Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Marcel Schäfer

Erstellung und Evaluation eines digitalen Zwillings für die Offline-Programmierung einer Palettier-Anlage mithilfe von Visual Components und Codesys

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Engineering and Computer Science
Department of Automotive and Aeronautical Engineering

Marcel Schäfer

Erstellung und Evaluation eines digitalen Zwillings für die Offline-Programmierung einer Palettier-Anlage mithilfe von Visual Components und Codesys

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Mechatronik
am Department Fahrzeugtechnik und Flugzeugbau
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

in Zusammenarbeit mit:
Lorenscheit Automatisierungs-Technik GmbH
Abteilung Robotik
Marienauer Weg 7
21368 Dahlenburg

Erstprüfer/in: Prof. Dr. Thomas Frischgesell
Zweitprüfer/in: Dipl.-Wirtsch.-Ing. (FH) Thomas Lorenscheit

Zusammenfassung

Marcel Schäfer

Thema der Bachelorthesis
Erstellung und Evaluation eines digitalen Zwillings für die Offline-Programmierung einer Palettier-Anlage mithilfe von Visual Components und Codesys

Stichworte
Digitaler Zwilling, Visual Components, Palettierung, Codesys, Universal Robots, OPC UA, Raspberry Pi, Automatisierungstechnik, Simulation

Kurzzusammenfassung
In dieser Arbeit steht die Erstellung eines digitalen Zwillings mit Visual Components im Fokus. Der digitale Zwilling bildet einen Flex Cobot Palletizer (FCP) der Firma Lorenscheit-Automatisierungs-Technik GmbH ab. Ziel soll es sein, das Testen von Programmcode für Robotersteuerung, HMI und SPS zu ermöglichen. Dabei gilt es, einen realitätsnahen digitalen Zwilling zu erstellen, um anschließend dessen Möglichkeiten zu evaluieren und um eine möglichst genaue Abbildung der realen Maschine umsetzen zu können. Der digitale Zwilling sorgt dafür, den Code frühzeitig testen und mögliche Fehler vermeiden zu können. So können neue Funktionen und die damit verbundenen Abläufe geprüft werden. Um das Roboterprogramm ausführen zu können, wird ein realer UR-10-Roboter verwendet. Für das SPS-Programm kommt eine Soft-SPS mit Codesys zum Einsatz. Die HMI läuft auf einem Raspberry Pi über Java. Die Ergebnisse dieser Arbeit dienen der Bewertung eines digitalen Zwillings im Vergleich zur Programmierung an der realen Anlage.

Abstract

Marcel Schäfer

Title of the paper

Creation and evaluation of a digital twin for the offline programming of a palletizing system using Visual Components and Codesys

Keywords

Digital twin, Visual Components, palletizing, Codesys, Universal Robots, OPC UA, Raspberry Pi, automation technology, simulation

Abstract

This thesis focuses on the creation of a digital twin with Visual Components. The digital twin represents a Flex Cobot Palletizer (FCP) from the company Lorenscheit-Automatisierungs-Technik GmbH. The intention is to enable the testing of code for robot control, HMI and PLC, with the objective of creating a realistic digital twin. This enables the subsequent evaluation of its capabillities and the implementation of a representation that closely mirrors the real machine. The digital twin ensures that code can be tested at an early stage and possible errors can be avoided. This allows new functions and the associated processes to be tested. A real UR-10 robot is used to execute the robot program. A soft PLC with Codesys is used for the PLC program. The HMI runs on a Raspberry Pi via Java. The results of this work are used to evaluate a digital twin in comparison to programming on the real system.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Abkürzung	Bedeutung
AS	Ablaufsprache
AWL	Anweisungsliste
CAD	computer aided design
CFC	continuous function chart
codesys	controller development system
FBD	Function Block Diagramm
FBS	Funktionsbausteinsprache
FCP	Flex Cobot Palletizer
ggf.	Gegebenenfalls
GmbH	Gesellschaft mit beschränkter Haftung
HMI	Human Maschine Interface
IEC	International Electrotechnical Commission
IL	Instruktion List
KOP	Kontaktplan
LAT	Lorenscheit Automatisierungs-Technik
LD	Ladder diagram
LxBxH	Länge · Breite · Höhe
mm	Millimeter
ms	Millisekunden
OLP	Offlineprogrammierung
OPC UA	Open platform communications unified architechture
PnP	Plug and Play
s	Sekunde
SFC	Sequential function
SFP	Signalflussplan
SPS	Speicherprogrammierbare Steuerung
ST	Strukturierter Text / structured text

U	Umdrehung
UR	Universal Robots
usw.	Und so weiter
V	Volt
VCMX	Visual Components Model Data
z.B.	Zum Beispiel

1 Einleitung

1.1 Motivation

Die Lorenscheit-Automatisierungs-Technik (LAT) GmbH ist ein Unternehmen mit einer breiten Vielzahl an Kunden. Entsprechend viele Anlagen dieser sind somit in der Industrie im Einsatz. Bei einer hohen Anzahl an Serien- und Sondermaschinen steigt allerdings auch die Wahrscheinlichkeit, auf Probleme zu stoßen. Diese Fehler können sich bereits bei der Konstruktion äußern oder erst Jahre später in der Produktion beim Kunden. Da diese Fehler mit zusätzlichen Kosten verbunden sind, gilt es, diese so früh wie möglich zu vermeiden. Sollte es trotzdem zu einem Fehler einer Anlage kommen, ist es wichtig, diese gezielt und schnell beheben zu können, um die Kosten möglichst gering zu halten.

Diese Arbeit und die damit verbundene Erstellung eines digitalen Zwillings, vom FCP [1] dient somit zum einen der Entwicklung und Weiterentwicklung des FCPs und zum anderen als Prototyp für andere Anlagen. Der digitale Zwilling soll ermöglichen, die Programmierung des FCP vor dem Bau dessen testen und anpassen zu können. Dies ermöglicht auch, aufgetretene Fehler in der digitalen Welt nachzustellen und offline zu beseitigen, ohne vor Ort sein zu müssen.

Ebendarum ist es wichtig die Abbildung des FCP möglichst realitätsnah in die digitale Welt umzusetzen, um sowohl eine genauere Untersuchung des Potenzials als auch der Grenzen des digitalen Zwillings zu ermöglichen, um anschließend die Anwendung dieser neuen technologischen Möglichkeit evaluieren zu können.

1.2 Firmenvorstellung – Lorenscheit Automatisierungs-Technik

Die Lorenscheit Automatisierungs-Technik GmbH ist ein Automatisierungstechnikunternehmen mit Hauptsitz in Dahlenburg [2]. Sie wurde 2010 von Dipl.-Wirtsch.-Ing. Thomas Lorenscheit und Dipl.-Wirtschjur. Nadine Lorenscheit gegründet und hat seit 2021 einen Zweitsitz in Metzingen. Angeboten wird hierbei von Serienprodukten wie das FCP bis hin zu einmaligen Sondermaschinen alles. Grundsätzlich gilt es für neue Probleme moderne Lösungen zu finden und die Automatisierungstechnik einfach verständlich, aber trotzdem effizient umzusetzen. Dafür arbeiten bei LAT mittlerweile mehr als 40 Mitarbeiter um dieses Ziel zu erreichen. Unteranderem ist LAT auch Distributor- und Integrationspartner von Universal Robots, dem Marktführer für kollaborierende Roboter.

1.3 Struktur der Arbeit

Die Arbeit kann in fünf grobe Abschnitte unterteilt werden, zu sehen in Abbildung 1.1. Im ersten Kapitel geht es um die Motivation der Erstellung dieser Arbeit, die Zielsetzung und die Zeitplanung. Das zweite Kapitel widmet sich den Grundlagen des digitalen Zwillings, Programmen und Komponenten, um jedes von ihnen zunächst zu begreifen, bevor die Erstellung des digitalen Zwillings folgt. Im dritten Kapitel wird die eigentliche Erstellung des

digitalen Zwillings unter Außerachtlassung der Kommunikation zur SPS und zum Roboter und der Schwierigkeiten, die bei der Erstellung aufgetreten sind. Im vierten Teil der Arbeit werden der reale Roboter und die SPS mit dem digitalen Zwilling verbunden. Folglich wird betrachtet, wodurch sich die Kommunikation dieser auszeichnen soll und wie dies erreicht, wird. Das fünfte Kapitel beschäftigt sich mit dem Vergleich des Zwillings zur realen Anlage, stellt Unterschiede fest und bewertet diese bezogen auf die Relevanz für die Einsetzbarkeit des digitalen Zwillings. Im folgenden Kapitel wird da drauf eingegangen und ein Fazit gezogen. Im letzten Kapitel folgt ein Ausblick, wie sich die Entwicklung des digitalen Zwillings in Zukunft gestalten könnte und welche Möglichkeiten der Optimierung es gibt.

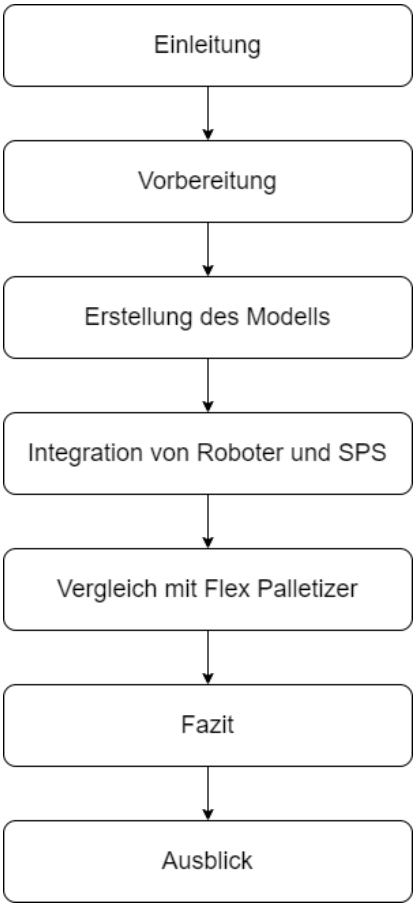


Abbildung 1.1: Struktur der Arbeit

1.4 Zielsetzung

Im Rahmen dieser Arbeit soll ein digitaler Zwilling des FCP erarbeitet werden, der die Offline-Programmierung dieser Anlage ermöglicht. Der Schwerpunkt dieser Arbeit befasst sich mit der Erstellung des digitalen Zwillings und dem wissenschaftlichen Vergleich dessen mit der realen Anlage. Durch den digitalen Zwilling soll es durch Programmieren ermöglicht werden, Änderungen am Roboter und der SPS vorzunehmen und diese anschließend vollumfänglich testen zu können, ohne eine reale Anlage zu benötigen. Um dieses Ziel zu erreichen, wird zunächst eine Anforderungsliste, zu lesen unter Punkt 2.3, aufgestellt, welche die Rahmenbedingungen klärt und Ziele definiert.

1.5 Zeitplanung

Die zeitliche Planung der Bachelorarbeit lässt sich in vier Phasen unterteilen. Die Darstellung in Abbildung 1.2 bietet eine grobe Übersicht über die Reihenfolge der Teilschritte und unterstützt bei der Erstellung der genaueren Zeitplanung.

Abbildung 1.2: Geplanter Ablauf der Bachelorarbeit

In der Vorbereitungsphase sollen die Lesenden mit dem Inhalt und der Struktur dieser Ausarbeitung vertraut gemacht werden. Zudem gibt die Motivation Auskunft über die Gründe des Erstellens dieser Ausarbeitung. Zum Schluss dieses ersten Teils wird die Zeitplanung der Arbeit festgelegt. In der nächsten Phase „Erstellung des Modells“ geht es zum einen darum, sich mit dem Programm Visual Components vertraut zu machen und zum anderen um die Zusammensetzung von CAD-Dateien zu einer fertigen Anlage. Im Anschluss erhalten Komponenten wie die Sensoren oder die Linearachse ihre Funktion. Die nachfolgende Phase widmet sich dem Verbinden des Roboters und der SPS mit den bereits existierenden Programmen der realen Anlage und der Herstellung der Kommunikation zwischen der Simulation, dem Roboter und der SPS. Anschließend steht der Vergleich zwischen dem digitalen Zwilling und der realen Anlage im Vordergrund. Hier soll festgestellt werden, welche Gemeinsamkeiten oder Unterschiede es zwischen den beiden gibt. Aus dieser Analyse können folglich die Limitierungen des digitalen Zwillings geschlussfolgert und die Anwendbarkeit für kommende Projekte bewertet werden.

1.5.1 Agile-Gantt-Diagramm

Aus dieser groben zeitlichen Planung lässt sich nun ein Agile-Gantt-Diagramm ableiten. Dieses zeigt den gesamten Zeitraum für die Erstellung dieser Arbeit an, so kann jeder Aufgabe ein Startdatum zugewiesen und eine Dauer festgelegt werden, was die Überwachung der Fortschritte sowie ein terminorientiertes Arbeiten ermöglicht. Das Agile-Gantt-Diagramm kann im Anhang unter Abschnitt A eingesehen werden.

2 Vorbereitung

In diesem Kapitel werden die grundlegenden Informationen zur Erstellung eines digitalen Zwillings erläutert. Zunächst geht es um den Stand der Technik bezogen auf den digitalen Zwilling. Im nächsten Schritt wird eine Anforderungsliste erstellt, welche definiert, was der digitale Zwilling können sollte, um ein klares Ziel dieser Arbeit definieren zu können. Abschließend werden die für diese Arbeit verwendeten Grundlagen der angewandten Programme, Bauteile und Technologien dargelegt.

2.1 Übersicht der Programmiersprachen

Der FCP besteht aus verschiedenen Komponenten, die miteinander kommunizieren, um den gesamten Prozessablauf zu ermöglichen. Um Programmcode von einem der Bauteile testen zu können benötigt es auch den Programmcode aller anderen Bauteile und die mechanische Anlage. In Abbildung 2.1 ist zu sehen welche Programmiersprachen wo verwendet werden.

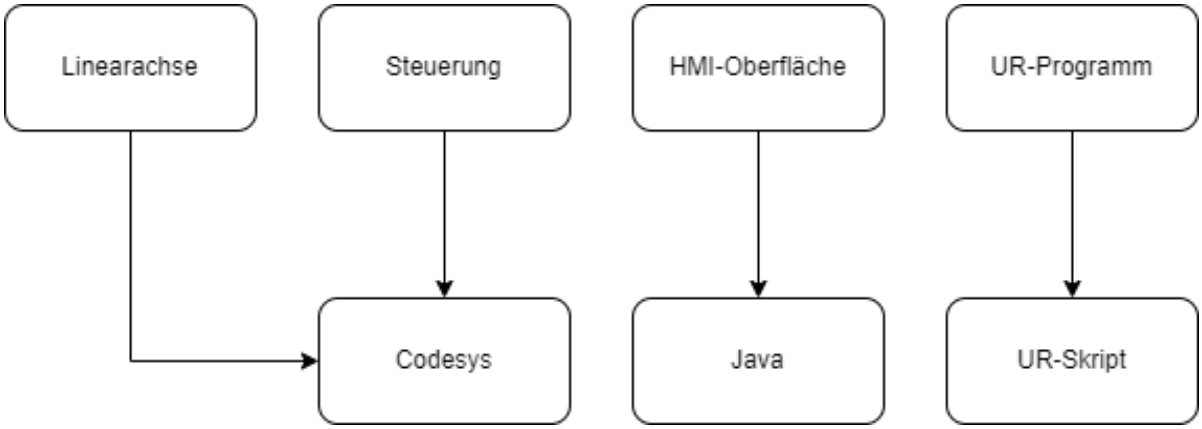


Abbildung 2.1: Programmiersprachen der Komponenten des FCP

Durch diese drei verschiedenen Programmiersprachen Codesys, Java und UR-Skript kann es durchaus vorkommen, dass drei verschiedene Programmierer nötig sind, um eine Änderung am FCP durchzuführen und dann testen zu können.

2.2 Stand der Technik

Zum Zeitpunkt dieser Arbeit ist das Thema „digitaler Zwilling“ kein neues Themengebiet mehr. In der Vergangenheit wurden digitale Zwillinge in vielen Bereichen eingesetzt, um ein bestimmtes Verhalten eines Bauteils zu simulieren. So wurden unter anderem Turbinen auf mechanische Belastung geprüft [3], das Regelverhalten von Reglern nachgeahmt oder elektrische Schaltungen getestet [4]. Solche Simulationen lassen sich als digitale Zwillinge definieren. Die Erstellung eines digitalen Zwillings einer ganzen Maschine und deren Abläufe ist hingegen eine zeitlich aktuelle Thematik [5].

Der erste digitale Zwilling wurde bereits 1960 im Rahmen der Apollo 13-Mission erstellt. Durch diesen digitalen Zwilling konnte der Startvorgang simuliert werden. Während des Fluges explodierten Sauerstofftanks der Apollo 13-Rakete. Durch den digitalen Zwilling gelang es Wissenschaftlern auf der Erde, Aussagen über den Schaden an den Triebwerken zu treffen und die Besatzung sicher zur Erde zurückzubringen [6].

Heutzutage werden digitale Zwillinge in vielseitigen Formen in vielen verschiedenen Bereichen verwendet. So werden etwa Flugzeuge simuliert, um die Sicherheit auch bei extremen äußerlichen Einflüssen gewährleisten zu können [7]. Städte geplant, um eine möglich effiziente Infrastruktur bauen zu können [8] oder in der Medizin digitale Zwillinge von Menschen erstellt, um sich auf die Gegebenheiten bei schwierigen Operationen vorbereiten zu können [9]. Aber auch in der Industrie beim Bau von Anlagen kommen vermehrt digitale Zwillinge dieser zum Einsatz.

Durch Industrie 4.0 und die weltweit stetig wachsende Industrie [10] und damit auch wachsende Nachfrage an der Automatisierung von Prozessen gibt es immer mehr Anbieter von Automatisierungslösungen [11]. Daher wird es immer wichtiger sich auf diesem Markt behaupten zu können, zum Beispiel mit kurzen Lieferzeiten, individuellen Lösungen und einer hohen Zuverlässigkeit von Maschinen. Der digitale Zwilling kann hierbei helfen, eine schnellere Produktion zu ermöglichen, da Fehler minimiert und auftretende Fehler schneller behoben werden können. Zusätzlich ermöglicht der digitale Zwilling, das Verhalten von Systemen noch in der Planungsphase zu überprüfen und ggf. anzupassen. Des Weiteren wird eine Offline-Programmierung ermöglicht, sodass dezentral an der Software für Systeme gearbeitet werden kann.

2.3 Anforderungsliste

Für eine zielführende Umsetzung des digitalen Zwillings wurden die folgenden Anforderungen an den digitalen Zwilling erarbeitet. Diese dienen als Leitfaden zur Erstellung des digitalen Zwillings. Die Ziele wurden hierbei nach eigenem Ermessen ausgewählt. Die Ziele sind so gewählt, um einen nutzbaren digitalen Zwilling zu erhalten. Die Sinnhaftigkeit und Umsetzbarkeit dieser wird im Laufe dieser Arbeit genauer erarbeitet.

- Der digitale Zwilling muss, um die Prozessabläufe darstellen und mögliche Kollisionen des Roboters feststellen zu können, mit der realen Anlage bezüglich der Mechanik übereinstimmen. Dafür müssen zum einen, die CAD-Dateien mit den realen Bauteilen und zum anderen der Zusammenbau in der Software mit dem realen Ebenbild übereinstimmen. So kann sichergestellt werden, dass die Abmaße des digitalen Zwillings mit denen der realen Anlage übereinstimmen.

- Die digitalen Sensoren und Aktuatoren müssen die Funktionsweise der realen imitieren können. Dabei gilt es sowohl die Funktionen als auch die zeitliche Komponente so nah wie möglich an die realen Bedingungen anzupassen. Hierzu zählen sowohl die Flächensicherheitsscanner und die Lichtschranke als Sicherheitseinrichtung als auch die kapazitiven Sensoren zur Palettenerkennung und Paketerkennung.

- Die Bewegungen der Linearachse und des Roboters müssen in Weg, Geschwindigkeit und Beschleunigung mit denen der realen Äquivalente übereinstimmen. Ziel hierbei ist es, die reale Position dieser zu jedem Zeitpunkt darstellen zu können. Dadurch sollen Taktzeitanalysen der Anlage anhand des digitalen Zwillings möglich werden.

- Die Programmcodes vom Roboter und der SPS sollten mit denen der realen Anlage übereinstimmen, um die Offlineprogrammierung durch Anpassungen am realen Programm zu ermöglichen. Dies ermöglicht es, den geänderten Programmcode ohne Anpassungen wieder auf der realen Anlage verwenden zu können.

- Die Kommunikation der Simulation muss mit dem Roboter, der SPS und ggf. der Anlage in Echtzeit passieren, um eine Taktzeitanalyse zu ermöglichen. Die Taktzeitabweichungen des digitalen Zwillings zur realen Anlage sollten hierbei kleiner als 5 % sein, was einer Abweichung von drei Sekunden pro simulierter Minute entspricht. Je kleiner diese Abweichung ist, umso genauer kann eine Aussage über die Taktzeit getroffen werden.

