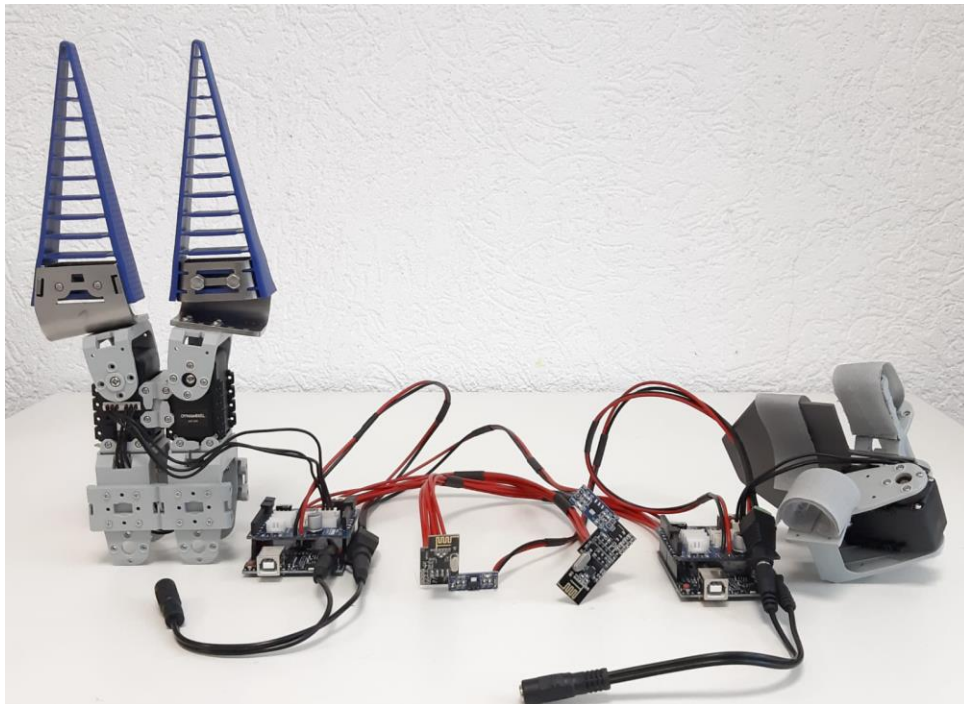


Technikerarbeit

Elektromechanischer Greifer

mit

Force-Feedback-Funktion



ILJA RUSCH

MARCEL RENÉ STEIN

Carl-Benz-Schule Koblenz

Wehrtechnische Dienststelle 41

Abgabe: 24.04.2020

Eidesstattliche Erklärung

Mitwirkende Personen:

Ilja Rusch

Marcel René Stein

Hiermit erklären wir an Eides statt, dass wir die vorliegende Arbeit selbstständig und nur unter Zuhilfenahme der ausgewiesenen Hilfsmittel angefertigt haben.

Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach anderen gedruckten oder im Internet verfügbaren Werken entnommen sind, haben wir durch genaue Quellenangaben kenntlich gemacht.

Ort, Datum, Ilja Rusch

Ort, Datum, Marcel René Stein

Vorwort

Die Entstehung der Idee

Bei einem Besuch der Wehrtechnische Dienststelle 41 (WTD41) in Koblenz, am Tag der Bundeswehr, während der Vorstellung des ferngesteuerten Roboters mit Manipulatorarm ist aufgefallen, dass der elektromechanische Greifer einfach nur per Knopfdruck geschlossen und wieder geöffnet werden kann. Bei der Vorführung der Funktion des Manipulatorarms ist aufgefallen, dass die gegriffenen Objekte sporadisch wieder aus dem Greifer herausgefallen sind.

Dabei ist die Idee entstanden, eine Kraftrückkopplung in den elektromechanischen Greifer zu implementieren, um die auftretenden Fehler auf ein möglichst kleines Niveau zu reduzieren.

Nach Rücksprache mit einem Vertreter der WTD41 aus dem Bereich der Robotik wurden die Rahmenbedingungen und Unterstützungsmöglichkeiten zur Realisierung der Lösung mit der Kraftrückkopplung festgestellt. Daraus wurden die Anforderungen für das Projekt „Elektromechanischer Greifer mit Force-Feedback-Funktion“ abgeleitet und die WTD41 als zukünftiger Betreuer für das Abschlussprojekt in Aussicht gestellt.

Inhaltsverzeichnis

1	Projekteinführung	1
2	Projektauftrag	2
3	Projektplan	3
4	Konzept	4
4.1	Zielsetzung und Ablaufplan	4
4.2	Design der Hauptkomponenten	4
4.2.1	Problematik des Designs	4
4.2.2	Lösungsvorschläge	4
4.2.3	Lösungsauswahl	5
4.3	Datenverarbeitung	6
4.3.1	Problematik der Datenverarbeitung	6
4.3.2	Lösungsvorschläge	6
4.3.3	Lösungsauswahl	6
4.4	Datenübertragung	6
4.4.1	Problematik der Datenübertragung	6
4.4.2	Lösungsvorschläge	7
4.4.3	Lösungsauswahl	7
5	Materialplanung	7
5.1	Zielsetzung und Ablaufplan	7
5.2	Umsetzung	7
6	Hardware	10
6.1	Zielsetzung und Ablaufplan	10
6.2	Elektromechanischer Greifer	11
6.2.1	Montagegestell	11
6.2.2	Greifer-Antrieb	12
6.2.3	Greiffinger	13
6.3	Aktor-Sensor-Einheit (ASE)	13
6.3.1	Trägerplatte	14
6.3.2	ASE-Antrieb	14
6.3.3	Daumenführung	15
6.4	Datenverarbeitungs- und Datenübertragungseinheit	15
6.4.1	NRF24L01 Pinbelegung/Anschluss	17
7	Software	19
7.1	Zielsetzung und Ablaufplan	19
7.1.1	Programmierungssoftware	19

7.1.2	Visualisierungssoftware	19
7.1.3	Betriebssoftware für die Dynamixel AX-12A-Servomotoren.....	20
7.1.4	Programmaufbau.....	20
7.1.5	Datenanalyse	21
7.2	Funkübertragung	22
7.2.1	Problematik durch Unterbrechungen des Funkbetriebs.....	25
7.2.2	Lösungsansatz und Umsetzung.....	25
7.3	Einfache Ansteuerung des Greifers über die Funkverbindung	25
7.3.1	Problematik durch ungleichmäßiges Fahrverhalten	29
7.3.2	Lösungsansatz und Umsetzung.....	29
7.4	Force-Feedback-Funktion	31
7.4.1	Problematik durch erschwerte Bedienung der Force-Feedback-Funktion.....	36
7.4.2	Lösungsansatz und Umsetzung.....	37
7.4.3	Problematik durch unbeständiges Verhalten der Selektionen	38
7.4.4	Lösungsansatz und Umsetzung.....	38
7.4.5	Problematik durch ungewollte Nebeneffekte des Greifers	39
7.4.6	Lösungsansatz und Umsetzung.....	39
7.4.7	Problematik durch Lastrichtungsänderungen des Greifers.....	40
7.4.8	Lösungsansatz und Umsetzung.....	41
7.4.9	Problematik durch fehlende Startroutine und fehlende Sicherheit.....	44
7.4.10	Lösungsansatz und Umsetzung.....	44
8	Validierung	47
9	Schlussbetrachtung.....	48
10	Ausblick.....	49
11	Summary	50
12	Abbildungsverzeichnis.....	52
13	Tabellenverzeichnis.....	53
14	Quellenverzeichnis.....	53
15	Anhang.....	54
15.1	Bestellliste	54
15.2	Programme.....	55
15.2.1	Funkübertragung ASE	55
15.2.2	Funkübertragung Greifer	56
15.2.3	Einfache Servoansteuerung ASE	57
15.2.4	Einfache Servoansteuerung Greifer	58
15.2.5	Einfache Servoansteuerung Zusatzprogramm	60

15.2.6	Force-Feedback-Funktion ASE	61
15.2.7	Force-Feedback-Funktion Greifer	66

1 Projekteinführung

In der WTD41 soll für den Manipulatorarm des ferngesteuerten Roboters Telexmax PRO von der Firma Telerob der Prototyp eines elektromechanischen Greifers mit Krafrückkopplung (Force-Feedback) auf die Hand des Gerätebedieners konzipiert, entworfen und angefertigt werden.

Dazu werden zwei Hauptkomponenten benötigt. Zum einen ein Bedienelement, das sowohl für die Steuerung des elektromechanischen Greifers, als auch für die Haptik in Form einer Aktor-Sensor-Einheit (ASE) fungieren soll. Zum anderen der elektromechanische Greifer, der den Anweisungen der ASE Folge leisten, Objekte greifen und dabei entstehende Kräfte an diese rückführen soll.

Sowohl die ASE als auch der elektromechanische Greifer werden mittels Servomotoren realisiert, die über eine ausreichend hohe Geschwindigkeit verfügen und ihre Positionen sowie die auftretenden Kräfte im Nennbetrieb messen und ausgeben können.

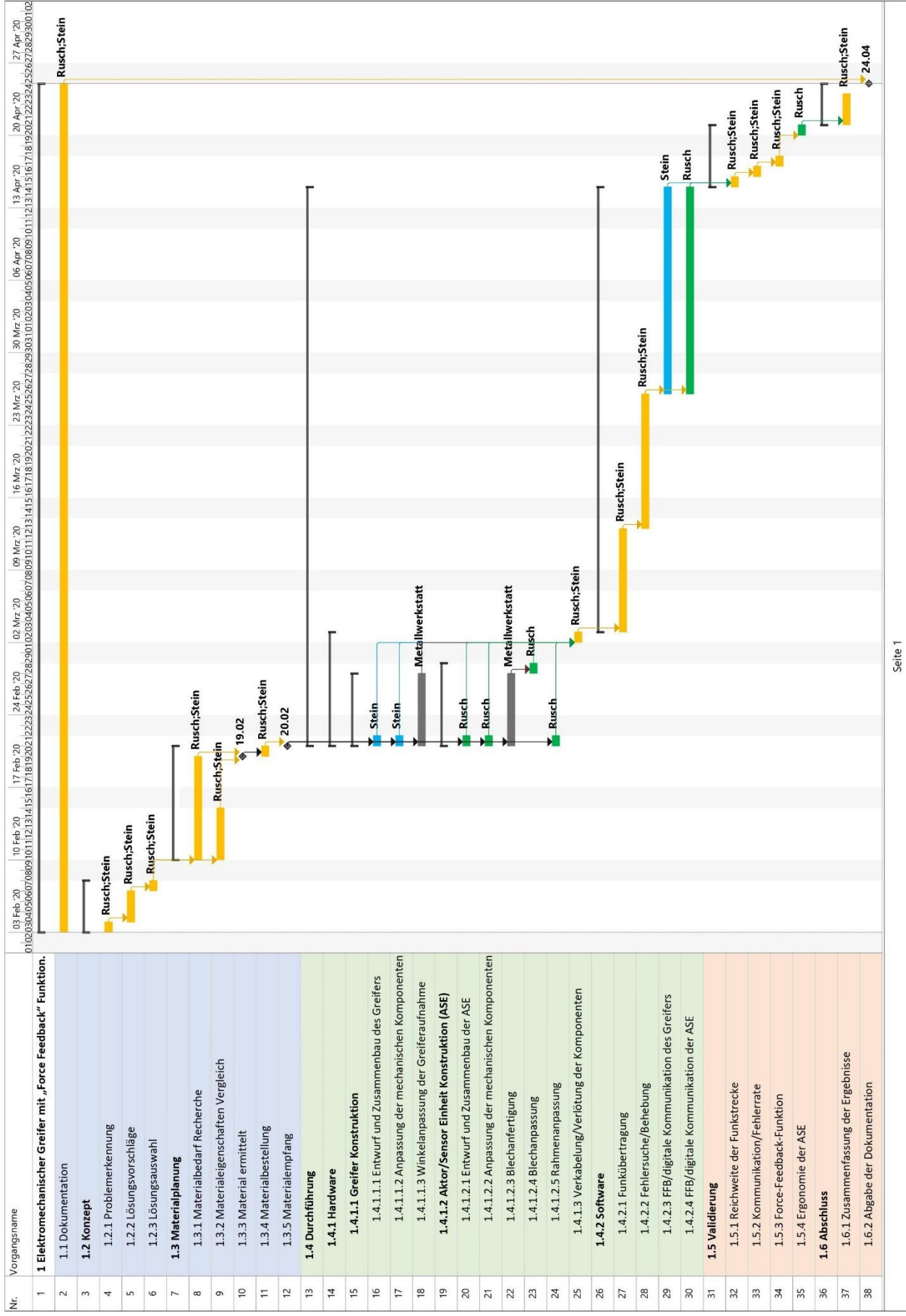
Des Weiteren soll die Datenübertragung der Signale durch eine Funkstrecke stattfinden. Die Datenübertragungsgeschwindigkeit muss möglichst echtzeitfähig sein, damit nur geringe Verzögerungszeiten zwischen der ASE-Bedienung und der Greifer-Krafrückkopplung entstehen. Um eine Fernsteuerung gewährleisten zu können, ist eine Mindestentfernung von ca. 10m Luftlinie einzuhalten.

Gefordert ist eine Datenverarbeitung mit einem Mikrocontroller, der eine hohe Verfügbarkeit aufweist und für die Prototypenentwicklung geeignet ist.

2 Projektauftrag

<u>Projektauftrag</u>		
Projektname:	FSTEE 18 Modul 14	
Auftraggeber:	BBS-Technik Koblenz	Projektteam: Ilja Rusch Marcel René Stein
Datum:	17.12.19	
Projektthema:		
Realisieren der „Force Feedback“ Funktion für einen elektromechanischen Greifer.		
Projektziele:		
Sachziel:	Elektromechanischer Greifer mit „Force Feedback“ Funktion.	
Kostenziel:	Maximal 1000 €	
Terminziel:	03.04.2020	
Hauptaufgaben:		
Einlesen, verarbeiten und ausgeben der elektrischen Signale von Aktorik und Sensorik des elektromechanischen Greifers und vom Steuerelement (Realisiert als Handschuh).		
Realisierung der WLAN-Datenübertragung zwischen dem elektromechanischen Greifer und des Steuerelements mittels Mikrocontroller/Mikrocomputer.		
Übertragung der Steuerungssignale und Messwerte über Bussysteme.		
Ansteuerung der Servomotoren.		
.		
Organisation:		
Auftraggeber:	BBS-Technik-Koblenz	
Industriepartner:	Bundeswehr, WTD 41	
Projektressourcen:		
Projektbudget:	1000€	
Sonstige Ressourcen:	Werkstatt, 3D Drucker.	
Auftragsbedingungen:		
Die Werkstatt und 3D Drucker stehen uns während der regulären Arbeitszeiten zur Verfügung.		
Termine:		
Projektstart:	03.02.2020	
Zwischenpräsentation:	06.03.2020	
Dokumentationsabgabe:	24.04.2020	
Präsentation/ Kolloquium:	03.06.2020/ 04.06.2020	
Auftraggeber:		
Datum:		
Unterschriften:		

3 Projektplan



4 Konzept

4.1 Zielsetzung und Ablaufplan

Durch die Aufstellung eines strukturierten Konzeptes sollte ein Leitfaden angefertigt werden, mit dem die Realisierung des Projektes geordnet und mit entsprechend hoher Anforderung an die vorausgesetzte Qualität erfüllt werden konnte.

Dafür sollten sämtliche Probleme der gestellten Aufgaben erkannt und Lösungsvorschläge zusammengetragen werden. Die beste Lösung wurde dann zum Ziel gesetzt und bearbeitet.

4.2 Design der Hauptkomponenten

4.2.1 Problematik des Designs

Zunächst musste ein Design für die Hauptkomponenten gefunden werden. Die Aktor-Sensor-Einheit (ASE) sollte sowohl funktional sein als auch ergonomisch angenehm für eine optimale Bedienung durch die Hand des Benutzers. Auch der elektromechanische Greifer sollte Objekte gut greifen können und mit der Geschwindigkeit der ASE mithalten, um deren Bewegungen möglichst echtzeitfähig umsetzen zu können.

4.2.2 Lösungsvorschläge

Für die ASE ergaben sich zwei Lösungsansätze. Der erste Ansatz war ein Handschuh, bei dem der kleine Finger, Ringfinger, Mittelfinger und Zeigefinger fest miteinander verbunden sind und nur der Daumen beweglich bleibt. Daraus ergeben sich zwei voneinander unabhängige und bewegliche Teile des Handschuhs: Der Daumen und die Gruppierung der anderen Finger. Diese Teile sollten zur Steuerung an der Oberseite des Handschuhs, an dem Drehpunkt, mit der Achse des Servomotors verbunden werden.

Der zweite Ansatz für das Design der ASE war eine Variante, bei der der Servomotor in der Hand liegt, aber ebenfalls durch den Daumen bedient wird. Der kleine Finger, Ringfinger, Mittelfinger und Zeigefinger werden auf einer Platte fixiert, die wiederum mit dem Servomotor verbunden ist. Dies dient als Haltepunkt, um mit dem Daumen über eine Verbindung zur Achse des Servomotors steuern zu können.

Für den elektromechanischen Greifer ergaben sich drei Lösungsansätze. Der erste Vorschlag war ein Greifer, bei dem drei Finger zum Greifen verwendet werden, welche durch einen Servomotor über eine entsprechende Übersetzung angesteuert werden.

Der zweite Ansatz war ein Greifer mit zwei Fingern, die über einen Gewindeantrieb von einem Servomotor angesteuert werden und sich auf einem Schlitten zusammen- oder auseinander schieben lassen.

Die dritte Idee war ein Greifer mit zwei Fingern, die jeweils von einem eigenen Servomotor angesteuert werden. Die zwei Servomotoren sind miteinander verbunden, um eine Einheit zu bilden.

4.2.3 Lösungsauswahl

Bei der ASE fiel die Entscheidung auf die zweite Variante. Da der Servomotor in der Hand liegt und nicht darauf, ist zum einen das Gewicht zentraler verteilt und die Größe der ASE fällt kleiner aus. Sie ist somit kompakter und von der haptischen Wahrnehmung her besser als der erste Ansatz.

Bei dem elektromechanischen Greifer wurde der dritte Ansatz gewählt. Ausschlaggebend war hier die Tatsache, dass die Echtzeitfähigkeit durch direkte Ansteuerung der Greiffinger von den Servomotoren am besten gegeben ist und es ist kein zusätzlicher Aufwand einer mechanischen Übersetzung notwendig.

4.3 Datenverarbeitung

4.3.1 Problematik der Datenverarbeitung

Um die von den Servomotoren ausgegebenen Daten der Positionen und Kräfte einlesen, auswerten, verarbeiten und wieder ausgeben zu können, musste ein passendes Rechensystem gewählt und angewendet werden.

4.3.2 Lösungsvorschläge

Für die korrekte Datenverarbeitung wurden drei Lösungsvorschläge erarbeitet. Der erste war eine Verarbeitung der Daten mit dem Mikrocontroller ESP8266 WIFI Witty. Die zweite Idee war eine Realisierung mit dem Mikrocontroller Arduino Uno und dritter Ansatz war die Verwendung des PC-basierten Mess-, Steuer- und Regelungssystems NI myRIO von National Instruments.

4.3.3 Lösungsauswahl

Der Lösungsansatz mit der NI myRio Messbox von National Instruments wurde wieder verworfen, da es einen wesentlich größeren Kostenaufwand sowie die ungerechtfertigte Verwendung eines PC-basierten Systems zur Folge hätte. Für die gestellten Aufgaben ist es ausreichend, Mikrocontroller zu benutzen. Da die Ausmaße des gesamten Systems und der damit einhergehende spätere Leistungsbedarf und die Anzahl der Schnittstellen noch nicht bekannt waren, wurden die Möglichkeiten beider Mikrocontroller, also ESP8266 WIFI Witty und Arduino Uno, offengehalten.

4.4 Datenübertragung

4.4.1 Problematik der Datenübertragung

Da das System flexibel und portabel sein sollte, musste eine kabellose Verbindung zur Datenübertragung zwischen der ASE und dem elektromechanischen Greifer hergestellt werden.

4.4.2 Lösungsvorschläge

Zur Erfüllung dieser Aufgabe ergaben sich zwei Lösungswege. Zum einen die Möglichkeit einer WLAN-Verbindung zwischen den jeweiligen ESP8266 WIFI Witty von der ASE und dem elektromechanischen Greifer und zum anderen eine Funkverbindung beider Mikrocontroller, also ESP8266 WIFI Witty oder Arduino Uno, mittels Funkmodulen.

4.4.3 Lösungsauswahl

Um eine direkte Verbindung der Mikrocontroller ohne den Umweg einer Kommunikation über einen Access Point herzustellen, wurde die Entscheidung gefällt, die Datenübertragung mithilfe von Funkmodulen, die an den Mikrocontrollern angeschlossen werden, zu realisieren.

5 Materialplanung

5.1 Zielsetzung und Ablaufplan

Ziel der Materialplanung sollte es sein, geeignetes Material für die Umsetzung des Projekts zu finden und zu bestellen. Das Material musste den Anforderungen gerecht werden und sich dennoch in einem finanziell möglichen Rahmen befinden.

Aus diesem Grund sollte das Material zuerst recherchiert, dann verglichen, ermittelt und letztlich bestellt werden.

5.2 Umsetzung

Das grundlegende Material für die ASE, den elektromechanischen Greifer und den Mikrocontroller wurden in Zusammenarbeit mit den erfahrenen Experten der Robotik der WTD41 recherchiert und bestellt.

Somit ergaben sich für die zwei Hauptkomponenten folgende Materialien:

Von der Firma Robotis:

F3 Montagerahmen für die Servomotoren - seitlicher kleiner Rahmen, womit z.B. 2 Servos miteinander seitlich verbunden werden. Auch hinten am Servomotor montierbar

F4 Montagerahmen lang, zur Montage an die Servomotor-Achse (Schwinger) - Bügel für den Servomotor

ROBOTIS SMPS 12V 5A – Netzteil für die Spannungsversorgung für den Mikrocontroller Arduino Uno und das Board Dynamixel Arduino Shield

U2D2 Power Hub - Stromanschluss-Komponente für den USB2Dynamixel

ROBOTIS Schrauben & Muttern Set BNS-10 – Schrauben für die Robotis-Komponenten

Von der Firma Dynamixel:

Dynamixel AX-12A – Servomotoren

Dynamixel Arduino Shield – Schnittstelle zwischen dem Mikrocontroller Arduino Uno und den Dynamixel-Servomotoren

Von der Firma Festo:

Greiffinger Festo DHAS 120 mm – Flexible Finger für den Greifer

DHAS-MA-B6-120 Befestigungswinkel - Befestigung für den Festo Finger

DHAS-ME-H9-120 Befestigungsbausatz - Befestigung für den Festo Finger

Von der Firma Arduino:

Arduino AG Entwicklungsboard UNO WIFI REV2 – Mikrocontroller für die Datenverarbeitung und Steuerung der Servomotoren

Arduino Entwicklungsboard UNO – Mikrocontroller für die Datenverarbeitung und Steuerung der Servomotoren

Für die Funkübertragung wurden mehrere Funkmodule recherchiert, miteinander verglichen und bestellt. Zum Vergleich wurden die Funkmodule NRF24L01, NRF24L01 + PA + LNA, CC1101, HC12 und HC05 herausgesucht. Die beste Wahl, was Kosten, Übertragungsgeschwindigkeit (Echtzeitfähigkeit) und Reichweite angeht, waren folgende Module:

Von dem Anbieter az-delivery:

NRF24L01, 2.4GHz Funkmodul – Funkmodul für die Datenübertragung.

NRF24L01 + PA + LNA, 2.4GHz Funkmodul - Funkmodul für die Datenübertragung mit einer höheren Reichweite.

Beide Funkmodule operieren im ISM-Frequenzband¹ 2,4GHz.

Gemäß den Bestimmungen der Bundesnetzagentur darf die maximale Strahlungsleistung (EIRP) im erlaubten Frequenzbereich^{2 3} von 2400 bis 2483,5MHz 100mW^{2 3} nicht überschreiten. Zur Berechnung der EIRP-Strahlungsleistung schreibt die Bundesnetzagentur folgende Formel⁴ für die Berechnung vor:

$$EIRP = 10^{\left(\frac{g}{10}\right)} * P(Watt)$$

P: Senderausgangsleistung

g: Antennengewinn [dBi]

Antennen in der Größenordnung wie bei dem NRF24L01 + PA + LNA, 2.4GHz Funkmodul haben in der Regel ein Antennengewinn von etwa 2dBi. Da es sich hierbei allerdings um ein No-Name-Produkt handelt ist die genaue Beschaffenheit der Antenne unklar. Somit kann keine genaue Berechnung der EIRP-Strahlungsleistung vorgenommen werden. Aus diesem Grund ist diese Funkmodul für das Projekt ungeeignet.

Das NRF24L01, 2.4GHz Funkmodul ist mit einer maximalen Senderausgangsleistung⁵ von 0dBm angegeben.

$$P_{(mW)} = 10^{\frac{P_{(dBm)}}{10}}$$

Nach dieser Formel errechnet sich bei 0dBm eine Senderausgangsleistung von 1mW. Aufgrund der Tatsache, dass die errechnete Senderausgangsleistung weit unter den Bestimmungen der Bundesnetzagentur liegt wurde das NRF24L01, 2.4GHz Funkmodul für das Projekt NRF24L01, 2.4GHz Funkmodul ausgewählt.

