FACHHOCHSCHULE
WIENER NEUSTADT
Austrian Network for Higher Education

Aufbau einer Delta-Roboterzelle für universelle Anwendungen

Bachelorarbeit

Eingereicht von: **Mustafa Algan**

Matrikelnummer: **51819221**

am **Fachhochschul-Bachelorstudiengang Mechatronik/Mikrosystemtechnik**

Begutachter: **Dipl.-Ing. Dr. Alexander Nemecek**

Wiener Neustadt, 21.Mai 2020

EIDESSTATTLICHE ERKLÄRUNG

Titel der Bachelorarbeit 2:

Aufbau einer Delta-Roboterzelle für universelle Anwendungen

Eingereicht von:	Mustafa ALGAN
Matrikelnummer:	51819221
am:	Fachhochschul Bachelorstudiengang
	Mechatronik/Mikrosystemtechnik
	Micro- and Nanoengineering
Begutachterin/Begutachter:	Dipl.-Ing. Dr. Alexander Nemecek
Wiener Neustadt am:	Mai 2020

Ich erkläre an Eides statt, dass diese Arbeit ausschließlich von mir selbst verfasst wurde und ich diese Arbeit nicht zuvor an einer anderen Bildungseinrichtung zum Zwecke der Erlangung eines akademischen Grades vorgelegt habe.

Insbesondere wurden Beiträge anderer Personen entsprechend kenntlich gemacht sowie die in dieser Arbeit verwendeten Daten entsprechend der dargestellten Verfahren gewonnen und richtig wiedergegeben.

Wiener Neustadt,_____

Datum Unterschrift

Kurzzusammenfassung:

Diese Bachelorarbeit behandelt den Aufbau einer Delta-Roboterzelle und Simulation einer Pick-and-Place Anwendung. Der zu montierende Delta-Roboter wird von der FH Wiener Neustadt bereitgestellt. Die Roboterzelle wurde aus Aluminium-Profilen und einem Stahl-Rahmen zusammengebaut. Zusätzlich wurde der Aufbau mit Peripheriegeräten ausgestattet, damit verschiedene Anwendungen getestet werden können. Zum Testen wurde eine Simulation von dem Aufbau in der Robotersoftware RobotStudio® durchgeführt. Für die Simulation eignete sich eine Pick-and-Place Anwendung, sodass eine Taktzeitanalyse durchgeführt werden konnte. Danach wurde eine externe Kamera mit Hilfe von der Software MATLAB® zur Objekterkennung benutzt und mit der Pick-and-Place Anwendung verknüpft.

Schlagworte:

Delta-Roboter, Objekterkennung, Pick-and-Place, Roboterzelle

Abstract:

This bachelor thesis deals with the design of a delta robot cell and simulation of a pick-and-place application. The delta robot to be assembled is provided by the FH Wiener Neustadt. The robot cell was assembled from aluminium profiles and a steel frame. In addition, the assembly was equipped with peripheral devices so that different applications can be tested. A simulation of the assembly was carried out in the robot software RobotStudio® for testing. A pick-and-place application was suitable for the simulation so that a cycle time analysis could be carried out. Afterwards an external camera was used for object recognition with the help of the MATLAB® software and linked to the pick-and-place application.

Keywords:

Delta-Robot, Object detection, Pick-and-Place, Robot cell

Danksagung

An erster Stelle möchte ich mich sehr bei meinem Eltern danken, die mir mein Studium ermöglicht und mich in all meinen Entscheidungen unterstützt haben. **Vielen Dank!**

Des Weiteren möchte ich mich auch bei meinen Freunden danken, die mir nützliche Tipps beim Schreiben zu dieser Arbeit beigetragen haben.

Ein herzliches Dankeschön bei meinen Kommilitonen und Kommilitonin Sulaiman Al-Muqaini, Andreas Fechete und Melissa Özdemir, da wir schwere Zeiten als auch schöne Zeiten gemeinsam im Bachelor-Studium verbracht haben.

An dieser Stelle möchte ich mich sehr bei meinem Betreuer Herrn Dipl.-Ing. Dr. Alexander Nemecek danken. Er hat mich richtungsweisend und mit viel Engagement während meiner Arbeit begleitet.

Weiterhin möchte ich mich beim Herrn Dipl.-Ing. Dr. Koller Christian und all jenen danken, die durch ihre fachliche und persönliche Unterstützung mich während meines Studiums begleitet haben.

Inhaltsverzeichnis

1 Aufgabenstellung

Das Ziel dieser Arbeit war für einen bereits vorhandenen **Flexpicker**-Roboter, von der Fa. ABB, eine Roboterzelle mit Demoanwendung und Objekterkennung aufzubauen , sowie eine Taktzeitanalyse dazu durchzuführen. Dabei wurde die Arbeit in bestimmte Bereiche eingeteilt.

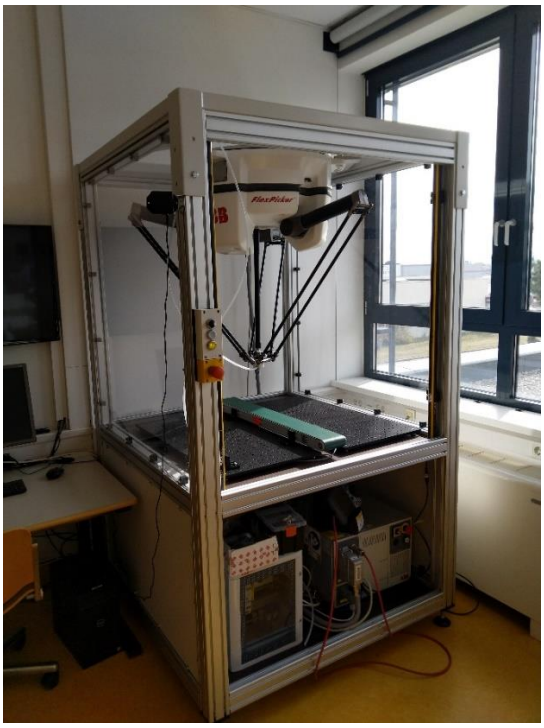


Abbildung 1: Gesamtaufbau des IRB 360's.

Aufbau der Roboterzelle

Das Gerüst besteht hauptsächlich aus Aliminium-Profilen. Für die Montage vom **Flexpicker**-Roboter wurde ein Stahlrahmen konstruiert[1], zusätzlich verschafft dies der Roboterzelle mehr Robustheit, siehe Abbildung 1. Um eine Pick-and-Place Anwendung durchführen zu können, wurde ein vollständiges Vakuumsystem für den Delta-Roboter integriert. Des Weiteren wurde ein Sicherheitskonzept an der Roboterzelle vorgenommen. Dieses besteht aus einem Lichtvorhang, um das Eingreifen in den Arbeitsbereich zu verhindern. Außerdem befindet sich eine Not-

[1] Konstruktion und Aufbau der modularen Roboterzelle für den Delta-Roboter wurde erforderlich, da die Einbringung einer fertig verschweißten Standard-Roboterzelle aus Stahlrohr durch die Labortüre nicht möglich war.

Aus Taste am Rahmen. Zusätzlich zum Not-Aus wurden weitere Tasten mit verschiedenen Funktionalitäten platziert. In der Roboterzelle befindet sich ein zentrales Förderband zum Transport der Werkobjekte. Damit eine übersichtliche elektrische Verkabelung z.B. von den Tasten gegeben ist, wurde ein Schaltschrank vorgesehen. Zusätzlich wurde die Roboterzelle mit einer Kamera erweitert, um bestimmte Objekte zu erkennen. Die Objekterkennung ermöglicht dem Manipulator, die Objekte intelligent und selbstständig zu orten und zu bewegen.

Simulation

In der Software RobotStudio®, von der Fa. ABB, wurde die reale Roboterzelle in ein virtuelles Modell umgesetzt. Dadurch wurde eine bessere Veranschaulichung bewirkt. Anschließend wurde mit dem virtuellen Modell eine Simulation durchgeführt. Mit dieser Simulation konnte der Arbeitsbereich von dem Delta-Roboter exakt bestimmt werden. Für die Simulation wurde eine Pick-and-Place Anwendung programmiert. Ein Video der Simulation ist hier abrufbar – Link [1].

