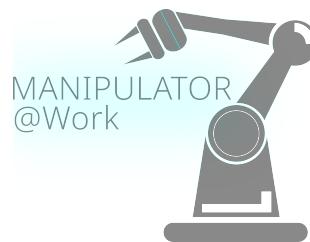

Technische Hochschule Nürnberg Georg Simon Ohm
Fakultät Elektrotechnik Feinwerktechnik Informationstechnik

Studiengang Bachelor Mechatronik / Feinwerktechnik
BMF 6 / BMF 7

Fachspezifisches Projekt:

ENTWICKLUNG EINES SENSORKOPFES MIT GREIFER FÜR EINEN MOBILEN WETTKAMPFROBOTER

Projektarbeit
Sommersemester 2023
Wintersemester 2023/2024



Projektteilnehmer:

Lucas Baptistella	Matr. Nr.: 2965311
Steffen Dillmann	Matr. Nr.: 3567920
Kim Sitzmann	Matr. Nr.: 3571426

Abgabedatum: 16. Januar 2024

Betreuer:
Prof. Dr.-Ing. Sander
Marco Masannek

Danksagungen

An dieser Stelle möchten wir uns bei allen bedanken, die uns bei der Erstellung dieser Projektarbeit begleitet, unterstützt und motiviert haben.

Zunächst gebührt unser Dank Herrn Prof. Dr.-Ing. Sander, der uns die Projektarbeit im AutonOHM -Team @Work ermöglicht hat. Besonders die hilfreichen Anregungen und die konstruktive Kritik sowie die Freiräume bei der Umsetzung der Projektarbeit haben die erfolgreiche Umsetzung ermöglicht.

Weiterhin möchten wir uns bei Marco Masannek für die Betreuung innerhalb des AutonOHM -Teams bedanken. Mit seiner fachlichen Expertise über die alte sowie die neue Roboterplattform stand er uns bei Anliegen jederzeit zur Verfügung.

Abschließend möchten wir uns bei dem gesamten AutonOHM -Team für das Projektthema bedanken. Die einzelnen Teammitglieder konnten bei fachbezogenen Fragen jederzeit unterstützen.

Zusammenfassung

Die Projektarbeit verfolgt das Ziel der Entwicklung eines Sensorkopfes für den Anschluss an einen Roboterarm auf einem mobilen Wettkampfroboter. Die Entwicklung teilt sich in die Bereiche Mechanik, Elektrotechnik und Informatik auf. Der mechanische Anschluss des Sensorkopfes wurde entsprechend dem LARA 3 Roboterarm, später einem Igus Roboterarm, konzipiert. Der Sensorkopf soll die bestehende Sensorik der alten Roboterplattform aufweisen und weiterhin eine Umgebungsbeleuchtung sowie eine Statusbeleuchtung besitzen. Die vollständige Funktionalität des Sensorkopfes für den Einsatz bei Wettbewerben ist bei der Bearbeitung des Projektes sicherzustellen.

Abstract

The purpose of the project is to develop a sensor head for the connection to a robot arm on a mobile competition robot. The development is organized in the sections of mechanical engineering, electrical engineering and computer science. The mechanical interface of the sensor head was designed according to the LARA 3 robot arm, later an Igus robot arm. The sensor head should feature the existing sensor technology of the old robot platform and continue to have ambient lighting and status lighting. The complete functionality of the sensor head for the usage in competitions has to be ensured during the working process on the project.

Inhaltsverzeichnis

Danksagungen	I
Zusammenfassung	II
1 Einleitung	1
2 Theoretische Grundlagen	3
2.1 Mechanik	3
2.1.1 SolidWorks	3
2.1.2 Autodesk Inventor	3
2.1.3 Prusa Slicer	3
2.2 Elektrotechnik	4
2.2.1 Mikrocontroller	4
2.2.2 Arduino Nano	4
2.2.3 Teensy 4.0	4
2.2.4 PMIC	5
2.3 Informatik	5
2.3.1 Nodes	6
2.3.2 Topics	6
2.3.3 Messages	7
2.3.4 Parameter Server	7
2.3.5 Master	7
2.3.6 Package	7
2.3.7 Rosserial und PlatformIO	8
3 Stand der Technik	9
3.1 Mechanik	9
3.2 Elektrotechnik	10
3.3 Informatik	10
3.3.1 Umgebungsbeleuchtung	10
3.3.2 Statusbeleuchtung	11
4 Umsetzung Sensorkopf	13
4.1 Mechanik	13
4.1.1 Konstruktion	13
4.1.1.1 Grundstruktur	13
4.1.1.2 Gehäuse	13
4.1.1.3 Greifer	14
4.1.1.4 Statusring	14
4.1.2 Fertigung	15
4.1.2.1 Aluminiumelemente	15
4.1.2.2 Kunststoffelemente	15
4.1.3 Montage	16
4.1.3.1 Sensorkopf	16
4.1.3.2 Greifer	16
4.1.3.3 Statusring	17
4.1.3.4 Gehäuse USB-Hub	17
4.1.3.5 Gehäuse Leiterplatte	17
4.1.3.6 Gehäuse FishEye-Kamera	17

4.1.3.7	Gehäuse 3D-Kamera	17
4.1.4	Anpassungen	18
4.1.4.1	Anpassungen für alternativen Roboterarm	18
4.1.4.2	Anpassungen Umgebungsbeleuchtung	18
4.2	Elektrotechnik	19
4.2.1	Bauteile und Beschaffung	19
4.2.1.1	LOGILINK UA0170 USB 3.0 Hub	19
4.2.1.2	Servomotor Dynamixel MX-28R	19
4.2.1.3	3D Intel RealSense D435 Tiefenkamera	20
4.2.1.4	180° Fisheye Kamera	20
4.2.1.5	SK6812 RGB LED	20
4.2.1.6	Teensy 4.0	21
4.2.2	Stromlaufplan, USB-Hub und Teensy-Anschlüsse	21
4.2.3	Umsetzung Lochrasterplatine	23
4.3	Informatik	24
4.3.1	Umgebungsbeleuchtung	24
4.3.2	Statusbeleuchtung	25
4.3.3	Ansteuerung Teensy-Mikrocontroller	25
4.3.3.1	Adafruit NeoPixel-Bibliothek	26
4.3.3.2	ROS	28
4.3.3.3	Lichtfunktionen Statusbeleuchtung	29
4.3.3.4	State-Machine	32
4.3.3.5	Ablauf Skript Teensy	32
4.3.4	Greifer-Einstellung	33
5	Ausblick	34
5.1	Mechanik	34
5.2	Elektrotechnik	34
5.3	Informatik	34
5.3.1	LED-Steuerung	34
5.3.2	Greifer-Einstellung	35
Abbildungsverzeichnis		36
Literaturverzeichnis		37

1 Einleitung

Im Zuge des Fachsemester 6 des Studiengangs Mechatronik / Feinwerktechnik soll ein Studienprojekt bearbeitet werden. Dieses fachspezifische Projekt konnte im Rahmen des AutonOHM @Work-Teams in einer Gruppe aus drei Personen umgesetzt werden.

Das Robotik-Team AutonOHM der technischen Hochschule Nürnberg forscht im Bereich der mobilen Robotik. In den beiden Disziplinen @Work und Rescue nimmt das AutonOHM-Team seit Jahren an den unterschiedlichen RoboCup-Wettbewerben weltweit teil. [1]

Die Disziplin @Work innerhalb des AutonOHM-Teams beschäftigt sich hierbei mit der Entwicklung und Konstruktion diverser Arbeitsroboter. Diese sind allgemein konzipiert verschiedene Arbeitsaufgaben aus den Anforderungen des RoboCup-Wettbewerbs zu erfüllen.

Die Anforderungen aus dem Wettbewerb sind hierbei spezifisch definiert. Die Arbeitsaufgaben gliedern sich in die Kategorien der allgemeinen Manipulation, dem Transportieren sowie Navigieren und das präzise Positionieren von Objekten. Diese werden bei der Durchführung unter anderem miteinander kombiniert. Es soll ein generelles Spektrum industrieller Robotikaufgaben abgebildet werden. Um die Aufgaben zu erfüllen, benötigt ein Wettbewerbsroboter unterschiedliche Komponenten. [2]

Die im Jahr 2020 von dem AutonOHM-Team hierfür entwickelte Roboterplattform weist alle benötigten Komponenten auf. Hierfür sind auf einer fahrbaren Plattform mit ausgewiesenen Transportflächen ein Towerarm mit einem Greiferaufsatz montiert. Weiterhin existieren unterschiedlichste Sensoren über alle Positionen auf der Roboterplattform verteilt. Mit diesem Industrieroboter konnte im Jahr 2021 der RoboCup in der Disziplin @Work gewonnen werden. [3]

Im Anschluss an den bestehenden Roboter soll eine neue optimierte Roboterplattform entwickelt werden. Neben der fahrbaren Plattform muss auch dieser Roboter einen Greifer für Manipulationsaufgaben aufweisen. Im Gegensatz zu der bestehenden Plattform wird anstelle eines intern entwickelten Towerarms ein industriell eingesetzter Roboterarm genutzt.

Von dem Team wurde vorgesehen den LARA 3 von Neura Robotics einzusetzen. Es handelt sich hierbei um einen Roboterarm mit sechs Freiheitsgraden und definierten Anschlussflächen für diverse Bauteile. [4]

Die Komponenten der bestehenden Roboterplattform können an diesen definierten Anschlussflächen jedoch nicht mehr genutzt werden. Weiterhin weist der LARA 3 Roboterarm keine Funktionalität für das Greifen von Objekten auf. Entsprechend muss für Manipulationsaufgaben ein neues Greifersystem entwickelt werden.

Das fachspezifische Projekt beschäftigt sich allgemein mit der Entwicklung eines neuen Sensorkopfes zum Anschluss an den LARA 3 Roboterarm. Der Sensorkopf soll hierbei Greiferaufgaben erfüllen und die weiterhin bestehende Sensorik aufweisen.

Neben der Kombination bereits bestehender Komponenten, sind für den neuen Sensorkopf Entwicklungen aus den unterschiedlichen Bereichen der Mechanik, Elektrotechnik und Informatik notwendig. In dem Bereich der Mechanik ist die Konstruktion eines neuen Sensorkopf-Gehäuses von Bedeutung. Es ist die Montage an dem Roboterarm-Flansch des LARA 3, die Halterung des Greifers, der Kameras

sowie der Beleuchtung sicher zu stellen. Weiterhin ist auf eine einfache Montage und Demontage zu achten. Das Arbeitspaket der Elektrotechnik beschäftigt sich vorwiegend mit der Spannungsversorgung aller benötigten Komponenten. Die korrekte Verteilung mit einfachen Anschlüssen sowie das Sicherstellen der Datenverteilung an einer zentralen Position ist erforderlich. Der Schwerpunkt in dem Bereich der Informatik besteht in der Entwicklung einer LED-Steuerung für den Sensorkopf. Die Ansteuerung der Umgebungsbeleuchtung sowie einer neuen Statusbeleuchtung in dem Sensorkopf, entsprechend der im Robotersystem angeforderten Zustände, ist umzusetzen. Weiterhin ist in dem Bereich der Informatik die korrekte Funktionalität des Greifers sicherzustellen.

In dem nachfolgenden Bericht werden nach einer theoretischen Einführung und dem Stand der Technik die Umsetzungen innerhalb des Projekts beschrieben.

Dennoch haben sich während der Entwicklung die Anforderungen in dem Projekt geändert. Eine Umsetzung des Sensorkopfes unter Verwendung des LARA 3 Roboterarms war nicht mehr gegeben. Innerhalb des AutonOHM-Teams wurde kurzfristig ein Einsatz eines Igus Roboterarms beschlossen. Die bereits bestehenden Entwicklungen des Sensorkopfes mussten hierfür angepasst werden.

2 Theoretische Grundlagen

In dem fachspezifischen Projekt werden in den drei Bereichen der Entwicklung unterschiedliche Komponenten und bestehende Konzepte verwendet. Für die nachfolgende Entwicklung des neuen Sensorkopfes werden diese angewendeten Komponenten und Programme zunächst theoretisch beschrieben.

2.1 Mechanik

In dem Bereich der Mechanik werden unterschiedliche Programme für die Entwicklung des Gehäuses des Sensorkopfes genutzt. Diese werden für das Erstellen der CAD-Daten sowie der Fertigungsdaten benötigt.

2.1.1 SolidWorks

Das Programm SolidWorks ist ein 3D-CAD-Programm. Es bietet allgemein die bewährten Funktionen wie weitere verbreitete 3D-CAD-Programme. Mithilfe der Software können 3D-Modelle, technische Zeichnungen und STL-Dateien erstellt werden.

SolidWorks wird in dem Projekt vor allem in den Anfangsphasen verwendet. Viele bestehende 3D-Modelle, welche im weiteren Verlauf zur Konstruktion des neuen Sensorkopfes genutzt werden, sind als SolidWorks Part-Dateien vorhanden (vgl. Kapitel 3.1).

2.1.2 Autodesk Inventor

Auch das Programm Autodesk Inventor ist ein 3D-CAD-Programm. Im Allgemeinen bietet es keinen größeren Funktionsumfang im Vergleich zu SolidWorks. Es ist auch hier die bewährte Funktionalität von 3D-CAD-Programmen gegeben. Dennoch sind die Konstruktionen in dem Projekt in Inventor umgesetzt. Aufgrund von fortgeschrittenen Erfahrungen in dem Umgang mit Inventor ist das Nutzen sinnvoll.

Das Übernehmen bereits bestehender Dateien, wie beispielsweise der SolidWorks-Part-Dateien, ist jedoch in Inventor nicht direkt möglich. Ein erneutes Erstellen der entsprechenden Bauteile bei einer Nutzung ist notwendig.

Bereits bei der Konstruktion können Beziehungen und Parameter der Bauteile in Inventor gewählt werden, sodass späterer Anpassungen leicht vorgenommen werden können. Auch ist ein direktes Zuweisen von Werkstoffen zu den Bauteilen möglich. Eigenschaften wie das Gewicht der Bauteile können über „iProperties“ ausgelesen werden. In dem Prozess der Konstruktion können hierdurch vor der Fertigung der Einzelteile bereits entsprechende Anpassungen durchgeführt werden. Beispielsweise kann hierdurch bereits vorab Gewicht eingespart werden.

Die erstellten Inventor-Part-Dateien lassen sich anschließend in STL-Dateien umwandeln, wobei weitere Einstellungen vorgenommen werden können. So können die Einheiten der gesetzten Maße im Bauteil definiert werden. Auch kann die Auflösung der zu generierenden Flächen feiner eingestellt werden. Bei der Umwandlung in die STL-Dateien werden die Flächen des Volumenkörpers in dreieckige Flächen umgewandelt. Durch das Vorgeben einer kleinen, maximalen Kantenlänge, werden die Dreiecke kleiner und die Auflösung in Folge größer.

2.1.3 Prusa Slicer

Mit dem Slicer können die erzeugten STL-Dateien in den für die Fertigung mit 3D-Druck nötigen Gcode umgewandelt werden. Der Prusa Slicer ermöglicht auch hierbei verschiedene Einstellungen. Über das Einstellen der Temperatur von dem Heizbett und dem Extruder kann die Haftung und das Fließverhalten des gewählten Kunststoffs verändert werden.

Weiterhin kann durch das Einstellen der Infill-Struktur die Form und Dichte des Gitters im inneren der Bauteile geändert werden. Das Einstellen hat großen Einfluss auf die Stabilität und das Gewicht

der gefertigten Bauteile. Nach dem Festlegen der verschiedenen Parameter werden in dem Slicer die Daten für den 3D-Drucker berechnet. Weiterhin wird das Gewicht und die Druckzeit als Informationen ausgegeben.

2.2 Elektrotechnik

In dem Bereich der Elektrotechnik werden diverse Standard-Bauteile in der Umsetzung verwendet. Bereits in der bestehenden Roboterplattform als auch bei der Entwicklung des neuen Sensorkopfes werden diese angewendet.

2.2.1 Mikrocontroller

Für die Entwicklungen in dem Projekt müssen diverse Mikrocontroller verwendet werden. Mikrocontroller sind hierbei Halbleiterchips, welche einen Prozessor enthalten. Ein Mikrocontroller besteht zusätzlich zum Prozessor aus einem Speicher und Input-/Output-Pins, welche es dem Controller ermöglichen, mit anderen Geräten oder der Umgebung zu kommunizieren. [5]

Mikrocontroller finden in verschiedenen Bereichen Ihre Verwendung. Im Bereich Embedded Systems (Deutsch: eingebettete Systeme) werden sie zum Beispiel in Haushaltsgeräten, Autos, medizinischen Geräten usw. eingesetzt. Des Weiteren werden Prozesse und Steuerungen beispielsweise in Robotern, Fabrikanlagen oder Smart-Home-Geräten automatisiert. Auch im Bereich Prototyping und zur Entwicklung von Elektronikprojekten werden Mikrocontroller verwendet, da sie zumeist kostengünstig, vielseitig und leicht verfügbar sind.

Für die Nutzung des Mikrocontrollers muss zunächst ein Programm (Skript) entwickelt werden. Um weiterhin eine ordnungsgemäße Funktion sicherzustellen, muss die richtige Stromversorgung gewählt werden. Abhängig von den Projektanforderungen müssen zusätzliche Komponenten wie Sensoren, Aktuatoren oder Displays mit dem Mikrocontroller verbunden werden. Dafür ist beim Einbau auf die richtige Pin-Belegung und die Anschlüsse zu achten. [5]

2.2.2 Arduino Nano

Der Arduino Nano ist eine kompakte Variante des Arduino-Boards und kann allgemein den Mikrocontrollern zugeordnet werden. Er ist leicht in kleineren Projekten einzusetzen, in welchen Platz ein begrenzender Faktor ist. Die Stromversorgung des Mikrocontrollers kann über einen Mini-B-USB-Anschluss oder eine geregelte externe 5-V-Stromversorgung an Pin 27 erfolgen. Je nach verwendetem Mikrocontroller verfügt der Arduino Nano entweder über einen 16 KB Flash-Speicher mit dem ATmega168 oder einen 32 KB Flash-Speicher mit dem ATmega328, um Programmcode zu speichern. Um beispielsweise Sensoren, Aktuatoren oder andere Komponenten anzuschließen, stehen 14 digitale Input- bzw. Output-Pins zur Verfügung. Für jeden Pin kann die Funktion pinMode(), digitalWrite() oder digitalRead() genutzt werden. [6]

2.2.3 Teensy 4.0

Der von PJRC entwickelte Teensy 4.0 ist ein kompakter und leistungsstarker Mikrocontroller auf Basis eines ARM-Cortex-M7-Prozessors. Mit seiner Taktfrequenz von bis zu 600 MHz und einem 2 MB großen Flash-Speicher, ist der Teensy in der Lage anspruchsvolle Berechnungen und Aufgaben mit höherer Geschwindigkeit auszuführen als der vorher beschriebene Arduino Nano. Die Spannungsversorgung von 5 V erfolgt über ein Standard-Micro-B-USB-Kabel. Er bietet weiterhin 40 digitale Input-/Output-Pins, 31 PWM-fähige Output-Pins und 14 analoge Input-Pins. [7]

Der Teensy verfügt somit über einen leistungsfähigeren Prozessor mit einer höheren Taktfrequenz, einen größeren Flash-Speicher aber eine etwas größere Bauform als der Arduino Nano.

