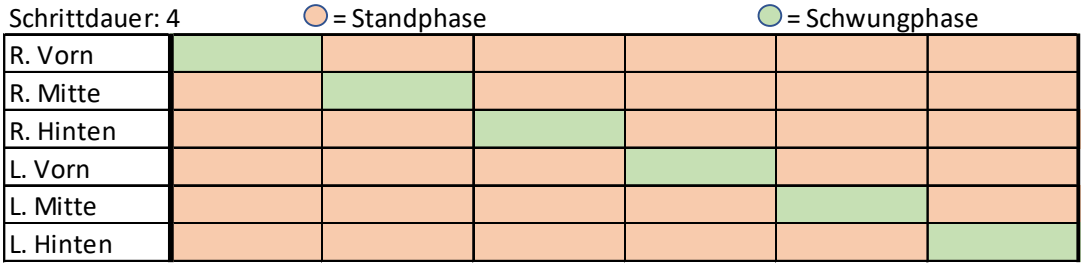

Abbildung 11: Wave-Laufmuster

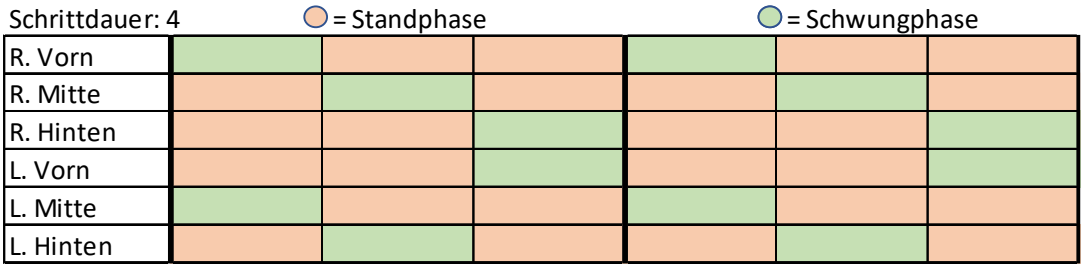

Abbildung 12: Ripple-Laufmuster

2.4 Erweiterungsmöglichkeiten des Roboters mit zur Verfügung stehender Ausrüstung

Der Roboter bietet laut Hersteller die perfekte Grundlage, eigene Projekte umsetzen zu können (Conrad Electronic SE 2020). Aufgrund der Open-Source-Eigenschaft der Software lassen sich die Funktionsweisen des Roboters einfacher verstehen und theoretisch auch manipulieren. Zusätzlich können sich Benutzer an der Lokomotion-Betriebssoftware orientieren und eigene Erweiterungen der Funktionalität des Roboters realisieren (Conrad Electronic SE 2020). Um den Lokomotion-Controller und seine begrenzte Rechenleistung, die ohnehin schon durch das Berechnen der Laufalgorithmen nahezu ausgelastet ist, nicht zu überlasten, bietet der Lokomotion-Controller die Möglichkeit ein Benutzer-Board anzuschließen. Dieses Benutzer-Board kommuniziert dann mit dem Lokomotion-Controller über eine UART-Schnittstelle. Von ihm aus können Funktionen des Lokomotion-Controllers aufgerufen werden und zusätzlich eigene Programme auf dem Benutzer-Board selbst durchlaufen werden, wie zum Beispiel das Auslesen von beziehungsweise Reagieren auf Sensordaten. Grundsätzlich werden im Handbuch des Roboters drei mögliche Erweiterungen mit Benutzer-Boards erwähnt (Conrad Electronic SE 2020). Im Folgenden werden die technischen Eigenschaften und grundsätzlichen Funktionsweisen dieser und anderer, vom Fachgebiet zur Verfügung gestellten, Erweiterungsmöglichkeiten genauer beschrieben. Neben den nachfolgend genannten Elementen verfügt das Fachgebiet auch über zwei zusätzliche Modellbauservos, über deren Einsatz der Benutzer frei verfügen kann.

2.4.1 Eigenschaften eines Arduino UNO-Mikrocontroller-Boards

Arduino gilt als die bekannteste Plattform für Mikrocontroller, zu der Hard- und Software zählen. Die Hauptkomponente und Recheneinheit dieses Benutzer-Boards ist ein Atmel ATmega328P Mikrocontroller, weshalb der Arduino UNO selbst auch als Mikrocontroller oder Mikrocontroller-Board bezeichnet wird (Arduino 2020b). Das Programmieren dieser Boards über die USB-Schnittstelle mit der Arduino IDE wird empfohlen. In einem damit erstellten Arduino-Programm sind charakteristisch zwei Funktionen enthalten (Brühlmann 2019). In der ersten, der *Setup*-Funktion, werden beim Hochfahren des Boards einmalig Variablen initialisiert, Pinmodi festgelegt oder Bibliotheken eingebunden. Dahingegen wird der, sich in der *Loop*-Funktion befindliche Code kontinuierlich wiederholt aufgerufen (Brühlmann 2019). Der Originalhersteller handelt konform mit der Open-Source-Idee, was die Hard- und Software des Mikrocontroller-Boards betrifft (D'Ausilio 2012). So sind mittlerweile auch baugleiche, meist kostengünstigere Alternativen anderer Hersteller

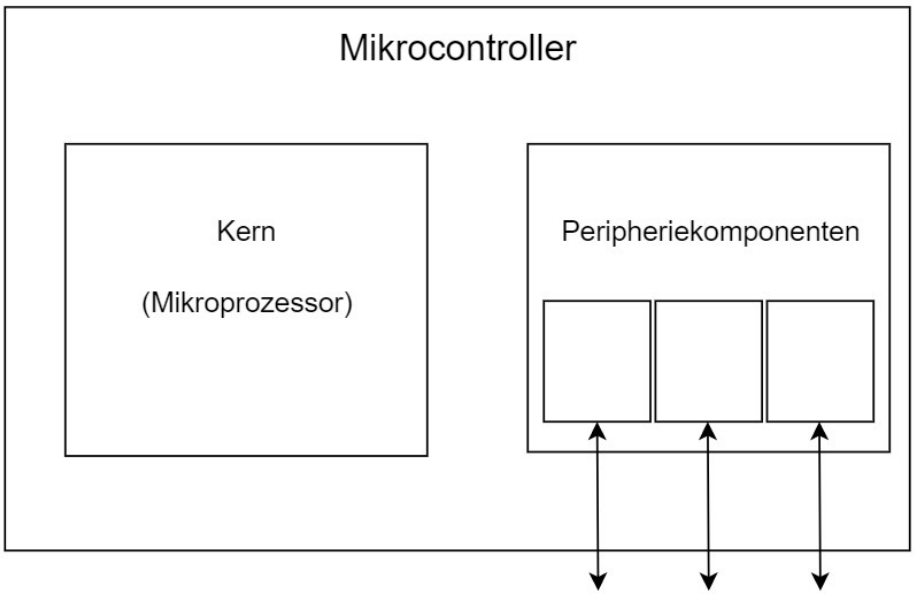

Abbildung 13: Aufbau des Mikrocontrollers (nach Wüst 2011, S. 119)

dieser Benutzer-Boards erhältlich. Grundsätzlich besitzen alle Mikrocontroller einen Kern und eine Peripherie, dessen Komplexität den Anforderungen des Einsatzes entsprechend ausfällt, um Signale mit der Umgebung auszutauschen (Abbildung 13). Dieser Kern ist ein meist einfacher Mikroprozessor (Wüst 2011). Der Controller des Arduino-Boards verfügt über insgesamt 14 digitale Input/Output- und 6 analoge Input-Stecker, eine *Serial-Peripheral-Interface*- (SPI), eine I2C- und eine UART-Schnittstelle. Diese Schnittstellen erlauben unter anderem die Kommunikation mit anderen Mikrocontrollern. So wird bei Erweiterung des Roboters mit einem Arduino UNO (Abbildung 14 (A)) die UART-Schnittstelle verwendet (Conrad Electronic SE 2020).

2.4.2 Eigenschaften eines NodeMCU-Mikrocontroller-Boards

Ein weiteres, neben dem Arduino UNO, weit verbreitetes Open-Source-Mikrocontroller-Board, welches sich jedoch stark in Ausstattung und Größe vom Arduino UNO unterscheidet, ist das NodeMCU. Auf ihm verbaut ist ein ESP8266 Mikrocontroller, welcher sich ebenfalls mit der Arduino IDE programmieren lässt. Das Benutzer-Board besitzt 12 Input-/Output-Stecker, ist circa halb so groß wie ein Arduino UNO und wesentlich günstiger. Zusätzlich zu den SPI-, I2C- und UART-Kommunikationsschnittstellen ist dieses Board mit einer WLAN-Schnittstelle ausgestattet (Espressif 2019). Es ist zu beachten, dass ausschließlich die Ausführung mit verbauter CP2102 USB-UART-Brücke aufgrund dessen Pinabständen mit dem Roboter kompatibel ist (Abbildung 14 (B)). Die

Kommunikation mit dem Lokomotion-Controller des Roboters erfolgt hier ebenfalls über die UART-Schnittstelle (Conrad Electronic SE 2020).

2.4.3 Eigenschaften eines Raspberry Pi-Computers

Der Raspberry Pi gilt, anders als die restlichen Erweiterungsmöglichkeiten, nicht als Mikrocontroller, sondern als Einplatinencomputer mit Ein-Chip-System (SoC) (The Raspberry Pi Foundation 2020b). Während der Arduino UNO, NodeMCU und andere Mikrocontroller-Boards lediglich in der Lage sind, zuvor aufgespielte Programme wiederholt nacheinander abzuarbeiten, ist ein Raspberry Pi aufgrund seiner komplexeren Peripherie, wie zum Beispiel dessen Grafikprozessor (The Raspberry Pi Foundation 2020), durchaus in der Lage, deutlich anspruchsvollere Aufgaben zu erfüllen. So kann dieser Einplatinencomputer mehrere Programme sowohl zeitgleich als auch nacheinander ausführen (Upton und Halfacree 2014). Der Raspberry Pi funktioniert wie ein eigenständiger, programmierbarer Persönlicher Computer (PC), meist betrieben von einem auf Linux basierenden und kostenlos vom Hersteller angebotenen Betriebssystem, wie zum Beispiel *Raspbian*. Die auf dem Raspberry Pi üblich verwendete Programmiersprache ist *Python* (Follmann 2018). Der vom Fachgebiet zur Verfügung gestellte Raspberry Pi 3 Model B hat bereits einen WLAN- und Bluetooth-Chip verbaut, sodass hierfür keine zusätzlichen USB-Anschlüsse belegt werden müssen und besitzt 40 frei programmierbare, G*eneral-Purpose-Input/Output*-Pins (GPIO) (The Raspberry Pi Foundation 2020b). Neben diesen Schnittstellen existieren auf dem Board weitere Anschlüsse. Dazu zählen vier USB-Anschlüsse, ein Klinkenstecker, ein HDMI-Anschluss, ein MPI-CSI-2-Steckverbinder für eine Kamera, ein serieller Display-Schnittstellensteckverbinder (DSI) für LCD-Bildschirme, ein Ethernet-Anschluss, ein microSD-Kartensteckplatz und einen Micro-USB-Anschluss zur Stromversorgung (The Raspberry Pi Foundation 2020b). Der Einplatinencomputer wird, anders als die anderen Benutzer-Boards, nicht direkt in dafür vorgesehene Anschlussleisten des Lokomotion-Controllers gesteckt, sondern über Jumper-Kabel mit der freien Steckerleiste des Roboters verbunden (Abbildung 14 (C)). An dieser Universalanschlussleiste lassen sich aber auch alternative SBC, Funk- oder Bluetooth-Module anschließen (Conrad Electronic SE 2020). Die Kommunikation zwischen Lokomotion-Controller und Raspberry Pi erfolgt dann mit einer UART-Schaltung.

A B C

Abbildung 14: Erweiterungen auf Lokomotion-Controller montiert (Makerfactory 2020b)

2.4.4 Aufbau und Funktionsweise des Ultraschallsensors

Ganz allgemein gehören Abstandssensoren, die die Entfernung aller in Reichweite befindlichen Gegenstände messen können, zu der Grundausstattung eines mobilen Roboters (Brühlmann 2019). Neben den sehr geläufigen Ultraschallsensoren gibt es außerdem die Möglichkeit, Distanzen über einen Infrarotsensor zu bemessen. Diese Sensoren finden in hier jedoch keine Verwendung, da diese am ITPL hierfür nicht zur Verfügung stehen. Ein Ultraschallsensor misst die Strecke zwischen Sensor und einem im Weg befindlichen Gegenstand indem er ein Signal (einen Schall, der über dem von Menschen hörbaren Frequenzbereich liegt) sendet und das vom Gegenstand zurückgeworfene Signal empfängt (Brühlmann 2019). Mit der zwischen Senden und Empfangen vergangenen Zeit kann dann der Abstand relativ genau bemessen werden. Bei den in dieser Arbeit verwendeten Ultraschallsensoren, welche vom Fachgebiet bereitgestellt werden, handelt es sich um das von verschiedenen Herstellern vertriebene Modell HC-SR04. Neben zwei Pins für die benötigte 5-V-Stromversorgung (VCC und GND) besitzt dieses Messmodul außerdem einen Trigger- und einen Echo-Pin, die im Programmcode des Einplatinencomputers als Output-, beziehungsweise Input-Pin definiert werden müssen. Es handelt sich bei den Trigger- und Echo-Signalen um PWM-Signale. Zu Beginn der Messung setzt der Einplatinencomputer den Trigger-Pin mindestens 10 µs auf HIGH. Nachdem der Trigger-Pin wieder auf LOW gesetzt ist, beginnt das Modul damit, acht kurze 40kHz Pulse zu senden. Nach versenden der acht Pulse schaltet der Echo-Pin von LOW auf HIGH solange, bis das Modul das reflektierte Signal wieder erfasst hat. Der Einplatinencomputer berechnet dann mit der Dauer, die der Echo-Pin ein HIGH-Signal gesendet hat, die Entfernung über die Formel 1.

$$Abstand = \frac{1}{2} * \ Schallgeschwindigkeit \left[\frac{m}{s}\right] * gemessene\ Zeit\ [s]$$

Formel 1: Formel zur Abstandsberechnung (nach Brühlmann 2019, S. 234)

Das Produkt aus Schallgeschwindigkeit und gemessener Zeit muss noch halbiert werden, da letzteres die Zeit für Hin- und Rückweg des Signal beinhaltet. Ein Codebeispiel, wie eine solche Messung zu implementieren ist, befindet sich im elektronischen Anhang (E). Des Weiteren muss bei der Erweiterung des Roboters darauf geachtet werden, dass es sich bei dem vorliegenden Modell, das Abstände von maximal 4 m messen kann, um ein Modul mit einer Betriebsspannung von 5 Volt handelt. Soll das Modul Beispielsweise an einem Raspberry Pi, dessen ARM-Prozessor mit einer Betriebsspannung von nur 3,3 V arbeitet, angeschlossen werden, so ist das Zwischenschalten eines Spannungsmodulators unabdingbar. Andernfalls sind Schäden am Raspberry Pi zu befürchten (Joy-IT 2017).

2.4.5 Beschreibung der Tiefenkamera

Neben der Möglichkeit des Roboters, Abstandssensoren zur Orientierung im Raum zu verwenden, können auch Bildsensoren hierfür eingesetzt werden. Einige Kameramodule sind speziell auf den Einsatz mit Einplatinencomputer ausgerichtet (The Raspberry Pi Foundation 2020a) (PixyCam 2020). Das Fachgebiet schaffte zur Erweiterung des Roboters eine Tiefenkamera der Intel-RealSense-Reihe an. Eine solche Kamera ist durch zwei verbaute Objektive und einen Infrarotprojektor dazu in der Lage, auch Tiefen zu erkennen und Abstände zu berechnen (Intel 2019a). Dies geschieht durch die so genannte *Stereo Vision*, bei der Unterschiede in den Bildern der zwei Bildquellen identifiziert werden und daraus Tiefenwerte auf Distanzen von bis zu 10 m für die einzelnen Pixel der Bilder berechnet werden (Intel 2019a). Das Infrarotmodul projiziert permanent ein Infrarotmuster, um die Genauigkeit der Tiefenberechnung auch in Umgebungen mit wenig Textur zu unterstützen (Intel 2019a). So gelingt es der speziellen Software, die von Intel kostenfrei bereitgestellt wird, ein komplettes Bild mit akkuraten Tiefenwerten zu generieren. Da diese Software höhere Rechenleistung benötigt als die Verarbeitung eines gewöhnlichen Kamerabilds, kann die Kamera jedoch nicht mit einem Mikrocontroller-Board verbunden werden (Intel 2019a). Ist der Einsatz der Kamera am in dieser Arbeit beschriebenen Laufroboter gewünscht, so muss dieser mit einem Linux oder Windows fähigen (ab Windows 10) Computer, beispielsweise einem Raspberry Pi 3, per USB-Kabel verbunden werden,

damit dieser die Bilddaten verarbeiten kann. In den Konstruktionen des Benutzers kann die Kamera mit Schrauben der Gewindenormen M3 oder 1/4 “ 20 UNC befestigt werden.