Pick-and-Place Anwendung

Als Anwendung wurde ein Pick-and-Place Programm geschrieben, bei der ein Objekt einen bestimmten Bewegungsablauf automatisch durchläuft. Als Objekt wurden fünf Ronden auf einer Startposition platziert. Die Ronden werden nun vom Delta-Roboter auf einem Förderband abgelegt und an die Zielposition transportiert. Diese Anwendung wird im Folgenden auch für die Taktzeitanalyse verwendet.

Taktzeitanalyse

Bei der Taktzeitanalyse wird ein Objekt von der Position A an die Position B transportiert. Dazu wurde die obige Pick-and-Place Anwendung verwendet. Für die Analyse wurden die Zeiten von den einzelnen Bahnen, die der Roboter fährt, gemessen. Zusätzlich wurde die gesamte Simulationszeit optimiert.

Objekterkennung

Damit die zufällig platzierten Ronden von der Pick-and-Place Anwendung vom Delta-Roboter aufgenommen werden können, wurde eine einfache Objekterkennung eingebaut. Die Erkennung umfasst alle kreisförmigen Objekte und wurde in der Software MATLAB® durchgeführt.

2 Stand der Technik

Das Bestreben in der Industrie war schon immer die Effizienz der Fertigung durch Erhöhung des Durchsatzes zu verbessern. Demzufolge hat das zu einer verstärkten Entwicklung von industriellen Manipulatoren, sogenannten Industrierobotern geführt. Die Robotermanipulatoren wurden generisch entwickelt, sodass ein Manipulator verschiedene Anwendungen in verschiedenen Produktionslinien durchführen kann. Es gibt eine Vielzahl von Anwendungen, darunter Bestückung, Lackierung, Montage, Produktionsinspektion, Qualitätskontrolle und Prüfung. Bei bestimmten Aufgaben kann es vorkommen, dass diese Robotermanipulatoren für die Aufgabe nicht optimal geeignet sind. Vor allem bei Pick-and-Place Tätigkeiten in Produktionslinien kommt es auf die Geschwindigkeit der Manipulatoren an. Daher ist es unbedingt erforderlich einen geeigneten Roboter zur Maximierung der Produktionsleistung einzusetzen. Viele Pick-and-Place Aufgaben benötigen die Komplexität herkömmlicher 6-Achsen-Roboter nicht, deshalb ist es üblich hier einen Delta-Roboter (auch Parallel Roboter genannt) einzusetzen. Der Delta-Roboter hat in den letzten Jahrzehnten verschiedene Transformationen durchlaufen und ist eine der effizientesten Lösungen für schnelle industrielle Bestückungsanwendungen. Der Delta-Roboter wird meistens mit einer Kamera unterstützt, um die Objekte auf einem Förderband in der Produktionslinie zu erkennen. Somit können die Manipulatoren eigenständig die bestimmte Aufgabe mit hoher Genauigkeit erfüllen.

Robotik Hersteller statten die Roboterplattformen regelmäßig mit neuen Funktionen aus, sodass die Breite der potenziellen Anwendungen und die Grenzen der Autonomie erweitert werden. In diesem Kapitel werden im Folgenden einige Delta-Roboter vorgestellt und anhand der Eckdaten verglichen.

2.1 Delta-Roboter Fanuc Typ M-2iA/3SL

Fanuc ist seit 1956 im Bereich der Industrieroboter aktiv und bietet bei den Delta-Robotern drei verschiedene Serien an. Die Serien unterscheiden sich hauptsächlich durch die Größe des Manipulators. Die Serie M-2iA ist die Mittelklasse unter den Fanuc Delta-Robotern. Der Unterschied zwischen den Modellen liegt auch in deren Arbeitsbereich, Tragkraft und der Anzahl der Achsen. Das Modell M-2iA/3SL [2], siehe Abbildung 2, hat vier Achsen, eine Reichweite mit einem Durchmesser von

d=1130mm, eine Traglast von m=3kg und ein Gewicht von M=120kg. Dieser Manipulator kann Pick-and-Place Aufgaben rasch und präzise mit einer Genauigkeit w=0,1mm leicht meistern. Die M-2iA Serie greift und platziert Objekte mit einer Pickrate von p=182picks/min. Optional kann der M-2iA/3SL mit einer IP69K Schutzklasse gewählt werden, um in Reinräumen eingesetzt werden zu können.

Abbildung 2: Delta-Roboter Fanuc Typ M-2iA/3SL mit 3 Achsen [2].

2.2 Delta-Roboter Omron Typ Hornet 565

Ein weiterer Delta-Roboter ist der von Firma Omron hergestellte Typ Hornet 565 [3], siehe Abbildung 3. Im Gegensatz zu Fanuc M-2iA/3SL besteht die Einzigartigkeit des Modells Hornet 565 darin, dass die Steuerung ganz in dem Manipulator integriert ist. Die integrierte Steuerung bietet äquivalente Funktionalität wie bei einer externen Steuerung, spart jedoch bis zu 50% Platz im Vergleich zu herkömmlichen Delta-Robotern. In der integrierten Steuerung können bis zu zwei Encoder angeschlossen werden, um die Förderbandposition zu bestimmen. Der Hornet 565 wiegt M=52kg, wird mit drei oder vier Achsen ausgeliefert, hat als Durchmesser eine Reichweite von d=1130mm und eine Traglast bis m=3kg bei der Variante mit vier Achsen. Der Manipulator hat eine Wiederholgenauigkeit von w=0,1mm mit einer Pickrate p=162picks/min bei einer Last m=1kg.

Abbildung 3: Delta-Roboter Omron Hornet Typ 565 [3].

2.3 Delta-Roboter Igus Typ drylin 360

Die Firma Igus hat mit einem Baukastensystem den Delta-Roboter Typ drylin 360 entwickelt [4], siehe Abbildung 4. Die Vorteile von dem System sind das geringe Gewicht M=15kg und die leichte Montage vom Manipulator. Im Unterschied zu anderen Modellen hat der drylin eine lineare Bewegung der Gelenke statt rotatorischen. Der Manipulator kann Werkstücke bis m=5kg heben, weist einen Arbeitsraumdurchmesser von d=360mm und der Roboter hat eine Pickrate von p=30picks/min bei m=1kg. Der einzige Nachteil ist die Genauigkeit von nur w=0,5mm.

Abbildung 4: Delta-Roboter Igus Typ drylin 360 [4].

2.4 SCARA-Roboter Stäubli Typ FAST picker TP80

Der SCARA-Roboter (engl. Selective Compliance Assembly Robot Arm) Typ FAST picker TP80, siehe Abbildung 5, ist ein vier-Achs-Roboter und wurde von Fa. Stäubli entwickelt [5]. Dieser Manipulator wird in Pick-and-Place Aufgaben verwendet und wird mit einem Gewicht von M=68kg an der Wand montiert. Dadurch, dass die maximale Tragkraft bei m=1kg liegt, können nur leichte Objekte damit transportiert werden. Im Gegensatz zu einem Delta-Roboter ist der Arbeitsbereich kleiner, die Reichweite liegt bei einem Radius r=800mm und die Arbeitshöhe bei h=100mm. Die Pickrate kann bei leichten Objekten bis zu p=200picks/min erreichen. Im Gegensatz zu Delta-Robotern hat der Fast picker TP80 aufgrund der höheren mechanischen Steifigkeit dieser Bauart eine erhöhte Genauigkeit von w=0,05mm.

Abbildung 5: SCARA-Roboter Stäubli Typ FAST picker T80 [5].

2.5 Industrieroboter ABB Typ IRB1200-5

Eine andere Variante für Pick-and-Place Anwendungen ist der sechs-Achs-Roboter Typ IRB1200-5 der Fa. ABB [6] , siehe Abbildung 6. Dieser Manipulator hat aufgrund der seriellen Kinematik mit dem Radius r=901mm einen viel größeren Arbeitsraum als die bisher genannten Typen. Im Vergleich zu den Delta-Robotern kann der IRB1200-5 mit der hohen Pickrate nicht ganz mithalten und erreicht maximal p=142picks/min. Mit einem Gewicht von M=54kg kann der Manipulator aufrecht befestigt werden und kann maximal eine Masse m=5kg bewegen. Ein großer Vorteil ist hier die gute Wiederholgenauigkeit, die bei w=0,025mm liegt.

Abbildung 6: Industrieroboter ABB Typ IRB1200-5 [6].