2.2.4 PMIC

Ein PMIC (Power Management Integrated Circuit) ist ein elektronisches Bauteil und wird üblicherweise verwendet um die Energieversorgung sowie die Energieverteilung in einem elektronischen Gerät zu verwalten.

Zu den Aufgaben gehören beispielsweise die Stromregelung, indem der Stromfluss überwacht und gesteuert wird. Hierdurch wird eine stabile Stromversorgung sichergestellt, insbesondere wenn Baulemente unterschiedliche Stromanforderungen haben. Weiterhin wird die Energieeffizienz verbessert, indem die Energieversorgung optimiert und Energieverluste minimiert werden können.

Ein PMIC kann zudem als Schutzfunktion – sowohl als Überstrom-, als auch als Überspannungsschutz – verwendet werden.

Des Weiteren kann mit einem PMIC Spannung auf ein benötigtes Niveau geregelt oder durch mögliche programmierbare Funktionen die Energieverwaltung nach Bedarf angepasst werden. [8]

2.3 Informatik

In dem Bereich der Informatik muss die allgemeine Entwicklung weiterer Software-Komponenten, in Bezug auf die gesamte Roboterplattform, entsprechend einer nahtlosen Integration zu den vorhandenen Software-Bestandteilen erfolgen. Bereits bestehende Software-Komponenten sollen durch Änderungen und Neu-Entwicklungen verschiedener weiterer Komponenten nicht beeinflusst werden und weiterhin betriebsbereit bleiben. Dies setzt für die Neu- und Weiterentwicklung von Software-Paketen voraus, dass hierbei kompatible Umgebungen und beispielsweise auch identische Programmiersprachen, respektive Schnittstellen, genutzt werden.

Sowohl die bisher vorhandene Plattform als auch die neue Roboterplattform werden mit dem Roboter-Betriebssystem „ROS 1“ („Robot Operating System“, nachfolgend „ROS“) betrieben [9]. ROS ist das open-source Meta-Betriebssystem, welches auf einem zentralen Computer innerhalb der Roboterplattform mit einem Ubuntu-Betriebssystem ausgeführt wird. Es bietet allgemein Möglichkeiten zur Hardware-Abstraktion, Gerätesteuerung, unterschiedliche Implementierungen zur Nachrichtenübermittlung zwischen verschiedenen Prozessen, sowie diverse Integrationen von Tools und Bibliotheken. [9]

In der bestehenden Roboterplattform sowie in den Änderungen der neuen Plattform, ist der Hauptanteil der Software-Pakete in der objektorientierten Programmiersprache C++ verfasst [10]. Grundsätzlich wird durch ROS nicht erzwungen, ausschließlich eine einzelne Programmiersprache in dem gesamten System zu nutzen [11]. Dennoch werden die nachfolgenden Aufgaben zur einheitlichen Verwendung in den identischen Programmiersprachen realisiert.

Das Meta-Betriebssystem ROS weist weiterhin eine für die Entwicklung und den Betrieb von Robotern grundlegende charakteristische Architektur auf. Diese Architektur ist für die Entwicklung und Integration der Software-Komponenten von Bedeutung. Die wichtigsten Merkmale und Funktionalitäten der ROS - Peer to Peer - Architektur zur Datenverarbeitung werden deshalb nachfolgend beschrieben. Die beschriebenen Zusammenhänge sind zudem in Abbildung 2.1 dargestellt. [9], [12]

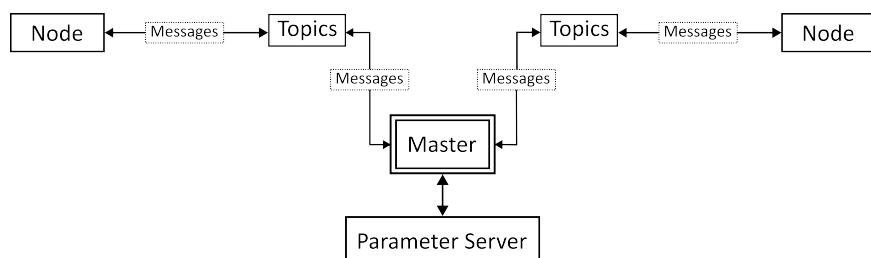


Abbildung 2.1: Visualisierung ROS-Architektur computational graph [13]

2.3.1 Nodes

Nodes (Deutsch: Knoten) stellen in ROS jeweils einen einzelnen Prozess dar, welcher entsprechend seiner Aufgabe die direkten Berechnungen und Steuerungen ausführt. [12], [14]

Die vielfältigen Berechnungsprozesse innerhalb eines gesamten Robotersystems werden bei einer Implementierung in ROS üblicherweise in jeweils einzelne Knoten aufgeteilt. Innerhalb einer gesamten ROS-Roboterumgebung gibt es deshalb meist viele verschiedene Knoten. Das Aufteilen in jeweils unterschiedliche Knoten hat diverse Vorteile für den Aufbau der Architektur in der gesamten Roboterplattform. [12], [14]

Ist ein einzelner Knoten fehlerhaft oder bricht während der Bearbeitungsaufgabe unvorhergesehen ab, so beeinflusst dies die weiteren laufenden Prozesse nur geringfügig. Entsprechend der Bedeutung des Knotens können einzelne Bearbeitungsaufgaben des Roboter-Systems beeinträchtigt sein, es ist jedoch nicht das gesamte Robotersystem von dem direkten Absturz betroffen. Funktionen, welche entsprechend nicht in einem unmittelbaren Zusammenhang mit dem fehlerhaften Knoten stehen, werden weiterhin ausgeführt. Zudem ist es möglich, nur bestimmte Knoten zur Bearbeitung zu aktivieren. Die Funktion einzelner Knoten kann entsprechend unabhängig von weiteren aktiven Knoten getestet werden. Auch kann hierdurch eine Selektion der ausgeführten Funktionen bei dem Betrieb des Roboters durchgeführt werden. [11], [14]

Weiterhin kann durch das Konzept der Knoten die gesamte Komplexität des Quellcodes der Roboterplattform übersichtlicher gestaltet werden. Einzelne Funktionsabschnitte werden, wie beschrieben, in die zuständigen Knoten gegliedert. Durch einen anschließenden Datenaustausch zwischen den einzelnen Prozessen sind definierte Schnittstellen zwischen den unterschiedlichen Funktionen mit erwarteten Ergebnissen vorhanden. [11], [14]

Entsprechend müssen die verschiedenen Knoten untereinander kommunizieren können. Dies ist unter anderem durch Topics (vgl. Kapitel 2.3.2) möglich. Weitere Kommunikation kann durch Services und den Parameter-Server stattfinden (vgl. Kapitel 2.3.4). [12]

Jeder Knoten besitzt einen eindeutigen Namen und Typ, welcher den Knoten im gesamten Roboter-System identifiziert. [14]

2.3.2 Topics

Mittels der in der ROS-Architektur vorhandenen Topics können die verschiedenen Knoten kommunizieren. Topics sind hierbei allgemeine BUS-Verbindungen, welche nach ihrer entsprechenden Definition individuell benannt sind. Die tatsächliche Kommunikation von Daten zwischen teilnehmenden Knoten findet durch den Austausch von Messages über ein Topic statt (vgl. Kapitel 2.3.3). [11], [12], [15]

Charakteristisch arbeiten Topics mit einer anonymen publisher/subscriber-Architektur. Knoten, welche zur Bearbeitung Daten als Input benötigen, müssen das entsprechende Topic abonnieren (subscribe). Dagegen veröffentlichen (publish) Knoten, welche Daten generieren, diese auf dem betreffenden Topic. Zu einem Topic kann es jeweils mehrere Subscriber- und Publisher-Knoten geben. Der Austausch und der Empfang einzelner Messages auf einem Topic sind hierbei nicht direkt voneinander abhängig. Auch in den einzelnen Knoten ist bei dieser Architektur nicht bekannt, mit welchen weiteren Knoten kommuniziert wird. Es ist ausschließlich relevant, welche Messages über welches Topic empfangen und gesendet werden können. Diese Information muss entsprechend der Systemdefinition der Roboter-Plattform festgelegt werden. [15]

Allgemein sind Topics in der ROS-Architektur für einen unidirektionalen Austausch von Daten konzipiert. In Knoten können über Topics weder Daten / Messages angefordert werden, noch sind Bestätigungen über das Erhalten von Daten vorgesehen [15]. Für eine bidirektionale Kommunikation

zwischen Knoten sind in der ROS-Architektur sogenannte „services“ vorhanden. [12] Diese werden nachfolgend jedoch nicht genauer betrachtet, da eine Anwendung bei der Entwicklung des Sensorkopfes nicht gegeben ist.

Weiterhin haben Topics keine Abhängigkeit zu verschiedenen Datentypen und sind nicht allgemein typisiert. Eine Identifizierung findet nur anhand der individuellen Benennung statt. Da der Daten austausch über Topics mittels der Messages stattfindet, lässt sich jedes Topic jedoch durch den entsprechenden Message-Typ charakterisieren. Es findet dennoch keine allgemeine Durchsetzung eines bestimmten Daten- / Message-Typ innerhalb eines Topics statt. Grundsätzlich kann jeder Message-Typ veröffentlicht werden. Einzelne Knoten können dennoch nur Daten mit dem identisch definierten Datentyp empfangen. [15]

2.3.3 Messages

Wie bereits beschrieben findet der eigentliche Datenaustausch zwischen unterschiedlichen Knoten durch Messages (Deutsch: Nachrichten), welche über definierte Topics versendet werden, statt. Die Knoten veröffentlichen Messages auf einem Topic oder empfangen diese. [12], [16]

Nachrichten bestehen hierbei aus einfachen typisierten Datenstrukturen. Solche sind beispielsweise die konventionellen Typen, wie Integer, Boolean und Floating Point. Aber auch Arrays aus diesen Typen sind in einer Nachricht möglich. Nachrichten können weiterhin aus beliebig verschachtelten Strukturen aus diesen Typen oder weiteren Nachrichten bestehen. [12], [16]

Die Struktur und der Typ von Nachrichten können in „msg“-Dateien frei definiert werden. „msg“-Dateien sind hierbei einfache Textdateien, welche die enthaltenen Datentypen und Werte beschreiben. Aus diesen Dateien wird mithilfe der ROS-Tools der entsprechende Quellcode für den Nachrichtentyp automatisch erzeugt. Die „msg“-Dateien sind als Bestandteil eines ROS-Packages (vgl. Kapitel 2.3.6) in einem „msg“-Unterverzeichnis hinterlegt. [16]

2.3.4 Parameter Server

Der Parameter-Server ist ein gemeinsam genutztes dictionary, über welches Knoten während ihrer Laufzeit auf definierte Parameter zugreifen können. Er erlaubt das Speichern von Daten an einer zentralen Stelle, auf die, über entsprechende Benennung, zugegriffen werden kann. Auch das Ändern der enthaltenen Daten ist möglich. Der Parameter-Server ist entsprechend einer globalen Sichtbarkeit für alle Knoten ausgelegt. Er eignet sich besonders für definierte statische Konstanten. [12], [17] Die einzelnen Parameter sind entsprechend der üblichen ROS-Namenskonvention benannt. Diese folgen hierbei einem hierarchischen Aufbau, sodass verschiedene Parameter aufgrund ihrer Benennung nicht kollidieren. Auch ist es hierdurch möglich, alle Parameter eines hierarchischen Zweiges abzufragen. [17]

Der Parameter-Server kann Daten der Standard-Typen, wie beispielsweise String, Integer, Float und Boolean speichern. [17]

2.3.5 Master

Die tatsächliche Kommunikation in einem ROS-System wird durch den ROS-Master ermöglicht. Die allgemeine Funktion einer ROS-Roboterplattform ist ohne aktiven Master nicht gegeben. [12], [18] Der Master stellt an zentraler Stelle die Namensservices und Registrierungsservices für die einzelnen Knoten in dem System bereit. Den Knoten soll somit ermöglicht werden, sich gegenseitig zu lokalisieren. Er verfolgt weiterhin die verschiedenen publisher und subscriber zu den einzelnen Topics. Zudem stellt der ROS-Master den Parameter-Server bereit. [12], [18]

2.3.6 Package

In ROS werden die einzelnen Softwarekomponenten innerhalb eines Dateisystem in einzelnen Packages realisiert. Ein ROS-Package kann üblicherweise verschiedene Nodes sowie von ROS unabhängige

Bibliotheken, verschiedene Konfigurationsdateien oder auch externe Software-Pakete enthalten. [12], [19]

Das Ziel einzelner Pakete ist es, die Funktionsabschnitte innerhalb des gesamten ROS-Dateisystems in Abschnitten zu sammeln. [19]

Die einzelnen Pakete sind die kleinsten nutzbaren Abschnitte innerhalb des ROS-Systems. Diese können individuell erstellt, respektive kompiliert, werden und stellen somit unabhängig voneinander einzelne Funktionen bereit. Üblicherweise folgt der Aufbau eines Packages einer definierten Dateistruktur. [19]

2.3.7 Rosserial und PlatformIO

Neben dem direkten ROS-System werden in der Umsetzung der Software in dem Projekt diverse Mikrocontroller verwendet. Bei diesen werden die ROS-Strukturen nur zur Integration sowie als Schnittstelle in die bestehende Roboter-Architektur genutzt, eine direkte Anwendung der Struktur in den Mikrocontrollern ist jedoch nicht gegeben.

Auf den Mikrocontrollern werden individuelle Skripte betrieben, welche sich in der ROS-Struktur mittels „rosserial“ als Node einbinden lassen. Rosserial ist eine ROS-Bibliothek, welche über ein Protokoll das Senden und Empfangen der bekannten ROS-Nachrichten über beispielsweise einen seriellen Port ermöglicht. ROS-Funktionalitäten können hiermit auch in eingebetteten Systemen, identisch zu tatsächlichen Nodes, verwendet werden. In den Skripten der Mikrocontroller können anschließend auch die bekannten Funktionen, wie das Abonnieren und Veröffentlichen von Nachrichten über Topics, genutzt werden. [20]

Die Entwicklung der Skripte für die Mikrocontroller muss jedoch entsprechend unabhängig stattfinden. Die Skripte müssen weiterhin auf die Flash-Speicher der Mikrocontroller geschrieben werden, sodass diese bei der Aktivierung des Mikrocontrollers ausgeführt werden. Hierfür wird bei der Entwicklung das Werkzeug PlatformIO genutzt. PlatformIO stellt hierbei allgemein diverse Werkzeuge zur Entwicklung von Software für eingebettete Systeme bereit. Neben dem Erstellen und Kompilieren des Codes kann dieser hiermit auch auf die entsprechenden Mikrocontroller geschrieben werden. Es bestehen weiterhin umfassende Funktionalitäten für das Debugging und das Nutzen verschiedener zusätzlicher Bibliotheken, dass von besonderer Bedeutung in dem Projekt ist. PlatformIO wurde bei der Entwicklung der Skripte in dem Projekt, als Erweiterung in dem Code-Editor Visual Studio Code verwendet. [21], [22]

3 Stand der Technik

Wie bereits zu Beginn beschrieben (vgl. Kapitel 1) existiert eine funktionsfähige Roboterplattform als Vorgänger für die neue Roboterplattform. Auch in dieser Plattform ist ein Arm mit Greifer realisiert. Entsprechend existieren bereits diverse Komponenten und Konzepte, welche auch in dem neuen Sensorkopf angewendet werden sollen. Allgemein soll der neue Sensorkopf mindestens die identischen Funktionen erfüllen, wie in der aktuell bestehenden Realisierung des Roboterarms mit Greifer. Nachfolgend werden die bereits bestehenden Funktionen beschrieben. Auf dieser Grundlage wird anschließend die Entwicklung des neuen Sensorkopfes umgesetzt.

3.1 Mechanik

In dem Bereich der Mechanik und der Konstruktion ist auf der Roboterplattform ein Towerarm montiert (vgl. Abbildung 3.1). Dieser wurde von dem Roboterteam eigens konstruiert. Die allgemeinen Bewegungsmöglichkeiten sind hierbei vergleichbar mit der Arbeitsweise eines Krans. Alle Bewegungen erfolgen mittels 3D-gedruckter Zahnräder sowie Zahnstangen.

Der Vorteil der bestehenden Konstruktion ist eine großzügige Platzverteilung in dem oberen Bereich des Roboterarms. Insbesondere für die Unterbringung der Elektronikkomponenten und der Sensorik sowie die Verlegung der benötigten Kabel ist ausreichend Platz vorhanden.

An dem Towerarm ist weiterhin ein Greifer montiert. Über dem Greifer befindet sich die verbaute Sensorik. So befindet sich an dieser Position auch die Halterung der 3D-Kamera. Diese zeigt in der Positionierung fest nach unten. Um ein Objekt erkennen zu können, muss der gesamte Greifer mit einem Motor in eine seitliche Position abgewinkelt werden.



Abbildung 3.1: Bestehende Roboterplattform mit Towerarm



Abbildung 3.2: Bestehende Greiferkonstruktion

Weiterhin besteht die direkte Konstruktion des Greifers (vgl. Abbildung 3.2). Diese ist in der Entwicklung des neuen Sensorkopfes in der Funktionsweise zu übernehmen. Dennoch bestehen insbesondere bei den zugehörigen CAD-Daten diverse Probleme.

Die CAD-Daten sind teilweise unvollständig und müssen für eine anschließende Konstruktion ergänzt werden. Zudem sind die Dateien in älteren Versionen der CAD-Software SolidWorks erstellt, wodurch Beziehungen zwischen einzelnen Maßen nicht korrekt funktionieren. Anpassungen an den 3D-Daten stellen sich deshalb sehr aufwändig dar.

Darüber hinaus sind einzelne Maße in den bestehenden Konstruktionen nicht fertigungsfreundlich ausgelegt. Diese weisen meist ungerade Werte auf, bei einer Definition auf 3 Nachkommastellen. Für eine Fertigung der Bauteil mit 3D-Druck-Verfahren stellt dies meist kein Problem dar, jedoch sind die Maße für eine Fertigung mit alternativen Verfahren nicht geeignet.

3.2 Elektrotechnik

In dem Towerarm der bestehenden Roboterplattform sind die unterschiedlichen elektrotechnischen Bauteile bereits vollständig verbaut. Es handelt sich hierbei um die gleichen Sensoren und Bauteile, welche auch in dem neuen Sensorkopf vorhanden sein sollen (vgl. Kapitel 4.2). Wie bereits in der Mechanik beschrieben (vgl. Kapitel 3.1) besteht in dem Towerarm eine großzügige Platzverteilung. Die elektronischen Bauteile konnten somit in größeren Abständen voneinander verbaut werden.

Die allgemeine Spannungsversorgung der Bauteile sowie die Datenübertragung findet über mehrere verbaute Kabel statt. Die Kabel sind hierbei als Kabelstrang durch den gesamten Towerarm geführt. Die Verteilung der Datenübertragung findet mittels zwei verbauter „LOGILINK UA0170 USB 3.0“-Hubs, sowie einer selbst gefertigten Steckerleiste auf einer Lochrasterplatine, statt.

Neben der allgemeinen Sensorik befinden sich auf dem verfahrbaren Greifer auch diverse LED-Streifen als Umgebungsbeleuchtung (vgl. auch Kapitel 3.3.1). Die Ansteuerung der Umgebungsbeleuchtung erfolgt über einen Arduino Nano mit einem PMIC (vgl. Kapitel 2.2.2 und Kapitel 2.2.4).

