Betrieb: exceeding solutions GmbH

Betreuer: M. Eng. André Gehrmann

Semester: SoSe 2024

Prüfer: Prof. Dr. Uwe Heuert

Fachbereich: INW

Masterarbeit

Entwicklung einer autonom fliegenden Indoor-Drohne

Edgar Lumpe

02.07.2024

Danksagung

Auf der Suche nach einer geeigneten Firma und einem Thema für meine Masterarbeit bin ich auf exceeding solutions gestoßen. Dort durfte ich nicht nur an einem spannenden und zukunftsweisenden Thema arbeiten, sondern auch Teil eines wundervollen Teams werden, das mich bei der Erstellung der Masterarbeit begleitet hat. Bevor ich zur eigentlichen Arbeit komme, möchte ich mich bei mehreren Menschen bedanken, die eine große Unterstützung für mich waren.

Mein erster Dank gilt meinem Professor Dr. Uwe Heuert, der diese Erfahrungen möglich gemacht hat. Vielen Dank für das spannende Thema, dein Vertrauen in mich und deine hilfreichen Anregungen für die Arbeit!

Ein besonderer Dank geht an meine beiden Betreuer Kevin Saalmann und André Gehrmann, deren Erfahrungen und Wissen von unschätzbaren Wert für mich und diese Arbeit waren. Vielen Dank, dass ich zu jeder Zeit mit Fragen und Ideen auf euch zukommen konnte und dafür, dass ihr mich so tatkräftig unterstützt habt!

Auch meiner Familie möchte ich für die Unterstützung danken. Danke, dass ihr mir stets den Rücken freigehalten habt, sodass ich mich voll und ganz auf das Studium konzentrieren konnte!

Einen besonderen Dank möchte ich meiner Freundin Milica aussprechen, die die Arbeit Korrektur gelesen hat und mir beim Formatieren und Umformulieren geholfen hat. Danke, dass du die ganze Zeit über für mich da warst und mir mit Rat und Tat zur Seite standest! Ohne dich wäre diese Arbeit nicht möglich gewesen. Vielen Dank für alles!

Nochmals vielen Dank an alle, die zu dieser Arbeit beigetragen haben.

Edgar Lumpe

Inhalt

1 Einleitung 5
2 Theoretische Grundlagen 7
2.1 Unbemannte Flugsysteme 7
2.2 Kommunikation mit 5G 8
2.3 Hardware 9
2.3.1 Sensorik 9
2.3.1.1 Tiefenkamera 9
2.3.1.2 Optische Fernerkundungstechnik (LIDAR) 10
2.3.1.3 Modulierter Dauerstrichradar (FMCW) 10
2.3.2 Bürstenloser Gleichstrommotor 11
2.4 Software: Docker 12
2.4.1 Dockerfile 13
2.4.2 Befehle zum Bauen und Starten von Containern 14
2.4.3 Netzwerke in Docker 15
2.4.4 Docker Compose 16
3 Entwicklung der Drohne 20
3.1 Bau der Drohne 20
3.1.1 Vorbetrachtungen 20
3.1.2 Drohnenarten 23
3.1.3 Bestandteile einer Drohne 24
3.1.4 Auswahl konkreter Bauteile 27
3.1.5 Aufbau der Drohne 31
3.1.6 Positionsbestimmung der Drohne 31
3.2 Entwicklung eines Sensorkonzepts 34
3.2.1 Vorbetrachtungen 34
3.2.2 Steuerung 35
3.2.3 FMCW-Radar 36
3.2.4 Tiefenkamera 38
3.2.5 LIDAR 41
3.2.6 Entwurf eines ersten Sensorkonzepts 49
3.2.6.1 Komponenten 50
3.2.6.2 Befestigung der Komponenten 51
3.2.7 Diskussion des Sensorkonzepts 55

3.3 Anpassung des Sensorkonzepts 57
3.3.1 Vorbetrachtungen 57
3.3.2 3D-Druck 60
3.3.3 Toradex-Board 64
3.3.4 Wechsel zu Toradex 67
3.3.4.1 LIDAR 67
3.3.4.2 Tiefenkamera 68
3.3.4.3 Host Container 71
3.3.4.4 Docker Compose 74
3.3.5 Anbindung an das 5G-Campusnetz 75
3.4 Ergebnisse des finalen Aufbaus 76
4 Schluss 80

Anhang
I Abbildungsverzeichnis
II Literaturverzeichnis
III Eigenständigkeitserklärung

CD-ROM

1 Einleitung

In der modernen Lagerlogistik sind Effizienz und Genauigkeit entscheidende Erfolgsfaktoren. Insbesondere in Hochregallagern stellt die Inventur eine zeitaufwändige und kostspielige Herausforderung dar. Um Kosten und Zeit einzusparen, bietet es sich an, autonom fliegende Drohnen einzusetzen, da sie dank ihrer Bewegungsfreiheit diverse Einsatzmöglichkeiten bieten und neben den genannten Vorteilen auch die Sicherheit und Genauigkeit der Bestandsaufnahme verbessern. Im Rahmen dieser Masterarbeit wird eine autonom fliegende Drohne gebaut, die zur Durchführung von Inventuren im Hochregallager der Firma Schaeffler eingesetzt werden soll. Schaeffler produziert Komponenten und Systemlösungen für diverse Fahrzeugarten (vgl. Schaeffler Technologies AG & Co. KG o.A.). Im Star Park in Halle (Saale) hat die Firma ein Logistikzentrum errichtet, das ein umfassendes, vollautomatisiertes Hochregallager beinhaltet. Diese Arbeit ist Teil des Forschungsprojekts „5G-POUST“ in der Firma exceeding solutions.

Exceeding solutions ist ein Anbieter ganzheitlicher und integrierter Ingenieurs- und IT-Dienstleistungen. Die Firma beschäftigt sich hauptsächlich mit intelligenten Stromzählern, auch „Smart Meter“ genannt. Dank der Heterogenität der Portfolios der Mitarbeiter können zusätzliche Leistungen in den Bereichen Kryptographie, Informations- und Kommunikationstechnologie sowie Mess-, Steuerungs- und Automatisierungstechnik angeboten werden (vgl. exceeding solutions GmbH o.A.a).

POUST ist eine Abkürzung für „präzise Organisieren und smarte Telemetrie“. Das Projekt wird hauptsächlich durch Halle Saale Investvision und die Hochschule Merseburg unterstützt. Daneben sind weitere Firmen wie z.B. Schaeffler und das Fraunhofer IIS in 5G-POUST involviert. In dem Forschungsprojekt geht es darum, industrielle Anwendungen mit Hilfe des Mobilfunkstandards 5G zu (teil-)automatisieren und damit zu optimieren.

Heutzutage werden hochauflösende Sensoren für eine Vielzahl von industriellen Prozessen verwendet. Dabei wird eine große Menge an unterschiedlichen Daten erzeugt, die in ihrem Umfang möglichst effizient verarbeitet werden muss, um Anforderungen in Echtzeit erfüllen zu können. Zur Überwachung und Steuerung der Daten ist eine Technik notwendig, die die „technologischen Abhängigkeitsketten“ (exceeding solutions GmbH o.A.b) erkennt und verbessert, den Energiestrom der daraus resultierenden Daten strukturiert und „energetische[] Abhängigkeiten“ (ebd.)

erfasst und beeinflusst. Um diese Technik zu realisieren, ist geplant, diese Anlagen zu vernetzen und zu überwachen, um zunächst ein umfassendes Bild über den aktuellen Zustand und die Leistungsfähigkeit zu gewinnen. Dazu wird ein 5G-Campusnetz genutzt, in das Sensoren und Aktoren integriert werden, die detaillierte und zeitnahe Messdaten erheben. Diese Daten werden anschließend von einer Künstlichen Intelligenz (KI) in Cloud- oder Edge-Computing-Geräten verarbeitet und analysiert. Die daraus gewonnenen Erkenntnisse ermöglichen eine direkte Reaktion im Produktionsprozess, sodass Optimierungen und Anpassungen in Echtzeit vorgenommen werden können. Das Projekt hat nicht nur zum Ziel, mit Hilfe dieser und anderer Lösungen Industrieanwendungen zu entwickeln, die mit 5G betrieben werden. Die in den dafür vorgesehenen Tests erworbenen Erkenntnisse tragen dazu bei, die zukünftige, anwendungsnahe Forschung mit 5G voranzutreiben.

In dieser Arbeit wird Drohnentechnologie mit Sensorintegration und 5G-Kommunikation kombiniert. Das Ziel ist es, ein umfassendes Drohnensystem zu entwickeln, das die Effizienz der Inventur im Hochregallager der Firma Schaeffler steigert. Konkret hat die Drohne die Aufgabe, den Bestand des Lagers zu überprüfen.

Um das Ziel zu erreichen, werden zunächst die Grundlagen zum Bau einer Drohne erläutert. Anschließend erfolgt die Konstruktion einer eigenen Drohne und der erste Entwurf eines Sensorkonzepts, mit dem autonomes Fliegen ermöglicht werden soll. Anschließend wird dieser Entwurf bewertet, um mit daraus resultierenden Schwächen Verbesserungen vornehmen zu können. Auf dieser Basis wird ein neuer Aufbau realisiert, der unter Einbindung eines fortschrittlicheren Sensorkonzepts optimiert wird.

Die hierfür genutzte Kommunikation über das 5G-Campusnetz demonstriert das Potential des Mobilfunkstandards für industrielle Anwendungen. Insgesamt leistet diese Arbeit einen wertvollen Beitrag zur Weiterentwicklung der Lagerlogistik.

2 Theoretische Grundlagen

Im ersten Teil dieser Arbeit wird die Funktionsweise von Drohnen und der Mobilfunktechnologie 5G allgemein betrachtet, bevor die notwendige Hard- und Software für den Bau einer Drohne untersucht wird.

2.1 Unbemannte Flugsysteme

Ein UAS (eng. UAV) ist ein unbemanntes Luftfahrzeug, welches aus der Entfernung von einem Computer oder von einer Person gesteuert wird. Die Abkürzung UAS steht für „unmanned aircraft system“ (Baumgärtel/Landrock 2018: 2) (eng. „unmanned aerial vehicel“ (ebd.: 3). Dabei umfasst der Begriff UAS nicht nur die fliegenden Komponenten des Systems, sondern bezieht sich auch auf Sensorik am Gerät und auf die Steuerung. In der Umgangssprache werden diese unbemannten Luftfahrzeuge auch Drohnen genannt. (Vgl. ebd.: 2-3)

UASs lassen sich in zwei Kategorien unterteilen: zum einen in die vollkommen selbständig fliegenden UAS und zum anderen in die RPAS. RPAS, kurz für „remotly piloted aviation system“ (ebd.: 3), sind Luftfahrzeuge, welche zwar von einem Piloten gesteuert werden, diesen aber nicht an Bord mit sich führen. Diese Art Pilot wird auch als Fernpilot bezeichnet.

Weiterhin kann zwischen verschiedenen Drohnenbauarten unterschieden werden – dem Starrflügler und dem Multicopter. Starrflügler zeichnen sich vor allem durch ihre Fähigkeit aus, weite Strecken zurücklegen zu können, während sie teilweise schwere Lasten dabei transportieren. Da diese Drohnenart aber vor allem im militärischen Bereich vorkommt, wird sie in dieser Arbeit nicht weiter betrachtet. Das ausschlaggebende Kriterium für einen Multicopter sind die horizontal rotierenden Propeller. Diese ermöglichen es der Drohne, senkrecht zu starten und zu landen. Größe, Form, die Art des Antriebs sowie der Einsatzzweck variieren bei den Propellern stark. (Vgl. ebd.: 3)

In dieser Arbeit stehen Drohnen aus dem industriellen Bereich im Mittelpunkt. Diese werden in den meisten Fällen mit Motoren und Propellern in der Luft gehalten. Dabei ist die am weitesten verbreitete Bauform der Quadrocopter. Dieser besteht aus vier Motoren, die an den Ecken der Drohne montiert sind und dafür sorgen, dass genug Auftrieb vorhanden ist, um die Drohne fliegen zu lassen. (Vgl. ebd.: 3-4)

2.2 Kommunikation mit 5G

Die fünfte Generation der Mobilfunktechnologie (5G) stellt einen bedeutenden Fortschritt gegenüber den Vorgängergenerationen dar. Sie wurde entwickelt, um die Anforderungen einer zunehmend vernetzten Welt zu erfüllen und bietet erhebliche Verbesserungen in Bezug auf Geschwindigkeit und Kapazität. Diese Eigenschaften machen 5G nicht nur für den Alltag attraktiv, sondern auch für verschiedene Industriezweige wie die Produktion, das Gesundheitswesen und das Transportwesen. Das neu eingeführte „Network Slicing“ ermöglicht das Unterteilen eines physischen Netzwerkes in mehrere kleine, virtuelle Netzwerkteile mit unterschiedlichen Eigenschaften, die auf spezifische Anwendungen zugeschnitten sind (vgl. Donner/Luber 2019). Damit können verschiedene Dienste mit unterschiedlichen Anforderungen auf derselben Infrastruktur parallel betrieben werden. Auch nutzt 5G ein breiteres Spektrum an Frequenzbändern als frühere Generationen (vgl. Telekom Deutschland GmbH o.A.). Diese Frequenzen ermöglichen höhere Datenübertragungsraten und Kapazitäten, erfordern jedoch auch eine dichtere Netzwerkinfrastruktur aufgrund der begrenzten Reichweite höherer Frequenzen.
5G bietet sich für die folgenden drei Anwendungsbereiche an (vgl. ebd.):
Der erste Bereich ist eMBB (Enhanced Mobile Broadband). Die mit 5G einhergehende schnellere Datenübertragung ermöglicht das Erstellen einer qualitativ hochwertigen, wirklichkeitsgetreuen, virtuellen Welt (auch Virtual Reality genannt), die vor allem in der Gaming-Industrie beliebt ist, aber auch zu Trainings- und Schulungszwecken genutzt werden kann. Beim Augmented Reality werden „neben den realen Wahrnehmungen Zusatzinformationen zur Verfügung [gestellt], die einen unmittelbaren Bezug zum Gesehenen haben“ (ebd.). Auf der Homepage der Telekom wird als Beispiel aufgeführt, dass es zur Weltfußballmeisterschaft im Jahr 2022 bereits für die Zuschauer dank 5G möglich gewesen ist, Informationen zu den einzelnen Spielern mit einem auf den Fernseher gerichteten 5G-fähigen Smartphone abzurufen.

Ein weiterer Bereich, der von 5G profitiert ist mMTC („Massive Machine Type Communications“). Hierbei wird eine Vielzahl an Geräten zu Kommunikationszwecken miteinander verbunden. Anwendungsbeispiele sind intelligente Ampel-Systeme, die den Verkehr in Echtzeit optimieren, unbemannte Agrarmaschinen, die den Acker bearbeiten oder auch autonome Transportmittel wie die Drohne, die zur Verbesserung der Logistik in verschiedenen Bereichen eingesetzt werden. (Vgl. ebd.)

Der letzte Bereich, der hier Erwähnung findet, ist uRLLC („Ultra-Reliable and Low-Latency Communications“). 5G ist ebenfalls dazu in der Lage, die Latenzzeit zu verringern, was für allem für zeitkritische Anwendungen wie das autonome Fahren oder automatische Fahrassistenten notwendig ist. (Vgl. ebd.)

2.3 Hardware

Unter diesem Punkt wird auf die wichtigsten Bestandteile der Hardware eingegangen. Das sind die Sensoren sowie die Motoren, mit denen die Drohne fliegen soll.

2.3.1 Sensorik

2.3.1.1 Tiefenkamera

Eine Tiefenkamera ist ein Sensor, welcher Objekte nicht nur in 2D aufzeichnet, sondern auch die Tiefe bzw. Entfernung der Objekte im Sichtfeld misst. Das ermöglicht eine dreidimensionale Erfassung der Umgebung. Tiefenkameras werden vielfach gebraucht: zur Gesichtserkennung und in autonom fahrenden Verkehrsmitteln, aber auch in der Robotik und im Bereich der Augmented Reality.
Auch bei den Tiefenkameras kann zwischen verschiedenen Arten unterschieden werden: Stereokameras und ToF-Kameras. An Stereokameras sind zwei Kameras in einer festgelegten Distanz nebeneinander angebracht, die jeweils ein Bild aufnehmen (vgl. ClearView Imaging GmbH o.A.). Im aufgenommenen Bild sucht die Kamera nach sich ähnelnden Pixeln in den beiden Bildern und bestimmt mit Hilfe von Triangulation die Entfernung zwischen Kamera und Objekt (ebd.). Zusätzlich kann ein Infrarot-Projektor befestigt werden, der ein Muster wirft, das von beiden Kameras erkannt und zur Berechnung der Tiefenwerte einbezogen wird (vgl. Intel Corporation 2024).
ToF-Kameras („Time of Flight“-Kameras) bestehen aus drei Komponenten – der Beleuchtungseinheit, dem Objektiv und dem Bildsensor. Zur Ermittlung der Entfernung sendet die Beleuchtungseinheit ein „intensitätsmoduliertes Licht im nahen Infrarotbereich aus“ (Metrilus GmbH o.A.), das von der Oberfläche des zu messenden Objekts reflektiert wird. Im Anschluss gelangt es durch das Objektiv auf den Bildsensor. Die Entfernung jedes Pixels wird durch Korrelation der empfangenen und ausgesandten Werte bestimmt. Mit Hilfe der bekannten Geschwindigkeit von Licht und

der Flugzeit des Lichtstrahls zum Objekt kann die Entfernung des Objekts zur Kamera berechnet werden (vgl. Kamwa 2022):

$$d = \frac{c}{2f_{mod}} * \frac{\psi_d}{2\pi}$$

d ist die Entfernung in Meter und c die Lichtgeschwindigkeit in $\frac{m}{s}$. f_{mod} ist die Frequenz des Kosinussignals, mit dem die Intensität des von der Kamera ausgesandten, inkohärenten Lichtsignals moduliert wird. Die Variable ψ_d beschreibt die Phasenverschiebung des ausgesendeten und des empfangenen Lichts. (Vgl. Metrilus o.A.)

2.3.1.2 Optische Fernerkundungstechnik (LIDAR)

LIDAR, kurz „Light Detection and Ranging“ (Large 2024), ist eine optische Fernerkundungstechnik, die mit Hilfe von Laserstrahlen Entfernungen zu Objekten in der Umgebung misst. Das Grundprinzip beruht auf der Messung der Laufzeit eines ausgesandten Laserstrahls, der von einem Objekt reflektiert wird und zum Sensor zurückkehrt (vgl. Zauner 2023).
Diese Technologie ermöglicht die Erstellung präziser dreidimensionaler Karten und findet Anwendung in verschiedenen Bereichen wie Kartographie, Geodäsie, Umweltüberwachung und autonomes Fahren. (Vgl. passpunkt.de o.A.)
Die Vorteile von LIDAR-Sensoren sind die hohe Präzision, die hohe Auflösung, die Unabhängigkeit von Tageslicht und die Fähigkeit zur Erfassung von komplexen Umgebungen. Von Nachteil ist, dass die Sensoren empfindlich gegenüber Witterungsbedingungen sind und dass die Sensoren Schwierigkeiten haben, reflektierende und lichtabsorbierende Oberflächen zu erfassen. (Vgl. Large 2024)

2.3.1.3 Modulierter Dauerstrichradar (FMCW)

FMCW steht für „Frequency Modulated Continuous Wave“ (Wolff 2018), auf Deutsch frequenzmodulierte, kontinuierliche Welle. Hierbei handelt es sich um eine spezielle Radartechnologie, die zur Abstandsmessung und Geschwindigkeitsbestimmung genutzt wird. FMCW-Radare sind in vielen Bereichen zu finden, darunter in der Automobilindustrie, für Überwachungssysteme und in der industriellen Messtechnik. Die Sensoren, die dieses Messprinzip verwenden, senden kontinuierlich ein Signal aus, dessen Frequenz stetig verändert wird. Die Änderung der Frequenz bleibt dabei

konstant. Das ausgesandte Signal wird von der Umwelt reflektiert und vom Sensor wieder empfangen. Die empfangene Frequenz wird mit der aktuellen verglichen. Dadurch kann im Anschluss ermittelt werden, wie lange das Signal unterwegs war und wie weit die Gegenstände vom Sensor entfernt sind, die das Signal reflektiert haben. In der folgenden Abbildung wird ein solches Signal dargestellt:

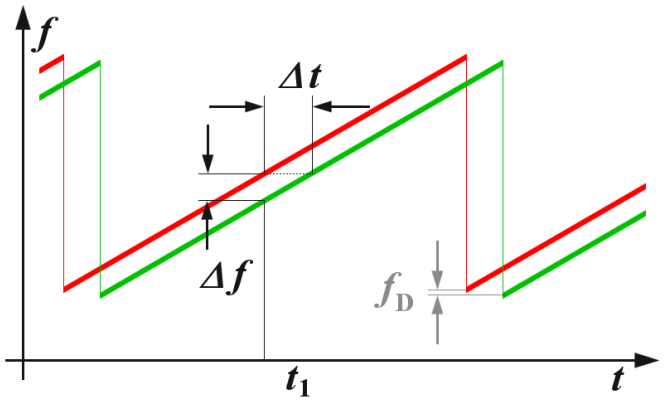


Abbildung 1: FMCW-Signal (übernommen von: Wolff 2018)

In der Abbildung sind zwei Signale zu sehen, die zeitlich versetzt sind. Das rote Signal ist das gesendete Signal, das grüne das empfangene. Mit Hilfe von Δt kann die Entfernung des reflektierenden Gegenstands zum Sensor errechnet werden. (Vgl. Wolff 2018)

2.3.2 Bürstenloser Gleichstrommotor

Ein bürstenloser Gleichstrommotor, auch „Brushless Motor“ genannt, ist ein Elektromotor. Im Gegensatz zu traditionellen Gleichstrommotoren, die mechanische Bürsten und einen Kommutator verwenden, um den Stromfluss zu den Wicklungen zu steuern, nutzt ein bürstenloser Motor elektronische Steuerungstechnologien. Dabei besteht der Rotor des Motors aus einem Permanentmagneten. Im Stator des Motors wird ein sich drehendes Magnetfeld erzeugt, das dafür sorgt, dass sich der Rotor im Inneren bewegt. Der Motor wird gesteuert, indem das sich drehende Magnetfeld im Inneren des Motors gesteuert wird. Vorteile von bürstenlosen Gleichstrommotoren gegenüber anderen Motoren sind der niedrigere Stromverbrauch, die kleinere Wärmeentwicklung und die Langlebigkeit aufgrund des verschleißarmen Prinzips. (Vgl. Baruschke 2021)

2.4 Software: Docker

Docker ist eine Open-Source-Softwareplattform, die auf allen gängigen Betriebssystemen läuft. Anwendungen werden in einem sogenannten „Container“ verpackt und können damit auf anderen Systemen verwendet werden.
Um mit Docker arbeiten zu können, sind nur wenige Komponenten notwendig: zum einen Docker Desktop und zum anderen eine IDE, in welcher der Code für die Container geschrieben werden kann. Docker Desktop besitzt eine grafische Oberfläche und wird auf Windows und Macintosh benötigt, um Container zu bauen. Auf Linux ist das nicht notwendig, da die Docker Engine dort direkt lauffähig ist. Docker Desktop ist eine virtuelle Maschine, die das Managen der verschiedenen Komponenten auf diesen Systemen möglich macht. (Vgl. inray Industriesoftware GmbH o.A.)
Das Herzstück von Docker sind die Container. Auf der Seite OPC Router von inray Industriesoftware wird ein Container als „[…] eine Standardsoftwareeinheit, die einen Code in all seinen Abhängigkeiten speichert“ (ebd.) beschrieben. Diese Speicherung ermöglicht Docker, problemlos Container auf einem System zu bauen und diese dann auf einem anderen System zu nutzen. Die einzige Vorbereitung, welche dafür getroffen werden muss, ist, Docker auf dem Ziel-Gerät zu installieren. Damit das funktioniert, sind in einem Container folgende Dinge verbaut: der Programmcode, die RunTime-Engines, die Systemtools, die Systembibliotheken und die Einstellungen. Die Vorstufe eines Containers ist das Docker Image. Dieses kann auf der Onlinedatenbank Docker Hub hochgeladen und auf anderen Geräten heruntergeladen werden. Mit dem Befehl „docker run [Imagename]“ wird das Image gestartet und zu einem Container umgewandelt. (ebd.)

2.4.1 Dockerfile

In der Datei „dockerfile“ befinden sich alle Anweisungen für das Erstellen eines Docker Images, die im Folgenden chronologisch erklärt werden:

- FROM muss immer am Anfang eines Dockerfiles stehen. Mit diesem Befehl wird ein bereits fertiges Image definiert, auf das aufgebaut wird. Dieser Befehl legt den Grundstein für das, was der Container später können soll. Alle verwendbaren Images können auf der Internetseite von Docker Hub abgerufen werden. Die Images, die verwendet werden können, reichen von von Firmen veröffentlichten Images, die ganze Programme enthalten wie z.B. nginx oder Python bis hin zu Images, die von Usern für private Projekte erstellt werden. (Vgl. Docker Inc. o.A.)
- WORKDIR gibt das Arbeitsverzeichnis innerhalb des Containers an, in dem alle nachfolgenden Befehle ausgeführt werden sollen. (ebd.)
- RUN wird genutzt, um alle nachfolgenden Anweisungen auszuführen. Mit dem Befehl wird eine neue Schicht oberhalb des in FROM definierten Images erstellt. Der Befehl wird unter anderem. dafür verwendet, Abhängigkeiten für Programme in den Containern zu installieren. Beispielsweise lassen sich mit dem folgenden Befehl alle Bibliotheken bei Python-Applikationen hinter RUN setzen, die mit pip installiert werden: „RUN pip install [Bibliothek]“. An dieser Stelle kann zusätzlich die Version der heruntergeladenen Bibliothek bestimmt werden. (ebd.)
- COPY nimmt Dateien oder ganze Arbeitspfade aus dem angegebenen Pfad vom Computer und kopiert sie in den Pfad des Filesystems des Containers. Der Befehl braucht dementsprechend zwei Argumente. Mit COPY werden die eigens erstellten Programme in den Container kopiert. (ebd.)
- CMD gibt an, welche Befehle innerhalb des Containers ausgeführt werden sollen, sobald dieser startet. Hier kann z.B. festgelegt werden, dass beim Starten des Containers das vorher hinzugefügte Python-Skript ausgeführt werden soll. Ein solcher Befehl kann wie folgt aussehen: „CMD [‚python3‘, ‚main.py‘]“. Der Befehl ist der gleiche, der in Linux verwendet wird, um ein Python-Skript auszuführen. (ebd.)

