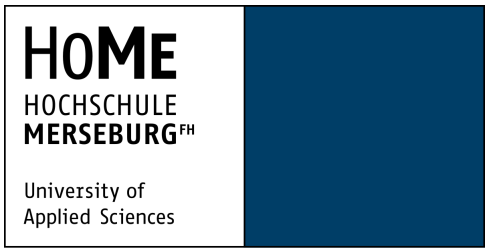


Hochschule Merseburg

Fachbereich Ingenieurwissenschaften

Abschlussarbeit zum Thema

Steuerung einer Bandanlage mittels QR-Codescanner

zur Erlangung des Grades Bachelor of Engineering

Angefertigt von
Cindy Kawetzki
Matrikelnummer 21607
Maschinenbau | Mechatronik | Physiktechnik

Erstgutachter
Prof. Dr. Peter Helm
Zweitgutachter
Dipl.-Ing. Ralph Seela

Vorgelegt am 12. Juli 2018

Eidesstattliche Erklärung

Ich versichere eidesstattlich durch eigenhändige Unterschrift, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Cindy Kawetzki

Zusammenfassung

Diese Arbeit befasst sich mit der Programmierung einer modellhaften Bandanlage, welche anhand von Speicherprogrammierbaren Steuerungen (SPS) und daran angeschlossener Peripheriegeräte gesteuert werden soll. Die Bandanlage befindet sich in einem Labor der Hochschule Merseburg und steht dort für diverse Projekte zum Erlernen des Umgangs mit Speicherprogrammierbaren Steuerungen zur Verfügung. Die Funktionalität der Anlage beschränkt sich grundlegend auf den Transport und die Einlagerung kleiner Werkstücke aus verschiedenem Material. Nachdem die komplette Steuereinheit an der Anlage mit Modulen der Firma FESTO ausgestattet worden ist, soll diese nach einem festgelegten Sortierprinzip wieder in Betrieb genommen werden. Die gesamte Programmierung erfolgt in der Umgebung von CODESYS V2.3 und V3.5.

Derzeit werden die Werkstücke erst auf den Förderbändern von verschiedenen Sensoren erkannt und anschließend je nach Material eingeordnet. Dieses System soll durch die Erweiterung der Anlage um einen QR-Codescanner der Firma ifm vereinfacht werden. Dieser soll es ermöglichen, die zu bearbeitenden Werkstücke schon am Anfang des Prozesses mittels eines zuvor aufgebrachten QR-Codes zu erkennen und anhand des Ergebnisses weiter zu verfahren. Nach Fertigstellung der Programmierung soll zusätzlich eine überschaubare Visualisierung zur Überwachung der Anlage am PC erstellt werden.

Als Ergebnis entstand im Zuge dieser Arbeit ein automatisierter Anlagenprozess, welcher die Anforderungen, die an die Anlage gestellt wurden, erfüllt. Sowohl mit dem QR-Codescanner als auch ohne ihn ist es möglich, die Anlage zu betreiben. Dabei wird ein festgelegter Sortieralgorithmus abgearbeitet. Ebenso kann der Status der Anlage und des Scanvorgangs in der geforderten Visualisierung überwacht werden.

Abstract

This thesis deals with the programming of a model conveyor system which is to be controlled by programmable logic controllers (PLC) and peripherals connected to them. The conveyor system is located in a laboratory of Merseburg University of Applied Sciences and is available there for various projects aimed at teaching how to handle programmable logic controllers. The functionality of the system is limited to the transport and storage of small workpieces made of different materials. After having been equipped with modules from FESTO, the complete control unit installed on the system should be put back into operation according to a defined sorting principle. All programming is based on the environment of CODESYS V2.3 and V3.5.

Currently, the workpieces are only detected on the conveyor belts by various sensors and then classified according to the material. This system will be simplified by the extension of the system with a QR code scanner made by ifm. This should make it possible to identify the workpieces already at the beginning of the process by means of a previously applied QR code and proceed on the basis of the result. After programming is completed, a manageable visualization for monitoring the system on the PC is to be created additionally.

As a result of this work, an automated system process was created, which meets the requirements of the system. Both, with the QR code scanner and without it, it is possible to operate the system. A defined sorting algorithm is executed. Likewise, the status of the system and the scanning process can be monitored using the visualization required.

Inhaltsverzeichnis

Abkürzungen

AS Ablaufsprache

AWL Anweisungsliste

COB-ID Communication Object Identifier

CPU Central Processing Unit

CSMA/CD Carrier Sense Multiple Access with Collision Detection

DP Dezentrale Peripherie

FBS Funktionsbausteinsprache

KOP Kontaktplansprache

PROFIBUS Process Field Bus

QR Quick Response

SPS Speicherprogrammierbare Steuerung

ST Strukturierte Text

UDP User Datagram Protocol

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Motivation

Die Fachgruppe der Automatisierungstechnik an der Hochschule Merseburg ist bereits seit geraumer Zeit in Besitz einer modellhaften Bandanlage, welche mittels verschiedener Sensorik Sortieraufgaben übernehmen kann. Die Steuerung der Anlage funktioniert anhand eines Systems mehrerer Speicherprogrammierbarer Steuerungen (SPS). Zu Testzwecken und im Rahmen der fortschreitenden Digitalisierung werden immer wieder neue Aufbauten und Ideen an der Anlage umgesetzt. Zuletzt wurden die direkt an der Bandanlage verwendeten SPS komplett durch Steuerungen der Firma *Festo* ersetzt. Gleichzeitig soll die Anlage um einen QR-Codescanner erweitert werden, um die zu sortierenden Objekte direkt anhand eines aufgebrachten Quick Response (QR)-Codes zu erkennen und zuordnen zu können. Bei der Erweiterung oder dem Umbau von solchen Systemen muss zunächst die Funktionalität getestet oder auch wiederhergestellt werden. Aus diesem Grund wurde sich dazu entschieden, aus der Thematik eine Aufgabenstellung für eine Abschlussarbeit zu entwickeln.

Die fertige Arbeit soll vor allem den nachfolgenden Studenten, welche an der Anlage lernen und arbeiten sollen, eine Hilfe sein, um im Rahmen von Praktika Programmieraufgaben an der Anlage bearbeiten zu können. Sie soll als Grundlage und Dokumentation zum Nachschlagen zur Verfügung stehen. Da die volle Funktionalität einmal hergestellt wurde, soll es danach auch anderen Studenten möglich sein, auf dieser Grundlage den Umgang mit der verwendeten Technik zu erlernen und die notwendigen Einstellungen vornehmen zu können.

1.2. Aufgabenstellung

Im Rahmen dieser Arbeit soll das bereits existierende Modell einer Bandanlage mit Verteilfunktion mit einer neuen Steuerungshardware der Firma Festo ausgestattet werden. Zudem soll das Verteilen der einzelnen Objekte mittels eines QR-Codescanners geschehen, welcher die auf den Objekten angebrachten QR-Codes liest. Die gelesenen Informationen sollen anhand eines Programms verarbeitet und an die Steuerung der Anlage weitergegeben werden. Die Kommunikation der Hardware untereinander wird über den PROFIBUS/PROFINET-Standard realisiert.

Auf dieser Basis ist eine beispielhafte Lösung der Anlagenfunktion zu erstellen. Diese umfasst die Hardwarekonfiguration, die SPS-Programmierung und die Realisierung der Verteilfunktion auf Basis von QR-Codes der einzelnen Werkstücke. Die Umsetzung der Programmierung soll in CODESYS V2.3 sowie V3.5 erfolgen. Aufgrund der neuen Möglichkeiten, welche der QR-Codescanner der Anlage offenlegt, kann im Zuge dieser Arbeit ebenso ein neues Konzept zur Sortierung der Bauteile erstellt werden. Je nach Informationsgehalt des verwendeten QR-Codes können die Werkstücke auf der Bandanlage einem festen Lager zugeordnet werden, anstatt diese nach Material und Farbe anhand von Sensorik zu ordnen. Der derzeitige Aufbau der Bandanlage ist in Abbildung 1.1 dargestellt.

Abbildung 1.1.: Aufbau der gesamten Anlage

Zusätzlich zur Programmierung der Bandanlage mit und ohne QR-Codescanner soll in CODESYS eine sogenannte Webvisualisierung erstellt werden. Die Visualisierung beinhaltet Anzeigeelemente, auf denen der Status der Anlage ersichtlich ist und diese

auch gesteuert werden kann. Es soll möglich sein, diese erstellte Visualisierung in einem Webbrowser aufzurufen.

1.3. Stand der Technik

Das neu aufgebaute System der Bandanlage besteht aus fünf Einzelstationen, von denen jede über diverse Module der Firma FESTO verfügt. Der Zugriff auf jedes einzelne FESTO-Gerät erfolgt über eine SPS der Firma WAGO. Somit sind insgesamt fünf SPS für die Steuerung der Anlage verantwortlich. Die SPS sind über Ethernet miteinander verbunden. Die Verbindung zwischen den WAGO-SPS und FESTO-Modulen erfolgt mittels PROFIBUS-DP. Der schematische Aufbau der verwendeten Hardware ist in Abbildung 1.2 zu erkennen. Dabei stellen die Blöcke „FESTO I/O“ die Ein- und Ausgangsmodule von FESTO dar.

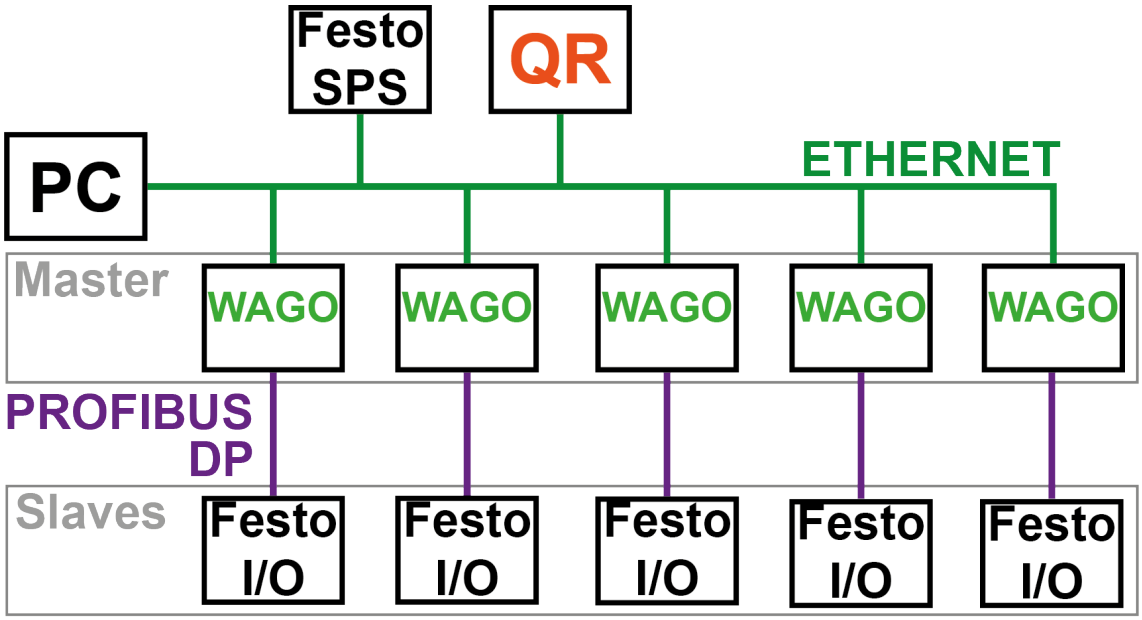


Abbildung 1.2.: Schematischer Aufbau der verwendeten Hardware

Die verwendeten PROFIBUS-DP-Module der Firma FESTO stammen aus der CPX-Reihe. Das elektrische CPX-Terminal ist ein modulares Peripheriesystem für Ventilinseln. Da alle Bewegungen der Bandanlage pneumatisch gesteuert werden, müssen die Steuerungen in der Lage sein, die benötigten Ventile an- und auszuschalten. Das CPX-System ist auf solche Anwendungen ausgelegt und bietet daher alle Komponenten, die hierfür benötigt werden. In diesem Fall wurden drei Arten von Modulen verbaut. Die wichtigste Einheit ist das Pneumatik-Interface VMPAL, welches die Verbindung

zwischen dem CPX-Terminal und der verbauten Ventilinsel MPA-L herstellt. Hierüber werden die Signale zum Öffnen und Schließen der Pneumatikventile weitergegeben. Zur weiteren Überwachung und Ansteuerung werden zudem ein Eingangsmodul mit 16 digitalen Eingängen (CPX-L-16DE) sowie ein Ein- und Ausgangsmodul mit je acht Ein- und Ausgängen (CPX-L-8DE-8DA) verwendet. Für nähere Informationen zu den genannten Modulen wird auf den von FESTO bereitgestellten Katalog zur CPX-Reihe verwiesen, welcher im Anhang C zu finden ist.

Als Master-SPS für die Steuerung der FESTO-Geräte dienen PFC200 Controller der Firma WAGO. Diese sind direkt mit einem Rechner verbunden und können so mit den darauf erstellten Programmen direkt bespielt werden. Die Kommunikation der SPS untereinander erfolgt dann mittels Ethernet. Genauere Informationen zum verwendeten Controller können dem Datenblatt im Anhang C entnommen werden.

Die Anlage wurde zudem um einen Multicode Reader O2I100 der Firma ifm erweitert. Dieser ist in der Lage, QR-Codes zu erkennen und die daraus gewonnenen Informationen weiterzugeben. Angesteuert wird der Multicode Reader anhand einer SPS der Firma FESTO, welche ebenfalls mittels Ethernet mit den übrigen Steuerungen verbunden ist. Das Gerät ist zu Beginn der Arbeit bereits an der Anlage verbaut, jedoch noch nicht in Betrieb genommen. Eine genauere Beschreibung des Multicode Readers findet sich im Anhang C. Die Erkennung der Werkstücke erfolgte bisher mit Hilfe von verschiedenen Sensoren, welche direkt am zugehörigen Lager angebracht sind. Die Sensoren reagieren auf die verschiedenen, verwendeten Materialien der Werkstücke. Somit ist es möglich, zu erkennen, ob sich ein Werkstück mit einem bestimmten Material gerade am vorgesehenen Lagerplatz befindet.

2. Technische Grundlagen

2.1. Grundlagen der SPS

Der Begriff Speicherprogrammierbare Steuerung drückt letztlich bereits aus, zu was ein solches Gerät in der Lage sein muss: Die Ausführung eines Programms, das eigens im Gerät gespeichert und jederzeit den Anforderungen des Anwenders angepasst werden kann. Die DIN IEC 60050-351 versteht unter einer SPS „eine rechnergestützte programmierte Steuerung, deren logischer Ablauf über eine Programmiereinrichtung, zum Beispiel ein Bedienfeld, einen Hilfsrechner oder ein tragbares Terminal, veränderbar ist.“ [4]

Vorteilhaft bei der Verwendung einer SPS sind vor allem der kompakte, platzsparende Aufbau und die Flexibilität bei der Veränderung des laufenden Programms. Zudem sind moderne SPS meist modular aufgebaut. Dadurch kann das Steuerungssystem an die zu erfüllende Aufgabe leicht angepasst und bei Bedarf erweitert werden. [16, S. 136 ff.]

2.1.1. Aufbau und Funktionsweise

Der grundlegende Aufbau einer SPS ist in Abbildung 2.1 erkennbar. Prinzipiell besteht diese aus einem Prozessor, einem Programmspeicher und diversen Ein- und Ausgabeeinheiten. Das im Programmspeicher befindliche Programm wird anhand eines externen Programmiergeräts erstellt und anschließend im Speicher der SPS abgelegt. Am Ausgang befinden sich die zu steuernden Stellglieder, welche abhängig vom aufgespielten Programm und den Signalen am Eingang ein- beziehungsweise ausgeschaltet werden können. Die vorhandene Anzahl an möglichen Ein- und Ausgangssignalen ändert sich bei einer SPS nicht, jedoch können je nach gewünschtem

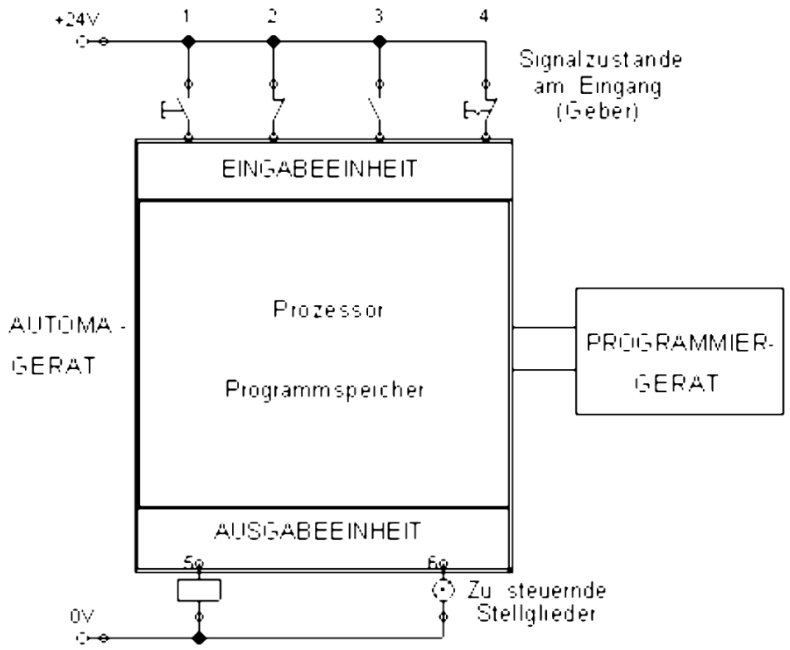


Abbildung 2.1.: Grundlegender Aufbau einer SPS [8, S. 158]

Verhalten anhand des erstellten Programms nur bestimmte Ein- und Ausgänge genutzt werden. Jeder Signalzustand am Eingang wird auf einem Bit der Eingabeeinheit gespeichert, wobei jedes Bit über eine eigene Adresse verfügt. Die Adressen können über die verwendete Programmiereinheit angesprochen und ausgelesen werden. [8, S. 158 f.]

2.1.2. Programmiersprachen nach DIN EN 61131-3

Die Programmierung einer Speicherprogrammierbaren Steuerung unterliegt der SPS-Programmiernorm DIN EN 61131-3 [3], welche einige grundlegende Konzepte beinhaltet. Hierzu zählen

- die Datentyp- und Variablenkonzeption,
- das Programmorganisationskonzept,

- das Fachsprachenkonzept sowie
- das Taskkonzept.

Anhand dieser allgemeingültigen Konzepte ist es möglich, herstellerunabhängige Programmierkonzepte zu entwerfen sowie wiederverwendbare Programme zu entwickeln. Ausgehend vom Fachsprachenkonzept gibt es fünf Programmiersprachen zur Entwicklung von SPS-Programmen. Die Kontaktplansprache (KOP) und Funktionsbausteinsprache (FBS) werden zur grafischen Programmierung genutzt, die Anweisungsliste (AWL) und der Strukturierte Text (ST) zur textuellen Programmierung. Hinzu kommt die Ablaufsprache (AS), welche sowohl textuelle als auch grafische Elemente beinhaltet. Da sich im Rahmen dieser Arbeit dazu entschieden wurde, die Programme als Strukturierten Text zu erstellen, wird im weiteren Verlauf der Arbeit allein auf diese Programmiersprache eingegangen. [17, S. 2 f.]

2.1.3. Strukturierter Text ST

Der Strukturierte Text ähnelt in seinem Aufbau den allgemein bekannteren Programmiersprachen wie beispielsweise Java oder C/C++. Die IEC-Norm schreibt hierfür die in der Tabelle 2.1 aufgelisteten Symbole zur abstrakten Beschreibung von Funktionsgleichungen vor.

Die Reihenfolge der Auflistung in der Tabelle beschreibt gleichzeitig die Priorität der Operationen im Programmablauf. Die oben platzierten Operationen haben die höchste Priorität, während die am weitesten unten angeordneten die niedrigste besitzen. Ebenfalls möglich ist die Verwendung von aus der Programmierung bekannten Anweisungen:

- Schleifen bspw. mit `FOR, WHILE`
- Bedingungen mit `IF, ELSE`
- Zuweisungen von Variablen etc.

