2016

Dokumentation des betrieblichen Auftrages zum Thema

„Erweiterung einer Industrie 4.0 Anlage um einen QR-Code-Scanner über eine SPS mit Visualisierung“

Im Rahmen der Abschlussprüfung Teil 2
in der Ausbildung zum
Elektrotechniker für Automatisierungstechnik

Durchführung/Autor: Herr Simon Häußler

Zeitraum: 17.04.2016 – 31.05.2016

Betreuer: Herr Stefan Manemann

Volkswagen Group Academy

Berufsbildende Schulen 2 Wolfsburg

Inhaltsverzeichnis

1 Einleitung

Die Wandlung der Industrie ist in vollem Gange. Der Schritt von Industrie 3.0 zur Industrie 4.0 gelangt im Wesentlichen über die Vernetzung jeglicher industrieller Komponenten untereinander und mit dem Menschen. Ein weiterer Punkt ist die dezentrale Überwachung und Steuerung einer Anlage über das Internet. Auch das Thema *Big Data*, bei dem es unter anderem darum geht, sämtliche Prozessdaten in einer Datenbank persistent abzuspeichern, spielt bei der vierten industriellen Revolution eine immer größer werdende Rolle, wodurch Mensch, Maschine und letzten Endes auch das Produkt über das weltweit verbreitete Internet vernetzt sind.

Um diese Thematik den Schülern der Berufsbildenden Schulen 2 (BBS2) in Wolfsburg praktisch näher zu bringen, wurde im Auftrag von Herrn Manemann eine M&M-Abfüllanlage im Sinne der vierten industriellen Revolution entwickelt. Dieses Projekt wurde vom New Automation e.V. finanziell unterstützt und kürzlich zum *Leuchtturmprojekt 2015* gekürt. Das bedeutet, dass dieses Projekt die aktuellen Techniktrends der Industrie aufzeigt und gleichzeitig ein Vorreiter für andere Schulen ist, die diese Anlage nachbauen können.

Seit Ende der 60er Jahre werden eindimensionale Strichcodes in Handelsunternehmen zur elektronischen Identifizierung verwendet (Wikipedia, 2016). Auch heute werde unterschiedliche 1D- und 2D-Codes zur eindeutigen Identifizierung von z.B. Komponenten der Automobilproduktion verwendet. In die Kategorie der 2D-Codes fallen auch die QR-Codes (**Q**uick-**R**esponse-**C**odes), mit denen Informationen so codiert werden, dass sie maschinell besonders schnell gefunden und gelesen werden können. Dies ist auch in der Industrie eine aktuelle, optimal angepasste und häufig angewendete Technik. Aus diesem Grund wird sie auch in der M&M-Abfüllanlage zur Identifizierung der einzeln produzierten Dosen verwendet.

2 Projektbeschreibung

2.1 Auftragsbeschreibung

Die Industrie 4.0 Anlage (oder auch M&M-Abfüllanlage) wurde in der BBS2, Kleiststraße 44 in 38440 Wolfsburg entwickelt und konnte im Raum C213 von vier dualen Studenten (mich eingeschlossen) im Rahmen der technischen Abschlussprüfung erweitert werden.

Der Ausgangszustand
Die Anlage befüllt einzeln kleine Dosen mit M&Ms, setzt mittels einer pneumatischen Pick&Place-Einheit einen Deckel auf die Dosen und bedruckt diesen nach Bedarf mit einem QR-Code und einem individuell einzugebenden Namen. Dabei werden sämtliche erfasste Produktionsdaten zu jeder einzelnen Dose, wie zum Beispiel (z.B.) Umgebungstemperatur, Datum und Uhrzeit in einer zentralen Datenbank auf einem Server der Firma Strato in Berlin gespeichert. Diese können über eine URL, die hinter dem QR-Code steckt, zur übersichtlichen Darstellung abgerufen werden. Die Anlage ist über die Datenbank und einer Android-App samt Smartwatch mit dem Anlagenführer verbunden und informiert diesen über den aktuellen Zustand der Anlage oder warnt ihn bei eventuellen Ausfällen der Anlage, wenn z.B. die

M&Ms zuneige gehen. Über die App-Oberfläche kann die Stückzahl-Eins-Produktion einer Dose mit individueller Namensbeschriftung gestartet werden. Dadurch wird ein Eintrag in der Datenbank am Ende einer virtuellen Warteschlange getätigt. Diese Warteschlange wird kontinuierlich von der Anlage abgearbeitet, solange sie sich im Automatikbetrieb befindet. Die Schnittstelle zum Internet bietet hierbei ein Raspberry Pi, das via Ethernet mit dem Rest der Anlage, wie z.B. der steuernden speicherprogrammierten Steuerung (SPS) von Phoenix Contact oder dem Drucker von Bluhm Systeme vernetzt ist.

Eine Vorrichtung für ein Visualisierungsgerät oder ein 2D-Codeleser ist noch nicht vorhanden.

Das Ziel ist die Integration eines 2D-Codelesers in die beschriebene Industrie 4.0 Anlage, der das Werkstück beziehungsweise (bzw.) die Dose im laufenden Produktionsprozess über den aufgedruckten QR-Code identifiziert. Über eine SPS sollen die Daten zur Weiterverarbeitung an das Raspberry Pi versendet werden. Hier sollen die entsprechenden Daten des Werkstückes aus der Datenbank gelesen werden und zur Weiterverarbeitung wieder zurück zur SPS geschickt werden, um sie dann übersichtlich auf einem Visualisierungsgerät bzw. Touch Panel darzustellen. Des Weiteren soll die Dose im nächsten Produktionsschritt von der Seite entsprechend der Daten aus der Datenbank beschriftet werden.

Anschließend soll die Arbeit auf der Lernplattform www.xplore.dna.net so dokumentiert werden, dass sie von anderen zum Nachbau nachvollzogen werden kann.

Die Rahmenbedingungen
Die zu verwendenden Komponenten wurden vom Auftraggeber Herrn Manemann bereitgestellt. Dazu gehören eine S7-1200 SPS von Siemens, der SR-1000 2D-Codeleser der Firma Keyence und das Visualisierungsgerät KTP600 Basic Color PN ebenfalls von Siemens. Das Lastenheft zur genauen Beschreibung des Auftrages wurde erstellt und kann dem Anhang 5.4 entnommen werden.

2.2 Informationsphase

Am Anfang meines Auftrages erhielt ich eine Unterrichtung über die geltende Laborordnung (siehe Anhang 5.1) und die Betriebsanweisung (siehe Anhang 5.2) von meinem Auftraggeber Herrn Manemann. Zudem habe ich ein zertifiziertes Selbstlernprogramm zum Thema „Die 5 Sicherheitsregeln“ der BG ETEM durchgeführt. Das Zertifikat dazu kann dem Anhang 5.3 entnommen werden.
Danach wurde mir der Funktionsumfang der bisher entwickelten M&M Abfüllanlage durch das vorherige Projektteam aus dualen Studenten für Elektrotechnik des Jahrgangs 2014 ausführlich erklärt. Dazu bekam ich das SPS-Programm, das Java-Programm des Raspberry Pi und den Schaltplan der Anlage.

Ich erhielt dann von meinem Auftraggeber das Lastenheft (siehe Anhang 5.4), welches meinen Auftrag für die Abschlussprüfung erläuterte. Ich besprach den Auftragsumfang mit Herrn Manemann und beschäftigte mich danach mit den Unterlagen zu der Anlage und las mir einzelne Passagen des Benutzerhandbuches des 2D-Codelesers (Keyence Corporation, 2016), des Touch Panels (Siemens, 2016) und zu PHP Skripten (Theis, 2016) hinsichtlich der Umsetzung des Auftrages durch und notierte mir Ansatzpunkte. Dazu informierte ich mich auch über einzuhaltende Vorschriften und Normen. Anschließend habe ich mich mit Herrn

Manemann noch einmal bezüglich meiner Art der Umsetzung des Auftrages abgestimmt und offene Fragen geklärt. Ich erhielt einen festen Arbeitsplatz im Raum C213 mit PC und einen Ansprechpartner für Metallarbeiten, Herrn Böspflug. Arbeitsmaterial und Werkzeug wurde mir zur Verfügung gestellt.

Auf die Absprachen hin beschäftigte ich mich näher mit der Funktionsweise des 2D-Codelesers SR-1000 mithilfe des Benutzerhandbuches und des Datenblattes, um ihn gemäß der Vorgaben (via Profinet, im laufenden Prozess, etc…) zu integrieren. Dieselbe Informationsbeschaffung führte ich auch für die S7-1200 SPS und das Visualisierungsgerät KTP600 Basic Color PN durch.

2.3 Auftragsplanung

Infolge der umfassenden Informationsbeschaffung fing ich an, den Auftrag zu planen. Ich plante den zeitlichen Verlauf des Auftrages inklusive wichtiger Meilensteine während des Projektverlaufes und grober Arbeitsschritte (siehe Anhang 6.1). Daraufhin erstellte ich je eine Liste für Materialien und Bauteile (siehe Anhang 6.2), Werkzeuge (siehe Anhang 6.3) und Prüfmittel (siehe Anhang 6.3) zusammen. Da ich das Werkzeug, die Prüfmittel und sämtliche benötigte Komponenten inklusive der Verbrauchsmaterialien, wie Aderendhülsen und Aderleitung bereits von Herrn Manemann bereitgestellt bekommen habe oder sie mir aus den Räumen der Berufsschule beschaffen konnte, musste nichts bestellt werden. Ich informierte mich trotzdem über den Bestellprozess in der BBS, da die Regel anders ist.
Als nächstes bereitete ich die benötigten Protokolle für die Auftragskontrolle vor, damit diese geplant und strukturiert verlaufen kann und die Revisionsunterlagen gesammelt an den Kunden übergeben werden können.

Anschließend befasste ich mich mit der Position der Komponenten. Die SPS sollte ich in den seitlichen Schaltschrank –U2 setzen. Der 2D-Codeleser musste zwangsläufig im Produktionsverlauf hinter den ersten Drucker, da dieser den QR-Code, der gelesen werden soll, druckt. Ich entschied mich dafür, den Leser über der Rutsche am Bandende zu befestigen, da ich dann die Lichtschranke –B1 der Rutsche zur Triggerung des Codelesers verwenden konnte. Zur Befestigung wählte ich ein Item-Profil aus. Um den Leser an dem Profil zu befestigen benötigte ich allerdings einen Metall-Winkel. Dafür erstellte ich eine exakte Maßzeichnung (siehe Anhang 6.4) für die Herstellung eines solchen Winkels durch Herrn Böspflug.

Danach nahm ich die Erweiterung des vorhandenen Schaltplanes mit der Software ePLAN Education Version 2.5 (siehe Anhang 6.5) vor und achtete schon hier auf die korrekte Bezeichnung der Betriebsmittel und Leitungen nach DIN EN 81346-2.

Als letztes erstellte ich ein Verteilungsdiagramm, in dem ich mir die Kommunikation der Komponenten aus meinem Auftrag visualisierte und plante und ein Sequenzdiagramm, in dem der ungefähre Ablauf des Java-Programms erläutert wird. Die Diagramme befinden sich in den Anhängen 6.6 und 6.7.

2.4 Auftragsdurchführung

Um an meinem Arbeitsplatz ein sicheres Arbeiten zu gewährleisten und somit den Sicherheitsvorschriften zu genügen, hielt ich stets Ordnung und achtete auf die Einhaltung der 5 Sicherheitsregeln (DIN VDE 0105), der allgemeinen Unfallverhütungsvorschriften (UVV)

bzw. der Berufsgenossenschaftlichen Vorschriften (BGV) und der Laborordnung. Die Laborordnung kann dem Anhang 5.1 entnommen werden. Während meiner Arbeit beachtete ich die DIN VDE Normen z.B. bei der Verdrahtung und Kennzeichnung der Betriebsmittel.

Ich begann mit der Positionierung und Kennzeichnung der SPS auf der geerdeten Hutschiene im Schaltschrank –U2 nach DIN EN 60715. Dann verdrahtete ich die SPS mit den genormten Aderfarben (DIN VDE 0113-1) und -querschnitten (DIN VDE 0100-540). Danach verdrahtete ich das Touch Panel. Zum Schluss habe ich den 2D-Codeleser mit dem Item-Profil und dem von Herrn Böspflug gefertigten Metall-Winkel über der Rutsche am Bandende befestigt, über die mitgelieferte Steuerungsleitung an die Versorgungsspannung und mit der mitgelieferten Ethernet-Leitung an das Netzwerk angeschlossen. Danach habe ich den Leser mit einer Betriebsmittelkennzeichnung gemäß des Schaltplanes versehen und mittels der Software AutoID Network Navigator (SR-H5W) von Keyence den Leser ausgerichtet und konfiguriert. Die Konfiguration des Lesers beinhaltete unter anderem die automatische Einstellung von Belichtungszeit, Filter und Fokus für verschiedene Umgebungsbedingungen, um den Codeleser auf die gewünschte Anwendung abzustimmen. Den zu scannende Bereich und die Geschwindigkeit, mit der die Werkstücke am Leser vorbei rutschen habe ich festgelegt. Auch die Konfiguration der Profinet-Schnittstelle, sprich IP-Adressvergabe und Profinet-Namen habe ich eingestellt. Bei der Kommunikation habe ich mich für ein Verfahren ohne Handshake zwischen Codeleser und SPS entschieden, da die Anwendung eine sehr schnelle Datenübertragung benötigt und ein Handshake einige Latenzzeiten mit sich bringt. Des Weiteren habe ich die Triggerung eines Lesevorganges via Profinet aktiviert und die Signalausgabe über die Steuerungsleitung deaktiviert, da diese Signale nicht benötigt werden.

Als nächstes projektierte ich die drei Komponenten mit dem Software-Programm *Tia Portal V13* von Siemens und nahm die Hardwarekonfiguration der SPS vor. Zudem habe ich die Schnittstellen der SPS zu dem 2D-Codeleser und dem Touch Panel parametriert. Hierzu musste ich die .gsd-Datei des Codelesers, die mir von Keyence kostenlos zur Verfügung gestellt wurde, installieren.
Nachdem ich die Konfiguration des 2D-Codelesers und die Projektierung der Hardware erfolgreich abgeschlossen hatte, begann ich mit der Programmierung der SPS und des Touch Panels mit dem Tia Portal V13. Hierzu entwarf ich zunächst zusammen mit meinem Kollegen Peter Levermann, der für die Bedruckung und Drehung der Dosen am Ende der Rutsche zuständig war, eine Schrittkette, die sich im Anhang 7.1 befindet. Diese haben wir dann gemeinsam in ein lauffähiges SPS-Programm umgewandelt, welches in Anhang 7.2 zu finden ist. Nebenbei habe ich für das Touch Panel das Layout und die Ausgabefelder für die Visualisierung der Produktionsdaten erstellt und mit den passenden Variablen der SPS verknüpft. Das Layout der Visualisierung ist in Anhang 7.3 abgebildet.

Als nächstes habe ich das Java-Programm des Raspberry Pi erweitert, sodass es die Daten der SPS via TCP/IP empfangen, auswerten und darauf reagieren konnte. Dabei war besonders darauf zu achten, dass auch Fehlerfälle, wie z.B. ein fehlerhafter Lesevorgang im Programmablauf berücksichtigt wurden. Ein Teil des hinzugefügten Programmcodes befindet sich im Anhang 7.4.
Eine weitere Besonderheit waren die regulären Ausdrücke, mit denen die ID des Datenbankeintrages aus der URL extrahiert werden sollte. Mittels eines PHP-Skriptes habe ich dann eine Datenbankabfrage mit dieser ID programmiert. Als Antwort erhielt das Java-Programm die Produktionsdaten. Diese mussten entsprechend der Datenformate der SPS umgewandelt und via TCP/IP an die SPS gesendet werden, um sie temporär zu speichern und auf dem Visuali-

sierungsgerät anzuzeigen. Eine Übersicht über die Kommunikation meiner Erweiterung ist in dem zuvor erstellten Verteilungsdiagramm im Anhang 6.6 zu sehen. Die ganze Erweiterung am Java-Programm wurde zwischenzeitlich natürlich von Tests unterbrochen, um die Teilfunktionalität der Erweiterungen sicherzustellen.

Zum Schluss habe ich die erforderlichen Revisionsunterlagen für das Abnahmeprotokoll zusammengestellt. Dazu habe ich mir im Vorfeld eine Vorlage des in der Berufsschule verwendeten Protokolls von meinem Auftraggeber Herrn Manemann besorgt. Dieses habe ich dann ergänzt, sodass die Sichtkontrolle und die Funktionsprüfung auf meine Erweiterung zutreffend sind. Dazu habe ich mit Herrn Manemann Absprachen bezüglich der durchzuführenden Messungen getätigt. Diese beinhalteten beispielsweise die Missachtung der Messung der Restspannung an der Anlage, da es keine Komponenten gibt, die eine Restspannung aufweisen könnten.

Die Dokumentation meiner Arbeit auf der Moodle-Lernplattform www.plore-dna.net lief hierzu parallel ab. Hier habe ich Beiträge zu den Themen QR-Code, QR-Codeleser und Datenbankanbindung erstellt. Dies werde ich noch weiter verfeinern und erweitern.

2.5 Auftragskontrolle

In der abschließenden Inbetriebnahme mit Herrn Manemann nach DIN VDE 0100-600 habe zuerst eine Besichtigung der gesamten Anlage durchgeführt und in Bezug auf die von mir neu hinzugefügten Betriebsmittel und Leitungen ein Sichtprotokoll erstellt (siehe Anhang 8.2 Blatt 1). Dann habe ich an der gesamten Anlage aufbauend auf der erfolgreichen Sichtprüfung verschiedene Messungen durchgeführt (siehe Anhang 8.2 Blatt 2 und 3). Dazu verwendete ich Messgeräte, die nach DIN VDE 0411 für die Messungen zugelassen sind. Eine Prüfmittelliste ist im Anhang 6.3 zu finden. Ich begann mit der Niederohmmessung, bei der ich die zuvor ausgewählten und berechneten Widerstände des Schutzleitersystems überprüfte, im spannungsfreien Zustand der Anlage. Dann folgte die Isolationsmessung. Anschließend konnte ich die Anlage mit Spannung beschalten und als erstes die Schleifenimpedanz des Schaltschrankes bestimmen und mit dem vorher berechneten Werten vergleichen. Danach habe ich die RCD-Messung durchgeführt, wobei ich die Berührungsspannung, den Auslösestrom und die Auslösezeit gemessen habe. Schließlich folgten noch die Messung der verschiedenen Spannungen der Anlage, die Überprüfung der Polarität im Steuerstromkreis und eine Drehsinnprüfung. Nach den erfolgreich durchgeführten Messungen nahm ich die Erprobung der gesamten Anlage vor. Dazu überprüfte ich die Wirksamkeit des Not-Aus-Tasters und der anschließenden Quittierung des Not-Halts. Des Weiteren habe ich die richtige Drehrichtung des Motors und die korrekte Funktion der Meldeleuchten kontrolliert. Die Dokumentation zu der Erprobung befindet sich in Anhang 8.2 Blatt 3. Die Funktionskontrolle habe ich mit Herrn Manemann zum Schluss durchgeführt, da die Anlage dafür eingeschaltet sein muss. Hier überprüfte ich hauptsächlich die Funktionen, die ich neu hinzugefügt habe und fasste die Ergebnisse in einem Protokoll in Anhang 8.2 Blatt 3 und 4 zusammen.

Nachdem ich die Abnahme mit Herrn Manemann durchgeführt hatte, habe ich die Revisionsunterlagen zusammengefasst mit der Stückliste, den Datenblättern, dem SPS-Programm und dem Java-Programm digital auf dem Schulserver IServ abgelegt. Anschließend habe ich eine endgültige Kostenabrechnung der Arbeitszeit und des Materialverbrauches erstellt. Diese kann dem Anhang 8.1 entnommen werden und bezieht sich auf Gesamtkosten von knapp 3600 € brutto.

2.6 Betriebsmittelvorschriften und Normen

Die Erweiterung der Anlage habe ich nach DIN VDE erstellt. Da die Anlage mit Drehstrom bzw. Starkstrom bis 1000 V betrieben wird, muss ich die DIN VDE 0100-300 beachten. Für die Inbetriebnahme der Anlage habe ich die DIN VDE 0100-600 und die DIN VDE 0100-410 angewendet. Die BGV A3 habe ich somit auch eingehalten. Die Betriebsmittelkennzeichnung habe ich normgerecht nach DIN EN 81346-2 gewählt. Bei der Auswahl der Leitungen bezüglich des Querschnitts habe ich die DIN VDE 0100-540 und die DIN VDE 0100-520 verwendet. Die Betriebsmittelvorschriften der Volkswagen AG finden bei dieser Anlage keine Anwendung, da die Anlage nicht im Werk der Volkswagen AG betrieben wird.

