Thema: Diplomarbeit MPS 2019 hftm Biel

Technischer Bericht

Betreuer: Alain Rohr

Experte: Reto Koenig

Autoren: Manuel Stöckli, Sven Blaser

Datum: 5. September 2019

1 ZUSAMMENFASSUNG

Jährlich findet der RoboCup mit der Logistics League statt. Das steigende Level der Teams bringt das Bedürfnis einer verbesserten Spielumgebung mit sich. Die MPS Stationen bilden dabei einen zentralen Punkt. Diese werden von der hftm unterhalten. Im Jahr 2019 wurde deswegen die Kommunikation der SPSen auf OPC UA umgebaut und ein Produktetracking erstmals eingesetzt. Das Produktetracking lief während den Wettkämpfen nicht zuverlässig genug, um es durchgängig einzusetzen. Auch der Rest des Programmes kann nach den neusten Anpassungen und Einbindungen einfacher, lesbarer und wiederverwendbar aufgebaut werden.

Das Programm wurde deshalb neu geschrieben. Dabei wurde eine objektorientierte Lösung angestrebt. Das Funktionieren des Produkte Trackings ist beim neuen Programm ein Muss. Wird ein Barcode im Programmablauf nicht erkannt, wird dies als Fehler ausgegeben. Somit wird eine falsche Punktevergabe nicht mehr möglich sein.

Mit dem neuen Programm werden mehr Fehlerarten überwacht. Somit können Benachteiligungen von Teams durch Fehler der MPS Stationen vermieden werden. Ereignet sich ein Fehler ist der Ort und Zeitpunkt im Programmablauf in der Web Visu ersichtlich. Nach dem Fehlerbehebung des Schiedsrichters auf der Station kann der Auftrag erneut gesendet werden und das Spiel verläuft normal weiter.

Die Web Visu wurde Benutzerfreundlich und möglichst selbsterklärend aufgebaut. Jede Station hat neu zwei speziell für sie angefertigte Bilder. Dies hilft bei der Unterteilung der verschiedenen Verwendungszwecke. Die wichtigsten Parameter sind mit ihr erreichbar und änderbar geworden. So ist nicht immer eine Verbindung mit Codesys zur Station nötig, um Anpassungen vorzunehmen.

Auch der nicht konstante Aufbau der MPS Stationen ab Werk wird kein Problem mehr sein. Die Parametrierungen der verschiedenen Objekte, wie z.B. die eines Zylinders können einfach über die Web Visu der SPS angepasst werden.

2 INHALT

3 EINLEITUNG

Die hftm ist für die Funktion der MPS Stationen in der Logistics League des RoboCups verantwortlich. Das steigende Level der Liga bewegt sie deshalb die Fähigkeiten und Zuverlässigkeit der Stationen zu erhöhen. Durch die Grösse des Projektes und derer Komplexität bot dies die Möglichkeit einer Diplomarbeit.

Die objektorientierte Programmierung ist in der SPS-Welt noch nicht so weit fortgeschritten wie in der IT-Welt. Auf Grund mehrerer Änderungen im MPS Stationen Programm wurde das Programm langsam unleserlich. Dies bot die Gelegenheit es in einer neuen, objektorientierten Version zu schreiben und zu verbessern.

In einem Projekt des 2. Studienjahres programmierten wir bereits die MPS Storage Station. Auch am RoboCup waren wir mit dabei. Somit waren wir bereits gut mit dem Thema bekannt. Die Möglichkeit seine Fähigkeiten in der ST-Programmierung verbessern und festigen zu können war ein grosser Punkt, weshalb es diese Diplomarbeit unter die Top drei unsere Favoritenliste der Diplomarbeiten geschafft hat.

4 AUFGABESTELLUNG

Das komplette Aufgabenheft ist im digitalen Anhang zu finden.

2 Aufgabenstellung

Ausgangslage

Festo stellt mit 2*7 MPS (Mechatronic Production Systems) die Infrastruktur für den RoboCup zur Verfügung. Seit 2019 werden für diese Systeme die neusten Festo-SPSen eingesetzt werden.

Die SPS-Software existiert bereits in einer funktionsfähigen Version mit SPS in einer knapp getesteten Alpha-Version mit einer neuen Kommunikation.

Ziele des Auftrags

Die aktuell existierende Beta-Version der SPS-Software soll im Sinne von Einfachheit/Lesbarkeit/Wiederverwendbarkeit überarbeitet, optimiert und ergänzt werden.

Die Zuverlässigkeit und Stabilität des Produkte-Tracking (Barcode-Reader) ist MPS-seitig zu erhöhen.

Weiter soll die neue Storage Station V2 analysiert werden und soweit wie (beim aktuellen Stand von Festo) möglich ins System integriert werden

Definition der Arbeiten in der Diplomarbeit

- Analyse/Verstehen der aktuellen OPC-UA Kommunikation und MPS-Software
- Optimierung/Neu-Implementation einzelner Bausteine (generische Bausteine soweit es Sinn macht)
- Objektorientierte Konzepte sinnvoll einbauen.
- Komplettieren/Einbauen des Fehlerhandlings
- Erstellen eines HMI's / Visualisierung
- Analyse Storage Station V2
- Implementationgrad SS nach Absprache mit Auftraggeber (Analyse V2) (Konzept 2.te SPS mit Logik-Abstraktion/Mockup)

Die Dokumentation des Systems und programmierter Teile unterstützt obigen Bedürfnisse (Aufzählung ist nicht abschliessend)

- Planung der Arbeit
- Nachvollziehbares Testing (mit Refbox-Mockup-Tools)
- Erarbeitung eines technischen Berichtes
- Präsentation

Die Verwaltung/Planung von (Milestones/Issues) sowie die Dokumentation sollen auf GitLab umgesetzt werden. https://gitlab.com/solidus/mps

hftm | AufgabenstellungDA_BlaserStoeckli2019.docx
5

5 ZEITPLANUNG

Die Zeitplanung wurde im Excel durchgeführt und befindet sich im Anhang 1.5. In der Aufgabenstellung wurde gefordert, dass die Planung und Dokumentation im GitLab umgesetzt wird. Nach Absprache mit dem Dozenten konnten wir die Zeitplanung im Excel durchführen.

Wir haben gegen Ende der Arbeit noch eine Woche Puffer eingebaut, damit wir bei allfälligen Verzögerungen noch genügend Zeit haben. Auch die Wunsch Punkte im Pflichtenheft, können in dieser Zeit noch erledigt werden.

Die Konzeptphase fiel kürzer aus als geplant. Dadurch konnten wir früher in die Umsetzungsphase übergehen. Da wir einige Bausteine bereits testen wollten, musste die Visualisierung früher starten. So konnten wir fortlaufend die neuen Programmbausteine mit der Visualisierung testen.

Die ganze Umsetzungsphase zog sich ein wenig in die Länge. Einige Programmteile wurden nach Besprechungen wieder umprogrammiert und weiterentwickelt. Am Ende wurde der Puffer vor allem für die Dokumentation verwendet, da während der Umsetzungsphase nicht so viel dokumentiert wurde wie zuerst angedacht.

Im Grossen und Ganzen konnte der Zeitplan gut eingehalten werden und wir kamen nicht in den Stress.

6 ANALYSE

6.1 ANALYSE STORAGE STATION V2

6.1.1 Ansteuerung

Die Station kann über die IEEE 488 /24 Pin kompatible Steckverbindung mit einem SPS-Board oder mit einer E/A-Simulationsbox angesteuert werden. (Festo, Betriebsanleitung)

Abbildung 1 IEEE488-Stecker (rs-online, 19)

Es gibt verschiedene Befehle, die an die Lagersteuerung gesendet werden können:

- Referenzfahrt
- Ein- und Auslagern mit Farbcode
- Auslagern über Web-Visualisierung
- Durchschleusen von Werkstücken/Kartons
- Umschalten zwischen Ein- und Auslagern
- Ein- und Auslagern auf bestimmte Regalpositionen

Für Details dieser Befehle kann im Dokument 8082795_Ablaufdiagramme von Festo auf Seiten 6 -12 nachgeschlagen werden.

Dies bedeutet, dass eine zweite SPS benötigt wird um eine Ein-/Auslagerung zu steuern. In dieser SPS würde auch die Lagerverwaltung abgewickelt. Auch ältere Storage Station Modelle müssten mit zwei SPS ausgestattet werden. Auf der einen wäre dabei das Achscontrolling. Die Andere würde wie bei der neuen Storage Station für Ein-/Auslagerung und Lagerverwaltung verwendet.

Die Möglichkeit die Ein-/Auslagerung auf der bereits vorhandenen Steuerung zu implementieren würde auch bestehen. Dies hat jedoch zur Folge, dass nicht mehr jede MPS mit demselben Programm geladen werden kann.

6.1.1.1 I/O Tabelle

In diesen Tabellen sind alle Inputs und Outputs aller Modis zusammengefasst.

Inputs

Bit	Beschreibung
I0.0	Station referenziert
I0.2	Station beschäftigt (busy)
I0.4	Station verarbeitet Arbeitsauftrag
I0.5	Station bereit Farb- & Zahlencode zu empfangen

Tabelle 1 Storage Station V2 Inputs

Outputs

Bit	Beschreibung
Q0.0	Station referenzieren
Q0.3	Farb- / Zahlencode Bit 0
Q0.4	Farb- / Zahlencode Bit 1
Q0.5	Farb- / Zahlencode Bit 2
Q0.6	Strobe (Code übernehmen)
Q0.7	Automatikmodus freischalten (EnAuto)

Tabelle 2 Storage Station V2 Outputs

6.2 Analyse Hardware

6.2.1 Objekteaufbau Stationen

	BS	CS	RS	DS	SS
Zylinder zwei Weg Zwei Endlagen	3	1	2	2	
Zylinder federrückgestellt Ohne Endlage		1			
Zylinder zwei Weg Endlage RP		1	2		
Barcodescanner	2	1	1	1	2
Motor	1	1	1	1	1
Vacuum		1	2		
Light	3	3	3	3	3
Gate			1		1

Tabelle 3 Stationen Hardware Aufbau

6.3 Analyse Programm

Das vorhandene Programm hat im Ansatz objektorientierte Züge. Es besitzt für jeden Stationstypen einen eigenen Programmbaustein. Das Ganze wird im Main-Baustein aufgerufen. Dieser selber wird zyklisch aufgerufen.

Aufbau der wichtigsten Bausteine, Listen & Struct's:

6.3.1 Main (Programmbaustein)

Im Main werden die Variablen Basic und In in die im Programm verwendeten Variablen ActMsg und ActBasicMsg überschrieben. Diese zwei Variablen legen in vielen Cases den aktuellen Case fest.

Das Wichtigste im Main ist der Aufruf der Stations Programmbausteine. Dies geschieht in einem Case. Im selben Case werden auch die Lampen geschalten.

6.3.2 G (Globale Variablenliste)

In der G Liste sind alle Ausgänge deklariert und «beschrieben». Da nicht bei jeder Anlage der Ausgang auf derselben Klemme liegt, ist es wichtig diese ganz oder grösstenteils in das neue Programm zu übernehmen.

6.3.3 Proto (Eigener Datentyp)

Die Protoliste ist ein eigener Datentyp. Dieser wird im Kapitel 6.5.1 genauer erläutert.

6.4 Analyse Barcodescanner

Der Barcodescanner ist an einem Raspberry Pi angeschlossen. Der Raspberry Pi verbindet sich mit dem OPC UA-Server und schreibt den Barcode direkt in ein bestimmtes Register. Dieses wird zurzeit nicht überprüft, sondern direkt von der Refbox ausgelesen. Das führte aber zu Problemen, da nicht immer ein Code erkannt wurde. So musste das Barcode Tracking am Cup wieder ausgeschaltet werden. Hier könnte durch eine einfache Validierung das ganze viel stabiler und weniger Fehleranfällig gemacht werden.

6.5 Analyse OPC UA

Die Kommunikation mit der Refbox und dem Barcodescanner erfolgt über den OPC UA Standart. Der Datenaustausch mit der Refbox erfolgt über die zwei Variablen In und Basic. Diese sind vom Datentyp ProtoUnion. Dies ist eine Union von einem ARRAY[0..4] OF WORD und dem Datentyp Proto.

Der Datentyp Proto ist jedoch grösser als fünf Words. Somit stellt sich die Union in Frage. Nach Rücksprache mit Jonas Jauslin konnten wir feststellen, dass die Union eine alte Leiche aus der Zeit der Verwendung von Modbus ist. Diese kann für das neue Programm eliminiert werden.

6.5.1 Datentyp Proto

Der Datentyp Proto ist wie folgt aufgebaut:

Variable	**Dateityp**
ActionId	WORD
Data	ARRAY[0..1] OF WORD
Status	StatBits
Error	BYTE
SlideCnt	WORD
BarCode	DWORD
PRG_Version	STRING[4]

Tabelle 4 Datentyp Proto

6.5.1.1 Detaillierte Beschreibung Proto

Diese Tabelle war bereits vorhanden und muss so belassen werden, da die Kommunikation zur Refbox nicht verändert werden darf. Die Tabelle befindet sich in voller Grösse im Anhang 2.6.

MPS	Description	Action ID	Payload (Word 1 & 2)		8 Status Bits								Error Nr Byte	Corresponding RefboxMsg	Reading Registers		
		Word 0	Word 1	Word 2	7	6	5	4	3	2	1	0			Word 4	Word 5	Word 6
									ENABLE	ERR	RDY	BUSY			SlideCnt Ring-Station	BarCode	Version-control
All	NoJob	0															
All	MachineTyp	10	1=BS/2=RS/3=CS/4=DS/5=SS														
Lights	reset	20															
	red	21	0=off/1=on/2=blink	0=always/>0=time in s										SetSignalLight			
	yellow	22	0=off/1=on/2=blink	0=always/>0=time in s										SetSignalLight			
	green	23	0=off/1=on/2=blink	0=always/>0=time in s										SetSignalLight			
		25	00000ryg	00000bbb													
BS	Reset	100															
	GetBase	101	1-3											BSPushBase			
	BandOnUntil	102	1=in//3=out	1=dirToOut/2=dirToIn										MoveConveyerBelt			
RS	Reset	200															
	WaitForXBases	201	not implemented														
	BandOnUntil	202	1=in/2=mid/3=out	1=dirToOut/2=dirToIn										MoveConveyerBelt			
	MountRing	203	1-2											RSMountRing			
CS	Reset	300															
	BandOnUntil	302	1=in/2=mid/3=out	1=dirToOut/2=dirToIn										MoveConveyerBelt			
	Cap	301	1=retrieve/2=mount											CSTask			
DS	Reset	400															
	DeliverToSlot	401	1-3											DSActivateGate			
SS	Reset	500															
	BandOnUntil	502	3	1										SSTask			

Abbildung 2 OPC UA Kommunikation

6.5.2 Datentyp StatBits

Die Statusbits sind ein Teil des Datentyps Proto.

