Masterarbeit

im Studiengang

Elektrotechnik

# Konzeption und technische Realisierung der physischen Komponenten eines kognitiven Agenten sowie dessen physischer Umgebung

# Design and technical realization of the physical components of a cognitive agent as well as its physical environment


| | | |
|---|---|---|
| Betreuer | : | Dr.-Ing. Ronald Römer |
| Zweitprüfer | : | Dipl.-Ing. Christian Richter |
| Vorgelegt am | : | 04.10.2022 |
| Vorgelegt von | : | Friedrich Eckert<br>Matrikelnummer: 3248994<br>friedrich.eckert@b-tu.de |

Brandenburgische Technische Universität Cottbus - Senftenberg

Lehrstuhl Kommunikationstechnik

## Abstract

This thesis documents the design, construction, testing, and evaluation of an artificial intelligence demonstrator. The goal is to make the operation of a cognitive system more tangible, especially for people outside this field, as well as to facilitate the further development of such systems for researchers. The system represents a mouse in a maze that can find and consume food on the maze panels. After defining the requirements and considering possible solutions, a robot with two-wheel drive, stepper motors, and skid steering is created. It orients itself by active LED markers on the fields and detects obstacles by means of collision sensors. With a circular footprint of $75\,\mathrm{mm}$, it is well suited to navigate the maze fields with $100\,\mathrm{mm}$ edge length. The wall configuration of the maze with $5x5$ fields is easily changeable for demonstration of different scenarios. A MATLAB interface allows communication with the robot and connection to artificial intelligence. A functional test confirms the successful implementation of the given requirements. Based on the findings of the prototype, the paper concludes with suggestions for further improvements.

Die vorliegende Arbeit dokumentiert Konzeption, Aufbau, Test und Bewertung eines Demonstrators für künstliche Intelligenz. Ziel ist es, die Funktionsweise eines Kognitiven Systems, insbesondere für Personen außerhalb dieses Fachgebietes, greifbarer zu machen, sowie die Weiterentwicklung solcher Systeme für Forschende zu erleichtern. Das System stellt eine Maus in einem Labyrinth dar, die Nahrungsmittel auf den Labyrinthfeldern finden und konsumieren kann. Nach Festlegung der Anforderungen und Abwägung der Lösungsmöglichkeiten, entsteht ein Roboter mit Zweiradantrieb, Schrittmotoren und Panzerlenkung. Er orientiert sich durch aktive LED Markierungen auf den Feldern und detektiert Hindernisse mittels Kollisionssensoren. Mit einer kreisförmigen Grundfläche von $75\,\mathrm{mm}$, ist er gut geeignet um auf den Labyrinthfeldern mit $100\,\mathrm{mm}$ Kantenlänge zu navigieren. Die Wandkonfiguration des Labyrinths mit $5x5$ Feldern ist zur Demonstration verschiedener Szenarien leicht änderbar. Eine MATLAB-Schnittstelle ermöglicht die Kommunikation mit dem Roboter und die Anbindung an die künstliche Intelligenz. Die erfolgreiche Umsetzung der gegebenen Anforderungen bestätigt ein Funktionstest. Basierend auf den Erkenntnissen des Prototyps schließt die Arbeit mit Vorschlägen zu weiteren Verbesserungen.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| **ToF** | Time of Flight |
| **LIDAR** | Light Detection and Ranging |
| **LED** | Light emitting Diode |
| **POM** | Polyoxymethylen |
| **NEMA** | National Electrical Manufacturers Association |
| **FDM** | Filament Deposition Modeling |
| **SWD** | Serial Wire Debug |
| **SMD** | Surface Mounted Device |
| **DMA** | Direct Memory Access |
| **PWM** | Pulsweitenmodulation |
| **RGB** | Rot Grün Blau |
| **ROM** | Read Only Memory |
| **RAM** | Random Acess Memory |
| **RFID** | Radio Frequency Identification |
| **UART** | Unsiversal Asynchronous Receiver Transmitter |
| **SPI** | Serial Peripherial Interface |
| **IC** | Integrated Circuit |
| **LASER** | Ligh amplification and stimulated emission of Radiation |
| **USB** | Universal Serial Bus |
| **FFT** | Fast Fourier Transform |
| **LiPo** | Lithium Polymer |
| **ISM** | Industrial Scientific Medical |
| **WLAN** | Wireless Local Area Network |
| **ADC** | Analog to Digital Converter |
| **FSM** | Finite State Machine |
| **MOSFET** | Metal Oxide Field Effect Transistor |
| **HAL** | Hardware Abstraction Layer |
| **FIFO** | First in first out |
| **CRC** | Cyclic Redundancy Check |
| **GPIO** | General Purpose Input Output |

**SoC** System on Chip

**TCP** Transmission Control Protocol

# 1. Einleitung

Künstliche Intelligenz ist allgegenwärtig, sie taucht häufig in der Öffentlichkeit und den Medien auf. Bekannte Anwendungsfelder sind zum Beispiel Sprach- und Bilderkennung, autonomes Fahren, personalisierte Werbung und Suchmaschinen. Diese Systeme haben gemein, dass sie auf ihren Anwendungszweck trainiert werden und gelerntes Wissen abrufen können. Sie sind nicht dafür bekannt mitzudenken oder implizite Zusammenhänge zu verstehen. Auch sind diese schwer bis gar nicht überprüfbar. Die Verknüpfungen in diesen künstlichen neuronalen Netzen können so umfangreich werden, dass kaum die Möglichkeit besteht, ihre Handlungsweise vorherzusagen, oder auszuschließen, dass bestimmte ungewollte Verhaltensweisen auftreten. Kognitive Systeme dagegen sollen Zusammenhänge verstehen und mitdenken. Auch ist ihr Verhalten erklärbar [MEY21, S.2].

Der Lehrstuhl Kommunikationstechnik an der BTU Cottbus-Senftenberg ist auf die Erforschung und Entwicklung technischer kognitiver Systeme spezialisiert. Um die Funktionsweise zu veranschaulichen und diese Systeme sowohl der Öffentlichkeit näher zu bringen, als auch die Forschung zu unterstützen, soll ein Demonstrator entwickelt werden. Der Aufbau der aktuellen Forschungsarbeit liegt ein Experiment von Claude C. Shannon zugrunde, der einen Roboter in Form einer Maus in einem Labyrinth aussetzte [MIT]. Die Maus konnte nach Erkunden des Labyrinths von jedem Punkt aus den Ausgang finden. Shannon nannte sie, angelehnt an die Figur der griechischen Mythologie, Theseus. Ziel der Forschung ist, beispielhaft die Maus Theseus um kognitive Aspekte zu erweitern und sie zu befähigen, mehr Probleme zu lösen, als nur den Ausgang zu finden.

Theseus ist nur einer von mehreren Agenten, die im Projektkontext nach mythologischen Figuren benannt sind [MEY21, S.7]. Der Golem ist dabei der ausführende Agent. Die Interaktion mit der Umwelt findet über ihn statt. Der Homunkulus übernimmt die Handlungsauswahl und Wissensverarbeitung. Untereinander stehen sie in Form des Perzeptions-Aktions-Zyklus im Austausch, wobei Perzeptionen der Umwelt vom Golem an den Homunkulus geleitet werden und Aktionsanweisungen in umgekehrter Richtung laufen. Übergeordnet stehen Theseus und Argus, die die Funktion des „Sollen“ und „Dürfen“ steuern. Unter Sollen fällt die Erkundung der Umgebung und das Finden von Objekten, während das Dürfen die Unversehrtheit sichern soll, in-

dem zum Beispiel gefundene Objekte nach Genießbarkeit bewertet werden. Ziel dieser Arbeit ist, einen physischen Golem und eine passende Umgebung herzustellen. Der Golem soll mit dem Homunkulus, der mit allen übergeordneten Agenten als Simulation auf einem Rechner existiert, in Verbindung treten können, sowie mit seiner Umwelt interagieren können. Feldinhalte sollen detektiert und konsumiert werden können. Das transportable Labyrinth muss eine Größe von $5x5$ Feldern haben, die Trennwände sollen verstellbar, sowie gerade und diagonal montierbar sein. Das System soll über eine Schnittstelle aus Matlab erreichbar sein, um alle Perzeptionen entgegenzunehmen, sowie die Aktionen zum Golem weiterzuleiten.

Die vorliegende Arbeit gliedert sich in fünf Abschnitte. Zu Beginn werden die Anforderungen spezifiziert. Sie leiten sich aus der Aufgabenstellung. Im Anschluss folgen Vorüberlegungen zur Lösung der Problemstellung. Verschiedene Ansätze sind gegeneinander abzuwägen, zum Schluss soll eine Liste der zu implementierenden Systeme feststehen. Es folgt die Dokumentation der Umsetzung, einschließlich der auftretenden Probleme und deren Lösungen. In einem Funktionstest werden die Fähigkeiten des Systems überprüft und anschließen ausgewertet. Abschließend werden Verbesserungsvorschläge basierend auf den erlangten Erkenntnissen angeführt.

# 2. Anforderungen

Anhand der vereinbarten Aufgabenstellung werden im Folgenden die Ziele für das zu entwickelnde System definiert.

### Transportables Labyrinth mit 5x5 Feldelementen

Das Labyrinth muss klein genug sein, um transportabel zu bleiben, muss sich in der Mindestgröße jedoch nach dem Roboter richten, da dieser nicht beliebig klein gebaut werden kann. Eine quadratische Feldgröße von $100\,\text{mm}$ Kantenlänge erscheint realisierbar, woraus sich eine Labyrinthkantenlänge von $0{,}5\,\text{m}$ ergibt.

### Flexible, räumliche Konfiguration durch verstellbare Wände, die auch diagonal angeordnet werden können

Die Trennwände müssen entlang der Feldkanten, sowie diagonal über die Felder platzierbar sein. Sie müssen, für den Betrachter deutlich sichtbar, dem Roboter den Weg versperren und von diesem detektierbar sein. Der Roboter darf das Hindernis nicht überwinden können. Die Wandhöhe sollte sich daher an der Höhe des Roboters orientieren. Weiterhin müssen die Trennwände leicht austauschbar sein um eine neue Labyrinthkonfiguration zu erstellen.

### Herstellung der Interaktion mit dem Agenten, Motorik und Sensorik für Zielfindung und -bewertung sowie Konsumtion, Interaktion mit der Umgebung (Feldelemente und Wände des Labyrinths)

Der Roboter muss mit jedem Labyrinthfeld interagieren können. Die Art der Interaktionen umfasst dabei die Detektion und den Konsum von Feldinhalten, sowie zur Einrichtung des Testszenarios im Labyrinth auch die Manipulation von Feldern. Speziell sollen das Platzieren, Aufnehmen und Ablegen von Feldinhalten möglich sein. Um dem Betrachter die Konsumgüter auf den Feldern sichtbar zu machen, müssen diese zusätzlich visuell angezeigt werden.

Zur Navigation des Roboter soll die Umgebung berücksichtigt werden. Falls technisch bedingte Hilfen nötig sind, sollen diese dem Betrachter möglichst verborgen bleiben um die eigenständige Handlungsfähigkeit des Roboters nicht fragwürdig erscheinen zu lassen.

### Anbindung an die Matlab-Programme zur Verhaltenssteuerung

Zur Entscheidungsfindung ist deutlich mehr Rechenleistung notwendig, als sich in den Prototyp des Roboters einbetten lässt. Daher müssen durch die Sensorik erfasste Werte an einen Computer weitergeleitet sowie die daraus getroffenen Handlungsanweisungen von diesem entgegengenommen werden. Die Bewegungsfreiheit des Roboters kann nur mittels drahtloser Kommunikation gewährleistet werden. Auch zur Anbindung an den Computer bietet sich vor dem Hintergrund der Systemportabilität eine Funkschnittstelle an. Unabhängig von der physischen Schnittstelle muss Matlab mit dieser interagieren können.

# 3. Konzeption

Es bestehen viele Möglichkeiten die festgelegten Ziele und Anforderungen zu erfüllen. Dieses Kapitel wägt zwischen den Optionen ab, welche zur Verfügung stehen und setzt die am besten geeigneten Lösungsansätze fest. Die Konzeptionsphase steckt den Rahmen zur Umsetzung ab und stellt somit einen rudimentären Bauplan dar. Begonnen wird mit dem Antrieb, dessen Varianten in Tabelle 1 zusammengefasst sind.

Tabelle 1.: Übersicht der Antriebsarten

| System | Beschreibung | Vorteile | Nachteile | Genutzt |
|---|---|---|---|---|
| Mecanumräder | Räder mit diagonalen Walzen ermöglichen beliebige Bewegung und Drehung in der Ebene | Beliebige Bewegungen möglich, Fahrweg leicht korrigierbar, sehr wendig, schnelle Fahrmanöver | Viele bewegliche Teile, nicht verfügbar in benötigter Größe, vier Motoren, hoher Platzbedarf und Hardwareaufwand | nein |
| Omniwheel (Allseitenrad) | Räder mit querstehenden Walzen ermöglichen beliebige Bewegung und Drehung in der Ebene | Beliebige Bewegungen möglich, Fahrtweg leicht korrigierbar, sehr wendig, schnelle Fahrmanöver, nur drei Motoren | Viele bewegliche Teile, jedes Rad einzeln angetrieben, hoher Platzbedarf und Hardwareaufwand | nein |
| Kettenlaufwerk Panzerlenkung | Bewegung vor und zurück, sowie Kurvenfahrten und Drehung auf der Stelle | Sehr wendig, nur zwei Motoren notwendig, stabiler Stand | leichtgängige Kette ist Spezialanfertigung, nicht gummierte Kette rutscht auf Untergrund | nein |
| Vierrad Panzerlenkung | Bewegung vor und zurück, sowie Kurvenfahrten und Drehung auf der Stelle | Sehr wendig, nur zwei Motoren notwendig, stabiler Stand, Räder einfacher beschaffbar als Kette | Getriebe oder Riemenantrieb nötig für Allradantrieb, höherer Platzbedarf als für zwei Räder | nein |
| Zweirad Panzerlenkung | Bewegung vor und zurück, sowie Kurvenfahrten und Drehung auf der Stelle | Sehr wendig, nur zwei Motoren notwendig, minimaler Platzbedarf und Materialaufwand | Zusätzliche Gleitfüße für sicheren Stand nötig | ja |

## 3.1. Antriebsstrang

### 3.1.1. Mecanumräder

Das Mecanum-Rad wurde von der schwedischen Firma Mecanum AB vorgestellt [Mec]. Es ist umlaufend mit drehbaren Walzen besetzt, die $45°$ zur axialen Richtung gedreht sind, wie Abbildung 1 zeigt.

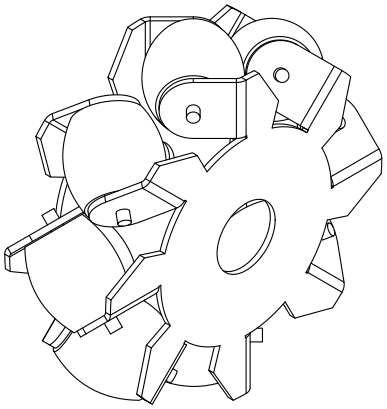

Abbildung 1.: Mecanum-Rad
Quelle: Eigene Abbildung

Durch punktsymmetrische Spiegelung eines Rades am Fahrzeugmittelpunkt entsteht die passende Konfiguration, die beliebige Bewegungen in der Ebene ermöglicht. Durch Gleichlauf aller Räder werden die translatorischen Bewegungen vor und zurück erreicht. Werden die Räder reihum nummeriert, so lässt sich durch Gegenlauf der geraden zu den ungeraden Radnummern eine translatorische Bewegung nach links und rechts erzielen. Die Rotation der vorderen und hinteren Räder hebt sich dabei auf. Durch die Walzen bleibt nur noch der Freiheitsgrad seitwärts bestehen. Wenn ausschließlich die geraden oder ungeraden Räder angetrieben sind, bewegt sich das Fahrzeug diagonal und Rotation entsteht bei Gegenlauf zwischen den linken und den rechten Rädern. Das Fahrzeug ist also sehr wendig und benötigt keinen Rangierbereich. Allerdings müssen die Räder sehr präzise gefertigt sein, damit sie stets Bodenkontakt haben und einen ruhigen Lauf gewährleisten. Mecanumräder mit einem Durchmesser unter $50$ mm waren nicht beschaffbar und müssten demzufolge selbst gefertigt werden. Durch die Walzen haben sie auch ein recht breites Profil, was dem geringen Bauraum ebenso entgegensteht, wie die vier Motoren, die zur unabhängigen Steuerung der einzelnen Räder nötig sind. Trotz klarer Vorteile in der Manövrierbarkeit ist der Einsatz dieser Räder somit nicht sinnvoll.

### 3.1.2. Allseitenräder (Omniwheels)

Allseitenräder folgen einem ähnlichen Prinzip wie Mecanum-Räder. Hier sind die Walzen jedoch, wie in Abbildung 2 dargestellt, axial zum Umfang des Rades montiert[Aca].

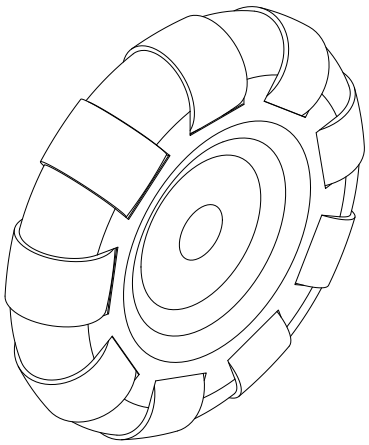

Abbildung 2.: Allseitenrad, engl. omnidirectional wheel (omniwheel)
Quelle: Eigene Abbildung

Ein Fahrzeug ist mit drei dieser Räder ausgestattet, deren Achsen jedoch nicht parallel verlaufen, sondern jeweils um $120°$ zu den benachbarten Achsen gedreht sind. Fahrzeugdrehungen werden durch Gleichlauf aller Räder erzeugt, Translationen im einfachsten Fall durch Gegenläufige Rotation zweier Räder, wobei bei gleicher Drehzahl die Bewegung axial zum ruhenden Rad stattfindet. Durch anteiligen Antrieb aller Räder mit entsprechend angepasster Drehzahl, lassen sich beliebige Bewegungen ausführen. Dieses System besitzt die Wendigkeit von Mecanum-Rädern, benötigt jedoch einen Motor weniger. Allerdings sind auch die Nachteile ähnlich. Viele bewegliche Teile und eine Radbreite, die durch die Walzen diktiert wird, machen den Aufbau groß und anfällig für Störungen. Bereits eine blockierte Walze kann das System temporär unbrauchbar machen. Von der Verwendung von Allseitenrädern wird vor Allem wegen des dritten Motors und des Platzbedarfes der sternförmigen Anordnung abgesehen.

### 3.1.3. Antriebe mit Panzerlenkung

Die im Folgenden beschriebenen Variationen dieses Antriebs vereint, dass sie mit zwei Motoren auskommen, sodass eine kompaktere Baugröße realisierbar ist.

#### Vierradantrieb mit Panzerlenkung

Vier Räder sorgen für einen stabilen Stand und dank der Panzerlenkung sind Drehungen um den Fahrzeugmittelpunkt möglich. Hierbei werden die Räder links gegenläufig zu denen der rechten Seite gedreht. Diagonal und Seitwärtsbewegungen müssen jedoch durch eine Kombination aus Drehungen und geraden Bewegungen ersetzt werden, was die Zeitdauer für Fahrmanöver gegenüber den Mecanum- oder

Allseitenrädern erhöht. Für einen Allradantrieb müssen alle Räder durch ein Getriebe gekoppelt werden, oder es kommt ein Kettenlaufwerk zum Einsatz. Dieses hat den Nachteil, dass entweder viele bewegliche Teile involviert sind, oder eine sehr flexible Gummikette gefunden werden muss, die auch mit geringem Drehmoment flüssige Bewegungen erlaubt. Drehungen auf der Stelle mit vier Rädern haben des Weiteren den Nachteil, dass diese sich auch seitwärts bewegen müssen, da der Drehpunkt in der Fahrzeugmitte liegt. Die Räder oder die Ketten müssen also seitwärts über den Boden rutschen können, was bei einem leichten Roboter mit rutschfesten Reifen/ Ketten zu Problemen führen kann.

**Zweiradantrieb mit Panzerlenkung**
Speziell das Problem mit seitwärts rutschenden Rädern lässt sich beheben, indem nur zwei Räder in einer Achse mit dem Fahrzeugmittelpunkt montiert werden. Für einen stabilen Stand müssen Auflagepunkte vorn und hinten am Roboter geschaffen werden, die zwar Bewegungen in alle Richtungen zulassen, jedoch nicht unbedingt rutschfest sein müssen, da sie keine Traktion besitzen. Diese Auflagepunkte können in diesem Fall einfache Gleitflächen sein, bei höherer Last können drehbar gelagerte Kugeln verwendet werden. Der Vorteil von nur zwei Motoren und dem schmalen Radprofil, da bewegliche Walzen entfallen, machen den Zweiradantrieb mit Panzerlenkung zur favorisierten Antriebsart.

### 3.1.4. Motorauswahl

Ein Erstversuch zeigte, dass Gleichstrommotoren ohne Positionsrückkopplung mittels Encoder nicht geeignet sind, um präzise Bewegungen auszuführen. Da sich der Roboter auf ebener Fläche bewegen soll, sind keine Drehmomentsprünge zu erwarten. Aus diesem Grund kann eine Positionsregelung mittels Encoder durch eine Positionssteuerung per Schrittmotor ersetzt werden. Dies spart viel Rechenkapazität ein und der Platz, der sonst vom Encoder belegt wird, kann für einen stärkeren Motor verwendet werden. Miniaturmotoren wie der SMC10-20 von Minebea[Min] sind mit $10\,\mathrm{mm}$ Durchmesser und $12{,}5\,\mathrm{mm}$ Länge zwar klein genug, um sogar als Nabenmotor in einem Mecanum-Fahrwerk eingesetzt zu werden, allerdings sind dieses und identisch aussehende namenlose Motormodelle als Stellmotoren konzipiert. Im Dauerbetrieb überhitzen sie schnell, wie sich im Test gezeigt hat. Für diese Anwendung sollte auf Schrittmotoren zurückgegriffen werden, die für eine längere Betriebsdauer ausgelegt sind.

