Zur Erlangung des Grades

eines

Bachelor of Engineering (B. Eng.)

von Herrn Sebastian Stolze

geboren am: 21.Dezember.1992

in: Merseburg

vorgelegte Abschlussarbeit:

Thema:

„Entwicklung und Implementierung eines Netzwerk-Protokolls für die Steuerung eines biomechanischen Trainingsgeräts“

Erstprüfer: Prof. Dr.-Ing. Peter Helm

Zweitprüfer: M. Eng. Lars Agricola

Merseburg, 06.10.2015

Bibliografische Beschreibung

Stolze, Sebastian: Entwicklung und Implementierung eines Netzwerk-Protokolls für die Steuerung eines biomechanischen Trainingsgeräts, Hochschule Merseburg, Bachelorarbeit, Erfurt 2015, 55 S., 22 Abb., 15 Tab.

Inhaltsverzeichnis **Seite**

1 Abkürzungsverzeichnis

CPM:	Continuous Passive Movement
CPU:	Central Processing Unit
HMI:	Human Machine Interface
KPB:	Kontinuierliche Passive Bewegung
PLC:	Programmable Logic Controller
SPS:	Speicherprogrammierbare Steuerung
ST:	Strukturierter Text
RAT:	Rex Automatisierungstechnik GmbH

2 Einleitung

Die Firma *Rex Automatisierungstechnik GmbH* arbeitet an Automatisierungslösungen im Bereich der Steuerungstechnik für Produktions- und Fertigungsmaschinen. Es werden sowohl PLC-Programmierung, als auch HMI-Programmierung abgedeckt und so komplette Maschinen aus unterschiedlichsten Produktionsbereichen automatisiert. So werden z.B. Maschinen mit Steuerungen unterschiedlicher Firmen wie z.B. Siemens, Wago, Eckelmann, etc. je nach Kundenwunsch automatisiert. Die Rex Automatisierungstechnik GmbH steht in Kooperation mit der *Eckelmann Group* und verwendet deshalb bevorzugt Steuerungen der Firma Eckelmann, welche mit CoDeSys V2.3 programmiert werden.

Zur Verfügung steht ein Trainingsgerät aus der Medizintechnik, welches aus einem elektrischen Antrieb und einem Controller besteht. Für die Einstellung des Trainingsgeräts wird eine Bedien-Software verwendet. Die Software und der Controller werden benötigt, um das Trainingsgerät in Bewegung zu versetzen. Der bis jetzt genutzte Controller soll durch eine neue Steuerung der Firma Eckelmann ersetzt werden (siehe Abbildung 1). Zu entwickeln ist eine Kommunikationsschnittstelle zwischen der vorhandenen Bedien-Software und einer neuen Steuerung. Hierbei muss die Kommunikation zwischen Controller und Bedien-Software untersucht werden, um sie der neuen Steuerung zur Bearbeitung bereitzustellen. Die Programmierung der zu implementierenden Steuerung erfolgt mit CoDeSys V2.3.

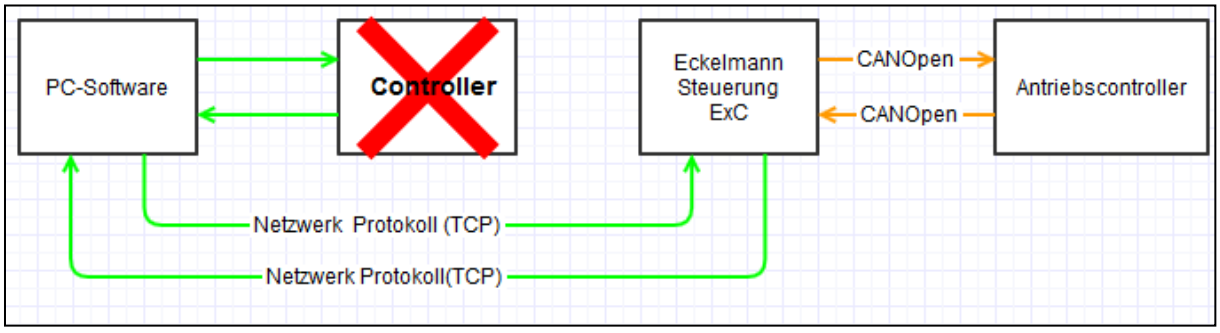


Abb. 1, Kommunikationsschema, Eigene Darstellung

Zusätzlich müssen aus der alten Steuerung die vorhandenen Umrechnungen und Skalierungen von Sensoren und Parametern beachtet werden. Zum Aufruf der Trainingsprogramme und für die Übermittlung der benötigten Eingangs- und Ausgangswerte muss hierzu die Kommunikation zwischen Bedien-Software und eingebautem Controller untersucht werden. Anschließend sind ein geeignetes Netzwerkprotokoll zu ermitteln und der Telegrammaufbau festzulegen. Es ist zu ermitteln, welche Konzepte zur Implementierung eines Netzwerkprotokolls und dem damit verbundenen Telegrammaufbau technisch sinnvoll sind und zu bewerten, mit welchem Aufwand und Nutzen diese eingebunden werden können.

Das Trainingsgerät enthält ein Antriebs-System mit folgenden Komponenten:

- Motor
- Getriebe
- Drehmoment- Messeinrichtung
- Drehwinkel- Messeinrichtung
- Verstellbare Endlagen- Schalter
- Antriebs- Regler

Dieses Trainingsgerät dient als Antrieb und Messeinrichtung und findet seinen Einsatz:

- im Sporttraining
- bei Übungen von Arbeitsbewegungen
- in der Physiotherapie
- bei wissenschaftlichen Untersuchungen

(1)….Grundgestell
(2)....Sitz
(3)....Rückenlehne
(4)....Führungsprofile
(5)....Trainingsgerät
(6)....Hubsäule
(7)....Schwenkarm
(8)....Kabelschutzschlauch
(9)....Verbindungsstecker Controller

Abb. 2: Trainingsgerät, [2] S.39

3 Entwicklung der Kommunikation

3.1 Technische Grundlagen Trainingsgerät

Für das Trainingsgerät sind verschiedene Trainingsabläufe einstellbar, dabei untergliedert sich die Auswahl in Betriebsart und Belastungsart.

Bei der Betriebsart stehen folgende Trainingsabläufe zur Auswahl:

- Freie Belastung
- Geschwindigkeit begrenzt
- Geschwindigkeit definiert
- Last definiert

Eine genauere Beschreibung und Gliederung der Betriebsarten folgt auf der nachfolgenden Seite.

Die Belastungsarten gliedern sich in:

- Exzentrisch:

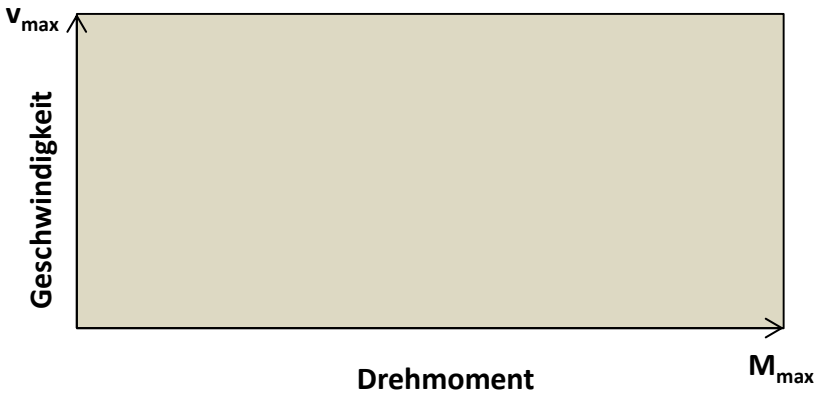
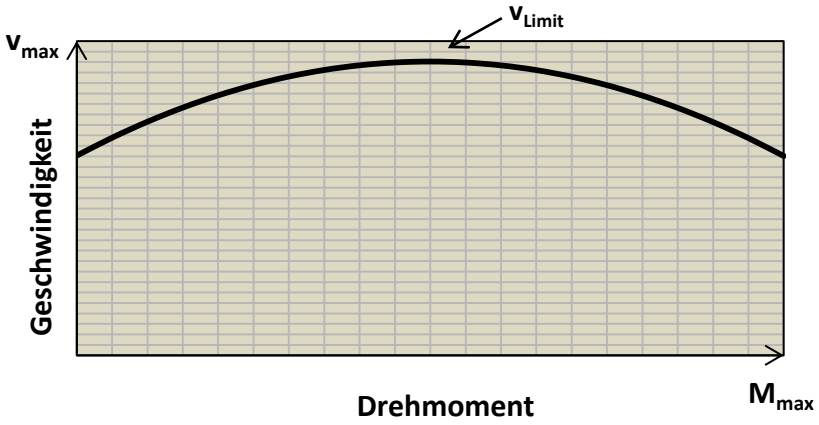
  Eine exzentrische Belastung, ist die Belastung des Muskels entgegen der Bewegungsrichtung, dabei wird der Muskel durch die äußere Last verlängert z.B. Gewicht ablegen.

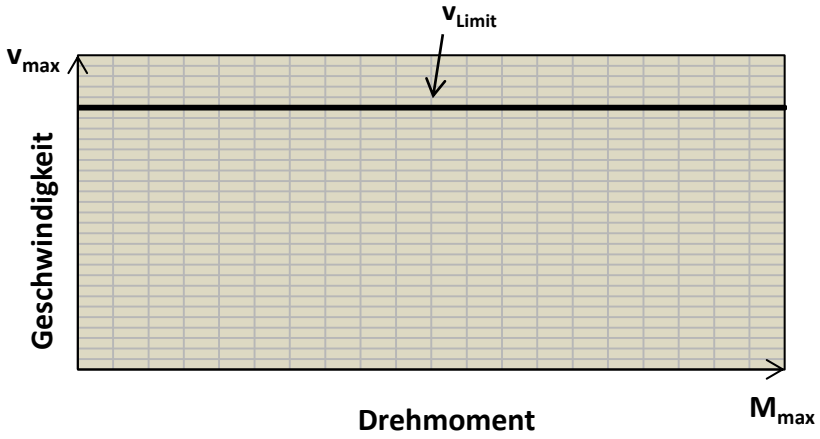
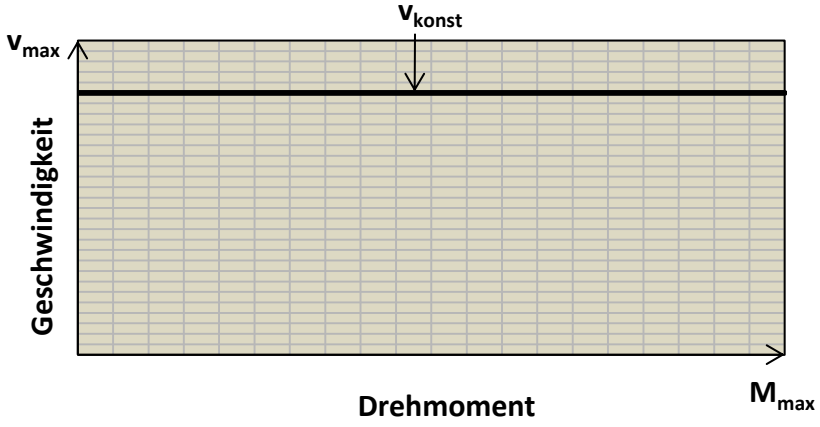
- Konzentrisch:

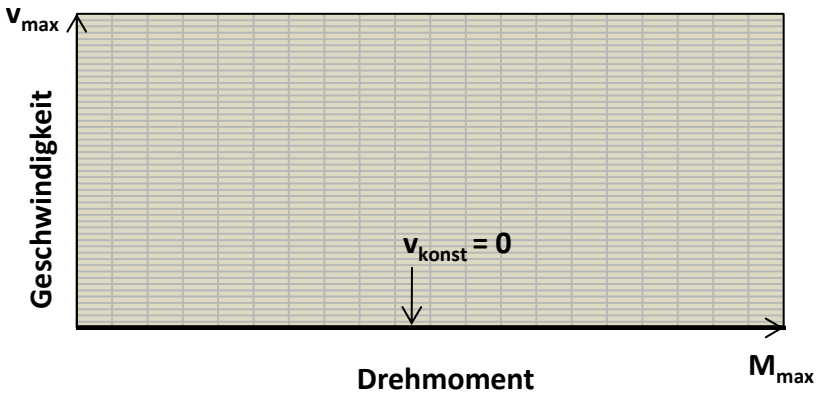
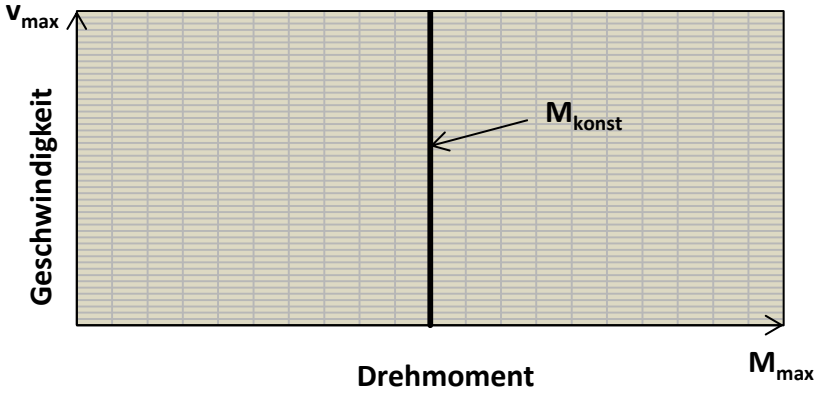
  Bei der konzentrischen Belastung hingegen wird der Muskel in Bewegungsrichtung belastet und dabei durch die äußere Last verkürzt z.B. Gewicht heben.

- Kontinuierlich Passive Bewegung (KPB):

  Bei der Kontinuierlich Passiven Bewegung wird der Muskel vom Trainingsgerät mit gleichbleibender Geschwindigkeit bewegt. Dabei kann der Proband selber Kraft aufbringen ohne eine Geschwindigkeitsänderung herbeizurufen, oder der Muskel wird vom Trainingsgerät selbstständig bewegt ohne dass der Proband selber Kraft aufbringen muss.

Betriebsart	Erklärung
Freie Bewegung	Der Proband kann sich am Trainingsgerät frei bewegen ohne zusätzliche Belastung durch den Antrieb zu erhalten. Dabei werden das aufgebrachte Drehmoment und die Geschwindigkeit des Probanden aufgezeichnet. 
Geschwindigkeit begrenzt	Der Proband trainiert mit einem Geschwindigkeitslimit. Das bedeutet, es ist ihm nicht möglich diese Geschwindigkeit zu überschreiten, jedoch kann er diese unterschreiten. Es kann somit mehr Drehmoment vom Probanden aufgebracht werden. Die Geschwindigkeit bleibt dabei am Limit. Das Geschwindigkeitslimit ist variabel zum aufgebrachten Drehmoment des Probanden. 