3.3 Informatik

In dem Bereich der Informatik sind die grundlegenden Funktionen der LED-Steuerungen bereits auch in der bestehenden Roboterplattform vorhanden. Auch bei der bestehenden Plattform existiert eine Umgebungsbeleuchtung. Weiterhin existiert eine Statusbeleuchtung. Diese ist im Boden der Roboterplattform verbaut.

Entsprechend der vorhandenen Beleuchtungen, besteht auch die allgemeine Software-Funktionalität zur Ansteuerung der LEDs. In der Roboterplattform unterscheidet sich die Hardware-Realisierung zwischen den beiden LED-Kategorien jedoch grundlegend. Hierdurch existieren zwei verschiedene Ansätze zur Ansteuerung der LED-Beleuchtungen.

Nachfolgend wird zunächst für die beiden Beleuchtungskategorien die unterschiedliche Realisierung der Beleuchtung und der Ansteuerung in der Software beschrieben.

3.3.1 Umgebungsbeleuchtung

Bei den LEDs der Umgebungsbeleuchtung handelt es sich um einfache LED-Streifen. Auf diesen LED-Streifen sind diverse einzelne LEDs mit einer weißen Lichtfarbe verbaut. Eine individuelle Ansteuerung dieser einzelnen LEDs, sowie eine direkte Steuerung des gesamten LED-Streifen ist nicht möglich.

Bei den LEDs handelt es sich um spannungsgesteuerte Leuchtdioden, welche keine Funktionalität für potenzielle Farbwechsel und ähnliche Eigenschaften bereitstellen. Es kann ausschließlich über zwei Kabelanschlüsse eine Eingangsspannung angelegt werden. Wird eine Spannung entsprechend der Nennspannung der LEDs angelegt, so leuchten diese in ihrer maximalen Intensität. Eine lineare Abstufung der Spannung ist möglich. Hierdurch lässt sich die Leuchtintensität der LEDs ändern. Ist an den LED-Streifen keine Eingangsspannung angelegt, so sind diese deaktiviert.

Auch bei der vorhandenen Ansteuerung der Umgebungsbeleuchtung ist das Variieren der Leuchtintensität vorgesehen [10]. Entsprechend der verbauten LEDs muss diese jedoch über die Spannungsmodulation erfolgen. Wie in den Kapiteln der Elektrotechnik dargestellt, ist hierfür ein Power-Management-IC (PMIC) verbaut (vgl. Kapitel 3.2). Dieser regelt die Ausgangsspannung in dem definierten Spannungsbereich linear. Am Eingangspin des PMIC wird hierfür ein proportionales PWM-Signal (Pulsweitenmodulation) benötigt.

Das PWM-Signal wird in der bestehenden Roboterplattform mittels eines verbauten Arduino Mikro-

controller erzeugt. In dem Software-Skript des Arduino wird das PWM-Signal für die gewünschten Leuchtintensität an dem Ausgangspin des Mikrocontrollers aktiviert.

Der Arduino ist über eine serielle Schnittstelle (USB) mit dem ROS-System der Roboterplattform verbunden. Für die Umgebungsbeleuchtung werden Float-Werte in dem hierfür definierten Topic „`detection_leds/intensity`“ gesendet. In dem Skript des Arduino ist dieses Topic entsprechend abonniert. [10]

Die Float-Werte stellen die gewünschte Leuchtintensität der Umgebungsbeleuchtung dar. Der erwartete Wertebereich repräsentiert hierbei den Prozentwert der geforderten Beleuchtungsintensität zwischen 0.0 und 1.0. Für die Ausgabe des PWM-Signals auf dem Ausgangspin des Arduinos werden Werte zwischen 0 und 255 benötigt.

Das Skript des Arduinos weist die konventionelle Struktur eines Arduino-Skript im Zusammenhang mit der ROS-spezifischen Funktionalität auf [23]. Die gesendeten Float-Werte werden in einer Callback-Funktion des Subscribers empfangen und direkt auf Ausgangswerte zwischen 0 und 255 umgerechnet. Bereits in der Callback-Funktion werden die umgerechneten Intensitätswerte durch Aufruf einer weiteren Funktion auf den Ausgangspin des Arduino geschrieben. Dies bewirkt, dass nach dem Erhalten neuer Intensitätswerte, diese ohne weitere Verzögerungen auf die Umgebungsbeleuchtung angewendet werden. In der Struktur eines Arduino-Skriptes existiert zudem ein definierter „`setup`“-Bereich, welcher einmalig bei Start des Arduinos ausgeführt wird. [23] Bei der bestehenden Roboter-Plattform ist in diesem Bereich das dreimalige Aufleuchten der Umgebungsbeleuchtung als Start-Signal der Roboterplattform implementiert.

3.3.2 Statusbeleuchtung

Die Statusbeleuchtung in der bestehenden Roboterplattform ist mit variablen LEDs realisiert. An dem Untergrund der Roboterplattform sind hierbei zwei identische LED-Streifen in paralleler Position angebracht (vgl. Abbildung 3.3). Diese beleuchten den Boden unterhalb des Roboters und sind für den Betrachter von außerhalb nicht direkt sichtbar.

Im Gegensatz zu der Umgebungsbeleuchtung handelt es sich bei den verwendeten LED-Streifen um individuell direkt ansteuerbare LEDs. Auf den LED-Streifen sind jeweils einzelne Leuchtdioden verbaut. Hierbei sind auf jedem Streifen jeweils 25 Leuchtdioden vorhanden (vgl. Abbildung 3.3). Bei diesen einzelnen Leuchtdioden können die Leuchteigenschaften, wie beispielsweise Leuchtintensität oder die RGB-Farbmischung individuell verändert werden. Dies ermöglicht ein breites Spektrum verschiedener Einstellungen und Leuchtkombinationen der Statusbeleuchtung, auch bei Einstellungen, welche eine Variation der Eigenschaften der Leuchtdioden innerhalb eines LED-Streifens benötigen.

Jede einzelne Leuchtdiode stellt in der weiteren Ansteuerung allgemein einen Pixel auf dem gesamten LED-Streifen dar. Die einzelnen Pixel auf den Streifen sind entsprechend der Ansteuerung individuell adressierbar. Eine direkte Ansteuerung ist mit dem Adafruit Neopixel-Protokoll möglich. Dieses kann nicht nur für die speziellen Neopixel-LEDs genutzt werden, sondern ermöglicht auch eine Ansteuerung von allgemein adressierbaren LEDs [24]. Jedem Pixel lassen sich die genannten Leuchteigenschaften individuell zuweisen, womit auch komplexe Lichteffekte in der Ansteuerung realisiert werden können. [25]

Für die Ansteuerung weisen die LED-Streifen neben den üblichen Anschlüssen für die Spannungsversorgung zwei weitere Datenleitungen auf. Eine Datenleitung wird hierbei als Eingangspin genutzt, während der weitere Datenpin als Ausgang geschaltet ist. Die einzelnen LEDs sind auf der Datenleitung seriell verschaltet. Durch weitere serielle Verschaltung der Datenpins an den Ein- und Ausgängen der LED-Streifen lassen sich mehrere LED-Streifen kombinieren. Die Adressierung der einzelnen Pixel beginnt mit der ersten Leuchtdiode nach dem Eingang des LED-Streifens und wird anschließend fortlaufend gezählt. Sind mehrere LED-Streifen verschaltet, so läuft die Adressierung der Leuchtdioden in den nachfolgenden LED-Streifen unverändert weiter (vgl. Abbildung 3.3).

Für die Ansteuerung der LED-Streifen werden digitale Pins, welche „digitalRead“ und „digitalWrite“-Funktionen zur Verfügung stellen, benötigt. In der bestehenden Roboterplattform ist die Steuerung mit einem Teensy-Mikrocontroller realisiert. Dieser hat die genannten digitalen Pins verbaut.

Auf dem Teensy-Mikrocontroller wird, ähnlich der Steuerung der Umgebungsbeleuchtung, ein Skript zur Ansteuerung der Status-LEDs ausgeführt. Die Struktur des Skripts orientiert sich hierbei ebenfalls an dem konventionellen Aufbau von Software für Mikrocontroller. Auch hier existiert entsprechend ein „Setup“-Bereich, welcher einmalig bei Start des Mikrocontrollers ausgeführt wird, sowie ein „Loop“-Bereich, welcher während der Laufzeit dauerhaft wiederholt wird. Weiterhin sind auch in diesem Skript die ROS-spezifischen Anwendungen vorhanden. [23]

Als Schnittstelle zu dem ROS-System, erhält die Steuerung der Statusbeleuchtung über das Topic „status_monitor/rgb_led_control“ eine speziell definierte Nachricht. Die Nachricht enthält Informationen über die Leuchteigenschaften, wie beispielsweise die RGB-Werte, der Leuchtmodus und die Leuchthelligkeit. Die Struktur der gesendeten Nachricht wird in Kapitel 4.3.3.2 detailliert dargestellt. Im Gegensatz zu der Umgebungsbeleuchtung werden die empfangenen Eigenschaften in der Callback-Funktion des Subscribers für die interne Weiterverarbeitung nur in Variablen gespeichert. Eine weitere Prozessierung der Daten findet nicht statt. [10]

Für die verschiedenen Leuchteffekte der Statusbeleuchtung sind in dem Skript diverse Funktionen definiert. Durch das interne Aufrufen dieser Funktionen werden die jeweiligen Leuchteffekte gestartet. Der Lichteffekt, welcher zu dem jeweils aktuellen Zeitpunkt dargestellt werden soll, ist durch den gesendeten Modus definiert. Anhand einer festen Zuordnung des Index der verschiedenen Modi zu den internen Leuchtfunktionen, wird nach dem Übermitteln des aktuellen Modus in der Nachricht die interne Leuchtfunktion aufgerufen.

Die anschließende tatsächliche Ansteuerung der einzelnen Leuchtdioden auf den LED-Streifen wird durch die Adafruit Neopixel-Bibliothek zur Verfügung gestellt. Das Steuern der LEDs wird hierbei durch definierte Funktionen in der Bibliothek stark vereinfacht [25]. Eine detaillierte Beschreibung der Funktionalitäten der Bibliothek sowie der Integration der Bibliothek in der Umsetzung der Skripte, wird in Kapitel 4.3.3.1 dargestellt.

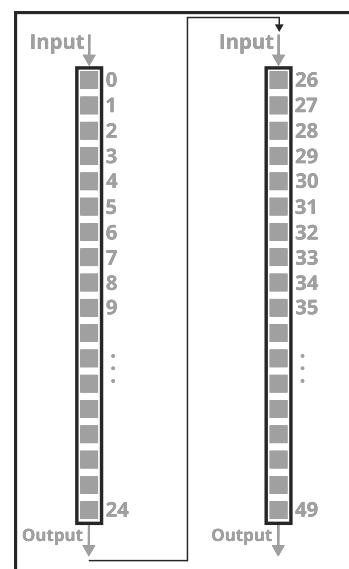


Abbildung 3.3: Statusbeleuchtung Untergrund, schematische Darstellung

4 Umsetzung Sensorkopf

In den nachfolgenden Kapiteln wird die Entwicklung der unterschiedlichen Bestandteile des neuen Sensorkopfes beschrieben. Hierbei werden jeweils die Bereiche der Mechanik und Konstruktion, der Elektrotechnik sowie der Informatik erläutert. Die beschriebenen Zusammenhänge und Umsetzungen stellen den finalen Bearbeitungsstand des neuen Sensorkopfes dar.

4.1 Mechanik

Die Umsetzung des Sensorkopfes in dem Bereich der Mechanik kann in 3 Bearbeitungsschritte unterteilt werden. So findet die Konstruktion, anschließend die Fertigung der Bauteile, sowie die abschließende Montage, statt. Nachfolgend werden diese dargestellt.

4.1.1 Konstruktion

Bei der Konstruktion wurde der Sensorkopf mit Greifer in verschiedene Untergruppen aufgeteilt. Diese setzen sich aus der Grundstruktur, dem Gehäuse, dem Greifer und dem Statusring zusammen.

4.1.1.1 Grundstruktur

Die Grundstruktur ist hierbei das Kernstück. An dieser werden alle anderen Untergruppen befestigt. Um eine höhere Stabilität und dennoch die Anforderung eines geringen Gewichts zu realisieren wird als Werkstoff Aluminium gewählt.

Der Vorteil einer Realisierung mit Aluminium ist zudem, dass im Gegensatz zu Kunststoffen kein elasto-plastisches Verhalten zu erwarten ist. In Folge findet mit der Zeit keine Verformung statt. Ein Ausgleichen von Fertigungsun genauigkeiten der weiteren Kunststoffelementen und somit eine ausreichende Positioniergenauigkeit ist hierdurch möglich.

Weiterhin sind in dem Bauteil ausreichende Aussparungen vorgesehen, sodass Kabeldurchführungen gewährleistet sind und eine weitere Gewichtsreduktion erreicht wird.

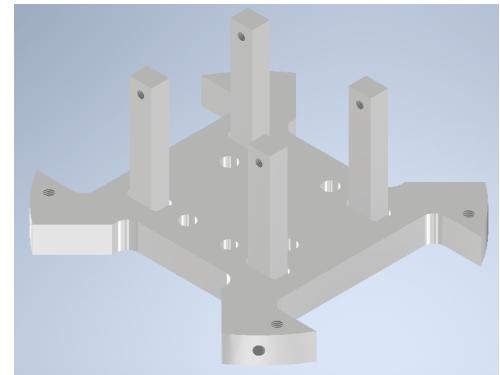


Abbildung 4.1: Grundstruktur

4.1.1.2 Gehäuse

Das Gehäuse wird in vier Gehäuseteile aufgeteilt. Es befinden sich zwei Gehäuseteile oberhalb der Grundstruktur, sowie zwei Gehäuseteile unterhalb.

An den Schalen oberhalb ist die Sensorik, wie die FishEye- sowie die 3D-Kamera befestigt. Weiterhin befindet sich auf der Seite der FishEye-Kamera ein Schalter mit dem die Stromversorgung des Greifers unterbrochen werden kann.

In den Halbschalen unterhalb der Grundstruktur sind die weiteren elektronischen Bauteile untergebracht. Hierbei ist auf einer Seite der USB-Hub montiert. Auf der gegenüberliegenden Seite befindet sich die Leiterplatte mit dem verbauten Teensy-Mikrocontroller.

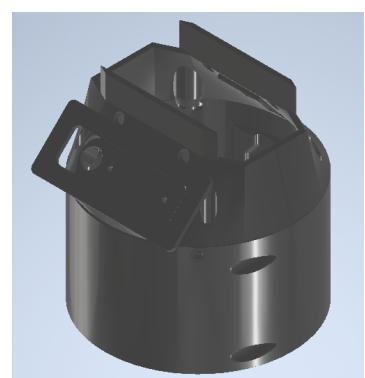


Abbildung 4.2: Gehäuse

Als Werkstoff wird für die Gehäuseteile Kunststoff verwendet. Dieser ist elektrisch nicht leitend und deshalb als Befestigung für die Elektronik gut geeignet. Zudem können die komplexen Strukturen mit dem 3D-Druck schnell und kostengünstig gefertigt werden. Auch das vergleichsweise niedrige Gewicht das bei einer Realisierung mit dem Werkstoff vorhanden ist, wirkt sich bei der Anzahl der Gehäuseelemente positiv auf das Gesamtgewicht aus.

4.1.1.3 Greifer

Die Konstruktion des Greifers orientiert sich in der Funktionsweise an der schon bestehenden Greifer-Konstruktion. Allerdings wurden entsprechend kleinere Optimierungen an der Führung der Greiferfinger vorgenommen.

Diese wurden zuvor durch Gewindestangen und Schrauben, welche in einem Langloch geführt sind, realisiert. Hierbei ist die Auflagefläche zwischen Führung (Langloch) und Führungsschlitten (Gewinde) sehr gering. Das Langloch aus Kunststoff stützt den Führungsschlitten somit wenig. Die Folge ist ein größeres Spiel.

Für die Verbesserung der Führung werden die Gewindestangen durch Stäbe ersetzt, welche ausschließlich an den Enden Gewinde aufweisen. Die bisher verbauten Schrauben sind durch Schrauben mit längsplan-abgedrehtem Absatz ersetzt. Hierdurch wird eine höhere Auflagefläche und somit bessere Stützeigenschaften erreicht.

Die Befestigung des Greifers an der Grundstruktur ist entsprechend einer schnellen Montage sowie Demontage realisiert. Hierbei ist die Grundplatte des Greifers auf die Grundstruktur aufgesteckt und wird von vier Schrauben seitlich fixiert. So kann durch das Lösen der Schrauben und Abstecken des Motors der gesamte Greifer schnell gewechselt werden.

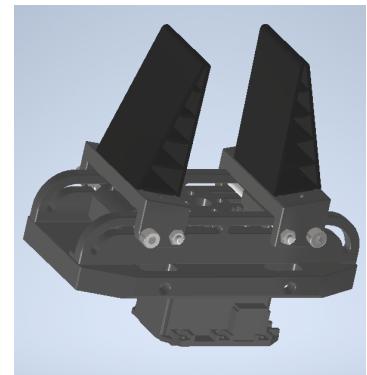


Abbildung 4.3: Greifer

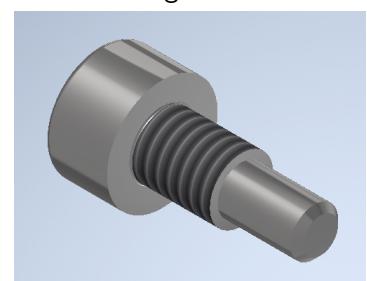


Abbildung 4.4: Führungs-schraube

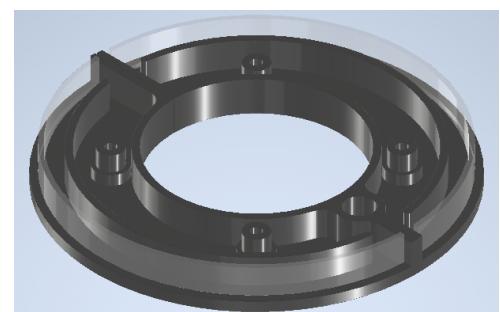


Abbildung 4.5: Statusring

4.1.1.4 Statusring

Die beiden Elemente des Statusrings werden unterhalb der Gehäusehalbschalen montiert und bestehen ebenfalls aus 3D-gedruckten Teilen. Neben dem schwarzen Kunststoff, welcher bei allen 3D-Druckteilen genutzt wird, ist ein Bereich aus transparentem Filament vorgesehen. Dieser wird als Diffusor für das Licht der LED-Streifen eingesetzt. Durch die verschiedenen Schichten, die bei der Fertigung entstehen, wirkt die Scheibe trüb. Bei dem Betrieb der Statusbeleuchtung entsteht hierdurch der Eindruck eines durchgehenden Lichtringes.

Bei der Konstruktion ist zudem vorgesehen, dass der Diffusor sich innerhalb der Fassung verspannt. Dieser ist somit form- und kraftschlüssig fixiert.

4.1.2 Fertigung

Bei der Fertigung wird zwischen der Fertigung von den Aluminiumelementen und den Kunststoffelementen unterschieden. Die Aluminiumelemente werden allgemein durch eine zerspanende Bearbeitung gefertigt. Die Kunststoffelemente werden durch additive Fertigungsverfahren hergestellt.

4.1.2.1 Aluminiumelemente

Die Grundstruktur ist vollständig aus Aluminium gefertigt.