2.6 Kollaborativer Roboter KUKA Typ LBR iiwa 7 R800

Ein relativ neuer Trend der Robotik ist die Mensch-Roboter Kollaboration (MRK). Darunter versteht man gefährdungsfreies Arbeiten direkt mit einem speziellen kollaborativen Roboter. Dazu besitzt der MRK-Roboter speziell runde Formen, um Klemmen oder Quetschen zu reduzieren. Durch zahlreiche Sensoren können Kollisionen erkannt und automatisch gestoppt werden. Weiters verfügen MRK-Roboter über eine deutlich reduzierte Traglast sowie geringere Geschwindigkeiten

[7]. Der Leichtbauroboter (LBR) „intelligent industrial work assistent“ (iiwa) von KUKA ist ein MRK-Roboter, siehe Abbildung 7. Der Typ. LBR iiwa 7 R800 [8] hat eine maximale Reichweite von r=1140mm und kann eine Last von m=7kg heben. Durch das geringe Gewicht M=23,9kg, kann der LBR einfach eingesetzt werden. Bei der Positioniergenauigkeit w=0,1mm hält der LBR mit den vorgestellten Delta-Robotern mit.

Abbildung 7: Kollaborativer Roboter KUKA Typ LBR iiwa 7 R800 [8].

2.7 Delta-Roboter ABB Typ IRB 360-3/1130

In dieser Arbeit steht der ABB **Flexpicker** mit der Ausführung IRB 360-3/1130 [9] zur Verfügung, siehe Abbildung 8. Der Manipulator hat eine Handhabungskapazität bis m=3kg und besitzt vier Achsen. Dieses Modell kann mit einer Pickrate p=150picks/min Werkstücke transportieren und mit einer Positioniergenauigkeit w=0,1mm platzieren. Die Arbeitsdurchführung kann in einem relativ großen Arbeitsbereich mit Durchmesser d=1130mm verrichtet werden. Bei der Montage ist mit einem Gesamtgewicht von M=120kg zu rechnen.

Abbildung 8: Delta-Roboter ABB Typ IRB 360-3/1130 [9].

Tabelle 1: Vergleich der Manipulatoren.

Hersteller	Fanuc [2]	Omron [3]	Igus [4]	Stäubli [5]	ABB [6]	KUKA [8]	ABB [9]
							
Bezeichnung	M-2iA/3SL	Hornet 565 (3+1)	drylin 360	FAST picker TP80	IRB1200-5	LBR iiwa	IRB360-3/1130
Traglast	3kg	3kg	5kg	1kg	5kg	7kg	3kg
Arbeitsbereich	Ø 1130x400mm	Ø 1130x425mm	Ø 360x120mm	r=800x100mm	r=910mm	r=1140mm	Ø 1130x300mm
Wiederhol-genauigkeit	0,1mm	0,1mm	0,5mm	0,05mm	0,025mm	0,1mm	0,1mm
Picks pro Minute	182picks/min	162picks/min	30picks/min	170picks/min	142picks/min		150picks/min
Gewicht	120kg	52kg	15kg	68kg	54kg	23,9kg	120kg

2.8 RoboTenis Vision System

Der Arbeitszyklus von einem Roboter wird meist fix vorprogrammiert, sodass der Roboter durchgehend dieselbe Bewegung immer wieder ausführt. Damit der Roboter bei anspruchsvollen Anwendungen adaptives Verhalten zeigen kann, wird die Orientierung und die Position des Werkstücks benötigt. In der Publikation [10] wurde ein Delta-Roboter verwendet, um einen Ping Pong Ball zu verfolgen. Dafür wurde der Delta-Roboter mit einem Kamerasystem ausgestattet, um die Daten vom Werkobjekt zu erfassen. Der Ping Pong Ball war schwarz und wurde an einem Gerüst mit einem weißen Hintergrund aufgehängt, siehe Abbildung 9. Die Ergebnisse haben gezeigt, dass das System bis zu einer Geschwindigkeit v=1m/s den Ball verfolgen konnte. Um Erkennungsfehler zu verringern, war der Abstand zwischen der Kamera und dem Ball konstant s=600mm. Der erzielte Tracking-Error der Anordnung beträgt e=6,3% bzw. e=32,5% bei einer Ball-Geschwindigkeit v=200mm/s bzw. v=800mm/s.

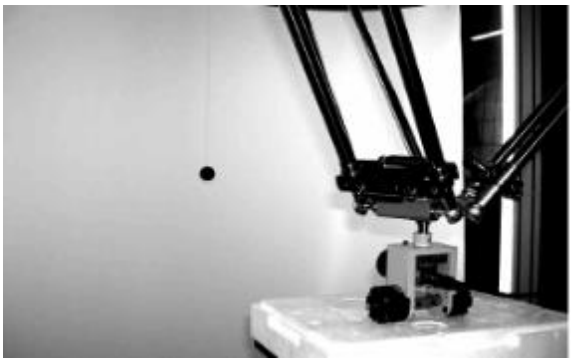


Abbildung 9: Kamerasystem mit Ping Pong Ball [10].

2.9 Modularer Mechanismus der Orientierung

Roboter haben eine gewisse Anzahl an Freiheitsgrade, um sich in einem Arbeitsraum bewegen können. Am Markt existieren viele Arten von Robotern, davon haben einige keine rotatorischen Freiheitsgrade. In der Publikation [11], wird ein montierbarer Mechanismus entworfen und simuliert, mit dem eine zusätzliche Orientierung vom Roboterhandgelenk ermöglicht wird, siehe Abbildung 10. Der Mechanismus beruht auf von Motoren gezogen Seilen, um eine rotatorische Bewegungen des Endeffektors auszuführen, siehe Abbildung 11. Das System befindet sich außerhalb des Roboters um das System möglichst wenig zu beeinflussen.

Abbildung 10: Modularer Mechanismus der Orientation [11].

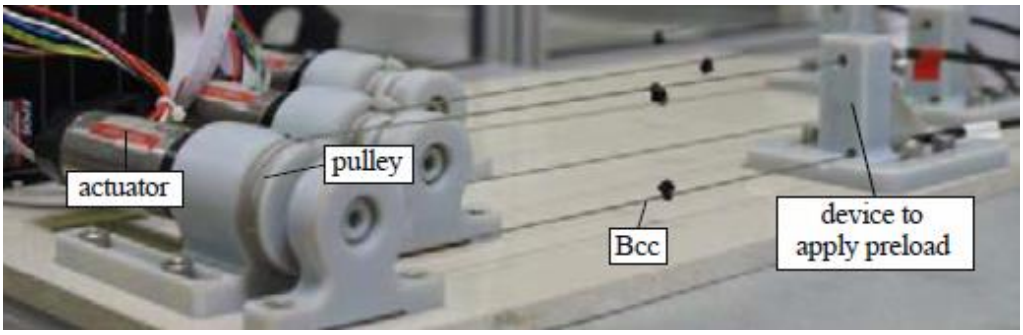


Abbildung 11: Externes System mit den Motoren [11].

2.10 Kostengünstiges Pick-and-Place

In der Publikation [12] wurde ein einfaches Pick-and-Place System entworfen, bei dem kostengünstige Komponenten benutzt werden. Das System hat eine Arbeitsfläche auf dem die Werkstücke liegen. Mit einem Sensor wird hier die Farbe des Werkstücks überprüft, siehe Abbildung 12. Die Werkstücke haben dazu eine bestimmte Größe und Form. Das System bewegt sich auf einer Linearachse hin und her. In einem Zyklus sind Erfassen, Halten, Ablegen des Objekts und das Bewegen des Pick-and-Place Arms in die Ausgangsstellung enthalten. Es wurden zwei Messungen mit einem schwarzen und einem weißen Box gemacht. In dem Zyklus mit der schwarzen Box wurde eine Distanz von s=190mm in der Zeit von t=7s abgeschlossen. Bei der weißen Box wiederum, war ein Zyklus mit einer Distanz von s=325mm in der Zeit von t=12s beendet.

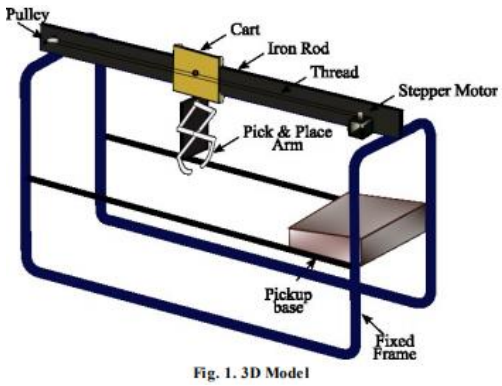


Abbildung 12: Kostengünstiges Pick-and-Place System [12].