Permanentmagnet-Schrittmotoren besitzen einen permanentmagnetischen Rotor und einen Stator mit Magnetspulen. Anhand des elektromagnetisch eingeprägten Drehfeldes richtet sich der Rotor aus. Reluktanz-Schrittmotoren dagegen besitzen einen

ferromagnetischen Weicheisenkern, der kein eigenes Magnetfeld erzeugt. Stattdessen sind die zueinander zeigenden Flächen von Rotor und Stator gezahnt. Das eingeprägte Magnetfeld sorgt nun dafür, dass die Zähne sich derart zueinander ausrichten, dass der Luftspalt minimiert wird. Da jedoch die Winkel zwischen diesen an Rotor und Stator ungleich sind, gibt es für jede angesteuerte Spule nur einen stabilen Zustand. Durch ein Drehfeld, dreht sich auch der Rotor weiter. Über die Anzahl der Zähne kann der kleinste Vollschrittwinkel konstruktiv festgelegt werden. Die Kombination dieser Systeme wird hybrid Schrittmotor genannt. Sie haben die feine Schrittauflösung des Reluktanz-Schrittmotors und ein hohes Drehmoment, wie der Permanentmagnet-Schrittmotor [Sch, S.4].

Die National Electrical Manufacturers Association (NEMA) hat sich auf einen Standard für die Abmaße von hybriden Schrittmotoren geeinigt. Die Maße des quadratischen Flansches in zehntel Inch wird an das NEMA Kürzel angehängt [NEM, S.84]. So steht beispielsweise ein NEMA 17 Motor für ein Flanschmaß von $1{,}7$ inch, also etwa $43$ mm. Der kleinste verfügbare Motor zum Zeitpunkt der Recherche ist das Modell 8HS15-0604S vom Typ NEMA 08 und hat Abmaße von $\square 20 \times 38$ mm [OYOb]. Während der Arbeiten am Prototyp konnte jedoch noch eine Beschaffungsquelle für das kleinere NEMA 06 Modell 6HS12-0304S mit Abmaßen von $\square 14 \times 30$ mm gefunden werden [OYOa]. Für die Folgeversion des Prototyps wird dieser Motortyp empfohlen, da der geringere Stromverbrauch die Akkulaufzeit verlängern kann und so auch mehr Platz für eine Verkleidung des Roboters zur Verfügung steht.

### 3.1.5. Motortreiberauswahl

Schrittmotoren können im einfachsten Fall im Vollschrittbetrieb laufen. Dabei springt der Rotor jeweils einen Vollschritt weiter. Da die beiden Motormodelle $200$ Schritte pro Umdrehung haben, bedeutet das einen Sprung von je $1{,}8°$, was eine starke Vibration und Laufunruhe zur Folge hätte. Abhilfe schafft die Eingliederung von Zwischenspannungen mit denen der Rotor mittig zwischen zwei Vollschritten gehalten werden kann. Dies wird dann Halbschrittbetrieb genannt. Spezielle Motortreiber wie der TMC2130 von Trinamic sind in der Lage einen Vollschritt in bis zu 256 Mikroschritte [Tri, S.1] zu unterteilen und so de Facto ein sinusförmiges Drehfeld zu erzeugen. Der Motor läuft damit sehr vibrationsarm und auch sehr leise.

### 3.1.6. Motoranordnung und Getriebe

Die ausgewählten NEMA 08 Schrittmotoren sind zwar innerhalb der Normreihe sehr klein, relativ zur gewünschten Robotergröße nehmen sie jedoch viel Platz in Anspruch. Ein Nabenantrieb entfällt automatisch, da die Motoren nebeneinander keinen Platz

haben. Die einzige Möglichkeit ist, die Motoren versetzt anzuordnen und das Drehmoment per Getriebe auf die Räder zu übertragen. Hierbei stehen verschiedene Über- und Untersetzungen zur Auswahl. Aus Sicht der Montierbarkeit ist es wünschenswert, wenn vor dem Motor nur ein kleines Zahnrad sitzt, damit Platz zur Befestigung des Motorflansches bleibt. Daraus folgt, dass ein Untersetzungsgetriebe eingesetzt wird. Die Untersetzung beschränkt sich auf die verfügbaren Zahnräder. Sie müssen groß genug sein, um die Radachse neben dem Motor platzieren zu können, jedoch nicht größer als das Rad selbst. Eine Untersetzung von $2:1$ erscheint sinnvoll, da so der ideale Abstand zwischen Motoren und Radachse eingehalten werden kann.

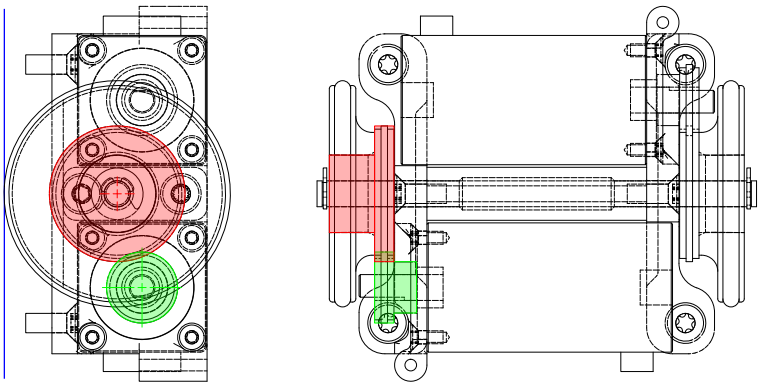

Abbildung 3.: Antriebsstrang des Roboters
Quelle: Eigene Abbildung

Abbildung 3 zeigt den Antriebsstrang in Seiten- und Draufsicht. Das Ritzel ist grün, das größere Zahnrad rot markiert, welches direkt an dem Rad montiert ist. Die blaue Linie zeigt das Bodenniveau. Die Zeichnung zeigt, dass die Zahnradanordnung die Verschiebung der Radachse unter die Motorwellen ermöglicht, was mehr Platz für die Elektronik und Sensorik schafft, die unter dem Motor liegen soll. Durch diese Achsverschiebung verringert sich auch der Abstand zwischen den Motoren.

## 3.2. Leitsysteme

Die physische Ausführung der von außen gegebenen Befehle erfordert, dass der Roboter sich orientieren sowie Abweichungen vom Weg erkennen und korrigieren kann. Diese müssen während sie auftreten oder nachträglich korrigiert werden, sodass der Roboter immer wieder einen definierten Zustand einnimmt. Hierfür werden Systeme benötigt, die die Umwelt sensorisch wahrnehmen und damit den Roboter leiten können. In Tabelle 2 sind interne Leitsysteme beschrieben, mit denen der Roboter sich selbst orientieren kann. Sie sind nicht auf speziell präparierte Umgebungen angewiesen und müssen nicht versteckt werden, da der Roboter offensichtlich direkten Zugriff darauf hat. Anschließend werden Systeme vorgestellt, die Veränderungen der Umwelt bedürfen, in diesem Fall des Labyrinths. Diese externen Systeme ermöglichen dem

Roboter sich zu orientieren, laufen jedoch Gefahr von außen erkennbar zu sein und den Roboter als eigenständigen Akteur infrage zu stellen, wie bereits in den Anforderungen festgestellt. Daher ist die Unsichtbarkeit eines Leitsystems auch ein wichtiges Kriterium bei der Findung geeigneter Strategien.

Tabelle 2.: Übersicht über robotergebundene Leitsysteme

| System | Beschreibung | Vorteile | Nachteile | Genutzt |
|---|---|---|---|---|
| Statisches LIDAR | Mehrere im Bogen angeordnete ToF Distanzsensoren | Ermöglicht vorausschauendes Fahren und Hindernisvermeidung | Nur wenige Messpunkte, hohe Hardwarekosten, hoher Rechenaufwand um Muster in Punktewolke zu erkennen | nein |
| LIDAR | Rotierender Distanzsensor | Ermöglicht vorausschauendes Fahren und Hindernisvermeidung, Mehr Messpunkte als statisches LIDAR | Teuer, hoher Rechenaufwand für Mustererkennung, kein kompaktes System für kurze Distanzen verfügbar | nein |
| Vereinfachtes LIDAR (VL6180) | Einzelner ToF Distanzsensor tastet Umgebung ab, während Roboter sich auf der Stelle dreht | Preisgünstiger als andere LIDAR-Varianten, beliebig hohe Winkelauflösung | Sehr langsam, hoher Rechenaufwand für Mustererkennung | ja |
| ToF ArraySensor (VL53L5CX) | ToF Distanzsensor mit Feld aus 8x8 Messpunkten | Ermöglicht vorausschauendes Fahren und Hindernisvermeidung, Mehr Messpunkte als statisches LIDAR, Kostengünstig, 60 Hz Abtastrate | Hoher Rechenaufwand für Mustererkennung, war zu Planungsbeginn noch nicht verfügbar | nein* |
| Bewegungsverfolgung | Roboter detektiert und summiert Wegänderungen zur Positionsbestimmung | Hohe Ortsauflösung, direkte Rückmeldung zur Fehlererkennung | Großer Platzbedarf im Roboter, keine Einzelbeschaffung möglich | ja |
| Kollisionsdetektor | Erfasst direkte Berührung von Hindernissen | Sehr preiswert, notwendig zur Schadensabweisung | Ermöglicht keine Orientierungserkennung | ja |

* System sollte in zukünftiger Version verwendet werden

### 3.2.1. LIDAR

Light Detection and Ranging (LIDAR) ist ein optisches Verfahren zur Distanzmessung [Til18, S.30]. Hierbei wird ein Laserstrahl über einen Raumwinkel gescannt und mittels Laufzeitmessung oder Triangulation die Distanz zum Ziel bestimmt. So kann ein zwei- oder dreidimensionales Bild der Umgebung rekonstruiert werden. In der gän-

gigen Bauform rotiert das Messsystem und erreicht so einen Rundumblick. Es ist zur Zeit der Recherche jedoch kein System verfügbar, das klein genug für den Roboter ist. Auch die minimale Messdistanz stellt ein Problem dar. Bei dem kleinsten gefundenen LIDAR System beträgt sie $100$ mm, was einer ganzen Feldlänge entspricht. Hindernisse direkt vor dem Roboter können so nicht erkannt werden. Eine Alternative dazu ist der VL6180, ein Time of Flight (ToF) Sensor von ST Microelectronics. Er zeichnet sich durch eine kompakte Bauform von $4{,}8 \times 2{,}8 \times 1$ mm aus und hat einen Messbereich von $0$ bis $62$ cm [ST b, S.8]. Hiermit ließe sich ein statisches LIDAR realisieren, bei dem mehrere dieser Sensoren im Bogen montiert werden und so die Distanz zu Hindernissen in verschiedene Richtungen messen. Da alles Sensoren dies gleichzeitig tun können, dauert die Messung selbst nicht länger als bei einem einzelnen Sensor. Die Messpunkte sind allerdings auf die Sensoren begrenzt, was die Auswertung und Erkennung von Hindernissen erschwert. Zur Erhöhung der Messpunktanzahl könnte der Roboter sich um einen kleinen Winkel drehen und die Ergebnisse einer erneuten Messung mit den alten Werten kombinieren. Um Sensoren einzusparen kann auch ein einzelner Sensor genutzt werden um während einer Drehung die Umgebung zu scannen. Im Test hat sich diese Lösung jedoch als zu langsam erwiesen. Alle vorgestellten Systeme erfassen die Distanz nur innerhalb einer Ebene, sodass es passieren kann, dass der Sensor über ein Hindernis hinweg schaut. Da auch die Erkennung und Klassifizierung von Hindernissen viel Rechenkapazität beansprucht und die zur Verfügung stehenden Lösungen ungeeignet sind, wird auf die Anwendung von LIDAR verzichtet. Allerdings wird ein einzelner VL6180 Sensor an der Vorderseite des Roboters montiert, da er kostengünstig ist und für einfache Distanzmessungen in Fahrtrichtung geeignet ist.

Nach dem Aufbau des Prototyps stellte ST Microelectronics das Nachfolgemodell ihres Time of Flight Sensors in Form eines 8x8 Feldes vor. Der VL53L5CX hat die gleichen Abmaße wie die Vorgängersensoren [ST a, S.3], ist jedoch in der Lage das Sichtfeld von $45°$ horizontal und vertikal in etwa $5{,}6°$ Sektoren zu unterteilen und diese mit $60$ Hz Bildrate abzutasten. Die Mindestdistanz von $20$ mm erfordert eine sorgfältige Auswahl der Montageposition. Die Möglichkeiten, die dieser Sensor bietet, könnten in Nachfolgemodellen des Roboters jedoch eine große Rolle spielen. Der Vorteil des Sensors gegenüber den anderen vorgestellten LIDAR Systemen ist, dass er schnell abtastet und nicht nur innerhalb einer Ebene misst.

### 3.2.2. Bewegungsverfolgung

Eine weit verbreitete Technik um die Bewegung auf einer Fläche zu verfolgen, wird in optischen Computermäusen verwendet. Eine Kamera mit nur einigen hundert Bildpunkten [Ava, S.23] nimmt Bilder der Oberfläche auf, die anschließend miteinander

verglichen werden. So kann die Verschiebung in X- und Y-Richtung zwischen Messungen bestimmt werden. Um auch die Rotation zu erfassen, sind zwei Sensoren notwendig. Werden sie z.B. in X-Messrichtung auf gleicher Höhe und symmetrisch zum Fahrzeugmittelpunkt montiert, müssen lediglich die Messwerte der Y-Richtung berücksichtigt werden, da Seitwärtsbewegungen aufgrund der Räder nicht möglich sind. Abbildung 4 zeigt den detektierten Y-Wert des linken Sensors in rot und des rechten Sensors in grün. Die bogenförmige Verzerrung des Pfades kann vernachlässigt werden, da die Ausleseintervalle sehr klein sind und bei den entsprechend kleinen Änderungen eine lineare Approximation ausreicht.

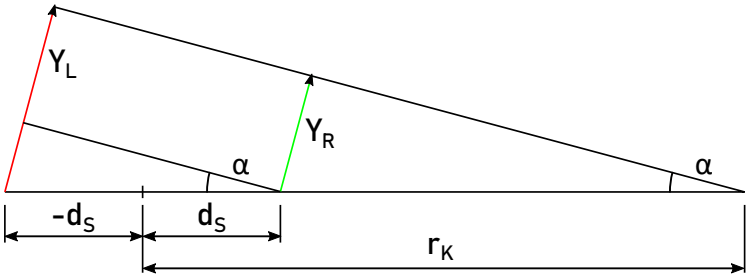


Abbildung 4.: Auswertung der Sensoren zur Bewegungsverfolgung
Quelle: Eigene Abbildung

Der Winkel $\alpha$ um den der zurückgelegte Weg von der Ausgangsposition abweicht berechnet sich durch Gleichung 3.1.

$$\alpha = arcsin\left(\frac{Y_L - Y_R}{2d_S}\right) \tag{3.1}$$

Um den Kurvenradius $r_K$ zu bestimmen, der sich auf den Fahrzeugmittelpunkt bezieht, bietet sich ein virtueller Sensor in genau diesem Punkt an. Die im Fahrzeugmittelpunkt zurückgelegte Distanz ist das arithmetische Mittel beider Messwerte $Y_L$ und $Y_R$. So ergibt sich Gleichung 3.2.

$$r_K = \frac{Y_L + Y_R}{2 \cdot sin(\alpha)} \tag{3.2}$$

Für $sin(\alpha)$ ist mit Gleichung 3.1 bereits eine implizite Lösung vorhanden. Durch einsetzen ergibt sich Gleichung 3.3 für den zurückgelegten Weg.

$$r_K = d_S \cdot \frac{Y_L + Y_R}{Y_L + Y_R} \tag{3.3}$$

Tabelle 3 zeigt, wie die berechneten Ergebnisse zu interpretieren sind.

Maussensoren sind so spezialisiert auf die Anwendung in Computermäusen, dass keiner

Tabelle 3.: Auswertungsschema für Winkel und Distanz

| | Vorwärts | Rückwärts | Punktuell |
|---|---|---|---|
| Linkskurve | $\alpha < 0,\ r_K < 0$ | $\alpha > 0,\ r_K < 0$ | $\alpha < 0,\ r_K = 0$ |
| Rechtskurve | $\alpha > 0,\ r_K > 0$ | $\alpha < 0,\ r_K > 0$ | $\alpha > 0,\ r_K = 0$ |

der großen Bauteilhändler sie noch im Sortiment führt. Die Folge ist, dass diese Sensoren aus Computermäusen zurückgewonnen werden müssen. Die zwei gängigen Arten dieser Sensoren sind mit LED- oder mit LASER-Beleuchtung des Untergrundes, auf dem die Maus liegt. Die LED-Variante scheint die preisgünstigere und leichter beschaffbare zu sein, hat aber den Nachteil, dass viel Platz für die separate LED eingenommen wird. Die LASER-Lichtquelle ist in den entsprechenden Sensoren, wie dem ADNS-7530 oder dem ADNS-7550 von Avago, bereits enthalten und spart Platz, weshalb diese Bauweise bevorzugt wird. Darüber, ob die Bewegungsverfolgung wirklich notwendig ist, kann zu diesem Zeitpunkt keine sichere Aussage getroffen werden. Eher hängt dies vom Spiel innerhalb des Antriebsstranges und den anderen Leitsystemen ab, da die Maussensoren nur die relative Positionsänderung verfolgen und keine absolute Orientierungsfindung ermöglichen. Auch zu den Toleranzen des Messsystems selbst, liefert das Datenblatt keine genauen Angaben. Da dieses System jedoch das einzige ist, mit dem eine Bewegungsverfolgung möglich ist, es viel Platz benötigt und auch einen bestimmten Abstand zum Boden braucht, muss es direkt eingeplant werden. Ein Nachrüsten wäre nicht möglich.

### 3.2.3. Kollisionssensor

Der einfachste Sensor zur Wegfindung ist ein Kollisionssensor, der Hindernisse bei Berührung erkennt. Er lässt sich durch einen leichtgängigen Taster realisieren, wovon hier jedoch aus mehreren Gründen abgesehen wurde. Zum einen muss bei den meisten Tastern aufgrund der erwünschten haptischen Rückmeldung ein bestimmter Druckpunkt überwunden werden. Dies erschwert jedoch die Detektion von Hindernissen, da er die Erkennungsschwelle anhebt. Zum anderen ist zum Zeitpunkt der Bauteilrecherche kein Taster mit ausreichend kompakter Bauform bei gleichzeitig niedriger Betätigungskraft verfügbar. Der Lösungsansatz ist stattdessen gefederte Puffer zu verwenden, deren Verschiebung von einer Lichtschranke detektiert wird. Die Betätigungskraft kann so frei eingestellt werden.

Tabelle 4.: Übersicht über umgebungsgebundene Leitsysteme

| System | Beschreibung | Vorteile | Nachteile | Genutzt |
|---|---|---|---|---|
| Optisch mit Linien (Wegfolger) | Roboter folgt aufgedruckten Linien | Sehr kostengünstig, einfache Detektion | Sichtbar für Betrachter, Störanfällig bei Abnutzung | nein |
| Optisch mit Linien und Markern | Kamera erkennt aufgedruckte Linien und Marker | Hohe Ortsauflösung, sehr genaue Positionierung | Hoher Hardware und Rechenaufwand, Maussensor hat als Kamera zu geringe Abtastrate | nein |
| eingelassene LED Marker | Roboter detektiert in Bodenpanelen eingelassene Infrarot LEDs | Kaum sichtbar, kodierbar, kostengünstig, hohe Ortsauflösung | Mit Kameras ohne Infrarotfilter sichtbar | ja |
| seitliche optische Marker | LEDs oder Laser überstrahlen das Labyrinth im Raster von der Seite her | Für Betrachter kaum sichtbar, kodierbar | Trennwände erschweren Sichtkontakt, zusätzlicher Hardwareaufwand am Labyrinth, genaue Ausrichtung und Kalibrierung notwendig | nein |
| Induktiv mit Führungsschienen | Roboter detektiert und folgt ferromagnetischen Schienen unter den Bodenpanelen | Von außen unsichtbar | Großer Schaltungsaufwand, wahrscheinlich geringe Ortsauflösung, Wegkreuzungen sind schwer aufzulösen | nein |
| Elektromagnetische Führungsschienen | Roboter detektiert magnetisches Feld stromdurchflossener Leiterbahnen | Kodierbar um Kreuzungsproblem zu lösen, kann unsichtbar für Betrachter sein | Sehr hoher Stromverbrauch, strahlt Störsignale ab, sehr großer Hardwareaufwand um Leiterkreuzungen zu vermeiden | nein |
| Magnetischer Kompass | Roboter detektiert Position und Orientierung von Magneten mittig unter Bodenpanelen | Von Außen unsichtbar, magnetischer Encoder IC macht Detektion einfach und kostengünstig, instantane Orientierungserkennung | Detektion außerhalb der Feldmitte nicht zuverlässig (sprunghafte und unbrauchbare Messwerte) | nein |
| RFID | Roboter nutzt RFID Tags zur Positionierung und Orientierung | Kann von außen unsichtbar sein, Mitnutzung kostengünstig, da zur Interaktion notwendig | Geringe Ortsauflösung, Orientierung nur durch Abgleich mit benachbartem Feld möglich (hoher Navigationsaufwand) | nein** |

** System zwar verwendet, jedoch nicht zur Orientierung in der Umgebung

Tabelle 2 stellt Systeme vor, die dem Roboter mit externen Vorrichtungen ermöglichen, seine Position und Orientierung zu bestimmen. In den nachfolgenden Kapiteln werden Vor- und Nachteile der genannten Lösungen ausgeführt und abgewägt.

### 3.2.4. Linienfolger

Ein sehr einfaches Leitsystem stellt Folgen von aufgedruckten Linien mit hoher Reflektivität dar. Der Detektor besteht aus einer Lichtquelle, die den Boden beleuchtet und zur linken und rechten Seite davon je einem Detektor. Wann immer ein Detektor eine stärkere Lichtreflexion empfängt, ist das Fahrzeug zu stark in diese Richtung vom Pfad abgewichen und es muss gegengesteuert werden. Ein offensichtlicher Nachteil dieses Systems ist, dass die Linien sichtbar sind. Ebenfalls getestet wurden Farbmarker, die unter ultraviolettem Licht fluoreszieren. Diese sind dann zwar mit bloßem Auge schwer bis nicht erkennbar, benötigen jedoch einen hellen Hintergrund, was den Kontrast zur Linie jedoch auch senkt. Da sich mit diesem System auch nicht die Orientierung bestimmen lässt, wird darauf verzichtet.

### 3.2.5. Optische Markererkennung

Wird der vorherige Lösungsansatz erweitert und eine Kamera zur Detektion verwendet, so lassen sich auch Marker erkennen, die die Himmelsrichtung im Labyrinth anzeigen. Um den Rechenaufwand gering zu halten sollte eine Kamera mit niedriger Auflösung zum Einsatz kommen. Zu diesem Zweck kann der im Unterabschnitt 3.2.2 vorgestellte Maussensor als Kamera verwendet werden. Tests mit mehreren verschiedenen Sensortypen ergaben jedoch, dass die Bildrate mit etwa 1 Hz viel zu gering ist, um eine Bilderkennung während der Fahrt zu realisieren. Grund dafür ist, dass alle Bildpunkte mit einem eigenen Lesezugriff ausgelesen werden müssen, nachdem das entsprechende Ready-Flag aktiv geworden ist. Diese Tatsache macht die Bilderkennung via Maussensor unbrauchbar.

