Diplomarbeit

5AHET 2019/20

Höhere Technische Bundeslehr- und Versuchsanstalt Salzburg

Abteilung für Elektrotechnik

Entwicklung eines mobilen, gestengesteuerten Robotergreifsystems mit haptischem Feedback

Höhere Technische Bundeslehr- und
Versuchsanstalt Salzburg

Itzlinger Hauptstraße 30

5020 Salzburg

www.htl-salzburg.ac.at

Eidesstattliche Erklärung

Wir erklären an Eides statt, dass wir die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht haben. Wir versichern, dass wir dieses Diplomarbeitsthema bisher weder im In- noch im Ausland (einer Beurteilerin oder einem Beurteiler) in irgendeiner Form als Prüfungsarbeit vorgelegt haben.

Jakob Buchsteiner

Thomas Eibl

Sebastian Neuhofer

Moritz Taferner

Ort, Datum

Vorwort

Im Zuge der Ausbildung an der HTBLuVA Salzburg wird durch die Diplomarbeit ein erster umfangreicher Einblick in wissenschaftliches und selbstständiges Arbeiten gegeben. Um die Diplomarbeit in einem derartigen Ausmaß zu verwirklichen, war die Unterstützung einiger Personen allerdings unabdingbar.

Primär gilt unser Dank unserem Hauptbetreuer Roland Holzer, welcher nicht nur durch fachliche Kompetenz, sondern auch durch seinen engen Kontakt zum Unternehmen Bernecker&Rainer essenzielle Unterstützung liefern konnte.

Weiters bedanken wir uns bei unserem zweiten Betreuer Peter Lindmoser, welcher ebenfalls eine tragende Rolle in der Umsetzung des Projektes und der Diplomarbeit einnahm.

Außerdem bedanken wir uns bei den Fachlehrern Gotthard Fränzl, Albert Grazer, Franz Rieser und Stephan Häuserer für deren Unterstützung bei der mechanischen Fertigung sowie bei Anton Haslauer für Bemühungen im Bereich des Sponserings.

Abschließend gilt unser Dank den Unternehmen B&R Industrial Automation GmbH und Igus GmbH, welche benötigte Hardware vollständig oder vergünstigt zur Verfügung stellten. Insbesondere Michael Eckschlager und Johannes Rauscher – unseren Kontaktpersonen zu B&R Industrial Automation GmbH und Igus GmbH – danken wir für den ständigen Kontakt bezüglich technischen Supports.

Auch Lara Hirnsperger, Jonas Leitner und Franz Eibl trugen im Bereich der Motivation und Ideenfindung eine nicht unwesentliche Rolle, welche an dieser Stelle keinesfalls unerwähnt bleiben sollte.

Diplomarbeit

Dokumentation

Namen der Verfasser	Jakob Buchsteiner, Thomas Eibl, Sebastian Neuhofer, Moritz Taferner
Jahrgang Schuljahr	5AHET 2019/20
Thema der Diplomarbeit	Entwicklung eines mobilen, gestengesteuerten Robotergreifsystems mit haptischem Feedback
Kooperationspartner	B&R Industrial Automation GmbH HTBLuVA Salzburg
Aufgabenstellung	Die Aufgabenstellung bestand darin, ein funktionsfähiges mobiles Robotersystem mit Greifer zu entwickeln. Das System soll mithilfe eines Fernsteuergeräts in der Form eines Handschuhs durch Gesten kontrolliert werden, um eine intuitive Fernsteuerung zu ermöglichen. Dazu soll für den Roboter ein Antriebssystem und ein Akkusystem sowie ein Fernsteuergerät inklusive Sensorik zur Aufnahme der Bewegungsdaten entwickelt werden.
Realisierung	Für die Greifvorrichtung wird ein 5-Achsen-Roboterarm zusammen mit einem bionischen 5-Finger Greifer verwendet. Diese Komponenten befinden sich auf einem Fahrwerk. Zum Antrieb des Fahrwerks werden vier Synchronmotoren verwendet, an deren Motorwellen jeweils ein Rad befestigt ist. Die Aufnahme der Bewegungsdaten der menschlichen Hand erfolgt durch zahlreiche Sensoren, welche am Fernsteuer-Handschuh befestigt ist. Die Systeme kommunizieren untereinander mittels Bluetooth, RS232 und CAN-Bus.
Ergebnisse	Zum Abgabezeitpunkt befindet sich das Robotersystem sowie das Fernsteuergerät in einem fertigen Prototypen-Status. Bewegungsdaten der menschlichen Hand werden korrekt interpretiert und an das Robotersystem gesendet, wo diese dann als Bewegungen des Fahrwerks oder des Greifsystems ausgeführt werden. Dazu kann zwischen den 2 Modi „Greifen“ und „Fahren“ gewählt werden. Die wesentlichen angestrebten Funktionen des haptischen Feedbacks sind einsatzbereit – einzig die Ausführung des kinästhetischen Feedbacks befindet sich in der Entwicklungsphase.

Foto (mit Erläuterung)	 3D-Modell des gesamten Robotersystems und des Fernsteuergeräts

Möglichkeit der Einsichtnahme in die Arbeit	Die Diplomarbeit ist in gebundener Form sowohl in der Schulbibliothek als auch bei AV Prof. Dipl-Ing. (FH) Roland Holzer einzusehen. Darüber hinaus besitzt jedes Mitglied des Projektteams eine vollständige Version in gebundener und digitaler Form.

Approbation (Datum/Unterschrift)	Prüfer/Prüferin	Abteilungsvorstand

Diploma Thesis

Documentation

Author(s)	Jakob Buchsteiner, Thomas Eibl, Sebastian Neuhofer, Moritz Taferner
Form Academic year	5AHET 2019/20
Topic	Development of a mobile, gesture-controlled robot gripping system with haptic feedback
Co-operation Partners	B&R Industrial Automation GmbH HTBLuVA Salzburg
Assignment of Tasks	The task was to develop a functional mobile robot with a gripper. The system is to be controlled by gestures using a remote control device in the form of a glove, to enable intuitive remote control. For this purpose, a drive system and a battery system for the robot, as well as a remote control device including sensor technology for recording the movement data are to be developed.
Realization	A 5-axis robot arm together with a bionic 5-finger gripper is used for the gripping device. These components are located on a mobile chassis, which is powered by four synchronous motors, each driving one wheel. The motion data of the human hand is recorded by numerous sensors, which are located inside the remote control glove. The systems communicate via Bluetooth, RS232 and CAN-Bus.
Result	At the time of submission, the robot system and the remote control unit are in a finished prototype state. Motion data from the human hand is correctly interpreted and sent to the robot system, where it is executed as movements of the chassis or gripping system. For this purpose, it is possible to choose between the two modes „gripping“ and „driving“. The main intended functions of the haptic feedback are ready for use – only the execution of the kinesthetic feedback is still in the development phase.

Illustrative Graph, Photo (with explaination)	 3D-model of the entire robot and the remote control unit
Accessibility of Diploma Thesis	The diploma thesis is available in the school library as well as in the office of the head of department Prof. Dipl-Ing. (FH) Roland Holzer. In addition, each member of the project team has a complete version in hardcover and digital form.

Approval (Date/Sign)	Examiner	Head of department

Inhaltsverzeichnis

I Einführung **1**
1 Projektteam . . . 1
2 Projektbetreuer . . . 2
3 Aufgabenteilung . . . 2

II Einleitung **5**
1 Motivation . . . 5
2 Zielsetzung . . . 6
3 Topologie des Gesamtsystems . . . 7
3.1 Gesamtansicht . . . 7
3.2 Projekt-Teilbereiche . . . 8
4 Funktionsbeschreibung . . . 9
4.1 Aufladen der Systeme . . . 9
4.2 Starten der Systeme . . . 10
4.3 Benutzeroberfläche des Fernsteuergeräts . . . 10
4.4 Referenzieren/Kalibrieren der Systeme . . . 11
4.5 Bedienmodi . . . 12
4.6 Abschalten der Systeme . . . 12
5 Leitfaden . . . 13

III Stand der Technik **15**
1 Synchronmaschinen . . . 15
1.1 Allgemeines . . . 15
1.2 Synchronmaschinen mit Dauermagneterregung . . . 16
2 Akkusysteme . . . 18
2.1 Batteriearten . . . 18
2.2 Batteriemanagementsystem . . . 20
2.3 Einsatzfelder für Lithium-Ionen-Batterien . . . 22
3 Sensorik . . . 24
3.1 Ultraschallsensoren . . . 25
3.2 Drucksensoren . . . 26
3.3 Biegesensoren . . . 28
3.4 Inertiale Messeinheiten . . . 30
4 Haptisches Feedback in Robotik-Systemen . . . 32
4.1 Definition von Haptik . . . 32
4.2 Bereitstellung von haptischem Feedback . . . 32
4.3 Aktuelle Anwendungsbereiche . . . 38
5 Bussysteme und Kommunikationsschnittstellen . . . 41
5.1 UART und RS232 . . . 41
5.2 Serial Peripheral Interface (SPI) . . . 41
5.3 Inter integrated Circuit . . . 42

5.4 CAN-Bus . . . 43
5.5 Bluetooth Low Energy . . . 44
6 Steuerungshardware . . . 45
6.1 Mikrocontroller . . . 45
6.2 Speicherprogrammierbare Steuerungen . . . 46
7 3D-Druck . . . 47
7.1 Druckverfahren . . . 47

IV Robotersystem 49
1 Übersicht . . . 50
1.1 Anforderungen an das Gehäuses . . . 50
1.2 Aufgaben der SPS . . . 50
1.3 Aufgaben des Fahrwerks . . . 50
1.4 Aufgaben des Robolink . . . 50
1.5 Aufgaben der Abstandssensorik . . . 50
2 Gehäuse . . . 51
2.1 Dimensionierung . . . 51
2.2 Gerüst . . . 51
2.3 Abdeckungen . . . 52
2.4 SPS-Gehäuse . . . 55
3 Speicherprogrammierbare Steuerung . . . 56
3.1 Compact-S CPU . . . 57
3.2 Schrittmotormodule . . . 58
3.3 ACOPOSmicro Wechselrichtermodule . . . 60
3.4 Leitungsschutzschalter . . . 61
3.5 DC-DC-Wandler . . . 61
4 Fahrwerk . . . 62
4.1 Motor . . . 62
4.2 Motorhalter . . . 62
4.3 Mecanum-Wheels . . . 63
5 Robolink . . . 64
5.1 Technische Daten . . . 64
5.2 Abtriebsencoder . . . 65
5.3 Greifer . . . 66
6 Abstandssensorik . . . 67
6.1 Ultraschall-Sensormodul HC-SR04 . . . 67
6.2 Receiver-Platine . . . 69
6.3 Platinengehäuse . . . 74
6.4 Verdrahtung . . . 75

V Energieversorgung 77
1 Übersicht . . . 78
1.1 Aufgaben der Energieversorgung als Gesamtprojekt . . . 78
1.2 Aufgaben des Batteriemanagement . . . 78
1.3 Aufgaben der Elektrischen Installation . . . 78
2 Batteriemanagement . . . 79
2.1 Akkumulatoren . . . 79
2.2 Batteriemanagementsystem . . . 87
2.3 Laderegelung . . . 95
3 Elektrische Installation . . . 99

3.1 Dimensionierung . . . 99
3.2 Versorgung . . . 100
3.3 SPS Verschaltung . . . 102
3.4 ACOPOSmicro Verschaltung . . . 103

VI Fernsteuergerät 105
1 Übersicht . . . 106
1.1 Aufgaben des Fernsteuergerätes . . . 106
1.2 Aufgaben der Mechanik . . . 106
1.3 Aufgaben der Elektronik-Hardware . . . 106
2 Mechanik . . . 107
2.1 Handschuh-Design . . . 107
2.2 Konstruktions-Tool . . . 107
2.3 Baugruppen . . . 107
3 Feedback-Funktionen . . . 113
3.1 Taktiles Feedback . . . 113
3.2 Kinästhetisches Feedback . . . 114
4 Hardware-Design . . . 119
4.1 Spannungsversorgung . . . 119
4.2 Sensorik . . . 119
4.3 Steuerung des taktilen Feedbacks . . . 125
4.4 Steuerung des kinästhetischen Feedbacks . . . 127
4.5 Layout der Platinen . . . 132

VII Software 139
1 Übersicht . . . 140
1.1 Topologie . . . 140
1.2 Aufgaben der Software im Fernsteuergerät . . . 140
1.3 Aufgaben des Embedded Systems am Roboter . . . 140
1.4 Aufgaben der speicherprogrammierbaren Steuerung . . . 141
2 Toolchain . . . 141
2.1 Ordnerstruktur . . . 141
2.2 PlatformIO . . . 142
2.3 Visual Studio Code . . . 142
2.4 Automation Studio . . . 142
3 Entwicklung des Übertragungsprotokolls . . . 143
3.1 Übersicht . . . 143
3.2 Aufbau . . . 143
3.3 Nutzdaten . . . 144
3.4 Verwendete Nachrichten . . . 144
3.5 Verbindungsaufbau und -überwachung . . . 145
3.6 API der Bibliothek . . . 146
4 Embedded System im Fernsteuergerät . . . 148
4.1 Übersicht . . . 148
4.2 Kommunikation . . . 151
4.3 Auswertung des 9-Achsen-Sensors . . . 153
4.4 Auswertung der Biege- und Drucksensoren . . . 155
4.5 Steuerung des haptischen Feedbacks . . . 156
4.6 Benutzeroberfläche . . . 157
5 Embedded System am Roboter (Receiver) . . . 160

5.1 Übersicht . . . 160
5.2 Kommunikation mit dem Fernsteuergerät . . . 161
5.3 Kommunikation mit der SPS . . . 164
5.4 Ansteuerung des Greifers . . . 165
5.5 Auswertung der Motorströme . . . 165
5.6 Auswertung der Ultraschall-Sensoren auf dem ATmega328P . . . 165
5.7 Übertragung der Abstandsdaten über I^2C . . . 166
5.8 Auslesen der der Abstandsdaten am ESP32 . . . 166
6 Speicherprogrammierbare Steuerung . . . 168
6.1 Übersicht . . . 168
6.2 Datenaustausch zwischen den Tasks . . . 169
6.3 Kommunikation mit dem Receiver . . . 170
6.4 Fahrwerk-Steuerung . . . 173
6.5 Steuerung des Roboterarms . . . 178
6.6 Auswertung des Batteriemanagements . . . 184

VIII Ergebnisse und Ausblick **185**

Anhang A Code **187**

Anhang B CAD-Zeichnungen **193**
1 Robotersystem . . . 194
2 Energieversorgung . . . 205
3 Fernsteuergerät . . . 210

Anhang C Schaltpläne und Layouts **221**
1 Robotersystem . . . 222
2 Fernsteuergerät . . . 249

Literaturverzeichnis **255**

Abbildungsverzeichnis **259**

Tabellenverzeichnis **263**

Codeverzeichnis **264**

I Einführung

1 Projektteam

Jakob Buchsteiner

Thomas Eibl

Sebastian Neuhofer

Moritz Taferner

2 Projektbetreuer

AV Dipl.-Ing.(FH) Roland Holzer unterstützte Jakob Buchsteiner bei der Softwareentwicklung sowie Moritz Taferner bei der Auslegung des Energieversorgungs-Systems

Prof. Dipl.-Ing. Peter Lindmoser unterstützte Thomas Eibl und Sebastian Neuhofer bei der Hardwareentwicklung.

3 Aufgabenteilung

Jakob Buchsteiner

- Projektleitung
- Projektfindung und Projektplanung
- Projektaufteilung
- Erstellen der Einreichdokumente
- Entwickeln der Software für Fernsteuergerät und Empfänger
- Entwickeln der SPS-Software für Fahrwerk und Roboterarm
- Vorbereiten der Präsentation
- Verfassen der Dokumentation

Thomas Eibl

- Projektfindung und Projektplanung
- Projektaufteilung
- Erstellen der Einreichdokumente
- Design der Mechanik
- Aufbau der Mechanik
- Entwicklung der Receiver-Platine
- Vorbereiten der Präsentation
- Verfassen der Dokumentation

Sebastian Neuhofer

- Projektfindung und Projektplanung
- Projektaufteilung
- Erstellen der Einreichdokumente
- Entwicklung der Fernsteuer-Mechanik
- Konzeptionierung des Hardwaredesigns
- Planung und Implementierung der Feedback-Funktionen
- Vorbereiten der Präsentation
- Verfassen der Dokumentation

Moritz Taferner

- Projektfindung und Projektplanung
- Projektaufteilung
- Erstellen der Einreichdokumente
- Planung und Umsetzung der elektrischen Installation
- Entwicklung eines Akkusystems
- Vorbereiten der Präsentation
- Verfassen der Dokumentation

II Einleitung

1 Motivation

Mit dem Schuljahr 2020/21 startet in der Elektrotechnik-Abteilung der HTBLuVA Salzburg unter anderem der neue Schwerpunkt „Autonome Robotik“. Um zukünftige Schülerinnen und Schüler für die vielfältige Ausbildung im Bereich Robotik zu begeistern, ergab sich die Idee, ein Projekt zum Schwerpunkt zu realisieren.

Robotik ist ein breiter Bereich und vereint Elemente aus Mechanik, Elektrotechnik und Informatik, daher soll auch das Projekt möglichst viele Aspekte der Robotik veranschaulichen. Es soll aber nicht nur ein „Spielzeug“ oder Ausstellungsobjekt sein, sondern einen Prototyp für eine reale Anwendung darstellen, wobei wir uns für die sogenannte „Teleoperation“ entschieden.

Es gibt eine große Anzahl an Personen, die in der Ausübung ihres Berufes manuelle Tätigkeiten durchführen müssen, bei denen sie nicht vor Ort sein können, ohne ihre Gesundheit zu gefährden. Das schließt zum Beispiel Ärzte in Zeiten von Epidemien oder Pandemien ein, aber auch Forstarbeiter, Bombenentschärfungspersonal und andere Personen, die gefährliche Stoffe handhaben. In diesen Fällen ist ein möglichst mobiles System wichtig. Daher haben wir uns zum Ziel gesetzt, ein System zu entwickeln, das Teleoperation ermöglicht.

Wenngleich auch im Rahmen der Diplomarbeit die Entwicklung eines in der Realität einsetzbaren Systems – aus zeitlichen sowie budgetären Gründen – unrealistisch ist, soll zumindest ein Prototyp veranschaulichen, wie ein derartiges System verwirklicht werden kann.

2 Zielsetzung

Ziel des Projektes war, ein funktionsfähiges mobiles Robotersystem mit Greifer zu entwickeln. Das System soll mithilfe eines Fernsteuergeräts in Form eines Handschuhs durch Gesten gesteuert werden, um eine intuitive Fernsteuerung zu ermöglichen. Dazu soll für den Roboter ein Antriebssystem und ein Akkusystem sowie ein Fernsteuergerät inklusive Sensorik zur Aufnahme der Bewegungsdaten entwickelt werden.

Um eine möglichst benutzerfreundliche Steuerung zur ermöglichen, sollen die tatsächlichen Bewegungen der Hand in Echtzeit vom Roboterarm imitiert werden – heißt, dass zum Beispiel eine Bewegung der Hand nach oben oder nach vorne auch genau so vom Roboter ausgeführt wird. Am vorderen Ende des Roboterarms soll ein Greifer in Form einer menschlichen Hand (bionischer Greifer) befestigt werden, dessen fünf Finger unabhängig voneinander über Bewegungen der eigenen Finger gesteuert werden sollen.

Weiters soll dem Benutzer ein haptisches Feedback gegeben werden, sobald der Greifer ein Objekt umschließt, wodurch ihm vermittelt wird, dass er seine Hand nicht mehr weiter schließen muss bzw. soll.

3 Topologie des Gesamtsystems

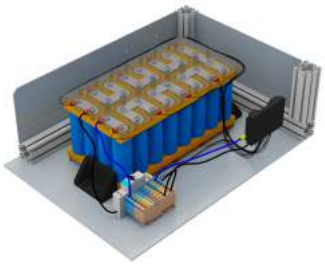
3.1 Gesamtansicht

Im folgenden Abschnitt wird eine kurze Übersicht über die Teilkomponenten des Projekts gegeben, sowie das Zusammenspiel zwischen den Teilbereichen umrissen.

3.2 Projekt-Teilbereiche

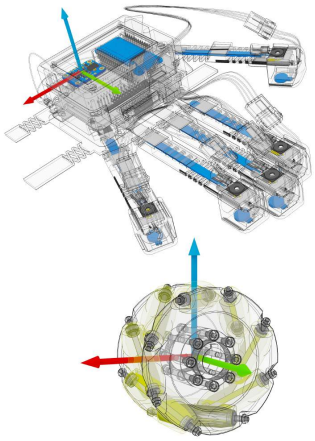
Das Gesamtprojekt gliedert sich inhaltlich in 4 Teilbereiche:

Robotersystem



Energieversorgung

Fernsteuergerät



Software

Im Folgenden wird ein kurzer topologischer Überblick über die Zusammenhänge der 4 Teilbereiche gegeben.

Wie in Abbildung II.1 ersichtlich, befinden sich auf dem Robotersystem zwei Steuerungssysteme. Zum einen die SPS (Speicherprogrammierbare Steuerung), zum anderen der Receiver (engl. Empfänger), der durch einen Mikrocontroller realisiert wurde. Diese beiden Systeme kommunizieren über eine RS232 Verbindung.

Die SPS ist dafür zuständig, den Roboterarm und das Fahrwerk anzusteuern, sowie die Daten des Batteriemanagements über den CAN-Bus auszuwerten.

Der Empfänger ist – wie der Name nahelegt – zuständig für das Empfangen der Befehle und Bewegungsdaten des Fernsteuergeräts über Bluetooth Low Energy, die er dann an die SPS weitergibt. Außerdem steuert er direkt den mechanischen Greifer an und wertet die rund um den Roboter platzierten Ultraschallsensoren aus.

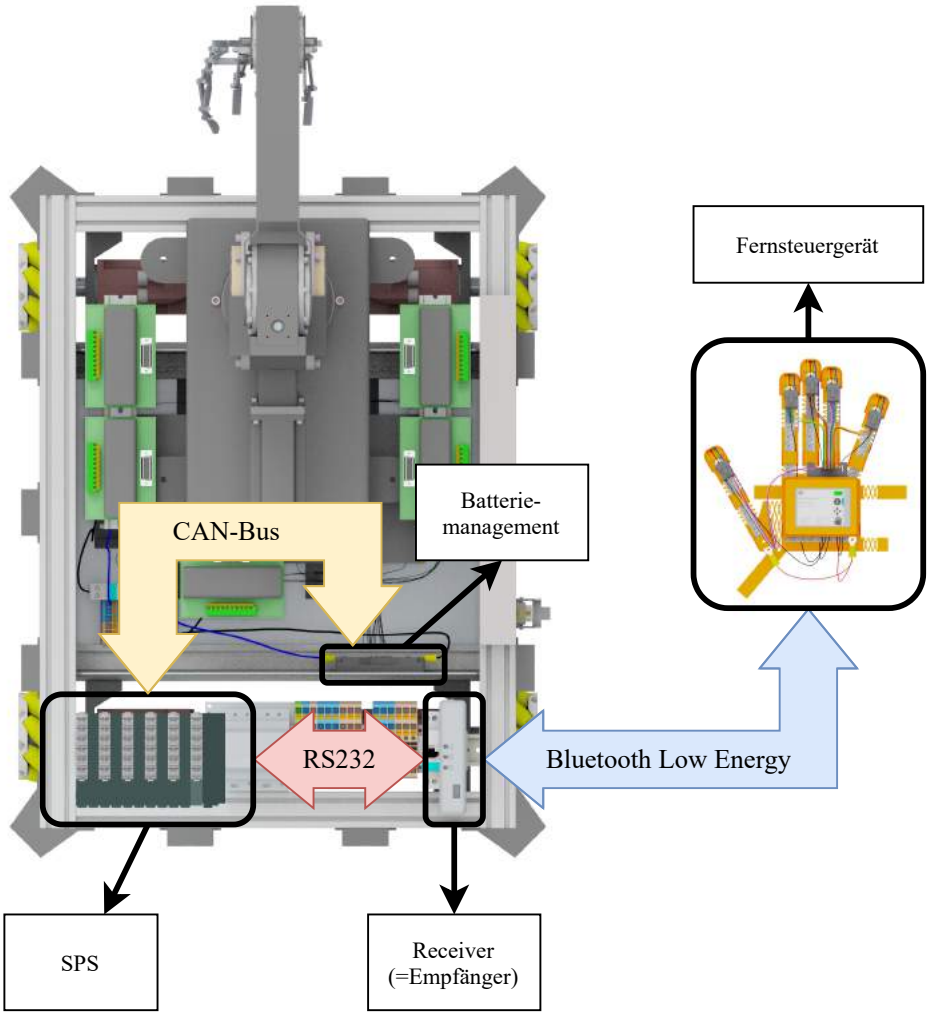


Abbildung II.1: Zusammenspiel der Teilkomponenten

4 Funktionsbeschreibung

Im Folgenden wird eine detaillierte Funktionsbeschreibung gegeben, welche zur Inbetriebnahme des Robotersystems und dessen Benutzung unabdingbar ist. Außerdem soll sie dabei helfen, sich vorab einen Überblick über die verwirklichten Funktionen zu verschaffen, deren Umsetzung in den nachfolgenden Kapiteln beschrieben wird.

4.1 Aufladen der Systeme

Roboterystem Der Akku wird mithilfe eines CAN-Chargers (Kapitel *Energieversorgung* Abschnitt 2.3) aufgeladen. Zuerst ist dieser per D-Sub und dem zweipoligen Ladekabel mit dem Robotersystem zu verbinden, danach muss der CAN-Charger an eine herkömmliche 230 V Steckdose angesteckt werden. Das Ladegerät erkennt anschließend den Akku und beginnt je nach Ladungszustand mit dem Ladevorgang.

Der genaue Zustand des Akkus kann über die App „EMUS BMS Mini“ des Herstellers ermittelt werden. Dazu muss sich das Smartphone nur über Bluetooth mit dem Batteriemanagement verbinden.

Fernsteuergerät Das Fernsteuergerät wird über ein Micro-USB-Kabel direkt an der Elektronik-Box (siehe *Fernsteuergerät* Abschnitt 2.3.2) aufgeladen. Dieses wird einfach in die dafür vorgesehene Aussparung an der Vorderseite der Box angesteckt. Wichtig ist, dass während dem Aufladevorgang das Fernsteuergerät **eingeschalten sein muss**, da der Schiebeschalter die Versorgungsleitungen des Akkus all polig trennt. Der Schaltzustand „ON“ wird durch ein Schieben des Schalters zur Seite des Daumens erreicht.

4.2 Starten der Systeme

Fernsteuergerät Zum Starten des Fernsteuergeräts muss lediglich der Schiebeschalter in den Zustand „ON“ verschoben werden (siehe Beschriftung oder Abschnitt 4.1). Zur Befestigung des Fernsteuer-Handschuhs werden die Klettverschluss-Schlaufen verwendet, welche im Kapitel *Fernsteuergerät* Abschnitt 2.3.1 beschrieben werden.

Roboterystem Das Robotersystem kann einfach durch die Hauptsicherung eingeschaltet werden (siehe *Energieversorgung* Abschnitt 3.2). Es startet danach vollautomatisch und verbindet sich automatisch mit dem Fernsteuergerät.

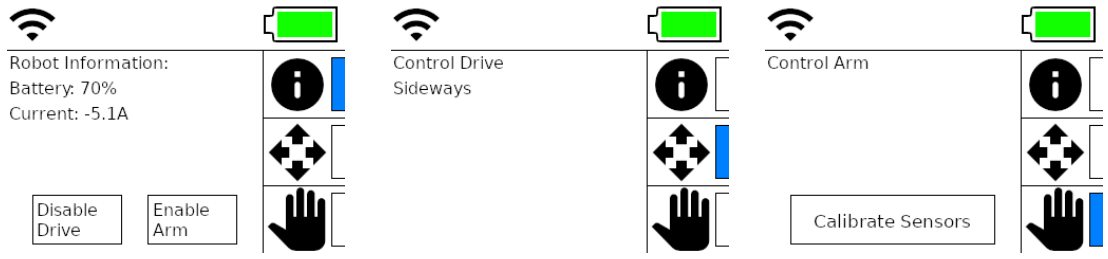
4.3 Benutzeroberfläche des Fernsteuergeräts

Wie in Abbildung II.2 zu erkennen, gibt es drei Schaltflächen, mit denen verschiedenen Bedienmodi ausgewählt werden können. Der aktuelle ausgewählte Modus wird mit einem blauen Balken angezeigt.

Durch Auswählen der oberen Schaltfläche (Info-Symbol), können Informationen zum Robotersystem angezeigt werden. Außerdem kann dort das Robotersystem in Betrieb genommen werden.

Wenn die mittlere Schaltfläche (Navigationskreuz) gewählt wird, kann durch die Bewegung der Hand das Fahrwerk des Roboters gesteuert werden (Bedienmodus „Fahren“).

Die untere Schaltfläche (Hand) wählt den Bedienmodus „Greifen“ aus. Hier kann durch Bewegen der Hand der Roboterarm bewegt werden. Durch Öffnen und Schließen der eigenen Finger können die Finger des Greifers gesteuert werden.



(a) Info-Tab (b) Fahren-Tab (c) Greifen-Tab

Abbildung II.2: Benutzeroberfläche des Fernsteuergeräts

In der Leiste oben werden zwei Statusinformationen angezeigt. Das Batteriesymbol in der rechten oberen Ecke zeigt den aktuellen Akkustand des Fernsteuergerätes. In der linken oberen Ecke wird der aktuelle Verbindungsstatus angezeigt. Abbildung II.3 zeigt die möglichen Symbole und deren Bedeutung.

(a) Keine Verbindung

(b) Verbindung zum Receiver, aber keine Verbindung zum Roboter

(c) Verbindung aufrecht

Abbildung II.3: Mögliche Zustände der Verbindungs-Statusanzeige

4.4 Referenzieren/Kalibrieren der Systeme

Die Referenzierung bzw. Kalibrierung beider Systeme – Robotersystem und Fernsteuergerät – erfolgt direkt über das Touch-Display, welches sich am Fernsteuergerät befindet.

Dazu wird, wie in Abbildung II.2 ersichtlich, bei den drei Menüpunkten das Info-Symbol ausgewählt (blauer Balken symbolisiert die Auswahl).

Robotersystem Ist es dem Roboterarm nicht möglich eine automatische Referenzierung durchzuführen (siehe *Software* Abschnitt 6.5.3), erscheint auf dem Fernsteuergerät ein Dialog, in dem die Achsen manuelle referenziert werden können (siehe Abbildung II.4). Dabei wird für jede Achse die Abfrage, ob der Referenzschalter in positiver oder in negativer Richtung gesucht werden soll. Wurden die Referenzschalter erfolgreich gefunden, bewegt sich der Roboterarm auf die Ausgangsstellung.

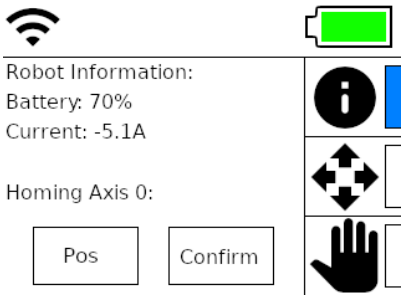
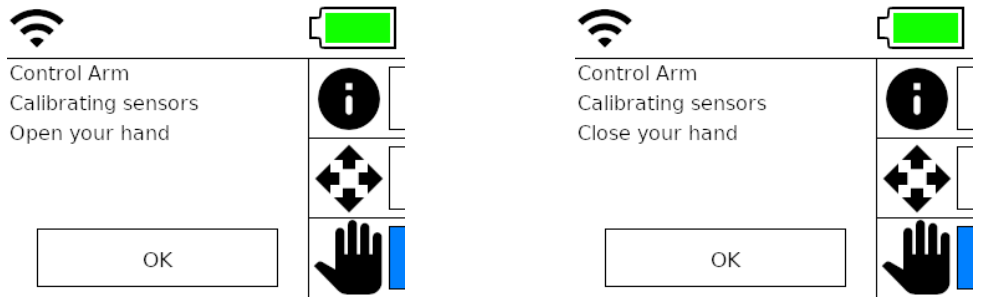


Abbildung II.4: Display-Darstellung der manuellen Referenzierung des Roboters

Fernsteuergerät Für die exakte Aufnahme der Sensordaten des Fernsteuergeräts werden bei jedem Benutzer die eingebauten Sensoren neu kalibriert. Die Kalibrierung kann gestartet werden, indem im Bedienmodus „Greifen“ die Schaltfläche „Calibrate sensors“ betätigt wird. Der Benutzer wird über den Text am Display dazu aufgefordert, seine Hand vollständig zu öffnen und dann zu bestätigen. Darauffolgend wird selbiges mit einer geschlossenen Faust durchgeführt.



(a) Kalibrierung mit geöffneter Hand (b) Kalibrierung mit geschlossener Hand

Abbildung II.5: Display-Darstellung bei der Kalibrierung des Fernsteuergeräts

4.5 Bedienmodi

Nachdem die obigen Schritte durchgeführt wurden, sind die Systeme betriebsbereit. Bei der Verwendung des Fernsteuergeräts wird zwischen zwei Bedienmodi unterschieden, welche über das Display gewählt werden können.

Fahren Im Bedienmodus „Fahren“ kann das Fahrwerk des Robotersystems kontrolliert werden. Dabei wird die Neigung der Hand dazu genutzt, die Geschwindigkeit und Richtung des Roboters vorzugeben. Eine Neigung des Handschuhs vor- und zurück resultiert in einer Vorwärts- bzw. Rückwärtsbewegung. Analog dazu kann durch Neigen des Handschuhs nach links bzw. rechts eine Seitwärtsbewegung durchgeführt werden. Um eine Drehbewegung auszuführen, muss der Benutzer die Finger zu einer Faust ballen. Danach führt eine Neigung des Handschuhs nach links oder rechts eine Drehung anstatt einer Seitwärtsbewegung aus. Durch den Mecanum-Wheel-Antrieb (siehe *Robotersystem* Abschnitt 4 und *Software* Abschnitt 6.4) ist jede Kombination dieser drei Elementarbewegungen möglich.

Greifen Auch im Modus „Greifen“ wird die Neigungsinformation der Hand genutzt. In diesem Fall gibt der Neigungswinkel die Position des Roboters vor. Wird die Hand zum Beispiel ganz nach vorne gekippt, fährt der Roboterarm in die vordere Endlage, analog dazu können alle anderen Endlagen und Zwischenpositionen angefahren werden.

Zudem wird in diesem Fall auch die Winkelinformation der Finger genutzt, um die Bewegungen der eigenen Finger mit denen des mechanischen Greifers zu synchronisieren. Außerdem werden Berührungen des Greifers durch Vibrationen in den Fingerspitzen des Fernsteuergeräts gemeldet.

4.6 Abschalten der Systeme

Fernsteuergerät Um das Fernsteuergerät auszuschalten, muss einfach der Schiebeschalter in die andere Position geschoben werden. Ein „Herunterfahren“ des Systems über das Display ist nicht vorgesehen.

Robotersystem Das Robotersystem wird wiederum über die Sicherung abgeschaltet, mit der es auch gestartet wurde. Auch hier entfällt ein „Herunterfahren“

5 Leitfaden

Die folgenden Kapitel gehen nun genauer auf die verschiedenen Aspekte des Projekts ein. In Kapitel III wird zu Beginn ein Überblick über die in diesem Projekt eingesetzten Technologien gegeben. Darauf folgt in Kapitel IV die Beschreibung des Robotersystems, dabei wird auf den mechanischen Aufbau, den Roboterarm und die verbaute Sensorik eingegangen. Anschließend beschreibt Kapitel V das Akkusystem und die elektrische Installation des Roboters. Kapitel VI gibt Einblicke in die Entwicklung und Fertigung des Fernsteuergerätes, hierbei wird auf die Mechanik,Elektronik und Feedback-Implementierung eingegangen. Zu guter Letzt wird in Kapitel VII ein Überblick über die gesamte Software, die für Fernsteuergerät und Robotersystem entwickelt wurde, gegeben.

In nachfolgenden Teilbereichen gilt in den einzelnen Kapiteln eine interne Referenzierung auf einen Unterpunkt in selbigem Kapitel – heißt: „siehe Abschnitt 2.3“ bedeutet, dass besagter Punkt innerhalb des Kapitels nachzuschlagen ist.

Wird allerdings auf Abschnitte oder Abbildungen verwiesen, die sich in einem anderen Kapitel befinden, so wird dies beispielsweise als „siehe Kapitel *Stand der Technik* Abschnitt 2.1.3“ angegeben. Die Namen der Kapitel sind dabei stets mit kursiver Schrift hervorgehoben, um eine bessere Lesbarkeit zu schaffen.

III Stand der Technik

1 Synchronmaschinen

1.1 Allgemeines

Synchronmaschinen finden größtenteils als Generatoren in Kraftwerken aller Art ihre Anwendung. Erst seit der Entwicklung der Frequenzumrichter werden Synchronmotoren als drehzahlregelbare Antriebe in Leistungsbereichen von Kleinstmotoren (z.B. in Uhren) bis hin zu Größtleistungs-Anwendungen wie Hochofengebläsen verwendet. Den Namen Synchronmaschine tragen sie, weil sie exakt mit ihrem Drehfeld synchron laufen.

1.1.1 Bauformen

Unterteilt werden Synchronmaschinen zunächst nach ihrer Bauform. Vollpolmaschinen (Abb. III.1 a) haben als Läufer eine massive Stahlwalze, in die radial Nuten eingefräst sind. In diesen Nuten wird die Erregerwicklung geführt, welche auf mehrere konzentrische Spulen verteilt ist. Schenkelpolmaschinen in der Innenpol-Ausführung (Abb. III.1 b) besitzen im Läufer ausgeprägte Pole. Um diese ist die Erregerwicklung geführt. Die Polschuhe sorgen für einen möglichst sinusförmigen Feldstrom. Schenkelpolmaschinen in der Außenpol-Ausführung (Abb. III.1 c) haben ihre Pole im Ständer – ähnlich einer Gleichstrommaschine – und werden zur bürstenlosen Erregung von Generatoren verwendet. Im Läufer befindet sich eine Drehstromwicklung[1].

[1] vgl. [25], S. 340ff.

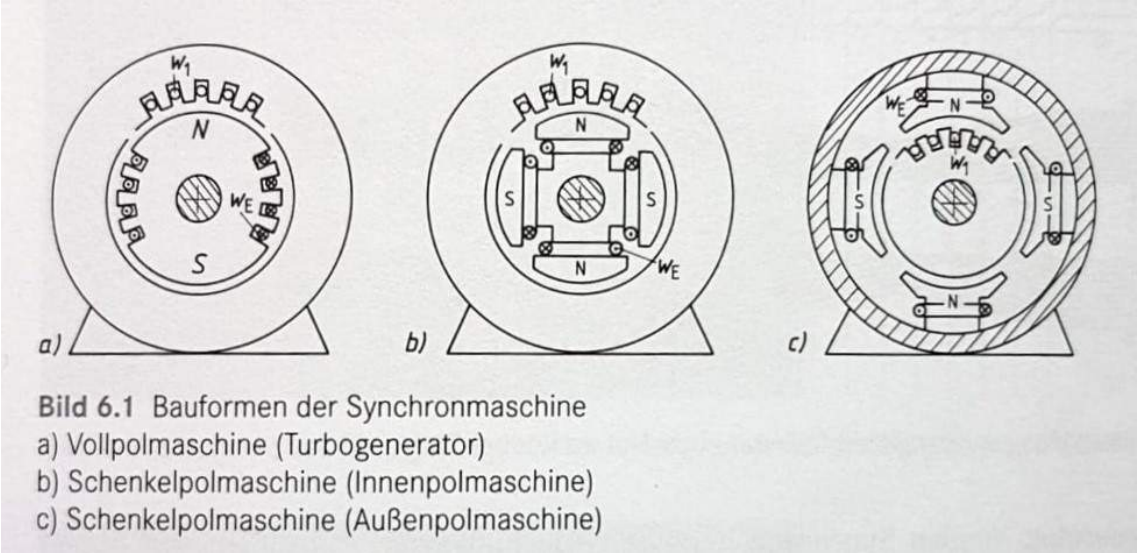


Bild 6.1 Bauformen der Synchronmaschine
a) Vollpolmaschine (Turbogenerator)
b) Schenkelpolmaschine (Innenpolmaschine)
c) Schenkelpolmaschine (Außenpolmaschine)

Abbildung III.1: Bauformen von Synchronmaschinen[25]

1.1.2 Funktionsweise

Bei Generatorbetrieb wird der Rotor über Schleifringe mit einer externen Gleichstromquelle verbunden, wodurch er wie ein Permanentmagnet wirkt. Durch eine Drehung induziert er in die Ständerspule eine beinahe ideale Sinusspannung. Das Drehfeld kann dabei nur synchron mit dem Läufer rotieren. Soll eine Synchronmaschine nun als Motor betrieben werden, muss das Drehfeld durch Anlegen eines Drehstromsystems an die Statorwicklungen erzeugt werden. Der Läufer wird dann synchron mit dem erzeugten Feld mitgezogen. Der Polradwinkel[2] ändert sich je nach Art der Belastung. Unbelastet laufen die beiden gemeinsam, ohne eine Verschiebung. Im belasteten Generatorbetrieb eilt der Läufer dem Drehfeld etwas voraus, der Polradwinkel wird größer als 0°. Im belasteten Motorbetrieb ist genau das Gegenteil der Fall: Der Rotor wird, wie bereits erwähnt, vom Drehfeld mitgezogen, was bedeutet, dass der Winkel kleiner als 0° wird.

1.2 Synchronmaschinen mit Dauermagneterregung

Im Projekt wurden Synchronmaschinen mit Dauermagneterregung verwendet. Der Läufer ist dabei wie in Abb. III.2 ausgeführt. Entlang dessen Umfang sind Permanentmagneten angebracht, im Läuferblech sind Aussparungen, welche das Trägheitsmoment reduzieren. Der Ständer ist meist sechspolig ausgeführt, da hier eine gute Ausnutzung des Bauvolumens besteht. Bei gleicher Baugröße erreichen Permanenterregte Synchronmotoren eine mindestens doppelt so große Bemessungsleistung wie Asynchronmaschinen und haben gleichzeitig geringere Gesamtverluste.

Weitere Vorteile der PSM[3] sind:

- Durch die Permanentmagneten muss von der Ständerwicklung keine Blindstromkomponente zur Magnetisierung aufgenommen werden. Dies bedeutet höhere Wirkströme und somit höhere Leistungen.
- Durch höhere Luftspaltflussdichten ($B_{rem} = 1,5T$) lassen sich ebenfalls höhere Leistungen erreichen.

[2] Winkel zwischen Drehfeld und Läufer
[3] Permanentmagneterregte Synchronmaschine

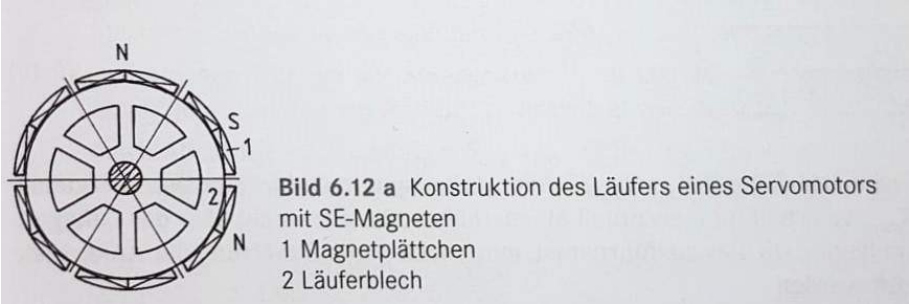


Abbildung III.2: Läufer einer dauermagneterregten Synchronmaschine[25]

Für den Betrieb von PSM werden immer Frequenzumrichter benötigt (siehe Abb. III.3). Außerdem müssen ein Drehzahl- und ein Polradlagegeber im Motorgehäuse integriert sein. Das ist notwendig, da im Betrieb des Motors mit sinusförmigen Strömen die exakte Läuferfeldlage bekannt sein muss, weshalb auch ein Resolver oder ein hochauflösender inkrementaler Geber nötig ist[4].

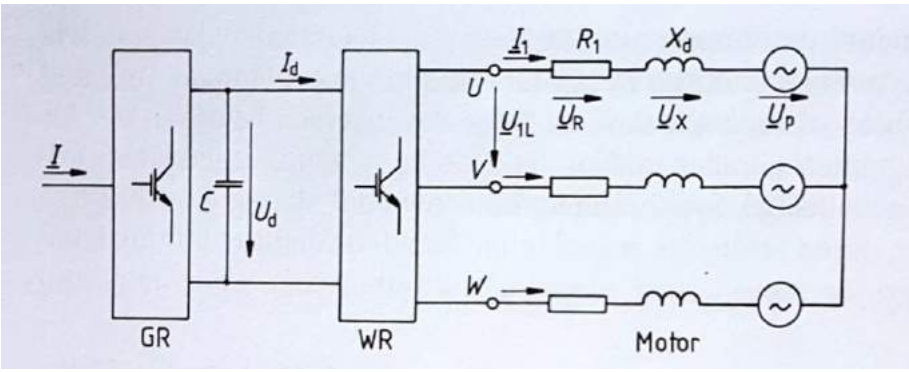


Abbildung III.3: Aufbau eines AC-Servomotorsystems[25]

Der Eingangsgleichrichter (im Bild GR) kann als B6-Brücke mit IGBTs realisiert werden, was den Vorteil hat, die Netzrückwirkungen des Umrichters zu minimieren. Der Wechselrichter auf der Motorseite (WR) wird durch eine fixe Zwischenkreisspannung U_d gespeist. Ein Mikroprozessor steuert über diesen die drei Sinusströme I_1 des Motors[5].

[4] vgl. [25], S. 405.
[5] vgl. [25], S. 406.

2 Akkusysteme

Elektrochemische Speicher haben in den letzten Jahren mehr an Bedeutung gewonnen und werden diesen Trend in Zukunft fortsetzen. Verwendet werden sie in Form von kleinen Lithium-Ionen Akkus in Smartphones, bis hin zu großen stationären Energiespeichern für erneuerbare Energien. Ebenfalls sind elektrische Energiespeicher ein wichtiger Bestandteil für den Erfolg der Elektromobilität geworden. Dabei gibt es viele verschiedene Arten von Batterien, die sich in ihrem chemischen Aufbau unterscheiden. Besonders hat sich, aufgrund der chemischen Eigenschaften, der Lithium-Ionen-Akku durchgesetzt[6].

2.1 Batteriearten

2.1.1 Bleiakkumulator

Mitte des 19. Jahrhunderts wurde der Bleiakkumulator zum ersten Mal untersucht. Dieser besitzt eine Zellspannung von 2V, was eine sehr hohe Spannung für sogenannte „wässrige Systeme“ ist. Der Name kommt daher, da wässrige Schwefelsäure als Elektrolyt verwendet wird. Weiters besteht die Kathode aus Blei und die Anode aus Bleioxid. Heutzutage werden die meisten Akkus mit einem festgelegtem Elektrolyt hergestellt. Beim Blei-Gel-Akku zum Beispiel, wird das Elektrolyt durch Zusatz von Kieselsäure geliert. Der größte Nachteil des Bleiakkumulators ist das Gewicht, da nur 30 bis 40Wh/kg erreicht werden können. Auch die Zellenstabilität ist im Vergleich zu anderen Akkus gering. Dennoch kann man sie kurzzeitig mit hohem Strom belasten, weshalb Bleiakkus in Autos als Starterbatterie zum Einsatz kommen. Unter anderem besitzt der Bleiakku 50% des Batteriemarktes – ein Großteil wird in Autos verbaut[7].

2.1.2 Nickel-Cadmium- und Nickel-Metallhybrid-Akkumulatoren

Nickel-Cadmium- und Nickel-Metallhybrid-Akkumulatoren zeichnen sich dadurch aus, dass sie einen internen Überlade- und Entladeschutz integriert haben. Daher muss keine aufwendige elektronische Beschaltung durchgeführt werden. Um 1900 wurde der Nickel-Cadmium Akkumulator von W. Jungner entwickelt. Verwendet wird als Kathodenmaterial Nickeloxidhydroxid und als Elektrolyt Kalilauge. Die Anode besteht aus Cadmium, welches eine hohe spezifische Ladung besitzt (477Ah/kg). Mit einer Zellenspannung von 1,2 V ergibt sich eine spezifische Energie von 60Wh/kg. Ebenfalls zeichnet sich der Akkumulator dadurch aus, dass der Betrieb bis $-40°C$ möglich ist und durch sehr hohe Strombelastbarkeit. Jedoch wurde die Verwendung, von der EU, auf medizinische und sicherheitsrelevante Bereiche begrenzt[8].

Der Nickel-Metallhydrid-Akkumulator, der erst 1990 entwickelt wurde, verwendet als Elektrolyt – anstelle von Cadmium – Wasserstoffspeicherlegierungen aus Nickel und Seltenen Erden. Diese Akkumulatoren erreichen bis zu 80Wh/kg. Seitdem sie vom Lithium-Ionen-Akku aus der Consumerelektronik verdrängt wurden, werden diese hauptsächlich in Hybridfahrzeugen eingebaut[9].

[6] vgl. [37], S.3.
[7] vgl. [37], S.5.
[8] vgl. [37], S.5.
[9] vgl. [37], S.6.

2.1.3 Lithium-Ionen-Batterie

1991 kommerzialisierte SONY die Lithium-Ionen-Batterie und verdrängte damit die Nickel-Metallhybrid-Technologie im Consumerbereich der Mobiltelefone und portablen PC's. Ebenso wurden die Nickel-Cadmium-Zellen 2005 als Spitzenreiter von den Lithium-Ionen-Batterien verdrängt. Bei Li-Ion-Akkus besteht die positive Elektrode aus $LiCoO_2$ und die negative aus Kohlenstoff. Das Elektrolyt besteht aus organischen Carbonaten und Lithiumhexafluorophosphat ($LiPF_6$)[10]

Die chemischen Eigenschaften von Lithium-Ionen-Batterien können sich grundsätzlich mit der Zeit verändern. Das liegt daran, dass die Zellen aus Materialien bestehen, die miteinander reagieren können. Diese chemische Reaktion kann unter anderem, durch zunehmende Temperaturen beschleunigt werden. Die Kapazität einer Akkuzelle nimmt mit der Zeit ab, durch diese Abnahme steigt der Innenwiderstand der Zelle und gleichzeitig wird die Leistung verringert. Heutzutage werden jedoch die Zellen so ausgelegt, dass die spezifische Kapazität bis zum Ende der Lebensdauer gegeben ist.

Bei Lithium-Ionen-Batterien gibt es noch weitere Elektrodenmaterialien. In Abbildung III.4 sind weitere Elektrodenkmaterialien und deren wichtigsten Eigenschaften abgebildet.

System	Zellspannung	Temperatur	spez. Energiedichte		Zyklen	Leistung
	V	°C	Wh/kg	Wh/l		
$LiCoO_2$	3,6	-20/60	140-190	360-500	800-1200	N-M
NCA	3,5	-20/60	220-240	500-630	800-1200	N-M
NCM	3,7	-20/60	100-150	230-400	500-700	M-H
Mn spinel	3,7	-20/60	130-150	300-320	500-700	H
Fe phosphate	3,3	-30/70	100-140	250-380	>1000	SH
Nexelion	3,5	-20/60	160	480	1000	M-H

N ... Niedrig
M ... Mittel
H ... Hoch
SH ... Sehr Hoch

Abbildung III.4: Übersicht verschiedener Elektrodenmaterialien[40] [55]

Bei hohen Temperaturen oder Spannungen kommt es zu einer starken thermischen Reaktion. Wie zum Beispiel beim „Eisen-Phosphat-System“ liegt die Ladeschlussspannung bei 3,6V[11]. Wird diese überschritten, kann die Batterie zerstört werden. Gleiches gilt für die Unterschreitung der Entladeschlussspannung – diese liegt bei 2,5V. Aus Sicherheitsgründen sollte dieser Spannungsbereich genaustens eingehalten werden[12].

Seit dem Jahr 1991 werden Lithium-Ionen-Batterie in mobilen Consumer-Geräten – wie z.B in Notebooks oder Mobil Telefons – eingebaut, weil sie trotz ihres geringen Gewichtes, eine hohe Energiedichte aufweisen. Li-Ion-Akkus sind auch häufig in mobilen Werkzeugmaschinen wiederzufinden. In der Elektromobilität spielen sie heutzutage auch eine große Rolle und werden zum Beispiel für Elektroroller, Pedelecs oder andere Automotive Anwendungen verwendet[13].

[10] vgl. [37], S.8.
[11] [39], Aus Tabelle entnommen.
[12] [55], vgl.
[13] vgl. [37], S.13-14.

2.2 Batteriemanagementsystem

Grundsätzlich besteht die Aufgabe eines BMS[14] darin, die Batterie zu schützen sowie die Lebenszeit und Zyklenzahl zu erhöhen. Besonders wichtig ist dies bei Lithium-Ionen-Batterien, da diese sehr empfindlich gegenüber hohen Spannungen und Temperaturen sind[15].

2.2.1 Komponenten

Ein Akkusystem besteht meistens aus mehreren Zellen. Die erste Komponente des BMS ist zuständig für die Spannungs- und Temperaturüberwachung der einzelnen Zellen. Ebenfalls ist sie für den Ladungsausgleich zwischen den Zellen verantwortlich. Solch eine Überwachungseinheit wird auch Cell Supervising Circuit (CSC) genannt.

Die nächste Komponente wird Kontrolleinheit genannt. Diese übernimmt die Berechnung des SOC[16] sowie des SOH[17] und steuert den Ladungsausgleich. Weiters kommuniziert sie auch mit dem Fahrzeug (z.B mittels CAN-Bus Abschnitt 5.4) und übermittelt unter anderem den SOF[18]. Die Kontrolleinheit versetzt sich auch selbstständig in einen Ruhezustand, um den eigenen Stromverbrauch auf ein Minimum zu reduzieren.

Im Fall eines Fehlers (Kurzschluss oder Übertemperatur), wird die Batterie mithilfe eines HS-Kontaktors von der Last getrennt, um die Batterie zu schützen. Des Weiteren ist auch eine Schmelzsicherung für den Fall eines Kurzschlusses eingebaut. Dieser Kontaktor übernimmt ebenso die Trennung der Batterie im Ruhezustand.

Die vorletzte Komponente ist eine Strommessvorrichtung. Dabei werden oft zwei voneinander unabhängige Systeme verwendet. Heutzutage wird meistens ein Widerstand als Messsensor oder eine Messung über das vorhandene elektromagnetische Feld verwendet.

Die letzte Komponente ist zuständig dafür, dass die Batterie innerhalb eines optimalen Temperaturbereiches betrieben wird. Diese ist notwendig, da sich – wie bereits im Abschnitt 2.1.3 erwähnt – die Temperatur der Batterie auf die Lebensdauer auswirkt[19].

2.2.2 Balancing

Balancing, bezogen auf Batterien, bedeutet Ladungsausgleich. Da sich jede Zelle in der chemischen Struktur minimal unterscheidet, laden oder entladen einige schneller oder langsamer.

[14] kurz für Batteriemanagementsystem
[15] vgl. [37], S.117.
[16] State of Charge, beschreibt den momentenen Ladezustand
[17] State of Health, beschreibt den aktuellen Zustand der Batterie
[18] State of Function, gibt die Funktionalität der Batterie an
[19] vgl. [37], S.179.

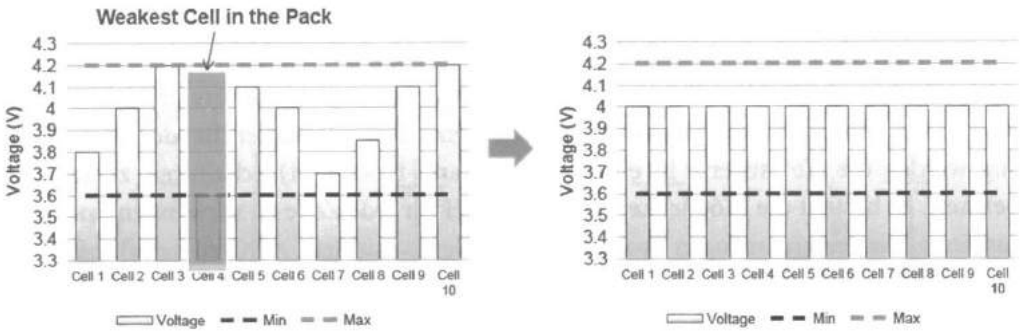


Abbildung III.5: Ungleichmäßige Ladungen [37]

Die Dauer des Ladevorgangs richtet sich immer nach der Zelle mit dem niedrigsten Potenzial. Erst wenn die letzte Zelle geladen ist, kann der Ladevorgang abgeschlossen werden. Um dabei alle Zellen möglichst gleichmäßig aufzuladen, wurde eine Methode entwickelt, welche „Balancing“ genannt wird.

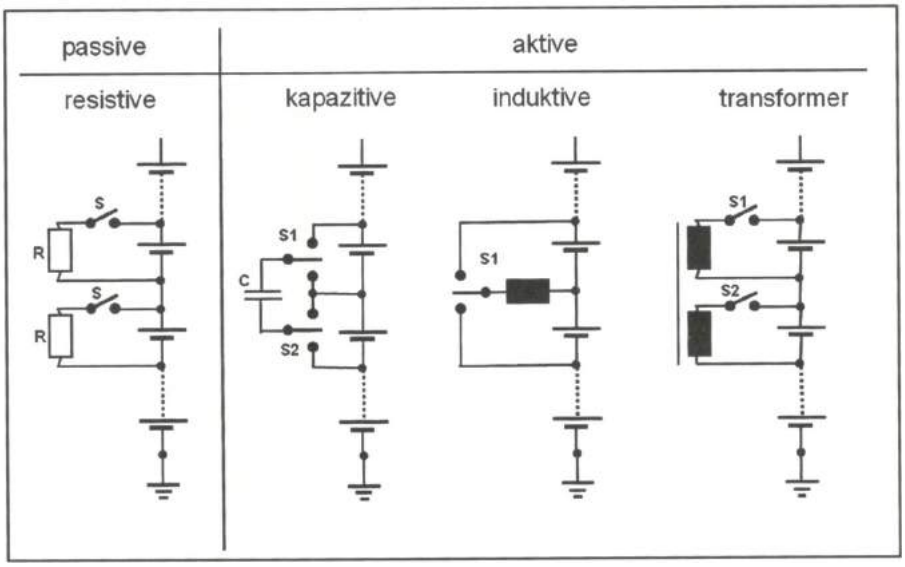


Abbildung III.6: Ladungsausgleich Methoden [37]

Dabei unterscheidet man zwischen „passive Balancing“ und „active Balancing“.

Beim „passive Balancing“ wird die überschüssige Energie in Wärme umgewandelt. Wie man in Abbildung III.6 erkennen kann, wird dies mit einem parallel geschalteten Widerstand realisiert. Wenn sich eine Zelle schneller aufladen würde als die anderen, wird diese Zelle über einen Schalter (S) mit dem Widerstand parallel geschlossen. Somit kann man beim Laden dafür sorgen, dass alle Zellen gleich schnell aufgeladen werden. Beim Entladen kann diese Methode nicht angewendet werden.

Grundsätzlich wird beim „active Balancing“ Energie von einer Zelle in eine andere Zelle transferiert. Diese Ladungsausgleichsmethode kann man mit drei verschiedene Arten realisieren. Bei der ersten Art wird überschüssige Energie in einen Kondensator geladen, dieser gibt die gespeicherte Energie anschließend an eine andere Zelle weiter. Dabei werden S1 und S2 jeweils mit dem unteren Kontakt verbunden (siehe Abbildung III.6). Der

Kondensator wird anschließend aufgeladen. Wenn die beiden Schalter mit dem oberen Kontakt verbunden werden, entlädt sich der Kondensator über die obere Zelle. Die induktive Art basiert auf demselben Prinzip. Die Spule wird aufgeladen und die Energie wird anschließend auf eine andere Zelle abgegeben. Diese zwei Arten werden aber nur bei Akkusystemen mit einer kleinen Zellenanzahl verwendet, da die Effizienz geringer als bei der verbleibenden Art ist. Bei dieser wird die Batterie über einen Schalter mit einer Wicklung eines Transformators verbunden. Bei Bedarf kann die überschüssige Energie einer Zelle – über den Transformator – zu einer Anderen transferiert werden.

„Active Balancing“ kann auch beim Entladen angewendet werden. Wenn sich Zelle 1 schneller entlädt als Zelle 2, kann Energie von der Zelle 2 zu Zelle 1 transferiert werde. Damit kann man erzielen, dass die Entladeschlussspannung der Zelle 1 später erreicht wird[20].

2.3 Einsatzfelder für Lithium-Ionen-Batterien

Da es irgendwann keine fossilen Energieträger mehr geben wird, muss der Mensch sich auf die Suche nach Alternativen begeben. Vor allem in puncto Mobilität. Elektromobilität gibt es schon seit 100 Jahren, jedoch hauptsächlich nur in Nischenanwendungen. Da elektrochemische Speicher, im Gegensatz zu chemischen Speichern (Benzin und Öl), klar im Nachteil sind (Blei-Säure-Batterie ca. 40 Wh/kg, Benzin ca. 12.000 Wh/kg), ist die Reichweite von Elektrofahrzeugen erheblich eingeschränkt. Deswegen ist man auf der Suche nach elektrochemischen Speichern mit hohen Energiedichten. Lithium-Ionen-Batterien sind bereits eine große Verbesserung gegenüber der Blei-Säure-Batterie, jedoch noch immer weit entfernt von der Energiedichte des Benzins. Es gibt jedoch Übergangslösungen, wie Hybridfahrzeuge (HEV), die mit einem Verbrennungsmotor oder Elektromotor angetrieben werden können[21].

2.3.1 Automobile Anwendungen

Im Grunde gibt es drei Faktoren, die dazu beitragen, Elektrochemische Batterien in der E-Mobilität zu verbessern: Gewichtsreduktion, Rekuperation und Downsizing. Für die Gewichtsreduktion wird im klassischen Automobil die Blei-Säure-Batterie durch eine Lithium-Batterie ersetzt. Bei der Rekuperation, wird die Bremsenergie wieder in elektrische Energie umgewandelt. Downsizing bedeutet, dass bei Hybridfahrzeugen der Elektromotor beim Beschleunigen hauptsächlich als Unterstützung dient.

Bei Elektroautos entspricht 1 kWh ungefähr 5-10 km an Reichweite. Hybridfahrzeuge haben oft Batterien mit 10kWh eingebaut, was einer Reichweite von ungefähr 50km entspricht. Diese Batterien können auch von zu Hause aufgeladen werde. Um eine Alltagstaugliche Reichweite bei reinen Elektrofahrzeugen zu generieren, werden Akkus mit Speicherkapazitäten von 20 kWh bis 60 kWh benötigt. Heutzutage liegen die meisten Elektroautos bei einer Reichweite von ungefähr 300 km. Um bei Elektroautos die Reichweite noch etwas zu erhöhen, gibt es sogenannte „EV mit Range Extender“, bei diesem wird die Batterie zusätzlich mit einem Verbrennungsgenerator aufgeladen.

[20] vgl. [37], S.182-184.
[21] vgl. [37], S.383.

	xEV	Batteriegröße	Verhältnis Leistung/ Energie (P/E)	Energieein-sparung	Primäre Energiequelle
Start-Stop		≤1 kWh	10	Leerlauf	Benzin/Diesel
Micro-Hybrid	HEV	≤1 kWh	20	Rekuperation	Benzin/Diesel
Hybrid	HEV	1–2 kWh	20	Rekuperation /Downsizing	Benzin/Diesel
Plug-in Hybrid	PHEV	5–15 kWh	5–15	Rekuperation /Downsizing /Stromnetz	Benzin /Stromnetz
Full EV	BEV	20–60 kWh	2–3	Rekuperation /Stromnetz	Stromnetz
EV mit Range Extender		20–40 kWh	2–3	Rekuperation /Stromnetz	Stromnetz/ Benzin/Diesel

Abbildung III.7: Charakterisierung verschiedener Elektrofahrzeuge [37]

Da Elektroautos eine sehr geringe Reichweite (durchschnittlich 300 km) haben, muss die Batterie fast jeden Tag aufgeladen werden. Dabei kann die Steckdose zu Hause benutzt und damit die ganze Nacht geladen werden. Alternativ kann eine Schnellladestation verwendet werden, die den Akku nach einer Stunde bereits vollständig aufladen hat. Dies ist aber technisch sehr anspruchsvoll und es wird dafür eine gut ausgebaute Infrastruktur benötigt. Es gibt auch schon Konzepte, bei denen der entladene Akku gegen einen vollgeladenen ausgetauscht wird. Dies würde die Wartezeit auf ein Minimum reduzieren[22].

[22] vgl. [37], S.387-388.

3 Sensorik

Sensoren sind Elemente, die es ermöglichen, physikalische, chemische oder biologische – also nicht elektrische – Größen in elektrische Messsignale umzuwandeln, welche dann zur Steuerung und Regelung von mechatronische Systemen weiterverarbeitet werden können[23].

Die Funktionsweise kann mit dem Blockschaltbild aus Abbildung III.8 beschrieben werden.

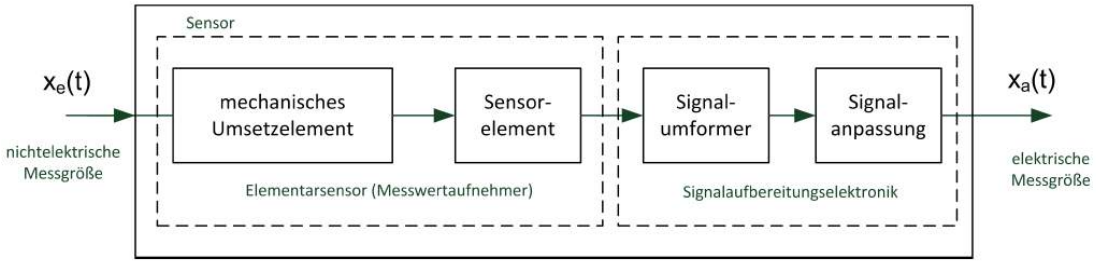


Abbildung III.8: Blockschaltbild eines Sensors

Beim mechanischen Umsetzelement handelt es sich um eine Einrichtung, welche die zu messende Größe in eine andere mechanische Größe umwandelt (zum Beispiel Kraft in Dehnung), wobei das Sensorelement – physikalisch mit dem Umsetzelement gekoppelt – diese dann in eine elektrische Größe (z.B. Widerstandsänderung) wandelt. Diese zwei Komponenten bilden zusammen den Elementarsensor, welcher auch Messwertaufnehmer genannt wird.

Moderne, meist teurere, Sensoren bieten zusätzlich die Möglichkeit, die elektrischen Ausgangsgrößen als digitale Daten über ein serielles Bussystem weiterzugeben.

[23] vgl. [62], S. 3.

3.1 Ultraschallsensoren

Ultraschallsensoren verwenden Schallwellen im nicht-hörbaren Frequenzbereich und werden zur Abstandsmessung, Objekterfassung und Objekterkennung in der Automatisierungstechnik eingesetzt. Die dabei gängigste Ausführung stellt der sogenannte „Ultraschall-Sonar-Sensor“ dar, welcher aus einem Sende- und einem Empfangswandler besteht. Je nach Anordnung spricht man von einem Streckensensor oder einem Distanzsensor (siehe III.9). Streckensensoren detektieren Veränderung in der Signalstrecke, um beispielsweise Strömungen zu erkennen oder Durchflussmessungen durchzuführen. Distanzsensoren erfassen Objekte aus der Entfernung über deren Reflexion der ausgesendeten Schallwelle. Sie werden zur Kollisionsvermeidung, Enfernungsmessung und Füllstandsmessung verwendet.

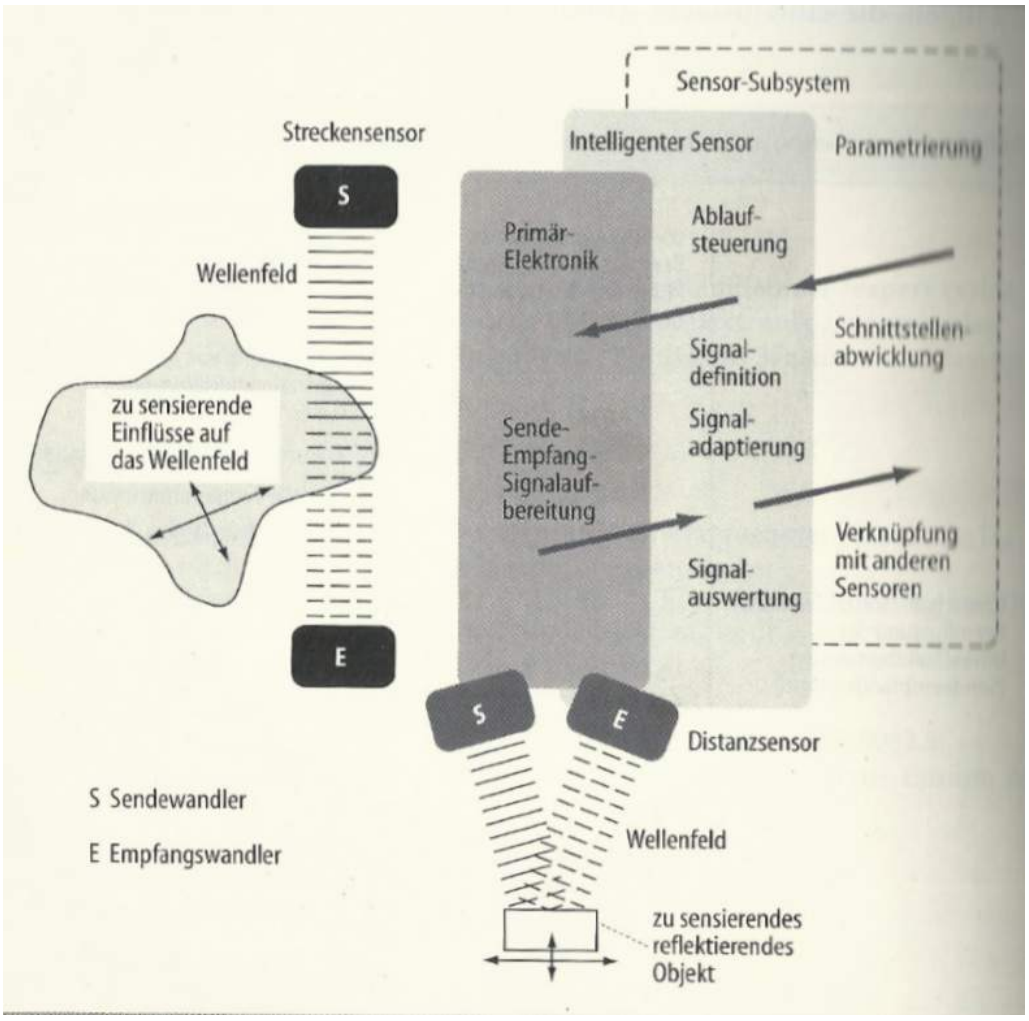


Abbildung III.9: Strecken- und Distanzsensoren[62]

Im Falle eines Distanzsensors lässt sich der Abstand d zwischen Messobjekt und Sensor sehr leicht nach folgender Formel bestimmen:

$$d = \frac{\tau c}{2} \tag{III.1}$$

τ ist dabei die Laufzeit des Echos und c die Schallgeschwindigkeit im Ausbreitungsmedium (343,2 m/s für Luft bei 20°C)[24].

[24] vgl. [62], S. 511ff.

3.2 Drucksensoren

Im Folgenden wird ein kurzer Auszug von Sensoren vorgestellt, durch die physikalischer Druck gemessen werden kann.

Dehnmessstreifen[25]

Die in der Praxis meistgenutzte Form eines Drucksensors stellt das Prinzip des Dehnmessstreifens – kurz DMS – dar. Zwar gibt es beim DMS weiters verschiedene Unterkategorien, allerdings wird hier nur der sogenannte „metallische Dehnmessstreifen“ als Beispiel herangezogen.

Um den Druck, also die mechanische Eingangsgröße, in eine elektrische Ausgangsgröße zu wandeln, wird ein mehrstufiges Verfahren angewandt, welches in Abbildung III.10 schematisch dargestellt ist.

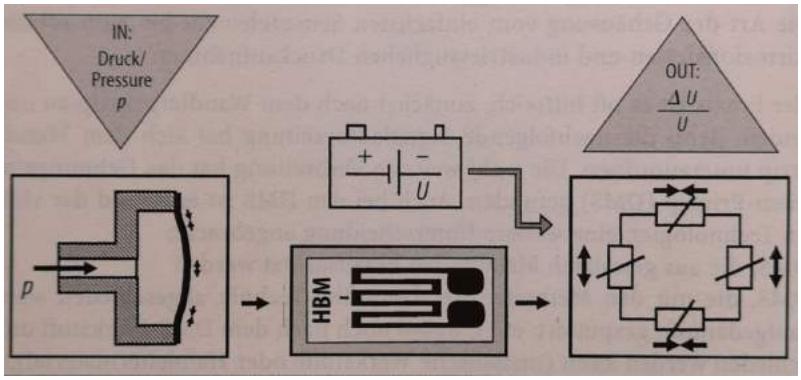


Abbildung III.10: Messgrößentransformation bei DMS Druckaufnehmer [62]

Zuerst wird der zu messende Druck in eine Oberflächendehnung transformiert, wodurch eine Deformation einer Membran hervorgerufen wird. Die resultierende Dehnung/Stauchung in Abhängigkeit zur Längenänderung kann folgendermaßen definiert werden.

$$\epsilon = \frac{\Delta l}{l} \tag{III.2}$$

Die Dehnung des Messstreifens ruft eine Widerstandsänderung hervor, welche in Abhängigkeit zur Proportionalitätskonstante k und der Dehnung steht:

$$\frac{\Delta R}{R} = k * \epsilon \tag{III.3}$$

ΔR ist dabei die Änderung des Widerstandswertes, wobei R den ursprünglichen Nennwiderstand charakterisiert.

Mittels einer sogenannten Wheatstone-Brücke (siehe Abbildung III.11) wird die Widerstandsänderung in eine proportionale Spannungsänderung umgewandelt.[26]

[25] vgl. [62], S. 329-333.
[26] vgl [62], S.329-333.

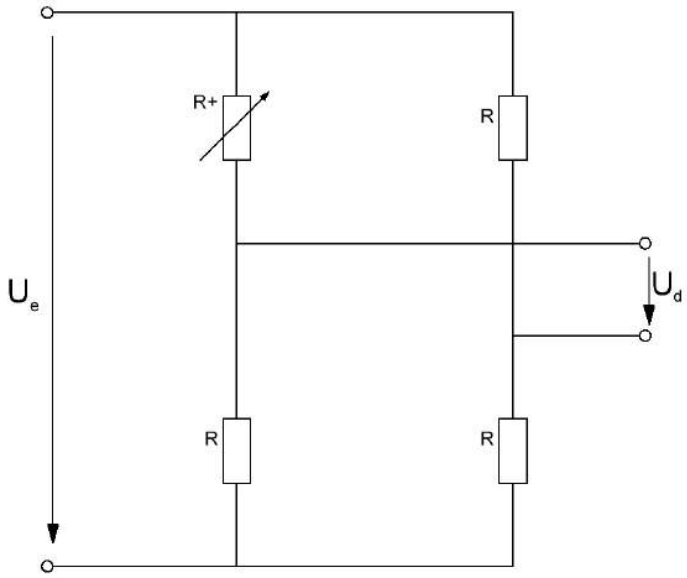


Abbildung III.11: Wheatstone-Brücke in Ausführung einer 1/4-Brücke

Kapazitive Drucksensoren Eine weitere Möglichkeit zur Messung von Druck wird mittels kapazitiven Drucksensoren realisiert. Dabei wird die durch Druck verursachte Abstandsänderung zwischen zwei Elektroden und die daraus resultierende Kapazitätsänderung für die Ermittlung des Drucks herangezogen.

Der Vorteil von kapazitiven Drucksensoren besteht darin, dass neben dem geringen Energieverbrauch eine hohe Basisempfindlichkeit sowie eine geringe Temperaturabhängigkeit des Ausgangssignals charakteristisch ist.[27]

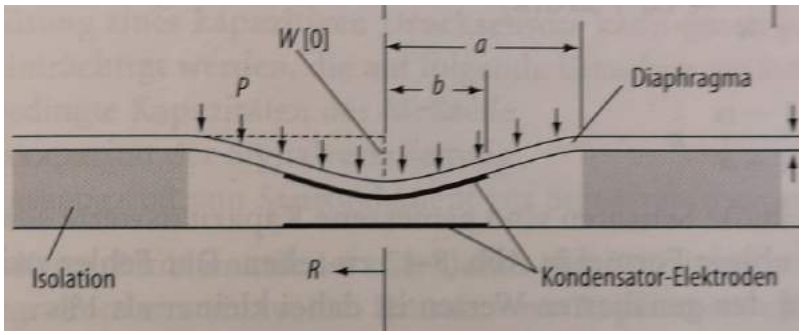


Abbildung III.12: Darstellung eines kapazitiven Drucksensors [62]

[27] vgl. [62], S. 375.

Force Sensing Resistor Wie auch bei einem Dehnmessstreifen, handelt es sich hierbei um eine Art von Sensor, bei dem der zu messende Druck über eine resistive Wirkungsweise ermittelt wird. Beim Force Sensing Resistor – kurz FSR – besteht die einfachste Ausführung aus zwei Membranen, die an den Rändern durch eine dünnes klebendes Material getrennt werden – ersichtlich in Abbildung III.13.

Die in der Abbildung obere Membran ist mit einer elektrisch leitenden Tinte beschichtet, welche durch die Anwendung von Druck gegen die Kontaktflächen der unteren gedrückt wird. Dadurch ändert sich der Gesamtwiderstand zwischen den Anschlussleitungen des Sensors in Abhängigkeit des Drucks, dessen Erhöhung einen geringeren Widerstandswert zur Folge hat.[28]

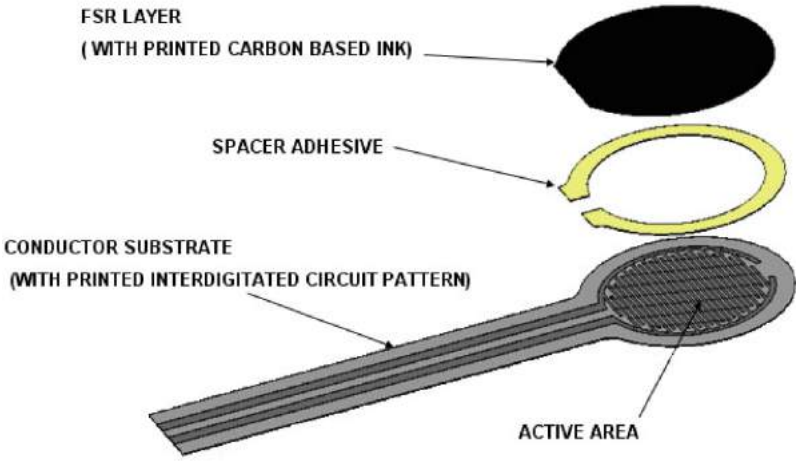


Abbildung III.13: Darstellung des Aufbaus eines FSRs [35]

3.3 Biegesensoren

Für die Messung von Biegungen bzw. Winkeln im Verhältnis zu anderen, fixierten Komponenten werden verschiedenste Sensoren eingesetzt. In dieser Arbeit werden lediglich zwei der Möglichkeiten vorgestellt.

DMS Die bereits in Abschnitt 3.2 vorgestellten Dehnmessstreifen können ferner als Biegesensoren verwendet werden, indem sie geschickt auf dem zu messenden Objekt platziert werden. In Abbildung III.14 wird eine exemplarische Platzierung der DMS veranschaulicht, die als „Clip-On-Cage“ bezeichnet wird.

Dabei ist eine Seite des Biegebalkens am Montagegehäuse angebracht und wird auf der anderen nach oben oder unten gedrückt.

Die aus der Dehnung der DMS resultierende Widerstandsänderung kann mittels geeigneter Auswerteschaltungen (z.B. Vollbrücke) als Biegewinkel interpretiert werden.[29]

[28] vgl. [35], 2.0 Theory of Operation.
[29] vgl. [62], S. 479-481.

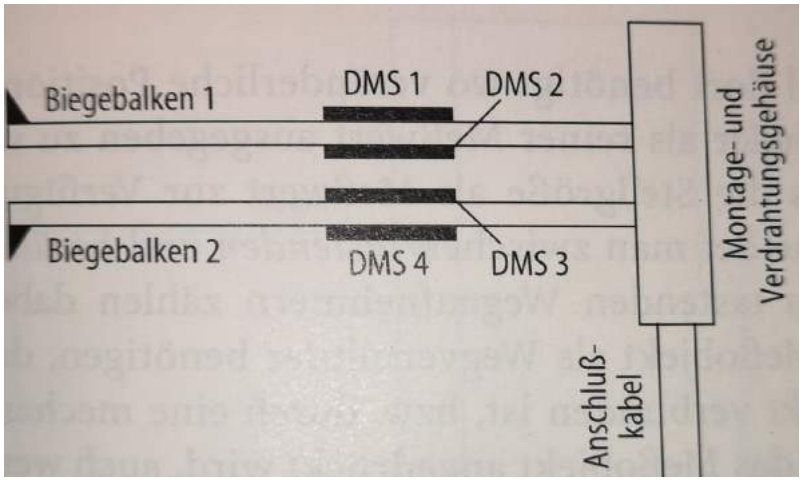


Abbildung III.14: Messung von Biegungen mittels Clip-On-Cage-DMS [62]

Flex-Sensor Die zweite in dieser Arbeit vorgestellte Möglichkeit zur Messung von Biegungen ist der „Flex-Sensor“. Wie auch beim DMS erfolgt eine Auswertung mittels eines resistiven Elements, allerdings wird besagter Sensor ausschließlich bei Messungen von Biegungen angewendet.

Das Produkt des Unternehmens „SpectraSymbol“ eignet sich laut Hersteller unter Anderem für Anwendungen in Robotik, Medizin, Gaming (Virtual Motion). Dabei wird durch eine Biegung des Sensors eine Widerstandsänderung hervorgerufen, welche in einem hohen Bereich liegt, sodass auf eine zusätzliche Auswerte- bzw. Verstärkerschaltung verzichtet werden kann.

Befindet sich der Sensor in nicht-gebogenem Zustand erzielt man den kleinsten Widerstandswert, wohingegen bei erhöhtem Biegewinkel der Widerstandswert erhöht wird.[30]

Abbildung III.15: Flex-Sensor von SpectraSymbol[58]

[30] vgl. [57].

Über das Funktionsprinzip des Sensors sind zwar in Internet-Foren einige Spekulationen aufzufinden, allerdings werden diese vom Hersteller weder kommentiert, noch von eigener Seite beschrieben.

Auf Nachfrage beim Hersteller heißt es „When the sensor is bent, it induces a physical change in the sensor element that increases the resistance. We do not share a lot of details beyond that because we consider them to be proprietary“ – zu deutsch: „Wenn der Sensor gebogen wird, induziert dies eine physikalische Änderung des Sensorelements, welches den Widerstandswert erhöht. Wir teilen nicht viele Informationen, da wir diese als Firmeneigentum ansehen.“

3.4 Inertiale Messeinheiten

Eine Inertiale Messeinheit (IMU, engl. für inertial measurement unit) ist eine Kombination mehrere Inertialsensoren wie Drehratensensoren (Gyrosensoren) und Beschleunigungssensoren.

Die Kombination dieser Sensoren erlaubt es, die Lage und Bewegung des Sensors im Raum zu messen. Diese Sensoren können auch als integrierter Schaltkreis in der sogenannten MEMS[31]-Technik ausgeführt werden. Dieses Konzept wird heute in der Unterhaltungselektronik – zum Beispiel in Smartphones – vielfach eingesetzt[32].

3.4.1 Drehratensensoren (Gyrosensoren)

Digitale Gyroskope (= Gyrosensoren) liefern die Winkelgeschwindigkeit um eine oder mehrere Achsen als digitales Signal[33].

3.4.2 Beschleunigungssensoren

Beschleunigungssensoren messen die Beschleunigung des Sensors in einer oder mehreren Achsen. Sie können zum Beispiel als kapazitive Sensoren in MEMS-Technik ausgeführt sein. Dazu wird eine seismische Testmasse verwendet, die aufgrund des Newton'schen Trägheitsgesetzes mit der Kraft $F = m \cdot a$ ausgelenkt wird. Diese Verschiebung verändert die Abstände und damit die Kapazitäten zweier Plattenkondensatoren, wodurch die Auswerteelektronik auf die Beschleunigung zurückrechnen kann.

Da diese Kapazitätsänderung nur sehr gering ist, wird diese Mikromechanik und die Auswerteelektronik in einem gemeinsamen IC realisiert, um Störeinflüsse zu vermeiden[34].

3.4.3 Magnetfeldsensoren

Um eine absolute Lage im Raum zu berechnen, wird bei manchen inertialen Messeinheiten ein Magnetfeldsensor integriert. Diese können die magnetische Flussdichte in einer oder mehreren Achsen messen und so über das Erdmagnetfeld die Ausrichtung im Raum ermitteln. Sie können auch als Mikrosensoren ausgeführt werden und basieren auf dem magnetoresistiven Effekt. Dabei ändert sich der Widerstand des Materials (je nach Typ verschieden, NiFe oder CuCo) je nach Winkel und Stärke der Flussdichte durch das Sensorelement[35].

[31] Micro-Electro-Mechanical-System
[32] vgl. [38].
[33] vgl. [43], S. 236.
[34] vgl. [16], S. 147f.
[35] vgl. [16], S. 146f.

3.4.4 9-Achsen-Sensor BNO055

Der Sensor BNO055 des Herstellers Bosch Sensortec ist eine inertiale Messeinheit die ein 3-Achsen-Gyroskop, einen 3-Achsen-Beschleunigungssensor und einen 3-Achsen-Magnetfeldsensor – daher 9-Achsen – beinhaltet. Außerdem ist ein Arm Cortex M0+ Mikrokontroller integriert, der die Messwerte der Einzelsensoren in eine absolute Lage im Raum umrechnet.

Der Sensor kalibriert sich laufend selbst und gibt über I^2C oder UART (siehe 5) die Sensorwerte und die berechneten Lagewerte mit einer Frequenz von 100 Hz aus[36].

[36] vgl. [12].

4 Haptisches Feedback in Robotik-Systemen

Der Mensch ist daran gewöhnt, ununterbrochen verschiedenste Sinneseindrücke – Feedback – wahrzunehmen. Vor allem beim Ertasten und Greifen von unterschiedlichsten Gegenständen mit der Hand werden zahlreiche Informationen des betroffenen Gegenstands aufgenommen. So lässt sich zum Beispiel beim blinden Ertasten klar ein harter Gegenstand von einem weicheren unterscheiden. Auch Oberflächen, Feuchtigkeit und viele weitere Eigenschaften werden völlig unbewusst vom menschlichen Körper wahrgenommen.

Durch immer fortschrittlicher werdende Robotersysteme, ist es bereits möglich, hochkomplexe Anwendungen (z.B. Operationen, bei denen der Chirurg mittels eines hochpräzisen Roboters operiert, siehe 4.3) durch verschiedenste Arten einer Fernsteuerung zu betreiben.

Ohne der Implementierung von imitierenden Feedback-Funktionen, würden keine der eben genannten Sinneseindrücke an den Bediener des Robotersystems weitergegeben werden, wodurch selbst die einfachsten Anwendungen zu großen Problemen führen würden.

4.1 Definition von Haptik

Die Haptik setzt sich im Wesentlichen aus zwei Teilbereichen zusammen: der „taktilen“ und der „kinästhetischen“ Wahrnehmung.

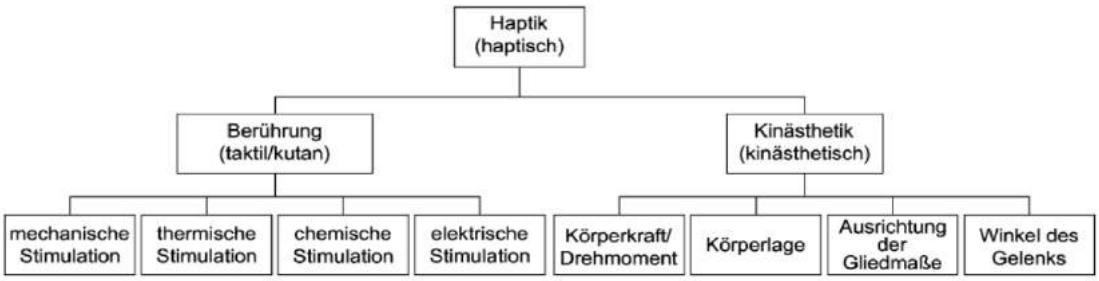


Abbildung III.16: Zusammensetzung der Haptik[23]

Während es sich bei der taktilen Wahrnehmung primär um das Empfinden der durch reine Berührung verursachten Eindrücke – also der Oberflächencharakteristik – handelt, beschäftigt sich der Teilbereich der Kinästhetik mit der Empfindung von Körperteilen im Verhältnis zueinander[37].

Die in Abbildung III.16 ersichtlichen Unterkategorien der beiden Teilbereiche werden in dieser Arbeit nicht weiter behandelt.

4.2 Bereitstellung von haptischem Feedback

Um taktiles sowie kinästhetisches Feedback bereitstellen zu können, sind verschiedenste Umsetzungsmöglichkeiten denkbar, von denen einige im Folgenden erläutert werden.

[37] vgl. [48], 2.1 Haptische Wahrnehmung.

4.2.1 Taktiles Feedback

hydraulisch Eine Möglichkeit für die Umsetzung ist die Anwendung von taktilen Displays, welche durch ein geschlossenes hydraulisches System mittels eines Servomotors betrieben werden. Das hydraulische System wird mit Wasser gefüllt, welches durch die Aktivierung des Servomotors in Richtung eines beweglichen Bolzen geführt wird, welcher dann gegen die Fingerspitze gedrückt wird. Das Funktionsprinzip wird grafisch in Abbildung III.17 dargestellt[38].

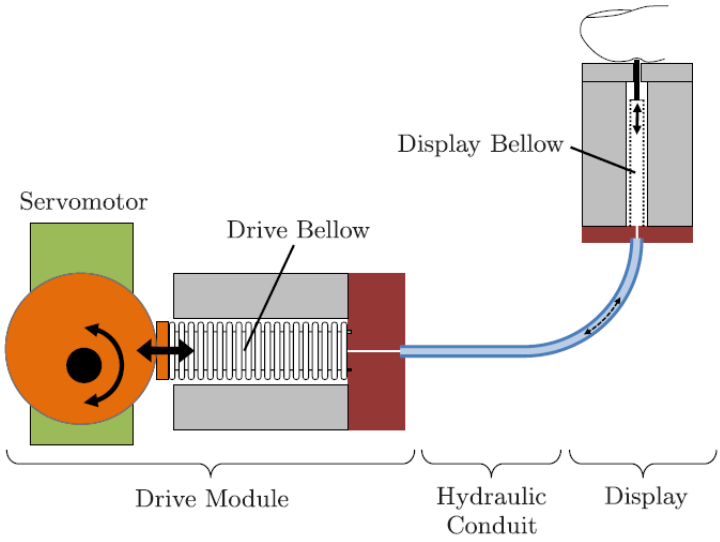


Abbildung III.17: Taktiles Feedback durch Hydraulik[29]

Vibrotaktil Eine weitere – in der Umsetzung weitaus einfachere – Möglichkeit für die Bereitstellung des taktilen Feedbacks erfolgt mittels Aktoren, welche Vibrationen abgeben.

Um deutlich spürbare Vibrationen zu erzeugen, wird ein unwuchtiger Schwungkörper durch einen Motor in eine Kreisbewegung versetzt[39].

Abbildung III.18: Taktiles Feedback mit Vibrationsmotor[53]

[38] vlg. [29], PrototypeDesign.
[39] vgl. [52], 3.2.1. Vibrotaktiles Feedback.

Elektrostimulation Der letzte in dieser Arbeit vorgestellte Ansatz zur Realisierung eines taktilen Feedbacks ist die Verwendung von elektrischen Stimulationen. Dabei werden hauptsächlich portable EMS[40]-Geräte verwendet, welche kleine Ströme durch den menschlichen Körper leiten, wodurch die betroffenen Muskeln stimuliert werden. Als Kontaktierung zwischen dem Gerät und der Haut dienen meist Elektroden an Kleidungsstücken wie Armbänder und Westen.

Abbildung III.19: EMS-Gürtel zum Muskelaufbau[10]

Ein beträchtlicher Vorteil im Vergleich zu den Varianten aus Abschnitt 4.2.1 sowie 4.2.1 liegt darin, dass der Stromverbrauch und der Geräuschpegel deutlich niedriger sind[41].

4.2.2 Kinästhetisches Feedback[42]

Anders als beim taktilen Feedback, wird in diesem Abschnitt nur auf eine konkrete Umsetzung eines kinästhetischen Feedbacks eingegangen, da diese essenziell für spätere Erläuterungen im Kapitel *Fernsteuergerät* Abschnitt 3.2 ist.

In dem von der ETH-Zürich stammenden Projekt „DextrES“ – kurz für dexterous electrostatic brake; also geschickte elektrostatische Blockade – wird ein kinästhetisches Feedback realisiert, welches in Form eines tragbaren Handschuhs zum Einsatz kommt. Mit besagtem Projekt ist es möglich, die menschliche Hand in der Schließbewegung – also das Anwinkeln der Finger – zu hindern, wodurch durch eine Integration in VR-Umgebungen bisherige Feedback-Möglichkeiten deutlich erweitert werden können.

Funktionsprinzip Das kinästhetische Feedback erfolgt grundsätzlich über zwei Stahlfolienstreifen, von denen einer – der sogenannte „hand-strip“ – fest am Handrücken montiert ist. Der andere Streifen – der sogenannte „finger-strip“ ist an der Fingerspitze montiert und wird mittels 3D-gedruckten Führungselementen planar mit dem hand-strip überlappt geführt.

[40] elektrische Muskelstimulation
[41] vgl. [52], 3.2.2. Elektrostimulation.
[42] vgl. [30].

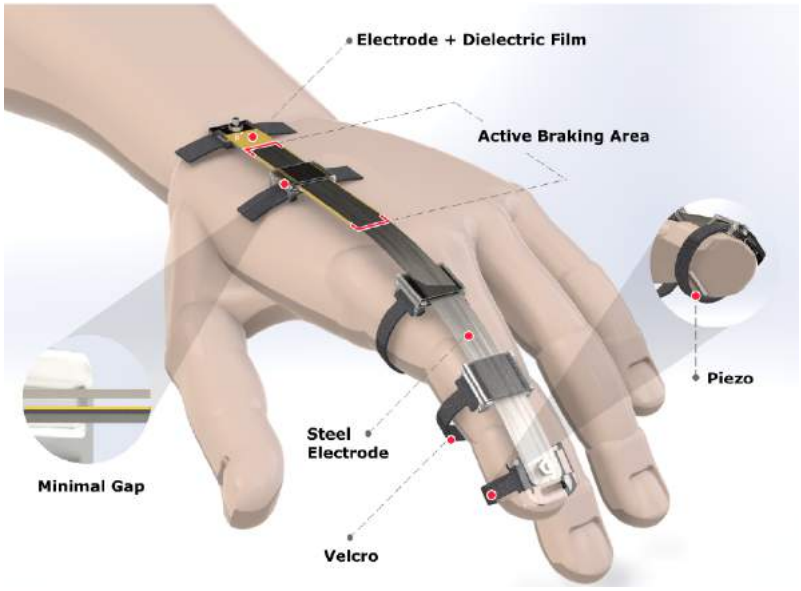


Abbildung III.20: DextrES Komponenten-Darstellung[30]

Zwischen den beiden Strips befindet sich eine elektrisch-isolierende Polyamide-Folie mit hoher elektrischer Durchschlagsfestigkeit, welche als Dielektrikum wirkt. Somit bilden die zwei Stahlstreifen zusammen mit der Folie elektrisch gesehen einen Kondensator mit der Kapazität C_{Strip}, wobei es sich bei den Stahlstreifen um die Elektroden handelt.

Für folgende Formelzeichen gilt:

- Elektrische Feldkonstante: ϵ_0
- relative Dielektrizitätszahl: ϵ_r
- überlappende Elektrodenfläche: A
- Elektrodenabstand: d
- Elektrodenspannung: U

Für die Kapazität des Kondensators gilt:

$$C_{Strip} = \frac{\epsilon_0 * \epsilon_r * A}{d} \tag{III.4}$$

Durch eine Spannung an den Elektroden folgt eine Anziehungskraft $F_{compression}$, welche die beiden Stahlstreifen aneinander zieht. Diese berechnet sich mit:

$$F_{compression} = \frac{\epsilon_0 * \epsilon_r * A * U^2}{2 * d^2} \tag{III.5}$$

Das verwendete Dielektrikum weist eine relative Dielektrizitätszahl $\epsilon_r = 3,4$ auf und hat eine Dicke von $d = 13\mu$m. Dieses wird mittels doppelseitigem Klebeband an einem der beiden Strips festgeklebt, welches elektrisch leitende Eigenschaften besitzt, um den Elektrodenabstand nicht weiter zu erhöhen.

Aus der Kompressionskraft zwischen den Strips bildet sich proportional zum Reibungskoeffizienten der Folie μ eine Reibungskraft $F_{friction}$, welche teilweise oder vollständig die Fingerbewegung blockiert.

$$F_{friction} \leq \mu * F_{compression} \tag{III.6}$$

Somit herrscht diese Kraft nur dann, wenn eine ausreichend hohe Spannung an den Elektroden anliegt, wodurch die ES-Brake aktiviert wird. Trennen von der Spannung deaktiviert die Brake.

Für die Fertigung der Strips wurden 18 cm lange und 1 cm breite Stücke mittels Laser-Cutter von der 100 μm dünnen Folie abgetrennt.

In Abbildung III.21 ist die Funktion von DextrES nochmals grafisch dargestellt.