Operation	Symbol
Klammerfunktion	(Ausdruck), Identifikator(Argument, z.B. LN(a)
Exponent	**, z.B. A**B = EXP(B*LN(a)
Negation	-
Complement	NOT
Multiplikation	*
Division	/
Modulo	MOD
Addition	+
Subtraktion	-
Vergleichen	<, >, <=, >=
Äquivalenz	=
Nichtäquivalenz	<>
Binär UND	&, AND
Binär ODER	OR
Binär EXOR	XOR

Tabelle 2.1.: Symbole für die Funktionsbeschreibung im Strukturierten Text [2, S. 138]

2.1.4. CODESYS als Programmierumgebung

CODESYS ist eine Software-Plattform, die unter anderem zur Erstellung von Programmen für Speicherprogrammierbare Steuerungen dient. Das Programmiertool *CODESYS Development System* basiert auf der Basis der IEC 61311-3 Norm, was die Verwendung der dafür festgelegten Programmiersprachen ermöglicht. Vorteilhaft ist hierbei die Möglichkeit der Projektierung - jede Anwendung zur Steuerung kann als ein Projekt erstellt und schließlich mit anderen Projekten verknüpft werden. Im Gegensatz zu der von Siemens für die firmeneigenen SPS zur Verfügung gestellten Software *Step 7* unterliegt CODESYS nicht einem bestimmten Hersteller, sondern bietet Lösungen zur Programmierung der Hardware zahlreicher Firmen an. Zudem ist die Nutzung der Software für den Anwender kostenlos. [5]

2.2. Kommunikation und Bussysteme

2.2.1. ISO/OSI-Referenzmodell

Zum Verständnis darüber, wie die Kommunikation zwischen einzelnen Hardwarekomponenten vonstatten geht, sollte zunächst ein Blick auf das ISO/OSI-Referenzmodell geworfen werden. Dieses Modell teilt die Kommunikation abstrakt in sieben Ebenen, auch Schichten genannt, auf, von denen jede eine bestimmte Funktionalität besitzt. Aufgrund dieser Aufteilung ist auch der Begriff OSI-Schichtenmodell geläufig. „Das OSI-Schichtenmodell beschreibt die Kommunikation von Partnerprozessen auf einer abstrakten Ebene. Es sind keine Angaben darüber enthalten, wie die einzelnen Schichten letztendlich implementiert werden sollen. Damit ist die Basis für ein offenes System geschaffen, das durch die Definition der Inhalte der einzelnen Schichten und durch die Festlegung der Schnittstellen auch dann genutzt werden kann, wenn ein Gesamtsystem aus Komponenten mehrerer Hersteller zusammengefügt wird.“ [14, S. 9]

Bei der Einteilung der Schichten wird grob in zwei verschiedene Arten unterschieden. Die Schichten 1 bis 4 werden als untere Schichten beziehungsweise Übertragungsschichten bezeichnet. Sie sind für die Datenübertragung zwischen den Endgeräten zuständig. Die oberen Schichten werden als Anwendungsschichten bezeichnet. Zu ihnen gehören die Schichten 5 bis 7, welche für die Koordination zwischen dem Anwenderprogramm und dem Betriebssystem des verwendeten Rechners zuständig sind. Eine Übersicht und kurze Erklärung zu den einzelnen Schichten des Modells ist in Tabelle 2.2 dargestellt.

2.2.2. Feldbus

Der Feldbus ist ein Bussystem, mit dessen Hilfe sogenannte Feldgeräte, wie beispielsweise Aktoren und Sensoren, mit Automatisierungsgeräten, zum Beispiel SPS, verbunden werden können. Da meist mehrere Teilnehmer in einem Netzwerk auf

Nr.	Bezeichnung	Erläuterungen
7	Anwendungsschicht (Application Layer)	stellt die auf dem netzwerk basierenden Dienste für die Programme des Endanwenders bereit (Datenübertragung, elekronische Post usw.)
6	Darstellungsschicht (Presentation Layer)	legt die Anwenderdaten-Strukturen fest und konvertiert die Daten, bevor sie zur Sitzungs- bzw. Anwendungsschicht gegeben werden (Formatierung, Verschlüsselung, Zeichensatz)
5	Sitzungsschicht (Session Layer)	definiert eine Schnittstelle für den Auf- und Abbau von Sitzungen, d. h. zur Benutzung der logischen Kanäle des Transportsystems
4	Transportschicht (Transport Layer)	stellt fehlerfreie logische Kanäle für den Datentransport zwischen den Teilnehmern bereit
3	Netzwerkschicht (Network Layer)	transportiert die Daten von der Quelle zum Ziel und legt die Wege der Daten im Netz fest
2	Datenverbindungsschicht (Data Link Layer)	legt die Datenformate für die Übertragung fest und definiert die Zugriffsart zum Netzwerk. Sie wird in die „Zugriffssteuerung für das Medium“ (MAC) und die „Logische Ankopplungs-Steuerung“ (LLC) unterteilt
1	Physikalische Schicht (Physical Layer)	definiert die elektrischen und mechanischen Eigenschaften der Leitung, Pegel

Tabelle 2.2.: Übersicht der Schichten im ISO/OSI-Referenzmodell [14, S. 10]

dieselbe Datenleitung zurückgreifen, muss über ein definiertes Protokoll festgelegt werden, welcher Teilnehmer wann die Berechtigung zum Senden von Daten erhält.

Zusätzlich wird auch Art und Weise, wie die einzelnen Komponenten miteinander verbunden werden, definiert. Diese wird als Topologie bezeichnet, wovon vor allem die folgenden vier zu nennen sind:

Zweipunkttopologie Zwei Teilnehmer sind direkt über eine Leitung miteinander verbunden.

Bustopologie Alle Teilnehmer sind mit demselben Übertragungsmedium (Bus) verbunden.

Sterntopologie Es gibt einen zentralen Teilnehmer, an den alle anderen Teilnehmer mittels Zweipunktverbindung angeschlossen sind.

Ringtopologie Es sind immer jeweils zwei Teilnehmer mittels Zweipunktverbindung miteinander verbunden, dabei entsteht ein geschlossener Ring.

Die im Zuge dieser Arbeit verwendeten Geräte, welche mittels Ethernet miteinander kommunizieren, also die SPS der Firma WAGO und FESTO sowie der QR-Codescanner, sind in Form der Bustopologie miteinander vernetzt. Vorteilhaft dabei ist die einfache Verkabelung und Erweiterbarkeit des Netzwerkes. Der schematische Aufbau einer solchen Bustopologie ist in Abbildung 2.2 zu erkennen. Die Kommunikation der WAGO SPS mit den Peripheriegeräten von FESTO erfolgt anhand einer Zweipunktverbindung.

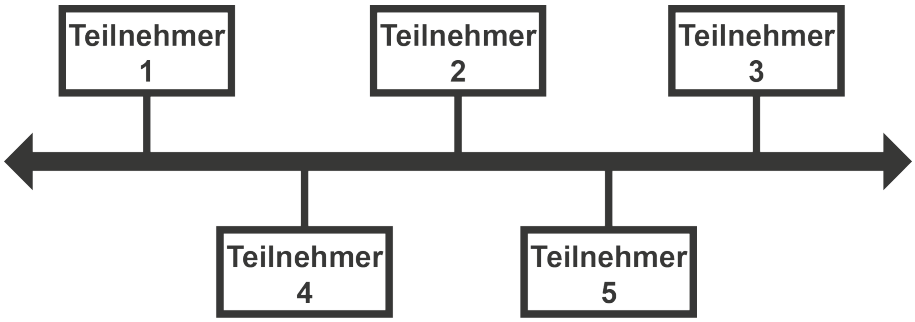


Abbildung 2.2.: Schematischer Aufbau einer Bustopologie

2.2.3. PROFIBUS

Der Process Field Bus (PROFIBUS) ist ein universeller Feldbus, welcher durch die internationale Normenreihe IEC 61158 standardisiert ist. Bei der Konzipierung des PROFIBUS ging es vor allem um die Entwicklung eines offenen und herstellerneutralen Feldbusstandards, um ein breites Anwendungsgebiet zu schaffen. Viele

Kommunikationssysteme sind an einen bestimmten Gerätehersteller gebunden, dies ist bei PROFIBUS nicht der Fall. In Bezug auf das ISO/OSI-Referenzmodell werden bezüglich des PROFIBUS nur drei der sieben Schichten genau beschrieben. In der ersten Schicht wird die Übertragungstechnik definiert, also die physikalische Schnittstelle des Gerätes sowie das damit verbundene Übertragungsmedium.

Die zweite Schicht bestimmt das Buszugriffsprotokoll. Das PROFIBUS-System arbeitet mit einem Master/Slave-Verfahren, wobei die Teilnehmer folgendermaßen definiert werden:

Master Aktiver Teilnehmer, darf Nachrichten ohne externe Aufforderungen senden, solange er die Buszugriffsberechtigung hat

Slave Passiver Teilnehmer, sind beispielsweise Ein-/Ausgangsgeräte, Aktoren oder Sensoren. Sie können Nachrichten empfangen, dürfen jedoch nur nach Aufforderung des Masters senden.

Die aktiven Teilnehmer im System erhalten ihre Buszugriffsberechtigung mittels des sogenannten Token-Passings. Es ist ein hybrides Zugriffsverfahren und stellt eine Kombination zwischen dem Master/Slave-Verfahren und dem Token-Bus dar. Das Token ist ein spezielles Zeichen oder eine spezielle Nachricht, welches in einem logischen Ring unter den aktiven Teilnehmern im Netzwerk weitergereicht wird. Nur jenes Gerät, welches derzeit im Besitz des Tokens ist, hat auch die Buszugriffsberechtigung und darf Nachrichten senden. Hat dieses Gerät seine Aufgaben durchgeführt, gibt es das Token an den nächsten aktiven Teilnehmer weiter und darf erst dann wieder senden, wenn das Token bei ihm angelangt ist. Die Funktionsweise des Token-Passings ist in Abbildung 2.3 dargestellt.

Die Schichten 3 bis 6 des ISO/OSI-Referenzmodells sind für PROFIBUS nicht ausgeprägt. Die Anwendungsschicht, also Schicht 7, schließt sich demnach direkt an die Datenverbindungsschicht an. [1, 14]

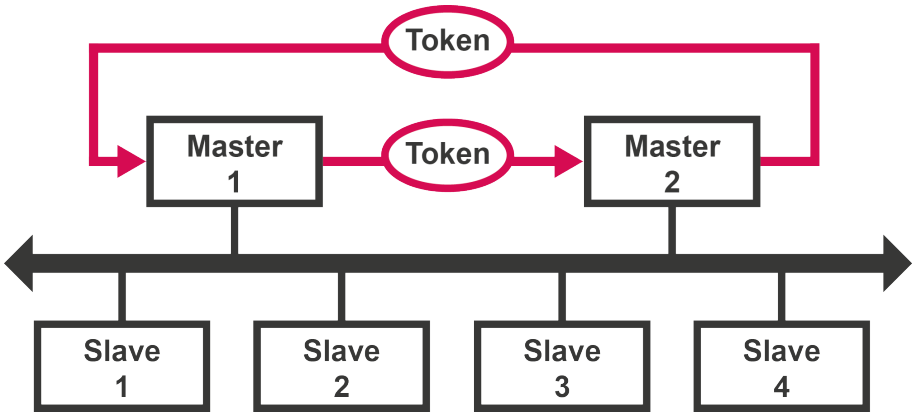


Abbildung 2.3.: Funktionsprinzip des Token-Passings

PROFIBUS-DP

Um einen effizienten und schnellen Datenaustausch in der Feldebene zu schaffen, wurde der PROFIBUS-Dezentrale Peripherie (DP) konzipiert. Hierbei erfolgt die Kommunikation jeweils zwischen genau einem Master und einem Slave. Andere Master sind nur in der Lage, lesend auf die nicht zu ihnen selbst gehörenden Slaves zuzugreifen. Dadurch wird ein unkontrolliertes Setzen von Ausgängen verhindert, was zu unerwünschtem Anlagenverhalten führen könnte. [14, S. 202 f.]

2.2.4. Ethernet

Ethernet bezeichnet eine Übertragungstechnik für (lokale) Netzwerke. Im ISO/OSI-Referenzmodell werden dabei nur die Schichten 1 und 2 definiert. Damit sind die Busphysik, also beispielsweise die Übertragungsmedien und Stecker, sowie das Buszugriffsverfahren festgelegt. Alle Stationen sind an einen gemeinsamen Bus angeschlossen und teilen sich somit die Übertragungsleitung. Die angeschlossenen Teilnehmer sind untereinander gleichberechtigt. Nach einer Überprüfung, ob der Bus frei ist, beginnt eine Station mit dem Senden von Daten. Tritt der Fall auf, dass zwei Stationen gleichzeitig beginnen zu senden, kommt es zu einer Kollision. Die Übertragung der Daten wird bei beiden Stationen abgebrochen und nach einer zufälligen Zeit wiederholt. Dieses Buszugriffsverfahren, welches auftretende Kollisionen berücksichtigt, wird

als Carrier Sense Multiple Access with Collision Detection (CSMA/CD) bezeichnet. Der Nachteil an dieser Übertragungsform besteht darin, dass es bei zunehmender Teilnehmerzahl im Netzwerk immer häufiger zu Kollisionen kommt und die Übertragungszeit dadurch nicht eindeutig vorhersehbar ist. Bei der Datenübertragung erhält ein Ethernet-Paket eine Quell- sowie Zieladresse. Diese Adressen heißen auch MAC-Adressen, sind einzigartig und werden bereits vom Hersteller festgelegt. Daher kann es bei der Adressierung eines Pakets nicht zu Verwechselungen kommen.

2.3. Funktionsweise von QR-Codes

Das Prinzip des QR-Codes wurde 1994 von der japanischen Firma DENSO WAVE INCORPORATED entwickelt. Die bisweilen verwendeten Barcodes konnten aufgrund ihrer Eindimensionalität maximal 20 alphanumerische Zeichen beinhalten. Da die Anforderungen an maschinenlesbaren Codes jedoch immer weiter stiegen, wurde ein System benötigt, welches mehr Daten enthalten und dennoch schnell gelesen werden kann. Der QR-Code ist ein zweidimensionaler Code, welcher nach heutigem Stand maximal 7.089 numerische Zeichen beinhalten kann. Im Juni 2000 erhielt der QR-Code Standard eine ISO Zertifizierung, die ISO/IEC18004. [7]

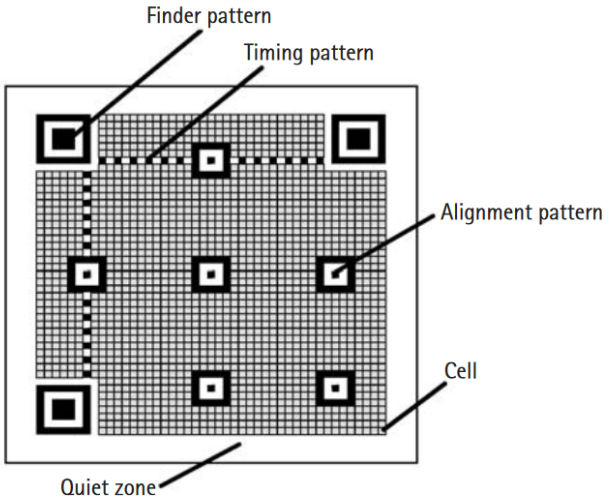


Abbildung 2.4.: Aufbau eines QR-Codes [15, S. 66]

Der Vorteil von QR-Codes besteht nicht nur in ihrem großen Speichervermögen, sondern auch in der schnellen Lesezeit. Aufgrund ihres einzigartigen Aufbaus können QR-Codes von entsprechenden Geräten schnell erkannt und zudem von anderen Codes oder Gegenständen unterschieden werden. Der schematische Aufbau eines QR-Codes ist in Abbildung 2.4 zu sehen. Das *Finder Pattern* liefert Informationen über Position, Größe und Winkel des Codes. Damit ist es unerheblich, aus welchem Blickwinkel gescannt wird. Das *Alignment Pattern* gibt zusätzlich Aufschluss über die exakte Ausrichtung des Codes, was vor allem bei stärkeren Verzerrungen hilft, das Muster dennoch genau zu erkennen. Mithilfe des *Timing Patterns* wird der Mittelpunkt jeder Zelle des QR-Codes ermittelt. Die *Quiet Zone* dient dazu, den Code eindeutig von seiner restlichen Umgebung abzugrenzen. Schlussendlich bleibt die *Data Area*, wo die eigentlichen Informationen, die mittels des Codes übertragen werden sollen, gespeichert sind. Die Kodierung dieser Informationen erfolgt anhand der Anordnung der schwarzen und weißen Punkte innerhalb der *Data Area*. [15, S. 66 f.]

Neben der Möglichkeit, QR-Codes auch aus sehr verzerrten Winkeln erkennen zu können, bietet der Standard eine Fehlerkorrektur, bei der ein Code beispielsweise trotz Beschädigung oder Verschmutzung von bis zu 30 % lesbar bleibt. Je nach Umgebung, in welcher der Code angebracht ist, sollte sich für ein passendes Korrekturlevel entschieden werden. [6]

Neben den Vorteilen, die der QR-Code mit sich bringt, gibt es allerdings auch Nachteile. Um einen einfachen Strichcode zu lesen, wird lediglich ein System benötigt, welches einen Laserstrahl erzeugen und damit die weißen Zwischenräume in Längsrichtung des Codes erkennen kann. Aufgrund dessen, dass ein QR-Code zweidimensional ist, wird hier jedoch eine Kamera sowie die passende Bildverarbeitung benötigt, um das gescannte Bild auszuwerten. Solch ein Kamerasystem ist weitaus kostenintensiver als ein einfacher Strichcodeleser. Daher sollte für industrielle Zwecke abgewogen werden, ob für die gewünschte Funktionalität ein Strichcodeleser, welcher 20 Zeichen beinhalten kann, genügt, oder ob mehr Daten übermittelt werden müssen.

3. Die Bandanlage

3.1. Aufbau

Die verwendete Bandanlage besteht im Groben aus zwei Transportbändern, an deren Seiten sich Lagerplätze zur Sortierung der verwendeten Werkstücke befinden. Die Werkstücke werden in einem am Anfang des ersten Bands befindlichen Magazin gelagert und bei Bedarf ausgeschoben. Zwischen den Bändern steht ein Umsetzer, welcher ein Werkstück vom ersten auf das zweite Band transportieren kann. Auf dem zweiten Band kann das Objekt zusätzlich von einem Wender um 180° gedreht werden. Mit diesem Aufbau wird die Sortierung der in Abbildung 3.1 gezeigten Werkstücke vorgenommen.

Abbildung 3.1.: Verwendete Werkstücke: Aluminium, weißer und schwarzer Kunststoff

Alle beweglichen Komponenten der Anlage werden pneumatisch gesteuert. Hierfür sind die an den jeweiligen Stationen angebrachten Ventilinseln zuständig. Je nach gewünschter Art der Bewegung können dort Ventile geöffnet oder geschlossen werden. Für die Öffnung eines solchen Ventils genügt ein kurzes impulsartiges Signal. Daher würde eine einmal angefangene Bewegung auch bei einem Notaus oder Stopp der Anlage noch zu Ende ausgeführt werden. Jedoch handelt es sich meist um sehr kleine Bewegungen, die ohnehin in einem kurzen Zeitfenster zu Ende geführt werden. Zum besseren Verständnis über die Funktionsweise der Bandanlage sind im Folgenden deren einzelne Stationen genauer beschrieben. Jede Station besitzt spezielle Aufgaben

und wird über eine eigens für sie vorgesehene SPS und deren zugehörige Ventilinsel gesteuert.

3.1.1. Magazin

Im Magazin werden die Werkstücke übereinander gelagert und bei Bedarf auf das erste Band (Abschnitt 3.1.2) gesetzt. Dies funktioniert über einen Schieber, welcher das unterste Objekt des Stapels auf das Band schiebt und anschließend wieder zurück fährt. Dies ist in Abbildung 3.2 erkennbar. Mit der SPS des Magazins ist zudem die Steuereinheit zur Bedienung der Bandanlage gekoppelt. Hier finden sich die nötigen Schalter zum Ein- bzw. Ausschalten, der Notaus sowie für das Richten (Abschnitt 4.3.6 S. 46). Der Betriebszustand der Anlage ist hier zudem durch farbige Lampen (Tabelle 4.2 S. 43) sowie einem akustischen Signal ersichtlich.

