



University of Applied Sciences

**HOCHSCHULE
EMDEN·LEER**

***Industrie 4.0-fähige Kommunikation
zwischen einen Industrie-Roboter und einen
Flexibel Transport System.
(Implementierung und Validierung)***

Masterarbeit

Autor:	Faissal Hammouda
Matr.-Nr.:	7012301
Studiengang:	Master Maschinenbau
Erstprüfer:	Prof. Dr. rer. nat. Elmar Wings
Zweitprüfer:	Prof. Dr.-Ing. Armando Walter Colombo
Abgabedatum:	06.12.2021

Erklärung

Soweit meine Rechte berührt sind, erkläre ich mich einverstanden, dass die vorliegende Arbeit Angehörigen der Hochschule Emden/Leer für Studium / Lehre / Forschung uneingeschränkt zugänglich gemacht werden kann.

Faissal Hammouda

Emden, 06.12.2021

Eidestatliche Versicherung

Ich, der/die Unterzeichnende, erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Quellenangaben und Zitate sind richtig und vollständig wiedergegeben und in den jeweiligen Kapiteln und im Literaturverzeichnis wiedergegeben. Die vorliegende Arbeit wurde nicht in dieser oder einer ähnlichen Form ganz oder in Teilen zur Erlangung eines akademischen Abschlussgrades oder einer anderen Prüfungsleistung eingereicht.

Mir ist bekannt, dass falsche Angaben im Zusammenhang mit dieser Erklärung strafrechtlich verfolgt werden können.

Faissal Hammouda

Emden, 06.12.2019

Zusammenfassung

Im Rahmen dieser Masterarbeit wird die Kommunikation zwischen einem KUKA KR C4 Compact Roboter und einem Extended Transport System (XTS) der Firma Beckhoff unter Verwendung der OPC UA-Kommunikationstechnologie beschrieben. Dieses Projekt unterliegt dem Begriff Industrie 4.0. Das Projekt konzentriert sich auf die Implementierung eines OPC UA Servers in jeder der beiden digitalisierten Maschinen sowie auf die Implementierung eines OPC UA Python Clients in einem Computer, von dem die Daten, die zwischen den beiden Maschinen ausgetauscht werden sollen, abgerufen und gesendet werden. Nach erfolgreichen Kommunikationsstudien zwischen den beiden digitalisierten Maschinen wurde eine Applikation erstellt, die das Prinzip der Kommunikation zwischen den beiden Maschinen über das OPC UA-Kommunikationsprotokoll verkörpert. Im Rahmen dieses Projekts wird auch ein HMI spezifiziert und prototypisch umgesetzt, das es dem Benutzer ermöglicht, Steuerdaten einzugeben, die Applikation zu steuern und die verschiedenen Schritte der Applikation zu überwachen. Außerdem kann der Benutzer die verschiedenen Meldungen der Applikation lesen. Darüber hinaus wird eine Validierung der Projektergebnisse durchgeführt, bei der die Einhaltung der Anforderungen und das Erreichen der Projektziele überprüft werden.

Abstract

This master thesis describes the communication between a KUKA KR C4 Compact robot and an Extended Transport System (XTS) from Beckhoff using OPC UA communication technology. This project is subject to the concept of Industrie 4.0 and focuses on the implementation of an OPC UA server in each of the two digitalized machines, as well as the implementation of an OPC UA Python client in a computer from which the data to be exchanged between the two machines will be retrieved and sent. After successful communication studies between the two digitalized machines, an application was created that embodies the principle of communication between the two machines using the OPC UA communication protocol. This project also specifies and prototypes an HMI that allows the user to enter control data, control the application, and monitor the various steps of the application. In addition, the user will be able to read the various messages of the application. Furthermore, a validation of the project results will be carried out to verify that the requirements are met and that the project goals have been achieved.

Danksagung

An dieser Stelle würde ich gerne den vielen Personen danken, die mich bei diesem Projekt unterstützt haben.

Somit richte ich meinen Dank in erster Linie an Herrn Prof. Dr. rer. nat. Elmar Wings von der Hochschule Emden/Leer, der die Arbeit an diesem Projekt nicht nur ermöglicht, sondern bis zum Ende unterstützt hat. Auch an Herrn Dipl.-Ing. Thoma Peetz richte ich meinen besonderen Dank für die technische Unterstützung im Labor T65 der Hochschule Emden/Leer sowie an Herrn Prof. Dr.-Ing. Armando Walter Colombo von der Hochschule Emden/Leer, der mir bei der Erstellung der Masterarbeit kontinuierlich mit seiner Kompetenz zur Seite stand.

Zudem möchte ich auch meinen Kollegen im Labor danken, die mit mir durch die guten und schlechteren Phasen des Projekts gegangen sind und mich stets motiviert haben.

Inhaltsverzeichnis

Abbildungsverzeichnis	
Tabellenverzeichnis	
Abkürzungen	
1. Einführung	1
2. Stand der Technik.....	5
2.1. Industrie 4.0	5
2.2. Kommunikationstechnologie OPCUA.....	8
2.2.1. OPC Classic.....	8
2.2.2. OPC UA	9
2.3. Programmierungssprache Python.....	14
2.4. Human Machine Interface (HMI)	14
2.5. KUKA-Roboter.....	15
2.5.1. Roboter-Software	16
2.5.2. KUKAVARPROXY	17
2.5.3. Roboter-Kinematik	17
2.6. Extended Transport System (XTS)	22
2.6.1. XTS Spezifikationen	22
2.6.2. Automatisierungssoftware TwinCat 3	24
3. Spezifikation einer Kommunikation Infrastruktur.....	25
3.1. Lösung.....	25
3.2. Roboter-Kommunikation über OPCUA [86]	27
3.2.1. KUKAVARPROXY Message Format und KUKAVARPROXY.....	27
3.2.2. Funktionsweise von KUKAVARPROXY Message Format.....	28
3.2.3. Installation von KUKAVARPROXY in der KRC4 Compact Steuerung	28
3.3. XTS-Kommunikation über OPCUA	30
3.3.1. OPC UA Server TF6100	31
3.3.2. TwinCat OPC UA Server installieren	31
3.3.3. TwinCat OPC UA Server lizenzieren.....	35
3.3.4. TwinCat OPC UA Server konfigurieren	37
3.4. Python Applikation	41
3.4.1. Kommunikation der Python-Applikation mit dem KUKAVARPROXY-Server	42
3.4.2. Kommunikation der Python-Applikation mit dem OPC-UA-Server des XTS	44
4. Implementation der Kommunikation Infrastruktur	45
4.1. Überblick	45
4.2. Programmierung des KUKA Roboters	46
4.2.1. Robotersicherheit.....	46

4.2.2.	Datenkonfiguration	54
4.2.3.	Roboter-Programm	55
4.3.	Programmierung des XTS	56
4.3.1.	Lizenzen	57
4.3.2.	Bibliotheken.....	60
4.3.3.	XTS-Programm.....	73
4.4.	Python-Programmierung.....	83
4.4.1.	Variabel MoveKUKA:	83
4.4.2.	Variabel MoveXTS:	84
4.4.3.	Aktivitätsdiagramm: Python-Programmierung	85
5.	Human Machine Interface (HMI)	86
5.1.	Auswahl einer Lösung.....	86
5.2.	TE2000 TwinCAT 3 HMI Engineering	86
5.3.	Erstellung der HMI.....	88
5.3.1.	Main-Seite	88
5.3.2.	Configuration-Seite.....	94
5.3.3.	Messages-Seite	98
6.	Validierung der Projekt-Ergebnisse	100
6.1.	Erfüllung von Anforderungen.....	100
6.2.	Erreichen der Projekt-Ziele.....	100
7.	Zusammenfassung und Ausblicke	102
8.	Anhänge.....	105
9.	Literatur.....	126

Abbildungsverzeichnis

Abbildung 1: Abbildung: Darstellung der vier industriellen Revolutionen durch den Arbeitskreis Industrie 4.0 (Quelle DFKI 2011)	5
Abbildung 2: OPC-Struktur [82]	9
Abbildung 3: OPC UA-Schichtenarchitektur [17]	11
Abbildung 4: OPC UA Kommunikationsmodell [83]	11
Abbildung 5: RAMI 4.0 [19]	12
Abbildung 6: Industrieroboter [86]	16
Abbildung 7: Drehrichtung der Roboterachsen [27]	18
Abbildung 8: Grafik Roboterarbeitsbereich [84]	19
Abbildung 9: Bewegungsart PTP [29]	20
Abbildung 10: Bewegungsart LIN [29]	21
Abbildung 11: Bewegungsart CIRC [29]	21
Abbildung 12: XTS-Komponenten [31]	23
Abbildung 13: Erster Vorschlag für die Kommunikationsarchitektur	25
Abbildung 14: Kommunikationsarchitektur mit EL6695	26
Abbildung 15: Hardware-Kommunikation	27
Abbildung 16: Client-Server-Architektur	27
Abbildung 17: XTS – Computer Kommunikation	30
Abbildung 18: Akzeptieren der Lizenzvereinbarung	32
Abbildung 19: Benutzerdaten eingeben	32
Abbildung 20: Vollversion von TwinCat 3 installieren	33
Abbildung 21: Installation starten	33
Abbildung 22: Dialog bestätigen	34
Abbildung 23: Konfiguration beenden	34
Abbildung 24: Solution Explorer	35
Abbildung 25: Manage Licenses	35
Abbildung 26: Informationen zur Runtime	36
Abbildung 27: 7 Tage Testlizenz aktivieren	36
Abbildung 28: Konfiguration der SPS-Variablen für den OPC-UA-Zugang	37
Abbildung 29: Konfiguration der Symboldatei	37
Abbildung 30: TwinCAT Connectivity-Projekt	38
Abbildung 31: TwinCAT OPC-UA Server Projekt	38
Abbildung 32: Neuen Gerätetyp hinzufügen	39
Abbildung 33: Konfiguration der Verbindungsparametern	39
Abbildung 34: Einstellungen für UA-Endpunkte	40
Abbildung 35: Verbindung zum OPC UA-Server	40
Abbildung 36: Konfiguration der Einstellungen für das Zertifikatvertrauen	41
Abbildung 37: Kommunikationsstruktur des Python-Clients mit den beiden Servern	42
Abbildung 38: Implementierung des KUKAVARPROXY Message Format in der Python Client-Applikation	43
Abbildung 39: KVP verbunden mit dem Client	43
Abbildung 40: Implementierung von FREE OPC UA PYTHON in der Python Client-Applikation ...	44
Abbildung 41: XTS-Maschine	45
Abbildung 42: Registerkarte Backup-Konfiguration	48
Abbildung 43: Registerkarte Signalschnittstelle	48

Abbildung 44: Bedienoberfläche - Robotersteuerung	50
Abbildung 45: Image auf Disk aufspielen – Robotersteuerung	50
Abbildung 46: Image von Disk erzeugen – Robotersteuerung.....	51
Abbildung 47: Registerkarte TCP/IP Adressen – Robotersteuerung	51
Abbildung 48: Registerkarte Experten Einstellungen – Robotersteuerung	52
Abbildung 49: Seite CONFIG.Data des Roboters.....	54
Abbildung 50: Aktivitätsdiagramm des Roboterprogramms.....	56
Abbildung 51: Zustandsdiagramm [52].....	60
Abbildung 52: Bausteinfunktion MC_Power [53]	62
Abbildung 53: Bausteinfunktion MC_Reset [54].....	63
Abbildung 54: AXIS_REF Datentyp [55]	63
Abbildung 55: Timer TON.....	64
Abbildung 56: Funktionsbaustein MC_UngroupAllAxes [68]	68
Abbildung 57: Funktionsbaustein MC_GroupHalt [69].....	68
Abbildung 58: Funktionsbaustein MC_GroupStop [70]	69
Abbildung 59: Funktionsbaustein MC_MoveAbsoluteCA [71]	69
Abbildung 60: Funktionsbaustein MC_GroupEnable [72]	69
Abbildung 61: Funktionsbaustein MC_GroupEnable [73]	70
Abbildung 62: Funktionsbaustein MC_GroupReset [74].....	70
Abbildung 63: Funktionsbaustein MC_GroupReadStatus [75].....	71
Abbildung 64: Funktionsbaustein MC_GroupReadError [76]	71
Abbildung 65: Funktionsbaustein MC_AddAxisToGroup [77].....	71
Abbildung 66: Struktur AXES_GROUP_REF [78].....	72
Abbildung 67: Aktivitätsdiagramm FB_PowerAll.....	74
Abbildung 68: Aktivitätsdiagramm FB_Add_All_Axes	76
Abbildung 69: Aktivitätsdiagramm FB_Group_Halt.....	77
Abbildung 70: Aktivitätsdiagramm MC_MoveAbsoluteCA	79
Abbildung 71: Aktivitätsdiagramm: Data_Loaded_Diagramm	80
Abbildung 72: Aktivitätsdiagramm: MC_GroupReset.....	81
Abbildung 73: Aktivitätsdiagramm: Movers Detection	82
Abbildung 74: Aktivitätsdiagramm: Python-Programmierung	85
Abbildung 75: HMI-Architektur [85].....	87
Abbildung 76: Content-Datei	88
Abbildung 77: Main-Seite	88
Abbildung 78: Menüdaten des HMI-Navigation-Bar.....	89
Abbildung 79: Login-Logout	89
Abbildung 80: Sequenzdiagramm: Reset	90
Abbildung 81: Sequenzdiagramm: Start	91
Abbildung 82: Sequenzdiagramm: Stop.....	91
Abbildung 83: Sequenzdiagramm: Connect.....	92
Abbildung 84: DataGrid-Columns	93
Abbildung 85: Configuration-Seite	94
Abbildung 86: Sequenzdiagramm: Inputs	97
Abbildung 87: Sequenzdiagramm: Load.....	97
Abbildung 88: Sequenzdiagramm: SaveData.....	98
Abbildung 89: Messages-Seite	98
Abbildung 90: EventGrid-Spalten.....	99

Tabellenverzeichnis

Tabelle 1: Bewegungsbereich der Achsen [27]	18
Tabelle 2: Roboter-Betriebsart [28].....	20
Tabelle 3: KUKAVARPROXY Message Format [34]	28
Tabelle 4: Bewegungsbereich der Achsen	47
Tabelle 5: Timer TON	64
Tabelle 6: Primitive Datentypen [59]	66
Tabelle 7: Tc2_Math Funktionen [79]	72
Tabelle 8: Bewegung des Roboters in Abhangigkeit von der Anzahl der Mover und deren Zustand ..	84

Abkürzungen

OPC: Open Platform Communications

OPC UA: Open Platform Communications Unified Architecture

XTS: eXtended Transport System

DCOM: Distributed Component Object Model

SPS: speicherprogrammierbaren Steuerung,

OPC DA: Open Platform Communications Data Access

OPC AE: Open Platform Communications Alarms and Events

OPC HDA: Open Platform Communications History Data Access

XML DA: Extensible Markup Language Data Access

OPC AC: OPC Alarme und Bedingungen

CPPS: Cyber-Physische-Produktionssysteme

4.0: Industrie 4.0

RAMI 4.0: Referenzarchitekturmodell Industrie 4.0

IIoT: Industrial Internet der Dinge (industrial internet of things)

IP: Internetprotokoll

IT: Information Technology

HMI: Human Machine Interface

KRC: KUKA Robot Controller

KSS: KUKA SystemSoftware

KVP: KUKAVARPROXY

NC: Numerical Control

CNC: Computerized Numerical Control

IPC: Industrial PC

TCP/IP: Transmission Control Protocol/Internet Protocol

ADS: Automation Device Specification

SOA: Service oriented Architecture

1. Einführung

Der Bereich der Automatisierung und Informatisierung ist einer der wichtigsten Bereiche in unserer Welt, insbesondere bei der Entwicklung der Produktionsindustrie, die ohne Automatisierung und Digitalisierung heute nicht mehr möglich ist. Selbst in Bereichen, in denen die menschliche Arbeitskraft aufgrund ihrer Spezialisierung oder Komplexität die effizienteste Produktionsmethode ist, gewinnt die Automatisierung durch ständige Weiterentwicklung immer mehr an Boden. Daher muss sich die manuelle Produktion weiterentwickeln und die Möglichkeiten der Automatisierung nutzen. Die unterschiedlichen Stärken von Menschen und Maschine müssen so kombiniert werden, dass sie sich gegenseitig optimal ergänzen. [\[1\]](#)

Die industrielle Kommunikation ist heutzutage einer der am weitesten entwickelten Bereichen, da sie die menschlichen Aufgaben im Fertigungsprozess erleichtern und die Produktion mit hoher Qualität verbessern. Damit hat sie eine Revolution im industriellen Bereich ausgelöst.

Dank der industriellen Kommunikation wird die Kommunikation zwischen verschiedenen Maschinen in allen Arbeitsbereichen einfacher und effizienter.

Diese Entwicklung wurde von mehreren Unternehmen vorangetrieben, die sich auf die Entwicklung der industriellen Kommunikation spezialisiert haben, darunter auch die OPC Foundation. [\[2\]](#)

Die OPC Foundation hat mit OPC UA einen Kommunikationsstandard geschaffen, der sich durch seine sichere, zuverlässige und herstellerunabhängige Kommunikation auszeichnet.

Mit seinen enormen Leistungsmerkmalen nimmt OPC UA einen zentralen Platz in der industriellen Kommunikation ein. Die Konnektivitätsnorm zielt darauf hin, den Integrationsaufwand für neue Produkte oder Systeme verschiedener Hersteller zu verringern (Plug and Play oder Plug and Production). [\[3\]](#) [\[87\]](#)

➤ Ziel und Zweck des Projekts

Mit der Entwicklung der Kommunikation zwischen den Maschinen und den derzeit aktuellen Anforderungen in der Kommunikationstechnik, ist es notwendig, Kommunikationssysteme zu entwickeln, die in der Lage sind, autonom zu arbeiten, um mit dieser schnellen Entwicklung Schritt zu halten. Automatisierungssysteme haben viele Vorteile, da sie spezifische

Fähigkeiten und Vorteile bieten, um das Ziel eines jeden Projekts in diesem Rahmen zu erreichen.

Der Vorschlag der Projektbetreuer für das Abschlussprojekt fällt unter dieses allgemeine Konzept. Der Projektplan umfasst die folgenden Aufgaben:

- a) Untersuchung und Programmierung des KUKA-Roboters, der im Labor zur Verfügung steht
- b) Untersuchung und Programmierung des Extended Transport System (XTS), das im Labor zur Verfügung steht
- c) Untersuchung und Verwendung des OPC UA-Protokolls als Mittel zur Kommunikation mit einem KUKA Roboter, der im Labor vorhanden ist.
- d) Untersuchung und Nutzung des OPC UA-Protokolls als Kommunikationsmittel mit dem Extended Transport System (XTS) der Firma Beckhoff, das im Labor vorhanden ist.
- e) Ein Beispiel für die Kommunikation zwischen den beiden in a) und b) genannten Apparaten.
- f) Realisierung des Dashboards für das Beispiel.

Im ersten Kapitel des Berichts wird eine allgemeine Einleitung verfasst, um den Leser des Berichts zu führen und anschließend werden die Aufgaben und das zu erreichende Ziel beschrieben.

Im zweiten Kapitel des Berichts werden die wichtigsten Grundbegriffe beschrieben, die im Projekt verwendet werden, wie z.B. der Begriff Industrie 4.0, auf dem das abgeschlossene Projekt basiert sowie die Definition des Kommunikationsprotokolls OPC UA, das den Kern des Projekts bildet, wobei seine Eigenschaften und Architektur definiert werden und die Definition der Programmiersprache Python, die zur Erstellung des OPC UA-Clients verwendet wird. Das Konzept der HMI und die damit verbundenen Funktionen werden ebenfalls definiert. Der KUKA-Roboter, der im T65-Labor zur Verfügung steht, wird ebenfalls vorgestellt, wobei seine verschiedenen Teile, seine Kinematik und sein Arbeitsbereich sowie die Software, mit der er funktioniert, erklärt werden. Die Beckhoff XTS-Maschine wird ebenfalls definiert, wobei ihre verschiedenen Komponenten sowie das Programm, mit dem diese Maschine programmiert wird, festgelegt werden.

Das dritte Kapitel des Berichts befasst sich mit der Kommunikation zwischen den beiden Maschinen, die im Labor zur Verfügung stehen. In diesem Zusammenhang werden die möglichen Lösungen und die mögliche Kommunikation zwischen den beiden Maschinen untersucht. Anschließend werden die möglichen Vorschläge geprüft und eine Lösung

gewählt, die den Anforderungen des Projekts entspricht und seine Ziele berücksichtigt. Außerdem werden die Eigenschaften des in diesem Projekt verwendeten Open Source-Protokolls definiert und als Alternative zum OPC UA-Protokoll vorgestellt, das auf dem im Labor vorhandenen Roboter verwendet wird.

Das dritte Kapitel ist in drei Teile gegliedert. Im ersten Teil liegt der Schwerpunkt auf der Kommunikation mit dem Roboter über den KUKAVARPROXY-Server, im zweiten Teil auf der Kommunikation mit der Beckhoff XTS-Maschine über deren OPC-UA-Server. Der dritte Teil konzentriert sich auf den OPC-UA-Client im Computer und wie dieser mit den beiden Servern der beiden Maschinen kommuniziert.

Das vierte Kapitel des Berichts ist den Studien über die in diesem Projekt implementierte Applikation vorbehalten. Im ersten Teil liegt der Schwerpunkt auf der Programmierung des Roboters entsprechend den Anforderungen der auszuführenden Applikation. Zunächst wird auf die Sicherheitsanforderungen beim Einsatz des Roboters eingegangen, und es wird erläutert, wie die mit der XTS-Maschine auszutauschenden Variablen zu konfigurieren sind, um dann den Roboter entsprechend den Anforderungen der Applikation zu programmieren. Im zweiten Teil liegt der Schwerpunkt auf der Programmierung der XTS-Maschine. Die bei der Programmierung verwendeten Lizenzen und Bibliotheken werden identifiziert, und dann werden die verschiedenen Schritte des XTS-Programms beschrieben.

Im dritten Teil dieses Kapitels wird die Programmierung des OPC UA Clients und die Art und Weise des Datenaustauschs zwischen dem OPC UA Client und den beiden Servern der beiden Maschinen besprochen.

Das fünfte Kapitel ist der Identifizierung der verschiedenen Komponenten der realisierten HMI gewidmet. Dieses Kapitel ist in drei Hauptseiten unterteilt. Auf der ersten Seite wird das LogIn des Benutzers, die Befehlsschaltflächen und die NavigationBar, mit der der Benutzer zwischen den Seiten navigieren kann, untersucht. Außerdem wird eine Tabelle erstellt, die die Identifikation jedes Mover, seine Position, seinen Status (voll oder leer) und ob er sich in Fixposition oder Warteposition befindet oder nicht, enthält. Auf der zweiten Seite des HMI werden die verschiedenen Konfigurationsanforderungen des KUKA Roboters und des Beckhoff XTS realisiert. Auf der dritten Seite wird eine Tabelle erstellt, in der die Meldungen zu den verschiedenen Schritten der Applikation angezeigt werden, wobei angegeben wird, dass jeder Schritt erfolgreich abgeschlossen wird oder dass bei einem bestimmten Schritt der Applikation ein Fehler auftritt.

Das sechste Kapitel des Berichts befasst sich mit der Validierung der Projektergebnisse, indem die Einhaltung der Anforderungen und die Erreichung der Projektziele überprüft werden.

Zuletzt wird die Arbeit zusammengefasst und einige Ausblicke vorgeschlagen.

2. Stand der Technik

2.1. Industrie 4.0

Der Begriff Industrie 4.0 (I 4.0) wurde 2011 öffentlich bekannt, als eine Initiative namens Industrie 4.0 - eine Vertretervereinigung aus Wirtschaft, Politik und Wissenschaft - die Idee eines Ansatzes zur Stärkung der Wettbewerbsfähigkeit der deutschen Fertigungsindustrie unterstützte. [4] [87]

Obwohl sich zahlreiche akademische Veröffentlichungen, Konferenzen und praktische Artikel mit diesem Thema befassen, ist der Begriff Industrie 4.0 noch nicht endgültig definiert.

Aufgrund der Unklarheit des Begriffs Industrie 4.0, haben die Unternehmen Schwierigkeiten bei der Identifizierung und Umsetzung von Industrie 4.0-Szenarien [5].

Mit dem Begriff "Industrie 4.0" wird die vierte industrielle Revolution bezeichnet, die sich derzeit vollzieht.

In der Geschichte der Menschheit sind drei weitere industrielle Revolutionen der aktuellen Revolution vorausgegangen. Ab der zweiten Hälfte des 18. Jahrhunderts und während des gesamten 19. Jahrhunderts fanden die erste industrielle Revolution statt, die sich in der Einführung mechanischer Produktionsmittel manifestierten. Die zweite industrielle Revolution begann in den 1870er Jahren und basierte auf der Elektrifizierung und der Arbeitsteilung. Die dritte Revolution, die auch als "digitale Revolution" bezeichnet wird, fand in den 1970er Jahren statt. Dank der Entwicklung von Elektronik und Informatik, die die Automatisierung von Produktionsprozessen ermöglichen. [6]

In der folgenden Abbildung sind die vier industriellen Revolutionen dargestellt.

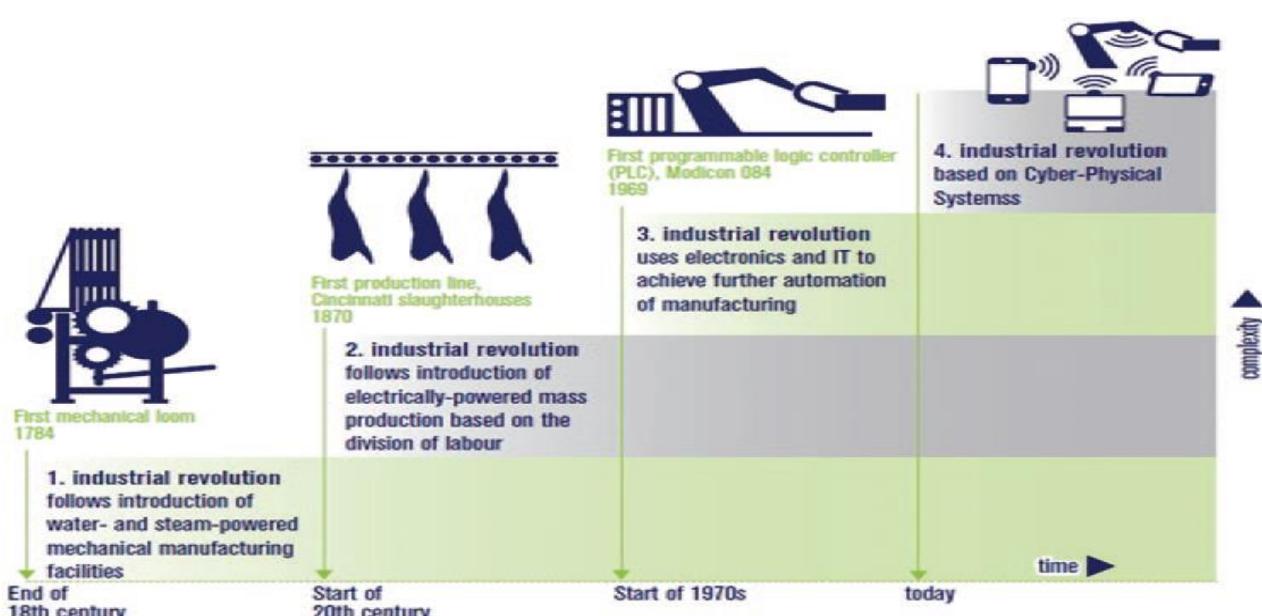


Abbildung 1: Abbildung: Darstellung der vier industriellen Revolutionen durch den Arbeitskreis Industrie 4.0 (Quelle DFKI 2011)

Die Faszination von Industrie 4.0 ist zweifach. Zum einen wird zum ersten Mal eine industrielle Revolution im Vorfeld vorhergesagt und nicht erst im Nachhinein beobachtet. Dadurch ergeben sich für Unternehmen und Forschungseinrichtungen vielfältige Möglichkeiten, die Zukunft aktiv zu gestalten. Zum anderen sollen die wirtschaftlichen Auswirkungen dieser industriellen Revolution enorm sein, denn Industrie 4.0 verspricht eine erhebliche Steigerung der betrieblichen Effizienz sowie die Entwicklung völlig neuer Geschäftsmodelle, Dienstleistungen und Produkte. [7] [87]

Die deutsche Initiative Industrie 4.0 wird von verschiedenen Gruppen geleitet, die die Industrie unterstützen. Mehrere Unternehmen, Organisationen und Universitäten arbeiten an Aspekten der Industrie 4.0, von denen die grundlegenden Anforderungen abgeleitet werden, die den Großteil der laufenden Arbeiten bestimmen [8]:

- Investitionsschutz: Fabriken müssen Industrie 4.0 schrittweise integrieren.
- Stabilität: An Produktionssysteme werden Anforderungen in Bezug auf Echtzeit, Zuverlässigkeit, Verfügbarkeit und Robustheit gestellt, deshalb darf Industrie 4.0 keine Produktionsunterbrechungen oder -ausfälle verursachen.
- Vertraulichkeit der Daten: Der Schutz des Know-hows im Unternehmen erfordert die Kontrolle des Zugangs zu produktionsbezogenen Daten und Dienstleistungen.
- Cyber-Sicherheit: Um ökologische oder wirtschaftliche Schäden sowie Schäden für Menschen zu vermeiden, muss Industrie 4.0 den unbefugten Zugriff auf Produktionssysteme verhindern.

Um die Idee von Industrie 4.0 besser zu verstehen, wurde 2013 ein Bericht von Professor Henning Kagermann (Nationale Akademie der Wissenschaften und Technik), Professor Wolfgang Wahlster (Deutsches Forschungszentrum für Künstliche Intelligenz) und Professor Johannes Helbig (Deutsche Post AG) verfasst, der die Zukunft der deutschen Industrie untersuchen soll [9].

Der Bericht stellt fest, dass in den folgenden acht Schlüsselbereichen Handlungsbedarf besteht [10]:

- Standardisierung und offene Standards für eine Referenzarchitektur:

Durch die Wertevernetzung wird die Industrie 4.0 die Vernetzung und Integration vieler verschiedener Unternehmen beinhalten.

Dafür ist die Entwicklung gemeinsamer Standards notwendig. Daher ist eine Referenzarchitektur erforderlich, die eine technische Beschreibung dieser Normen liefert und ihre Umsetzung erleichtert.

➤ Management komplexer Systeme:

Es besteht die Notwendigkeit, eine Basis geeigneter Planungs- und Erklärungsmodelle bereitzustellen, um die zunehmende Komplexität von Produkten und Fertigungssystemen zu bewältigen.

Die Ingenieure müssen daher mit den Methoden und Werkzeugen ausgestattet werden, um solche Modelle zu entwickeln.

➤ Bereitstellung einer umfassenden Breitbandinfrastruktur für die Industrie:

Die wesentlichen Anforderungen von Industrie 4.0 sind zuverlässige, umfassende und qualitativ hochwertige Kommunikationsnetze. Daher ist der groß angelegte Ausbau der Breitband-Internet-Infrastruktur notwendig.

➤ Sicherheit und Schutz als kritische Faktoren für den Erfolg von Industrie 4.0:

Der Erfolg intelligenter Fertigungssysteme basiert auf Sicherheit und Schutz. Die Produktionsanlagen und die Produkte selbst müssen sicher sein und dürfen keine Gefahr für Mensch und Umwelt darstellen. Gleichzeitig müssen die Produktionsanlagen und Produkte, insbesondere die darin enthaltenen Daten und Informationen, vor Missbrauch und unberechtigtem Zugriff geschützt werden.

➤ Arbeitsorganisation und Arbeitsgestaltung im digitalen Industriezeitalter:

Die Rolle des Arbeitnehmers wird sich in intelligenten Fabriken deutlich verändern. Arbeitsinhalte, Arbeitsprozesse und das Arbeitsumfeld werden sich durch eine zunehmend echtzeitorientierte Steuerung transformieren.

Der Schwerpunkt liegt auf der Umsetzung eines soziotechnischen Systemansatzes (STS) in Arbeitsorganisationen, der den Mitarbeitern die Möglichkeit bietet, mehr Verantwortung zu übernehmen und ihre persönliche Entwicklung zu verbessern.

➤ Ausbildung und berufliche Weiterbildung für Industrie 4.0:

Die Berufsprofile und Qualifikationen der Arbeitnehmer werden sich durch die Industrie 4.0 radikal verändern. Daher müssen geeignete Ausbildungsstrategien durchgeführt und die

Arbeit organisiert werden, dass das Lernen gefördert wird, indem lebenslanges Lernen ermöglicht wird, wobei digitale Lerntechniken erforscht werden sollten.

➤ Regulatorischer Rahmen:

Industrie 4.0 beinhaltet neue Fertigungsprozesse und Unternehmensnetzwerke. Diese müssen rechtskonform sein, so dass auch die bestehenden Rechtsvorschriften angepasst werden müssen, um den neuen Innovationen Rechnung zu tragen. Zu den Herausforderungen gehören der Schutz von Unternehmensdaten, Haftungsfragen und der Umgang mit personenbezogenen Daten.

➤ Ressourceneffizienz:

Die Produktionsindustrie verbraucht große Mengen an Rohstoffen und Energie, was zu Risiken für die Umwelt und die Versorgungssicherheit führt.

Industrie 4.0 wird Produktivitäts- und Effizienzsteigerungen bei den Ressourcen ermöglichen.

Es wird notwendig sein zu berechnen, welche Kompromisse zwischen den zusätzlichen Ressourcen, die in intelligente Fabriken investiert werden müssen, und den potenziellen Einsparungen zu erzielen sind [11].

2.2. Kommunikationstechnologie OPCUA

2.2.1. OPC Classic

Open Platform Communications (OPC) ist eine 1995 entwickelte Technologie für die Interoperabilität von Industriesystemen, die von mehreren Automatisierungsunternehmen in Zusammenarbeit mit Microsoft definiert wurde.[12]

OLE (Object Linking and Embedding) ist der Ursprung des Namens OPC, einer von Microsoft entwickelten Technologie, die die Einbettung und Verknüpfung von Dokumenten und anderen Objekten ermöglicht. Die Spezifikationen dieser Version von OPC sind heute als OPC Classic bekannt. Der OPC-Standard war ursprünglich auf das Windows-Betriebssystem beschränkt. Die OPC Classic-Spezifikation basiert auf der Microsoft Windows-Technologie, die COM/DCOM für den Datenaustausch zwischen Softwarekomponenten auf vernetzten Computern verwendet.[12]

Wie in der Abbildung dargestellt, basiert OPC auf einem Client/Server-Kommunikation, d. h. es gibt einen oder mehrere Server, die auf Anfragen mehrerer Clients warten.[12]

Der Server ist mit den Komponenten, z. B. SPS und Steuerungsgeräten, verbunden und führt die gewünschten Aktionen aus. Bei OPC ist es der Client, der entscheidet, wann und welche Daten der Server von den zugrunde liegenden Systemen abruft. [12] [89]

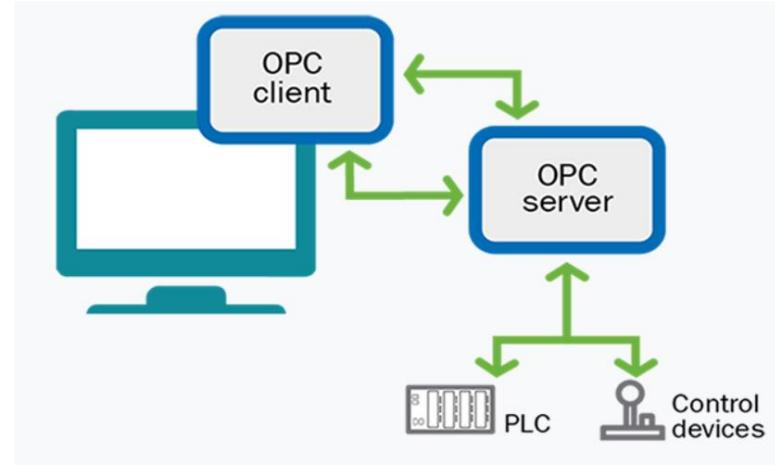


Abbildung 2: OPC-Struktur [82]

In OPC Classic sind die Protokolle wie folgt [13]:

- OPC-Datenzugriff (OPC DA)

Die OPC DA Spezifikation definiert den Austausch von Daten einschließlich Werten, Zeit- und Qualitätsinformationen.

- OPC Alarne & Ereignisse (OPC AE)

Die OPC A&E-Spezifikation definiert den Austausch von Alarm- und Ereignismeldungen sowie von Variablenstatus und Statusmanagement.

- OPC Historischer Datenzugriff (OPC HDA)

Die OPC HDA Spezifikation definiert Abfragemethoden und Analysen, die auf historische, zeitgestempelte Daten angewendet werden können.

- XML DA: XML-Datenzugriff

2.2.2. OPC UA

Die OPC Unified Architecture (Open Platform Communications United Architecture) wurde im Jahr 2008 als plattformunabhängige, service-orientierte Architektur veröffentlicht. Es ermöglicht die Integration aller Funktionen der einzelnen OPC Classic Spezifikationen in ein erweiterbares Framework. [14]

OPC UA ist ein Datenaustauschstandard für die industrielle Kommunikation, genauer gesagt für die Maschine-zu-Maschine- oder Computer-zu-Maschine-Kommunikation. Dieser offene Schnittstellenstandard ist unabhängig vom Hersteller des Anwendungssystems, von der Programmiersprache, mit der die Software programmiert wurde, und vom Betriebssystem, auf dem die Anwendung funktioniert. [\[15\]](#)

Der wichtigste Unterschied zwischen OPC Classic und OPC UA besteht darin, dass OPC UA nicht auf den OLE- oder DCOM-Technologien von Microsoft basiert und somit auf jeder Plattform integriert werden kann. [\[11\]](#)

Mit OPC UA können Maschinendaten nicht nur transportiert, sondern auch semantisch in maschinenlesbarer Form beschrieben werden. Es ermöglicht den Zugriff auf eine Vielzahl von Daten sowohl in vertikaler als auch in horizontaler Richtung. OPC UA kann über Gateways und Aggregations-Server direkt in die Geräte und Steuerungen von Maschinen und Anlagen integriert werden. [\[15\]](#)

2.2.2.1. Datenmodellierung und Kommunikation

Die Datenmodellierung definiert die grundlegenden Regeln und Bausteine, die benötigt werden, um ein Informationsmodell mit OPC UA darzustellen. Sie definiert auch die Einstiegspunkte in den Adressraum und die Basistypen, die zum Aufbau einer Typenhierarchie verwendet werden. [\[16\]](#) [\[89\]](#)

OPC UA Dienste sind die Schnittstelle zwischen Servern als Anbieter eines Informationsmodells und Clients als Nutzer dieses Informationsmodells. Sie verwenden Transportmechanismen, um Daten zwischen dem Client und dem Server auszutauschen. [\[16\]](#)

Dieses Grundkonzept von OPC UA ermöglicht es einem OPC UA Client auf die kleinsten Daten zuzugreifen, ohne das gesamte Modell komplexer Systeme verstehen zu müssen.

Die OPC UA Spezifikation besteht aus dreizehn Teilen, wobei sich die ersten sieben Teile auf die grundlegenden Spezifikationen beziehen, nämlich das Konzept, das Sicherheitsmodell, das Adressraummodell, die Dienste, das Informationsmodell, die Dienstzuordnungen und die Profile. Die Gruppierung von Teil acht bis elf bezieht sich auf die Spezifikationen von Zugriffsarten wie Datenzugriff (DA), Alarme und Bedingungen (AC), Programme (Prog), historischer Zugriff (HA). Die letzte Gruppierung von Teil zwölf und dreizehn bezieht sich auf die Spezifikationen von Hilfsarten wie Entdeckung und Aggregation. [\[17\]](#)

Die folgende Abbildung zeigt die verschiedenen Ebenen von Informationsmodellen:

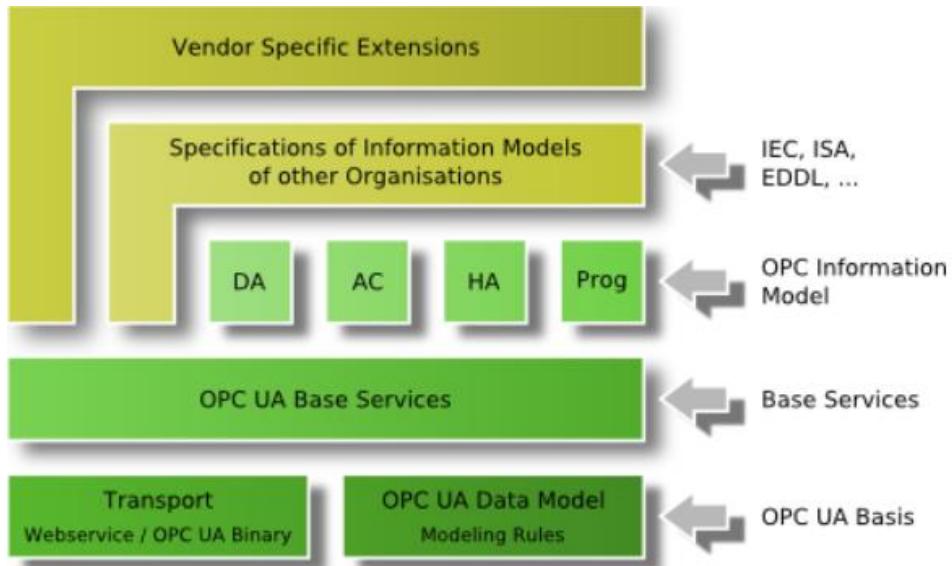


Abbildung 3: OPC UA-Schichtenarchitektur [17]

Die Kommunikation zwischen dem Client und dem Server besteht aus zwei Schichten, um den Informationsfluss oberhalb der Standardtransportschichten zu sichern [17].