Das Dockerfile bietet deutlich umfangreichere Möglichkeiten. Die hier dargestellten Befehle beschränken sich auf diejenigen, welche benötigt werden, um ein eigenes Image zu erstellen, welches dann auf anderen Geräten als Container gestartet werden kann.

2.4.2 Befehle zum Bauen und Starten von Containern

Im nächsten Abschnitt soll es um die Befehle gehen, die benötigt werden, um Images zu bauen, sie zu starten und von einem Gerät auf ein anderes zu übertragen. Dabei wird auch hier nur auf Funktionen und Möglichkeiten eingegangen, die für die Drohne benötigt werden. Das sind (in chronologischer Abfolge): docker build, docker push, docker pull und docker run.
Der erste Befehl ist „docker build [Directory]“. Dieser Befehl wird verwendet, um das Image zu erstellen. Er benötigt genau einen Parameter. Dieser ist der Pfad (Directory) des Dockerfiles, das zu einem Image gebaut werden soll. Hier können noch einige zusätzliche Anweisungen mit auf den Weg gegeben werden, die benötigt werden, um die Funktionalität des Images zu gewährleisten. Die erste Anweisung, die mitangegeben werden kann, ist der Name des Images: „-t [Imagename]“. Mit einem vordefinierten Namen kann das Image in den folgenden Schritten angesprochen werden. Allerdings kann mit dieser Variante nicht der nächste Befehl ausgeführt werden. Um ein Image hochladen zu können, braucht es im Namen den Benutzernamen des Accounts des Entwicklers. Das sieht wie folgt aus: „-t [Benutzername]/[Imagename]“. Der Benutzername kann beim Erstellen eines Accounts bei Docker frei gewählt werden. Eine zweite Anweisung wird gegebenenfalls benötigt, um den Container auf anderen Betriebssystemen lauffähig zu machen. Ein Image, welches z.B. unter Windows erstellt wird, kann ohne die folgende Anweisung nicht unter Linux gestartet werden: „--platform=[Plattform]". Jede Plattform hat eine eigene Bezeichnung, die im Docker Wiki nachgelesen werden kann. Wenn ein Image, welches unter Windows gebaut wird, auf Linux laufen soll, muss der Befehl wie folgt aussehen: „--platform=linux/arm64“.
Mit dem Befehl „docker push [Imagename]“ wird das im vorherigen Schritt erstellte Image auf Docker Hub hochgeladen. Durch das Ausführen dieses Befehls wird das Image auf Docker Hub gespeichert und kann dann auf anderen Geräten mit dem folgenden Befehl heruntergeladen werden: „docker pull [Imagename]“.

Im nächsten Schritt muss das Image gestartet werden, um zum Container zu werden. Das geschieht mit folgendem Befehl: „docker run [Imagename]“. Auch hier werden in manchen Fällen zusätzliche Anweisungen benötigt, um die Funktionalität des Containers zu gewährleisten. Wenn z.B. mit einem Sensor, welcher per UART angeschlossen ist, kommuniziert werden soll, ist das nicht direkt möglich. Das liegt daran, dass der Container keine Rechte hat, um mit dem Sensor zu interagieren. Docker Container haben beim Starten aus Sicherheitsgründen nur sehr eingeschränkte Rechte. Es gibt aber verschiedene Möglichkeiten, einem Container das Recht zu geben, mit besagtem Sensor zu interagieren. Die erste Möglichkeit, die eher vermieden werden sollte, ist, den Container als privilegiert zu kennzeichnen. Das kann mit folgendem Zusatz geschehen: „--privileged". Dadurch bekommt der Container das Recht, auf alle Inhalte auf dem Computer zuzugreifen, wodurch wiederrum erhebliche Sicherheitslücken entstehen können. Aus diesem Grund sollte eher die zweite Variante genutzt werden. Diese sieht folgendermaßen aus: „--device=[device-Pfad]/ [device-Pfad]“. Mit diesem Zusatz bekommt der Container nur das Recht, mit dem definierten Gerät zu interagieren.

2.4.3 Netzwerke in Docker

Neben eigens von Nutzern erschaffenen Netzwerken, erschafft Docker beim Starten drei Standardnetzwerke, auf die im Folgenden eingegangen werden soll. Auch hier existieren deutlich mehr Möglichkeiten, als in der Arbeit thematisiert werden.

Das erste Netzwerk ist das Bridge-Netzwerk. In dieses Netzwerk werden standardmäßig alle neu erstellten Container integriert. Es wird genutzt, um Container in Isolation zu betreiben. Hier können sich Container mit anderen Containern verbinden und mit diesen kommunizieren. Docker sorgt dafür, dass von außen nicht auf diese Container zugegriffen werden kann. Die Container können aber auf das Internet zugreifen. Dieses Netzwerk ist eher zur Entwicklung geeignet, da die Container beim Starten jedes Mal eine neue IP-Adresse bekommen. (Vgl. Choudhary 2023)

Das zweite Netzwerk ist das Host-Netzwerk. Wenn ein Container in diesem Netzwerk ist, ist er mit dem Netzwerk der Host-Maschine verbunden. Durch dieses Netzwerk kann dann mit Hilfe der IP des Hosts auf Ports der Container zugegriffen werden. Es findet dann Verwendung, wenn nicht das Networking von Docker sondern das der Host-Maschine verwendet werden soll. Ein Nachteil des Host-Netzwerks ist, dass es

nur unter Linux nutzbar ist. Ein anderer Nachteil ist, dass nicht mehrere Container betrieben werden können, die den gleichen Port verwenden.
Als Letztes soll das None-Netzwerk betrachtet werden. Wie der Name es schon vermuten lässt, gibt es hier kein Netzwerk. Container, die sich hier befinden, können weder auf andere Netzwerke zugreifen, noch können sie mit anderen Containern kommunizieren. (Vgl. Choudhary 2023)

2.4.4 Docker Compose

Docker Compose ist ein Tool, welches es ermöglicht, ein Netzwerk von Containern zu betreiben. Dieses Tool kann mit einem Befehl mehrere Container und Netzwerke gleichzeitig starten, überwachen oder auch wieder stoppen. Es ist besonders nützlich für Anwendungen, bei denen mehr als ein Container benötigt wird. Zusätzlich vereinfacht Docker Compose das Networking dieser Anwendungsfälle, indem vordefiniert werden kann, in welchem Netzwerk sich welcher Container befindet. Auch können mit diesem Tool neue Netzwerke erstellt werden, die an die eigenen Bedürfnisse angepasst werden können. Im Zentrum von Docker Compose steht das sogenannte „docker-compose.yml“-File. In dieser Datei werden alle Einstellungen vorgenommen, die beim Starten des Container-Netzwerkes ausgeführt werden sollen. Im Folgenden findet sich ein Beispiel für ein „docker-compose.yml“-File:

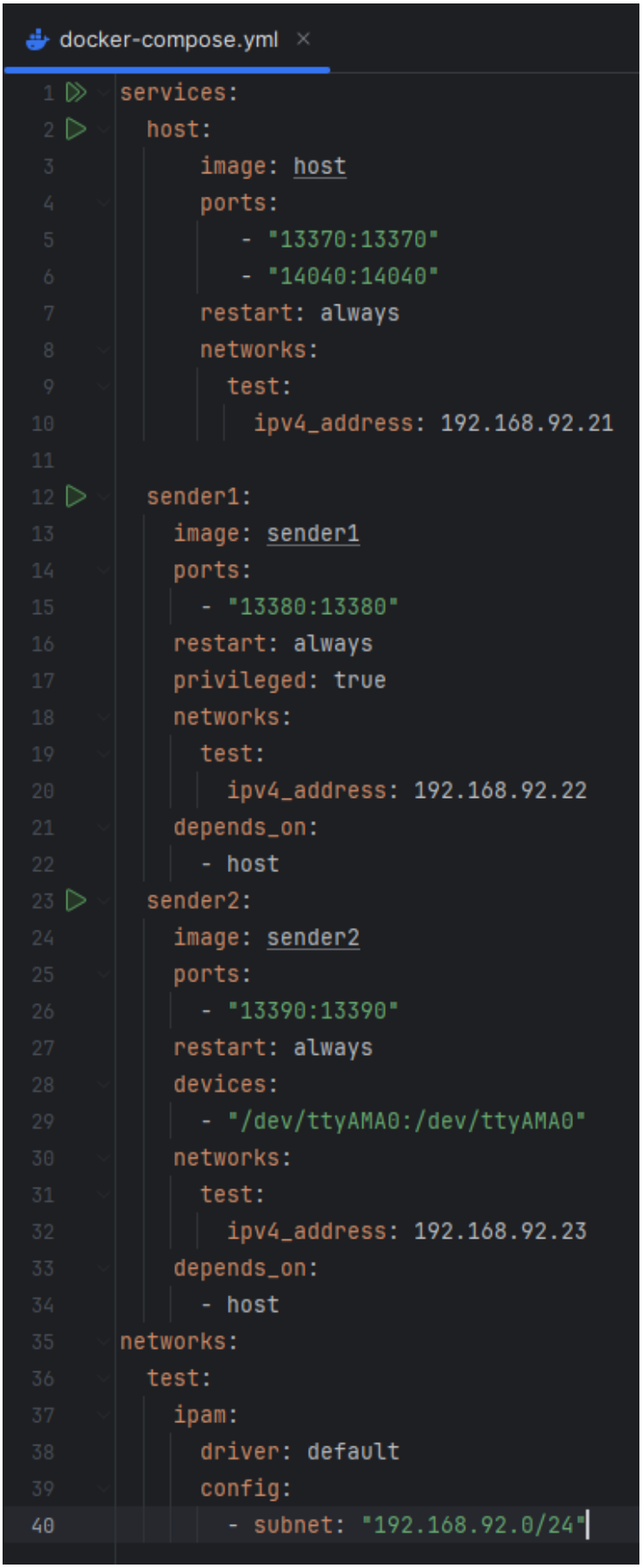


Abbildung 2: "docker-compose.yml“-File

In der Abbildung ist ein Docker Compose File für ein einfaches Netzwerk zu sehen. Das Netzwerk besteht aus drei Containern. Dabei ist ein Container der Host, der von den beiden anderen Containern Daten empfängt, die z.B. von Sensoren aufgenommen werden. In diesem Beispiel besteht die Datei aus zwei unterschiedlichen Abschnitten. Im „services“-Abschnitt werden die einzelnen Container, die gestartet werden sollen, konfiguriert. Im zweiten Abschnitt „networks“ wird ein neues Netzwerk kreiert, in das die Container eingebunden werden.

Im ersten Abschnitt bekommen die Container zur Konfiguration zunächst Namen, unter welchen sie laufen. Im Beispiel hier sind das „host“, „sender1“ und „sender2“. Im darauffolgenden Schritt wird das jeweilige Image bestimmt, welches verwendet werden soll. Diese haben in diesem Beispiel den gleichen Namen wie die später laufenden Container. In der drauffolgenden Zeile werden die jeweiligen Ports konfiguriert, welche nach außen hin freigegeben werden sollen. Beim Host werden dabei zwei Ports eingerichtet, da der Host die Daten der Sender auf unterschiedlichen Ports empfangen soll. In der nächsten Zeile wird bestimmt, was geschieht, wenn der Container stoppt. Mit der Einstellung „always“ wird festgelegt, dass der Container immer wieder neu gestartet wird, sobald er anhält. Mit dem „network flag“ werden die Container dem selbst erstellten Netzwerk zugeteilt. Dabei wird jedem Container eine IP-Adresse zugewiesen. Einige Container bekommen zusätzliche Einstellungen, da sie verschiedene Aufgaben zu erfüllen haben. Mit der Einstellung „depends_on“ wird die Reihenfolge festgelegt, in der die Container gestartet werden. Im Beispiel sollen beide Sender erst nach dem Host starten. Auch sind im Beispiel verschiedene Berechtigungen zur Kommunikation zwischen Sensoren und Containern zu sehen. Sender 1 wird privilegiert, damit er uneingeschränkten Zugriff auf alles auf der Host-Maschine bekommt. Bei Sender 2 wird nur ein einzelnes Gerät freigegeben, mit dem der Container kommunizieren kann.

Der zweite Abschnitt des Compose File beschäftigt sich mit dem Netzwerk, welches erschaffen werden soll. Im Beispiel hier wird nur ein Netzwerk kreiert, es besteht aber auch die Möglichkeit, mehrere Netzwerke zu erstellen. Der Name des hier erschaffenen Netzwerkes ist „test“. Im zweiten Schritt werden persönliche ipam-Einstellungen für das Netzwerk vorgenommen. Zunächst wird der Treiber des Netzwerkes bestimmt. Hier ist das der Default driver. Im Anschluss daran werden mit den in der Abbildung erkennbaren Befehlen die möglichen IP-Adressen des Netzwerkes eingegrenzt. (Vgl. Docker Inc. o.A.)

Mit diesem fertigen Docker Compose File kann ein Netzwerk mit mehreren Containern mit einem Befehl gestartet werden. Dazu muss in dem Arbeitspfad, in dem sich das Docker Compose File befindet, der Befehl: „docker compose up“ eingegeben werden. Damit werden Container und Netzwerk automatisch gestartet und nicht vorhandene, aber benötigte Images heruntergeladen. Soll das Netzwerk wieder gestoppt werden, muss in demselben Arbeitspfad der Befehl „docker compose down“ ausgeführt werden. Dadurch werden alle zuvor gestarteten Container und selbst erstellten Netzwerke gestoppt.

3 Entwicklung der Drohne

In diesem Abschnitt wird der Weg der konkreten Drohne inklusive Sensorkonzept und Kommunikation über 5G von der Planung bis zur Umsetzung beschrieben. Die in diesem Punkt genutzten Programme finden sich im digitalen Anhang der Arbeit.

3.1 Bau der Drohne

3.1.1 Vorbetrachtungen

Die Drohne spielt innerhalb der Aufgabenstellung eine zentrale Rolle. Aus diesem Grund ist es essenziell, dass bei der Drohne alle Parameter richtig gewählt werden. Wichtige Parameter sind dabei z.B. die Größe, die Flugzeit und die Menge an Payload, der mitgeführt wird. Auch muss darauf geachtet werden, dass die einzelnen Anforderungen sinnvoll gewichtet werden.
Die erste Frage, die vor Anschaffung der Technik beantwortet werden muss, ist, ob eine bereits fertige Drohne gekauft wird oder ob sie selbst gebaut werden soll. Auf den ersten Blick erscheint das Bauen einer Drohne deutlich komplizierter, da Drohnen hohe Hard- und Softwareansprüche stellen, um überhaupt fliegen zu können. Da Drohnen in den letzten Jahren aber stark an Beliebtheit gewonnen haben, ist die Einstiegshürde in diesem Sektor stark gefallen. Von Fertigbausets, welche nur noch verschraubt und verlötet werden müssen, bis hin zum einzelnen Bauteil gibt es mittlerweile alles online zu kaufen. Auch bei der Software wurde aufgerüstet. Vorgefertigte Open-Source-Lösungen, die meist schon vorinstalliert sind, sorgen dafür, dass oftmals nur wenige Werte im Konfigurator geändert werden müssen, um mit der Software zu arbeiten. Das eigene Schreiben eines Programms, welches die Werte der Fernbedienung verarbeitet und die Motoren steuert, ist nicht mehr notwendig. Auch das Aufbauen der Drohne stellt dank zahlreicher Anleitungen im Internet kaum noch eine Herausforderung dar.

Die fertigen Drohnen haben in den letzten Jahren ebenfalls an Qualität gewonnen, während der Preis gesunken ist. Einfache Drohnen können online bereits für unter 100€ erworben werden. Qualitativ hochwertige Drohnen halten ihre Preise weiterhin hoch, dafür bieten sie aber einige Features, die von Geräten im unteren Preissegment nicht erwartbar sind. Ein Beispiel für ein solches Feature ist die automatische Verfolgung von Personen, welche vor allem für Videoaufnahmen genutzt wird. Ein Beispiel für eine Firma, die qualitativ hochwertige Drohnen herstellt, ist Da-

Jiang Innovations Science and Technology Co., Ltd, kurz DJI, aus China. Welche Eigenschaften die Drohne für diese Arbeit haben muss, wird nachfolgend thematisiert. Bevor entschieden werden kann, ob eine Drohne gekauft oder selbst gebaut wird, müssen zunächst die Anforderungen an die Drohne geprüft werden. Das Ziel ist es, eine Drohne in einem Hochregallager fliegen zu lassen, dort die einzelnen Fächer zu scannen und den Inhalt zu überprüfen. Dieser Flug soll autonom stattfinden und über das bestehende System gesteuert werden können. Daher muss die Drohne zunächst innerhalb des Regallagers fliegen können. Das heißt konkret, dass sie zwischen den Regalen entlang fliegen können muss. Abbildung 3 zeigt das untere Ende einer Schiene, die zwischen den Regalen entlangläuft:

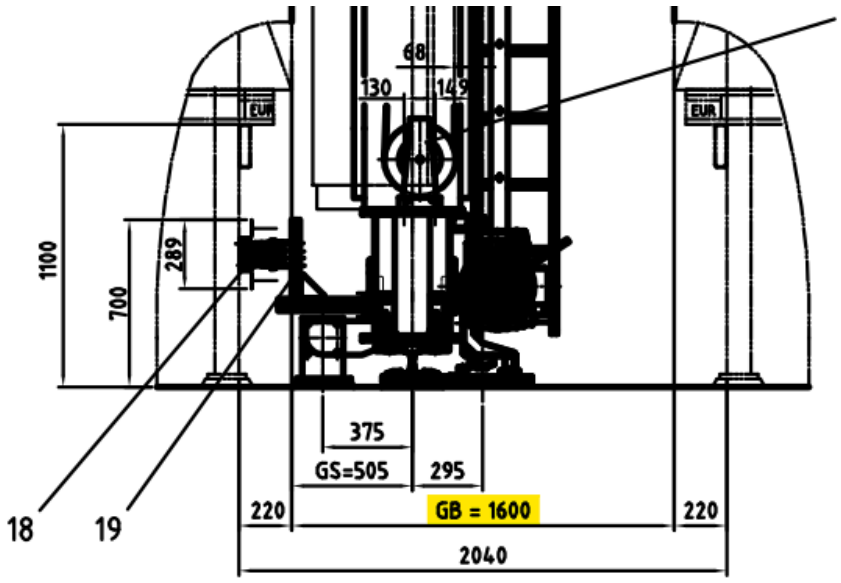


Abbildung 3: Maße des Hochregallagers

Zwischen den einzelnen Regalen ist 1600 mm Platz (siehe Markierung in Abbildung 3). Die Drohne darf dementsprechend nicht zu groß werden, da sie ansonsten zwischen den Regalen hängen bleiben könnte. Sowohl eine selbstgebaute als auch eine gekaufte Drohne können so gebaut bzw. gewählt werden, dass sie klein genug sind, um zwischen die Regale zu passen. Drohnen, die frei verkäuflich sind, sind in den meisten Fällen kompakt gebaut, um die gesetzliche Grenze von 250 Gramm nicht zu überschreiten. Ab 250 Gramm ist eine Fernpilotenlizenz nötig, um die Drohne fliegen zu können, wodurch weniger Käufer angesprochen werden.

Ein weiterer wichtiger Aspekt, der betrachtet werden muss, ist der Payload. Der Begriff umfasst alles, was mit der Drohne mitfliegt. In diesem Fall hat die Drohne nur die Aufgabe, das Regal abzuscannen und muss keine Waren transportieren. Dementsprechend wird der Payload klein gehalten. Die Drohne muss lediglich die

Sensorik mit sich führen, die für das autonome Fliegen benötigt wird. Beim Payload hat eine selbstgebaute Drohne einen klaren Vorteil gegenüber einer gekauften. Bei der selbstgebauten Drohne können die Teile so gewählt werden, dass der vorgesehene Payload optimal mittransportiert werden kann. Bei der gekauften Drohne fallen gleich mehrere Probleme an. Erstens sind die Bauteile bei vorgefertigten Drohnen auf das Gewicht der Drohne abgestimmt. Zusätzliches Gewicht kann zu Schwierigkeiten beim Fliegen führen. Das zweite Problem ist, dass sich der Payload schlecht an der gekauften Drohne anbringen lässt, da sie nicht für diesen Verwendungszweck ausgelegt sind.

Die Autonomie der Drohne ist ein weiterer zentraler Faktor. Die Drohne muss von externen Quellen gesteuert werden können. In diesem Fall ist keine Fernbedienung von einer Person vorgesehen, sondern die Fernsteuerung über einen externen Computer, der die Wegfindung und anderes für die Drohne übernimmt. Auch hier setzt sich die selbstgebaute Drohne gegen die gekaufte durch. Wie bereits erwähnt, wird bei selbstgebauten Drohnen oftmals auf Open-Source-Lösungen gesetzt, welche meist gut dokumentiert sind und Möglichkeiten bieten, Änderungen daran vorzunehmen. Gekaufte Drohnen sind mit einer eigenen, firmenspezifischen Software ausgestattet, auf die nur schwer zugegriffen werden kann, da die Hersteller aufgrund des Marktwettbewerbs den Zugriff auf die Software in den meisten Fällen einschränken. Weitere Faktoren, die mitberücksichtigt werden können, bei der Auswahl aber eine eher untergeordnete Rolle spielen, ist z.B. die Möglichkeit, die einzelnen Komponenten zu reparieren.

Insgesamt wird deutlich, dass das Bauen einer Drohne für dieses Projekt mehr Vorteile zählt als das Kaufen eines fertigen Produkts. Daher wird die Drohne für das Projekt gebaut. Der ausschlaggebende Punkt ist hierbei die Flexibilität, die eine eigene Drohne bietet. Sollte sich beispielsweise im Laufe des Projekts herausstellen, dass die gewählte Sensorik nicht die Ergebnisse liefert, die verlangt werden, kann sie noch angepasst werden. Ein weiterer Vorteil ist, dass man als Bauer der Drohne an spezifischer Expertise gewinnt, die später nützlich werden kann, wenn zum Beispiel Reparaturen durchgeführt werden müssen. Allerdings ist bei dieser Lösung mit Komplikationen hinsichtlich des Flugverhaltens zu rechnen, da fertigte Produkte von Anfang an über eine gut optimierte Software verfügen, die beispielsweise das ruhige Stehen in der Luft erlaubt. Sollte diese Fähigkeit oder eine andere relevant werden,

muss sie durch zusätzliche Hardwarekomponenten oder Softwareeinstellungen erst noch realisiert werden.

3.1.2 Drohnenarten

Bevor die Drohne gebaut und die Bauteile bestellt werden können, muss eine Drohnenart gewählt werden. Aufgrund seiner Bewegungsfreiheit im Raum, fällt die Entscheidung auf einen Multicopter. Wie aus dem Wort bereits hervorgeht, handelt es sich hierbei um ein Fluggerät, das mit mehreren Motoren in der Luft gehalten wird. Typischerweise hat ein Multicopter vier Motoren, allerdings gibt es auch Drohnen mit zwei, drei, sechs, acht oder mehr Triebwerken. Ab einer gewissen Anzahl an Motoren ist jedoch kein Vorteil mehr erkennbar, sodass mehr Motoren nicht gleich besser sind. Um eine Bauart für die Drohne in dieser Arbeit festzulegen, müssen zunächst die Vor- und Nachteile der einzelnen Bauarten überprüft werden. (Vgl. Premium-Modellbau 2020)

Die erste Bauform, die betrachtet werden soll, ist der klassische Quadrocopter. Hierbei handelt es sich um eine Drohne mit vier Motoren. Von diesen drehen sich zwei im Uhrzeigersinn und zwei entgegen dem Uhrzeigersinn, damit sich die Kräfte der Motoren in der Luft ausgleichen können. Dadurch kann sich die Drohne auch in der Luft auf der Stelle drehen. Vorteile dieser Bauform sind das stabile Flugverhalten und die Anpassungsfähigkeit hinsichtlich des Aufbaus. Durch die geringe Motorenanzahl können verschiedene Größen umgesetzt werden: von kleinen Drohnen mit einer Propellergröße von zwei bis drei Zoll bis hin zu großen Drohnen mit bis zu 15 Zoll. (ebd.)

Eine andere mögliche Bauform ist der Hexacopter. Hierbei handelt es sich um einen Multicopter mit sechs Motoren. Von Vorteil ist hier, dass aufgrund der größeren Motorenanzahl mehr Payload mitgeführt werden kann. Das höhere Gesamtgewicht sorgt für ein ruhigeres Flugverhalten und aufgrund der erhöhten Redundanz kann die Drohne auch in der Luft bleiben, sollte ein Motor ausfallen. (ebd.)

Die letzte Bauform, die hier kurz Erwähnung finden soll, ist der Octocopter, welcher mit zwei Motoren mehr gebaut wird. Die Vorteile sind die gleichen wie beim Hexacopter und werden durch die acht Motoren noch verstärkt. Nachteilig ist hier allerdings, dass die Drohne durch die vielen Triebwerke größer wird. (ebd.)

Da die Vorteile des Hexa- und Octacopters für die Aufgabenstellung nicht benötigt werden, wird ein Quadrocopter gewählt. Dass zwischen den einzelnen Regalen nur

wenig Platz ist, bestärkt die Wahl einer Drohnenbauform, die weniger Motoren besitzt. Die damit einhergehende geringere Absturzsicherheit (vgl. Baumgärtel/Landrock 2018) ist für die Aufgabe der Drohne nicht so relevant, dass sie eine andere Drohnenform rechtfertigen würde.

3.1.3 Bestandteile einer Drohne

Im zweiten Schritt müssen alle Komponenten zusammengestellt werden, die für den Bau der Drohne notwendig sind. Dazu werden im Folgenden die wichtigsten Komponenten aufgezählt, wichtige Parameter betrachtet und ihre Funktionen innerhalb der Drohne erläutert.

Ein sehr wichtiger Bauteil ist der Frame der Drohne, der Rahmen, welcher Form und Größe der Drohne vorgibt. Hierbei ist wichtig, dass dieser an die Propeller- und Motorgröße angepasst wird. Daneben sollte er möglichst leicht und stabil sein, um die Motoren nicht zusätzlich zu belasten. Dazu wird ein Rahmen aus Carbon gewählt.