	Eine spezielle Form dieser Betriebsart ist die konstante Festlegung eines Geschwindigkeitslimits. Hierbei kann das Geschwindigkeitslimit nicht überschritten werden. Das Geschwindigkeitslimit bleibt in jeder Position gleich. Dadurch passt sich das Drehmoment des Trainingsgeräts, je nach aufgebrachtem Drehmoment des Probanden an.  In dieser Betriebsart werden Drehmoment, Geschwindigkeit und Position gemessen und aufgezeichnet.
Geschwindigkeit definiert	Das Weg bzw. Geschwindigkeitsprofil wird abgefahren und das ausgeübte Drehmoment des Probanden wird gemessen. Standardfall ist die Festlegung der Geschwindigkeit auf einen konstanten Wert. Dabei bewegt sich das Trainingsgerät mit gleichbleibender Geschwindigkeit. Hierbei wird zum Beispiel der Proband bewegt, ohne selbst irgendeine Kraft aufbringen zu müssen. 

	Ein Spezialfall ist, die konstante Geschwindigkeit auf 0 zu setzen. Dann ist keine äußere Bewegung des Trainingsgeräts möglich.  In dieser Betriebsart wird nur das Drehmoment aufgezeichnet und gemessen.
Last definiert	In jeder Position wirkt das gleiche Drehmoment. Das Trainingsgerät bewegt sich erst, wenn das vom Probanden aufgebrachte Drehmoment größer ist als ein vordefiniertes Drehmoment. 

Tab.1: Betriebsarten, Eigene Darstellung

In den meisten Betriebsarten ist sowohl eine *konzentrische*, als auch eine *exzentrische* Belastung möglich. Jedoch ist nicht jede Betriebsart mit jeder Belastungsart kombinierbar. Es gibt verschiedene Trainingsprogramme, die den Trainingsablauf definieren. Die Trainingsprogramme umfassen drei für den Bewegungsablauf wichtige Positionen. Die mechanische Nullposition, womit die Ausgangslage des Trainingsgeräts und die Ausgangsposition des Probanden eingestellt werden, sowie die beiden Begrenzungspositionen bis zu welchen sich das Trainingsgerät maximal bewegen darf. Zur Übersicht wird in Tabelle 2 eine Auflistung der Trainingsprogramme dargestellt. Die verschiedenen Belastungsarten können je nach Bewegungsrichtung an den Umkehrpunkten der Bewegung geändert werden.

Betriebsart	**1. Belastungsart** (Bewegungsrichtung 1)	**2. Belastungsart** (Bewegungsrichtung 2)
Freie Bewegung	-	-
Geschwindigkeit begrenzt	konzentrisch	Konzentrisch
	konzentrisch	Exzentrisch
	exzentrisch	Konzentrisch
	exzentrisch	Exzentrisch
Geschwindigkeit definiert/ begrenzt	KPB	Konzentrisch
	KPB	Exzentrisch
	konzentrisch	KPB
	exzentrisch	KPB
Geschwindigkeit definiert	KPB	KPB
Geschwindigkeit definiert 0	-	-
Last definiert	konzentrisch	Konzentrisch
	konzentrisch	Exzentrisch
	exzentrisch	Konzentrisch
	exzentrisch	Exzentrisch

Tab.2: Trainingsprogramme, Eigene Darstellung

3.2 Theoretische Grundlagen

3.2.1 Netzwerkkommunikation

Für die Nachrichtenübertragung wird die Verbindung mittels Ethernet genutzt. Das Ethernet-Protokoll deckt hierbei die Bitübertragungsschicht und die Sicherungsschicht im OSI-Referenzmodell ab (siehe Abbildung 3). In der Transportschicht wird das TCP (Transmission Control Protocol) zwischen Bedien-Software und Steuerung verwendet. Es bietet im Vergleich zu UDP (User Datagram Protocol) den Vorteil eines zuverlässigen, verbindungsorientierten Transportprotokolls.

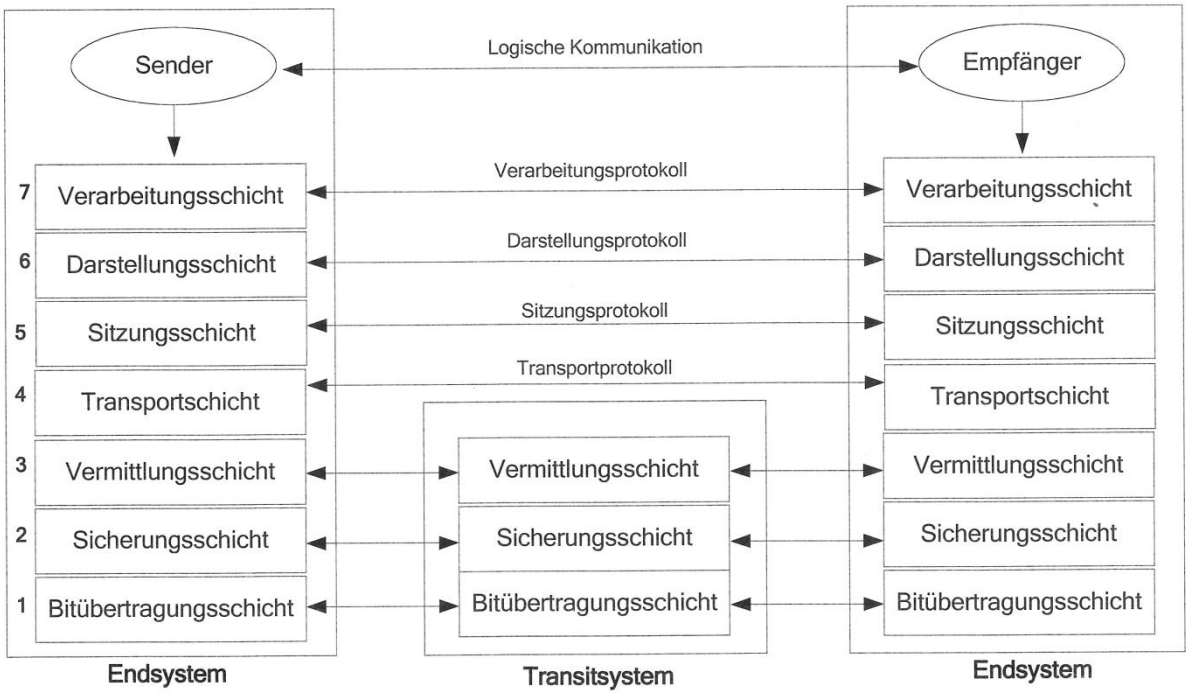


Abb. 3: OSI-Referenzmodell, [6] S. 2

Bei einer verbindungsorientierten Übertragung wird nach dem Verbindungsaufbau für eine stetige Verbindung gesorgt, diese bleibt so lange bestehen, bis sie explizit wieder getrennt wird. Dadurch muss sich bei der Kommunikation nicht mit der Verbindung beschäftigt werden, sondern nur noch mit der Datenübertragung. TCP garantiert dabei eine vollständige und korrekte Übertragung der Daten. Eine Sicherung der Daten geschieht hierbei durch das Verwenden einer Prüfsumme, um die Korrektheit der Daten zu garantieren. Desweiteren wird der Verlust von Nachrichten durch einen Timer verhindert. Dieser Timer wird beim Versenden der Nachricht gestartet und nach einer Bestätigung durch den Empfänger wieder zurückgesetzt. Sollte die Zeit des Timers jedoch abgelaufen sein, so wird wiederholt versucht die Nachricht zu versenden. Die Reihenfolge der Daten ist über eine Sequenznummer gesichert.

Im Gegensatz dazu bietet UDP nur einen Leistungsvorteil, wenn eine gewisse Unzuverlässigkeit in Kauf genommen werden kann. Mit UDP ist nicht garantiert, dass die Daten unverfälscht und vollständig übermittelt werden. Die Überprüfung und Kontrolle dieser Nachrichten, müsste bei UDP in der Sitzungs- oder Darstellungsschicht des OSI-Referenzmodells implementiert werden, um eine Sicherung der Daten zu garantieren. [Vgl.[6], S. 187-188]

Der Verbindungsaufbau beim TCP-Protokoll (siehe Abbildung 4) erfolgt zwischen TCP-Server und TCP-Client. Hierbei ist der Client der aktive Teilnehmer, welcher ein Signal zum Verbindungsaufbau versendet und der Server der passive Teilnehmer, welcher auf ein eingehendes Signal wartet. Die Bedien-Software ist TCP-Client und versendet auf einem festgelegten Port ein TCP-Protokoll an die Steuerung. Auf Grundlage dieser Server-Client-Beziehung ist der Verbindungsaufbau entsprechend zu implementieren. Die Bedien-Software ist als Client schon voreingestellt, somit muss die Steuerung die Rolle des Servers übernehmen und als solcher programmiert und eingestellt werden. Hierfür muss der Steuerung mitgeteilt werden, auf welchen Port sie „hören“ muss. Bei Eingang eines Synchronisations Signals, werden Port und IP-Adresse des Senders als Verbindung festgelegt und an den Server gebunden. Die Verbindung wird über ein Bestätigungstelegramm gestartet und die Datenübertragung über den festgelegten Port kann beginnen. Die Daten können hierbei so lange übertragen werden, bis die Verbindung wieder über ein Telegramm geschlossen wird.

Schema TCP-Verbindungsaufbau:

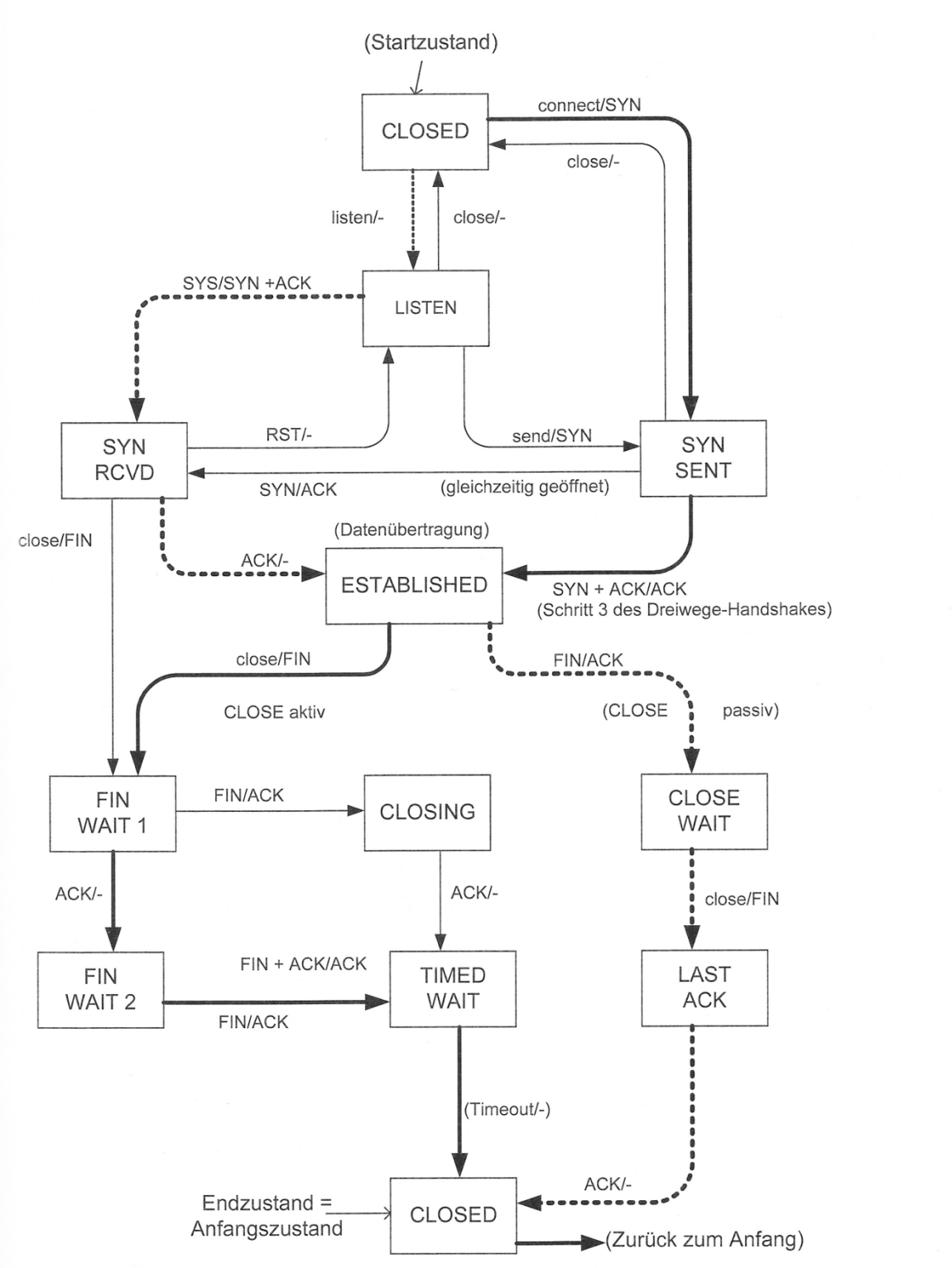


Abb. 4: TCP-Verbindungsaufbau, [6] S.215

3.2.2 Kommunikationsschnittstelle

Zu implementieren ist eine Kommunikationsschnittstelle zwischen Bedien-Software und Steuerung, welche die Datenübertragung realisiert und eine Auswertung der empfangenen Informationen auf der Steuerung vornimmt. Bei der Übertragung des Telegramms sind prinzipiell zwei Konzepte möglich.

Zum Einen können die Daten der Bedien-Software direkt an die Steuerung weitergeleitet, dort empfangen und verarbeitet werden. Zum Anderen können die Rohdaten an eine zweite Software gesendet werden. In dieser Zwischensoftware wird ein neues Telegrammprotokoll erzeugt und das so erstellte Telegramm an die Steuerung übermittelt. Im Folgenden gilt es, die beiden Übertragungskonzepte detaillierter zu beschreiben und abzuwägen, welches dieser beiden Konzepte geeigneter für die Implementierung ist.

3.2.2.1 Datenübertragung und Auswertung über SPS

Das erste Übertragungskonzept ist in Abbildung 5 dargestellt und wird in diesem Abschnitt näher erläutert. Die Datenübertragung zwischen SPS und Bedien-Software findet direkt über Ethernet statt. Die Informationen von der Bedien-Software werden über die Ethernet-Schnittstelle per TCP/IP-Protokoll versendet. Nach Empfang des Protokolls werden die ASCII codierten Nutzdaten vom TCP-Protokoll extrahiert, von der SPS gespeichert und zur Verarbeitung bereitgestellt. In der Steuerung müssen die empfangenen ASCII Kommandos überprüft und je nach Kommando eine entsprechende Anweisung an den Antriebscontroller und die Bedien-Software gesendet werden. Hierbei müssen alle Kommandos von der Steuerung interpretierbar sein, die von der Bedien-Software zur Steuerung gesendet werden. Für die Interpretation der Kommandos wird eine gezielte Analyse der Kommunikation zwischen Bedien-Software und SPS benötigt. Zum Einen ist von Bedeutung, welche Befehle die Steuerung übermittelt und zum Anderen, wie die Daten an die Software zurückgesendet werden müssen. Eine falsche Formatierung der zu sendenden Daten oder eine nicht zu erwartende Antwort der Steuerung, würde die Software entweder zum Abbruch bringen oder eine Verfälschung von Messungen/ Trainingseinstellungen bzw. eine Fehlinterpretation hervorrufen. Sobald die Daten in der Steuerung ausgewertet sind, wird eine Antwort an die Software zurückgesendet, die den Empfang der Daten bestätigt und benötigte Informationen zurückliefert. Hierbei muss der Inhalt der zu sendenden Nutzdaten ebenfalls ASCII codiert sein. Außerdem ist zu beachten, dass die Skalierungen und Wertebereiche von Messdaten gleich den Formaten der Bedien-Software sind.