Im weiteren Verlauf dieser Bachelorarbeit werden diese Punkte als Leitfaden zur Erstellung des digitalen Zwillings verwendet. Diese Anforderungen werden im Laufe der Arbeit auf Umsetzbarkeit und Sinnhaftigkeit bewertet.

2.4 Theoretische Grundlagen

Bevor das Modell erstellt wird, ist es wichtig, die theoretischen Grundlagen des digitalen Zwillings zu kennen, um diese in Visual Components umsetzen zu können. Im Folgenden werden erst die Grundlagen eines digitalen Zwillings erklärt, anschließend die verwendeten Programme und zum Schluss wird auf die Kommunikationsschnittstellen und die dafür genutzte Hardware zwischen Roboter, SPS und Simulation eingegangen.

2.4.1 Digitaler Zwilling

Ein digitaler Zwilling ist die Darstellung eines Produkts, Prozesses oder einer Dienstleistung in einer digitalen Form. Der digitale Zwilling muss also nicht die Simulation einer ganzen Anlage sein, sondern kann auch die Simulation einer Feder, eines Triebwerks oder ähnlichem sein. In dieser Arbeit wird die Darstellung des Palettierprozesses, des FCP, fokussiert [12]. Durch diese digitale Darstellung kann das aktuelle und potenziell mögliche Verhalten eines Systems vorhergesagt und dargestellt werden. Das ermöglicht es, Optimierungen an Prozessen und Objekten vornehmen zu können. Für die Darstellung des Systems verwendet der digitale Zwilling Sensoren, die in Echtzeit ausgewertet werden. Diese Sensoren können Daten sowohl von der realen Anlage in die Simulation als auch von der Simulation über eine SPS zurück an die reale Anlage überführen.

Der digitale Zwilling bildet zusammen mit der realen Anlage eine Art Datenkreislauf, zu sehen in Abbildung 2.2 [13]. Die reale Anlage erfasst Daten beispielsweise über einen Sensor und gibt diese weiter an den digitalen Zwilling. Nachdem diese Daten im digitalen Zwilling verarbeitet wurden, können Parameter am digitalen Zwilling geändert und anschließend simuliert werden. Diese Ergebnisse können in Form von neuen Programmen an die reale Anlage zurückübertragen werden, wo aufgrund der Änderung neue Prozessabläufe entstehen. Von diesem Punkt ausgehend können die Daten im digitalen Zwilling ausgewertet und überarbeitet werden, bis das Ergebnis der realen Anlage dem gewünschten Ergebnis entspricht.

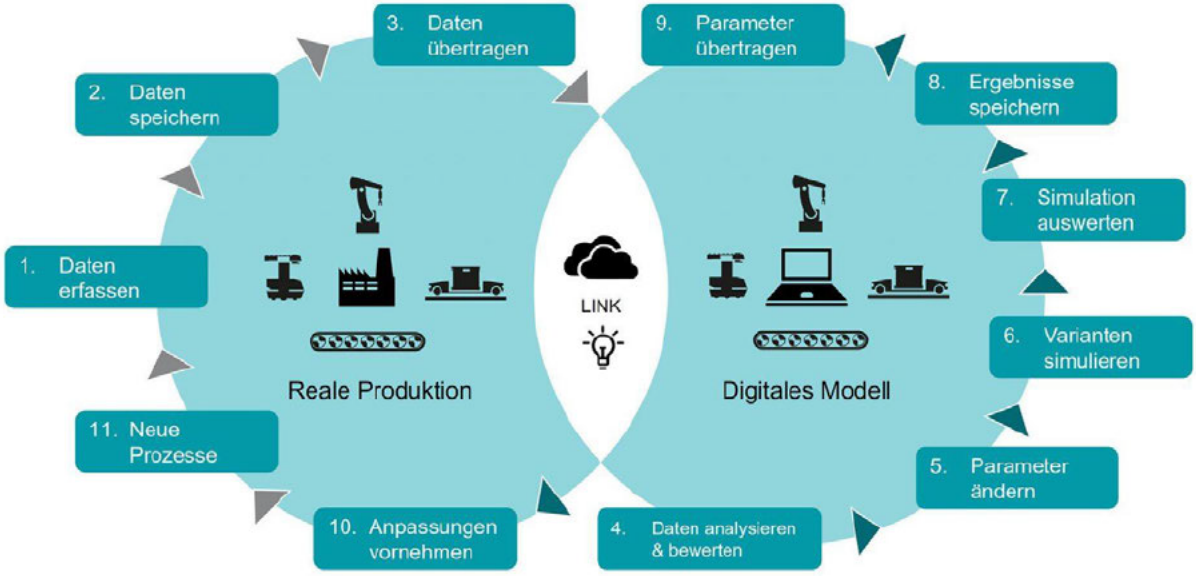


Abbildung 2.2: Funktionsweise eines digitalen Zwillings

2.4.2 Visual Components

Visual Components ist eine 3D-Simulation- und OLP-Software, welche vom gleichnamigen Unternehmen Visual Components entwickelt wird [14]. Die erste Version von Visual Components kam 2016 auf den Markt. 2017 wurde das Unternehmen von KUKA übernommen, einem Unternehmen, das die Software als Meilenstein für die Simulation ihrer Roboter ansah. Die Software unterstützt dabei aber nicht nur die hauseigenen KUKA-Roboter, sondern 60 weitere Roboterhersteller. Visual Components gibt es in drei verschiedenen Modelltypen, Essential, Professional und Premium. In dieser Arbeit kommt die Premiumversion zum Einsatz.

Im Folgenden werden die Funktionen bzw. Möglichkeiten von Visual Components grob erklärt. Visual Components bietet eine Liste von Komponenten/Produkten, eCatalog genannt, auf die beim Aufbau einer Simulation zurückgegriffen werden kann. So kann z. B. der UR10 für die Simulation direkt aus Visual Components genommen werden, zu sehen in Abbildung 2.3. Das hat den Vorteil, dass in diesem Modell Achsenlimits, Geschwindigkeit und Beschleunigung bereits hinterlegt sind.

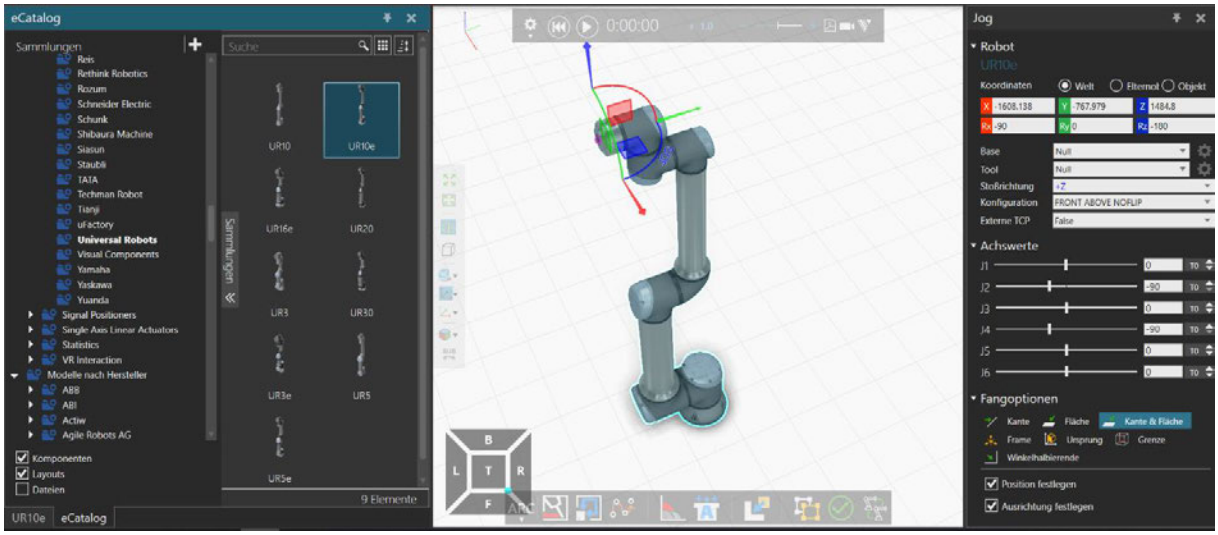


Abbildung 2.3: UR10e aus dem eCatalog

Des Weiteren gibt es die Möglichkeit Modellierungen von Komponenten vorzunehmen, so können Bauteile, die es bisher nicht im eCatalog gibt, erstellt und bearbeitet werden und anschließend lässt sich diesen eine Logik oder Verhaltensweise zuweisen. So ist es ebenso möglich, Sensoren, Gelenke, Linearachsen usw. innerhalb der Simulation abzubilden, zu sehen in Abbildung 2.4.

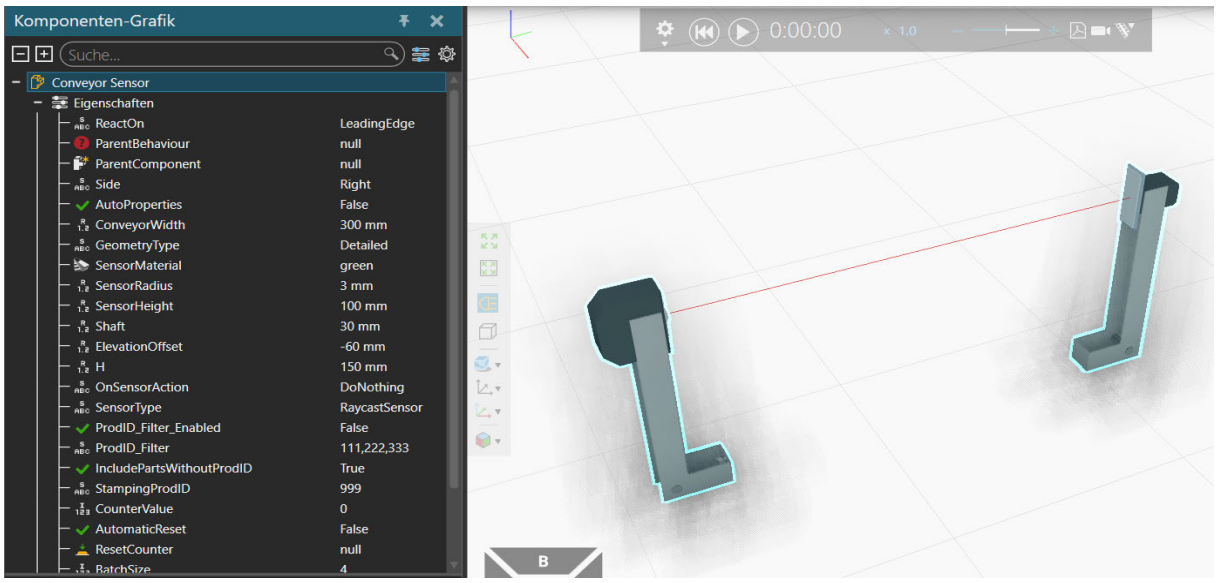


Abbildung 2.4: Modellierung und Konfiguration optischen Sensors

Unter dem Menüpunkt „Prozess“ gibt es die Möglichkeit, einen Materialfluss einzustellen, gezeigt wird dies in Abbildung 2.5. Produkte können auf diese Weise von einer Maschine bearbeitet werden oder zwischen zwei Orten transportiert werden. In diesem Beispiel gibt es einen Feeder der Produkte und eine Process Node erstellt, welche die Produkte an ein Förderband weitergibt. Der Roboter ist hier als Transportmittel ausgewählt. Der Pfad, den der Roboter für diese Aufgabe fahren muss, wird hierbei von Visual Components berechnet.

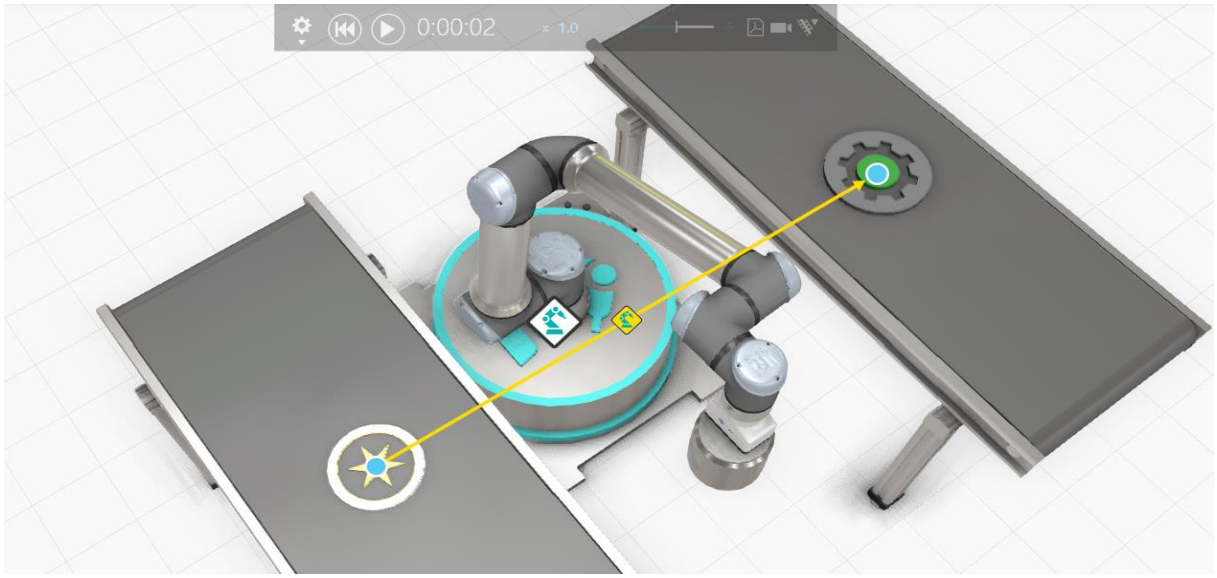


Abbildung 2.5: Materialfluss zwischen zwei Förderbändern mit einem UR10e

Visual Components ermöglicht auch, Roboterprogramme direkt innerhalb der Simulation zu programmieren. Damit können Erreichbarkeits- und Taktzeitanalysen gemacht, Abläufe eines Prozesses nachgestellt und Programme geschrieben werden. Mit der Premium-Lizenz von Visual Components ist es möglich, den digitalen Robotercode zu exportieren und auf einen realen Roboter zu laden.

Die Software bietet zudem an, von der aufgebauten Simulation Fertigungszeichnungen zu erstellen, abgebildet in Abbildung 2.6. So können Zeichnungen von einzelnen Komponenten sowie Zusammenbauzeichnungen erstellt werden. Die Funktionen sind hier aber sehr beschränkt und fokussieren sich nur auf die grundlegenden Features. Visual Components ermöglicht somit, Bauteile zu bemaßen und Stücklisten zu erstellen, da Alternativen wie Schnittansichten oder Durchmesserbemaßungen an dieser Stelle von Visual Components nicht möglich sind.

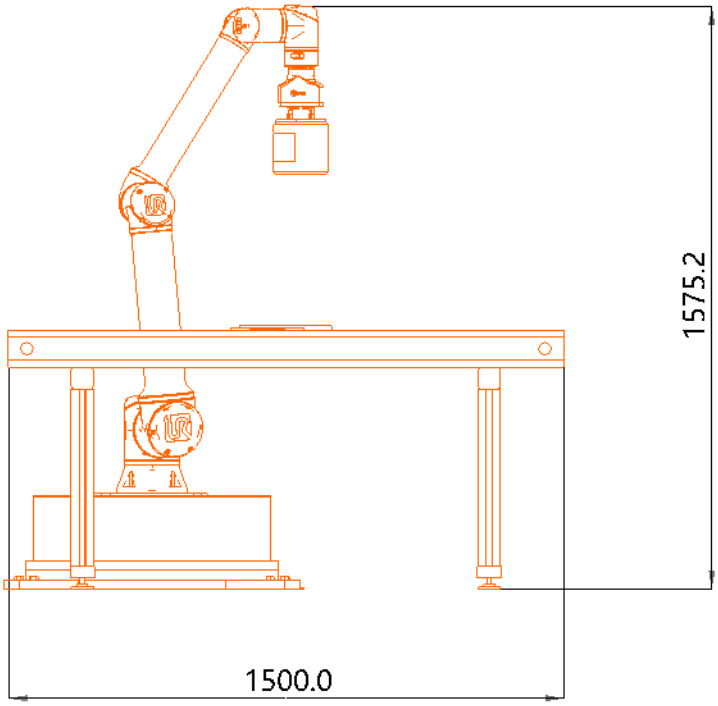


Abbildung 2.6: Gesamtzeichnung eines UR10e mit Förderband

Zuletzt ist die Möglichkeit zu erwähnen, Visual Components mit verschiedenen SPS- und Robotersteuerungen direkt zu verbinden, was Abbildung 2.7 zeigt. Je nachdem welche Art der Schnittstelle gewählt wird, müssen ein paar Informationen wie die Serveradresse und Logindaten angegeben werden. Außerdem besteht die Möglichkeit mit mehreren Servern gleichzeitig zu kommunizieren. Visual Components gibt bei erfolgreicher Verbindung mit dem Server eine Liste der Variablen aus, die der Server sendet und die von Visual Components genutzt werden können. Diese Variablen werden mit Bauteilen aus der Simulation verbunden. Hierbei wird unterschieden zwischen „Simulation zum Server“ und „Server zur Simulation“. Damit kann entschieden werden, welche Variablen zum Server gesendet und empfangen werden sollen. So können z. B. die Winkeldaten der Roboterarmgelenke vom Robotercontroller ausgelesen und wiederum diese Werte mit den Gelenken des Roboters in der Simulation verbunden werden. Diese Daten sind regelmäßig nach einem Intervall oder alternativ, wenn neue Daten vorhanden sind, zu aktualisieren. Der Roboter in der Simulation hat somit mit nicht wahrnehmbarer Verzögerung zu jeder Zeit dieselbe Achsstellung wie der echte Roboter.

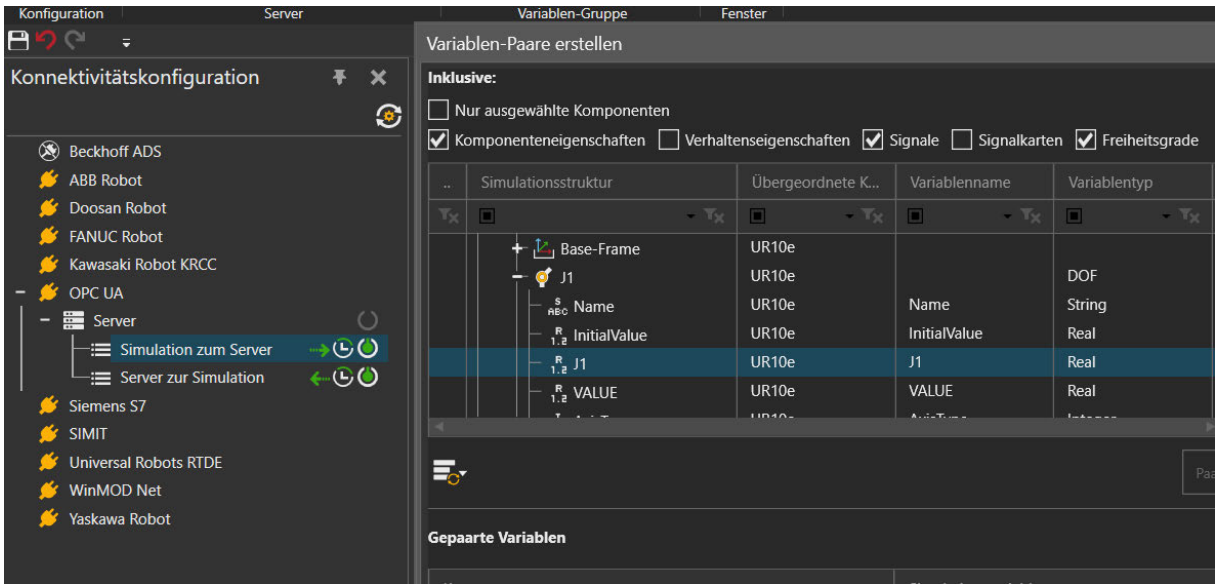


Abbildung 2.7: Kommunikationseinstellungen über OPC UA

2.4.3 Codesys

Codesys, kurz für Controller Development System, ist ein Programm zur Programmierung von speicherprogrammierbaren Steuerungen nach IEC-61131-3 [15]. Die erste Version von Codesys wurde 1994 von 3S-Smart Software Solutions veröffentlicht. Diese Firma wurde 2020 zur Codesys GmbH umfirmiert. Mit Codesys können sechs unterschiedliche Programmiersprachen verwendet werden. Die fünf Sprachen, die in der IEC-61131-3 enthalten sind, und SFP.

