Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

David Schmeding

Entwicklung eines Systems zur Steuerung und Überwachung einer Produktionsanlage

Fakultät Technik und Informatik
Department Maschinenbau
und Produktion

Faculty of Engineering and Computer Science
Department of Mechanical Engineering
and Production Management

David Schmeding

Entwicklung eines Systems zur Steuerung und Überwachung einer Produktionsanlage

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Maschinenbau, Entwicklung und Konstruktion
am Department Maschinenbau und Produktion
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Erstprüfer: Prof. Dr.-Ing. Wolfgang Schulz
Zweitprüfer: Prof. Dr.-Ing. Thomas Frischgesell

Eingereicht am: 26. Februar 2016

David Schmeding

Thema der Arbeit
Entwicklung eines Systems zur Steuerung und Überwachung einer Produktionsanlage

Stichworte
Automatisierung, Prozessvisualisierung, Speicherprogrammierbare Steuerung, Profinet, Industrie 4.0

Kurzzusammenfassung
In dieser Arbeit wird das Prinzip der Entwicklung eines Systems zur Steuerung und Überwachung einer Produktionsanlage anhand eines Modells von der Aufgabenstellung, über die Konzeptionierung und die Implementierung bis zur fertigen Steuerung beschrieben. Des Weiteren wird das Modell in einen Industrie 4.0.-Kontext integriert.

David Schmeding

Title of the paper
Developing a system for controlling and monitoring a production plant

Keywords
Automation, Process Visualisation, Programmable Logic Control, Profinet, Industry 4.0

Abstract
This bachelor thesis depicts the approach to development of a system for controlling and monitoring the model of a production plant. It covers the problem, the system concept, the programming and the finished control. Furthermore the model is integrated into an Industry 4.0 context.

Inhaltsverzeichnis

1. Einführung

1.1. Einleitung

Speicherprogrammierbare Steuerungen, kurz SPS, sind heutzutage nicht nur ein elementarer Bestandteil in der Automatisierung der modernen Industrie sondern übernehmen auch im Alltag häufig die Kontrolle im Verborgenen, sei es beim Fahrstuhl, der Ampelanlage oder dem Parkhaus.

Diese Bachelorarbeit befasst sich mit der Programmierung einer SPS und einer zugehörigen Bedienungssoftware für ein Modell einer industriellen Produktionsanlage. Zur Anwendung kommt dabei das sogenannte „Modulare Produktionssystem“ der Firma Festo, kurz MPS. Im Zuge der zunehmenden Digitalisierung von sämtlichen Produktionsprozessen, befasst sich diese Bachelorarbeit weitergehend mit der Übertragung klassischer Produktionssysteme in den „Industrie 4.0“-Kontext.

1.2. Aufgabenstellung

Für ein modulares Produktionssystem, bestehend aus Verteil-, Prüf- und Sortierstation, soll eine Steuerung entworfen und realisiert werden. Zur Verfügung steht hierzu eine industrielle Steuerungshardware mit Profinet-Komponenten. Die Bedienung und Visualisierung des Prozesses soll über eine Ethernet-Verbindung möglich sein.

Die Aufgabenstellung beinhaltet im Einzelnen:

1. Einarbeitung in die Funktionsweise des modularen Produktionssystems
2. Entwicklung einer geeigneten Steuerungssoftware auf Basis der IEC61131 mit Schutzfunktionen und Betriebsarten
3. Aufbau einer Bedienschnittstelle auf Basis von LabVIEW
4. Gegebenenfalls sind Überlegung zu weiteren Schnittstellen im Hinblick auf Industrie 4.0 durchzuführen

2. Konzeptionierung

2.1. Verwendete Programmiersprachen und -systeme

2.1.1. IEC 61131

Die IEC 61131 ist eine weltweit gültige Norm für Programmiersprachen von speicherprogrammierbaren Steuerungen und legt für die SPS-Programmierung einige grundlegende Programmier-Konzepte fest (vgl. zum Folgenden Wellenreuther und Zastrow, 2011, S. 2f.):

Datentyp- und Variablenkonzeption

Es besteht eine verbindliche Definition von Datentypen, Konstanten und Variablen, welche die Grundlage für eine explizit auszuführende Variablendeklaration bilden.

Programmorganisationskonzept

Die Programmierung wird mit den sogenannten Programmorganisationseinheiten (POE, engl. POU) strukturiert. Es wird in Programme, Funktionsbausteine und Funktionen unterteilt (siehe Abschnitt 5.3). Unter diesen POEs besteht eine hierarchische Gliederung, bei der sich Programme an der Spitze befinden, darunter die Funktionsbausteine und zu unterst die Funktionen. Die Entwicklung von Programmteilen in POEs erhöht die Übersichtlichkeit eines Projekts und vereinfacht auch die Wiederverwendbarkeit dieser als Bausteine in späteren Anwendungen.

Fachsprachenkonzept

Die SPS-Norm sieht zur Programmierung fünf Fachsprachen vor, welche zwei grafische (Kontaktplansprache KOP und Funktionsbausteinsprache FBS), zwei textbasierte (Anweisungsliste AWL und Strukturierter Text ST) sowie die hybride Ablaufsprache AS umfassen.

Taskkonzept

Das Taskkonzept erlaubt eine Ausführungssteuerung über sogenannte Tasks, mit denen definierbar ist, in welchem Zyklus und mit welcher Priorität oder durch welches Ereignis eine Programmorganisationseinheit ausgeführt werden soll.

2.1.2. PC WORX

PC WORX ist die Programmierumgebung für alle Steuerungen von Phoenix Contact, welche die Programmierung der SPS nach IEC 61131 erlaubt. PC WORX beinhaltet alle Instrumente zur Programmierung, Kommunikation und Konfiguration von Phoenix Contact SPS und ermöglicht des Weiteren die Feldbuskonfiguration und Anlagendiagnose.

2.1.3. LabVIEW

Die von National Instruments entwickelte Programmiersprache und Entwicklungsumgebung LabVIEW (**Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench) funktioniert grafisch nach dem Prinzip eines Datenflussdiagramms. Zwischen den als Virtuelle Instrumente (VIs) bezeichneten Funktionsblöcken wird mit Verbindungslinien ein Datenfluss definiert. Jedes so erzeugte Programm kann seinerseits wieder als VI in anderen Programmen benutzt werden. Ein wichtiges Konzept ist dabei auch die Programmierung in zwei Ebenen, bei dem in der einen Ebene das sogenannte Frontpanel, die Bedienoberfläche, erstellt wird und in der zweiten Ebene das Blockdiagramm, welches das Programm hinter dem Frontpanel darstellt und die Eingabewerte im Datenfluss verarbeitet und Ausgabewerte produziert (vgl. Georgi und Metin, 2012, S. 31f.).

Durch die einfache Erstellung von Bedienoberflächen und Visualisierungen, übersichtliche Möglichkeiten zur Datenerfassung, Auswertung und Darstellung, sowie integrierten Kommunikationslösungen zu OPC-Servern zum Datenaustausch mit SPSen, bietet sich LabVIEW zur Realisierung der Prozessvisualisierung für den PC an.

2.1.4. PC WebVisit

Zur Erstellung einer browserbasierten und damit endgerätunabhängigen Visualisierung bietet Phoenix Contact das Programm WebVisit an. Die mit WebVisit erstellte Visualisierung kann auf bestimmte SPS-Modelle geladen werden, welche dann zusätzlich als Web-Server fungieren und die Visualisierung in HTML5 oder als Java Applet zur Verfügung stellen. Um bei der Programmierung mit WebVisit auf Prozessdaten zugreifen zu können, müssen die betreffenden

Variablen bei der SPS-Programmierung in PC WORX als Process Data Points (PDD) markiert werden.

Um das Web-Panel auch auf Android-Geräten darstellen zu können, muss dieses in HTML5 zur Verfügung gestellt werden, da Java Applets von Android nicht unterstützt werden. Die verwendete SPS ILC 350 PN unterstützt mit der aktuellen Firmware 01/4.6F/3.95 als Webserver jedoch lediglich Java Applets. Eine Unterstützung von HTML5 soll für kommende Firmware-Versionen vorgesehen sein. Für die Prozessvisualisierung auf einem Android-Tablet muss daher eine andere Lösung gefunden werden.

2.1.5. App Inventor 2

Der App Inventor ist eine ursprünglich von Google entwickelte Programmierumgebung zur grafischen Programmierung von Android Apps für Smartphones und Tablets. Seit 2012 wird die browserbasierte Plattform vom Massachusetts Institute of Technology (MIT) betrieben und steht inzwischen in der zweiten Version zur Verfügung.

Das Programmierkonzept teilt sich ähnlich wie bei LabVIEW in zwei Ebenen, einen Design Editor, mit dem die Bedienoberfläche erstellt wird, und einen Blocks Editor, mit dem die Beziehungen der Designkomponenten und Parameter sowie Kontrollstrukturen erstellt werden. Es wird jedoch nicht nach dem Datenflussprinzip programmiert, sondern eher in einer grafischen Darstellung textuellen Codes, in der Verschachtelungen und Abhängigkeiten als Blöcke ineinander gesteckt werden. Jede Änderung kann, soweit sie die Ausführbarkeit des Programms nicht behindert, unmittelbar auf einem drahtlos verbundenen Smartphone oder Tablet getestet werden.

Der App Inventor eignet sich nicht für die Entwicklung spezialisierter mobiler Anwendungen, da komplexe Problemstellungen aufgrund der einfachen Funktionsweise und begrenzten Möglichkeiten nicht abbildbar sind. Er eignet sich allerdings hervorragend für kleinere Projekte, bei denen schnelle und einfache Lösungen mit wenig Vorwissen im Gebiet von Mobile Applications entwickelt werden sollen.

2.1.6. Cybus

Im Hinblick auf „Industrie 4.0“, wo der Datenaustausch zwischen sämtlichen Systemkomponenten möglichst flexibel sein sollte, wird ein Server benötigt, der eine entsprechende Entkopplung der Kommunikation ermöglicht. Im Rahmen dieser Arbeit nimmt er die Vermittlerfunktion zwischen einem Tablet und LabVIEW ein. Zum Einsatz kommt in diesem Fall die Serversoftware Cybus, welche unter anderem einen MQTT-Broker (siehe Abschnitt 3.4) und einen HTTP-

Server umfasst. Der Server läuft im Rechenzentrum und ist HAW-intern unter der Adresse cybus.rzbt.haw-hamburg.de zu erreichen. Unter cybus.rzbt.haw-hamburg.de/admin können Zugriffsrechte verwaltet werden und unter cybus.rzbt.haw-hamburg.de/workbench wird zusätzlich die Möglichkeit geboten, serverinterne Datenflüsse, Verarbeitung und Schnittstellen zu erstellen.

2.2. Programmierkonzept

Da das zu entwickelnde System aus einem Zusammenspiel der Bedienschnittstelle an einem PC und dem Programm, das physisch auf der SPS läuft, besteht, werden die Komponenten parallel zueinander programmiert, aneinander angepasst und weiterentwickelt. Beispielsweise ist die Visualisierung auf die Prozessvariablen der SPS angewiesen und umgekehrt ist es von Vorteil, die korrekte Funktionsweise der SPS oder die Kommunikation zwischen beiden Komponenten mit simulierten Prozessdaten direkt mit der Visualisierung zu testen.

Abbildung 2.1.: Teilnehmer des Datenaustauschs

Es wird jedoch nicht gleichzeitig an zwei Bildschirmen mit zwei verschiedenen Programmen programmiert, sondern es werden abwechselnd Etappenziele bei der Entwicklung beider Komponenten gesetzt, welche im Zusammenspiel neue Funktionen oder Möglichkeiten eröffnen.

Mit der Erweiterung zum Thema „Industrie 4.0“ tritt noch eine dritte Komponente ins Spiel, die mobile Tablet-App zur Überwachung und Steuerung der Anlage. Diese Komponente ist zwar abhängig von den anderen beiden, da sie auf die Funktionen dieser angewiesen ist, umgekehrt ist die Entwicklung der beiden lokalen Komponenten, bis auf die in die Prozessvisualisierung integrierte Schnittstelle zum Zugriff auf die OPC-Variablen, jedoch völlig unabhängig von der Tablet-App. Diese muss daher nicht zwingend parallel entwickelt werden.

Die zur SPS-Programmierung verwendeten Programmiersprachen nach IEC 61131 sind Funktionsbausteinsprache, Ablaufsprache und Strukturierter Text. Das Programm wird aus Funktionsbausteinen zusammengesetzt, die jeweils einen Teil des Prozesses abdecken und in Ablaufsprache programmiert sind. Eine sinnvolle Unterteilung des Prozesses wird durch den modularen Aufbau der Anlage angeboten, sodass die Funktionsbausteine „Verteilen“, „Prüfen“ und „Sortieren“ den Kern des Programms bilden und den sich wiederholenden Ablauf enthalten.

Zusätzlich werden noch die Funktionsbausteine „Initialisieren“, „Reset“ und „LED-Steuerung“ verwendet, welche nicht den Kernprozess betreffende Aufgaben abdecken.

Die Vorteile der Programmierung in Ablaufsprache liegen in der prozessorientierten Darstellung der Steuerungsaufgaben. Beim Entwurf der Steuerung kann sich eng an der Anlage orientiert und jeder Prozessschritt mit seinen Aufgaben und Bedingungen für den Übergang zum nächsten Schritt definiert werden. Der Ablauf-Funktionsplan ist damit ein das Denken unterstützendes, anschauliches und leicht nachvollziehbares Entwurfsinstrument, welches nach richtiger Erstellung gleichzeitig auch die Lösung der Steuerungsaufgabe darstellt. Er hilft außerdem bei der Planung, Inbetriebnahme und Störungssuche, da im Debug-Modus in PC WORX jederzeit klar zu erkennen ist, welcher Schritt aktiv und welche Bedingungen wann gesetzt sind (vgl. zu diesem Abschnitt Wellenreuther und Zastrow, 2011, S. 378).

3. Kommunikation

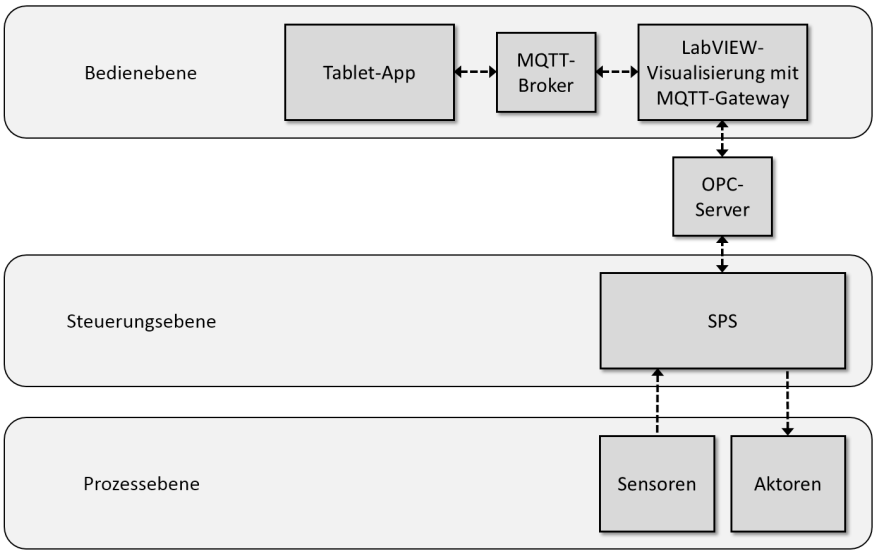

Abbildung 3.1.: Schema des Datenflusses

3.1. Ethernet

Ethernet ist ein Software- und Hardwarestandard zum kabelgebundenen Austausch von Daten in einem lokalen Netzwerk unter Verwendung von Netzwerkprotokollen, insbesondere TCP/IP. Spezifiziert sind dabei sowohl die Übertragungsformen, als auch die Kabeltypen und Stecker. Ursprünglich zur Vernetzung von Rechnern genutzt, fand Ethernet im Zuge von *Industrial Ethernet* auch für die Vernetzung von Maschinen und Anlagen in der Industrie Verwendung. Hierzu wurde jedoch die Entwicklung zusätzlicher Standards notwendig, da Ethernet keine Echtzeitanforderungen unterstützt.

In diesem Projekt wird Ethernet zur Kommunikation zwischen dem PC und der SPS zur Konfiguration und Programmierung, sowie zur Abfrage der OPC-Daten von der SPS mit dem auf dem PC laufenden OPC-Server genutzt.

3.2. PROFINET

PROFINET (Process Field Network) ist ein Industrial Ethernet-Standard, der von einem Zusammenschluss einiger Industrieunternehmen für die Automatisierung entwickelt wurde. Neben anderen Feldbussystemen wie bspw. Profibus, weist auch Profinet gegenüber der klassischen Verdrahtung jedes einzelnen Feldgerätes mit der SPS (Abbildung 3.2, (a)) erhebliche Vorteile auf, darunter die Einsparung von Kabeln, eine höhere Auflösung der Messwerte oder die Übertragung zusätzlicher Geräteinformationen (vgl. Seitz, 2012, S. 34). Profinet basiert auf TCP/IP und kann als „PROFINET IO“ zur Anbindung dezentraler Peripherie an eine zentrale Steuerung (Abbildung 3.2, (b)) oder zur Kommunikation mehrerer Steuerungen untereinander in Echtzeit genutzt werden. Des Weiteren können über Profinet auch busfähige Feldgeräte eingebunden werden (Abbildung 3.2, (c)).

In dieser Arbeit wird Profinet zur Demonstration der Anbindung dezentraler Peripherie an die SPS genutzt. Die Kommunikation findet über einen Ethernet- und Profinet-fähigen Switch statt, über den sowohl die Profinet-Geräte als auch der PC an die SPS angeschlossen werden. Des Weiteren werden Überlegungen über eine mögliche Anbindung eines busfähigen RFID-Feldgerätes angestellt (siehe Abschnitt 7.2.2).

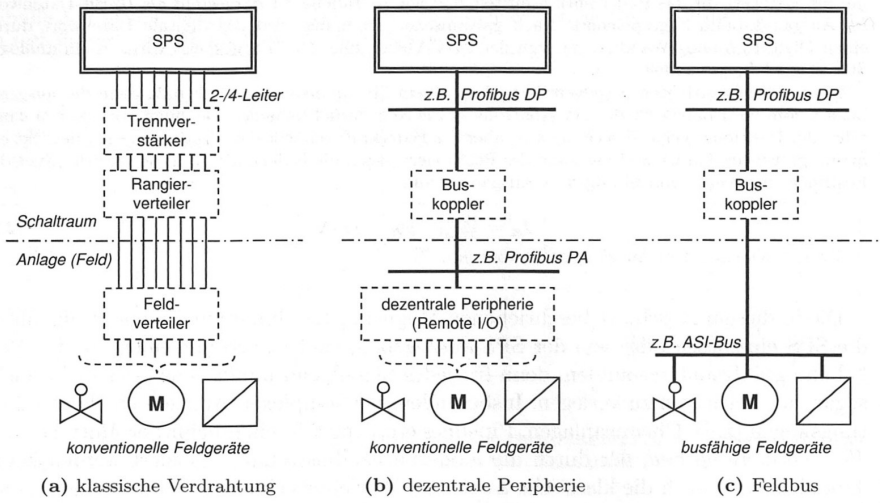

Abbildung 3.2.: Quelle: Seitz (2012, S. 34)

3.3. OPC

OPC (Open Platform Communications) bietet eine standardisierte Zugriffsmöglichkeit auf Prozessdaten einer Hardware, unabhängig vom Hersteller. Um einen Datenaustausch zwischen der SPS und der Visualisierung zu ermöglichen, wird ein OPC-Server benötigt, der die aktuellen Variablenwerte der SPS liest, diese im OPC-Format als sog. *Items* abspeichert und im TCP/IP-Protokoll zur Verfügung stellt (vgl. Seitz, 2012, S. 41). Bei der Entwicklung des Visualisierungsprogramms kann ein OPC-Client angelegt und konfiguriert werden, der lokal oder entfernt auf den OPC-Server und somit auf die Prozessdaten zugreifen kann.

3.4. MQTT

MQTT (Message Queue Telemetry Transport) ist ein Nachrichtenprotokoll, das sich durch seinen geringen Energieverbrauch und Datenaufwand auszeichnet und trotz hoher Verzögerungen und unzuverlässigen Netzwerken die Übertragung von Telemetrie-Daten ermöglicht. Es funktioniert nach einem publish/subscribe-Prinzip, wodurch von einem veröffentlichenden Teilnehmer neu eingehende Daten vom MQTT-Server, dem sogenannten „Broker“, direkt an registrierte Empfänger weitergeleitet werden. Dadurch wird eine sehr starke Entkopplung der teilnehmenden Komponente erreicht. MQTT ist vor allem für IoT (siehe Abschnitt 7.1) Anwendungen prädestiniert und standardisiert, findet durch den Fokus auf Machine-to-Machine-Kommunikation zwischen Sensoren und Aktoren, eingebetteten Systemen und mobilen Endgeräten aber auch seinen Nutzen in der Industrie 4.0.

4. Anlage

4.1. Allgemein

Die Anlage ist ein von Festo Didactic entwickeltes und vertriebenes modulares Ausbildungssystem, welches für eine praxisnahe Ausbildung gedacht ist. Das System umfasst eine Vielzahl von unterschiedlichen Stationen, insbesondere die in diesem Fall verwendeten „Verteilen“, „Prüfen“ und „Sortieren“, des Weiteren bspw. noch „Bearbeiten“, „Puffern“ oder „Montieren“.

Die Stationen können universell miteinander kombiniert werden und bilden so ein Modell einer Produktionsanlage, welches prinzipiell die gleichen Aufgaben zu erfüllen hat, wie eine Anlage im produktiven Einsatz. Vor allem das Prinzip der Programmierung einer Steuerung ist sehr gut übertragbar, da diese von den räumlichen Ausmaßen der Anlage und den darin wirkenden Kräften nicht betroffen ist, sondern lediglich Sensoren liest und Aktoren steuert.