2.11 Grafische Benutzeroberfläche zur Steuerung

Visuelle Darstellungen vereinfachen die Kommunikation von Menschen mit Robotern. In der Publikation [13] wurde eine grafische Oberfläche für einen Delta-Roboter programmiert. Hierfür wurde die Qt Plattform benutzt und in der Programmiersprache C entwickelt. Qt wird für das Erstellen von grafischen Benutzeroberflächen benutzt [14]. Mit der entwickelten grafischen Benutzeroberfläche (engl. englisch graphical user interface – GUI) war es möglich aus Bildern von Objekten mit Hilfe von einem Algorithmus die Kontur und die Position zu ermitteln, siehe Abbildung 13. Zusätzlich kann auch die Bewegungsgeschwindigkeit des Manipulators eingestellt und ein Saug-Greifer bedient werden.

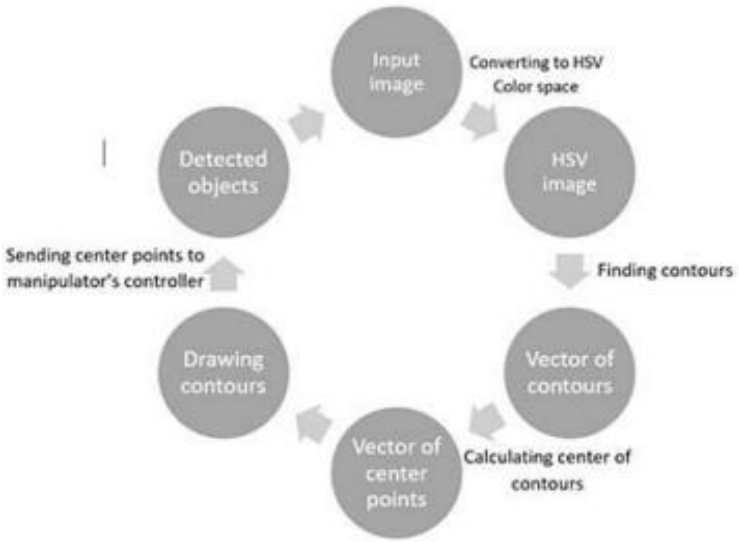


Abbildung 13: Algorithmus zur Objekterkennung [13].

2.12 Fractional-Order PID-Regler

Roboter müssen sich wiederholende Bewegungen mit hoher Genauigkeit und Dynamik ausführen und dabei werden die Roboterbewegungen geregelt. In der Publikation [15] werden dazu verschiedene Regelungsstrategien angewandt, wie z.B. PD Regler, PD Regler mit Gravitationskompensation, PD Regler mit Pre-Gravitationskompensation und P Regler mit Geschwindigkeitsfeedback, um externe

Beeinflussungen zu kompensieren. In dieser Arbeit wird als Vergleich zu einem PID-Regler ein Fractional-Order-PID Regler (FOPID) bei einem Delta-Roboter, getestet, um die Stabilität zu überprüfen, siehe Abbildung 14. FOPID-Regler werden normalerweise nicht bei Robotern eingesetzt. Der Delta-Roboter wurden drei Tests unterzogen, bei denen externe Störungen angelegt worden sind. Aus den erhaltenen Ergebnissen hat sich gezeigt, dass der FOPID Regler in Summe weniger Fehler aufweist und robuster ist als der PID Regler.

Abbildung 14: Getesteter Delta-Roboter [15].

3 Aufbau der Roboterzelle

Eine Roboterzelle ist ein komplettes System, dass den Roboter, die Steuerung, sowie verschiedene Peripheriegeräte z.B. ein Förderband und Sicherheitseinrichtungen umfasst. Zu Beginn der Konstruktion wurde für die Roboterzelle eine Skizze vom gesamten Aufbau entworfen, siehe Abbildung 15. Dazu war es wichtig die vorgegebenen Außenmaße Länge l=1250mm und Breite b=1250mm aufgrund begrenztem Platzbedarf einzuhalten. Deshalb werden in diesem Kapitel die Parameter besprochen, die einen Einfluss auf die Dimensionierung der Roboterzelle hatten.

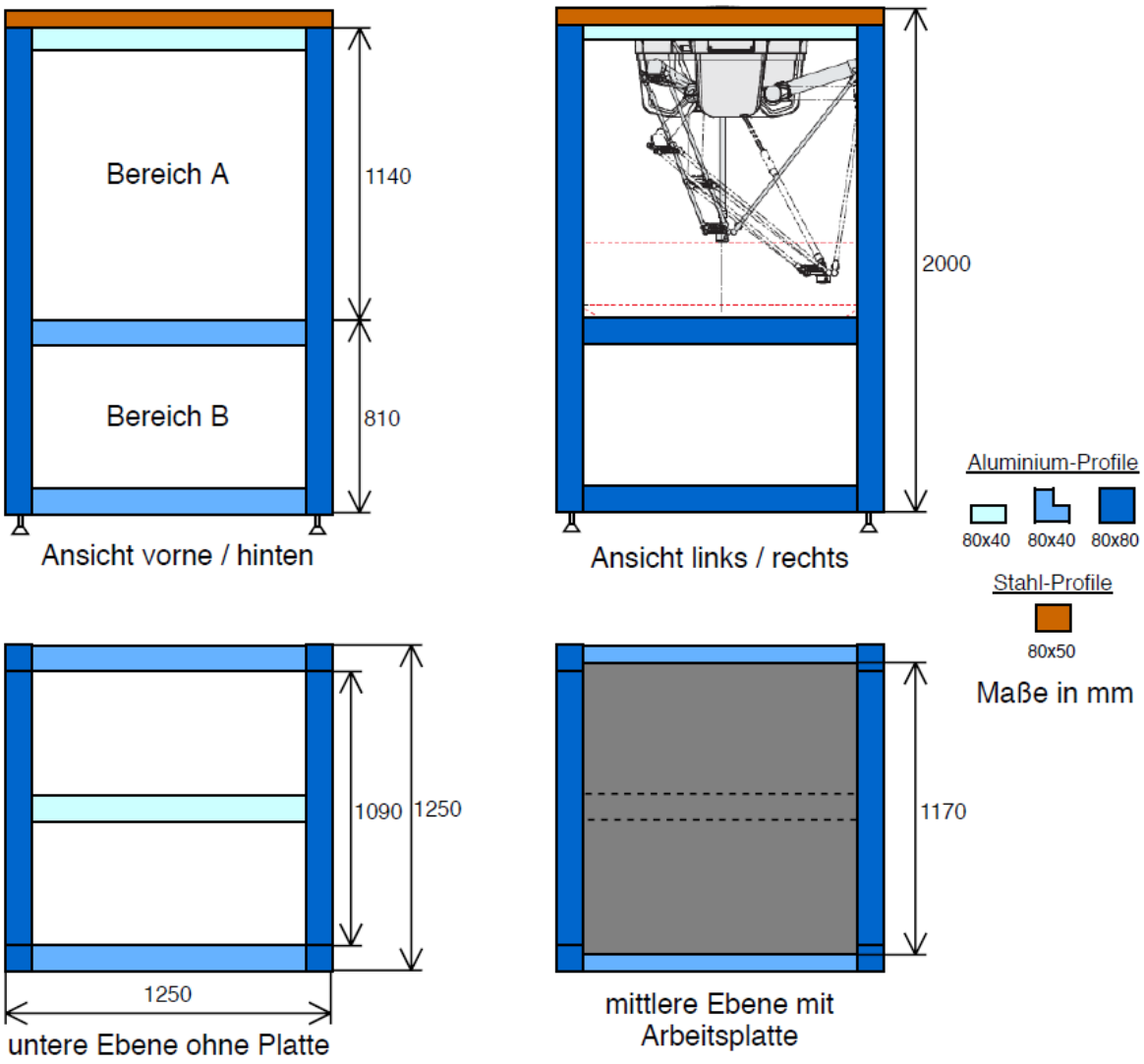


Abbildung 15: Skizze der Roboterzelle.

3.1 Deltaroboter

In diesem Projektabschnitt wurde der Delta-Roboter IRB 360-3/1130 , siehe Kapitel 2.7, in der Roboterzelle montiert. Die Montage vom Delta-Roboter am Stahlrahmen erfolgte über drei M12 Schrauben mit einem Winkelabstand von 120° zueinander, siehe Abbildung 16.