Variable	Dateityp
Busy	BOOL
Ready	BOOL
Error	BOOL
Enable	BOOL
unused0	BOOL
unused1	BOOL
inSensor	BOOL
outSensor	BOOL

Tabelle 5 Datentyp StatBits

Die ersten vier Bits werden für die Kommunikation mit der Refbox verwendet. Für die letzten Beiden wurde kein Verweis im Programm gefunden. Diese können also auch als «unused» markiert werden. Auch hier befindet sich eine detailliertere Beschreibung auf den nächsten Seiten.

6.5.3 Variablen In und Basic

Die beiden Variablen sind vom gleichen Datentyp, unterscheiden sich jedoch durch ihren möglichen Inhalt.

Die Basic-Variable enthält nur Werte für die Lampen und die Information, welche Station diese Steuerung ist.

Die In-Variable kann alle Aktionen, welche ausgeführt werden könnten enthalten.

6.5.3.1 ActionID

Dieses Word wird von der Refbox geschrieben und von der SPS gelesen. Die ActionID kann alle Werte, welche auf der Abbildung 3 sichtbar sind, besitzen. Die Werte 10, 20 -23 und 25 werden nur in die Basic Variable geschrieben, die anderen werden alle auf die In Variable geschrieben. Die Zahlen sind nach einer bestimmten Logik aufgebaut.

MPS	Description	Action ID
		Word 0
All	NoJob	0
All	MachineTyp	10
Lights	reset	20
	red	21
	yellow	22
	green	23
		25
BS	Reset	100
	GetBase	101
	BandOnUntil	102
RS	Reset	200
	WaitForXBases	201
	BandOnUntil	202
	MountRing	203
CS	Reset	300
	BandOnUntil	302
	Cap	301
DS	Reset	400
	DeliverToSlot	401
SS	Reset	500
	BandOnUntil	502

Abbildung 3 OPC UA Action ID

Jede Station besitzt eine dreistellige Zahl. Die erste Stelle ist der Machinentyp, die zweite und dritte sind die Befehlsziffern.

- Die Endung 00 ist der Reset Befehl, der die Station in Ausgansstellung zurückbringen soll.
- Die Endung 02 ist für alle Stationen der Befehl BandOnUntil.
- Die Endungen 01 und 03 sind für stationsspezifische Befehle.

6.5.3.2 Payload (Word 1 & 2)

Diese beiden Word's werden von der Refbox geschrieben und von der SPS gelesen. Im Payload befinden sich die Daten zur dazugehörigen ActionID. Im Bild unterhalb ist der Ausschnitt der CapStation sichtbar.

Kommt als Beispiel die ActionID 302 mit der Ziffer 2 im ersten und der Ziffer 1 im zweiten Word an, wird die Station das Band Richtung Output starten. Sobald beim Sensor in der Mitte ein Signal kommt wird das Band abgestellt und der Job ist beendet.

CS	Reset	300		
	BandOnUntil	302	1=in/2=mid/3=out	1=dirToOut/2=dirToIn
	Cap	301	1=retrieve/2=mount	

Abbildung 4 OPC UA Word0 &Word1

6.5.3.3 StatusBits

Die Statusbits werden von der SPS geschrieben und von der Refbox gelesen und teilweise auch geschrieben.

- Das Bit Busy wird jedes Mal auf True gesetzt, wenn ein Job abgearbeitet wird.
- Das Bit RDY steht für Ready. Dieses wird gesetzt, wenn ein Bauteil am Output der Station wartet. Dieses bleibt solange True bis das Bauteil ab dem Band genommen wird.
- Das Bit ERR wird geschrieben, wenn ein Fehler im Programm der SPS geschehen ist. Das Fehlerhandling wurde jedoch noch nicht fertig umgesetzt und auch auf Seite Refbox wird dieses Bit noch nicht ausgewertet.
- Das Bit Enable wird verwendet, wenn ein neuer Auftrag gesendet wird. Es wird von der Refbox auf true gesetzt und muss anschliessend von der SPS wieder auf false gesetzt werden.

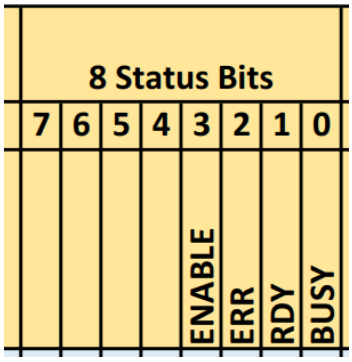


Abbildung 5 OPC UA Status Bits

6.5.3.4 Error Byte

Das Errorbyte ist vorbereitet, wird im Main-Programm auch geschrieben. Da die Fehlerauswertung aber wie bereits bekannt noch nicht fertig umgesetzt wurde wird es noch nicht weiterverwendet. Auch die Refbox wertet dieses Byte noch nicht aus.

6.5.3.5 SlideCount RingStation

Das Word SlideCount wird nur bei der Ringstation verwendet. Die Ringstation besitzt zum normalen Band dazu noch ein zusätzliches Slide. Dort können Bases heruntergelassen werden. Diese werden gezählt und der Refbox über dieses Word mitgeteilt.

6.5.3.6 Barcode

In dieses DWord wir der Barcode direkt vom Raspi geschrieben. Die Refbox wertet diesen Wert direkt aus. Zurzeit wird dieser Wert von der SPS Seite nicht beachtet oder verändert.

6.5.3.7 PRG_Version

Die Programmversion zeigt die aktuelle Version des Programmes an. Dies dient als Hilfe, wenn neue Programme geladen werden. So sieht man auf den ersten Blick ob bereits die aktuelle Version vorhanden ist.

6.6 Analyse Visualisierung

Die Visualisierung besteht aus einem Visualisierungsbild. Der linke Teil der Visualisierung ist mit Elementen aufgebaut, welche für jede Station verwendet werden können. Der rechte Teil besteht aus Elementen, die je nach Stationswahl nicht verwendbar sind.

Die Visu muss zuerst mit dem «Enable Visu» Button eingeschaltet werden, um damit Manipulationen vorzunehmen. Die gewünschte Station kann unter «Select Station Type» gewählt werden. Die geladene Programmversion wird ersichtlich und der letzte gelesene Barcode wird ausgegeben.

Unter «Comands» werden die aktuellen Werte im OPC UA Server ausgegeben. Der Zustand der Leuchte wird zudem im Status Feld angezeigt.

Die Leuchte kann farbspezifisch getestet werden. Die Farben können separat ein, aus oder blinkend geschaltet werden. Mit dem Schieberegler wird die Laufzeit der Schaltung eingestellt. Alle Steuerungen der Leuchte müssen mit dem Accept Button bestätigt werden.

Alle spezifischen Jobs für BS, DS, CS und RS müssen mit dem «Accept specific Job» bestätigt werden.

Bei der Base Station kann gewählt werden, welche Base ausgegeben werden soll.

Der Delivery Station kann gesagt werden in welchen Slot geliefert werden soll.

Mit der Cap Station kann das Aufsetzen oder das Entfernen eines Caps simuliert werden.

Mit der Ring Station kann das Aufsetzen eines Ringes aus dem 1. oder 2. Lager simuliert werden. Unter «Slide Counter» sind die gezählten Slides ersichtlich.

Im Band Job kann gewählt werden in welche Richtung das Band drehen soll, bis der gewünschte Sensor (In Mid oder Out) einschaltet. Mit «Set Band» kann die Aktion ausgeführt werden.

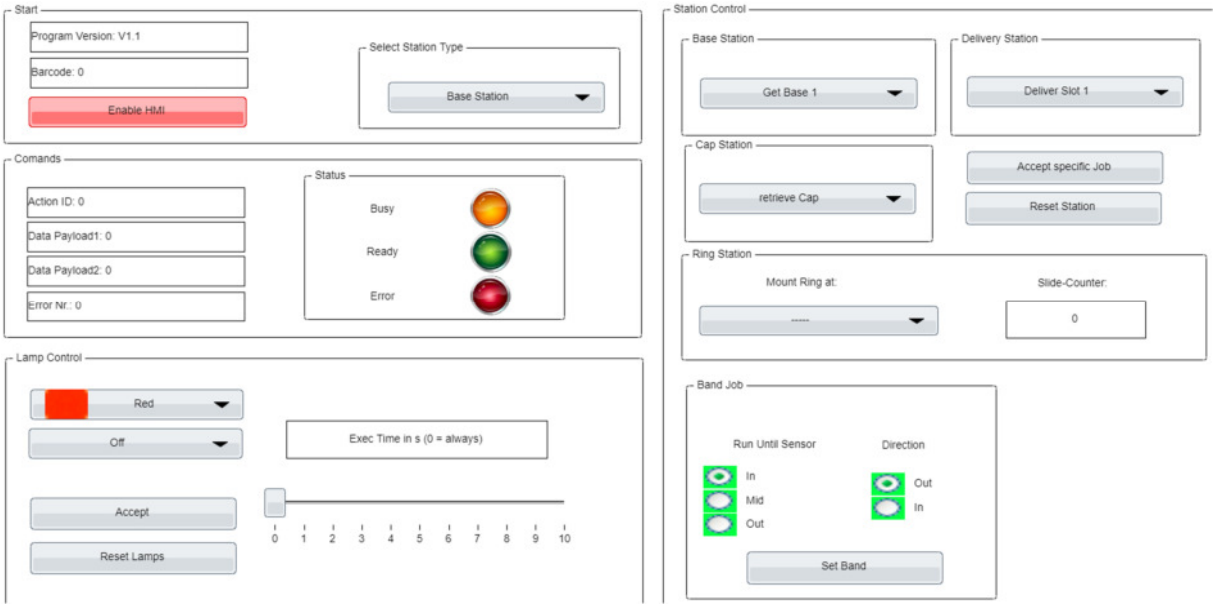


Abbildung 6 Webvisu des alten Programmes

6.7 Analyse Error Handling

Für die Objekte Cylinder, Band und Vakuum existiert ein Errorhandling. Tritt ein Error auf, wird ein Error Byte in einer Error Aktion auf eine Error Nummer gesetzt. Dieses Byte wird in den Stationen POU's weiterverwendet.

In jedem Stationen POU(BS_Main, CS_Main..) wird aus dem benötigten Zylinder oder Band FB ein möglicher Fehler ausgelesen. Der Error Status(vqx_Err) und der Error Typ (vqb_Err) werden im POU Main ins OPC UA geschrieben, wo er von der Refbox ausgelesen werden kann.

```
// Error
vqb_Err := INT_TO_BYTE(Cylinder[1].vqi_ErrorNr + Cylinder[2].vqi_ErrorNr + Band.vqi_ErrorNbr);
vqx_Err := vqb_Err <> 0;
```

Abbildung 7 Error Handling Vakuum altes Programm

Die Fehlerzeit kann in der globalen Variablenliste gesetzt werden. Da das Error Handling bis anhin nur angedacht war, war die Zeit auf 1000h gesetzt. Die Error Zeit ist für jeden FB dieselbe. Dies ist nicht optimal, da die Arbeitszeit eines Zylinders nicht gleich ist, wie die des Vakuums.

```
// Time till Error
MAX_ERROR_TIMEOUT: TIME := T#1000H;
```

Abbildung 8 Deklaration Error Zeit altes Programm

Ein Fehler am Vakuum würde mit einem Timer detektiert werden. Der Timer würde über den Befehl Vakuum erzeugen oder Überdruck erzeugen gestartet und ist nicht mit der Vakuum Kontrolle zusammengeschaltet.

```
Timer_Error(IN:=vsx_VacuumGenerate OR vsx_VacuumRemove,PT:=vit_TimeError,Q=>vsx_Error);
```

Eine bessere Lösung wäre es, den Vakuum Befehl mit der Vakuum Überwachung zu verknüpfen und dies zeitverzögert als Error weiter zu geben.

Bsp:

```
Timer_Error(IN:=vsx_VacuumGenerate AND vsx_VakuumControl,PT:=vit_TimeError,Q=>vsx_Error);
```

6.7.1 Mögliche Fehler Zylinder

Fehler	Const. Name	Datentyp	Wert
Fehler beim Einziehen	ErrorRetracting	INT	1
Fehler wenn eingezogen	ErrorRetracted	INT	2
Fehler beim Ausstossen	ErrorExtracting	INT	4
Fehler wenn ausgestossen	ErrorExtracted	INT	5

Tabelle 6 Zylinder mögliche Fehler

6.7.2 Mögliche Fehler Band

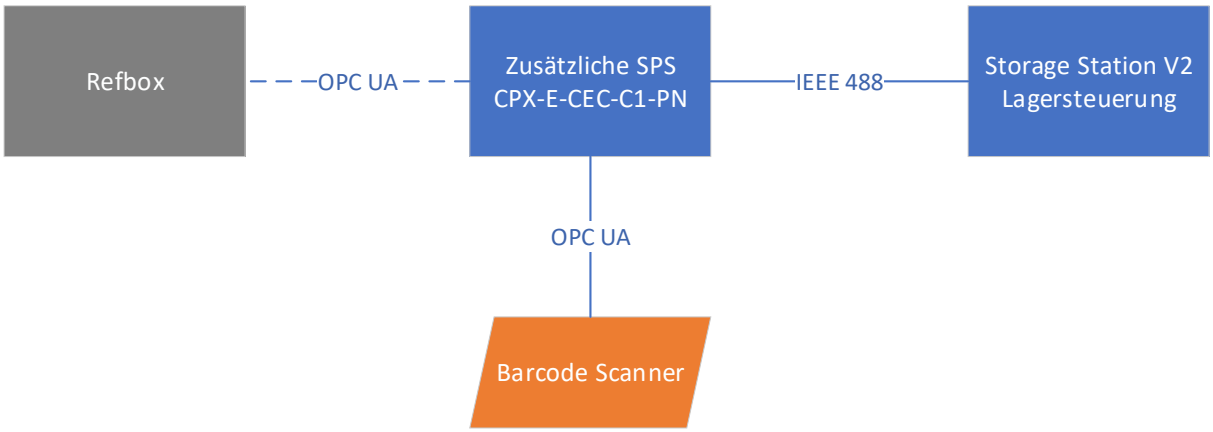
Beim Band wird kein spezieller Fehler ausgegeben. Die einzige Fehler Nummer, welche ausgegeben werden kann, ist ein INT mit dem Wert 8.