Abbildung 4.1.: Ein schwarzes Werkstück

4.2. Grundaufbau

Das symbolische Werkstück (Abbildung 4.1) durchläuft insgesamt drei Stationen: Zunächst wird es aus einem Stapelmagazin einzeln ausgeschoben und anschließend mit einem pneumatischen Schwenkarm zur nächsten Station umgesetzt. Hier wird das Teil auf Farbe und Material geprüft, anschließend auf der Hebeplattform zu einem Prüfzylinder transportiert und dort die Höhe des inneren Absatzes getestet. Entspricht die Höhe dieses Absatzes nicht den Soll-Vorgaben, wird das Teil als Ausschuss aussortiert. Liegt die Höhe des Absatzes innerhalb der Soll-Grenzen, passiert das Teil eine Rutsche zur nächsten Station, wo es über ein Förderband von Weichen nach Farbe und Material in eines der drei Lager einsortiert wird.

4.3. Stationen

Die Stationen sind die Module, die unabhängig voneinander zu erwerben sind und von denen jedes eine Kernfunktion hat, nach der es benannt ist. Die Stationen sind jeweils auf einer Grundplatte montiert und verfügen über separate E/A-Anschlussterminals für die Steuerung.

4.3.1. Station Verteilen

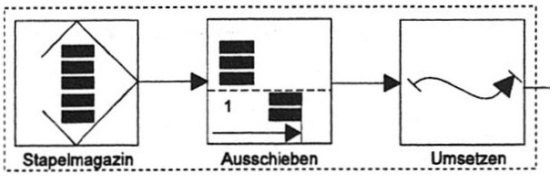

Abbildung 4.2.: Quelle: Festo Didactic, Modulares Produktions-System Handbuch

Die Station Verteilen besteht aus dem Stapelmagazin mit pneumatischem Ausschiebezylinder und dem pneumatischen Schwenkarm. Das Stapelmagazin kann acht Teile aufnehmen und ist mit einer Lichtschranke zur Erkennung eines leeren Magazins ausgestattet. Der Ausschiebezylinder befördert immer das unterste Teil aus dem Magazin und ist zur Erkennung der Endpositionen mit magnetisch betätigten Signalgebern ausgestattet. Der Schwenkarm kann Teile mit einem Vakuumsauger aufnehmen und diese von 0° bis 180° umsetzen. Die Endpositionen können variabel eingestellt werden und werden durch Tastschalter erkannt.

4.3.2. Station Prüfen

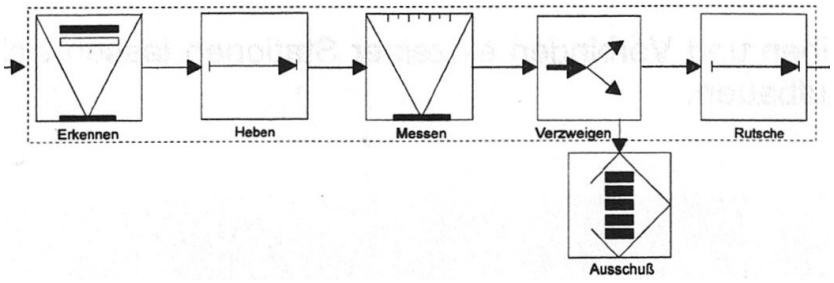

Abbildung 4.3.: Quelle: Festo Didactic, Modulares Produktions-System Handbuch

Die Station Prüfen besteht aus einer pneumatisch bewegten Hebeplattform mit einem Auswurfzylinder zum Befördern der Teile auf die Rutsche, einer Transportrutsche zur nächsten Station mit einem pneumatisch ausfahrbaren Stopper und diversen Sensoren. Die Stellung der Hebeplattform wird an der oberen und unteren Position mit magnetisch betätigten Signalgebern erkannt. In der unteren Position befinden sich ein kapazitiver Sensor zur Werkstückerkennung, ein induktiver Näherungsschalter zum detektieren von metallischen Werkstoffen und ein optischer Grauwertsensor zum Bestimmen der Werkstückfarbe. In der oberen Position befindet sich der Prüfzylinder mit einem Linearpotentiometer als Wegmesssensor und einem magnetisch betätigten Signalgeber in der ausgefahrenen Position.

4.3.3. Station Sortieren

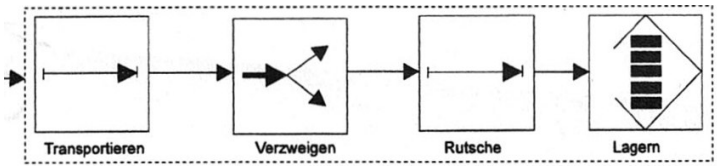

Abbildung 4.4.: Quelle: Festo Didactic, Modulares Produktions-System Handbuch

Die Station Sortieren besteht aus einem elektrisch betriebenen Förderband, von dem die Teile in die ersten beiden Lager mit pneumatischen Weichen und in das dritte Lager mit einem statischen Umlenkteil einsortiert werden. Die pneumatischen Weichen haben jeweils Stellungssensoren für ein- und ausgefahrene Position. Außerdem gibt es eine Lichtschranke zum Erkennen von Teilen auf dem Förderband, sowie eine weitere Lichtschranke zum Registrieren von Teilen, die die Lagerrutsche passieren.

4.4. Inbetriebnahme

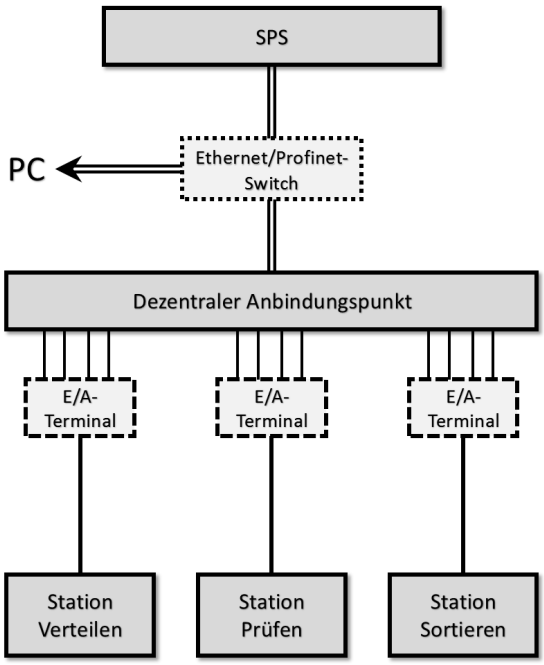

Abbildung 4.5.: Schema der Anlagensteuerung

Der Anschluss der Steuerung an die Anlage wird nach dem Schema aus Abbildung 4.5 aufgebaut. Die an den Eingangs-/Ausgangs-Terminals (siehe Anhang, Abbildung A.1) der jeweiligen Stationen zusammengefassten Ein- und Ausgangssignale werden auf der Seite des dezentralen Anbindungspunktes wieder getrennt und nach den Anschlusstabellen (siehe Anhang, Tabellen A.1 bis A.3) mit den Klemmen des Profinet-Geräts verbunden. Der dezentrale Anbindungspunkt wird mit einem Ethernetkabel mit einem Ethernet/Profinet-fähigen Switch verbunden. Mit diesem wird auch die SPS verbunden und der PC zur Programmierung und Steuerung der SPS.

Der Anschluss der Geräte an die 24 V-Spannungsversorgung von Phoenix Contact ist in den Betriebsanleitungen beschrieben. Die Sensoren und Aktoren jeder Station werden über das E/A-Terminal mit einem 21-poligen Kabel mit Amphenol-Tuchel-24-pol.-Stecker mit der SPS verbunden und ebenfalls mit 24 V (max. 5 A) versorgt. Die Druckluftquelle darf einen maximalen Druck von 8 bar haben und wird an den Druckreglern auf 6 bar eingestellt.

5. Speicherprogrammierbare Steuerung

Die Speicherprogrammierbare Steuerung (SPS) ist das „Gehirn“ der Maschine. In ihr laufen alle Sensoren und Aktoren zusammen und es werden Abhängigkeiten und kausale Zusammenhänge geknüpft oder Zustände bewertet. Diese werden in Form eines Steuerungsprogrammes vorgegeben, welches von der Central Processing Unit (CPU) ausgeführt wird.

5.1. Modellbeschreibung

Die verwendete SPS **ILC 350 PN** ist ein Controller für das Inline-I/O-System Interbus von **Phoenix Contact** mit Profinet-Schnittstelle. Die Steuerung kann modular mit digitalen und analogen I/O-Funktionsklemmen erweitert und so an die gestellten Anforderungen angepasst werden. Die Programmierung, Parametrierung und Diagnose erfolgt entweder über die Ethernet- oder die serielle Schnittstelle in PC WORX.

5.2. Profinet-Komponente

Zur Demonstration des Prinzips der *dezentralen Peripherie* kommuniziert die SPS über Profinet mit einer **IL PN BK DI8 DO4 2TX-PAC**, welche keine selbstständige SPS ist, sondern einen dezentralen Anbindungspunkt für digitale oder analoge Ein- und Ausgänge darstellt, der von der SPS gesteuert wird.

5.3. Programmierung

5.3.1. Variablen

Variablen werden in jedem POE in einer Variablenliste geführt, in der auch Variablen erstellt, bearbeitet oder gelöscht werden können. Des Weiteren können Variablen entweder in Strukturiertem Text deklariert werden oder beim grafischen Programmieren in einem Dialogfenster, welches alle möglichen Optionen aufführt.

Jede Variable wird zwangsläufig durch ihren Namen und Datentyp definiert, bei Verwendung in Funktionsbausteinen oder Funktionen zusätzlich durch ihre Verwendung als Eingangs-, Ausgangs-, lokale oder globale Variable. Um die Hardware Ein- und Ausgänge in das Programm einzubinden oder Daten zwischen Programmen auszutauschen werden globale Variablen benötigt.

Wird die Option „RETAIN“ für eine Variable gesetzt, bleibt der letzte Wert der dieser auch bei einem Abschalten der SPS gespeichert und kann bei einem sogenannten „Warmstart“ weiterverwendet werden. Ein Anfangswert kann optional vergeben werden und ist per Default FALSE bzw. Null. Von den zusätzlichen Optionen sind noch PDD und OPC für diese Anwendung von Bedeutung. Wird eine Variable als Process Data Point (PDD) definiert, wird sie in das Prozessdatenverzeichnis eingetragen, was ein Zugreifen auf sie mit Anwendungen wie bspw. WebVisit erlaubt. Wird die Option OPC aktiviert, wird die Variable für den Zugriff mit einem OPC-Server freigegeben.

Im Prozess treten an verschiedenen Stellen Variablen auf, deren Namen mit „Start_“ bzw. „Ende_“ beginnen. Zu deren Funktion siehe Abschnitt 5.3.3, *LED_Steuerung*.

Die Variable *UI_Trigger* ist im gesamten Programm die von der Bedienschnittstelle (User Interface) zu erfüllende Weiterschaltbedingung.

5.3.2. Programm

Das Programm ist das höchste Element in der POE Hierarchie und wird hauptsächlich aus Funktionsbausteinen und Funktionen zusammengesetzt, die das Verhalten der SPS steuern. Um ein Programm auszuführen, muss eine Instanz davon einem Task der CPU zugewiesen werden, wodurch eine Ausführungssteuerung ermöglicht und damit das Taskkonzept erfüllt wird.

Das Programm Main (siehe Anhang, Abbildungen A.3 bis A.5) enthält diverse Funktionsbausteine und Funktionen. Die Grünen sind benutzerdefiniert und die Roten sind Standard-Bausteine wie beispielsweise „AND“, „OR“ oder „TON“.

5.3.3. Funktionsbausteine

Funktionsbausteine können in unterschiedlichen Programmiersprachen geschrieben werden, zum Beispiel als Strukturierten Text oder Ablaufsprache, und können von einem Programm oder anderen Funktionsbausteinen aufgerufen werden. Die Eingangsvariablen werden auf der linken Seite eines Bausteins übergeben, die Ausgangsvariablen werden auf der rechten Seite ausgegeben. Durch das Verpacken von eventuell kompliziertem Code kann dieser einfach wiederverwendet werden und das Programm behält des Weiteren seine Übersichtlichkeit. Jede

Instanz, die von einem Funktionsbaustein erzeugt wird, ist vergleichbar mit einer komplexen Variable und kann Werte über mehrere Zyklen speichern.

Funktionsbausteine, von denen häufig mehrere Instanzen erstellt werden, sind die Standard-Funktionsbausteine, wie beispielsweise Flankenerkennung, Einschaltverzögerungen (TON) oder Speicherbausteine (RS-Flipflop).

Verteilen

Abbildung 5.1.: Funktionsbaustein Verteilen

Der Funktionsbaustein *Verteilen* steuert die erste Station der Anlage. Er ist in Ablaufsprache programmiert und befindet sich zu Anfang im Initialisierungsschritt. Sobald der Funktionsbaustein *Initialisieren* die Variable *Initialisieren_abgeschlossen* auf TRUE setzt und die Variable *UI_Trigger* von der Bedienschnittstelle gesetzt wird, wird der zweite Schritt aktiviert und der Prozess beginnt. Er läuft nun je nach Einstellung automatisch ab oder führt jeden weiteren Schritt nach manueller Betätigung aus.

Der Prozess startet mit dem Ausschieben eines Werkstücks aus dem Magazin. Erreicht der Ausschiebezylinder die Endposition, wird der Schwenkarm zum ausgeschobenen Teil bewegt. Damit das Ventil für die Bewegung des Schwenkarms beim Erreichen der Endposition auch geschlossen wird, wenn *UI_Trigger* für den nächsten Schritt nicht gesetzt ist, wurde an dieser Stelle der aktionslose Warteschritt *S005* eingebaut. Ist *UI_Trigger* gesetzt, wird das Unterdruckventil des Saugnapfes geöffnet. Ob das Teil erfolgreich angesaugt wurde, wird vom Vakuumschalter signalisiert und der Schwenkarm wird mit dem Teil zur Prüfstation bewegt. Nach dem Erreichen der Endposition wird 0.5 s gewartet, damit das Teil nicht aus der Bewegung losgelassen wird. Nach dem Lösen des Unterdrucks wird eine weitere halbe Sekunde gewartet, in der das Teil in die richtige Position des Prüfstandes fallen und sich beruhigen kann, bis der Prüfvorgang beginnt. Nach Abschluss des Teilprozesses *Verteilen* wird im letzten Schritt des

Ablauf-Funktionsplanes mit einem Puls der Variable *Pruefen_aktivieren* der Funktionsbaustein *Pruefen* aktiviert und der Schwenkarm in die Startposition gebracht.

Das Ablaufdiagramm und die Deklarationstabelle des Funktionsbausteins *Verteilen* befinden sich im Anhang (siehe Abbildungen A.6 und A.7).

Prüfen

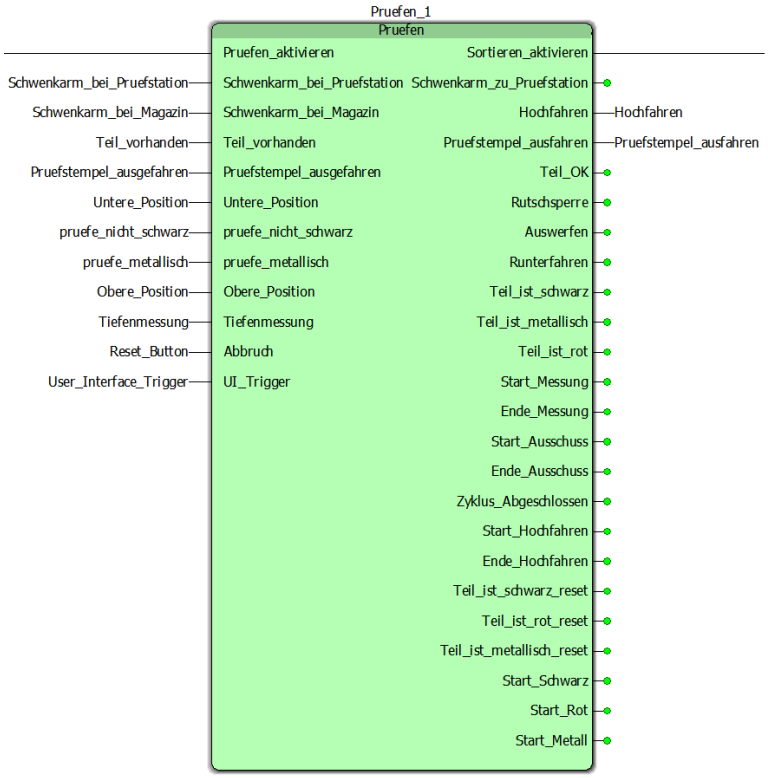

Abbildung 5.2.: Funktionsbaustein Prüfen

Der Funktionsbaustein *Prüfen* ist in Ablaufsprache programmiert und steuert die Prüfstation der Anlage. Im Startschritt werden alle eventuell gesetzten Variablen zurückgesetzt. Der zweite Schritt wird von der Variable *Pruefen_aktivieren* im Funktionsbaustein *Verteilen* aktiviert. In diesem Schritt befindet sich der Funktionsbaustein im aktivierten Zustand und der Ablauf wird beim Erfüllen aller prozessabhängigen Bedingungen gestartet. Das Aktivieren des Funktionsbausteins ist notwendig, damit dessen Ablauf nicht an einer anderen Stelle des Prozesses, an der unter Umständen die gleichen Bedingungen erfüllt sind, parallel zu einem anderen

Funktionsbaustein gestartet wird. Auch in diesem Baustein werden die Schritte nach Erfüllen aller prozessabhängigen Bedingungen von der Variable *UI_Trigger* weitergeschaltet.

Wurde das vom Schwenkarm abgelegte Teil an der Prüfstation erkannt, beginnt in der unteren Position die Messung der Farbe und des Materials. Ist die Messdauer abgelaufen und der Schwenkarm wurde zum Magazin bewegt, da er ansonsten beim Hochfahren der Plattform im Weg wäre, fährt diese nach oben. Ist sie in der oberen Position angekommen, wird der Prüfstempel zum Messen der inneren Absatzhöhe ausgefahren. Ist die Messdauer abgelaufen, wird der Stempel wieder eingefahren. An dieser Stelle tritt eine Alternativ-Verzweigung auf. War die Absatzhöhe des Teils innerhalb der Soll-Grenzen, wird es auf die Rutsche befördert und der Kolben zum Stoppen des Teils vor dem Rutschenende ausgefahren. War die Absatzhöhe nicht in Ordnung, wird das Teil mit der Plattform wieder nach unten gefahren und dort als Ausschuss aussortiert.

War das zu prüfende Teil in Ordnung, endet der Ablauf mit dem Aktivieren des Funktionsbausteins *Sortieren* über einen Puls der Variable *Sortieren_aktivieren*. War das Teil nicht in Ordnung, wird der Prozesszyklus mit der Variable *Zyklus_Abgeschlossen* beendet.

Tabelle 5.1.: Sensorkombinationen für Eigenschaftsbestimmung

nicht schwarz	metallisch	Eigenschaft
0	0	schwarz
0	1	Fehler
1	0	rot
1	1	metallisch

In den meisten Fällen werden die von den Schritten durchgeführten Aktionen durch einfaches Setzen bzw. Zurücksetzen oder zeitverzögertes Setzen von Variablen definiert. In diesem Funktionsbaustein werden vier Aktionen jedoch auch durch Anweisungen in Strukturiertem Text programmiert. Soll eine Aktion abhängig von Vergleichen mit anderen Variablen oder Konstanten sein, ist dies einfach in Strukturiertem Text zu realisieren. In dem Beispiel in Abbildung 5.3 wird das Teil auf seine Farbe und sein Material geprüft. Aus den möglichen Kombinationen der zwei Sensorwerte für „Teil ist nicht schwarz“ und „Teil ist metallisch“ wird bestimmt, ob das Teil schwarz, rot oder metallisch ist oder ob ein Fehler vorliegt (siehe Tabelle 5.1). Im letzten Fall wird das Teil als Ausschuss markiert.

```
1  IF  pruefe_nicht_schwarz = FALSE AND pruefe_metallisch = FALSE THEN
2      Teil_ist_schwarz := TRUE;
3  ELSIF pruefe_nicht_schwarz = TRUE AND pruefe_metallisch = FALSE THEN
4      Teil_ist_rot := TRUE;
5  ELSIF pruefe_nicht_schwarz = TRUE AND pruefe_metallisch = TRUE THEN
6      Teil_ist_metallisch := TRUE;
7  ELSE
8      Teil_OK := FALSE;
9  END_IF;
```

Abbildung 5.3.: ST *Teil_prüfen_1*

Die Farbe des zu prüfenden Teils und ob dieses OK ist, wird außerhalb des Funktionsbausteins in RS-Blöcken gespeichert, damit diese auch von anderen POEs des Programms an beliebiger Stelle zurückgesetzt werden können. Konkret findet dies beispielsweise beim Abschließen eines Zyklus durch den Funktionsbaustein *Sortieren* statt oder auch durch *Reset*.

Das Ablaufdiagramm und die Deklarationstabelle des Funktionsbausteins *Pruefen*, sowie die weiteren ST-Anweisungen befinden sich im Anhang (siehe Abbildungen A.8 bis A.12).