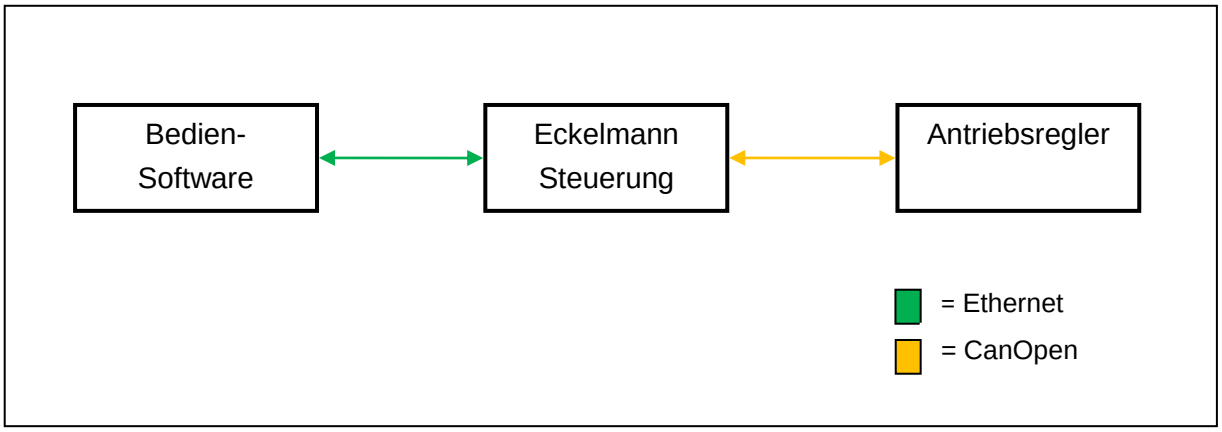


Abb. 5, Konzept 1, Eigene Darstellung

3.2.2.2 Neuer Telegrammaufbau und Verarbeitung mit Software und SPS

Eine zweite Möglichkeit die Kommunikation zwischen Steuerung und Bedien-Software zu realisieren, ist es den Datenaustausch über eine Kommunikations-Software zu realisieren (siehe Abbildung 6). Dabei werden die versendeten Daten der Bedien-Software von der Kommunikations-Software empfangen und direkt verarbeitet. Die ausgewerteten Daten werden dann über ein zu entwickelndes Telegramm an die Steuerung gesendet. Aufgabe der Kommunikations-Software ist es die empfangenen Informationen zu interpretieren und über den neuen Telegrammaufbau an die Steuerung zu senden. Daraufhin muss die Steuerung die im Telegramm enthaltenen Informationen auswerten, ausführen und eine Antwort an die Kommunikations-Software senden. Die Kommunikations-Software dient hierbei als Schnittstelle zwischen Bedien-Software und Steuerung

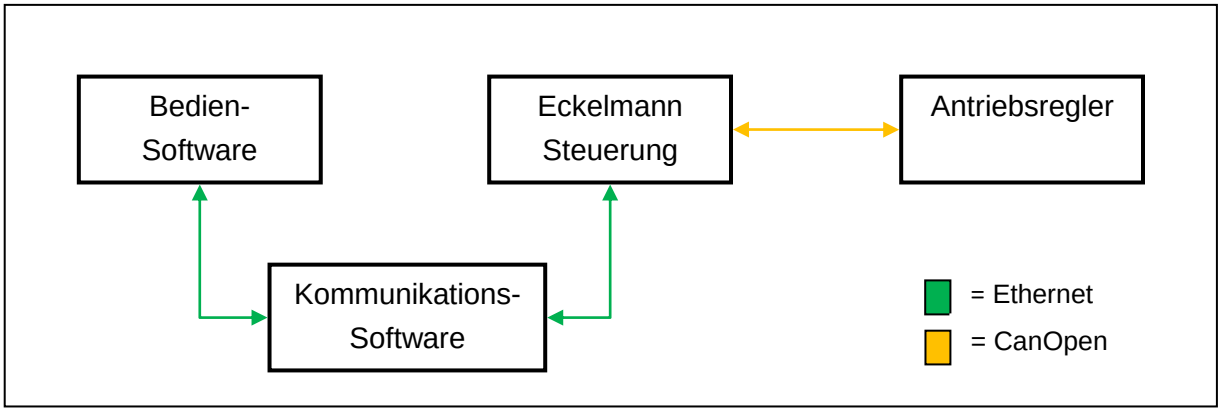


Abb. 6, Konzept 2, Eigene Darstellung

Ein Telegrammaufbau ist wie folgt realisierbar:

STX	Header	Länge	TLN_ID	SEQ_ID	TYPE_ID	Message	ETX
1 Byte	1 Byte	2 Byte	2 Byte	2 Byte	2 Byte	x Byte	1 Byte
0x02	0x44					ASCII	0x03

Abb. 7: Telegrammaufbau, Eigene Darstellung

- Steuerbyte STX:
  - Startkennung des Telegramms
- Header-Version:
  - Unterscheidung von möglichen Headern
- Länge:
  - Angabe der Nachrichtenlänge 0 bis 525
  - TCP ermöglicht maximal 536 Byte Datenlänge pro Paket. Abzüglich der 11 Byte für den Telegramm-Header ergeben sich maximal 525 Byte Nachrichtenlänge.
- Teilnehmernummer TLN_ID:
  - Angabe des sendenden Teilnehmers
- Sequenznummer SEQ_ID:
  - Angabe der Sequenz zur Zuordnung von Nachrichten
- Typnummer TYP_ID:
  - Angabe des Nachrichtentyps
- Message:
  - Nutzdaten
- Endbyte ETX:
  - Endekennung des Telegramms

Mit diesem Telegrammaufbau sind definierte Nachrichtentypen notwendig, welche über die TYP_ID festgelegt werden. Der Nachrichtentyp gibt an, welche Art von Nachricht im Telegramm übertragen wird. Ein Aufbau der Nachrichtentypen unterscheidet sich je nach Übertragungsrichtung. Die Nachricht kann entweder von der Kommunikations-Software an die Steuerung oder von der Steuerung an die Kommunikations-Software gesendet werden. Bei der Kommunikation sind Daten wie: Bestätigungen, Messwerte, Parameter und Programmabläufe zu übertragen. Eine nähere Beschreibung dieser Daten erfolgt in der Kommunikationsanalyse in Kapitel 3.3. Die entwickelte Unterteilung der Nachrichtentypen ist in Tabelle 3 und 4 dargestellt.

Übertragung: Kommunikations-Software → Steuerung:

TYP_ID	Beschreibung	Message
0	Bestätigungstelegramm	keine Nachricht oder Abfrage von Zuständen (z.B.: Fehler, Statusanzeigen)
1	Messwertübertragung (Drehmomentmesswerte abfragen)	-
2	Messwertübertragung (Positionsmesswerte abfragen)	-
3	Programmaufruf (Trainingsprogramme oder Programme zum Einstellen)	Programm-Nummer (1- 20)
4	Parameter übertragen	Einstellbare Parameter (Trainingsparameter)

Tab. 3: Nachrichtentypen an die SPS, Eigene Darstellung

Übertragung: Steuerung → Kommunikations-Software:

TYP_ID	Beschreibung	Message
0	Bestätigungstelegramm	Rückmeldung Zustandswert (z.B.: Motor läuft)
1	Messwertübertragung (Drehmoment)	Drehmomentmesswerte
2	Messwertübertragung (Position)	Positionsmesswerte
3	Bestätigung Programmaufruf	Bestätigung Programm-Nummer (1- 20) gestartet
4	Bestätigung Parameter übertragen	Bestätigung Einstellbare Parameter geladen

Tab. 4: Nachrichtentypen von der SPS, Eigene Darstellung

3.2.2.3 Auswertung und Fazit

Im Folgenden gilt es zu entscheiden, ob das Kommunikationskonzept mit oder ohne zusätzlicher Kommunikations-Software zu realisieren ist. Hierfür werden einige Vor- und Nachteile der jeweiligen Konzepte gegenübergestellt und miteinander verglichen (siehe Tabelle 5).

Bei der Realisierung ohne Kommunikations-Software wird keine zusätzliche Software benötigt und die Struktur der Programmierung bleibt einfach und übersichtlich. Die Programmierung findet nur auf der Steuerung statt und es wird keine zusätzliche Programmierung einer PC-Software benötigt. Desweiteren ist die Übertragungszeit bei diesem Konzept geringer, weil keine zusätzliche Bearbeitungszeit, in welcher die Daten von der Kommunikations-Software verarbeitet werden, entsteht. Ein Nachteil dieses Konzepts, ist die Kommunikationsfähigkeit mit einer neuen Bedien-Software. Wenn ein Konzept ohne Kommunikations-Software gewählt wird, ist es möglich die neue Steuerung oder den alten Controller über die derzeit verwendete Bedien-Software in Betrieb zu nehmen. Wenn jedoch eine neue Bedien-Software entwickelt werden soll, ist es von Vorteil mit einer Kommunikations-Software zu arbeiten. Diese würde jegliche Kombination zwischen alter/ neuer Bedien-Software und alter/ neuer Steuerung ermöglichen. Allerdings ist es auch möglich eine neue Bedien-Software mit den gleichen Kommunikationseigenschaften, wie die bisher verwendete Bedien-Software zu programmieren und so ohne zusätzliche Software auf die gleichen Kombinationsmöglichkeiten zu kommen. Dabei würde man sich allerdings an die alte Kommunikation binden.

Da noch keine neue Bedien-Software geplant ist und somit die aufgezählten Vorteile der Kommunikation ohne zusätzliche Software überwiegen, wird die Kommunikation über die Steuerung ausgeführt.

Kriterium	**Kommunikation über SPS**	**Kommunikation über Software und SPS**
Programmierung	einfache Struktur, übersichtlich (nur auf SPS)	SPS-Programmierung und PC-Programmierung (zusätzliche Abhängigkeit durch zweite Software)
Zusätzliche Software	Nein	Ja
Zeit für Nachrichtenübertragung	Nur Übertragungszeit (Taktzeitersparnis)	Übertragungszeit + Bearbeitungszeit
Verwendung einer neuen Bedien-Software	Protokoll muss der alten Bedien-Software entsprechen	Die Kommunikations-Software kann direkt als Grundlage für eine neue Software genutzt werden

Tab. 5: Konzeptvergleich, Eigene Darstellung

3.3 Untersuchung der Kommunikation

Die übermittelten Rohdaten sind von der Bedien-Software ASCII codiert und werden an die Steuerung als Kommandos versendet. Diese Kommandos sind spezifisch auf den noch aktuellen Controller zugeschnitten, welcher direkt auf Basis dieser Befehle die empfangenen Programmabläufe abarbeitet. Die Codierung der Kommandos durch ASCII erfordert eine Auswertung und Interpretation durch die neu einzusetzende Steuerung.

Um die Übertragungsstrukturen der Kommunikation nachzuvollziehen, muss eine Analyse des Datenverkehrs zwischen Bedien-Software und verbautem Controller durchgeführt werden. Hierfür wird die Software „Wireshark“ genutzt, welche Datenübertragungen aufzeichnet und verschiedenste Protokolle auswertet. Die Analyse des Datenverkehrs war hierbei nötig, da keinerlei Dokumentation über die Übertragung zur Verfügung stand.

3.3.1 Datenübertragung zwischen Bedien-Software und Controller

Die Kommunikation wird zunächst allgemein untersucht. Dabei wird der Übertragungsablauf ermittelt, sowie Eingänge und Ausgänge des Systems aufgelistet und beschrieben. Nachfolgend wird erläutert wie die Analyse der Kommunikation erfolgt und welche besonderen Elemente zur Übertragung genutzt werden.

1. Wireshark: Analyse Netzwerkkommunikation
2. Setpoint-Array (Array für Trainingsparameter)
3. Controller-Programme
4. Analoge und digitale Inputs/Outputs

3.3.1.1 Wireshark: Analyse Netzwerkkommunikation

Zur Analyse der Kommunikation wird nur der Datenverkehr über Port 5000 zwischen Bedien-Software und Controller, mit der Software „Wireshark“ aufgezeichnet. Der Port ist im vorhandenen Trainingsgerät fest definiert und wird zur TCP/IP Kommunikation zwischen Server und Client genutzt.

Bei der Aufzeichnung wurde die Bedien-Software zur Einstellung der Trainingsparameter genutzt. Hierbei wurde eines der Trainingsprogramme ausgewählt und die nötigen Einstellungen des Trainingsgeräts, wie das Einstellen des Maschinen-Nullpunkts und die Positionierung der mechanischen Grenzschalter für die maximal mögliche Bewegung vorgenommen.

Mit Hilfe der Aufzeichnung wurde festgestellt, dass ASCII codierte Daten an den Controller geschickt werden, welche der Programmiersprache des Controllers gleichen. Somit werden jeweils einzelne Befehle an den Controller gesendet, damit dieser den jeweiligen Befehl sofort ausführt. Als Antwort auf diese Kommandos wird der Bedien-Software eine Bestätigung von der Steuerung zurück gesendet. Bei Nachrichten ohne Rückgabewert, wird ein definiertes Zeichen als Bestätigung zurück gesendet. Bei Übertragungen von Nachrichten mit Rückgabewert werden Messwerte oder Parameter übermittelt und mit dem gleichen Zeichen als Telegrammabschluss versendet.

Für eine bessere Übersicht und ein leichteres Verständnis wird ein Ausschnitt der „Wireshark“ Aufzeichnung, während der Initialisierung nachfolgend dargestellt und beschrieben. Die Kommunikation zwischen Bedien-Software und Steuerung ist in Abbildung 8 so dargestellt, dass die rot gefärbten Befehle von der Bedien-Software an die Steuerung gesendet werden und die blau gefärbten Befehle die Antworten der Steuerung an die Bedien-Software sind. Die Befehle werden von der Steuerung durch einen Zeilenumbruch und einen folgenden Doppelpunkt bestätigt.

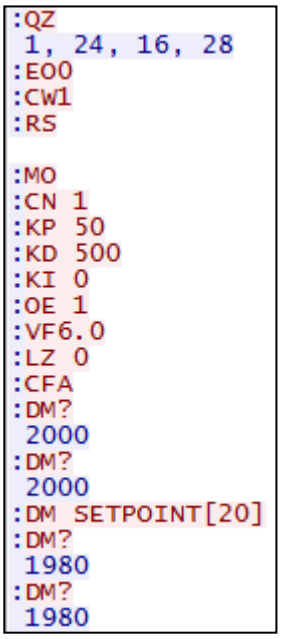


Abb. 8: Kommandoübersicht, Eigene Darstellung

Zum besseren Verständnis werden die ersten 10 Befehle kurz tabellarisch beschrieben, um den Ablauf besser nachzuvollziehen.