Bei den Motoren wird auf bürstenlose Motoren gesetzt, da diese besonders langlebig sind und eine geringe thermische Belastung aufweisen (vgl. Baruschke 2021). Es gibt sie in verschiedenen Größen und für verschiedene Anwendungszwecke. Bei der Auswahl von bürstenlosen Motoren müssen mehrere Punkte beachtet werden: die Größe der Motoren, deren KV-Zahl und mit welcher Art von Akku sie betrieben werden können. Die Größe ist wichtig, da sie vorgibt, welche Propeller zusammen mit dem Motor verwendet werden können. Die KV-Zahl gibt die Umdrehungszahl pro Volt im Leerlauf an. Ein Motor mit einer KV-Zahl von 2200 dreht sich 2200 Mal die Minute, wenn an diesen eine Spannung von einem Volt angelegt wird. Ein Motor mit einer höheren KV-Anzahl dreht sich dementsprechend schneller und hat damit mehr Kraft. Die KV-Zahl bewegt sich für die meisten Motoren zwischen 500 und 3000 KV. Motoren, die mit höherer Spannung betrieben werden, haben meist kleinere KV-Zahlen. Alle wichtigen Werte lassen sich im Datenblatt des Motors finden. Im Folgenden wird beispielhaft der Ausschnitt des Datenblatts für den „T-Motor F60 Pro V FPV Race Brushless Motor 1750kv V5“ (Premium-Modellbau o.A.) betrachtet, um zu zeigen, wie die Werte voneinander abhängen:

Propeller	Type	Throttle	Voltage (V)	Current (A)	RPM	Thrust (g)	Power (W)	Efficiency (g/W)
T-MOTOR T5147-3	1750KV	20%	25.2	2.1	11132	227.0	53.6	4.24
		40%	25.2	6.6	16928	558.6	165.2	3.38
		60%	25.1	13.7	21409	910.7	342.4	2.66
		80%	25.0	24.3	26122	1403.4	606.9	2.31
		100%	24.8	40.5	30527	1882.5	1002.8	1.88
	1950KV	20%	25.2	2.6	11979	257.8	65.9	3.91
		40%	25.2	8.6	18222	635.8	215.4	2.95
		60%	25.1	17.6	22540	996.7	441.3	2.26
		80%	24.9	30.5	27582	1559.5	758.4	2.06
		100%	24.7	49.3	31401	1990.4	1215.8	1.64
	2020KV	20%	25.2	2.7	12105	265.1	67.5	3.93
		40%	25.2	9.6	18557	667.1	240.2	2.78
		60%	25.0	19.8	22983	1054.7	494.3	2.13
		80%	24.9	33.7	28069	1636.1	837.9	1.95
		100%	24.6	52.7	31628	2025.5	1297.1	1.56
	2550KV	20%	16.8	2.9	10441	188.3	49.2	3.83
		40%	16.7	9.2	16163	492.1	153.4	3.21
		60%	16.6	18.3	20173	783.9	303.6	2.58
		80%	16.5	31.6	24339	1199.9	520.9	2.30
		100%	16.3	50.7	28250	1629.4	824.6	1.98

Abbildung 4: Datenblatt eines T-Motors (übernommen von: Premium-Modellbau o.A.)

Ganz links in Abbildung 4 in der Tabelle ist die Art von Propeller angegeben, auf den sich die folgenden Werte beziehen. Die Werte, die im Datenblatt vermerkt sind, wurden durch Tests mit verschiedenen Propellern ermittelt. An der zweiten Zeile ist zu sehen, dass diese Propeller auch für baugleiche Motoren mit anderen KV-Zahlen getestet wurden. Für dieses Beispiel betrachten wir den Motor mit der KV-Zahl 1750. In der dritten Spalte wird der Gas-Wert angegeben, auf den sich die Werte Spannung (Spalte 4) und Stromstärke (Spalte 5) beziehen. Aus der Spannung lässt sich ablesen, dass dieser Motor für einen 6S-Akku ausgelegt ist. Woran man das erkennt, wird am Ende dieses Unterpunkts erklärt. Die Stromstärke spielt bei der Wahl des ESC(Reglers) eine entscheidende Rolle. RPM in Spalte 6 gibt die Drehzahl pro Minute an, während in Spalte 7 die Schubkraft visualisiert wird. Die Schubkraft gibt in Gramm an, wie viel Kraft der Motor bei entsprechender Leistung aufwenden kann. Hier gilt die Faustregel, dass alle Motoren zusammen bei maximaler Leistung doppelt so viel Schub aufbringen sollten, wie die Drohne am Ende schwer ist. In der letzten Spalte wird die Leistung des Motors in Watt angegeben. (Vgl. Premium-Modellbau 2020/ o.A.)

Ein weiteres wichtiges Bauteil sind die Propeller. Hierbei müssen die Größe, die Steigung, die Anzahl an Flügeln pro Propeller und die Ausrichtung beachtet werden. Die Größe der Propeller wird meist in Zoll angegeben. Unterschiedliche Propellergrößen sind für unterschiedliche Anwendungszwecke geeignet. Kleine Propeller werden zum Beispiel häufig für Renn-Drohnen verwendet. Mit zunehmender Größe der Propeller steigt die Effizienz. Die Steigung eines Propellers gibt an, wie viel Strecke ein Propeller bei einer Umdrehung zurücklegt. Eine höhere Steigung erfordert mehr Leistung, allerdings erhöht diese auch die Geschwindigkeit der Drohne. Klassischerweise besitzen Propeller drei Flügel, es gibt aber auch Propeller, welche nur zwei oder mehr als drei Flügel besitzen. Mehr Flügel machen den Propeller zunehmend träger. Daher werden mehr Propeller oftmals nur bei kleineren Drohnen eingesetzt. Wichtig ist abschließend noch, dass ein Gleichgewicht bei den Drehrichtungen geschaffen wird. (ebd.)

Der Regler, auch ESC („Electronic Speed Controller“) genannt, hat die Aufgabe, die Motoren zu steuern. Damit das gelingt, bekommt der Regler Vorgaben von der Flugsteuerung, wie sich die Motoren zu drehen haben. Man unterscheidet zwischen zwei verschiedenen Bauformen. Eine Möglichkeit besteht darin, für jeden Motor einen einzelnen ESC zu verwenden. Ebenso ist es möglich, einen etwas größeren 4in1 ESC zu nutzen, welcher alle Motoren auf einmal ansteuern kann. Wichtig ist, darauf zu achten, dass der ESC genug Strom aushalten kann. Im Datenblatt des Beispielmotors in Abbildung 6 ist beispielsweise ablesbar, dass der dort abgebildete T-Motor bei 100% Schubkraft 40,5A benötigt. Der ESC muss dementsprechend diesen 40,5A standhalten können. (ebd.)

Ebenso wichtig für die Drohne ist der Akku. Er macht einen Großteil des Gewichts der Drohne aus und bestimmt, welche Motoren und welche ESCs verbaut werden können. Wichtige Werte beim Akku sind die Zellenzahl, die Kapazität und die Entladerate. Der Akku besteht aus mehreren Zellen, die jeweils eine Spannung von 3,7V abgeben. Die Zellenzahl gibt die Spannung an, die der Akku insgesamt abgeben kann. Ein 4S-Akku, ein Akku mit vier Zellen, besitzt zum Beispiel eine Nennspannung von 14,8V. Die Kapazität des Akkus gibt die Menge an Energie an, die er beim Aufladen speichern kann. Die Entladerate informiert darüber, wie viel Strom der Akku abgeben kann und wird meist in C angegeben. Zur Berechnung der Entladerate wird der C-Wert mit der Kapazität multipliziert. Ein 5000mAh Akku, der einen C-Wert von 30 besitzt, hat beispielsweise eine Entladerate von 150A. Im RC-Bereich werden meistens LiPo-

Akkus verwendet. Das liegt daran, dass diese Art von Akkus eine der wenigen Arten ist, deren Entladerate hoch genug ist, um die bürstenlosen Motoren zu versorgen. Ein erneuter Blick auf das Beispiel in Abbildung 6 zeigt, dass ein Motor bei maximaler Auslastung über 50A ziehen kann. Hier müsste der Akku dementsprechend über 200A liefern, um die vier Motoren maximal nutzen zu können. (ebd.)
Das letzte für die autonom fliegende Drohne relevante Bauteil ist die Flugsteuerung (bzw. „Flight Controller“ oder FC). Der FC ist das „Gehirn“ der Drohne. Auf ihm läuft die Software, die die Daten der Sensoren auswertet, um damit die Motoren zu steuern. Die Sensorik umfasst ein Gyroskop und einen Beschleunigungssensor, welche meist direkt auf dem FC verbaut sind. Andere Sensorik wie z.B. GPS kann in den meisten Fällen problemlos nachgerüstet werden. Der FC steuert die Motoren nicht direkt, sondern schickt Signale an den oder die ESCs. Für gewöhnlich bekommt der FC von einem Empfänger die Werte der Fernsteuerung übermittelt. Empfänger und Fernsteuerung werden hier nicht weiter beleuchtet, da sie im finalen Aufbau zugunsten der Autonomie der Drohne nicht mehr genutzt werden sollen. (ebd.)

3.1.4 Auswahl konkreter Bauteile

Im Folgenden werden konkrete Teile für die Drohne bestellt. Es gibt keine direkte Budget-Grenze. Allerdings wird dazu angeraten, das Preis-Leistungs-Verhältnis insbesondere bei Bauteilen, die keine besonderen Ansprüche haben, mitzuberücksichtigen.
Der erste zu betrachtende Punkt ist die Spannung, mit der die Drohne operieren soll. Zur Auswahl steht ein 4S-Akku mit 14,8V und ein 6S-Akku mit 22,2V. Da zu diesem Zeitpunkt noch nicht klar ist, wie sich die Drohne im Flug verhalten wird, kann noch keine endgültige Kaufentscheidung an dieser Stelle getroffen werden. Um sich beide Optionen offen zu halten, werden alle folgenden Bauteile so gewählt, dass sie sowohl mit 4S als auch mit 6S kompatibel sind.
Für die Propeller werden möglichst große Flügel gewählt, da im vorhergehenden Punkt aufgezeigt wird, dass diese effizienter sind. Zwei Faktoren, welche die Größe der Propeller begrenzen, sind der Platz zwischen den Regalen und die Größe des Rahmens. Typische Rahmengrößen für Quadrocopter sind fünf Zoll und alles darunter. Einen passenden Rahmen zu finden, ist nicht leicht, da die Auswahl begrenzt ist. Nach langer Recherche wird der „TBS Source One V5 5" FPV Frame“ von Copterfarm gekauft. Er wird ausgewählt, da es für ihn ein Upgradekit gibt, welches dafür sorgt,

dass er mit sieben Zoll großen Propeller genutzt werden kann. Der Frame mitsamt des Upgradekits wird in der folgenden Abbildung dargestellt:

Abbildung 5: 7-Zoll-Rahmen „TBS Source One V5 5" FPV Frame“ (übernommen von: Copterfarm o.A.)

In Abbildung 5 ist zu erkennen, dass die Arme des Rahmens sehr lang sind und damit genug Platz für sieben Zoll große Propeller bieten.

Für den Motor wird der „FETtec FT2207 1100 KV 10S“ genutzt. Dieser Motor ist eine Spezialanfertigung, die von T-Motor für FETtec hergestellt wird. Aus diesem Grund gibt es noch kein Datenblatt für ihn. Laut Angaben des Herstellers wird das Datenblatt womöglich nachgereicht. Aus den auf der Produktseite angegebenen Informationen im Namen des Produkts kann aber abgeleitet werden, dass dieser Motor viele Kriterien erfüllt, die gefordert sind. Die 22 in FT2207 ist der Durchmesser und die 07 die Höhe des Stators in cm. Diese Maße sind passend für den ausgewählten Rahmen. 1100KV im Namen gibt die Motordrehzahl an. Sie ist im Durchschnitt etwas niedriger als bei anderen Motoren und hat zur Folge, dass der Motor sich bei geringeren Spannungen langsamer dreht und dementsprechend weniger Strom verbraucht. Die letzte Information im Namen, 10S, gibt an, dass der Motor für ein 10S-Akku ausgelegt ist. Allerdings kann der Motor auch mit Akkus verwendet werden, die weniger als 37V (10S) liefern. Der ausgewählte Motor wird in der folgenden Abbildung dargestellt:

Abbildung 6: Drohnenmotor „FETtec FT2207 1100 KV 10S“ (übernommen von: FETTEC UG o.A.)

Im Vergleich zu ähnlichen Motoren bringt diese Bauform mit den passenden Propellern deutlich mehr Kraft auf, als benötigt wird. Falls es zu Problemen beim Abheben der Drohne aufgrund des Gewichts kommen sollte, kann die zugeführte Spannung erhöht werden, um den Motoren mehr Leistung zu entnehmen.

Passend zum Motor werden die „Gemfan 7042 7x4.2 Flash 2 Blatt Propeller Schwarz 7 Zoll“-Propeller gekauft. Auch bei diesen lassen sich alle relevanten Informationen aus dem Namen entnehmen. Die Propeller sind sieben Zoll groß und haben eine Steigung von 4.2. Für diese Größe ist die Steigung eher niedrig gehalten. Zwar bringt eine höhere Steigung mehr Schubkraft, gleichzeitig wird mit einer höheren Steigung aber auch mehr Strom aufgenommen. Hier wird zugunsten einer längeren Flugzeit auf eine höhere Schubkraft verzichtet. Aus ebendiesem Grund wird auch bei der Anzahl der Flügel gespart. Mit zwei Flügeln ist die Drohne effizienter.

Während für die Auswahl des Motors und der Propeller die fehlenden Datenblätter kein Problem darstellen, wird man bei der Wahl des/der ESC(s) vor eine Herausforderung gestellt, da aus dem Namen des Motors nicht hervorgeht, wie viel Strom bei maximaler Leistung gezogen wird. Daher wird der ausgewählte Motor erneut mit ähnlichen Modellen verglichen. Die Vergleichsmotoren ziehen bei maximaler Auslastung ca. 30A. Daher wird/werden (ein) ESC(s) gesucht, die mindestens 40A aushalten.

Ausgewählt wird der „SpeedyBee F405 V3 BLS 50A 30x30 FC&ESC Stack“ des Unternehmens Zhuhai KuaiFeng Technology. Ein „Stack“ ist eine beliebte Bauform, da sie sehr kompakt ist und damit Platz spart. Das Bauteil besteht aus einem FC, der sich oberhalb befindet und einem 50A 4in1ESC, der unter dem FC angebracht ist. Wichtig ist an dieser Stelle zu erwähnen, dass das ESC nicht insgesamt 50A herausgibt, sondern 50A an jedem seiner Ausgänge ausgeben kann. Mit diesem Wert erfüllt er die vorher gesetzte Vorgabe von auszuhaltenden 40A im vollen Maß. Er liefert mehr Strom als die Motoren bei maximaler Leistung benötigen. Bei dem hier verbauten FC handelt es sich um den Speedybee F405 V3. Da an diesen keine Anforderungen gestellt

werden, ist die Art nicht relevant. Allerdings können die zusätzlichen Features wie das eingebaute Barometer und die vier verwendbaren UARTs später von Nutzen sein. Das Stack ist in der folgenden Abbildung dargestellt:

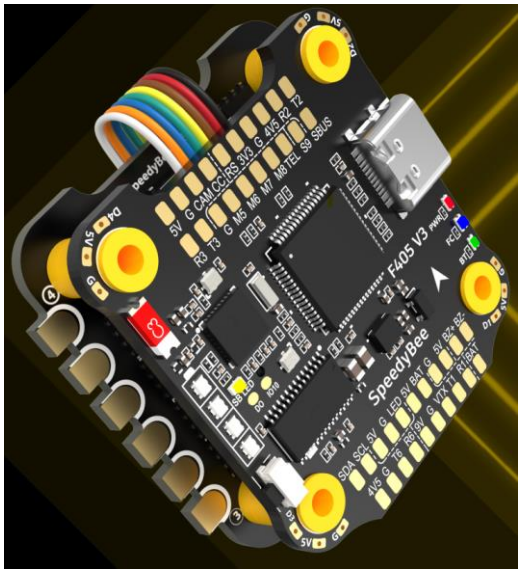


Abbildung 7: ESC-FC-Kombination „SpeedyBee F405 V3 BLS 50A 30x30 FC&ESC Stack“ (übernommen von: Zhuhai KuaiFeng Technology Co. o.A.)

In der Abbildung ist erkennbar, dass die beiden Komponenten FC und ESC über ein Flachbandkabel miteinander verbunden sind. Die kleinen goldenen Pats sind dafür da, die Sensorik an den FC anzuschließen. Die großen Pats am ESC sind für die Motoren vorgesehen.

Zum Testen werden einige weitere Komponenten besorgt, die für den finalen Aufbau nicht zwingend benötigt werden oder die später durch andere Bauteile ersetzt werden. Dazu zählen der Empfänger „ELRS Lite RX|2.4GHz Flat Antenna“ der Firma BETAFPV, die Fernsteuerung „FrSKY Taranis X9D plus“ von FrSky und ein Akku der Firma Tatu mit den Werten 4S, 1550mAh und einer Entladerate von 120C.

3.1.5 Aufbau der Drohne

Für den Aufbau der Drohne sind mehrere Kriterien zu beachten: Zunächst müssen die Motoren an die ihnen zugehörigen Pads am ESC angelötet werden. Welches der drei Kabel dabei an welche der drei Pats angelötet wird, ist irrelevant. Der zweite zu beachtende Punkt ist, dass an die Verbindung zum Akku ein Kondensator mitverlötet wird. Dieser hat die Aufgabe, mögliche Spannungsspitzen beim Einschalten der Drohne zu dämpfen, damit die Elektronik keinen Schaden nimmt. Weiterhin ist darauf zu achten, dass kein Gegenstand in den Weg der Rotoren gelangt. Kabel müssen entweder gekürzt oder mit Kabelbindern fixiert werden. Schließlich müssen die vier Propeller an die ihrer Drehrichtung entsprechenden Motoren angeschlossen werden.

3.1.6 Positionsbestimmung der Drohne

Um die Drohne präziser steuern zu können, muss zu jedem Zeitpunkt eindeutig bestimmt werden können, wo sich die Drohne innerhalb des Lagers befindet. Um das umzusetzen, gibt es verschiedene Ansätze. Eine Möglichkeit ist es, mit Hilfe von Funk und von TDoA („Time Difference of Arrival“) die Position der Drohne zu bestimmen. Die andere Möglichkeit ist, die Drohnenposition mit Hilfe von Sensoren, welche an der Drohne montiert sind, zu überwachen.
Die erste Möglichkeit mit Funk und TDoA bietet den Vorteil, dass zur Positionsüberwachung das 5G-Campusnetz genutzt werden kann, an dem im Rahmen des 5G POUST Projekts geforscht wird. TDoA ist ein Verfahren zur Lokalisierung von Emittern. Im Mittelpunkt des Verfahrens steht die Zeit, die zwischen dem Aussenden und dem Empfangen eines Signals vergeht. Um die Position des Empfängers konkret bestimmen zu können, muss sie mehrfach vorliegen. In der folgenden Abbildung wird deutlich, wie das gemeint ist:

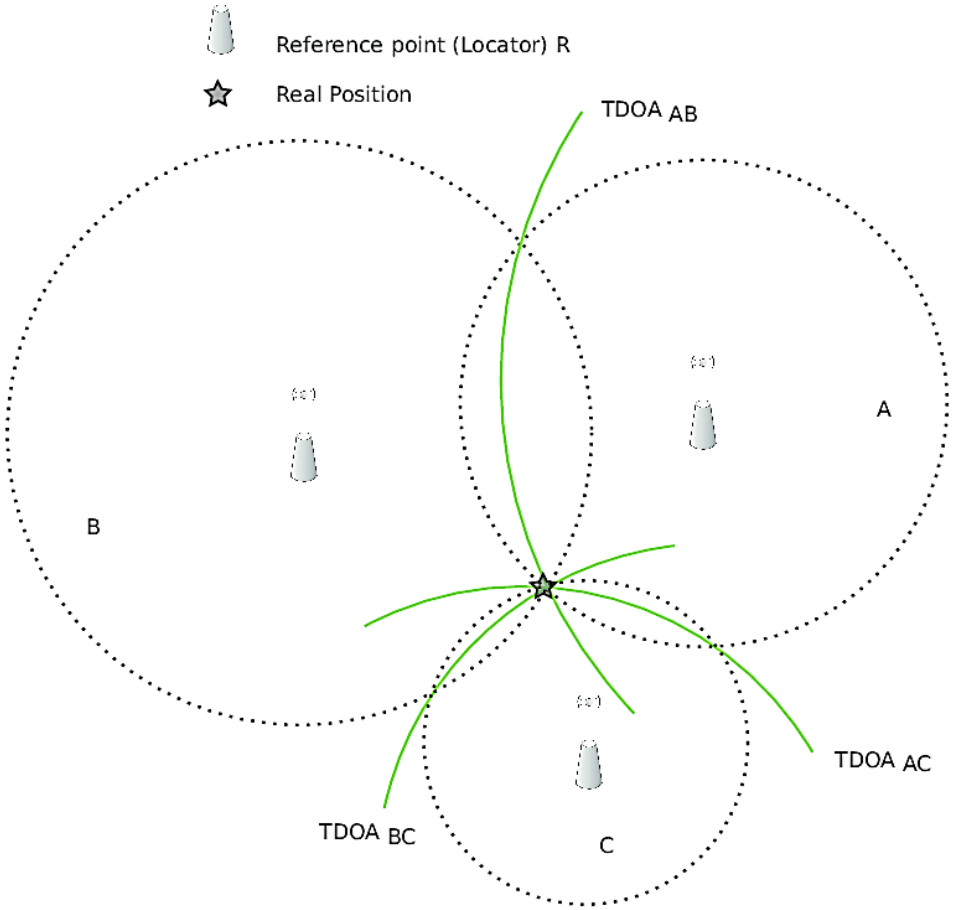


Abbildung 8: TDoA-Prinzip (übernommen von Alarifi et al. 2016)

In der Abbildung ist erkennbar, dass es einen Sender und mehrere Empfänger gibt. Die Zeit, die das Signal vom Sender zu den einzelnen Empfängern braucht, spannt einen Kreis um die Empfänger auf. Der Radius dieser Kreise entspricht der Signallaufzeit vom Signal zum zugehörigen Empfänger. Da es mehrere Empfänger gibt, werden mehrere Kreise gebildet. Mithilfe der Schnittpunkte der Kreise kann der Standort des Empfängers ermittelt werden. Wie in der Abbildung zu sehen, sind drei Empfänger nötig, um die Position des Senders im zweidimensionalen Raum zu bestimmen. Um die Position eines Senders im dreidimensionalen Raum zu ermitteln, wären vier Empfänger notwendig. Genau das wurde bereits am Fraunhofer Institut mit Hilfe von 5G Technologie getestet (vgl. Eidloth/Kasparek/Seitz 2022).

Leider gibt es ein Problem, das das Benutzen dieser Variante in diesem Fall unmöglich macht. Innerhalb des Hochregallagers ist noch kein vollständiges Campusnetz etabliert. Somit kann auf dieses nicht zurückgegriffen werden. In der Theorie kann das entsprechende Netz einfach nachgerüstet werden. Allerdings ist das praktisch aufgrund der Struktur des Hochregallagers nicht möglich. Die Regale innerhalb des Lagers bestehen alle aus Metall. Diese große Anzahl an Metallstreben, die sich auf

engem Raum befinden, würden für Interferenzen sorgen und das Signal stören, wodurch die Drohne nicht mehr genau lokalisiert werden könnte. Zu den Metallstreben des Regals gesellen sich die Güter, die sich innerhalb des Regals befinden. Da die Firma Schaeffler Ersatzteil-Kits für Autos vertreibt, bestehen die Güter zu großen Teilen auch aus Metall, welches das Signal zusätzlich stören würde. Schließlich kommen als weitere Störquelle sich bewegende, metallische Körbe dazu, welche die Waren ein- und auslagern.

Daher muss auf die zweite Variante zurückgegriffen und die Position der Drohne mit Hilfe von Sensoren überwacht werden, die an der Drohne angebracht werden. Auch wenn diese Variante genutzt werden kann, birgt sie verschiedene Herausforderungen. Die verschiedenen Anforderungen an die Drohne führen zu einem komplexen Vorgehen, das bei jedem Schritt Fragen aufwirft. Die Drohne soll sich zwischen den Regalen frei bewegen können und zu jeder Zeit genau wissen, wo sie sich befindet. Das Regal bietet allerdings keine Orientierungspunkte für die Drohne aufgrund der Baugleichheit der Regale. Was soll demnach alles von den Sensoren überwacht werden? Reicht eine einfache Navigation der Drohne oder sollen die Sensoren auch eine Kollision vermeiden können? Wie hoch muss die Redundanz bei der Navigation und beim Kollisionsschutz sein? Im weiteren Verlauf werden diese Fragen beantwortet.

3.2 Entwicklung eines Sensorkonzepts

3.2.1 Vorbetrachtungen

Bevor ein erster Entwurf eines Sensorkonzepts aufgestellt werden kann, müssen folgende Fragen beantwortet werden: Was für Sensoren sollen genutzt werden – mit welchem Messverfahren und welche Sensorenart? Soll der Sensor nur in eine Richtung schauen oder soll er einen größeren Bereich abdecken? Wie sollen die Sensoren an die Drohne angebracht werden?

Die Vorgaben, die von den anderen Komponenten des Gesamtkonstrukts kommen, liefern hilfreiche Hinweise zur Beantwortung der Fragen. Da die Drohne so konstruiert wird, dass sie kaum Gewicht mit sich trägt, kann sowohl ein schwererer Sensor verbaut werden, der einen großen Bereich abdeckt als auch mehrere leichte, die jeweils andere Bereiche ins Visier nehmen. Die Anzahl der Sensoren wird von der mitfliegenden Recheneinheit bestimmt. Damit die Software schnell reagieren kann, ist es wichtig, dass die Gesamtbelastung nicht zu hoch ist. Bei der mitfliegenden Recheneinheit handelt es sich um einen Raspberry Pi 4B. Dieser wird verbaut, da er ein geringes Gewicht hat, eine hohe Rechenleistung für einen Einplatinenrechner aufweist und kompatibel mit den anderen Komponenten ist.

Das Anbringen der Sensoren hat nur wenig Einfluss auf die Wahl der Sensoren, da für die Drohne nur darauf geachtet werden muss, dass der Bereich unter- und oberhalb der Propeller freigehalten wird. Konstruktionen zum Anbringen der Sensoren können mittels 3D-Drucker realisiert werden.

Für die Sensorenart bzw. Wahl der Messprinzipien können im ersten Schritt mehrere Verfahren ausgeschlossen werden. Messverfahren, bei denen es zu Berührungen mit dem Regal kommt, können nicht genutzt werden, da hierbei die Wahrscheinlichkeit hoch ist, dass die Drohne abstürzt. Auch sollte auf bewegliche Messgeräte weitestgehend verzichtet werden, da diese ebenfalls negativ auf das Flugverhalten der Drohne einwirken können. Nach längerer Recherche setzen sich drei Messprinzipien durch: die Messung mit FMCW-Radar, einer Tiefenkamera und LIDAR. Da die genauen Funktionsweisen der einzelnen Messverfahren bereits im Theorieteil erklärt wurden, wird ab Punkt 2.2.3 die praktische Umsetzung in den Blick genommen.