- (IL) Instruktion List, im Deutschen (AWL) Anweisungsliste
- (ST) Structured Text, im Deutschen (ST) strukturierter Text
- (LD) Ladder Diagramm, im Deutschen (KOP) Kontaktplan
- (FBD) Function Block Diagramm, im Deutschen (FBS) Funktionsbausteinsprache
- (SFC) Sequential Function Chart, im Deutschen (AS) Ablaufsprache
- (CFC) Continuous Function Chart, im Deutschen (SFP) Signalflussplan

Innerhalb von Codesys wird eine Baumstruktur aufgebaut, welche ein komplettes SPS-Programm ergibt, auch zu sehen in Abbildung 2.8. Der Baum fängt oben mit der Programmbenennung an und geht über die verwendete Hardware, die das SPS-Programm ausführt, zur Programmierung der einzelnen Unterprogramme hin zur externen Hardware, mit der kommuniziert werden soll. Eine genauere Erklärung von Codesys erfolgt unter Punkt 4.1 am Beispiel des SPS-Programms des FCP.

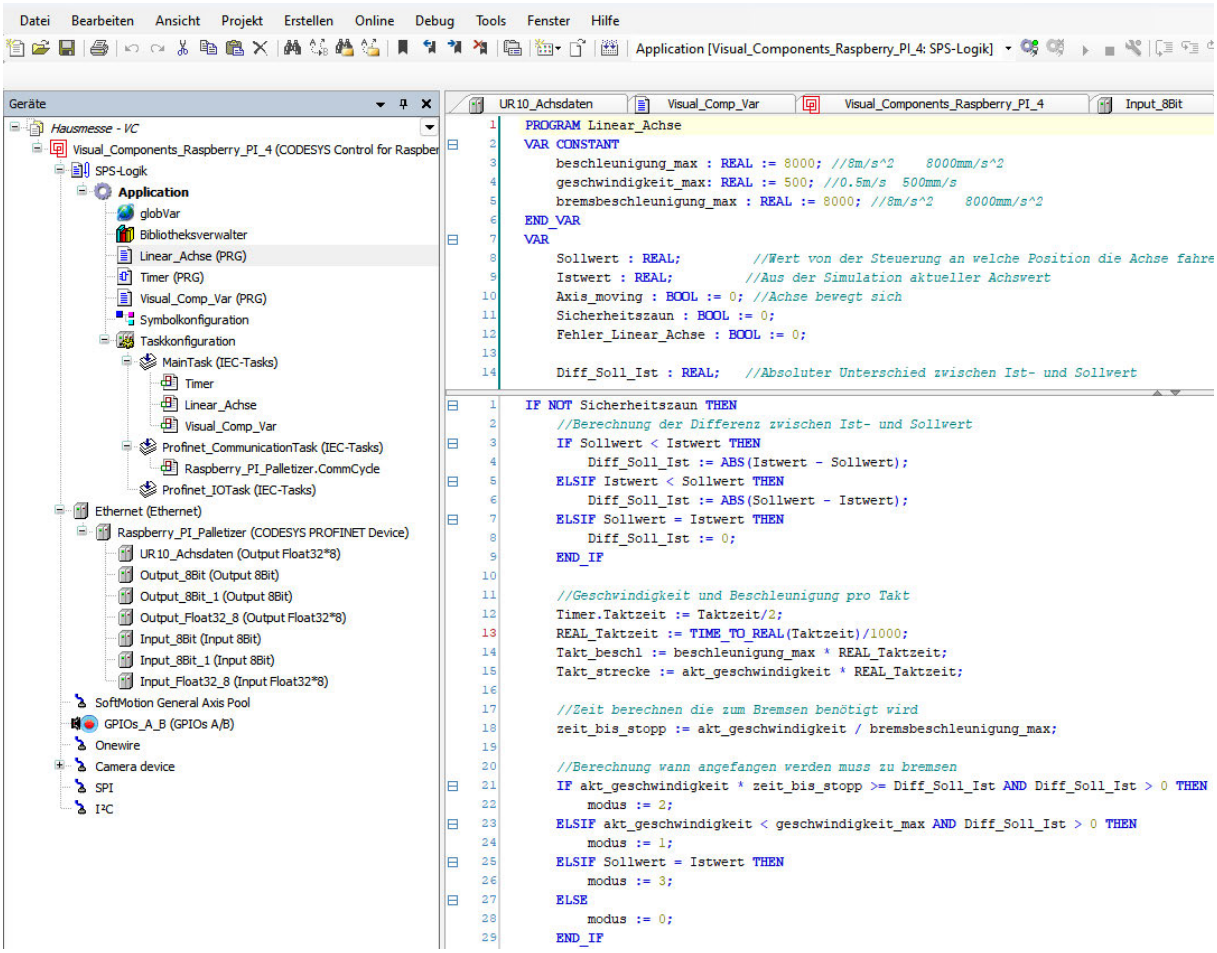


Abbildung 2.8: Codesys mit geladenem Programm

2.4.4 Profinet

Profinet steht für Process Field Network und ist ein Industrie-Ethernet-Standard zur Kommunikation von unterschiedlichen Busteilnehmern im Netzwerk [16]. Profinet wurde von Siemens zusammen mit der Profibus-Nutzerorganisation entwickelt. Außerdem wird Profinet auch im FCP verwendet und ist echtzeitfähig, was eine Grundvoraussetzung für den digitalen Zwilling ist. Im FCP wird Profinet verwendet, um Daten zwischen Soft-SPS und Busteilnehmern zu senden und zu empfangen.

2.4.5 OPC UA

OPC UA ist die Abkürzung für Open Platform Communications Unified Architecture. OPC wurde 2006 zur universellen Kommunikation entwickelt, um Maschinendaten senden und empfangen zu können [17]. Verwaltet und weiterentwickelt wird dieser Kommunikationsstandard von der OPC Foundation. Dieser Standard bietet zusätzliche Sicherheitsstandards, um eine sichere Kommunikation zwischen Teilnehmern zu ermöglichen. In dieser Arbeit wird OPC UA verwendet, um eine Kommunikation zwischen Codesys bzw. der SPS und Visual Components und dem PC herzustellen. OPC UA ist ebenfalls echtzeitfähig,

um Daten aus der realen Welt in Echtzeit in der Simulation darstellen zu können und andersherum.

2.4.6 Raspberry Pi

Der Raspberry Pi ist ein kompakter Einplatinencomputer, der von der Raspberry Pi Foundation entwickelt wurde [18]. Der erste Raspberry Pi kam 2012 auf den Markt und sollte jedem einen kostengünstigen Einstieg in die Programm- und Hardwareentwicklung ermöglichen.

Zum Aufbau des digitalen Zwillings werden 2 Raspberry Pis verwendet. Einer dient dem Ausführen des SPS-Programms des FCP mittels Codesys und einer Soft-SPS, die auf dem Raspberry läuft. Der Zweite stellt die Schnittstelle zwischen dem ersten Raspberry und der Simulation her. Auch auf dem zweiten Raspberry läuft dafür eine Codesys-Soft-SPS. Die Erklärung, warum zwei Raspberry zum Einsatz kommen, wird in Kapitel 4.1 genauer behandelt.

3 Erstellung des Modells

Im folgenden Kapitel geht es um die Erstellung des Grundmodells für den digitalen Zwilling. Dabei werden die vorhandenen CAD-Dateien aufbereitet, zusammengebaut und mit ihrer gewünschten Funktion ausgestattet. Um diesem Kapitel besser folgen zu können, sind in Abbildung 3.1 Bilder des fertigen FCP-Modells, um dessen Erstellung es in diesem Kapitel geht, eingefügt.

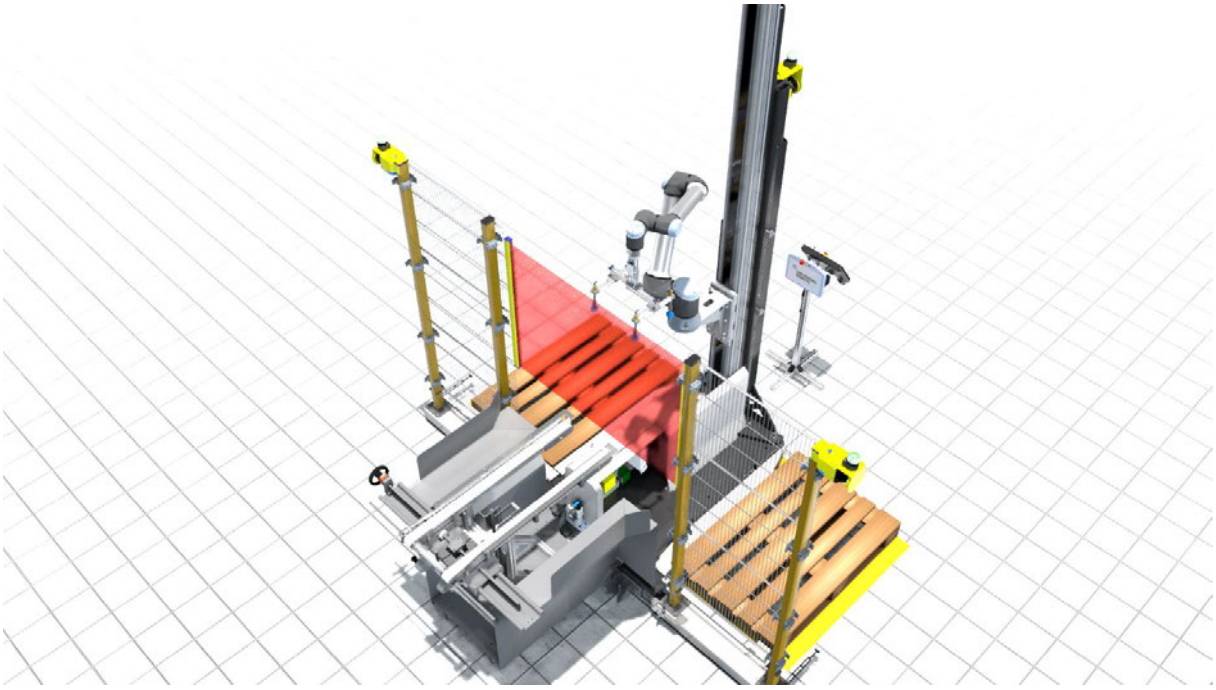


Abbildung 3.1: Modell des FCPs in Visual Components

3.1 Umsetzung des Modells in Visual Components

Um den FCP in Visual Components als digitalen Zwilling nachzubauen, sind verschiedene Schritte notwendig. Zu Beginn müssen die CAD-Dateien importiert und aufbereitet werden. Im nächsten Schritt wird der Sicherheitszaun, welcher auf und zugeklappt werden kann, modelliert. Darauf folgen die Palettensensoren, die Sicherheitsscanner, die Lichtschranke und das Förderband. Abschließend wird der Materialfluss simuliert.

3.1.1 CAD-Dateien importieren und aufbereiten.

Die CAD-Dateien des FCP liegen als „Pack and Go-Datei“ vor. Diese Datei kann von SolidWorks erzeugt werden und bündelt alle Einzelteile und eine Assembly von allen Teilen in der zusammengebauten Form als Zip-Datei. Visual Components kann diese Assembly-Datei nach dem Entpacken direkt importieren.

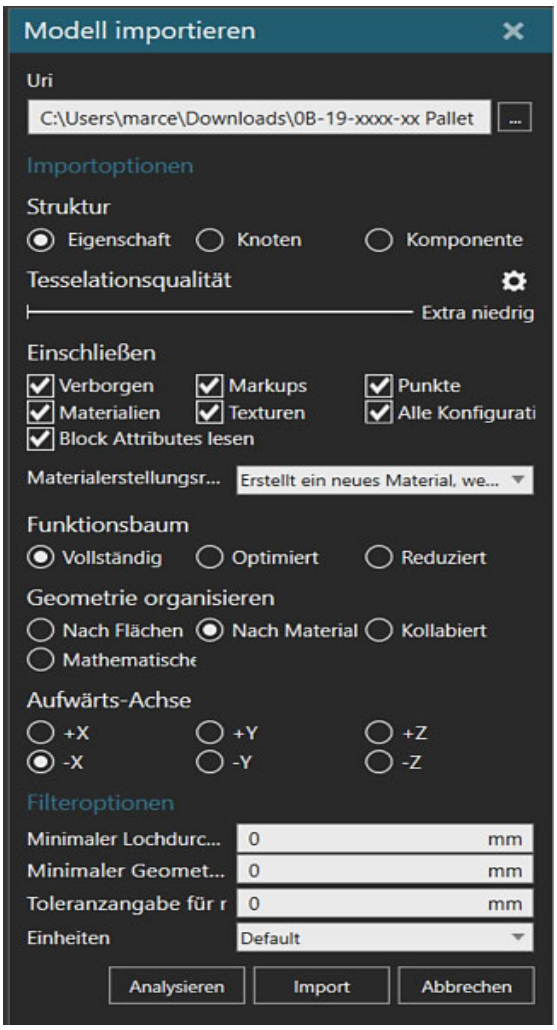


Abbildung 3.2: Importierungseinstellungen in Visual Components

Innerhalb von Visual Components gibt es einige Einstellungsmöglichkeiten für den Import des Modells, zu sehen in Abbildung 3.2. Nachdem die zu importierende Datei ausgewählt wurde, ist drunter der Punkt „Struktur“ zu finden. Dieser verändert, wie das Modell in den Modellbaum geladen wird. In Abbildung 3.3 ist das Ergebnis der verschiedenen Strukturoptionen am Beispiel des FCP zu sehen.

Die erste Importoption heißt „Eigenschaft“. Hier werden alle importierten Komponenten zusammengefasst zu einer Baugruppe. Hierbei wird für jede ehemalige Baugruppe ein Ortsvektor erstellt, in welchen alle weiteren Komponenten untergeordnet werden. Diese bekommen wiederum ihren eigenen Ortsvektor. Dieser gibt die Position von Baugruppen und Komponenten im Raum im Verhältnis zum Ursprung des simulierten Raums an. Diese Importfunktion eignet sich für einzelne Komponenten, die wiederum anderen Baugruppen untergeordnet werden können. Große Baugruppen damit zu importieren, sorgt dafür, dass die Einzelteile nur über die Ortsvektoren verschoben werden können, was die Arbeit mit einem großen Modell schnell kompliziert und mühselig werden lässt.

Die zweite Option „Knoten“ funktioniert im Vergleich zur ersten Methode ohne Ortsvektoren. Die einzelnen Baugruppen des FCP werden hierbei jeweils als einzelne Verbindungen

gespeichert, denen wieder einzelne andere Verbindungen untergeordnet sein können. Den Verbindungen können folglich Funktionen zugeordnet werden. So können z. B. Linearachsen, Gelenke etc. abgebildet werden.

Die dritte Option zum Importieren heißt „Komponente“. Hier werden, wie bei Option Knoten zwei Verbindungen erstellt, allerdings wird hier versucht, den ursprünglichen Modellbaum aus dem CAD-Programm herzustellen. Das kann zu einem ähnlichen Ergebnis wie Option zwei führen, kann aber auch bewirken, dass jedes Teil als einzelne Komponente importiert wird. In diesem Fall ist es nicht mehr möglich alle importierten Teile zusammen zu verschieben, ohne jedes einzeln anzuwählen.

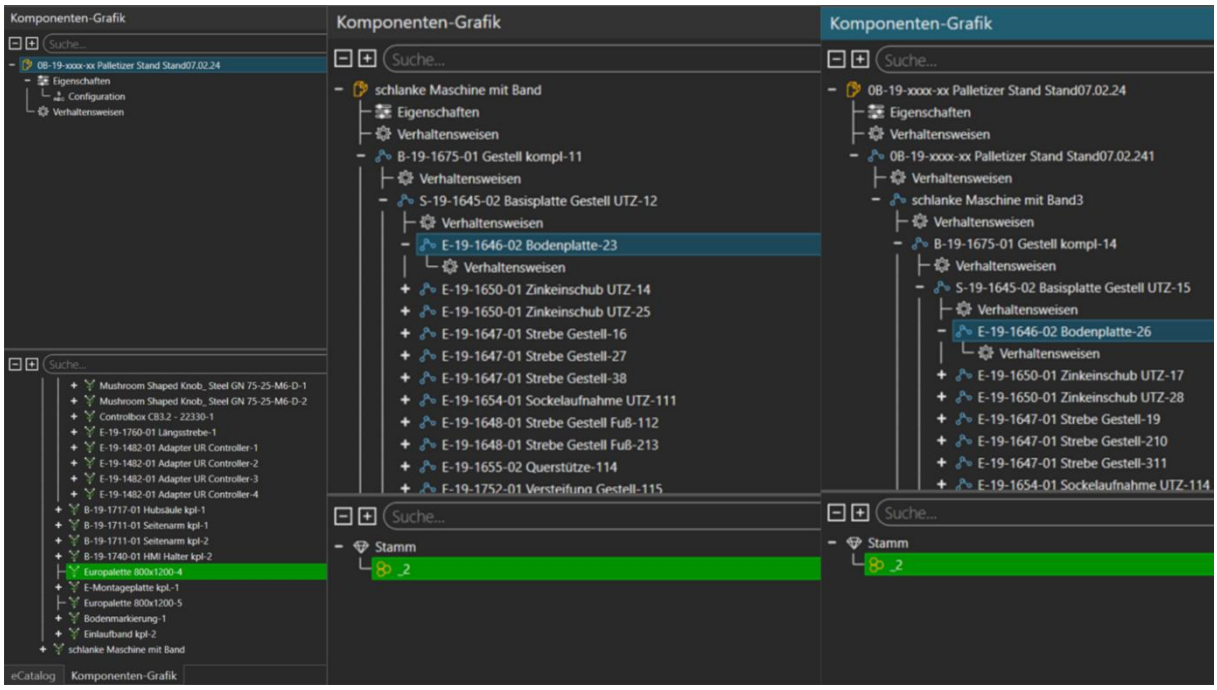


Abbildung 3.3: Links: Eigenschaften; Mitte: Knoten; Rechts: Komponente

Zur Erstellung des Modells, des FCP, wird in dieser Arbeit die zweite Option „Knoten“, zum Importieren des Modells verwendet. Dadurch wird es später möglich, verschiedenen Baugruppen eine Funktion zuzuweisen bei trotzdem leichter Handhabung.

Die nächste Einstellung beim Importieren ist die Tesslationsqualität, hier kann eingestellt werden, wie hoch die Polygondichte für Körper sein soll. Eine kleine Polygondichte sorgt für eine gute Performance, aber dafür werden z. B. Rundungen eckiger dargestellt. Von Visual Components wird empfohlen, die Polygonanzahl unter 100.000 zu halten, um das Modell performant zu halten. Beim digitalen Zwilling wird die Qualität auf niedrig gestellt.

Unter dem Punkt „Einschließen“ kann ausgewählt werden, welche Modelle und Eigenschaften aus den CAD-Dateien übernommen werden sollen. Für den FCP werden alle Haken drin gelassen, sodass das Modell mit allen Informationen, die zur Verfügung stehen geladen werden.

Die letzte Einstellung beim Importieren des Modells werden auf Standard gelassen, hier könnte noch die Baumstruktur des Modells vereinfacht werden, der Aufbau dieser Struktur oder die Orientierung des Modells festgelegt werden.

3.1.2 Modellierung des Sicherheitszauns

Der Sicherheitszaun des FCPs ist dafür da, Personen daran zu hindern, in den Arbeitsbereich des Roboters und der Linearachse zu geraten. Der Zaun bietet hierbei eine physische Grenze. Der FCP ist flexibel aufgebaut, daher können die Zäune jeweils links und rechts abgeklappt werden, wenn nur eine Palette palettiert werden soll. Der Zaun ist in Abbildung 3.4 zu sehen.

Die Funktion des klappbaren Zauns soll auch in Visual Components abgebildet werden, damit Kollisionen zwischen dem Roboter und einem eingeklappten Zaun simuliert werden können.

Dadurch, dass beim Importieren „Knoten“ ausgewählt wurde, haben die Baugruppen, wie z. B. der Sicherheitszaun, ihre eigene Verbindung, der alle Geometrien untergeordnet sind. Innerhalb dieser Verbindungen können Verhaltensweisen für alle untergeordneten Geometrien eingestellt werden. Dadurch können verschiedene Gelenkarten abgebildet werden.

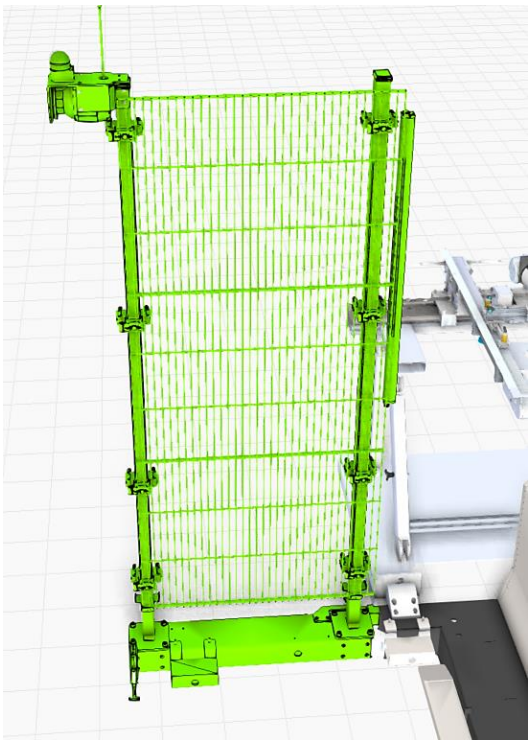


Abbildung 3.4: Schutzzaun in Visual Components

In den Verbindungseigenschaften, hier Abbildung 3.5, der jeweiligen Verbindung des Sicherheitszauns können Einstellungen des Gelenks getätigt werden. So kann unter JointType die Art des Gelenks ausgewählt werden. Da sich dieser Zaun um eine Achse drehen soll, wird hier „Drehend“ ausgewählt. Als Nächstes ist der Ursprung des Koordinatensystems an dem Punkt, um den sich der Zaun drehen soll zu positionieren. In dem Fall der FCPs ist es eine

Schraube. Zuletzt ist auszuwählen, um welche Achse des Koordinatensystems gedreht werden soll, hier die Hochachse, genauer gesagt die Z-Achse.