Befehl	**Beschreibung**
QZ	Fragt generelle Controllereigenschaften ab
EO0	Echo deaktivieren, kein Rücksenden der Daten
CW1	Setzt das MSB(Most significant Bit) auf 1
RS	Löschen des Controller-Programms (Reset)
MO	Motor ausschalten
CN 1	Konfiguration, Eingänge sind high aktiv
KP	Verstärkungsfaktor Antriebsregler
KD	Vorhaltezeit Antriebsregler
KI	Nachstellzeit Antriebsregler
VF6.0	Messwertformatierung: 6 stelliger Wert ohne Nachkommastellen

Tab. 6: Befehlsübersicht, Eigene Darstellung

Außerdem werden komplette Trainingsprogramme je nach Auswahl über die Bedien-Software in den Controller geladen, gespeichert und dort ausgeführt. Die Programme bestehen aus einem Ablauf von aneinandergereihten Befehlen und werden solange ausgeführt bis der Controller ein Reset Signal durch die Bedien-Software zugesendet bekommt. Dadurch wird das Programm vom Controller gelöscht und ein neues Programm kann geladen werden. Parallel zu den Programmen können jederzeit einzelne Befehle durch den Controller verarbeitet werden. Es können z.B. während eines Trainingsprogramms Ein- oder Ausgänge abgefragt, Messwertübertragungen angefordert oder Parameter gesetzt werden.

Die Trainingsprogramme folgen einem definierten Ablauf, der ebenfalls durch die „Wireshark“ Aufzeichnung ermittelt werden konnte. Hierbei wird untergliedert in:

a) Initialisierung
b) Einstellen des Trainingsgeräts
c) Messwertübertragung
d) Ausführen des Trainingsprogramm

a) Initialisierung

Bei der Initialisierung werden folgende Daten/ Parameter übertragen:

- Controllerinformationen:
  - Anzahl der Achsen
  - Anzahl Bytes für allgemeinen Status
  - Anzahl Bytes für Bewegungsstatus
  - Anzahl Bytes für Achsinformationen
- Befehl zum Motor ausschalten
- Reglerparameter für den Positions- und Geschwindigkeitsregelkreis
- Erstellen von Arrays für:

Arrayname	**Arraygröße**	**Bemerkung**
Status	3	Zustand, Fehler-Code, Abbruch-Code
Gravitation	360	Gravitationseinfluss für jede Position (Eine Position entspricht einem Grad)
SetPoint	20	Eingestellte Werte für das Training (genauere Beschreibung der Parameter folgt in Abschnitt 3.3.1.2)
Kommando	1	Stop-Kommando,
Messwertpuffer (Drehmoment)	100	Speicher für Drehmomentmesswerte
Messwertpuffer (Position)	100	Speicher für Positionsmesswerte
Positionsprofil	360	Aufzeichnung von Drehmoment und Geschwindigkeit für jede Position (Eine Position entspricht einem Grad)

Tab. 7, Arrays, Eigene Darstellung

b) Einstellen des Trainingsgeräts

Zum Einstellen des Trainingsgeräts werden folgende Controller-Programme durchlaufen:

- Finden der Nullposition:

  Der erste Schritt beim Einstellen des Trainingsgeräts, ist die Positionierung des Motors in Nullposition. Das Einstellen des Nullpunkts ist wichtig, um dem Motor eine fest definierte Ausgangsposition mitzuteilen. Das ist notwendig, weil für die Winkelmessung ein inkrementelles Mess-System verwendet wird, aber absolute Winkel gemessen werden sollen. Dabei muss der Motor so lange manuell eingestellt werden, bis er sich der definierten Nullposition nähert, um dann automatisch auf die genaue Position fahren zu können. Wenn die Position erreicht ist wird in der Steuerung der *Status[0]* auf 4 gesetzt, um der Bedien-Software zu signalisieren, dass der Nullpunkt erreicht wurde. Der *Status[0]* gibt den aktuellen Zustand des Trainingsgeräts an. Während das Programm aktiv ist, wird zyklisch der Zustand abgefragt, um nach Programmende den erfolgreichen Abschluss des Programms an die Bedien-Software zu signalisieren. Daraufhin wird ein Reset Befehl an den Controller gesendet und das Programm aus dem Speicher gelöscht. Anschließend sind die mechanischen Endlagenschalter, welche in Abbildung 9 blau dargestellt sind, auf die maximal mögliche Bewegungsspanne des Probanden einzustellen, um eine mögliche Verletzung zu vermeiden. Das Traningsgerät lässt sich in zwei Richtungen bewegen. Zum Einen in positiver Richtung mit dem Uhrzeigersinn (gelber Pfeil) und zum Anderen in negativer Richtung entgegen dem Uhrzeigersinn (grüner Pfeil). Die Nullposition wird in Abbildung 9, beispielhaft durch den schwarzen Pfeil dargestellt.

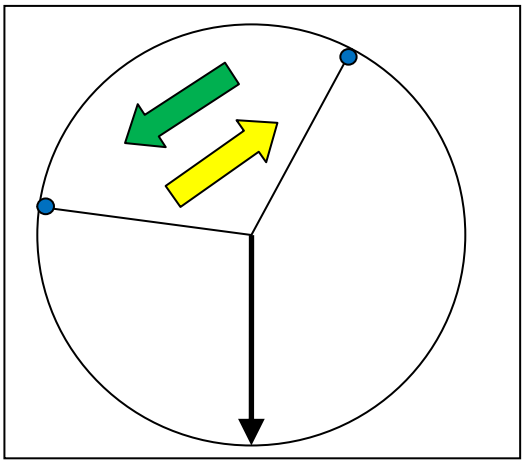


Abb. 9:Positionierung, Eigene Darstellung

- Überprüfen der Grenzwerte für die Bewegung:

Wenn sich das Trainingsgerät in Nullposition befindet und die mechanischen Endlagenschalter positioniert sind, wird das Programm zur Überprüfung der Endlagen gestartet. Dabei wird überprüft, ob der Bewegungsbereich des zu absolvierenden Trainingsprogramms innerhalb der mechanischen Endlagen liegt. Wenn die Endlagen zu nah beieinander oder zu weit auseinander liegen, ist ein Bewegen innerhalb der Grenzen nicht möglich. Die Überprüfung des Bewegungsbereichs erfolgt so lange, bis die Prüfung erfolgreich abgeschlossen wurde. Ist ein Bewegen innerhalb der mechanischen Endlagen nicht möglich müssen die Endlagen erneut positioniert werden. Das Trainingsgerät muss für die Prüfung jeweils an die mechanischen Endlagen in beiden Bewegungsrichtungen, siehe Abbildung 9 grüner und gelber Pfeil, herangefahren werden. Sobald sich der Motor den Endlagenpositionen nähert wird er abgebremst, um ein Auslösen der Schalter zu vermeiden. Wenn ein Schalter auslöst, wird das System in den Not-Aus Zustand geschaltet. Ist die Bewegungsspanne richtig abgefahren, wird das Programm vom Controller gelöscht und das Gravitationsprogramm kann geladen werden.

- Einlesen der Gravitationswerte:

Nachdem der Bewegungsbereich eingestellt wurde, werden für alle Positionen Gravitationswerte ermittelt und im Gravitation-Array gespeichert. Diese Gravitationswerte dienen dem Ausgleich von Drehmomenten, die durch zusätzliches Gewicht von Anbauteilen oder durch das Körpergewicht des Probanden entstehen. Anbauteile sind z.B. Adapter und Gurte, die für die Durchführung des Trainings benötigt werden. Die Gravitation wird dabei über das aktuell anliegende Drehmoment und einen Umrechnungsfaktor für die Drehmoment-Skalierung pro Position ermittelt. Dabei sind 360 Positionen vorhanden. Das bedeutet für jede Drehung um ein Grad, gibt es jeweils eine Position. In jeder Position wird das aktuell anliegende Drehmoment gemessen und für den Trainingsablauf kompensiert. Wenn die Gravitationswerte ermittelt wurden, wird das Programm vom Controller gelöscht und das eigentliche Trainingsprogramm kann geladen werden.

c) Messwertübertragung

Es gibt 2 Arrays, welche für die Zwischenspeicherung und Übertragung von Messwerten benötigt werden. Beide Arrays besitzen eine Speichergröße von jeweils 100 Messwerten und werden als Ringpuffer deklariert.

- PDATA[..]: Array für Positionsmesswerte
- TDATA[..]: Array für Drehmomentmesswerte

Die Messdaten müssen ASCII codiert in festem Format versendet werden. Dabei wird ein Byte für das Vorzeichen und ein weiteres Byte für das Trennzeichen benötigt. Die restlichen sechs Byte stehen für die Messwerte zur Verfügung. Ein Auffüllen der Länge ist immer notwendig, d.h. wenn eine Null versendet wird, müssen alle sechs Stellen mit Nullen gefüllt werden.

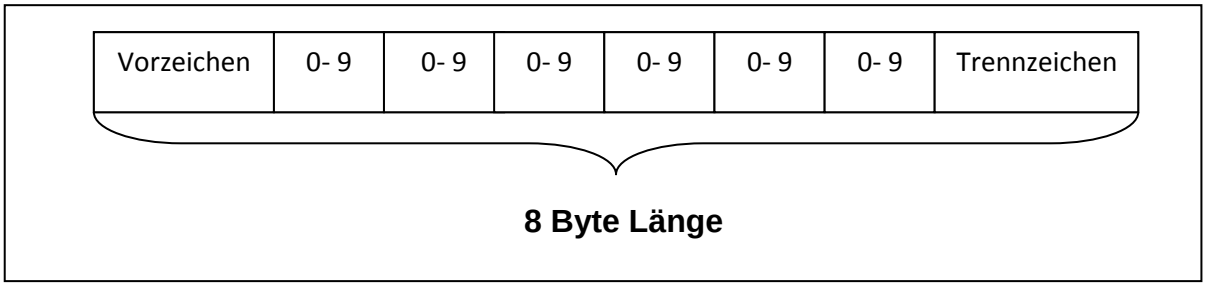


Abb. 10: Messwertformat, Eigene Darstellung

Die Daten in den beiden Ringpuffern sind vom Typ *Integer* auf der Steuerung gespeichert .Für das Versenden müssen die Messwerte in das benötigte Messwertformat, nach Abbildung 10 konvertiert werden. Beim Versenden der Messwerte wird nicht das ganze Array versendet, sondern immer nur ein definierter Teil, weil die Messwerte sofort versendet werden müssen sobald ein Befehl von der Bedien-Software empfangen wird. Dafür wird ein Pointer für den Puffer benötigt, welcher die aktuelle Position im Array wiedergibt, an der neue Messwerte gespeichert werden können. Zur Übersicht ist die Struktur des Ringpuffers im nachfolgenden Schema dargestellt.

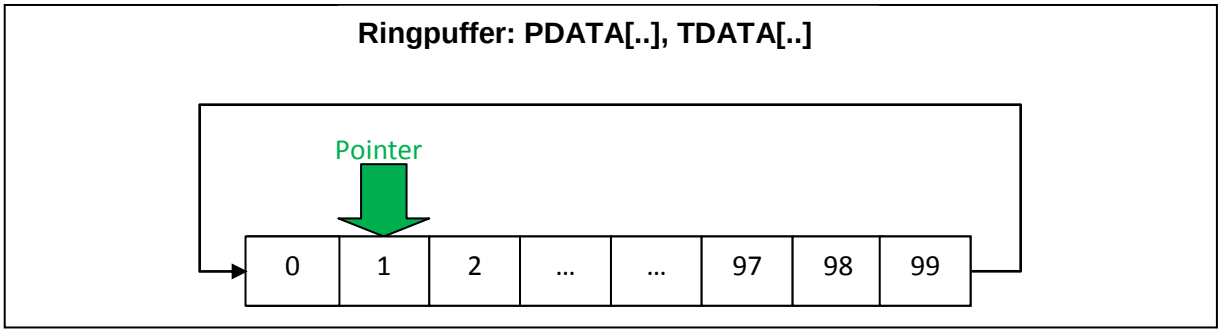


Abb. 11: Schema Ringpuffer, Eigene Darstellung

Durch den Pointer wird der Software übermittelt bis zu welcher Position Messwerte abgerufen werden können. Daraufhin fordert die Software alle Messwerte vom zuletzt gelesen Messwert bis zur Pointer-Position an. Durch das folgende Beispiel soll die Kommunikation veranschaulicht werden.

Kommandobeschreibung

- MG _RD:
  Liefert den Wert des aktuellen Pointers zurück, bis zu welchem Messwerte eingelesen wurden.
- QU TDATA[]/ PDATA[], (Startposition, Endposition, Trennsymbol: Komma):
  Fordert einen Upload der Arrays mit den angeforderten Positionen.

In der ersten Messwertübertragung werden alle Messwerte von Position 89 bis 97 der beiden Arrays an die Software gesendet. Die Grenze der Arrays wird nicht überschritten.

```
MG _RD
000098
QU TDATA[],89,97,1
000656,-000656,-000656,-000656,-000656,-000640,-000624,-000608,-000608
QU PDATA[],89,97,1
000000, 000000, 000000, 000000, 000000, 000000, 000000, 000000, 000000
```

Abb. 12: Messwertübertragung Beispiel 1, Wireshark

Im zweiten Beispiel werden alle Daten von Position 98 bis 5 gesendet. Hierbei wird die obere Grenze des Arrays von 99 überschritten und es werden zweimal Messwerte hintereinander angefordert. Einmal alle restlichen Messwerte bis zum Arrayende und danach alle Elemente bis zum angegebenen Ende durch den Pointer.

```
MG _RD
000006
QU TDATA[],98,99,1
000592,-000592
QU PDATA[],98,99,1
000000, 000000
QU TDATA[],0,5,1
000592,-000608,-000592,-000592,-000592,-000576
QU PDATA[],0,5,1
000000, 000000, 000000, 000000, 000000, 000000
```

Abb. 13: Messwertübertragung Beispiel 2, Wireshark

3.3.1.2 Setpoint-Array

Das Setpoint-Array wird genutzt, um einstellbare Parameter, wie Geschwindigkeiten, Drehmomente, etc. von der Bedien-Software zur Steuerung zu senden. Es wird während der Laufzeit von der Bedien-Software aktualisiert und Änderungen werden an die Steuerung versendet.