3.2.2 Steuerung

In Folgenden werden zwei Möglichkeiten vorgestellt, von wo aus die Steuerbefehle erzeugt werden können:
Die erste Möglichkeit ist, dass auf der Drohne selbst gerechnet wird. Das heißt, dass alle Daten von den Sensoren auf der mitgeführten Recheneinheit verarbeitet und daraus Steuerbefehle generiert werden. In dieser Lösung wird die 5G-Verbindung genutzt, um einerseits oberflächliche Befehle zu geben, wie z.B. „Fliege Route X ab“. Andererseits wird 5G mit in dieser Lösung dafür verwendet, um aufgenommene Daten an die Außenwelt abzugeben. Damit sind die aufgenommenen Bilder der Kamera gemeint, die verwendet werden sollen, um zu überprüfen, was sich in den einzelnen Regalen befindet.
Die bestehende 5G-Verbindung ermöglicht aber auch eine zweite Möglichkeit der Rechenarchitektur. Bei dieser werden alle Daten, die von den Sensoren aufgenommen werden, über 5G an einen externen Rechner verschickt. Dieser übernimmt die Umwandlung der Sensordaten in Steuerbefehle. Diese Befehle werden dann über 5G zurück an die Recheneinheit der Drohne gesendet, welche diese dann nur noch an ihren FC weiterleitet. Diese Art der Umsetzung wird durch eine geringe Latenzzeit von 5G ermöglicht. Ein Punkt, der für diese zweite Lösung spricht, ist, dass die Recheneinheit der Drohne hier kleiner dimensioniert werden kann, da sie weniger Rechenleistung aufbringen muss. In der Praxis kann diese Lösung allerdings nicht umgesetzt werden. Für das externe Rechnen wird zwingend eine stabile Funkverbindung benötigt, da die Drohne aufgrund von fehlenden Signalen abstürzen könnte. Die Architektur des Hochregallagers sorgt jedoch dafür, dass nicht sichergestellt werden kann, dass zu jeder Zeit eine stabile 5G-Verbindung vorliegt.

Eine Lösung, die beide Konzepte miteinander vereint, indem die Drohne bei einer bestehenden Verbindung von außen und bei dem Ausfallen der Verbindung intern geleitet wird, bietet ebenfalls keine Vorteile. Die mitfliegende Hardware müsste die gleiche Rechenleistung aufweisen wie der externe Rechner, um allein agieren zu können. Zusätzlich ist der Übergang von einem auf das andere System eine potenzielle Fehlerquelle. Daher wird auch von dieser kombinierten Lösung Abstand genommen. Aus den genannten Gründen wird die erste Lösung umgesetzt. Die Steuerbefehle werden von der mitfliegenden Recheneinheit auf der Drohne erzeugt.

3.2.3 FMCW-Radar

Für die Messung mit FMCW wird der Sensor „OPS241-B Short Range FMCW Radar Sensor“ der Firma OmniPreSense gewählt. Dieser wird in der folgenden Abbildung dargestellt:

Abbildung 9: FMCW-Radar „OPS241-B Short Range FMCW Radar Sensor“ (übernommen von: OmniPreSense o.A.)

Der Sensor hat eine Erfassungsreichweite von einem bis 20 Metern. Für das Hochregallager ist die Reichweite ausreichend, da die Lücken zwischen den Regalen sehr klein sind. Der Erfassungswinkel beträgt 78°, wodurch die Wahrscheinlichkeit hoch ist, dass alle Regalstreben erfasst werden. Ebenfalls von Vorteil ist, dass der Sensor sowohl über USB als auch über UART ansprechbar ist. Demzufolge kann er gut mit dem Raspberry kombiniert werden. Größe und Gewicht passen ebenfalls gut in das Gesamtkonzept. Mit elf Gramm und den Maßen 53 x 59 x 12 mm ist der Sensor nicht nur sehr leicht, sondern auch sehr handlich. Das Ziel des Sensors ist es, die Drohne davor zu bewahren, zu nah an ein Regal heranzufliegen. Um die Funktionalität des Sensors zu testen, müssen seine Daten ausgelesen werden. In der nachfolgenden Abbildung 10 wird ein Programm betrachtet, das mit Raspberry Pi geschrieben wurde und die Daten des Sensors ausliest. Vorbereitend muss der Sensor entweder über USB oder UART an den Raspberry angeschlossen werden.

```python
import serial
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(37,GPIO.OUT)
ser=serial.Serial("/dev/ttyACM0", baudrate=19200)

ser.write(b'uM' + b'F2' + b'04')

while True:
    x=ser.readline().decode('utf-8').strip('r\n')
    print(x)
    if x < str(1):
        print('WARNING')
        GPIO.output(37,1)
    else:
        GPIO.output(37,0)
```

Abbildung 10: Programm zum Auslesen der Daten des FMCW-Radars

Das Programm liest den Messwert des Sensors aus und schaltet eine LED ein, sobald der gemessene Wert einen zuvor festgelegten Schwellenwert unterschreitet. Um den Sensor für die Zusammenarbeit mit der Drohne einzustellen, werden im ersten Schritt die benötigten Bibliotheken importiert. Das sind hier: „Serial“, die für die serielle Kommunikation zwischen Sensor und Raspberry zuständig ist und „RPi.GPIO“, die für die Steuerung der GPIO-Pins benötigt wird. Im folgenden Schritt werden die GPIO-Pins konfiguriert, bevor im Anschluss die serielle Kommunikation vorbereitet wird. Für die Kommunikation sind zwei Parameter entscheidend. Einerseits ist das „‘/dev/tty/ACM0‘“, was dem Programm signalisiert, an welcher Stelle der Senor angeschlossen ist. In diesem Fall ist der Sensor über USB mit dem Raspberry verbunden. Der zweite Parameter ist die Baudrate, welche vom Gerät verwendet wird. Aus dem Datenblatt lässt sich entnehmen, dass diese standardmäßig 19200 beträgt. Die Baudrate des Sensors ist veränderbar und kann zu anderen Standardwerten geändert werden. Der Befehl „ser.write“ wird dazu genutzt, Einstellungen am Sensor vorzunehmen. Mit „b'uM‘“ wird die Einheit des Sensors auf Meter eingestellt, während „b’F2‘“ dem Sensor kommuniziert, dass er zwei Nachkommastellen ausgeben soll. In der While-Schleife wird der Messwert des Sensors kontinuierlich empfangen und mit einem vordefinierten Wert verglichen. Unterschreitet der Sensor diesen, wird der Nutzer mit der LED-Leuchte gewarnt.

3.2.4 Tiefenkamera

Für die Tiefenkamera wird das Modell „Intel® RealSense™ Depth Camera D435i“ genutzt, das in den Abbildungen 11 und 12 zu sehen ist:

Abbildung 11: Tiefenkamera „Intel® RealSense™ Depth Camera D435i“ von vorne (übernommen von: Intel Corporation o.A.)

Abbildung 12: Tiefenkamera „Intel® RealSense™ Depth Camera D435i“ von hinten (übernommen von: BOTLAND B. DERKACZ SP. K. o.A.)

Die Kamera hat eine ideale Reichweite von 0,3 bis ca. 3m, die tatsächliche Reichweite liegt bei über 10m. Die Kamera wird mittels USB3.0 verbunden und kann zwei unterschiedliche Arten von Bildern aufnehmen: zum einen ein „normales“ RGB-Bild und zum anderen ein Depth-Frame, auch Tiefenbild genannt. In einem Tiefenbild werden die Entfernungen der einzelnen Objekte innerhalb des Bildes dargestellt. Verdeutlicht wird dies meistens mit Farben in Form einer „Heatmap“. Objekte, die sich näher an der Kamera befinden, werden meist mit kalten Farben dargestellt, während Gegenstände, die weiter entfernt sind, oftmals in wärmere Farben getaucht werden. Im der folgenden Abbildung ist ein solches Tiefenbild zu sehen:

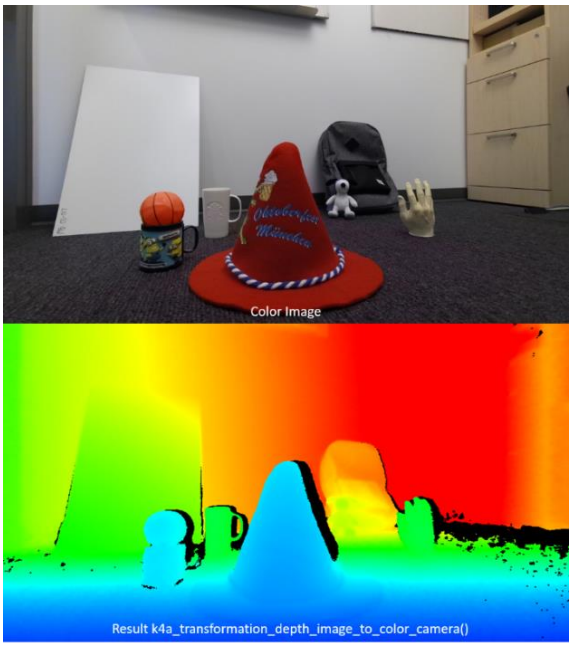


Abbildung 13: Tiefenbild (übernommen von: Microsoft Corporation o.A.)

Die Auflösung des Tiefenbildes beträgt 1280x720, wenn die Kamera über USB3.0 verbunden ist. Bei einer Verbindung über USB2.0 sinkt die Auflösung. Die Kamera kann bis zu 90 Tiefenbilder pro Sekunde anfertigen. Unter idealen Umständen kann das RGB-Bild eine Auflösung von 1920x1080 erreichen. Dadurch dass die Kamera sowohl Tiefen- als auch RGB-Bilder schießt, kann auf eine weitere Kamera, die den Inhalt der Regale sichtet, verzichtet werden.

In der folgenden Darstellung ist ein Programm zu sehen, das die Daten der Kamera ausliest.

```
import pyrealsense2 as rs
import numpy as np

# Configure depth and color streams
pipeline_1 = rs.pipeline()
config_1 = rs.config()
config_1.enable_device('233722070206')
config_1.enable_stream(rs.stream.depth , 640 , 480 , rs.format.z16 , 30)
config_1.enable_stream(rs.stream.color , 640 , 480 , rs.format.bgr8 , 30)

# Start streaming
pipeline_1.start(config_1)


while True:

    frames_1 = pipeline_1.wait_for_frames()
    depth_frame_1 = frames_1.get_depth_frame()
    depth_image_1 = np.asanyarray(depth_frame_1.get_data())
    print("Image1=" , depth_image_1)
```

Abbildung 14: Programm zum Auslesen der Daten der Tiefenkamera

Das Programm gibt die Tiefenwerte als Array aus. Wie beim Radar werden auch hier zunächst die Bibliotheken importiert. Für dieses Programm werden zwei benötigt – „pyrealsense2“ und „NumPy“. „pyrealsense2“ ermöglicht das Arbeiten mit der D435i in Python. „NumPy“ wird für das Arbeiten mit Arrays innerhalb von Python genutzt. Im zweiten Schritt wird der Daten- und Farb-Stream konfiguriert. Dazu wird als Erstes ein Kontextobjekt erstellt, das zum Interagieren mit dem angeschlossenen Realsense-Gerät benötigt wird. In Zeile 7 wird das genaue Gerät eingetragen. Das ist in diesem Fall die Seriennummer der Kamera, die auf der Unterseite des Geräts zu finden ist. Danach kann die Auflösung eingestellt werden, mit welcher die Daten ausgegeben werden. Im hier gezeigten Beispiel beträgt die Auflösung 640*480 Pixel (siehe Zeile 8-9). Schließlich werden in Zeile 12 die Einstellungen an Pipeline übergeben und der Stream wird gestartet. In der danach folgenden While-Schleife werden die Frames entgegengenommen und als Array ausgegeben. Hier wird nur das Array der Tiefenwerte betrachtet, da das Array der Farbwerte nicht zur Steuerung beiträgt. Für das Bild, das die RGB-Kamera aufnimmt, ist vorgesehen, dass es nicht weiterverarbeitet, sondern nur vom Personal gesichtet wird.

Das Programm zum Auslesen der Tiefenwerte kann nicht direkt auf den Raspberry Pi übertragen werden, da die „pyrealsenese2“-Bibliothek nicht für Linux-Systeme ausgelegt ist und dort dementsprechend nicht funktioniert. Für dieses Problem gibt es zwei Lösungen. In der ersten Lösung wird pyrealsense2 von Grund auf mit cmake auf dem Raspberry gebaut. In der Praxis birgt diese Lösung jedoch viele Fehlerquellen. Daher wird eine zweite Lösung vorgezogen. Diese sieht vor, mittels Docker einen Container auf dem Raspberry laufen zu lassen. Das Vorgehen wird in 3.3.4.2. erklärt.

3.2.5 LIDAR

Der letzte Sensor, der an dieser Stelle betrachtet wird, ist LIDAR „DTOF LIDAR LD19“ der Firma Waveshare. Dieser wird in der folgenden Abbildung dargestellt:

Abbildung 15: DTOF LIDAR LD19 (übernommen von: Waveshare o.A.)

Dieser LIDAR kann Entfernungen in einem Winkel von 360° erfassen. Dazu dreht sich der Licht-Emitter im oberen runden Teil des LIDARs. Er ist 38.59*38.59*33.50mm groß und hat eine Reichweite von bis zu 12m. Die Verbindung wird über UART hergestellt. Dabei kann er entweder direkt an die Pins befestigt oder mit Hilfe eines mitgelieferten Adapters an einen USB-Port angeschlossen werden. Um die Drohne nicht zusätzlich durch das Gewicht eines Adapters zu belasten, wird der Sensor direkt an die Pins angeschlossen.

Für das Programm zum Auslesen und Ausgeben der Daten von LIDAR wird zuerst recherchiert, ob bereits eine fertige Bibliothek existiert, die dazu genutzt werden kann. Ein Blick in das Wiki des Sensors zeigt, dass es einen fertigen Quellcode für den LIDAR gibt. Allerdings ist dieser für ROS geschrieben. Daher wird ein eigenes Programm geschrieben, das den rohen Datenstrom empfängt und diesen zu verwertbaren Daten konvertiert. Dazu wird der Datenstrom aus dem Wiki des Sensors im ersten Schritt analysiert, welcher in der folgenden Abbildung zu sehen ist:

Header	VerLen	Speed		Start angle		Data	End angle		Timestamp		CRC check
54H	2CH	LSB	MSB	LSB	MSB	……	LSB	MSB	LSB	MSB	1Byte

Measuring point 1			Measuring point 2			…	Measuring point n		
distance		intensity	distance		intensity		distance		intensity
LSB	MSB	1 Byte	LSB	MSB	1 Byte	…	LSB	MSB	1 Byte

Abbildung 16: Roher Datenstrom des LIDARs „DTOF LIDAR LD19“ (übernommen von: Shenzhen youyeetoo Tech o.A.)

In Bild 16 ist ein gesamter Datenframe abgebildet. An erster Stelle steht der Header, dessen Aufgabe es ist, den Beginn des Datenpakets zu kennzeichnen. Er hat einen konstanten Wert von 54H. An zweiter Stelle folgt VerLen. Die ersten drei Bits geben die Art des Pakets an, während die letzten fünf Bits die Anzahl der Messpunkte zeigen. Auch diese beiden Werte sind konstant. Der Pakettyp bleibt 1 und es werden 12 Messpunkte gezählt. Daher bleibt der Wert des zweiten Bytes bei konstanten 2CH. Die folgenden zwei Bytes geben die Geschwindigkeit in Grad pro Sekunde an, mit der sich der LIDAR Sensor dreht. An dieser Stelle ist zu beachten, dass zuerst das LSB (Least Significant Bit) und danach das MSB (Most Significant Bit) gesendet wird. Demzufolge müssen die beiden Bytes bei der Übersetzung vertauscht werden. Ist LSB beispielsweise 68H und MSB 08H, muss 0868H und nicht 6808H in einen dezimalen Wert umgewandelt werden. Die Geschwindigkeit beträgt in diesem Beispiel nicht 26632 Grad pro Sekunde, sondern 2152 Grad pro Sekunde. Die nächsten zwei Bytes geben den Startwinkel der Messung an. Da dieser LIDAR die Entfernung in einem Winkel von 360° misst, wird zu jedem Messpunkt ein Winkel angegeben. Dieser lässt sich aus dem Start- und Endwinkel aus dem Datenpaket errechnen.

Auf der Oberfläche des LIDARs ist ein Pfeil eingraviert, der anzeigt, in welcher Richtung sich 0° befinden:

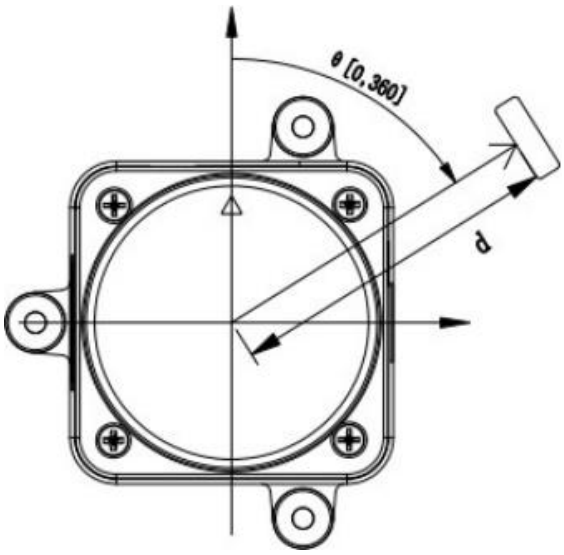
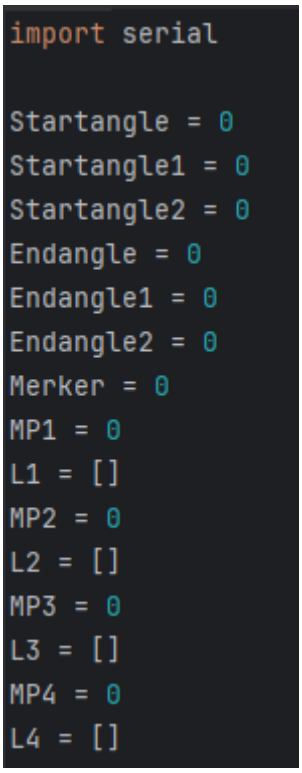


Abbildung 17: Schema des LIDARs von oben (übernommen von: Shenzhen youyeetoo Tech o.A.)

Die zwölf Messpunkte bestehen aus jeweils drei Bytes. Die ersten beiden Bytes geben die Entfernung in mm an, während das dritte Byte die Intensität der Reflektion enthält. Zunächst werden die Messpunkte gesendet, dann die Endwinkel. Aus diesem kann dann, mit Hilfe des Startwinkels, der Winkel der einzelnen Messpunkte ermittelt werden. Daran anknüpfend wird ein Timestamp gesendet. Dieser umfasst zwei Bytes und ermöglicht es, zu ermitteln, wie viel Zeit zwischen zwei Datenpaketen vergeht. Schließlich wird ein letztes Byte als CRC-Check Summe geschickt. Damit können alle bereits gesendeten Daten innerhalb des Datenpaketes verifiziert werden. Verschiedene Tests zeigen allerdings, dass dieser LIDAR so zuverlässig ist, dass keine Überprüfung der Daten mittels CRC-Check Summe notwendig ist. Daher wird an dieser Stelle nicht weiter auf den CRC-Check eingegangen. Im Folgenden wird das Programm vorgestellt, welches genutzt wird, um diesen LIDAR zu betreiben.
Im ersten Schritt wird „pyserial“ ins Programm importiert:



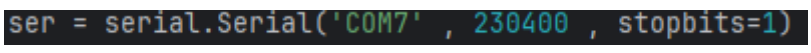
```
import serial

Startangle = 0
Startangle1 = 0
Startangle2 = 0
Endangle = 0
Endangle1 = 0
Endangle2 = 0
Merker = 0
MP1 = 0
L1 = []
MP2 = 0
L2 = []
MP3 = 0
L3 = []
MP4 = 0
L4 = []
```

Abbildung 18: LIDAR-Programm: Importieren der Bibliotheken und Definieren von Variablen

„pyserial“ ermöglicht die Kommunikation mit dem Sensor. Die Abbildung zeigt den Vorgang des Importierens der Library. Zu sehen ist, wie die benötigten Variablen für die ersten vier Messpunkte bestimmt werden. Als Erstes werden drei Variablen für den Startwinkel definiert. Die beiden nummerierten Variablen werden im nächsten Schritt zu einer verbunden, welche nicht nummeriert ist. Mit der gleichen Vorgehensweise werden Variablen für den Endwinkel bestimmt. Daneben werden ein Merker, der für spätere Berechnungen relevant ist, und jeweils zwei Variablen für jeden Datenpunkt definiert. Eine Variable speichert die Entfernung, die andere Variable ist ein Array, in dem Entfernung und Winkel hintereinander stehen. Bei allen weiteren Messpunkten wird auf die gleiche Weise vorgegangen, bevor zum Schluss die Variable L mit „L = []“ definiert wird. In dieser Variablen, die ebenfalls ein Array ist, werden alle Messpunkte abschließend zusammengefasst.

Im nächsten Schritt wird die serielle Verbindung wie folgt vorbereitet:



```
ser = serial.Serial('COM7' , 230400 , stopbits=1)
```

Abbildung 19: LIDAR Programm: serielle Verbindung

Besonders wichtig sind hier die Daten, welche in der Klammer eingetragen sind. Die erste Angabe ist der Port, an welchem das Gerät angebracht ist. Danach folgt die Baudrate und als Letztes kommt die Anzahl der Stopbits. Aus dem Wiki des Sensors

lässt sich ablesen, welche Werte konkret eingetragen werden müssen. Die Baudrate muss 230400 betragen und es muss ein Stopbit eingestellt werden.
Im dritten Schritt wird eine While-Schleife erstellt. Jeder unter ihr stehende Quellcode ist Teil dieser Schleife und wird in dieser endlos wiederholt. In der folgenden Abbildung ist der erste Teil der Schleife dargestellt:

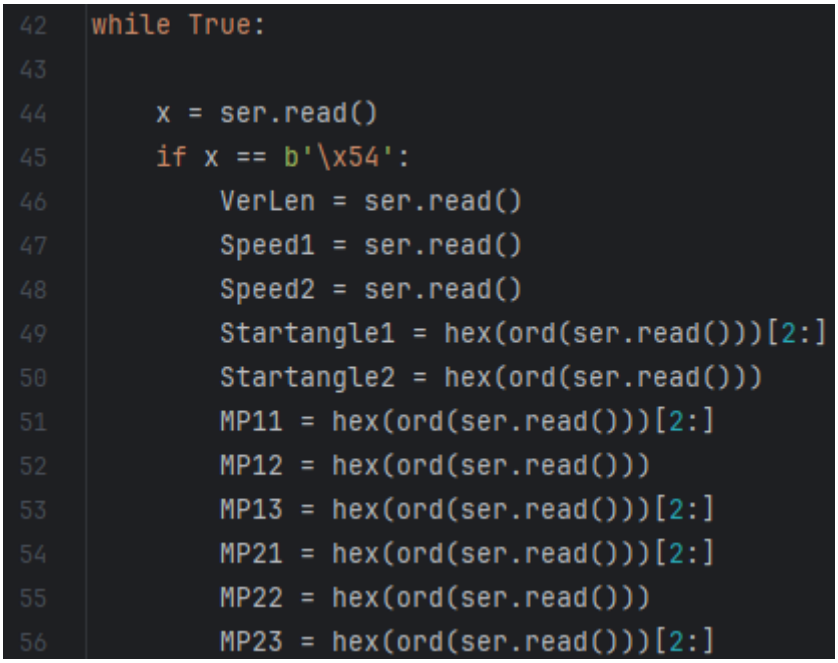


Abbildung 20: LIDAR-Programm: Beginn der While-Schleife und Einlesen der Daten

Zu Beginn liest das Programm die Daten so lange ein, bis es den Header erkennt. Danach beginnt das Programm mit dem Einlesen des gesamten Datenframes. Zuerst werden VerLen und die Geschwindigkeit aufgenommen, dann der Startwinkel. Wie in der Abbildung zu erkennen, werden einige Operationen an den Daten beim Einlesen vorgenommen, damit die zwei eingelesenen Bytes in der Abbildung im nächsten Schritt zu einem Wert vereinigt werden können. Die Daten werden in Hex-Werte umgewandelt und das „\x“ beim LSB entfernt, da es beim Zusammenführen der Daten stören würde. Dieses „\x“ am Anfang eines Wertes wird von pyserial automatisch hinzugefügt.
Im nächsten Schritt werden die Datenpunkte eingelesen. Dabei werden für jeden Punkt drei Bytes aufgenommen, die, wie in der Abbildung erkennbar, nummeriert werden. Die Daten bei den Datenpunkten werden umgewandelt, damit sie weiterverarbeitet werden können.
In der folgenden Abbildung werden die restlichen Daten analog zu den anderen eingelesen. Aufgrund der Redundanz wird das Einlesen der restlichen Messpunkte nicht bebildert.

```
MP121 = hex(ord(ser.read()))[2:]
MP122 = hex(ord(ser.read()))
MP123 = hex(ord(ser.read()))[2:]
Endangle1 = hex(ord(ser.read()))[2:]
Endangle2 = hex(ord(ser.read()))
Time1 = hex(ord(ser.read()))[2:]
Time2 = hex(ord(ser.read()))
CRCcheck = hex(ord(ser.read()))[2:]
```

Abbildung 21: LIDAR-Programm: Ende des Einlesens der Daten

Im nächsten Schritt werden die eingelesenen Bytes in Integer-Werte umgewandelt. Die nachfolgende Abbildung 22 zeigt diesen Vorgang. Zur Umwandlung werden aus den Hex-Werten zuerst Strings gemacht, damit sie, wenn sie addiert werden, hintereinander stehen. Das kann beispielsweise wie folgt aussehen: „str(\x39)+ str(28) wird str(\x3928)“. Im Anschluss daran werden diese Strings dann zu Integer umgewandelt. Dabei wird die Basis 0 verwendet, um Dezimalzahlen zu bilden. Eine Besonderheit stellen hier die Winkel da. Diese werden durch 100 dividiert, da die Winkel ohne Komma, aber mit zwei Nachkommastellen übertragen werden. Als Letztes wird der Wert des Endwinkels in einem Merker gespeichert.

```
Speed = int(str(ord(Speed2)) + str(ord(Speed1)))
Startangle = (int((str(Startangle2) + str(Startangle1)) , 0)) / 100
MP1 = int((str(MP12) + str(MP11)) , 0)
MP2 = int((str(MP22) + str(MP21)) , 0)
MP3 = int((str(MP32) + str(MP31)) , 0)
MP4 = int((str(MP42) + str(MP41)) , 0)
MP5 = int((str(MP52) + str(MP51)) , 0)
MP6 = int((str(MP62) + str(MP61)) , 0)
MP7 = int((str(MP72) + str(MP71)) , 0)
MP8 = int((str(MP82) + str(MP81)) , 0)
MP9 = int((str(MP92) + str(MP91)) , 0)
MP10 = int((str(MP102) + str(MP101)) , 0)
MP11 = int((str(MP112) + str(MP111)) , 0)
MP12 = int((str(MP122) + str(MP121)) , 0)
Endangle = int((str(Endangle2) + str(Endangle1)) , 0) / 100
Merker = Endangle
```