Sortieren

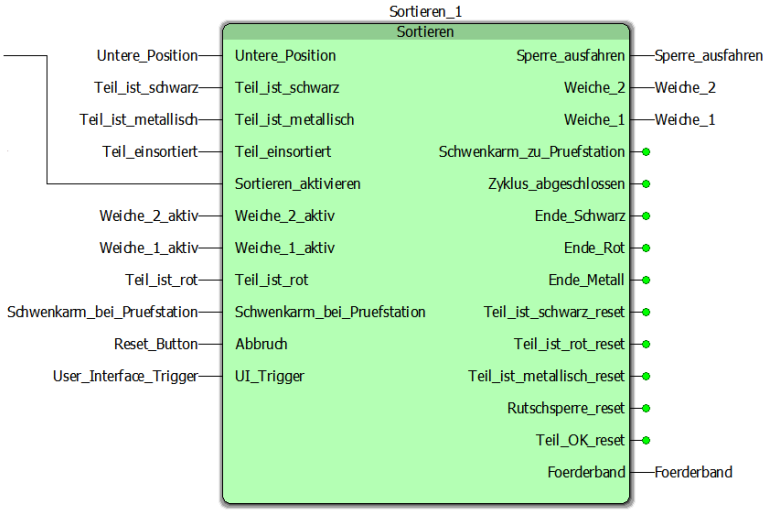

Abbildung 5.4.: Funktionsbaustein Sortieren

Der Funktionsbaustein *Sortieren* ist in Ablaufsprache geschrieben und steuert die Sortierstation am Ende des Prozesses. Im Startschritt werden alle eventuell gesetzten Variablen zurückgesetzt. Der zweite Schritt wird von der Variable *Sortieren_aktivieren* im Funktionsbaustein *Pruefen*

aktiviert und erfüllt den gleichen Zweck wie auch in diesem. Der folgende Ablauf wird nach Erfüllen aller prozessabhängigen Bedingungen von der Variable *UI_Trigger* weitergeschaltet.

Der Ablauf teilt sich anschließend bei einer Alternativ-Verzweigung in drei Zweige, von denen einer über die Variablen *Teil_ist_schwarz*, *Teil_ist_rot* oder *Teil_ist_metallisch* aktiviert wird und das Teil anschließend in die jeweilige Lagerrutsche einsortiert. Dazu wird das Förderband aktiviert und gegebenenfalls die Weiche am jeweiligen Lager ausgefahren. Soll das Teil in das erste Lager sortiert werden, wird gleichzeitig die Sperre an der ersten Rutsche aktiviert, welche ein Umkippen des Werkstücks verhindern soll und eingefahren wird, sobald das Teil auf dem Förderband liegt. Hat die Weiche ihre ausgefahrenen Position erreicht, wird die Rutschsperre gelöst und anschließend gewartet, bis die Lichtschranke das Teil auf einer Lagerrutsche registriert hat. Danach wird die Anlage in den Anfangszustand versetzt, alle Werkstückeigenschaften zurückgesetzt und die Variable *Zyklus_abgeschlossen* gesetzt, um den Beginn eines neuen Zyklus auszulösen.

Das Ablaufdiagramm und die Deklarationstabelle des Funktionsbausteins *Sortieren* befinden sich im Anhang (siehe Abbildungen A.13 und A.14).

Initialisieren

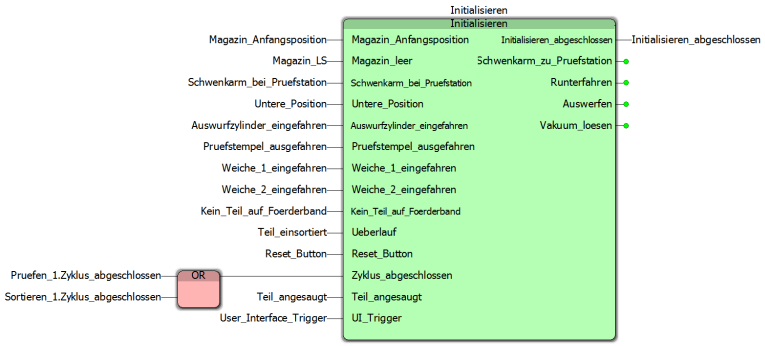

Abbildung 5.5.: Funktionsbaustein Initialisieren

Der Funktionsbaustein *Initialisieren* hat die Aufgabe, beim Einschalten der Anlage oder nach einem Reset, die Startposition anzufahren und zu prüfen, ob alle Startbedingungen erfüllt sind. Ist dies der Fall, wird die Ausgangsvariable *Initialisieren_abgeschlossen* TRUE gesetzt, wodurch der Funktionsbaustein *Verteilen* aktiviert wird.

Anders, als der Name und die Hauptfunktion vermuten lassen, wird der Block jedoch nicht nur am Anfang und nach einem Reset ausgeführt, sondern nach jedem abgeschlossenen

Prozesszyklus, um zu überprüfen, ob sich die Anlage wieder in Startposition befindet (Abbildung A.15) und um den Funktionsbaustein *Verteilen* zu aktivieren (Abbildung A.17).

Sollte eine Startbedingung nicht erfüllt sein, wird der Initialisierungsablauf aktiviert (siehe Abbildung A.16). Hier wird als erstes die Position der Hebeplattform geprüft, sollte diese sich nicht unten befinden, wird zur Sicherheit der Schwenkarm zum Magazin gefahren, bevor die Plattform heruntergefahren und in den Zweig für die untere Position gesprungen wird. Dort wird die Position des Schwenkarms geprüft. Befindet sich dieser nicht bei der Prüfstation, wird geprüft, ob bei der Prüfstation schon ein Teil vorhanden ist. Ist dies der Fall, wird zu Schritt *S061* gesprungen, wo das Teil ausgeworfen und anschließend wieder in den Schritt *Unten* gesprungen wird. Hier wird erneut die Position des Schwenkarms geprüft und falls dieser nicht bei der Prüfstation, dort hingefahren. Jetzt wird geprüft, ob ein Teil am Saugnapf ist und falls ja, dieses gelöst. Sollte ein Teil jetzt auf der Plattform liegen, wird der Schwenkarm weggefahren und das Teil ausgeworfen und anschließend wieder zu Schritt *Unten* gesprungen, wodurch der Arm wieder zur Prüfstation gefahren und anschließend das Initialisieren beendet wird.

LED_Steuerung

Der Funktionsbaustein *LED_Steuerung* ist in Ablaufsprache geschrieben und läuft parallel zum Kernprozess. Er wertet sogenannte Flags aus, Variablen die im Prozess an Stellen gesetzt werden, an denen der Visualisierung neue Zustände signalisiert werden sollen und deren Namen mit „Start_“ bzw. „Ende_“ beginnen.

Der Ablaufplan des Funktionsbausteins (siehe Anhang, Abbildungen A.18 und A.19) bildet den Prozess in schlanker Form ab, in dem die Flags als Weiterschaltbedingungen die Schritte triggern, welche die OPC-Variablen zur Visualisierungssteuerung setzen. Durch diese Form der Visualisierungssteuerung werden eine höhere Übersichtlichkeit und bessere Kontrolle über die Variablen erreicht.

Reset

Die Hauptfunktion des Funktionsbausteins *Reset* ist, die im Programm per RS-Block gesetzten Prozessvariablen bei einem Abbruch des laufenden Prozesszyklus zurückzusetzen, indem er auf den *RESET1*-Eingang jedes RS-Blocks einen Puls gibt.

Seine Ausführung wird von der Variable *Reset_Button* ausgelöst, welche mit der Bedienschnittstelle gesetzt werden kann. Diese Variable führt des Weiteren in allen anderen Funktionsbausteinen zum Abbruch des jeweiligen Ablaufs. Dazu wurde an jeder Transition ein Alternativzweig mit der Bedingung der Abbruchsvariablen eingebaut, welcher den jeweiligen Ablauf

mit einem Sprung in den Initialschritt versetzt. Durch das Positionieren der Abbruch-Zweige auf der linken Seite der normalen Bedingung, wird diesen eine höhere Priorität zugewiesen, sollten die Weiterschaltbedingungen beider Zweige erfüllt sein.

TON

Der TON-Funktionsbaustein ist eine Einschaltverzögerung. Eine steigende Flanke an Eingang *IN* wird um eine an Eingang *PT* übergebene Zeit verzögert an Ausgang *Q* ausgegeben. Der Ausgang *ET* gibt die seit einer steigenden Flanke am Eingang *IN* verstrichene Zeit aus. Eine fallende Flanke an Eingang *IN* wird ohne Verzögerung an Ausgang *Q* weitergegeben.

Die Einschaltverzögerung wird beispielsweise genutzt, um das kurzzeitige Signal von der Magazin-Lichtschranke abzufangen, welches beim Ausschieben eines Teils aus dem Magazin auftritt, um keine „Magazin leer“-Meldung auszulösen.

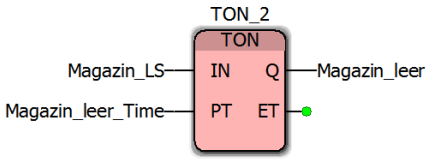

Abbildung 5.6.: TON Block

RS

Der RS-Funktionsbaustein (Flipflop) ist ein Speicher, der durch einen Impuls am Eingang *SET* gesetzt wird und am Ausgang *Q1* ein Signal ausgibt, bis durch ein Signal am Eingang *RESET1* der Speicher und damit das Signal an Ausgang *Q1*, zurückgesetzt wird.

Der RS-Funktionsbaustein wird beispielsweise genutzt, um einen Aktor aus einem Funktionsbaustein einzuschalten und aus einem anderen auszuschalten.

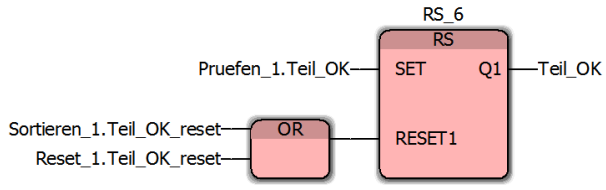

Abbildung 5.7.: RS Block

5.3.4. Funktionen

Funktionen können unterschiedlich viele Eingangsparameter besitzen, haben aber stets nur einen Ausgabewert und können Werte nicht über mehrere Zyklen speichern. Die Programmierung in Ablaufsprache ist für Funktionen unzulässig. Sehr häufige Anwendung finden Funktionen als vorgegebene Basisfunktionen wie Datentypkonvertierungen, mathematische Operationen, Vergleiche oder Verknüpfungen.

OR

Das Ausgangssignal einer OR- oder ODER-Funktion hat den Wert „1“, wenn einer der Eingangswerte „1“ ist.

Die OR-Verknüpfung wird beispielsweise dazu genutzt, um in Ablaufsprache einen nachfolgenden Steuerungsschritt in Abhängigkeit von einer von mehreren Bedingungen zu aktivieren oder um einen Ausgang anzusteuern, wenn dieser von mehreren Funktionsbausteinen aktiviert werden kann.

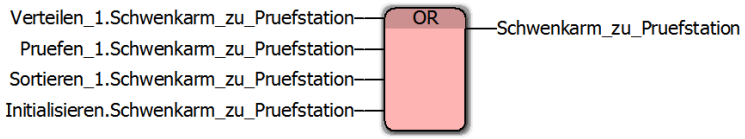

Abbildung 5.8.: OR-Funktion zur Steuerung einer Prozessvariable

AND

Das Ausgangssignal einer AND- oder UND-Funktion hat den Wert „1“, wenn beide Eingangswerte „1“ sind.

Die AND-Verknüpfung wird beispielsweise dazu genutzt, um in Ablaufsprache einen nachfolgenden Steuerungsschritt in Abhängigkeit von mehreren Bedingungen zu aktivieren.

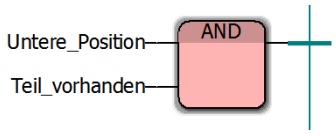

Abbildung 5.9.: AND-Funktion zur Verknüpfung mehrerer Bedingungen eines Gatters in AS

5.3.5. Buskonfiguration

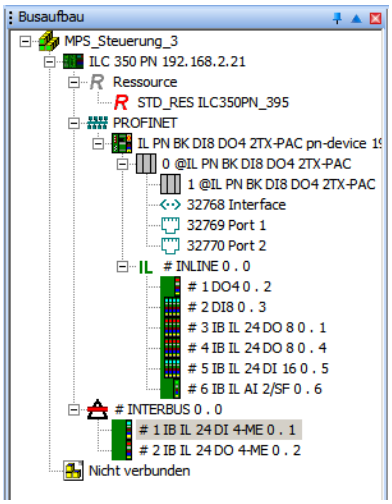

Abbildung 5.10.: Busaufbau

Um das Projekt auf die SPS übertragen zu können, muss diese zunächst korrekt angeschlossen und konfiguriert werden. Zur Konfiguration über Ethernet muss die IP-Adresse der SPS bekannt sein. Ist dies nicht der Fall, muss die Konfiguration zunächst über die serielle Schnittstelle stattfinden und mindestens die IP-Adresse festgelegt werden.

Ist die IP-Adresse bekannt, kann die SPS in PC WORX über Ethernet konfiguriert werden. Dazu muss im Gerätekatalog oder bei der Projekterstellung die verwendete Steuerung mit passender Firmware ausgewählt werden. In der Buskonfiguration wird im Fenster „Gerätedetails“, unter der Registerkarte „Kommunikation“ die bekannte IP-Adresse eingetragen, die Verbindung getestet und bei Erfolg übernommen. Die Verbindungskonfiguration findet sich nun in den IP Einstellungen wieder, wo auch der DNS Name der SPS vergeben werden kann.

Als nächstes findet die eigentliche Buskonfiguration im Fenster „Angeschlossener Interbus statt“. Als „Ausgewählte Steuerung“ wird nun die vorher verbundene SPS ausgewählt. PC WORX stellt die Verbindung her und zeigt erkannte Interbus-Module in grauer Farbe an. Beim Klick auf die Schaltfläche „In Projekt übernehmen“ muss man noch die genaue Bezeichnung der angeschlossenen I/O-Klemmen angeben, danach wird die Buskonfiguration in den Busaufbau übernommen.

Die Konfiguration von Profinet-Geräten erfolgt durch Rechtsklick auf „PROFINET“ im Strukturbaum des Busaufbaus und anschließende Auswahl von „PROFINET einlesen...“. Korrekt

verbundene Profinet-Geräte werden so automatisch erkannt, die vollständige Bezeichnung muss noch ausgewählt werden.

Im Strukturbaum wird nun wie in Abbildung 5.10 der komplette Busaufbau des Projekts angezeigt. Unter den verwendeten SPSen werden die über Interbus und Profinet angeschlossenen Geräte mit ihren I/O-Anschlüssen angezeigt.

5.3.6. Prozessdatenzuordnung

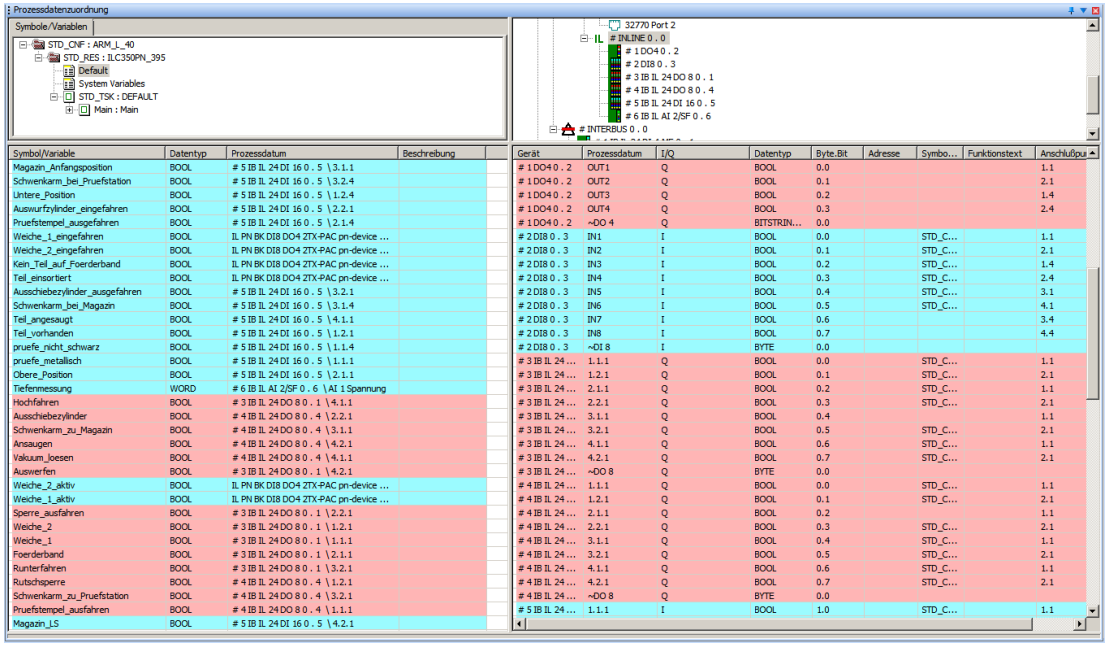

Abbildung 5.11.: Prozessdatenzuordnung

Bei der Prozessdatenzuordnung werden die globalen Variablen der Programme (in Abbildung 5.11 auf der linken Seite) mit den Adressen der physischen Klemmen der SPS (in der Abbildung auf der rechten Seite) verbunden. Dazu wird das „Prozessdatum“ einer Klemme der vorher definierten globalen Variable, welche den angeschlossenen Aktor oder Sensor im Programm widerspiegelt, in der Standard-Ressource per Drag&Drop zugewiesen.

6. Prozessvisualisierung

6.1. Allgemein

Die Prozessvisualisierung dient der grafischen Darstellung, Verfolgung und Überwachung eines Prozesses in bspw. der Automatisierungstechnik oder Verfahrenstechnik, im produzierenden Betrieb oder auch in der Forschung. Sie stellt die Arbeitsschritte einer Anlage als Schema im aktuell vorliegenden Zustand dar. Dabei kann die Prozessvisualisierung unterschiedlich detailliert sein.

Die in Automatisierungsanwendungen zahlreich gemessenen Sensordaten können, aber müssen nicht, alle dargestellt werden. Ausschlaggebend ist dabei, welche Information und welcher Erkenntnisgewinn aus diesen abgeleitet werden kann. Ist der Erkenntnisgewinn aus der Verknüpfung eines Wertes mit anderen Bedingungen des Prozesses zu einem System größer, so wird nicht jeder Sensorwert explizit angezeigt. Es müssen der Visualisierung jedoch zu jeder Zeit alle benötigten oder kritischen Werte entnehmbar sein.

6.2. Programmierung

Die Programmierung der Visualisierung wird in LabVIEW an zahlreichen Punkten unterstützt. So sind Funktionen zur Netzwerkkommunikation insbesondere mit OPC-Servern, visualisierungstypische Anzeige- und Bedienelemente wie Regler, Diagramme oder Füllstandsanzeigen, häufig benötigte Auswerte-Algorithmen oder Datenmanagementfunktionen integriert. Da ein Schwerpunkt der Programmierung mit LabVIEW bei der Datenerfassung und -verarbeitung liegt, unterstützt das Konzept des Datenflussdiagrammes das Hineindenken in ein Programm und seinen Ablauf.

6.2.1. Frontpanel

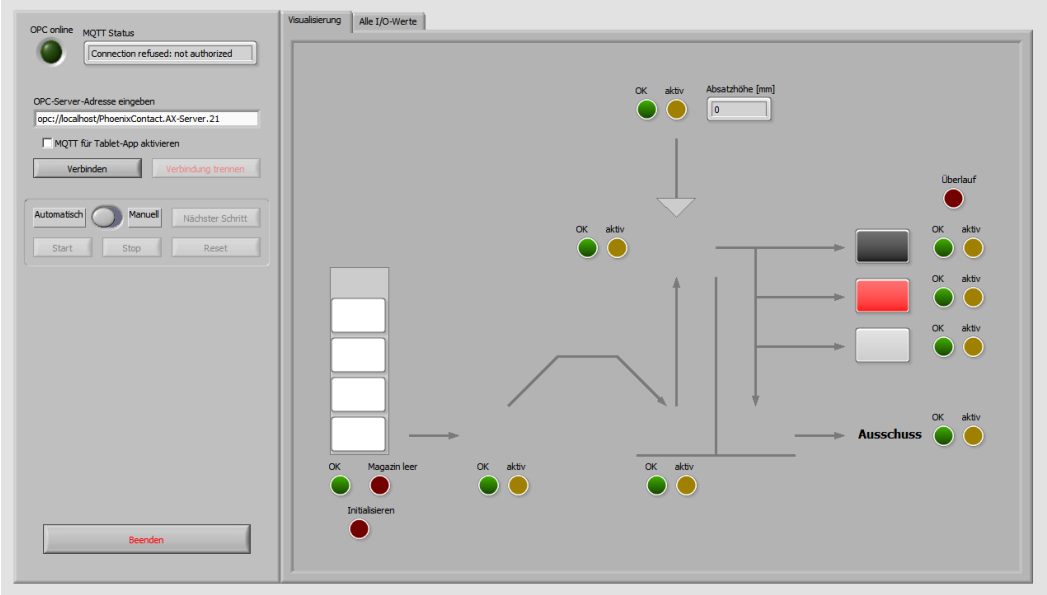

Abbildung 6.1.: Frontpanel

Das Frontpanel ist die Bedienoberfläche der Prozessvisualisierung. Auf ihr werden Bedien- oder Anzeigeelemente positioniert oder Gestaltungselemente angeordnet. Jedes interaktive Element verfügt über einen Anschlusspunkt im Blockdiagramm, über den es in das Programm eingebunden werden kann. Zur Visualisierung stehen entweder Anzeigeelemente wie LEDs, Graphen oder Textanzeigen zur Verfügung. Alternativ können programmatisch auch Eigenschaften von Gestaltungselementen verändert werden, um visuelle Effekte zu erzeugen.

6.2.2. Blockdiagramm

Im Folgenden sind Ausschnitte bzw. Unterprogramme, sogenannte SubVIs, des Programms beschrieben. Das gesamte Blockdiagramm des MPS-Panels befindet sich im Anhang (Abbildungen A.20 bis A.23).

Initialisieren

Das Blockdiagramm besteht aus mehreren verschachtelten Programmstrukturen und beginnt mit einer *Flachen Sequenz*. Die Flache Sequenz ist eine Programmstruktur, in der dem Programmablauf eine feste Reihenfolge zugewiesen wird. Die Sequenzfenster werden von links nach

rechts abgearbeitet. Diese äußerste Programmstruktur wird in diesem Fall dazu genutzt, um bestimmte Funktionen direkt nach Programmstart oder unmittelbar vor dem Beenden des Programmes auszuführen. Sie übernimmt damit eine Art Initialisierungsfunktion. Im wesentlichen werden hier Initialisierungswerte von Bedienelementen festgelegt.