Danach lassen sich die Grenzen, maximalen Geschwindigkeiten und Beschleunigungen festlegen. Die Grenzen sind beim FCP 0° minimal bis 90° maximal, das bedeutet der Zaun kann maximal, bis er parallel zum Standfuß des FCPs eingeklappt ist oder bis er senkrecht zu diesem steht, ausgeklappt werden. Der Sicherheitszaun kann jetzt um diese Achse gedreht und von anderen Komponenten wie dem Roboter angesteuert werden. Die Verzögerungs- und Einschwingzeit, genau wie die Geschwindigkeit und Beschleunigung, sind hierbei keine relevanten Werte, da Zäune vor dem Prozessstart geöffnet oder geschlossen werden und während des Verlaufs des Prozesses in dieser Position bleiben.

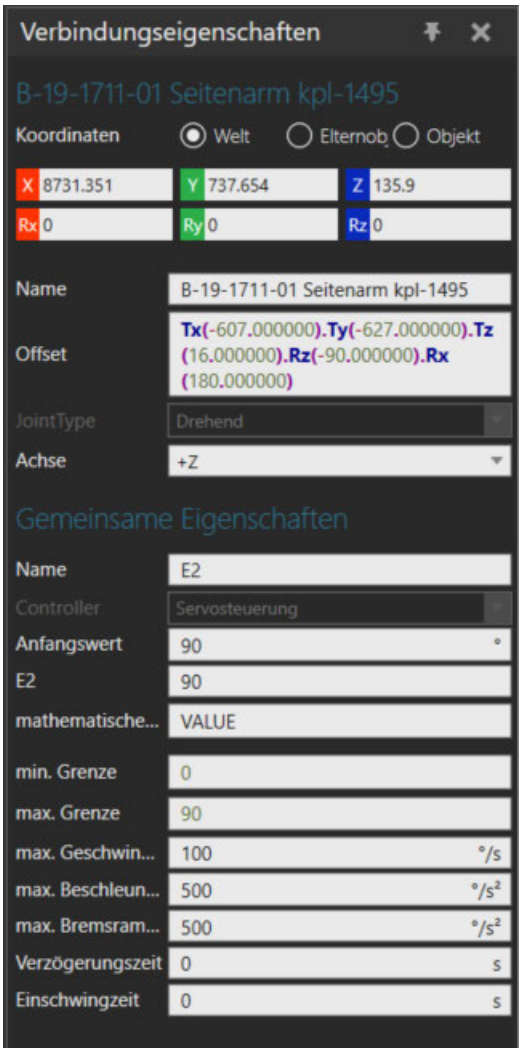


Abbildung 3.5: Konfiguration des Seitenarms in Visual Components

3.1.3 Modellierung der Palettensenoren und Erstellung des Prozessablaufs

Der FCP besitzt zwei Palettenstellplätze. Bei diesen Sensoren handelt es sich um kapazitive Sensoren, welche somit auch Holz oder Plastik der Palette erkennen können, sodass der Roboter die Information bekommt wann ein Palettierprozess gestartet werden kann. Einer

befindet sich auf der linken und einer auf der rechten Seite. Die beiden Palettensensoren sitzen auf der Innenseite, am Standfuß des FCPs. Diese Palettenplätze können mit verschiedenen Palettengrößen besetzt und anschließend vom Roboter beladen werden.

Im digitalen Zwilling müssen die Palettensensoren detektieren, wann eine Palette am linken oder rechten Beladeplatz steht und daraufhin ein Signal an die Steuerung weitergeben. Im ersten Schritt reicht es, wenn ein Signal geschaltet wird, sobald eine Palette vorhanden ist, die Verbindung zur SPS wird in einem späteren Schritt behandelt. Im Zuge der Erstellung der Palettensensoren wird auch der Materialfluss der Paletten konfiguriert, sodass in der Simulation ein Mitarbeitender leere Paletten mit einem Hubwagen anliefert. Das hilft dabei, später realitätsnähere Taktzeiten zu simulieren, da die benötigte Zeit zum Wechseln voller Paletten zu leeren auch simuliert werden kann. In der folgenden Abbildung 3.6 kann dieser Ablauf angesehen werden.

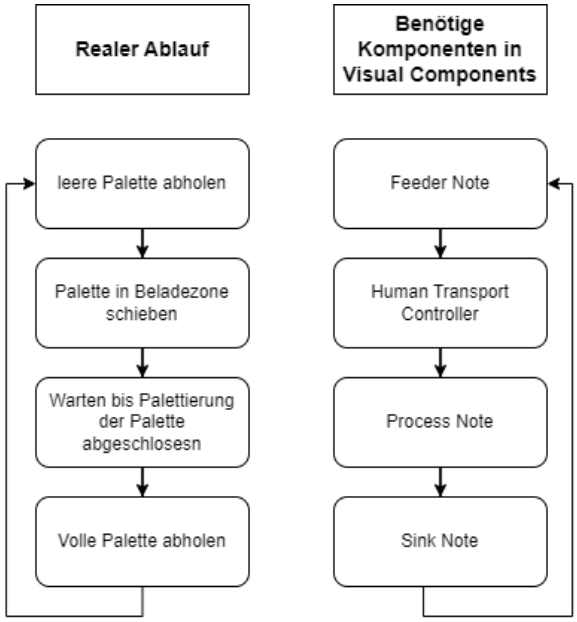


Abbildung 3.6: Materialfluss und Umsetzung in Visual Components

Umgesetzt wird das Ganze über einen Prozessablauf mit festgelegten Produkten. In Visual Components wird dafür als Erstes eine Feeder Node, aus dem eCatalog, an die Stelle gezogen, wo später die leeren Paletten abgeholt werden sollen. Für den gesamten Ablauf wird noch eine Process Node an der Stelle, wo die Paletten beladen werden, und ein Sink Process, benötigt. Zusätzlich werden ein Hubwagen und eine Person an eine Stelle gestellt, wo sie in der Simulation befinden sollen, beim Start der Simulation.

Für den Process Ablauf muss zuerst, ein sogenanntes „Produkt“, in Visual Components, erstellt werden, welches zur jeweils benötigten Node transportiert wird. Im Fall des FCPs ist das Produkt eine Europalette, welche es bereits fertig im eCatalog gibt. Unter dem Menüpunkt „Process → Produkte → neues Produkt“ kann ein neues Produkt erstellt werden. Dafür wird ein Name festgelegt und als Geometrie die Europalette ausgewählt. In der Feeder Node wird das eben erstellte Produkt ausgewählt. Die Feeder Node erstellt dabei in einem festgelegten Intervall eine eingestellte Menge an Produkten. Die Position des erstellten Produkts wird

unmittelbar auf die Feeder Node eingestellt, das Intervall zur Erstellung dieser wird auf 30 Sekunden gestellt und die Anzahl wird nicht begrenzt. Diese Einstellungen wurden so gewählt, dass die Paletten immer vorhanden sind, wenn eine benötigt wird, da die Prozesszeit des Roboters um ein Vielfaches größer ist als die Zeit zwischen der Erstellung von Paletten.

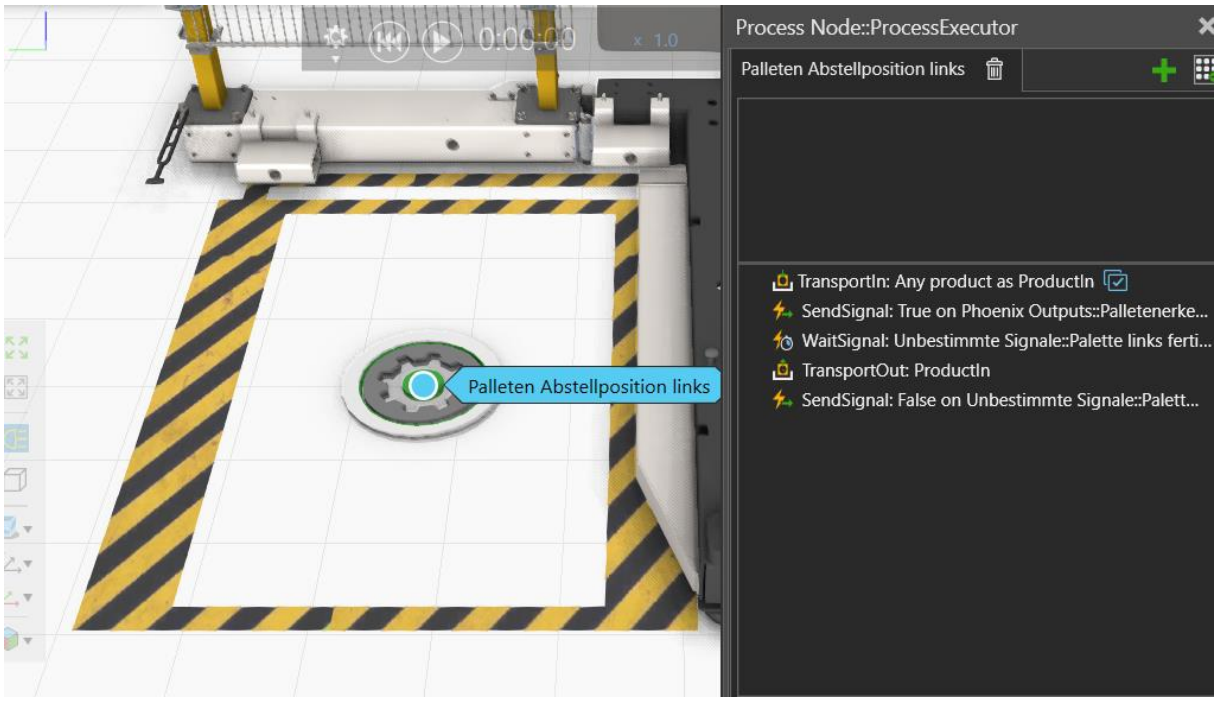


Abbildung 3.7: Process Node und zugehöriger Prozessablauf

Als Nächstes wird die Process Node konfiguriert, zu sehen in Abbildung 3.7. Diese fungiert hier als zum einen als Abstellposition für die Europaletten als auch als Sensor dieser. Innerhalb von Visual Components gibt es zwar Sensoren, diese sind aber für Förderbänder ausgelegt und daher hier nicht einsetzbar. Der Prozessablauf innerhalb der Process Node ist in verschiedene Programmanweisungen wie folgt, aufgeteilt:

1. Die Process Node erwartet eine Anlieferung eines Produkts, in diesem Fall eine Europalette. Nach der Anlieferung können keine anderen Produkte an dieselbe Process Node geliefert werden.
2. Als Nächstes wird ein Signal zur Schnittstelle der SPS ausgesendet, dass eine Palette links vorhanden ist. Mit diesem Signal wird damit der Palettensensor innerhalb der Simulation abgebildet.
3. Im dritten Schritt wartet die Process Node auf ein externes Signal von der SPS, dass die Palettierung der Palette abgeschlossen ist.
4. Im vierten Schritt wird ein Signal an den Prozessablauf gesendet, dass die Palette abgeholt werden kann. Wer und was die Palette daraufhin abholt, wird hierbei nicht von der Process Node verwaltet.
5. Im letzten Schritt wird das Signal des Palettensensors zurückgesetzt, sobald die Palette abgeholt wurde. Das Programm startet danach erneut bei Schritt 1.

Dieser Ablauf und die dazugehörigen Einstellungen werden für den zweiten Palettensensor auf der rechten Seite wiederholt, mit dem Unterschied, dass dieser seine eigenen Signale und eine andere Position hat.

Um den Transport der Europaletten zu verwalten, wird in Visual Components ein „Human Transport Controller“ verwendet. Dieser ist bereits im eCatalog zu finden und regelt, welche Palette von welchem simulierten Menschen, mit welchem Werkzeug, abgeholt werden soll. Dafür werden alle verfügbaren Ressourcen mit diesem Controller verbunden, die Logik, für die effizienteste Ressourcennutzung, wird daraufhin vom Controller selbstständig berechnet. Das Einzige, was in dem Controller noch eingestellt wurde, ist die Anfahr- und Ablagerichtung, aus der die Produkte aufgenommen und abgelegt werden, sodass die Paletten weder verdreht aufgenommen noch abgelegt werden.

Als letzten Schritt im Ablauf des Prozesses steht die Sink Node. Diese Node löscht alle Produkte, die angeliefert werden, sodass nicht zu viele Produkte gleichzeitig in der Simulation vorhanden sind, um die Performance aufrechtzuerhalten. In dieser Node müssen keine Einstellungen geändert werden.

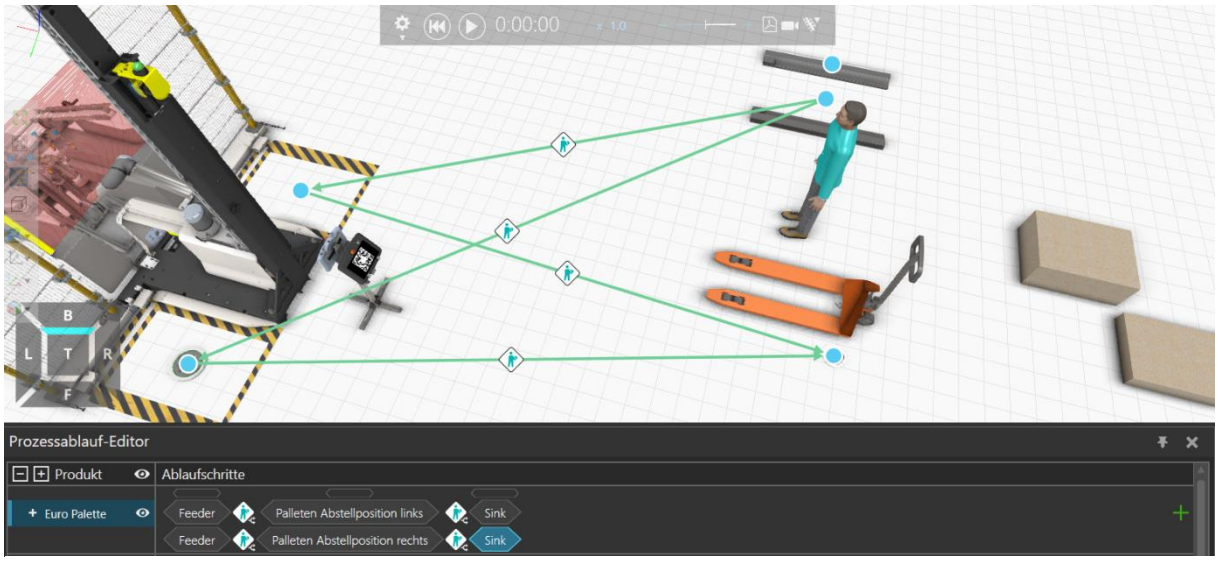


Abbildung 3.8: Prozessablauf in Visual Components

Im letzten Schritt werden alle Komponenten, die für den Materialfluss nötigt, sind verbunden, um die Flussrichtung festzulegen, zu sehen in Abbildung 3.8. Unter dem Menüpunkt „Prozess → Ablauf“ können alle „Flow Components“, also die Feeder, Process und Sink Node, mit Pfeilen, in Ablaufrichtung, verbunden werden. Innerhalb dieser Pfeile ist auszuwählen, wer den Materialfluss verwaltet. In diesem Fall wird hier der „Human Transport Controller“ ausgewählt. Das sorgt dafür, dass die Ressourcen, die in diesem Controller hinterlegt sind, benutzt werden, um die Produkte zu befördern. Im Prozessablauf-Editor müssen anschließend alle Nodes in die richtige Reihenfolge gebracht werden. Dadurch können Prioritäten festgelegt werden, da aber hier die linke und die rechte Seite des FCPs gleichwertig sind, werden alle Abläufe parallel eingestellt und somit keine Seite priorisiert behandelt.

3.1.4 Modellierung der Sicherheitsscanner

An dem FCP sind drei Flächensicherheitsscanner, von Datalogic, verbaut. Einer befindet sich vorne an der Linearachse und jeweils einer an den Außenkanten der Sicherheitszäune links und rechts. Beim FCP sorgen sie für die Sicherheit, denn sollte jemand den Sicherheitsbereich betreten, in dem sich der Roboter bewegen kann, stoppt dieser und die Linearachse ihre Bewegung, bis der Bereich, über die HMI, wieder freigegeben wird. Da in Visual Components eine Umsetzung des Quittierens nicht so einfach umsetzbar ist, sollte es vorerst ausreichen die Quittierung automatisch vorzunehmen, wenn der Sicherheitsbereich wieder freigegeben wird.

Innerhalb von Visual Components gibt es einen generischen Flächensicherheitsscanner im eCatalog. Da dieser allerdings andere Abmaße und infolgedessen einen verschobenen Sicherheitsbereich hat, wurde der Datalogic Sicherheitsscanner in Visual Components nachmodelliert. Der bereits vorhandene Sicherheitsscanner dient dabei dennoch als Grundlage, da die vorhandenen Funktionen nahezu dieselben sind.

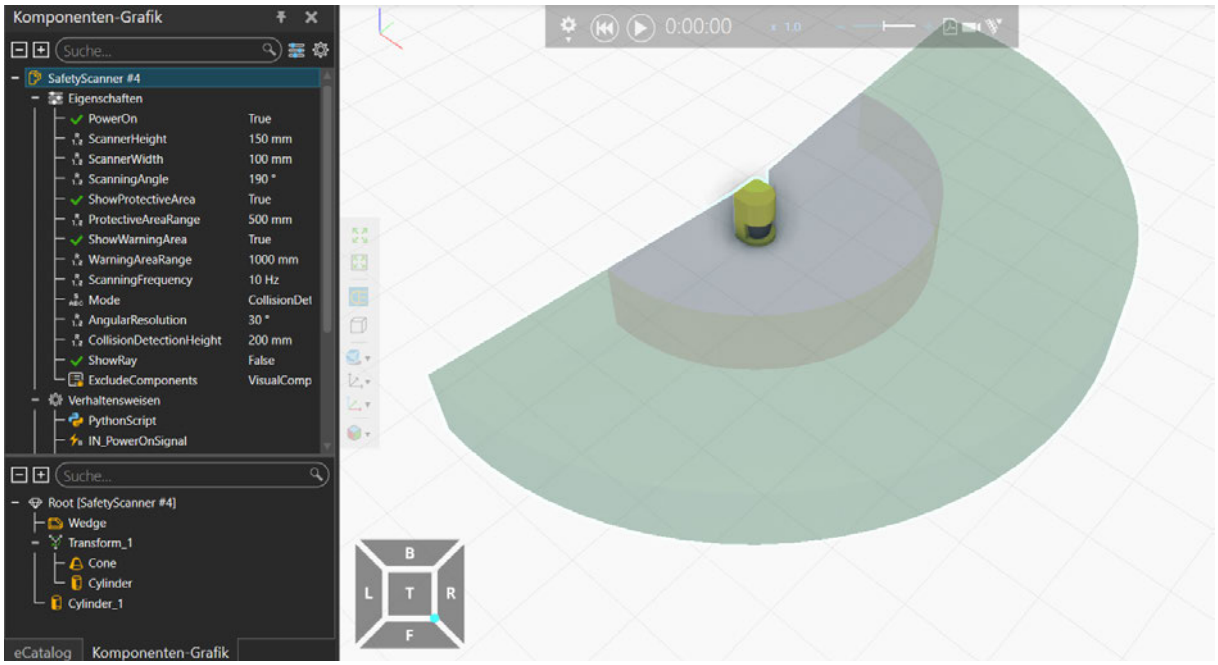


Abbildung 3.9: Modellierung des Flächensicherheitsscanners in Visual Components

Zum Erstellen des Flächensicherheitsscanners wird eine neue, leere Datei in Visual Components geöffnet und der Sicherheitsscanner später in das Modell des FCP importiert. In Abbildung 3.9 ist der Modellaufbau des in Visual Components vorhandenen Sicherheitsscanners zusehen. Auf der linken Hälfte in Abbildung 3.9 ist die „Komponenten-Grafik“ zusehen. Diese legt die anpassbaren Einstellungen, die Signale und die Geometrien fest. Die Einstellungen im oberen Teil können beim Laden der Komponente aus dem eCatalog angepasst werden, sodass sie in der jeweiligen Simulation verwendet werden können. Das Python-Skript, weiter unten im Modellbaum, liest diese Eigenschaften aus und gibt je nach Situation Signale, zum Scannerbereich aus, die wiederrum an andere Komponenten, innerhalb der Simulation oder eine externe SPS weitergegeben werden können. So kann in dem angegebenen Bereich mit der angegebenen Frequenz nach anderen Komponenten gescannt

und bei Verletzung des Sicherheitsbereichs, ein Signal ausgegeben werden. Im unteren Teil des Modellbaums befinden sich die Geometrien, aus denen der in Abbildung 3.9 zusehende Sicherheitsscanner zusammengesetzt ist. Diese Geometrien sind dabei parametrisch abhängig von den oben eingegebenen Eigenschaften „ScannerHeight“ und ScannerWidth“.