Für die Spannungsversorgung und die Energieverteilung der Funkmodule wurden ebenfalls mehrere Komponenten und Geräte ausgesucht und miteinander verglichen.

Für die Anforderungen geeignet und bestellt wurden folgende:

Von dem Anbieter kollino-elektronik:

AMS1117-5 DC-DC Spannungsregler – Spannungsregler zur Versorgung der Funkmodule

Von dem Anbieter technicalaim:

AMS1117-3,3 DC-DC Spannungsregler – Spannungsregler zur Versorgung der Funkmodule

Von dem Anbieter vs-onlineshop:

DC-GM Standard Adapter für Hohlsteckerbuchsen – Adapter zur Verteilung der Eingangsspannung vom Netzteil für die Funkmodule und den Dynamixel Arduino Shield

Von dem Anbieter sommernachtyr:

5.5/2.5 DC – Y-Verteilerkabel – Verteilung für die Spannungsversorgung des DC-GM Standard Adapters und dem Arduino Uno

Eine ausführliche Liste der Komponenten und deren Preise befindet sich im Anhang der Dokumentation.

6 Hardware

6.1 Zielsetzung und Ablaufplan

Bei der Hardware sollte die Konstruktion der Hauptkomponenten, die sich aus der Aktor-Sensor-Einheit (ASE) und dem elektromechanischen Greifer zusammensetzen, sowie die Geräte zur Datenübertragung und Datenverarbeitung das Ziel darstellen.

Dafür wurde geplant, zunächst den elektromechanischen Greifer zusammenzubauen, infolgedessen die ASE und zum Schluss die Einheit der Datenverarbeitung und Datenübertragung.

6.2 Elektromechanischer Greifer

Der elektromechanische Greifer wurde wie im Konzept beschrieben geplant, entworfen und konstruiert. Er musste einen stabilen Aufbau haben sowie seiner Hauptfunktion dem Greifen von Objekten gerecht werden. Aus diesem Grund wurde die unten aufgeführte Konstruktion erarbeitet.

Der elektromechanische Greifer besteht hauptsächlich aus drei Bestandteilen: Dem **Montagegestell**, dem **Greifer-Antrieb** und den **Greiffingern**. In der Abbildung 1 ist das gesamte System des elektromechanischen Greifers zu sehen. Zusammen mit den Schrauben zur Verbindung der drei Bestandteile setzt sich der komplette Greifer aus insgesamt 114 Einzelteilen zusammen. Das Ergebnis ist eine sehr robuste Konstruktion, die die Anforderung und damit geforderten Aufgaben erfüllt.



Abbildung 1: Greifer

6.2.1 Montagegestell

Das Montagegestell dient der einfacheren Handhabung und Montagehilfe des eigentlichen Greifsystems. Abbildung 2 zeigt das Montagegestell in seiner fertigen Konstruktion. Diese besteht aus 56 Einzelteilen. Die Einzelteile des Gestells sind sechs F3 Montagerahmen, zwei F4 Montagerahmen lang, acht S1 Schrauben mit Feingewinde, 16 S3 Schrauben mit Feingewinde und 24 N1 Muttern mit Feingewinde. Das Montagegestell erwies sich als sehr stabil und sehr gut geeignet für die Verwendung des elektromechanischen Greifers.

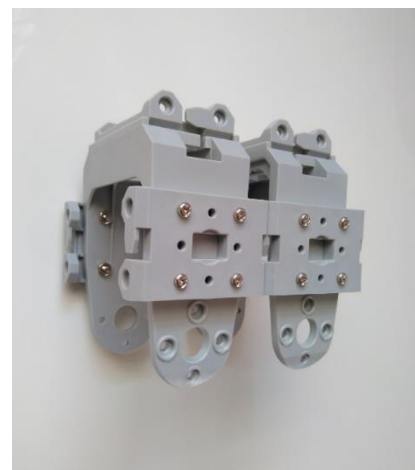


Abbildung 2: Montagegestell

Zudem hat es noch den zusätzlichen Nutzen, als Kabelführung zum Schutz der Leitungen für die Ansteuerung und Datenübertragung der Servomotoren. Auch die Gefahr von Quetschungen durch die Greiffinger im Fahrbetrieb werden durch das Montagegestell verringert.

6.2.2 Greifer-Antrieb

Der Greifer-Antrieb ist das Herzstück des elektromechanischen Systems und ist zuständig für die Ansteuerung der Greiffinger. Ebenso ist er verbunden mit dem Montagegestell. Abbildung 3 veranschaulicht den Aufbau der Konstruktion. Das Konstrukt besteht im Gesamten aus 16 Einzelteilen. Diese sind die zwei Dynamixel AX-12A – Servomotoren, zwei dazugehörigen Bügel für die Achsen der Servomotoren, zwei F3 Montagerahmen, acht S1 Schrauben mit Feingewinde und zwei S-B

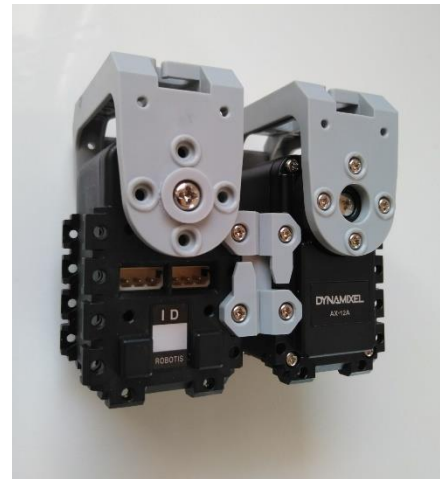


Abbildung 3: Greifer-Antrieb

Schrauben mit Feingewinde. An den zwei Bügeln werden die Greiffinger befestigt. Das stellt die Verbindung zu den Achsen der Servomotoren her. Ein Zusammenschluss der beiden Servomotoren wird über die zwei F3 Montagerahmen erreicht. Zudem sind die AX-12A - Servomotoren um 180° zur Längsachse verdreht montiert, da es die Ansteuerung erleichtert und kein zusätzlicher Programmieraufwand betrieben werden muss. Mit dem gleichen Befehl fahren die Servomotoren in entgegengesetzte Richtungen und es wird der Effekt des Öffnens oder Schließens der Greiffinger erzielt.

6.2.3 Greiffinger

Die Greiffinger werden, wie der Name schon sagt, die zu manipulierenden oder zu bewegenden Objekte greifen. Starre und unflexible Greiffinger würden die Vielfältigkeit der Funktion erheblich einschränken. Aus diesem Grund wurden die flexiblen Greiffinger von der Firma Festo für das System ausgewählt. Abbildung 4 zeigt die besagten Greiffinger. Jeder Finger hat fünf Einzelteile. Bestehend aus einem Festo Greiffinger DHAS 120 mm, dem DHAS-MA-B6-120 Befestigungswinkel und dem dreiteiligen DHAS-ME-H9-120 Befestigungsbausatz. Beide Komponenten kommen somit auf zusammengerechnet zehn Einzelteile. Die Befestigungswinkel der beiden Greiffinger werden an den Bügeln der Servomotoren befestigt und bilden damit die Verbindung zum Antrieb.



Abbildung 4: Greiffinger

6.3 Aktor-Sensor-Einheit (ASE)

Die ASE wurde ebenfalls nach der Lösungsauswahl im Konzept geplant, entworfen und konstruiert. Sie musste sowohl die Funktion erfüllen, die Steuersignale an den elektromechanischen Greifer weitergeben zu können, als auch die Kraftrückkopplung auf die Hand des Bedieners zu übertragen. Zudem sollte die ASE ergonomisch angenehm sein, um eine effiziente Haptik zu erreichen und Verletzungsgefahren vermeiden zu können.



Abbildung 5: Aktor-Sensor-Einheit (ASE)

Drei wesentliche Bestandteile ergeben die ASE: Die **Trägerplatte**, der **ASE-Antrieb** und die **Daumenführung**. Abbildung 5 zeigt die komplette Aktor-Sensor-Einheit und eine verdeutlichte Darstellung der Bedienung durch die Hand. Alle Einzelteile, inklusive Schrauben zur Verbindung, summieren sich auf eine Gesamtzahl von 38. Das Resultat

der Konstruktion war sehr stabil, angenehm für die Hand und einfach zu bedienen. Damit ist auch die ASE ideal für die Erfüllung der geforderten Aufgabe.

6.3.1 Trägerplatte

Die Trägerplatte der ASE dient als Halterung des Servomotors und zugleich als Fixierung von Zeigefinger, Mittelfinger, Ringfinger und dem kleinen Finger. Dadurch wird eine gute Bedienbarkeit erzielt, die im Wesentlichen die Daumenführung übernimmt. Die Trägerplatte besteht aus 19 Einzelteilen: Der Metallplatte, einem kurzen Bügel vom Servomotor, drei F3 Montagerahmen, zwei Klettbänder, acht S1 Schrauben mit Feingewinde und vier N1 Muttern mit Feingewinde. Das Konstrukt der Trägerplatte bietet einen guten Halt und Stabilität. In Abbildung 6 ist die Konstruktion zu sehen.

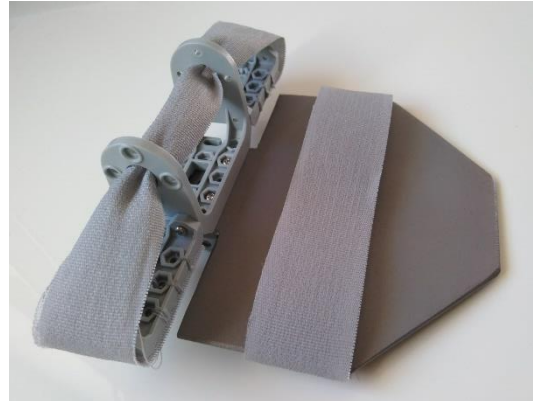


Abbildung 6: Trägerplatte

6.3.2 ASE-Antrieb

Herzstück des Krafrückkopplungssystems ist der ASE-Antrieb. Es enthält 7 Teile: Einen Dynamixel AX-12A – Servomotor, einen F4 Montagerahmen lang, vier S1 Schrauben mit Feingewinde und eine S-B Schraube mit Feingewinde. Hier dient der Servomotor zum einen als Sensor, um die Steuersignale wie die Position an den elektromechanischen Greifer zu übermitteln, und zum anderen als Aktor zur Wiedergabe der Krafrückkopplung an die Hand des Bedieners. Der lange Montagerahmen ist die Verbindung zwischen der Servomotor-Achse und der Daumenführung. Zusammen bilden diese zwei Komponenten den ASE-Antrieb, der in Abbildung 7 dargestellt ist.



Abbildung 7: ASE-Antrieb

6.3.3 Daumenführung

Die Daumenführung wird die Befestigung des Daumens an dem Bügel der ASE-Antriebseinheit. Sie besteht aus einem Klettband, einer S-B Schraube mit Feingewinde, einer N2 Mutter mit Feingewinde und einem Plastikelement. Der Daumen des Bedieners wird mit dem Klettband auf dem Plastikelement fixiert. Da diese beweglich mit

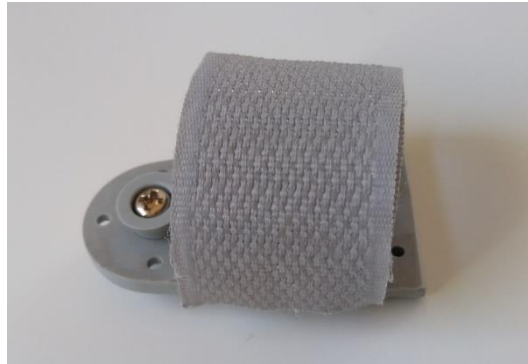


Abbildung 8: Daumenführung

dem Bügel des Servomotors verbunden ist, kann die Handbewegung so auf die Achse des Servomotors übertragen werden. Abbildung 8 zeigt die Daumenführung.

6.4 Datenverarbeitungs- und Datenübertragungseinheit

Die Einheit zum Verarbeiten und zur Übertragung der Daten von dem elektromechanischen Greifer und der ASE besteht aus zwei Komponenten. Zum einen aus dem **Arduino Uno Entwicklungsboard mit aufgestecktem Dynamixel Arduino Shield** und zum anderen aus dem **NRF24L01, 2.4GHz Funkmodul mit angeschlossenen AMS1117-3,3 DC-DC und AMS1117-5 DC-DC Spannungsreglern**. Abbildung 9 zeigt das Ergebnis der fertigen und einsatzbereiten Konstruktion.

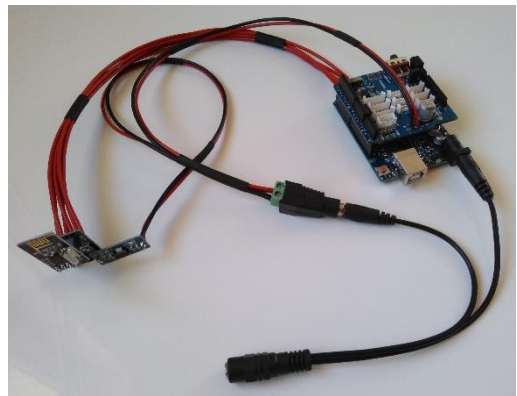


Abbildung 9: Datenverarbeitungs- und Datenübertragungseinheit

Der Arduino Uno wird für die Datenverarbeitung verantwortlich sein und das aufgesteckte Dynamixel Shield wird die Schnittstelle zwischen dem Mikrocontroller und den Servomotoren bilden. Bei der Verwendung des ESP8266 Mikrocontrollers müsste einerseits eine passende Schnittstelle für die Servomotoren zusammengebaut werden und andererseits ist

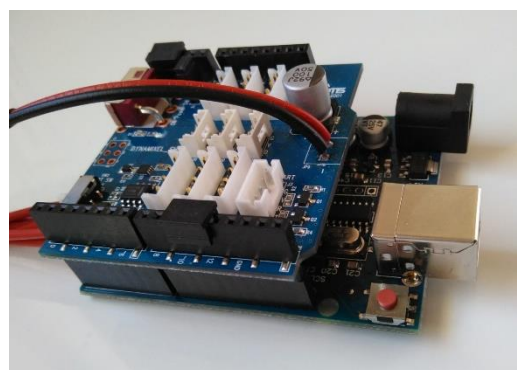


Abbildung 10: Arduino Uno Entwicklungsboard mit aufgestecktem Dynamixel Arduino Shield

bei ersten Versuchen, die Funkübertragung zwischen zwei Geräten herzustellen, herausgefunden worden, dass die Spannung der ESP8266 zusammenbricht und zur Abschaltung führt. Dies bedeutete ebenfalls einen zusätzlichen Aufwand, da die ESP8266 entsprechend umgebaut werden müssten. Um unnötige Verbraucher zu eliminieren, müsste die mehrfarbige LED aus- und ein Pull-up-

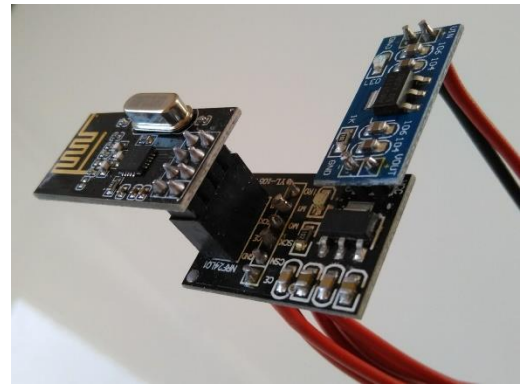


Abbildung 11: NRF24L01, 2.4GHz Funkmodul mit
angeschlossenem AMS1117-3,3 DC-DC und
AMS1117-5 DC-DC Spannungsregler

Widerstand zur Spannungsregelung eingelötet werden. Aus diesem Grund wurde sich für die Verwendung vom Arduino Uno entschieden. Das Dynamixel Shield ist passend für die Arduino Uno Mikrocontroller gebaut und kann einfach aufgesteckt werden. Zudem beinhaltet es eine eigene Spannungsversorgung. Abbildung 10 stellt das Arduino Uno Entwicklungsboard mit aufgestecktem Dynamixel Shield dar. Um dennoch eine stabile Aufrechterhaltung der Spannung an den Funkmodulen zu garantieren, wurden diese zusätzlich mit den Spannungsreglern AMS1117-5 DC-DC (blau) und AMS1117-3,3 DC-DC (schwarz) ausgestattet. Zu sehen ist das auf Abbildung 11. Bei den NRF24L01 Funkmodulen handelt es sich um sogenannte Single-Chip-Transceiver. Diese Funkmodule können sowohl als Sender wie auch als Empfänger arbeiten. Sie kommunizieren abwechselnd über einen gemeinsamen Kanal (Halbduplex-Kommunikation). Zudem sind verschiedene Leistungsstufen softwareseitig einstellbar. Die möglichen Einstellungen sind in aufsteigender Reihenfolge MIN, LOW, HIGH und MAX. Es gibt auch einen Power-Down-Modus, in dem das Funkmodul grade mal eine Stromaufnahme von 900nA hat. Auch die Datenübertragungsrate ist softwaretechnisch einstellbar. Hier kann man die Übertragungsraten 250kB/s, 1MB/s oder 2MB/s einstellen. Je mehr Daten pro Sekunde übertragen werden, desto geringer wird die Funkübertragungsreichweite der Funkmodule. Bei den ersten Tests mit dieser Konstruktion stellte sich heraus, dass erhebliche Probleme in Form von Wackelkontakten auftraten. Dies machte es notwendig, alle steckbaren Leitungsverbindungen zwischen dem Arduino Uno und dem Funkmodul durch Einzeladern zu ersetzen und diese fest zu verlöten - sowohl für die Datenverarbeitungs- und Datenübertragungseinheit der ASE als auch für die des elektromechanischen Greifers. Die Problematik der Wackelkontakte trat daraufhin

nicht mehr auf. Die Einheiten zur Datenverarbeitung und Datenübertragung konnten verwendet werden.

6.4.1 NRF24L01 Pinbelegung/Anschluss

Das NRF24L01 Funkmodul verwendet eine SPI-Schnittstelle für die Kommunikation mit dem Arduino Uno Mikrocontroller. Normalerweise benötigt das Funkmodul keine zusätzliche Spannungsversorgung und kann direkt an den 3,3V Pin des Mikrocontrollers angeschlossen werden. Zur Stabilität und Entlastung des Arduino Unos wurden allerdings externe Spannungsregler an den Funkmodulen verbaut. Jedes Modul hat acht Anschlüsse. Die unten aufgeführte Tabelle 1 listet und beschreibt die Anschlüsse des NRF24L01 Funkmoduls.

Tabelle 1: NRF24L01 Anschlüsse

Anschluss	Beschreibung
Vcc	Versorgungsspannung von 1,9V bis 3,6V
GND	Masse
IRQ	Interrupt Pin für eine Unterbrechungsanforderung des Moduls
CE	Chipfreigabe für Sendemodus oder Empfangsmodus
CSN	SPI Chip Select Pin
SCK	SPI Timer
MISO	SPI Pin für Daten Input
MOSI	SPI Pin für Daten Output

Die folgende Tabelle 2 zeigt den Anschluss des NRF24L01 Funkmoduls an den Arduino Uno Mikrocontroller:

Tabelle 2: Anschluss Funkmodul und Mikrocontroller

NRF24L01 Funkmodul	Arduino Uno Mikrocontroller
IRQ	Nicht angeschlossen
CE	Pin 9 / Digitaler Pin / PWM
CSN	Pin 10 / Digitaler Pin / SPI-SS / PWM
SCK	Pin 13 / Digitaler Pin / SPI-SCK
MISO	Pin 12 / Digitaler Pin / SPI-MISO
MOSI	Pin 11 / Digitaler Pin / SPI-MOSI

Eine bildliche Veranschaulichung von der Verdrahtung des Arduino Uno Mikrocontrollers und des NRF24L01 Funkmoduls zeigt die mit dem Visualisierungsprogramm von Fritzing bearbeitete Abbildung 12.

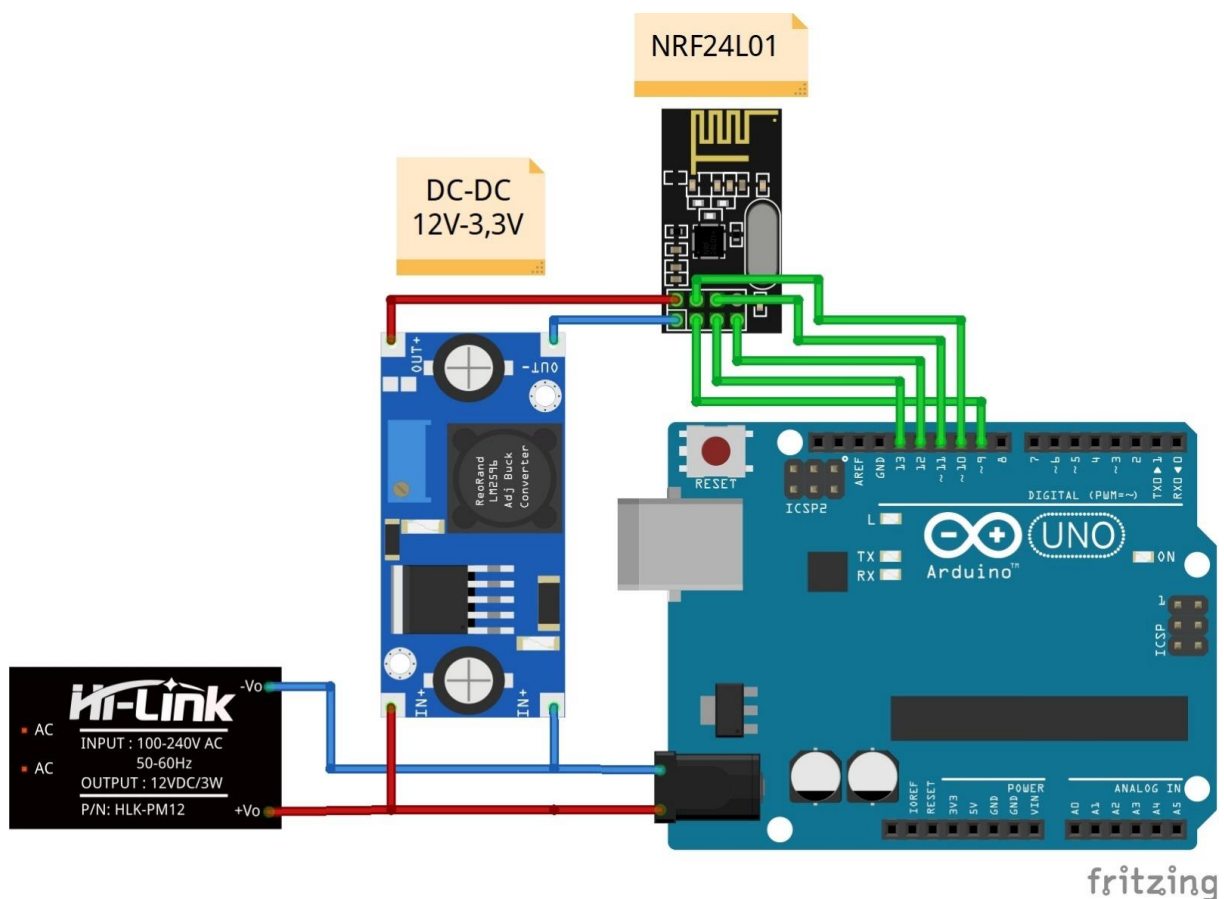


Abbildung 12: Anschluss des NRF24L01 Funkmoduls an dem Arduino Uno Mikrocontroller

7 Software

7.1 Zielsetzung und Ablaufplan

Die Programmierung des Systems zur Realisierung des elektromechanischen Greifers mit Force-Feedback-Funktion sollte die digitalisierte Umsetzung zur Lösung des Problems sein.

Aus dem Grund musste zuerst eine geeignete Programmierungssoftware ausgesucht und installiert werden. Dann sollten die Aufgaben aufgeteilt werden auf die zwei Mikrocontroller der beiden Hauptkomponenten: Der ASE und den Greifer. Die Programmierung sollte gestaffelt werden und mit der Funkübertragung in beide Richtungen anfangen. Danach wurde geplant, die einfache Ansteuerung der Servomotoren über die Funkverbindung umzusetzen und zum Schluss die zum Ziel führende Programmierung der Force-Feedback-Funktion durchzuführen.

7.1.1 Programmierungssoftware

Für die Programmierung der Arduino Uno Entwicklungsboards wurde die Open-Source-Arduino-Software (IDE) Version 1.8.12 für Windows 10 verwendet. Diese Software kann mit jedem Arduino-Board verwendet werden. Sie ermöglicht es Programme schreiben, kompilieren und auf die Mikrocontroller hochladen zu können. Die Umgebung ist in Java geschrieben und basiert auf Processing und anderer Open-Source-Software. Die Programmiersprache ist C bzw. C++.

Da dieses Programm und dessen Funktionen im Unterricht der Techniker-Schule behandelt wurden, war eine Einarbeitung in die Grundlagen nicht mehr nötig.

7.1.2 Visualisierungssoftware

Für die Verdeutlichung und grafische Unterstützung von Arbeitsprozessen wurde die Visualisierungssoftware von Fritzing verwendet. Mit dieser Software können Verbindungen zwischen Bauteilen und Geräten dargestellt werden.

Zur bildlichen Veranschaulichung der Funktionsweisen von geschriebenen Programmen wurde die Visio-UML-Software von Microsoft verwendet.