Verbindungsaufbau

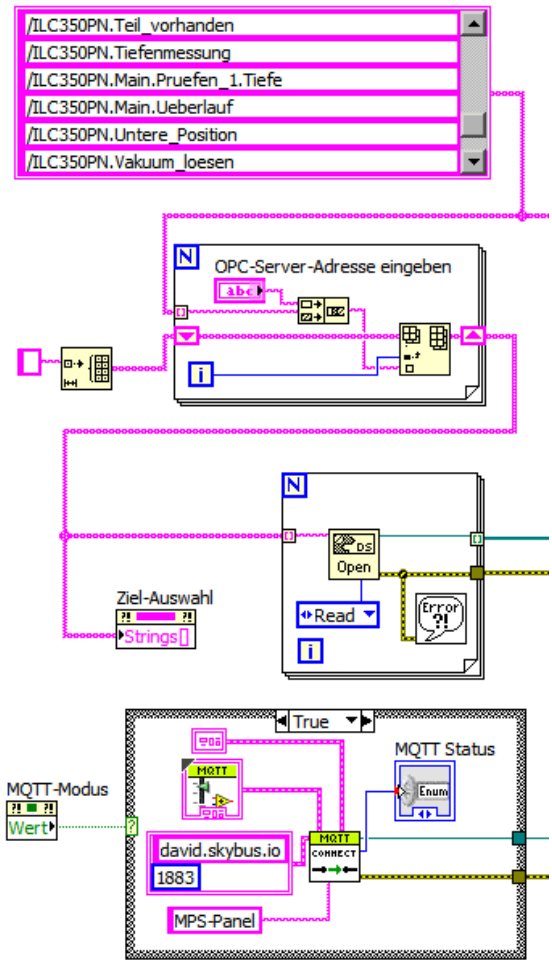

Abbildung 6.2.: Verbindungsaufbau

Im zweiten Sequenzfenster beginnt eine While-Schleife, welche das Programm bis zum Abbruchkriterium, das Betätigen des „Beenden“-Knopfes, ausführt. In der While-Schleife befindet sich eine Case-Struktur, die durch Betätigen des „Verbinden“-Knopfes den True-Case aktiviert.

Im True-Case wird wiederum eine Flache Sequenz betreten, in deren ersten Fenster Bedienelemente aktiviert bzw. deaktiviert werden. Der größte Teil des Programmes befindet sich in dem folgenden Sequenzfenster. Hier befindet sich ein Array, das eine Liste aller relevanten Items auf dem OPC-Server enthält. Die Items werden mit der zuvor eingegeben OPC-Server-Adresse verknüpft und anschließend Verbindungen zu diesen über die LabVIEW-Funktion *Datasocket: Öffnen*, in einer durch das Adressen-Array auto-indizierten For-Schleife, hergestellt. Die For-Schleife gibt ein Array mit den Verbindungs-IDs aus. Parallel dazu wird bei aktiviertem MQTT-Modus eine Verbindung zum MQTT-Broker hergestellt und ebenfalls die Verbindungs-ID weitergegeben.

Werte vom OPC-Server lesen

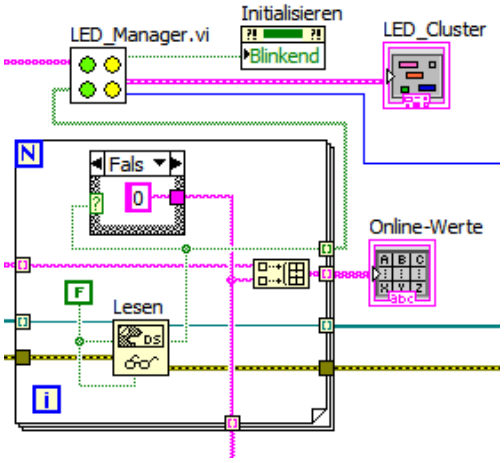

Abbildung 6.3.: OPC Lesen

Sind diese Schritte ausgeführt, liegen alle Eingangswerte für die folgende While-Schleife vor, wodurch diese betreten wird. Hier werden nun wiederholt die Werte vom OPC-Server gelesen und eventuell an den MQTT-Broker weitergeleitet. Dazu wird das Array mit den Verbindungs-IDs an eine auto-indizierte For-Schleife mit der Funktion *Datasocket: Lesen* übergeben. In der For-Schleife wird der vom Server gelesene Wert einerseits als boolescher Wert in ein Array geschrieben und andererseits in einen String konvertiert und anschließend mit dem zugehörigen Item-Namen in ein Array geschrieben wird, welches nach Durchlauf der Schleife in eine Tabelle geschrieben wird.

Die Elemente des booleschen Arrays werden vom SubVI *LED_Manager* den zugehörigen Anzeigeelementen des Frontpanels zugeordnet.

SubVI LED_Manager

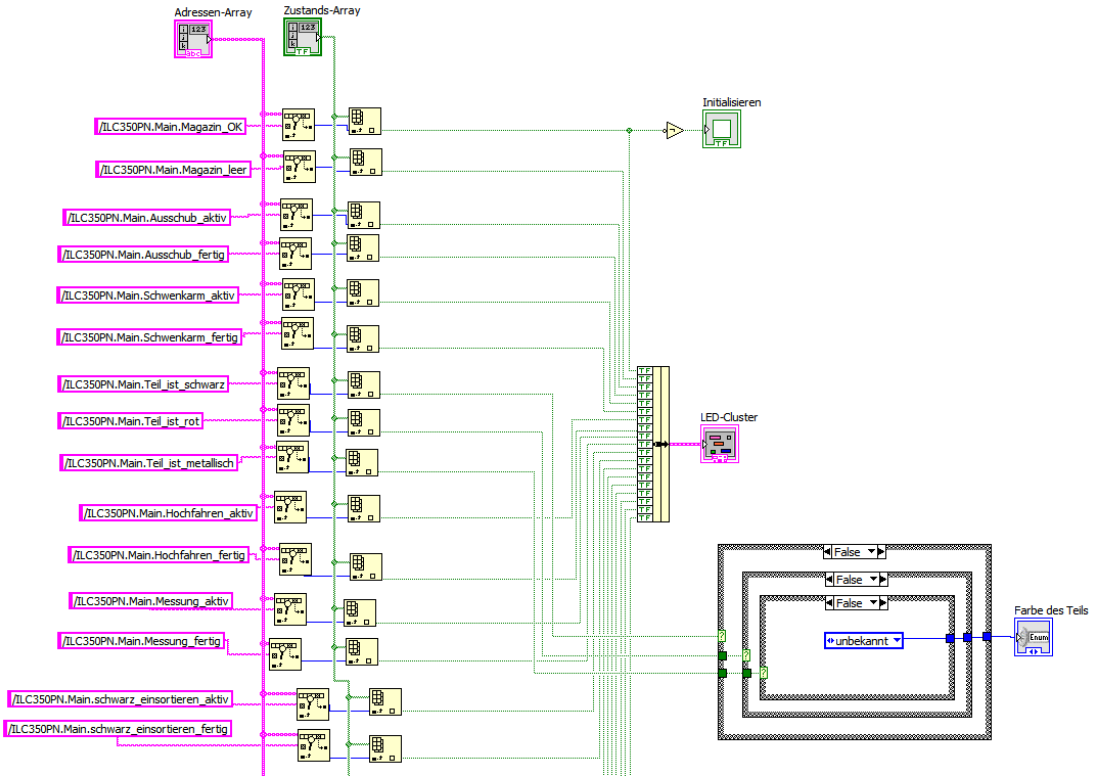

Abbildung 6.4.: LED_Manager

Der *LED_Manager* ist ein SubVI, an welches zunächst das OPC-Adressen-Array und das Item-Zustands-Array übergeben werden. Da die Werte in der Reihenfolge des Adressen-Arrays vom OPC-Server gelesen und in das Zustandsarray geschrieben wurden, haben die Werte und die zugehörigen Item-Namen in den zwei Arrays die gleiche Reihenfolge. Um jetzt einem Anzeigeelement der Prozessvisualisierung den zugehörigen Wert zuzuordnen, wird das Adressen-Array nach dem Namen des betreffenden Items durchsucht und der Index ausgegeben. Aus dem Zustandsarray wird nun der Wert mit dem vorherig ermittelten Index gesucht und an die Anzeige weitergegeben. Die Vielzahl an Anzeigeelementen wurden, der Übersichtlichkeit wegen, in einem Cluster zusammengefasst.

Werte über MQTT veröffentlichen und empfangen

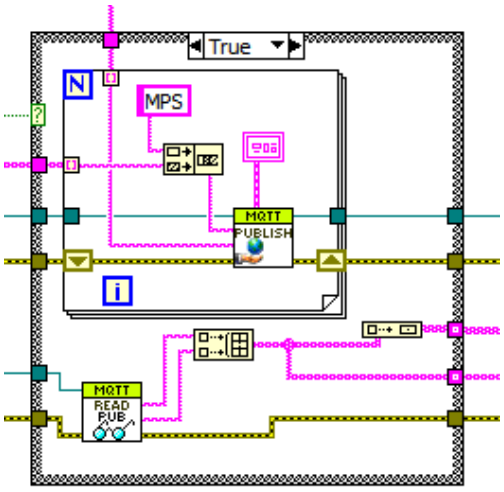

Abbildung 6.5.: SubVI *MQTT_Pub_and_Sub.vi*

Ein Teil der vom OPC-Server gelesenen Daten wird bei aktiviertem MQTT-Gateway für die Tablet-App an den MQTT-Server gesendet. Im MQTT Protokoll wird dieser Vorgang „publish“ (dt. veröffentlichen) genannt, da der MQTT-Server die aktuellen Werte an registrierte „Subscriber“ (dt. Abonnenten) automatisch weiterleitet.

Dazu werden die Werte des Arrays der OPC-Daten im SubVI *MQTT_Pub_and_Sub* in einer FOR-Schleife einzeln an das SubVI *MQTT_Publish* übergeben. Ebenfalls wird der Name, unter dem der Wert auf dem Server hinterlegt werden soll, als sogenanntes *Topic* übergeben. Das *Topic* wird aus dem Zusatz „MPS“ und dem Item-Namen zusammengesetzt.

Im SubVI *MQTT_Pub_and_Sub* werden ebenfalls von der App gesendete und vom Server weitergeleitete Kommandos empfangen. Das SubVI *MQTT_Read_Published_Message* gibt nach seiner Ausführung die vom Server empfangene Nachricht und das Topic aus, welche als Array zusammengefügt werden. Falls die App also einen Wert auf dem Server aktualisiert hat, würde dieser an dieser Stelle empfangen. Die empfangene Nachricht enthält als *Topic* den betreffenden Parameter (bspw. „MPS/App_Start“) und als *Message* den aktualisierten Wert, welcher „1“ oder „0“ sein sollte. Außerhalb des SubVIs *MQTT_Pub_and_Sub* werden diese Kommandos an die Bedienelemente des MPS-Panels gekoppelt und steuern damit direkt die Anlage.

Die Library mit den SubVIs der MQTT-Funktionen steht im Internet zur Verfügung: https://github.com/pda1982/Quaxo_MQTT_LabVIEW

Betriebsart einstellen

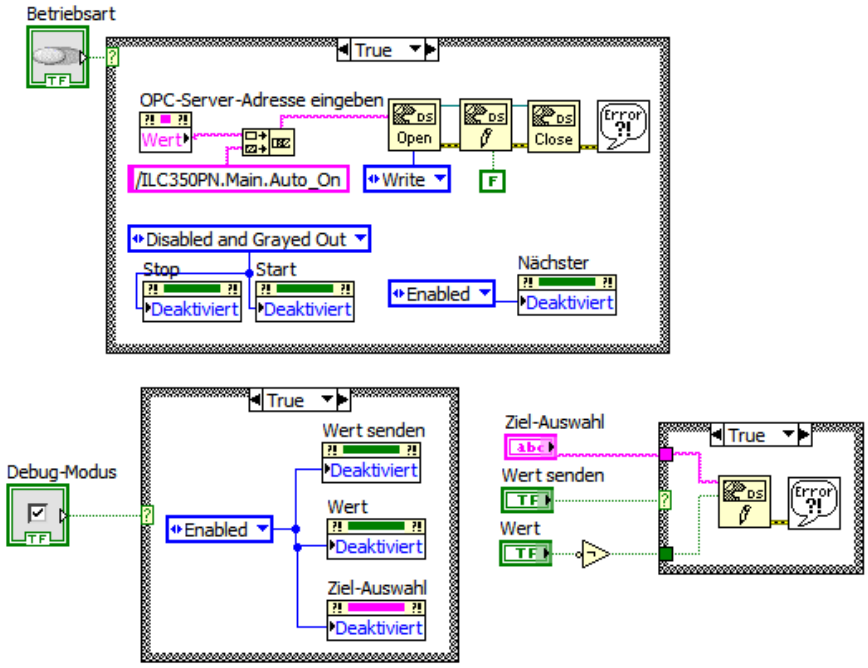

Abbildung 6.6.: Betriebsarten und Debug-Modus

Die Betriebsart wird mit einem Schiebeknopf eingestellt. Gibt dieser den Wert FALSE aus, werden in einer Case-Struktur die Schaltflächen für den Automatikmodus aktiviert und jene für den manuellen Modus deaktiviert. Ist der Schiebeknopf auf TRUE gesetzt, geschieht der entgegengesetzte Fall.

Im automatischen Modus wird der Prozess mit dem „Start“-Knopf gestartet, indem die OPC-Variable *User_Interface_Trigger* dauerhaft auf TRUE gesetzt wird. Angehalten wird der Prozess mit dem „Stop“-Knopf, der die Variable zurücksetzt. Im manuellen Modus hingegen wird diese Variable beim Betätigen von „Nächster Schritt“ nur kurzzeitig gesetzt, sodass ein einzelner Schritt getriggert wird.

Debug-Modus

Ist des Kontrollkästchen für den Debug-Modus aktiviert, werden die Kontrollelemente dafür von einer Case-Struktur freigeschaltet. Ansonsten sind diese ausgegraut und deaktiviert. Wird die Schaltfläche „Wert senden“ betätigt, gibt diese einen Puls an eine Case-Struktur, welche einmalig den ausgewählten Wert in die Ziel-Variable schreibt.

Absatzhöhe

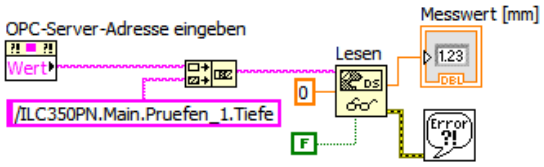

Abbildung 6.7.: Messwertanzeige

Die im Prozess gemessene Absatzhöhe des aktuellen Werkstückes ist der einzige Analogwert, der verarbeitet wird. Da alle anderen Daten als boolesche Werte vom OPC-Server gelesen und in ein Array geschrieben werden, muss der Messwert der Absatzhöhe separat verarbeitet werden, da ein Array immer nur Werte eines Datentyps erfassen kann. Der Messwert wird als Datentyp Double vom Server gelesen und im Anzeigeelement ausgegeben.

Visuelle Effekte

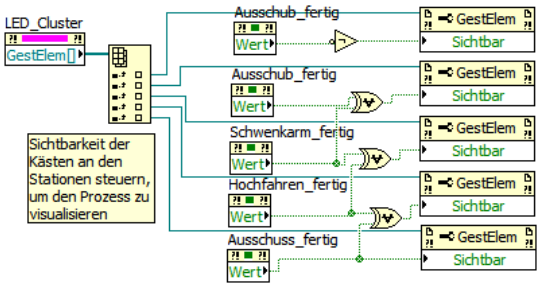

Abbildung 6.8.: Werkstück an aktueller Position einblenden

Zur Erzeugung visueller Effekte werden die Farbe und Position von Gestaltungselementen des Frontpanels programmatisch verändert. Der Eindruck eines sich von Station zu Station bewegenden Werkstücks wird durch das Ein- und Ausblenden unterschiedlicher Blöcke an den jeweiligen Positionen des Schemas, in Abhängigkeit von dem als letztes beendeten Prozessschritt, erzeugt. Dies geschieht über die Sichtbarkeits-Eigenschaftsknoten der Blöcke (siehe Abbildung 6.8). Die Farbe des Werkstücks wird im SubVI *LED_Manager* aus den Sensorwerten ermittelt und im Hauptprogramm an eine Case-Struktur übergeben, welche mit den Farb-Eigenschaftsknoten der Blöcke an jeder Position deren Farbe verändert (siehe Abbildung 6.9).

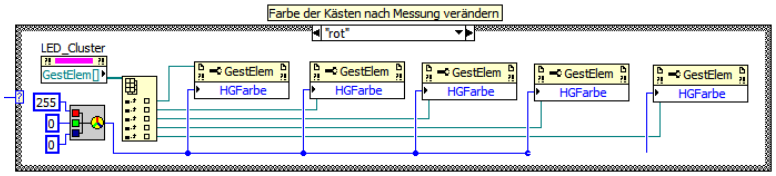

Abbildung 6.9.: Farbgebung der Werkstücke

6.3. Bedienung

Die Prozessvisualisierung wird mit der auf der CD vorhandenen Datei *MPS-Panel.exe* gestartet. Auf dem PC müssen die LabVIEW Runtime Engine und der Phoenix Contact OPC-Server installiert und für den Zugriff auf die SPS konfiguriert sein.

6.3.1. Verbindung zu OPC- und MQTT-Server herstellen

Nach dem Start der Visualisierungssoftware muss eine Verbindung zum OPC-Server und wahlweise zum MQTT-Broker hergestellt werden. Die Verbindung zum OPC-Server ist in jedem Fall zum Betrieb notwendig, um mit der SPS zu kommunizieren. Die Verbindung zum MQTT-Broker wird benötigt, wenn zur Überwachung und Steuerung auch die Tablet-App eingesetzt werden soll. In diesem Fall stellt die Software eine Verbindung zu einem Server her, der das MQTT-Protokoll unterstützt. (Der Standardserver ist cybus.rzbt.haw-hamburg.de, dieser kann nur in LabVIEW geändert werden. Hierbei ist auch die Anpassung der Connect Flags und von Username und Passwort zu beachten. Des Weiteren muss im HAW-Netz der TCP Port 1883 für die Verbindung zwischen dem Rechner, auf dem die Prozessvisualisierung läuft, und der Broker-URI freigegeben werden.)

Die Bedienung der Prozessvisualisierung am PC ändert sich durch das Aktivieren des MQTT-Gateways nicht.

6.3.2. Initialisieren

Solange die Lampe „Initialisieren“ blinkt, befindet sich die Anlage noch nicht im betriebsbereiten Zustand. Entweder wird die Anlage von der Steuerung noch in die Anfangsposition gebracht oder die Anlage ist nicht korrekt für den Betrieb eingerichtet. Zu überprüfen sind in diesem Fall folgende Punkte:

- alle Ventile geöffnet und Druck auf Kompressor

- kein Teil auf Förderband
- das Magazin ist nicht leer
- Lichtschranke der Lagerrutschen nicht blockiert
- Verbindung zwischen SPS und OPC-Server hergestellt

6.3.3. Betriebsarten

Bei erfolgreichem Verbinden zum OPC-Server leuchtet die LED oben links grün. Wenn die Anlage angeschlossen, mit dem PC verbunden und korrekt konfiguriert ist, bedeutet dies Einsatzbereitschaft. Sobald die Lampe „Initialisieren“ erloschen ist, kann der Prozess gestartet werden. Je nach gewählter Betriebsart, kann dies mit einem Klick auf „Start“ bzw. auf „Nächster Schritt“ geschehen.

„Start“ startet den Prozess im Automatikmodus und lässt diesen so lange laufen, bis „Stop“ oder "Reset“ gedrückt oder die Verbindung getrennt wird, das Magazin leer ist oder ein Fehler auftritt. „Nächster Schritt“ aktiviert im Manuellen Modus nur den nächsten Schritt im Prozess, sofern der vorherige vollständig ausgeführt wurde. Die den ausgewählten Modus nicht betreffenden Schaltflächen werden ausgegraut und deaktiviert.

6.3.4. Visualisierung

Die Visualisierung läuft von links nach rechts ab. Solange die Lampe „Initialisieren“ blinkt, befindet sich die Anlage nicht im Betriebsbereiten Zustand. Die Teile im Schema des Stapelmagazins sind weiß, da ihre Eigenschaften unbekannt sind. Das unterste Teil aus dem Magazin durchläuft das Prozessschema und nimmt an den Messstationen die gemessenen Eigenschaften an (Farbe). Die Absatzhöhe zeigt nach dem Messen die für das aktuelle Teil gemessene Höhe des inneren Absatzes an, welche das Ausschusskriterium darstellt. An den gelben und grünen LEDs zu jedem Schritt kann nachvollzogen werden, ob dieser gerade aktiv oder fertig ausgeführt ist.

6.3.5. Registerkarten

Über die Registerkarten kann zwischen „Visualisierung“ und „Alle I/O-Werte“ umgeschaltet werden. Unter „Visualisierung“ wird das Schema der Anlage dargestellt, mit dem aktuellen Zustand und den Eigenschaften des aktuellen Werkstückes, sofern diese schon geprüft wurden. Unter „Alle I/O-Werte“ befindet sich eine Liste aller über den OPC-Server ausgetauschten Variablen mit den aktuellen Werten.

6.3.6. Debug-Modus

Das Kontrollkästchen zum aktivieren des Debug-Modus befindet sich auf der Registerkarte „Alle I/O-Werte“ und dient der Manipulation von OPC-Variablen. Nach dem Aktivieren kann die als Ziel ausgewählte Variable durch „Wert senden“ auf den mit dem Kippschalter ausgewählten Wert gesetzt werden. So können bestimmte Zustände simuliert werden, um die LabVIEW Oberfläche oder das Verhalten der SPS zu testen. Dabei gilt es zu beachten, dass dies nur möglich ist, solange nicht die SPS die Kontrolle über die Variable hat, da sie ansonsten nach dem Manipulieren unmittelbar wieder überschrieben wird.