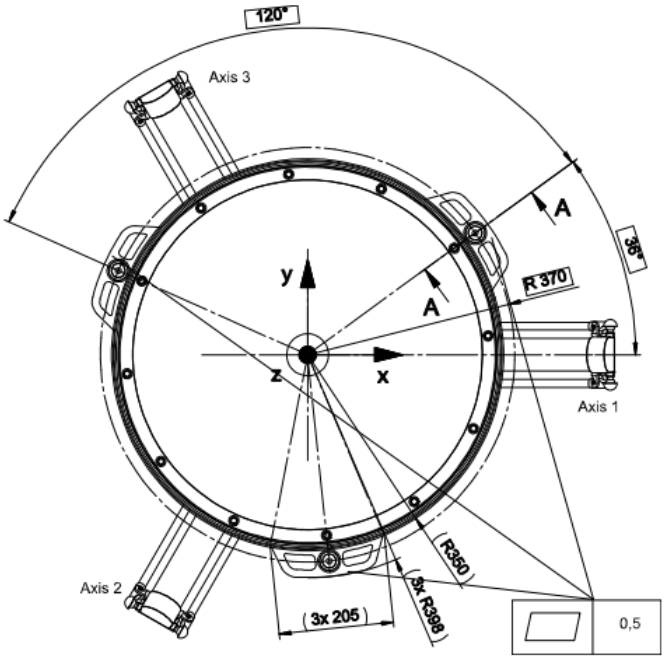


Abbildung 16: Draufsicht des Delta-Roboters [16].

In der Abbildung 17, wird der Arbeitsraum für die jeweiligen Modelle vom IRB 360 gezeigt. Bei dem verwendeten Modell beträgt die maximale Höhe h=300mm und der maximale Durchmesser d=1130mm.

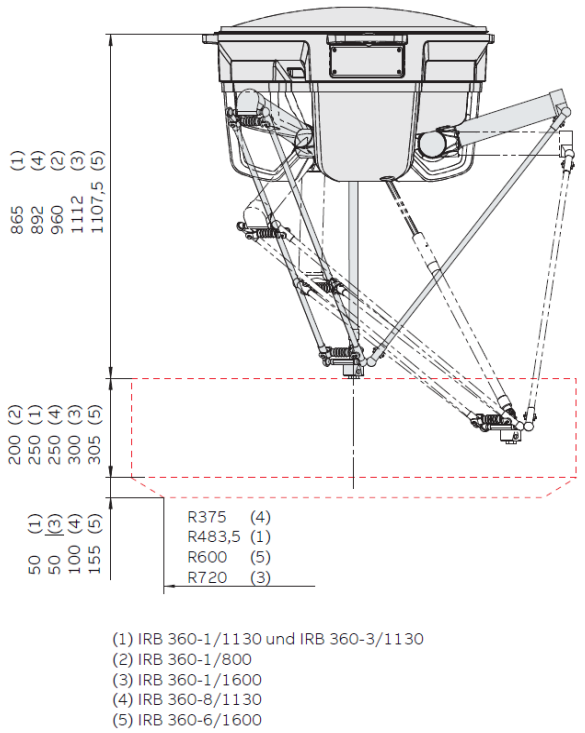


Abbildung 17: Arbeitsbereiche der Modelle IRB 360 [9].

Der Delta-Roboter wird nach der Montage an die Robotersteuerung IRC5 Compact [17] angeschlossen, siehe Abbildung 20. Mit der Steuerung lassen sich die Achsen des Roboters und die Peripheriegeräte steuern. Die Steuerung hat als Abmessungen Länge, Breite und Höhe l=320mm, b=449, h=442mm. Die Steuerung wird im unteren Bereich B platziert und benötigt als elektrische Versorgung die Netzspannung U=230V.

An der Steuerung wird zusätzlich ein Bedienfeld angeschlossen, das sogenannte **FlexPendant** [18]. Das **FlexPendant** ermöglicht es den Roboter direkt zu bedienen, anzusteuern und zu programmieren. Es besteht aus einem Touchscreen, Steuerknüppel und Tasten, siehe Abbildung 18.

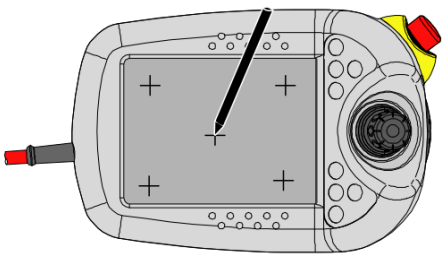


Abbildung 18: Vorderansicht FlexPendant [18].

3.2 Mechanischer Aufbau

Der mechanische Aufbau besteht zunächst aus zwei verschiedenen Profilen. Das Aluminium-Profil stellt die Zelle dar. Das zweite Profil bildet den Stahl-Rahmen und wird für die Befestigung vom Manipulator verwendet. Der Stahl-Rahmen wird wie in Abbildung 15 ersichtlich auf die Aluminium-Profile montiert. Dimensionen und Stückanzahl der Aliminium Profile sind in der Tabelle 2 eingetragen.

Tabelle 2: Verwendete Aliminium-Profile.

Typ	Stück	Länge	Breite	Tiefe
	4	80mm	80mm	1950mm
	4	80mm	80mm	1090mm
	5	80mm	40mm	1090mm
	5	80mm	40mm	1090mm

Die Kombination aus Alu-Profil und Stahlrahmen hat den Vorteil, dass die Aluminium-Profile einfach zusammengebaut werden können während der Stahl-Rahmen eine hohe Steifigkeit. Durch die hohe Steifigkeit kann der Rahmen den Manipulator mit einem Gesamtgewicht M=120kg tragen. Des Weiteren wurde der Stahl-Rahmen mit den Montagebohrungen des Manipulators und der Zelle extern von einer Schlosserei gefertigt und Pulverbeschichtet, siehe Abbildung 16 und Abbildung 19.

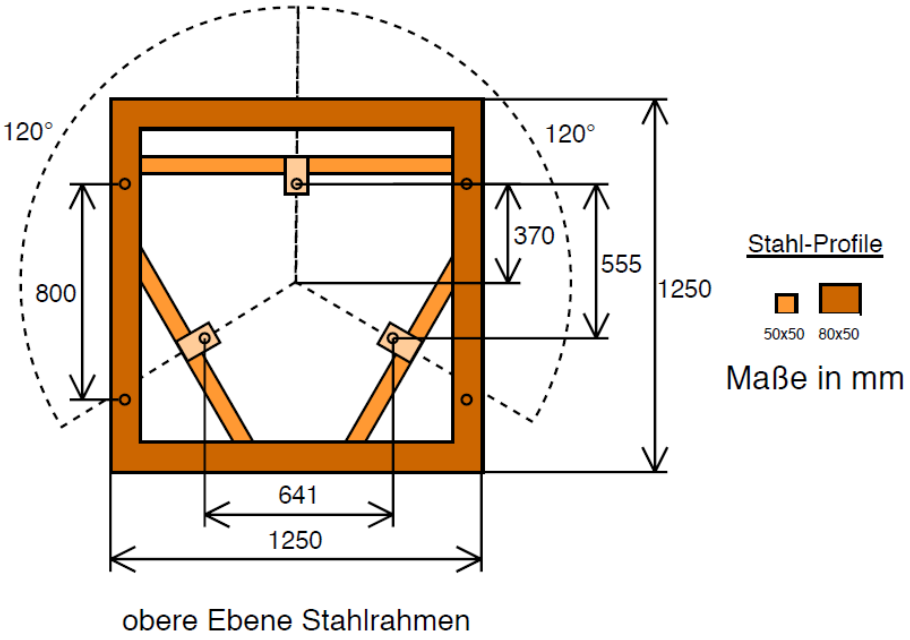


Abbildung 19: Stahl-Rahmen mit Bemaßungen.

Die Roboterzelle teilt sich in Bereich A und Bereich B auf, siehe Abbildung 15. Bereich A ist der Arbeitsbereich in dem sich der Roboter und die Werkstücke befinden. Bereich B wiederum wird als Stauraum für die Steuerung, das Vakuumsystem und der Schaltschrank verwendet. Die Unterteilung erfolgt mit einer Arbeitsplatte auf dem die Werkobjekte platziert werden können. In den folgenden Kapitel 3.3 bis 3.9 werden diese Inhalte näher erläutert.