6.7.3 Mögliche Fehler Vakuum

Mögliche Fehler des Vakuums würden beim Vakuum erzeugen oder Überdruck erzeugen auftreten und unterschiedlich ausgegeben.

7 KONZIPIERUNG

7.1 KONZEPT IMPLEMENTIERUNG STORAGE STATION V2



Zur Lagerverwaltung und Steuerung der Ein-/Auslagerung wird die zusätzliche SPS verwendet. Diese sendet über ein IEEE 488 Kabel den Befehl einer Ein- oder Auslagerung an die bereits in der Storage Station V2 integrierte SPS (genannt Lagersteuerung) weiter. Die Lagersteuerung steuert anschliessend die Ein- oder Auslagerung sowie die Achsbewegungen.

Das Einzige, was die Lagersteuerung erhält sind die Zielkoordinaten. Der Rest der Logik wird auf der externen SPS umgesetzt. Für das wird eine Datenverwaltung mit einem dreidimensionalen Array erstellt. Diese speichert einzig die Barcode ID des Produktes. Die Auslösung sowie die ID wird über OPC UA jeweils von der Refbox bestellt.

Um das Konzept auf einer älteren Version der Storage Station anwenden zu können, muss die SPS der Station so programmiert werden, dass sie die Befehle der zusätzlichen SPS in Ein- oder Auslagerungen umsetzen kann.

7.2 KONZEPT NEUES PROGRAMM

Das neue Programm wird so weit wie möglich objektorientiert umgesetzt. Dazu wurden zwei Klassendiagramme (Anhang 2.5) erstellt mit zwei verschiedenen Varianten.

7.2.1 Variante 1

Hier wurde angedacht, dass jeder Stationstypen einen eigenen Baustein besitzt. Jeder dieser Stationstypen erbt von einem Baustein. Für jeden Stationstypen werden anschliessend nur die Bausteine instanziiert, welche diese Station auch besitzt. Die Umsetzung war jedoch noch nicht ganz ersichtlich, deshalb wurde noch eine zweite Variante erstellt.

7.2.2 Variante 2

Hier wird das Ganze in einem einzigen Baustein umgesetzt. Dieser beinhaltet für jeden Stationstypen eine Methode. Alle Bausteine wie z.B. «Carrier» oder «Belt» werden als VAR_INST in der jeweiligen Methode gespeichert.

Es wurden zwei unterschiedliche Klassendiagramme erstellt. Nach dem Entscheid an unserer ersten Sitzung vom 9. August 2019 mit Alain Rohr, wurde die Variante 2 umgesetzt.

7.2.3 Persistente Variablen

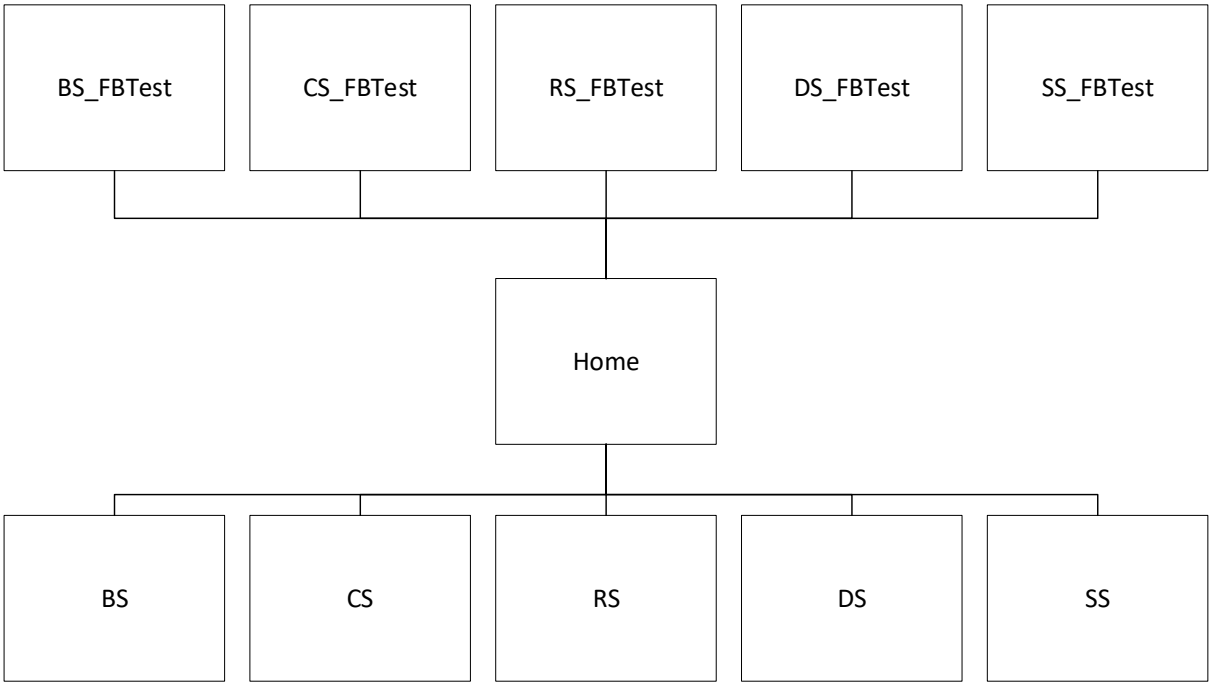
Um einige Werte wie z.B. Zeiten für Fehler nicht im Code ändern zu müssen, muss eine persistente Variablenliste erstellt werden. Durch die persistente Gestaltung kann bei einem Stationsfehler während dem Spiel ganz einfach die Station durch Aus- und wieder Einschalten neu gestartet werden, ohne dass es Probleme mit der Refbox gibt.

7.3 Konzept Barcodescanner

Für den Barcode Scanner wird ein eigener Baustein erstellt. Dieser speichert die letzten Scans und überprüft diese. Um diese Auswertung zu erstellen wird eine neue Variable auf dem OPC UA Server freigegeben. Auf diese Variable schreibt der Raspberry die erkannten Werte. Der Baustein liest diesen Wert aus, speichert ihn und schreibt den Wert auf dem OPC UA Server auf 0. Mit der Variabel «numOfCorrectCode» kann angegeben werden, wie oft der Code gesehen werden muss. Der Nummernbereich liegt zwischen 1 und 10. Wird der gleiche Code zweimal gesehen, wird er ins OPC UA Register der Refbox geschrieben.

7.4 KONZEPT VISUALISIERUNG

Idee der neuen Visualisierung ist es, dass es für jeden Stationstyp zwei Bilder gibt. Ein Bild dient zum Testen der einzelnen Komponenten wie Band und Zylinder. Das andere Bild dient zur Simulation und zum Testen von denselben Befehlen, welche auch von der Refbox versendet werden. Damit wird eine spezifischere und übersichtlichere Visualisierung erreicht. Auf einem Startbild wird die gewünschte Maschine und die gewünschte Operation gewählt, wodurch man automatisch auf das gewünschte Visualisierungsbild gelangt.



7.5 Konzept Error Handling

Für das Error Handling wird bei jeder Methode ein Byte zurückgegeben. Dieses Byte ist gemäss der folgenden Tabelle aufgebaut. Bit 0 wird auf true gesetzt, wenn die Methode fertig ist. Tritt ein Fehler auf, wird dieser mit den anderen sieben Bits als Zahl mitgeteilt.

Return Byte							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Error Code							Operation done

Tabelle 7 Return Byte Konzept

Tritt ein Error auf wird die State Machine, falls vorhanden, in den Reset-State gesetzt.

Grundsätzlich wird ein Fehler immer der Refbox gemeldet. Das Programm selbst setzt sich in die Ausgangsstellung zurück.

Zum Konzept des Error Handlings gehört auch die im Anhang enthaltene Prüfliste mit all den zu prüfenden Fehlerzuständen.

8 ZUSAMMENSTELLUNG DER ERGEBNISSE

Während der Konzeptphase wurde nicht alles angedacht und es gab ein paar weitere Herausforderungen. Zum Beispiel die Umsetzung der Visualisierung auf der Seite des Programms. Diese wurden im Verlauf der Arbeit noch umgesetzt.

8.1 HAUPTPROGRAMM

Während der Konzeptphase wurden zwei Varianten diskutiert. Eine, in welcher alle Stationen als Methoden in einer Klasse sind (Variante 2). In der anderen Variante (Variante 1) ist jeder Stationstyp eine Klasse. Nach der Konzeptphase entschied man sich auf Grund klarerer Umsetzungsvorstellung für die Variante 2, welche bis zu einer funktionierenden Version umgesetzt wurde. Während der Programmierungsphase der Variante 2 wurde das Wissen und Verständnis über das objektorientierte Programmieren grösser und man sah jetzt auch einen Lösungsweg zur Umsetzung der Version 1. Durch die Anwendung eines Strategiepattern, welches vom Experten erwähnt wurde, könnte das Programm noch viel einfacher gestaltet werden. Der Entscheid im Team war, dass die Programmseite der Visualisierung so umgesetzt wird. Also wurde für jeden Stationstypen ein Baustein erstellt, welcher vom Vaterbaustein «BasicStationVisu» erbt. Dazu wird das Interface «StationVisu» implementiert. Methoden und Variablen, die von allen Stationen gleich verwendet werden, wurden im Vaterbaustein «BasicStationVisu» gespeichert und programmiert.

Die Visualisierung war nun objektorientiert umgesetzt und es ergaben sich noch mehr Vorteile. Doch ein Problem gab es noch. Dieses wurde auch während der Konzeptphase gar nicht angedacht. Für die Visualisierung wurden alle Bausteine ein zweites Mal instanziiert, um über die Visualisierung darauf zuzugreifen. So mussten auch alle Ausgänge zweimal geschrieben werden. Dies kann zu Problemen führen, falls mal beide Bausteine gleichzeitig den gleichen Ausgang beschreiben. Würde jeder Baustein nur einmal verwendet werden, könnte das Überschreiben verhindert werden. So wurde der Entschluss gefasst, falls nach erfolgreicher Programmierung der Variante 2 noch Zeit bleibt, auch die 1. Variante umzusetzen.

Da wir ausreichend Zeit hatten, wurden beide Versionen erstellt. In der Abgabe befinden sich nun die beiden Programmversionen. Die Version V1.00 Alpha ist die Umsetzung der Variante 2, die Version V1.00 Beta die Umsetzung der Variante 1.

8.2 PARAMETRIERUNG DER ANLAGE

Wie bereits im Konzept angedacht, wollten wir Werte verändern, speichern und nach Neustart wiederverwenden können. So entstanden mehrere Datentypen, um alle Optionen der wichtigsten Bausteine übergeben zu können. Alle Parameter der Anlage können während dem Betrieb verändert werden. Im Detail werden die Parameter im nächsten Abschnitt erklärt. Wie die Benutzereingabe funktioniert wird in Abschnitt 8.8.3.2 erklärt.

8.3 OBJEKTE

Die Objekte wurden Objektbasiert programmiert. So kann z.B. die Klasse Cylinder für einen Zylinder mit oder ohne Endlagesensoren verwendet werden.

8.3.1 Cylinder

Im alten MPS Programm mussten beim Cylinder Aufruf sehr viele Variablen mitgegeben werden. Neu wird dies alles über Properties gelöst. Dies gestaltet das Programm übersichtlicher.

Die Klasse Cylinder sollte für jede Art von Zylinder verwendet werden können.

Die Parameter des Zylinders können über das UDT CylinderOptions parametriert werden.

Name	Daten Typ	Funktion
retractPos	BOOL	Angabe ob Endlagensensor bei ausgestossener Position
sensorRetract	BOOL	Angabe ob der Endlagensensor NC (true) oder NO (false) ist
extractPos	BOOL	Angabe ob Endlagensensor bei eingefahrener Position
sensorExtract	BOOL	Angabe ob der Endlagensensor NC (true) oder NO (false) ist
startPosInvertet	BOOL	Angabe ob die Startposition invertiert ist (true) oder nicht (false)
exTime	REAL	Aus- / Einfahrzeit, in welcher der Zylinder den Endlagensensor erreicht haben soll oder wenn kein Endlagensensor anliegt, der Zylinder aus-/ eingefahren ist
reTime	REAL	
exPresureTime	REAL	Zeit, in welcher der Endlagensensor bei einem ex- oder retracting verlassen sein sollte, wenn ein Druck anliegt
rePresureTime	REAL	

Tabelle 8 Cylinder Parametrierung

8.3.2 Vacuum

Der einzige Parameter, welcher beim Vakuum gesetzt werden kann, ist die Zeit, in welcher ein Vakuum am Vakuum Sensor erkannt werden soll, wenn das Vakuum Gas einzieht.

8.3.3 Belt

Die Klasse Belt sollte für jede Art von Band verwendet werden können. So funktioniert sie bei der DS, welche kein Motorsteuergerät besitzt, wie auch bei allen anderen Stationen, welche eines besitzen.

Die Parameter des Zylinders können über das UDT BeltOptions parametriert werden.