Nummer	Bezeichnung	Bemerkung
0	RangeLimit 1	Position für maximale Bewegung (positive Endlage)
1	RangeLimit 2	Position für maximale Bewegung (negative Endlage)
2	Speed 1	Geschwindigkeit in positiver Bewegungsrichtung
3	Speed 2	Geschwindigkeit in negativer Bewegungsrichtung
4	Torque 1	Drehmoment in positiver Bewegungsrichtung
5	Torque 2	Drehmoment in negativer Bewegungsrichtung
6	StopMove	Stopp-Position
7	Mode	Parameter für Modus Einstellungen
7.1	CTYPE	Controller Typ
7.2	STYPE	System Typ
7.3	SRATE	Abtastzeit
7.4	RAMP	Beschleunigungsrampe (schnell, normal, langsam)
7.5	GAIN	Stromverstärkung
7.6	MODE	Bewegungsmodus (klassisch, ballistisch)
9	MaxTorque	Maximal zulässiges Drehmoment
10	Inertia	Erwartete Trägheit des Adapters
11	Pgain	Verstärkung Controller
12	Sgain	Verstärkung zum Ausgleich des Drehmoment
13	Rgain	Verstärkung bei aktuellem Drehmoment
14	Smoothing	Unabhängige Filter-Zeitkonstante
15	Memory	Zuletzt genutztes Drehmoment
16	Acceleration	Beschleunigungsfaktor
17	Deceleration	Abbremsfaktor
18	Torque/ Speed Adjustment	Drehmoment/ Geschwindigkeits Justierung je nach Trainingsprogramm
19	Torque Offset	Drehmomentwert beim Setzen der Nullposition

Tab. 8: Setpoint-Array, Eigene Darstellung

3.3.1.3 Controller-Programme

Bei den Controller-Programmen handelt es sich um einen Ablauf von Befehlen, die komplett in die Steuerung geladen, gespeichert und ausgeführt werden, bis das Programm vom Controller gelöscht wird. Zu diesen Programmen zählen unter anderem die kompletten Trainingsprogramme, welche so lange ausgeführt werden, bis das Trainingsziel erreicht wird oder es zu einem Abbruch kommt. Desweiteren werden während der Initialisierung mehrere Programme benötigt, um das Trainingsgerät auf den Probanden einzustellen. Diese Programme sind:

1. Finden der Nullposition
2. Überprüfen der Grenzwerte für die Bewegung
3. Einlesen der Gravitationswerte

Während die Programme ausgeführt werden, können parallel Messwerte oder Parameter übertragen werden. Bei der Kommunikation werden diese Programme über ein Download-Kommando an den Controller versendet. Die Programme liegen als Textdateien der Bedien-Software zur Verfügung und werden byteweise als ASCII-Code über TCP versendet. Die Textdateien für die Controller-Programme werden mit der Bedien-Software installiert und müssen jedes Mal wenn sie benötigt werden, neu in den Controller geladen werden. Dadurch wird bei jedem Aufruf eines Programms Übertragungszeit benötigt und das komplette Programm ist auf dem Übertragungsweg sichtbar. Hierbei könnten die Programme einfach von einer externen Quelle ausgelesen werden. Wenn sich die Programme hingegen permanent auf einer Steuerung befinden, wird die Übertragungszeit reduziert, ein Fremdzugriff durch Dritte wird vermieden und eine Manipulation der Textdateien kann verhindert werden. Eine Verbesserung der Datensicherheit wird also durch die Implementierung einer neuen Steuerung gewährleistet. Weitere Informationen zum Aufbau und Ablauf der Programme werden in Kapitel 3.2.2 gegeben.

3.3.1.4 Analoge/ Digitale Eingänge

Folgende digitale Eingänge werden vom Controller eingelesen und in ein Eingangs-Array geschrieben, das komplett an die Software gesendet wird.

Array-Position IN[..]	Bezeichnung
1	Servo-Fehler
2	Grundstellungssensor
3	Verriegelungssensor
4	Not-Aus
5	Taster Handbetrieb Bewegungsrichtung 1
6	Taster Handbetrieb Bewegungsrichtung 2

Tab. 9: Digitale Eingänge, Eigene Darstellung

Bei den Eingängen zur Messwertaufnahme wird der Eingang des Drehmoments analog über +/- 10V gemessen und die Winkelposition des Motors wird über einen Encoder ermittelt.

Das Drehmoment wird im Bereich von +10V bis -10V gemessen und an die Steuerung als *Integer* Wert im Bereich von +32767 bis -32768 übergeben. Bei einem Messbereich von 0 bis 750Nm entspricht ein Volt Spannung, 75Nm Drehmoment. Das Vorzeichen des Messwerts gibt die Richtung an in die das Drehmoment wirkt. Die Umrechnung des analogen Messwerts erfolgt in der Steuerung über einen Skalierungsfaktor.

Die Position wird über einen Encoder mit einer Auflösung von 4000 Inkrementen ermittelt. Da sich ein Getriebe auf der Motorwelle, hinter dem Encoder befindet wird die Auflösung der Position entsprechend der Übersetzung des Getriebes im Verhältnis 1:59 erhöht. Das sind 59* 4000, also 236000 Inkremente im kompletten Umfang. Das entspricht 655 Positionen pro Grad Drehwinkel.

Eine dritte Größe, die Geschwindigkeit, wird zudem über die zeitliche Veränderung der Position berechnet. Die Position differenziert über die Zeit, entspricht der Geschwindigkeit.

3.3.2 Eigenschaften der Controller-Programme

Zur Implementierung der Controller-Programme müssen Umrechnungen und Skalierungen von Messwerten berücksichtigt, die Struktur der Controller-Programme ermittelt und ein Ablauf für die Trainingsprogramme definiert werden.

3.3.2.1 Umrechnungen und Skalierungen

Beim Einlesen der Messwerte für Position und Drehmoment müssen verschiedene Skalierungen und Umrechnungen beachtet werden. Je nach Auswahl des Trainingsprogramms müssen verschiedene Skalierungen und Auflösungen eingestellt werden. Die verschiedenen Trainingsprogramme können in zwei Bewegungsarten ausgeführt werden. Eine Möglichkeit ist die Bewegung in linearer Richtung z.B. Training mittels Beinpresse, die zweite Möglichkeit ist das Training in einer Kreisbewegung, z.B. Arm- oder Rückentraining. Die Bewegungsarten besitzen unterschiedliche Messgrößen und somit auch verschiedene Auflösungen und Formatierungen.

Bewegungsart	**Wert**	**Wertebereich**	**Auflösung**	**ASCII-Format**
Linearbewegung	Distanz	0 – 1000mm	0,1mm	0 – 10000
	Geschwindigkeit	0 – 1000mm/s	0,1mm/s	0 – 10000
	Kraft	0 – 4000N	0,1N	0 – 40000
Kreisbewegung	Winkelposition	0 – 360°	0,1°	0 – 3600
	Geschwindigkeit	0 – 500°/s	0,1°/s	0 – 5000
	Drehmoment	0 – 750Nm	0,1Nm	0 – 7500

Tab. 10: Messwertauflösung, Eigene Darstellung

Zusätzlich konnten mit Hilfe der Controller-Programme Skalierungswerte für Position, Geschwindigkeit, Drehmoment und Beschleunigung ermittelt werden. Die Skalierungswerte sind fest definierte Werte, die je nach Trainingsadapter genutzt werden. Die Skalierungswerte können sich aber auch dynamisch ändern und werden dann mittels Setpoint-Array übertragen.

Zur Übersicht werden die Skalierungsvariablen mit ihren zugehörigen Skalierungswerten nachfolgend dargestellt.

Parameter	**Variable**	**Skalierungswert**	**Beschreibung**
Position	G_SCALE	655	
Geschwindigkeit	P_SCALE	65,5 (Kreisbewegung) 31,3 (Linearbewegung)	Je nach Modus wird verschieden skaliert.
Drehmoment	M_SCALE	10/(Setpoint[9]/Verstärkung)	Dynamisch berechnet
Beschleunigung	AC_SCALE DC_SCALE	1 bei Ramp= 0 0,5 bei Ramp= 1 0,25 bei Ramp= 2	Je nach Wert von Ramp wird stärker oder schwächer beschleunigt und gebremst. (100%, 50%, 25%)

Tab. 11: Skalierungswerte, Eigene Darstellung

3.3.2.2 Ablauf Controller-Programme

Alle Controller-Programme besitzen einen ähnlichen Ablauf. Sie unterscheiden sich lediglich in der Kombination aus Belastungsart und Bewegungsrichtung.

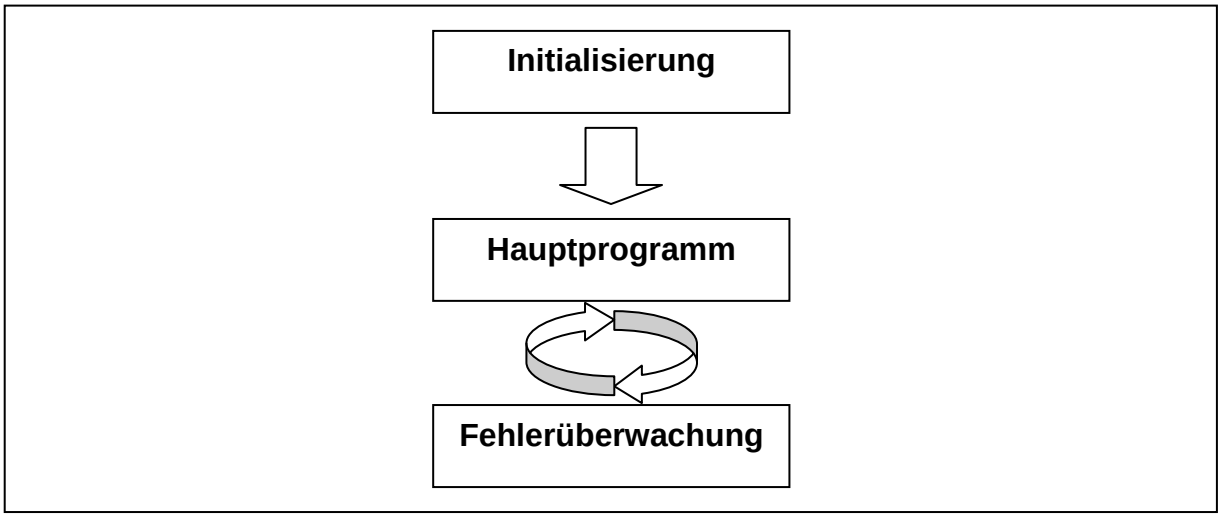


Abb. 14: Ablauf Controller-Programme, Eigene Darstellung

Die Initialisierung erfolgt bei allen Programmen gleich, dabei werden der Status des Trainingsgeräts, die Zeiten zur Fehlerüberwachung und das Kommandobyte für einen Abbruch zurückgesetzt. Sobald alle nötigen Parameter zurückgesetzt sind, können neue Werte in die Steuerung geladen werden. Hierbei werden zunächst in jedem Controller-Programm die Parameter des Setpoint-Array eingelesen und in der Steuerung überschrieben. Nachdem alle Parameter durch das Programm eingelesen wurden, ist die Initialisierung abgeschlossen und das Trainingsprogramm kann durchgeführt werden.

Im Hauptprogramm wird je nach ausgewähltem Trainingsprogramm die Belastungsart und die Bewegungsrichtung definiert. Die Bewegung beginnt und wird bis zur jeweiligen maximalen Bewegungsposition durchgeführt. Sobald sich die aktuelle Position des Trainingsgeräts dem Umkehrpunkt der Bewegung nähert, wird die Geschwindigkeit bis auf Null reduziert. Nach Erreichen des Umkehrpunkts wird die Bewegungsrichtung umgekehrt und eine Änderung der Belastungsart kann durchgeführt werden. Die Belastungsarten sind als Unterprogramme im Hauptprogramm gekapselt und können abwechselnd aufgerufen werden. Bei zwei verschiedenen Belastungsarten, wird somit immer zwischen zwei Unterprogrammen im Hauptprogramm gewechselt. Während der Proband im Trainingsbetrieb ist wird permanent eine Fehlerüberwachung durchgeführt.

In allen Fehlerfällen wird das Trainingsprogramm abgebrochen und der jeweilige Fehler über einen definierten Fehler-Code an die Bedien-Software gesendet. Folgende Kriterien werden in der Fehlerüberwachung kontrolliert:

- Überwachung des Drehmoments:
  Das Drehmoment des Trainingsgeräts wird auf seinen maximal eingestellten Grenzwert überwacht. Sollte der Grenzwert überschritten werden, kommt es zum Fehler. Der Grenzwert wird im Setpoint-Array definiert, siehe Kapitel 3.3.1.2.

- Überwachung des Antriebs:
  Der digitale Eingang 1 kommt vom Antriebsregler. Dieser Eingang übermittelt einen Servo-Fehler. Wenn ein high Signal an die Steuerung gesendet wird, liegt ein Servo-Fehler am Antrieb vor.

- Watchdog-Überwachung:
  Zur Vermeidung einer Zeitüberschreitung in der Software wird eine Zeitüberwachung über einen Watchdog-Timer durchgeführt. In jedem Zyklus des Controller-Programms wird der Watchdog-Timer auf die Systemzeit des Controllers zurückgesetzt. Daraufhin wird überprüft, ob die abgelaufene Zeit, seit dem Zurücksetzen des Timers, einen Wert von 4ms nicht überschreitet.