Diese besteht aus einem Flansch und vier Leisten, welche als Abstandhalter dienen.

Für die Herstellung des Flansches wurde zunächst der Grundkörper an der Drehmaschine gedreht. Hierfür wurde eine Weiler Praktikant Drehmaschine genutzt. Bei dieser ist eine Fertigung mit einer Genauigkeit von bis zu 0,01 mm möglich.

Anschließend wurden die Aussparungen für die Kabelführung, sowie diverse Taschen für eine einfache Positionierung der Leisten, ausgefräst. Hierbei wurde eine Deckel FP2 Fräsmaschine eingesetzt. Abhängig von der Bedienung können hierbei auch Genauigkeiten von 0,01 mm erreicht werden.

Weiterhin wurden die Abstandsleisten ebenfalls mit der identischen Maschine auf die entsprechenden Maße gefräst. Um ein besseres Einspannen zu erreichen, sowie eine geringe Verletzungsgefahr zu gewährleisten, wurde anschließend von allen Kanten der Grat entfernt.

Weiterhin sind alle Bohrungen und Gewinde in dem Bauteil mit einem Höhenreißer angerissen, gekörnt und anschließend mit einer AEG Flott Säulenbohrmaschine gebohrt.



Abbildung 4.6: Grundstruktur aus Aluminium

4.1.2.2 Kunststoffelemente

Für die Herstellung der gesamten Kunststoffteile wird der 3D-Druck verwendet. Die CAD-Daten der Bauteile müssen hierfür zunächst in STL-Daten umgewandelt werden. Hierbei werden die Flächen der Volumenkörper in den CAD-Daten in einzelne Dreiecke aufgeteilt.

Zu beachten ist, dass die Dateien mit korrekten Einheiten exportiert werden. Weiterhin ist vorab die maximale Kantenlänge zu definieren. Diese wurden manuell verringert, sodass die geschwungenen Flächen in den Bauteilen besser dargestellt werden. Dennoch sind bei den ersten gedruckten Bauteilen vereinzelt die Dreiecksflächen zu erkennen.

Die STL-Daten können anschließend mit dem Prusa Slicer in die direkten Maschinenbewegungen des 3D-Druckers umgewandelt werden. Wie beschrieben (vgl. Kapitel 2.1.3) bestehen auch hierbei verschiedene Einstellmöglichkeiten. Bei den meisten Möglichkeiten können die Standardeinstellungen übernommen werden. Um eine höhere Stabilität zu erreichen, wurde jedoch die Menge an Infill erhöht.



Abbildung 4.7: Kunststoffbauteil mit Heatinserts

Bei der Fertigung wurden nicht alle benötigten Gewinde direkt in die Kunststoffelemente geschnitten. Es wurden hierbei Heatinserts verwendet. Die Messing-Gewindegülsen können mit

dem Lötkolben in die 3D-Druckteile eingeschmolzen werden. Diese sind robuster als direkt geschnittenen Gewinde.

Zu dem Fertigstellen der Bauteile müssen diese entgratet und einzelne Bohrungen weiter aufgebohrt werden. Weiterhin mussten Aussparungen teils größer gefeilt werden. Die nachträglichen Anpassungen wurden sofern möglich im 3D-Modell übernommen, um bei einer erneuter Fertigung einen niedrigen Arbeitsaufwand sicherzustellen.

4.1.3 Montage

Die Montage sowie die mögliche Demontage der einzelnen Bauteile zu dem gesamten Sensorkopf kann flexibel erfolgen. Ein vollständiges Entfernen aller Bauteile, um einzelne Komponenten zu entfernen oder Zugang zu einzelnen Schrauben zu erhalten, ist nicht notwendig. Nachfolgend wird die einzelne Montage beschrieben.

4.1.3.1 Sensorkopf

Um den vollständigen Sensorkopf befestigen zu können muss der Greifer demontiert sein. Der Sensorkopf kann anschließend auf die Anschlussfläche am Roboterarm angesteckt werden. Die Schrauben werden von oben festgezogen.

Bei der Montage ist zu beachten, dass keine Kabel zwischen Roboterarm und Anschlussflansch eingeklemmt werden.

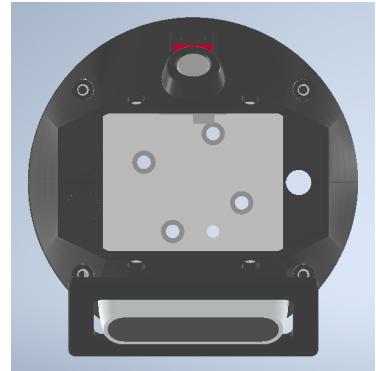


Abbildung 4.8: Montage Sensorkopf

4.1.3.2 Greifer

Bei Arbeiten am Greifer ist zu Beachten, dass der rote Kippschalter deaktiviert ist. Der Greifer ist somit stromlos geschaltet.

Zunächst muss der Anschlussstecker in den Motor eingesteckt werden. Anschließend kann die vollständige Greifer-Einheit am Sensorkopf aufgesteckt werden.

Für die Befestigung müssen die vier Befestigungsschrauben von den Seiten (vgl. Abbildung 4.9 rot markiert) befestigt werden. Die Montage ist abgeschlossen.

Durch das Aktivieren des Schalters ist die Stromversorgung zu dem Motor hergestellt und der Greifer ist einsatzbereit.



Abbildung 4.9: Montage Greifer

4.1.3.3 Statusring

Die beiden Halbringe des Statusrings werden unten an den Gehäusehalbschalen mit je zwei Schrauben befestigt. Die Stromversorgung der LED-Strips erfolgt dabei über Kabel, die durch Aussparungen in der Unterseite der Gehäuse gesteckt werden.

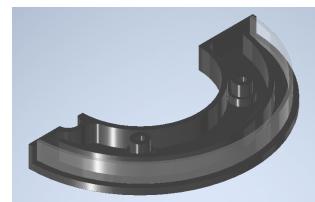


Abbildung 4.10: Montage Statusring

4.1.3.4 Gehäuse USB-Hub

Bevor der USB-Hub in das Gehäuse eingebaut werden kann, muss der USB-Stecker in den Hub eingesteckt werden. Danach kann der Hub mit den Klemmelementen in das Gehäuse gespannt werden. Im Anschluss kann das Gehäuse mit zwei Schrauben am Anschlussflansch festgeschraubt werden. Falls die Gehäusehalbschale der Leiterplatte montiert ist, können beide mit vier Schrauben verbunden werden.

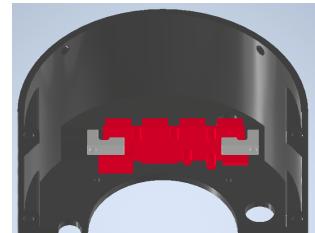


Abbildung 4.11: Montage USB-Hub

4.1.3.5 Gehäuse Leiterplatte

Für einer leichtere Montage kann man die Stecker vor dem Befestigen der Leiterplatte anstecken. Die Leiterplatte wird dann mit drei Schrauben in das Gehäuse geschraubt. Die restliche Montage funktioniert wie die des USB-Hub Gehäuses.

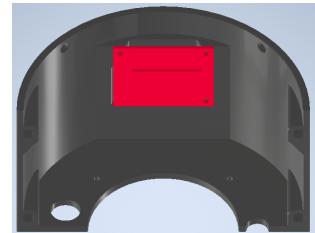


Abbildung 4.12: Montage Leiterplatte

4.1.3.6 Gehäuse FishEye-Kamera

Im Gehäuse der FishEye-Kamera wird die FishEye-Kamera mit Schrauben befestigt, anschließend wird der Stecker der Kamera angesteckt. Dann kann der Wippschalter in die Aussparung gesteckt werden, bis er einrastet. Wenn der Schalter angeklemmt ist, kann das Gehäuseelement von oben mit zwei Schrauben an der Grundstruktur befestigt werden.

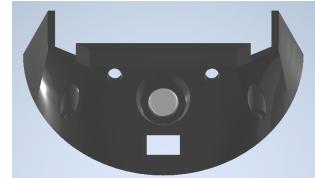


Abbildung 4.13: Montage Fish-Eye-Kamera

4.1.3.7 Gehäuse 3D-Kamera

An dem Gehäuse der 3D-Kamera werden die Kamera und die Umgebungsbeleuchtung befestigt. Die Montage erfolgt identisch zu dem FishEye-Kamera-Gehäuse.



Abbildung 4.14: Montage 3D-Kamera

4.1.4 Anpassungen

Nach der Konstruktion und der Fertigung der Teile sind im Verlauf des Projektes Änderungen notwendig gewesen. Der Roboterarm, für welchen die Konstruktionen entworfen sind, wurde nicht eingesetzt. Statt dem LARA 3 Roboterarm von Neura wird der ReBeL Cobot 6 DOF von Igus verwendet. Zudem mussten bei der Umgebungsbeleuchtung Änderungen vorgenommen werden, um konstruktiv die Fehleranfälligkeit im Bereich der Elektrotechnik zu minimieren.

4.1.4.1 Anpassungen für alternativen Roboterarm

Der alternative Roboterarm hat veränderte Anschlussmaße, womit eine Montage ohne Nachbearbeitung nicht möglich ist.

Bei dem neuen Arm ist das vordere Segment kürzer und der Teilkreis für die Befestigungsschrauben unterscheidet sich. Um dennoch eine Montage zu ermöglichen, wird ein Distanzstück (vgl. Abbildung 4.16 grün markiert) zwischen Roboterarm und Sensorkopf eingesetzt. Diese dient zusätzlich als Bohrschablone um den passenden Teilkreis in den Anschlussflansch zu bohren. Durch die kurzfristige Änderung des Roboterarms musste das Distanzstück 3D-gedruckt werden. Das Verformen des Kunststoffes mit der Zeit ist somit nicht mehr ausgeschlossen. Die Spannung der Befestigungsschrauben lässt nach und das Bauteil kann einen Defekt aufweisen.

Zusätzlich ist der Durchmesser des vordersten Segments größer als bei dem Neura Arm, weshalb der Sensorkopf nicht einfach aufgesteckt werden kann. Als Lösung wurde die Öffnung des Sensorkopf durch manuelles schleifen vergrößert.

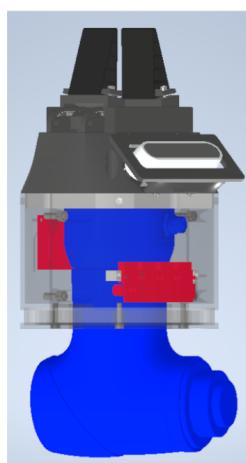


Abbildung 4.15: Sensorkopf Neura

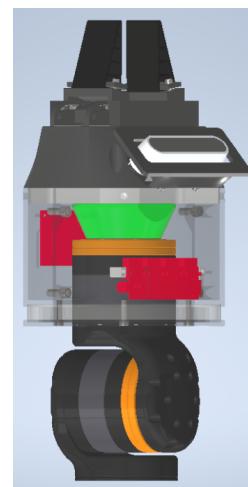


Abbildung 4.16: Sensorkopf Igus

4.1.4.2 Anpassungen Umgebungsbeleuchtung

Bei der Umgebungsbeleuchtung führte die Anordnung der LED-Strips zu Problemen an den Lötstellen. Diese waren U-förmig um die 3D-Kamera angeordnet. Aufgrund der U-Form konnten die LED-Streifen nur schwierig gefertigt werden. Zudem bestand die Gefahr, dass die Lötstellen brechen oder abreißen.

Als Lösung wurde die 3D-gedruckte Halterung angepasst, sodass ein Streifen der Beleuchtung über der Kamera und ein Streifen unterhalb ist. Die Verbindung der LED-Streifen erfolgt nun über Stecker.

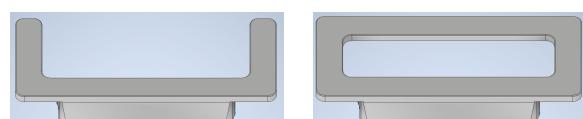


Abbildung 4.17: Vorher Abbildung 4.18: Nachher

4.2 Elektrotechnik

Bei der Umsetzung der Elektronik werden unterschiedliche Bauteile in dem Sensorkopf verwendet. Diese müssen weiterhin entsprechend der jeweiligen Spezifikationen mit Spannung versorgt werden. Nachfolgend werden zunächst die in dem Sensorkopf verbauten Bauteile vorgestellt. Anschließend erfolgt eine Beschreibung der direkten Stromversorgung sowie die Darstellung der Umsetzung. Die Spannungsversorgung zu dem Sensorkopf erfolgt über ein dreiadriges Kabel, welches entlang des Roboterarms geführt wird. Dabei ist eine Ader für die 5 V und eine Ader für die 12 V Versorgungsspannung vorhanden. Weiterhin existiert eine Ader für die gemeinsame Masseleitung. Für die Datenleitungen wird ein USB-B-Kabel mit dem Spannungskabel verlegt. Die Spannungsverteilung an die Bauelemente erfolgt über eine zentral liegende Lochrasterplatine, wofür zunächst die Kabelverbindungen zwischen Platine und den jeweiligen Bauteilen hergestellt werden müssen.

4.2.1 Bauteile und Beschaffung

Bei der Auswahl der einzelnen Bauteile, sowie der anschließenden Erstellung der Bauteillisten wurde sich an den Komponenten des bereits vorhandenen Greifers orientiert.

4.2.1.1 LOGILINK UA0170 USB 3.0 Hub

Als USB-Hub wird ein USB 3.0 Hub mit vier Ports von Logilink verwendet. Dieser ist mit seiner maximalen Übertragungsgeschwindigkeit von bis zu 5 GBit/s ausreichend schnell für die benötigten Datenübertragungen. Aufgrund der kompakten Bauweise des Sensorkopfes ist die Verwendung von einem USB-Hub ausreichend – im Vergleich zu den zwei benötigten bei der Realisierung in der alten Roboterplattform.

Um den USB-Hub im Gehäuse verbauen zu können wurde das Hub-Gehäuse entfernt. Die Spannungsversorgung des aktiven USB-Hubs erfolgt über die Lochrasterplatine. Dazu wurden zwei von der Platine kommende Adern – eine für die 5 V Versorgungsspannung, eine weitere für die Masse – auf die Rückseite der USB-Hub-Platine gelötet. Für den Datenanschluss kann das USB-B-Kabel direkt verwendet werden.

Um Komplikationen im Falle von zwei anliegenden Versorgungsspannungen zu verhindern, wurden im USB-B-Kabel die beiden Adern für VBUS (+ 5 V) und GND durchtrennt. Alle Adern sind anschließend mithilfe von Schrumpfschläuchen vollständig isoliert worden. [26] Die Abbildung 4.19 zeigt den USB-Hub, welcher bei dem Elektronikfachmarkt reichelt elektronik verfügbar ist.



Abbildung 4.19: USB-Hub [26]

4.2.1.2 Servomotor Dynamixel MX-28R

Die Bewegung der Greiferfinger erfolgt über einen programmierbaren Servomotor des Typs MX-28R der Firma Dynamixel. Der Betriebsspannungsbereich liegt zwischen 10 V und 14,8 V. Verwendet wird die empfohlene Nennspannung von 12 V.

Der Servomotor ermöglicht eine 360° Positionsregelung ohne Totbereich und besitzt dabei eine präzise Auflösung. Die Auflösung liegt bei einem Schrittewinkel von 0,088°, dies entspricht 4,096 Schritten pro Umdrehung. Eine präzise Bewegungssteuerung ist somit möglich. Die Datenübertragung liegt in dem Geschwindigkeitsbereich von 8000 bps bis etwa 3 Mbps. [27]

Abbildung 4.20 zeigt den verwendeten Servomotor. Für den Anschluss wurde ein Kabel gefertigt, welches mit zwei Adern die Versorgungsspannung und die Masse zur Lochrasterplatine bereitstellt. Weiterhin besteht eine Datenleitung zu dem USB-Hub. In der Leitung für die Versorgungsspannung ist zudem ein Ein-Aus-Kippschalter verbaut. Mit diesem kann im Falle von Komplikationen an dem Sensorkopf, sowie für Arbeiten an dem Greifer, die Versorgungsspannung zum Servomotor getrennt werden.



4.2.1.3 3D Intel RealSense D435 Tiefenkamera

Als Teil des Kamerasystems wird die 3D Intel RealSense Tiefenkamera des Typs D435 verwendet. Diese bietet mit ihrem weiten Sichtfeld im Bereich der Robotik die benötigte Lösung für die Aufgaben der Roboternavigation und Objekterkennung. Die verwendete Kombination aus Infrarot- und RGB-Sensoren erfasst präzise Tiefeinformationen. Dadurch können hochauflösende Tiefenkarten mit einer Genauigkeit von bis zu einem Millimeter und einer Reichweite von bis zu 10 Metern erstellt werden. Die hohe Lichtempfindlichkeit der globalen Shutter Sensoren ermöglicht zudem das Navigieren des Roboters bei schlechten Lichtverhältnissen. [28]

Die Spannungsversorgung und Datenübertragung der 3D Kamera erfolgt über einen USB 3.0 Anschluss. Die definierte Leistung von 4,5 W, respektive einem Strom von 900 mA, wird somit über einen Anschluss an die 5 V Spannung des USB-Hubs erreicht. Die Abbildung 4.21 stellt die 3D Kamera von vorne dar. [28]



Abbildung 4.21: Intel RealSense [28]

4.2.1.4 180° Fisheye Kamera

Für den zweiten Teil des Kamerasystems wird eine 180° Fisheye Kamera verwendet. Diese wird an der gegenüberliegenden Seite zur 3D Intel RealSense Tiefenkamera zur Kollisionsüberwachung angebracht. Über das Ultraweitwinkelobjektiv wird Licht von einem sehr weiten Bereich gesammelt, wodurch ein extrem großes Sichtfeld abgedeckt wird. Die Fisheye Kamera benötigt eine Versorgungsspannung von 5 V. Bei der Umsetzung wurde ein dreidriges Kabel mit Versorgungsspannung, Masse und Datenleitung für einen Anschluss an dem USB-Hub gefertigt. [29] Die Kamera wird in Abbildung 4.22 dargestellt.



Abbildung 4.22: Fisheye-Kamera [29]

4.2.1.5 SK6812 RGB LED

Für die Umgebungs- und die Statusbeleuchtung sollen der gleiche Typ von LEDs verwendet werden. Die Umgebungsbeleuchtung ist um die Intel RealSense Tiefenkamera angeordnet. Die Statusbeleuchtung zeigt unter anderem den derzeitigen Arbeitsmodus des Roboters an. Für die Statusbeleuchtung

unterhalb der neuen Roboterplattform werden digitale, adressierbare RGB LEDs vom Typ SK6812 verwendet. Aufgrund der guten Verfügbarkeit sowie der leichten Austauschbarkeit der LED-Klebestreifen, wurden diese LEDs auch für die beiden Beleuchtungsarten im Sensorkopf gewählt.