7.1.3 Betriebssoftware für die Dynamixel AX-12A-Servomotoren

Um die Betriebseigenschaften der Servomotoren zu ermitteln und motorinterne Parameter zu verändern, wurde das DYNAMIXEL Wizard 2.0 Programm installiert und verwendet. Auch die Funktion der Servomotoren kann mithilfe des Programms überprüft werden.

7.1.4 Programmaufbau

Alle geschriebenen Programme haben die gleiche Grundstruktur. Diese teilt sich auf in drei Abschnitte. Im ersten Abschnitt werden Bibliotheken eingebunden, Objekte instanziiert sowie globale Variablen deklariert und gegebenenfalls initialisiert, die für die Ausführung des Programms benötigt werden. Zudem können Bedingungen implementiert werden, um eine Kommunikationsvariante nach Art des angeschlossenen Mikrocontrollers auszuwählen. Der zweite Abschnitt ist der Setup-Teil. Hier werden beim Programmstart einmalige Befehle, wie feste Einstellungen von Objektparametern, ausgeführt. In dem Fall der folgenden Programme sind das die Parameter der NRF24L01 Funkmodule und der AX-12A Servomotoren. Der dritte Teil des Quellcodes ist der Loop-Teil. In ihm werden in ständiger Wiederholung Schleifen, Selektionen und Anweisungen für die Funkübertragung, Ansteuerung der Servomotoren und der Force-Feedback-Funktion ausgeführt.

7.1.5 Datenanalyse

Für die Erstellung der Programme wurden die Datentypen aus Tabelle 3 ausgewählt und verwendet:

Tabelle 3: Datenanalyse

	Name	Typ	Funktion
Eingabedaten	ASE_DIR_PIN	const uint8_t	ASE Direction Pin
	ASE_ID	const uint8_t	ASE-ID
	Timeout	const uint32_t	Timeout
	ASE_Protocol_Version	const float	ASE-Protokollversion
	Addresses[][6]	byte	Pipe
	Greifer_DIR_PIN	const uint8_t	Greifer Direction Pin
	Greifer_ID1	const uint8_t	Greifer-ID1
	Greifer_ID2	const uint8_t	Greifer-ID2
	Timeout	const uint32_t	Timeout
	Greifer_Protocol_Version	const float	Greifer-Protokollversion
	Addresses[][6]	byte	Pipe
Zwischendaten	Daten_Empfangen[2]	int	Empfangsdatenspeicher
	Daten_Senden[2]	int	Sendedatenspeicher
	ASE_Last_Aktuell	int	ASE-Last aktuell
	ASE_Position_Aktuell	int	ASE-Position aktuell
	Greifer_Last	int	Greifer-Last
	Greifer_Position	int	Greifer-Position
	Daten_Empfangen[2]	int	Empfangsdatenspeicher
	Daten_Senden[2]	int	Sendedatenspeicher
	Position	int	Position von der ASE
	ASE_Last	int	ASE-Last
	Greifer_Last	int	Greifer-Last
	Greifer_Last_Alt	int	Greifer-Last alt
Ausgabedaten	ASE_Last_Smooth	int	Gleitender Mittelwert ASE-Last
	ASE_Position_Smooth	int	Gleitender Mittelwert ASE-Position
	ASE_Position_Senden	int	ASE-Position senden
	Greifer_Position	int	Greifer-Position
	Greifer_Last_Smooth	int	Gleitender Mittelwert Greifer-Last

7.2 Funkübertragung

Für die Funkübertragung sollten Programme geschrieben und getestet werden, in denen jeweils Daten von einem Mikrocontroller zum anderen gesendet und empfangen werden. Im Folgenden wird das Wesentliche der Programme gezeigt und erklärt; auch die aufgetretenen Probleme und deren Lösungen werden beschrieben. Abbildung 13 veranschaulicht grafisch das Funktionsprinzip der Funkübertragung.

Funkübertragung

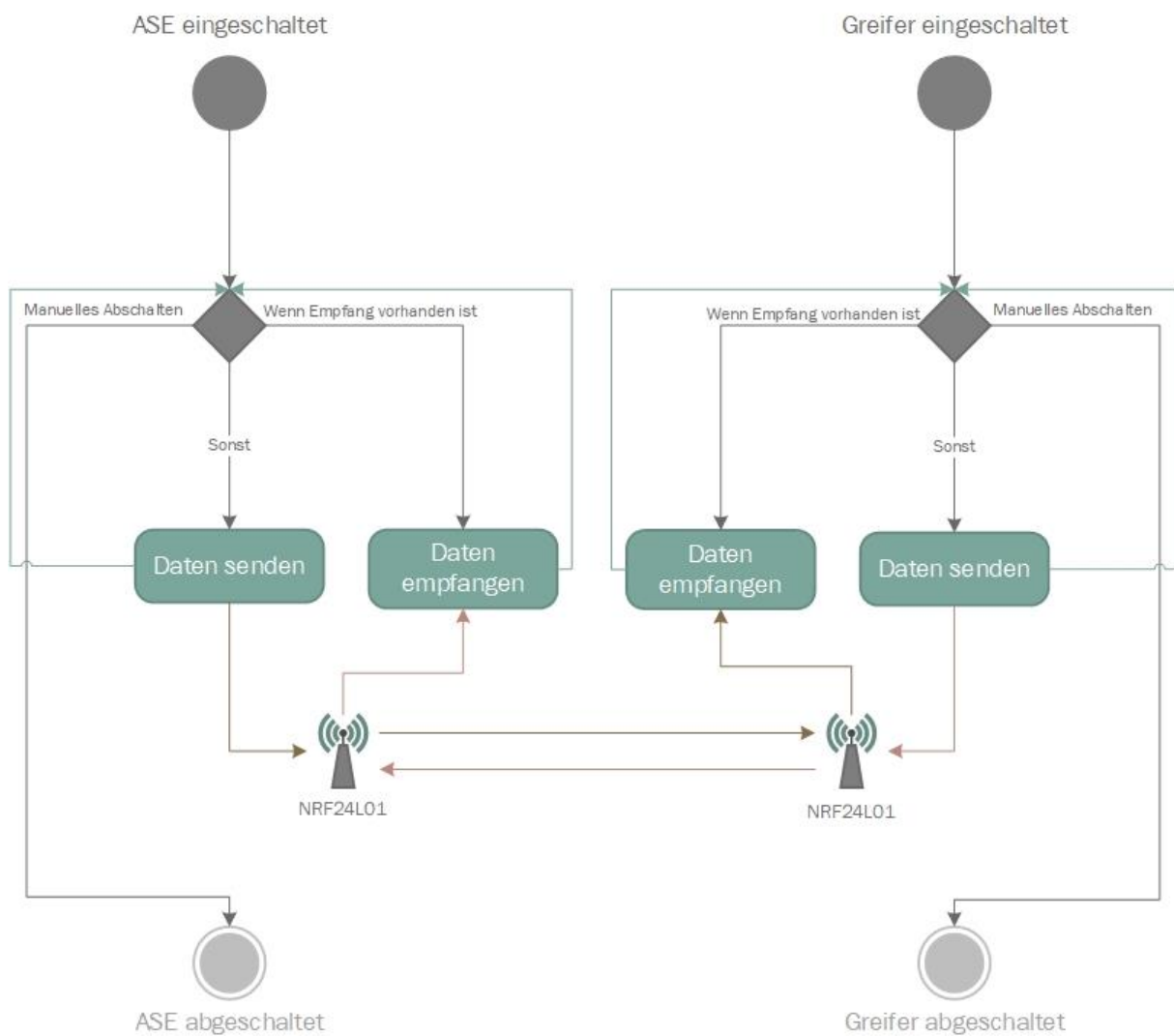


Abbildung 13: Funktionsprinzip der Funkübertragung

In dem Programm für die Funkübertragung werden zwei Integer-Werte in ein Array geschrieben, welches die Daten über die Funkmodule an den anderen Mikrocontroller versendet. Die empfangenen Daten des anderen Mikrocontrollers werden in einem zweiten Array abgelegt und auf dem seriellen Monitor ausgegeben. Zur Unterscheidung der gesendeten und empfangenen Daten des jeweils anderen Mikrocontrollers wurden unterschiedliche Integer-Werte in die Sendedatenspeicher geschrieben. Abbildung 14 und 15 zeigen die Arrays der jeweiligen Mikrocontroller mit den unterschiedlichen Integer-Werten.

```
int Daten_Senden[2]={1111,2222};    // Deklaration und Initialisierung des Sendedatenspeichers
int Daten_Empfangen[2];             // Deklaration des Empfangsdatenspeichers
```

Abbildung 14: Arrays zum Senden und Empfangen vom Mikrocontroller 1

```
int Daten_Senden[2]={3333,4444};    // Deklaration und Initialisierung des Sendedatenspeichers
int Daten_Empfangen[2];             // Deklaration des Empfangsdatenspeichers
```

Abbildung 15: Arrays zum Senden und Empfangen vom Mikrocontroller 2

Um eine Kommunikation zwischen den Funkmodulen der beiden Mikrocontroller aufzubauen, werden eine einheitliche Frequenz und eine einheitliche Pipe ausgewählt. Bei dem NRF24L01, 2.4GHz Funkmodul können alle möglichen Einstellungen der Sendeleistung zum Test und zur Realisierung der Funkübertragung verwendet werden. Für die Geschwindigkeit und Menge an versendeten Daten wird die Einstellung der Übertragungsrate verändert. Alle drei Einstellmöglichkeiten von 250kB/s, 1MB/s sowie 2MB/s wurden ausgetestet. Der Betrieb mit allen Übertragungsraten funktioniert. Um das Maximum an Übertragungsgeschwindigkeit und Übertragungsmenge zu erreichen, wurde die Einstellung mit 2MB/s übernommen. Für die dynamische Funktion der Paketgröße und einer Empfangsbestätigung wurden entsprechende Methoden in das Programm implementiert. Zu Beginn wird eine Pipe geöffnet und der Empfang gestartet. Diese Befehle zur Startroutine der Kommunikation sind bei beiden Mikrocontrollern gleich und befinden sich im Setup-Teil des Programms^{5 6}. Sie sind auf Abbildung 16 zu sehen.

```
ASE_Funk.setChannel(10);              // Einstellung der Sende-/Empfangsfrequenz
ASE_Funk.setPALevel(RF24_PA_MAX);     // Einstellung der Sendeleistung
ASE_Funk.setDataRate(RF24_2MBPS);     // Einstellung der Datenübertragungsrate
ASE_Funk.setAutoAck(1);               // Aktivierung der Acknowledge-Funktion
ASE_Funk.enableAckPayload();          // Aktivierung der Acknowledge-Funktion der
                                     // Paketgröße
ASE_Funk.enableDynamicPayloads();     // Aktivierung der dynamischen Paketgröße
ASE_Funk.openReadingPipe(1, Addresses[0]); // Öffnen einer Pipe zum Empfangen
ASE_Funk.startListening();            // Startet den Empfang
```

Abbildung 16: Startroutine der Funkübertragung

Im Loop-Teil findet der Datenaustausch der Integer-Werte statt. Solange das Funkmodul keine Daten empfängt, ist das Programm im Sendemodus und sendet permanent die Daten aus dem Array des Sendedatenspeichers. Erst wenn erkannt wird, dass Daten empfangen werden, wird auch der Empfangsmodus aktiviert und die Daten in das Array des Empfangsdatenspeichers überschrieben. Die Werte in den jeweiligen Arrays werden auf dem seriellen Monitor ausgegeben. Bei einer fehlerfreien Funkübertragung findet ein ständiger Wechsel zwischen dem Empfangsmodus und dem Sendemodus statt. Abbildung 17 zeigt den Teil des beschriebenen Datenaustausches.

```

if ( ASE_Funk.available() ) {                                     // Prüft, ob Daten empfangen
                                                                // werden

    while (ASE_Funk.available()){                               // Solange Daten empfangen
                                                                // werden

        ASE_Funk.read( &Daten_Empfangen, sizeof(Daten_Empfangen) ); // Datengröße wird ermittelt
                                                                // und im vorgesehenem
                                                                // Speicher abgelegt

    }

    Serial.println("-----"); // Serielle Ausgabe des
                                // Textes

    Serial.print("ASE empfängt1: "); // Serielle Ausgabe des
                                // Textes

    Serial.println(Daten_Empfangen[0]); // Serielle Ausgabe des
                                // Empfangsspeichers

    Serial.print("ASE empfängt2: "); // Serielle Ausgabe des
                                // Textes

    Serial.println(Daten_Empfangen[1]); // Serielle Ausgabe des
                                // Empfangsspeichers

    Serial.println("-----"); // Serielle Ausgabe des
                                // Textes

}

delay(100); // Wartezeit für den multitasking
            // Betrieb der Funkübertragung

ASE_Funk.stopListening(); // Stoppt den Empfang

ASE_Funk.openWritingPipe(Addresses[0]); // Öffnen einer Pipe zum Senden

ASE_Funk.write(&Daten_Senden, sizeof(Daten_Senden)); // Datengröße wird ermittelt und aus
                                                    // vorgesehenem Speicher ausgelesen und
                                                    // gesendet

Serial.print("ASE sendet1: "); // Serielle Ausgabe des Textes
Serial.println(Daten_Senden[0]); // Serielle Ausgabe des Sendespeichers
Serial.print("ASE sendet2: "); // Serielle Ausgabe des Textes
Serial.println(Daten_Senden[1]); // Serielle Ausgabe des Sendespeichers

ASE_Funk.openReadingPipe(1, Addresses[0]); // Öffnen einer Pipe zum Empfangen

ASE_Funk.startListening(); // Startet den Empfang

```

Abbildung 17: Datenaustausch der Funkübertragung

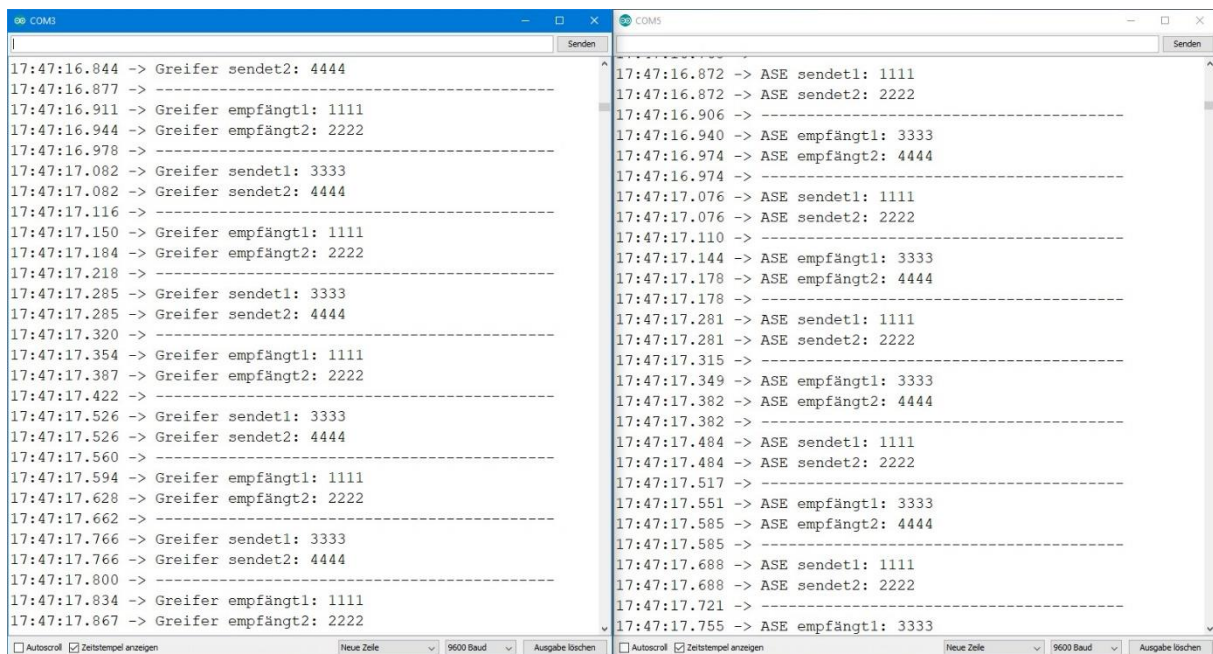
Die kompletten Programme der Funkübertragung befinden sich im Anhang.

7.2.1 Problematik durch Unterbrechungen des Funkbetriebs

Bei den ersten Versuchen traten erhebliche Fehler bei der Funkübertragung auf. Diese Fehler zeigten sich durch sehr ungleichmäßige Wechsel zwischen dem Empfangsmodus und dem Sendemodus. Teilweise stoppte das Programm mitten im Funkbetrieb. Es stellte sich heraus, dass zahlreiche Wackelkontakte in den verwendeten Mikrocontrollern und den Jumper-Kabeln dafür verantwortlich waren.

7.2.2 Lösungsansatz und Umsetzung

Die defekten Arduino Uno Mikrocontroller wurden durch neue ersetzt und alle steckbaren Verbindungen zwischen den Mikrocontrollern und den Funkmodulen wurden durch 0,75mm² Einzeladern ausgetauscht und fest verlötet. Das Problem mit den Wackelkontakten wurde dadurch beseitigt und eine funktionierende Funkübertragung stellte sich ein, wie die Ausgabe der seriellen Monitore auf Abbildung 18 zeigt:



The image shows two side-by-side serial monitor windows. The left window, titled 'COM3', displays a log of messages from a 'Greifer' (gripper) to an 'ASE' (Actor-Sensor Unit). The messages show a sequence of sending and receiving data (1111 and 2222) with timestamps. The right window, titled 'COM5', displays a log of messages from an 'ASE' to a 'Greifer'. It shows a similar sequence of sending and receiving data with timestamps. Both windows have a 'Senden' button at the top right and a status bar at the bottom with options like 'Autoscroll', 'Zeitstempel anzeigen', and 'Ausgabe löschen'.

```
17:47:16.844 -> Greifer sendet2: 4444
17:47:16.877 -> -----
17:47:16.911 -> Greifer empfängt1: 1111
17:47:16.944 -> Greifer empfängt2: 2222
17:47:16.978 -> -----
17:47:17.082 -> Greifer sendet1: 3333
17:47:17.082 -> Greifer sendet2: 4444
17:47:17.116 -> -----
17:47:17.150 -> Greifer empfängt1: 1111
17:47:17.184 -> Greifer empfängt2: 2222
17:47:17.218 -> -----
17:47:17.285 -> Greifer sendet1: 3333
17:47:17.285 -> Greifer sendet2: 4444
17:47:17.320 -> -----
17:47:17.354 -> Greifer empfängt1: 1111
17:47:17.387 -> Greifer empfängt2: 2222
17:47:17.422 -> -----
17:47:17.526 -> Greifer sendet1: 3333
17:47:17.526 -> Greifer sendet2: 4444
17:47:17.560 -> -----
17:47:17.594 -> Greifer empfängt1: 1111
17:47:17.628 -> Greifer empfängt2: 2222
17:47:17.662 -> -----
17:47:17.766 -> Greifer sendet1: 3333
17:47:17.766 -> Greifer sendet2: 4444
17:47:17.800 -> -----
17:47:17.834 -> Greifer empfängt1: 1111
17:47:17.867 -> Greifer empfängt2: 2222

17:47:16.872 -> ASE sendet1: 1111
17:47:16.872 -> ASE sendet2: 2222
17:47:16.906 -> -----
17:47:16.940 -> ASE empfängt1: 3333
17:47:16.974 -> ASE empfängt2: 4444
17:47:16.974 -> -----
17:47:17.076 -> ASE sendet1: 1111
17:47:17.076 -> ASE sendet2: 2222
17:47:17.110 -> -----
17:47:17.144 -> ASE empfängt1: 3333
17:47:17.178 -> ASE empfängt2: 4444
17:47:17.178 -> -----
17:47:17.281 -> ASE sendet1: 1111
17:47:17.281 -> ASE sendet2: 2222
17:47:17.315 -> -----
17:47:17.349 -> ASE empfängt1: 3333
17:47:17.382 -> ASE empfängt2: 4444
17:47:17.382 -> -----
17:47:17.484 -> ASE sendet1: 1111
17:47:17.484 -> ASE sendet2: 2222
17:47:17.517 -> -----
17:47:17.551 -> ASE empfängt1: 3333
17:47:17.585 -> ASE empfängt2: 4444
17:47:17.585 -> -----
17:47:17.688 -> ASE sendet1: 1111
17:47:17.688 -> ASE sendet2: 2222
17:47:17.721 -> -----
17:47:17.755 -> ASE empfängt1: 3333
```

Abbildung 18: Serielle Ausgabe der Funkübertragung

7.3 Einfache Ansteuerung des Greifers über die Funkverbindung

Für die einfache Ansteuerung des Greifers über die bereits realisierte Funkverbindung sollten Programme^{5 6} geschrieben und getestet werden, in welchen die Bewegungen der Aktor-Sensor-Einheit (ASE) auf den elektromechanischen Greifer übermittelt

Einfache Servoansteuerung

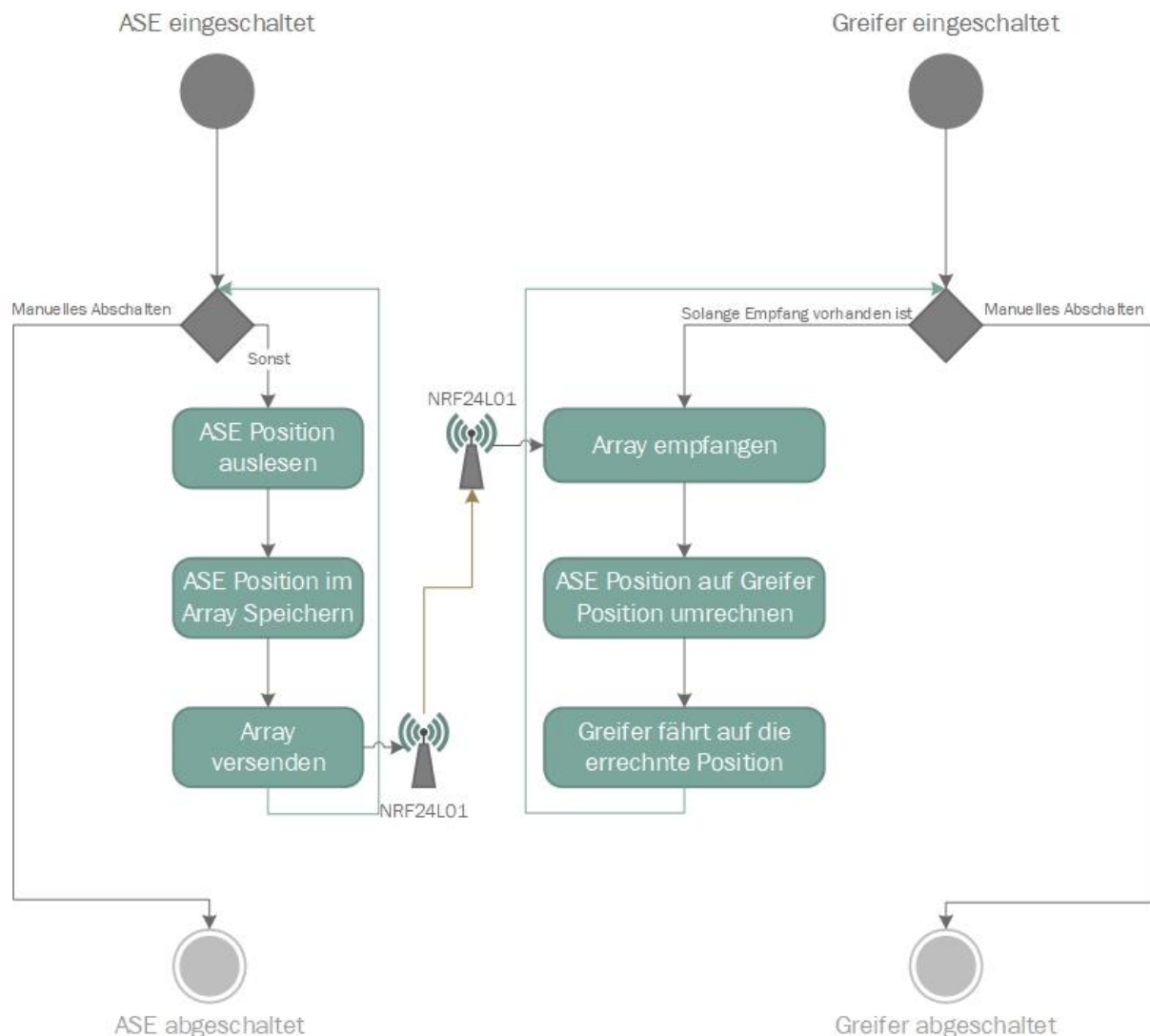


Abbildung 19: Funktionsprinzip der einfachen Ansteuerung des Greifers über die Funkverbindung

werden. Der Greifer muss daraufhin auf die Position der ASE fahren. Im Folgenden Abschnitt werden die wesentlichen Programmteile gezeigt und erklärt - ebenso aufgetretene Probleme und Lösungen. Abbildung 19 veranschaulicht das Funktionsprinzip der einfachen Ansteuerung des Greifers über die Funkverbindung.