3.3.2.3 Zustandsdiagramm für Programmablauf

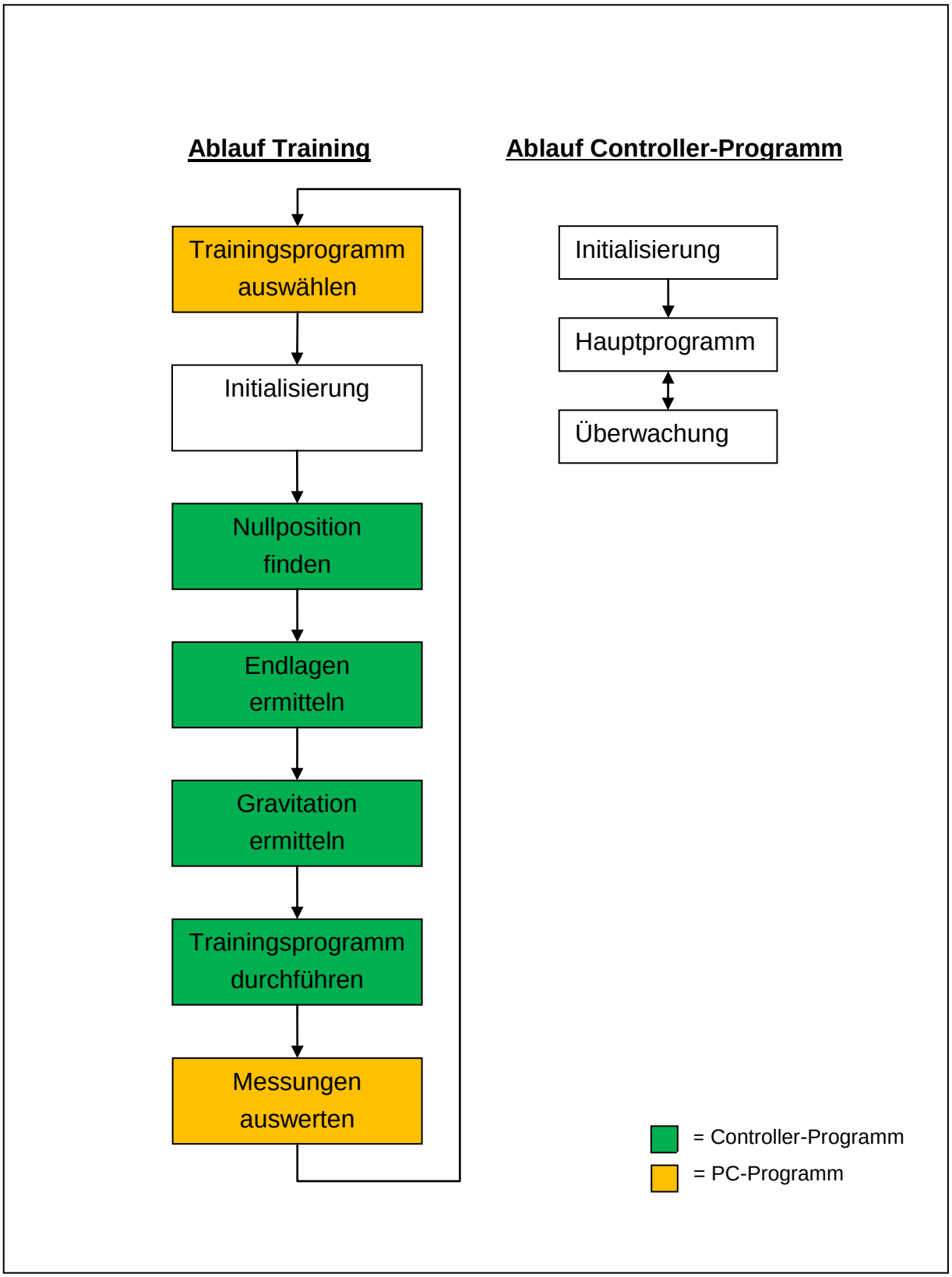


Abb. 15: Zustandsdiagramm, Eigene Darstellung

3.4 Implementierungen in CoDeSys

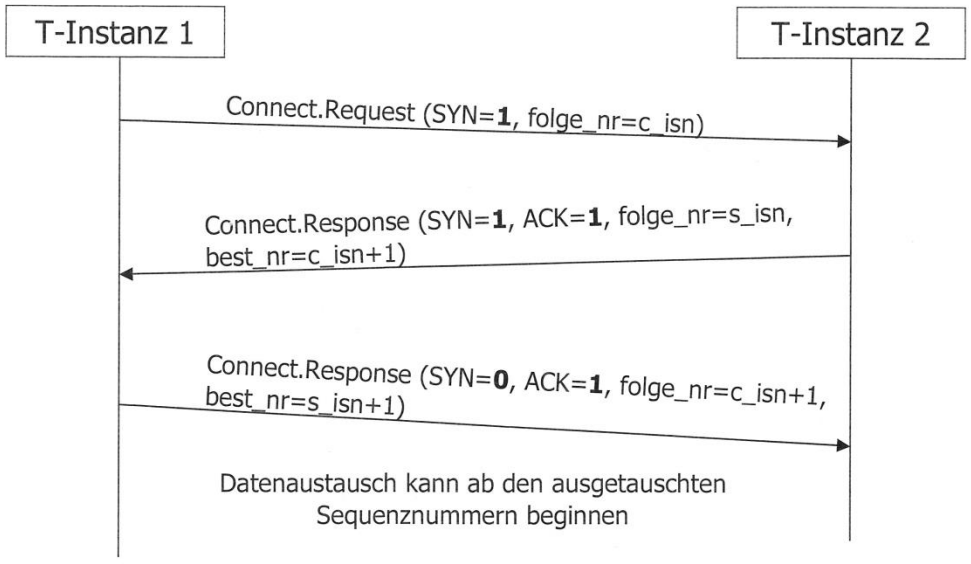
Die ermittelten Kommunikationseigenschaften müssen mittels CoDeSys als struktuierter Text programmiert werden. Die Programmierung erfolgt in folgenden Schritten:

1. TCP-Verbindung aufbauen
2. Datenempfang, Auswertung und Versenden
3. Implementierung der Controller-Programme

3.4.1 TCP-Verbindung aufbauen

Zur Herstellung der Verbindung über das TCP-Protokoll, muss die neue Steuerung als Master für die Kommunikation über TCP/IP eingerichtet werden. Dafür wird eine Standard Bibliothek der Firma 3S verwendet, welche eine Funktion zur Verbindungsherstellung zur Verfügung stellt. Der Verbindungsaufbau basiert auf dem folgendem Verbindungsablauf (siehe Abbildung 16).

Zunächst wird der TCP-Server mit der IP-Adresse der Steuerung instanziert. Nachdem der Server erstellt wurde, wird ihm mitgeteilt auf eingehende Verbindungssignale des Ports 5000 zu reagieren. Wenn der TCP-Client sich mit dem Server verbindet, wird eine Bestätigung an den Client zurück gesendet und die IP-Adresse, sowie der Port als Socket im Server gebunden und gespeichert. Zum Abschluss sendet der Client eine letzte Bestätigung an den Server um die Verbindung zu bestätigen. Jetzt wird die Verbindung so lange aufrecht erhalten, bis der Socket wieder geschlossen wird.



c_isn = Initial Sequence Number des Clients (Instanz 1)
s_isn = Initial Sequence Number des Servers (Instanz 2)

Abb. 16: TCP-Verbindungsschema, [6] S. 205

3.4.2 Datenempfang, Auswertung und Versenden

Nachdem die Verbindung zwischen Steuerung und Bedien-Software besteht, kann der Datenempfang beginnen. Sobald die Steuerung ein TCP-Telegramm empfängt wird die Nachricht in einen Zwischenspeicher byteweise abgesichert. Der Speicher ist auf 512 Byte festgelegt, damit eine definierte maximal Anzahl an Messwerten pro Telegramm übertragen werden kann. Mit jedem SPS-Zyklus wird die Länge der eingehenden Nachrichten überprüft. Ist die Länge der Nachricht größer als Null, muss ein neues Telegramm eingegangen sein. Die empfangene Nachricht wird im Zwischenspeicher so lang gehalten bis eine Antwort zurück gesendet wird. Danach wird der Empfangsspeicher wieder geleert und die nächste Nachricht kann empfangen werden.

Die im Empfangsspeicher liegende Nachricht ist anschließend von der Steuerung auszuwerten. Für die Auswertung der Befehle werden die ASCII codierten Bytes der Nachricht in einen String umgewandelt. Die Zeichenketten sind somit übersichtlicher auszuwerten, als wenn man sie byteweise überprüfen würde. Beispielhaft ist die Konvertierung vom Array in einen String nachfolgend in Abbildung 17 dargestellt. Die Bytes sind in dezimaler Schreibweise dargestellt.

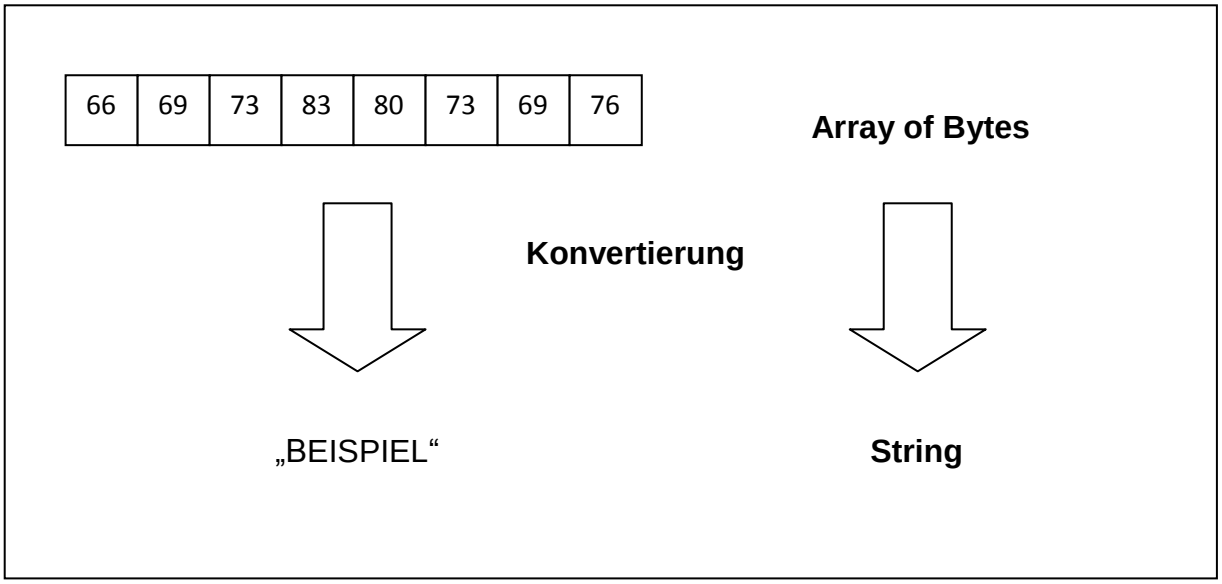


Abb. 17: Array zu String Konvertierung, Eigene Darstellung

Die Daten stehen als Strings zur Verfügung und müssen im nächsten Schritt interpretiert und in der Steuerung verarbeitet werden. Hier zeigt sich der Vorteil mit Strings zu arbeiten, denn eine Auswertung der eingehenden Befehle ist auf diese Weise übersichtlich und verständlich für den Programmierer nachzuvollziehen. Die eingehenden Nachrichten werden unterteilt in:

1. Befehle/Kommandos
2. Abruf der Messdaten (Drehmoment/ Position)
3. Ausnahmen

3.4.2.1 Befehle/ Kommandos

Zur Auswertung der Befehle ist eine Funktion in CoDeSys implementiert worden, welche alle in der Kommunikationsanalyse ermittelten Befehle erkennt und die entsprechenden Antworten für die Übertragung bereitstellt. Die Befehle unterteilen sich hierbei in drei verschiedene Kategorien. Es gibt Befehle die von der Steuerung nur bestätigt werden müssen, Befehle die neue Parameter an die Steuerung übermitteln und Befehle die Parameter in der Steuerung abfragen.

Befehlskategorie	**Beispiele**
Nachricht mit Bestätigung	• Motor ausschalten • Motor einschalten • Abbruch • Reset
Parameter an SPS senden	• Verstärkung • Nachstellzeit • Vorhaltezeit • Setpoint-Werte
Parameter von SPS abfragen	• Status • Pointerposition • Digitale Eingänge

Tab. 12: Auswertung Befehle, Eigene Darstellung

Zur Verdeutlichung und besseren Beschreibung der einzelnen Befehle, wird jeweils ein Beispiel näher erläutert und dargestellt.

a) Befehle nur mit Bestätigung als Rückmeldung

Bei diesen Befehlen wird von der Steuerung nur eine Bestätigung an die Bedien-Software zurück gesendet. Dabei wird je nach Befehl eine entsprechende Aktion in der Steuerung ausgeführt. Im Vergleich zum alten Controller sind die Befehle analog dazu in die neue Steuerung zu implementieren. So ist z.B. bei einem „Motor ausschalten“ Kommando, der Motor von der Steuerung auch auszuschalten.

Einige Befehle sind leicht in die neue Steuerung zu übernehmen. Das sind solche Befehle, die lediglich einen boolschen Zustand der Steuerung ändern, wie z.B das Ein- oder Ausschalten des Motors. Andere Befehle hingegen sind komplexer zu implementieren oder brauchen nicht mehr durch die neue Steuerung bearbeitet werden. Ein Beispiel für einen Befehl der von der neuen Steuerung nicht berücksichtigt werden muss, ist das Einrichten der Arrays und das Einlesen der Trainingsabläufe. Die Trainingsabläufe, als auch die Arrays werden im Speicher in der neuen Steuerung hinterlegt und müssen nicht zur Laufzeit erstellt oder eingelesen werden (siehe Kapitel 3.3.1.3). Der Reset Befehl hingegen ist schwieriger in die Steuerung einzubinden. Zum Einen wird mit dem Befehl das jeweilige Programm beendet, jedoch wird ein Löschen der Programme von der Steuerung nicht mehr benötigt, weil die Programme komplett im Speicher der Steuerung vorhanden sind.

Befehl	**Beschreibung**	**Bemerkung**
MO	Motor ausschalten	Boolsche Variable, dadurch einfach zu implementieren
RS	Reset, Löschen des Controller Programms	Speicherung der Controller-Programme in der Steuerung, jedoch muss bei Reset trotzdem das Programm gestoppt werden
DM Setpoint[20]	Einrichten eines Arrays mit Bezeichnung „Setpoint“	Entfällt, weil die Arrays im neuen Steuerungsprogramm direkt erstellt werden

Tab. 13: Übersicht Befehle mit Bestätigung, Eigene Darstellung

b) Befehle die Parameter an die SPS senden

Zur Übertragung von Parametern, wie Verstärkungsfaktoren für Regler oder Trainingseinstellungen aus dem Setpoint-Array, werden Befehle zur Übermittlung verwendet. Diese Befehle setzen sich aus dem Variablennamen der zu sendenden Parameter und dem Wert des Parameters zusammen. Wenn ein Parameter an die Steuerung geschickt wird, muss die Steuerung zunächst den empfangenen Befehl der richtigen Variablen im Speicher zuordnen. Danach wird der zu übermittelnde Wert des Parameters vom restlichen Befehl getrennt. Der neue Parameterwert ist immer noch ein String in ASCII Codierung. Dieser String muss in einen entsprechenden Datentyp umgewandelt werden. Je nach Variablen unterscheidet sich der Datentyp zwischen *Integer* und *Real*. Als Beispiel wird nachfolgend der Setpoint-Wert für die Geschwindigkeit in positiver Bewegungsrichtung auf 500 gesetzt, was einer realen Geschwindigkeit von 50°/s entspricht.

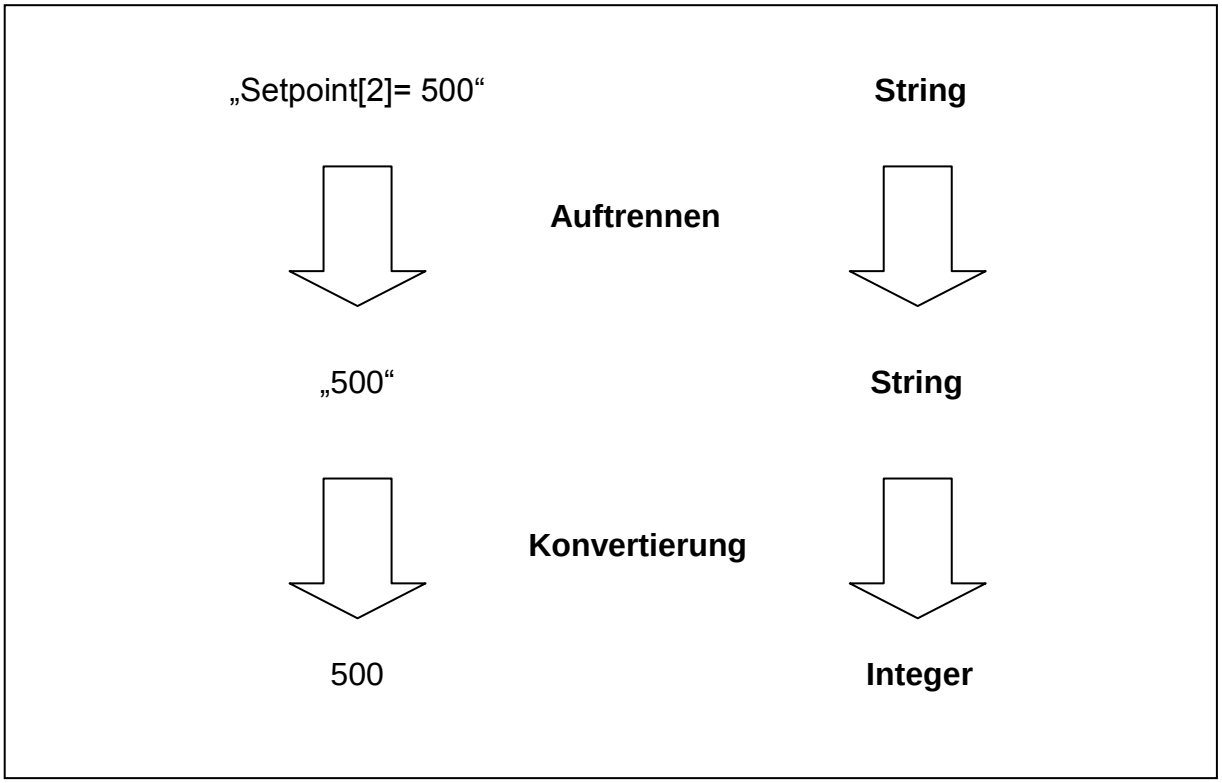


Abb. 18: Konvertierung Parameter an SPS, Eigene Darstellung

c) Befehle die Parameter von der SPS anfordern

Befehle von der Bedien-Software, welche Parameter von der Steuerung anfordern sind z.B.: Statusabfragen oder Abfragen von digitalen Eingängen. Die empfangenen Befehle müssen von der Steuerung ausgewertet werden. Dabei gilt es zu bestimmen, welche Parameter von der Bedien-Software angefordert werden. Die Steuerung konvertiert dann die Parameterwerte, die in der Steuerung als *Integer* oder *Real* abgespeichert sind, zu Strings. Diese Strings müssen im letzten Schritt in einzelne ASCII Zeichen aufgeteilt und in einem Array für das Versenden von Nachrichten abgespeichert werden. Das Aufspalten der Strings geschieht über zeichenweise Konvertierung des Strings in ASCII-Code. Über dieses Array können die Messwerte im definierten Format an die Steuerung gesendet werden. Zur Veranschaulichung wird hier ein Beispiel zur Abfrage des *Status[0]*, welcher den Zustand des Trainingsgeräts angibt, abgebildet.