Um diese Scannereigenschaften auf den verbauten Scanner zu übernehmen, müssen in erster Linie die Geometrien vom vorhandenen Scanner geändert werden. Zunächst werden dafür alle vorhandenen Geometrien gelöscht und durch die CAD-Dateien des Datalogic-Scanners ersetzt. Dafür werden als Erstes die CAD-Daten als „Eigenschaft“ importiert und dessen Geometrien ausgeschnitten und in den vorhandenen Scanner eingefügt. Über „Verschieben“ werden die Komponenten anschließend in Position gebracht. Als Nächstes wird die Höhe des Scanbereichs auf 0,4 mm heruntergesetzt. Das entspricht der im Datenblatt angegebenen Erkennungshöhe. Daher wird auch die Berechtigung, diesen Wert zu ändern, im Modell entzogen, sodass dieser Wert für das Modell konstant ist.

Der Scanner ist nun einsatzbereit, allerdings hat das geladene CAD-Modell noch ein Kabel an der Rückseite des Scanners, welches gerade heraussteht. Damit die Option besteht, das Kabel später schnell aus- und einzublenden, wird das Modell des Kabels parametrisch aufgebaut. Dafür muss zunächst das Kabel von dem Modell des Scanners isoliert werden. Mit einem Rechtsklick auf das Modell des Scanners öffnet sich die Schnellwerkzeugleiste. Unter „Werkzeuge → Explodieren“ kann das Modell in kleinere Teile unterteilt werden. Welche Flächen zusammen eine neue Geometrie ergeben, rechnet Visual Components dabei selbstständig aus. Alle Geometrien, die zum Kabel gehören, können nachfolgend angewählt und mit „Rechtsklick → Features vereinen“ wieder zu einer Geometrie gemacht werden, dasselbe gilt für den Rest des Scanners. Der Scanner besteht somit aus zwei Geometrien, eine fürs Kabel und eine für den Scanner selbst. Unter dem Punkt „Eigenschaften“ wird eine boolesche Variable namens „ShowCable“ hinzugefügt. Diese kann per Klick und Setzen eines Hakens auf True oder False gesetzt werden. Im Modellbau, wo die Geometrien zu finden sind, wird ein „Schalter“ hinzugefügt und die Geometrie des Schalters diesem untergeordnet. In dem Schalter wird darauffolgend die eben erstellte Variable ausgewählt. Ist „ShowCable“ auf True gesetzt, ist somit das Kabel sichtbar und wenn nicht ausgeblendet.

Als Letztes wird noch der Ursprung des Flächenscanners bearbeitet. Dieser Punkt dient beim Platzieren dessen als Referenzpunkt. Unter „Modellierung → Positionieren“ kann der Punkt des Ursprungs angepasst werden. In diesem Fall ergibt es am meisten Sinn, die Mitte der zwei auf der Rückseite des Scanners befindlichen Anschraubpunkte als Ursprungspunkt auszuwählen, da bei der Positionierung die Anschraubpunkte als Referenz auswählbar sind.

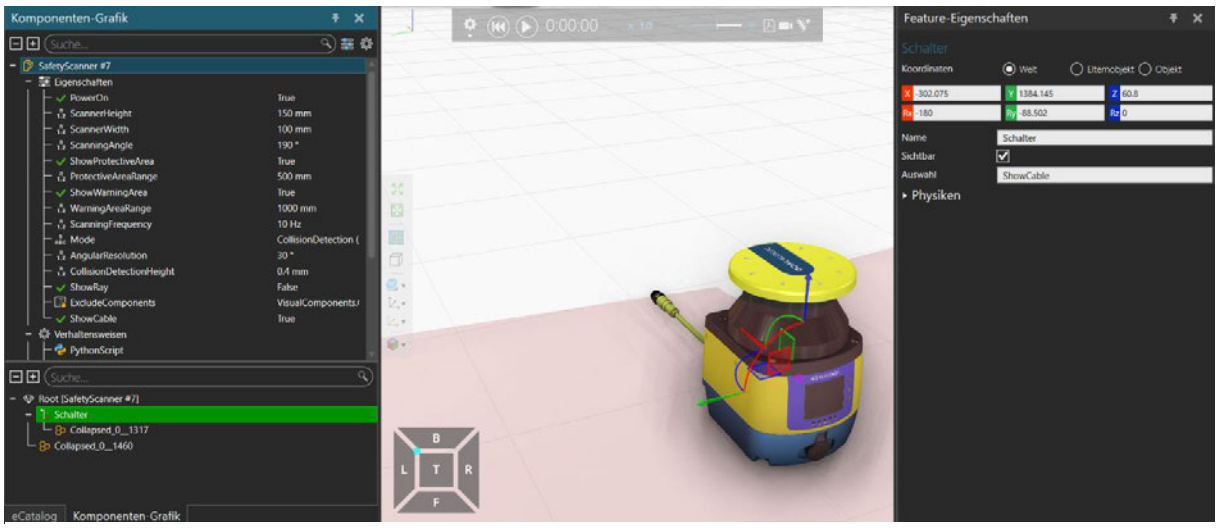


Abbildung 3.10: Modellierung des Sicherheitsscanners in Visual Components

Nachdem die Modellierung des Scanners so weit abgeschlossen ist, wird dieses Modell dem eCatalog hinzugefügt und kann von da aus immer wieder verwendet werden. So können auch schnell drei Scanner in das Modell des FCP geladen werden. Wie das Hinzufügen von Komponenten zum eCatalog funktioniert, wird in Kapitel 3.1.8 genauer behandelt.

Die Scanner werden nach dem Laden aus dem eCatalog positioniert. Dafür werden über „Modellierung → Andocken“ die beiden Anschraubpunkte für jeden Scanner ausgewählt und dieser wird somit an die richtige Position verschoben. Die zwei Scanner am Sicherheitszaun sollen sich jedoch mitbewegen, wenn diese eingeklappt werden. Dafür wird erst über „Start → Fügen“ der Scanner ausgewählt und anschließend der Sicherheitszaun. Der Scanner folgt nun den Bewegungen des Sicherheitszauns.

Als Letztes werden noch die Sicherheitszonen der Scanner so eingestellt, dass sie den kompletten Bereich um den Loader, an der Vorderseite und seitlich erfassen können. Hier gibt es einen großen Unterschied zum realen Scanner, da der Scannerbereich dort komplett modular in alle Richtungen eingestellt werden kann, während in Visual Components nur ein runder Erkennungsbereich möglich ist. Mehr zu den Unterschieden in Kapitel 5.2. Des Weiteren macht das Einbinden der Sicherheitsscanner aus der Simulation in die reale Anlage Probleme, mehr dazu in Kapitel 4.1.

3.1.5 Modellierung der Lichtschranke

Das FCP besitzt eine Lichtschranke auf der Seite des Förderbandes. Diese Lichtschranke sorgt dafür, dass die Linearachse nicht mehr fahren kann, sondern nur noch der Roboter. Der Unterschied, der beiden ist, dass der Roboter kollaborativ ist und die Linearachse nicht. Das heißt, der Roboter hält bei einer leichten Berührung an, während die Linearachse weiterfahren würde, bis das Motorlimit erreicht ist. Der Roboter fährt während des Palettierprozesses beim Abholen der Pakete mehrmals durch die Lichtschranke und löst diese damit aus. Damit in der Simulation die Linearachse auch nicht mehr bewegt werden kann, wenn der Roboter sich im Sicherheitsbereich der Lichtschranke befindet, muss diese mit ihren Funktionen in Visual Components nachmodelliert werden.

Wie bei den Flächensicherheitsscannern gibt es auch hier wieder eine generische Lichtschranke innerhalb des eCatalog von Visual Components, die hier als Grundlage verwendet wird. Wie auch bei der Erstellung des Flächensicherheitsscanners wird hier eine leere Datei geöffnet, in die der generische „Safety Light Curtain“ hineingezogen wird.

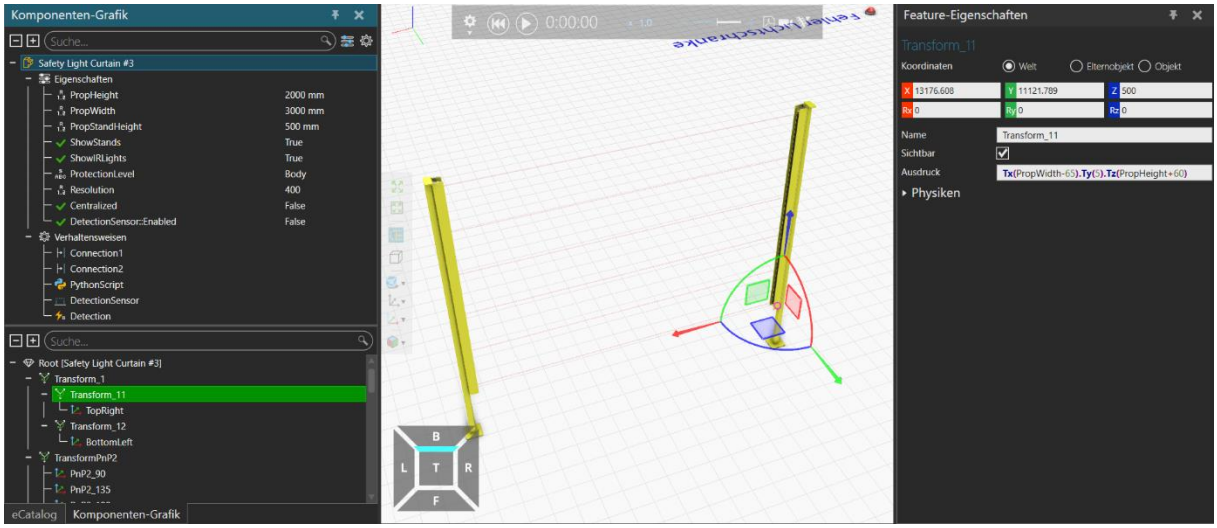


Abbildung 3.11: Generische Lichtschrank aus dem eCatalog in Visual Components

Der größte Unterschied zur Erstellung des Modells des Flächensicherheitsscanners ist, dass die Lichtschranken hier modularer aufgebaut sind als die Flächenscanner, was die Modellierung schwieriger macht. Durch diese Modularität können unter anderem die Höhe der Lichtschranken, aber auch der Erkennungsabstand oder der Ursprung eingestellt werden.

Zu Beginn werden die Standfüße der Lichtschranken entfernt, da die verwendeten Lichtschranken am FCP diese nicht besitzen. Dafür können die entsprechenden Geometrien ausgewählt und gelöscht werden. Darauffolgend werden die anderen Geometrien ausgetauscht durch die jeweiligen Geometrien der Datalogic-Lichtschranke. Dafür wird als Erstes das CAD-Modell als „Eigenschaft“ importiert und unter „Modellierung → Rechtsklick → Explodieren“ in seine einzelnen Geometrien aufgeteilt. Diese können 1:1 mit denen der generischen Lichtschranke ausgetauscht werden. Das Problem, das sich hierbei jedoch ergibt, ist, dass die Höhe der generischen Geometrien eine andere ist als die der Datalogic-Lichtschranke. Das führt dazu, dass die eingebbare und die tatsächliche Höhe nicht mehr übereinstimmen. Um die Skalierung anzupassen, müssen die Vektortransformationen, in denen die jeweiligen Geometrien gespeichert sind, um den Faktor der Größenänderung, angepasst werden.

$$\frac{Höhe_{generisch}}{Höhe_{Datalogic}} = Faktor_{Transformation}$$

Rechnung muss für jede Geometrie gemacht werden, die mit der Höhe skaliert werden soll. Die oberen Endkappen der Lichtschranke lesen die berechnete Höhe aus der Transformation aus und addieren einen festen Offset, um immer am oberen Ende der Lichtschranke zu sein. Zu sehen ist das Ganze in Abbildung 3.12. Der neue berechnete Wert steht dabei rechts unter

„Ausdruck“. Das „Tx()“ steht hierbei für eine Verschiebung in X-Richtung. Der Ausdruck in der Klammer gibt den Wert der Verschiebung an. Ein „Sx()“ gibt eine Skalierung in X-Richtung um den Faktor in der Klammer an.

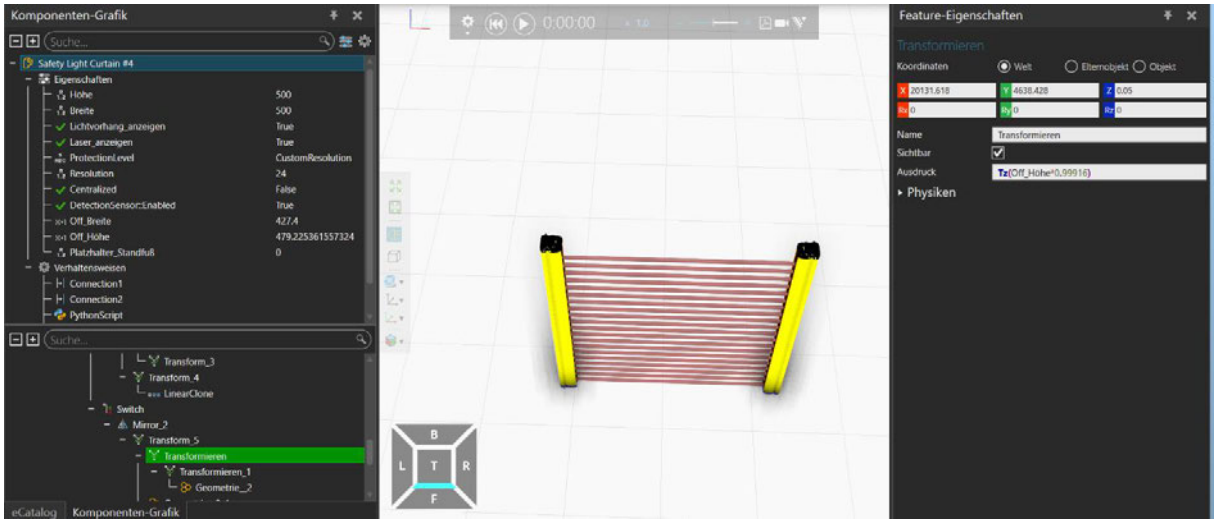


Abbildung 3.12: Transformationsvektor der Endkappe einer Datalogic-Lichtschranke

Über die einer Geometrie übergeordneten Transformationsvektoren können diese modular verschoben oder verändert werden. So ändert z. B. die „Resolution“ den Abstand zwischen den Laserstrahlen durch einen höheren Skalierungsfaktor. Der rechte Teil der Lichtschranke ist hierbei eine Spiegelung des linken Teils in der Mitte der Lichtschranke. Wie auch bei den Flächenscannern wird hier eine boolesche Variable hinzugefügt, die dafür das ist, die Laserstrahlen sichtbar oder unsichtbar zu machen, je nach Bedarf. Die Funktion der Lichtschranke ist durch das PythonSkript abgebildet, welches ein auswertbares Signal ausgibt, je nachdem, ob der Bereich frei ist oder nicht. Das fertige Modell wird folglich wieder in den eCatalog aufgenommen und beim nächsten Mal eine schnellere Verwendung ermöglicht, mehr dazu in Kapitel 3.1.8.

3.1.6 Modellierung des Förderbands

Das Förderband des FCPs ist die mechanische Schnittstelle zwischen dem Förderband des Kunden zum FCP. Dieses Förderband bietet im Vergleich zu vielen anderen einige Funktionen, die es ermöglichen, je nach Bedarf unterschiedlichste Pakete, Kisten etc. vom Band greifen zu können und dabei trotzdem noch kompatibel mit den verschiedensten Förderbandhöhen auf der Kundenseite zu bleiben. Des Weiteren ermöglicht das Förderband, die geförderten Produkte von der zum Förderband zeigenden Seite zu greifen.

Diese Funktionen werden durch drei verschiedene Verstellmöglichkeiten und einen speziellen Bandaufbau realisiert. Zum einen ist das Förderband ein Doppelgurtförderer, sodass die Pakete nur an den beiden Kanten links und rechts auf dem Förderband liegen und die Mitte frei ist für den Greifer, um die Pakete von unten zu greifen. Zum anderen kann das Förderband über eine Kurbel um 20° im Winkel angehoben werden, um eine kollisionsfreie Fahrt des Roboters zu gewährleisten. Damit aber trotzdem verschiedene Breiten von Produkten auf dem Förderband entlangfahren können, kann das Förderband über eine zweite Kurbel in der Breite

verstellt und somit auf jede Paketgröße zwischen 360 und 1000 mm Breite eingestellt werden. Die letzte Verstellmöglichkeit ist die Höhe des Bandes über eine dritte Kurbel. Dieses sorgt dafür, dass Produkte von jeder Förderbandhöhe zwischen 1000 und 1400 mm auf das FCP hinüberfahren können. Zusätzlich befindet sich am Ende des Förderbandes noch ein Sensor, der die abholbereiten Pakete erkennt.

Abbildung 3.13: Förderband des FCP

Diese Funktionen müssen in Visual Components ebenfalls umgesetzt werden, um jede mögliche Konfiguration beim Kunden abbilden zu können und so Fehler entdecken und vermeiden zu können. Dafür werden als Erstes wieder die CAD-Dateien des Förderbands importiert als „Eigenschaft“, da hier aus den einzelnen Geometrien ein neues Modell des Förderbandes zusammengebaut wird und das der schnellste Weg ist.

Die Abbildung 3.14 zeigt die Planung für die Umsetzung des Förderbands. Die einzelnen Blöcke zeigen dabei jeweils eine Verbindung an. Blöcke, die Links stehen, sind denen, die weiter rechts stehen, übergeordnet. So sind alle Verbindungen z. B. mit dem starren Unterteil verbunden, was bedeutet, dass sich diese zwar auch einzeln bewegen können, allerdings den Bewegungen des starren Unterteils folgen. „Starrer Unterteil“ heißt hierbei nicht, dass das Förderband nicht ortsveränderlich ist, sondern dass sich innerhalb der Baugruppe des Förderbands dieses Teil sich nicht bewegt. Im gleichen Prinzip werden alle Verbindungen auf der rechten Seite nach oben gefahren, sobald die Höhenverstellung hochgefahren wird. Dieses Diagramm dient dabei der Übersichtlichkeit beim Erstellen des Modells.

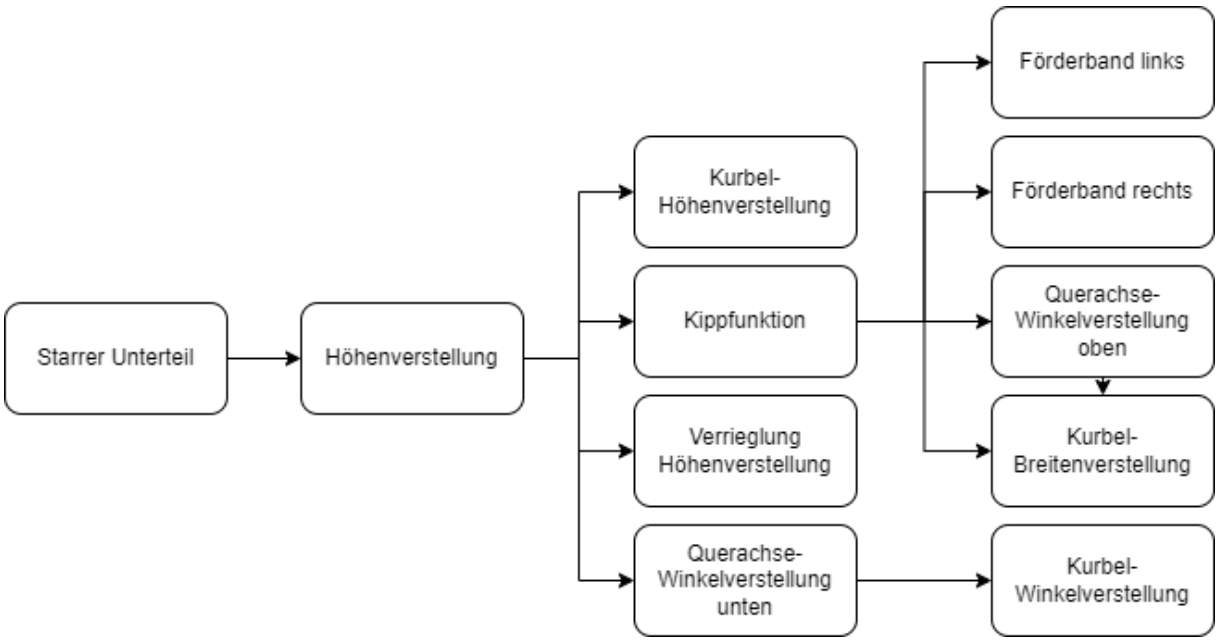


Abbildung 3.14: Struktur der Verbindungen des Förderbands in Visual Components

Zunächst wird die starre Gruppe des Förderbands, im Modell, zusammengebaut. Diese besteht aus einer Außenverkleidung mit Rädern, auf der der Rest des Förderbands aufgebaut ist. In der Mitte befindet sich ein Gestell, welches die Linearführung für die Höhenverstellung hält. Zusätzlich sitzt in der Mitte eine Verriegelung, die verhindert, dass sich das Förderband ungewollt absenken kann. Auf dieses Unterteil wird als Nächstes die Höhenverstellung aufgebaut.