Name	Daten Typ	Funktion
barcodeOut	BOOL	Angabe ob sich der Barcode Check am Ein- und/oder Ausgang befindet
barcodeIn	BOOL	
chuechleOut	BOOL	Angabe ob der Chuechle Ablauf am Ein- und/oder Ausgang gemacht werden soll
chuechleIn	BOOL	
posOut	BOOL	Angabe welche Anfahrpositionen/Sensoren es gibt
posMid	BOOL	
posIn	BOOL	
sensorInTyp	BOOL	Angabe ob der Sensor der Position NC (true) oder NO (false) ist
sensorMidTyp	BOOL	
sensorOutTyp	BOOL	
errorTime	REAL	Die maximale Zeit, in welcher der Sensor dir gewünschten Position erreichen sollte
waitTimeMid	REAL	Zeitverzögerung der Bandausschaltung in der Mittelposition, um z.B. den Endanschlag des Gates zu erreichen
waitTimeDir	REAL	Zeit welche vor zwischen der Richtungsänderung des Bandes in Form von einem Halt liegen soll -> zur Schonung der physischen Komponenten
checkTimeBC	REAL	Dauer, welche das Band zum Checken des Barcodes Stillstehen soll, bevor es in einen nächsten Checkversuch oder einen Error läuft
directionIn	BOOL	Mögliche Fahrtrichtungen des Bandes -> zur Überprüfung von Bandbefehlen
directionOut	BOOL	
nrOfChecksBC	USINT	Anzahl von Barcode Check Versuchen, welche vor einem Error unternommen werden sollen

Tabelle 9 Belt Parametrierung

8.4 Lösung des Barcode Problems

Im Ablauf der Aufträge wird neu an der Barcode Check Stelle auf einen erkannten Barcode gewartet. Wird kein Barcode erkannt, bewegt sich das Band vor und zurück. Die Vor- und Zurück Bewegung läuft unter der Methode chuechle ab, welche im Kapitel 8.7.18.7.1 weiter erläutert wird. Dieser Vorgang wird so oft wiederholt, bis der Barcode erkannt wird (nächster Schritt) oder der parametrierte Wiederholungswert erreicht wird (Error).

8.5 Storage Station

8.5.1 Ansteuerung der Storage Station V2

Die SS V2 wird mit einer zusätzlichen Steuerung versehen, auf welche dasselbe Programm geladen wird, wie auf den anderen MPS Stationen.

Die Ansteuerung der V2 wurde nach den online zur Verfügung stehenden Ablauf Diagrammen der Kommunikation und Auftragsübergabe programmiert. Es wurde ein

Mockup erstellt, mit welchem man eine Storage Station V2 Kommunikation virtuell simulieren kann. Eine Funktion mit einer reellen Storage Station V2 kann leider nicht garantiert werden, da während der Diplomarbeit keine solche Station zur Verfügung stand.

8.5.2 Lagerverwaltung

Das Programm der zusätzlichen Steuerung wurde mit einer Lagerverwaltung versehen. Die Lagerverwaltung ermöglicht es, dass weder die Teams noch die Refbox die genaue Position der eingelagerten Produkte wissen müssen. Das Team gibt somit nur eine Produkte ID an die Refbox weiter. Die Refbox leitet diese ID über den üblichen OPC UA Weg weiter an die SS wo das Produkt aus- oder eingelagert wird.

8.6 Wichtige DataUserTypes

In diesem Abschnitt werden die wichtigsten DataUserTypes (DUT) erklärt. Weitere DUT's werden in den Kapiteln der verwendeten Orte erläutert.

8.6.1 MethodReturn

Ein Rückgabewert aus den Methoden war unbedingt nötig. Dieser sollte jedoch auch verschiedene Nachrichten zurückgeben können. So etwa einen Fehler im Ablauf oder dass der Ablauf erfolgreich endete. So entstand das DUT MethodReturn aus einem Gespräch mit Alain Rohr, welches sich über die ganze Programmierungsphase bewies.

Diese Datenstruktur ermöglicht die einfache Weitergabe von Done's und Error's, sowie derer Informationen, welche von untergeordneter Methode an übergeordnete Methoden weitergegeben werden können.

Name	Daten Typ	Funktion
error	BYTE	Übergibt den Error Typ anhand eines Bytes
onError	BOOL	Gibt zurück ob sich ein Error ereignet hat
done	BOOL	Gibt zurück ob die Methode/Aufgabe abgeschlossen ist
name	STRING	Gibt den Namen des Objekts/Ortes des Fehler Ereignisses zurück

Tabelle 10 Datentyp MethodReturn

8.7 Spezielle Methoden

8.7.1 Chuechle

Chuechle (Chüechlä) kommt aus dem Berndeutschen. Es heisst so viel wie Backen, wobei sich das Ein- und Ausfahren des Bandes vom einschieben des Kuchens in den Backofen und das wieder Herausnehmen ableiten lässt.

Chuechle ist eine private Methode der Klasse Belt. Sie wird in zwei Fällen verwendet.

1. Das Band fährt gegen In/Out Richtung In/Out. Damit das Teil schön am Ende des Bandes zu stehen kommt braucht es die Methode chuechle.
2. Der Barcode soll gecheckt werden. Es wird nach abgelaufener Check Zeit kein Code erkannt. Die Methode chuechle wird aufgerufen um das Teil neu zu Positionieren.

Grundprinzip:
Das Band fährt bis die fallende Flanke des Sensors kommt und wechselt dann die Bandrichtung. Das Band fährt in die andere Richtung weiter, bis wieder die positive Flanke des Sensors kommt und stoppt das Band endgültig. Die Methode chuechle ist done.

8.8 Visualisierung

Das Konzept der Visualisierung (Visu) konnte gut umgesetzt werden. Es gibt für jede Stationsart zwei Bilder. Ein Bild dient zur Konfigurierung und zum Testen der einzelnen Objekte wie Zylinder oder Vakuum. Das andere Bild dient zum Testen der OPC UA Kommandos, sprich der Station wie sie auch im Spiel von der Refbox angesteuert wird.

Beim Aufbau der Visu Oberfläche wurde auf Benutzerfreundlichkeit geachtet. Nach erfolgreichem Aufbau der Verbindung, kommt man zum Home Bildschirm. Ein Klick auf den Enable Button (wird noch erläutert) und man befindet sich bereits im Stationshauptbild. Nur ein Klick mehr und man befindet sich im Konfigurationsbereich. Es braucht nur wenige Klicks, um sein Ziel zu erreichen.

Um Zeit zu sparen und einen einheitlichen Look zu generieren wurden Elemente geschaffen, welche für mehrere Bilder verwendet werden können.

Auch bei der Wahl der Icons wurden nur Icons derselben Website verwendet, um auch dort ein einheitliches Bild zu bewahren.

8.8.1 Home Screen

Der Home-Screen startet beim Systemstart automatisch. Dieser sieht wie folgt aus:

Abbildung 9 Visu Home Screen

Der Home Screen ist sehr simpel aufgebaut. Zentral liegt ein Button, durch welchen man auf die Seite der aktuell gewählten Station gelangt. Rechts sieht man ein Zahnrad. Wird dieses angewählt, öffnet sich ein Dialogfenster, durch welches man auf eines der zehn anderen Stationsbilder gelangt.

8.8.2 Task Bar

Unten befindet sich die Task Bar, welche man in allen Bildern findet.

Sie wird folgend von links nach rechts erläutert.

Home-Button. Durch einen Klick auf diesen Button gelangt man zu jeder Zeit zurück auf den Home Screen.

Enable Button. Durch ihn wird einerseits die Visu Benutzung freigegeben, andererseits wechselt man direkt auf die Stationsseite, wenn eine Station gewählt ist.

Angabe- und Änderungsfeld der Station. Wie der Name schon sagt, kann damit der Stationstyp gewählt werden. Die gewählte Station wird hier ausgegeben.

Error Feld. Ereignet sich ein Fehler auf der Station oder wird ein falscher Befehl geschickt, wird dieser hier ausgegeben.

Versionsfeld. Hier ist die momentan geladene Programmversion ersichtlich.

8.8.3 Konfigurationsbild

Unten sehen wir das Konfigurationsbild der Ringstation. Wie auch alle anderen Konfigurationsbilder besitzt auch dieses die drei Lampentestfelder und das Bandtestfeld. Mit dem Button Rechts unter dem hftm Logo können die Standartparameter geladen werden. Falls der physische Aufbau einer Station einen Andere Konfiguration verlangt, können diese Anpassungen einfach über das Anwählen der Zahnräder geändert werden. Mit dem Pfeil oben links gelangt man zurück in das Hauptbild der Station.

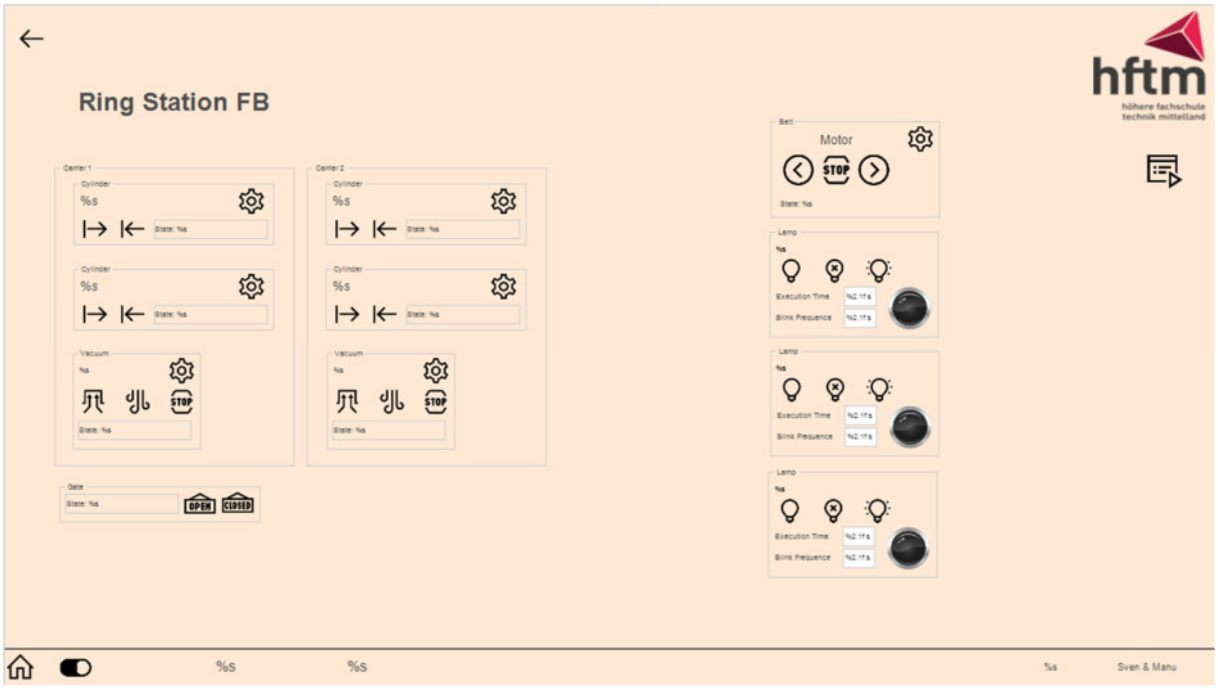


Abbildung 10 Visu Konfigurationsbild RS

8.8.3.1 Testen von einzelnen Objekten

Die Visu wurde modular aufgebaut. Es wurden Grundfenster erstellt, welche sich einfach und mehrmals in die verschiedenen Hauptfenster einbinden lassen. Beispiel für ein solches Grundfenster ist das Testelement des Zylinders. Dies macht die Visualisierung sehr Objektorientiert und flexibel. Einige Elemente wurden direkt in die Hauptbilder eingebaut, weil diese Elemente nicht mehrmals verwendet werden.

Abbildung 11 Visu Cylinder

8.8.3.2 Parameteränderung von Objekten

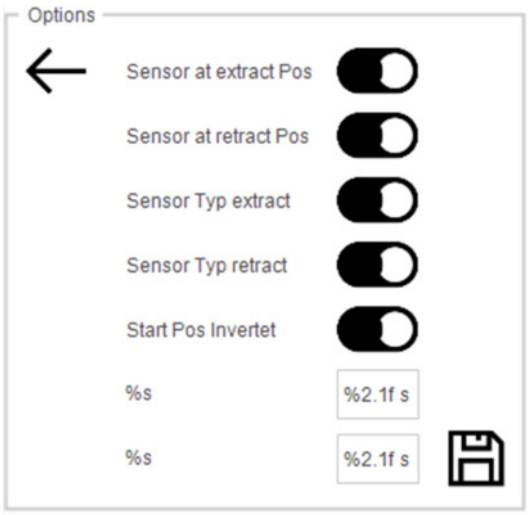


Abbildung 12 Visu Optionen Cylinder

Die Parameteränderung der verschiedenen Objekte kann via Konfigurationsbild und dem Parameterbutton (Zahnrad) des gewünschten Objekterahmens erreicht werden. Nach dem wählen des Parameterbutton öffnet sich ein Dialogfenster, in welchem man zum Beispiel die Art der Sensoren (NC od. NO) wählen kann oder ob der Zylinder überhaupt einen Endlagensensor hat oder nicht. Speicher und Schliessen der Dialogfenster geschieht in allen Dialogfenstern über dieselben Symbole welche selbsterklärend sind. So dient der Pfeil nach links zum Schliessen und die Diskette zum Speichern, wodurch das Dialogfenster gleichzeitig auch geschlossen wird.

8.8.4 Hauptbild Stationen

Unten ist das Hauptbild der Station BS ersichtlich. Auch hier wurde mit wiederverwendbaren Elementen gearbeitet. So ist nur das Feld Slot kein fester Bestandteil der Hauptbilder. Mit diesem Feld kann ein Ausstoss Befehl für den gewünschten Base Slot gesendet werden. Alle Befehle welche über die Hauptbilder gegeben werden, werden an das OPC UA Register übergeben.