3.3 Sicherheitseinrichtung

Ohne Sicherheitseinrichtung besteht bei einer Roboterzelle die Gefahr, dass eine Person verletzt werden kann. Um dies zu verhindern, wurde ein Sicherheitskonzept erstellt. In der Skizze in Abbildung 20 ist ein Überblick der Sicherheitskomponenten gegeben.

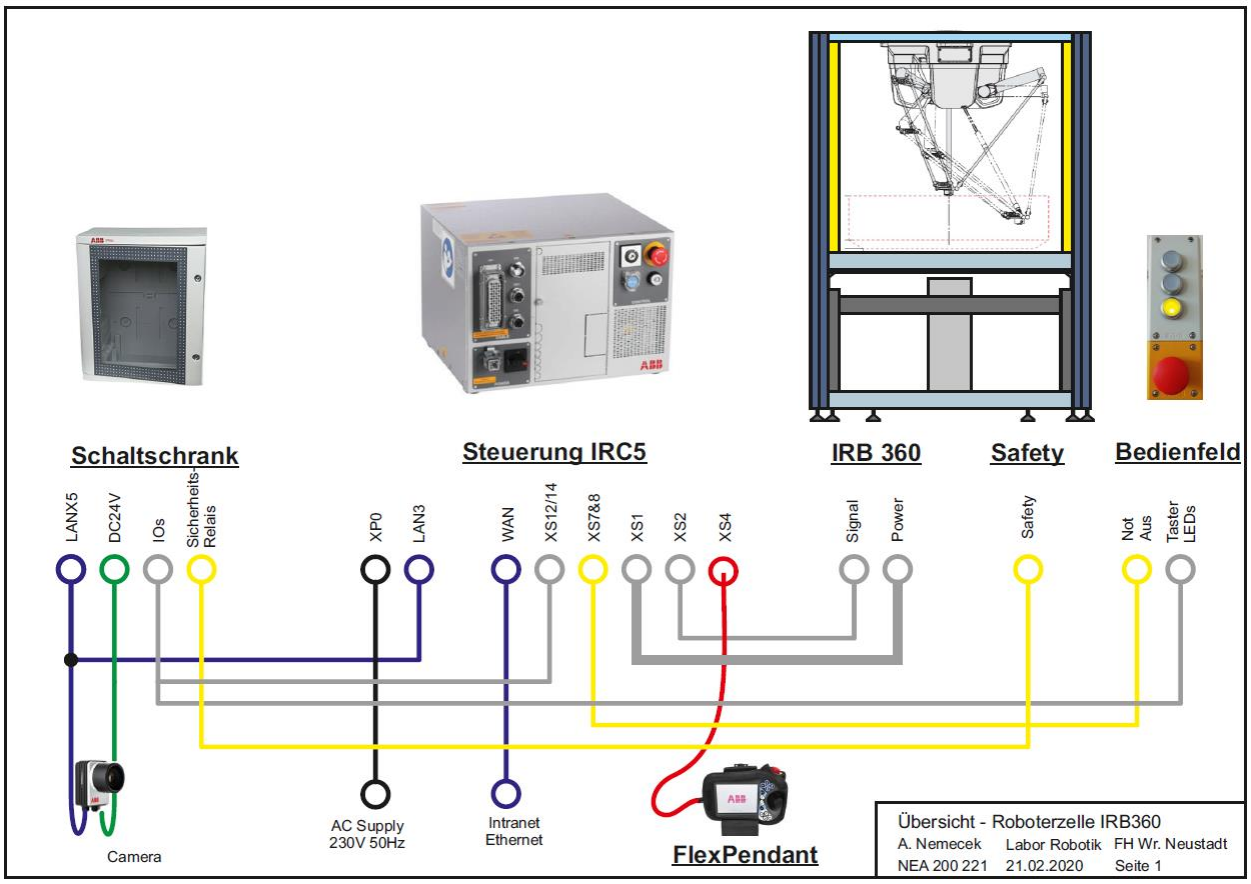


Abbildung 20: Sicherheitskonzept.

Die Robotersteuerung IRC5 Compact hat einen internen Sicherheitskreis mit eigenen Inputs und Outputs. Diese können verwendet werden, um externe Sicherheitseinrichtungen zu integrieren. Somit kann ein Not-Aus-Schalter von EATON Typ. M22-PV/KC11/IY [21] direkt an die Steuerung angeschlossen werden. Der vorgesehene Lichtvorhang von SICK wiederum kann nicht direkt mit der Steuerung verbunden werden. Dazu wird ein Sicherheitsrelais von SICK Typ UE48-20S2D2 [20] eingesetzt. Eine Unterbrechung des Lichtweges des Lichtvorhanges führt zu einer Deaktivierung des Sicherheitsrelais und somit wird der Sicherheitskreis unterbrochen. Damit das Sicherheitsrelais reaktiviert wird, muss die gelbe Taste auf dem Bedienfeld betätigt werden.

In der Abbildung 21 befinden sich alle externen Sicherheitskomponenten.

Abbildung 21: Lichtvorhang [19] , Sicherheitsrelais [20] und Not-Aus-Schalter [21].

Der Lichtvorhang besteht aus zwei Teilen, dem Sender Typ C4C-SA18030A10000 und dem Empfänger Typ. C4C-EA18030A10000 der Firma Sick [19]. Beide haben eine Höhe von h=1080mm und wurden jeweils seitlich innen vorne an den Säulen montiert. Beim Befestigen muss der Sender und der Empfänger parallel zueinander ausgerichtet werden, um den ganzen Frontbereich A abzudecken und gute Signalqualität sicherzustellen. Die Abbildung 22 zeigt die Montageanordnung von Lichtvorhang, Not-Aus-Schalter, Bedienfeld und Sicherheitsrelais im Schaltschrank.

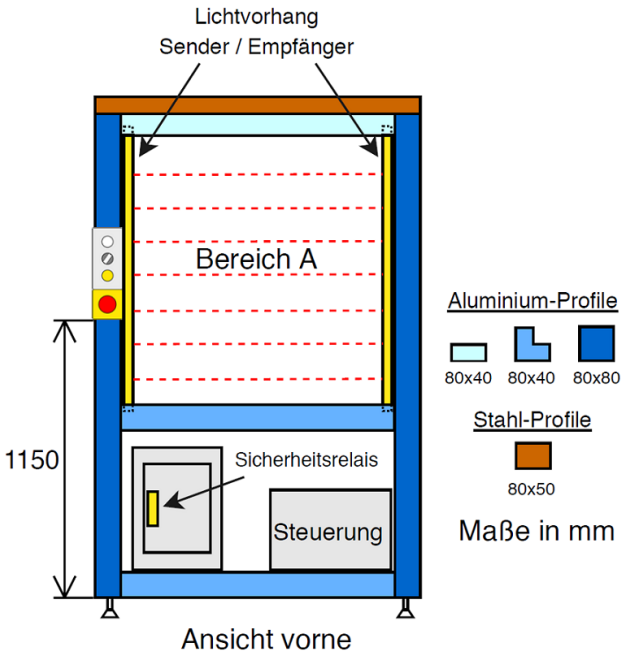


Abbildung 22: Montageanordnung der Sicherheitskomponenten.

3.4 Bedienfeld

Auf der linken Säule wurde ein Bedienfeld angebracht. Auf diesem befinden sich zwei Tasten und ein Schalter plus jeweils eine LEDs zur Signalisierung, siehe

Abbildung 23. Die obere Taste kann zusammen mit der LED als Input- und Output-Signal in die Programmierung einbezogen werden. In der Mitte platziert ist ein Schalter mit dem sich das Kamerasystem Ein- und Ausschalten lässt. Ist die Kamera eingeschaltet, leuchtet die LED im Schalter. Unten befindet sich eine gelbe Taste für die Freigabe des Lichtvorhanges. Falls der Lichtvorhang betätigt wird, leuchtet die LED auf und signalisiert somit die Unterbrechung.

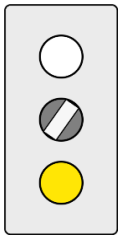


Abbildung 23: Bedienfeld.

3.5 Förderband

Viele Aufgaben für die Delta-Roboter beinhalten ein Förderband, um den Transport von Werkstücken gewährleisten zu können. Beim Aufbau der Roboterzelle wird aus diesem Grund ein Förderband integriert. Dafür wird das Förderband von der Fa. Vetter Typ FR-40-120 [22] mit den Abmessungen Länge, Breite, Höhe l=800mm b= 120mm, h=40mm verwendet, siehe Abbildung 24.