6.3.7. Verbindung trennen

Die Verbindungen zu den Servern können über „Verbindung trennen“ unterbrochen werden. Weder die Visualisierung am PC noch das Tablet haben nun Kontrolle über die Anlage. Sofern beim Trennen eine intakte Verbindung zur SPS besteht, wird vorher die Anlage angehalten. Die Schaltfläche „Beenden“ trennt die Verbindungen, falls verbunden und beendet das Programm.

7. Industrie 4.0

7.1. Was ist „Industrie 4.0“?

Der Begriff „Industrie 4.0“ wurde 2011 als Name einer Initiative aus Wirtschaft und Wissenschaft zur Erarbeitung von Umsetzungsempfehlungen der *Vierten industriellen Revolution* für die Bundesregierung bekannt. Er bezeichnet die Weiterentwicklung der produzierenden Wirtschaft auf der Grundlage von Vernetzung und kommunizierenden Systemen zu intelligenten Fabriken und „selbst organisierenden, unternehmensübergreifenden Wertschöpfungsnetzwerke“(Plattform Industrie 4.0, 2015, S. 8), sowie die im Zuge einer industriellen Revolution auftretenden Strukturwandlungsprozesse des gesellschaftlichen Lebens. Durch die Optimierung von Kosten, Verfügbarkeit und Ressourcenverbrauch und durch die Maximierung der Flexibilität in der Produktion wird versucht, den steigenden Löhnen und dem durch Kundenwünsche wachsenden Bedarf an individualisierter Fertigung zu begegnen. „Basis ist die Verfügbarkeit aller relevanten Informationen in Echtzeit durch Vernetzung aller an der Wertschöpfung beteiligten Instanzen sowie die Fähigkeit, aus den Daten den zu jedem Zeitpunkt optimalen Wertschöpfungsfluss abzuleiten.“(Plattform Industrie 4.0, 2015, S. 8). Mit den technologischen Errungenschaften des letzten Jahrzehnts, der flächendeckenden Internetverfügbarkeit, erweiterten Netzwerktechnologien für Telemetrie (Fernmessung), immer kleiner und kostengünstiger werdenden Kommunikationschips, die es ermöglichen jedem Objekt eine ID oder sogar IP-Adresse zuzuordnen, und den universellen Benutzerschnittstellen in Form von mobilen Endgeräten (Smartphones, Tablets), sind heutzutage alle Bedingungen für ein „Internet der Dinge“ (engl. Internet of Things, Abk.: IoT) erfüllt, in dem Geräte und Objekte ihre Zustandsinformationen bereitstellen und auf Veränderungen in ihrer Umgebung oder Prozesskette reagieren können. Nach Glanz und Jung (2010, S. 18) steht für diesen „automatisierten Datenaustausch zwischen Maschinen“ auch der Begriff „Machine-to-Machine“ (Abk.: M2M) wobei „es unerheblich ist, ob es sich um Maschinen im herkömmlichen Sinn (Motoren, Verkaufsautomaten) oder um virtuelle Maschinen (Software) handelt.“. M2M und IoT stellen damit zwei wichtige Standbeine der Industrie 4.0 dar, welche für die umfassende Idee der selbstständigen Organisation und Steuerung einer vertikal komplett integrierten Wertschöpfungskette steht.

7.2. Wie konnte „Industrie 4.0“ in die Arbeit eingebracht werden?

7.2.1. Remote Monitoring

Durch die Vernetzung im Zuge von Industrie 4.0 und die dadurch entstehende Verfügbarkeit der Telemetrie- und Sensordaten über das Internet, ist ein naheliegender erster Schritt zu deren Nutzung, sie Anzeigen zu können und über sie auch eine Steuerung der Anlage zu realisieren, die über einen Server erfolgt. Dies wurde in dieser Arbeit durch die Programmierung einer Anwendung für ein Android-Tablet realisiert (siehe Abschnitt 7.3).

Durch die Verwendung eines Web-Servers wäre es theoretisch möglich, die Anlage mit einem mobilen Endgerät von jedem Ort der Welt aus zu überwachen und zu steuern. Voraussetzung dafür wäre allerdings, dass die Prozessvisualisierung auf dem PC läuft und an die Anlage angeschlossen ist und dass jemand das Magazin stets von Hand nachfüllt. Daher bleibt dieser Nutzen in dieser Arbeit eher theoretischer Natur, demonstriert jedoch das Potential, das die Vernetzung im Zuge von „Industrie 4.0“ bietet. Produktionsketten, die auf unterschiedliche Standorte verteilt sind, können zentral gesteuert, aufeinander abgestimmt und die ordnungsgemäße Funktion überwacht und im Störfall sehr schnell automatisiert und global an jedem von den Auswirkungen betroffenen Standort reagiert werden, um Ausfälle und Verluste zu minimieren. Durch die kontinuierliche Auswertung von Sensordaten in Vorhersagemodellen sind unter Umständen sogar Verhaltensänderungen oder eine Effizienzverringerung einer Anlage registrierbar, auf welche noch vor dem Auftreten eines Störfalls reagiert und dieser durch die ordentliche Durchführung einer Wartung vermieden werden kann. Dieses Vorgehen wird als „Predictive Maintenance“ bezeichnet.

7.2.2. RFID-Konzept

Als weiteren Aspekt von Industrie 4.0 sollen die einzelnen Werkstücke über eine ID erkannt, überprüft und der Produktionsstatus vermerkt werden. Nach dem Ausschieben aus dem Magazin soll die ID des jeweiligen Werkstücks von einem Lesegerät erfasst und Daten wie Auftraggeber und Soll-Eigenschaften aus einer Datenbank abgerufen werden. So kann in einer modernen Produktionsanlage, in der immer weniger in Serie, sondern zunehmend nach Kundenwünschen individualisiert gefertigt wird, jederzeit der Fertigungsstatus abgerufen und die geforderten Eigenschaften überprüft werden. Dazu wird, wie vorher erwähnt, jedes Teil mit einer Identifikationsnummer ausgestattet. Für einen schnellen und reibungslosen Ablauf des Produktions- und der Logistikprozesse ist es wichtig, dass jedes Teil die ID bei sich trägt und

diese schnell und unkompliziert ausgelesen werden kann. Barcodes beispielsweise erfordern zum Auslesen Sichtkontakt und sind anfällig gegen Beschädigung oder Verschmutzung. Als Alternative dazu gibt es die RFID (engl. radio-frequency identification), welche elektromagnetisch von einem Transponder (sog. RFID Tag) ausgelesen werden kann. Diese Technologie funktioniert automatisch, berührungslos, in Bewegung und ohne nötigen Sichtkontakt.

RFID gibt es in drei Frequenzbereichen: 125 kHz (LF), 13,56 MHz (HF) und 890 MHz (UHF), welche für unterschiedliche Anwendungszwecke ausgelegt sind.

Die für die vorliegenden Fall relevanten Anforderungen sind folgende:

- RFID-Transponder im Etikettenformat, maximal 17x100 mm, Stärke <= 1 mm
- Erfassungsabstand zwischen 50 mm und 100 mm
- RFID-System mit Profinet-Anbindung

Mögliche Probleme:

- reduzierter Erfassungsabstand durch Werkstücke aus Kunststoff
- reduzierter Erfassungsabstand durch zusätzlich verchromte Werkstücke aus Kunststoff
- Beeinträchtigung des kapazitiven Näherungsschalters
- Beeinträchtigung des induktiven Näherungsschalters zur Erkennung verchromter Werkstücke
- Beeinträchtigung des optischen Sensors zur Grauanteilsbestimmung beim äußeren aufbringen der Etiketten
- Erfassungsprobleme bei Etiketten auf Umfang eines Zylinders mit r = 20 mm
- bei Verwendung von UHF: Begrenzung des Erfassungsbereich auf <= 100 mm möglich?
- bei Verwendung von LF: keine Transponder im Etikettenformat

Problembewertung

In Kontakt mit dem Support des Sensorherstellers Leuze electronic, der auch RFID-Geräte führt, konnte der Großteil der möglichen Probleme ausgeschlossen werden. Beeinträchtigungen des Erfassungsabstandes treten durch jede Art von Metall im Erfassungsbereich auf, wurde im Fall des verchromten Kunststoffes vom Support jedoch als nicht kritisch eingeschätzt.

Bedingt durch die angestrebte geringe Größe der Transponder wurde UHF als Alternative zu HF vorgeschlagen. Der normalerweise relativ große Erfassungsbereich von UHF Geräten ist nach Aussage des Supports, häufig per Konfiguration auf unter 100 mm reduzierbar.

Letztendlich haben Tests mit einem Muster-RFID-Tag jedoch ergeben, dass der induktive Näherungsschalter durch die Metallfolie des RFID-Tags, sowohl beim aufbringen auf die Außen- wie auch auf die Innenseite der Werkstücke, beeinflusst wird. Als Folge werden auch nicht verchromte Werkstücke als metallisch erkannt. Eine Integration der RFID-Technologie in die bestehende Anlage ist damit ohne Weiteres nicht realisierbar.

7.3. Tablet-App

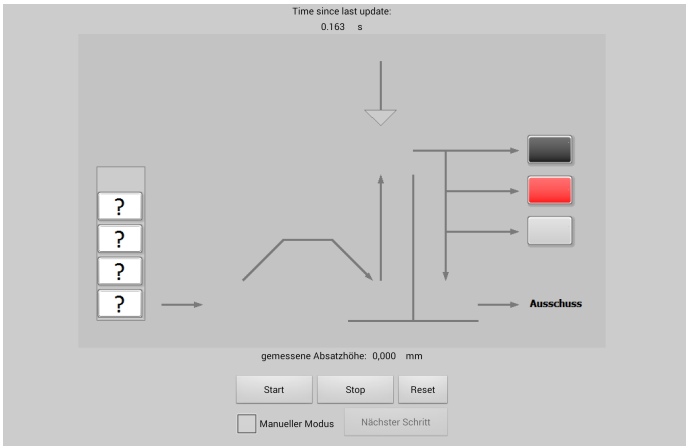

Abbildung 7.1.: Tablet App

Zur Anwendung des Remote Monitoring im Industrie 4.0-Kontext dieser Arbeit, wurde eine App für ein Android Tablet programmiert. Mit der Tablet App kann der Betrieb der Anlage über einen Web-Server theoretisch von jedem Ort aus überwacht und gesteuert werden. Die App zeigt dazu die Prozessvisualisierung an und Buttons zum Einstellen des Betriebsmodus zwischen „Automatisch“ und „Manuell“, Start und Stop des Automatikmodus, aktivieren des nächsten Schrittes im manuellen Modus, sowie einen Button zum Reset der Anlage.

In der Praxis befindet sich der Server im Rechenzentrum Berliner Tor und ist nur aus dem HAW-Netz erreichbar, könnte theoretisch jedoch auch leicht für einen externen Zugriff eingerichtet werden. Da dies praktisch für die Anlage im Labor jedoch nicht relevant ist, bleibt es ein theoretisches Modell zur Demonstration der Möglichkeiten mit Industrie 4.0 und wird nur im kleinen Maßstab zum Remote Monitoring innerhalb des HAW-Netzes genutzt.

7.3.1. Programmierung

Bedienoberfläche

Abbildung 7.2.: Liste der nicht sichtbaren Elemente

Die Gestaltung der Bedienoberfläche (Abbildung 7.1) mit dem App Inventor 2 erfolgt im sogenannten Designer. Bedien- und Anzeigeelemente wie Buttons, Labels oder Checkboxes werden in Arrangements angeordnet. Dazu werden auch nicht sichtbare Elemente eingefügt, welche Timer, Notifier-Funktionen oder Web-Verbindungen bereitstellen. Die Properties dieser sogenannten Nonvisibles (Abbildung 7.2) werden ebenfalls im Designer bearbeitet, wie beispielsweise die URL einer Web-Verbindung. Für jede benötigte URL kann hier eine Web-Verbindung erstellt werden, wodurch ermöglicht wird, im Block-Designer auf einfache Weise eine Verbindung auszuwählen.

Initialisieren

Abbildung 7.3.: Initialisieren

Die Anfragen der App an den Server erfolgen im HTTP-Protokoll. Um eine Authentifizierung vom Server zu erhalten muss den Anfragen ein Request Header angefügt werden, der den Benutzernamen und gegebenenfalls ein Passwort enthält. Die Authentifizierung wird in der Form Benutzername:Passwort Base64-codiert an der Server gesendet. Für den verwendeten Benutzernamen „MPS_App“ ohne Passwort erhält man Base64-codiert den String „TVBTX0FwcDo=“.

Der Request Header wird für jede Web-Verbindung beim Initialisieren der App erstellt. Beim Initialisieren werden auch Timer Zeiten für zeitgesteuerte Funktionen der App festgelegt.

Buttons

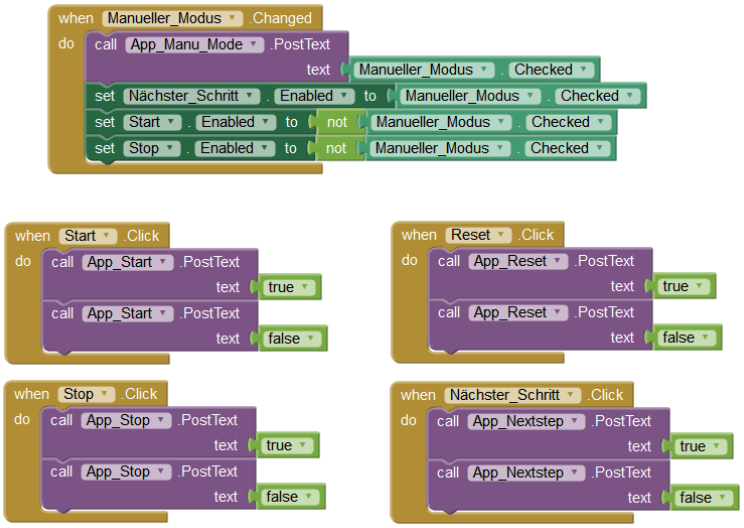

Abbildung 7.4.: Buttons

Beim Klick eines Buttons zur Steuerung der Anlage wird die PostText-Prozedur eines Web-Verbindungs-Elements aufgerufen und zunächst der Wert TRUE an die betreffende Adresse des HTTP-Servers geschickt. Sobald die Nachricht am Server angekommen ist, sendet dieser eine MQTT-Message an den LabVIEW-Client, welcher die jeweilige Funktion ausführt. Da die Message an LabVIEW unmittelbar nach Erhalt des HTTP-Requests veröffentlicht und von LabVIEW empfangen wird, kann der Wert auf dem Server auch direkt wieder zurückgesetzt werden, was von der App mit dem anschließenden HTTP-Request erfolgt.

Bei der Checkbox für den manuellen Modus wird immer nur eine PostText-Prozedur ausgeführt, die ihren aktualisierten Wert an den Server sendet. Zusätzlich werden hier noch die für den aktiven Modus benötigten Bedienelemente ein- bzw. die nicht benötigten ausgeblendet.

Datenabfrage vom Server

Abbildung 7.5.: Get Request

Der Zustand der Anlage wird halbsekündlich vom Server abgefragt. Der Server generiert zu jedem beim MQTT-Broker eingetragenen Wert eine URL, über die der aktuell vorhandene Wert auch über HTTP abgefragt werden kann. Die App führt bei Ablauf des Timers *Clock1* nacheinander Get-Requests für alle benötigten Werte aus. Sobald eine Antwort empfangen wurde, wird dies über den Parameter „GotText“ signalisiert, was die Auswertung der Antwort auslöst und bei neuen Werten die Visualisierung triggert.

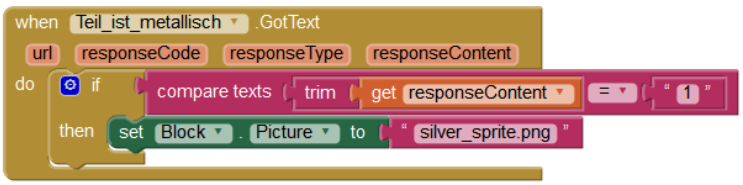

Abbildung 7.6.: GotText

Um den vollen Nutzen aus dem verwendeten MQTT-Broker zu ziehen, würde an dieser Stelle im Optimalfall nicht über HTTP sondern über MQTT mit dem Server kommuniziert. Die App bräuchte in diesem Fall nicht halbsekündlich den Server nach den aktuellen Werten fragen, sondern würde jeden neuen Wert automatisch zugestellt bekommen. Da der App Inventor jedoch ausschließlich HTTP unterstützt, musste darauf ausgewichen werden.

7.3.2. Bedienung

Die App wird auf einem Android Tablet mit der auf der CD vorhandenen Datei *MPS_Panel_v2.apk* installiert.

Die Visualisierung der Tablet-App orientiert sich an der LabVIEW Bedienoberfläche und funktioniert in der Bedienung ebenfalls analog zu dieser. Die Hauptfunktionen sind das Starten und Anhalten der Anlage, sowie die Resetfunktion und das Umschalten des Betriebsmodus.

Voraussetzung für die Benutzung der App ist, dass die LabVIEW-Anwendung läuft und als Gateway zwischen dem MQTT-Broker und dem OPC-Server fungieren kann. Die Oberfläche der App wird halbsekündlich vom Server geupdatet. Sollte ein Verbindungsproblem vorliegen, wird die in der obersten Zeile angezeigte „Time since last update“ deutlich über eine Sekunde ansteigen. In diesem Fall sind die Netzwerkverbindungen zwischen allen beteiligten Punkten zu überprüfen. Die Adresse des MQTT-Brokers, bzw. dessen HTTP-Servers, sowie Benutzername und Passwort sind fest implementiert und müssen bei einer Veränderung über den App Inventor angepasst werden.

8. Fazit

8.1. Zusammenfassung

In der Arbeit wurde das Programm zur Steuerung des MPS-Modells mit einer SPS erfolgreich realisiert. Die Anlage arbeitet wahlweise vollautomatisch oder halbautomatisch mit manuellem Weiterschalten der Prozessschritte. Mit einer Resetfunktion kann die Anlage in die Startposition versetzt werden. Die Realisierung der Steuerung erfolgte hauptsächlich in Ablaufsprache, was eine unkomplizierte und intuitive Umsetzung des Prozesses in ein Programm ermöglichte.

Die Realisierung der Prozessvisualisierung in LabVIEW hat ein auf der LabVIEW Runtime Engine ausführbares Programm hervorgebracht, welches über einen auf dem lokalen Rechner laufenden OPC-Server mit der SPS kommuniziert und darüber Zustandsinformationen erhält und Befehle sendet um die Anlage zu steuern. Des Weiteren ist in der Prozessvisualisierung ein Gateway zur Kommunikation mit einem MQTT-Broker integriert, um die Zustandsinformationen der Anlage universell verfügbar zu machen und damit die Basis für eine Integration von Industrie 4.0 zu schaffen. Über diese Schnittstelle können auch mit der Tablet App die Grundfunktionen zur Steuerung bedient und der Prozess live verfolgt werden.

Ein beträchtlicher Teil des Aufwandes fiel an, nachdem die groben Gerüste der Programme erstellt und die Grundfunktionen betriebsbereit waren. Der Teufel steckte hier, wie so häufig, im Detail und in den kleinen Bugs. Ein Fehler, der in seltenen Fällen zum Verbindungsverlust zwischen Gateway und MQTT-Broker führt, konnte bis zuletzt nicht behoben werden.

Dass ein Gateway zur Kommunikation mit dem MQTT-Broker in LabVIEW eingebaut werden musste, war eher eine zweckmäßige Notlösung, da der Zugriff auf die Prozessvariablen des OPC-Servers mit dem MQTT-Broker sich auf anderem Wege sehr schwierig gestaltet hätte und eine andere direkte Zugriffsmöglichkeit auf die Prozessvariablen der SPS nicht existiert. Erschwert wurde dieser Umstand noch dadurch, dass SPS und OPC-Server sich nicht wie der MQTT-Broker im HAW-Netz befinden, sondern in einem eigenen lokalen Netzwerk, zu dem der Zugang aus dem HAW-Netz nur über den PC erfolgt. Die Verwendung von OPC-Servern wird außerdem zunehmend von aktuelleren Technologien wie bspw. OPC UA abgelöst und bringt seine Schwierigkeiten bei der Integration in erweiterte Anwendungszwecke mit sich.

Der Versuch, die Visualisierung für das Tablet mit PC WebVisit browserbasiert mit einem Webserver auf der SPS zu realisieren, ist daran gescheitert, dass die verwendete relativ alte SPS diese neuere Möglichkeit mit der aktuellen Firmware nicht unterstützt.

Im Hinblick auf Industrie 4.0 wurden Überlegungen zur Integration eines RFID-Systems in die bestehende Anlage angestellt. Aufgrund der von der Metallfolie der RFID-Tags verursachten Problemen bei der Materialerkennung der Werkstücke, wurde das Konzept allerdings wieder verworfen.

8.2. Ausblick

Die Zukunft der Speicherprogrammierbaren Steuerung wird von einem starken Wandel geprägt sein. Durch Industrie 4.0 werden die Vernetzungs- und Flexibilitätsanforderungen an Automatisierungskomponenten immer wichtiger und in neuen Anlagen zu Standards werden (vgl. Lantermann, 2013). Ob die klassische SPS in diesem Wandel verschwinden und von intelligenten cloudbasierten Automatisierungen abgelöst werden wird hängt davon ab, ob es gelingt, durch Innovationen und Weiterentwicklungen ihre Nische offen zu halten. Diese Arbeit hat jedoch auch demonstriert, dass Industrie 4.0 Konzepte sich gut in bestehende Anlagen einpflegen lassen und diese theoretisch auch in selbstorganisierende Wertschöpfungsnetzwerke integriert werden können.