Die erste Schicht verwaltet die Session und die zweite Schicht etabliert einen sicheren Kanal zwischen dem Client und dem Server, wie in der folgenden Abbildung dargestellt:

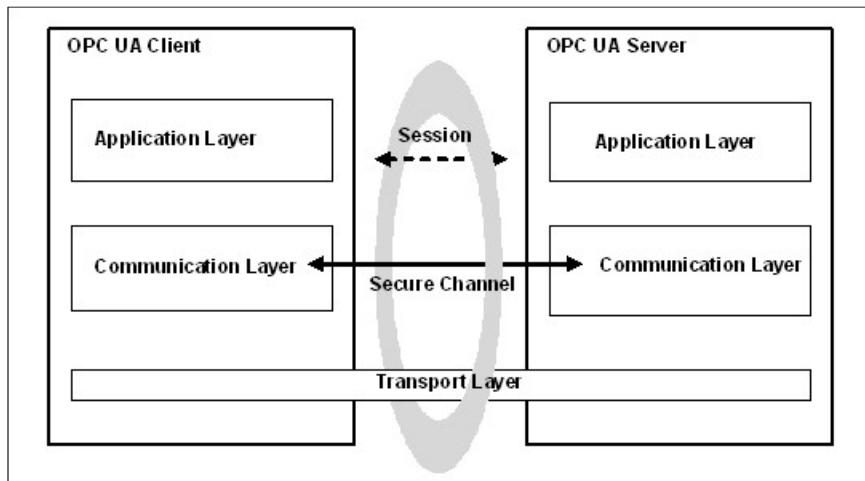


Abbildung 4: OPC UA Kommunikationsmodell [83]

2.2.2.2. *OPC UA und Industrie 4.0*

Industrie 4.0 verwandelt einfache Wertschöpfungsketten mit Hilfe der Digitalisierung und integrierten Vernetzung in voll digitalisierte Wertschöpfungsnetzwerke und schafft neue Geschäftsmodelle. Mehrere R&D-Themen ergeben sich aus der Konzeption und Implementierung von CPPS und der Standardisierung von Schnittstellen für den vertikalen und horizontalen Datenaustausch über Netze hinweg. [18]

Die deutsche Plattform Industrie 4.0, bestehend aus ZVEI, VDMA und BITKOM, hat gemeinsam wichtige Schritte in der Standardisierung von Industrie 4.0 unternommen. Die erste Version eines Referenzarchitekturmodells für Industrie 4.0 (RAMI 4.0), das Industrie 4.0-konforme Produktionsanlagen präzise beschreibt, wurde entwickelt. [19] [87]

Das Modell ist eine dreidimensionale Karte, die aufzeigt, wie das Thema Industrie 4.0 strukturiert angegangen werden kann, um sicherzustellen, dass sich alle an der Diskussion über Industrie 4.0 Beteiligten gegenseitig verstehen. [20]

Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0)

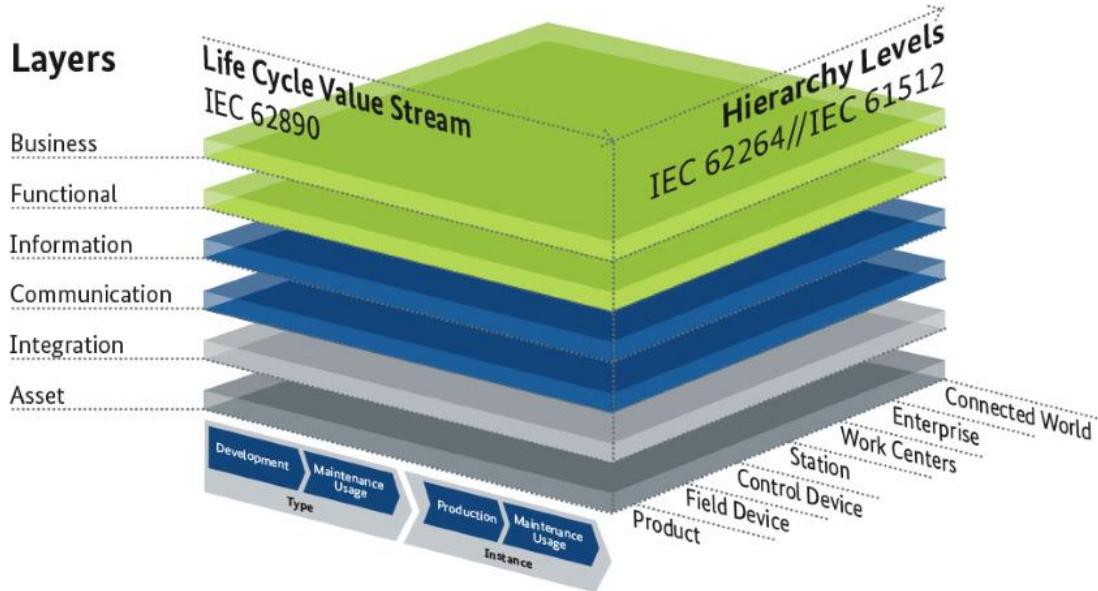


Abbildung 5: RAMI 4.0 [19]

Eine zentrale Herausforderung von Industrie 4.0 und dem Industrial Internet der Dinge (IIoT) ist der sichere und standardisierte Austausch von Daten und Informationen zwischen Geräten, Maschinen und Diensten, auch aus unterschiedlichen Branchen. [15]

Das Referenzarchitekturmodell für Industrie 4.0 (RAMI 4.0) führt IEC 62541 OPC Unified Architecture (OPC UA) als einzige Empfehlung für die Implementierung der Kommunikationsschicht auf. [15] [88] [89]

Die Grundvoraussetzung für die Nutzung von OPC UA für die Industrie 4.0-Kommunikation ist ein auf dem Internetprotokoll (IP) basierendes Netzwerk. Wer also mit dem Label "Industrie 4.0-fähig" werben will, muss mit OPC UA funktionieren (integriert oder über ein

Gateway). Die Informationsmodellierungseigenschaft von OPC UA wird ausdrücklich hervorgehoben. [15]

Die neue Welt der digitalisierten Geräte hilft uns Menschen, die "Dinge" schneller und einfacher zu verstehen, denn sie bieten nun nicht nur Dienste, sondern vor allem auch deren Bedeutung. Dies ist in der Welt der IT nicht neu, aber diese serviceorientierte Architektur (SoA) dehnt sich nun auf die "Objekte" selbst aus und genau hier kommt OPC UA als Teil des industriellen Interoperabilitätsrahmens ins Spiel. Geräte-im RAMI und Maschinenhersteller beschreiben die objektorientierten Informationen ihres Systems und definieren auch Zugriffsrechte mit integrierter IT-Sicherheit. [15]

2.2.2.3. Grundprinzipien der OPC UA-Modellierung

OPC UA verwendet modell-orientierte Techniken, die zum Entwurf komplexer Softwaresysteme verwendet werden und den Prozess des Entwurfs von Informationsmodellen enorm vereinfachen. [18]

Im RAMI4.0-Modell beschreiben die Schichten der vertikalen Achse wie Assets (z.B. eine Maschine) auf der Grundlage ihrer strukturierten Eigenschaften Schicht für Schicht zerlegt werden kann. Mit Hilfe der drei Achsen des RAMI 4.0-Modells ist es möglich, die kritischen Aspekte eines Systems abzubilden. [18] [88]

Die Basiskomponente des OPC UA Metamodells sind die Knoten. Verschiedene Knotenklassen werden durch Spezialisierung der Basisknotenklasse definiert, z.B. Objekte und Variablen.

Jeder der Knoten besteht aus einer Reihe von Attributen, die von seiner Klasse abhängen. Es gibt obligatorische und optionale Attribute. Jede Knotenklasse erfordert beispielsweise eine eindeutige "NodeId", die den Knoten definiert, aber ihre Beschreibung ist optional. [21]

Neben dem Kommunikationsteil ist die Datenmodellierung die zweite Grundlage von OPC UA. Die Grundprinzipien der UA-Modellierung sind im Folgenden aufgeführt [18]:

- Verwendung objektorientierter Techniken mit Typenhierarchien und Vererbung.
- Typinformationen werden bereitgestellt und können auf die gleiche Weise wie bei einer Instanz abgerufen werden.
- Ein vollständig vernetztes Netz von Knoten ermöglicht es, Informationen auf unterschiedliche Weise zu verbinden.
- Erweiterbarkeit von Typenhierarchien sowie von Referenztypen zwischen Knoten.

- Keine Einschränkung bei der Modellierung, um eine angemessene Konzeption des Informationsmodells zu ermöglichen.
- Die Modellierung von OPC UA Informationen erfolgt immer auf dem Server.

2.3. Programmierungssprache Python

Python wurde von Guido van Rossum entwickelt und am 20. Februar 1991 zum ersten Mal veröffentlicht. Es handelt sich um eine objektorientierte, interpretierte Hochsprachenprogrammierung mit dynamischer Semantik, die weithin für die allgemeine Programmierung verwendet wird. [\[22\]](#)

Die Entwicklung von Python ist das Ergebnis der kontinuierlichen Arbeit von Tausenden von Programmierern, Testern, Benutzern und Enthusiasten auf der ganzen Welt. [\[22\]](#)

Guido Van Rossum identifizierte im Jahr 1999 die Ziele für Python, die in den folgenden Punkten dargestellt werden [\[22\]](#):

- eine einfache und intuitive Sprache, die genauso leistungsfähig ist wie die der wichtigsten Wettbewerber
- eine offene Quelle, so dass jeder zu ihrer Entwicklung beitragen kann
- einen Code, der so verständlich ist wie Alltags-Englisch.
- angepasst an alltägliche Aufgaben, was kurze Entwicklungszeiten ermöglicht

Heute, nach zwanzig Jahren, nimmt Python immer noch einen Platz in den Top 10 der Popularitäts-Indizes von Programmiersprachen ein.

Python kann in verschiedenen Bereichen eingesetzt werden, z. B. in der Web- und Internetentwicklung, im wissenschaftlichen und digitalen Rechnen (eine Sammlung von Paketen für mathematische, wissenschaftliche und ingenieurwissenschaftliche Anwendungen), im Bildungswesen, in Desktop-GUIs, in der Softwareentwicklung, in Spielen usw. [\[22\]](#)

2.4. Human Machine Interface (HMI)

Die Human Machine Interface (HMI) ist die Hardware oder Software, über die ein Bediener mit einer Steuerung interagiert. Eine HMI kann von einem physischen Bedienfeld mit Tasten und Anzeigeleuchten bis hin zu einem Industrie-PC mit einem Farbgrafikbildschirm mit spezieller HMI-Software reichen. [\[23\]](#)

Bei der HMI handelt es sich um die Software und Hardware, die es dem Bediener ermöglicht, den Zustand eines zu steuernden Prozesses zu überwachen, Steuerungseinstellungen zu

ändern, um das Steuerungsziel zu verändern, und automatische Steuerungsvorgänge in Notfällen manuell zu übersteuern. Die HMI ermöglicht es einem Steuerungsingenieur oder Bediener auch, Set-Points oder Steuerungsalgorithmen und Parameter in der Steuerung zu konfigurieren. Die HMI zeigt außerdem Informationen zum Prozessstatus, historische Informationen, Berichte und andere Informationen für Bediener, Administratoren, Manager, Geschäftspartner und andere autorisierte Benutzer an. Bediener und Ingenieure verwenden HMIs zur Überwachung und Konfiguration von Set-Points, Regelalgorithmen, zum Senden von Befehlen und zum Einstellen und Festlegen von Parametern in der Steuerung. Die HMI zeigt auch Informationen zum Prozessstatus und historische Informationen an. [\[23\]](#)

2.5. KUKA-Roboter

Der Roboter, der in diesem Projekt verwendet wird, ist im Automatisierungslabor T65 der Hochschule Emden/Leer verfügbar.

Der Industrieroboter besteht aus den folgenden Komponenten

- KUKA Manipulator KR600
- Robotersteuerung KUKA KRc4 Compact
- Programmierhandgerät smartPAD
- Verbindungsleitungen
- Software

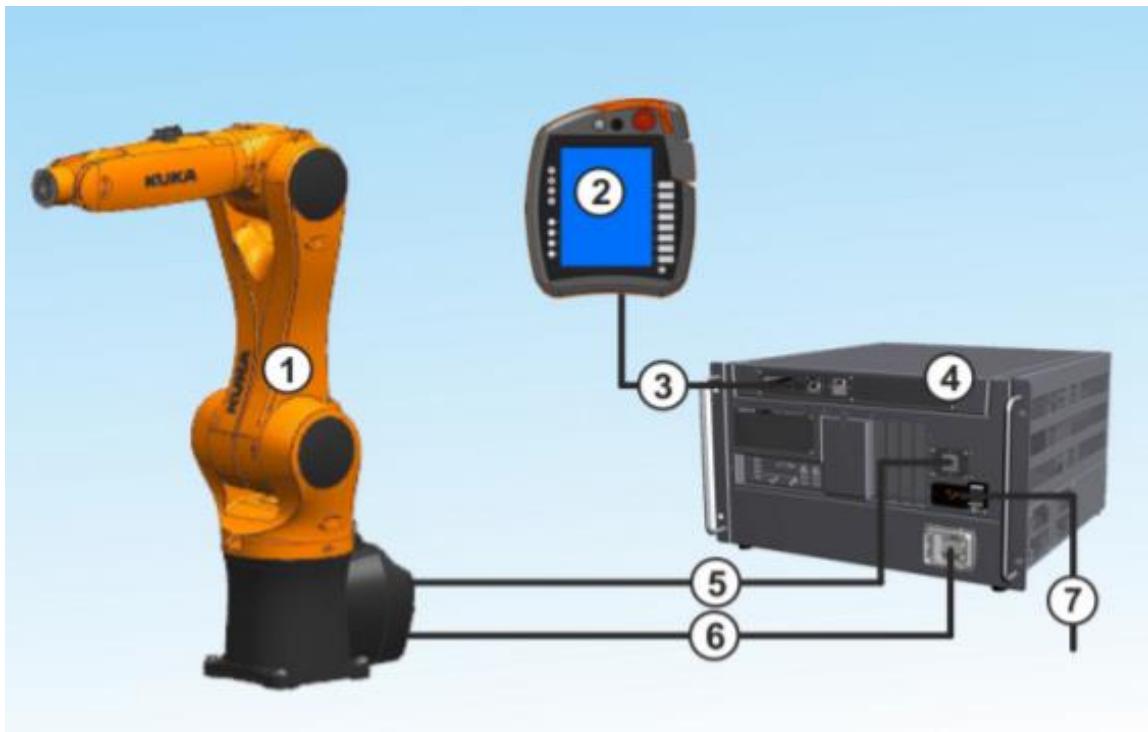


Abbildung 6: Industrieroboter [86]

1 Manipulator

2 Programmierhandgerät

3 Verbindungsleitung/smartPAD

4 Robotersteuerung

5 Verbindungsleitung/Datenleitung

6 Verbindungsleitung/Motorleitung

7 Geräteanschluss-Leitung

2.5.1. Roboter-Software

Die KUKA Systemsoftware (KSS) ist für alle Grundfunktionen des Industrieroboters zuständig, wie z.B. Bahnplanung, Management von Ein- und Ausgängen, Daten- und Dateiverwaltung usw. [24]

Die KUKA Systemsoftware ist ausschließlich für den Betrieb eines KUKA Industrieroboters oder einer kundenspezifischen Kinematik bestimmt. [24]

Im Labor der Hochschule Emden Leer wird KSS auf der Robotersteuerung betrieben.

Zusätzliche Technologiepakete, die anwendungsspezifische Anweisungen und Konfigurationen enthalten, können ebenfalls installiert werden, jedoch sind die Kosten für diese Pakete sehr hoch. Forscher und Programmierer haben erschwinglichere Programmierwerkzeuge und Software gefunden, um diese Anforderungen zu erfüllen, da die Verwendung von Robotern in der akademischen und industriellen Welt sehr verbreitet ist. [\[24\]](#)

In diesem Projekt wurde eine Open-Source-Softwarelösung verwendet, nämlich KUKAVARPROXY.

2.5.2. KUKAVARPROXY

KUKAVARPROXY (KVP) wird von der Firma IMTS S.r.L. gegründet. Es wurde zuerst zusammen mit OpenShowVar auf Sourceforge veröffentlicht. [\[25\]](#)

Massimiliano Fago war die erste Person, die KVP und OpenShowVar in der Programmiersprache C++ entwickelte [\[25\]](#).

KVP ist ein Server, der auf der KUKA Robotersteuerung ausgeführt wird und mit dem internen System des Roboters kommunizieren kann.

KVP funktioniert als Hintergrundanwendung auf der Robotersteuerung und kann über KUKA CrossCom mit der VxWorks-Seite der Steuerung kommunizieren. [\[26\]](#)

KVP ist in verschiedenen Forschungsprojekten sehr gut einsetzbar, wurde aber auf dem KUKA-Roboter im T65-Automatisierungslabor der Hochschule Emden Leer in vorherigen Projekten nicht eingesetzt.

2.5.3. Roboter-Kinematik

Der KUKA-Roboter, der im Automatisierungslabor T65 zur Verfügung steht, besteht aus einem KR6 R900 sixx Manipulator. Jede der Achsen kann sich sowohl in positiver als auch in negativer Richtung drehen.

Die Bewegungsrichtung und die Anordnung der verschiedenen Achsen sind in der folgenden Abbildung dargestellt:

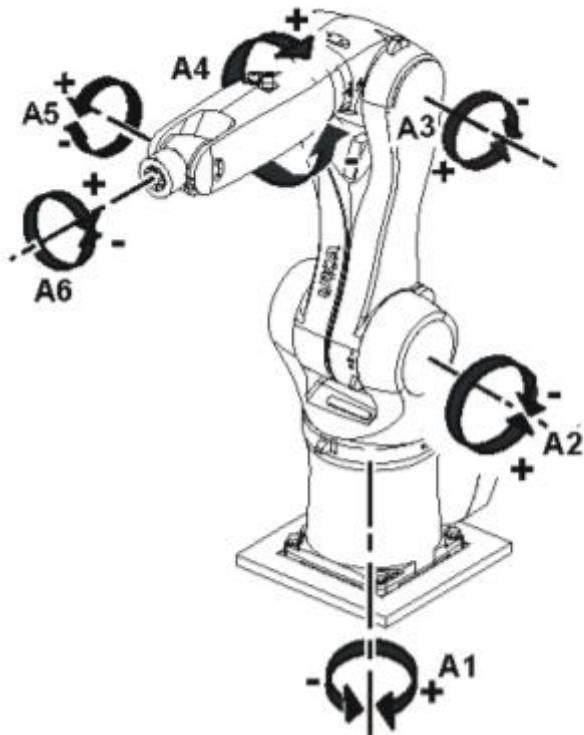


Abbildung 7: Drehrichtung der Roboterachsen [27]

Die kinematischen Achsen werden vom Hersteller durch Begrenzungen der Bewegungsamplituden abgegrenzt. [27]

Diese Bewegungsbereiche sind für jede der Achsen unterschiedlich, wie in der folgenden Tabelle dargestellt [27]:

Achse	Bewegungsbereich
A1	+/-170°
A2	+45° bis -190°
A3	+156° bis -120°
A4	+/-185°
A5	+/-120°
A6	+/-350°

Tabelle 1: Bewegungsbereich der Achsen [27]

Die Form und Größe des Arbeitsbereichs dieses Modells eines Robotermanipulators ist in der folgenden Abbildung dargestellt:

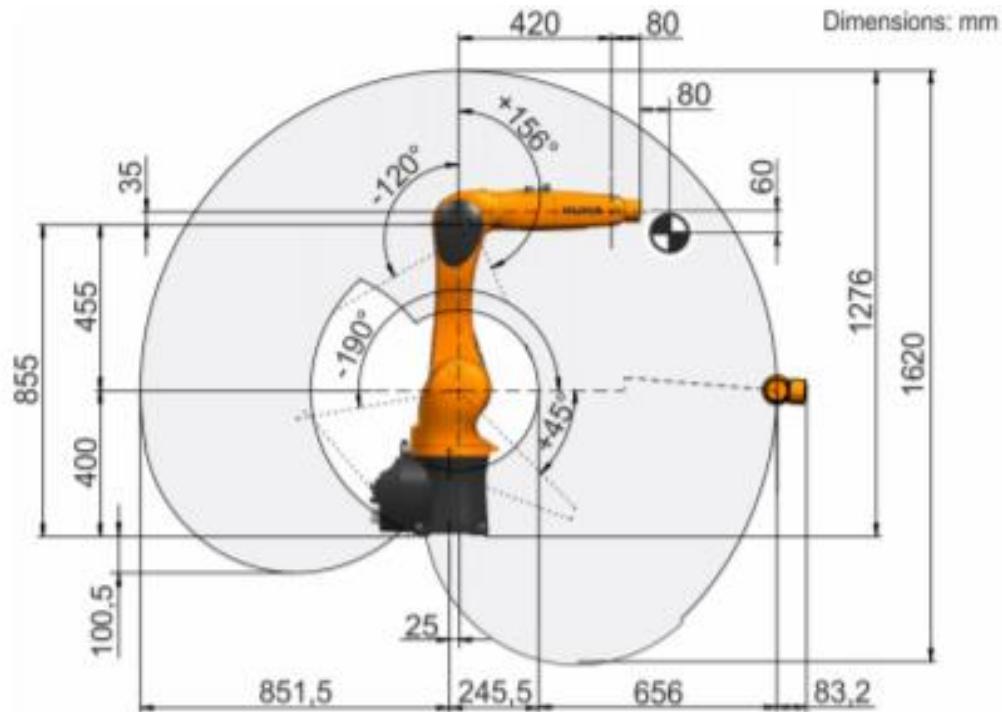


Abbildung 8: Grafik Roboterarbeitsbereich [84]

Dieser Roboter ermöglicht es dem Benutzer, ihn in verschiedenen Betriebsarten zu verwenden. Für jede Betriebsart gibt es Bedingungen, die in Bezug auf die Art der Nutzung und die Geschwindigkeit beachtet werden müssen. [28]

In der folgenden Tabelle sind die vier Betriebsarten und ihre Geschwindigkeitsbedingungen dargestellt. [28] [86]

Betriebsart	Nutzung	Geschwindigkeit
T1	Für Testbetrieb, Programmierung und Unterricht	Programmierte Geschwindigkeit, maximal 250 mm/s
T2	Für Testbetrieb	Programmierte Geschwindigkeit
AUT	Für Industrieroboter ohne übergeordnete Steuerungen	Programmierte Geschwindigkeit
AUT EXT	Für Industrieroboter mit übergeordneten Steuerungen	Programmierte Geschwindigkeit

Tabelle 2: Roboter-Betriebsart [\[28\]](#)

Der Roboter unterstützt mehr als eine Bewegungsart. Es gibt drei Arten von Bewegungen, die für die Programmierung bekannt sein müssen.

Durch geschickte Kombination ist es möglich, fließende Bewegungsabläufe zu schaffen.

Die drei Arten der Roboterbewegung werden im Folgenden erläutert. [\[86\]](#)

a. Point-to-Point (PTP)

PTP ist eine achsenspezifische Bewegungsart, die für den Roboter am schnellsten durchführbar ist. Dieses wird aber nicht unbedingt der kürzeste Weg sein, da sich die Achsen rotierend bewegen und dadurch geschwungene Bahnen entstehen. Die Bewegungen sind aus diesem Grund nicht vorhersehbar und können in alle Richtungen geschwungen sein. [\[29\]](#)

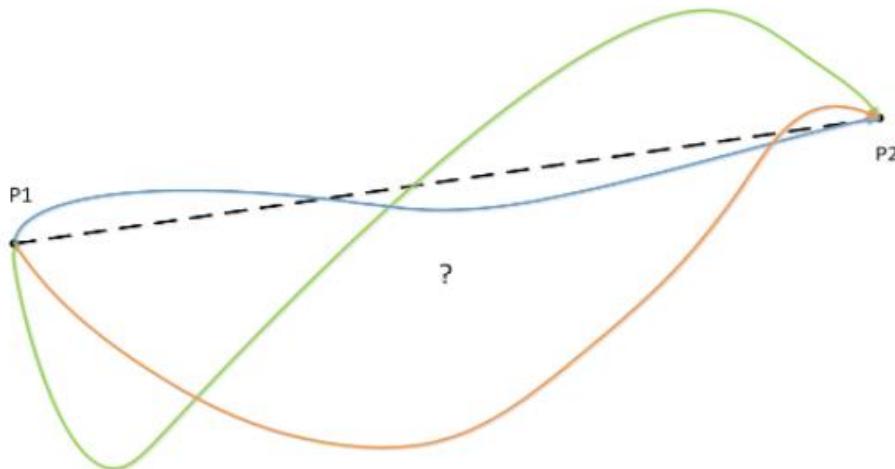


Abbildung 9: Bewegungsart PTP [\[29\]](#)

b. Linear (LIN)

LIN ist eine geradlinige Bewegung. Der Greifer wird in einer konstanten Geschwindigkeit und in definierter Orientierung vom Start (P1) zum Zielpunkt (P2) bewegt. Die Achsen verfahren also so, dass der Greifer eine gerade Luftlinie zurücklegt. [\[29\]](#)



Abbildung 10: Bewegungsart LIN [\[29\]](#)

c. Circular (CIRC)

CIRC eine kreisförmige Bahnbewegung, die vom Startpunkt (P1) aus über einen Hilfspunkt (HP) zum Zielpunkt (ZP) ausgeführt wird. Der Hilfspunkt ist dabei als äußerste Stelle des Kreises definiert. [\[29\]](#)



Abbildung 11: Bewegungsart CIRC [\[29\]](#)

2.6. Extended Transport System (XTS)

2.6.1. XTS Spezifikationen

Das eXtended Transport System (XTS) ist für Maschinen und Anlagen mit höchsten Anforderungen an Dynamik und Positioniergenauigkeit konzipiert. [\[30\]](#)

Dieses Produkt bietet völlig neue Freiheitsgrade im Maschinenbau. Die XTS Maschine besitzt die Vorteile bewährter linearer und rotatorischer Antriebssysteme. Außerdem verfügt die Maschine auch über ein Antriebssystem, dass das konventionelle Linearmotorprinzip erweitert. [\[31\]](#)

XTS ist ein Linearmotor, der sich im Kreis bewegt. Motor, Leistungselektronik und Wegmessung sind in einem gemeinsamen Gerät vereint. Ein oder mehrere Mover können mit hohen Geschwindigkeiten von bis zu 4 m/s entlang einer beliebigen und flexiblen Bahn bewegt werden. [\[31\]](#)

Der XTS ist eine Revolution im Bereich der Antriebstechnik und ermöglicht völlig neue und platzsparende Maschinenkonzepte, [\[31\]](#)

XTS ist ein mechatronisches System, das aus wenigen Komponenten besteht, welche die für den Betrieb notwendigen Funktionen bereitstellen [\[31\]](#):

- ein modularer, voll integrierter Linearmotor mit Leistungselektronik und Wegmessung in einem Gerät.
- einen oder mehrere Movers als bewegliche Teile.
- ein mechanisches Führungsschienensystem.
- einen industriellen PC mit der Steuerungssoftware TwinCAT.

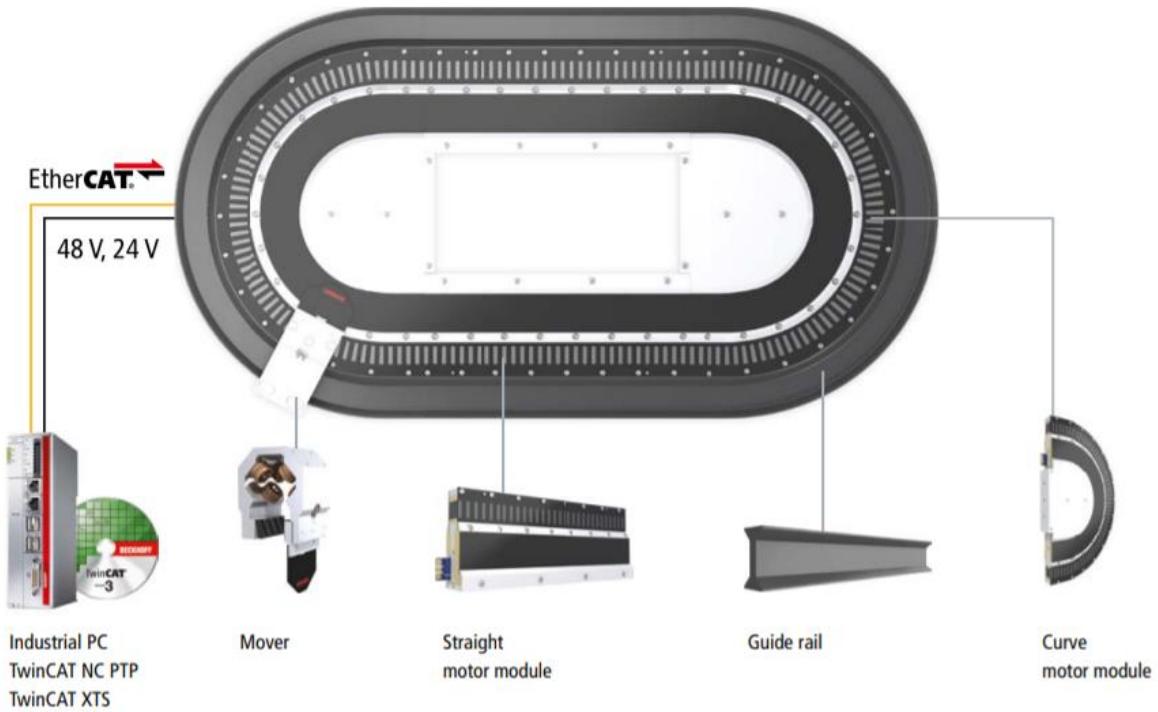


Abbildung 12: XTS-Komponenten [31]

- Motor-Modul:

Der voll integrierte Linearmotor mit Leistungselektronik und Wegmessung stellt eine mechatronische Einheit dar. Es enthält die elektromagnetischen Spulen und alle anderen aktiven Funktionen, die für den Betrieb des Systems erforderlich sind. [31]

Es werden lediglich eine Stromversorgung und ein EtherCAT-Anschluss benötigt.

- Führungsschiene:

Das Schienensystem kann je nach Bedarf mit geraden und gebogenen Abschnitten ergänzt werden. Die gewünschten Geometrien, Längen und Radien werden durch die Anzahl und die Auswahl der Komponenten gebildet. In Kombination mit der Kontaktfläche der Laufrollen erreicht das System gute Laufeigenschaften und geringen Verschleiß. [31]

- Mover:

Der Mover enthält Magnetplatten, die zusammen mit den Spulen des Motormoduls eine kontrollierte Antriebskraft erzeugen.

Der Mover nimmt die Anziehungskräfte der Magnete auf beiden Seiten auf und kompensiert sie weitgehend gegenüber dem Führungsmechanismus. [\[31\]](#)

- Industrie-PC:

Die Industrie-PC-Serie C69xx zeichnet sich durch ihre kompakte Bauweise, das robuste Aluminiumgehäuse und eine besonders große Auswahl an möglichen Komponenten und Schnittstellen aus.

Die Palette der Industrie-PCs mit 3½-Zoll-Beckhoff-Motherboard reicht vom sehr kleinen C6905 für Anwendungen mit mittleren Leistungsanforderungen bis hin zum C6930 als leistungsstarke Plattform für hochkomplexe Anwendungen im Maschinen- und Anlagenbau, zum Beispiel mit der Automatisierungssoftware TwinCAT. [\[32\]](#)

Sie sind zudem mit Prozessoren bis zu Intel® Core™ i7 der neuesten Generation ausgestattet und eignen sich für leistungshungrige Anwendungen der Steuerungstechnik. [\[32\]](#)

2.6.2. Automatisierungssoftware TwinCat 3

Das Herzstück des Steuerungssystems ist die Automatisierungssuite TwinCAT (The Windows Control and Automation Technology). [\[33\]](#)

Sie verwandelt nahezu jedes PC-basierte System in eine Echtzeitsteuerung mit mehreren SPS-, NC-, CNC- und/oder Roboterausführungssystemen. [\[33\]](#)

TwinCat 3 definiert die Welt der Automatisierungstechnik neu und zeichnet sich durch verschiedene Stärken aus, wie z.B. ein einziges Softwarepaket für die Programmierung und Konfiguration. Sie bietet mehr Freiheit bei der Wahl der Programmiersprache und unterstützt die objektorientierte Erweiterung der Norm IEC 61131-3. TwinCat 3 integriert eine Steuerungslösung von NC PTP, Robotik bis CNC. Dank seiner offenen Schnittstellen ermöglicht TwinCat 3 die Erweiterung und Anpassung an bestehende Tools, eine flexible Laufzeitumgebung und die aktive Unterstützung von Multi-Core-CPUs. [\[33\]](#)

TwinCat 3 unterstützt 32 und 64 Bit Betriebssysteme (Windows CE, Windows 7, Windows 10, TwinCAT/BSD). [\[33\]](#)

Die TwinCat-Automatisierungsschnittstelle generiert automatisch Code und erstellt Projekte. [\[33\]](#)

TwinCat 3 ermöglicht die Anbindung von Daten an Bedienoberflächen und andere Programme über offene Standards (OPC, ADS, etc.). [\[33\]](#)

3. Spezifikation einer Kommunikation Infrastruktur

Je nach der Beschreibung des OPC UA Kommunikationsprotokolls basiert die Kommunikation auf einem OPC UA Client und Server.

Der erste Vorschlag für die Kommunikationsarchitektur zwischen dem KUKA KRC4 Compact-Roboter und dem XTS über das OPC-UA-Kommunikationsprotokoll war folgender:



Abbildung 13: Erster Vorschlag für die Kommunikationsarchitektur

Der erste Vorschlag basiert auf der Idee, einen OPC-UA-Server innerhalb des Beckhoff C6930 IPC des XTS zu implementieren. Der OPC-UA-Client ist die Steuerung KRC4 Compact des KUKA Roboters. Der Client kann in diesem Fall mit dem Server kommunizieren, indem er Variablen liest und schreibt.

3.1. Lösung

Das Problem in diesem Vorschlag besteht darin, dass der KUKA-Roboter in seiner KRC4 Compact-Steuerung standardmäßig keine OPC-UA-Funktionalität unterstützt. KUKA hat die OPC-UA-Funktionalität in diesen Robotern ab dem Jahr 2019 standardmäßig implementiert, und der im T65-Labor vorhandene Roboter wurde 2016 hergestellt.

Zur Lösung dieses Problems wurden bereits Forschungsarbeiten durchgeführt.

Eine Lösung wäre der Kauf einer KOP-Lizenz von der Firma KUKA, um die OPC-UA-Funktionalität standardmäßig in den im T65-Labor verfügbaren Robotern zu implementieren, aber die Kosten für den Kauf und die Installation sind hoch, daher wird diese Lösung in diesem Projekt nicht verwendet.

Die andere Lösung ist die Integration des Beckhoff Ethercat EL6695 Kopplers zwischen dem Beckhoff IPC des XTS und der KRC4 Compact Steuerung des KUKA Roboters, so dass die

Signalaustausch zwischen beiden Maschinen verständlich ist, aber in diesem Fall entfällt die Verwendung des OPC UA Konzepts, das in diesem Projekt eine Voraussetzung ist.

Daher scheidet auch diese Lösung aus.



Abbildung 14: Kommunikationsarchitektur mit EL6695

Dieses Problem hat uns dazu veranlasst, nach einem anderen Vorschlag für eine Kommunikationsarchitektur zu suchen, die die Kommunikation zwischen den beiden Steuerungen der beiden Maschinen gewährleistet und gleichzeitig die Tatsache berücksichtigt, dass die Kommunikation über das Kommunikationsprotokoll OPC UA erfolgt.

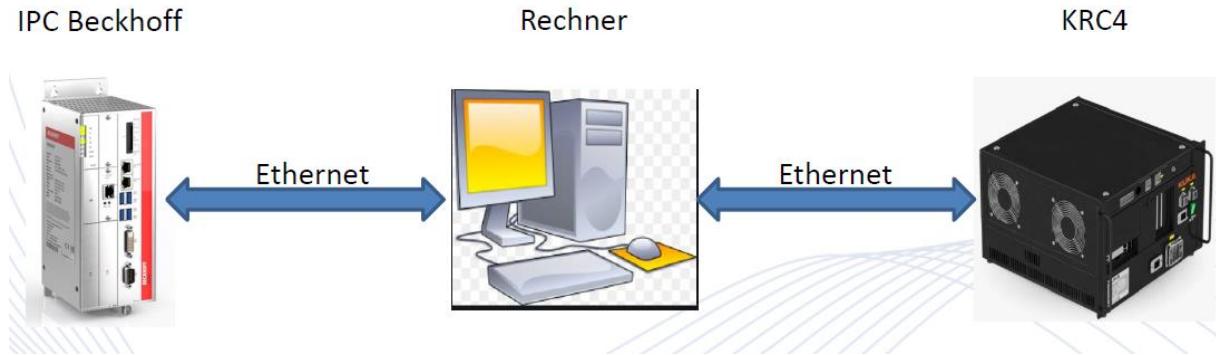
Nach einigen Nachforschungen fanden wir eine Masterarbeit von Aksel Øvern, die an der Norwegian University of Science and Technology durchgeführt wurde. Aksel Øvern hat das gleiche Problem gefunden. Die Lösung ist die Verwendung einer Opensource-Kommunikationsschnittstelle mit der Robotersteuerung KRC4 Compact. Diese alternative Schnittstelle muss die gleichen Eigenschaften der OPC UA gewährleisten. [\[11\]](#)

Die Antwort ist die Verwendung einer Opensource-Kommunikationsschnittstelle zur Robotersteuerung namens KUKAVARPROXY Message Format und KUKAVARPROXY (KVP) (siehe Abschnitt [3.2.1](#). und [3.2.2](#).).

Das Kommunikationsprinzip in diesem Projekt basiert auf drei Hauptteile, die der OPC UA Norm entsprechen. Die drei Teile sind wie folgt:

- ein KUKAVARPROXY-Server, der innerhalb der KRC4 Compact-Steuerung des Roboters läuft.
- ein OPC UA Server, der innerhalb des BECKHOFF IPC des XTS läuft.
- ein KUKAVARPROXY Message Format -Client läuft auf einem Computer.

Die folgende Abbildung veranschaulicht die Kommunikationsarchitektur zwischen den drei Geräten, die in diesem Projekt zur Verfügung stehen.



Zum besseren Verständnis der Kommunikationsphase ist dieser Teil des Berichts in drei Teile gegliedert. Der erste Teil konzentriert sich auf die Kommunikation zwischen dem Roboter und dem Computer unter Berücksichtigung des OPC UA Standards. Der zweite Teil befasst sich mit der Kommunikation zwischen dem XTS und dem Computer über den OPC UA-Standard und der letzte Teil erläutert die Kommunikationsphase innerhalb des Computers.

3.2. Roboter-Kommunikation über OPCUA [86]

3.2.1. KUKAVARPROXY Message Format und KUKAVARPROXY

KVP ist der TCP/IP-Server, der auf der Robotersteuerung funktioniert, während KUKAVARPROXY Message Format der Client ist, der die Serververbindung ermöglicht [38]. Die Architektur dieser Kommunikation zwischen KUKAVARPROXY Message Format und KVP ist in der folgenden Abbildung dargestellt.

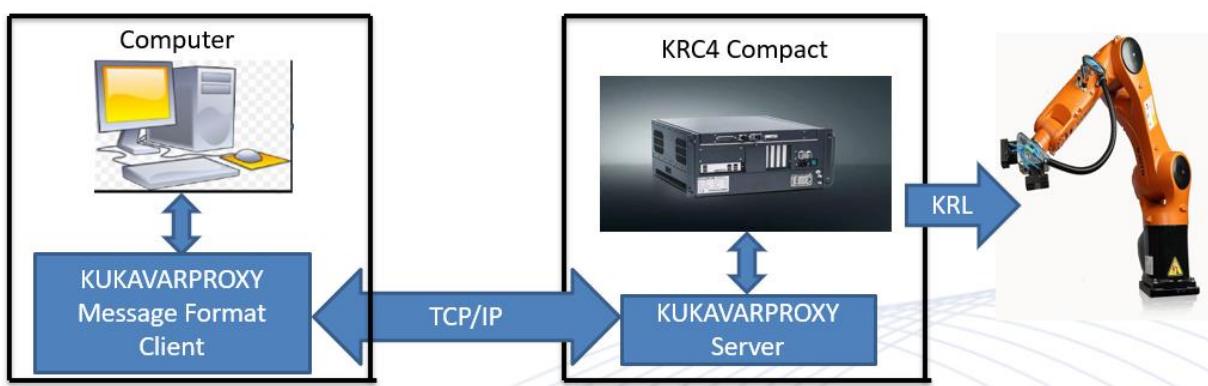


Abbildung 16: Client-Server-Architektur

Die Verwendung von KUKAVARPROXY Message Format ermöglicht das Lesen und Schreiben von globalen Variablen auf der Robotersteuerung von einem Computer aus [34]. (Ahmad Saeed 2017) KUKAVARPROXY Message Format ist mit allen KUKA Robotern, die KRC4 verwenden, kompatibel und läuft als Client auf einem entfernten Computer, der mit der Robotersteuerung verbunden ist.

3.2.2. Funktionsweise von KUKAVARPROXY Message Format

KVP ist ein TCP/IP-Server, der auf dem Port 7000 auf Netzwerknachrichten wartet [34]. Es liest und schreibt Daten in die KRC-Systemvariablen. Das KUKAVARPROXY Message Format ist wie folgt:

KUKAVARPROXY Message Format	Größe
Nachrichten-ID in HEX	2 Bytes
Nachrichtenlänge in HEX	2 Bytes
Lesen (0) oder Schreiben (1)	1 Byte
Länge des Variablenamens in HEX	2 Bytes
Variablenname in ASCII	# Bytes
Länge des Variablenwerts in HEX	2 Bytes
Wert der Variablen in ASCII	# Bytes

Tabelle 3: KUKAVARPROXY Message Format [34]

3.2.3. Installation von KUKAVARPROXY in der KRC4 Compact Steuerung

KUKAVARPROXY ist eine TCP/IP-Schnittstelle zu Kuka-Robotern, die das Lesen und Schreiben von Variablen und Datenstrukturen der gesteuerten Manipulatoren ermöglicht. [34]

Um KVP auf der KRC4 Compact Robotersteuerung zu installieren, ist eine Open-Source-Version von KVP auf sourceforge.net verfügbar. [35]

Ahmad Saeed hat auf Github.com eine übersichtliche Anleitung zur Installation von KVP auf einer KRC4-Steuerung des KUKA-Roboters erstellt [34].

3.2.3.1. Verwendung

Die KRC-Robotersteuerung läuft auf dem Betriebssystem Microsoft Windows. Das SmartPad zeigt ein "HMI", ein von KUKA entwickeltes Programm, das unter Windows läuft und die Schnittstelle darstellt, über die der Roboterbenutzer den Roboter bedienen kann. Um ein Ethernet (TCP/IP) -Verbindung herzustellen, muss zuerst die Ausführung von KVP auf dem

Betriebssystem der Steuerung erfolgen und dann die Konfiguration der Netzwerkverbindung vom KUKA „HMI“ durchgeführt werden. [34]

3.2.3.2. Kopieren KUKAVARPROXY auf das Betriebssystem auf der KRC

Um KVP auf dem Betriebssystem Windows XP auf die KRC zu kopieren, müssen die folgenden Schritte befolgt werden [34]:

- 1- Erhalten KVP von der Website. [35]
- 2- Entpacken und kopieren den Ordner auf einen USB-Stick.
- 3- Anschließen den USB-Stick an den KRC4 Compact Controller.
- 4- Anmelden sich als Experte, indem auf KUKA Menü → Konfiguration → Benutzergruppe klicken. Das Standardkennwort ist kuka.
- 5- Minimieren die HMI. Klicken einfach auf KUKA Menü → Startup → Service → HMI minimieren.
- 6- Kopieren den Ordner KUKAVARPROXY auf den Desktop.
- 7- Starten KUKAVARPROXY.exe
- 8- Wenn es ein Problem mit der Datei csksk32.ocx gibt, besteht die Lösung darin, den Befehl regsvr32.exe in der Eingabeaufforderung des Administrators c:\asdf\csksk32.ocx zu verwenden und dabei die asdf mit dem wahren Zugriffspfad auf die Datei zu ändern.
- 9- Damit KUKAVARPROXY beim Neustart automatisch gestartet wird, muss eine Verknüpfung zu KUKAVARPROXY.exe unter Windows Start → Alle Programme → Rechtsklick auf Start → Öffnen erstellen.

3.2.3.3. HMI-Netzwerkkonfiguration

Damit KUKAVARPROXY eine Kommunikation herstellen kann, müssen die Konfigurationsschritte in der HMI des Roboters durchgeführt werden. Diese Konfigurationsschritte werden in den folgenden Punkten erläutert [34]:

- 1- Verbinden den Roboter über ein Ethernet-Kabel mit dem Computer. Der X66-Anschluss der KRC4 Compact Steuerung ermöglicht die Ethernet-Kommunikation mit dem Ethernet-Anschluss des Computers.
- 2- Konfigurieren die IP-Adresse. Klicken einfach auf KUKA Menü → Startup → Netzwerkkonfiguration.

Die IP-Adresse des Roboters lautet 147.31.1.147, der Ethernet-Anschluss des Computers wurde mit einer IP-Adresse von 147.31.1.148 konfiguriert.

- 3- Entriegeln den Port 7000, indem auf KUKA Menü →Startup
→Netzwerkkonfiguration →Erweitert klicken.
- 4- Hinzufügen von Port 7000 durch Anklicken von NAT →Port hinzufügen
→Portnummer 7000
- 5- Definieren die zulässigen Protokolle: TCP/UDP.

3.3. XTS-Kommunikation über OPCUA

Damit TwinCat 3 mit Industrie 4.0 kompatibel ist, entwickelt die Firma BECKHOFF ihre Automatisierungssoftware TwinCat 3 in Bezug auf die Konnektivität weiter, indem sie das Kommunikationsprotokoll OPC UA in ihre TwinCat-3-Software implementiert, so dass die Beckhoff-Produkte über das OPC-UA-Protokoll, das den Kern von Industrie 4.0 darstellt, mit den Maschinen anderer Hersteller verbunden werden können.

Die Verwendung des OPC-UA-Servers erfordert in diesem Fall das Herunterladen der TF6100-Lizenz von der offiziellen Beckhoff-Website. [\[36\]](#)

Die Kommunikationsarchitektur zwischen dem BECKHOFF IPC und dem Computer ist in der folgenden Abbildung dargestellt:

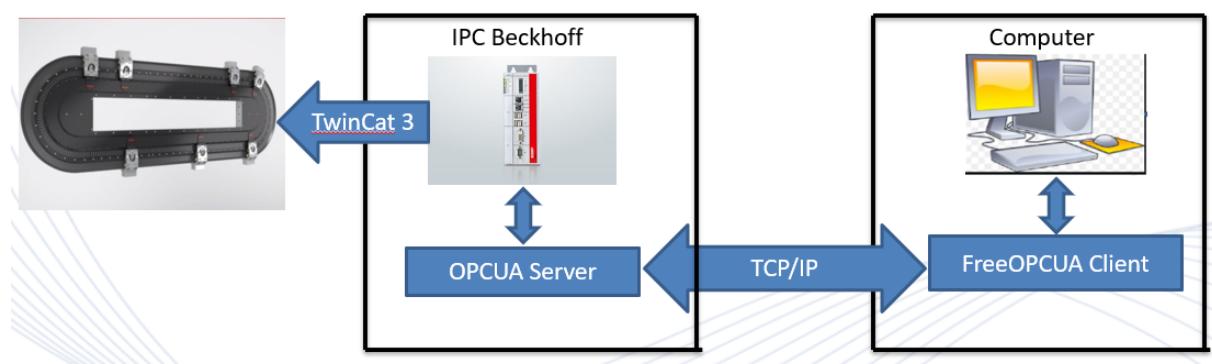


Abbildung 17: XTS – Computer Kommunikation

Die Kommunikation in diesem Teil des Projektes wird zwischen dem BECKHOFF IPC und dem Computer über ein Ethernet-Kabel realisiert. Auf dem IPC befindet sich das TwinCat 3 Programm mit seinem OPC UA Server und auf dem Rechner läuft der OPC UA Client.