2.7 Fotos

Abbildung 2.1: Gesamte M&M-Abfüllanlage mit allen Komponenten

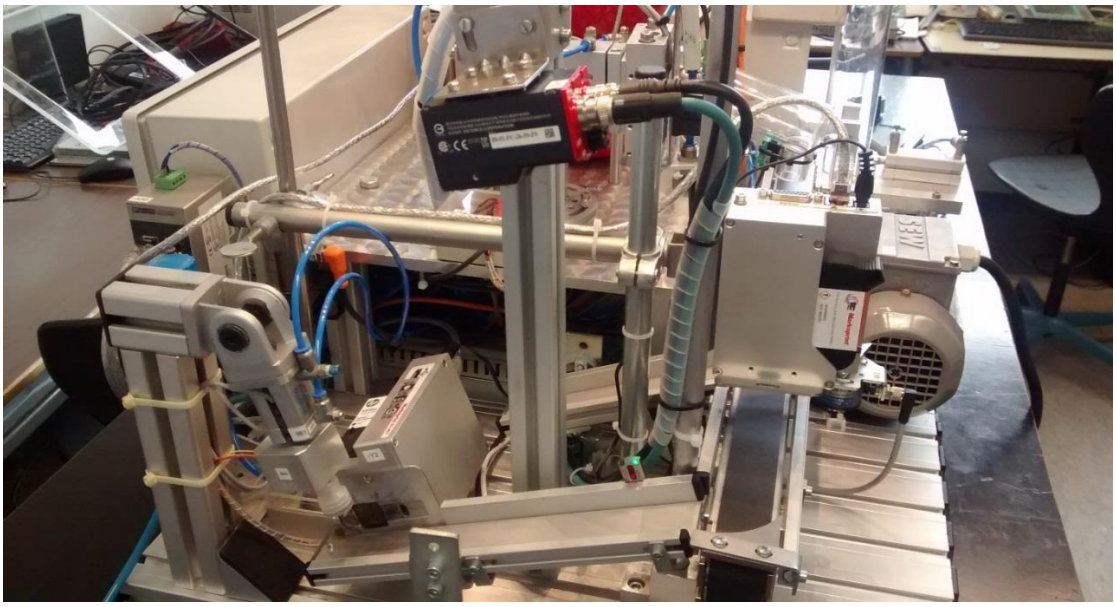


Abbildung 2.2: Übersicht über den Teil der Endbedruckung

3 Anhang zur Informationsbeschaffung

3.1 Lastenheft

Lastenheft

Lastenheft	**Auslesen eines QR-Codes im Produktionsprozess und Darstellung der Produktionsdaten auf einem Touchpanel**	**Datum:** 11.03.16 **Version:** 1.0
Auftraggeber	Herr Manemann	
Projektleiter	Herr Manemann	
Projektteam	Simon Häußler	
Zeitplanung	**Start:** 08.03.16 **Ende:** nach Absprache	
Projektmotivation	QR-Codes werden in Produktionsprozessen immer häufiger eingesetzt. Über QR-Codes auf einem Werkstück können Produktionsdaten eindeutig zugewiesen und über eine Datenbank verwaltet werden.	
Beschreibung des geplanten Projektes	○ Der QR-Code soll im laufenden Produktionsprozess von einem Werkstück übernommen und an eine Visualisierung übergeben werden. ○ +Wurde der QR-Code erkannt, soll das Werkstück im nächsten Produktionsschritt entsprechend der Daten in der Datenbank beschriftet werden. ○ +Der QR-Code-Reader ist in die Industrie-4.0-Anlage der BBS 2 Wolfsburg zu integrieren. ○ Das Touchpanel zur Anzeige der Produktionsdaten ist ebenfalls in der Anlage zu implementieren. ○ Die Funktionsweise der QR-Code-Technologie ist in der Lernplattform www.xplore-dna.net zu dokumentieren.	
Beschreibung IST-Zustand	○ In der Anlage ist bislang noch QR-Code-Reader integriert.	
Anlagen	○ SPS-Programm der bestehenden Anlage ○ CAD-Zeichnungen der Anlage in EPLAN P8 Education. ○ Zeichnungsunterlagen und Zeichnungsdaten der Anlage	
Budget	○ 100 €	

Abbildung 3.1: Lastenheft zum betrieblichen Auftrag

4 Anhang zur Auftragsplanung

4.1 Zeitplanung

Tabelle 4-1: Übersichtliche Zeitplanung mit Meilensteinen des Projektverlaufes

Projektplanung pro Woche — **Verantwortliche(r):** Simon Häußler — **Stand:** 18.05.2016

Bitte Aufgaben, Wer, Zeit und Termin eintragen!	Wer	Zeit in h	Termin (Woche), Status		1. Woche Startdatum: Mo	Di	Mi	Do	Fr	2. Woche Mo	Di	Mi	Do	Fr	3. Woche Mo	Di	Mi	Do	Fr
Aufgaben						BBS					BBS								BBS
Informationsphase					*Mit Aufgaben belegte Tage kennzeichnen!*														
1 Einweisung in die bestehende Anlage	*SH*	*0,5*	*18.04.16*	✓	X														
2 Übergabe des Lastenheftes mit Besprechung	SH	0,25	18.04.16	✓	X														
3 Datenblätter, Benutzerhandbücher heraussuchen	SH	0,5	20.04.16	✓			X												
4 Ansatzpunkte zur Bearbeitung notieren	SH	0,25	20.04.16	✓			X												
5 Einarbeitung in Richtlinien, Normen und Vorschriften	SH	0,25	20.04.16	✓			X												
6 Abstimmung mit Auftraggeber, Fragenklärung	SH	0,25	20.04.16	✓			X												
7 Meilenstein: Informationen zusammengestellt	SH	0	21.04.16	✓				X											
8																			
9																			
10																			
11															Pfingsten				
Summe Stunden Informationsphase:		2																	
Planungsphase																			
12 Erstellung einer Zeitplannung mit Meilensteinen	SH	0,5	21.04.16	✓				X											
13 Erstellung einer Materialliste	SH	0,25	21.04.16	✓				X											
14 Erstellung von Werkzeug- und Prüfmittelliste	SH	0,25	21.04.16	✓				X											
15 Zusammensuchen der Materialien gemäß der Liste	SH	0,5	22.04.16	✓					X										
16 Zusammensuchen der Werkzeuge gemäß der Liste	SH	0,25	22.04.17	✓					X										
17 Zusammensuchen der Prüfmittel gemäß der Lis	SH	0,25	22.04.18	✓					X										
18 Vorbereitung von Revisionsunterlagen	SH	0,5	22.04.19	✓					X										
19 Festlegung der Betriebsmttelbefestigung und -position	SH	1	09.05.16	✓						X									
20 Festlegung der Betriebsmittelanordnung	SH	0,5	09.05.16	✓						X									
21															Pfingsten				
Summe Stunden Planungsphase:		4																	
Durchführungsphase																			
22 Montage der Betriebsmittel	SH	1,5	11.05.16	✓								X							
23 Leitungen auswählen und Betriebsmittel verbinden	SH	0,5	11.05.16	✓								X							
24 QR-Codeleser ausrichten und konfigurieren	SH	1	12.05.16	✓									X						
25 Projektierung der SPS mit Hardwarekonfiguration	SH	0,75	12.05.16	✓									X						
26 Schnittstellen konfigurieren	SH	0,5	12.05.16	✓									X						
27 Steuerungsprogramm der SPS erstellen	SH	2	12.05.16	✓									X						
28 Programmierung des Raspberry Pi für Datenbankabruf	SH	2	13.05.16	✓										X					
29 Programmierung des Touch Panels	SH	0,5	17.05.16	✓											Pfingsten	X			
30 Inbetriebnahme nach DIN VDE 0100-600	SH	0,75	17.05.16	✓												X			
31 ggf. Optimierungsmaßnahmen ergreifen	SH	0,5	17.05.16	✓												X			
Summe Stunden Durchführungsphase:		10																	
Kontrollphase																			
32 Durchführung einer Funktionsprüfung	SH	0,5	18.05.16	✓													X		
33 Übergabe an den Kunden	SH	0,5	18.05.16	✓													X		
34 Übergabe Sicht- und Messprotokoll	SH	0,25	18.05.16	✓													X		
35 Schaltplan ausgedruckt in den Schaltschrank legen	SH	0,25	18.05.16	✓													X		
36 Stückliste, Datenblätter, Technologieschema übergeben	SH	0,25	18.05.16	✓											Pfingsten		X		
37 Erstellung einer Kostenabrechnung	SH	0,25	18.05.16	✓													X		
Summe Stunden Kontrollphase:		2																	
Gesamte Dauer in Stunden		18																	

Projekt: "Integration eines QR-Code Scanners" **Stand:** Mai 2016

Pos.	Stück	Hersteller	Artikelnummer	Typ / Produktbeschreibung	Funktion	Einzelpreis	Gesamtpreis
SPS und Komponenten							
1	1	Siemens	6ES7214-1AG40-0XB0	S7-1200 SPS	Steuerung des Endprozesses	348,44 €	348,44 €
2	1	Siemens	6AV6647-0AD11-3AX0	KTP600 Basic Color PN Touch Panel	Visualisierung der Produktionsdaten	499,80 €	499,80 €
3	1	Keyence	SR-1000	2D-Codeleser mit Autofokus und Polarisationsfilter	Scannung von QR-Codes	2.500,00 €	2.500,00 €
4	1	Keyence	OP-87866	Montagehalterung für SR-Baureihe	Befestigung des Codelesers	mitgeliefert	0,00 €
5	1	Keyence	PR-FB30P3	PR-FB30P3, Reflexionslichttaster, 30 mm, PNP, 3 m Leitung	Lichtschranke Rutsche am Bandende	111,50 €	111,50 €
6	1	Hr. Böspflug	-	Winkelprofil auf Aluminium mit mehreren Bohrungen	Halterung des Codelesers		0,00 €
7	1	Bestand	-	Item-Profil, 30 mm X 30 mm	Halterung des Codelesers		0,00 €
Antrieb und Zubehör							
Klemmleiste und Zubehör							
Leintungen und Zubehör							
8	1	Keyence	OP-87353	Steuerungsleitung, 2 m, NFPA-79 konform, 12-polig	Stromversorgung des Codelesers	mitgeliefert	0,00 €
9	1	Keyence	OP-87230	Ethernetleitung,2 m, NFPA-79 konform	Ethernetanbindung des Codelesers	mitgeliefert	0,00 €
10	2	Bestand	-	Ethernetleitungen, 1 m	Vernetzung der SPS und des Touch		0,00 €
Schaltschrank und Zubehör							
Verbrauchsmaterial							
11	5 m	4510022	Lapp Kabel	H05V-K 0,75 mm² blau, Aderleitung	Verdrahtung Steuerstromkreis	0,09 €	0,45 €
12	1	1208979	Phoenix Contact	Aderendhülsen, 0,75 mm², 8 mm, teilisoliert, grau, 500 Stck.	gasdichtes Verschließen der Aderenden	4,98 €	4,98 €
13	1	Bestand	-	Spiralband	zusammenführen der Leitungen		0,00 €
14	10	Bestand	-	Kabelbinder weiß	Leitungsbefestigung		0,00 €
15	diverse	Bestand	-	Schrauben, Muttern, Unterlegscheiben	Befestigungen		0,00 €
					Gesamt		**3.465,17 €**

Abbildung 4.1: Stückliste der verwendeten Materialien

4.3 Werkzeug- und Prüfmittelliste

Tabelle 4-2: Übersicht über die verwendeten Werkzeuge, Software und Prüfmittel

Lfd. Nr.	Bezeichnung
Werkzeuge	
1	Aderendhülsenquetschzange (Crimpzange)
2	Diverse Kreuzschraubendreher
3	Diverse Schlitzschraubendreher
4	Diverse Innensechskantschlüssel (Imbusschlüssel)
5	Kraftseitenschneider
6	Micro Seitenschneider
7	USB-Beschriftungsgerät DYMO
8	Holzgliedermaßstab
9	Schieblehre
Software - Werkzeuge	
10	ePlan Education Version 2.5
11	Tia-Portal V13von Siemens
12	AutoID Network Navigator (SR-H5W) von Keyence
13	FluidSIM 5
Prüfmittel	
14	zweipoliger Spannungsprüfer
15	Fluke 1653 Multifunctiontester
16	Multimeter PeakTech 3335 DMM

4.4 Maßzeichnung Halterung QR-Codeleser

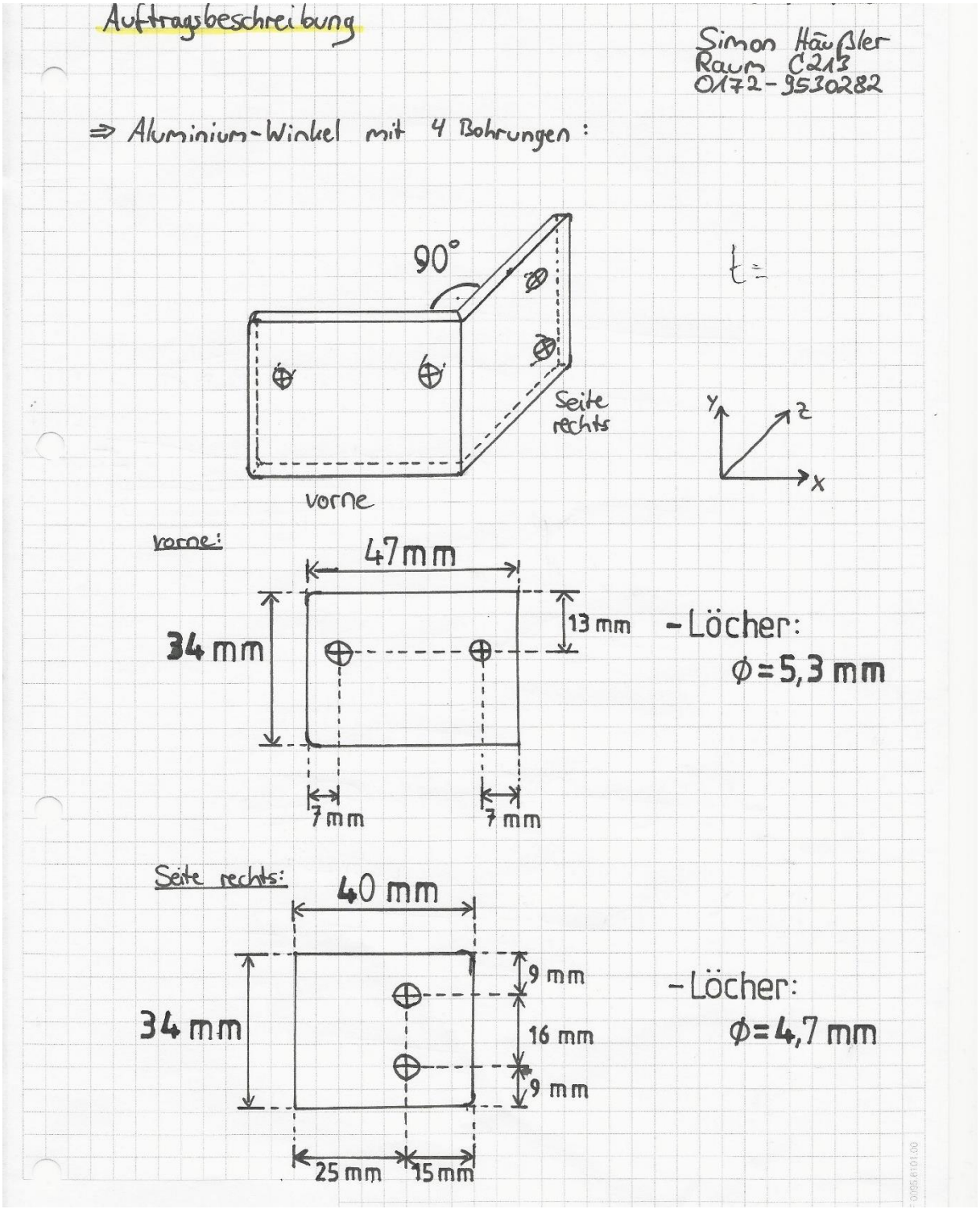


Abbildung 4.2: Maßzeichnung des Befestigungswinkels des 2D-Codelesers für Herrn Böspflug

4.5 Schaltplan der Erweiterung

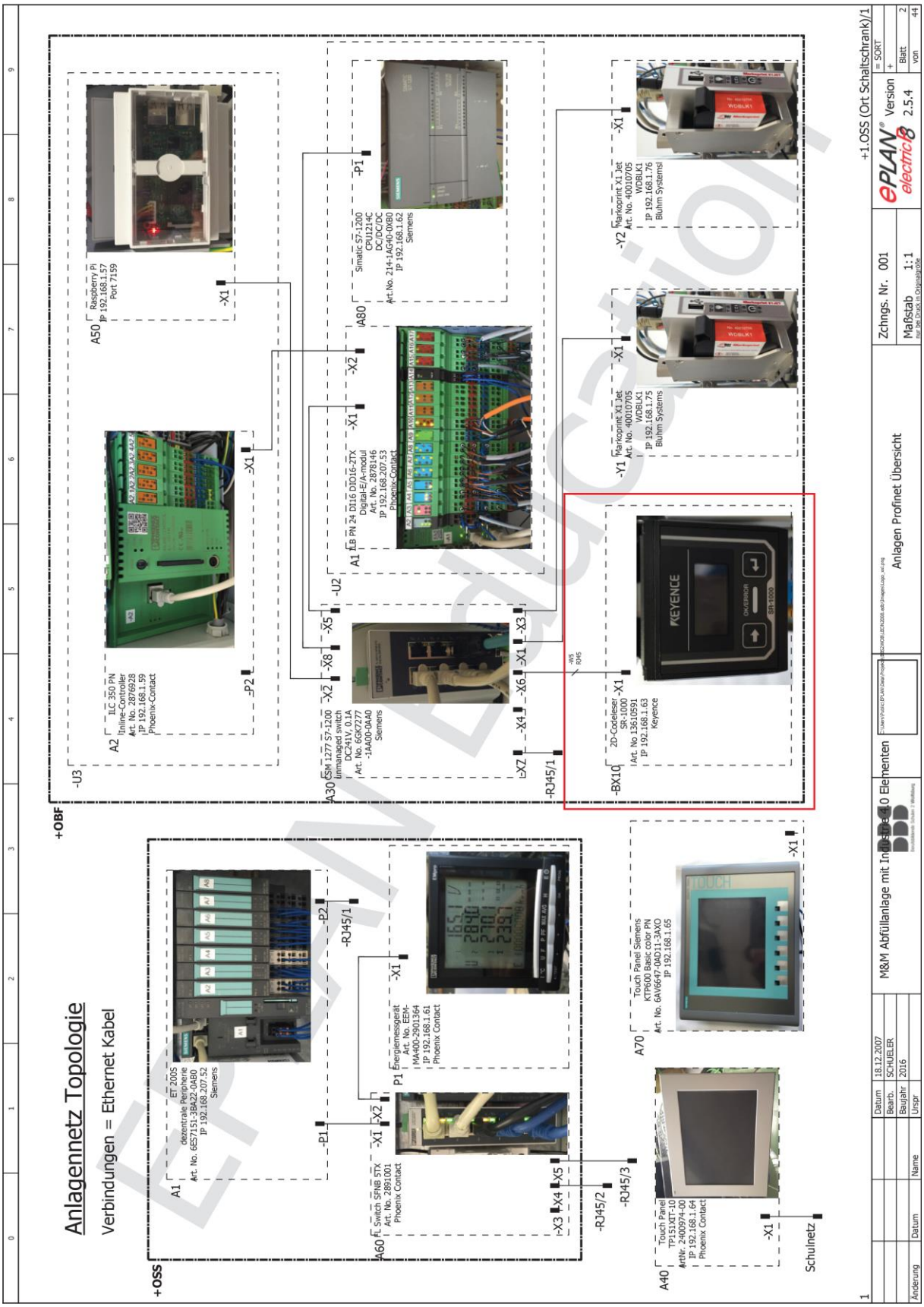


Abbildung 4.3: Übersicht der Profinet-Komponenten der Anlage

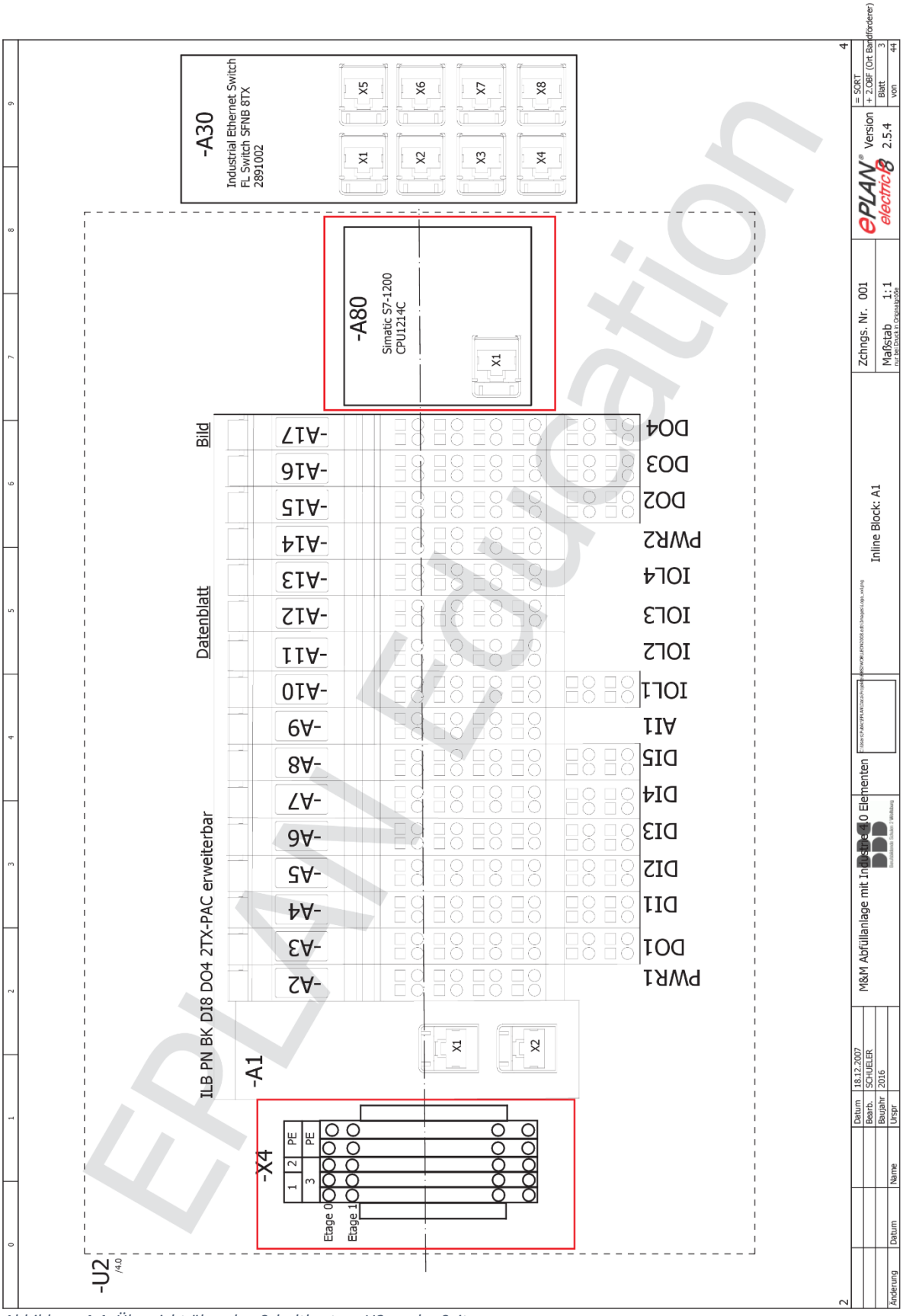


Abbildung 4.4: Übersicht über den Schaltkasten -U2 an der Seite

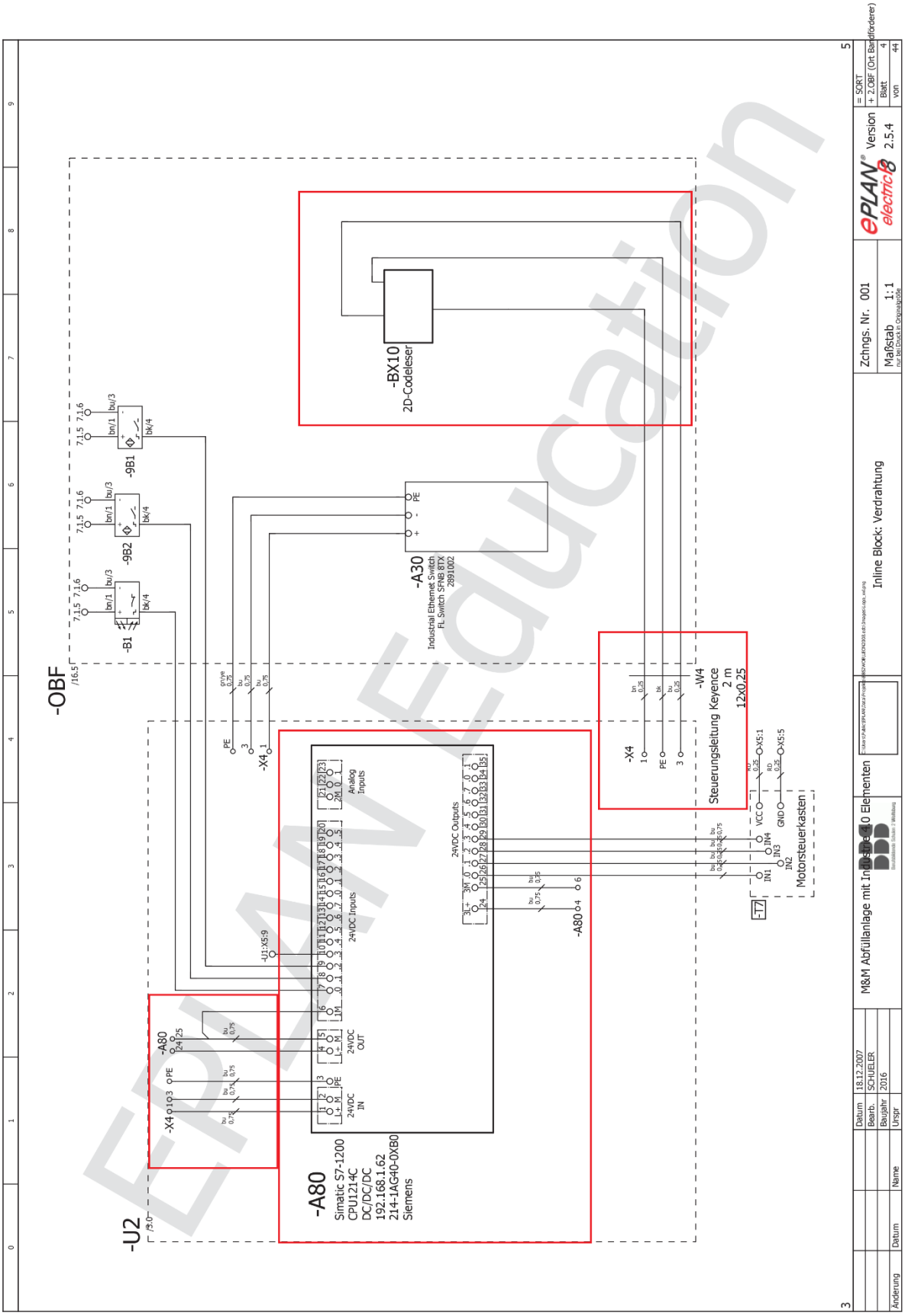


Abbildung 4.5: Verschaltung der SPS im Schaltkasten und Anschluss des QR-Codelesers