### 3.2.6. LED Marker

Da aufgedruckte Linien für den Betrachter sichtbar sind und den Roboter in seinen Fähigkeiten als simpel oder eingeschränkt wirken lassen könnten, wurde diese Lösung verworfen. Die einfache Realisierung und die zuverlässige Arbeitsweise, zumindest unter Idealbedingungen, werden mit einem aktiven optischen Leitsystem aufgegriffen und verbessert. Hierbei werden Infrarot LEDs im Bodenpanel montiert. Eine markiert die Feldmitte, eine weitere gibt die Orientierung an. Infrarotlicht liegt außerhalb des sichtbaren Spektrums und bleibt dem Betrachter somit verborgen. Auch leichter Staub

kann das System nicht stören, da die LEDs durch ihn hindurch leuchten können. Um beide Marker LEDs unterscheiden zu können werden diese mit unterschiedlichen Frequenzen gepulst. Der passende Sensor besteht aus drei Teilsensoren, die entlang der Mittelachse in Fahrtrichtung an der Roboterunterseite angebracht sind. Ein punktförmiger Sensor sitzt genau in der Mitte, davor und dahinter je ein länglicher Sensor.

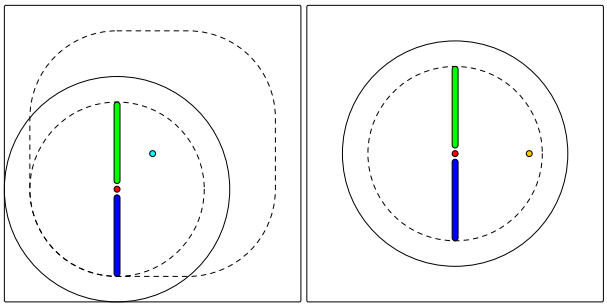

Abbildung 5.: Anordnung des LED Markersystems und der Detektoren
Quelle: Eigene Abbildung

Abbildung 5 zeigt den Aufbau und die Randbedingungen des Systems. Im linken Bild ist zu sehen, wie lang die äußeren Sensoren sein müssen, um von jedem Punkt auf dem Feld den mittleren Marker (türkis dargestellt) finden zu können. Der vom Sensor abdeckbare Bereich ist mit einer Strichlinie umrandet. Der linke Teil von Abbildung 5 zeigt die mögliche Lage des gelb eingefärbten Richtungsmarkers. Er muss vom Mittelpunkt aus durch eine Drehung detektierbar sein, sollte aber nicht zu dicht am Mittelpunkt liegen, da die Winkelunsicherheit bei einem kleineren Schwenkradius auch sinkt. Dieser Effekt kommt dadurch zustande, dass die Sensoren auf der gesamten Länge eine konstante Erfassungsbreite haben, die großer als null ist. Ist diese Breite zum Beispiel zwei Millimeter, so bedeutet dies bei geringem Abstand zum Robotermittelpunkt einen größeren Detektionswinkel, als bei hohem Abstand. Mit dem Detektionswinkel, in dem der Sensor sichtbar ist, steigt auch die Winkelunsicherheit bei der Ausrichtung des Roboters. Um sich auf dem Feld auszurichten, sucht der Roboter zunächst mit einer Drehung auf dem Feld nach dem Mittelpunktmarker. Anschließend bewegt er sich in die Richtung, in der der längliche Sensor angeschlagen hat, bis der Mittlere Sensor sich über dem Mittelpunktmarker befindet. Nun dreht sich der Roboter in der Feldmitte, bis der Orientierungsmarker erfasst wird. Damit ist eine mittige Positionierung und exakte Ausrichtung sichergestellt. Die Sensorauswertung findet im Frequenzbereich statt. Mit je einer FFT wird das Spektrum der drei Sensorsignale bestimmt und in den entsprechenden Frequenzbereichen eine Amplitudenauswertung vorgenommen. Die Fensterlänge wird so lang wie möglich gewählt, muss aber kurz genug sein, dass der Roboter sich bei maximaler Drehgeschwindigkeit nicht innerhalb eines Fensters am Marker vorbeibewegen kann. Diese Lösung ist vielversprechend, da sie Ausrichtung und Orientierungsfindung ermöglicht und das Aussehen der Bodenpanele nicht beeinträchtigt.

### 3.2.7. Seitliche optische Marker

Mit Laserdioden können die Fahrwege von der Feldseite überleuchtet und damit markiert werden. Sie stellen einen externen Spurhalteassistenten dar. Die Verwendung für dieses Projekt wurde jedoch aus Praktikabilitätsgründen wieder verworfen, da Laserdioden und Sensoren auf einer Höhe über den Trennwänden montiert werden müssten und daher einen deutlichen Mehraufwand in der Fertigung darstellen würden. Die Möglichkeiten unterscheiden sich kaum von denen der LED Marker im Boden, die Umsetzung ist jedoch aufwendiger.

### 3.2.8. Induktiver Wegfolger

Werden unter den Feldern ferromagnetische Drähte entlang der Hauptachsen platziert, so lassen sich diese induktiv durch die Platine hindurch detektieren und der Roboter kann der Spur folgen. Der Vorteil dabei ist die Unsichtbarkeit gegenüber aufgedruckten Linien, die Detektion ist jedoch deutlich aufwändiger. Auch die Orientierung ist hiermit nicht bestimmbar. Aufgrund besserer Alternativen wird auf einen Test dieses Systems verzichtet.

Um die Orientierung doch zu bestimmen, könnte der magnetische Draht durch Leiterbahnen ersetzt werden, auf die ein gepulster Strom eingeprägt wird. Die Pulsfrequenz könnte zur Identifikation genutzt werden. Nachteil dieses Systems ist der hohe Stromverbrauch, die umständliche Detektion und die intensiven Streufelder dank des gepulsten Stromes. Außerdem müssten die Bodenplatinen mit ausreichend Lagen gefertigt werden, um alle Leitdrähte gerade und diagonal, sowie die RFID Spule unterzubringen. Aus den genannten Gründen wird von der Verwendung abgesehen.

### 3.2.9. Magnetischer Kompass

Die Feldmitte und Orientierung wird bei diesem Lösungsansatz mit einem diametral polarisierten Magneten unter dem Bodenpanel markiert. Der Roboter kann dann mit einem magnetischen Encoder die Polarisierungsrichtung bestimmen und sich am Verlauf des Magnetfeldes ausrichten. Gegenüber der Methode mit LED Markern ist hierfür nur eine einzige Messung notwendig. Solange der Roboter genau über dem Magneten steht, hat sich dieses Verfahren im Test bewiesen. Doch schon bei geringen Abweichungen kann die Orientierung nicht mehr eindeutig bestimmt werden. Auch das Auffinden der Mitte ist so nicht möglich, daher wird dieses Verfahren nicht implementiert.

### 3.2.10. RFID

Da jedes Feld mit einem RFID Tag ausgestattet ist, liegt es nah, dieses auch zur Erkennung der Feldmitte zu verwenden. Die prinzipielle Funktionsweise konnte im Test gezeigt werden. Um die Ortsauflösung zu erhöhen, müssen jedoch die Durchmesser der Spulen des Lesegerätes und des Tags so klein wie möglich gehalten werden. Dies resultierte in einer schwächeren magnetischen Kopplung und damit einhergehend in einer schlechten Auslesezuverlässigkeit. Dies wäre hinderlich für den eigentlichen Verwendungszweck, weswegen auf die Nutzung von RFID zur Ausrichtung des Roboters verzichtet wird.

## 3.3. Kommunikation und Stromversorgung

In Kapitel 2 wurde bereits festgesetzt, dass eine WLAN Schnittstelle zur Anbindung an den PC genutzt werden soll. Die kostengünstigste Lösung, WLAN Funktionalität zu nutzen, ist wahrscheinlich der ESP32 Mikrocontroller von Espressif. Zusammen mit den Programmbibliotheken bildet er eine flexible und anpassbare Plattform. Da die Funkschnittstelle jedoch viel Strom verbraucht, kann diese nicht direkt auf dem Roboter montiert werden. Der ESP32 benötigt im WLAN Sende- und Empfangsbetrieb [Esp, S.46] je etwa $180\,\mathrm{mA}$ und $100\,\mathrm{mA}$. Die Lösung ist ein Brückenmodul, das die Nachrichten zwischen beiden Funksystemen weiterleitet. Für die Roboterseitige Schnittstelle hat sich aus anderen Projekten der CC2500 von Texas Instruments bewährt. Da auch bereits getestete Treiber zur Verfügung stehen, wurde auf eine Recherche nach weiteren Systemen verzichtet. Der CC2500 sendet und empfängt im $2{,}54\,\mathrm{GHz}$ ISM-Band, genau wie WLAN. Der Stromverbrauch beläuft sich auf ungefähr $20\,\mathrm{mA}$ im Sende- und Empfangsbetrieb und auf unter $1\,\mu\mathrm{A}$ im Standby [Tex, S.7,8]. In Summe lässt sich so der Stromverbrauch des Roboters so deutlich senken und demzufolge ein kleinerer Akku verbauen. Die geeignetste Lösung zur Stromversorgung des Roboters ist ein Lithium-Akku, da dieser Typ weit verbreitet und preiswert ist, als auch eine sehr hohe Energiedichte besitzt [SAU21, S.4]. Für ein kompaktes System mit möglichst langer Akkulaufzeit bietet sich demzufolge diese Technologie an. Bei den großen Bauteilhändlern können vor allem die Lithium Polymer (LiPo)-Akkus aufgrund ihrer Quaderform überzeugen, da diese sehr platzsparend einbaubar sind.

## 3.4. Interaktion mit der Umwelt

Damit der Roboter virtuelle Gegenstände auf einem Feld sehen und diese auch aufnehmen, ablegen und konsumieren kann, ist eine drahtlose Schnittstelle zu jedem Bodenpanel notwendig. Neben Lösungen wie optischer Kommunikation über Infrarot-

licht, oder einem Funksystem, wie bereits zur Kommunikation mit dem PC geplant, bietet sich Radio Frequency Identification (RFID) als die kostengünstigste Variante an. ST Microelectronics bietet aktive RFID Tags an, die sich sowohl drahtlos, als auch drahtgebunden lesen und beschreiben lassen. Der Vorteil hierbei ist, dass der Roboter zum Programmieren der Felder verwendet werden kann und die Mikrocontroller der Felder selbst ihren Inhalt lesen, prüfen und ergänzen können. Beispielsweise sollen die Feldkoordinaten zusammen mit den Gegenständen auslesbar sein und der Feldinhalt visuell mit RGB LEDs angezeigt werden. All das ist mit den genannten ICs möglich. Es sollen deshalb Bauteile von ST Microelectronics zum Einsatz kommen, da das Ökosystem dann nicht verlassen werden muss. Sowohl alle Mikrocontroller, der ToF Sensor, als auch der RFID Lese- und Schreib IC stammen von diesem Hersteller. Auch sind passende Programmbibliotheken verfügbar, die die Entwicklungszeit verkürzen.

## 3.5. Systemkonzept

Nach ausführlichen Abwägungen und Systemtests stehen die wesentlichen Systemkomponenten fest. Der Roboter bewegt sich mit einem Zweiradantrieb mit hybriden Schrittmotoren fort. Die Panzerlenkung ermöglicht Drehungen auf der Stelle und macht den Roboter sehr wendig. Mikroschrittbetrieb wird vom Motortreiber unterstützt, sodass die Motoren geräuscharm und mit minimalen Vibrationen arbeiten können. In Verbindung mit der Getriebeuntersetzung sind sehr präzise Bewegungen möglich. Der Roboter kann seine Umwelt mit Kollisionssensoren vorn und hinten, sowie mit einem Laser Distanzsensor vorn, wahrnehmen. Zwei Maussensoren auf der Unterseite geben Rückmeldung über die ausgeführten Bewegungen. Sie detektieren gerade Bewegungen und Drehungen. Außerdem erkennt der Roboter die Feldmittelpunkte und Himmelsrichtung mit einem Sensorfeld auf der Unterseite, das die Infrarot LEDs in den Bodenpanelen detektieren kann. Dieses System ist von außen kaum sichtbar und kann kostengünstig realisiert werden. Mit den Feldern des Labyrinths interagiert der Roboter via RFID Tags, die sowohl drahtlos, als auch vom Feld drahtgebunden beschrieben und ausgelesen werden können. RGB LEDs zeigen im Labyrinth die jeweiligen Feldinhalte an, sodass diese von Außen sichtbar sind.

# 4. Umsetzung des Labyrinths

In diesem und den zwei folgenden Kapiteln wird die physische Realisierung des Demonstratorsystems dokumentiert. Da sich viele Baugruppen gegenseitig bedingen und die Entwicklung nicht exakt in der hier verwendeten Reihenfolge stattgefunden hat, kann es dazu kommen, dass Parameter als Grundlage verwendet werden, deren Herkunft jedoch erst in einem späteren Abschnitt erklärt wird. Insgesamt werden drei Teilbereiche beschrieben: Das Labyrinth, der Roboter und die MATLAB Schnittstelle.

## 4.1. Rahmenkonstruktion

Das Labyrinth in dem sich der Roboter bewegen soll benötigt einen stabilen Untergrund. Nur so kann eine glatte Oberfläche gewährleistet werden, auf dem der Roboter mit $0{,}8\,\mathrm{mm}$ Bodenfreiheit nicht aufsetzt. Auch zum Transport ist eine solide Platte vielen Einzelsegmenten vorzuziehen. Eine Metallplatte bietet sich daher als Träger der Bodenplatinen an. Um Gewicht einzusparen und die Funktion der RFID Tags nicht zu beeinflussen, sollte das Metall nicht ferromagnetisch sein und möglichst große Aussparungen haben. Die im Leichtbau beliebten Metalle Titan und Magnesium sind zu teuer und neigen bei der Zerspanung zu Funkenbildung, müssen also unter besonderen Sicherheitsvorkehrungen bearbeitet werden. So verbleibt Aluminium in der Materialauswahl.

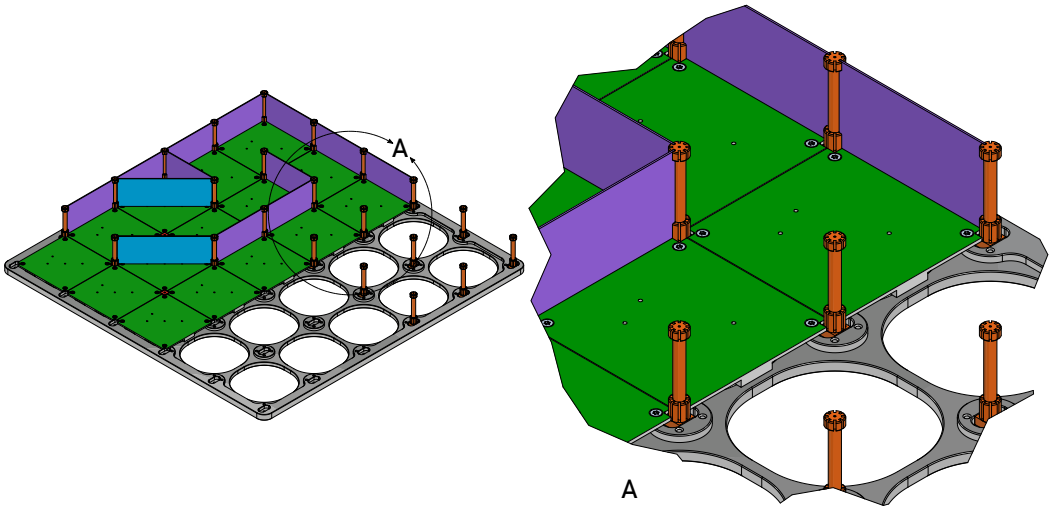


Abbildung 6.: Aufbau des Labyrinths mit Ausschnittsvergrößerung
Quelle: Eigene Abbildung

Wie erwähnt lassen sich RFID Tags nicht auslesen, wenn sie direkt auf einer leitfähigen Oberfläche aufliegen. Daher wird der Bereich hinter den Tags aus dem Tragrahmen ausgespart, wie es in Abbildung 6 dargestellt ist. Ebenfalls müssen die Steckverbinder und Bauteile auf den Platinen Platz finden, auch hierfür wird der nötige Raum durch Aussparungen geschaffen. Die Platinen liegen nur mit den Ecken auf dem Aluminium auf. Um den Rahmen gleichzeitig als elektrischen Masseverbinder zu nutzen, haben die Platinen entsprechende Kontaktflächen im Auflagebereich. Diese werden direkt durch die Montageschrauben auf den Rahmen gepresst und sorgen für eine sichere Verbindung.

Um Trennwände im Labyrinth aufstellen zu können, werden Halter benötigt. Die erste Lösung sieht vor, die Wände aus $2\,\mathrm{mm}$ starkem Verbundmaterial herzustellen, in deren Seitenkanten kleine Magnete eingelassen werden. An den Ecken aller Labyrinthfelder werden ferromagnetische Pfosten montiert, an denen sich die Wände in beliebigem Winkel befestigen lassen. Die filigrane Magnetmontage macht diese Idee jedoch unwirtschaftlich. Auch ist fraglich, ob sich mehrere Wände ohne nennenswerte gegenseitige Beeinflussung nebeneinander montieren lassen. Die einfacher umzusetzende Lösung sieht Wände mit einer Dicke unter $1\,\mathrm{mm}$ vor. Die Pfosten sind axial mit Nuten in allen gewünschten radialen Winkeln versehen. Um den Fräsaufwand zu verringern, sind die Nuten in der Mitte unterbrochen. Nur an den Enden wird die Wand an zwei Punkten gehalten. Die Stabilität leidet darunter nicht, der Fräsaufwand ist jedoch deutlich geringer als bei Nuten auf ganzer Länge. Im vergrößerten Ausschnitt in Abbildung 6 ist der Aufbau der Pfosten und das Zusammenspiel mit den Trennwänden deutlich zu erkennen. Die Pfosten werden in einer gefrästen Tasche arretiert und von der Rückseite mit einer Schraube fixiert. So muss zwar beim Aufstellen einer diagonalen Wand der Pfosten, der dann den Weg versperrt, mit Werkzeug entfernt

werden, die alternative Variante, die Pfosten magnetisch in Position zu halten, wäre jedoch deutlich aufwendiger und kann bei Bedarf jederzeit nachgerüstet werden.

## 4.2. Panelbefestigung

Die Grundfläche des Labyrinths muss vom Roboter gut befahrbar sein. Die Oberfläche muss eben und ohne hervorstehende Bauteile oder Schraubenköpfe sein. Die Reibung zwischen Fläche und Rädern muss hoch genug sein, dass der Roboter nicht rutscht. Da Elektronik in den Bodenflächen verbaut werden muss, liegt es nahe, Leiterplatten als Bauteilträger und mechanisches Bodenelement zu verwenden. Eine einzige Platine von der Größe des gesamten Labyrinths bietet zwar Einsparpotenzial bei den Bauteilen, ist jedoch in der Herstellung sehr kostenintensiv. Speziell das Auflöten der Bauteile mit Heißluft würde sich kompliziert gestalten, da in Teilabschnitten gelötet werden müsste und die umliegenden Bauteile stets thermisch mitbelastet würden. Um die genannten Nachteile zu umgehen, soll nur eine Feld entworfen werden, das dann gekachelt in der gewünschten Anzahl das Labyrinth bedeckt.

Um alle Felder mit Strom zu versorgen, müssen diese elektrisch verbunden werden. Eine elegante Möglichkeit ist die Stechverbinder so zu montieren, dass jedes Panel nach oben abgezogen und ausgetauscht werden kann. Dafür müssen die Stecker senkrecht zur Platine sitzen und weitere Adapterplatinen auf dem Rahmen, die die eigentliche Querverbindung gewährleisten. Das bedeutet jedoch erheblichen Mehraufwand, da doppelt so viele Stecker verwendet werden, wie bei einer direkten Panel zu Panel Verbindung. Auch müssen die Verbinderplatinen je nach verwendetem Steckverbinder sehr präzise montiert werden, um ein Verklemmen und Beschädigen zu verhindern. Mit günstigen Stiftleisten besteht ausreichend Spiel um dieses Problem zu umgehen, jedoch wird dann die Bodenplatte um die Steckerhöhe von etwa $15\,\mathrm{mm}$ dicker. Eine platzsparende Lösung mit spielausgleichenden Federkontakten wäre die Lösung zu den genannten Problemen. Die Kontaktstifte würden seitlich direkt auf das passende Gegenstück am nächsten Panel drücken und so Kontakt herstellen. Mit eingekerbten Kontaktflächen kann sogar ein Einrasteffekt erzielt werden. Nachteil dieser Lösung ist, dass die Federkontakte sehr teuer sind. Der Hersteller MillMax bietet passende Produkte an, die Preise belaufen sich jedoch auf etwa 1 € pro Kontakt, was je nach Pinkonfiguration in Summe etwa 500 € bis 600 € ergibt.

Aus genannten Gründen ist eine direkte Feld zu Feld Verbindung mit Stiftleisten am sinnvollsten. Zum Austausch eines Panels müssen zwar alle Felder zu einer Seite demontiert werden, jedoch ist nicht mit einem häufigen Feldwechsel zu rechnen. Direkte Verbinder sind günstig und von allen Varianten mit dem geringsten Arbeitsaufwand montierbar.

## 4.3. Adressierung

Jedes Feld braucht eine eigene Adresse, um dem Roboter Rückmeldung über seine Absolutposition zu geben. Die erste Version soll mit der physischen Anordnung der Felder arbeiten. Dabei erkennt jedes Feld, ob es einen Vorgänger und/oder Nachfolger hat. Das Feld ohne Vorgänger in X und Y Richtung ist automatisch Feld 1 mit der Adresse (0,0). Es signalisiert seinem Nachfolger in X Richtung, dass es seine Y-Adresse empfangen soll und sendet dann über den globalen UART Bus eine Zwei. Alle Felder können auf diesen Bus zugreifen, jedoch akzeptieren nur die vom Vorgänger angesprochenen Felder die Datenpakete. Dieser Ablauf wiederholt sich, bis ein Feld ohne Nachfolger erreicht wird. Das letzte Feld in der Reihe signalisiert mit einem speziellen Befehl dem ersten Feld, welche Adresse in X Richtung es hat, wie lang also die Reihe in X-Richtung ist. Daraufhin löst das Startfeld nacheinander für jedes Feld der ersten Reihe diesen Enumerationsprozess aus, woraufhin jedes dieser Felder seine Nachfolger in Y-Richtung abfragt. Auf diese Weise bekommen alle Felder dieser Reihen die X-Position des ersten Feldes zugewiesen. Am Ende kennt das Startfeld die Maße des Labyrinths. Falls Felder defekt sind und im Enumerationsprozess nicht auf Anfrage reagiert haben, kann das Startfeld den größtmöglichen rechteckigen Bereich bestimmen und signalisiert dann allen Feldern die Absolutmaße des Labyrinths. Felder, die aufgrund einer Begrenzung außerhalb des validen Bereiches liegen, können sich deaktivieren oder mittels LED ihren Zustand signalisieren. So fällt direkt auf, wenn Felder defekt sind.