Im Programm der ASE für die Ansteuerung der Servomotoren des elektromechanischen Greifers wird für den Kommunikationsaufbau zwischen dem Servomotor der ASE und dem Mikrocontroller zunächst eine Baudrate eingestellt. Hier gibt es zwei Einstellwerte, die gewählt werden können: 115200 Baud pro Sekunde und

1000000 Baud pro Sekunde. Bei Tests und durch Recherchen ist festgestellt worden, dass eine Baudrate von 115200 Baud pro Sekunde das bestmögliche Ergebnis erzielt. Bei 1000000 Baud pro Sekunde kommt es vereinzelt zu kurzzeitigen Kommunikationsstörungen.

Um die ASE bewegen zu können muss das Drehmoment deaktiviert werden. Bei eingeschaltetem Drehmoment ist der Anker unbeweglich und hält seine Position. Des Weiteren wird der Operationsmodus des Servomotors ausgewählt: Es gibt einen Wheel-Mode und einen Position-Mode. Beim Wheel-Mode wird der Servomotor in einen einfachen Fahrbetrieb versetzt, in welchem er lediglich als Antrieb verwendet werden kann. Im Position-Mode wird der Servomotor über Inkremente auf die Positionen gefahren. Dieser Position-Mode wird für den benötigten Betrieb verwendet. Abbildung 20 zeigt den Programmteil, in dem die Grundeinstellungen des Servomotors der ASE vorgenommen werden.

```
ASE.begin(115200); // Beginn der Kommunikation zwischen
                  ASE und Arduino Uno mit eingestellter
                  Baudrate

ASE.setPortProtocolVersion(ASE_PROTOCOL_VERSION); // Setzen der Protokoll Version
ASE.torqueOff(ASE_ID); // Deaktivierung des Servos um ihn
                      bewegen zu können

ASE.setOperatingMode(ASE_ID, OP_POSITION); // Setzen des Operationsmodus
                                           (Fahrbetrieb über Position)
```

Abbildung 20: Grundeinstellungen des ASE-Servomotors

Nachdem die Grundeinstellungen des ASE-Servomotors vorgenommen und das Funkmodul aktiviert wurden, wird in dem Loop-Teil des Programms die ASE-Position mittels entsprechender Methode ausgelesen, in dem Array des Sendedatenspeichers abgelegt und versendet. Bei den Dynamixel AX-12A Servomotoren werden Rohdaten ausgegeben, um die aktuelle Position des Ankers zu bestimmen. Der Bereich dieser Rohdaten liegt zwischen 0 bis 1023. Auch die Eingabe für eine Zielposition des Ankers wird über die beschriebenen Rohdaten gesetzt. Zusätzlich wird die ID des ASE-Servomotors in den Sendedatenspeicher gelegt, um zwei unterschiedliche Werte zu übermitteln. Programmtechnisch zu sehen ist das auf der Abbildung 21.

```
int Daten[2]={ASE_ID,ASE.getPresentPosition(1)}; // Speichern der ASE ID und der aktuellen
                                                  ASE Position im Sendedatenspeicher

ASE_Funk.write(&Daten, sizeof(Daten)); // Datengröße wird ermittelt und aus
                                       vorgesehenem Speicher ausgelesen und
                                       gesendet
```

Abbildung 21: Auslesen und Übertragung der ASE-Position

In dem Programm des elektromechanischen Greifers werden wie bei der ASE zunächst die Grundeinstellungen vorgenommen und erstmalig der Empfang gestartet. Da der Greifer über zwei Servomotoren verfügt, müssen für jeden einzelne Objekte instanziiert und entsprechend eingestellt werden. In dem Programm werden diese Objekte als Gripper1 und Gripper2 bezeichnet. Abbildung 22 stellt den Programmteil dar:

```
Greifer_Funk.begin();           // Aktivierung des NRF24L01 Funkmoduls
Greifer_Funk.setAutoAck(1);     // Aktivierung der Acknowledge_Funktion
Greifer_Funk.setRetries(0,15);  // Die Zeit zwischen den Versuchen den Empfänger
                                // zu erreichen, Anzahl der Versuche

Greifer_Funk.enableAckPayload(); // Erlaubt das Senden der Daten auf die Anfrage
                                // des Empfängers

Greifer_Funk.setPayloadSize(32); // Packetgröße in Byte
Greifer_Funk.openReadingPipe(1,address[0]); // Öffnen einer Pipe zum Empfangen
Greifer_Funk.setChannel(45);    // Einstellung der Sende-/Empfangsfrequenz
Greifer_Funk.setPALevel (RF24_PA_MAX); // Einstellung der Sendeleistung
Greifer_Funk.setDataRate (RF24_2Mbps); // Einstellung der Datenübertragungsrate
Greifer_Funk.powerUp();         // Arbeitsbeginn
Greifer_Funk.startListening();  // Startet den Empfang
```

Abbildung 22: Grundeinstellungen der Greifer-Servomotoren

Nach der Startroutine werden die empfangenen Daten, ASE-ID und ASE-Position in dem vorher deklarierten Array des Empfangsdatenspeichers abgelegt. Da die Position der ASE nicht direkt auf die Servomotoren des Greifers übertragen werden können, muss diese erst umgerechnet werden. Wird die ASE durch die Hand des Bedieners gedrückt, werden Positionswerte sinkend in einem Bereich zwischen 820 (ASE ist auf) bis 500 (ASE ist zu) ausgegeben. Um den elektromechanischen Greifer einfahren zu können müssen diese Positionswerte aufsteigend in einem Bereich zwischen 356 (Greifer ist auf) bis 522 (Greifer ist zu) übermittelt werden. Um die Greiffinger nicht komplett aufeinander zufahren zu lassen, wurde ein maximaler Wert von 512 als Rahmen für die Position gesetzt. Die umgerechnete ASE-Position wird als Zielposition in die jeweilige Methode zum Fahren der Greiffinger übergeben. Abbildung 23 zeigt diesen Programmteil des Greifers.

```

int Daten[2]; // Deklaration des Empfangsdatenspeichers
while( Greifer_Funk.available(&PipeNo)){ // Solange Daten empfangen werden
    Greifer_Funk.read( &Daten, sizeof(Daten) ); // Datengröße wird ermittelt und im
                                                vorgesehenem Speicher abgelegt

    int Empfangen1=Daten[0]; // Speichern der empfangenen ASE ID
    int Empfangen2=Daten[1]; // Speichern der empfangenen ASE Position

    Position= 512-(Empfangen2-512); // Rechnet die empfangene ASE Position um
                                    für die Greifer Position

    Gripper1.setGoalPosition(Greifer_ID, Position ); // Gripper1 fährt auf die umgerechnete
                                                        Position
    Gripper2.setGoalPosition(Greifer_ID, Position ); // Gripper2 fährt auf die umgerechnete
                                                        Position

```

Abbildung 23: Umrechnung der ASE-Position und Fahrbetrieb des Greifers

Die kompletten Programme für die einfache Ansteuerung des Greifers über die Funkverbindung befinden sich im Anhang.

7.3.1 Problematik durch ungleichmäßiges Fahrverhalten

Bei den ersten Versuchen, den elektromechanischen Greifer durch die ASE fernsteuern zu können, wurde bemerkt, dass die Servomotoren des Greifers öfter Ausfälle aufweisen. Dieses ungleichmäßige Fahrverhalten machte einen Betrieb des elektromechanischen Greifers unmöglich.

7.3.2 Lösungsansatz und Umsetzung

Um der Fehlerursache auf den Grund gehen zu können, sollten die empfangenen Positionsdaten der ASE und deren Umrechnung für den Greifer angezeigt werden.

Dazu wurde eine serielle Ausgabe einprogrammiert, um zu prüfen, ob die ASE-Position übermittelt und korrekt für den Betrieb des elektromechanischen Greifers umgerechnet wird. Es wurde festgestellt, dass die ASE-Position am Greifer ankommt und die Funktion der Umrechnung richtig umgesetzt wurde.

Als nächstes wurde ein Programm geschrieben, um das Fehlverhalten der Servomotoren des Greifers im Betrieb genau analysieren zu können. Dazu wurden die

Greiffinger in festgelegten Zeitabständen auf- und wieder zufahren gelassen. Der Quellcode dieses Programnteils ist auf der Abbildung 24 zu sehen.

```
Gripper1.setGoalPosition(Greifer_ID, 512);    // Gripper1 fährt zu
Gripper2.setGoalPosition(Greifer_ID, 512);    // Gripper2 fährt zu
delay(1000);                                  // Wartezeit
Gripper1.setGoalPosition(Greifer_ID, 356);    // Gripper1 fährt auf
Gripper2.setGoalPosition(Greifer_ID, 356);    // Gripper2 fährt auf
delay(1000);                                  // Wartezeit
```

Abbildung 24: Testprogramm für die Fehlersuche der einfachen Ansteuerung des Greifers

Auch hier waren die unregelmäßigen Ausfälle des Fahrbetriebs zu erkennen. Um sich die Eigenschaften der einzelnen Servomotoren anzuschauen, wurde das Betriebsprogramm DYNAMIXEL Wizard 2.0, passend für die Dynamixel AX-12A Servomotoren, installiert und angewandt. Es wurde erkannt, dass keine fortlaufenden Identifikationsnummern vergeben waren. Alle ID's der Servos sind werksseitig auf 1 gesetzt. Abbildung 25 zeigt die Benutzeroberfläche des DYNAMIXEL Wizard 2.0 Programms und die darauf zu erkennende ID des Servomotors.

Address	Item	Decimal	Hex	Actual
0	Model Number	12	0x000C	AX-12A
2	Firmware Version	24	0x18	
3	ID	1	0x01	ID 1
4	Baud Rate (Bus)	16	0x10	115200 bps = 2M / (16 + 1)
5	Return Delay Time	127	0x7F	254 [μsec]
6	CW Angle Limit	0	0x0000	0.00 [°]
8	CCW Angle Limit	1023	0x03FF	300.00 [°]
11	Temperature Limit	70	0x46	70 [°C]
12	Min Voltage Limit	60	0x3C	6.00 [V]
13	Max Voltage Limit	140	0x8C	14.00 [V]
14	Max Torque	1023	0x03FF	100.00 [%]
16	Status Return Level	2	0x02	Return for all
17	Alarm LED	36	0x24	
18	Shutdown	36	0x24	
24	Torque Enable	0	0x00	OFF
25	LED	0	0x00	OFF
26	CW Compliance Margin	1	0x01	0.29 [°]
27	CCW Compliance Marg_	1	0x01	0.29 [°]
28	CW Compliance Slope	32	0x20	

AX-12A

Factory Reset Reboot

Joint: Joint

Torque: ☐

LED: ☐

Position: 147.21 [°]
 Speed: 0.00 [rev/min]
 Load: 0.00 [%]
 Temperature: 31 [°C]
 Voltage: 12.00 [V]

Model Number

Decimal: 12
 Hex: 0x000C
 Actual: AX-12A




Abbildung 25: Ausschnitt des Programms DYNAMIXEL Wizard 2.0

Unterschiedliche ID's der Servomotoren wurden vergeben. Greiffinger1 bekam die ID1, Greiffinger2 ID2 und die ASE die ID3. Nach erneuten Versuchen ergab sich eine fehlerfreie Ansteuerung des elektromechanischen Greifers durch die ASE. Videoaufnahmen zur Veranschaulichung des Problems und der Lösung sind dem Datenträger im Anhang der Dokumentation beigelegt.

7.4 Force-Feedback-Funktion

Der Grundgedanke zur Realisierung der Force-Feedback-Funktion war der permanente Vergleich von den Lastwerten der ASE und des elektromechanischen Greifers. Zuerst sollten beide Geräte sich über die Übermittlung der Positionswerte und Lastwerte durch die Funkübertragung synchronisieren, um ihre Positionen abzugleichen und die Lastwerte zu empfangen. Daraufhin müssten die Werte der Lasten von der ASE und dem Greifer miteinander verglichen werden. Solange die Last der ASE größer ist als die des Greifers, muss die Position der ASE um ein Inkrement verkleinert oder vergrößert werden, dann den aktuellen Wert der Position an den Greifer senden, um den auf die neue Position fahren zu lassen. Danach muss der Greifer seine neue Position an die ASE zurücksenden. Die ASE fährt auf die vom Greifer empfangene Position. Dies führt zur kontinuierlichen Synchronisation zwischen der ASE und dem Greifer. Ob die ASE ihre Position um ein Inkrement verkleinert oder vergrößert, bestimmt die Lastrichtung. Die Lastrichtung wird durch eine Drehbewegung nach rechts (ASE wird geöffnet) oder durch eine Drehbewegung nach links (ASE wird geschlossen) des ASE-Servomotors bestimmt. Wenn ein Objekt gegriffen wird, wird die Last am Greifer größer und die ASE hält solange ihre Position, bis die eigene Last durch festeres Drücken wieder größer wird. Bei der Umsetzung der Force-Feedback-Funktion ergaben sich mehrere Versionen, um zum einen regelmäßige Backups zu erstellen und zum anderen um Fortschritte oder Probleme genau abzustecken. Im folgenden Abschnitt werden die wesentlichen Programmteile der Force-Feedback-Funktion gezeigt und erklärt. Auch aufgetretene Probleme und deren Lösungen werden beschrieben. Abbildung 26 zeigt das Funktionsprinzip der gesamten Force-Feedback-Funktion.

Force-Feedback-Funktion

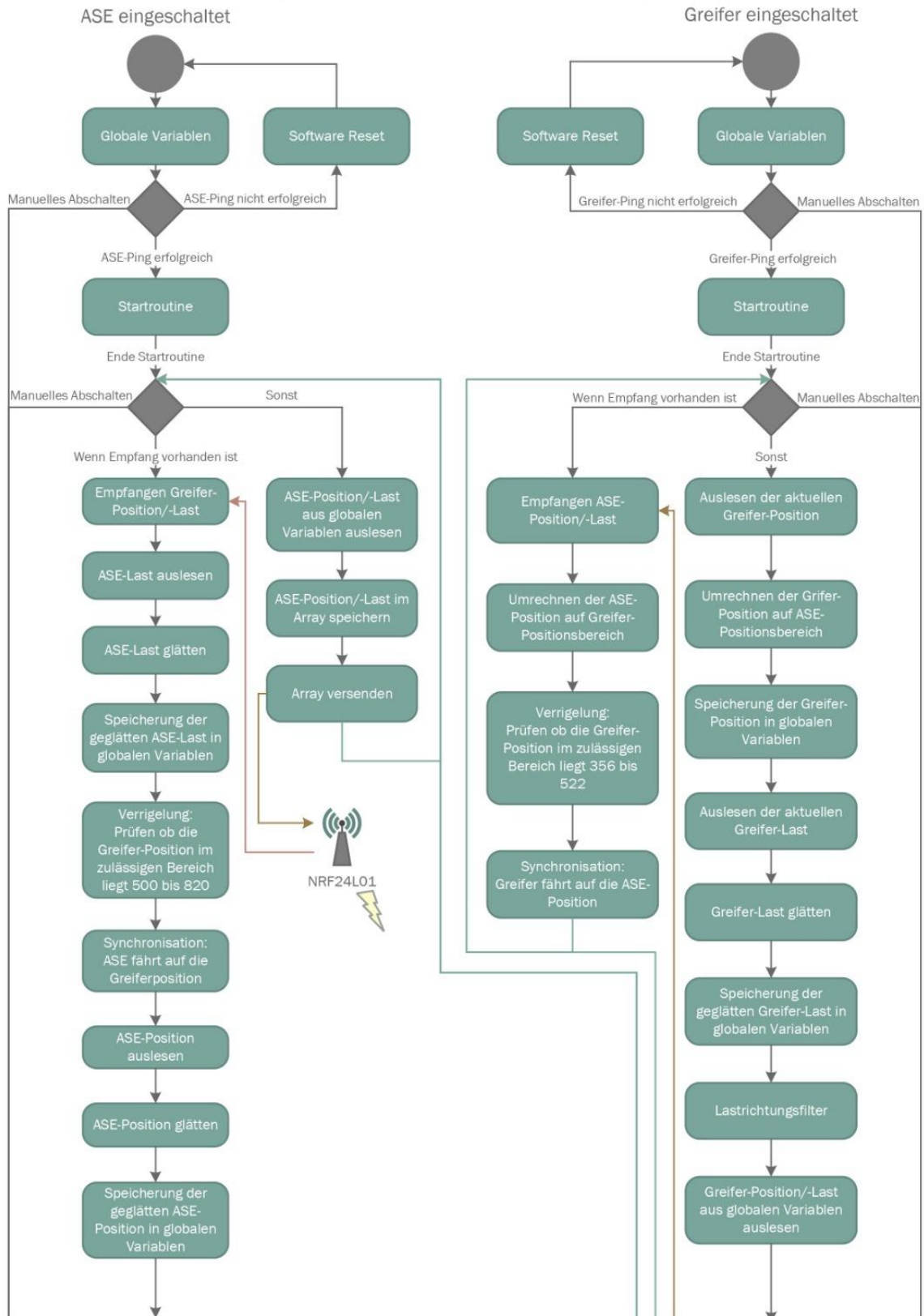


Abbildung 26: Funktionsprinzip der Force-Feedback-Funktion (1)

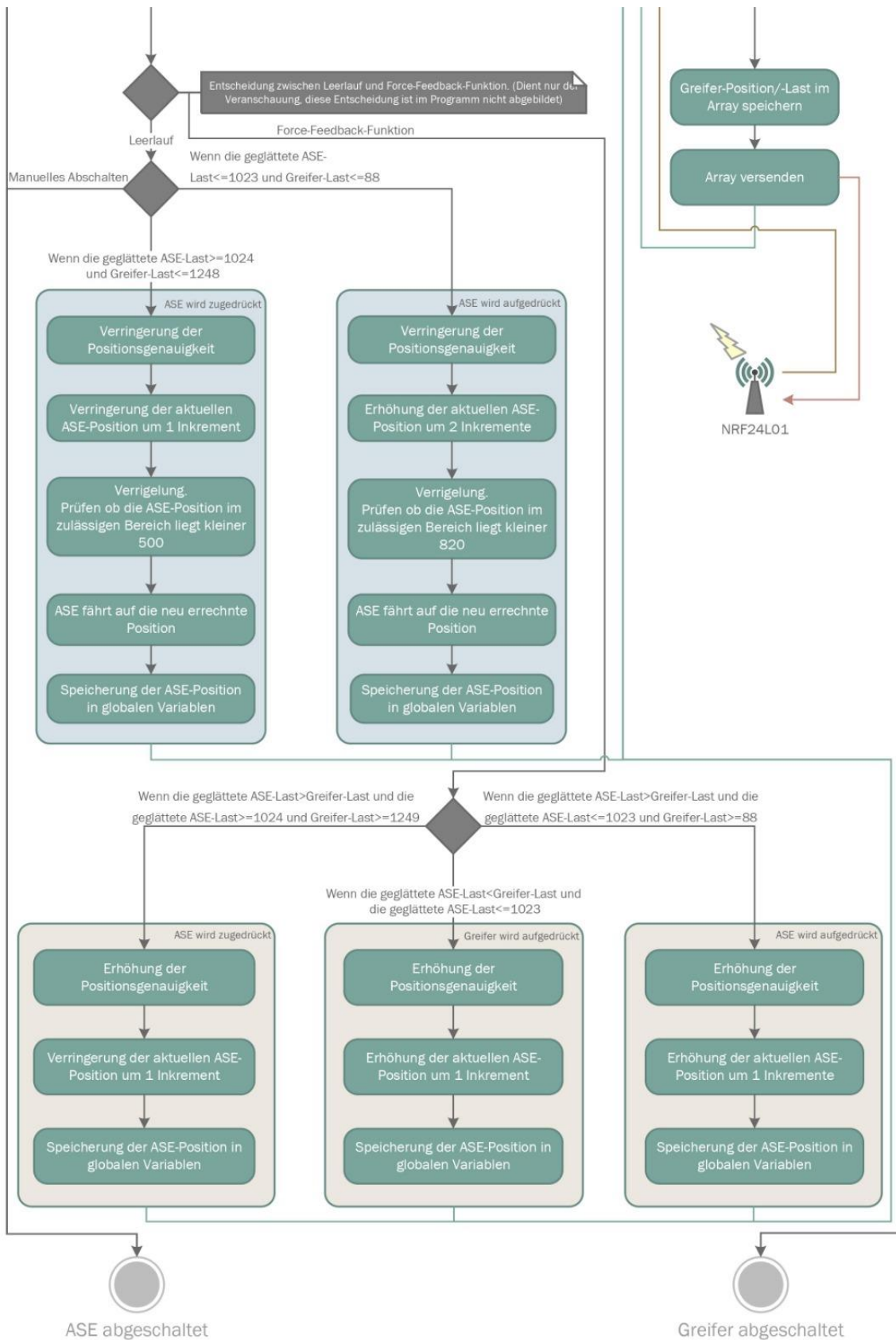


Abbildung 27: Funktionsprinzip der Force-Feedback-Funktion (2)

Im Programm der ASE werden zunächst die empfangenen Positions- und Lastwerte des Greifers zur Kontrolle auf dem seriellen Monitor ausgegeben und anschließend in den Empfangsspeicherspeicher gelegt. Wie in dem vorherigen Abschnitt der einfachen Ansteuerung des Greifers über die Funkverbindung schon beschrieben wurde, bewegen sich die Positionswerte der Servomotoren in einem Bereich von 0 bis 1023. Die Lastwerte dagegen ändern sich je nach Lastrichtung. Wird der Anker des Servomotors nach links (ASE auf) gedrückt, befinden sich die Werte je nach Stärke der einwirkenden Kraft in einem Bereich von 0 bis 1023. Wird der Anker des Servomotors nach rechts (ASE zu) gedrückt, bewegen sich die Werte im Bereich von 1024 bis 2047. Die Positions- und Lastwerte der ASE werden ebenfalls ausgelesen. Vor dem Vergleich der beiden Lasten findet zunächst die Synchronisation der Positionen von der ASE und dem Greifer statt. Dafür entnimmt die ASE die Position des Greifers aus dem Empfangsspeicherspeicher und fährt darauf. Danach wird verglichen, ob die Last der ASE größer ist als die des Greifers und in welche Richtung die ASE gedrückt wird (auf oder zu). Wird die ASE zugeedrückt, wird von der eigenen aktuellen Position ein Inkrement subtrahiert, als neue ASE-Position definiert und in den Sendespeicherspeicher gelegt. Wird die ASE aufgedrückt, verhält es sich andersherum und ein Inkrement wird auf die eigene aktuelle Position hinzuaddiert. Diese Funktionen sind nötig, um einen Fahrbetrieb des Servomotors zu erreichen. Im vorherigen Abschnitt der einfachen Ansteuerung des Greifers über die Funkverbindung wurde das Drehmoment abgeschaltet, um den Anker des Servos frei bewegen zu können. Da das Prinzip der Force-Feedback-Funktion auf dem Vergleich der Lasten zwischen der ASE und dem Greifer basiert, muss das Drehmoment eingeschaltet werden, um die Lastwerte auslesen zu können. Die neue ASE-Position und ihre aktuelle Last werden an den Greifer gesendet und zur Kontrolle ebenfalls auf dem seriellen Monitor ausgegeben. Der Programmteil der ASE wird auf Abbildung 28 gezeigt.


```

Serial.println("-----"); // Serielle Ausgabe des Textes
Serial.print("Position Greife Empfangen= "); // Serielle Ausgabe des Textes
Serial.println(Daten_Empfangen[0]); // Serielle Ausgabe des
                                     Empfangdatenspeichers
Serial.print("Last Greifer Empfangen= "); // Serielle Ausgabe des Textes
Serial.println(Daten_Empfangen[1]); // Serielle Ausgabe des
                                     Empfangdatenspeichers
Serial.println("-----"); // Serielle Ausgabe des Textes

Greifer_Last = Daten_Empfangen[1]; // Speichern der empfangenen Greifer
                                     Last
Greifer_Position = Daten_Empfangen[0]; // Speichern der empfangenen Greifer
                                     Position

ASE.setGoalPosition(3,Greifer_Position); // ASE fährt auf die empfangene
                                     Greifer Position
ASE_Position = ASE.getPresentPosition(3); // Speichern der aktuellen ASE
                                     Position
ASE_Last = ASE.readControlTableItem (PRESENT_LOAD,3,Timeout); //Auslesen und speichern
                                     der aktuellen ASE Last

//===== ASE wird zugeedrückt =====
if (ASE_Last > Greifer_Last && ASE_Last >= 1024){ // Prüft, ob die ASE Last größer als
                                                    die Greifer Last ist und ob die
                                                    ASE im Lastbereich zu ist
    ASE_Position=ASE_Position-1; // Berechnung der neuen ASE Position
}

//===== ASE wird aufgedrückt =====
if (ASE_Last > Greifer_Last && ASE_Last <= 1023){ // Prüft, ob die ASE Last größer als
                                                    die Greifer Last ist und ob die
                                                    ASE im Lastbereich auf ist
    ASE_Position=ASE_Position+1; // Berechnung der neuen ASE Position
}

Serial.print("Position ASE Senden= "); // Serielle Ausgabe des Textes
Serial.println(Daten_Senden[0]); // Serielle Ausgabe des Sendedatenspeichers
Serial.print("Last ASE Senden= "); // Serielle Ausgabe des Textes
Serial.println(Daten_Senden[1]); // Serielle Ausgabe des Sendedatenspeichers

```

Abbildung 28: Force-Feedback-Funktion. Programmteil der ASE (1)

Im Programm des Greifers werden erst wie im Programm der ASE die empfangenen Daten zur Kontrolle der Funkübertragung auf dem seriellen Monitor ausgegeben. Dann werden die Positionsdaten der ASE für den Greifer umgerechnet. Der Greifer fährt auf die neue Zielposition, rechnet seine eigene aktuelle Position wieder für die ASE um und legt diese zum Senden in den Sendedatenspeicher. Für die aktuelle Last des Greifers wird ein Mittelwert der Lastwerte vom Gripper1 und Gripper2 gebildet und in den Sendedatenspeicher gelegt. Die neuen Positions- und Lastwerte des Greifers werden an die ASE versendet. Zur Kontrolle des Sendevorgangs werden auch diese

Daten auf dem seriellen Monitor ausgegeben. Abbildung 29 zeigt den beschriebenen Programmteil des Greifers.

```

Serial.println("-----");           // Serielle Ausgabe des Textes
Serial.print("Position ASE Empfangen= "); // Serielle Ausgabe des Textes
Serial.println(Daten_Empfangen[0]);     // Serielle Ausgabe des
                                        Empfangdatenspeichers

Serial.print("Last ASE Empfangen=      "); // Serielle Ausgabe des Textes
Serial.println(Daten_Empfangen[1]);     // Serielle Ausgabe des
                                        Empfangdatenspeichers

Serial.println("-----");           // Serielle Ausgabe des Textes


int Position= 512-(Daten_Empfangen[0]-512); // Rechnet die empfangene ASE Position um für
                                        die Greifer Position


Gripper1.setGoalPosition(1,Position);     // Gripper1 fährt auf die umgerechnete
                                        Position

Gripper2.setGoalPosition(2,Position);     // Gripper2 fährt auf die umgerechnete
                                        Position


Greifer_Position = 1024-Gripper1.getPresentPosition(1); // Rechnet die aktuelle Position des
                                        Greifers um für die ASE


int Last= (Gripper1.readControlTableItem
(PRESENT_LOAD,1,Timeout)+Gripper2.readControlTableItem(PRESENT_LOAD,2,Timeout))/2;
                                        // Bildet den Mittelwert der Lasten
                                        vom Gripper1 und vom Gripper2


Serial.print("Position Greifer Senden= "); // Serielle Ausgabe des Textes
Serial.println(Daten_Senden[0]);           // Serielle Ausgabe des Sendedatenspeichers
Serial.print("Last Greifer Senden=      "); // Serielle Ausgabe des Textes
Serial.println(Daten_Senden[1]);           // Serielle Ausgabe des Sendedatenspeichers

```

Abbildung 29: Force-Feedback-Funktion. Programmteil des Greifers (1)

Die ASE empfängt die neuen Daten des Greifers, synchronisiert sich mit der Position und startet erneut den Lastenvergleich. Dieser Ablauf verläuft in ständiger Wiederholung.