4.6 Verteilungsdiagramm der erweiterten Komponenten

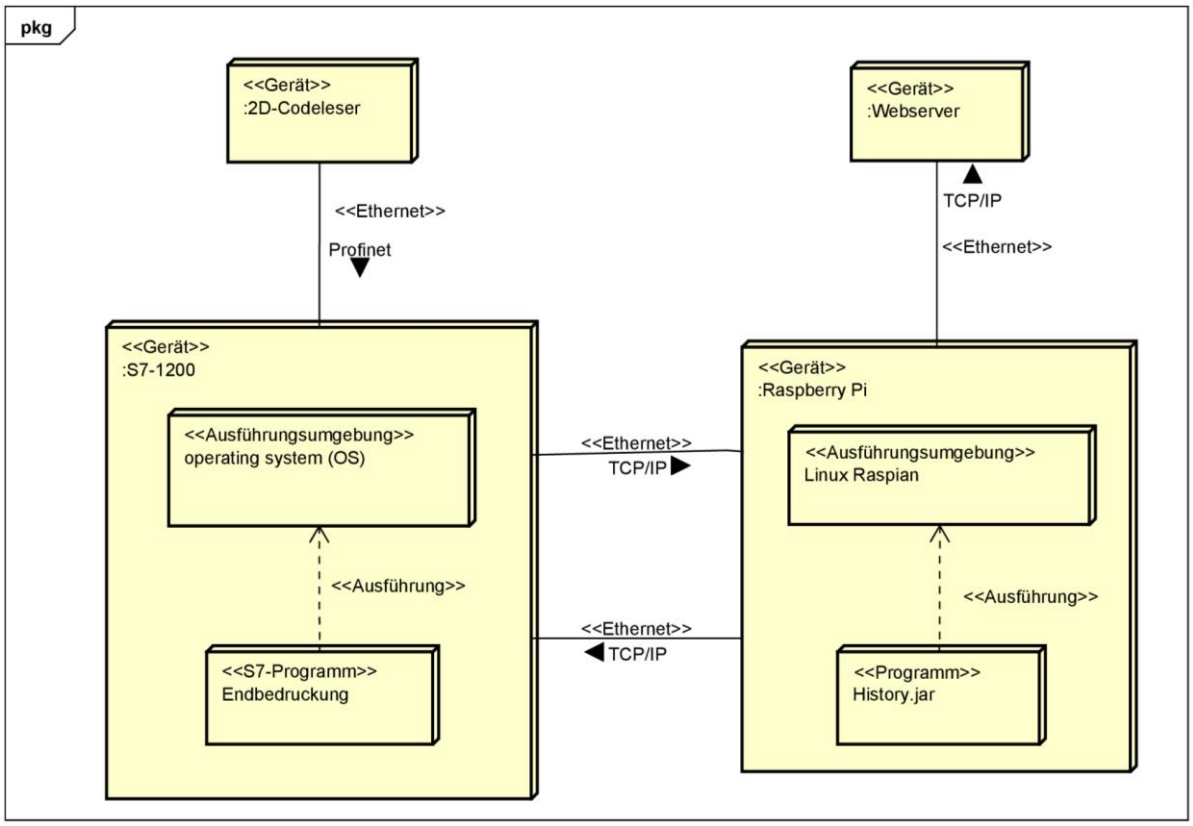


Abbildung 4.6: Verteilungsdiagramm der verschiedenen Komponenten und deren Kommunikation

4.7 Sequenzdiagramm der Methode processQRScan()

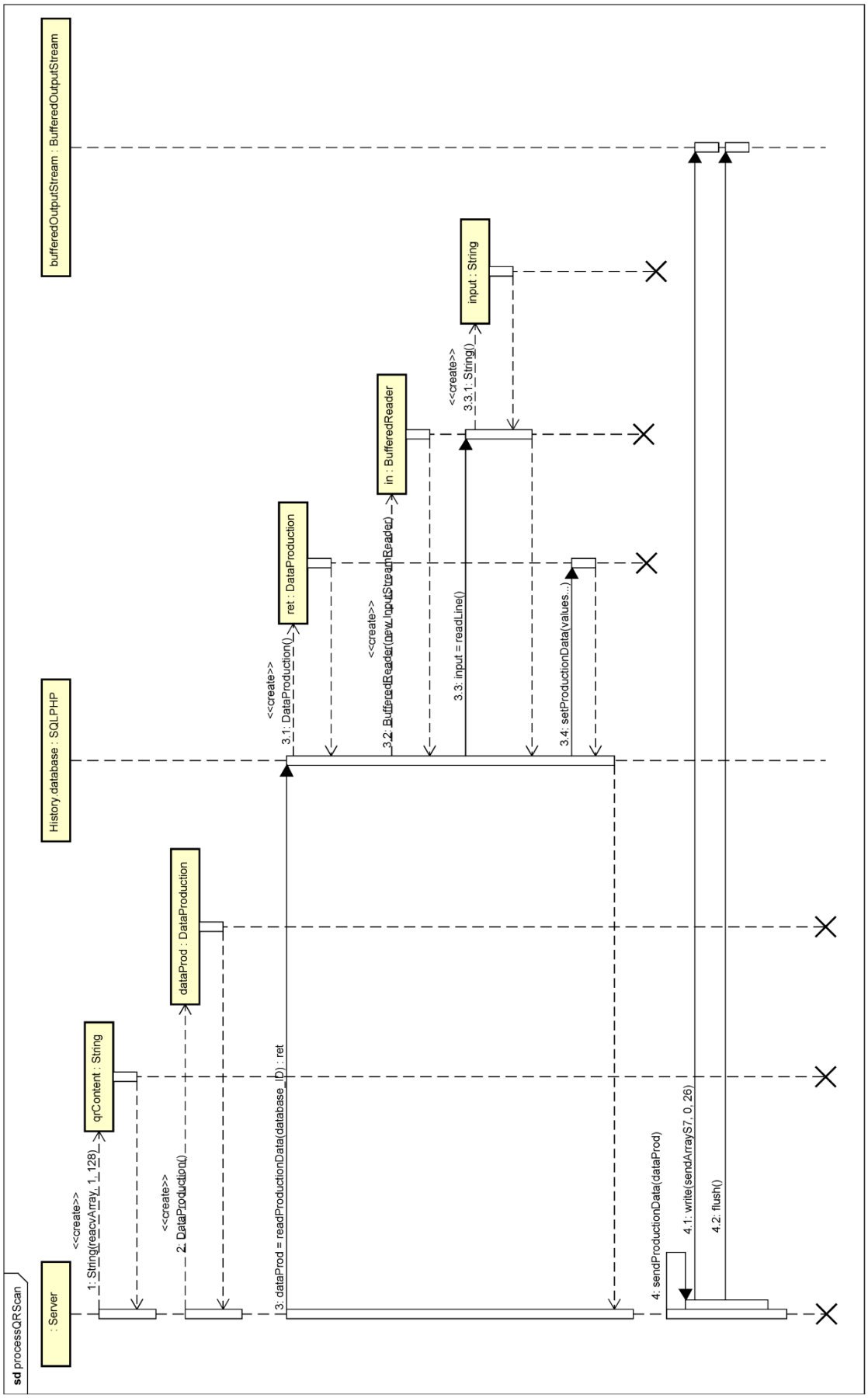


Abbildung 4.7: Sequenzdiagramm der Methode processQRScan() aus der Klasse Server.java