Im Test konnte die Funktionsfähigkeit dieses Verfahrens zwar gezeigt werden, die Zuverlässigkeit war jedoch nicht ausreichend. Da auch alle Felder aufeinander angewiesen sind, gestaltet sich das debuggen als kompliziert. Die Einschaltzeitpunkte müssen synchronisiert werden und für jede Programmänderung müssen alle Felder neu beschrieben werden, was bei dieser komplexen Lösung sehr häufig vorkommt. Statt der vollautomatischen Enumeration ist auch eine statische Adressierung denkbar, da die Panele, wie bereits festgelegt, direkt miteinander verbunden werden und nicht einfach tauschbar sind. Eine häufige Positionsänderung ist daher nicht zu befürchten. Die Adresse ließe sich mit durchtrennbaren Leiterbahnen, Lötbrücken, oder Kodierschaltern realisieren. Zur besseren Ablesbarkeit und Handhabbarkeit eignen sich dezimale Kodierschalter besonders, da diese nicht binär abgelesen werden müssen und ohne Spezialwerkzeug, wie einer Lötstation, einstellbar sind. Allerdings ist dies von allen Lösungen die teuerste. Der schlussendlich implementierte Enumerationsablauf ist im Abschnitt 4.5 (Firmware) ausführlich beschrieben.

## 4.4. Platinenlayout

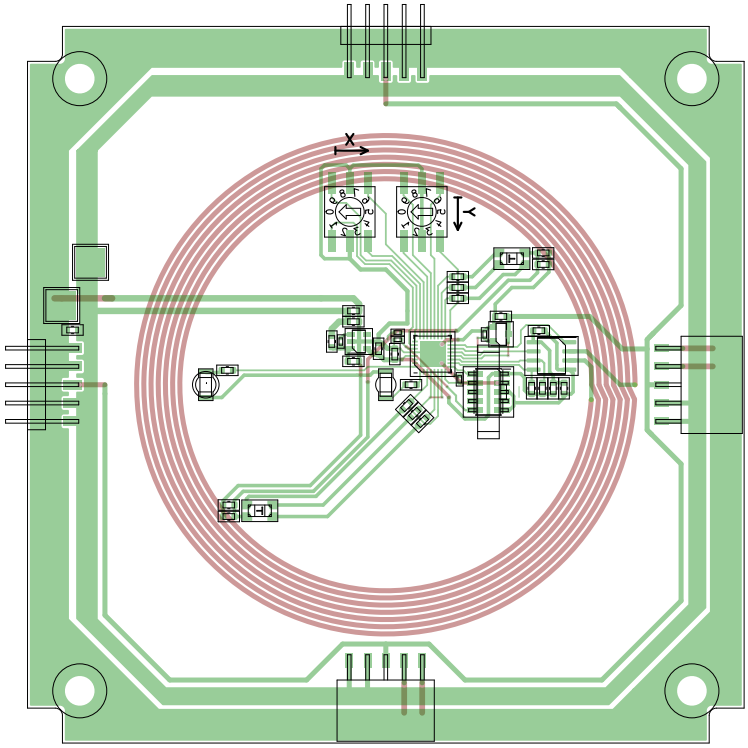


Abbildung 7.: Platinenlayout eines Bodenpanels
Quelle: Eigene Abbildung

Abbildung 7 zeigt das Bodenpanel Layout. Zu sehen sind zunächst die Montagebohrungen, die auf $6{,}3$ mm Durchmesser gesenkt werden, das passende Maß für M3 Senkkopfschrauben. Die eckigen Aussparungen bilden, wenn die Panele gekachelt werden, den Freiraum für die Pfosten, die die Wände fixieren. Umlaufend ist ein Massering zu sehen, welcher im Bereich um die Bohrungen vom Lötstopplack freigestellt ist, um elektrischen Kontakt zum Rahmen zu ermöglichen. Weiter innen ist ein Leiterring für die positive Spannungsschiene platziert. Zentriert an jeder Seite sind die Steckverbinder zwischen den Panelen zu finden. Im Layout der gefertigten Platinen ist die Pinreihenfolge von zwei der vier Steckverbinder invertiert, sodass beim verbinden die Betriebsspannung kurzgeschlossen wird. Dieser Fehler wird im Prototyp durch direkte Verdrahtung statt durch Steckverbinder gelöst. Abhilfe schafft hierbei eine symmetrische Pinbelegung, sodass die Pineihenfolge irrelevant wird. Abbildung 7 zeigt bereits eine korrigierte Version. In Rot ist die Spule für den RFID IC zu sehen, der rechts platziert ist. Im oberen Bereich der Spule sind die Kodierschalter für die Feldadresse in X und Y Richtung montiert. In der Mitte des Panels und im linken Bereich auf der waagerechten Mittelachse sind die Infrarot LEDs zur Orientierung des Roboters verortet, links unten und rechts oben innerhalb der Spule die RGB LEDs zur Visualisierung des Feldinhaltes. Die restlichen Bauteile sind hauptsächlich der Mikrocontroller, der Spannungsregler zur Stromversorgung und ein MOSFET als Leitungstreiber für

den UART-Bus. Der Mikrocontroller wurde nach dem günstigsten Preis ausgewählt, musste aber mit mindestens $5\,\text{kB}$ RAM und $45\,\text{kB}$ ROM ausgestattet sein, um die RFID-Tag Bibliothek zuzüglich eigenem Programmcode aufnehmen zu können. Außerdem sind zwei Timer für unterschiedliche Frequenzen der Marker LEDs und weitere Timer mit insgesamt sechs Timerkanälen für die beiden RGB LEDs nötig. Ein UART Modul ist in jedem STM32 Mikrocontroller zu finden und muss daher nicht als eigene Anforderung aufgeführt werden.

Anhand der geringen Komplexität des Schaltplanes und der vergleichsweise hohen Stückzahl der Platinen steht direkt fest, dass nur zwei Lagen verwendet werden dürfen, um die Kosten gering zu halten. Die RFID Spule orientiert sich im Durchmesser an der Robotergröße und damit den Abmaßen der Leser-Spule. Die LEDs sind alle innerhalb der Spule platziert, um die Leitungen kurz und das Abstrahlpotenzial der PWM Signale gering zu halten. Während die RGB LEDs einen weiten Abstrahlwinkel für gute Sichtbarkeit haben, strahlen die Infrarot LEDs nur mit $15°$ ab, sodass die Quelle präziser lokalisiert werden kann. Alle hier verwendeten LEDs haben ein reverse mount Gehäuse, werden also auf der Rückseite montiert und leuchten durch eine Bohrung Richtung Oberseite.

## 4.5. Firmware

Nach Einschalten der Betriebsspannung wird zunächst die Enumeration kontrolliert, die von der Adressierung durch die Kodierschalter vorgegeben wurde. Das Feld mit der eingestellten Adresse (0,0) übernimmt die Organisation des Prozesses und zeigt dies mit einer violetten LED Farbe an. Alle anderen Felder leuchten in gedimmtem Rot. Das erste Feld fragt nun der Reihe nach alle möglichen Feldkoordinaten ab und wartet auf eine Antwort. Zunächst wird die Reihe $Y = 0$ abgearbeitet. Sobald eine X Position nicht antwortet, steht die Anzahl erreichbarer Felder fest und die nächste Y Reihe wird getestet. Am Schluss wird das größtmögliche rechteckige Feld berechnet und die Maße an alle Felder via UART übertragen. Nun wird für eine Sekunde optisch angezeigt, welchen Status jedes Feld hat. Grün steht für jedes Feld, das in der aktiven Fläche liegt und helles Rot sind ansprechbare Felder außerhalb des aktiven Bereiches. Alle Felder die noch immer gedimmtes Rot zeigen, waren im Enumerationsprozess nicht ansprechbar, oder haben die Feldgröße nicht empfangen. Abschließend wird der RFID Tag mit der Feldposition und Labyrinthgröße beschrieben. Alle Feldinhalte werden zurückgesetzt. Lese- und Schreibzugriffe von außen auf den RFID Tag werden registriert und als Folge dessen der Inhalt auf Validität kontrolliert und angepasst. So werden die Feldparameter immer vom Panelcontroller eingetragen und können über die drahtlose Schnittstelle nicht verändert werden. Außerdem wird,

falls ein Konsumgut auf dem Feld platziert wurde, dessen Menge größer als Null ist, der hinterlegte Farbwert auf die RGB LEDs übertragen.

# 5. Umsetzung des Roboters

## 5.1. Räder

Da die Räder des Roboters platzoptimiert und an das Getriebe angebunden sein müssen, kommt nur eine Maßanfertigung in Frage. Das Rad selbst kann 3D gedruckt werden und um Rutschen auf den Bodenpanelen zu vermeiden wird ein Dichtungsring als Reifen aufgezogen. Die Anbindung an das Getriebe soll so wenig Manipulationen am Zahnrad benötigen, wie möglich, da der Aufwand sehr hoch ist. Eine Vergrößerung der Bohrung, um Lager einzupressen, oder axiale Bohrungen für Splinte, entfallen demnach. Die Verwendung des Zahnrades als Gleitlager erscheint, unter Beachtung der geringen Gewichtskraft und der niedrigen Drehzahlen, ausreichend. Es bleibt folglich nur noch die Befestigung des Rades am Zahnrad. In einer frühen Testversion sind zwei Schrauben als Passfederersatz in Bohrungen entlang der Kontaktfläche zwischen beiden Bauteilen geschraubt, wie Abbildung 8 darstellt.

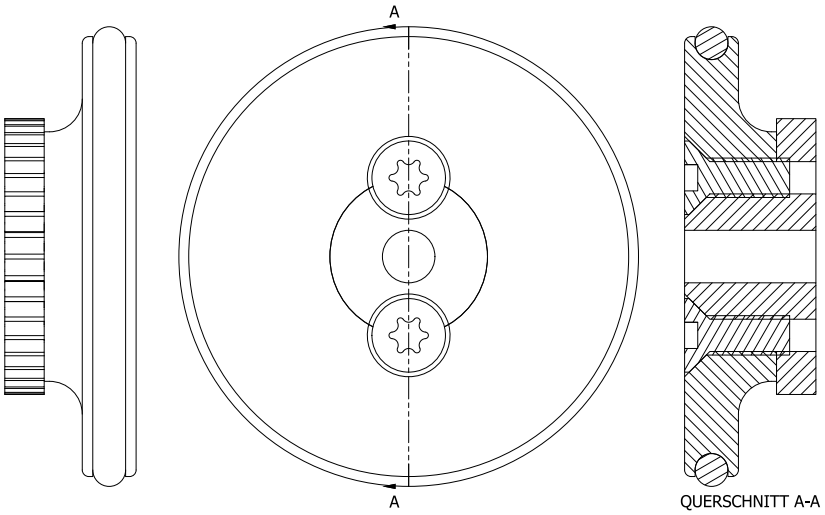


Abbildung 8.: Mit Schrauben fixiertes Rad
Quelle: Eigene Abbildung

Der Vorteil einer sehr stabilen Verbindung wird durch die Nachteile aufgehoben. Nicht nur sorgt die Toleranz des per Hand geschnittenen Gewindes dafür, dass eine koaxiale Montage beider Bauteile kaum möglich ist, das Zahnrad wird auch verzogen. Durch die Kraft, die von den Schrauben und den Kegelsenkungen in das Zahnrad eingebracht

werden, verzieht es sich zu einer leicht ovalen Form, was zu Verklemmungen mit dem Ritzel führt.

Stattdessen wird die herstellungsbedingt leicht konische Hülse am Zahnrad verwendet, um eine Presspassung mit dem 3D gedruckten Rad zu realisieren, wie in Abbildung 9 dargestellt. Im Praxistest hat sich die Verbindung als sehr stabil erwiesen. Ein Verrutschen des Rades auf dem Zahnrad ist auszuschließen.

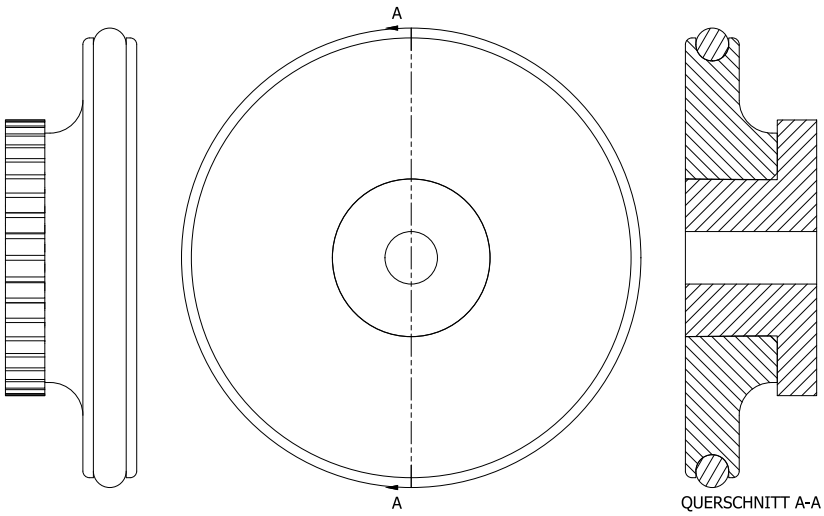


Abbildung 9.: Durch Presspassung fixiertes Rad
Quelle: Eigene Abbildung

## 5.2. Motoren und Chassis

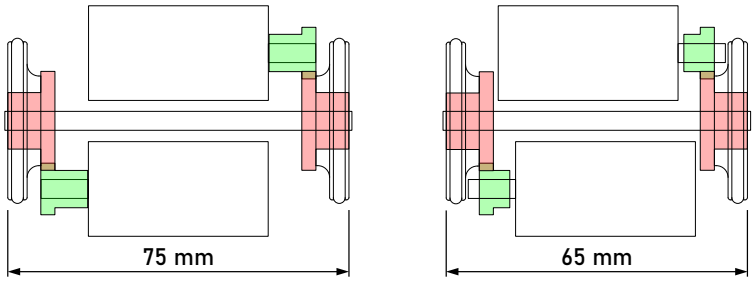


Abbildung 10.: Platzoptimierung des Antriebes
Quelle: Eigene Abbildung

Der Antrieb ist das platzintensivste Modul des Roboters und kann, wie in Abbildung 10 zu sehen, optimiert werden. Die Hülse des Ritzels, die als verlängertes Gleitlager und als Abstandhalter dient, wird hierbei soweit eingekürzt, dass das Rad zur Motorwelle noch einen Millimeter Abstand hat. Außerdem kann der Motor für die Gegenseite auch bis auf einen Millimeter Abstand zum Rad verlegt werden. Die eingesparte Breite beträgt somit $10\,\mathrm{mm}$.

Für das Chassis hat sich nach mehreren Ideen die folgende als beste erwiesen. Abbildung 11 zeigt das Chassis und den Antriebsstrang mit einem waagerechten Teilschnitt auf Höhe der Achse. In blau zu sehen sind die Motoren, die je mit einer Flanschplatte verbunden sind. Diese dienen auch der Befestigung des Unterbaus, bestehend aus zwei Platinen und den Puffern des Kollisionssensors. Die Flanschplatten werden mit dem Achsblock, der rot dargestellt ist, verschraubt. In Türkis markiert, durchzieht die Achse alle Bauteile und sorgt für korrekte Ausrichtung, sowie Stabilität. Die Räder sind werden auf der Achse durch Sicherungsringe befestigt, können sich aber frei drehen. Die Zahnräder aus POM dienen dabei als Gleitlager.

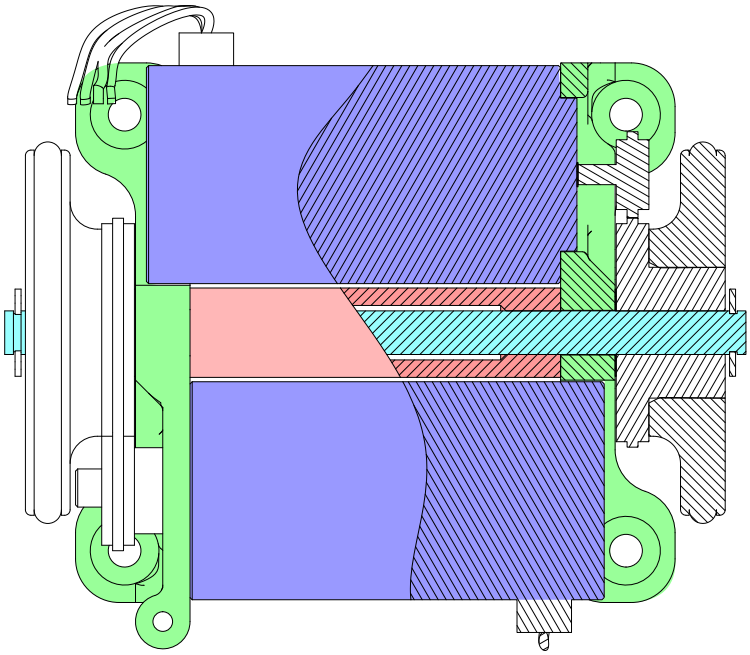

Abbildung 11.: Chassis in Draufsicht, waagerechter Teilschnitt durch Achse
Quelle: Eigene Abbildung

Abbildung 11 zeigt die letzte von vielen Iterationen, die primär für die Fertigung im FDM-Druck Optimiert ist. Eine sichtbare Anpassung ist zum Beispiel die Aufweitung der Achsbohrung im Achsblock, die nur im 3D Druck Verfahren problemlos fertigbar ist. Da die Bohrung nach dem Druckvorgang auf ihr Endmaß aufgebohrt werden muss und ein spielfreier Sitz gewünscht ist, lässt sich die Achse leichter einführen, wenn sie nicht auf der gesamten Bohrungslänge klemmt. So berührt sie innerhalb des Achsblocks kein Material und wird an den äußeren Punkten spielfrei fixiert. Die Achse selbst ist an beiden Enden mit Nuten ausgestattet. Diese werden mit einer rotierenden Trennscheibe in die ebenfalls rotierende Achse eingeschliffen. Passende Sicherungsringe halten die Räder in Position.

Auf der Antriebsseite muss der Motorschaft mit dem Zahnrad spielfrei verbunden werden. Jedes Spiel im Antriebsstrang sorgt für einen Versatz beim manövrieren, der

nachträglich kompensiert werden muss. Es ist also vorteilhaft, wenn Toleranzen von Anfang an gering sind. Der Motorschaft hat eine Flachstelle, die formschlüssig mit dem entsprechenden Gegenstück interagieren kann. Das Zahnrad hat jedoch eine runde Bohrung. Eine kraftschlüssige Verbindung entfällt, da sich POM nur schwer kleben lässt [Kle] und sich die Durchmesser von Motorschaft und Zahnrad nicht nachträglich für eine Übermaßpassung verändern lassen. Das Durchbohren von Zahnrad- und Motorschaft und fixieren mittels Splint ist aufgrund der Bauteilgröße und Fragilität des Motors hinfällig. Jede Art von metallischen und ferromagnetischen Fremdkörpern die in den Motor gelangt, würde die Funktionsweise be- oder verhindern. Auch das seitliche Einschrauben eines Gewindestiftes (Madenschraube) in das Zahnrad entfällt aufgrund der Erfahrung mit den Rädern, da sich hierbei das Zahnrad verziehen würde.

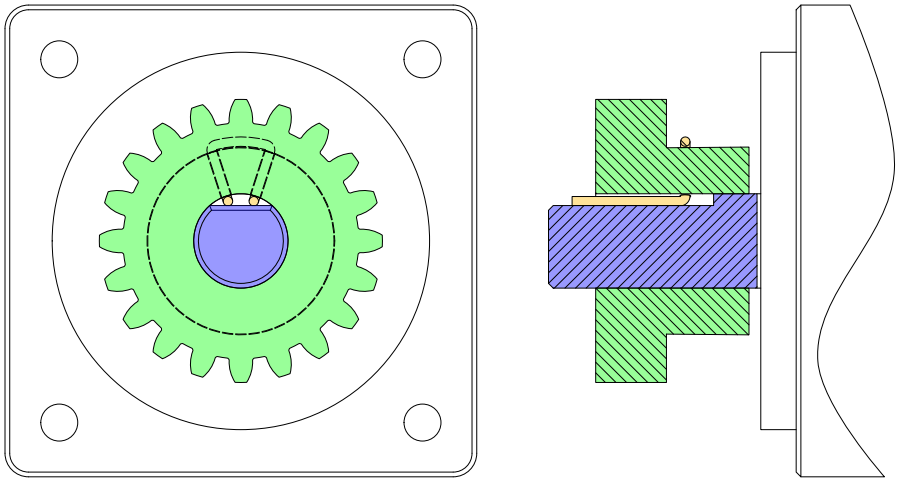

Abbildung 12.: Montage des Ritzels auf der Motorwelle, rechts Schnittansicht
Quelle: Eigene Abbildung

Die gewählte Lösung sieht vor, wie in Abbildung 12 dargestellt, radial zwei Löcher mit $30°$ bis $40°$ Abstand in die Zahnradhülse zu bohren und eine Schlaufe Kupferlitzen, hier Gelb markiert und vereinfacht dargestellt, mit den Enden voran von außen in die Bohrungen einzufädeln. Die offenen Enden werden vom Motor weg zeigend umgebogen. Die Litzen füllen nach der Montage des Zahnrades (Grün) auf der Motorwelle den Hohlraum, der durch die Flachstelle des Schaftes (Blau) entsteht. Zur Fixierung wird schnell aushärtendes Epoxidharz durch die verbleibenden Öffnungen in das Kupfergeflecht eingepresst. So kann der Klebstoff sich mit den Kupferlitzen verbinden und die Schlaufe verhindert ein Herausrutschen des Verbundmaterials. Das Epoxydharz geht keine Verbindung mit dem Zahnrad ein, weshalb dieses nach dem Montieren nicht mehr demontiert werden sollte.

## 5.3. Kollisionssensor

Wie im Unterabschnitt 3.2.3 angesprochen, sollen Kollisionen mit Hindernissen optomechanisch detektiert werden. Der Platz für die mechanischen, gefederten Puffer beschränkt sich auf den Platz zwischen den Sensorplatinen unterhalb der Motoren. Damit der Aufbau keine filigranen Spiralfedern benötigt, werden gerade Abschnitte aus Federstahl verwendet.