2.4.6 Erweiterungen mittels additiver Fertigungsverfahren

Der Begriff 3D-Druck stellt einen Überbegriff für verschiedene additiven Fertigungsverfahren dar (Feldmann und Gorj 2017). Diese Verfahren zeichnen sich dadurch aus, dass die Geometrien eines zu fertigenden Bauteils schichtweise durch physikalische oder chemische Schmelz- oder Härtungsverfahren aufgebaut werden (Feldmann und Gorj 2017). Als Grundlage für diesen schichtweisen Aufbau dient in der Regel ein zuvor konstruiertes *Computer-Aided-Design*-Modell (CAD-Modell) (Feldmann und Gorj 2017). Die Anwendungsbereiche dieser 3D-Druckverfahren sind vielfältig und reichen von der Fertigung von Prototypen, künstlerischen Objekten, Ersatzteilen bis hin zur (Klein-)Serienproduktion und sind speziell aufgrund von hoher Individualisierbarkeit und leichter Bauweise durch die internen Wabenstrukturen der Bauteile bei Branchen wie der Luftfahrt-, Automobil- oder Medizintechnik beliebt (Feldmann und Gorj 2017). Wegen der hohen Individualisierbarkeit von Bauteilen ist der 3D-Druck auch für diese Arbeit relevant. Das in dieser Arbeit verwendeten 3D-Druck-Verfahren basiert auf dem Prinzip der Schmelzschichtung, bei dem Kunststoffe oder Kunststoffmischungen aufgeschmolzen und dann in kleinen Abständen geschichtet werden, während Computer und Mikrocontroller die Prozessparameter, wie Düsen- und Heizbretttemperatur oder Düsendurchmesser, kontrollieren und die Aktoren (Antriebselemente) steuern (Feldmann und Gorj 2017). Da sich die Möglichkeit einer Erweiterung des Roboters nicht nur auf die in den vorherigen Kapiteln erwähnten elektronischen Elemente beschränken soll, sieht der Hersteller auch eine Veränderung des Erscheinungsbilds durch das Montieren von gedruckten Bauteilen am Roboter vor (Conrad Electronic SE 2020). Im Downloadpaket (elektronischer Anhang (F)) befinden sich dafür bereits diverse druckbare CAD-Modelle für den Zusammenbau eines Anbauteils für den Roboter, in dem wahlweise ein Ultraschallsensor oder eine Kamera integriert werden kann und sich durch zwei zusätzliche Servomotoren schwenken und neigen lässt (Conrad Electronic SE 2020). Selbstverständlich kann an der dafür vorgesehenen Befestigungsmöglichkeit im Robotergerüst auch ein eigenkonstruiertes und gedrucktes Element montiert werden. Um dies zu ermöglichen hat das Fachgebiet einen eigenen 3D-Drucker angeschafft. Der S3 der Firma Ultimaker ist in der Lage, Bauteile mit maximalen Maßen von 230 x 190 x 200 mm mit einer Geschwindigkeit von circa 24 mm^3/s in einer Schichtauflösung von bis zu 20 µm zu drucken (Ultimaker 2019).

3. Vorstellung des Fachlabors und vergleichbarer Lehrkonzepte

In der Vergangenheit haben sich Forschungen immer wieder damit beschäftigt, Lehrprogramme im Themengebiet der Robotik zu konzeptionieren, die Anhand praktischer Versuche an Robotern Kompetenzen in Mechanik, Elektrotechnik und Informatik vermitteln sollen (Siegwart 2001). Diese Lehrkonzepte richteten sich meist sowohl an Schülerinnen und Schüler (SuS) der Sekundarstufe (Kurebayashi u. a. 2006) oder Studierende ohne Abschluss, wie Bachelorstudierende (beispielsweise bei Klassner 2002). Die Metastudie von Major, Kyriacou und Brereton (2012) betrachtet 34 solcher Studien, von denen der Großteil (75 %) den Einsatz eines Roboters in ihrem Lehrkonzept als effektiv beschrieben. Jedoch merken die drei Autoren an, dass die übliche Bewertungsgrundlage der Effektivität meist eine Befragung der beteiligten SuS und Studierenden war. Kontrollgruppenbasierte Studien, in denen die erlernten Kompetenzen von SuS und Studierenden mit, beziehungsweise ohne Zugang zu einem Roboter beim Lernen verglichen wurden, kommen hier zu einem weniger ausdrucksstarkem Ergebnis (Major u. a. 2012). Von diesen kommen 66 % zu dem Ergebnis, dass der Einsatz von Robotern effektiv war (Major u. a. 2012). Bei den meisten der in der Studie betrachteten Roboter handelt es sich um Modelle der LEGO Mindstorms-Produktserie. Diese Roboter zeichnen sich durch einen simplen Aufbau und einfache Programmierbarkeit aus (Klassner 2002). Die Metastudie kommt hinsichtlich der Untersuchung der Effektivität unterschiedlicher eingesetzter Robotervarianten zu keinem klaren Ergebnis (Major u. a. 2012). Aus den Ergebnissen der in der Vergangenheit durchgeführten Studien lässt sich also keine klare Aussage über die Effektivität eines Einsatzes des in dieser Arbeit behandelten Laufroboters in der Lehre treffen. Dessen ungeachtet soll der Roboter in Zukunft in dem vom ITPL angebotenen Fachlabor zur Fabrikautomation für Studierende als Experimentierplattform Verwendung finden. Dieses Labor richtet sich primär an Master-Studierende der Fächer Wirtschaftsingenieurwesen und Logistik. Schwerpunkt bildet hier die praktische Umsetzung von Automatisierungstechniken in einer Modellfabrik mit geeigneten IT-Komponenten (ITPL 2020). Derzeit wird dort das Wissen lediglich mit Hilfe eines zu automatisierenden Fahrstuhlmodells gelehrt. Die Veranstaltung gliedert sich in zwei Elemente. Zunächst werden in Präsenzterminen programmiertechnische Grundlagen und Grundlagenwissen der Sensorik, Aktorik, Fabrikautomation und in der Überwachung technischer Systeme vermittelt (ITPL 2020). Anschließend werden in zusammengeschlossenen Entwicklerteams eigenständig Softwaremodule ausgearbeitet, die abschließend schriftlich sowie im Rahmen einer Disputation präsentiert werden (ITPL 2020). Inwieweit sich relevante Kompetenzen der Fabrikautomation

anhand des neu angeschafften Roboters vermitteln lassen, soll im nachfolgenden Kapitel 4 ausführlicher behandelt werden. Es lässt sich nach der Lehrveranstaltungsbeschreibung des Fachlabors jedoch bereits eine grundsätzliche Kompatibilität hinsichtlich der geforderten Programmierkompetenzen bestätigen (ITPL 2020). So wird hier die Programmiersprache *C++* verwendet, die in einer vereinfachten Form, aber mit derselben Syntax, auch in der Programmierung des Laufroboters eingesetzt wird (ITPL 2020) (Conrad Electronic SE 2020). Zum angebotenen Fachlabor, in dem der hier behandelte Laufroboter später Verwendung finden soll, vergleichbare Lehrveranstaltungen anderer Hochschulen bietet beispielsweise das Institut für Mechatronische Systeme der Leibniz Universität Hannover an. Im Rahmen des Projektes „Roboterfabrik“, dessen Ziel es ist, Hannover zu einem führenden Standort in der Robotik zu machen, werden hier Lehrveranstaltung in zwei Varianten zu aktuellen Themen der Robotik angeboten (Institut für Mechatronische Systeme 2020a). So kann sich eine Lehrveranstaltung aus einer Vorlesungsreihe und einem anschließendem Labor (hier *Robothon* genannt), in dem die erlernten Kenntnisse praktisch an Robotern umgesetzt werden sollen, zusammensetzen (Institut für Mechatronische Systeme 2020a). Alternativ werden auch Lehrveranstaltungen angeboten, die nur aus einem Labor bestehen, beispielsweise die Veranstaltung zu mobiler Robotik, in der eine Applikation eines industrienahem Robotersystems im Rahmen einer Blockveranstaltung programmiert werden soll (Institut für Mechatronische Systeme 2020b). Lehrveranstaltungen, die den praktischen Gebrauch eines komparablen Laufroboters, wie ihn das Fachgebiet angeschafft hat, beinhalten, konnten in den Recherchen zu dieser Arbeit nicht aufgefunden werden.

4. Implementierungsmöglichkeiten des Laufroboters in das Lehrkonzept

Mit dem Wissen der in Kapitel 2 ausführlich behandelten Grundlagen kann nun damit begonnen werden, den Roboter hinsichtlich seiner Verwendbarkeit im Lehrbetrieb zu bewerten. Als Bewertungsgrundlage sollen die, in diesem Kapitel beispielhaft konstruierten Aufgabenstellungen dienen. Diese drei hier vorgestellten Konzepte werden insbesondere nach Komplexität der durchzuführenden Arbeitsschritte als auch dem daraus entstehenden Aufwand beurteilt. Ferner wird ein Ausblick über mögliche Erweiterungen oder Modifikationen dieser Lehrkonzepte gegeben, um Einfluss auf die aufzubringende Leistung der Studierenden zu nehmen. Die hier präsentierten Konzepte stellen jedoch keine strikten Vorgaben, sondern Handelsempfehlungen dar. Sie sollen eine reine Hilfestellung bieten und aufzeigen, welche Möglichkeiten der Roboter besitzt, individualisiert zu werden. So stehen diese Konzepte unter der Prämisse, eine möglichst große Verschiedenheit in den verwendeten Benutzer-Boards und den resultierenden Roboterfunktionen aufzuweisen, um den Umfang der zu erlernenden Kompetenzen zu maximieren. Neben der Erweiterung mit Hilfe der Benutzer-Boards ist in diesen Konzepten auch der Gebrauch der übrigen, zusätzlich vom Fachgebiet angeschafften Ausrüstung vorgesehen. Es ist zu erwähnen, dass die hier angewandten Arbeitsschritte meist nicht die einzige Möglichkeit darstellen, die beschriebene Problemstellung zu bearbeiten. Während in Kapitel 4.1 und 4.2 Konzepte und eine mögliche Umsetzungsweise nur theoretisch ausgearbeitet werden, ist in Kapitel 4.3, neben einer Aufführung des möglichen Lösungsansatzes, auch seine praktische Umsetzung dokumentiert. Die Bearbeitungen der verschiedenen Konzepte beinhalten abschließend allesamt eine Evaluierung, in der das jeweilige Konzept seine Beurteilung erfährt. Gegenstand dieser Beurteilung sind die bereits erwähnten Aspekte Komplexität, Aufwand und Erweiterbarkeit. Im Detail werden hier zudem die benötigten Kompetenzen des Studierenden erwähnt.

4.1 Konstruktion eines Überwachungsroboters

In diesem Unterkapitel wird der Umbau des Laufroboters zu einem fernsteuerbaren Aufklärungs- und Überwachungsroboter instruiert. Nach dem Zusammentragen der gewünschten Funktionen wird eine mögliche Umsetzungsweise vorgeschlagen und erläutert. Abschließend wird das Konzept hinsichtlich der weiter oben genannten Bewertungskriterien beurteilt.

4.1.1 Formulierung der Problemstellung

Der korrekt zusammengebaute Roboter im Auslieferungszustand stellt hier die Ausgangssituation dar. Nach Abschluss der Modifikation des Laufroboters soll dieser kabellos steuerbar sein. Außerdem soll er dazu in der Lage sein, über eine drahtlose Verbindung eine Video- oder Bildübertragung auf ein vom Benutzer gewähltes Gerät zu senden. Der Benutzer kann dann auf entsprechende Ereignisse, die im einsehbaren Umfeld des Roboters geschehen, reagieren. Mögliche Anwendungsbereiche eines Roboters mit solchen Funktionen lassen sich viele identifizieren. So ist beispielsweise der Einsatz in Krisengebieten denkbar, in denen menschliche Aufklärungseinheiten einem hohen Risiko ausgesetzt wären. Als mögliches Einsatzfeld im privaten Bereich ist eine Verwendung des modifizierten Roboters als Überwachungsroboter vorstellbar, der unbefugten Aufenthalt im Wohnraum des Benutzers anzeigt. Um einen solchen Roboter zu konstruieren, müssen im Detail folgende Aufgaben gelöst werden:

1. Herstellen einer Verbindung des Roboters über eine geeignete drahtlose Verbindung zu einem externen Gerät des Benutzers
2. Implementierung einer Kamera in Hard- und Software des Roboters, um eine qualitativ hochwertige Video- und Bildübertragung zu erreichen
3. Aufbau einer Benutzerplattform, auf der sowohl das übertragene Bild einsehbar als auch die Steuerung des Roboters über diese möglich ist

4.1.2 Vorstellung einer möglichen Lösung

Zu allererst muss geklärt werden, welche Veränderungen am Roboter zwingend vollzogen werden müssen, damit dieser in der Lage ist, die oben aufgeführten Aufgaben zu erfüllen. Da zu diesen jedoch meistens unterschiedliche Lösungsansätze mit ähnlichen Resultaten existieren, wird ebenfalls aufgeführt, warum sich für die hier ausgearbeitete Lösungsmöglichkeit entschieden wurde. Im Folgenden werden die drei Aufgaben schrittweise abgearbeitet, ausführlich genug, um das resultierende Ergebnis reproduzieren zu können.

Ad 1.

Zur Identifizierung der benötigten Erweiterungen müssen die technischen Eigenschaften des Lokomotion-Controller-Boards aus Kapitel 2 herangezogen werden. Wie in Abbildung 1 beziehungsweise Tabelle 1 zu erkennen ist, wurde auf dem Board neben dem Gamepad-Empfänger

kein weiteres Funkmodul verbaut. Somit muss, um eine für die Bildübertragung und Bewegungssteuerung geeignete drahtlose Verbindung herstellen zu können, ein externes Bauteil angeschlossen werden. Die zwei weitverbreiteten Kommunikationsstandards WLAN und Bluetooth kommen als Verbindungsarten hierfür in Frage. Um eine dieser Verbindungen zu realisieren, ist ein entsprechendes Funkmodul mit dem Roboter zu verbinden. Hierfür stehen auf dem Board genügend freie Anschlussleisten zur Verfügung (Conrad Electronic SE 2020). Aufgrund der nach aktuellen Standards höheren Bandbreite verglichen zu einer Bluetooth-Verbindung, wird sich hier für den Einsatz einer WLAN-Verbindung entschieden. Eine höhere Bandbreite lässt eine höhere übertragende Datenmenge zu, sodass die Videoqualität nicht begrenzt werden muss und trotzdem ein stabiles Bild übertragen werden kann. Da das Fachgebiet kein einzelnes WLAN-Modul besitzt, muss auf einen der bereitgestellten Einplatinencomputer zurückgegriffen werden. Die mit WLAN-Chips ausgestatteten Einplatinencomputer NodeMCU und Raspberry Pi können hierfür verwendet werden. Da der zur Verfügung stehende Bildsensor, also die in Kapitel 2.4.5 beschriebene Tiefenkamera, nur an einen USB-Anschluss angeschlossen werden kann und das NodeMCU keinen solchen Anschluss besitzt, eignet sich lediglich der Raspberry Pi als Erweiterung. Aufgrund des hohen benötigten Arbeitsstrom des Einplatinencomputers wird geraten, diesen zusätzlich mit einer externen Energiequelle wie einer Powerbank zu betreiben, damit die Leistung des Raspberry Pi sich nicht herabsetzt (Makerfactory 2020b). Wie dieser auf dem Roboter zu montieren ist, zeigt bereits Abbildung 14 (C). Die Abbildung 15 zeigt die genaue Verdrahtung der beiden Boards. Zunächst sollte der Mikrocomputer aber getrennt vom Roboter programmiert werden. Befindet sich der Raspberry Pi im Auslieferungszustand, so muss dieser zuerst eingerichtet werden. Dazu zählen neben dem Bereitstellen und Aktualisieren des Betriebssystems auch das empfohlene Einrichten einer Fernzugriffverbindung. Mit einer solchen Verbindung kann auf den Raspberry Pi von einem externen PC aus zugegriffen und von diesem aus programmiert werden, solange er sich im selben lokalen Netzwerk befindet. Wie ein Raspberry Pi und der Fernzugriff auf einen solchen von einem PC korrekt eingerichtet wird, ist im Werk von Kofler, Kühnast und Scherbeck (2016, S. 137ff.) nachzulesen. Nachdem alle Vorbereitungen getroffen sind, kann damit begonnen werden, die UART-Verbindung in der Software aufzubauen. Dafür muss jedoch die Schnittstelle des Raspberry Pi zunächst in seinen Einstellungen aktiviert werden. Da Programme auf dem Raspberry Pi üblicherweise in der Sprache Python geschrieben werden, ist zu beachten, dass Arduino-Programme nicht ohne vorherige Übersetzung auf dem Mikrocomputer ausgeführt werden können. Um die serielle Schnittstelle auf dem Raspberry Pi zu programmieren, muss also ein Programm in Python geschrieben werden.