5 Anhang zur Auftragsdurchführung

5.1 Schrittkette des SPS-Programmes

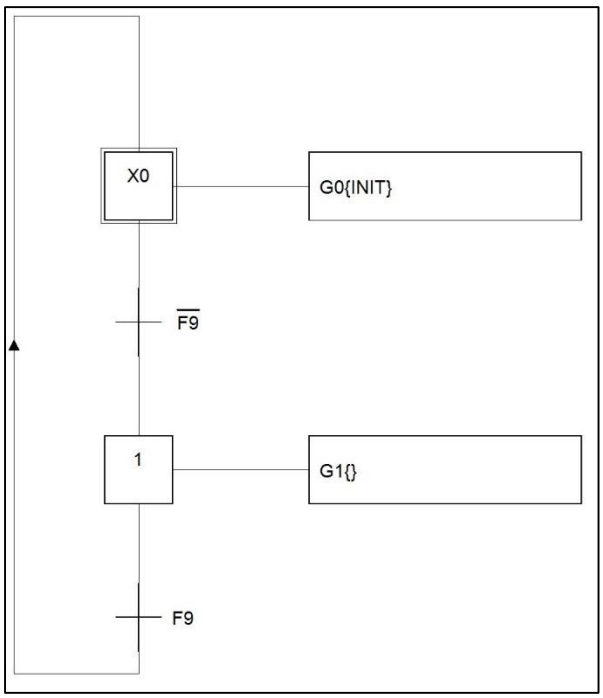


Abbildung 5.1: Sicherheitsschrittkette G0 des Programmablaufes

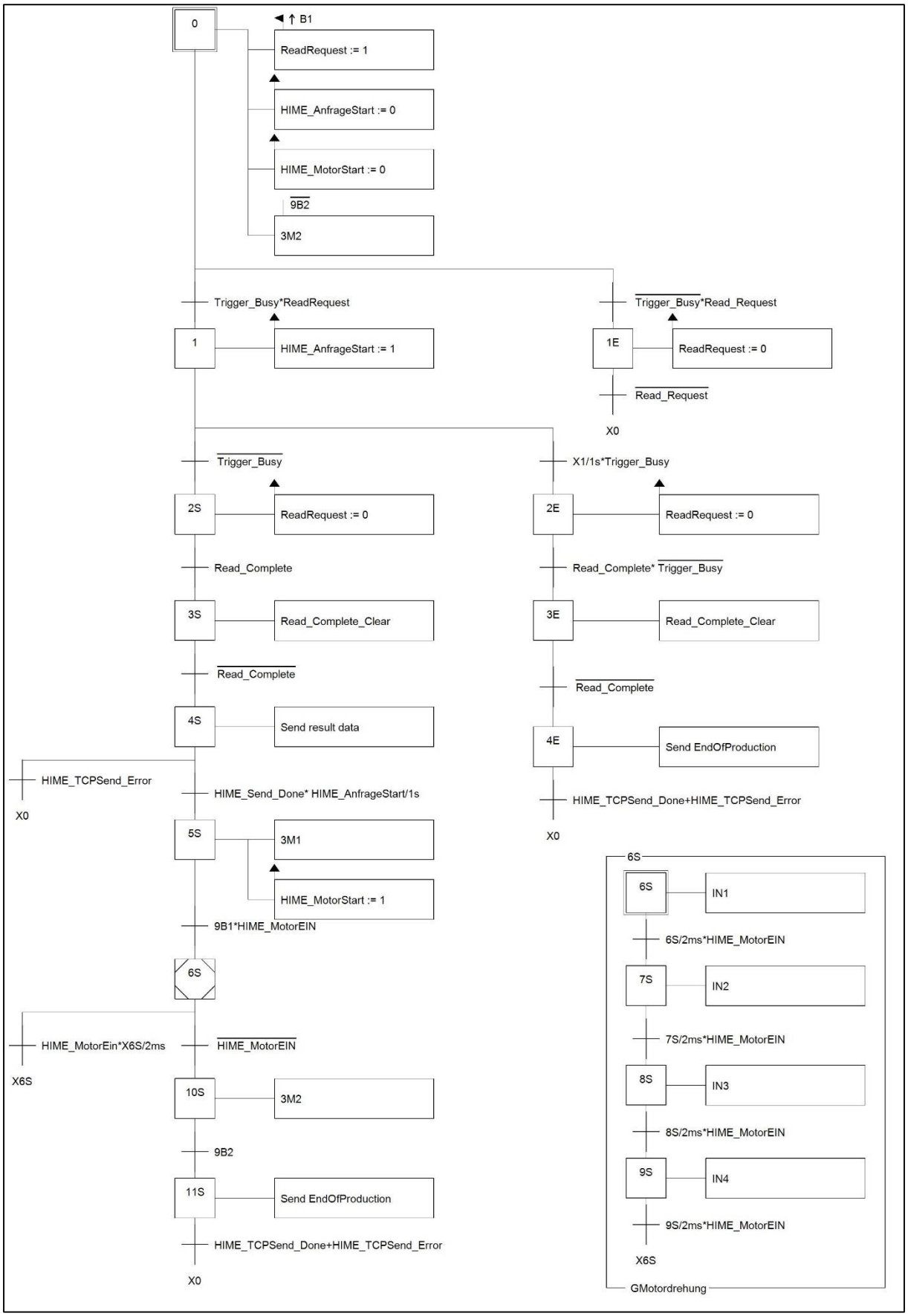


Abbildung 5.2:Hauptschrittkette G1 des Ablaufes einer Endbedruckung einer Dose

5.2 SPS-Programm

5.2.1 OB1-Main

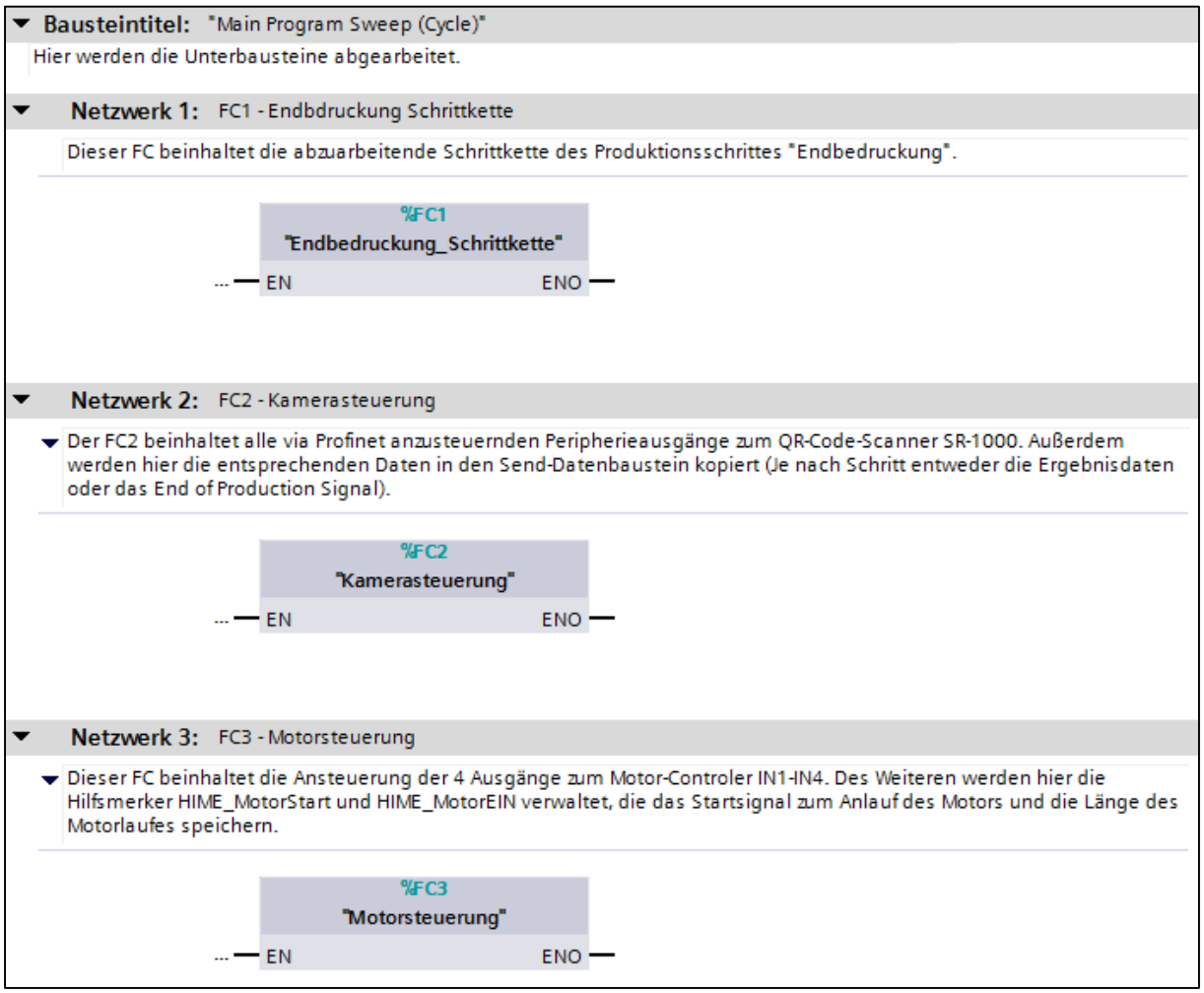


Abbildung 5.3: OB1 – Main Netzwerk 1-3

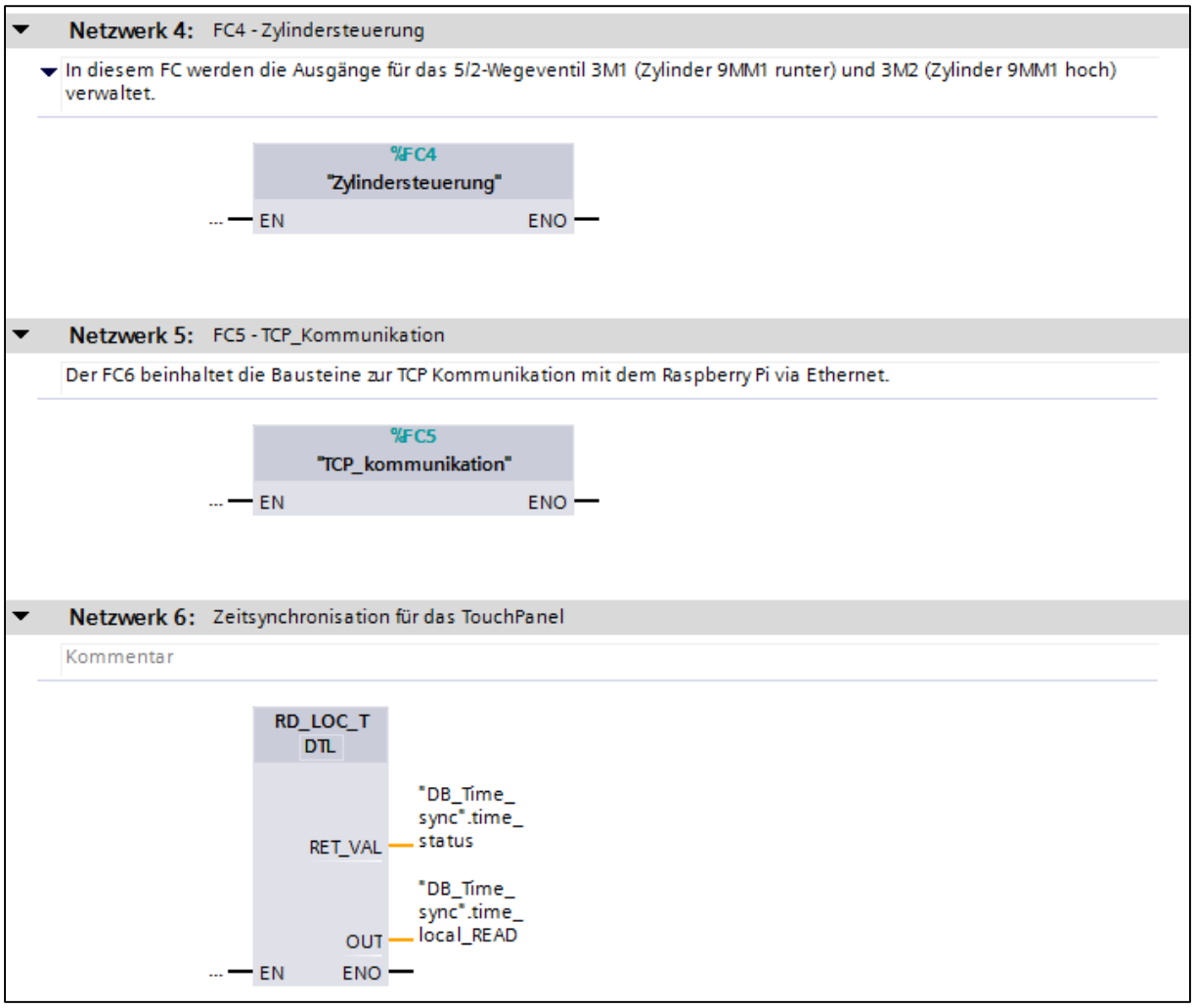


Abbildung 5.4: OB1 - Main Netzwerk 4-6

5.2.2 FC1 – Endbedruckung_Schrittkette

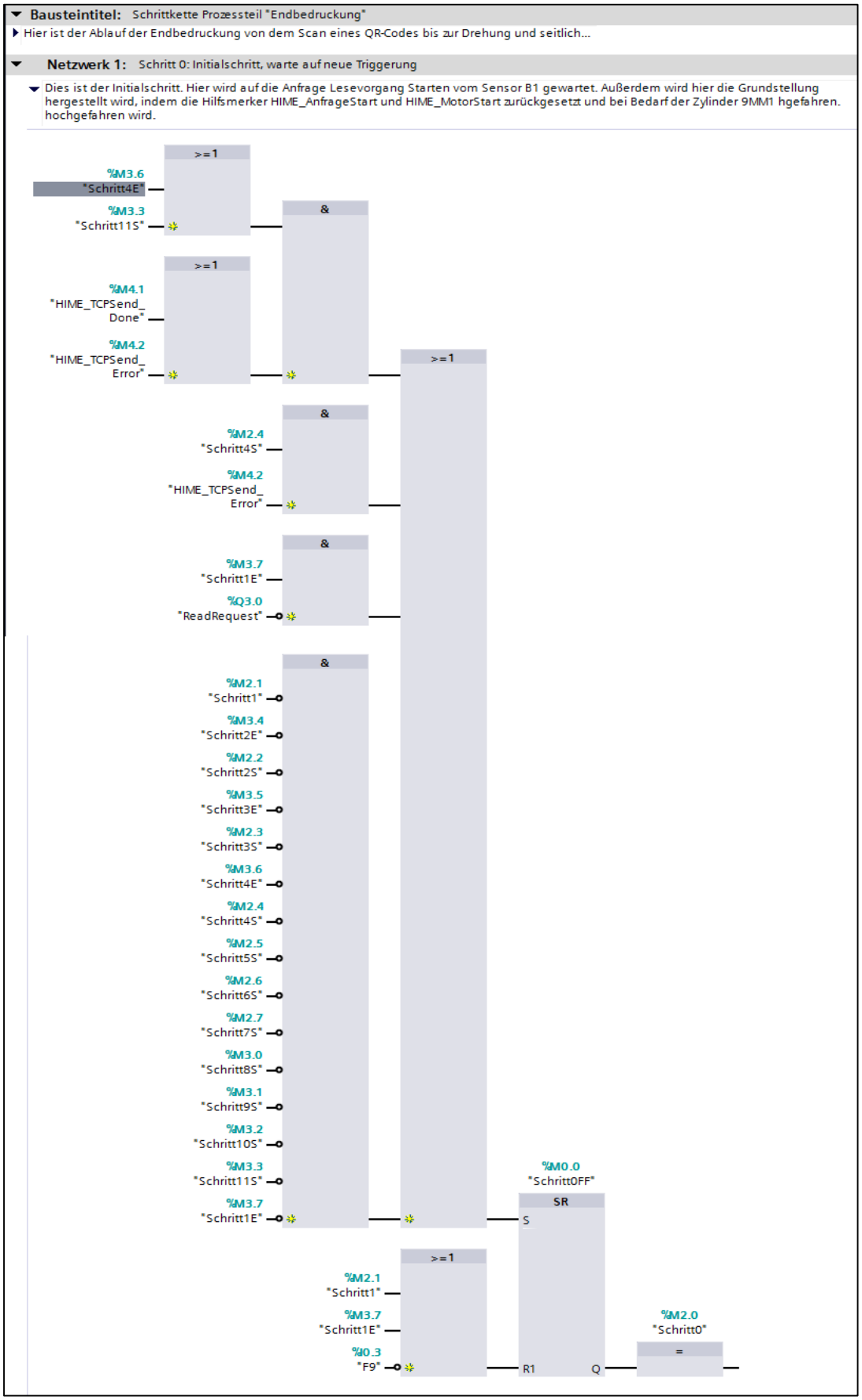


Abbildung 5.5: FC1 Netzwerk 1

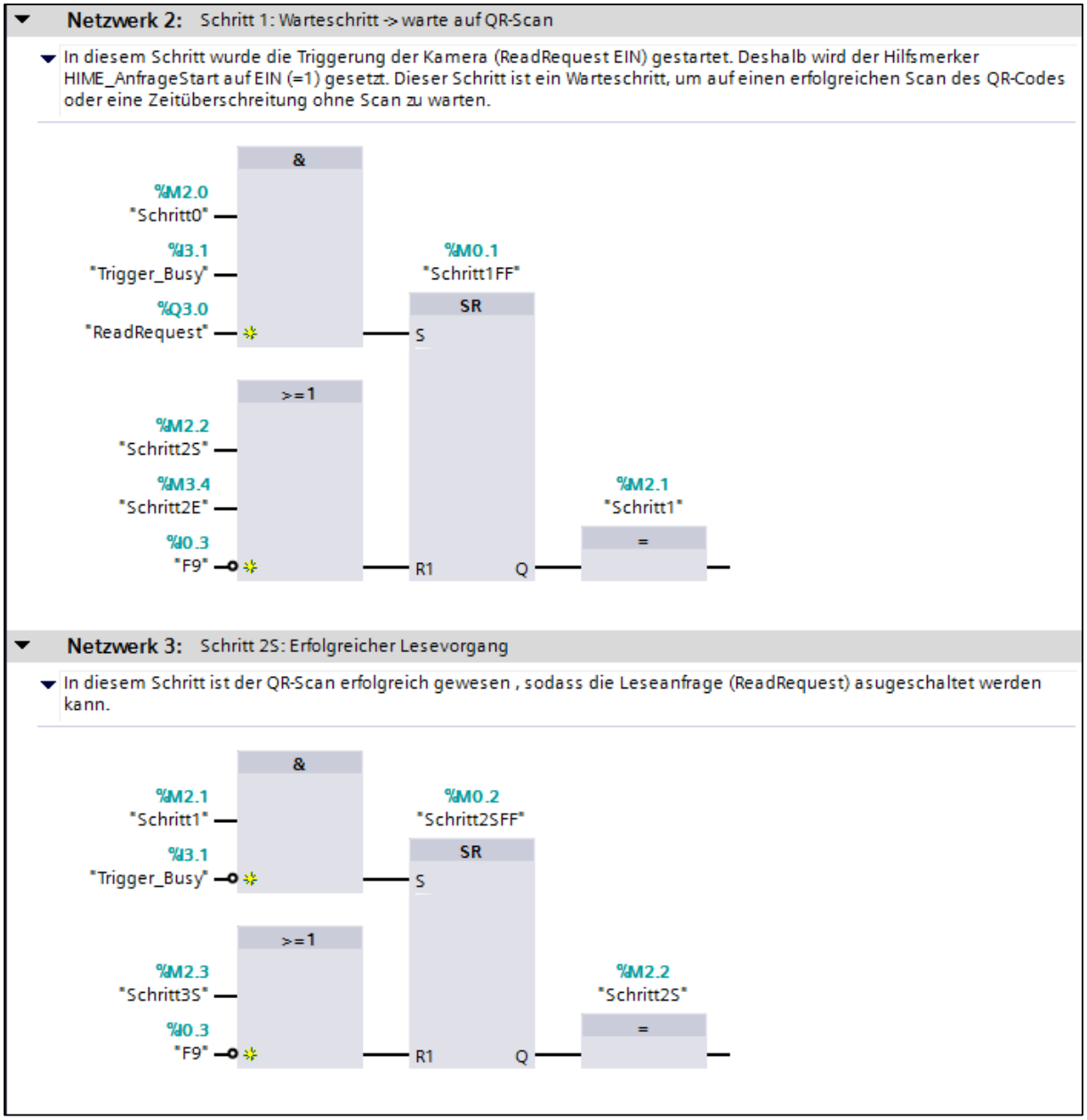


Abbildung 5.6: FC1 Netzwerk 2,3

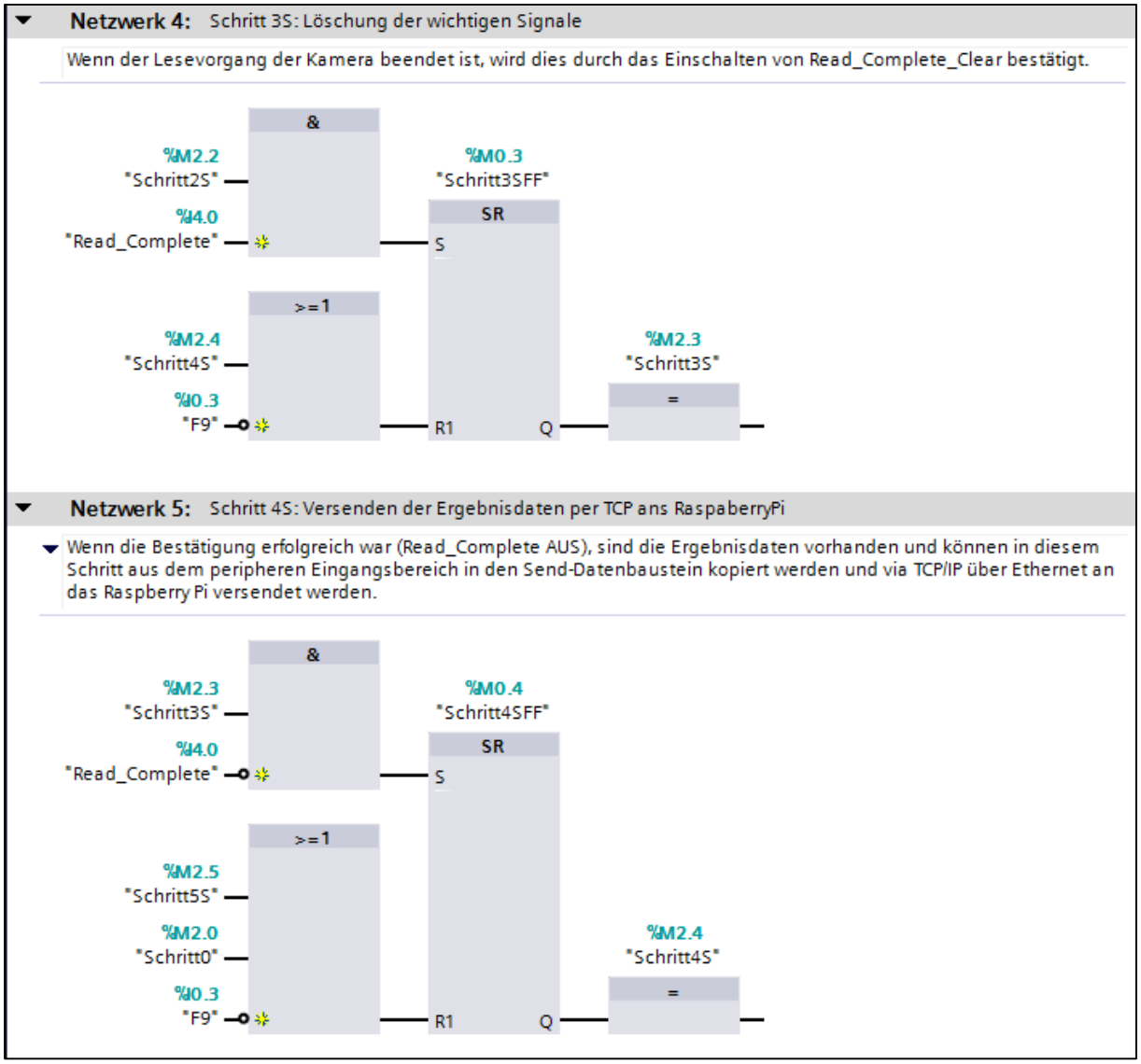


Abbildung 5.7: FC1 Netzwerk 4,5

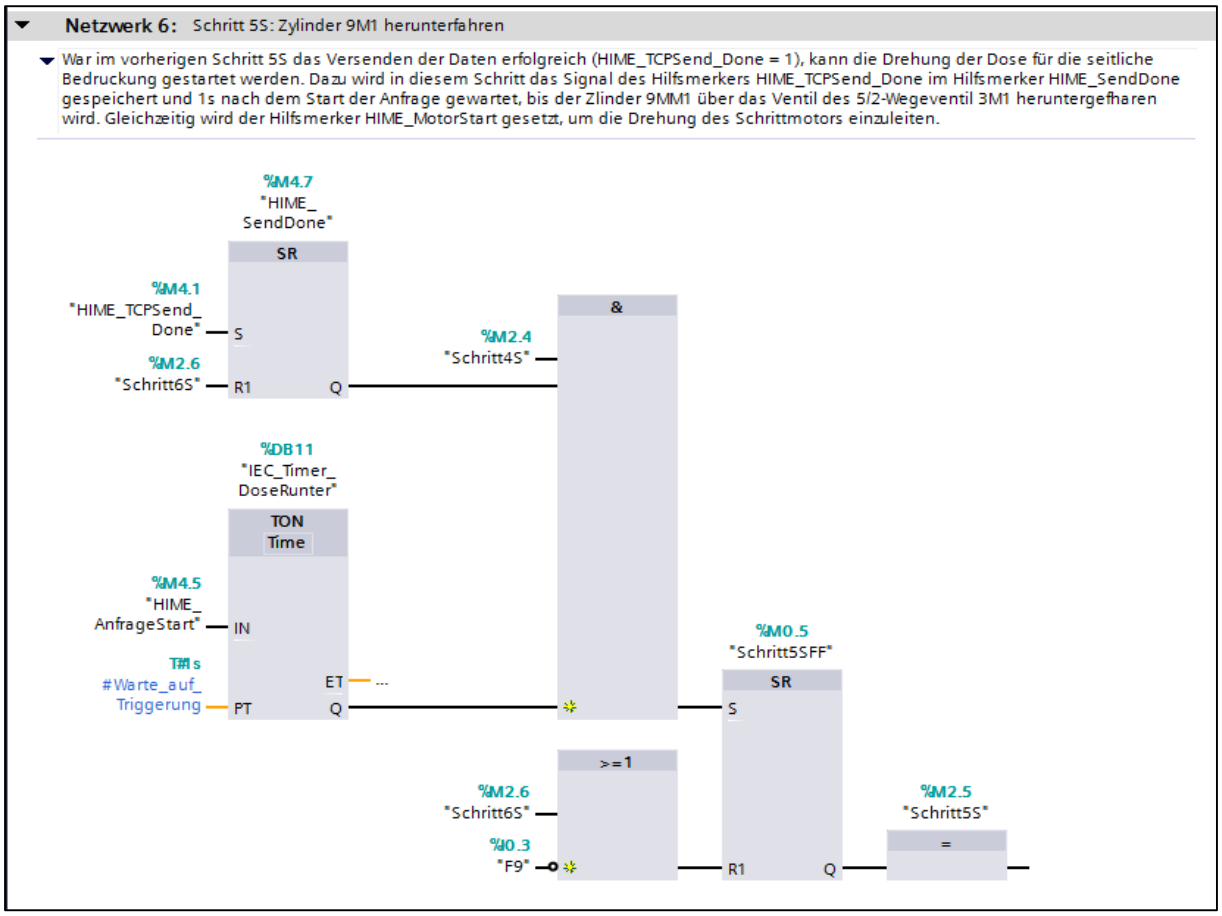


Abbildung 5.8: FC1 Netzwerk 6

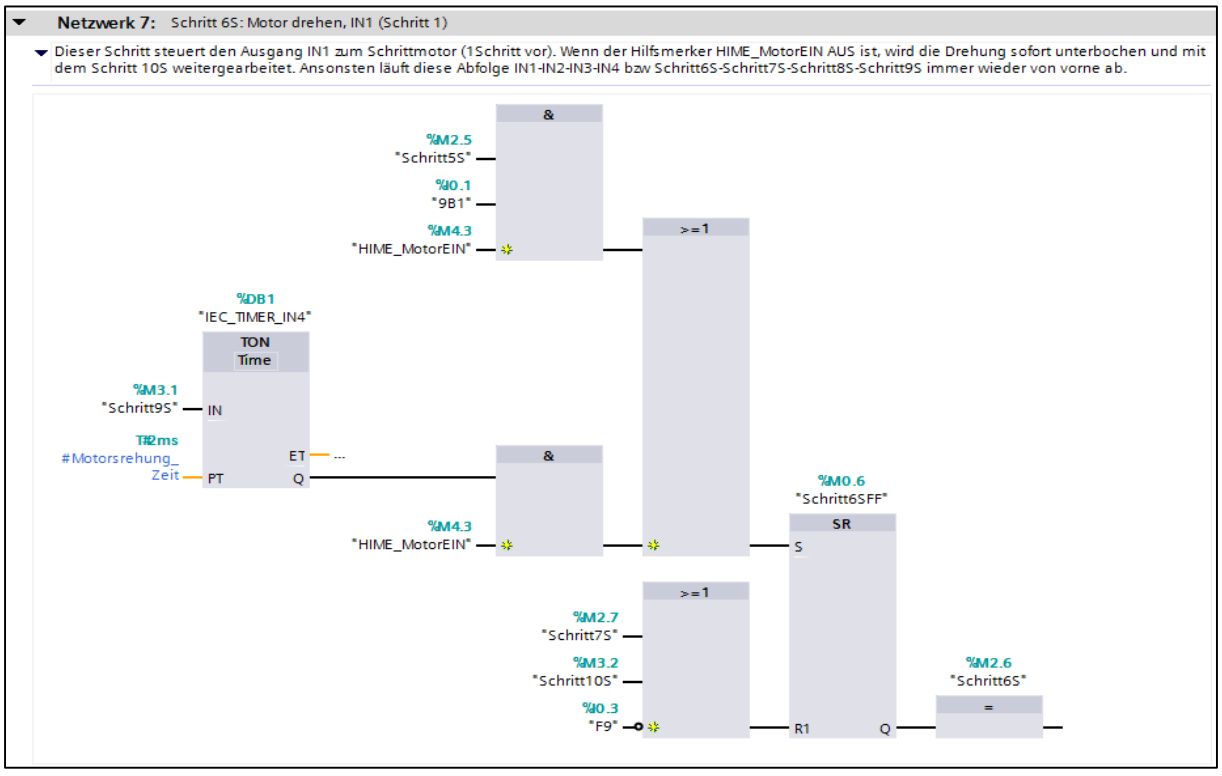


Abbildung 5.9: FC1 Netzwerk 7

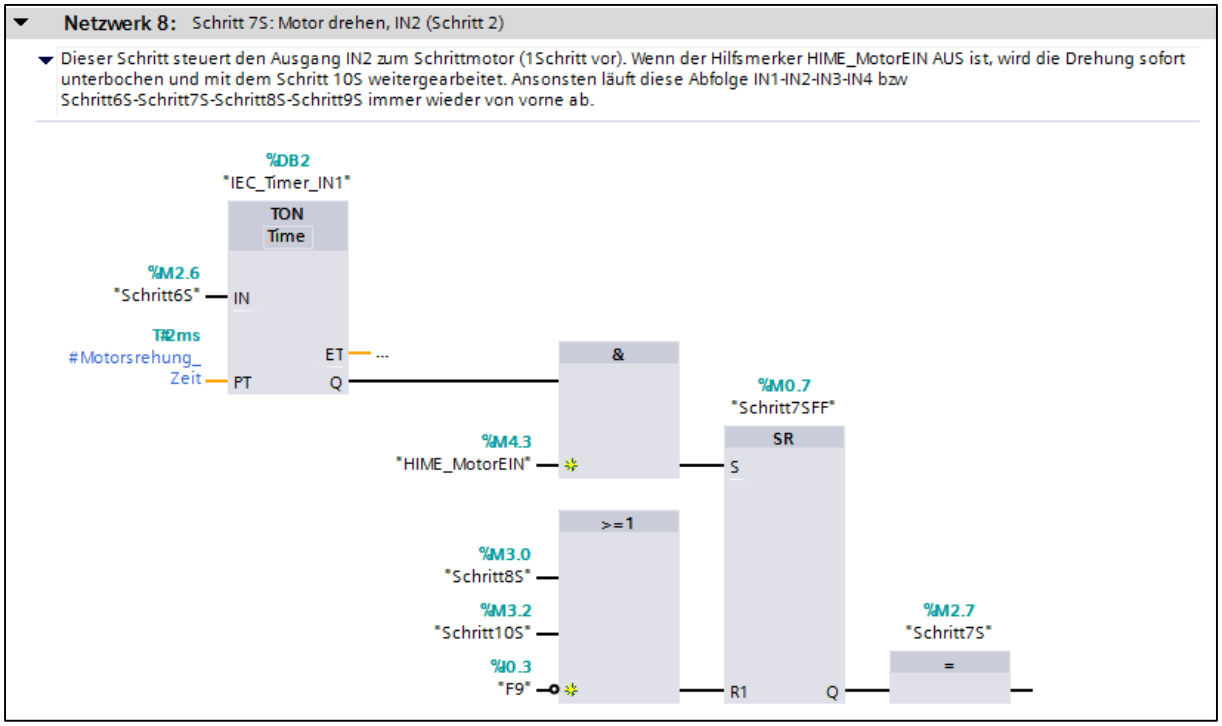


Abbildung 5.10: FC1 Netzwerk 8

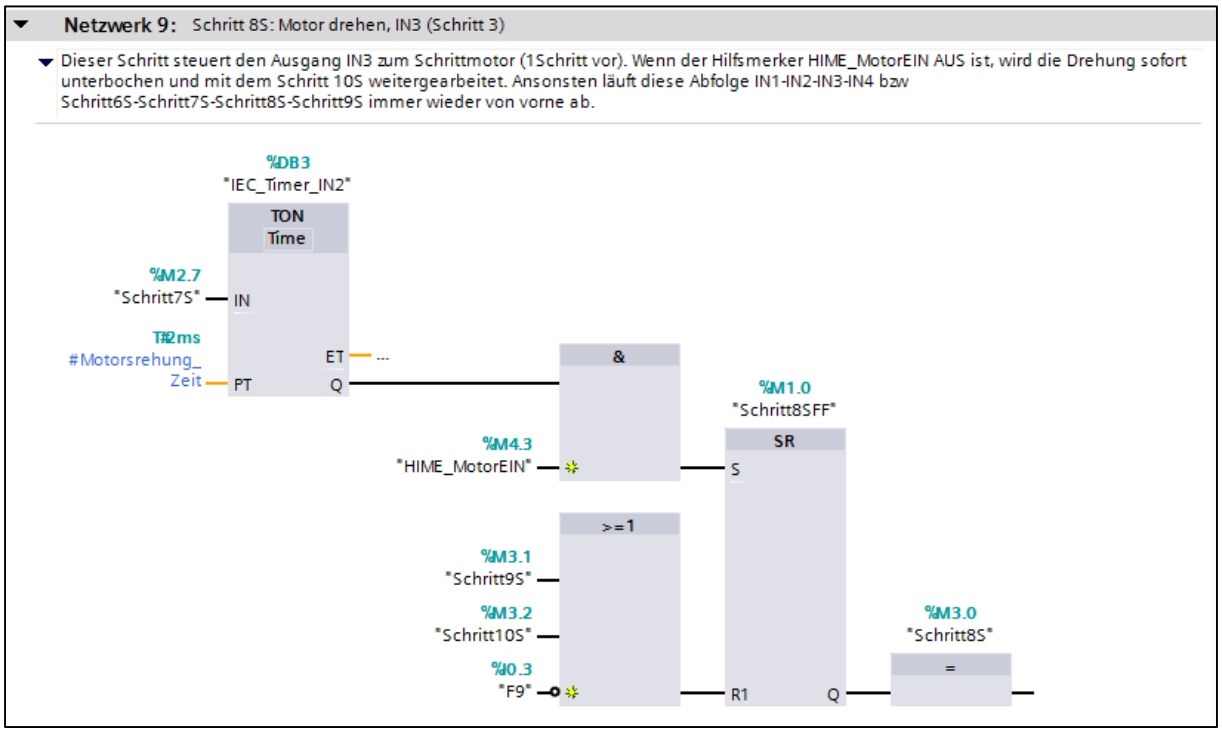


Abbildung 5.11: FC1 Netzwerk 9

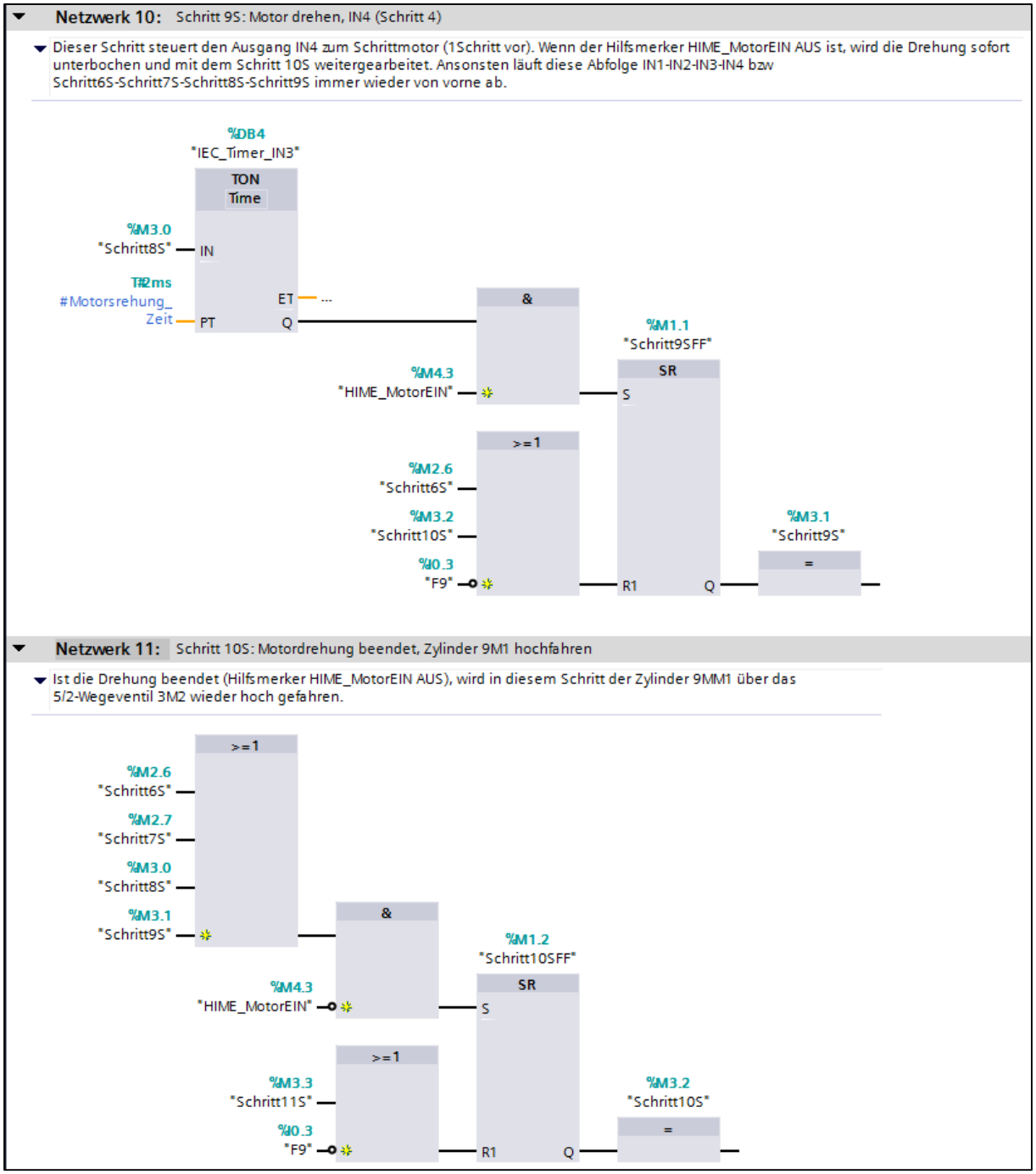


Abbildung 5.12: FC1 Netzwerk 10,11

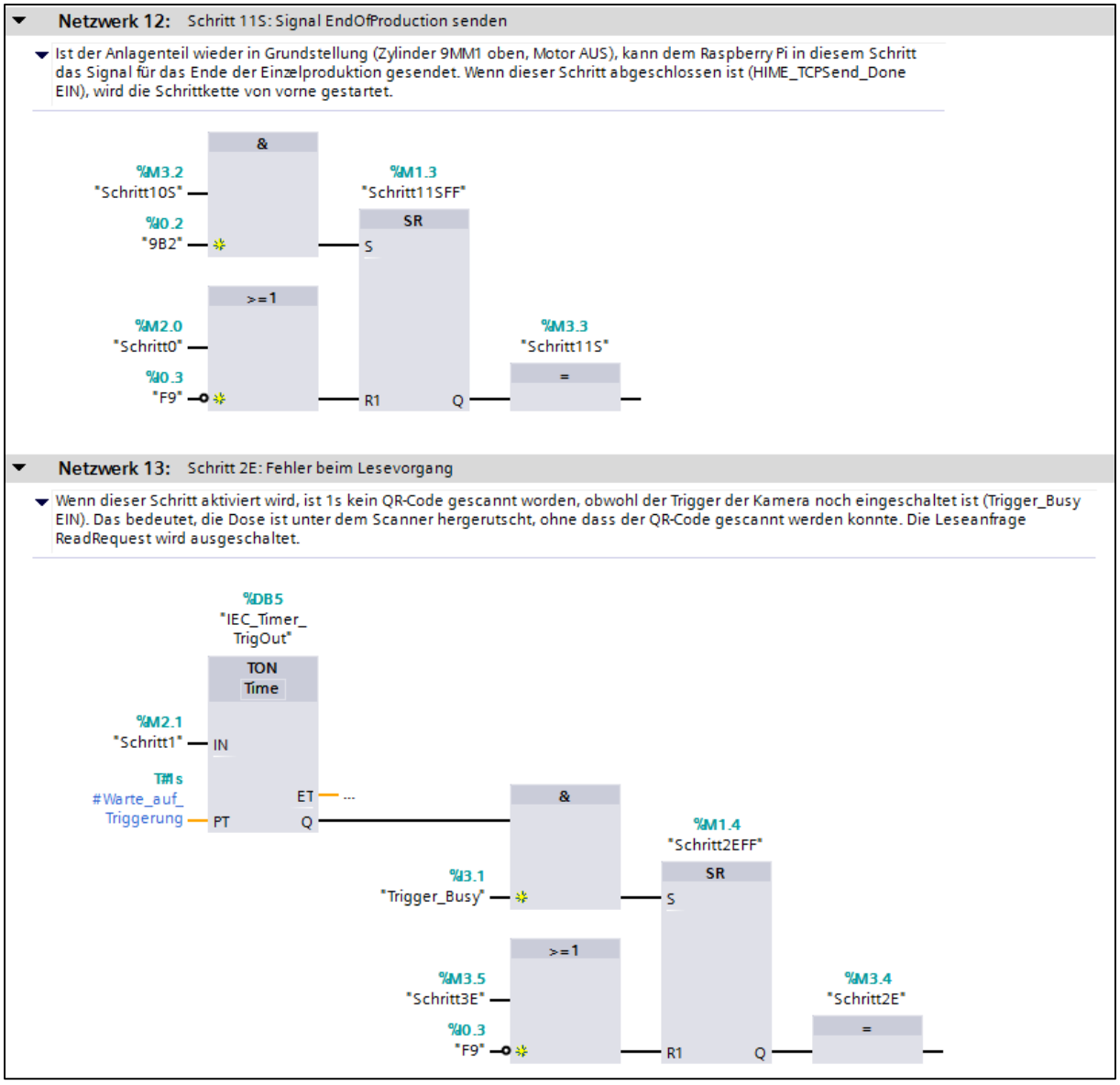


Abbildung 5.13: FC1 Netzwerk 12,13

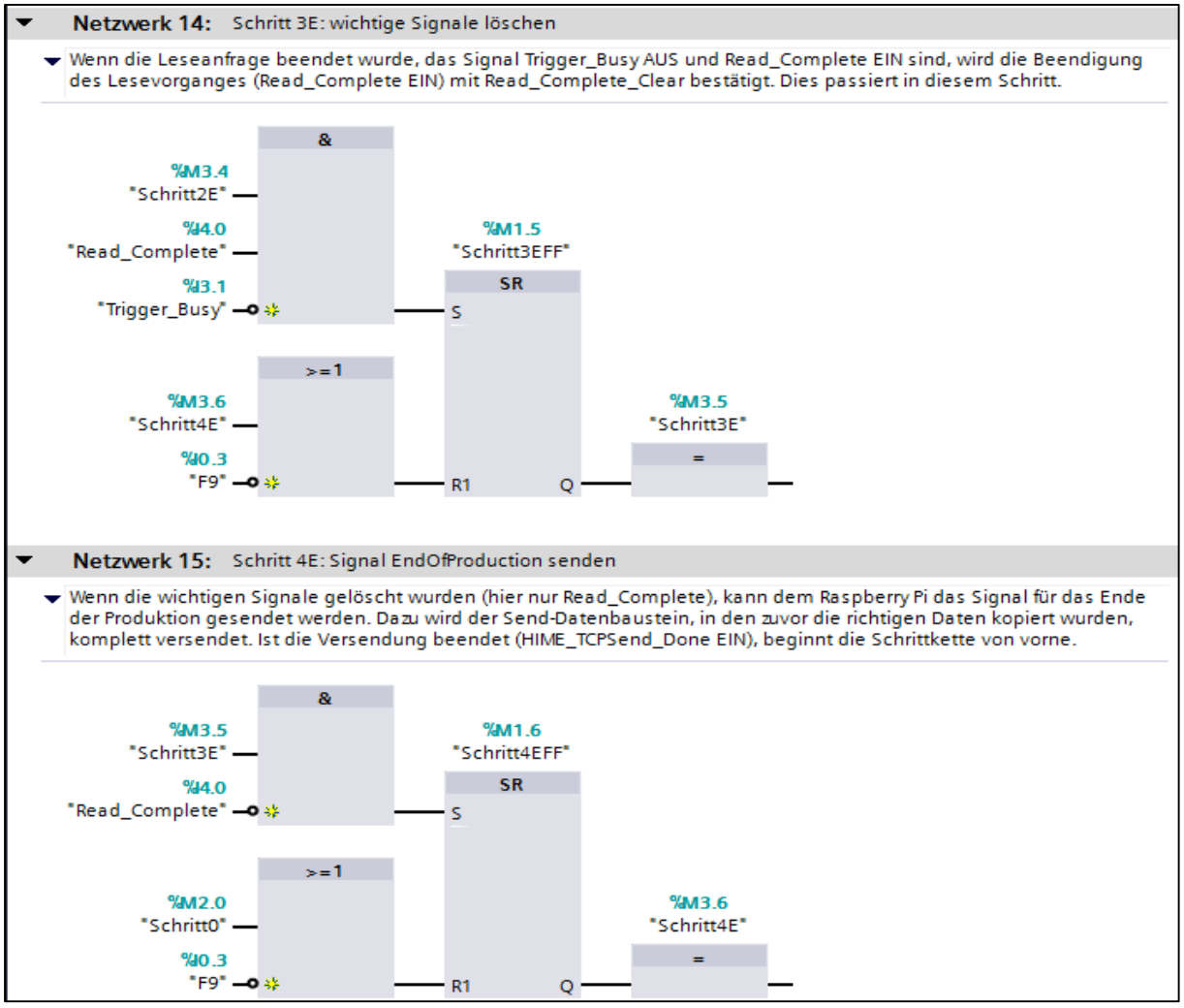


Abbildung 5.14: FC1 Netzwerk 14,15

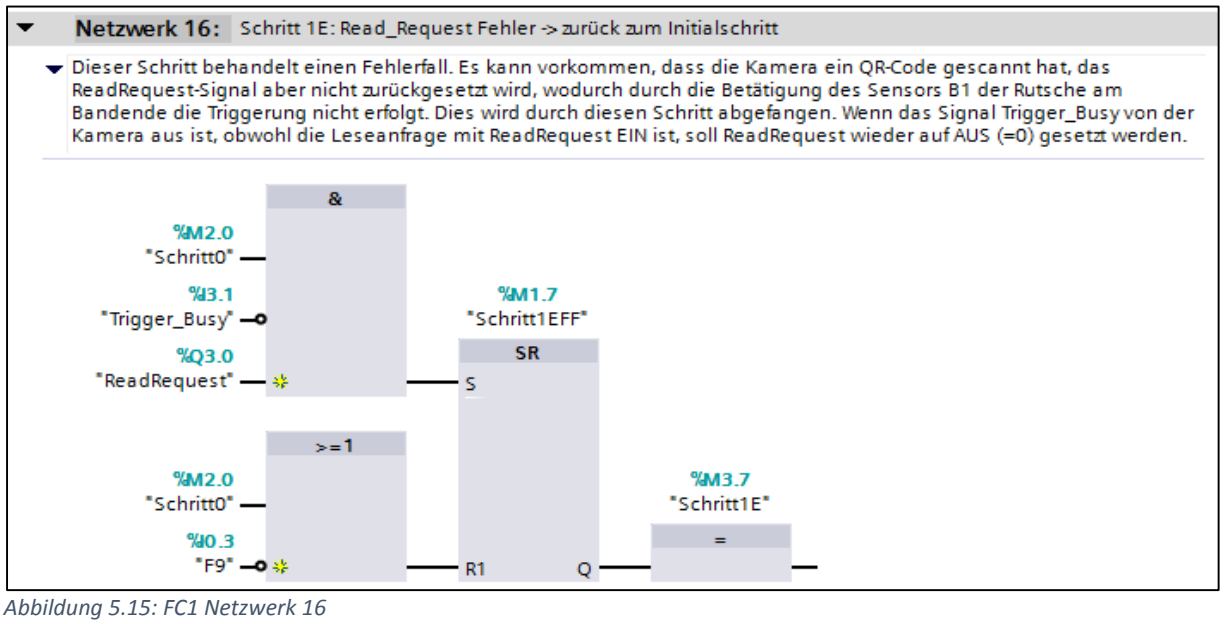


Abbildung 5.15: FC1 Netzwerk 16

5.2.3 FC2 – Kamerasteuerung

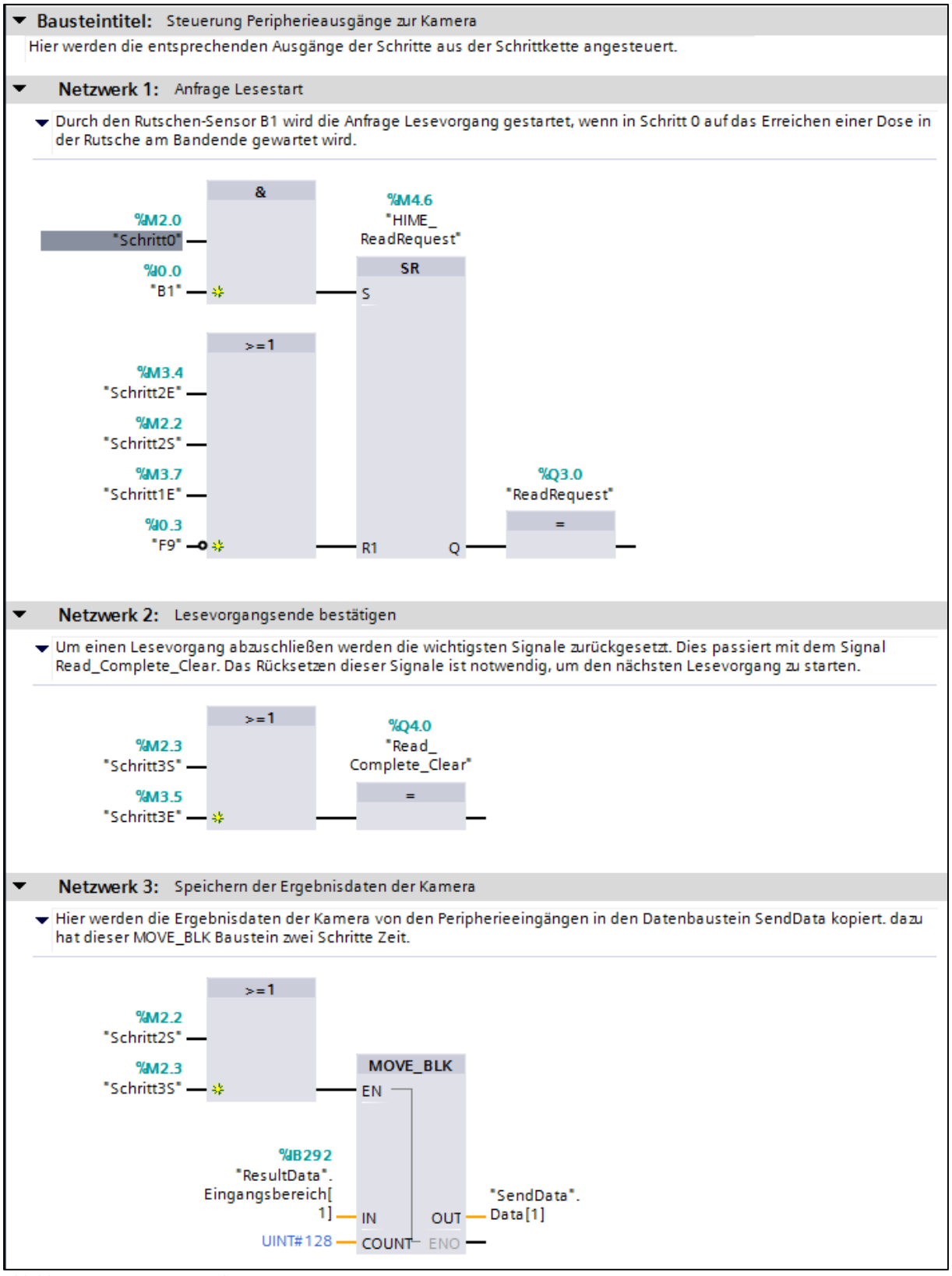


Abbildung 5.16: FC2 Netzwerk 1-3

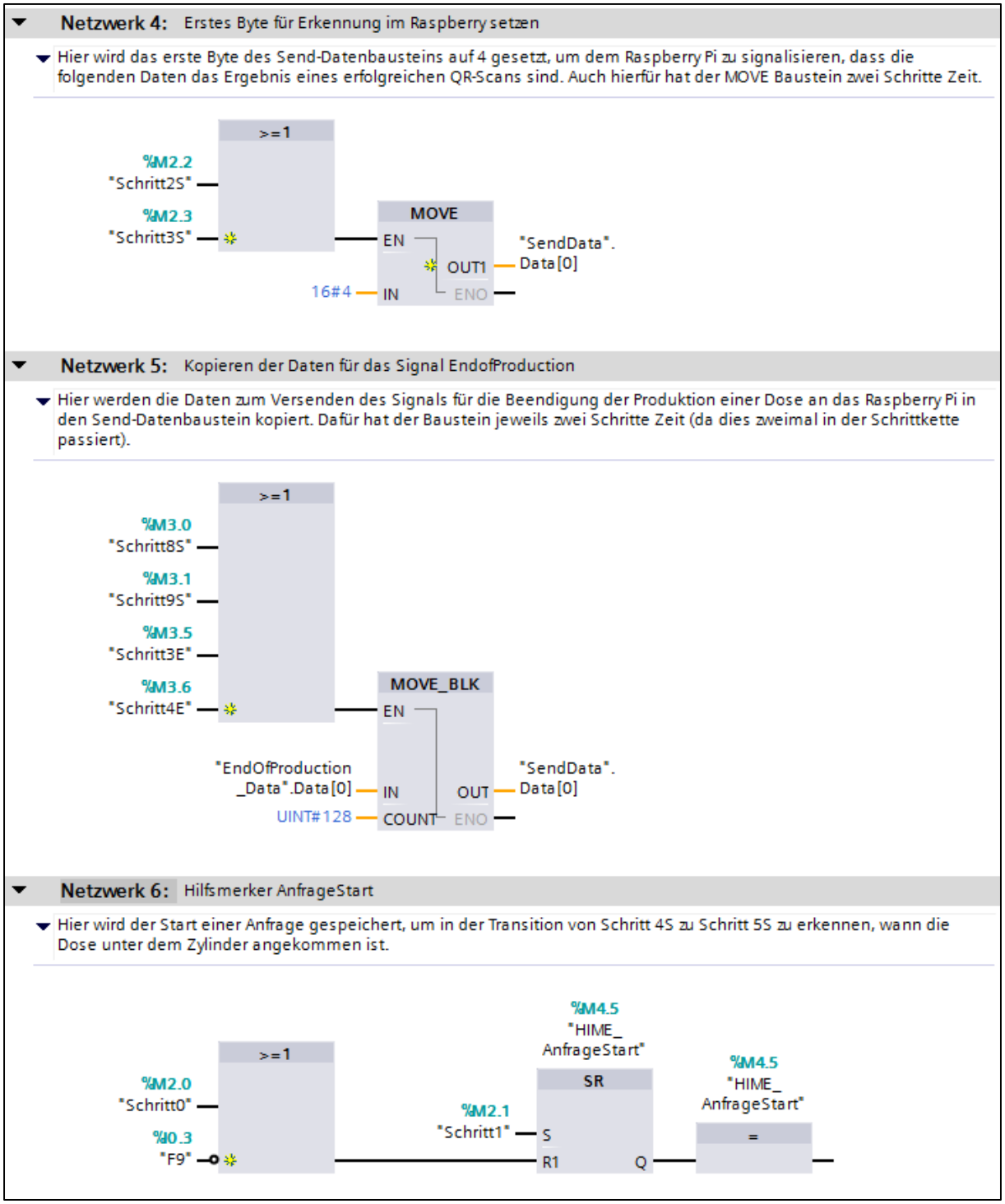


Abbildung 5.17: FC2 Netzwerk 4-6

5.2.4 FC5 – TCP_Kommunikation

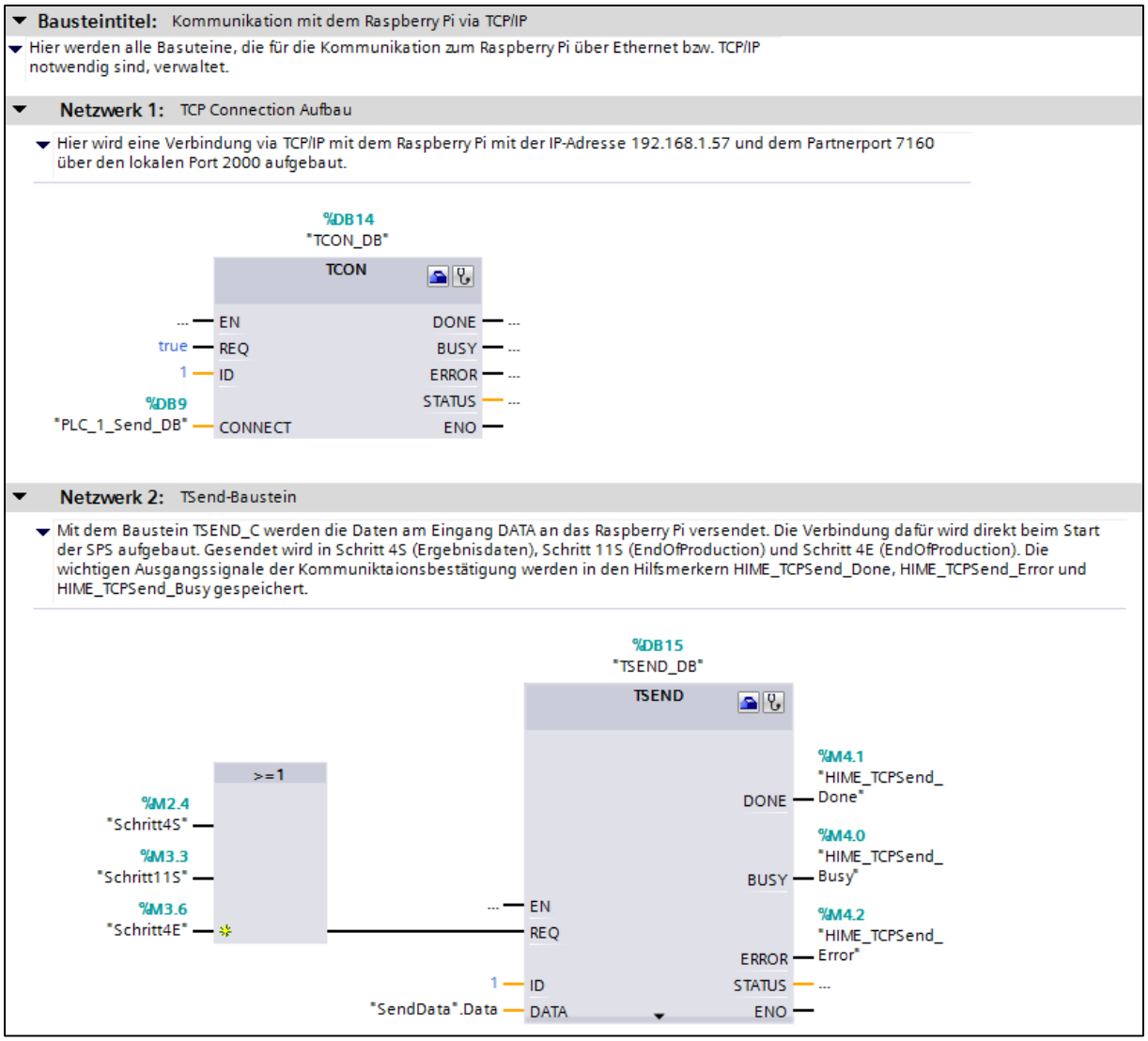


Abbildung 5.18: FC5 Netzwerk 1,2

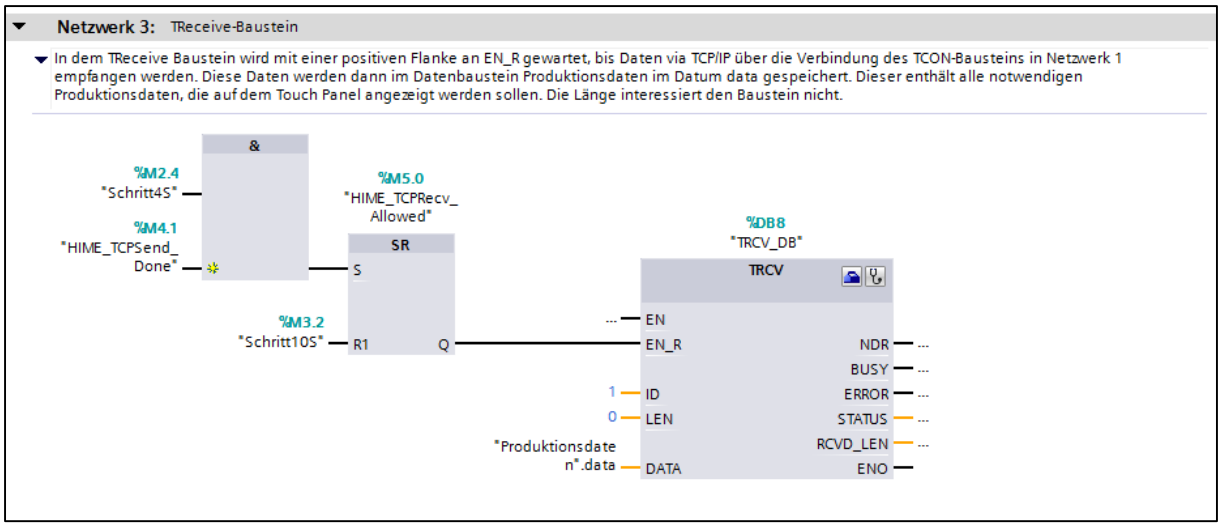


Abbildung 5.19: FC5 Netzwerk 3

5.3 Visualisierung

Abbildung 5.20: Visualisierung auf dem Touch Panel in deutscher Sprache

5.4 Java-Programmcode

History.java

```
 1 package classes;
 2
 3 import java.io.*;
 4 import java.util.Scanner;
 5 import java.util.Calendar;
 6 import java.util.concurrent.TimeUnit;
 7 import javax.mail.MessagingException;
 8
 9
10
11 public class History
12 {
13     // Constants:
14     private final static int PORT=7158; // port of general PLC
15     private final static int PORT_QR = 7160; // port of QR-Code-and-box-turn-PLC
16
17     public static int idshift;
18     public static int automatic_mode;
19     public static double mintemp;
20     public static double maxtemp;
21     public static double minhum;
22     public static double maxhum;
23     public static boolean connection;
24     public static double press = 0.0;
25     public static Server server;
26     public static Server server_QR;
27     public static SQLPHP database;
28     public static Energy_data energy_data = new Energy_data(); // used in Server to save the
   Energy values and in production to write the values in the SQL Database
29     public static GetFill fill = new GetFill();
30     public static int boxFillLevel; // How many boxes are available?
31     public static Production production;
32     public static DataProduction dataproduction = new DataProduction();
33     private static double temp =0.0;
34     private static double hum = 0.0;
35
36
37
38     public static void main(String[] args)
39     {
40         idshift = 0;
41         server = new Server(PORT);
42         //neuen Server eröffnen, um Daten von der S7-1200 Station zu empfangen
43         server_QR = new Server(PORT_QR);
```

Abbildung 5.21: Java-Programmcode des geänderten Teils aus der Main-Klasse History.java, Zeile 1-43

Server.java

```
 1 package classes;
 2
 3 /*  TODO:
 7
 8 import java.io.*;
 9 import java.net.*;
10 import java.util.regex.*;
11
12
13 public class Server extends Thread
14 {
15
16     private Socket client = null;
17     private int port = 0;
18     private BufferedInputStream bufferedInputStream = null;
19     private BufferedOutputStream bufferedOutputStream = null;
20     private byte[] recvArray = new byte[129];
21     private byte[] sendArray = new byte[4];
22     private String data = null;
23     private ServerSocket serverSocket = null;
24
25     public boolean fill_level = true; // used for measuring the fill_level and to send the
   email in history
26
27     // constants to identify different received data
28     private final static int CON_ERROR = 0;
29     private final static int LED_SETTING= 1;
30     private final static int SENSOR_DATA= 2;
31     private final static int ENERGY_DATA= 3;
32     private final static int QR_SCAN = 4;   //received Data are content of the scaned
   QR-Code
33     private final static int END_OF_PRODUCTION = 5; //to get an info about the Production
   status.
34     private final static String PRINTER2_IP_ADDRESS = "192.168.1.77";
35     int i = 0;   // used as index for the communication between the EEM-MA600 and the
   Raspberry
36     int energy = 0; // used as buffer for the energy values
37
38
39     public Server(int port)
58
60     public void run()
98
99     private void read()
120
121    private void processReceivedData()
161
162    /*Diese Methode wird aufgerufen, wenn die empfangenen Daten Daten von dem
   QR-Code-Scanner von der S7-1200-Station
165    private void processQRScan()
271
272    private void processEnergyData()
348
349    private void processSensorData()
372
373    private void processLedSetting()
426
427    private void warteAufAnmeldung() throws IOException
438
439    private void leseNachricht() throws IOException
446
447    public void sendStartProduction() throws IOException
```

Abbildung 5.22: Java-Programmcode, Übersicht der Klasse Server.java, Zeile 1-447

Server.java

```
462
463     private void sendEndOfProduction() throws IOException
480
481     /*Sendet die Produktionsdaten an die S7-1200 Station
487     private void sendProductionData(DataProduction dataProduction)
565
567     public void finalize()
581
582     //Diese Methode konvertiert ein float-Wert in ein Integer-Wert (32 Bit) unter Beachtung
583     //der Norm IEEE 754, die für Gleitkommzahlen die Form der Bits gibt.
584     public byte[] convertFloatToByteArray(float value)
601 }
602
```

Abbildung 5.23: Java-Programmcode, Übersicht der Klasse Server.java, Zeile 447-602

Server.java

```
121     private void processReceivedData()
122     {
123         switch(recvArray[0])
124         {
125
126         case CON_ERROR:
127             if(recvArray[1] == 0 && recvArray[2] == 0 && recvArray[3] == 0)
128             {
129                 System.out.println("Server.java: Error: Received zero. Connection broken?");
130             }
131             break;
132
133         case LED_SETTING:
134             processLedSetting();
135             break;
136
137         case SENSOR_DATA:
138             processSensorData();
139             break;
140
141         case ENERGY_DATA:   // Writes the energy values in the history.energy object
142             processEnergyData();
143             break;
144
145         case QR_SCAN:
146             // Auswertung der von der S7-1200 gesendeten QR-Scan-Ergebnisdaten
147             processQRScan();
148             break;
149
150         case END_OF_PRODUCTION:
151             if (recvArray[1] == 1)
152             {
153                 Production.status = false;
154             }
155             break;
156
157         default:
158
159         }
160     }
```

Abbildung 5.24: Java-Programmcode der Methode processReceivedData() aus der Klasse Server.java

Server.java

```
162     /*Diese Methode wird aufgerufen, wenn die empfangenen Daten Daten von dem QR-Code-Scanner