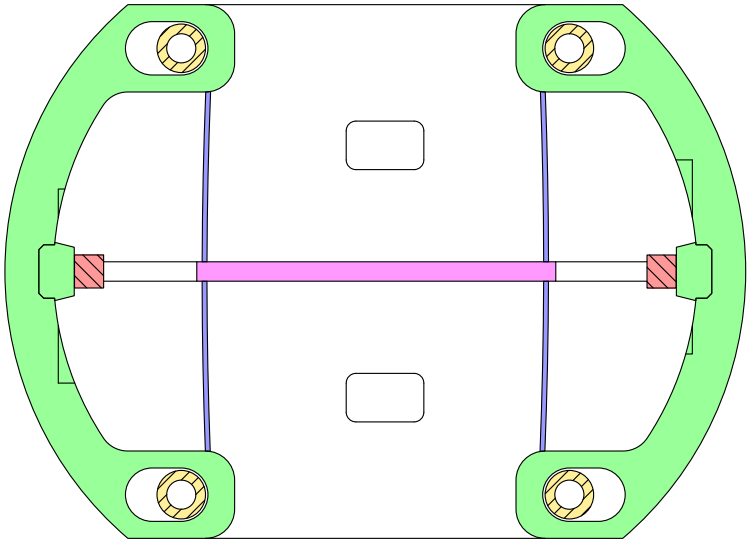

Abbildung 13.: Aufbau des Kollisionssensors mit Puffern und Detektoren
Quelle: Eigene Abbildung

Abbildung 13 zeigt in Grün die beiden Puffer, welche durch die Montage auf den Gewindehülsen (Gelb) einen festgelegten Bewegungsspielraum haben. Durch die beiden Federstahldrähte in Blau und den Abstandhalter in Violett dazwischen, werden beide Puffer nach außen gedrückt. Die Drahtkonstruktion ist dank geringer Vorspannung und Vermeidung filigraner Einzelteile sehr leicht zu montieren. Zur Detektion einer Kollision dienen je eine Reflexlichtschranke pro Puffer, hier Rot eingefärbt. Sie sind auf der Unterseite der Platine über dem Puffermechanismus montiert. Der kleine Vorsprung wird bei Kollisionen unter die Reflexlichtschranke verschoben und reflektiert eine detektierbare Menge Licht. Der Kollektorstrom durch den Fototransistor steigt dadurch an und erhöht den Spannungsabfall über einen Messwiderstand. Zur einfacheren Auswertung durch den Mikrocontroller wird das Signal per Schwellwertunterscheidung auf einen von zwei möglichen diskreten Spannungen gerundet. Ein zusätzlicher Transistor ermöglicht das Ein- und Ausschalten der Infrarot-LED. So kann der Stromverbrauch gesenkt werden, wenn keine Kollisionserkennung nötig ist, beispielsweise, wenn der Roboter sich nicht bewegt.

Die resultierende Betätigungskraft entspricht der Federkraft eines Federdrahtes. Abbildung 14 zeigt den äquivalenten Aufbau des Puffer-Feder-Systems. Die Komponenten

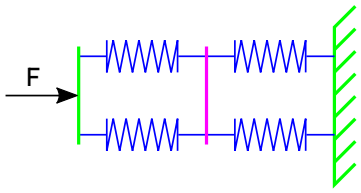


Abbildung 14.: Ersatzschaltbild des Kollisionsdetektors
Quelle: Eigene Abbildung

sind genauso eingefärbt wie bereits in Abbildung 13. Die angreifende Kraft ist mit einem Pfeil gekennzeichnet. Jeder der Federdrähte wird an den Enden von einem Puffer belastet. Da die Kraft von einem einzigen Bauteil ausgeht und sich gleichmäßig auf beide Enden aufteilt, können diese Punkte als verbunden angesehen werden. Es resultiert eine Parallelschaltung zweier Federn mit gleicher Federkonstante zwischen dem Puffer und dem Abstandhalter. Dieser verbindet beide Seiten miteinander und sorgt so für eine Reihenschaltung von je zwei parallelgeschalteten Federn mit gleicher Federkonstante, da beide Drähte mittig am Abstandhalter befestigt sind. Für die Parallelschaltung summieren sich die Federkonstanten [MBWa], während sie sich, aufgrund der Gleichheit beider Seiten in der Reihenschaltung wieder halbieren [MBWb]. In Summe ergibt sich daher die Federkraft eines Federdrahtes für den Fall, dass nur ein Puffer betätigt wird. Gleichzeitige Kraft auf den Puffer der Gegenseite verändert die Betätigungskraft. Wie hoch diese tatsächlich ist, wird experimentell mit den verfügbaren Federstahldrähten bestimmt werden. Als Stellschrauben stehen der Drahtdurchmesser, der Abstand der Auflagepunkte an den Puffern, sowie die Länge des Abstandhalters zur Verfügung.

## 5.4. Platinendesign

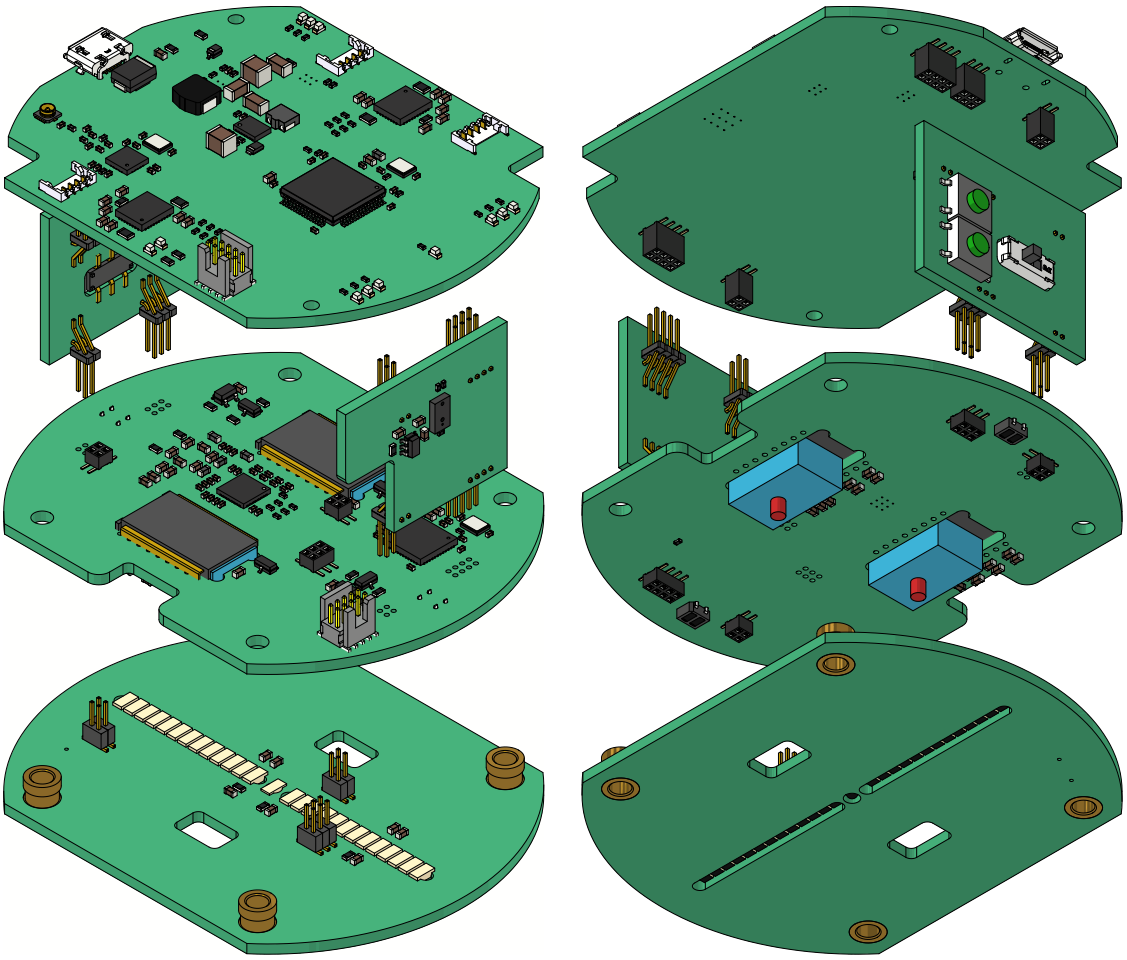

Abbildung 15.: Übersicht aller im Roboter verbauten Platinen
Quelle: Eigene Abbildung

Aus Platzgründen wird das elektronische System auf mehrere Leiterplatten verteilt. Im Detail auf drei Systemplatinen und zwei senkrechte Verbinderplatinen wie in Abbildung 15 zu sehen. Ganz oben befindet sich die Hauptplatine mit Funksystem, Powermanagement und Motortreibern. Die Verbinderboards leiten Signale und Stromversorgung nach unten zum Sensorcontroller weiter. Außerdem werden hier die Motoranschlüsse abgezweigt, sodass die Zuleitungen kurz gehalten werden können. Der Sensorcontroller wertet sämtliche Sensoren aus und stellt die Ergebnisse dem Hauptcontroller zur Verfügung. So wird dieser entlastet und es ist möglich, umfangreichere Auswertungsstrategien zu realisieren. Die Platine trägt neben dem Controller auch das RFID Lese- und Schreibsystem, sowie die Fototransistoren des Positionssensors, die Maussensoren und die Kollisionssensoren. Die unterste Platine dient als Halter für die Maussensoroptik und die Puffer zur Kollisionserkennung. Eingefräste Öffnungen bilden die Blende für die Fototransistoren. Auch die RFID Antenne ist hier aufgedruckt. Durch Auslagerung auf die unterste Platine steigt der Abstand zur Sensorik und rückt

so dicht, wie möglich, an die Bodenpanele heran. So sollen Störungen vorgebeugt und die Kopplung zur Empfangsspule maximiert werden. Die Senkrechtverbinder bieten Platz für den Hauptschalter und einen Laser Distanzsensor.

### 5.4.1. Sensorplatine

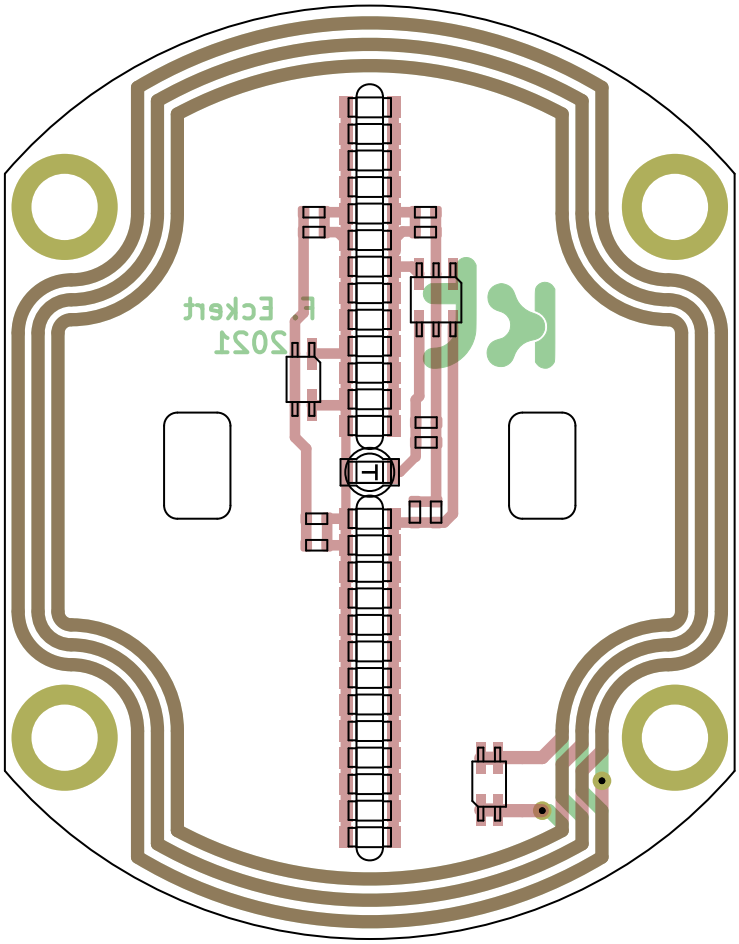


Abbildung 16.: Layout der Sensorplatine
Quelle: Eigene Abbildung

Die Sensorplatine, oder auch als Sensorboard bezeichnet, ist die unterste der Platinen im Roboter und ist in Abbildung 16 auf der linken Seite abgebildet. Entgegen der ersten Idee alle Bauteile auf die mittlere Platine zu montieren und auf der untersten lediglich die RFID Spule und Öffnungen für die optischen Sensoren zu platzieren, macht der hohe Platzbedarf der Maussensoren diesen Plan zunichte. Infolge dessen müssen die Fototransistoren zur Markerdetektion auf die untere Platine versetzt, was jedoch ein Problem mit sich bringt. In der ersten Version sitzen die Fototransistoren auf der Unterseite der mittleren Platine, sodass Infrarotstrahlung, die durch die Blende in der unteren Platine einfällt, detektiert werden kann. Auf der Unterseite der unteren Platine ist jedoch aufgrund der geringen Bodenfreiheit des Roboters kein Platz für

Bauteile, weswegen diese nun verkehrt herum auf der Oberseite montiert werden und durch die Platine blicken müssen. Für diesen Fall gibt es eine umgekehrt montierbare Bauform, in der Fototransistoren bei den betrachteten Händlern jedoch nicht verfügbar sind. So müssen Bauteile gefunden werden, die sich bei händischer Montage auch rückwärtig anbringen lassen. Hierfür sind umlaufende Kontaktflächen wichtig, die auch verkehrt herum auf der Platine aufliegen und lang genug sind, dass sie die Lötpads in dem vom Platinenfertiger vorgegebenen Abstand zur Bohrung erreichen. Nur so ist eine sichere Lötverbindung möglich. Außerdem muss die optische Abdeckung mit den mechanischen Gegebenheiten kompatibel sein. Alle Teile der Linse müssen in der Platinenaussparung versenkbar sein, damit die Kontakte auf der Lötfläche aufliegt. Für die äußeren Detektorgruppen fällt die Entscheidung auf Fototransistoren mit weitem Öffnungswinkel und hervorstehender Linse zur selbstständigen Arretierung. So wird der Erfassungsbereich gleichmäßig abgedeckt. Die Linsen sind jedoch eckig ausgeführt, weshalb eine runde Bohrung den Durchmesser der Diagonalen haben muss. Dann würden die Kontakte aber die Lötflächen nicht mehr überlappen.

Die Lösung ist, statt mehrerer runder Bohrungen eine lange Nut auszufräsen. Die abgerundeten Bereiche werden so neben den Sensorblock verschoben. Der Mittlere Sensor hat einen sehr geringen Öffnungswinkel um den Mittelpunkt so exakt wie möglich zu lokalisieren. Er lässt sich auch in einer einzelnen kleineren Bohrung montieren, da die Linse anders geformt ist.

### 5.4.2. Sensorcontroller

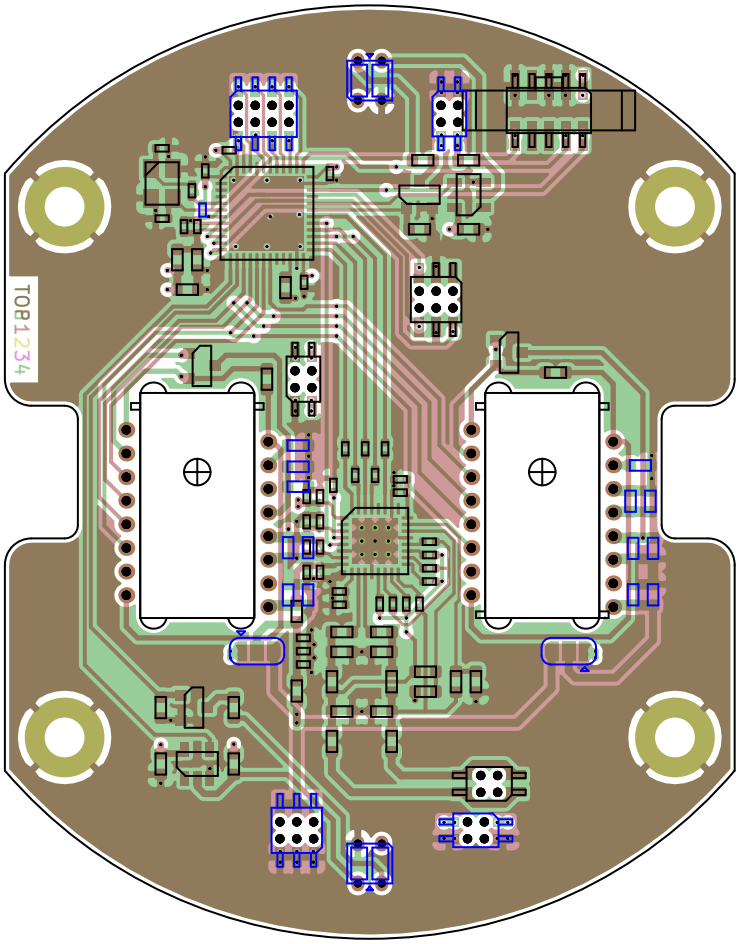


Abbildung 17.: Layout der Sensorcontrollerplatine
Quelle: Eigene Abbildung

Die Sensorcontrollerplatine, in Abbildung 17 zu sehen, hat zunächst die Besonderheit, dass links und rechts je eine Aussparung für die Zahnräder zu berücksichtigen ist. Außerdem steht die Position der Maussensoren fest. Die Messpunkte müssen symmetrisch zur Mittelachse und auf der Linie zwischen den Radmittelpunkten liegen. Der Abstand zum Mittelpunkt kann in einem kleinen Bereich variiert werden, größere Abstände sorgen jedoch für eine höhere Auflösung bei Kurvenfahrten, da die Bogenlänge des Messpunktes größer wird.

Auch die Reflexlichtschranken der Kollisionsdetektoren haben einen festen Montagebereich. Sie müssen auf der Platinenunterseite entlang der Mittelachse platziert werden. Weil diese Bauteile im bedrahteten Gehäuse vorliegen, muss der Abstand zur Platinenkante mit den senkrechten Verbinderplatinen abgestimmt werden, da diese genau Platz zwischen den Lötstellen finden muss. Alternativ könnte die Verbinderplatine ausgefräst werden, wodurch jedoch Platz auf den Verbindern verloren geht.

Die ideale Orientierung der Reflexlichtschranke wurde in einem Testaufbau bestimmt.

Experimentell ergab sich dabei, dass der Weg den ein Reflektor zurücklegen muss um den vollen Signalausschlag zu erreichen am kürzesten ist, wenn die gedachte Linie zwischen Sender und Empfänger rechtwinklig zur Bewegungsrichtung liegt, also beide gleichzeitig vom Reflektor verdeckt werden. Die nachgeschaltete Elektronik zur Bereinigung des Analogsignals wird in der Nähe der Lichtschranken platziert.

Die Steckverbinder zum Hauptcontroller werden durch die Position der Senkrechtverbinder festgelegt. Da sie den Motorblock vorn und hinten umschließen, ist es sinnvoll, den Mikrocontroller und den SWD Stecker zum programmieren in der Nähe der Steckverbinder zu platzieren.

Auf der vorderen Verbinderplatine befindet sich der Distanzsensor, welcher den vorwärtigen Fahrbereich abdeckt, auf der hinteren der Hauptschalter. Der Mikrocontroller muss folglich nur mit der vorderen Platine interagieren und wird dementsprechend auch in deren Nähe platziert. Eine Aussparung in der vorderen Verbinderplatine ermöglicht es, den SWD Stecker zu erreichen. Aufgrund der beengten Platzverhältnisse wird der 14 polige Stecker durch einen 10 poligen mit weniger Debugsignalen ersetzt. Der 14 polige Stecker des Verbindungskabels des Programmieradapters „STLinkV3Mini “, sowie die zwei äußeren Adern auf jeder Seite des Flachkabels müssen dazu entfernt und ein 10 poliger Steckverbinder angebracht werden.

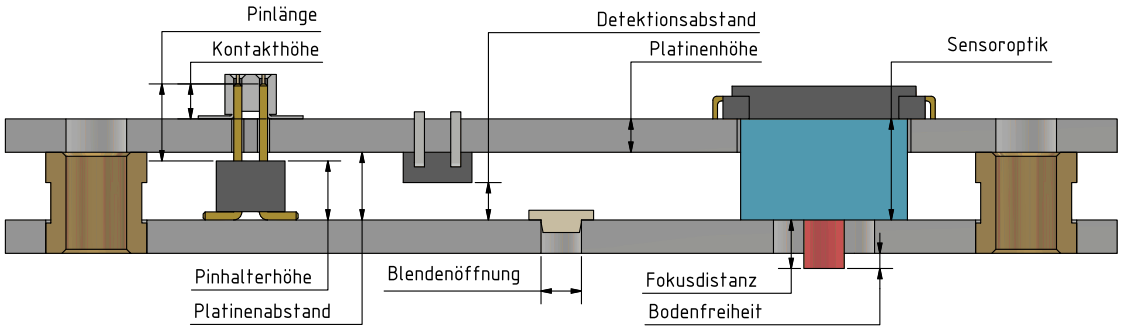


Abbildung 18.: Seitenansicht der Sensorplatinen mit wichtigen Maßen
Quelle: Eigene Abbildung

Abbildung 18 zeigt den Unterbau des Roboters, bestehend aus der Sensorplatine ganz unten und darüber dem Sensorcontroller. Der Abstand zwischen den Platinenoberseiten wird durch die Höhe der Maussensoroptik bestimmt und beträgt fünf Millimeter. Um trotz variabler Platinenstärke diesen Abstand zu gewährleisten, werden fünf Millimeter lange Gewindehülsen in die untere Platine eingesteckt und verlötet. Die obere Platine liegt dann auf der Hülse auf und wird von oben mit dem Chassis verschraubt. So stehen auf der Unterseite des Roboters keine Schraubenköpfe hervor.

Ebenfalls durch den Platinenabstand beeinflusst werden der Detektionsabstand des Kollisionssensors und die maximale Pinhalterhöhe der Board zu Board Steckverbin-

der. Als Detektionsabstand wird hier der Freiraum unterhalb der Reflexlichtschranke bezeichnet, in dem der Reflektor des Puffers Platz finden muss. Ein kleiner Abstand zwischen Puffer und Lichtschranke ist zum Überkoppeln des Lichts, sowie zur Prävention mechanischer Kollision notwendig. Die Buchsen auf der oberen Platine haben eine spezifizierte Kontakthöhe. Um sicheren Kontakt zu gewährleisten, müssen die Pins bis über diese Höhe in die Buchse hineinreichen. Die benötigte Pinlänge variiert folglich auch mit der Platinenstärke. Eine Montage der Buchsen auf der Unterseite der oberen Platine ist aufgrund des geringen Platinenabstandes nicht möglich. Die verbleibende Bodenfreiheit kann ebenfalls durch die Verwendung einer dünneren Platine erhöht werden. Der Abstand zwischen der Maussensoroptik und dem Boden, also die Fokusdistanz, ist dabei der limitierende Faktor.