Außerdem wird ein Open Source FREE OPCUA PYTHON auf den Computer heruntergeladen. [\[37\]](#)

3.3.1. OPC UA Server TF6100

Die Funktion TF6100 installiert einen OPC UA Server (und Client) auf dem Gerät. Der Server wird benötigt, um den OPC-UA-Clients die SPS-Methoden zur Verfügung zu stellen. Wenn das SPS-Programm auf die Runtime heruntergeladen wird, importiert der OPC UA Server automatisch die erste SPS-Runtime in seinen Namensraum. Alle SPS-Variablen und -Methoden, die als über OPC UA verfügbar gekennzeichnet sind, werden in den OPC UA-Namensraum importiert und stehen den Clients zur Verfügung. [\[38\]](#)

Der TwinCAT OPC UA Server stellt eine standardisierte Kommunikationsschnittstelle zur Verfügung, um auf Symbolwerte aus der TwinCAT-Laufzeit zuzugreifen. Dies erleichtert die Integration von Drittanbieter-Software zum Lesen oder Schreiben von Variablenwerten. [\[39\]](#)

Der TwinCat OPC UA Server unterstützt OPC UA Funktionen wie Datenzugriff, Zugriff auf historische Daten, Alarme und Bedingungen. Er ermöglicht die Zwischenspeicherung von Daten auf dem Server, d. h. eine Unterbrechung der Kommunikationsverbindung führt nicht zum Verlust von Daten. [\[36\]](#)

Es bietet einen grafischen Konfigurator für die einfache Verwaltung von lokalen und entfernten OPC UA Servern. [\[36\]](#)

Der TwinCat OPC UA Server definiert benutzer- und gruppenbasierte Zugriffsrechte auf Namespace- und Knotenebene und unterstützt Client/Server-Zertifikate, um eine sichere Kommunikation aufzubauen. [\[36\]](#)

3.3.2. TwinCat OPC UA Server installieren

In diesem Abschnitt werden die Installationsschritte für die TwinCat 3 TF6100-Funktion für Windows-basierte Betriebssysteme beschrieben. [\[40\]](#)

Die Installationsdatei muss zunächst von der BECKHOFF-Website heruntergeladen werden.
<https://www.beckhoff.com/en-en/products/automation/twincat/tfxxxx-twincat-3-functions/tf6xxx-tc3-connectivity/tf6100.html>

Die Installationsschritte für TwinCat 3 TF6100 werden im Folgenden beschrieben [\[40\]](#):

- 1- Die Installationsdatei als Administrator ausführen.

Dazu wählen im Kontextmenü der Datei den Befehl Run As Admin. Dann öffnet sich der Installationsdialog.

- 2- Akzeptieren Sie zunächst die Endbenutzer-Lizenzvereinbarung und klicken Sie dann auf Next.

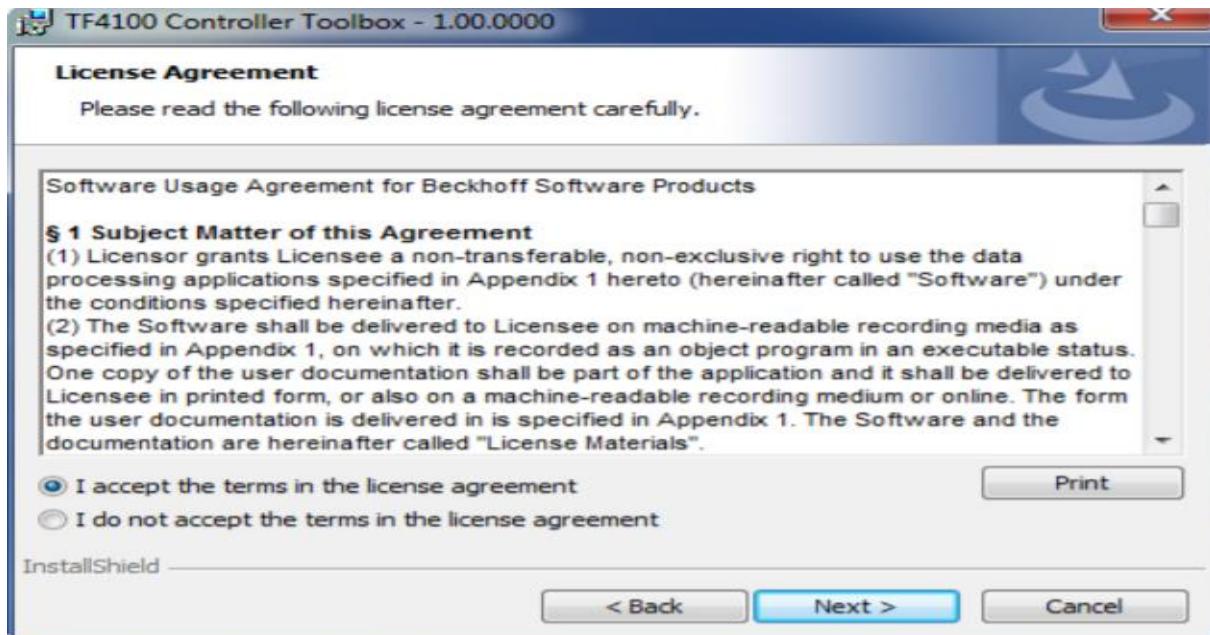


Abbildung 18: Akzeptieren der Lizenzvereinbarung

- 3- Die Benutzerdaten eingeben.

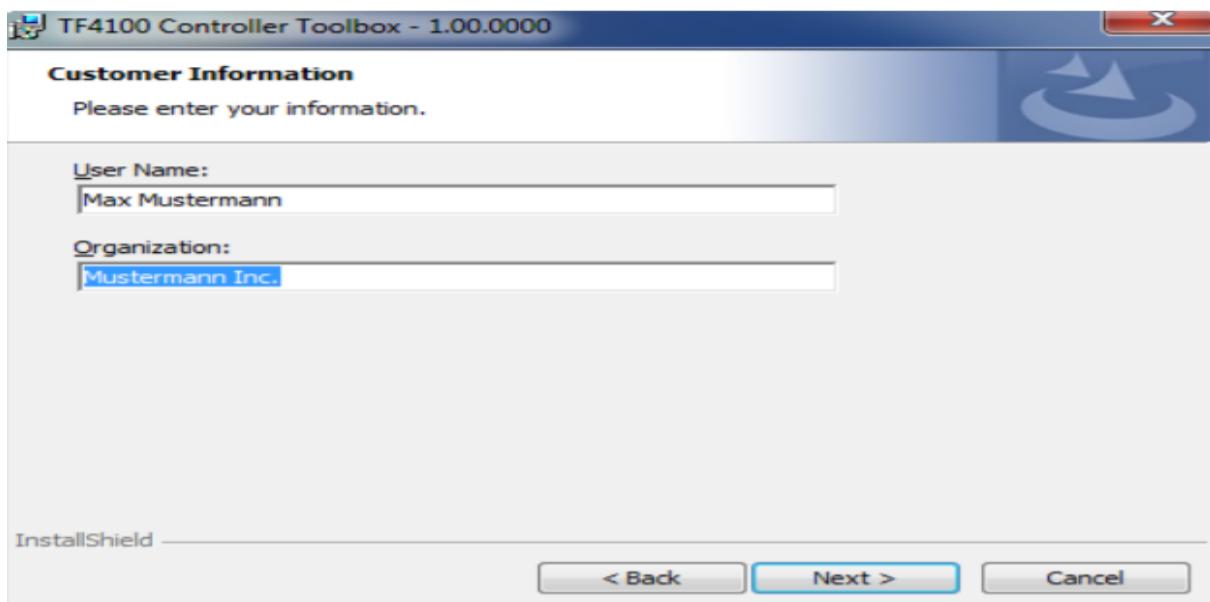


Abbildung 19: Benutzerdaten eingeben

- 4- Wenn Sie die vollständige Version von TwinCat 3 installieren möchten, dann markieren Sie diese als Installationstyp Complete. Wenn Sie einzelne Komponenten der TwinCat 3 Funktion installieren möchten, wählen Sie Custom.

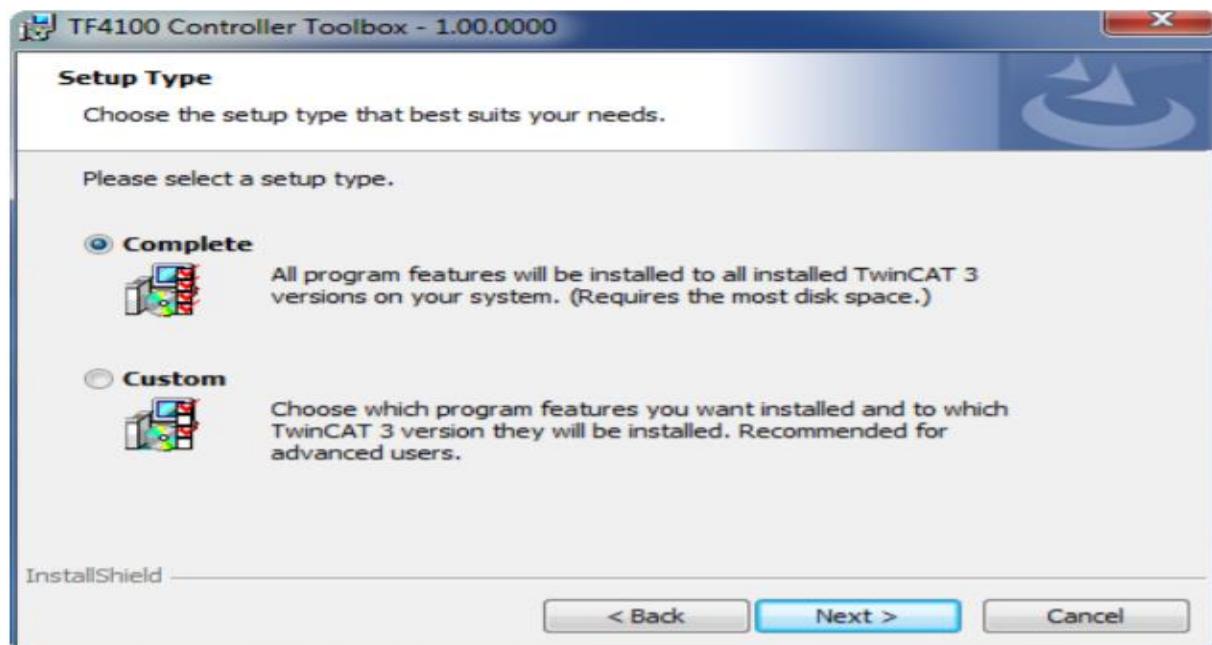


Abbildung 20: Vollversion von TwinCat 3 installieren

Bei diesem Projekt wurde die komplette Version der TF6100 TwinCat 3 Funktion gewählt.

- 5- Wählen Sie Next und dann Install, um die Installation zu starten.

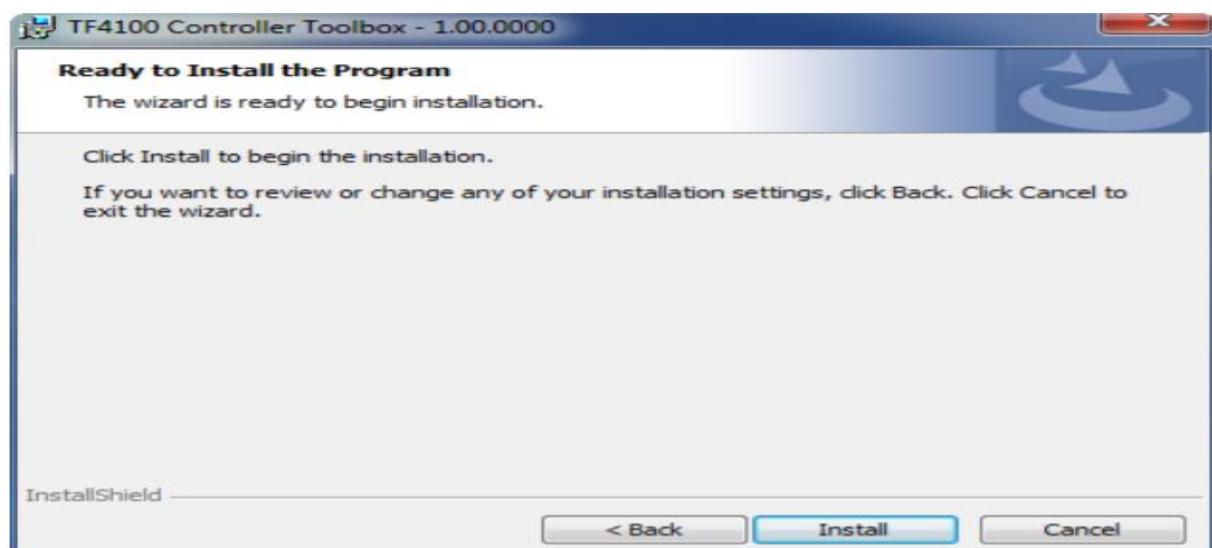


Abbildung 21: Installation starten

Eine Nachricht im Dialogfeld informiert darüber, dass das TwinCat-System heruntergefahren werden muss, um mit der Installation fortfahren zu können.

- 6- Bestätigen Sie den Dialog mit Yes.

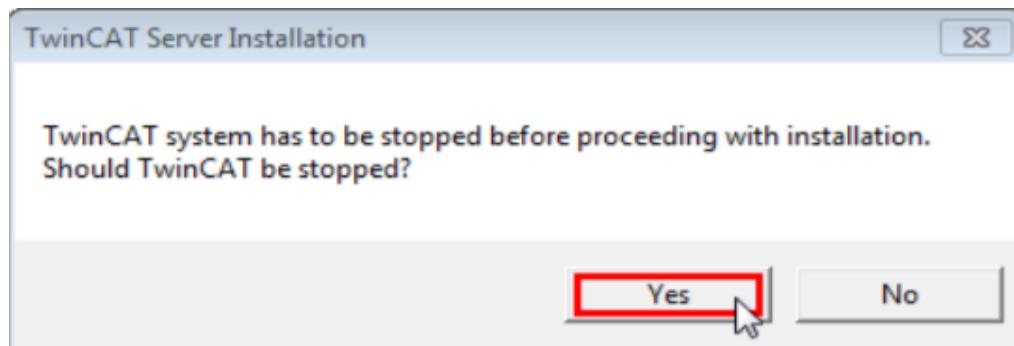


Abbildung 22: Dialog bestätigen

- 7- Wählen Sie Finish, um die Konfiguration zu beenden.

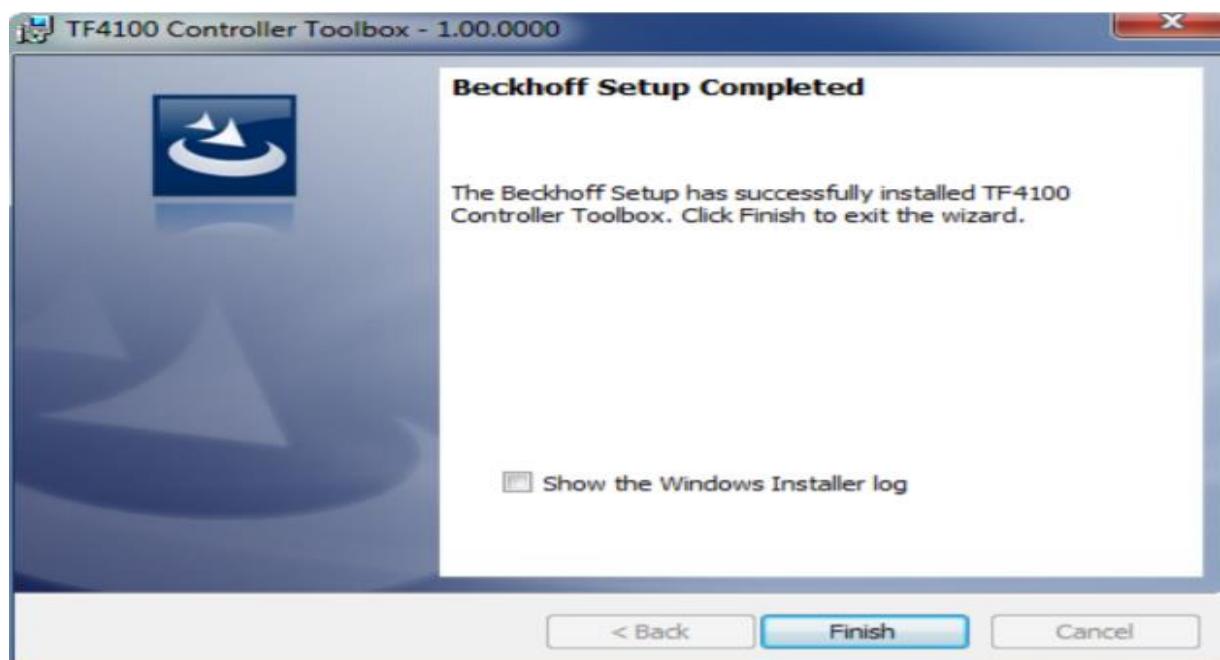


Abbildung 23: Konfiguration beenden

Nach diesen sieben Schritten ist die TwinCAT 3 Funktion erfolgreich installiert worden.

3.3.3. TwinCat OPC UA Server lizenzieren

Die TwinCat OPC UA Serverfunktion kann auf zwei Arten aktiviert werden, entweder als Vollversion oder als 7 Tage verlängerbare Testversion. Beide Lizenztypen können über die TwinCat 3 XAE Entwicklungsumgebung aktiviert werden. [41]

Bei diesem Projekt wurde die Entscheidung für die Aktivierung der Lizenz als 7-tägige erneuerbare Testversion gewählt.

Die Lizenzierung der 7-Tage-Testversion einer TwinCAT-3-Funktion erfolgt durch die folgenden Schritte [41]:

- 1- Die TwinCAT 3 Entwicklungsumgebung (XAE) starten.
- 2- Öffnen Sie ein bestehendes TwinCAT 3 Projekt oder erstellen Sie ein neues Projekt.
- 3- Machen Sie Doppelklicken im Solution Explorer auf License im Unterbaum SYSTEM.

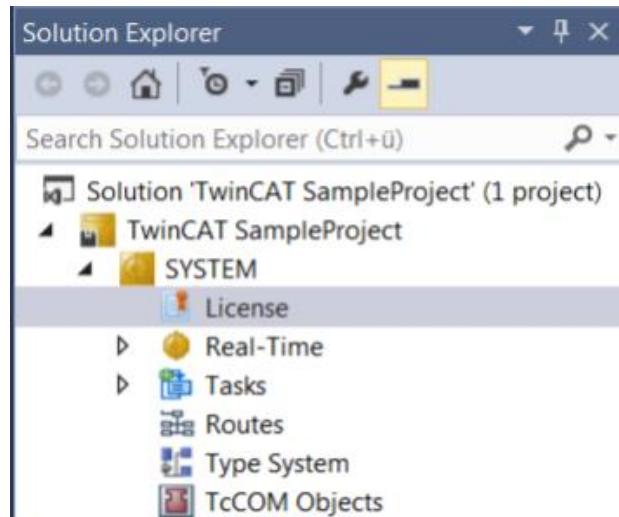


Abbildung 24: Solution Explorer

Der TwinCAT 3 License Manager öffnet sich.

- 4- Öffnen Sie die Registerkarte Manage Licenses. Aktivieren Sie in der Spalte Add License das Kontrollkästchen für die Lizenz, die dem Projekt hinzugefügt werden soll.

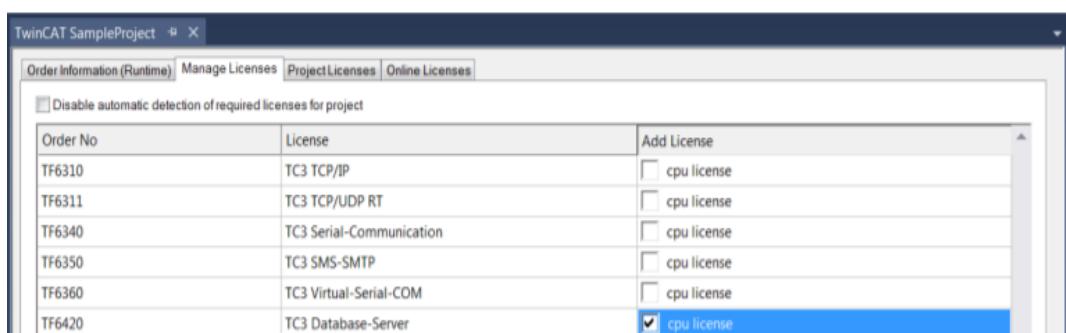


Abbildung 25: Manage Licenses

5- Öffnen Sie das Register Informationen zur Runtime.

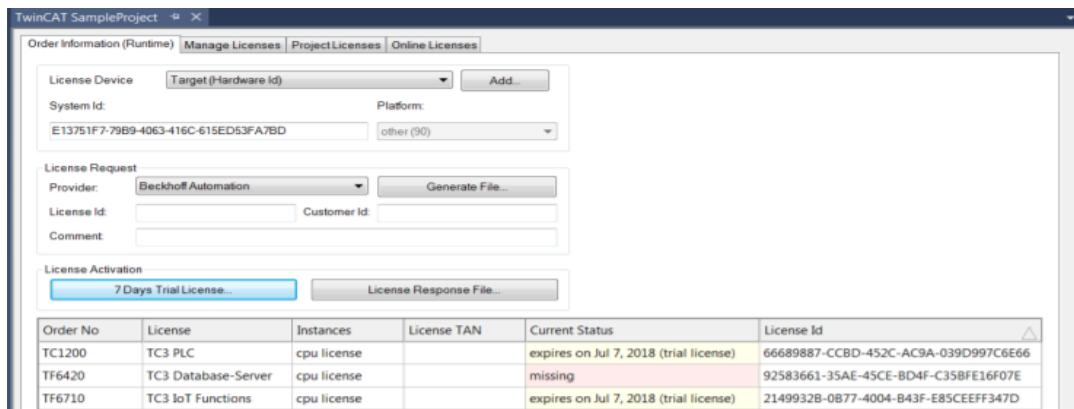


Abbildung 26: Informationen zur Runtime

In der tabellarischen Lizenzübersicht wird die zuvor ausgewählte Lizenz mit dem Status "missing" angezeigt.

6- Klicken Sie auf 7-Day Trial License, um die 7-Tage-Probelizenz zu aktivieren.

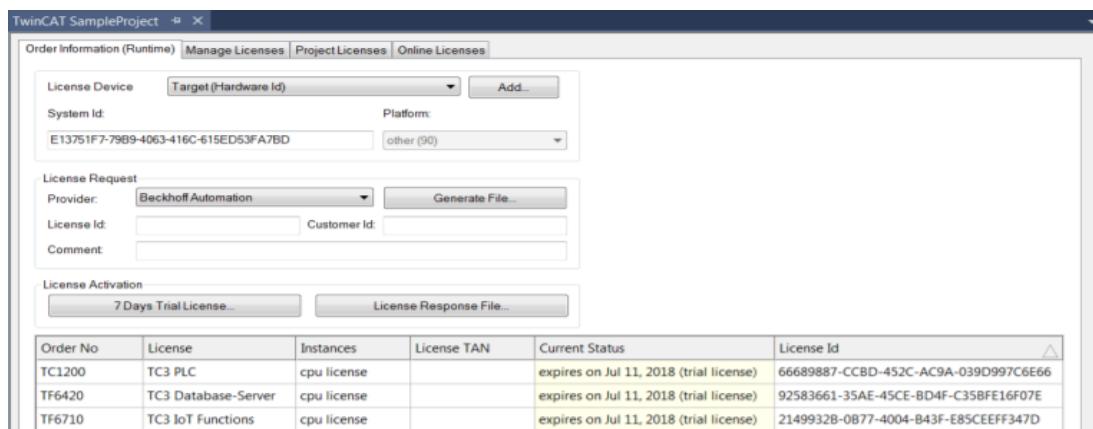


Abbildung 27: 7 Tage Testlizenz aktivieren

In der Übersichtstabelle der Lizenzen zeigt der Lizenzstatus nun das Ablaufdatum der Lizenz an.

7- Neustart des TwinCAT-Systems.

3.3.4. TwinCat OPC UA Server konfigurieren

Nach erfolgreicher Installation und Lizenzierung der TF6100-Funktion sind folgende Schritte erforderlich, um die SPS-Variablen über den TwinCAT OPC UA Server verfügbar aufzurufen. [42]

- Schritt 1: SPS-Variablen für den OPC-UA-Zugang konfigurieren. [42]

Öffnen Sie ein bestehendes SPS-Projekt und einfügen den folgenden Kommentar vor den ausgewählten Variablen.

```
{attribute 'OPC.UA.DA' := '1'}  
bVariable : BOOL;
```

Abbildung 28: Konfiguration der SPS-Variablen für den OPC-UA-Zugang

- Schritt 2: Herunterladen der Symboldatei konfigurieren. [42]

Standardmäßig baut der TwinCAT OPC UA Server eine Verbindung zur ersten SPS-Laufzeit auf dem lokalen System auf und verwendet die entsprechende Symboldatei zum Aufbau des Namensraums.

Um die Symboldatei verfügbaren zu können, muss der Symboldateidownload in den Einstellungen des SPS-Projekts aktiviert werden.

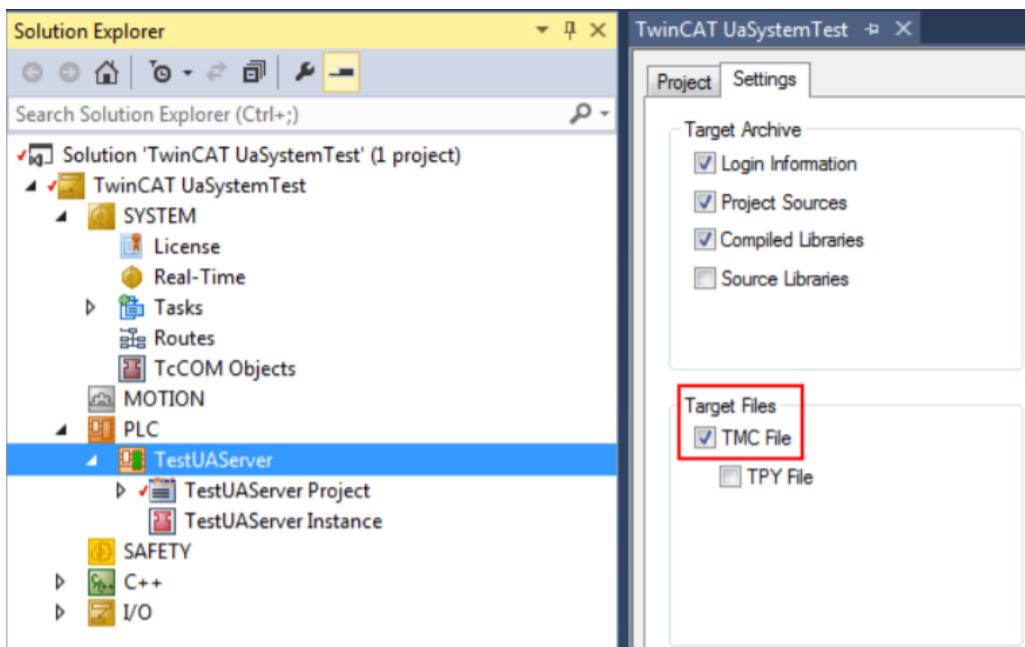


Abbildung 29: Konfiguration der Symboldatei

- Schritt 3: Lizenz für den Server aktivieren (siehe Abschnitt [3.3.3](#)). [\[42\]](#)

- Schritt 4: Erstellung eines neuen Projekts [\[43\]](#)

Das Projektpaket des OPC UA Konfigurators ist in das sogenannte Connectivity-Paket integriert.

Die Projekt-Vorlage "TwinCAT Connectivity Project":

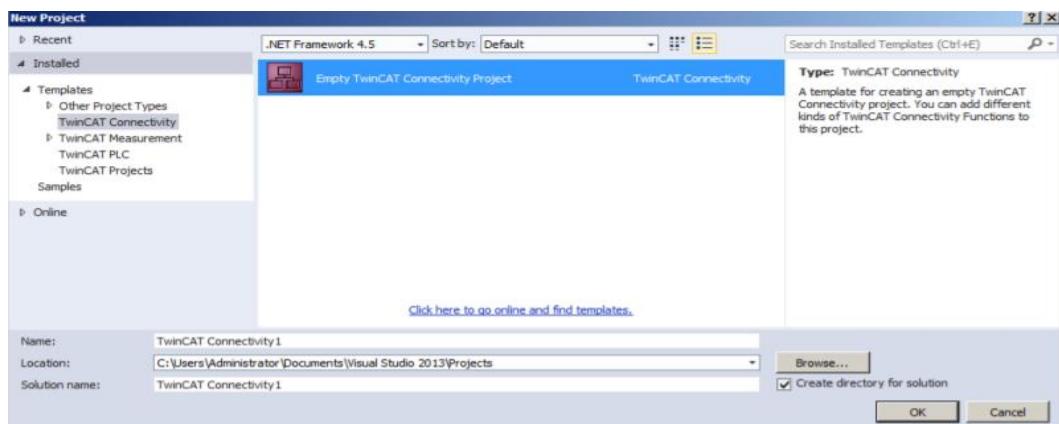


Abbildung 30: TwinCAT Connectivity-Projekt

Die Projektvorlage "TwinCAT OPC-UA Server Project":

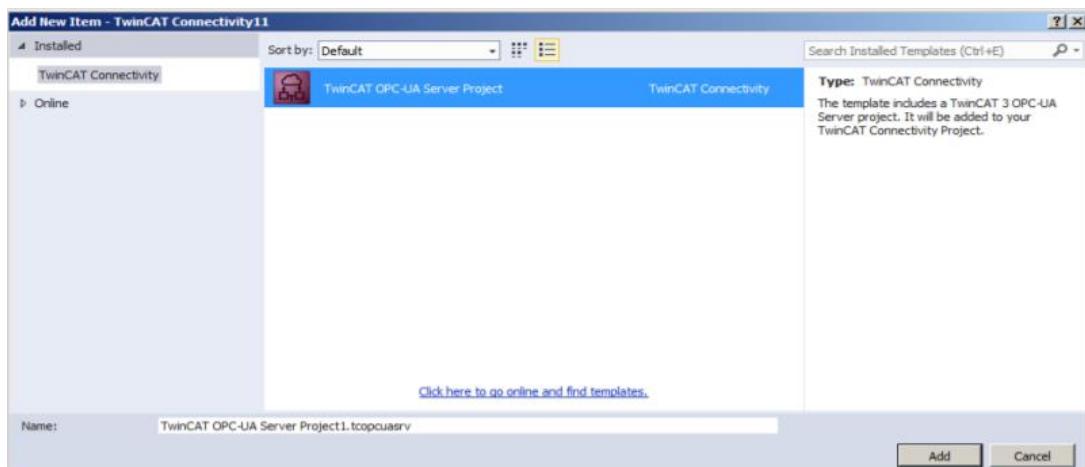


Abbildung 31: TwinCAT OPC-UA Server Projekt

- Schritt 5: ADS-Geräte hinzufügen [\[44\]](#)

Der OPC-UA-Server kann mit einem oder mehreren ADS-Geräten "kommunizieren". Um eine Verbindung herzustellen, ist eine Verbindung zu dem jeweiligen ADS-Gerät erforderlich. Im OPC UA Configurator werden in der Facette Data Access ADS Geräte angelegt, konfiguriert und dem OPC UA Server bekannt gegeben.

Neue ADS-Geräte werden der Konfiguration über den Befehl Add New Device Type im Kontextmenü hinzugefügt.

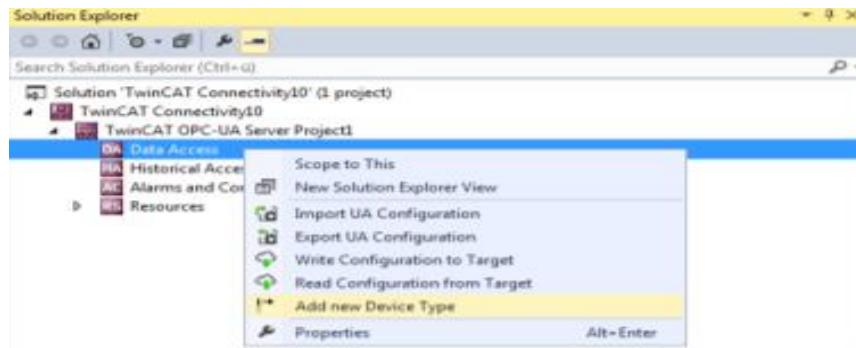


Abbildung 32: Neuen Gerätetyp hinzufügen

Wenn der Befehl ausgeführt wird, öffnet sich ein Dialogfenster, in dem die Verbindungsparameter für dieses Gerät konfiguriert werden können, z.B. AMS Net ID, ADS Port oder Symboldatei.

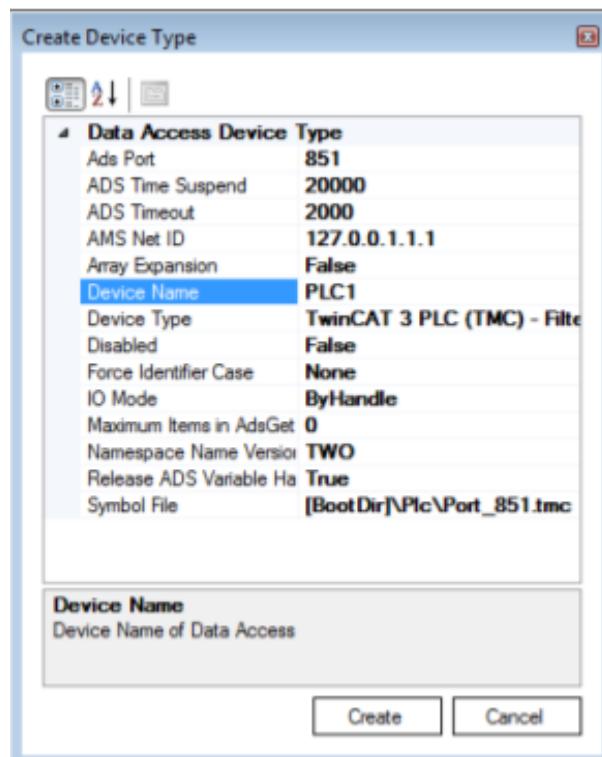


Abbildung 33: Konfiguration der Verbindungsparametern

-Schritt 6: Konfigurieren des Endpunkts [\[45\]](#)

Die OPC UA Server-Endpunkte legen fest, welche Sicherheitsmechanismen beim Aufbau einer Client-Verbindung verwendet werden sollen. Diese reichen von "unverschlüsselt" bis "verschlüsselt und signiert" und basieren auf unterschiedlichen Schlüsselstärken.

Endpunkte können über den Konfigurator aktiviert und deaktiviert werden.

Die Endpunkte werden direkt auf der Projektebene des OPC UA Servers konfiguriert. Durch Doppelklick auf das Projekt können die entsprechenden Einstellungen auf der Registerkarte UA-Endpunkte vorgenommen werden.

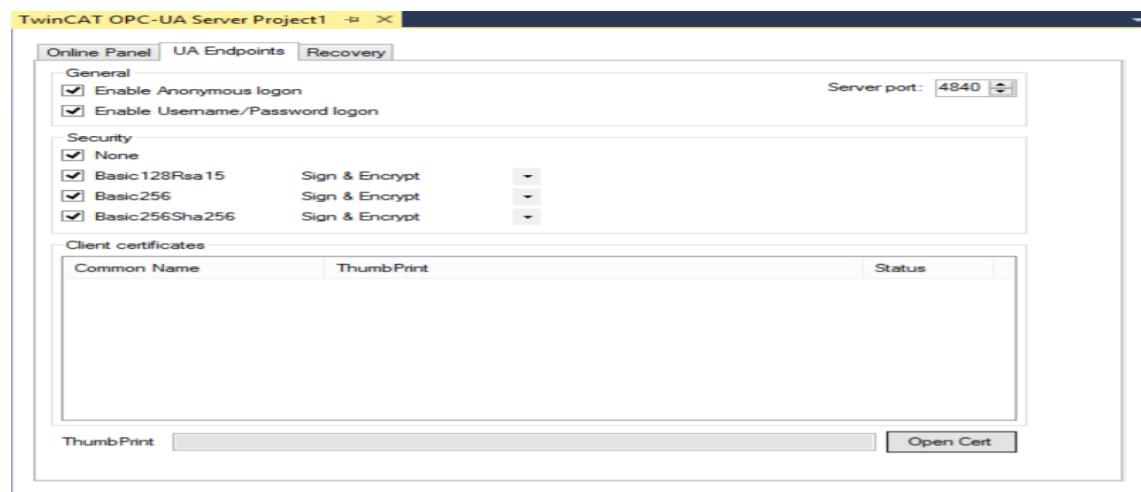


Abbildung 34: Einstellungen für UA-Endpunkte

-Schritt 7: Verbindung mit dem OPC UA Server herstellen [42]

Um eine Verbindung von einem OPC UA Client aufzubauen, muss der Client eine Verbindung mit der URL des OPC UA Servers aufbauen. z.B. opc.tcp://192.168.1.1:4840.

Nach erfolgreicher Verbindung zum Server finden sich die SPS-Variablen unter dem Objekt PLC1.

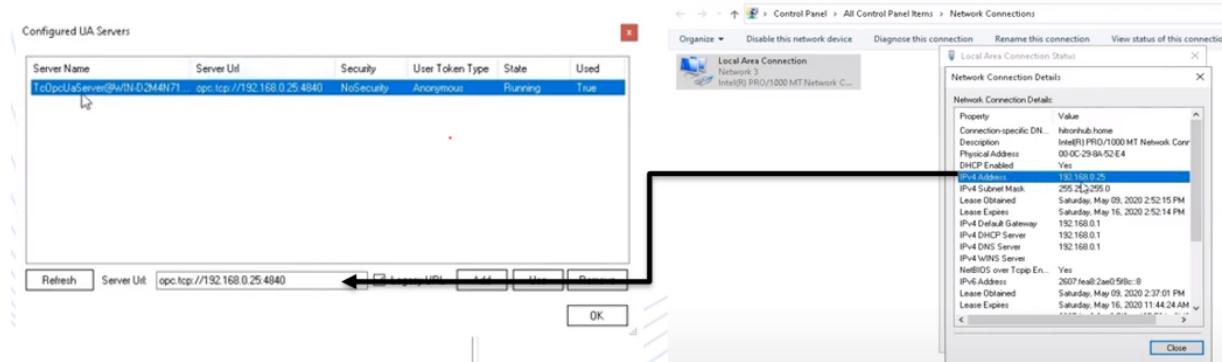


Abbildung 35: Verbindung zum OPC UA-Server

-Schritt 8: Konfigurieren von Einstellungen für das Zertifikatsvertrauen [46]

Der Konfigurator erleichtert die Verwaltung von Client-Zertifikaten auf dem Server. In den Projekt-Einstellungen auf der Registerkarte UA-Endpunkte im Bereich Client-Zertifikate können Zertifikate als vertrauenswürdig eingestuft oder abgelehnt werden.

Nachdem ein OPC UA Client zum ersten Mal versucht hat, sich mit einem sicheren Endpunkt auf dem Server zu verbinden, wird das Client-Zertifikat auf dem Server gelöscht und als "abgelehnt" deklariert. Der Serveradministrator kann dann das Zertifikat aktivieren. Ein anschließender Versuch des Clients, eine Verbindung zu einem sicheren Endpunkt herzustellen, wird dann erfolgreich sein.

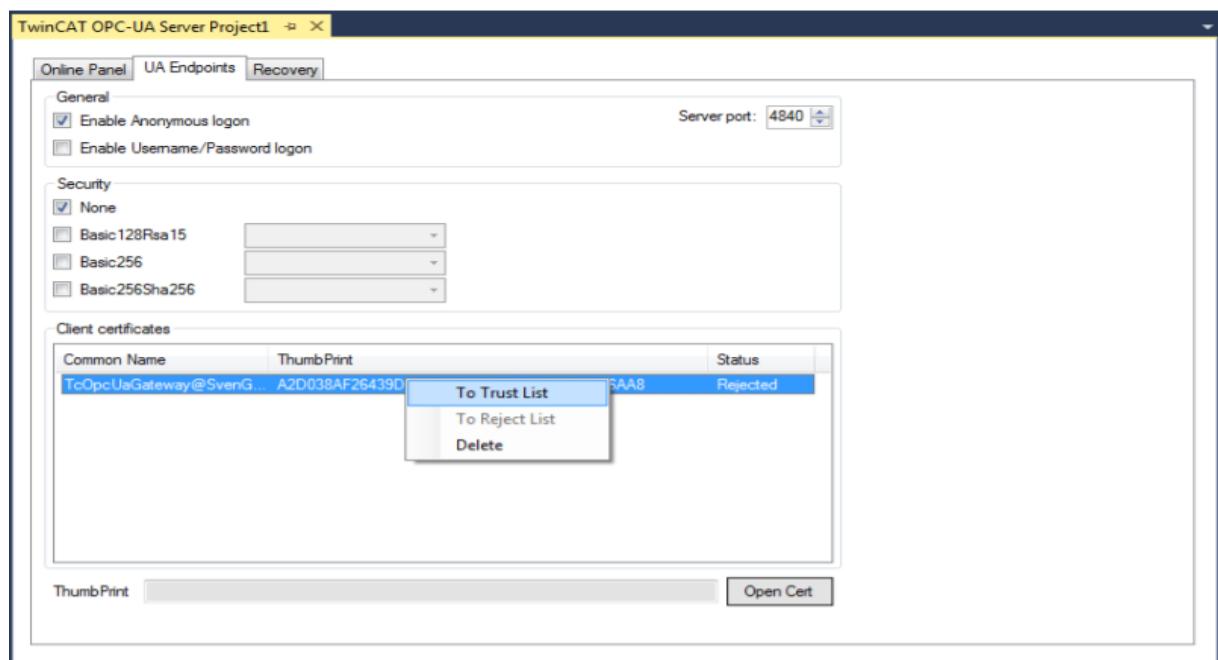


Abbildung 36: Konfiguration der Einstellungen für das Zertifikatvertrauen

3.4. Python Applikation

In diesem Projekt wurde die Programmiersoftware Python (siehe Abschnitt 2.3.) verwendet. Für die Kommunikationsphase des Projekts wurde auf dem Computer Python in der Version 3.9 installiert, wobei zwei Python-Bibliotheken installiert wurden, damit der Computer die Rolle eines Clients zwischen den beiden Servern KUKAVARPROXY des KRC4 Compact Roboters und dem OPC UA Server des XTS BECKHOFF übernehmen kann.

Beide Bibliotheken sind zum kostenlosen Herunterladen und Installieren verfügbar.

Die erste Bibliothek ist FREE OPC UA Python, sie ist als Open Source auf der Website Github.com unter dem Link <https://github.com/FreeOpcUa/python-opcua> verfügbar.

Die zweite Bibliothek ist Python KUKAVARPROXY Message Format, sie ist ebenfalls als Open Source auf der Website Github.com unter dem Link <https://github.com/ahmad-saeed/kukavarproxy-msg-format> verfügbar.

Die folgende Abbildung zeigt die Kommunikation des Python-Clients mit den beiden KVP-Servern des Roboters und dem OPC-UA-Server des XTS.

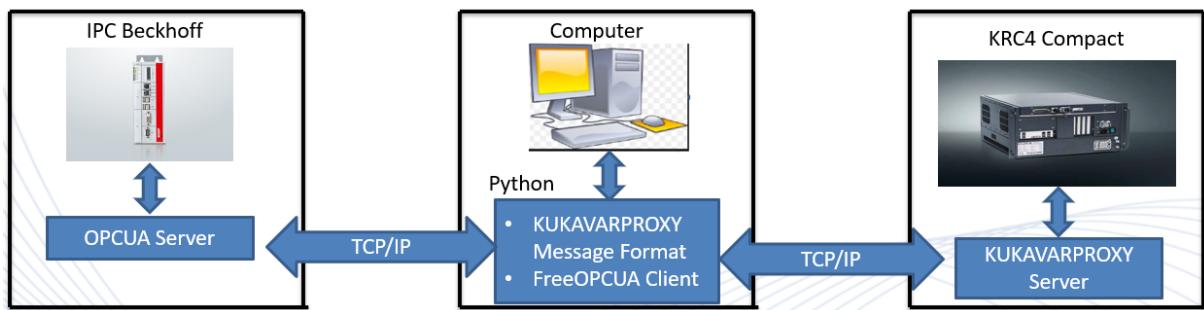


Abbildung 37: Kommunikationsstruktur des Python-Clients mit den beiden Servern

Um die Python-Applikation und ihre Rolle als Client, der mit den beiden Servern kommuniziert, besser zu verstehen, wird der Erklärungsteil in zwei Unterabschnitte unterteilt. Der erste Teil konzentriert sich auf den Kommunikationsteil zwischen der Python-Applikation und KUKAVARPROXY und der zweite Teil auf die Erläuterung der Kommunikation der Python-Applikation mit dem OPC-UA-Server des XTS.

3.4.1. Kommunikation der Python-Applikation mit dem KUKAVARPROXY-Server

Nach der Implementierung des KUKAVARPROXY-Servers in die KRC4-Compact-Steuerung des Roboters (siehe Abschnitt [3.2.](#)) und dem Herunterladen der KUKAVARPROXY Message Format Bibliothek in das Python-Programm, muss die Python-Hauptseite geöffnet werden, um die Applikation zu starten.