    von der S7-1200-Station
163     * sind. Das erste Byte ist dann gleich 4.
164     */
165     private void processQRScan()
166     {
167         //Inhalt der übermittelten Daten (idealerweise eine URL) in einem String speichern
168         String qrCodeContent = new String(recvArray, 1, 128); //Empfangene String-Daten
169 
170         //DEBUG
171         System.out.println("----------------------------------------");
172         System.out.println("SPS-Daten von der S7-1200 empfangen: " + qrCodeContent);
173 
174         //Erstellen eines Produktionsdatenobjektes für die spätere Versendung der
    Produktionsdaten
175         DataProduction dataProd;
176 
177         //default Datenbank-ID erstellen
178         int database_ID = -1;
179 
180         try
181         {
182             /*
183              * \\?id= -> ist Startmarke
184              * .*?    -> nicht gieriger Ausdruck, der alles zwischen den Marken liefern
185              * \\?    -> ist Endmarke
186              */
187             Pattern p_string = Pattern.compile("\\?id=.*?\\?"); //Vergleichswert für
    '?id=....?'
188             Pattern p_result = Pattern.compile("\\d.*\\b");     //Vergleichswert für die reine
    ID
189             Pattern p_error = Pattern.compile("ERROR");         //Vergleichswert -> ERROR
190 
191             Matcher m_string = p_string.matcher(qrCodeContent); //Matcher1 erstellen zum
    durchsuchen nach regulären Ausdrücken
192 
193             if(m_string.find())//Vergleich -> String-Teil ?id=....? (s.o.) finden
194             {
195 
196                 /*
197                  * Teil-String ?id=....? gefunden!
198                  * \\d.* -> liefert den String-Teil, in dem nur Zahlen vorkommen und alles
    danach.
199                  *       -> liefert '....?'
200                  * \\b   -> liefert alles bis zu einem Buchstaben
201                  *       -> liefert '....'
202                  */
203                 Matcher m_result = p_result.matcher(m_string.group());  //Matcher erstellen
204 
205                 if(m_result.find()) //Vergleich -> String-Teil finden, der die ID enthält
206                 {
207                     /*
208                      * ID gefunden!
209                      * ID aus dem gefundenen String als Integer speichern.
210                      * -> trim() liefert die ID als String zurück.
211                      * -> in database_ID ist dann die ID als Integer gespeichert.
212                      */
213                     database_ID = Integer.parseInt(m_result.group().trim());
214 
215                     /*
216                      * Wenn die ID erfolgreich ausgelesen wurde, soll der Druckauftrag
217                      * zum zweiten Drucker geschickt werden.
```

Abbildung 5.25: Java-Programmcode der Methode "processQRScan()" der Klasse Server.java, Zeile 162-217

Server.java

```
218                      * Ansonsten soll nichts geschehen...
219                      */
220                     if(database_ID > 0)
221                     {
222                         try
223                         {
224                             //Versenden des Druckauftrages
225                             new SidePrinter(Production.fullName, PRINTER2_IP_ADDRESS);
226                         }
227                         catch (NullPointerException ex)
228                         {
229                             ex.printStackTrace();
230                         }
231                     }
232 
233                     //Einlesen der Produktionsdaten mit der gefilterten ID aus der Datenbank
234                     //Abspeicherung der Produktionsdaten in dem Produktionsdatenobjekt
235                     dataProd = History.database.readProductionData(database_ID);
236 
237                     //DEBUG
238                     System.out.println("Datenbankabfrage fertig, Produktionsdaten
    empfangen:");
239                     System.out.println(dataProd.toString());
240 
241                     //VErsendung der Produktionsdaten
242                     sendProductionData(dataProd);
243                 }
244                 else
245                 {
246                     //keine ID gefunden
247                     System.err.println("QR-Read: Keine ID gefunden.");
248                     return; //Abbruch!
249                 }
250             }
251             else
252             {
253                 /*
254                  * Ausdruck "ERROR" mit dem String vergleichen. Wenn er gleich ist,
255                  * gab es einen Fehler beim Auslesen des QR-Codes. D.h. die Zeit des Scans
256                  * hat nicht ausgereicht, um den QR-Code auf dem Deckel zu lesen.
257                  */
258                 Matcher m_error = p_error.matcher(qrCodeContent);   //Matcher erstellen
259                 if(m_error.find())  //Nur, wenn der gesamte String nur aus ERROR besteht...
260                 {
261                     System.err.println("QR-Read: ERROR - Kein Code gelesen.");
262                     return; //Abbruch!
263                 }
264             }
265         }
266         catch (IllegalStateException ex) //falls keine ID gefunden wurde
267         {
268             System.err.println("QR-Read: Exception geworfen! Fehler beim Lesen aus dem
    String.");
269         }
270     }
```

Abbildung 5.26: Teil 2 des Java-Programmcodes der Methode "processQRScan()" der Klasse Server.java, Zeile 218-270

Server.java

```
481     /*Sendet die Produktionsdaten an die S7-1200 Station
482     * Daten: Produktions- und Ablaufdatum, Uhrzeit, Luftdruck, Luftfeuchtigkeit,
483     *           Temperatur, Anzahl der M&Ms, verbrauchte Energie
484     * Versendung per TCP/IP
485     * 26 Bytes werden in dem entsprechenden Format für die SPS versendet
486     */
487     private void sendProductionData(DataProduction dataProduction)
488     {
489         //Erstellung eines Byte-Arrays zum Versenden der Produktionsdaten zur SPS
490         byte[] sendArrayS7 = new byte[26];
491 
492         //Erstellung eines 4 Byte Arrays für die Gleitkommazahlen
493         byte[] output = new byte[4];
494 
495         try
496         {
497             //Berechnung der Tagen seit 1-1-1990
498             char dateInDays = dataProduction.getDateInDays();
499             //Datum in Tagen seit 1-1-1990
500             sendArrayS7[0] = (byte) (dateInDays >> 8);
501             sendArrayS7[1] = (byte) (dateInDays);
502 
503             //Berechnung der Tagesuhrzeit in Millisekunden
504             int timeOfDay = dataProduction.getTimeOfDay();
505             //Uhrzeit in Millisekunden seit 00:00
506             sendArrayS7[2] = (byte) (timeOfDay >> 24);
507             sendArrayS7[3] = (byte) (timeOfDay >> 16);
508             sendArrayS7[4] = (byte) (timeOfDay >> 8);
509             sendArrayS7[5] = (byte) timeOfDay;
510 
511             //Anzahl MundMs
512             sendArrayS7[6] = (byte) dataProduction.amountMuM;
513             //Dummy
514             sendArrayS7[7] = (byte) 0x00;
515 
516             //Berechnung der zu versendenden Daten nach IEE754
517             output = convertFloatToByteArray((float) dataProduction.temp);
518             //Temperatur
519             sendArrayS7[8] = output[0];
520             sendArrayS7[9] = output[1];
521             sendArrayS7[10] = output[2];
522             sendArrayS7[11] = output[3];
523 
524             //Berechnung der zu versendenden Daten nach IEE754
525             output = convertFloatToByteArray((float) dataProduction.press);