Es stehen zwei Platinenfertiger zur Auswahl, deren Produktqualität bereits in vorherigen Projekten überzeugt hat, MultiCB und Beta LAYOUT. Beide fertigen in Deutschland, wodurch lange Wartezeiten und Zollgebühren entfallen. Neben kleinen Differenzen in den geforderten Abstandsmaßen ist der hauptsächliche Unterschied, dass Beta LAYOUT auch vierlagige Platinen mit einem Millimeter Dicke ohne Aufpreis anbietet [Bet]. MultiCB liefert zum Grundpreis nur Platinen mit $1{,}55\,\mathrm{mm}$ Nennstärke [Mul]. In beiden Fällen sorgt der Lötstopplack für etwa $0{,}1\,\mathrm{mm}$ Aufmaß. Da MultiCB jedoch auch die $2\,\mathrm{mm}$ starken Bodenpanele fertigen kann und ohne Aufpreis etwas kleinere Abstandsmaße und Leiterbahnbreiten unterstützt, fällt die Auswahl auf diesen Fertiger.

### 5.4.3. Hauptplatine

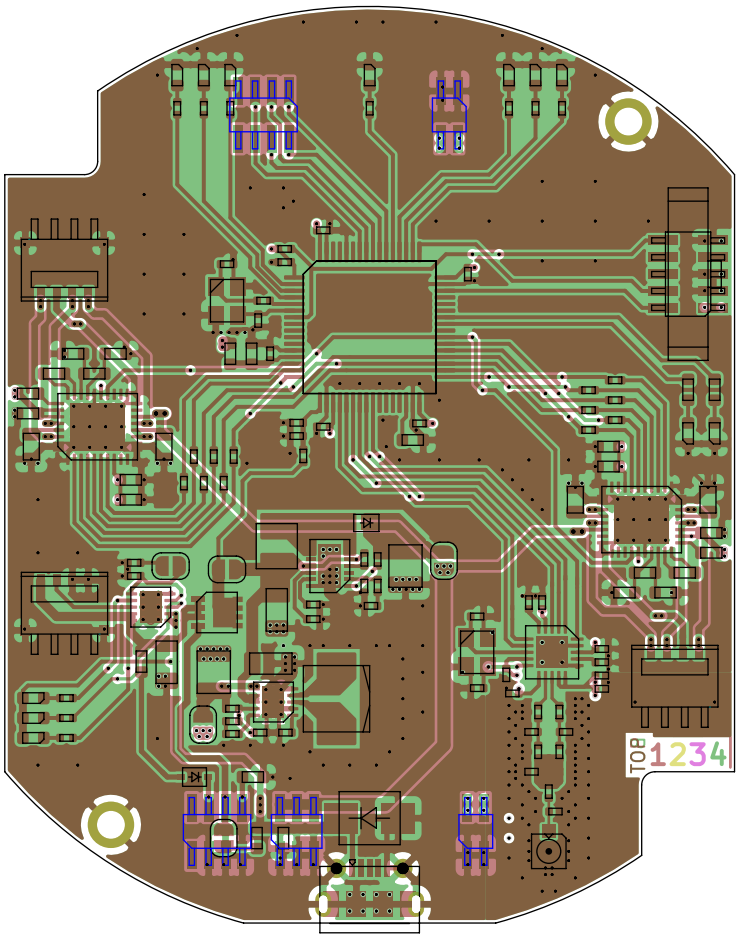


Abbildung 19.: Platinenlayout des Mainboards
Quelle: Eigene Abbildung

Da die Hauptplatine, in Abbildung 19 dargestellt, oben auf dem Roboter platziert ist und sich darüber lediglich der Akku befindet, sind weniger Abhängigkeiten vorhanden. Im Wesentlichen stehen die beiden Montagebohrungen, die Steckverbinder für die Verbinderplatinen und der USB Stecker zum Aufladen des Akkus fest. Die Platine liegt direkt auf den Motoren auf, was keine Bauteile auf der Unterseite in diesem Bereich zulässt. Damit die Fläche auf der Oberseite möglichst uneingeschränkt nutzbar ist, müssen die Verbinderplatinen mit SMD Steckverbindern angeschlossen werden. Diese sind so angeordnet, dass ein Vertauschen der Richtung oder Verwechseln der vorderen und hinteren Verbinderplatine nicht möglich ist oder zumindest beim Anschließen auffallen würde. Die Motorstecker und die Treiberschaltungen sind aufgrund der doppelten Ausführung punktsymmetrisch zum Mittelpunkt der Platine aufgebaut und angeordnet. Aussparungen in der Platine bieten den Motorleitungen Platz, ohne das Lichtraumprofil zu vergrößern. Der Ladestecker ist mittig an der hinteren Platinenkante platziert. Er ist ebenfalls leicht eingerückt, um die Außenmaße zu wahren. Da der Akku die meiste Fläche über der Platine verdeckt, ist der Anschlussstecker an der

Seite platziert, sodass die Leitung in einer Schleife um den Akku herum liegen kann. Dies minimiert Zugspannungen und erleichtert die Montage. Alle weiteren Bauteile wie das Powermanagement, bestehend aus dem Laderegler und zwei Schaltwandlern für 3,3V und 5V, sowie dem Mikrocontroller sind im Mittelteil der Platine unter dem Akku angeordnet. Auch die Funkschnittstelle mit Balun findet hier Platz.

### 5.4.4. Verbinderplatinen

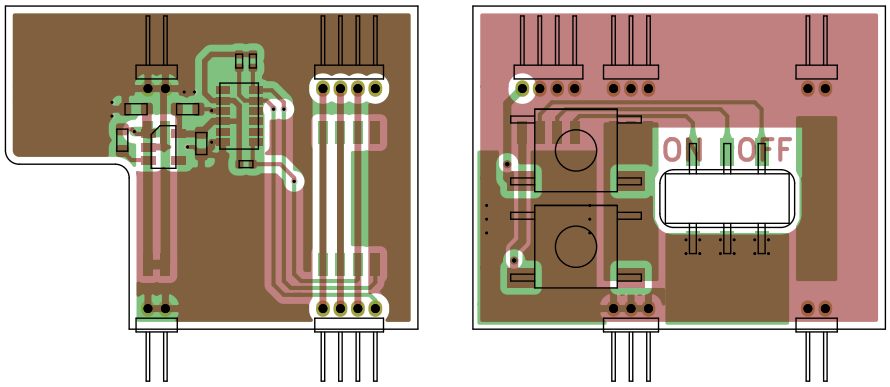


Abbildung 20.: Layout der vorderen (links) und hinteren (rechts) Verbinderplatine
Quelle: Eigene Abbildung

Die Verbinderplatinen sind die Brücke zwischen Sensorcontroller und Mainboard. Abbildung 20 zeigt auf der rechten Seite oben die vordere Verbinderplatine. Die Platinenaussparung schafft Platz für den SWD Stecker des Sensorcontrollerboards und der mittig oben platzierte Distanzsensor blickt, dank der senkrechten Platinenmontage direkt nach vorn. Er ist so platziert, dass er Erfassungskegel bei einer Feldlänge Abstand möglichst nicht den Boden erfasst, sondern hauptsächlich die Trennwand. Daneben ist die Spannungsversorgung des Sensors zu sehen. Auf dem hinteren Verbinderboard, in Abbildung 19 rechts unten zu sehen, ist der Hauptschalter montiert. Diese liegt in einem SMD Gehäuse vor, wird aber, um Platz zu sparen, rückseitig in einer Aussparung der Platine aufgelötet. So ragt er nicht aus dem Lichtraumprofil heraus. Daneben finden zwei Taster Platz, die für beliebige Zwecke verwendet werden können. Ursprünglich soll damit das Programmieren der Felder mittels Roboter ermöglicht werden. Aufgrund eines Planungsfehlers sind diese Schalter im Prototyp jedoch nicht mit dem Mikrocontroller verbunden, was sich auch nachträglich nicht ergänzen lässt. Auf diese zusätzliche Funktionalität muss daher verzichtet werden.

## 5.5. Firmware

Die folgenden zwei Kapitel beschreiben den generellen Programmablauf in den jeweiligen Modulen. Anschließend wird detailliert auf einige Programmelemente eingegangen.

### 5.5.1. Sensorcontroller

Der Sensorcontroller prüft in der Hauptschleife kontinuierlich Statusflags auf Handlungsbedarf ab. Ein Timer setzt alle $2$ ms ein Flag, das das Packen der aktuellen Sensormesswerte in eine Nachricht veranlasst, die dann dem DMA Controller zur Weiterleitung an die SPI Schnittstelle übergeben wird. Anschließend wird dem Hauptcontroller mit einem Ausgangspin signalisiert, dass Daten bereitstehen. Mit den PWM Kanälen des Timers wird zeitversetzt, aber mit gleicher Frequenz die Distanzmessung und die Kollisionsdetektion ausgelöst. Der ADC tastet mit $100$ kHz die Infrarotdetektoren ab, wobei für jeden der drei Sensoren $512$ Messwerte aufgenommen werden. Dies bedeutet, dass nur alle $5{,}12$ ms ein neuer Wert zur Verfügung steht, die Nachrichten werden jedoch häufiger gesendet, um bei Kollisionen schneller reagieren zu können. Die aktuellen Daten der Maussensoren werden direkt vor dem Packen jeder Nachricht ausgelesen. Neben den Bewegungsdaten liefern sie auch einen Index für die Oberflächenqualität, der Aufschluss darüber gibt, wie sicher die Detektion aktuell ist. Sinkt dieser Wert gegen Null, so ist keine Fläche im Fokusbereich. Der Roboter wurde dann angehoben, oder ist in einen Abgrund gefahren. Das Auslesen von Feldinhalten oder andere Interaktionen via RFID dauern etwa $100$ ms, weshalb es nur auf Anfrage durch den Hauptcontroller durchgeführt wird. Da die SPI Schnittstelle Lese- und Schreibzugriffe gleichzeitig erledigt, empfängt der Sensorcontroller beim Auslesen lassen der Sensordaten einen Befehl. Dieser wird nach Abschluss der Transaktion geprüft und, wenn valide, ausgeführt. Währenddessen werden keine Sensordaten verschickt. Die nächste Nachricht an den Hauptcontroller findet nach Beendigung der RFID Interaktion statt, hier werden auch die ausgelesenen Daten übermittelt.

### 5.5.2. Hauptcontroller

Der Hauptcontroller wartet in der Hauptschleife darauf, dass Sensordaten bereitstehen. Diese werden ausgelesen und zur internen Verwendung bereitgehalten. Gleichzeitig sind die eintreffenden Nachrichten der Taktgeber für viele weitere Prozesse, da diese auf die aktuellen Sensordaten angewiesen sind. Zum Beispiel werden nach einem Befehl ein Feld via RFID zu manipulieren, automatisch Lesebefehle nachgeschoben, um den neuen Feldzustand zu kennen. Auch der für die Navigation und

Bewegung verantwortliche Zustandsautomat wird hier aufgerufen. Zwei mal pro Sekunde werden Funknachrichten mit dem Status des Roboters gesendet, auf die die Funkbrücke in einem Zeitfenster mit einem neuen Befehl antworten kann. Diese Strategie ermöglicht, das Funkmodul in der Zeit zwischen Nachrichten in einen Ruhezustand zu versetzen und den Stromverbrauch währenddessen praktisch zu eliminieren. Der Befehl wird geprüft und ausgeführt, wenn er interpretiert werden kann. Wird ein Fahrbefehl gesendet, so wird ein Navigationsauftrag ausgelöst. Alle Vorbereitungen, wie die Berechnung der Beschleunigungsrampen und das Einstellen der Motortreiber werden erledigt und der Zustand der Finite State Machine (FSM) wird initialisiert. Anschließend wird die Handlung des gesetzten Zustandes ausgeführt. Alle weiteren Aktionen folgen schrittweise, wenn die Navigationsroutine nach Erhalt der nächsten Sensordaten wieder gestartet wird. Wird während einer Geradeausfahrt eine Kollision festgestellt, stoppt der Roboter und je nach Bewegungszustand kehrt er anschließend in den Ausgangszustand zurück, also auf das Feld, auf dem das Fahrmaneuver begonnen hat. Fahrbefehle auf ein anderes Feld werden erst ausgeführt, nachdem der Roboter sich ausgerichtet hat und die Position und Orientierung bekannt ist. Sollte eine Rückkehr zum Ausgangsfeld fehlschlagen, so wird im Status die Orientierung auf „nicht definiert “ gesetzt und es muss vom PC ein Ausrichtungsbefehl gegeben werden. Nach erfolgreicher Beendigung einer Bewegung wird des Feld ausgelesen und die Orientierung aktualisiert, sodass nach abgeschlossener Fahrt stets der Zustand des Roboters bekannt ist.

Neben dieser Übersicht des Ablaufs, soll im Folgenden noch detaillierter auf einige Firmwareabschnitte eingegangen werden, die von besonderer Bedeutung sind und deren Entwicklung die meiste Zeit in Anspruch nimmt.

### 5.5.3. Berechnung der Beschleunigungsrampen

Der Roboter kann nicht sprunghaft beschleunigt werden, da er massebehaftet ist. Damit die Räder beim Anfahren und Anhalten nicht ins Rutschen kommen, soll ein weicher Bewegungsübergang mit einer konstanten Beschleunigung implementiert werden. Besondere Problematik in diesem Fall ist, dass Schrittmotoren zeit- und winkeldiskret angesteuert werden müssen, also muss auch die Rampe diskretisiert werden. Als Grundlage zur Berechnung hierfür dient die Application Note AVR446 von Microchip (ehemals ATMEL) [Mic, S.5]. Ziel ist es, Gleichungen zu finden, mit denen aus einer gegebenen Beschleunigung und Zielgeschwindigkeit die Schrittanzahl pro Rampe und die Verzögerungszeit für den ersten Schritt berechnet werden können. Folgende Größen sind von Relevanz:

$\omega, \dot{\omega}$ - Winkelgeschwindigkeit, Winkelbeschleunigung
$t_{Timer}$ - elementare Verzögerungszeit, Kehrwert der Timerfrequenz $f_{Timer}$
$alpha$ - elementarer Drehwinkel, Winkelauflösung des Antriebs
$n \in [1, N]$ - Schrittindex
$C_n$ - Anzahl von elementaren Verzögerungen für n-ten Schritt

Aus dem Appnote werden folgende Gleichungen verwendet:

$$\omega = \frac{\alpha}{\delta t},\ mit\ \delta t = t_{Timer} \cdot C_n \tag{5.1}$$

$$C_n = C_0 \cdot (\sqrt{n-1} - \sqrt{n}),\ mit\ n = [1, N] \tag{5.2}$$

$$C_0 = \frac{1}{t_{Timer}} \cdot \sqrt{\frac{2\alpha}{\dot{\omega}}} \tag{5.3}$$

Da $t_{Timer}$ die kleinste Verzögerung ist, die der Timer realisieren kann, ist es äquivalent mit dem Kehrwert der Timerfrequenz, die $108\,\text{MHz}$ beträgt. $\alpha$ ist der kleinste Schrittwinkel, den das Rad ausführen kann. Eine Drehung hat $200$ Schritte, jeder Schritt $256$ Mikroschritte und dank der Getriebeuntersetzung dreht sich das Rad ein halbes Mal pro Ritzeldrehung. Insgesamt ist pro Mikroschritt also eine Raddrehung von $61{,}35\,\mu\text{rad}$ zu erwarten. Diese beiden Parameter sind demnach Konstanten.

Gleichung 5.3 kann umgeschrieben werden zu Gleichung 5.4

$$C_0 = f_{Timer} \cdot \sqrt{\frac{2\alpha}{\dot{\omega}}} \tag{5.4}$$

Somit lässt sich die Anzahl $C_0$ von Verzögerungseinheiten $t_t$ für eine Rampe mit der Beschleunigung $\dot{\omega}$ bestimmen. Um die Anzahl N der Rampenschritte berechnen zu können, wird zunächst für $C_n$ die Schrittanzahl des letzten Rampenschrittes $C_N$ eingesetzt und Gleichung 5.1 zu Gleichung 5.5 umgeschrieben.

$$\omega = \frac{\alpha}{t_{Timer} \cdot C_N} = \frac{\alpha \cdot f_{Timer}}{C_N} \tag{5.5}$$

Nach $C_N$ umgestellt und in Gleichung 5.2 eingesetzt, folgt daraus Gleichung 5.6.

$$\sqrt{N-1} - \sqrt{N} = \frac{\alpha \cdot f_{Timer}}{\omega \cdot C_0} \tag{5.6}$$

Nach der Schrittanzahl N aufgelöst, ergibt sich Gleichung 5.7.

$$N = \frac{(X^2-1)^2}{4X^2},\ mit\ X = \frac{\alpha \cdot f_{Timer}}{\omega \cdot C_0} \tag{5.7}$$

Jeder Verzögerungsschritt der Beschleunigungsrampe lässt sich mit Gleichung 5.2 berechnen. Der Vorteil besteht darin, dass sich auch Abwärtsrampen mit geringen Programmänderungen errechnen lassen, indem n von N bis 1 läuft. Der Nachteil ist, dass zwei Quadratwurzeln pro Schritt zu lösen sind, was viel Zeit in Anspruch nimmt. Hierfür schlägt die Appnote Gleichung 5.8 vor, welche durch Taylorreihenentwicklung aus Gleichung 5.2 hervorgeht.

$$C_n = C_{n-1} - \frac{2 \cdot C_{n-1}}{4n+1} \tag{5.8}$$

Für $n = 1$ besteht jedoch eine Abweichung vom tatsächlichen Wert, der durch Multiplikation von $C_{(n=1)-1} = C_0$ mit $0{,}676$ korrigiert werden kann. Die Berechnung geht schneller, da keine Wurzeln mehr enthalten sind. Nachteil hiervon ist, dass die Reihenfolge nur aufwärts gehen kann, da immer der Vorgängerschritt einbezogen ist. In der Implementierung macht diese Lösung die Rechnung ebenfalls umständlicher, da neben der Schrittanzahl noch das letzte $C_n$ gespeichert werden muss. Auch die Sonderstellung von $C_0$ müsste berücksichtigt werden. Aus diesem Grund wird eine hybride Methode angewendet, bei der der erste Schritt stets direkt mit Gleichung 5.2 berechnet wird und alle folgenden mit Gleichung 5.8. So muss der Einsprungpunkt in der Rampe nicht separat gespeichert werden und die Sonderstellung von $C_0$ für $n = 1$ entfällt, da wegen der erzwungenen Aufwärtsrichtung der Wert von $C_1$ immer mit Gleichung 5.2 direkt berechnet wird.

Die Implementierung auf dem Mikrocontroller lässt sich aufgrund der kurzen Pulsweiten nicht durch Berechnung des nächsten Schrittintervalls im vorhergehenden Schrittintervall lösen. Bei hohen Schrittfrequenzen kann es passieren, dass die Berechnung der nächsten Verzögerungszeit nicht rechtzeitig abgeschlossen werden kann, sodass die vorherige Schrittzeit vom Timermodul automatisch wiederholt wird. So werden mehr Schritte ausgeführt, als gewünscht. Außerdem ist der Prozessor in dieser Situation vollkommen mit Berechnungen ausgelastet, sodass keine Kapazitäten für andere Prozesse, wie die Auswertung von Sensoren, mehr bleibt. Eine zuverlässige Navigation wird so unmöglich. Um diese Probleme zu umgehen, muss ein Segment der Rampe im Voraus berechnet und mittels Direct Memory Access (DMA) ohne Prozessorbeteiligung an das Timermodul weitergeleitet werden. Ist das Segment zur Hälfte abgefahren, wird ein Interrupt ausgelöst. Der abgearbeitete Bereich wird nun mit dem

Abschnitt der Rampe gefüllt, der nahtlos an den Teil anknüpft, der gerade aus der zweiten Speicherhälfte abgearbeitet wird. Ist das Speicherende erreicht, wird wieder ein Interrupt ausgelöst. Der Speicher wird wieder von Vorn ausgelesen und die obere Hälfte erhält neue Daten. So kann ein kontinuierlicher Betrieb stattfinden, da der Overhead der Interruptannahme nur einmal anfällt, statt nach jedem Puls.

### 5.5.4. Ablauf einer Fahrbewegung

Der gesamte Fahrprozess kann in die zwei Abschnitte Vorbereiten und Ausführen unterteilt werden, wobei zur Ausführung Beschleunigen, Fahren mit Zielgeschwindigkeit und Abbremsen gehören. Die Vorbereitung besteht in der Bestimmung der Motordrehrichtungen. Positive Distanzen werden als Vorwärtsbewegung, negative als Rückwärtsbewegung interpretiert. Drehungen um einen positiven Winkel sind Rechtsdrehungen, um einen negativen Winkel Linksdrehungen. Außerdem müssen die Motoren bei einer Drehung gegenläufig rotieren. Die Tatsache, dass die Motorwellen in eingebautem Zustand in entgegengesetzte Richtungen zeigen, muss ebenfalls berücksichtigt werden. Die gewünschte Distanz wird mithilfe der definierten Fahrwerkparameter wie Raddurchmesser, Radabstand, Untersetzung des Getriebes, Schritte des Motors pro Umdrehung und Anzahl der Mikroschritte pro Vollschritt in die Anzahl der zu fahrenden Schritte umgerechnet. Ist diese Anzahl von Schritten kleiner oder genauso lang wie zwei Beschleunigungsrampen, so passt das gesamte Fahrmanöver in den vorgesehenen DMA-Buffer. Es wird eine Aufwärtsrampe berechnet, und eine Kopie davon gespiegelt angehängt. Dem DMA Controller wird später nur die relevante Bufferlänge übergeben. Passt das Manöver nicht komplett in den Speicher, so wird zunächst nur ein Teil berechnet. Nach dieser Vorbereitung beginnt die Ausführung. Dabei wird die erste Verzögerungszeit an das Timermodul übergeben und anschließend der Timer im DMA Modus gestartet. Der Timer fordert dann nach Ablauf der ersten Pulsdauer die nächste Verzögerungszeit vom DMA Controller an. Dieser löst jeweils einen Interrupt aus, wenn die erste und zweite Hälfte der verfügbaren Pulszeiten an den Timer übertragen wurden. Diese werden im Hardware Abstraction Layer (HAL) von ST Microelectronics als Half-Complete Interrupt und Complete Interrupt bezeichnet. Der Vorteil dieses Systems ist, dass bei Erreichen der zweiten Speicherhälfte die erste wieder aufgefüllt werden kann und umgekehrt, während der Timer zeitgleich mit neuen Verzögerungszeiten versorgt wird. So ist eine unterbrechungsfreie Wertübertragung an den Timer möglich.