Auf dieser Seite wurde die Python-Bibliothek KUKAVARPROXY Message Format importiert und die IP-Adresse des Roboters 172.31.1.147 und der Port 7000 festgelegt, damit der Client mit dem im Roboter implementierten KVP-Server kommunizieren kann.

```

# KUKAVARPROXY Message Format
import socket # Used for TCP/IP communication

# Initializing client connection

class KUKA(object):
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    ID=1
    def __init__(self, TCP_IP):
        try:
# Open socket. kukavarproxy actively listens on TCP port 7000
            self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            self.client.connect((TCP_IP, 7000))

        except:
            self.error_list(1)
            self.ID = 1
        else:
            self.ID = 0
# Home Page
from kukavarproxy import *

```

Abbildung 38: Implementierung des KUKAVARPROXY Message Format in der Python Client-Applikation

Es wurde ein Kommunikationstest durchgeführt, um sicherzustellen, dass die Kommunikation funktioniert hat.

Das Ergebnis ist, dass der gebaute Client mit dem Roboter-Server kommuniziert.

Dieses Ergebnis wird durch die Kontrolle der Schnittstelle des KUKAVARPROXY-Servers im SmartPad des Roboters verifiziert, und daher wird ein Client über den KVP-Server mit dem Roboter verbunden, wie in der folgenden Abbildung gezeigt wird, die dem SmartPad des Roboters entnommen wurde.

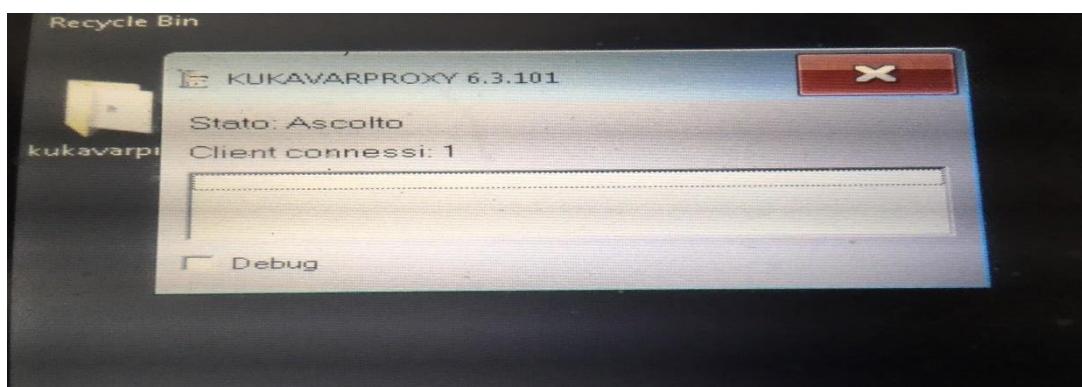


Abbildung 39: KVP verbunden mit dem Client

3.4.2. Kommunikation der Python-Applikation mit dem OPC-UA-Server des XTS

Nachdem der XTS OPC UA Server durch die TwinCat 3 Software implementiert wurde (siehe Abschnitt [3.3.](#)) und die FREE OPC UA PYTHON Bibliothek in das Python Programm heruntergeladen wurde, beginnt der Aufruf der OPC UA Client auf der Hauptseite der Python Applikation aus der vorherigen Bibliothek.

Dann wurde die IP-Adresse 169.254.146.109 des XTS OPC UA Servers und sein Port 4840 ermittelt, damit der OPC UA Python Client mit dem XTS OPC UA Server kommunizieren kann.

```
# Home Page  
  
from opcua import Client, ua  
url = "opc.tcp://169.254.146.109:4840"
```

Abbildung 40: Implementierung von FREE OPC UA PYTHON in der Python Client-Applikation

Es wurde ein Kommunikationstest durchgeführt, um sicherzustellen, dass die Kommunikation erfolgreich ist.

Das Ergebnis ist, dass der OPC UA Client mit dem OPC UA Server des XTS kommuniziert.

Dieses Ergebnis wird überprüft, indem eine Integer-Variable im TwinCat 3-Programm des XTS erstellt und von der erstellten Python-Applikation aufgerufen wird, außerdem wird ihr Wert geändert und auf dem Server überprüft, ob er sich geändert hat oder nicht.

Das Ergebnis ist, dass die Variable geändert wurde und die Kommunikation zwischen dem Python OPC UA Client und dem XTS OPC UA Server bereits erfolgt ist.

4. Implementation der Kommunikation Infrastruktur

4.1. Überblick

Nach der technischen Untersuchung der Kommunikation in Teil 3, wird in diesem Teil des Projekts die Implementierung der Kommunikationsinfrastruktur zwischen dem KUKA-Roboter und der XTS BECKHOFF-Maschine, über das Kommunikationsprotokoll OPC UA, untersucht.

Um diesen Teil besser zu verstehen, wird eine Applikation erstellt, die den Datenaustausch zwischen den beiden Maschinen ermöglicht und dabei die Anforderungen des Projektproblems berücksichtigt, d.h. die Kommunikation zwischen den beiden Maschinen unter Verwendung des OPC UA Kommunikationsprotokolls, genauer gesagt, wie der Datenaustausch zwischen der Python-Applikation, die die Rolle des OPC UA Clients spielt, und den beiden KUKAVARPROXY Message Format Servern und dem OPC UA Server des XTS.

Um die tatsächliche Applikation besser zu verstehen, wird ein allgemeiner Überblick gegeben.

Die XTS-Maschine ist eine Standard-XTS von 2000 mm Länge mit fünf Movers vom Typ AT9011-1240.



Abbildung 41: XTS-Maschine

Wenn sich der Mover in der Fixstation befindet, erhält der OPC-UA-Client vom OPC-UA-Server des XTS die Information, dass sich der Mover in dieser Position befindet. Dann sendet der Client eine Information an den KUKAVARPROXY-Server des Roboters, damit dieser, je nach Zustand des Mover (voll) oder (leer), ein Objekt auf den Mover legt. Nachdem der KUKA-Roboter seine Aufgabe beendet hat, erhält der Client Informationen vom KVP-Server und der Client sendet sie an den OPC-UA-Server des XTS. Danach begibt sich der Mover zu einer Wartestation und der nächste Mover bewegt sich zur Fixstation und durchläuft die gleiche Vorgehensweise. Der Mover in der Wartestation fährt dann in einer bestimmten Reihenfolge zur Position der Fixstation zurück.

Die Bewegung des Movers unterliegt einer gut vorbereiteten Ordnung, um alle Ausfälle und Risiken zu vermeiden.

Zum besseren Verständnis des gerade beschriebenen Verlaufs, folgt nun eine detaillierte Beschreibung.

Der erste Teil beschäftigt sich mit der Programmierung des Roboters, der zweite Teil konzentriert sich auf die Programmierung des XTS und der dritte Teil behandelt die Python-Programmierung auf dem Computer.

4.2. Programmierung des KUKA Roboters

Der Teil der Roboterprogrammierung ist in drei Teile gegliedert. Der erste Teil konzentriert sich auf die Sicherheitsanforderungen an den KUKA Roboter. Der zweite Teil beschreibt die Konfiguration der Variablen, die zwischen dem KUKAVARPROXY Roboterserver und dem OPC UA Client übertragen und empfangen werden sollen. Der dritte Teil behandelt das Programm, das für die Applikation geschrieben wurde.

4.2.1. Robotersicherheit

Jede missbräuchliche Verwendung des Roboters kann zu Sach- und Personenschäden führen. Aus diesen Gründen ist es notwendig, vor dem Einsatz des Roboters die notwendigen Sicherheitseinstellungen vorzunehmen.

4.2.1.1. Begrenzung des Arbeitsraums

Der erste Sicherheitsparameter ist die Begrenzung des Arbeitsraums des Roboters, d. h. die Begrenzung der Bewegungen der Roboterachsen, so dass die Roboterbewegungen keine Gefahr für die Arbeitsumgebung darstellen (siehe Abschnitt [2.5.3](#)).

In diesem Projekt wird der Arbeitsbereich des Roboters durch die Begrenzung der Bewegungen dieser sechs Drehachsen begrenzt, wie in der folgenden Tabelle dargestellt.

Achse	Bewegungsbereich
A1	+/-120°
A2	+0° bis -145°
A3	+156° bis -45°
A4	+/-185°
A5	+/-120°
A6	+/-180°

Tabelle 4: Bewegungsbereich der Achsen

4.2.1.2. Backup-Manager

Nach der Begrenzung des Arbeitsraums und bevor das Roboterprogramm für die Applikation geschrieben wird, muss der Schritt Backup-Manager durchlaufen werden.

Mit dem Backup-Manager können Projekte, Optionspakete und DRC-Daten gesichert und wiederhergestellt werden.

Bei Verlust von Parameterdaten oder alten Projekten ist der Backup-Manager in der Lage, diese Daten wiederherzustellen.

Die Schritte zur Konfiguration des Backup-Managers sind wie folgt:

- 1- Voraussetzung: Benutzergruppe Experte / Betriebsart T1 oder T2
- 2- Vorgehensweise: Im Hauptmenü Datei → Backup-Manager → Backup-Konfiguration wählen
- 3- Backup-Konfiguration enthält die allgemeinen Einstellungen. Außerdem kann hier bei Bedarf die automatische Sicherung konfiguriert werden.

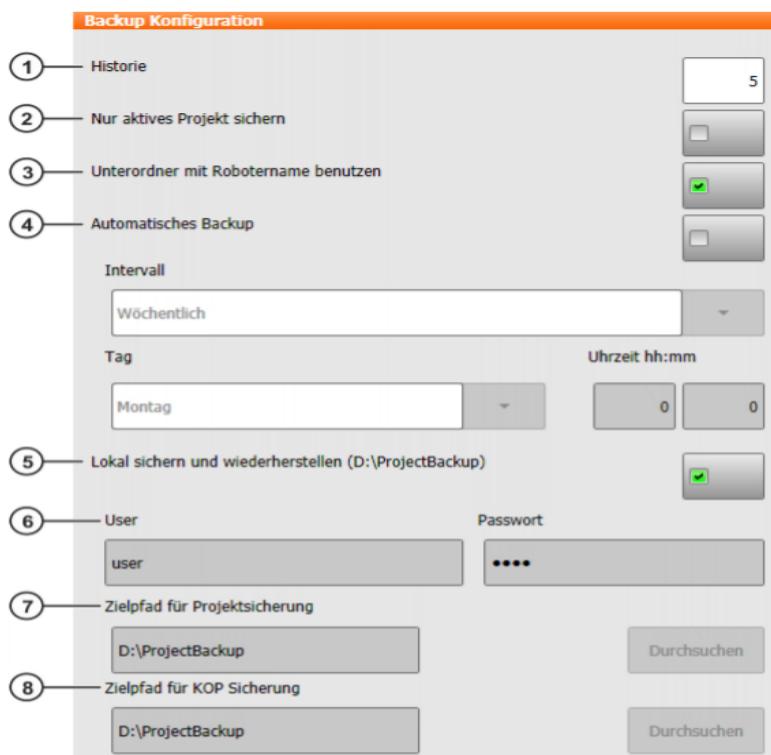


Abbildung 42: Registerkarte Backup-Konfiguration

Unter Signalschnittstelle kann bei Bedarf die E/A Ansteuerung konfiguriert werden.

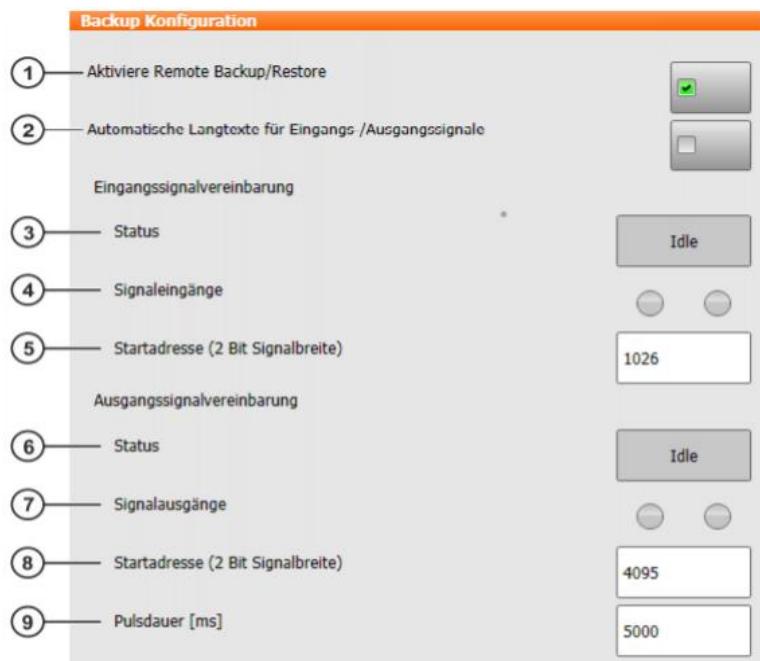


Abbildung 43: Registerkarte Signalschnittstelle

4- Sicherungsmethode

Erforderliche Schritte zur Durchführung eines Backups bei einem Kuka-Roboter mit der Steuerung KR C4.

- USB in den Controller einstecken
- unter Hauptmenü → Konfiguration → Benutzergruppe → Experte wählen
- Unter Hauptmenü → Datei → Archiv → USB(Kabine) → alle/ Applikation/ Systemdaten/ Protokoll-daten wählen.

5- Wiederherstellungsmethode

- USB-Stick mit den Sicherungsdateien in den Schalschrank einstecken
- Unter Menü → Datei → Wiederherstellen → USB(Steuerung) → Alle / Anwendung / Systemdaten wählen.

4.2.1.3. KUKA RecoveryUSB [90]

Die IP-Adresse des Roboters ist eine feste Adresse, falls der Benutzer diese IP-Adresse ändert, wird das Betriebssystem des Roboters nicht mehr funktionieren. Um es wiederherzustellen, verwendet man KUKA RecoveryUSB.

KUKA.RecoveryUSB ist eine Archivierungssoftware, die auf der Erstellung und Wiederherstellung eines Festplatten-Images (Backup) für eine Robotersteuerung basiert.

KUKA.RecoveryUSB kann nur für die folgenden Anwendungen verwendet werden:

- Zum Einspielen von nicht finalisierten Master- oder Finalisierten-Images (neues Aufsetzen einer Steuerung)
- Zur Erstellung und Wiederherstellung eines Festplatten-Images einer Steuerung (zur Sicherung einer vorhandenen Konfiguration, Backup)

Wenn der KUKA.RecoveryUSB-Stick mit der Robotersteuerung verbunden ist, erscheint diese Benutzeroberfläche. Bei der Robotersteuerung KR C4 Compact muss ein externer Bildschirm angeschlossen werden.

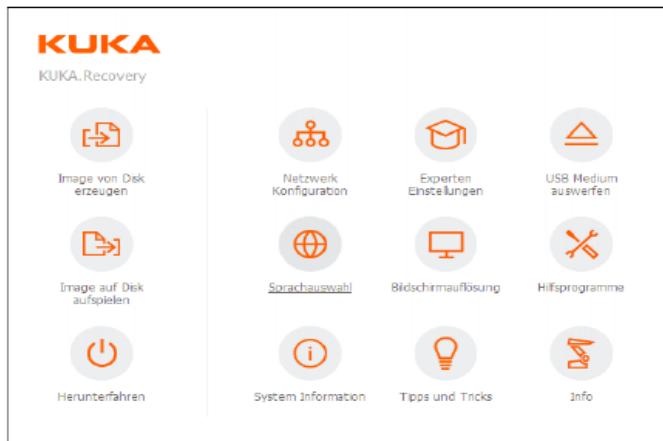


Abbildung 44: Bedienoberfläche - Robotersteuerung

1- Image erstellen und wiederherstellen

- Image auf Disk aufspielen

Stellen Sie mit der Schaltfläche "Image auf Disk aufspielen " eines Festplatten-Images wieder her, indem die folgenden Schritte ausführen:

- klicken Sie auf die Schaltfläche Image auf Disk aufspielen. Ein Fenster wird geöffnet
- Image-Datei und das Laufwerk auswählen
- Auf OK klicken
- Es erscheint eine Meldung, die darauf hinweist, dass die Festplatte überschrieben wird. Die Meldung mit OK bestätigen.

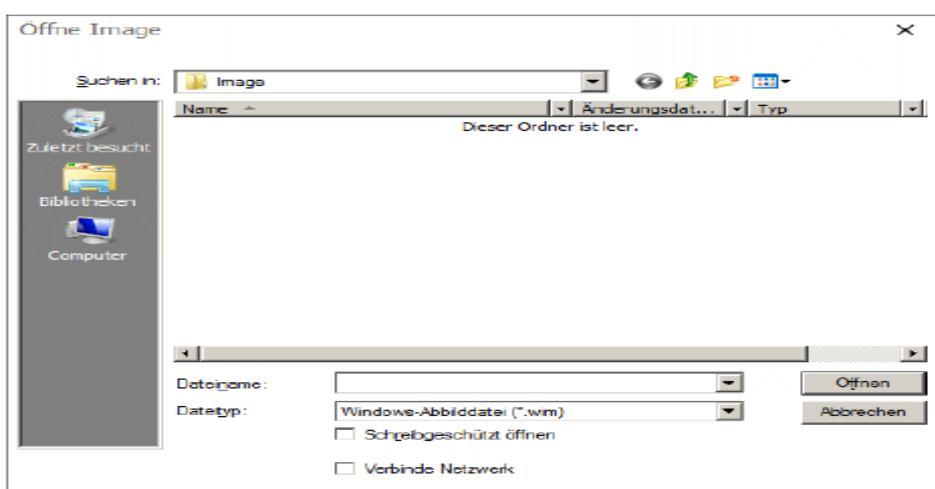


Abbildung 45: Image auf Disk aufspielen – Robotersteuerung

➤ Image von Disk erzeugen

Erstellen Sie eines Festplatten-Image mit der Schaltfläche "Abbild von Festplatte erstellen", indem Sie die folgenden Schritte ausführen:

- Im Menü auf Image von Disk erzeugen klicken
- Speicherort für das Image auswählen
- Auf OK klicken

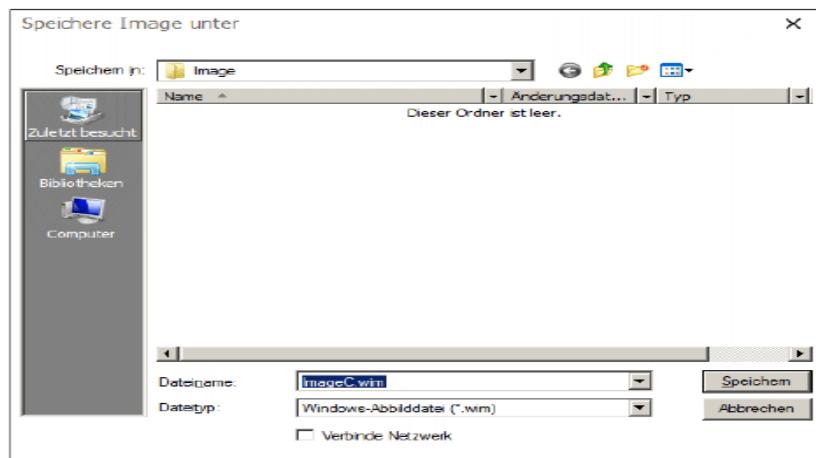


Abbildung 46: Image von Disk erzeugen – Robotersteuerung

2- Netzwerk Konfiguration

➤ Registerkarte TCP/IP Adressen

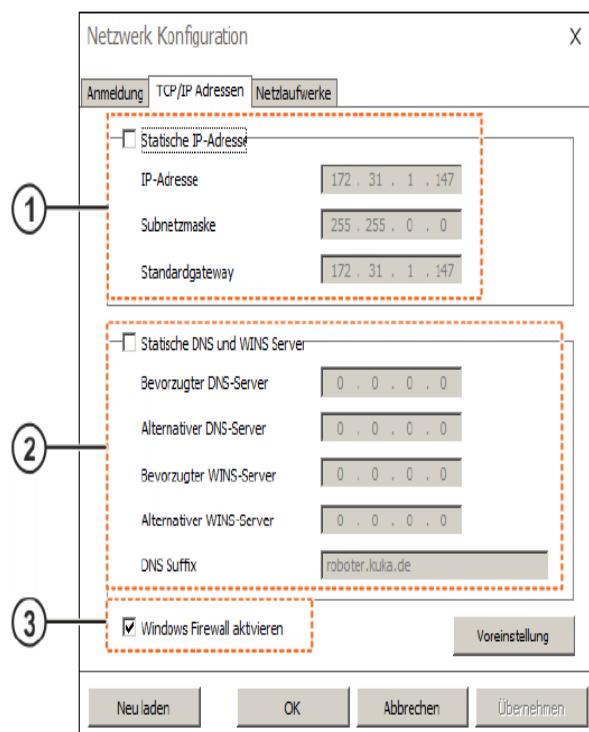


Abbildung 47: Registerkarte TCP/IP Adressen – Robotersteuerung

- 1- Statische IP-Adresse: Hier wird ausgewählt, ob eine statische IP-Adresse verwendet werden soll.
 - 2- Statische DNS und WINS Server: Hier wird ausgewählt, ob eine statische Konfiguration für DNS und WINS Server verwendet werden soll.
 - 3- Windows Firewall aktivieren: Hier wird ausgewählt, ob bei einer aktiven Netzwerkverbindung, die Windows Firewall aktiv oder nicht aktiv sein soll.
- Experten Einstellungen

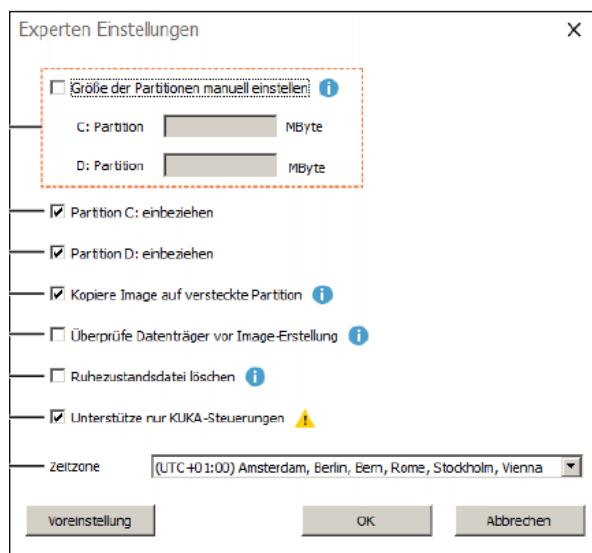


Abbildung 48: Registerkarte Experten
Einstellungen – Robotersteuerung

- Partition C: einbezogen wird die Partition C bei dem Erstellen oder Wiederherstellen von Images berücksichtigt
- Partition D: einbezogen wird die Partition D bei dem Erstellen oder Wiederherstellen von Images berücksichtigt
- Kopiere Image auf versteckte Partition: Die Image-Dateien, die beim Wiederherstellen angewendet werden
- Unterstütze nur KUKA-Steuerungen: Hier kann konfiguriert werden, ob KUKA.RecoveryUSB auch mit einer unbekannten Hardware betrieben werden darf
 - 1- Automatischer Modus ausführen

Je nach Konfiguration wird im automatischen Modus ein Image erstellt oder wiederhergestellt

2- KR C4 Image im Automatischen Modus erstellen

- KUKA.RecoveryUSB Stick an der Robotersteuerung anstecken
- Robotersteuerung starten
- CSP LED 1 blinkt: KUKA.RecoveryUSB bootet die Steuerung
- CSP LED 1 leuchtet: Bootvorgang ist abgeschlossen
- CSP LED 2 blinkt: Image der C-Partition wird erstellt
- CSP LED 2 leuchtet: Imageerstellung der C-Partition ist abgeschlossen
- CSP LED 3 blinkt: Image der D-Partition wird erstellt
- CSP LED 3 leuchtet: Imageerstellung der D-Partition ist abgeschlossen
- CSP LED 1-6 leuchtet für 1 Sekunde: Vollständiges Image ist erstellt
- CSP LED 2 blinkt: Steuerung wurde heruntergefahren
- Steuerung über den Hauptschalter ausschalten
- KUKA.RecoveryUSB Stick entfernen
- Steuerung über den Hauptschalter einschalten

3- KR C4 Image im Automatischen Modus wiederherstellen

- KUKA.RecoveryUSB Stick an der Robotersteuerung anstecken.
- Robotersteuerung starten.
- CSP LED 1 blinkt: KUKA.RecoveryUSB Stick bootet die Steuerung.
- CSP LED 1 leuchtet: Bootvorgang ist abgeschlossen.
- CSP LED 2 blinkt: Images der C-Partition wird wieder hergestellt und auf die versteckte Partition kopiert.
- CSP LED 2 leuchtet: Wiederherstellung der C-Partition ist abgeschlossen
- CSP LED 2 blinkt: Images der D-Partition wird wieder hergestellt und auf die versteckte Partition kopiert.
- CSP LED 2 leuchtet: Wiederherstellung der D-Partition ist abgeschlossen.
- CSP LED 1-6 leuchtet für 1 Sekunde: Vollständiges Image ist wieder hergestellt.
- CSP LED 2 blinkt: Steuerung wurde heruntergefahren.
- Steuerung über den Hauptschalter ausschalten.
- KUKA.RecoveryUSB Stick entfernen.
- Steuerung über den Hauptschalter einschalten.

4.2.2. Datenkonfiguration

Alle Daten, die der OPC UA Python-Client mit dem KUKAVARPROXY-Roboterserver empfängt und sendet, werden mit dem SmartPad des Roboters konfiguriert, bevor mit dem Schreiben des Roboterprogramms begonnen wird.

Die Deklaration der Variablen erfolgt unter dem Ordner \$ CONFIG.DAT (der Ordnerpfad im SmartPad ist R1 / SYSTEM / \$ CONFIG.DAT /).

die beschriebenen Variablen sind eine Variable M vom Typ Integer mit einer Initialisierung des Werts 0, eine Variable X vom Typ Boolean mit einer Initialisierung des Werts TRUE und eine Variable MoveXTS vom Typ Boolean mit einer Initialisierung des Werts TRUE.

```
1 DEFDAT $CONFIG
2 DECL BOOL b_PRODUKTION=TRUE
3 DECL BOOL b_WARTEN_DATEN=FALSE
4 DECL BOOL X=TRUE
5 DECL INT M=0
6
7 DECL BOOL MoveXTS=TRUE
8 DECL BOOL M_1=FALSE
9 DECL BOOL M_2=FALSE
10 ENDDAT
```

Abbildung 49: Seite CONFIG.Data des Roboters

Die Applikation wird im Roboter im manuellen Modus T1 programmiert und nach Abschluss der Programmierung wird der Robotermodus in den Automatikmodus umgeschaltet, so dass sich der Roboter ohne menschliches Eingreifen automatisch bewegt (siehe Abschnitt [2.5.3](#)).

Die Rolle der Variablen, die in der Datei \$ CONFIG.DAT konfiguriert werden, wird im folgenden Abschnitt ausführlich beschrieben.

4.2.3. Roboter-Programm

Das Programm des KUKA Roboters ist wie folgt geschrieben:

Die Schleife, die in diesem Programm verwendet wird, ist eine WHILE-Schleife, deren Bedingung die boolesche Variable X ist.

Die Variable X ist standardmäßig auf TRUE gesetzt, damit das Programm automatisch ablaufen kann.

Unter der WHILE-Schleife wurde eine SWITCH-CASE-Schleife erstellt, deren Variable M (Integer-Variable) ist. 10 CASES hängen von dem Wert von M ab (M nimmt Integer-Werte von 1 bis 10 an).

Für jedes CASE führt der Roboter genau festgelegte Bewegungen aus, je nachdem, welcher Mover sich in der Fixstation der XTS-Maschine befindet und wie sein Zustand ist (leer oder voll).

Neben den beiden Maschinen befindet sich ein Kasten mit 5 Feldern (siehe Box im Anhang), aus dem der Roboter das Objekt auf das entsprechende Feld nehmen oder legen kann.

Wenn die Variable M einen ungeraden Zahlenwert annimmt (1,3,5,7 und 9), nimmt der Roboter das Objekt vom Mover in der Fixposition und legt es in ein entsprechendes Feld in dem Kasten.

Wenn die Variable M einen geraden Zahlenwert annimmt (2,4,6,8 und 10), nimmt der Roboter das Objekt aus dem entsprechenden Feld in dem Kasten und legt es in den Mover an der Fixposition.

Wenn sich ein Mover in der Fixposition des XTS befindet, empfängt der OPC UA Python Client vom OPC UA Server des XTS eine boolesche MoveKUKA-Variable mit dem Wert TRUE. Die MoveKUKA-Variable wird im Paython-Client verarbeitet (siehe Abschnitt [4.4.](#)) und sendet einen Wert von 1 bis 10 an die M-Variable im KUKAVARPROXY-Roboter-Server. Anschließend erhält das KRL-Programm den genauen Wert der Variablen M und der Roboter arbeitet entsprechend des entsprechenden Feldes.

Wenn der Roboter seine Operation beendet hat, erhält der OPC-UA-Python-Client eine boolesche Variable MoveXTS mit dem Wert TRUE und sendet sie an den OPC-UA-Server des XTS, woraufhin der Mover in der FixPosition zur Wartestation fährt und ein anderer Mover direkt dahinter zur festen Position fährt und denselben Vorgang durchläuft.

Das folgende Aktivitätsdiagramm erklärt, wie das Roboterprogramm genau funktioniert.

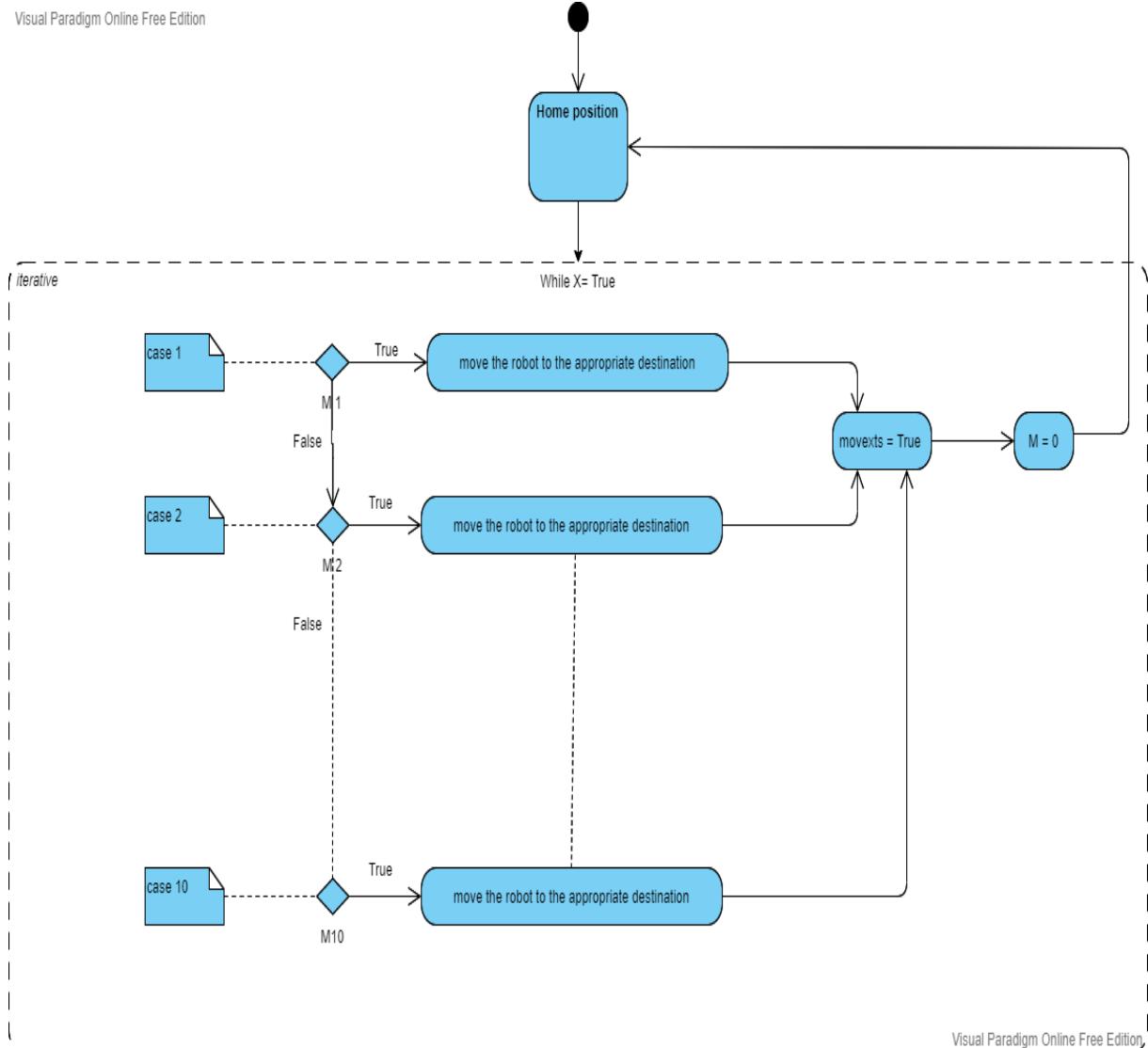


Abbildung 50: Aktivitätsdiagramm des Roboterprogramms

4.3. Programmierung des XTS

Die Programmierphase der XTS-Maschine (siehe Abschnitt [2.6.](#)) wird von der Beckhoff TwinCat 3 Software durchgeführt (siehe Abschnitt [2.6.1.](#)).

Diese Software ist das Programmiertool für die verschiedenen Beckhoff-Produkte. Es wird ständig in Bezug auf Lizenzen und Bibliotheken verbessert.

In diesem Rahmen wurde die Automatisierung der Beckhoff XTS-Maschine mit dieser Software programmiert, wobei verschiedene Lizenzen und Bibliotheken verwendet wurden, um die Applikation auf dem XTS zu programmieren und die Kommunikation mit dem OPC

UA Python-Client mit dem über den OPC UA-Server des XTS geschriebenen Programm herzustellen.

Zum besseren Verständnis dieses Abschnitts wurde er in drei Teile gegliedert. Der erste Teil befasst sich mit den in diesem Projekt verwendeten Lizenzen, der zweite Teil beschreibt die verschiedenen verwendeten Bibliotheken und der dritte Teil erläutert das geschriebene Programm.

4.3.1. Lizenzen

Die Lizenzen für TwinCat 3, die in dieser Applikation verwendet werden, sind wie folgt

4.3.1.1. TC 1200 Lizenz

TwinCAT 3 PLC (TC 1200) implementiert eine oder mehrere SPS auf einem Industrie-PC. Für die Programmierung der SPS wird die internationale Norm IEC 61131-3 3e verwendet und alle in dieser Norm beschriebenen Programmiersprachen werden unterstützt. Verschiedene komfortable Debugging-Möglichkeiten erleichtern die Fehlersuche und Inbetriebnahme. Programmänderungen können jederzeit und in beliebiger Größe online, d.h. bei laufender SPS, vorgenommen werden. Alle Variablen sind symbolisch über ADS verfügbar und können in den entsprechenden Anwendungen gelesen und geschrieben werden.

[\[47\]](#)

Die Merkmale des TC1200 werden in den folgenden Punkten beschrieben [\[47\]](#):

- Die Größe des Prozessabbilds, der Bereich der Indikatoren, die Programmgröße, die POU-Größe und die Anzahl der Variablen sind nur durch die Größe des RAMs begrenzt.
- Zykluszeit ab 50 µs.
- Verknüpfungszeit: 1 µs/1000 Codierungszeilen (Intel® Core™ i Prozessoren).
- objektorientierte Programmierung möglich.
- Online-Änderungen von Programmen und Variablen.
- Online-Verbindung mit SPS-Laufzeitsystem weltweit über TCP/IP oder Feldbus.
- Online-Überwachung von Variablen in Variablenlisten, Überwachungsfenstern, Editoren.
- Online-Status und Energiefluss (Akkumulator Inhalt) von Programmen und Instanzen.
- Auslösen, Erzwingen und Setzen von Variablen.
- leistungsfähiges Debugging mit Single Cycle, Break Points, Step In, Step Over, Anzeige des aktuellen Call Stacks, Watchlist zeigt Auswahl der Variablen, Trace Funktionen.
- Online-Verwaltung aller Variablennamen und -strukturen über das gesamte System hinweg.

- remanente und persistente Daten, UPS-unterstützte Speicherung auf Harddisk, Speicherung in NOVRAM als Option.
- Zugriff auf Variablen zum Lesen und Schreiben über ADS, OPC.
- zertifiziert nach PLCopen Base Level (IL/ST).
- strukturierte Programmierung mit modularer Programmverwaltung.
- optionale Speicherung des Quellcodes im Zielsystem.
- komfortable Bibliotheksverwaltung.
- leistungsfähiger Compiler mit inkrementeller Kompilierung für verschiedene Zielsysteme.
- alle gängigen Datentypen, Strukturen, Arrays, einschließlich mehrdimensionaler Arrays.
- einfache Verknüpfung mit Quellcode-Verwaltungstools durch Einbettung in Microsoft Visual Studio®.

4.3.1.2. *TF 5000 Lizenz*

TwinCAT 3 NC PTP 10 Axes (TF 5000) implementiert die Motion Control für Punkt-zu-Punkt-Bewegungen in der Software. Die Achsen werden durch Achsobjekte repräsentiert und stellen eine zyklische Schnittstelle, z.B. zur SPS, dar. Dieses Achsenobjekt wird dann mit einer entsprechenden physikalischen Achse verknüpft. Auf diese Weise können verschiedenste Achsentypen mit unterschiedlichsten Feldbus-Schnittstellen auf abstrakte Weise mit den Achsobjekten verbunden werden, die immer eine identische Konfigurationsoberfläche bieten. Die Steuerung der Achsen kann in verschiedenen Konfigurationen (Positions- oder Geschwindigkeitsinterface) und mit verschiedenen Reglern realisiert werden. Die Achsen werden im TwinCAT Engineering konfiguriert. [\[48\]](#)

Die Merkmale des TF 5000 werden in den folgenden Punkten beschrieben [\[48\]](#):

- bis zu 10 Achsen, erweiterbar auf maximal 255 Achsen.
- unterstützt elektrische und hydraulische Servoantriebe, Frequenzumrichter-Antriebe, Schrittmotorantriebe, DC-Antriebe, geschaltete Antriebe (schnelle/langsame Achsen), Simulationsachsen und Encoder-Achsen.
- unterstützt verschiedene Feldbus-Schnittstellen wie EtherCAT.
- Standard-Achsfunktionen wie Start/Stop/Reset/Referenz, Geschwindigkeits-Override, Master/Slave-Kopplungen, elektronisches Getriebe, Online-Abstandskompensation.
- Die Programmierung erfolgt über PLCopen-konforme IEC 61131-3 Funktionsbausteine.
- praktische Möglichkeiten der Achsinbetriebnahme.

- Online-Überwachung aller Achszustandsvariablen wie zum Beispiel actual/setpoint werte, Freigaben, Kontrollwerte.
- Online-Achsabstimmung.
- Konfiguration aller Achsparameter, wie Mess-System, Antriebsparameter und Positionsregelung.
- Konfigurierbare Regler Strukturen: P-Regelung, PID-Regelung, PID mit Geschwindigkeitsvorregelung, PID mit Geschwindigkeits- und Beschleunigungsvorregelung.
- Online-Master/Slave- und Slave/Master-Umwandlung.
- Multi-Master-Kopplung.

4.3.1.3. *TF 5850 Lizenz*

Das linear transport system XTS ermöglicht die individuelle Bewegung von XTS-Movern entlang eines bestimmten Streckenverlaufs. Die XTS-Extension TF5850 ist das Basis-Softwarepaket für den Einsatz des XTS und seine Integration in die TwinCAT-3-Umgebung. Der Benutzer kann weiterhin von den extensiven Möglichkeiten von TwinCAT und dem XTS profitieren. Das Setup enthält alle Treiber und Tools für ein effizientes und intuitives Arbeiten mit dem XTS. [\[49\]](#)

Die Eigenschaften der TF 5000 werden in den folgenden Punkten beschrieben [\[49\]](#):

- Integrierte XTS-Simulationsfunktionen erleichtern die Anlagenplanung.
- Die XTS-Softwaretools unterstützen eine einfache Konfiguration und schnelle Inbetriebnahme.
- XTS Mover werden wie Servo-Achsen gehandhabt und bieten somit den konventionellen Funktionsumfang.
- Motion-Funktionsbausteine reduzieren den Engineering-Aufwand (Lizenz für TF5400 TwinCAT 3 Advanced Motion Pack ist integriert).
- XTS Track Management ermöglicht maximale Flexibilität des Teiletransports.
- Unterstützung zahlreicher Feldbusse zur Integration in bestehende Systemlösungen.

4.3.1.4. *TF 6421 Lizenz*

Der TwinCAT XML-Server (TF 6421) bietet eine SPS-Bibliothek, mit der ein Schreib- und Lesezugriff auf XML-Dateien realisiert werden kann. Der XML-Server zeichnet sich durch seine einfache Handhabung aus. Besonders eignet er sich beispielsweise für das Laden von Initialisierungsdaten, so wie es häufig beim Aufstarten einer Maschine benötigt wird. [\[50\]](#)

4.3.2. Bibliotheken

In dem Projekt wurden verschiedene Bibliotheken verwendet, um die Beckhoff XTS Maschine in der gewünschten Weise zu programmieren. Diese Bibliotheken wurden in der TwinCat 3 Software unter dem Ordner References aufgerufen.

Im Folgenden wird jede Bibliothek mit ihrer Definition vorgestellt, damit im nächsten Teil die Programmierung des XTS verständlicher wird.

4.3.2.1. Bibliothek Tc2_MC2

Dies ist eine konvertierte TwinCAT-2-TcMC2-Bibliothek. Bestehende Projekte, die noch die TwinCAT-2-TcMC-Bibliothek verwenden, müssen zunächst an die TcMC2-Bibliothek angepasst werden, bevor sie für TwinCAT 3 konvertiert werden können. [51]

Das folgende Zustandsdiagramm definiert das Verhalten einer Achse für den Fall, dass mehrere Funktionsbausteine für diese Achse gleichzeitig aktiv sind. Die Kombination mehrerer Funktionsbausteine ist nützlich, um komplexere Bewegungsprofile zu erzeugen oder um in einem Programmablauf Ausnahmezustände zu behandeln. [52]

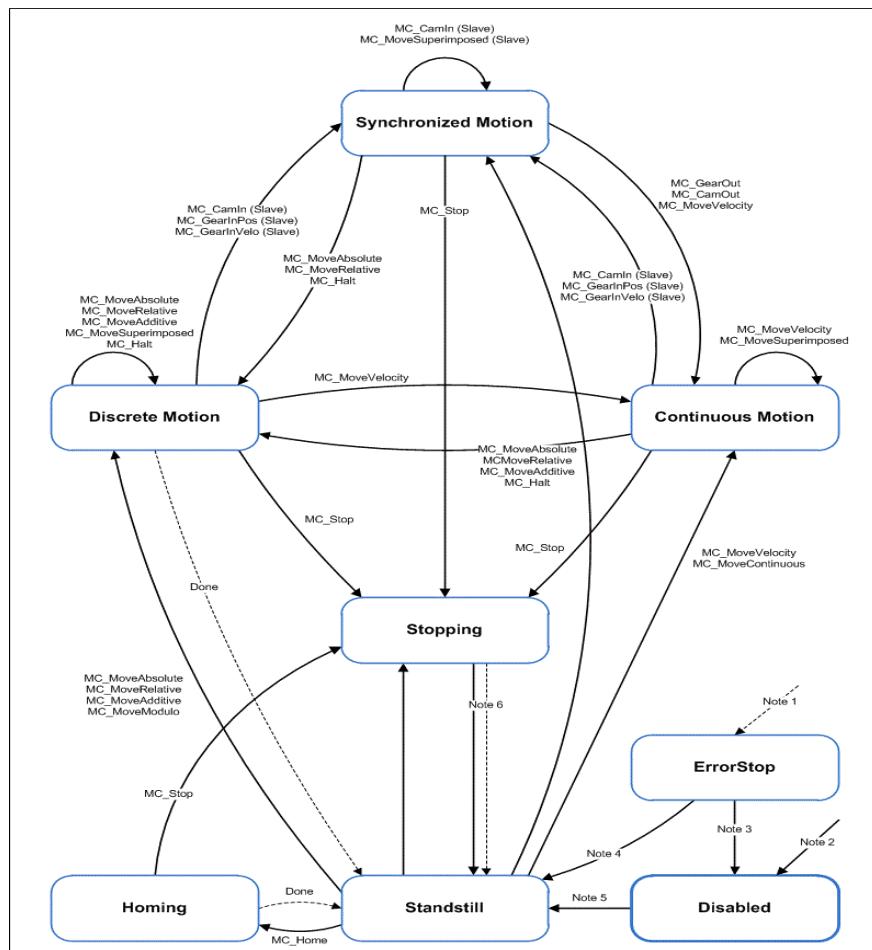


Abbildung 51: Zustandsdiagramm [52]

- Note 1: Aus irgendeinem Zustand, in dem ein Fehler auftritt
- Note 2: Aus irgendeinem Zustand, wenn MC_Power.Enable = FALSE und die Achse hat keinen Fehler.
- Note 3: MC_Reset und MC_Power.Status = FALSE
- Note 4: MC_Reset und MC_Power.Status = TRUE und MC_Power.Enable = TRUE
- Note 5: MC_Power.Status = TRUE und MC_Power.Enable = TRUE
- Note 6: MC_Stop.Done = TRUE und MC_Stop.Execute = FALSE

Bewegungskommandos werden grundsätzlich sequenziell abgearbeitet. Alle Kommandos arbeiten in dem beschriebenen Zustandsdiagramm. [\[52\]](#)

Die Achse befindet sich immer in einem der definierten Zustände. Jedes Bewegungskommando, das eine Transition verursacht, ändert den Zustand der Achse und damit das Bewegungsprofil. Das Zustandsdiagramm ist eine Abstraktionsebene, die den realen Zustand der Achse, vergleichbar zum Prozessabbild von E/A-Punkten, wiederspiegelt. Der Zustand der Achse wechselt sofort mit dem auslösenden Kommando. [\[52\]](#)

Das Zustandsdiagramm bezieht sich auf Einzelachsen. Multi-Achsenbausteine, wie MC_CamIn oder MC_GearIn, beeinflussen die Zustände mehrerer Achsen, die jedoch immer auf einen Einzelachszustand der beteiligten Achsen zurückgeführt werden können. Beispielsweise kann ein Kurvenscheiben-Master im Zustand „Continous Motion“ und der zugehörige Slave im Zustand „Synchronized Motion“ sein. Das Ankoppeln eines Slaves hat keinen Einfluss auf den Zustand des Masters. [\[52\]](#)

Der Zustand „Disabled“ beschreibt den Grundzustand einer Achse. In diesem Zustand kann die Achse durch keinen Funktionsbaustein bewegt werden. Wenn der Funktionsbaustein MC_Power mit Enable = TRUE aufgerufen wird, wechselt die Achse in den Zustand „Standstill“ oder im Fehlerfall in den Zustand „ErrorStop“. Wenn der Funktionsbaustein MC_Power mit Enable = FALSE aufgerufen wird, wechselt der Zustand nach „Disabled“. [\[52\]](#)

Zweck des Zustands „ErrorStop“ ist es, die Achse zu stoppen und anschließend keine weiteren Kommandos anzunehmen, bis ein Reset ausgelöst wurde. Der Zustandsübergang „Error“ bezieht sich nur auf tatsächliche Achsfehler und nicht auf Ausführungsfehler eines Funktionsbausteins. Achsfehler können aber auch am Fehlerausgang eines Funktionsbausteins angezeigt werden. [\[52\]](#)

Funktionsbausteine, die im Zustandsdiagramm nicht aufgeführt werden, beeinflussen den Zustand der Achse nicht (MC_ReadStatus, MC_ReadAxisError, MC_ReadParameter, MC_ReadBoolParameter, MC_WriteParameter, MC_WriteBoolParameter, MC_ReadActualPosition und MC_CamTableSelect). [\[52\]](#)

Der Zustand „Stopping“ zeigt an, dass sich die Achse in einer Stopprampe befindet. Der Zustand wechselt nach dem vollständigen Stopp nach „Standstill“. [\[52\]](#)

Bewegungskommandos wie MC_MoveAbsolute, die aus dem Zustand „Synchronized Motion“ herausführen, sind nur dann möglich, wenn sie in den Achsparametern explizit erlaubt werden. Abkoppelkommandos wie MC_GearOut sind unabhängig davon möglich. [\[52\]](#)

In diesem Projekt wurden Bausteinfunktionen und Datentypen verwendet.