526             //Luftdruck
527             sendArrayS7[12] = output[0];
528             sendArrayS7[13] = output[1];
529             sendArrayS7[14] = output[2];
530             sendArrayS7[15] = output[3];
531 
532             //Berechnung der zu versendenden Daten nach IEE754
533             output = convertFloatToByteArray((float) dataProduction.hum);
534             //Luftfeuchtigkeit
535             sendArrayS7[16] = output[0];
536             sendArrayS7[17] = output[1];
537             sendArrayS7[18] = output[2];
538             sendArrayS7[19] = output[3];
539 
540             //Berechnung der zu versendenden Daten nach IEE754
541             output = convertFloatToByteArray((float) dataProduction.work);
542             //Energie
```

Abbildung 5.27: Java-Programmcode der Methode "sendProductionData()" der Klasse Server.java, Zeile 481-542

Server.java

```
543             sendArrayS7[20] = output[0];
544             sendArrayS7[21] = output[1];
545             sendArrayS7[22] = output[2];
546             sendArrayS7[23] = output[3];
547
548             //Ablaufdatum
549             sendArrayS7[24] = (byte) 0x00;
550             sendArrayS7[25] = (byte) 0x00;
551
552             //Versendung des Byte-Arrays
553             bufferedOutputStream.write(sendArrayS7, 0, 26);
554             //Sicherung, dass auch alle Bytes gesendet wurden
555             bufferedOutputStream.flush();
556
557             System.out.println("Produktionsdaten an die S7-1200 übermittelt.");
558             System.out.println("-----------------------------------");
559         }
560         catch (IOException ex)
561         {
562             ex.printStackTrace();
563         }
564     }
```

Abbildung 5.28: Teil 2 des Java-Programmcodes der Methode "sendProductionData()" der Klasse Server.java, Zeile 543-564

Server.java

```
582     //Diese Methode konvertiert ein float-Wert in ein Integer-Wert (32 Bit) unter Beachtung
583     //der Norm IEEE 754, die für Gleitkommzahlen die Form der Bits gibt.
584     public byte[] convertFloatToByteArray(float value)
585     {
586         //Rückgabe Array erstellen
587         byte[] ret = new byte[4];
588
589         //Integer-Wert erstellen, der die gemäß der IEEE754 umgewandelte Gleitkommazahl
    beinhaltet.
590         int bits = Float.floatToIntBits((float) value);
591
592         //abspeichern der einzelnen Bytes des Integer-Werts in dem Rückgabe-Array
593         for (int i = 0; i < 4; i++)
594         {
595             //Verschieben der Bits des Integer-Werts, um das entsprechende Byte im Array zu
    speichern.
596             ret[i] = (byte) (bits >> (3-i) * 8);
597         }
598         return ret;
599     }
```

Abbildung 5.29: Java-Programmcode der Methode convertFloatToByteArray() der Klasse Server.java

DataProduction.java

```
 1 package classes;
 2
 3 import java.util.Date;
 4 import java.util.Calendar;
 5
 6 //this class is used to save data from the table 'productionData'
 7
 8 public class DataProduction
 9 {
10     public int id;
11     public Date time;
12     public int amountMuM;
13     public double temp;
14     public double hum;
15     public double press;
16     public double work;
17     private Calendar calendar = Calendar.getInstance();
18
19     public Boolean stop = false;
20     public Boolean monitor = false;
21
22
23     public Boolean end_of_production(Boolean status)
24     {
25         if(status == false && monitor == false)
26         {
27             stop = true;
28             monitor = true;
29         }
30         else
31         {
32             stop = false;
33         }
34
35         if(status == false)
36            monitor = false;
37
38         return stop;
39     }
40
41     /*Funktion, um das Produktionsdatum per String zu setzen.
42     * Das Format des Strings muss wie folgt gegeben sein:
43     * yyyy-mm-dd hh:mm:ss
44     **/
45     public void setProductionDate(String date)
46     {
47         int year = Integer.parseInt(date.substring(0, 4));      //Variable, die das Jahr
   aus dem String enthält.
48         int month = Integer.parseInt(date.substring(5, 7));     //Variable, die den Monat
   aus dem String enthält.
49         int day = Integer.parseInt(date.substring(8, 10));      //Variable, die den Tag aus
   dem String enthält.
50         int hour = Integer.parseInt(date.substring(11, 13));    //Variable, die die Stunden
   des Tages aus dem String enthält.
51         int minute = Integer.parseInt(date.substring(14, 16)); //Variable, die die Minuten
   aus dem String enthält.
52         int second = Integer.parseInt(date.substring(17));      //Variable, die die
   Sekunden aus dem String enthält.
53
54         Calendar calendar = Calendar.getInstance();             //Kalender erzeugen, in dem
   per Integer-Werten eine UTC-Zeit erstellt wird.
55         calendar.set(year, month, day, hour, minute, second);   //erstellen dieser UTC-Zeit
```

Abbildung 5.30: Java-Programmcode der Klasse DataPoduction.java, Zeile 1-55

DataProduction.java

```
56
57          //abspeichern der Produktionszeit mit Datum in der time Variable mit Übergabe der
    Zeit in
58          //Millisekunden seit 1970-1-1 00:00:00:00
59          time = new Date(calendar.getTimeInMillis());
60      }
61
62      //Diese Methode gibt das Produktionsdatum in Tagen seit 1-1-1990 als character zurück.
63      //Benutzt wird dabei die Variable time aus diesem Objekt
64      public char getDateInDays()
65      {
66          //Erstellung eines Standard-Kalenders, der das Referenzdatum 1-1-1990 enthält.
67          //Damit wird die Entfernung des Produktionsdatums zu dem Referenzdatum berechnet.
68          calendar.set(1990, 1, 1);
69
70          //Berechnung der Millisekunden von dem Produktionsdatum zum Referenzdatum 1-1-1990
71          long dateInMillis = time.getTime() - calendar.getTimeInMillis();
72
73          //Umwandlung des Datums inklusive Tageszeit in Millisekunden in ein Datum in Tagen
    seit 1990
74          char dateInDays = (char) (dateInMillis/1000/60/60/24);
75
76          return dateInDays;
77      }
78
79      //Diese Methode gibt den Tagesuzeitpunk der Produktion in Millisekunden seit 00:00 als
    Integer zurück.
80      public int getTimeOfDay()
81      {
82          //Erstellung eines Kalenders mit dem Produktionsdatum
83          calendar.setTime(time);
84
85          //Auslesen der Stunden, Minuten und Sekunden des Tages zur Versendung an die S7-
    1200
86          int hour = calendar.get(Calendar.HOUR_OF_DAY);  //Variable für die Stunde am
    Produktionstag
87          int minute = calendar.get(Calendar.MINUTE);     //Variable für die Minute in der
    Stunde des Produktionstages
88          int second = calendar.get(Calendar.SECOND);     //Variable für die Sekunde in der
    Minute der Stunde des Produktionstages
89
90          //Berechnung der Tagesuhrzeit, wann die Dose produziert wurde in Millisekunden
91          int timeOfDay = ((hour * 60 + minute) * 60 + second) * 1000;
92
93          return timeOfDay;
94      }
95
96      public String toString()
97      {
98          String ret = "";
99          ret = "ID = " + id + "\n";
100         ret += "Produktionsdatum: " + time.toString() + "\n";
101         ret += "Anzahl M&Ms: " + amountMuM + "\n";
102         ret += "Temperatur: " + temp + "\n";
103         ret += "Luftfeuchtigkeit: " + hum + "\n";
104         ret += "Pneumatikdruck: " + press + "\n";
105         ret += "Energie: " + work + "\n";
106         return ret;
107     }
108 }
```