Die LED-Streifen sind jeweils über eine Datenleitung mit dem Mikrocontroller auf der Lochrasterplatine und über eigens gefertigte Stecker mit der 5 V Anbindung auf der Lochrasterplatine verbunden. Die Abbildung 4.23 zeigt die verwendeten LED-Streifen, welche von MOUSER ELECTRONICS besorgt wurden. Die Streifen besitzen über die Länge von 100 mm eine Anzahl von 15 LEDs. [30]



Abbildung 4.23: LED-Streifen
[30]

4.2.1.6 Teensy 4.0

Für die Ansteuerung der Umgebungs- und Statusbeleuchtung wird der in Abbildung 4.24 dargestellte Teensy 4.0 verwendet. Dadurch wird der bisherige Arduino Nano ersetzt und es kann auf den PMIC verzichtet werden. Der verwendete Teensy wurde in Kapitel 2.2.3 bereits genauer vorgestellt. [31]



Abbildung 4.24: Teensy 4.0
[31]

4.2.2 Stromlaufplan, USB-Hub und Teensy-Anschlüsse

Für die Spannungsversorgung der Bauteile werden ein 5 V und ein 12 V Spannungsniveau benötigt. Alle Bauteile müssen weiterhin an einer gemeinsamen Masse angeschlossen sein. Die folgende Abbildung 4.25 zeigt den entworfenen Stromlaufplan:

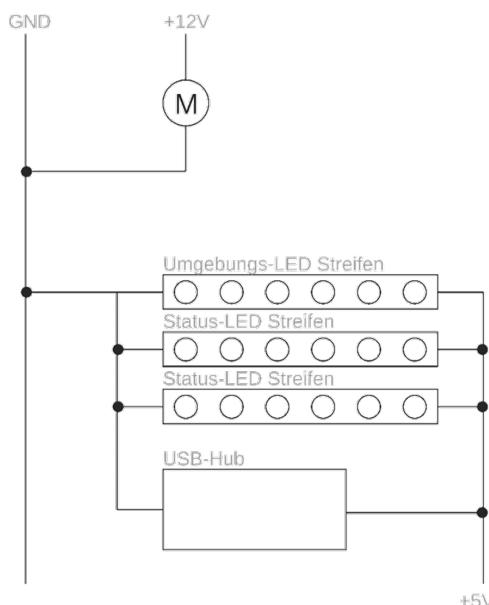


Abbildung 4.25: Stromlaufplan Sensorkopf

Die 12 V Versorgungsspannung ist nur mit dem Dynamixel Servomotor verbunden. An der 5 V Spannung sind in einer Parallelschaltung die drei benötigten LED-Streifen – einen für die Umgebungsbeleuchtung und zwei weitere für die Statusbeleuchtung – und der aktive USB-Hub angeschlossen. Nachfolgend wird der Stromlaufplan um die Anschlüsse am USB-Hub erweitert. Dies wird in der folgenden Abbildung 4.26 dargestellt:

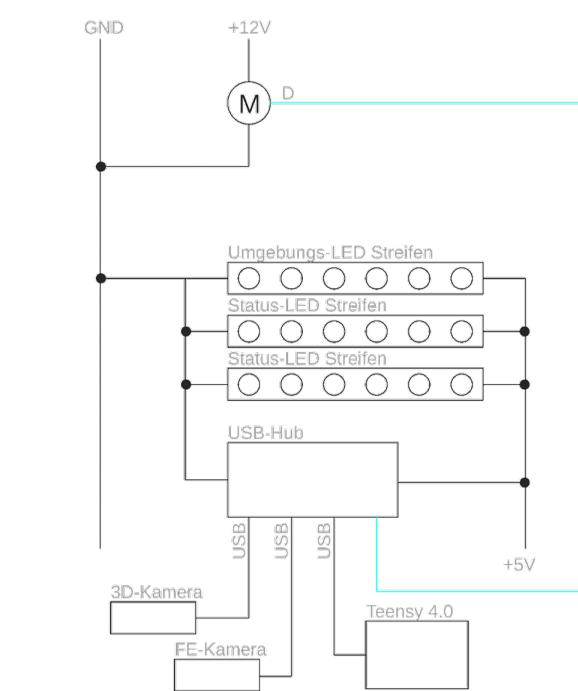


Abbildung 4.26: Anschlüsse USB-Hub

Die Intel RealSense 3D Kamera, die Fisheye Kamera und der Teensy 4.0 sind über USB-Kabel mit dem USB-Hub verbunden. Über diese Verbindung findet die Spannungsversorgung mit 5 V und die Datenübertragung statt. Der Dynamixel Servomotor besitzt zudem eine Verbindung zum USB-Hub über eine Datenader. Der USB-Hub ist wie bereits beschrieben für die Datenübertragung über ein USB-B-Kabel mit der weiteren Roboterplattform verbunden.

Die Ansteuerung der LED-Streifen findet über den Teensy 4.0 statt. In Abbildung 4.27 werden die verwendeten Pins des Teensy für die benötigten Datenleitungen zu den LED-Streifen dargestellt.

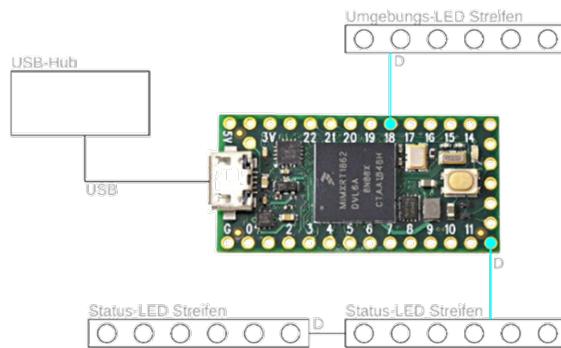


Abbildung 4.27: Anschlüsse Teensy

Aufgrund der Steckerplatzierungen auf der Lochrasterplatine (vgl. Kapitel 4.2.3), werden für die Datenleitungen die Pins 12 und 18 verwendet.

Wichtig für die Verkabelung der LED-Streifen ist es, dass die Spannungsversorgung über eine Parallelschaltung stattfindet, die Datenleitung bei zusammengehörigen LED-Streifen – in Abbildung 4.27 an den beiden Status-LED Streifen zu sehen – zu einer Reihenschaltung führt. Dadurch werden die LEDs der beiden Status-LED Streifen in aufsteigender Reihenfolge bei Null beginnend durchnummieriert (vgl. Kapitel 4.3.3.1).

4.2.3 Umsetzung Lochrasterplatine

Die vorab dargestellten Stromlaufpläne werden durch manuelles Löten auf einer Rochlasterplatine realisiert. Aufgrund der hohen Verfügbarkeit sowie der möglichen schnellen Umsetzung wurde diese Lösung einer gefertigten Platine vorgezogen.

Auf der Lochrasterplatine soll der Teensy-Mikrocontroller sowie alle benötigten Steckerverbindungen zu den Bauteilen montiert werden. In Abbildung 4.28 wird die gefertigte Lochrasterplatine dargestellt.

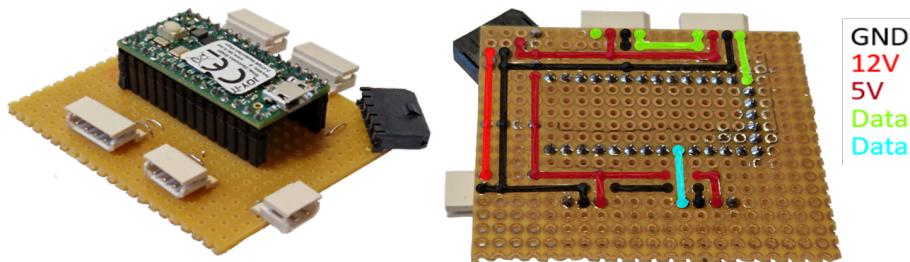


Abbildung 4.28: Lochrasterplatine

Die Lochrasterplatine wurde so konzipiert, dass die Bahnen so nahe wie möglich aneinander liegen. Zudem wurden die Stecker möglichst so angelötet, dass sie in Richtung der Bauteile zeigen.

Der schwarze Stecker nimmt das dreiadrige Spannungskabel auf, welches den Roboterarm von der Verteilerplatine der Roboterplattform zum Sensorkopf hochgelegt wurde. Der Teensy sitzt in der Mitte der Lochrasterplatine. Um ihn herum wurden zunächst die gemeinsamen Masseleitungen – in Abbildung 4.28 in schwarz dargestellt – als Parallelschaltung gelegt. Die hellrote Leitung versorgt den Stecker für den Dynamixel Servomotor mit der empfohlenen Spannung von 12 V. Die dunkelroten Leitungen stellen das in Parallelschaltung verlegte 5 V System dar. Dabei sind die beiden Stecker auf Seiten der Versorgungsspannung für die Statusbeleuchtung. Vom Pin 12 des Teensy's verläuft die in grün eingezeichnete Datenleitung. Wie zu erkennen ist, werden die Datenleitungen der beiden Stecker in Reihe geschaltet. Durch den identischen Steckeraufbau ist es zudem egal, welcher Status-LED-Stecker in welche Buchse gesteckt wird.

Gegenüberliegend befinden sich noch zum einen der Stecker für den USB-Hub sowie der Stecker für die Umgebungsbeleuchtung. Von Pin 18 des Teensy ist in blau die Datenleitung zur Umgebungsbeleuchtung dargestellt.

Bei der Auswahl der Stecker und Buchsen wurde darauf geachtet, dass der Stecker des Motors nicht in eine Buchse des 5 V Systems gesteckt werden kann. Zudem ist der USB-Hub Stecker der einzige mit drei Pins. Die LED-Stecker könnten theoretisch vertauscht werden. Es ist somit darauf zu achten, dass die von unten durch den Sensorkopf geführten Statusbeleuchtungskabel in die unteren Buchsen der Lochrasterplatine gesteckt werden und das von oben geführte Umgebungsbeleuchtungskabel in die obere Buchse.

4.3 Informatik

Anhand der Umsetzung der Beleuchtung in der bestehenden Roboterplattform wurde die Beleuchtung für den neuen Sensorkopf entwickelt. Diese wird nachfolgend detailliert beschrieben. Hierbei wird auf das allgemeine Leuchtkonzept eingegangen sowie die grundlegenden Funktionalitäten und Änderungen in der Ansteuerung dargestellt. Weiterhin wird allgemein auch bei der neuen Umsetzung zwischen einer Umgebungsbeleuchtung und einer Statusbeleuchtung unterschieden. Es sind jedoch diverse Änderungen in der Hardware vorhanden, welche eine technische Unterscheidung der Beleuchtung im neuen Sensorkopf nicht mehr notwendig macht. Es wird dennoch für jeweils beide Kategorien die Umsetzung differenziert dargestellt.

Im Allgemeinen besteht in dem Projekt die Anforderung, die Ansteuerung der LEDs in dem Sensorkopf an einem zentralen Hardware-Punkt zu bündeln. Hierfür soll für die Steuerung beider LED-Kategorien ein Teensy-Mikrocontroller verwendet werden. Auf diesem einzelnen Mikrocontroller kann weiterhin nur ein Skript, welches in den Flash-Speicher geschrieben wird, ausgeführt werden. Entsprechend müssen die in der vorherigen Plattform auf mehrere Mikrocontroller aufgeteilten Funktionalitäten zur Ansteuerung der LEDs in einen Programmablauf zusammengefasst werden.

4.3.1 Umgebungsbeleuchtung

Auch an dem neuen Sensorkopf soll eine Umgebungsbeleuchtung vorhanden sein. Die Umgebungsbeleuchtung soll entsprechend auch bei der Entwicklung des neuen Sensorkopfes das direkte Umfeld möglichst vollständig ausleuchten. Weiterhin ist hierbei die Variation mit der Beleuchtungshelligkeit vorgesehen. Das Ausleuchten mit maximaler Leuchttintensität sowie das vollständige Deaktivieren der Umgebungsbeleuchtung muss weiterhin möglich sein. Für die Lichtfarbe und weitere Eigenschaften der Beleuchtung sind keine direkten Anforderungen vorhanden.

Bereits in frühen Entwicklungsphasen des Projektes wurde eine Änderung der Hardware der Umgebungsbeleuchtung festgelegt. So werden bei der neuen Umgebungsbeleuchtung individuell adressierbare LED-Streifen eingesetzt. Die LED-Streifen sind außerhalb des Sensorkopfes in einem für Außenstehende sichtbaren Bereich angebracht. Die Umsetzung mit der geänderten LED-Hardware hat im Allgemeinen keine negativen Auswirkungen auf die Ausleuchtung des Umfeldes. Vielmehr zeigen sich für die Umsetzung in dem gesamten Projekt sowie in der Ansteuerung der LED-Streifen diverse Vorteile. Es können im Bereich der Elektrotechnik diverse Bauteile eingespart werden. So entfällt der PMIC, eine kompaktere Bauweise der Komponenten innerhalb des Sensorkopfes ist somit möglich. Bei der Ansteuerung können die LED-Streifen somit direkt über das Neopixel-Protokoll gesteuert werden. In Folge bestehen mehr Möglichkeiten für Variationen in den diversen Leucht-Eigenschaften der LEDs. Weiterhin ist eine vereinheitlichte Umsetzung der Ansteuerungen der diversen LED-Streifen in dem neuen, für beide Beleuchtungskategorien genutzten, Teensy-Skript gegeben.

Die LED-Streifen weisen, identisch zu den LEDs der vorhandenen Statusbeleuchtung, vier Anschlüsse auf. Neben den zwei Anschlüssen zur dauerhaften Spannungsversorgung existieren die zwei weiteren Daten-Eingangs- und Ausgangs-Pins. Für die Ansteuerung der adressierbaren LEDs werden an dem Teensy weiterhin digitale Pins mit den „digitalWrite“- und „digitalRead“-Funktionen benötigt [32]. Im Vergleich zu der bestehenden Realisierung (vgl. Kapitel 3.3.1), ist eine Anpassung der Ausgangspins notwendig. Dies stellt eine weitere Anforderung an den Bereich der Elektrotechnik dar (vgl. Kapitel 4.2.2). Die tatsächliche Steuerung der LED-Streifen kann nun direkt über das Adafruit Neopixel-Protokoll erfolgen.

4.3.2 Statusbeleuchtung

Der neue Sensorkopf soll eine eigene Statusbeleuchtung aufweisen. Die Statusbeleuchtung wird ebenfalls mit adressierbaren LED-Streifen umgesetzt. Hierbei werden dieselben LED-Streifen-Typen wie bei der Umgebungsbeleuchtung genutzt. Die Eigenschaften sind somit ebenfalls identisch zu der Umgebungsbeleuchtung. Angesteuert werden die LED-Streifen über den gleichen Teensy-Mikrocontroller, jedoch über einen separaten Ausgangspin als bei der Umgebungsbeleuchtung.

Während die Statusbeleuchtung im Untergrund der neuen Roboterplattform für außenstehende Personen nicht direkt sichtbar ist, wird die neue Statusbeleuchtung in dem Sensorkopf als sichtbares Design-Element realisiert. Der Betrachter des Sensorkopfes kann hierbei die einzelnen Pixel der LED-Streifen von außerhalb erkennen. Diese sind einzig durch eine diffuse Sichtabdeckung gedämpft.

Grundsätzlich soll die Statusbeleuchtung bei der Realisierung in dem Sensorkopf verschiedene Lichteffekte anhand der Zustände des Roboters darstellen. Eine optisch ansprechende Realisierung der Effekte, welche weiterhin mit den Darstellungen der Statusbeleuchtung im Untergrund des Roboters harmonieren, ist entsprechend notwendig. Im Vergleich zu der Statusbeleuchtung im Untergrund der Roboterplattform werden jedoch durch die Sichtbarkeit der einzelnen LED-Pixel weitere Leuchteffekte in der Umsetzung möglich.

Weiterhin sind bei der Statusbeleuchtung im Untergrund, unverändert zu der bestehenden Roboterplattform (vgl. Kapitel 3.3.2), zwei parallele LED-Streifen angebracht. Auch bei der Statusbeleuchtung in dem neuen Sensorkopf sollen zwei verbundene LED-Streifen zum Einsatz kommen (vgl. Kapitel 4.2.2). Diese werden jedoch als Ringlicht realisiert. Die LED-Streifen sind um den zylinderförmigen Sensorkopf konstruiert (vgl. Kapitel 4.1.1.4). Im Querschnitt ergibt sich somit eine kreisförmige Form der Statusbeleuchtung. Durch die Änderung der Form ergeben sich weitere mögliche Leuchteffekte, welche umgesetzt werden können. Dennoch existieren diverse Lichteffekte der Statusbeleuchtung im Untergrund, bei welchen eine Übertragung auf die Statusbeleuchtung in dem Sensorkopf aufgrund der geänderten Form nicht direkt möglich ist. Die betroffenen Lichteffekte müssen durch neue harmonisierende Effekte ersetzt werden.

4.3.3 Ansteuerung Teensy-Mikrocontroller

Wie bereits zu Beginn beschrieben, besteht die Anforderung, die Ansteuerung der Beleuchtungen im neuen Sensorkopf an einer zentralen Stelle zu realisieren. Es soll der Teensy-Mikrocontroller verwendet werden. Bei der Steuerung der diversen LED-Streifen handelt es sich jedoch um keine rechenintensive Anwendung. Die Nutzung eines Teensy-Mikrocontrollers ist demnach mit der hohen Verfügbarkeit für das Projekt zu begründen.

Auf dem Teensy wird ein Skript ausgeführt. Dieses ist in den Flash-Speicher geschrieben und wird bei aktiver Stromversorgung des Mikrocontrollers automatisch gestartet. In dem Skript werden die Funktionen zur Ansteuerung der beiden Beleuchtungskategorien umgesetzt.

Die allgemeine Struktur des Skripts zur Ansteuerung auf dem Teensy folgt der bekannten Gliederung in einen „setup“-Bereich sowie in einen „loop“-Bereich. Der setup-Bereich wird bei der Aktivierung des Mikrocontrollers einmalig ausgeführt. [23] Hier werden entsprechend einmalige Initialisierungsaufgaben und Abläufe definiert. Der loop-Bereich wird während der Laufzeit kontinuierlich wiederholt. Entsprechend werden in diesem Bereich die eigentlichen Funktionalitäten definiert. Neben den beiden Bereichen können weiterhin beliebig viele Funktionen definiert werden. Diese lassen sich durch Funktionsaufrufe während der Laufzeit aktivieren. [23]

Durch den Teensy müssen entsprechend die adressierbaren LED-Streifen direkt angesteuert werden. Das für die LEDs verwendete Adafruit-Neopixel-Protokoll kann durch die Funktionen der Adafruit-Neopixel-Bibliothek vereinfacht genutzt werden [25]. Weiterhin werden in dem Skript die definierten Schnittstellen zu dem ROS-System benötigt. Sowohl für die Umgebungsbeleuchtung als auch für die Statusbeleuchtung sollen die identischen Daten empfangen werden, wie bei der bereits bestehenden Roboterplattform.