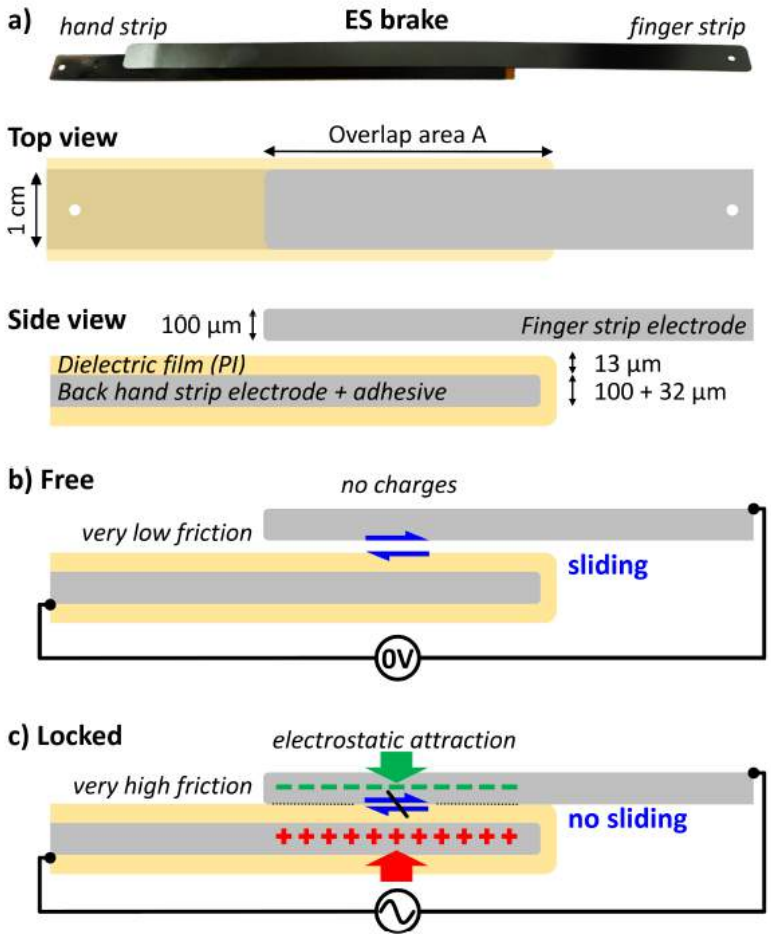


Abbildung III.21: Grafische Darstellung des Funktionsprinzip von DextrES [30]

Durch die vorgegebenen Materialeigenschaften kann nun eine ausreichend hohe Reibungskraft nur über den Wert der Spannung erreicht werden, welcher im Bereich von 1500 V liegt. Als Spannungsart wurde dabei eine bipolare Rechteckspannung gewählt, welche von einem 5 V zu 1500 V DC-DC-Wandler stammt und mittels einer H-Brücke in besagtes bipolare Signal umgeformt wird.

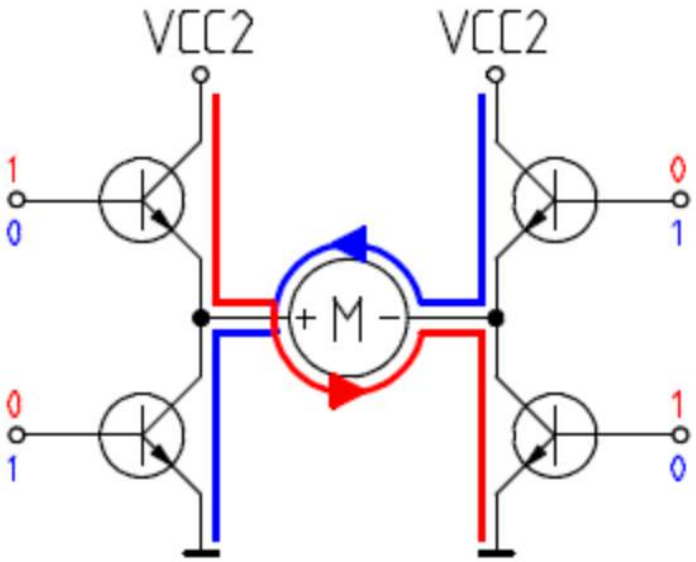


Abbildung III.22: Erzeugen einer bipolaren Rechteckspannung mittels H-Brücke[27]

Durch verschiedene Taktungen der vier enthaltenen Schaltelemente können Frequenzen realisiert werden, mit denen der Strom periodisch seine Richtung ändert.

Resultate Das Forschungsteam der ETH-Zürich konnte nach Messungen an ihrem Produkt verschiedenste Reibungskräfte in Abhängigkeit der angelegten Spannung feststellen. Die überlappende Fläche der Elektroden betrug dabei stets $11cm^2$. Die Messung wurde unter einer Spannungsfrequenz von 10 Hz durchgeführt.

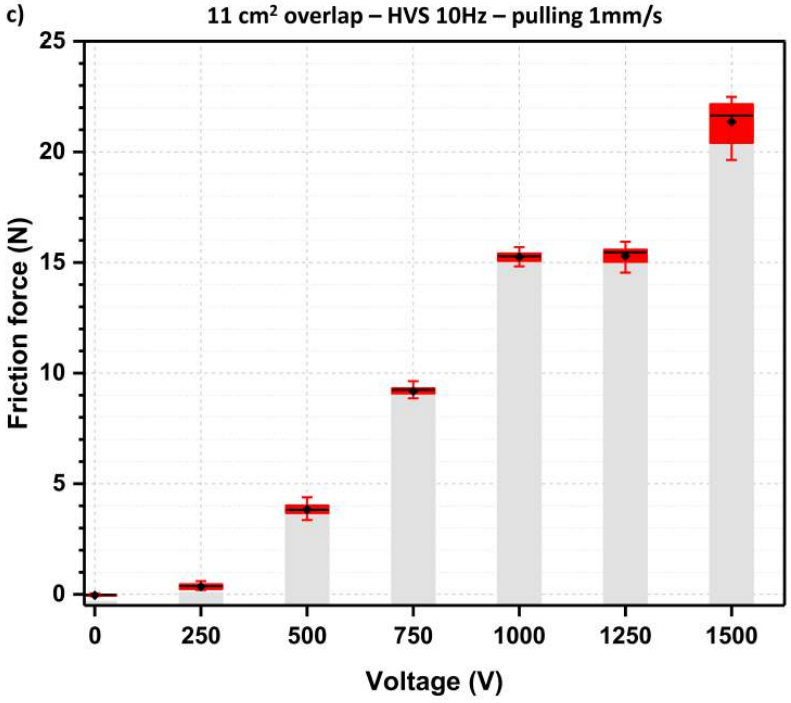


Abbildung III.23: Resultierende Reibungskraft bei verschiedenen Spannungen [30]

Aus Abbildung III.23 ist zu entnehmen, dass die resultierende Reibungskraft bis zu einer Spannung von 750 V deutlich unter 10 Newton liegt. Erst ab 1000 V gelingt es, eine ausreichende hohe – für das Feedback nötige – Kraft hervorzurufen. Die besten Resultate mit der höchsten Reibungskraft wurden mit einer Spannung von 1500 V erreicht.[43]

4.3 Aktuelle Anwendungsbereiche

Die Verwendung ferngesteuerter Robotersysteme – sogenannte Teleoperation-Roboter – findet in immer mehr Bereichen eine sinnvolle Anwendung. Aber nicht nur bei der Teleoperation, sondern auch in andern Teilgebieten der Robotik sind Feedback-Funktionen bereits ein wesentlicher Bestandteil. Im Folgenden wird ein kurzer Auszug verschiedener Anwendungsbereiche vorgestellt.

Chirurgie Wie bereits zu Beginn des Kapitels 4 als Beispiel angeführt, haben ferngesteuerte Roboter in der Chirurgie bereits einen immer höher werdenden Stellenwert. Dabei handelt es sich um eine Roboter-assistierte Chirurgie[44],die vor allem im Bereich der Onkologie und der Kopf-Hals-Chirurgie zunehmend Einzug hält. Zwar sind bereits zwei Modelle – die Systeme „DaVinci“ und „FLEX“ – klinisch zugelassen, allerdings haben sich diese außerhalb des asiatischen Raumes noch nicht vollständig durchsetzen können. Gründe dafür stellen eine aufwendige Handhabung der Systeme und finanzieller Mehraufwand dar. Außerdem sind derzeit haptische Feedback-Funktionen nur im kleinen Ausmaß integriert, welche allerdings teilweise von 3D-Bildgebung kompensiert werden können[45].

(a) DaVinci Patientenwagen (b) Bedien-Konsole

Abbildung III.24: DaVinci RAC-System[17]

[43] [30], vgl.
[44] kurz RAC
[45] vgl. [54].

Der Chirurg bedient hierbei den Roboter (Patientenwagen) aus Abbildung III.24a mittels der Bedienkonsole, von der aus er eine 3D-Bildgebung der Anatomie in HD-Auflösung betrachten kann[46].

In Abbildung III.24b ist dargestellt, wie sich die Nutzungszahlen im Zeitraum 2011 bis 2016 entwickelten. So stieg die Nutzung von DaVinci-Systemen in den USA in diesem Zeitraum von 300.000 auf knapp 600.000 Fallzahlen, wodurch sich diese – wie auch der in Fallzahlen niedrigere weltweite Anteil – fast verdoppelte. Die weltweiten Fallzahlen lagen 2016 bei knapp 200.000.

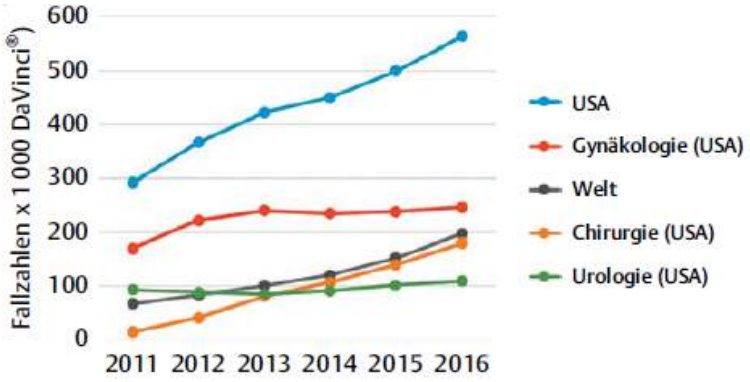


Abbildung III.25: Nutzungszahlen des DaVinci-Systems[54]

Telepräsenz im Weltraum[47]

Neben der im vorherigen Abschnitt vorgestellten Methode der Teleoperation, besteht zusätzlich die Möglichkeit, das System auf eine Telepräsenz auszuweiten. Das bedeutet, dass der Roboter zum einen ferngesteuert wird, und sich zum anderen der Bediener nicht am selben Ort befinden muss, sondern mittels virtual-reality visuell an den Arbeitsplatz des Roboters versetzt wird.

Das Projekt „Robonaut 1“ der NASA beschäftigt sich mit einem hochkomplexen Robotersystem (siehe Abbildung III.26), bei dem der menschliche Bediener mittels Telepräsenz 43 individuelle Freiheitsgrade des Roboters steuern soll.

[46] vgl. [18], da Vinci Systems.
[47] vgl. [46].

Abbildung III.26: Robonaut[46]

Der Bediener steuert dabei die Bewegungen der Arme, Finger sowie des Kopfs mittels intuitiven Bewegungen, welche durch das Master-Slave-Prinzip vom Robonauten ausgeführt werden. Um die hohe Anzahl der Bewegungen des Masters – des Menschen – in den Freiheitsgraden möglichst originalgetreu auf den Slave – den Roboter – übertragen zu können, werden verschiedene Verfahren angewendet, um die Bewegungen zu messen, also zu „tracken“. Die Fingerbewegung wird dabei durch Biege-sensitive Sensoren in einem Handschuh gemessen, welchen der Bediener ständig trägt. Das Tracking der Arme und des Kopfes erfolgt über Magnet-basierte Positions- und Orientierungssensoren.

Durch die in den Händen des Robonauten integrierten Kraft Sensoren kann mittels eines mechanischen Exoskeletts haptisches Feedback an den Bediener übermittelt werden.

Ziel des Projekts ist weiters, dass in Zukunft die notwendigen Arbeiten im Weltraum durch besser erforschte Methoden der Telepräsenz bei deutlich erhöhtem Sicherheitsfaktor und Leistung ausgeführt werden können, bei denen sich der menschliche Bediener – der Teleoperator – nicht selbst in Gefahr begeben muss.[48]

[48] vgl. [46].

5 Bussysteme und Kommunikationsschnittstellen

Da das Projekt aus zahlreichen eigenständigen Komponenten besteht, die jedoch miteinander kommunizieren müssen, wird in diesem Abschnitt ein genereller Überblick über die verwendeten Schnittstellen und Bussysteme gegeben. Ein Bussystem ist eine Kommunikationsschnittstelle, die mehr als zwei Kommunikationsteilnehmer miteinander verbindet. Dabei nutzen alle Geräte die gleiche Leitung, wodurch durch geringen Verdrahtungsaufwand viele Geräte miteinander verbunden werden können[49].

5.1 UART und RS232

UART und RS232 sind keine Bussysteme, sondern vielmehr eine Punkt-zu-Punkt-Verbindung zwischen zwei Systemen.

UART bedeutet „Universal Asynchronous Receiver/Transmitter“, also ein universelles Empfangs- und Sendemodul für asynchrone Datenübertragung, das zur Standardausstattung der meisten Mikrocontroller gehört. Asynchron bedeutet, dass kein Taktsignal zwischen den Systemen übertragen wird, sondern beide Systeme unabhängig ein Taktsignal generieren müssen. UART überträgt Daten in Paketen, die 5 bis 9 Bit lang sind, üblich sind 8 Bit (1 Byte). Außerdem kann noch ein optionales „Parity-Bit“ übertragen werden, das der Fehlererkennung dient. Auf die genauen Details der Signalverläufe und Datenübertragung wird hier nicht eingegangen, da diese von der Hardware der verwendeten Mikrocontroller übernommen wird[50].

RS232 ist die bekannteste Schnittstelle auf Basis von UART. Diese war früher weit verbreitet als universelle Schnittstelle zwischen PC und Peripheriegeräten und findet heute noch in der Steuerungstechnik Anwendung. RS232 verwendet negative Logik. Dabei wird eine logische Null durch eine Spannung zwischen +3 und +15 V dargestellt, während eine logische Eins von einem Pegel zwischen -3 und -15 V repräsentiert wird. Da aber die UART-Module der Mikrocontroller positive Logik und den internen Spannungspegel (meist 3,3 V oder 5 V) verwenden, müssen die Signale zuerst angepasst werden[51].

5.2 Serial Peripheral Interface (SPI)

Ein wichtiges Bussystem in der Mikrocontroller-Technik ist das Serial Peripheral Interface. Dieses ist ein synchrones Bussystem, das heißt es wird ein Taktsignal übertragen. SPI verwendet das Master-Slave-Prinzip, dabei gibt es am Bus einen Master, der für die Erzeugung des Taktsignals zuständig ist und mehrere Slaves, die vom Master durch ein Signal aktiviert werden können. Für die Kommunikation werden grundsätzlich drei Datenleitungen benötigt: MISO (Master In Slave Out, Datenleitung vom Slave zum Master), MOSI (Master Out Slave In, Datenleitung vom Master zum Slave) und SCK (Serial Clock, Taktsignal). Für jeden Slave am Bus wird außerdem ein eigenes „Slave Select“-Signal (SS) benötigt, ist dieses auf einem Low-Pegel ist der Slave aktiviert und reagiert auf Datenübertragung, andernfalls nicht.

Wenn nun Daten übertragen werden sollen, aktiviert der Master den gewünschten Slave über die entsprechende Slave-Select-Leitung. Mit jeder Taktflanke wird jetzt gleichzeitig ein Bit vom Master zum Slave und ein Bit vom Slave zum Master gesendet, wodurch nach 8 Takten ein ganzes Byte übertragen worden ist. Durch die gleichzeitige Datenübertragung

[49] vgl. [43].
[50] vgl. [43], S. 152ff.
[51] vgl. [28], S. 484ff.

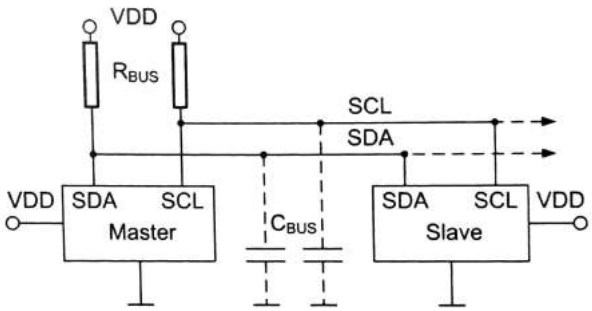


Abbildung III.27: Schematische Darstellung I²C-Bus

ist es manchmal notwendig, das der Master oft weitere Daten sendet (deren Inhalt nicht relevant ist), um dem Slave zu ermöglichen, Daten zurück an den Master zu schicken[52].

5.3 Inter-Integrated-Circuit (I²C)

I²C ist ein von Philips entwickeltes Bussystem, das für die Verbindung von Mikrocontrollern mit Peripheriekomponenten wie Analog-Digital-Wandlern, Speicherbausteinen und Port-Erweiterungen verwendet wird. Ähnlich wie SPI ist I²C ein synchrones Bussystem und baut auch auf dem Master-Slave-Prinzip auf, benötigt allerdings nur zwei Datenleitungen: SDA („Serial Data“) und SCL („Serial Clock“). Im Normalfall gibt es auf einem I²C-Bus einen Master und einen oder mehrere Slaves.

Elektrisch sind die Anschlüsse eines I²C-Gerätes als Open-Drain-Ausgänge realisiert. Da die Leitungen SDA und SCL mit Pull-Up-Widerständen versehen sind, liegen die Busleitungen im Ruhezustand auf einem High-Pegel, was einer logischen Eins entspricht. Eine logische Null liegt am Bus an, wenn eines der Geräte die Leitung auf den Low-Pegel zieht. Die Datenübertragung in I²C ist immer Byte-basiert, es werden also 8 Bit hintereinander übertragen.

Die Kommunikation läuft nun nach folgendem Schema ab: Der Master initiiert die Kommunikation mit einem Startsignal (genannt Startbedingung) und ist dafür zuständig, das Taktsignal (SCL) zu generieren. Danach wird eine 7 Bit lange Adresse übertragen, die auswählt, welcher Slave angesprochen werden soll, gefolgt von einem Bit das angibt ob Daten im Slave geschreiben (0) oder gelesen (1) werden soll. Der angesprochene Slave antwortet daraufhin mit einem „Acknowledge“. Sollen nun Daten geschrieben werden, wird noch ein oder mehrere Bytes übertragen. Werden hingegen Daten gelesen, kehrt sich die Datenrichtung der Datenleitung (SDA) um, und der Slave sendet ein oder mehrere Bytes an den Master. Die Übertragung läuft nur, solange der Master ein Taktsignal auf der SCL Leitung zur Verfügung stellt, der Slave kann nicht von sich aus Daten übertragen[53].

5.3.1 Registerbasierte Kommunikation

Aufbauend auf diesem Protokoll verwenden die meisten I²C-Bausteine eine auf Registern basierende Kommunikation. Bei den in diesem Projekt eingesetzten Geräten werden die Details der Registeransteuerung von Bibliotheken übernommen, allerdings wird hier

[52] vgl. [43], S. 160ff.
[53] vgl. [28], S. 484ff.

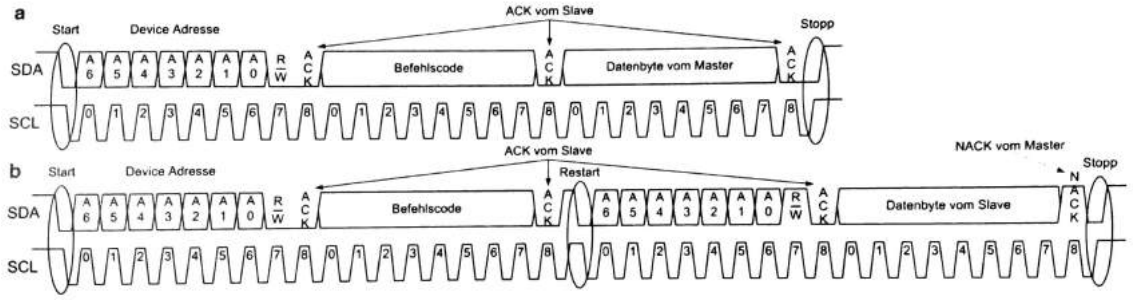


Abbildung III.28: Lese- und Schreibvorgang bei registerbasierter I²C-Kommunikation

trotzdem näher darauf eingegangen, weil für das Projekt ein solcher I²C-Slave selbst implementiert wurde (siehe Kapitel *Software* Abschnitt 5.7).

Dabei definiert der Slave mehrere Register, die meist eine ein Byte lange Adresse besitzen. Will nun der Master ein Register ansprechen, beginnt er nun wie oben beschrieben die Kommunikation, indem er den Slave adressiert und das Read / Write-Bit auf null setzt (Schreibzugriff). Als nächstes Byte überträgt der Master die Adresse des Registers (auch Befehlscode genannt).

Soll jetzt auf das Register geschrieben werden, setzt der Master die Kommunikation einfach fort (da sich der Bus bereits im Schreib-Modus befindet) und überträgt als nächstes die Daten, die ins Register geschrieben werden sollen.

Ist allerdings ein Lesezugriff auf das Register erwünscht, muss der Master ein Restart-Signal senden. Danach wird der Slave erneut adressiert, jetzt allerdings mit dem Read / Write-Bit gesetzt (Lesezugriff). Dadurch kehrt sich die Datenrichtung am Bus um und der Slave sendet jetzt den Wert des Registers mit der zuvor übertragenen Adresse[54].

5.4 CAN-Bus

Der CAN-Bus (Controlled Area Network) ist aufgrund seiner Einfachheit, seiner hohen Datenraten und seiner Robustheit besonders in der Automobilbranche beliebt. In den letzten Jahren etablierte sich das Bussystem auch in der Automatisierungstechnik. Der CAN-Bus wurde erstmals unter der ISO 11898 Norm von Bosch in Teilen 1 bis 6 aufgeteilt. Der erste Teil gibt die Spezifikation an. Der High- und Low- Speed CAN wird von den Teilen 2 und 3 festgelegt. Teil 4 legt den sogenannten „Time Triggered CAN“ fest. Der low-power mode und ein selektiver Wakeup wird von den letzten beiden Teilen festgelegt. Grundsätzlich wird der CAN-Bus als symmetrische Zweidrahtleitung in Wired-AND-Technik ausgelegt, man kann ihn aber auch als Eindrahtleitung spezifizieren.

Um beim Senden Kollisionen zu vermeiden, wird die CSMA/CA Technik verwendet. Dies bedeutet, dass alle Teilnehmer ruhig sind, solange einer sendet, da alle Teilnehmer immer „mithören“. Senden jedoch zwei Teilnehmer gleichzeitig ein Signal, wird derjenige Teilnehmer das Problem zuerst bemerken, wenn er eine logische 1 gesendet hat, obwohl am Bus ein dominantes Signal 0 anliegt.

Weiters verfügt das Bussystem über eine automatische Fehlererkennung, welches dem CAN-Bus ermöglicht fehlerhafte Teilnehmer abzuschalten[55].

[54] vgl. [43], S. 182ff.
[55] vgl. [43], S. 186ff.

5.5 Bluetooth Low Energy

Für das Projekt wurde eine geeignete drahtlose Kommunikationsschnittstelle benötigt, wobei die Wahl auf Bluetooth Low Energy fiel. Bluetooth Low Energy (BLE) ist seit Version 4.0 Teil der Bluetooth Spezifikation. BLE hat im Vergleich zum sogenannten „Bluetooth Classic“ einen geringeren Energieverbrauch und ist damit ideal für die Verwendung in kleinen Sensor- und Aktor-Systemen. Außerdem hat es geringere Latenzzeiten, ist allerdings nicht für hohe Datenraten optimiert, sondern für sporadische Datenpakete, die möglichst schnell und sicher übertragen werden sollen[56].

Die physikalische Übertragung von Bluetooth Low Energy besitzt folgende Eckdaten:

- Kommunikation im lizenzfreien 2,4 GHz Frequenzband
- Datenraten von bis zu 2 Mbit/s

5.5.1 Architektur

Bluetooth Low Energy arbeitet nach dem Client-Server-Modell. Der Server (auch „Peripheral“ genannt) ist meist der Sensor oder Aktor und stellt die Daten zur Verfügung. Der Client (auch „Central“ genannt) sucht nach verfügbaren Peripherals und verbindet sich mit dem gewünschten Sensor.

Dieser Datenaustausch findet über das „Generic Access Protocol“ (GAP) und das „Generic Attribute Protocol“ (GATT) statt. Letzteres ist dafür zuständig, die Struktur der Daten, die das Peripheral unterstützt, zu definieren, und sie zu sogenannten „Services“ zusammenzufassen. Jedem GATT-Service wird eine eindeutige Nummer zugewiesen. Der Service besteht dann aus mehreren „Characteristics“, welche die Struktur der Daten repräsentieren. Jede Characteristic hat ebenfalls eine eindeutige Nummer, einen zugeordneten Datentyp und kann mehrere Operationen unterstützen, unter anderem *Read* (Lesen), *Write* (Schreiben) oder *Notify* (Benachrichtigung bei einem neuen Wert). Ein Beispiel für einen solchen Service folgt in Abschnitt 5.5.2.

Über GAP findet das sogenannte Advertising statt, wodurch die von einem Gerät unterstützen GATT-Services an alle anfragendenen Clients (=„Central“-Geräte) übermittelt werden[57].

5.5.2 Nordic UART Service

In diesem Projekt wird der Nordic UART Service eingesetzt. Dieses ist ein GATT-Service, der vom Hersteller Nordic Semiconductor standardisiert wurde, und zum emulieren einer seriellen Schnittstelle über Bluteooth Low Energy verwendet wird.

Der Service hat die ID `6E400001-B5A3-F393-E0A9-E50E24DCCA9E` und besitzt zwei Characteristics:

Auf die TX-Characteristic (`6E400002-B5A3-F393-E0A9-E50E24DCCA9E`) kann geschrieben werden, womit Daten an das Peripherie-Gerät gesendet werden.

Mit der RX-Characteristic (`6E400003-B5A3-F393-E0A9-E50E24DCCA9E`) werden Daten vom Peripherie-Gerät empfangen. Die Characteristic unterstützt „Read“ (Lesen) und „Notify“ (Benachrichtigung bei neuen Daten)[58].

[56] vgl. [2].
[57] vgl. [11].
[58] vgl. [49].

6 Steuerungshardware

Um die Steuerung von Abläufen, Auswertung von Sensoren und Ansteuerung von Aktoren durch Software zu realisieren gibt es mehrere mögliche Hardware-Platformen, die das ermöglichen. Hauptsächlich – wie auch in unserem Projekt – werden Mikrocontroller und speicherprogrammierbare Steuerungen eingesetzt, über die im Folgenden ein kurzer Überblick gegeben wird.

6.1 Mikrocontroller

Ein Mikrocontroller ist ein IC[59], der einen Befehlsprozessor (CPU) und damit verbundene Peripheriekomponenten beinhaltet. Diese Peripheriekomponenten können unter anderem Speicher, Ein- und Ausgangsports, Analog-Digital-Wandler und Kommunikationsschnittstellen wie UART sein. Mikrocontroller werden meist mit C oder C++ programmiert und haben üblicherweise kein Betriebssystem.

Im Projekt werden drei verschiedene Typen von Mikrocontrollern eingesetzt, auf diese soll in den folgenden Abschnitten kurz eingegangen werden[60].

6.1.1 ATmega328P

Der ATmega328P gehört der AVR 8-Bit-Mikrocontroller-Familie an. Der Mikrocontroller besitzt Schnittstellen wie UART, SPI und I^2C, ist sonst aber Recht einfach aufgebaut. Im Vergleich zu den beiden anderen eingesetzten Mikrocontrollern ist der ATmega328P mit einem maximalen Takt von 20 MHz sehr leistungsschwach[61].

6.1.2 ESP32

Der ESP32 ist ein Mikrocontroller des Herstellers Espressif Systems. Der Chip vereint eine Wifi und Bluetooth Schnittstelle mit dem Xtensa dual-core 32-bit LX6 Mikroprozessor, der einen Takt von 240 MHz erreicht. Der ESP32 besitzt zahlreiche Schnittstellen die auf beliebige Pins verbunden werden können, darunter auch UART, SPI und I^2C[62].

Der ESP32 wird oft – wie auch in diesem Projekt – in Form eines Development-Boards genutzt, wodurch die Beschaltung des Prozessors wegfällt. Das verwendete Entwicklungsboard fügt sonst jedoch keine Funktionen hinzu.

6.1.3 Adafruit Feather M0 Bluefruit LE

Der Adafruit Feather M0 Bluefruit LE ist ein Entwicklungs-Board des Unternehmens Adafruit. Darauf befindet sich als Hauptprozessor ein ARM Cortex M0+ Prozessor (ATSAMD21G18), der in der Leistungsfähigkeit zwischen ESP32 und ATmega328P angesiedelt ist. Außerdem ist auf dem Entwicklung-Board ein nRF51822 Bluetooth Low Energy Modul und ein Laderegler für LiPo-Akkus verbaut, was es ideal für den Einsatz im Fernsteuergerät macht[63].

[59] Integerated Circuit; Integrierter Schaltkreis
[60] vgl. [31].
[61] vgl. [45].
[62] vgl. [24].
[63] vgl. [26].

6.1.4 Arduino-Framework

Um die Programmierung der drei Mikrocontroller zu vereinfachen, wurde das Arduino[64] Framework verwendet. Dieses abstrahiert die Details der darunterliegenden Hardware-Architektur, was den Programmieraufwand besonders bei komplexen Prozessoren wie dem ESP32 erheblich mindert. Außerdem sind dadurch Bibliotheken für Peripheriegeräte und Anwenderprogramme kompatibel zu mehreren Prozessoren, ohne den Programmcode ändern zu müssen.

6.2 Speicherprogrammierbare Steuerungen

Speicherprogrammierbare Steuerungen (SPS) sind ein wichtiger Bestandteil der heutigen Automatisierungstechnik. Mit ihnen können Steuer- und Regelungsaufgaben sowie Berechnungen ausgeführt werden. Eine SPS besteht grundsätzlich aus den gleichen Baugruppen wie jedes Informationsverarbeitungssystem. Anders als einfache Mikrocontroller sind sie meist modular aufgebaut, beispielsweise gibt es ein Modul, in dem das Programm abläuft (CPU) sowie mehrere Ein- und Ausgabemodule. Diese sind durch ein Bussystem oder Netzwerk miteinander verbunden.

Die Abarbeitung des Programms erfolgt zyklisch mit einer festen Zykluszeit. Das bedeutet, dass periodisch mit einem fixen Zeitintervall zuerst die Eingänge gelesen, dann das Steuerungsprogramm ausgeführt und anschließend die Ausgänge wieder geschrieben werden. Die Einhaltung der vorgegebenen Zeit überwacht der „Watchdog“, durch den die Steuerung bei Nichteinhalten der Zeitvorgabe in einen Fehlerzustand wechselt. Ein solches System nennt man auch deterministisch oder echtzeitfähig, da die Reaktionszeit auf ein Ereignis im Voraus bekannt ist.

[64] https://www.arduino.cc

7 3D-Druck

Additive Fertigung, auch Additive Manufacturing (kurz AM) oder 3D-Druck genannt, ist ein Fertigungsverfahren, bei dem ein virtuell vorliegendes 3D-Modell durch einen Schichtweisen, additiven Aufbau hergestellt wird. „Dabei werden Schichten in einer x-y-Ebene maschinell und werkzeuglos generiert und unter Verwendung einer z-Achse in der dritten Dimension miteinander verbunden.“ - so Adamek und Piwek. Ihren Ursprung findet die additive Fertigung im Prototypenbau, durch immer besser und billiger werdende Verfahren wird sie aber auch für die Einzel- bis Kleinserienproduktion interessant. Da bei der Produktion neuer Bauteile keine Spezialwerkzeuge oder Formen gefertigt werden müssen, können Spezialwünsche von Kunden kostengünstig und zeitnah umgesetzt werden. Auch im Bereich bionischer Optimierungsstrategien bringt AM einen großen Vorteil, da hochkomplexe Bauteile, die mit herkömmlichen Verfahren Teilweise unmöglich herzustellen wären, schnell und günstig erzeugt werden können[65].

Aufgrund der tragenden Rolle, die der 3D-Druck in unserem Projekt spielt, werden im folgenden Abschnitt einige wesentliche Druckverfahren und die verwendete Slicing-Software[66] vorgestellt.

7.1 Druckverfahren

Grundsätzlich werden die additiven Fertigungsverfahren in Feststoff- und Flüssigkeitsbasierte Verfahren unterteilt. Weitere Klassifikationen sind der Abb. III.29 zu entnehmen. Nachfolgend werden die heutzutage prominentesten Verfahren erläutert.

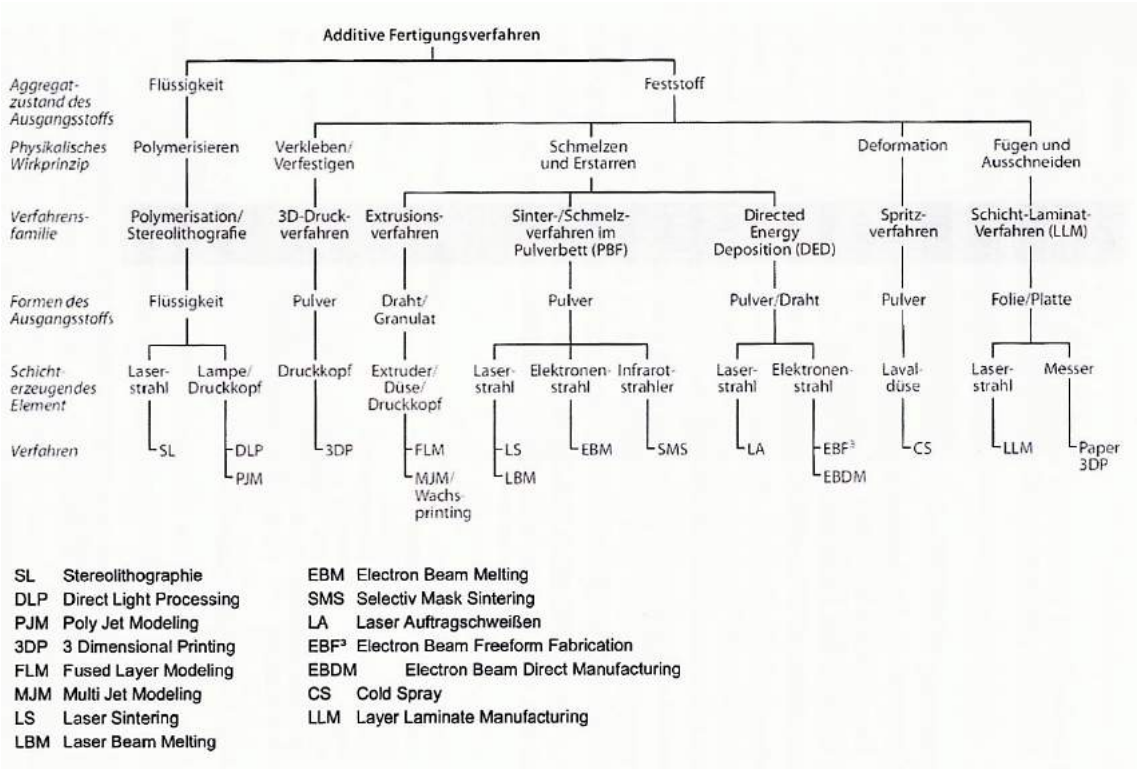


Abbildung III.29: Einteilung additiver Fertigungsverfahren[1]

[65] vgl. [1], S. 1.
[66] Software zum Erstellen von 3D-Druckanweisungen aus einem 3D-Modell

7.1.1 Fused Layer Modeling[67]

Fused Layer Modeling (kurz FLM), auch Fused Deposition Modeling (kurz FDM) genannt, gehört zu den Feststoff-basierten Extrusionsverfahren und wird auch von den 3D-Druckern der HTBLuVA-Salzburg verwendet. Über eine beheizte Düse, welche frei im Bauraum des Druckers verfahrbar ist, wird ein thermoplastisches Material erhitzt und linien- oder tröpfchenförmig auf die Bauplatte aufgetragen. Das zum Drucken verwendete Filament wird in Drahtform geliefert und über angetriebene Rollen durch Schläuche in den Druckkopf befördert, wo es erhitzt und verflüssigt wird. Das im Projekt verwendete Filament aus Polyactid (kurz PLA) ist eines der Standard-Materialien für FLM. Für das Fernsteuergerät und den Kabelschoner wurde ein flexibles Filament aus thermoplastischem Polyurethan[68] verwendet, welches auf mehr als seine dreifache Ausgangslänge dehnbar ist.

Mit FLM können nur unter Verwendung von Stützstrukturen Hohlräume oder Überhänge gedruckt werden. Ausßerdem ist die Oberfläche aufgrund des Extruderdurchmessers sehr ungenau, weshalb das Verfahren eher im Privatbereich und in Kleinunternehmen Anwendung findet. Zu diesem Zweck ist es aber, dank der niedrigen Anschaffungskosten von Drucker und Filament, sehr gut geeignet.

7.1.2 Laser-Stereolithographie[69]

Die Laser-Stereolithographie ist ein Flüssigkeits-basiertes Verfahren, welches das Prinzip der Polimerisation nutzt, um Bauteile zu fertigen. Polimerisation ist das (selektive) Bestrahlen von flüssigen, monomeren Harzen mit UV-Laserlicht, um diese zu festen Polymeren zu verbinden. Dabei wird die Kontur einer Schicht des Bauteils von einem UV-Laser auf die Oberfläche eines Harzbades gezeichnet. Sobald die Schicht fertig polimerisiert wurde, wird die Bauplatte um eine Schichtdicke abgesenkt und die nächste Schicht wird ausgehärtet. Dieser Prozess benötigt für Überhänge und Hohlräume zwar immer noch Stützstrukturen, allerdings entsteht dabei die detaillierteste Oberflächenstruktur aller additiven Fertigungsverfahren.

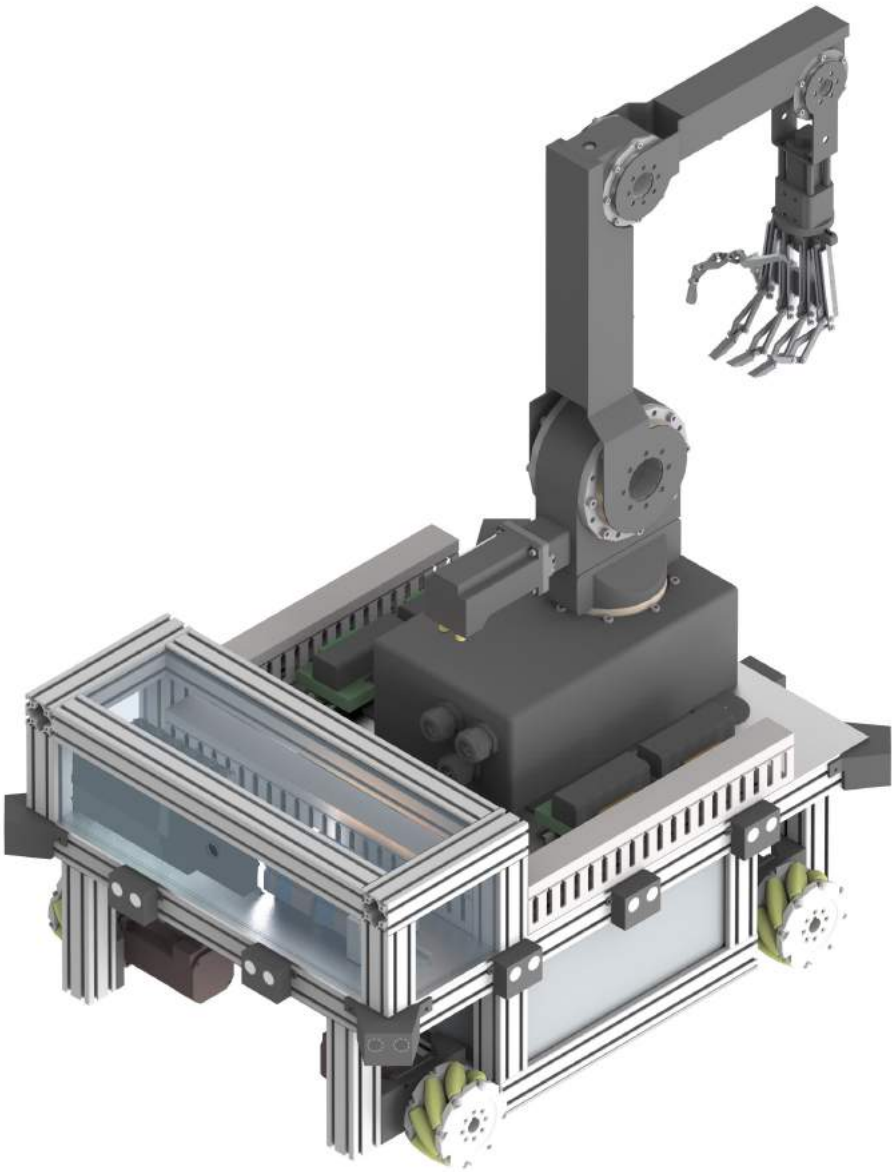
7.1.3 Selective Laser Sintering[70]

Das selektive Lasersintern (kurz SLS) ist ein Feststoff-basiertes Fertigungsverfahren, das pulverförmige Materialien, aus Partikeln von 20 bis 50 μm Durchmesser, mithilfe eines Lasers schmilzt. Nach dem Abkühlen und Erstarren des Materials entsteht eine feste Schicht. Die Bauplatte wird um eine Schichtdicke abgesenkt und eine neue Pulverschicht wird von einem Wischer verteilt. Das Verfahren benötigt beim Verarbeiten von Kunststoffen keine Stützstruktur, verfestigte Pulverreste in kleinen Hohlräumen bedeuten allerdings oft einen hohen Reinigungsaufwand. Auch die Fertigung metallischer Bauteile ist mit SLS möglich, hier werden Stützstrukturen aber nach wie vor benötigt.

Die Oberflächenqualität hängt bei dem SLS-Verfahren von dem Partikeldurchmesser ab und kommt somit an die Qualität konventioneller Fertigungsverfahren heran. Die mechanische Belastbarkeit von lasergesinterten Bauteilen ist, verglichen mit anderen Verfahren, bei Verwendung der selben Materialien, sehr gut. Durch diese Eigenschaften findet SLS bereits in der Medizintechnik zur Herstellung von Prothesen und Implantaten Anwendung.

[67] vgl. [1], S. 10ff.
[68] vgl. [51].
[69] vgl. [1], S. 5.
[70] vgl. [1], S. 6f.

IV Robotersystem



1 Übersicht

Im Kapitel Robotersystem werden mehrere Teilbereiche behandelt:

- das Gehäuse
- das Fahrwerk
- die SPS-Hardware
- der Robolink
- die Abstandssensorik

1.1 Anforderungen an das Gehäuses

Das Gehäuse dient in erster Linie einer platzsparenden und sinnvollen Unterbringung aller notwendigen Elemente des Gesamtsystems. Dabei wurde darauf geachtet, eine übersichtliche elektrische Installation zu gewährleisten, alle Komponenten räumlich nach Aufgabenbereichen zu gruppieren und ein möglichst ansehnliches Design zu entwickeln. Zusätzlich muss das Gehäuse mobil sein und darf sowohl elektrisch als auch durch seine Bewegung keine Gefahr für den Benutzer darstellen.

1.2 Aufgaben der SPS

Die speicherprogrammierbare Steuerung übernimmt die Steuerung des Roboterarms und des Fahrwerks. Dazu ist sie mit den Schrittmotormodulen und den ACOPOSmicro Wechselrichtermodulen verbunden.

1.3 Aufgaben des Fahrwerks

Das Antriebssystem sorgt dafür, dass das Gesamtsystem einfach zu seinem Einsatzort bewegt werden kann. Um auf einer Ebene jeden Punkt anfahren zu können, ohne über eine aufwendige Lenkung verfügen zu müssen, wurden sogenannte „Mecanum-Wheels“ eingesetzt.

1.4 Aufgaben des Robolink

Das „Robolink RL-DC - Komplettmodul mit 5 Freiheitsgraden“ der Firma igus wurde im Projekt als Roboterarm eingesetzt. Ausgestattet mit einem bionischen Greifer führt der Robolink die vom Bediener vorgegebenen Bewegungen des Arms und der Finger aus.

1.5 Aufgaben der Abstandssensorik

Aufgabe der Abstandssensorik ist es, Kollisionen des Gefährts mit Hindernissen und Personen zu vermeiden. Das Gehäuse ist zu diesem Zweck mit 14 Ultraschall-Sensoren ausgestattet. Befindet sich nun ein Hindernis zu nahe am Robotersystem, bleibt dieses stehen, bevor es zu einer Kollision kommen kann.

2 Gehäuse

2.1 Dimensionierung

Vor der Wahl aller Komponenten musste das Gesamtsystem grob dimensioniert werden, um die mechanischen Belastungen, sowie die notwendigen Hauptmaße des Systems abschätzen zu können.

2.1.1 Gewicht

Für die Wahl der Antriebsmotoren, der Räder und der verwendeten Materialien spielte das Gesamtgewicht eine entscheidende Rolle. In der folgenden Tabelle ist die Berechnung des Gesamtgewichts zu finden. Alle Werte wurden entweder aus den Datenblättern übernommen oder berechnet, bzw. geschätzt.

	Gewicht in kg
Roboterarm	21
Akku	13,6
Motoren	5,8
Omniwheels	3
Aluprofile	7
Grund- und Abdeckplatten	8
SPS	1,5
Elektroinstallation	3,1
Schrauben etc.	0,5
Rest	1,5
Gesamtgewicht	65

Tabelle IV.1: Berechnung des Gesamtgewichts

2.1.2 Hauptmaße

Die Hauptmaße richten sich nach dem Platzbedarf aller Komponenten des Roboter-Greifsystems bei einer sinnvollen räumlichen Aufteilung. Die Gesamtlänge beträgt 69 cm, die Breite des Systems 52 cm. Die Höhe des Gehäuses beträgt 35 cm. Genauere Details sind den Renderings, sowie den Zeichnungen im Anhang zu entnehmen.

2.2 Gerüst

Das Gehäuse besteht aus einem Gerüst aus Aluminium-Profilen. Die Profile sind ausreichend robust, um das Gewicht des Gesamtsystems zu tragen und verfügen gleichzeitig über ein äußerst geringes Eigengewicht. Des Weiteren ist deren Montage sehr unproblematisch.

Gewählt wurde ein Aluminium-Konstruktionsprofil[1] aus Al Mg Si 0,5 F 25 mit einer Dichte von 2,7 kg/dm^3. Es wurde eine Ausführung mit zwei Nuten pro Seite und einem Hohlraum im Kern verwendet, welcher das Gewicht drastisch reduziert und als Kabelkanal für die Abstandssensorik verwendet wird (siehe Abschnitt 6.4).

[1] vgl. [36].

Wichtig bei der Planung des Gerüsts war nicht nur eine sinnvoll nach Aufgaben gegliederte Platzierung aller Komponenten, sondern auch ein professionelles Aussehen. Zu diesem Zweck wurde der Akkuraum zwischen die Räder des Fahrwerks platziert und die SPS mit einem eigenen Gehäuse überdeckt. Die Räder wurden aus sicherheitstechnischen und ästhetischen Gründen so angebracht, dass sie seitlich nicht über das Gehäuse herausragen, ähnlich einem PKW.

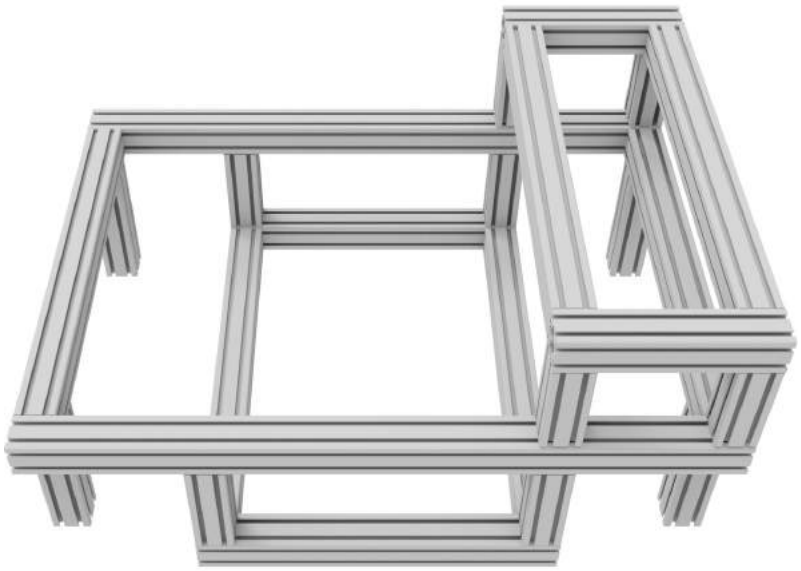


Abbildung IV.1: Gerüst aus Alu-Konstruktionsprofilen

2.2.1 Montage

Das Verbinden der Konstruktionsprofile erfolgt mittels Winkeln mit Zentriernasen, Nutensteinen mit Steg und Feder sowie Schrauben der Größe M4. Diese Elemente ermöglichen eine schnelle und verdrehsichere Montage. Da sich die Schrauben durch Vibrationen des Gestells im Fahrbetrieb lockern könnten, wurden zwischen Schraube und Winkel Fächerscheiben angebracht.

2.3 Abdeckungen

2.3.1 Bodenplatte

Die Bodenplatte befindet sich am tiefsten Punkt des Gehäuses und trägt den Akku und eine Hutschiene. Sie besteht aus 4 mm starkem Aluminiumblech und ist auf der Unterseite am Gerüst angebracht. Zur Isolierung des Akkus gegen das Gehäuse wurde unter dem Akku eine isolierende Hartgummi-Matte angebracht.

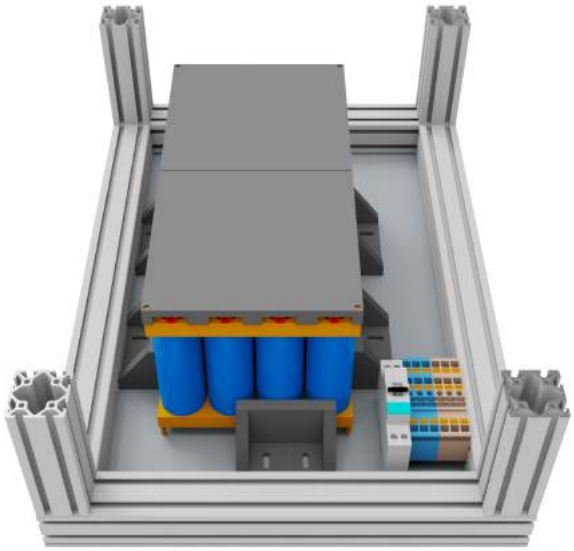


Abbildung IV.2: Bodenplatte mit Akku und Hutschiene

2.3.2 Seitenplatten

Die Seitenplatten bestehen aus 3 mm Alublech und dienen hauptsächlich dem Sicht- und Berührungsschutz des Akkumulators und der dazugehörigen Verdrahtung. An der in Abbildung IV.3 rechts gelegenen Seitenwand wurde das Batteriemanagementsystem befestigt, in der hinten liegenden wurden zwei Steckverbinder eingebaut. Diese dienen dem Laden des Akkus (siehe *Energieversorgung* Abschnitt 2.3.2). Aus optischen Gründen wurden die Außenseiten der Platten sandgestrahlt und mit einem Klarlack-Spray überzogen.

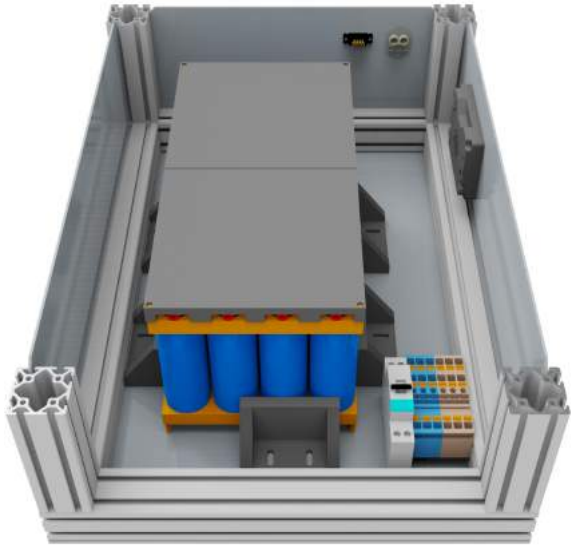


Abbildung IV.3: Seitenplatten mit BMS und Stecker

Da im Fahrbetrieb des Systems, wie bereits erwähnt, Vibrationen entstehen, würden die Seitenplatten, welche in der jeweils äußeren Nut des Gerüsts stehen, eine starke Lärmquelle darstellen. Um diese bestmöglich einzudämmen wurden kleine Winkel entworfen, die die Seitenplatten an den äußeren Rand der Nut pressen. Dadurch sind die Platten nicht mehr beweglich und können somit keine Störgeräusche erzeugen. Zusätzlich kann so kein

Spalt zwischen Aluminium-Profil und Seitenplatte entstehen, was die Optik entscheidend verbessert. Die Winkel wurden mittels 3D-Druck gefertigt und an der jeweils inneren Nut festgeschraubt.

2.3.3 Akkumulatorzugang

Um einen Zugang in den Akkuraum zu schaffen, wurde eine von Magneten in Position gehaltene Abdeckung entwickelt. Sollten Änderungen an dem Batteriemanagementsystem, dem Akku oder der Hutschiene vorgenommen werden müssen, können diese ohne großen Aufwand erfolgen. Die Abdeckung kann mithilfe eines Griffes durch Ziehen entfernt werden.

Für diesen Mechanismus wurden Magnethalter (siehe Abb. IV.4) entworfen und mittels 3D-Druck gefertigt. Der Teil (1) wurde an der inneren Nut des Gerüsts befestigt. Ein Zentrierstift (2) führt die Abdeckung beim Schließen in eine Kerbe (3) und verhindert im geschlossenen Zustand das Verrutschen. Der Teil (4) wurde mit einem Sofortklebstoff für Kunststoffe und Elastomere auf die Rückseite der Abdeckung geklebt. Der selbe Klebstoff hält auch die Magneten in den Halterungen fest. Die Halterungen wurden so entworfen, dass die Magneten sich im geschlossenen Zustand nicht aus den Haltern lösen können, da sie durch die magnetische Anziehungskraft in die für sie vorgesehene Mulde gedrückt werden (5).

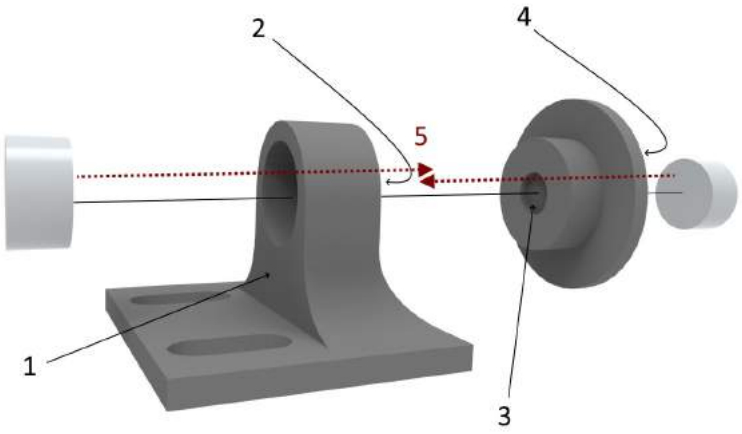


Abbildung IV.4: Explosionsdarstellung des Magnethalters

2.3.4 Grundplatte

Die Grundplatte befindet sich auf der Oberseite des Gerüsts und bedeckt den Akkuraum und das Fahrwerk. Sie ist aus 4 mm starkem Aluminiumblech und trägt den Roboterarm mit seinen Encoder-Adaptern (siehe 5.2.1), Kabelkanäle und das SPS-Gehäuse. Auf ihrer Unterseite, in den Hohlräumen über dem Fahrwerk, wurden die ACOPOSmicro Wechselrichtermodule (siehe 3.3) montiert. Für die Verlegung der Versorgungs- und Datenleitungen in den Akkuraum wurde ein Schlitz vorgesehen. Um die Leitungen vor Schäden an den entstandenen Kanten zu bewahren und die Durchführung möglichst Blickdicht zu halten, wurde ein Kantenschoner entworfen (siehe Abb. IV.5 (orange)) und aus einem flexiblen 3D-Druck Filament (siehe *Stand der Technik* Abschnitt 7) gefertigt.

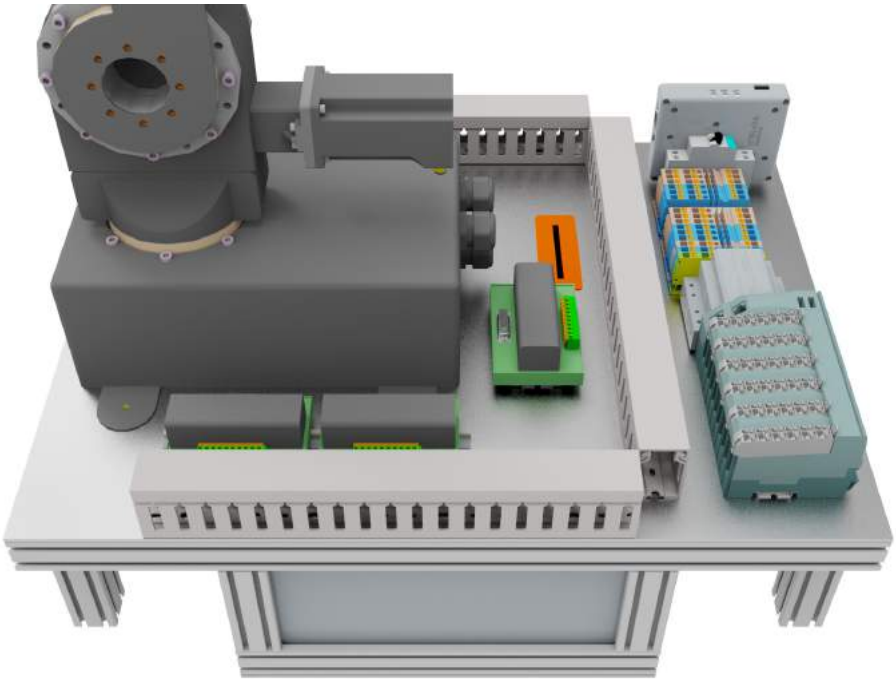


Abbildung IV.5: Grundplatte

2.4 SPS-Gehäuse

Das SPS-Gehäuse befindet sich hinter dem Roboterarm auf der Grundplatte. Es dient als Abdeckung der SPS, der Sicherungen, des Leitungsschutzschalter, der DC-DC-Wandler, einiger Klemmen und der Receiverplatine (siehe 6.2). Vier der fünf Seiten wurden mit durchsichtigen, 4 mm starken Plexiglas-Platten bedeckt, welche, ähnlich der Montage der Seitenplatten, in der äußeren Nut des Gerüsts befestigt wurden. Auf der dem Robolink zugewandten Seite wurde ein Kabelkanal an Stelle der Platte angebracht. Auf der Oberseite befindet sich eine Abdeckung, die sich öffnen lässt, um eine Verbindung zwischen SPS und Laptop herzustellen, die Sicherungen oder Leistungsschutzschalter zu schalten, oder um unkompliziert Änderungen vornehmen zu können. Das Öffnen erfolgt durch denselben Mechanismus wie der des Akkuzugangs. Auf der Rückseite des Gehäuses wurde der Not-Aus-Schalter angebracht.

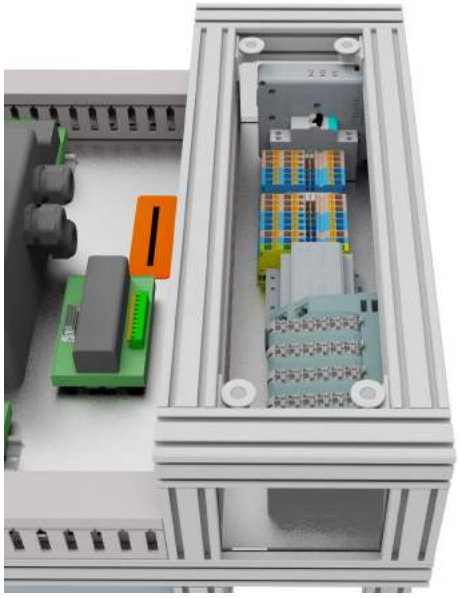


Abbildung IV.6: SPS-Gehäuse

3 Speicherprogrammierbare Steuerung

Die Hardware der SPS[2] wir unterteilt in:

- Compact-S CPU
  - X20 Compact-S CPU (X20CP0484-1)
  - X20 Compact-S Busbasis (X20BB52)
  - X20 Einspeisemodul (X20PS9600)
  - X20 Feldklemme (X20TB12)
- Schrittmotormodule
  - X20 Busmodul (X20BM31)
  - X20 Schrittmotormodul (X20SM1446-1)
  - X20 Feldklemme (X20TB12)
- ACOPOSmicro Wechselrichtermodul
  - Synchron-Motor
  - Resolver- und Motorkabel

[2]kurz für speicherprogrammierbare Steuerung

3.1 Compact-S CPU

Die X20 Compact-S CPU von Bernecker & Rainer wurde hauptsächlich aus Platzsparenden Gründen verwendet. Dennoch besitzt sie alle Funktionen, die für die Umsetzung des Projektes erforderlich sind. Eine RS-232 Schnittstelle, die notwendig für die Kommunikation mit der Receiver Platine 6.2 ist. Einen Anschluss für einen CAN-Bus, der für die Kommunikation mit dem Batteriemanagementsystem (siehe *Energieversorgung* Abschnitt 2) erforderlich ist und einen Powerlink Anschluss, der für die Verbindung zu den Wechselrichtermodulen 3.3 benötigt wird.
Genauere Informationen zu den Verschaltungen sind im Kapitel *Energieversorgung* Abschnitt 3 zu finden.

Abbildung IV.7: X20 Compact-S CPU (X20CP0484-1)

Beginnend mit der Compact-S CPU wird der ganze Steuerungsblock an der Hutschiene angebracht. Im folgenden Bild IV.8 wird der Aufbau der CPU dargestellt. Die Busbasis (1) ist der Unterbau, darauf wird anschließend die Compact-S CPU (3) gesteckt. Danach wird das Einspeisemodul (2) mit der Busbasis (1) verbunden und zum Schluss werden noch die Feldklemmen (4) auf das Einspeisemodul (2) gesteckt.

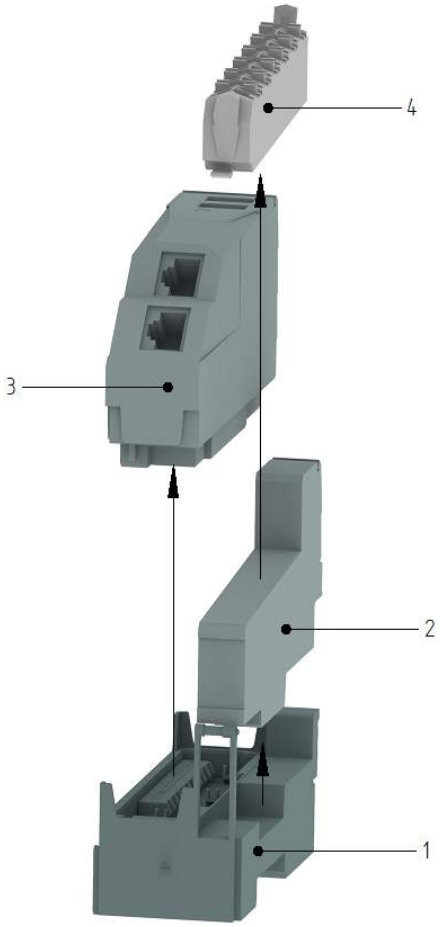


Abbildung IV.8: Aufbau der Compact-S CPU

3.2 Schrittmotormodule

Die Schrittmotormodule werden benötigt, um die Schrittmotoren, die an den verschiedenen Achsen des Roboterarms angebracht sind, anzusteuern. Mit einem Modul kann ein Schrittmotor von 24 V bis 48 V +/− 25% mit einem Dauerstrom von 5 A (10 A Spitze) betrieben werden. Weiters besitzt das Modul vier Eingänge, an denen der Abtriebsencoder (siehe Abschnitt 5.2) angeschlossen werden kann.
Genaue Informationen zur Verschaltung sind im Kapitel *Energieversorgung* Abschnitt 3 zu finden.

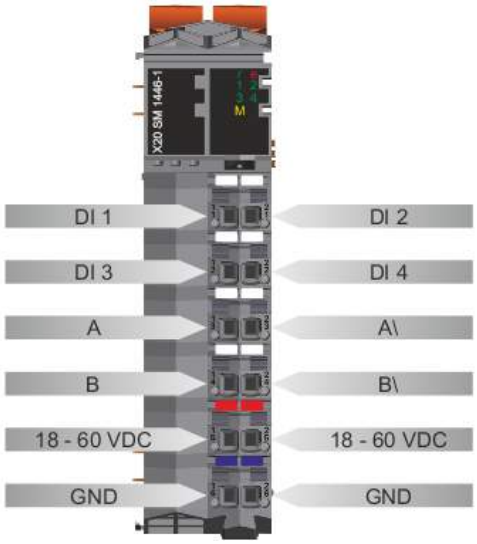


Abbildung IV.9: X20 Schrittmotormodul (X20SM1446-1)

Auf die Hutschiene werden anschließend fünf Schrittmotormodule gesteckt (für jede Achse eines). In Abbildung IV.10 wird der Aufbau eines Schrittmotormoduls dargestellt. Als Basis dient wieder ein Busmodul (1), auf welches ein Schrittmotormodul (2) gesteckt wird. Zum Schluss wird wieder die Feldklemme (3) mit dem Schrittmotormodul (2) verbunden.

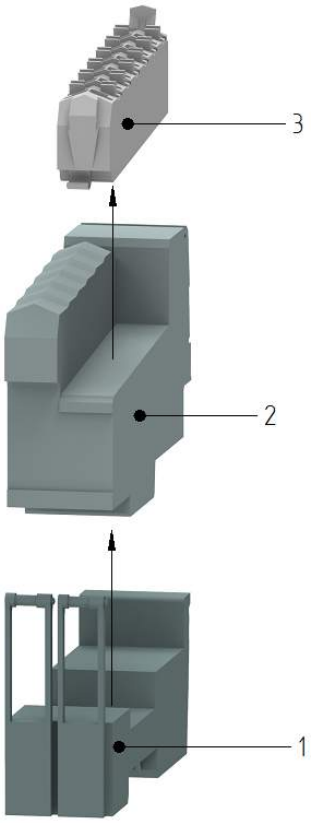


Abbildung IV.10: Aufbau der Schrittmotormodule

Die zusammengestellte Compact-S CPU wird anschließend mit den fünf aufeinanderfolgenden Schrittmotormodulen zusammengesteckt. Die Module werden beim Zusammenstecken mittels X2X-Link verbunden. Der X2X-Link ist zuständig für die Kommunikation und Versorgung zwischen den Modulen. In Abbildung IV.11 wird die fertig zusammengestellte SPS dargestellt.

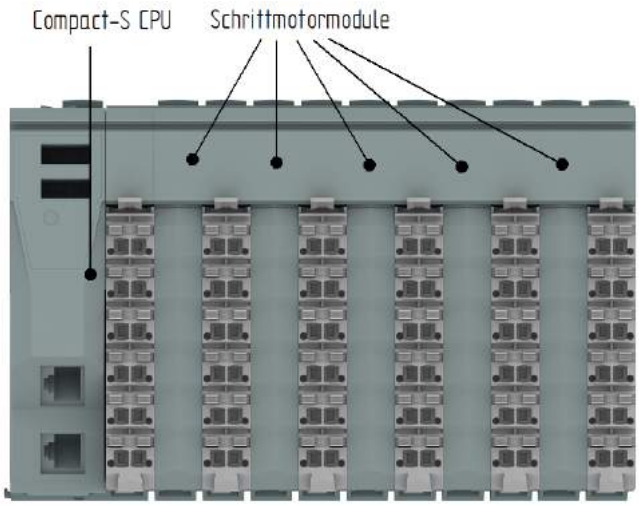


Abbildung IV.11: Zusammengestellte SPS

3.3 ACOPOSmicro Wechselrichtermodule

Für die Ansteuerung der Synchronmotoren des Antriebs (siehe 4.1) wurden die ACOPOSmicro Wechselrichtermodule[3], welche von B&R Industrial Automation zur Verfügung gestellt wurden, verwendet. Entscheidend bei der Auswahl war deren kompaktes Design. Mit jedem ACOPOSmicro lassen sich zwei Motoren ansteuern, weshalb in dem Robotersystem zwei davon verbaut wurden. Sie befinden sich auf der Unterseite der Grundplatte über dem Fahrwerk. Die Verbindung zwischen ACOPOSmicro und SPS/Motoren erfolgt mit den mitgelieferten Kabeln.

[3] vgl. [8].

Abbildung IV.12: ACOPOSmicro Wechselrichtermodul

3.4 Leitungsschutzschalter

Ein Leitungsschutzschalter wird benötigt, um im Fehlerfall (z.B. Kurzschluss), die Steuerungskomponenten zu schützen. Ein Leitungsschutzschalter besitzt einen Überlast- und Kurzschlussschutz. Den Überlastschutz übernimmt ein Bimetallauslöser. Wenn durch einen hohen Strom eine Erwärmung des Bimetalls entsteht, verbiegt sich dieses und löst einen Schalter aus, der den Stromkreis unterbricht. Der Kurzschlussschutz ist durch eine magnetische Auslösung gegeben. Wenn ein Kurzschlussstrom fließt, wird eine Magnetspule angezogen und unterbricht somit den Stromkreis[4].

Für dieses Projekt wurden zwei Leitungsschutzschalter verwendet (C40 und C32), welche für Gleichstromanwendungen ausgelegt sind. Im Kapitel *Energieversorgung* Abschnitt 3, wird die Dimensionierung der Leitungsschutzschalter genauer erläutert.

3.5 DC-DC-Wandler

Ein DC-DC-Wandler ist ein Gleichspannungswandler, der die Eingangspannung in eine andere Spannung wandelt. Der erste Gleichspannungswandler wird verwendet, da der Akku eine Nennspannung von 48 V (51,2 V) besitzt, die SPS jedoch eine Versorgungsspannung von 24 V benötigt. Dieser hat einen Eingangsspannungsbereich von 18-75 V und eine Ausgangsspannung von 24 V mit einer Leistung von 60 W. Da die SPS für die Steuerung keine großen Ströme benötigt, sind 60 W ausreichend. Ein zweiter DC-DC-Wandler wird für die 5 V Versorgung der Receiver Platine (siehe Abschnitt 6.2) benötigt. Dieser hat ebenfalls einen Eingangsspannungsbereich von 18-75 V bei einer Leistung von 60 W, was für die Receiver-PLatine ausreichend ist[5].

[4] vgl. [34], 6.2 Leitungsschutzschalter.
[5] vgl. [42], Specification.

4 Fahrwerk

Das Fahrwerk hat die Aufgabe das Robotersystem zu mobilisieren. Es besteht aus 4 identischen Modulen, die jeweils aus Motor, Motorhalter, Mecanum-Wheel und Flansch zusammengesetzt sind. Diese werden nachfolgend genauer erläutert.

4.1 Motor

Für das Fahrwerk wurde der „8LVA23.R0030D000-0“-Motor[6] von B&R Industrial Automation zur Verfügung gestellt. Dieser wurde gewählt, da dessen Nennmoment (1,3 Nm) und Nenndrehzahl (3000 min^{-1}) die Anforderungen weit übertreffen und dessen maximal zulässige Achslast die benötigten 16,25 kg übersteigt. Es handelt sich bei dem Motor um einen permanentmagneterregten Synchronmotor (siehe *Stand der Technik* Abschnitt 1) mit einer Nennleistung von 408 W. Die Ansteuerung erfolgt, wie bereits erwähnt, mit ACOPOSmicro Wechselrichtermodulen.

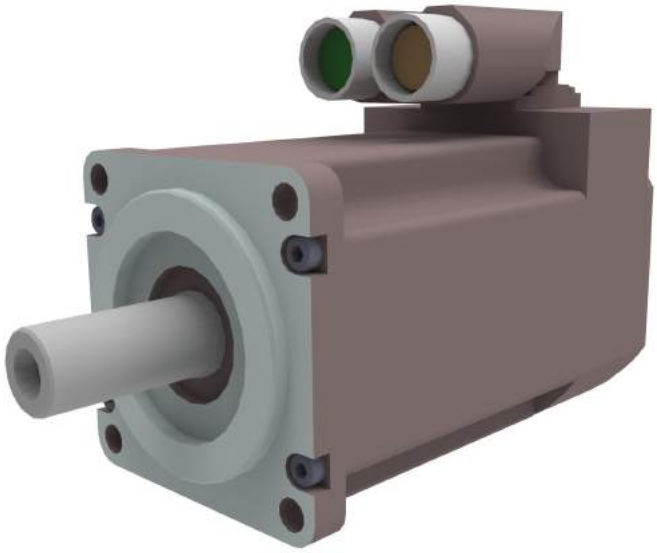


Abbildung IV.13: Motor

4.2 Motorhalter

Um die Motoren am Gerüst befestigen zu können, wurden Motorhalter entworfen und mittels 3D-Druck gefertigt. Da diese jeweils ein Gewicht von ungefähr 15 kg tragen, mussten die Druck-Parameter geändert werden, um diesen Belastungen standzuhalten (siehe *Stand der Technik* Abschnitt 7). Die Halterungen wurden beidseitig am Gerüst befestigt, um einen robusten Einbau zu ermöglichen.

[6] vgl. [6].

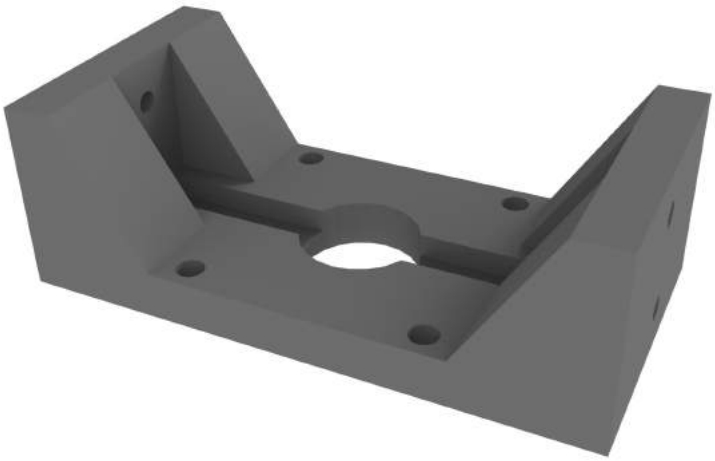


Abbildung IV.14: Motorhalter

4.3 Mecanum-Wheels

Um ein Fahren in alle Richtungen zu ermöglichen, ohne eine Lenkung einbauen zu müssen, wurden Mecanum-Wheels[7] eingesetzt. Diese verfügen über Rollen, die in einem 45° Winkel zur Radachse und auf die Rad-Außenfläche stehen und gleichmäßig über den Umfang verteilt sind. Durch eine spezielle Art der Ansteuerung (siehe *Software* Abschnitt 6.4) kann das Gefährt damit in alle Richtungen fahren, ohne eine Lenkvorrichtung zu benötigen.

Die Räder haben einen Durchmesser von 100 mm, eine Tiefe von 54 mm und verfügen über jeweils 8 Rollen. Sie wiegen 0,75 kg und haben eine Traglast von je 25 kg.

Abbildung IV.15: Mecanum-Wheel

4.3.1 Flansch

Da die Mecanum-Wheels nicht auf die Motorwelle passen, musste ein Flansch gefertigt werden, der mit den Rädern verschraubt werden kann. Die Bohrung, welche mit der Motorwelle verbunden wird, wurde mit der Passung 14H7 gefertigt, die Motorwelle ist 14h6.

[7] vgl. [47].

Um ein Verrutschen des Flansches zu verhindern, wird dieser mit einem Gewindestift auf der Motorwelle fixiert.

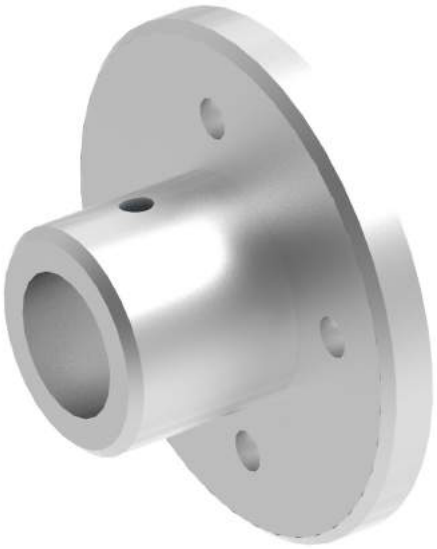


Abbildung IV.16: Flansch

5 Robolink

Für das Projekt wurde das „Robolink RL-DC - Komlettmodul mit fünf Freiheitsgraden[8]“ der Firma igus gewählt. Dabei spielten einige Faktoren eine wichtige Rolle:

- Die Reichweite
- Die maximale Traglast
- Das Eigengewicht
- Der Preis

Das Robolink Komplettsystem wurde speziell für kostengünstige Automatisierung in einem für Menschen gesundheitsschädlichen oder gefährlichen Umfeld entwickelt. Durch die modulare Bauweise konnte der Roboter den Ansprüchen des Roboter-Greifsystems angepasst werden. Die Erweiterung des Roboters mit einem Greifer oder anderen Modulen ist ebenfalls möglich.

5.1 Technische Daten

Der Roboterarm hat im voll durchgestreckten Zustand eine Reichweite von 790 mm[9]. Er kann dabei horizontal von seiner Referenzposition in beide Richtungen um bis zu 140° geschwenkt werden, was eine maximale Drehung von 280° bedeutet (siehe Abb. IV.17 links). Vertikal lässt sich jede der drei Gelenksachsen um 140° bewegen, was den Bereich in Abb. IV.17 links) rechts ergibt. Der aus der Bewegungsfreiheit aller Achsen entstehende Schwenkbereich kommt dem des menschlichen Arms sehr nahe. Da die Ansteuerung des

[8] vgl. [33].

[9] Die Abbildung zeigt die Maße der "kleinen Version"des Robolink, für die "große Version", welche im Projekt verwendet wurde, war leider kein Bild verfügbar.

Robolink durch bewegen des eigenen Arms erfolgt (siehe Kapitel VI), ist er für diesen Zweck gut geeignet. Die maximale Traglast des Robolink beträgt 2500 g.

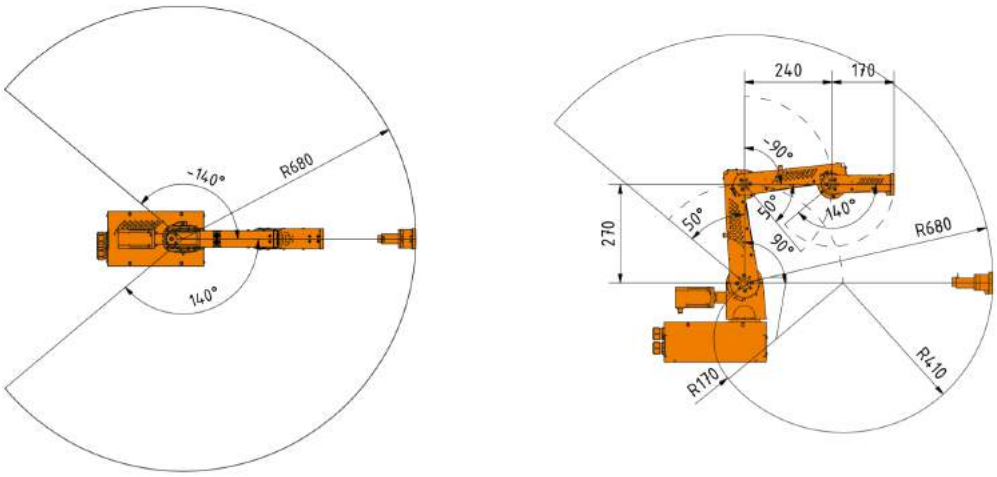


Abbildung IV.17: Schwenkbereich des Robolink[33]

5.2 Abtriebsencoder

Die Gelenke des Robolink verfügen über magnetische Inkrementalencoder an den Achsen, welche für die Positionsbestimmung des Roboterarms verwendet werden. Diese müssen bei jedem Systemneustart neu referenziert werden. Die Signalerzeugung erfolgt durch einen Hall-Sensor, welcher bei einer Drehung der Achse über einen magnetischen Polring fährt und die Änderungen im Magnetfeld nach dem Signalverlauf in Abb. IV.18 ausgibt. Pro Umdrehung geben die Encoder, je nach Baugröße der Achse, zwischen 8.960 und 20.992 Flanken pro Umdrehung aus, welche von der Software erfasst und ausgewertet werden[10].

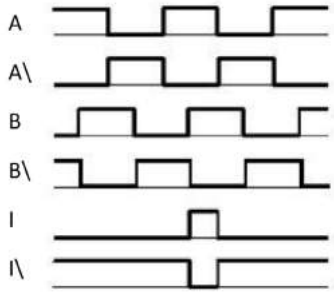


Abbildung IV.18: Signalverlauf der Abtriebsencoder[33]

5.2.1 Encoder-Adapter

Um die Abtriebsencoder mit den Schrittmotormodulen (siehe 3.2) zu verbinden, wird ein Encoder-Adapter[11] benötigt, der die Encoder mit 5 V versorgt und deren Signale von 5 V auf die 24 V wandelt, welche die Schrittmotormodule benötigen. Wie die Encoder-Adapter anzuschließen sind, ist im Kapitel *Energieversorgung* Abschnitt 3.3.2 nachzulesen.

[10] vgl. [32].
[11] vgl. [7].

Die insgesamt fünf Encoder-Adapter wurden rund um den Robolink auf der Grundplatte platziert und auf Hutschienen befestigt. Ihre Kabel wurden in die Kabelkanäle geführt und von dort zu den Schrittmotormodulen, beziehungsweise dem Robolink, verbunden.

5.3 Greifer

Um die Bewegungen der Hand vom Fernsteuergerät bestmöglich auf den Greifer übertragen zu können, musste ein Greifsystem gewählt werden, das einer menschlichen Hand nachempfunden wurde. Die Entscheidung fiel auf die „bionische Roboterhand (rechts)“ von DFRobot[12]. Sie hat fünf einzeln ansteuerbare Finger, ein Eigengewicht von ca. 900 g und eine Nutzlast von 500 g. Das Gewicht konnte noch auf ca. 650 g reduziert werden, da für die Montage die Handbasis entfernt wurde.

Die fünf Servomotoren der Finger benötigen eine Betriebsspannung von 4,8 - 6 V bei einem Betriebsstrom von 2 A. Die Versorgung und Ansteuerung erfolgt über drei Drähte (Rot: VCC, Braun: GND, Orange: Positioniersignal).

Abbildung IV.19: bionischer Robotergreifer[20]

5.3.1 Montage am Robolink

Zur Befestigung des Greifers am Robolink wurde ein Aufsatz für den Drehteller des robolinks entworfen und mittels 3D-Durck gefertigt. Der Aufsatz kann am Drehteller verschraubt werden. Die Handbasis des Greifers wurde entfernt und die bestehenden Bohrlöcher verwendet, um die Hand an den Drehtelleraufsatz zu montieren. Für die Stromversorgung und Ansteuerung wurde ein 20-Poliges Bandkabel verwendet (fünf Adern bleiben für etwaige Erweiterungen frei), welches in dem Kabelkanal des Robolink bis zum Drehteller verlegt wurde.

[12] vgl. [19].

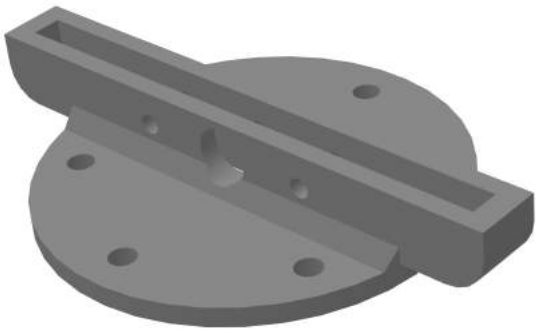


Abbildung IV.20: Drehtelleraufsatz zur Befestigung des Greifers

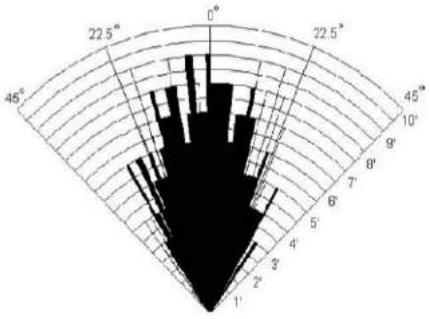
6 Abstandssensorik

6.1 Ultraschall-Sensormodul HC-SR04

Gewählt wurde für die Abstandssensorik das Ultraschall-Modul HC-SR04[13], welches sehr günstig und leicht erhältlich ist. Es ist für einen Messbereich von ungefähr 2 cm bis 3 m geeignet und misst dabei die Entfernung in Inkrementen von etwa 3 mm. Abgedeckt wird dabei ein Bereich von etwa 30° (siehe IV.21b). Die Versorgung erfolgt mit 5 V DC und einer Stromstärke von 2 mA (PIN1: VCC, PIN4: GND).

Angesteuert wird das Sensormodul über zwei Pins (PIN2: Trigger, PIN3: Echo). Das genaue Funktionsprinzip der Auswertung ist im Kapitel *Software* Abschnitt 5.6 beschrieben.

(a) Ultraschall-Modul HC-SR04



(b) Messwinkel ca. 30°

Abbildung IV.21: Ultraschall-Sensormodul HCSR04[15]

[13] vgl. [15].

6.1.1 Sensorhalter

Um die Sensoren am Gehäuse zu befestigen, wurden zwei verschiedene Arten von Sensorhaltern designed:

- Sensorhalter zur Montage an den Ecken
- Sensorhalter zur Montage entlang der Kanten

Diese Kombination erlaubt eine möglichst großflächige und lückenlose Abtastung von Hindernissen in der Nähe des Robotergreifsystems. Zusätzlich bedecken die Sensorhalter die für das Verlegen der Sensorkabel notwendigen Löcher im Alu-Profil (siehe 6.4).

Zunächst wurden die Sensoren vermessen, um passgenaue Gehäuse zu schaffen, welche per 3D-Druck-Verfahren gefertigt wurden. Um sicherzustellen, dass die beiden Ultraschallkapseln stets bündig mit dem Sensorhalter abschließen, wurde eine Klebelasche (1) vorgesehen. Die Montage am Gerüst erfolgt über Verschraubung in den Nuten (2).

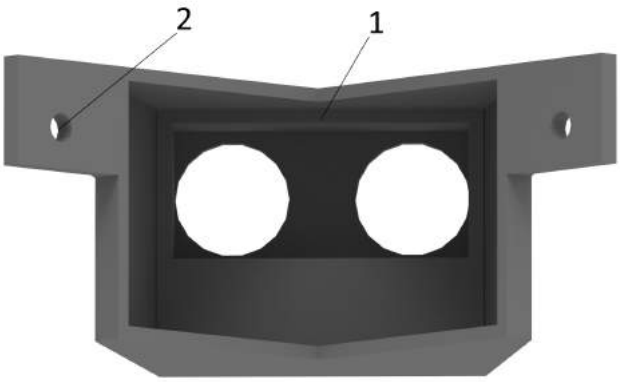


Abbildung IV.22: Sensorhalter

6.1.2 Ausrichtung und Platzierung

Ursprünglich wurden Überlegungen dahingehend angestellt, ein Abstandssensor-System zu entwickeln, das mit möglichst wenigen Sensoren den gesamten Bereich um das Roboter-Greifsystem abdeckt. Darauf basierend entstand ein Konzept mit nur acht Sensoren, wovon jeweils zwei teilweise zueinander gerichtete eine Seite abdecken sollten (siehe Abb. IV.23). Aus dem „Sichtbereich“ (rot) sollte der Bereich, in dem der Gegenüberliegende Sensor ein Echo verursacht, herausgefiltert werden, indem Werte aus genau dessen Entfernung ignoriert würden. Der dabei entstehende „blinde Bereich“ (violett) müsste in Kauf genommen werden. Hinter den Sensoren und etwas von der Kante des Roboters entfernt entstünden ebenfalls unüberwachte Bereiche (hellblau).

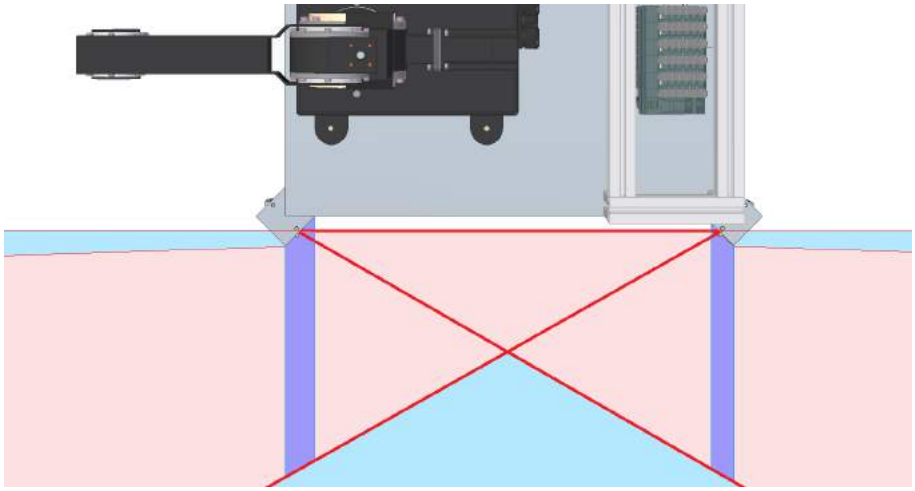


Abbildung IV.23: Erster Entwurf der Sensorausrichtung

Dieses Konzept musste allerdings verworfen werden, da das angedachte „Ausblenden“ des gegenüberliegenden Sensors nicht umsetzbar ist. Bekommt nämlich der Echo Pin ein Signal (wird ein Hindernis erkannt), kann der Sensor nicht mehr auf ein weiteres Signal warten, weil er seinen Tastzyklus sofort beendet. Weil der gegenüberliegende Sensor aber selbst ein Hindernis darstellt, können hinter ihm folglich keine Werte mehr erkannt werden, da der Messvorgang beendet wird. Weiters würde mit dieser Ausrichtung der Sensoren (beinahe parallel zur Kante) keine Aussage über den Abstand eines Hindernisses getroffen werden können, nur, ob sich ein Hindernis zu nahe befindet.

Aus diesen Überlegungen wurde eine effektivere Lösung entwickelt. Dabei wurden 14 Sensoren gleichmäßig am Rahmen verteilt, mit den Messbereichen normal zum Rahmen. Damit können die Abstände in alle Richtungen genau genug bestimmt werden.

Abbildung IV.24: Finales Sensorlayout

6.2 Receiver-Platine

Die Receiver-Platine erfüllt mehrere Aufgaben. Sie ist für die Ansteuerung und Auswertung der Ultraschallsensor-Module und der Servomotoren des Greifers zuständig. Außerdem stellt sie eine Verbindung zwischen dem Fernsteuergerät und der SPS her. Nachfolgend wird das elektronische Funktionsprinzip der Platine erläutert.

6.2.1 Funktionsprinzip

Die Spannungsversorgung der Platine erfolgt über die Anschlussklemmen X1-1 und X1-2. Eine Sicherung F1 und ein Stützkondensator zur Stabilisierung wurden eingebaut. Die Sicherung ist auf 5 A ausgelegt, da die Motoren des Greifers insgesamt 2 A benötigen und Leiterbahnen der Platine auf über 5 A ausgelegt sind. Der Stützkondensator dient dem Ausgleichen von Stromspitzen bei Einschaltvorgängen der Motoren.

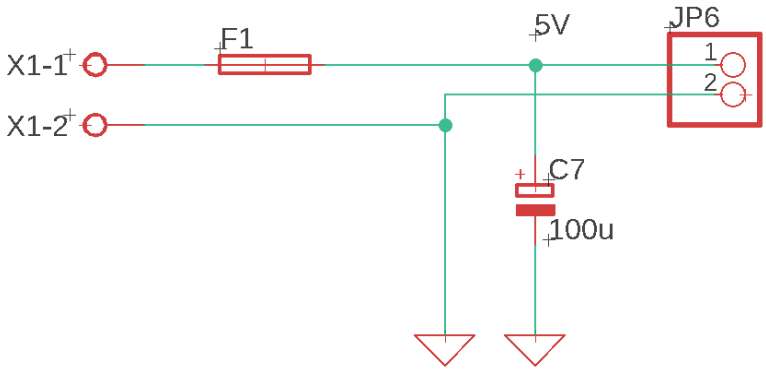


Abbildung IV.25: Spannungsversorgung der Receiverplatine

Zur Überwachung der Platine wurden drei Status-LEDs vorgesehen. LED 1 leuchtet, sobald die Platine mit Spannung versorgt wird. LED 2 zeigt an, ob eine BLE Verbindung besteht und LED 3, ob eine Verbindung mit dem I^2C-Bus besteht.

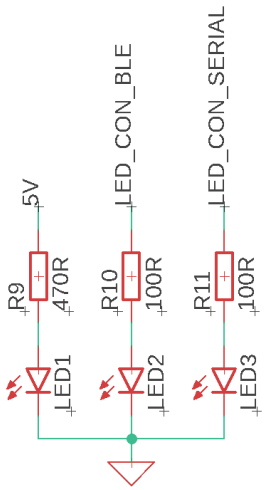


Abbildung IV.26: Status-LEDs der Receiverplatine

Die 4-Poligen Kabel der Ultraschallsensoren (siehe 6.4) wurden über Pin header (Abb. IV.27) mit der Platine verbunden.

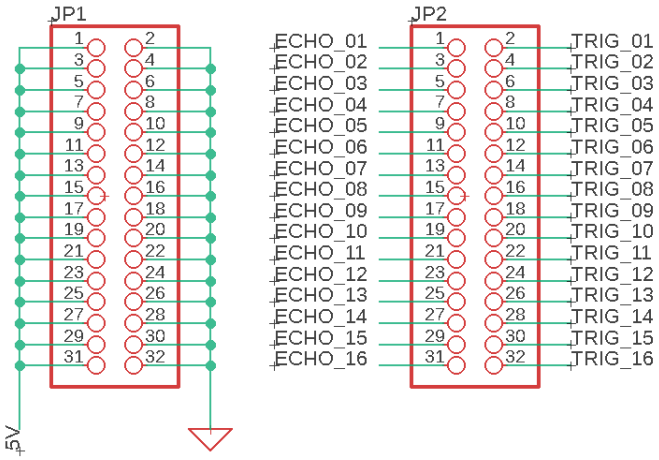


Abbildung IV.27: Anschluss der Ultraschall-Sensormodule

Der ATmega328P ist für die Auswertung der Ultraschall-Sensormodule verantwortlich. Versorgt wird der Mikrocontroller über 5 V und GND. Reset, RXD und TXD werden zum Programmieren verwendet. An den Pins 9 und 10 ist ein Quarzkristall, welcher eine Taktfrequenz von 16 MHz generiert, angeschlossen. Die Kondensatoren C1 (22 pF) und C3 (22 pF) sorgen für die Oszillation des Kristalls. SDA und SCL sind die Datenleitungen des I^2C-Busses. Die übrigen Pins wurden für die Trigger und Echo Signale (siehe 6.1) der Ultraschall-Sensormodule verwendet und führen zu den Pin-Header in Abb. IV.27, an denen die Sensorkabel angeschlossen werden. Da ein einzelner ATmega328P nicht über ausreichend freie Pins verfügt, um alle Sensoren anzuschließen, mussten auf der Platine zwei verbaut werden.

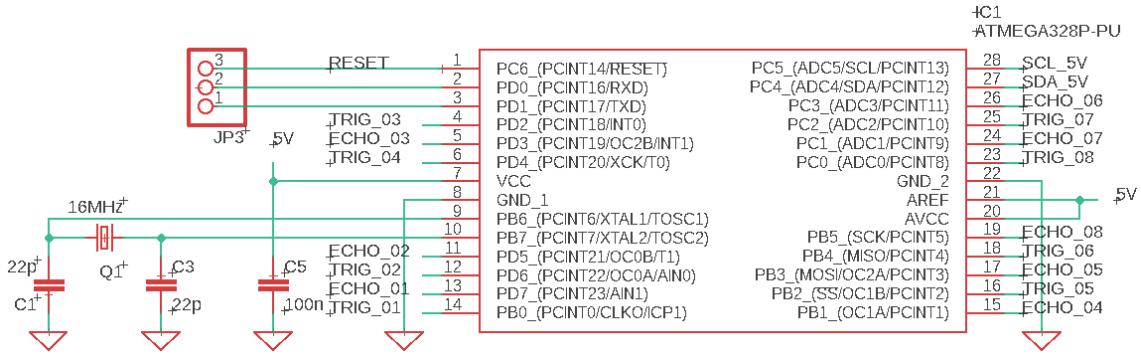


Abbildung IV.28: Schaltplan ATmega328P

Für den Fall, dass im Programm des ATmega328P Fehler auftreten, wurde ein Reset-Taster vorgesehen.

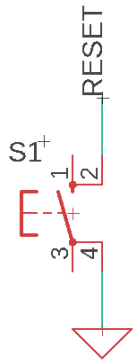


Abbildung IV.29: Reset-Taster des ATmega328P

Der ESP32 stellt eine Verbindung zwischen dem Fernsteuergerät (über BLE) und der SPS (über eine serielle Schnittstelle) her. Die Funktionsbeschreibung ist im Kapitel *Software* Abschnitt 5 nachzulesen. Auf dem Modul ist ein Linearregler verbaut, der die 5 V Versorgungsspannung auf 3,3 V herunterregelt. SCL und SDA sind die I^2C-Bus Anschlüsse. Über den Pin D15 lässt sich der Reset des ATmega328P auch softwaretechnisch auslösen. Die Pins RX und TX werden über einen UART-to-RS232 Konverter mit der SPS verbunden (siehe *Stand der Technik* Abschnitt 5.1). Die Status-LEDs werden von den Pins D18 und D5 gesteuert.