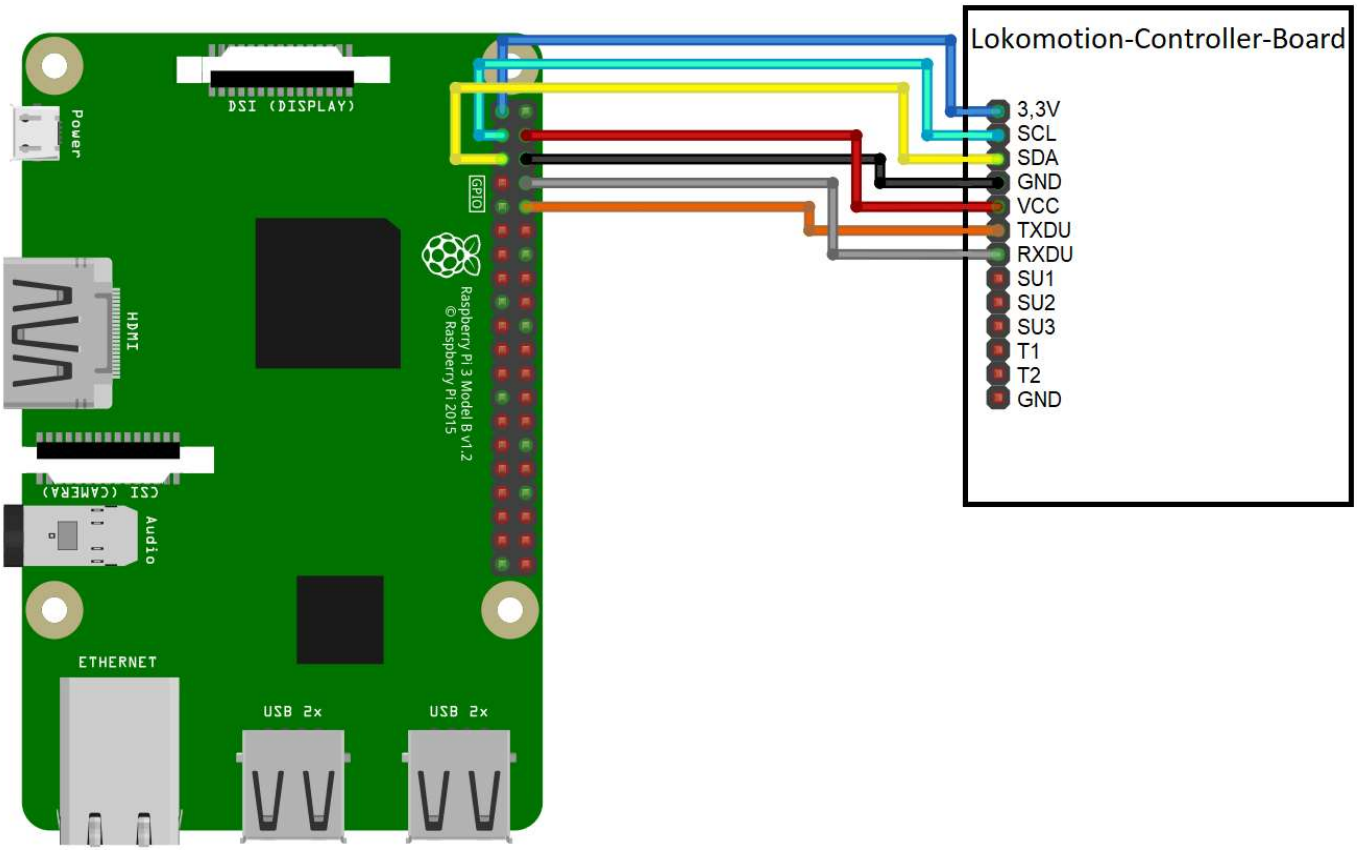

Abbildung 15: Anschluss des Raspberry Pi an der Universalanschlussleiste des Lokomotion-Controller-Boards (Fritzing-Darstellung nach Conrad Electronic SE 2020)

Wie dabei genau vorzugehen ist, kann in weiterführender Literatur nachgelesen werden (Kofler u. a. 2016, S. 484f.). Üblicherweise muss die Schnittstelle auch auf dem rezipierendem Ende der UART-Verbindung, was hier der Lokomotion-Controller darstellt, durch einen kurzen Programmcode initialisiert werden. Dieser ist aber bereits in seiner Betriebssoftware abgespeichert. Im elektronischen Anhang (E) befindet sich das Programm „Serielle_Schnittstelle_Lokomotion“, welches der Bibliothek „Data_Input“ entnommen wurde und zeigen soll, wie das Protokoll der Verbindung aufgebaut ist. Für eine erfolgreiche Kommunikation zwischen Lokomotion-Controller und Raspberry Pi ist es unabdingbar, im Python-Programm eine zum Protokoll kompatible Verbindung zu initiieren. Sollten diesbezüglich Unklarheiten aufkommen, so ist es ratsam, das bereits in Kapitel 2.2.1 erwähnte Beispielprogramm „Moving_01“ näher zu betrachten. Hier wird eine solche Verbindung zum Lokomotion-Controller von einem externen Arduino-Gerät aufgebaut. Das Skript lässt sich aber auch in Python und somit für den Raspberry Pi übersetzen.

Ad 2.

Grundsätzlich ist der Anschluss jeder handelsüblichen Kamera über die USB-Schnittstelle des Raspberry Pi möglich (The Raspberry Pi Foundation 2020b). Für die hier beschriebene Implementierung eignen sich bereits herkömmliche Webcams oder das über den Kameraanschluss verbindbare Raspberry Pi Kameramodul der Raspberry Pi Foundation (The Raspberry Pi Foundation 2020a). Bei der Wahl der zu verwendenden Kamera sollte jedoch auf die Treiberkompatibilität der Kamera und des verwendeten Betriebssystems des Mikrocomputers geachtet werden. Die vom Fachgebiet bereitgestellte Tiefenkamera ist neben dem Raspbian Betriebssystem auch mit anderen Linux-Distributionen wie *Ubuntu MATE* kompatibel. Dieses Betriebssystem, welches ebenfalls mit dem Raspberry Pi kompatibel ist, zeichnet sich durch wesentlich aktuellere Softwareversionen von Python und anderen Softwarepaketen aus, weshalb das zu wählende optimale Betriebssystem immer einer anwendungsspezifischen Einzelfallentscheidung obliegt (Kofler u. a. 2016). Wie dieses Betriebssystem zu installieren ist, kann ebenfalls in weiterführender Literatur, beispielsweise im Werk von Kofler, Kühnast und Scherbeck von 2016 (S. 262f.), nachgelesen werden. Für diese spezielle Anwendung sind die Softwarepakete des Raspbian Betriebssystems jedoch völlig ausreichend. Anleitungen, wie das von Intel herausgegebene Softwareentwicklungspaket (SDK) auf den beiden verschiedenen Betriebssystemen mit Hilfe von GitHub-Repositorien zu installieren ist, befinden sich im elektronischen Anhang (B) dieser Arbeit. Der Inhalt dieses Pakets ist jedoch für die Anwendung der Kamera in diesem Beispiel weitestgehend ohne Bedeutung, da es für deutlich komplexere Anwendungen ausgelegt ist. Hier wird jedoch lediglich die Installation des darin befindlichen Treibers benötigt, damit die Tiefenkamera auch wie eine herkömmliche Webcam verwendet werden kann. Nach der erfolgreichen Installation dieses Entwicklungspakets kann die Kamera über das USB-Kabel an den Raspberry Pi angeschlossen und die im Paket enthaltenden Anwendungen, wie der „Intel RealSense Viewer“, gestartet werden (Intel 2019b). Der hier Verwendung findende Raspberry Pi verfügt lediglich über USB-Anschlüsse des Typs 2.0 (The Raspberry Pi Foundation 2020b). Da die von der Kamera übertragende Datenmenge jedoch die maximal mögliche Datenübertragungsrate eines USB-2.0-Anschlusses übersteigt, wird die Leistungsfähigkeit der Tiefenkamera stark beeinträchtigt (Intel 2019b). Diese Beschränkung der Leistung kann sich dann in Form von einer abnehmenden Bildrate (FPS) oder Auflösung bemerkbar machen. Nach der Einrichtung der Software und der Herstellung einer physischen Verbindung zum Raspberry Pi muss sich mit der Befestigung der Kamera am Robotergerüst auseinandergesetzt werden. Eine Ästhetik konforme Lösungsmöglichkeit bietet

hier der 3D-Drucker des Fachgebiets. Mit diesem lässt sich eine simple, auf dem Roboter montierbare Halterung der Tiefenkamera fertigen. Eine solch individualisierte Konstruktion ist ebenfalls als im 3D-Druck typische STL-Datei in zwei Montagevarianten (Befestigung an der Rückseite beziehungsweise Unterseite der Kamera) der Arbeit im elektronischen Anhang (A) angefügt. Sollte der Überwachungsroboter von anderen Bildsensoren Gebrauch machen, so lassen sich mit Hilfe von für Studierende frei zugänglichen CAD-Anwendungen, wie Autodesk Fusion 360, auch andere Halterungen erstellen.

Ad 3.

Damit der Roboter nun seinen in der Problemstellung beschriebenen Zweck erfüllen kann, benötigt der Benutzer eine Plattform mit der er den Roboter kabellos kontrollieren kann. Darüber hinaus kann es für die Anwendungen hilfreich sein, den Benutzer die Möglichkeit zu geben, das von der Kamera übertragende Bild zeitgleich einsehen zu können. Möglichkeiten eine Plattform zu erstellen, die beide Funktionen vereint, gibt es einige. In dieser Ausarbeitung wird sich auf den Aufbau einer solchen Plattform als Webpage beschränkt. Auf die so entstandene Webpage soll dann von externen Geräten, die mit demselben lokalen Netzwerk verbunden sind, in welchem sich auch der Roboter befindet, zugegriffen werden können. Aufgrund ihrer einsteigerfreundlichen Syntax empfiehlt es sich für das Erstellen einer solchen Webpage, die Programmiersprache *PHP* zu verwenden (Kofler u. a. 2016, S. 822). Der Unterschied zu beispielsweise HTML-Webpages liegt darin, dass es sich bei PHP um dynamische Webpages handelt. Anders als bei den statischen (wie HTML-Webpages) wird hier erst bei einem Zugriff auf die Webpage von einem PHP-Interpreter eine HTML-Page erzeugt und zurückgeliefert (Bühler u. a. 2018). Dank dieser Dynamik kann Raspberry Pi-Hardware bereits mit kleinen PHP-Programmen auch über das Internet gesteuert werden. Zum übertragen solcher, mit PHP erstellten Seiten, wird üblicherweise die Webserveranwendung *Apache* verwendet (Kofler u. a. 2016). Mit Apache lassen sich Webserver einrichten, die dann Webpages mit statischen (HTML) oder dynamischen Inhalten (beispielsweise mit PHP realisiert) hosten. Wie diese Anwendung auf dem Raspberry Pi zu installieren ist und wie anschließend damit ein Webserver erstellt werden kann, ist in einem Tutorium im elektronischen Anhang (B) hinterlegt. Nach erfolgreicher Bearbeitung dieser Anleitung wird sich nun der Vorbereitung der Bildübertragung zugewandt. Damit das Bild der Kamera in Echtzeit in einer PHP-Webpage eingebunden werden kann, wird die Installation einer Bildübertragungssoftware empfohlen. In diesem Fall wird von der eigentlich für Bewegungserkennung verwendeten Software *Motion* Gebrauch gemacht.

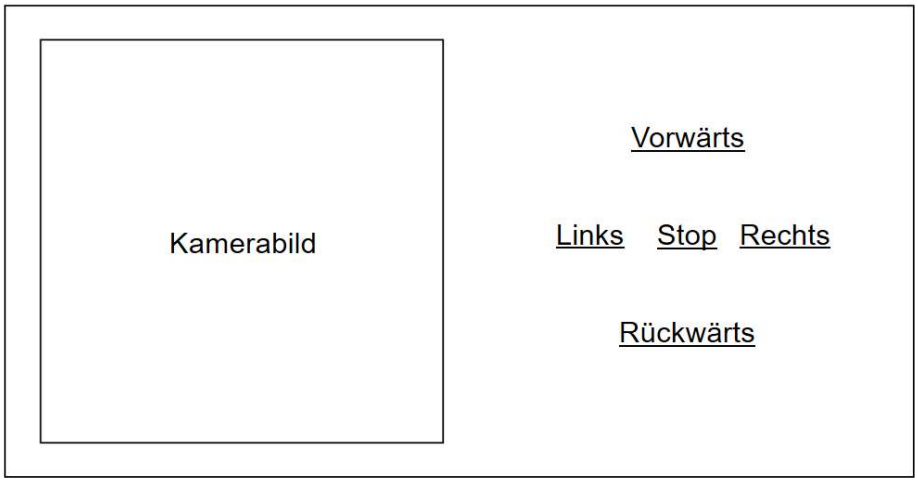

Abbildung 16: Elemente der Weboberfläche

Für die Bearbeitung der oben aufgestellten Problemstellung wird die Bewegungserkennungsfunktion jedoch erst einmal nicht benötigt und kann ignoriert werden. Mit diesem Programm lässt sich eine Bildübertragung realisieren, die individuell beispielsweise in FPS, Auflösung oder Bildhelligkeit angepasst werden kann. Die Dokumentation dieses Programmes erläutert das Vorgehen bei der Konfiguration dieser Bildübertragung (Mr Dave 2020). Anschließend muss eine Verbindung zwischen der Weboberfläche und der Funktionen des Roboters hergestellt werden. Die über die Webpage eingegebenen Bewegungsbefehle sollen über den Raspberry Pi an den Lokomotion-Controller weitergegeben werden. Dazu soll ein Python-Programm dienlich sein. In diesem sollen, abhängig von den durch das PHP-Skript übermitt n Werten, Funktionen ausgeführt werden, die über die zuvor beschriebene serielle Schnittstelle die entsprechenden Bewegungsbefehle der Lokomotion-Betriebssoftware aufrufen. Ist dies einmal programmiert, kann nun mit der Einrichtung der PHP-Webpage fortgefahren werden. Die Oberfläche soll sich, um den Programmieraufwand möglichst gering zu halten, zunächst auf die in Abbildung 16 dargestellten Elemente beschränken. Nach welchen Mustern speziell die Schaltflächen einer solchen Webpage mit PHP zu programmieren sind, kann in ausführlicher Form samt Quelltext in weiterführenden Werken, beispielsweise in dem von Kofler, Kühnast und Scherbeck (2016, S. 1048f.), nachgelesen werden.

4.1.3 Reflektion des Beispiels

Da nun alle nötigen Schritte zum Bau des Überwachungsroboters durchgeführt wurden, kann der Roboter in den in der Problemstellung genannten Aufgabenfeldern eingesetzt werden. Im Folgenden

soll das Vorgehen anhand der Kriterien Komplexität, Aufwand und Erweiterbarkeit beurteilt werden. Schon während der Konstruktion des Roboters fallen einige Schwachstellen auf, die sich jedoch durch kleine Veränderungen im Aufbau beheben lassen. Beispielsweise könnten dem Benutzer mit dem Gebrauch des aktuelleren Modells der Raspberry Pi Foundation, dem Raspberry Pi 4, neben zwei USB-2.0- auch zwei USB-3.0-Anschlüsse zur Verfügung stehen (The Raspberry Pi Foundation 2019). Damit keine Beeinträchtigungen in der Leistung der Kamera akzeptiert werden müssen, ist die Verwendung des Modells 4 bei diesem Projekt also ratsam. Grundsätzlich erscheint diese Beispielkonstruktion in einigen Aspekten sehr wandelbar. So ist auch der Einsatz der Tiefenkamera in dieser Anwendung zu hinterfragen, da sie durch den für dieses Projekt irrelevanten, großen Funktionsumfang auch eine umfangreichere Installation erfordert. Der dadurch entstehende Mehraufwand und die Bildung potentieller Fehlerquellen lassen sich durch den Einsatz einer herkömmlichen Webcam mit USB-2.0-Anschluss vermeiden. Ein weiteres mögliches Problem könnte die mangelnde Energieversorgung der Kamera oder des Roboters darstellen. Werksseitig sind die USB-Anschlüsse eines Raspberry Pi auf eine maximale Stromstärke von 600 mA begrenzt (Kofler u. a. 2016). Diese kann zwar in den Einstellungen des Miniaturcomputers auf die doppelte Höhe angehoben werden, jedoch sollte die im Werkszustand vorfindbare Stromversorgung für das Betreiben herkömmlicher USB-Webcams (Stromverbrauch kleiner als 200 mA) völlig ausreichend sein. Obwohl die Tiefenkamera mit einem vom Hersteller genannten Stromverbrauch von 700 mA diesen Wert übertrifft (Intel 2019a), konnte die Kamera, in den im Rahmen dieser Arbeit durchgeführten Experimenten, auch mit der Standardeinstellung des Raspberry Pi betrieben werden. Ein geringerer Stromverbrauch der verbauten Kameraeinheit hätte zudem positive Auswirkungen auf die Laufzeit des Roboters im Akkubetrieb.