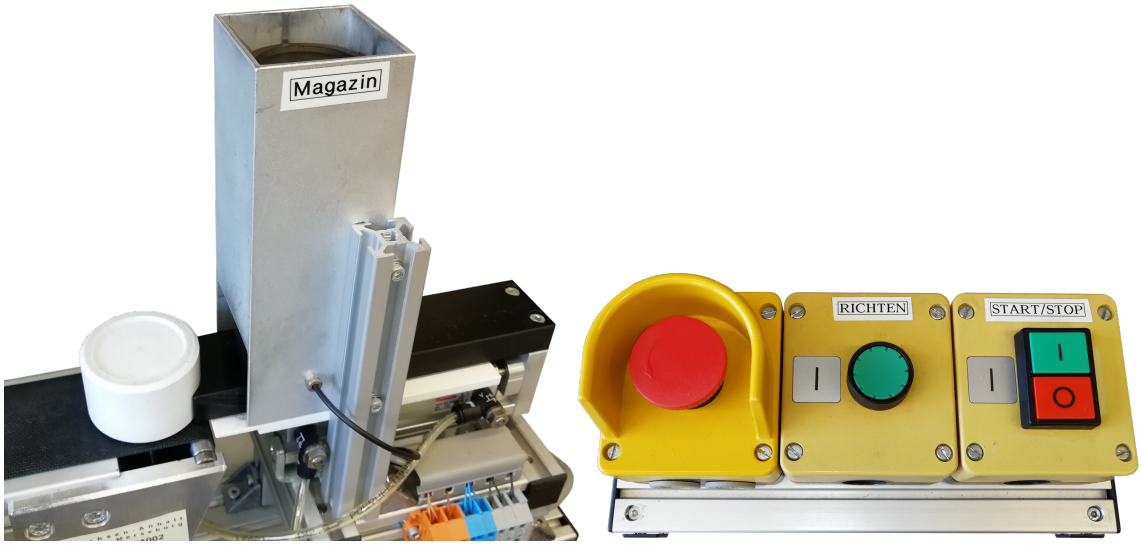


Abbildung 3.2.: Magazin mit ausgefahrenem Schieber und Bedienungseinheit

3.1.2. Band 1

Das bereitgestellte Werkstück fährt über das erste Band entweder bis zu einem der Lager, die seitlich von diesem angebracht sind, oder bis zum Ende des Bands, je nach

realisiertem Sortieralgorithmus im Programm. Um die Werkstücke an den Lagern zu erkennen, sind an deren Positionen verschiedene Sensoren angebracht, welche in der Lage sind, die unterschiedlichen Werkstücke zu erkennen. Der erste Sensor ist ein induktiver Sensor, welcher nur in der Lage ist, das Werkstück aus Aluminium zu erkennen. Am zweiten Lager befindet sich ein optischer Sensor, welcher so eingestellt ist, dass er nur das weiße, jedoch nicht das schwarze Werkstück erkennt. Am letzten Lagerplatz ist ebenfalls ein optischer Sensor angebracht. Dieser ist in der Lage, alle verwendeten Materialien zu erfassen. Am Ende des Bands befindet sich zudem eine Lichtschranke, um zu erkennen, wann ein Objekt am Ende des Förderbands angelangt ist. In Abbildung 3.3 sind die einzelnen angebrachten Komponenten an Band 1 erkennbar.

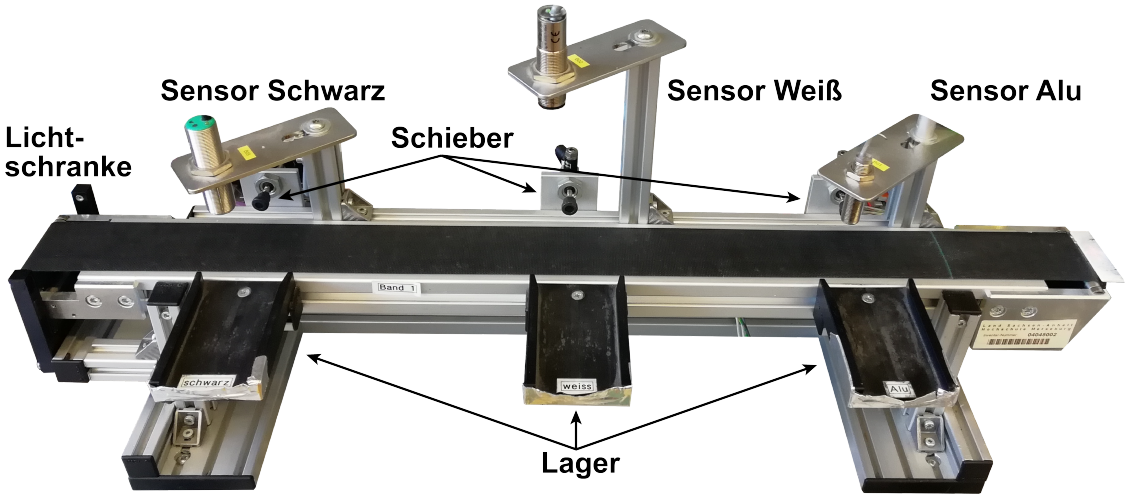


Abbildung 3.3.: Aufbau vom ersten Band

3.1.3. Umsetzer

Der in Abbildung 3.4 abgebildete Umsetzer ist dafür zuständig, ein am Ende des ersten Bands angelangtes Werkstück auf den Anfang des zweiten Bands (siehe 3.1.4) zu befördern. Dazu wird das Objekt aufgenommen, angehoben, zur anderen Seite geschwenkt und dort wieder abgesetzt.

Abbildung 3.4.: Pneumatischer Umsetzer zwischen Band 1 und Band 2

3.1.4. Band 2 mit Wender

Das zweite Band ist bezogen auf die Lager und den dort befindlichen Sensoren genauso aufgebaut wie das erste Band (vgl. Kapitel 3.1.2). Der Unterschied besteht darin, dass am Anfang des Bandes, also noch vor den Lagern, ein Wender angebracht ist. Fährt ein Objekt über das zweite Band, besteht die Möglichkeit, dieses um 180° zu drehen, bevor es weiter zu den Lagern transportiert wird. Der Wender befindet sich ein Stück oberhalb des Bands und kann beim Erkennen eines Objekts nach unten fahren, dieses greifen und den Wendevorgang durchführen. Anschließend wird das Objekt wieder auf dem Band abgesetzt. Der Wender ist in Abbildung 3.5 dargestellt. Auch am Ende dieses Bands befindet sich eine Lichtschranke, um komplett durch gefahrene Werkstücke zu erkennen.

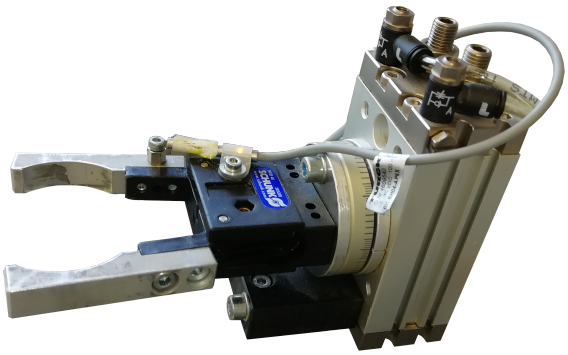


Abbildung 3.5.: Wender

3.2. Anforderungen und Randbedingungen

Zu Beginn dieser Arbeit musste eine Festlegung getroffen werden, wie die Bandanlage die einzelnen Werkstücke sortieren soll. Zunächst sollte dies ohne Verwendung des QR-Codescanners umgesetzt werden, das heißt, die bereits verbauten Sensoren waren für die Erkennung der Werkstücke zuständig. Unter folgenden Aspekten sollte die Anlage programmiert werden:

Der Prozess beginnt, wenn am Bedienterminal der Startschalter betätigt wird. Jeweils zu Beginn eines Durchlaufs wird ein Werkstück aus dem Magazin heraus geschoben. Das erste und dritte Objekt einer Farbe soll im jeweiligen Lager des ersten Bands untergebracht werden, das zweite und vierte in den Lagern des zweiten Bands. Zudem sollen die Werkstücke vor dem Einordnen auf dem zweiten Band vom Wender einmal gedreht werden. Sind alle Lager voll, fährt das Werkstück bis zum Ende des zweiten Bands durch, ohne gewendet zu werden. Ist es dort angekommen, endet der Prozess.

Die gesamte Anlage muss während des laufenden Betriebs bei Betätigung des Stoppschalters angehalten werden können und beim erneuten Start dort weiter machen, wo sie zuvor gestoppt wurde. Bei Betätigung eines Notaus *darf der Prozess erst weitergeführt werden, wenn die Anlage gerichtet wurde, das heißt, alle Anlagenteile leer gefahren wurden.*

Bei der späteren Verwendung des QR-Codescanners sollten die zurvor genannten Aspekte prinzipiell beibehalten werden. Allerdings ist es nun nicht mehr notwendig,

von vornherein festzulegen, welches Objekt einer bestimmten Farbe in welches Lager geordnet werden muss. Die Verwendung von QR-Codes erlaubt es, die Position des Werkstückes selbst im Code zu definieren, beispielsweise dass ein bestimmter weißer Stein immer in das hintere Lager von Band 2 platziert werden soll. Diese Variante wurde im Rahmen der Arbeit jedoch nicht mehr umgesetzt.

4. Umsetzung und Programmierung ohne QR-Codescanner

Die Programmierung der einzelnen Anlagenstationen musste aufgrund der genutzten Hardware mit CODESYS V2.3 realisiert werden, auch wenn diese nicht die aktuellste Version der Programmierumgebung darstellt. Die verwendeten Master-SPS der Firma WAGO sind mit der aktuellen Version CODESYS V3.5 nicht kompatibel.

4.1. Einbinden der SPS in CODESYS

Um die eigentliche Steuerung der SPS in CODESYS V2.3 realisieren zu können, ist es zunächst nötig, das Programm ordnungsgemäß zu konfigurieren. Zuerst wird nach dem Programmstart von CODESYS ein neues Projekt angelegt. Dabei wird direkt nach den *Zielsystem Einstellungen* gefragt. Hier muss, wie in Abbildung 4.1 zu sehen, die richtige Master-SPS als Konfiguration ausgewählt werden. In diesem Fall ist das die WAGO 750-8208.

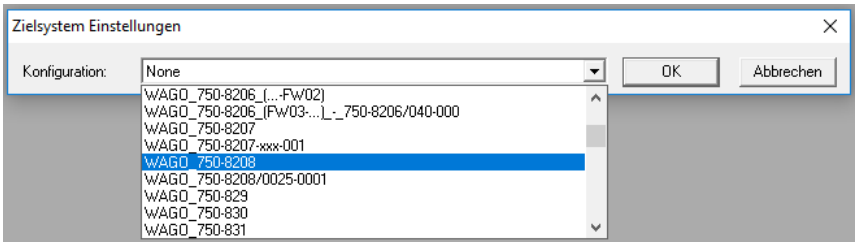


Abbildung 4.1.: Konfiguration der *Zielsystem Einstellungen*

Anschließend müssen die Einstellungen für das spätere Hauptprogramm festgelegt werden. Das zugehörige Fenster für *Neuer Baustein* ist in Abbildung 4.2 zu sehen.

Der Typ des Bausteins ist in diesem Fall *Programm*. Dieses wird bei der späteren Ausführung auf der SPS automatisch aufgerufen und abgearbeitet, es stellt sozusagen

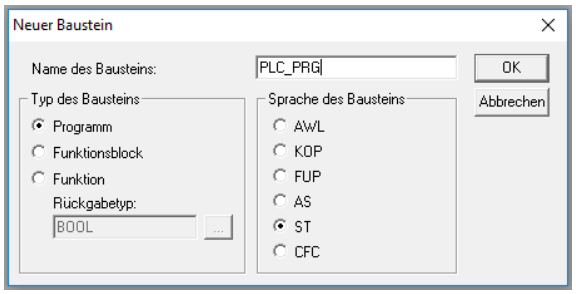


Abbildung 4.2.: Konfiguration für *Neuer Baustein*

das Hauptprogramm dar. Zusätzlich kann noch ein neuer Name für den Baustein vergeben werden. Standardmäßig erhält das Hauptprogramm in CODESYS den Namen PLC_PRG, was so beibehalten wurde. In der Auswahl für die Sprache des Bausteins stehen die nach DIN EN 61131-3 geforderten Programmiersprachen zur Verfügung. Da in der gesamten Arbeit mit dem Strukturierten Text gearbeitet wird, muss hier ST ausgewählt werden. Nach Bestätigung der Einstellungen öffnet sich in CODESYS der Editor zum Bearbeiten des soeben erstellten Hauptprogramms.

Um jedoch mit der Programmierung beginnen zu können, sind einige weitere Schritte notwendig. Hierfür muss über den Reiter *Ressourcen* die *Steuerungskonfiguration* ausgewählt werden. Diese sieht nach dem Öffnen wie in Abbildung 4.3 dargestellt aus.

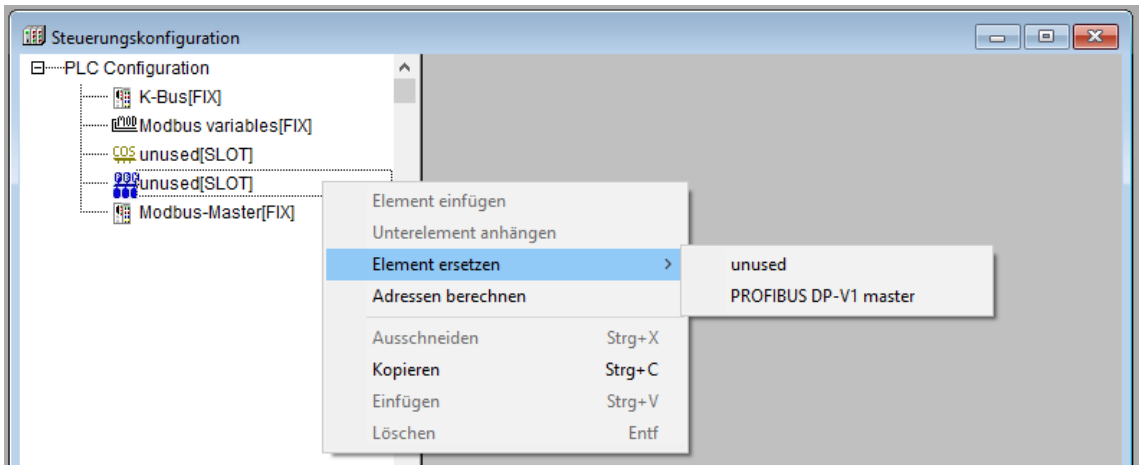


Abbildung 4.3.: *Steuerungskonfiguration*

Wie zu sehen ist, muss der zweite *unused(SLOT)* zunächst mit dem *PROFIBUS DP-V1 master* ersetzt werden. Dadurch wird die verwendete *WAGO* SPS als Master festgelegt. Zudem muss der ebenfalls gezeigte *K-Bus[FIX]* bearbeitet werden. Über einen Rechtsklick und anschließend *Bearbeiten* gelangt man zum *WAGO I/O Check*. Hier werden die vorhandenen Ein- und Ausgänge des verwendeten Systems automatisch überprüft und an das Programm übermittelt. Die verwendete WAGO SPS wird dabei wie in Abbildung 4.4 im Programm dargestellt. Darauf sind alle Anschlüsse, Klemmen und Statusleuchten erkennbar, welche auf dem echten Modul angebracht sind.

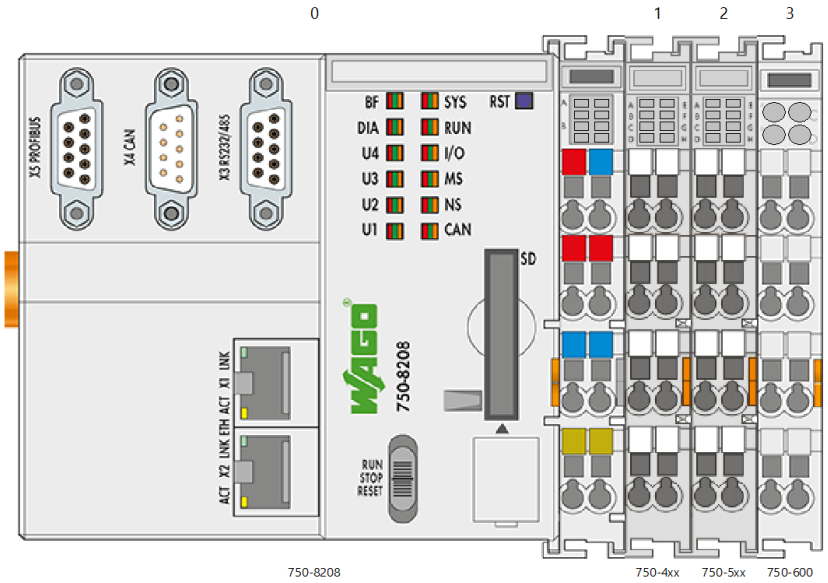


Abbildung 4.4.: Schematische Darstellung der verwendeten WAGO 750-8208

Damit das Programm im Netzwerk auf die richtige Master-SPS zugreifen kann, muss im Programm noch die zugehörige IP-Adresse der SPS angegeben werden. In den oberen Reitern von CODESYS wird dazu *Online* und anschließend *Kommunikationsparameter* ausgewählt. In dem sich öffnenden Fenster kann nun unter *Neu* ein neuer Kanal erstellt werden. Dieser benötigt einen Namen sowie die Einstellung *TCP/IP*. Nach dem erfolgreichen Anlegen muss nun der als *Adresse* angezeigte Wert *locahost* mit der IP-Adresse der zu programmierenden WAGO-SPS ersetzt werden. Die zu den Stationen zugehörigen IP-Adressen sind in Tabelle 4.1 ersichtlich.

Der festgelegte Master benötigt an dieser Stelle noch seine Unterkomponenten beziehungsweise Slaves. In diesem Fall erhält jeder Master nur einen Slave, da jedes FESTO CPX-Terminal an der Anlage mit einer eigenen WAGO SPS verbunden ist. Um den Slave festzulegen, hängt man an den eingestellten *PROFIBUS DP-V1 master*, wie in Abbildung 4.5 zu sehen, ein Unterelement. Standardmäßig waren in der Auswahlliste nur Komponenten der Firma WAGO vorhanden. Zur Einbindung des FESTO-Terminals musste zunächst die Konfigurationsdatei vom Hersteller geladen werden. Diese erhält man im Support Portal der Firma FESTO. [10]

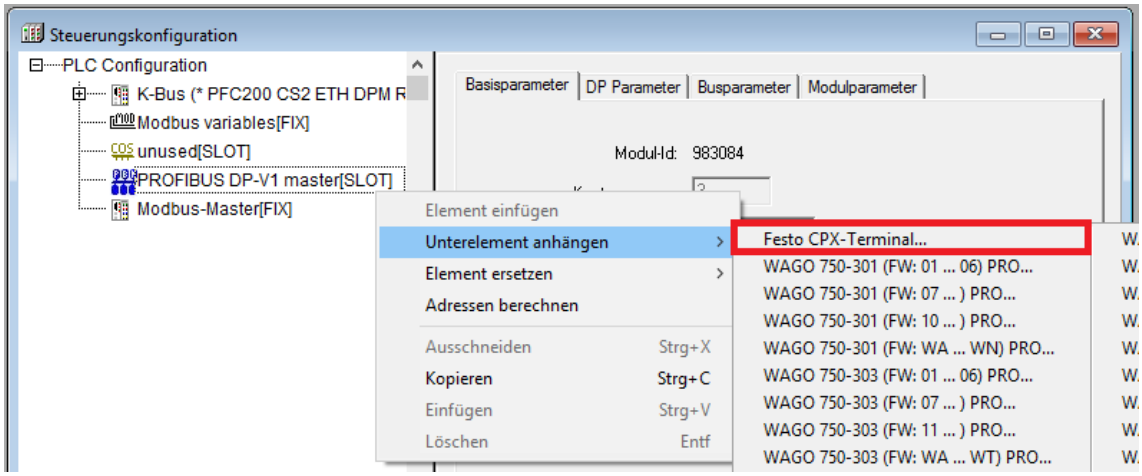


Abbildung 4.5.: Unterelement hinzufügen

Nun müssen im ausgewählten *Festo CPX-Terminal* die jeweils daran angeschlossenen Module ausgewählt werden. Genaueres zu den einzelnen Modulen kann in Kapitel 1.3 nachgelesen werden. Die Auswahl der Module erfolgt im Reiter *Ein-/Ausgänge* bei angewähltem *Festo CPX-Terminal[VAR]*. Dort finden sich geordnet alle Arten von Eingabe-, Ausgabe, Ein-/Ausgabe- sowie Leermodulen, die für das CPX-Terminal in Frage kommen. In Abbildung 4.6 sind beispielhaft alle der maximal drei verwendeten Module ausgewählt worden. Diese Aufstellung kommt allerdings nur für die Steuerung von Band 1 und 2 zum Einsatz, da sie die einzigen sind, die ein Modul mit 16 digitalen Eingängen besitzen. Für die anderen Stationen entfällt dieses. Im gleichen Konfigurationsmenü, unter dem Reiter *DP-Parameter*, muss zuletzt noch die richtige Stationsadresse eingegeben werden. Jede FESTO-Station besitzt eine eigene, eindeutige Adresse, welche sich aus vier Bits zusammensetzt. Fehlt diese in

der Konfiguration, ist es nicht möglich, mit dem FESTO-Terminal zu kommunizieren. Die Bits für diese Stationsadresse sind als Schalter an der jeweiligen Station einsehbar und sind in der Tabelle 4.1 übersichtlich aufgelistet.