Ein Funktionstest ergab eine Kraftrückkopplung an der ASE, sobald sich die Last am Greifer, z.B. durch greifen eines Objektes, erhöht hat. Somit war das grundsätzliche Ziel der Force-Feedback-Funktion erreicht.

7.4.1 Problematik durch erschwerte Bedienung der Force-Feedback-Funktion

Der Betrieb der ASE und des Greifers war sehr schwergängig und stockend. Die ASE musste relativ feste gedrückt werden, um bewegt zu werden. Der Fahrbetrieb verlief

ebenfalls sehr unregelmäßig mit unterschiedlichen Pausenzeiten zwischen den Positionswechseln.

7.4.2 Lösungsansatz und Umsetzung

Um die Force-Feedback-Funktion verbessern zu können, musste der komplette Fahrbetrieb erleichtert werden. Ein Lösungsansatz war die Überprüfung der Lastvergleiche.

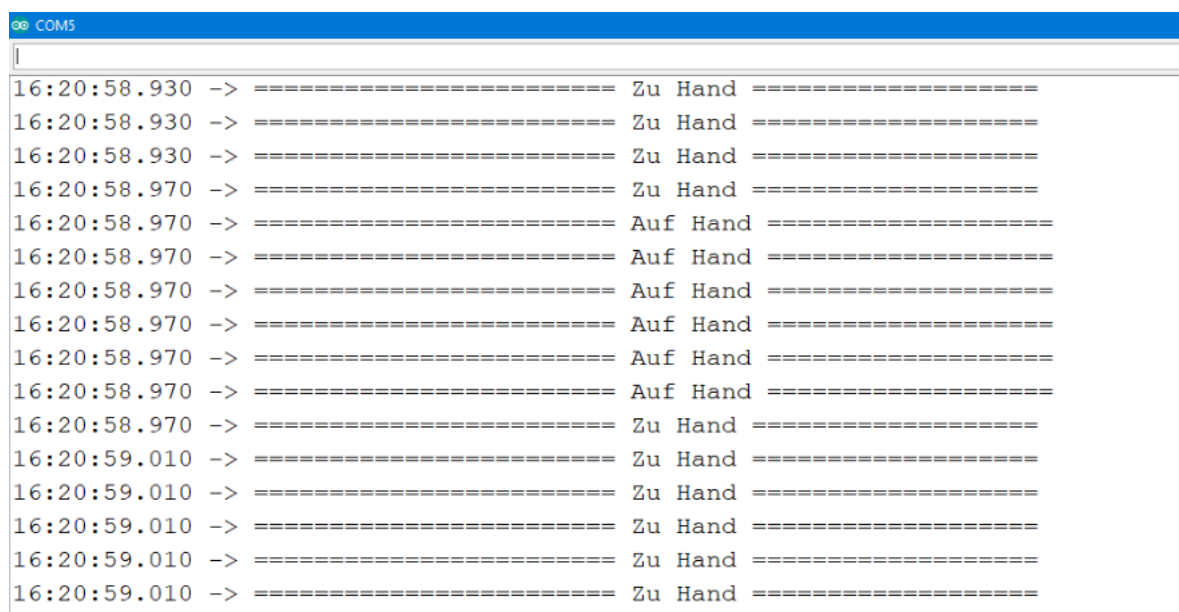
Dazu wurde eine entsprechende Ausgabe in der jeweiligen IF-Bedingung einprogrammiert, um zu ermitteln, in welcher das Programm sich im Betrieb befindet. Auf Abbildung 30 ist die zusätzliche Funktion der seriellen Ausgabe in dem ASE-Programm dargestellt.

```
if (ASE_Last > Greifer_Last && ASE_Last >= 1024)                // ASE wird geschlossen
{
    Serial.println("===== Zu Hand ====="); // Serielle
                                                    Ausgabe ASE
                                                    wird
                                                    gedrückt

    if (ASE_Last > Greifer_Last && ASE_Last <= 1023)            // ASE wird geöffnet
    {
        Serial.println("===== Auf Hand ====="); // Serielle
                                                        Ausgabe ASE
                                                        wird
                                                        gedrückt
    }
}
```

Abbildung 30: Force-Feedback-Funktion. Programmteil der ASE (2)

Abbildung 31 zeigt die serielle Ausgabe des veränderten ASE-Programms.



```
COM5
16:20:58.930 -> ===== Zu Hand =====
16:20:58.930 -> ===== Zu Hand =====
16:20:58.930 -> ===== Zu Hand =====
16:20:58.970 -> ===== Zu Hand =====
16:20:58.970 -> ===== Auf Hand =====
16:20:58.970 -> ===== Auf Hand =====
16:20:58.970 -> ===== Auf Hand =====
16:20:58.970 -> ===== Auf Hand =====
16:20:58.970 -> ===== Auf Hand =====
16:20:58.970 -> ===== Zu Hand =====
16:20:59.010 -> ===== Zu Hand =====
16:20:59.010 -> ===== Zu Hand =====
16:20:59.010 -> ===== Zu Hand =====
16:20:59.010 -> ===== Zu Hand =====
16:20:59.010 -> ===== Zu Hand =====
```

Abbildung 31: Serielle Ausgabe der Force-Feedback-Funktion 1

7.4.3 Problematik durch unbeständiges Verhalten der Selektionen

Es ergab sich ein ungewollter und ungleichmäßiger Wechsel zwischen den IF-Bedingungen des Lastenvergleichs von der ASE und dem Greifer. Die Servomotoren bekamen innerhalb kurzer Zeiträume die Befehle, ihre Positionswerte zu verringern und zu erhöhen. Daraus resultierte die erschwerte Bedienung der ASE und der stockende Gesamtbetrieb der Force-Feedback-Funktion.

7.4.4 Lösungsansatz und Umsetzung

Die Fehlerquelle der ungewollten Bedingungswechsel des Lastenvergleichs musste gefunden und behoben werden. Hier war der Lösungsansatz, die Positionen und Lasten der ASE und des Greifers während des Betriebs aufzunehmen und in einem Graphen anzeigen zu lassen. Dadurch sollte sich eine Veranschaulichung des Problems ergeben.

Zunächst wurden die Positionswerte und Lastwerte vom Greifer aufgenommen und grafisch dargestellt. Abbildung 32 zeigt das Liniendiagramm der Positions- und Lastaufnahme des Greifers.

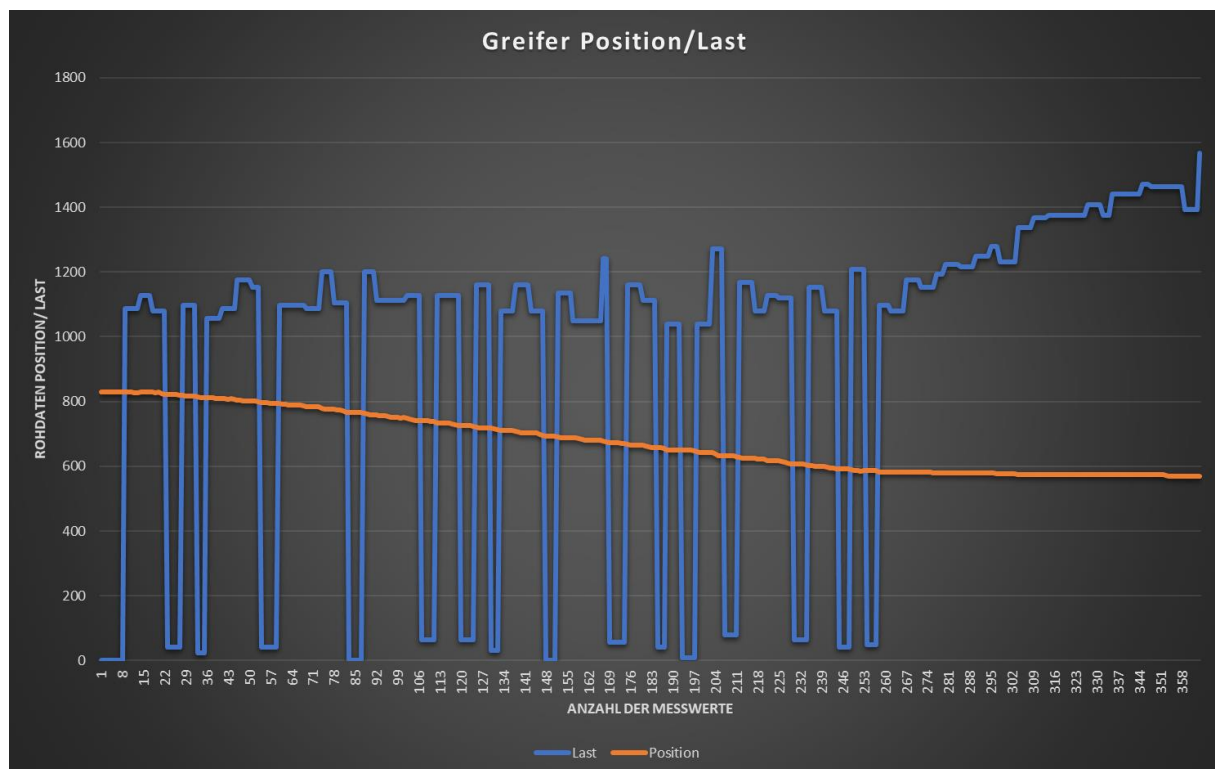


Abbildung 32: Liniendiagramm von den Positions- und Lastwerten des Greifers

7.4.5 Problematik durch ungewollte Nebeneffekte des Greifers

Mithilfe des Liniendiagramms aus der Last und der Position des Greifers wurde festgestellt, dass die Positionswerte sich wie erwartet verhalten. Die Lastwerte dagegen ändern sprunghaft ihre Werte und die Lastrichtung, die sich in den Wertebereichen zwischen 0 bis 1023 und zwischen 1024 und 2047 bewegen. Zusätzlich wurde festgestellt, dass die Bildung des Mittelwertes der Lasten beider Servomotoren des Greifers zur ungewollten Lastrichtungsänderung beiträgt.

7.4.6 Lösungsansatz und Umsetzung

Um einen direkten Vergleich der Betriebsverhalten von der ASE und dem Greifer festzustellen, sollten von beiden Komponenten die Positions- und Lastwerte auf einem Graphen angezeigt werden. Zudem sollte eine Lastglättung das sprunghafte Verhalten der Werte verringern, welches aufgrund des Nachschwingens der Servomotoren mit den Greiffingern im Fahrbetrieb auftritt. Außerdem wurde der Lösungsansatz mit der Bildung des Mittelwertes beider Lasten der Servomotoren verworfen. Für die Auswertung der Lasten wurde lediglich nur ein Servomotor des Greifers berücksichtigt.

Für die Lastglättung wurden in beiden Programmen (ASE und Greifer) die dynamische Funktion eines gleitenden Mittelwertes implementiert. Zusätzlich werden die Werte zur Mittelwertbildung unterschiedlich gewichtet. Eine empirische Ermittlung der Gewichtungsverteilung ergab das bestmögliche Ergebnis bei 80:20. Dies bedeutet eine größere Gewichtung des vorherigen Wertes und eine geringere Gewichtung des nächsten Wertes. In diesem Fall hat der aktuell aufgenommene Wert einen Einfluss von 20% und der Wert davor hat einen Einfluss von 80% für die Mittelwertbildung. Dadurch hat ein stark abweichender Spitzenwert keine größere Auswirkung auf die Bildung des Mittelwertes. Abbildung 33 zeigt die Funktion der Mittelwertbildung im Programm der ASE und auf Abbildung 34 ist die Funktion im Programm des Greifers zu sehen.

```
ASE_Last_Smooth = 0.8 * ASE_Last_Smooth + 0.2 * ASE_Last_Aktuell;    //Bildung des  
                                                                       gleitenden  
                                                                       Mittelwerts von  
                                                                       den gewichteten  
                                                                       Lastwerten der ASE
```

Abbildung 33: Gleitende Mittelwertbildung von den gewichteten Lastwerten der ASE

```
Greifer_Last_Smooth = 0.80 * Greifer_Last_Smooth + 0.20 * Greifer_Last; // Bildung des
                                                                           gleitenden
                                                                           Mittelwerts von
                                                                           den gewichteten
                                                                           Lastwerten des
                                                                           Greifers
```

Abbildung 34: Gleitende Mittelwertbildung von den gewichteten Lastwerten des Greifers

Die benötigten Werte von der ASE und dem Greifer wurden aufgenommen und grafisch dargestellt. Auf Abbildung 35 ist die Darstellung zu sehen.

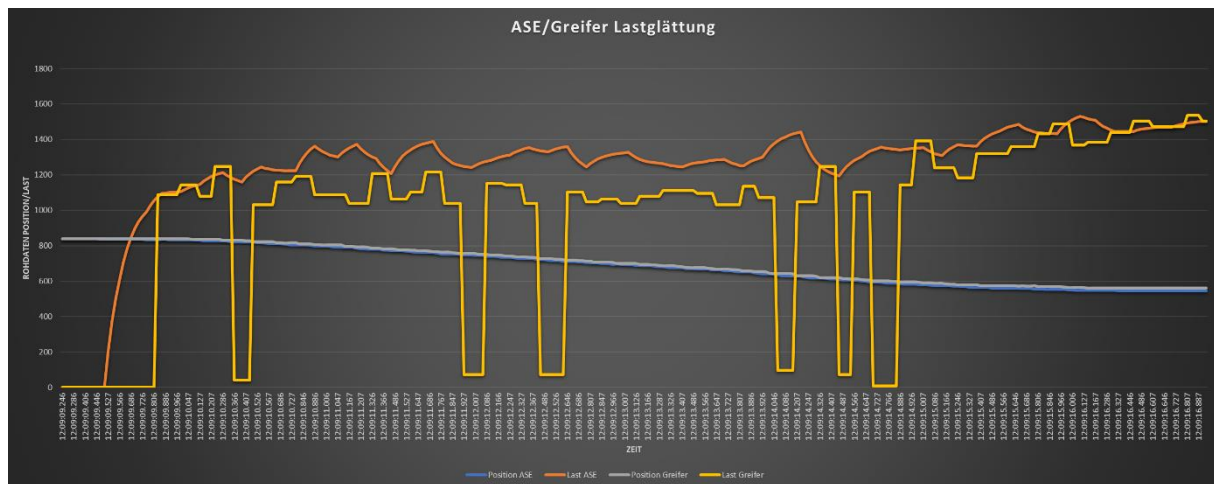


Abbildung 35: Liniendiagramm von den Positions- und Lastwerten des Greifers und der ASE nach der Lastglättung

Zu erkennen ist eine Verbesserung des sprunghaften Verhaltens der Lastwerte des Greifers. Bei der ASE ist wie bei dem Greifer ein korrekter Verlauf der Positionswerte zu erkennen. Die Lastwerte der ASE zeigen ebenfalls ein sich wechselndes Verhalten. Hier ist allerdings zu erkennen, dass die Werte in einer Lastrichtung bleiben. Das sprunghafte Verhalten der ASE ist durch die Lastrichtungsänderung des Greifers und die wechselnde Kraftwirkung des Bedieners auf die ASE zu erklären. Auch zu erkennen ist, dass die Lastrichtungsänderung so lange auftritt, bis ein Objekt gegriffen wurde. Auf Abbildung 35 ist es bei der Zeit von 12.09.14.926, nach dem letzten Ausschlag der Lastwerte vom Greifer in die falsche Richtung zu sehen. Es stellt sich somit ein Betriebszustand im Leerlauf ein und ein Betriebszustand unter Last.

7.4.7 Problematik durch Lastrichtungsänderungen des Greifers

Die Lastrichtungsänderung des Greifers beeinträchtigt die Force-Feedback-Funktion durch den daraus resultierenden stockenden Betrieb des Greifers und der erschwerten Bedienung der ASE, solange noch kein Objekt gegriffen wurde.

7.4.8 Lösungsansatz und Umsetzung

Die Lastrichtungsänderung musste verhindert werden. Zudem musste unterschieden werden zwischen einem Betrieb des Greifers im Leerlauf und unter Last. Dafür sollte für jeden Fall ein entsprechender Filter einprogrammiert werden: Für den Fall der ungewollten Lastrichtungsänderung und um die zwei festgestellten Betriebszustände zu unterscheiden.

Für die Unterscheidung zwischen dem Lastbetrieb und dem Leerlauf wurden zunächst die Grenzen festgelegt. Auf Abbildung 35 ist zu erkennen, dass der Leerlauf sich in einem Bereich von 88 bis 1248 einstellt. Diese Unterscheidungsgrenzen wurden festgelegt und entsprechend in der ASE einprogrammiert. Zusätzlich wurde die Sensibilität der Positionsgenauigkeit im Leerlauf heruntergesetzt, um die Leichtgängigkeit des Fahrbetriebs zu erhöhen. Im Lastbetrieb wird die Sensibilität wieder heraufgesetzt. Abbildung 36 zeigt die Umsetzung der beschriebenen Funktionen im Programm der ASE.

```
//===== Leerlauf ASE Auf =====
    if( ASE_Last_Smooth <= 1023 && Greifer_Last <= 88 )                // Prüft, ob die ASE im
                                                                        richtigen Lastbereich
                                                                        ist und ob der Greifer
                                                                        im Leerlauf
                                                                        ist
    {
        ASE.writeControlTableItem(CW_COMPLIANCE_MARGIN,3,8,Timeout); // Verringerung der
                                                                        Positionsgenauigkeit
                                                                        der ASE (Zu)
        ASE.writeControlTableItem(CCW_COMPLIANCE_MARGIN,3,8,Timeout); // Verringerung der
                                                                        Positionsgenauigkeit
                                                                        der ASE (Auf)
        ASE_Position_Smooth = ASE_Position_Smooth + 2;                // Berechnung der neuen
                                                                        ASE Position
    }
//===== Leerlauf ASE Zu =====
    if( ASE_Last_Smooth >= 1024 && Greifer_Last <= 1248)              // Prüft, ob die
                                                                        ASE im
                                                                        richtigen
                                                                        Lastbereich ist
                                                                        und ob der
                                                                        Greifer im
                                                                        Leerlauf ist
    {
        ASE.writeControlTableItem(CW_COMPLIANCE_MARGIN,3,8,Timeout); // Verringerung
                                                                        der
                                                                        Positionsgenauigkeit
                                                                        der ASE (Zu)
```

```

ASE.writeControlItem(CCW_COMPLIANCE_MARGIN,3,8,Timeout); // Verringerung der
                                                         Positionsgenauigkeit
                                                         der ASE (Auf)

ASE_Position_Smooth = ASE_Position_Smooth - 1;           // Berechnung der neuen
                                                         ASE Position
    }
//===== ASE wird zugeedrückt =====
    if (ASE_Last_Smooth > Greifer_Last && ASE_Last_Smooth >= 1024 && Greifer_Last >=
1249) // Prüft, ob die ASE Last größer ist als die Greifer Last und ob die ASE Last im
    richtigen Lastbereich ist und ob die Greifer Last im Force-Feedback Bereich ist
    {
        ASE.writeControlItem(CW_COMPLIANCE_MARGIN,3,1,Timeout); // Erhöhung der
                                                         Positionsgenauigkeit
                                                         der ASE (Zu)

        ASE.writeControlItem(CCW_COMPLIANCE_MARGIN,3,1,Timeout); // Erhöhung der
                                                         Positionsgenauigkeit
                                                         der ASE (Auf)

        ASE_Position_Aktuell = ASE_Position_Aktuell -1;       // Berechnung der neuen
                                                         ASE Position

        ASE_Position_Senden = ASE_Position_Aktuell;           // Speichern der
                                                         aktuellen ASE Position
    }
//===== Greifer wird aufgedrückt =====
    if (ASE_Last_Smooth < Greifer_Last && ASE_Last_Smooth <= 1023) // Prüft, ob die ASE
                                                         Last kleiner ist als
                                                         die Greifer Last und
                                                         ob die ASE im
                                                         richtigen Lastbereich
                                                         ist
    {
        ASE.writeControlItem(CW_COMPLIANCE_MARGIN,3,1,Timeout); // Erhöhung der
                                                         Positionsgenauigkeit
                                                         der ASE (Zu)

        ASE.writeControlItem(CCW_COMPLIANCE_MARGIN,3,2,Timeout); // Erhöhung der
                                                         Positionsgenauigkeit
                                                         der ASE (Auf)

        ASE_Position_Aktuell = ASE_Position_Aktuell + 1;       // Berechnung der neuen
                                                         ASE Position

        ASE_Position_Senden = ASE_Position_Aktuell;           // Speichern der
                                                         aktuellen ASE
                                                         Position
    }
//===== ASE wird aufgedrückt =====
    if (ASE_Last_Smooth > Greifer_Last && ASE_Last_Smooth <= 1023 && Greifer_Last >=
88) //89 Prüft, ob die ASE Last größer ist als die Greifer Last und ob die ASE Last im
    richtigen Lastbereich ist und ob die Greifer Last im Force-Feedback Bereich ist
    {
        ASE.writeControlItem(CW_COMPLIANCE_MARGIN,3,1,Timeout); // Erhöhung der
                                                         Positionsgenauigkeit
                                                         der ASE (Zu)

```



```

ASE.writeControlItem(CCW_COMPLIANCE_MARGIN,3,1,Timeout); // Erhöhung der
                                                         Positionsgenauigkeit
                                                         der ASE (Auf)

ASE_Position_Aktuell = ASE_Position_Aktuell + 1;          // Berechnung der neuen
                                                         ASE Position

ASE_Position_Senden = ASE_Position_Aktuell;              // Speichern der aktuellen
                                                         ASE Position
}

```

Abbildung 36: Force-Feedback_Funktion mit Leerlauf- und Lastbetrieb im ASE-Programm

Im Programm des Greifers wurde der Filter für die Lastrichtungsänderung realisiert. Hierfür geben die Lastwerte der ASE dem Greifer vor, in welcher Lastrichtung er sich bewegen und welche Ausschläge in die falsche Richtung er herausfiltern soll. Abbildung 37 zeigt den Lastrichtungsfilter im Programm des Greifers.

```

//===== Lastrichtungsfilter =====

if(ASE_Last >= 1024 && Greifer_Last >= 1024)              // Prüft, ob die ASE im Lastbereich
                                                         ZU ist und ob der Greifer im
                                                         Lastbereich ZU ist

{
    Greifer_Last_Alt=Greifer_Last;                      // Setzt die alte Greifer Last auf
                                                         den Wert der aktuellen Greifer
                                                         Last
}

if(ASE_Last >= 1024 && Greifer_Last < 1024)              // Prüft, ob die ASE im Lastbereich
                                                         ZU ist und ob der Greifer im
                                                         Lastbereich AUF ist

{
    Greifer_Last = Greifer_Last_Alt;                    // Setzt die aktuelle Greifer Last
                                                         auf den Wert der alten Greifer
                                                         Last
}

if(ASE_Last <= 1023 && Greifer_Last <= 1023)            // Prüft, ob die ASE im Lastbereich
                                                         AUF ist und ob der Greifer im
                                                         Lastbereich AUF ist

{
    Greifer_Last_Alt = Greifer_Last;                    // Setzt die alte Greifer Last auf
                                                         den Wert der aktuellen Greifer
                                                         Last
}

if(ASE_Last <= 1023 && Greifer_Last >1023)              // Prüft, ob die ASE im Lastbereich
                                                         AUF ist und ob der Greifer im
                                                         Lastbereich ZU ist

{
    Greifer_Last = Greifer_Last_Alt;                    // Setzt die aktuelle Greifer Last
                                                         auf den Wert der alten Greifer
                                                         Last
}
}

```

Abbildung 37: Lastrichtungsfilter im Greifer-Programm

Nach der Umsetzung des Lastrichtungsfilters und der Unterscheidung zwischen dem Leerlauf und dem Lastbetrieb wurden die Daten erneut ausgegeben und visualisiert. In Abbildung 38 ist das Ergebnis zu sehen.