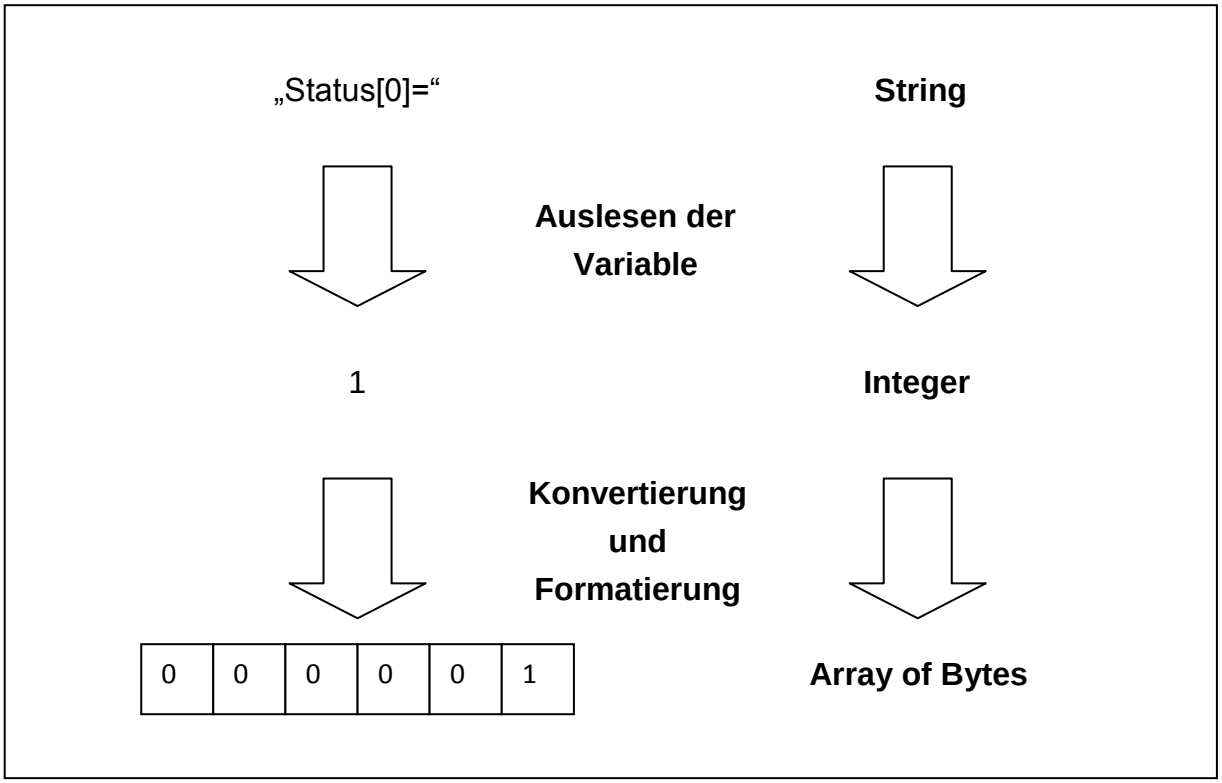


Abb. 19: Konvertierung Parameter von SPS, Eigene Darstellung

3.4.2.2 Abruf der Messdaten (Drehmoment/ Position)

Bei der Übertragung der Messdaten muss zunächst durch die Analyse des Befehls ermittelt werden, welcher Messwert übersendet werden soll. Die beiden Messwert-Arrays für Position und Drehmoment stehen hier zur Auswahl. Wie in Kapitel 3.3 beschrieben, werden die Messwerte in einem Ringpuffer gespeichert. Hierbei ist bei der Implementierung zu beachten, dass ein Lese- und ein Schreib-Pointer benötigt werden. Beim Einlesen der Messwerte wird der Schreib-Pointer genutzt, um die aktuell zu beschreibende Position im Array anzugeben. Mit jedem neu gelesenen Messwert wird der Pointer um eine Stelle weiter verschoben. Die Bedien-Software fragt die Position des Schreib-Pointers vor einem Abruf der Messdaten ab und begrenzt somit die Menge an zu übertragenden Daten. Der Lese-Pointer gibt hingegen die Stelle im Array an, von der Messwerte gesendet werden können. Der Lese-Pointer wird nach dem Versenden der Messwerte um die Anzahl der versendeten Messwerte im Ringpuffer verschoben.

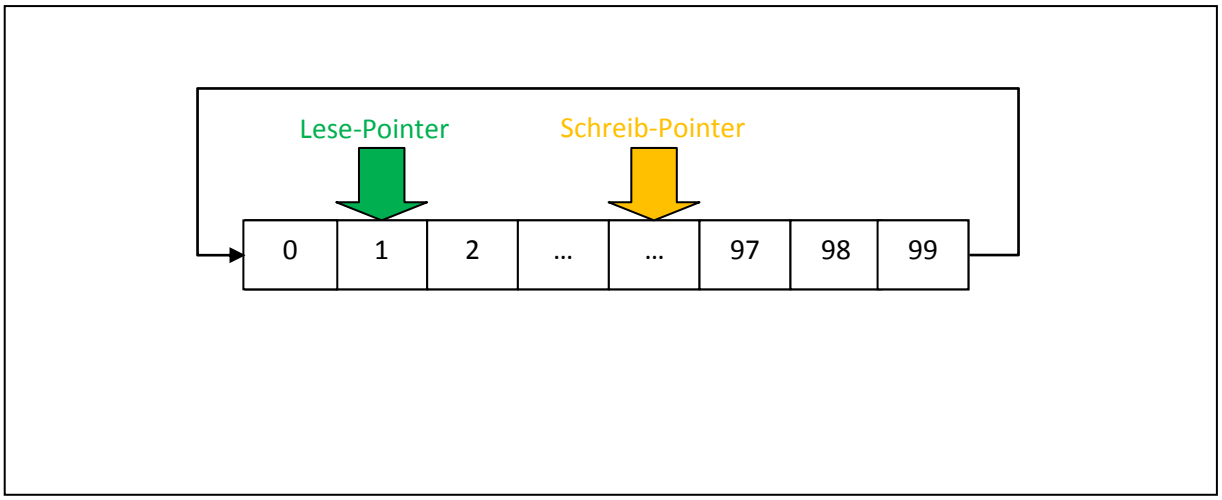


Abb. 20: Messwertspeicher CoDeSys, Eigene Darstellung

Die Messwerte im Zwischenspeicher sind vom Datentyp *Integer* müssen aber, wie alle anderen Daten auch, als ASCII-Zeichen versendet werden. Hierbei ist das definierte Format bei der Datenübertragung zu beachten. Die Messdaten sind mit Vorzeichen, sechsstellig und mit einem Trennzeichen (Komma) zwischen den einzelnen Messwerten zu versenden. Im Folgenden sind die Messwerte von *Integer* in dieses definierte Format zu konvertieren. Ähnlich dem Versenden der Parameter müssen die Daten mit Nullen vor dem Messwert aufgefüllt werden. Jedoch ist die Anzahl der Nullen abhängig vom jeweiligen Messwert und entspricht hier keiner definierten Anzahl.

Zur Konvertierung der Messwerte werden diese nacheinander und einzeln aus dem jeweiligen Ringpuffer geholt. Für das erste Element ist das die Position des Lese-Pointers. Nach dem Einlesen wird die Anzahl der Vorkommastellen des Messwerts bestimmt. Die Anzahl der Vorkommastellen wird mit der Anzahl der Nachkommastellen addiert und ergibt die Länge des Messwerts. Alle Messwerte besitzen eine Auflösung von einem Zehntel und besitzen daher genau eine Nachkommastelle. Die Länge des Messwerts ist nötig für die Bestimmung der Nullen vor dem Messwert und für die Aufspaltung der einzelnen Stellen in ASCII-Code.

Wenn die Länge ermittelt ist, wird zunächst das Vorzeichen des Messwerts bestimmt und an die erste Position geschrieben. Ist der Messwert positiv wird ein Freizeichen übertragen, ist der Messwert hingegen negativ, ein Minus. Danach wird der Messwert in seine einzelnen Stellenwerte zerlegt und mit der nötigen Anzahl an Vornullen gefüllt. Hierbei wird die Anzahl an Vornullen über die Differenz der maximalen Messwertlänge (6) und der Länge des aktuellen Messwerts berechnet. Der sechsstellige Messwert wird dann von *Integer* in ASCII konvertiert und in den Speicherbereich nach dem Vorzeichen gelegt. Zum Schluss wird das Trennzeichen in die letzte Position geschrieben und der Lese-Pointer eine Position weiter im Ringpuffer verschoben. Der nächste Messwert kann konvertiert werden, bis der Lese-Pointer die Position des Schreib-Pointers subtrahiert um 1 erreicht.

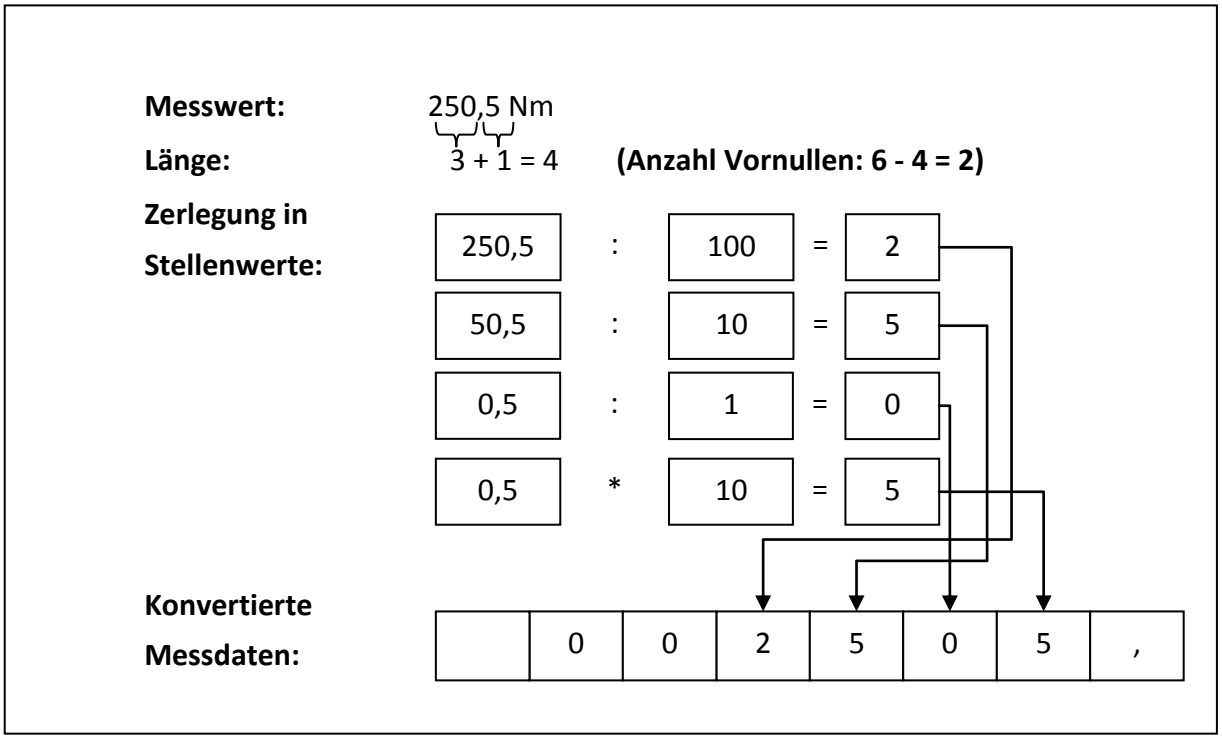


Abb. 21: Konvertierung der Messdaten, Eigene Darstellung

3.4.2.3 Ausnahmen

Bei der Datenübertragung werden nicht nur Zeichen übermittelt, die sich in Strings umwandeln lassen. Zu diesen Zeichen gehören alle ASCII-Zeichen die einem Dezimalwert kleiner gleich 31 entsprechen. Sie lassen sich nicht in einen sinnvollen String konvertieren, weil sie den Eingabetasten wie Space oder Enter entsprechen. Solche verwendeten ASCII-Zeichen mit ihrer Bedeutung für die Kommunikation sind in Tabelle 14 dargestellt.

Dezimalwert (ASCII)	**Symbol**	**Beschreibung**	**Verwendung**
10	LF	Line Feed	Bestätigung des Eingangs von einem Anteil der eingehenden Messwerte. Nur wenn Messwerte auf mehrere Telegramme aufgeteilt werden müssen.
13	CR	Carriage Return	Endekennung von Befehlen von der Bedien-Software
26	SUB	Substitute	Endekennung von Controller-Programmen

Tab.14: Ausnahmen bei der Kommunikation, Eigene Darstellung

Damit diese Befehle ebenfalls korrekt von der Steuerung ausgewertet werden können, müssen die Positionen der Zeichen in der Nachricht bekannt sein. Die Endekennung der Controller-Programme sowie die Bestätigung eingegangener Messwerte werden jeweils als einzelne Zeichen versendet und stehen damit an erster Stelle in der Nachricht. Bei diesen beiden Befehlen muss also lediglich das erste empfangene Zeichen der Nachrichten überprüft werden. Hierfür wird bei allen eingehenden Nachrichten von der Bedien-Software das erste Byte mit dem jeweiligen Dezimalwert des ASCII-Zeichens abgeglichen. Wird eines dieser Zeichen erkannt, muss die empfangene Nachricht nicht mehr in einen String konvertiert werden und eine direkte Bestätigung wird an die Bedien-Software zurück gesendet. Bei der Endekennung von Befehlen, ist es notwendig über die Telegrammlänge auf die Position des letzten empfangenen Bytes zu schauen. Ist das Byte am Ende des Befehls gesetzt, so wurde der Befehl komplett übertragen und kann in einen String konvertiert und weiter verarbeitet werden.