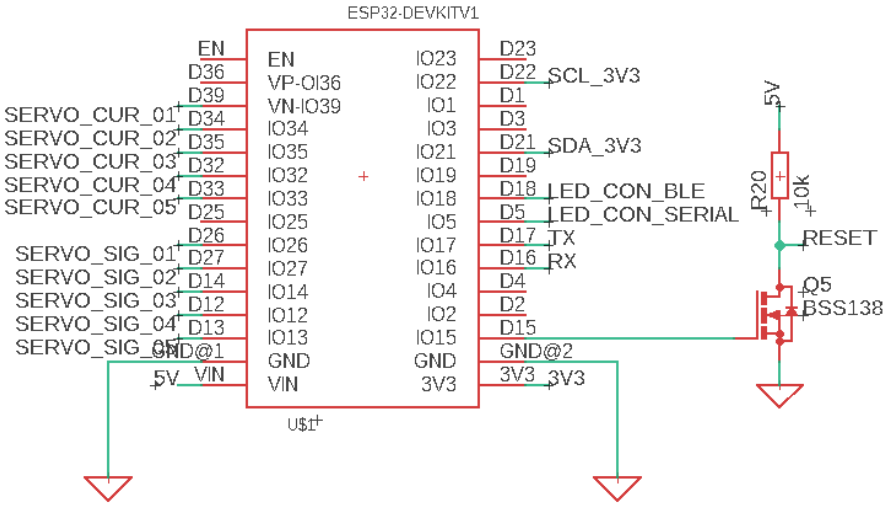


Abbildung IV.30: Schaltplan ESP32

Da der ESP32 über eine Betriebsspannung von 3,3 V und der ATmega328P über eine Betriebsspannung von 5 V verfügt, muss in den I^2C-Bus, der die beiden verbindet, ein Level-Shifter eingebaut werden (siehe *Stand der Technik* Abschnitt 5.3). Eine sehr simple, aber effektive Ausführung eines bidirektionalen Level-Shifters, stellt ein diskreter MOSFET dar[14].

[14] vgl. [50].

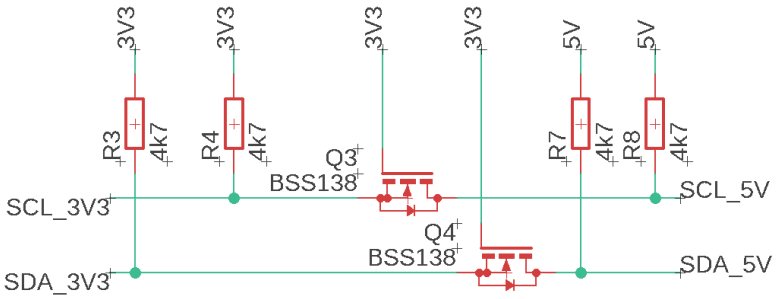


Abbildung IV.31: I^2C Level-Shifter

Am Stecker JP4 werden die Servomotoren des Greifers mit 5 V versorgt und bekommen ein PWM-Signal zur Ansteuerung. Die Masseanschlüsse der Motoren wurden über Messwiderstände auf GND verbunden, um über deren Spannungsabfälle die Stromaufnahme berechnen zu können, welche für die Aktivierung des taktilen Feedbacks verwendet wird (siehe Kapitel *Fernsteuergerät* Abschnitt 3.1). Dieser wird über einen RC-Filter gefiltert, da der Strom vom internen Positionsregler der Motoren gepulst wird. Der geglättete Spannungswert wird mit den analogen Eingängen des ESP32 verbunden.

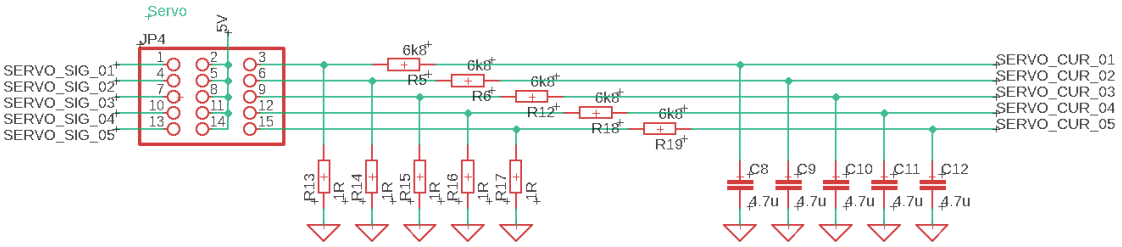


Abbildung IV.32: Anschluss der Servomotoren des Greifers

6.2.2 Layout

Nachfolgend ist das Layout der Receiver-Platine einzusehen. Die bereits erwähnten Bauteile befinden sich auf dem Bottom-Layer der Platine und sind in Bild IV.34 zu erkennen.

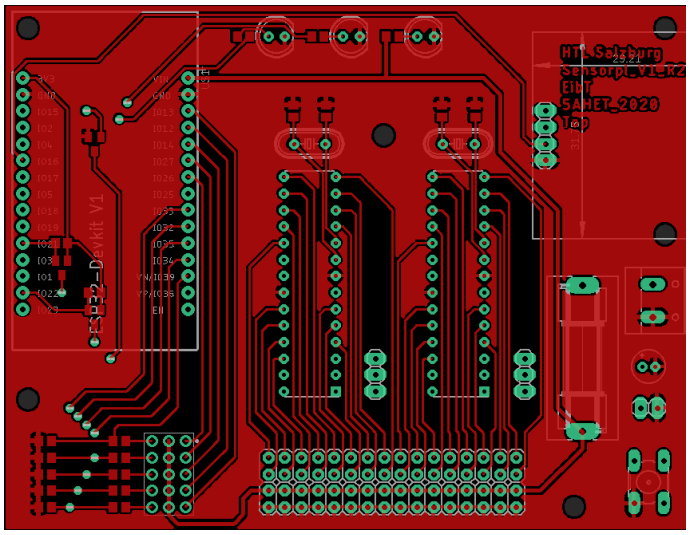
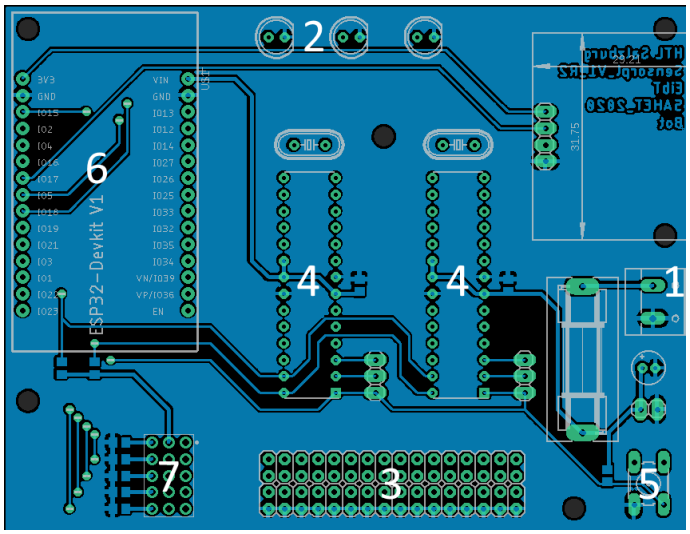


Abbildung IV.33: Receiver-Platine Top



1. Spannungsversorgung
2. Status-LEDs
3. Pin-header für Sensorkabel
4. ATmega328P
5. Reset-Taster
6. ESP32
7. Servomotor-Anschlüsse

Abbildung IV.34: Receiver-Platine Bottom

6.3 Platinengehäuse

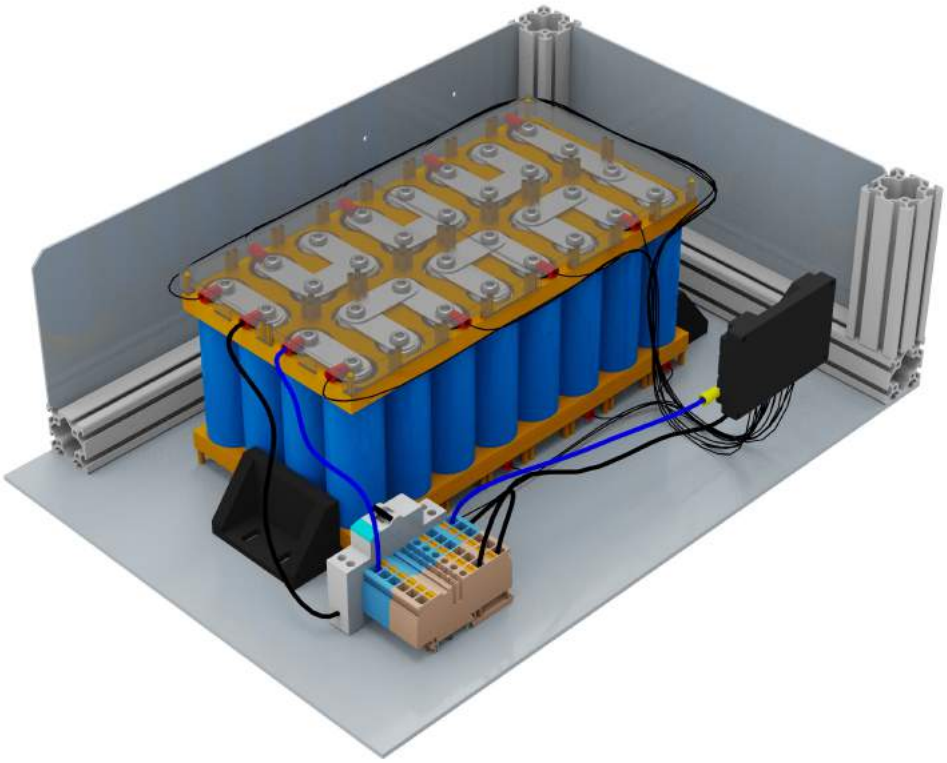
Für die Befestigung der Platine auf der Hutschiene im SPS-Gehäuse wurde ein Platinengehäuse entworfen. Es besitzt oben drei Aussparungen für die Status-LEDs und vorne ein Fenster, um die Kabel der Ultraschallsensoren sowie die der Servomotoren des Greifers anstecken zu können. Auf der Unterseite ist die Befestigungsklammer für die Hutschiene zu sehen. Die genauen Dimensionen sind den Zeichnungen im Anhang zu entnehmen.

Abbildung IV.35: Gehäuse der Receiver-Platine

6.4 Verdrahtung

Für die Versorgung und Ansteuerung der Sensormodule musste eine intuitive Lösung gefunden werden, da bei 14 Sensoren mit je vier Drähten insgesamt 56 Adern verlegt werden mussten. Dafür wurden 4-Polige Kabel gefertigt, die von der Sensorplatine im SPS-Gehäuse bis zu den Sensoren verlegt wurden. Um Übersichtlichkeit zu bewahren und das Anbringen mehrerer Kabelkanäle zu vermeiden, wurden die Kabel in dem Hohlraum der Alu-Konstruktionsprofile des Gehäuserahmens verlegt. Dafür mussten diese an der Innenseite an zwei Stellen und an der Außenseite unter jedem Sensorhalter angebohrt und die Kabel anschließend eingezogen werden.

V Energieversorgung



1 Übersicht

1.1 Aufgaben der Energieversorgung als Gesamtprojekt

Eine stabile akkubasierte Energieversorgung, die mit den Verbraucher-Komponenten verbunden ist, wird benötigt, um dem Robotersystem Mobilität zu verleihen. Daher besteht die Aufgabe der Energieversorgung, alle Komponenten mit ausreichend Spannung zu versorgen und diese auch gegen jegliches elektrisches Versagen zu schützen (z.B. Kurzschluss). In dem folgenden Kapitel wird erläutert, wie die Energieversorgung geplant und anschließend auch umgesetzt wurde.

1.2 Aufgaben des Batteriemanagement

Ein Batteriemanagement ist notwendig, um Messwerte – Spannung und Strom – vom Akku aufzunehmen, diese zu verarbeiten und anschließend an die SPS weiterzugeben. Weiters ist es noch für den Schutz des Akkus zuständig und für eine geregelte Ladung mithilfe eines externen Ladegeräts.

Das Einlesen der Daten und überwachen der einzelnen Akkuzellen übernimmt das Batteriemanagementsystem. Diese Daten werden dann anschließend im BMS verarbeitet und über einen CAN-Bus weiter an die SPS gegeben.

Ein externes Ladegerät wird benötigt, das bei Bedarf mit dem Batteriemanagement und dem Akku verbunden wird, um eine schnellstmögliche geregelte Aufladung zu ermöglichen.

1.3 Aufgaben der Elektrischen Installation

Um den Akku und die verschiedenen Komponenten miteinander zu verbinden wird eine elektrische Installation benötigt. Diese muss im Vorhinein sorgfältig geplant werden, um spätere Fehler zu vermeiden. Hierbei ist es besonders wichtig auf die Dimensionierung der Kabel und Leitungsschutzschalter zu achten. Ebenfalls ist es wichtig bei der Realisierung der elektrischen Installation sorgfältig zu arbeiten um Fehler – wie zum Beispiel Kurzschlüsse durch falsche Verdrahtung – zu vermeiden.

2 Batteriemanagement

2.1 Akkumulatoren

2.1.1 Dimensionierung

Um die Energie effizient zu speichern wurde ein Lithium-Eisen-Phosphat Akkusystem verwendet. Da der Roboterarm und der Antrieb mit 48 V gespeist werden, fiel die Entscheidung auf ein 48 V Versorgungssystem. Ebenfalls ist es wichtig, dass man bei der Dimensionierung auf die Lade- und Entladeströme achtet. Beim Ladestrom musste nur der Strom des Ladegeräts berücksichtigt werden (mehr dazu im Abschnitt 2.3). Um einen Richtwert für den maximalen Entladestrom zu erhalten, wurde der Nennstrom von sämtlichen Verbrauchern addiert. Wie bereits im Kapitel *Einführung* Abschnitt 4.5 erklärt wurde, gibt es zwei verschiedene Bedienmodi. Es kann entweder der Roboterarm oder der Antrieb bedient werden. Deswegen muss für beide Bedienmodi der maximale Strom berechnet werden. In der Tabelle V.1 werden die Ströme der einzelnen Achsen, die der Receiver Platine und die der Steuerungselemente summiert.

Die Nennströme der Achsen wurden aus dem Datenblatt des Roboterarms entnommen. Der Stromverbrauch der Encoder-Adapter wurde ebenfalls aus deren Datenblätter entnommen. (siehe Tabelle V.1)

Im Kapitel *Robotersystem* Abschnitt 6.2.1 wurde bereits begründet, warum bei der Receiver Platine eine 5 A Sicherung verwendet wird. Diese 5 A wurden anschließend auch als Stromverbrauch der Platine angenommen.

Der Verbrauch der Steuerungselemente setzt sich aus der SPS und den ACOPOSmicro zusammen. Die SPS benötigt 2,5 W und die beiden ACOPOSmicro benötigen gemeinsam eine Leistung von 16 W. Diese Komponenten werden mit einer Spannung von 24 V gespeist.

$$I_{SPS} = \frac{P_{SPS}}{U} = \frac{2,5W}{24V} = 100mA \tag{V.1}$$

$$I_{ACOPOS} = \frac{P_{ACOPOS}}{U} = \frac{8W}{24V} = 334mA \tag{V.2}$$

Der Stromverbrauch des Batteriemanagementsystems wurde für diese Berechnung nicht berücksichtigt, da dieser bei wenigen mA liegt.

	U in V	I in A	P in W
Roboterarm			
Achse 1	48	4,2	202
Achse 2	48	4,2	202
Achse 3	48	4,2	202
Achse 4	48	1,8	86
Achse 5	48	1	48
Encoder-Adapter	24	1,25	30
Receiver Platine	5	5	25
Steuerungselemente	24	0,83	20
Summe	-	20,5	814

Tabelle V.1: Stromberechnung Bedienmodus „Greifen“

Die Receiver Platine und die Steuerungselemente werden dauerhaft mit Strom versorgt, daher wird beim Bedienmodus „Fahren“ der Strom, der einzelnen Achsen des Roboterarms, durch den Strom den die Antriebsmotoren benötigen, ersetzt.

Die Nennströme der Motoren wurden aus deren Datenblättern entnommen. (siehe Tabelle V.2)

Die Ströme der Receiver Platine und die der Steuerungselemente verändern sich bei diesem Bedienmodus nicht. Tabelle V.2 zeigt den Stromverbrauch des zweiten Bedienmodus.

	U in V	I in A	P in W
Antrieb			
Motor 1	48	5,8	278,4
Motor 2	48	5,8	278,4
Motor 3	48	5,8	278,4
Motor 4	48	5,8	278,4
Receiver Platine	5	5	25
Steuerungselemente	24	0,83	20
Summe	-	27,0	1159

Tabelle V.2: Stromberechnung Bedienmodus „Fahren“

Nachfolgendes Bild V.1 enthält die Daten einer einzelnen Akkuzelle. Aus diesem kann der maximale Entladestrom entnommen werden, dieser beträgt 100 A, welcher ausreichend für beide Bedienmodi ist.

Zellchemie:	LiFePO4
Nennspannung:	3,2V (3,3V)
Nenn-Kapazität:	10Ah
Entladestrom :	100A
Entladestrom (Puls ≤10sec.):	150A
Ladestrom max.:	30A, empfohlen ≤0,5C
Arbeitsbereich:	2,5 bis 3,6V (nicht unter 2,0V)
Innenwiderstand (mΩ):	≤4
Temperatur (Entladen):	-20°C bis +60°C
Temperatur (Laden):	0°C bis +45°C
Lagertemperatur:	-40°C bis +60°C
Zyklenfestigkeit:	>1000 (100%DOD), >2000 (80%DOD)
Eigenentladung (monatlich):	<5%
Anschlüsse:	M6
Gehäuse:	Metall
Gewicht:	330g ± 10g
Abmessung (lxbxh):	147x38mm (inkl. Schrauben)

Abbildung V.1: Eigenschaften Lithium-Eisen-Phosphat Akkuzelle[39]

Die Nennspannung einer Akkuzelle liegt bei 3,2 V. Um ein System mit 48 V zu erreichen wurden 16 Zellen in Serie geschlossen.

$$U_{Gesamt} = 3,2V * 16 = 51,2V \tag{V.3}$$

Die Gesamte Nennspannung beträgt nun 51,2 V.

Die Nennkapazität einer Akkuzelle liegt bei 10 Ah. Um anschließend die Kapazität zu erhöhen wurden jeweils zwei Akkuzellen Parallel geschaltet.

$$Q_{Gesamt} = 10Ah * 2 = 20Ah \tag{V.4}$$

2.1.2 Zusammenstellung

Für die Zusammenstellung des Akkus wurde ein Bausatz benutzt, der die Halterungen, Verbindungsstücke und Schrauben enthielt.

Abbildung V.2: Explosionsdarstellung einer Doppelzelle

Die Akkuzellen wurden anschließend in einem 4x8 Muster angeordnet.

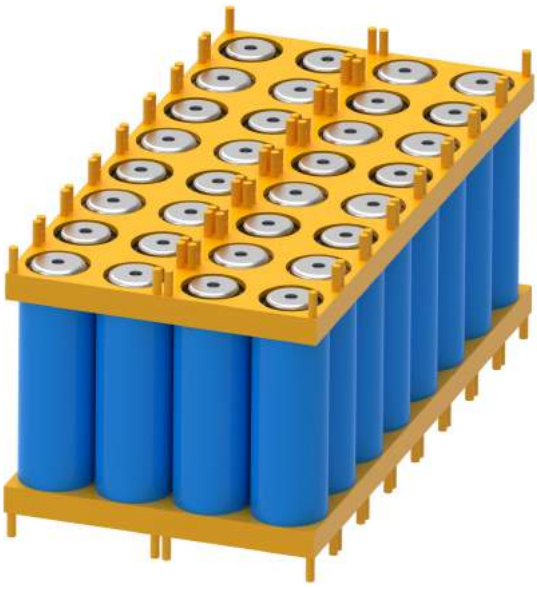


Abbildung V.3: Zusammengestellter Akku

2.1.3 Verschaltung

Wie schon im Kapitel Dimensionierung angesprochen werden 16 Zellen in Serie geschlossen und jeweils zwei parallel.

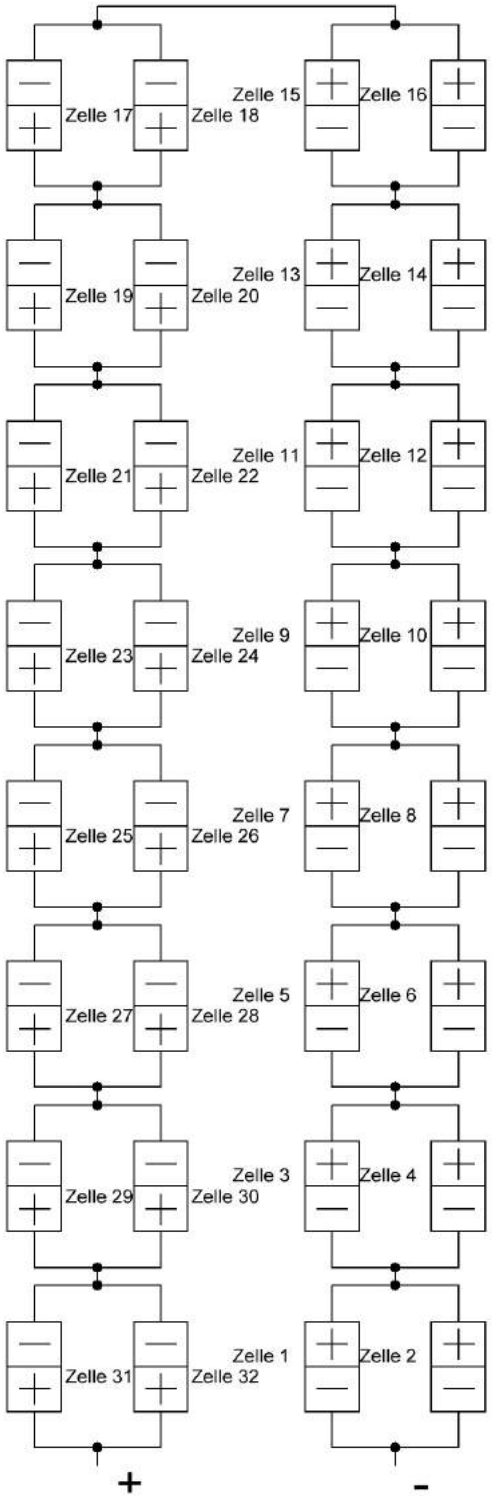
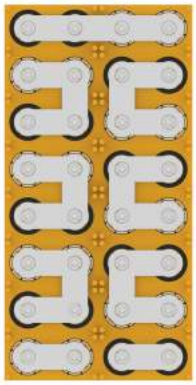


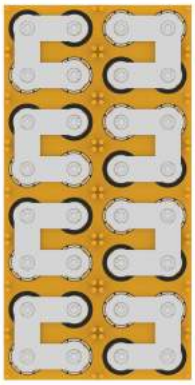
Abbildung V.4: Verschaltung des Akkus

Um die Verschaltung so einfach wie möglich zu gestalten, wurde jede zweite Doppelzelle umgedreht. In den folgenden Bildern V.5a und V.5b ist die Verschaltung der Zellen abgebildet. Die schwarze Umrandung stellt den negativen Pol der Akkuzelle dar.

Das Bild V.5a zeigt die Draufsicht des Akkus. Im rechten unteren Eck befindet sich das erste Zellenpaar und im linken unteren Eck das letzte. Die Nummerierung der Zellenpaare ist später, für die Anschlüsse der Zell-Spannungsmessung, notwendig. Weiteres dazu im Abschnitt Batteriemanagement Verschaltung 2.2.2.



(a) Ansicht oben



(b) Ansicht unten

Abbildung V.5: Akku verschiedene Ansichten

2.1.4 Akkuhalter

Im Kapitel *Robotersystem* Abschnitt 2.3.1 wurde bereits erwähnt, dass der Akku auf eine Aluminiumplatte gestellt wird. Um diesen gegen Verrutschen zu sichern, wurden Akkuhalter konstruiert. Diese wurden anschließend mittels 3D-Druck gefertigt. Weiters wurde noch eine Unterlage an der Aluminiumplatte befestigt. Diese dient zum einen als Antirutsch-Unterlage und zum anderen als Isolierung zwischen Aluminium und Akku.

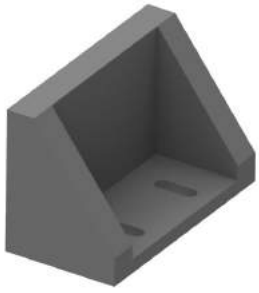


Abbildung V.6: Akkuhalter

2.1.5 Abdeckung

Um einen Schutz gegen direktes Berühren an der Oberseite zu erstellen, wurde eine Abdeckung konstruiert und mittels 3D Druck gefertigt. Auf den Seiten wurden Ausschnitte hinzugefügt. Diese werden für die Verkabelung des Batteriemanagementsystems benötigt.

Abbildung V.7: Abdeckung des Akkus

2.1.6 Geschätzte Laufzeit

In diesem Abschnitt wird eine ungefähre Berechnung der Laufzeit durchgeführt. Im Abschnitt Batteriemanagement Dimensionierung 2.1.1 wurde bereits der Stromverbrauch der verschiedenen Bedienmodi berechnet. Um eine Näherung für die Laufzeit zu erstellen, wurden jeweils die Leistungen addiert.

Zuerst wird die gesamte Energie berechnet, die der Akku speichern kann. (siehe Berechnung V.5)

$$W_{akku} = Q_{Gesamt} * U_{Gesamt} = 20Ah * 51,2V = 1024Wh \tag{V.5}$$

Beim ersten Bedienmodus wird die meiste Leistung vom Roboterarm verbraucht, da sich bei diesem aber nicht alle Achsen gleichzeitig und nicht immer mit voller Geschwindigkeit bewegen, wird eine Auslastung von 50% angenommen. Die Encoder-Adapter müssen ebenso in die Berechnung einbezogen werden. (siehe Berechnung V.6)

$$\begin{aligned} P_{RGesamt} &= P_{Achse1} + P_{Achse2} + P_{Achse3} + P_{Achse4} + P_{Achse5} + P_{Encoder\,Adapter} \\ P_{RGesamt} &= 202W + 202W + 202W + 86W + 48W + 30W = 770W \end{aligned} \tag{V.6}$$

Von dieser gesamten Leistung wird dann, wie bereits erwähnt, nur 50% für die Näherungsrechnung mit einbezogen. (siehe Berechnung V.7)

$$P_{RoboterarmNäherung} = P_{RGesamt} * 50\% = 385W \tag{V.7}$$

Beim Antrieb wurden ebenfalls Überschlagswerte verwendet. Zuerst werden alle Nennleistungen der Motoren zusammenaddiert (siehe Berechnung V.8). Anschließend wird von der gesamten Leistung nur 50% in die Näherungsrechnung mit einbezogen (siehe Berechnung V.9).

$$\begin{aligned} P_{AGesamt} &= P_{Motor1} + P_{Motor2} + P_{Motor3} + P_{Motor4} \\ P_{AGesamt} &= 278,4W + 278,4W + 278,4W + 278,4W = 1113,6W \end{aligned} \tag{V.8}$$

$$P_{AntriebNäherung} = P_{AGesamt} * 50\% = 556,8W \tag{V.9}$$

Danach müssen die Ergebnisse noch addiert werden. Dabei muss beachtet werden, dass es zwei verschiedene Bedienmodi gibt, welche nie zur selbe Zeit aktiv sind. Daher wird angenommen, dass der Roboterarm nur 1/3 der gesamten Zeit aktiviert ist, der Antrieb hingegen 2/3 der gesamten Zeit. Ebenfalls wird noch die Leistung der Receiver Platine und die der Steuerungselemente dazu addiert. (siehe Berechnung V.6)

$$\begin{aligned} P_{Gesamt} &= P_{RoboterarmNäherung} * \frac{1}{3} + P_{AntriebNäherung} * \frac{2}{3} \\ &\quad + P_{Steuerungselemente} + P_{ReceiverPlatine} \\ P_{Gesamt} &= 385W * \frac{1}{3} + 556,8W * \frac{2}{3} + 20W + 25W = 545W \end{aligned} \tag{V.10}$$

Zum Schluss wird die Laufzeit berechnet.

$$t_{Näherung} = \frac{W_{Akku}}{P_{Gesamt}} = \frac{1024Wh}{545W} = 1,87h = 113min \tag{V.11}$$

Die geschätzte Laufzeit beträgt in etwa 113 Minuten.

2.2 Batteriemanagementsystem

Der zweite Teil, der für ein funktionierendes Akkusystem benötigt wird, ist ein Batteriemanagementsystem. Dessen Aufgabe ist es, die Akkuzellen zu überwachen und die Daten an die SPS weiterzugeben.

Verwendet wurde das „BMS mini“ des Herstellers „EMUS, UAB“, welches ideal für kleinere Anwendung ist. Das BMS kann maximal 16 Zellen überwachen (wird auch 16S System genannt). Im Abschnitt 2.1.1 wurde jedoch beschrieben, dass 32 Zellen im Einsatz sind. Da aber jeweils zwei Zellen parallel geschaltet sind, können diese als eine Zelle betrachtet werden.

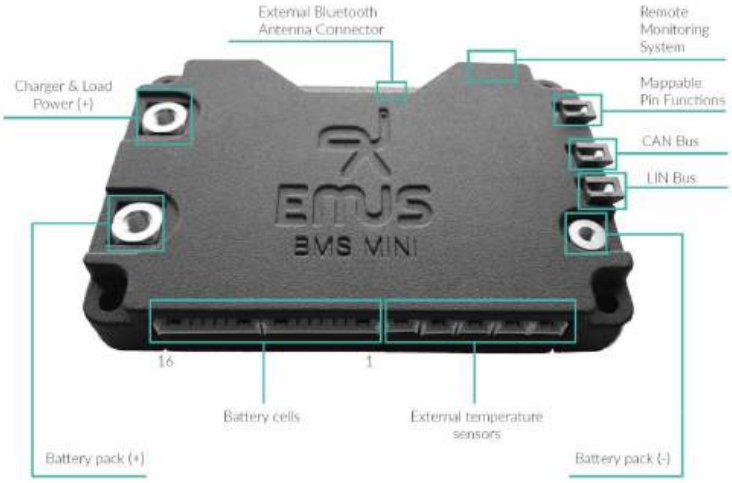


Abbildung V.8: Emus Batteriemanagementsystem[22]

2.2.1 Funktionen

In den folgenden Unterpunkten werden die einzeln Funktionen[1] des Batteriemanagementsystems aufgezählt und genauer erläutert.

Balancing:

Im Kapitel *Stand der Technik* Abschnitt 2.2.2 wurde „active“ und „passive Balancing“ bereits erläutert. Das bei diesem Projekt benutzte Batteriemanagementsystem, verwendet „passive Balancing“.

Spannungsmessung:

Lithium-Eisen-Phosphat Zellen sind sehr anfällig gegen Überspannung und Unterspannung, deswegen hat das BMS die Aufgabe die Spannung jedes einzelnes Zellenpaar zu messen und bei gegebener Über- oder Unterspannung abzuschalten. Der Überspannungsschutz wird dann eingeschaltet, wenn es möglicherweise zu ungewollter Rückspeisung kommt. Unterspannungsschutz tritt ein, wenn der Unterspannungsparameter unterschritten wird, danach wird die Last von dem Akku getrennt, sodass sich der Akku nicht mehr weiter entladen kann. Die Spannungsmessung wird ebenfalls verwendet, um die Spannungen der einzelnen Zellen zu messen, welche später dargestellt werden. (siehe *Fernsteuergerät* Abschnitt 1.1)

[1] vgl. [22], S.14ff.

Strommessung:

Beim BMS ist ein interner Strommesser eingebaut, der den Entlade- und Ladestrom misst. Der Ladestromschutz wird aktiviert wenn – wie bereits beim Überspannungsschutz erwähnt wurde – es zu einer Rückspeisung kommen würde. Wenn der Akku zu viel Strom abgeben würde, wird der Entladestromschutz aktiviert. Die Strommessung wird auch benötigt, um den Ladezustand anzuzeigen.

Temperaturmessung:

LiFePo4 Akkus neigen dazu, sich durch eine große Stromentnahme zu erwärmen, dabei kann es sein, dass sich die Akkus bei hohen Temperaturen selbst Schaden zufügen. Daher muss die Temperatur des Akkus überwacht werden und bei zu hoher oder niedriger Temperatur wird der Akku von der Last getrennt, um weitere Erwärmungen zu vermeiden.

2.2.2 Verschaltung

Die Verschaltung des Batteriemanagementsystems wurde aus dessen Datenblatt entnommen[2].

Da die Spannung von jeder einzelnen Zelle gemessen werden muss, wird jede Zelle mit dem BMS verbunden. Um eine sichere Installation durchzuführen ist es wichtig zu beachten, dass der Minuspol des Akkus zuerst mit dem Terminal (-) verbunden wird. Anschließend verbindet man den Pluspol des Akkus mit dem Terminal (+). Als letztes werden die einzelnen Zellen, beginnend mit der Zelle 1, mit dem BMS verbunden.

Um eine Tiefentladung zu vermeiden, wird zwischen dem Pluspol des Akkus und dem Terminal (+) eine Sicherung eingebaut, diese dient dazu den Akku über einen längeren Zeitraum von der Last zu trennen. Dies ist jedoch nur eine zusätzliche Schutzmaßnahme, da das BMS bereits einen Tiefentladeschutz integriert hat.

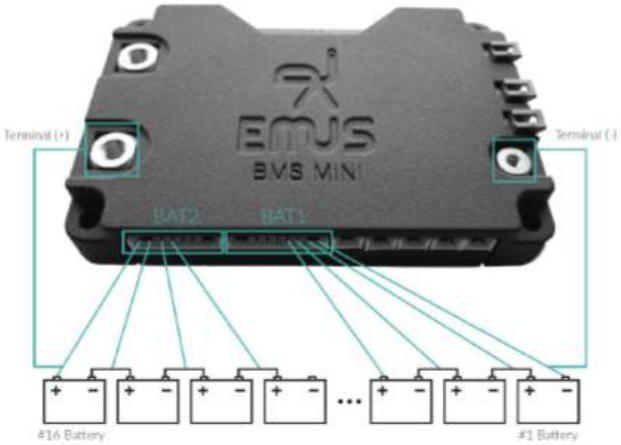


Abbildung V.9: Anschlussplan der Batteriezellen Spannungsmessung[22]

Weiters müssen noch die Temperatursensoren am BMS angeschlossen werden. Diese werden gleichmäßig innerhalb des Akkus verteilt angebracht und mit dem BMS verbunden.

[2] vgl. [22], S. 6ff.

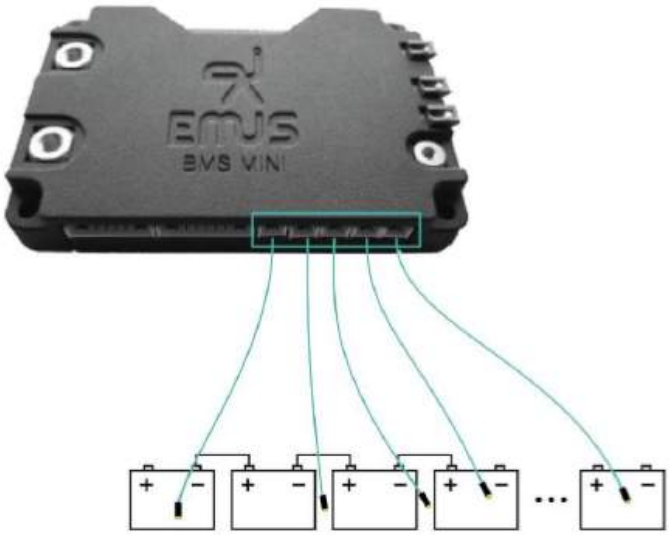


Abbildung V.10: Anschlussplan der Batteriezellen Temperaturmessung[22]

Zum Schluss wird die Last am Batteriemanagementsystem angeschlossen. Dabei wird das Power (+) terminal mit dem Load (+) terminal (Pluspol der Last) verbunden. Ebenso wird das Battery (-) terminal mit dem Load (-) terminal (Minuspol der Last) verbunden. Man sollte dabei beachten, dass die Last nicht sofort Strom benötigt, dies wird mittels eines Leitungsschutzschalters zwischen der Last und dem Power (+) terminal realisiert. Näheres dazu im Abschnitt 3.2.

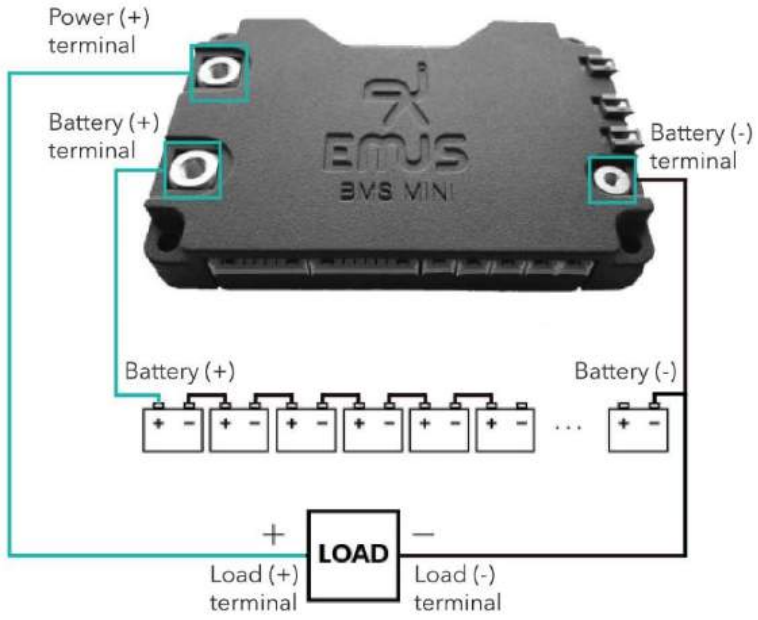


Abbildung V.11: Anschlussplan der Last[22]

Wie bereits in Kapitel Topologie 3 erwähnt wurde, kommuniziert das BMS mittels CAN-Bus mit der SPS. In Abbildung V.12 ist die Topologie des CAN-Netzwerkes abgebildet. Im Abschnitt 2.3 wird die Verbindung des Ladegerätes mit dem BMS genauer erklärt.

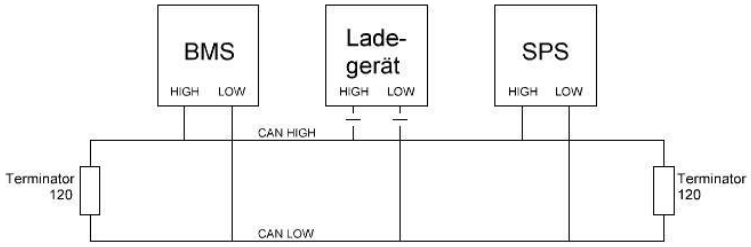


Abbildung V.12: High Speed CAN-Netzwerk

Um bei einer Übertragung von Signalen über den CAN-Bus keine Übertragungsprobleme zu bekommen, muss am Ende zwischen CAN High und CAN Low ein Leitungsabschlusswiderstand (Terminator) integriert werden. Da das BMS und die SPS dauerhaft verbunden sind, wird an diesen zwei Komponenten der Terminator angebracht. Bei der SPS erfolgt dies durch einen Schalter, der sich auf dem Bus Basis Modul befindet. Wenn dieser aktiviert ist, sollte am Einspeisemodul eine LED „T“ leuchten. Um bei dem Batteriemanagementsystem den Widerstand zu aktivieren, muss der zweite und fünfte Pin (Abb. V.14) miteinander verbunden werden[3].

Verwendet wurde anschließend ein geschirmtes Kabel mit vier Adern (Weiß, Blau, Rot und Schwarz). Um später keine Fehler bei der CAN-Verbindung zu bekommen, muss eine fixe Belegung festgelegt werden.

Weiß...CAN high
Blau...CAN low
Rot...12 V
Schwarz...Ground

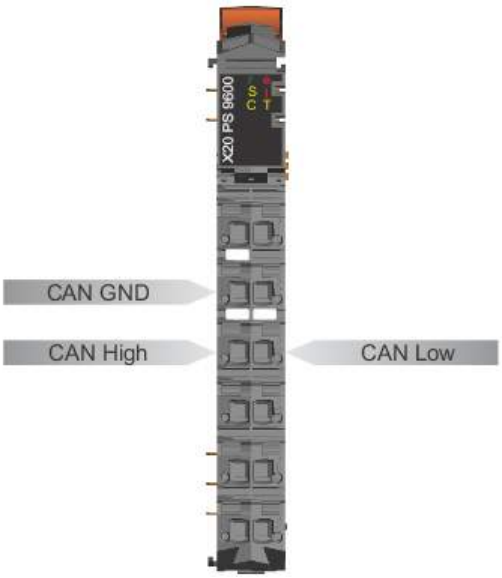


Abbildung V.13: Anschluss CAN-Bus an der SPS [5]

[3] [5], vgl.

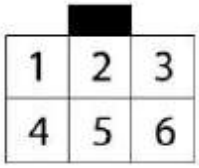
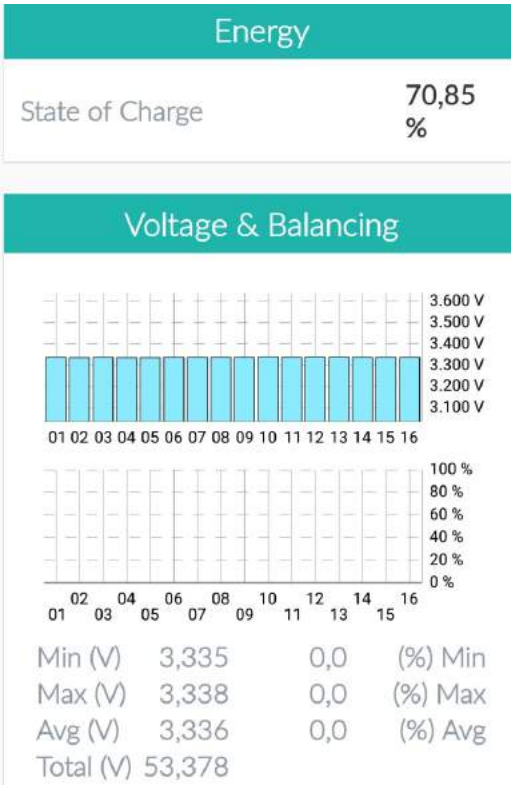


Abbildung V.14: CAN-Bus Steckerbelegung BMS[22]

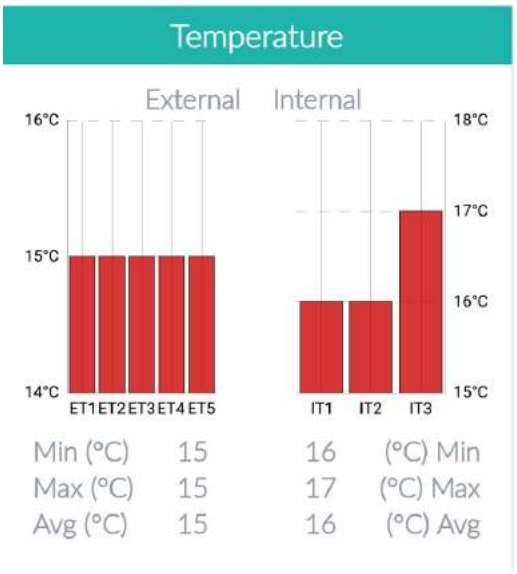
1. 12 V Versorgung
2. CAN Terminator
3. CAN high
4. Ground
5. CAN Terminator
6. CAN low

2.2.3 BMS mini Applikation

Für die „EMUS BMS Mini“-Applikation wird ein Android Handy mit Bluetooth benötigt. Die App kann im „Play Store“ heruntergeladen und anschließend installiert werden. Wenn man das BMS wie in Abbildung V.9 angeschlossen hat, schaltet sich das BMS ein. Danach kann man sich über die Applikation mit dem Batteriemanagementsystem verbinden. In der App kann das BMS konfiguriert werden (mehr dazu im Abschnitt 2.2.4). Weiters wird auch der Ladezustand sowie die Temperatur des Akkus angezeigt. (siehe Abbildung V.15)



(a) Spannungszustand & Balancing



(b) Temperaturzustand

Abbildung V.15: EMUS BMS Mini Applikation

Ebenso werden in der App jegliche Events aufgezeichnet (z.B. total discharge und charge). Auch der derzeitige Strom der aus oder in die Batterie fließt wird angezeigt.

2.2.4 Konfiguration

Die Konfiguration[4] findet ausschließlich über die BMS mini App statt. Im Abschnitt BMS mini Applikation 2.2.3 wurde bereits erklärt, wie eine Verbindung mit dem BMS hergestellt werden kann. Folgende Parameter wurden konfiguriert:

Battery Cells Settings:

Number of Cells: 16
Capacity 20,0 Ah
Maximum Voltage: 3,600 V
Minimum Voltage: 2,500 V
Maximum und Minimum Voltage gibt den Spannungsbereich der Zellen an.

[4] vgl. [22], S. 17ff.

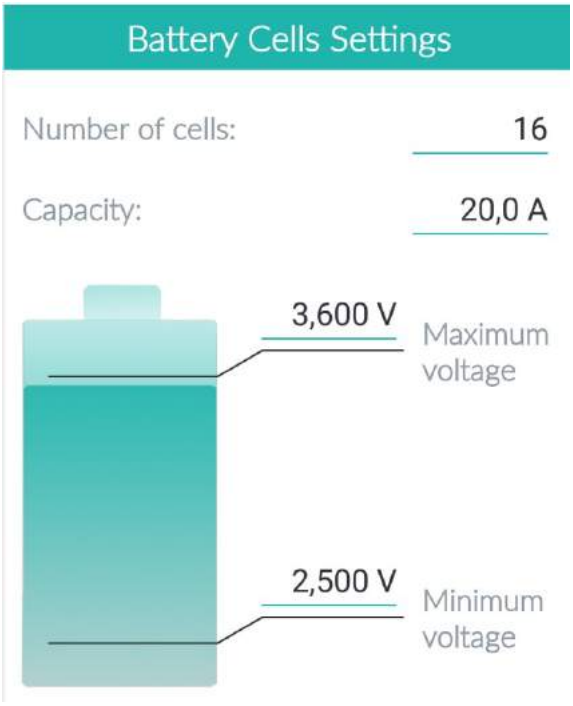


Abbildung V.16: Battery Cells Settings

Voltage Settings:
Balancing threshold: 3,200 V
Das BMS beginnt mit dem Balancing Vorgang wenn einer der Zellen den Schwellwert überschreitet.

Recharging threshold: 3,000 V
Dieser Schwellwert gibt an, bei welcher Spannung der Akku wieder aufgeladen wird, falls ein Ladegerät angeschlossen ist.

Disbalance: 15 mV

Current Settings:
Maximum charging current (continuous): 20,0 A

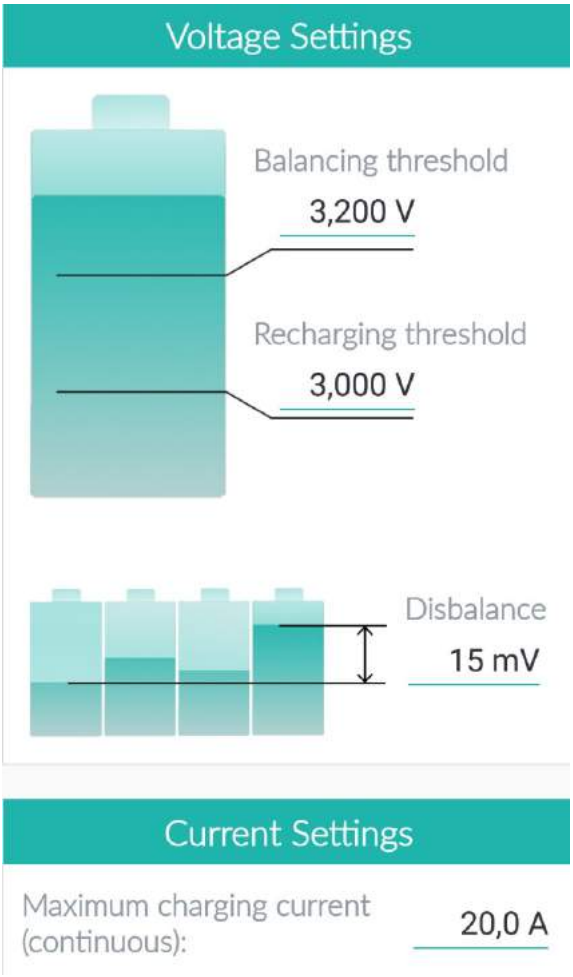


Abbildung V.17: Voltage & Current Settings

Protections:

Auf Basis der oben genannten Eingaben werden anschließend automatisch die Over/Under-Voltage und Over-Charging und Discharging Current Protection berechnet. Activation ist der Schwellwert bei dem der Schutz aktiviert wird. Fällt der Wert wieder unter den Deactivation Schwellwert, wird der Schutz wieder deaktiviert und dies jeweils mit einer bestimmten Verzögerung.

Over Voltage:
Activation: 3,600 V
Deactivation: 3,500 V
delay: 500 ms

Under Voltage:
Activation: 2,500 V
Deactivation: 2,550 V
delay: 500 ms

Charging Over Current:
Activation: 20,0 A
Deactivation: 19,0 A

delay: 200 ms

Discharging Over Current:
Activation: -20,0 A
Deactivation: -19,0 A
delay: 200 ms

Weiters müssen auch der Temperaturschutz konfiguriert werden.

Over Temperature:
Activation: 40 °C
Deactivation: 35 °C
delay: 1000 ms

Under Temperature:
Activation: 10 °C
Deactivation: 14 °C
delay: 1000 ms

2.3 Laderegelung

Eine Laderegelung[5] wird benötigt, um den Akku aufzuladen. Es gibt mehrere verschiedene Arten von Laderegelungen, die mit den BMS kompatibel sind.

1. „uncontrolled Non-CAN charging“
2. „controlled Non-CAN charging“
3. „CAN-based charging“

Bei der ersten Art wird ein Ladegerät verwendet, das direkt an die Batterie angeschlossen ist. Bei dieser Methode hat das BMS keinen Einfluss auf die Ladespannung und Ladestrom.

Bei der zweiten Art wird ein Ladegerät verwendet, das mit zusätzlichen Steuerungskomponenten verbunden ist. Bei dieser Methode kann das BMS das Ladegerät aktiveren und deaktivieren.

Bei der dritten und letzten Art wird ein Ladegerät verwendet, welches zusätzlich mittels CAN-Bus mit dem BMS verbunden ist. Diese Methode ist am effizientesten, da das BMS Ladestrom und Ladespannung genau vorgeben kann. Ebenfalls ist es die schnellste und sicherste Methode, die Akkus aufzuladen.

Verwendet wurde anschließend ein TC CAN-Charger, welcher kompatibel mit dem Emus BMS mini ist. Das Ladegerät besitzt einen Spannungsbereich von 18-68 V DC sowie einen Ladestrom von 25 A und ist damit zum Laden des eingesetzten Akkus geeignet.

[5] vgl. [60], S. 1ff.

Abbildung V.18: TC CAN-Charger[60]

2.3.1 Ladeprinzip

Abbildung V.19 zeigt das grundsätzliche Ladeprinzip des TC CAN-Chargers. Diese Methode wird „CC-CV Charging“ (constant current, constant voltage) genannt, welche geeignet für Lithium-Eisen-Phosphat Akkus ist.

Wenn der Akku sich zu tief entladen hat, bzw. die Spannung zu gering ist (unter dem Wert U1 liegt), beginnt der Charger mit dem „Pre-charge“ Vorgang. Dabei wird mit einem geringen Strom I1 die Spannung langsam bis zu U2 erhöht. Wenn dies nicht der Fall ist, wird dieser Vorgang übersprungen. Wenn die Spannung anschließend den Wert U2 erreicht hat, beginnt der Charger mit dem „CC-charging“. Bei diesem Vorgang wird ein konstanter Strom verwendet, um den Akku aufzuladen, die Ladespannung steigt dadurch bis U3. Danach endet der „CC-charging“ Vorgang. Zum Schluss beginnt der „CV-charging“ Vorgang, hierbei wird die Spannung konstant gehalten und der Strom wird langsam auf den Wert I3 verringert. Wenn dieser erreicht ist, wird der Ladevorgang beendet.

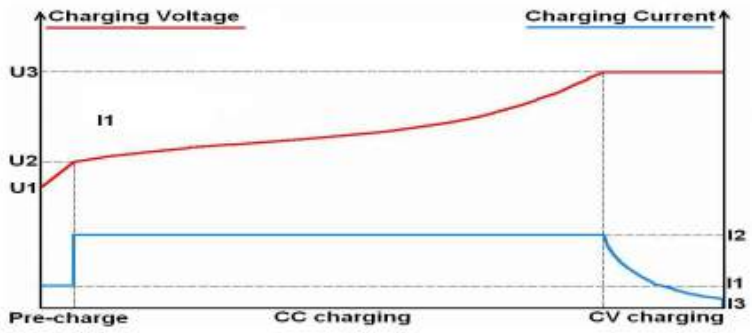


Abbildung V.19: Ladevorgang[60]

2.3.2 Verschaltung

Wie bereits erwähnt, wird das Ladegerät mittels CAN-Bus mit dem BMS verbunden, dieser dient jedoch nur zur Kommunikation für die Regelung des Ladevorgangs. Das Ladegerät wird über das BMS mit dem Akku verbunden (siehe Abb.V.20).

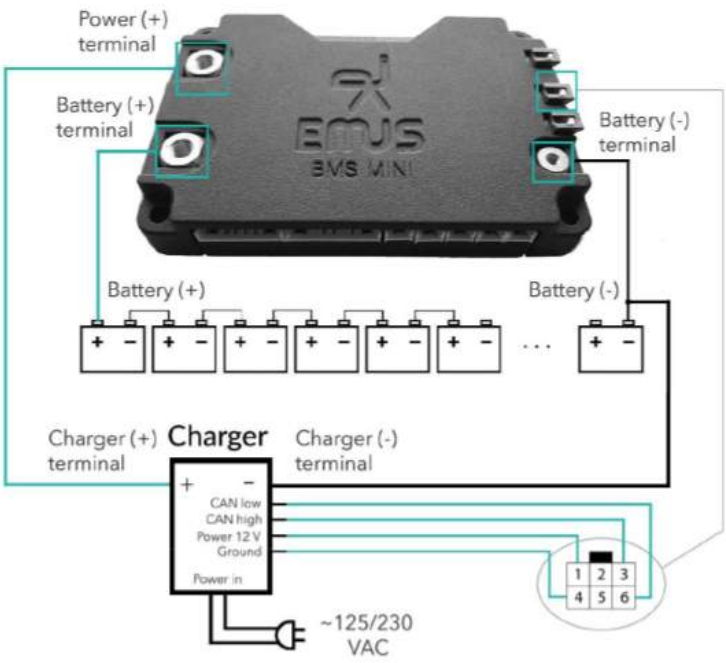


Abbildung V.20: Anschlussplan Charger[60]

Der CAN-Charger hat 3 Anschlüsse.

Der erste Anschluss wird verwendet, um dem Charger an einer Steckdose anzustecken (Input Connector V.21). Dabei wurde ein dreipoliges Kabel mit einem Schuko-Stecker[6] verwendet.

Fire Line...Außenleiter (Braun)
GND...Erdung (Gelb-Grün)
Null Line...Neutralleiter (Blau)

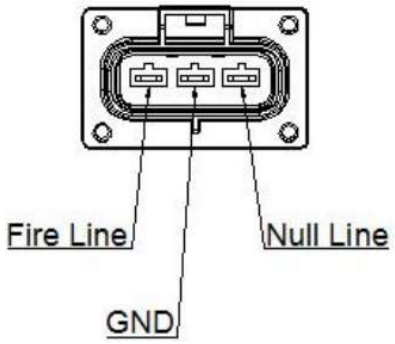


Abbildung V.21: Charger Steckerbelegung Input[60]

[6]kurz für Schutzkontakt, herkömmlicher Stecker für die Steckdose mit Erdungsanschluss

Der zweite Anschluss (Output Connector V.22) wird, wie bereits erwähnt, mit dem BMS verbunden. Output + wird mit dem BMS Anschluss Power (+) terminal verbunden und Output - mit dem Battery (-) terminal. Für diese Verbindung wurde ein schwarzer Draht für den Pluspol und ein blauer Draht für den Minuspol verwendet. Um eine Steckverbindung zwischen CAN-Charger und Batteriemanagementsystem zu schaffen, wurde ein Stecker mittels Wandmontage an das Gehäuse angebracht.

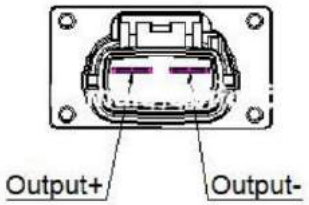


Abbildung V.22: Charger Steckerbelegung Output[60]

Der letzte Anschluss (Signal Connector V.23) wird für den CAN-Bus verwendet. Wie man in der Abbildung erkennen kann, könnte dieser Anschluss auch für andere Zwecke benutzt werden. Für den CAN-Bus jedoch werden nur vier Kontakte benötigt: CAN H, CAN L, 12V+ und LED GND or 12V GND. In Abschnitt 2.2.2 wurde bereits die fixe Belegung des CAN-Bus Steckers erwähnt.

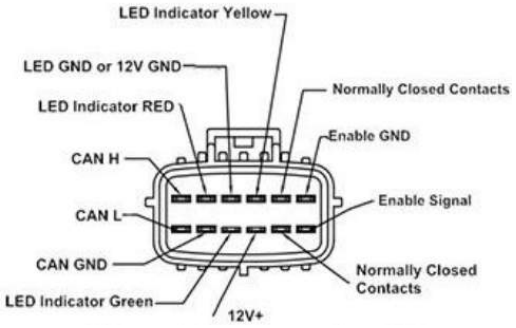


Abbildung V.23: Charger Steckerbelegung Signal[60]

Um eine Steckverbindung zwischen CAN-Charger und CAN-Bus des Batteriemanagementsystems zu schaffen, wurde eine female D-Sub-Buchse und ein male D-Sub-Stecker verwendet. Die female D-Sub-Buchse wurde mittels Wandmontage am Gehäuse des Robotersystems angebracht. In Abbildung V.24 wird die Belegung des D-Sub-Steckers dargestellt.

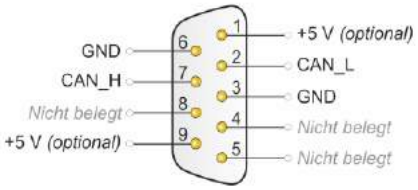


Abbildung V.24: D-Sub-Stecker Belegung[3]

Dabei wurde Anschluss 1 für +5V, 2 für CAN low, 3 für Ground und 7 für CAN high verwendet.

3 Elektrische Installation

In diesem Abschnitt wird die elektrische Installation erläutert, diese beschreibt, wie die verschiedenen Komponenten miteinander verbunden sind. Der vollständige Schaltplan ist im Anhang zu finden.

3.1 Dimensionierung

Die Dimensionierung der Leitungen ist ein wichtiger Bestandteil der elektrischen Installation. Die Leitungen müssen so dimensioniert werden, dass sie den berechneten Nennstrom aushalten. Im Abschnitt Batteriemanagement Dimensionierung 2.1.1 wurde bereits der maximal auftretende Nennstrom berechnet, dieser beträgt 27 A. Verwendet wurde anschließend ein Leitungsschutzschalter mit dem nächst größerem Bemessungsstrom (C32). Der Buchstabe vor dem Bemessungsstrom bezeichnet die Auslösecharakteristik, diese gibt an, wie schnell der Leitungsschutzschalter auslöst. Typ C ist geeignet für Stromkreise mit Motoren, da diese höhere Anlaufströme benötigen[7].

Die Leitungen wurden dann entsprechend dem Leitungsschutzschalter 32C dimensioniert. Die darunter folgende Abbildung V.25 zeigt einen Ausschnitt aus der TAEV.

	Aderleitungen und Mantelleitungen										flexible Leitungen	Kabel	
Isolierwerkstoff	PVC												
Zulässige Betriebstemperatur	70 °C												
Umgebungstemperatur	25 °C												20 °C
Verlegeart	A		A2		B		B2		C		Haushalts- und Handgeräte	Kabel in Luft	Kabel in Erde
belastete Adern	2	3	2	3	2	3	2	3	2	3	2	3	3
Nennquerschnitt (mm²) Cu	maximal zulässiger Nennstrom einer Überstrom-Schutzeinrichtung mit der **Auslösekennlinie B/C/D** in Ampere												
1,5	13	13	13	13	16	16	16	16	16	16	16	16	25
2,5	20	16	16	16	25	20	20	20	25	25	20	25	25
4	25	25	25	20	25	25	25	25	35	35	25	35	40
6		25		25		35		35		40		40	50
10		40		35		50		50		63		63	63
16		50		50		63		63		80		80	80
25		63		63		80		80		100		100	100
35		80		80		100		100		125		125	125
Nennquerschnitt (mm²) Cu	maximal zulässiger Nennstrom einer Überstrom-Schutzeinrichtung mit der **Auslösekennlinie gG**(gL) *) bzw. L/U **) in Ampere												
1,5	12	12	12	12	12	12	16	12	16	12	12	16	20
1,5*)	13	13	13	13	13	13	16	13	16	13	13	16	20
2,5	16	16	16	16	20	16	20	16	20	20	16	20	25
4	20	20	20	20	25	25	25	20	25	25	20	25	35
6		25		25		32		25		35		35	50
10		35		35		40		40		50		50	63
16		50		40		50		50		63		63	80
25		63		50		63		63		80		80	100
35		80		63		80		80		100		100	125

*) gilt für Sicherungen mit Auslösekennlinie gG gemäß Reihe ÖVE/ÖNORM EN 60269 (Ersatz für Sicherungen mit Auslösekennlinie gL gemäß der zurückgezogenen ÖVE-SN 40).

**) Leitungsschutzschalter mit Auslösekennlinie L bzw. U sowie Sicherungen mit Auslösekennlinie gL können in bestehenden Anlagen weiterhin eingesetzt werden. Es wird jedoch darauf hingewiesen, dass für derartige Leitungsschutzschalter die Konformitätsvermutung nach der Niederspannungsgeräteverordnung nicht gegeben ist.

Abbildung V.25: Leitungsdimensionierung[59]

[7] vgl. [34], Abschnitt 6.2 Leitungsschutzschalter.

Um den Leitungsquerschnitt zu dimensionieren, muss eine Verlegeart bestimmt werden. Die Verlegeart C wurde ausgewählt.

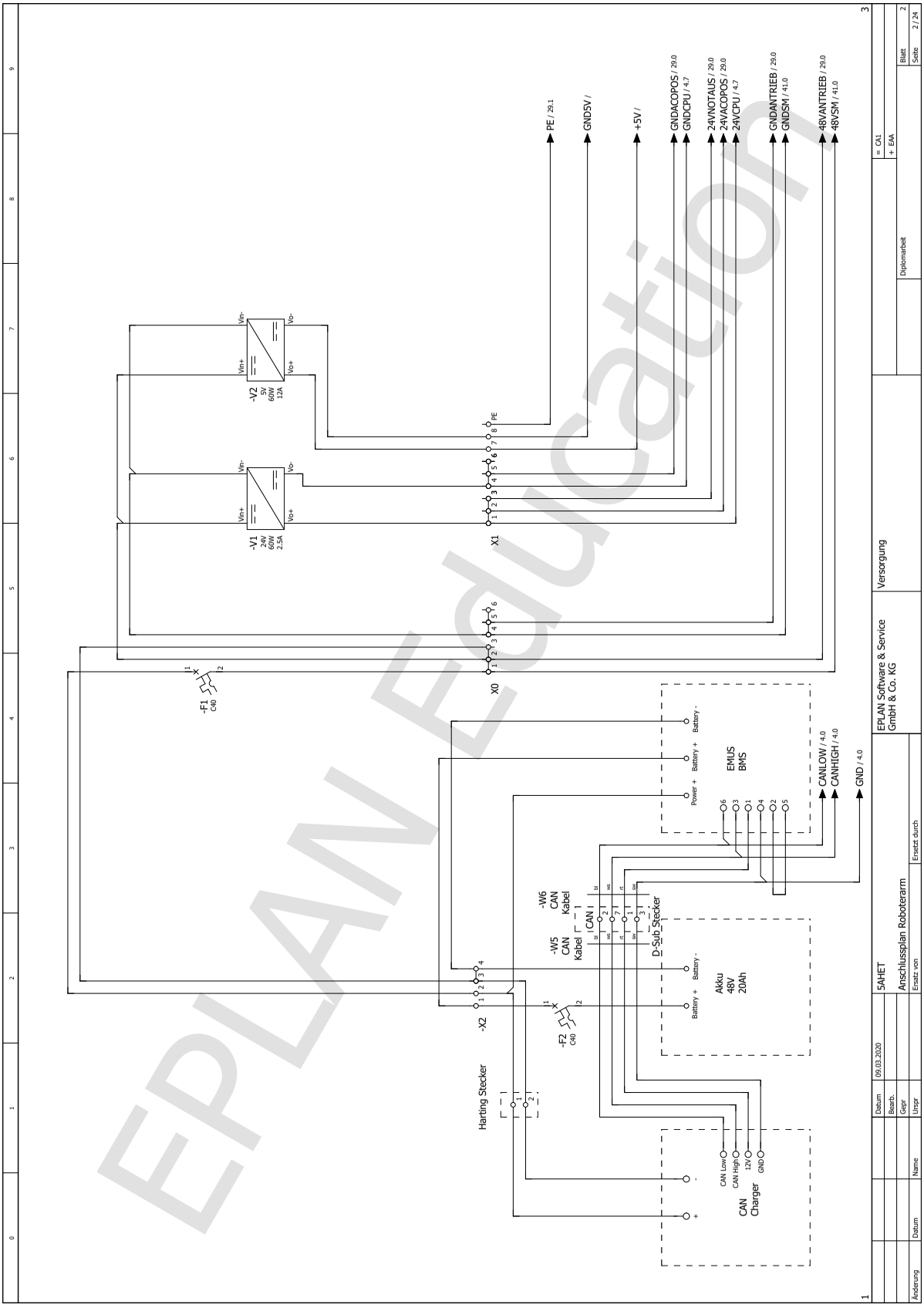
Definition: „mehradrige Mantelleitungen ohne Rohr unter Putz oder frei oder in offenen Kanälen auf Wänden, Decken oder Fußböden; Stegleitungen unter Putz.“[8]

Unter der Bedingung zwei belastende Adern ergibt sich ein Nennquerschnitt von $4mm^2$. Alle Leitungen vom Akku bis zum Klemmenblock -X0 werden mit einem Querschnitt von $4mm^2$ verlegt. Die restlichen Leitungen die ein Potenzial von 48 V führen, „48VANTRIEB“, „48VSM“ und deren Rückleitungen werden mit einem Nennquerschnitt von $2,5mm^2$ verlegt. Die Leitungen, die nach den beiden Gleichspannungswandler liegen, wurden mit $1mm^2$ dimensioniert, da diese hauptsächlich Steuerleitungen mit wenig Strom sind.

3.2 Versorgung

In diesem Unterpunkt wird ein Ausschnitt des Schaltplans gezeigt. Dieser beinhaltet die Versorgung, deren Klemmenabgänge und einen Teil der Verschaltung des Batteriemanagements.

[8] [59].



In der linken unteren Ecke befindet sich der Akku, das Batteriemanagementsystem und das Ladegerät. Wie bereits erwähnt fungieren diese drei Komponenten gemeinsam als Akkusystem. Weiters sind auch die zwei Stecker zu erkennen, welche per Wandmontage an eine Seitenplatte des Robotersystems angebracht wurden. Nach dem Akku folgt ein Leitungsschutzschalter (-F2), welcher rein als Abschalter zwischen Akku und Batteriemanagementsystem fungiert. Danach folgt der Klemmenblock -X2, dieser wird benötigt, um die Installation zu erleichtern.

Der Leitungsschutzschalter -F1 dient als Überstrom- und Kurzschlussschutz. Danach folgt der Klemmenblock -X0, diese Klemmen werden als Verteiler für die 48 V Versorgung benutzt. Von diesen Klemmen aus werden die zwei Gleichspannungswandler versorgt. Die 24 V und 5 V werden wieder auf den Klemmen von Klemmenblock -X1 verbunden und als Verteiler benutzt. Zum Schluss sind noch die Abgänge zur Receiver Platine (5 V), zu den Wechselrichtermodulen (24 V), zur Compact CPU (24 V), zum Antrieb (48 V) und zu den Schrittmotormodulen (48 V) eingezeichnet.

3.3 SPS Verschaltung

Im Kapitel *Robotersystem* Abschnitt 3 wurde bereits der Aufbau der SPS erklärt. In diesem Unterpunkt wird nun die nötige Verschaltung erläutert.

3.3.1 Compact-S CPU Verschaltung

Am X20 Einspeisemodul wird die CPU grundsätzlich mit 24 V versorgt. Diese 24 V werden von einem der zwei DC-DC-Wandler zur Verfügung gestellt. An diesem Modul sind zusätzlich noch die Anschlüsse für den CAN-Bus, der mit dem Batteriemanagementsystem verbunden ist, und der RS-232 Anschluss für die Receiver Platine vorhanden.

3.3.2 Schrittmotormodul Verschaltung

Die Schrittmotormodule sind zuständig für die Steuerung des Roboterarms. Diese kommunizieren mittels X2X-Link mit der CPU. Die Schrittmotoren des Roboterarms werden mit einer Spannung von 48 V betrieben, diese Spannung wird direkt vom Akku zur Verfügung gestellt. Im darunter folgenden Bild wird ein Anschlussbeispiel von einem Schrittmotormodul gezeigt.

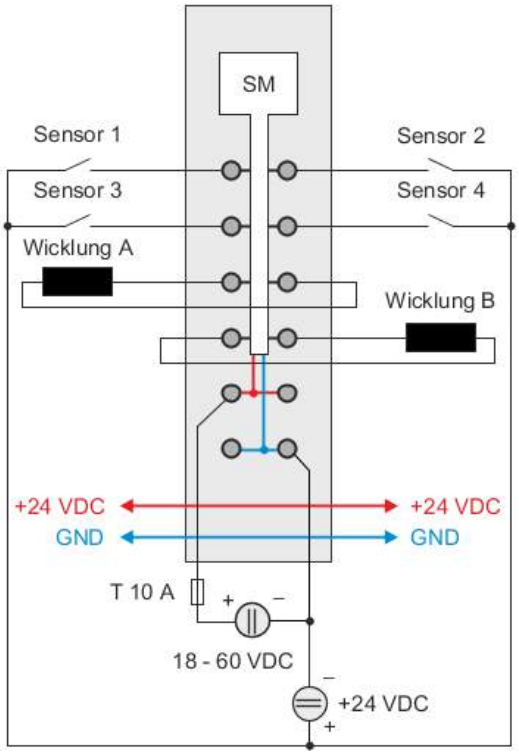


Abbildung V.26: Anschlussbeispiel Schrittmotormodul [5]

Die Anschlüsse 1-4 sind die Anschlüsse für den Encoder. Für den Abtriebsencoder werden sogenannte Encoder-Adapter benötigt. Die Abtriebsencoder die am Roboterarm angebracht sind, werden jeweils mit einem Encoder-Adapter verbunden. Diese werden anschließend mit dem Schrittmotormodul verbunden. Für den Encoder werden jedoch nur drei Anschlüsse benötigt.

Die Anschlüsse 5-8 sind die Anschlüsse für die Schrittmotoren am Roboterarm. Über die letzten vier Anschlüsse wird Versorgungsspannung der Schrittmotoren bereitgestellt.

Die genaue Anschlussbelegung der Encoder-Adapter ist im Anhang unter Schaltpläne zu finden.

3.4 ACOPOSmicro Verschaltung

Abbildung V.27 zeigt die Anschlussbelegung des ACOPOSmicro. Wie die Schrittmotoren vom Roboterarm werden auch die beiden ACOPOSmicro mit 48 V versorgt, also direkt vom Akku. Die 48 V werden an Klemme X1 angeschlossen. Von dieser Klemme wird auch direkt der zweite ACOPOSmicro versorgt. Der mittlere PE Anschluss wird mit einer Erdungsklemme verbunden, die sich auf der Hutschiene befindet.

An der Klemme X2 befindet sich der Anschluss für die 24 V Versorgung des CPU Teils. Ebenfalls sind diese Klemmen mit einem Not-Aus Schalter verbunden. Wenn dieser betätigt wird, bleiben die Motoren sofort stehen. 24 V Enable und COM Enable müssen auch mit 24 V versorgt werden, da diese dafür zuständig sind, die Spannung (48 V) am Ausgang zu aktivieren.

Die Anschlüsse X3A und X3B werden benötigt, um die Compact CPU mittels Powerlink

mit den ACOPOSmicro zu verbinden. Da die Compact CPU nur einen Anschluss für Powerlink hat, wird der zweite ACOPOSmicro über den ersten ACOPOSmicro mit dem Powerlink-Kabel verbunden.

Alle weiteren Verbindungen wurden mit fertigen Kabeln von B&R Industrial Automation realisiert. Jeden dieser Anschlüsse gibt es doppelt, da ein ACOPOSmicro bis zu zwei Motoren ansteuern kann. X4A und X4B sind Temperaturfühler, welche die Temperatur des Motors messen. An die Klemmen X5A und X5B werden die Synchronmotoren angeschlossen. Diese beinhalten die Anschlüsse Motorphase U, V, W und PE. Zum Schluss werden noch die Klemmen X6A und X6B, die als Gebereingänge fungieren, mit den Motoren verbunden.

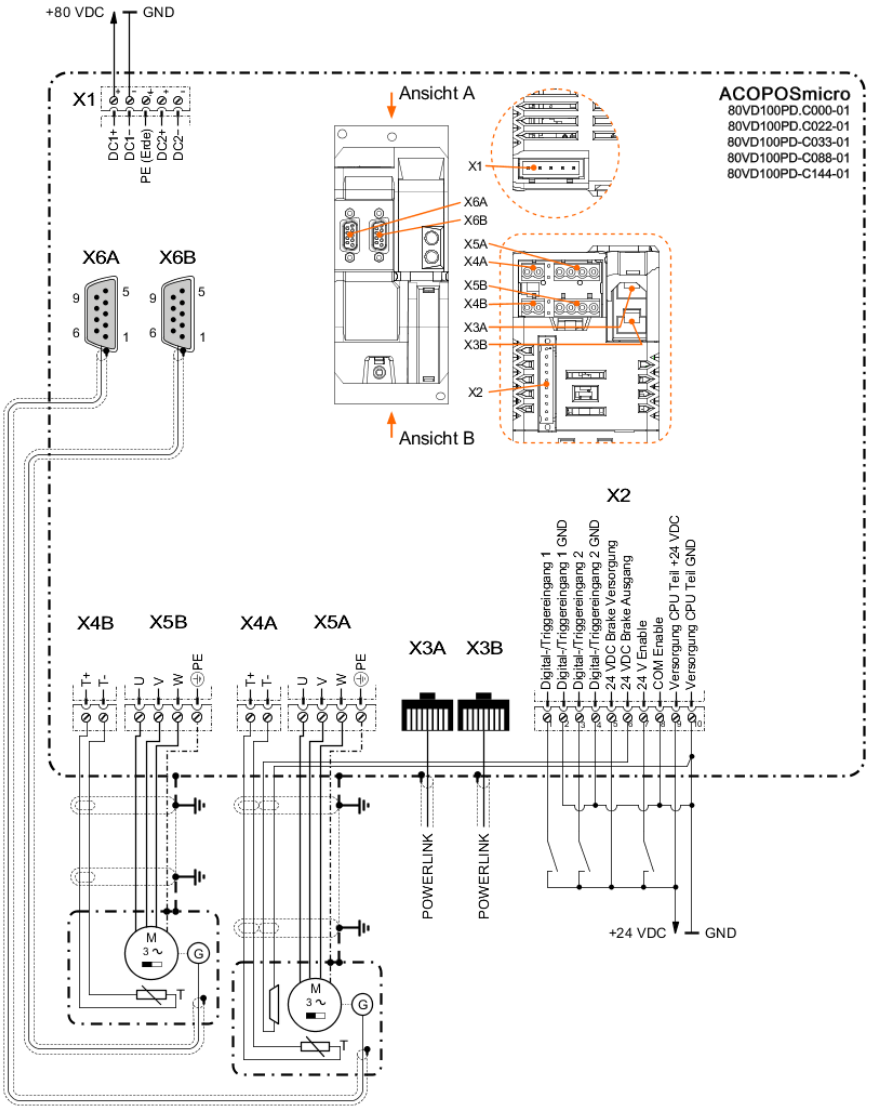
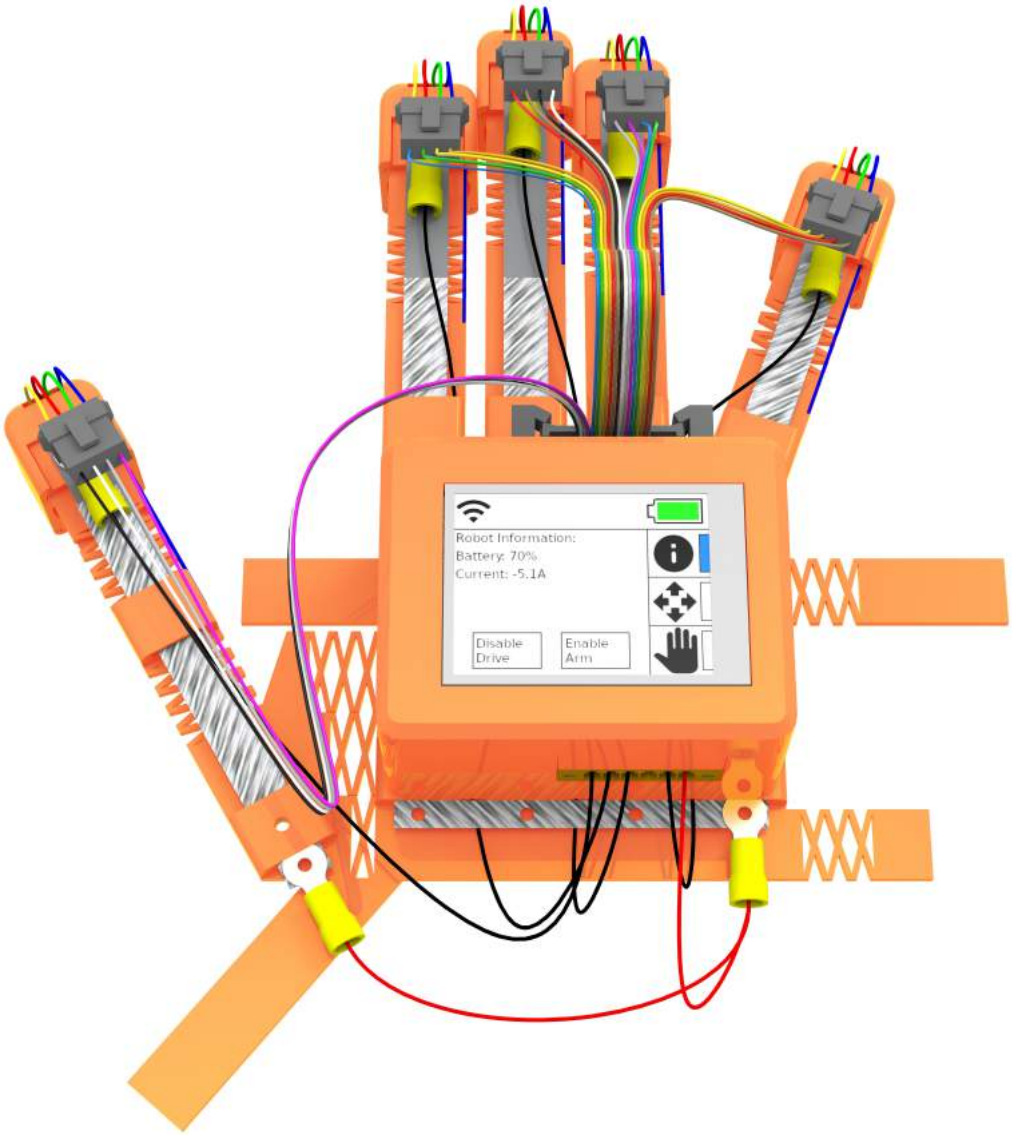


Abbildung V.27: Anschlussbelegung ACOPOSmicro [5]

VI Fernsteuergerät



1 Übersicht

Das Projekt „Fernsteuergerät“ ist in Bezug auf das Gesamtprojekt die Steuerkomponente für das Robotergreifsystem sowie für das Fahrwerk.

Das Projekt beinhaltet im Wesentlichen drei Teilbereiche:

- Mechanik
- Feedback-Implementierung
- Hardware-Design

1.1 Aufgaben des Fernsteuergerätes

Betreffend der Aufgabenstellung erfolgt die Steuerung des Robotersystems mittels Gestensteuerung, wobei die Kinematik im Kapitel *Einführung* Abschnitt 4 erklärt wurde. Dem Anwender des Systems ist es durch das Tragen des Fernsteuergerätes möglich, durch einfache Bewegungen das Gesamtsystem zu steuern. Außerdem fungiert es als Feedback-Geber für verschiedene mögliche Fälle, welche im späteren Verlauf (siehe 3) detailliert beschrieben werden. Um eine ortsunabhängige Energieversorgung zu ermöglichen, wird die Versorgung von einem Akku bezogen, welcher möglichst platzsparend in das Fernsteuergerät eingebaut wird.

Über ein Touch-Display werden sämtliche Informationen des Systems (Akkustand, Status, Modus, etc.) ausgegeben. Weiters wird die Möglichkeit gegeben, zwischen den Bedien-Modi (siehe *Einführung* Abschnitt 4.5) umzuschalten.

1.2 Aufgaben der Mechanik

Der Bereich der Mechanik beschäftigt sich mit dem Design und der Ausführung des zum Großteil 3D-gedruckten Fernsteuergeräts in Form eines Handschuhs sowie mit der Implementierung der Feedback-Funktionen und der elektronischen Komponenten.

Aufgrund der uneinheitlichen Handgrößen verschiedener Personen erfolgt ein Design-Konzept, welches für diverse Hand-Größen geeignet ist.

1.3 Aufgaben der Elektronik-Hardware

Die enthaltenen elektronischen Komponenten sind zuständig für die Aufnahme der Kinematik zur Gestenerkennung, die Steuerung der Feedbackfunktionen und den Informationsaustausch mittels Display.

2 Mechanik

2.1 Handschuh-Design

Ausgangslage für das Handschuhdesign ist, dass ein Handgrößen-universeller Handschuh mittels 3D-Druck gefertigt werden soll. Da im Verlauf der späteren Anwendung des Gesamtsystems die Möglichkeit besteht, den am Roboterarm montierten Greifer – in Form einer menschlichen Hand – und dessen 5 Finger unabhängig voneinander ansteuern zu können, ist das Handschuh-Design an diese Anforderung angepasst.

2.2 Konstruktions-Tool

Die Konstruktion der gesamten Mechanik erfolgt mit dem 3D CAD System „Solid Edge“, welches vom Herausgeber Siemens PLM an Schüler und Studenten als Vollversion zur Verfügung gestellt wird.

2.3 Baugruppen

Die Konstruktion des Handschuhs teilt sich im Wesentlichen in drei Baugruppen ein:

- Handrücken
- Elektronik-Box
- Fingerspitzen

Diese werden nun detailliert erklärt.

2.3.1 Handrücken

Den flächenmäßig größten Teil des Fernsteuergerätes nimmt der „Handrücken“ ein. Er ist der elementare Hauptteil, an dem alle anderen Komponenten montiert werden.

Grundlegend für die Konstruktion des Handrückens ist, dass alle Finger sowie der Daumen – aus Gründen der Feedback-Implementierung (siehe 3.2.2) – so vom Handrücken bedeckt werden, dass in der Achse der Bewegungsrichtung stets die „Oberseite“ des Fingers/Daumens bedeckt ist.

Wie eingangs bereits erwähnt, wird der Handrücken allerdings mit dem 3D-Druck-Verfahren gefertigt, welches im Kapitel *Stand der Technik* Abschnitt 7 beschrieben wurde. Dies bereitet insofern Schwierigkeiten, da die „Oberseite“ des Daumens nicht planar mit denen der Finger und dem Rest des Handrückens verläuft – was es für den 3D-Druck aber müsste.

Aufgrund dessen musste eine planare (2-dimensionale) Abwicklung des Daumengliedes erstellt werden, welche sich zusammen mit dem restlichen Handrücken als gesamtes Objekt drucken lässt. Durch die Eigenschaften des verwendeten flexiblen Filaments, kann nach abgeschlossenem Druckvorgang das Daumenglied wieder in die reale notwendige Position gebracht werden.

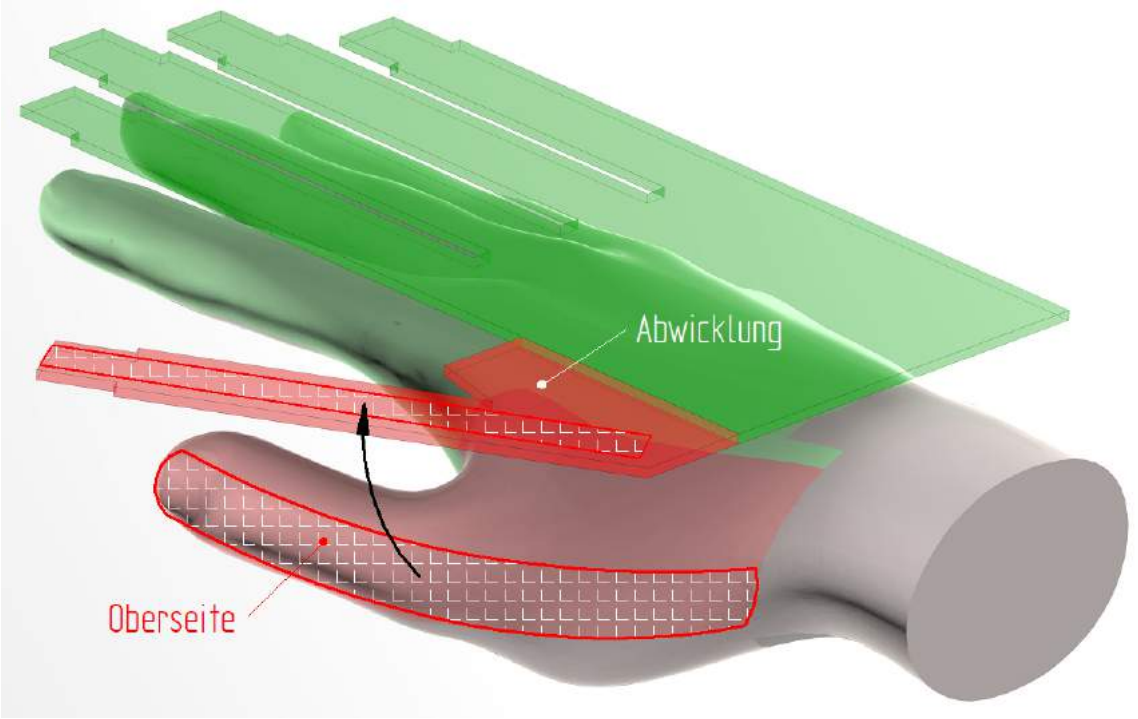
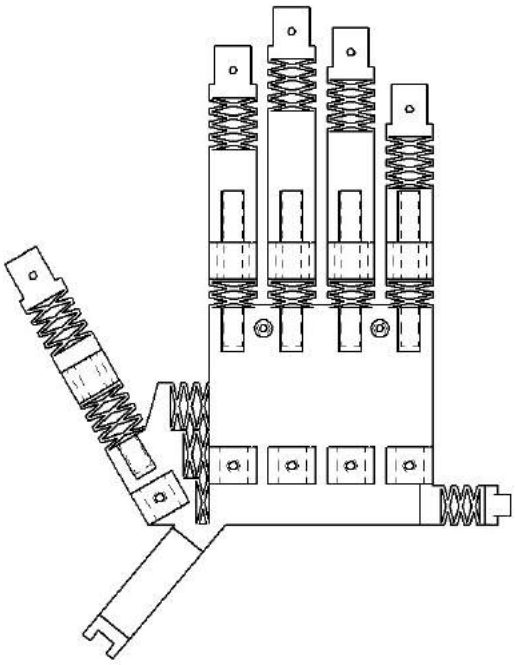
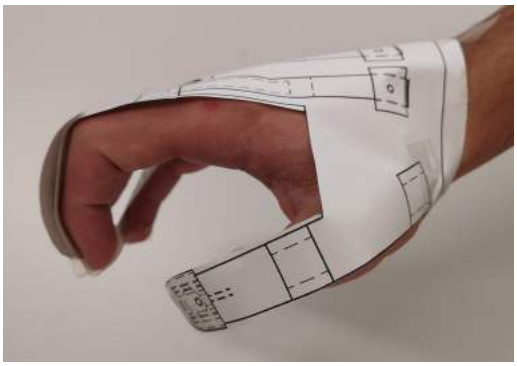


Abbildung VI.1: Darstellung der Daumen-Abwicklung

Die passenden Maße der Abwicklung konnten zwar durch ungefähre Abschätzungen angenähert, allerdings erst durch empirische Ermittlungen perfektioniert werden. Dazu wurden Maße adaptiert, der Druck des Handrückens mittels üblichen Papierausdrucks simuliert und auf die Korrektheit der Maße geprüft.



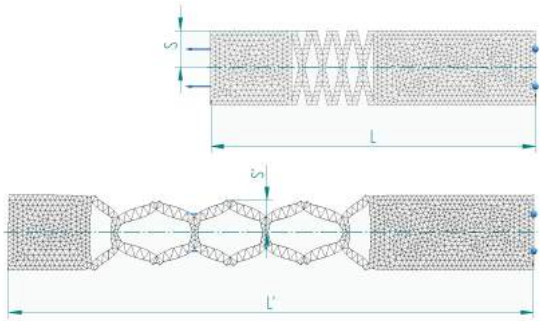
(a) Papierausdruck



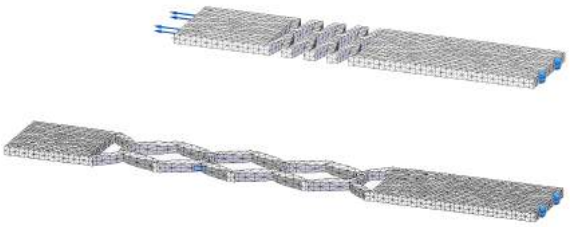
(b) Anprobe des Papier-Handschuhs

Abbildung VI.2: Papierausdruck des Handschuhs zur Ermittlung der Maße

Durch die Aneinanderreihung von mehreren „X-Mustern“ (siehe Abbildung VI.3) kann durch Zug oder Druck entlang der Fingerachse eine Längenänderung bewirkt werden, indem sich die Schenkel der X-Muster im Verlauf der jeweiligen Mittelachse zusammenziehen bzw. auseinander drücken.



(a) Verlängerungsmechanik 2D



(b) Verlängerungsmechanik 3D

Abbildung VI.3: Verlängerungsmechanik

Für die Implementierung der im Abschnitt 3 erwähnten Feedback-Funktionen, werden entlang des Handrückens (siehe Abbildung VI.4) Erhöhungen (1) vorgesehen, durch die die Feedback-Elemente geführt werden. Zur Befestigung von Sensoren werden ebenfalls (kleinere) Erhöhungen (2) angebracht. Zur Befestigung der Elektronik-Box und der Fingerspitzen dienen Bohrungen am Handrücken (3) (5). Außerdem ist am Handrücken eine Handgelenkschlaufe angebracht (4), die durch das Abwickeln des Daumengliedes das Handgelenk im vollen Umfang umschließt. Zusätzlich dient eine weitere Befestigungsschlaufe (6) dazu, einen besseren Halt des Handrückens an der menschlichen Hand zu ermöglichen.

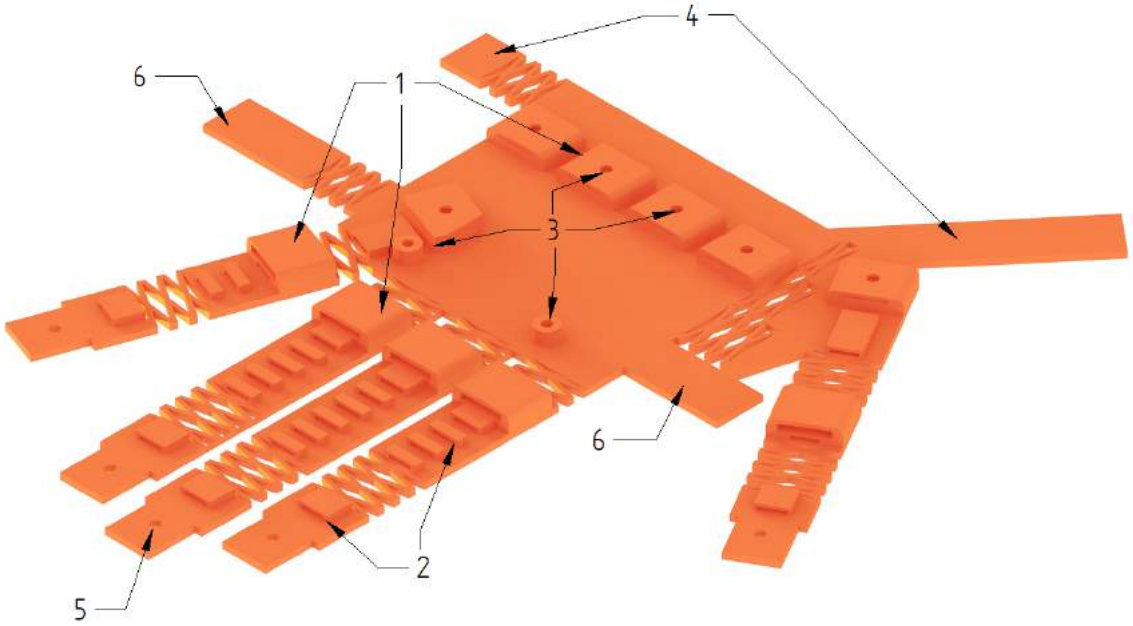


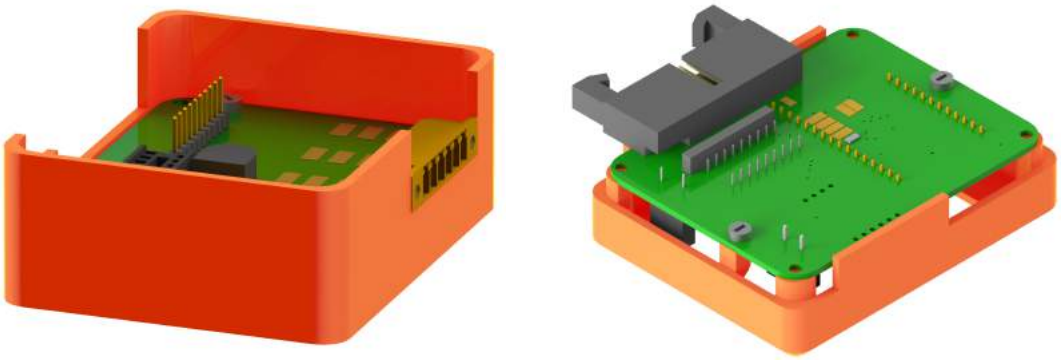
Abbildung VI.4: Handrücken

2.3.2 Elektronik-Box

Sämtliche Elektronikkomponenten (Abschnitt 4) werden möglichst platzsparend in einer Box zusammengefasst, die ebenfalls als Halterung für ein 2,4 Zoll Display dient. Diese teilt sich in einen Unter- und Oberteil, welche mittels Schraubmontage zusammengefügt werden. Die Fertigung erfolgt mittels 3D-Druck.

Die zwei verwendeten Platinen werden darin übereinandergestapelt (sogenannte „Huckepack-Platinen“) und an der Box-Innenseite montiert, wobei jeweils eine Platine an Unter-/Oberteil mittels Schraubmontage befestigt wird. Dafür sind eigene Befestigungsvorrichtungen angebracht.

(Auf die Platinen wird im Abschnitt 4.5 eingegangen.)

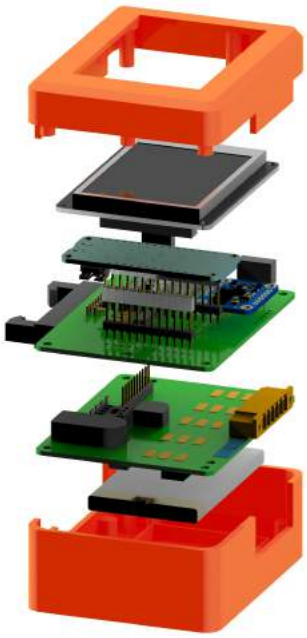


(a) Unterteil mit HV-Platine (b) Oberteil mit LV-Platine

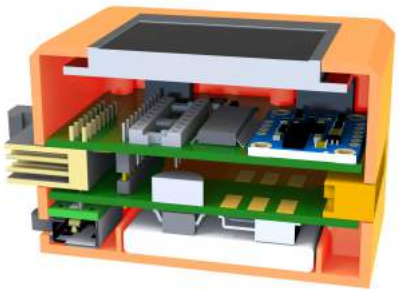
Abbildung VI.5: Unter-/Oberteil der Elektronik-Box

Außerdem befindet sich im Inneren der Box ein Akku, der für die Energieversorgung des Fernsteuergeräts verantwortlich ist, sowie ein Schiebeschalter, welcher zur Trennung der Versorgung dient.

In folgender Explosions- und Schnittdarstellung kann das gesamte Innenleben der Box deutlich betrachtet werden:



(a) Explosionsdarstellung



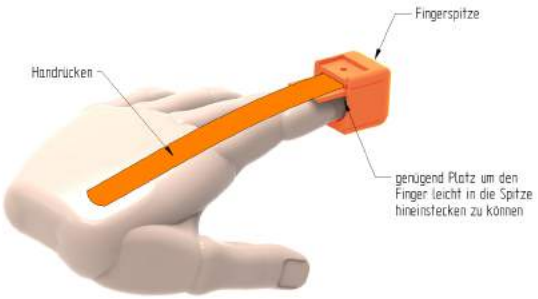
(b) Schnittdarstellung

Abbildung VI.6: Explosions- und Schnittdarstellung der Elektronik-Box

2.3.3 Fingerspitze

Die Letzte der drei Hauptkomponenten bildet die, aus zwei Einzelteilen bestehende, Fingerspitze, welche für die vier Finger in identischer Form vorliegt, und für die Spitze am Daumen in etwas größerer Skalierung ausgeführt ist.