➤ MC_Power:

MC_Power aktiviert die Software-Freigabe für eine Achse. Die Freigabe kann für beide Fahrtrichtungen oder nur für eine Richtung aktiviert werden. Am Ausgang "Status" wird die Betriebsbereitschaft der Achse angezeigt. [\[53\]](#)

Ein Geschwindigkeits-Override beeinflusst die Geschwindigkeit aller Fahrbefehle um einen bestimmten Prozentsatz. [\[53\]](#)

Je nach Antriebstyp signalisiert "Status" auch die Betriebsbereitschaft des Antriebs. Digitale Antriebe geben eine Rückmeldung über die Betriebsbereitschaft, während analoge Antriebe ihre Betriebsbereitschaft nicht anzeigen können. Im letzteren Fall zeigt Status nur die Betriebsbereitschaft der Steuerungsseite an. [\[53\]](#)

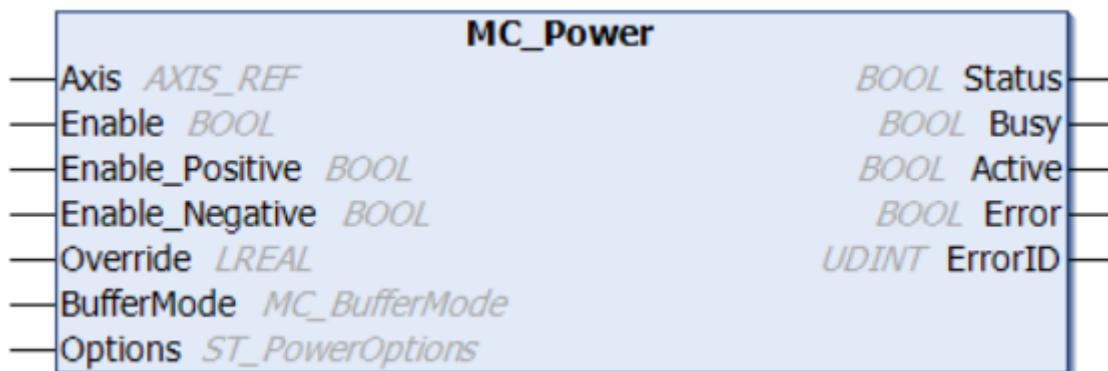


Abbildung 52: Bausteinfunktion MC_Power [\[53\]](#)

➤ MC_Reset:

MC_Reset setzt die NC-Achse zurück. Dies führt in vielen Fällen auch zu einem Reset eines angeschlossenen Antriebsgerätes. Je nach Bussystem oder Antriebstypen kann in manchen Fällen ein separater Reset für das Antriebsgerät erforderlich sein. [\[54\]](#)



Abbildung 53: Bausteininfunktion MC_Reset [\[54\]](#)

➤ AXIS_REF:

Der Datentyp AXIS_REF enthält Achsinformationen. AXIS_REF ist eine Schnittstelle zwischen der SPS und der NC. Er wird als Achsreferenz zu MC-Funktionsbausteinen hinzugefügt. [\[55\]](#)

(Beckhoff Automation GmbH)AXIS_REF-Datenstruktur Der FBAXIS_REF ist eigentlich kein Funktionsblock, sondern eine Datenstruktur, die die Achsen-E/A-Variablen sowie zusätzliche Informationen enthält. Der Grund für die Verwendung einer STRUCT ist, dass Strukturen keine lokalisierten E/A-Variablen enthalten können. Der Anwender soll den Datentyp AXIS_REF verwenden, der intern den Typ auf diese Funktionsbausteineindeinition (Alias) umleitet. [\[55\]](#)

```

TYPE AXIS_REF :
  VAR_INPUT
    PlcToNc AT %Q* : PLCTONC_AXIS_REF;
  END_VAR
  VAR_OUTPUT
    NcToPlc AT %I* : NCTOPLC_AXIS_REF;
    ADS          : ST_AdsAddress;
    Status       : ST_AxisStatus;
    DriveAddress : ST_DriveAddress;
  END_VAR
END_TYPE

```

Abbildung 54: AXIS_REF Datentyp [\[55\]](#)

4.3.2.2. Bibliothek Tc2_Standard

Die Standard-Bibliothek beinhaltet alle IEC61131-3 POUs. Die POUs können eingeteilt werden in [56]:

- Bistabile Funktionsblöcke (RS und SR)
- Zähler (CTD, CTU und CTUD)
- Timer (TOF, TON und TP)
- Timer (LTIME) (LTOF, LTON und LTP)
- Trigger (F_TRIG und R_TRIG)
- Stringfunktionen (CONCAT, DELETE, FIND, INSERT, LEFT, LEN, MID, REPLACE und RIGHT)
- Stringfunktionen (WSTRING) (WCONCAT, WDELETE, WFIND, WINSERT, WLEFT, WLLEN, WMID, WREPLACE und WRIGHT)

Der TON-Timer wurde in diesem Projekt verwendet.

➤ TON:

Timer Einschaltverzögerung. Q ist True, PT Millisekunden nachdem IN eine steigende Flanke hatte.

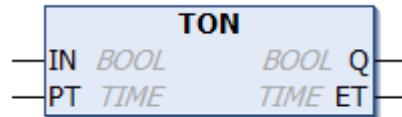


Abbildung 55: Timer TON

Name	Typ	Kommentar
IN	BOOL	startet Timer mit steigender Flanke, setzt Timer mit fallender Flanke zurück.
PT	TIME	Zeit, die vergeht, bevor Q gesetzt wird.
Q	BOOL	ist True, PT Sekunden nachdem IN eine steigende Flanke hatte.
ET	TIME	verstrichene Zeit.

Tabelle 5: Timer TON

4.3.2.3. Bibliothek Tc2_System

Nicht alle Funktionsbausteine und Funktionen, die häufig in SPS Applikationen benötigt werden, sind in der IEC61131-3 genormt. Die Bibliothek Tc2_System enthält solche Funktionen und Funktionsbausteine des TwinCAT-Systems, die nicht zum Standardumfang der IEC61131-3 gehören und dementsprechend herstellerspezifisch sind. [\[57\]](#)

T_MaxString wurde in diesem Projekt verwendet.

- T_MaxString

Die Variable dieses Typs ist eine PLC-String mit der maximalen Länge (TwinCat PLC-String mit einer maximalen Länge von 255 Bytes + 1 Byte Null-Begrenzer). [\[58\]](#)

4.3.2.4. Bibliothek Tc2_XmlDataSrv

Die SPS-Bibliothek Tc2_XmlDataSrv wird mit dem TC3 XML Server mitgeliefert und während der Installation in den Ordner ...C:\TwinCAT\3.1\Components\Plc\Managed Libraries\Beckhoff Automation GmbH kopiert. [\[59\]](#)

Es gibt jeweils zwei Funktionsbausteine zum Auslesen von Variablen aus der XML-Datei [\[59\]](#):

- FB_XmlSrvRead
- FB_XmlSrvReadByName

und zwei Funktionsbausteine zum Schreiben von SPS-Variablen in die XML-Datei [\[59\]](#):

- FB_XmlSrvWrite
- FB_XmlSrvWriteByName

Die erste Variante (FB_XMLSrvRead, FB_XMLSrvWrite) verwendet die Adresse und die Größe der Variable in der SPS, um die Variable zu spezifizieren. Die zweite Variante (FB_XMLSrvReadByName, FB_XMLSrvWriteByName) verwendet den Symbolnamen, um die Variable zu spezifizieren. Die erste Variante ist performanter. Zusätzlich zu Adresse und Größe bzw. Symbolname muss der Pfad der XML-Datei und der Standort der Variablen im XML-Dokument den Funktionsbausteinen als Eingangsparameter übergeben werden. [\[59\]](#)

Die unterstützten primitiven Datentypen sind in der folgenden Tabelle dargestellt:

Datentyp	Beispiel in der SPS	Beispiel in XML
UDINT (32), DINT (32), UINT (16), INT (16), USINT (8), SINT (8), DWORD (32), WORD (16), BYTE (8)	value1: DINT: = -1; value2: UDINT := 65535;	<dataentry> <MAIN.value1>-1</ MAIN.value1> <MAIN.value2>65535</ MAIN.value2> </dataentry>
LREAL (64), REAL (32)	value1: LREAL = 1.2;	<dataentry> <MAIN.value1>1.2</ MAIN.value1> </dataentry>
STRING (8)	str1: STRING = 'hallo';	<dataentry> <MAIN.str1>hallo</ MAIN.str1> </dataentry>
TIME (32), DATE (32), TOD, DT	date1: DATE: =D#2005- 05-04; (* Time-Typen werden in der XMLDatei als DWORD gespeichert*)	<dataentry> <MAIN.date1>1115164800</ MAIN.date1> </dataentry>
BOOL (8)	bool1: BOOL: = TRUE; bool2: BOOL := FALSE;	<dataentry> <MAIN.bool1>true</ MAIN.bool1> <MAIN.bool2>false</MAIN.bool2> </dataentry>

Tabelle 6: Primitive Datentypen [\[59\]](#)

die folgenden Bausteinfunktionen wurden in diesem Projekt verwendet.

➤ FB_XmlSrvRead:

Der Funktionsbaustein FB_XmlSrvRead kann verwendet werden, um eine SPS-Variable mit Daten aus einer XML-Datei zu initialisieren. Die Eingangsvariable sXPath muss auf einen gültigen Knoten in der über sFilePath angegebenen XML-Datei zeigen. Das zu initialisierende Symbol wird durch die Symboladresse und die Größe eindeutig identifiziert. [\[60\]](#)

➤ FB_XmlSrvWrite:

Mit dem Funktionsbaustein FB_XmlSrvWrite kann der Wert einer SPS-Variablen in eine XML-Datei geschrieben werden. Die Eingangsvariable sXPath muss dabei auf einen gültigen Knoten in der mit sFilePath angegebenen XML-Datei zeigen. Das Symbol, das geschrieben werden soll, wird anhand der übergebenen Symboladresse und Symbolgröße eindeutig identifiziert. [\[61\]](#)

4.3.2.5. *Bibliothek Tc3_EventLogger*

Die SPS-Bibliothek Tc3_EventLogger umfasst Funktionen und Funktionsbausteine zur Verwendung des TwinCAT 3 EventLogger. [\[62\]](#)

Die Funktionen und Blockfunktionen, die in diesem Projekt verwendet werden, sind die folgenden.

- FB_TcEventLogger:

Dieser Funktionsbaustein stellt den TwinCAT 3 EventLogger selbst dar. [\[63\]](#)

- FB_TcMessage:

Dieser Funktionsbaustein stellt eine Meldung des TwinCAT 3 EventLoggers dar. [\[64\]](#)

- I_TcClearLoggedEventsSettings:

Bietet die Möglichkeit, anzugeben, welche Events aus dem Speicher entfernt werden sollen. [\[65\]](#)

- I_TcMessage:

Dieses Interface stellt Methoden und Eigenschaften für das Message-Handling bereit. [\[66\]](#)

4.3.2.6. *Bibliotheken Tc3_McCollisionAvoidance und Tc3_McCoordinatedMotion*

Mit TF5410 TC3 Collision Avoidance wird PTP um Collision Avoidance ergänzt. Mover können auf einer eindimensionalen Bahn bewegt werden, wobei sichergestellt wird, dass sie nicht miteinander kollidieren und einen benutzerdefinierten Mindestabstand zu anderen Movern einhalten. Dafür werden die PTP-Achsen zu einer Gruppe verbunden. Die Motion Funktionsbausteine für die Collision Avoidance sind in der Bibliothek Tc3_McCollisionAvoidance enthalten. Die administrativen Funktionsbausteine sind in der Bibliothek Tc3_McCoordinatedMotion enthalten. [\[67\]](#)

Die Bausteinfunktionen dieser beiden Bibliotheken, die in diesem Projekt verwendet werden, sind:

- MC_UngroupAllAxes:

Dieser Funktionsbaustein entfernt alle Achsen und deaktiviert die Gruppe. Wenn der Funktionsbaustein erfolgreich ist, befindet sich die Gruppe anschließend im Zustand GroupDisabled. [\[68\]](#)

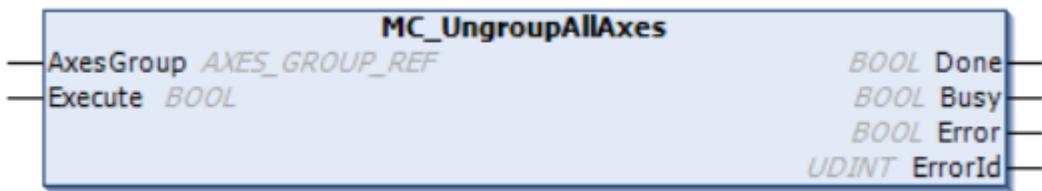


Abbildung 56: Funktionsbaustein MC_UngroupAllAxes [\[68\]](#)

- MC_GroupHalt:

Der Funktionsbaustein MC_GroupHalt stoppt eine Gruppe mit einer definierten Verzögerungsrampe. Im Gegensatz zu "MC_GroupStop" wird die Gruppe nicht für weitere Fahrbefehle gesperrt. Daher kann die Gruppe während der Verzögerungsrampe oder nach dem Stoppen durch einen anderen Befehl wieder gestartet werden. [\[69\]](#)



Abbildung 57: Funktionsbaustein MC_GroupHalt [\[69\]](#)

- MC_GroupStop:

Der Funktionsbaustein stoppt die Gruppe und alle zugehörigen Achsen mit einer definierten Verzögerungsrampe und sperrt die Achse für Fahrbefehle. Während sich die Gruppe im Zustand GroupStopping befindet, kann kein anderer Funktionsbaustein eine Achse der Gruppe bewegen. [\[70\]](#)

Die Gruppe kann erst wieder bewegt werden, wenn das Signal Execute auf FALSE gesetzt wurde, nachdem die Geschwindigkeit 0. [\[70\]](#)



Abbildung 58: Funktionsbaustein MC_GroupStop [\[70\]](#)

➤ MC_MoveAbsoluteCA:

Dieser Funktionsbaustein weist eine einzelne Achse an, die im Funktionsbaustein definierte absolute Position unter Berücksichtigung der Kollisionsvermeidung anzufahren. Die Kollisionsvermeidung hat höhere Priorität als der Fahrbefehl. Daher kann die Achse langsamer werden oder warten, während der Fahrbefehl ausgeführt wird, um eine Kollision zu vermeiden. Der Funktionsbaustein gibt das Signal Done erst aus, wenn die Achse ihre Zielposition erreicht hat. [\[71\]](#)

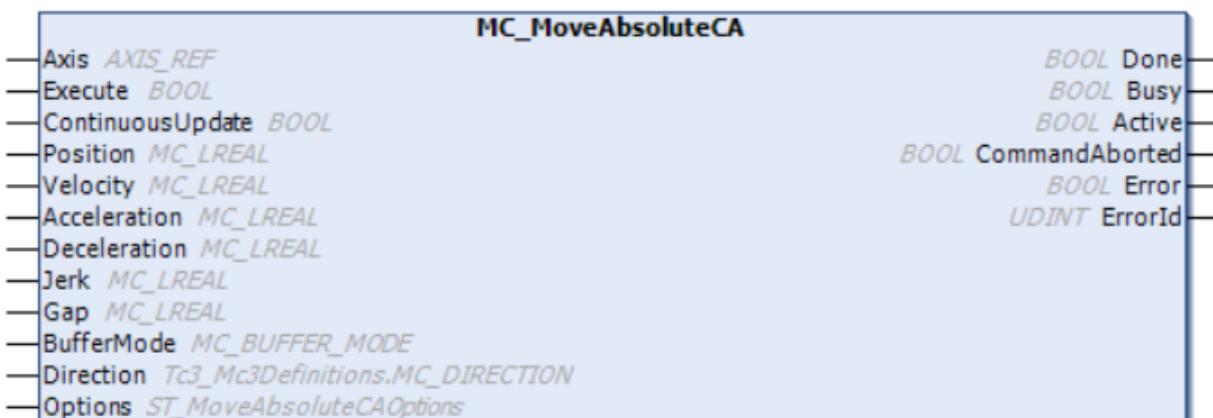


Abbildung 59: Funktionsbaustein MC_MoveAbsoluteCA [\[71\]](#)

➤ MC_GroupEnable

Dieser Funktionsbaustein gibt die Gruppe frei. Wenn er erfolgreich ist und alle Achsen bereit sind, befindet sich die Gruppe anschließend im Zustand GroupStandby. [\[72\]](#)



Abbildung 60: Funktionsbaustein MC_GroupEnable [\[72\]](#)

➤ MC_GroupDisable:

Dieser Funktionsbaustein deaktiviert die Gruppe. Nach erfolgreicher Ausführung ändert die Gruppe ihren Zustand in GroupDisabled. [\[73\]](#)



Abbildung 61: Funktionsbaustein MC_GroupEnable [\[73\]](#)

➤ MC_GroupReset:

Dieser Funktionsbaustein ermöglicht das Zurücksetzen aller anstehenden Fehler auf der Roboterseite. [\[74\]](#)

Weitere Aktionen können durch den Funktionsbaustein MC_GroupReset ausgeführt werden. [\[74\]](#)

- Telegramme, die zwischen SPS und Robotersteuerung übertragen werden, können in einer Sackgasse enden (z.B., wenn ein Gerät während des Datenaustauschs neu gestartet wird). [\[74\]](#)
- In dieser Situation setzt der Funktionsbaustein das Kommunikationsprotokoll zurück. Danach können wieder neue Telegramme ausgetauscht werden. [\[74\]](#)



Abbildung 62: Funktionsbaustein MC_GroupReset [\[74\]](#)

➤ MC_GroupReadStatus:

Dieser Funktionsbaustein liest den Zustand einer Achsgruppe. [\[75\]](#)

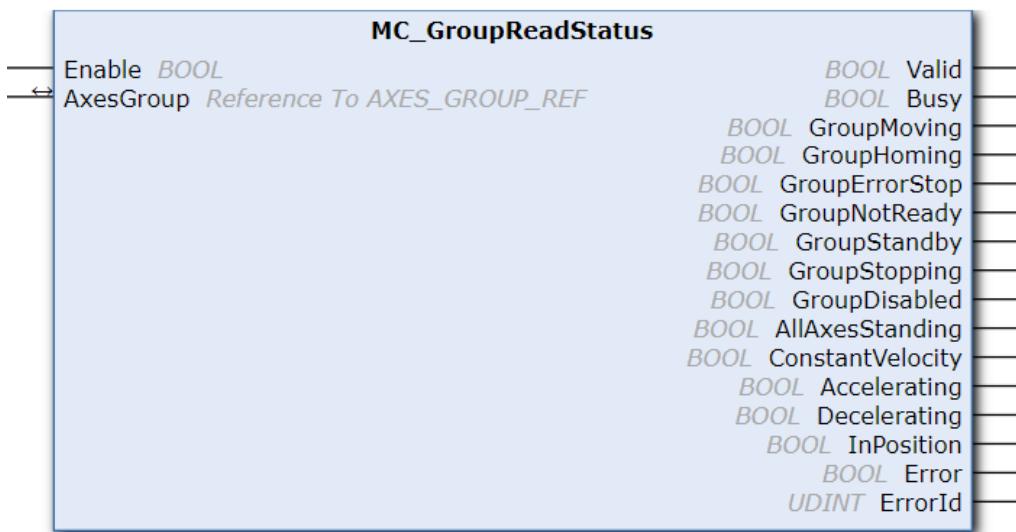


Abbildung 63: Funktionsbaustein MC_GroupReadStatus [\[75\]](#)

➤ MC_GroupReadError:

Dieser Funktionsbaustein gibt den Fehlercode für die Gruppe zurück. Er gibt keine Fehler für Funktionsbausteine zurück (z.B. ungültige Parametrierung). [\[76\]](#)



Abbildung 64: Funktionsbaustein MC_GroupReadError [\[76\]](#)

➤ MC_AddAxisToGroup:

Dieser Funktionsbaustein fügt eine Achse zu einer Gruppe hinzu. [\[77\]](#)



Abbildung 65: Funktionsbaustein MC_AddAxisToGroup [\[77\]](#)

➤ AXES_GROUP_REF:

Die zyklische Gruppenschnittstelle ermöglicht den zyklischen Datenaustausch zwischen SPS und einem NC-Gruppenobjekt. Die Gruppenschnittstelle enthält die Richtungen NcToPlc und

PlcToNc. Beide Richtungen sind in gemeinsame und gruppenspezifische Daten unterteilt.

[78]

```
TYPE AXES_GROUP_REF :  
STRUCT  
    PlcToNc AT %Q*      : PLCTOMC_GROUP_REF;  
    NcToPlc AT %I*      : MCTOPLC_GROUP_REF;  
END_STRUCT  
END_TYPE
```

Abbildung 66: Struktur AXES_GROUP_REF [78]

4.3.2.7. Bibliothek Tc2_Math

Die SPS-Bibliothek Tc2_Math enthält erweiterte mathematische Funktionen für die TwinCAT SPS. [79]

Die Funktionen der Bibliothek Tc2_Math sind in der folgenden Tabelle aufgeführt [64]:

Funktion	Beschreibung
FLOOR	Die FLOOR Funktion ermittelt einen ganzzahligen Wert aus einer Floating-Point-Zahl, der gerade kleiner oder gleich dieser Zahl ist.
FRAC	Die FRAC Funktion ermittelt den Nachkommaanteil einer Floating-Point-Zahl.
LMOD	Die LMOD Funktion führt eine Modulo-Division durch und gibt den vorzeichenbehafteten Divisionsrest zurück.
LTRUNC	Die LTRUNC Funktion ermittelt den ganzzahligen Anteil einer Floating-Point-Zahl.
MODABS	Die MODABS Funktion führt eine Modulo-Division durch und ermittelt den vorzeichenlosen Modulowert innerhalb des Modulobereichs.
MODTURNS	Die MODTURNS Funktion führt eine Modulo-Division durch und ermittelt den vorzeichenbehafteten ganzzahligen Anteil.
F_GetVersionTcMath	Liefert die Versionsinformationen der Bibliothek

Tabelle 7: Tc2_Math Funktionen [79]

4.3.2.8. *Bibliothek Tc3_XTS_Utility*

Die Bibliothek Tc3_XTS_Utility enthält XtsUnit und XTSEnvironment.

XTSUnit ist in der TF5850 XTS Technologie-Erweiterung enthalten und zeichnet sich durch seine Diagnose- und Visualisierungsfunktionalitäten in der SPS, automatische zyklische Updates aller Diagnosedaten, Zugriff auf die kompletten Systeminformationen über die ST_XtsUnit Struktur und durch Zugriffsmethoden auf SoftDrive, NC und TcIoXtsDriver Objekte und Parameter aus. [\[80\]](#)

XtsEnvironment ist Bestandteil der XTS-Technologie-Erweiterung TF5850 und zeichnet sich durch seine Diagnosefunktionalität in der SPS, ereignisgesteuerte Aktualisierung von Diagnosedaten, strukturierte Aufrufkette und durch Zugriffsmethoden auf XtsProcessingUnit-Objekte und -Parameter sowie auf die CoE-Daten der Module aus. [\[80\]](#)

4.3.3. XTS-Programm

Die Beckhoff-XTS-Maschine wurde mit der Beckhoff-Software TwinCat 3 programmiert. Die bereits in den beiden vorangegangenen Abschnitten erwähnten Lizenzen und Bibliotheken wurden verwendet, um das XTS in der für das Projekt gewünschten Weise zu programmieren.

4.3.3.1. *Powering der XTS-Maschine*

Dieser Teil des Programms besteht darin, die XTS-Maschine mit Strom zu versorgen. Dies geschieht mit Hilfe des Funktionsbausteins Mc_Power aus der Bibliothek Tc2_MC2. Bevor wir das Programm Power_All_Axes erstellen, muss in Global Variables (GVL) ein Array von 1 bis AxesCount vom Typ AXIS_REF definiert werden.

AXIS_REF ist eine AXIS-REF-Datenstruktur. Der FBAXIS_REF ist eigentlich kein Funktionsbaustein, sondern eine Datenstruktur. Sie enthält die Achsen-E/A-Variablen sowie zusätzliche Informationen. Der Anwender soll den Datentyp AXIS_REF verwenden, der intern den Typ auf die Funktionsbausteineinfinition (Alias) umleitet.

AxesCount ist eine konstante Variable, die in GVL als VAR_GLOBAL CONSTANT deklariert ist. Sie hat den Wert 5 und gibt die Anzahl der Achsen (d.h. die Anzahl der Mover) an.

4.3.3.2. Status und Fehler der Gruppe ablesen

Dieser Teil des Programms besteht darin, den Status und die Fehler der Achsengruppe zu lesen. Dies geschieht mit den beiden Funktionsbausteinen MC_GroupReadStatus und MC_GroupReadError aus der Bibliothek Tc3_McCoordinatedMotion.

Um diese beiden Funktionsbausteine zu verwenden, muss in der GVL die Achsengruppe (GroupAxes) vom Typ AXES_GROUP_REF deklariert werden.

4.3.3.3. Aktivitätsdiagramm: FB_Power_All

Das folgende Aktivitätsdiagramm konzentriert sich auf das Powering aller Achsen der XTS-Maschine sowie auf das Ablesen des Status und des Fehlers der Achsengruppe.

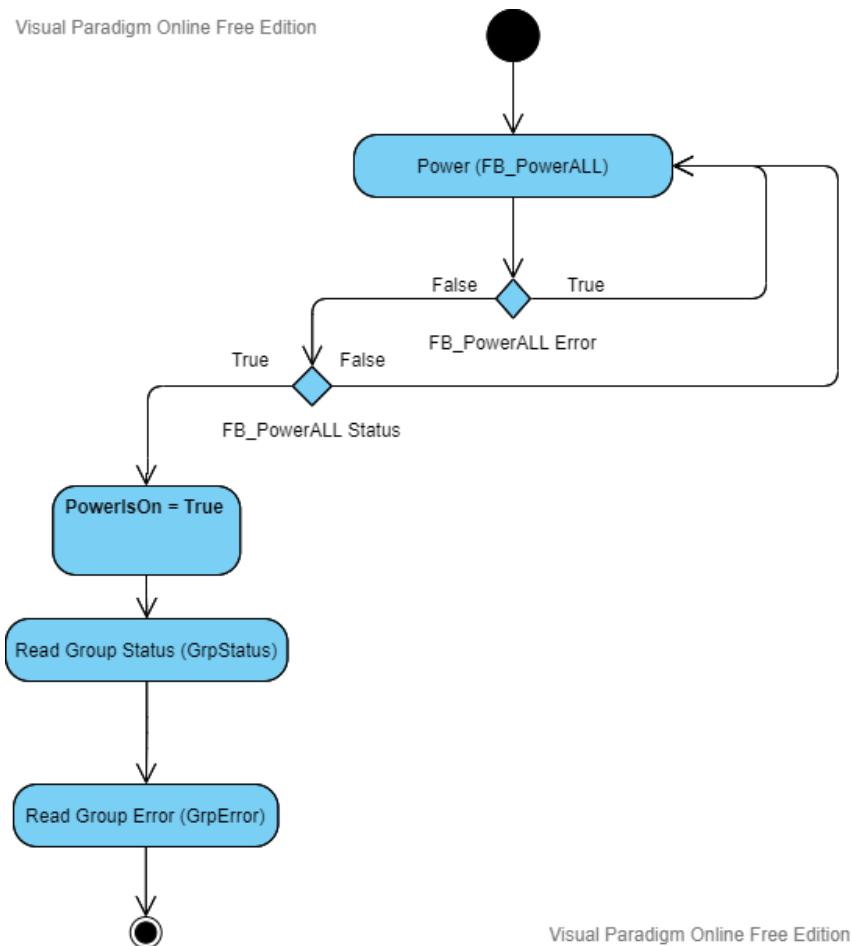


Abbildung 67: Aktivitätsdiagramm FB_PowerAll

4.3.3.4. Achsen hinzufügen

Dieser Teil des Programms besteht aus dem Hinzufügen der Achsen. Dies geschieht mit dem Funktionsbaustein MC_AddAxisToGroup der Bibliothek Tc3_McCoordinatedMotion.

4.3.3.5. Fehler beim Hinzufügen der Achsen

Wenn ein Fehler oder ein Busy-Zustand in AddAllAxes auftritt, wird ein MC_GroupStop-Funktionsbaustein aus der Tc3_McCoordinaedMotion-Bibliothek verwendet, um den AddAllAxes-Funktionsbaustein zu stoppen.

4.3.3.6. Hinzufügen der Achsen

Wenn kein Fehler- oder Busy-Signal in AddAllAxes vorliegt, wird ein MC_GroupEnable-Funktionsbaustein aus der Bibliothek Tc3_McCoordinaedMotion zur Aktivierung des AddAllAxes-Funktionsbausteins verwendet.

4.3.3.7. Aktivitätsdiagramm: FB_Add_All_Axes

Das folgende Aktivitätsdiagramm konzentriert sich auf das Hinzufügen von Achsen zur XTS-Maschine sowie auf die Fehler- und Tatsachenfälle beim Hinzufügen von Achsen.

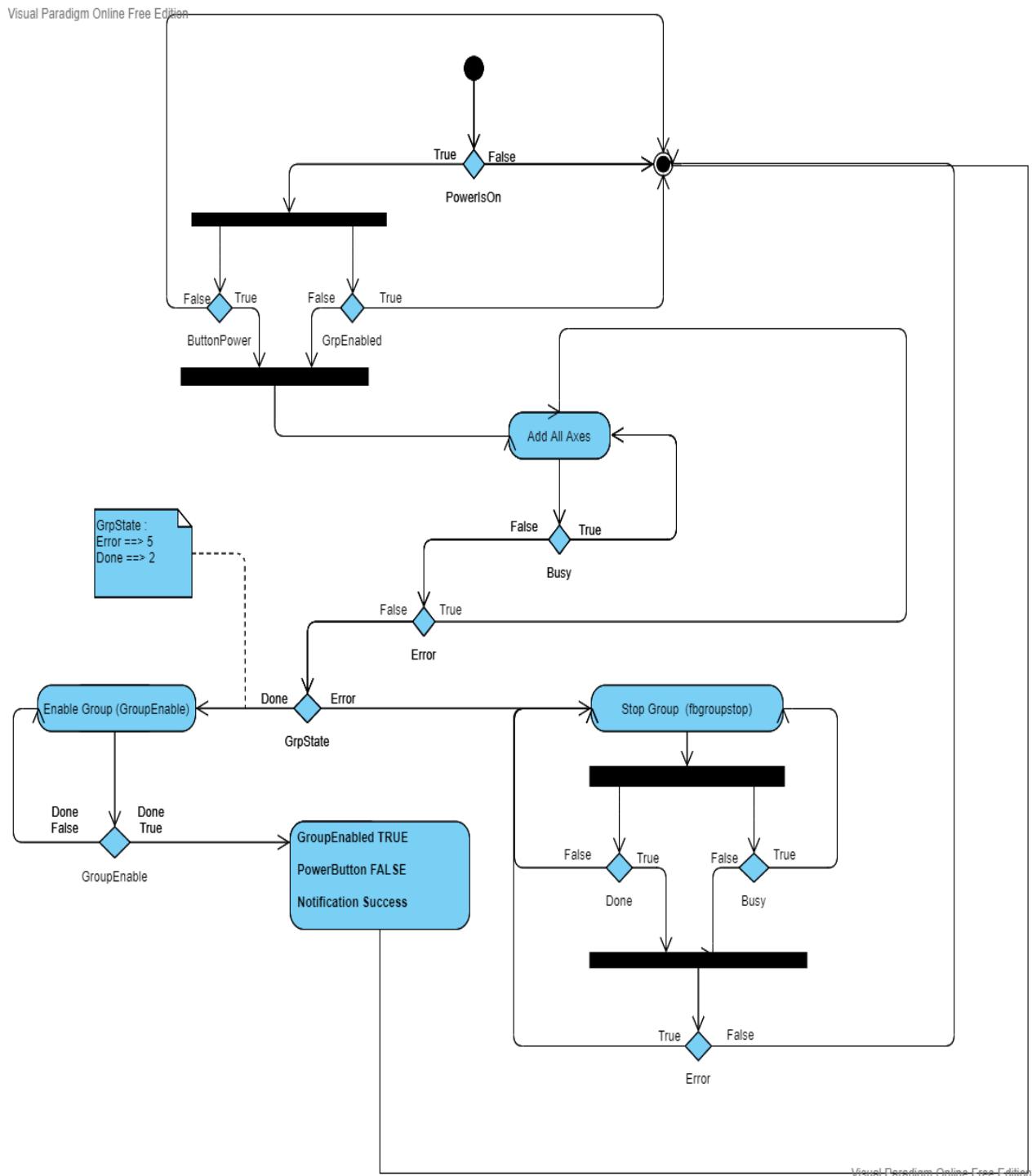


Abbildung 68: Aktivitätsdiagramm FB_Add_All_Axes

4.3.3.8. Stoppen der Achsengruppe

Dieser Teil des Programms besteht aus der Programmierung des Gruppenstopps der Mover. Dies geschieht mit dem Funktionsbaustein MC_GroupHalt aus der Bibliothek Tc3_McCoordinatedMotion.

4.3.3.9. Fälle von Fehler oder Besetzung

Bei einem Fehler oder einem besetzten Funktionsbaustein MC_GroupHalt wird ein Funktionsbaustein MC_UngroupAllAxes aus der Bibliothek Tc3_McCoordinatedMotion ausgeführt. Dieser Funktionsblock hebt die Gruppierung aller Achsen auf.

4.3.3.10. Aktivitätsdiagramm: FB_Group_Halt

Das folgende Aktivitätsdiagramm konzentriert sich auf den Gruppenstopp der XTS-Maschinenachsen und deren Fehler- und Besetzfälle.

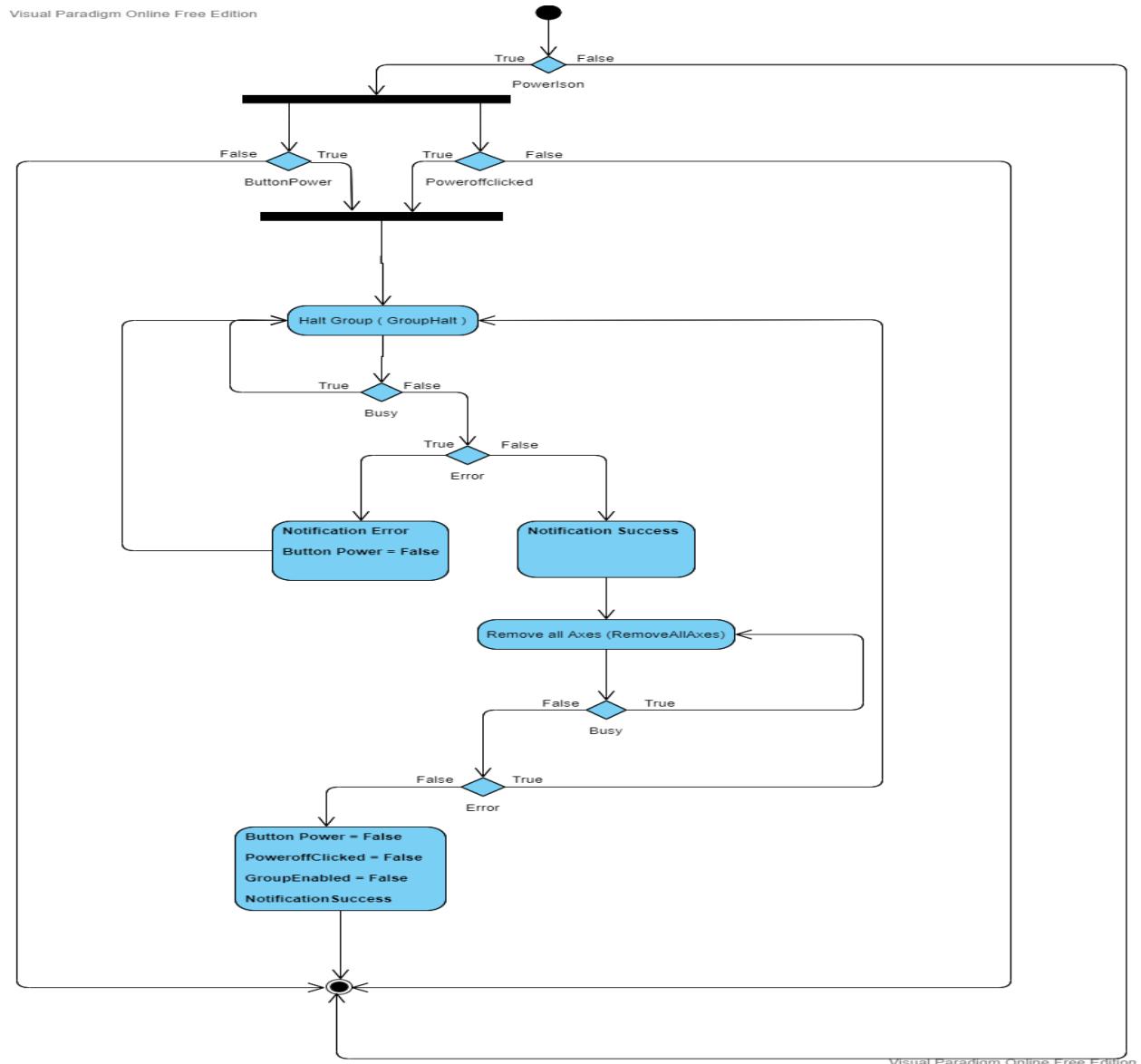


Abbildung 69: Aktivitätsdiagramm FB_Group_Halt

4.3.3.11. Movers Bewegungen

Dieser Teil des Programms konzentriert sich auf die Bewegung der Movers. Zunächst fährt die Gruppe von 5 Movern zu einer Fixposition (600 mm) im XTS. Dies geschieht mit dem Funktionsbaustein MC_MoveAbsoluteCA aus der Bibliothek Tc3_McCollisionAvoidance.

MC_MoveAbsoluteCA ist ein Funktionsbaustein, der eine Einzelachsbewegung zu der im Funktionsbaustein angegebenen absoluten Position mit Kollisionsvermeidung steuert. Die Kollisionsvermeidung hat eine höhere Priorität als die Bewegungssteuerung. Während der Ausführung kann die Achse langsamer werden oder warten, um eine Kollision zu vermeiden. Der Funktionsbaustein meldet jedoch erst dann fertig, wenn die Achse ihre Zielposition erreicht hat. Mit dieser Bausteinfunktion kann die Geschwindigkeit der Mover und der Abstand zwischen zwei aufeinanderfolgenden Movern (Gap) gesteuert werden.

Solange sich einer der Mover in der Fixstation befindet (jeder Mover hat eine ID), sendet der OPC UA Python Client dem Roboter eine MoveKUKA True-Variabel, um seine Arbeit zu erledigen. Nachdem der Roboter seine Aufgabe beendet hat, sendet der OPC UA Python Client eine MoveXTS True Variabel an das XTS Programm. Dann bewegt sich der Mover in der Fixstation zur Wartestation, und der Mover direkt davor bewegt sich zur Fixstation und durchläuft das gleiche Verfahren.

Nach der Wartezeit bewegt sich der Mover in der Warteposition kurz vor den letzten Mover und wartet, bis er wieder in der Fixposition ist.

4.3.3.12. Aktivitätsdiagramm: MC_MoveAbsoluteCA

Das folgende Aktivitätsdiagramm konzentriert sich auf die Bewegungen der Movers.