Station	Stationsadresse	IP-Adresse
Magazin	10	149.205.120.79
Band 1	11	149.205.120.80
Umsetzer	12	149.205.120.81
Band 2	13	149.205.120.63
Wender	14	149.205.120.62

Tabelle 4.1.: Stations - und IP-Adressen der einzelnen Anlagenstationen

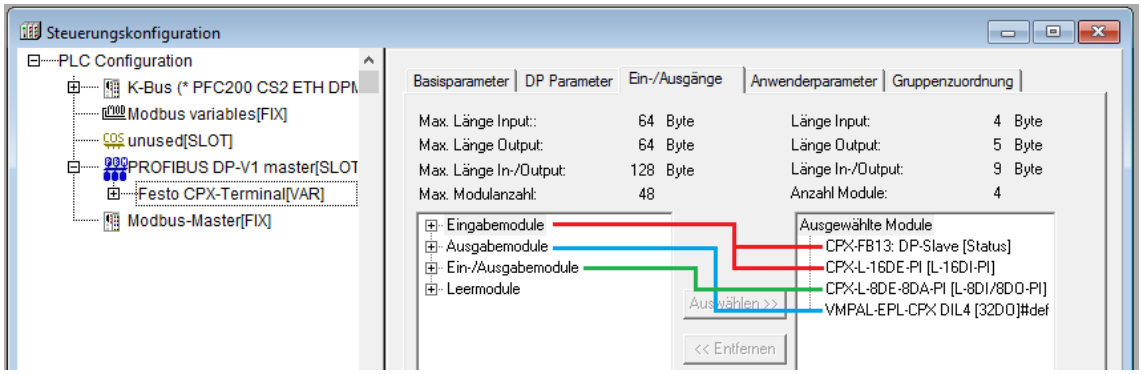


Abbildung 4.6.: Auszuwählende Module für das CPX-Terminal

4.2. Einstellen der Netzwerkvariablen

Bei der Programmierung der einzelnen Elemente der Bandanlage wurden die notwendigen Funktionalitäten zunächst manuell für jede einzelne Station getestet. Damit die Anlage jedoch voll automatisiert und ohne Eingriff von außen funktioniert, müssen alle vorhandenen Stationen miteinander kommunizieren können. Hierfür stellt CODESYS sogenannte Netzwerkvariablen zur Verfügung. Diese Variablen stehen nach erfolgreicher Konfiguration des Projekts allen Stationen im gleichen Netzwerk zum Lesen und/oder Schreiben zur Verfügung.

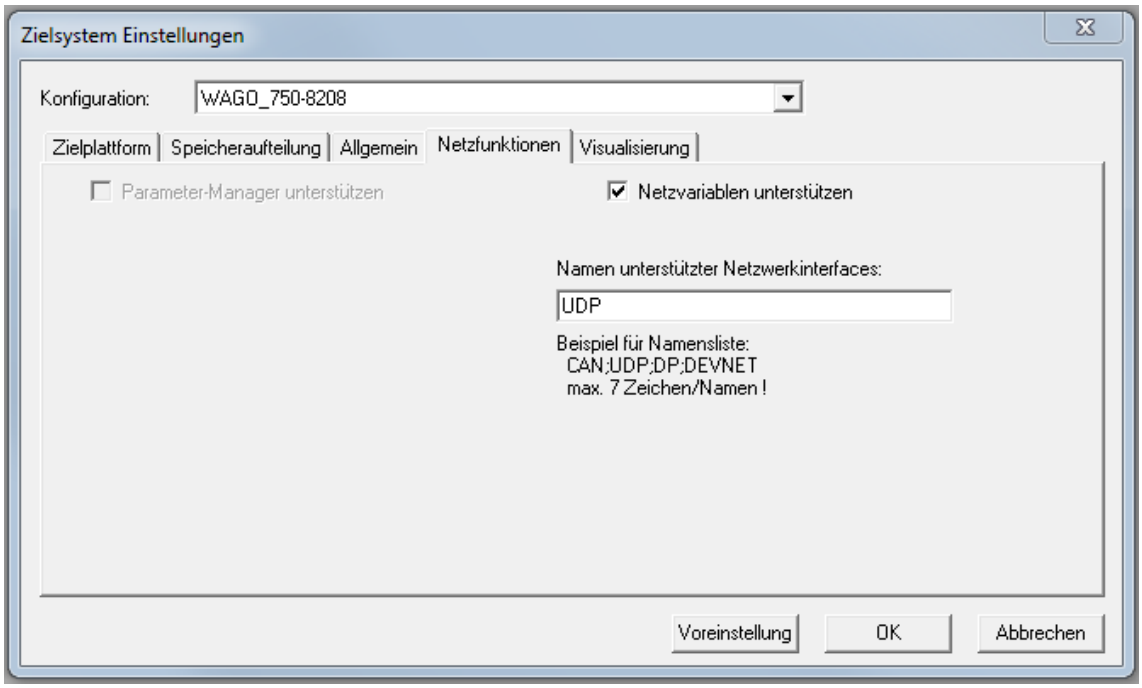


Abbildung 4.7.: Aktivierung der Netzwerkvariablen

Zunächst muss in den *Zielsystemeinstellungen* des Projekts festgelegt werden, dass Netzwerkvariablen verfügbar sind. Hierzu muss, wie in Abbildung 4.7 zu sehen, der Haken bei *Netzwerkvariablen unterstützen* gesetzt werden. Ist dieser Punkt aktiviert, muss außerdem das zu verwendende Kommunikationsprotokoll festgelegt werden. Alle Stationen sollen mittels User Datagram Protocol (UDP) miteinander kommunizieren. Dies ist ein verbindungsloses, ungesichertes Übertragungsprotokoll für einzelne Datenpakete, sogenannte Datagramme. Die Eigenschaften dieses Protokolls definieren sich wie folgt:

Verbindungslos Vor der Datenübertragung muss nicht erst eine Verbindung aufgebaut werden, ein Netzwerkteilnehmer kann sofort senden.

Ungesichert Es gibt keine Garantie, dass ein gesendetes Datenpaket ankommt, die Reihenfolge des Ankommens der Daten ist nicht zwingend gleich der Reihenfolge des Sendens.

Der Vorteil von UDP liegt vor allem in dessen einfacher Handhabung und Geschwindigkeit. Da das Aufbauen einer Verbindung entfällt, entstehen anfangs keine Verzögerungen. Zudem können Datenpakete immer an mehrere Empfänger gesendet werden, da es durch den Verzicht auf Bestätigungsnachrichten unerheblich ist, wie viele Stationen adressiert werden. Da für diese Arbeit alle Elemente der Anlage Variablen von teilweise mehreren Stationen verarbeiten müssen, ist dieser sogenannte Multicast ein großer Vorteil. Indem die Datenpakete periodisch immer wieder gesendet werden, ist es vor allem bei kleinen Datenmengen unerheblich, wenn eines verloren geht. Die Kommunikation erfolgt dennoch schnell genug für den hier vertretenen Anwendungsfall. [12, S. 127 f.]

Da nach der vorgenommenen Einstellung in den Zielsystemeinstellungen nun Netzwerkvariablen genutzt werden können, müssen diese noch weiter konfiguriert werden. Zunächst wird eine neue Variablenliste, in der die Netzwerkvariablen enthalten sind, erstellt. Im Grunde genommen sind die Netzwerkvariablen auch globale Variablen, nur mit anderen Einstellungen. Während globale Variablen vom gesamten Projekt mit ihrem vergebenen Namen angesprochen werden können, stehen die Netzwerkvariablen allen anhand des Netzwerkes verbundenen Projekten und Komponenten zur Verfügung. Zunächst wird eine neue Variablenliste angelegt, dies geschieht im Menüpunkt *Ressourcen*. Im Unterpunkt *Globale Variablen* lässt sich ein neues Objekt einfügen. Dieses wird die neue Variablenliste. Die Liste muss hier mit einem Namen versehen werden. Im umgesetzten Projekt wurde diese als *Netzwerk_Variablen* bezeichnet.

Nach der Erstellung der neuen Variablenliste muss diese nun konfiguriert werden. Unter den *Objekteigenschaften* öffnet sich das Fenster, wie in Abbildung 4.8 dargestellt. Die komplette Ansicht öffnet sich erst, nachdem *Netzwerk hinzufügen* angewählt wurde. Die mitunter wichtigste Einstellung ist die Communication Object Identifier (COB-ID) der erstellten Netzwerkvariablenliste. Die Variablenliste steht für alle Teilnehmer bereit, welche dieselbe COB-ID besitzen. Der Wert kann beliebig gewählt werden, in diesem Fall wurde sich für 50 entschieden. Des Weiteren benötigen die Teilnehmer im Netzwerk Lese- und Schreibzugriff, um den Status der Variablen erkennen und ändern zu können. Die Übertragung der Variablen soll für diesen Anwendungsfall immer dann erfolgen, wenn eine Änderung dieser stattgefunden

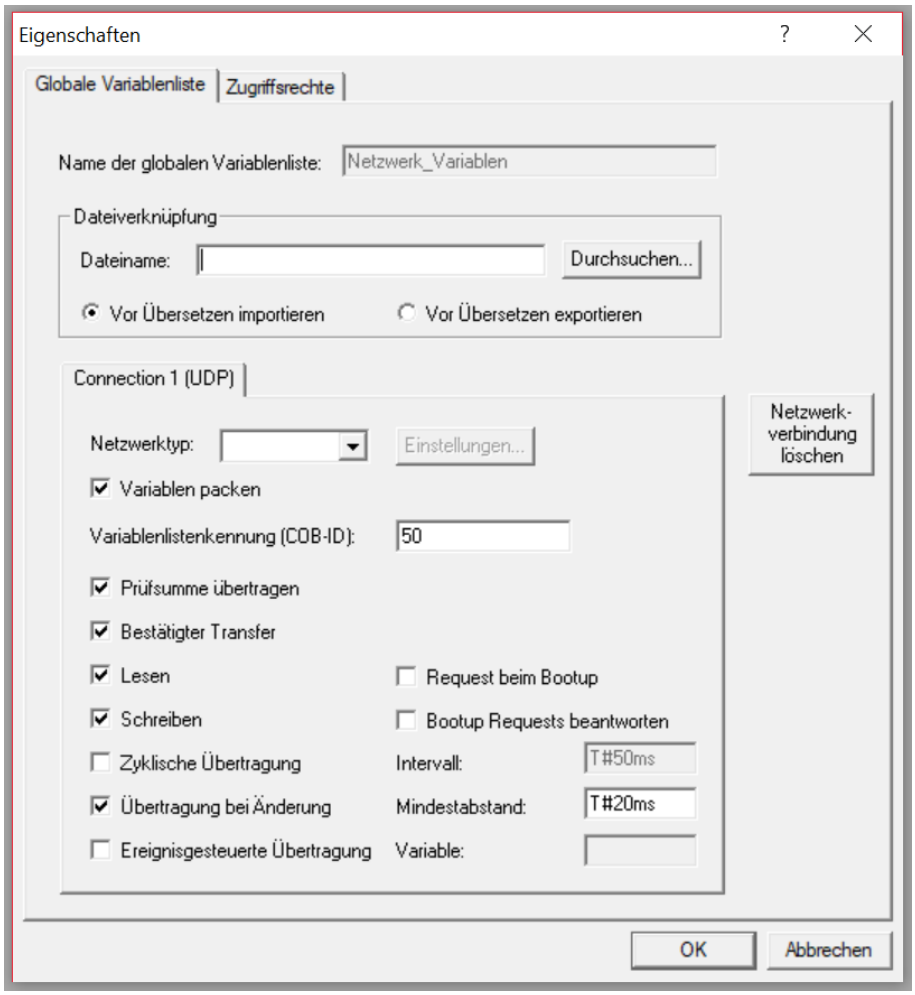


Abbildung 4.8.: Einstellung der Netzwerkvariablen

hat. Zur Absicherung sollen die Werte der Variablen zusätzlich in einem definierten Zeitabstand übertragen werden. Hier gibt CODESYS ein voreingestelltes Zeitintervall von 20 ms an, welches so beibehalten wurde. Ebenso ist es möglich, einzustellen, ob eine Prüfsumme beim Empfangen der Daten übertragen wird. Diese Punkte wurden zur Absicherung einer möglichst sicheren Kommunikation aktiviert. Damit ist die Konfiguration der Netzwerkvariablen abgeschlossen.

Wenn alle Einstellungen vorgenommen wurden, können die eigentlichen, benötigten Variablen definiert werden. Je nachdem, zwischen welchen Einzelstationen die Kommunikation stattfinden soll, können mehrere Netzwerkvariablenlisten erstellt werden.

So wird es im Fall der Bandanlage eine größere Liste geben, welche die benötigten Variablen enthalten, die für den prinzipiellen Ablauf der Steuerung verantwortlich sind und von allen Teilnehmern genutzt werden. Diese ist im Tabelle A.2 erläutert. Daneben gibt es kleinere Listen, deren Inhalt nicht für alle Stationen relevant sind und nur die Kommunikation zwischen zwei bis drei einzelnen Stationen abdecken. Wichtig ist, dass die Variablen innerhalb einer Liste den gleichen Namen erhalten. Um die Netzwerkvariablen besser von den normalen Variablen im Programmcode unterscheiden zu können, wurden diese alle mit dem Präfix `net_` versehen.

4.3. Programmierung der Anlage

Bei der Programmierung ohne QR-Codescanner soll die Steuerung der Bandanlage dem in Abschnitt 3.2 beschriebenen Ablauf folgen. Durch die erprobte Funktionstüchtigkeit mit dem bisherigen Aufbau der Anlage ist es später einfacher, das Zusatzelement des QR-Codescanners in das Programm einzufügen. Um die einzelnen Funktionen der Anlage jedoch nutzen zu können, musste zunächst herausgefunden werden, wie die Ein- und Ausgänge der jeweiligen *FESTO*-Module belegt sind. Grundlegend funktioniert die Zuweisung der Ein- und Ausgänge über Bits. Ein Eingangsmodul mit 16 digitalen Eingängen stellt für jeden Eingang ein Bit zur Verfügung. Jedes einzelne Bit besitzt eine eigene Adresse und kann darüber angesteuert werden. Um deren derzeitige Belegung herauszufinden, wurden Testprogramme auf die Steuerung geladen. Durch gezieltes An- und Ausschalten der einzelnen Bits im Programm konnte die Zuweisung dieser ermittelt werden. Das Ergebnis ist im Anhang B ersichtlich. Dort sind die Ein- und Ausgänge schematisch dargestellt und ihre Funktion sowie deren Adresse zur Ansteuerung im Code aufgelistet.

Für die Steuerung der Ventilinseln stehen insgesamt 32 Bits zur Verfügung. Im Normalfall gilt bei der Belegung von in diesem Fall 32 Bits, dass es 2^32 Möglichkeiten der Kombination dieser gibt. Allerdings ist in diesem Fall jedes einzelne Bit für die Ansteuerung eines Ventils zuständig. Das heißt, ein Ventil wird geöffnet, wenn das Bit den Wert 1 annimmt, für den Wert 0 ist es geschlossen. Kombinationen werden dabei nicht berücksichtigt, daher stehen hierbei exakt 32 Möglichkeiten zur Belegung der

Bits zur Verfügung. Für die Funktionalität der Bandanlage wurden für jede Station maximal sechs Bits benötigt. In der Tabelle A.1 im Anhang sind die Bitadressen von den Ventilinseln der einzelnen Stationen mit ihrer jeweiligen Funktion dargestellt.

Anhand der gesammelten Informationen über die Belegung der Ein- und Ausgänge der Stationen können die einzelnen Programme erstellt werden. Ebenso ist es möglich, jedem Ein- beziehungsweise Ausgang direkt in der Steuerungskonfiguration Namen zuzuweisen, über welche diese im späteren Programm angesprochen werden können. So wurden für diesen Zweck Namen vergeben, die gleich darauf schließen lassen, welche Funktion dem jeweiligen Ein- oder Ausgang zugeordnet ist.

4.3.1. Programmablauf am Magazin

Am Magazin befinden sich die Schalter für Start, Stopp, Notaus sowie Richten. Ebenso können die Lampen für den jeweiligen Zustand der Anlage und ein akustisches Signal angesteuert werden. Es gibt zwei verschiedene Situationen, in denen das Magazin ein Werkstück auf das Band schieben darf. Zum einen, wenn der Startschalter erstmalig zum Starten der gesamten Anlage gedrückt wird. Zum anderen muss nach dem erfolgreichen Einsortieren eines Werkstückes das nächste vom Magazin bereitgestellt werden. Hierfür liefert die Netzwerkvariable `net_TB` den Wert `TRUE`, sollte dies der Fall sein. Erst dann darf ein neues Werkstück auf das Band geschoben werden.

Ebenso muss darauf geachtet werden, dass nach dem Anhalten der Anlage mittels der Stopptaste beim erneuten Start kein neues Werkstück bereitgestellt wird. Das heißt, die Funktion des Ausfahrens am Magazin darf nicht aktiviert werden, sollte sich ein Werkstück während des Haltevorgangs noch im Prozess befinden. Hierfür wurden mehrere Netzwerkvariablen angelegt, welche die Position des Werkstückes aufzeigen. Sollte eine davon den Wert `TRUE` besitzen, darf beim Drücken der Starttaste am Magazin kein neues Werkstück ausgefahren werden.

Je nach Zustand der Anlage werden an dieser Station ebenfalls die verschiedenfarbigen Lampen an- oder ausgeschaltet. Eine Übersicht der Bedeutung der verschiedenen Lampenfarben ist in Tabelle 4.2 ersichtlich.

Farbe(n)	Bedeutung
Grün	Anlage befindet sich im Normalbetrieb
Grün + Gelb	Anlage wurde angehalten
Gelb	Anlage befindet sich im Richtenprozess
Rot	Notaus wurde betätigt, Anlage steht

Tabelle 4.2.: Bedeutung der Farbkombinationen der Lampen

Der Status der Lampen ist an das Drücken der jeweiligen Schalter gebunden. Eine Ausnahme bildet hierbei das Richten. Die gelbe Lampe wird ausgestellt, sobald die Anlage fertig ist mit dem Richten. Hierfür wird die Netzwerkvariable `net_Richten` abgefragt. Die gelbe Lampe leuchtet so lange, wie die Variable den Wert `TRUE` hat, andernfalls ist sie aus. `net_Richten` wird dann `FALSE`, wenn die letzte Station erfolgreich gerichtet wurde und die Anlage wieder gestartet werden kann.

4.3.2. Programmablauf am Band 1

Das Band 1 wird dann gestartet, wenn die Netzwerkvariable `net_TAAB1` vom Magazin auf `TRUE` gesetzt wurde. Die Variable signalisiert, dass sich ein Werkstück am Anfang vom ersten Band befindet. Sobald einer der Sensoren das jeweilige Werkstück anhand von Material oder Farbe erkennt, wird das Band gestoppt und das Objekt mittels eines Schiebers in das Lager geschoben.

Eingeordnet werden auf Band 1 jedoch nur das jeweils erste und dritte Werkstück seiner Art so wie es in Abschnitt 3.2 definiert wurde. Daher werden Counter benötigt, welche die Anzahl der bereits erkannten Objekte zählt. Beim Auslösen einer der Sensoren wird zunächst der Status des Counters abgefragt. Danach wird entschieden, ob das Werkstück eingeordnet wird oder nicht. Wichtig ist, dass nach Auslösen eines Sensors alle anderen sozusagen in einen Stillstand versetzt werden, bis das nächste Werkstück bearbeitet werden muss. Der Grund hierfür ist der Sensor am Lager für schwarze Werkstücke. Dieser ist nicht nur in der Lage, den schwarzen Kunststoff zu erkennen, sondern ebenfalls alle anderen Materialien und Farben. Würde beispielsweise bei einem weißen Werkstück der Sensor am entsprechenden

Lager auslösen, die anderen jedoch aktiv bleiben, würde der schwarze Sensor dieses ebenfalls erkennen und den Counter für schwarze Werkstücke erhöhen. Hierfür wurde die Variable `Ausgelöst` eingeführt. Diese erhält den Wert `TRUE`, sobald einer der Sensoren ein Werkstück erkannt und erfolgreich bearbeitet hat. Dabei ist es unerheblich, ob dieses eingelagert oder weiter geschickt wurde.

Wurde ein Objekt in ein Lager einsortiert, wird die Netzwerkvariable `net_TB` auf `TRUE` gesetzt und das Magazin darf ein neues Werkstück auf das Band schieben. Wird das Objekt nicht eingelagert, fährt dieses bis zum Ende des Bands durch. Dort befindet sich eine Lichtschranke, bei deren Aktivierung das Band angehalten wird. In diesem Fall wird die Netzwerkvariable `net_TAEB1` auf `TRUE` gesetzt, welche für den Umsetzer das Signal zum Starten ist.