Eine Besonderheit bei der Berechnung auf diesem Mikrocontroller ist, dass die Timer eine Zählregisterlänge von $16$ bit haben. Bei langsamen Schrittfrequenzen, also zu Beginn einer Aufwärts- und zum Ende einer Abwärtsrampe können Verzögerungszeiten größer als $2^{16} - 1 = 0xFFFF$ auftreten. Der Präfix $0x$ markiert hier eine Zahl im

Hexadezimalformat. Der Wert läuft dann über und erzeugt effektiv eine zu hohe Schrittfrequenz, die zu ruckartigem Fahrverhalten führen. Werte über $2^{16} - 1$ müssen deshalb auf diesen Wert begrenzt werden. Die Beschleunigung folgt dann zwar nicht exakt einem linearen Verlauf, der Unterschied wirkt sich jedoch nicht negativ aus. Damit nicht jeder Wert der Rampe überprüft werden muss, was viel Zeit kostet, kann die Tatsache genutzt werden, dass alle Rampen aufwärts berechnet werden, um die iterative optimierte Rechenweise nutzen zu können. Zu Beginn wird einfach der Ergebniswert überprüft und wenn nötig korrigiert. Sobald der erste Wert kleiner als $2^{16} - 1 = 0xFFFF$ ist, wird ohne Überprüfung fortgesetzt, da alle weiteren Werte kleiner sein werden. Auch beim Einsprung in eine Rampe wird diese Überprüfung durchgeführt, obwohl sie dann nur einmalig durchlaufen wird. Erfahrungsgemäß sind nur die ersten fünf bis zehn Werte davon betroffen, die Rampenlänge kann mehrere tausend Schritte betragen.

Der Roboter ist mit Kollisionssensoren und einem Laser Distanzsensor ausgestattet und kann auf diese Weise Hindernisse in der Ferne und auch unmittelbar bei der Kollision erkennen. Auf den zweiten Fall muss mit einer Notbremsung reagiert werden, bei der die Motoren schlagartig blockieren. Hierbei kann der Roboter leicht verrutschen, was eine unbekannte Ausrichtung zur Folge hat. Im Test zeigt sich jedoch, dass der Versatz zu keinen Problemen bei der Rückkehr zum letzten Feldmittelpunkt führt. Soll der Distanzsensor zum Einsatz kommen, ist eine planbare Bremsung mit einer Beschleunigungsrampe möglich. Damit im Falle einer solchen Schnellbremsung nicht erst die passende Rampe berechnet werden muss, wird diese im Voraus bei Vorbereitung der Fahrt präpariert. Um den Bremsweg zu verkürzen, es handelt sich dabei um einen Nothalt, soll eine höhere Beschleunigung als bei der normalen Rampe verwendet werden. Zunächst wird die maximal erreichbare Geschwindigkeit der geplanten Fahrt bestimmt. Ausgehend davon wird die Rampenlänge der Schnellbremsung mit höherer Beschleunigung als der normalen Fahrt berechnet. Mit diesen Parametern wird nun der Speicher von hinten nach vorn mit der passenden Bremsrampe gefüllt. Der verbleibende vordere Speicherteil wird für die eigentliche Fahrt verwendet. Im Fall einer Schnellbremsung mit Abbremsrampe kann nun die aktuelle Geschwindigkeit aus den Fahrdaten ermittelt und über eine Verhältnisrechnung mit den beiden Beschleunigungen, bzw. Rampenlängen der Einsprungpunkt in der Bremsrampe bestimmt werden. So wird ein weicher Übergang in die Bremsung sichergestellt.

### 5.5.5. Ausrichtung und Orientierung

Die initiale Ausrichtung des Roboters auf dem Feld findet mit geringer Drehgeschwindigkeit statt, da die diskret abgetasteten Sensoren dann mehr Messpunkte pro Winkelschritt liefern können und die Winkelauflösung auf diese Weise erhöht wird. Die

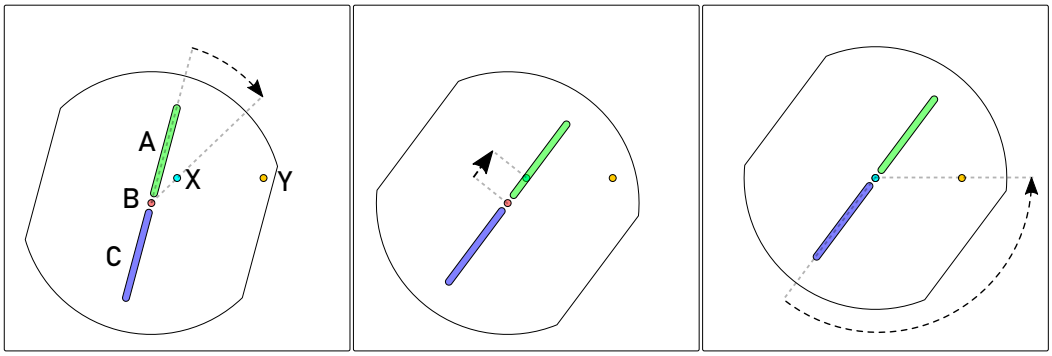


Abbildung 21.: Prinzipieller Ablauf der Ausrichtung des Roboters
Quelle: Eigene Abbildung

verringerte Geschwindigkeit stört im Gesamtablauf nicht, da sie nur einmalig zu Beginn ausgeführt wird. Die Sensoren sind wie in Unterabschnitt 3.2.6 beschrieben aufgebaut. In der folgenden Beschreibung der Abläufe werden, wie in Abbildung 21 dargestellt, mit Buchstaben abgekürzt. Die Marker LEDs und deren Signale werden mit X (Mittelpunktmarker) und Y (Orientierungsmarker) im Bodenpanel abgekürzt. Gleichermaßen beschreiben A (Sensor vorn), B (Sensor mitte), C (Sensor hinten) die drei Sensoren im Roboter und deren ausgewertete Signale. Die Marker und Sensoren sind zusätzlich farblich markiert, um mit Abbildung 5 konsistent zu sein und die Orientierung des Roboters zu kennzeichnen. Sollte zu Beginn der Sequenz B bereits X detektieren, so fährt der Roboter ein Stück rückwärts und dreht sich etwas gegen den Uhrzeigersinn, um den weiteren Prozess ausführen zu können. Die zu verfahrenden Winkel und Entfernungen wurden experimentell bestimmt und müssen zum Verständnis nicht exakt benannt werden. Der Roboter dreht sich nun solange im Uhrzeigersinn, bis A oder C das Signal von X detektieren, höchstens jedoch um 180°. Sollte X nicht gefunden werden können, so ist die Initialisierung fehlgeschlagen. Wurde X gefunden, so fährt der Roboter in die Richtung des signalgebenden Sensors. Bei A also vor und bei C zurück, bis B die maximale Amplitude von X erfasst. Abschließend dreht sich der Roboter gegen den Uhrzeigersinn, bis A oder C ein Signal von Y erfassen. Auch diese Drehung ist auf 180° begrenzt. Die Initialisierung ist mit Erkennen von Y abgeschlossen.

### 5.5.6. Ermittlung des Zielwinkels

Bei erfolgreicher Initialisierung wird die erkannte Ausrichtung des Roboters intern vermerkt. Gespeichert wird diese als Integerwert von Null bis Sieben, wobei mit Norden begonnen und in 45° Schritten gegen den Uhrzeigersinn durchnummeriert wird. Die Festlegung ist willkürlich gewählt. Eine unbekannte Orientierung wird mit einer Acht vermerkt. Alternativ wurde auch nach einem Schema gesucht, das direktes Addieren der Himmelsrichtungen zulässt. Beispielsweise soll N+S keine Bewegung verursachen,

N+O dagegen eine Bewegung in Richtung NO auslösen. Dieses Schema wurde getestet, hat aber Uneindeutigkeiten in anderen Programmabschnitten nach sich gezogen, weshalb die Wahl auf die erstgenannte Reihenfolge zurückfällt. Die Addition von Himmelsrichtungen ist nun in einer Funktion gekapselt und läuft über eine tabellarische Wertzuweisung, wie sie in Tabelle 5 dargestellt ist.

Tabelle 5.: Lookuptable zum Ermitteln des Zielwinkels

| | ↑ | ← | ↓ | → | × |
|---|---|---|---|---|---|
| ↑ | ↑ | ↖ | × | ↗ | ↑ |
| ← | ↖ | ← | ↙ | × | ← |
| ↓ | × | ↙ | ↓ | ↘ | ↓ |
| → | ↗ | × | ↘ | → | → |
| × | ↑ | ← | ↓ | → | × |

Die lineare Nummerierung reduziert eine Unterscheidung der Hauptachsen und Zwischenrichtungen auf eine Unterscheidung zwischen gerader und ungerader Codezahl. Die Funktion addiert nur Hauptrichtungen, da sonst feinere Unterteilungen in $22{,}5°$ Schritten berücksichtigt werden müssten, die in diesem Kontext jedoch unerwünscht sind. Da von den acht möglichen Richtungen jetzt nur noch die vier Hauptrichtungen übrig sind und deren Indizes stets gerade, also Restlos durch zwei teilbar sind, kann durch Halbieren die Lookup-Tabelle um $75\,\%$ in Größe reduziert werden. Hinzu kommt jedoch noch eine Zeile und eine Spalte, mit der auch die unbekannte Orientierung, hier mit einem $\times$ symbolisiert, berücksichtigt werden kann. So ergibt sich in Tabelle 5 die Zuordnung. Zwei gleiche Richtungen ändern sich nicht, eine valide Richtung und undefiniert ergibt wieder die valide Richtung, zwei gegensätzliche Richtungen heben sich zu undefiniert auf und zwei $90°$ versetzte valide Richtungen ergeben ihre $45°$ Mischform.

### 5.5.7. Ablauf einer Bewegung

Mit den beschriebenen Abläufen zur Positionsbestimmung und Richtungsberechnung, können die Navigationsabläufe aufgebaut werden. Sie nehmen den Befehl entgegen, den Roboter in eine bestimmte Zielrichtung zu bewegen. Im Testbetrieb hat sich eine bestimmte Bewegungsstrategie als besonders effizient erwiesen. Statt den Roboter auf den exakten Zielwinkel zu drehen, wird dieser mit einem Offset beaufschlagt. Da das aktuelle Postitionierungssystem hohe Winkeltoleranzen hat, wird der Roboter nicht exakt mit dem Detektor auf der Marker LED am Zielfeld eintreffen und muss sich in jedem Fall neu orientieren. Mit einem Offset kann sichergestellt werden, dass der Marker mit einem Schwenk in eine bestimmte Richtung auffindbar ist, was Zeit beim erneut Ausrichten einspart. Da der Drift von der idealen Bewegungslinie proportional

mit der Entfernung ansteigt, muss für Bewegungen entlang den Hauptrichtungen ein kleinerer Offset gewählt werden, als für diagonale Bewegungen. Zur Unterscheidung wird geprüft, ob der Code der Zielrichtung gerade oder ungerade ist. Anschließend wird der kleinstmögliche Drehwinkel zur Zielorientierung berechnet. Hierfür wird die Differenz zwischen den Codes der aktuellen Orientierung und der Zielorientierung gebildet. Ist der Wert kleiner als $-4$ oder größer als $4$, so würde eine Drehung um mehr als $180°$ stattfinden. Die effizientere Drehung ist in diesen Fällen in der anderen Richtung zu suchen. Dies geschieht durch Addition oder Subtraktion von acht, sodass der Betrag des Ergebnisses immer kleiner gleich vier ist. Der Winkel ergibt sich aus dem berechneten vorzeichenbehafteten Rotationscode multipliziert mit $45°$.

Nun sind alle Vorbereitungen getroffen und die eigentliche Ausführung beginnt. Der Roboter wird zunächst um den Rotationswinkel zuzüglich Offset in die Zielrichtung rotiert und dann versucht, ihn bis kurz vor den Mittelpunkt des Zielfeldes zu bewegen. Wird der frontale Kollisionssensor während der Geradeausfahrt betätigt, so stoppt der Roboter schlagartig und wird um die bereits gefahrene Distanz abzüglich einiger Zentimeter Puffer rückwärts zum Ausgangsfeld zurückbewegt, wo er sich auf dem Feld ausrichtet. Kann die Fahrt vorwärts ungehindert ausgeführt werden, so findet auf dem Zielfeld der Ausrichtungsprozess mit der anfangs genannten Vereinfachung dank des Winkeloffsets statt. Um den Ablauf weiter zu beschleunigen, werden manche Drehungen aufgeteilt. Zunächst wird in einer schnellen Bewegung grob nach dem Marker gesucht und sofort gestoppt, dann mit einer langsamen Drehung einige Grad in die Gegenrichtung die exakte Markerposition bestimmt. Diese Art der Drehung wird zum Überbrücken großer Winkel verwendet, also nicht zum Finden des Mittelpunktes bei Erreichen des Zielfeldes. Hier sorgt die Offsetstrategie für einen kurzen Suchbereich. Zum Auffinden des Orientierungsmarkers ist jedoch meist eine längere Drehung nötig. Hier bietet sich die zweigeschwindige Suche an. Wurde der Roboter auf einem Feld ausgerichtet und steht wieder in Ausgangsposition für den nächsten Befehl, so wird abschließend der Inhalt des aktuellen Labyrinthfeldes gelesen, um Feedback über den Erfolg der Bewegung zu ermöglichen.

### 5.5.8. Funkschnittstelle

Die eigentliche Schnittstelle zwischen dem PC und dem Robotersystem wird mittels WLAN realisiert. Das verwendete System on Chip (SoC) „ESP32“ bietet dafür alle Voraussetzungen und es sind passende Bibliotheken verfügbar. Aufgrund der Komplexität soll auf die WLAN Funktionalität nicht im Detail eingegangen werden. Speziell unter dem Aspekt, dass die Funktionalität durch externe Bibliotheken übernommen wird, liegt der Eigenanteil hauptsächlich in der Adaption externer Beispielprogramme. Da das WLAN Modul des ESP32 verhältnismäßig viel Strom verbraucht und

außerdem noch als Accesspoint dienen soll, kann dieses nicht direkt auf dem Roboter platziert werden. Die Akkulaufzeit würde sich durch das WLAN Modul drastisch reduzieren. Die Alternative ist, eine stromsparende Funkschnittstelle zwischen Roboter und das WLAN Modul zu platzieren. Die entstehende Funkbrücke muss dann die Daten zwischen beiden Systemen weiterleiten. Als stromsparendes Funksystem kommt der CC2500 von Texas Instruments zum Einsatz, da dieser bereits in früheren Projekten erprobt wurde. Dieser Chip arbeitet im $2{,}45$ GHz ISM Band, erreicht Datenraten von bis zu $500\,\frac{\text{kbit}}{\text{s}}$ und hat einen Spitzenstromverbrauch unter 30mA. Der Roboter initiiert regelmäßig Datentransaktionen und der Funkchip kann die Ruheperioden im Schlafmodus verbringen, was den mittleren Stromverbrauch stark senkt.

Um auch die Sendedauer zu reduzieren, muss eine flexible Paketlänge unterstützt werden. So können lange Nachrichten, wie beispielsweise der Inhalt eines RFID Tags auf einem Feld, gleichermaßen wie kurze Empfangsbestätigungen in minimaler Sendezeit übertragen werden. Zur Umsetzung der variablen Paketlänge muss neben den entsprechenden Registereinstellungen im Funkchip auch eine spezielle Sende- und Leseroutine implementiert werden. Der FIFO-Speicher im CC2500 umfasst nur $64$ Byte. Um Nachrichten bis zu $255$ Byte senden und empfangen zu können, muss der FIFO rechtzeitig aufgefüllt oder ausgelesen werden. Eine Lösung ist, den Füllstand des Speichers periodisch auszulesen, was jedoch Zeit kostet. Besser ist es, einen der zwei GPIO des Funkchips zu nutzen, um den Füllstand des Speichers zu signalisieren. Hierbei stehen verschiedene relevante Optionen zur Auswahl, die nachfolgend auch erwähnt werden.

### 5.5.9. Nachrichtenaufbau

Eine Nachricht ist folgendermaßen aufgebaut. Das erste Byte gibt die Länge der nachfolgenden Nachricht an, es wird in der Zählung nicht eingeschlossen. Aus diesem Grund können die eigentlichen Nachrichten auch nur $255$ Byte lang sein, das $256$te Byte enthält die Nachrichtenlänge. Dieser Parameter ist bei variabler Paketgröße essentiell und wird vom Funkchip für den gesamten Ablauf berücksichtigt. Das zweite Byte, also das erste der eigentlichen Nachricht, ist die Zieladresse, wobei sich null und $255$ auch als Broadcastadresse konfigurieren lassen, die von allen Empfängern akzeptiert wird. Dieser Wert wird vom Empfänger ausgewertet und bei falscher Adresse wird der Empfang abgebrochen. Nun folgt die eigentliche Payload. Eine CRC Prüfsumme wird automatisch von der Nachricht berechnet und angehängt. Beim Empfang wird diese automatisch kontrolliert und wieder entfernt. Das Ergebnis dieses Tests ist Teil der zwei Statusbytes, die sich optional der Nachricht beim Auslesen anhängen lassen [Tex, S.29].

### 5.5.10. Ablauf des Nachrichtenaustausches

Zum Senden einer Nachricht wird der FIFO des CC2500 zunächst geleert und anschließend mit der gewünschten Nachricht gefüllt. Ist die Nachricht zu lang, wird nur der passende Abschnitt weitergegeben. Anschließend wird der Chip in den Sendemodus versetzt. Er durchläuft eine automatische Taktkalibrierung und beginnt dann mit dem Senden der Nachricht. Die Funktion des Interrupt Pin wird im Vorfeld so eingestellt, dass es bei Unterschreiten eines Füllstandsschwellwertes im FIFO aktiviert und bei wieder Überschreiten deaktiviert wird. Solange nun Daten zum Senden vorhanden sind, wird bei jedem Interrupt der FIFO aufgefüllt, bevor er ganz geleert wurde. Der Prozessor wird so nur aktiv, wenn es nötig ist, statt regelmäßig auf Handlungsbedarf zu prüfen. Ist das letzte Byte der Nachricht in den FIFO übertragen, wird die Konfiguration des Interrupt Pins geändert, sodass es das Ende des Paketes anzeigt. Wird nun der nächste Interrupt ausgelöst, so wurde die Nachricht vollständig gesendet und das Funkmodul kann für alle Prozesse wieder freigegeben werden.

Das Empfangen einer Nachricht läuft ähnlich ab, wird jedoch zusätzlich von einer Funktion auf Zeitüberschreitung überwacht. Die Timeoutfunktion wird zu Beginn des Empfangsprozesses scharfgestellt und bricht bei Ablauf der maximalen Zeitdauer die Übertragung ab. So wird verhindert, dass Fehler das Funkmodul dauerhaft blockieren. Der FIFO wird geleert und der Interrupt Pin so konfiguriert, dass er bei der geringstmöglichen Schwelle von vier Byte im FIFO und außerdem zum Ende des Paketes aktiviert wird. Abschließend wird der Chip in den Empfangsmodus versetzt und wartet dann auf eintreffenden Nachrichten. Sobald die ersten vier Byte einer Nachricht eingetroffen sind, werden diese ausgelesen. Die erwartete Länge steht direkt am Anfang der Nachricht. Sie wird für weitere Verarbeitung separat gespeichert. Der Datenschwellwert wird auf $52$ Byte angehoben, um größere Datensegmente auszulesen und den Prozessor zu entlasten. Zusätzlich wird der laufende Timeoutzähler gestoppt und mit einer Zeit von sechs Millisekunden neu gestartet. Diese Zeit ist etwas länger als die Empfangsdauer eines Paketes mit maximaler Nachrichtenlänge. So kann neben dem Hauptprozess auch der eigentliche Empfangsvorgang vor Blockade geschützt werden. Bei jedem weiteren Interrupt der entweder durch Überschreiten der FIFO Schwelle oder durch das Ende der Übertragung ausgelöst werden kann, wird zunächst die verfügbare Datenmenge im Speicher ermittelt und anschließend ausgelesen. Hat der Lesezähler die Paketlänge erreicht, wird die Übertragung beendet. Der Timeout Zähler wird gestoppt und die empfangenen Daten aus dem temporären in den vorher vorgegebenen Datenspeicher übertragen. Das Funkmodul ist nun für die nächste Transaktion bereit.

### 5.5.11. Schnittstelle zur Funkbrücke

Entgegen des ursprünglichen Planes den Roboter via WLAN mit dem externen Computer kommunizieren zu lassen, sorgen Probleme mit der Netzwerkbibliothek des ESP32 dafür, dass dieser bei Übertragung von Datenpaketen zum PC abstürzt und neu startet. Ein Verlust der Verbindung und des zuletzt gesendeten Datenpaketes ist die Folge, Kommunikation mit dem Roboter ist so nicht möglich. Die Lösung ist, den CC2500 direkt mit einem Raspberry Pi zu verbinden und diesen als Netzwerkschnittstelle einzusetzen, jedoch kabelgebunden über Ethernet. Da das System kurzfristig zum Einsatz kommen soll, wird aufgrund entsprechender Vorkenntnisse und der Wiederverwendbarkeit erprobter Projekte Python als Programmiersprache gewählt. Der Raspberry Pi ist mit Python jedoch nicht in der Lage in Echtzeit auf die SPI Schnittstelle des Raspberry Pi zuzugreifen. Die Verzögerungen überschreiten die Sende- und Empfangsfenster des Roboters, welche dementsprechend angepasst werden müssen. Ebenfalls kommt eine Paketlänge über $64$ Byte nicht mehr infrage, da der interne FIFO Speicher des CC2500 nur genau so lang ist und überlaufen würde, da die Daten nicht rechtzeitig ausgelesen werden können. Eine langfristige Lösung wäre ein zusätzlicher Mikrocontroller, der die zeitkritischen Funktionen übernimmt und die Daten in einem größeren FIFO zur Verfügung stellt. Die kurzfristige Lösung verschickt Nachrichten mit reduzierter Länge. Die Sende- und Empfangsfunktionen des Roboters wurden nicht verändert, während der Raspberry Pi beim Empfang stets den gesamten FIFO ausliest und den Datensatz erst im Nachhinein auf die tatsächliche Nachricht einkürzt.

# 6. Umsetzung der Matlab Schnittstelle

Um mit dem Roboter interagieren zu können, soll laut Anforderungen eine Schnittstelle in Matlab bereitgestellt werden. Netzwerkinteraktionen in Matlab lassen sich sehr komfortabel über das Einbinden eines Python Programms lösen, was den Vorteil hat, dass die Funktionen direkt weitergenutzt werden kann, sollte das Kontrollsystem von Matlab entkoppelt werden. Alle Interaktionsmöglichkeiten sind in Funktionen gekapselt, die zunächst die übergebenen Parameter in einen Befehl einbinden und diesen zu einem Python String umwandeln. Der wird im Anschluss an das Python Script übergeben, das folgenden Ablauf ausführt. Geschützt von einer Timeoutfunktion die nach $15\,\mathrm{s}$ den Prozess abbricht, wird eine TCP Verbindung zum Raspberry Pi aufgebaut. Der Befehl wird abgeschickt und auf eine Antwort gewartet. Diese wird vom Raspberry Pi erst gesendet, wenn der Befehl durch den Roboter ausgeführt ist. Ziel ist es, nach dem Senden eines Befehls als nächste Antwort den aktuellen Status des Roboters nach Ausführung des Kommandos zu erhalten. Die empfangene Nachricht wird anschließend auf Fehlermeldungen geprüft, die vom Raspberry Pi zurückgegeben werden und diese dann vorausgewertet an die Befehlsfunktion zurückgegeben, wo die Antwort dekodiert und in einer Datenstruktur an das Matlabscript zurückgegeben wird. Im aktuellen Zustand der Software ist allerdings noch nicht gewährleistet, dass die Antwort nach einem Befehl bereits den aktuellen Status enthält, weshalb dieser im Anschluss durch einen Lesebefehl aktualisiert werden sollte.