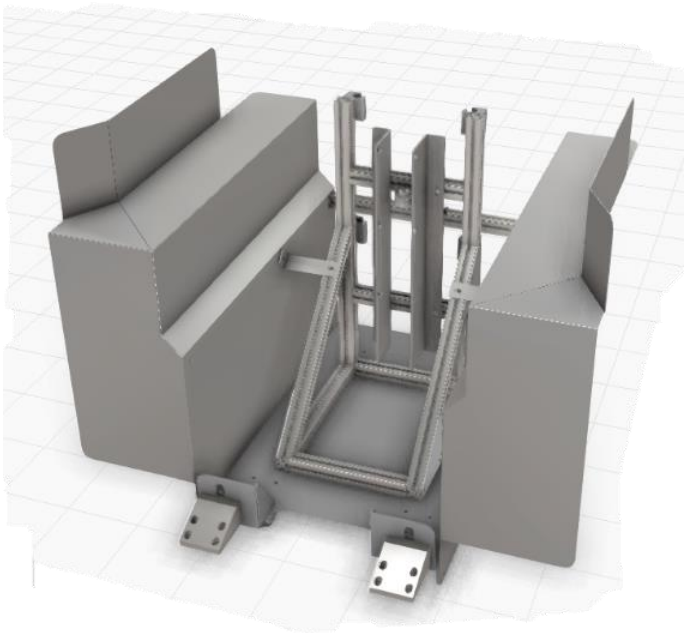


Abbildung 3.15: Starres Förderband Unterteil

An der Linearführung des starren Unterteils wird der Grundaufbau für die Höhenverstellung gebaut. Diese sorgt dafür, dass sich alle Bauteile, die dieser Verbindung untergeordnet werden, in der Höhe verstellen, wenn das Förderband verstellt wird. Die Einstellung für die Höhenverstellung wird auf „linear folgend“ gestellt. Das sorgt dafür, dass die Höhe von anderen Bauteilen abhängig gemacht werden kann. In diesem Fall die Kurbel der Höhenverstellung. Für jede Umdrehung der Kurbel erhöht sich der Aufbau um 400 mm. Das entspricht nicht der Realität, sorgt aber für eine einfache Bedienung innerhalb der Simulation.

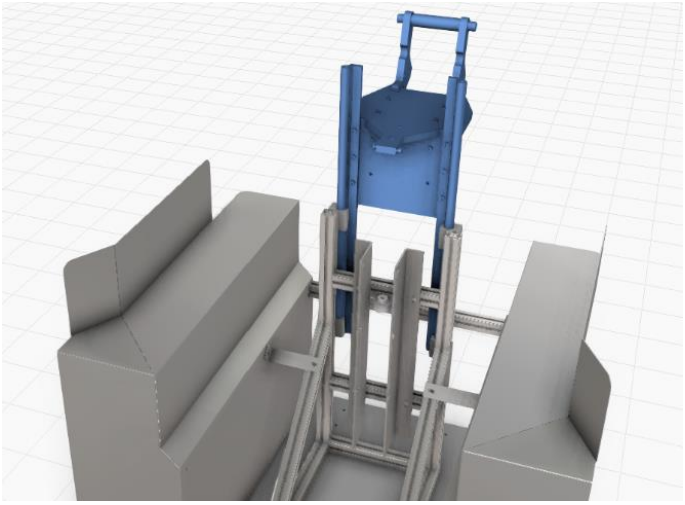


Abbildung 3.16: Förderbandunterteil der Höhenverstellung

Die nächste Verbindung ist die Querachse und Spindel für die Winkelverstellung. Diese sind auf „drehend folgend“ eingestellt und drehen sich um die untere Welle, in der die Spindel steckt. Diese folgt der Kurbel für die Winkelverstellung pro Umdrehung um 1,1°, sodass die obere Welle sich immer in der oberen Führung befindet, die im letzten Schritt hinzugefügt wird. Die Spindel, in der Mitte der Querachse, sorgt hierbei für die Verstellung des Winkels, wenn diese über eine Kurbel gedreht wird.

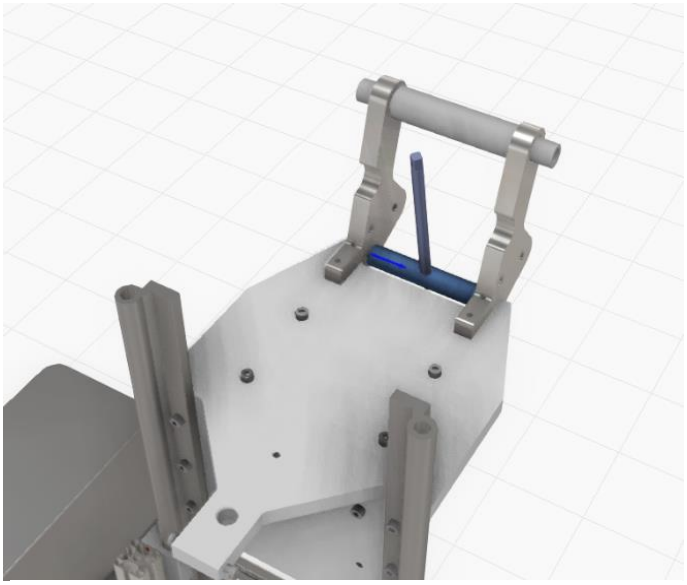


Abbildung 3.17: Winkelverstellung des Förderbands

Darauffolgt der Einbau der Verrieglung in das Modell. Diese kann in zwei Positionen verriegelt werden. Die Verriegelung wird auf „drehend“ gestellt, wobei sie um die Anschraubstelle dreht. Der Weg ist dabei allerdings auf 2,5° begrenzt, da nur Bewegungen innerhalb des Führungsbleches möglich sein sollen. Diese Bewegungen der Verrieglung sind im Modell allerdings rein optischer Natur, da es keine Kollisionen zwischen den einzelnen Bauteilen gibt, sodass auch die Verriegelung nicht bewegt werden muss, um die Höhe einzustellen.

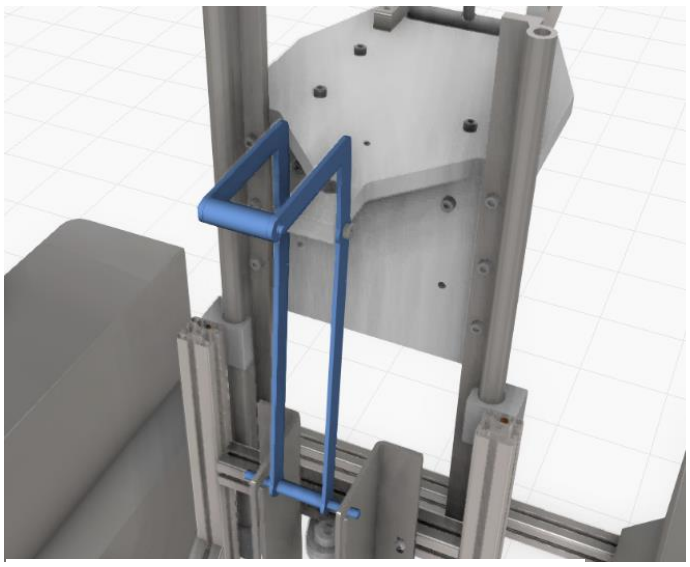


Abbildung 3.18: Förderband Höhenverriegelung

Danach wird die Spindel für die Höhenverstellung ins Modell eingepflegt. Von dieser Spindel sind die anderen Verbindungen abhängig, heißt, wenn sich die Spindel dreht, werden, bis auf das starre Unterteil, die andere Verbindung im Faktor 40 mm/U nach oben respektive nach unten gefahren. Da die Spindel der Verbindung „Höhenverstellung“ untergeordnet ist, verändert sich auch ihre eigene Höhe um 40 mm/U.

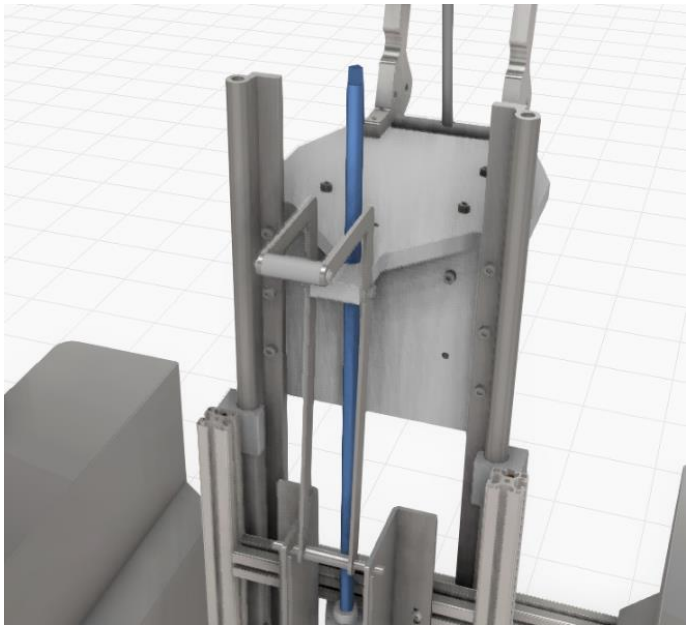


Abbildung 3.19: Spindel der Höhenverstellung

Anschließend werden die beiden Förderbänder und die Verstelleinheit für die Breite gleichzeitig auf das Förderband montiert. Dieses Bauteil ist die Verbindung „Kippfunktion“ mit allen Unterverbindungen, wobei die beiden Förderbänder der Kurbel für die Breitenverstellung folgen, mit dem Unterschied das, dass linke Band der Bewegung gleichgerichtet und das rechte dem entgegengerichtet folgt, sodass die beiden Bänder weiter zusammen oder weiter auseinanderfahren können.

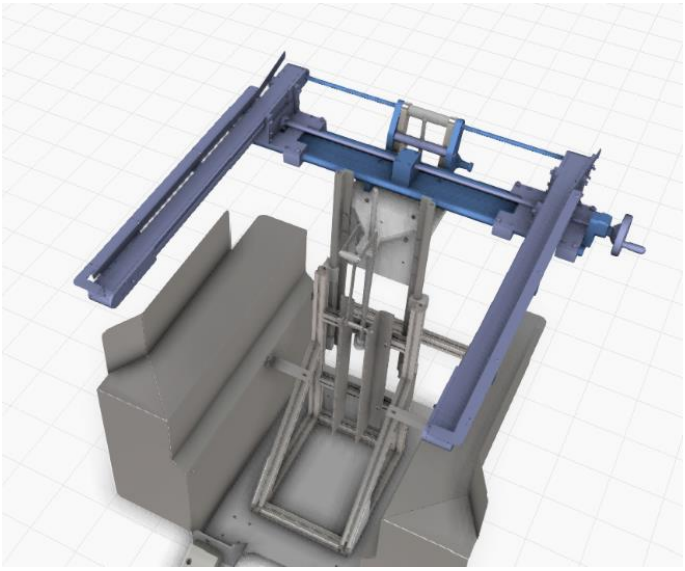


Abbildung 3.20: Obere Teil des Förderbands

Das Förderband kann nun mit dem „Interagieren-Button“ verstellt werden oder durch die Eingabe von verschiedenen Werten in die Höhen-, Winkel- oder Breitenverstellung. Zusätzlich können diese Daten über eine andere Komponente angesteuert oder genauer gesagt verändert werden.

Um Pakete in der Simulation über dieses Band fördern zu können, muss allerdings das Förderband noch als ein solches konfiguriert werden. Dafür wird, untergeordnet zur Verbindung „Kippfunktion“ eine weitere Verbindung, mithilfe des Assistenten „Förderer“ erstellt. Dieser Assistent erstellt dabei zwei „Frames“, einen Start- und eine Endframe. Zwischen diesen zwei Frames werden, die zu fördernde Produkte linear vom Start zum Ende befördert, des Weiteren wird ein Interface am Bandanfang und am Bandende erstellt. An diesem Interface kann ein Feeder angeschlossen werden, der in einem festgelegten Intervall Produkt, z. B. Pakete, auf das Förderband legt. Da am Förderbandende die Pakete verschwinden, wenn nichts am Ausgangsinterface angeschlossen ist, wird mit dem PnP-Tool ein „End-Block“ an das Förderband angeschlossen. Dieser End-Block stoppt alle Pakete an seiner Position. Der Endblock wird dabei so weit verschoben, bis die Pakete am Ende des Förderbands angekommen sind. Die Geometrien des End-Blocks werden daraufhin ausgeblendet.

3.1.7 Modellierung der Linearachse

Die Linearachse des FCPs sorgt dafür, dass der UR10e Pakete bis zu einer Höhe von 3 m auf Paletten stapeln kann. Auf dieser Linearachse ist der Roboter montiert, sodass dieser sowohl

die unteren als auch die oberen Paketschichten erreichen kann. Diese Achse hat dabei einen Verfahrweg von 2,5 m und eine Gesamtlänge von 3,15 m.

In Visual Components wird, wie bei der Erstellung der anderen Modelle, zunächst die CAD-Datei aus dem Modell in eine neue einzelne Datei geladen. Danach werden die Geometrien unter „Modellierung → Explodieren“ in ihre Einzelkomponenten zerlegt. Anschließend wird die Halterung des URs, genauso wie die Linearachse, wieder zu jeweils einem Teil verbunden, sodass der sich bewegende und der starre Teil der Achse vereinzelt sind.

Die Halterung für den UR wird hierbei in einer eigenen Verbindung gespeichert. Diese Verbindung wird auf „linear“ gestellt und die Bewegungsrichtung in Richtung des oberen Endes der Linearachse festgelegt. Als Nächstes müssen die obere und untere Bewegungsgrenze auf 0 mm und 2500 mm festgelegt werden. Die Linearachse kann sich jetzt zwar schon bewegen, allerdings nur ohne Roboter, da es kein „Mountingpoint“ gibt.

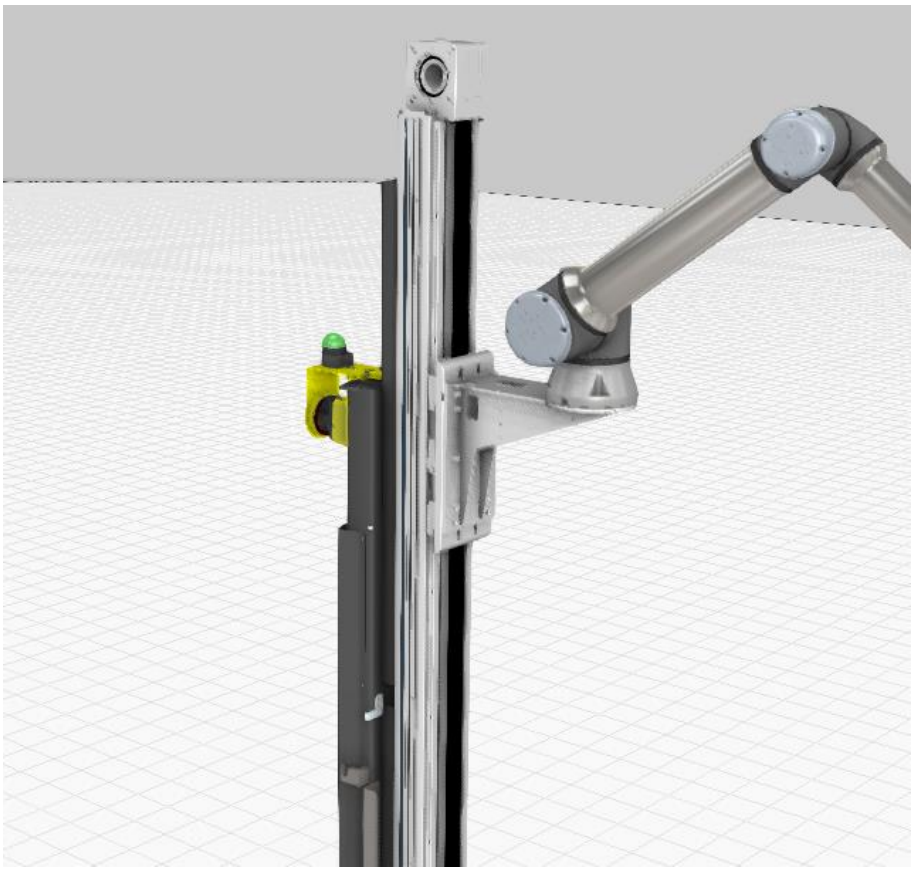


Abbildung 3.21: Linearachse des FCPs mit UR10e

In Visual Components lässt sich eine Linearführung für einen Roboter mit einem Assistenten leicht erstellen. Unter „Modellierung → Assistenten → Positionierer“ kann dieser gestartet werden. Als Nächstes wird die Achse ausgewählt und der Assistent ausgeführt. Dieser erzeugt im Modellbaum der Linearachse ein „RootFrame“, welcher mit dem „Verschieben-Tool“ auf die Mitte der Roboterhalterung platziert wird. Unter dem Menüpunkt „Start → PnP“ kann der Roboter auf diesen Punkt verschoben werden, an welchem er dann einrastet. Die Linearachse bewegt nun bei Bewegungen den Roboter mit sich und der Roboter kann, so wie andere Komponenten auch, in seinem Programm diese Achse ansteuern oder bewegen.

Die funktionierende Linearachse kann in den eCatalog aufgenommen werden, von wo sie in das Modell des FCP integriert werden kann. Wie genau die Aufnahme in den eCatalog funktioniert, ist in Kapitel 3.1.8 zu finden.

3.1.8 Modelle in den eCatalog aufnehmen

Um die erstellen Modell häufiger, schnell verwenden zu können wurden diese in den lokalen eCatalog von Visual Components aufgenommen, wodrum es in diesem Kapitel geht. Der eCatalog ist hierbei nicht zwingend für die Erstellung des digitalen Zwillings notwendig allerdings erleichtert es den Prozess der Erstellung.

Innerhalb des eCatalogs kann unter „Sammlung → + → Quellen bearbeiten → Quelle hinzufügen“ ein neuer Dateipfad ausgewählt werden. Aus diesem Dateipfad werden alle enthaltenen Komponenten geladen. Dafür müssen zunächst alle erstellten Modelle als VCMX-Datei innerhalb des angegebenen Pfads gespeichert werden.

Beim Speichern eines Modells können noch weitere Informationen angegeben werden, die folglich im eCatalog angezeigt werden, zu sehen in Abbildung 3.22.

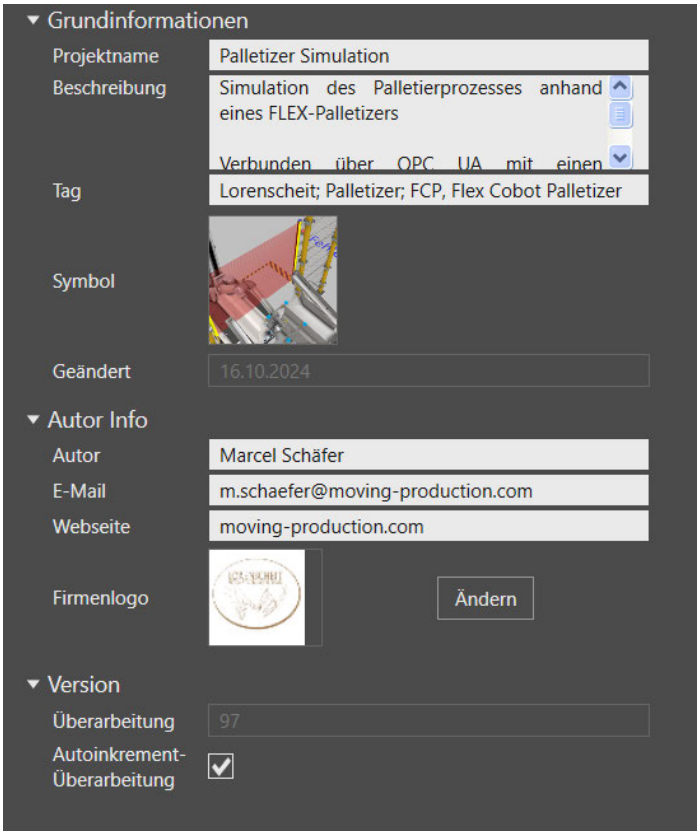


Abbildung 3.22: Speichern erstellter Modelle

Als erstes wird der Projektname angeben, was gleichzeitig der Anzeigename innerhalb des eCatalogs ist. Unter dem Punkt „Beschreibung“ kann eine Erklärung zur Verwendung des Modells gemacht werden. Diese Beschreibung wird auch im eCatalog angezeigt. Unter dem Punkt „Tag“ können Schlagwörter festgelegt werden, unter denen das jeweilige Modell im eCatalog zusätzlich gefunden werden. Das Symbol ist die Vorschau, die auch im eCatalog

angezeigt wird und wird anhand des Bildausschnitts im Moment des Speicherns festgelegt. Unter „Autor Info“ können Informationen zum Modellersteller hinterlegt werden. Zuletzt befindet sich noch unten die aktuelle Versionsnummer des jeweiligen Modells.