Bei jeder Einlagerung eines Werkstückes wird, abhängig davon, das wievielte es war, eine Netzwerkvariable gesetzt, welche in der Visualisierung für die Lagerfüllstände zuständig ist.

4.3.3. Programmablauf am Umsetzer

Befindet sich ein Teil am Ende von Band 1 und ist die Variable `net_TAEB1 TRUE`, darf der Umsetzer seinen Programmablauf starten. Dieser umfasst das Greifen des Werkstückes, das Übersetzen auf Band 2 und das Zurückfahren in die Ausgangsposition, welche sich über dem Ende von Band 1 befindet. War das Umsetzen des Werkstückes erfolgreich, wird die Netzwerkvariable `net_TAAB2` wahr, welche dem zweiten Band signalisiert, dass sich auf deren Anfang ein Objekt befindet.

4.3.4. Programmablauf am Band 2

Sobald `net_TAAB2` auf `TRUE` gesetzt wurde, wird das Band gestartet. Den einzigen Unterschied zum ersten Band stellt der Wender dar, welcher sich vor den Lagern befindet. Seine Funktionsweise ist im Abschnitt 4.3.5 beschrieben. Die Einlagerung der Werkstücke funktioniert prinzipiell wie beim ersten Band (siehe Abschnitt 4.3.2). Auch

hier wird ein Counter benötigt, jedoch nicht, um einzelne Werkstücke durchfahren zu lassen, sondern um zu überprüfen, ob ein Lager voll ist. Sollte dies für eine Werkstückart der Fall sein, wird das betreffende Werkstück bis zum Ende von Band 2 gefahren. Dort befindet sich ebenfalls eine Lichtschranke, nach deren Auslösen das Band angehalten wird. Tritt dieser Fall ein, wird die Anlage gestoppt.

Nun müssen zunächst die Lager manuell leer geräumt und die Anlage mittels der Starttaste neu gestartet werden. Auch die Counter an beiden Bändern müssen in diesem Fall zurückgesetzt werden. Es ist möglich, dies automatisch zu machen, sobald ein Werkstück am Ende des zweiten Bands angekommen ist. Da jedoch für die Fertigstellung eine rechnergestützte Visualisierung gewünscht war, wurde das Zurücksetzen der Counter mithilfe einer Schaltfläche in dieser realisiert (siehe Abschnitt 6.1).

4.3.5. Programmablauf am Wender

Sobald das Werkstück auf Band 2 die Lichtschranke am Wender passiert, wird dessen Programmablauf gestartet. Da der Wender schnell genug nach unten fährt, war es nicht nötig, für den Wendeprozess das Band extra anzuhalten. Der Wender dreht das Werkstück wie in Abschnitt 3.1.4 beschrieben. Nachdem das Werkstück wieder auf dem Band abgesetzt wurde, soll sich zusätzlich der Greifer in seine Ausgangsposition zurück drehen. Damit dieser dabei nicht mit dem gerade abgesetzten Werkstück kollidiert, wurde eine Wartezeit von zwei Sekunden nach dem Absetzen und Hochfahren des Wenders eingefügt. Wenn die Zeit abgelaufen ist, dreht sich der Greifer zurück.

Ein weiteres Argument dafür, das Band während des Wendeprozesses weiter fahren zu lassen war, dass das Anhalten des Bands mit dem verwendeten Aufbau nur sehr unzuverlässig umsetzbar ist. Die Verzögerungen zwischen dem Erkennen des Werkstückes und dem darauf folgenden Anhalten des Bandes waren teilweise sehr verschieden. Es konnte auch nach mehreren Versuchen keine allgemein gültige Verzögerungszeit festgestellt werden. Mal fuhr das Werkstück zu weit, dann wieder nicht weit genug, um vom Greifer umschlossen zu werden. Die sofortige Aktivierung des Greifers bei

fahrendem Band funktionierte jedoch sehr zuverlässig, weshalb sich für diese Variante entschieden wurde.

4.3.6. Notaus und Richten

Beim Drücken des Schalters für den Notaus muss die komplette Anlage sofort stoppen. Dabei ist es egal, welcher Prozess in diesem Moment ausgeführt wurde. Auch darf es danach nicht möglich sein, die Anlage mittels der Starttaste wieder in Betrieb zu nehmen. Das Betätigen des Notaus macht das Richten notwendig. Hierbei soll die Anlage leer gefahren werden, um sicherzustellen, dass sich kein Werkstück mehr in der Anlage befindet. Sobald der Notaus betätigt wurde, nimmt die Netzwerkvariable `net_Richten` den Wert `TRUE` an. Solange dies der Fall ist, kann keine der Stationen ihre normalen Abläufe ausführen.

Beim Drücken des Schalters für Richten erhält zunächst das Magazin das Signal zum Richten. Falls der Schieber im Moment des Notaus gerade ausgefahren war, wird dieser wieder eingefahren. Sobald das Magazin fertig ist, gibt es der nächsten Station, also Band 1, ebenfalls das Signal, zu richten. Auch hier wird ein etwaig ausgefahrener Schieber eingefahren. Sollten sich alle Schieber in Ausgangsposition befinden, wird das Band für eine Zeitspanne von zehn Sekunden gestartet. Passiert hierbei ein Werkstück die Lichtschranke am Ende des Bands, wird dieses angehalten. Andernfalls wird das Band nach Ablauf der Zeit ebenfalls gestoppt. Das Richten endet am Band 1 entweder nach dem Einfahren eines zuvor ausgefahrenen Schiebers oder nach dem Fahren des Bands.

Erst danach darf der Umsetzer richten. Dieser beendet dabei entweder einen bereits begonnen Prozess, das heißt, wenn sich ein Objekt zum Zeitpunkt des Notaus gerade beim Umsetzen befand, oder er setzt ein am Ende des ersten Bands befindliches Objekt wie auch im normalen Ablauf auf das zweite Band um. Nachdem der Umsetzer fertig ist, wird dem Wender die Freigabe zum Richten erteilt. Sollte sich noch ein Werkstück im Wendeprozess befinden, wird dieser abgeschlossen und das Werkstück auf das zweite Band gesetzt. Andernfalls macht der Wender nichts und gibt das Signal zum Richten an das zweite Band. Dieses arbeitet dann wie auch das erste

Band, es kann ein ausgefahrener Schieber eingefahren oder das Band laufen gelassen werden. Bei letzterem endet das Richten dann, wenn entweder ein Werkstück das Ende von Band 2 erreicht oder die Fahrzeit von ebenfalls zehn Sekunden abgelaufen ist.

Sobald das Band 2 fertig gerichtet ist, wird `net_Richten` auf `FALSE` gesetzt. Nun ist es wieder möglich, die Anlage mittels des Startschalters in Betrieb zu nehmen. Es sollte dabei beachtet werden, dass das Richten gleichbedeutend mit dem Leerräumen der Anlage ist. Die Counter der Lager werden nach einem Notaus zurückgesetzt, also müssen diese auch frei geräumt werden.

4.3.7. Aufgetretene Probleme

Während der Fertigstellung der grundlegenden Programmierung fiel auf, dass das Anhalten beider Bänder, wenn ein Werkstück an deren Ende angelangt, zeitverzögert geschieht. Die Lichtschranke am Ende der Bänder ist so ausgerichtet, dass das dort befindliche Objekt, im Falle des ersten Bands, ideal vom Umsetzer gegriffen werden kann. Aufgrund der Verzögerung setzte der Umsetzer mit seiner Greifzange jedoch auf dem Objekt auf, statt dieses zu umschließen.

Zunächst wurde überprüft, ob das Problem innerhalb des Programms auf der SPS liegt. Die Kommunikation mit der Lichtschranke erfolgt komplett intern auf der SPS, das Eingangssignal für diese ist am gleichen Modul angeschlossen, welches das Ausgangssignal zum Anhalten des Bands liefert. Daher sollte die Zeit, welche die SPS zum Abarbeiten des Hauptprogramms benötigt, nicht der ausschlaggebende Punkt für die Verzögerung sein. Zum Absichern dieser Theorie wurde ein kleines Testprogramm geschrieben und geladen, welches nur in der Lage ist, das Band zu starten und beim Erkennen eines Objekts an der Lichtschranke dieses wieder anzuhalten. Jedoch trat auch hier der gleiche Fehler auf.

Beim Beobachten der Leuchtdioden, welche den Status der Ein- und Ausgänge des FESTO-Moduls anzeigen, fiel auf, dass die Lichtschranke zwar sofort schaltet, sobald ein Objekt sie berührt. Jedoch wird die Leuchtdiode, die den Status des

Motors anzeigt, erst einige Zeit später ausgeschaltet. Hier wird deutlich, wie groß die auftretende Verzögerung sein muss, wenn sie für das menschliche Auge problemlos erkennbar ist. Das Problem softwareseitig zu lösen, war auch nach mehreren Tests nicht möglich. Daher blieb der Schluss, dass die Hardware für die Verzögerung verantwortlich ist. Es ist jedoch nicht von einem direkten Defekt auszugehen, da der Fehler bei beiden Bändern in gleicher Form auftritt und daher angenommen werden kann, dass dieser Zustand sozusagen „normal“ ist.

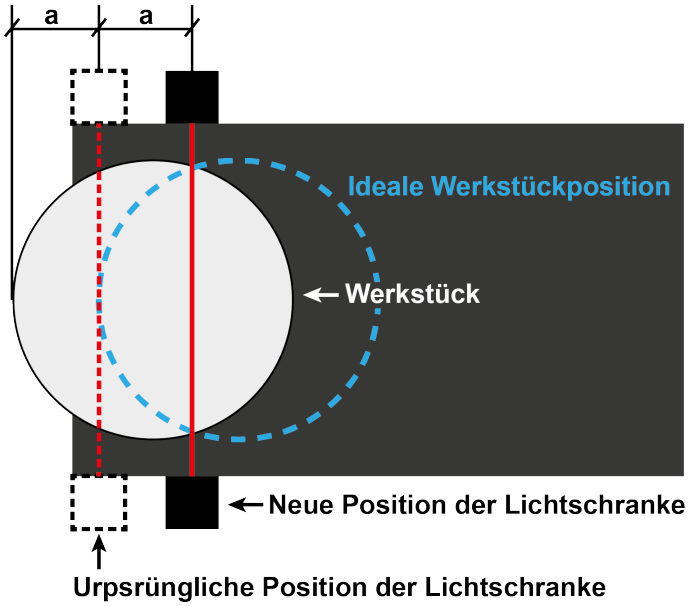


Abbildung 4.9.: Verschiebung der Lichtschranke an Band 1

Damit die Anlage trotzdem automatisiert laufen kann, wurde letztendlich die Lichtschranke am Ende von Band 1, wie in Abbildung 4.9 zu sehen, um das gleiche Maß weiter nach vorn versetzt, welches sich das Werkstück zuvor zu weit hinten befand. Am Aufbau von Band 2 wurde nichts geändert, da die genaue Endposition des Werkstücks an dessen Ende keinen Einfluss auf die Funktionalität der Anlage hat.

5. Einbinden des QR-Codescanners

Der für die Anlage gewählte Scanner ist ein Multicode Reader der Firma ifm. Das vorhandene Modell O2I ist in der Lage, ein- und zweidimensionale Codes, unabhängig von deren Ausrichtung, zu erkennen. Als Controller, beziehungsweise Master-SPS, dient eine SPS von FESTO. Im Zuge dieser Arbeit soll das Gerät zweidimensionale Codes, also QR-Codes, auslesen. Ziel ist es, dass das Lesegerät automatisiert mit den anderen Programmteilen arbeitet und die gelesenen Informationen aus dem QR-Code an diese weitergibt. Hierfür müssen zunächst diverse Einstellungen getroffen und das passende Programm erstellt werden.

5.1. Konfiguration des Scanners

Zu Beginn muss eine grundlegende Verbindung zum Multicode Reader hergestellt werden. Dies geschieht im von ifm mitgelieferten Programm *Dualis Multicode*. Nach Starten des Programms wird in der oberen Leiste der Reiter Verbindungen und anschließend IP-Adresse ausgewählt. Es öffnet sich das in Abbildung 5.1 dargestellte Fenster. Hier kann nun die IP-Adresse des Geräts eingegeben werden.

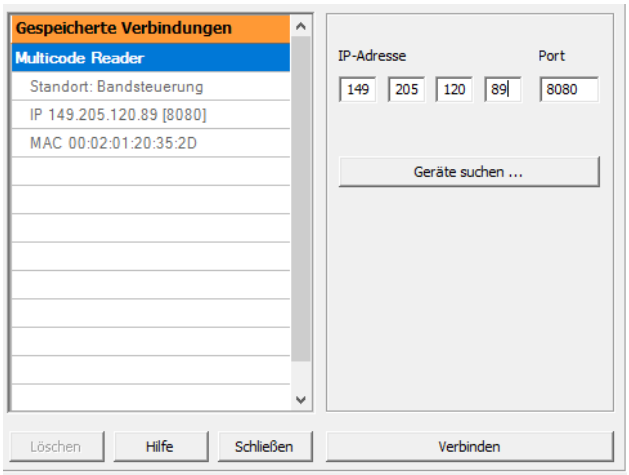


Abbildung 5.1.: Eingabe der IP-Adresse des Multicode Readers

Diese lautet 149.205.120.89. Nach der Eingabe wird auf die darunter liegende Schaltfläche Verbinden gedrückt. In dem sich dabei öffnenden Fenster müssen keine Einstellungen vorgenommen werden.

Im sich links befindlichen Menü wird die Schaltfläche *Konfiguration* betätigt. Das sich daraufhin öffnende Fenster enthält eine Liste wie in Abbildung 5.2 dargestellt.

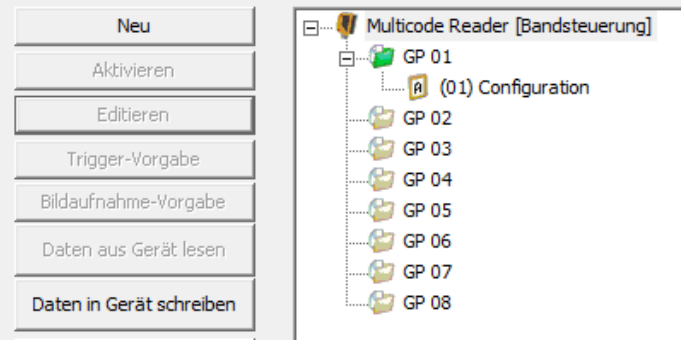


Abbildung 5.2.: Konfigurationsliste für den Multicode Reader

Dort gibt es zunächst mehrere Gruppen. Mit einem Rechtsklick und anschließend auf *Neu* kann eine neue Konfiguration erstellt werden. Hiernach müssen außerdem die *Globalen Geräteeinstellungen* angepasst werden. Dort ist es nötig, die *Netzwerk-Parameter* ,wie in Abbildung 5.3 zu sehen, einzustellen. Ebenso sollte überprüft werden, ob in den Einstellungen für die *Prozess-Schnittstelle* und dort unter *Erweiterte Einstellungen* der TCP/IP-Port auf 50010 gesetzt ist. Sind diese Konfigurationen abgeschlossen, kann die erstellte Konfiguration ausgewählt und links davon auf *Editieren* gedrückt werden.

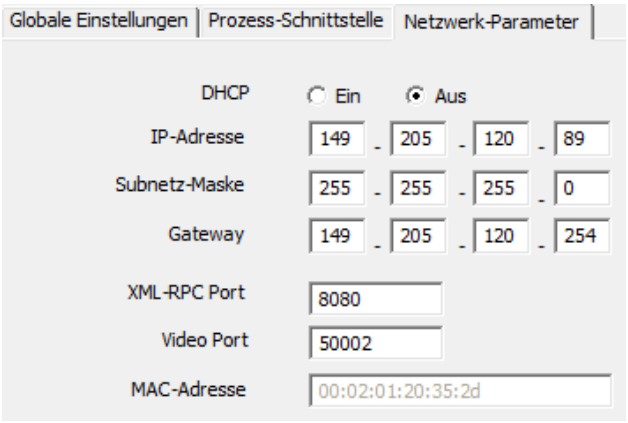


Abbildung 5.3.: Globale Geräteeinstellungen für den Multicode Reader

Das sich nun öffnende Fenster enthält alle wichtigen Einstellungen zur Anpassung des Lesens von Codes. Unter *Code definieren* wird eine Übersicht der benötigten Einstellungen sowie das Bild, welches die Kamera aufgenommen hat, angezeigt. Die Einstellungen sollten, wie in Abbildung 5.4 gezeigt, vorgenommen werden. Wird nun ein Werkstück unter das Lesegerät gelegt und anschließend auf *Code lesen* gedrückt, wird das angezeigte Bild aktualisiert und, sofern alle Einstellungen richtig vorgenommen wurden, das Ergebnis des Scans angezeigt. In diesem Fall wurde die Funktionalität mit einem Werkstück aus Aluminium getestet, dessen aufgebrachter Code den Text

„Aluminium“ enthält. Rechts neben der Bildanzeige ist erkennbar, dass der QR-Code fehlerfrei gelesen wurde.

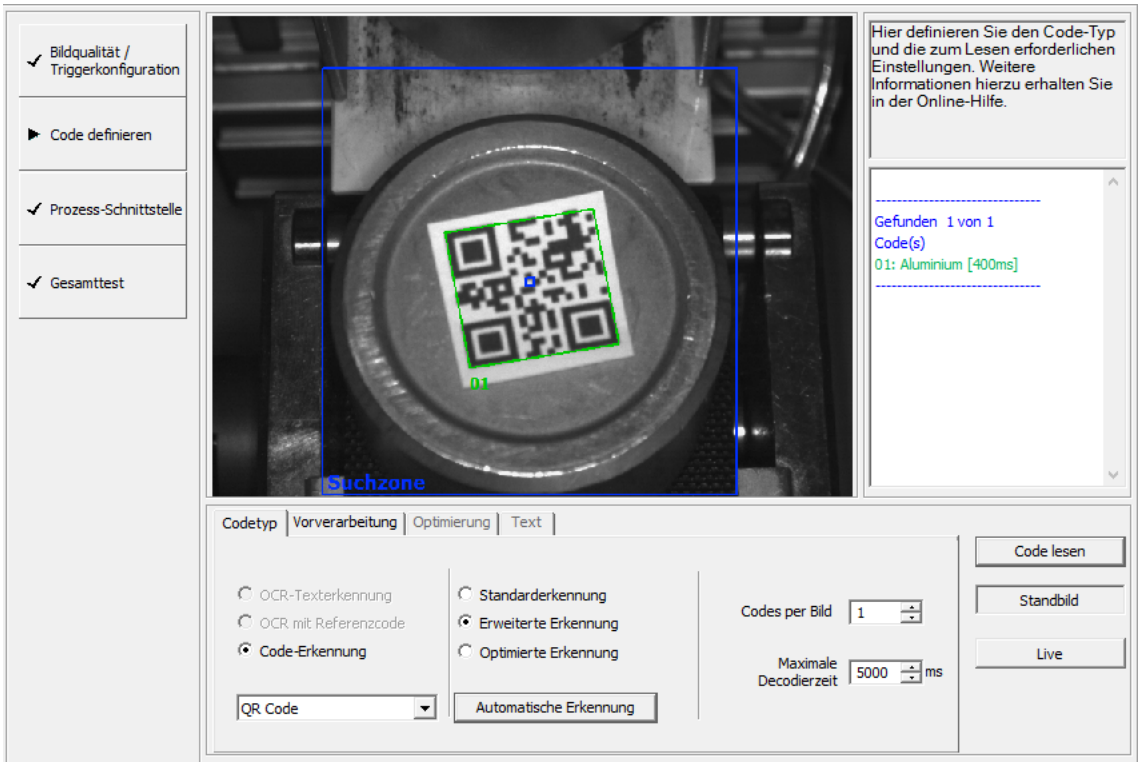


Abbildung 5.4.: Einstellungen zum Lesen des Codes

Nachdem die Kommunikation mit dem Multicode Reader erfolgreich konfiguriert ist, muss nun das benötigte Programm erstellt werden, um automatisch auf die Lesefunktion des Geräts zugreifen zu können.

5.2. Beispielprogramm von ifm

Für den verwendeten Multicode Reader existiert ein Beispielprogramm von der Firma ifm, welches in CODESYS V3.5 erstellt wurde. Das Programm ist darauf ausgelegt, sich mit einem ebenfalls von ifm hergestellten Controller zu verbinden und neben dem Scannen eines Codes diversen anderen Anforderungen zu genügen. Da die Kommunikation mit dem Lesegerät in diesem Programm bereits abgedeckt ist, wird

es im Folgenden weiter verwendet und auf das Zusammenspiel mit der Bandanlage angepasst.