Rechts unter dem hftm Logo sind zwei Buttons ersichtlich. Mit dem Button der wie eine Uhr aussieht kann ein Reset Befehl gesendet werden. Egal ob die Station zurzeit im Arbeits oder im Error Modus liegt, wird sie zurückgesetzt. Der Werkzeug Button dient zum Erreichen des Konfigurationsbildes.

Alle weiteren Elemente sind selbsterklärend.

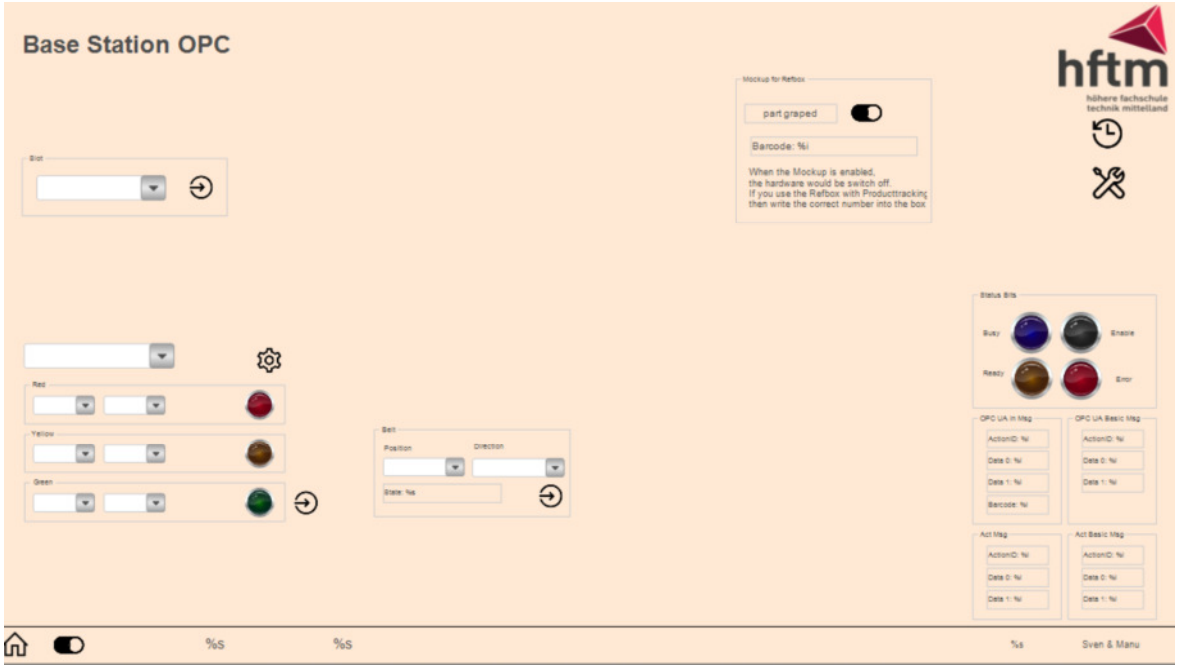


Abbildung 13 Visu Hauptbild BS

8.9 Error Handling

Mit dem neuen Error Handling können nahezu alle Fehler erkannt und ausgegeben werden. Ereignet sich während dem Betrieb ein Fehler oder wird ein falscher Auftrag gesendet, wird eine Fehlermeldung an die Refbox in das OPC UA Register geschrieben. Nach jedem Fehler ist ein Reset über das OPC UA Register nötig.

In der Webvisualisierung der Stationen werden die Fehler auch angezeigt. Es ist ersichtlich bei welchem Objekt und während welcher Aktion sich der Fehler ereignet hat. Dies kann bei der Fehlersuche viele Vorteile bringen.

Folgend werden mögliche Fehler der verschiedenen Objekte beschrieben.

8.9.1.1 Cylinder

- Der Zylinder verlässt den Endlagensensor beim Aus-/Einfahren nicht in einer gewissen Zeit.
  - Fehler: kein Druck (no pressure)
- Der Zylinder erreicht den Endlagesensor während einer Bewegung nicht in der vorgegebenen Zeit
  - Fehler: Fehler während Ein-/Ausfahren (error while re-/extracting)

8.9.1.2 Vacuum

- Das Vakuum soll erzeugt werden. Während der vorgegebenen Zeit wird jedoch kein Vakuum vom Sensor erkannt
  - Fehler: Es konnte kein Vakuum erzeugt werden (got no vacuum)
- Ein Vakuum wurde erkannt doch der Sensor erkennt keines mehr
  - Fehler: Das Vakuum ist abgefallen (vacuum lost)

8.9.1.3 Belt

- Das Band dreht in die vorgegebene Richtung, es erreicht jedoch nichts den Sensor
  - Fehler: Kein Bauteil hat den Sensor erreicht

8.9.1.4 Shelf/ Storage Station

- Es wird Ein- oder Auslagerung eines Teiles gewünscht, welches keinen Platz mehr hat oder nicht im Lager verfügbar ist.
  - Fehler: Bauteil im Lager nicht vorhanden (part not found in storage) od. Lager ist voll (storage full)

8.10 Verbesserungen

Durch das einbauen einer Mockup Funktion, welche zur Simulierung von MPS Stationen dient, wobei aber eine SPS benötigt wird, werden Simulierungen von ganzen Spielfeldern ohne eine echte MPS in Zukunft einfacher. Um eine solche Simulierung erschwinglicher zu machen, könnte das Programm in seiner Hardware so gestaltet werden, dass es auch auf einem Raspberry Pi mit aufgespieltem Codesys laufen würde.

8.11 Node-RED Simulation

Im Node-RED wurde zur Simulation der Station und des Produktionsablaufes ohne Robotinos ein kleines Programm aufgebaut. Man kann so die BS, RS und DS in Serie stellen. Durch den Start Button im Node-RED wird aus dem ausgewählten Slot der BS ein Teil ausgeworfen. Über die RS, in welcher die Base von beiden Carriern etwas aufgesetzt bekommt, Läuft das Produkt weiter in den im Node-RED gewählten Lieferslide der DS.

Eine Simulation mit dem Node-RED Programm testet auch die OPC UA Kommandos der Refbox, da die Simulation mit den genau gleichen Befehlen arbeitet .

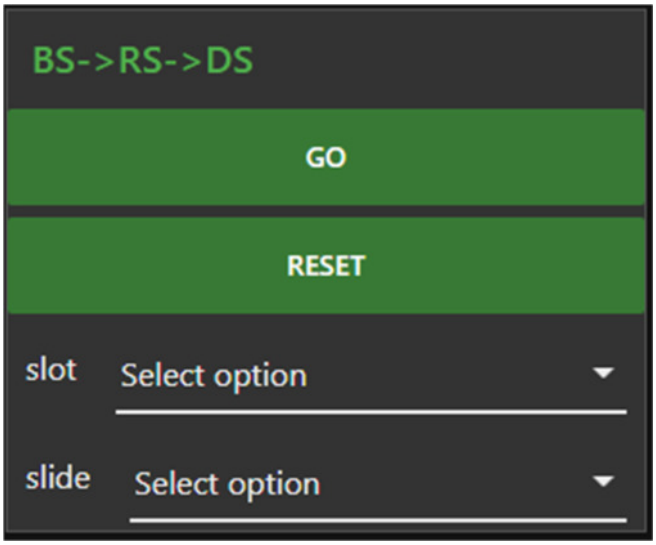


Abbildung 14 Node-RED Bedienpanel Simulation

9 SCHLUSSWORT

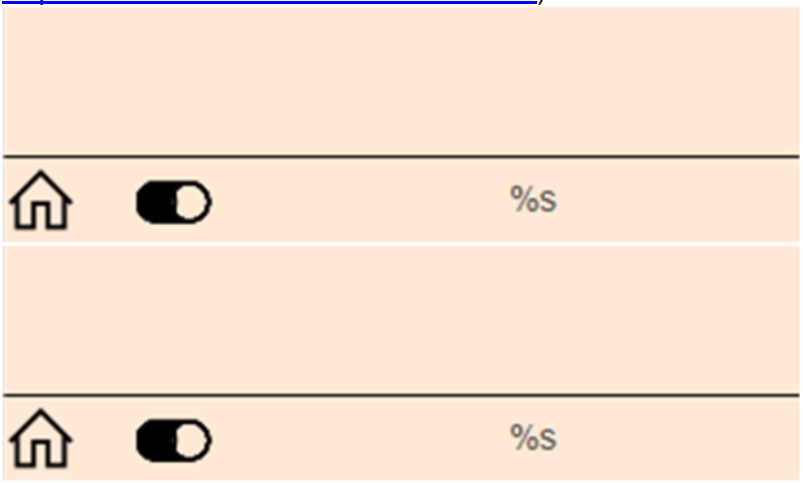
Die Diplomarbeit war noch einmal eine super Möglichkeit unsere Kenntnisse in der objektorientierten Programmierung von SPS Programmen mit der Programmiersprache strukturierter Text zu festigen und vertiefen. Oft trafen wir auf Schwierigkeiten wo einer allein keine Lösung sah, jedoch gemeinsam als Team fanden wir die richtige Lösung.

Durch unsere Vorgabe mit der Arbeit schon nach 4 Wochen fertig zu sein wurde der Zeitdruck nicht kleiner. Letztendlich reichte die Zeit dank längeren Arbeitstagen und genügend aktiv denkenden Pausen trotzdem für die Fertigstellung der praktischen Arbeit. Auch Toilettengänge wurden als inspirierende Tätigkeiten entdeckt! Der Technische Bericht war durch das schematische Einhalten der vorangehenden Schritte Analyse und Konzipierung schon fast von allein geschrieben. Das zu Papier bringen der erreichten Ergebnisse gestaltete sich dadurch nur noch als Formsache.

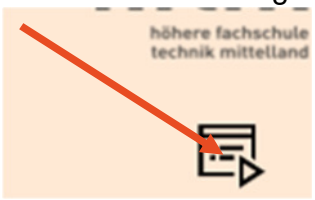
1 ANHANG

1.1 INBETRIEBNAHME

1. Starten Sie das Codesys und stellen Sie die Verbindung zur Station her.
2. Loggen Sie sich auf die Station ein und laden Sie dabei das aktuelle Programm
3. Erstellen Sie eine Bootapplikation
4. Wählen Sie in der Webvisualisierung (Bsp. http://192.168.2.24:8080/webvisu.htm) die Station aus



5. Laden Sie im Konfigurationsbild die Standartparameter



6. Die Station sollte nun bereit sein

1.2 Error Codes

Error Code				
Binär	**Dezimal**	**Beschreibung**	**Description**	**Ort**
1	1	Falsche Bestellung. Ausführung nicht möglich	Wrong Order. Not possible to execute	Global
10100	20	Fehler beim ausfahren	error while retracting	Cylinder
10110	22	Fehler im Zustand Ausgefahren Gründe: - Sensor verloren	error in position retracted reasons: - sensor lost	Cylinder
11000	24	Fehler beim Einfahren Gründe: -Aufruf zum Einfahren, wenn eingefahren -keine Endlage	error while extracting reasons: - no Endposition	Cylinder
11010	26	Fehler im Zustand eingefahren Gründe: -Aufruf zum Ausfahren, wenn ausgefahren -Sensor verloren	error in position extracted reasons: - send task extracting in position extracted - lost sensor	Cylinder
11100	28	Fehler bei Initialisierung Gründe: - Kein Druck	error during initialisation reasons: - No pressure	Cylinder
11110	30	Kein Druck	No pressure	Cylinder
101000	40	Es konnte kein Vakuum erzeugt werden	got no vacuum	Vacuum
101010	42	Das Vakuum ist abgefallen	vacuum lost	Vaccum
111100	60	Richtung nicht definiert	Direction not defined	Belt
111101	61	Position nicht definiert	Position not defined	Belt
111110	62	Kein Bauteil hat den Sensor erreicht	no part has reached the sensor	Belt
111111	63	Position nicht möglich	no possible Position	Belt
1000000	64	Richtung nach Eingang nicht möglich	direction to in not Posible	Belt
1000001	65	Richtung nach Ausgang nicht möglich	direction to out not Posible	Belt

1010000	80	Es wurde kein Barcode in der vorgegebenen Zeit erkannt	no barcode in time	Barcode
1100101	101	Kein Bauteil in Slot 1	No part at slot 1	BaseStation
1100110	102	Kein Bauteil in Slot 2	No part at slot 2	BaseStation
1100111	103	Kein Bauteil in Slot 3	No part at slot 3	BaseStation
1111000	120	Bauteil im Lager nicht vorhanden	part not found in Storage	StorageStation
1111001	121	Lager ist voll	storage full	StorageStation
10000010	130	Servos wurden noch nicht referenziert	Servos not referenced	StorageStation
10000011	131	Keine korrekte Bestellung	no correct Order	StorageStation
11001000	200	Keine gültige Station	invalid Station	Visualisation
11111111	255	Fehler nicht definiert	Error not definded	Global

1.3 TESTPROTOKOLLE

Testprotokoll Nr. 1.				
Für: BS	**Projekt: Diplomarbeit MPS 2019**	**Kurs: Diplomarbeit**	**Datum: 05/09/2019**	
Nr.	**Was/Objekt**	**Bemerkung**	**Anpassung**	**OK**
	Cylinders			☒
1.	Slot 1 extracting			☒
2.	Slot 1 retracting			☒
3.	Slot 1 Error extracting			☒
4.	Slot 1 Error retracting			☒
5.	Slot 1 Error no pressure			☒
6.	Slot 2 extracting			☒
7.	Slot 2 retracting			☒
8.	Slot 2 Error extracting			☒
9.	Slot 2 Error retracting			☒
10.	Slot 2 Error no pressure			☒
11.	Slot 3 extracting			☒
12.	Slot 3 retracting			☒
13.	Slot 3 Error extracting			☒
14.	Slot 3 Error retracting			☒

15.	Slot 3 Error no pressure			☒
	Belt			☒
16.	Belt turn to output, sensor output			☒
17.	Belt turn to input, sensor input			☒
18.	Belt Error no workpeace on sensor detected			☒
19.	**Barcode**			☒
20.	Red Light on, off, blink			☒
21.	Yellow Light on, off, blink			☒
22.	Green Light on, off, blink			☒
23.	**OPC UA**			☒
24.	Belt to output direction out			☒
25.	Belt to input direction in			☒
26.	Slot 1			☒
27.	Slot 2			☒
28.	Slot 3			☒
29.	Lights			☒
30.	Reset			☒

Testprotokoll Nr. 2.				
Für: CS	**Projekt: Diplomarbeit MPS 2019**	**Kurs: Diplomarbeit**	**Datum: 05/09/2019**	
Nr.	**Was/Objekt**	**Bemerkung**	**Anpassung**	**OK**
	Cylinders			☒
1.	CylinderH extracting			☒
2.	CylinderH retracting			☒
3.	CylinderH Error extracting			☒
4.	CylinderH Error retracting			☒
5.	CylinderH Error no pressure			☒
6.	CylinderH Error während Reset	Musste noch eingebaut werden		☒
7.	CylinderV extracting			☒
8.	CylinderV retracting			☒
9.	CylinderV Error retracting			☒
10.	CylinderV Error no pressure			☒
11.	CylinderV Error während Reset	Musste noch eingebaut werden		☒
	Vacuum			☒
12.	Vacuum push			☒

13.	Vacuum pull			☒
14.	Vacuum Error got no vacuum			☒
15.	Vacuum Error vacuum lost			☒
	Belt			☒
16.	Belt turn to output, sensor middle			☒
17.	Belt turn to output, sensor output			☒
18.	Belt turn to input, sensor middle	Ist keine sinnvolle Funktion	Wird neu überwacht und wenn ein Aufruf kommt als Fehler ausgegeben	☐
19.	Belt Error no workpeace on sensor detected			☒
	Seperator			☒
20.	invertiert			☒
21.	open			☒
22.	close			☒
23.	**Barcode**			☒
24.	Red Light on, off, blink			☒
25.	Yellow Light on, off, blink			☒
26.	Green Light on, off, blink			☒

	OPC UA			☒
27.	Belt to middle			☒
28.	Mount Cap			☒
29.	Retrieve Cap			☒
30.	Belt to out			☒
31.	Lights			☒
32.	Reset			☒

Testprotokoll Nr. 3.