Ein Punkt, an dem die Arbeit fortzuführen wäre, ist zum einen die Wiederaufnahme des RFID-Konzepts. Das in dieser Arbeit behandelte Konzept umfasste die Anforderung, dass die bestehende Anlage nicht einschneidend verändert werden sollte. Es wäre allerdings denkbar, auf das Prüfen der Farbe und des Materials, bei welchem durch die Metallfolie der RFID-Etiketten Fehler auftraten, zu verzichten und die in einer Datenbank hinterlegten Eigenschaften des aktuellen Werkstücks über eine Abfrage mit dessen ID zu bestimmen, und nach der Prüfung der Absatzhöhe dementsprechend einzusortieren.

Ein weiterer Punkt könnte die Entwicklung einer umfangreicheren Tablet App sein, welche auch die Kommunikation über das MQTT-Protokoll erlaubt. Die Vorteile dieses Protokolls, welche auch in der komfortablen Handhabung trotz starker Entkopplung der Systeme bestehen, wurden bisher nur in der Kommunikation zwischen dem Gateway und dem Broker genutzt. Interessant könnte für diese Anwendung auch das neue Cloud-Gateway mit MQTT-Unterstützung für PROFINET-Netzwerke NT IQ52-FE-RE von Hilscher sein, welches bei Anfertigung dieser Arbeit jedoch noch nicht erhältlich war. Dieses würde das MQTT-Gateway in LabVIEW ersetzen und über Profinet die direkte Kommunikation zwischen der SPS und dem Cybus Server ermöglichen.

A. Anhang

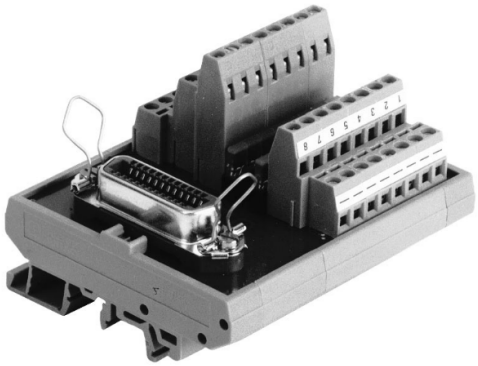

Abbildung A.1.: E/A-Terminal, Quelle: Festo Didactic E/A-Terminal-Datenblatt 034035

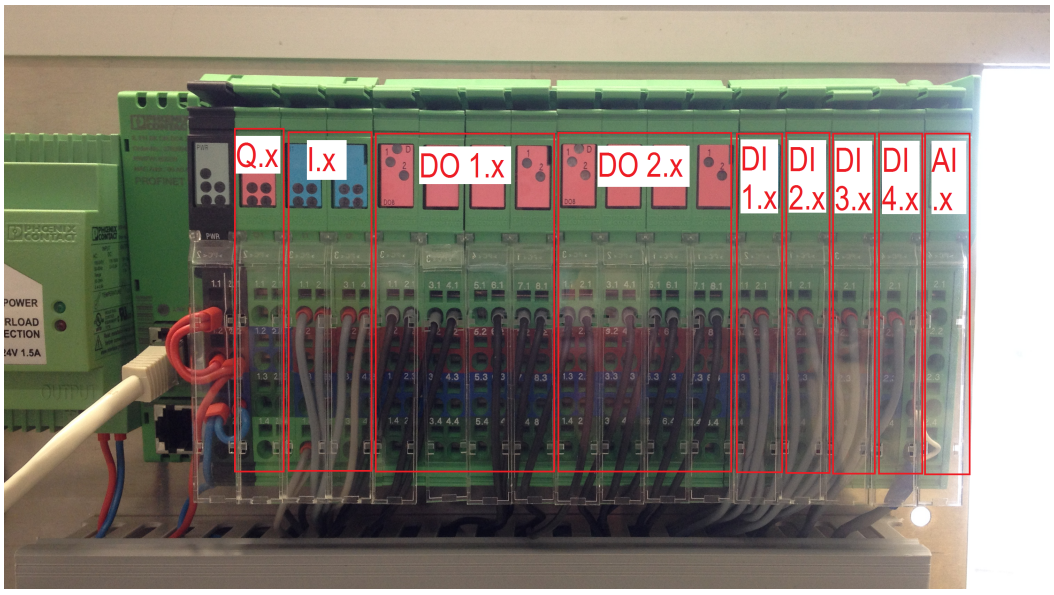

Abbildung A.2.: Anschlüsse am dezentralen Anbindungspunkt

Tabelle A.1.: Anschlusstabelle Station Verteilen

SPS Anschluss	Terminal Pin	Funktion
DI 3.1.1	10	Ausschiebezylinder in Anfangsstellung
DI 3.1.4	12	Schwenkarm bei Magazin
DI 3.2.1	11	Ausschiebezylinder in Endstellung
DI 3.2.4	13	Schwenkarm bei Prüfstation
DI 4.1.1	15	Teil angesaugt
DI 4.2.1	16	Magazin leer
DO 2.4.1	0	Teil ausschieben
DO 2.5.1	1	Schwenkarm zu Magazin
DO 2.6.1	2	Schwenkarm zu Prüfstation
DO 2.7.1	3	Ansaugen lösen
DO 2.8.1	4	Ansaugen

Tabelle A.2.: Anschlusstabelle Station Prüfen

SPS Anschluss	Terminal Pin	Funktion
DI 1.1.1	10	Teil ist metallisch
DI 1.1.4	12	Teil ist nicht schwarz
DI 1.2.1	11	Teil vorhanden
DI 1.2.4	13	Hebebühne untere Position
DI 2.1.1	14	Hebebühne obere Position
DI 2.1.4	16	Prüfstempel ausgefahren
DI 2.2.1	15	Auswurfzylinder eingefahren
AI 1.1	17	Messwert
DO 1.6.1	0	Hebebühne runterfahren
DO 1.7.1	1	Hebebühne hochfahren
DO 1.8.1	2	Auswurfzylinder ausfahren
DO 2.1.1	3	Prüfstempel ausfahren
DO 2.2.1	4	Rutschsperre ausfahren

Tabelle A.3.: Anschlusstabelle Station Sortieren

SPS Anschluss	Terminal Pin	Funktion
I 1.1	10	1. Weiche eingefahren
I 1.4	12	2. Weiche eingefahren
I 2.1	11	1. Weiche ausgefahren
I 2.4	13	2. Weiche ausgefahren
I 3.1	14	Kein Teil auf Förderband
I 4.1	15	Teil einsortiert
DO 1.1.1	0	1. Weiche aktivieren
DO 1.2.1	1	2. Weiche aktivieren
DO 1.3.1	2	Förderband ein
DO 1.4.1	3	Sperre ausfahren

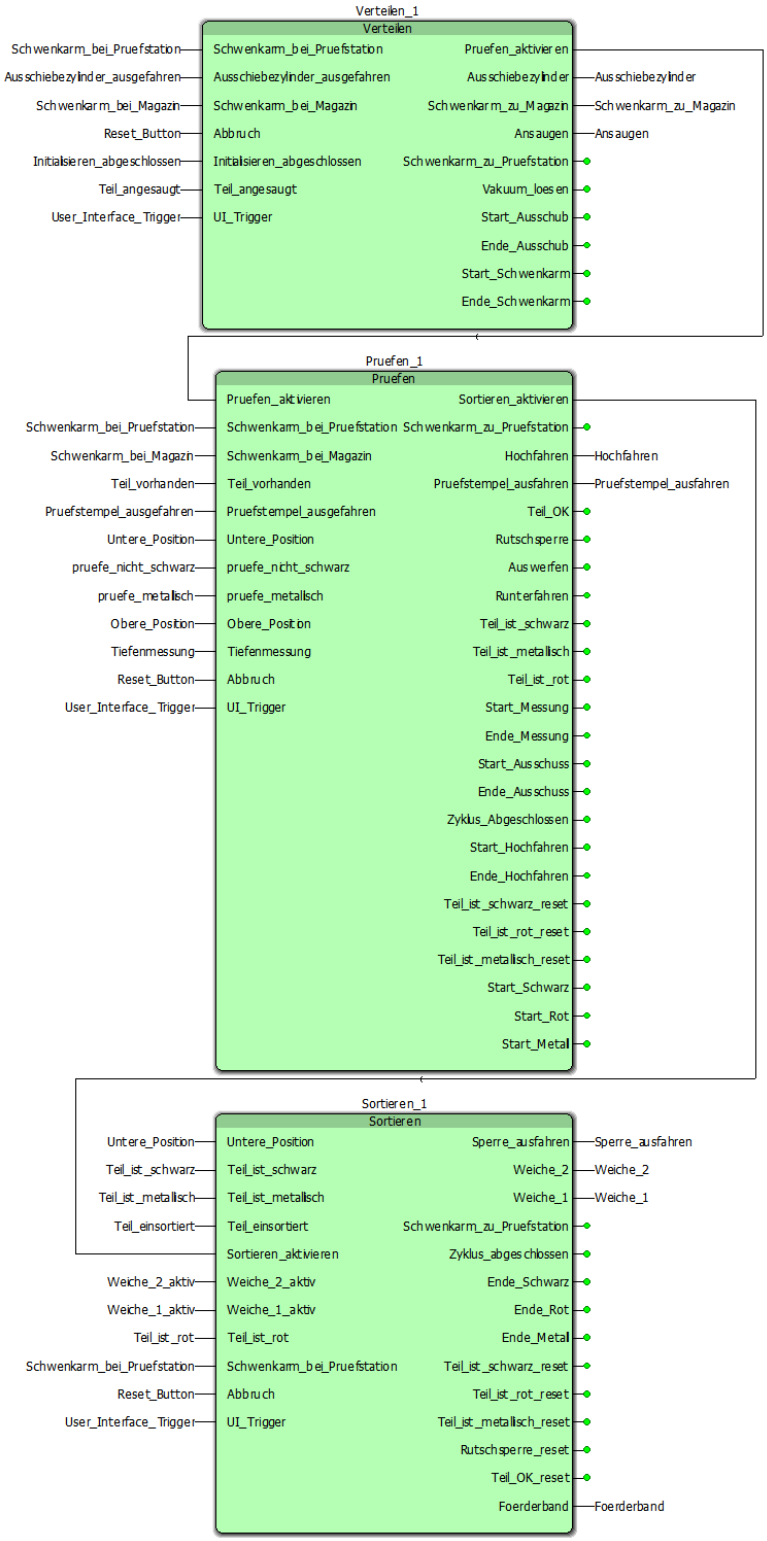

Abbildung A.3.: Programm Main, Kernprozess

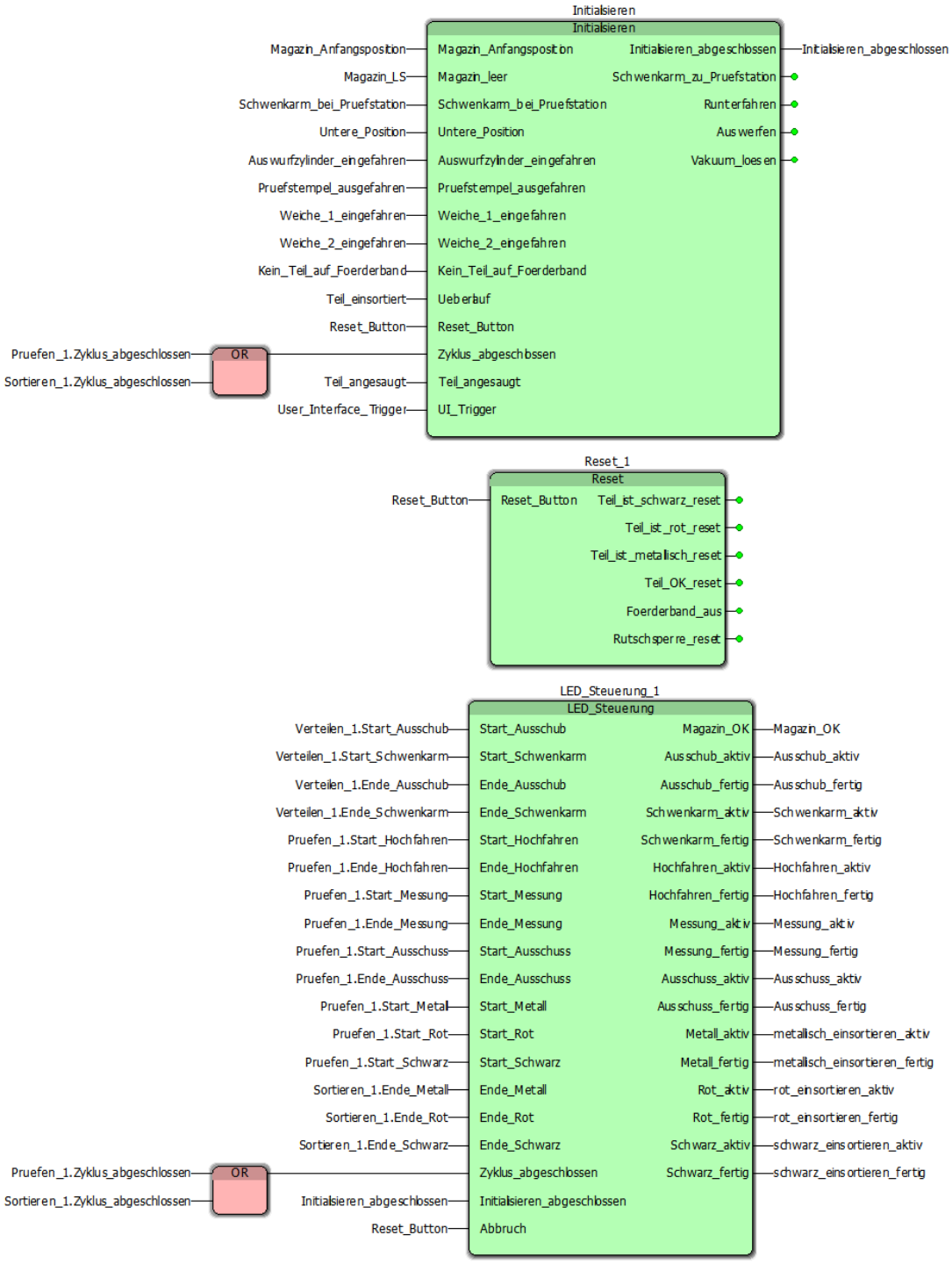

Abbildung A.4.: Programm Main, Nebenfunktionsbausteine

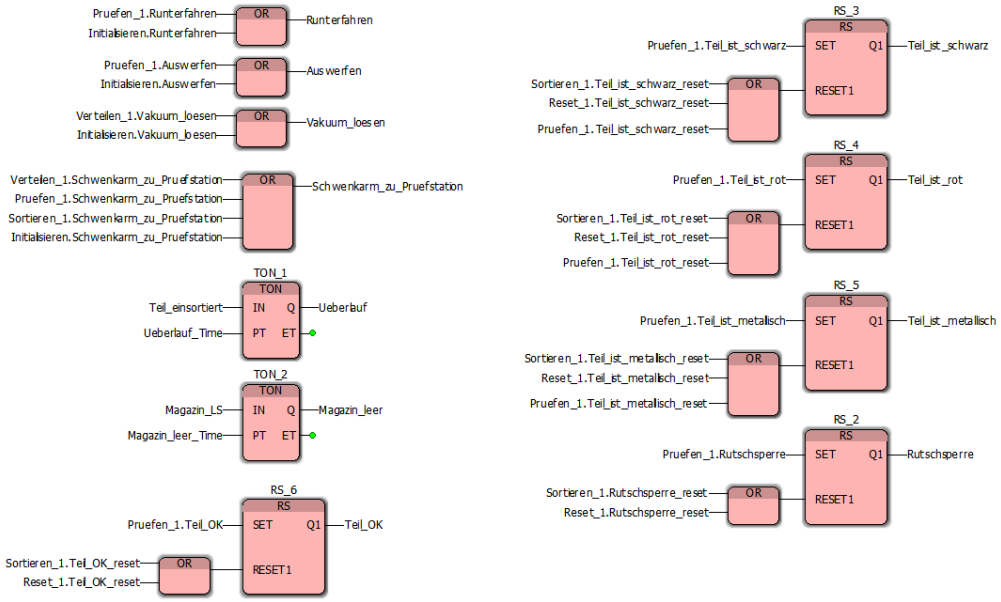

Abbildung A.5.: Programm Main, Nebenfunktionen

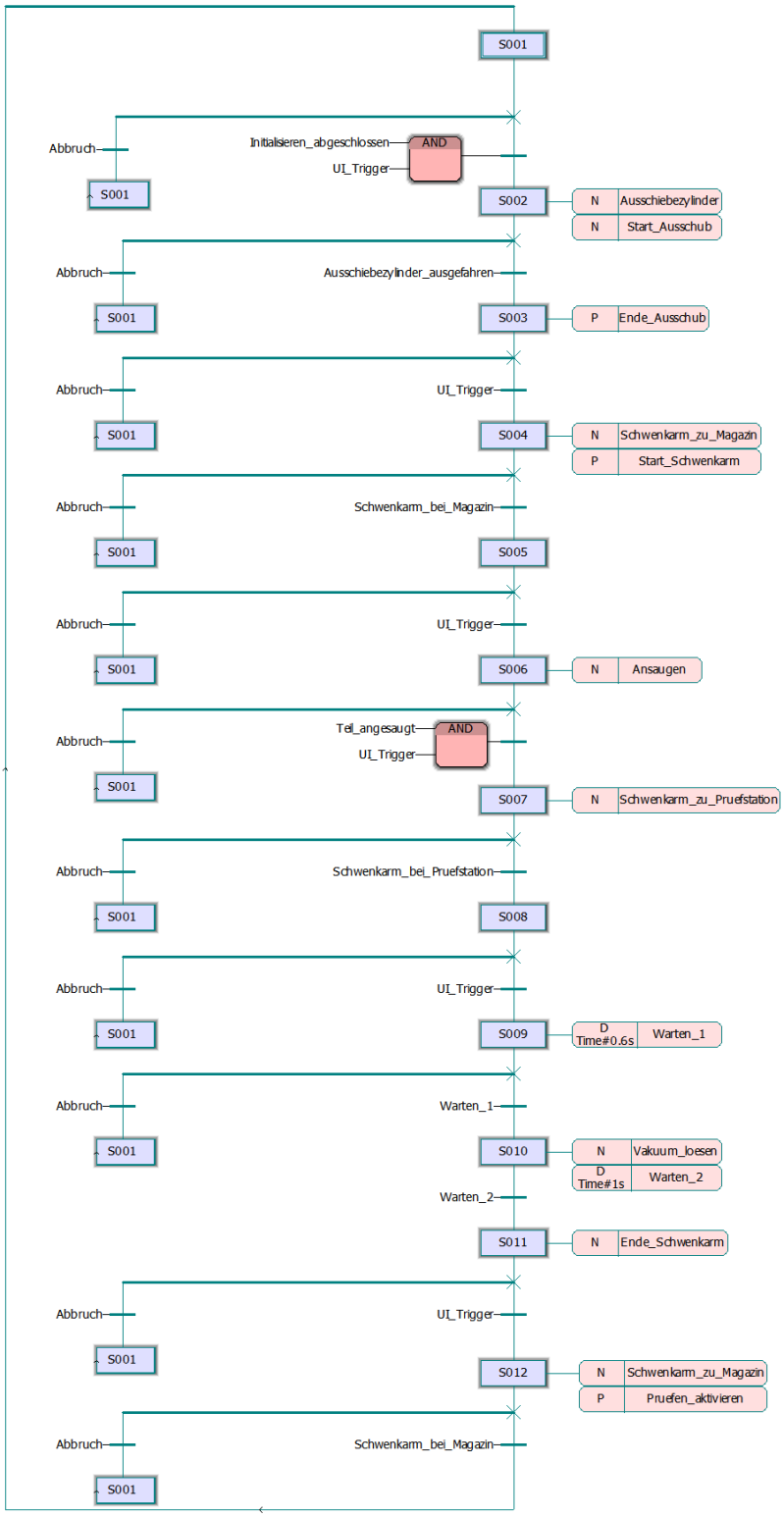

Abbildung A.6.: Ablaufplan Verteilen

Name	Typ	Verwendung
Default		
Abbruch	BOOL	VAR_INPUT
Ansaugen	BOOL	VAR_OUTPUT
Ausschiebezylinder	BOOL	VAR_OUTPUT
Ausschiebezylinder_ausgefahren	BOOL	VAR_INPUT
Ende_Ausschub	BOOL	VAR_OUTPUT
Ende_Schwenkarm	BOOL	VAR_OUTPUT
Initialisieren_abgeschlossen	BOOL	VAR_INPUT
Pruefen_aktivieren	BOOL	VAR_OUTPUT
Schwenkarm_bei_Magazin	BOOL	VAR_INPUT
Schwenkarm_bei_Pruefstation	BOOL	VAR_INPUT
Schwenkarm_zu_Magazin	BOOL	VAR_OUTPUT
Schwenkarm_zu_Pruefstation	BOOL	VAR_OUTPUT
Start_Ausschub	BOOL	VAR_OUTPUT
Start_Schwenkarm	BOOL	VAR_OUTPUT
Teil_angesaugt	BOOL	VAR_INPUT
UI_Trigger	BOOL	VAR_INPUT
Vakuum_loesen	BOOL	VAR_OUTPUT
Warten_1	BOOL	VAR
Warten_2	BOOL	VAR