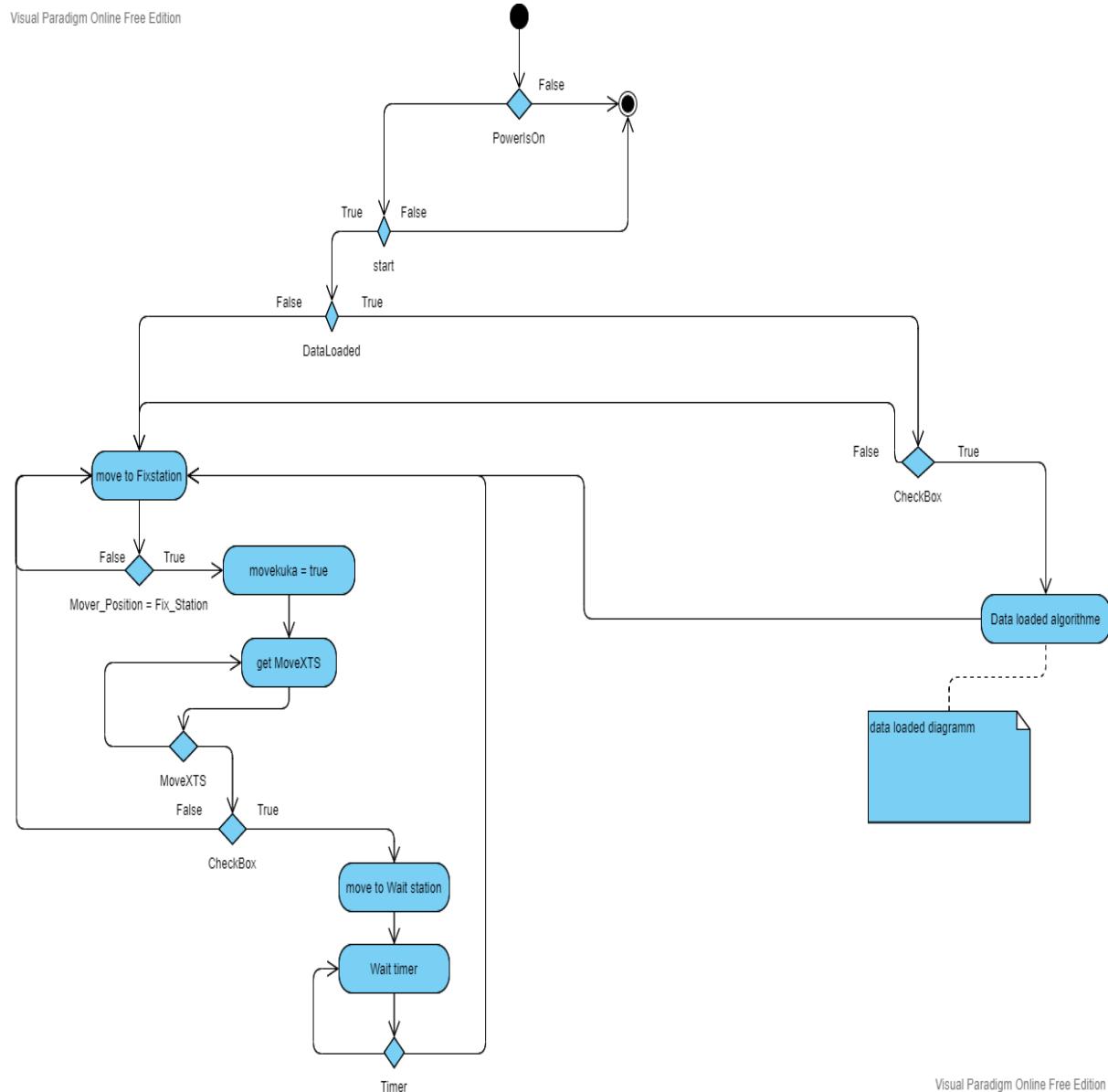


Abbildung 70: Aktivitätsdiagramm MC_MoveAbsoluteCA

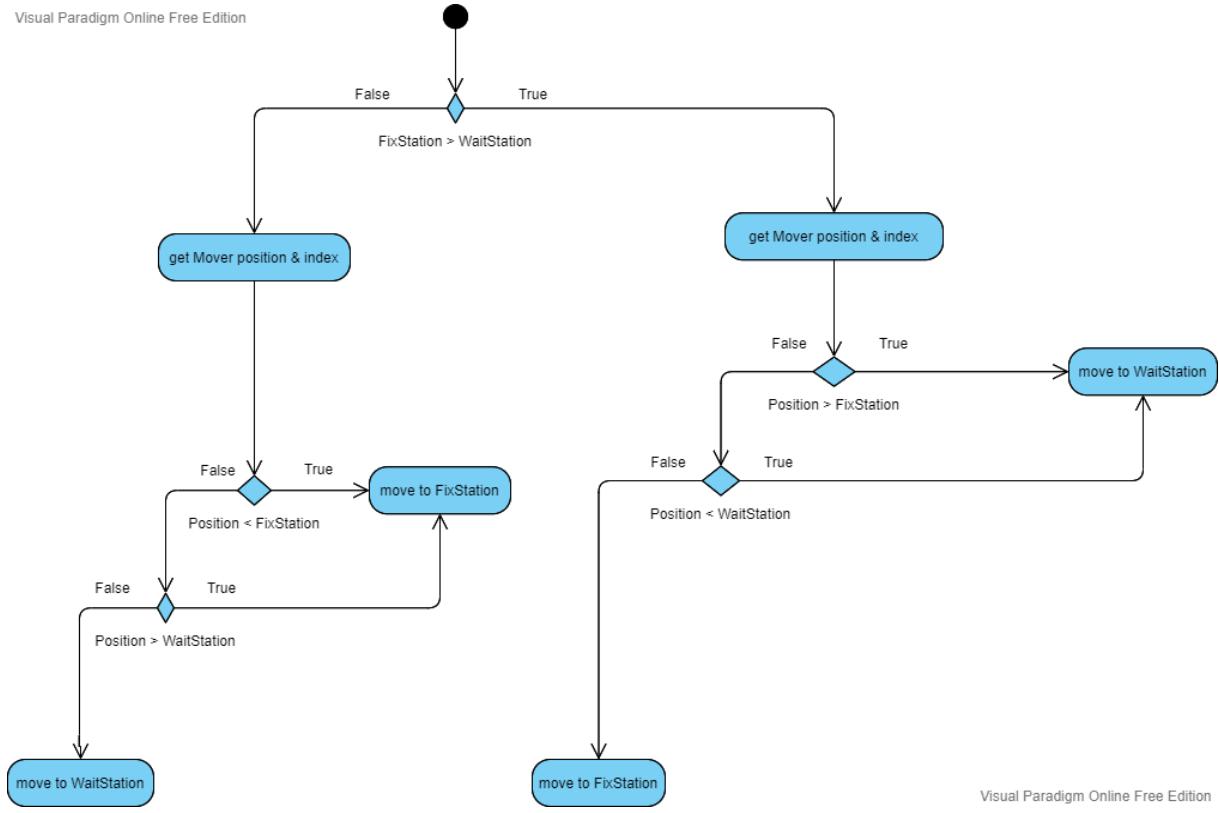


Abbildung 71: Aktivitätsdiagramm: Data_Loaded_Diagramm

4.3.3.13. Reset

Der Reset-Teil der Programmierung wird durch den Funktionsbaustein MC_GroupReset der Bibliothek Tc3_McCoordinatedMotion durchgeführt.

4.3.3.14. Aktivitätsdiagramm: MC_GroupReset

Das folgende Aktivitätsdiagramm konzentriert sich auf die Reset-Gruppe.

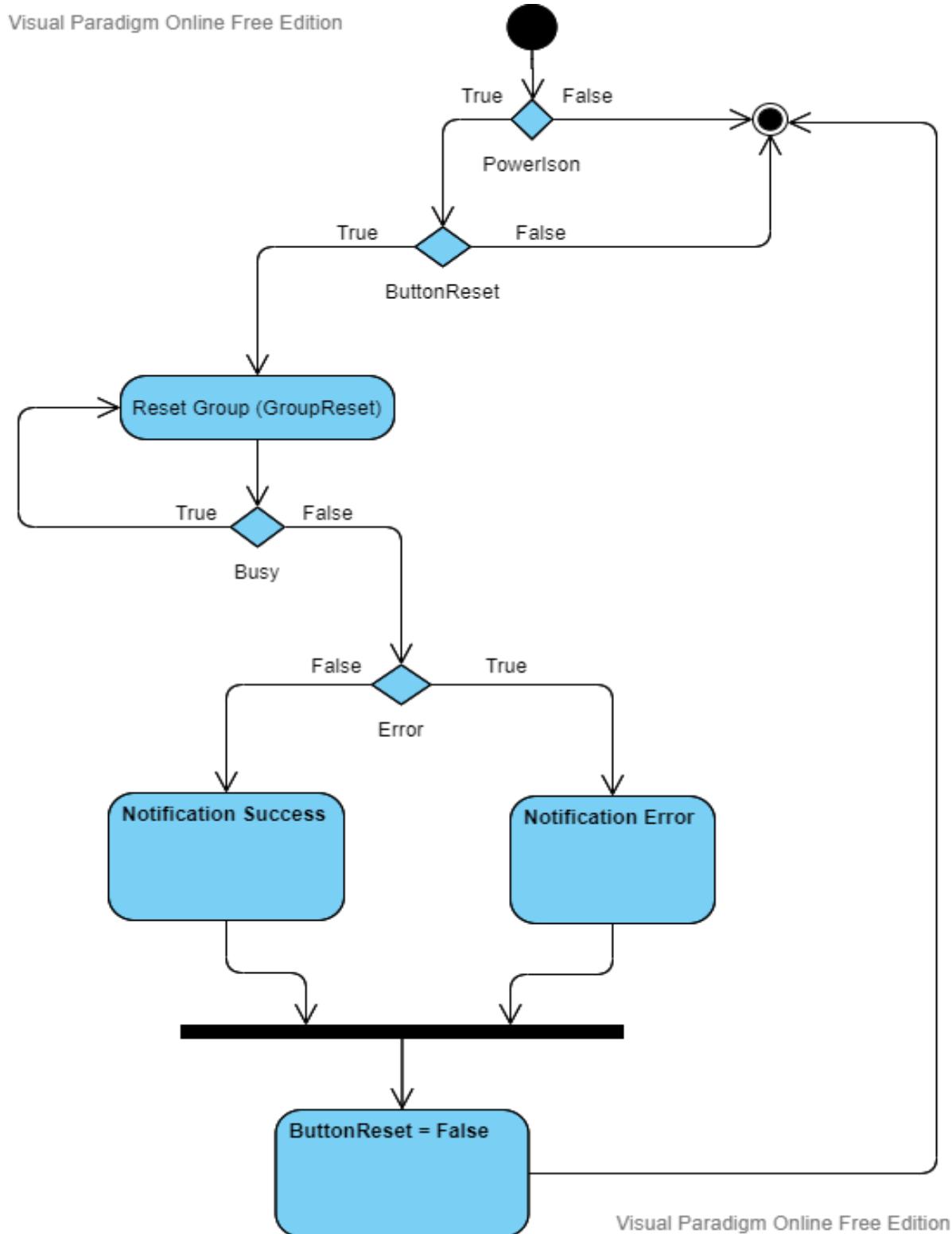


Abbildung 72: Aktivitätsdiagramm: MC_GroupReset

4.3.3.15. Movers Detection

Movers Detection ist ein Teil der Programmierung, der dafür verantwortlich ist, die Environments des XTS, die Anzahl der Module, die Anzahl der Mover, die Anzahl der Parts, die Anzahl der Tracks sowie die Position und den Status jedes Mover zu ermitteln.

4.3.3.16. Aktivitätsdiagramm: Movers Detection

Das folgende Aktivitätsdiagramm zeigt die Funktionen, die im Programmteil "Movers Detection" verwendet werden.

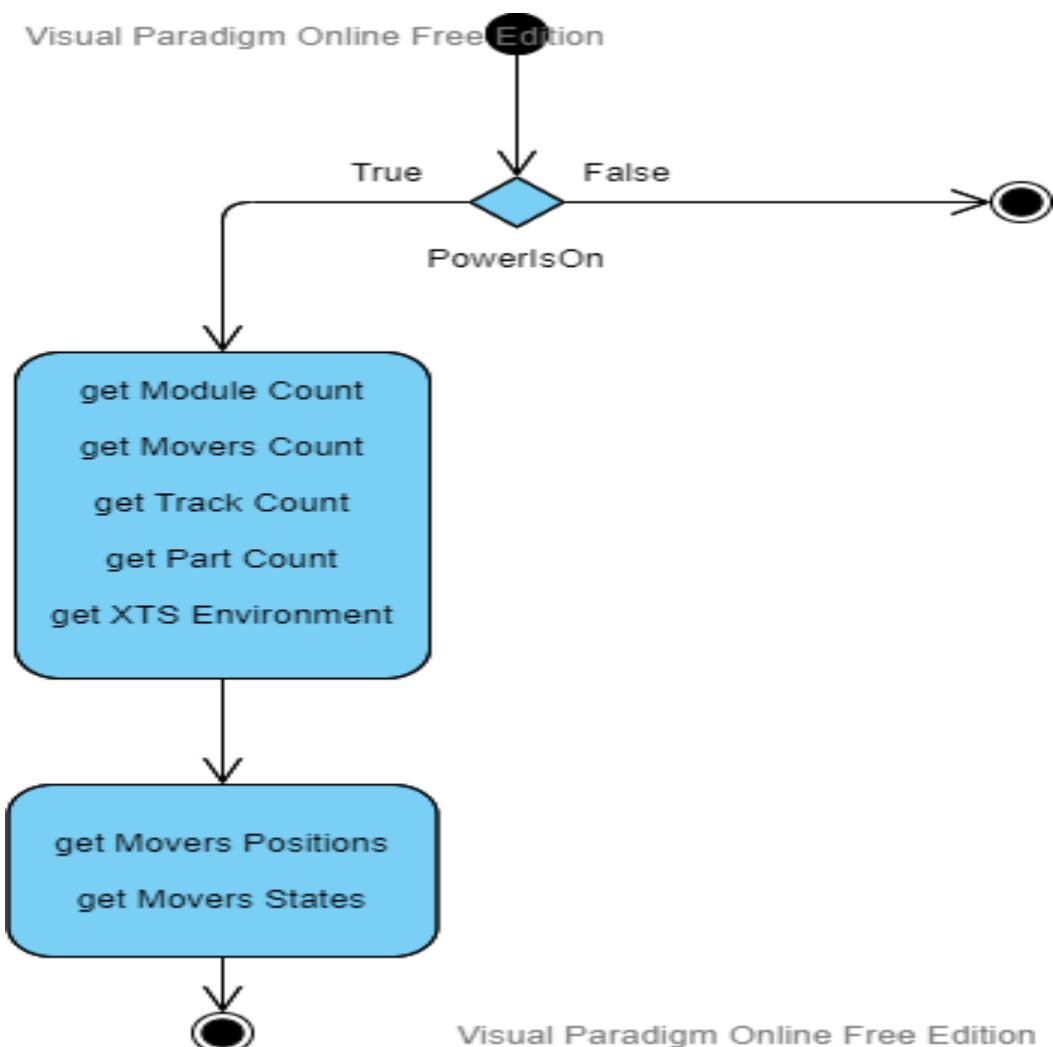


Abbildung 73: Aktivitätsdiagramm: Movers Detection

4.4. Python-Programmierung

Die Übertragung von Variablen zwischen dem KUKA Roboter und der XTS Maschine erfolgt über den OPC UA Python Client.

Zwei boolesche Variablen, MoveKUKA und Move XTS, werden mit Hilfe des OPC UA Python Clients zwischen den beiden Servern KUKAVARPROXY und OPC UA Server des XTS übertragen (siehe Abschnitt 3.4.).

4.4.1. Variabel MoveKUKA:

Das ist eine boolesche Variable mit der Initialisierung FALSE. Sobald sich der Mover in der Fixposition befindet, ändert sich der Wert von MoveKUKA auf TRUE. Der OPC UA Python Client erhält den Wert von MoveKUKA. Wenn MoveKUKA True ist, wird im Client ein Programm geschrieben, bevor Informationen an den Roboterserver gesendet werden.

das Programm wird in Abhängigkeit vom Zustand des Movers und seiner Bezeichnung (Index) geschrieben

Index nimmt den Wert von 1 bis 5 an, d. h. die Nummer des Movers.

Der Zustand des Movers nimmt den Wert 1 an, wenn der Mover voll ist, und 0, wenn der Mover leer ist.

Wenn der Mover voll ist, nimmt die Variable M einen geraden Wert von 2 bis 10 an. Der Client sendet diesen Wert an den Roboter. Die geraden Werte von M werden im Roboter programmiert, um den Mover leer zu machen (jeder Mover erhält einen geraden Wert).

Wenn der Mover leer ist, nimmt die Variable M einen ungeraden Wert von 1 bis 9 an. Der Client sendet diesen Wert an den Roboter. Die ungeraden Werte von M werden im Roboter programmiert, um den Mover mit dem Objekt zu füllen (jeder Mover erhält einen ungeraden Wert).

Der OPC UA Python-Client sendet den M-Wert an den Roboter, wobei er den Mover-Index und dessen Zustand wie in der folgenden Tabelle dargestellt berücksichtigt.

Index	Zustand des Mover's	Roboterbewegung	Beschreibung
1	Voll (1)	Case 1 (M=1)	Den Mover 1 entleeren.
	Leer (0)	Case 2 (M=2)	Den Mover 1 ausfüllen.
2	Voll (1)	Case 3 (M=3)	Den Mover 2 entleeren.
	Leer (0)	Case 4 (M=4)	Den Mover 2 ausfüllen.
3	Voll (1)	Case 5 (M=5)	Den Mover 3 entleeren.
	Leer (0)	Case 6 (M=6)	Den Mover 3 ausfüllen.
4	Voll (1)	Case 7 (M=7)	Den Mover 4 entleeren.
	Leer (0)	Case 8 (M=8)	Den Mover 4 ausfüllen.
5	Voll (1)	Case 9 (M=9)	Den Mover 5 entleeren.
	Leer (0)	Case 10 (M=10)	Den Mover 5 ausfüllen.

Tabelle 8: Bewegung des Roboters in Abhängigkeit von der Anzahl der Mover und deren Zustand

4.4.2. Variabel MoveXTS:

Das ist eine boolesche Variable mit der Initialisierung False. Sobald der Roboter seine Bewegung beendet hat und in seine Home-Position zurückkehrt, erhält der OPC UA Python-Client vom KUKAVARPROXY-Server die Information, dass die Variable MoveXTS True ist. Dann sendet der Client an den OPC UA Server des XTS, dass MoveXTS seinen Wert auf TRUE ändert. Das bedeutet, dass der Mover, der sich in der Fixposition befindet, vom Roboter bearbeitet wurde, von wo aus er sich bewegt und der Mover direkt dahinter seinen Platz einnimmt.

4.4.3. Aktivitätsdiagramm: Python-Programmierung

Das Aktivitätsdiagramm des Python-Programmierungsteils sieht folgendermaßen aus.

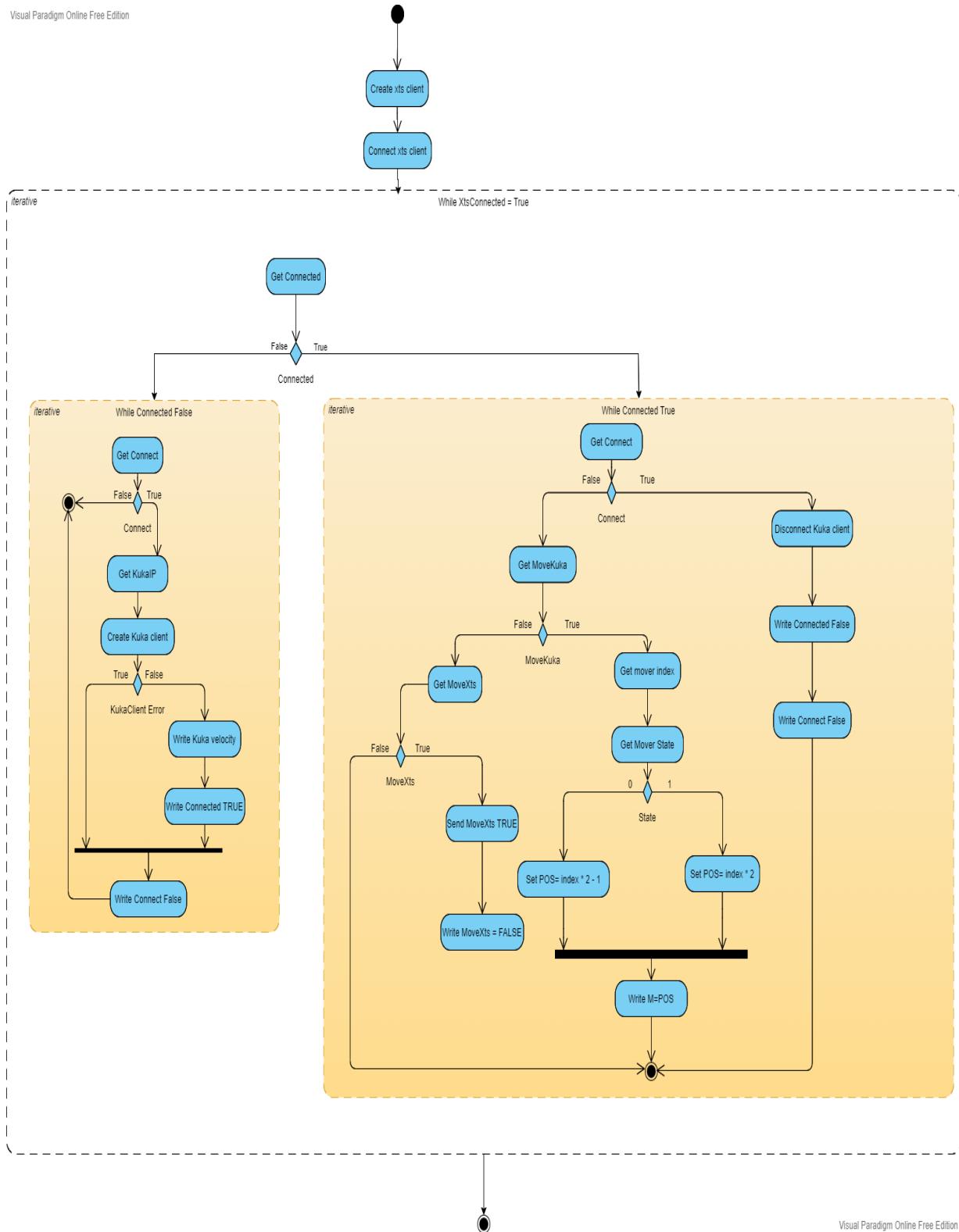


Abbildung 74: Aktivitätsdiagramm: Python-Programmierung

5. Human Machine Interface (HMI)

5.1. Auswahl einer Lösung

In diesem Teil des Projekts wurde untersucht, wie eine HMI für das im vorherigen Kapitel beschriebene Projekt erstellt werden kann.

Die erste Lösung besteht darin, ein Anzeigepanel im Programm TwinCat 3 zu erstellen, jedoch ist diese Lösung sehr einfach und bietet nicht viele Optionen zur Gestaltung des Ziels.

Die zweite gefundene Lösung ist die Erstellung einer HMI unter Verwendung der Beckhoff TE2000 Lizenz. Diese Lizenz erlaubt es, eine HMI mit mehr Optionen als die erste Möglichkeit zu erstellen. Sie ist auch professioneller und verfügt über verschiedene Informationen und Kontrollverfahren des Steuerungssystems.

5.2. TE2000 TwinCAT 3 HMI Engineering

Das TwinCAT 3 HMI (Human Machine Interface) integriert sich in die gewohnte Entwicklungsumgebung von Visual Studio®. Basierend auf aktuellen Webtechnologien (HTML5, JavaScript) ermöglicht es, plattformunabhängige Bedienoberflächen zu entwickeln. Diese agieren „responsive“ und passen sich automatisch der Auflösung, Größe und Orientierung an. Der grafische What-You-See-Is-What-You-Get (WYSIWYG)-Editor ermöglicht es, Controls einfach per Drag-and-drop auf der Oberfläche anzuordnen und mit Echtzeitvariablen zu verbinden. [\[81\]](#)

Das HMI ist auf allen Ebenen erweiterbar: Der Mix aus Standard-Controls und eigenen Designelementen erleichtert die Individualisierung. Außerdem lassen sich User-Controls aus den Standard-Controls erstellen und parametrieren, sodass der Baukasten von Controls einfach erweiterbar ist. Zur Erzeugung aufwendiger Seiten können definierte Vorlagen – z. B. von Designspezialisten – eingebunden werden. [\[81\]](#)

Die Logik des HMI kann clientseitig in JavaScript oder als sogenannte Server-Extension in .NET implementiert werden, wodurch das Know-how geschützt werden kann. [\[81\]](#)

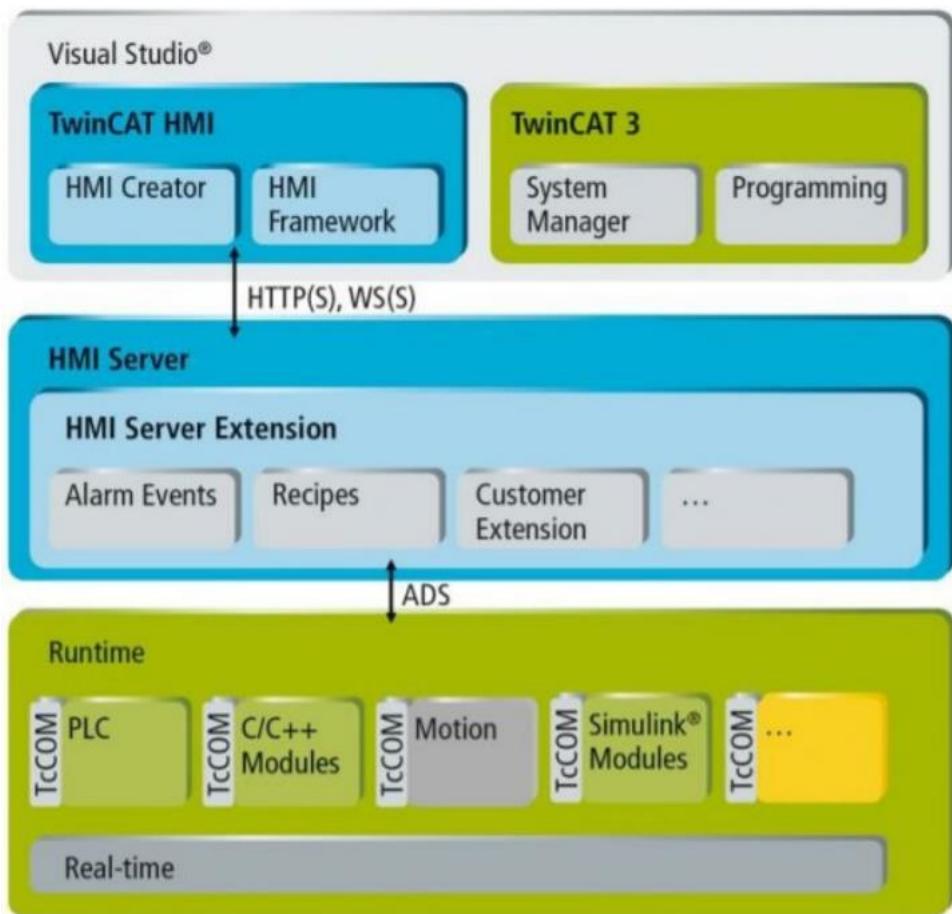


Abbildung 75: HMI-Architektur [85]

Die TwinCAT HMI integriert sich in die Entwicklungsumgebung Visual Studio® von Microsoft. TwinCAT3 stellt die kostenlose Visual Shell zur Verfügung, die auch mit der TwinCAT HMI genutzt werden kann. Damit ist es möglich, die SPS und die HMI in einem Tool zu entwickeln. Der TwinCAT HMI Creator ist ein Editor für die grafische Entwicklung der HMI. Das TwinCAT-HMI-Framework wird mit vordefinierten Steuerungen ausgeliefert. Die Kommunikation mit dem TwinCAT HMI Server findet über sichere Netzwerkverbindungen statt. Der TwinCAT HMI Server ist modular aufgebaut. HMI Server Extensions stellen zusätzliche Funktionen zur Verfügung. Darüber hinaus können eigene Kundenerweiterungen entwickelt werden, um nicht echtzeitrelevante Geschäftslogiken zentral im Server auszuführen. Es wird das TwinCAT-ADS-Protokoll (Automation Device Specification) unterstützt, das die Kommunikation mit allen TwinCAT-Geräten ermöglicht.

5.3. Erstellung der HMI

Mit Hilfe von TE2000 wurde die HMI des Projekts erstellt. Verschiedene Informationen, Variablen und Meldungen des Projekts hatten ihren Platz in der HMI.

Die realisierte HMI besteht aus drei Seiten, von denen jede in den folgenden Abschnitten beschrieben wird.

Die Seiten werden unter der Content-Datei des HMI-Projekts in der TwinCat 3 Software hinzugefügt.

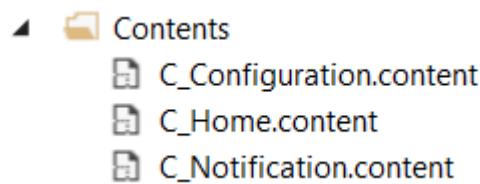


Abbildung 76: Content-Datei

5.3.1. Main-Seite

Die Main-Seite enthält verschiedene Schaltflächen und Informationen, wie in der folgenden Abbildung dargestellt:



Abbildung 77: Main-Seite

5.3.1.1. HMI-Navigation-Bar

Es ermöglicht die Navigation zwischen den drei Seiten des HMI.

Diese Navigation-Bar wird mit der Option Responsive Navigation Bar aus der HMI-Toolbox erstellt.

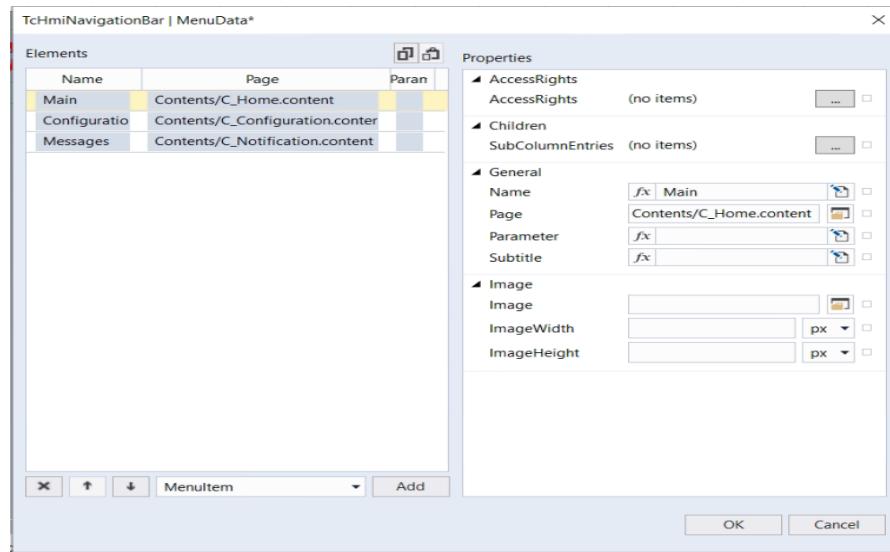


Abbildung 78: Menüdaten des HMI-Navigation-Bar

5.3.1.2. Login-Logout

Es identifiziert die Verwendung der HMI je nach Benutzertyp.

Ein Textblock neben der Schaltfläche zeigt an, welche Art von Benutzer das HMI verwendet.

Die Login-Logout erfolgt durch die Identifizierung der Benutzergruppe unter TwinCAT HMI Configuration.

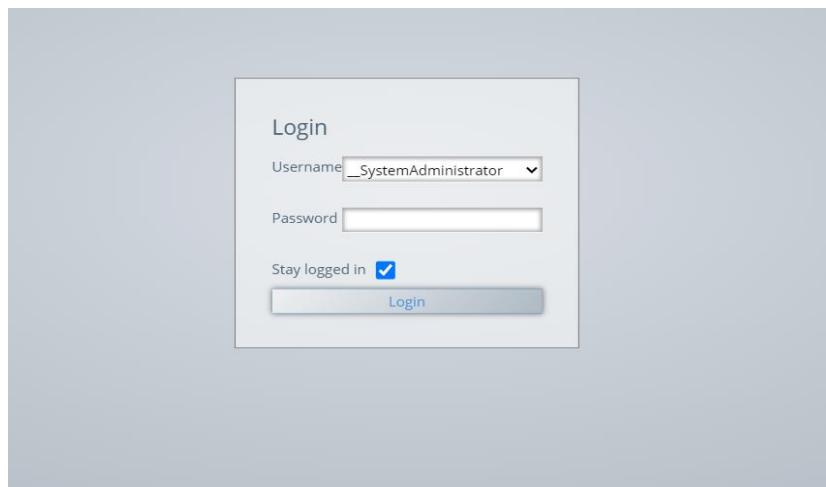


Abbildung 79: Login-Logout

5.3.1.3. Date-Time-Display

Sie zeigt das aktuelle Datum und die Uhrzeit auf dem HMI an.

Dies geschieht über den Date Time Display, der sich unter Common in der Toolbox befindet.

5.3.1.4. Power

Diese Schaltfläche ist für das Ein- und Ausschalten des XTS-Geräts zuständig.

Diese Schaltfläche entspricht der ButtonPower-Variablen in der globalen Variablenliste des GVL_HMI-Programms.

Wenn der Benutzer auf die Power-Taste klickt, wechselt deren Farbe von rot auf grün und eine Meldung über der HMI zeigt an, dass der Benutzer die Maschine eingeschaltet hat. Wenn der Benutzer zum zweiten Mal auf die Schaltfläche klickt, wird eine Bestätigungsmeldung angezeigt. Wenn der Benutzer das Ausschalten akzeptiert, indem er auf OK klickt, wechselt die Farbe der Schaltfläche wieder zu Rot.

5.3.1.5. Reset

Diese Schaltfläche wird zum Reset des Programms verwendet und entspricht der Variable Button Reset in GVL_HMI des Programms.

Wenn der Benutzer des HMI auf die Schaltfläche Reset klickt, wird eine Meldung darüber angezeigt, dass er einen Reset durchgeführt hat.

Wenn Sie auf die Schaltfläche "Reset" klicken, wechselt die Farbe zu gelb und wird wieder grau, wenn der Reset abgeschlossen ist.

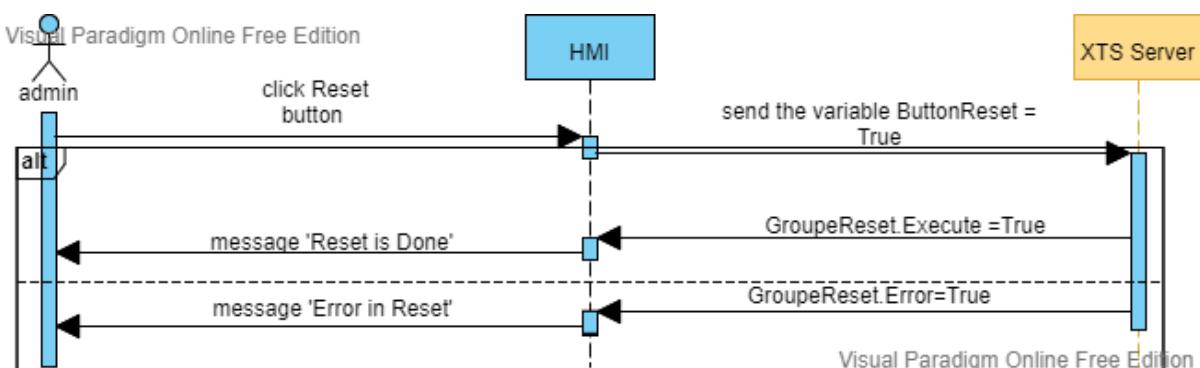


Abbildung 80: Sequenzdiagramm: Reset

5.3.1.6. Start-Stop

Diese Schaltfläche ist für das Starten oder Stoppen des Programms verantwortlich und entspricht der Variable ButtonStart in GVL_HMI des Programms.

Wenn sich das Programm nicht in der Startphase befindet, nimmt die Schaltfläche die grüne Farbe an, auf der das Wort START steht. Wenn sich das Programm im START-Modus befindet, färbt sich die Taste rot und zeigt das Wort STOP an.

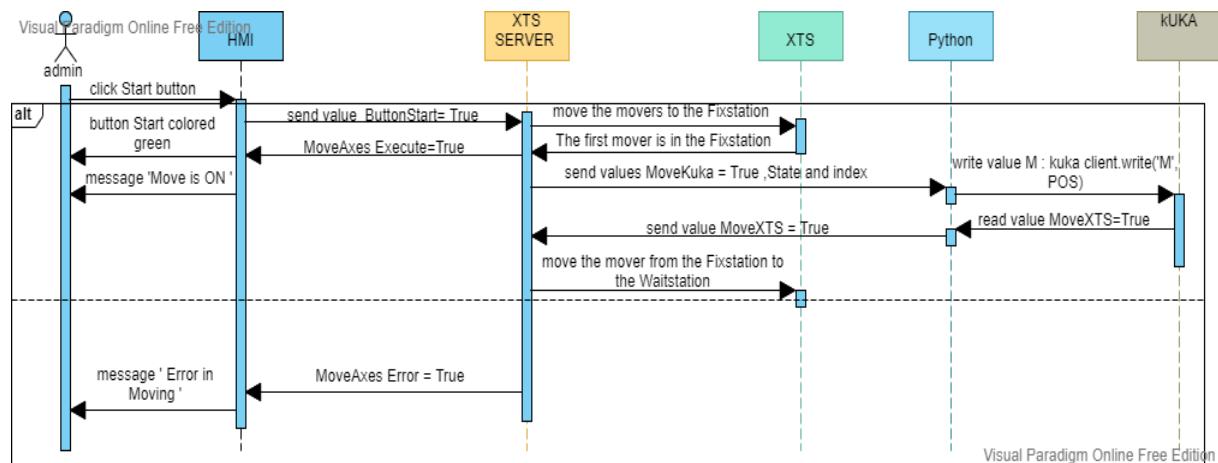


Abbildung 81: Sequenzdiagramm: Start

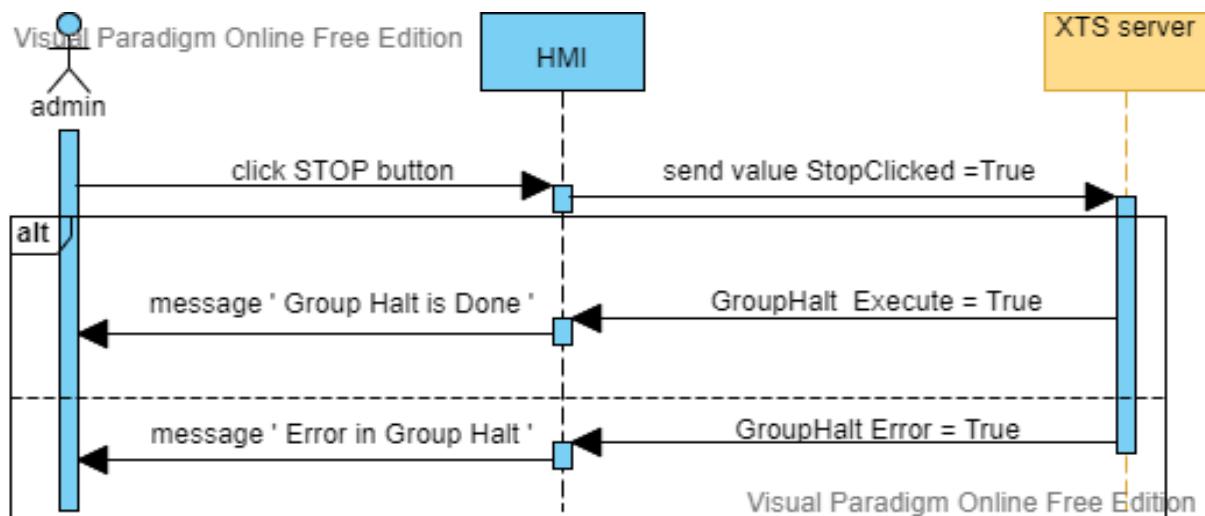


Abbildung 82: Sequenzdiagramm: Stop

5.3.1.7. Connect

Mit dieser Schaltfläche kann die Verbindung zwischen den beiden Programmen des Roboters und des XTS hergestellt werden, sie entspricht den beiden Variablen Connect und Connected von GVL_HMI.

die Schaltfläche nimmt die Farbe Rot an und enthält das Wort " Not Connected", wenn die Verbindung nicht hergestellt ist. Wenn man auf die Schaltfläche klickt, um die Verbindung herzustellen, nimmt sie die Farbe Grün an und zeigt das Wort "Connected" an.

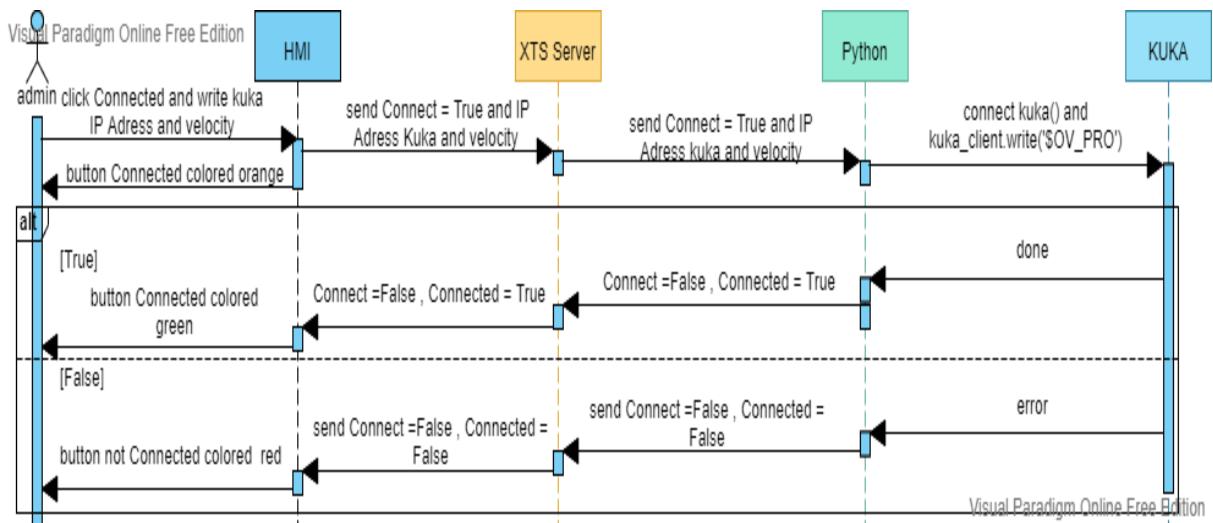


Abbildung 83: Sequenzdiagramm: Connect

5.3.1.8. XTS-Informations

In diesem Teil werden die Informationen über das XTS-Gerät angezeigt.

- XTS_Parts: ermöglicht es dem Benutzer, die Anzahl der Parts des XTS zu sehen, sein TextBlock entspricht der Part_Count-Variablen im MoversDetection-Teil des Programms.
- XTS_Tracks: ermöglicht es dem Benutzer, die Anzahl der Tracks des XTS zu sehen, sein TextBlock entspricht der Variabel Track_Count im MoversDetection-Teil des Programms.
- XTS_Movers: Ermöglicht es dem Benutzer, die Anzahl der Movers des XTS zu sehen, sein TextBlock entspricht der Variabel Movers_Count in der GVL_Config des Programms.
- XTS_Modules: ermöglicht es dem Benutzer, die Anzahl der Module des XTS zu sehen. Sein TextBlock entspricht der Variabel Module_Count im MoversDetection-Teil des Programms.

5.3.1.9. Datagrid

Data Grid ist eine Funktion unter Common in der HMI Toolbox.

die verschiedenen DataGrid-Spalten entsprechen der GVL-Variablen M im Programm.

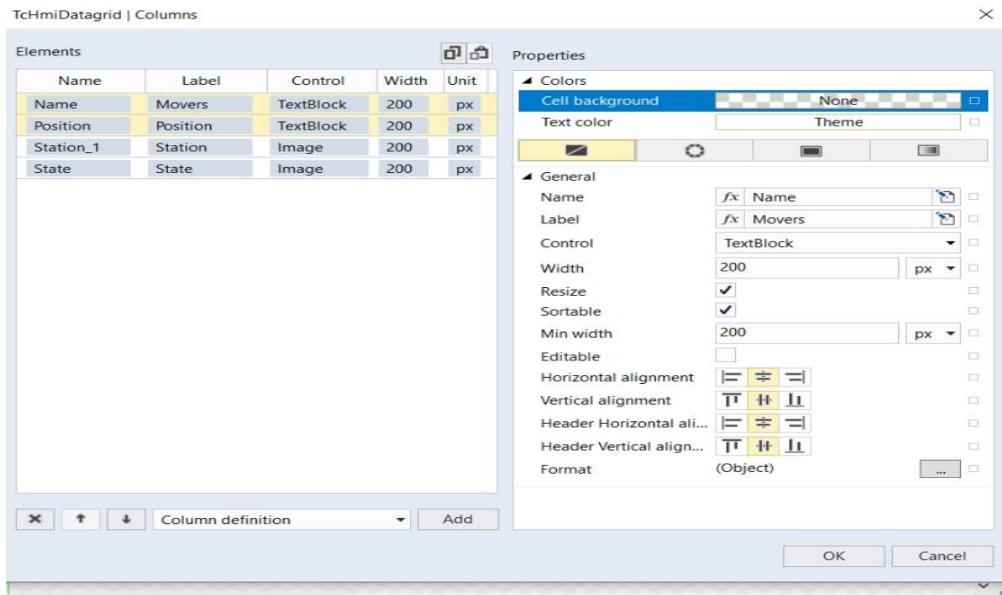


Abbildung 84: DataGrid-Columns

DataGrid enthält die folgenden Informationen:

a. Movers

Dies ist die erste Spalte, in der die Anzahl der jeweiligen Mover angegeben ist.

b. Position

In dieser Spalte wird die Position der einzelnen Mover im XTS in Millimetern angegeben.

c. Station

Dies ist die Spalte der Fixstation, in der die Mover gefüllt oder geleert werden. Jeder Mover kann je nach Status drei Farben annehmen:

- Grüne Farbe: Der Mover befindet sich in der Fixstation.
- Orange Farbe: Der Mover befindet sich nicht in der Fixstation.
- Rote Farbe: Es liegt ein Fehler vor.

d. Station 2

Dies ist die Spalte der Wartestation, in der die Mover eine Zeit lang anhalten. Jeder Mover kann je nach Status drei Farben annehmen:

- Grüne Farbe: Der Mover befindet sich in der Wartestation.
- Orange Farbe: Der Mover befindet sich nicht in der Wartestation.
- Rote Farbe: Es liegt ein Fehler vor.

e. State

In dieser Spalte wird der Status des Movers angezeigt (voll oder leer).

- Grüne Farbe: der Mover ist leer.
- Rote Farbe: Der Mover ist voll.