Für: DS	**Projekt: Diplomarbeit MPS 2019**	**Kurs: Diplomarbeit**	**Datum: 05/09/2019**

Nr.	**Was/Objekt**	**Bemerkung**	**Anpassung**	**OK**
	Cylinders			☒
1.	Seperator 1 extracting			☒
2.	Seperator 1 retracting			☒
3.	Seperator 1 Error extracting			☒
4.	Seperator 1 Error retracting			☒
5.	Seperator 1 Error no pressure			☒
6.	Seperator 2 extracting			☒
7.	Seperator 2 retracting			☒

8.	Seperator 2 Error extracting			☒
9.	Seperator 2 Error retracting			☒
10.	Seperator 2 Error no pressure			☒
	Belt			☒
11.	Belt turn to output, sensor output	In der Visualisierung keine Fahrtrichtung gewählt werden, da die BS keinen Motor Controller zur Änderung der Richtung besitzt.		☐
12.	Belt Error no workpeace on sensor detected			☒
13.	**Barcode**			☒
14.	Red Light on, off, blink			☒
15.	Yellow Light on, off, blink			☒
16.	Green Light on, off, blink			☒
17.	**OPC UA**			☒
18.	Delivery slide 1			☒
19.	Delivery slide 2			☒
20.	Delivery slide 3			☒
21.	Lights			☒
22.	Reset			☒

Testprotokoll Nr. 4.				
Für: RS	**Projekt: Diplomarbeit MPS 2019**	**Kurs: Diplomarbeit**	**Datum: 05/09/2019**	
Nr.	**Was/Objekt**	**Bemerkung**	**Anpassung**	**OK**
	Cylinders			☒
1.	Cyl. H Carrier 1 extracting			☒
2.	Cyl. H Carrier 1 retracting			☒
3.	Cyl. H Carrier 1 Error extracting			☒
4.	Cyl. H Carrier 1 Error retracting			☒
5.	Cyl. H Carrier 1 Error no pressure			☒
6.	Cyl. V Carrier 1 extracting			☒
7.	Cyl. V Carrier 1 retracting			☒
8.	Cyl. V Carrier 1 Error extracting			☒
9.	Cyl. V Carrier 1 Error retracting			☒
10.	Cyl. V Carrier 1 Error no pressure			☒
11.	Cyl. H Carrier 2 extracting			☒

12.	Cyl. H Carrier 2 retracting			☒
13.	Cyl. H Carrier 2 Error extracting			☒
14.	Cyl. H Carrier 2 Error retracting			☒
15.	Cyl. H Carrier 2 Error no pressure			☒
16.	Cyl. V Carrier 2 extracting			☒
17.	Cyl. V Carrier 2 retracting			☒
18.	Cyl. V Carrier 2 Error extracting			☒
19.	Cyl. V Carrier 2 Error retracting			☒
20.	Cyl. V Carrier 2 Error no pressure			☒
	Vacuum			☒
21.	Vacuum 1 push			☒
22.	Vacuum 1 pull			☒
23.	Vacuum 1 Error got no vacuum			☒
24.	Vacuum 1 Error vacuum lost			☒
25.	Vacuum 2 push			☒

26.	Vacuum 2 pull			☒
27.	Vacuum 2 Error got no vacuum			☒
28.	Vacuum 2 Error vacuum lost			☒
	Belt			☒
29.	Belt Turn to output, sensor middle			☒
30.	Belt Turn to output, sensor output			☒
31.	Belt Error no workpeace on sensor detected			☒
	Gate			☒
32.	open			☒
33.	close			☒
34.	**Barcode**			☒
35.	Red Light on, off, blink			☒
36.	Yellow Light on, off, blink			☒
37.	Green Light on, off, blink			☒
38.	**OPC UA**			☒
39.	Belt drive to middle direction out			☒

40.	Belt drive to output direction out			☒
41.	Mount slot 1			☒
42.	Mount slot 2			☒
43.	Lights			☒
44.	Reset			☒

1.4 ARBEITSJOURNAL

Arbeitsjournal

Nr.	Datum	Person	Tagesziel	erreicht	Schwierigkeit	Lösung
Woche 1						
1.	05.08.19	Sven	• Dokumente vorbereiten • Vorlagen erstellen • Storage Station (SS) V2 analysieren • Konzept SS V2 erstellen	Ja Ja Ja Ja	Bei diesen Aufgaben sind keine Schwierigkeiten aufgetreten	-
2.	05.08.19	Manuel	• Zeitplanung erstellen • Analyse SS V2 erstellen • Konzept SS V2 erstellen	Ja Ja Ja	Zeitplanung ist schwer abzuschätzen, da noch ungewiss wegen SS V2	Genug Puffer am Ende

3.	06.08.19	Sven	• Bestehendes Programm verstehen • Mit der Konzipierung beginnen	Ja Ja	Es tauchten viele Spezialfälle auf, welche berücksichtigt werden müssen.	Jonas Jauslin wurde kontaktiert für: Barcode ->• wird nur einmal gesendet. • Momentanes Problem ist, dass sie teilweise nicht erkannt werden. Fehlerhandling ->• wenn die Endlage nicht erreicht wird, wird zurzeit kein Fehler ausgegeben.
4.	06.08.19	Manuel				
5.	07.08.19	Sven	• Weiterarbeiten an Analyse (Errorhandling abschliessen)	Ja	Es war nicht einfach zu verstehen ob und wie es funktioniert, weil das Errorhandling noch nie in Betrieb war.	
6.	07.08.19	Manuel	• Weiterarbeiten an Analyse (OPC UA abschliessen) • Barcode Teil analysieren	Ja Ja	Es war schwierig zu unterscheiden welche Teile Leichen aus der Modbus Zeit sind und welche es noch braucht.	Kurze Besprechung mit Jonas Jauslin
7.	08.08.19	Sven	• Visualisierung analysieren • Konzept Errorhandling erstellen • Visualisierung konzipieren • Testprogramm konzipieren	Ja Ja Ja Ja	Das Testprogramm wird auf Grund des Wunsches so vielseitig wie möglich zu machen ziemlich komplex. Es kann noch nicht alles so genau geplant werden.	Erste Schritte in der Konzipierung wurden gemacht. Während dem Programmieren werden die restlichen Probleme gelöst.
8.	08.08.19	Manuel				

9.	09.08.19	Sven	• 1.Sitzung mit Alain Rohr halten • Konzept Testen der Funktionen • Programmskelett erstellen • Erste Klassen bearbeiten (Cylinder, Vacuum)	Ja Nein Ja Ja	Für das Konzept des Programmes wurden zwei Klassendiagramme erstellt. Die Entscheidung welche Variante gewählt werden soll viel uns nicht leicht.	Alain Rohr wurde an der 1. Sitzung zu seiner Meinung über die Diagramme gefragt. Die 2. Variante wurde nun gewählt.
10.	09.08.19	Manuel				
Woche 2						
11.	12.08.19	Sven	• An Programm weiter Arbeiten -Cylinder abschliessen -Vacuum abschliessen -Carrier abschliessen	 Ja Ja Ja	Die FB_Init methode habe ich vorher noch nicht gekannt.	Manuel hat mit diese erklärt.
12.	12.08.19	Manuel	• Rückgabewerte von Methoden inkl. Fehler •	Ja	Nach dem eigentlichen Entscheid an der Sitzung vom 09.08.2019 haben wir uns nochmal Gedanken gemacht.	Entscheid mit Alain: Eigene Struktur für den Rückgabewert erstellen. Diese beinhaltet ein Fehlerbit, eine Fehlernummer als Byte, sowie ein Done Bit

13.	13.08.19	Sven	• Signalleuchte Programmieren • Carrier verbessern	Ja Ja	Der Bitzugriff kann nicht mit einer Variable durchgeführt werden (z.B. word.i)	Der Bitzugriff wurde nun über eine Bitmaske flexibel gestaltet.
14.	13.08.19	Manuel	• An Simulationsbaustein weiterarbeiten (Simulation RS) • Mit Bandbaustein beginnen	Ja Ja		
15.	14.08.19	Sven	• Programm auf RS testen, anpassen und verbessern	Ja		
16.	14.08.19	Manuel				
17.	15.08.19	Sven	• Programm testen • Programteile so gestalten, dass sie bei den anderen Maschinen eingesetzt werden können.	Ja Konnte nur zu 50% getestet werden	Der Carrier funktionierte bis er von Methoden in eine State Machine umgeschrieben wurde. Dies wurde gemacht, um eine Reset einzubauen.	Wir wechselten wieder auf die Methoden Variante und der Reset wurde anders realisiert.
18.	15.08.19	Manuel				
19.	16.08.19	Sven	• 2. Sitzung mit Alain Rohr • Arbeiten am Error Handling • Programm testen und anpassen	Ja Ja Ja		
20.	16.08.19	Manuel				

Woche 3						
21.	19.08.19	Sven	• Programm RS testen mit der Refbox • Mit anderen Maschinen beginnen • CS abschliessen • BS beginnen • DS beginnen	Ja Ja Ja Ja Ja	Die Verbindung konnte am Anfang nicht hergestellt werden.	In der OPC UA Kommunikation musste der UDT ProtoUnion, wiederverwendet werden.
22.	19.08.19	Manuel				
23.	20.08.19	Sven	• BS abschliessen • DS abschliessen • Expertensitzung	BS und DS konnte nicht beendet werden. Die Sitzung wurde gehalten.	Die persistenten Variablen aus einem Projekt wurden beim Laden zu Fehlern. Es sei zu wenig Speicherplatz vorhanden. Vom Experten wurde vorgeschlagen die Eigenschaften des Bandes und des Cylinders als UDT weiter zu geben.	Es werden nicht mehr alle Zylinder Zeiten einzeln gespeichert, sondern nur noch die Zeiten der maximal vier Zylindern einer Station. Die Werte werden nun auch durch das GUI änderbar. Dies hilft beim Implementieren der Eigenschaften und beim Testen.
24.	20.08.19	Manuel				
25.	21.08.19	Sven	• Die Visualisierung sauber gestalten	Zu 70%	Die Werte, welche im Dialogfenster übergeben wurden, wurden nicht in die Variablen überschrieben.	In den Einstellungen des Dialogfensters muss definiert werden, bei welchen Events die Variablen aktualisiert werden müssen.
26.	21.08.19	Manuel				

27.	22.08.19	Sven	• Die Visualisierung bis zum Testen abschliessen • SS abschliessen • Setzen der Standartparameter maschinenspezifisch und Objekt orientiert gestalten	95% Ja Ja	Die Visualisierung funktionierte nach dem zusammenführen der Programme nicht mehr.	Am nächsten Morgen funktionierte sie wieder. Wahrscheinlich wäre nur ein Kalt Reset nötig gewesen.
28.	22.08.19	Manuel				
29.	23.08.19	Sven	• Testen so vieler Anlagen wie möglich • Weiter verbessern des Programmes und Visualisierung	Ja Ja	Es wurden noch diverse nicht verknüpfte Sachen gefunden und es mussten auch noch einige Anpassungen vorgenommen werden.	Da das ganze Programm Objekt orientiert programmiert wurde waren Fehler und Anpassungen sehr schnell behoben.
30.	23.08.19	Manuel				
Woche 4						
31.	26.08.19	Sven	• Maschinen mit der Refbox testen	Nein, die Refbox funktionierte nicht. Test deshalb mit der Visu	Beim aktuellen Refboxprogramm wird der Aussfahrbefehl nicht ausgegeben	Das Problem wurde auch mit dem alten Programm vom Laptop von Jonas Jauslin aus belegt. Wir können so nicht alles Testen und auch nichts dafür tun. Alain Rohr wird eine neue Refbox aufsetzen, womit es möglicherweise funktionieren wird.
32.	26.08.19	Manuel				