Abbildung 24: Vetter Förderband Typ. FR-40-120 [22].

Seitlich vom Förderband befinden sich Alu-Profile für T-Nutensteine mit Nutenbreite b=5mm. Mit den Nutensteinen werden vier Winkel angeschraubt. Die Winkel wiederum sind auf den Arbeitsplatten montiert. In Abbildung 25 ist die Skizze der Montage zu sehen und im Anhang befinden sich die technischen Zeichnungen von dem Förderband und der Winkel.

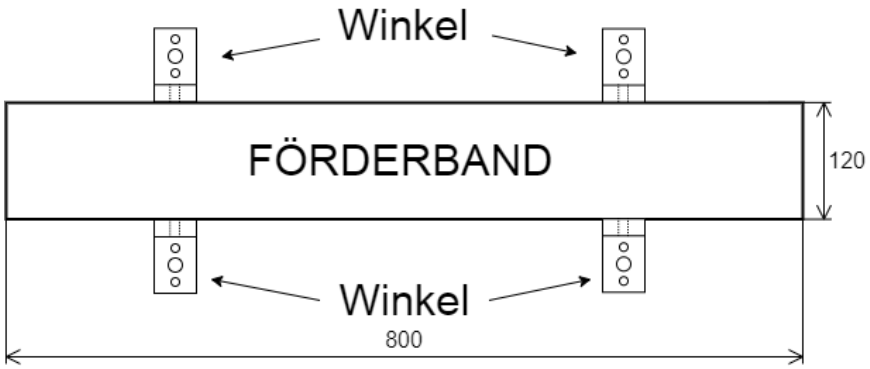


Abbildung 25: Förderband mit den Winkeln.

In dem Förderband befindet sich ein bürstenloser DC-Servomotor mit integriertem Speed Controller. Der Speed Controller lässt sich mit einem variablen Analogsignal einstellen, sodass sich der Gurt mit einer Geschwindigkeit v=0…10m/min bei einer Gewichtskraft F=50N bewegt. Das analoge Signal kann als Spannung U=0…11V mit einem Potentiometer von Vetter Typ. VSW-11 [23] eingestellt werden, siehe Abbildung 26. Das Potentiometer und auch das Förderband werden über ein Netzteil U=24V versorgt.

Der dazugehörige Anschlussplan befindet sich im Anhang.

Abbildung 26: Vetter Potentiometer [23].

3.6 Vakuumsystem

Jeder Industrieroboter benötigt ein Werkzeug zum Greifen von Werkstücken. Bei einem Delta-Roboter eignet sich dazu ein Vakuumsystem hervorragend für Pick-and-Place Aufgaben. Dabei wird ein Unterdruck erzeugt damit das Werkstück an dem Vakuumsauger haften bleibt.

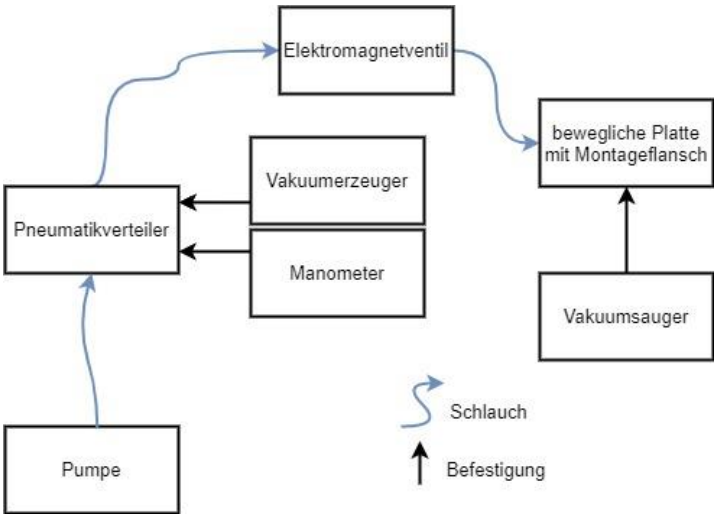


Abbildung 27: Anordnung Vakuumsystem.

Zunächst wurde für den Aufbau die Vakuumpumpe Typ. N 816.3 KT.18 der Fa. KNF [25] über einen Schlauch mit dem Verteiler verbunden, siehe Abbildung 27. Mit dem Pneumatikverteiler verbunden sind ein Manometer, ein Vakuumerzeuger und am dritten Ausgang über einen Schlauch das Elektromagnetventil. Des Weiteren ist das Ventil über einem Schlauch mit der beweglichen Platte des Delta-Roboters verbunden, siehe Abbildung 28. Die bewegliche Platte hat einen drehbaren Montageflansch auf dem der Vakuumsauger montiert ist. Alle Verbindungen wurden mit einem Gewindedichtungsband abgedichtet.

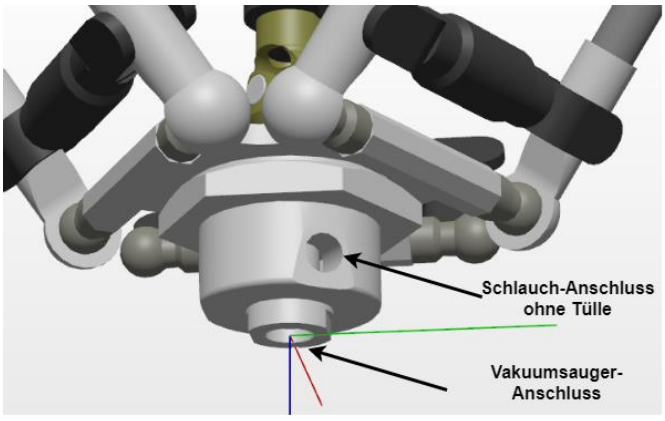


Abbildung 28: Bewegliche Platte mit Anschlussstelle.

Die Steuerung des Vakuums geschieht über das Elektromagnetventil. Das verwendete Ventil von SMC Typ VT307 [24] besitzt drei Durchflusswege (A, R und P) und zwei Schaltstellungen, siehe Abbildung 29. Wenn keine Spannung an der Spule anliegt, bewegt die Feder die Schubstange nach oben. Somit gleicht sich der

Druck an die Atmosphäre, denn die Wege A und R werden verbunden. Um die Schaltstellung zu ändern, wird an die Spule eine Spannung U=24V angelegt und die Schubstange drückt die Federung nach unten. Dadurch werden die Wege A und P verbunden und im System herrscht Unterdruck, damit wird das Werkstück an den Vakuumsauger gezogen.

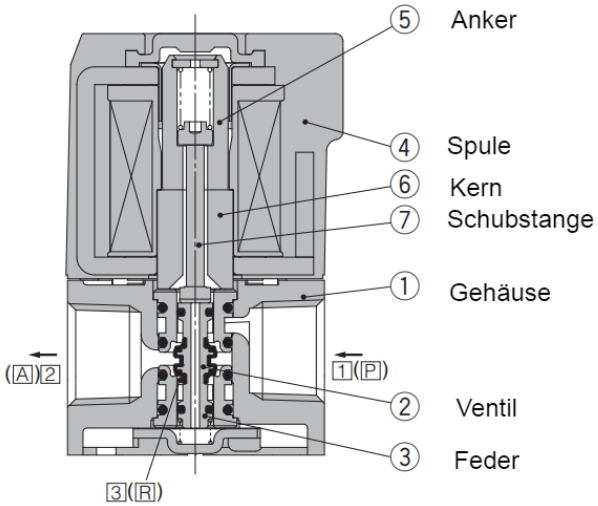


Abbildung 29: Funktion Elektromagnetventil [24].

Zwischen dem Ventil und dem 24V-Netzteil befindet sich ein Relais, welches über die Steuerung angesteuert werden kann.

In der Abbildung 30 sind alle Komponente des Vakuumsystems zu sehen.

Abbildung 30: Vakuumpumpe [25], Vakuumsauger [26], Magnetventil [24] und Pneumatikverteiler [27].

3.7 Kamera Cognex DSQC1021

Kamerasysteme ergänzen Roboter für komplexere Herausforderungen und erhöhen die Flexibilität des Roboters. Typische Anwendungen sind

Teilepositionierung, visuelle Teilekontrolle, Sortierung und mehr. Die verfügbare Cognex Kamera Typ DSQC1021 [28] hat einen monochromen Sensor mit einer Auflösung B=1280x1024, siehe Abbildung 31.

Abbildung 31: Kamera Cognex Typ DSQC1021 [29].