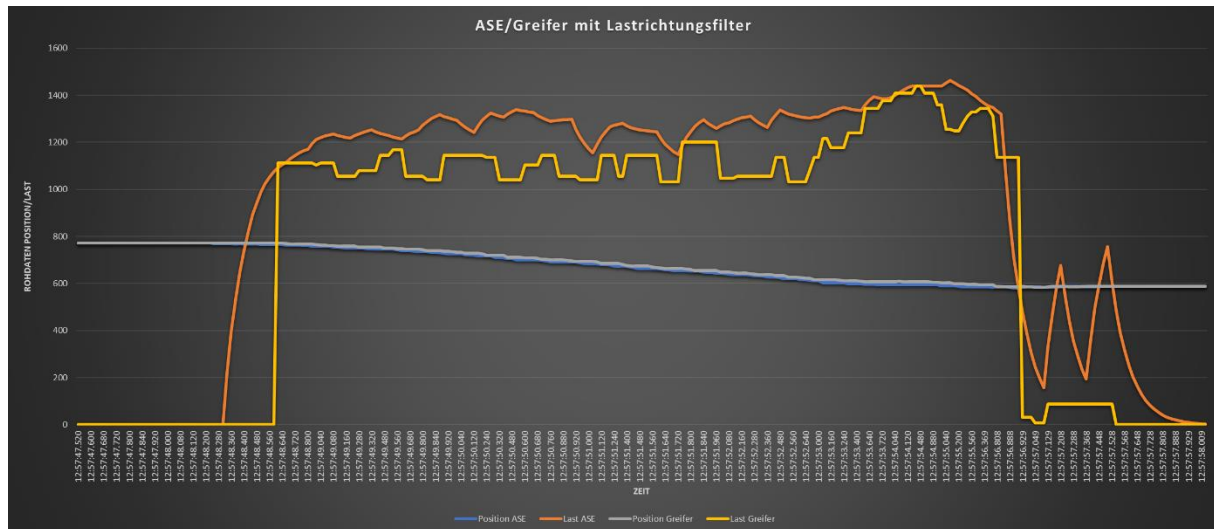


Abbildung 38: Resultat der Force-Feedback-Funktion nach dem Lastrichtungsfilter und der Unterscheidung zwischen Leerlauf und Lastbetrieb

Auf dem Liniendiagramm in Abbildung 38 ist zu erkennen, dass die Force-Feedback-Funktion optimiert wurde. Es findet keine ungewollte Lastrichtungsänderung mehr statt und die Unterscheidung zwischen dem Leerlauf und dem Lastbetrieb wurde verstärkt. Die Glättung der Werte hat zu einer zeitlichen Verzögerung geführt. Diese hat aber keine spürbaren Auswirkungen auf die Echtzeitfähigkeit des Systems. Das Ergebnis ist eine optimierte Funktion der Krafrückkopplung.

7.4.9 Problematik durch fehlende Startroutine und fehlende Sicherheit

Beim Start der ASE und des Greifers findet eine ruckartige Synchronisation statt, wenn die Positionen beider Geräte unterschiedlich sind.

7.4.10 Lösungsansatz und Umsetzung

Um auch eine optimale Startroutine zu realisieren und auch sicherheitsrelevante Aspekte durch die Vermeidung von Verletzungen zu berücksichtigen sollten entsprechende Funktionen in die jeweiligen Programme eingefügt werden.

Dafür wurden sowohl im Programm der ASE als auch im Programm des Greifers Anweisungen eingefügt, die beide Geräte zuerst auf 90° (komplett auf), danach auf 0° (komplett zu) und zum Schluss auf 45° (mittlere Position) fahren lassen. Diese Startroutine wird erst ausgeführt, nachdem die Kommunikation zwischen den AX-12A

Servomotoren und den Arduino Shields hergestellt wurde. Solange keine Kommunikation aufgebaut wurde, wird ein permanenter Software-Reset⁶ durchgeführt. Für diese Startroutine wird von der ASE und dem Greifer sowohl das Drehmoment als auch die Geschwindigkeit auf 20% herabgesenkt. Nachdem die Anweisungen der Startroutine abgearbeitet wurden, werden die Parameter des Drehmoments und der Geschwindigkeit wieder auf das Maximum gesetzt. Abbildung 39 zeigt den Programmteil der Startroutine von der ASE und Abbildung 40 zeigt den Programmteil des Greifers.

```
ASE.writeControlTableItem(PUNCH,3,32,Timeout);           // Herabsetzen der Punch_Funktion
                                                         für einen weicheren Betrieb
ASE.writeControlTableItem(TORQUE_LIMIT,3,204,Timeout);    // Herabsetzen des ASE Drehmoments
                                                         zur Erhöhung der Sicherheit bei
                                                         der Startroutine (20%)
ASE.writeControlTableItem(MOVING_SPEED,3,204,Timeout);    // Herabsetzen der ASE
                                                         Geschwindigkeit zur Erhöhung
                                                         der Sicherheit bei der
                                                         Startroutine (20%)

delay(2000);           // Wartezeit zur Verarbeitung der oben aufgeführten
                       Einstellungen
ASE.setGoalPosition(3,820); // Fahren der ASE auf 90°
delay(2000);           // Wartezeit für den Fahrbetrieb auf 90°
ASE.setGoalPosition(3,500); // Fahren der ASE auf 0°
delay(2000);           // Wartezeit für den Fahrbetrieb auf 0°
ASE.setGoalPosition(3,660); // Fahren der ASE auf 45°
delay(2000);           // Wartezeit für den Fahrbetrieb auf 45°
ASE.writeControlTableItem(TORQUE_LIMIT,3,1023,Timeout);    // Heraufsetzen des ASE
                                                         Drehmoments für den Nennbetrieb
ASE.writeControlTableItem(MOVING_SPEED,3,1023,Timeout);    // Heraufsetzen der ASE
                                                         Geschwindigkeit für den
                                                         Nennbetrieb
```

Abbildung 39: Startroutine von der ASE

```
Gripper1.writeControlTableItem(PUNCH,1,32,Timeout);      // Herabsetzen der
                                                         Punch_Funktion für einen
                                                         weicheren Betrieb vom
                                                         Gripper1
Gripper2.writeControlTableItem(PUNCH,2,32,Timeout);      // Herabsetzen der
                                                         Punch_Funktion für einen
                                                         weicheren Betrieb vom
                                                         Gripper2
Gripper1.writeControlTableItem(TORQUE_LIMIT,1,204,Timeout); // Herabsetzen des Gripper1
                                                         Drehmoments zur Erhöhung
                                                         der Sicherheit bei der
                                                         Startroutine (20%)
Gripper1.writeControlTableItem(MOVING_SPEED,1,204,Timeout); // Herabsetzen der Gripper1
                                                         Geschwindigkeit zur
                                                         Erhöhung der Sicherheit
```

```

// bei der Startroutine
// (20%)
Gripper2.writeControlTableItem(TORQUE_LIMIT,2,204,Timeout); // Herabsetzen des Gripper2
// Drehmoments zur Erhöhung
// der Sicherheit bei der
// Startroutine (20%)
Gripper2.writeControlTableItem(MOVING_SPEED,2,204,Timeout); // Herabsetzen der Gripper2
// Geschwindigkeit zur
// Erhöhung der Sicherheit
// bei der Startroutine
// (20%)

delay(2000); // Wartezeit zur Verarbeitung der oben aufgeführten
// Einstellungen
Gripper1.setGoalPosition(1,356); // Fahren des Gripper1 auf 90°
Gripper2.setGoalPosition(2,356); // Fahren des Gripper2 auf 90°
delay(2000); // Wartezeit für den Fahrbetrieb auf 90°
Gripper1.setGoalPosition(1,522); // Fahren des Gripper1 auf 0°
Gripper2.setGoalPosition(2,522); // Fahren des Gripper2 auf 0°
delay(2000); // Wartezeit für den Fahrbetrieb auf 0°
Gripper1.setGoalPosition(1,439); // Fahren des Gripper1 auf 45°
Gripper2.setGoalPosition(2,439); // Fahren des Gripper2 auf 45°
delay(2000); // Wartezeit für den Fahrbetrieb auf 45°
Gripper1.writeControlTableItem(TORQUE_LIMIT,1,1023,Timeout); // Heraufsetzen des Gripper1
// Drehmoments für den
// Nennbetrieb
Gripper1.writeControlTableItem(MOVING_SPEED,1,1023,Timeout); // Heraufsetzen der Gripper1
// Geschwindigkeit für den
// Nennbetrieb
Gripper2.writeControlTableItem(TORQUE_LIMIT,2,1023,Timeout); // Heraufsetzen des Gripper2
// Drehmoments für den
// Nennbetrieb
Gripper2.writeControlTableItem(MOVING_SPEED,2,1023,Timeout); // Heraufsetzen der Gripper2
// Geschwindigkeit für den
// Nennbetrieb

```

Abbildung 40: Startroutine von dem Greifer

Durch die Funktion dieser Startroutine wurde die Synchronisation von der ASE und dem Greifer optimiert und sicherer gestaltet.

Um eine weitere Sicherheitsfunktion zu implementieren wurden zusätzlich Verriegelungen in die Programme der ASE und des Greifers einprogrammiert. Durch diese Verriegelungen werden maximale Positionsendwerte gesetzt um den Fahrbereich zu begrenzen und Quetschgefahren zu verhindern. Beide Geräte wurden auf maximale Endpositionen von 0° bis 90° verriegelt. Bei der ASE ist das ein Wertebereich von 500 bis 820. Bei dem Greifer sind das Werte von 356 bis 522. Abbildung 41 zeigt den Programmteil der Verriegelung der ASE und Abbildung 42 zeigt den Programmteil des Greifers.

```

if (ASE_Position_Smooth < 500)                // Prüft, ob der maximale Grenzwert der ASE
                                                Position unterschritten wurde
{
    ASE_Position_Smooth = 500;                // Setzt die ASE Position auf den minimalen
                                                Grenzwert
}
if (ASE_Position_Smooth > 820)                // Prüft, ob der maximale Grenzwert der ASE
                                                Position überschritten wurde
{
    ASE_Position_Smooth = 820;                // Setzt die ASE Position auf den maximalen
                                                Grenzwert
}

```

Abbildung 41: Funktion der Positionsverriegelung im ASE-Programm

```

if (Position > 522)                          // Prüft, ob der maximale Grenzwert der Greifer Position
                                                überschritten wurde
{
    Position = 522;                          // Setzt die Greifer Position auf den maximalen Grenzwert
}

if (Position < 356)                          // Prüft, ob der minimale Grenzwert der Greifer Position
                                                unterschritten wurde
{
    Position = 356;                          // Setzt die Greifer Position auf den minimalen Grenzwert
}

```

Abbildung 42: Funktion der Positionsverriegelung im Greifer-Programm

8 Validierung

Bei der Validierung sollte die maximale Reichweite der Funkstrecke, die Umsetzung der Force-Feedback-Funktion, Fehlerrate der Kommunikation sowie die Ergonomie der ASE überprüft werden.

Bei der Überprüfung der maximalen Reichweite der Funkstrecke ergab sich eine Entfernung von ca. 60m (Luftlinie). Diese maximale Reichweite variiert durch Wettereinflüsse und die Beeinflussung der Umgebung in Form von Reflexionen.

Die maximal gemessene Kraft, mit der Objekte gegriffen werden können, beträgt 16,187N. Dabei wurde eine Kraft von 16,677 auf die ASE ausgeübt. Die Differenz von 0,49N ist auf die flexible Beschaffenheit der Greiffinger zurückzuführen.

Für die Validierung der Force-Feedback-Funktion wurden mehrere Objekte gegriffen und aufgenommen. Diese Objekte waren eine Küchenrolle, Toilettenpapier, Desinfektionsmittel, Plastikbecher, Bierglas, Stressball, Apfel, eine Glühbirne, ein rohes Ei, ein bemaltes Ei und ein Feuerzeug. Bei allen Objekten war die

Kraftrückkopplung gegeben. Die Umsetzung der Force-Feedback-Funktion wurde demnach nachweislich umgesetzt. Lediglich das Desinfektionsmittel sowie das bemalte Ei konnten zwar gegriffen, aber nicht aufgenommen werden. Dieser Umstand ergab sich durch die glatten Oberflächenbeschaffenheiten der Greiffinger, des Plastikbehälters des Desinfektionsmittels sowie der bemalten Eierschale. (Das Videomaterial der Validierung befindet sich auf dem Datenträger im Anhang der Dokumentation.) Auch das Gewicht des Objekts spielte eine entscheidende Rolle. Die erwartete Echtzeitfähigkeit zeigte sich durch keine spürbaren Verzögerungen zwischen der Bedienung der ASE und der Ausführung des elektromechanischen Greifers.

Durch eine gesamte Betriebszeit des Systems und der aufgetretenen Fehler in Form von kurzzeitigen Aussetzern der Funkübertragung ergab sich eine Fehlerrate von etwa 1 Fehler pro Stunde.

Aufgrund der Fixierungen der Finger und des Daumens an der ASE durch die Klettbänder und die Trägerplatte ergibt sich eine angenehme Bedienung. Auch die Bewegungsfreiheit der Hand ist durch die Gewichtsverteilung des Servomotors, der in der Handfläche liegt optimal gegeben und nicht beeinträchtigt.

9 Schlussbetrachtung

Das Projektziel war die Realisierung eines elektromechanischen Greifers mit Kraftrückkopplung (Force-Feedback) auf die Hand des Gerätebedieners. Dafür war die Erstellung von zwei Hauptkomponenten notwendig. Zum einen der elektromechanische Greifer, um Objekte greifen und aufnehmen zu können, und zum anderen das Bedienelement, das der Steuerung des elektromechanischen Greifers und dessen Haptik dient. Die zwei Hauptkomponenten sollten mittels geeigneter Servomotoren umgesetzt werden. Eine Datenübertragung musste auf eine Mindestentfernung von 10m sichergestellt werden und eine Echtzeitfähigkeit aufweisen können. Zudem wurde eine drahtlose Verbindung gefordert. Die Datenverarbeitung musste über zwei geeignete Mikrocontroller realisiert werden.

Unter Berücksichtigung der Vorkenntnisse im Bereich der Robotik, der zur Verfügung gestellten Bearbeitungszeit und in Anbetracht der Validierungsergebnisse wurde das

Ziel zur Realisierung des elektromechanischen Greifers mit Force-Feedback-Funktion vollumfänglich und wie geplant erreicht. Ebenso wurden die Ressourcenplanung sowie die Kostenplanung eingehalten. Lediglich der Zeitplan des Projektes musste aufgrund von Verzögerungen angepasst werden. Diese Verzögerungen wurden durch defekte Mikrocontroller, fehlerbehaftete Leitungen, daraus resultierende Fehlersuchen, Nachbestellungen und Fehlerbehebungen verursacht.

Gewonnene Erfahrungen und Erkenntnisse, die während der Bearbeitung des Projektes gesammelt wurden, haben zu einem erfolgreichen Abschluss beigetragen. Dieser Abschluss wurde unter anderem durch die Tatsache erreicht, dass auf Kompromisse zwischen der Geschwindigkeit und der Präzision des Force-Feedback-Systems eingegangen werden mussten.

10 Ausblick

Obwohl es sich um ein voll funktionsfähiges System der Force-Feedback-Funktion handelt, ist dennoch Optimierungspotential vorhanden: Unter anderem die Verschlüsselung der Funkübertragung zur Erhöhung der IT-Sicherheit, die Verwendung eines Kalman-Filters zur Erhöhung der Gesamtstabilität und Geschwindigkeit für die Funktionsvoraussetzung der Echtzeitfähigkeit sowie eine dynamische Anpassung der Unterscheidungsgrenzen zwischen dem Leerlauf und dem Lastbetrieb des Systems in Abhängigkeit der ASE-Geschwindigkeit.

Um das System vor Umwelteinflüssen wie Staub, Feuchtigkeit und Temperaturschwankungen zuverlässig schützen zu können, ist die Anfertigung eines passenden Gehäuses für die elektronischen Bauteile notwendig.

Für die Verwendung in einem Gebiet ohne verfügbare Spannungsversorgung wäre der Betrieb über Akkumulatoren oder Batterien erforderlich.

Während der Validierung wurde festgestellt, dass durch die Materialbeschaffenheit der Greiffinger Objekte mit glatten Oberflächen nur bedingt aufgenommen werden konnten. Aus diesem Grund wäre eine Erhöhung der Rutschfestigkeit der Greiffinger notwendig.

Zusätzlich zur softwareseitigen Verriegelung der maximalen Positionsbereiche für die Verringerung der Quetschgefahr ist eine Hardwarelösung notwendig (zum Beispiel: Einbau von Positionsendschalter).

Die Verwendung eines Touchscreen Displays an der ASE würde zur Erhöhung der Flexibilität im reellen Einsatz führen. (Zum Beispiel durch visuelle Ausgaben von Kanalwechsel, Force-Feedback-Sensibilität, Stand der Energieversorgung usw.)

11 Summary

The project goal is the development of an electromechanical gripper with force feedback on the hand of the device operator under the supervision and support of the Wehrtechnischen Dienststelle 41. Two components are required for this. On the one hand the electromechanical gripper to be able to grip and pick up objects and on the other hand the control element that serves to control the electromechanical gripper and its feel. The two main components are to be implemented using suitable servo motors. Data transmission must be ensured to a minimum distance of 10m and have real-time capability. A wireless connection is also required. Data processing must be implemented using two suitable microcontrollers.

The basic principle of the force feedback function is the permanent comparison of loads and positions between the actuator sensor unit (German abbreviation: ASE) and the electromechanical gripper in real time.

A distinction is made between two operating modes. The idle and the load operation. Idling is a pure driving operation without force feedback. During load operation, the difference between the loads of the ASE and the gripper increases. The size of the difference (limit of differentiation) determines the occurrence of the respective operating state. The limits of distinction between idling and load operation are in a range between 88 and 1248.

Basically, the servomotors of both main components keep their current position. If the ASE is pressed together, a right-acting force acts in a value range from 1024 to 2047. If the ASE is pressed on, a left-acting force acts in a value range from 0 to 1023.

Idle:

If the gripper is idle and the ASE is pressed on, the result is that the current position of the ASE is increased by four increments. This is sent to the gripper. The gripper moves to the received position of the ASE. The gripper then sends its new load values and position values back to the ASE. The ASE moves to the received position of the gripper. This ensures continuous synchronization between the ASE and the gripper. If the gripper is idle and the ASE is pressed closed, the result is that the current position of the ASE is reduced by two increments and the previously described process of synchronization is carried out.

Load operation (ASE is closed):

If the received load of the gripper is less than the load of the ASE, the gripper is in load operation and the ASE is pressed down, the result is that the current position of the ASE is reduced by one increment. This position and the current load of the ASE are sent to the gripper. The gripper evaluates the position and the load and moves to the received position of the ASE. The gripper then sends its new load values and position values back to the ASE. The ASE moves to the received position of the gripper. This ensures continuous synchronization between the ASE and the gripper.

Load operation (gripper is pressed open):

If the received load of the gripper is greater than the load of the ASE and the gripper is pressed open, this has the consequence that the current position of the ASE is increased by one increment and the previously described process of synchronization is carried out.

Load operation (ASE is pressed on):

If the received load of the gripper is less than the load of the ASE, the gripper is in load operation and the ASE is pressed on, this means that the current position of the ASE is increased by one increment and the previously described process of synchronization is carried out .

12 Abbildungsverzeichnis

Abbildung 1: Greifer	11
Abbildung 2: Montagegestell	11
Abbildung 3: Greifer-Antrieb	12
Abbildung 4: Greiffinger	13
Abbildung 5: Aktor-Sensor-Einheit (ASE)	13
Abbildung 6: Trägerplatte	14
Abbildung 7: ASE-Antrieb	14
Abbildung 8: Daumenführung	15
Abbildung 9: Datenverarbeitungs- und Datenübertragungseinheit	15
Abbildung 10: Arduino Uno Entwicklungsboard mit aufgestecktem Dynamixel Arduino Shield	15
Abbildung 11: NRF24L01, 2.4GHz Funkmodul mit angeschlossenem AMS1117-3,3 DC-DC und AMS1117-5 DC-DC Spannungsregler	16
Abbildung 12: Anschluss des NRF24L01 Funkmoduls an dem Arduino Uno Mikrocontroller	18
Abbildung 13: Funktionsprinzip der Funkübertragung	22
Abbildung 14: Arrays zum Senden und Empfangen vom Mikrocontroller 1	23
Abbildung 15: Arrays zum Senden und Empfangen vom Mikrocontroller 2	23
Abbildung 16: Startroutine der Funkübertragung	23
Abbildung 17: Datenaustausch der Funkübertragung	24
Abbildung 18: Serielle Ausgabe der Funkübertragung	25
Abbildung 19: Funktionsprinzip der einfachen Ansteuerung des Greifers über die Funkverbindung ..	26
Abbildung 20: Grundeinstellungen des ASE-Servomotors	27
Abbildung 21: Auslesen und Übertragung der ASE-Position	27
Abbildung 22: Grundeinstellungen der Greifer-Servomotoren	28
Abbildung 23: Umrechnung der ASE-Position und Fahrbetrieb des Greifers	29
Abbildung 24: Testprogramm für die Fehlersuche der einfachen Ansteuerung des Greifers	30
Abbildung 25: Ausschnitt des Programms DYNAMIXEL Wizard 2.0	30
Abbildung 26: Funktionsprinzip der Force-Feedback-Funktion (1)	32
Abbildung 27: Funktionsprinzip der Force-Feedback-Funktion (2)	33
Abbildung 28: Force-Feedback-Funktion. Programmteil der ASE (1)	35
Abbildung 29: Force-Feedback-Funktion. Programmteil des Greifers (1)	36
Abbildung 30: Force-Feedback-Funktion. Programmteil der ASE (2)	37
Abbildung 31: Serielle Ausgabe der Force-Feedback-Funktion 1	37
Abbildung 32: Liniendiagramm von den Positions- und Lastwerten des Greifers	38
Abbildung 33: Gleitende Mittelwertbildung von den gewichteten Lastwerten der ASE	39
Abbildung 34: Gleitende Mittelwertbildung von den gewichteten Lastwerten des Greifers	40
Abbildung 35: Liniendiagramm von den Positions- und Lastwerten des Greifers und der ASE nach der Lastglättung	40
Abbildung 36: Force-Feedback_Funktion mit Leerlauf- und Lastbetrieb im ASE-Programm	43
Abbildung 37: Lastrichtungsfilter im Greifer-Programm	43
Abbildung 38: Resultat der Force-Feedback-Funktion nach dem Lastrichtungsfilter und der Unterscheidung zwischen Leerlauf und Lastbetrieb	44
Abbildung 39: Startroutine von der ASE	45
Abbildung 40: Startroutine von dem Greifer	46
Abbildung 41: Funktion der Positionsverriegelung im ASE-Programm	47
Abbildung 42: Funktion der Positionsverriegelung im Greifer-Programm	47

13 Tabellenverzeichnis

Tabelle 1: NRF24L01 Anschlüsse	17
Tabelle 2: Anschluss Funkmodul und Mikrocontroller	18
Tabelle 3: Datenanalyse	21

14 Quellenverzeichnis

- ¹ NRF24L01 Datenblatt Seite 1 (Auf dem Datenträger im Anhang)
- ² Bestimmungen Bundesnetzagentur 1 (Auf dem Datenträger im Anhang)
- ³ Bestimmungen Bundesnetzagentur 2 (Auf dem Datenträger im Anhang)
- ⁴ https://emf3.bundesnetzagentur.de/e_glossar.html 23.04.20 19:00 Uhr
- ⁵ NRF24L01 Datenblatt Seite 25 (Auf dem Datenträger im Anhang)
- ⁶ <http://tmrh20.github.io/RF24/classRF24.html#af2e409e62d49a23e372a70b904ae30e1> 23.04.20 19:00 Uhr
- ⁷ <https://github.com/nRF24/RF24> 23.04.20 19:00 Uhr
- ⁸ http://emanual.robotis.com/docs/en/parts/interface/dynamixel_shield/ 23.04.20 19:00 Uhr
- ⁹ <https://github.com/ROBOTIS-GIT/Dynamixel2Arduino> 23.04.20 19:00 Uhr
- ¹⁰ ARDUINO_Befehlsübersicht Seite 166 (Auf dem Datenträger im Anhang)