5.3.2. Configuration-Seite

Die Configuration-Seite enthält verschiedene Programmsteuerungselemente und Informationen, wie in der folgenden Abbildung dargestellt:

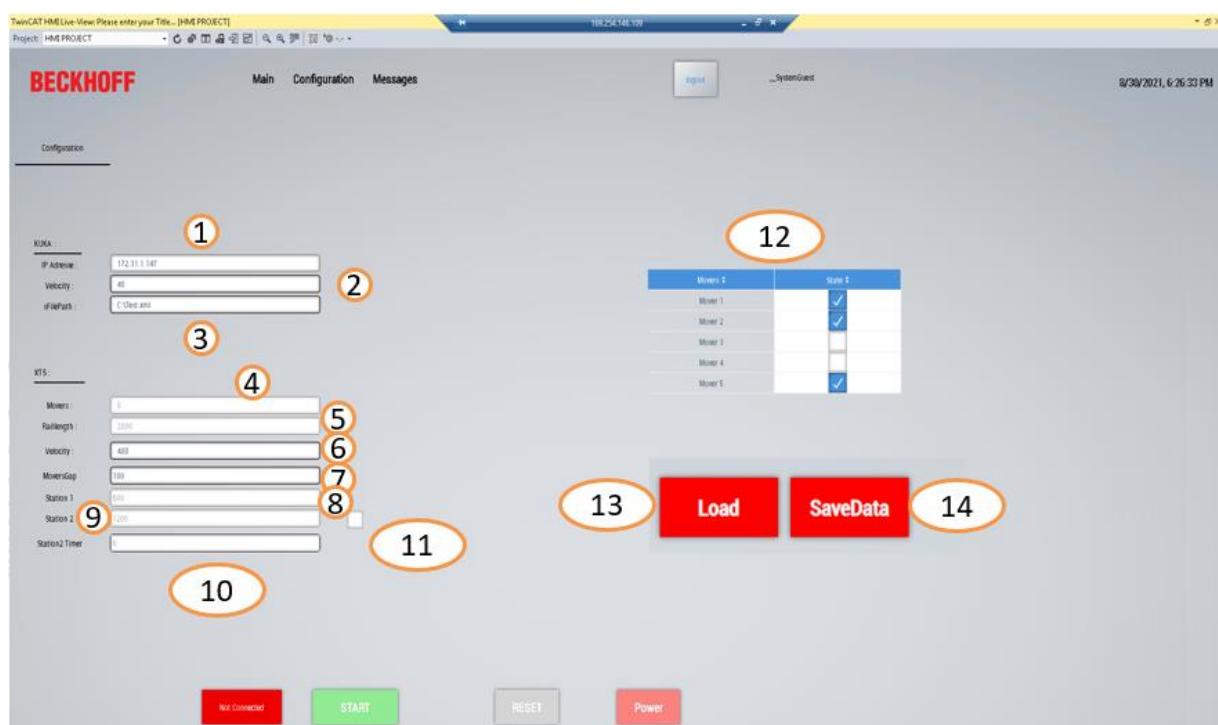


Abbildung 85: Configuration-Seite

5.3.2.1. KUKA-IP-Adress

Es handelt sich um einen Input, der über die Variabel KUKA_IP_Adress in der GVL_Config des Programms entspricht.

Es ermöglicht dem Benutzer, die IP-Adresse des KUKA-Roboters einzugeben.

5.3.2.2. KUKA-Velocity

Dies ist ein numerischer Input, die der Variabel KUKA_Velocity in der GVL_Config des Programms entspricht.

Mit diesem Eingang kann der Benutzer die Geschwindigkeit des Roboters eingeben. Die Geschwindigkeit muss zwischen 0 und 100 liegen. Das HMI zeigt eine erklärende Meldung an, wenn der eingegebene Wert nicht in diesem Bereich liegt.

5.3.2.3. sFilePath

Dies ist ein Input, mit dem der Benutzer den Speicherort der XML-Datei für den Fall eines Stromausfalls angeben kann. Dieser Input entspricht der Variabel sFilePath im Main-Programm.

5.3.2.4. Movers

Dies ist ein numerischer Input, die der Benutzer die Anzahl der im XTS vorhandenen Mover ablesen kann.

Dieser Input entspricht der Variabel Movers_Count der GVL_Config des Programms.

5.3.2.5. Raillength

Dies ist ein numerischer Input, mit der der Benutzer die Länge der XTS-Maschine ablesen kann.

Dieser Input entspricht der Variabel RailLength von GVL_Config des Programms.

5.3.2.6. XTS-Velocity

Dies ist ein numerischer Input, es erlaubt dem Benutzer, die Geschwindigkeit der Mover einzugeben. Diese Geschwindigkeit muss zwischen 0 und 700 liegen, sonst wird eine erklärende Meldung angezeigt.

Dieser Input entspricht der Variabel MoversVelocity der GVL_Config des Programms.

5.3.2.7. *Movers-Gap*

Dies ist ein numerischer Input, es ermöglicht dem Benutzer, den Abstand zwischen zwei aufeinander folgenden Movern einzugeben. Dieser Abstand muss mindestens 100 mm und maximal 400 mm betragen, sonst wird eine erklärende Meldung angezeigt.

Dieser Input entspricht der Variabel MoversGap der GVL_Config des Programms.

5.3.2.8. *Station-1*

Es handelt sich um einen numerischen Input, der dem Benutzer erlaubt, die Position der Fixstation (600 mm) im XTS abzulesen.

Dieser Input entspricht der Variabel Station von GVL_Config des Programms.

5.3.2.9. *Station-2*

Dies ist ein numerischer Input, es ermöglicht dem Benutzer, die Warteposition in Millimetern im XTS einzugeben. Diese Station muss zwischen 0 mm und Raillength liegen, sonst wird eine erklärende Meldung angezeigt.

Dieser Input entspricht der Variabel Station2 von GVL_Config des Programms.

5.3.2.10. *Station-2-Timer*

Dies ist ein numerischer Input, es ermöglicht dem Benutzer, die Wartezeit in Sekunden des Movers in der Wartestation einzugeben. Diese Wartezeit muss zwischen 0 s und 100 s liegen, sonst wird eine erklärende Meldung angezeigt.

Dieser Input entspricht der Variabel Station2_timer von GVL_Config des Programms.

5.3.2.11. *CheckBox*

Mit der Checkbox kann der Benutzer entscheiden, ob er die Wartestation benutzen möchte oder nicht.

5.3.2.12. Movers-State-Initialisation

Dies ist eine Datagrid, mit der der Benutzer den Status jedes Movers (voll oder leer) eingeben kann, bevor er das Programm startet.

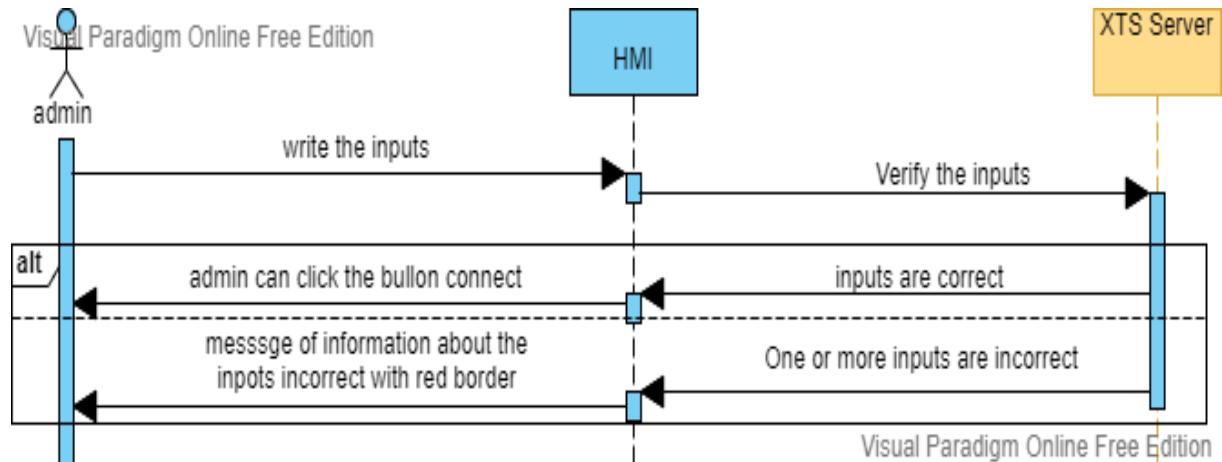


Abbildung 86: Sequenzdiagramm: Inputs

5.3.2.13. Load

Es handelt sich um eine Schaltfläche, mit der die Applikation ab dem Zeitpunkt des Stromausfalls betrieben werden kann.

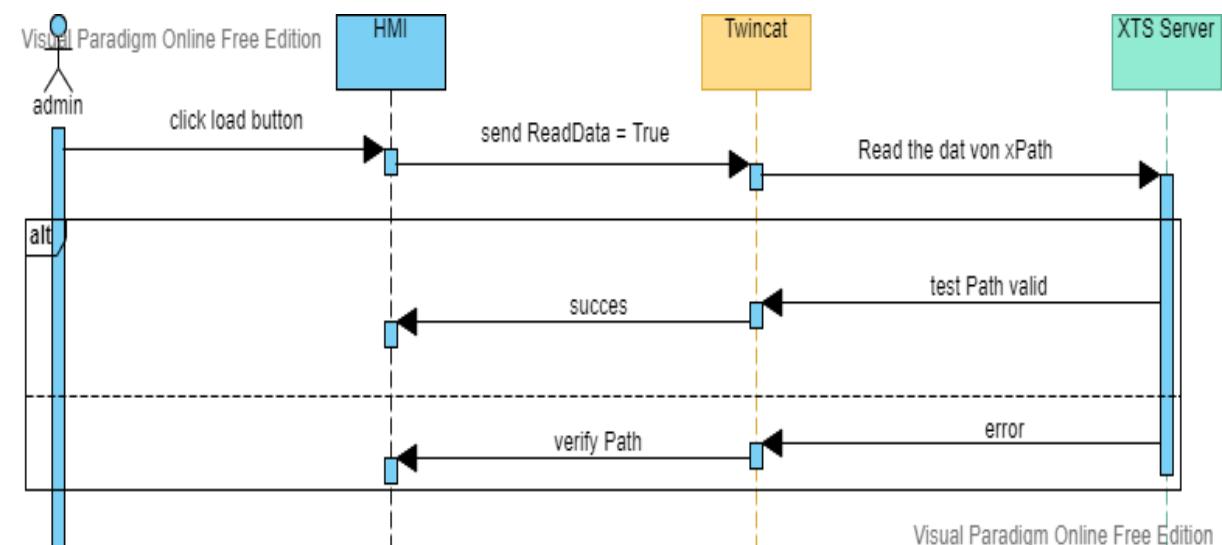


Abbildung 87: Sequenzdiagramm: Load

5.3.2.14. SaveData

Dies ist eine Schaltfläche, die mit der Funktion Read Data des Programms verbunden ist. Sie kann verschiedene Daten aus der Applikation aufzeichnen, wie z.B. die IP-Adresse des Roboters, seine Geschwindigkeit und die Positionen der Mover, ...

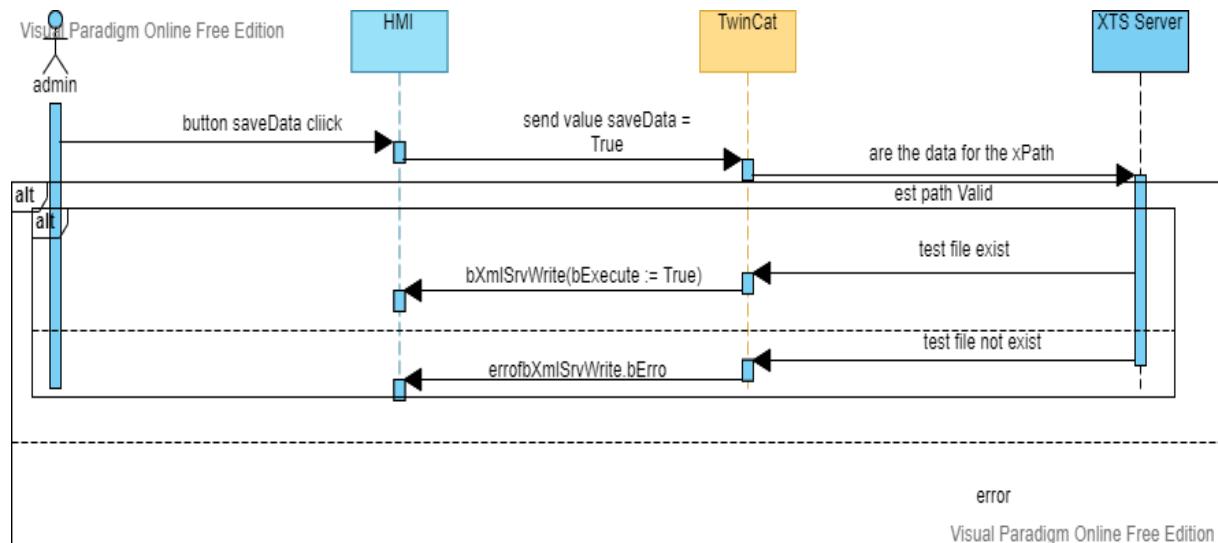


Abbildung 88: Sequenzdiagramm: SaveData

5.3.3. Messages-Seite

Die Messages-Seite enthält verschiedene Informationen über das Programm, wie in der folgenden Abbildung dargestellt:

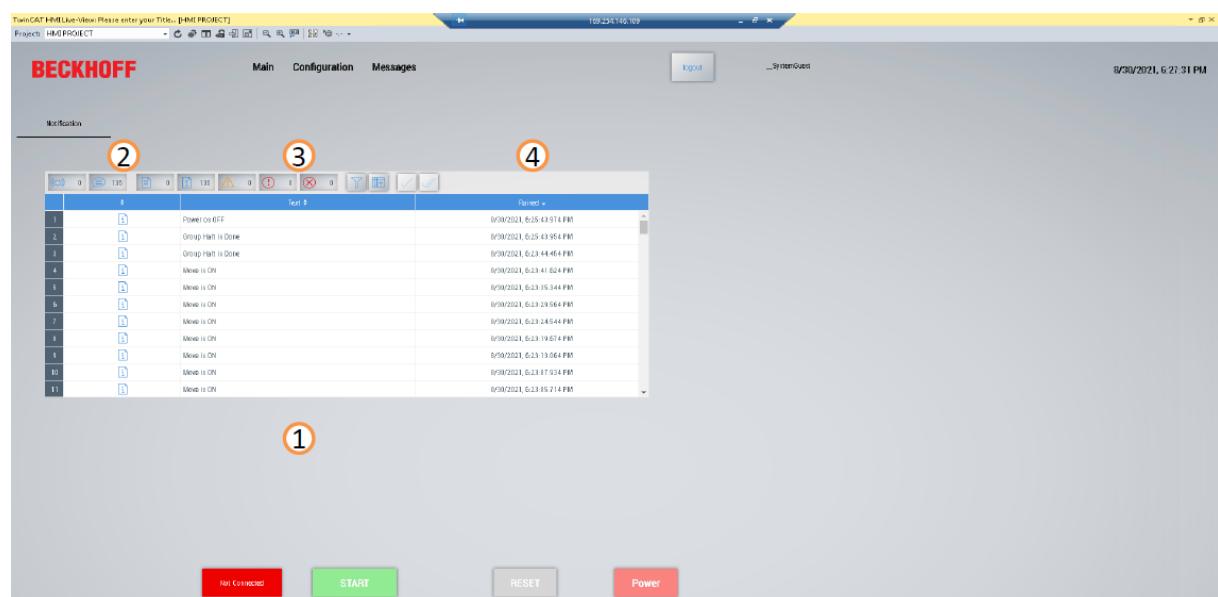


Abbildung 89: Messages-Seite

5.3.3.1. Event-Grid

Das EventGrid wird in drei Spalten erstellt, wie in der folgenden Abbildung dargestellt:

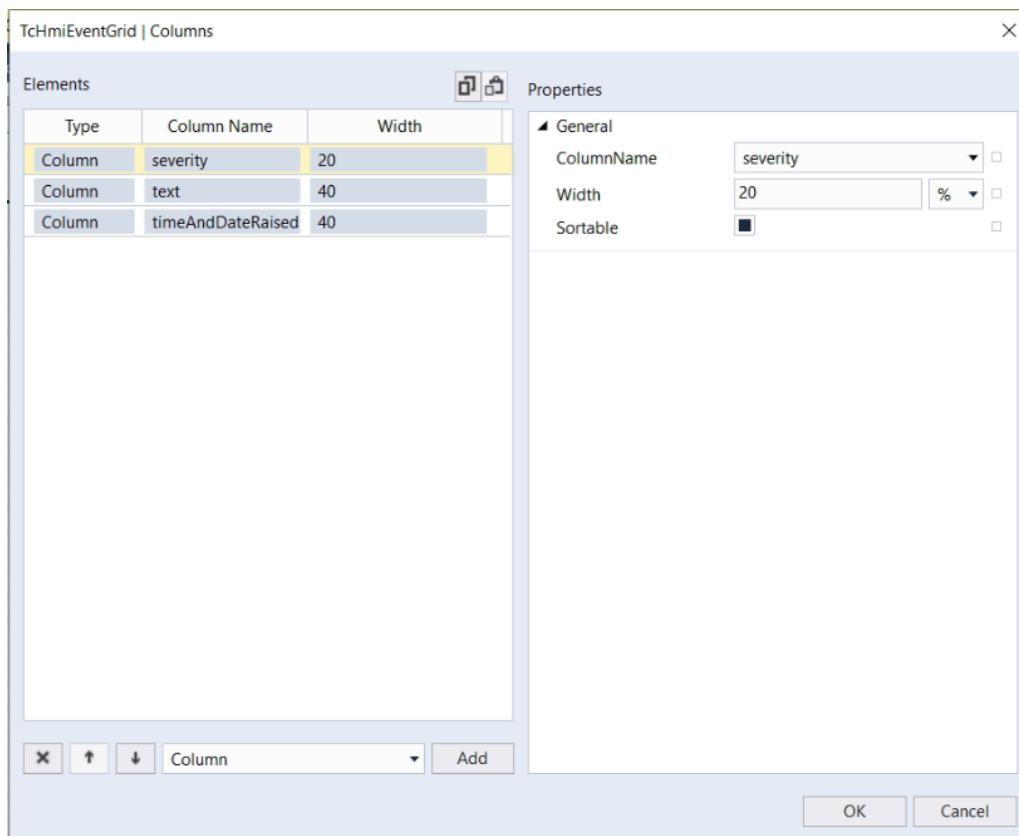


Abbildung 90: EventGrid-Spalten

Event-Grid ermöglicht die Anzeige von Informationen, Warnungen oder Fehlern zu verschiedenen Funktionen und Schritten des Programms.

5.3.3.2. Symbol

Diese Spalte ist für die Anzeige des Meldungssymbols (Information, Warnung oder Fehler) zuständig.

5.3.3.3. Text

Diese Spalte ist für die Anzeige der Meldung zuständig.

5.3.3.4. Raised

Diese Spalte zeigt das Datum und die Uhrzeit der Meldung an.

6. Validierung der Projekt-Ergebnisse

6.1. Erfüllung von Anforderungen

Die Kommunikation zwischen der Beckhoff XTS-Maschine und dem KUKA KRC4 Compact-Roboter über das Kommunikationsprotokoll OPC UA ist eine notwendige Anforderung des Projekts.

Diese Kommunikation wird erreicht, indem auf beiden Maschinen zwei OPC UA Server eingerichtet werden und ein OPC UA Client auf einem Computer läuft, der über zwei Ethernet-Kabel mit beiden Maschinen verbunden ist.

Die Programmierung der XTS-Maschine über die Software TwinCat 3 ist ebenfalls eine Anforderung des Projekts.

Auch die Programmierung des KUKA-Roboters ist eine Anforderung des Projekts.

Ziel ist der Austausch von Variablen zwischen den beiden Programmen der beiden Maschinen über die beiden OPC-UA Server und den OPC-UA Client.

Die Erstellung einer HMI, die es dem Benutzer ermöglicht, die Befehle der Beispielanwendung zu steuern und die verschiedenen Phasen der Applikation zu visualisieren, ist eine Anforderung des Projekts.

Diese Anforderungen werden in dem Projekt alle erfüllt.

6.2. Erreichen der Projekt-Ziele

Das Hauptziel des Projekts ist die Kommunikation, die durch die Implementierung eines OPC UA Servers mit der TF 6100 Lizenz innerhalb des TwinCat 3 Programms erreicht wird.

Das Problem besteht darin, dass der im T65-Labor vorhandene KUKA-Roboter die OPC UA-Technologie nicht unterstützt. Die Lösung besteht in der Verwendung eines Open Source KUKAVARPROXY, der die gleichen Funktionen des OPC UA-Servers unterstützt. KUKAVARPROXY ist in der Robotersteuerung implementiert und funktioniert als OPC UA Server.

Der OPC UA Client wird als Python Applikation auf einem Computer erstellt. Dieser Client erzeugt einen Aufruf der beiden Python-Bibliotheken "KUKAVARPROXY Message Format" und "FREEOPCUA PYTHON".

Diese beiden Bibliotheken werden aufgerufen, um einen Python OPC UA Client zu erstellen, der Daten zwischen den beiden Servern austauschen kann.

Das Beckhoff XTS-Maschinenprogramm wird von der TwinCat 3-Software geschrieben, wobei die Sicherheitsanforderungen eingehalten werden, solange sich die Movers bewegen, um das Risiko einer Kollision zu vermeiden.

Das Programm des KUKA Roboters wird unter Berücksichtigung der Sicherheitsanforderungen geschrieben, indem der Arbeitsbereich des Roboters definiert wird.

Die Kommunikation zwischen den beiden Maschinen erfolgt erfolgreich über den Computer, wobei jede Maschine einen OPC UA Server und der Computer einen OPC UA Client enthält.

In der HMI werden alle Projektanforderungen wie die Konfiguration der Projektbefehle, die Visualisierung der Applikationsschritte, die Visualisierung der Informationen der beiden Maschinen und die Anzeige von Benachrichtigungen und Erfolgs-, Besetzt- und Fehlermeldungen in der erstellten HMI dargestellt.

Auf diese Weise wurden alle Projektziele erfolgreich erreicht.

7. Zusammenfassung und Ausblicke

Im Rahmen der Masterarbeit wurden verschiedene Schritte durchgeführt, um das Projekt in der gewünschten Weise abzuschließen und die Anforderungen des Projekts zu erfüllen. Die Forschungsphase konzentrierte sich auf das Konzept der Industrie 4.0, insbesondere auf die Methoden und Vorgehensweisen, die dabei angewendet wurden. Danach wurde eine Recherche zur Kommunikationstechnologie OPC UA durchgeführt, die den Begriff OPC Classic, die Spezifikationen des Kommunikationsprotokolls OPC UA sowie dessen Daten- und Kommunikationsmodellierung, die Beziehung zwischen OPC UA und Industrie 4.0 sowie die Grundprinzipien der Modellierung von OPC UA spezifizierte. Es wurde eine Forschung zur Charakterisierung der Programmiersprache Python durchgeführt. Eine Recherche zum Begriff HMI wurde durchgeführt, indem seine Spezifikation und seine Rolle in der Industrie beschrieben wurden. Außerdem wurde eine Recherche über den KUKA KR C4 Compact-Roboter durchgeführt, indem seine Komponenten, die zugehörige Software, der in seiner Steuerung implementierte KUKAVARPROXY-Server, seine Kinematik, sein Bewegungsbereich, seine Betriebsarten und seine Bewegungsarten definiert wurden. Außerdem wurde eine Forschung über das Beckhoff Extended Transport System (XTS) durchgeführt, indem seine Komponenten, Eigenschaften und die zugehörige Programmiersoftware spezifiziert wurden.

Danach wurde die Kommunikation zwischen dem KUKA-Roboter und der XTS-Maschine untersucht und eine vollständige Kommunikationsarchitektur zwischen diesen beiden digitalisierten Maschinen über das OPC UA-Kommunikationsprotokoll spezifiziert. Der erste Teil befasste sich mit der Untersuchung der Kommunikation des KUKA-Roboters über die OPC UA-Kommunikationstechnologie. In dieser Phase wurde der KUKAVARPROXY-Server in den KUKA-Roboter implementiert, die einzelnen Schritte der Implementierung beschrieben und der KUKAVARPROXY Message Format Client auf dem Computer installiert. Der zweite Teil befasste sich mit der Untersuchung der Kommunikation der XTS-Maschine über das OPC UA-Kommunikationsprotokoll. In dieser Phase wurde die Lizenz TF 6100 identifiziert, die für die Erstellung eines OPC UA-Servers im IPC Beckhoff verantwortlich ist, sowie die verschiedenen Schritte der Installation, Lizenzaktivierung und Konfiguration definiert und spezifiziert. Im dritten Teil wurde die Kommunikation zwischen dem Python-Client untersucht, der in einem Computer erstellt wurde, der mit den beiden digitalisierten Maschinen (dem KUKA-Roboter und der XTS-Maschine) über zwei Ethernet-Kabel verbunden wurde. Dabei wurde untersucht, wie der Server des KUKAVARPROXY-

Roboters mit dem Python-Client über die KUKAVARPROXY Message Format Library verbunden wurde und wie der OPC UA-Server, der in der XTS-Maschine installiert ist, mit dem Python-Client über die Free OPC UA Python Bibliothek verbunden wurde. Danach wurde der Austausch von Variablen zwischen den beiden Servern OPC UA der XTS-Maschine und KUKAVARPROXY des Roboters und dem Python-Client, der auf dem Computer erstellt wurde, getestet.

Im weiteren Verlauf wurde eine Applikationsstudie durchgeführt. Im ersten Teil lag der Schwerpunkt auf der Programmierung des Roboters entsprechend den Anforderungen der auszuführenden Applikation. Zunächst wurden die Sicherheitsanforderungen für den Einsatz des Roboters behandelt und erklärt, wie die Variablen, die mit der XTS-Maschine ausgetauscht werden sollen, konfiguriert wurden, um anschließend den Roboter entsprechend den Anforderungen der Applikation programmieren zu können. Im zweiten Teil lag der Schwerpunkt auf der Programmierung der XTS-Maschine. Die bei der Programmierung verwendeten Lizenzen und Bibliotheken wurden identifiziert und anschließend die einzelnen Schritte des XTS-Programms beschrieben. Danach wurde die Programmierung des OPC UA-Clients behandelt und wie Daten zwischen dem OPC UA-Client und den beiden Servern der beiden digitalisierten Maschinen (KUKA-Roboter und XTS-Maschine) ausgetauscht werden.

Als Nächstes wurde die Identifizierung der verschiedenen Komponenten der realisierten HMI untersucht. Dieser Teil wurde in drei Hauptseiten unterteilt. Auf der ersten Seite wurden das LogIn des Benutzers, die Befehlsschaltflächen und die NavigationBar untersucht, die es dem Benutzer ermöglicht, zwischen den Seiten zu navigieren. Außerdem wurde eine Tabelle erstellt, die die Identifikation jedes Movers, seine Position, seinen Status (voll oder leer) und ob er sich in einer festen Position oder in der Warteposition befindet oder nicht, enthält. Auf der zweiten Seite des HMI wurden die verschiedenen Konfigurationsanforderungen für den KUKA-Roboter und den XTS von Beckhoff realisiert. Auf der dritten Seite wurde eine Tabelle erstellt, in der die Meldungen zu den verschiedenen Schritten der Applikation angezeigt wurden. Dabei wurde angegeben, dass jeder Schritt erfolgreich abgeschlossen wurde oder dass bei einem bestimmten Schritt der Applikation ein Fehler aufgetreten war.

Zuletzt wurde sich auf die Validierung der Projektergebnisse konzentriert, indem überprüft wurde, ob die Anforderungen eingehalten und die Projektziele erreicht wurden.

Um einen Ausblick auf die Zukunft dieses Projekts zu geben, wenn der KUKA-Roboter KR C4 Compact die OPC UA-Kommunikationstechnologie direkt unterstützt, kann die Python-Anwendung und damit auch der Computer überflüssig gemacht werden. Daher ist es möglich,

die beiden digitalisierten Maschinen (den KUKA-Roboter und die XTS-Maschine) direkt zu verwenden, während ein OPC UA Server im Beckhoff IPC der XTS-Maschine und ein OPC UA Client in der Robotersteuerung erstellt wird. Dadurch wird die Kommunikationsarchitektur weiter vereinfacht. Außerdem können Sensoren für jedes Case der konstruierten Box verwendet werden, um sicherzustellen, dass sich die Teile in ihrem Case befinden oder nicht. Zusätzlich könnte man eine reale Visualisierung der beiden digitalisierten Maschinen (KUKA-Roboter und XTS-Maschine) in das HMI integrieren, aber diese Verbesserung ist derzeit noch nicht möglich, weil die Firma Beckhoff noch dabei ist, sie zu entwickeln.

8. Anhänge

Add-all-axes-Programm

```
done := TRUE;
busy := FALSE;
error := FALSE;

FOR i:=1 TO AxesCount DO
    fbaddtoaxis[i] (AxesGroup:=GroupAxes, Axis := Axes[i], Execute :=
execute, IdentInGroup := UDINT_TO_IDENTINGROUP(i) );

    IF (NOT fbaddtoaxis[i].done) THEN
        done := FALSE;
        IF NotifAxesAdded THEN
            Notifications :=
fbMessage.CreateEx(TC_EVENTS.AddAxes.Axes_Added,0);
            Notifications := fbMessage.Send(0);
        END_IF

    END_IF
    IF (fbaddtoaxis[i].busy) THEN
        busy := TRUE;
    END_IF
    IF (fbaddtoaxis[i].Error) THEN
        error := TRUE;
    END_IF
END_FOR
IF done THEN
    Notifications := fbMessage.CreateEx(TC_EVENTS.AddAxes.Axes_Added,0);
    Notifications := fbMessage.Send(0);
END_IF
IF error THEN
    Notifications :=
fbMessage.CreateEx(TC_EVENTS.AddAxes.AddAxes_Error,0);
    Notifications := fbMessage.Send(0);
END_IF
```

Power-Programm

```
PowerIsOn := TRUE;
error:=FALSE;
FOR i:=1 TO AxesCount DO
    FBPower[i] (Axis:=Axes[i],
        Enable          := enable,
        Enable_Positive := enable,
        Enable_Negative := enable
    (* Status      : BOOL;    (* B *)
       Busy        : BOOL;    (* V *)
       Active      : BOOL;    (* V *)
       Valid       : BOOL; *) (* E *) (* not implemented - use Busy
       Error       : BOOL;    (* B *)
       ErrorID     : UDINT;   (* E *) *)
    );
    IF (FBPower[i].Error) THEN
        PowerIsOn := FALSE;
        error:=TRUE;
    END_IF
    IF (NOT FBPower[i].Status) THEN
        PowerIsOn := FALSE;
    END_IF
END FOR
```

Reset-Programm

```
done := TRUE;
busy := FALSE;
error := FALSE;
FOR i:=1 TO AxesCount DO
    fbreset[i] (Axis:=Axes[i], Execute := execute);
    IF (NOT fbreset[i].done) THEN
        done := FALSE;
    END_IF
    IF (fbreset[i].busy) THEN
        busy := TRUE;
    END_IF
    IF (fbreset[i].Error) THEN
        error := TRUE;
    END_IF
END FOR
```

MoversDetection-Programm

```
IF NOT bInit THEN
    IF fbXtsEnvironment.Init(TRUE) THEN
        fbXtsEnvironment.Init(FALSE);
        bInit:= TRUE;
    END_IF
    RETURN;
END_IF
IF NOT CInit THEN
    IF fbXtsXpuPart.Init(bExecute:=TRUE, nObjectId :=
fbXtsEnvironment.XpuTcIo(1).P_PartOids[1]) THEN
        fbXtsXpuPart.Init(bExecute:=FALSE, nObjectId :=
fbXtsEnvironment.XpuTcIo(1).P_PartOids[1]);
        CInit:= TRUE;
    END_IF
    RETURN;
END_IF
GVL_Config.Movers_Count := fbXtsEnvironment.XpuTcIo(1).GetMoverCount();
Module_Count := fbXtsEnvironment.XpuTcIo(1).PartTcIo(1).GetModuleCount();
Track_Count := fbXtsEnvironment.XpuTcIo(1).GetTrackCount();
Part_Count := fbXtsEnvironment.XpuTcIo(1).GetPartCount();
fbXtsEnvironment.XpuTcIo(1).GetMoverPositions();
Z:= fbXtsEnvironment.XpuTcIo(1).P_MoverPositions;
FOR i:=1 TO 5 DO
    M[i].Name := CONCAT('Mover ', TO_STRING(i));
    M[i].Position := LREAL_TO_INT(Z[i].fPartPosition) MOD
LREAL_TO_INT(RailLength) ;
IF Main.PowerAll.PowerIsOn = FALSE THEN
    M[i].Station_1 := '2';
ELSIF Main.PowerAll.PowerIsOn = TRUE AND M[i].Position =
GVL_Config.Station THEN
    M[i].Station_1 := '0';

ELSIF M[i].Position <> GVL_Config.Station AND M[i].Position <
GVL_Config.RailLength THEN
    M[i].Station_1 := '1';
END_IF
IF Main.PowerAll.PowerIsOn = FALSE THEN
    M[i].Station_2 := '2';
ELSIF Main.PowerAll.PowerIsOn = TRUE AND M[i].Position =
GVL_Config.Station2 THEN
    M[i].Station_2 := '0';
ELSIF M[i].Position <> GVL_Config.Station2 AND M[i].Position <
GVL_Config.RailLength THEN
    M[i].Station_2 := '1';
END_IF
END_FOR
IF NOT ButtonStart THEN
    FOR i:=1 TO 5 DO
        IF M[i].BoolState = TRUE THEN
            M[i].State := '0';
        ELSIF M[i].BoolState = FALSE THEN
            M[i].State := '1';
        END_IF
    END_FOR
END_IF
```

Main-Programm

```
IF NOT clearedEvents THEN
    events.ClearLoggedEvents(ipClearSettings:=settings);
    clearedEvents:=TRUE;
END_IF

IF GVL_HMI.ButtonSaveData THEN

    fbXmlSrvWrite(
        nMode      := XMLSRV_ADDMISSING,
        pSymAddr   := ADR(Data),
        cbSymSize  := SIZEOF(Data),
        sFilePath  := sFilePath,
        sXPath     := sXPath,
        bExecute   := TRUE
    );
    IF NOT fbXmlSrvWrite.bBusy THEN
        IF NOT fbXmlSrvWrite.bError THEN
            GVL_HMI.ButtonSaveData:=FALSE;
            fbXmlSrvWrite(bExecute := FALSE);
        END_IF
    END_IF
END_IF

IF GVL_HMI.ButtonReadData THEN
    fbXmlSrvRead(
        nMode      := XMLSRV_ADDMISSING,
        pSymAddr   := ADR(Data),
        cbSymSize  := SIZEOF(Data),
        sFilePath  := sFilePath,
        sXPath     := sXPath,
        bExecute   := TRUE
    );
    IF NOT fbXmlSrvRead.bBusy THEN
        IF NOT fbXmlSrvRead.bError THEN
            GVL_Config.KUKA_IP_Adress := Data.KUKA_IP_Adress;
            GVL_Config.KUKA_Velocity := Data.KUKA_Velocity;
            M := Data.M;
            GVL_Config.Movers_Count:= Data.Movers_Count;
            GVL_Config.MoversGap:=Data.MoversGap;
            GVL_Config.Station:=Data.Station;
            GVL_Config.Station2:=Data.Station2;
            GVL_HMI.EnableDisableCheckBox:=data.EnableDisableCheckBox;
            fbXmlSrvRead(bExecute := FALSE);
            //ReadData:=FALSE;
        END_IF
    END_IF
END_IF

MoversDetection();

PowerAll(enable:=TRUE);
GrpStatus(AxesGroup:=GroupAxes, enable:=TRUE);
GrpError(AxesGroup:=GroupAxes, enable:=TRUE);
```

```

IF PowerAll.PowerIsOn THEN

    IF ButtonPower AND NOT GVL_Config.GroupEnabled THEN
        CASE AddEnableState OF
            0: AddAllAxes(Execute:=FALSE);
                AddAllAxes(Execute:=TRUE);
                AddEnableState := 1;
            1: AddAllAxes();
                IF (NOT AddAllAxes.Busy) THEN
                    IF (NOT AddAllAxes.Error) THEN
                        GrpNcToPlc:=GroupAxes.NcToPlc.Common;
                        GroupState:=GrpNcToPlc.GroupStatus.State;
                        IF (GroupState = mcGroupStateErrorStop) THEN
                            AddEnableState := 2; // group stop
                        ELSE
                            AddEnableState := 4;
                        END_IF

                    END_IF
                END_IF
                // group stop
            2: fbgroupstop(AxesGroup:=GroupAxes, Execute:=FALSE);
                fbgroupstop(AxesGroup:=GroupAxes, Execute:=TRUE);
                AddEnableState := 3;
            3: fbgroupstop(AxesGroup:=GroupAxes);
                fbgroupstop.Execute := (NOT unlockgrpstop);
                IF (fbgroupstop.Done AND (NOT fbgroupstop.Busy)) THEN
                    IF (NOT fbgroupstop.Error) THEN

                        unlockgrpstop := FALSE;
                        AddEnableState := 0;
                    END_IF
                END_IF

                // enable group
            4: GroupEnable(AxesGroup:=GroupAxes, execute:=FALSE);
                GroupEnable(AxesGroup:=GroupAxes, execute:=TRUE);
                AddEnableState := 5;
            5: GroupEnable(AxesGroup:=GroupAxes);
                IF (GroupEnable.Done) THEN
                    Notifications :=
fbMessage.CreateEx(TC_EVENTS.GroupEnable.GroupEnable_Done,0);
                    Notifications := fbMessage.Send(0);
                    AddEnableState := 0;
                    GVL_Config.GroupEnabled:=TRUE;
                    ButtonPower:=FALSE;
                END_IF

            END_CASE
        END_IF
    END_IF

```

```

IF ButtonPower AND PowerOffClicked THEN
    CASE PowerOffState OF

        0: GroupHalt(AxesGroup:=GroupAxes, Execute:=FALSE);
            GroupHalt(AxesGroup:=GroupAxes, Execute:=TRUE);
            PowerOffState := 1;
        1: GroupHalt(AxesGroup:=GroupAxes);
            IF (NOT GroupHalt.Busy) THEN
                IF (NOT GroupHalt.Error) THEN
                    PowerOffState := 2;
                    NotifGrpHalt:=0;
                    Notifications :=

fbMessage.CreateEx(TC_EVENTS.GroupHalt.GroupHalt_Done,0);
                    Notifications := fbMessage.Send(0);

                ELSE
                    Notifications :=
fbMessage.CreateEx(TC_EVENTS.GroupHalt.GroupHalt_Error,0);
                    Notifications := fbMessage.Send(0);
                    ButtonPower:=FALSE;
                    PowerOffState := 0;

                END_IF
            END_IF
        2: RemoveAllAxes(AxesGroup:=GroupAxes, Execute:=FALSE);
            RemoveAllAxes(AxesGroup:=GroupAxes, Execute:=TRUE);
            PowerOffState := 3;
        3: RemoveAllAxes(AxesGroup:=GroupAxes);
            IF (NOT RemoveAllAxes.Busy) THEN
                IF (NOT RemoveAllAxes.Error) THEN
                    ButtonPower:=FALSE;
                    PowerOffClicked:=FALSE;
                    GVL_Config.GroupEnabled:=FALSE;
                    Notifications :=

fbMessage.CreateEx(TC_EVENTS.Power_Notification.PowerButton_OFF,0);
                    Notifications := fbMessage.Send(0);
                END_IF
                PowerOffState := 0;
            END_IF
    END_CASE
    ///////////////////////////////////////////////////
END_IF

```

```

IF ButtonReset THEN

    CASE ResetState OF

        0:  GroupReset(AxesGroup:=GroupAxes, Execute:=FALSE);
            GroupReset(AxesGroup:=GroupAxes, Execute:=TRUE);
            ResetState := 1;
        1:  GroupReset(AxesGroup:=GroupAxes);
            IF (NOT GroupReset.Busy) THEN
                IF (NOT GroupReset.Error) THEN
                    Notifications :=
                Notifications :=

fbMessage.CreateEx(TC_EVENTS.ResetAll.Reset_Done,0);
                Notifications := fbMessage.Send(0);

                ResetState := 2;
                NotifReset:=TRUE;

                ELSE
                    NotifResetError:=TRUE;

                    Notifications :=
fbMessage.CreateEx(TC_EVENTS.ResetAll.Reset_Error,0);
                    Notifications := fbMessage.Send(0);
                END_IF
            END_IF

        END_CASE
        fbTonDelay(IN := NOT fbTonDelay.Q, PT:= T#1S);
        IF fbTonDelay.Q THEN
            NotifReset:=FALSE;
            NotifResetError:=FALSE;
            ButtonReset:=FALSE;
            ResetState := 0;
        END_IF

    END_IF

```

```

IF ButtonStart THEN
    StopClicked:=FALSE;

    CASE Station2State OF

        0: FOR i:=1 TO AxesCount DO
            IF PowerOffClicked THEN
                ButtonStart:=FALSE;
                StopClicked:=TRUE;

                EXIT;
            END_IF
            MoveAxes[i] (axis:=Axes[i], execute:=FALSE);
            MoveAxes[i] (axis:=Axes[i], execute:=TRUE,
position:=GVL_Config.Station, velocity:=GVL_Config.MoversVelocity,
gap:=GVL_Config.MoversGap,Direction:=1);
            END_FOR

            Station2State :=1;

        1:
            FOR i:=1 TO AxesCount DO
                IF M[i].Position = GVL_Config.Station THEN
                    Movekuka :=TRUE;

                    index:=i;
                    State:=M[i].State;
                    Station2State :=2;
                    IF M[index].State= '1' THEN
                        M[index].State := '0';
                        M[index].BoolState := TRUE;
                    ELSE
                        M[index].State := '1';
                        M[index].BoolState := FALSE;
                    END_IF
                    EXIT;
                END_IF

            END_FOR

        2:
        IF index>0 THEN
            IF M[index].Position > GVL_Config.Station+1 THEN
                IF NOT EnableDisableCheckBox THEN
                    Station2State :=3;
                ELSE
                    Station2State :=4;
                END_IF

            END_IF
        END_IF
    END_CASE

```

```

IF MoveXTS THEN
    FOR i:=1 TO AxesCount DO
        IF PowerOffClicked THEN
            ButtonStart:=FALSE;
            StopClicked:=TRUE;
            EXIT;
        END_IF
        IF M[i].Position = GVL_Config.Station
THEN
            MoveAxes[i] (axis:=Axes[i],
execute:=FALSE);
            MoveAxes[i] (axis:=Axes[i],
execute:=TRUE, position:=GVL_Config.Station + 50,
velocity:=GVL_Config.MoversVelocity,
gap:=GVL_Config.MoversGap,Direction:=1);
            MoveXTS := FALSE;
            Notifications :=
fbMessage.CreateEx(TC_EVENTS.Move_Movers.Move_ON,0);
            Notifications := fbMessage.Send(0);
            EXIT;

            END_IF
        END_FOR
    END_IF
    3: MoveAxes[index] (axis:=Axes[index], execute:=FALSE);
    MoveAxes[index] (axis:=Axes[index], execute:=TRUE,
position:=GVL_Config.Station, velocity:=GVL_Config.MoversVelocity,
gap:=GVL_Config.MoversGap,Direction:=1);
    Station2State :=1;
    4: MoveAxes[index] (axis:=Axes[index], execute:=FALSE);
    MoveAxes[index] (axis:=Axes[index], execute:=TRUE,
position:=GVL_Config.Station2, velocity:=GVL_Config.MoversVelocity,
gap:=GVL_Config.MoversGap,Direction:=1);
    Station2State :=1;
END_CASE

IF EnableDisableCheckBox THEN
    FOR i:=1 TO AxesCount DO
        IF M[i].Position = GVL_Config.Station2 THEN
            index2:=i;
            EXIT;
        END_IF
    END_FOR
    IF index2>0 THEN
        ConvertToTime :=
LREAL_TO_TIME(GVL_Config.station2_timer*1000);
        fbTonDelay2(IN := NOT fbTonDelay2.Q, PT:=
ConvertToTime);
        IF fbTonDelay2.Q THEN
            MoveAxes[index2] (axis:=Axes[index2],
execute:=FALSE);
            MoveAxes[index2] (axis:=Axes[index2],
execute:=TRUE, position:=GVL_Config.Station,
velocity:=GVL_Config.MoversVelocity,
gap:=GVL_Config.MoversGap,Direction:=1);

            END_IF
        END_IF
    END_IF
END_IF

```

```

IF StopClicked THEN
    Station2State:=0;
CASE HaltState OF

    // GroupHalt
    0: GroupHalt(AxesGroup:=GroupAxes, Execute:=FALSE);
        GroupHalt(AxesGroup:=GroupAxes, Execute:=TRUE);
        HaltState := 1;
    1: GroupHalt(AxesGroup:=GroupAxes);
        IF (NOT GroupHalt.Busy) THEN
            IF (NOT GroupHalt.Error) THEN

                HaltState := 0;
                StopClicked:=FALSE;
                Notifications :=
fbMessage.CreateEx(TC_EVENTS.GroupHalt.GroupHalt_Done,0);
                Notifications := fbMessage.Send(0);
            ELSE
                Notifications :=
fbMessage.CreateEx(TC_EVENTS.GroupHalt.GroupHalt_Error,0);
                Notifications := fbMessage.Send(0);
            END_IF
        END_IF
    END_CASE

END_IF

END_IF
IF MoveXTS THEN

    FOR i:=1 TO AxesCount DO
        IF PowerOffClicked THEN
            ButtonStart:=FALSE;
            StopClicked:=TRUE;
            EXIT;
        END_IF
        IF M[i].Position = GVL_Config.Station
THEN
            MoveAxes[i] (axis:=Axes[i],
execute:=FALSE);
            MoveAxes[i] (axis:=Axes[i],
execute:=TRUE, position:=GVL_Config.Station + 50,
velocity:=GVL_Config.MoversVelocity,
gap:=GVL_Config.MoversGap,Direction:=1);
            MoveXTS := FALSE;
            Notifications :=
fbMessage.CreateEx(TC_EVENTS.Move_Movers.Move_ON,0);
            Notifications := fbMessage.Send(0);
            EXIT;
        END_IF

        END_FOR
    END_IF
    3: MoveAxes[index] (axis:=Axes[index], execute:=FALSE);
        MoveAxes[index] (axis:=Axes[index], execute:=TRUE,
position:=GVL_Config.Station, velocity:=GVL_Config.MoversVelocity,
gap:=GVL_Config.MoversGap,Direction:=1);
        Station2State :=1;