Bei der Konstruktion der Fingerspitze war zu beachten, dass zum einen ein adäquater Halt der Finger in den Spitzen vorhanden ist, um den Teil des Handrückens analog zur Abwinkelung des Fingers mitzubewegen, und zum anderen, dass eine Implementierung der Feedbackfunktion in den Fingerspitzen möglich ist.



(a) ausgestreckter Finger

(b) abgewinkelter Finger

Abbildung VI.7: Finger-Abwinkelung

Die größte Schwierigkeit, in Bezug auf die Konstruktion der Fingerspitze, bestand darin, dass innerhalb der Spitze eine möglichst unauffällige Verkabelung des Feedbackgebers (siehe Abschnitt 3.1) und der Sensorik-Elemente (siehe Abschnitt 4.2) erfolgen soll.

So ergab sich die Lösung, dass nur einer der beiden Teile als tatsächliche Fingerspitze dient,

also der Teil, welcher die menschliche Fingerspitze voll umschließt. Sämtliche Verdrahtung wird durch diesen Teil hindurchgeführt und am hinteren Ende der Fingerspitze mit den Drähten verlötet, die dann am anderen Ende an einem Stecker kontaktiert sind.

Der zweite Teil der gesamten Fingerspitze dient in Folge lediglich zur Abdeckung der Verkabelung, sodass diese von außen nicht sichtbar ist.

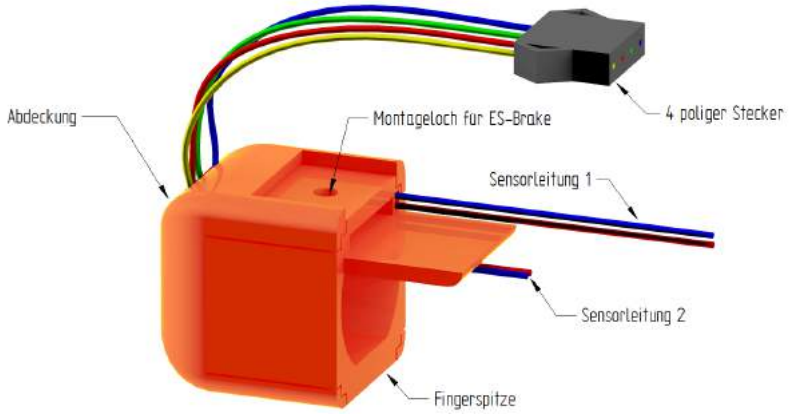
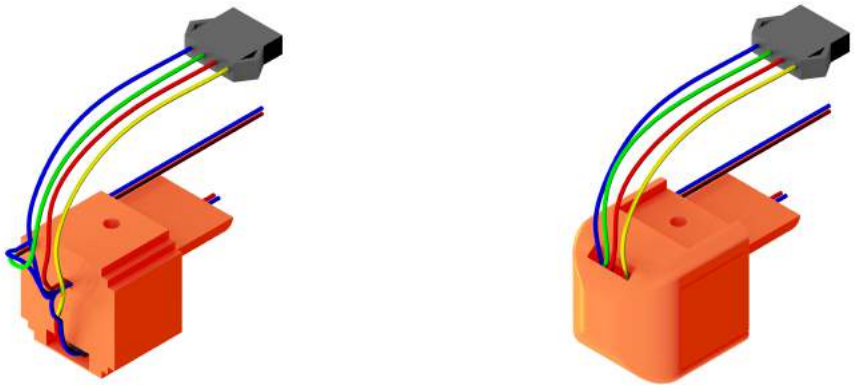


Abbildung VI.8: Fingerspitze



(a) Verkabelung ohne Abdeckung (b) Verkabelung mit Abdeckung

Abbildung VI.9: Verkabelung der Fingerspitze

3 Feedback-Funktionen

3.1 Taktiles Feedback

Für die Umsetzung eines taktilen Feedbackgebers wurde aus den in Abschnitt 4.2.1 beschriebenen Umsetzungsmöglichkeiten die einfachste ausgewählt; nämlich die mit einem kleinen DC-Motor, welcher Vibrationen verursacht. Zur Anwendung kommt ein DC-Motor – mit relativ kleinem Durchmesser von 10 mm – welcher mit einer Versorgung von 3 V DC und einem Strom von weniger als 80 mA betrieben werden kann. Sobald der Motor ausreichend versorgt ist (erfolgt durch eine Steuerelektronik, siehe Abschnitt 4.3), vibriert er in genügender Stärke, um eine deutlich wahrnehmbare Vibration auf den menschlichen Finger zu übergeben.

Abbildung VI.10: Vibrationsmotor[21]

3.1.1 Implementierung

Der taktile Feedbackgeber befindet sich in allen Fingerspitzen und wird von hinten in das Fingerspitzenelement eingeführt, sodass der Motor nach dem Aufsetzen der Abdeckung nicht mehr sichtbar ist:

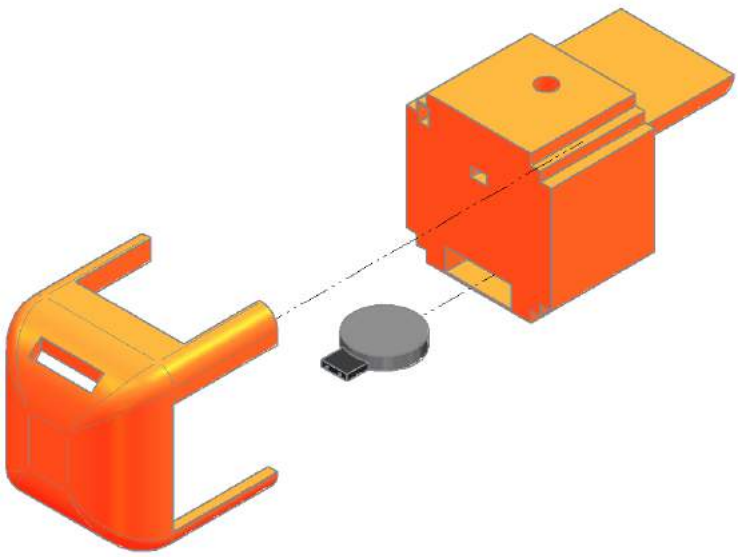


Abbildung VI.11: Explosionsdarstellung Fingerspitze und Vibrationsmotor

3.1.2 Anwendung

Zwar könnte der Feedbackgeber in zahlreichen Situationen eingesetzt werden, allerdings wird die Verwendung auf einen konkreten Fall beschränkt. Nämlich, nur wenn der Greifer am Roboter (siehe *Robotersystem* Abschnitt 5.3) einen Gegenstand umschlossen hat, soll das Feedback gegeben werden. Damit kann gegenüber dem Anwender des Fernsteuergerätes deutlich indiziert werden, dass eine Berührung erfolgt ist, und der Greifer nicht mehr weiter schließen muss.

3.2 Kinästhetisches Feedback

Für die Umsetzung des kinästhetischen Feedbacks wurde die publizierte Variante des sogenannten „DextrES: Wearable Haptic Feedback“ gewählt, welche bereits im Kapitel *Stand der Technik* Abschnitt 4.2.2 erläutert wurde. Die Bezeichnung „Electrostatic Brake“ – kurz: ES-Brake – sowie einige Bauteilbezeichnungen werden aus besagter Arbeit übernommen.

Dadurch, dass eine derartige Anwendung noch durchaus als experimentell bezeichnet werden kann und kommerziell nicht erhältlich ist, musste diese selbst erstellt werden, wobei zahlreiche Probleme auftraten, die im Abschnitt 3.2.3 beschrieben werden.

3.2.1 Funktionsprinzip

Wie im besagten Abschnitt beschrieben, besteht die ES-Brake aus einem Plattenkondensator (bestehend aus zwei Elektroden und einem dazwischenliegenden Dielektrikum), an dem eine hohe Spannung (bis zu 1500 V) anliegt. Die daraus resultierende Kraft zwischen den beiden Kondensator-Elektroden – welche mittels 100 μm dicker Stahlfolie gebildet werden – zieht diese so stark zusammen, dass durch die entstehende Reibungskraft ein Auseinanderbewegen in Längsrichtung der Stahlfolie-Streifen nicht mehr möglich ist[1].

Als Dielektrikum kommt ein 13 μm dickes Polyamide-Band zum Einsatz, welches mit einer Dielektrizitätszahl von ϵ_r einen ähnlichen Wert wie das „Vorbild“-Dielektrikum besitzt, und außerdem eine genügend hohe Durchschlagsfestigkeit (>1500 V) aufweist[2]. Dieses wird, wie auch das des Vorbilds, mittels elektrisch leitendem doppelseitigen Klebeband fixiert, um den nicht-leitenden Abstand zwischen den Elektroden möglichst gering zu halten[3].

3.2.2 Implementierung

Die Implementierung des kinästhetischen Feedbacks erfolgt mittels der beschriebenen Komponenten (Stahlstreifen, Dielektrikum, elektrisch leitendes Klebeband). Eine der Elektroden – bezeichnet als „Hand-Strip“[4] – wird linear unbeweglich am Handrücken mittels Verschraubung fixiert. Die andere Elektrode – bezeichnet als „Finger-Strip“[5] – wird nur an der dafür vorgesehenen Montagebohrung an der Fingerspitze montiert und ist damit frei beweglich.

Die Spannungsversorgung der Strips (siehe 4.4) wird am Stecker an der Rückseite der Elektronik-Box abgegriffen und mittels dünnen Drähten zu den jeweiligen Elektroden geführt.

[1] vgl. [30], operating principle.
[2] vgl. [14], Technical Specifications.
[3] vgl. [30], Fabrication of the ES brake.
[4] vgl. [30], ELECTROSTATIC BRAKING MECHANISM.
[5] vgl. [30], ELECTROSTATIC BRAKING MECHANISM.

3.2.3 Schwierigkeiten

Abgesehen von den elektronischen Komponenten (siehe 4.4), besteht ein großes Schwierigkeits-Potenzial beim Zusammenbau der ES-Brakes. Denn, wie aus der Formel III.5 (siehe *Stand der Technik* Abschnitt 4.2.2) zu entnehmen ist, ist der Abstand zwischen den Elektroden ein wesentlicher Faktor für die resultierende Kraft.

Für eine deutlich wahrnehmbare Blockade der Fingerbewegung sollte die, aus der Anziehungskraft F_{anz} zwischen den beiden Elektroden resultierende, Reibungskraft F_R (Berechnung mit III.6, *Stand der Technik* Abschnitt 4.2.2) einen minimalen Wert von ca. 20 Newton aufweisen.

In nachfolgendem Diagramm wird deutlich veranschaulicht, wie stark sich der Elektroden-Abstand d bei gleichbleibender Spannung U und konstanter überlappender Fläche A bemerkbar macht. Für die in Abbildung VI.12 dargestellten Kräfte-Kurven wird von folgenden Werten ausgegangen:

- Spannung an den Elektroden $U = 1300V$
- Überlappende Fläche der Elektroden $A = 10cm^2$
- Elektrodenabstand exklusive Dielektrikum $d_{exkl.Dielektrikum} = 0\mu\text{m}$
- Elektrische Feldkonstante $\epsilon_0 = 0,885419 * 10^{-11} \frac{F}{m}$[6]
- Dicke des Dielektrikums $d_{Dielektrikum} = 12,7\mu\text{m}$
- relative Dielektrizitätszahl der Polyamide-Folie bei 23°C $\epsilon_r = 4$[7]
- Reibungskoeffizient der Folie $\mu_R = 0,5$

[6] vgl. [9], 3.5.1 Elektrisches Feld.
[7] vgl. [14], Technical Specifications.

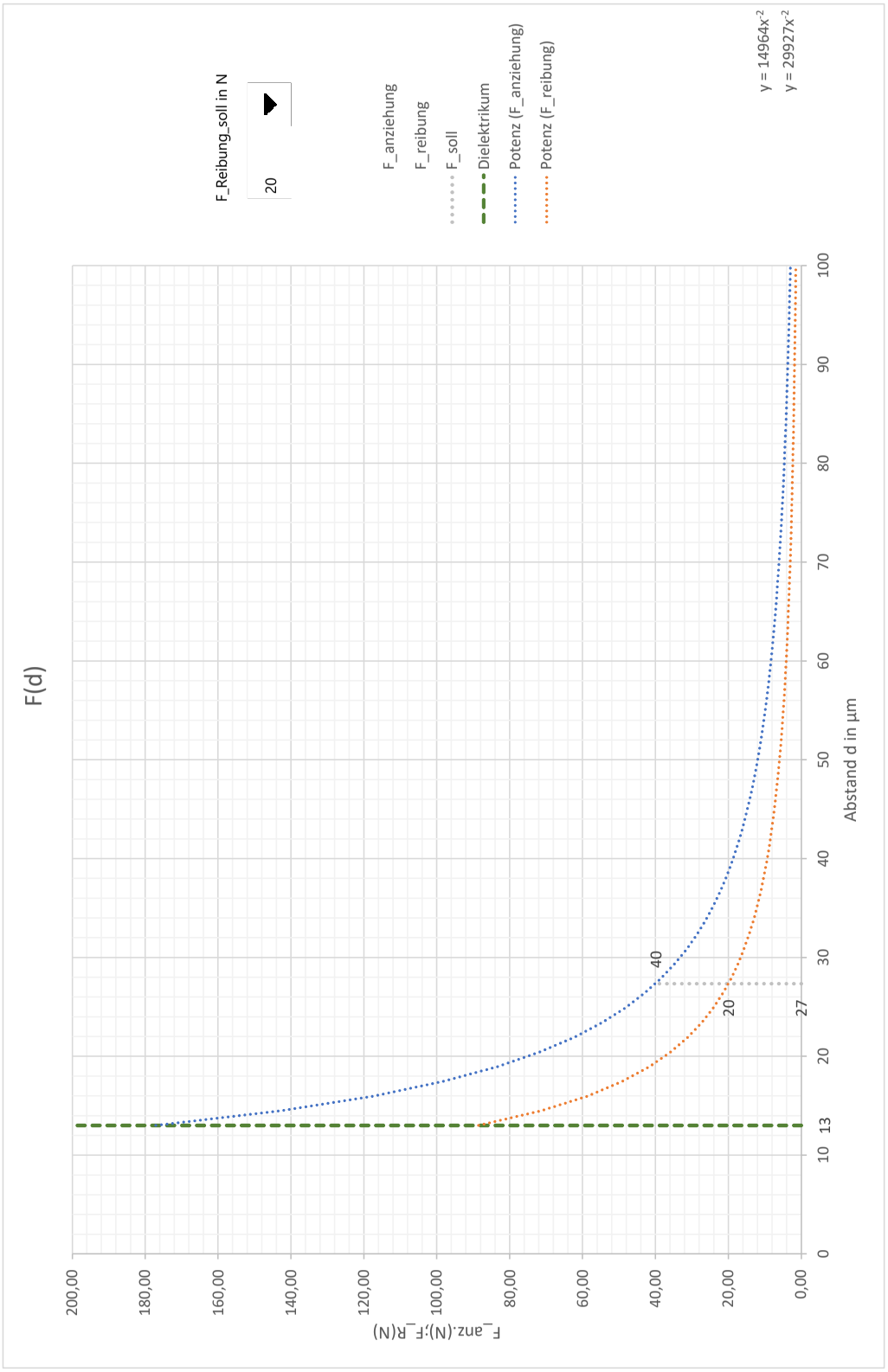


Abbildung VI.12: Einfluss des Elektrodenabstands auf die resultierenden Kräfte

Der gesamte Bereich rechts von der grünen Linie „Dielektrikum“ kann aufgrund der Dicke des Dielektrikums nicht erreicht werden, wodurch eine theoretische Maximalkraft F_{anzMax} von ca. 180 N entsteht. Diese führt unter der Annahme eines relativen Reibungskoeffizienten des Dielektrikums $\mu_{Dielektrikum} = 0{,}5$ zu einer Reibungskraft F_{RMax} von 90 N. Mit steigendem Elektrodenabstand nehmen die beiden Kräfte mit einer quadratischen Funktion ab. Diese steile Abnahme der Kräfte führt dazu, dass bereits bei einem Abstand von ca. 27 μm die geforderte Minimalkraft 20 N auftritt. Schon ab ca. 70 μm liegt F_R in einem Bereich, in welchem eine Anwendung des Feedbacks nicht mehr möglich ist.

Hinzu kommt, dass es sich bei besagten Werten um den totalen Elektrodenabstand – also inklusive der Dicke des Dielektrikums – handelt. So ergibt sich für den Toleranzbereich der Vorrichtung ein Wert von:

$$d_{Toleranz} = d_{Gesamt,akzeptabel} - d_{Dielektrikum} = 27\mu m - 13\mu m = 14\mu m \tag{VI.1}$$

Ein weiteres Problem besteht darin, die Streifen der Stahlfolie für Hand- und Finger-Strips so zuzuschneiden, dass keine scharfen Kanten entstehen, welche die dünne Polyamide-Folie aufschneiden und somit einen Kurzschluss zwischen den Elektroden verursachen würden.

Zusammenfassend ist festzuhalten, dass das kinästhetische Feedback (unter konstanter und genügend hoher Spannungsversorgung) nur dann angewandt werden kann, wenn eine Zusammensetzung der Vorrichtung unter diesen Voraussetzungen umgesetzt wird.

3.2.4 Anwendung

Zur Anwendung kommt die ES-Brake, also das kinästhetische Feedback, zusammen mit dem taktilen Feedback. Schließt der Anwender seine Hand, so wird bei der Position, an der die Sensorik des Greifers einen Kontakt meldet, die Blockade aktiv. Dadurch ist ein weiteres Schließen der Hand (jeder Finger separat) nicht mehr möglich.

Das bedeutet, dass die eigene Hand beim Greifvorgang eine Finger-Abwinkelung einnimmt, die der realen Abwinkelungen beim tatsächlichen physischen Umgreifen des Gegenstands sehr ähnelt.

4 Hardware-Design

4.1 Spannungsversorgung

Die Spannungsversorgung erfolgt über einen 3,7 V (Nominale Spannung) LiPo[8] Akku, welcher für eine zufriedenstellende Laufzeit eine Kapazität von 1200 mAh[9]aufweist. Über den 3,7 V Akku kann der Mikrocontroller direkt betrieben werden[10].

Dadurch, dass der verwendete Mikrocontroller eine integrierte Lade-Elektronik besitzt, die mit dem Akku kompatibel ist, kann auf eine eigens designte Elektronik verzichtet werden. Außerdem ermöglicht der enthaltene Festspannungsregler die Verwendung des Akkus, obwohl dieser eine Schwankung von $U_{entladen} = 3V$ und $U_{laden} = 4,2V$ aufweist[11].

Angeschlossen wird der Akku an den μC[12] über einen Wechselschalter, von dem allerdings nur ein Kontaktpaar benützt wird. Zwar ist es nicht unbedingt notwendig den Akku allpolig – also Plus- und Minuspol – zu schalten, allerdings bietet es sich an, da die Leitung des Akkus sowieso an mindestens einem Pol getrennt werden muss.

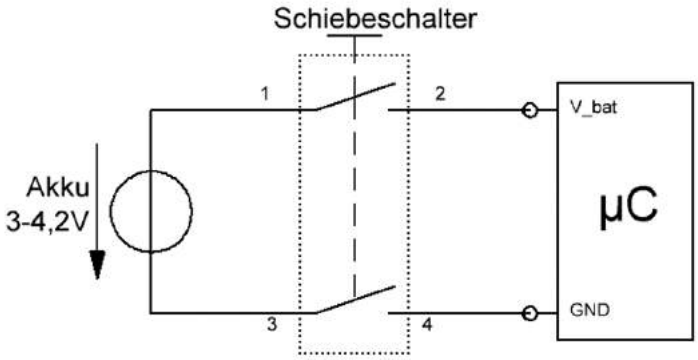


Abbildung VI.13: Anschluss des Akkus

4.2 Sensorik

Für die räumliche Bewegungsaufnahme der menschlichen Hand und für die Abwinkelung der Finger/Daumen musste eine Sensorik konzipiert werden, die diese hinreichend genau erfassen kann. Außerdem ist infolge des kinästhetischen Feedbacks (Blockade der Finger) ein Sensor nötig, der eine Aufhebung der Blockade hervorruft.

[8] Lithium Ionen Polymer
[9] vgl. [56], SPECIFICATION.
[10] vgl. [26], Battery + USB Power.
[11] vgl. [56], 2. SPECIFICATION.
[12] kurz für Mikrocontroller

4.2.1 Räumliche Bewegungsaufnahme

Die räumliche Bewegungsaufnahme, welche zur Interpretation der Hand-Bewegungen benötigt wird, erfolgt über einen 9-Achsen-Inertialsensor, welcher im Kapitel *Stand der Technik* Abschnitt 3.4 detailiert erklärt wird.

Versorgt wird der Sensor über die vom Mikrocontroller stammenden 3,3 V. SDA und SCL sind die Datenleitungen des I^2C-Busses. Genauere Verschaltungsdetails sind aus Abbildung VI.14 zu entnehmen.

Die Ansteuerung und Auswertung des Sensors wird im Kapitel *Software* Abschnitt 4.3 beschrieben.

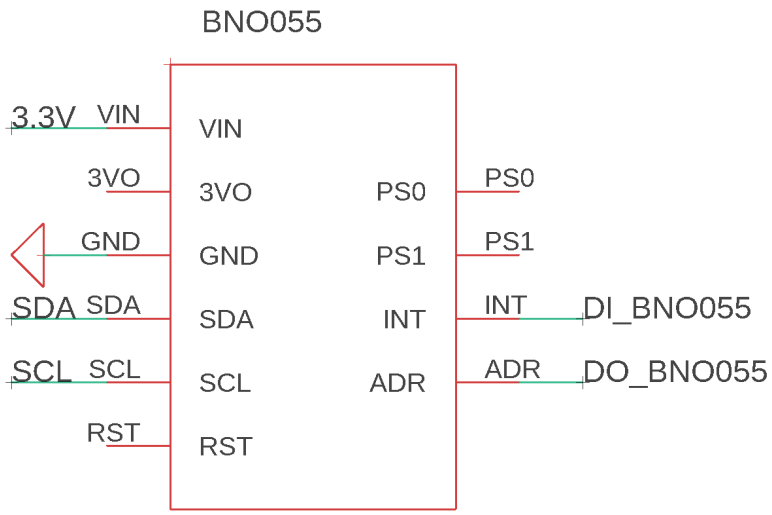


Abbildung VI.14: Verschaltung des Inertialsensors

4.2.2 Finger-Abwinkelung

Bei der Messung der Winkel der Fingerglieder kommt der im Kapitel *Stand der Technik* Abschnitt 3.3 beschriebene Biegesensor zum Einsatz. Durch eine geschickte Anbringungs-Position eines Sensors an jedem Finger, kann mittels einfacher Elektronik ein genauer Wert der jeweiligen Winkel ermittelt werden. Die Befestigung der Sensoren erfolgt durch Einführen in die dafür vorgesehenen Lücken am Handrücken (siehe Abbildung VI.4 (2)). Durch die enge Ausführung der Lücken und den dadurch entstehenden Halt, kann auf eine komplizierte Montage verzichtet werden.

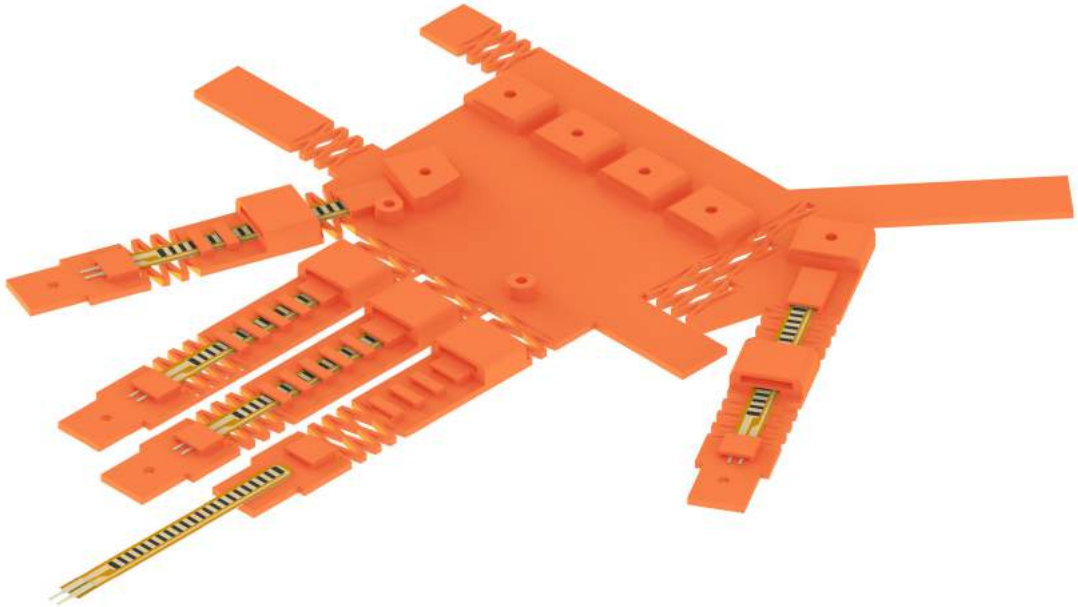


Abbildung VI.15: Biegesensoren auf dem Handrücken

Beim Abwinkeln eines Fingers biegt sich der Biegesensor ebenfalls proportional mit, wodurch sich sein Widerstandswert R ändert. Diese Widerstandsänderung kann über eine einfache Spannungsteiler-Schaltung mittels eines konstanten Referenzwiderstandes und dem daraus resultierenden Spannungsabfall ermittelt werden. So ergibt sich für jeden beliebigen Winkel des Fingers ein bestimmter Spannungswert, welcher dann von der Software interpretiert werden kann.

In nachfolgender Grafik steht bei der Leitung zum analogen Eingang „FLEX_X“ das X für einen bestimmten Finger.

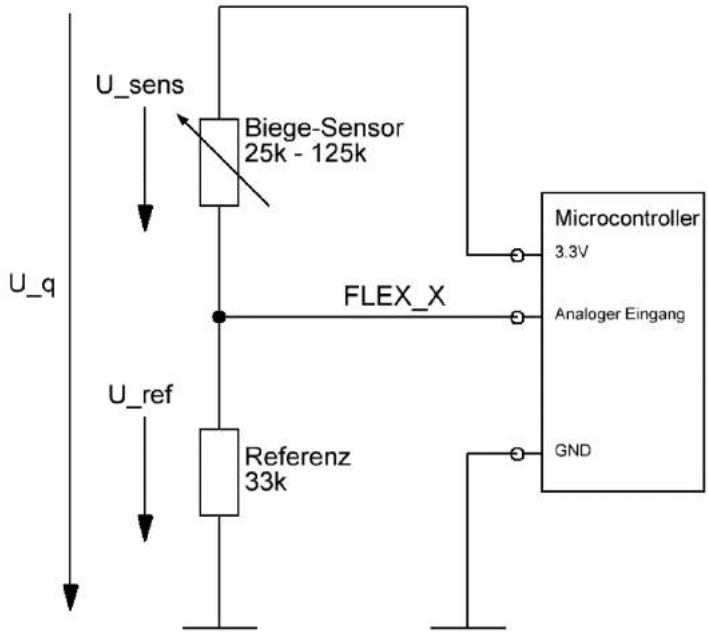


Abbildung VI.16: Auswerteschaltung der Biegesensoren

Durch den Spannungsteiler ergibt sich mit einer Versorgungsspannung U_q von 3,3 V (Mikrocontroller als Quelle) eine Referenzspannung.

Für den Spannungsteiler gilt:

$$\frac{U_{ref}}{U_q} = \frac{R_{ref}}{R_{sens} + R_{ref}} \rightarrow U_{ref} = U_q * \frac{R_{ref}}{R_{sens} + R_{ref}} \tag{VI.2}$$

Daraus folgt, dass mit zunehmendem Widerstandswert vom Sensor die zu verarbeitende Referenzspannung sinkt. Aus der Funktionsweise des Sensors ergibt sich, dass der Widerstandswert durch den größeren Biegungswinkel steigt.

Ist nun der Finger gerade ausgestreckt, ergibt sich am Referenzwiderstand die maximal auftretende Referenzspannung U_{refMax}, wohingegen bei maximal abgewinkeltem Finger die minimale Referenzspannung U_{refMin} anliegt.

Die weitere Auswertung der Sensorwerte ist im Kapitel *Software* Abschnitt 4.4 detailliert beschrieben.

Aufgrund dessen, dass der verwendete Mikrocontroller nur eine begrenzte Anzahl von analogen Inputs (AI) besitzt (maximal 6), und diese nicht für die Verwendung aller Sensoren ausreicht, muss mittels eines externen ADCs[13] die Kapazität aufgestockt werden.

Durch den zusätzlichen ADC ist es möglich, alle Biegesensoren sowie alle Drucksensoren (siehe 4.2.3) einzulesen. Prinzipiell kann frei gewählt werden, ob dieser bei den Biegesensoren oder bei den Drucksensoren angewandt wird, allerdings kann über die Auflösung eine Auswahl getroffen werden. Weil der Wert der Biegesensoren möglichst genau eingelesen und verarbeitet werden soll, ist es vernünftig hierfür den externen ADC einzusetzen, da

[13] Analog-Digital-Converter: wandelt analoge Eingangssignale in digitale Ausgangssignale um

dieser im Vergleich zum internen 10-Bit-Wandler eine 12-Bit Auflösung besitzt. Verwendet wurde dabei die Type „MCP23017-E/SO“.

Folgend wird die Schaltung aus Abbildung VI.16 um den ADC erweitert, wobei die Grundlegende Auswertung des Sensors mittels dem Spannungsteiler erhalten bleibt.

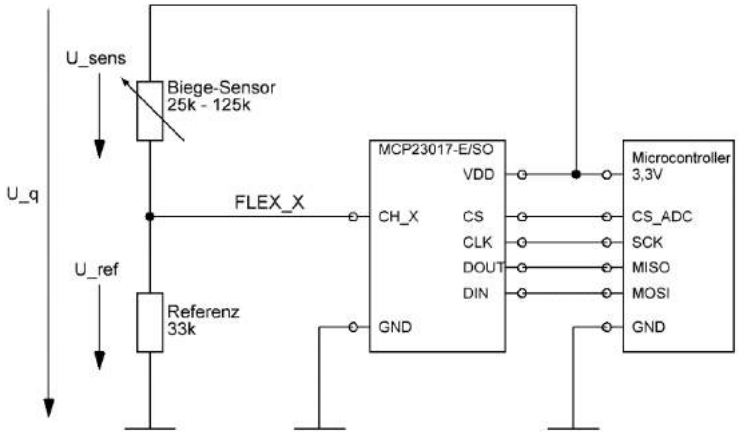


Abbildung VI.17: Auswerteschaltung der Biegesensoren mit externem ADC

Der MCP23017 wird über den 3,3 V Pin des Mikrocontrollers mit Spannung versorgt.

Die übrigen Pins/Leitungen (CS-CS_ADC, CLK-SCK, DOUT-MISO, DIN-MOSI) werden für die Anbindung an den SPI-Bus verwendet, welcher bereits im Kapitel *Stand der Technik* Abschnitt 5.2 erklärt wurde.

Wie auch bei den Drucksensoren (siehe 4.2.3) erfolgt eine Ausführung in fünffacher Form, also jeweils für einen Finger. In nachfolgender Abbildung ist die Verwendung des ADCs für alle fünf Finger anschaulich dargestellt. So nimmt zum Beispiel die Sensorleitung des Daumens Channel 0 in Anspruch.

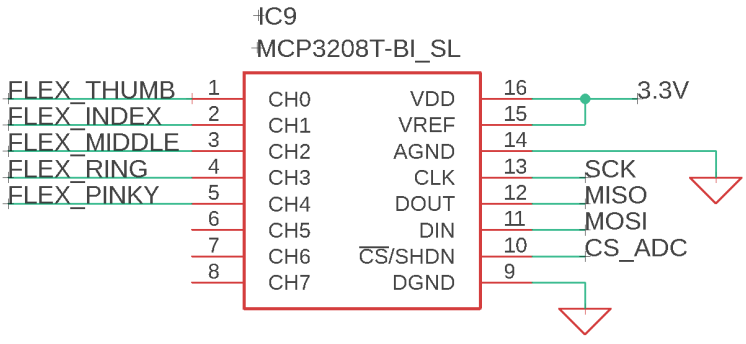


Abbildung VI.18: Verschaltung des ADCs

4.2.3 Drucksensor

Kommt der Anwender in die Situation, in dem das kinästhetische Feedback aktiv ist, er also seine Finger aufgrund der elektrostatischen Blockade nicht mehr bewegen kann, käme er allerdings gleichzeitig in die Lage, dass er seine Hand nicht mehr öffnen könnte. Denn die Blockade wirkt selbstverständlich in beide Richtungen, was allerdings nicht das Ziel des Feedbacks ist.

Um dieses Problem zu lösen, ist an der Fingerspitze ein resistiver Drucksensor angebracht, welcher im Abschnitt 3.2 beschrieben ist. Der Druck, welcher beim gewollten Öffnen der Hand aufgrund der Blockade (wirkt als mechanisches Lager) an der Oberseite der Fingerspitze entsteht, kann mittels Drucksensor aufgenommen werden.

Die Montage der Drucksensoren erfolgt mittels doppelseitigem Klebeband, welches bereits am Drucksensor angebracht ist. Wichtig dabei ist, dass der Sensor genau so positioniert ist, dass genügend Druck entstehen kann, um die Aufhebung der Blockade veranlassen zu können.

Abbildung VI.19: Montage des Drucksensors

Weil der Drucksensor ein sogenannter FSR – also force sensing resistor – ist, und somit als physikalische Ausgangsgröße einen ohmschen Widerstand darstellt, kann als Auswertungsschaltung dieselbe Spannungsteiler-Schaltung wie bei den Biegesensoren (siehe Abschnitt 4.2.2) verwendet werden.

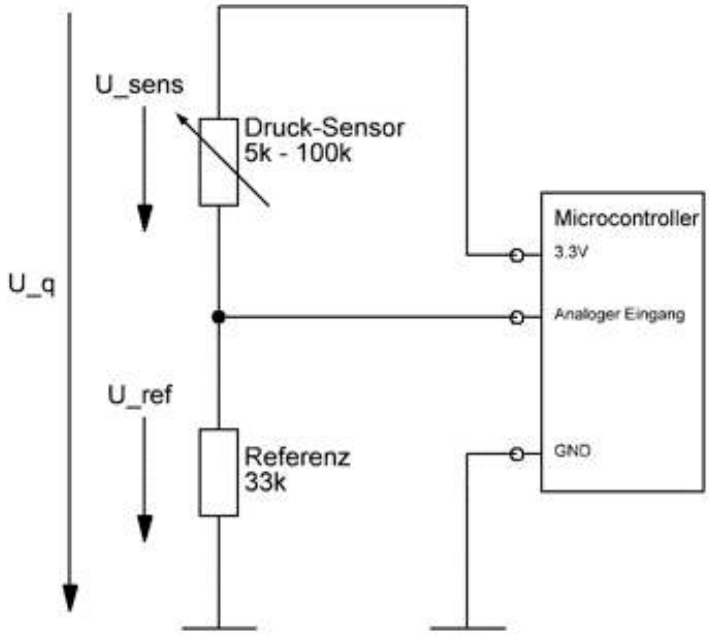


Abbildung VI.20: Auswerteschaltung der Druck-Sensoren

4.3 Steuerung des taktilen Feedbacks

Prinzipiell stellt zwar die Ansteuerung eines einzelnen DC-Motors (verwendet als Vibrations-Motor) keine Problematik dar, allerdings kann durch die gleichzeitige Verwendung mehrerer Motoren Probleme entstehen.

4.3.1 Steuerung

Grundlegend für die Auslegung der Ansteuerungsweise ist der benötigte Strom zum Betreiben der Motoren. Da die Strombelastbarkeit der Mikrocontroller-IOs nur einige wenige Milliampere beträgt, und daher nicht genug Strom zum Betreiben der Motoren geliefert werden kann, werden die Motoren über ein Darlington-Transistor-Array vom Typ ULN2803A angesteuert.

4.3.2 Schaltungsbeschreibung

Alle Motoren sind mit dem Pluspol ununterbrochen an die 3,3 V Versorgung angeschlossen und über einen 20 Ω Strombegrenzungswiderstand mit dem Darlington-Transistor verbunden.

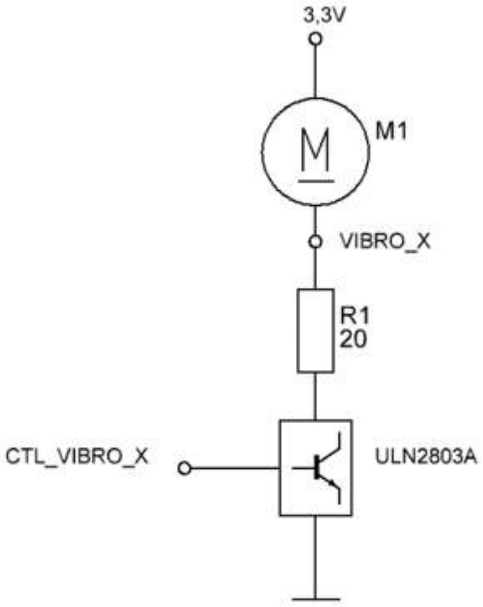


Abbildung VI.21: Ansteuerung des Vibrationsmotors

Das Array wird direkt über den Mikrocontroller (3,3 V) versorgt. Die Ansteuerung eines Motors erfolgt über die Steuerleitung CTL_VIBRO_X, welche jeweils mit einem digitalen Ausgang des Mikrocontrollers verbunden ist. „X“ steht dabei für die verschiedenen Finger. Ein High-Pegel auf der Steuerleitung legt den Minuspol des Motors über den Strombegrenzungswiderstand auf Masse.

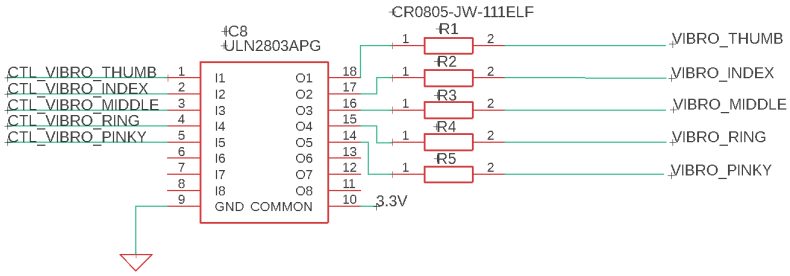


Abbildung VI.22: Verschaltung des Darlington-Transistor-Arrays

Die Ansteuerung der DC-Motoren sowie der Aktoren des kinästhetischen Feedbacks (in Abbildung VI.23 die Leitungen CTL_STRIP_X, siehe 4.4) erfolgt allerdings nicht direkt über den Mikrocontroller, da dieser nicht ausreichend viele GPIOs[14] besitzt. Stattdessen wird eine zusätzliche IO-Erweiterung angewendet, welche die fehlenden GPIOs kompensiert. Verwendet wird dabei die Type „MCP32017“, welche über den I^2C-Bus angesteuert wird.

[14]General Purpose Input/Output

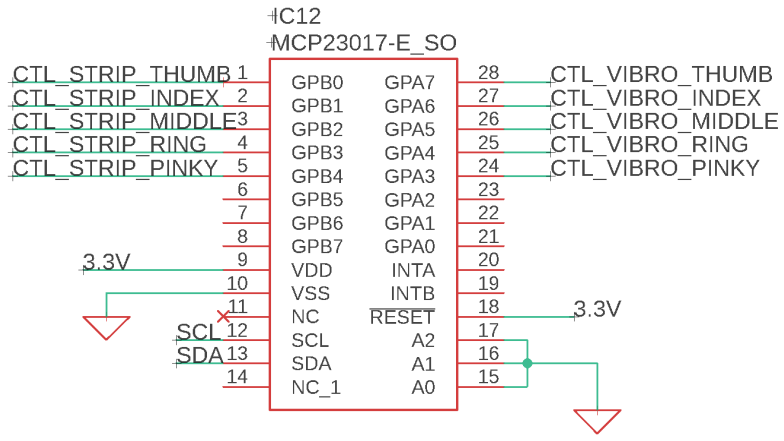


Abbildung VI.23: Verschaltung der IO-Erweiterung

4.4 Steuerung des kinästhetischen Feedbacks

4.4.1 Steuerung

Die grundlegende Voraussetzung für die Umsetzung einer Steuerung des kinästhetischen Feedbacks ist die Art der Spannungsform (AC[15] oder DC[16]) sowie deren Effektivwert.

Nach ausführlicher Recherche, unter primären Berücksichtigung der Schaltspannung und Baugröße, wurde entschieden, dass aufgrund der leichteren Handhabung und die Möglichkeit der Verwendung von MOS-FETs , eine DC-Spannungsversorgung verwendet wird. (Anders als die Verwendete Spannungsform von DextrES, siehe 4.2.2)

Als Hochspannungsquelle wird ein DC-DC-Wandler verwendet, welcher von einer 5 V Spannung auf 1500 V wandelt.

4.4.2 Schaltungsbeschreibung

Zunächst musste die Beschaltung für einen Schaltspannungsregler („TPS613222ADBVR“) konzipiert werden, welcher die Ausgangsspannung des Akkus V_{BAT} auf besagte 5 V wandelt. Die dafür nötigen Bauteilwerte waren dem Datenblatt des Schaltspannungsreglers zu entnehmen[17].

[15] Alternating Current, Wechselstrom
[16] Direct Current, Gleichstrom
[17] vgl. [61], Abschnitt 9.2.2.

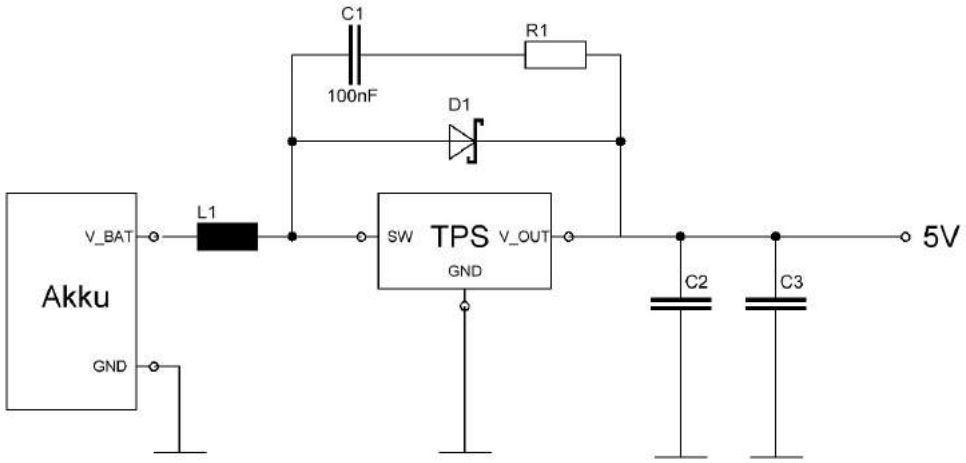


Abbildung VI.24: Schaltsspannungsregler V_{BAT} auf 5 V

Aufgrund dessen, dass der verwendete Hochspannungswandler „A20P-5“ vom Hersteller „XP-Power“ ungeregelt ist – also in unbelastetem Zustand eine viele höhere Spannung liefert (siehe Abbildung VI.25), welche zur Zerstörung der Elektronik führen würde – muss eine Spannungsregelung vorgenommen werden. Diese erfolgt durch einen einfachen Operationsverstärker (OPV), welcher als nicht-invertierender Verstärker (siehe Abbildung VI.26) betrieben wird.

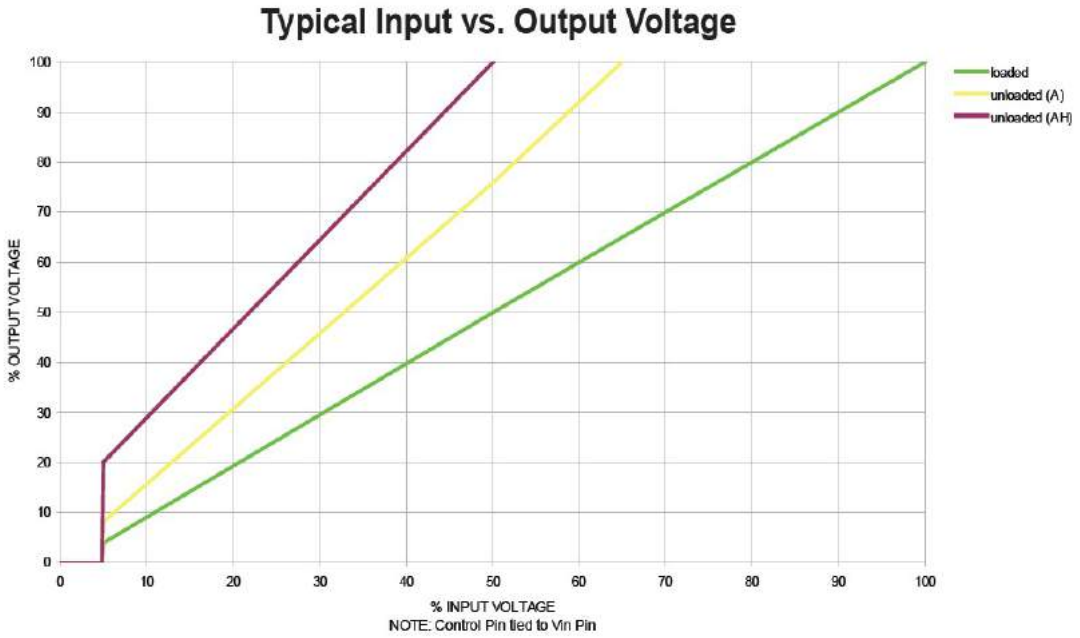


Abbildung VI.25: HV Output unter verschiedenen Belastungen [63]

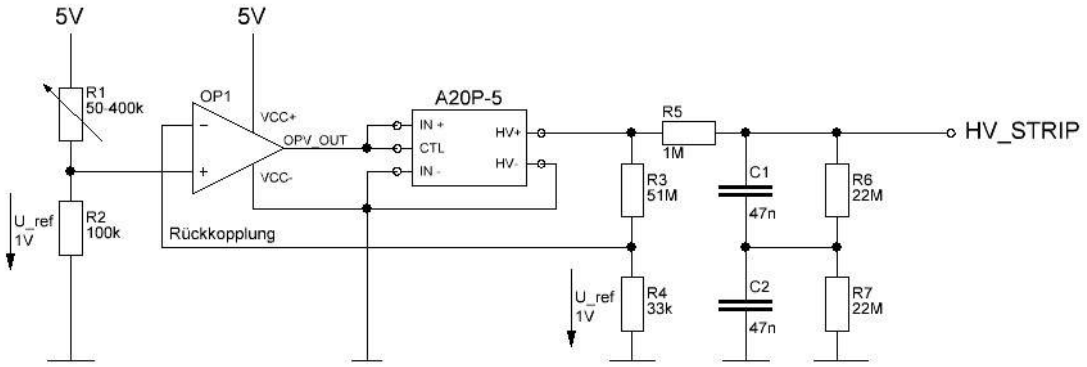


Abbildung VI.26: Hochspannungsregelung und Strombegrenzung

Die Widerstände R1 & R2 bilden zusammen mit R3 & R4 die Spannungsregelung. Das Widerstandsverhältnis $\frac{R_1}{R_2}$ wird durch den Trimmer R_1 so eingestellt, dass es den Wert 4 annimmt, wodurch an R_2 stets 1 V anliegt. Durch das Widerstandsverhältnis $\frac{R_3}{R_4} \approx 1500$ liegt bei der Soll-Ausgangsspannung des Spannungswandlers an R_4 ebenfalls 1 V an.

Tritt allerdings der Fall auf, dass der Spannungswandler unbelastet ist, und dieser dann eine sehr viel höhere Spannung als 1500 V abgibt, erhöht sich ebenfalls die Spannung an R_4, welche über die Rückkopplung an den invertierten Eingang des OPVs rückgeführt wird. Dadurch wird die Ausgangsspannung des OPVs – somit auch die Eingangsspannung IN+ des Hochspannungswandlers – und folglich die Ausgangsspannung HV+ reduziert, sodass wieder $U_{R_4} = U_{R_1}$ gilt.

Der Widerstand R_5 dient als Strombegrenzungswiderstand und nimmt aufgrund des maximal verfügbaren Stroms des Spannungswandlers $I_{Max} = 660\ \mu\text{A}$[18] einen sehr hohen Wert von $1M\Omega$ an.

Die Serienkondensatoren C_1 und C_2 dienen zusammen zur Spannungsstabilisierung der ES-Brake. Um diese wieder entladen zu können, sind jeweils parallel zu einem Kondensator die Widerstände $R_{6,7}$ geschaltet.

In Abbildung VI.27 kann die Antwort einer Belastungsänderung in den Kurven $V_{HV}(t)$ und $V_{OPVOUT}(t)$ betrachtet werden. Zum Zeitpunkt $t = 0.2$ wird das Feedback aktiviert, also die Elektroden an HV geschaltet. Durch die entstehende Belastungsänderung – in dem Fall wird die Belastung erhöht – bricht die Ausgangsspannung HV+ um einige Volt ein, woraufhin der OPV die Spannung sprunghaft nachregelt und sich danach wieder auf einen konstanten Wert einstellt. Bei $t \approx 0,4s$ wird der Wandler wieder entlastet. Da allerdings der OPV in der Simulation bereits in sehr kurzer Zeit regelt, steigt der Wert $V_{HV}(t \approx 0,4s)$ nur minimal an.

So bleibt die nötige Hochspannung am Ausgang des Hochspannungswandlers nahezu konstant bei ca. 1565 V, unabhängig von der Belastung.

[18] vgl. [63], PRODUCT SELECTION TABLE.

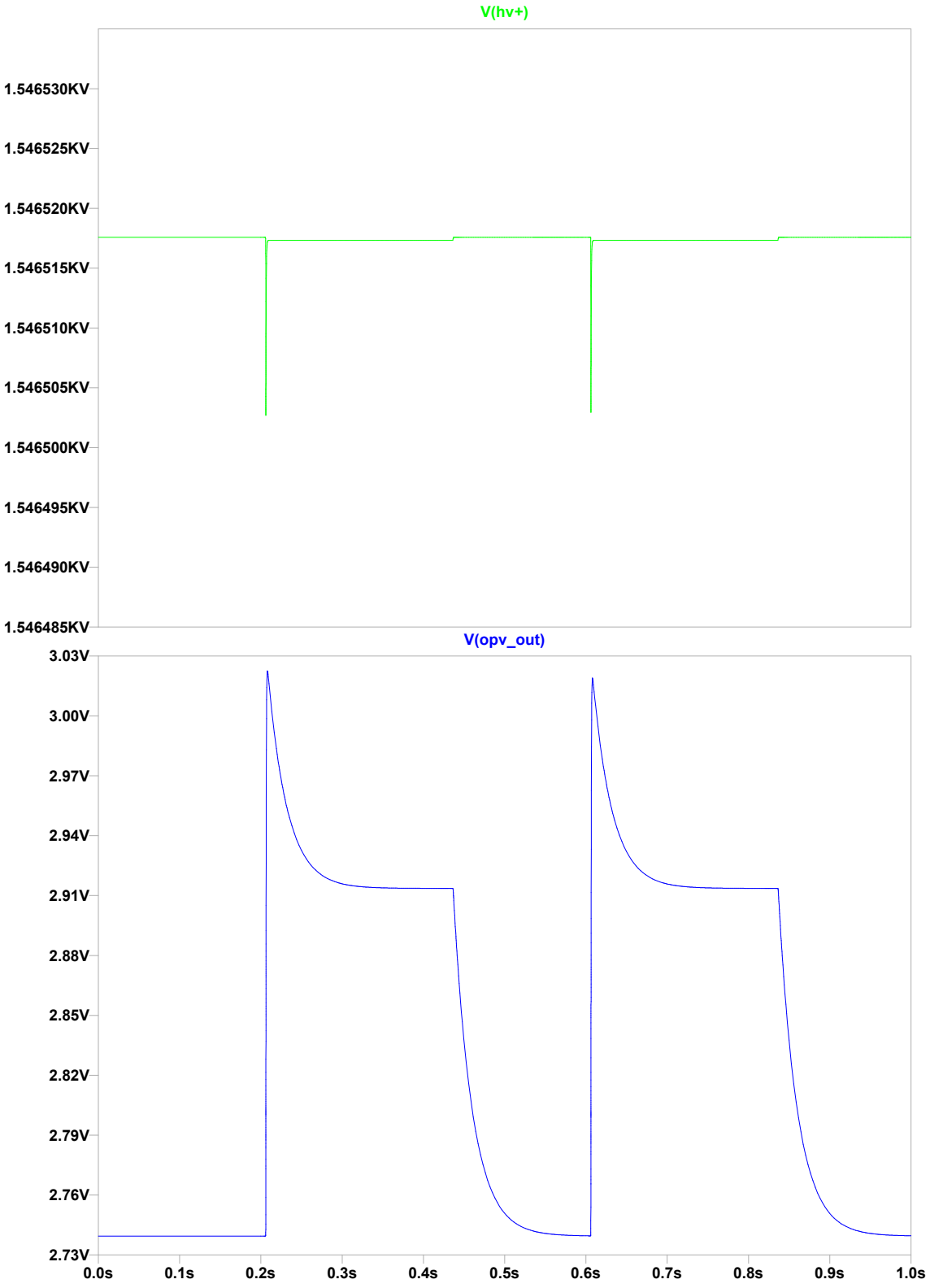


Abbildung VI.27: Simulation der Hochspannungs-Regelung

Um die Hochspannung gefahrlos mit dem μC schalten zu können, erfolgt eine Potenzialtrennung zwischen μC (LV-Seite) und HV-Seite mittels Optokopplern, wie in Abbildung VI.28 dargestellt wird.

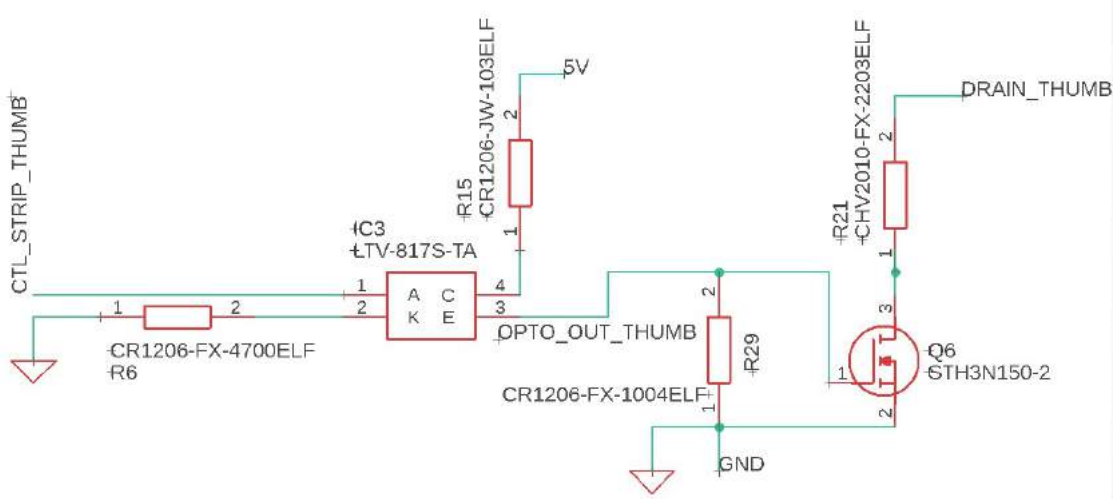


Abbildung VI.28: Ansteuerung des kinästhetischen Feedbacks

Die LV-Seite des Optokopplers wird über einen Vorwiderstand (hier R6) über den μC angesteuert. Bei einem genügend hohen Spannungspegel der Steuerleitung CTL_STRIP_X (3,3 V) schaltet dieser an der HV-Seite durch. Folgend liegen 5 V am Gate des MOS_FETs an, wodurch dieser leitend wird, und „DRAIN_X“ auf GND schaltet. Diese „DRAIN_X“ Leiterbahnen werden – wie auch die „HVSTRIP“-Leitungen – zum Stecker der Hochspannungsleitungen geführt, an dem die Elektroden der ES-Brake angeschlossen werden. Die Widerstände R_{30-39} dienen als Entladewiderstände der ES-Brakes.

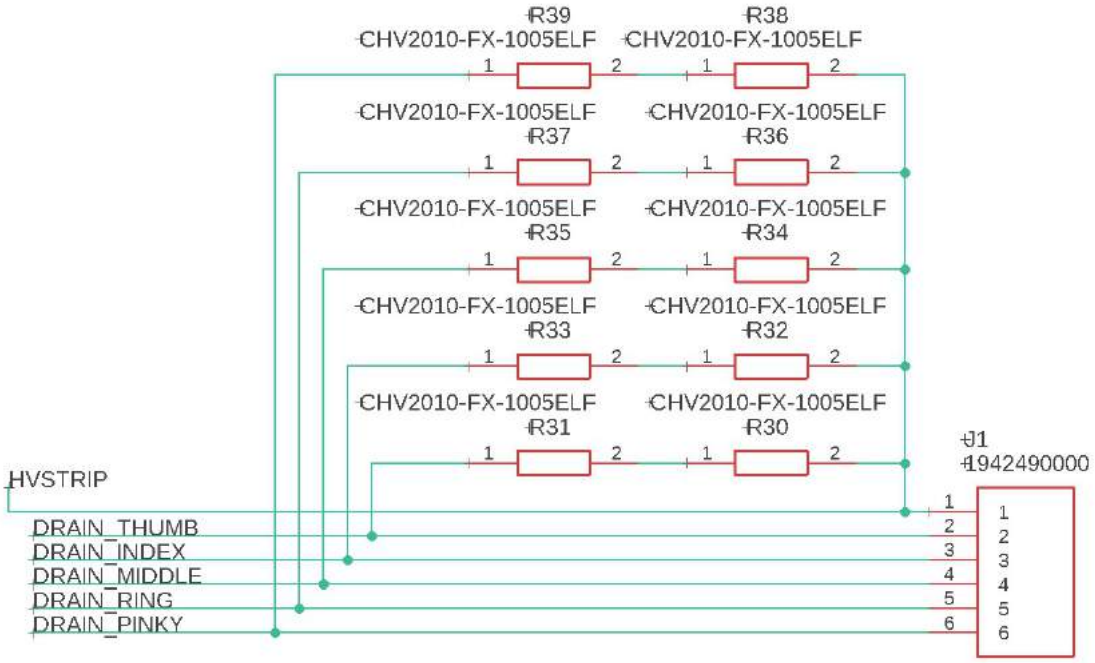


Abbildung VI.29: Führung der Stripleitungen zum Stecker

4.5 Layout der Platinen

Zwar wäre es deutlich platzsparender, die gesamte Elektronik auf einer einzigen Platine unterzubringen, allerdings ist dies aufgrund der Steuerung des kinästhetischen Feedbacks nicht möglich. Denn wie im Kapitel 4.4 beschrieben, benötigt man zur Steuerung des Feedbacks zahlreiche Hochspannungskomponenten. Diese erfordern einen größeren Abstand zwischen Bauteilen bzw. Leiterbahnen.

Deshalb wurde beschlossen, die Elektronik in einen ungefährlichen LV[19]-Teil und einen HV[20]-Teil zu trennen und diese auf separaten Platinen umzusetzen.

4.5.1 Low-Voltage Platine

Der LV-Teil der Platine beinhaltet:

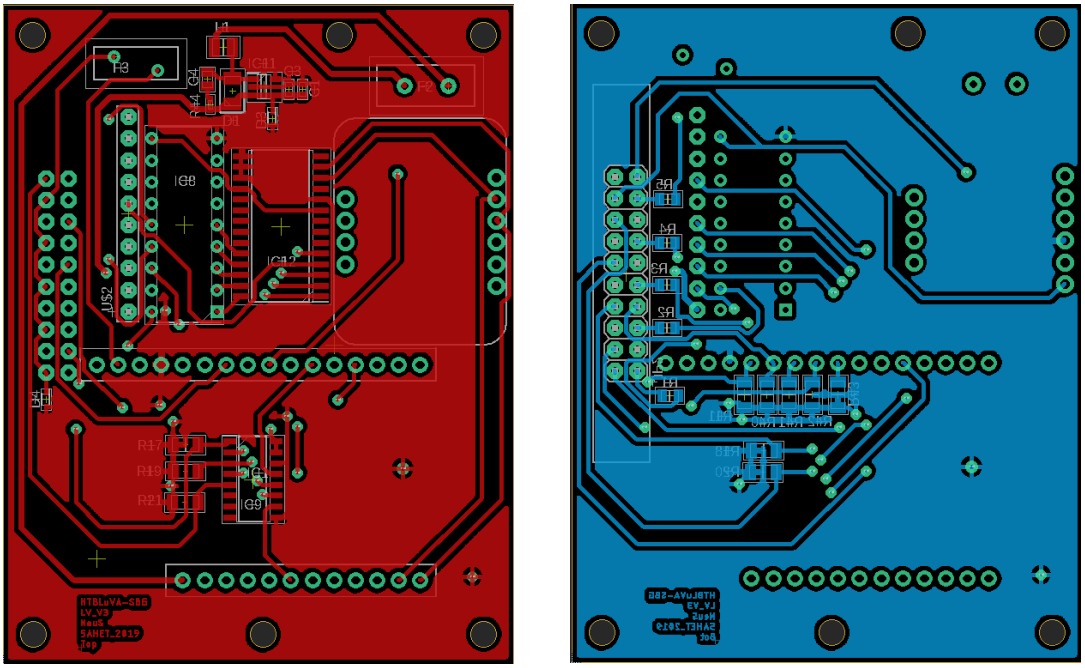
- Mikrocontroller, externer ADC, IO-Erweiterung
- Auswerteschaltungen der Sensoren
- Steuerung des taktilen Feedbacks
- Eingangsspannungsregelung auf 5 V
- Anschlussbuchse für Leitungen der:
  - Drucksensoren
  - Biegesensoren
  - Vibrationsmotoren

In den folgenden Grafiken ist in Abbildung VI.30a die obere Ansicht (TOP, Rot) des Platinenlayouts zu sehen, wobei in Abbildung VI.30b die Unterseite (BOT, Blau) dargestellt ist.

Die sichtbaren grünen Kreise deuten an, dass es sich dabei um eine sogenannte „Durchkontaktierung“ handelt, also eine galvanische Verbindung zwischen TOP und BOT. Diese werden zum einen bei Nicht-SMD[21] Bauteilen – also bei Bauteilen mit Durchsteckmontage – oder auch wenn es das Routing[22]verlangt, verwendet.

Die großen eingefärbten Flächen besitzen GND-Potenzial, was durchaus üblich bei derartigen Platinen ist.

[19] Low Voltage, Kleinspannung
[20] High Voltage, Hochspannung
[21] surface mounted device, Bauteil das direkt an der Oberfläche angebracht wird
[22] Art der Verlegung der Leiterbahnen (Abstand, Pfad, Länge, etc)



(a) LV-TOP (b) LV-Bot

Abbildung VI.30: Layout der LV-Platine

4.5.2 High-Voltage Platine

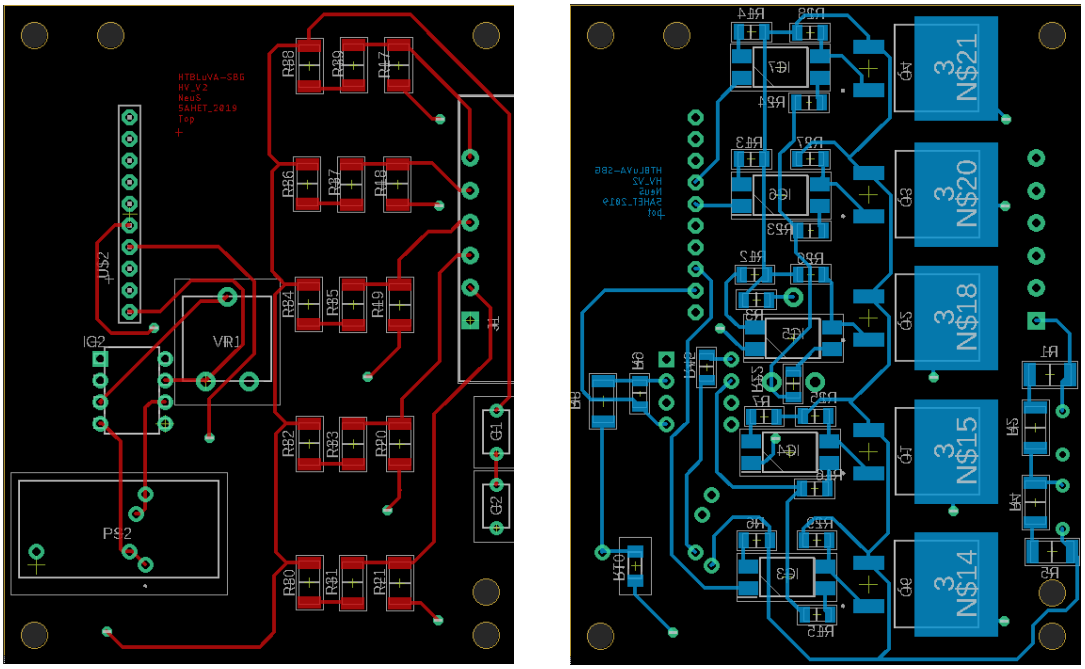
Der HV-Teil der Platine beinhaltet:

- Hochspannungs-Wandlung
- Regelung der Hochspannung
- Steuerung des kinästhetischen Feedbacks
- Anschlussbuchse für Leitungen des kinästhetischen Feedbacks

In den Abbildungen VI.31a sowie VI.31b ist im Vergleich zu denen in Abbildung VI.30 deutlich erkennbar, dass wesentliche Unterschiede in der Art des Routings bestehen. So sind zum einen die verwendeten Bauteile sehr viel weiter voneinander entfernt und zum anderen die Leiterbahnen mit einem weitaus größeren Abstand zueinander verlegt. Auf die im Abschnitt 4.5.1 beschriebenen GND-Flächen wurde hierbei sogar gänzlich verzichtet.

All diese Maßnahmen sind notwendig, da durch die unüblich hohe Spannung (bis zu 1500 V) die Gefahr des Spannungs-Überschlags[23] besteht.

[23] Lichtbogen zwischen zwei elektrisch-nichtverbundenen Leitern



(a) HV-TOP (b) HV-Bot

Abbildung VI.31: Layout der HV-Platine

Außerdem wurde bei sämtlichen hochspannungsführenden Bauteilen der HV-Platine auf die Betriebsspannungen geachtet. Der charakteristische Kennwert der Bauteile im Bezug auf die maximale Spannung ist aus dem Datenblatt des Herstellers zu entnehmen, wobei generell eine größere Bauform der Komponente meist eine höhere Spannung verspricht.

So kann zum Beispiel aus diesem Auszug eines exemplarischen Datenblatts des Widerstandes vom Typ „CHV“ vom Hersteller „Bourns“ entnommen werden, dass für eine Spannung von 1500 V mindestens die Bauform „CHV2010“ gewählt werden sollte.

Electrical Characteristics

Specification		Model				
		CHV0603	CHV0805	CHV1206	CHV2010	CHV2512
Power Rating @ 70 °C		0.1 W	0.125 W	0.25 W	0.5 W	1.0 W
Operating Temperature Range		-55 °C to +155 °C				
Maximum Operating Voltage		200 V	400 V	800 V	2000 V	3000 V
Maximum Working Voltage		400 V	800 V	1600 V	3000 V	4000 V
Resistance Range	1 % E-96 + E-24	100 kΩ – 10 MΩ				
	5 % E-24	100 kΩ – 22 MΩ		100 kΩ – 100 MΩ		
Temperature Coefficient	1 %	±100 PPM/°C				
	5 %	±200 PPM/°C				

Abbildung VI.32: Exemplarisches Datenblatt für einen Hochspannungswiderstand [13]

4.5.3 Huckepack-Platine

Um das Gesamtvolumen der beiden Platinen und somit der gesamten Elektronik-Box möglichst gering zu halten, werden die zwei Platinen mit einem möglichst geringen Abstand übereinander gestapelt.

Weil das verwendete Display bereits an der Rückseite eine Steckvorrichtung für die elektrische sowie mechanische Verbindung mit dem Mikrocontroller besitzt, wird diese auch benutzt. Dabei wird der Controller einfach mit den bereits vorhandenen Male-Pin-Header in die dafür vorgesehenen Female-Header hineingesteckt. Parallel zu jedem einzelnen Steckplatz befindet sich ein weiterer Female-Header, der mit dem anderen elektrisch verbunden ist. Durch diesen kann mittels eines zusätzlichen Male-Headers „μC-Header“ jeder Mikrocontroller-Pin zu den Platinen geführt werden. Die dafür notwendigen Durchkontaktierungen an der LV-Platine sind in Abbildung VI.34 zu sehen.

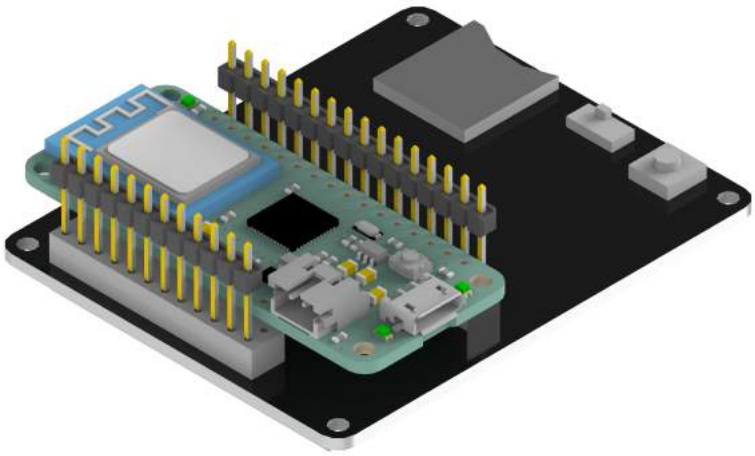


Abbildung VI.33: Anbringung des Mikrocontrollers an der Rückseite des Displays

Wie schon im Kapitel 2.3.2 sowie in den Abbildungen VI.5a und VI.5b beschrieben, sind die zwei Platinen an einem Teil der Elektronik-Box befestigt. Mittels eines weiteren Male-Headers an der HV-Platine, sowie eines Female-Headers an der LV-Platine, können die zwei Platinen elektrisch Verbunden werden (bzw. die notwendigen Versorgungsspannungen, Steuerleitungen, etc. hingeführt werden).

Wichtig dabei ist, dass die zwei eben angesprochenen Header direkt übereinander liegen, da sonst ein Zusammenstecken samt Elektronik-Box – also dem Gehäuse – nicht möglich ist. Dies ist vorallem beim Design des Platinenlayouts zu beachten.

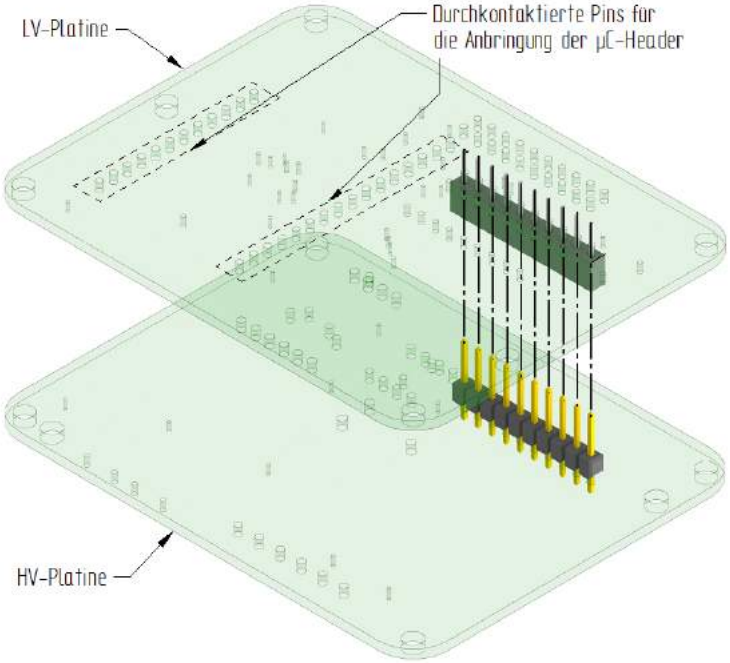
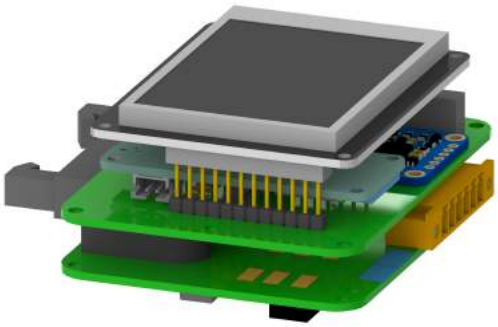
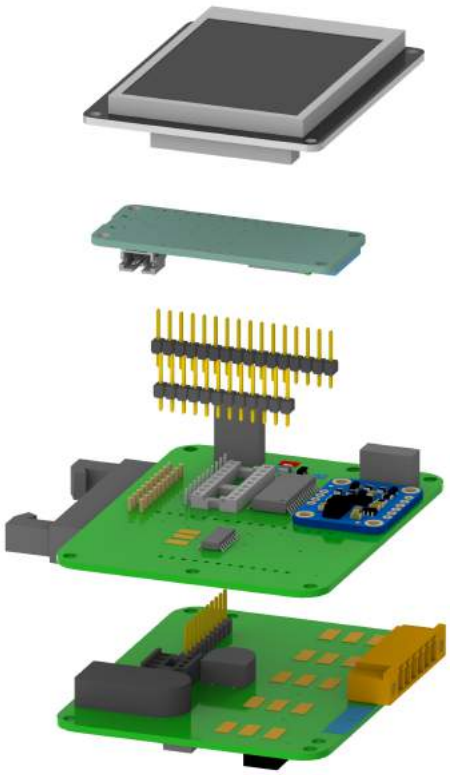


Abbildung VI.34: Pin-Header für Huckepack-Platinen

In den nachfolgenden Grafiken wird nochmals die Kompaktheit der gesamten Elektronik-Komponenten (zwei Platinen, μC, Display, etc), sowie eine detaillierte Explosionsansicht der Hardware dargestellt.



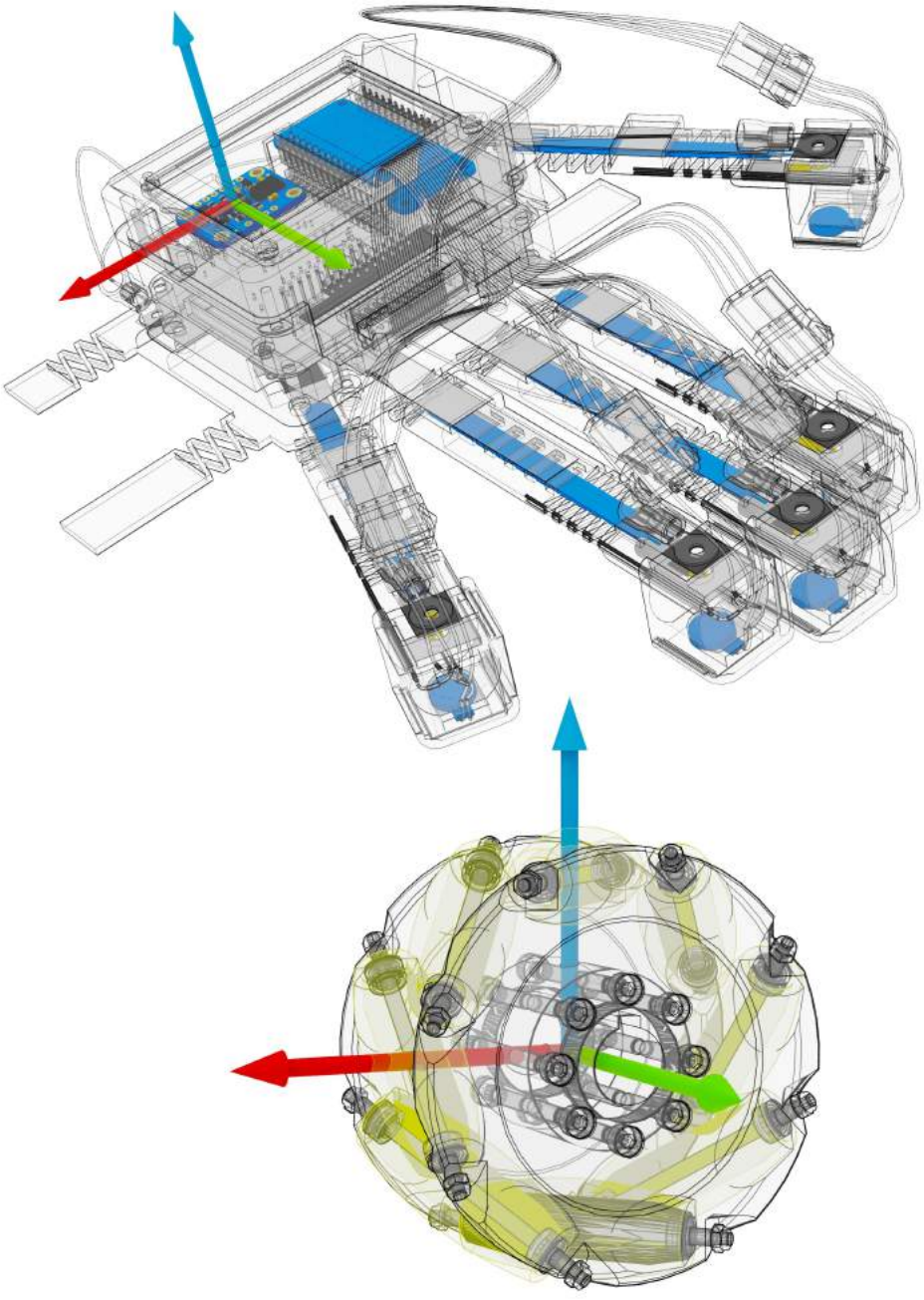
(a) Zusammengesteckte Huckepack-Platinen



(b) Explosionsdarstellung der Huckepack-Platinen

Abbildung VI.35: Huckepack-Platinen

VII Software



1 Übersicht

1.1 Topologie

Das Projekt beinhaltet drei Systeme, für die die Software entwickelt werden musste:

- das Embedded System im Fernsteuergerät (im Programmcode auch „Control“ genannt)
- das Embedded System am Roboter (im Programmcode auch „Receiver“ genannt)
- die speicherprogrammierbare Steuerung (SPS) am Roboter

Diese Systeme übernehmen verschiedene Aufgaben und kommunizieren über ein gemeinsames Übertragungsprotokoll. Zwischen den Embedded Systems im Fernsteuergerät und am Roboter besteht eine Verbindung über Bluetooth Low Energy, das Embedded System und die SPS am Roboter sind über eine RS232-Schnittstelle verbunden.

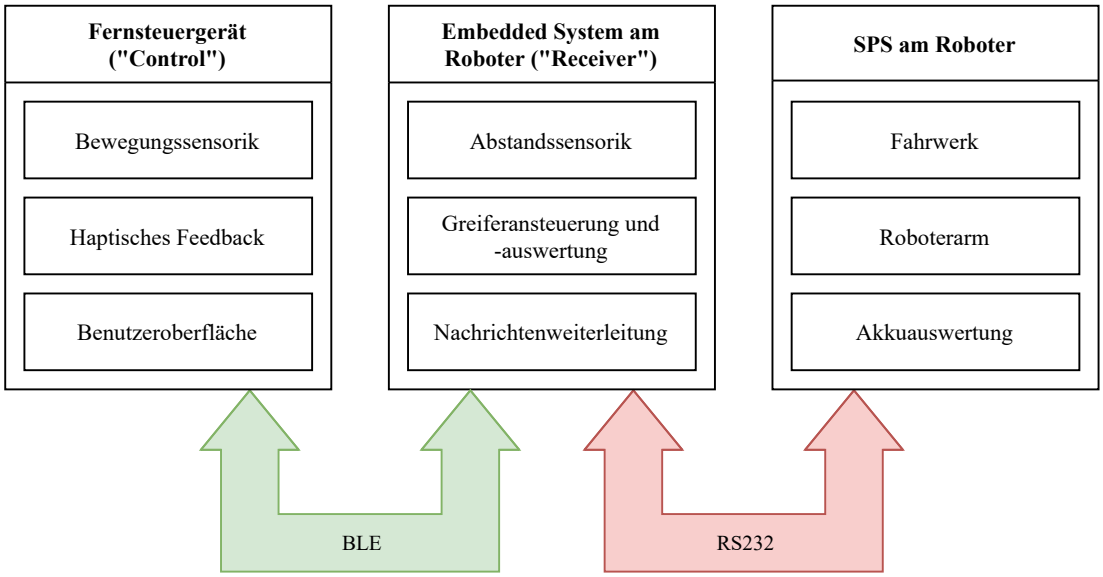


Abbildung VII.1: Topologie der Software

1.2 Aufgaben der Software im Fernsteuergerät

Die Software des Fernsteuergerätes übernimmt die Sensorauswertung, die Steuerung des haptischen Feedbacks und die Kommunikation mit dem Empfänger am Roboter. Das System wertet den 9-Achsen-Sensor am Handrücken, die Biegesensoren in den Fingern und die Drucksensoren an den Fingerspitzen aus. Diese Werte werden gefiltert, verarbeitet und als Bewegungsbefehle an das Robotersystem gesendet. Über die Kommunikationsschnittstelle bekommt das Fernsteuergerät Informationen über das Robotersystem und dessen Umgebung. Diese Daten werden einerseits verwendet, um das haptische Feedback anzusteuern, anderseits um Informationen wie den Akkustand am Display anzuzeigen.

1.3 Aufgaben des Embedded Systems am Roboter

Das Embedded System am Roboter hat drei Zuständigkeitsbereiche. Zum einen dient es als Empfänger für das Fernsteuergerät und leitet Nachrichten, die über Bluetooth Low Energy empfangen wurden, an die SPS über die RS232-Schnittstelle weiter. Die zweite

Aufgabe besteht in der Ansteuerung des Greifers, sowie Auswertung der Motorströme zur Erkennung von Objekten im Greifer. Außerdem werden die rund um den Roboter platzierten Ultraschallsensoren ausgewertet, um eine Kollisionsvermeidung zu realisieren.

1.4 Aufgaben der speicherprogrammierbaren Steuerung

Die SPS übernimmt die Steuerung des Roboterarms sowie der vier Fahrwerk-Motoren. Dazu müssen aus den vom Embedded System empfangenen Befehlen die Bewegungen der Motoren berechnet werden.

Die folgenden Abschnitte gehen nun weiter ins Detail der Implementierung der einzelnen Systeme. Abschnitt 2 geht kurz auf die benötigte Software ein. In Abschnitt 3 wird das gemeinsame Übertragungsprotokoll beschrieben. Darauf folgen in Abschnitt 4, 5 und 6 jeweils die Beschreibung der drei Softwaresysteme.

2 Toolchain

Im folgenden Abschnitt wird eine kurze Übersicht über die Struktur des Programmcodes und die für das Kompilieren benötigten Programme (= Toolchain) gegeben, um eventuellen Folgeprojekten die Fortsetzung zu erleichtern. Außerdem werden einige wichtige Einstellungen für das erfolgreiche Kompilieren des Programmcodes erläutert.

2.1 Ordnerstruktur

- `software`
  - `common` ..... Gemeinsam verwendete Bibliotheken
    - `pattern` ..... Bibliothek für das Abspielen von Mustern
    - `protocol` ..... Bibliothek für das gemeinsame Übertragungsprotokoll
    - `system` ..... Bibliothek für die Embedded Systems im Fernsteuergerät und am Roboter
  - `control` ..... PlatformIO-Projekt Fernsteuergerät
    - `include` ..... Header-Dateien
    - `sdcard` ..... Dateien die auf die SD-Karte am Display geladen werden müssen
    - `src` ..... Quellcode-Dateien
  - `receiver` ..... PlatformIO-Projekt Receiver (ESP32)
    - `include` ..... Header-Dateien
    - `src` ..... Quellcode-Dateien
  - `ultrasonic` ..... PlatformIO-Projekt Ultraschallsensorauswertung (ATmega328P)
    - `include` ..... Header-Dateien
    - `src` ..... Quellcode-Dateien
  - `roboterarm` ..... Automation Studio Projekt SPS am Roboter

2.2 PlatformIO

PlatformIO[1] ist ein sogenannter „Build-Manager“ für die Softwareentwicklung von Embedded Systems. Dieser erleichtert den Entwicklungsprozess, indem automatisch die erforderlichen Programme installiert werden. Außerdem können benötigte Bibliotheken angegeben werden, die dann automatisch heruntergeladen und beim Kompilieren inkludiert werden.

Ein PlatformIO-Projekt hat die folgende Ordner-Struktur:

```
/
├── include ..... Header-Dateien
├── lib ..... Interne Bibliotheken
├── src ..... Quellcode-Dateien
├── test ..... Automatisierte Tests
└── platformio.ini ..... Konfigurationsdatei des Projekts
```

In der `platformio.ini`-Datei wird definiert, für welche Systeme kompiliert werden soll. Außerdem können Einstellungen wie der „Upload-Port“ und die Baud-Rate der seriellen Schnittstelle eingestellt werden.

Von diesem Ordner aus kann dann das Kommando `platformio run --target upload` ausgeführt werden, womit das gesamte Projekt kompiliert und auf den Mikrocontroller geladen wird.

2.3 Visual Studio Code

Die Software für die Mikrocontroller wurde in Visual Studio Code, einem Code-Editor von Microsoft, entwickelt. Dieser ist nicht auf eine bestimmte Programmiersprache festgelegt, sondern kann durch Plugins an die Erfordernisse angepasst werden. PlatformIO stellt ein offizielles Plugin für Visual Studio Code zur Verfügung, dass automatisch PlatformIO installiert und grafische Assistenten für das Erstellen und Ausführen von Projekten zur Verfügung stellt.

2.4 Automation Studio

Automation Studio ist die Entwicklungsumgebung für speicherprogrammierbare Steuerungen von Bernecker u. Rainer.

[1] https://platformio.org/

3 Entwicklung des Übertragungsprotokolls

3.1 Übersicht

Für die Kommunikation zwischen den drei Systemen wurde ein einheitliches Übertragungsprotokoll entworfen, das auf der seriellen Datenübertragung über RS232 bzw. über Bluetooth Low Energy basiert. Für diese Schnittstellen werden von den Mikrocontrollern und der SPS Bibliotheken zur Verfügung gestellt, die das Senden von Daten als einzelne Bytes ermöglichen. Das bedeutet, dass der Anwendungscode einen unstrukturierten Datenstrom erhält, der erst durch ein selbst definiertes Protokoll eine Bedeutung erhält.

Aufgrund der begrenzten Bandbreite wurde anstatt eines textbasierten ein binäres Protokoll entwickelt, um die Daten möglichst platzsparend zu versenden. Das Protokoll ist in einer von allen Softwaresystemen gemeinsam verwendeten C++ Bibliothek implementiert. Diese übernimmt das Dekodieren der Datenströme anhand des definierten Protokolls, sowie das Konvertieren von Nutzdaten in eine Folge von Bytes, die dann über das Medium versendet werden können.

3.2 Aufbau

Das Protokoll gliedert sich einzelne Datenpakete (genannt Messages, engl. „Nachrichten“). Nachfolgend wird die Grundstruktur einer Nachricht anhand von Abbildung VII.2 beschrieben:

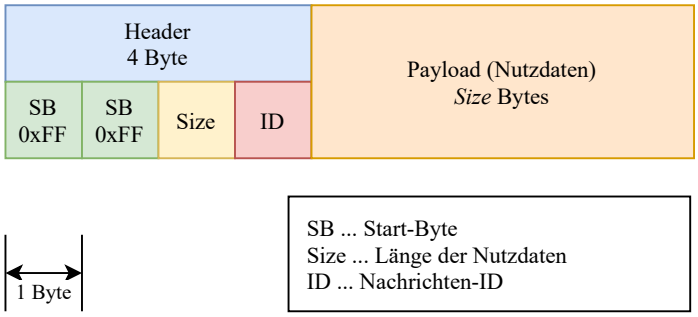


Abbildung VII.2: Aufbau einer Nachricht im Protokoll

Der Nachrichten-Header besteht aus vier Byte. Die ersten zwei Byte des Headers sind die sogenannten „Start-Bytes“, die den Anfang einer Nachricht markieren. Danach wird ein Byte mit der Länge der Nutzdaten übertragen, wodurch die maximale Länge der Nutzdaten auch auf 256 Byte begrenzt wird. Als Nächstes wird die ein Byte lange „Message-ID“ übertragen, die den Typ der Nachricht angibt. Nach dem Header werden die Nutzdaten der Nachricht übertragen.

3.3 Nutzdaten

Die Struktur der Nutzdaten hängt vom Nachrichtentyp ab. Jedem Typ ist eine C-Struktur zugeordnet, die den Aufbau der Nutzdaten angibt, nachfolgend beispielhaft eine Definition der Nutzdaten für die Nachricht `msg::init`:

Listing VII.1: Definition der Nutzdaten für `msg::init`

```
1 struct __packed msg_data_init_t : msg_data<msg::init> {
2   uint8_t version_code;
3 };
```

In C kann durch Pointer direkt auf die einzelnen Byte einer Variablen zugegriffen werden, so wie sie im Speicher liegen. Das funktioniert auch bei zusammengesetzten Datentypen. Mit der Funktion `memcpy` aus der Standardbibliothek kann ein beliebiger Speicherbereich von einer Quelladresse auf eine Zieladresse kopiert werden. So wird dieser Funktion beim Senden ein Zeiger auf eine Strukturvariable übergeben, die die Nutzdaten enthält, und als Zielort der Sendepuffer. Beim Empfangen wird dieser Prozess rückwärts durchlaufen. Hier wird ein Zeiger auf die empfangenen Nutzdaten (vom Typ `uint8_t*`, also ein Zeiger auf einzelne Byte) durch Pointer-Casting in einen Zeiger auf eine Strukturvariable umgewandelt. Über diesen Pointer kann dann wie gewohnt auf die einzelnen Elemente der Struktur zugegriffen werden.

Der oben beschriebene Prozess funktioniert allerdings nur, wenn sichergestellt werden kann, dass die beiden kommunizierenden Systeme die Struktur im Speicher exakt gleich abbilden. Wenn nur Ganzzahlen als Elemente der Struktur zugelassen werden, gibt es zwei Effekte zu beachten.

Erstens muss die sogenannte „Endianness“ aller Prozessoren übereinstimmen[2]. Diese gibt an, ob bei der Darstellung von Ganzzahlen, die mehr als ein Byte lang sind, das Most Significant Byte oder das Least Significant Byte auf der niedrigsten Adresse stehen. Die in unserem Projekt eingesetzten Systeme sind alle Little-Endian-Systeme, d.h. das Least Significant Byte steht an erster Stelle im Speicher.

Manche Systeme sind außerdem dahingehend optimiert, dass nur auf jede zweite Adresse zugegriffen wird. Das heißt Variablen, die mehr als ein Byte belegen, immer auf einer geraden Adresse beginnen. In Strukturen kann der Compiler deshalb Lücken zwischen den Variablen lassen, bzw. die Struktur „vergrößern“ damit sie im Speicher eine gerade Anzahl an Byte belegt. Durch das Angeben des Attributs `__packed` bei der Deklaration der Struktur wird dem Compiler mitgeteilt, die Struktur genau wie angegeben im Speicher abzubilden.

3.4 Verwendete Nachrichten

Die verwendeten Nachrichten sind alle in der Datei `messages.h` definiert (siehe Anhang A Listing A.2). Der Inhalt der Datei entspricht dem derzeitigen Stand der Software, sollten in Zukunft zusätzliche Funktionen zum Projekt hinzugefügt werden, können auch weitere Nachrichten definiert werden. Einige wichtige Nachrichten werden in späteren Abschnitten noch erwähnt und näher beschrieben, wobei der Anhang als Referenz dienen kann.

[2] vgl. [43], S. 242.

3.5 Verbindungsaufbau und -überwachung

Der Ablauf des Verbindungsaufbaus ist in Abbildung VII.3 dargestellt. Den ersten Verbindungsaufbau übernimmt immer das Embedded System auf dem Roboter. Dieses verbindet sich zum Fernsteuergerät und zur SPS. Beim Verbindungsaufbau (1) wird zuerst von einer Seite eine Initialisierungsnachricht gesendet (`msg::init`). Diese beinhaltet eine Versionsnummer des Protokolls, um sicherzustellen, dass alle Systeme die gleiche Software bzw. Firmware verwenden. Wenn die Protokoll-Versionen übereinstimmen, antwortet die Gegenstelle mit einer positiven Bestätigungsnachricht (2), um den erfolgreichen Verbindungsaufbau zu bestätigen.

Nach dem erfolgreichen Aufbau der Verbindung senden beide Geräte jeweils im Abstand von einer Sekunde (3a, 3b, 4a, 4b) eine Heartbeat-Nachricht (engl. „Herzschlag“), mit der die Funktionstüchtigkeit der Verbindung bestätigt wird. Der Receiver sendet mit der Heartbeat-Nachricht außerdem die Information mit, ob die jeweils andere Verbindung noch aufrecht ist. Somit erhält das Fernsteuergerät die Information, ob der Receiver noch zur SPS verbunden ist und umgekehrt.

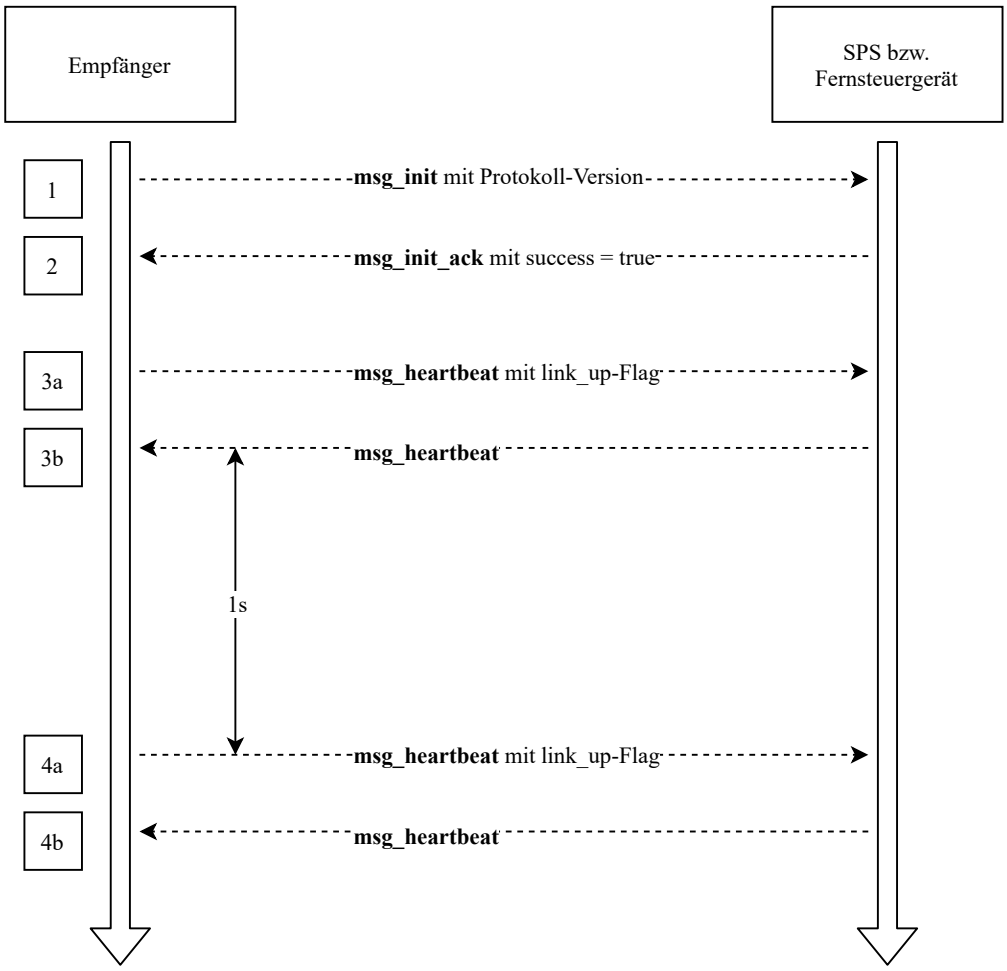


Abbildung VII.3: Ablauf bei einem erfolgreichen Verbindungsaufbau

3.6 API der Bibliothek

Das Protokoll ist in C++ durch die Klasse `Protocol` implementiert. Diese übernimmt die oben beschriebene Unterteilung des Datenstroms in Nachrichten sowie die Konvertierung der Nutzdaten zwischen Byte-Arrays und Strukturvariablen. Der restliche Code muss sich dann nur noch mit der API[3] der Bibliothek befassen.

Listing VII.2: API der `Protocol`-Klasse

```
1  typedef void(*msg_callback_t)(uint8_t* msg_data, void* user_data);
2  typedef void(*default_callback_t)(bool handled, msg msg_code, uint16_t msg_size, uint8_t*
       msg_data, void* user_data);
3  typedef void(*write_func_t)(uint8_t* data, uint16_t length, void* user_data);
4
5  class Protocol {
6
7    public:
8      Protocol();
9      ~Protocol();
10
11     void setWriteFunction(write_func_t write_function, void* write_func_user_data);
12     void registerMessageCallback(msg msg_type, msg_callback_t callback, void* user_data =
           nullptr);
13     void unregisterMessageCallback(msg msg_type);
14     void registerDefaultCallback(default_callback_t callback, void* user_data = nullptr);
15     void unregisterDefaultCallback();
16     void handle(uint8_t* data, uint16_t length);
17     void sendRaw(msg msg_code, uint16_t size, const void* data);
18
19     template<class M>
20     void send(M* msg) {
21         sendRaw(M::MSG_CODE, (uint16_t)sizeof(*msg), msg);
22     }
23 };
```

Um mit dem Protokoll über eine Schnittstelle zu kommunizieren, muss eine Instanz der Klasse `Protocol` angelegt werden, in Listing VII.2 ist die Deklaration der Klasse zu sehen. Die Klasse übernimmt die Details des Protokolls, für die verschieden Softwaresysteme muss nur noch die (systemspezifische) Ansteuerung der Schnittstelle programmiert werden. Das erfolgt, in dem die Funktion `handle` mit den neu eingetroffen Daten aufgerufen wird und über die Funktion `setWriteFunction` eine Funktion festgelegt wird, die Daten erhält, die über die Schnittstelle übertragen werden müssen. Diese Implementierung wird in der Beschreibung der jeweiligen Softwaresysteme in den Abschnitten „Implementierung des Übertragungsprotokolls“ beschrieben[4].