33.	27.08.19	Sven	• Dokumentieren • Reset auch während einem Laufenden Auftrag ermöglichen	Ja Ja	Der errorTimer.IN der sich im BeltTurnUntil befindet wird bei einem Reset nicht auf FALSE gesetzt.	Die Deklaration des errorTimer wurde ins Belt(FB) verschoben und auch der enableTurn welcher auf errorTimer.IN schreibt. Somit kann aus der reset Methode auf das IN des errorTimers geschrieben werden.
34.	27.08.19	Manuel				
35.	28.08.19	Sven	• Dokumentieren • Konzeptversion 1 umsetzen • Simulationsprogramm Serie Stellung der Stationen in NodeRed erstellen	Ja Ja Ja	NodeRed Funktionen werden nur bei einem Eingangssignal aufgerufen. Wenn nun die Statemachine in der Funktion liegt und drei Eingangssignale praktisch gleichzeitig und zyklisch kommen, führt dies zu einem Chaos.	Es wurde eine Filterfunktion erstellt, die vor die Statemachine gesetzt wird und die Messages nur bei einer Signaländerung durchlassen.
36.	28.08.19	Manuel				
37.	29.08.19	Sven	• Präsentation der Stationen und Testprotokolle • Dokumentieren • Verbessern und verschönern des Programmes	Ja Ja Ja		
38.	29.08.19	Manuel				
39.	30.08.19	Sven	• Abschliessen der Dokumentation	99%		
40.	30.08.19	Manuel				

Woche 5						
41.	02.09.19	Sven	• Maschinen mit neuer Version Testen	Ja		
42.	03.09.19	Sven	• Korrekturen in Bericht anpassen	Ja		
43.	05.09.19	Sven	• Abgabeordner erstellen	Ja		

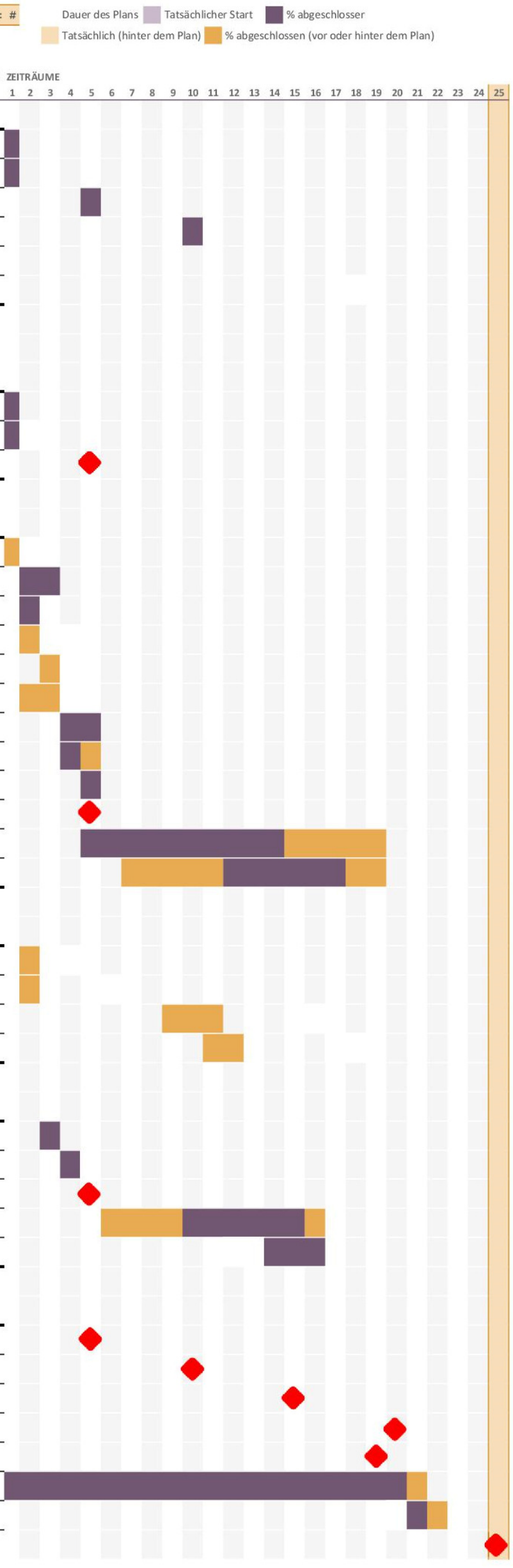
1.5 ZEITPLANUNG

Projektplan MPS Anlagen

Hervorgehobener Zeitraum: #

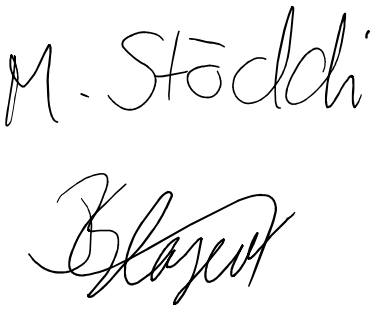
Dauer des Plans | Tatsächlicher Start | % abgeschlosser
Tatsächlich (hinter dem Plan) | % abgeschlossen (vor oder hinter dem Plan)

AKTIVITÄT	START DES PLANS	DAUER DES PLANS	TATSÄCHLICHER START	TATSÄCHLICHE DAUER	PROZENT ABGESCHLOSSEN
Administratives					
Planung erstellen	1	1	1	1	**100%**
Vorlagen vorbereiten	1	1	1	1	**100%**
Besprechung 1	5	1	5	1	**100%**
Besprechung 2	10	1	10	1	**100%**
Besprechung 3	15	1	0	0	**0%**
Besprechung 4	18	1	0	0	**0%**
Storage Station (SS) V2					
Analyse	1	1	1	1	100%
Konzept Implementierung	1	2	1	1	100%
Entscheidung weiteres Vorgehen					
Software					
Vorstellung vorhandenes (Jonas)	2	1	1	1	100%
Analyse OPC UA	2	2	2	2	100%
Analyse Hardware (MPS)	2	1	2	1	100%
Analyse aktuelle Software	3	2	2	1	100%
Analyse Errorhandling	4	1	3	1	100%
Konzept "neue" Software	4	2	2	2	100%
Konzept Errorhandling	4	2	4	2	100%
Konzept Testprogramm	4	1	4	2	100%
Konzept Testing	5	1	5	1	100%
Vorstellung Konzept					
Erstellen der Software	5	10	5	15	100%
Tests	12	6	7	13	100%
Produkttraking					
Analyse	4	1	2	1	100%
Konzept für Optimierung	6	1	2	1	100%
Umsetzung	15	3	9	3	100%
Tests mit Hardware	17	2	11	2	100%
Visualisierung					
Analyse	3	1	3	1	100%
Konzept	4	2	4	1	100%
Vorstellung Konzept					
Erstellung	10	6	6	11	100%
Tests	12	5	14	3	100%
Dokumentation/Abgabe					
Abgabe Arbeitsjounale Woche 1					
Abgabe Arbeitsjounale Woche 2					
Abgabe Arbeitsjounale Woche 3					
Abgabe Arbeitsjounale Woche 4					
Abnahme MPS Anlagen					
Dokumentation	1	20	1	21	100%
Überprüfung Dokumentation	20	2	21	2	100%
Abgabe Projekt					



1.6 SELBSTÄNDIGKEITSERKLÄRUNG

Wir bestätigen hiermit, dass wir die vorstehende Diplomarbeit selbstständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und sowohl wörtliche als auch sinngemäss verwendete Textteile, Grafiken oder Bilder kenntlich gemacht haben. Diese Arbeit ist in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt worden.

Ort, Datum	Unterschriften
Langnau, 5. September 2019	

2 VERZEICHNISSE

2.1 LITERATURVERZEICHNIS

Festo. (kein Datum). 8082795 Storing/Conveyor station.

Festo. (kein Datum). Betriebsanleitung. *8082795 Storing/Conveyor station, S.24 Abs.11.1*.

rs-online. (05. 08 19). Von https://ch.rs-online.com/web/p/products/2177893/?grossPrice=Y&cm_mmc=CH-PLA-DS3A-_-google-_-PLA_CH_DE_Kabel_Und_Dr%C3%A4hte-_-Computerkabel_Konfektioniert%7CParallel-Kabel_Konfektioniert-_-PRODUCT_GROUP&matchtype=&pla-407363529313&gclid=Cj0KCQjwp5_qBRDBAR abgerufen