```

```

4: MoveAxes[index] (axis:=Axes[index], execute:=FALSE);
      MoveAxes[index] (axis:=Axes[index], execute:=TRUE,
position:=GVL_Config.Station2, velocity:=GVL_Config.MoversVelocity,
gap:=GVL_Config.MoversGap,Direction:=1);
      Station2State :=1;
END_CASE

IF EnableDisableCheckBox THEN
  FOR i:=1 TO AxesCount DO
    IF M[i].Position = GVL_Config.Station2 THEN
      index2:=i;
      EXIT;
    END_IF
  END_FOR
  IF index2>0 THEN
    ConvertToTime :=
LREAL_TO_TIME(GVL_Config.station2_timer*1000);
    fbTonDelay2(IN := NOT fbTonDelay2.Q, PT:=
ConvertToTime);
    IF fbTonDelay2.Q THEN
      MoveAxes[index2] (axis:=Axes[index2],
execute:=FALSE);
      MoveAxes[index2] (axis:=Axes[index2],
execute:=TRUE, position:=GVL_Config.Station,
velocity:=GVL_Config.MoversVelocity,
gap:=GVL_Config.MoversGap,Direction:=1);

      END_IF
    END_IF
  END_IF
END_IF

IF StopClicked THEN
  Station2State:=0;
CASE HaltState OF

  // GroupHalt
  0: GroupHalt(AxesGroup:=GroupAxes, Execute:=FALSE);
      GroupHalt(AxesGroup:=GroupAxes, Execute:=TRUE);
      HaltState := 1;
  1: GroupHalt(AxesGroup:=GroupAxes);
      IF (NOT GroupHalt.Busy) THEN
        IF (NOT GroupHalt.Error) THEN

          HaltState := 0;
          StopClicked:=FALSE;
          Notifications :=
fbMessage.CreateEx(TC_EVENTS.GroupHalt.GroupHalt_Done,0);
          Notifications := fbMessage.Send(0);
        ELSE
          Notifications :=
fbMessage.CreateEx(TC_EVENTS.GroupHalt.GroupHalt_Error,0);
          Notifications := fbMessage.Send(0);
        END_IF
      END_IF
    END_CASE

  END_IF

END_IF

```

Roboter-Programm

```
&ACCESS RVP
&REL 375
&PARAM EDITMASK = *
&PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
DEF xts( )
INI

PTP HOME VEL= 100 % DEFAULT
WHILE X==TRUE
SWITCH M
CASE 1

PTP HOME VEL= 100 % DEFAULT
PTP P58 VEL=100 % PDAT58 TOOL[1] BASE[0]
PTP P59 VEL=100 % PDAT59 TOOL[1] BASE[0]

WAIT TIME=1 SEC
OUT 7 '' STATE=FALSE
OUT 10 '' STATE=TRUE
WAIT TIME=1 SEC
PTP P60 VEL=100 % PDAT60 TOOL[1] BASE[0]
PTP P61 VEL=100 % PDAT61 TOOL[1] BASE[0]
PTP P62 VEL=100 % PDAT62 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=TRUE
OUT 10 '' STATE=FALSE
WAIT TIME=1 SEC
PTP P63 VEL=100 % PDAT63 TOOL[1] BASE[0]
PTP P64 VEL=100 % PDAT64 TOOL[1] BASE[0]
PTP HOME VEL=100 % DEFAULT
M=0
MoveXTS = TRUE

CASE 2
PTP P65 VEL=100 % PDAT65 TOOL[1] BASE[0]
PTP P66 VEL=100 % PDAT66 TOOL[1] BASE[0]
PTP P67 VEL=100 % PDAT67 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=FALSE
OUT 10 '' STATE=TRUE
WAIT TIME=1 SEC
PTP P68 VEL=100 % PDAT68 TOOL[1] BASE[0]
PTP P69 VEL=100 % PDAT69 TOOL[1] BASE[0]
PTP P70 VEL=100 % PDAT70 TOOL[1] BASE[0]
PTP P71 VEL=100 % PDAT71 TOOL[1] BASE[0]
```

```

PTP P72 VEL=100 % PDAT72 TOOL[1] BASE[0]
PTP P73 VEL=100 % PDAT73 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=TRUE
OUT 10 '' STATE=FALSE
WAIT TIME=1 SEC
PTP P74 VEL=100 % PDAT74 TOOL[1] BASE[0]
PTP HOME VEL=100 % DEFAULT
WAIT TIME=1 SEC
M=0
MoveXTS = TRUE
CASE 3

```

```

PTP P143 VEL=100 % PDAT143 TOOL[1] BASE[0]
PTP P144 VEL=100 % PDAT144 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=FALSE
OUT 10 '' STATE=TRUE
WAIT TIME=1 SEC
PTP P145 VEL=100 % PDAT145 TOOL[1] BASE[0]
PTP P146 VEL=100 % PDAT146 TOOL[1] BASE[0]
PTP P147 VEL=100 % PDAT147 TOOL[1] BASE[0]
PTP P148 VEL=100 % PDAT148 TOOL[1] BASE[0]
PTP P149 VEL=100 % PDAT149 TOOL[1] BASE[0]
PTP P150 VEL=100 % PDAT150 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=TRUE
OUT 10 '' STATE=FALSE
WAIT TIME=1 SEC
PTP P151 VEL=100 % PDAT151 TOOL[1] BASE[0]
PTP P152 VEL=100 % PDAT152 TOOL[1] BASE[0]
PTP HOME VEL=100 % DEFAULT
M=0
MoveXTS = TRUE
CASE 4

```

```

PTP P134 VEL=100 % PDAT134 TOOL[1] BASE[0]
PTP P135 VEL=100 % PDAT135 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=FALSE
OUT 10 '' STATE=TRUE
WAIT TIME=1 SEC
PTP P136 VEL=100 % PDAT136 TOOL[1] BASE[0]
PTP P137 VEL=100 % PDAT137 TOOL[1] BASE[0]

```

```

PTP P138 VEL=100 % PDAT138 TOOL[1] BASE[0]
PTP P139 VEL=100 % PDAT139 TOOL[1] BASE[0]
PTP P140 VEL=100 % PDAT140 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=TRUE
OUT 10 '' STATE=FALSE
WAIT TIME=1 SEC
PTP P141 VEL=100 % PDAT141 TOOL[1] BASE[0]
PTP P142 VEL=100 % PDAT142 TOOL[1] BASE[0]
PTP HOME VEL=100 % DEFAULT
WAIT TIME=1 SEC
M=0
MoveXTS = TRUE
CASE 5

PTP P125 VEL=100 % PDAT125 TOOL[1] BASE[0]
PTP P126 VEL=100 % PDAT126 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=FALSE
OUT 10 '' STATE=TRUE
WAIT TIME=1 SEC
PTP P127 VEL=100 % PDAT127 TOOL[1] BASE[0]
PTP P128 VEL=100 % PDAT128 TOOL[1] BASE[0]
PTP P129 VEL=100 % PDAT129 TOOL[1] BASE[0]
PTP P130 VEL=100 % PDAT130 TOOL[1] BASE[0]
PTP P131 VEL=100 % PDAT131 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=TRUE
OUT 10 '' STATE=FALSE
WAIT TIME=1 SEC
PTP P132 VEL=100 % PDAT132 TOOL[1] BASE[0]
PTP P133 VEL=100 % PDAT133 TOOL[1] BASE[0]
PTP HOME VEL=100 % DEFAULT
M=0
MoveXTS = TRUE
CASE 6

PTP P115 VEL=100 % PDAT115 TOOL[1] BASE[0]
PTP P116 VEL=100 % PDAT116 TOOL[1] BASE[0]
PTP P117 VEL=100 % PDAT117 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=FALSE
OUT 10 '' STATE=TRUE
WAIT TIME=1 SEC

```

```

PTP P118 VEL=100 % PDAT118 TOOL[1] BASE[0]
PTP P119 VEL=100 % PDAT119 TOOL[1] BASE[0]
PTP P120 VEL=100 % PDAT120 TOOL[1] BASE[0]
PTP P121 VEL=100 % PDAT121 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=TRUE
OUT 10 '' STATE=FALSE
WAIT TIME=1 SEC
PTP P123 VEL=100 % PDAT123 TOOL[1] BASE[0]
PTP P124 VEL=100 % PDAT124 TOOL[1] BASE[0]
PTP HOME VEL=100 % DEFAULT
WAIT TIME=1 SEC
M=0
MoveXTS = TRUE
CASE 7

PTP P96 VEL=100 % PDAT96 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=FALSE
OUT 10 '' STATE=TRUE

WAIT TIME=1 SEC
PTP P97 VEL=100 % PDAT97 TOOL[1] BASE[0]
PTP P98 VEL=100 % PDAT98 TOOL[1] BASE[0]
PTP P99 VEL=100 % PDAT99 TOOL[1] BASE[0]
PTP P100 VEL=100 % PDAT100 TOOL[1] BASE[0]
PTP P101 VEL=100 % PDAT101 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=TRUE
OUT 10 '' STATE=FALSE
WAIT TIME=1 SEC
PTP P102 VEL=100 % PDAT102 TOOL[1] BASE[0]
PTP P103 VEL=100 % PDAT103 TOOL[1] BASE[0]
PTP HOME VEL=100 % DEFAULT
M=0
MoveXTS = TRUE
CASE 8

PTP P104 VEL=100 % PDAT104 TOOL[1] BASE[0]
PTP P105 VEL=100 % PDAT105 TOOL[1] BASE[0]
PTP P106 VEL=100 % PDAT106 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=FALSE
OUT 10 '' STATE=TRUE
WAIT TIME=1 SEC
PTP P107 VEL=100 % PDAT107 TOOL[1] BASE[0]

```

```

PTP P108 VEL=100 % PDAT108 TOOL[1] BASE[0]
PTP P109 VEL=100 % PDAT109 TOOL[1] BASE[0]
PTP P110 VEL=100 % PDAT110 TOOL[1] BASE[0]
PTP P111 VEL=100 % PDAT111 TOOL[1] BASE[0]
PTP P112 VEL=100 % PDAT112 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=TRUE
OUT 10 '' STATE=FALSE
WAIT TIME=1 SEC
PTP P113 VEL=100 % PDAT113 TOOL[1] BASE[0]
PTP P114 VEL=100 % PDAT114 TOOL[1] BASE[0]
PTP HOME VEL=100 % DEFAULT
WAIT TIME=1 SEC
M=0
MoveXTS = TRUE
CASE 9

PTP P75 VEL=100 % PDAT75 TOOL[1] BASE[0]
PTP P76 VEL=100 % PDAT76 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=FALSE
OUT 10 '' STATE=TRUE
WAIT TIME=1 SEC
PTP P77 VEL=100 % PDAT77 TOOL[1] BASE[0]
PTP P78 VEL=100 % PDAT78 TOOL[1] BASE[0]
PTP P79 VEL=100 % PDAT79 TOOL[1] BASE[0]
PTP P80 VEL=100 % PDAT80 TOOL[1] BASE[0]
PTP P81 VEL=100 % PDAT81 TOOL[1] BASE[0]
PTP P82 VEL=100 % PDAT82 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=TRUE
OUT 10 '' STATE=FALSE
WAIT TIME=1 SEC
PTP P83 VEL=100 % PDAT83 TOOL[1] BASE[0]
PTP P84 VEL=100 % PDAT84 TOOL[1] BASE[0]
PTP HOME VEL=100 % DEFAULT
M=0
MoveXTS = TRUE
CASE 10

PTP P85 VEL=100 % PDAT85 TOOL[1] BASE[0]
PTP P86 VEL=100 % PDAT86 TOOL[1] BASE[0]
PTP P87 VEL=100 % PDAT87 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=FALSE
OUT 10 '' STATE=TRUE

```

```

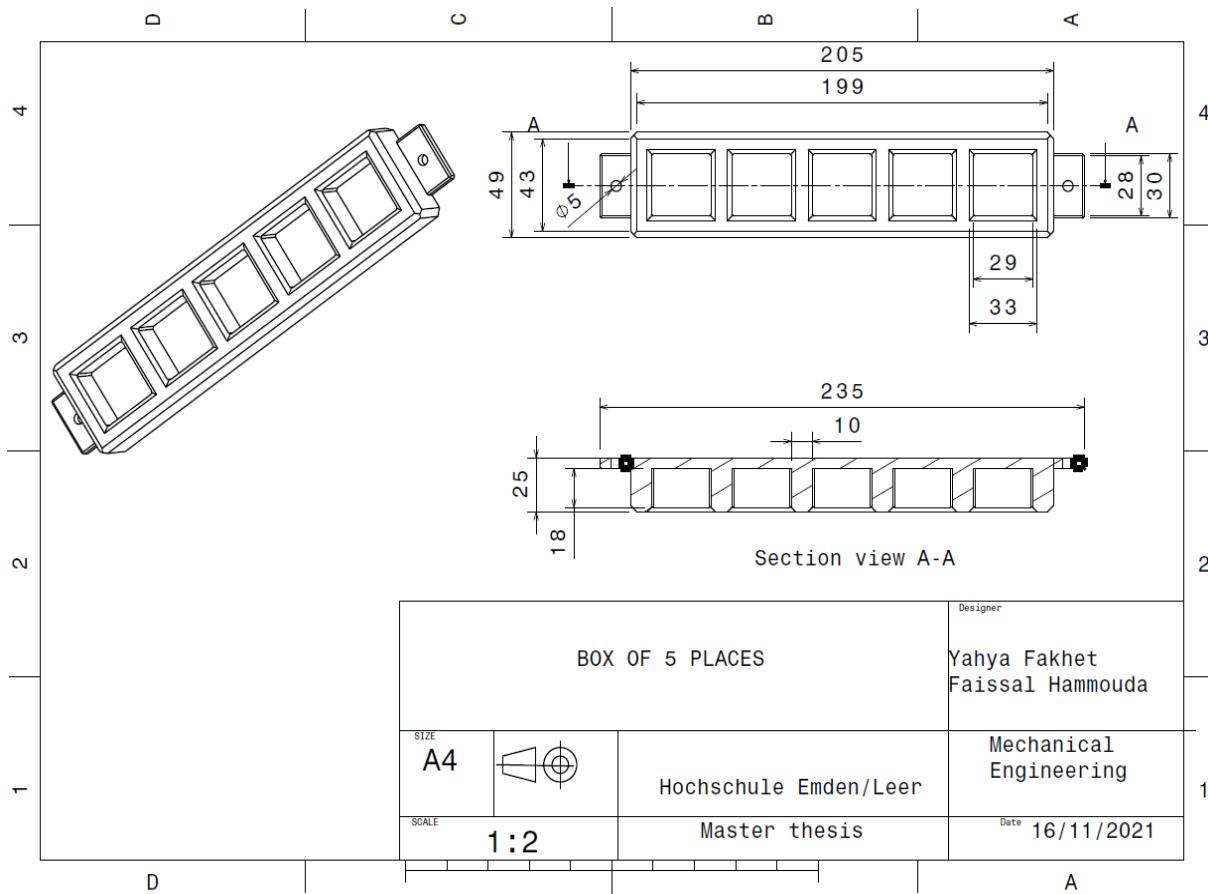
WAIT TIME=1 SEC
PTP P88 VEL=100 % PDAT88 TOOL[1] BASE[0]
PTP P89 VEL=100 % PDAT89 TOOL[1] BASE[0]
PTP P90 VEL=100 % PDAT90 TOOL[1] BASE[0]
PTP P91 VEL=100 % PDAT91 TOOL[1] BASE[0]
PTP P92 VEL=100 % PDAT92 TOOL[1] BASE[0]
WAIT TIME=1 SEC
OUT 7 '' STATE=TRUE
OUT 10 '' STATE=FALSE
WAIT TIME=1 SEC
PTP P93 VEL=100 % PDAT93 TOOL[1] BASE[0]
PTP P94 VEL=100 % PDAT94 TOOL[1] BASE[0]
PTP HOME VEL=100 % DEFAULT
WAIT TIME=1 SEC
M=0
MoveXTS = TRUE
DEFAULT
PTP HOME VEL= 100 % DEFAULT
ENDSWITCH
:
ENDWHILE
-END
:

```

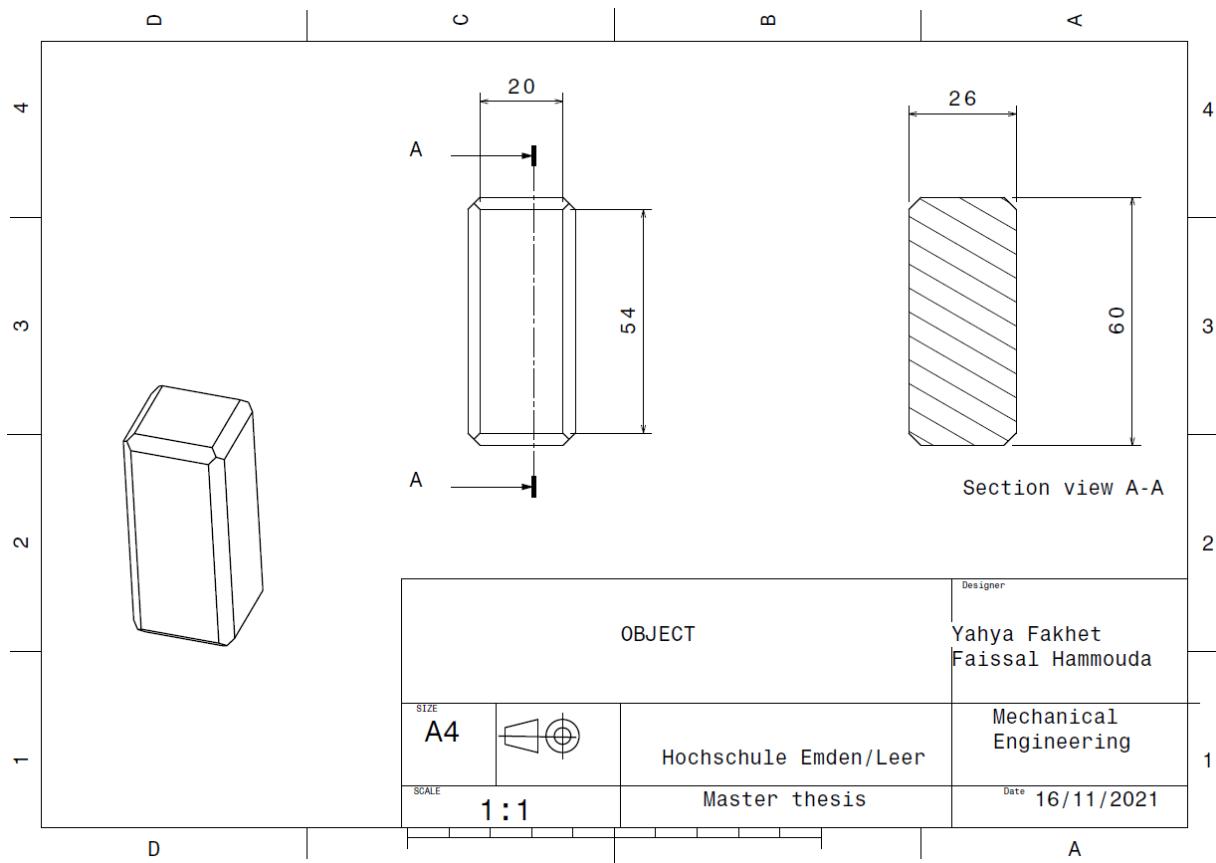
Applikation-Python-Programm

```
from kukavarproxy import *
from opcua import Client,ua
url = "opc.tcp://169.254.146.109:4840"
client_xts = Client(url)
client_xts.connect()
XtsConnected = True
KUKA_Velocity = client_xts.get_node("ns=4;s=GVL_Config.KUKA_Velocity")
Connect = client_xts.get_node("ns=4;s=GVL_Config.Connect")
Connected = client_xts.get_node("ns=4;s=GVL_Config.Connected")
MoveKuka = client_xts.get_node("ns=4;s=GVL.MoveKuka")
MoveXts = client_xts.get_node("ns=4;s=GVL.MoveXTS")
State = client_xts.get_node("ns=4;s=GVL.State")
index = client_xts.get_node("ns=4;s=GVL.index")
GroupEnabled = client_xts.get_node("ns=4;s=GVL_Config.GroupEnabled")
KukaIP = client_xts.get_node("ns=4;s=GVL_Config.KUKA_IP_Adress")
GroupEnabled.set_value(ua.DataValue(ua.Variant(False, ua.VariantType.Boolean)))
Connect.set_value(ua.DataValue(ua.Variant(False, ua.VariantType.Boolean)))
Connected.set_value(ua.DataValue(ua.Variant(False, ua.VariantType.Boolean)))
kuka_client = None
while XtsConnected:
    if Connect.get_value():
        while Connected.get_value() == False:
            if KukaIP.get_value() != '':
                kuka_client = KUKA(KukaIP.get_value())
                if kuka_client.getErrorID() == 0:
                    kuka_client.write('$OV_PRO', KUKA_Velocity.get_value())
                    Connected.set_value(ua.DataValue(ua.Variant(True,
ua.VariantType.Boolean)))
                    Connect.set_value(ua.DataValue(ua.Variant(False,
ua.VariantType.Boolean)))
                while Connected.get_value():
                    if Connect.get_value():
                        kuka_client.disconnect()
                        Connected.set_value(ua.DataValue(ua.Variant(False,
ua.VariantType.Boolean)))
                        Connect.set_value(ua.DataValue(ua.Variant(False,
ua.VariantType.Boolean)))
                    if MoveKuka.get_value():
                        MoveKuka.set_value(ua.DataValue(ua.Variant(False,
ua.VariantType.Boolean)))
                        if State.get_value() == '1':
                            POS = index.get_value() * 2
                        else:
                            POS = index.get_value() * 2 - 1
                        kuka_client.write('M', POS)
                        print(POS)
                    if kuka_client.read('MoveXTS') == 'TRUE' :
                        MoveXts.set_value(ua.DataValue(ua.Variant(True,
ua.VariantType.Boolean)))
                        kuka_client.write('MoveXTS', False)
```

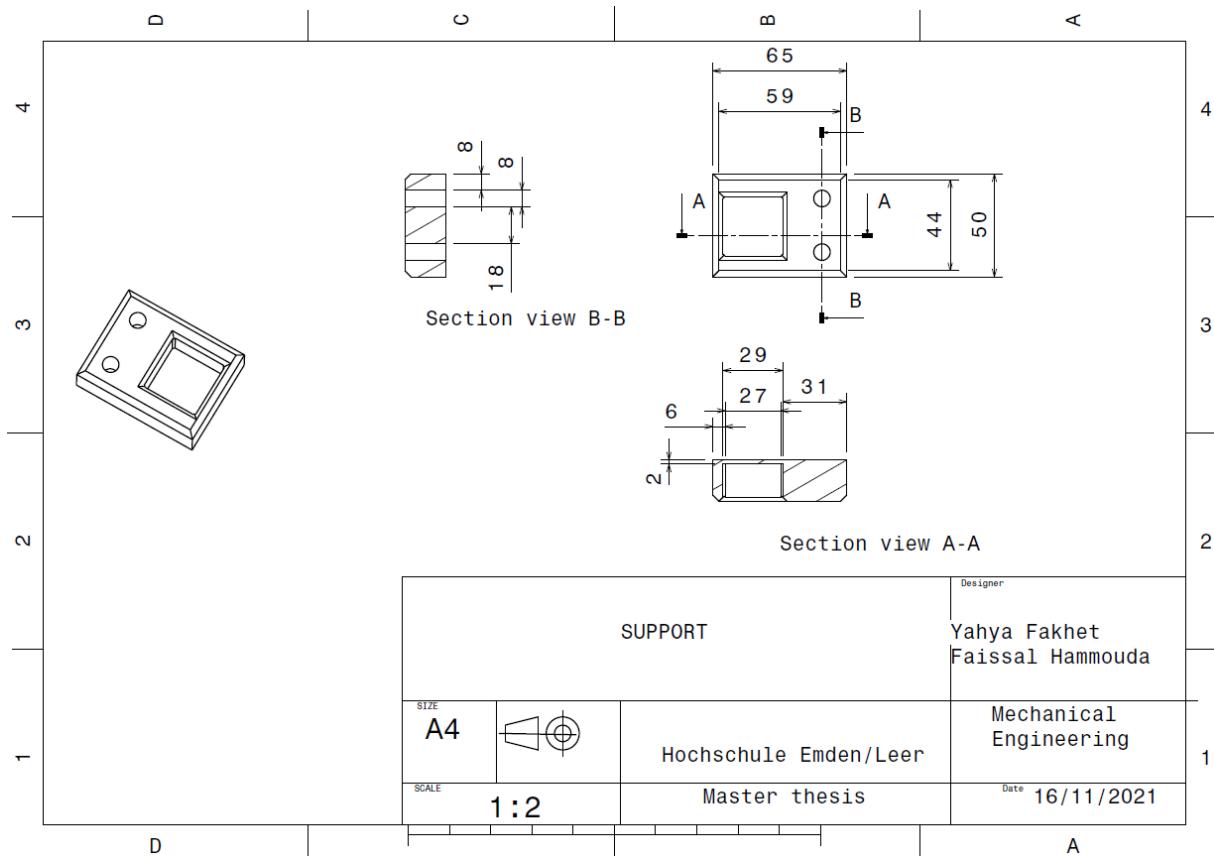
Technische Zeichnung: Box (CATIA V5)



Technische Zeichnung: Objekt (CATIA V5)



Technische Zeichnung: Support(CATIA V5)



9. Literatur

- [1] Optimum datamanagement solutions GmbH: Willkommen in der Welt der Digitalisierung. Link: https://www.optimumgmbh.de/?gclid=Cj0KCQjwqKuKBhCxARIsACf4XuF2yRmNgg0pxP8gd9H3hCXO7woa-SF06Wia9sQmsbz0Ys76UU_5IzoaAjhiEALw_wcB&fbclid=IwAR0Fh35pCpi9EgLrXMe3M_C7Am4dedWRksXS_Sm8miNgfANICtvybqnFQ, Zugriff am 5/10/2021
- [2] OPC Foundation: Home Page - OPC Foundation. Link: <https://opcfoundation.org/>, Zugriff am 5/10/2021.
- [3] Hendrik Boomgaarden (2019): Entwurf und Implementierung einer Schnittstelle zwischen FPGA und Cyber-physischen System auf Basis von VHDL, Python und OPC-UA, S. 1–2, Zugriff am 5/10/2021.
- [4] Mario Hermann; Tobias Pentek; Boris Otto (Eds.) (2016): Design Principles for Industrie 4.0 Scenarios. 2016 49th Hawaii International Conference on System Sciences (HICSS). Koloa, HI, USA, 1/5/2016 - 1/8/2016: IEEE, S.3928, Zugriff am 6/10/2021.
- [5] Mario Hermann; Tobias Pentek; Boris Otto (Eds.) (2016): Design Principles for Industrie 4.0 Scenarios. 2016 49th Hawaii International Conference on System Sciences (HICSS). Koloa, HI, USA, 1/5/2016 - 1/8/2016: IEEE, S.3929, Zugriff am 6/10/2021.
- [6] Mario Hermann; Tobias Pentek; Boris Otto (Eds.) (2016): Design Principles for Industrie 4.0 Scenarios. 2016 49th Hawaii International Conference on System Sciences (HICSS). Koloa, HI, USA, 1/5/2016 - 1/8/2016: IEEE, S.3929, Zugriff am 6/10/2021.
- [7] Mario Hermann; Tobias Pentek; Boris Otto (Eds.) (2016): Design Principles for Industrie 4.0 Scenarios. 2016 49th Hawaii International Conference on System Sciences (HICSS). Koloa, HI, USA, 1/5/2016 - 1/8/2016: IEEE, S.3929, Zugriff am 6/10/2021.
- [8] Drath, Rainer; Horch, Alexander (2014): Industrie 4.0: Hit or Hype? In *IEEE Ind. Electron. Mag.* 8 (2). DOI: 10.1109/MIE.2014.2312079, S.3, Zugriff am 8/10/2021.
- [9] Drath, Rainer; Horch, Alexander (2014): Industrie 4.0: Hit or Hype? In *IEEE Ind. Electron. Mag.* 8 (2). DOI: 10.1109/MIE.2014.2312079, S.2, Zugriff am 8/10/2021.
- [10] Hennig, Kagermann; Wolfgang, Wahlster; Johannes, Helbig (2013): Recommendations for implementing the strategic initiative INDUSTRIE 4.0, S.38-62, Zugriff am 9/10/2021.
- [11] Øvern, Aksel (2018): Industry 4.0 - Digital Twins and OPC UA, Zugriff am 9/10/2021.
- [12] OPC Foundation (2017): What is OPC? - OPC Foundation. Link: <https://opcfoundation.org/about/what-is-opc/>, updated on 6/15/2017, Zugriff am 10/10/2021.
- [13] OPC Foundation (2020): Classic - OPC Foundation. Link: <https://opcfoundation.org/about/opc-technologies/opc-classic/>, updated on 10/23/2020, Zugriff am 10/10/2021.
- [14] OPC Foundation (2019): Unified Architecture - OPC Foundation. Link: <https://opcfoundation.org/about/opc-technologies/opc-ua/>, updated on 9/26/2019, Zugriff am 11/10/2021.
- [15] Schreier, Jürgen (2019): Was ist OPC UA? Definition, Architektur und Anwendung. In *Industry of Things*, 2/6/2019. Link: <https://www.industry-of-things.de/was-ist-opc-ua-definition-architektur-und-anwendung-a-727188/>, Zugriff am 11/10/2021
- [16] C++ UA Server SDK: Introduction to OPC UA (2015). Link: <https://documentation.unified-automation.com/uasdkcpp/1.5.0/html/L2OpcUaOverview.html>, updated on 11/2/2015, Zugriff am 12/10/2021.
- [17] C ++ UA Server SDK: OPC UA Specifications (2015). Link: <https://documentation.unified-automation.com/uasdkcpp/1.5.0/html/L2OpcUaSpecifications.html>, updated on 11/2/2015, Zugriff am 12/10/2021.
- [18] Pauker, Florian; Frühwirth, Thomas; Kittl, Burkhard; Kastner, Wolfgang (2016): A Systematic Approach to OPC UA Information Model Design. In *Procedia CIRP* 57, S.321-326 DOI: 10.1016/j.procir.2016.11.056, Zugriff am 14/10/2021.
- [19] OPC Foundation (2021): OPC UA in the Reference Architecture Model RAMI 4.0 – OPC Connect. Link: <https://opcconnect.opcfoundation.org/2015/06/opc-ua-in-the-reference-architecture-model-rami-4-0/>, updated on 11/29/2021, Zugriff am 14/10/2021.
- [20] Wagner, Johanna: RAMI4.0 - 2018 – DE, S.27, Zugriff am 14/10/2021.

- [21] Stefan-Helmut Leitner; Wolfgang Mahnke: OPC UA – Service-oriented Architecture for Industrial Applications, S.1-6, Zugriff am 14/10/2021.
- [22] Python Institut: About Python | Python Institute. Link: <https://pythoninstitute.org/what-is-python/>, Zugriff am 16/10/2021
- [23] Human-Machine Interface (HMI) - Glossary | CSRC (2021). Link: https://csrc.nist.gov/glossary/term/human_machine_interface, Zugriff am 16/10/2021.
- [24] KUKA Roboter GmbH (2015): KUKA System Software 8.3, S17-18, Zugriff am 16/10/2021.
- [25] Imts (2018): We are open-sourcing KUKAVARPROXY - Imts. Link: <https://www.imts.eu/open-sourcing-kukavarproxy/>, Zugriff am 17/10/2021.
- [26] Lars Ivar Hatledal (2017): GitHub - aauc-mechlab/JOpenShowVar: Allows reading and writing variables to KUKA robots using a TCP/IP connection. Link: <https://github.com/aauc-mechlab/JOpenShowVar>, Zugriff am 17/10/2021.
- [27] KUKA Roboter GmbH (2015): KR AGILUS sixx, S.17-19, Zugriff am 17/10/2021.
- [28] KUKA Roboter GmbH (2015): KUKA System Software 8.3, S. 27–28, Zugriff am 17/10/2021.
- [29] Xplore-DNA (2018): Grundlagen der Robotik: Programmierung. Link: <https://www.xplore-dna.net/mod/page/view.php?id=451>, Zugriff am 18/10/2021.
- [30] Beckhoff Automation GmbH (2016): Beckhoff Information System - German. Link: <https://infosys.beckhoff.com/index.htm>, Zugriff am 18/10/2021.
- [31] net, www pc-control (2012): XTS – eXtended Transport System: Drive Technology – rethought, https://www.pc-control.net/pdf/022012/products/pcc_0212_cover_e.pdf, Zugriff am 10/18/2021.
- [32] Beckhoff Automation GmbH: C6930 | Schaltschrank-Industrie-PC. Link: <https://www.beckhoff.com/de-de/produkte/ipc/pcs/c69xx-kompakte-industrie-pcs/c6930.html>, Zugriff am 18/10/2021.
- [33] Beckhoff Automation GmbH: TwinCAT 3 | Automatisierungsssoftware. Link: https://www.beckhoff.com/de-de/produkte/automation/twincat/#text_bild_2, Zugriff am 10/18/2021.
- [34] Ahmad Saeed (2017): KUKAVARPROXY and KUKAVARPROXY Message Format. Link: <https://github.com/akselov/kukavarproxy-msg-format>, Zugriff am 19/10/2021.
- [35] SourceForge.net: Find out more about openshowvar. Link: <https://sourceforge.net/projects/openshowvar/postdownload>, Zugriff am 19/10/2021.
- [36] Beckhoff Automation GmbH: TF6100 | TwinCAT 3 OPC UA. Link: <https://www.beckhoff.com/en-en/products/automation/twincat/tfxxxx-twincat-3-functions/tf6xxx-tc3-connectivity/tf6100.html>, Zugriff am 20/10/2021.
- [37] Oroulet (2016): FreeOpcUa/python-opcua. Link: <https://github.com/FreeOpcUa/python-opcua>, Zugriff am 20/10/2021.
- [38] Beckhoff Automation GmbH (2021): Manual TwinCAT 3 | OPC UA, S.19, Zugriff am 20/10/2021.
- [39] Beckhoff Automation GmbH (2021): Manual TwinCAT 3 | OPC UA, S.38, Zugriff am 20/10/2021.
- [40] Beckhoff Automation GmbH (2021): Manual TwinCAT 3 | OPC UA, S.22-25, Zugriff am 20/10/2021.
- [41] Beckhoff Automation GmbH (2021): Manual TwinCAT 3 | OPC UA, S.35-36, Zugriff am 20/10/2021.
- [42] Beckhoff Automation GmbH (2021): Manual TwinCAT 3 | OPC UA, S.38-39, Zugriff am 20/10/2021.
- [43] Beckhoff Automation GmbH (2021): Manual TwinCAT 3 | OPC UA, S.41, Zugriff am 20/10/2021.
- [44] Beckhoff Automation GmbH (2021): Manual TwinCAT 3 | OPC UA, S.42-43, Zugriff am 20/10/2021.
- [45] Beckhoff Automation GmbH (2021): Manual TwinCAT 3 | OPC UA, S.54-55, Zugriff am 20/10/2021.
- [46] Beckhoff Automation GmbH (2021): Manual TwinCAT 3 | OPC UA, S.55-56, Zugriff am 20/10/2021.
- [47] Beckhoff Automation GmbH: TC1200 | TwinCAT 3 PLC. Link: <https://www.beckhoff.com/en-en/products/automation/twincat/tc1xxx-twincat-3-base/tc1200.html#>, Zugriff am 22/10/2021.
- [48] Beckhoff Automation GmbH: TF5000 | TwinCAT 3 NC PTP 10 Axes. Link: <https://www.beckhoff.com/en-en/products/automation/twincat/tfxxxx-twincat-3-functions/tf5xxx-tc3-motion-control/tf5000.html>, Zugriff am 22/10/2021.

[49] Beckhoff Automation GmbH: TF5850 | TwinCAT 3 XTS Extension. Link: <https://www.beckhoff.com/en-en/products/automation/twincat/tfxxxx-twincat-3-functions/tf5xxx-tc3-motion-control/tf5850.html>, Zugriff am 23/10/2021.

[50] Beckhoff Automation GmbH: TF6421 | TwinCAT 3 XML Server. Link: <https://www.beckhoff.com/de-de/produkte/automation/twincat/tfxxxx-twincat-3-functions/tf6xxx-tc3-connectivity/tf6421.html>, Zugriff am 23/10/2021.

[51] Beckhoff Automation GmbH (2021): Manual TwinCAT 3 | PLC Library: Tc2_MC2, S.9, Zugriff am 24/10/2021.

[52] Beckhoff Automation GmbH (2021): Manual TwinCAT 3 | PLC Library: Tc2_MC2, S.10-12, Zugriff am 24/10/2021.

[53] Beckhoff Automation GmbH: Beckhoff Information System - English. Link: https://infosys.beckhoff.com/english.php?content=../content/1033/tcpclib_tc2_mc2/70049419.html&id=, Zugriff am 25/10/2021.

[54] Beckhoff Automation GmbH: Beckhoff Information System - English. Link: https://infosys.beckhoff.com/english.php?content=../content/1033/tcpclib_tc2_mc2/70050955.html&id=, Zugriff am 25/10/2021.

[55] Beckhoff Automation GmbH: Beckhoff Information System - English. Link: https://infosys.beckhoff.com/english.php?content=../content/1033/tcpclib_tc2_mc2/70132363.html&id=, Zugriff am 26/10/2021.

[56] Beckhoff Automation GmbH & Co. KG (2021): Handbuch TwinCAT 3 | PLC-Bibliothek: Tc2_Standard, S.7, Zugriff am 26/10/2021.

[57] Beckhoff Automation GmbH & Co. KG (2021): Handbuch TwinCAT 3 | PLC Lib: Tc2_System, S.9-12, Zugriff am 27/10/2021.

[58] Beckhoff Automation GmbH: Beckhoff Information System - English. Link: https://infosys.beckhoff.com/english.php?content=../content/1033/tcpclibsystem/html/tcpclibsys_t_maxstring.htm&id=, Zugriff am 27/10/2021.

[59] Beckhoff Automation GmbH & Co. KG (2020): Handbuch TwinCAT 3 | XML Server, S.16, Zugriff am 27/10/2021.

[60] Beckhoff Automation GmbH: Beckhoff Information System - English. Link: https://infosys.beckhoff.com/english.php?content=../content/1033/tcxmldatasrv/html/tcxmldatasrv_fb_xmlsrvread.htm&id=3784880098735847093, Zugriff am 28/10/2021.

[61] Beckhoff Automation GmbH: Beckhoff Information System - English. Link: https://infosys.beckhoff.com/index.php?content=../content/1031/tf6421_tc3_xml_server/187123979.html&id=, Zugriff am 28/10/2021.

[62] Beckhoff Automation GmbH & Co. KG: Handbuch TwinCAT 3 | PLC-Bibliothek Tc3_EventLogger, S. 9, Zugriff am 28/10/2021.

[63] Beckhoff Automation GmbH: Beckhoff Information System - English. Link: https://infosys.beckhoff.com/english.php?content=../content/1033/tcpclib_tc3_eventlogger/5002818315.html&id=, Zugriff am 28/10/2021.

[64] Beckhoff Automation GmbH: Beckhoff Information System - English. Link: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_eventlogger/5003041163.html&id=, Zugriff am 28/10/2021.

[65] Beckhoff Automation GmbH: Beckhoff Information System - English. Link: https://infosys.beckhoff.com/english.php?content=../content/1033/tcpclib_tc3_eventlogger/9956769291.html&id=, Zugriff am 28/10/2021.

[66] Beckhoff Automation GmbH: Beckhoff Information System - English. Link: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_eventlogger/5003786635.html&id=, Zugriff am 28/10/2021.

[67] Beckhoff Automation GmbH & Co. KG (2021a): Handbuch TwinCAT 3 | Motion Collision Avoidance, S.7, Zugriff am 29/10/2021.

- [68] Beckhoff Automation GmbH: Beckhoff Information System - English. Link:
https://infosys.beckhoff.com/english.php?content=../content/1033/tf5410_tc3_collision_avoidance/8893152395.html&id=, Zugriff am 29/10/2021.
- [69] Beckhoff Automation GmbH: Beckhoff Information System - English. Link:
https://infosys.beckhoff.com/english.php?content=../content/1033/tf5410_tc3_collision_avoidance/8893322251.html&id=, Zugriff am 29/10/2021.
- [70] Beckhoff Automation GmbH: Beckhoff Information System - English. Link:
https://infosys.beckhoff.com/english.php?content=../content/1033/tf5410_tc3_collision_avoidance/8893296011.html&id=, Zugriff am 29/10/2021.
- [71] Beckhoff Automation GmbH: Beckhoff Information System - English. Link:
https://infosys.beckhoff.com/english.php?content=../content/1033/tf5410_tc3_collision_avoidance/1533223819.html&id=, Zugriff am 29/10/2021.
- [72] Beckhoff Automation GmbH: Beckhoff Information System - English. Link:
https://infosys.beckhoff.com/english.php?content=../content/1033/tf5410_tc3_collision_avoidance/8893110923.html&id=, Zugriff am 29/10/2021.
- [73] Beckhoff Automation GmbH: Beckhoff Information System - English. Link:
https://infosys.beckhoff.com/english.php?content=../content/1033/tf5410_tc3_collision_avoidance/8893096843.html&id=, Zugriff am 30/10/2021.
- [74] Beckhoff Automation GmbH: Beckhoff Information System - English. Link:
https://infosys.beckhoff.com/english.php?content=../content/1033/tf5130_tc3_unival_plc/9821284107.html&id=, Zugriff am 30/10/2021.
- [75] Beckhoff Automation GmbH: Beckhoff Information System - English. Link:
https://infosys.beckhoff.com/english.php?content=../content/1033/tf5410_tc3_collision_avoidance/8893138315.html&id=, Zugriff am 30/10/2021.
- [76] Beckhoff Automation GmbH: Beckhoff Information System - English. Link:
https://infosys.beckhoff.com/english.php?content=../content/1033/tf5410_tc3_collision_avoidance/8893124235.html&id=, Zugriff am 30/10/2021.
- [77] Beckhoff Automation GmbH: Beckhoff Information System - English. Link:
https://infosys.beckhoff.com/english.php?content=../content/1033/tf5410_tc3_collision_avoidance/8893068299.html&id=, Zugriff am 30/10/2021.
- [78] Beckhoff Automation GmbH: Beckhoff Information System - English. Link:
https://infosys.beckhoff.com/english.php?content=../content/1033/tf5410_tc3_collision_avoidance/8892089867.html&id=, Zugriff am 30/10/2021.
- [79] Beckhoff Automation GmbH & Co. KG: Handbuch TwinCAT 3 | PLC-Bibliothek: Tc2_Math, S.7, Zugriff am 30/10/2021.
- [80] Beckhoff Automation GmbH: XTS Utility, Zugriff am 30/10/2021.
- [81] Beckhoff Automation GmbH: TE2000 | TwinCAT 3 HMI Engineering. Link:
<https://www.beckhoff.com/de-de/produkte/automation/twincat/texxxx-twincat-3-engineering/te2000.html>, Zugriff am 30/10/2021.
- [82] LynnClaessen (2020): OPC in old PLC systems. In *PLC Support India*, 5/19/2020. Link:
<https://plcsupport.in/opc-in-old-plc-systems/>, Zugriff am 6/10/2021.
- [83] OPC Foundation (2020): SecurityLayers.jpg - UaCapabilities, Zugriff am 8/10/2021.
- [84] Marketing; KUKA Deutschland GmbH (2021): KR 6 R900 sixx, Zugriff am 8/10/2021.
- [85] Beckhoff Automation GmbH & Co. KG: Handbuch TwinCAT 3 | TE2000 TC3 HMI Engineering, S.15, Zugriff am 30/10/2021.

[86] Armando W. Colombo (2021) Skript Modul “Robotics Systems”, Master Industrial Informatics, Hochschule Emden/Leer, Deutschland.

[87] Armando W. Colombo (2021) Skript Modul “Industrial Cyber-Physical Systems”, Master Industrial Informatics, Hochschule Emden/Leer, Deutschland.

[88] Armando W. Colombo (2021) Skript Modul “Digitalization of ICPS“, Master Industrial Informatics, Hochschule Emden/Leer, Deutschland.

[89] Armando W. Colombo (2021) Skript Modul “Industrial Data Transport Technology “, Master Industrial Informatics, Hochschule Emden/Leer, Deutschland.

[90] KUKA Roboter GmbH (2015): KUKA.RecoveryUSB 3.0, Zugriff am 17/10/2021.