Abbildung A.7.: Deklarationstabelle Verteilen

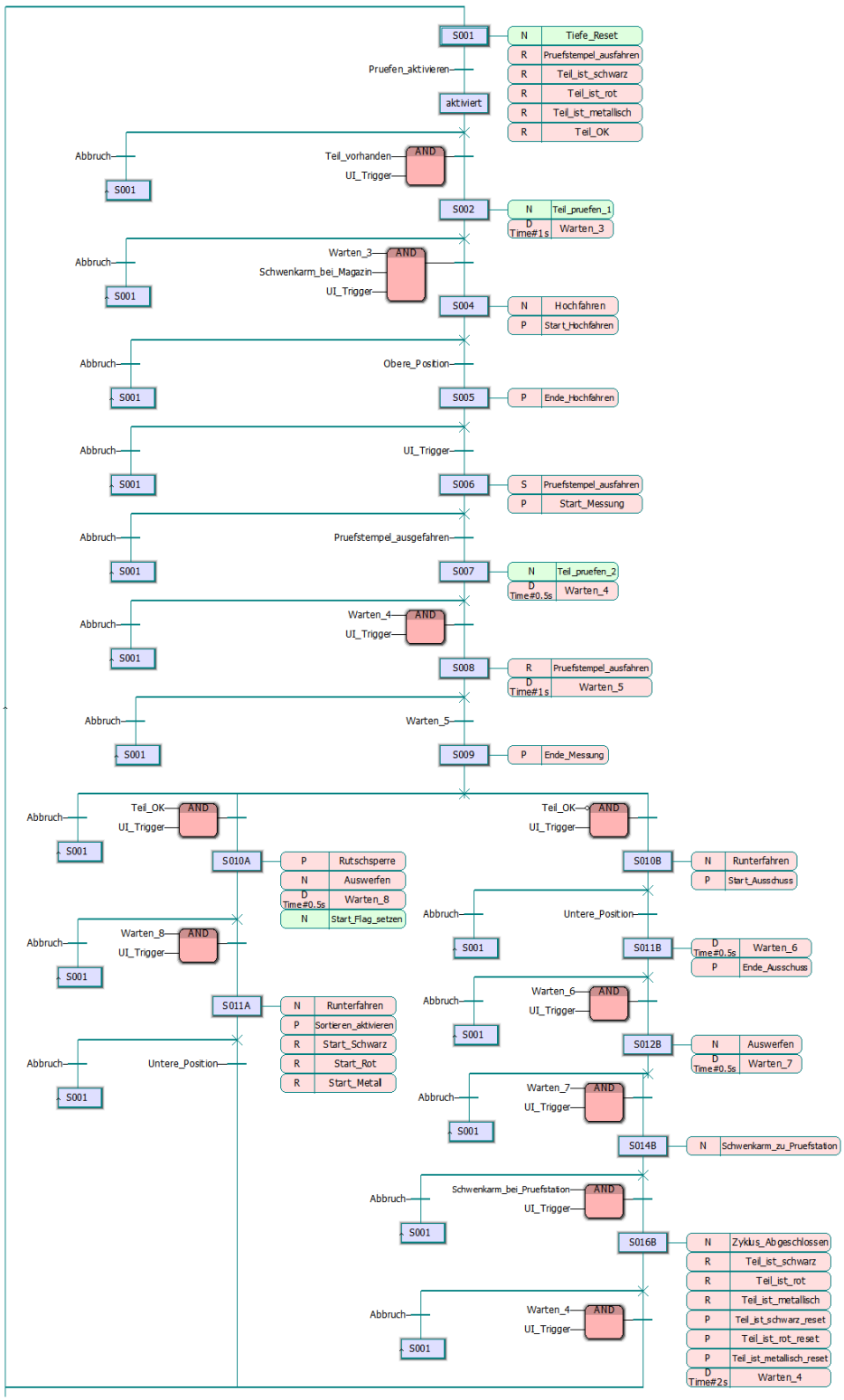

Abbildung A.8.: Ablaufplan Prüfen

Name	Typ	Verwendung
Abbruch	BOOL	VAR_INPUT
Auswerfen	BOOL	VAR_OUTPUT
Ende_Ausschuss	BOOL	VAR_OUTPUT
Ende_Hochfahren	BOOL	VAR_OUTPUT
Ende_Messung	BOOL	VAR_OUTPUT
Hochfahren	BOOL	VAR_OUTPUT
Obere_Position	BOOL	VAR_INPUT
pruefe_metallisch	BOOL	VAR_INPUT
pruefe_nicht_schwarz	BOOL	VAR_INPUT
Pruefen_aktivieren	BOOL	VAR_INPUT
Pruefstempel_ausfahren	BOOL	VAR_OUTPUT
Pruefstempel_ausgefahren	BOOL	VAR_INPUT
Runterfahren	BOOL	VAR_OUTPUT
Rutschsperre	BOOL	VAR_OUTPUT
Schwenkarm_bei_Magazin	BOOL	VAR_INPUT
Schwenkarm_bei_Pruefstation	BOOL	VAR_INPUT
Schwenkarm_zu_Pruefstation	BOOL	VAR_OUTPUT
Sortieren_aktivieren	BOOL	VAR_OUTPUT
Start_Ausschuss	BOOL	VAR_OUTPUT
Start_Hochfahren	BOOL	VAR_OUTPUT
Start_Messung	BOOL	VAR_OUTPUT
Start_Metall	BOOL	VAR_OUTPUT
Start_Rot	BOOL	VAR_OUTPUT
Start_Schwarz	BOOL	VAR_OUTPUT
Teil_ist_metallisch	BOOL	VAR_OUTPUT
Teil_ist_metallisch_reset	BOOL	VAR_OUTPUT
Teil_ist_rot	BOOL	VAR_OUTPUT
Teil_ist_rot_reset	BOOL	VAR_OUTPUT
Teil_ist_schwarz	BOOL	VAR_OUTPUT
Teil_ist_schwarz_reset	BOOL	VAR_OUTPUT
Teil_OK	BOOL	VAR_OUTPUT
Teil_vorhanden	BOOL	VAR_INPUT
Tiefe	REAL	VAR
Tiefenmessung	WORD	VAR_INPUT
UI_Trigger	BOOL	VAR_INPUT
Untere_Position	BOOL	VAR_INPUT
Warten_3	BOOL	VAR
Warten_4	BOOL	VAR
Warten_5	BOOL	VAR
Warten_6	BOOL	VAR
Warten_7	BOOL	VAR
Warten_8	BOOL	VAR
Zyklus_Abgeschlossen	BOOL	VAR_OUTPUT
Zyklus_Abgeschlossen_IN	BOOL	VAR_INPUT

Abbildung A.9.: Deklarationstabelle Prüfen

```
1    Tiefe := 0.0;
```

Abbildung A.10.: ST *Tiefe_Reset*: Reset der Absatzhöhe

```
1    Tiefe := WORD_TO_REAL(Tiefenmessung) * 0.00033333 * 3.0 + 9.0;
2    (* Tiefe [mm] = Messwert [1] * 333.33 [µV/1] * 3 [mm/V] *)
3
4    IF Teil_ist_schwarz = TRUE AND Tiefe >= 22.4 AND Tiefe <= 23.0 THEN
5        Teil_OK := TRUE;
6    ELSIF (Teil_ist_rot = TRUE OR Teil_ist_metallisch = TRUE) AND Tiefe >= 24.4 AND Tiefe <= 25.0 THEN
7        TEIL_OK := TRUE;
8    ELSE
9        Teil_OK := FALSE;
10   END_IF;
```

Abbildung A.11.: ST *Teil_prüfen_2*: IST/SOLL Vergleich der Absatzhöhe

```
1    IF  Teil_ist_schwarz = TRUE THEN
2        Start_Schwarz := TRUE;
3    ELSIF Teil_ist_rot = TRUE THEN
4        Start_Rot := TRUE;
5    ELSIF Teil_ist_metallisch = TRUE THEN
6        Start_Metall := TRUE;
7    END IF;
```

Abbildung A.12.: ST *Start_Flags_setzen*: Das jeweilige Start Flag setzen

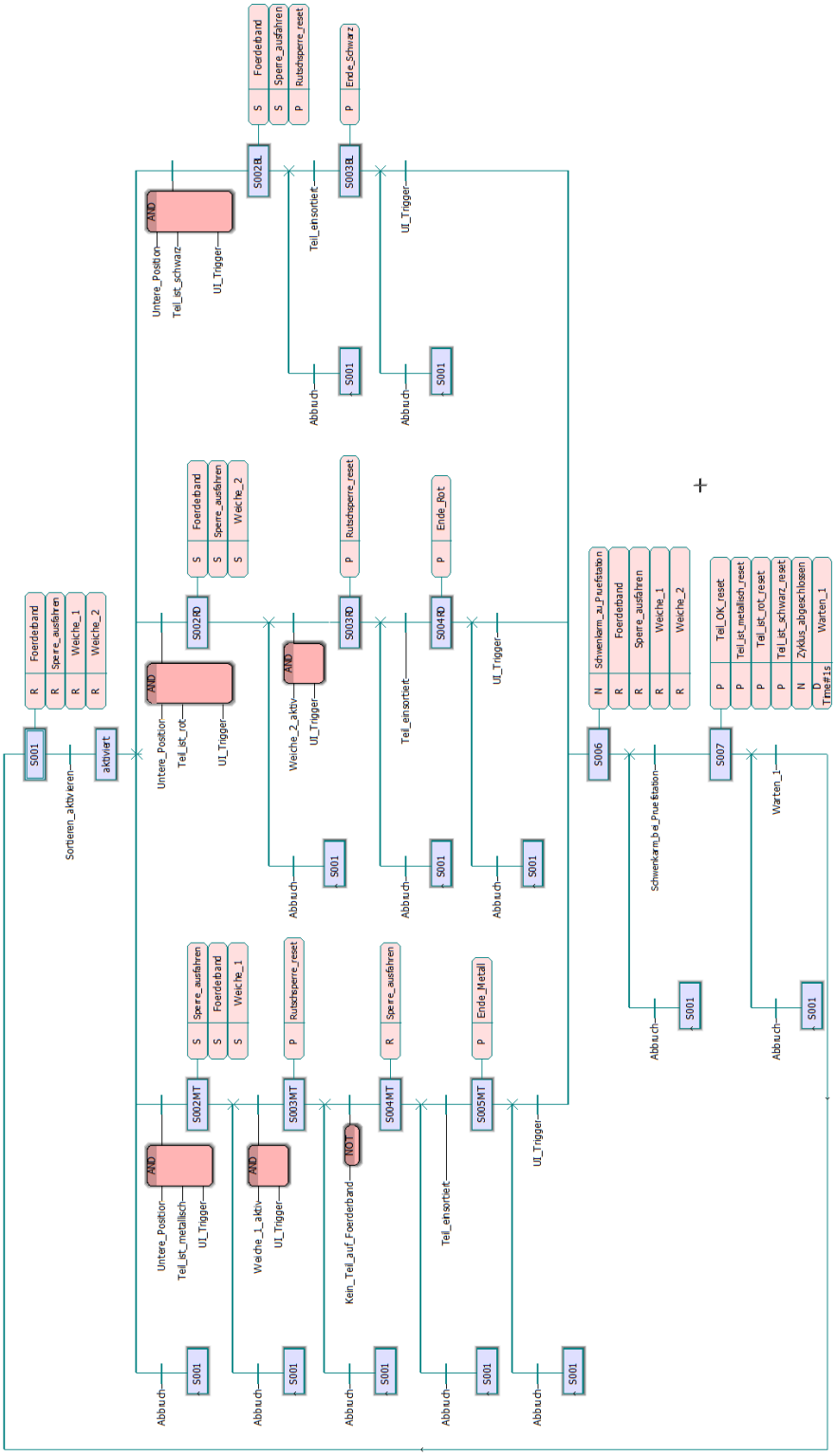

Abbildung A.13.: Ablaufplan Sortieren

Name	Typ	Verwendung
Abbruch	BOOL	VAR_INPUT
Ende_Metall	BOOL	VAR_OUTPUT
Ende_Rot	BOOL	VAR_OUTPUT
Ende_Schwarz	BOOL	VAR_OUTPUT
Foerderband	BOOL	VAR_OUTPUT
Kein_Teil_auf_Foerderband	BOOL	VAR_EXTERNAL
Rutschsperre_reset	BOOL	VAR_OUTPUT
Schwenkarm_bei_Pruefstation	BOOL	VAR_INPUT
Schwenkarm_zu_Pruefstation	BOOL	VAR_OUTPUT
Sortieren_aktivieren	BOOL	VAR_INPUT
Sperre_ausfahren	BOOL	VAR_OUTPUT
Teil_einsortiert	BOOL	VAR_INPUT
Teil_ist_metallisch	BOOL	VAR_INPUT
Teil_ist_metallisch_reset	BOOL	VAR_OUTPUT
Teil_ist_rot	BOOL	VAR_INPUT
Teil_ist_rot_reset	BOOL	VAR_OUTPUT
Teil_ist_schwarz	BOOL	VAR_INPUT
Teil_ist_schwarz_reset	BOOL	VAR_OUTPUT
Teil_OK_reset	BOOL	VAR_OUTPUT
UI_Trigger	BOOL	VAR_INPUT
Untere_Position	BOOL	VAR_INPUT
Warten_1	BOOL	VAR
Weiche_1	BOOL	VAR_OUTPUT
Weiche_1_aktiv	BOOL	VAR_INPUT
Weiche_2	BOOL	VAR_OUTPUT
Weiche_2_aktiv	BOOL	VAR_INPUT
Zyklus_abgeschlossen	BOOL	VAR_OUTPUT