15 Anhang

15.1 Bestellliste

Bestellliste Projekt

Shop	Artikel	Anzahl	Preis	Gesamtpreis
Zerodna	F3 Montagerahmen für den Servo	1	18,00 €	18,00 €
	F4 Montagerahmen lang, zur Montage an die Servo-Achse(Schwinger)	1	13,00 €	13,00 €
MyBotShop	ROBOTIS SMPS 12V 5A	1	35,00 €	35,00 €
	U2D2 Power Hub	2	30,00 €	60,00 €
	ROBOTIS Schrauben & Muttern Set BNS-10	1	29,00 €	29,00 €
	Dynamixel AX-12A	3	46,00 €	138,00 €
	DYNAMIXEL Arduino Shield	2	28,95 €	57,90 €
Conrad	Greiffinger Festo DHAS 120 mm	2	53,00 €	106,00 €
	DHAS-MA-B6-120 Befestigungswinkel	2	39,00 €	78,00 €
	DHAS-ME-H9-120 Befestigungsbausatz	2	21,00 €	42,00 €
az-delivery	NRF24L01 2,4GHz Funkmodul	2	1,49 €	2,98 €
ebay	Arduino Uno Mikrocontrollerboard	2	4,99 €	9,98 €
	2Pcs Socket Adapter plate Board For 8Pin NRF24L01+ Wireless Transceiver Module	2	1,49 €	2,98 €
	2Pcs DC-Verteiler 5,5/2,1mm 1x Kupplung / 2x Stecker Y-Kabel Netzteile für LEDs	2	3,19 €	6,38 €
	DC Steckverbinder Hohlbuchse 5,5x2,1mm Adapter Netzteil Kupplung Videoüberwachun	2	0,99 €	1,98 €
	Spannungsregler AMS1117 5V 800mA LDO Step Down Power Modul für Arduino, Pi & Co.	2	1,91 €	3,82 €
				605,02 €

15.2 Programme

15.2.1 Funkübertragung ASE

```
#include <SPI.h> // Einbindung der SPI Bibliothek (Serial Peripheral
Interface)
#include <RF24.h> // Einbindung der RF24 Bibliothek (WLAN Kommunikation)

RF24 ASE_Funk (9, 10); // Instanziierung des ASE_Funk Objekts
byte Addresses[][6] = {"0"}; // Deklaration und Initialisierung der Pipe für die
Funkübertragung

int Daten_Senden[2]={1111,2222}; // Deklaration und Initialisierung des Sendedatenspeichers
int Daten_Empfangen[2]; // Deklaration des Empfangsdatspeichers

//-----

void setup() { // Void setup
    Serial.begin(9600); // Kommunikation des seriellen Monitors mit
                        // eingestellter Baudrate
    ASE_Funk.begin(); // Aktivierung des NRF24L01 Funkmoduls
    ASE_Funk.setChannel(10); // Einstellung der Send-/Empfangsfrequenz
    ASE_Funk.setPALevel(RF24_PA_MAX); // Einstellung der Sendeleistung
    ASE_Funk.setDataRate( RF24_2MBPS ); // Einstellung der Datenübertragungsrate
    ASE_Funk.setAutoAck(1); // Aktivierung der Acknowledge-Funktion
    ASE_Funk.enableAckPayload(); // Aktivierung der Acknowledge-Funktion der
                                // Paketgröße
    ASE_Funk.enableDynamicPayloads(); // Aktivierung der dynamischen Paketgröße
    ASE_Funk.openReadingPipe(1, Addresses[0]); // Öffnen einer Pipe zum Empfangen
    ASE_Funk.startListening(); // Startet den Empfang
}

//-----

void loop() { // Void loop
    if ( ASE_Funk.available() ) { // Prüft, ob Daten empfangen
                                // werden
        while (ASE_Funk.available() ) { // Solange Daten empfangen
                                // werden
            ASE_Funk.read( &Daten_Empfangen, sizeof(Daten_Empfangen) ); // Datengröße wird ermittelt
                                // und im vorgesehenem
                                // Speicher abgelegt
        }
        Serial.println("-----"); // Serielle Ausgabe des
                                // Textes
        Serial.print("ASE empfängt1: "); // Serielle Ausgabe des
                                // Textes
        Serial.println(Daten_Empfangen[0]); // Serielle Ausgabe des
                                // Empfangsspeichers
        Serial.print("ASE empfängt2: "); // Serielle Ausgabe des
                                // Textes
        Serial.println(Daten_Empfangen[1]); // Serielle Ausgabe des
                                // Empfangsspeichers
        Serial.println("-----"); // Serielle Ausgabe des
                                // Textes
    }

    delay(100); // Wartezeit für den multitasking
                // Betrieb der Funkübertragung

    ASE_Funk.stopListening(); // Stoppt den Empfang
    ASE_Funk.openWritingPipe(Addresses[0]); // Öffnen einer Pipe zum Senden
    ASE_Funk.write(&Daten_Senden, sizeof(Daten_Senden)); // Datengröße wird ermittelt und aus
                                                        // vorgesehenem Speicher ausgelesen
                                                        // und gesendet

    Serial.print("ASE sendet1: "); // Serielle Ausgabe des Textes
    Serial.println(Daten_Senden[0]); // Serielle Ausgabe des Sendespeichers
    Serial.print("ASE sendet2: "); // Serielle Ausgabe des Textes
    Serial.println(Daten_Senden[1]); // Serielle Ausgabe des Sendespeichers

    ASE_Funk.openReadingPipe(1, Addresses[0]); // Öffnen einer Pipe zum Empfangen
    ASE_Funk.startListening(); // Startet den Empfang
}
```

15.2.2 Funkübertragung Greifer

```
#include <SPI.h> // Einbindung der SPI Bibliothek (Serial Peripheral
                // Interface)
#include <RF24.h> // Einbindung der RF24 Bibliothek (WLAN Kommunikation)

RF24 Greifer_Funk (9, 10); // Instanziierung des Greifer_Funk Objekts
byte Addresses[][6] = {"0"}; // Deklaration und Initialisierung der Pipe für die
                             // Funkübertragung

int Daten_Senden[2]={3333,4444}; // Deklaration und Initialisierung des Sendedatenspeichers
int Daten_Empfangen[2]; // Deklaration des Empfangsdatenspeichers

//-----

void setup() { // Void setup
    Serial.begin(9600); // Kommunikation des seriellen Monitors mit
                        // eingestellter Baudrate
    Greifer_Funk.begin(); // Aktivierung des NRF24L01 Funkmoduls
    Greifer_Funk.setChannel(10); // Einstellung der Sende-/Empfangsfrequenz
    Greifer_Funk.setPALevel(RF24_PA_MAX); // Einstellung der Sendeleistung
    Greifer_Funk.setDataRate( RF24_2MBPS ); // Einstellung der Datenübertragungsrate
    Greifer_Funk.setAutoAck(1); // Aktivierung der Acknowledge-Funktion
    Greifer_Funk.enableAckPayload(); // Aktivierung der Acknowledge-Funktion der
                                    // Paketgröße
    Greifer_Funk.enableDynamicPayloads(); // Aktivierung der dynamischen Paketgröße
    Greifer_Funk.openReadingPipe(1, Addresses[0]); // Öffnen einer Pipe zum Empfangen
    Greifer_Funk.startListening(); // Startet den Empfang
}

//-----

void loop() { // Void loop
    if ( Greifer_Funk.available() ) { // Prüft, ob Daten
                                     // empfangen werden
        while (Greifer_Funk.available()){ // Solange Daten
                                           // empfangen werden
            Greifer_Funk.read( &Daten_Empfangen, sizeof(Daten_Empfangen) ); // Datengröße wird
                                                                              // ermittelt und im
                                                                              // vorgesehenem Speicher
                                                                              // abgelegt
        }
        Serial.println("-----"); // Serielle Ausgabe des
                                     // Textes
        Serial.print("Greifer empfängt1: "); // Serielle Ausgabe des
                                              // Textes
        Serial.println(Daten_Empfangen[0]); // Serielle Ausgabe des
                                             // Empfangspeichers
        Serial.print("Greifer empfängt2: "); // Serielle Ausgabe des
                                              // Textes
        Serial.println(Daten_Empfangen[1]); // Serielle Ausgabe des
                                             // Empfangspeichers
        Serial.println("-----"); // Serielle Ausgabe des
                                     // Textes
    }

    delay(100); // Wartezeit für den multitasking
                // Betrieb der Funkübertragung

    Greifer_Funk.stopListening(); // Stoppt den Empfang
    Greifer_Funk.openWritingPipe(Addresses[0]); // Öffnen einer Pipe zum Senden
    Greifer_Funk.write(&Daten_Senden, sizeof(Daten_Senden)); // Datengröße wird ermittelt und
                                                             // aus vorgesehenem Speicher
                                                             // ausgelesen und gesendet

    Serial.print("Greifer sendet1: "); // Serielle Ausgabe des Textes
    Serial.println(Daten_Senden[0]); // Serielle Ausgabe des
                                     // Sendespeichers
    Serial.print("Greifer sendet2: "); // Serielle Ausgabe des Textes
    Serial.println(Daten_Senden[1]); // Serielle Ausgabe des
                                     // Sendespeichers

    Greifer_Funk.openReadingPipe(1, Addresses[0]); // Öffnen einer Pipe zum Empfangen
    Greifer_Funk.startListening(); // Startet den Empfang
}
```

15.2.3 Einfache Servoansteuerung ASE

```
#include <SPI.h> // Einbindung der SPI Bibliothek (Serial Peripheral Interface)
#include <RF24.h> // Einbindung der RF24 Bibliothek (WLAN Kommunikation)
#include <Dynamixel2Arduino.h> // Einbindung der Dnamixel2Arduino Bibliothek (Kommunikation
                                Arduino und Dynamixel Servos)

//===== Global Servos =====

#if defined(ARDUINO_AVR_UNO) || defined(ARDUINO_AVR_MEGA2560) // Selektion ob Arduino Uno
                                                                // oder Arduino Mega definiert
                                                                // wurde
    #define ASE_SERIAL Serial // Definition der
                                // Kommunikationsart Serial
    const uint8_t ASE_DIR_PIN = 2; // Deklaration und
                                    // Initialisierung der
                                    // Dynamixel Shield Direction
                                    // Pin Variable
#else // Bei nicht zutreffender
    #define ASE_SERIAL Serial1 // Definition der
                                // Kommunikationsart Serial1
    const uint8_t ASE_DIR_PIN = 2; // Deklaration und
                                    // Initialisierung der
                                    // Dynamixel Shield Direction
                                    // Pin Variable
#endif // Ende der Selektion

const uint8_t ASE_ID = 3; // Deklaration und
                           // Initialisierung der ASE ID
                           // Variable
const float ASE_PROTOCOL_VERSION = 1.0; // Deklaration und
                                          // Initialisierung der
                                          // Protokollversions Variable

Dynamixel2Arduino ASE(ASE_SERIAL, ASE_DIR_PIN); // Instanziierung des ASE
                                                  // Objekts

//===== Global WLAN Kommunikation =====

RF24 ASE_Funk(9, 10); // Instanziierung
                       // des ASE-Funk
                       // Objekts

byte Address[][6] = {"1Node", "2Node", "3Node", "4Node", "5Node", "6Node"}; // Deklaration
und Initialisierung der Pipe für die
Funkübertragung

//=====

void setup() // Void setup
{
//===== Setup Servos =====

    ASE.begin(115200); // Beginn der Kommunikation zwischen ASE
                       // und Arduino Uno mit eingestellter
                       // Baudrate
    ASE.setPortProtocolVersion(ASE_PROTOCOL_VERSION); // Setzen der Protokoll Version
    ASE.torqueOff(ASE_ID); // Deaktivierung des Servos um ihn
                           // bewegen zu können
    ASE.setOperatingMode(ASE_ID, OP_POSITION); // Setzen des Operationsmodus
                                                // (Fahrbetrieb über Position)

//===== Setup WLAN Kommunikation =====

    ASE_Funk.begin(); // Aktivierung des NRF24L01 Funkmoduls
    ASE_Funk.setAutoAck(1); // Aktivierung der Acknowledge_Funktion
    ASE_Funk.setRetries(0, 15); // Die Zeit zwischen den Versuchen den Empfänger zu
                                // erreichen, Anzahl der Versuche
    ASE_Funk.enableAckPayload(); // Erlaubt das Senden der Daten auf die Anfrage des
                                // Empfängers
    ASE_Funk.setPayloadSize(32); // Packetgröße in Byte
    ASE_Funk.openWritingPipe(Address[0]); // Öffnen einer Pipe zum Senden
    ASE_Funk.setChannel(45); // Einstellung der Sende-/Empfangsfrequenz
    ASE_Funk.setPALevel(RF24_PA_MAX); // Einstellung der Sendeleistung
    ASE_Funk.setDataRate(RF24_2MBPS); // Einstellung der Datenübertragungsrate
    ASE_Funk.powerUp(); // Arbeitsbeginn
    ASE_Funk.stopListening(); // Stoppt den Empfang
```

```

}

//=====

void loop() {                                     // Void loop

    int Daten[2]={ASE_ID,ASE.getPresentPosition(3)}; // Speichern der ASE ID und der aktuellen
                                                    ASE Position im Sendedatenspeicher

    ASE_Funk.write(&Daten, sizeof(Daten));          // Datengröße wird ermittelt und aus
                                                    vorgesehenem Speicher ausgelesen und
                                                    gesendet

}

```

15.2.4 Einfache Servoansteuerung Greifer

```

#include <SPI.h>                                // Einbindung der SPI Bibliothek (Serial Peripheral Interface)
#include <RF24.h>                                // Einbindung der RF24 Bibliothek (WLAN Kommunikation)
#include <Dynamixel2Arduino.h>                  // Einbindung der Dnamixel2Arduino Bibliothek (Kommunikation
                                                    Arduino und Dynamixel Servos)

//===== Global Servos =====

#if defined(ARDUINO_AVR_UNO) || defined(ARDUINO_AVR_MEGA2560) // Selektion ob Arduino Uno
                                                                oder Arduino Mega definiert
                                                                wurde

    #define Greifer_SERIAL    Serial            // Definition der
                                                    Kommunikationsart Serial
    const uint8_t Greifer_DIR_PIN = 2;          // Deklaration und
                                                    Initialisierung der
                                                    Dynamixel Shield Direction
                                                    Pin Variable
#else                                           // Bei nicht zutreffender
                                                    Selektion

    #define Greifer_SERIAL    Serial1           // Definition der
                                                    Kommunikationsart Serial1
    const uint8_t Greifer_DIR_PIN = 2;          // Deklaration und
                                                    Initialisierung der
                                                    Dynamixel Shield Direction
                                                    Pin Variable
#endif                                         // Ende der Selektion

const uint8_t Gripper_ID1 = 1;                // Deklaration und
                                                    Initialisierung der
                                                    Gripper1 ID Variable
const uint8_t Gripper_ID2 = 2;                // Deklaration und
                                                    Initialisierung der
                                                    Gripper2 ID Variable
const uint32_t Timeout=100;                  // Deklaration und
                                                    Initialisierung der timeout
                                                    Variable für die
                                                    Kommunikation zwischen
                                                    Dynamixel Shield und
                                                    Dynamixel Servo in ms

const float Greifer_PROTOCOL_VERSION = 1.0;   // Deklaration und
                                                    Initialisierung der
                                                    Protokollversions Variable

Dynamixel2Arduino Gripper1(Greifer_SERIAL, Greifer_DIR_PIN); // Instanziierung des Gripper1
                                                                Objekts
Dynamixel2Arduino Gripper2(Greifer_SERIAL, Greifer_DIR_PIN); // Instanziierung des Gripper2
                                                                Objekts

//===== Global WLAN Kommunikation =====

RF24 Greifer_Funk(9,10);                     // Instanziierung des
                                                    Greifer-Funk
                                                    Objekts
byte Address[][6] = {"1Node","2Node","3Node","4Node","5Node","6Node"}; // Bezeichnen der
                                                                Pipes
int Position;                                 // Deklaration der
                                                    Variable für die
                                                    umgerechnete
                                                    Greifer Position

```



```

//=====

void setup() {                                     // Void setup

//===== Setup Servos =====

    Gripper1.begin(115200);                        // Beginn der Kommunikation
                                                    // zwischen Gripper1 und Arduino
                                                    // Uno mit eingestellter
                                                    // Baudrate
    Gripper2.begin(115200);                        // Beginn der Kommunikation
                                                    // zwischen Gripper2 und Arduino
                                                    // Uno mit eingestellter
                                                    // Baudrate
    Gripper1.setPortProtocolVersion(Greifer_PROTOCOL_VERSION); // Setzen der Protokoll Version
    Gripper2.setPortProtocolVersion(Greifer_PROTOCOL_VERSION); // Setzen der Protokoll Version
    Gripper1.torqueOff(Gripper_ID1);               // Deaktivierung des Servos vom
                                                    // Gripper1
    Gripper2.torqueOff(Gripper_ID2);               // Deaktivierung des Servos vom
                                                    // Gripper2
    Gripper1.setOperatingMode(Gripper_ID1, OP_POSITION); // Setzen des Operationsmodus
                                                    // (Fahrbetrieb über Position)
    Gripper2.setOperatingMode(Gripper_ID2, OP_POSITION); // Setzen des Operationsmodus
                                                    // (Fahrbetrieb über Position)
    Gripper1.torqueOn(Gripper_ID1);                // Aktivierung des Servos vom
                                                    // Gripper1
    Gripper2.torqueOn(Gripper_ID2);                // Aktivierung des Servos vom
                                                    // Gripper2

//===== Setup WLAN Kommunikation =====

    Greifer_Funk.begin();                          // Aktivierung des NRF24L01 Funkmoduls
    Greifer_Funk.setAutoAck(1);                    // Aktivierung der Acknowledge_Funktion
    Greifer_Funk.setRetries(0,15);                 // Die Zeit zwischen den Versuchen den Empfänger
                                                    // zu erreichen, Anzahl der Versuche
    Greifer_Funk.enableAckPayload();                // Erlaubt das Senden der Daten auf die Anfrage
                                                    // des Empfängers
    Greifer_Funk.setPayloadSize(32);                // Packetgröße in Byte
    Greifer_Funk.openReadingPipe(1,Address[0]);     // Öffnen einer Pipe zum Empfangen
    Greifer_Funk.setChannel(45);                   // Einstellung der Send-/Empfangsfrequenz
    Greifer_Funk.setPALevel (RF24_PA_MAX);          // Einstellung der Sendeleistung
    Greifer_Funk.setDataRate (RF24_2MBPS);          // Einstellung der Datenübertragungsrate
    Greifer_Funk.powerUp();                         // Arbeitsbeginn
    Greifer_Funk.startListening();                 // Startet den Empfang
}

void loop() {                                     // Void loop
    byte PipeNo;                                  // Deklaration der Variable für die Pipe
                                                    // Nummer
    int Daten[2];                                  // Deklaration des Empfangsdatenspeichers
    while( Greifer_Funk.available(&PipeNo)){       // Solange Daten empfangen werden
        Greifer_Funk.read( &Daten, sizeof(Daten) ); // Datengröße wird ermittelt und im
                                                    // vorgesehenem Speicher abgelegt

        int Empfangen1=Daten[0];                   // Speichern der empfangenen ASE ID
        int Empfangen2=Daten[1];                   // Speichern der empfangenen ASE Position

        Position= 512-(Empfangen2-512);            // Rechnet die empfangene ASE Position um
                                                    // für die Greifer Position

        Gripper1.setGoalPosition(Gripper_ID1, Position ); // Gripper1 fährt auf die umgerechnete
                                                    // Position
        Gripper2.setGoalPosition(Gripper_ID2, Position ); // Gripper2 fährt auf die umgerechnete
                                                    // Position
    }
}

```

15.2.5 Einfache Servoansteuerung Zusatzprogramm

```
#include <Dynamixel2Arduino.h> // Einbindung der
                                // Dnamixel2Arduino Bibliothek
                                // (Kommunikation Arduino und
                                // Dynamixel Servos)

#if defined(ARDUINO_AVR_UNO) || defined(ARDUINO_AVR_MEGA2560) // Selektion ob Arduino Uno oder
                                                                // Arduino Mega definiert wurde
    #define Greifer_SERIAL    Serial // Definition der
                                      // Kommunikationsart Serial
    const uint8_t Greifer_DIR_PIN = 2; // Deklaration und
                                      // Initialisierung der Dynamixel
                                      // Shield Direction Pin Variable
#else // Bei nicht zutreffender
    #define Greifer_SERIAL    Serial1 // Definition der
                                       // Kommunikationsart Serial1
    const uint8_t Greifer_DIR_PIN = 2; // Deklaration und
                                       // Initialisierung der Dynamixel
                                       // Shield Direction Pin Variable
#endif // Ende der Selektion

const uint8_t Greifer_ID = 1; // Deklaration und
                               // Initialisierung der Greifer
                               // ID Variable
const float Greifer_PROTOCOL_VERSION = 1.0; // Deklaration und
                                              // Initialisierung der
                                              // Protokollversions Variable

Dynamixel2Arduino Greifer(Greifer_SERIAL, Greifer_DIR_PIN); // Instanziierung des Greifer
                                                            // Objekts

void setup() { // Void setup

    Greifer.begin(115200); // Beginn der Kommunikation
                           // zwischen Greifer und Arduino
                           // Uno mit eingestellter
                           // Baudrate

    Greifer.setPortProtocolVersion(Greifer_PROTOCOL_VERSION); // Setzen der Protokoll Version
    Greifer.torqueOff(Greifer_ID); // Deaktivierung der Servos
    Greifer.setOperatingMode(Greifer_ID, OP_POSITION); // Setzen des Operationsmodus
                                                         // (Fahrbetrieb über Position)
    Greifer.torqueOn(Greifer_ID); // Aktivierung der Servos
}

void loop() { // Void loop

    Greifer.setGoalPosition(Greifer_ID, 512 ); // Greifer fährt auf die
                                                // Position
    delay(1000); // Wartezeit

    Greifer.setGoalPosition(Greifer_ID, 105, UNIT_DEGREE); // Greifer fährt auf die
                                                            // Position
    delay(1000); // Wartezeit
}
```