In den nächsten Absätzen werden die Schlüsselfunktionen der `Protocol`-Klasse noch einmal genauer erläutert, während die genaue Ansteuerung der Schnittstellen (Bluetooth Low Energy und Serielle Schnittstelle) in den Abschnitten des betroffenen Softwaresystems beschrieben wird.

[3] Application Programming Interface, von außen sichtbare Klassen und Methoden einer Bibliothek
[4] siehe Abschnitt 4.2.1, 5.2.1, 5.3.1 und 6.3.1

`setWriteFunction(write_func_t write_function, void* write_func_user_data)`

Mit dieser Funktion kann ein Pointer auf eine Funktion (`write_function`) übergeben werden, die aufgerufen wird, um die Daten über die Schnittstelle zu übertragen. Die Funktion erhält drei Parameter: einen Pointer zu den Daten die übertragenen werden sollen (`uint8_t*`), die Anzahl der zu übertragenden Bytes (`uint16_t`) und einen Pointer zum sogenannten User-Data-Objekt. Dieses wird bei jedem Funktionsaufruf der Write-Function als letztes Argument übergeben und dient dazu, eine Referenz zu einer beliebigen Variablen zu übergeben.

`handle(uint8_t* data, uint16_t length)`

Diese Funktion übergibt empfangene Daten an die `Protocol`-Klasse, die diese dekodiert und die passenden „Callbacks“ (siehe unten) aufruft.

`send(M* msg)`

Die Funktion `send` sendet eine Nachricht. Durch Templates wird für jeden Nutzdatentyp eine Funktion generiert. Die Funktion findet dann über die statische Klassenvariable `MSG_CODE` die dazugehörige Nachrichten-ID. Die Nachricht wird dann zum Senden anhand des oben definierten Protokolls in ein Byte-Array konvertiert und an die Write-Function übergeben.

`registerMessageCallback(msg mst_type, msg_callback_t callback, void* user_data = nullptr)`

Durch diese Methode kann eine Funktion registriert werden, die aufgerufen wird, wenn eine Nachricht vom Typ `msg_type` empfangen wird. Diese Funktion erhält dann als Parameter einen Pointer zu den empfangenen Nutzdaten sowie den User-Data-Pointer.

`registerDefaultCallback(default_callback_t callback, void* user_data = nullptr)`

Diese Funktion ist ähnlich zur vorigen, jedoch wird sie bei jeder empfangenen Nachricht aufgerufen. Die Callback-Funktion erhält neben Pointern zu den Nutzdaten und dem User-Data-Objekt auch die Nachrichten-ID der empfangenen Nachricht und eine `bool`-Flag die angibt, ob ein normales Message-Callback für diese Nachricht registriert (und damit schon aufgerufen) wurde.

4 Embedded System im Fernsteuergerät

4.1 Übersicht

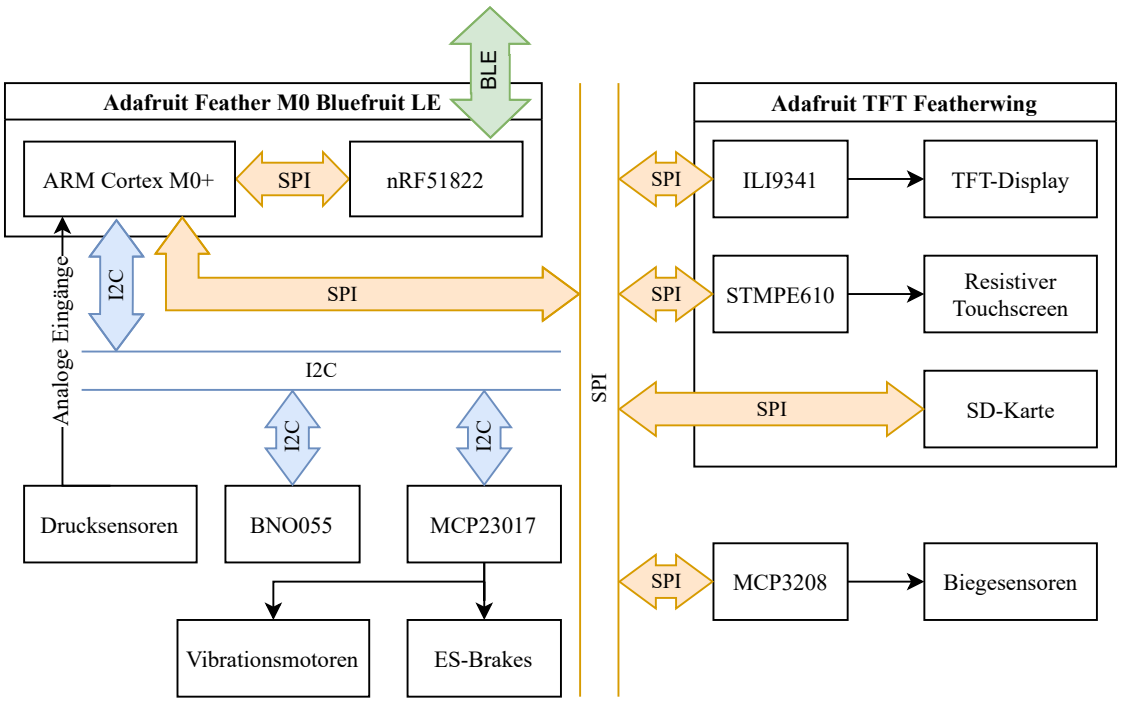


Abbildung VII.4: Topologie Embedded System im Fernsteuergerät

Das Herzstück des Fernsteuergerätes ist das Adafruit Feather M0 BLE Development Board. Auf diesem befindet sich ein ARM Cortex M0+ Prozessor (ATSAMD21G18) und ein nRF51822 Bluetooth Low Energy Modul, das über einen internen SPI-Bus mit dem Hauptprozessor kommuniziert[5]. Das Development Board ist über die Bussysteme SPI und I^2C mit zahlreichen Peripheriekomponenten verbunden:

- Display, Touchscreen und SD-Karte
- BNO055 zur Aufnahme der Bewegungsdaten
- MCP3208 ADC zur Auswertung der Biegesensoren
- MCP23017 GPIO-Baustein zur Ansteuerung der Vibrationsmotoren und ES-Brakes

Um die einzelnen Teilaufgaben zu trennen und die Software zu strukturieren, wird jede Aufgabe von einer eigenen Klasse übernommen (siehe Abbildung VII.5). Jede dieser Klassen (genannt Systeme) hat eine Initialisierungsfunktion, die aus dem `setup`-Teil des Hauptprogramms aufgerufen wird, und eine Update-Funktion, die in festgelegten Zeitintervallen vom `loop`-Teil aufgerufen wird. Allerdings ist im Gegensatz zu speicherprogrammierbaren Steuerung keine Zykluszeitüberwachung implementiert (siehe *Stand der Technik* Abschnitt 6.2).

[5] vgl. [26].

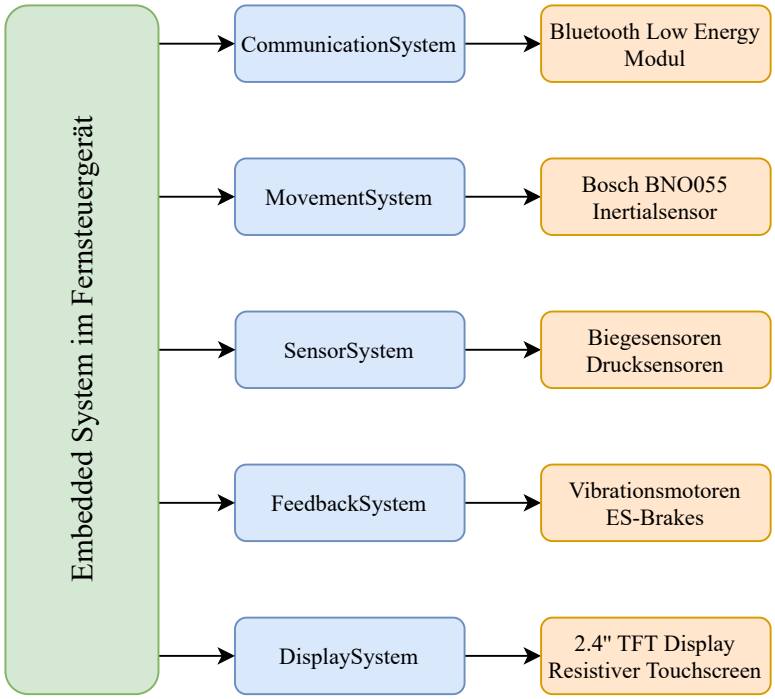


Abbildung VII.5: Zuständigkeiten der Systeme im Fernsteuergerät

Um einen Datenaustausch zwischen den Systemen zu ermöglichen, wird jedem System ein Zeiger zu einer Struktur (`SharedData`, siehe Listing VII.3) übergeben, über deren Elemente Daten ausgetauscht werden können.

Listing VII.3: `SharedData`-Struktur im Fernsteuergerät

```
enum class OperationMode {
    none,
    drive,
    move,
    info
};

enum class ConnectionStatus {
    //General connection states
    disconnected = 0x01,
    connecting,
    connected,

    //Connetion error states
    error_version_mismatch = 0xE1,
    error_timeout,
    error_unknown
};

struct Finger {
    uint16_t force = 0;
    uint8_t bendLevel = 0;
    uint8_t brakeLevel = 0;
    VibrateMode vibrateMode = VibrateMode::None;
    bool brake = false;
    bool vibrate = false;
};

```

```
29 enum class FingerCalibrationState {
30     CalibrationDone,
31     CalibrationOpenHand,
32     CalibrationCloseHand,
33     CalibrationWorking
34 };
35
36 struct FingerData {
37     Finger fingers[static_cast<uint8_t>(FingerType::COUNT)];
38     FingerCalibrationState calibration_state = FingerCalibrationState::CalibrationDone;
39     bool cmd_advance_calibration = false;
40     bool hand_closed;
41 };
42
43 struct ImuOrientation {
44     int16_t yaw, pitch, roll;
45 };
46
47 struct ImuAcceleration {
48     int16_t x, y, z;
49 };
50
51 struct ImuCalibration {
52     uint8_t sys, gyro, accel, mag;
53     uint8_t data[22];
54 };
55
56 struct ImuData {
57     ImuOrientation orientation;
58     ImuAcceleration acceleration;
59     ImuCalibration calibration;
60     bool available = false;
61 };
62
63 struct RobotData {
64     uint8_t battery_level = 0;
65     bool charging = false;
66     DriveState drive_state = DriveState::Off;
67     RobotArmState arm_state = RobotArmState::Off;
68
69     bool cmd_enable_drive = false;
70     bool cmd_enable_arm = false;
71     bool cmd_disable_drive = false;
72     bool cmd_disable_arm = false;
73     bool* home_flags = nullptr;
74 };
75
76 struct SharedData {
77     ConnectionStatus connection_status = ConnectionStatus::disconnected;
78     bool robot_link_up = false;
79     OperationMode operation_mode = OperationMode::info;
80     ImuData imu;
81     RobotData robot;
82     FingerData finger_data;
83 };
```

In den folgenden Abschnitten werden diese fünf Teilsysteme beschrieben.

4.2 Kommunikation

Wie aus der Gesamttopologie (siehe *Einleitung* Abschnitt 3) zu entnehmen, kommuniziert das Fernsteuergerät über Bluetooth Low Energy mit dem Roboter. Diese Schnittstelle wird durch das verbaute BLE-Modul bereitgestellt, wobei das Fernsteuergerät als Peripheriegerät fungiert (siehe *Stand der Technik* Abschnitt 5.5). Für die Ansteuerung wird die vom Hersteller zur Verfügung gestellte Arduino Bibliothek `Adafruit BluefruitLE nRF51` verwendet. Die Logik der Kommunikation wird in der Klasse `CommunicationSystem` implementiert.

4.2.1 Implementierung des Protokolls

Vor Beginn der Kommunikation muss das Bluetooth-Low-Energy-Modul initialisiert werden (siehe Listing VII.4). Mit der Funktion `begin()` wird das BLE-Modul initialisiert. Das Modul ist bereits so konfiguriert, dass es den Nordic UART Service (siehe *Stand der Technik* Abschnitt 5.5.2) unterstützt. Standardmäßig ist das Modul im sogenannten Command Mode, d. h. über die Funktionen `write()`, `print()`, etc. können sogenannte AT-Kommandos zum Konfigurieren des Moduls gesendet werden.

Listing VII.4: Initialisierung des BLE-Moduls

```
1   //Initialize BLE
2   Log.notice(F("Initializing Bluefruit"));
3   if (!_ble.begin(VERBOSE_MODE))
4   {
5       manager->notify_error(EC_BLUEFRUIT_NOT_DETECTED);
6   }
7   Log.notice(F("Bluefruit initialized"));
8
9   if(do_ble_factory_reset) {
10      _ble.factoryReset();
11      Log.notice(F("Performed Bluefruit factory reset"));
12      if(!_ble.atcommand("AT+GAPDEVNAME", PROTOCOL_BLE_DEVICE_NAME)) {
13          Log.warning("Failed to set BLE device name");
14      }
15      _ble.reset();
16  }
17
18  Log.notice(F("-- BLE Info --"));
19  _ble.info();
20  Log.notice(F("--------------"));
21
22  _ble.setMode(BLUEFRUIT_MODE_DATA);
```

Durch den Funktionsaufruf `setMode(BLUEFRUIT_MODE_DATA)` wechselt das Modul in den Data Mode, woraufhin die oben genannten Funktionen die Daten direkt über den Nordic UART Service an die Gegenstelle senden. Wie in Punkt 3.6 beschrieben, muss für die Implementierung des Protokolls die Write-Funktion implementiert werden sowie die Methode `handle` der Klasse `Protocol` periodisch mit neu hereingekommen Daten aufgerufen werden. Die Implementierung der Funktionen gestaltet sich aufgrund der bereits vorhandenen Bibliotheken einfach:

Listing VII.5: Write-Funktion für BLE am Fernsteuergerät

```
void CommunicationSystem::handleWrite(uint8_t* data, uint16_t length, void* user_data) {
    CommunicationSystem* comm = (CommunicationSystem*) user_data;
    comm->_ble.write(data, length);
}
```

Die von der `Protocol`-Klasse übergebenen Daten können direkt mit der `write`-Funktion übertragen werden. Im Konstruktor der `CommunicationSystem`-Klasse wird diese Funktion als „Write-Function“ festgelegt. Da nur statische Funktionen und keine Methoden übergeben werden können, wird der `this`-Pointer als „User-Data-Objekt“ übergeben.

Listing VII.6: Konstruktor der `CommunicationSystem`-Klasse

```
CommunicationSystem::CommunicationSystem(SharedData* sharedData) :
    System(COMM_UPDATE_RATE_US), _sharedData(sharedData), _ble(BLUEFRUIT_SPI_CS,
    BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST)
{
    _protocol.setWriteFunction(handleWrite, this);
    _protocol.registerMessageCallback(msg::init, handleInitMsg, this);
    _protocol.registerMessageCallback(msg::heartbeat, handleHeartbeatMsg, this);
    _protocol.registerMessageCallback(msg::vibro_trigger, handleVibroMsg, this);
    _protocol.registerMessageCallback(msg::brake_update, handleBrakeMsg, this);
    _protocol.registerMessageCallback(msg::battery_state, handleBatteryStateMsg, this);
    _protocol.registerMessageCallback(msg::robot_state, handleRobotStateMsg, this);
}
```

Im zyklischen Teil des `CommunicationSystem` werden dann die vom Bluetooth LE Modul empfangenen Daten in ein Byte-Array gelesen und dann an das `Protocol`-Objekt übergeben.

Listing VII.7: Übergeben der empfangenen Daten

```
    uint16_t available;
    while((available = _ble.available())) {
        uint16_t length = _ble.readBytes(rx_buffer, min(available, sizeof(rx_buffer)));
        _protocol.handle(rx_buffer, length);
    }
```

4.2.2 Zyklisches Senden der Bewegungsdaten

Je nach Bedienmodus des Fernsteuergerätes werden die vom `MovementSystem` und `SensorSystem` ausgewerteten Sensordaten als unterschiedliche Nachrichten gesendet.

Im Bedienmodus „Fahren“ wird die Lage des 9-Achsen-Sensors ausgewertet. Die Neigung des Sensors wird als Vorwärts- und Seitwärts-Geschwindigkeit verwendet. Dazu wird der Neigungswinkel auf eine Skala von -100% bis $+100\%$ umgerechnet. Diese Werte werden dann mit der Nachricht `drive_velocity` an das Robotersystem übertragen.

Im Bedienmodus „Greifer“ werden die linearen Bewegungen des 9-Achsen-Sensors und die Winkel der Finger gesendet. Die Bewegungen der Hand sind schon vom `MovementSystem` in ein geeignetes Format umgerechnet worden, das `SensorSystem` rechnet den Winkel der Finger auch auf eine Skala von 0% bis 100% um. Diese Werte müssen nur noch kopiert und mit der Nachricht `robot_move` an das Robotersystem gesendet worden.

4.3 Auswertung des 9-Achsen-Sensors

Der 9-Achsen-Sensor BNO055 ist über den I^2C-Bus an den Mikrocontroller angeschlossen. Zuständig für die Auswertung des Sensors ist die Klasse `MovementSystem`, intern übernimmt die Ansteuerung des Sensors die Klasse `Adafruit_BNO055` der gleichnamigen Bibliothek.

4.3.1 Initialisierung

Listing VII.8 zeigt die Initialisierungssequenz des 9-Achsen-Sensors. Nachdem der Sensor mit der Methode `begin()` initialisiert wurde, beginnt der Sensor zu arbeiten. Während der Messungen kalibriert sich der Sensor laufend selbst. Der Fortschritt der Kalibrierung kann mit der Funktion `getCalibration()` ausgelesen werden. Diese gibt den Status der Kalibrierung für das Gesamtsystem, sowie der drei einzelnen Sensoren auf einer Skala von 0 – 3 aus, wobei der Wert 3 eine vollständige Kalibrierung bedeutet.

Listing VII.8: Initialisierungssequenz BNO055

```
1  void MovementSystem::init(SystemManager* manager) {
2    /* Initialise the sensor */
3    if(!_bno.begin())
4    {
5      /* There was a problem detecting the BNO055 ... check your connections */
6      manager->notify_error(EC_IMU_NOT_DETECTED);
7    }
8
9    delay(200);
10
11   /* Use external crystal for better accuracy */
12   _bno.setExtCrystalUse(true);
13
14   /* Load calibration data */
15   _bno.setSensorOffsets(IMU_CALIBRATION_DATA);
16
17   /* Display some basic information on this sensor */
18   displaySensorDetails(&_bno);
19 }
```

4.3.2 Auslesen der Daten

Die Daten vom BNO055 werden zyklisch alle 10 ms eingelesen. Dazu wird – wie in Listing VII.9 zu sehen – die Funktion `getEvent()` aufgerufen, welche die Daten in eine Struktur vom Typ `sensors_event_t` speichert. Die Winkel in den drei Raumrichtungen müssen dann nur noch an die Einbaurichtung des BNO055 angepasst werden (Listing VII.9 Zeile 7 bis 9 und Abbildung VII.6).

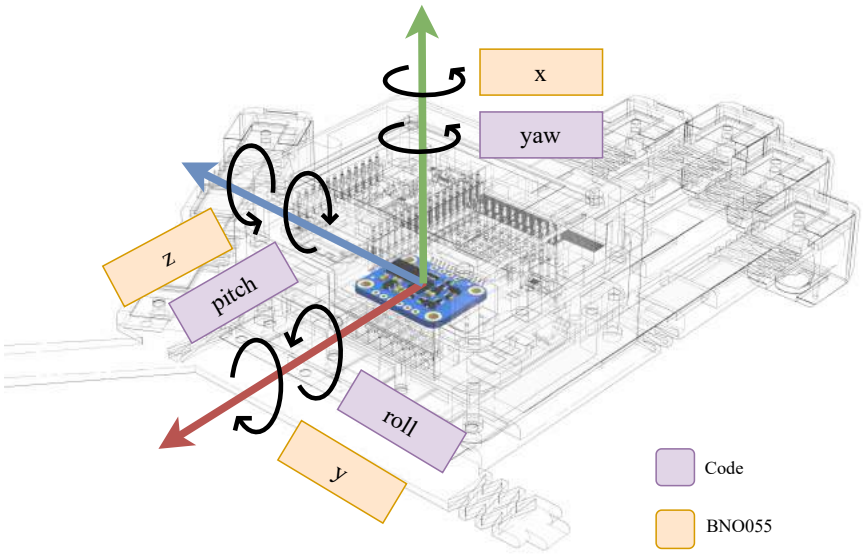


Abbildung VII.6: Achsrichtungen BNO055

Danach werden die Daten zusammen mit dem aktuellen Kalibrierungszustand an die `SharedData`-Variable übergeben, damit sie vom `CommunicationSystem` über Bluetooth Low Energy gesendet werden können.

Listing VII.9: Auslesen der BNO055-Sensordaten

```
1  ImuData* imu = &(_sharedData->imu);
2
3  //Read sensor calibration status
4  _bno.getCalibration(
5    &(imu->calibration.sys),
6    &(imu->calibration.gyro),
7    &(imu->calibration.accel),
8    &(imu->calibration.mag)
9  );
10
11 imu->available = imu->calibration.sys > 1;
12
13 if(imu->available) {
14   //Get a new sensor event
15   sensors_event_t event;
16   _bno.getEvent(&event);
17
18   //Map BNO axes to yaw, pitch and roll
19   imu->orientation.yaw = (int16_t)event.orientation.x;
20   imu->orientation.pitch = -(int16_t)event.orientation.z;
21   imu->orientation.roll = -(int16_t)event.orientation.y;
22
23   //Read calibration data
24   _calib_count++;
25   if(_calib_count > CALIBRATION_UPDATE_INTERVAL) {
26     _calib_count = 0;
27     _bno.getSensorOffsets(imu->calibration.data);
28   }
29 }
```

4.4 Auswertung der Biege- und Drucksensoren

Die Klasse `SensorSystem` ist zuständig für das Auswerten der Biege- und Drucksensoren. Die Biegesensoren sind über einen Spannungsteiler an einen externen ADC angeschlossen (siehe Kapitel *Fernsteuergerät* Abschnitt 4.2.2). Die Drucksensoren werden ebenfalls über einen Spannungsteiler ausgelesen, allerdings mit dem internen ADC des Mikrocontrollers (siehe Kapitel *Fernsteuergerät* Abschnitt 4.2.3). Das Auslesen beider Werte erfolgt zyklisch alle 50 ms.

4.4.1 Ansteuerung des MCP3208

Der externe ADC MCP3208 ist über den SPI-Bus an den Mikrocontroller angeschlossen. Die Ansteuerung dieses Bausteins wurde ohne Bibliothek realisiert, da diese nach einem einfachen Protokoll[6] funktioniert, welches nachfolgend kurz anhand des Codes beschrieben wird.

Listing VII.10: Auslesen von Analogwerten mit dem MCP3208

```
1  uint16_t SensorSystem::readADC(uint8_t channel) {
2      SPI.beginTransaction(SPISettings(1600000, MSBFIRST, SPI_MODE0));
3      digitalWrite(MCP3208_SPI_CS, LOW);
4
5      uint16_t cmd = 0;
6      uint16_t data = 0;
7      cmd |= bit(10); //Start bit
8      cmd |= bit(9); //Single ended bit
9      cmd |= (channel & 0b111) << 6; //Set channel
10
11     SPI.transfer((cmd >> 8) & 0xFF); //Transfer high byte of command
12     data |= (SPI.transfer(cmd & 0xFF) & 0x0F) << 8; //Transfer low byte of command and
           receive high byte of result
13     data |= SPI.transfer(0x00); //Transfer 0 and receive low byte of result
14
15     digitalWrite(MCP3208_SPI_CS, HIGH);
16     SPI.endTransaction();
17
18     return data;
19 }
```

Nachdem der SPI-Bus initialisiert wurde (Zeile 2), wird das Slave-Select-Signal (hier `MCP3208_SPI_CS` für Chip-Select) des MCP3208 auf Low gelegt (Zeile 3), um den MCP3208 zu aktivieren.

Zuerst wird der zwei Byte lange Lesebefehl berechnet (Zeile 5 bis 9). Die ersten fünf Bit (Bits 15-11) des Befehls sind null, danach wird das Start-Bit (Bit 10, mit dem Wert 1) übertragen, das den Start der Messung signalisiert. Das nächste Bit (9) erhält auch den Wert 1, um eine Messung gegen Masse anstatt einer Messung zwischen zwei Eingängen durchzuführen. Darauf folgen drei Bit (Bits 8-6), die den zu lesenden Eingang (= Kanal) festlegen. Die restlichen Bit werden vom MCP3208 ignoriert und bleiben daher null.

Nun findet die eigentliche Übertragung statt. Zuerst werden die Bits 15 bis 8 des Befehls übertragen, währenddessen sendet der MCP3208 keine Daten zurück (Zeile 11). Danach werden die Befehls-Bits 7 bis 0 übertragen, während der ADC bereits die ersten vier Bit der Messung zurückgibt (Zeile 12). Zu guter Letzt muss noch ein Byte übertragen werden (dessen Wert ist egal), damit der MCP3208 noch die letzten acht Bit der Messung

[6] vgl. [44], Figure 6-1.

zurückliefern kann[7] (Zeile 13). Damit ist der Analogwert fertig ausgelesen, der ADC wird über die Slave-Select-Leitung deaktiviert und der Messwert zurückgegeben.

4.4.2 Auslesen des internen ADCs

Der Zustand der Drucksensoren wird über den internen ADC ausgelesen, wofür das Arduino-Framework die Funktion `analogRead()` bereitstellt.

4.4.3 Skalierung der gemessenen Werte

Die Rohwerte werden anhand einer linearen Funktion auf eine Skala von 0 bis 100 umgerechnet. Das Arduino-Framework stellt für eine solche Umrechnung die Funktion `map`[4] zur Verfügung. Die Parameter der Umrechnungsfunktion wurden experimentell durch Messen der maximalen und minimalen Werte des ADCs bestimmt und für jeden Finger in einem Array im Programmcode hinterlegt. Abbildung VII.7 zeigt beispielhaft eine solche Skalierungskennlinie für den Biegesensor des Zeigefingers.

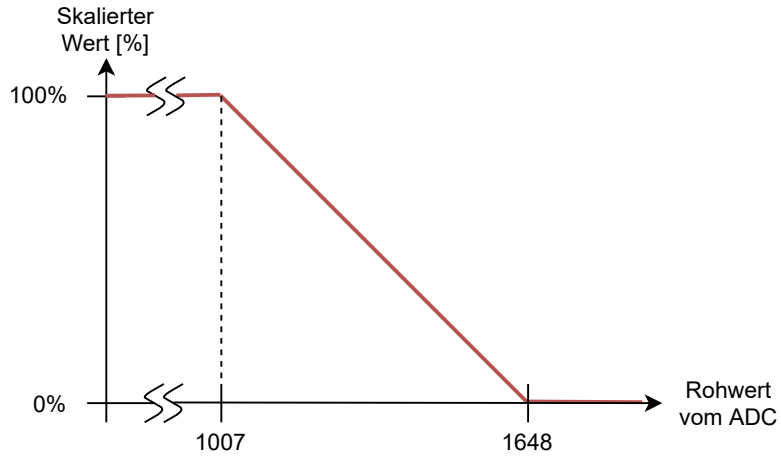


Abbildung VII.7: Skalierungskennlinie Biegesensor

4.5 Steuerung des haptischen Feedbacks

Die Steuerung des haptischen Feedbacks wird von der Klasse `FeedbackSystem` übernommen und besteht aus der Ansteuerung der ES-Brakes und der Vibrationsmotoren. Beide werden über den externen GPIO-Baustein MCP23017, der am I^{2}C-Bus angeschlossen ist, angesteuert (siehe Abbildung VI.23).

4.5.1 MCP23017

Der MCP32017 ist ein externer GPIO-Baustein, der über 16 Input/Output-Ports verfügt. Die Details der Ansteuerung werden von der Bibliothek `Adafruit MCP23017 Arduino Library` übernommen, weshalb auf das genaue I^{2}C Ansteuerungsprotokoll nicht näher eingegangen wird. Die API der Bibliothek ist dem Arduino-Framework nachempfunden, zum Beispiel sind auch die Funktionen `pinMode` und `digitalWrite` verfügbar.

[7] siehe *Stand der Technik* Abschnitt 5.2, der Slave kann von sich aus keine Daten übertragen

4.5.2 Vibrationsmotoren

Über die SharedData-Struktur kann für jeden Finger unabhängig ein Vibrationsmuster gestartet werden. Diese Muster sind als Arrays im Code gespeichert, das Format wird am Beispiel des folgenden Musters erläutert:

```
const uint32_t PATTERN_DOUBLE[] = {500, 500, 500, 0};
```

Die Elemente des Arrays sind jeweils Zeiten in Millisekunden, die angeben, wie lange der Vibrationsmotor ein- und ausgeschaltet wird. In diesem Fall wird der Motor 500 ms lang ein-, dann 500 ms lang aus- und schließlich wieder 500 ms lang eingeschalten. Am Ende des Arrays ist immer eine null eingetragen, damit der Code erkennen kann, dass das Ende des Musters erreicht ist.

4.5.3 ES-Brakes

Die Logik für die Aktivierung der ES-Brakes ist vergleichsweise einfach. Dazu werden für jeden Finger aus der `SharedData` Struktur die Werte `bendLevel`, `brakeLevel` und `force` herangezogen. Solange der Wert `bendLevel` größer als `brakeLevel` ist, ist die Bremse aktiviert. Liegt der Wert `force` jedoch über einem Schwellwert – was bedeutet, dass der Nutzer die Finger wieder ausstrecken will – wird die Bremse auf jeden Fall deaktiviert.

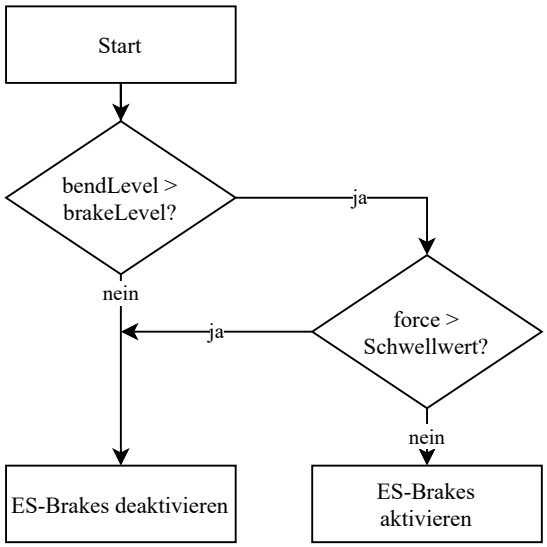


Abbildung VII.8: Flussdiagramm ES-Brake Aktivierung

4.6 Benutzeroberfläche

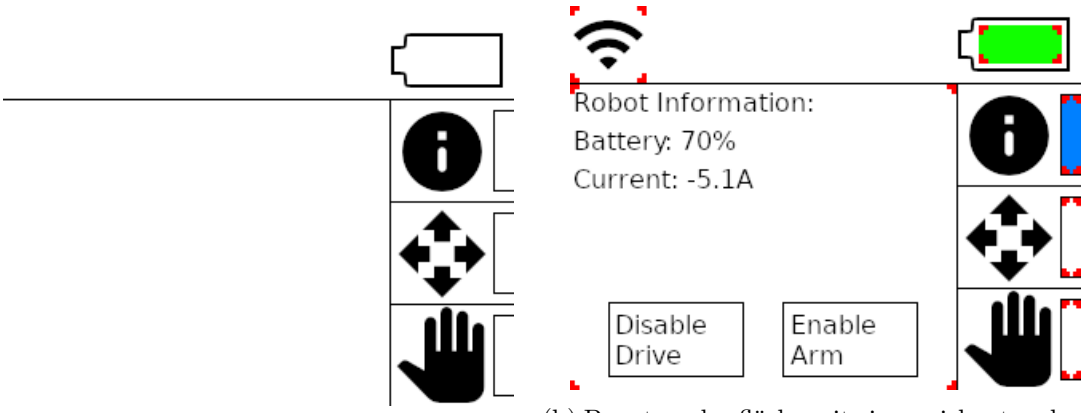
Das Fernsteuergerät verfügt über ein Display mit Touchscreen, über das Informationen angezeigt werden können und zwischen Bedienmodi gewechselt werden kann. Die für die Oberfläche verwendeten Grafiken benötigen mehr Speicher als der Mikrocontroller zur Verfügung stellen kann, deshalb ist auf der Rückseite des Displays auch ein SD-Karten-Slot verbaut, auf dem größere Grafiken gespeichert werden können. Display, Touchscreen und SD-Karten-Slot sind alle, wie in der Topologie beschrieben, am SPI-Bus angeschlossen. Zuständig für die Darstellung der Benutzeroberfläche auf dem Bildschirm ist die Klasse `DisplaySystem`, die dazu mehrere Bibliotheken benötigt:

- `Adafruit GFX Library`: Darstellung von Grafiken und Text auf Displays (unabhängig vom Typ des Displays)

- `Adafruit ILI9341`: Konkrete Implementierung der `Adafruit GFX Library` für das verwendete Display
- `Adafruit STMPE610`: Ansteuerung des resistiven Touchscreen-Controllers
- `Adafruit ImageReader Library`: Laden von Bildern von der SD-Karte
- `SdFat - Adafruit Fork`: Lesen von FAT Dateisystemen, benötigt von der `Adafruit ImageReader Library`
- `Adafruit SPIFlash`: Ansteuerung von SD-Karten über SPI, benötigt von der `Adafruit ImageReader Library`
- `Adafruit EPD`: Benötigt zum Kompilieren der `Adafruit ImageReader Library`

4.6.1 Anforderungen an die Oberfläche

Das Design der Benutzeroberfläche wurde von verschiedenen Anforderungen beeinflusst. Erstens sollte die Oberfläche einfach gehalten werden (große Schaltflächen), um eine sichere Bedienung auf dem verhältnismäßig kleinem Display zu ermöglichen. Außerdem ist durch den SPI-Bus die Zeit, die benötigt wird, um das Bild am Display zu ändern sehr hoch. Deshalb wird ein statisches Hintergrundbild (siehe Abbildung VII.9) verwendet und nur einzelne Regionen des Displays (rot eingezeichnet) werden neu gezeichnet.



(a) Statische Hintergrundgrafik

(b) Benutzeroberfläche mit eingezeichneten dynamischen Regionen

Abbildung VII.9: Minimierung der zu zeichnenden Fläche

4.6.2 Implementierung

Die Implementierung der Benutzeroberfläche ist recht einfach gehalten. Beim Start des Mikrocontrollers wird zuerst das in Abbildung VII.9a gezeigte Hintergrundbild gezeichnet. Im zyklischen Teil des Programms wird alle 50ms der in Abbildung VII.10 dargestellte Ablauf ausgeführt.

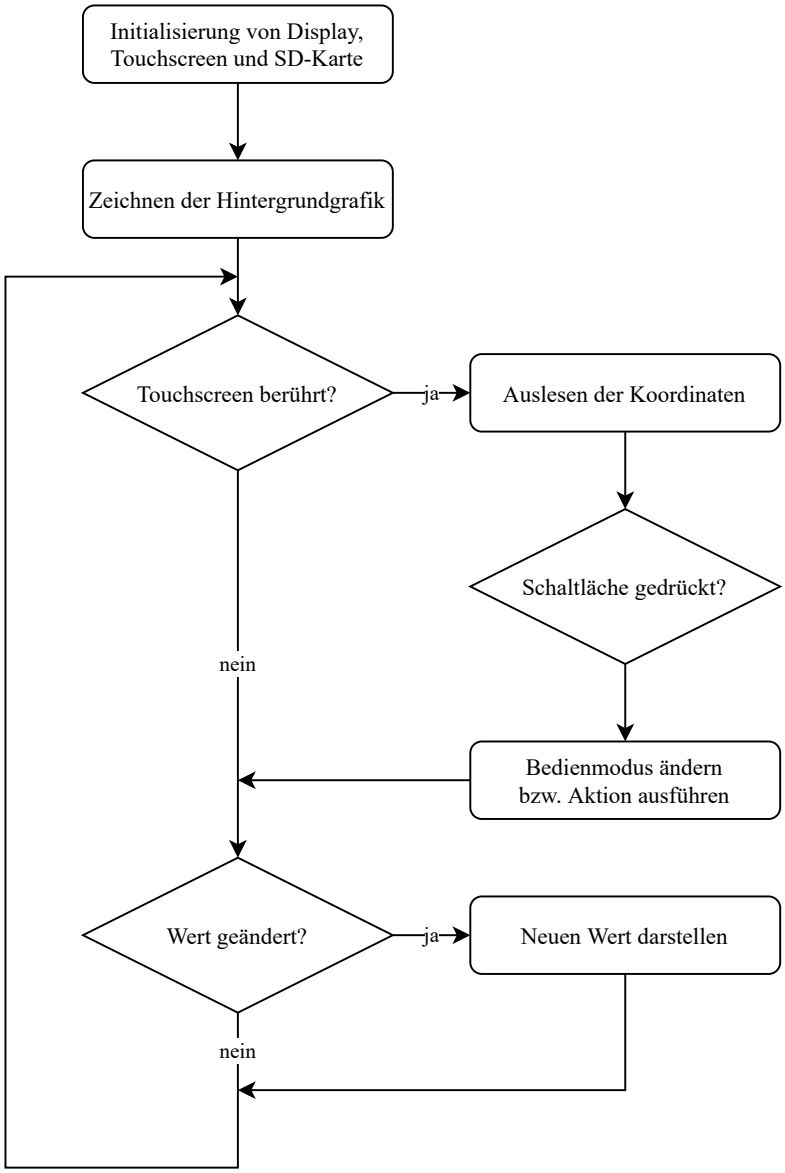


Abbildung VII.10: Ablaufdiagramm des Display-Systems

Zuerst wird der Touchscreen-Controller ausgewertet und überprüft, ob eine Schaltfläche betätigt worden ist. Ist dies der Fall, wird die entsprechende Aktion ausgeführt, zum Beispiel wechseln des Bedienmodus. Dazu wird einfach der gewünschte Wert in der `SharedData`-Struktur geändert, um dieses Ereignis den anderen Systemen mitzuteilen. Danach wird überprüft, ob sich ein relevanter Wert der `SharedData`-Struktur geändert hat, indem er mit dem zuletzt dargestellten Wert verglichen wird. So werden nur jene Werte aktualisiert, die sich auch geändert haben, was wieder der Minimierung der zu zeichnenden Pixelanzahl dient.

5 Embedded System am Roboter (Receiver)

5.1 Übersicht

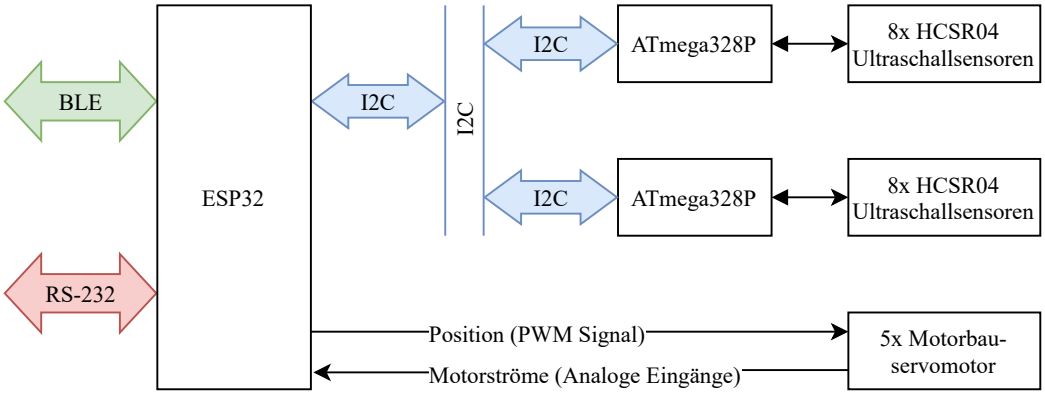


Abbildung VII.11: Topologie Embedded System am Roboter

Das Embedded System am Roboter besteht aus mehreren Mikrocontrollern, die über den I^2C-Bus verbunden sind. Die zentrale Rolle nimmt der ESP32 ein, der auch am I^2C-Bus die Rolle des Masters übernimmt. Er ist zuständig für die Kommunikation mit dem Fernsteuergerät über die integrierte Bluetooth-Schnittstelle, sowie die Kommunikation mit der SPS über eine serielle Schnittstelle. Außerdem übernimmt er die Ansteuerung der Modellbau-Servos im Greifer und die Messung deren Motorströme. Die beiden ATmega328P-Controller sind für die Auswertung der Ultraschallsensoren verantwortlich. Sie sind als Slaves am I^2C-Bus angeschlossen und übermitteln auf Anfrage die Abstands-Daten an den ESP32.

Auch im Receiver werden die Aufgabenbereiche durch verschiedene „Systeme“ übernommen, Abbildung VII.12 zeigt die Aufgabenteilung:

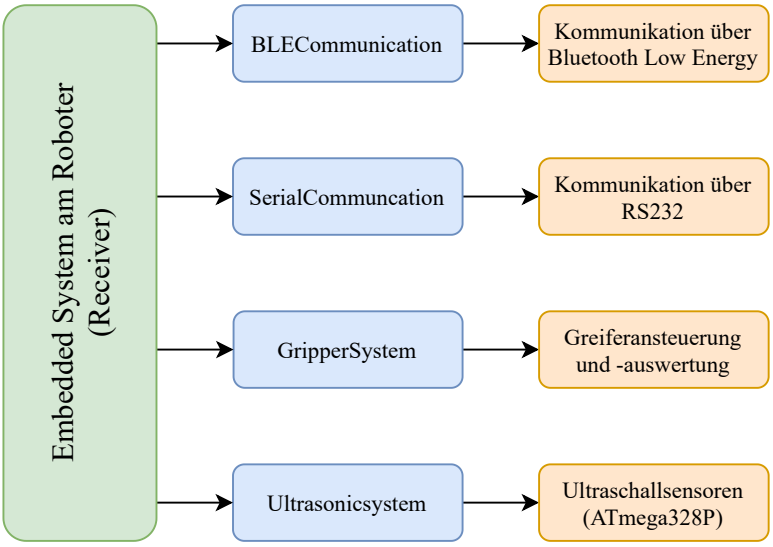


Abbildung VII.12: Zuständigkeiten der Systeme im Receiver

Der Datenaustausch erfolgt wieder über eine globale Strukturvariable (Typ `SharedData`), deren Deklaration in Listing VII.11 zu sehen ist.

Listing VII.11: `SharedData`-Struktur im Receiver

```
#ifndef SHARED_DATA_H
#define SHARED_DATA_H

#include "inttypes.h"
#include "connection_lifecycle.h"

struct Finger {
    float target_position;
    float current_position;
    uint16_t current_load;
    bool block;
};

struct SharedData {
    ConnectionStatus bleStatus = ConnectionStatus::disconnected;
    ConnectionStatus serialStatus = ConnectionStatus::disconnected;
    bool gripperEnabled = false;
    Finger fingers[5];
    bool ultrasonic1_available = false;
    bool ultrasonic2_available = false;
    uint8_t distances[16];
};

#endif
```

5.2 Kommunikation mit dem Fernsteuergerät

Die Kommunikation mit dem Fernsteuergerät erfolgt über Bluetooth Low Energy und ist in der Klasse `BLECommunication` implementiert. Dabei stellt der ESP32 den Client (siehe *Stand der Technik* Abschnitt 5.5) dar und verbindet sich mit dem Serial Port Service des Fernsteuergeräts. Das Arduino-Framework stellt eine Bibliothek zur Ansteuerung der integrierten Bluetooth Schnittstelle zur Verfügung.

5.2.1 Implementierung des Übertragungsprotokolls

Abbildung VII.3 stellt den groben Ablauf des Verbindungsaufbaus dar. Zuerst wird über die Bibliothek nach verfügbaren Bluetooth LE Geräten gesucht, danach wird überprüft, ob das Gerät den Serial Port Service unterstützt. Wenn ein solches Gerät gefunden wurde, wird ein Verbindungsversuch unternommen. Ist dieser erfolgreich, wird auf der *RX-Characteristic* ein *Notify-Handler* (siehe *Stand der Technik* Abschnitt 5.5.2) registriert, der bei neu eintreffenden Daten aufgerufen wird. Für das Senden von Daten muss nur über die Bibliothek auf *TX-Characteristic* geschrieben werden.

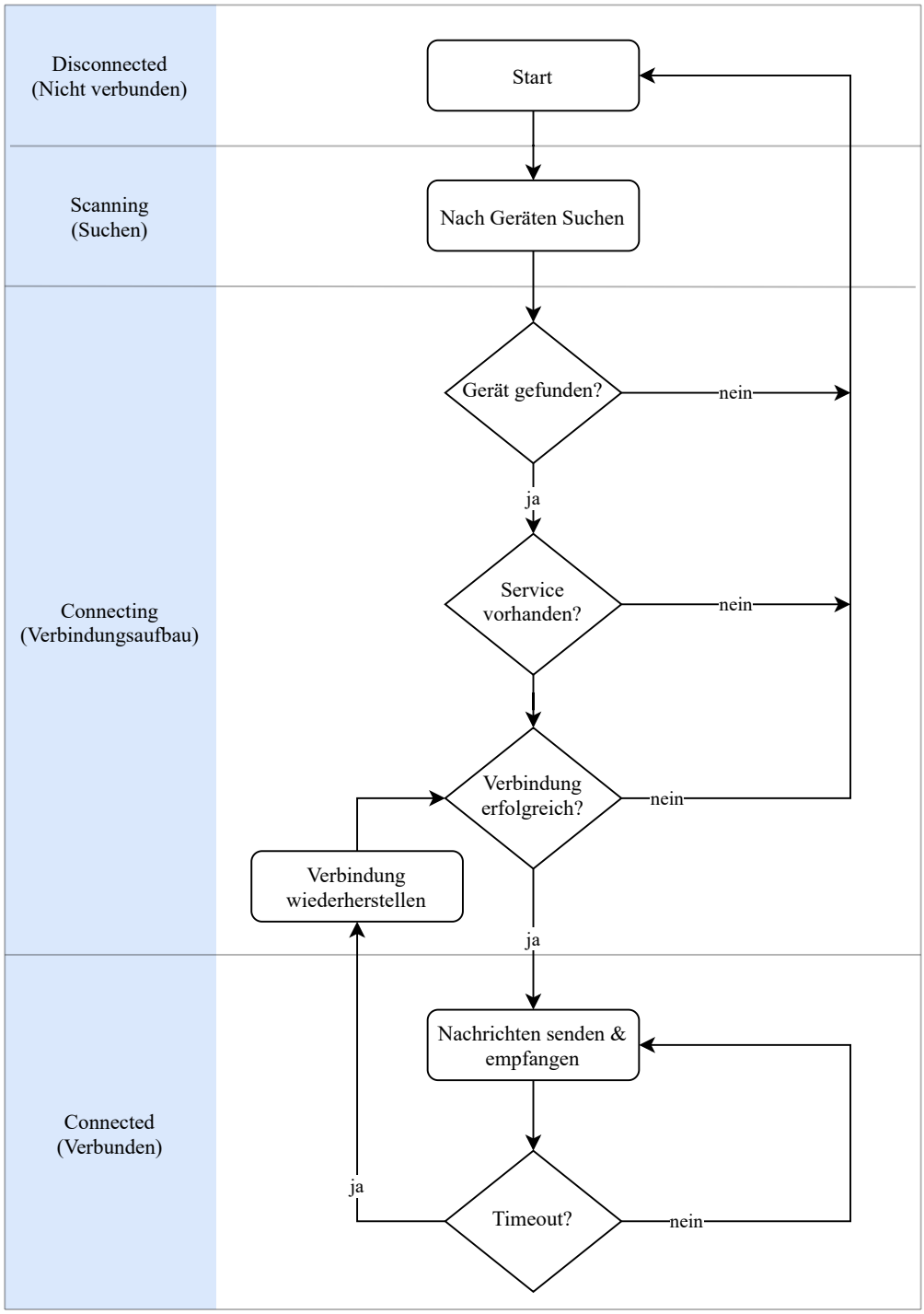


Abbildung VII.13: Ablaufdiagramm Verbindungsaufbau zum Fernsteuergerät

Die Details der Kommunikation über das in Abschnitt 3 beschriebene Protokoll übernimmt wieder die Klasse `Protocol`. Dazu wird im Notify-Handler die Funktion `Protocol::handle` mit den neu empfangen Daten aufgerufen. Die `writeHandler`-Funktion überträgt die zu sendenden Daten mit der Funktion `BLECharacteristic::writeValue` aus dem Arduino-Framework über die TX-Characteristic an das Fernsteuergerät.

Listing VII.12: Implementierung des Protokolls für die BLE-Schnittstelle am Receiver

```
    BLECommunication* comm = (BLECommunication*) user_data;
    if(!comm->_connected) {
        Log.warning(F("[BLETask] Interface not connected, cannot write messages"));
        return;
    }
    while(length > 0) {
        uint16_t to_write = min(length, (uint16_t)20);
        if(comm->_pTxCharacteristic == nullptr) {
            Log.error("[BLETask] TXCharacteristic == null");
            return;
        }
        comm->_pTxCharacteristic->writeValue(data, to_write);
        length -= to_write;
        data += to_write;
    }
}

void BLECommunication::notifyCallback(
  BLERemoteCharacteristic* pBLERemoteCharacteristic,
  uint8_t* pData,
  size_t length,
  bool isNotify) {
    BLECommunication* comm =
        BLECommunication::_notfiyCharacteristicMap[pBLERemoteCharacteristic];
    if(comm != nullptr) {
        comm->_protocol.handle(pData, length);
    }
}
```

5.2.2 Empfangen der Greifer-Daten

Über die Nachricht `msg::finger_update` empfängt der Receiver die Winkel-Daten der 5 Finger vom Fernsteuergerät. Diese werden in der `SharedData`-Variable gespeichert, damit sie vom `GripperSystem` zur Ansteuerung des Greifers ausgelesen werden können.

5.2.3 Senden der Feedback-Informationen

Die `BLECommunication`-Klasse bekommt vom `GripperSystem` die Information, welche Finger „blockieren“ (siehe Abschnitt 5.5). Wenn sich dieser Zustand ändert, wird die neue Information mit der Nachricht `msg::vibro_trigger` (zum Aktivieren der Vibrationsmotoren) und der Nachricht `msg::brake_update` (zum Ansteuern der ES-Brakes) an das Fernsteuergerät gesendet.

Listing VII.13: Senden der Feedbackinformation

```
    uint8_t new_block_flags = 0;
    for (uint8_t i = 0; i < 5; i++)
    {
        if(_sharedData->fingers[i].block) {
            new_block_flags |= (1<<i);
        }
    }
    if(new_block_flags != _gripper_block_flags) {
        _gripper_block_flags = new_block_flags;
        msg_data_brake_update_t brake_update;
        for (uint8_t i = 0; i < 5; i++)
        {
```

```
13              brake_update.brake_levels[i] = (uint8_t)
                    _sharedData->fingers[i].current_position;
14          }
15          msg_data_vibro_trigger_t vibro_trigger;
16          vibro_trigger.mode = VibrateMode::Short;
17          vibro_trigger.mask = new_block_flags;
18
19          _protocol.send(&brake_update);
20          _protocol.send(&vibro_trigger);
21      }
```

5.2.4 Nachrichtenweiterleitung

Wie bereits in der Aufgabenbeschreibung erwähnt, ist der Receiver auch dafür zuständig, Nachrichten vom Fernsteuergerät an die SPS weiterzuleiten. Dazu werden alle Nachrichten, die dem Receiver unbekannt sind, einfach an die serielle Schnittstelle weitergegeben.

5.3 Kommunikation mit der SPS

Neben der Bluetooth Low Energy Verbindung zum Fernsteuergerät ist der ESP32 über eine serielle Schnittstelle (RS232) auch mit der SPS verbunden. Zuständig für die Kommunikation über diese ist die Klasse `SerialCommunication`.

5.3.1 Implementierung des Übertragungsprotokolls

Da über die Klasse `Serial` mit den Funktionen `read` und `write` die Ansteuerung der seriellen Schnittstelle sehr einfach ist, fällt die Implementierung des Protokolls für die Kommunikation mit der SPS sehr einfach aus.

Im zyklischen Teil wird die Schnittstelle mit der Funktion `available` auf neue Daten überprüft, die dann mit `Protocol::handle` zur Dekodierung übergeben werden. Die „Write-Function“ übergibt das Datenarray einfach an die Funktion `Serial.write()`, um die Daten über die serielle Schnittstelle zu übertragen.

Listing VII.14: Implementierung des Protokolls für die serielle Schnittstelle am Receiver

```
1  void SerialCommunication::handleWrite(uint8_t* data, uint16_t length, void* user_data) {
2      SERIAL_COMM_INTERFACE.write(data, length);
3  }
4
5  void SerialCommunication::update() {
6      //Interface update
7      int available;
8      while((available = SERIAL_COMM_INTERFACE.available())) {
9          uint16_t length = SERIAL_COMM_INTERFACE.readBytes(rx_buffer,
               min((int)sizeof(rx_buffer), available));
10         _protocol.handle(rx_buffer, length);
11     }
12 }
```

5.3.2 Senden der Abstandsdaten an die SPS

Der Receiver wertet die Abstandsdaten der Ultraschallsensoren aus (siehe 5.6). Diese werden mit der Nachricht `msg::distance_readings` an die SPS zur Kollisionsvermeidung übertragen.

5.3.3 Nachrichtenweiterleitung

Damit auch Nachrichten von der SPS ans Fernsteuergerät gesendet werden können, leitet auch die `SerialCommunication`-Klasse unbekannte Nachrichten über `BLECommunication`-Klasse an das Fernsteuergerät weiter.

5.4 Ansteuerung des Greifers

Der eingesetzte Greifer wird durch fünf Modellbauservos angetrieben. Diese können durch ein einfaches Rechtecksignal auf eine bestimmte Position gefahren werden. Dabei gibt die Zeit, in der das Signal auf einem High-Pegel liegt, die Position an. Zur Erzeugung dieses Signals werden die 16-Bit Timer des ESP32 verwendet, womit der Duty-Cycle mit Werten von 0 bis 65535 eingestellt werden kann.

Welche Werte zu einem voll ausgestreckten bzw. angewinkelten Finger führen, wurde experimentell ermittelt. Die Werte vom Fernsteuergerät, die von 0 (voll ausgestreckt) bis 100 (voll angewinkelt) reichen, werden durch die Funktion `map` auf die benötigten Tastverhältnisse umgerechnet.

5.5 Auswertung der Motorströme

Um erkennen zu können, ob der Greifer ein Objekt gegriffen hat, sind in die Rückleitungen der Servos Strommesswiderstände eingebaut (siehe Hardware). Der Spannungsabfall an diesen Widerständen wird über einen Tiefpassfilter mit den analogen Eingängen am ESP32 gemessen.

Nach erneutem Filtern durch einen gleitenden Durchschnitt ergibt sich ein Wert, der ein Maß für die Last am Servomotor ist. Ein hoher Wert kann prinzipiell zwei Ursachen haben: einerseits steigt der Strom während dem Beschleunigungsvorgang des Motors an, anderseits, wenn der Motor die gewünschte Position aufgrund eines mechanischen Widerstands nicht erreichen kann. Der erstgenannte Effekt ist aufgrund der geringen Eigenmasse der Finger-Mechanik wesentlich kleiner, was die Erkennung von Objekten im Greifer erleichtert. So wird ein Objekt im Greifer erkannt, sobald ein gewisser Schwellwert überschritten wird.

5.6 Auswertung der Ultraschall-Sensoren auf dem ATmega328P

An den ATmega328P sind können jeweils acht Ultraschallsensoren vom Typ HC-SR04 angeschlossen werden (nur sieben Anschlüsse in Verwendung). Diese Sensoren benötigen je zwei Datenleitungen zur Ansteuerung: Echo und Trigger. Durch einen Puls auf der Trigger-Leitung wird die Messung gestartet, wie in *Stand der Technik* Abschnitt 3.1 beschrieben. Der Messwert wird dann als Puls über die Leitung Echo ausgegeben. Die Länge des Pulses entspricht der Zeit die der Schall vom Sender, zum Objekt und wieder zurück zum Empfänger benötigt hat. Damit kann dann mithilfe von Gleichung III.1 unter Berücksichtigung der Schallgeschwindigkeit auf die Distanz zurückgerechnet werden:

$$d = \frac{\tau c}{2} = \frac{343\frac{m}{s} * 10^{-}6\frac{s}{\mu s} * 100\frac{cm}{m} * t}{2}$$
$$\Rightarrow d = 0,01715\frac{cm}{\mu s} * t \approx \frac{t}{58\frac{\mu s}{cm}} \quad \text{(VII.1)}$$

Mit dieser Formel ist die Implementierung der Distanzmessung mithilfe des Arduino-Frameworks trivial:

Listing VII.15: Auswertung der Ultraschallsensoren

```
}

uint8_t readSensor(uint8_t sensor) {
    digitalWrite(trigPin(sensor), HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin(sensor), LOW);
    uint32_t duration = pulseIn(echoPin(sensor), HIGH, 15000); //max 15ms, ca. 255cm
    if(duration == 0) return 255; //0 duration means timeout -> return maximum distance
    uint32_t cm = duration / 58; //distance = v*t / 2 = 343 m/s * 0.000001 s/µs * 100
        cm/m * t[µs] / 2 = t[µs] / 58
    if(cm > 255) cm = 255;    //clamp to 255cm max
```

5.7 Übertragung der Abstandsdaten über I²C

Die Abstandsdaten werden anschließend wie eingangs erwähnt über I²C übertragen. Dazu wurde die ATmega328P-Controller so programmiert, dass sie als I²C-Slaves arbeiten. Die Übertragung basiert wie in Kapitel *Stand der Technik* Abschnitt 5.3 beschrieben über Register, die durch den I²C Master, in diesem Fall den ESP32, angesprochen werden können. Tabelle VII.1 listet die definierten Register auf.

Name	Adresse	R/W	Länge	Beschreibung
`COMM_TEST`	0x01	R/W	1 Byte	Kommunikations-Testregister
`ENABLE`	0x10	W	1 Byte	Messungen aktivieren
`PING_INTERVAL`	0x11	W	2 Byte	Zeit zwischen den Messungen in *ms*
`READ_SENSOR_0`	0x20	R	1 Byte	Distanz von Sensor 0 in *cm*
`READ_SENSOR_1`	0x21	R	1 Byte	Distanz von Sensor 1 in *cm*
`READ_SENSOR_2`	0x22	R	1 Byte	Distanz von Sensor 2 in *cm*
`READ_SENSOR_3`	0x23	R	1 Byte	Distanz von Sensor 3 in *cm*
`READ_SENSOR_4`	0x24	R	1 Byte	Distanz von Sensor 4 in *cm*
`READ_SENSOR_5`	0x25	R	1 Byte	Distanz von Sensor 5 in *cm*
`READ_SENSOR_6`	0x26	R	1 Byte	Distanz von Sensor 6 in *cm*
`READ_SENSOR_7`	0x27	R	1 Byte	Distanz von Sensor 7 in *cm*
`READ_SENSORS_ALL`	0x30	R	8 Byte	Distanz von allen Sensoren in *cm*

Tabelle VII.1: I²C Register zur Ansteuerung der ATmega328P-Mikrocontroller

5.8 Auslesen der der Abstandsdaten am ESP32

Der ESP32 liest die Abstandsdaten über die zuvor beschriebenen Register aus. Nach dem Start des Systems werden die beiden ATmega328P-Slaves über den Reset-Ausgang akti-

viert (siehe Kapitel *Robotersystem* Abschnitt 6.2.1). Danach wird durch das `COMM_TEST`-Register überprüft, ob die Kommunikation mit den Slaves funktioniert. Dazu wird ein zufälliger Wert in das Register geschrieben und danach wieder gelesen. Stimmen diese überein, ist der Verbindungstest erfolgreich, andernfalls wird dieser bis zu zehnmal wiederholt, bevor mit einer Fehlermeldung abgebrochen wird.

Nach dem Verbindungsaufbau wird das `PING_INTERVAL` auf $1000ms$ gesetzt und die Messwertaufnahme mit dem `ENABLE`-Register gestartet.

Im zyklischen Teil werden alle 16-Ultraschallsensoren ausgewertet und in der `SharedData`-Struktur gespeichert. Diese Werte werden wie bereits beschrieben von der `SerialCommunication`-Klasse an die SPS übertragen.

6 Speicherprogrammierbare Steuerung

6.1 Übersicht

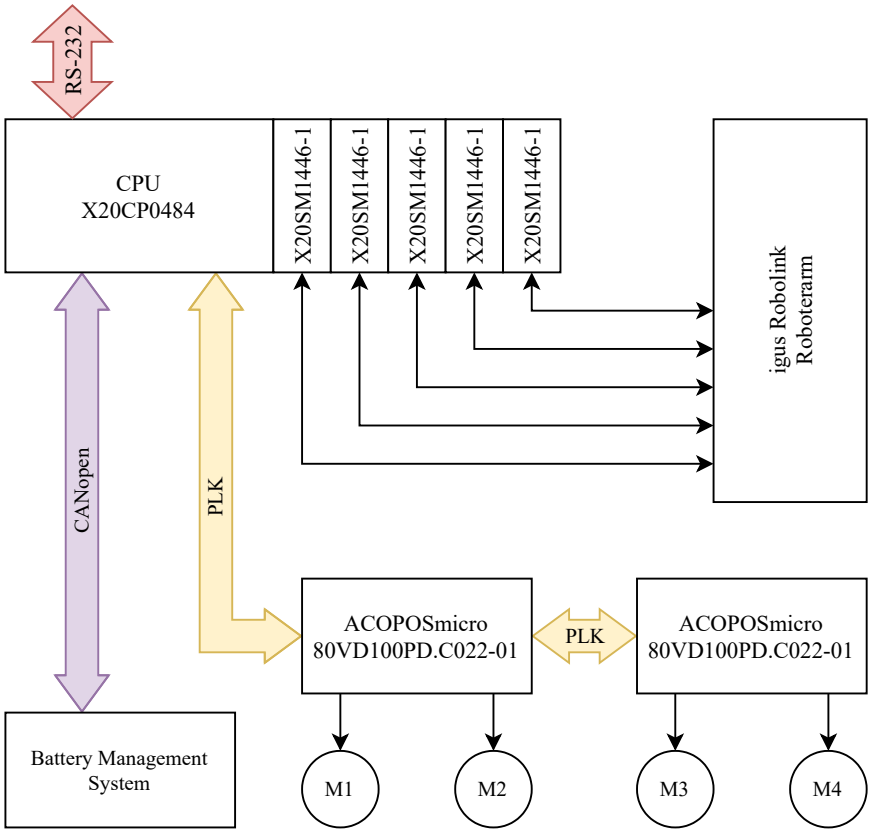


Abbildung VII.14: Topologie SPS-System am Roboter

Die SPS am Roboter ist über die RS232-Schnittstelle mit dem Receiver verbunden, von dem sie die Befehle des Fernsteuergeräts empfängt. Über Powerlink (PLK) die ACPOPOS-micro-Module an die SPS angeschlossen. Durch diese kann die SPS die Motoren des Fahrwerks ansteuern. Neben der CPU angereiht sind fünf Schrittmotormodule (X20SM1446-1), die jeweils eine der fünf Schrittmotorachsen des Roboterarms ansteuern, sowie die dazugehörigen Encoder auswerten.

Die diversen Teilaufgaben werden von verschiedenen Tasks übernommen:

- `Communication`: Kommunikation über die RS232-Schnittstelle
- `DriveCtrl`: Steuerung des Fahrwerks
- `RoboCtrl`: Steuerung des Roboterarms
- `BatteryCtrl`: Auswertung der BMS-Daten
- `Manager`: Koordinierung der einzelnen Tasks

Alle Tasks sind in Structured Text (ST) geschrieben, mit Ausnahme des `Communication`-Tasks, der in C++ implementiert wurde, um die bereits entwickelte Protokoll-Bibliothek nutzen zu können.

6.2 Datenaustausch zwischen den Tasks

Der Austausch von Daten und Befehlen zwischen den Tasks erfolgt mittels der globalen Strukturvariable `global` vom Typ `global_var_typ`. Diese besitzt mehrere Elemente, von denen manche selbst wieder Strukturvariablen sind, wodurch sich folgende Baumstruktur der globalen Variablen entsteht:

- `global`
  - `robo` ..... Globale Daten des Roboterarm-Tasks
    - `enable` ..... Starten der Motoren und des Homing-Ablaufs
    - `activate` ..... Aktiviert das Anfahren der Zielposition
    - `reset_error` ..... Quittieren des Fehlerzustandes
    - `confirm_home` ..... Manuelles Homing bestätigen
    - `state` ..... Derzeitiger Betriebszustand
    - `move` ..... Bewegungsbefehl (Zielkoordinaten)
  - `drive` ..... Globale Daten des Fahrwerk-Tasks
    - `enable` ..... Starten der Fahrwerk-Motoren
    - `reset_error` ..... Quittieren des Fehlerzustandes
    - `state` ..... Derzeitiger Betriebszustand
    - `move` ..... Bewegungsbefehl
      - `velocity` ..... Zielgeschwindigkeit
      - `duration` ..... Gültigkeitsdauer des Bewegungsbefehls
  - `battery` ..... Globale Daten des Batterieauswertungs-Tasks
    - `state_of_charge` ..... Akkustand in %
  - `comm` ..... Globale Daten des Kommunikations-Tasks
    - `enable` ..... Kommunikation aktivieren
    - `receiver_connected` ..... Verbindung zum Receiver aufrecht
    - `control_connected` ..... Verbindung zum Fernsteuergerät aufrecht

Die genauen Bedeutungen der Variablen werden dann in den Abschnitten erklärt, die sich mit dem jeweiligen Task befassen.

6.3 Kommunikation mit dem Receiver

Auch im Programm der SPS stellt die Kommunikation einen zentralen Bestandteil des Softwareprojekts dar. Übernommen wird diese Aufgabe vom Task `Communication`, der hier kurz beschrieben wird. Zuerst wird auf die Implementierung der Schnittstelle eingegangen, danach auf den darauf aufsetzenden Nachrichtenverkehr, welcher wiederum das in Abschnitt 3 beschriebene Übertragungsprotokoll verwendet.

6.3.1 Implementierung der Schnittstelle

Die Kommunikation über die RS232-Schnittstelle wurde mithilfe der Bibliothek `dvframe` implementiert. Diese stellt Funktionsblöcke zur Verfügung, mit denen die Schnittstelle angesprochen werden kann.

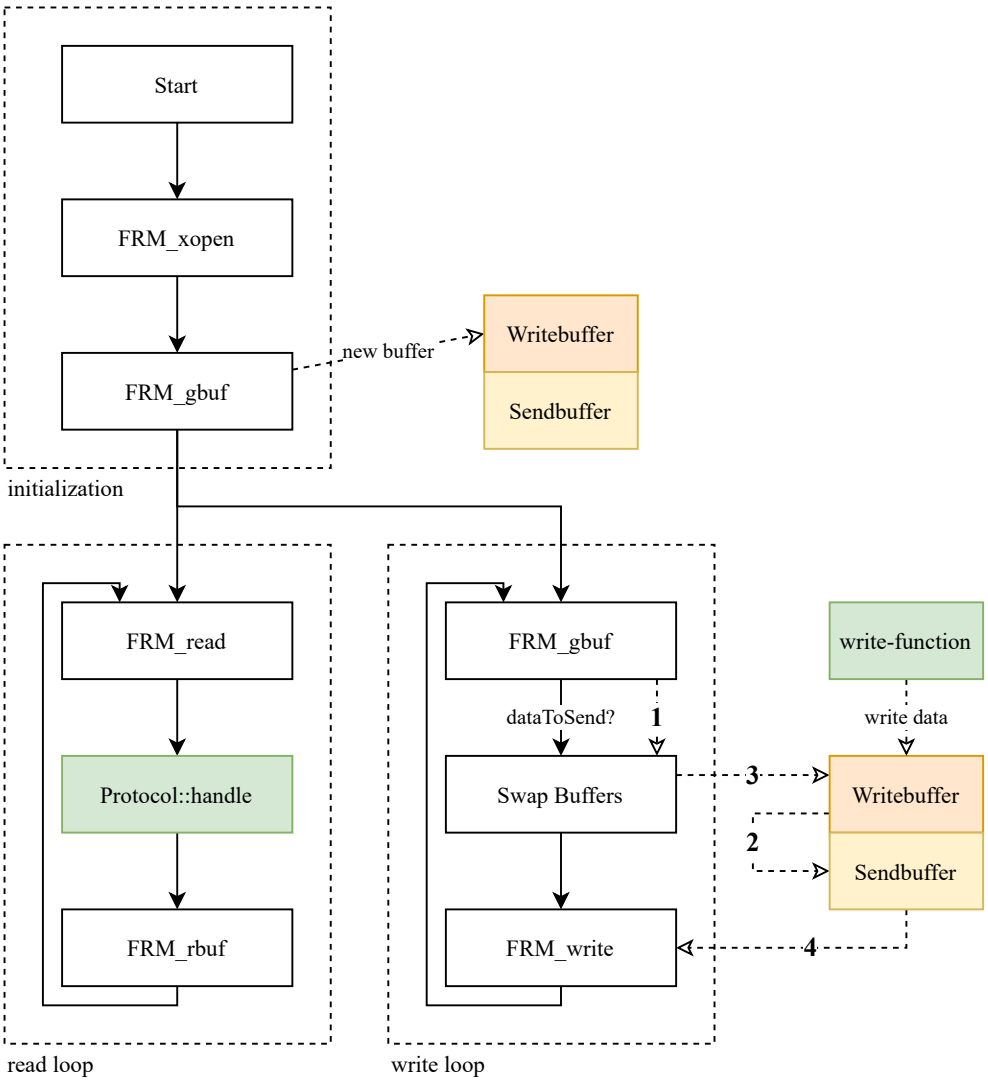


Abbildung VII.15: Flussdiagramm: Kommunikation über die serielle Schnittstelle der SPS

Abbildung VII.15 stellt dar, wie die Kommunikation über die Schnittstelle abläuft, wobei der Einfachheit halber auf die Darstellung der Fehlerbehandlung verzichtet wurde. Nach dem Start der SPS beginnt der Ablauf automatisch. Zunächst wird der Funktionsblock `FRM_xopen` aufgerufen, um die RS232-Schnittstelle zu initialisieren (dabei werden Para-

meter wie Schnittstellenadresse, Baudrate und Parity-Bits übergeben). Danach folgt der Aufruf des Funktionsblocks `FRM_gbuf`, womit ein Datenspeicher (engl. Buffer) angefordert wird, der später zum Senden der Daten verwendet wird. Ein Zeiger zu diesem Speicher wird auf der Variable „Writebuffer“ gespeichert, der Zweck dieses Zeigers wird später noch erläutert. Danach erfolgt das Lesen und Schreiben der Schnittstelle parallel.

In der Leseschleife wird `FRM_read` aufgerufen, wodurch die von der Schnittstelle empfangenen Daten ausgelesen werden. Diese werden dann mit der Methode `handle` an das `Protocol`-Objekt übergeben. Danach wird der Speicher der gelesen Daten mit dem Funktionsblock `FRM_rbuf` wieder freigegeben und der Ablauf beginnt von vorne.

Der „write loop“ (engl. Schreibschleife) ist komplexer aufgebaut und verwendet eine Technik ähnlich einem Ringpuffer mit nur zwei Speicherplätzen[8]:

- Dazu wird zunächst mit `FRM_gbuf` neben dem im Initialisierungsschritt angeforderten Datenspeicher ein weiterer angefordert (1).
- Dem `Protocol`-Objekt wurde mittels `setWriteFunction` die Funktion mitgeteilt, die zum Senden der Daten verwendet wird. Diese nützt unterdessen den Speicher, auf den der „Writebuffer“-Pointer zeigt, um die kodierten Daten der Nachrichten zu schreiben und setzt nach dem Schreibvorgang die Variable `dataToSend` auf `true`.
- Sobald das im „write loop“ erkannt wird, werden die Zeiger zu den Datenspeichern getauscht: Der Zeiger „Sendbuffer“ wird auf den Wert des „Writebuffer“-Zeigers gesetzt (2). Danach wird auf den „Writebuffer“-Zeiger die Adresse des in (1) angeforderten Datenspeichers gelegt (3). Somit zeigt „Sendbuffer“ jetzt auf den ehemaligen „Writebuffer“, während „Writebuffer“ auf einen frischen Buffer zeigt. Dieser Tausch findet in einem Zyklus statt, wodurch die „write-Function“ sofort wieder einen Datenspeicher zum Schreiben zur Verfügung hat.
- Nach dem Tausch der Datenspeicher können nun die Daten im „Sendbuffer“ nun über `FRM_write` gesendet werden (4), wodurch auch automatisch der vom „Sendbuffer“ verwendete Speicher freigegeben wird. Danach beginnt der Ablauf wieder von vorne.

Benötigt wird dieses Verfahren, da der Funktionsblock `FRM_write` mehrere Zyklen zur Ausführung benötigt, und trotzdem sichergestellt werden soll, dass in jedem Zyklus ein gültiger Speicherplatz für die ausgehenden Daten zur Verfügung steht.

6.3.2 Empfangene Nachrichten

Die SPS verarbeitet mehrere verschiedene Nachrichtentypen, die zur Steuerung der verschiedenen der SPS zugeordneten Komponenten des Robotersystems dienen.

`msg::enable_drive` Mit dieser Nachricht kann das Fahrwerk des Robotersystems ein- bzw. ausgeschaltet werden. Dazu setzt der `Communication`-Task den Wert der Variablen `global.drive.enable` auf den Paramter `enabled` der Nachricht (`true` oder `false`).

`msg::enable_arm` Diese Nachricht schaltet die Steuerung des Roboterarms ein- und aus. Analog zur vorigen Nachricht wird die Variable `global.robo.enable` auf den entsprechenden Wert gesetzt. Außerdem kann mit dem Parameter `try_auto_home` der Nachricht bestimmt werden, ob ein automatisches Homing (siehe Abschnitt 6.5.3) versucht werden soll.

[8] vgl. [43], S. 115ff.

`msg::confirm_home` Diese Nachricht dient dazu, das manuelle Homing des Roboterarms zu bestätigen, und gleichzeitig die Startrichtungen (siehe Abschnitt 6.5.3) der Achsen vorzugeben.

`msg::drive_velocity` Wenn diese Nachricht empfangen wird, wird das entsprechende Bewegungskommando über die Variable `global.drive.move.velocity` gesetzt. Die Gültigkeitsdauer des Bewegungskommandos – näheres dazu in Abschnitt 6.4.5 – wird nicht von der Nachricht festgelegt, sondern auf eine kurze, vorab definierte Zeit gesetzt. So soll sichergestellt werden, dass das Fahrwerk auch bei einem unerwarteten Kommunikationsabbruch oder Softwarefehler zum Stillstand kommt.

`msg::robot_move` Mit diesem Kommando kann die Zielposition des Roboterarms festgelegt werden. Diese wird aus der Nachricht in die Variable `global.robo.move` übertragen.

6.3.3 Gesendete Nachrichten

Um Informationen des Robotersystems an das Fernsteuergerät rückzuübertragen, sendet die SPS periodisch bestimmte Information aus.

`msg::robot_state` Diese Nachricht gibt die aktuellen Betriebszustände des Fahrwerks und des Roboterarms weiter.

`msg::battery_state` Durch diese Nachricht werden periodisch Informationen zum Zustand der Batterie übermittelt, unter anderem der Akkustand in Prozent und der aktuelle Lade- bzw. Entladestrom.

6.4 Fahrwerk-Steuerung

Als nächstes wird auf die Steuerung des Fahrwerkes eingegangen. Wie schon in Kapitel *Robotersystem* Abschnitt 4.3 erwähnt, basiert das Fahrwerk des Robotersystems auf einem Mecanum-Wheel-Antrieb.

Prinzipiell besteht die Fahrwerk-Steuerung aus drei Teilen:

- Ermittlung der Bewegungsgleichungen für den Mecanum-Wheel-Antrieb
- Konfiguration der Achsen und ACOPOSmicro-Servoinverter in Automation Studio
- Implementierung der Steuerung einer Achse als Funktionsblock `Wheel`
- Koordinierung der vier Achsen im Hauptprogramm (`DriveCtrl`), unter Zuhilfenahme der Bewegungsgleichungen des Mecanum-Wheel-Antriebs

6.4.1 Mathematischer Hintergrund

Zunächst mussten die erforderlichen Gleichungen ermittelt werden, die es ermöglichen, die benötigten Einzelgeschwindigkeiten der Räder für eine beliebige Linear- und Rotationsgeschwindigkeit des Robotersystems zu berechnen.

Dafür wird zunächst ein Koordinatensystem (siehe Abbildung VII.16) definiert, auf das sich die Positionen der Räder und die Geschwindigkeiten beziehen.

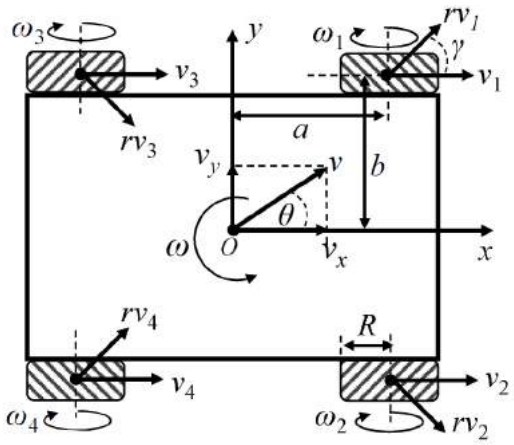


Abbildung VII.16: Schematische Darstellung des Mecanum-Wheel-Antriebs[41]

Nach [41] ergibt sich dann folgende Bewegungsgleichung:

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{bmatrix} = \frac{1}{R} * \begin{bmatrix}1 & -1 & -(a+b)\\ 1 & 1 & (a+b)\\ 1 & 1 & -(a+b)\\ 1 & -1 & (a+b)\end{bmatrix} * \begin{bmatrix}v_x\\ v_y\\ \omega\end{bmatrix} \tag{VII.2}$$

Gleichung VII.2 wurde dann in vier Einzelgleichungen umgeformt, was in den folgenden Gleichungen resultiert, wobei $v_\omega = \omega * (a+b)$ definiert wurde:

$$\omega_1 = v_x - v_y - v_\omega \tag{VII.3}$$
$$\omega_2 = v_x + v_y + v_\omega \tag{VII.4}$$
$$\omega_3 = v_x + v_y - v_\omega \tag{VII.5}$$
$$\omega_4 = v_x - v_y + v_\omega \tag{VII.6}$$

6.4.2 Konfiguration der Motoren

Die Inbetriebnahme und Konfiguration der Fahrwerkmotoren erfolgte über die von B&R Industrial Automation zur Verfügung gestellte Bibliothek „mapp Motion“. Diese übernimmt die Details der Ansteuerung und mindert den Programmieraufwand, da die meisten Einstellungen in Automation Studio konfiguriert werden können und nicht im Programm festgelegt werden müssen. Im Folgenden wird kurz auf die für die Inbetriebnahme der Motoren nötigen Schritte eingegangen.

Definition der Achsen Zu Beginn müssen die Achs-Objekte definiert werden. Diese beinhalten allgemeine Parameter der Achse und werden später mit dem ACOPOSmicro verknüpft. In den Parametern wird der Typ der Achse angegeben, sowie welche Grenzwerte die Achse aufweist. Abbildung VII.17 zeigt die Konfiguration der Achsen. Die Achse wird als periodisch (= keine Endanschläge) festgelegt und die später im Programm verwendete Maßeinheit auf Grad eingestellt. Um die Grenzwerte für jede Achse gleich zu definieren werden diese in einer externen Datei festgelegt (siehe Abbildung VII.18).

Name	Value	Unit	Description
gAxisD1			
Base type	Rotary periodic		Defines the basic movement possibilities of the axis
Measurement unit	Degrees		Measurement unit for the axis
Measurement res...	0.01	Measurement units	Possible resolution of measurement unit that can be achieved
Count direction	Standard		Direction of the axis in which the position value is increasing
Period settings			Possible position value range of a periodic axis
Period	360.0	Measurement units	The value range for axis positions is [0 , Period[
Movement limits	External		Various limit values that will be considered for axis movements
Limit reference	LimitSet_Drive		Name of the limit reference
Alarms	None		

Abbildung VII.17: Konfiguration der Achs-Objekte des Fahrwerks

Name	Value	Unit	Description
LimitSet_Drive			
Type	Rotary limit		Limit type
Position	Not used		Movement range of the axis via two position boundaries
Velocity	Basic		Velocity limits
Velocity	18000	Measurement units/s	Velocity limit in any movement direction
Acceleration	Basic		Acceleration limits
Acceleration	36000	Measurement units/s²	Acceleration limit in any movement direction
Deceleration	Basic		Deceleration limits
Deceleration	36000	Measurement units/s²	Deceleration limit in any movement direction
Jerk	Not used		Jerk limits
Torque	Not used		Torque limits

Abbildung VII.18: Grenzwerte der Fahrwerk-Achsen

Konfiguration der ACOPOSmicro-Module Nachdem die Achs-Objekte erstellt worden sind, können sie einem ACOPOSmicro zugewiesen werden. Dazu wird, wie in Abbildung VII.19 ersichtlich, das Achs-Objekt als „Axis reference“ eingetragen. In diesem Fall wurde die Fahrwerksachse 1 dem Kanal 1 auf dem ACOPOSmicro Modul 1 zugewiesen. Außerdem wird der digitale Eingang „Trigger 2“ (zuständig für die Not-Aus-Funktion) als invertierter Eingang festgelegt, da dieser als Öffner ausgeführt ist (siehe *Energieversorgung* Abschnitt 3.4). Alle anderen Parameter behalten ihre Standardwerte.

Channel 1			
Real axis			
Axis reference	gAxisD1		Name of the referenced axis component
Mechanical elements			Parameter of hardware elements situated between motor encoder and load which influence the scaling
Encoder link	One encoder		
Motor and position encoder	Encoder X6A		
Controller			Axis controller parameters
Homing			Homing mode and parameters which can be used within the application program as preconfigured setting
Stop reaction			Reactions of the axis in case of certain stop conditions
Movement error limits			Limit values that result in a stop reaction when exceeded
Position error	360	Measurement uni...	Lag error limit for stopping a movement
Velocity error monitoring	Not used		Velocity error monitoring mode
Jerk filter	Not used		Jerk filter
Digital inputs			Various digital input functionalities e.g. like homing switch or triggers
Homing switch			Homing switch input functionality
Positive limit switch			Positive limit switch input functionality
Negative limit switch			Negative limit switch input functionality
Trigger 1			Trigger 1 input functionality
Trigger 2			Trigger 2 input functionality
Level	Low		Level of the digital input hardware which leads to an active level of the functionality
Quickstop			Quickstop functionality
Simulation			Parameters which influence the simulation possibilities of this axis
Axis features			Features for an axis

Abbildung VII.19: Konfiguration des ACOPOSmicro

Damit sind die Motoren so weit konfiguriert, dass sie aus dem Programm gesteuert werden können. Das erfolgt über Funktionsblöcke aus der mapp Motion Bibliothek, deren Anwendung im nächsten Abschnitt erläutert wird.

6.4.3 Ansteuerung der Einzelachsen

Jede Achse des Antriebes muss prinzipiell den gleichen Ablauf und die gleichen Aufgaben ausführen: Die Achse muss initialisiert werden und einer zyklischen Geschwindigkeitsvorgabe folgen. Um diese Aufgaben von der Berechnung der Radgeschwindigkeit zu trennen, wurden sie in den Funktionsblock `Wheel` ausgelagert, der für die Ansteuerung einer einzelnen Achse zuständig ist.

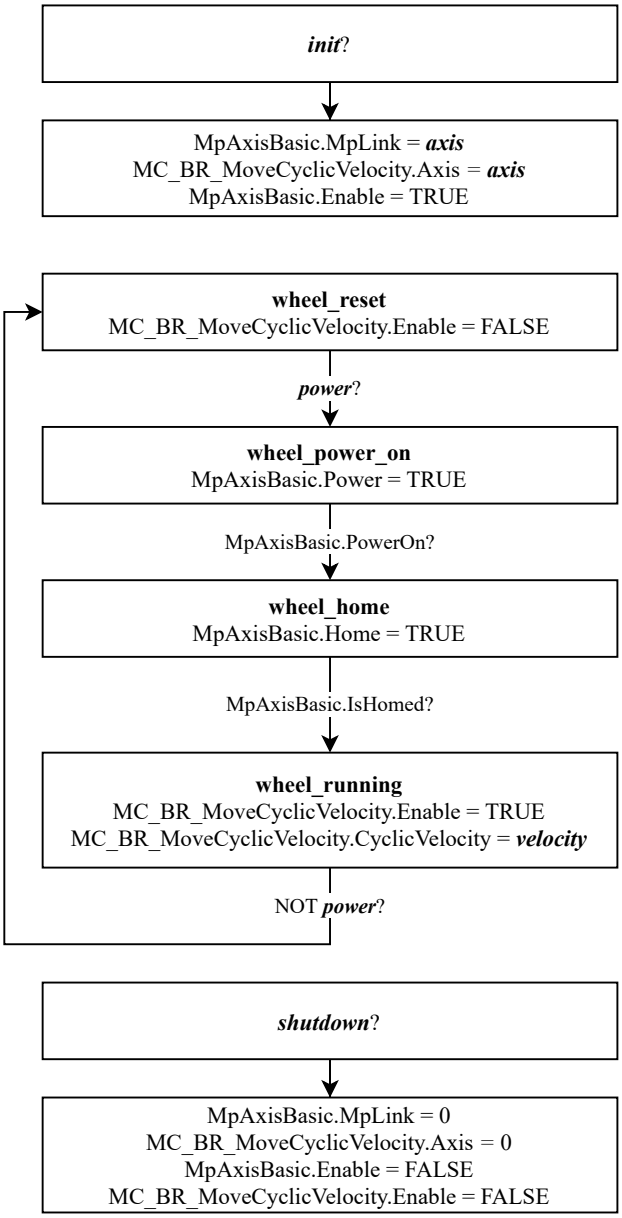


Abbildung VII.20: Flussdiagramm `Wheel`-Funktionsblock

Abbildung VII.20 zeigt den internen Ablauf des Funktionsblocks. Mit dem Eingang `init` werden die internen Funktionsblöcke – mit der Achse, die im Parameter `axis` angegeben wird – initialisiert. Zu Beginn befindet sich der Funktionsblock im Zustand `wheel_reset`. Sobald die Variable `power` gesetzt wird, wechselt der Funktionsblock zu `wheel_power_on`. Anschließend wird mit dem Funktionsblock `MpAxisBasic` die Achse initialisiert und ein direktes Homing[9] durchgeführt. Jetzt befindet sich der Funktionsblock im Zustand `wheel_running`. Hier wird der Funktionsblock `MC_BR_MoveCyclicVelocity` genutzt, um der Achse einen zyklischen Geschwindigkeitssollwert vorzugeben, der mit dem Eingangsparameter `velocity` angegeben wird.

[9]Direktes Homing bedeutet, dass die Achse keine Referenzposition hat, sondern einfach die derzeitige Position als Nullposition übernimmt.

6.4.4 Koordinierung der Achsen im Hauptprogramm

Der `Wheel`-Funktionsblock wird nun im `DriveCtrl`-Task für jede der vier Achsen einmal instanziiert. Die Zustandsmaschine des Tasks spiegelt die der einzelnen Achsen wider, weshalb nicht näher auf diese eingegangen wird. Mit der Variable `global.drive.enable` wird das gesamte Fahrwerk ein- und ausgeschaltet. Sind alle Achsen im Zustand `wheel_running`, kann die Berechnungen der Radgeschwindigkeiten nach Gleichung VII.3 bis VII.6 ausgeführt werden.

Die jeweiligen Winkelgeschwindigkeiten müssen nur noch von $\frac{rad}{s}$ in $\frac{^\circ}{s}$ umgerechnet und dann auf den `velocity`-Eingang des entsprechenden `Wheel`-Funktionsblockes kopiert werden.

Listing VII.16: Implementierung der Bewegungsgleichungen im Programm

```
1 tmp.lengths := params.mechanics.wheel_dist_front + params.mechanics.wheel_dist_side; //a
      + b
2 tmp.v_omega := tmp.local_velocity.a * tmp.lengths; //omega * (a + b)
3 tmp.inv_radius := 1.0 / params.mechanics.wheel_radius; // 1 / R
4
5 drive.wheels[0].velocity := RAD_TO_DEG * tmp.inv_radius * (+ tmp.local_velocity.x -
      tmp.local_velocity.y - tmp.v_omega);
6 drive.wheels[1].velocity := RAD_TO_DEG * tmp.inv_radius * (+ tmp.local_velocity.x +
      tmp.local_velocity.y + tmp.v_omega);
7 drive.wheels[2].velocity := RAD_TO_DEG * tmp.inv_radius * (+ tmp.local_velocity.x +
      tmp.local_velocity.y - tmp.v_omega);
8 drive.wheels[3].velocity := RAD_TO_DEG * tmp.inv_radius * (+ tmp.local_velocity.x -
      tmp.local_velocity.y + tmp.v_omega);
```

6.4.5 Ausführen der Bewegungsbefehle

Der letzte Teil, der für die Funktion des Fahrwerkes notwendig ist, ist die Verbindung zwischen den Bewegungsbefehlen und der vorhin beschrieben Antriebssteuerung. Die Bewegungskommandos werden über die globale Variable `global.drive.move` vom `Communication`-Task zum Fahrwerk-Task übergeben.

Wenn `global.drive.move.duration` einen Wert ungleich null aufweist, wird ein Bewegungskommando gestartet. Dazu wird der Wert von `global.drive.move.velocity` als Zielgeschwindigkeit übernommen. Gleichzeitig wird ein Timer mit einer Zeit von `global.drive.move.duration` gestartet, nach dessen Ablauf die Zielgeschwindigkeit wieder auf null gesetzt wird. Erhält der Task allerdings vor Ablauf des Timers einen neuen Bewegungsbefehl, wird der Timer einfach mit der neuen Zeit neugestartet. So wird erreicht, dass das Fahrwerk bei Ausbleiben von weiteren Bewegungsbefehlen automatisch zum Stillstand kommt.

Die oben angesprochene Zielgeschwindigkeit wird nicht direkt an den Algorithmus zur Berechnung der Radgeschwindigkeiten übergeben. Damit ein gleichmäßiges Beschleunigen der Räder gewährleistet ist, wird die maximale Beschleunigung schon vor der Umrechnung auf die Radgeschwindigkeiten begrenzt. Dazu speichert das Programm die derzeitige Geschwindigkeit mit und gleicht diese mit einer vorgegebenen Beschleunigung an die Zielgeschwindigkeit an.

6.4.6 Regler-Einstellung

Nachdem die Steuerung der Achsen vollständig ist, müssen nur noch die Regler der Achsen eingestellt werden. Dies geschieht über die Konfiguration der ACOPOSmicro-Module und ist notwendig, damit angeschlossenen Synchronmotoren optimal auf die tatsächliche Belastung angepasst sind.

Durchgeführt wurde das Einstellen der Regler mit „mapp Cockpit“, einem Softwarepaket für Steuerungen von B&R Industrial Automation, das eine Browser-basierte Benutzeroberfläche bereitstellt, die zur Wartung und Diagnose der Steuerung dient. Dieses Modul erkennt automatisch alle Konfigurierten Achsen und ermöglicht es, diese interaktiv zu kontrollieren.

Über die Oberfläche kann mit dem Befehl „Autotune“ eine automatische Ermittlung der Reglereinstellungen durchgeführt werden, die als Ausgangspunkt dienen können. Danach wurden die Reglerparameter experimentell verändert, um bessere Fahreigenschaften zu erzielen.

6.5 Steuerung des Roboterarms

Die zweite große Komponente des Robotersystems ist der Roboterarm, für dessen Steuerung der Task `RoboCtrl` zuständig ist. Wie bereits im Kapitel *Robotersystem* Abschnitt 5 erwähnt, wird der 5-Achs-Roboterarm „Robolink“ des Herstellers igus verwendet. Die Ansteuerung und Berechnung der Bewegungen übernimmt wie beim Fahrwerk die Bibliothek „mapp Motion“.

Im Folgenden wird zuerst die Konfiguration der Einzelachsen beschrieben, danach die Verknüpfung der Achsen zu einem 5-Achs-Roboter mit „mapp Motion“. Zum Schluss folgt die Beschreibung des Steuerungsprogramms, das die Bewegungsdaten vom Fernsteuergerät in Bewegungen des Roboterarms umsetzt.

6.5.1 Konfiguration der Achsen

Zunächst müssen – ähnlich wie beim Fahrwerk – die Einzelachsen konfiguriert werden. Es wurden fünf Achs-Objekte erstellt und jeweils Geschwindigkeits-, Beschleunigungs- und Winkel-Limits eingestellt. In folgender Grafik sind die Parameter für Achse 1 dargestellt:

gAxisQ1			
Base type	Rotary bounded		Defines the basic movement possibilities of the axis
Measurement unit	Degrees		Measurement unit for the axis
Measurement res...	0.01	Measurement units	Possible resolution of measurement unit that can be achieved
Count direction	Inverse		Direction of the axis in which the position value is increasing
Movement limits	Internal		Various limit values that will be considered for axis movements
Position			Movement range of the axis via two position boundaries
Lower limit	-90	Measurement units	Lower software limit position
Upper limit	90	Measurement units	Upper software limit position
Velocity	Basic		Limits for the velocity of the axis
Velocity	40	Measurement unit...	Velocity limit in any movement direction
Acceleration	Basic		Limits for the acceleration of the axis
Acceleration	200	Measurement unit...	Acceleration limit in any movement direction
Deceleration	Basic		Limits for the deceleration of the axis
Deceleration	200	Measurement unit...	Deceleration limit in any movement direction

Abbildung VII.21: Konfiguration von Achse 1 des Roboterarms

Anschließend wurden die Achs-Objekte mit den Schrittmotormodulen verknüpft. Das erfolgt ähnlich wie bereits in Abschnitt 6.4.2 für die ACOPOSmicro-Module beschrieben.

6.5.2 Konfiguration des 5-Achs-Roboterarms

Nachdem die Einzelachsen konfiguriert wurden, können sie nun miteinander verknüpft werden. Dazu wird zunächst die Mechanik definiert, wie in Abbildung VII.22 ersichtlich ist. Als Typ wird ein 5-Achs-Roboterarm ausgewählt und die Abmaße der einzelnen Achsen in der Konfiguration definiert.

Name	Value	Unit	Description
MechSys_5AxRobA			
Type	5-axes roboarm		Type of mechanical system
Subtype	5-axes roboarm (A)		Subtype of mechanical system
Description	Standard		Description of the mechanical system
Dimensions			Dimensions of the mechanical system
Translation from base to Q1			Translation from base to Q1
X	0	mea...	Translation in X direction
Y	0	mea...	Translation in Y direction
Z	140	mea...	Translation in Z direction
Translation from Q1 to Q2			Translation from Q1 to Q2
X	0	mea...	Translation in X direction
Y	0	mea...	Translation in Y direction
Z	146	mea...	Translation in Z direction
Translation from Q2 to Q3			Translation from Q2 to Q3
X	0	mea...	Translation in X direction
Y	0	mea...	Translation in Y direction
Z	355	mea...	Translation in Z direction
Translation from Q3 to Q4			Translation from Q3 to Q4
X	270	mea...	Translation in X direction
Y	0	mea...	Translation in Y direction
Z	0	mea...	Translation in Z direction
Translation from Q4 to Q5			Translation from Q4 to Q5
X	0	mea...	Translation in X direction
Y	0	mea...	Translation in Y direction
Z	-180	mea...	Translation in Z direction
Translation from Q5 to flange			Translation from Q5 to flange
X	0	mea...	Translation in X direction
Y	0	mea...	Translation in Y direction
Z	0	mea...	Translation in Z direction

Abbildung VII.22: Konfiguration der Roboterarm-Mechanik

Danach wird mit den vorhin erstellten Achsen eine Achsgruppe für Pfadberechnungen gebildet, welche die Achs-Objekte miteinander und mit dem mechanischen System verknüpft.

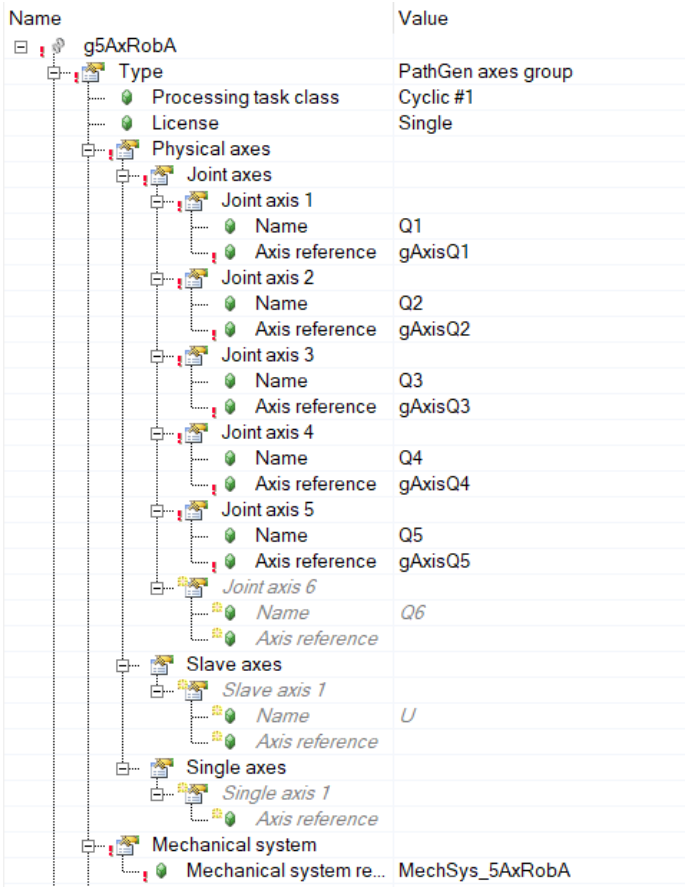


Abbildung VII.23: Erstellen der Achs-Gruppe

Damit ist der Roboterarm fertig konfiguriert und kann wie im nächsten Abschnitt beschrieben programmiert werden.