Abbildung 22: LIDAR-Programm: Umwandlung der Daten von Hex in Integer

Als Nächstes wird ein Array für jeden Datenpunkt erstellt, in dem der Abstand und der Winkel gespeichert werden. Dafür muss der Winkel der einzelnen Datenpunkte

errechnet werden. Wie das geschieht, wird in den nächsten beiden Darstellungen gezeigt:

```
if Endangle < Startangle:
    Endangle = Endangle + 360
Angle = Endangle - Startangle
Angle = Angle / 10
L1.append(MP1)
L1.append(Startangle)
if (Startangle + Angle) > 360:
    Startangle = Startangle - 360
L2.append(MP2)
L2.append(round((Startangle + Angle),2))
if (Startangle + Angle * 2) > 360:
    Startangle = Startangle - 360
L3.append(MP3)
L3.append(round((Startangle + (2 * Angle)),2))
if (Startangle + Angle * 3) > 360:
    Startangle = Startangle - 360
```

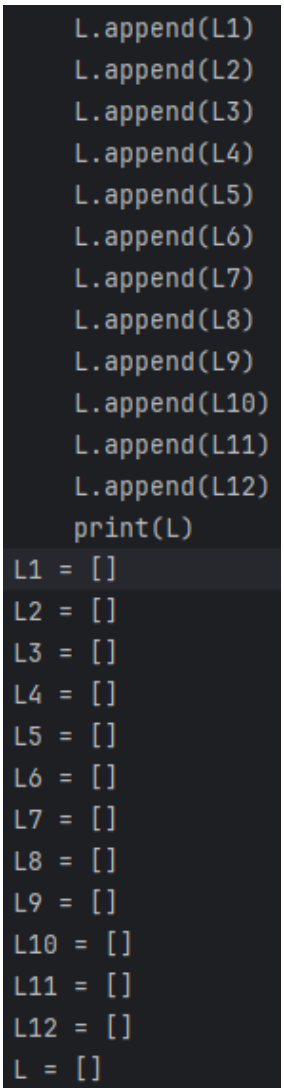
Abbildung 23: LIDAR-Programm: Beginn der Erstellung von Arrays für die Datenpunkte 1-3

```
L7.append(MP7)
L7.append(round((Startangle + (6 * Angle)),2))
if (Startangle + Angle * 7) > 360:
    Startangle = Startangle - 360
L8.append(MP8)
L8.append(round((Startangle + (7 * Angle)),2))
if (Startangle + Angle * 8) > 360:
    Startangle = Startangle - 360
L9.append(MP9)
L9.append(round((Startangle + (8 * Angle)),2))
if (Startangle + Angle * 9) > 360:
    Startangle = Startangle - 360
L10.append(MP10)
L10.append(round((Startangle + (9 * Angle)),2))
if (Startangle + (10 * Angle)) > 360:
    Startangle = Startangle - 360
L11.append(MP11)
L11.append(round((Startangle + (10 * Angle)),2))
L12.append(MP12)
L12.append(Merker)
```

Abbildung 24: LIDAR-Programm: Beendigung der Erstellung von Arrays für die Datenpunkte 7-12

Zur Berechnung wird im ersten Schritt die Differenz zwischen Start- und Endwinkel bestimmt. Sollte der Endwinkel die 360° überschreiten, ist es möglich, dass sein Wert kleiner als der vom Startwinkel ausfällt. Überschreitet ein Wert die 360°, muss er um 360° reduziert werden. Der entstandene Abstand wird durch zehn geteilt und auf die zwölf Datenpunkte verteilt. Er wird durch zehn und nicht durch zwölf geteilt, da der erste und der letzte Punkt den Wert vom Start- und dem Endwinkel haben. Für jeden Datenpunkt wird der erhaltene Integer-Wert an das Array angehangen, bevor im Anschluss der Winkel berechnet und angehangen wird. Der Winkel für die einzelnen Datenpunkte errechnet sich aus dem Startwinkel plus dem entsprechenden Vielfachen des durch zehn geteilten Abstandes. Der Winkel wird auf zwei Nachkommastellen gerundet, bevor er final angehangen wird. Auch in Abbildung 23 und 24 wurden Datenpunkte aus Gründen der Redundanz ausgelassen.

Am Ende des Programms werden drei letzte Operationen vorgenommen. Zunächst werden alle Datenpunkte in einem großen Array L zusammengefasst. Danach wird dieses entstandene Array ausgegeben. Als Letztes werden alle verwendeten Arrays gelöscht, um im nächsten Durchgang wieder verwendet werden zu können. Der Vorgang ist im Folgenden abgebildet:



```
    L.append(L1)
    L.append(L2)
    L.append(L3)
    L.append(L4)
    L.append(L5)
    L.append(L6)
    L.append(L7)
    L.append(L8)
    L.append(L9)
    L.append(L10)
    L.append(L11)
    L.append(L12)
    print(L)
L1 = []
L2 = []
L3 = []
L4 = []
L5 = []
L6 = []
L7 = []
L8 = []
L9 = []
L10 = []
L11 = []
L12 = []
L = []
```

Abbildung 25: LIDAR-Programm: Ausgabe der Daten

3.2.6 Entwurf eines ersten Sensorkonzepts

Im Folgenden wird die erste Version eines Sensorkonzepts dargestellt. Dazu wird zuerst auf die einzelnen Komponenten eingegangen, bevor erklärt wird, wo und wie diese an die Drohne angebracht werden und welche Funktion sie im Konzept erfüllen. Schließlich werden Tests mit den Sensoren durchgeführt und Stärken und Schwächen des Aufbaus diskutiert.

3.2.6.1 Komponenten

Der Startpunkt ist hier die fertig gebaute Drohne. Ihre einzelnen Teile werden daher hier nicht erneut thematisiert. An diese fertige Drohne werden zwei Sensorarten verbaut: eine Tiefenkamera und zwei Radarsensoren. In diesem Aufbau kommt LIDAR noch nicht vor, da er zu diesem Zeitpunkt in vielen Aspekten als ungünstiges Bauteil erschien. Auch wird an dieser Stelle kein 5G-Modem verbaut, da die Entwicklung zu diesem Zeitpunkt noch nicht so weit ist. Allerdings wird hier der im vorangegangenen Punkt angesprochene Raspberry Pi 4B montiert, welcher die Daten der Sensoren aufnimmt und verarbeitet. Daneben fliegt eine Powerbank mit einem Output von 5V und 3A mit, die den Raspberry und die Sensoren mit Strom versorgt. Im folgenden Bild wird der geplante erste Aufbau dargestellt:

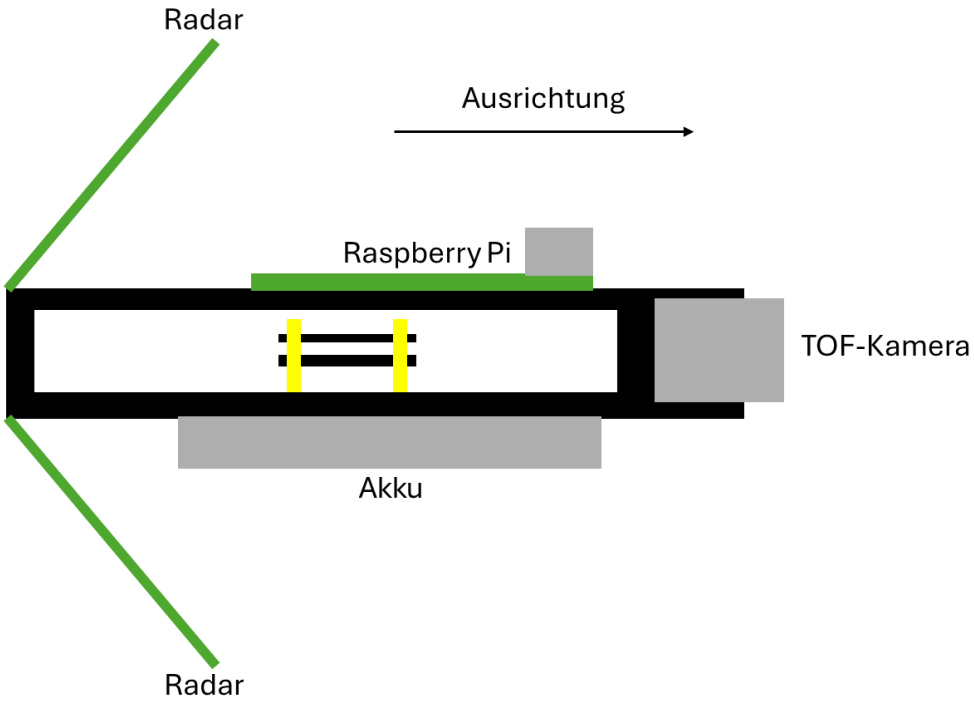


Abbildung 26: Entwurf eines ersten Sensorkonzepts

In dieser Darstellung ist die Drohne von der Seite dargestellt. Die Beine und Motoren werden zum besseren Verständnis hier weggelassen. Zu erkennen ist, dass die Tiefenkamera vorne teilweise in den Rahmen der Drohne integriert wird. Die zwei Radarsensoren sind hinten an der Drohne angebracht. Dabei ist jeweils einer unten und oben angebracht. Der Raspberry ist oben in der Mitte der Drohne befestigt. Der Akku ist ebenfalls mittig unterhalb der Drohne angebracht.
Die Tiefenkamera bildet in diesem Aufbau das Herzstück. Sie ist dafür verantwortlich, die Entfernung zum Regal zu bestimmen und die Orientierung am Regal zu bewahren.

Zusätzlich hat die Kamera die Aufgabe, Aufnahmen vom Inhalt des Regals zu machen. Diese sollen später von Mitarbeitern ausgewertet werden können, um den genauen Inhalt des Regals zu bestimmen. Die zwei Radarsensoren haben die Aufgabe, die Entfernung zum Regal zu regulieren. Auch wird der hohe Erfassungswinkel der Kameras zu Nutzen gemacht, um potenzielle Hindernisse von oben oder von unten gleichzeitig zu erkennen. Der Raspberry Pi wird in der Mitte angebracht, um die Verkabelung der einzelnen Komponenten möglichst einfach zu gestalten. An dieser Stelle ist er auch besser vor einem Absturz geschützt als auf der Unterseite der Drohne. Da die mitzuführende Powerbank deutlich robuster ist als der Raspberry Pi, wird sie an der Unterseite der Drohne angebracht. Auf dieser Seite wird auch der Akku befestigt, da er aufgrund seines ähnlichen Aufbaus gut mit der Powerbank zusammen befestigt werden kann.

3.2.6.2 Befestigung der Komponenten

Zur Befestigung der Komponenten werden Teile mit der Onlineplattform Tinkercad konstruiert, die anschließend mit dem 3D-Drucker „Ultimaker 3 Extended“ hergestellt werden. Als Filament wird das ABS Filament „TitanX“ genutzt. ABS hat den Vorteil, dass es sehr stabil ist. Das wiederrum führt dazu, dass die einzelnen Komponenten mit geringem Infill gedruckt werden können, um das Gewicht zu reduzieren. Infill gibt den Prozentsatz der Füllung beim 3D-Druck an.
Zur Befestigung der Tiefenkamera wird folgendes Bauteil gedruckt:

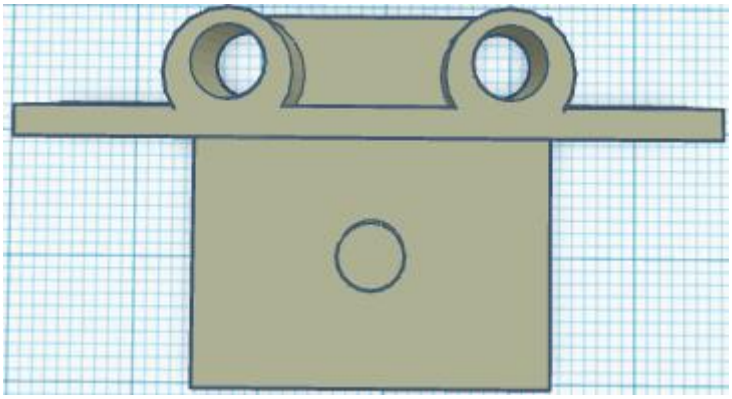


Abbildung 27: Halterung der Tiefenkamera von oben

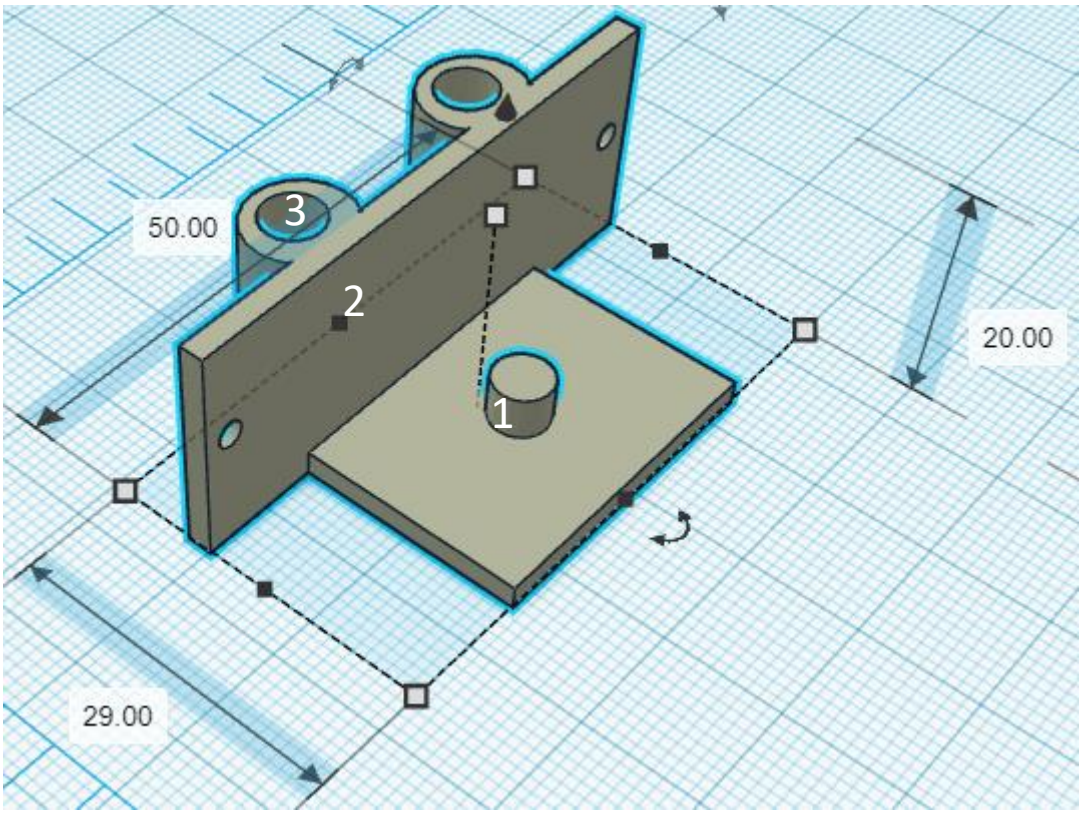


Abbildung 28: Halterung der Tiefenkamera von der Seite

In Abbildung 28 sind die einzelnen Komponenten des Bauteils nummeriert und dessen Maße in mm angegeben. Die erste Komponente des Bauteils ist die Bodenplatte. Auf diese wird ein Zylinder angebracht, der so groß ist, dass er in die Stativ-Öffnung der Kamera passt und diese an der Stelle fixiert. Auch die zweite Komponente des Bauteils ist dazu da, die Kamera zu befestigen. Das Hauptaugenmerk liegt hier auf den zwei Löchern an den zwei Enden. Dies werden verwendet, um die Kamera mit dem Bauteil zu verschrauben. Das ist möglich, da die Kamera, wie in Abbildung 12 zu sehen, an der Rückseite zwei kleine Löcher zum Verschrauben hat. Im letzten Abschnitt des Bauteils wird das Bauteil mit der Drohne verbunden. Dazu werden die Metallstangen verwendet, die den Rahmen der Drohne zusammenhalten. Die zwei vordersten Stangen werden entfernt, damit die Kamera Platz innerhalb des Rahmens hat. Die beiden Löcher sind genau so ausgemessen, dass die Stangen hindurch passen. Mit der Wiederanbringung des oberen Teil des Rahmens kann das Bauteil sich nicht mehr bewegen. Dadurch wird die Kamera innerhalb des Rahmens fixiert und dadurch zusätzlich geschützt.

Das nächste Bauteil hat zum Ziel, den Raspberry Pi und einen der beiden Radarsensoren zu fixieren:

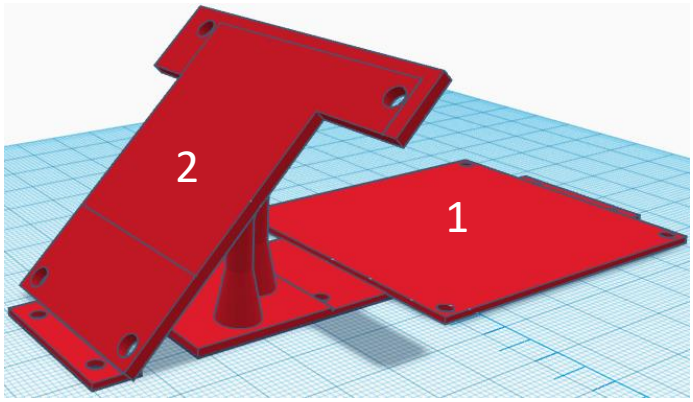


Abbildung 29: Halterung für den Raspberry Pi und die Radarsensoren von der Seite vorne

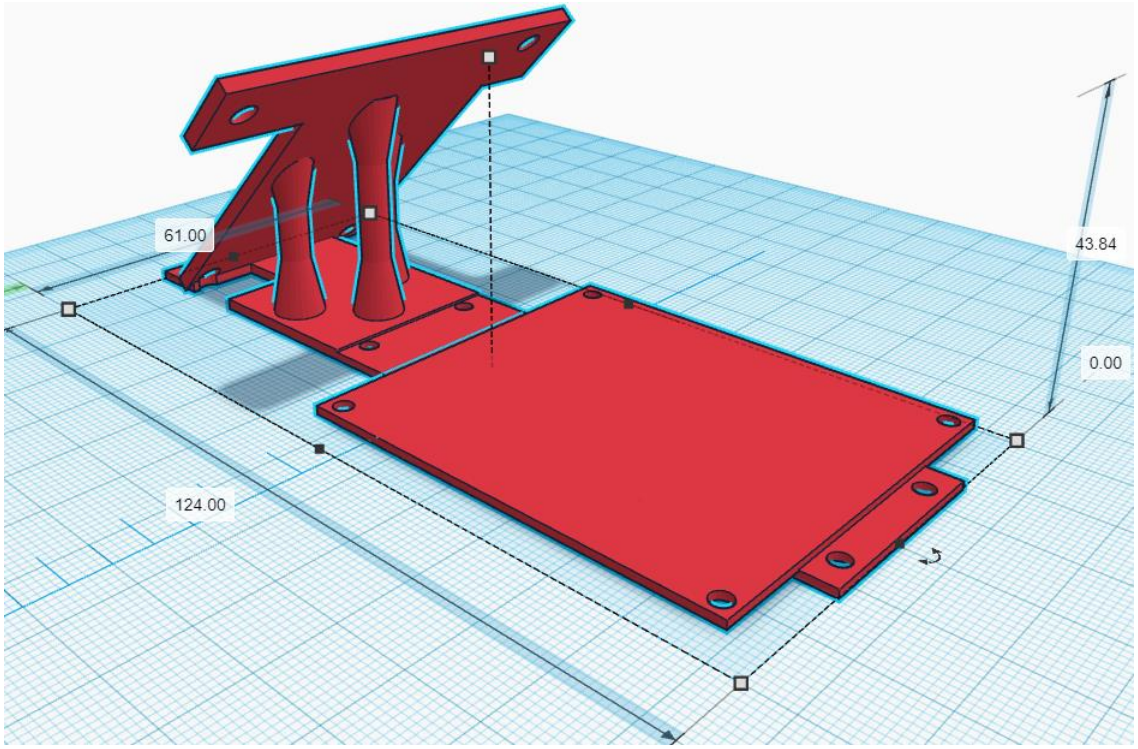


Abbildung 30: Halterung für den Raspberry Pi und die Radarsensoren von der Seite hinten

In Abbildung 30 sind die Maße des Bauteils zu finden, während in Abbildung 29 die einzelnen Abschnitte des Konstrukts nummeriert sind. Die große Fläche in Abschnitt 1 dient der Befestigung des Raspberry Pis. Die Löcher in der Fläche entsprechen in Größe und Abstand den Löchern des Raspberry Pis. Daher muss dieser an diesen Stellen nur verschraubt werden. An der angewinkelten Platte in Abschnitt 2 wird ein Radarsensor angebracht. Wie auch der Raspberry Pi besitzt er bereits Löcher, die zum Befestigen vorgesehen sind. Die drei Säulen an der Platte dienen zur Sicherung der angewinkelten Fläche, damit diese nicht zusammenbricht. Das gesamte Bauteil wird oben auf den Rahmen geschraubt. Dazu sind weitere Löcher vorhanden, die den Maßen der im Rahmen vorgefertigten Löchern entsprechen.

Schließlich werden vier Füße entworfen, die dazu beitragen sollen, dass die Drohne besser starten kann und beim Landen nicht auf den Akku oder die Powerbank fällt. In der folgenden Abbildung wir einer der Füße dargestellt:

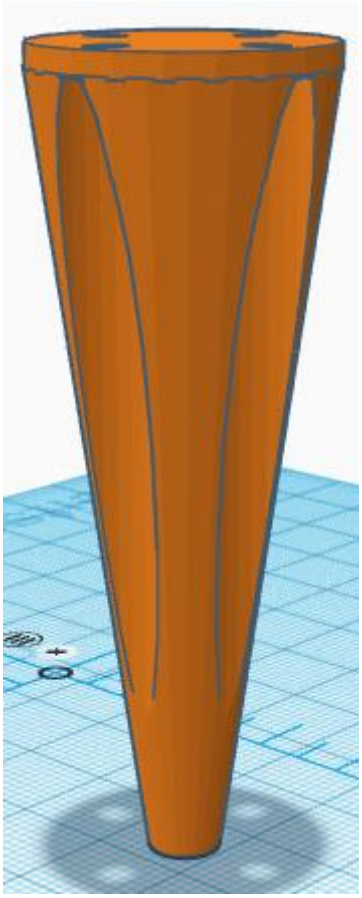


Abbildung 31: Drohnen-Fuß

Der Fuß ist so konzipiert, dass die Schrauben, die genutzt werden, um die Motoren festzuschrauben, auch verwendet werden können, um die Füße an der Drohne zu befestigen.

Powerbank und Akku werden mithilfe von Kabelbindern an der Drohne befestigt. Für den zweiten Radarsensor wird keine Halterung entworfen. Die Gründe dafür werden im nächsten Punkt dieser Arbeit diskutiert.

In der folgenden Abbildung wird der vorläufige, reale Aufbau dargestellt:

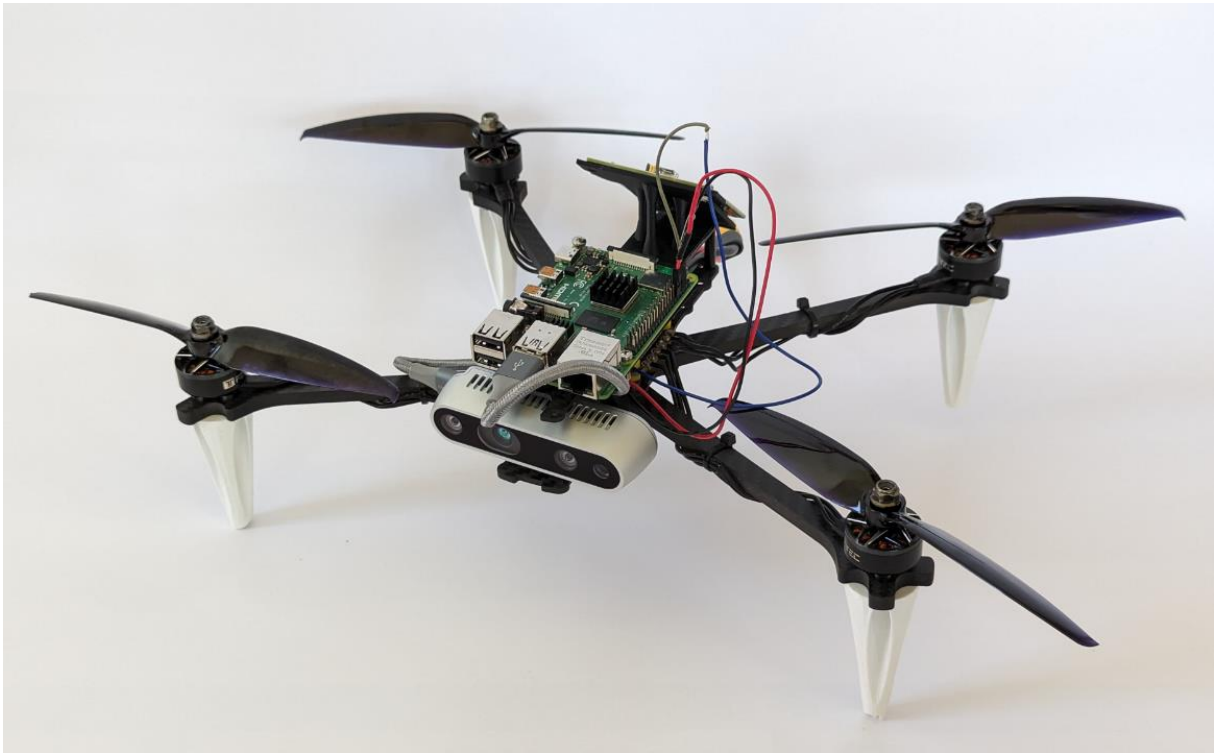


Abbildung 32: Erster Aufbau der Drohne

Die abgebildeten Kabel werden mit Kabelbindern fixiert, damit sie die Rotoren nicht behindern. Zudem wird geprüft, dass alle Teile fest fixiert sind, um zu verhindern, dass sich Teile im Flug lösen und kaputt gehen. Damit soll auch ausgeschlossen werden, dass die Drohne abstürzt. Zur Prüfung werden mehrere Flugtests an verschiedenen Orten durchgeführt. Es wurden auch Tests innerhalb des Hochregallagers vorgenommen. Bei diesen stand die Funktionalität der Sensoren im Vordergrund.