4.3.3.1 Adafruit NeoPixel-Bibliothek

Die Adafruit NeoPixel-Bibliothek ist ursprünglich eine Arduino-Bibliothek, welche vereinfachte Funktionen zur Ansteuerung adressierbarer LED-Beleuchtung über das Adafruit NeoPixel-Protokoll bereitstellt. Dennoch wird auch der Teensy-Chipsatz unterstützt, weshalb die Bibliothek auch bei der Umsetzung in diesem Projekt genutzt werden kann [25]. Grundsätzlich wird bei der Entwicklung des Skriptes die NeoPixel-Bibliothek in dem PlatformIO Projekt innerhalb des „lib“-Ordners verwendet. Um die Funktionen aus der Bibliothek in dem Skript nutzen zu können, muss die entsprechende Header-Datei eingebunden werden. [33]

Für den Zugriff auf die LED-Streifen, während dem Ablauf des Skripts, müssen diese zunächst spezifiziert werden. Es wird die Anzahl der Leuchtdioden auf dem LED-Streifen, der auf dem Teensy verwendete Ausgangspin sowie der Typ der verwendeten LED-Streifen als Informationen benötigt. Mit diesen Informationen wird für jeden individuell genutzten LED-Streifen ein Neopixel-Objekt definiert. Dies erfolgt durch einen Aufruf des Konstruktors der Neopixel-Klasse, bei welchem das Objekt weiterhin eine Benennung erhält [33], [34]. Die benötigten Informationen werden hierbei als Parameter übergeben, wobei in der Struktur des Skripts die Anzahl der Pixel sowie die Nummer des Ausgangspins an dem Teensy in Form von Variablen vorab definiert sind. Der Typ des LED-Streifens ist entsprechend der verwendeten Leuchtdioden definiert. Bei den in dem Projekt genutzten LED-Streifen handelt es sich um RGB-Leuchtdioden, welche einen Datenstrom von 800KHz erwarten. Die Typendefinition für das Neopixel-Objekt lautet entsprechend der möglichen Konventionen: „NEO_GRB NEO_KHZ800“. Für den anschließenden Zugriff auf die LED-Streifen ist im weiteren Verlauf ausschließlich der Objektname notwendig. Durch das Aufrufen von Funktionen aus der Bibliothek für das entsprechende Objekt, wird der definierte LED-Streifen angesteuert. [33], [34]

Für die beiden Kategorien der Beleuchtung sind demnach zwei unterschiedliche Neopixel-Objekte definiert. Sowohl die Umgebungsbeleuchtung als auch die Statusbeleuchtung ist jeweils in einem individuellen Objekt, mit unterschiedlichen Eigenschaften, zusammengefasst. Auch wenn bei den jeweiligen Beleuchtungen bei der Hardware mehrere LED-Streifen zum Einsatz kommen, sollen diese nicht unabhängig voneinander angesteuert werden. Viel mehr summiert sich, durch die serielle Verschaltung der LED-Streifen einer Beleuchtungskategorie, die gesamte Pixel-Anzahl auf. In der Ansteuerung sind diese mehreren LED-Streifen entsprechend einem einzelnen LED-Streifen mit einer höheren Pixel-Anzahl zu verwenden.

Im weiteren Ablauf des Skripts, muss der Ausgangspin des Teensys zunächst für die Neopixel-Ausgabe vorbereitet werden. Hierfür stellt die Neopixel-Bibliothek bereits entsprechende Funktionen bereit. In dem setup-Bereich wird durch einen Aufruf der „begin()“-Funktion für das gewünschte Neopixel-Objekt die Datenausgabe auf dem definierten Ausgangspin aktiviert. Der Aufruf erfolgt in dem Skript zweifach, für jedes Objekt der beiden Beleuchtungskategorien separat. [33], [34]

Die tatsächliche Ansteuerung der LED-Streifen erfolgt im Allgemeinen durch das Setzen der verschiedenen Leuchteigenschaften für jeden LED-Pixel. Entsprechend der verwendeten LED-Streifen und der somit verbauten Leuchtdioden, kann die Eigenschaft der Leuchtfarbe für jeden Pixel individuell, sowie die Beleuchtungsintensität für den gesamten LED-Streifen gesetzt werden. Für das Setzen der diversen Eigenschaften für jeden Pixel, müssen die einzelnen Pixel individuell angesprochen werden. [33], [34] Wie bereits in dem Kapitel Stand der Technik (vgl. Kapitel 3.3.2) beschrieben, können die einzelnen Leuchtdioden auf dem LED-Streifen durch den Pixel-Index adressiert werden. Dieser Index stellt eine fortlaufende ganze Zahl dar. Die erste Leuchtdiode nach den Eingangspins an den LED-Streifen hat hierbei den Index 0, die nachfolgenden Leuchtdioden entsprechend inkrementiert (vgl. Abbildung 4.29)

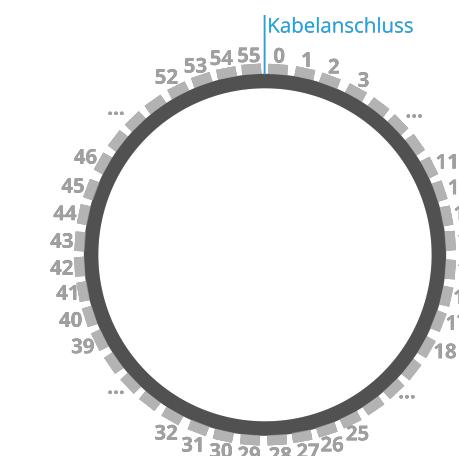


Abbildung 4.29: Darstellung Indexierung einzelner Pixel LED-Streifen bei Ringlicht

Für das Setzen der diversen Leucht-Eigenschaften der Pixel existieren in der Neopixel-Bibliothek definierte Funktionen. So setzt die Funktion „`setPixelColor()`“, welche in verschiedenen Ausführungen existiert, die Leuchtfarbe des einzelnen Pixels. Als Parameter müssen dieser Funktion im Allgemeinen der Pixel-Index des adressierten Pixels sowie die gewünschte Leuchtfarbe, entsprechend der Ausführung der Funktion, übergeben werden [34]. Weiterhin existiert die Funktion „`setBrightness()`“ in der Neopixel-Bibliothek. Diese Funktion setzt die Leuchtintensität für den gesamten LED-Streifen. Entsprechend müssen keine einzelnen LED-Indizes übergeben werden, vielmehr ist nur eine Spezifikation der Helligkeit im Bereich zwischen 0 und 255 notwendig. [34]

Die definierten Funktionen, um die Eigenschaften der LEDs festzulegen, haben dennoch keinen direkten Einfluss auf die LED-Streifen. Die gesetzten Änderungen der Eigenschaften erfolgen zunächst in Daten auf dem RAM-Speicher und werden nicht auf den LED-Streifen und die einzelnen Pixel geschrieben. Um die Änderungen auch in den Leuchteigenschaften der tatsächlichen LEDs umzusetzen, ist ein weiterer Funktionsaufruf notwendig. Mit dem Aufruf der Funktion „`show()`“ für das entsprechende Objekt des gewünschten LED-Streifens, werden die einzelnen Pixel-Daten aus dem RAM des Mikrocontrollers auf die NeoPixel geschrieben. Der Funktion müssen zudem keine weiteren Parameter übergeben werden. Die vorab definierten Leuchteigenschaften werden nach Aufruf auf den LED-Streifen angezeigt. [33], [34]

Die Realisierung verschiedener Lichteffekte erfolgt nur durch Variation der genannten Eigenschaften für jeden Pixel individuell. Hierbei kann mittels diverser Schleifen für jeden Pixel fortlaufend die gewünschte Eigenschaft gesetzt werden (vgl. Kapitel 4.3.3.3). Ein anschließender Aufruf der Show-Funktion zu der entsprechend Zeit, aktualisiert die Leuchteigenschaft auf dem LED-Streifen. Entsprechend muss bei komplexen Lichteffekten, jede einzelne Änderung eines Pixels durch die Funktionen auf den LED-Streifen geschrieben werden. In der Struktur des Skriptes werden die unterschiedlichen Leuchteffekte in jeweils einzelnen Funktionen definiert (vgl. Kapitel 4.3.3.3). Diese Funktionen beinhalten jeweils einen vollständigen Ablauf des gewünschten Leuchteffekts, welcher direkt auf dem LED-Streifen umgesetzt wird. Im weiteren Ablauf des Skriptes, muss somit ausschließlich die gewünschte Effekt-Funktion aufgerufen werden.

4.3.3.2 ROS

Bei beiden Beleuchtungskategorien ist die Eigenschaft der Ansteuerung von Daten des Roboters abhängig. So soll die Umgebungsbeleuchtung entsprechend der geforderten Helligkeit leuchten. Weiterhin sind bei der Statusbeleuchtung jeweils unterschiedliche Leuchteffekte für die diversen Bearbeitungszustände des Roboters vorhanden.

Damit eine Ansteuerung der Beleuchtungen abhängig von diesen Bedingungen möglich ist, müssen die entsprechenden Daten von dem restlichen ROS-System in dem Teensy-Skript empfangen werden.

Im Betrieb des Roboters wird der Teensy mit dem Skript über die serielle Schnittstelle mittels USB mit rosserial (vgl. Kapitel 2.3.7) als Knoten in das restliche Robotersystem eingebunden. Der Teensy könnte somit allgemein auf die entsprechenden Daten in dem ROS-System, identisch zu echten Knoten, zugreifen. In dem Teensy-Skript müssen weiterhin die ROS-Funktionalitäten implementiert werden. Hierfür muss zunächst die entsprechende Header-Datei „ros.h“ eingebunden werden. Anschließend kann auch in dem Skript auf alle ROS-Funktionalitäten entsprechend eines echten Knotens zugegriffen werden.

Weiterhin müssen die ROS-spezifischen Definitionen für einen einzelnen Knoten implementiert werden. Nach der Definition und Initialisierung des Knotens kann auf die, für den Ablauf des Skriptes, relevanten Funktionen zurückgegriffen werden. Für den weiteren Ablauf in dem Skript wird in dem loop-Bereich die Knoten-spezifische „spinOnce()“-Funktion aufgerufen. Hierdurch werden mögliche abonnierte Topics auf neue Nachrichten abgefragt und auf die entsprechenden Callback-Funktionen verwiesen.

Sowohl für die Daten zur Steuerung der Umgebungsbeleuchtung als auch für die Daten der Statusbeleuchtung existieren in der ROS-Architektur des Roboters bereits Topics mit definierten Messages. In der Umsetzung des Skripts werden hierbei die identischen Topics und Messages, welche bereits in der bestehenden Roboterplattform implementiert sind, genutzt (vgl. Kapitel 3.3). Um die Messages empfangen zu können, müssen die Topics in dem Ablauf zunächst abonniert werden. Das Abonnieren der Topics findet in dem setup-Bereich des Skripts statt, da der Aufruf nur einmalig benötigt wird. Um die Messages für die Ansteuerung der Umgebungsbeleuchtung zu erhalten, wird das Topic „detection_leds/intensity“ abonniert. Die Nachrichten mit den Daten zur Ansteuerung der Statusbeleuchtung werden auf dem Topic „status_monitor/rgb_led_control“ gesendet. [10] Dieses wird als weiteres Topic abonniert. Weiterhin wird bei der Definition der Subscriber für die entsprechenden Topics die Callback-Funktion definiert.

Auf den beiden Topics werden für die unterschiedlichen Beleuchtungen verschiedene Messages mit speziellem Aufbau versendet. So weist die Nachricht für die Umgebungsbeleuchtung einen einfachen Aufbau auf. Hierbei wird ein einziger Float64-Wert über definierte Standard-Messages versendet. Wie bereits in dem Kapitel Stand der Technik (vgl. Kapitel 3.3.1) beschrieben, stellt der gesendete Wert in Prozent die geforderte Leuchttintensität der Umgebungsbeleuchtung dar. Der erwartete Wertebereich liegt zwischen 0.0 und 1.0. [10]

Die Nachrichten für die Statusbeleuchtung weisen, im Vergleich zu der Umgebungsbeleuchtung, einen komplexeren Aufbau auf. Es handelt sich hierbei um einen speziell definierten Nachrichtentyp. Eine Nachricht enthält hierbei sieben individuelle Werte, jeweils von dem identischen Datentyp uint8. Die definierte Nachricht ist chronologisch aufgebaut und enthält folgende Eigenschaften: id; status; color_r; color_g; color_b; brightness; mode [10].

Nach dem Erhalt neuer Nachrichten auf den Topics werden die jeweils definierten Callback-Funktionen aufgerufen. Ähnlich zu der bereits vorhandenen Umsetzung in der bisher bestehenden Roboterplattform findet in der Callback-Funktion der Umgebungsbeleuchtung bereits die direkte Ansteuerung der

LED-Streifen statt. Nach einer Prüfung der empfangenen Werte auf valide Daten werden die Werte zwischen 0.0 und 1.0 auf den Wertebereich zwischen 0 und 255 umgerechnet. Diese berechneten Werte werden anschließend direkt der `setBrightness()`-Funktion übergeben. Hiermit ist die Leuchttintensität der gesamten Leuchtdioden für den LED-Streifen gesetzt. Weiterhin wird die Leuchtfarbe mit weißem Licht definiert. Durch den anschließenden Aufruf der `show()`-Funktion in der Callback-Funktion wird die Umgebungsbeleuchtung an dem Sensorkopf aktiviert. Wird bei der Prüfung auf valide Werte in den Messages der Wertebereich nicht eingehalten, so wird die Umgebungsbeleuchtung deaktiviert. Eine weitere Ansteuerung während der Laufzeit des Skripts der LED-Streifen der Umgebungsbeleuchtung außerhalb der Callback-Funktion findet im Allgemeinen nicht statt. Eine Ausnahme stellt jedoch das implementierte dreimalige Aufleuchten der Umgebungsbeleuchtung in dem „setup“-Bereich des Skriptes dar.

In der Callback-Funktion für die Statusbeleuchtung werden die empfangenen Werte in Variablen innerhalb des Skripts gespeichert. Für die weitere Bearbeitung sind hierbei hauptsächlich die Werte für die RGB-Farben (`color_r; color_g; color_b`) sowie der Modus (`mode`) und die Leuchttintensität (`brightness`) relevant. Mit diesen werden jeweils die unterschiedlichen Lichtfunktionen zur Ansteuerung der Statusbeleuchtung aufgerufen.

4.3.3.3 Lichtfunktionen Statusbeleuchtung

Die Ansteuerung der LED-Streifen der Statusbeleuchtung findet durch das Aufrufen einzelner Funktionen in dem Skript statt. Die unterschiedlichen Lichteffekte und Leuchteigenschaften der Statusbeleuchtung sind jeweils in einzelnen Funktionen definiert. In dem Skript existieren in Folge dessen viele unterschiedliche Funktionen, welche für die Beleuchtung aufgerufen werden können.

Im Allgemeinen sind in den Lichtfunktionen die verschiedenen Lichteffekte durch unterschiedliche Schleifen festgelegt. Es handelt sich bei den meisten Funktionen um `for`-Schleifen, welche die einzelnen Pixel-Indizes durchzählen. Hierdurch wird der Ablauf der Ansteuerung der einzelnen Pixel definiert (vgl. Abbildung 4.30). Weiterhin wird in vielen Lichtfunktionen auf die erhaltenen Daten bezüglich der Leuchteigenschaften aus den Nachrichten-Werten zurückgegriffen. So lassen sich die meisten Lichteffekte mit den unterschiedlichen RGB-Farben sowie in verschiedenen Leuchttintensitäten realisieren.

Für die Definition der verschiedenen Lichteffekte der Statusbeleuchtung wird diese zunächst in Abschnitte unterteilt. Wie bereits beschrieben, können die Pixel in einer kreisförmigen Anordnung betrachtet werden. Die Lichteffekte sollen entsprechend über diesen Kreis projiziert werden. Über die kreisförmige Anordnung sind in Summe 56 einzelne Pixel verteilt. Für Effekte, welche über die gesamte Statusbeleuchtung dargestellt werden sollen, ist ausschließlich der Index des letzten Pixels von Bedeutung. Durch eine weitere symmetrische Aufteilung der Pixel kann durch Kombination simpler Lichteffekte in einzelnen Bereichen eine große Anzahl an komplexen Lichteffekten reali-

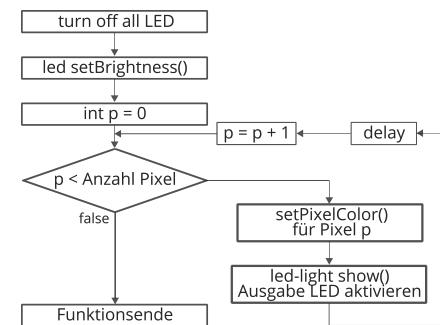


Abbildung 4.30: Schematische Darstellung Schleife: LEDs sequentiell aktivieren

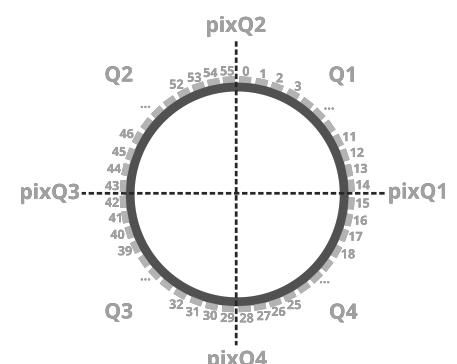


Abbildung 4.31: Aufteilung Bereiche LED-Indizes

siert werden. Hierfür wird die kreisförmige Anordnung in Viertel-Abschnitte gegliedert. Vor allem die Indizes der Pixel an den Übergängen sind anschließend von Bedeutung (vgl. Abbildung 4.31). Für die einzelnen Quadranten in dem Kreis sind in dem Code die jeweils verschiedenen Lichtfunktionen definiert. Entsprechend ist bei späterer Anwendung ein Quadrant der kleinste Bereich, welcher durch einen Funktionsaufruf einzeln aktiviert werden kann.

Die verschiedenen Funktionen folgen einer hierarchischen Definition. So werden für die diverseren Definitionsbereiche zunächst allgemeine Lichtfunktionen definiert. Dies umfasst einfache Funktionen wie das Einschalten oder auch das Ausschalten der Beleuchtung, jeweils mit den in den Nachrichten definierten Eigenschaften (vgl. Abbildung 4.32).

Weiterhin sind hierbei auch die Funktionen, wie das Ein- und Ausschalten mit Blendübergängen (fade) (vgl. Abbildung 4.33) sowie das Ein- und Ausschalten chronologisch der einzelnen Pixel in dem zugeordneten Bereich (füllen) (vgl. Abbildung 4.34) definiert.

Diese Grundfunktionen sind sowohl für die gesamte Statusbeleuchtung als auch für die einzelnen Bereiche (linker-, rechter-, oberer-, unterer-Bereich, einzelne Quartale) (vgl. Abbildung 4.35) implementiert.