6.5.3 Steuerungsprogramm des Roboterarms

Die Steuerung des Roboterarms folgt dem in Abbildung VII.24 dargestellten Ablauf. Zu Beginn befindet sich der Roboterarm im Zustand `robot_reset`. Hier ist der Roboter ausgeschaltet und alle Funktionsblöcke sind deaktiviert. Mit dem Befehl `enable` wechselt der Roboter in den Zustand `robot_poweron`, in dem alle Achsen des Roboters aktiviert werden. Danach wird das Homing[10] durchgeführt.

Dazu gibt es zwei Variablen im permanenten Speicher der SPS: `robo_auto_home` (gibt an, ob ein automatisches Homing durchgeführt werden kann) und `robo_positions` (Beinhaltet immer die neueste Position der Roboterachsen). Abhängig von diesen Variablen wird jetzt entweder ein manuelles oder ein automatisches Homing durchgeführt.

Wenn `robo_auto_home TRUE` ist, referenziert der Roboter automatisch. Dazu verwendet das Programm den Homing-Modus `mcHOMING_DIRECT` und setzt dadurch die Ist-Postion der Achsen auf die Werte aus der Variablen `robo_positions`, die noch vom letzten Start im permanenten Speicher liegen.

[10]Homing (bzw. Referenzieren) einer Achse bedeutet, die Achse durch einen festgelegten Bewegungsablauf auf eine definierte Position zu bringen.

Andernfalls muss ein manuelles Referenzieren durchgeführt werden. Das ist notwendig, weil die Achsen keine Endschalter besitzen und der Homing-Schalter in der Mitte der Achse liegt, wodurch die SPS nicht hardwaremäßig erkennen kann, in welche Richtung die Referenzierbewegung starten muss. Deshalb muss vom Benutzer für jede Achse festgelegt werden, in welcher Richtung der Homing-Schalter liegt. Anschließend wird das Homing mit `home_directions_checked` bestätigt und der Roboterarm führt das Referenzieren mit dem Modus `mcHOMING_SWITCH_GATE` durch und fährt bis zum Referenzschalter.

Nach dem Homing fährt der Roboter im Schritt `robot_home_initial_position` in der Ausgangsposition. Ist das geschehen, wechselt er in den Ruhezustand (`robot_idle`). Wenn die Variable `follow` gesetzt wird, wechselt das Programm in den Zustand `robot_follow`, in dem zyklisch eine Position angefahren wird. Dieser Prozess wird im nächsten Abschnitt beschrieben. Durch Rücksetzen der Variablen `follow` begibt sich der Roboterarm wieder in den Ruhezustand.

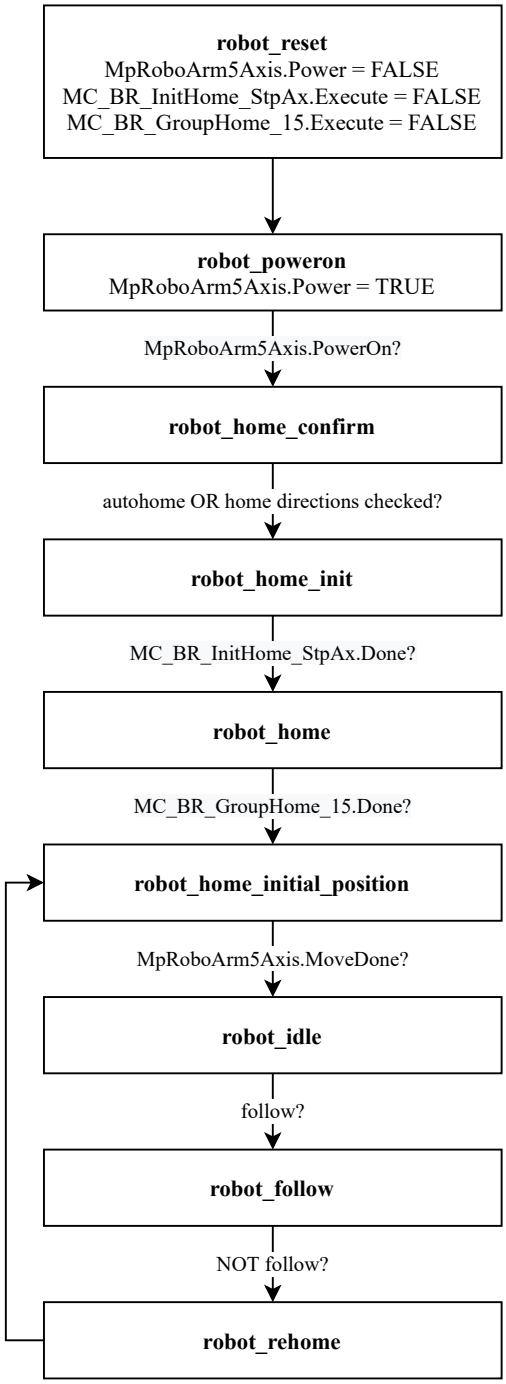


Abbildung VII.24: Ablaufdiagramm Roboter-Steuerung

6.5.4 Zyklisches Anfahren der Zielposition

Im Zustand `robot_follow` fährt der Roboter zyklisch eine Position relativ zur Ausgangsstellung an. Diese Position wird durch die Variable `global.robo.move` vorgegeben, die über den `Communication`-Task vom Fernsteuergerät empfangen wird.

Da die Zielposition einer normalen Positionierbewegung nicht geändert werden kann, ohne die vorhergehende Bewegung zu unterbrechen, wird die Position indirekt über „Jogging“ angefahren. Mit Jogging können die Achsen mit einer festgelegten Geschwindigkeit ver-

fahren werden, was normalerweise für den Handbetrieb verwendet wird. Es ist sowohl möglich, die Achsgeschwindigkeiten vorzugeben, als auch die Lineargeschwindigkeit des Tool-Center-Points[11] im Raum. Letzteres macht es möglich, mit Jogging einen einfachen Positionsregler zu realisieren, der versucht die Ist-Position (als Punkt im Raum) an die Soll-Position anzugleichen. So ist es möglich, einigermaßen flüssige Bewegungen zu realisieren.

Der Regler ist als einfacher P-Regler (mit Sättigung) ausgeführt, wodurch sich die in Abbildung VII.25 dargestellte Regelcharakteristik ergibt. Dabei ist v_{max} die maximale Geschwindigkeit des Roboterarms in einer Raumrichtung und d_{sat} die Positionsabweichung, ab der sich der Roboterarm mit maximaler Geschwindigkeit bewegt.

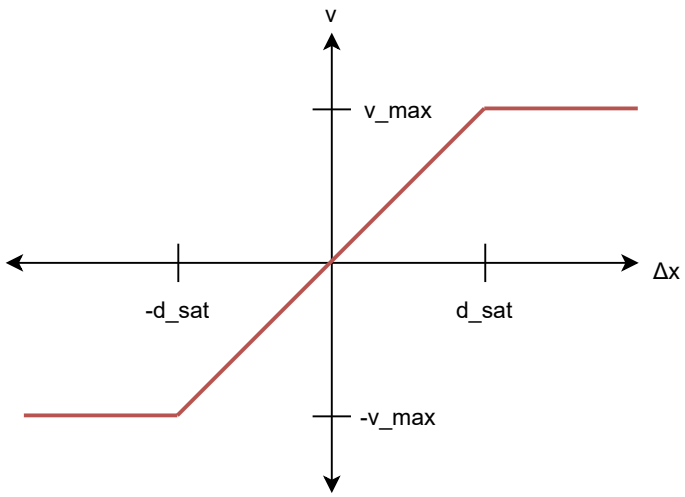


Abbildung VII.25: Regelcharakteristik des Positionsreglers für den Roboterarm

Damit ist die Steuerung des Roboterarms fertig implementiert und der Roboterarm kann einer zyklisch vorgegebenen Position folgen.

[11] Der Tool-Center-Point (TCP) ist der Punkt, an dem das Werkzeug am Roboter befestigt ist. In unserem Fall ist das die mechanische Hand.

6.6 Auswertung des Batteriemanagements

Die letzte Aufgabe, die noch von der SPS übernommen wird, ist die Auswertung des Batteriemanagementsystems. Das BMS ist über den CAN-Bus an der SPS angeschlossen (siehe Kapitel *Energieversorgung* Abschnitt 2.2.2). Da das Batteriemanagement über das CANopen-Protokoll[12] kommuniziert, können die Daten von der SPS eingelesen werden, als währen sie normale Eingänge der SPS. Dazu wird die vom Hersteller zur Verfügung gestellte „Electronic Datasheet"-Datei („.eds") genutzt, die beschreibt, welche Daten das BMS über CANopen zur Verfügung stellt.

Die „.eds"-Datei wird in Automation Studio importiert, wodurch das BMS als Modul eingefügt und mit der SPS verbunden werden kann. Das Resultat ist in Abbildung VII.26 sichtbar, der „CANopen Slave" stellt das BMS dar.

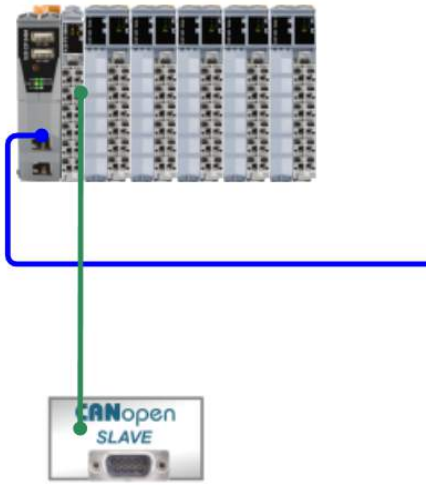


Abbildung VII.26: Einfügen des BMS als CANopen-Gerät in Automation Studio

Danach sind die Daten des BMS als Ein- und Ausgänge verfügbar und können mit Prozessvariablen verbunden werden.

GeneralAlarmSummary_2011sub1	USINT	General alarm summary
BatteryStatus_6000	USINT	Battery status
ChargerStatus_6001	USINT	Charger status
Temperature_6010	INT	Temperature
AhReturnedDuringLastCharge_6052	UINT	Ah returned during last charge
BatteryVoltage_6060	UDINT	Battery voltage
ChargeCurrentRequested_6070	UINT	Charge current requested
ChargerStateOfCharge_6080_In	USINT	Charger state of charge
BatteryStateOfCharge_6081	USINT	Battery state of charge

Abbildung VII.27: Verfügbare Datenpunkte des BMSmini

Der `BatteryCtrl`-Task liest die Daten ein und leitet sie über den `Communication`-Task an das Fernsteuergerät weiter, wo sie dann dem Benutzer angezeigt werden können.

[12] CANopen ist ein auf dem CAN-Bus aufbauendes Protokoll

VIII Ergebnisse und Ausblick

Zum Abgabezeitpunkt befindet sich das Robotersystem sowie das Fernsteuergerät in einem fertigen Prototypen-Status. Bewegungsdaten der menschlichen Hand werden korrekt interpretiert und an das Robotersystem gesendet, wo diese dann als Bewegungen des Fahrwerks oder des Greifsystems ausgeführt werden. Dazu kann zwischen den zwei Modi „Greifen“ und „Fahren“ gewählt werden. Die wesentlichen angestrebten Funktionen des haptischen Feedbacks sind einsatzbereit – einzig die Ausführung des kinästhetischen Feedbacks befindet sich in der Entwicklungsphase. Alle angesprochenen Referenzierungs- und Kalibrierungsabläufe sind voll automatisiert und bieten somit eine größtmögliche Benutzerfreundlichkeit.

Projektgruppen haben zukünftig die Möglichkeit, das Projekt in verschiedensten Aspekten weiterzuentwickeln. So würde es sich vor allem anbieten, diverse autonome Arbeitsabläufe zu integrieren. Auch die Finalisierung des kinästhetischen Feedbacks ist aufgrund des zeitlich hoch-intensiven Fertigungsverfahrens ausständig.

Ferner ist es durchaus überlegenswert, einen hochwertigeren bionischen Greifer zu installieren, da der hier verwendete nur sehr beschränkt tatsächliche Greifvorgänge durchführen kann. So könnten Gespräche bezüglich Sponsoring mit dem Unternehmen Schunk GmbH & Co. KG wiederaufgenommen werden, die während der Konzeptionierung des Projekts zu Jahresende 2019 scheiterten. Das Sortiment des besagten Unternehmens beinhaltet einen hoch-entwickelten – in einer Preisklasse von ca. 50.000 € – bionischen Greifer, welcher optimal für die angestrebten Funktionen geeignet wäre.

Sollte in Zukunft eine Funktionstüchtigkeit im Bereich der Telepräsenz (siehe *Stand der Technik* Abschnitt 4.3) angestrebt werden, also ein Bedienen des Robotersystems ohne sich unmittelbar am selben Ort zu befinden, können problemlos einige Änderungen vorgenommen bzw. das System erweitert werden. Die Installation einer oder mehrerer Kameras und eine Änderung der Kommunikation von Bluetooth auf ein System, das über weitere Distanzen zuverlässig funktioniert, wäre dafür ein weitgreifender Schritt. Eine überaus vielversprechende Möglichkeit zur Umsetzung besagter Kommunikation stellt das neue 5G-Netz dar, da dieses trotz großer Distanzen weiterhin kurze Latenzzeiten aufweist und eine Bedienung über das Internet möglich machen würde.

Anhang A

Code

Listing A.1: protocol/shared_types.h

```
#ifndef SHARED_TYPES_H
#define SHARED_TYPES_H

#include "inttypes.h"

enum class FingerType: uint8_t {
    Pinky, Ring, Middle, Index, Thumb, COUNT
};

enum class VibrateMode: uint8_t {
    None, Cancel, Short, Long, Double, NUM_MODES
};

enum class DriveState : uint8_t {
    Off, On, Error
};

enum class RobotArmState : uint8_t {
    Off, HomeDirectionConfirmation, Homing, On, Error
};

#endif
```

Listing A.2: protocol/messages.h

```
#ifndef MESSAGES_H
#define MESSAGES_H

#include "inttypes.h"
#include "shared_types.h"

enum class msg: uint8_t {
  //Lifecycle messages
  init = 0x01,
  init_ack,
  heartbeat,
  connection_lost,

  //Motion message
  imu_update = 0x10,
  drive_velocity,
  robot_move,
  finger_update,

  //Feedback messages
  brake_update = 0x20,
  vibro_trigger,

  //Information messages
  battery_state = 0x30,
  robot_state,
  distance_readings,

  //Control Configuration messages
  config_sensor = 0x40,
  config_data_mode,

  //Robot Setup messages
  enable_drive = 0x50,
  enable_arm,
  confirm_home
};

template<msg msg_code>
struct msg_data {
  static const msg MSG_CODE = msg_code;
};

//Lifecycle messages
struct __packed msg_data_init_t : msg_data<msg::init> {
  uint8_t version_code;
};

struct __packed msg_data_init_ack_t : msg_data<msg::init_ack> {
  bool success = false;
};

struct __packed msg_data_heartbeat_t : msg_data<msg::heartbeat> {
  bool link_up; //Used by receiver to indicate whether the other link is up (PLC or
      control)
};

struct __packed msg_data_connection_lost_t : msg_data<msg::connection_lost> {

};
```

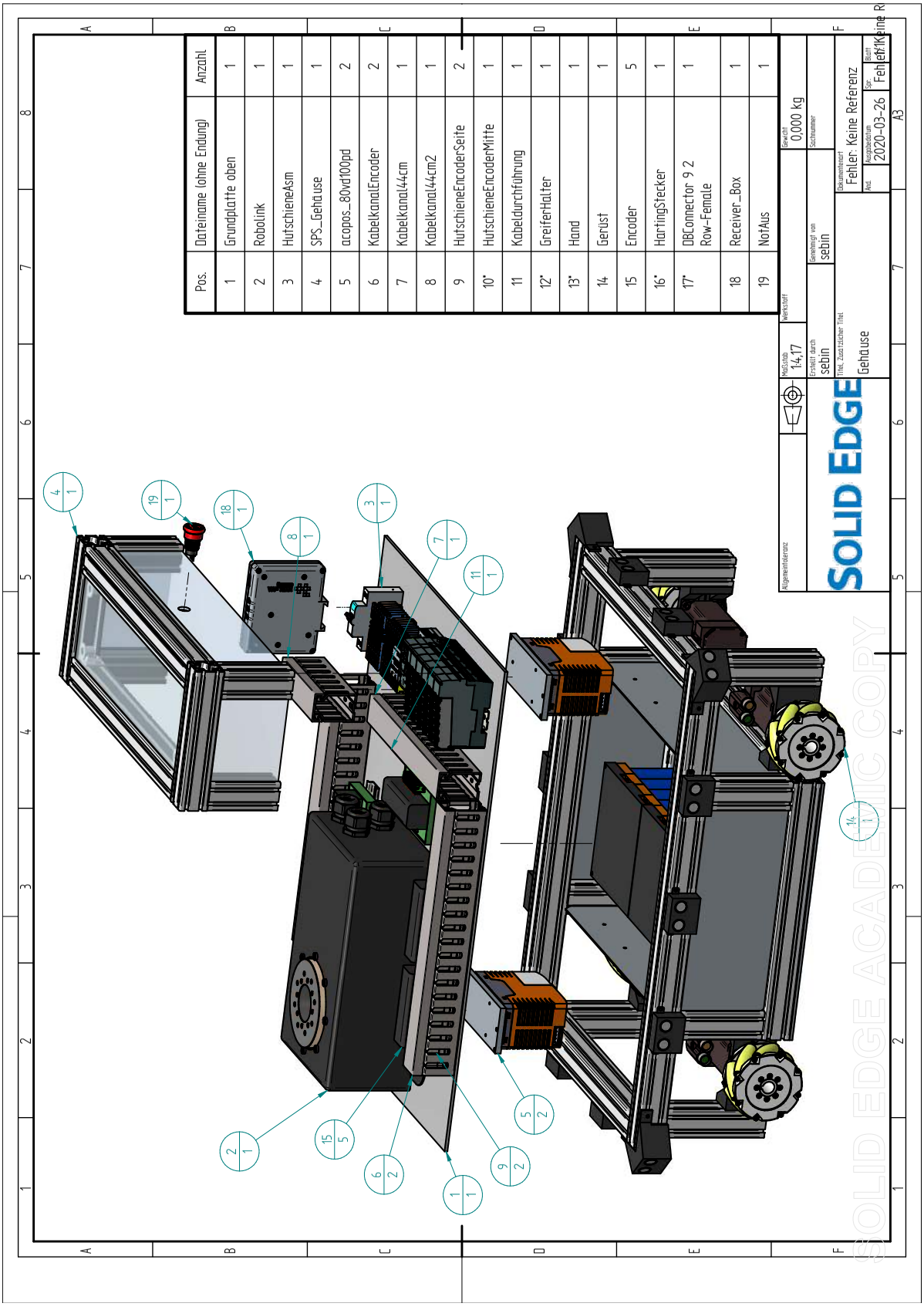
```
60
61 //Motion messages
62 struct __packed msg_data_imu_update_t : msg_data<msg::imu_update> {
63   int16_t orientation_x;
64   int16_t orientation_y;
65   int16_t orientation_z;
66   int16_t orientation_w;
67   uint8_t cal_sys, cal_gyro, cal_accel, cal_mag;
68   int16_t accel_x, accel_y, accel_z;
69 };
70
71 struct __packed msg_data_drive_velocity_t : msg_data<msg::drive_velocity> {
72   int8_t velocity_x; //velocity from -100% to 100%
73   int8_t velocity_y; //velocity from -100% to 100%
74   int16_t angle;    //angle in degrees
75 };
76
77 struct __packed msg_data_robot_move_t : msg_data<msg::robot_move> {
78   int8_t position_x; //position from -100% to 100%
79   int8_t position_y; //position from -100% to 100%
80   int8_t position_z; //position from -100% to 100%
81 };
82
83 struct __packed msg_data_finger_update_t : msg_data<msg::finger_update> {
84   uint8_t finger_levels[5];
85 };
86
87 //Feedback messages
88 struct __packed msg_data_brake_update_t : msg_data<msg::brake_update> {
89   uint8_t brake_levels[5];
90 };
91
92 struct __packed msg_data_vibro_trigger_t : msg_data<msg::vibro_trigger> {
93   uint8_t mask;
94   VibrateMode mode;
95 };
96
97 //Information messages
98 struct __packed msg_data_battery_state_t : msg_data<msg::battery_state> {
99   bool charging;
100  uint8_t level;
101 };
102
103 struct __packed msg_data_robot_state_t : msg_data<msg::robot_state> {
104   DriveState drive;
105   RobotArmState arm;
106 };
107
108 struct __packed msg_data_distance_readings_t : msg_data<msg::distance_readings> {
109   uint8_t distances[16];
110 };
111
112 //Control Configuration Messages
113 #define MSG_CONFIG_SENSOR_TYPE_MASK 0xF0
114 #define MSG_CONFIG_SENSOR_LIMIT_MASK 0x0F
115 #define CONFIG_SENSOR_BEND 0x10
116 #define CONFIG_SENSOR_FORCE 0x20
117 #define CONFIG_SENSOR_LOW 0x01
118 #define CONFIG_SENSOR_HIGH 0x02
119 #define CONFIG_SENSOR_INDEX_ALL 0xFF
120 struct __packed msg_data_config_sensor_t : msg_data<msg::config_sensor> {
121   uint8_t config_type;
```

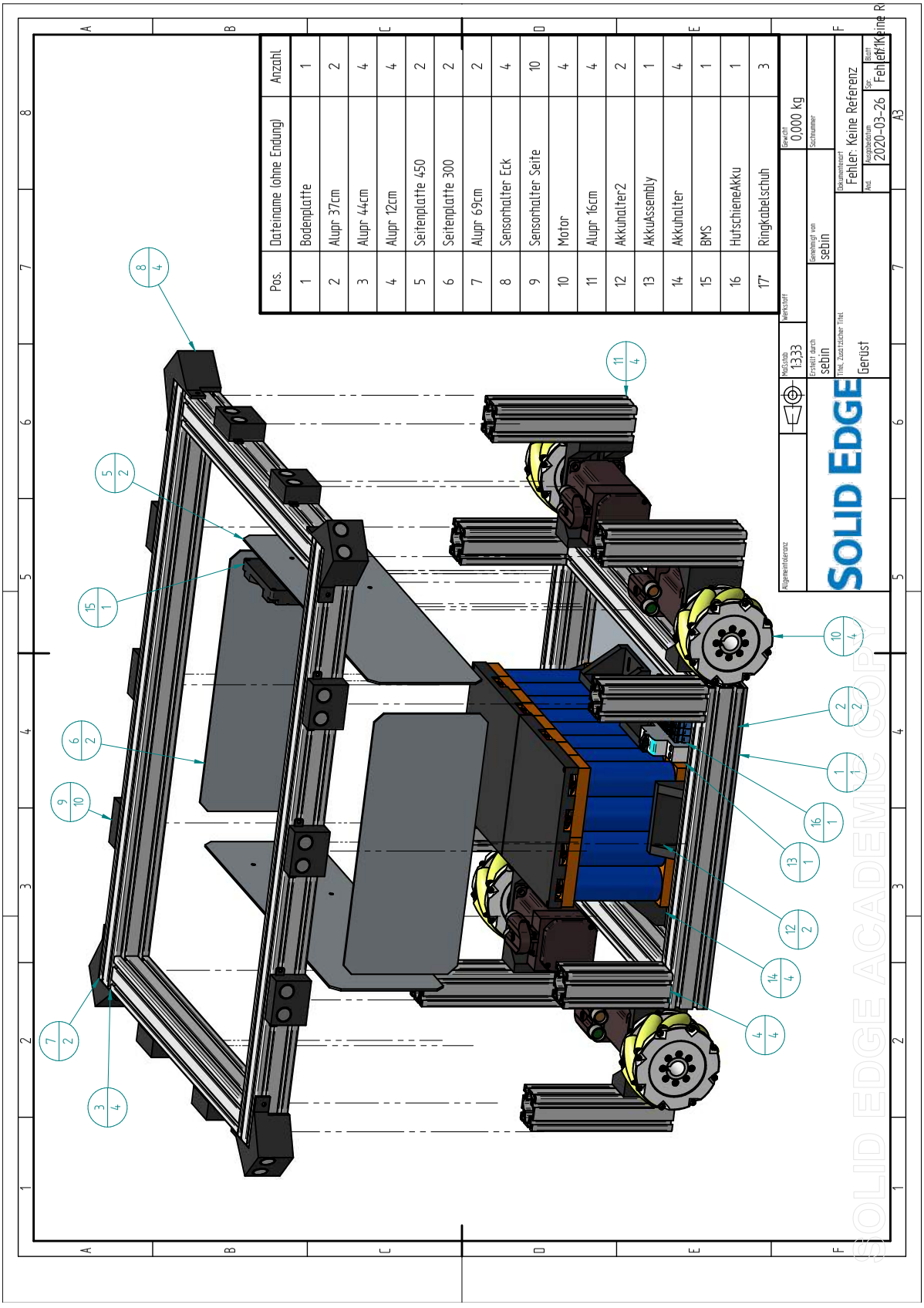
```
122   uint8_t sensor_index;
123 };
124
125 struct __packed msg_data_config_data_mode_t : msg_data<msg::config_data_mode> {
126   bool send_imu_data = false;
127 };
128
129 //Robot Setup Messages
130 struct __packed msg_data_enable_drive_t : msg_data<msg::enable_drive> {
131   bool enabled;
132 };
133
134 struct __packed msg_data_enable_arm_t : msg_data<msg::enable_arm> {
135   bool enabled;
136   bool try_auto_home;
137 };
138
139 struct __packed msg_data_confirm_home_t : msg_data<msg::confirm_home> {
140   bool home_flags[5];
141 };
142
143 #endif
```

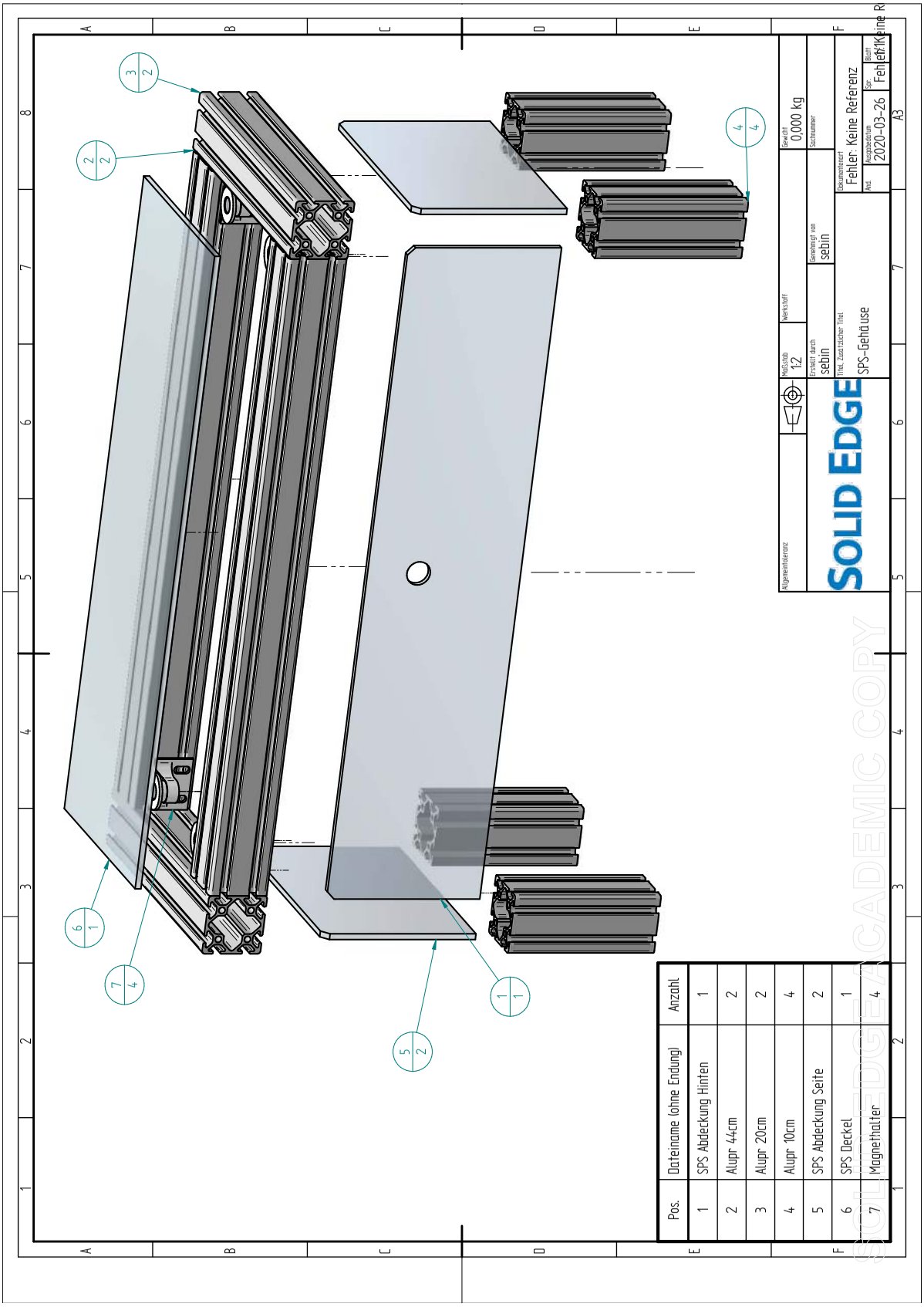
Anhang B

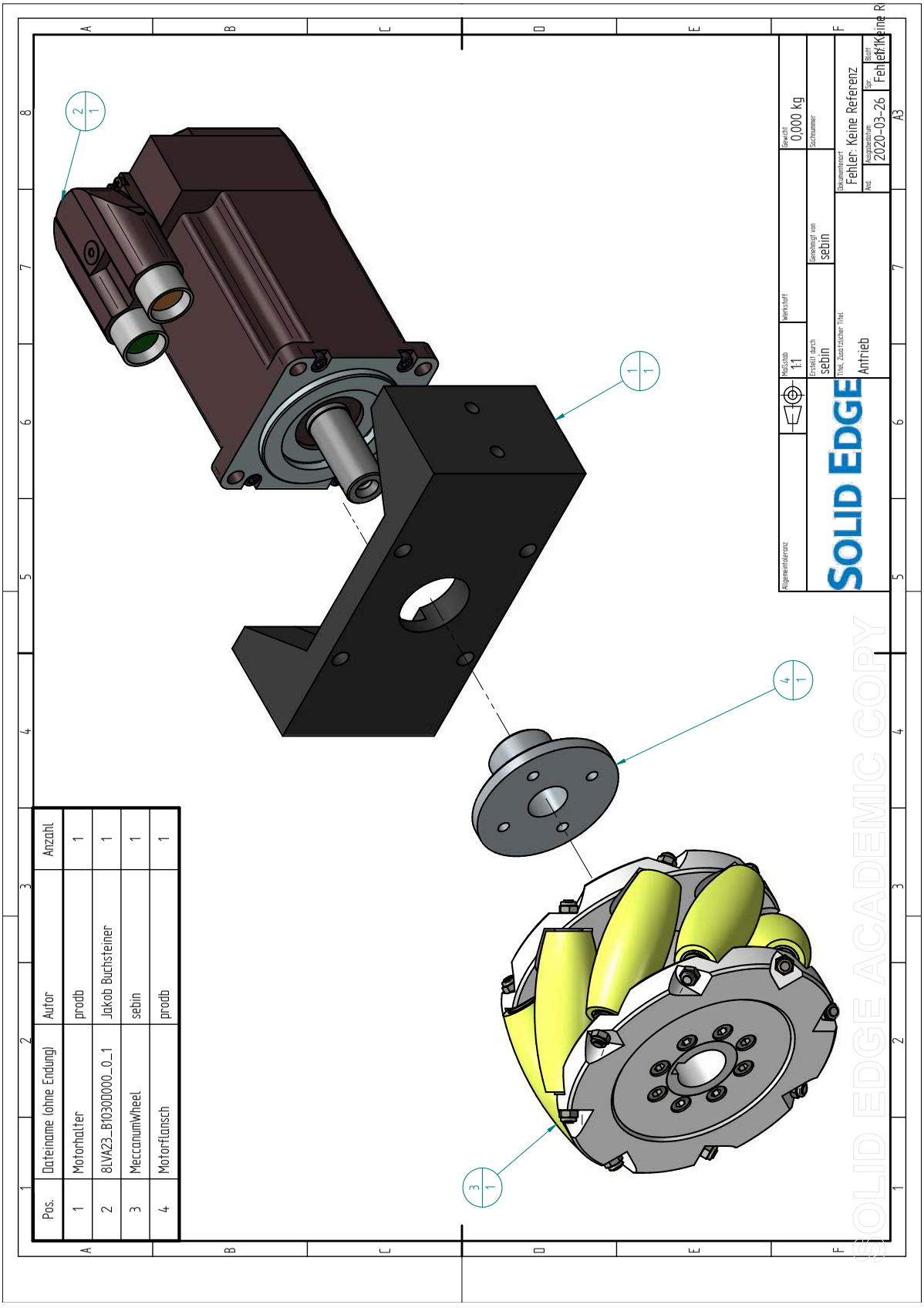
CAD-Zeichnungen

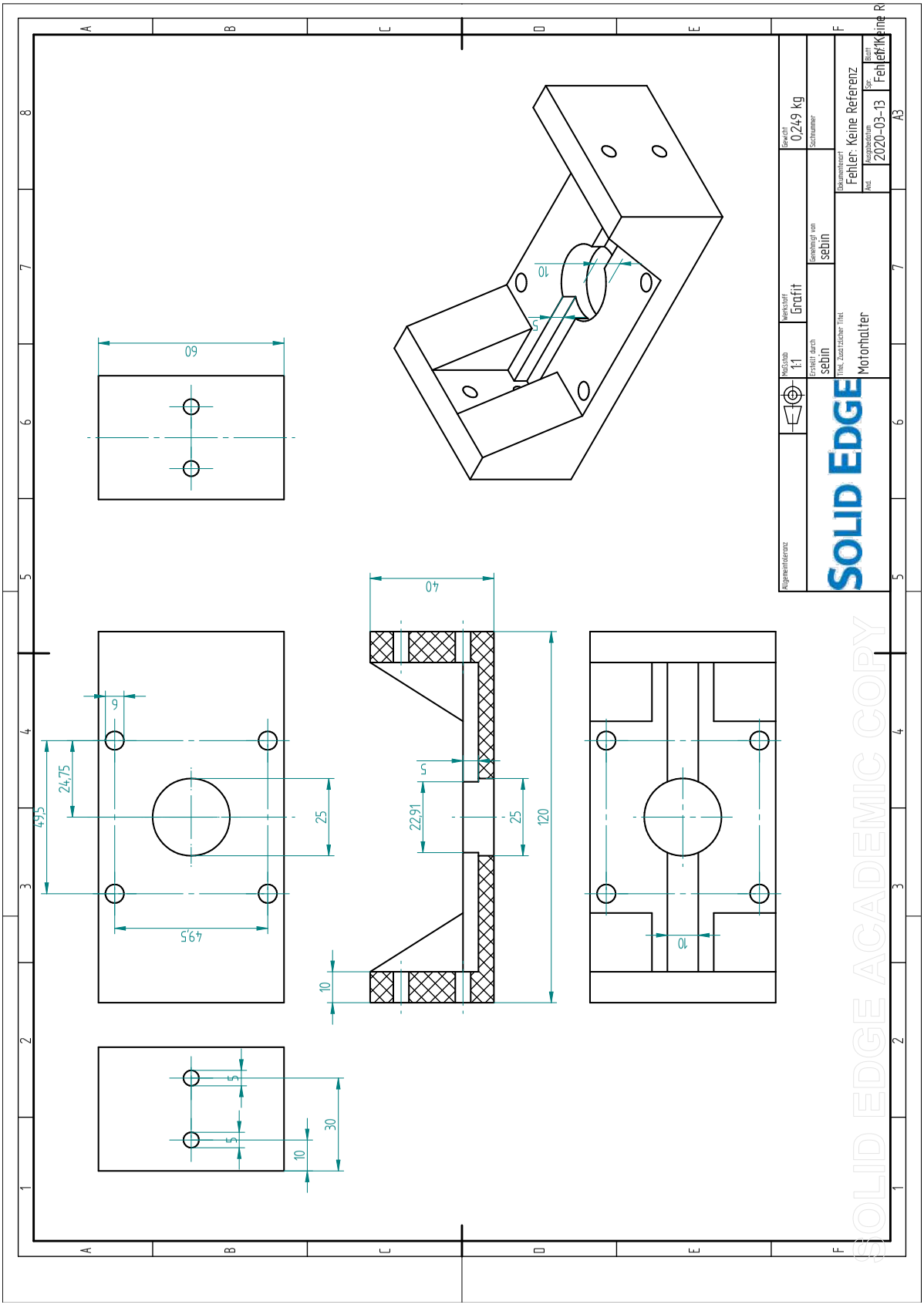
1 Robotersystem

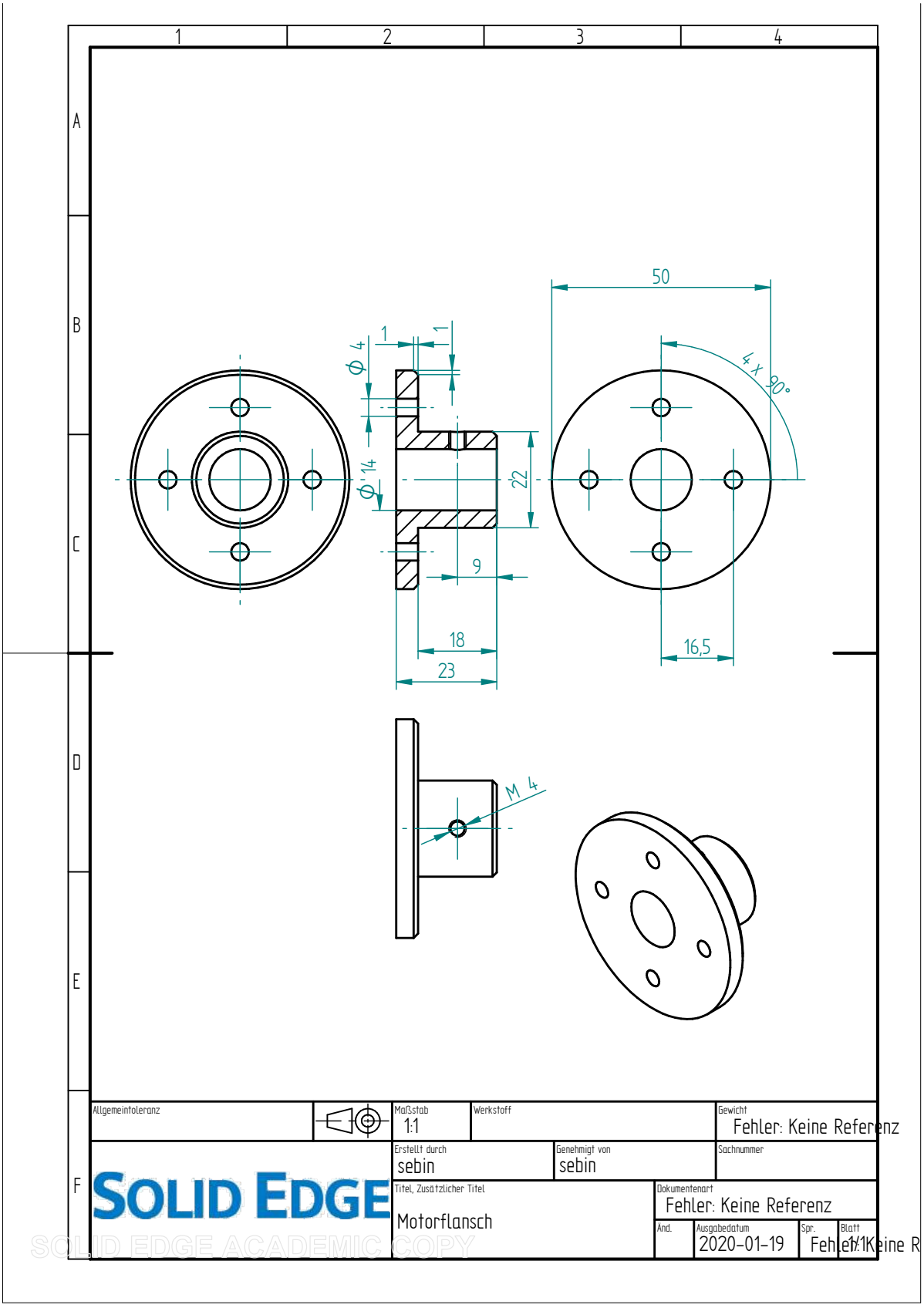


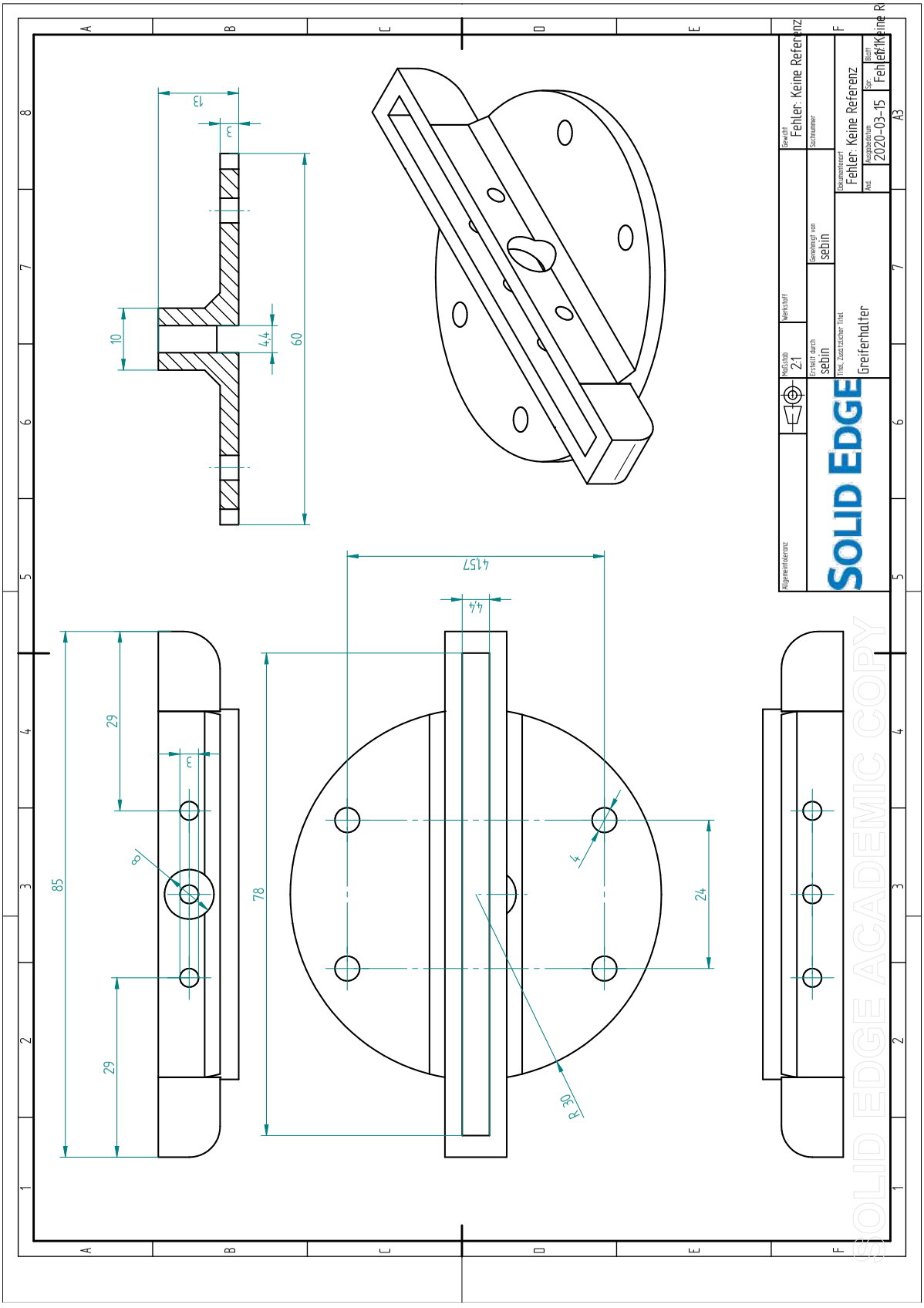

Pos. Dateiname (ohne Endung) Anzahl
1 Bodenplatte 1
2 Alupr 37cm 2
3 Alupr 44cm 4
4 Alupr 12cm 4
5 Seitenplatte 450 2
6 Seitenplatte 300 2
7 Alupr 69cm 2
8 Sensorhalter Eck 4
9 Sensorhalter Seite 10
10 Motor 4
11 Alupr 16cm 4
12 Akkuhalter2 2
13 AkkuAssembly 1
14 Akkuhalter 4
15 BMS 1
16 HutschieneAkku 1
17* Ringkabelschuh 3
Gewicht 0,000 kg
Maßstab 1333
Erstellt durch sebin
Titel, Zusätzlicher Titel
Gerüst
Dokumententyp Fehler: Keine Referenz
Ausgabedatum 2020-03-26
Blatt Fehler: Keine Ref
SOLID EDGE
SOLID EDGE ACADEMIC COPY

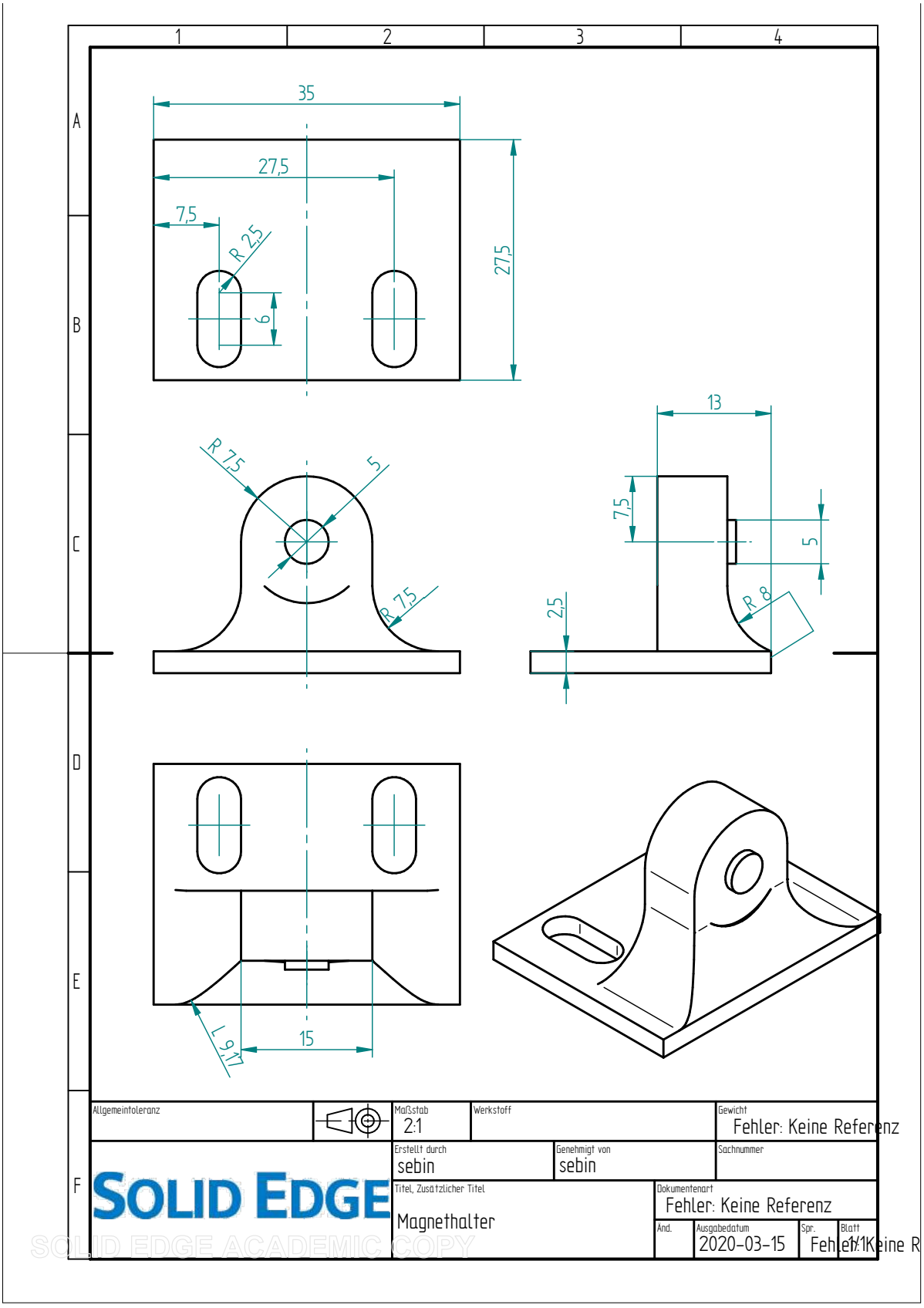

SOLID EDGE
Fehler: Keine Referenz
SPS-Gehäuse
sebin
0,000 kg
2020-03-26
Fehler!Keine Referenz
Pos. | Dateiname (ohne Endung) | Anzahl
1 | SPS Abdeckung Hinten | 1
2 | Alupr 44cm | 2
3 | Alupr 20cm | 2
4 | Alupr 10cm | 4
5 | SPS Abdeckung Seite | 2
6 | SPS Deckel | 1
7 | Magnethalter | 4
SOLID EDGE ACADEMIC COPY

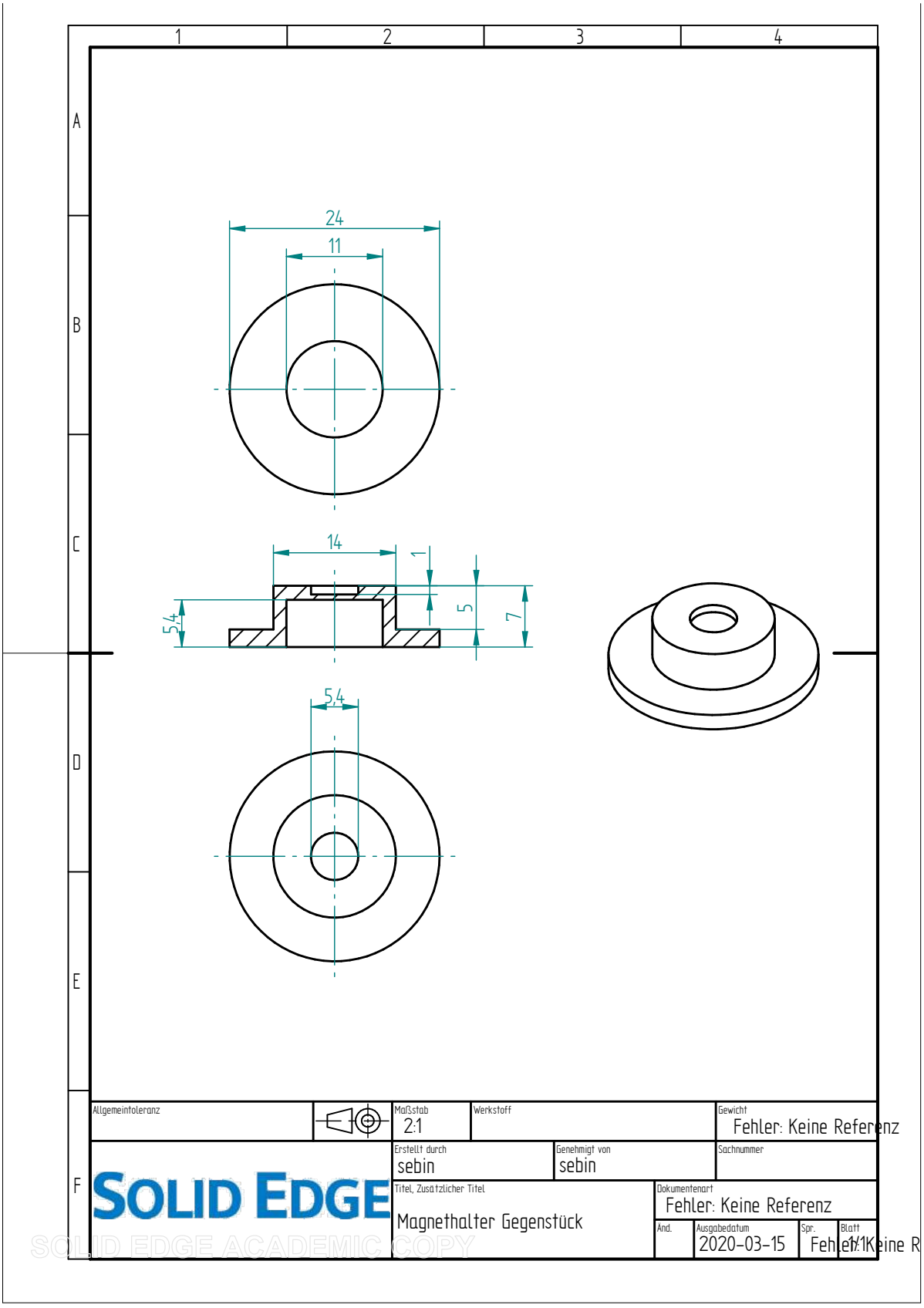

Pos. | Dateiname (ohne Endung) | Autor | Anzahl
1 | Motorhalter | prodb | 1
2 | 8LVA23_B1030D000_0_1 | Jakob Buchsteiner | 1
3 | MecanumWheel | sebin | 1
4 | Motorflansch | prodb | 1
Gewicht 0,000 kg
Sachnummer
Maßstab 1:1
Werkstoff
Erstellt durch sebin
Geändert durch sebin
Titel, Zusätzlicher Titel
Antrieb
Dokumententyp Fehler: Keine Referenz
Ausgabedatum 2020-03-26
Blatt 1/1
SOLID EDGE
SOLID EDGE ACADEMIC COPY

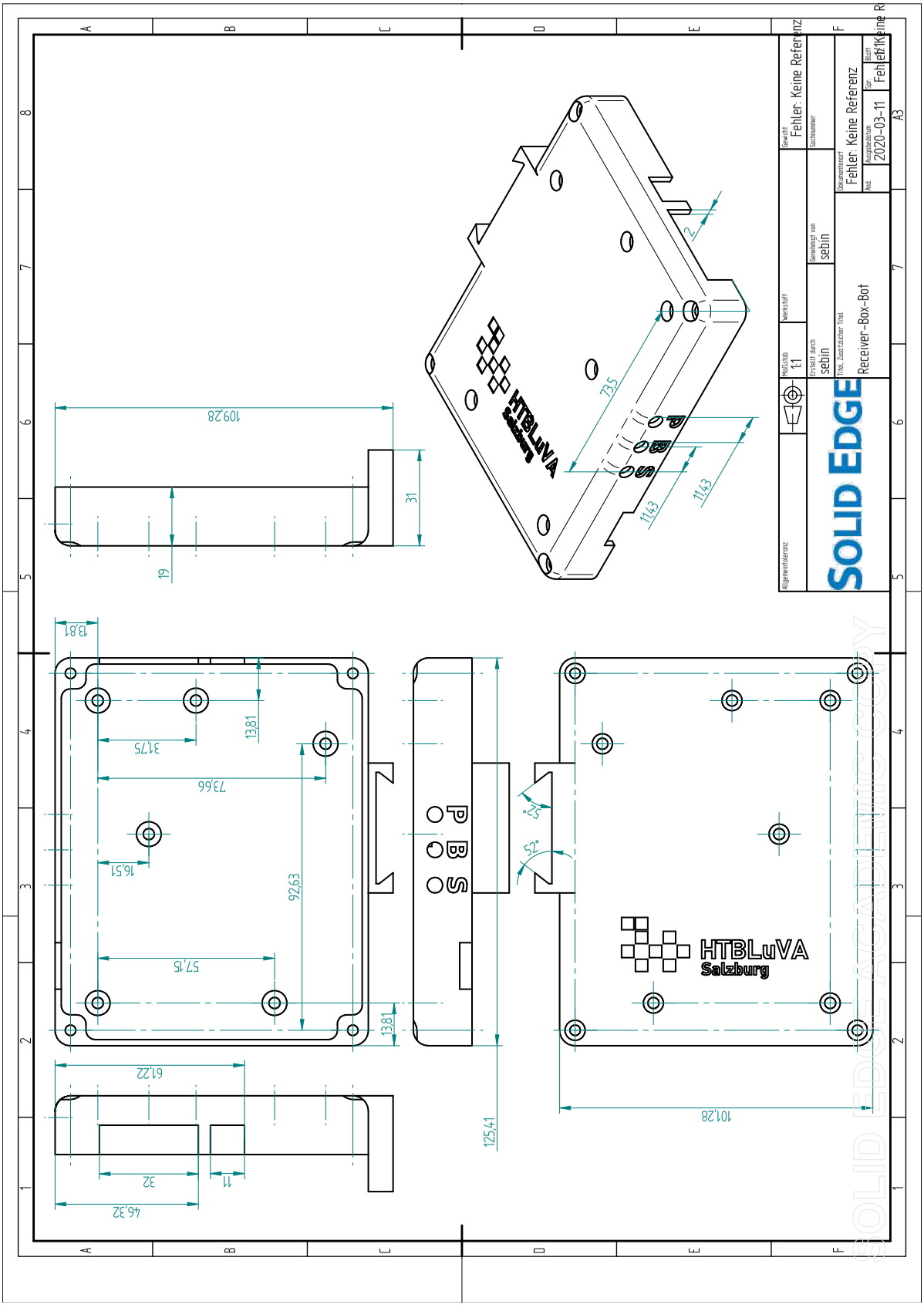

Gewicht 0,249 kg
Sachnummer
Dokumenteigenschaft
Fehler: Keine Referenz
Ausgabedatum 2020-03-13
Blatt
Fehler: Keine Ri
Maßstab 1:1
Werkstoff Grafit
Gezeichnet von sebin
Erstellt durch sebin
Titel, Zusätzlicher Titel
Motorhalter
Allgemeintoleranz
SOLID EDGE
A3
60
40
120
25
24,75
49,5
22,91
30
SOLID EDGE ACADEMIC COPY

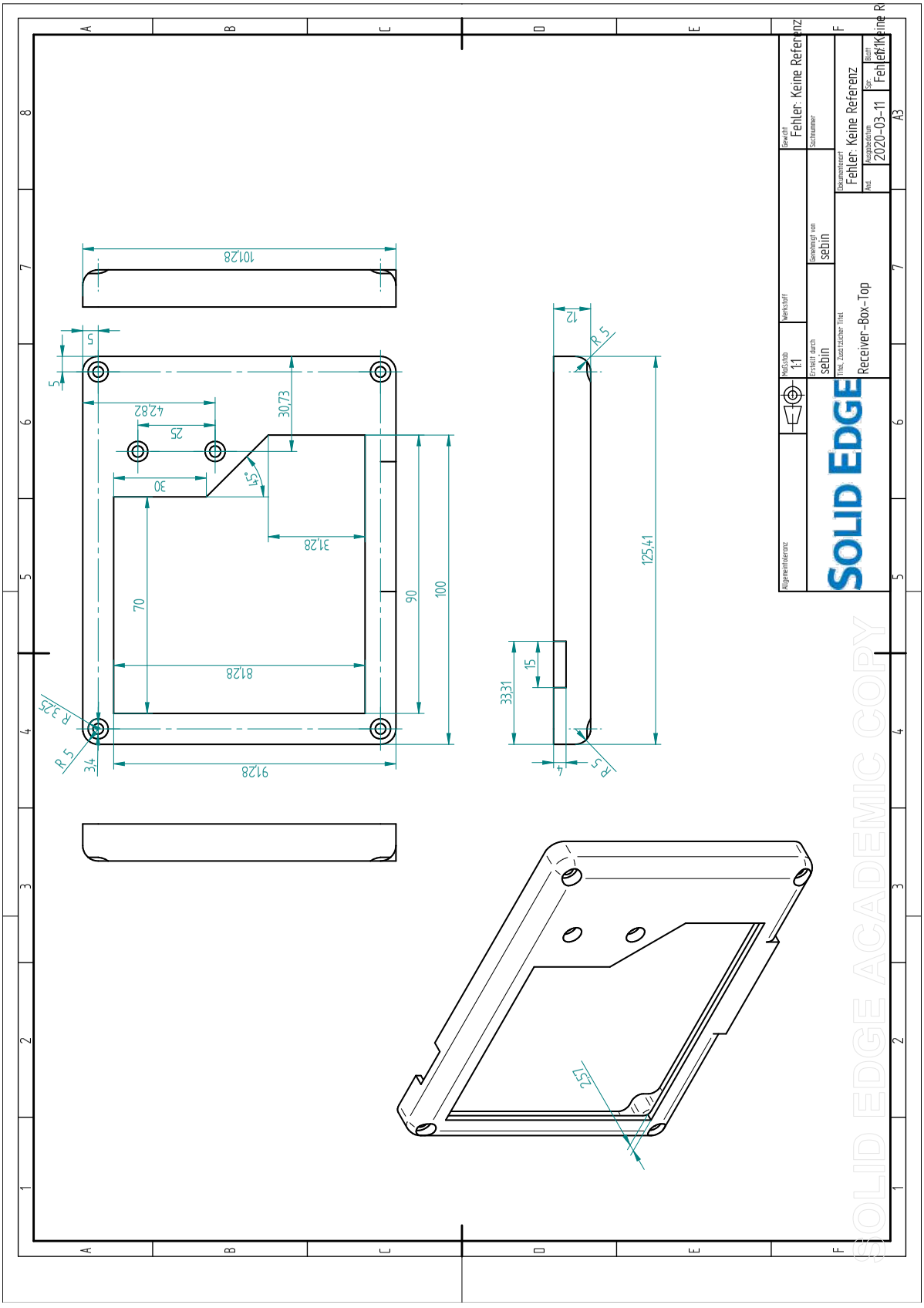

1
2
3
4
A
B
C
D
E
F
50
1
1
Ø 4
Ø 14
22
9
18
23
4 x 90°
16,5
M 4
Allgemeintoleranz
Maßstab
1:1
Werkstoff
Gewicht
Fehler: Keine Referenz
Erstellt durch
sebin
Genehmigt von
sebin
Sachnummer
Titel, Zusätzlicher Titel
Motorflansch
Dokumentenart
Fehler: Keine Referenz
And.
Ausgabedatum
2020-01-19
Spr.
Blatt
1/1
SOLID EDGE
SOLID EDGE ACADEMIC COPY


13
3
10
44
60
41,57
4,4
29
3
8
85
78
4
24
R30
29
Fehler: Keine Referenz
Fehler: Keine Referenz
2020-03-15
2:1
sebin
sebin
Greiferhalter
SOLID EDGE
SOLID EDGE ACADEMIC COPY

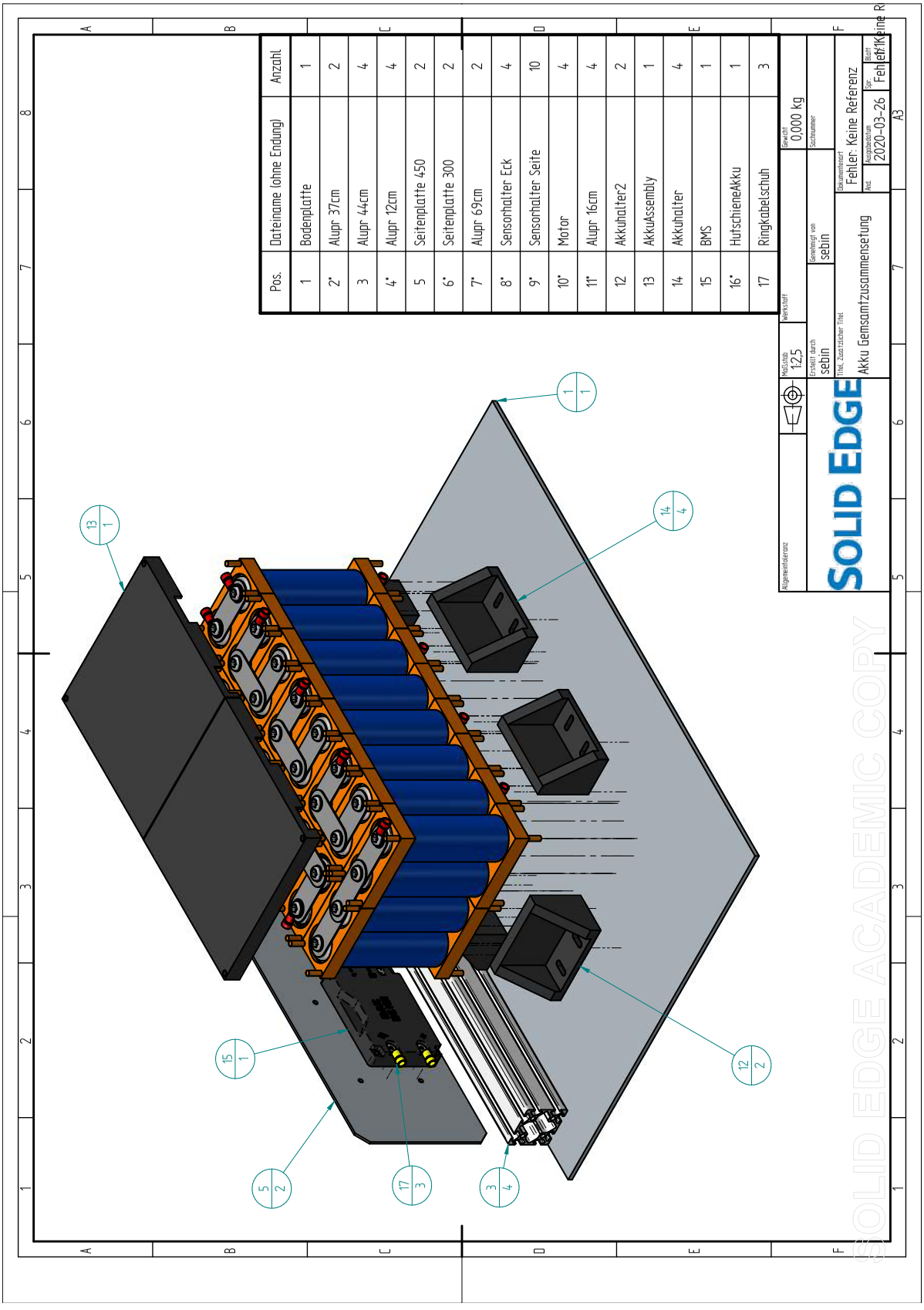

1
2
3
4
A
B
C
D
E
F
35
27,5
7,5
R 2,5
6
27,5
R 7,5
5
R 7,5
13
7,5
5
2,5
R 8
15
9,17
Allgemeintoleranz
Maßstab
2:1
Werkstoff
Gewicht
Fehler: Keine Referenz
Erstellt durch
sebin
Genehmigt von
sebin
Sachnummer
SOLID EDGE
Titel, Zusätzlicher Titel
Magnethalter
Dokumentenart
Fehler: Keine Referenz
And.
Ausgabedatum
2020-03-15
Spr.
Blatt
1/1
SOLID EDGE ACADEMIC COPY

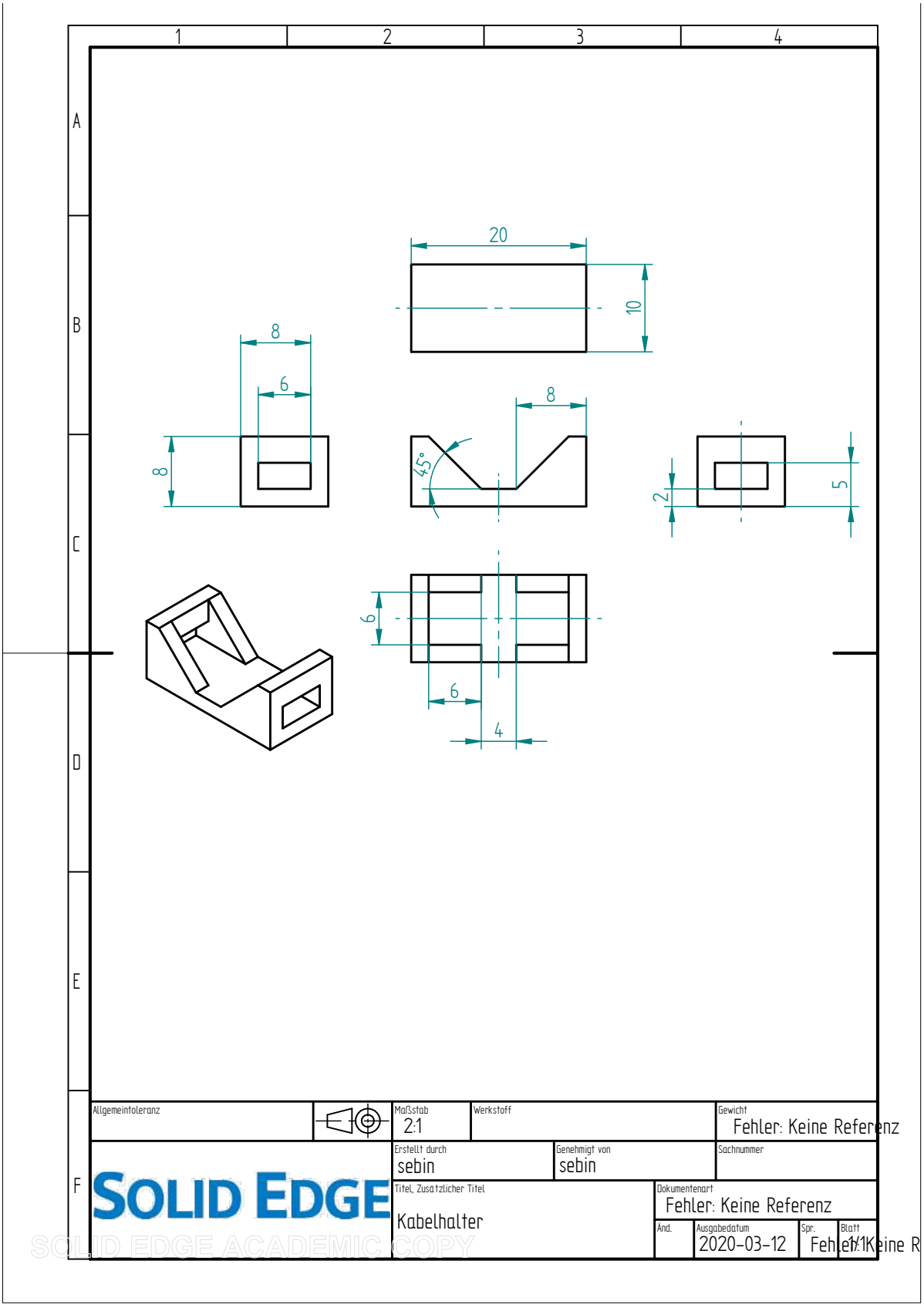

1
2
3
4
A
B
C
D
E
F
24
11
14
1
5
7
5,4
5,4
Allgemeintoleranz
Maßstab
2:1
Werkstoff
Gewicht
Fehler: Keine Referenz
Erstellt durch
sebin
Genehmigt von
sebin
Sachnummer
SOLID EDGE
Titel, Zusätzlicher Titel
Magnethalter Gegenstück
Dokumentenart
Fehler: Keine Referenz
And.
Ausgabedatum
2020-03-15
Spr.
Blatt
1:1
SOLID EDGE ACADEMIC COPY

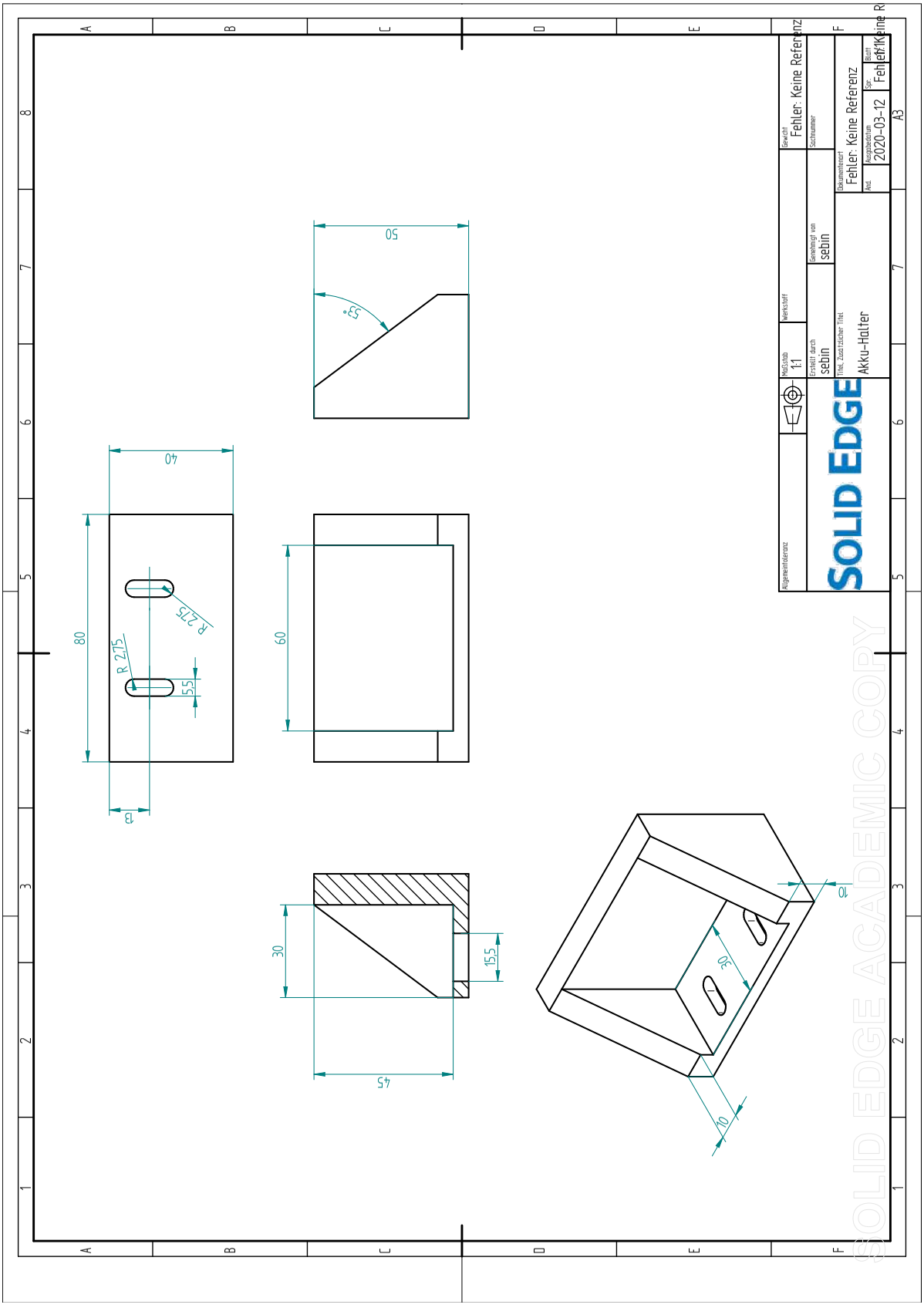

Fehler: Keine Referenz
Fehler: Keine Referenz
2020-03-11
sebin
sebin
1:1
Receiver-Box-Bot
SOLID EDGE
109,28
31
19
73,5
114,3
114,3
2
13,81
31,75
73,66
16,51
92,63
57,15
13,81
61,22
32
11
46,32
125,41
101,28
52°
52°
HTBLuVA
Salzburg

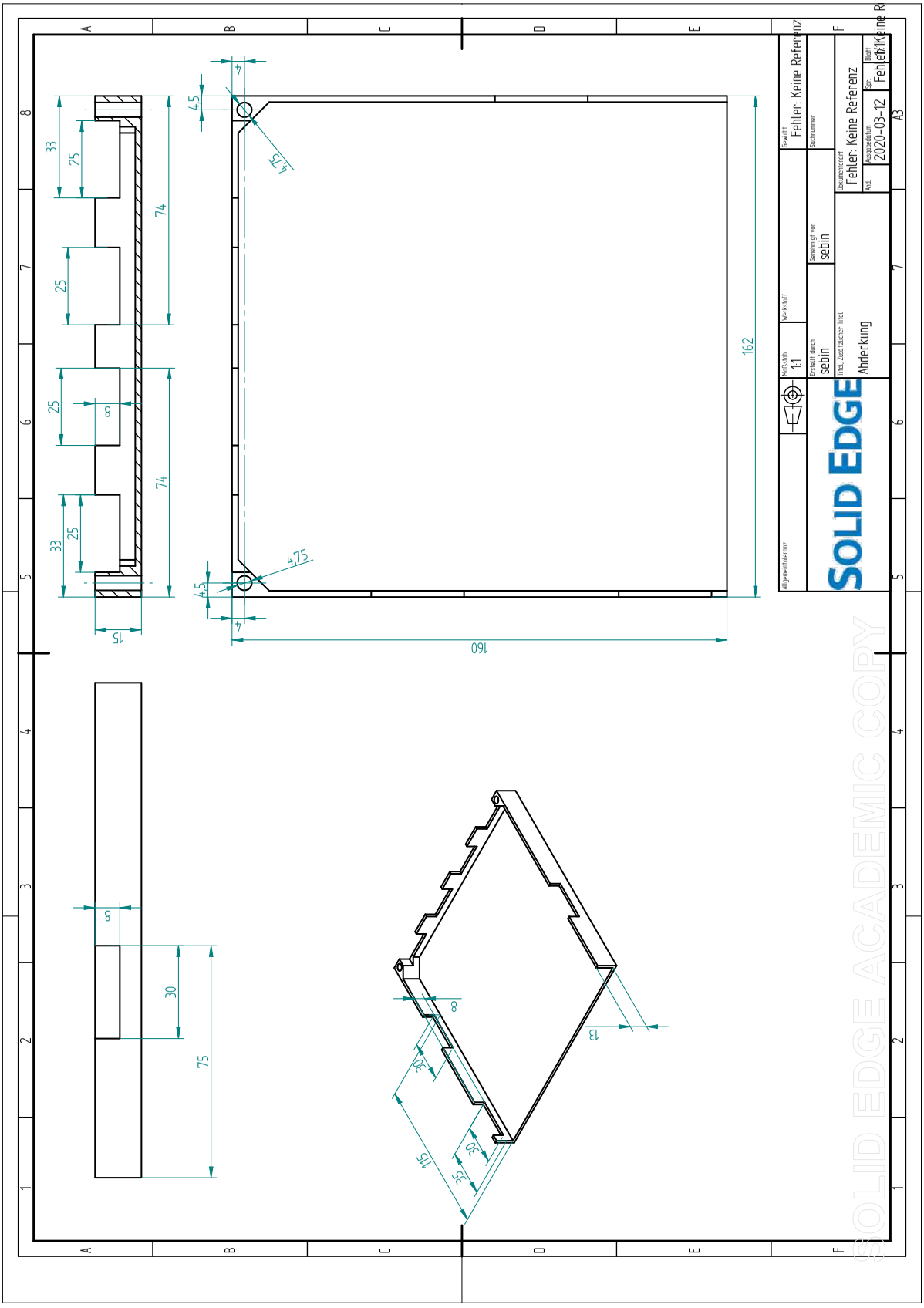

101,28
5
5
42,82
25
30,73
45°
30
31,28
70
81,28
90
100
91,28
R 3,25
R 5
34
12
R 5
125,41
33,31
15
4
R 5
25,57
Fehler: Keine Referenz
Fehler: Keine Referenz
Receiver-Box-Top
sebin
sebin
2020-03-11
1:1
SOLID EDGE
SOLID EDGE ACADEMIC COPY

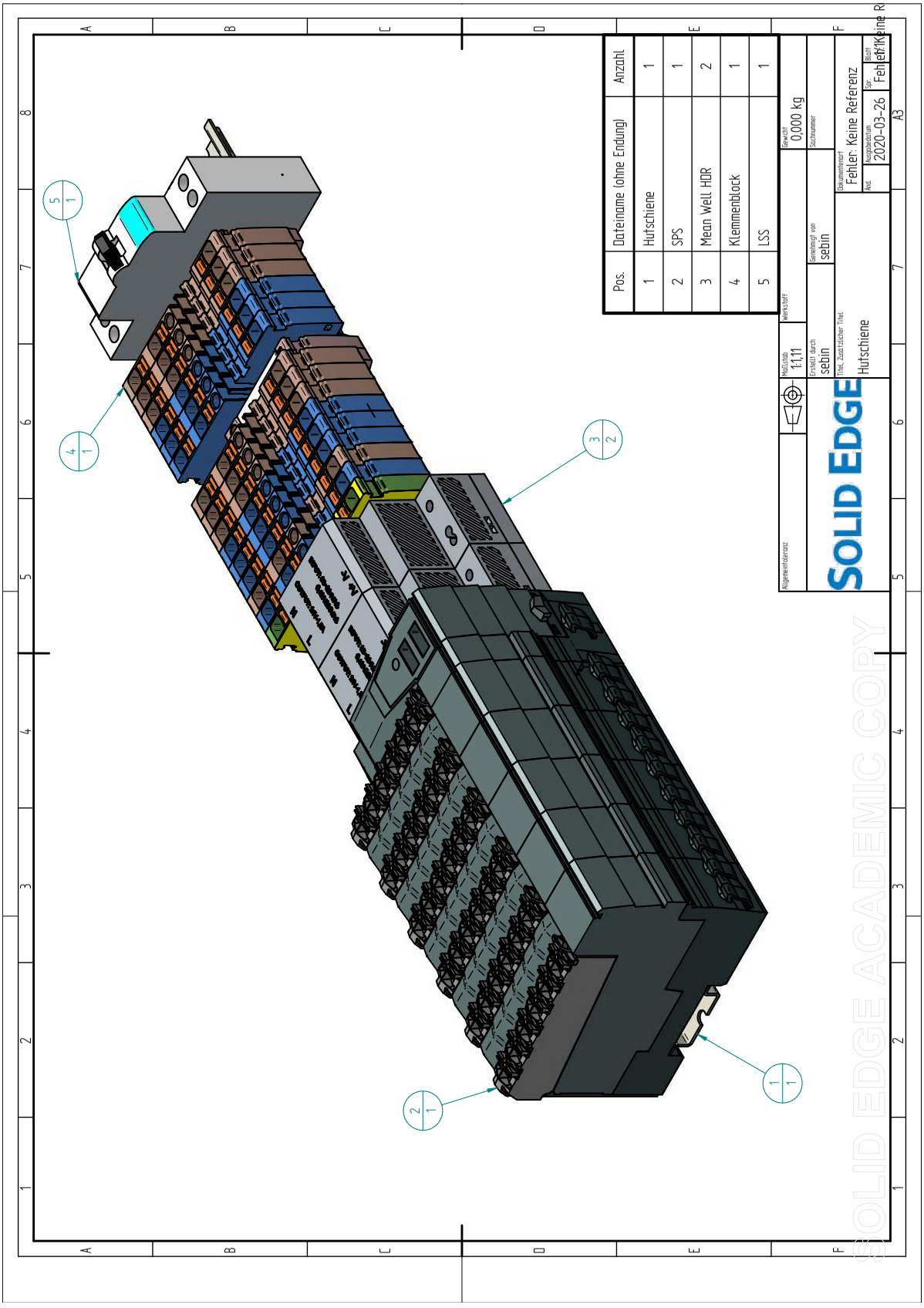
2 Energieversorgung



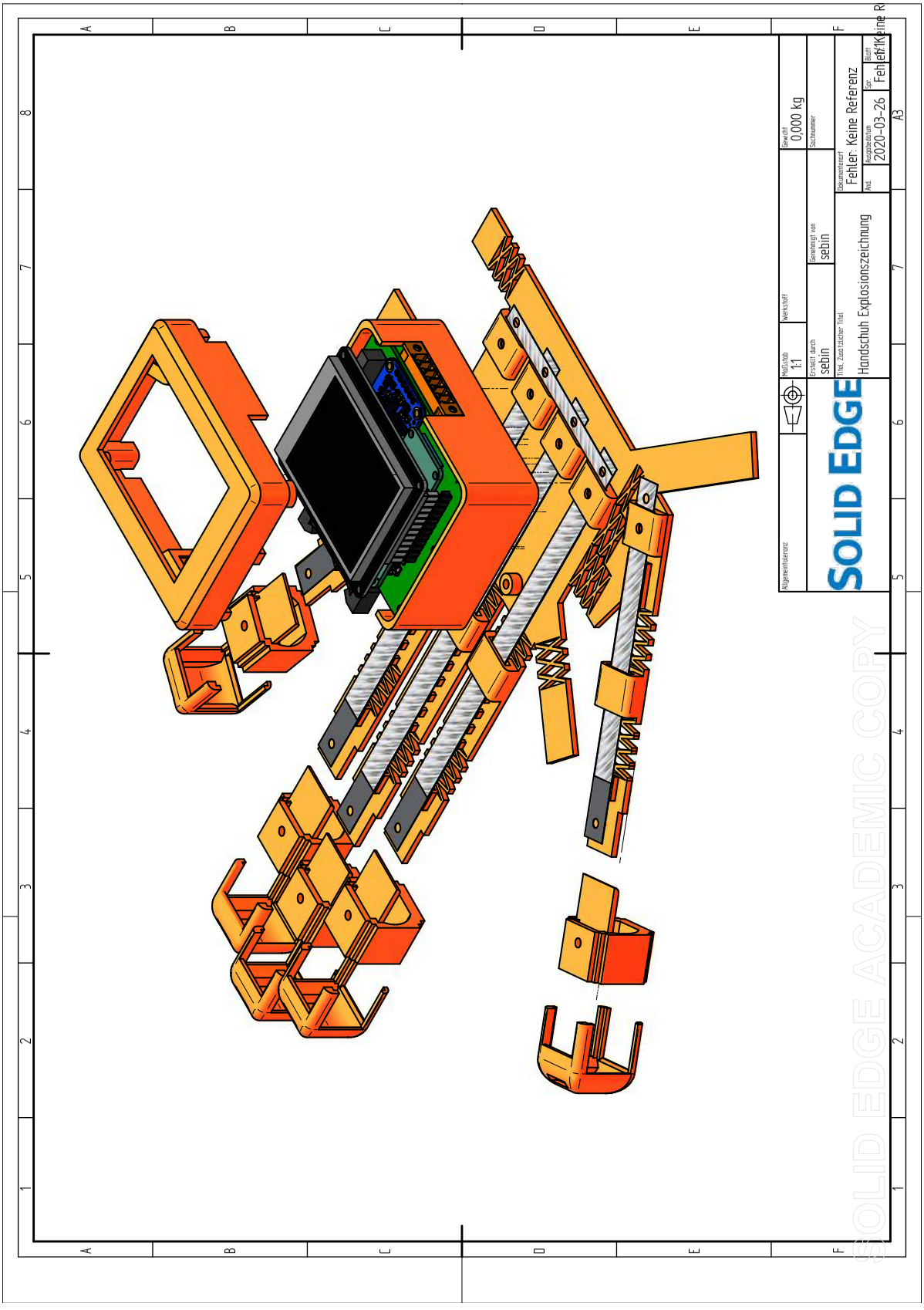

1
2
3
4
A
B
C
D
E
F
20
10
8
6
8
8
45°
2
5
6
6
4
Allgemeintoleranz
Maßstab
2:1
Werkstoff
Gewicht
Fehler: Keine Referenz
Erstellt durch
sebin
Genehmigt von
sebin
Sachnummer
SOLID EDGE
Titel, Zusätzlicher Titel
Kabelhalter
Dokumentenart
Fehler: Keine Referenz
And.
Ausgabedatum
2020-03-12
Spr.
Blatt
1/1
SOLID EDGE ACADEMIC COPY

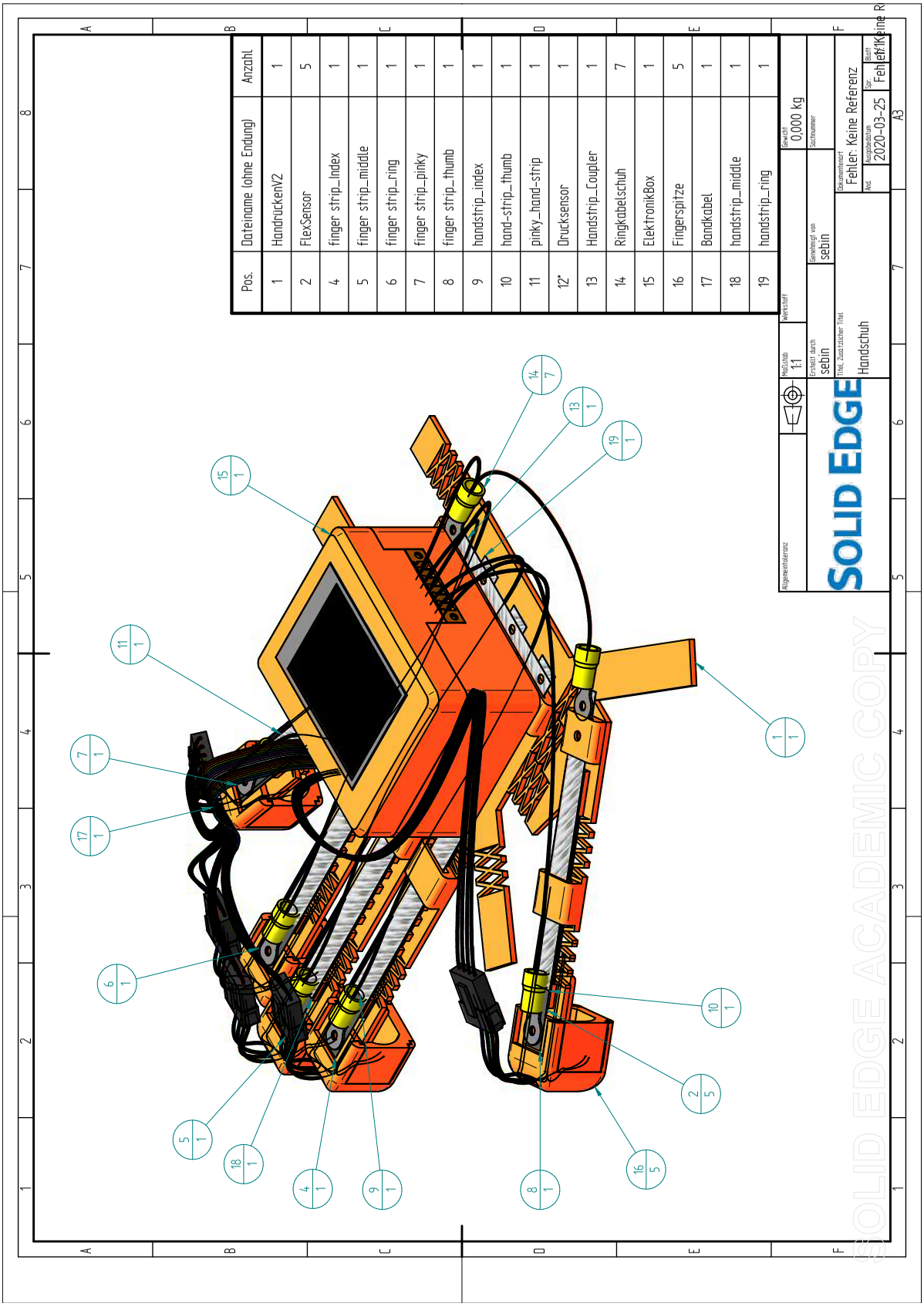

50
53°
40
80
R 2,75
R 2,75
5,5
13
60
30
15,5
45
10
20
10
Fehler: Keine Referenz
Akku-Halter
sebin
1:1
2020-03-12
SOLID EDGE
SOLID EDGE ACADEMIC COPY

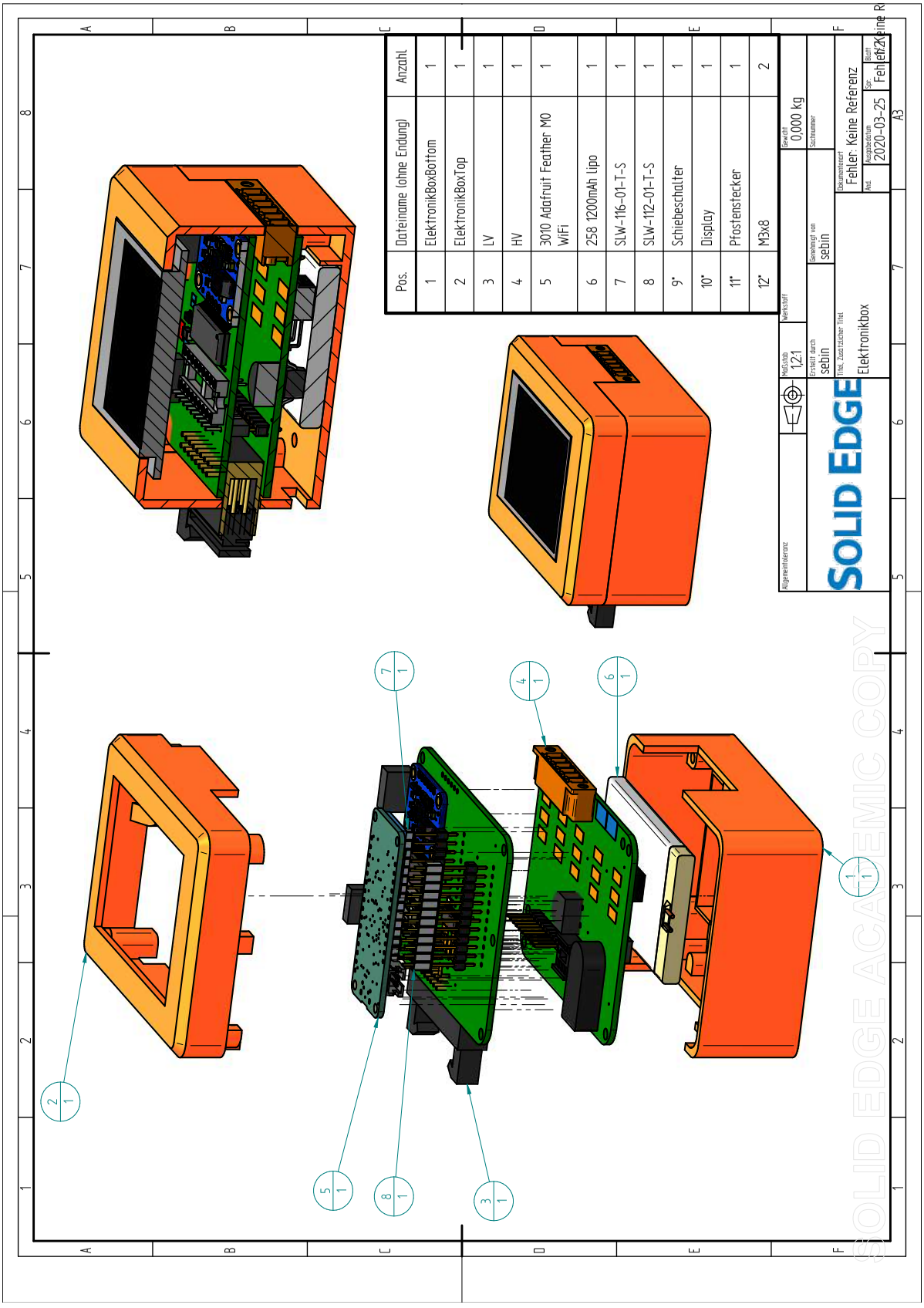

SOLID EDGE
Fehler: Keine Referenz
Abdeckung
sebin
1:1
2020-03-12
162
160
74
75
30
115
SOLID EDGE ACADEMIC COPY

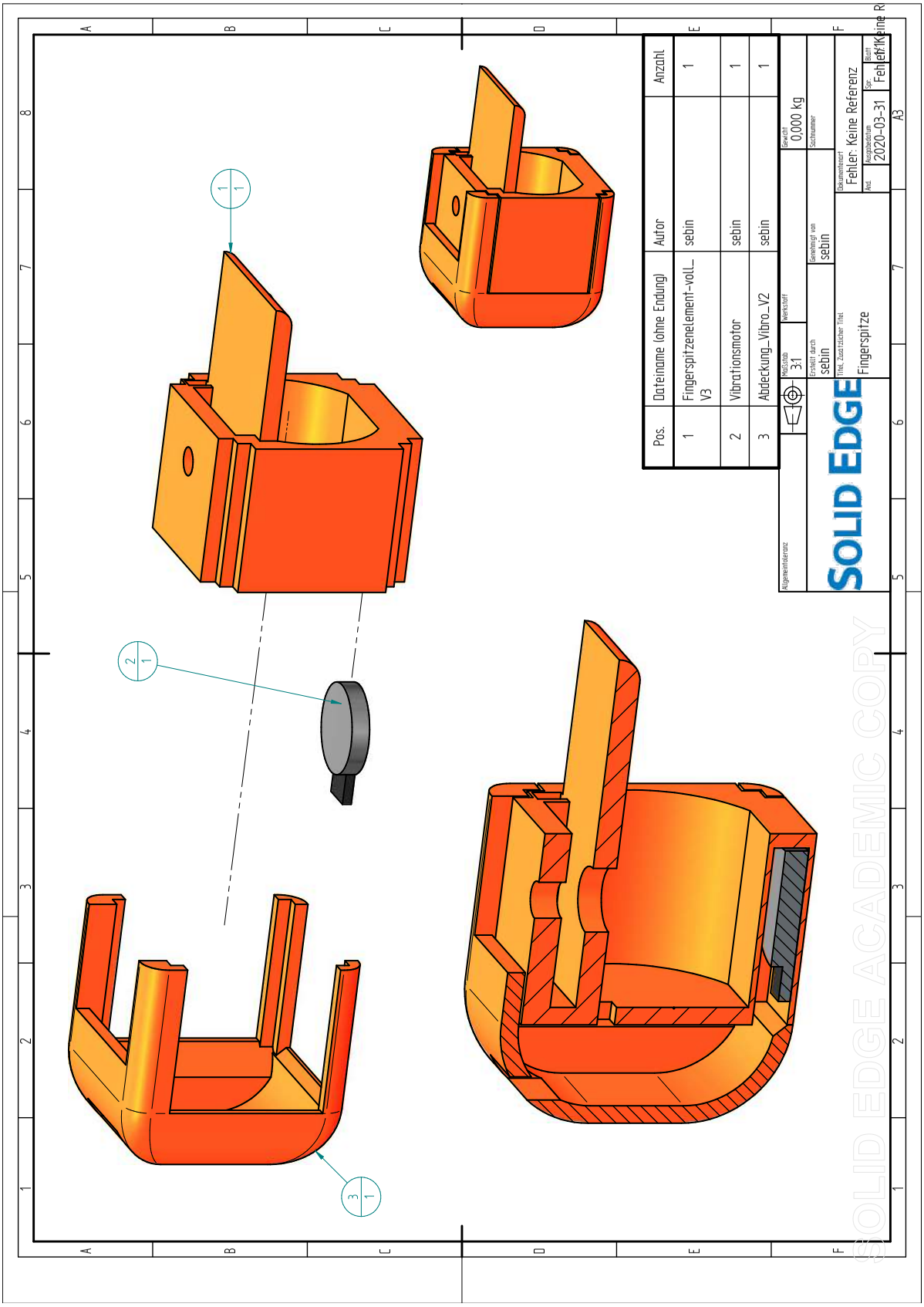

Pos. Dateiname (ohne Endung) Anzahl
1 Hutschiene 1
2 SPS 1
3 Mean Well HDR 2
4 Klemmenblock 1
5 LSS 1
Gewicht 0,000 kg
Sachnummer
Dokumentenart Fehler: Keine Referenz
Ausgabedatum 2020-03-26
Blatt Fehler: Keine Ri
Maßstab 1:1,11
Werkstoff
Erstellt durch sebin
Geändert von sebin
Titel, Zusätzlicher Titel Hutschiene
SOLID EDGE
SOLID EDGE ACADEMIC COPY