In der hier dargestellten Form ist der zu betreibende Aufwand und die Komplexität dieses Projektes gering: Viele Schritte in der oben genannten Anleitung beschränken sich auf das Einrichten eines Raspberry Pi und das Installieren essentieller Softwarepakete auf dem Mikrocomputer. Einmal abgeschlossen erstreckt sich der übrige, in Eigenleistung zu erbringende Programmieraufwand lediglich auf das implementieren der PHP-Webpage und der dazugehörigen Python-Anwendung. Hieraus lassen sich die für die Umsetzung dieses Projektes erforderlichen Kompetenzen des bearbeitenden Studierenden ableiten. So sollten diese in der Lage sein, sicher mit Python- und PHP-Quellcode umzugehen, die Grundfunktionen eines Raspberry Pi zu verstehen und mit GitHub-Repositorien arbeiten zu können. Aufwand und Komplexität dieses Projektes können jedoch noch

weiter variiert werden. So ist der Überwachungsroboter in dieser Ausführung um zahlreiche zusätzliche Funktionen erweiterbar und kann so an sein final bestimmtes Einsatzfeld angepasst werden. Denkbar wäre der Einsatz des Überwachungsroboters in Krisengebieten zur Aufklärung. Hierfür könnte es sich als nützlich erweisen, einige Modifikationen an diesem vorzunehmen. Um den Roboter auch über größere Distanzen sicher durch unbekanntes Territorium manövrieren zu können, sollte dieser mit einem Mobilfunkmodem ausgerüstet sein, sodass weltweit auf den Roboter zugegriffen werden kann, ohne dass sich dieser in einem lokalen Netzwerk befinden muss. Der einfachste Weg dies zu realisieren, ist über einen Surfstick eines beliebigen Internetanbieters, der mit einem der USB-Anschlüsse des Raspberry Pi verbunden wird. Damit dann auf den Raspberry Pi und somit den Roboter ebenfalls über einen Webbrowser zugegriffen werden kann, benötigt der Raspberry Pi anstelle seiner dynamischen eine statische IP-Adresse, unter der er den Webserver hostet. Sobald der Raspberry Pi an ein lokales Netzwerk angeschlossen wird, weist der Router ihm eine dynamische IP Adresse zu. Diese kann sich mit jedem erneuten Verbinden ändern. Damit die IP-Adresse konstant und auch von außerhalb des lokalen Netzwerkes zugreifbar bleibt, muss dem Raspberry Pi einmalig eine statische IP-Adresse zugeordnet werden. Entsprechende Anleitungen hierfür finden sich in weiterführender Literatur (Kofler u. a. 2016, S. 176ff.). Ist eine solche Adresse einmal eingerichtet, kann die Webpage des Roboters weltweit eingesehen und gesteuert werden. Die Einrichtung einer Sicherheitsbarriere, wie ein zuvor benötigtes Kennwort, ist ratsam, um Missbrauch zu verhindern. Abgesichert lässt sich der Roboter mit statischer Adresse auch optimal im Heimnetzwerk als Überwachungsroboter einsetzen. Der Benutzer ist dann, auch wenn er sich nicht im Heimnetzwerk befindet, über ein internetfähiges Gerät in der Lage, das Umfeld des Roboters zu erkunden. Des Weiteren kann der Umfang der Funktionen des Roboters und somit auch der Aufwand der Implementierung durch das Einsetzen von zusätzlichen Servomotoren gesteigert werden. Additionale Motoren können beispielsweise ein Schwenken und Neigen des Kameramoduls ermöglichen. Dafür müssen passende Gelenke und Motorenhalterungen entworfen und mit dem 3D-Drucker gefertigt werden. Mit dem dreidimensionalen Konstruieren der benötigten Teile steigen, neben dem Aufwand, auch die von den Studierenden benötigten Kompetenzen. Über die hierfür erforderlichen zusätzlichen Servomotoren verfügt das Fachgebiet bereits. Mit Hilfe dieser kann alternativ auch ein Greifarm den Überwachungsroboter erweitern. Die Konstruktion der dafür benötigten Elemente und das Erweitern der PHP-Webpage um zusätzliche Steuerfunktionen ist hier ebenso notwendig wie bei einem Umbau der Kamerahalterung und lässt den zu betreibenden Aufwand in etwa gleichem Maße anwachsen. Die zusätzlichen Motoren können dabei wahlweise an

den Raspberry Pi oder das Lokomotion-Controller-Board angeschlossen werden. Letzteres sollte sich in der Implementierung allerdings als komplizierter erweisen, da hierfür zusätzliche Funktionen in der Betriebssoftware des Laufroboters eingefügt werden müssten. Ist der Roboter mit einem funktionsfähigem Greifarm ausgestattet, bietet sich die Implementierung von Reaktionen auf Bewegungen an. Denkbar ist eine Abwehrroutine, die gestartet wird, sobald der in Alarmbereitschaft versetzte Roboter Bewegungen in seinem Blickfeld bemerkt. Innerhalb dieser Routine wäre dann ein Verfolgen des sich bewegenden Objekts denkbar. Zur Auslösung dieser Routine kann die bereits installierte Software *Motion* genutzt werden. In ihren Einstellungen kann eine Empfindlichkeit justiert werden, mit der die Software auf Änderungen im Bild der Kamera reagiert (Kofler u. a. 2016). Denkbare Reaktionen wären, neben dem Auslösen eines Abwehrverhaltens des Roboters, auch das Speichern des Bilds oder eines Videomitschnittes, gefolgt vom Versenden einer Information an den Benutzer über eine potentiell imminente Gefahr. Sollte der Roboter eher im Außenbereich als Aufklärungsroboter eingesetzt werden, ist dessen Erweiterung um ein GPS-Modul ebenfalls ratsam. Da die UART-Schnittstelle des Raspberry Pi bereits mit dem Lokomotion-Controller verbunden werden muss, stehen zur Verbindung eines solchen Moduls lediglich die noch nicht belegten USB-Anschlüsse zur Verfügung. Wie das GPS-Signal auf einer Karte mit Hilfe der Google-Maps-Programmierschnittstelle (API) mit *JavaScript* in Webpages eingebunden werden kann, ist in der Dokumentation dieser API nachlesbar (Google 2019). Da die in diesem Beispiel erstellte Webpage jedoch in PHP geschrieben wurde, bedarf es einiger Anpassungen am Quellcode. Die Einbindung der GPS-Verfolgung des Roboters ist daher von sehr hoher Komplexität geprägt. Um auch den Umfang der den Studierenden vermittelten Kompetenzen zu erhöhen, ist bei diesem Beispiel auch das Programmieren einer Anwendungssoftware für mobile Endgeräte denkbar, über die der Benutzer dann in der Lage ist, auf sämtliche Funktionen des Roboters zuzugreifen. Mit kostenfreien Entwicklerplattformen wie dem *MIT App Inventor* können individualisierte Anwendungen für mobile Endgeräte modular und somit zeitsparend aufgebaut werden (Massachusetts Institute of Technology 2020). Die hier behandelte Implementierungsmöglichkeit des Roboters in das Lehrkonzept des Fachgebiets zeichnet sich insbesondere durch ihre Vielzahl an Erweiterungsmöglichkeiten aus. So kann die ursprüngliche Problemstellung bedenkenlos um die zusätzlich erwähnten Funktionen erweitert werden. Bei der Auswahl dieser Funktionen sollten die dabei vermittelten Kompetenzen beachtet und danach gezielt ausgewählt werden, um den Studierenden ein möglichst zweckmäßiges Arbeiten zu garantieren. Ohne eine nachträgliche Anpassung der Aufgabenstellung ist der aufzubringende Aufwand in diesem Projekt begrenzt. Die zu lösenden Aufgaben können in wenigen

Stunden effektiver Arbeitszeit abgearbeitet werden. Ist das zur Bearbeitung benötigte Wissen einmal akkumuliert, so schätzen sich die übrigen Aufgaben auf einen Umfang von weniger als 10 Stunden. Die zu bearbeitenden Aufgaben beschränken sich bei diesem Beispielprojekt, neben dem Verbinden der verwendeten Hardware (Kamera, Raspberry Pi und optional zusätzliche Servomotoren) mit dem Roboter, ausschließlich auf Programmierarbeiten am Raspberry Pi. Studierende sind also in der Lage, beinahe unabhängig von einem Zugang zum Laufroboter, diese Aufgabenstellung zu bearbeiten, sofern ihnen ein Raspberry Pi und eine Kamera zur Verfügung gestellt wird. So wäre es mit genügend Raspberry Pis und Kameras möglich, diese Problemstellung von einigen Studierendenteams parallel bearbeiten zu lassen. Es wird empfohlen, den Teams in regelmäßigen Abständen es zu ermöglichen mit Implementierungsversuchen ihre Software testen zu können. Da an dem Laufroboter selbst keine Programmierarbeiten durchgeführt werden, besteht die Gefahr, ohne eine entsprechende Erweiterung der Problemstellung, dass das Vermitteln einiger Kompetenzen, wie der Umgang mit und die Programmierung von Sensorik und Aktorik (insbesondere das für die Programmierung von Mikrocontroller-Boards relevante Erlernen von C++) ausbleibt.

4.2 Konstruktion einer bildverarbeitenden Laufmaschine

Im Folgenden wird ein zweites Anwendungsbeispiel des Roboters im Lehrbetrieb des ITPLs entwickelt. Hierzu wird ebenfalls zunächst eine Handlungsanweisung formuliert, ein möglicher Lösungsweg vorgestellt und das Beispiel abschließend reflektiert. Bei dieser Reflektion sollen die Faktoren Erweiterbarkeit, Komplexität und Aufwand des Projektes als Bewertungskriterien dienlich sein. In diesem Beispiel soll der Fokus auf der Implementierung maschinellen Sehens liegen.

4.2.1 Formulierung der Problemstellung

Die in diesem Kapitel entwickelte Laufmaschine soll mit Hilfe maschinellen Sehens dazu in der Lage sein, in ihrer Umgebung autonom zu navigieren und auf das Geschehen in dieser reagieren zu können. Maschinelles Sehen lässt sich in vielfältigen Anwendungsfeldern integrieren (Szeliski 2010). Der relevante Anwendungsbereich beschränkt sich bei dem hier konzeptionierten Roboter im Allgemeinen jedoch auf die Navigation und Identifizierung von Objekten. Der Roboter soll im Detail bestimmte Gegenstände in seinem Sichtfeld ausfindig machen und diesen folgen können, ohne dabei mit anderen Objekten zu kollidieren. Ruht ein Gegenstand hingegen, soll der Abstand zu diesem geschlossen werden. In diesem Beispiel soll der Roboter einem roten Ball folgen können, theoretisch sollen aber auch andere Gegenstände verfolgbar sein. Eine solche Fähigkeit ist vielseitig einsetzbar.

So ist mit einigen anderen Funktionserweiterungen ein Einsatz des Laufroboters beispielsweise als Erntemaschine vorstellbar, bei dem er während des Durchwanderns der Anbauflächen nur die reifen, roten Früchte aufsammeln oder pflücken würde. Aber auch in einem industriellen Umfeld ist der Einsatz einer solchen Laufmaschine denkbar. Hier könnte diese als Reinigungseinheit unterstützen, indem sie den Hallenboden einer Fabrik nach Abfall absuchen und diesen in die dafür vorgesehenen Abfallsammelstellen transportieren würde. Im Detail sind zu diesem Zweck die folgenden Aufgaben sukzessiv zu bearbeiten:

1. Ausstattung des Roboters mit für die Anwendung genügender Rechenkapazität
2. Integration eines geeigneten Bildsensors in Hard- und Software des Roboters
3. Implementierung eines Bildverarbeitungsalgorithmus zur Identifizierung und Differenzierung von Gegenständen und Programmierung der darauf abgestimmten Reaktionen

4.2.2 Vorstellung einer möglichen Lösung

Im Folgenden wird eine mögliche Lösungsvariante der oben aufgeführten Handlungsanweisung präsentiert. Die bearbeitenden Studierenden können diese Empfehlung als Hilfestellung zur Entwicklung einer eigenen Umsetzung der Modifikation des Roboters nutzen.

Ad 1.

Zunächst muss der Roboter für die Integration der Bilddatenverarbeitung vorbereitet werden. Die dafür benötigte Rechenleistung steht dem Lokomotion-Controller, aufgrund eines fehlenden Grafikprozessors, nicht zur Verfügung. Daher ist eine Erweiterung des Roboters mit externen Modulen unverzichtbar. Sollen Kameramodule mit bereits integrierter Bildverarbeitungshardware, wie der *PixyCam*, eingesetzt werden, so ist auch eine Erweiterung des Roboters mit dem Arduino UNO ausreichend. Hier wird das Kameramodul an dem *In-Circuit-Serial-Programming*-Pins (ICSP) des Arduino angeschlossen (PixyCam 2020). Wird jedoch eine herkömmliche Webcam oder die Tiefenkamera des ITPLs als Bildsensor verwendet, so ist die Erweiterung des Roboters um den Raspberry Pi notwendig, da für die Verbindung dieser ein USB-Anschluss erforderlich ist. Wie der Mikrocomputer an das Lokomotion-Controller-Board anzuschließen ist, zeigt Abbildung 15. Die Vorgehensweise bei der Initialisierung der seriellen Schnittstelle und somit dem Ermöglichen der Kommunikation zwischen Raspberry Pi und Lokomotion-Controller kann in Kapitel 4.1.2 (Ad 1.)

nachgelesen werden. Da das Fachgebiet lediglich über ein Kameramodul mit einer USB-Schnittstelle verfügt, muss der Raspberry Pi an dem Lokomotion-Controller-Board angeschlossen werden.

Ad 2.

Im nächsten Schritt soll die Integration eines geeigneten Bildsensors in die Soft- und Hardware des nun erweiterten Roboters durchgeführt werden. Grundsätzlich lassen sich an den Raspberry Pi auch Kameras ohne einen USB-Anschluss, wie beispielsweise die PixyCam oder das Kameramodul der Raspberry Pi Foundation, an dem Kameraanschluss des Einplatinencomputers oder über eine serielle Schnittstelle (UART, I2C, SPI) anschließen (The Raspberry Pi Foundation 2020b). Die vom Fachgebiet zur Verfügung gestellte Tiefenkamera wird jedoch mit einem USB-Kabel mit dem Raspberry Pi verbunden. Für das Anbringen der Kamera an den Roboter können auch hier die CAD-Modelle aus dem elektronischen Anhang (A) genutzt werden, um eine passende Halterung zu fertigen. Die zur Nutzung der Kamera benötigte Installation des von Intel bereitgestellten SDK wird bereits in Kapitel 4.1.2 (Ad 2.), beziehungsweise in der im elektronischen Anhang (B) befindlichen Anleitung beschrieben.

Ad 3.

Zur Verarbeitung der durch die Tiefenkamera übermittelten Bilder wird ein auf dem Raspberry Pi laufendes Programm benötigt. Mit der Benutzung der Intel Tiefenkamera stehen zwei Möglichkeiten zur Auswahl, ein Programm zur Identifizierung des Beispielobjekts (hier ein roter Ball) zu entwickeln. Das Programm kann einerseits mit den Funktionen des Softwareentwicklungspakets von Intel oder andererseits mit dem Open-Source-Programm *OpenCV* programmiert werden. Wie dieses Programm auf dem Raspberry Pi zu installieren ist, zeigt das Werk von Cicolani (2018, S. 299ff.). Das Programmieren mit dem SDK von Intel kann den Funktionsumfang des Roboters deutlich steigern, da mit diesem auch die Tiefenfunktionen der Kamera genutzt werden können. Allerdings ist zu beachten, dass die hiermit entwickelte Software ausschließlich mit den Tiefenkameras der Intel RealSense-Reihe kompatibel ist. Bei der Nutzung von mit OpenCV erstellten Programmen lassen sich in der bildverarbeitenden Laufmaschine neben der Tiefenkamera auch herkömmliche Webcams einsetzen. Dieses Programm zur Bilddatenverarbeitung lässt sich dann wahlweise in Python oder C++ programmieren. Wird sich für die Möglichkeit entschieden, das Programm mit dem SDK von Intel aufzubauen, ist es ratsam, in C++ zu programmieren. Dies wird damit begründet, dass die zahlreichen

Anwendungsbeispiele des SDK in der Dokumentation von Intel überwiegend in C++ verfasst sind. Dadurch lassen sich die Beispielprogramme als Vorlage nutzen, ohne sie zuvor in einen Python-Quellcode übersetzen zu müssen. Um auf dem Raspberry Pi auch C++ Programme ausführen zu können, bedarf es diesem eines Compilers für C++. Wie ein solcher Compiler auf dem Mikrocomputer installiert wird, kann in weiterführender Literatur nachgeschlagen werden (Follmann 2018, S. 73ff.). Wie bei der genauen Programmierung eines solchen Programmes ohne die Nutzung des SDKs von Intel vorzugehen ist, zeigt weiterführende Literatur (Cicolani 2018, S. 333ff.). In dem Programm sind verschiedene Funktionen zu etablieren: Der Roboter soll zunächst seine Umgebung nach dem roten Ball absuchen, ohne dabei mit Gegenständen zu kollidieren. Hat er das Objekt einmal über das Bild der Kamera ausfindig machen können, soll er dem Objekt folgen. Dies soll mit der Prämisse, den Ball stehts im Mittelpunkt des Kamerabilds zu halten, erreicht werden. Weicht das Objekt dem Bildzentrum zur linken Seite aus, hat der Roboter seine Position durch eine Linksdrehung anzupassen. Analog gilt dies für eine Bewegung des Balls auf die rechte Seite des Bildmittelpunktes. Erfolgt eine Abweichung des Beispielobjekts aus dem Bildzentrum nach oben, soll der Roboter sich vorwärts- und für den Fall einer Abweichung nach unten rückwärtsbewegen. Das Bildzentrum muss gegebenenfalls abhängig von den Dimensionen des Balls und der Pose der installierten Kamera nachjustiert werden.