5.2.1. Konfigurieren des Beispielprogramms

Das Beispielprogramm von ifm enthält bereits die benötigten Aspekte, welche zur Realisierung des Scanvorgangs im Betrieb der Anlage gewünscht sind. Problematisch hierbei ist jedoch, dass das Programm um ein Vielfaches mehr kann, als benötigt wird. Daher muss es zunächst grundlegend angepasst werden. Prinzipiell soll es nur möglich sein, dem Scanner das Signal zum Einlesen des QR-Codes vom Magazin aus zu senden, sobald sich ein Werkstück an der dafür vorgesehenen Position befindet und das Ergebnis des Auslesens wieder zurück an das Magazin zu senden.

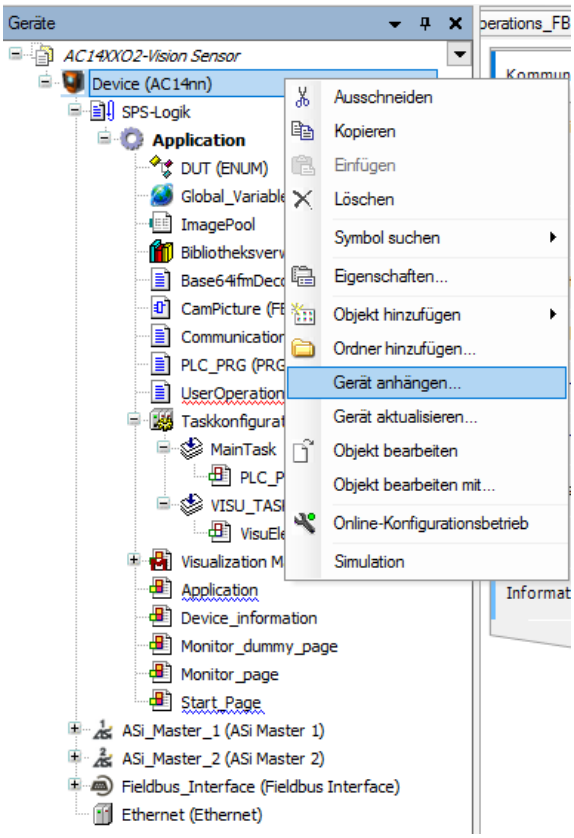


Abbildung 5.5.: Ersetzen des Controllers im Beispielprogramm

Um das Programm überhaupt verwenden zu können, müssen zunächst die verwendeten Geräte angepasst werden. Der QR-Codescanner wird, genau wie alle anderen Stationen der Anlage, mittels einer SPS angesteuert. Als Standardgerät ist hier jedoch noch der AC14-Controller von ifm eingestellt. Um dies zu ändern, wird, wie in Abbildung 5.5 gezeigt, anhand eines Rechtsklicks auf den Controller und anschließend *Gerät aktualisieren* das Menü der vorhandenen Geräte aufgerufen. Aus der dortigen Liste wird das Gerät *CPX-CEC-C1-V3* ausgewählt. Dieses stellt die Master-SPS der Firma FESTO dar. Hardwareseitig ist die gewählte SPS über Ethernet mit dem Multicode Reader verbunden. Beim

Aufspielen eines Programms auf die Steuerung kann diese, bei richtiger Konfiguration des Programms, mit dem Lesegerät kommunizieren. Anschließend kann das Beispielprogramm noch vor der Bearbeitung des eigentlichen Quellcodes weitestgehend bereinigt werden. Das Programm enthält beispielsweise eine komplett eigene Visualisierung mit umfangreichen Funktionen. Da die Visualisierung jedoch allein über das Magazin laufen soll, kann die im Programm verwendete vollständig entfernt werden. Einzig der Teil der Visualisierung, welche das Ergebnis des Scanvorgangs anzeigt, bleibt vorerst erhalten. Dort sind wichtige Informationen zur Funktionsweise des Programms enthalten, welche später zur Anpassung des Quellcodes benötigt werden.

5.2.2. Anpassung des Programmcodes

Es musste zunächst herausgefunden werden, wie das mitgelieferte Programm funktioniert und wo das Ergebnis des Scanvorgangs gespeichert wird. Am einfachsten kann dies über die im Programm enthaltene Visualisierung herausgefunden werden. In Abbildung 5.6 ist ein Ausschnitt aus der als *Monitor_page* bezeichneten Visualisierung zu sehen. Dort wird bei laufendem Programm das Triggersignal zum Lesen eines Codes an den Scanner gesendet. Unter *Result* wird laut Annahme das Ergebnis des vom Scanner ausgelesenen Strings angezeigt. Die Bezeichnung `%s` bedeutet, dass in der Visualisierung auf eine bestimmte Variabele zugegriffen und an dieser Stelle angezeigt werden soll, in diesem Fall ein `String` (s). Dahinter steht die Variable `STRResultTriggerData`.

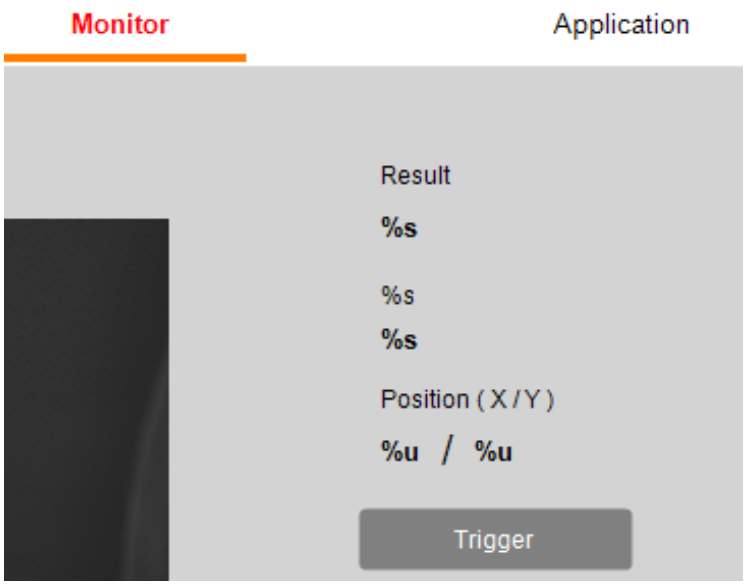


Abbildung 5.6.: Ausschnitt der alten Visualisierung

Nun muss die Variable zurückverfolgt werden. Für die komplette Abarbeitung des Scanvorgangs ist der Funktionsbaustein *UserOperations_FB* zuständig. Laut der dortigen Variablendeklaration wird, wie in Abbildung 5.7 zu sehen, die Variable `STRResultTriggerData` für das Abspeichern des Scanergebnisses verwendet, jedoch nicht für die eigentlich vom Scanner erhaltenen Daten.

```
STRResultTriggerData: STRING; (*Stores data to be displayed under "Result" field on Monitor page,
                               after Trigger command*)
```

Abbildung 5.7.: Beschreibung der Variable `STRResultTriggerData`

Die Vermutung, dass dies der eingelesene String ist, hat sich damit zwar nicht bewahrheitet, jedoch taucht bei der Variablendeklaration auch die Variable `STRTriggerData` auf. Laut des erklärenden Kommentars dazu stellt diese den Datenstring dar, der die Antwort auf das Triggersignal des Scanners liefert. Daher wird nun das Programm explizit nach dieser Variable abgesucht, um herauszufinden, wo sie verwendet wird und letztendlich das Ergebnis ausgegeben wird.

Nach Verschaffen eines Überblicks über den Programmcode wird deutlich, dass das Programm prinzipiell folgendes macht: Es erhält ein Triggersignal aus der Visualisierung, anhand derer der eigentliche Scanvorgang gestartet wird. Nun wird jedoch nicht nur gescannt, sondern unter anderem ermittelt, welcher Scanner verwendet wird, welches Protokoll darauf liegt und welche Funktion der Benutzer in diesem Moment verwenden möchte. Diese Dinge sind für den gewünschten Gebrauch des Scanners jedoch überflüssig. Das Programm soll einzig dazu in der Lage sein, vom Magazin ein Startsignal zum Scannen zur erhalten, damit den Scanvorgang zu starten und anschließend das Ergebnis dessen an das Magazin zurück zu schicken. Daher muss nun zum einen das geschriebene Programm verstanden und danach aussortiert werden.

Funktionsbausteine im Programm
Im Beispielprogramm wird auf mehrere Funktionsbausteine zurückgegriffen. Einige davon sind für das grundlegende Funktionieren des Scanners verantwortlich und müssen daher so erhalten bleiben, wie sie sind. Vor allem der Baustein *Communication_FB* ist wichtig, da er die Verbindung zum Scanner aufbaut. Der für den Benutzer wichtige Teil spielt sich komplett im *UserOperations_FB* ab. Daher wird auch nur dieser in den nächsten Schritten überarbeitet. Die Bausteine *CamPicture* und *Base64ifmDecode* sind für die Funktionalität nicht notwendig und wurden daher gelöscht.

Zunächst war es wichtig, den Weg der Variable `STRTriggerData` zu verfolgen, um alle wichtigen Schritte bis zur Speicherung des Ergebnisses vorliegen zu haben. Die Ereignisse im Programm laufen zu einem großten Teil in einer `CASE`-Struktur ab, bei der eine Variable (`Istm`) vom Typ `INT` abgefragt wird. In der Kommentierung zur `CASE`-Struktur steht frei aus dem Englischen übersetzt geschrieben: „Case 50x: In diesem Bereich (500 bis 560) erhält der Sensor ein Triggersignal, wenn in der Visualisierung der Triggerschalter gedrückt wird.“ Das Geschehen während der restlichen Zustände von `Istm` ist vorerst uninteressant. Bei einem Blick in die Bereiche in denen Istm den Wert 500 bis 560 annimmt, fällt vor allem der Zustand 520 auf. Dort wird der vom Scanner gelesene Wert in eine temporäre Variable (`STRParseString`) geschrieben. Anschließend wird der so erhaltene String auf Fehler überprüft und im Falle eines nicht lesbaren Codes eine Fehlermeldung in die auf der Visualisierung ausgegebene Variable geschrieben. An dieser Stelle wird zudem aufgeschlüsselt, in welchem Format der String vom Scanner an das Programm übermittelt wird. Die Erklärung hierzu ist in Abbildung 5.8 zu sehen.

```
(*Result format after t command is given below*)
(*'0002startA13579B0203;0418;stop                    '*)
(*<start><deooded data><x postion> <;> <Y postion> <;> <stop>*)
```

Abbildung 5.8.: Format des vom Scanner eingelesenen Strings

Der Teil, welcher ausgewertet werden muss, ist *<decoded data>*. Dieser Part enthält den im QR-Code hinterlegten Text. Somit ist geklärt, an welcher Stelle im Programm die Daten des Codes verarbeitet werden. Dieser Abschnitt muss funktionsfähig erhalten bleiben und wird später für eigene Zwecke genutzt. Es muss jedoch noch geklärt werden, unter welchen Umständen die `CASE`-Anweisung an diese Stelle springt. Dazu wird noch einmal auf den Anfang geschaut.

Der Ausgangszustand von `Istm` ist „0“. In diesem Fall wird zunächst der Verbindungsstatus des Scanners abgefragt. Falls der in Ordnung ist, werden erneut Abfragen gestartet. Bei einer davon wird an die Stelle 500 gesprungen. Die Anweisung sieht im Original zunächst aus wie in Abbildung 5.9. Allerdings ist die einzig interessante Variable, welche dort abgefragt wird, `XeXecuteTrigger`. Diese wird dann `WAHR`, falls das Triggersignal aus der Visualisierung betätigt wurde.

```
ELSIF XexecuteTrigger AND NOT XexecuteChApp AND NOT XexecuteStatistics AND xInitDone THEN
    Istm := 500; // Trigger command is given
    Twait(In:= FALSE);
```

Abbildung 5.9.: Bedingung zum Starten des Scanvorgangs

Zusammenfassend sehen die Erkenntnisse nun also folgendermaßen aus:

1. Die Speicherung des benötigten Strings erfolgt in der Variable `STRTriggerData`
2. Die Speicherung selbst erfolgt im 500er Bereich der `CASE`-Anweisung, welche über die Variable `Istm` gesteuert wird
3. Um `Istm` auf 500 zu setzen, muss im Ausgangszustand die richtige Bedingung wahr sein
4. Wann die Bedingung erfüllt wird, wird vor der `CASE`-Anweisung durch das Setzen einer Variable bestimmt

Nach diesem Prinzip lässt sich nun das gesamte Programm neu konzipieren. Durch viel Ausprobieren wurde getestet, ob das Programm nach der Entfernung einiger Teile trotzdem noch funktionsfähig ist. Insgesamt wurde der *UserOperations_FB* von zuvor 2170 Zeilen Programmcode auf 277 gekürzt.

In der neuen Variante des Programms wird zunächst auf ein Triggersignal gewartet. Dieses kommt dann letztendlich vom Magazin, wenn das Werkstück zum Scannen bereitgestellt wurde. Wird das Signal `TRUE`, wird `XeXecuteTrigger` ebenfalls `TRUE`. Damit erfüllt sich die Bedingung im Ausgangszustand der `CASE`-Anweisung, da hier nur noch auf die Triggervariable gewartet wird, und `Istm` wird auf 500 gesetzt. In den Schritten 500 und 510 wird ein Lese- und Schreibfunktionsblock aufgerufen. Diese waren vorgefertigt und stellen die Funktionalität des Lesens vom Scanner her. Sind diese Schritte abgeschlossen, nimmt `Istm` den Wert 520 an. Hier wird nun, wie auch zuvor, der gelesene Code auf Fehler geprüft. Nach dem gleichen Prinzip, wie auch nach Fehlern gesucht wird, wurde nun eigener Code hinzugefügt, welcher den gelesenen String auf den Inhalt „Alu“, „Black“ oder „White“ überprüft. Wird einer dieser Inhalte gefunden, so werden die entsprechenden Variablen, je nach Farbe, auf `TRUE` gesetzt und später an das Magazin weiter gegeben.

Das Programm wurde auf Funktionalität geprüft, indem eine kleine Visualisierung eingefügt wurde, in der manuell ein Triggersignal zum Scannen gegeben werden konnte. Zudem wurde das Ergebnis des Scans dort ausgegeben. Nachdem dies fehlerfrei funktionierte, musste das Programm noch in die restliche Bandanlage integriert werden.

5.3. Auslesen der QR-Codes

Da das Auslesen der Codes nach der Anpassung des Programms jetzt funktioniert, konnte sich mit den verwendeten QR-Codes beschäftigt werden. Zum Zeitpunkt der Bearbeitung des Beispielprogramms gab es je ein Werkstück jeden Materials, welches einen QR-Code besaß. Da jedoch der Code des schwarzen Werkstückes stark beschädigt und damit unbrauchbar war, wurden alle Codes neu erstellt. Die Idee war nun, die kreisrunde Fläche auf jedem der Werkstücke komplett ausnutzen, das heißt, der Aufkleber wird rund und enthält mittig den QR-Code. Bei der Erstellung der Codes war die Webseite QRCode-Monkey [13] sehr hilfreich. Diese stellt ein kostenloses Tool zur Erstellung von QR-Codes zur Verfügung, wobei es möglich ist, diese nach eigenem Ermessen in Bezug auf Form und Farbe zu gestalten. In diesem

Fall wurden einfache Codes im Standarddesign erstellt. Die Codes enthalten nur den Text „Aluminium“, „Schwarz“ oder „Weiss“ und sind in Abbildung 5.10 dargestellt.

Abbildung 5.10.: Werkstücke mit aufgebrachten QR-Codes

5.4. Einbindung des QR-Codescanners in die Bandanlage

5.4.1. Netzwerkvariablen zwischen CODESYS V2.3 und V3.5

Um die Kommunikation zwischen den für die einzelnen Stationen der Anlage verantwortlichen Programmen zu realisieren, wurde auch im Fall des QR-Codescanners wieder auf Netzwerkvariablen zurückgegriffen. Allerdings tauchte hierbei das Problem auf, dass CODESYS V2.3 zwar problemlos Variablen von V3.5 lesen kann, dies jedoch in umgekehrter Reihenfolge ohne weiteres nicht möglich ist. Die versionsübergreifende Kommunikation wird in der CODESYS Hilfe [11] beschrieben und im Folgenden ausführlich erläutert.

Grundlegend ist nur eine Variable nötig, die das Magazin in V2.3 bereitstellt und vom Scanner erkannt werden soll. Es handelt sich um das Triggersignal, welches den Scanvorgang startet. Zunächst wird eine Netzwerkvariablenliste in CODESYS V2.3 angelegt, wie es auch schon im Abschnitt 4.2 geschildert wurde. Wichtig ist, das dies eine neue Liste wird und somit auch eine andere COB-ID erhält. Zudem ist wichtig, dass die Liste Schreibzugriff bekommt. In dieser Liste wird die Variable

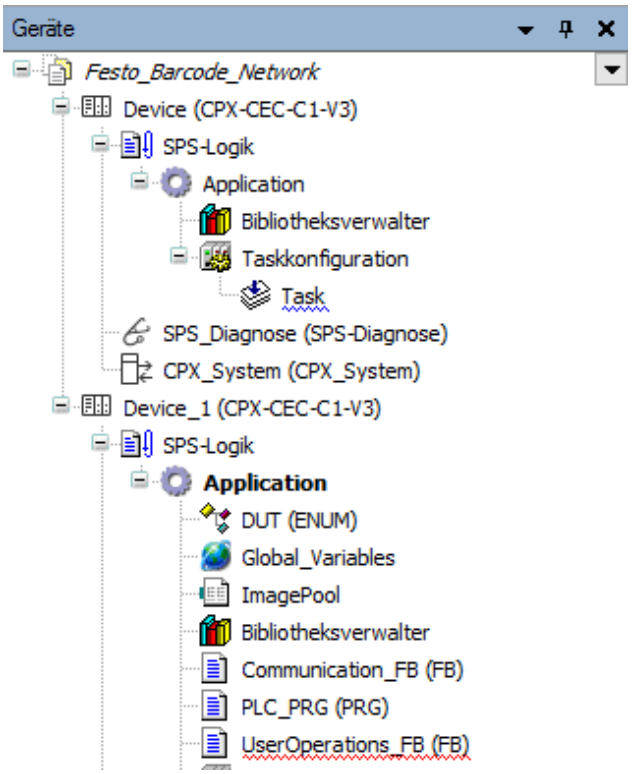


Abbildung 5.11.: Geräteliste in CODESYS V3.5

definiert, welche als Trigger dienen soll, in diesem Fall wurde sie als `net_Scannen` bezeichnet. Damit diese Variable nun im Programm des QR-Codescanners gelesen werden kann, muss zunächst eine sogenannte „Dummy-CPU“ erstellt werden. Die Central Processing Unit (CPU) ist das Gerät, auf welche das erstellte Programm gespielt wird, in diesem Fall die FESTO-SPS. Mit einem Rechtsklick auf einen leeren Bereich in dem Feld, welches die Geräte und Programmteile anzeigt, und anschließend *Gerät anhängen*, gelangt man in die Geräteauswahl. Hier wird erneut das Gerät *CPX-CEC-C1-V3* ausgewählt und hinzugefügt. Anschließend sollte die Liste wie in Abbildung 5.11 dargestellt aussehen. Mit einem Rechtsklick auf *Application* und *Objekt hinzufügen* wird aus der vorliegenden Liste *Netzwerkvariablenliste (Sender)* ausgewählt und hinzugefügt. Danach muss in dem sich öffnenden Fenster die Liste genauso eingestellt werden, wie diejenige welche zuvor im CODESYS V2.3 Projekt erstellt wurde und mit der kommuniziert werden soll. Die Einstellungen sind in Abbildung 5.12 zu sehen. Wichtig ist, dass die COB-ID beider Listen übereinstimmt.