Während des Speichervorgangs ist darauf zu achten, dass alle erstellten Produkte und Serververbindungen innerhalb des Modells auch im eCatalog mitgespeichert werden und beim Importieren in andere Modelle wieder geladen werden. So kann es passieren das ungewollt Servereinstellungen oder erstellte Produkt mit importiert werden. Innerhalb des eCatalogs können Modelle mit der Ordnerstruktur organisiert bzw. kategorisiert werden. Ist das Modell im eCatalog kann es per „Drag and Drop“ in ein anderes Modell importiert werden.

4 Integration des Roboters und der SPS

In diesem Kapitel wird die, im letzten Kapitel erstellte Simulation, der Roboter und die SPS eingebunden, dafür muss die Kommunikation zwischen den einzelnen Teilnehmern hergestellt werden. Die Kommunikation zwischen der Codesys-SPS und Visual Components läuft über OPC UA.

4.1 Codesys

Als Erstes muss die OPC UA-Schnittstelle von der SPS bereitgestellt werden. Da das SPS-Programm, welches auf dem FCP zum Einsatz kommt, getestet werden soll und nicht eine veränderte Version, gilt es nur die nötigsten Änderungen vorzunehmen. Dafür wird eine Liste mit den benötigten Daten erstellt. Diese leitet sich aus denen im SPS-Programm eingelesenen oder gesendeten Signalen ab.

Tabelle 1: Variablenaustausch zwischen SPS und Visual Components

Variablenaustausch zwischen SPS und Visual Components	
Visual Components zur SPS	**SPS zu Visual Components**
Palettenerkennung links	Toggle On/Off Zuführband
Palettenerkennung rechts	UR-Remote-On
Bauteilerkennung am Greifer	UR-Remote-Off
Vakuumerkennung am Greifer	Ampel grün
Zuführband Produkt 1 auf Entnahmeposition	Ampel gelb
Zuführband Produkt 2 auf Entnahmeposition	Ampel rot
	Gripper off
	Gripper on
	Roboterachse J1-J6 (6 Variablen)
	Linearachse J1

Innerhalb des SPS-Programms wird jetzt überprüft, über welche Schnittstelle die Variablen gesendet und empfangen werden. Im Folgenden wird darauf eingegangen, welche Dinge mit Codesys im SPS-Programm angepasst wurden, um die Variablen auslesen und senden zu können.

Um den ständigen Dateiaustausch mit der Simulation nicht auf Kosten der Rechenleistung des Raspberry zu machen, wird ein zweiter Raspberry zwischen Visual Components und den ersten Raspberry geschaltet. Zum einen rechnet dieser die Position der Linearachse aus zum anderen verwaltet er den Datenaustausch. Eine Übersicht des gesamten Datenaustausches ist in Abbildung 4.1 zu sehen.

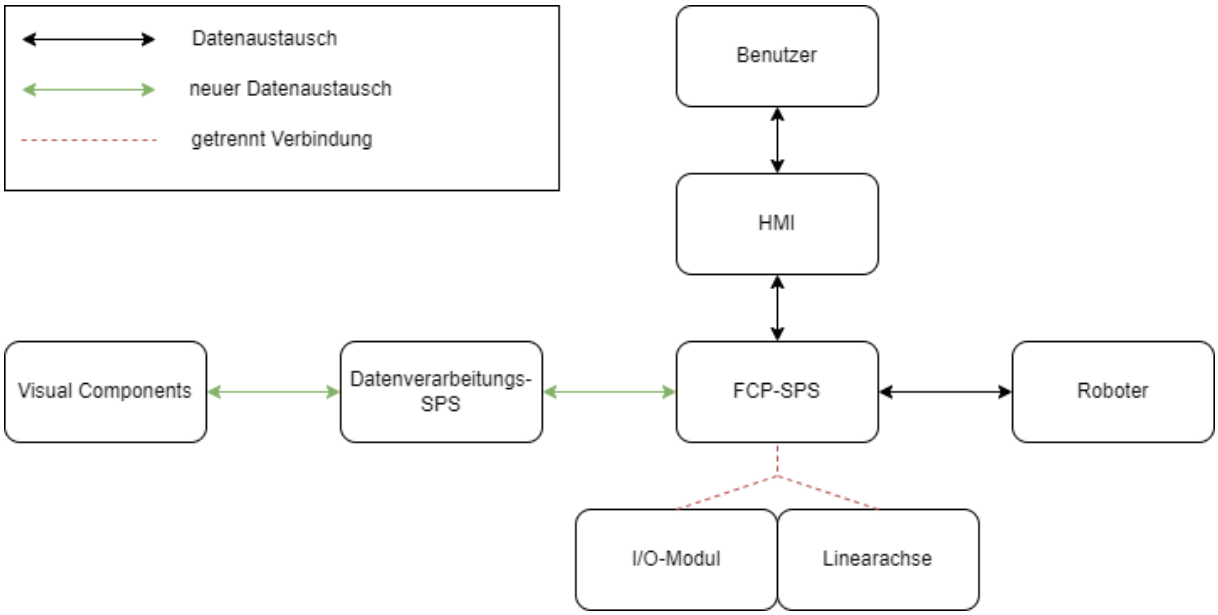


Abbildung 4.1: Übersicht des Datenaustausches im FCP

Zunächst wird dafür der zweite Raspberry mit dem ersten über Profinet verbunden, zu sehen in Abbildung 4.2. Dafür wird in Codesys, im SPS-Programm, ein neues Gerät hinzugefügt und mit einer IP-Adresse konfiguriert.

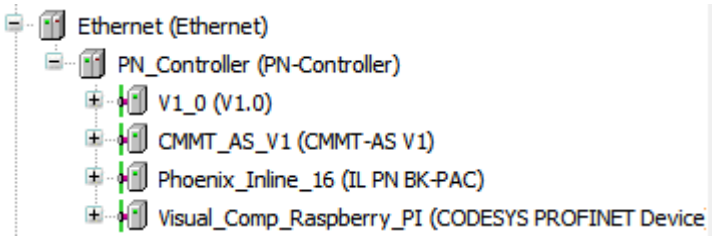


Abbildung 4.2: Einbindung des zweiten Raspberry in Codesys

Neben dem eben eingebunden Raspberry sind noch drei weitere Geräte über Ethernet mit dem ersten Raspberry verbunden. „V1_0“ ist hierbei der UR10e, der weiterhin vorhanden ist, wenn der Ablauf des FCP simuliert werden soll. Der reale UR10e fährt also die Bewegungen ab, die daraufhin in die Simulation übertragen werden. Das soll zum einen die Abweichungen zur Simulation klein halten, größtenteils ermöglicht das aber die Anpassung des

Roboterprogramms direkt am Roboter, die somit wieder mit dem digitalen Zwilling getestet werden kann.

„CMMT_AS_V1“ bezeichnet die Linearachse und „Phoenix_Inline_16“ das I/O-Modul. Diese beiden Komponenten werden kontinuierlich auf Verbindung mit dem Raspberry geprüft, sodass das Programm bei Verbindungsverlust gestoppt wird. Da diese Komponenten bei der Simulation des Prozesses nicht vorhanden sind, muss im SPS-Programm an dieser Stelle eine Änderung vorgenommen werden. Zu sehen sind diese Änderungen, Beispielhaft für das I/O-Modul, in Abbildung 4.3.

```
1  // Automatisches Hardwaremapping
2  IF Phoenix_inline_16.xRunning AND Visual_Components.simulation_enable = 0 THEN // 1x IL PN BK-PAC 16er Modules
3  input[0]:= Phoenix_inline_16_inputs[0];
4  input[1]:= Phoenix_inline_16_inputs[1];
5  Phoenix_inline_16_outputs[0]:=output[0];
6  Phoenix_inline_16_outputs[1]:=output[1];
7  fehler:=FALSE;
8  ELSIF Visual_Components.simulation_enable = 1 THEN
9      input[0].0 := Visual_Components.Palletenerkennung_links;
10     input[0].1 := Visual_Components.Palletenerkennung_rechts;
11     input[0].2 := Visual_Components.Vakuumerkennung_am_Greifer;
12     input[0].3 := Visual_Components.Bauteilerkennung_am_Greifer;
13     input[0].4 := Visual_Components.Zufuehrband_Teil_auf_Entnahmeposition;
14     input[0].5 := Visual_Components.Zufuehrband_Teil_auf_Bandende;
15
16     Visual_Components.Quittierung_Sicherheitsscanner := output[0].0;
17     Visual_Components.Gripper_on := output[0].1;
18     Visual_Components.Gripper_off := output[0].2;
19     Visual_Components.Ampel_rot := output[0].3;
20     Visual_Components.Ampel_gelb := output[0].5;
21     Visual_Components.Ampel_gruen := output[0].4;
22     Visual_Components.UR_Remote_On := output[0].7;
23     Visual_Components.UR_Remote_Off := output[0].6;
24     Visual_Components.Zufuehrband_Toggle := output[1].0;
25     fehler:=FALSE;
26 ELSE
27 fehler:=TRUE;
28 END_IF
```

Abbildung 4.3: Änderung im SPS-Programm

Der zweite Raspberry sendet, wenn er mit dem ersten verbunden ist, diesem eine boolesche Variable, sodass in dem Programm des ersten Raspberry die Überprüfung auf Verbindung deaktiviert und gleichzeitig die Zuweisung der Variablen geändert wird. Der erste Raspberry sendet und empfängt demnach keine Daten mehr an die Achse, sondern sendet und empfängt das Gleiche vom zweiten Raspberry. Gleichzeitig wurde in einem anderen Teil des Codes die Fehlermeldung „Achse nicht verbunden“ deaktiviert, wenn der zweite Raspberry verbunden ist.

Eine weitere Änderung, die noch gemacht werden muss, ist die Berechnung der Achsdaten des UR10e. Visual Components soll diese Daten verwenden, um die Darstellung des Roboters korrekt abzubilden. Dafür werden in der SPS die Daten in Form von Bytes vom Roboter abgefragt. Da Visual Components diese Bytes allerdings nicht in Achswinkel umrechnen kann, müssen die Bytes vorher von Codesys zu Integer umgerechnet werden.

Die Umsetzung dieser Umrechnung kann im Anhang unter Abschnitt B nachverfolgt werden. Die Daten jedes einzelnen Achswinkels kommen jeweils als vier einzelne Bits an. Zunächst werden dafür alle Bytes eines Achsgelenks hintereinander in eine neue Variable geschrieben. Zu dieser Variable wird ein Pointer erstellt, was dafür sorgt, dass neue Werte innerhalb der

Bytes sofort neu umgerechnet werden können. Zuletzt wird das Ganze von Bogenmaß zu Grad umgerechnet. Dieser Vorgang wird für jedes der sechs Gelenke wiederholt. Die neue Variable kann von Visual Components eingelesen und verarbeitet werden. Eine neue Berechnung der Achswinkel passiert hierbei einmal pro Zykluszeit, was grob einer Zeit von 15 ms entspricht.

Die Achswinkel werden zusammen mit den anderen Variablen an den zweiten Raspberry gesendet oder empfangen.

Ein weiteres Problem, das beim Simulieren des FCP auftritt, welches an der realen Anlage nicht auftritt, ist die Berechnung der Achsposition. An der realen Anlage wird der Linearachse eine Zielposition gesendet und die Achse verfährt anschließend auf die Zielposition. Zudem meldet die Achse permanent ihre aktuelle Position an Codesys zurück. In Visual Components gibt es auch die Möglichkeit, die Achse auf einen bestimmten Wert fahren zu lassen und das sogar mit den realen Geschwindigkeits- und Beschleunigungswerten. Allerdings muss die reale Linearachse bei Unterbrechung der Lichtschranke wieder neu initialisiert werden. Das dauert jedes Mal drei Sekunden, in denen die Achse nicht fährt. Da die Lichtschranke aber nicht vor jeder Fahrt unterbrochen wird, wirkt auch die in Visual Components einstellbare Verzögerungszeit dem nicht entgegen.

Daher wird das Ausrechnen der Position von dem zweiten Raspberry übernommen und die IST-Position an Visual Components übertragen. Dafür wurden zunächst alle Daten zur Bewegung der Linearachse aus dem Datenblatt dieser gelesen und ins Programm als Konstanten angelegt. Zu diesen Konstanten zählen die Beschleunigung, Geschwindigkeit und die Bremsbeschleunigung.

Ein Ausschnitt des folgenden Codes ist im Anhang unter Abschnitt B zu finden. Wird eine neue Soll-Position an den Raspberry gesendet, wird über einen Timer jede Sekunde die neue Position berechnet. Außerdem wird vorrausschauend berechnet, wann die Achse mit der aktuellen Geschwindigkeit anfangen muss zu bremsen, um rechtzeitig zum Stehen zu kommen.

Diese Art der Berechnung ist nur eine Annäherung an den realen Bewegungsablauf, da die Geschwindigkeit sich in der Berechnung nicht kontinuierlich ändert, sondern nur bei Diskreten Zeitschritten berechnet wird.

Zu guter Letzt werden die Variablen, die für die Simulation benötigt werden, per OPC UA bereitgestellt, zu sehen in Abbildung 4.4. Dafür werden in Codesys unter „Symbolkonfiguration“ alle Variablen, die über OPC UA einsehbar sein sollen, ausgewählt. Des Weiteren kann festgelegt werden, ob diese Variablen geschrieben und/ oder gelesen werden können.

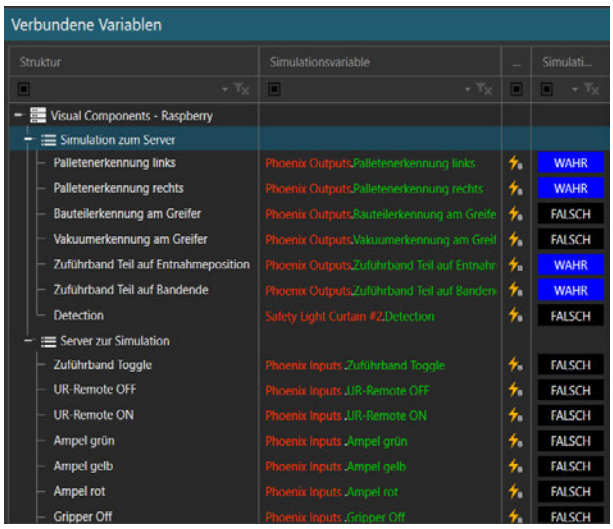


Abbildung 4.4: Variablen in Visual Components

4.2 Visual Components

Innerhalb von Visual Components muss zuerst die SPS per OPC UA verbunden werden. Danach müssen die von Codesys bereitgestellten Variablen mit den richtigen Komponenten verknüpft werden.

In Visual Components muss unter „Konnektivität“ ein OPC UA-Server hinzugefügt werden. Zum Verbinden mit Codesys wird die Serveradresse mit der IP-Adresse des Raspberrys gefüllt. Untergeordnet zu diesem Server können zwei Variablenlisten hinzugefügt werden, einerseits „Simulation zum Server“ und andererseits „Server zu Simulation“. Diese Listen entscheiden, ob Variablen aus der Sicht von Visual Components gelesen oder geschrieben werden. Im nächsten Schritt können alle von Codesys bereitgestellten Variablen mit den Signalen oder Achswinkel von Komponenten verknüpft werden. So werden z.B. die berechneten Achswinkel mit den jeweiligen Achsen vom simulierten Roboter lesend verbunden. Wird anschließend die Simulation gestartet, werden diese Variablen alle 50ms ausgelesen und visualisiert bzw. verarbeitet.

4.3 Universal Robots – Polyscope

Der UR10e muss, bevor die Simulation gestartet werden kann, eingerichtet werden. Dabei gibt es den Vorteil, dass dieser wie in der realen Anlage, mit dem ersten Raspberry kommuniziert und daher die gleichen Einstellungen wie bei der realen Anlage verwendet werden können.

Als erstes wird dafür das Roboterprogramm auf den UR10e hochgeladen. Die Einrichtung der IP-Adresse und die Verbindung zum Raspberry ist in diesem bereits hinterlegt und muss somit nicht geändert werden. Da allerdings die Bewegungen vom realen Roboter ausgelesen und in der Simulation angezeigt werden sollen, ist es wichtig, dass dieser die gleichen Freiheiten im Raum hat wie der simulierte Roboter, um überall hinzufahren, wo der Roboter auf der realen Anlage auch hinfahren würde.

5 Vergleich mit FLEX COBOT PALLETIZER

Im folgenden Kapitel geht es um den Vergleich der Simulation zur realen Welt. Zu Beginn werden die auftretenen Probleme, die daraus resultierenden Unterschiede und ihre Folgen für die Simulation erläutert. Zusätzlich wird am Ende des Kapitels noch die Nutzbarkeit des digitalen Zwillings und die fertige Simulation im Bezug auf die anfangs gesteckten Ziele bewertet.

5.1 Probleme und resultierende Unterschiede

Zwischen dem simulierten und realen FCP gibt es trotz vieler Gemeinsamkeiten auch Unterschiede. Der erste große Unterschied ist die Umsetzung der Flächensicherheitsscanner. Die Funktion dieser ist in Visual Components zwar umgesetzt, allerdings gestaltete sich die Einbindung an den realen Roboter schwieriger als eingangs gedacht.

An der realen Anlage stoppt der Roboter und die Linearachse sobald einer der drei Sicherheitsbereiche verletzt wird. Bedienende der Anlage müssen an der HMI quittieren, sodass sich keiner mehr im Sicherheitsbereich befindet. Daraufhin fährt der FCP mit der Palettierung fort. Wird ein Scannerbereich betreten, sendet der dazugehörige Flächensicherheitsscanner ein Signal an ein Sicherheitsrelais und dieses Relais wiederum ein zweikanaliges Signal an den Roboter und die Achse, welche folglich anhalten.

Abbildung 5.1: Flächensicherheitsscanner in Visual Components

Das größte Hindernis bei der Umsetzung dieser Funktion in der Simulation ist die Hardwarelimitation. Visual Components könnte zwar die Signale der simulierten Scanner an den Raspberry senden, dieser hat jedoch keinen 24V-Output. Der UR10e braucht allerdings dieses 24V-Signal zum Detektieren der Eingänge. Dieses Hindernis könnte mit zusätzlicher Hardware umgangen werden, allerdings wurde sich aufgrund der Einfachheit des digitalen Zwillings gegen diese Funktion entschieden, nicht zuletzt da diese durch die Bediener der Anlage stark variieren können und damit auch die Taktzeit. Das hat zur Folge, dass bei vielen Unterbrechungen der Scanner an der realen Anlage die Abweichung der Taktzeit mehrere Sekunden betragen kann.

Ein weiteres Problem, welches beim Simulieren aufgetreten ist, ist die Verfahrzeit der Linearachse. Die Linearachse in der Simulation ist hierbei bedeutend schneller als die reale Linearachse. Die reale Achse braucht für eine Fahrt von ganz unten nach ganz oben 8,1 s, die simulierte Achse hingegen nur 4,5 s. Dieser Unterschied ist dadurch zu erklären, dass die im Datenblatt angegebenen Geschwindigkeits- und Beschleunigungswerte nur bei einer Leerfahrt erreicht werden können. Um dieses Problem zu beheben, wurden die maximale Geschwindigkeit der Linearachse über eine Distanz von einem Meter ermittelt und in der Simulation angepasst. Die Beschleunigung wurde folglich über den direkten Vergleich der Simulation mit der realen Anlage eingestellt.

Abbildung 5.2: Linearachse des FCP

Ein weiterer großer Unterschied zwischen der Simulation und der realen Anlage sind die Kollisionen und Verhaltensweisen von verschiedenen Gegenständen. In der Simulation ist es möglich den Gegenständen eine physikalische Berechnung hinzuzufügen, sodass diese mit anderen kollidieren können und von der Erdanziehung beeinflusst werden. Diese Einstellungen sorgen allerdings dafür, dass Kisten nicht vernünftig gestapelt werden können, da die Physiksimulation für mehrere Gegenstände, die präzise gestapelt werden müssen, nicht geeignet ist, da diese voneinander rutschen. Deshalb ist das Verhalten der Produkte ohne

Physiksimulation näher an der realen Anlage als das Verhalten mit Physiksimulation. Für einzelne Gegenstände ist das Physikmodell gut verwendbar, jedoch nicht für diesen Anwendungsfall, da mehrere zu simulierende Gegenstände aufeinandertreffen. Außerdem gibt es keine Möglichkeit einen Vakuumgreifer zu simulieren, was für die Darstellung der Anlage nötig wäre. Dadurch könnte geschaut werden, ob die Pakete an dem Greifer halten oder nicht.