4.2.3 Reflektion des Beispiels

Grundsätzlich weißt dieses Beispielprojekt, speziell in den zutreffenden Vorbereitungen des Roboters, einige Similaritäten zu dem in Kapitel 4.1 behandelten Überwachungsroboter auf. Es kann jedoch aufgrund seiner erheblichen Komplexität, speziell in Hinblick auf die Programmierung einer OpenCV- beziehungsweise RealSense-Anwendung als ein eigenständiges Beispiel behandelt werden. Die Umsetzung des hier beschriebenen Roboters zeichnet sich durch einen hohen Grad an Komplexität und Umfang aus. Es bedarf einiger Programmiererfahrung. Des Weiteren sind erste Grundkenntnisse in der Entwicklung eigener Bildverarbeitungsalgorithmen und deren Implementierung mit dem RealSense SDK hilfreich. Alternativ sind auch andere Programme wie OpenCV hierfür verwendbar. Die Programmierung mit dieser Software hat den prominenten Vorteil, dass Studierende auch ohne einem ständigen Zugriff auf die Intel Tiefenkamera die von ihnen entwickelten Algorithmen mit herkömmlichen Webcams testen könnten, was auch ein paralleles Bearbeiten dieser Aufgabenstellung durch mehrere Studierendenteams als denkbar erscheinen lässt. Außerdem erscheinen die bei der Programmierung von OpenCV-Anwendungen erworbenen Kompetenzen aufgrund des höheren Verbreitungsgrades und der Vielseitigkeit der Verwendbarkeit

von OpenCV für Studierende als bedeutsamer. Deshalb wird nachträglich empfohlen, anstelle des SDKs von Intel OpenCV für die Programmierung der Bilddatenverarbeitung zu verwenden. Da die Integration der zusätzlichen Funktionen der Tiefenkamera den Umfang dieses Projektes ohnehin enorm ansteigen lässt, könnte erwogen werden, diese Kamera gegen eine herkömmliche Webcam auszutauschen. Dies hätte zusätzlich eine längere Laufzeit des Roboters aufgrund des geringeren Stromverbrauchs zur Folge. Mögliche Erweiterungen dieses Projektes, die den Stromverbrauch ebenfalls steigern ließen, lassen sich zahlreiche finden. Denkbar wäre beispielsweise die Installation zusätzlicher Abstandssensoren, zur Unterstützung der kollisionslosen Navigation des Roboters durch sein Umfeld. Eine weitere Möglichkeit zur Vergrößerung des für den Roboter einsehbaren Bereichs wäre die Implementierung zweier zusätzlicher Servomotoren, welche die Kamera dreh- beziehungsweise neigbar machen könnten. Die Erweiterbarkeit des Funktionsumfangs dieses Beispielprojektes ist demnach gegeben. Für Studierende, die bereits mit OpenCV gearbeitet haben, erscheint der bei der Bearbeitung der Aufgabenstellung zu betreibende zeitliche Aufwand als limitiert. Müssen die Studierenden aufgrund mangelnder Erfahrung eine intensive Literaturrecherche betreiben, um den Umgang mit OpenCV in Verbindung mit dem Laufroboter zu erlernen, kann sich der Aufwand rasant potenzieren. Aufgrund der steigenden Anzahl an möglichen Einsätzen in der Automobil-, Produktions- und Fertigungstechnik sowie der Robotik und der stetig sinkenden Kosten der Hardware sind Bildsensoren und die zur Auswertung der Bilddaten benötigten Systeme aus der heutigen Industrie kaum noch wegzudenken (Winner u. a. 2009). Diese Technologien sind somit als nützliches Werkzeug auch in der Fabrikautomation und somit dem vom ITPL angebotenen Fachlabor relevant. Dieses behandelt das Thema Bildverarbeitung aktuell noch nicht (ITPL 2020). Der Roboter bildet somit eine geeignete Plattform, diese Inhalte praktisch in dem Labor zu integrieren, indem er die Programmierung von Bildverarbeitungsalgorithmen mit der Steuerung der Aktorik (in diesem Fall die zur Fortbewegung des Roboters benötigten Motoren) vereint. Von der Anwendung dieses Implementierungsbeispiels wird jedoch trotz einer prinzipiellen Konformität mit dem Lehrplan des Labors aufgrund der benötigten Programmiervorkenntnisse bei der Entwicklung von bilddatenverarbeitenden Algorithmen speziell mit Hilfe von OpenCV eher abgeraten. Studierenden, die über diese Kompetenzen verfügen, soll eine Bearbeitung jedoch trotzdem ermöglicht werden.

4.3 Konstruktion eines autonomen Geländeroboters

Im letzten Beispiel zur Implementierung des Laufroboters in das Lehrkonzept des Fachgebiets soll der Fokus auf der Modifizierung des Roboters mit dem Ziel der Autonomie dessen liegen. Anders als die

zuvor erwähnten Projekte soll diese Konstruktion auch prototypisch durchgeführt und dokumentiert werden, um die Einsetzbarkeit des Roboters im Fachlabor auch praktisch zu erproben. Zuvor wird jedoch die konkrete Aufgabenstellung einschließlich aller gewünschten Funktionen des Geländeroboters formuliert und eine dazu passende mögliche Umsetzungsweise vorgestellt. Abschließend wird auch dieses Projekt hinsichtlich seines Umfangs, seiner Komplexität und Erweiterbarkeit sowie der zur Umsetzung benötigten Kompetenzen beurteilt. Die Dokumentation der Umsetzung soll dabei der Reproduzierbarkeit des Ergebnisses dienlich sein und einen Einblick auf das konkrete Arbeiten am Roboter gewähren.

4.3.1 Formulierung der Problemstellung

Wie in den beiden Beispielen zuvor stellt auch hier der unveränderte Laufroboter nach dem Zusammenbau die Ausgangssituation dar. Im Folgenden werden die gewünschten zusätzlichen Fähigkeiten beschrieben, die zu implementieren sind. Der Roboter soll nach Abschluss der Modifikation dazu in der Lage sein, sich selbstständig, auch in unbekanntem Terrain, störungsfrei fortzubewegen. Im Detail bedeutet dies, dass er fähig ist, Hindernisse zu um- beziehungsweise übergehen. Falls ein sich auf der Laufbahn des Roboters befindliches Hindernis nicht überwunden werden kann, soll der Roboter zurückweichen, einen alternativen Weg ausfindig machen und begehen. Zu den möglichen Hindernissen, auf die der Roboter stoßen kann, zählen beispielsweise Unebenheiten und abrupte Steigungen des Untergrunds, Sackgassen oder enge Schächte. Die potentiellen Einsatzfelder eines solchen Roboters sind nahezu grenzenlos. So kann der Roboter in weiterentwickelter Form unter anderem Einsatz in der automatisierten Produktion finden. Denkbare Aufgaben wären hier beispielsweise das Anliefern von Rohteilen und der Abtransport von Fertigteilen. Dabei ist ein zuverlässiges Meiden von Hindernissen unabdingbar. Ein weiteres interessantes mögliches Aufgabenfeld des autonomen Geländeroboters kann die Erkundung und Kartographierung unbekannter, für den Menschen unzugänglicher Umgebungen sein, für die eine Steuerung des Roboters durch Benutzer aus verschiedenen Gründen nicht umsetzbar oder nur wenig sinnvoll erscheint. Als konkrete Einsatzfelder kann hier das Vermessen von komplexen Höhlensystemen oder gar Planeten genannt werden. Dass es dem Geländeroboter für den Einsatz in diesen Anwendungsgebieten weiterer Modifikationen bedarf, die in dieser Arbeit jedoch nicht behandelt werden können, ist evident. Um einen Roboter mit diesen Anforderungen zu konstruieren, lassen sich folgende notwendige Arbeitsschritte identifizieren:

1. Erweiterung der Rechenleistung und Anzahl an frei zur Verfügung stehenden Steckplätzen für Erweiterungen des Roboters durch den Anschluss eines geeigneten Benutzer-Boards
2. Ausrüsten des Roboters mit der zur Erkennung von Hindernissen erforderlichen Sensorik und Aktorik zur Steuerung von Sensormodulen
3. Entwicklung eines Programmes, mit dem der Roboter die geeigneten Laufparameter, wie Körper oder Schritthöhe, wählen kann, um den erkannten Hindernissen auszuweichen
4. Entwerfen einer Beispielumgebung, in der die Funktionen des Roboters exemplarisch getestet werden können

4.3.2 Vorstellung einer möglichen Lösung

Auch dieses Beispiel kann ebenso wie die Projekte „Überwachungsroboter“ und „bildverarbeitende Laufmaschine“ auf verschiedene Art und Weisen umgesetzt werden. Im Folgenden wird bei der Umsetzung der oben genannten zu lösenden Aufgaben stets nur eine Variante ausführlich behandelt. Bestehen alternative Lösungsansätze zu den einzelnen Herausforderungen, werden diese jedoch ebenfalls erwähnt und warum sich gegen eine Umsetzung dieser Ansätze entschieden wurde. Die nun schrittweise behandelten Lösungen der Aufgaben aus der Problemstellung sollen als Grundlage der späteren prototypischen Umsetzung des Geländeroboters dienen.

Ad 1.

Das Lokomotion-Controller-Board besitz neben den Anschlussleisten für Benutzer-Boards über insgesamt 9 frei verwendbare Anschlüsse (SU1 bis SU3 und SA0 bis SA5). Grundsätzlich ist es also möglich, die für die Implementierung dieses Projektes benötigte Sensorik und Aktorik direkt mit dem Board zu verbinden. Durch den geplanten Einsatz eines jeden zusätzlichen Sensors oder Aktors wird jedoch mehr Rechenleistung des Mikrocontrollers benötigt. Damit dieser hinzukommende Rechenaufwand die Leistung des Lokomotion-Controllers nicht beeinflusst, soll hier eine geeignete Erweiterung des Roboters durchgeführt werden. Ein passendes Benutzer-Board soll dann die Sensordaten auswerten und die zusätzliche Aktorik steuern, da der Lokomotion-Controller in seiner Motion-Betriebssoftware bereits die kritischen Laufalgorithmen des Roboters berechnet. Das Benutzer-Board soll dann, basierend auf den ausgewerteten Daten, Bewegungsentscheidungen treffen und diese über eine serielle Schnittstelle an den Lokomotion-Controller weitergeben. Diese Struktur hat zusätzlich den Vorteil, dass die Programme der Betriebssoftware des Lokomotion-Controllers nicht editiert werden müssen. Der Benutzer hat folglich nur das Erweiterungsboard zu

programmieren. Da die in der Problemstellung erwähnten Grundfunktionen des hier zu konstruierenden Geländeroboters keine Internet- oder Bluetooth-Funkverbindung vorsehen, können diese Spezifika bei der Wahl des passenden Benutzer-Boards ignoriert werden. Es wird sich hier für die Erweiterung des Roboters mittels des Arduino UNO-Mikrocontroller-Boards entschieden. Dieses zeichnet sich gegenüber der übrigen zur Verfügung stehenden Benutzer-Boards durch einen geringen Stromverbrauch im Leerlauf von 50 mA (verglichen zu 400 mA am Raspberry Pi) und der hohen Anzahl frei verwendbarer Anschlüsse (freie GPIOs) zur Verbindung externer Geräte (insgesamt 18, also 7 mehr als beim NodeMCU) aus (Arduino 2020b) (Conrad Electronic SE 2020) (Espressif 2019) (The Raspberry Pi Foundation 2020b). Wie genau das Board an den Roboter angeschlossen wird, wurde bereits in Abbildung 14 (A) dargestellt. Es ist jedoch ratsam, aufgrund beschränkter Zugänglichkeit des USB-Anschlusses des Benutzer-Boards, den Arduino UNO erst nach dessen Programmierung auf dem Lokomotion-Controller-Board zu installieren. Der Einplatinencomputer soll über die Arduino IDE programmiert werden. Der Quellcode des auf das Benutzer-Board hochzuladenden Programmes hat neben der später behandelten Sensordatenverarbeitung und Aktorsteuerung die Initialisierung der seriellen Schnittstelle zu enthalten. Ist diese Schnittstelle nicht korrekt implementiert, ist der Arduino UNO nicht in der Lage, mit dem Lokomotion-Controller zu kommunizieren und die Bewegungsbefehle dessen Betriebssoftware aufzurufen. Nach welchem Muster diese serielle Schnittstelle zu programmieren ist, zeigt das Beispielprogramm „Moving_2“ des sich im elektronischen Anhang (F) befindlichen Downloadpakets.

Ad 2.

Damit der Roboter auch ohne menschliches Eingreifen Hindernissen ausweichen kann, muss er diese zu aller erst als solche erkennen können. Hierfür soll der Laufroboter mit Sensoren ausgestattet werden, die seine Umgebung permanent erfassen und ihm, die für das Manövrieren nützlichen, Informationen möglichst in verzögerungsarm übermitteln können. Vorstellbar ist auch hier der Einsatz der Tiefenkamera. Um diese aber mit dem Roboter zu verbinden, bedarf es dem Anschluss eines Raspberry Pi. Dieses Beispielprojekt soll jedoch den Arduino UNO verwenden. Auch wenn der simultane Einsatz beider Erweiterungen prinzipiell realisierbar erscheint, ist er aufgrund des erhöhten Energieverbrauchs nicht ratsam. Darüber hinaus genügt für die hier möglichen Umfelder der Einsatz alternativer Sensoren, die keinen Mikrocomputer mit bildverarbeitungsfähiger Hardware benötigen. Denkbar ist der Einsatz von Ultraschall- und Infrarotabstandssensoren, um die Anwesenheit und Abstände der in Reichweite befindlichen Gegenstände zu erfassen. Grundsätzlich

lässt sich festhalten, dass zur Ermittlung genauer Abstände Ultraschallsensoren besser geeignet sind als Infrarotsensoren, diese jedoch in bestimmten Ausführungen, wie zum Beispiel Sharps Infrared-Proximity-Sensor GPD2D120, einen größeren Bereich erfassen können (Brühlmann 2019). Da das Fachgebiet lediglich über Ultraschallsensoren des Modells HC-SR04 verfügt, wird im Folgenden auf die Implementierung dieser eingegangen. Wie alternativ Infrarotabstandssensoren in diesem Projekt zu verwenden sind, kann in weiterführender Literatur nachgeschlagen werden (Brühlmann 2019, S. 234f.). Neben dem Gebrauch von Abstandssensoren sind auch Fotowiderstände dazu in der Lage, ausgewählte Hindernisse, wie zum Beispiel Passagen mit tiefen Decken, zu identifizieren. Der Einsatz von Fotowiderständen ist jedoch aufgrund ihrer Abhängigkeit von konstanten Belichtungsverhältnissen nur selten ratsam. Nun muss analysiert werden, wie viele der Ultraschallsensoren am Roboter zu montieren sind. Dabei gilt es aufgrund der limitierten Energieversorgung, den begrenzt verfügbaren Sensoren und freien Anschlüssen des Benutzer-Boards die Zahl der eingesetzten externen Geräte zu minimieren. Da die hier eingesetzten Ultraschallsensoren üblicherweise den messbaren Bereich lediglich mit einem Messkegel von 15 ° abdecken können (Sparkfun 2020), benötigt es jedoch einiger Sensoren, um das gesamte Umfeld des Roboters permanent wahrzunehmen. Daher gilt es, die entscheidenden Bereiche dieses Umfeldes zu identifizieren, um die Zahl der benötigten Sensoren zu reduzieren. Der Roboter soll sich nach dem Start des Programmes zunächst vorwärtsbewegen und seine Laufrichtung erst ändern, falls er in seinem Laufweg ein Hindernis feststellt. Somit ist das Abdecken des Frontbereichs mit Ultraschallsensoren zu priorisieren.

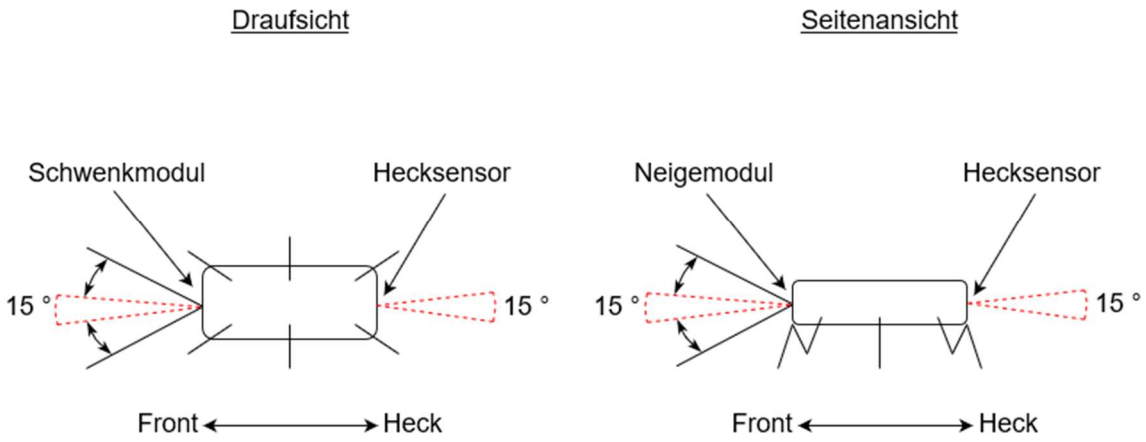

Abbildung 17: Einsehbares Umfeld der drei Ultraschallsensoren des Roboters

Damit nicht für jeden 15 ° weiten Bereich ein eigener Sensor installiert werden muss, ist die Konstruktion von einem Schwenk- und Neigemodul ratsam. Solche Module bestehen dann jeweils aus einem Ultraschallsensor, einem ebenfalls vom Fachgebiet zur Verfügung gestellten Servomotor und einer geeigneten Halterung, um die beiden Komponenten mit dem Roboter verbinden zu können. Sie sind nun dazu in der Lage, mit nur einem Sensor durch häufiges Messen und einem stetigen Schwenken zwischen den beiden Anschlägen des Servomotors, einen größeren Bereich zu erfassen. Die individualisierten Halterungen sollen als STL-Dateien konstruiert und anschließend mit dem 3D-Drucker des Fachgebiets realisiert werden. Um neben den Informationen über das frontale und seitliche Umfeld auch Informationen über den rückseitigen Bereich am Heck des Roboters zu erhalten, soll dort ein statischer Ultraschallsensor samt Halterung am Laufroboter befestigt werden. Dieser soll unter anderem sicherstellen, dass der Roboter sich mit einer Rückwärtsbewegung aus kritischen Situationen, wie einem Verirren in Sackgassen, befreien kann. Die drei so aufgebauten Module decken dann die in Abbildung 17 dargestellten Bereiche ab. Angeschlossen an den Arduino UNO sind sie somit im Programm des Benutzer-Boards zu implementieren.