Abbildung 5.31: Java-Programmcode der Klasse DataProduction.java, Zeile 56-108

SQLPHP.java

```
64      /*
65       * Diese Methode liest einen kompletten Datensatz von Produktionsdaten einer Dose
66       * aus der Datenbank, indem es eine Anfrage über das Internet an den Server
67       * stellt, das PHP Skript get_ProductionData.php auszuführen. Dieses bekommt
68       * den Schlüssel ID übergeben und liefert den Datensatz als String zurück.
69       * Die Kommunikation zwischen dem Server und dem Raspberry ist über TLS
70       * verschlüsselt (sprich HTTPS).
71       */
72      public DataProduction readProductionData(int id)
73      {
74          //Eine Return-Variable erzeugen.
75          DataProduction ret = new DataProduction();
76          try
77          {
78              //String bilden, der für die Authentifikation zuständig ist...
79              String authString = name + ":" + password;
80              byte[] authEncBytes = Base64.encodeBase64(authString.getBytes());
81              String authStringEnc = new String(authEncBytes);
82
83              //Anzuhängende Daten zur Übergabe der ID an das PHP-Skript
84              String data;
85              data = "?id=" + URLEncoder.encode("" + id, "UTF-8");
86
87              //URL erzeugen, um das PHP-Skript mit der id angehängt aufzurufen
88              URL url = new URL(url_database + "get_ProductionData.php" + data);
89
90              //Verbindung eröffnen
91              HttpURLConnection con = (HttpURLConnection) url.openConnection();
92
93              //Authentifikation durchführen
94              con.setRequestProperty("Authorization", "Basic " + authStringEnc);
95
96              //10s Timeout, falls keine Antwort vom Server
97              con.setConnectTimeout(10000);
98
99              //BufferedInputReader erzeugen, um Daten, die das PHP-Skript übermittelt zu
    empfangen
100             in = new BufferedReader(new InputStreamReader(con.getInputStream()));
101
102             //Input-String zum Einlesen von Daten
103             String input = in.readLine();
104
105             if(input !=null) //Wenn das readLine() geklappt hat
106             {
107                 //JSONObject erzeugen, um die Empfangenen Daten auszuwerten
108                 JSONObject jsonObject = new JSONObject(input);
109
110                 //Alle gelesenen Daten in den entsprechenden Variablen speicher
111                 ret.setProductionDate(jsonObject.getString("time"));
112                 ret.amountMuM = Integer.parseInt(jsonObject.getString("amountMuM"));
113                 ret.temp = Double.parseDouble(jsonObject.getString("temperature"));
114                 ret.press = Double.parseDouble(jsonObject.getString("pressure"));
115                 ret.hum = Double.parseDouble(jsonObject.getString("humidity"));
116                 ret.work = Double.parseDouble(jsonObject.getString("p1"));
117             }
118         }
119         catch (IOException e)
120         {
121             ret.setProductionDate("1990-01-01 00:00:00");
122             ret.amountMuM = 0;
123             ret.temp = 0.0;
124             ret.press = 0.0;
```

Abbildung 5.32: Java-Programmcode der Methode "readProuctionData()" der Klasse SQLPHP.java, Zeile 64-124

SQLPHP.java

```
125             ret.hum = 0.0;
126             ret.work = 0.0;
127             e.printStackTrace();
128             return ret;
129         }
130         catch ( JSONException e)
131         {
132             System.out.println("Fehler beim lesen aus dem JSONObject");
133 
134             ret.setProductionDate("1990-01-01 00:00:00");
135             ret.amountMuM = 0;
136             ret.temp = 0.0;
137             ret.press = 0.0;
138             ret.hum = 0.0;
139             ret.work = 0.0;
140             e.printStackTrace();
141             return ret;
142         }
143         return ret;
144     }
```

Abbildung 5.33: Teil 2 des Java-Programmcodes der Methode "readProductionData()" der Klasse SQLPHP.java, Zeile 125-144

5.5 PHP-Skript

```php
<?php

/*
 * Following code will check if the given id was already used
 */

// array for JSON response
$response = array();

// ../include db connect class
require_once __DIR__ . '/../include/pdo_factory.php';

// connecting to db
try
{
    $pdo = PDO_FACTORY::createPDO();
    //$pdo = new
    PDO('mysql:host='.$dbconfig['host'].';dbname='.$dbconfig['base'].';charset='.$dbconfig['ch
    ar'].';', $dbconfig['user'], $dbconfig['pass']);

}
catch(PDOException $e)
{
    exit('Unable to connect to Database.');
}
if (isset($_GET["id"]) ) {

    $id = $_GET['id'];

    try
    {
        $statement = $pdo->prepare("SELECT * FROM productiondata WHERE id = '$id'");
        $statement->execute();
        $result = $statement->fetch();
    } catch(PDOException $e)
    {
        exit('Unable to connect to Database.');
    }

    if (!empty($result)) {
        $response["success"] = 1;
        $response["time"] = $result["productionTime"];
        $response["amountMuM"] = $result["amount"];
        $response["temperature"] = $result["temperature"];
        $response["pressure"] = $result["pressure"];
        $response["humidity"] = $result["humidity"];
        $response["p1"] = $result["p1"];
        //echoing JSON data
        echo json_encode($response);
    } else {
        //no data found
        $response["success"] = 0;
        $response["message"] = "No data found2";
        //echoing JSON data
        echo json_encode($response);
    }
}
?>
```

Abbildung 5.34: PHP Skript für die Abfrage der Produktionsdaten von der Datenbank

6 Anhang zur Auftragskontrolle

6.1 Abnahmemessprotokoll

Abnahmemessprotokoll (Blatt 1)
Auswahl nach DIN VDE 0100 Teil 600

Prüfer: Simon Häußler **Datum:** 24.05.2016

Anlage: M&M-Abfüllanlage		**Anlagennummer:**	192.168.1.59
Typ:	**Industrie 4.0 - Demonstrationsanlage**	**Hersteller:**	BBS2 Wolfsburg
Netzspannung:	230/400 V 50 Hz	**Leistung:** 100 W	**Baujahr:** 2016

Grund der Prüfung:
☒ Erstprüfung ☐ Wiederholungsprüfung ☐ Änderung ☐ Instandsetzung

1. Sichtkontrolle
Achtung: Die Sichtkontrolle ist im spannungslosen Zustand durchzuführen.

		Mängel vorhanden		
Nr.	**Sichtkontrolle**	**Ja**	**Nein**	**Anmerkung**
1	Richtige und saubere Bezeichnung aller Bauteile		X	
2	Richtige Auswahl der Betriebsmittel (IP, Spannung, Anschlüsse,...)		X	
3	Fester Sitz aller Bauteile und Betriebsmittel (insb. Codeleser mit Halterung)		X	
4	Keine Beschädigungen an Bauteilen		X	
5	Keine Beschädigungen an Leitungen		X	
6	Richtige Leitungsauswahl (Farbe, Querschnitt, Flexibilität)		X	
7	Anschluss der Leitungen		X	
8	Richtiger Anschluss der Leitungen		X	
9	Saubere Leitungsverlegung		X	
10	Zuleitung und Steckverbinder ohne Beschädigung		X	
11	Berührungsschutz gegeben		X	
12	Beschädigungsfreie Pneumatikleitungen		X	
13	Beschädigungsfreie Pneumatikbauteile		X	
14	Zugfestigkeit an Klemmstellen		X	
15	Sauberes Sichtfenster des Codelesers		X	
16				
17				
18				
19				
20				
21				

Blatt 1

Abbildung 6.1: Abnahmemessprotokoll Seite 1(5), Sichtkontrolle

2. Messungen (Blatt 2)

Sichtkontrolle in Ordnung (Blatt 1) i.O. ☒

	Messungen	Mess-wert	geeigneter Wert	in Ordnung	nicht in Ordnung
	Schutzleitersystemprüfung (Schutzleiterdurchgang R_{SL} (R_{LO}))				
1	PE-Klemme > Einspeisung (CEE-Stecker)	0,02 Ω	0,23 Ω	☒	☐
2	PE-Klemme > Schaltschrank	0 Ω	0,2 Ω	☒	☐
3	PE-Klemme > Montageplatte	0 Ω	1,25 Ω	☒	☐
4	PE-Klemme > Schaltschranktür/Gestell	0 Ω	0,4 Ω	☒	☐
5	PE-Klemme > Schaltkasten -U2	0 Ω	1,4 Ω	☒	☐
6	PE-Klemme > Schaltkasten -U3	0 Ω	1 Ω	☒	☐
7	PE-Klemme > Phoenix Contact SPS	0 Ω	1,2 Ω	☒	☐
8	PE-Klemme > Siemens SPS	0 Ω	1,8 Ω	☒	☐
9	PE-Klemme > Antrieb	0 Ω	1,04 Ω	☒	☐
	Berechnung des geeigneten Wertes des Schutzleiterdurchgangs:				
	Wahl eines Übergangswiderstandes für Klemmstellen: 0,2 Ω Widerstandsbeläge: $R'_{1,5}$ = 12,58 mΩ/m $R'_{2,5}$ = 7,57 mΩ/m Berechnung: $R_{LO} = n * 0,2\ \Omega + l_{1,5} * R'_{1,5} + l_{2,5} * R'_{2,5}$	**n: Anzahl Klemmstellen** **$l_{1,5}$: Leitungslänge 1,5 mm²** **$l_{2,5}$: Leitungslänge 2,5 mm²**			
	Isolationsmessung R_{ISO}				
10	Isolationsprüfung ($U_{prüf}$ = 500 VDC) (L1 - PE)	>500 MΩ	≥ 1 MΩ	☒	☐
11	Isolationsprüfung ($U_{prüf}$ = 500 VDC) (L2 - PE)	>500 MΩ	≥ 1 MΩ	☒	☐
12	Isolationsprüfung ($U_{prüf}$ = 500 VDC) (L3 - PE)	>500 MΩ	≥ 1 MΩ	☒	☐
13	Isolationsprüfung ($U_{prüf}$ = 500 VDC) (N - PE)	>500 MΩ	≥ 1 MΩ	☒	☐
14	Isolationsprüfung FELV ($U_{prüf}$ = 500 VDC) (L1/L2/L3 - [+24V])	>500 MΩ	≥ 1 MΩ	☒	☐
15	Isolationsprüfung FELV ($U_{prüf}$ = 500 VDC) (PE - [+24V])	—	≥ 1 MΩ	☐	☐
	Schleifenimpedanzmessung Z_S				
16	Schleifenimpedanz Z_{Schl} Schaltschrank, Messort:	1,18 Ω	≤ 1,9 Ω	☒	☐
17				☐	☐
	Berechnung des/der geeigneten Werte(s) der Schleifenimpedanz: U_0: Außenleiterspannung $Z_{Schl} = U_0/I_a * 2/3$ I_a: Abschaltstrom des Leitungsschutzes I_a = charakteristik * I_{Nenn} Z_{Schl} = 230 V/(5 * 16 A) * 2/3 = 1,9 Ω				
18	Restspannung (nach 1 s < 60 V bei Maschinen mit Steckvorrichtungen)	–	< 60 V	☐	☐

1) Laut Norm muss der gemessene Widerstand in dem Bereich liegen, der entsprechend dem Querschnitt, der Länge und dem Material des Schutzleiters zu erwarten ist. Die Bestimmung des Widerstands erfolgt durch Berechnung.

weiter auf Blatt 3

Blatt 2

Abbildung 6.2: Abnahmemessprotokoll Seite 2(5), Messungen

weiter Messen (Blatt 3)

RCD-Messung: erforderlich ☒ nicht erforderlich ☐		Mess-wert	Grenz-werte	in Ordnung	nicht in Ordnung
19	Berührungsspannung U_L	1 VAC	≤ 50 V AC	☒	☐
20	Auslösestrom I_F	24mA	≤ 30 mA	☒	☐
21	Auslösezeit t_a	75ms	≤ 200 ms	☒	☐
22	RCD löst aus:			ja ☒	nein ☐

weitere Messungen					in Ordnung	nicht in Ordnung
23	Spannungsmessung	Leiterspannung	Strangspannung	Kleinspannung	☒	☐
		406,8 V	235,6 V	23,9 V		
24	Polarität der Kleinspannung				☒	☐
25	Drehsinnprüfung				☒	☐

3. Erproben

Messungen in Ordnung
☒ i.O. - wenn i.O., dann darf mit der Erprobung fortgefahren werden.

Erproben		in Ordnung	nicht in Ordnung
E1	Wirksamkeit der Schutzeinrichtungen: Not-Halt und Quittierung	☒	☐
E2	Motor dreht in der richtigen Richtung, Anlage arbeitet ordnungsgemäß (s. Funktionsprfg.)	☒	☐
E3	Funktionsfähigkeit von Anzeigeeinrichtungen (Leuchtmelder am Bedienfeld)	☒	☐

4. Funktionskontrolle

Funktionsprüfung der Anlage nach Blatt 4		in Ordnung	nicht in Ordnung
F1	Entspricht den Vorgaben/Teilfunktionen laut folgender Seite	☒	☐

Verwendete Messgeräte	
Hersteller	**Typ**
Fluke	1653 Multifunctiontester
Peak Tech	Multimeter 3335 DMM

Anmerkungen:

Unterschriften

Wolfsburg, 24.05.2016, Simon Häußler
Ort, Datum, Prüfer/in

Blatt 3

Wolfsburg, 24.05.2016,
Ort, Datum, Unternehmer/in

Abbildung 6.3: Abnahmemessprotokoll Seite 3(5), Messungen und Erprobung

4. Funktionskontrolle (Blatt 4)

Prüfer: Simon Häußler **Datum:** 24.05.2016

Anlage: M&M-Abfüllanlage

Achtung: Sofern bei Sichtkontrolle, Messungen und Erprobung keine Mängel festgestellt wurden, erfolgt die Funktionskontrolle. Die Funktionskontrolle erfolgt unter Spannung! Die Funktionskontrolle ist Bestandteil des Abnahmeprotokolls.

		Mängel vorhanden		
Nr.	**Funktionskontrolle**	**Ja**	**Nein**	**Anmerkung**
Anmerkungen: Einige mit * gekennzeichnete Punkte werden über Konsolenausgaben des Java Programmes auf dem Raspberry Pi bestätigt.				
1	Einschalten der Anlage über den Hauptschalter -Q0 und den Drucktaster -S1		X	
2	Triggerung des Codelesers SR-1000 durch die Betätigung des Reflexlichttasters -B1 der Rutsche am Bandende		X	
3	Scan des QR-Codes auf dem Dosendeckel und Einlesen der Ergebnisdaten in den SPS-Datenbaustein		X	
* 4 *	Versendung der Ergebnisdaten das Raspberry Pi via TCP/IP		X	
5	Drehung (Endschritt) nur bei erfolgreichem Lesevorgang		X	
* 6 *	Lesen der Produktionsdaten aus der Datenbank anhand der ID des Leseergbnisses		X	
* 7 *	Versendung der Ergebnisdaten via TCP/IP an die S7-1200 SPS		X	
8	Abspeicherung der aktuellen Produktionsdaten im Datenbaustein der SPS		X	
9	Anzeige der Produktionsdaten der aktuell produzierten Dose auf dem HMI KTP600 Basic color PN		X	
10	Not-Aus Funktion gegeben - Abschaltung der wichtigsten Betriebsmittel im Fehlerfall		X	
11	Kein Start des Endprozesses möglich, wenn Not-Aus betätigt		X	
12	Ausschalten der gesamten Anlage über den Drucktaster -S0 oder -Q0		X	
13	Bisherige Anlagenfunktion vor der Erweiterung		X	
14				
15				

Unterschrift Prüfer/in

Blatt 4

Abbildung 6.4: Abnahmemessprotokoll Seite 4(5), Funktionskontrolle

5. Dokumentation (Blatt 5)

Prüfer: Simon Häußler **Datum:** 24.05.2016

Anlage: M&M Abfüllanlage

		Mängel vorhanden		
Nr.	Dokumentation	Ja	Nein	Anmerkung
1	**Schaltplan der Erweiterung**		X	
2	Erweiterung Netzwerkkompnenten-Plan		X	
3	Erweiterte Bauteile gekennzeichnet		X	
4	Kabeldefintionen, Querschnitte und Adernfarben eingetragen		X	
5	Klemmstellen		X	
6	Klemmleisten		X	
7				
8	**Lastenheft**		X	
9	**Technische Daten SPS, QR-Code-Scanner und Touch Panel**		X	
10	**Betriebsanweisung und Laborordnung**		X	
11	**Java-Programmcode kommentiert**		X	
12	**SPS-Programm kommentiert** (Autor, Datum, OBs, FCs, Netzwerke …)		X	
13	**SPS-Programm und Schaltplan digital abgelegt**		X	
14	**Schrittkette SPS-Programm**		X	
15	**Stückliste**		X	
16	**Komplette Dokumentation als eine PDF-Datei mit allen Dokumenten: Projektdoku, Schaltplan, SPS-Programm, eingescanntes Messprotokoll, Zwischenberichte…**		X	
17				
18				
19				
20				

S. Häußler
Unterschrift Prüfer/in

Blatt 5

Abbildung 6.5: Abnahmemessprotokoll Seite 5(5), Dokumentationskontrolle

7 Auszüge aus Datenblättern

7.1 SR-1000 2D-Codeleser

Datenblatt KEYENCE

SR-1000

Autofokus-Codeleser

TECHNISCHE DATEN

Modell			SR-1000
Typ			Standardmodell
Empfänger	Sensor		CMOS-Bildempfänger
	Pixelanzahl		1280 × 1024 Pixel
Lichtquelle	Beleuchtung		Rote LED mit hoher Leuchtkraft
	Pointer		Grüne LED mit hoher Leuchtkraft
Fokuseinstellung			Autofokus*1
Technische Daten der Ein-und Ausgänge	Steuereingang	Anzahl	2
E/A		Eingangstyp	Bidirektionaler Spannungseingang
		Maximale Spannung	26,4 VDC
Technische Daten der Ein-und Ausgänge		Minimale Einschaltspannung	15 VDC
Technische Daten der Ein-und Ausgänge		Max. Ausschaltstrom	0,2 mA oder weniger
Technische Daten der Ein-und Ausgänge	Steuerausgang	Anzahl	3
Technische Daten der Ein-und Ausgänge		Ausgangstyp	MOS-Fotorelaisausgang
Technische Daten der Ein-und Ausgänge		Maximale Nennwerte	30 VDC
Technische Daten der Ein-und Ausgänge		Maximaler Laststrom	1 Ausgang : 50 mA oder weniger, 3 Ausgänge : 100 mA oder weniger
E/A		Leckstrom, wenn AUS	0,1 mA oder weniger
		Restspannung, wenn EIN	1 V oder weniger
Technische Daten der Ein-und Ausgänge	Ethernet	Kommunikationsstandard	IEEE-802.3-konform, 10BASE-T/100BASE-TX
		Unterstütztes Protokoll	TCP/IP, SNTP, FTP, BOOTP, MC Protocol, Omron SPS link, KV STUDIO, EtherNet/IP™, PROFINET
Technische Daten der Ein-und Ausgänge	Serielle Kommunikation	Kommunikationsstandard	RS-232C
Technische Daten der Ein-und Ausgänge	Serielle Kommunikation	Übertragungsgeschwindigkeit	9600, 19200, 38400, 57600, 115200 Bit/s
		Unterstütztes Protokoll	No-Protocol, MC-Protokoll, SYSWAY, KV STUDIO
E/A	USB	Kommunikationsstandard	Gemäß USB 2.0 (Fullspeed)
Lesen	Unterstützte Codes	2D	QR, MicroQR, DataMatrix (ECC200), GS1 DataMatrix, PDF417, Micro PDF417, GS1 Composite (CC-A, CC-B, CC-C)
		Strichcode	CODE39, ITF, 2of5 (Industrial 2of5), COOP 2of5, NW-7 (Codabar), CODE128, GS1-128, GS1 DataBar, CODE93, JAN/EAN/UPC, Trioptic CODE39, CODE39 Full ASCII, Pharmacode
	Kleinstmögliche Auflösung	2D	0,063 mm
		Strichcode	0,082 mm
	Leseabstand		110 bis 1000 mm
	Sichtfeld		122 × 97 mm (Richtwert in 400 mm Entfernung)
Nennwerte	Stromspannung		24 VDC ±10%
	Stromverbrauch		Circa 700 mA
Umgebungsbeständigkeit	Schutzart		IP65
	Umgebungstemperatur		0 bis +45°C
	Lagertemperatur		-10 bis +50°C
	Luftfeuchtigkeit im Betrieb		35 bis 85% r.F. (keine Kondensation)
	Lagerfeuchtigkeit		
	Umgebungsbeleuchtung		Sonnenlicht : 10000 Lux, Glühlampe : 6000 Lux, Leuchtstoffröhre : 2000 Lux
	Betriebsumgebung		Kein Staub, keine ätzenden Gase
	Vibrationen		10 bis 55 Hz, Doppelamplitude : 0,75 mm, je 3 Stunden in X-, Y- und Z-Richtung

KEYENCE DEUTSCHLAND 0800-KEYENCE (Kostenfrei) http://www.keyence.de/ 2016/05/18 Page:1/4