2.2 ABBILDUNGSVERZEICHNIS

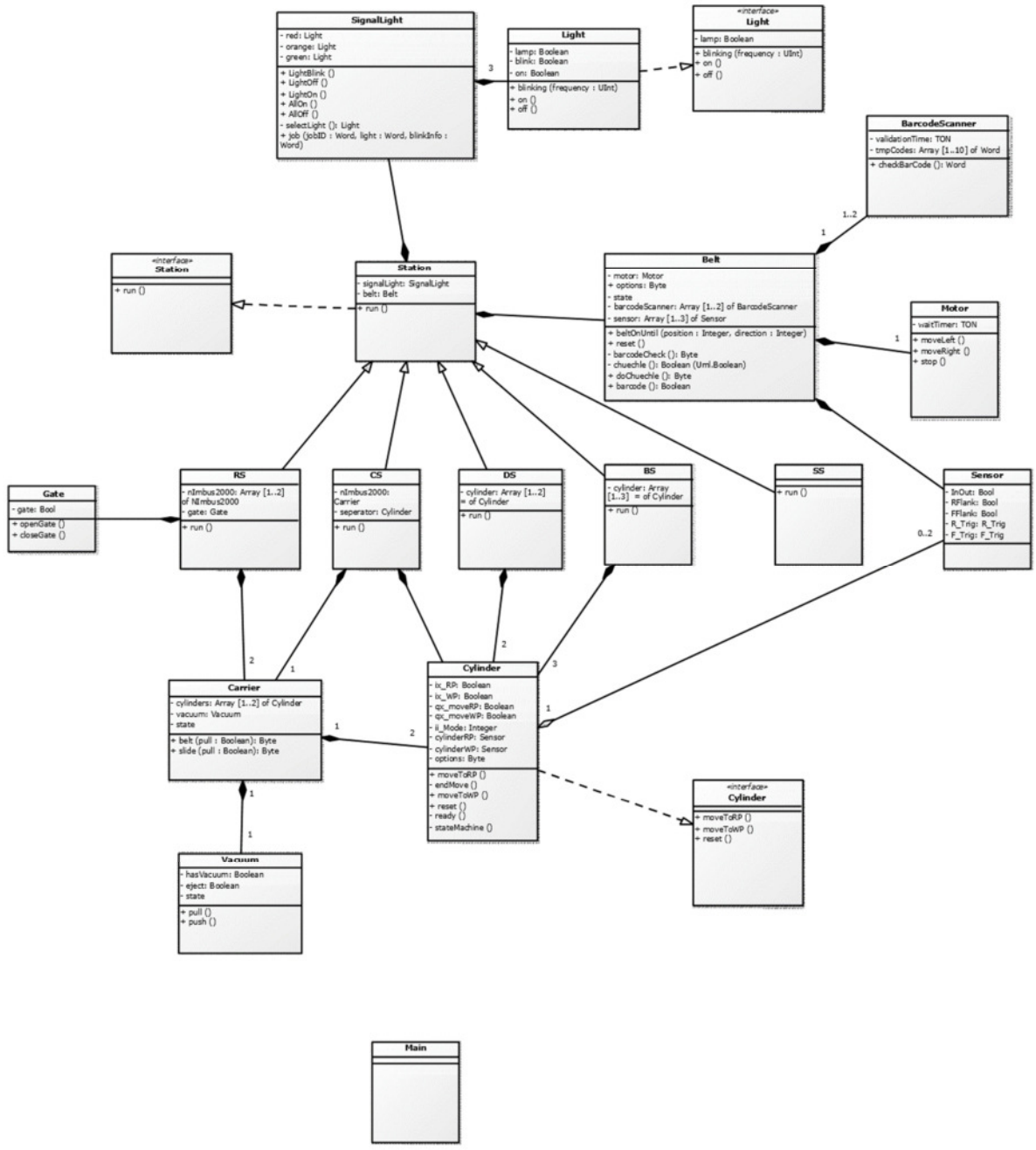
2.3 TABELLENVERZEICHNIS

2.4 STICHWORTVERZEICHNIS

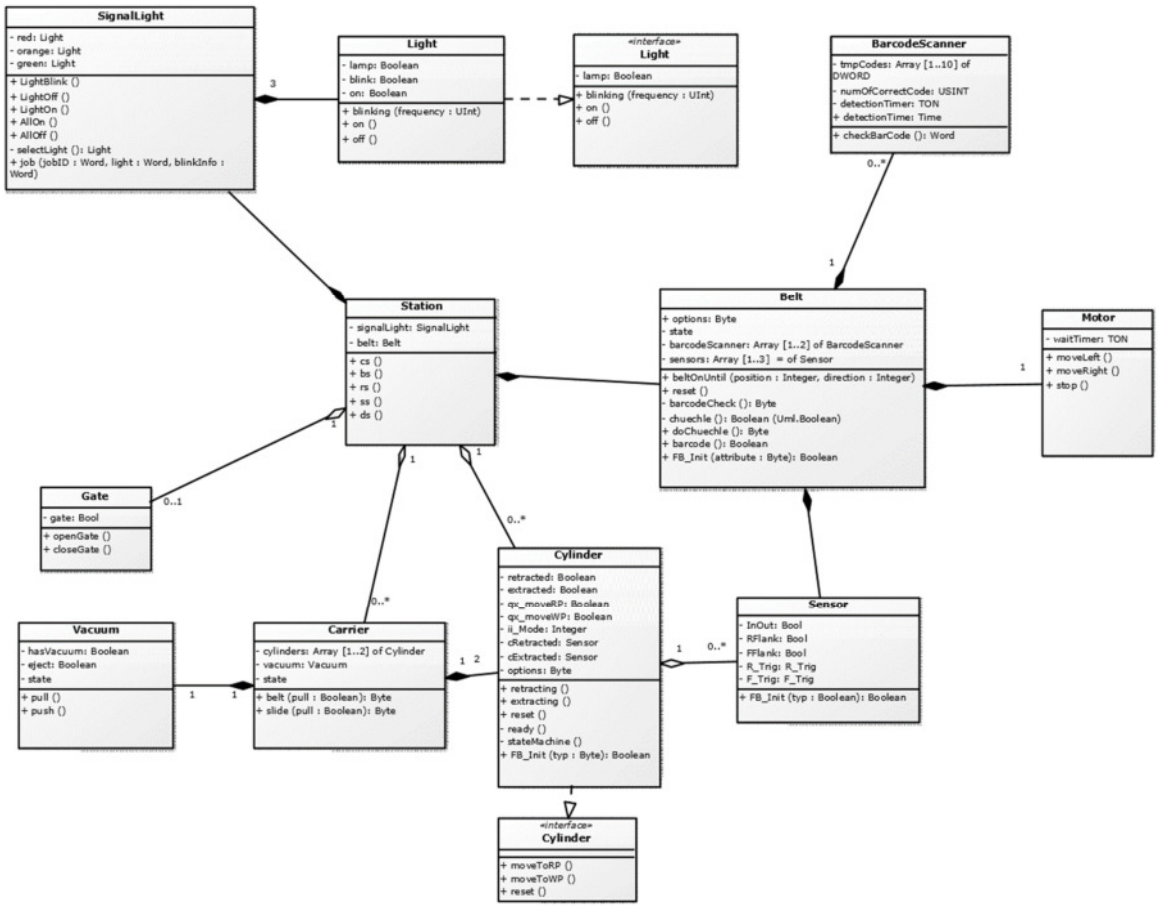
Wort	**Beschreibung**
OPC UA	**O**pen **P**latform **C**ommunications **U**nified **A**rchitecture wird für das Lesen und Schreiben von Variablen und Strukturen verwendet.
Persistente Variablen	Diese Variablen werden im File-System der SPS gespeichert und sind auch nach einem Stromausfall noch gespeichert. Um solche Variablen zu löschen, muss die SPS auf den Ursprung zurückgesetzt werden.
Refbox	Für den RoboCup wird die Refbox als zentrale Steuereinheit verwendet. Sie kommuniziert mit den einzelnen Stationen via OPC UA und sendet ihnen die Jobs.
SS / RS / CS / DS / BS	Abkürzungen für die einzelen Stationstypen (StorageStation, RingStation, CapStation, DeliveryStation, BaseStation)
Union	Eine Union wird verwendet, um mit zwei verschiedenen Variablen und/oder Datentypen den gleichen Speicherbereich zu nutzen.
VAR_INST	Variablen, die in einer Methode als VAR_INST deklariert werden, werden in der Instanz des Bausteins gespeichert und behalten ihren Wert bei erneutem aufrufen der Methode. Auf eine VAR_INST-Variable kann jedoch von einer anderen Methode im gleichen Baustein nicht zugegriffen werden.

2.5 KLASSENDIAGRAMME

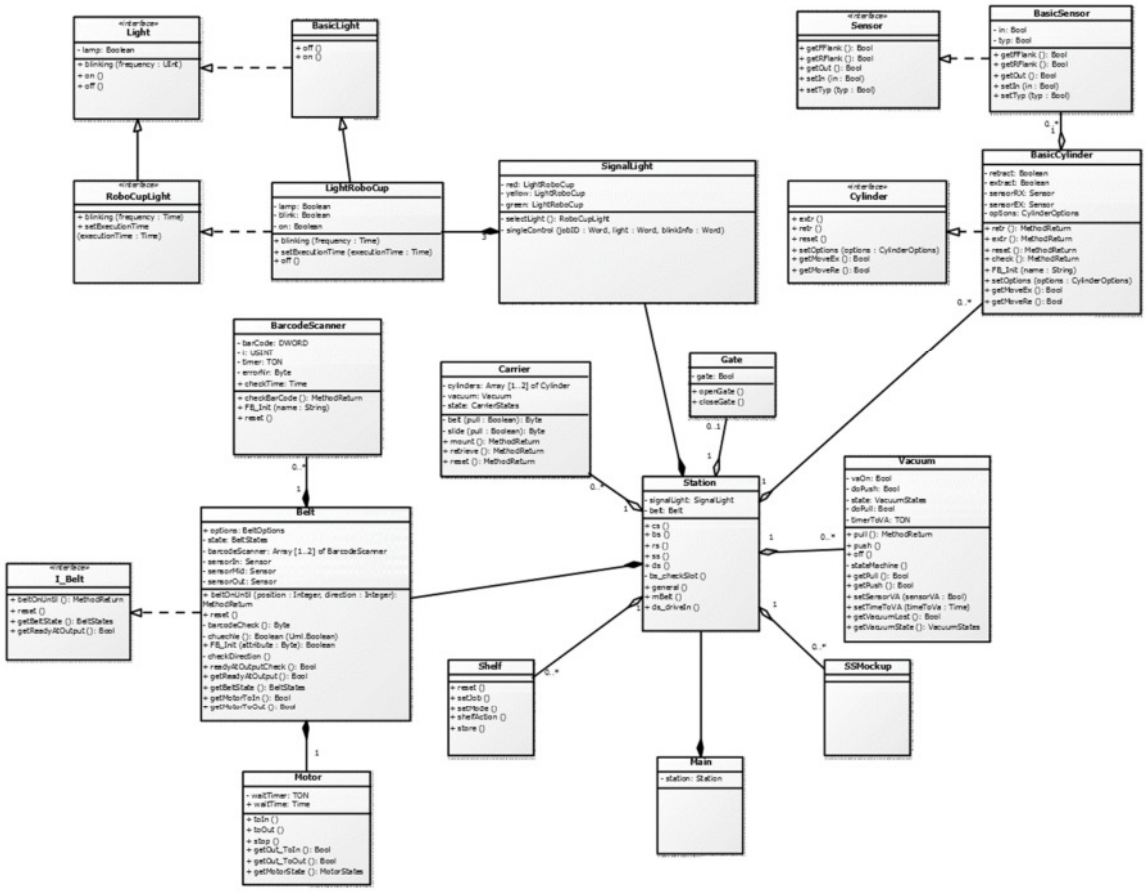
2.5.1 Konzept Version 1

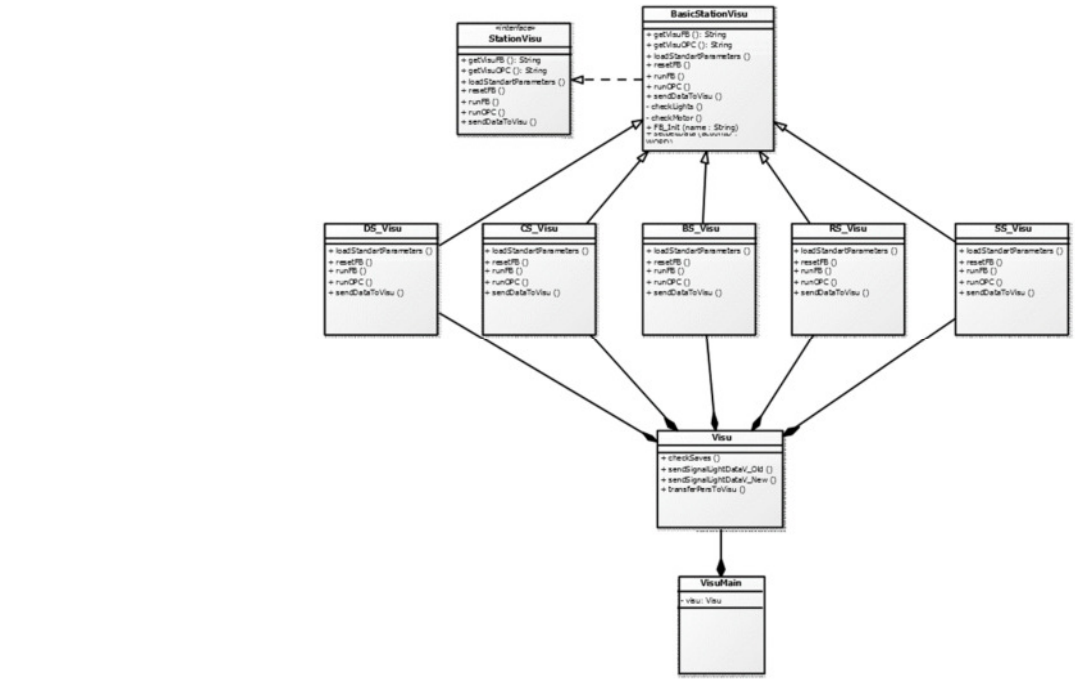


2.5.2 Konzept Version 2

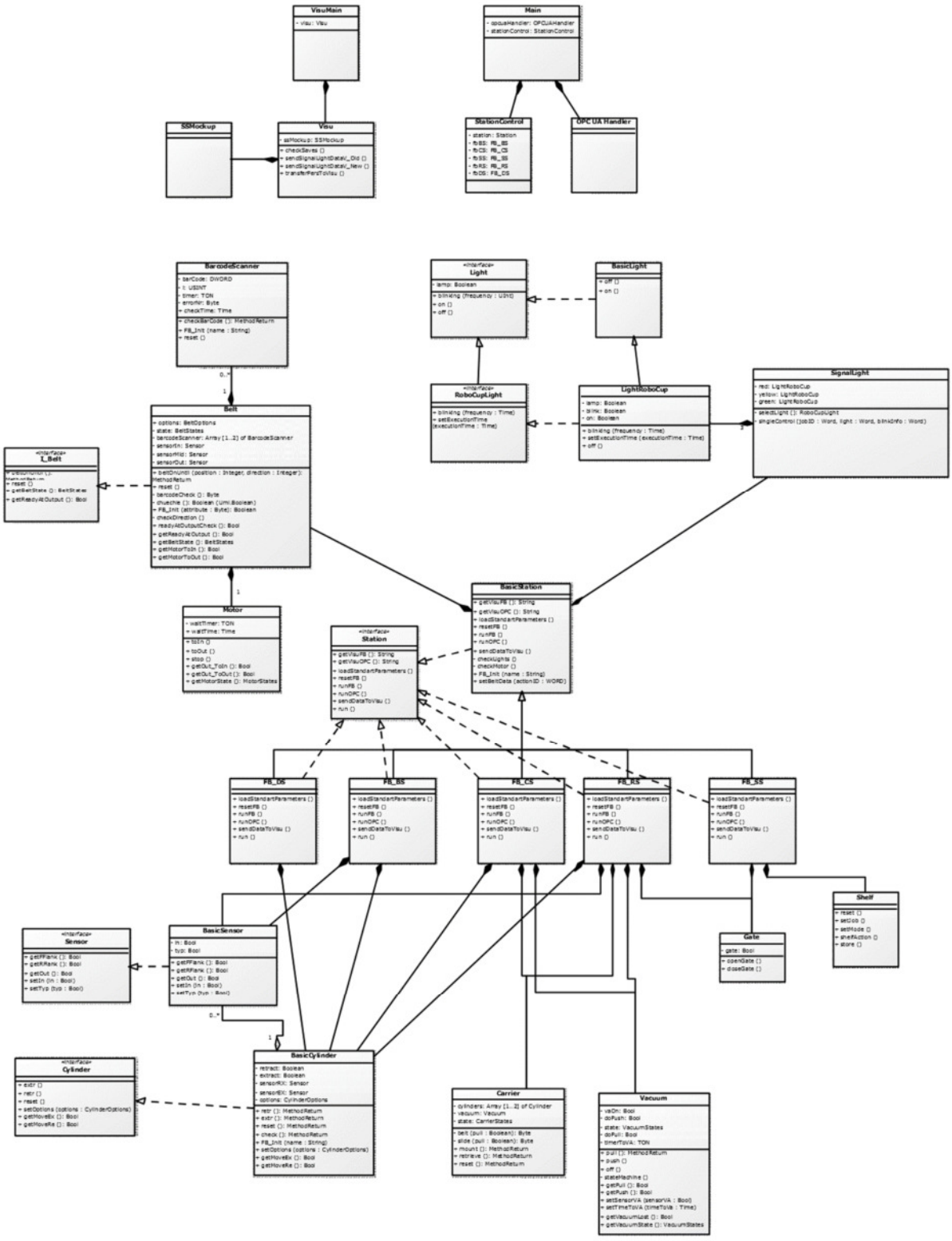


2.5.3 Version Alpha





2.5.4 Version Beta



2.6 OPC UA COMMANDS

Festo MPS Commands over OPC (from Raspi or Refbox)

MPS	Description	Action ID	Payload (Word 1 & 2)		8 Status Bits								Error Nr Byte	Corresponding RefboxMsg	Reading Registers		
		Word 0	Word 1	Word 2	7	6	5	4	3	2	1	0			Word 4	Word 5	Word 6
									ENABLE	ERR	RDY	BUSY			SlideCnt Ring-Station	BarCode	Version-control
All	NoJob	0															
All	MachineTyp	10	1=BS/2=RS/3=CS/4=DS/5=SS														
Lights	reset	20															
	red	21	0=off/1=on/2=blink	0=always/>0=time in s										SetSignalLight			
	yellow	22	0=off/1=on/2=blink	0=always/>0=time in s										SetSignalLight			
	green	23	0=off/1=on/2=blink	0=always/>0=time in s										SetSignalLight			
		25	00000ryg	00000bbb													
BS	Reset	100															
	GetBase	101	1-3											BSPushBase			
	BandOnUntil	102	1=in//3=out	1=dirToOut/2=dirToIn										MoveConveyerBelt			
RS	Reset	200															
	WaitForXBases	201	not implemented														
	BandOnUntil	202	1=in/2=mid/3=out	1=dirToOut/2=dirToIn										MoveConveyerBelt			
	MountRing	203	1-2											RSMountRing			
CS	Reset	300															
	BandOnUntil	302	1=in/2=mid/3=out	1=dirToOut/2=dirToIn										MoveConveyerBelt			
	Cap	301	1=retrieve/2=mount											CSTask			
DS	Reset	400															
	DeliverToSlot	401	1-3											DSActivateGate			
SS	Reset	500															
	BandOnUntil	502	3	1										SSTask			
	Store	501	1=store/2=unstore	ID of Part													

Enable Bit for starting a job
Busy Status while working

Basic Registers for general Jobs (MachineType, Lights)
Action Registers for machine specific Jobs (Band, Cylinders)