Ad 3.

Die Implementierung der nachträglich angebrachten Sensorik und Aktorik des Laufroboters geschieht durch das Einbinden in den später auf den Mikrocontroller hochzuladenden Programmcode. Dieses Programm soll mit Hilfe der Sensoren dazu im Stande sein, Hindernisse als solche zu erkennen und gegebenen Falls mit einem Anpassen der Laufparameter wie Laufrichtung und -geschwindigkeit, als auch der Schrittparameter, wie Schrittmuster, -höhe und -weite, zu reagieren. Wie der Roboter genau seine Laufentscheidungen zu treffen hat, ist in einem reduzierten Struktogramm, welches dem elektronischen Anhang (D) beigefügt wurde, festgehalten. Ein solches Struktogramm (auch Nassi-Shneidermann-Diagramm genannt) ermöglicht es, lineare Kontrollstrukturen eines Programmes grafisch darstellen zu können (Balzert 2010). Aufbauend auf diesem kann das Programm strukturiert programmiert werden. Wie genau die einzelnen Funktionen und Abfragen dabei auszugestalten sind, kann vom Programmierer frei interpretiert werden. Jedoch sei zu erwähnen, dass dem in dieser Arbeit entwickelten Struktogramm darüber hinaus noch weitere Funktionen hinzuzufügen sind. So soll der Roboter im Falle eines Zurückweichens, ausgelöst durch mangelnden zur Verfügung stehenden Abstand zum Hindernis oder aufgrund von einer sonst unvermeidbarer Kollision mit diesem, zunächst überprüfen, dass es auch in der Rückwärtsbewegung zu keinem Zusammenstoß mit einem in unmittelbarer Nähe zum Roboter befindlichen Hindernis kommt. Die Ergebnisse der

Abstandsmessungen des am Heck des Roboters befindlichen Ultraschallsensors sind hierfür heranzuziehen. Es wurde aus Gründen der Übersichtlichkeit davon abgesehen, diese Abfrage im Struktogramm aufzunehmen, was sie jedoch nicht weniger relevant macht. Nach dem Start des Programmes durch das kurzzeitige Betätigen des Tasters 1 am Lokomotion-Controller (Abbildung 1 (7b)) soll sich der Roboter kurzzeitig vorwärtsbewegen, sofern sich kein Hindernis in seiner unmittelbaren Umgebung befindet. Die genaue Definition dieser Umgebung bleibt stets durch den Anwender und den letztlich vorgesehenen Einsatzort des Roboters zu bestimmen. Aufgrund des benötigten Manövrierraums ist jedoch stets ein zu Hindernissen einzuhaltender Abstand von präterpropter 25 bis 30 cm ratsam. Nach jeder Iteration dieser Schleife soll der Roboter den von ihm einsehbaren Bereich inkrementell schwenken beziehungsweise neigen, um den gesamten vor ihm liegenden Bereich auf Hindernisse untersuchen zu können. Der Modellbauservomotor des Schwenk- und der des Neigemoduls bewegt sich dabei in einem Winkel zwischen einem zuvor festgelegten rechten und linken Anschlag beziehungsweise einem oberen und unteren Anschlag. Der maximal mit diesen Servos realisierbare abtastbare Bereich liegt in einem Winkel von 180 °. Die Definition der auch in Abbildung 17 dargestellten Winkel, in denen sich die 15-°-Kegel bewegen können, bleibt ebenso wie die vorher definierten kritischen Abstände dem Anwender überlassen. Nach dem erfolgreichen Identifizieren eines Hindernisses durch den Ultraschallsensor des Schwenkmoduls soll der Roboter, in Abhängigkeit der Lage des Hindernisses (links oder rechts der Körpermitte), eine Drehung in die entgegengesetzte Richtung ausführen. Erfasst jedoch der Sensor des Neigemoduls ein Hindernis, so muss dies anders umwunden werden. Wird ein Hindernis oberhalb des Roboters registriert und als ausweichbar wahrgenommen, so soll der Roboter die nötigen Anpassungen der Laufparameter vornehmen und unter diesem hindurch laufen. Solche Anpassungen können die Schritt- und Körperhöhe, aber auch das Laufmuster betreffen. Wird dahingegen ein Hindernis unterhalb des Roboters lokalisiert und als überwindbar eingeschätzt, muss auch hier der Roboter die Parameter seiner Bewegung anpassen. Im Detail sind hier ebenfalls die Schritt- und Körperhöhe, sowie Laufmuster und Schrittgeschwindigkeit zu nennen. Wie eine Änderung der Laufparameter im Quellcode zu integrieren ist, zeigt das Beispielprogramm „Moving_1“ des Downloadpakets. Die ursprünglichen Laufparameter können nach dem Überwinden des Hindernisses erneut aufgerufen werden, sofern der sich am Heck des Laufroboters befindliche Ultraschallsensor das auf dem Boden befindliches Hindernis ebenfalls registriert. Terminiert wird die Schleife durch ein kurzzeitiges betätigen des Tasters 2 (Abbildung 1 (7a)), welches gleichzeitig das Ende des Programmes darstellen soll.

Ad 4.

Abschließend soll in diesem Beispiel eine Modellumgebung für den Einsatz des Roboters konstruiert werden, in der der Roboter dank der zuvor implementierten Modifikationen dazu in der Lage ist, sich autonom und störungsfrei fortzubewegen. Um alle zuvor programmierten Funktionen des Laufroboters testen zu können, bedarf es der Entwicklung einer abwechslungsreich gestalteten Umgebung. Um dabei so realitätsnah wie möglich zu bleiben, wird die Planung und anschließende Umsetzung eines Hindernislaufes empfohlen. Es sollte dabei auf Kompaktheit des Modells geachtet werden, jedoch muss ebenso genügend Raum für Manöver des Roboters eingeplant werden. Hier ist es ratsam, sich vor der Konstruktion des Hindernislaufes mit den maximalen und minimalen Wendekreisen sowie den Dimensionen des Roboters vertraut zu machen. Neben Richtungswechseln soll die zurückzulegende Strecke ebenso Hindernisse am Boden sowie in der Höhe beinhalten, die den Roboter zu entsprechenden Reaktionen zwingen. Der Boden könnte so zum Beispiel über einen bestimmten Streckenabschnitt stufenweise angehoben und wieder abgesenkt werden. Bei der Gestaltung des Höhenhindernisses ist beispielsweise die Installation einer Schranke oder herabgesetzten Decke denkbar. Bei der genauen Ausgestaltung dieses Parcours wird den Studierenden ansonsten ein hoher Freiheitsgrad gelassen. Der Grundriss eines Beispiellaufs ist in Abbildung 18 dargestellt. In diesem soll der Roboter nach dem Überwinden einer Stufe in eine Kurve geführt werden und abschließend unter einer herabgesetzten Decke, die einen Tunnel simulieren soll, hindurchlaufen. Die Bahnbreite ist dabei variierbar, sollte aber zwecks der Konformität mit dem im vorangegangenen Arbeitsschritt implementierten Programm eine Mindestbreite von ungefähr 50 cm nicht unterschreiten, um ein Festlaufen des Roboters zwischen den Wänden zu verhindern.

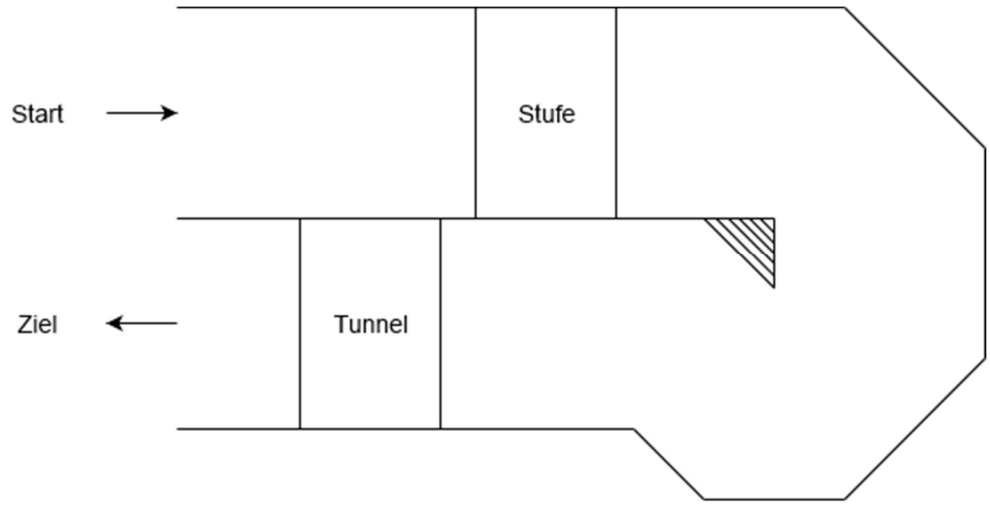

Abbildung 18: Grundriss eines beispielhaften Hindernislaufs

Einmal aufgebaut, kann der Roboter in die Teststrecke geführt werden und diese selbstständig durchlaufen. Die Ergebnisse mehrerer Testdurchläufe können anschließend dazu genutzt werden, den Quellcode des Roboters zu optimieren oder gegebenenfalls weiterzuentwickeln. Insbesondere die durch den Benutzer programmierten zulässigen Abstände zu Objekten sind dabei den Testbedingungen anzupassen.

4.3.3 Umsetzung der Aufgabenstellung

Anders als die zuvor vorgestellten beispielhaften Inklusionsweisen des Roboters in das Lehrkonzept des ITPLs soll dieses Projekt zudem konkret umgesetzt werden. Bei der prototypischen Konstruktion dieses autonomen mobilen Roboters wird die in Kapitel 4.3.1 aufgeführte Aufgabenstellung sukzessive bearbeitet. Dabei sollen insbesondere die dabei auftretenden Komplikationen dokumentiert und deren potentiellen Lösungsansätze herausgearbeitet werden. Die hierbei gewonnenen Erkenntnisse können anschließend in der auszuarbeitenden Reflektion des Projektes nach Abschluss der Konstruktion des Geländeroboters aufgegriffen werden.

Ad 1.

Es wird damit begonnen, den nichtmodifizierten Roboter auf die Installation der zusätzlichen Sensorik und Aktorik vorzubereiten. Dabei wird der Arduino UNO, wie bereits in Abbildung 14 (A) gezeigt, auf die für ihn vorgesehene Steckerleiste des Lokomotion-Controller-Boards gesteckt. Während der gesamten Vorbereitungsarbeiten sollte die Energieversorgung der Boards aus Sicherheitsgründen unterbrochen werden. Zum Schutz der auf den Boards verbauten Mikroelektronik vor elektrostatischen Entladungen und einer dadurch drohenden Zerstörung empfiehlt es sich ohnehin, den Körper vor der Arbeit mit diesen empfindlichen Boards zu entladen (Brühlmann 2019). Nachdem der Arduino angeschlossen wurde, kann damit begonnen werden, die serielle Kommunikation zwischen den beiden Boards aufzubauen. Auf dem Computer des Benutzers wird die Arduino IDE gestartet und eine neue Datei erstellt. In diesen Sketch werden zuerst alle benötigten Bibliotheken eingebunden. Im Detail sind das die Bibliothek „Servo“, die die Programmierung der später im Code zu implementierenden Servomotoren um ein Vielfaches erleichtert, und die im Downloadpaket des Herstellers enthaltene Bibliothek „Hexapod_Lib“. In dieser sind die Bewegungsfunktionen des Roboters sowie eine Methode zur Verbindung der seriellen Schnittstellen beider Boards bereits vordefiniert und müssen im Programm des Arduino UNO lediglich aufgerufen werden. Da in den Funktionen der „Hexapod_Lib“-Bibliothek auch

```
//Ruft den Befehl aus der Hexapod-Bibliothek mit der
//Festlegung der Übertragungsrate auf
SERIAL_CMD.begin(SERIAL_CMD_BAUD);

Serial.begin(SERIAL_STD_BAUD);
```

Abbildung 19: Initialisieren der seriellen Schnittstellen

Fehlermeldungen zurückgegeben werden können, wird neben der seriellen Schnittstelle über die UART-Pins auch eine weitere über den USB-Anschluss realisiert, durch die die Fehlermeldungen dann am seriellen Monitor der IDE einsehbar werden. Dadurch können die Fehler im Quellcode bei der Programmierung besser identifiziert werden. Die beiden seriellen Schnittstellen sind über die in Abbildung 19 dargestellten Codezeilen im Sketch zu initialisieren. Das erstellte Programm wird nach Abschluss der restlichen Arbeitsschritte über eine USB-Verbindung auf den Arduino UNO geladen. Der vollendete Sketch befindet sich ebenfalls in einer ausführlich kommentierten Form im elektronischen Anhang (E).

Ad 2.

Im nächsten Schritt werden die Sensoren und zusätzlichen Aktoren angebracht. Dazu bedarf es jedoch zunächst der Anfertigung passender Halterungen. Hierzu müssen die exakten Maße der einzubauenden Sensoren, Motoren und der möglichen Anschlusspunkte am Roboter sorgfältig berücksichtigt werden. In dieser Arbeit wird bei der Gestaltung der Halterungen mit dem CAD-Programm Autodesk Fusion 360 gearbeitet. Die hiermit entworfenen 3D-Modelle sind dem elektronischen Anhang (A) beigefügt. Anschließend werden diese mit Hilfe des 3D-Druckers des ITPLs ausgedruckt. Die Betriebssoftware des Druckers fügt den zu druckenden Modellen automatisch Stützstrukturen hinzu, die für das Drucken von Überhängen eines Körpers vonnöten sind. Das saubere Entfernen dieser Strukturen, welche durch den wasserlöslichen Kunststoff Polyvinylalkohol aufgebaut werden, erweist sich aber oftmals als Herausforderung, da sich der Kunststoff nur unter großen zeitlichen Aufwand sauber entgraten lässt (Ultimaker 2019). Daher sollte bereits beim Konstruieren dieser Modelle darauf geachtet werden, dass es bei der späteren Umsetzung keiner dieser Stützstrukturen bedarf. Die in dieser Arbeit durchgeführten Versuche die Modelle ohne Unterstützung zu drucken ergaben, dass bereits kleine Radien an den kritischen Kanten diese

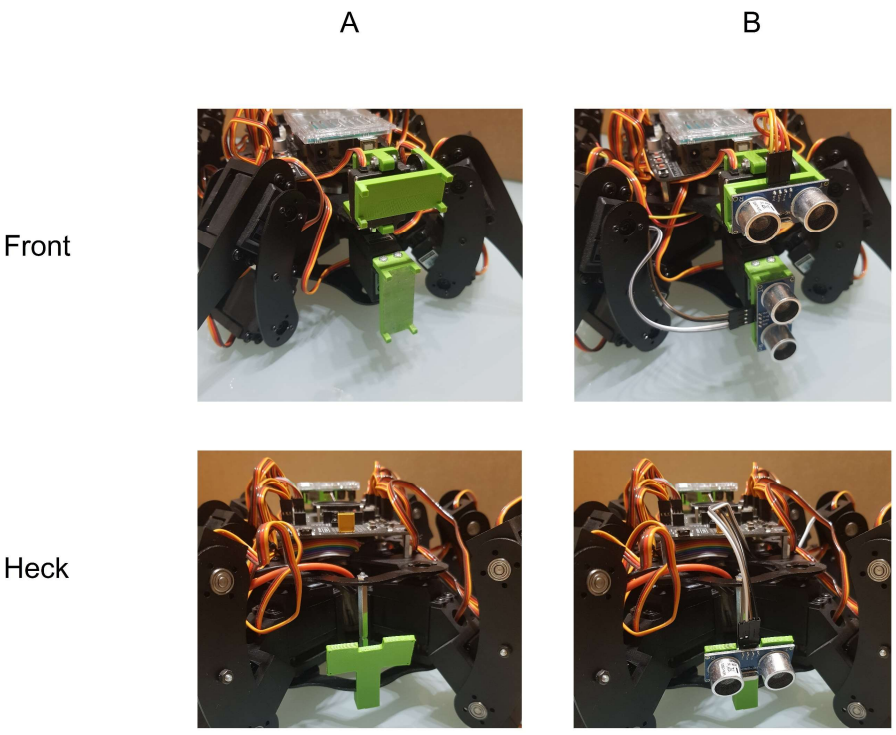

Abbildung 20: Montage der Halterungen und Ultraschallsensoren

Problematik vermeiden können. Die vier gedruckten Teile konnten nach einer sorgfältigen Entgratung an dem Grundgerüst des Roboters montiert werden. Abbildung 20 (A) zeigt die Module an der Front (Schwenk- und Neigeeinheit) und dem Heck (starre Halterung des Ultraschallsensors) des Roboters nach der Montage. In Abbildung 20 (B) sind zusätzlich die dann im nächsten Schritt an den Modulen zu befestigenden Ultraschallsensoren zu erkennen. Wie die Sensoren und Motoren genau mit dem Arduino verbunden werden, kann im Schaltplan, der im elektronischen Anhang (C) zu finden ist, nachvollzogen werden. Vorteilhaft am Einsatz eines Arduino UNO in diesem Projekt ist, dass es zum Verbinden der Ultraschallsensoren, anders als bei der Verbindung zu einem Raspberry Pi (Spannungslevel von 3,3 V im Prozessorkern), aufgrund der übereinstimmenden Betriebsspannung von 5 V keiner Pegelwandler bedarf (Arduino 2020b) (Joy-IT 2017).