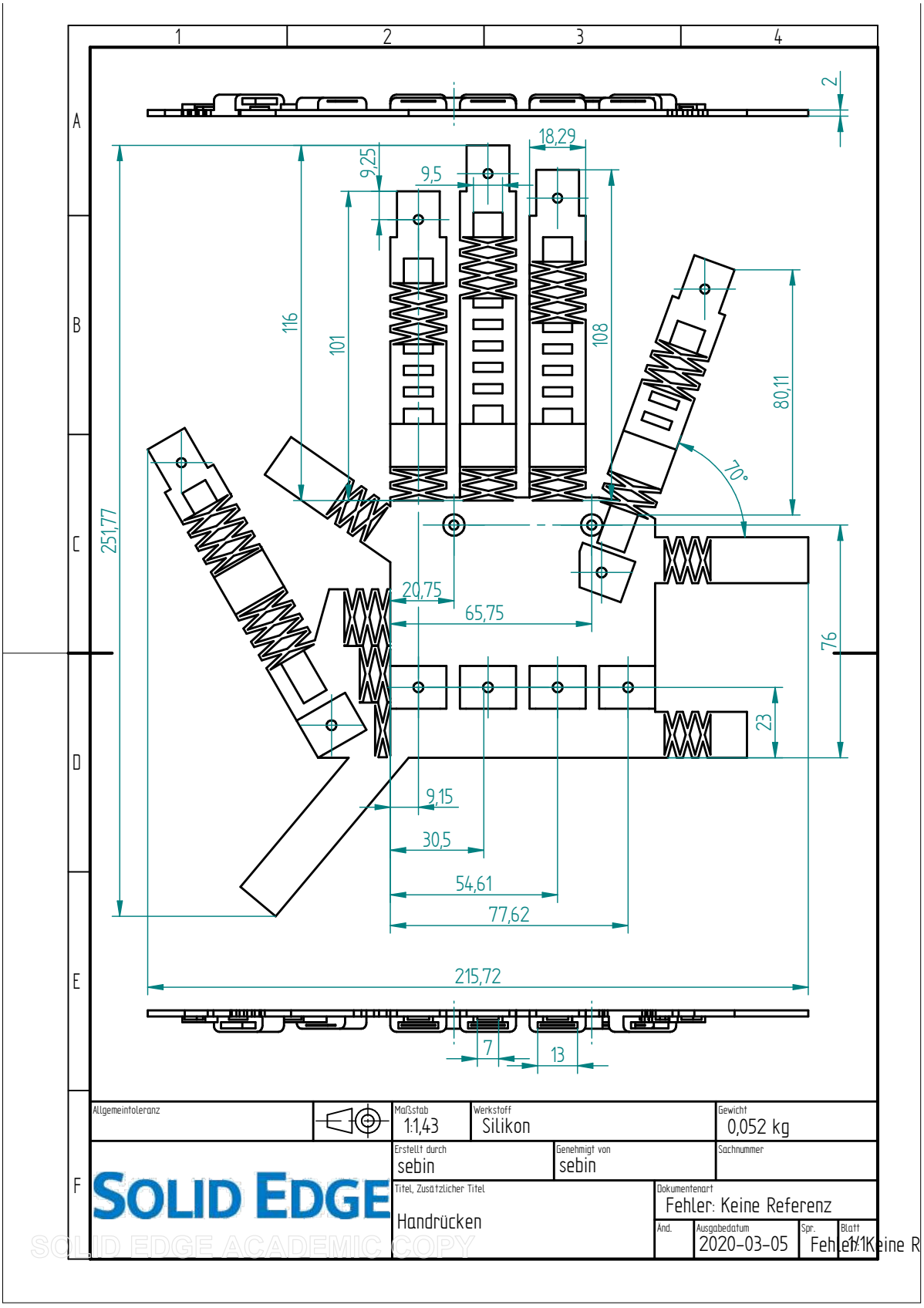
3 Fernsteuergerät

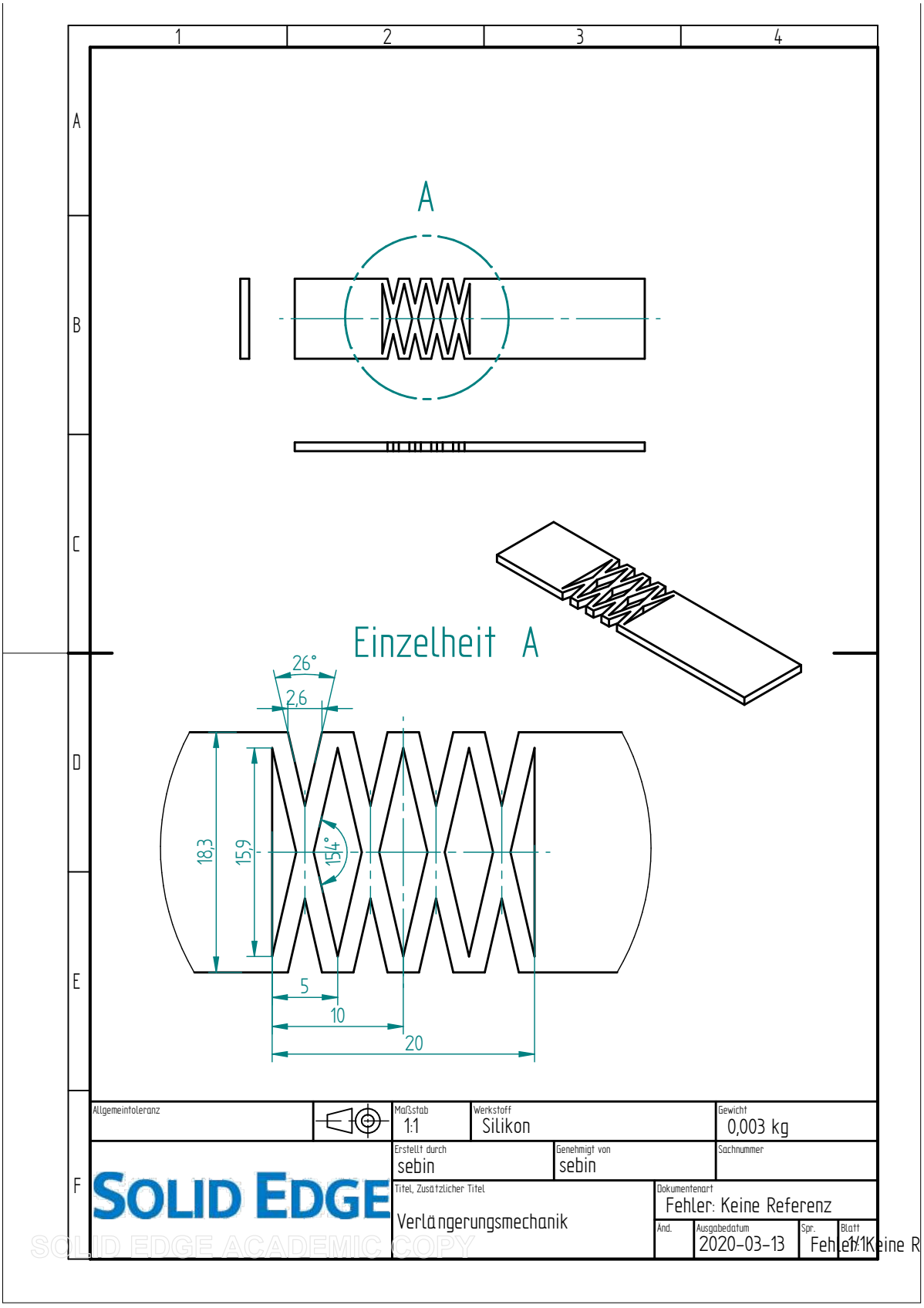


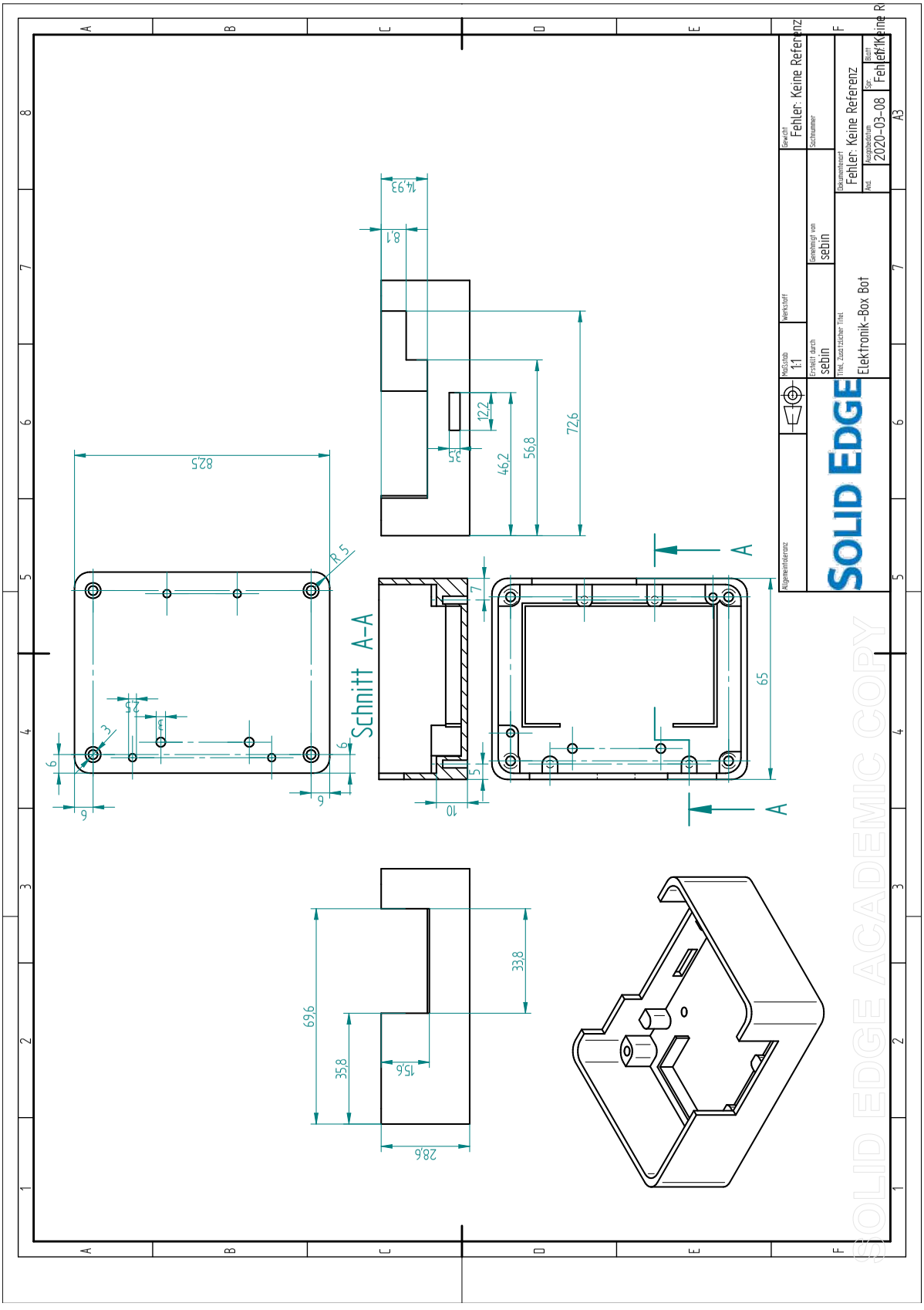

Pos. | Dateiname (ohne Endung) | Anzahl
1 | HandrückenV2 | 1
2 | FlexSensor | 5
4 | finger strip_Index | 1
5 | finger strip_middle | 1
6 | finger strip_ring | 1
7 | finger strip_pinky | 1
8 | finger strip_thumb | 1
9 | handstrip_index | 1
10 | hand-strip_thumb | 1
11 | pinky_hand-strip | 1
12* | Drucksensor | 1
13 | Handstrip_Coupler | 1
14 | Ringkabelschuh | 7
15 | ElektronikBox | 1
16 | Fingerspitze | 5
17 | Bandkabel | 1
18 | handstrip_middle | 1
19 | handstrip_ring | 1
SOLID EDGE
Handschuh
sebin
2020-03-25
Fehler: Keine Referenz
A3
SOLID EDGE ACADEMIC COPY

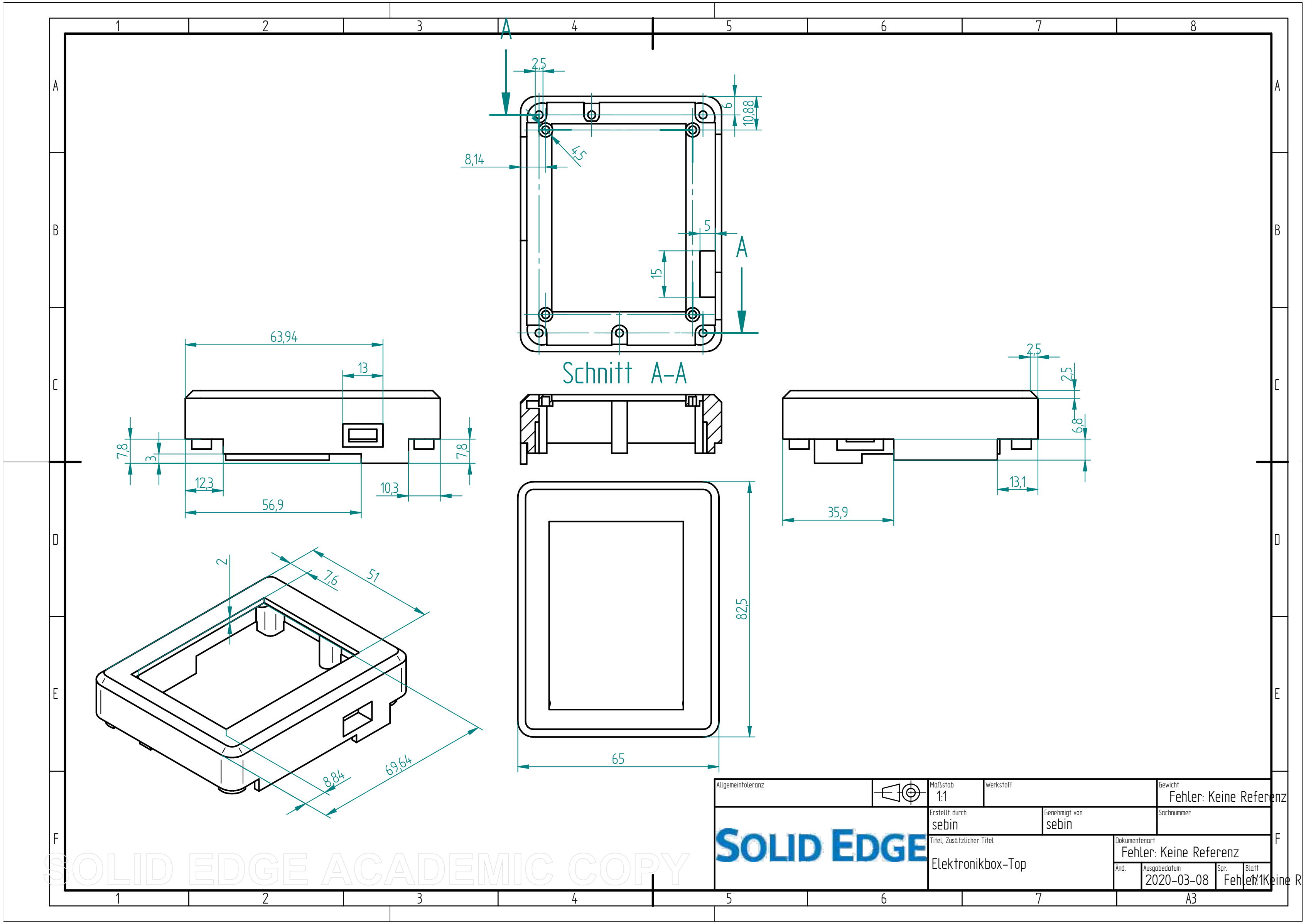

Pos. | Dateiname (ohne Endung) | Anzahl
1 | ElektronikBoxBottom | 1
2 | ElektronikBoxTop | 1
3 | LV | 1
4 | HV | 1
5 | 3010 Adafruit Feather M0 WiFi | 1
6 | 258 1200mAh lipo | 1
7 | SLW-116-01-T-S | 1
8 | SLW-112-01-T-S | 1
9* | Schiebeschalter | 1
10* | Display | 1
11* | Pfostenstecker | 1
12* | M3x8 | 2
Gewicht 0,000 kg
Maßstab 1:2:1
Werkstoff
Erstellt durch sebin
Geändert von sebin
Titel, Zusätzlicher Titel Elektronikbox
Dokumententyp Fehler: Keine Referenz
Ausgabedatum 2020-03-25
Blatt 1/1
SOLID EDGE
SOLID EDGE ACADEMIC COPY

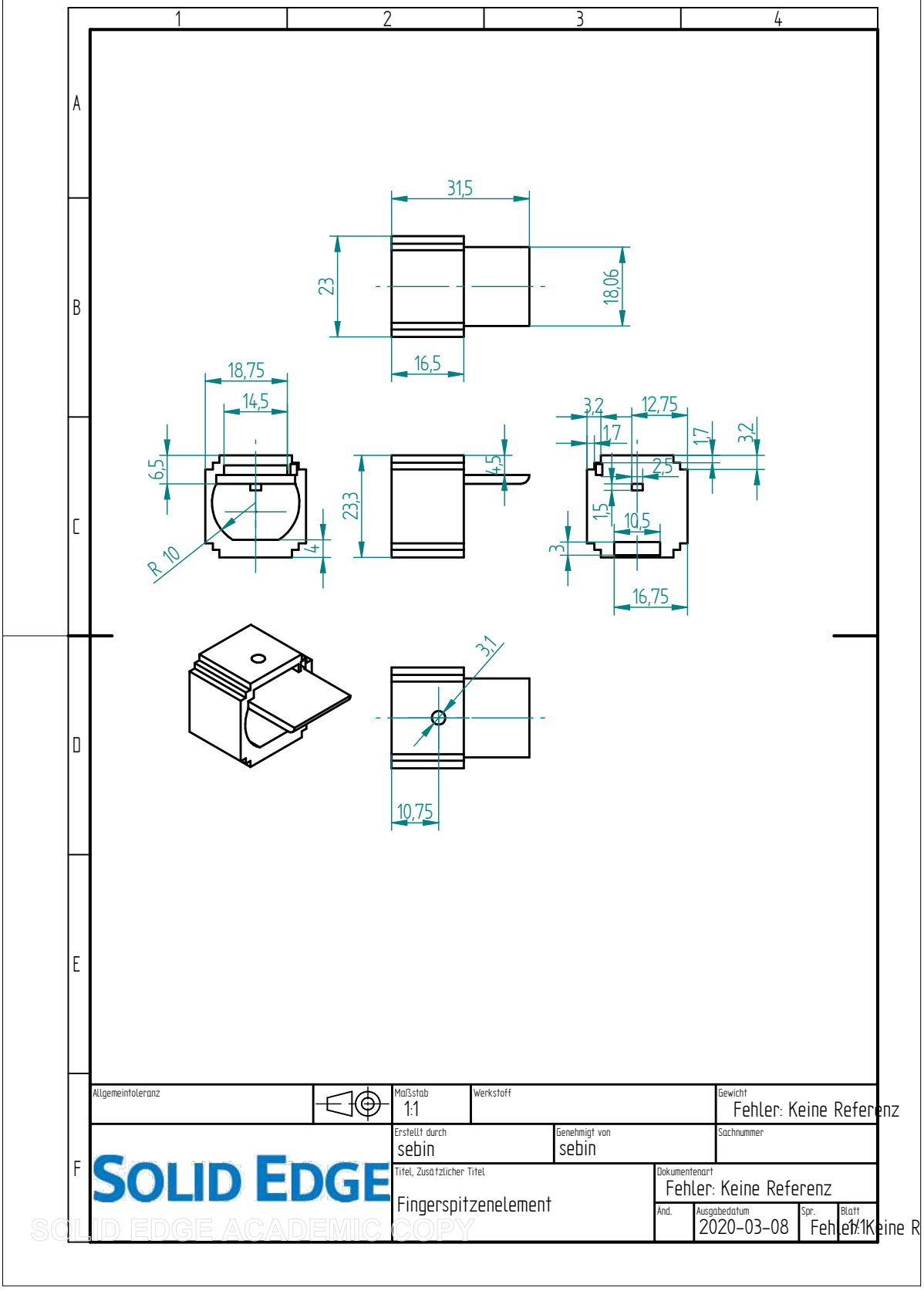

Pos. | Dateiname (ohne Endung) | Autor | Anzahl
1 | Fingerspitzenelement-voll_V3 | sebin | 1
2 | Vibrationsmotor | sebin | 1
3 | Abdeckung_Vibro_V2 | sebin | 1
Maßstab 31
Werkstoff
Erstellt durch sebin
Geändert durch sebin
Titel, Zusätzlicher Titel Fingerspitze
Gewicht 0,000 kg
Sachnummer
Dokumentenart Fehler: Keine Referenz
Ausgabedatum 2020-03-31
Allgemeintoleranz
SOLID EDGE
SOLID EDGE ACADEMIC COPY

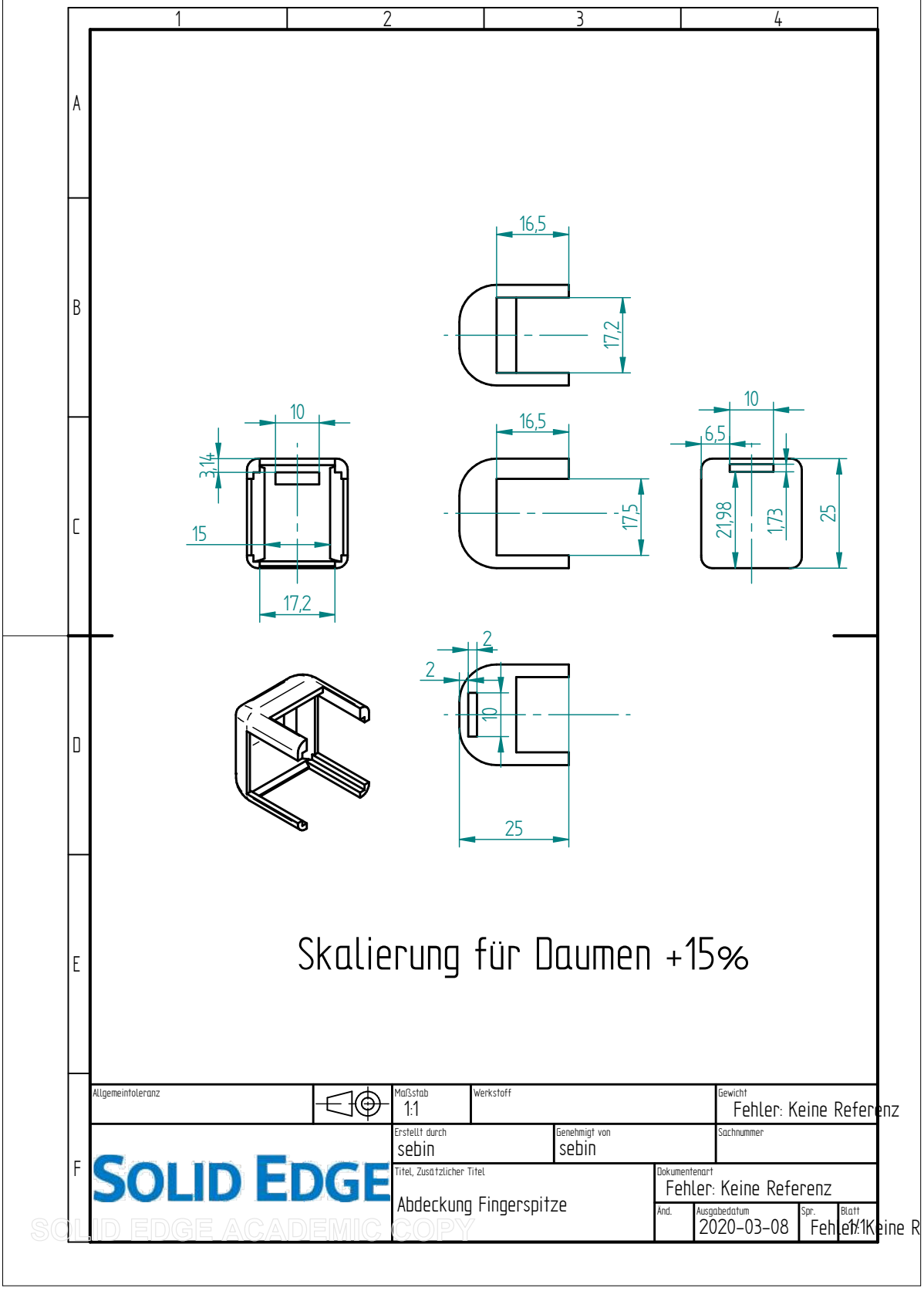

1
2
3
4
A
B
C
D
E
F
2
18,29
9,25
9,5
116
101
108
80,11
70°
251,77
20,75
65,75
76
23
9,15
30,5
54,61
77,62
215,72
7
13
Allgemeintoleranz
Maßstab
1:1,43
Werkstoff
Silikon
Gewicht
0,052 kg
Erstellt durch
sebin
Genehmigt von
sebin
Sachnummer
SOLID EDGE
Titel, Zusätzlicher Titel
Handrücken
Dokumentenart
Fehler: Keine Referenz
And.
Ausgabedatum
2020-03-05
Spr.
Blatt
1/1
SOLID EDGE ACADEMIC COPY


1
2
3
4
A
B
C
D
E
F
A
Einzelheit A
26°
2,6
18,3
15,9
154°
5
10
20
Allgemeintoleranz
Maßstab
1:1
Werkstoff
Silikon
Gewicht
0,003 kg
Erstellt durch
sebin
Genehmigt von
sebin
Sachnummer
SOLID EDGE
Titel, Zusätzlicher Titel
Verlängerungsmechanik
Dokumentenart
Fehler: Keine Referenz
And.
Ausgabedatum
2020-03-13
Spr.
Blatt
1/1
SOLID EDGE ACADEMIC COPY


Schnitt A-A
SOLID EDGE
Fehler: Keine Referenz
Elektronik-Box Bot
sebin
2020-03-08
SOLID EDGE ACADEMIC COPY


Schnitt A-A
63,94
13
7,8
3
12,3
56,9
10,3
8,14
4,5
2,5
6
10,88
5
15
82,5
65
35,9
13,1
6,8
2
7,6
51
8,84
69,64
Allgemeintoleranz
Maßstab
1:1
Werkstoff
Gewicht
Fehler: Keine Referenz
Erstellt durch
sebin
Genehmigt von
sebin
Sachnummer
SOLID EDGE
Titel, Zusätzlicher Titel
Elektronikbox-Top
Dokumentenart
Fehler: Keine Referenz
Ausgabedatum
2020-03-08
Blatt
A3
SOLID EDGE ACADEMIC COPY

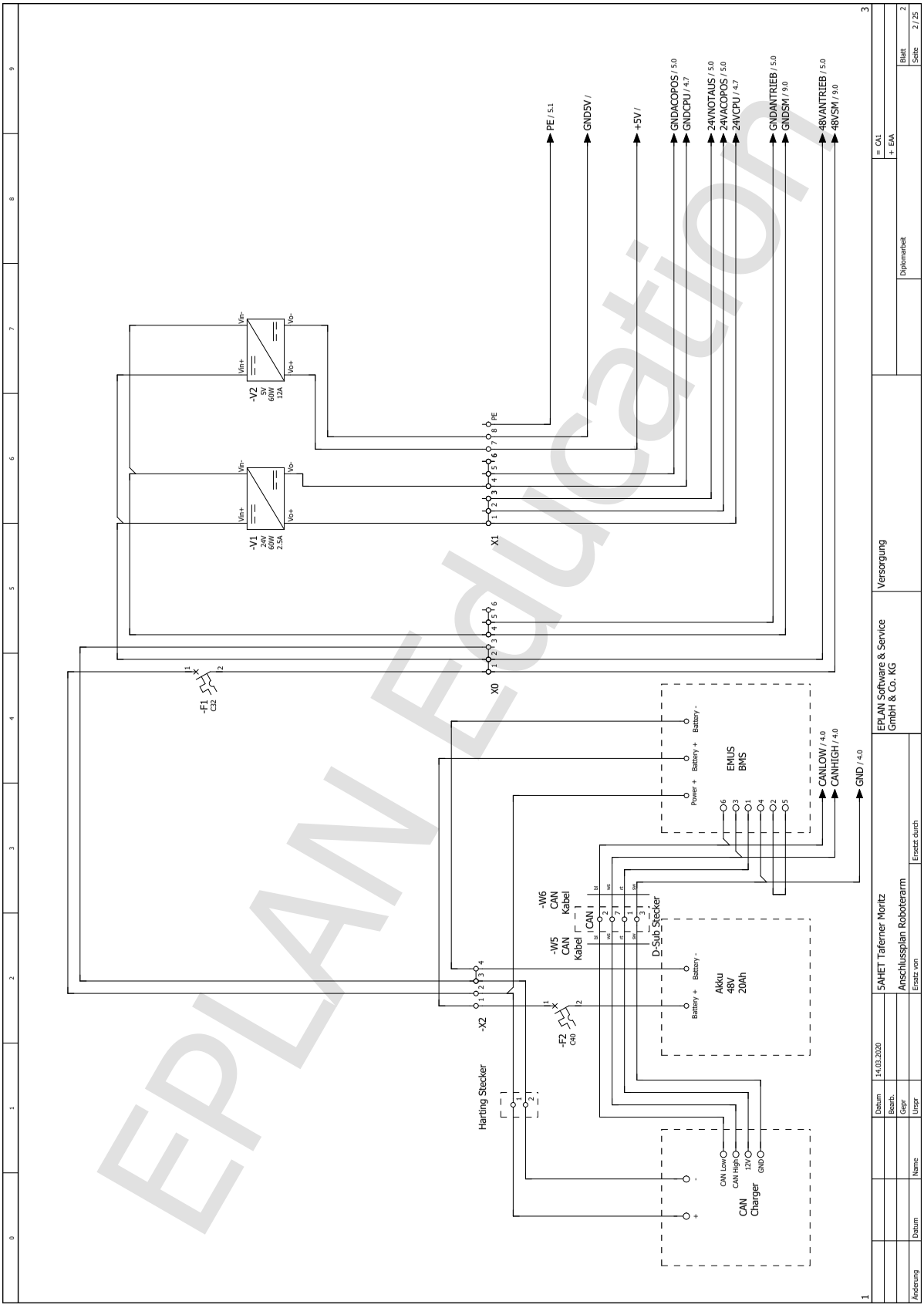

1
2
3
4
A
B
C
D
E
F
31,5
23
18,06
16,5
18,75
14,5
6,5
R 10
4
23,3
4,5
3,2
12,75
1,7
1,7
3,2
2,5
1,5
10,5
3
16,75
3,1
10,75
Allgemeintoleranz
Maßstab
1:1
Werkstoff
Gewicht
Fehler: Keine Referenz
Erstellt durch
sebin
Genehmigt von
sebin
Sachnummer
SOLID EDGE
Titel, Zusätzlicher Titel
Fingerspitzenelement
Dokumentenart
Fehler: Keine Referenz
And.
Ausgabedatum
2020-03-08
Spr.
Blatt
1/1
SOLID EDGE ACADEMIC COPY

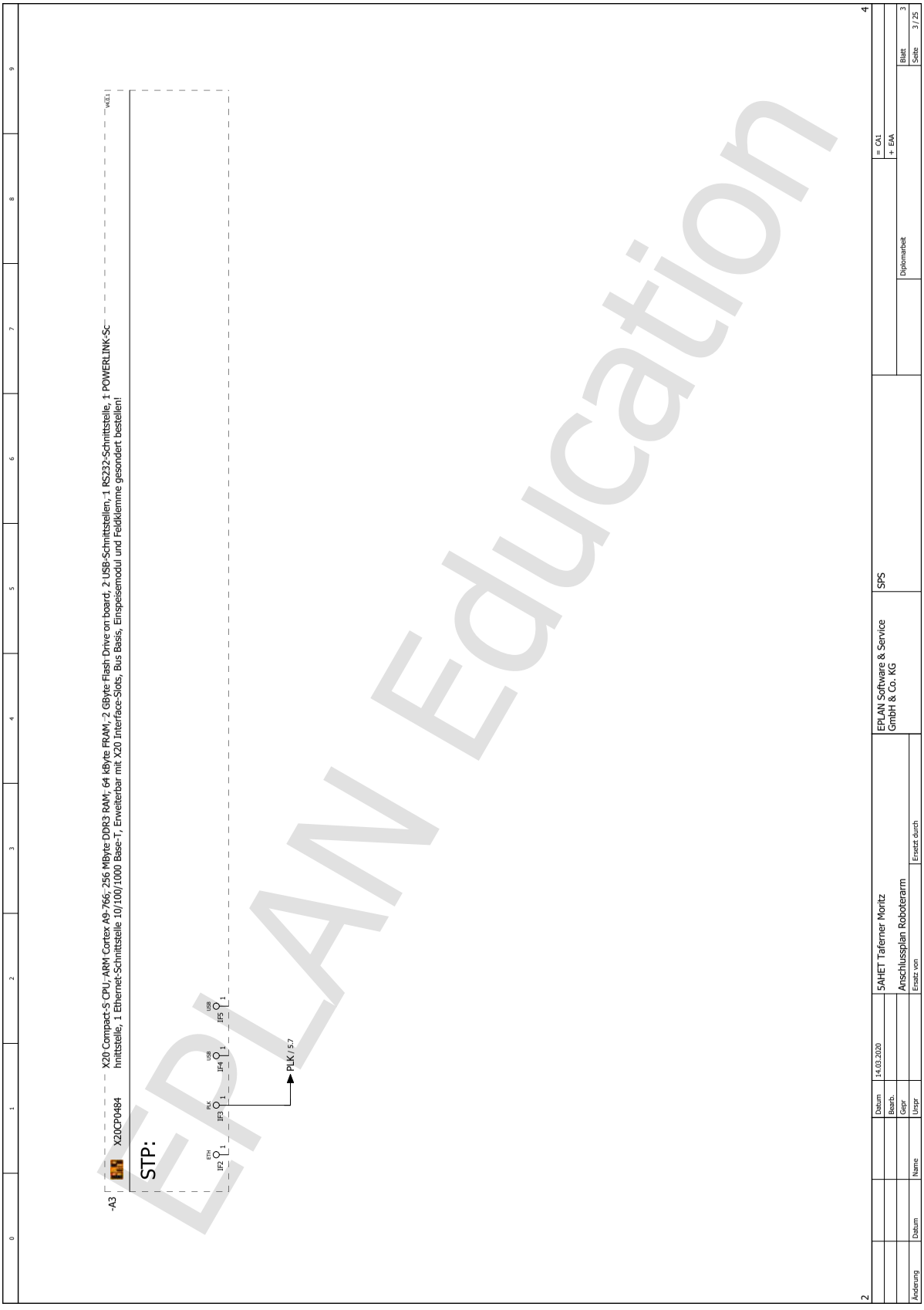

1
2
3
4
A
B
C
D
E
F
16,5
17,2
10
3,14
15
17,2
16,5
17,5
10
6,5
21,98
1,73
25
2
2
10
25
Skalierung für Daumen +15%
Allgemeintoleranz
Maßstab
1:1
Werkstoff
Gewicht
Fehler: Keine Referenz
Erstellt durch
sebin
Genehmigt von
sebin
Sachnummer
Titel, Zusätzlicher Titel
Abdeckung Fingerspitze
Dokumentenart
Fehler: Keine Referenz
And.
Ausgabedatum
2020-03-08
Spr.
Blatt
1 / 1
SOLID EDGE
SOLID EDGE ACADEMIC COPY

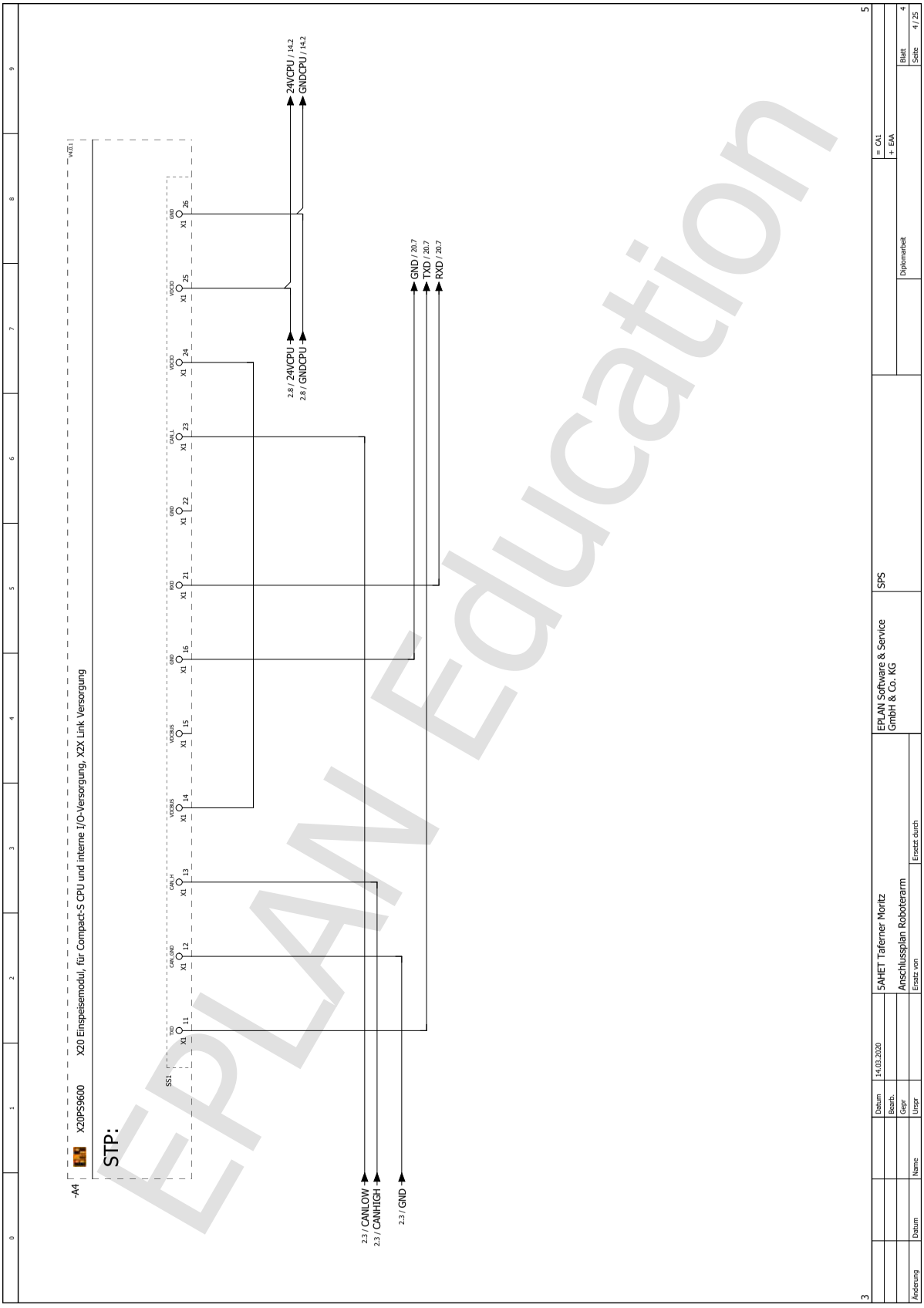
Anhang C

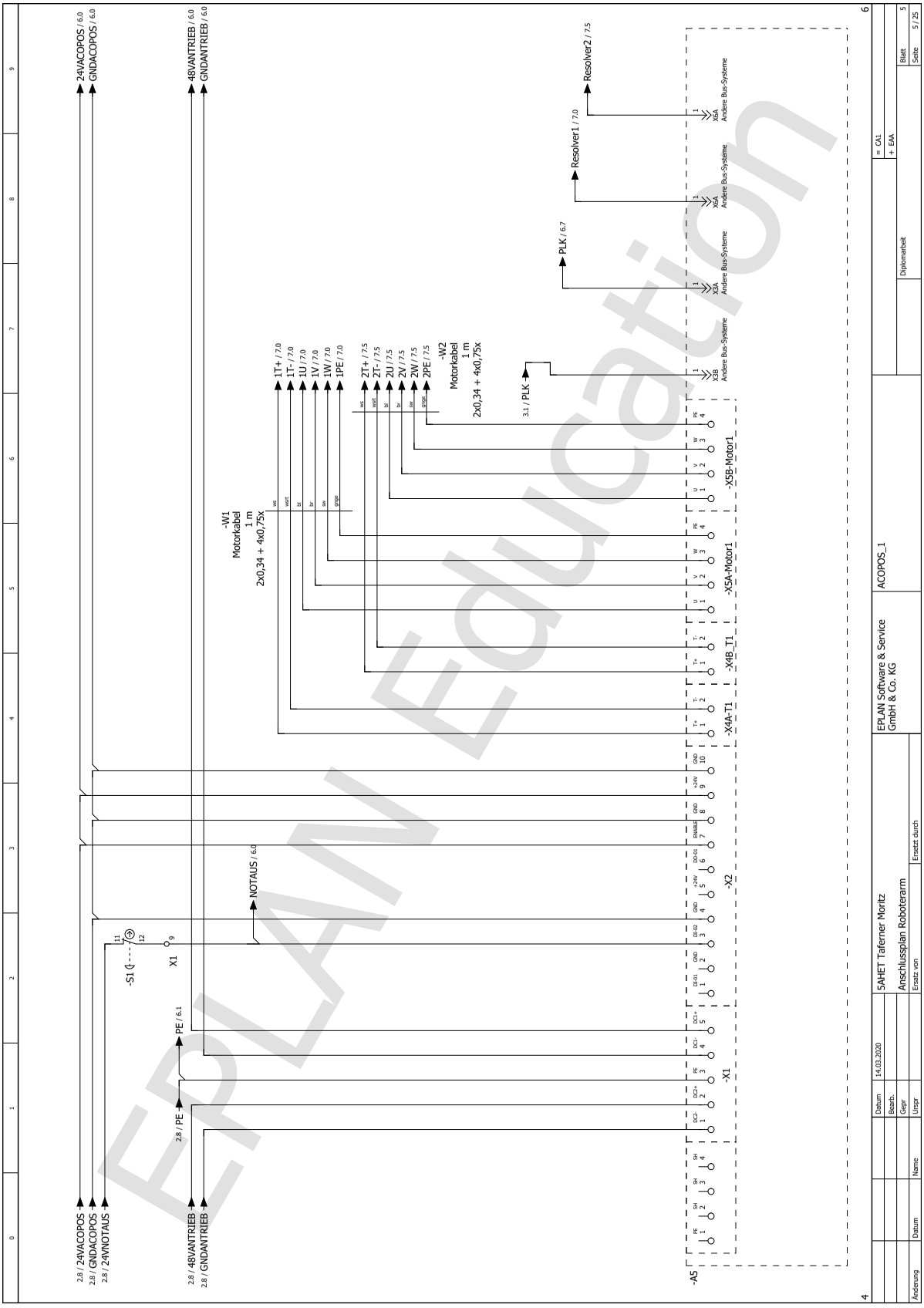
Schaltpläne und Layouts

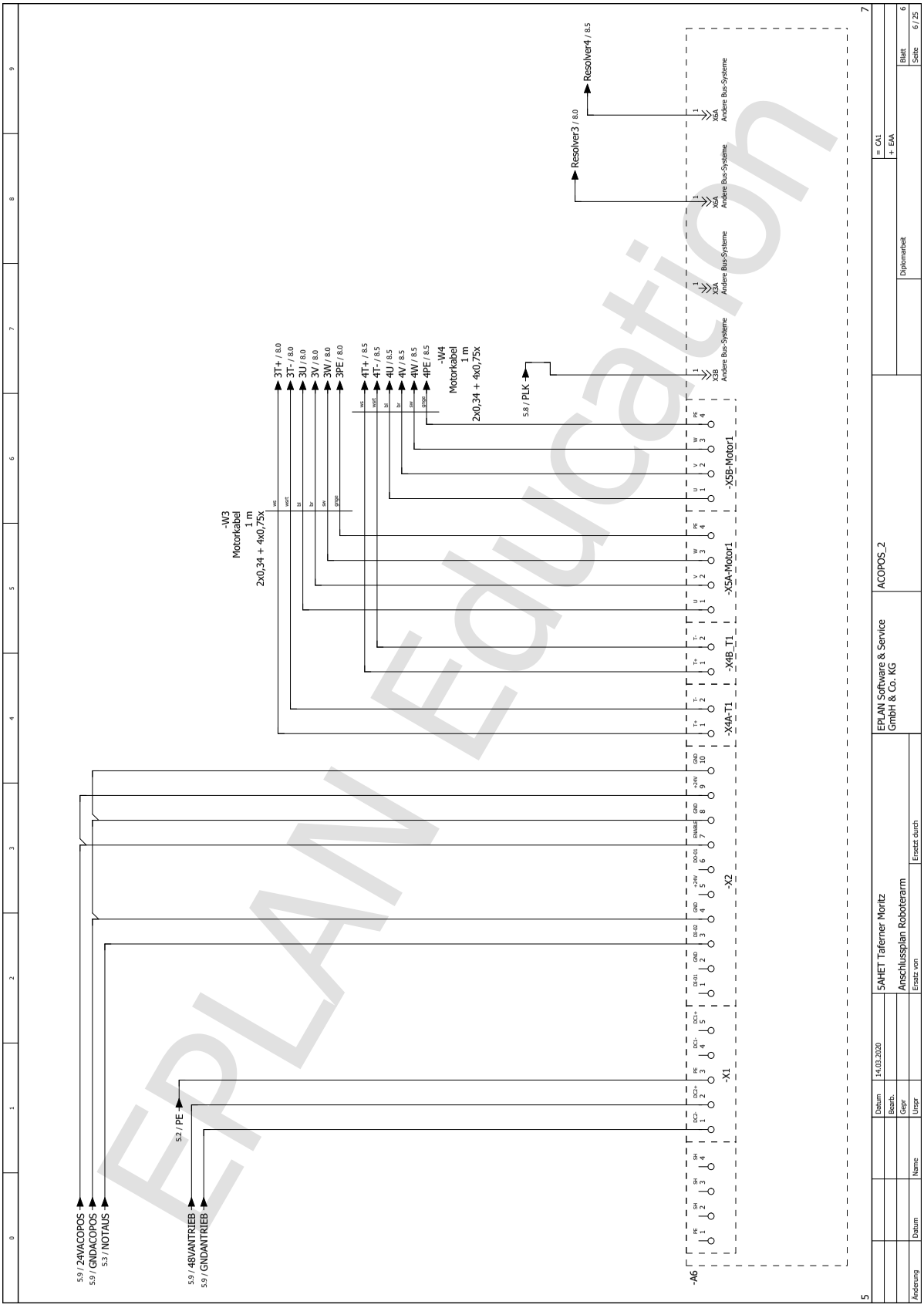
1 Robotersystem

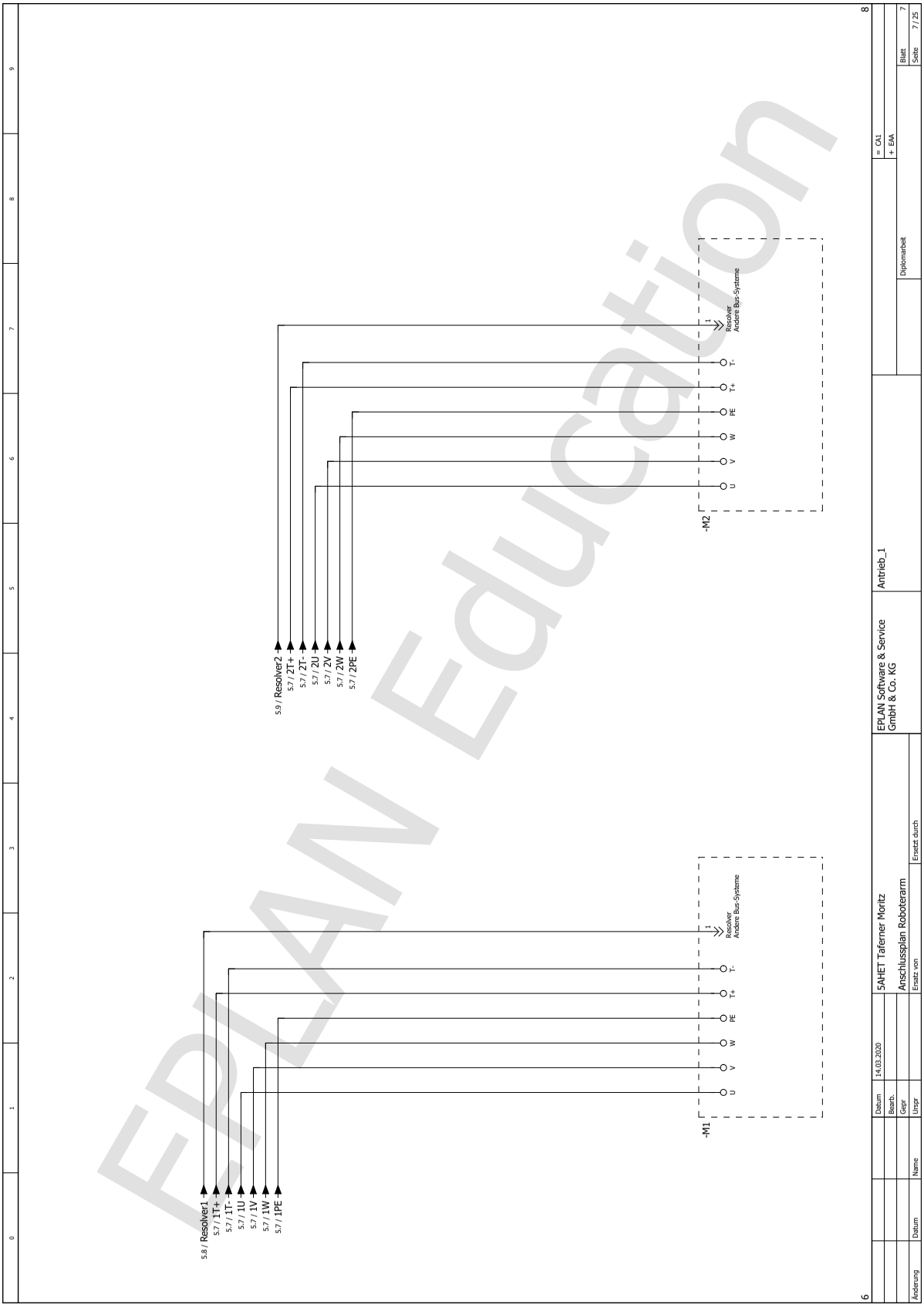


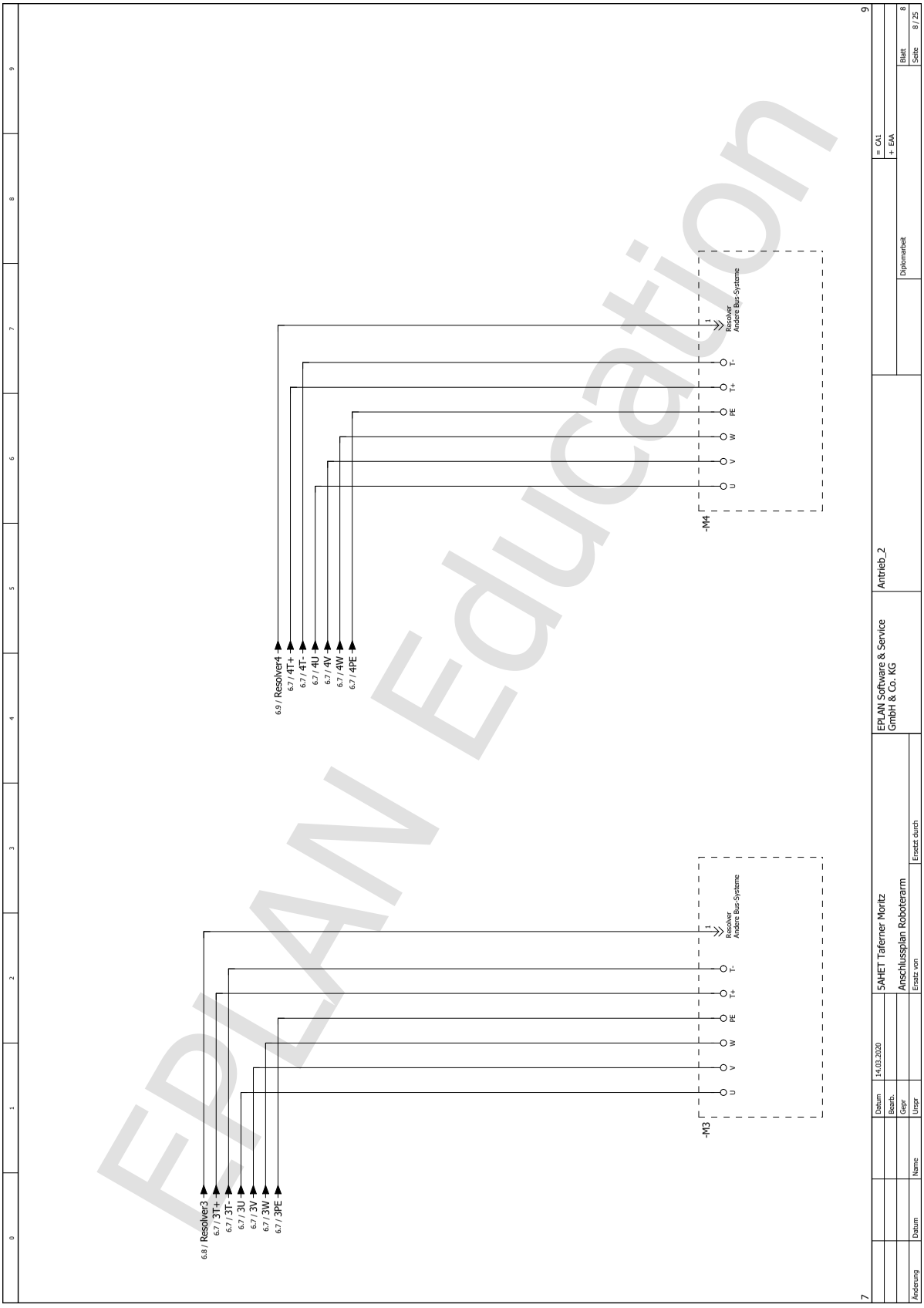

-A3
X20CP0484
X20 Compact-S CPU,-ARM Cortex A9-766; 256 MByte DDR3 RAM; 64 kByte FRAM,-2 GByte-Flash-Drive on board, 2 USB-Schnittstellen,-1 RS232-Schnittstelle,-1 POWERLINK-Sc
hnittstelle, 1 Ethernet-Schnittstelle 10/100/1000 Base-T, Erweiterbar mit X20 Interface-Slots, Bus Basis, Einspeisemodul und Feldklemme gesondert bestellen!
STP:
ETH
IF2
PLK
IF3
USB
IF4
USB
IF5
PLK / 5.7
EPLAN Software & Service GmbH & Co. KG
5AHET Taferner Moritz
Anschlussplan Roboterarm
SPS
Diplomarbeit
Datum
Name
Änderung
Bearb.
Gepr
Ursprr
Ersatz von
Ersetzt durch
Blatt 3
Seite 3 / 25
EPLAN Education

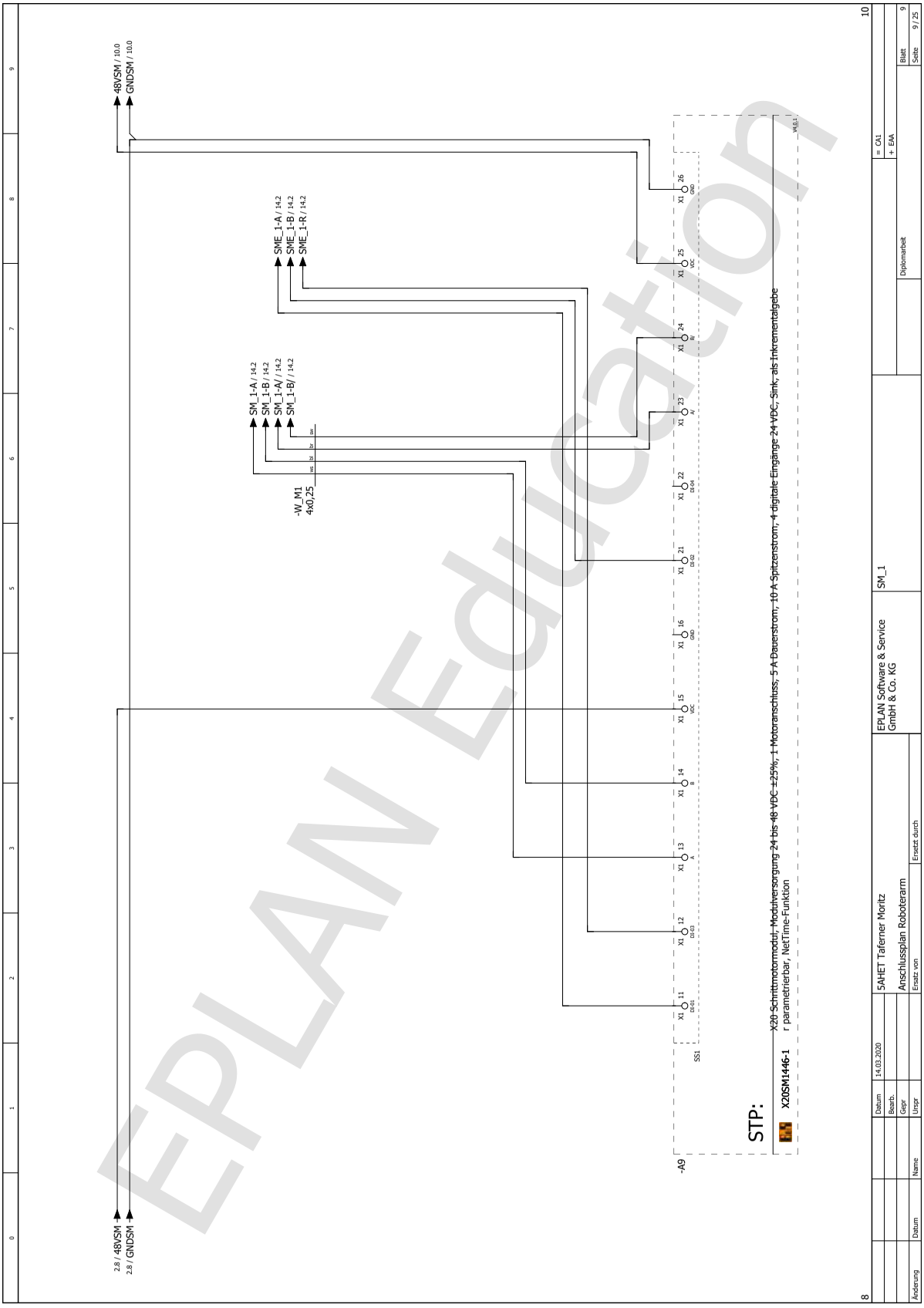

-A4
X20PS9600
X20 Einspeisemodul, für Compact-S CPU und interne I/O-Versorgung, X2X Link Versorgung
STP:
SS1
TXD
CAN GND
CAN_H
VCCBUS
VCCBUS
GND
RXD
GND
CAN_L
VCCIO
VCCIO
GND
X1
11
12
13
14
15
16
21
22
23
24
25
26
24VCPU / 14.2
GNDCPU / 14.2
2.8 / 24VCPU
2.8 / GNDCPU
GND / 20.7
TXD / 20.7
RXD / 20.7
2.3 / CANLOW
2.3 / CANHIGH
2.3 / GND
EPLAN Software & Service GmbH & Co. KG
SPS
5AHET Taferner Moritz
Anschlussplan Roboterarm
Datum 14.03.2020
Bearb.
Gepr
Urspr
Ersatz von
Ersetzt durch
Name
Datum
Änderung
Diplomarbeit
Blatt 4
Seite 4 / 25
EPLAN Education

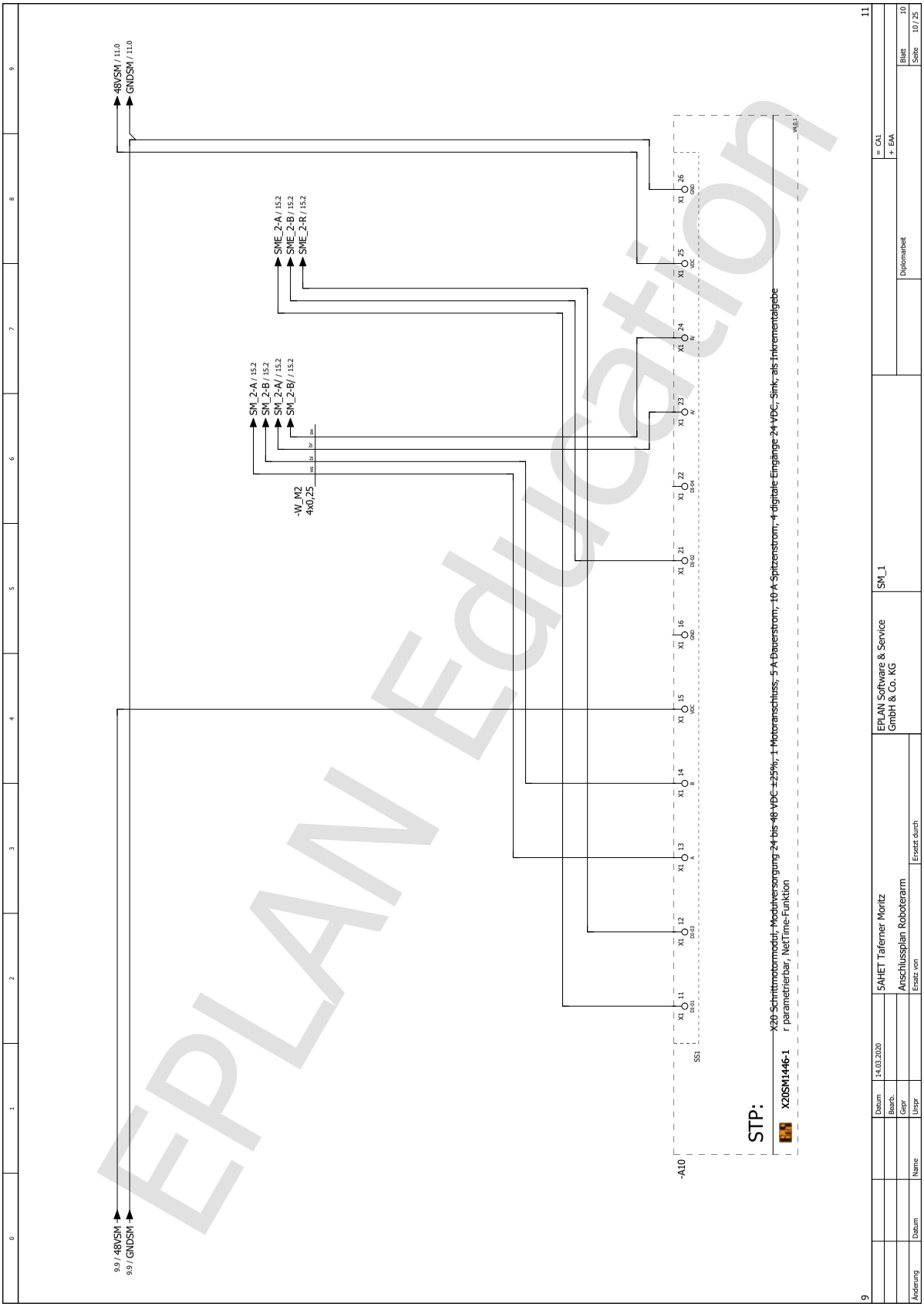

24VACOPOS / 6.0
GNDACOPOS / 6.0
48VANTRIEB / 6.0
GNDANTRIEB / 6.0
Resolver2 / 7.5
Resolver1 / 7.0
PLK / 6.7
-W1
Motorkabel
1 m
2x0,34 + 4x0,75x
-W2
Motorkabel
1 m
2x0,34 + 4x0,75x
1T+ / 7.0
1T- / 7.0
1U / 7.0
1V / 7.0
1W / 7.0
1PE / 7.0
2T+ / 7.5
2T- / 7.5
2U / 7.5
2V / 7.5
2W / 7.5
2PE / 7.5
3.1 / PLK
NOTAUS / 6.0
PE / 6.1
2.8 / PE
-S1
X1
-X1
-X2
-X4A-T1
-X4B_T1
-X5A-Motor1
-X5B-Motor1
-A5
2.8 / 24VACOPOS
2.8 / GNDACOPOS
2.8 / 24VNOTAUS
2.8 / 48VANTRIEB
2.8 / GNDANTRIEB
Andere Bus-Systeme
EPLAN Software & Service GmbH & Co. KG
ACOPOS_1
5AHET Taferner Moritz
Anschlussplan Roboterarm
Diplomarbeit
14.03.2020
EPLAN Education

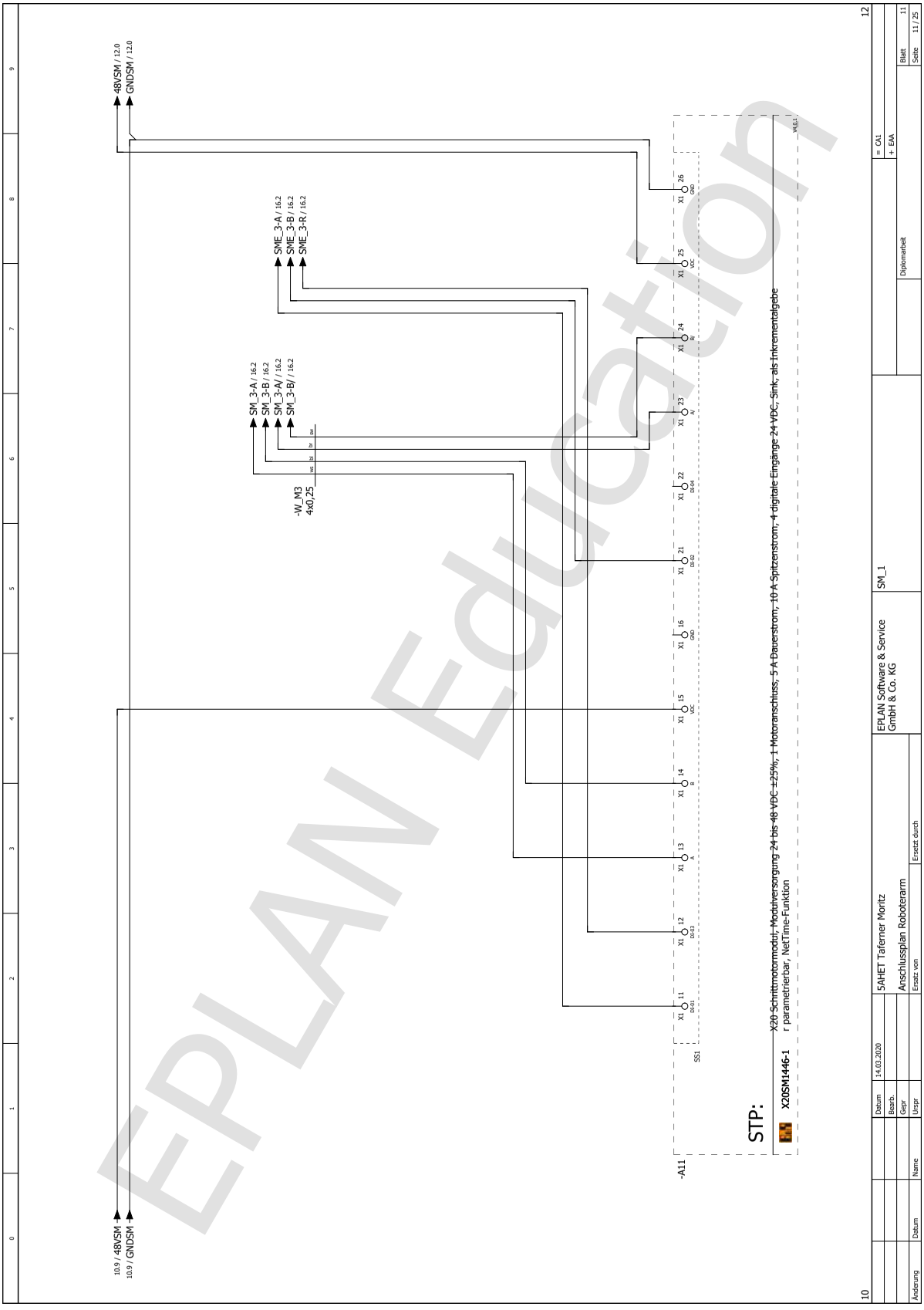

-A6
-X1
-X2
-X4A-T1
-X4B_T1
-X5A-Motor1
-X5B-Motor1
-W3
Motorkabel
1 m
2x0,34 + 4x0,75x
-W4
Motorkabel
1 m
2x0,34 + 4x0,75x
3T+ / 8.0
3T- / 8.0
3U / 8.0
3V / 8.0
3W / 8.0
3PE / 8.0
4T+ / 8.5
4T- / 8.5
4U / 8.5
4V / 8.5
4W / 8.5
4PE / 8.5
5.8 / PLK
Resolver3 / 8.0
Resolver4 / 8.5
Andere Bus-Systeme
5.9 / 24VACOPOS
5.9 / GNDACOPOS
5.3 / NOTAUS
5.2 / PE
5.9 / 48VANTRIEB
5.9 / GNDANTRIEB
EPLAN Software & Service GmbH & Co. KG
ACOPOS_2
5AHET Taferner Moritz
Anschlussplan Roboterarm
Diplomarbeit
14.03.2020
Datum
Name
Änderung
Ersatz von
Ersetzt durch
Blatt 6
Seite 6 / 25
EPLAN Education

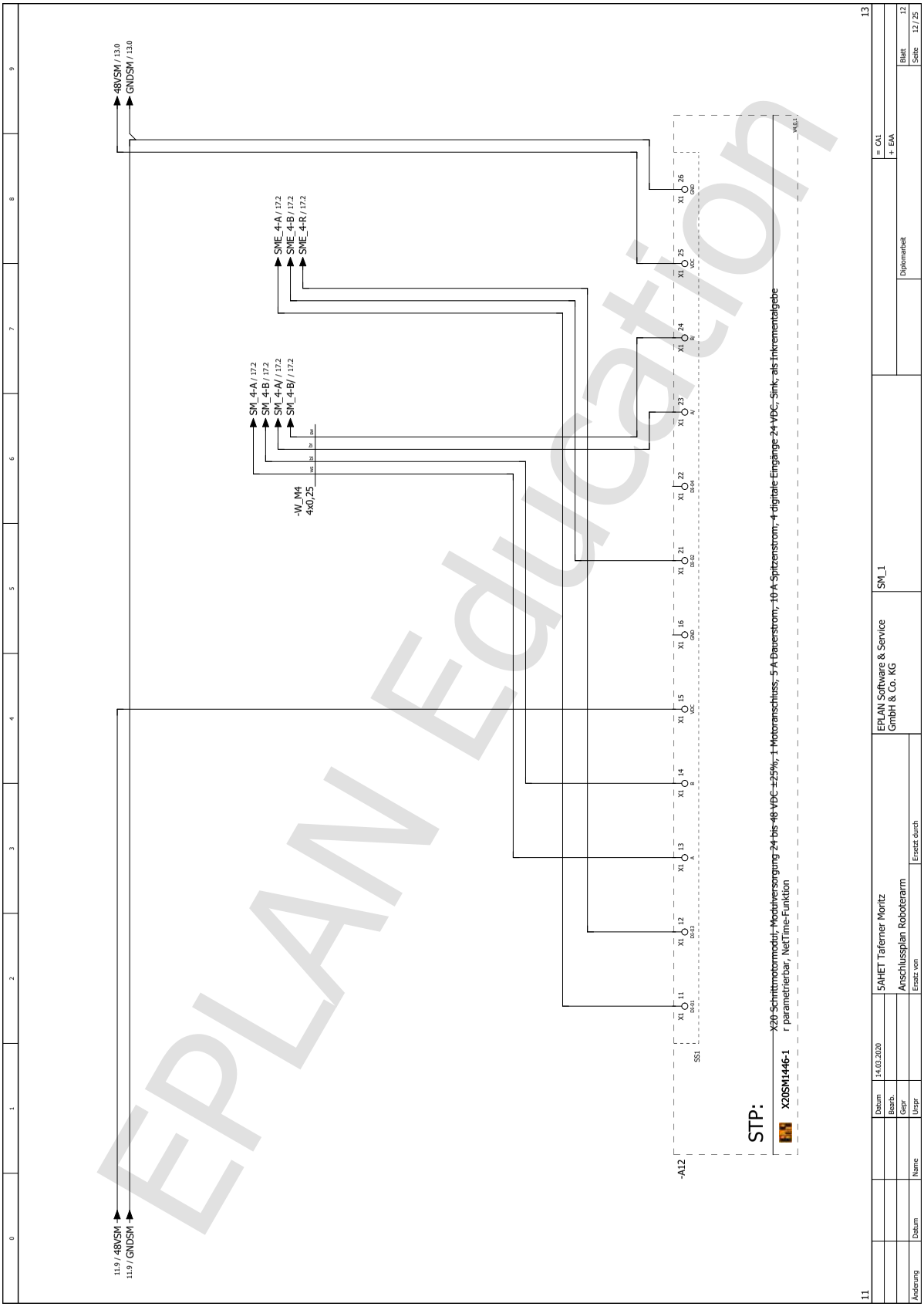

5.8 / Resolver1
5.7 / 1T+
5.7 / 1T-
5.7 / 1U
5.7 / 1V
5.7 / 1W
5.7 / 1PE
5.9 / Resolver2
5.7 / 2T+
5.7 / 2T-
5.7 / 2U
5.7 / 2V
5.7 / 2W
5.7 / 2PE
-M1
-M2
Resolver
Andere Bus-Systeme
T-
T+
PE
W
V
U
EPLAN Software & Service GmbH & Co. KG
Antrieb_1
5AHET Taferner Moritz
Anschlussplan Roboterarm
14.03.2020
Diplomarbeit
Blatt 7
Seite 7 / 25
EPLAN Education

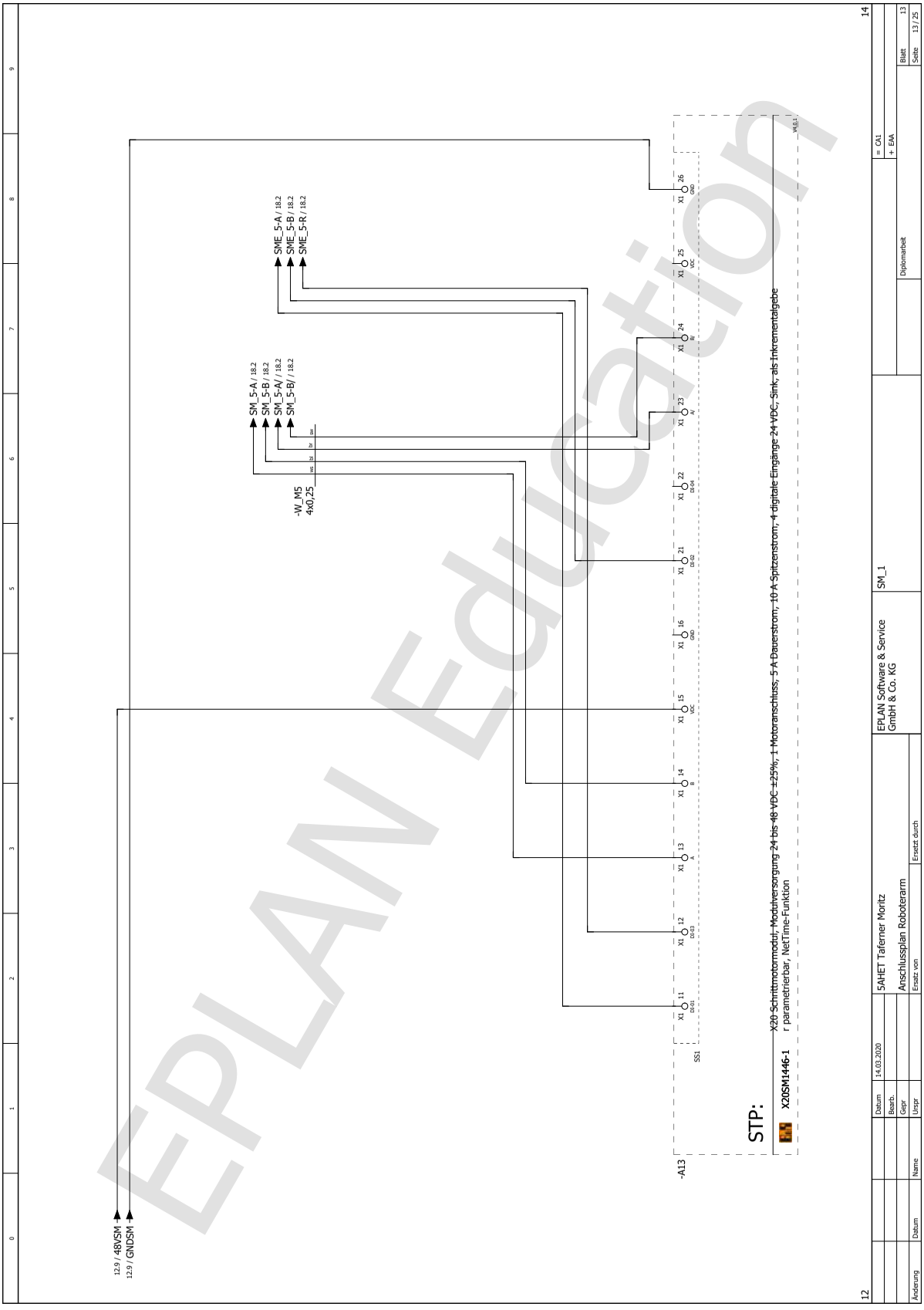


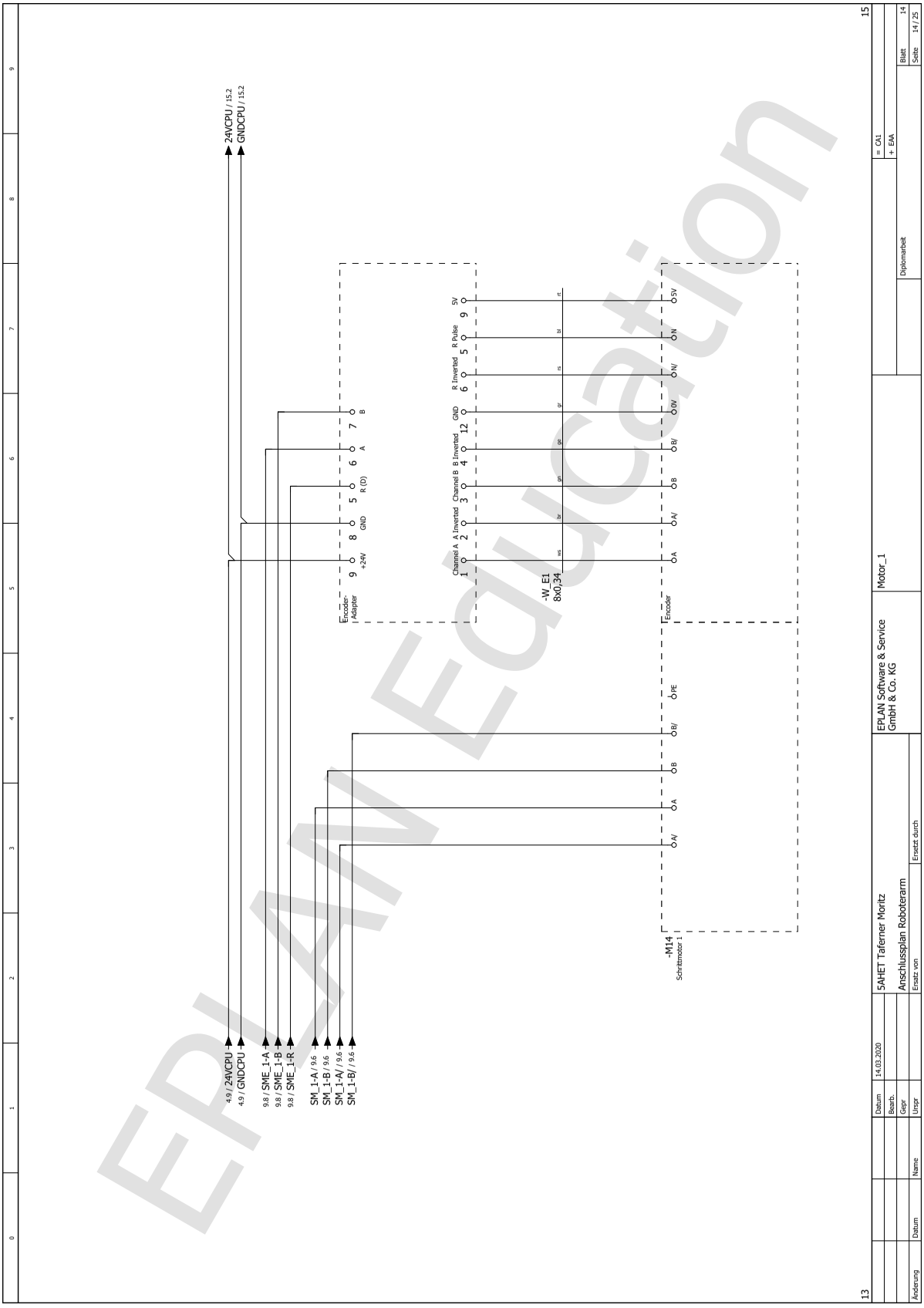

48VSM / 10.0
GNDSM / 10.0
SME_1-A / 14.2
SME_1-B / 14.2
SME_1-R / 14.2
SM_1-A / 14.2
SM_1-B / 14.2
SM_1-A/ / 14.2
SM_1-B/ / 14.2
-W_M1
4x0,25
-A9
SS1
STP:
X20SM1446-1
X20 Schrittmotormodul, Modulversorgung 24 bis 48 VDC ±25%, 1 Motoranschluss, 5 A Dauerstrom, 10 A Spitzenstrom, 4 digitale Eingänge 24 VDC, Sink, als Inkrementalgebe
r parametrierbar, NetTime-Funktion
2.8 / 48VSM
2.8 / GNDSM
EPLAN Software & Service GmbH & Co. KG
SM_1
5AHET Taferner Moritz
Anschlussplan Roboterarm
Diplomarbeit
14.03.2020
Datum
Bearb.
Gepr.
Urspr.
Ersatz von
Ersetzt durch
Name
Änderung
Blatt 9
Seite 9 / 25
EPLAN Education

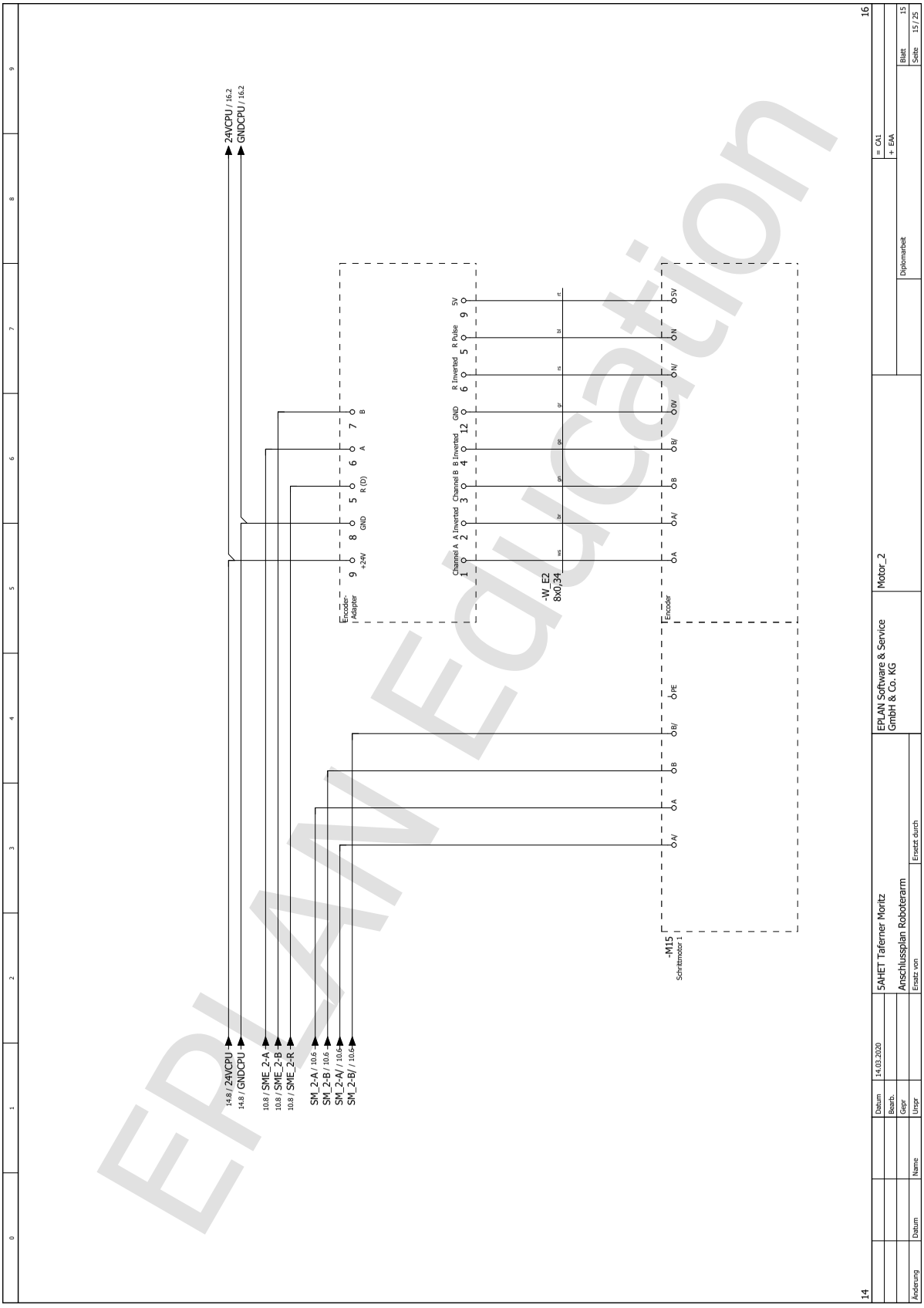

48VSM / 11.0
GNDSM / 11.0
SME_2-A / 15.2
SME_2-B / 15.2
SME_2-R / 15.2
SM_2-A / 15.2
SM_2-B / 15.2
SM_2-Ā / 15.2
SM_2-B̄ / 15.2
-W_M2
4x0,25
-A10
STP:
X20SM1446-1
X20 Schrittmotormodul, Moduversorgung 24 bis 48 VDC ±25%, 1 Motoranschluss, 5 A Dauerstrom, 10 A Spitzenstrom, 4 digitale Eingänge 24 VDC, Sink, als Inkrementalgebe
r parametrierbar, NetTime-Funktion
SS1
9.9 / 48VSM
9.9 / GNDSM
EPLAN Software & Service GmbH & Co. KG
SM_1
5AHET Taferner Moritz
Anschlussplan Roboterarm
Diplomarbeit
14.03.2020
EPLAN Education

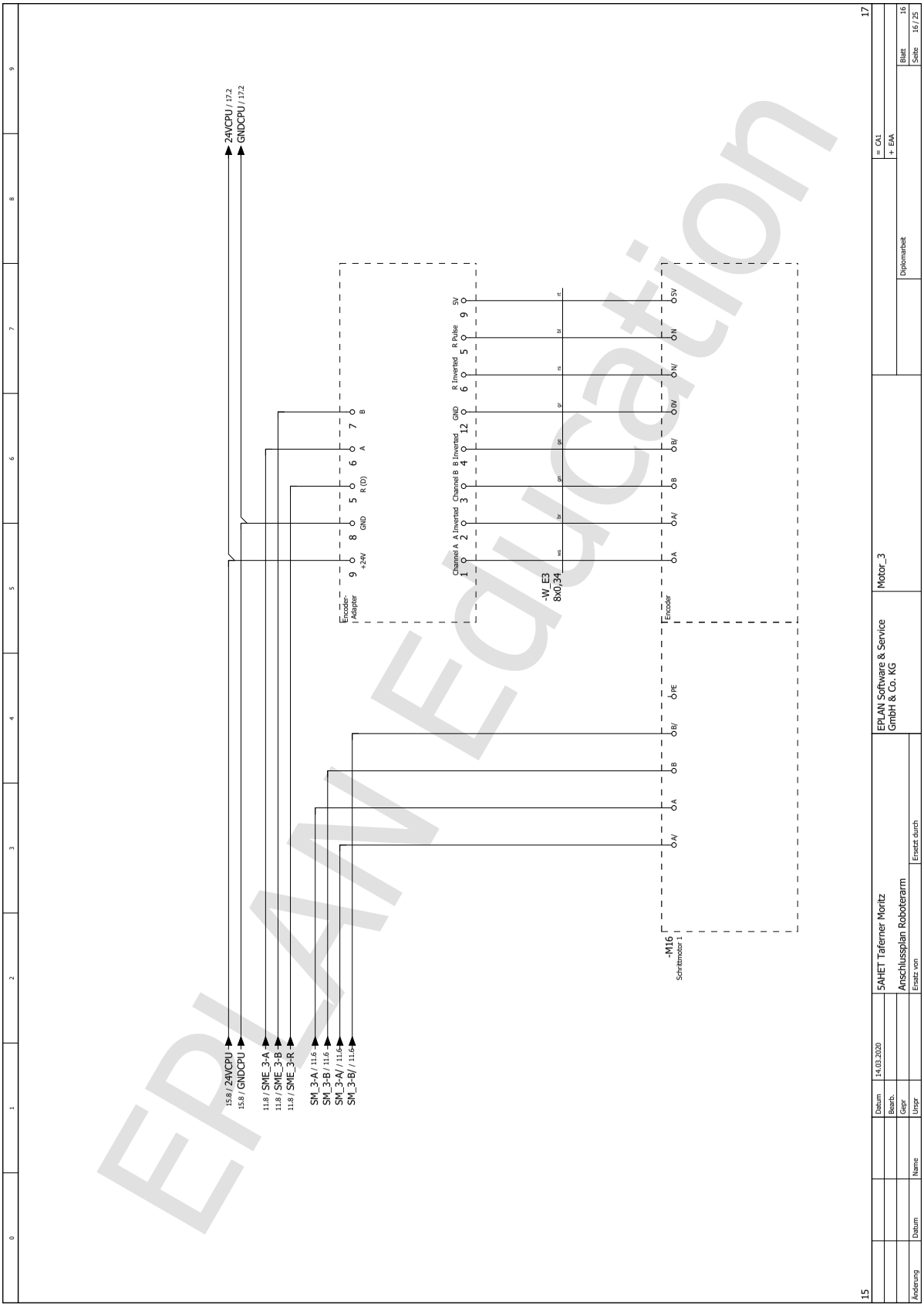

48VSM / 12.0
GNDSM / 12.0
SME_3-A / 16.2
SME_3-B / 16.2
SME_3-R / 16.2
SM_3-A / 16.2
SM_3-B / 16.2
SM_3-A/ / 16.2
SM_3-B/ / 16.2
-W_M3
4x0,25
-A11
STP:
X20SM1446-1
X20 Schrittmotormodul, Moduleversorgung 24 bis 48 VDC ±25%, 1 Motoranschluss, 5 A Dauerstrom, 10 A Spitzenstrom, 4 digitale Eingänge 24 VDC, Sink, als Inkrementalgebe
r parametrierbar, NetTime-Funktion
SS1
10.9 / 48VSM
10.9 / GNDSM
EPLAN Software & Service GmbH & Co. KG
SM_1
5AHET Taferner Moritz
Anschlussplan Roboterarm
Diplomarbeit
14.03.2020
Seite 11 / 25
EPLAN Education

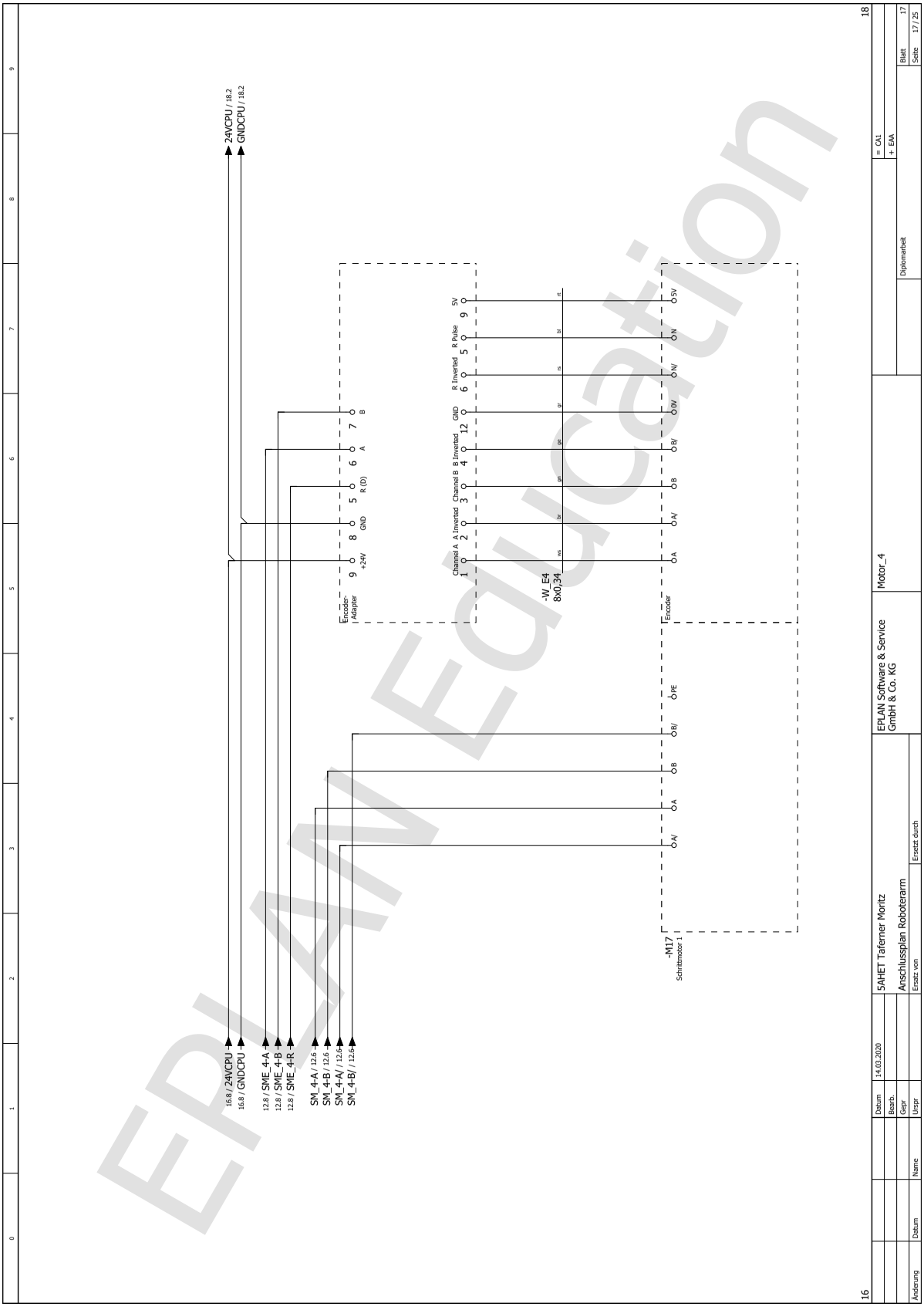

48VSM / 13.0
GNDSM / 13.0
SME_4-A / 17.2
SME_4-B / 17.2
SME_4-R / 17.2
SM_4-A / 17.2
SM_4-B / 17.2
SM_4-A/ / 17.2
SM_4-B/ / 17.2
-W_M4
4x0,25
-A12
X20SM1446-1
STP:
X20 Schrittmotormodul, Moduversorgung 24 bis 48 VDC ±25%, 1 Motoranschluss, 5 A Dauerstrom, 10 A Spitzenstrom, 4 digitale Eingänge 24 VDC, Sink, als Inkrementalgebe
r parametrierbar, NetTime-Funktion
11.9 / 48VSM
11.9 / GNDSM
EPLAN Software & Service GmbH & Co. KG
SM_1
5AHET Taferner Moritz
Anschlussplan Roboterarm
Diplomarbeit
14.03.2020
EPLAN Education