# 7. Funktionstest

Um die korrekte Funktionsweise des Roboters in verschiedenen Situationen zu testen, ist ein Testszenario zu entwerfen. Die in Tabelle 6 aufgeführten Funktionen und Verhaltensweisen müssen überprüft werden

Tabelle 6.: Übersicht über Funktionen und Teststrategien

| Funktion | Teststrategie |
|---|---|
| auf dem Feld ausrichten | Nach Senden des Befehls muss Orientierung bekannt sein |
| In jede Richtung bewegen | Quadrat aus vier Feldern abfrahren, auch diagonal |
| In jeder Richtung Kollisionen erkennen | In jeder Richtung des Quadrates doppelte Fahrbefehle, Außerdem Diagonalfahrt in Ecke, Fahrt in diagonale Sackgasse, Geradeausfahrt an diagonale Wand |
| nach Kollision auf Vorgängerfeld zurückkehren | Nach jedem zweiten der doppelten Fahrbefehle darf sich die Feldposition nicht ändern |
| Feldinhalt platzieren | Nach dem Befehl muss sich der Feldinhalt geändert haben |
| Feldinhalt auslesen | Feldinhalt in Matlab empfangen |
| Feldinhalt konsumieren | Anzahl muss nach dem Befehl um eins reduziert sein, nach Konsumtion bei Feldinhalt gleich eins muss der Feldinhalt verschwinden |
| Feldinhalt aufheben | Feldinhalt muss nach Befehl verschwunden sein |
| Feldinhalt ablegen | Feldinhalt muss nach Befehl dem aufgehobenen Feldinhalt entsprechen, Ablegen ohne vorher aufzuheben darf keine Änderung veranlassen |

Entsprechend Tabelle 6 kann ein Testszenario konstruiert werden. Die Umgebung wird wie in Abbildung 22 gezeigt, festgelegt und der Startpunkt arbiträr auf Feld (1,0) gesetzt.

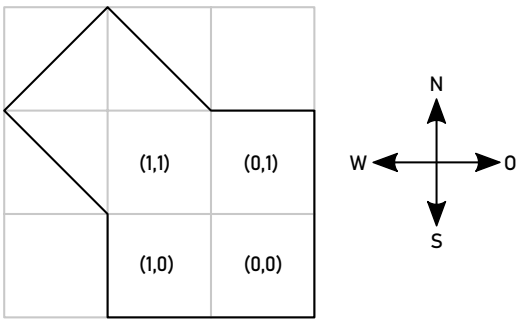


Abbildung 22.: Labyrinthaufbau für den Funktionstest
Quelle: Eigene Abbildung

Tabelle 7.: Testablauf mit erwarteten und gemessenen Ergebnissen

| | Feld | Befehl | Erwartete Änderung | Gemessene Änderung |
|---|---|---|---|---|
| 1 | (1,0) | Ausrichten | Feld: (1,0) | Position: (1 \| 0) |
| 2 | (1,0) | Platzieren: 1 x Käse | Inhalt: 1 x Käse | Panel contains: 1 x edible KAESE |
| 3 | (1,0) | Bewegen: N | Feld: (1,1) | Position: (1 \| 1) |
| 4 | (1,1) | Platzieren: 4 x Walnuss | Inhalt: 4 x Walnuss | Panel contains: 4 x edible WALNUSS |
| 5 | (1,1) | Bewegen: N | keine | Position: (1 \| 1), Panel contains: 4 x edible WALNUSS |
| 6 | (1,1) | Bewegen: O | Feld: (0,1) | Position: (0 \| 1) |
| 7 | (0,1) | Platzieren: 10 x Wasser | Inhalt: 10 x Wasser | Panel contains: 10 x edible WASSER |
| 8 | (0,1) | Bewegen: O | keine | Position: (0 \| 1), Panel contains: 10 x edible WASSER |
| 9 | (0,1) | Bewegen: S | Feld: (0,0) | Position: (0 \| 0) |
| 10 | (0,0) | Bewegen: NW | Feld: (1,1) | Position: (1 \| 1) |
| 11 | (1,1) | Konsumieren | Inhalt: 3 x Walnuss | Panel contains: 3 x edible WALNUSS |
| 12 | (1,1) | Bewegen: NW | keine | Position: (1 \| 1), Panel contains: 3 x edible WALNUSS |
| 13 | (1,1) | Bewegen: O | Feld: (0,1) | Position: (0 \| 1), Status jedoch erst beim nächsten Befehl aktualisiert |
| 14 | (0,1) | Aufheben | Inhalt: leer | Panel empty |
| 15 | (0,1) | Bewegen: S | Feld: (0,0) | Position: (0 \| 0) |
| 16 | (0,0) | Ablegen | Inhalt: 10 x Wasser | Panel contains: 10 x edible WASSER |
| 17 | (0,0) | Bewegen: SO | keine | Position: (0 \| 0), Panel contains: 10 x edible WASSER |
| 18 | (0,0) | Bewegen: W | Feld: (1,0) | Position: (1 \| 0), Status jedoch erst beim nächsten Befehl aktualisiert |
| 19 | (1,0) | Ablegen | keine | Position: (1 \| 0), Panel contains: 1 x edible KAESE |
| 20 | (1,0) | Bewegen: W | keine | Position: (1 \| 0), Panel contains: 1 x edible KAESE, Alignment |
| 21 | (1,0) | Konsumieren | Inhalt: leer | Panel empty |
| 22 | (1,0) | Bewegen: NO | Feld: (0,1) | Position: (0 \| 1), Status jedoch erst beim nächsten Befehl aktualisiert |
| 23 | (0,1) | Bewegen: NO | keine | Position: (0 \| 1), Panel empty |
| 24 | (0,1) | Bewegen: SW | Feld: (1,0) | Position: (1 \| 0) |
| 25 | (1,0) | Bewegen: SW | keine | Position: (1 \| 0), Panel empty |

In Tabelle 7 ist der gesamte Testablauf dargestellt. Von 15 durchgeführten Tests liefen alle so ab, wie es in der Spalte „Gemessene Änderungen“ dokumentiert ist. Ab dem dritten Testdurchlauf ist in Schritt 13, 17 und 22 noch ein zusätzlicher Lesebefehl ergänzt um den erfolgten Fahrvorgang auch im Testablauf dokumentieren zu können. Bei den genannten Schritten wird aus bisher unbekannten Gründen der Status trotz erfolgtem Feldwechsel erst nach dem nächsten Befehl zurückgegeben, der zusätzliche Lesebefehl listet den neuen Status bereits vor dem nächsten Fahrbefehl in der Konsole auf, womit die ausgeführte Bewegung nachgewiesen ist.

Nach manchen Bewegungen verliert der Roboter die Orientierung und muss neu ausgerichtet werten. Abbildung 23 zeigt für alle Tests die Anzahl dieser Vorgänge bei jedem Testschritt. In jedem Test wird der Roboter im ersten Schritt ausgerichtet, daher auch 15 mal. Bei 17 Schritten können keine Probleme festgestellt werden, wobei sämtliche Interaktionsvorgänge in diese Gruppe fallen. Sechs Schritte verzeichnen ein oder zwei Neuausrichtungen während des gesamten Testverlaufs. Im zwölften Testschritt dagegen werden neun Neuausrichtungen veranlasst. In diesem Schritt fährt der Roboter vom Feld $(1, 1)$ Nordwestlich in die diagonale Sackgasse. Die Orientierung auf dem Feld $(1, 1)$ bei der Rückkehr nach Kollision mit dem Hindernis, scheint Probleme zu bereiten.

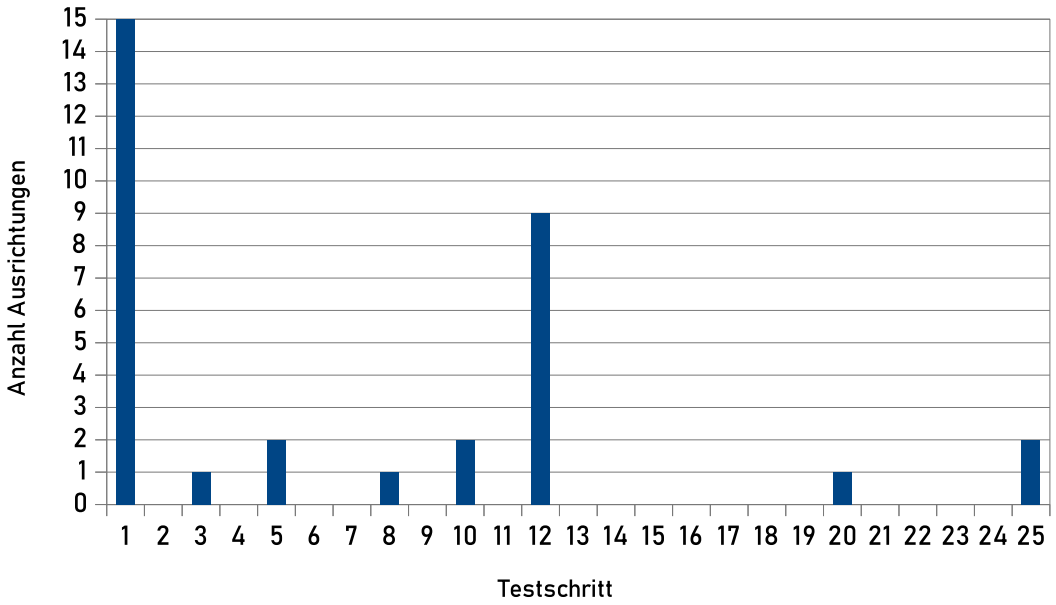


Abbildung 23.: Anzahl der Ausrichtungen pro Schritt
Quelle: Eigene Abbildung

Trotz kleiner Probleme ist der Test erfolgreich. Er hat gezeigt, dass der Roboter neben der Erfüllung der geforderten Funktionen auch in der Lage ist, nach Problemsituationen wieder einen stabilen Zustand anzunehmen. Probleme in der Ausführung sind per Software lösbar, ein manuelles Eingreifen von außen ist zu keinem Zeitpunkt notwendig.

# 8. Fazit

Die abschließende Einschätzung ist, dass der Roboter die gestellten Anforderungen erfüllt. Er ist in der Lage innerhalb der Umgebung zu navigieren, Hindernisse zu erkennen und nach Kollisionen wieder eine bekannte Position einzunehmen. Er kann mit den Labyrinthfeldern wie gefordert interagieren. Die Bodenpanele geben visuelle Rückmeldung zum Feldinhalt. Der Roboter kann über eine Funkschnittstelle Befehle von einem PC entgegennehmen, wobei diese mit Matlab steuerbar ist. Die Umgebung kann mit steckbaren Trennwänden verändert werden.

# 9. Ausblick

Trotz des erfolgreichen Testverlaufs ist zu sehen, dass der Roboter in manchen Durchläufen ein bis zwei mal die Orientierung verliert und neu ausgerichtet werden muss. Bei einigen Manövern ist auch eine ungenaue Ausrichtung dafür verantwortlich, dass Fahrten nahe entlang der Trennwand verlaufen und nur knapp einer seitlichen Kollision entgehen, die vom Roboter nicht detektiert werden könnte. Ebenso ist auf den Batteriestandswert kein Verlass. Die Zuverlässigkeit lässt sich insbesondere in den angesprochenen Punkten „Ausrichtungsgenauigkeit “ und „Batteriestandsmessung “ noch verbessern. Im Folgenden werden Vorschläge zur Optimierung des Prototyps vorgestellt.

## 9.1. Bodenpanele

Im Prototyp werden die RGB LEDs vom Roboter verdeckt und Interaktionen mit dem Panel so erst nach Verlassen desselben sichtbar. In der nächsten Version sollten die LEDs weiter außen platziert oder die Art der Visualisierung anders gestaltet werden.

## 9.2. Infrarotmarker und Markersensoren

Das Orientierungssystem ist sehr Toleranzbehaftet herausgestellt, da die Amplitude der Spektrallinien der Marker zu stark schwankt. So ist es schwer, den Mittelpunkt des Markers über die maximale Amplitude ausfindig zu machen. Dies kann umgangen werden, indem die Auswertung parallel im Zeit- und Frequenzbereich stattfindet, statt alles im Frequenzbereich zu erledigen. Eine analoge Auswertung der Spitzenwerte lässt eine fast verzögerungsfreie Bestimmung der Markerposition zu, woraufhin nur ein einziges Zeitfenster vom entsprechenden Detektor abgetastet und digitalisiert werden muss um anschließend im Spektrum den Marker zuzuordnen. Der Berechnungsaufwand sinkt enorm und die Zeitauflösung steigt. Da bislang langsame Fototransistoren als Detektoren dienen, könnte bei Bedarf auf schnelle Fotodioden mit einer passenden Verstärkerschaltung umgestellt werden. Die Marker- und Abtastfrequenzen könnten dann erhöht werden, um die Zeitdauer zum Digitalisieren eines Fensters zu senken. Vermutlich bietet jedoch eine Hüllkurvenauswertung der Fototransistoren schon die

gewünschte Responsivität.

### 9.3. ToF Sensoren

ST Microelectronics bietet mittlerweile den Nachfolger des verwendeten Time of Flight Distanzsensor an. Hierbei handelt es sich um ein $8x8$ Messpunktefeld, das in beide Richtungen einen Winkel von $60°$ vertikal und horizontal abdeckt. Mit diesem Sensor könnten Hindernisse dreidimensional abgetastet werden, was die ungefähre Bestimmung von Höhe, Form und Orientierung eines Ziels ermöglicht. Beispielsweise kann Höhe und Winkel einer Trennwand bestimmt werden, die ihrerseits an der flachen Vorderfläche identifizierbar ist. Der Distanzsensor des Prototyps ist vermutlich durch das Löten beschädigt und hat sporadisch kurze Aussetzer, weshalb er bislang nur testweise zur Umgebungserfassung genutzt wird. Ein Sensor mit 64 Messpunkten bei identischer Baugröße wie bisher, bietet viel Potenzial in der nächsten Version des Roboters von Nutzen zu sein.

### 9.4. Maussensoren

Aufgrund der hohen Toleranzen des aktuellen Orientierungssystems, und des präzisen Antriebs, ist der Einsatz dieser Sensoren bisher nicht notwendig. Auch hat sich gezeigt, dass die Maussensoren nicht als Messsysteme geeignet sind, da Chips aus unterschiedlichen Chargen unterschiedliche Distanzfehler aufweisen, die nachträglich durch Kalibrierung reduziert werden müssten. Da der Antriebsstrang präziser arbeitet, als es mit den Maussensoren gemessen werden kann, wären diese in keinem Fall eine Hilfe zur Verbesserung der Genauigkeit. Hinzu kommt, dass die Sensoren neben dem enormen Platzbedarf sehr schwer beschaffbar sind. Es ist keine Quelle für einzelne Sensoren verfügbar, sondern diese müssen aus Computermäusen ausgebaut werden. Die aktuelle Bodenfreiheit des Roboters von nur $0{,}8$ mm ist der kurzen Fokusdistanz der Sensoren geschuldet. In zukünftigen Versionen des Roboters sollte aus den genannten Gründen daher auf die Maussensoren verzichtet werden. Der freiwerdende Platz kann von anderen Baugruppen genutzt und die Bodenfreiheit erhöht werden, um auch bei Verunreinigungen der Fahrfläche eine zuverlässige Funktion zu gewährleisten.

### 9.5. Motoren und Motoransteuerung

Der Antrieb mit den Schrittmotoren in Verbindung mit einem sehr spielarmen Getriebe erweist sich als sehr präzise und mit hoher Wiederholgenauigkeit. Vorausberechnete Fahrten während müssen während ihrer Ausführung nicht durch leichte Kurvenfahrten

zu korrigiert werden. Auch Kurvenfahrten mit anderen Drehpunkten als dem Robotermittelpunkt, also Drehungen auf der Stelle, sind aus Gründen der benötigten hohen Wendigkeit nicht geeignet. Die Möglichkeit beide Motoren unabhängig voneinander anzusteuern ist damit nichtig. Indem beide Motoren von einem gemeinsamen Timermodul angesteuert und lediglich die Drehrichtung individuell eingestellt wird, ist eine Halbierung der benötigten Rechenleistung möglich. Mittlerweile ist ein noch kompakterer Motortyp verfügbar. Dessen Verwendung könnte den Stromverbrauch ein wenig senken und es möglich machen, den Roboter weiter zu verkleinern. So bleibt eventuell noch Platz für eine Verkleidung.

## 9.6. Platinenlayout

Die Aufteilung des Robotersystems in einen Sensor- und einen Hauptcontroller bietet zwar den Vorteil von höherer Rechenleistung und mehr Bauraum, unter Berücksichtigung der genannten Verbesserungsvorschläge, lässt sich dieser Mehraufwand jedoch vermeiden. Der Platz welcher durch Wegfallen der Maussensoren, eines Mikrocontrollers samt Peripherie und der Verwendung kleinerer Motoren frei wird sollte ausreichen, um die verbleibenden Baugruppen unterzubringen. Durch Reduktion der benötigten Rechenleistung für die Beschleunigungsrampen, den IR Sensor und das Wegfallen der Verzögerung und Rechenleistung durch die Kommunikation zwischen den Platinen, reicht ein einzelner Mikrocontroller aus. Eine Controllerplatine die sowohl Sensorik, als auch Aktorik und Kommunikationstechnik steuert, ist außerdem einfacher zu debuggen und spart Kosten in der Fertigung.

## Literaturverzeichnis


[Aca] ACADEMIC: *Allseitenrad*. https://de-academic.com/dic.nsf/dewiki/1050514. – zuletzt abgerufen am 25.09.2022

[Ava] AVAGO: *ADNS 5050*. – Datenblatt

[Bet] BETA LAYOUT: *Platinenvorgaben*. https://de.beta-layout.com/leiterplatten/infos/vorgaben/. – zuletzt abgerufen am 25.09.2022

[Esp] ESPRESSIF SYSTEMS: *ESP32 Series*. – Datenblatt

[Kle] *POM kleben*. https://www.klebeprofi.net/klebe-anleitungen/polyoxymethylen-pom-kleben/. – zuletzt abgerufen am 25.09.2022

[MBWa] *Federn in Parallelschaltung berechnen*. https://www.maschinenbau-wissen.de/skript3/mechanik/kinetik/279-feder-parallelschaltung. – zuletzt abgerufen am 25.09.2022

[MBWb] *Federn in Reihenschaltung berechnen*. https://www.maschinenbau-wissen.de/skript3/mechanik/kinetik/280-feder-reihenschaltung. – zuletzt abgerufen am 25.09.2022

[Mec] *Mecanum-Rad*. https://www.donkey-motion.de/donkeymotion/mecanum. – zuletzt abgerufen am 25.09.2022

[MEY21] MEYER, WERNER UND BORISLAVOV, BORISLAV UND ECKERT, FRIEDRICH UND RICHTER, CHRISTIAN UND RÖMER, RONALD UND BEIM GRABEN, PETER UND HUBER, MARKUS UND WOLFF, MATTHIAS: Formalisierung und Implementierung einer adaptiven kognitiven Architektur unter Verwendung von Strukturdiagrammen. (2021)

[Mic] MICROCHIP: *AVR446*. – Application Note

[Min] MINEBEA: *SMC10-20*. – Datenblatt

[MIT] *Mighty mouse*. https://www.technologyreview.com/2018/12/19/138508/mighty-mouse/. – zuletzt abgerufen am 25.09.2022

[Mul] MULTI CB: *Pooling Optionen*. https://www.multi-circuit-boards.eu/preise/leiterplatten/pooling-optionen.html. – zuletzt abgerufen am 25.09.2022

[NEM] NEMA: *ICS 16-2001*. – Norm

[OYOa] OYOSTEPPER: *6HS12-0304S*. – Datenblatt

[OYOb] OYOSTEPPER: *8HS15-0604S*. – Datenblatt

[SAU21] SAUER, Uwe und KOWAL, Julia: Batterietechnik Grundlagen und Übersicht. (2021)

[Sch] Schaad, St.: *Schrittmotor kurz erklärt*. `https://wiki.bu.ost.ch/infoportal/_media/hardware/sysp/bauteile/schrittmotor_kurz_erklaert_d.pdf`. – zuletzt abgerufen am 25.09.2022

[ST a] ST Microelectronics: *VL53L5CX*. – Datenblatt

[ST b] ST Microelectronics: *VL6180*. – Datenblatt

[Tex] Texas Instruments: *CC2500*. – Datenblatt

[Til18] Tille, Thomas: *Automobil-Sensorik 2*. 2018. – ISBN 978–3–662–56310–6

[Tri] Trinamic: *TMC2130*. – Datenblatt

## Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Stellen der Arbeit die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

---

Cottbus, 04.10.2022 Friedrich Eckert

# A. Inhalt des Datenträgers

### Testprotokolle (Textdateien)

Konsolenausabe in MATLAB während aller 15 Funktionstestdurchläufe

### Platinenlayouts (KiCAD Projekte)

FloorpanelBoard - Platinenlayout der Labyrinthfelder
WirelessBridge - obsoletes Platinenlayout der Funkbrücke mit ESP32
SensorBoard - Platinenlayout der Sensorplatine des Roboters
SensorControllerBoard - Platinenlayout des Sensorcontrollers des Roboters
FrontRiserBoard - Platinenlayout der vorderen Verbinderplatine
RearRiserBoard - Platinenlayout der hinteren Verbinderplatine
MainControllerBoard - Platinenlayout der Hauptplatine des Roboters

### CAD Modelle (Autodesk Inventor Projekte)

Labyrinth - Modell des Labyrinths, Hauptbaugruppe: Labyrinth.iam
Roboter - Modell des Roboters, Hauptbaugruppe: Roboter.iam

### Firmware (STM32CubeIDE Projekte/ Python Scripte)

F411_Roboter_SensorController - Firmware des Senssorcontrollers
F732_Roboter_MainController - Firmware des Maincontrollers
G071KBU6N_Roboter_FloorpanelController - Firmware der Bodenpanele
Raspberry Pi - Python Scripts der Datenbrücke
Matlab_Robot_Interface - Matlab Schnittstelle mit Testprogramm

### Dokumente (PDF Dateien)

Datenblätter, Application Notes und Normen