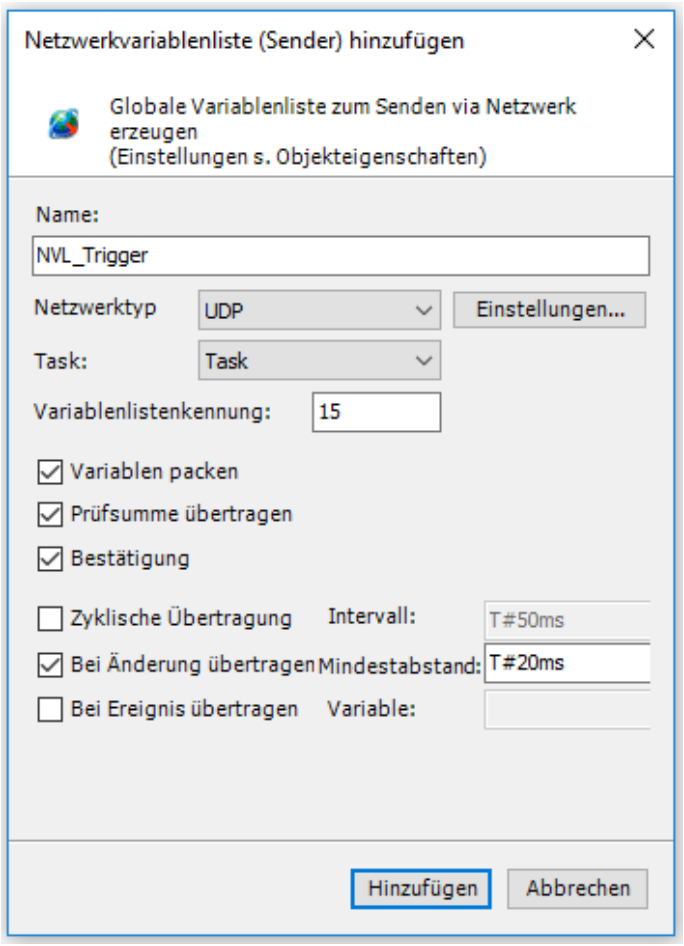


Abbildung 5.12.: Einstellungen der Netzwerkvariablenliste (Sender) in CODESYS V3.5

Nachdem die Variablenliste erstellt wurde, müssen ihre Eigenschaften aufgerufen werden. Unter *Verknüpfung mit Datei* wird festgelegt, dass die Liste vor dem Übersetzen exportiert werden soll und es wird ein Speicherort für diese festgelegt. Das zugehörige Fenster ist in Abbildung 5.13 dargestellt. Nach Bestätigung dieser Einstellung muss nun noch die benötigte Variable `net_Scannen` in die Liste geschrieben werden. Anschließend muss die *Application*, welche die Netzwerkvariablenliste beinhaltet, mit einem Rechtsklick und *Aktive Applikation setzen* aktiviert werden. Wird jetzt das Programm mit Betätigen der Taste *F11* auf der Tastatur übersetzt, wird auch die Datei mit der Variablenliste an den zuvor festgelegten Speicherort geschrieben.

Um die Liste nun auch im eigentlichen Programmteil nutzen zu können, wird zunächst

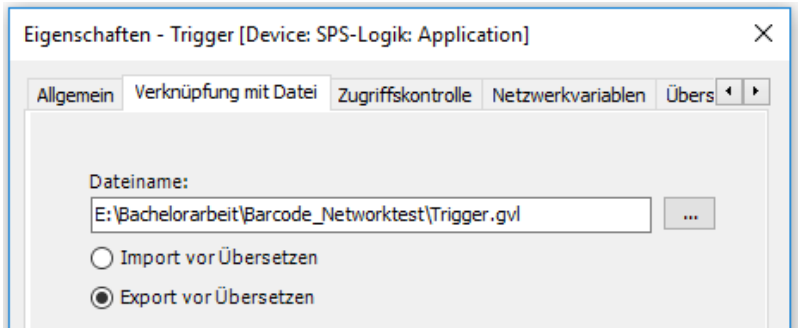


Abbildung 5.13.: Verknüpfung der Netzwerkvariablenliste mit einer Datei

die *Application*, in der sich die Programmbausteine befinden, wieder aktiviert. Nun wird genau wie zuvor eine Netzwerkvariablenliste unter der dortigen Applikation eingefügt, diesmal jedoch als Empfänger. Diese benötigt einen Namen und die Information, woher sie ihre Daten beziehen soll. Hier wird *Import aus Datei* ausgewählt, wie in Abbildung 5.14 zu sehen ist. Anschließend muss der Speicherort und die dort abgelegte, exportierte Variablenliste ausgewählt werden. Ist dies getan und wird das Programm erneut übersetzt, erscheint in der gerade erstellten Variablenliste ebenfalls die Variable `net_Scannen` und kann nun auch im Programm normal verwendet werden. Zu beachten ist hierbei jedoch, dass diese nur gelesen, aber nicht verändert werden kann. Das Magazin hat alleinigen Schreibzugriff auf die Variable.

Für solche Variablen, die in CODESYS V3.5 geschrieben und in V2.3 gelesen werden sollen, genügt es, in V3.5 eine Netzwerkvariablenliste unter der aktiven Applikation als Sender zu erstellen. Diese muss nicht exportiert werden. Um diese Liste in V2.3 lesen zu können, wird auch hier eine neue Netzwerkvariablenliste angelegt, welche die gleichen Einstellungen besitzt wie die in V3.5. Diese muss nur die Berechtigung zum Lesen der Variablen erhalten, schreiben darf ohnehin nur das Programm in CODESYS V3.5 in diese. Es ist darauf zu achten, dass beide Listen die gleichen Variablen enthalten.

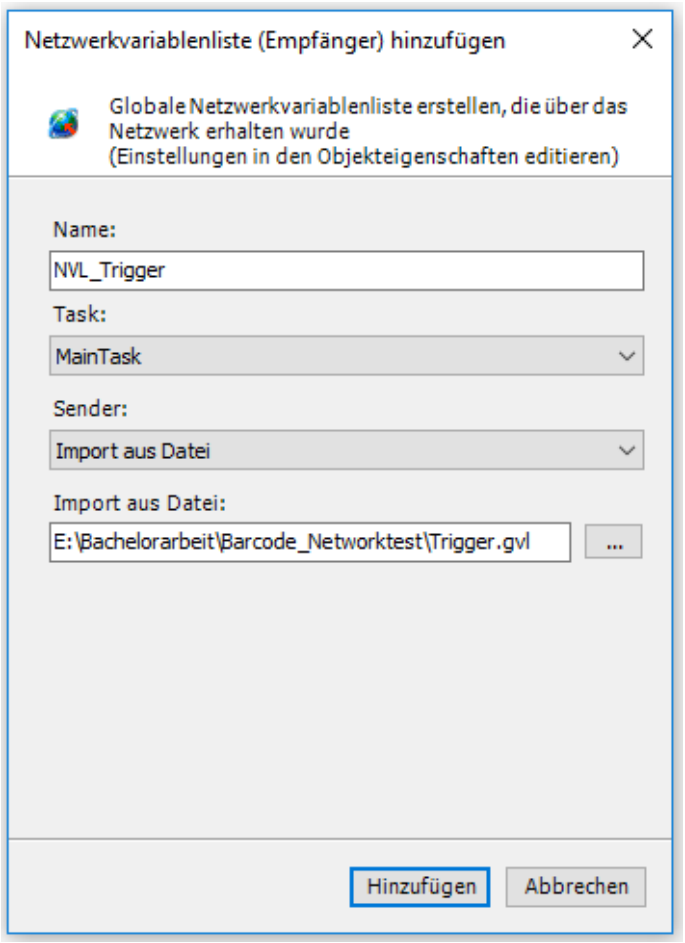


Abbildung 5.14.: Import der Netzwerkvariablenliste aus der Dummy-CPU

5.4.2. Ablauf bei gleich bleibendem Sortieralgorithmus

Mit der Verwendung des QR-Codescanners ergeben sich eine Reihe neuer Möglichkeiten, die Sortierung der Werkstücke vorzunehmen. Anfangs sollte diese jedoch wie schon zuvor erfolgen, das heißt, jedes erste und dritte Werkstück einer Sorte wird am ersten Band eingelagert, die jeweils zweiten und vierten auf dem zweiten Band. Hier gibt es mehrere Varianten, dies umzusetzen. Die einfachste ist sicherlich, in der vorhandenen Programmierung der Bänder bei den Sensoren eine weitere Bedingung hinzuzufügen. Die Bedingung besagt, dass der jeweilige Sensor nur auslösen darf, wenn am Magazin auch das Objekt mit dem passenden Material erkannt wurde. Die

Counter für die Lager laufen dann ebenfalls weiter über die Bänder.

Eine weitere Methode ist jedoch, dass schon beim Scanvorgang festgelegt wird, wohin welches Werkstück transportiert wird. In dem Fall dienen die Sensoren nur noch dazu, zu erkennen, ob sich das Werkstück auf der richtigen Position zum Einlagern befindet. Vorteilhaft hierbei ist, dass die Sensoren später durch einfache Lichtschranken ersetzt werden könnten, da es nun nicht mehr nötig ist, das Material der Werkstücke zu erkennen. Bei dieser Art der Einordnung wird der Programmierung des Magazins einiges hinzugefügt, dafür wird die der Bänder etwas gekürzt.

Das Magazin wird, wie zuvor auch, nach dem Drücken des Startschalters der Anlage, ein Werkstück auf das Band schieben. Der Scanbereich des Multicode Readers befindet sich genau dort, wo das Werkstück vom Magazin platziert wird. Sobald der Schieber ausgefahren und somit das Werkstück auf dem Band liegt, wird die Netzwerkvariable `net_Scannen` auf `TRUE` gesetzt. Diese wird vom QR-Codescanner abgefragt und dient als Triggersignal zum Starten des Scanvorgangs. `net_Scannen` wird kurz darauf wieder in den Ausgangszustand `FALSE` zurück versetzt, da diese nur kurz zum Auslösen des Scanvorgangs benötigt wird. Der Scanner erfasst nun den QR-Code. Je nachdem, welcher Code erkannt wurde, werden die Variablen wie folgt gesetzt:

Aluminium `net_Alu` wird `TRUE`

White `net_White` wird `TRUE`

Black `net_Black` wird `TRUE`

Da bei den erstellten und verwendeten Codes nur diese drei Arten von Text im Code stehen können, werden auch nur diese in der Programmierung berücksichtigt. Wird ein Code verwendet, den das Programm nicht kennt, gibt der Scanner zurück, dass das Lesen des Codes fehlgeschlagen und erneut versucht werden muss. Gleiches gilt auch bei Lesefehlern jeglicher Art.

Nachdem das Magazin die Antwort vom Scanner erhalten hat und diese einer der drei oben genannten Fällen entspricht, wird entschieden, was mit dem Werkstück passieren soll. Zunächst wird abgefragt, welcher Art das Werkstück entspricht und

wieviele davon bereits verarbeitet wurden. Ist es das erste oder dritte seiner Art, wird die neu eingeführte Netzwerkvariable `net_B1` wahr, beim zweiten oder vierten `net_B2`. Diese geben die Zurodnung zum jeweiligen Band an, welches der Nummerierung entspricht. Zudem wird entweder `net_A` für Aluminium, `net_B` für Schwarz oder `net_W` für Weiß auf `TRUE` gesetzt. Diese spiegeln die Farbe des Werkstückes wieder. Nachdem alle Variablen ordnungsgemäß gesetzt wurden, übermittelt das Magazin wie auch zuvor die Variable `net_TAAB1` an das erste Band, um zu signalisieren, dass der Abtransport des Werkstückes beginnen kann.

Je nachdem, welche Variablen gesetzt wurden, darf nur noch ein bestimmter Sensor reagieren, wenn das Werkstück diesen passiert. Dies richtet sich nach der zuvor erfassten Art des Werkstücks und der Zuordnung zum Band. Erfasst der betreffende Sensor das Objekt, wird das Band angehalten und dieses eingeordnet. Die Bänder sind nun nicht mehr selbst dafür verantwortlich, die Reihenfolge der auf ihnen eingelagerten Werkstücke zu überwachen. Sie besitzen lediglich noch eigene Counter zum Setzen der Triggervariablen für die Füllstandsanzeige in der Visualisierung. Diese sind jedoch unabhängig vom Gesamtcounter am Magazin. Durch die Zuweisung der Werkstücke schon am Anfang des Prozesses sind an den Bändern einige Abfragen weggefallen, wodurch das Programm an dieser Stelle etwas gekürzt wurde.

Der Prozess läuft nach dem Hinzufügen des QR-Codescanners weiterhin flüssig, alle Funktionen wie beispielsweise Notaus und Richten sind problemlos nutzbar. Der Prozess startet erst dann, wenn ein QR-Code erkannt wurde und einem Werkstückmaterial zugeordnet werden konnte.

6. Visualisierung

6.1. Elemente in der Visualisierung

Nachdem die Programmierung der Anlage abgeschlossen ist, sollte es nun auch möglich sein, diese an einem Bildschirm zu überwachen. Hier wurde die Webvisualisierung von CODESYS genutzt. Es ist möglich, innerhalb eines Projektes diverse Visualisierungen anzulegen, diese miteinander zu verknüpfen, um beispielsweise zwischen mehreren Ansichten wechseln zu können, und all das in einem Webbrowser aufzurufen. Da die hier zu überwachende Anlage eher klein und modellhaft ist, beschränkt sich auch die Visualisierung dieser auf nur eine Anzeige. Es bestand die Anforderung, dass folgende Aspekte angezeigt werden:

Status Der derzeitige Zustand der Anlage soll ersichtlich sein, so wie er an der Anlage selbst anhand der Statuslampen erkennbar ist.

QR-Code Der Status des QR-Codescanners soll zeigen, welche Farbe erkannt wurde oder ob das Lesen des Codes fehlgeschlagen ist.

Lagerfüllstände Der momentane Füllzustand der Lager soll erkennbar sein.

Die gesamte Visualisierung wurde im Magazin der Anlage umgesetzt. Der Grund hierfür war, dass sich dort alle wichtigen Informationen leicht abrufen lassen, wie beispielsweise der Status der verschiedenen Lampen. Dadurch wurden Kommunikationswege zwischen den Stationen für die Visualisierung teilweise vermieden.

Zunächst wurde der Status der Anlage veranschaulicht. Prinzipiell wurden hierbei die sich, wie auch im Original, übereinander befindlichen Statuslampen grafisch nachgebaut. Die Farben in der Visualisierung reagieren auf die Ausgänge der Lampen. Sind diese an, also der Ausgang auf `TRUE` gesetzt, so wird auch das grafische Abbild angeschaltet. Andernfalls bleibt das Element unsichtbar. Der Aufbau dieses Elements der Visualisierung ist in Abbildung 6.1 zu sehen. Die farbigen Felder über *Status*

sind entsprechend nur zu sehen, wenn auch die entsprechende Lampe an der Anlage angeschaltet ist. Zur besseren Darstellung werden in der Abbildung alle Farbfelder gezeigt. Neben der Statusanzeige selbst befindet sich dort auch eine Legende, damit Nutzer der Anlage sofort wissen, was die verschiedenen Lampenfarben zu bedeuten haben.

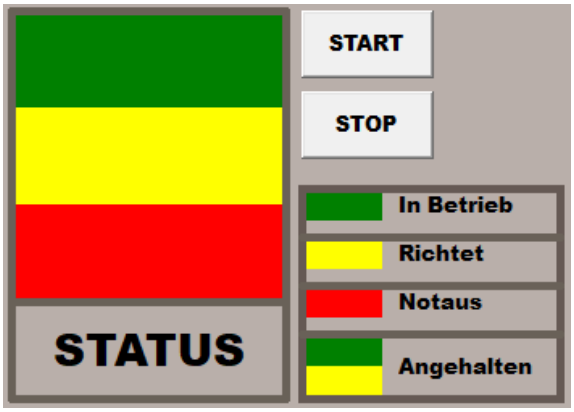


Abbildung 6.1.: Statusanzeige in der Visualisierung, mit Legende und Schaltern

Um die Anlage auch direkt in der Visualisierung bedienen zu können, wurde dort ein Start- und Stoppschalter eingefügt. Diese befinden sich ebenfalls direkt neben der Statusanzeige. Der Notaus sowie das Richten müssen weiterhin direkt an der Anlage erfolgen.

Zur Realisierung der Lagerfüllstände wurden die an beiden Bändern befindlichen Lager grafisch dargestellt. Die Lager sind übersichtlich nebeneinander angeordnet. Beim Start der Anlage sind diese zunächst leer. Jedes Werkstück wird, nachdem es in ein Lager einsortiert wurde, in der Visualisierung mit seiner Farbe dargestellt. So erscheinen für weiße Wertstücke auch weiße Kreise in der Darstellung des dafür vorgesehenen Lagers. Die Reihenfolge der Lager entspricht dabei der auf den Bändern. In Abbildung 6.2 sind die Lager im komplett gefüllten Zustand dargestellt.

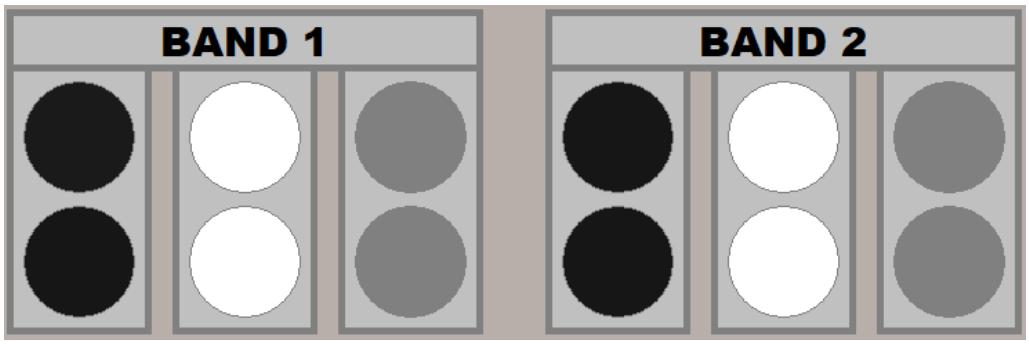
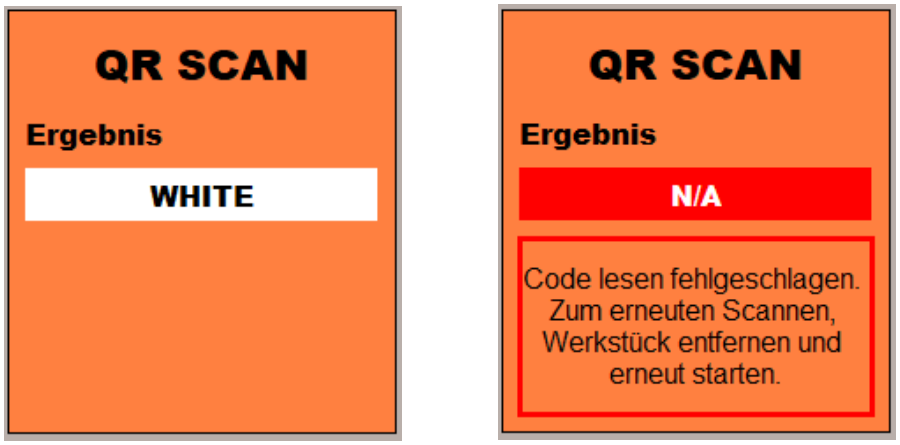


Abbildung 6.2.: Lagerfüllstände in der Visualisierung



(a) Anzeige bei erfolgreichem QR-Code Scan (weißes Werkstück)

(b) Anzeige bei gescheitertem QR-Code Scan

Abbildung 6.3.: Visualisierung des Status vom QR-Codescanner

Zuletzt war eine Darstellung des Ergebnisses beim Lesen des QR-Codes gefordert. Diese Anzeige erhielt ein eigenes Kästchen mit dem Titel *QR Scan*. Wird ein Code erfolgreich gelesen, so wird dort unter *Result* angezeigt, welches Material der Scanner erkannt hat. Zur schnelleren Erkennung wurde die Anzeige des Ergebnisses farblich an das erkannte Material angepasst. In Abbildung 6.3(a) ist die Anzeige bei einem erkannten weißen Werkstück zu sehen. Für den Fall, dass der QR-Codescanner einen Code nicht lesen kann, wird in der Anzeige ein Fehler ausgegeben, welcher den Nutzer auffordert, das Werkstück vom Band zu entfernen und für einen erneuten Versuch die Starttaste zu betätigen. Die Fehlermeldung ist in Abbildung 6.3(b) dargestellt.

Zusätzlich zu den erforderlichen Funktionen wurde die Visualisierung um ein paar kleinere Features erweitert. Es gibt zwei „Warnzustände“. Solche sind als violette Kästen mit weißer Schrift erkennbar. Zum einen wurde ein solcher eingearbeitet, um anzuzeigen, ob das Magazin leer ist. Ist dies der Fall, wird unter der Statusanzeige der Anlage *Magazin auffüllen!* angezeigt. Diese Meldung verschwindet, sobald das Magazin erkennt, dass sich wieder Werkstücke darin befinden. Des Weiteren gibt es den Fall, dass ein bestimmter Lagertyp der Anlage voll ist. In diesem Fall soll laut den in Abschnitt 3.2 festgelegten Anforderungen das nächste Werkstück dieser Art bis zum Ende des zweiten Bands durchfahren. Danach soll es möglich sein, die

Anlage neu zu starten. Allerdings ist vor dem Neustart eine Entleerung der Lager nötig. Anstatt diese Leerung im System automatisch zu generieren, wird der Benutzer in der Visualisierung darauf hingewiesen, dass er die Bänder leeren soll, wenn ein Werkstück das Ende von Band 2 erreicht. Diese Aktion muss er mit dem Betätigen der Schaltfläche *Bänder leeren*, welche sich über der Füllstandsanzeige von Band 2 befindet, bestätigen. Vorher ist es nicht möglich, die Anlage neu zu starten. Mit dem Einbau dieser Funktion ist die Wahrscheinlichkeit höher, dass vor einem Neustart des Prozesses die Lager auch tatsächlich leer geräumt werden und die Anlage nicht ungewollt versucht, Werkstücke in noch befüllte Lager zu schieben.