Abbildung 7.1: Datenblatt SR-1000 Seite 1

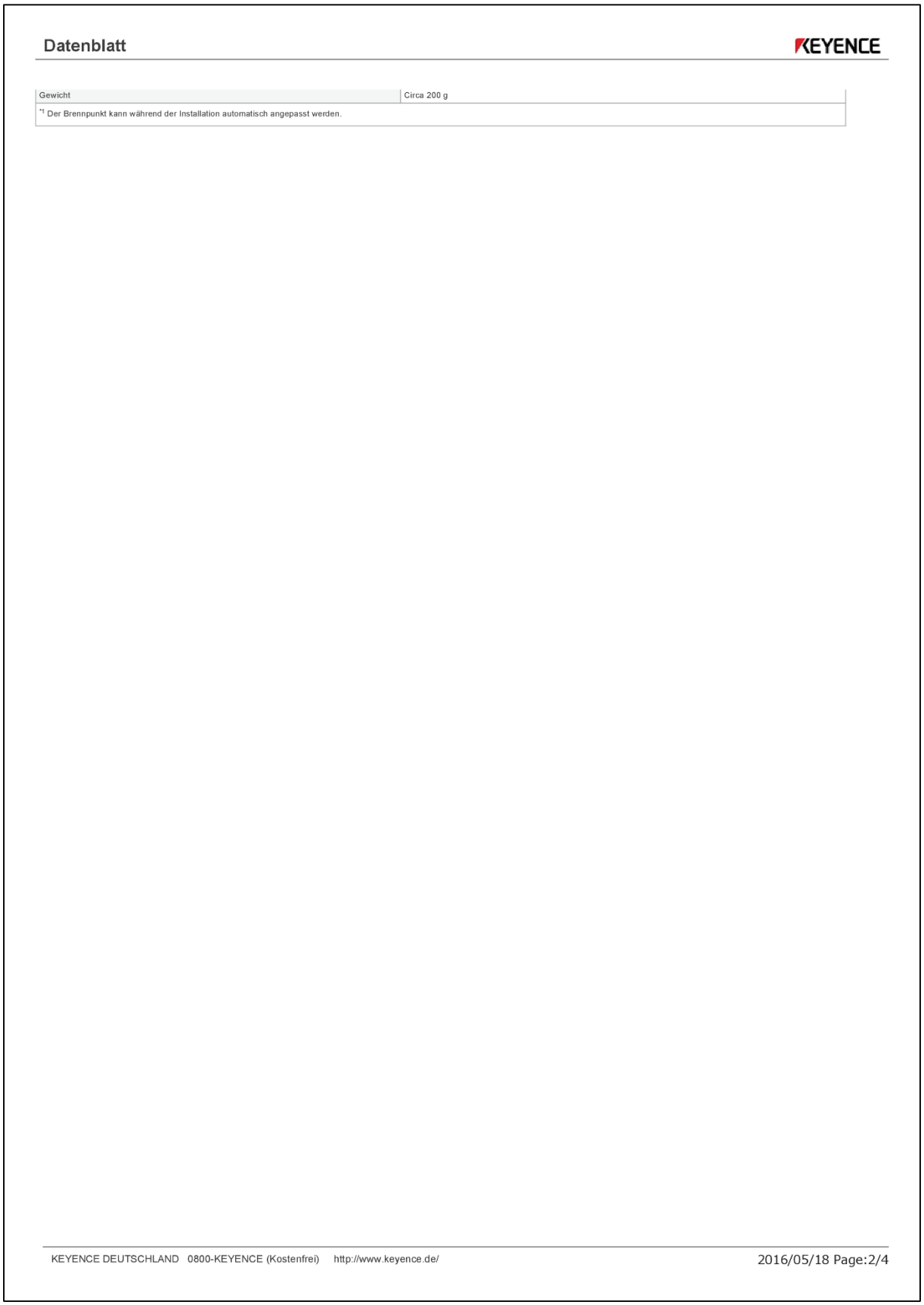
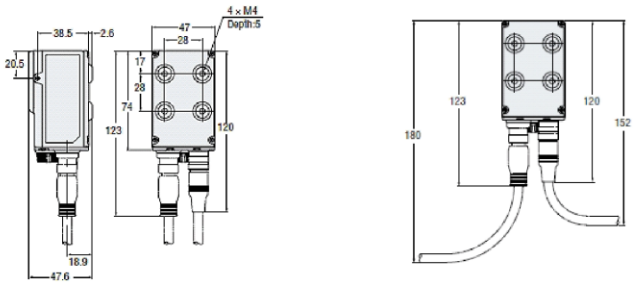


Abbildung 7.2: Datenblatt SR-1000 Seite 2

Datenblatt KEYENCE

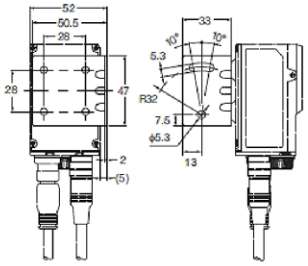
Abmessungen

SR-1000_1000W_dimension_01.gif

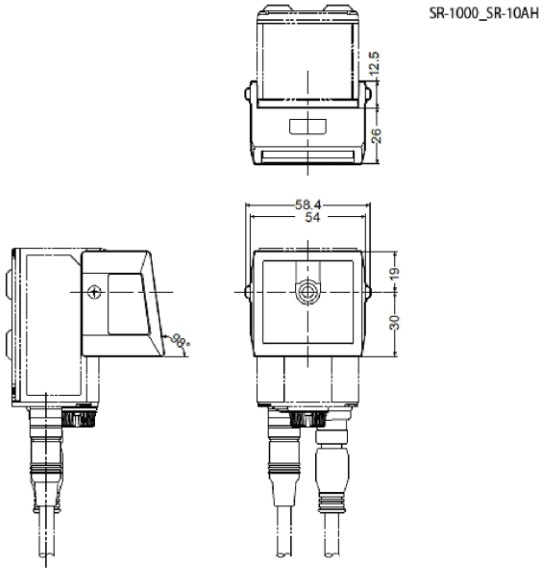


SR-1000_1000W_dimension_02.gif

■ When the mounting bracket (OP-87866) is used



sr1000_10ah_dimension_01.gif



sr1000_10ar_dimension_01.gif

Abbildung 7.3: Datenblatt SR-1000 Seite 3

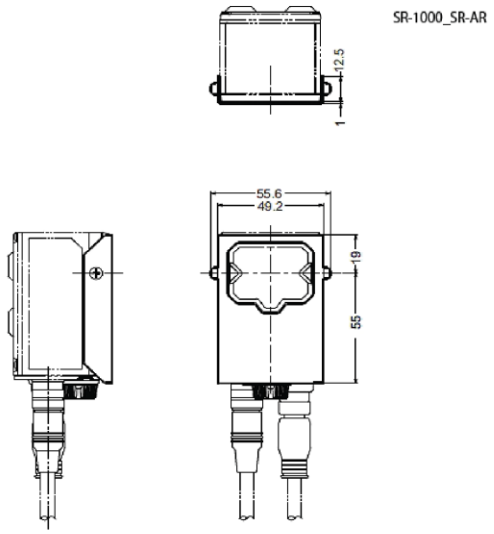


Abbildung 7.4: Datenblatt SR-1000 Seite 4

7.2 S7-1200 SPS

Hinweis

Nicht verwendete analoge Eingänge sollten kurzgeschlossen werden.

A.4 CPU 1214C

A.4.1 Allgemeine technische Daten und Leistungsmerkmale

Tabelle A- 44 Allgemein

Technische Daten	CPU 1214C AC/DC/Relais	CPU 1214C DC/DC/Relais	CPU 1214C DC/DC/DC
Bestellnummer	6ES7 214-1BG31-0XB0	6ES7 214-1HG31-0XB0	6ES7 214-1AG31-0XB0
Abmessungen (B x H x T) (mm)	110 x 100 x 75	110 x 100 x 75	110 x 100 x 75
Versandgewicht	475 Gramm	435 Gramm	415 Gramm
Leistungsverlust	14 W	12 W	12 W
Verfügbarer Strom (SM- und CM-Bus)	max. 1600 mA (5 V DC)	max. 1600 mA (5 V DC)	max. 1600 mA (5 V DC)
Verfügbarer Strom (24 V DC)	max. 400 mA (Geberversorgung)	max. 400 mA (Geberversorgung)	max. 400 mA (Geberversorgung)
Stromaufnahme digitaler Eingang (24 V DC)	4 mA/Eingang	4 mA/Eingang	4 mA/Eingang

Tabelle A- 45 CPU-Merkmale

Technische Daten		Beschreibung
Anwenderspeicher[1]	Arbeitsspeicher	75 KB
	Last	4 MB intern, erweiterbar bis auf SD-Kartengröße
	Remanent	10 KB
Integrierte digitale E/A		14 Eingänge/10 Ausgänge
Integrierte analoge E/A		2 Eingänge
Größe des Prozessabbilds		1024 Bytes Eingänge (E)/1024 Bytes Ausgänge (A)
Merker (M)		8192 Byte
Temporärer (lokaler) Speicher		• 16 KB für Anlauf und Programmzyklus (einschließlich der zugehörigen FBs und FCs) • 4 KB für Standardalarmereignisse, einschließlich FBs und FCs • 4 KB für Fehleralarmereignisse, einschließlich FBs und FCs
Zusätzliche Signalmodule		max. 8 SMs

Abbildung 7.5: Datenblatt S7-1200 Seite 1

Technische Daten	Beschreibung
Erweiterung SB, CB, BB	max. 1
Zusätzliche Kommunikationsmodule	max. 3 CM
Schnelle Zähler	6 insgesamt, siehe Tabelle Funktionsweise von schnellen Zählern (Seite 359) • Einphasenzähler: 3 bei 100 kHz und 3 bei 30 kHz Takt, • A/B-Zähler: 3 bei 80 kHz und 3 bei 20 kHz Takt
Impulsausgänge [2]	4
Eingänge für Impulsabgriff	14
Verzögerungs-/Weckalarme	4 mit Auflösung von 1 ms
Flankenalarme	12 steigend und 12 fallend (14 und 14 mit optionalem Signalboard)
Memory Card	SIMATIC Memory Card (optional)
Genauigkeit Echtzeituhr	+/- 60 Sekunden/Monat
Pufferung Echtzeituhr	Typ. 20 Tage/min. 12 Tage bei 40 °C (wartungsfreier Hochleistungskondensator)

[1] Die Größe des Anwenderprogramms, der Daten und der Konfiguration ist durch den verfügbaren Ladespeicher und den Arbeitsspeicher in der CPU begrenzt. Die Anzahl der unterstützten OBs, FCs, FBs und DBs und die Größe der einzelnen Bausteine ist nicht begrenzt. Die einzige Begrenzung ist die Gesamtspeicherkapazität.

[2] Bei CPU-Varianten mit Relaisausgängen müssen Sie ein digitales Signalboard (SB) installieren, um die Impulsausgänge zu verwenden.

Tabelle A- 46 Leistung

Art der Anweisung	Ausführungsgeschwindigkeit
Boolesch	0,08 µs/Operation
Wort übertragen	1,7 µs/Operation
Realzahlenarithmetik	2,3 µs/Operation

A.4.2 Von der CPU 1214C unterstützte Zeiten, Zähler und Codebausteine

Tabelle A- 47 Von der CPU 1214C unterstützte Bausteine, Zeiten und Zähler

Element		Beschreibung
Bausteine	Typ	OB, FB, FC, DB
	Größe	64 KB
	Anzahl	Bis 1024 Bausteine gesamt (OBs + FBs + FCs + DBs)
	Adressbereich für FBs, FCs und DBs	1 bis 65535 (z. B. FB 1 bis FB 65535)
	Schachtelungstiefe	16 aus Zyklus- oder Anlauf-OBs, 4 aus Verzögerungsalarm-, Uhrzeitalarm-, Weckalarm-, Prozessalarm-, Zeitfehler- oder Diagnosefehler-OBs
	Überwachung	Der Zustand von 2 Codebausteinen kann gleichzeitig überwacht werden.
OBs	Programmzyklus	Mehrere: OB 1, OB 200 bis OB 65535

Abbildung 7.6: Datenblatt S7-1200 Seite 2

Technische Daten
A.4 CPU 1214C

Element		Beschreibung
	Anlauf	Mehrere: OB 100, OB 200 bis OB 65535
	Verzögerungsalarme und Weckalarme	4[1] (1 pro Ereignis): OB 200 bis OB 65535
	Prozessalarme (Flanken und HSC)	50 (1 pro Ereignis): OB 200 bis OB 65535
	Zeitfehleralarme	1: OB 80
	Diagnosefehleralarme	1: OB 82
Zeiten	Typ	IEC
	Anzahl	Nur durch die Speicherkapazität begrenzt
	Speicherung	Struktur im DB, 16 Bytes pro Zeit
Zähler	Typ	IEC
	Anzahl	Nur durch die Speicherkapazität begrenzt
	Speicherung	Struktur im DB, Größe abhängig von der Zählart • SInt, USInt: 3 Byte • Int, UInt: 6 Byte • DInt, UDInt: 12 Byte

[1] Verzögerungs- und Weckalarme nutzen dieselben Ressourcen in der CPU. Es darf insgesamt maximal 4 dieser Alarme geben (Verzögerungs- plus Weckalarme). 4 Verzögerungsalarme und 4 Weckalarme sind nicht möglich.

Tabelle A- 48 Kommunikation

Technische Daten	Beschreibung
Schnittstellen	1
Typ	Ethernet
HMI-Gerät[1]	3
Programmiergerät (PG)	1
Anschlüsse	• 8 für die offene Benutzerkommunikation (aktiv oder passiv): TSEND_C, TRCV_C, TCON, TDISCON, TSEND und TRCV • 3 für die S7-Kommunikation über Server-GET/PUT (CPU-zu-CPU) • 8 für die S7-Kommunikation über Client-GET/PUT (CPU-zu-CPU)
Datenraten	10/100 MBit/s
Elektrische Trennung (externes Signal zu PLC-Logik)	Wandler potentialgetrennt, 1500 V AC, nur für kurzfristige Sicherheit
Kabelart	CAT5e geschirmt

[1] Die CPU bietet zweckbestimmte HMI-Verbindungen, um bis zu 3 HMI-Geräte zu unterstützen. (Sie können bis zu 2 SIMATIC Comfort Panels haben.) Wie viele HMI-Geräte insgesamt unterstützt werden, hängt von den Typen der HMI-Panels in Ihrer Konfiguration ab. Sie können beispielsweise bis zu drei SIMATIC Basic Panels an Ihre CPU anschließen, oder Sie können bis zu zwei SIMATIC Comfort Panels und ein zusätzliches Basic Panel anschließen.

Abbildung 7.7: Datenblatt S7-1200 Seite 3

Tabelle A- 49 Spannungsversorgung

Technische Daten		CPU 1214C AC/DC/Relais	CPU 1214C DC/DC/Relais	CPU 1214C DC/DC/DC
Spannungsbereich		85 bis 264 V AC	20,4 V DC bis 28,8 V DC 22,0 VDC bis 28,8 V DC bei Umgebungstemperatur -20 °C bis 0 °C	
Netzfrequenz		47 bis 63 Hz	--	
Einschaltstrom (max. Last)	Nur CPU	100 mA bei 120 V AC 50 mA bei 240 V AC	500 mA bei 24 V DC	
	CPU mit allen Erweiterungsbaugruppen	300 mA bei 120 V AC 150 mA bei 240 V AC	1500 mA bei 24 V DC	
Einschaltstrom (max.)		20 A bei 264 V AC	12 A bei 28,8 V DC	
Elektrische Trennung (Eingangsleistung zu Logik)		1500 V AC	Nicht elektrisch getrennt	
Kriechstrom an Erde, AC-Leitung an Funktionserde		max. 0.5 mA	-	
Verzögerungszeit (Spannungsverlust)		20 ms bei 120 V AC 80 ms bei 240 V AC	10 ms bei 24 V DC	
Interne Sicherung, nicht durch Anwender austauschbar		3 A, 250 V, träge		

Tabelle A- 50 Geberversorgung

Technische Daten	CPU 1214C AC/DC/Relais	CPU 1214C DC/DC/Relais	CPU 1214C DC/DC/DC
Spannungsbereich	20,4 bis 28,8 V DC	L + minus 4 V DC min. L+ minus 5 V DC min. bei einer Umgebungstemperatur von -20 °C bis 0 °C	
Nennausgangsstrom (max.)	400 mA (kurzschlussfest)		
Max. Welligkeit/Störströme (<10 MHz)	< 1 V SpitzeSpitze	Wie Eingangsleitung	
Elektrische Trennung (CPU-Logik zu Sensorversorgung)	Nicht elektrisch getrennt		

Abbildung 7.8: Datenblatt S7-1200 Seite 4

A.4.3 Digitale Eingänge und Ausgänge

Tabelle A- 51 Digitaleingänge

Technische Daten	CPU 1214C AC/DC/Relais	CPU 1214C DC/DC/Relais	CPU 1214C DC/DC/DC
Anzahl der Eingänge	14		
Typ	Stromziehend/stromliefernd (IEC Typ 1, wenn stromziehend)		
Nennspannung	24 V DC bei 4 mA, Nennwert		
Zulässige Dauerspannung	max. 30 V DC		
Stoßspannung	35 V DC für 0,5 s		
Signal logisch 1 (min.)	15 V DC bei 2,5 mA		
Signal logisch 0 (max.)	5 V DC bei 1 mA		
Elektrische Trennung (Feld zu Logik)	500 V AC für 1 Minute		
Potenzialgetrennte Gruppen	1		
Filterzeiten	0,2, 0,4, 0,8, 1,6, 3,2, 6,4 und 12,8 ms (wählbar in Gruppen zu je 4)		
HSC Eingangstaktfrequenzen (max.) (Pegel logisch 1 = 15 bis 26 V DC)	Einphasenzähler: 100 kHz (Ea.0 bis Ea.5) und 30 kHz (Ea.6 bis Eb.5) A/B-Zähler: 80 kHz (Ea.0 bis Ea.5) und 20 kHz (Ea.6 bis Eb.5)		
Anzahl gleichzeitig eingeschalteter Eingänge	• 7 (keine benachbarten Punkte) bei 60 °C horizontal oder 50 °C vertikal • 14 bei 55 °C horizontal oder 45 °C vertikal		
Leitungslänge (Meter)	500 m geschirmt, 300 m ungeschirmt, 50 m geschirmt für HSC-Eingänge		

Tabelle A- 52 Digitale Ausgänge

Technische Daten	CPU 1214C AC/DC/Relais und DC/DC/Relais	CPU 1214C DC/DC/DC
Ausgänge	10	10
Typ	Relais, Trockenkontakt	MOSFET, elektronisch (stromliefernd)
Spannungsbereich	5 bis 30 V DC oder 5 bis 250 V AC	20,4 bis 28,8 V DC
Signal logisch 1 bei max. Strom	--	min. 20 V DC
Signal logisch 0 bei 10 kΩ Last	--	max. 0,1 V DC
Strom (max.)	2,0 A	0,5 A
Lampenlast	30 W DC/200 W AC	5 W
Widerstand bei EIN	max. 0,2 Ω wenn neu	max. 0,6 Ω
Kriechstrom pro Ausgang	--	max. 10 µA
Einschaltstrom	7 A bei geschlossenen Kontakten	8 A für max. 100 ms
Überlastschutz	Nein	Nein
Elektrische Trennung (Feld zu Logik)	1500 V AC für 1 Minute (Spule zu Kontakt) Keine (Spule zu Logik)	500 V AC für 1 Minute
Isolationswiderstand	min. 100 MΩ, wenn neu	--
Elektrische Trennung zwischen offenen Kontakten	750 V AC für 1 Minute	--

Abbildung 7.9: Datenblatt S7-1200 Seite 5

Technische Daten	CPU 1214C AC/DC/Relais und DC/DC/Relais	CPU 1214C DC/DC/DC
Potentialgetrennte Gruppen	2	1
Induktive Klemmspannung	--	L+ minus 48 V DC, 1 W Verlustleistung
Schaltverzögerung (Aa.0 bis Aa.3)	max. 10 ms	max. 1,0 µs von Aus nach Ein max. 3,0 µs von Ein nach Aus
Schaltverzögerung (Aa.4 bis Ab.1)	max. 10 ms	max. 50 µs von Aus nach Ein max. 200 µs von Ein nach Aus
Maximale Schaltfrequenz Relais	1 Hz	--
Frequenz Impulsgenerator (Aa.0 und Aa.2)	Nicht empfehlenswert [1]	max. 100 kHz, min. 2 Hz[2]
Mechanische Lebensdauer (ohne Last)	10.000.000 Schaltspiele auf/zu	--
Lebensdauer der Kontakte bei Nennlast	100.000 Schaltspiele auf/zu	--
Verhalten bei Wechsel von RUN nach STOP	Letzter Wert oder Ersatzwert (Voreinstellung 0)	
Anzahl gleichzeitig eingeschalteter Ausgänge	• 5 (keine benachbarten Punkte) bei 60 °C horizontal oder 50 °C vertikal • 10 bei 55 °C horizontal oder 45 °C vertikal	
Leitungslänge (Meter)	500 m geschirmt, 150 m ungeschirmt	

[1] Bei CPU-Varianten mit Relaisausgängen müssen Sie ein digitales Signalboard (SB) installieren, um die Impulsausgänge zu verwenden.

[2] Je nach Impulsempfänger und Kabel kann ein zusätzlicher Lastwiderstand (bei mindestens 10% des Nennstroms) die Qualität der Impulssignale und die Störfestigkeit verbessern.

A.4.4 Analoge Eingänge

Tabelle A- 53 Analoge Eingänge

Technische Daten	Beschreibung
Anzahl der Eingänge	2
Typ	Spannung (Eintakteingang)
Vollausschlagsbereich	0 bis 10 V
Vollausschlag (Datenwort)	0 bis 27648
Überschwingbereich	10,001 bis 11,759 V
Überschwingbereich (Datenwort)	27.649 bis 32.511
Überlaufbereich	11,760 bis 11,852 V
Überlaufbereich (Datenwort)	32.512 bis 32.767
Auflösung	10 Bits
Max. Stehspannung	35 V DC

Abbildung 7.10: Datenblatt S7-1200 Seite 6

7.3 KTP600 Basic Color PN Touch Panel

8.8 Technische Daten

8.8.1 Technische Daten des KTP400 Basic und KTP600 Basic

Gewicht

	KTP400 Basic Mono PN	KTP600 Basic Mono PN	KTP600 Basic Color DP	KTP600 Basic Color PN
Gewicht ohne Verpackung	ca. 320 g	ca. 1070 g		

Display

	KTP400 Basic Mono PN	KTP600 Basic Mono PN	KTP600 Basic Color DP	KTP600 Basic Color PN
Typ	LCD mono FSTN		LCD-TFT	
Display-Bereich, aktiver	76,79 mm x 57,59 mm (3,8")	115,2 mm x 86,4 mm (5,7")		
Auflösung, Bildpunkte	320 x 240			
Farben, darstellbare	4 Graustufen		256	
Kontrastregelung	Ja		Nein	
Pixel-Fehlerklasse nach DIN EN ISO 13406-2	-		II	
Hintergrundbeleuchtung Half Brightness Life Time, typisch	LED 30.000 h	CCFL 50.000 h		

Eingabeeinheit

	KTP400 Basic Mono PN	KTP600 Basic Mono PN	KTP600 Basic Color DP	KTP600 Basic Color PN
Typ	Touchscreen, analog-resistiv			
Funktionstasten	4	6		
Beschriftungsstreifen	Ja			

Speicher

	KTP400 Basic Mono PN	KTP600 Basic Mono PN	KTP600 Basic Color DP	KTP600 Basic Color PN
Anwendungsspeicher	512 kByte			

Abbildung 7.11: Datenblatt KTP600 Seite 1

Schnittstellen

	KTP400 Basic Mono PN	KTP600 Basic Mono PN	KTP600 Basic Color DP	KTP600 Basic Color PN
1 x RS 422/RS 485	-	-	Max. 12 Mbit/s	-
1 x Ethernet	RJ45 10/100 Mbit/s	RJ45 10/100 Mbit/s	-	RJ45 10/100 Mbit/s

Versorgungsspannung

	KTP400 Basic Mono PN	KTP600 Basic Mono PN	KTP600 Basic Color DP	KTP600 Basic Color PN
Nennspannung	DC +24 V			
Bereich, zulässiger	19,2 V bis 28,8 V (–20 %, +20 %)			
Transienten, maximal zulässig	35 V (500 ms)			
Zeit zwischen zwei Transienten, mindestens	50 s			
Stromaufnahme • Typisch • Dauerstrom, maximal • Einschaltstromstoß I^2t	 ca. 70 mA ca. 150 mA ca. 0,5 A^2s	 ca. 240 mA ca. 350 mA ca. 0,5 A^2s	 ca. 350 mA ca. 550 mA ca. 0,5 A^2s	
Absicherung, intern	Elektronisch			

Sonstiges

	KTP400 Basic Mono PN	KTP600 Basic Mono PN	KTP600 Basic Color DP	KTP600 Basic Color PN
Echtzeituhr	Ja, ungepuffert			

Abbildung 7.12: Datenblatt KTP600 Seite 2