3.2.7 Diskussion des Sensorkonzepts

Anhand der durchgeführten Tests werden im Folgenden Stärken und Schwächen des Sensorkonzepts diskutiert. Das Ziel ist es, vorhandene Schwächen zu beseitigen und den Aufbau dementsprechend anzupassen.
Positiv hervorzuheben ist zunächst das wenige Zusatzgewicht. Sowohl der Raspberry Pi als auch die Radarsensoren fügen dem Gesamtkonstrukt nur wenig Gewicht hinzu. Die einzig schweren Komponenten sind die Tiefenkamera und die Powerbank. Die Summe der Bestandteile übersteigt das zulässige Maximalgewicht bei Weitem nicht. Der Vorteil zeigt sich vor allem daran, dass die Drohne an allen Testorten problemlos abheben kann und die Kraft der Drohne sogar reduziert werden muss, um sie besser steuern zu können. Insgesamt ist das Flugverhalten der Drohe konstant.

Die Tiefenkamera erkennt die Regalstreben fast immer und kann problemlos Bilder aufnehmen, wodurch sie sich gut ins Gesamtkonzept integriert. Allerdings bringt ihre gute Leistung auch einen Nachteil mit sich. Das Problem, das sich hier offenbart, ist, dass der Raspberry Pi nicht genug Rechenleistung besitzt, um den Datenstrom der Kamera flüssig auszuwerten. Die Bilder der Kamera werden teilweise mit weniger als einem fps angezeigt. Die Auswertung der Daten ist zu langsam. Es kann damit nicht gewährleistet werden, dass die Drohne rechtzeitig auf mögliche Hindernisse reagieren kann. Da sich die Kamera wie bereits angesprochen ansonsten sehr gut in das Gesamtkonzept einfügt soll nicht auf sie verzichtet werden. Daher muss die Hardware verbessert werden.

Eine weitere Herausforderung, die sich bei den Tests im Hochregellager zeigt, ist, dass die Radarsensoren Probleme haben, die Regalstreben zu erfassen. Die Werte der Sensoren sind die meiste Zeit zu hoch. Tests an leeren Regalen machen deutlich, dass das mangelnde Erkennen der Streben dazu führen kann, dass die Drohne in die Regale fliegt und möglicherweise abstürzt und/oder Ware innerhalb der Regale beschädigt.

Auch wenn der Aufbau einige Vorteile bietet, sind die Herausforderungen zu groß, um den Aufbau final auf diese Weise zu realisieren. Jedoch ergeben sich aus den Kritikpunkten wichtige Anforderungen an den nächsten Aufbau. Konkret wird eine stärkere Hardware benötigt, die genug Leistung aufbringt, um die Sensoren mit ausreichender hoher Geschwindigkeit auszulesen. Daneben muss die Überwachung der Abstände und der Umgebung umfassender und genauer werden. Zur Abstandsmessung sollen Sensoren befestigt werden, die nach oben und nach hinten gerichtet sind. Auch steht die Idee im Raum, eine Art sensorische Sphäre um die Drohne herum aufzuspannen, in welcher alle Hindernisse erkannt werden.

3.3 Anpassung des Sensorkonzepts

Im Folgenden werden die Erkenntnisse aus dem ersten Aufbau für die Anpassung der Drohne und des Sensorkonzepts genutzt.

3.3.1 Vorbetrachtungen

Aus dem vorherigen Aufbau geht hervor, dass die Radarsensoren nicht für den Einsatz im Hochregallager geeignet sind. Daher wird eine zusätzliche Tiefenkamera am hinteren Ende der Drohne angebracht, die die Entfernung zum Regal mitüberwacht. Ein LIDAR-Sensor wird verbaut. Funktionsweise und Vorteile, die mit diesem einhergehen, werden in 2.3.1.2 und 3.2.5 thematisiert.

Da die Regale im Hochregallager eng aneinandergereiht sind, ist es für die Tiefenkamera schwierig, die Regalstreben zu finden, wenn sie in der Mitte des Fachs fliegt. Um sicherzustellen, dass die Kamera zu jedem Zeitpunkt die Streben detektieren kann, wird sie um 90° gedreht.

Eine größere Herausforderung ergibt sich beim Verbauen der LIDAR-Sensoren. Diese haben einen Detektionswinkel von 360°. Um eine sensorische Sphäre aufzubauen, sollen drei Sensoren verbaut werden – zwei vertikal und eine horizontal. Der horizontale LIDAR hat die Aufgabe, die Umgebung links und rechts abzusichern, während die beiden vertikalen Sensoren nach vorne, hinten, oben und unten hin überwachen sollen. Damit die vertikalen Sensoren ihren gesamten Messbereich ausschöpfen können, muss sichergestellt werden, dass sich kein anderer Teil der Drohne innerhalb des Messbereiches befindet. Aktuell befinden sich die Arme der Drohne und die Propeller in diesem Bereich.

Eine mögliche Lösung für dieses Problem wird in den nachfolgenden Abbildungen dargestellt. Diese zeigen den angepassten Aufbau des Sensorkonzepts.

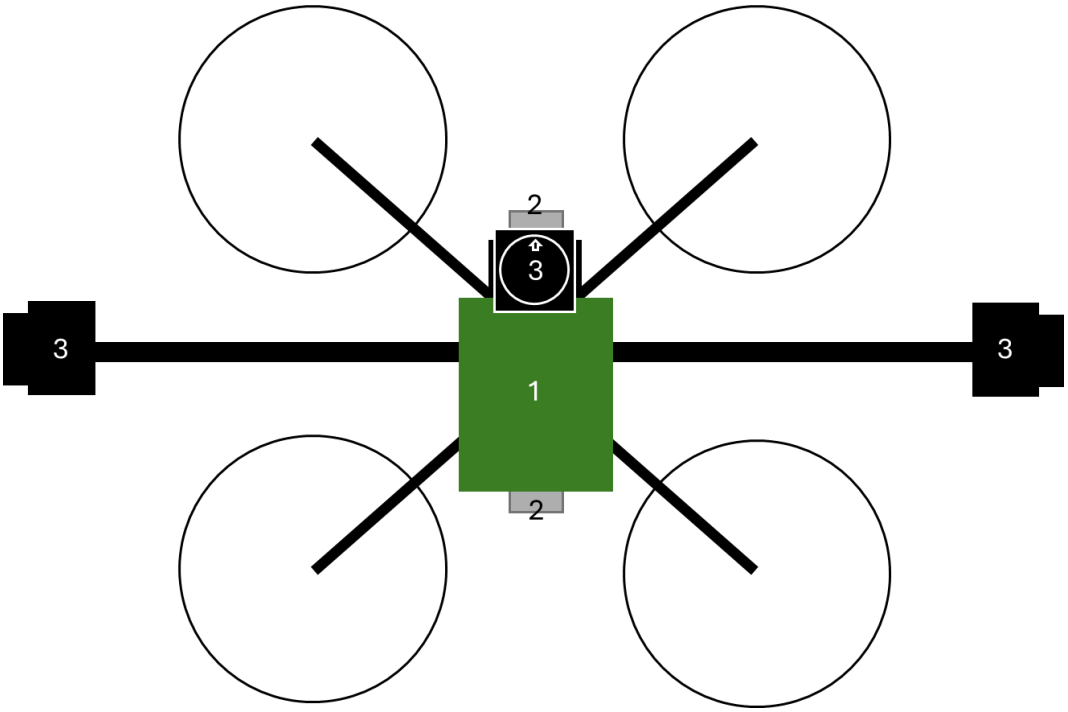


Abbildung 33: Anpassung von Drohne und Sensorkonzept von oben

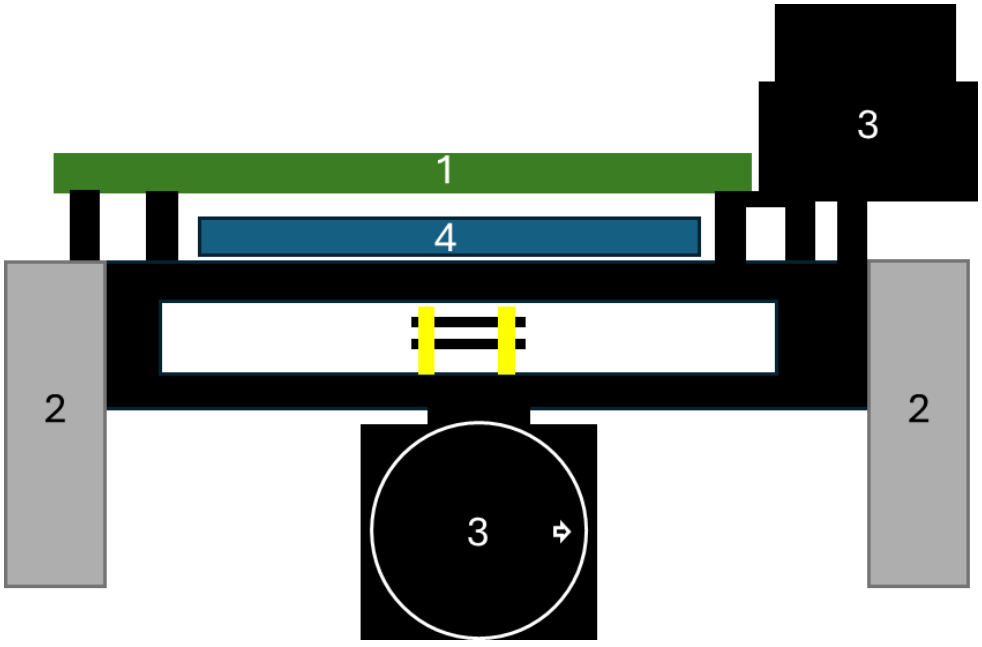


Abbildung 34: Anpassung von Drohne und Sensorkonzept von der Seite

In Abbildung 33 und 34 wird das neue Konzept zwei Perspektiven dargestellt. Es sind zwei Darstellungen nötig, da nicht alle Komponenten mit einer Darstellung abgebildet werden können. Mit den Zahlen in den Abbildungen werden die einzelnen Komponenten nummeriert:

1. Toradex-Board
2. Tiefenkamera
3. LIDAR-Sensor
4. 5G-Modem

In der ersten Abbildung ist zu erkennen, dass das Toradex-Board in der Mitte des Hauptkörpers oben befestigt wird. Daneben wird, ebenfalls mittig, der horizontale LIDAR-Sensor angebracht, der die Überwachung nach links und rechts abdeckt. Bei diesem Sensor muss darauf geachtet werden, dass er hoch genug angebracht wird, damit das Toradex-Board nicht in seinen Messbereich hineinragt. Die beiden vertikalen LIDAR-Sensoren befinden sich an den jeweiligen Enden der langen Ausleger, damit diese außerhalb der Reichweite der Propeller sind.
Wie in Abbildung 34 zu sehen, werden die Arme der Drohne aus Gründen der Übersichtlichkeit weggelassen. Jeweils vorne und hinten befindet sich eine Tiefenkamera. Diese sitzen nicht mehr auf dem Rahmen der Drohne, sondern finden sich an den Streben der Drohne und sind um 90° gedreht. Sie sitzen tiefer als der Rahmen, damit die Komponenten oben auf der Drohne mehr Platz haben. Dank neuer Drohnenfüße haben die Tiefenkameras genug Platz und berühren den Boden auch im Stand nicht.
Die Akkus werden in dieser Abbildung weggelassen. Ihre Position hat sich zum ersten Aufbau nicht verändert. Neu hingegen ist das 5G-Modem, das unterhalb des Toradex-Boards befestigt wird. Dadurch, dass es kleiner als das Toradex-Board ist, passt es ideal zwischen die Kameras. Dort ist auch genug Platz, um Befestigungen für das Toradex bereitzustellen.

3.3.2 3D-Druck

Im folgenden Abschnitt werden, wie beim ersten Entwurf, die Teile betrachtet, welche in 3D entworfen und anschließend gedruckt werden. Es wird der gleiche Drucker und das gleiche Filament genutzt.
Die erste hier vorzustellende Konstruktion ist die einzige, die aus dem ersten Entwurf übertragen und verändert wird. Dabei handelt es sich um die Halterungen der zwei Tiefenkameras. Diese werden in der folgenden Abbildung dargestellt:

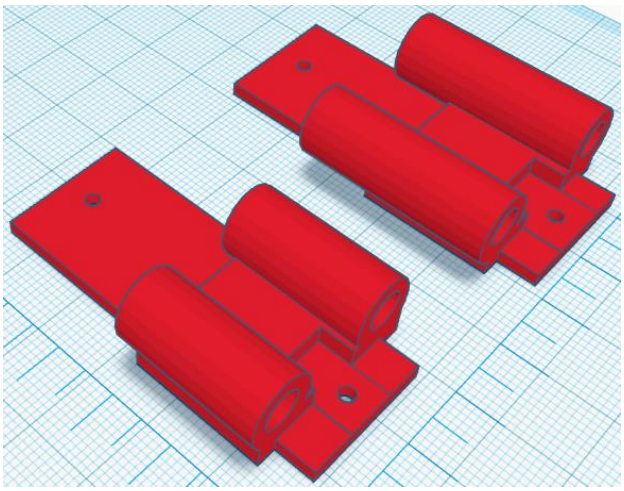


Abbildung 35: Anpassung der Halterungen der Tiefenkameras

Die Bodenplatte mit der Noppe wird in diesem Entwurf weggelassen, da die Noppe während der Versuche abbricht. Das fällt allerdings erst auf, nachdem die Konstruktion auseinandergebaut wird. Das heißt, die Noppe leistet auch keinen großen Beitrag zur Stabilität der Kamera. Die Ausrichtung der Kamera wird ebenfalls geändert. Dafür wird die Platte, an der die Kameras verschraubt werden, im Verhältnis zu den Röhren, die um die Streben des Rahmens der Drohne kommen, um 90° gedreht. Am oberen Schraubenloch wird eine Aussparung gelassen, damit der Schraubenkopf Platz hat. Erwähnenswert ist zudem, dass die Streben des Rahmens vorne und hinten eine unterschiedliche Länge haben. Aus diesem Grund sind die Röhren der zwei Halterungen unterschiedlich lang.

Als Nächstes wird eine Halterung für die LIDAR-Sensoren gebaut:

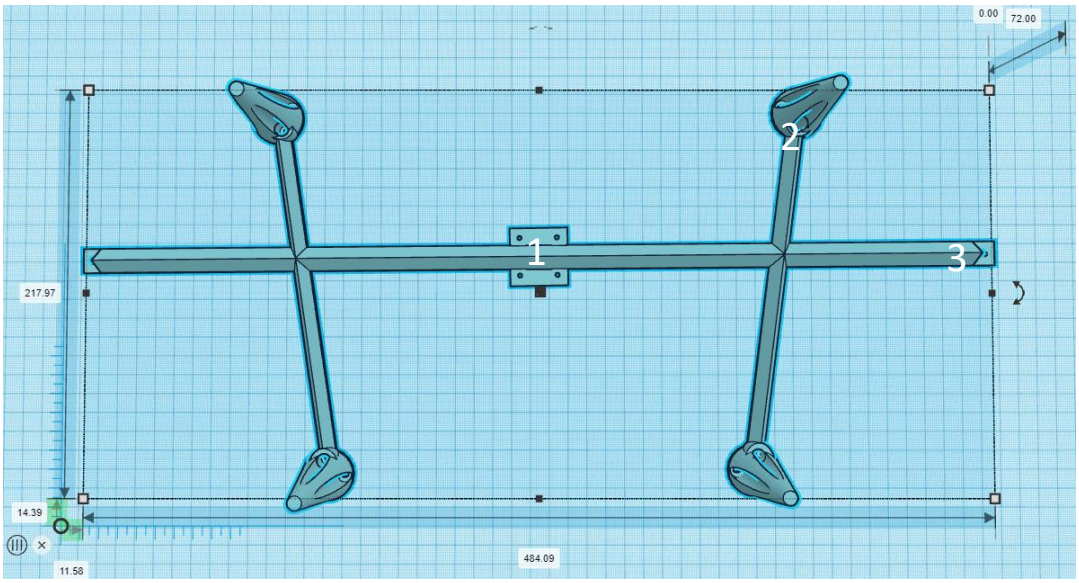


Abbildung 36: Gesamte LIDAR-Halterung

Zwei Anforderungen führen zum Bau der in Abbildung 40 dargestellten Halterung: einerseits sollen sich die LIDAR-Sensoren außerhalb der Reichweite der Propeller befinden und andererseits stabil auf einer Höhe montiert werden, statt herunterzuhängen. Eine lange Stange wäre allein nicht stabil und sicher genug. Daher wird eine weitere Stange zwischen die Füße der Drohne angebracht, die die quere Stange unterstützen soll. In Abbildung 36 mit Nummer 1 versehen ist der Mittelpunkt der Konstruktion. An dieser Stelle wird die Drohne verschraubt. Dafür werden die Löcher verwendet, an denen die Beine der Drohne verschraubt sind. Nummer 2 sind die Füße der Drohne. Diese werden aus dem ersten Entwurf entnommen und zusätzlich an die LIDAR-Halterung befestigt, was für mehr Stabilität bei den Sensoren sorgt. Mit Nummer 3 ist die Stelle gekennzeichnet, an der später die LIDAR-Sensoren befestigt werden. Dafür wird ein weiteres 3D-Teil verwendet, auf welches im Anschluss eingegangen wird.

Eine Herausforderung, die mit diesem Entwurf einhergeht, ist, dass der Rahmen keinen Spielraum lässt. Alle Maße und Winkel müssen stimmen, damit das Bauteil mit der Drohne verschraubt werden kann. Das heißt auch, dass die Konstruktion am Ende möglichst symmetrisch sein muss, um das Flugverhalten der Drohne nicht zu beeinflussen. Ebenfalls herausfordernd ist die Größe des Bauteils. Mit einer Länge von 484.09 mm ist es zu groß für die meisten 3D-Drucker. Daher muss die Halterung in vier Einzelteile zerlegt werden:

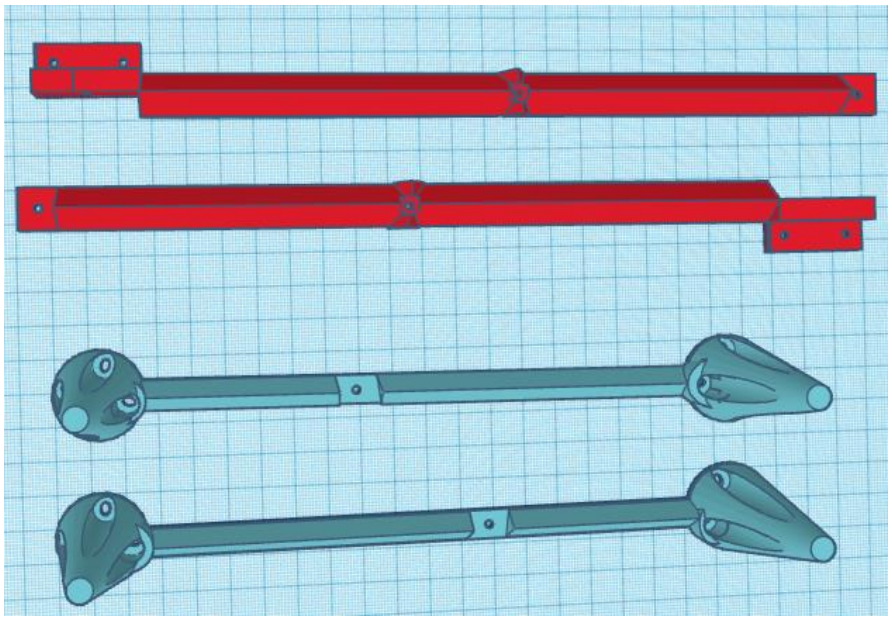


Abbildung 37: Zerlegte LIDAR-Halterung

Die lange Mittelstange wird in der Mitte geteilt und die beiden Stangen zwischen den Füßen werden von der Mittelstange gelöst. An den Stellen, an denen eine Trennung vorgenommen wird, wird darauf geachtet, dass die Verbindungsstücke mit Material gefüllt sind, damit keine Stabilität verloren geht. Ebenfalls muss beachtet werden, dass die Fußstücke auf die jeweils richtige Seite befestigt werden.

Das nächste Bauteil, welches betrachtet wird, ist das bereits erwähnte Bauteil, dass dazu verwendet werden soll, LIDAR-Sensoren mit der Halterung zu verschrauben. Es wird in der folgenden Abbildung dargestellt:

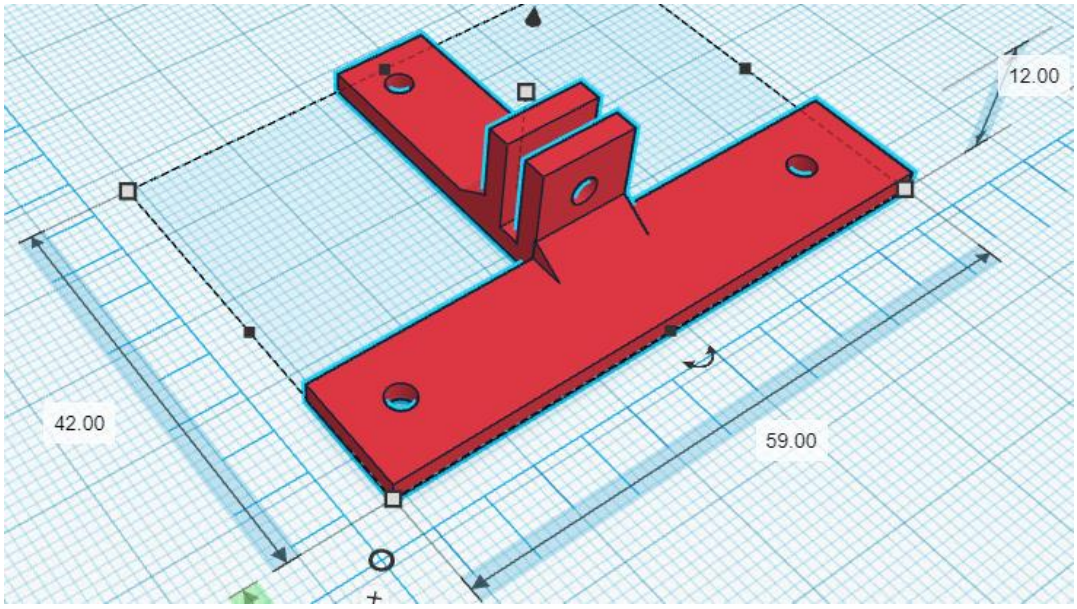


Abbildung 38: Zusatz zur LIDAR-Halterung

Zu sehen ist ein T-förmiges Bauteil. An den drei Enden findet sich jeweils ein Loch, das mit dem LIDAR verschraubt werden soll. In der Mitte sind zwei parallel zueinanderstehende Platten, zwischen die der Ausläufer des großen Halters angebracht wird. Dafür befinden sich in der Mitte zwei Löcher zum Verschrauben. Wichtig ist an dieser Stelle, dass der Ausläufer bis zum Ende der Öffnung reicht, damit sich der kleine Halter nicht nach links und rechts bewegen kann.
Das letzte Bauteil, das entworfen wird, ist die Halterung für das Toradex-Board, das 5G-Modem und den dritten LIDAR. Dieses Bauteil wird in den folgenden Abbildungen 39 und 40 dargestellt:

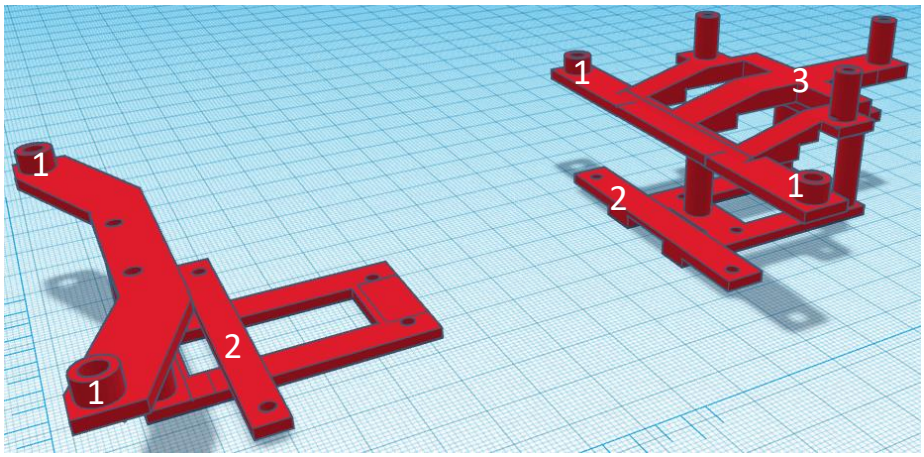


Abbildung 39: Toradex-Halterung von der Seite

Das Bauteil gliedert sich in zwei Teile. Da eine Verbindung keinen Vorteil bringen würde und in dieser Weise Material gespart werden kann, wird die Zweigliederung vorgenommen.
Mit der 1 werden die vier Löcher gekennzeichnet, an denen das Toradex-Board festgeschraubt wird. Dabei sind diese Löcher etwas erhöht, da das Toradex-Board keine flache Unterseite besitzt. Die Nummer 2 markiert die beiden Orte, an denen das 5G-Modem verschraubt wird. Im Gegensatz zum Toradex müssen die Löcher hier nicht erhöht werden. Das Modem sitzt zwischen der Drohne und dem Toradex-Board. Die 3 gibt den Bereich an, an dem der dritte LIDAR-Sensor befestigt werden soll. Dieser ist wieder erhöht, da sich auf dem Toradex ein Lüfter befindet, der nicht in den Messbereich des Sensors ragen soll. Die Erhöhung wird mithilfe von längeren Schrauben realisiert. Schließlich wird der untere Teil des Bauteils mit Streben mit dem oberen Abschnitt verbunden (siehe linkes Bauteil in Abbildung 39). Hier gibt es eine Besonderheit, die in der folgenden Abbildung zu sehen ist:

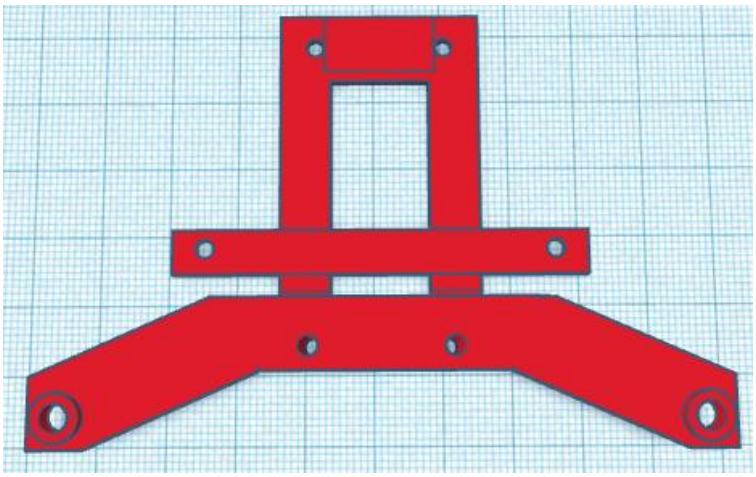


Abbildung 40: Toradex-Halter von hinten oben

Die Streben, welche in den Ecken des Bauteils sitzen, sind hohl. Grund dafür ist, dass die Streben in der vorherigen Version häufig abgebrochen sind, da der Übergang von der Platte zur Strebe eine Sollbruchstelle gebildet hat. Dadurch, dass durch diese Streben Schrauben verlaufen, die das Bauteil mit dem Rahmen der Drohne verbinden, können diese nicht abbrechen.