Durch mehrere Aufrufe der Funktionen und Kombination der Grundfunktionen lassen sich anschließend die unterschiedlichen Leuchteffekte einfach realisieren. Entsprechend sind in dem Skript die spezifischen Effektfunktionen implementiert. Dies umfasst verschiedene Blink-Muster der diversen Bereiche, unterschiedliche Lauflichter mit variierenden Geschwindigkeiten sowie diverse Kombinationen von Farbmustern. Eine detaillierte Aufstellung der in dem Skript implementierten Leuchtfunktionen für die Statusbeleuchtung als Ringlicht wird nachfolgend dargestellt:

Bereich Ringlicht	Funktionsname	Beschreibung	Grundfunktionen			
			Bereich Ringlicht	Funktionsname	Beschreibung	
Gesamter Bereich (alle Pixel)	turnLightON	Einschalten Beleuchtung mit definierter Farbe und Helligkeit	Q1	turnQ1LightON	Einschalten Beleuchtung mit definierter Farbe und Helligkeit	
	turnLightOFF	Ausschalten Beleuchtung		turnQ1LightOFF	Ausschalten Beleuchtung	
	fadeLightOn	Ein-Blenden Beleuchtung mit definierter Farbe und Helligkeit		fadeQ1LightOn	Ein-Blenden Beleuchtung mit definierter Farbe und Helligkeit	
	fadeLightOFF	Aus-Blenden Beleuchtung		fadeQ1LightOFF	Aus-Blenden Beleuchtung	
Linker Bereich (Q2+Q3)	turnLeftLightON	Einschalten Beleuchtung mit definierter Farbe und Helligkeit	Q2	turnQ2LightON	Einschalten Beleuchtung mit definierter Farbe und Helligkeit	
	turnLeftLightOFF	Ausschalten Beleuchtung		turnQ2LightOFF	Ausschalten Beleuchtung	
	fadeLeftLightON	Ein-Blenden Beleuchtung mit definierter Farbe und Helligkeit		fadeQ2LightOn	Ein-Blenden Beleuchtung mit definierter Farbe und Helligkeit	
	fadeLeftLightOFF	Aus-Blenden Beleuchtung		fadeQ2LightOFF	Aus-Blenden Beleuchtung	
Rechter Bereich (Q1+Q4)	turnRightLightON	Einschalten Beleuchtung mit definierter Farbe und Helligkeit	Q3	turnQ3LightON	Einschalten Beleuchtung mit definierter Farbe und Helligkeit	
	turnRightLightOFF	Ausschalten Beleuchtung		turnQ3LightOFF	Ausschalten Beleuchtung	
	fadeRightLightON	Ein-Blenden Beleuchtung mit definierter Farbe und Helligkeit		fadeQ3LightOn	Ein-Blenden Beleuchtung mit definierter Farbe und Helligkeit	
	fadeRightLightOFF	Aus-Blenden Beleuchtung		fadeQ3LightOFF	Aus-Blenden Beleuchtung	
Oberer Bereich (Q1+Q2)	turntopLightON	Einschalten Beleuchtung mit definierter Farbe und Helligkeit	Q4	turnQ4LightON	Einschalten Beleuchtung mit definierter Farbe und Helligkeit	
	turntopLightOFF	Ausschalten Beleuchtung		turnQ4LightOFF	Ausschalten Beleuchtung	
	fadetopLightON	Ein-Blenden Beleuchtung mit definierter Farbe und Helligkeit		fadeQ4LightOn	Ein-Blenden Beleuchtung mit definierter Farbe und Helligkeit	
	fadetopLightOFF	Aus-Blenden Beleuchtung		fadeQ4LightOFF	Aus-Blenden Beleuchtung	
Unterer Bereich (Q3+Q4)	turnbottomLightON	Einschalten Beleuchtung mit definierter Farbe und Helligkeit	Bereiche kombiniert: (Q1+Q3; Q2+Q4)	turnQ1_3LightON	Einschalten Beleuchtung mit definierter Farbe und Helligkeit	
	turnbottomLightOFF	Ausschalten Beleuchtung		turnQ1_3LightOFF	Ausschalten Beleuchtung	
	fadebottomLightON	Ein-Blenden Beleuchtung mit definierter Farbe und Helligkeit		fadeQ1_3LightON	Ein-Blenden Beleuchtung mit definierter Farbe und Helligkeit	
	fadebottomLightOFF	Aus-Blenden Beleuchtung		fadeQ1_3LightOFF	Aus-Blenden Beleuchtung	

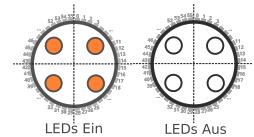


Abbildung 4.32: Visualisierung Ein- und Ausschalten

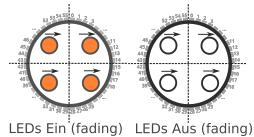


Abbildung 4.33: Visualisierung Ein- und Ausschalten mit Fade

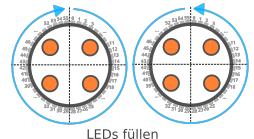


Abbildung 4.34: Sequentiell Ein- und Ausschalten (füllen)

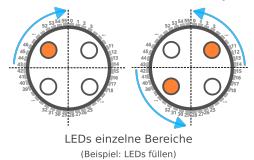


Abbildung 4.35: Einzelne Bereiche Ein- und Ausschalten

LED-Streifen füllen (Pixel chronologisch)					
Bereich Ringlicht	Funktionsname	Beschreibung	Bereich Ringlicht	Funktionsname	Beschreibung
Gesamter Bereich (alle Pixel) Richtung: gegen Uhrzeigersinn	fillTopCounterCLK	Einschalten Beleuchtung Pixel füllen mit definierter Farbe und Helligkeit (Start Pixel 0)	Gesamter Bereich (beidseitig symmetrisch)	fillMidBottom clearMidBottom	Einschalten / Ausschalten Beleuchtung Pixel symmetrisch füllen mit definierter Farbe und Helligkeit (Start Pixel pixQ2)
	clearTopCounterCLK	Ausschalten Beleuchtung Pixel füllen (Start Pixel 0)		fillMidTop clearMidTop	Einschalten / Ausschalten Beleuchtung Pixel symmetrisch füllen mit definierter Farbe und Helligkeit (Start Pixel pixQ4)
	fillBottomCounterCLK	Einschalten Beleuchtung Pixel füllen mit definierter Farbe und Helligkeit (Start Pixel pixQ4)		fillMidRight clearMidRight	Einschalten / Ausschalten Beleuchtung Pixel symmetrisch füllen mit definierter Farbe und Helligkeit (Start Pixel pixQ3)
	clearBottomCounterCLK	Ausschalten Beleuchtung Pixel füllen (Start Pixel pixQ4)		fillMidLeft clearMidLeft	Einschalten / Ausschalten Beleuchtung Pixel symmetrisch füllen mit definierter Farbe und Helligkeit (Start Pixel pixQ1)
	fillLeftCounterCLK	Einschalten Beleuchtung Pixel füllen mit definierter Farbe und Helligkeit (Start Pixel pixQ3)	Linker Bereich (Q2+Q3)	fillOnLeftCounterCLK fillOnLeftCLK	Einschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit
	clearLeftCounterCLK	Ausschalten Beleuchtung Pixel füllen (Start Pixel pixQ3)		ClearOnLeftCounterCLK ClearOnLeftCLK	Ausschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit
	fillRightCounterCLK	Einschalten Beleuchtung Pixel füllen mit definierter Farbe und Helligkeit (Start Pixel pixQ1)	Rechter Bereich (Q1+Q4)	fillOnRightCounterCLK fillOnRightCLK	Einschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit
	clearRightCounterCLK	Ausschalten Beleuchtung Pixel füllen (Start Pixel pixQ1)		ClearOnRightCounterCLK ClearOnRightCLK	Ausschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit
Gesamter Bereich (alle Pixel) Richtung: in Uhrzeigersinn	fillTopCLK	Einschalten Beleuchtung Pixel füllen mit definierter Farbe und Helligkeit (Start Pixel 0)	Oberer Bereich (Q1+Q2)	fillOnTopCounterCLK fillOnTopCLK	Einschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit
	clearTopCLK	Ausschalten Beleuchtung Pixel füllen (Start Pixel 0)		ClearOnTopCounterCLK ClearOnTopCLK	Ausschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit
	fillBottomCLK	Einschalten Beleuchtung Pixel füllen mit definierter Farbe und Helligkeit (Start Pixel pixQ4)	Unterer Bereich (Q3+Q4)	fillOnBottomCounterCLK fillOnBottomCLK	Einschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit
	clearBottomCLK	Ausschalten Beleuchtung Pixel füllen (Start Pixel pixQ4)		ClearOnBottomCounterCLK ClearOnBottomCLK	Ausschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit
	fillLeftCLK	Einschalten Beleuchtung Pixel füllen mit definierter Farbe und Helligkeit (Start Pixel pixQ3)	Q1	fillQ1CounterCLK fillQ1CLK clearQ1CounterCLK clearQ1CLK	Einschalten / Ausschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit
	clearLeftCLK	Ausschalten Beleuchtung Pixel füllen (Start Pixel pixQ3)		fillQ2CounterCLK fillQ2CLK clearQ2CounterCLK clearQ2CLK	Einschalten / Ausschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit
	fillRightCLK	Einschalten Beleuchtung Pixel füllen mit definierter Farbe und Helligkeit (Start Pixel pixQ1)	Q3	fillQ3CounterCLK fillQ3CLK clearQ3CounterCLK clearQ3CLK	Einschalten / Ausschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit
	clearRightCLK	Ausschalten Beleuchtung Pixel füllen (Start Pixel pixQ1)		fillQ4CounterCLK fillQ4CLK clearQ4CounterCLK clearQ4CLK	Einschalten / Ausschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit
			Bereich kombiniert Q1+Q3	fillQ1_Q3CounterCLK fillQ1_Q3CLK clearQ1_Q3CounterCLK clearQ1_Q3CLK	Einschalten / Ausschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit
			Bereich kombiniert Q2+Q4	fillQ2_Q4CounterCLK fillQ2_Q4CLK clearQ2_Q4CounterCLK clearQ2_Q4CLK	Einschalten / Ausschalten Beleuchtung Pixelbereich füllen mit definierter Farbe und Helligkeit

Effekt-Funktionen					
Bereich Ringlicht	Funktionsname	Beschreibung	Bereich Ringlicht	Funktionsname	Beschreibung
Gesamter Bereich (alle Pixel)	blinking blinkingSlow blinkingFast blinkingUltraFast	Beleuchtung blinken mit unterschiedlichen Geschwindigkeiten in definierter Farbe und Helligkeit	Gesamter Bereich (alle Pixel) Richtung: in Uhrzeigersinn	runningCLK runningCLKSlow runningCLKFast	Lauflicht (Pixel füllen und deaktivieren) mit unterschiedlichen Geschwindigkeiten in definierter Farbe und Helligkeit
Oberer Bereich (Q1+Q2)	blinkingTop blinkingTopDelay	Beleuchtung blinken mit definierter Geschwindigkeit in definierter Farbe und Helligkeit	Gesamter Bereich (alle Pixel) Richtung: gegen Uhrzeigersinn	runningCounterCLK runningCounterCLKSlow runningCounterCLKFast	Lauflicht (Pixel füllen und deaktivieren) mit unterschiedlichen Geschwindigkeiten in definierter Farbe und Helligkeit
Unterer Bereich (Q3+Q4)	blinkingBottom blinkingBottomDelay	Beleuchtung blinken mit definierter Geschwindigkeit in definierter Farbe und Helligkeit	Linker Bereich (Q2+Q3)	runningLeft runningLeftSlow runningLeftFast	Lauflicht in Bereich (Pixel füllen und deaktivieren) mit unterschiedlichen Geschwindigkeiten in definierter Farbe und Helligkeit
Linker Bereich (Q2+Q3)	blinkingLeft blinkingLeftDelay	Beleuchtung blinken mit definierter Geschwindigkeit in definierter Farbe und Helligkeit	Rechter Bereich (Q1+Q4)	runningRight runningRightSlow runningRightFast	Lauflicht in Bereich (Pixel füllen und deaktivieren) mit unterschiedlichen Geschwindigkeiten in definierter Farbe und Helligkeit
Rechter Bereich (Q1+Q4)	blinkingRight blinkingRightDelay	Beleuchtung blinken mit definierter Geschwindigkeit in definierter Farbe und Helligkeit	Unterer Bereich (Q3+Q4)	runningBottom runningBottomSlow runningBottomFast	Lauflicht in Bereich (Pixel füllen und deaktivieren) mit unterschiedlichen Geschwindigkeiten in definierter Farbe und Helligkeit
Q1	blinkingQ1	Beleuchtung blinken mit fester Geschwindigkeit in definierter Farbe und Helligkeit	Oberer Bereich (Q1+Q2)	runningTop runningTopSlow runningTopFast	Lauflicht in Bereich (Pixel füllen und deaktivieren) mit unterschiedlichen Geschwindigkeiten in definierter Farbe und Helligkeit
Q2	blinkingQ2	Beleuchtung blinken mit fester Geschwindigkeit in definierter Farbe und Helligkeit	Gesamter Bereich (alle Pixel) Richtung: in Uhrzeigersinn	runningQuarterCLK runningQuarterCLKSlow runningQuarterCLKFast	Lauflicht in Bereich (Pixel füllen und deaktivieren) mit unterschiedlichen Geschwindigkeiten in definierter Farbe und Helligkeit
Q3	blinkingQ3	Beleuchtung blinken mit fester Geschwindigkeit in definierter Farbe und Helligkeit	Gesamter Bereich (alle Pixel) Richtung: gegen Uhrzeigersinn	runningQuarterCounterCLK runningQuarterCounterCLKSlow runningQuarterCounterCLKFast	Lauflicht in Bereich (Pixel füllen und deaktivieren) mit unterschiedlichen Geschwindigkeiten in definierter Farbe und Helligkeit
Q4	blinkingQ4	Beleuchtung blinken mit fester Geschwindigkeit in definierter Farbe und Helligkeit	Gesamter Bereich (alle Pixel)	ringLight ringLightSlow ringLightFast	Ringlicht (Lauflicht gesamter Bereich) mit unterschiedlicher Geschwindigkeit in definierter Farbe und Helligkeit
Bereich kombiniert Q1+Q3	blinkingQ1_3	Beleuchtung blinken mit fester Geschwindigkeit in definierter Farbe und Helligkeit		rainbowStaticLight rainbowRingLight	Variation Farben (RGB-Farbmaschinen) über gesamte Beleuchtung in definierter Helligkeit
Bereich kombiniert Q2+Q4	blinkingQ2_4	Beleuchtung blinken mit fester Geschwindigkeit in definierter Farbe und Helligkeit		randomSparkleLight	Zufälliges Aufblitzen einzelner Pixel in definierter Farbe und Helligkeit
Gesamter Bereich (alle Pixel)	breathing breathingSlow breathingFast breathingUltra doubleBlinkWarning fastBlinkWarning rosBootConnectionLight rosConnectedLight	Dynamische Beleuchtung schwelend mit unterschiedlicher Geschwindigkeit in definierter Farbe und definierter maximaler Helligkeit Definierter Warnmuster Definierte Informations-Beleuchtungsmuster für ROS-Status		americanPoliceLight germanPoliceLight	Variation Blinkmuster mit definierten Farben ähnlich zu Signalbeleuchtung Polizei

4.3.3.4 State-Machine

Der Aufruf der einzelnen Lichtfunktionen erfolgt anhand des in den Nachrichten gesendeten Modus (mode). Jeder Modus-Wert ist in dem Skript an einer zentralen Stelle einer Beleuchtungsfunktion zugeordnet. Diese Zuordnung ist mit einer State-Machine realisiert. Hierfür existiert in dem Skript eine Funktion „`rgb_stateMachine()`“, welche in dem loop-Bereich kontinuierlich aufgerufen wird. Durch die wiederholten Aufrufe der State-Machine-Funktion werden die einzelnen zugeordneten Beleuchtungsfunktionen des zugeordneten Modes ebenfalls wiederholt aufgerufen. In Folge werden die Beleuchtungseffekte der Statusbeleuchtung, welche in den Funktionen, wie bereits beschrieben, nur mit einem einzelnen Ablauf definiert sind, für einen bestimmten Modus wie gewünscht kontinuierlich wiedergegeben.

Die Konfiguration der State-Machine kann entsprechend der Anwendung beliebig geändert werden. Ein Umschreiben der Funktionen zu den zugeordneten Modi ist hierfür notwendig. In der aktuellen Konfiguration sind die Beleuchtungsfunktionen wie nachfolgend dargestellt zugeordnet. Diese aktuelle Zuordnung ist in möglichst naher Übereinstimmung mit der bestehenden Statusbeleuchtung im Untergrund der Roboterplattform gewählt:

Belegung State-Machine					
Mode	Funktion	Mode	Funktion	Mode	Funktion
0	<code>rosConnectedLight</code>	13	Nicht definiert	26	<code>runningTopSlow</code>
1	<code>turnLightON</code>	14	Nicht definiert	27	<code>runningTopFast</code>
2	<code>blinkingSlow</code>	15	Nicht definiert	28	<code>runningQuarterCLKSlow</code>
3	<code>blinkingFast</code>	16	Nicht definiert	29	<code>runningQuarterCLKFast</code>
4	<code>blinkingUltraFast</code>	17	Nicht definiert	30	<code>ringLightSlow</code>
5	<code>blinkingTop</code>	18	Nicht definiert	31	<code>ringLightFast</code>
6	<code>blinkingBottom</code>	19	Nicht definiert	32	<code>rainbowStaticLight</code>
7	<code>blinkingLeft</code>	20	<code>runningCLKSlow</code>	33	<code>rainbowRingLight</code>
8	<code>blinkingRight</code>	21	<code>runningCLKFast</code>	34	<code>randomSparkleLight</code>
9	<code>blinkingQ2_4</code>	22	<code>runningCounterCLKSlow</code>	35	<code>americanPoliceLight</code>
10	<code>breathingSlow</code>	23	<code>runningCounterCLKFast</code>	36	<code>germanPoliceLight</code>
11	<code>breathingFast</code>	24	<code>runningBottomSlow</code>	37	<code>doubleBlinkWarning</code>
12	<code>breathingUltra</code>	25	<code>runningBottomFast</code>	38	<code>fastBlinkWarning</code>

4.3.3.5 Ablauf Skript Teensy

Wie in den vorherigen Abschnitten beschrieben, findet die eigentliche Steuerung der Statusbeleuchtung durch die Aufrufe der Funktionen in der State-Machine statt. Erst in den Leuchtfunktionen wird auf die Daten aus den Nachrichten zugegriffen. Die Steuerung der Umgebungsbeleuchtung wird direkt in der Callback-Funktion ausgeführt, hier sind keine weiteren Funktionen beteiligt. Der gesamte Ablauf des Skripts zur Steuerung der beiden Beleuchtungskategorien wird in Abbildung 4.36 dargestellt. Die gezeigten Abläufe der Funktionen werden hierbei, während der Laufzeit des Teensy, in dem loop-Bereich kontinuierlich wiederholt.

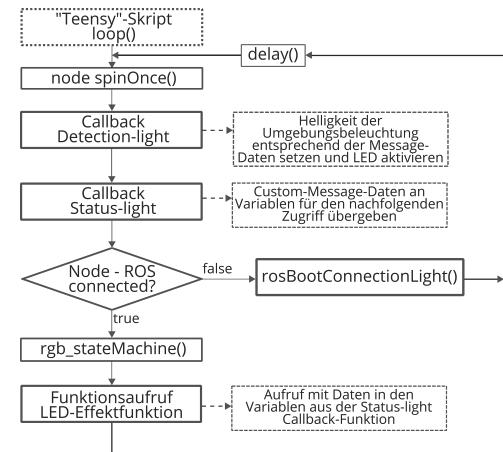


Abbildung 4.36: Darstellung Indexierung einzelne Pixel LED-Streifen bei Ringlicht

4.3.4 Greifer-Einstellung

Neben der hauptsächlichen Entwicklung der neuen LED-Steuerung für die diversen Beleuchtungen in dem neuen Sensorkopf muss weiterhin der Greifer bei der Umsetzung angepasst werden. Wie bereits in dem Bereich der Elektrotechnik beschrieben, wird der Greifer durch einen Dynamixel MX-28R Servomotor angetrieben. Dieser wird durch unterschiedliche Parameter definiert. Auf diesen Parametern basiert weiterhin die Ansteuerung der Greiferpositionen in dem ROS-System.

Durch die Anpassungskonstruktion des Sensorkopfes mit Greifer sowie die neue Montage, stimmen die in der vorhandenen Umsetzung eingestellten Parameter nicht mehr mit der neuen Hardware überein. Die Folge sind falsche Positionen des Greifers, was schließlich bei hoher Belastung zu diversen Beschädigungen führen kann.