Abbildung A.14.: Deklarationstabelle Sortieren

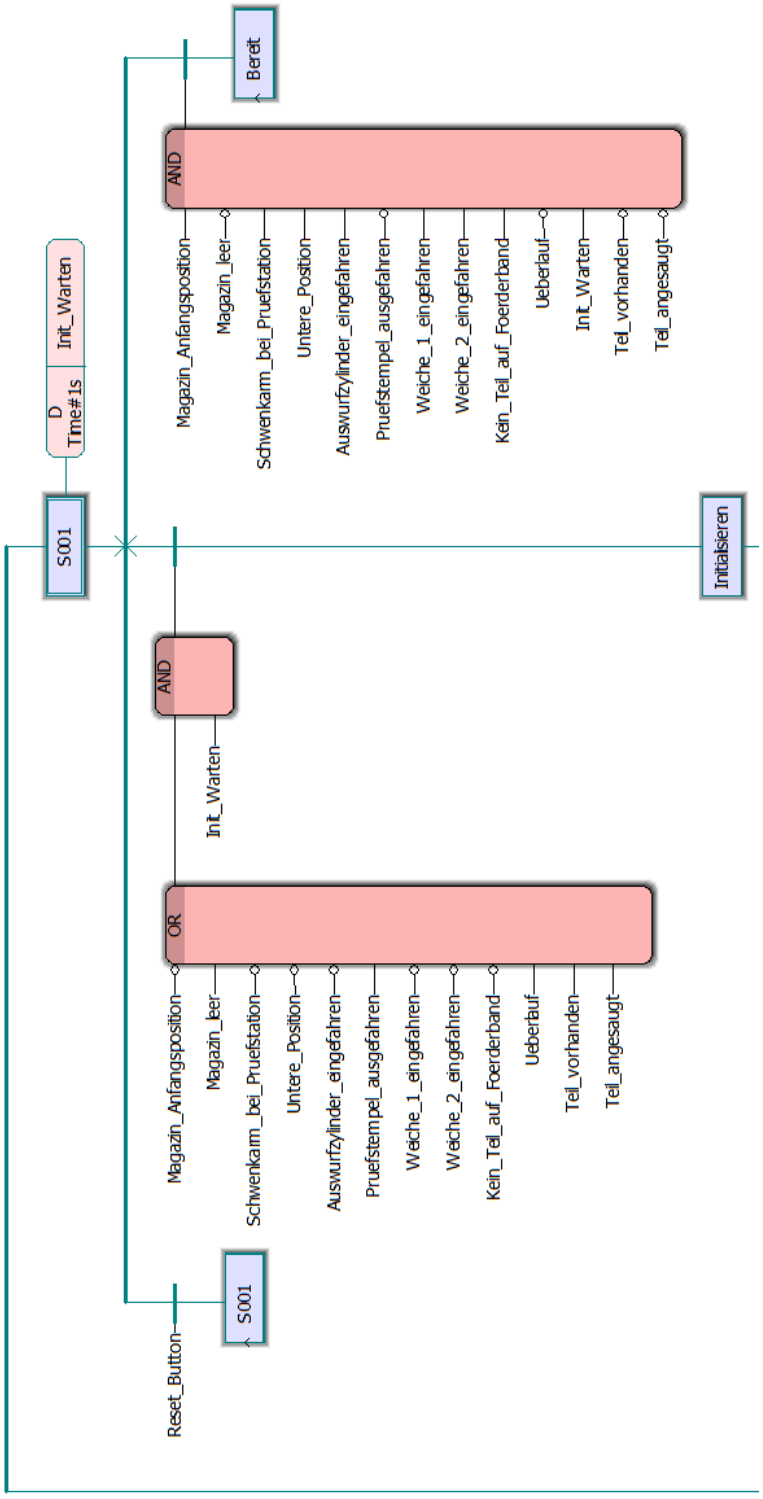

Abbildung A.15.: Ablaufplan Initialisieren, Teil 1

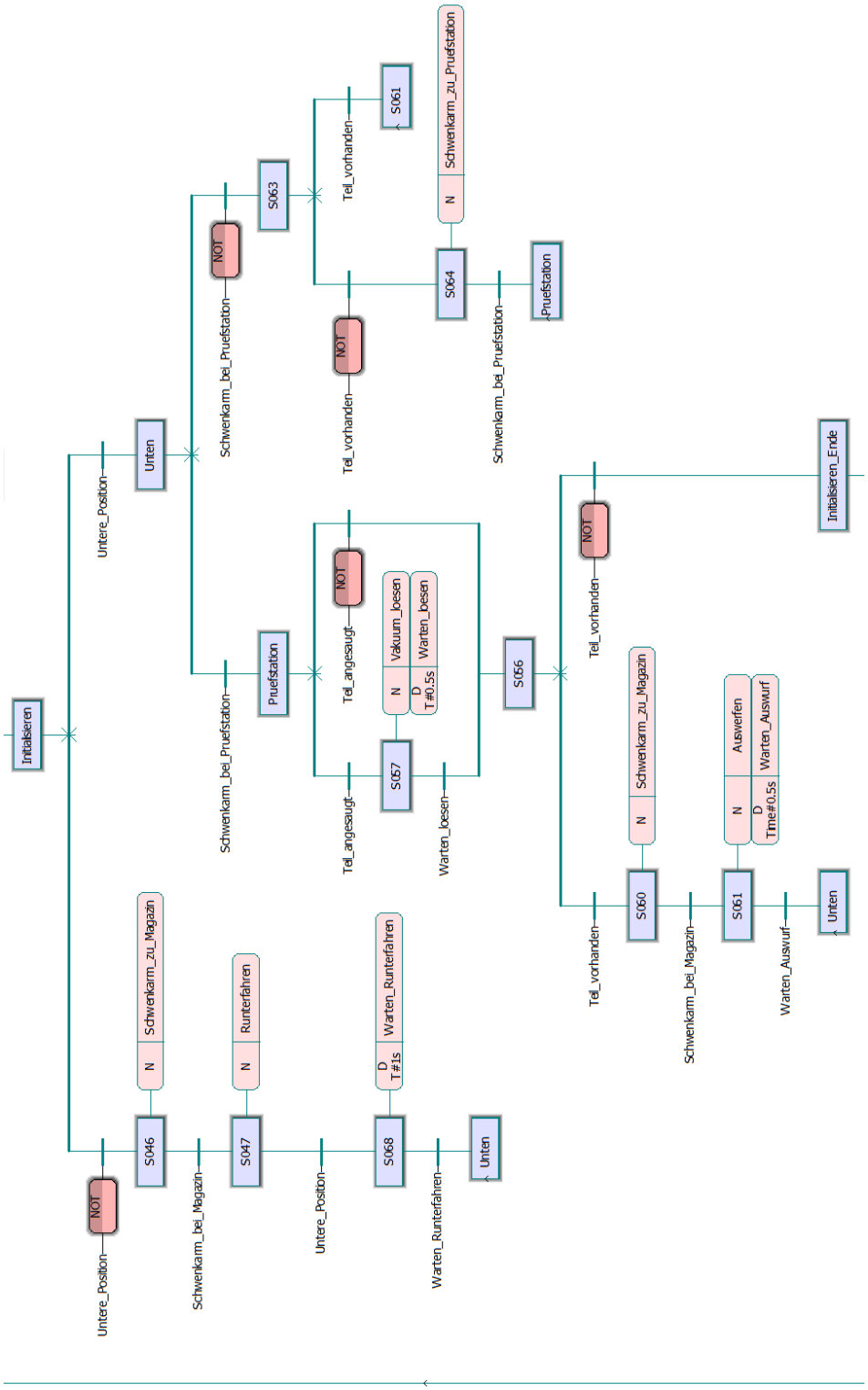

Abbildung A.16.: Ablaufplan Initialisieren, Teil 2

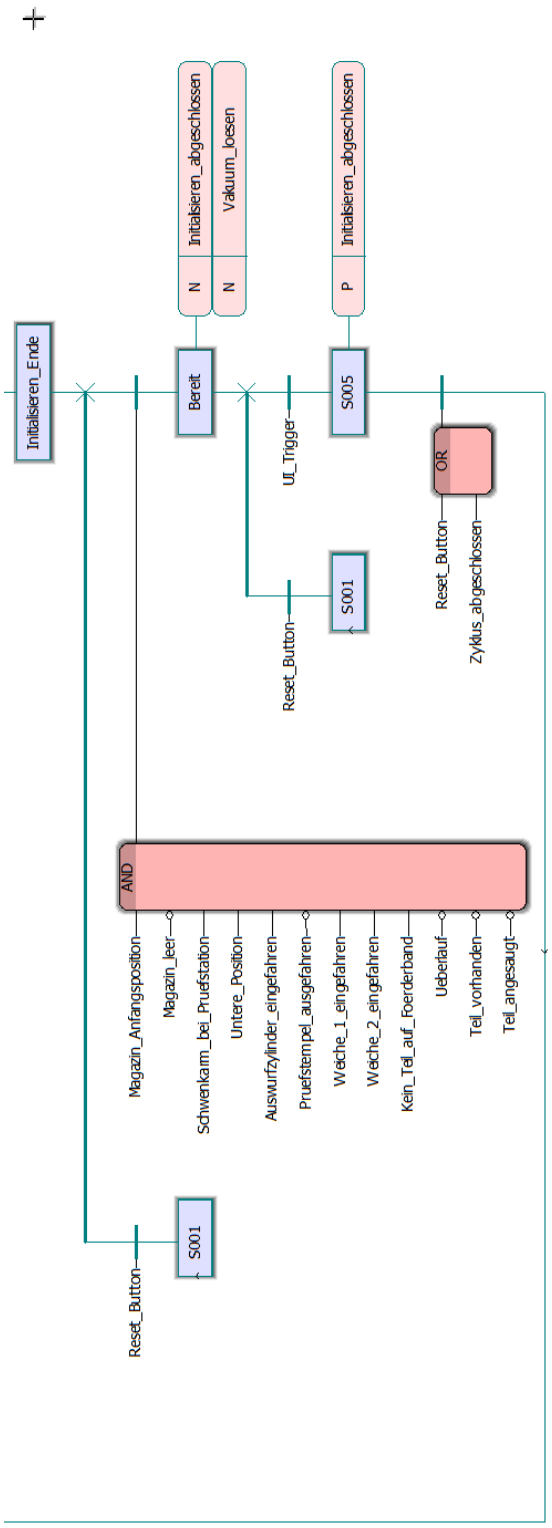

Abbildung A.17.: Ablaufplan Initialisieren, Teil 3

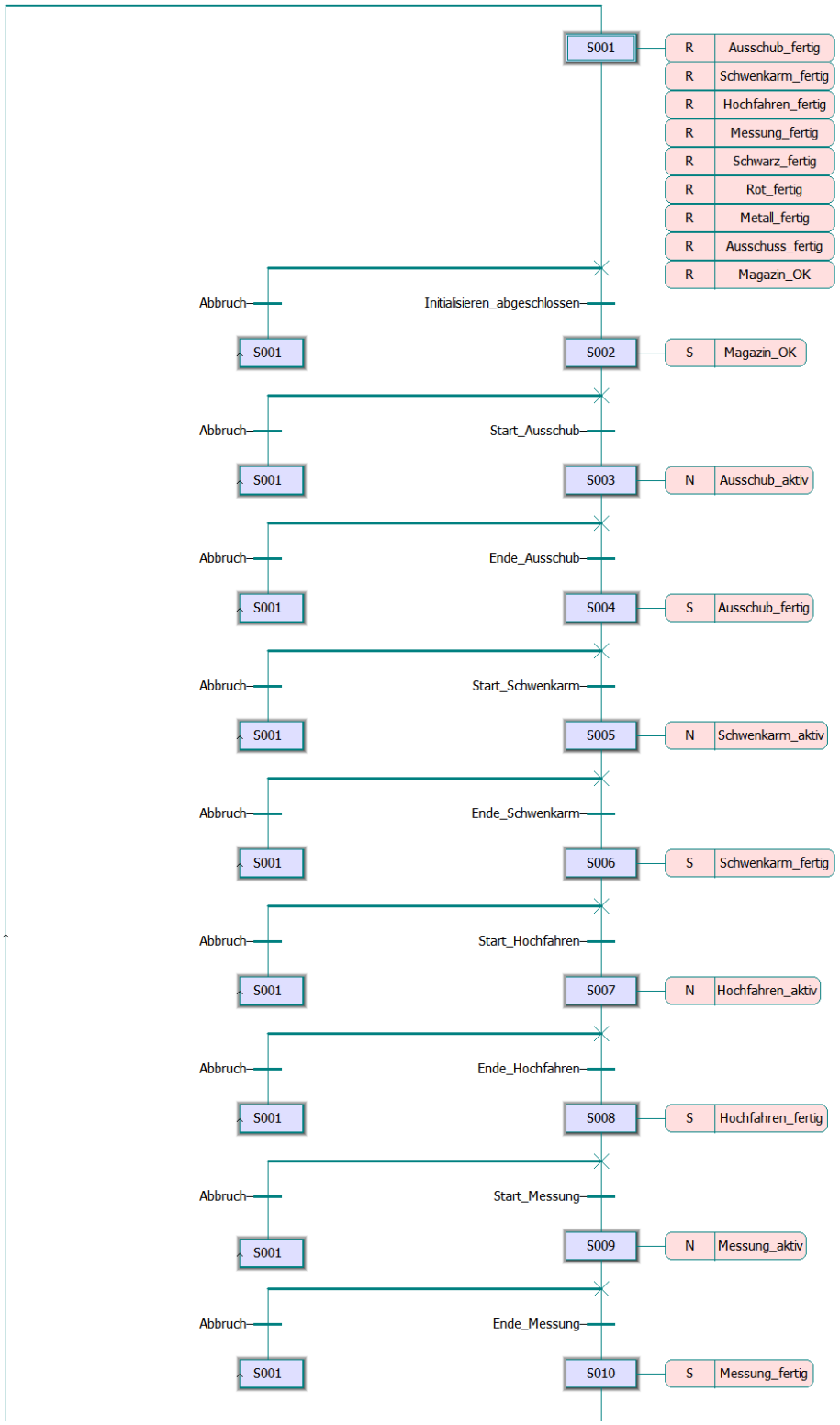

Abbildung A.18.: Ablaufplan LED_Steuerung, Teil 1

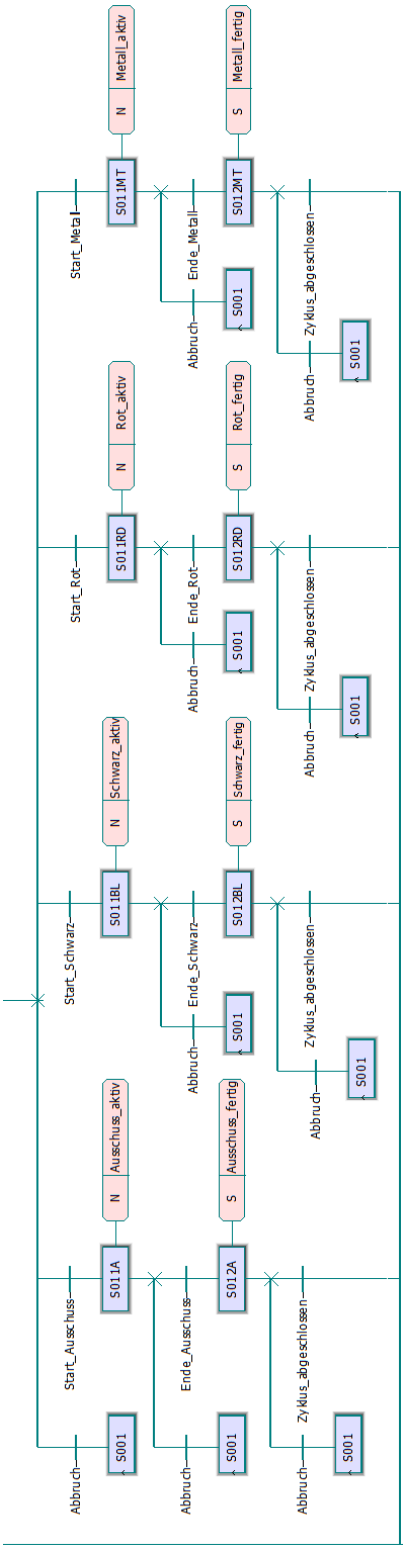

Abbildung A.19.: Ablaufplan LED_Steuerung, Teil 2

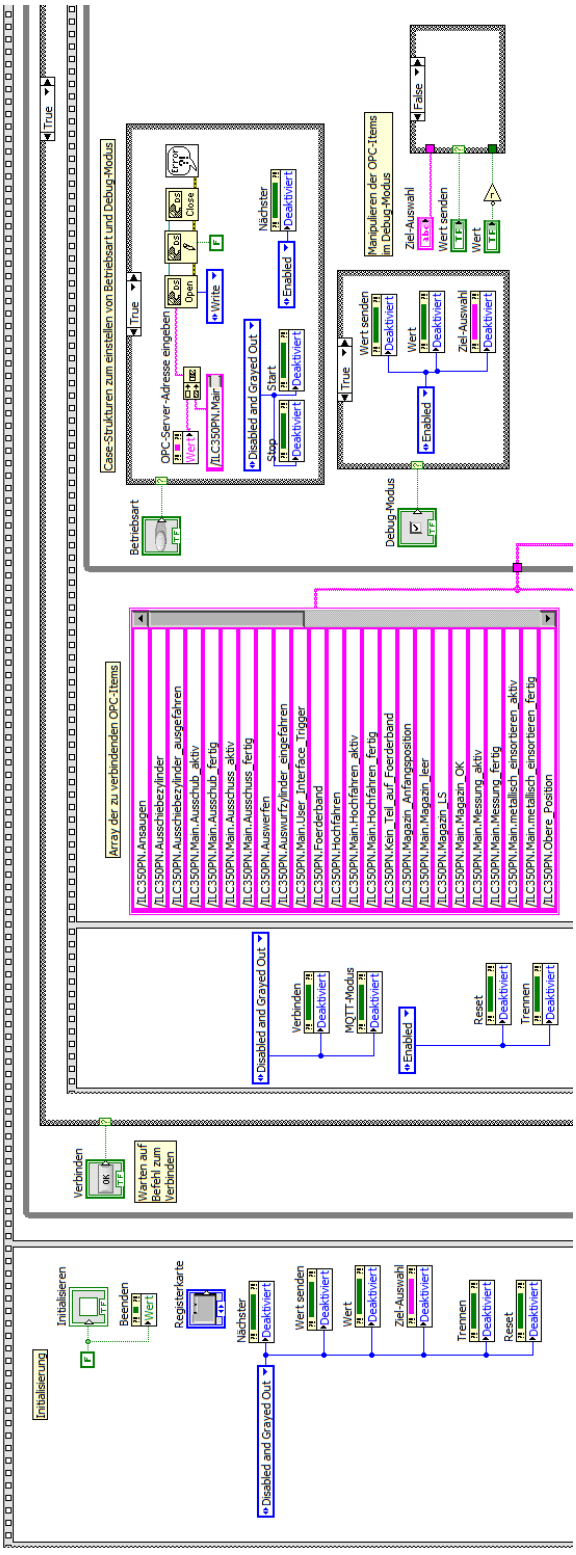

Abbildung A.20.: MPS-Panel Blockdiagramm, Teil 1

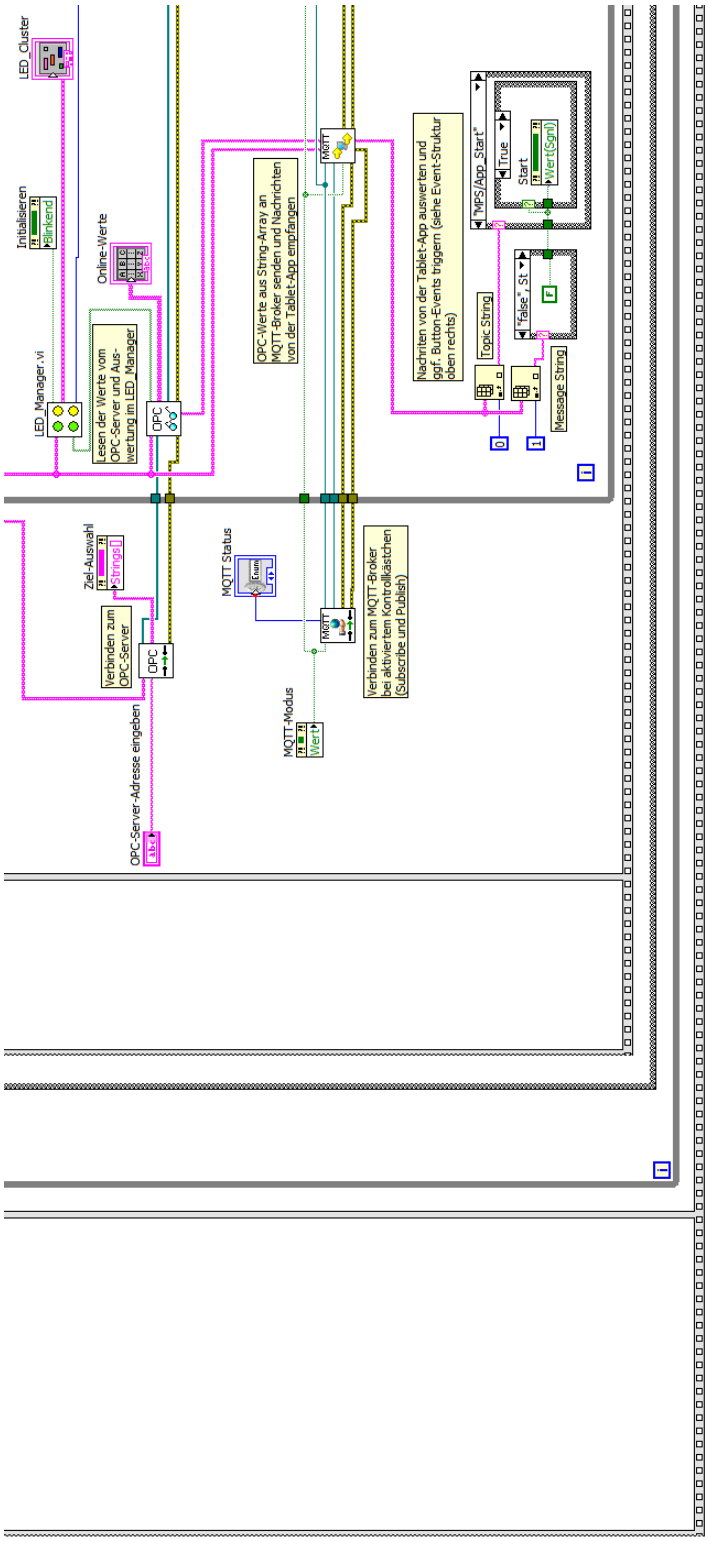

Abbildung A.21.: MPS-Panel Blockdiagramm, Teil 2

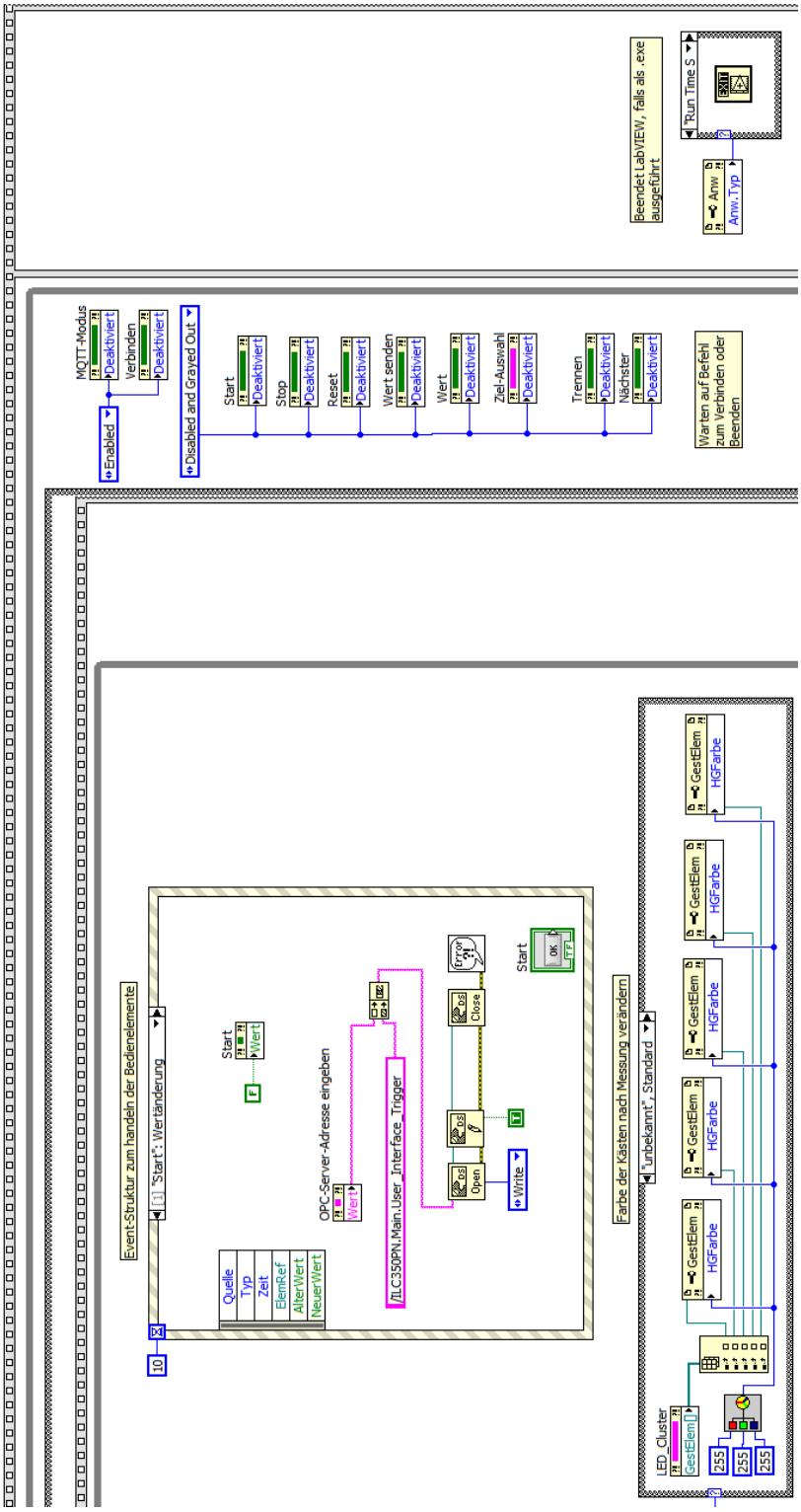

Abbildung A.22.: MPS-Panel Blockdiagramm, Teil 3

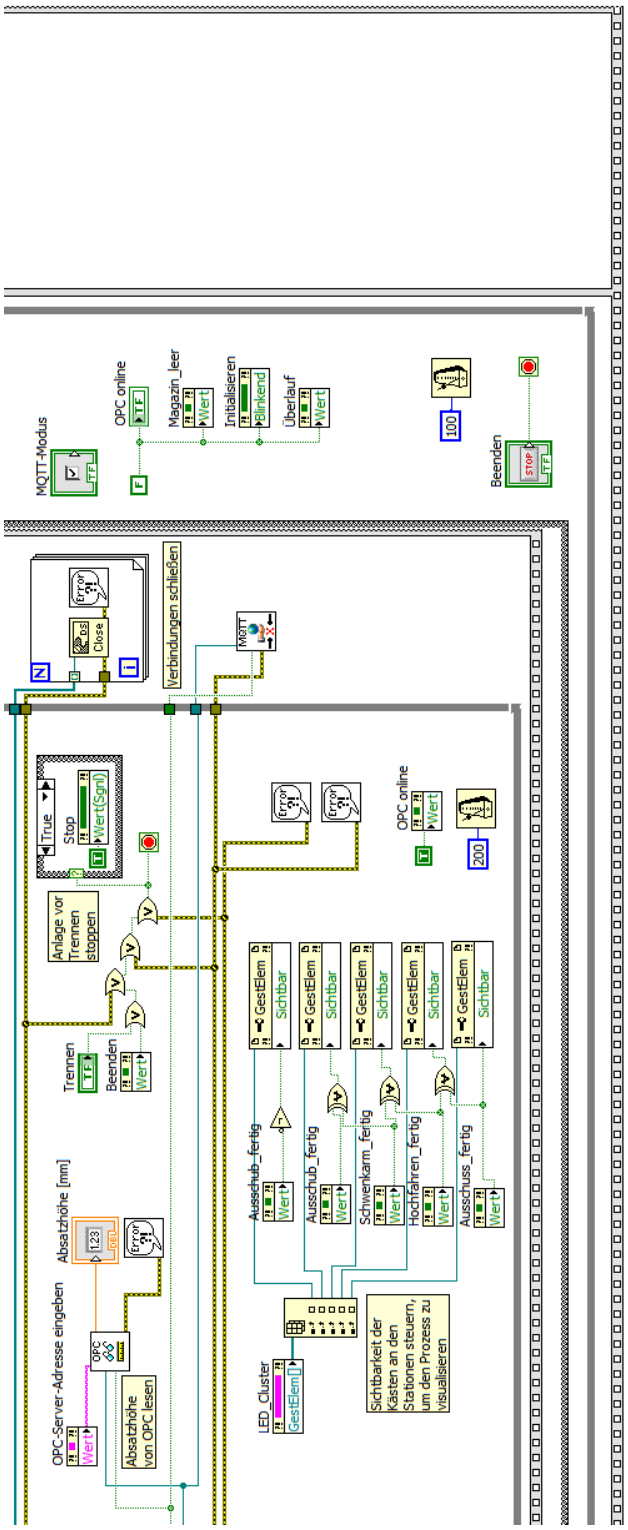

Abbildung A.23.: MPS-Panel Blockdiagramm, Teil 4

Literaturverzeichnis

[Georgi und Metin 2012] Georgi, Wolfgang ; Metin, Ergun: *Einführung in LabVIEW*. 5. Auflage. München : Carl Hanser Verlag, 2012

[Glanz und Jung 2010] Glanz, Axel ; Jung, Oliver: *Machine-to-Machine-Kommunikation*. Frankfurt am Main : Campus Verlag, 2010

[Lantermann 2013] Lantermann, Thomas: *Die Zukunft der SPS*. 2013. – URL `http://www.computer-automation.de/steuerungsebene/steuern-regeln/artikel/97620`. – Zugriffsdatum: 21.02.2016

[Plattform Industrie 4.0 2015] Plattform Industrie 4.0: *Umsetzungsstrategie Industrie 4.0*. 2015. – URL `https://www.bmwi.de/BMWi/Redaktion/PDF/I/industrie-40-verbaendeplattform-bericht`. – Zugriffsdatum: 17.02.2016

[Seitz 2012] Seitz, Matthias: *Speicherprogrammierbare Steuerungen für die Fabrik- und Prozessautomation*. 3. Auflage. München : Carl Hanser Verlag, 2012

[Wellenreuther und Zastrow 2011] Wellenreuther, Günter ; Zastrow, Dieter: *Automatisieren mit SPS - Theorie und Praxis*. 5., korrigierte und erweiterte Auflage. Wiesbaden : Vieweg+Teubner Verlag, 2011

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 26. Februar 2016 David Schmeding