3.3.3 Toradex-Board

Im ersten Schritt wird die Hardware verbessert. Wie sich in den Tests gezeigt hat, liefert der Raspberry Pi nicht genug Rechenleistung, um die Sensordaten mit ausreichender Geschwindigkeit auszulesen. Aus diesem Grund muss auf einen anderen Einplatinenrechner gesetzt werden. Die Entscheidung fällt hier auf das Toradex-Modul „Apalis iMX8“.

Abbildung 41: Toradex-Board „Apalis iMX8“ (übernommen von: Toradex Group AG o.A.a)

Die bessere Leistung wird durch den „leistungsstärksten Applikations-Prozessor der aktuellen NXP® i.MX 8-Serie – dem i.MX 8QuadMax“ (Toradex o.A.) bereitgestellt. Dieser beinhaltet „zwei Cortex-A72 und vier Cortex-A53 Applikations-Prozessor-Cores sowie zwei Cortex-M4F Microcontroller-Cores mit FPU“ (ebd.). Wie in Abbildung 41 zu erkennen ist, ist dieses Modul nicht für den alleinstehenden Gebrauch gedacht. Um das Modul vollumfänglich nutzen zu können, wird ein Carrier Board benötigt. Die Aufgabe dieses Carrier Boards ist es, dem Modul Apalis die Kommunikation mit der Außenwelt zu ermöglichen. Hier kommt ein „Ixora Carrier Board V1.2“ zum Einsatz, welches in der folgenden Abbildung zu sehen ist:

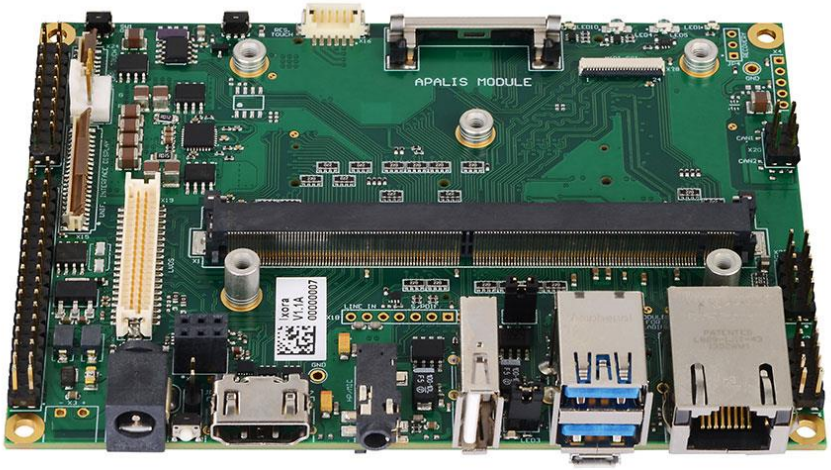


Abbildung 42: Carrier Board „Ixora Carrier Board V1.2“ (übernommen von: Toradex Group AG o.A.c)

Auf der Abbildung sind diverse Standard-Schnittstellen zu erkennen, welche genutzt werden können, um mit dem Apalis Modul zu interagieren. Dazu zählen z.B. Ethernet, HDMI, USB 2 und 3. Die folgende Abbildung stellt alle Anschlüsse des Carrier Boards dar und zeigt, wie diese mit dem Apalis Modul verbunden sind:

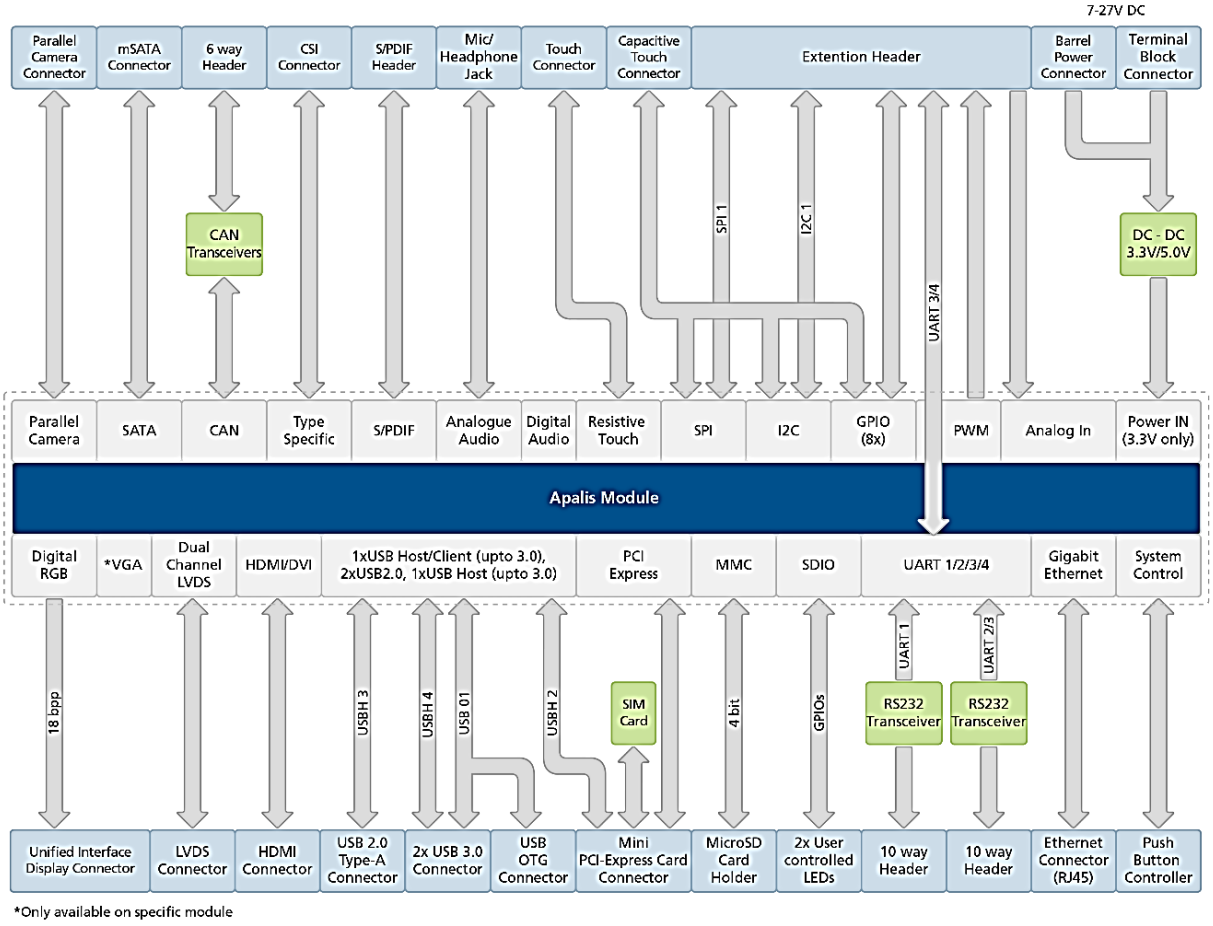


Abbildung 43: Anschlüsse des Carrier Boards „Ixora Carrier Board“ (übernommen von: Toradex Group AG o.A.c)

Sobald das Apalis Modul auf dem Carrier Board angebracht wird, kann damit gearbeitet werden. Softwaretechnisch bietet das Toradex diverse Möglichkeiten. Mit Hilfe des vorinstallierten „Toradex Easy-Installer“ können verschiedene Betriebssysteme wie z.B. FreeRTOS QNX und Android auf das Board installiert werden. Für diese Arbeit wird mit Torizon Embedded Linux OS gearbeitet. Diese hauseigene Linux-Variante von Toradex wurde geschaffen, um mit einem entwicklerfreundlichem Ökosystem den Entwicklungsprozess zu vereinfachen (Toradex Group AG o.A.b). Docker ist hier bereits vorinstalliert und stellt einen wichtigen Bereich des Betriebssystems dar.

3.3.4 Wechsel zu Toradex

Was Docker ist, welche Vor- und Nachteile es bietet und wie es funktioniert, darauf wird im Theorieteil ausführlich eingegangen. In diesem Abschnitt werden Änderungen besprochen, die vorgenommen werden müssen, um die Kompatibilität zwischen den Sensoren und dem Toradex Board zu gewährleisten. Dafür werden die einzelnen Container gelistet, die für den jeweiligen Sensor erstellt werden. Diese weichen quellcodetechnisch nur wenig von den bisher erstellten Programmen ab. Besonders interessant sind hier die für die Sensoren geschriebenen „dockerfiles“.

3.3.4.1 LIDAR

Die kleinsten Änderungen werden beim LD19 LIDAR vorgenommen. Der Quellcode bleibt fast derselbe. Nur „fast“, da die Variable für pySerial geändert wird. Statt „/dev/ttyACM0“ wird „/dev/ttyLP1“ verwendet, da die UARTs auf dem Toradex Board nicht mit ACM sondern mit LP angesprochen werden. Auf dem Toradex gibt es vier verwendbare UARTs. Davon kommunizieren zwei UARTs direkt mit dem Apalis Modul, während die Signale der anderen zwei UARTs erst durch einem RS232 Transceiver geschickt werden (siehe Abbildung 43). Die UARTs werden mit LP0-4 angesteuert. Dabei entspricht UART1=LP1, UART2=LP3, UART3=LP0 und UART4=LP2.

Eine andere Änderung, die am LIDAR-Code vorgenommen wird, wird in den beiden folgenden Abbildungen deutlich:

```
44    socket = socket.socket()
45    socket.connect((HOST , PORT))
```

Abbildung 44: Aufbau einer Socket-Verbindung in LIDAR

```
193           L = pickle.dumps(L)
194           socket.send(L)
```

Abbildung 45: Versenden von Daten in LIDAR über die Socket-Verbindung

Diese Zeilen werden benötigt, um die Daten der LIDAR-Sensoren an andere Container mittels tcp/ip zu versenden. Im Vergleich zum Empfangen von Daten, wird zum Verschicken deutlich weniger Quelltext benötigt. Um Daten zu verschicken, muss zuerst ein Socket-Objekt erstellt werden, das sich in der nächsten Zeile mit einem Host verbindet. IP-Adresse und Port sind gegebene Variablen, die im Vorfeld definiert werden. Im nächsten Schritt werden die Daten gepickelt, wodurch ein Byte-Stream

entsteht, welcher besser verschickt werden kann als einzelne Daten. (Vgl. Jennings o.A.)
Das Dockerfile des LIDAR-Sensors wird in der folgenden Abbildung dargestellt:

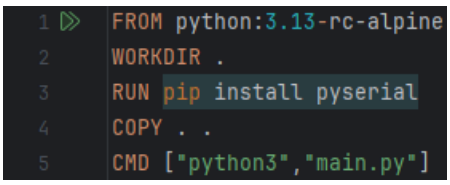


Abbildung 46: Docker-File der LIDAR-Sensoren

Die Basis für diesen Container ist das Image „python:3.13-rc-alpine“ (siehe Zeile 1). Alpine ist eine kleine Variante von Python, die weniger Features bietet, dafür aber auch weniger Speicherplatz benötigt. Bei Containern sollte generell darauf geachtet werden, sie möglichst ressourcensparend zu gestalten. Daher wird dieser Weg gewählt. In der zweiten Zeile von Abbildung 51 wird die Working Directory definiert, bevor in Zeile 3 die Library pySerial installiert wird. Im darauffolgenden Schritt werden die Daten aus dem aktuellen Verzeichnis in den Container kopiert. Zuletzt folgt die Festlegung, was geschehen soll, wenn der Container gestartet wird.

3.3.4.2 Tiefenkamera

Auch beim Programm der Kamera ändert sich wenig. Der ausschlaggebende Punkt ist an dieser Stelle das Dockerfile. Wie bereits beim letzten Aufbau erwähnt, sind die Bibliotheken der Kamera nicht von Natur aus mit Linux kompatibel. Daher muss im Dockerfile sichergestellt werden, dass das Programm auf dem auf Linux basierendem Toradex Board lauffähig ist. In der folgenden Abbildung wird zunächst das Programm vorgestellt, das benutzt wird, um die Daten von zwei Kameras auszulesen, welche dann an einen anderen Container geschickt werden:

main.py

```python
import pyrealsense2 as rs
import numpy as np
import cv2
import socket
import pickle
from time import sleep
HOST = '192.168.92.21'
PORT = 16969
sleep(5)
socket = socket.socket()
socket.connect((HOST , PORT))

# Configure depth and color streams...
# ...from Camera 1
pipeline_1 = rs.pipeline()
config_1 = rs.config()
config_1.enable_device('233722070206')
config_1.enable_stream(rs.stream.depth , 640 , 480 , rs.format.z16 , 30)
config_1.enable_stream(rs.stream.color , 640 , 480 , rs.format.bgr8 , 30)
# ...from Camera 2
pipeline_2 = rs.pipeline()
config_2 = rs.config()
config_2.enable_device('233622078911')
config_2.enable_stream(rs.stream.depth , 640 , 480 , rs.format.z16 , 30)
config_2.enable_stream(rs.stream.color , 640 , 480 , rs.format.bgr8 , 30)
# Start streaming from both cameras
pipeline_1.start(config_1)
pipeline_2.start(config_2)

while True:
    frames_1 = pipeline_1.wait_for_frames()
    depth_frame_1 = frames_1.get_depth_frame()
    depth_image_1 = np.asanyarray(depth_frame_1.get_data())

    frames_2 = pipeline_2.wait_for_frames()
    depth_frame_2 = frames_2.get_depth_frame()
    depth_image_2 = np.asanyarray(depth_frame_2.get_data())

    array_list=np.array([depth_image_1,depth_image_2])

    x=socket.recv(16)
    if x == b"start":
        socket.sendall(pickle.dumps(array_list))
```

Abbildung 47: Anpassung des Programms zum Auslesen der Daten der Tiefenkameras

Das gezeigte Programm unterscheidet sich vom Programm, das auf dem Raspberry verwendet wurde, in zweierlei Hinsicht. Als Erstes werden hier zwei Kameras anstatt einer ausgelesen. Der Quellcode, der zum Auslesen genutzt wird, ist daher ein zweites Mal vorhanden. Alle möglichen Variablen werden, ihrer Kamera entsprechend, entweder mit einer 1 oder einer 2 gekennzeichnet. Für die zweite Kamera muss eine andere Seriennummer eingetragen werden, die wie bei der ersten Kamera auf der

Unterseite des Geräts zu finden ist. Der zweite Unterschied zum ersten Programm ist der Quelltext, der verwendet wird, um die Daten an andere Container zu verschicken. Vor dem Versenden werden die Arrays beider Kameras in einer Variable zusammengefasst. Zusätzlich dazu wird eine Synchronisation durchgeführt. Diese ist notwendig, da der Host-Container und der Container der Kameras unterschiedliche Zeiten benötigen, um einsatzbereit zu sein. Dazu wird vom Host-Container ein Start-Signal gesendet, bevor mit der Übertragung begonnen wird.
Das Dockerfile für die Kameras sieht wie folgt aus:

Dockerfile

```
1  FROM nixone/pyrealsense2:focal-py3.8-rs2.53.1
2
3  WORKDIR .
4
5  RUN apt-get update
6  RUN pip install numpy
7  RUN pip install scipy
8  RUN pip install opencv-python
9  RUN pip install pyserial
10
11 COPY  main.py .
12
13 CMD ["python3", "main.py"]
```

Abbildung 48: Dockerfile für die Tiefenkameras

Der wichtigste Teil ist direkt in der ersten Zeile zu sehen. Anders als im ersten Aufbau wird wie bereits erwähnt ein anderes Image als Basis verwendet. Hier ist es „nixone/pyrealsense2:focal-py3.8-rs2.53.1“. Dieses Image wurde vom Nutzer nixone auf Docker-Hub hochgeladen und ermöglicht das Verwenden der Bibliothek pyrealsense2 innerhalb eines Docker Containers. Um diese nutzen zu können, müssen folgende Bibliotheken zusätzlich heruntergeladen werden: NumPy, das für das Arbeiten mit Arrays innerhalb von Python verwendet wird, SciPy, das wissenschaftliches Rechnen ermöglicht und OpenCV-Python, das zur Bildverarbeitung genutzt wird. Als Letztes wird pySerial für die Kommunikation mit den anderen Containern installiert.

3.3.4.3 Host Container

Das Dockerfile des Host Containers, welcher die Aufgabe hat, alle Daten zu empfangen und weiterzuverarbeiten, ist ähnlich aufgebaut wie das Dockerfile des LIDAR-Sensors. Anstelle von pySerial wird hier NumPy installiert. Diese Bibliothek wird zur Verarbeitung der Kameradaten eingesetzt. Der Code zur Datenverarbeitung kann in zwei Teile unterteilt werden. Der erste Teil wird in folgender Abbildung dargestellt:

main.py

```
import socket
import pickle
import json
import numpy

serversocket1 = socket.socket()
serversocket2 = socket.socket()
serversocket3 = socket.socket()
host = '192.168.92.21'
port1 = 13370
port2 = 14040
port3 = 16969
x = 0
y = 0
z = 0
i = 0
try:
    serversocket1.bind((host,port1))
except socket.error as e:
    print(str(e))
print('Waiting for cennection 1')
serversocket1.listen(5)
client1 , address1 = serversocket1.accept()
try:
    serversocket2.bind((host , port2))
except socket.error as e:
    print(str(e))
print('Waiting for cennection 2')
serversocket2.listen(5)
client2 , address2 = serversocket2.accept()
try:
    serversocket3.bind((host , port3))
except socket.error as e:
    print(str(e))
print('Waiting for cennection 3')
serversocket3.listen(5)
client3 , address3 = serversocket3.accept()
```

Abbildung 49: Erster Teil des Programms zur Datenverarbeitung im Container

In diesem Teil werden zuerst die Libraries importiert, bevor anschließend die Ports und die IP-Adresse definiert werden. Die Ports werden zu einem späteren Zeitpunkt im

Docker Compose für diesen Container freigegeben. Im nächsten Schritt werden die Variablen deklariert, die verwendet werden sollen und Verbindungen mit den einzelnen Containern etabliert. Dafür wird zuvor jeweils ein Socket-Objekt in Zeile 6 bis 8 erstellt. Mit dem Befehl „.bind()“ wird jeder Socket mit einem Netzwerk und einem Port verbunden. Nach dem Erstellen dieser Verbindung wird mit „.listen()“ der Socket zu einem empfangenen Socket konvertiert. Dies ermöglicht es ihm, die Verbindungen zu akzeptieren. Abschließend wird mit dem Befehl „.accept()“ ein neues Objekt erstellt. Dieses Objekt stellt die Verbindung dar, an welcher die Daten ankommen. (Vgl. Jennings o.A.)

Das Empfangen der Daten wird im zweiten Teil des Programms ersichtlich:

```
39  while True:
40      i=i+1
41      if i == 100:
42          data = b""
43          client3.sendall(b"start")
44          while True:
45              z = client3.recv(4096)
46              data += z
47              if len(data) >= 1228965: break
48          data_arr = pickle.loads(data)
49          print(data_arr)
50          i=0
51      x = client1.recv(1500)
52      y = client2.recv(1500)
53      try:
54          x = pickle.loads(x)
55          print("DataX:")
56          print(x)
57      except:
58          pass
59      try:
60          y = pickle.loads(y)
61          print("DataY:")
62          print(y)
63      except:
64          pass
```

Abbildung 50: Erster Teil des Programms zur Datenverarbeitung im Container

In Zeile 46, 55 und 56 wird deutlich, mit welchem Befehl die Daten empfangen werden. Dabei steht die Zahl, die in Klammern steht, für die Menge an empfangenen Bytes. 4096 ist das Maximum. Nach dem Empfangen der LIDAR-Daten werden diese von einem Bytestream zurückkonvertiert. Danach können die Daten nach Belieben weiterverwendet werden.

Das Empfangen der Daten der Kamera ist komplexer. Zuerst wird mit einer if-Bedingung dafür gesorgt, dass die Daten der Kameras bei jedem hundertsten Schleifendurchlauf ausgegeben werden. Die Ausgabe der Kameradaten würde ansonsten die Ausgabe der LIDAR Daten bremsen, da die Daten aufgrund der hohen Hardwareanforderungen in einer geringeren Frequenz ankommen. Da das Datenpacket, welche vom Kamera-Container gesendet wird, die Höchstzahl von 4096 Bytes überschreitet, muss Code hinzugefügt werden, um die Daten fehlerfrei empfangen zu können.
Dafür wird eine Variable definiert, deren Datentyp Bytes sind. Diese Festlegung setzt die Variable in fortlaufenden Schleifenwiederholungen auf ihren Ausgangswert zurück. Danach wird ein Startsignal an den Kamera-Container gesendet. In der darauffolgenden Schleife werden die Daten zunächst empfangen und danach an die vorher definierte Variable angehangen, bis die Übertragung vollendet ist. Sobald die Länge der Speichervariable 1228965 überschreitet, ist die Übermittlung der Daten abgeschlossen. Die Speichervariable wird von einem Bytestream zurückkonvertiert. Die Daten liegen schließlich wieder in der Form vor, in der sie losgeschickt wurden und können weiterverwendet werden.

3.3.4.4 Docker Compose

Das „docker-compose.yml“-File wird für das Netzwerk verwendet, das auf der Drohne laufen soll. Der Inhalt dieses Files lässt sich der folgenden Abbildung entnehmen:

docker-compose.yml

```yaml
version: "3.9"
services:
  datenverarbeitung:
      image: edgarlumpe/datenverarbeitung
      ports:
        - "13370:13370"
        - "14040:14040"
        - "16969:16969"
      restart: always
      networks:
        netz:
          ipv4_address: 192.168.92.21

  lidar1:
    image: edgarlumpe/lidar1
    ports:
      - "13380:13380"
    restart: always
    devices:
      - "/dev/ttyLP0:/dev/ttyLP0"
    networks:
      netz:
        ipv4_address: 192.168.92.22
    depends_on:
      - datenverarbeitung
  lidar2:
    image: edgarlumpe/lidar2
    ports:
      - "13390:13390"
    restart: always
    devices:
      - "/dev/ttyLP2:/dev/ttyLP2"
    networks:
      netz:
        ipv4_address: 192.168.92.23
    depends_on:
      - datenverarbeitung
```

Abbildung 51: Erster Teil des Docker Compose Files

```
38  cameras:
39    image: edgarlumpe/cameras
40    ports:
41      - "16970:16970"
42    restart: always
43    privileged: true
44    networks:
45      netz:
46        ipv4_address: 192.168.92.24
47    depends_on:
48      - datenverarbeitung
49  networks:
50    netz:
51      ipam:
52        driver: default
53        config:
54          - subnet: "192.168.92.0/24"
```

Abbildung 52: Zweiter Teil des Docker Compose Files

Das File ist dem Beispiel-File aus dem Theorieteil sehr ähnlich. Der einzige Unterschied zum File im Theorieteil besteht im Namen des Netzes. Es wird ein eigenes Netzwerk eingerichtet, damit allen Containern statische IP-Adressen zugeteilt werden können. Auch hier gibt es mehrere Container, die mit unterschiedlichen Einstellungen gestartet werden sollen. Die Container umfassen: einen Host Container, der alle Daten von den Sensoren empfangen und bündeln soll, einen Container, der beide Kameras ausliest und deren Daten an den Host Container schickt und zwei Containern, die jeweils die Daten eines LIDARs auslesen und diese an den Host schicken. Für jeden LIDAR wird ein eigener Container gestartet. Der Unterschied der beiden Container besteht darin, dass verschiedene Devices angesprochen und unterschiedliche Ports adressiert werden. Bei der Startreihenfolge der Container gibt es nur die Vorgabe, dass der Host als Erstes starten soll.

3.3.5 Anbindung an das 5G-Campusnetz

Die Anbindung an das 5G-Netzwerk geschieht hier durch ein vorkonfiguriertes Modem. Im Rahmen eines anderen Projektes in der Firma exceeding solutions wurde dieses Modem so konfiguriert, dass es angeschlossene Geräte mit dem 5G-Netz verbindet. Das Modem wird an das Toradex Board angeschlossen und ermöglicht damit eine Verbindung zu einem 5G-Netz. Diese Verbindung wird automatisch hergestellt und bedarf keiner weiteren Konfiguration. Das Docker Compose Netzwerk ist so eingestellt, dass die Container nach außen hin kommunizieren können. Der Host

Container kann dementsprechend über das 5G-Netz Daten an einen anderen Computer verschicken. Dabei wird auf die gleiche Kommunikationsart gesetzt, die auch verwendet wird, um die Daten der Sensoren an den Host zu schicken. Auch dem Empfangen von Daten sind keine Grenzen gesetzt. Die Drohne kann dementsprechend Befehle von außen erhalten, die ihr z.B. sagen, zu welchem Fach sie als Nächstes fliegen soll. Die einzige Schwierigkeit, die nicht ausgeschlossen werden kann, ist, dass das Hochregallager nicht flächendeckend mit 5G ausgeleuchtet werden kann. Da es dadurch zu Verbindungsabbrüchen kommen kann, ist es nicht möglich, direkte Steuerbefehle (wie z.B. links, rechts, hoch und runter) über 5G zu vergeben. Daher wird die 5G-Verbindung vor allem dafür genutzt, Bilder der Drohne an die Bodenstation zu liefern und zum anderen um der Drohne indirekte Befehle für das weitere Vorgehen zu senden, z.B. zu welchem Fach sie sich bewegen soll.