15.2.6 Force-Feedback-Funktion ASE

```
#include <SPI.h> // Einbindung der SPI Bibliothek (Serial Peripheral
                  // Interface)
#include <RF24.h> // Einbindung der RF24 Bibliothek (WLAN Kommunikation)
#include <Dynamixel2Arduino.h> // Einbindung der Dnamixel2Arduino Bibliothek
                              // (Kommunikation Arduino und Dynamixel Servos)

//===== Global Servos =====

#if defined(ARDUINO_AVR_UNO) || defined(ARDUINO_AVR_MEGA2560) // Selektion ob Arduino Uno
                                                              // oder Arduino Mega
                                                              // definiert wurde
    #define ASE_SERIAL    Serial // Definition der
                                  // Kommunikationsart Serial
    const uint8_t ASE_DIR_PIN = 2; // Deklaration und
                                    // Initialisierung der
                                    // Dynamixel Shield
                                    // Direction Pin Variable
#else // Bei nicht zutreffender
      // Selektion
    #define ASE_SERIAL    Serial1 // Definition der
                                   // Kommunikationsart Serial1
    const uint8_t ASE_DIR_PIN = 2; // Deklaration und
                                    // Initialisierung der
                                    // Dynamixel Shield
                                    // Direction Pin Variable
#endif // Ende der Selektion

const uint8_t ASE_ID = 3; // Deklaration und
                           // Initialisierung der ASE
                           // ID Variable
const uint32_t Timeout=100; // Deklaration und
                             // Initialisierung der
                             // timeout Variable für die
                             // Kommunikation zwischen
                             // Dynamixel Shield und
                             // Dynamixel Servo in ms
const float ASE_PROTOCOL_VERSION = 1.0; // Deklaration und
                                           // Initialisierung der
                                           // Protokollversions
                                           // Variable
Dynamixel2Arduino ASE(ASE_SERIAL, ASE_DIR_PIN); // Instanziierung des ASE
                                                  // Objekts

//===== Global WLAN Kommunikation =====

RF24 ASE_Funk (9, 10); // Instanziierung des ASE-Funk Objekts
byte Addresses[][6] = {"0"}; // Deklaration und Initialisierung der Pipe für die
                               // Funkübertragung
int Daten_Empfangen[2]; // Deklaration des Empfangsdatenspeichers
int Daten_Senden[2]; // Deklaration des Sendedatenspeichers

//===== Global Datenverarbeitung =====

int ASE_Last_Smooth = 0; // Deklaration und Initialisierung der Variable für den
                          // gleitenden Mittelwert der ASE Last
int ASE_Last_Aktuell = 0; // Deklaration und Initialisierung der Variable für die
                           // aktuelle ASE Last
int ASE_Position_Aktuell; // Deklaration der Variable für die aktuelle ASE Position
int ASE_Position_Smooth; // Deklaration der Variable für den gleitenden Mittelwert
                           // der ASE Position
int ASE_Position_Senden; // Deklaration der Variable für die zu versendende ASE
                           // Position
int Greifer_Last; // Deklaration der Variable für die Greifer Last
int Greifer_Position; // Deklaration der Variable für die Greifer Position
byte Zustand = 0;

void setup() { // Void setup

//===== Setup Servos =====

    ASE.begin(115200); // Beginn der Kommunikation
                       // zwischen ASE und Arduino Uno
                       // mit eingestellter Baudrate
    ASE.setPortProtocolVersion(ASE_PROTOCOL_VERSION); // Setzen der Protokoll Version
```

```

ASE.setOperatingMode(ASE_ID, OP_POSITION);           // Setzen des Operationsmodus
                                                       // (Fahrbetrieb über Position)
ASE.torqueOn(ASE_ID);                               // Aktivierung des Servos

if(ASE.ping(ASE_ID) == false)                       // Prüft, ob die Kommunikation
                                                       // zwischen dem Servomotor und dem
                                                       // Arduino Shield unterbrochen ist

{
    software_Reset();                               // Aktivierung der Reset Funktion
}

else                                                  // Ausführung bei bestehender
                                                       // Kommunikation zwischen dem
                                                       // Servomotor und dem Arduino
                                                       // Shield

{
    ASE.writeControlItem(PUNCH,3,32,Timeout);        // Herabsetzen der Punch_Funktion
                                                       // für einen weicheren Betrieb
    ASE.writeControlItem(TORQUE_LIMIT,3,204,Timeout); // Herabsetzen des ASE Drehmoments
                                                       // zur Erhöhung der Sicherheit bei
                                                       // der Startroutine (20%)
    ASE.writeControlItem(MOVING_SPEED,3,204,Timeout); // Herabsetzen der ASE
                                                       // Geschwindigkeit zur Erhöhung
                                                       // der Sicherheit bei der
                                                       // Startroutine (20%)

    delay(2000);                                     // Wartezeit zur Verarbeitung der
                                                       // oben aufgeführten Einstellungen
    ASE.setGoalPosition(3,820);                      // Fahren der ASE auf 90°
    delay(2000);                                     // Wartezeit für den Fahrbetrieb
                                                       // auf 90°
    ASE.setGoalPosition(3,500);                      // Fahren der ASE auf 0°
    delay(2000);                                     // Wartezeit für den Fahrbetrieb
                                                       // auf 0°
    ASE.setGoalPosition(3,660);                      // Fahren der ASE auf 45°
    delay(2000);                                     // Wartezeit für den Fahrbetrieb
                                                       // auf 45°
    ASE.writeControlItem(TORQUE_LIMIT,3,1023,Timeout); // Heraufsetzen des ASE
                                                       // Drehmoments für den Nennbetrieb
    ASE.writeControlItem(MOVING_SPEED,3,1023,Timeout); // Heraufsetzen der ASE
                                                       // Geschwindigkeit für den
                                                       // Nennbetrieb
    ASE_Position_Smooth = ASE.getPresentPosition(3); // Speichern der aktuellen ASE
                                                       // Position in die Variable für
                                                       // den gleitenden Mittelwert

//===== Setup WLAN Kommunikation =====

ASE_Funk.begin();                                   // Aktivierung des NRF24L01 Funkmoduls
ASE_Funk.setChannel(45);                           // Einstellung der Sende-/Empfangsfrequenz
ASE_Funk.setPALevel(RF24_PA_MAX);                  // Einstellung der Sendeleistung
ASE_Funk.setDataRate( RF24_2MBPS );                // Einstellung der Datenübertragungsrate
ASE_Funk.openReadingPipe(1, Addresses[0]);          // Öffnen einer Pipe zum Empfangen
ASE_Funk.startListening();                          // Startet den Empfang
ASE_Funk.setAutoAck(1);                             // Aktivierung der Acknowledge_Funktion
ASE_Funk.enableDynamicPayloads();                  // Aktivierung der dynamischen Paketgröße
ASE_Funk.setCRCLength(RF24_CRC_16);                 // Setzen der CRC-Prüflänge auf 16 Bit
}

}

void loop() {                                       // Void loop

if(!ASE_Funk.available() && Zustand < 1)//Prüft, ob keine Daten empfangen werden und Zustand
    <1 ist.
{
    delay(1000);                                   //Wartezeit
    Zustand++;                                     //Zustandswert wird um 1 erhöht
    ASE_Position_Senden = 660;                     //Setzt Position auf den Wert
}

if (ASE_Funk.available())                          // Prüft, ob Daten empfangen
                                                       // werden
{

```

```

while (ASE_Funk.available()) // Solange Daten empfangen
                                werden
{
    ASE_Funk.read( &Daten_Empfangen, sizeof(Daten_Empfangen) ); // Datengröße wird ermittelt
                                                                    und im vorgesehenem
                                                                    Speicher abgelegt
}

ASE_Last_Aktuell = ASE.readControlTableItem(PRESENT_LOAD,3,Timeout); //Auslesen und
                                                                    speichern der
                                                                    aktuellen ASE Last

ASE_Last_Smooth = 0.8 * ASE_Last_Smooth + 0.2 * ASE_Last_Aktuell; //Bilden des
                                                                    gleitender
                                                                    Mittelwerts der
                                                                    gewichteten
                                                                    Lastwerte der ASE

Greifer_Last = Daten_Empfangen[1]; //Speichern der
                                                                    empfangenen
                                                                    Greifer Last

Greifer_Position = Daten_Empfangen[0]; //Speichern der
                                                                    empfangenen
                                                                    Greifer Position

//===== Verriegelung der ASE Position im FFB =====

if (Greifer_Position > 820) // Prüft, ob der maximale Grenzwert der Greifer Position
                            überschritten wurde
{
    Greifer_Position = 820; // Setzt die Greifer Position auf den maximalen
                            Grenzwert
}

if (Greifer_Position < 500) // Prüft, ob der minimale Grenzwert der Greifer Position
                            unterschritten wurde
{
    Greifer_Position = 500; // Setzt die Greifer Position auf den minimalen
                            Grenzwert
}

//===== Ende der Verriegelung =====

ASE.setGoalPosition(3,Greifer_Position); // ASE fährt auf die empfangene Greifer
Position
ASE_Position_Aktuell = ASE.getPresentPosition(3); // Speichern der aktuellen ASE Position
ASE_Position_Smooth = 0.5 * ASE_Position_Smooth + 0.5 * ASE_Position_Aktuell; //Bilden
des gleitender Mittelwerts der gewichteten ASE Position

//===== Leerlauf ASE Auf =====

if( ASE_Last_Smooth <= 1023 && Greifer_Last <= 88 ) // Prüft, ob die ASE im
                                                    richtigen Lastbereich
                                                    ist und ob der Greifer
                                                    im Leerlauf
                                                    ist
{
    ASE.writeControlTableItem(CW_COMPLIANCE_MARGIN,3,8,Timeout); // Verringerung der
                                                                    Positionsgenauigkeit
                                                                    der ASE (Zu)
    ASE.writeControlTableItem(CCW_COMPLIANCE_MARGIN,3,8,Timeout); // Verringerung der
                                                                    Positionsgenauigkeit
                                                                    der ASE (Auf)
    ASE_Position_Smooth = ASE_Position_Smooth + 2; // Berechnung der neuen
                                                                    ASE Position
}

//===== Verriegelung der ASE Position im Leerlauf ASE Auf =====

if (ASE_Position_Smooth > 820) // Prüft, ob der maximale
                                Grenzwert der ASE
                                Position überschritten
                                wurde
{
    ASE_Position_Smooth = 820; // Setzt die ASE Position
                                auf den maximalen
                                Grenzwert
}

//===== Ende der Verriegelung =====

```

```

    ASE.setGoalPosition(3, ASE_Position_Smooth);           // Fahren auf die neu
                                                            // berechnete ASE Position
    ASE_Position_Senden = ASE_Position_Smooth;             // Speichern der aktuellen
                                                            // ASE Position
}

//===== Leerlauf ASE Zu =====

if( ASE_Last_Smooth >= 1024 && Greifer_Last <= 1248)       // Prüft, ob die ASE
                                                            // im richtigen
                                                            // Lastbereich ist und
                                                            // ob der Greifer im
                                                            // Leerlauf ist
{
    ASE.writeControlTableItem(CW_COMPLIANCE_MARGIN,3,8,Timeout); // Verringerung der
                                                            // Positionsgenauigkeit
                                                            // der ASE (Zu)
    ASE.writeControlTableItem(CCW_COMPLIANCE_MARGIN,3,8,Timeout); // Verringerung der
                                                            // Positionsgenauigkeit
                                                            // der ASE (Auf)
    ASE_Position_Smooth = ASE_Position_Smooth - 1;           // Berechnung der
                                                            // neuen ASE Position
}

//===== Verriegelung der ASE Position im Leerlauf ASE Zu =====

if (ASE_Position_Smooth < 500)                             // Prüft, ob der maximale Grenzwert
                                                            // der ASE Position unterschritten
                                                            // wurde
{
    ASE_Position_Smooth = 500;                               // Setzt die ASE Position auf den
                                                            // minimalen Grenzwert
}

//===== Ende der Verriegelung =====

    ASE.setGoalPosition(3,ASE_Position_Smooth);           // Fahren auf die neu berechnete ASE
                                                            // Position
    ASE_Position_Senden = ASE_Position_Smooth;             // Speichern der aktuellen ASE
                                                            // Position
}

//===== ASE wird zugedrückt =====

if (ASE_Last_Smooth > Greifer_Last && ASE_Last_Smooth >= 1024 && Greifer_Last >=
1249) // Prüft, ob die ASE Last größer ist als die Greifer Last und ob die ASE Last im
    richtigen Lastbereich ist und ob die Greifer Last im Force-Feedback Bereich ist
{
    ASE.writeControlTableItem(CW_COMPLIANCE_MARGIN,3,1,Timeout); // Erhöhung der
                                                            // Positionsgenauigkeit
                                                            // der ASE (Zu)
    ASE.writeControlTableItem(CCW_COMPLIANCE_MARGIN,3,1,Timeout); // Erhöhung der
                                                            // Positionsgenauigkeit
                                                            // der ASE (Auf)
    ASE_Position_Aktuell = ASE_Position_Aktuell - 1;         // Berechnung der neuen
                                                            // ASE Position
    ASE_Position_Senden = ASE_Position_Aktuell;             // Speichern der aktuellen
                                                            // ASE Position
}

//===== Greifer wird aufgedrückt =====

if (ASE_Last_Smooth < Greifer_Last && ASE_Last_Smooth <= 1023) // Prüft, ob die ASE
                                                            // Last kleiner ist als
                                                            // die Greifer Last und
                                                            // ob die ASE im
                                                            // richtigen Lastbereich
                                                            // ist
{
    ASE.writeControlTableItem(CW_COMPLIANCE_MARGIN,3,1,Timeout); // Erhöhung der
                                                            // Positionsgenauigkeit
                                                            // der ASE (Zu)
    ASE.writeControlTableItem(CCW_COMPLIANCE_MARGIN,3,2,Timeout); // Erhöhung der
                                                            // Positionsgenauigkeit
                                                            // der ASE (Auf)
    ASE_Position_Aktuell = ASE_Position_Aktuell + 1;         // Berechnung der neuen
                                                            // ASE Position
}

```

```

    ASE_Position_Senden = ASE_Position_Aktuell;                // Speichern der
                                                                aktuellen ASE
                                                                Position
}

//===== ASE wird aufgedrückt =====

88) if (ASE_Last_Smooth > Greifer_Last && ASE_Last_Smooth <= 1023 && Greifer_Last >=
    //89 Prüft, ob die ASE Last größer ist als die Greifer Last und ob die ASE Last im
    richtigen Lastbereich ist und ob die Greifer Last im Force-Feedback Bereich ist
    {
        ASE.writeControlItem(CW_COMPLIANCE_MARGIN,3,1,Timeout); // Erhöhung der
                                                                Positionsgenauigkeit
                                                                der ASE (Zu)
        ASE.writeControlItem(CCW_COMPLIANCE_MARGIN,3,1,Timeout); // Erhöhung der
                                                                Positionsgenauigkeit
                                                                der ASE (Auf)
        ASE_Position_Aktuell = ASE_Position_Aktuell + 1;        // Berechnung der neuen
                                                                ASE Position
        ASE_Position_Senden = ASE_Position_Aktuell;            // Speichern der aktuellen
                                                                ASE Position
    }
}

delay(10);                                                    // Wartezeit für den multitasking
                                                                Betrieb der Funkübertragung

Daten_Senden[0] = ASE_Position_Senden;                        // ASE Position in den Sendespeicher
                                                                legen
Daten_Senden[1] = ASE_Last_Smooth;                            // ASE Last in den Sendespeicher legen

ASE_Funk.stopListening();                                     // Stoppt den Empfang
ASE_Funk.openWritingPipe(Addresses[0]);                       // Öffnen einer Pipe zum Senden
ASE_Funk.write(&Daten_Senden, sizeof(Daten_Senden));         // Datengröße wird ermittelt und aus
                                                                vorgesehenem Speicher ausgelesen
                                                                und gesendet
ASE_Funk.openReadingPipe(1, Addresses[0]);                    // Öffnen einer Pipe zum Empfangen
ASE_Funk.startListening();                                    // Startet den Empfang

}

void software_Reset()                                         // Reset Methode
{
asm volatile (" jmp 0");                                       // Funktion der Reset Methode
}

```

15.2.7 Force-Feedback-Funktion Greifer

```
#include <SPI.h> // Einbindung der SPI Bibliothek (Serial Peripheral
                // Interface)
#include <RF24.h> // Einbindung der RF24 Bibliothek (WLAN Kommunikation)
#include <Dynamixel2Arduino.h> // Einbindung der Dnamixel2Arduino Bibliothek
                               (Kommunikation Arduino und Dynamixel Servos)

//===== Global Servos =====

#if defined(ARDUINO_AVR_UNO) || defined(ARDUINO_AVR_MEGA2560) // Selektion ob Arduino Uno
                                                             // oder Arduino Mega
                                                             // definiert wurde
    #define Greifer_SERIAL Serial // Definition der
                                   // Kommunikationsart Serial
    const uint8_t Greifer_DIR_PIN = 2; // Deklaration und
                                       // Initialisierung der
                                       // Dynamixel Shield
                                       // Direction Pin Variable
#else // Bei nicht zutreffender
    #define Greifer_SERIAL Serial1 // Definition der
                                    // Kommunikationsart Serial1
    const uint8_t Greifer_DIR_PIN = 2; // Deklaration und
                                       // Initialisierung der
                                       // Dynamixel Shield
                                       // Direction Pin Variable
#endif // Ende der Selektion

const uint8_t Gripper_ID1 = 1; // Deklaration und
                                // Initialisierung der
                                // Gripper1 ID Variable
const uint8_t Gripper_ID2 = 2; // Deklaration und
                                // Initialisierung der
                                // Gripper2 ID Variable
const uint32_t Timeout=100; // Deklaration und
                             // Initialisierung der
                             // timeout Variable für die
                             // Kommunikation zwischen
                             // Dynamixel Shield und
                             // Dynamixel Servo in ms

const float Greifer_PROTOCOL_VERSION = 1.0; // Deklaration und
                                              // Initialisierung der
                                              // Protokollversions
                                              // Variable

Dynamixel2Arduino Gripper1(Greifer_SERIAL, Greifer_DIR_PIN); // Instanziierung des
                                                             // Gripper1 Objekts
Dynamixel2Arduino Gripper2(Greifer_SERIAL, Greifer_DIR_PIN); // Instanziierung des
                                                             // Gripper2 Objekts

//===== Global WLAN Kommunikation =====

RF24 Greifer_Funk (9, 10); // Instanziierung des Greifer-Funk Objekts
byte Addresses[][6] = {"0"}; // Deklaration und Initialisierung der Pipe für die
                              // Funkübertragung
int Daten_Empfangen[2]; // Deklaration des Empfangsdatenspeichers

//===== Global Datenverarbeitung =====

int Position; // Deklaration der Variable für die Greifer Position
int Greifer_Position; // Deklaration der Variable für die Greifer Position für die ASE
int ASE_Last; // Deklaration der Variable für die ASE Last
int Greifer_Last; // Deklaration der Variable für die Greifer Last
int Greifer_Last_Alt; // Deklaration der Variable für die alte Greifer Last
int Greifer_Last_Smooth; // Deklaration der Variable für den gleitenden Mittelwert der
                          // Greifer Last

void setup() { // Void setup

//===== Setup Servos =====

    Gripper1.begin(115200); // Beginn der Kommunikation
                             // zwischen Gripper1 und
                             // Arduino Uno mit
                             // eingestellter Baudrate

    Gripper2.begin(115200); // Beginn der Kommunikation
                             // zwischen Gripper2 und
```



```

Gripper1.setPortProtocolVersion(Greifer_PROTOCOL_VERSION);
Gripper2.setPortProtocolVersion(Greifer_PROTOCOL_VERSION);
Gripper1.setOperatingMode(Gripper_ID1, OP_POSITION);

Gripper2.setOperatingMode(Gripper_ID2, OP_POSITION);

Gripper1.torqueOn(Gripper_ID1);
Gripper2.torqueOn(Gripper_ID2);
if (Gripper1.ping(Gripper_ID1) == false)
{
    software_Reset();
}

else
{
Gripper1.writeControlTableItem(PUNCH,1,32,Timeout);

Gripper2.writeControlTableItem(PUNCH,2,32,Timeout);

Gripper1.writeControlTableItem(TORQUE_LIMIT,1,204,Timeout);

Gripper1.writeControlTableItem(MOVING_SPEED,1,204,Timeout);

Gripper2.writeControlTableItem(TORQUE_LIMIT,2,204,Timeout);

Gripper2.writeControlTableItem(MOVING_SPEED,2,204,Timeout);

delay(2000);

Gripper1.setGoalPosition(1,356);
Gripper2.setGoalPosition(2,356);
delay(2000);
Gripper1.setGoalPosition(1,522);
Gripper2.setGoalPosition(2,522);
delay(2000);
Gripper1.setGoalPosition(1,439);
Gripper2.setGoalPosition(2,439);
delay(2000);
Gripper1.writeControlTableItem(TORQUE_LIMIT,1,1023,Timeout);
Gripper1.writeControlTableItem(MOVING_SPEED,1,1023,Timeout);

```

Arduino Uno mit
 eingestellter Baudrate
 // Setzen der Protokoll
 Version für Gripper1
 // Setzen der Protokoll
 Version für Gripper2
 // Setzen des
 Operationsmodus für
 Gripper1 (Fahrbetrieb
 über Position)
 // Setzen des
 Operationsmodus für
 Gripper2 (Fahrbetrieb
 über Position)
 // Aktivierung des Gripper1
 // Aktivierung des Gripper2
 // Methode des Software
 Reset
 // Herabsetzen der
 Punch_Funktion für einen
 weicheren Betrieb vom
 Gripper1
 // Herabsetzen der
 Punch_Funktion für einen
 weicheren Betrieb vom
 Gripper2
 // Herabsetzen des Gripper1
 Drehmoments zur Erhöhung
 der Sicherheit bei der
 Startroutine (20%)
 // Herabsetzen der Gripper1
 Geschwindigkeit zur
 Erhöhung der Sicherheit
 bei der Startroutine
 (20%)
 // Herabsetzen des Gripper2
 Drehmoments zur Erhöhung
 der Sicherheit bei der
 Startroutine (20%)
 // Herabsetzen der Gripper2
 Geschwindigkeit zur
 Erhöhung der Sicherheit
 bei der Startroutine
 (20%)
 // Wartezeit zur
 Verarbeitung der oben
 aufgeführten
 Einstellungen
 // Fahren des Gripper1 auf
 90°
 // Fahren des Gripper2 auf
 90°
 // Wartezeit für den
 Fahrbetrieb auf 90°
 // Fahren des Gripper1 auf
 0°
 // Fahren des Gripper2 auf
 0°
 // Wartezeit für den
 Fahrbetrieb auf 0°
 // Fahren des Gripper1 auf
 45°
 // Fahren des Gripper2 auf
 45°
 // Wartezeit für den
 Fahrbetrieb auf 45°
 // Heraufsetzen des Gripper1
 Drehmoments für den
 Nennbetrieb
 // Heraufsetzen der Gripper1
 Geschwindigkeit für den
 Nennbetrieb

```

Gripper2.writeControlItem(TORQUE_LIMIT,2,1023,Timeout); // Heraufsetzen des Gripper2
Drehmoments für den
Nennbetrieb
Gripper2.writeControlItem(MOVING_SPEED,2,1023,Timeout); // Heraufsetzen der Gripper2
Geschwindigkeit für den
Nennbetrieb

//===== Setup WLAN Kommunikation =====

Greifer_Funk.begin(); // Aktivierung des NRF24L01 Funkmoduls
Greifer_Funk.setChannel(45); // Einstellung der Sende
//Empfangsfrequenz
Greifer_Funk.setPALevel(RF24_PA_MAX); // Einstellung der Sendeleistung
Greifer_Funk.setDataRate(RF24_2MBPS); // Einstellung der Datenübertragungsrate
Greifer_Funk.openReadingPipe(1, Addresses[0]); // Öffnen einer Pipe zum Empfangen
Greifer_Funk.startListening(); // Startet den Empfang
Greifer_Funk.setAutoAck(1); // Aktivierung der Acknowledge_Funktion
Greifer_Funk.enableDynamicPayloads(); // Aktivierung der dynamischen
//Paketgröße
Greifer_Funk.setCRCLength(RF24_CRC_16); // Setzen der CRC-Prüflänge auf 16 Bit
}
}

void loop() { // Void loop

if (Greifer_Funk.available()) // Prüft, ob Daten
empfangen werden
{
while (Greifer_Funk.available()) // Solange Daten
empfangen werden
{
Greifer_Funk.read(&Daten_Empfangen, sizeof(Daten_Empfangen)); // Datengröße wird
ermittelt und im
vorgesehenem Speicher
abgelegt
}

int Position_Empfangen = Daten_Empfangen[0]; // Deklaration und
Initialisierung der
Variable

Position = map(Daten_Empfangen[0],500,820,522,356); // Rechnet die empfangene
ASE Position um für
die Greifer Position

//===== Verriegelung der Greifer Position =====

if (Position > 522) // Prüft, ob der maximale Grenzwert der Greifer Position
überschritten wurde
{
Position = 522; // Setzt die Greifer Position auf den maximalen Grenzwert
}

if (Position < 356) // Prüft, ob der minimale Grenzwert der Greifer Position
unterschritten wurde
{
Position = 356; // Setzt die Greifer Position auf den minimalen Grenzwert
}

//===== Ende der Verriegelung =====

Gripper1.setGoalPosition(1,Position); // Gripper1 fährt auf die umgerechnete Position
Gripper2.setGoalPosition(2,Position); // Gripper2 fährt auf die umgerechnete
Position
}

delay(10); // Wartezeit für den multitasking Betrieb der
Funkübertragung

Greifer_Funk.stopListening(); // Stoppt den Empfang
Greifer_Position = Gripper1.getPresentPosition(1); // Auslesen der aktuellen Greifer
-position

```

```

Greifer_Position=map(Greifer_Position,522,356,500,820);    // Rechnet die aktuelle
                                                         Position des Greifers um
                                                         für die ASE
Greifer_Last = Gripper2.readControlTableItem (PRESENT_LOAD,2,Timeout);    //Auslesen und
                                                         speichern der
                                                         aktuellen
                                                         Greifer Last
ASE_Last = Daten_Empfangen[1];    // Speichern der
                                                         empfangenen ASE
                                                         Last

//===== Lastrichtungsfilter =====

if(ASE_Last >= 1024 && Greifer_Last >= 1024)    // Prüft, ob die ASE im Lastbereich
                                                         ZU ist und ob der Greifer im
                                                         Lastbereich ZU ist
{
    Greifer_Last_Alt=Greifer_Last;    // Setzt die alte Greifer Last auf
                                                         den Wert der aktuellen Greifer Last
}

if(ASE_Last >= 1024 && Greifer_Last < 1024)    // Prüft, ob die ASE im Lastbereich ZU
                                                         ist und ob der Greifer im Lastbereich
                                                         AUF ist
{
    Greifer_Last = Greifer_Last_Alt;    // Setzt die aktuelle Greifer Last auf
                                                         den Wert der alten Greifer Last
}

if(ASE_Last <= 1023 && Greifer_Last <= 1023)    // Prüft, ob die ASE im Lastbereich AUF
                                                         ist und ob der Greifer im Lastbereich
                                                         AUF ist
{
    Greifer_Last_Alt = Greifer_Last;    // Setzt die alte Greifer Last auf den
                                                         Wert der aktuellen Greifer Last
}

if(ASE_Last <= 1023 && Greifer_Last >1023)    // Prüft, ob die ASE im Lastbereich AUF
                                                         ist und ob der Greifer im Lastbereich
                                                         ZU ist
{
    Greifer_Last = Greifer_Last_Alt;    // Setzt die aktuelle Greifer Last auf
                                                         den Wert der alten Greifer
                                                         Last
}

//===== Lastglättung =====

Greifer_Last_Smooth = 0.80 * Greifer_Last_Smooth + 0.20 * Greifer_Last;    // Bildet den
                                                         gleitenden
                                                         Mittelwert der
                                                         gewichteten
                                                         Lastwerte des
                                                         Greifers

int Daten_Senden[2]= {Greifer_Position,Greifer_Last_Smooth};    // Deklaration
                                                         und Befüllung
                                                         des
                                                         Sendedatenspeichers
                                                         mit der gefilterten
                                                         Greifer Position
                                                         und der geglätteten
                                                         Greifer Last

Greifer_Funk.openWritingPipe(Addresses[0]);    // Öffnen einer Pipe zum Senden
Greifer_Funk.write(&Daten_Senden, sizeof(Daten_Senden));    // Datengröße wird ermittelt und
aus vorgesehenem Speicher ausgelesen und gesendet
Greifer_Funk.openReadingPipe(1, Addresses[0]);    // Öffnen einer Pipe zum
Empfangen
Greifer_Funk.startListening();    // Startet den Empfang
}

void software_Reset()
{
    asm volatile (" jmp 0");
}

```