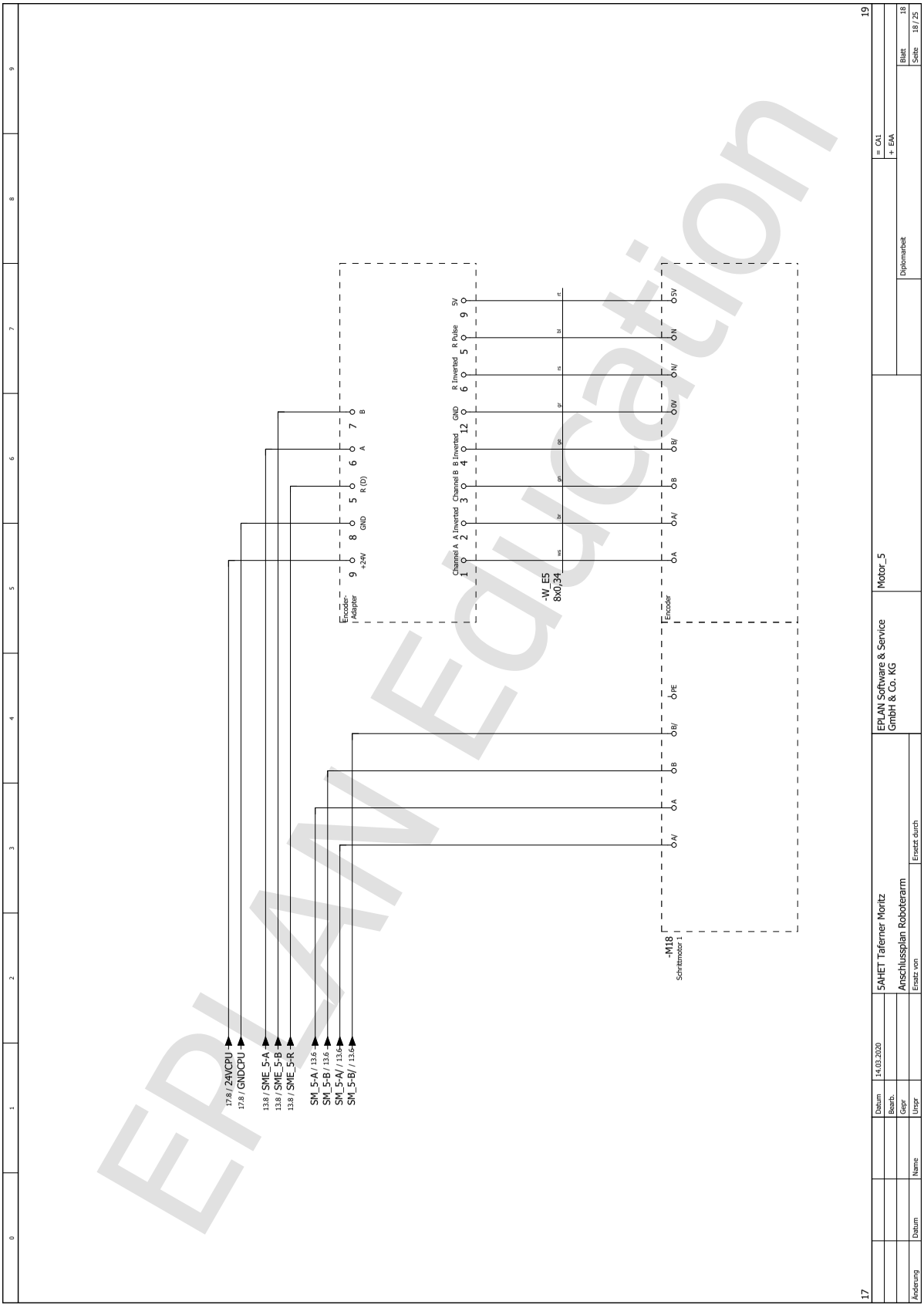

STP:
X20SM1446-1
X20 Schrittmotormodul, Modulversorgung 24 bis 48 VDC ±25%, 1 Motoranschluss, 5 A Dauerstrom, 10 A Spitzenstrom, 4 digitale Eingänge 24 VDC, Sink, als Inkrementalgebe
r parametrierbar, NetTime-Funktion
-A13
SS1
SM_5-A / 18.2
SM_5-B / 18.2
SM_5-Ā / 18.2
SM_5-B̄ / 18.2
-W_M5
4x0,25
SME_5-A / 18.2
SME_5-B / 18.2
SME_5-R / 18.2
12.9 / 48VSM
12.9 / GNDSM
EPLAN Software & Service GmbH & Co. KG
SM_1
5AHET Taferner Moritz
Anschlussplan Roboterarm
Diplomarbeit
14.03.2020
EPLAN Education

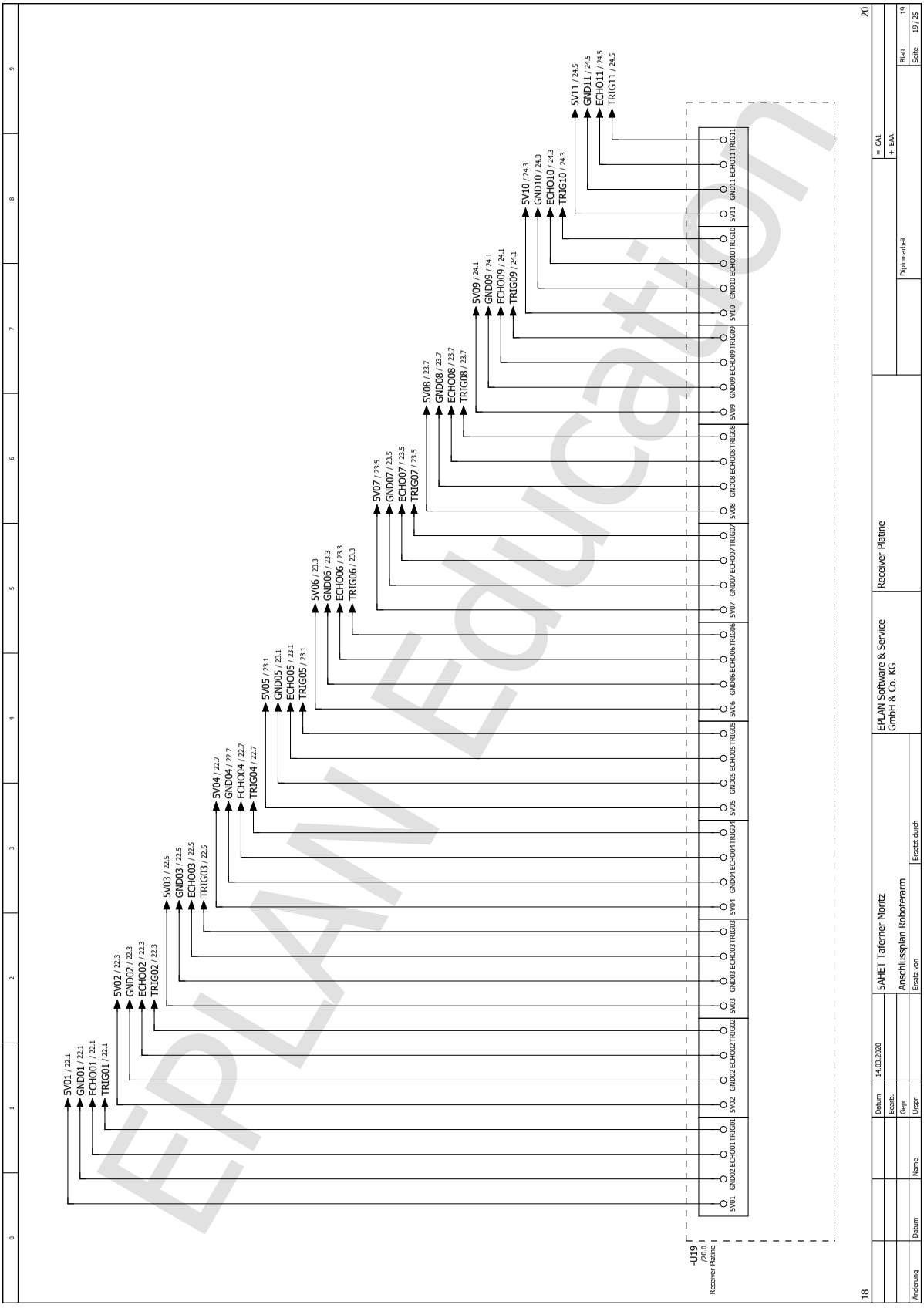

24VCPU / 15.2
GNDCPU / 15.2
4.9 / 24VCPU
4.9 / GNDCPU
9.8 / SME_1-A
9.8 / SME_1-B
9.8 / SME_1-R
SM_1-A / 9.6
SM_1-B / 9.6
SM_1-A/ / 9.6
SM_1-B/ / 9.6
-W_E1
8x0,34
-M14
Schrittmotor 1
Encoder
Adapter
Encoder
+24V
GND
R (D)
A
B
Channel A Inverted
Channel B
Channel B Inverted
GND
R Inverted
R Pulse
5V
EPLAN Software & Service GmbH & Co. KG
Motor_1
5AHET Taferner Moritz
Anschlussplan Roboterarm
14.03.2020
Diplomarbeit
Blatt 14
Seite 14 / 25
EPLAN Education

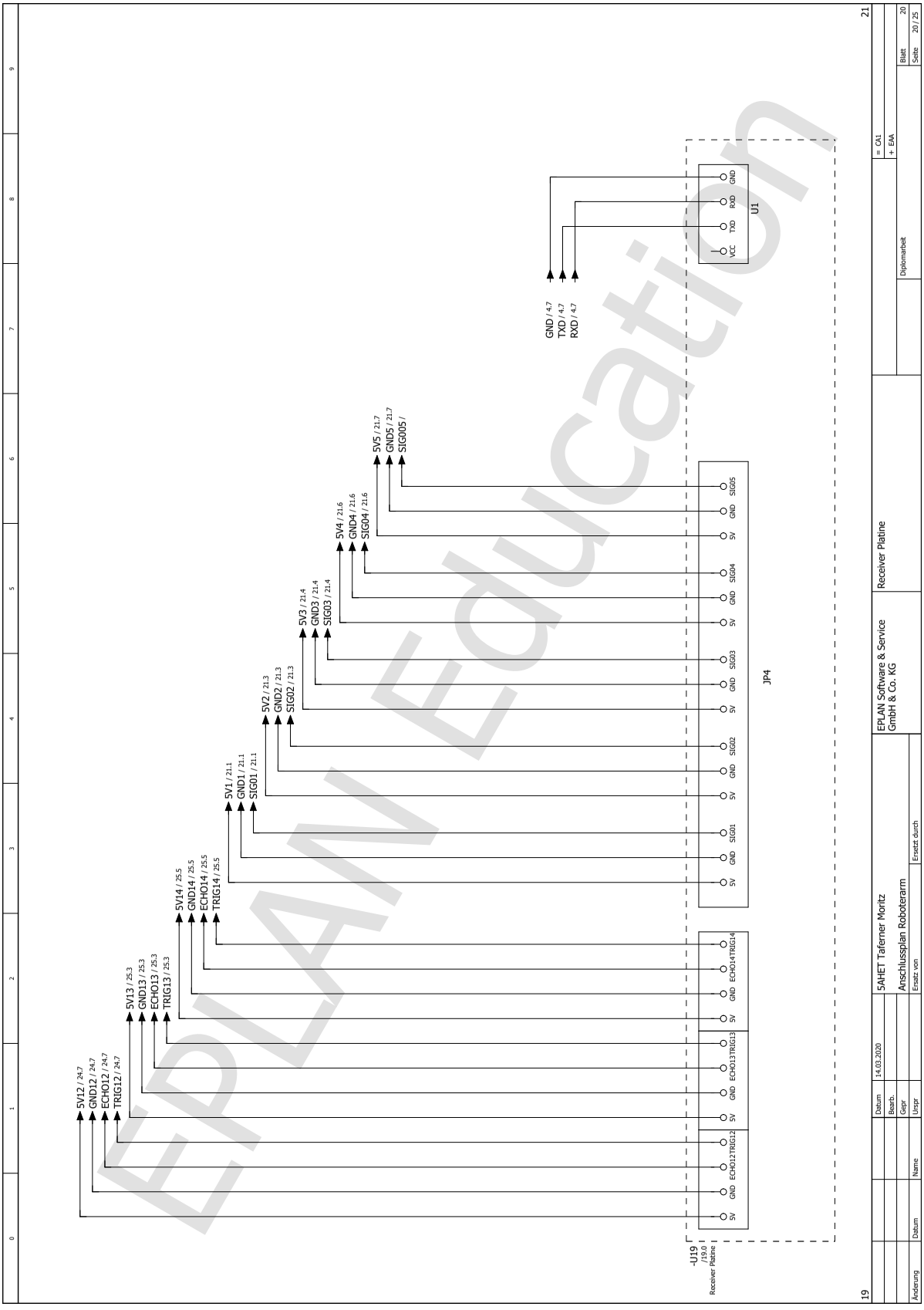


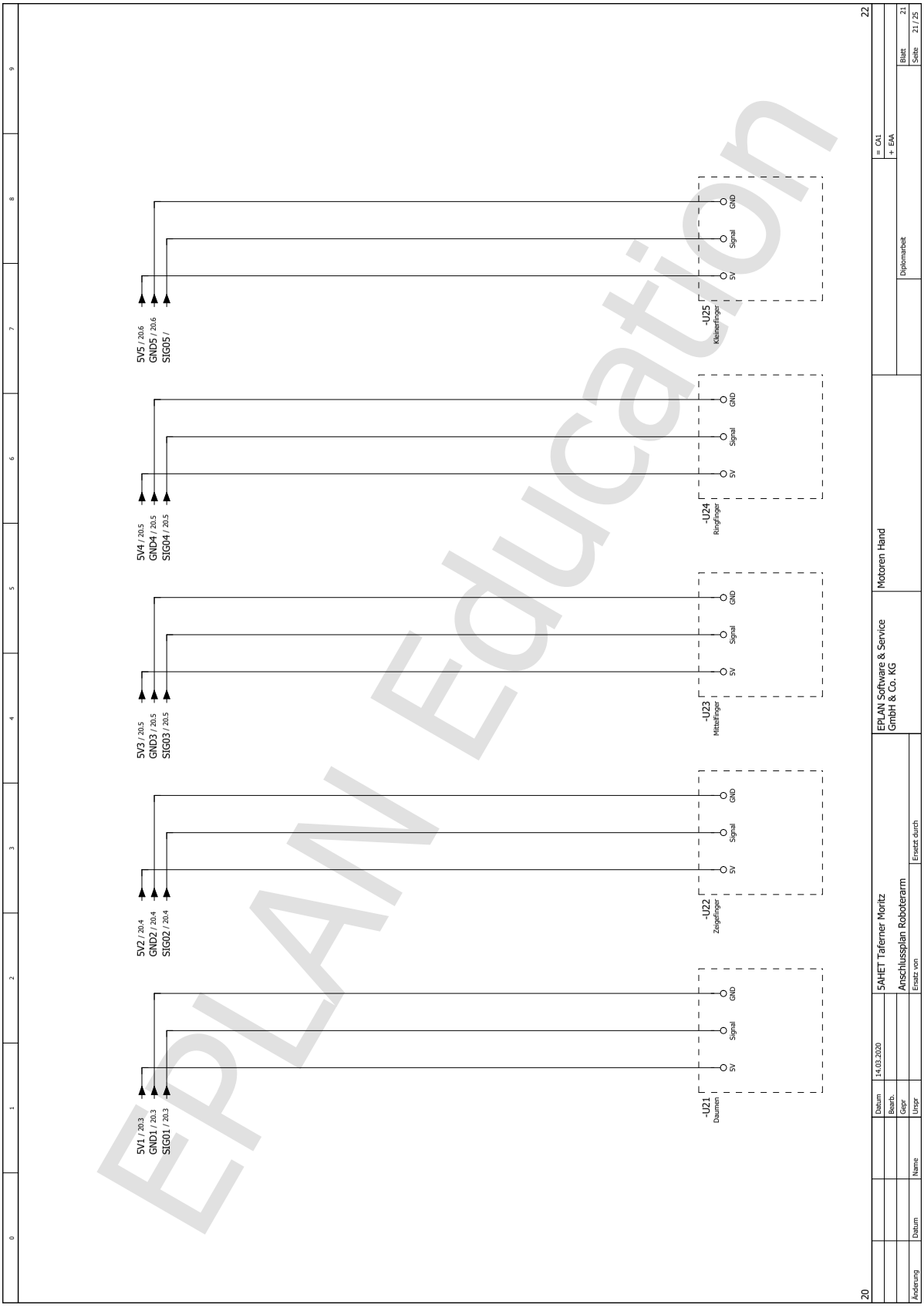

24VCPU / 17.2
GNDCPU / 17.2
15.8 / 24VCPU
15.8 / GNDCPU
11.8 / SME_3-A
11.8 / SME_3-B
11.8 / SME_3-R
SM_3-A / 11.6
SM_3-B / 11.6
SM_3-A/ / 11.6
SM_3-B/ / 11.6
-W_E3
8x0,34
-M16
Schrittmotor 1
Encoder
Adapter
Encoder
Channel A
Inverted
Channel B
Inverted
GND
R Inverted
R Pulse
5V
+24V
GND
R (D)
A
B
A/
A
B/
B
PE
A/
B/
N/
N
5V
0V
EPLAN Software & Service
GmbH & Co. KG
Motor_3
5AHET Taferner Moritz
Anschlussplan Roboterarm
Datum 14.03.2020
Bearb.
Gepr
Urspr
Ersatz von
Ersetzt durch
Änderung
Datum
Name
Diplomarbeit
Blatt 16
Seite 16 / 25
EDUCATION

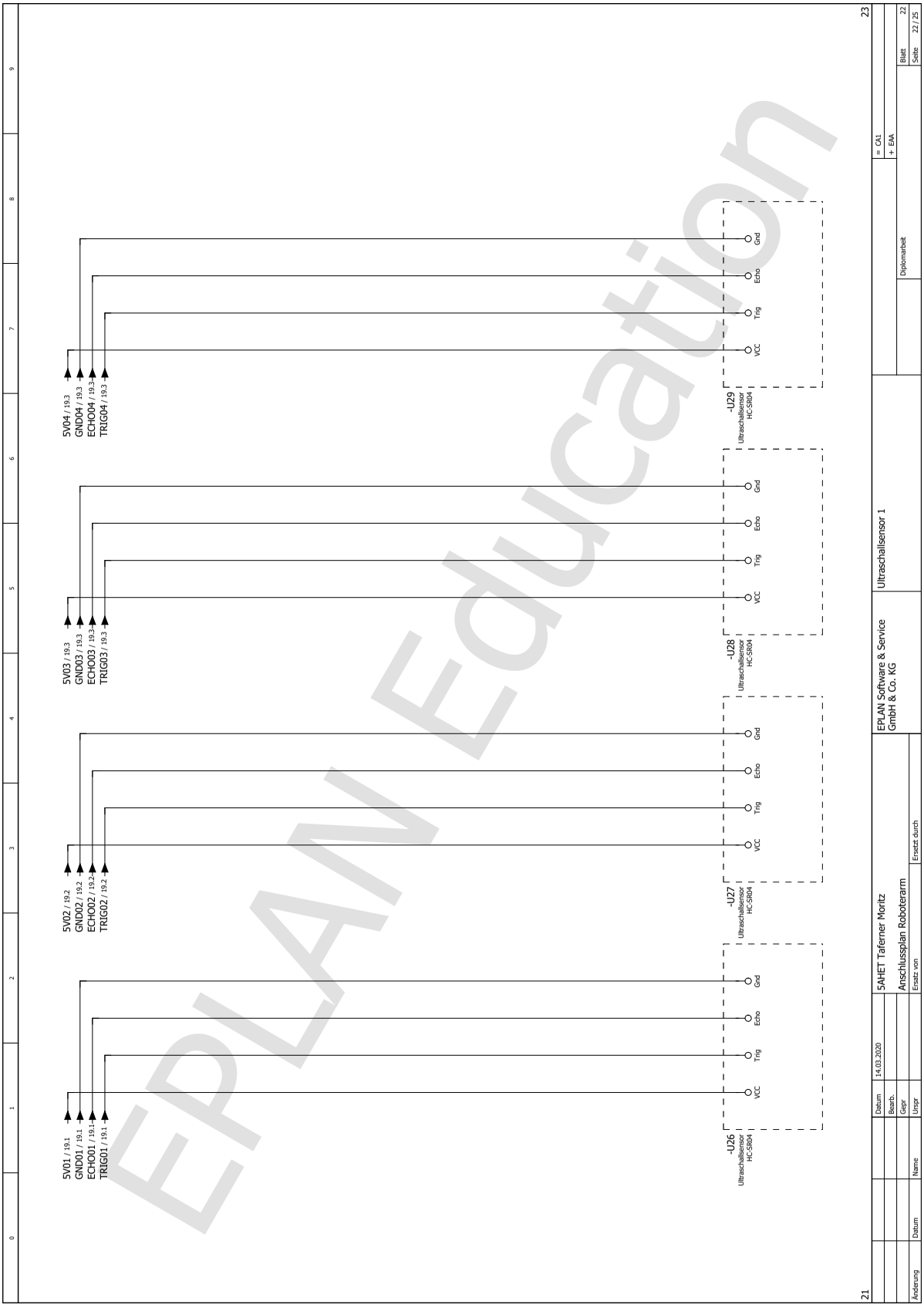


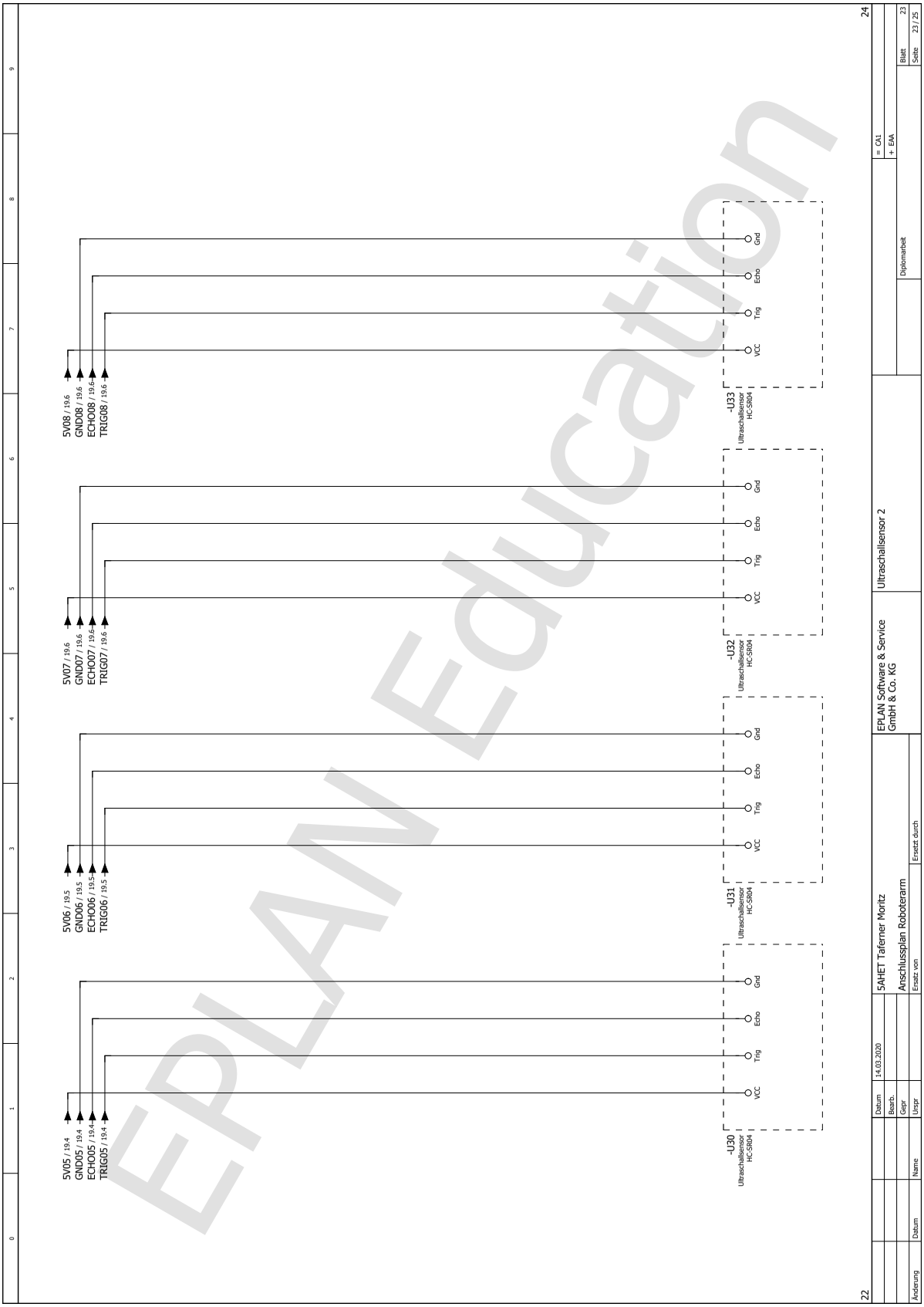

17.8 / 24VCPU
17.8 / GNDCPU
13.8 / SME_5-A
13.8 / SME_5-B
13.8 / SME_5-R
SM_5-A / 13.6
SM_5-B / 13.6
SM_5-A/ / 13.6
SM_5-B/ / 13.6
-M18
Schrittmotor 1
Encoder
-W_E5
8x0,34
Encoder
Adapter
Channel A
Channel A Inverted
Channel B
Channel B Inverted
GND
R Inverted
R Pulse
5V
+24V
GND
R (D)
A
B
EPLAN Software & Service GmbH & Co. KG
Motor_5
5AHET Taferner Moritz
Anschlussplan Roboterarm
Datum 14.03.2020
Bearb.
Gepr
Urspr
Ersatz von
Ersetzt durch
Name
Datum
Änderung
Diplomarbeit
Blatt 18
Seite 18 / 25
EPLAN Education

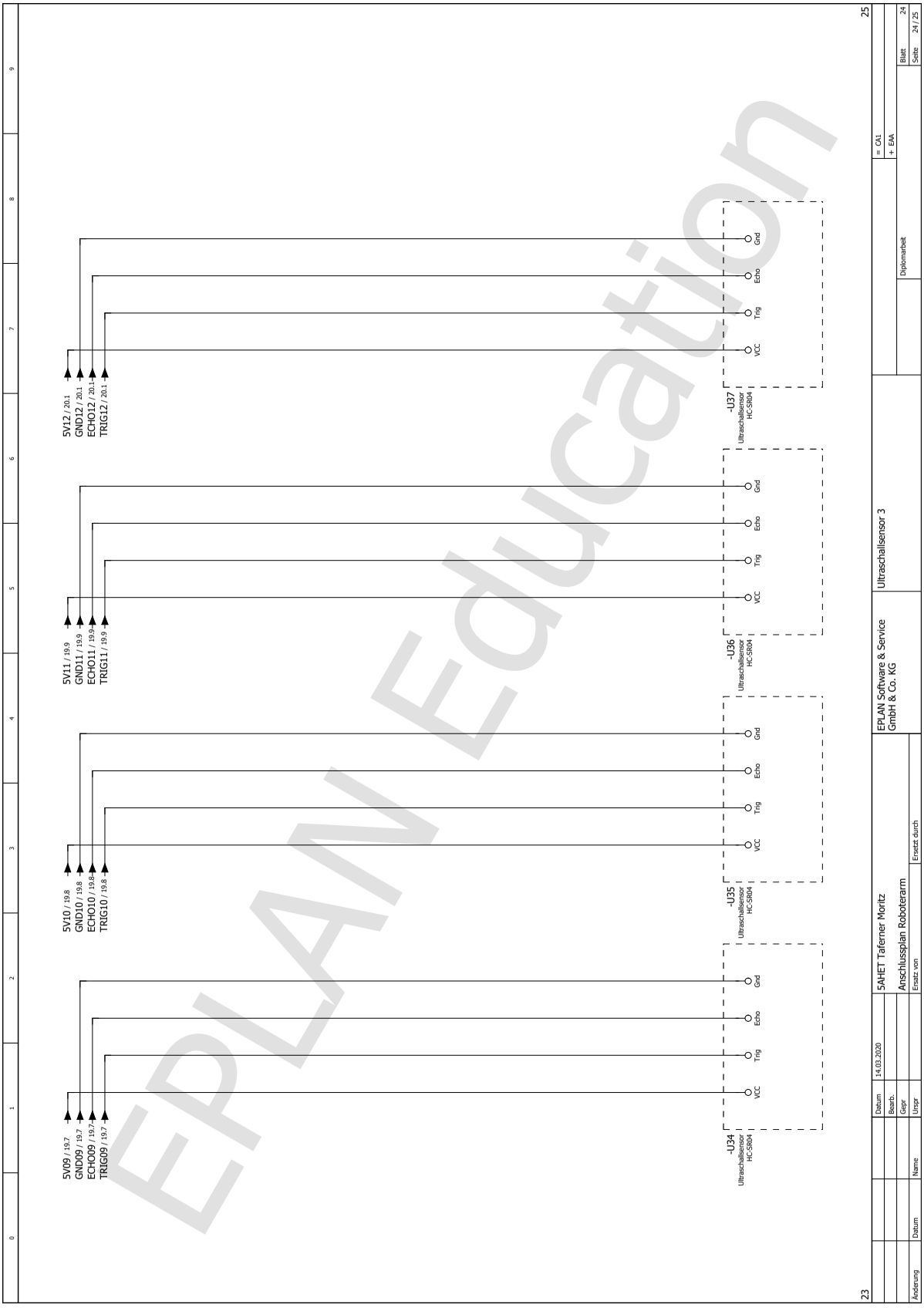

5V01 / 22.1
GND01 / 22.1
ECHO01 / 22.1
TRIG01 / 22.1
5V02 / 22.3
GND02 / 22.3
ECHO02 / 22.3
TRIG02 / 22.3
5V03 / 22.5
GND03 / 22.5
ECHO03 / 22.5
TRIG03 / 22.5
5V04 / 22.7
GND04 / 22.7
ECHO04 / 22.7
TRIG04 / 22.7
5V05 / 23.1
GND05 / 23.1
ECHO05 / 23.1
TRIG05 / 23.1
5V06 / 23.3
GND06 / 23.3
ECHO06 / 23.3
TRIG06 / 23.3
5V07 / 23.5
GND07 / 23.5
ECHO07 / 23.5
TRIG07 / 23.5
5V08 / 23.7
GND08 / 23.7
ECHO08 / 23.7
TRIG08 / 23.7
5V09 / 24.1
GND09 / 24.1
ECHO09 / 24.1
TRIG09 / 24.1
5V10 / 24.3
GND10 / 24.3
ECHO10 / 24.3
TRIG10 / 24.3
5V11 / 24.5
GND11 / 24.5
ECHO11 / 24.5
TRIG11 / 24.5
-U19
/20.0
Receiver Platine
5V01 GND01 ECHO01 TRIG01
5V02 GND02 ECHO02 TRIG02
5V03 GND03 ECHO03 TRIG03
5V04 GND04 ECHO04 TRIG04
5V05 GND05 ECHO05 TRIG05
5V06 GND06 ECHO06 TRIG06
5V07 GND07 ECHO07 TRIG07
5V08 GND08 ECHO08 TRIG08
5V09 GND09 ECHO09 TRIG09
5V10 GND10 ECHO10 TRIG10
5V11 GND11 ECHO11 TRIG11
EPLAN Software & Service GmbH & Co. KG
Receiver Platine
5AHET Taferner Moritz
Anschlussplan Roboterarm
Diplomarbeit
Datum 14.03.2020
Bearb.
Gepr
Urspr
Ersatz von
Ersetzt durch
Name
Datum
Änderung
= CA1
+ EAA
Blatt 19
Seite 19 / 25
18
20
EPLAN Education

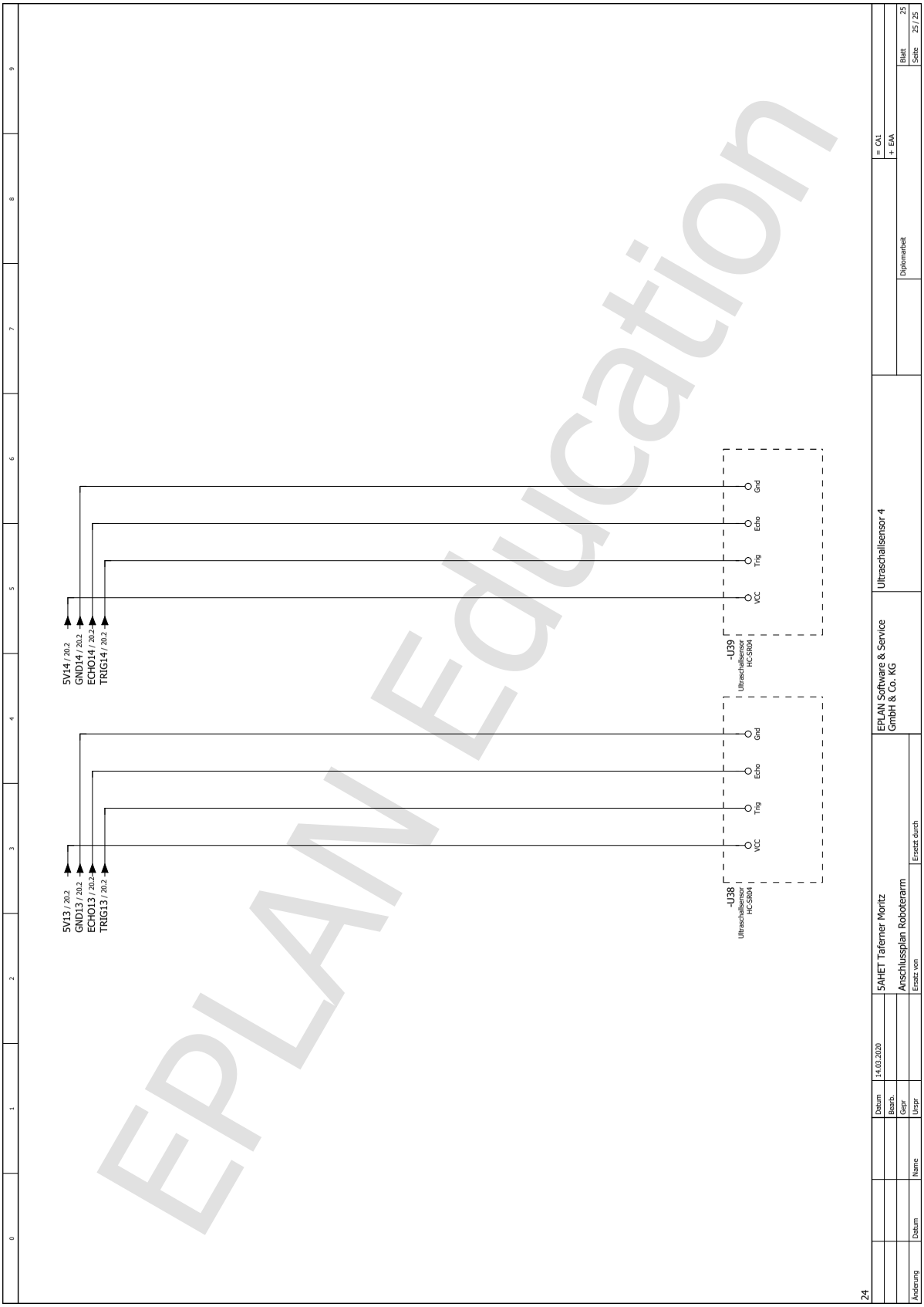

5V12 / 24.7
GND12 / 24.7
ECHO12 / 24.7
TRIG12 / 24.7
5V13 / 25.3
GND13 / 25.3
ECHO13 / 25.3
TRIG13 / 25.3
5V14 / 25.5
GND14 / 25.5
ECHO14 / 25.5
TRIG14 / 25.5
5V1 / 21.1
GND1 / 21.1
SIG01 / 21.1
5V2 / 21.3
GND2 / 21.3
SIG02 / 21.3
5V3 / 21.4
GND3 / 21.4
SIG03 / 21.4
5V4 / 21.6
GND4 / 21.6
SIG04 / 21.6
5V5 / 21.7
GND5 / 21.7
SIG005 /
GND / 4.7
TXD / 4.7
RXD / 4.7
-U19
/19.0
Receiver Platine
JP4
U1
VCC TXD RXD GND
5V GND ECHO12 TRIG12
5V GND ECHO13 TRIG13
5V GND ECHO14 TRIG14
5V GND SIG01
5V GND SIG02
5V GND SIG03
5V GND SIG04
5V GND SIG05
EPLAN Software & Service GmbH & Co. KG
Receiver Platine
5AHET Taferner Moritz
Anschlussplan Roboterarm
Datum 14.03.2020
Bearb.
Gepr
Urspr
Ersatz von
Ersetzt durch
Name
Datum
Änderung
= CA1
+ EAA
Diplomarbeit
Blatt 20
Seite 20 / 25
EPLAN Education

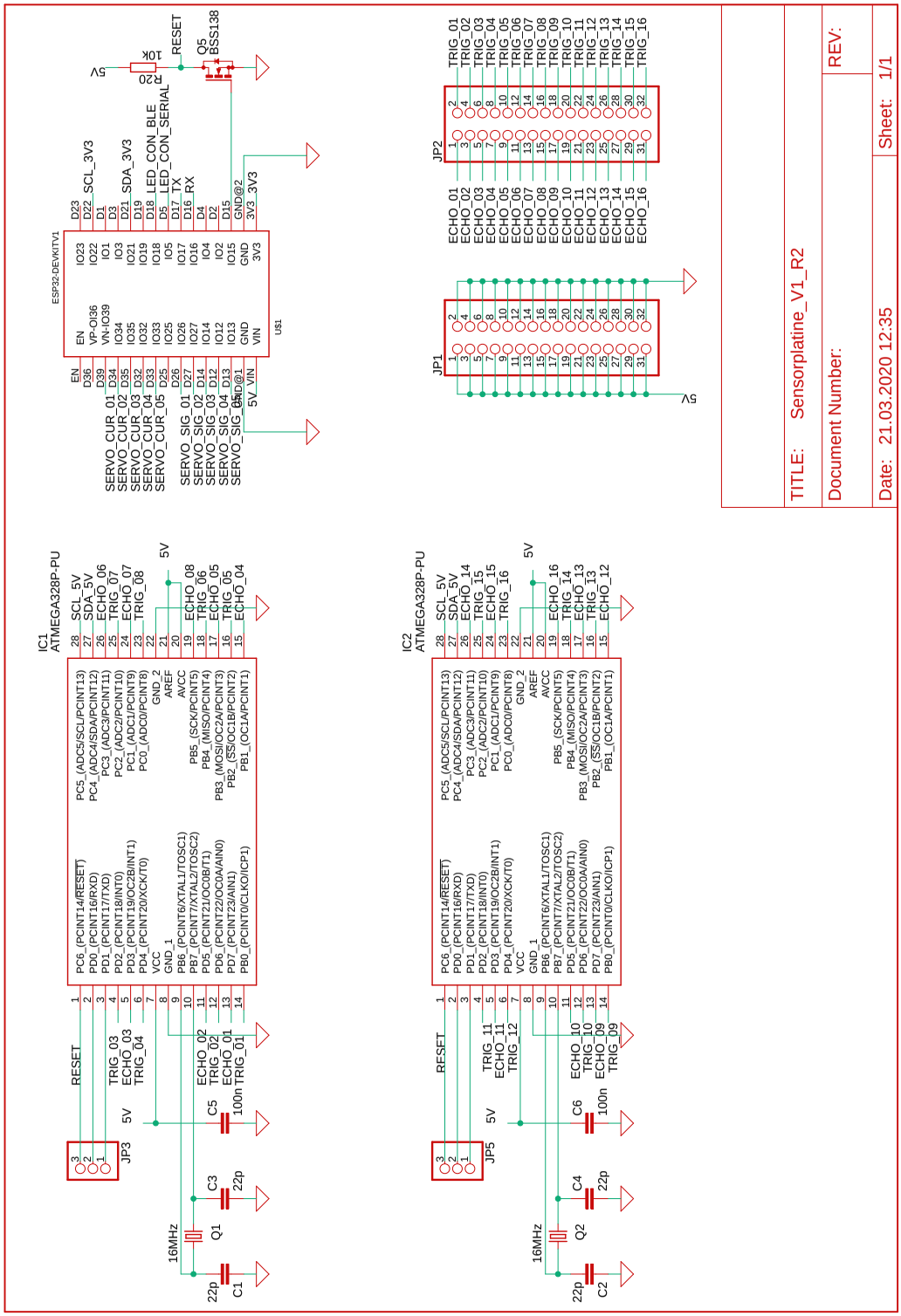

5V1 / 20.3
GND1 / 20.3
SIG01 / 20.3
5V2 / 20.4
GND2 / 20.4
SIG02 / 20.4
5V3 / 20.5
GND3 / 20.5
SIG03 / 20.5
5V4 / 20.5
GND4 / 20.5
SIG04 / 20.5
5V5 / 20.6
GND5 / 20.6
SIG05 /
-U21
Daumen
-U22
Zeigefinger
-U23
Mittelfinger
-U24
Ringfinger
-U25
Kleinerfinger
5V
Signal
GND
5AHET Taferner Moritz
Anschlussplan Roboterarm
EPLAN Software & Service GmbH & Co. KG
Motoren Hand
Diplomarbeit
Datum 14.03.2020
Bearb.
Gepr
Urspr
Name
Ersatz von
Ersetzt durch
Änderung
Datum
Blatt 21
Seite 21 / 25
EPLAN Education

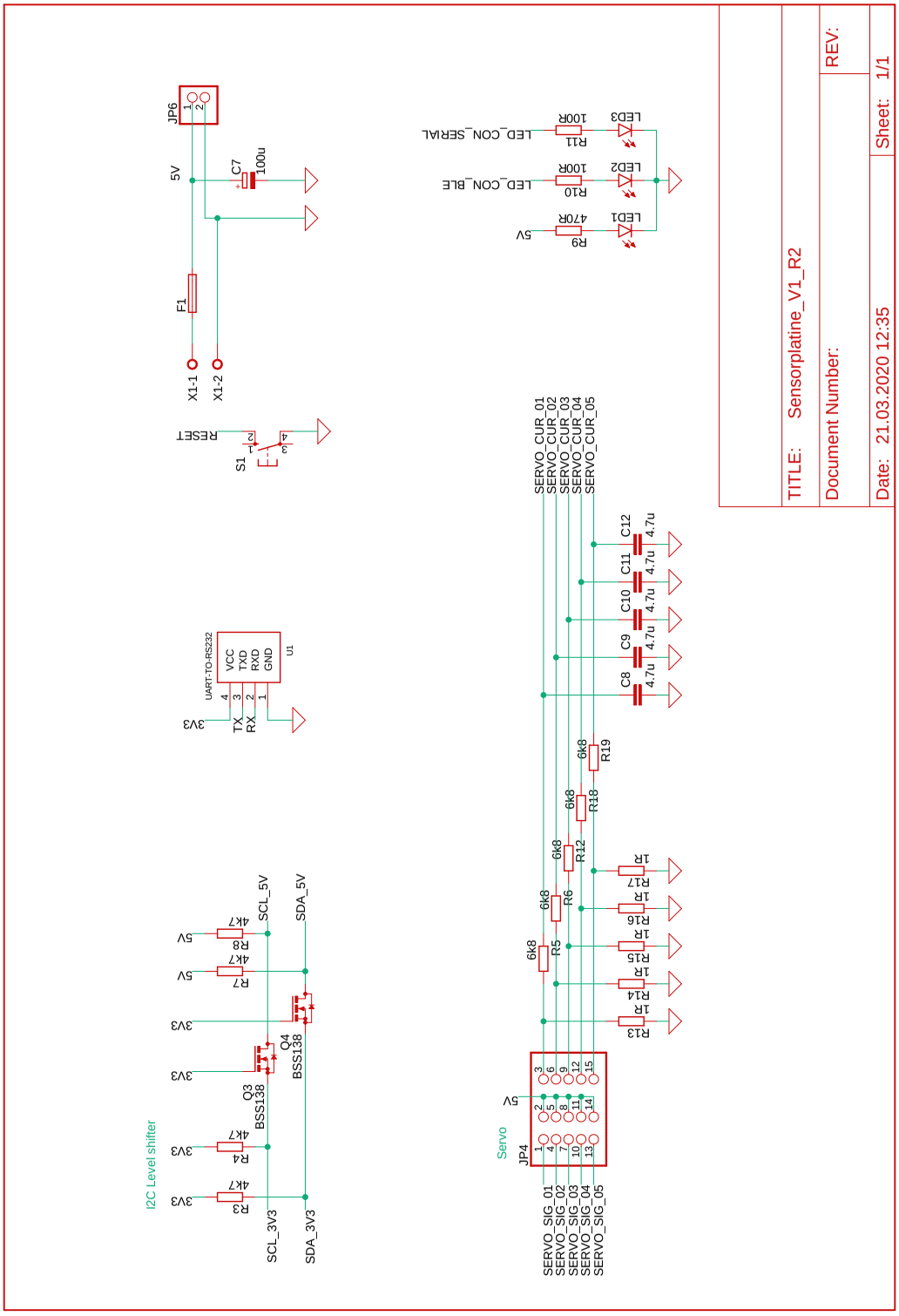

5V01 / 19.1
GND01 / 19.1
ECHO01 / 19.1
TRIG01 / 19.1
5V02 / 19.2
GND02 / 19.2
ECHO02 / 19.2
TRIG02 / 19.2
5V03 / 19.3
GND03 / 19.3
ECHO03 / 19.3
TRIG03 / 19.3
5V04 / 19.3
GND04 / 19.3
ECHO04 / 19.3
TRIG04 / 19.3
-U26
Ultraschallsensor
HC-SR04
-U27
Ultraschallsensor
HC-SR04
-U28
Ultraschallsensor
HC-SR04
-U29
Ultraschallsensor
HC-SR04
Gnd
Echo
Trig
VCC
EPLAN Software & Service GmbH & Co. KG
Ultraschallsensor 1
5AHET Taferner Moritz
Anschlussplan Roboterarm
14.03.2020
Diplomarbeit
EPLAN Education


5V05 / 19.4
GND05 / 19.4
ECHO05 / 19.4
TRIG05 / 19.4
5V06 / 19.5
GND06 / 19.5
ECHO06 / 19.5
TRIG06 / 19.5
5V07 / 19.6
GND07 / 19.6
ECHO07 / 19.6
TRIG07 / 19.6
5V08 / 19.6
GND08 / 19.6
ECHO08 / 19.6
TRIG08 / 19.6
-U30
-U31
-U32
-U33
Ultraschallsensor HC-SR04
Gnd
Echo
Trig
VCC
EPLAN Software & Service GmbH & Co. KG
Ultraschallsensor 2
5AHET Taferner Moritz
Anschlussplan Roboterarm
14.03.2020
Diplomarbeit
EPLAN Education

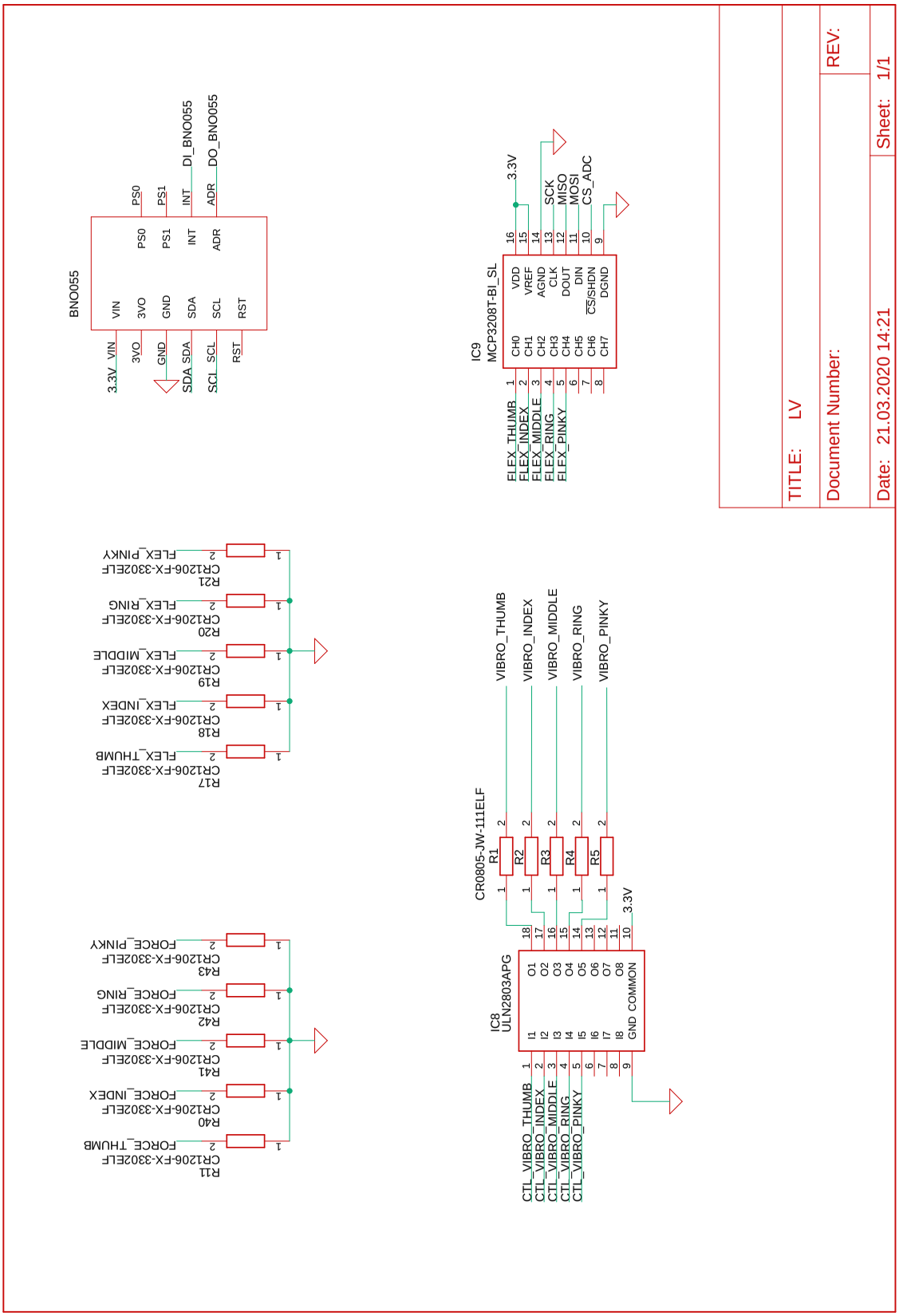

5V09 / 19.7
GND09 / 19.7
ECHO09 / 19.7
TRIG09 / 19.7
-U34
Ultraschallsensor
HC-SR04
Gnd
Echo
Trig
VCC
5V10 / 19.8
GND10 / 19.8
ECHO10 / 19.8
TRIG10 / 19.8
-U35
5V11 / 19.9
GND11 / 19.9
ECHO11 / 19.9
TRIG11 / 19.9
-U36
5V12 / 20.1
GND12 / 20.1
ECHO12 / 20.1
TRIG12 / 20.1
-U37
EPLAN Software & Service GmbH & Co. KG
Ultraschallsensor 3
5AHET Taferner Moritz
Anschlussplan Roboterarm
14.03.2020
Diplomarbeit
Blatt 24
Seite 24 / 25
EPLAN Education

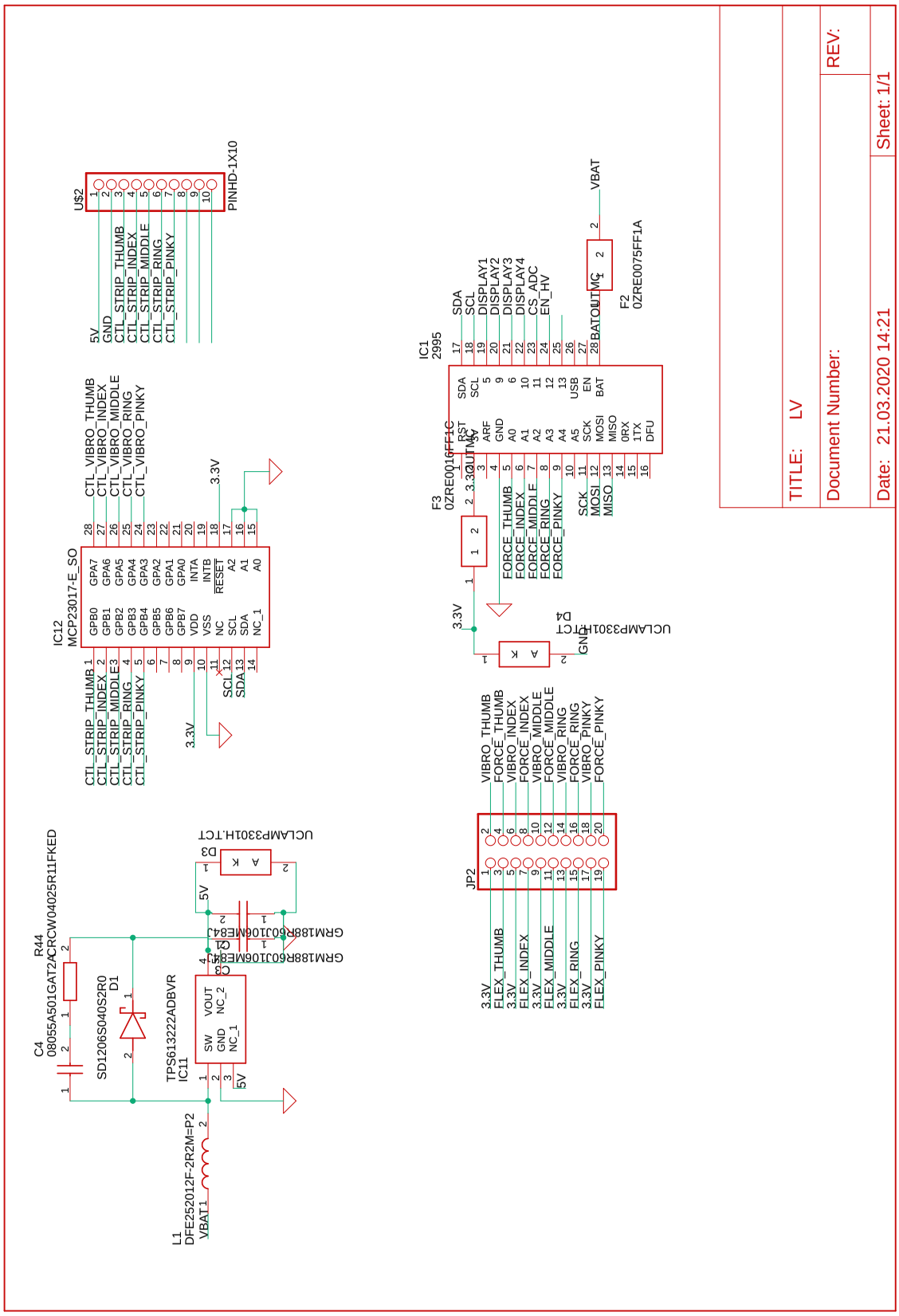

5V14 / 20.2
GND14 / 20.2
ECHO14 / 20.2
TRIG14 / 20.2
5V13 / 20.2
GND13 / 20.2
ECHO13 / 20.2
TRIG13 / 20.2
-U39
Ultraschallsensor
HC-SR04
-U38
Ultraschallsensor
HC-SR04
Gnd
Echo
Trig
VCC
24
EPLAN Software & Service GmbH & Co. KG
Ultraschallsensor 4
5AHET Taferner Moritz
Anschlussplan Roboterarm
Diplomarbeit
Datum 14.03.2020
Bearb.
Gepr.
Urspr.
Ersatz von
Ersetzt durch
Änderung
Datum
Name
Blatt 25
Seite 25 / 25
EPLAN Education

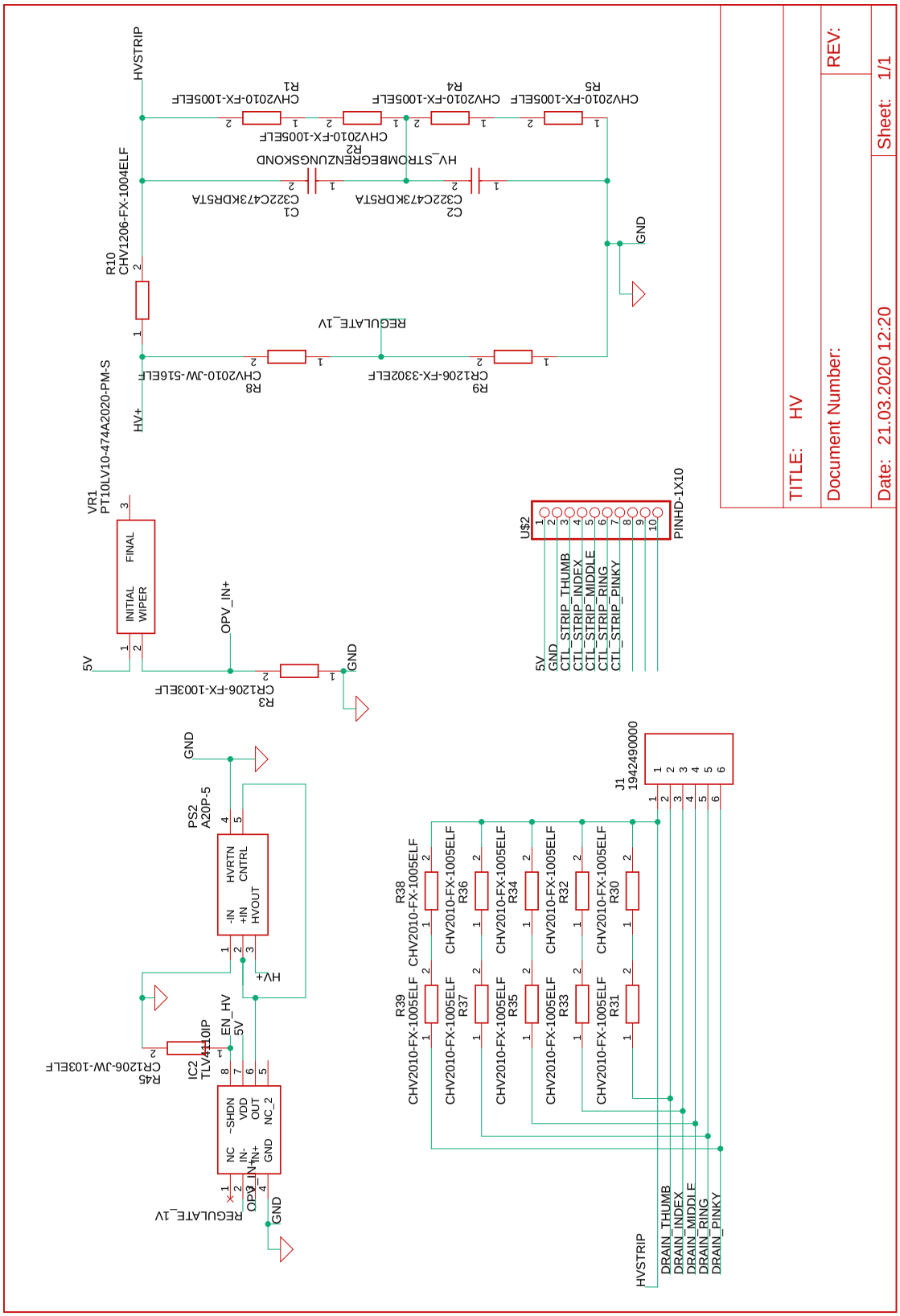

TITLE: Sensorplatine_V1_R2
Document Number:
REV:
Date: 21.03.2020 12:35
Sheet: 1/1
ESP32-DEVKITV1
IC1
ATMEGA328P-PU
IC2
ATMEGA328P-PU
JP1
JP2
JP3
JP5
Q1
Q2
Q5
BSS138
16MHz
5V
RESET
SCL_5V
SDA_5V
SCL_3V3
SDA_3V3
LED_CON_BLE
LED_CON_SERIAL
SERVO_CUR_01
SERVO_CUR_02
SERVO_CUR_03
SERVO_CUR_04
SERVO_CUR_05
SERVO_SIG_01
SERVO_SIG_02
SERVO_SIG_03
SERVO_SIG_04
SERVO_SIG_05
ECHO_01
ECHO_02
ECHO_03
ECHO_04
ECHO_05
ECHO_06
ECHO_07
ECHO_08
ECHO_09
ECHO_10
ECHO_11
ECHO_12
ECHO_13
ECHO_14
ECHO_15
ECHO_16
TRIG_01
TRIG_02
TRIG_03
TRIG_04
TRIG_05
TRIG_06
TRIG_07
TRIG_08
TRIG_09
TRIG_10
TRIG_11
TRIG_12
TRIG_13
TRIG_14
TRIG_15
TRIG_16

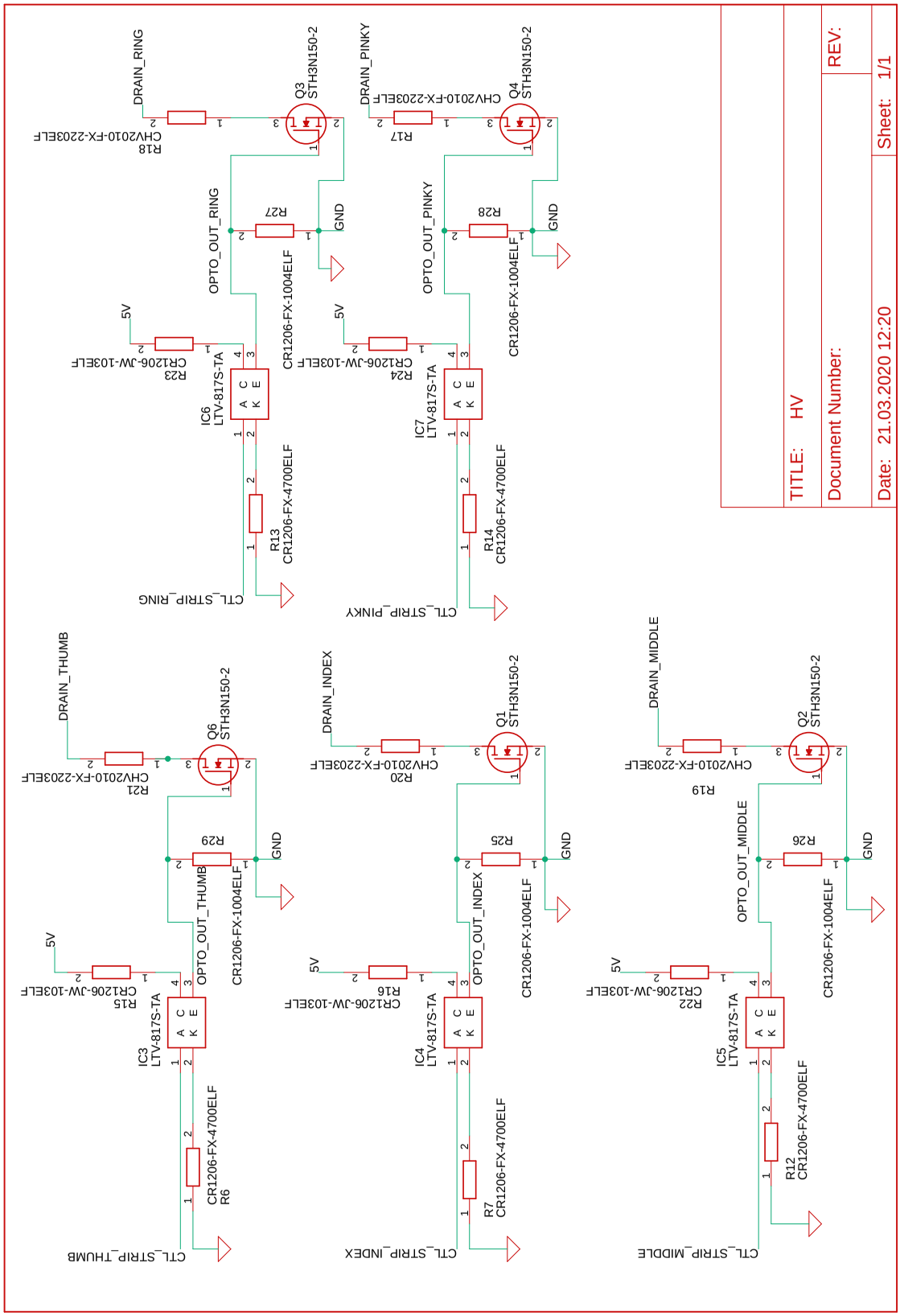

TITLE: Sensorplatine_V1_R2
Document Number:
REV:
Date: 21.03.2020 12:35
Sheet: 1/1
JP6
5V
C7
100u
F1
X1-1
X1-2
RESET
S1
LED_CON_SERIAL
LED_CON_BLE
R11
R10
R9
100R
470R
LED3
LED2
LED1
SERVO_CUR_01
SERVO_CUR_02
SERVO_CUR_03
SERVO_CUR_04
SERVO_CUR_05
C12
C11
C10
C9
C8
4.7u
R19
R18
R12
R6
R5
6k8
R17
R16
R15
R14
R13
1R
JP4
Servo
SERVO_SIG_01
SERVO_SIG_02
SERVO_SIG_03
SERVO_SIG_04
SERVO_SIG_05
U1
UART-TO-RS232
VCC
TXD
RXD
GND
TX
RX
3V3
I2C Level shifter
SCL_5V
SDA_5V
SCL_3V3
SDA_3V3
R8
R7
R4
R3
4K7
Q4
Q3
BSS138

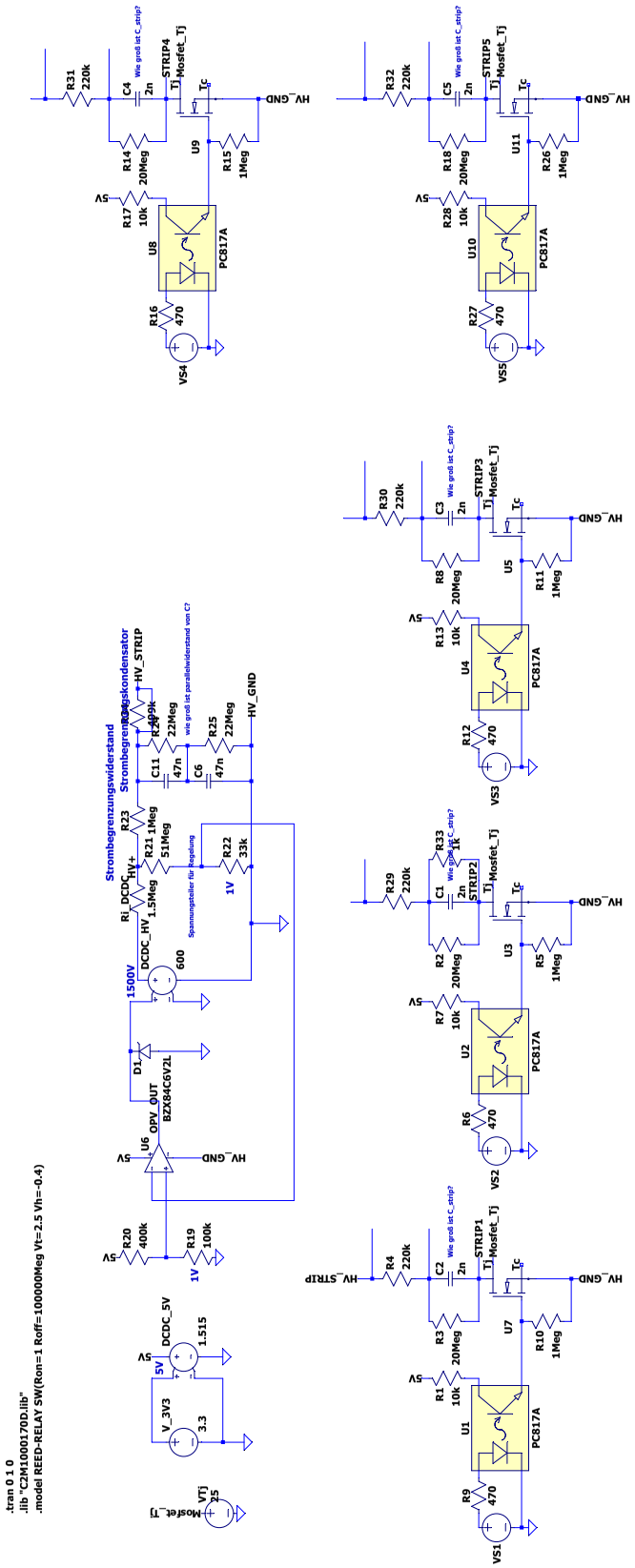
2 Fernsteuergerät




U$2
PINHD-1X10
5V
GND
CTL_STRIP_THUMB
CTL_STRIP_INDEX
CTL_STRIP_MIDDLE
CTL_STRIP_RING
CTL_STRIP_PINKY
IC12
MCP23017-E_SO
CTL_VIBRO_THUMB
CTL_VIBRO_INDEX
CTL_VIBRO_MIDDLE
CTL_VIBRO_RING
CTL_VIBRO_PINKY
3.3V
IC1
2995
F3
0ZRE0016FF1C
F2
0ZRE0075FF1A
VBAT
SDA
SCL
DISPLAY1
DISPLAY2
DISPLAY3
DISPLAY4
CS_ADC
EN_HV
FORCE_THUMB
FORCE_INDEX
FORCE_MIDDLE
FORCE_RING
FORCE_PINKY
SCK
MOSI
MISO
D4
UCLAMP3301H.TCT
JP2
VIBRO_THUMB
FORCE_THUMB
VIBRO_INDEX
FORCE_INDEX
VIBRO_MIDDLE
FORCE_MIDDLE
VIBRO_RING
FORCE_RING
VIBRO_PINKY
FORCE_PINKY
FLEX_THUMB
FLEX_INDEX
FLEX_MIDDLE
FLEX_RING
FLEX_PINKY
R44
ACRCW04025R11FKED
C4
08055A501GAT2A
D1
SD1206S040S2R0
IC11
TPS613222ADBVR
L1
DFE252012F-2R2M=P2
VBAT_1
C3
GRM188R60J106ME84J
D3
TITLE: LV
Document Number:
REV:
Date: 21.03.2020 14:21
Sheet: 1/1


HVSTRIP
CHV2010-FX-1005ELF
R1
CHV2010-FX-1005ELF
R4
CHV2010-FX-1005ELF
R5
CHV2010-FX-1005ELF
R2
HV_STROMBEGRENZUNGSKOND
C1
C3222C473KDR5TA
C2
C3222C473KDR5TA
GND
R10
CHV1206-FX-1004ELF
REGULATE_1V
HV+
CHV2010-JW-516ELF
R8
CR1206-FX-3302ELF
R9
VR1
PT10LV10-474A2020-PM-S
INITIAL
WIPER
FINAL
5V
OPV_IN+
CR1206-FX-1003ELF
R3
GND
US2
PINHD-1X10
5V
GND
CTL_STRIP_THUMB
CTL_STRIP_INDEX
CTL_STRIP_MIDDLE
CTL_STRIP_RING
CTL_STRIP_PINKY
GND
PS2
A20P-5
HVRTN
CNTRL
-IN
+IN
HVOUT
HV+
EN_HV
5V
CR1206-JW-103ELF
R45
IC2
TLV4110IP
~SHDN
VDD
OUT
NC_2
NC
IN-
IN+
GND
OPV_IN+
REGULATE_1V
GND
J1
1942490000
R38
CHV2010-FX-1005ELF
R36
CHV2010-FX-1005ELF
R34
CHV2010-FX-1005ELF
R32
CHV2010-FX-1005ELF
R30
CHV2010-FX-1005ELF
R39
CHV2010-FX-1005ELF
R37
CHV2010-FX-1005ELF
R35
CHV2010-FX-1005ELF
R33
CHV2010-FX-1005ELF
R31
CHV2010-FX-1005ELF
HVSTRIP
DRAIN_THUMB
DRAIN_INDEX
DRAIN_MIDDLE
DRAIN_RING
DRAIN_PINKY
TITLE: HV
Document Number:
REV:
Date: 21.03.2020 12:20
Sheet: 1/1


DRAIN_RING
DRAIN_PINKY
DRAIN_THUMB
DRAIN_INDEX
DRAIN_MIDDLE
Q3 STH3N150-2
Q4 STH3N150-2
Q6 STH3N150-2
Q1 STH3N150-2
Q2 STH3N150-2
R18 CHV2010-FX-2203ELF
R17 CHV2010-FX-2203ELF
R21 CHV2010-FX-2203ELF
R20 CHV2010-FX-2203ELF
R19 CHV2010-FX-2203ELF
OPTO_OUT_RING
OPTO_OUT_PINKY
OPTO_OUT_THUMB
OPTO_OUT_INDEX
OPTO_OUT_MIDDLE
R27
R28
R29
R25
R26
GND
CR1206-FX-1004ELF
5V
R23 CR1206-JW-103ELF
R24 CR1206-JW-103ELF
R15 CR1206-JW-103ELF
R16 CR1206-JW-103ELF
R22 CR1206-JW-103ELF
IC6 LTV-817S-TA
IC7 LTV-817S-TA
IC3 LTV-817S-TA
IC4 LTV-817S-TA
IC5 LTV-817S-TA
A C K E
R13 CR1206-FX-4700ELF
R14 CR1206-FX-4700ELF
R6 CR1206-FX-4700ELF
R7 CR1206-FX-4700ELF
R12 CR1206-FX-4700ELF
CTL_STRIP_RING
CTL_STRIP_PINKY
CTL_STRIP_THUMB
CTL_STRIP_INDEX
CTL_STRIP_MIDDLE
TITLE: HV
Document Number:
REV:
Date: 21.03.2020 12:20
Sheet: 1/1


.tran 0 1 0
.lib "C2M1000170D.lib"
.model REED-RELAY SW(Ron=1 Roff=100000Meg Vt=2.5 Vh=-0.4)
HV_GND
HV_STRIP
Stromgrenzwiderstand
Stromgrenzkondensator
Spannungsteiler für Regelung
wie groß ist C_strip?
PC817A
BZX84C6V2L
DCDC_5V
V_3V3
Mosfet_Tj
1500V
1V
5V

Literatur

[1] Jürgen Adamek und Volker Piwek. *Additive Fertigung - 3D-Druck: Stand der Technik, Anwendungsempfehlungen und aktuelle Entwicklungen*. ger. Fachbuch. 2019.

[2] Mats Andersson. „Use case possibilities with Bluetooth low energy in IoT applications“. In: (5. Dez. 2014). URL: https://www.spezial.com/sites/default/files/bluetoothlowenergy-iot-applications_whitepaper_ubx-14054580.pdf (abgerufen am 16. 03. 2020).

[3] *Anschlussbelegung High-Speed-CAN*. URL: https://www.peak-system.com/produktcd/Pdf/Deutsch/PCAN-USB_UserMan_deu.pdf (abgerufen am 04. 03. 2020).

[4] *Arduino-Referenz. map()*. URL: https://www.arduino.cc/reference/de/language/functions/math/map/ (abgerufen am 18. 03. 2020).

[5] B&R Industrial Automation GmbH. *B&R Help Explorer - Automation Help 4.7.2.96*. (Abgerufen am 26. 03. 2020).

[6] B&R Industrial Automation GmbH. *Datenblatt_8LVA2_de_V1.2*. URL: https://www.br-automation.com/index.php?eID=dumpFile&t=f&f=2%3A%2FBRP444000000000000504582&token=a7d820fe0f3b5c3a336c4ea0cd7e1e7ce2684147 (abgerufen am 16. 03. 2020).

[7] B&R Industrial Automation GmbH. *MA5VDCECAD-E*. URL: https://download.br-automation.com/BRP4440000000000000035801/MA5VDECAD_E_V20.pdf?px-hash=266daf97c8924fa5b1a8ea3424905f75&px-time=1584128744 (abgerufen am 12. 03. 2020).

[8] B&R Industrial Automation GmbH. *MAACPMICRO1-GER_V1.25*. URL: https://www.br-automation.com/index.php?eID=dumpFile&t=f&f=2%3A%2FBRP444000000000000626587&token=24281ad9c5e912c7fe2daf2d56024788aee00e5e (abgerufen am 17. 03. 2020).

[9] Heinz-Werner Beckmann u. a. *Friedrich Tabellenbuch Elektrotechnik/Elektronik*. ger. 585., überarbeitete und aktualisierte Auflage. Fachbuch. 2015.

[10] Beurer. *Beurer Bauchmuskel-Gürtel EM 37*. URL: https://www.beurer.com/web/de/produkte/active/schmerztherapie-muskelstimulation/bauchmuskel-guertel/em-37.php (abgerufen am 05. 03. 2020).

[11] Bluetooth Core Specification Working Group. *Bluetooth Core Specification*. Version v5.2. 31. Dez. 2019.

[12] Bosch Sensortec. *BNO055. Intelligent 9-axis absolute orientation sensor*. URL: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bno055-ds000.pdf (abgerufen am 20. 03. 2020).

[13] Bourns Inc. *CHV Series - Thick Film High Voltage Chip Resistors*. URL: https://docs.rs-online.com/fd69/0900766b813ff00a.pdf (abgerufen am 02. 03. 2020).

[14] CAPLINQ Corporation. *PIT0.5N | 0.5-mil Polyimide (Kapton) Film*. URL: https://www.caplinq.com/0.5-mil-polyimide-kapton-film-no-adhesive-pit0.5n-series.html?filter=3943,3935 (abgerufen am 02. 03. 2020).

[15] Cytron Technologies. *Product User Manual HC-SR04 Ultrasonic Sensor*. URL: http://web.eece.maine.edu/~zhu/book/lab/HC-SR04%20User%20Manual.pdf (abgerufen am 20. 03. 2020).
[16] Horst Czichos. *Mechatronik : Grundlagen und Anwendungen technischer Systeme*. 2015.
[17] Davincisurgery. *About da Vinci Systems*. URL: https://www.davincisurgery.com/da-vinci-systems/about-da-vinci-systems# (abgerufen am 06. 03. 2020).
[18] Davincisurgery. *Three components of the da Vinci system*. URL: https://www.davincisurgery.com/da-vinci-systems/about-da-vinci-systems# (abgerufen am 06. 03. 2020).
[19] DFRobot. *Bionic_Robot_Hand_Arduino_Connection*. URL: https://www.robotshop.com/de/de/bionische-roboterhand-rechts.html (abgerufen am 20. 03. 2020).
[20] DFRobot. *bionische Roboterhand (rechts*. URL: https://www.robotshop.com/media/catalog/product/cache/image/900x900/9df78eab33525d08d6e5fb8d27136e95/b/i/bionic-robot-hand-right.jpg (abgerufen am 15. 03. 2020).
[21] Digikey. *Vibrationsmotor 316040001*. URL: https://media.digikey.com/Photos/Seeed%20Technology%20Ltd/MFG_316040001.jpg (abgerufen am 04. 03. 2020).
[22] Emus BMS. *EMUS BMS mini User Manuel v0.9*. URL: https://emusbms.com/wp-content/uploads/2019/05/EMUS-BMS-mini-User-Manual-v0.9.pdf (abgerufen am 19. 02. 2020).
[23] *Ergonomie der Mensch-System-Interaktion_-Teil_910: Rahmen für die taktile und haptische Interaktion (ISO_9241-910:2011)*. Norm. 2011.
[24] Espressif Systems. *ESP32 Series Datasheet*. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (abgerufen am 07. 03. 2020).
[25] Rolf Fischer. *Elektrische Maschinen*. ger. Fachbuch. 2017.
[26] Limor Fried. *Adafruit Feather M0 Bluefruit LE*. URL: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-feather-m0-bluefruit-le.pdf?timestamp=1582655038 (abgerufen am 02. 03. 2020).
[27] Daniel G. *H-bridge*. URL: https://www.mikrocontroller.net/attachment/preview/120205.jpg (abgerufen am 06. 03. 2020).
[28] Winfried Gehrke u. a. *Digitaltechnik: Grundlagen, VHDL, FPGAs, Mikrocontroller*. 2016.
[29] Pauwel Goethals u. a. „Powerful Compact Tactile Display with Microhydraulic Actuators“.
[30] Ronan Hinchet u. a. „DextrES: Wearable Haptic Feedback for Grasping in VR via a Thin Form-Factor Electrostatic Brake“.
[31] Dipl.-Ing.(FH) Roland Holzer. *Automatisierungstechnik, Jahrgang 4*. Unterrichtsmaterial. 2018.
[32] igus. *Abtriebsencoder robolink - D*. Datenblatt von igus zugesendet.
[33] igus GmbH. *Technische Dokumentation robolink DC Version*. URL: https://www.igus.de/contentData/Products/Downloads/001robolink_dc_version_DINA5_v4.pdf (abgerufen am 14. 03. 2020).
[34] Ing. Hansjörg Praxmarer Ing. Sigurd Seyr. Ing. Günther Rösch. *Elektroinstallation - Blitzschutz - Lichttechnik*. 2017.
[35] Interlink Electronics. *Interlink Electronics FSR Force Sensing Resistor*. URL: https://eckstein-shop.de/Interlink-Drucksensor-Force-Sensing-Resistor-FSR-400-Short-02-N-20-N-76-x-03-mmR-x-H (abgerufen am 07. 03. 2020).

[36] item Industrietechnik GmbH. *item Technische Daten*. URL: https://shop.haberkorn.com/product/download?id=1093820-1-technische-daten&atid=5294 (abgerufen am 01. 03. 2020).
[37] Reiner Korthauer, Hrsg. *Handbuch Lithium-Ionen-Batterien*. 2013.
[38] Roman Lesjak. „Die Rolle einer inertialen Messeinheit in der Anwendung Moving-Base Gravimetry“. Magisterarb. Technische Universität Graz, Sep. 2010. URL: https://diglib.tugraz.at/download.php?id=576a714e65a32&location=browse (abgerufen am 21. 03. 2020).
[39] *LiFePO4 Akku 38120SE 10Ah 3,2V- Headway*. URL: https://www.i-tecc.de/shop/einzelzellen/lifepo4/headway/355/lifepo4-akku-38120se-10ah-3-2v-headway (abgerufen am 04. 03. 2020).
[40] M.Broussely und G.Pistoia. *Industrial Applications of Batteries*. 2007.
[41] Eka Maulana, Muhammad Aziz Muslim und Veri Hendrayawan. „Inverse kinematic implementation of four-wheels mecanum drive mobile robot using stepper motors“. In: *2015 International Seminar on Intelligent Technology and Its Applications (ISITIA)*. Mai 2015, S. 51–56.
[42] MEAN WELL. *60W DIN Rail Type DC-DC Converter*. URL: https://asset.conrad.com/media10/add/160267/c1/-/en/002113586DS01/datenblatt-2113586-mean-well-ddr-60l-24-hutschienen-dcdc-wandler-din-rail-24-vdc-25-a-60-w-1-x.pdf (abgerufen am 08. 03. 2020).
[43] Ansgar Meroth und Petre Sora. *Sensornetzwerke in Theorie und Praxis: Embedded Systems-Projekte erfolgreich realisieren*. 2018.
[44] Microchip. *2.7V 4-Channel/8-Channel 12-Bit A/D Converters with SPI Serial Interface*. URL: https://ww1.microchip.com/downloads/en/DeviceDoc/21298c.pdf (abgerufen am 13. 03. 2020).
[45] Microchip. *megaAVR Datasheet*. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf (abgerufen am 07. 03. 2020).
[46] NASA. *Robonaut 1, Telepresence*. URL: https://robonaut.jsc.nasa.gov/R1/sub/telepresence.asp (abgerufen am 06. 03. 2020).
[47] NexusRobot. *M100A Heavy Duty Mecanum Wheel*. URL: http://www.nexusrobot.com/wp-content/uploads/2019/01/NM100A-M0.1.pdf (abgerufen am 13. 03. 2020).
[48] Felix Nölte. „Ein Exoskelett für haptisches Feedback im virtuellen Raum“. 2019.
[49] Nordic Semiconductor. *UART Serial Port Emulation over BLE*. URL: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v14.0.0/ble_sdk_app_nus_eval.html (abgerufen am 16. 03. 2020).
[50] NXP. *AN10441: Level shifting techniques in I2C-bus design*. URL: https://www.nxp.com/docs/en/application-note/AN10441.pdf (abgerufen am 08. 03. 2020).
[51] Polymaker. *Technical Data Sheet PolyFlex TPU95*. URL: https://cdn-3d.niceshops.com/upload/file/PolyFlex_TPU95_TDS_V4.pdf (abgerufen am 08. 03. 2020).
[52] Ralf Pramberger. „Taktiles Feedback mittels Luft zur multimodalen Interaktion“.
[53] Reichelt. *VM 0610 A 3.0 Vibrationsmotor, 3 V, 150 mA, 10.000 U/min*. URL: https://cdn-reichelt.de/bilder/web/artikel_ws/A900/860210.jpg (abgerufen am 05. 03. 2020).
[54] Patrick J. Schuler. „Robotische Chirurgie – operiert der Roboter?“ 2018.
[55] Andreas Schuster. „Eigenschaften heutiger batterie- und wasserstoffspeichersysteme für eine nachhaltige elektrische mobilität“. In: *6. Internationalen Energiewirtschaftstagung an der TU Wien*. 2019.

[56] shenzhen pknergy energy co. ltd. *Li-Polymer Battery Technology Specification; LIPO503562 1200mAh 3.7V*. URL: https://ecksteinimg.de/Datasheet/ZB07002/LP503562%201200mAH%203.7V%201S1P%20A0.pdf (abgerufen am 20. 02. 2020).

[57] Spectra Symbol. *FLEX SENSOR*. URL: https://www.spectrasymbol.com/product/flex-sensors/ (abgerufen am 07. 03. 2020).

[58] SpectraSymbol. *Biegesensor 2.2"*. URL: https://www.generationrobots.com/de/401948-biegesensor-22.html (abgerufen am 07. 03. 2020).

[59] *TAEV*. 2012.

[60] TC Charger. *1.8KW HK-H Series Charger*. URL: https://www.elektromobilitaet-riester.de/app/download/15141527225/TC+Charger+1%2C8+kW.pdf (abgerufen am 04. 03. 2020).

[61] Texas Instruments. *TPS61322 6.5-µA Quiescent current, 1.8-A switch current boost converter*. URL: http://www.ti.com/lit/ds/symlink/tps61322.pdf?HQS=TI-null-null-mousermode-df-pf-null-wwe&DCM=yes&ref_url=https%3A%2F%2Fwww.mouser.at%2F&distId=26 (abgerufen am 27. 02. 2020).

[62] Univ.-Prof.Dr.-Ing. Hans-Rolf Tränkler und Univ.-Prof.Dr.-Ing Ernst Obermeier. *Sensortechnik:Handbuch für Praxis und Wissenschaft*. ger. Fachbuch. 1998.

[63] XP Power. *A SERIES ISOLATED, PROPORTIONAL DC TO HV DC CONVERTERS*. URL: https://www.xppower.com/portals/0/pdfs/SF_A_Series.pdf (abgerufen am 02. 03. 2020).

Abbildungsverzeichnis

II.1 Zusammenspiel der Teilkomponenten . . . 9
II.2 Benutzeroberfläche des Fernsteuergeräts . . . 10
II.3 Mögliche Zustände der Verbindungs-Statusanzeige . . . 11
II.4 Display-Darstellung der manuellen Referenzierung des Roboters . . . 11
II.5 Display-Darstellung bei der Kalibrierung des Fernsteuergeräts . . . 12

III.1 Bauformen von Synchronmaschinen[25] . . . 16
III.2 Läufer einer dauermagneterregten Synchronmaschine[25] . . . 17
III.3 Aufbau eines AC-Servomotorsystems[25] . . . 17
III.4 Übersicht verschiedener Elektrodenmaterialien[40] [55] . . . 19
III.5 Ungleichmäßige Ladungen [37] . . . 21
III.6 Ladungsausgleich Methoden [37] . . . 21
III.7 Charakterisierung verschiedener Elektrofahrzeuge [37] . . . 23
III.8 Blockschaltbild eines Sensors . . . 24
III.9 Strecken- und Distanzsensoren[62] . . . 25
III.10 Messgrößentransformation bei DMS Druckaufnehmer [62] . . . 26
III.11 Wheatstone-Brücke in Ausführung einer 1/4-Brücke . . . 27
III.12 Darstellung eines kapazitiven Drucksensors [62] . . . 27
III.13 Darstellung des Aufbaus eines FSRs [35] . . . 28
III.14 Messung von Biegungen mittels Clip-On-Cage-DMS [62] . . . 29
III.15 Flex-Sensor von SpectraSymbol[58] . . . 29
III.16 Zusammensetzung der Haptik[23] . . . 32
III.17 Taktiles Feedback durch Hydraulik[29] . . . 33
III.18 Taktiles Feedback mit Vibrationsmotor[53] . . . 33
III.19 EMS-Gürtel zum Muskelaufbau[10] . . . 34
III.20 DextrES Komponenten-Darstellung[30] . . . 35
III.21 Grafische Darstellung des Funktionsprinzip von DextrES [30] . . . 36
III.22 Erzeugen einer bipolaren Rechteckspannung mittels H-Brücke[27] . . . 37
III.23 Resultierende Reibungskraft bei verschiedenen Spannungen [30] . . . 37
III.24 DaVinci RAC-System[17] . . . 38
III.25 Nutzungszahlen des DaVinci-Systems[54] . . . 39
III.26 Robonaut[46] . . . 40
III.27 Schematische Darstellung I^2C-Bus . . . 42
III.28 Lese- und Schreibvorgang bei registerbasierter I^2C-Kommunikation . . . 43
III.29 Einteilung additiver Fertigungsverfahren[1] . . . 47

IV.1 Gerüst aus Alu-Konstruktionsprofilen . . . 52
IV.2 Bodenplatte mit Akku und Hutschiene . . . 53
IV.3 Seitenplatten mit BMS und Stecker . . . 53
IV.4 Explosionsdarstellung des Magnethalters . . . 54
IV.5 Grundplatte . . . 55

IV.6 SPS-Gehäuse . . . 56
IV.7 X20 Compact-S CPU (X20CP0484-1) . . . 57
IV.8 Aufbau der Compact-S CPU . . . 58
IV.9 X20 Schrittmotormodul (X20SM1446-1) . . . 59
IV.10 Aufbau der Schrittmotormodule . . . 59
IV.11 Zusammengestellte SPS . . . 60
IV.12 ACOPOSmicro Wechselrichtermodul . . . 61
IV.13 Motor . . . 62
IV.14 Motorhalter . . . 63
IV.15 Mecanum-Wheel . . . 63
IV.16 Flansch . . . 64
IV.17 Schwenkbereich des Robolink[33] . . . 65
IV.18 Signalverlauf der Abtriebsencoder[33] . . . 65
IV.19 bionischer Robotergreifer[20] . . . 66
IV.20 Drehtelleraufsatz zur Befestigung des Greifers . . . 67
IV.21 Ultraschall-Sensormodul HCSR04[15] . . . 67
IV.22 Sensorhalter . . . 68
IV.23 Erster Entwurf der Sensorausrichtung . . . 69
IV.24 Finales Sensorlayout . . . 69
IV.25 Spannungsversorgung der Receiverplatine . . . 70
IV.26 Status-LEDs der Receiverplatine . . . 70
IV.27 Anschluss der Ultraschall-Sensormodule . . . 71
IV.28 Schaltplan ATmega328P . . . 71
IV.29 Reset-Taster des ATmega328P . . . 72
IV.30 Schaltplan ESP32 . . . 72
IV.31 I^2C Level-Shifter . . . 73
IV.32 Anschluss der Servomotoren des Greifers . . . 73
IV.33 Receiver-Platine Top . . . 74
IV.34 Receiver-Platine Bottom . . . 74
IV.35 Gehäuse der Receiver-Platine . . . 75

V.1 Eigenschaften Lithium-Eisen-Phosphat Akkuzelle[39] . . . 81
V.2 Explosionsdarstellung einer Doppelzelle . . . 82
V.3 Zusammengestellter Akku . . . 82
V.4 Verschaltung des Akkus . . . 83
V.5 Akku verschiedene Ansichten . . . 84
V.6 Akkuhalter . . . 84
V.7 Abdeckung des Akkus . . . 85
V.8 Emus Batteriemanagementsystem[22] . . . 87
V.9 Anschlussplan der Batteriezellen Spannungsmessung[22] . . . 88
V.10 Anschlussplan der Batteriezellen Temperaturmessung[22] . . . 89
V.11 Anschlussplan der Last[22] . . . 89
V.12 High Speed CAN-Netzwerk . . . 90
V.13 Anschluss CAN-Bus an der SPS [5] . . . 90
V.14 CAN-Bus Steckerbelegung BMS[22] . . . 91
V.15 EMUS BMS Mini Applikation . . . 92
V.16 Battery Cells Settings . . . 93
V.17 Voltage & Current Settings . . . 94
V.18 TC CAN-Charger[60] . . . 96
V.19 Ladevorgang[60] . . . 96

V.20 Anschlussplan Charger[60] . . . 97
V.21 Charger Steckerbelegung Input[60] . . . 97
V.22 Charger Steckerbelegung Output[60] . . . 98
V.23 Charger Steckerbelegung Signal[60] . . . 98
V.24 D-Sub-Stecker Belegung[3] . . . 98
V.25 Leitungsdimensionierung[59] . . . 99
V.26 Anschlussbeispiel Schrittmotormodul [5] . . . 103
V.27 Anschlussbelegung ACOPOSmicro [5] . . . 104

VI.1 Darstellung der Daumen-Abwicklung . . . 108
VI.2 Papierausdruck des Handschuhs zur Ermittlung der Maße . . . 108
VI.3 Verlängerungsmechanik . . . 109
VI.4 Handrücken . . . 109
VI.5 Unter-/Oberteil der Elektronik-Box . . . 110
VI.6 Explosions- und Schnittdarstellung der Elektronik-Box . . . 111
VI.7 Finger-Abwinkelung . . . 111
VI.8 Fingerspitze . . . 112
VI.9 Verkabelung der Fingerspitze . . . 112
VI.10 Vibrationsmotor[21] . . . 113
VI.11 Explosionsdarstellung Fingerspitze und Vibrationsmotor . . . 113
VI.12 Einfluss des Elektrodenabstands auf die resultierenden Kräfte . . . 116
VI.13 Anschluss des Akkus . . . 119
VI.14 Verschaltung des Inertialsensors . . . 120
VI.15 Biegesensoren auf dem Handrücken . . . 121
VI.16 Auswerteschaltung der Biegesensoren . . . 122
VI.17 Auswerteschaltung der Biegesensoren mit externem ADC . . . 123
VI.18 Verschaltung des ADCs . . . 123
VI.19 Montage des Drucksensors . . . 124
VI.20 Auswerteschaltung der Druck-Sensoren . . . 125
VI.21 Ansteuerung des Vibrationsmotors . . . 126
VI.22 Verschaltung des Darlington-Transistor-Arrays . . . 126
VI.23 Verschaltung der IO-Erweiterung . . . 127
VI.24 Schaltsspannungsregler V_{BAT} auf 5 V . . . 128
VI.25 HV Output unter verschiedenen Belastungen [63] . . . 128
VI.26 Hochspannungsregelung und Strombegrenzung . . . 129
VI.27 Simulation der Hochspannungs-Regelung . . . 130
VI.28 Ansteuerung des kinästhetischen Feedbacks . . . 131
VI.29 Führung der Stripleitungen zum Stecker . . . 131
VI.30 Layout der LV-Platine . . . 133
VI.31 Layout der HV-Platine . . . 134
VI.32 Exemplarisches Datenblatt für einen Hochspannungswiderstand [13] . . . 134
VI.33 Anbringung des Mikrocontrollers an der Rückseite des Displays . . . 135
VI.34 Pin-Header für Huckepack-Platinen . . . 136
VI.35 Huckepack-Platinen . . . 137

VII.1 Topologie der Software . . . 140
VII.2 Aufbau einer Nachricht im Protokoll . . . 143
VII.3 Ablauf bei einem erfolgreichen Verbindungsaufbau . . . 145
VII.4 Topologie Embedded System im Fernsteuergerät . . . 148
VII.5 Zuständigkeiten der Systeme im Fernsteuergerät . . . 149

VII.6 Achsrichtungen BNO055 . . . 154
VII.7 Skalierungskennlinie Biegesensor . . . 156
VII.8 Flussdiagramm ES-Brake Aktivierung . . . 157
VII.9 Minimierung der zu zeichnenden Fläche . . . 158
VII.10 Ablaufdiagramm des Display-Systems . . . 159
VII.11 Topologie Embedded System am Roboter . . . 160
VII.12 Zuständigkeiten der Systeme im Receiver . . . 160
VII.13 Ablaufdiagramm Verbindungsaufbau zum Fernsteuergerät . . . 162
VII.14 Topologie SPS-System am Roboter . . . 168
VII.15 Flussdiagramm: Kommunikation über die serielle Schnittstelle der SPS . . . 170
VII.16 Schematische Darstellung des Mecanum-Wheel-Antriebs[41] . . . 173
VII.17 Konfiguration der Achs-Objekte des Fahrwerks . . . 174
VII.18 Grenzwerte der Fahrwerk-Achsen . . . 174
VII.19 Konfiguration des ACOPOSmicro . . . 175
VII.20 Flussdiagramm `Wheel`-Funktionsblock . . . 176
VII.21 Konfiguration von Achse 1 des Roboterarms . . . 178
VII.22 Konfiguration der Roboterarm-Mechanik . . . 179
VII.23 Erstellen der Achs-Gruppe . . . 180
VII.24 Ablaufdiagramm Roboter-Steuerung . . . 182
VII.25 Regelcharakteristik des Positionsreglers für den Roboterarm . . . 183
VII.26 Einfügen des BMS als CANopen-Gerät in Automation Studio . . . 184
VII.27 Verfügbare Datenpunkte des BMSmini . . . 184

Tabellenverzeichnis

IV.1 Berechnung des Gesamtgewichts 51

V.1 Stromberechnung Bedienmodus „Greifen“ 80
V.2 Stromberechnung Bedienmodus „Fahren“ 80

VII.1 I^2C Register zur Ansteuerung der ATmega328P-Mikrocontroller 166

Listings

VII.1 Definition der Nutzdaten für `msg::init` . . . 144
VII.2 API der `Protocol`-Klasse . . . 146
VII.3 `SharedData`-Struktur im Fernsteuergerät . . . 149
VII.4 Initialisierung des BLE-Moduls . . . 151
VII.5 Write-Funktion für BLE am Fernsteuergerät . . . 152
VII.6 Konstruktor der `CommunicationSystem`-Klasse . . . 152
VII.7 Übergeben der empfangenen Daten . . . 152
VII.8 Initialisierungssequenz BNO055 . . . 153
VII.9 Auslesen der BNO055-Sensordaten . . . 154
VII.10 Auslesen von Analogwerten mit dem MCP3208 . . . 155
VII.11 `SharedData`-Struktur im Receiver . . . 161
VII.12 Implementierung des Protokolls für die BLE-Schnittstelle am Receiver . . . 163
VII.13 Senden der Feedbackinformation . . . 163
VII.14 Implementierung des Protokolls für die serielle Schnittstelle am Receiver . . . 164
VII.15 Auswertung der Ultraschallsensoren . . . 166
VII.16 Implementierung der Bewegungsgleichungen im Programm . . . 177
A.1 protocol/shared_types.h . . . 188
A.2 protocol/messages.h . . . 189