3.4 Ergebnisse des finalen Aufbaus

Der Aufbau der Drohne kann zunächst wie geplant umgesetzt werden. Die fertige Drohne sieht wie folgt aus:

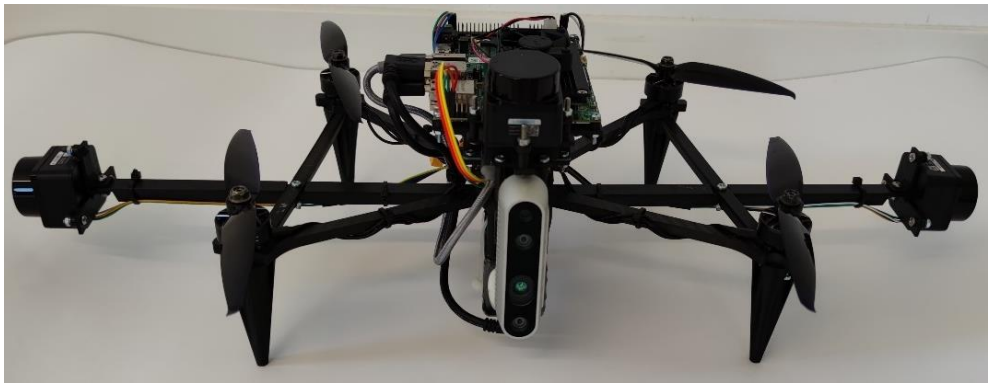


Abbildung 53: Finaler Aufbau der Drohne

Um die Drohne für ihre zukünftige Aufgabe zu testen, wurden erste Flugversuche durchgeführt. Die Sensoren waren dabei zwar eingeschaltet, ihre Daten wurden aber noch nicht ausgelesen. Da die Entwicklung des autonomen Fliegens zu diesem Zeitpunkt nicht fortgeschritten genug war, wurde die Drohne mittels Fernsteuerung geleitet. Die Ergebnisse der Versuche werden im Folgenden dargestellt:

Das angepasste Sensorkonzept funktioniert. Die Drohne ist in der Lage, ihre Umwelt korrekt zu erfassen und Hindernissen in der Theorie zu entgehen. Zudem ergänzen die Sensoren einander stimmig. Die LIDAR-Sensoren haben zuverlässige Daten gesendet. Die sich bewegenden Teile innerhalb der Sensoren zeigen keinen negativen

Einfluss auf das Flugverhalten. In den Tests hat sich lediglich gezeigt, dass die LIDAR-Sensoren ein Problem mit direkter Sonneneinstrahlung haben. Da im Hochregallager jedoch keine Fenster verbaut sind, ist das Problem für diesen Anwendungsfall vernachlässigbar.
Die Tiefenkameras konnten ebenso von sich überzeugen. Die gelieferten Daten waren durchweg stimmig und ließen sich durch das Format, in dem sie ausgegeben wurden, leicht weiterverwenden. In einigen Tests hat sich gezeigt, dass die Kamera Probleme damit hat, Kanten sauber zu erkennen. Das liegt daran, dass die beiden Tiefenkameras versetzt voneinander sind und damit einen unterschiedlichen Blickwinkel auf die Kanten haben. Diesem Problem konnte allerdings entgegengesteuert werden, indem ein Filter über das Tiefenbild zur Glättung der Kanten gelegt wurde. Das ist ein gängiges Vorgehen in der Bildverarbeitung. Durch die Öffnungen an der Kamera ist es möglich, die Kamera auch auf anderen Gestellen zu montieren. Beide Sensoren – sowohl die LIDAR- als auch die Tiefensensoren – sind aufgrund ihres einfachen Designs und ihres geringen Gewichts nicht nur für dieses Projekt gut geeignet, sondern können auch in zukünftige Konzepte gut integriert werden. Das etwas höhere Gewicht der Tiefenkameras im Vergleich zu den LIDAR-Sensoren ist in Anbetracht ihres Nutzens mehr als gerechtfertigt.
Ebenfalls positiv anzumerken ist, dass das Flugverhalten sich im Vergleich zum ersten Aufbau minimal verbessert hat. Das lässt sich mit dem zusätzlichen Gewicht der Sensoren, des Toradex Boards und der neuen Halterungen begründen, die das Flugverhalten stabilisieren. Gleichzeitig hat das erhöhte Gewicht aber auch Nachteile ergeben. Für das Abheben benötigt die Drohne deutlich mehr Kraft als zuvor, sie stößt häufig an ihre Kraftgrenze. Das dauerhafte Nutzen der Maximalkraft hat zur Folge, dass Bauteile schneller verschleißen und dass die Flugzeit deutlich sinkt. Während die Drohne im ersten Aufbau für 15 Minuten fliegen konnte, war in den Tests nach weniger als fünf Minuten Schluss. Ein vollgeladener Akku mit einer Spannung von 16,8V hatte nach drei bis vier Minuten nur noch eine Spannung von 13V. Die Spannung eines 4S LiPo Akkus darf, wie bereits erklärt, nicht unter 12V fallen, da der Akku sonst Schäden davonträgt. Diese Schäden sind irreparabel und sorgen dafür, dass der Akku nach kurzer Zeit weggeworfen werden muss. Daher ist die Drohne trotz zahlreicher Vorteile hinsichtlich des Sensorkonzepts mit diesem Aufbau nicht nutzbar. Das Problem ist klar erkennbar – die Drohne wurde für deutlich weniger Payload dimensioniert. Zum Vergleich: Die Drohne aus dem ersten Entwurf wog 500 Gramm weniger als die neue.

Da sich die Firma Schaeffler eine deutlich höhere Flugzeit gewünscht hat, sind hier große Änderungen notwendig, um das zu Beginn formulierte Ziel erreichen zu können. Das Optimieren der 3D-gedruckten Einzelteile oder der Verzicht auf einzelne Sensoren zur Reduktion des Gewichts würden nicht das gewünschte Ergebnis bringen, da nur wenige Gramm eingespart werden könnten. Auch sind die Sensoren nur in ihrer Kombination zufriedenstellend. An dieser Stelle kann maximal diskutiert werden, inwiefern eine Reduktion der Sensoranzahl von Vorteil wäre. Jedoch würde eine erneute Abstimmung der einzelnen Sensoren viel Zeit in Anspruch nehmen, die eventuell zu keinem zufriedenstellenden Ergebnis führt.

Für den weiteren Verlauf ergeben sich mehrere Vorgehensmöglichkeiten. Die erste Möglichkeit ist, die Drohne neu zu dimensionieren. Dafür würden zunächst größere Bauteile angeschafft werden. Größere Propeller wären deutlich effizienter als die aktuell verbauten und würden mehr Schubkraft geben. Sie müssten aber auch mit einem größeren Rahmen und größeren Motoren kombiniert werden, um zum Einsatz zu kommen. Gleichzeitig müssten mehrere Faktoren bei einer Neudimensionierung berücksichtigt werden. Das ist einerseits der begrenzte Raum zwischen den Regalen und andererseits die ESCs. Es müsste überprüft werden, ob der ESC die für die größeren Komponenten benötigte Spannung und Stromstärke aushalten würde oder ob dieser ebenfalls durch eine Neuanschaffung ausgetauscht werden müsste. Schließlich müsste zum Schluss auch ein neuer Akku angebracht werden, sobald das Gewicht der einzelnen Komponenten feststeht. Damit ließe sich die maximale Flugzeit bereits in der Theorie errechnen. Im zweiten Schritt würden die restlichen Komponenten optimiert werden. In den Versuchen hat sich gezeigt, dass das Carrier Board auf dem Toradex für die Entwicklung und nicht für den realen Einsatz geeignet ist. Konkret sind mehr Features verbaut, als notwendig sind. Zudem konnten zwei von vier UARTs aufgrund des RS232 Transceivers nicht angesprochen werden. In diesem Verfahren würde ein eigenes Carrier Board entworfen und dabei insbesondere darauf geachtet werden, dass genügend UARTs vorhanden wären, die direkt angesprochen werden können. Indem unnötige Features nicht verbaut werden, könnte mit dieser Lösung Gewicht gespart und die Form an die Drohne angepasst werden.

Eine andere Möglichkeit wäre das Kontaktieren einer Firma, die sich auf den Bau von Drohnen spezialisiert hat. Gemeinsam ließe sich eine Drohne entwickeln, die die gewünschten Anforderungen erfüllt.

Zuletzt ist es auch möglich, eine vorgefertigte Drohne zu kaufen. Aufgrund der mangelhaften Zugänglichkeiten auf die Hard- und Software war diese Lösung bis zum Ende dieser Arbeit nicht im Gespräch. Es hat sich allerdings herausgestellt, dass es Firmen gibt, die trotz Marktwettbewerbs den Zugang auf die einzelnen Komponenten gestatten.

Nach intensiver Recherche und ausgiebiger Diskussion wird aus Gründen der Einfachheit sowie der Zeitersparnis die dritte Lösung umgesetzt. Es wird eine „DJI Mavic 3E“ der Firma DJI gekauft und an die Anforderungen der Drohne für das Hochregallager angepasst. Die im Rahmen der Arbeit entwickelte Drohne wird mit dem bestehenden Sensorkonzept nichtsdestotrotz weiterentwickelt, auch wenn sie für die eigentliche Zielstellung eine eher untergeordnete Rolle spielen wird. Grund dafür ist, dass die Arbeit mit ihr wichtige Erfahrungen für das Forschungsprojekt 5G-POUST ermöglicht und trotz kurzer Flugzeit aufgezeigt werden kann, dass es möglich ist, eine autonom fliegende Drohne zu entwickeln, die die Inventur in einem Hochregallager vornimmt.

4 Schluss

Die Masterarbeit befasste sich ausführlich mit der Entwicklung einer selbstfliegenden Drohne, die im Hochregallager von Schaeffler zur Bestandsaufnahme genutzt werden soll. Unter Einbezug der theoretischen Grundlagen, wurde ein erster Prototyp gebaut. Allerdings war der dort verbaute Radarsensor, der im Hochregallager getestet wurde, nicht verlässlich. Diese und weitere Erkenntnisse führten zur Umsetzung eines zweiten Aufbaus, der mit einem erweiterten und verbesserten Sensorkonzept ausgestattet war. Obwohl das Sensorkonzept deutlich verbessert wurde, stellte sich heraus, dass die Flugdauer der Drohne nicht ausreicht, um eine effiziente Inventur im Hochregallager von Schaeffler vornehmen zu können. Die Ursache dafür ist die falsche Dimensionierung der einzelnen Drohnenkomponenten und das zu hohe Gewicht. Auf Dauer sind verschiedene negative Auswirkungen zu erwarten, die die Drohne zu einem kurzlebigen und unsicheren Helfer im Hochregallager macht.

Auch wenn das Endziel der Entwicklung einer autonom fliegenden Drohne für die Inventur im Hochregallager, nicht mit der selbstgebauten Drohne an dieser Stelle realisiert werden kann, leistet diese Arbeit trotzdem einen wertvollen Beitrag zur Automatisierung der Lagerlogistik. Aus den Ergebnissen des finalen Aufbaus geht hervor, dass es durchaus möglich ist, eine autonom fliegende Drohne zu bauen, die über 5G betrieben wird. Mit jeder Verbesserung und Verschlechterung der Drohne wurden wertvolle Erfahrungen gesammelt, die anderen Menschen dabei helfen können, in der Drohnenentwicklung voranzukommen und die zur Anpassung der neu gekauften Drohne von enormen Vorteil sind. Da die Entwicklung der hier gebauten Drohne noch nicht abgeschlossen ist, bleibt es weiterhin spannend, welche Erkenntnisse sie noch bringen wird. Insbesondere das im Rahmen dieser Arbeit entwickelte Sensorkonzept kann als Basis für weitere Forschung zur praktischen Anwendung autonom fliegender Drohnen im Hochregallager genutzt werden.

Anhang

I Abbildungsverzeichnis

Abbildung 1: FMCW-Signal (übernommen von: Wolff 2018) 11
Abbildung 2: "docker-compose.yml“-File 17
Abbildung 3: Maße des Hochregallagers 21
Abbildung 4: Datenblatt eines T-Motors (übernommen von: Premium-Modellbau o.A.) 25
Abbildung 5: 7-Zoll-Rahmen „TBS Source One V5 5" FPV Frame“ (übernommen von: Copterfarm o.A.) 28
Abbildung 6: Drohnenmotor „FETtec FT2207 1100 KV 10S“ (übernommen von: FETTEC UG o.A.) 29
Abbildung 7: ESC-FC-Kombination „SpeedyBee F405 V3 BLS 50A 30x30 FC&ESC Stack“ (übernommen von: Zhuhai KuaiFeng Technology Co. o.A.) 30
Abbildung 8: TDoA-Prinzip (übernommen von Alarifi et al. 2016) 32
Abbildung 9: FMCW-Radar „OPS241-B Short Range FMCW Radar Sensor“ (übernommen von: OmniPreSense o.A.) 36
Abbildung 10: Programm zum Auslesen der Daten des FMCW-Radars 37
Abbildung 11: Tiefenkamera „Intel® RealSense™ Depth Camera D435i“ von vorne (übernommen von: Intel Corporation o.A.) 38
Abbildung 12: Tiefenkamera „Intel® RealSense™ Depth Camera D435i“ von hinten (übernommen von: BOTLAND B. DERKACZ SP. K. o.A.) 38
Abbildung 13: Tiefenbild (übernommen von: Microsoft Corporation o.A.) 39
Abbildung 14: Programm zum Auslesen der Daten der Tiefenkamera 39
Abbildung 15: DTOF LIDAR LD19 (übernommen von: Waveshare o.A.) 41
Abbildung 16: Roher Datenstrom des LIDARs „DTOF LIDAR LD19“ (übernommen von: Shenzhen youyeetoo Tech o.A.) 42
Abbildung 17: Schema des LIDARs von oben (übernommen von: Shenzhen youyeetoo Tech o.A.) 43
Abbildung 18: LIDAR-Programm: Importieren der Bibliotheken und Definieren von Variablen 44
Abbildung 19: LIDAR Programm: serielle Verbindung 44
Abbildung 20: LIDAR-Programm: Beginn der While-Schleife und Einlesen der Daten 45
Abbildung 21: LIDAR-Programm: Ende des Einlesens der Daten 46
Abbildung 22: LIDAR-Programm: Umwandlung der Daten von Hex in Integer 46
Abbildung 23: LIDAR-Programm: Beginn der Erstellung von Arrays für die Datenpunkte 1-3 47
Abbildung 24: LIDAR-Programm: Beendigung der Erstellung von Arrays für die Datenpunkte 7-12 47
Abbildung 25: LIDAR-Programm: Ausgabe der Daten 49
Abbildung 26: Entwurf eines ersten Sensorkonzepts 50
Abbildung 27: Halterung der Tiefenkamera von oben 51

Abbildung 28: Halterung der Tiefenkamera von der Seite 52
Abbildung 29: Halterung für den Raspberry Pi und die Radarsensoren von der Seite vorne..... 53
Abbildung 30: Halterung für den Raspberry Pi und die Radarsensoren von der Seite hinten..... 53
Abbildung 31: Drohnen-Fuß 54
Abbildung 32: Erster Aufbau der Drohne 55
Abbildung 33: Anpassung von Drohne und Sensorkonzept von oben 58
Abbildung 34: Anpassung von Drohne und Sensorkonzept von der Seite..... 58
Abbildung 35: Anpassung der Halterungen der Tiefenkameras..... 60
Abbildung 36: Gesamte LIDAR-Halterung..... 61
Abbildung 37: Zerlegte LIDAR-Halterung 62
Abbildung 38: Zusatz zur LIDAR-Halterung..... 62
Abbildung 39: Toradex-Halterung von der Seite 63
Abbildung 40: Toradex-Halter von hinten oben..... 64
Abbildung 41: Toradex-Board „Apalis iMX8“ (übernommen von: Toradex Group AG o.A.a)..... 64
Abbildung 42: Carrier Board „Ixora Carrier Board V1.2“ (übernommen von: Toradex Group AG o.A.c) 65
Abbildung 43: Anschlüsse des Carrier Boards „Ixora Carrier Board“ (übernommen von: Toradex Group AG o.A.c)..... 66
Abbildung 44: Aufbau einer Socket-Verbindung in LIDAR 67
Abbildung 45: Versenden von Daten in LIDAR über die Socket-Verbindung..... 67
Abbildung 46: Docker-File der LIDAR-Sensoren 68
Abbildung 47: Anpassung des Programms zum Auslesen der Daten der Tiefenkameras..... 69
Abbildung 48: Dockerfile für die Tiefenkameras 70
Abbildung 49: Erster Teil des Programms zur Datenverarbeitung im Container..... 71
Abbildung 50: Erster Teil des Programms zur Datenverarbeitung im Container..... 72
Abbildung 51: Erster Teil des Docker Compose Files..... 74
Abbildung 52: Zweiter Teil des Docker Compose Files..... 75
Abbildung 53: Finaler Aufbau der Drohne..... 76

II Literaturverzeichnis

Alarifi, Abdulrahman et al. (2016): *Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances*. In: Sensors, 16(5). URL: https://www.mdpi.com/1424-8220/16/5/707 [zuletzt abgerufen am: 13.4.2024].

Baruschke, Peter (2021): *Bürstenloser Motor Vorteile & Nachteile. Wie funktioniert ein Brushless-Akkuschrauber?*. URL: https://www.selbst.de/brushless-motor-55785.html [zuletzt abgerufen am: 07.06.2024].

Baumgärtel, Anne/Landrock, Holm (2018): *Die Industriedrohne – der fliegende Roboter.* Wiesbaden: Springer Vieweg.

BETAFPV (o.A.): *ELRS Lite Receiver.* URL: https://betafpv.com/products/elrs-lite-receiver [zuletzt abgerufen am: 18.04.2024].

BOTLAND B. DERKACZ SP. K. (o.A.): *Intel RealSense Tiefenkamera D435i – stereoskopische Tiefenkamera.* URL: https://botland.de/eingestellte-produkte/16697-intel-realsense-tiefenkamera-d435i-stereoskopische-tiefenkamera-5032037148399.html [zuletzt abgerufen am: 27.5.2024].

Choudhary, Ashish (2023): *Understanding Docker Networking*. URL: https://earthly.dev/blog/docker-networking/ [zuletzt abgerufen am 14.06.2024].

ClearView Imaging GmbH (o.A.): *Stereo Vision für 3D-ildverarbeitungsanwendungen*. URL: https://www.clearview-imaging.com/de/blog/stereo-vision-f%C3%BCr-3d-bildverarbeitungsanwendungen#:~:text=Eine%20Stereokamera%20kopiert%20genau%2C%20wie,beiden%202D%2DEbenen%20zu%20triangulieren [zuletzt abgerufen am: 14.04.2024].

Copterfarm (o.A.): *TBS Source One HD 7 Zoll Arme Upgrade Set.* URL: https://www.copterfarm.de/tbs-source-one-hd-7-zoll-arme-upgrade-set-1292.html [zuletzt abgerufen am: 18.04.2024].

DJI (o.A.): *DJI Mavic 3E*. URL: https://store.dji.com/product/dji-mavic-3e-and-dji-care-enterprise-basic?site=brandsite&from=landing_page&vid=120111 [zuletzt abgerufen am 25.06.2024].

Docker Inc. (o.A.): *Dockerfile reference*. URL: https://docs.docker.com/reference/dockerfile/#cmd [zuletzt abgerufen am: 14.06.2024].

Donner, Andreas/ Luber, Stefan (2019): *Definition. Was ist Network Slicing?*. URL: https://www.ip-insider.de/was-ist-network-slicing-a-828834/ [zuletzt abgerufen am 29.06.2024].

Eidloth, Andreas/ Kasparek, Maximilian/ Seitz, Jochen (2022): *5G-Lokalisierung mit Uplink-TDOA (Time Difference of Arrival) im industriellen Umfeld*. URL: https://www.iis.fraunhofer.de/de/magazin/bereiche/lokalisierung-und-vernetzung/2022/5g-connect-forschende/5G-lokalisierung-mit-uplink-tdoa.html [zuletzt abgerufen am: 21.06.2024].

exceeding solutions GmbH (o.A.a): *5G-POUST. Präzise Organisieren Und Smarte Telemetrie.* URL: https://www.5g-poust.de/ [zuletzt abgerufen am 30.06.2024]

exceeding solutions GmbH (o.A.b): *Unternehmen. Portrait.* URL: https://www.exceeding-solutions.de/unternehmen/portrait [zuletzt abgerufen am 10.06.2024].

FETTEC UG (o.A.): *FETtec FT2207 1100 KV 10S*. URL: https://fettec.net/shop/elektronik/motoren/fettec-ft2207-1100-kv-10s [zuletzt abgerufen am: 18.05.2024].

inray Industriesoftware GmbH (o.A.): *Was ist Docker und was sind die Vorteile?*. URL: https://www.opc-router.de/was-ist-docker/ [zuletzt abgerufen am: 29.06.2024].

Intel Corporation (2024): *Intel® RealSense™ Product Family D400 Series. Datasheet.* URL: https://dev.intelrealsense.com/docs/intel-realsense-d400-series-product-family-datasheet [zuletzt abgerufen am: 05.05.2024].

Intel Corporation (o.A.): *Depth Camera D435i.* URL: https://www.intelrealsense.com/depth-camera-d435i/ [zuletzt abgerufen am: 05.05.2024].

Jennings, Nathan (o.A.): *Socket Programming in Python (Guide).* URL: https://realpython.com/python-sockets/ [zuletzt abgerufen am: 19.06.2024].

Kamwa, Alain Bruno (2022): *ToF-Lidar: 3D-Erfassung wie im Flug. Time-of-Flight-(ToF) -Verfahren mit Lidar-Systemen*. URL: https://www.all-electronics.de/elektronik-entwicklung/time-of-flight-und-40tof-und-41-verfahren-mit-lidar-systemen-632.html#:~:text=Das%20grundlegende%20Prinzip%20ist%20einfach,gemessenen%20Laufzeit%20des%20Lichts%20errechnet [zuletzt abgerufen am 13.05.2024].

Large, Martin (2024): *Einsatzmöglichkeiten nicht nur im Auto. Lidar: Was es ist, wie es funktioniert und was es kann*. URL: https://www.all-electronics.de/automotive-transportation/lidar-sensoren-automotive-575.html [zuletzt abgerufen am: 06.06.2024].

meilon GmbH (o.A.a): *Gemfan 7042 7x4.2 Flash 2 Blatt Propeller Schwarz 7 Zoll.* URL: https://www.fpv24.com/de/gemfan/gemfan-7042-7x42-flash-2-blatt-propeller-schwarz [zuletzt abgerufen am: 25.04.2024].

meilon GmbH (o.A.b): *Tattu R-Line V3 Batterie LiPo Akku 1550 mAh 4S1P 120C XT60*. URL: https://www.fpv24.com/de/tattu/tattu-r-line-v3-batterie-lipo-akku-1550-mah-4s1p-120c-xt60 [zuletzt abgerufen am 28.04.2024].

meilon GmbH (o.A.c): *TBS Source One 5 Zoll V5 FPV Frame*. URL: https://www.fpv24.com/de/team-blacksheep/tbs-source-one-v4-5-zoll-fpv-frame [zuletzt abgerufen am 18.04.2024].

Metrilus GmbH (o.A.): *Time-of-Flight-Kameras.* URL: https://de.metrilus.de/technology/time-of-flight [zuletzt abgerufen am: 05.05.2024].

Microsoft Corporation (o.A.): *Verwenden von Bildtransformationen des Azure Kinect Sensor SDK.* URL: https://learn.microsoft.com/de-de/azure/kinect-dk/use-image-transformation [zuletzt abgerufen am: 16.05.2024].

OmniPreSense (o.A.): *OPS241-B FMCW Radar Sensor.* URL: https://omnipresense.com/product/ops241-b-short-range-fmcw-radar-sensor/ [zuletzt abgerufen am: 10.05.2024].

passpunkt.de (o.A.): *Was ist LiDAR?. Definition und Anwendungsbereiche von LiDAR.* URL: https://www.passpunkt.de/was-ist-lidar [zuletzt abgerufen am: 13.05.2024].

Premium-Modellbau (2020): *Drohnen selber bauen - Der Guide.* URL: https://www.premium-modellbau.de/drohnen-selber-bauen-der-guide [zuletzt abgerufen am: 20.04.2024].

Premium-Modellbau (o.A.a): *FrSky Taranis X9D Plus 2019 2,4GHz*. URL: https://www.premium-modellbau.de/frsky-taranis-x9d-plus-2019-2-4ghz-accst-access-silber [zuletzt abgerufen am 18.04.2024].

Premium-Modellbau (o.A.b): *T-Motor F60 Pro V FPV Race Brushless Motor 1750kv V5 in Grau*. URL: https://www.premium-modellbau.de/t-motor-f60-pro-v-fpv-race-brushless-motor-1750kv-v5-in-grau [zuletzt abgerufen am: 21.04.2024].

Raspberry Pi (o.A.): *Raspberry Pi 4B.* URL: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ [zuletzt abgerufen am: 15.05.2024]

Schaeffler Technologies AG & Co. KG (o.A.): *Schaeffler in Deutschland.* URL: https://www.schaeffler.de/de/schaeffler-deutschland/ [zuletzt abgerufen am: 25.06.2024].

Shenzhen youyeetoo Tech (o.A.): *FHL-LD19P Lidar.* URL: https://wiki.youyeetoo.com/en/Lidar/D300 [zuletzt abgerufen am: 20.05.2024].

Telekom Deutschland GmbH (o.A.): *Was ist 5G?*. URL: https://www.telekom.de/unterwegs/was-ist-5g [zuletzt abgerufen am 29.06.2024].

Toradex Group AG (o.A.a): *Apalis iMX8*. URL: https://developer.toradex.com/hardware/apalis-som-family/modules/apalis-imx8/ [zuletzt abgerufen am: 30.06.2024].

Toradex Group AG (o.A.b): *Ixora Carrier Board V1.2.* URL: https://developer.toradex.com/hardware/apalis-som-family/carrier-boards/ixora-carrier-board/ [zuletzt abgerufen am: 30.06.2024].

Toradex Group AG (o.A.c): *Torizon OS. Einfach zu bedienende industrielle Linux-Software-Plattform.* URL: https://www.toradex.com/de/operating-systems/torizon-core [zuletzt abgerufen am: 30.06.2024].

Waveshare (o.A.): *DTOF LIDAR LD19*. URL: https://www.waveshare.com/wiki/DTOF_LIDAR_LD19 [zuletzt abgerufen am: 13.05.2024].

Wolff, Christian (2018): *Frequenzmoduliertes Dauerstrichradar (FMCW Radar)*. URL: https://www.radartutorial.eu/02.basics/Frequenzmodulierte%20Dauerstrichradarger%C3%A4te.de.html [zuletzt abgerufen am: 10.05.2024].

Zauner, Christoph (2023): *LiDAR.* URL: https://www.md-elektronik.com/de/lidar/ [zuletzt abgerufen am: 15.05.2024].

Zhuhai KuaiFeng Technology Co. (o.A.): *SpeedyBee F405 V3 BLS 50A 30x30 FC&ESC Stack.* URL: https://www.speedybee.com/speedybee-f405-v3-bls-50a-30x30-fc-esc-stack/ [zuletzt abgerufen am: 28.04.2024].

III Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle sinngemäßen und wörtlich übernommenen Textstellen wurden kenntlich gemacht.

Halle (Saale), 02.07.2024 Edgar Lumpe