Ad 3.

Nach erfolgreicher Integration in die Hardware, können die Funktionen der zusätzlichen Sensoren und Aktoren im Programm des Einplatinencomputers implementiert werden. Dazu soll das zuvor definierte Struktogramm als Vorlage dienen. Hierbei ist davon auszugehen, dass der Programmcode

qua Versuch und Irrtum sukzessive entwickelt werden muss. So lässt sich beispielsweise erst in den späteren Versuchen erkennen, welchen Bewegungsrestriktionen die Sensormodule unterliegen. Aufgrund der Regungen der beidseitigen Vorderbeine, die bei einer vollen 180-°-Drehung des Schwenkmoduls unweigerlich mit diesem im Lauf kollidieren würden, muss der maximal vom Sensor einsehbare Bereich seitlich begrenzt werden. Auch die maximale Ausschwenkung des Neigemoduls muss dabei eingeschränkt werden. Versuche haben ergeben, dass andernfalls der Sensor des Neigemoduls mit dem Sensor des Schwenkmoduls zusammenstieße. Anstelle einer Begrenzung der maximalen Auslenkung können auch alternative Sensorhalterungen entwickelt werden, die dieses Problem beheben. Da die Winkel, in denen nun Gegenstände erfasst werden können (circa 75 ° bei beiden Modulen), jedoch in den Tests ausreichend weit erschienen, wurde in dieser Arbeit davon abgesehen. Bei der Programmierung der Laufrichtungssteuerung des Roboters wird nun schrittweise vorgegangen. Der dabei entwickelte Quellcode befindet sich in ausführlich kommentierter Form im elektronischen Anhang (E). Dessen Beginn stellt, analog zum Struktogramm, eine Abfrage über den Zustand der Taster dar. Durch das kurzzeitige Betätigen des Tasters 1 soll der Roboter damit beginnen, autonom loszulaufen. Der Laufvorgang soll dann durch das Drücken des Tasters 2 beendet werden. Realisiert werden diese beiden Abfragen zum einen durch eine *if*-Anweisung und zum anderen durch eine *while*-Schleife. Die bedingte Anweisung wird ausgeführt, falls der Taster 1 für eine kurze Zeit gedrückt wurde. Die Anweisung beinhaltet dann die Schleife, die solange durchlaufen wird, bis der Taster 2 betätigt wurde. Innerhalb der Schleife sind dann die restlichen Funktionen enthalten, die den Roboter durch die hindernisreiche Umgebung führen können. Der Aufbau des Start- und Stoppvorgangs im Quellcode ist in Abbildung 21 dargestellt. Nach Beendigung der Schleife durch den Tastendruck fährt der Roboter seinen Rumpf auf Bodenhöhe und schaltet sich ab. Die dafür benötigten Funktionen sind außerhalb der Loop-Funktion zu definieren und können kommentiert in der Datei im elektronischen Anhang (E) aufgefunden werden. Dort ist ebenfalls die Setup-Funktion des Programmes nachzulesen, in der benötigte Variablen deklariert, initialisiert und die Pinmodi festgelegt werden. Auf diese Funktion wird hier aber nicht weiter eingegangen, sie ist aber ebenfalls in der Arduino-Datei im elektronischen Anhang (E) näher beschrieben. Der nächste Schritt im Struktogramm behandelt die Identifizierung von Hindernissen. Sofern keine Hindernisse erkannt werden, bewegt sich der Roboter strikt vorwärts. Wird jedoch ein solches in der Laufrichtung des Roboters von den Sensoren erkannt, soll eine Fallunterscheidung durchgeführt werden. Falls das Schwenkmodul beim Messen des vor sich befindlichen Bereichs ein Hindernis erkennt, wird durch

```
void loop()
{
//Normalerweise sind die Taster-Pins immer HIGH (true),
//für die Zeit, die sie betätigt werden wechseln sie auf
//LOW (false)
      if(!digitalRead(T1))
    {
        delay(50);
        if(!digitalRead(T1))
        {
//…
            while(digitalRead(T2))
            {
                   move_fwd(100);
//…
             }
            stop_move(100);
            shut_down();
        }
    }
 }
```

Abbildung 21: Start und Stopp durch Tasterabfragen

die aktuelle Stellung des Servomotors ermittelt, auf welcher Seite sich das Hindernis befindet und eine entsprechende Korrektur des Laufweges in die entgegengesetzte Richtung eingeleitet. Befindet sich das Objekt für ein solches Manöver bereits zu nah am Roboter, soll dieser vor der Rotation um seine eigene Achse eine Rückwärtsbewegung ausführen, vorausgesetzt der Hecksensor kann kein Hindernis erkennen. Kann dem Hindernis auf diesem Wege nicht ausgewichen werden, soll der Roboter abschalten. Für den Fall, dass der Ultraschallsensor des Neigemoduls ein Objekt unterhalb der zuvor definierten zulässigen Distanz erkennt, wird auch hier aus der Position des Motors die Art des Hindernisses bestimmt. Befindet sich der Motor bei der Detektion der potentiellen Beeinträchtigung oberhalb eines bestimmten Winkels, soll eine Kriechfunktion aufgerufen werden, die den Roboter und seine Bewegungen in seine tiefst mögliche Lage bringt. Sollte die Motorstellung des Neigemoduls die weitere definierte Stellung unterschreiten, lässt dies auf ein am Boden befindliches Hindernis schließen. Der Roboter soll dann eine Kletterfunktion ausführen, welche beendet wird, sobald der Hecksensor das Hindernis ebenfalls registriert hat. In dieser Kletterfunktion soll das Laufmuster angepasst und der Roboter und seine Bewegungen in seine höhst mögliche Lage gebracht werden. Damit der Sensor überwundene Hindernisse zu diesem Zweck zuverlässig erkennen

```
void climb(){
while(SR04_DISTANCE_REAR>CLIMB_END){

//Funktionen aus der Hexapod-Bibliothek zur Anpassung der
//Laufparameter: Laufgeschwindikeit, Körperhöhe,
//Laufmuster, Schrittweite, Schritthöhe)

  ROBOT_SPEED(MOVE_SPEED_CRCL);
  ROBOT_HEIGHT(CLIMBING_HEIGHT);
  ROBOT_GAINT_MODE(WAVE_24);
  ROBOT_DOUBLE_LENGTH_ON();
  ROBOT_DOUBLE_HEIGHT_ON();
  ROBOT_WALK_FWD();                }
//Herstellung der ursprünglichen Parameter bei
//Schleifenende
  ROBOT_SPEED(MOVE_SPEED_FWD);
  ROBOT_HEIGHT(35);
  ROBOT_GAINT_MODE(TRIPOD_6);
  ROBOT_DOUBLE_LENGTH_OFF();
  ROBOT_DOUBLE_HEIGHT_OFF();
           }
```

Abbildung 22: Kletterfunktion

kann, so haben die Versuche gezeigt, muss die Halterung des Sensors etwas nach unten neigen. Diese Korrektur der Halterung wurde nachträglich durch ein schrittweises Biegen des Kunststoffteils erreicht, um den Arbeitsbereich des Hecksensors weiter nach unten zu verschieben. Hierbei ist bei der Integration des Sensors in den Quellcode besonders darauf zu achten, dass der Untergrund zwar registriert, jedoch nicht als Hindernis identifiziert wird. Ein genaues Kalibrieren der kritischen Abstände zum Roboter verhindert dies. Die beiden speziellen Bewegungsfunktionen (Kriechen und Klettern) werden gesondert außerhalb der Loop-Funktion mit Verweisen auf Funktionen der „Hexapod_Lib“-Bibliothek programmiert. In diesen werden speziell Schritt- und Körperhöhe, Laufmuster und -geschwindigkeit angepasst. Der Code der Kletterfunktion ist in Abbildung 22 dargestellt. In diesem wird, aufgrund seiner hohen Stabilität, der Wave-Laufmodus ausgewählt, um dem Roboter auch beim Überschreiten des Hindernisses einen festen Stand zu garantieren. Befindet sich der Roboter derweilen nicht in der Kriech- oder Kletterfunktion, so bewegt sich der Laufroboter standardmäßig mit dem wesentlich schnelleren Tripod-Muster fort. Die beiden Funktionen können nach der Definierung innerhalb der Loop-Funktion verwendet werden. Zu den weiteren dort definierten Funktionen gehören auch die für die Links- und Rechtsdrehung, Vorwärts- und

Rückwärtsbewegung, den Start und Stopp des Roboters und die Abstandsmessung der drei Ultraschallsensoren. Auf die Programmierung dieser wird im Rahmen der hier vorliegenden Dokumentation aufgrund ihrer Trivialität jedoch nicht weiter eingegangen. Üblicherweise wird zu einer Programmierung des Mikrocontrollers vor dessen Einbau geraten. Da der Programmiervorgang hier aber abwechselnd mit wiederholten Tests in der Modellumgebung in Verbindung steht und es dadurch einem mehrfachen Ein- und Ausbauen des Benutzer-Boards bedürfte, soll zum Schutz der sensiblen Elektronik davon abgesehen werden und das angepasste Programm auf den Arduino stets im eingebauten Zustand hochgeladen werden.

Ad 4.

Nach der ersten Programmierung des Roboters sollen die implementierten Funktionen durch praktische Versuche überprüft werden. Dazu bedarf es der Umsetzung der bereits in Kapitel 4.3.2 (Ad 4.) entwickelten Testplattform. In dieser Arbeit wurde entschieden, hierfür Kartonage zu verwenden. Die Verwendung von solchen Packkisten zeichnet sich primär durch drei Vorteile aus: So erlaubt der so entstandene modulare Aufbau eine unkomplizierte Umgestaltung der Modellumgebung. Die Kartons müssen dazu lediglich verschoben werden. Eine weitere günstige Eigenschaft ist, dass die in dieser Arbeit verwendeten Ultraschallsensoren die Pappoberflächen gut erkennen können. Bei der Erkennung weicherer oder gekrümmterer Materialien kann es zu einer erhöhten Streuung des ausgesendeten Ultraschallsignals kommen, was ein genaues Messen der Distanzen erschwert (Gu u. a. 2002). Der Einsatz alternativer Abstandssensoren, speziell sind hier die Infrarotsensoren zu nennen, kann den Roboter dazu befähigen, auch in solchen Umgebungen agil zu bleiben. Der dritte entscheidende Vorteil der Kartonage ist, dass eine Fehlprogrammierung und die dadurch entstehenden Unfälle keine oder nur geringfügige Schäden an dem Parcours und Roboter hinterlassen können, was der Nachgiebigkeit des Materials zu verdanken ist. Der Aufbau des Hindernisparcours ist in Abbildung 23 dargestellt. Durch die bei der Durchführung der Versuche gemachten Beobachtungen können die nötigen Feineinstellungen der Parameter vorgenommen werden. Besonders wichtige Größen sind dabei die kritischen Abstände (im Programm mit „COLLISION_THRESH_*“ deklariert), die auf die Dimensionen der vorhandenen Umgebung abgestimmt werden müssen. Fahrlässig definierte Werte führen, so haben es Tests bestätigt, zu Unfällen mit Hindernissen oder Nichtausführung der Kriech- und Kletterfunktion sowie des Rückwärtslaufens trotz gegebenen Anlasses.

Abbildung 23: Hindernisparcours aus Kartonage

4.3.4 Reflektion des Beispiels

Nach Abschluss der prototypischen Umsetzung dieses Implementierungsbeispiels kann es nun retroperspektiv hinsichtlich der Komplexität, des Aufwands und der Erweiterbarkeit beurteilt werden. Anders als bei der Reflektion der vorangegangen Projekte, kann sich bei dieser Bewertung auch auf gemachte Erfahrungen in der Umsetzung berufen werden. So konnte hierbei bereits festgestellt werden, dass der Roboter schon in seiner Programmierung an sein späteres Einsatzfeld speziell angepasst werden muss, was letztendlich seine Flexibilität massiv beeinträchtigt. Um dem entgegen zu wirken, wäre eine Erweiterung des Laufroboters um zusätzliche Sensorik denkbar. Dazu eignen sich neben den bereits erwähnten Infrarotsensoren auch kraftempfindliche Widerstände (FSR). Die Montage solcher Widerstände an den Fußspitzen des Roboters ermöglicht es diesem, Informationen über die Lagebeziehungen zwischen dem Untergrund und den Roboterbeinen einzuholen, da der elektrische Widerstand an den Sensoren bei Auftritt des jeweiligen Beins abfällt. Die zusätzlichen Sensoren können mit den noch freien Anschlüssen SA0 bis SA5 (Abbildung 1 (8)) verbunden und dann im Betriebsprogramm des Lokomotion-Controller-Boards implementiert werden. Dort ließe sich dann auch ein eigenes Laufmuster programmieren, welches die durch FSR gewonnenen Informationen in der Schrittfolge berücksichtigt. Die Integration verschiedener Sensortypen hätte neben dem positiven Einfluss auf die Flexibilität des dabei entstehenden Geländeroboters auch eine steigende Komplexität und einen wachsenden Umfang des Projektes zur

Folge. Eine weitere Möglichkeit, den Roboter unter diesen Aspekten zu erweitern, ist die Einführung eines Kartierungssystems, wie es auch in modernen Saugrobotern zu finden ist (Hasan und Reza 2014). In diesem soll der Roboter mit Hilfe von rotierenden Abstandssensoren sein gesamtes Umfeld (360 °) vermessen und die Ergebnisse abspeichern, um so sukzessive eine Karte von seiner Umgebung aufzubauen und seine Position in ihr jederzeit bestimmen zu können. Ein solches *Simultaneous-Localization-and-Mapping*-System (SLAM) soll die Navigation des Roboters unterstützen, sodass beispielsweise Sackgassen maximal einmal angesteuert werden. Neben diesen programmierintensiven Erweiterungen bestehen zusätzlich simplere Optionen den Funktionsumfang des Roboters zu potenzieren. Um auch Abschnitte des Hindernisparcours mit engeren Passagen überwinden zu können, wäre die Integration zweier seitlichen Ultraschallsensoren ratsam. Mit Hilfe derer könnte der Roboter dann erkennen, wo der Mittelweg durch diesen Streckenabschnitt verläuft, diesen dann entlanglaufen und falls nötig die Stellung der Beine in Richtung Körpermitte anpassen, um so schmal wie möglich zu werden. Diese Veränderung des Laufmusters müsste jedoch in den zugrunde liegenden mathematischen Modellen der Motion-Betriebssoftware angepasst werden. Ohne die Erweiterung dieses Beispiels wird aber auch in diesem Projekt die Software des Lokomotion-Controllers nicht verändert. Dies hat analog zum Überwachungsroboter aus Kapitel 4.1 zu bedeuten, dass auch hier im Fachlabor Teams von Studierenden, mit jeweils einem zu programmierenden Arduino UNO ausgestattet, die Problemstellung parallel, mit abwechselnden Zugriff auf den Roboter, bearbeiten könnten. Aufgrund der durchzuführenden, zeitintensiven Iterationen von Versuch und Irrtum ist dies jedoch nur bedingt zu empfehlen. Vorausgesetzt die Teams arbeiten nicht mit denselben Erweiterungen (Schwenk- und Neigemodul), sodass der Roboter vor Beginn der Versuche aufwändig umgebaut werden muss, kann die Erprobung und Überarbeitung des Codes einen erheblichen zeitlichen Aufwand in Anspruch nehmen. Der in dieser Arbeit zu diesem Zweck erbrachte Aufwand beläuft sich auf schätzungsweise 14 Stunden. Aus den in der vorliegenden Arbeit vorgestellten Beispielen behandelt dieses den praktischen Umgang mit Sensorik und Aktorik und die Implementierung derer in Robotersysteme am intensivsten. Ferner können die Studierenden im Rahmen der Durchführung dieses Projektes ebenso Kompetenzen in Konstruktion und Druck von 3D-Modellen erwerben, die für eine erfolgreiche Umsetzung der Geländeroboters unabdingbar sind. Im Vergleich zu den übrigen Projekten wird hier jedoch die Programmierung von Python-Programmen oder einer PHP-Webpage nicht behandelt. Werden bei der Entwicklung des Quellcodes die im vorhergien Kapitel erwähnten Softwarebibliotheken verwendet, sollte mit diesen auch sicher umgegangen werden können. Ein Verwenden der Softwarebibliothek, insbesondere der

„Hexapod_Lib“-Bibliothek, beschränkt den bei der Programmierung des Benutzerboards zu betreibenden Aufwand immens. So könnte im Rahmen der Problemstellung den Studierenden eine Nutzung dieser untersagt werden, um die Komplexität und den Aufwand dieses Projektes einem Fachlabor entsprechend anzupassen. Die Funktionen der „Hexapod_Lib“-Bibliothek beinhalten bereits das Übertragungsprotokoll der Kommunikation von Benutzerboard zu Lokomotion-Controller-Board. Kann der Benutzer dieses einmal nachvollziehen, so ist es möglich das Kommunikationsprotokoll auch direkt in den eigenen Funktionen des Arduino-Programmes zu integrieren. Dies könnte zur Folge haben, dass die Studierenden auch eigene, verbesserte Lösungsansätze finden, die Kommunikation der beiden Boards zu gestalten. Eine solche Restriktion erscheint somit auch als wirkungsvolles Mittel, den Studierenden eine intensivere Beschäftigung mit der Lokomotion-Betriebssoftware nahezulegen. Die Aufführung ausgewählter Erweiterungsbeispiele zeigt, dass sich die andernfalls beschränkte Komplexität ebenso wie der ansonsten limitierte Umfang dieses Projektes beliebig anpassen lassen. Ohne eine nachträgliche Ausdehnung des in der Problemstellung gewünschten Funktionsumfangs des Geländeroboters oder der Definition von Restriktionen in dessen Umsetzung scheint dieses Implementierungsbeispiel, aufgrund des mangelnden Aufwands und der geringen Komplexität, für den Einsatz im Fachlabor des ITPLs ungeeignet.