5.2 Vergleich mit Erwartungen

In diesem Kapitel werden die anfangs genannten Erwartungen an den digitalen Zwilling überprüft und bewertet.

Als ersten Punkt der Anforderungen sollten die mechanischen Abmaße des FCP mit den Maßen in der Simulation übereinstimmen. Für die Überprüfung dieser Anforderungen wurden die Maße von verschiedenen wichtigen Punkten des realen FCP mit denen der Simulation verglichen.

Tabelle 2: Mechanische Abweichung von realer Anlage zur simulierten

Messpunkt 1	**Messpunkt 2**	**Abstand realer FCP**	**Abstand simulierter FCP**	**Abweichung in Prozent**
Roboterstandfuß	Förderbandanschlag	772 mm	769,2 mm	0,36 %
Förderbandfuß	Förderbandfläche	1000 mm	999,7 mm	0,03 %
Palettenplatz links	Palettenplatz rechts	1025 mm	1024 mm	0,09 %
FCP Standfuß	Linearachse Oberseite	3500 mm	3447,7 mm	1,5 %

Die größte Abweichung zwischen Simulation und realer Anlage sind 1,5 %, zwischen dem Standfuß des FCP und der Oberseite der Linearachse. Diese können zum einen durch die Höhenverstellbaren Standfüße des FCPs und zum anderen durch die Messabweichungen erklärt werden. Die Messabweichungen bei der Messung der Abstände ist hierbei größer zu bewerten als die 1,5 %, sodass die Abweichung einen nicht messbaren Wert beträgt. Da dieses CAD-Modell auch als Grundlage der Fertigungszeichnungen dient, sind diese Abweichungen nur minimal und in diesem Fall vernachlässigbar.

Der nächste Punkt der Anforderungen ist die Funktion aller Sensoren und Aktuatoren. Die Funktionen sind bei fast allen gegeben. Die Palettensensoren funktionieren, allerdings nicht über einen Sensor, der das Material der Palette in nahem Abstand detektiert, sondern über die Materialflusseinstellung in Visual Components, sodass die Sensoren ausgelöst werden, wenn der Materialfluss einen bestimmten Punkt erreicht. Für die Nutzbarkeit des digitalen Zwillings macht diese andere Funktionsweise keinen merklichen Unterschied. Allerdings kann

dadurch mit dem digitalen Zwilling die Funktionalität der realen Sensoren nicht überprüft werden.

Jedoch funktionieren die Flächensicherheitsscanner nicht im digitalen Zwilling. Der Roboter und die Linearachse werden somit nicht gestoppt, wenn z. B. eine neue Palette angeliefert wird. Dies kann sich auf die Taktzeit auswirken, allerdings kann der Prozess auch ohne diesen Aspekt simuliert werden. Da diese Unterbrechungen der Flächensicherheitsscanner, bis auf die Anlieferung und die Abholung der Paletten, nicht planbar sind, können dahingehend keine eindeutigen Aussagen bezüglich der Taktzeit getätigt werden.

Der nächste Punkt innerhalb der Anforderungen ist die Übereinstimmung der Bewegungen von Roboter und Linearachse. Um hier eine Aussage über die Bewegungen treffen zu können, wurden verschiedene Abläufe mit unterschiedlichen Roboter- und Achsenbewegungen simuliert und die Taktzeiten mit den realen Taktzeiten verglichen. Zusätzlich wird im Folgenden gleichzeitig der letzte Punkt der Anforderungsliste, die Taktzeitabweichung, überprüft und bewertet.

Verglichen wurden verschiedene Packmuster und Packhöhen von dem digitalen Zwilling mit den Taktzeiten der realen Anlage. In Tabelle 3 werden dafür die Taktzeiten für drei Tests aufgenommen und vergleichen.

- Test 1: Paketmaße LxBxH: 400x400x200, erste Schicht
- Test 2: Paketmaße LxBxH: 1000x600x425, erste Schicht
- Test 3: Paketmaße LxBxH: 1000x600x425, sechste Schicht

VC: Taktzeit in der Simulation durch Visual Components ermittelt

Real: Taktzeiten an der realen Anlage gemessen

Tabelle 3:Vergleich der Taktzeiten - Real zu Visual Components

Test 1 – VC	**Test 1 – Real**	**Test 2 – VC**	**Test 2 – Real**	**Test 3 – VC**	**Test 3 - Real**
28,2 s	27,6 s	34,0 s	33,3 s	35,2 s	34,6 s

Berechnung der prozentualen Abweichung der Taktzeit:

$$\Delta t_{Test1} = 28{,}2s - 27{,}6s = 0{,}6s$$

$$\frac{0{,}6s}{27{,}6s} \approx 0{,}022 \approx 2{,}2\%$$

$$\Delta t_{Test2} = 34{,}0s - 33{,}3s = 0{,}7s$$

$$\frac{0{,}7s}{33{,}3s} \approx 0{,}021 \approx 2{,}1\%$$

$$\Delta t_{Test3} = 35{,}2s - 34{,}6s = 0{,}6s$$

$$\frac{0{,}6s}{34{,}6s} \approx 0{,}017 \approx 1{,}7\%$$

In der Anforderungsliste wurde die Vorgabe einer Abweichung von kleiner als 5 % festgelegt. Bei diesen drei Tests wurde eine maximale Abweichung der Taktzeit von 2,2% gemessen, sodass diese Anforderung erfüllt wurde und der digitale Zwilling somit zur Vorhersage von Taktzeiten geeignet ist.

Der vierte Punkt in der Anforderungsliste ist die Nutzung vom Programmcode für Roboter und SPS, der auch auf der realen Anlage zum Einsatz kommt. Dieser Punkt wurde teilweise erfüllt. So wird für den digitalen Zwilling zwar das Roboterprogramm genauso wie es auch auf der realen Anlage zu finden ist, genutzt, allerdings musste das SPS-Programm angepasst werden. Hier wurden Ein- und Ausgänge des Raspberry neu zugewiesen und ein zweiter Raspberry mit der SPS verbunden. Zudem gibt es keine Überprüfung, ob Bauteile wie die Linearachse verbunden sind.

Diese Änderungen sind für die Simulation des FCP nicht ausschlaggebend und sind so ausgelegt, dass am grundlegenden Verhalten der Anlage nicht viel verändert wurde, trotz dessen muss Einem dieser Unterschied beim Simulieren bewusst sein.

6 Fazit

Abschließend lässt sich zu dem digitalen Zwilling sagen, dass nicht alle Anforderungen erfüllt wurden, jedoch ist es trotzdem möglich eine Simulation von gesamten Prozessablaufs zu erstellen und auch eine qualifizierte Aussage über die Taktzeit zu treffen. Probleme, die an der realen Anlage auftreten, können, mit einigen Einschränkungen, am digitalen Zwilling nachverfolgt werden und durch Ändern der Programmcodes von SPS und Roboter behoben werden. Der digitale Zwilling ermöglicht somit die Offlineprogrammierung des FCP, auch wenn zum aktuellen Zeitpunkt noch Hardware, wie die Raspberry und der UR10e benötigt werden.

7 Ausblick

Bei der Erstellung des digitalen Zwillings sind einige Aspekte aufgefallen, die verbesserungswürdig sind bzw. Änderungen, für die es noch keine geeignete Lösung gibt, die bei der Handhabung des digitalen Zwillings helfen würden. Um diese möglichen Verbesserungen geht es in diesem letzten Kapitel.

Der erste Punkt, der noch verbessert werden könnte, ist die Simulation der verwendeten Hardware. Aktuell werden ein UR10e und zwei Raspberry mit Spannungsversorgung benötigt um den digitalen Zwilling mit dem Robotercode laufen lassen zu können. Das bedeutet, dass der digitale Zwilling nicht zu jeder Zeit an jedem Ort gestartet und benutzt werden kann, sofern das Ziel ist, die realen Roboterbewegungen zu simulieren. Somit kann der Gesamtablauf noch nicht von jedem Programmierer individuell getestet werden. Hier wäre es praktisch den Robotercode und die Steuerungen am PC simulieren zu können. Für die beiden Steuerungen ist dies mit einer Soft-SPS bereits möglich. Beim Roboter muss die Umsetzbarkeit noch erforscht werden. Potenziell ist dies mit dem Universal Robots Programm „UR-Sim“ möglich, praktisch fehlt dafür noch die Erfahrung mit diesem Programm. Eine andere Möglichkeit wäre die Nutzung eines anderen Programms zu Erstellung des digitalen Zwillings, welches gleichzeitig die SPS und das Roboterprogramm simulieren kann.

Eine weitere mögliche Verbesserung wäre die Einbindung der Flächensicherheitsscanner und eine bessere Umsetzung der Palettensensoren. Für die Flächensicherheitsscanner gibt es die Möglichkeit, dies mit zusätzlicher Hardware umzusetzen. Bei den Palettensensoren könnte die Funktion wahrscheinlich mit einem Pythonskript umgesetzt werden, da Visual Components z. B. die Lichtschranken so realisiert.

Das Prinzip des digitalen Zwillings wird in Zukunft noch genauso auf andere Anlagen angewendet werden, um auch hier die Vorteile des digitalen Zwillings nutzen zu können.

Literaturverzeichnis

[1] Lorenscheit Automatisierungs-Technik: FLEX COBOT PALLETIZER, https://moving-production.com/lorenscheit-flex-series/flex-cobot-palletizer/, Abruf am 21.10.2024 14:00

[2] Lorenscheit Automatisierungs-Technik: Über Uns, https://moving-production.com/unternehmen/, Abruf am 21.10.2024 14:00

[3] Cadfem: Digitaler Zwilling schaut vorraus, https://www.cadfem.net/de/de/cadfem-informiert/media-center/cadfem-journal/digitaler-zwilling-im-wasserkraftwerk-fuer-praediktive-diagnosen.html, Abruf am 21.10.2024 14:01

[4] IBM: Was ist ein digitaler Zwilling?, https://www.ibm.com/de-de/topics/what-is-a-digital-twin, Abruf am 21.10.2024 14:01

[5] Frauenhofer IOSB: Digitale Zwillingssysteme – das Schlüsselkonzept für Industrie 4.0, https://www.iosb.fraunhofer.de/de/geschaeftsfelder/automatisierung-digitalisierung/anwendungsfelder/digitaler-zwilling.html, Abruf am 21.10.2024 14:01

[6] Yogini B. u.a. : Digital Twins, https://www.degruyter.com/document/doi/10.1515/9783110778861/html, S.20 Abschnitt 2.2

[7] Siemens: Digitale Zwillinge, https://www.siemens.com/de/de/unternehmen/stories/forschung-technologien/digitaler-zwilling/digital-twin.html, Abruf am 21.10.2024 14:02

[8] Darek F.: Digitaler Zwillinge Beispiele und Anwendungsfälle, https://www.onlogic.com/de/blog/digitale-zwillinge-beispiele-und-anwendungsfaelle, Abruf am 21.10.2024 14:02

[9] Peter A.: Was ist eigentlich ein digitaler Patient*innen-Zwilling?, https://www.siemens-healthineers.com/deu/perspectives/digital-patient-twin, Abruf am 21.10.2024 14:03

[10] Statistisches Bundesamt: G20 in Zahlen, https://www.destatis.de/DE/Themen/Laender-Regionen/Internationales/Thema/allgemeines-regionales/G20/G20.html?nn=625596#wirtschaft, Abruf am 21.10.2024 14:03

[11] Martin C.: Automatisierungsbranche verzeichnet starken Auftragszuwachs, https://www.vdi-nachrichten.com/technik/automation/automatisierungsbranche-verzeichnet-starken-auftragszuwachs/, Abruf am 21.10.2024 14:05

[12] Frauenhofer IOSB: Digitale Zwillingssysteme – das Schlüsselkonzept für Industrie 4.0, https://www.iosb.fraunhofer.de/de/geschaeftsfelder/automatisierung-digitalisierung/anwendungsfelder/digitaler-zwilling.html, Abruf am 21.10.2024 14:05

[13] Stefani T.: Digitaler Zwilling in der Logistik, https://www.knapp.com/wissen/blog/digitaler-zwilling-logistik-definition-vorteile-einsatzgebiete/, Abruf am 21.10.2024 14:06

[14] Visual Components: Über uns, https://www.visualcomponents.com/de/uber-uns/, Abruf am 21.10.2024 14:07

[15] Codesys: Unternehmensprofil, https://de.codesys.com/unternehmen.html, Abruf am 21.10.2024 14:07

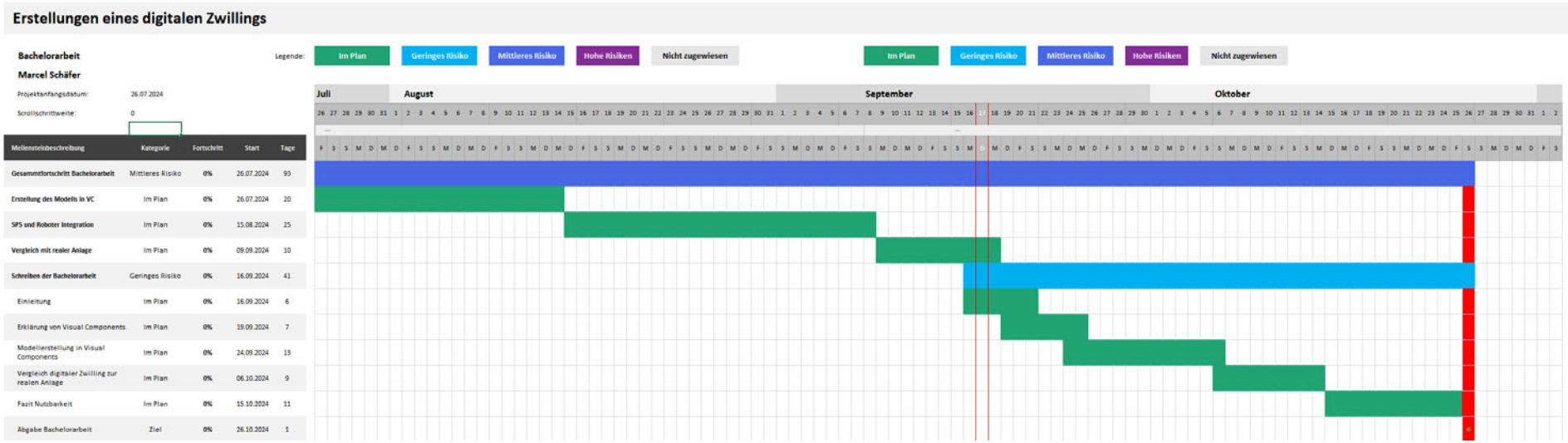
[16] Siemens: PROFINET – Echtzeit-Kommunikation im Feld, https://www.siemens.com/de/de/produkte/automatisierung/industrielle-kommunikation/profinet.html, Abruf am 21.10.2024 14:08

[17] Siemens: OPC UA – strukturierte Daten bis in die Cloud, https://www.siemens.com/de/de/produkte/automatisierung/industrielle-kommunikation/opc-ua.html, Abruf am 21.10.2024 14:09

[18] Raspberry Pi Foundadtion: We are Raspberry Pi. We make computers., https://www.raspberrypi.com/about/, Abruf am 21.10.2024 14:09

Anhang

Abschnitt A



Abschnitt B

```
PROGRAM VC_UR_Achsdaten
VAR CONSTANT
    pi : REAL := 3.14159265359;
END_VAR
VAR
    J0_Byte_1 : BYTE := 16#00;;
    J0_Byte_2 : BYTE;
    J0_Byte_3 : BYTE;
    J0_Byte_4 : BYTE;
    J0_Achswinkel_in_DEG : REAL;
    J0_Bin : DWORD;
    J0_Dez: REAL;
    Pt_J0:POINTER TO REAL;
```

```
J0_Bin := SHL (J0_Byte_1,24) + SHL (J0_Byte_2,16) + SHL(J0_Byte_3,8) + J0_Byte_4;
Pt_J0 := ADR(J0_Bin);
J0_Dez := Pt_J0^;
J0_Achswinkel_in_DEG := J0_Dez * 180/pi;

J1_Bin := SHL (J1_Byte_1,24) + SHL (J1_Byte_2,16) + SHL(J1_Byte_3,8) + J1_Byte_4;
Pt_J1 := ADR(J1_Bin);
J1_Dez := Pt_J1^;
J1_Achswinkel_in_DEG := J1_Dez * 180/pi;

J2_Bin := SHL (J2_Byte_1,24) + SHL (J2_Byte_2,16) + SHL(J2_Byte_3,8) + J2_Byte_4;
Pt_J2 := ADR(J2_Bin);
J2_Dez := Pt_J2^;
J2_Achswinkel_in_DEG := J2_Dez * 180/pi;

J3_Bin := SHL (J3_Byte_1,24) + SHL (J3_Byte_2,16) + SHL(J3_Byte_3,8) + J3_Byte_4;
Pt_J3 := ADR(J3_Bin);
J3_Dez := Pt_J3^;
J3_Achswinkel_in_DEG := J3_Dez * 180/pi;

J4_Bin := SHL (J4_Byte_1,24) + SHL (J4_Byte_2,16) + SHL(J4_Byte_3,8) + J4_Byte_4;
Pt_J4 := ADR(J4_Bin);
J4_Dez := Pt_J4^;
J4_Achswinkel_in_DEG := J4_Dez * 180/pi;

J5_Bin := SHL (J5_Byte_1,24) + SHL (J5_Byte_2,16) + SHL(J5_Byte_3,8) + J5_Byte_4;
Pt_J5 := ADR(J5_Bin);
J5_Dez := Pt_J5^;
J5_Achswinkel_in_DEG := J5_Dez * 180/pi;

J0_Achswinkel_in_DEG;
J1_Achswinkel_in_DEG;
J2_Achswinkel_in_DEG;
J3_Achswinkel_in_DEG;
J4_Achswinkel_in_DEG;
J5_Achswinkel_in_DEG;
```

Anhang

```
PROGRAM Linear_Achse
VAR CONSTANT
    beschleunigung_max : REAL := 8000; //8m/s^2    8000mm/s^2
    geschwindigkeit_max: REAL := 500; //0.5m/s  500mm/s
    bremsbeschleunigung_max : REAL := 8000; //8m/s^2    8000mm/s^2
END_VAR
VAR
```

```
IF NOT Sicherheitszaun THEN
    //Berechnung der Differenz zwischen Ist- und Sollwert
    IF Sollwert < Istwert THEN
        Diff_Soll_Ist := ABS(Istwert - Sollwert);
    ELSIF Istwert < Sollwert THEN
        Diff_Soll_Ist := ABS(Sollwert - Istwert);
    ELSIF Sollwert = Istwert THEN
        Diff_Soll_Ist := 0;
    END_IF

    //Geschwindigkeit und Beschleunigung pro Takt
    Timer.Taktzeit := Taktzeit/2;
    REAL_Taktzeit := TIME_TO_REAL(Taktzeit)/1000;
    Takt_beschl := beschleunigung_max * REAL_Taktzeit;
    Takt_strecke := akt_geschwindigkeit * REAL_Taktzeit;

    //Zeit berechnen die zum Bremsen benötigt wird
    zeit_bis_stopp := akt_geschwindigkeit / bremsbeschleunigung_max;

    //Berechnung wann angefangen werden muss zu bremsen
    IF akt_geschwindigkeit * zeit_bis_stopp >= Diff_Soll_Ist AND Diff_Soll_Ist > 0 THEN //Wenn die R
        modus := 2;
    ELSIF akt_geschwindigkeit < geschwindigkeit_max AND Diff_Soll_Ist > 0 THEN
        modus := 1;
    ELSIF Sollwert = Istwert THEN
        modus := 3;
    ELSE
        modus := 0;
    END_IF

    //Umsetzung der Modi (Modus 0: Geschwindigkeit halten, Modus 1: Beschleunigen, Modus 2: Bremsen)
    IF Timer.Takt AND NOT Takt_Merker THEN
        Takt_Merker := TRUE;
        CASE modus OF
            1:  IF akt_geschwindigkeit <> geschwindigkeit_max THEN
                    IF akt_geschwindigkeit + Takt_beschl > geschwindigkeit_max THEN
                        akt_geschwindigkeit := geschwindigkeit_max;
                    ELSE
                        akt_geschwindigkeit := akt_geschwindigkeit + Takt_beschl;
                    END_IF
                END_IF
            2:  IF akt_geschwindigkeit > 0 THEN
                    IF  akt geschwindigkeit - Takt beschl < 0 THEN
```

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Schäfer

Vorname: Marcel

dass ich die vorliegende Bachelorarbeit ☐ bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Erstellung und Evaluation eines digitalen Zwillings für die Offline-Programmierung einer Palettier-Anlage mithilfe von Visual Components und Codesys

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der -bitte auswählen- ☐ ist erfolgt durch:

[redacted]

Harlingen (Ort) 21.10.2024 (Datum)