6.2. Aufrufen der Visualisierung im Webbrowser

Zur Nutzung einer in CODESYS V2.3 angelegten Visualisierung in einem Webbrowser, muss zunächst eine zusätzliche Einstellung getroffen werden. Im betreffenden Projekt, in dem die Visualisierung erstellt wurde, müssen unter *Ressourcen* die dortigen *Zielsystem Einstellungen* geöffnet werden. In der angezeigten Kategorie *Visualisierung* muss der Haken bei *Web Visualisierung* gesetzt sein. Andernfalls wird diese nicht unterstützt. Nachdem diese Einstellung bestätigt wurde, muss das Programm erneut auf die Steuerung geladen werden. Hier empfiehlt es sich, zuvor den Programmspeicher zu leeren, da das Hochladen eines Programms, welches eine Web-Visualisierung enthält, sonst fehlerhaft vonstatten gehen kann. Wenn das Programm erfolgreich auf die Steuerung geladen und gestartet wurde, kann über einen Webbrowser auf die Visualisierung zugegriffen werden. Hierzu muss die IP-Adresse der Steuerung, deren Port sowie der Name der Visualisierungsdatei in die Adresszeile eingegeben werden. Für den Anwendungsfall ergibt sich dabei „http://149.205.120.79:8080:webvisu.htm“. Die Webvisualisierung kann genauso benutzt werden, wie auch in CODESYS selbst. Die benötigte Datei `webvisu.htm` wird automatisch von CODESYS generiert.

7. Zusammenfassung und Ausblick

Die zu Beginn gestellte Aufgabe, die Bandanlage unter Verwendung des QR-Codescanners in Betrieb zu nehmen, wurde im Rahmen dieser Arbeit erfüllt. Alle Komponenten funktionieren, sowohl allein, als auch miteinander. Das Einbinden des Scanners und die Kommunikation zwischen den unterschiedlichen Versionen von CODESYS gestaltete sich als Herausforderung. Im gesetzten Zeitrahmen wurde nur eine Möglichkeit, wie die Anlage mithilfe des QR-Codescanners Werkstücke sortiert, realisiert. Hierbei werden die Farben der Werkstücke anhand des QR-Codes erkannt und nach der in Abschnitt 3.2 definierten Reihenfolge in die Lager einsortiert. Für nachfolgende Projekte ist es durchaus denkbar, dass noch andere Varianten umgesetzt werden können. Die Bandanlage ist zwar nur ein Modell einer echten Industrieanlage, dennoch verfügt sie über viele Möglichkeiten, sich an ihr auszuprobieren und neue Mittel und Wege zu finden, eine gestellte Aufgabe umzusetzen. Diese Arbeit wird voraussichtlich als Referenz für nachfolgende, an der Bandanlage arbeitende Studenten dienen. Sie bildet ein Grundgerüst für den Umgang mit der derzeit verbauten Hardware. Die Programmierung wurde vollständig im Strukturierten Text mithilfe von CODESYS V2.3 und V3.5 umgesetzt. Eine weitere Herausforderung wäre es, diese auch in anderen Sprachen der IEC-Norm umzusetzen.

Die Arbeit mit Speicherprogrammierbaren Steuerungen ist ein sehr großes Gebiet im Bereich der Automatisierungstechnik. Zwar kann ein Projekt, wie es hier bearbeitet wurde, bei Weitem nicht alle nötigen Kenntnisse für dieses Thema beinhalten und vermitteln, allerdings wurde ein solider Grundstein im Umgang mit den SPS gelegt. Interessant hierbei war die Zusammenstellung der Hardware aus Modulen verschiedener Hersteller. Es wurde bewiesen, dass bei der Benutzung von Steuerungshardware nicht zwingend auf einen Hersteller zurückgegriffen werden muss, um eine funktionstüchtige Anlage zu realisieren. Wichtig ist, die passende Schnittstelle zwischen den Geräten zu verwenden, wie es hier mittels Ethernet und PROFIBUS getan wurde. Hier lag auch der Anreiz, sich gezielter mit Kommunikations- und Bussystemen zu

beschäftigen, um ein grundlegendes Verständnis dafür aufzubauen, was zwischen den Geräten passiert.

Zusammenfassend lässt sich sagen, dass die Anlage durch den QR-Codescanner erweitert konnte und die vorgegebenen Materialien den Anforderungen entsprechend einsortiert werden. Die geschriebene Arbeit kann somit für zukünftige Studenten als Dokumentation herangezogen werden. Die Erweiterung durch den QR-Codescanner bietet vielfältige Möglichkeiten, die Anlage zu konfigurieren und zu nutzen.

Literaturverzeichnis

[1] HMS Industrial Networks AB. *PROFIBUS*. URL: http://www.feldbusse.de/Profibus/profibus.shtml (abgerufen am 01. 07. 2018).

[2] Adolf Auer. *SPS, Aufbau und Programmierung*. Heidelberg: Hüthig, 1996.

[3] *Internationales Elektrotechnisches Wörterbuch - Teil 351: Leittechnik (IEC 60050-351:2013)*. Standard. Deutsches Institut für Normung und Europäische Norm, 06/2014.

[4] *Speicherprogrammierbare Steuerungen - Teil 3: Programmiersprachen (IEC 61131-3:2013)*. Standard. Deutsches Institut für Normung und International Electrotechnical Commission, 09/2014.

[5] 3S-Smart Software Solutions GmbH. *Warum CODESYS*. 2018. URL: https://de.codesys.com/das-system.html (abgerufen am 24. 04. 2018).

[6] DENSO WAVE INCORPERATED. *Error Correction Feature*. URL: http://www.qrcode.com/en/about/error_correction.html (abgerufen am 25. 04. 2018).

[7] DENSO WAVE INCORPERATED. *History of QR Code*. URL: http://www.qrcode.com/en/history/ (abgerufen am 25. 04. 2018).

[8] Cihat Karaali. *Grundlagen der Steuerungstechnik - Einführung mit Übungen*. Berlin Heidelberg New York: Springer-Verlag, 2013.

[9] Festo Vertrieb GmbH & Co. KG. *Festo CPX-Katalog*. URL: https://www.festo.com/cat/de_de/data/doc_de/PDF/DE/CPX_DE.PDF (abgerufen am 03. 07. 2018).

[10] Festo Vertrieb GmbH & Co. KG. *Festo Support Portal*. URL: https://www.festo.com/net/de_de/SupportPortal/default.aspx (abgerufen am 27. 04. 2018).

[11] *Konfigurieren eines Netzwerkvariablen-Austauschs.* URL: `https://help.codesys.com/webapp/_cds_configuring_network_variables_exchange;product=codesys;version=3.5.12.0` (abgerufen am 07. 07. 2018).

[12] Valentin Plenk. *Angewandte Netzwerktechnik kompakt - Dateiformate, Übertragungsprotokolle und ihre Nutzung in Java-Applikationen.* Berlin Heidelberg New York: Springer-Verlag, 2017.

[13] *QRCode Monkey.* URL: `https://www.qrcode-monkey.com/`.

[14] Gerhard Schnell. *Bussysteme in der Automatisierungstechnik* -. Berlin Heidelberg New York: Springer-Verlag, 2008.

[15] Tan Jin Soon. „QR code“. In: *Synthesis Journal* 2008 (2008), S. 59–78.

[16] Fritz Tröster. *Regelungs- und Steuerungstechnik für Ingenieure - Band 2: Steuerungstechnik.* Berlin: Walter de Gruyter GmbH & Co KG, 2015.

[17] Günter Wellenreuther und Dieter Zastrow. *Automatisieren mit SPS - Theorie und Praxis - Programmierung: DIN EN 61131-3, STEP7, CoDeSys, Entwurfsverfahren, Bausteinbibliotheken. Applikationen: Steuerungen, Regelungen, Antriebe, Safety. Kommunikation: AS-i-Bus, PROFIBUS, Ethernet-TCP/IP, PROFINET, Web-Technologien, OPC, WLAN.* Wiesbaden: Vieweg+Teubner Verlag, 2011.

Anhang

A. Tabellen

Adresse	Funktion
Magazin	
QX5001.8	Schieber ausfahren
QX5001.9	Schieber einfahren
Umsetzer	
QX5001.8	Nach oben fahren
QX5001.9	Nach unten fahren
QX5001.10	Nach rechts drehen
QX5001.11	Nach links drehen
QX5001.12	Greifer öffnen
QX5001.13	Greifer schließen
Wender	
QX5001.8	Nach unten fahren
QX5001.9	Nach oben fahren
QX5001.10	Nach rechts drehen
QX5001.11	Nach links drehen
QX5001.12	Greifer öffnen
QX5001.13	Greifer schließen
Band 1 und 2	
QX5001.8	Schieber Schwarz einfahren
QX5001.9	Schieber Schwarz ausfahren
QX5001.10	Schieber Weiß einfahren
QX5001.11	Schieber Weiß ausfahren
QX5001.12	Schieber Alu einfahren
QX5001.13	Schieber Alu ausfahren

Tabelle A.1.: Belegung der Adressen der Ventilinseln an den Stationen

Name	Typ	Bedeutung
net_Start	BOOL	TRUE, wenn Startschalter gedrückt wurde
net_Stop	BOOL	TRUE, wenn Stoppschalter gedrückt wurde
net_Notaus	BOOL	TRUE, wenn Notaus gedrückt wurde
net_Richten	BOOL	TRUE, wenn gerichtet werden muss
net_TAAB1	BOOL	TRUE, wenn sich ein **T**eil **A**m **A**nfang von **B**and **1** befindet
net_TAEB1	BOOL	TRUE, wenn sich ein **T**eil **A**m **E**nde von **B**and **1** befindet
net_TAAB2	BOOL	TRUE, wenn sich ein **T**eil **A**m **A**nfang von **B**and **2** befindet
net_TAEB2	BOOL	TRUE, wenn sich ein **T**eil **A**m **E**nde von **B**and **2** befindet
net_TE1	BOOL	TRUE, wenn an Band **1** ein **T**eil von einem Sensor **E**rfasst wurde
net_TE2	BOOL	TRUE, wenn an Band **2** ein **T**eil von einem Sensor **E**rfasst wurde
net_TB	BOOL	TRUE, wenn ein **T**eil **B**earbeitet (eingelagert) wurde
net_LSWender	BOOL	TRUE, wenn die **L**icht**S**chranke am **Wender** ein Objekt erfasst hat
net_GerichtetM	BOOL	TRUE, sobald das **M**agazin fertig **gerichtet** wurde
net_GerichtetB1	BOOL	TRUE, sobald das **B**and **1** fertig **gerichtet** wurde
net_GerichtetU	BOOL	TRUE, sobald der **U**msetzer fertig **gerichtet** wurde
net_GerichtetW	BOOL	TRUE, sobald der **W**ender fertig **gerichtet** wurde
net_GerichtetB2	BOOL	TRUE, sobald das **B**and **2** fertig **gerichtet** wurde

Tabelle A.2.: Netzwerkvariablen für alle Stationen der Anlage

B. Ein- und Ausgangsbelegung an den Stationen

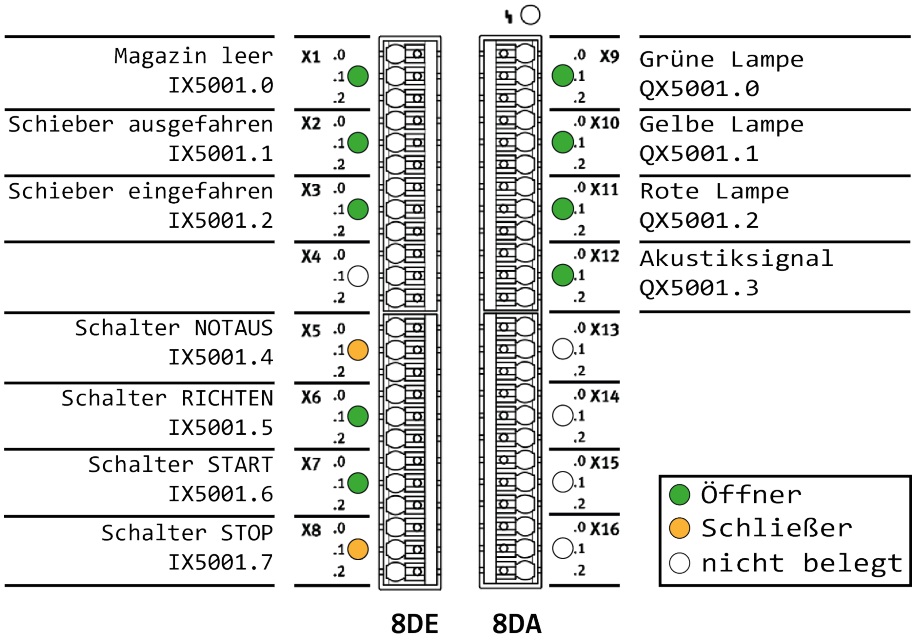


Abbildung B.1.: Ein-/Ausgänge am Magazin, Modul 8DE/8DA

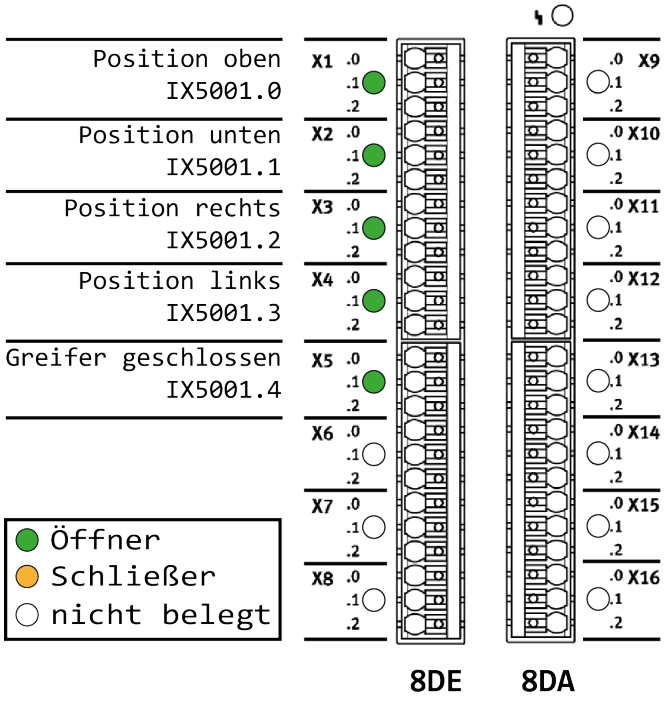


Abbildung B.2.: Ein-/Ausgänge am Umsetzer, Modul 8DE/8DA

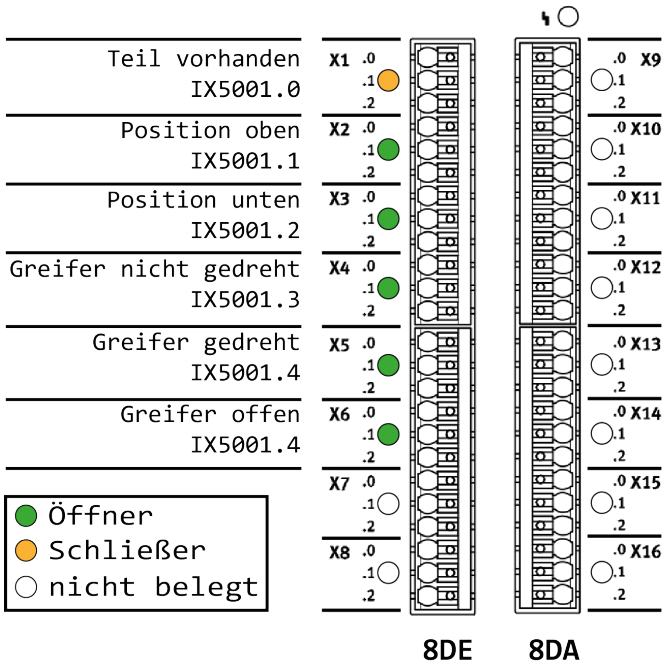


Abbildung B.3.: Ein-/Ausgänge am Wender, Modul 8DE/8DA

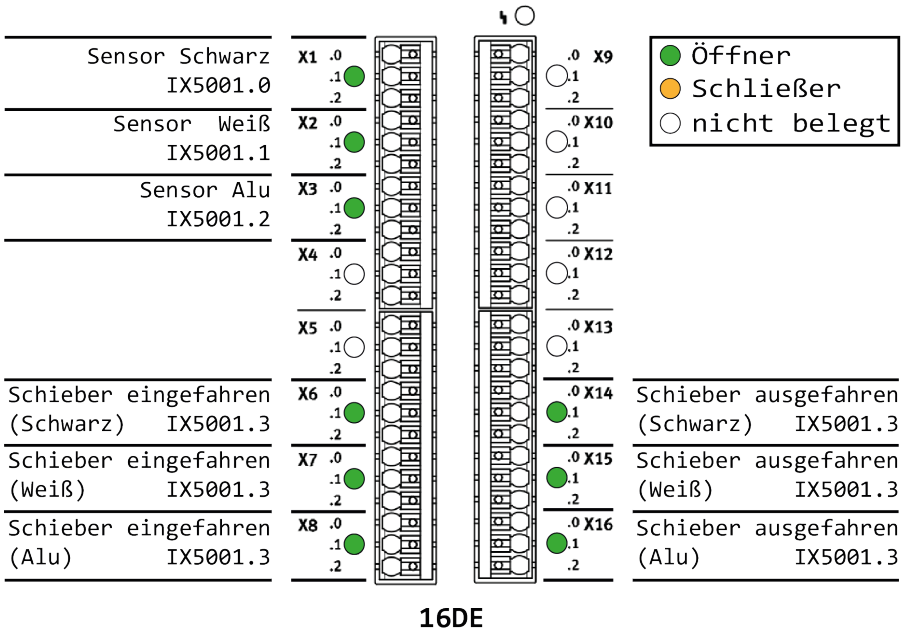


Abbildung B.4.: Eingänge an Band 1 und 2, Modul 16DE

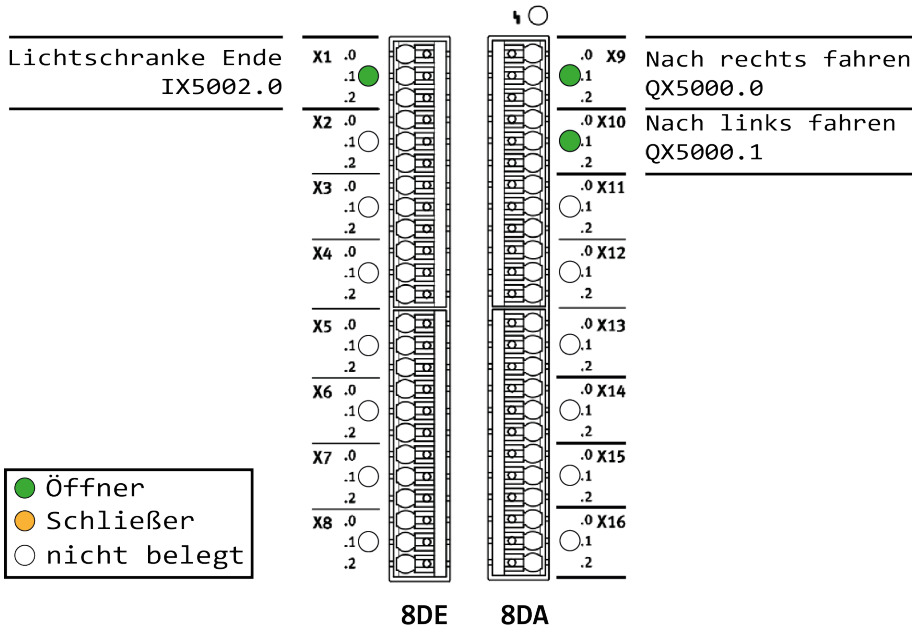


Abbildung B.5.: Ein-/Ausgänge an Band 1 und 2, Modul 8DE/8DA

C. Datenblätter

Die aufgelisteten Datenblätter in Tabelle C.1 können auf der am Ende dieser Arbeit beigelegten CD eingesehen werden.

Dateiname	Inhalt
`ifm_camera_DS.pdf`	Datenblatt zum O2I100 Multicode Reader von ifm
`WAGO_controller_DS.pdf`	Datenblatt zum WAGO PFC200 Controller
`FESTO_CPX_Catalogue.pdf`	Katalog zur CPX-Reihe von FESTO einschließlich aller Datenblätter

Tabelle C.1.: Dateienverzeichnis der beigefügten Datenblätter