3.4.3 Implementierung der Controller-Programme

Im letzten Schritt sind die Controller-Programme zu implementieren. Eine komplette Implementierung aller Programme war jedoch aus zeitlichen Gründen nicht möglich und wurde deshalb auf das erste Controller-Programm zur Einstellung des Maschinen-Nullpunkts begrenzt.

In der Bedien-Software wird das Auswählen des Trainings und das Einstellen des Trainingsgeräts über eine graphische Oberfläche gesteuert. Diese Oberfläche wechselt je nach Status des Trainingsgeräts. Zur besseren Übersicht und weil sie für die Implementierung der Controller-Programme von Bedeutung ist, wird die Angabe des Status in Tabelle 15 dargestellt.

Status[…]	**Bezeichnung**	**Wert**	**Beschreibung**
0	Zustand/ Status	0	Standby
		1	Keine Bewegung
		2	Drehmoment-Limit
		3	Stopp-Position
		4	Maschinen-Nullpunkt
		5	In Bewegung
1	Fehler-Code	0	Kein Fehler
		1	Limit überschritten
		2	Servo-Fehler
		4	Übertragungsfehler
		5	Watchdog-Fehler
2	Abbruch	0	Kein Abbruch
		9	Abbruch

Tab. 15: Status, Eigene Darstellung

Die Controller-Programme werden als Textdateien von der Bedien-Software an die Steuerung geschickt. Das war für den bisher genutzten Controller von unumgänglich, weil dieser die Programme zur Ausführung zwischenspeichern musste. In der neuen Steuerung liegen hingegen alle Controller-Programme im Speicher und müssen nur aufgerufen werden. Über einen Download-Befehl wird der Steuerung mitgeteilt, dass die nächsten empfangenen Nachrichten zusammen gehören und ein Controller-Programm beinhalten. Das Ende der Übertragung wird, wie in Abschnitt 3.4.2.3 beschrieben, über das ASCII-Zeichen mit dem Dezimalwert 26 gekennzeichnet. Zur Unterscheidung der Programme, wird in der ersten Nachricht der Name des jeweiligen Programms mitgeliefert. Somit kann diese Nachricht, nach einem Download-Befehl, für ein einfaches Handling in einen String konvertiert und mit den abgelegten Namen in der SPS verglichen werden. Ist eines der Controller-Programme identifiziert worden, wird das Programm ausgeführt.

Nachfolgend wird das Controller-Programm für das Finden des Maschinen-Nullpunkts in die Steuerung integriert. Hierfür wurde die Initialisierung und die Fehlerüberwachung programmiert. Im Hauptprogramm hingegen wurde nur ein Wechseln des Status berücksichtigt, um die Kommunikation zwischen Bedien-Software und Steuerung zu überprüfen.

Bei der Initialisierung werden alle aktuell anstehenden Fehler und Statusmeldungen zurückgesetzt und alle benötigten Trainingsparameter eingelesen. Zusätzlich wird die Systemzeit der Steuerung abgerufen und als Startzeit festgelegt. Die Startzeit wird für die Watchdog-Überwachung genutzt. Hierfür wird die Startzeit mit jedem Zyklus neu eingelesen und die Differenz aus Startzeit und aktueller Systemzeit der Steuerung berechnet. Ist diese Zeit größer als 4ms wird ein Watchdog-Fehler ausgelöst. Die Zykluszeit der Steuerung beträgt 1ms.

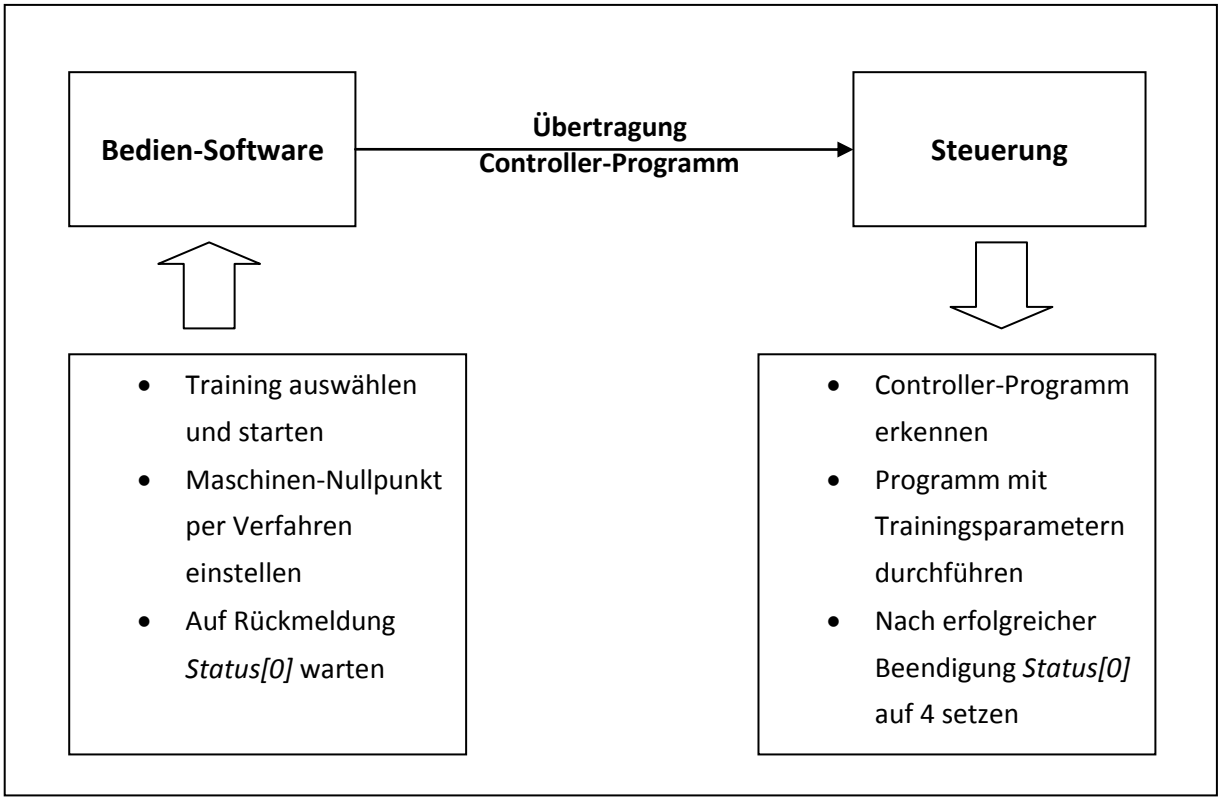


Abb. 22: Kommunikation Controller-Programm, Eigene Darstellung

3.5 Auswertung der Ergebnisse

Die Kommunikation des Trainingsgeräts wurde über die Bedien-Software und eine Steuerung der Firma Eckelmann realisiert. Dabei wurde zunächst das einfache Empfangen und Senden zwischen beiden Kommunikationsteilnehmern getestet. Anfänglich kam es hierbei zu häufigen Abstürzen der Bedien-Software, da die Kommunikationsschnittstelle schrittweise in die Steuerung implementiert wurde und erwartete Antworten nicht gesendet wurden. Nachdem ein Großteil der Übertragungselemente in der Steuerung programmiert war, konnte eine Verbindung zur Bedien-Software bis zum Einstellen des Maschinen-Nullpunkts gehalten werden. Ab hier konnte die Kommunikation nur noch über die Steuerung simuliert werden, da kein Trainingsgerät mit der Steuerung verbunden war. Ein Durchführen von Bewegungen und Rückmeldungen von externen Sensoren war nicht möglich und so wurde intern der Steuerung, der Rückgabewert für das Finden des Maschinen-Nullpunkts simuliert.

Bei der Verbindung zwischen Steuerung und Bedien-Software kam es hierbei zu keinerlei Zeitüberschreitungen. Die Zykluszeit der Steuerung von 1ms war zudem schnell genug um ein empfangen, verarbeiten und versenden der Nachrichten zu gewährleisten.

3.6 Weiterführende Arbeiten

Die Basis für die Kommunikation zwischen Steuerung und Bedien-Software ist bis zu diesem Punkt bereitgestellt. Desweiteren fehlt noch die Implementierung aller Controller-Programme und die Einbindung der externen Messgrößen (Drehmoment und Position) in die Steuerung. Die Umrechnungsfaktoren und Skalierungen für diese Messgrößen sind hierbei schon ermittelt worden.

Für die Implementierung der Controller-Programme ist eine Unterscheidung zwischen Trainingsprogrammen und Programmen zur Einstellung des Trainingsgeräts sinnvoll. Da alle Trainingsprogramme eine feste Anzahl an durchzuführenden Bewegungszyklen besitzen und sich hierbei lediglich die Kombination der Belastungsarten ändert, können die Trainingsprogramme über einen Ablauf geschehen. In diesem Trainingsablauf muss dann nur noch die jeweilige Art der Belastung und die Bewegungsrichtung des Trainingsgeräts der jeweiligen Bewegungsrichtung des Probanden zugeordnet werden.

Das nachfolgende Beispiel soll veranschaulichen, wie die Auswahl und Übergabe der Parameter für das Trainingsprogramm implementiert werden kann.

Ist beim Trainingsprogramm mit definierter Geschwindigkeit z.B. in positiver Bewegungsrichtung eine konzentrische Belastung eingestellt, muss der Proband Kraft entgegen der Bewegungsrichtung aufbringen. Das Trainingsgerät muss dabei den Probanden in Richtung des Umkehrpunktes drücken. Hier würde also von der Steuerung die Bewegungsrichtung positiv gesetzt werden und die Kraft des Probanden müsste permanent über ein erhöhen bzw. verringern des Drehmoments ausgeglichen werden um die Geschwindigkeit konstant zu halten. In negativer Bewegungsrichtung könnte hingegen eine exzentrische Belastung eingestellt sein und das Trainingsgerät müsste somit entgegen die Position des Umkehrpunktes drücken. Der Proband versucht hingegen das Trainingsgerät zum Umkehrpunkt zu bewegen. Die Bewegungsrichtung des Antriebs bleibt also positiv, wie während der ersten Bewegung und der Proband drückt es in die negative Bewegungsrichtung. Das Drehmoment muss hierbei so verändert werden, dass sich der Proband noch bewegen kann, jedoch dass das nicht schneller als die vordefinierte Geschwindigkeit passiert.

3.7 Zusammenfassung

In dieser Arbeit wird eine Kommunikationsschnittstelle für ein biomechanisches Trainingsgerät in der Medizintechnik erarbeitet. Die Kommunikation des Trainingsgeräts erfolgt hierbei über eine Bedien-Software und einen Controller, welcher das Trainingsgerät ansteuert. Zur Modernisierung soll der verbaute Controller durch eine Steuerung ersetzt werden, welche in CoDeSys V2.3 programmiert werden kann.

Nachfolgend werden die Entwicklungsschritte der Arbeit dargestellt:

- Analyse der Kommunikation
- Aufstellen und Vergleich der Kommunikationskonzepte
- Verbindung über TCP/IP herstellen
- Erstes Empfangen und Senden von Daten
- Datenanalyse und deren Verarbeitung
- Abspeichern von Parametern
- Versenden von Messwerten/ Parametern
- Erkennen von Controller-Programmen

Für das Implementieren der Software in dieser Steuerung, wurden die bereits bestehende Kommunikation analysiert. Daraufhin sind zwei verschiedene Konzepte für die Erstellung einer Kommunikationsschnittstelle näher betrachtet worden. Diese beiden Konzepte galt es zu vergleichen und das technisch und zeitlich bessere Konzept für diese Arbeit zu realisieren. Zudem wurde ein Vergleich zwischen UDP und TCP durchgeführt um die Übertragung über die Kommunikationsschnittstelle entsprechend zum Optimum zu integrieren.

Nachdem die Kommunikation analysiert worden ist, galt es die Nachrichtenübertragung mit der Steuerung herzustellen. Die Daten die von der Steuerung empfangen werden, sind mittels String-Auswertung analysiert worden. Die vorhandene Bedien-Software nutzt Befehle im ASCII Format und sendet diese an die unterlagerte Steuerung. Zusätzlich muss die Steuerung die Antwort, über ASCII-Zeichen an die Bedien-Software zurücksenden. Hierfür müssen intern Messwerte und Parameter entsprechend den vorhandenen Formaten entweder beim Empfang gespeichert oder bei Abfragen konvertiert und versendet werden.

3 Literaturverzeichnis

[1] 3S – Smart Software Solutions GmbH: Handbuch für SPS Programmierung mit CoDeSys 2.3, Kempten 2010 (PDF)

[2] Physiomed Elektromedizin AG: Physiomed User Manual MJ, Schnaittach 2011 (PDF)

[3] Kinco Automation: JD Servo User Manual, China 2014 (PDF)

[4] Walter Konhäuser: Industrielle Steuerungstechnik: Grundlagen und Anwendungen, Hanser Verlag, Wien 1998

[5] Gerhard Schnell, et al.: Bussysteme in der Automatisierungstechnik, 2. Auflage, Vieweg Verlag, Braunschweig/Wiesbaden 1996

[6] Peter Mandl, Andreas Bakomenko, Johannes Weiß: Grundkurs Datenkommunikation, 1. Auflage, Vieweg+Teubner, Wiesbaden 2008

[7] Klaus Lipinski: Lexikon TCP/IP Netzwerkprotokolle, MITP Verlag, Bonn 2001

[8] Frank J. Furrer: Industrieautomation mit Ethernet-TCP/IP und Web-Technologie, 3. Auflage, Hüthig Verlag, Heidelberg 2003

4 Glossar

CAN:	Der Can-Bus ist ein serielles Bussystem und gehört zu den Feldbussen.
CANopen:	CANopen ist ein auf CAN basierendes Kommunikationsprotokoll, welches seine Anwendung hauptsächlich in der Automatisierungstechnik findet.
Encoder:	Ein Encoder ist ein Sensor für Drehwinkel, welcher digitale Ausgangssignale liefert und am Auswertegerät decodiert.
exzentrisch:	Eine exzentrische Belastung, ist die Belastung des Muskels entgegen der Bewegungsrichtung, dabei wird der Muskel durch die äußere Last verlängert z.B. Gewicht ablegen.
konzentrisch:	Bei der konzentrischen Belastung hingegen wird der Muskel in Bewegungsrichtung belastet und dabei durch die äußere Last verkürzt z.B. Gewicht heben.

5 Tabellenverzeichnis

6 Abbildungsverzeichnis

Selbstständigkeitserklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.
Die Zustimmung der Firma zur Verwendung betrieblicher Unterlagen habe ich eingeholt. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder veröffentlicht noch einer anderen Prüfungsbehörde vorgelegt.
Ich versichere weiterhin, dass die auf elektronischem Wege eingereichten Unterlagen mit den schriftlichen Ausfertigungen übereinstimmen.

……..	…….
Ort, Datum	Unterschrift