Entsprechend müssen vor Inbetriebnahme diese Parameter auf den neuen Sensorkopf mit neuem Greifer angepasst werden. Das identische Einstellen des Motors und das Anpassen der Einbauposition in identischer Weise zu der bestehenden Plattform ist hierbei eine Möglichkeit. Alternativ können die Parameter direkt in der bestehenden Konfiguration, innerhalb der entsprechenden „.launch“- und config-Dateien auf die neuen Einbaupositionen angepasst werden.

Die Parameter für den Greifer in dem neuen Sensorkopf wurden zunächst durch einen identischen Einbau und die identische Konfiguration wie bei der ursprünglichen Roboterplattform übernommen.

5 Ausblick

Die Umsetzung des neuen Sensorkopfes wurde in den vorherigen Kapiteln dargestellt. Dies entspricht dem aktuell bestehenden Sensorkopf. Dennoch bestehen nach der Umsetzung und Nachbesserung einer ersten Version des Sensorkopfes diverse Verbesserungsmöglichkeiten, welche in möglichen nachfolgenden Versionen umgesetzt werden könnten.

5.1 Mechanik

Die Greifer-Mechanik hat bei diversen Tests ordnungsgemäß funktioniert.

Um die Stabilität des Greifers in Zukunft zu verbessern und das Spiel der Greiferfinger zu verringern können die Führungsschienen des Greifers, die momentan aus PETG mit dem 3D-Drucker hergestellt werden, aus POM gefertigt werden. Hierdurch werden bessere Gleiteigenschaften in der Führung erreicht. Außerdem kann das Bauteil aus einem Halbzeug mit einer CNC-Fräse ausgefräst werden. Es erhält eine wesentlich höhere Maßgenauigkeit und Oberflächengüte. Der Führungsschlitten läuft mit weniger Spiel, wodurch sich die Bauteile weniger leicht verkanten können.

Außerdem kann man die Umgebungsbeleuchtung noch weiter verbessern, indem man die Konstruktion mit beispielsweise Abdeckungen für die Kabel erweitert. Damit würden die Kabel für die LED-Strips verdeckt sein und das Risiko die Kabel zu beschädigen wäre minimiert.

5.2 Elektrotechnik

Die Versorgung der Bauteile mit der jeweils benötigten Spannung hat während der Wettkämpfe funktioniert und wurde somit erfüllt. Vor den Einsätzen entstanden jedoch Komplikationen durch die Spannungsversorgung. Durch das Versorgen des USB-Hubs und damit auch der anschließenden Komponenten über zwei Versorgungsspannungen – einmal über die Lochrasterplatine und einmal über das USB-B-Kabel – existierten zwei unterschiedliche Verbindungen. Diese wurden durch das Trennen der Spannungs- und Masseadern direkt in dem USB-Kabel behoben. Komplikationen sind nicht mehr zu erwarten.

Das Kabelmanagement konnte bisher nicht zufriedenstellend optimiert werden. Dazu müssten die Kabel mit Ihren Steckverbindungen neu gefertigt werden. Zudem kann eine Aufhängung am inneren Rand des Sensorkopfes oder die Verwendung von Kabelbindern dafür sorgen, dass die Kabel besser geordnet werden.

Die Verwendung von neuen Steckern kann im gleichen Zug dafür sorgen, dass die Kabel einfacher in die Lochrasterplatine und Bauteile gesteckt werden können.

Die bei unterschiedlichen Tests ausgefallene Umgebungsbeleuchtung durch ungünstige Lötkontaktierungen wurde durch einen Wechsel vom U-Design auf jeweils einen LED-Streifen über und einen unter der Intel RealSense Kamera behoben. Hierdurch sind kurze Lötverbindungen über das Eck bei den verwendeten drei LED-Streifen nicht mehr vorhanden.

5.3 Informatik

5.3.1 LED-Steuerung

Die Entwicklung des Skripts zur Steuerung der beiden Beleuchtungskategorien ist abgeschlossen. Die LED-Streifen werden entsprechend der gewünschten Funktionen korrekt angesteuert. Weiterhin funktioniert die Integration in das restliche ROS-System vollständig. Die Umgebungsbeleuchtung sowie die Statusbeleuchtung können mit der vollständigen Funktionalität genutzt werden. In unterschiedlichen Tests sowie einem finalen Einsatz sind keine fehlerhaften Funktionsabläufe aufgefallen.

Der Aufbau und die Funktionalität des Skripts zur Steuerung der Statusbeleuchtung folgt in der aktuellen Ausführung den bisher bestehenden Komponenten zur Ansteuerung der Beleuchtungen in dem Roboter. Vor allem aufgrund des weiteren Einsatzes der bestehenden Steuerung in der Status-

beleuchtung im Untergrund des Roboters, ist eine ähnliche Umsetzung der Skripte sinnvoll. Dennoch kann eine anschließende Weiterentwicklung der Ansteuerung der LED-Streifen, insbesondere für die Statusbeleuchtung weitere Vorteile bedeuten. Vor allem bei komplexeren Licht-Effekten oder auch weiteren Beleuchtungen an der Roboterplattform in unterschiedlichsten Formen bringt eine allgemeine mathematisch-basierte Realisierung der Ansteuerung der einzelnen Pixel deutliche Vorteile. Ähnlich dem Vorbild von Effektgeneratoren in professioneller Lichtsteuersoftware kann durch eine Realisierung der Lichtsteuerung mit einer Projektion mathematischer Funktionen über die diversen Eigenschaften der LED-Beleuchtung eine flexiblere sowie umfangreichere Umsetzung realisiert werden. [35], [36] Vor allem bei komplexen Leuchtmustern sowie individuellen Eigenschaften einzelner Pixel rechtfertigt sich der umfangreiche Entwicklungsaufwand mit einer simplen Realisierung der Beleuchtungseffekt [35]. Dennoch müssen für diese Umsetzung auch Anpassung an den Schnittstellen in das weitere bestehende ROS-System erfolgen. Allgemein kann zudem eine synchronisierte Umsetzung der Beleuchtung zwischen den verschiedenen Statusbeleuchtungen an der Roboterplattform sinnvoll sein. Eine zeitliche Differenzierung zwischen den Effekten sowie das Projizieren gemeinsamer Leuchteffekte über alle Komponenten der Statusbeleuchtung an der Roboterplattform ermöglichen auch hierbei weitere Beleuchtungsmuster.

Weiterhin ist für die Umgebungsbeleuchtung auch eine farbliche Differenzierung umsetzbar. Durch die Umstellung der Umgebungsbeleuchtung auf die identischen RGB-LED-Streifen kann anschließend nicht nur die Leuchtintensität beeinflusst werden. Auch unterschiedliche Farbvariationen sind mit der neuen Hardware möglich. Denkbar ist demnach eine Implementierung, welche für die optimale Ausleuchtung der Umgebung nicht nur die Helligkeit variiert, sondern, angepasst auf ein gutes Kamerabild, die farbliche Ausleuchtung der Umgebung anpassen kann. Für diese Implementierung ist kein signifikant zusätzlicher Aufwand in dem Skript notwendig. Die Leuchtfarbe wird hierbei wie beschrieben aktuell stets auf weißes Licht definiert. Vielmehr ist eine Anpassung der Schnittstellen in das bestehende ROS-System notwendig, um auch hier in den Nachrichten entsprechende Farbdefinitionen zu empfangen.

Neben diesen weiterführenden Anpassungen in der Ansteuerung der LED-Streifen können auch neue Beleuchtungsfunktionen in der bestehenden Skript-Struktur hinzugefügt werden. Bei neuen Umsetzungen verschiedener neuer Beleuchtungseffekte, können diese als weitere Beleuchtungsfunktion implementiert werden.

5.3.2 Greifer-Einstellung

Die Parameter für den Greifer in dem neuen Sensorkopf wurden zunächst, wie beschrieben, durch einen identischen Einbau und die identische Konfiguration wie bei der ursprünglichen Roboterplattform übernommen. Die Funktionalität war in den einzelnen Tests gegeben.

Dennoch zeigten sich während dem Betrieb Unregelmäßigkeiten mit der Ansteuerung, welche mehrmals einen vollständigen Ausfall des Greifers zur Folge hatten. Diese Unregelmäßigkeiten konnten dennoch bei erneuten Tests nicht reproduziert werden. Eine vollständige Funktionalität war gegeben. Mögliche Ausfälle des Greifers sollten weiterhin bei Testszenarien beobachtet werden. Weiterhin ist bei einem möglichen Ausfall zunächst die direkte Ansteuerung des Dynamixel-Motors über die Dynamixel-Treiber zu prüfen.

Weiterhin können zur Verbesserung der Greifergenauigkeit sowie der Prozesse bei der Greifersteuerung die zugeordneten Greiferparameter für die verschiedenen Objekte erneut eingestellt werden. Diese Parameter werden für die Erkennungen, ob das gewünschte Objekt gegriffen wurde oder ein erneutes Greifen notwendig ist, sowie die allgemeine Griff-Kraft-Erkennung benötigt. Für das Konfigurieren dieser Parameter werden sowohl die gewünschten Objekte sowie der vollständig funktionale Sensorkopf benötigt. Da diese bisher nicht vorhanden waren, konnte ein Einstellen der Parameter noch nicht erfolgen und sollte vor dem nächsten Einsatz durchgeführt werden.

Abbildungsverzeichnis

2.1	Visualisierung ROS-Architektur computational graph [13]	5
3.1	Bestehende Roboterplattform mit Towerarm	9
3.2	Bestehende Greiferkonstruktion	9
3.3	Statusbeleuchtung Untergrund, schematische Darstellung	12
4.1	Grundstruktur	13
4.2	Gehäuse	13
4.3	Greifer	14
4.4	Führungsschraube	14
4.5	Statusring	14
4.6	Grundstruktur aus Aluminium	15
4.7	Kunststoffbauteil mit Heatinserts	15
4.8	Montage Sensorkopf	16
4.9	Montage Greifer	16
4.10	Montage Statusring	17
4.11	Montage USB-Hub	17
4.12	Montage Leiterplatte	17
4.13	Montage Fish-Eye-Kamera	17
4.14	Montage 3D-Kamera	17
4.15	Sensorkopf Neura	18
4.16	Sensorkopf Igus	18
4.17	Vorher	18
4.18	Nachher	18
4.19	USB-Hub [26]	19
4.20	Servomotor Dynamixel MX-28R [27]	20
4.21	Intel RealSense [28]	20
4.22	Fisheye-Kamera [29]	20
4.23	LED-Streifen [30]	21
4.24	Teensy 4.0 [31]	21
4.25	Stromlaufplan Sensorkopf	21
4.26	Anschlüsse USB-Hub	22
4.27	Anschlüsse Teensy	22
4.28	Lochrasterplatine	23
4.29	Darstellung Indexierung einzelne Pixel LED-Streifen bei Ringlicht	27
4.30	Schematische Darstellung Schleife: LEDs sequentiell aktivieren	29
4.31	Aufteilung Bereiche LED-Indizes	29
4.32	Visualisierung Ein- und Ausschalten	30
4.33	Visualisierung Ein- und Ausschalten mit Fade	30
4.34	Sequentiell Ein- und Ausschalten (füllen)	30
4.35	Einzelne Bereiche Ein- und Ausschalten	30
4.36	Darstellung Indexierung einzelne Pixel LED-Streifen bei Ringlicht	32

Literaturverzeichnis

- [1] Technische Hochschule Nürnberg. "Mobile Robotik einfach erklärt." (3. Jan. 2022), Adresse: <https://www.th-nuernberg.de/news/4491-mobile-robotik-einfach-er/> (besucht am 13.01.2024).
- [2] Technische Hochschule Nürnberg. "Liga – Technische Hochschule Nürnberg Georg Simon Ohm." (13. Jan. 2024), Adresse: <https://www.th-nuernberg.de/fakultaeten/efi/forschung/forschungsaktive-labore/mobile-robotik/robocup-work/liga/> (besucht am 13.01.2024).
- [3] Technische Hochschule Nürnberg. "Wieder Weltmeister!" (20. Juli 2022), Adresse: <https://www.th-nuernberg.de/news/4678-wieder-weltmeister/> (besucht am 13.01.2024).
- [4] Neura Robotics. "LARA_Brochure_102023_EN.Pdf." (13. Okt. 2023), Adresse: https://neurarobotics.px.media/plk/LARA_Brochure_102023_EN.pdf (besucht am 13.01.2024).
- [5] RS. "Mikrocontroller Einfach Erklärt." (5. Jan. 2024), Adresse: <https://de.rs-online.com/web/content/discovery-portal/produktratgeber/mikrocontroller-leitfaden>.
- [6] elektorstore. "Arduino Nano." (5. Jan. 2024), Adresse: <https://www.elektor.de/arduino-nano>.
- [7] Electronic Projects. "Teensy® 4.0 Development Board. Electronic Projects." (15. Dez. 2023), Adresse: <https://www.pjrc.com/store/teensy40.html>.
- [8] Microchip. "Why Choose a PMIC." (5. Jan. 2024), Adresse: <https://www.microchip.com/en-us/products/power-management/pmic-power-management-ics/pmic-technology-101#power-management-single-package>.
- [9] ROS Wiki. "ROS/Introduction - ROS Wiki." (8. Aug. 2018), Adresse: <https://wiki.ros.org/ROS/Introduction> (besucht am 03.01.2024).
- [10] autonohm. "Autonohm GitHub Ohm_atwork," ohm_atwork. (13. Jan. 2024), Adresse: https://github.com/autonohm/ohm_atwork.
- [11] M. Quigley, B. Gerkey, K. Conley u. a., "ROS: An open-source Robot Operating System,"
- [12] ROS Wiki. "ROS/Concepts - ROS Wiki." (20. Sep. 2022), Adresse: <https://wiki.ros.org/ROS/Concepts> (besucht am 11.11.2023).
- [13] O'Reilly Media. "The computation graph level." (14. Jan. 2024), Adresse: <https://www.oreilly.com/library/view/ros-robotics-projects/9781838649326/b517d393-f115-4e0b-8271-d56ed1e2ca8f.xhtml> (besucht am 14.01.2024).
- [14] ROS Wiki. "Nodes - ROS Wiki." (4. Dez. 2018), Adresse: <https://wiki.ros.org/Nodes> (besucht am 11.11.2023).
- [15] ROS Wiki. "Topics - ROS Wiki," Topics. (20. Feb. 2019), Adresse: <https://wiki.ros.org/Topics> (besucht am 11.11.2023).
- [16] ROS Wiki. "Msg - ROS Wiki." (31. Jan. 2023), Adresse: <http://wiki.ros.org/msg> (besucht am 12.11.2023).
- [17] ROS Wiki. "Parameter Server - ROS Wiki." (8. Nov. 2018), Adresse: <http://wiki.ros.org/Parameter%20Server> (besucht am 12.11.2023).
- [18] ROS Wiki. "Master - ROS Wiki." (15. Jan. 2018), Adresse: <http://wiki.ros.org/Master> (besucht am 12.11.2023).

- [19] ROS Wiki. "Packages - ROS Wiki." (14. Apr. 2019), Adresse: <https://wiki.ros.org/Packages> (besucht am 04.01.2024).
- [20] ROS Wiki. "Rosserial - ROS Wiki." (1. Okt. 2018), Adresse: <http://wiki.ros.org/rosserial> (besucht am 04.01.2024).
- [21] PlatformIO. "PlatformIO: Your Gateway to Embedded Software Development Excellence," PlatformIO. (4. Jan. 2024), Adresse: <https://platformio.org> (besucht am 04.01.2024).
- [22] PlatformIO. "PlatformIO IDE for VSCode — PlatformIO Latest Documentation." (4. Jan. 2024), Adresse: <https://docs.platformio.org/en/latest/integration/ide/vscode.html#> (besucht am 04.01.2024).
- [23] Arduino. "Arduino Sketches | Arduino Documentation." (11. Jan. 2024), Adresse: <https://docs.arduino.cc/learn/programming/sketches> (besucht am 04.01.2024).
- [24] Zedfy GmbH. "Adressierbare LEDs wie Ws2812b Sk6812 Neopixel." (4. Jan. 2024), Adresse: <https://www.zedfy.shop/unterschied-adressierbarer-leds-wie-ws2812b-sk6812-neopixel-und-mehr> (besucht am 04.01.2024).
- [25] Adafruit GitHub. "Adafruit NeoPixel Library." (4. Jan. 2024), Adresse: https://github.com/adafruit/Adafruit_NeoPixel (besucht am 04.01.2024).
- [26] reichelt elektronik. "LOGILINK UA0170 USB 3.0 Hub, 4-Port, Schwarz, Inkl. Netzteil." (15. Dez. 2023), Adresse: https://www.reichelt.de/ch/de/usb-3-0-hub-4-port-schwarz-inkl-netzteil-logilink-ua0170-p188370.html?&trstct=pol_2&nbc=1.
- [27] Generation ROBOTS. "Servomotor Dynamixel MX-28R." (15. Dez. 2023), Adresse: <https://www.generationrobots.com/de/401088-servomotor-dynamixel-mx-28r.html>.
- [28] RS. "Intel D435 Tiefenkamera 1280 x 720 10m 0.2m Tiefendistanz." (15. Dez. 2023), Adresse: <https://de.rs-online.com/web/p/tiefenkameras/1720981>.
- [29] ELP. "180 DEGREE FISHEYE LENS FULL HD 1080P USB CAMERA MODULE USB2.0 OV2710 COLOR SENSOR MJPEG." (15. Dez. 2023), Adresse: <http://www.elpcctv.com/180-degree-fisheye-lens-full-hd-1080p-usb-camera-module-usb20-ov2710-color-sensor-mjpeg-p-206.html>.
- [30] Mouser Electronics. "Led Lighting Bars & Strips - M5 Stack A035." (15. Dez. 2023), Adresse: <https://www.mouser.de/ProductDetail/M5Stack/A035?qs=81r%252BiQLm7BR9%252BrYGJ%2Fehhw%3D%3D>.
- [31] reichelt elektronik. "TEENSY 4.0 Teensy 4.0, USB, Ohne Header." (15. Dez. 2023), Adresse: <https://www.reichelt.de/ch/de/teensy-4-0-usb-ohne-header-teensy-4-0-p269006.html?r=1>.
- [32] Adafruit. "Basic Connections," Adafruit Learning System. (13. Jan. 2024), Adresse: <https://learn.adafruit.com/adafruit-neopixel-oberguide/basic-connections> (besucht am 05.01.2024).
- [33] Adafruit. "Adafruit NeoPixel Arduino Library Use," Adafruit Learning System. (13. Jan. 2024), Adresse: <https://learn.adafruit.com/adafruit-neopixel-oberguide/arduino-library-use> (besucht am 05.01.2024).
- [34] Adafruit_Neopixel. "Adafruit NeoPixel Class Reference." (4. Jan. 2024), Adresse: https://adafruit.github.io/Adafruit_NeoPixel/html/class_adafruit___neo_pixel.html (besucht am 05.01.2024).

- [35] MA Lighting Technology GmbH. "Create an Effect in the Programmer - grandMA2 User Manual." (4. Jan. 2024), Adresse: http://help2.malighting.com/Page/grandMA2/effects_create_in_programmer/en/3.9 (besucht am 05.01.2024).
- [36] MA Lighting Technology GmbH. "Effects - grandMA2 User Manual." (4. Jan. 2024), Adresse: <http://help2.malighting.com/Page/grandMA2/effects/en/3.9> (besucht am 05.01.2024).