5. Zusammenfassung und Ausblick

Sensorik und Aktorik sind von großer Bedeutung in der heutigen Produktion und Logistik. Diese relevanten Themengebiete den Studierenden näherzubringen, soll unter anderem eine Aufgabe des vom Fachgebiet IT in Produktion und Logistik angebotenen Fachlabors sein. Im Rahmen dessen soll der Umgang mit durch Mikrocontroller gesteuerten und geregelten Sensor-Aktor-Systemen praktisch erprobt werden. Als Plattform hierfür wurde ein ebenfalls auf Mikrocontrollern basierender Laufroboter zusammen mit zahlreichen Peripheriegeräten angeschafft. Diese Anschaffungen wurden in der hier vorliegenden Ausarbeitung in Hinblick auf ihre Verwendungsmöglichkeiten im alltäglichen Lehrbetrieb des Fachgebiets untersucht. Hierfür mussten zunächst deren technischen Eigenschaften sowie deren Programmierbarkeit identifiziert werden. Dabei wurde speziell die Software des Laufroboters und dessen Erweiterbarkeit durch Benutzer-Boards untersucht. Diese Untersuchungen sollten die Grundlage für das Entwickeln drei potentieller Anwendungen im Fachlabor bilden. Darüber hinaus wurden Studien über solche Anwendungen mit vergleichbaren Robotern in der Lehre herangezogen und der Erfolg einer praktisch orientierten Lehrveranstaltung bezogen auf Kompetenzvermittlung eingeschätzt. Vor dem Hintergrund, die hier entwickelten Beispielprojekte im Lehrkonzept zu integrieren, wurden Handlungsanweisungen für die schrittweise Umsetzung der drei Beispiele angefertigt. Um die Legitimität dieser verifizieren zu können, wurde eines der Beispiele, die Konstruktion eines autonomen Geländeroboters, in der Praxis realisiert. Während dieser Realisation wurden die vorgenommenen Arbeitsschritte dokumentiert. Die drei Projekte wurden anschließend evaluiert, um Schwachstellen aufzuzeigen und sie nach den Kriterien Erweiterbarkeit, Komplexität und Umfang zu bewerten.

Zusammenfassend lässt sich auf Grundlage dieser Bewertungen festhalten, dass der Einsatz des Roboters im Lehrbetrieb des Fachbereichs empfohlen wird, auch wenn die betrachteten Studien die pädagogischen Vorteile von praktisch orientierten Lehrkonzepten gegenüber theorielastigen Veranstaltungen nicht eindeutig beweisen können. Dies wird damit begründet, dass die in der Bewertung der Implementierungsmöglichkeiten herausgearbeiteten zu erlernenden Kompetenzen der Studierenden weitestgehend konform mit denen im Curriculum des aktuell angebotenen Fachlabors des ITPLs sind. In den Bewertungen der Beispiele konnte auch ein hohes Maß an Flexibilität bezüglich Aufwand und Umfang dieser festgestellt werden. Diese Anpassungsfähigkeit ergibt sich aus der erheblichen Erweiterbarkeit der Projekte. Die Erweiterbarkeit des Roboters spielt dabei eine entscheidende Rolle. Dessen Funktionsumfang lässt sich mit Hilfe der spezifischen

Mikroelektronik und dem vom Fachgebiet ebenfalls bereitgestellten 3D-Drucker unkompliziert erweitern. Diese Mikroelektronik bietet gepaart mit dem Roboter eine kostengünstige Möglichkeit, industrienahe praktische Erfahrung im Umgang mit Steuerungs- und Regelungstechnik zu erwerben. Bei einer geschickten Auslegung der Aufgabenstellung im Labor können sogar mehrere Teams von Studierenden parallel an diesem Roboter ihre Projekte bearbeiten, sofern für diese kein ständiger Zugang zum Roboter erforderlich ist.

Auf dieser Arbeit aufbauend, können zukünftig weitere Problemstellungen, die eine Modifikation des Roboters in Soft- und Hardware vorsehen, entwickelt werden. Denkbare Modifikationen wären dabei neben alternativen, reinen Funktionserweiterungen, wie sie hier vorgestellt wurden, auch die komplette Neugestaltung einer eigenen Lokomotion-Software. In dieser könnten dann eigene mathematische Modelle für die Laufbewegungssteuerung implementiert werden, was jedoch erste Erfahrungen in der Robotik und speziell in der Programmierung kinematischer Modelle erfordert. Studierenden ohne praktische Erfahrung im Gebrauch von Mikrocontrollern und deren Peripherie sollte der Roboter mit seiner vorprogrammierten Betriebssoftware ebenfalls eine ideale Lernplattform darstellen, die Funktionsweisen von Sensorik und Aktorik zu ergründen.

Literaturverzeichnis

Abel, Dirk; Bollig, Alexander (2006): *Rapid control prototyping*. 1. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg.

Arduino (2020a): „Arduino - Libraries“. *Arduino Libraries*. Abgerufen am 02.01.2020 von https://www.arduino.cc/en/reference/libraries.

Arduino (2020b): „Arduino Uno Rev3 | Arduino Official Store“. *Arduino Uno Rev3 Store*. Abgerufen am 10.01.2020 von https://store.arduino.cc/arduino-uno-rev3.

Atmel (2014): *„ATmega2560 Datasheet*". Atmel ATmega2560. Abgerufen am 21.01.2020 von http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf.

Balzert, Helmut (2010): *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. 3. Heidelberg: Springer Spektrum.

Banzi, Massimo; Shiloh, Michael (2014): *Getting Started with Arduino: The Open Source Electronics Prototyping Platform*. 3. Santa Rosa, CA: Make Community, LLC.

Brühlmann, Thomas (2019): *Arduino: Praxiseinstieg*. 4. Frechen: MITP-Verlags GmbH & Co. KG.

Bühler, Peter; Schlaich, Patrick; Sinner, Dominik (2018): *Webtechnologien*. 1. Wiesbaden: Springer Vieweg.

Cicolani, Jeff (2018): *Beginning Robotics with Raspberry Pi and Arduino: Using Python and OpenCV*. 1. Berkeley, CA: Apress Media LLC.

Conrad Electronic SE (2020): „Download Paket Laufroboter“. *Conrad Electronic SE Download Center*. Abgerufen am 02.01.2020 von https://produktinfo.conrad.com/index.php?type=up&lang=&bl=EN&q=Hexapod+.

D’Ausilio, Alessandro (2012): „Arduino: A low-cost multipurpose lab equipment“. In: *Behavior Research Methods*. 44 (2), S. 305–313. Chicago, IL: Psychonomic Society.

DiBona, Chris; Ockman, Sam (1999): *Open sources: Voices from the Open Source Revolution*. 1. Sebastopol, CA: O’Reilly Media, Inc.

Espressif (2019): „ESP8266EX Resources | Espressif Systems“. *ESP8266EX Resources*. Abgerufen am 11.01.2020 von https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf.

Feldmann, Carsten; Gorj, Anneliese (2017): *3D-Druck und Lean Production*. 1. Wiesbaden: Springer Gabler.

Ferrell, Cynthia (1995): „A comparison of three insect-inspired locomotion controllers“. In: *Robotics and Autonomous Systems*. 16 (2–4), S. 135–159. Amsterdam: Elsevier BV.

Follmann, Rüdiger (2018): *Das Raspberry Pi Kompendium*. 2. Wiesbaden: Springer Vieweg.

Goldenberg, Andrew; Benhabib, Beno; Fenton, Robert (1985): „A complete generalized solution to the inverse kinematics of robots“. In: *IEEE Journal on Robotics and Automation*. 1 (1), S. 14–20. New York, NY: IEEE.

Google (2019): „Documentation Google Maps JavaScript API“. *Google Developers*. Abgerufen am 11.02.2020 von https://developers.google.com/maps/documentation/javascript/tutorial?hl=de.

Gu, Jason; Meng, Max; Cook, Al; Liu, Peter .X. (2002): „Sensor fusion in mobile robot: some perspectives“. In: Proceedings of the 4th World Congress on Intelligent Control and Automation. S. 1194–1199.

Hasan, Kazi Mahmud; Reza, Khondker Jahid (2014): „Path planning algorithm development for autonomous vacuum cleaner robots“. In: 2014 International Conference on Informatics, Electronics & Vision. S. 1–6.

Institut für Mechatronische Systeme (2020a): „Hackathon „Mobile Robotik“ – Institut für Mechatronische Systeme“. *Leibniz Universität Hannover*. Abgerufen am 23.01.2020 von https://www.imes.uni-hannover.de/de/studium/sonstiges-kursangebot/hackathon-mobile-robotik/.

Institut für Mechatronische Systeme (2020b): „Roboterfabrik - Lehrveranstaltungen“. *Leibniz Universität Hannover*. Abgerufen am 23.01.2020 von https://www.roboterfabrik.uni-hannover.de/285.html.

Intel (2019a): „Intel RealSense D400 Series Datasheet“. *Intel® RealSense™ Datasheet*. Abgerufen am 27.01.2020 von https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf.

Intel (2019b): „Using depth camera with Raspberry Pi 3“. *Intel® RealSense™ Developer Documentation*. Abgerufen am 06.02.2020 von https://dev.intelrealsense.com/docs/using-depth-camera-with-raspberry-pi-3.

Intel (2019c): „Intel® RealSenseTM SDK Installation Raspbian“. *Intel® RealSense™ GitHub*. Abgerufen am 27.02.2020 von https://github.com/IntelRealSense/librealsense.

Intel (2019d): „Intel® RealSenseTM SDK Installation Ubuntu“. *Intel® RealSense™ GitHub*. Abgerufen am 27.02.2020 von https://github.com/IntelRealSense/librealsense.

ITPL (2020): „Fachlabor ITF - IT in Produktion und Logistik - Fakultät Maschinenbau - TU Dortmund“. *ITPL Fachlabor*. Abgerufen am 01.02.2020 von http://www.itpl.mb.tu-dortmund.de/cms/de/studium/Lehrangebot/Aktuelle-Masterveranstaltungen/Fachlabor-ITF/index.html.

Jara, Carlos A; Candelas, Francisco A; Puente, Santiago T; u. a. (2011): „Hands-on experiences of undergraduate students in Automatics and Robotics using a virtual and remote laboratory“. In: *Computers & Education*. 57 (4), S. 2451–2461. Oxford: Elsevier Science Ltd..

Joy-IT (2017): „Joy-IT Ultraschallsensor“. *Joy-IT Ultraschallsensor*. Abgerufen am 26.01.2020 von https://asset.conrad.com/media10/add/160267/c1/-/en/001274216DS01/datenblatt-1274216-linker-kit-ultraschall-sensor.pdf.

Klassner, Frank (2002): „A case study of LEGO Mindstorms'™ suitability for artificial intelligence and robotics courses at the college level“. In: the 33rd SIGCSE technical symposium. S. 8–12.

Kofler, Michael; Kühnast, Charly; Scherbeck, Christoph (2016): *Raspberry Pi - Das umfassende Handbuch*. 2. Bonn: Rheinwerk.

Kurebayashi, Shuji; Kamada, Toshiyuki; Kanemune, Susumu (2006): „Learning Computer Programming with Autonomous Robots“. In: *Informatics Education – The Bridge between Using and Understanding Computers*. S. 138–149. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg.

Major, Louis; Kyriacou, Theocharis; Brereton, O Pearl (2012): „Systematic literature review: teaching novices programming using robots“. In: *IET Software*. 6 (6), S. 502–513. London: Institution of Engineering and Technology.

Makerfactory (2020a): „Demo programs - Makerfactory Documentation“. *Makerfactory Hexapod Documentation*. Abgerufen am 03.01.2020 von https://docs.makerfactory.io/educational-robotics/robobug-hexapod/demo_programs/.

Makerfactory (2020b): „Quick start - Makerfactory Documentation“. *Makerfactory Hexapod Documentation*. Abgerufen am 12.01.2020 von https://docs.makerfactory.io/educational-robotics/robobug-hexapod/quick-start/.

Massachusetts Institute of Technology (2020): „MIT App Inventor - About Us“. *MIT App Inventor.* Abgerufen am 11.02.2020 von http://appinventor.mit.edu/about-us.

Mr Dave (2020): „Motion - Documentation“. *Motion Documentation*. Abgerufen am 10.02.2020 von https://motion-project.github.io/motion_guide.html.

Orton, Kevin R. (1990): „RC Servo U.S. Patent No. 4,914,368“. Washington, DC.

Perens, Bruce (1999): „The open source definition“. In: *Open sources: Voices from the Open Source Revolution*. 1. S. 171–188. Sebastopol, CA: O'Reilly Media, Inc..

PixyCam (2020): „PixyCam Homepage“. *PixyCam*. Abgerufen am 27.02.2020 von https://pixycam.com/pixy-cmucam5/.

Siciliano, Bruno; Khatib, Oussama (2016): *Handbook of Robotics*. 2. Basel: Springer International Publishing.

Siegwart, Roland (2001): „Grasping the interdisciplinarity of mechatronics“. In: *IEEE Robotics & Automation Magazine*. 8 (2), S. 27–34. New York, NY: IEEE.

Sparkfun (2020): „HC-SR04 Datasheet“. *Sparkfun Datasheet*. Abgerufen am 27.02.2020 von https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf.

Spong, Mark W; Hutchinson, Seth; Vidyasagar, Mathukumalli (2006): *Robot Modeling and Control*. 1. Hoboken, NJ: John Wiley & Sons.

Szeliski, Richard (2010): *Computer Vision: Algorithms and Applications*. 1. Berlin: Springer Science & Business Media.

The Raspberry Pi Foundation (2020a): „Camera Module - Raspberry Pi Documentation“. *Raspberry Pi Camera Module Documentation*. Abgerufen am 27.01.2020 von https://www.raspberrypi.org/documentation/hardware/camera/README.md.

The Raspberry Pi Foundation (2020b): „Raspberry Pi 3 Model B Product Brief“. *Raspberry Pi 3 Model B Product Brief*. Abgerufen am 12.01.2020 von https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf.

The Raspberry Pi Foundation (2020c): „Setting up an Apache Web Server on a Raspberry Pi - Raspberry Pi Documentation“. *Apache Web Server Tutorial*. Abgerufen am 27.02.2020 von https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md.

The Raspberry Pi Foundation (2019): „Raspberry Pi 4 Product Brief“. *Raspberry Pi 4 Product Brief*. Abgerufen am 07.02.2020 von https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf.

Ultimaker (2019): „Ultimaker S3 Datasheet“. *Ultimaker S3 Datasheet*. Abgerufen am 23.01.2020 von http://images.3d.ultimaker.com/Web/UltimakerBV/%7B06c0522e-973a-402a-b671-60c5c5636ff0%7D_ultimaker-s3-product-data-sheet-en.pdf?utm_campaign=Download%20UIltimaker%20S3%20Product%20Datasheet%20&utm_medium=email&utm_source=Eloqua&elqTrackId=f898b15c47a843b284e2447f1ae5c693&elq=8afe9047d8d3483f9ffc8e29e2ae2c4c&elqaid=568&elqat=1&elqCampaignId=.

Upton, Eben; Halfacree, Gareth (2014): *Raspberry Pi User Guide*. 2. Hoboken, NJ: John Wiley & Sons.

Wilson, Donald M (1966): „Insect walking“. In: *Annual review of entomology*. 11 (1), S. 103–122. Palo Alto, CA: Annual Reviews.

Winner, Hermann; Hakuli, Stephan; Wolf, Gabriele (2009): *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. 1. Wiesbaden: Springer Vieweg.

Wüst, Klaus (2011): *Mikroprozessortechnik*. 4. Wiesbaden: Springer Vieweg.

Yamane, Katsu; Nakamura, Yoshihiko (2003): „Natural motion animation through constraining and deconstraining at will“. In: *IEEE Transactions on visualization and computer graphics*. 9 (3), S. 352–360. New York, NY: IEEE.

Anhang

Elektronischer Anhang A:	STL-Dateien
Elektronischer Anhang B:	Installationsanleitungen
Elektronischer Anhang C:	Anschlussplan des Geländeroboters
Elektronischer Anhang D:	Struktogramm des Geländeroboters
Elektronischer Anhang E:	Programme
Elektronischer Anhang F:	Conrad Downloadpaket (Conrad Electronic SE 2020)