ABB stellt für die Bildbearbeitung mit der Kamera die Anwendung **Intergrated Vision** [28] zur Verfügung, in dem sich bereits bis zu 50 intelligente vordefinierte Bildbearbeitungswerkzeuge befinden. Die Aufnahmen können über das **FlexPendant** auch verfolgt werden. Der Anschluss wird über eine LAN-Verbindung gewährleistet, siehe Abbildung 32.

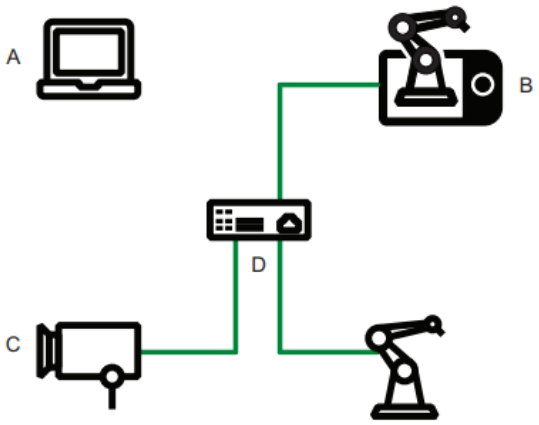


Abbildung 32: LAN Verbindung mit (A) PC, (B) FlexPendant, (C) Kamera und (D) Steuerung [28].

3.8 Conveyor-Tracking-Modul

Delta-Roboter werden meist mit einem Förderband und einer Kamera in der Produktion eingesetzt. Damit der Delta-Roboter die Bewegungsgeschwindigkeit von den Werkstücken erfassen und ausgleichen kann, wird ein Conveyor-Tracking-Module Typ DSQC2000 [30] von der Fa. ABB verwendet, siehe Abbildung 33. Das Modul bietet Anschlüsse für bis zu vier Encoder vom Förderband und bis zu acht

Kameras. Wenn das Förderband mit einer Geschwindigkeit v=150mm/s bewegt, befindet sich das Werkzeug des Roboters mit einer Genauigkeit w=2mm auf der Bahn.

Abbildung 33: Conveyor-Tracking-Module Typ DSQC2000 [31].

3.9 Schaltschrank

In den vorherigen Kapiteln wurden verschiedene Komponente beschrieben, die in der Roboterzelle integriert wurden. Deshalb wurde für eine übersichtliche elektrische Verkabelung ein Schaltschrank Typ Gemini von der Fa. ABB [32] zusätzlich zur Steuerung platziert, siehe Abbildung 34.

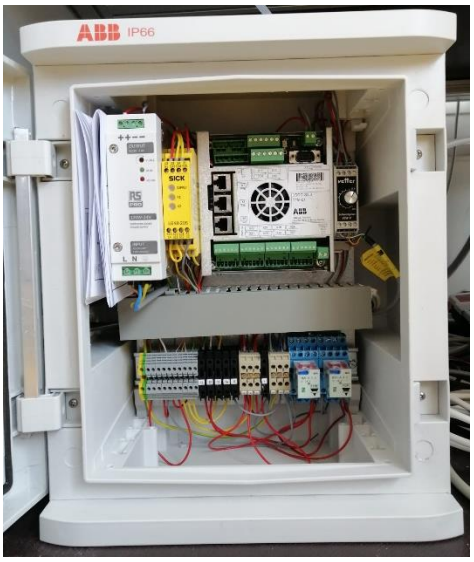


Abbildung 34: Schaltschrank geöffnet.

Damit die Verkabelungen übersichtlich sind, beinhaltet der Schaltschrank Klemmen auf einer Klemmleiste. In der Abbildung 35 werden die Verbindungen zwischen dem Schaltschrank und den Komponenten als Blockdiagramm gezeigt. Für eine detaillierte Schaltung siehe Anhang.

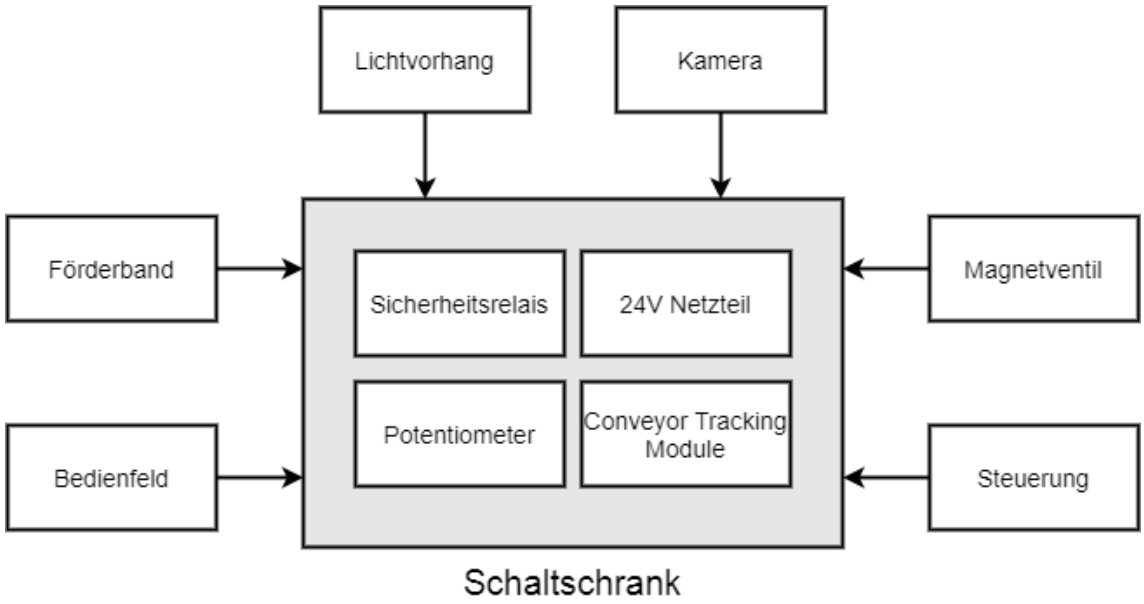


Abbildung 35: Blockdiagramm der Komponenten im Schaltschrank.

4 Objekterkennung

Roboter können vorprogrammierte Bahnen abfahren und Aufgaben in denen die Werkstücke stationär sind bearbeiten. Falls sich die Werkstücke nicht an vordefinierten Orten befinden, muss Objekterkennung durchgeführt werden. Soll der Roboter ein Werkstück nur greifen, ist eine einfache Erkennung ausreihend. Der Roboter kann anschließend das Werkstück an die Zielposition befördern. In der ausgeführten Pick-and-Place Anwendung werden Edelstahlronden verwendet, daher werden im folgenden kreisförmige Objekte behandelt, siehe Abbildung 36.

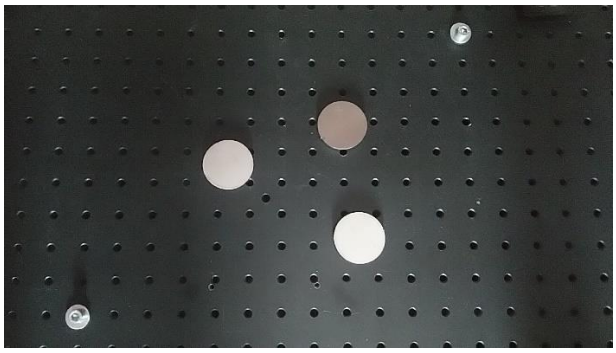


Abbildung 36: Edelstahlronden für die Bildbearbeitung.

4.1 Webcam

Wie in Kapitel 3.7 schon erwähnt, befindet sich ein fertiges Bildbearbeitungswerkzeug in RobotStudio® und kann mit der Cognex Kamera verwendet werden. Als Alternative wurde die in Abbildung 37 dargestellte universelle Webcam Logitech Typ C615 [36] verwendet, weil die vorhandene Cognex Kamera monochrom ist. Die Logitech Webcam kann Farbbilder mit einer Auflösung B=1920x1080 aufnehmen.

Abbildung 37: Webcam Logitech Typ C615 [36].

4.2 Methodik

Für die Bildbearbeitung wird die **Image Processing Toolbox** von MATLAB® verwendet. In der Toolbox befindet sich die Anwendung **Color-Thresholder**, die es unterstützt die Bilder in verschiedenen Farbräumen zu filtern. Unterstützt werden die RGB, HSV, YCbCr und L*a*b Farbbereiche, wie in der Abbildung 38 ersichtlich.

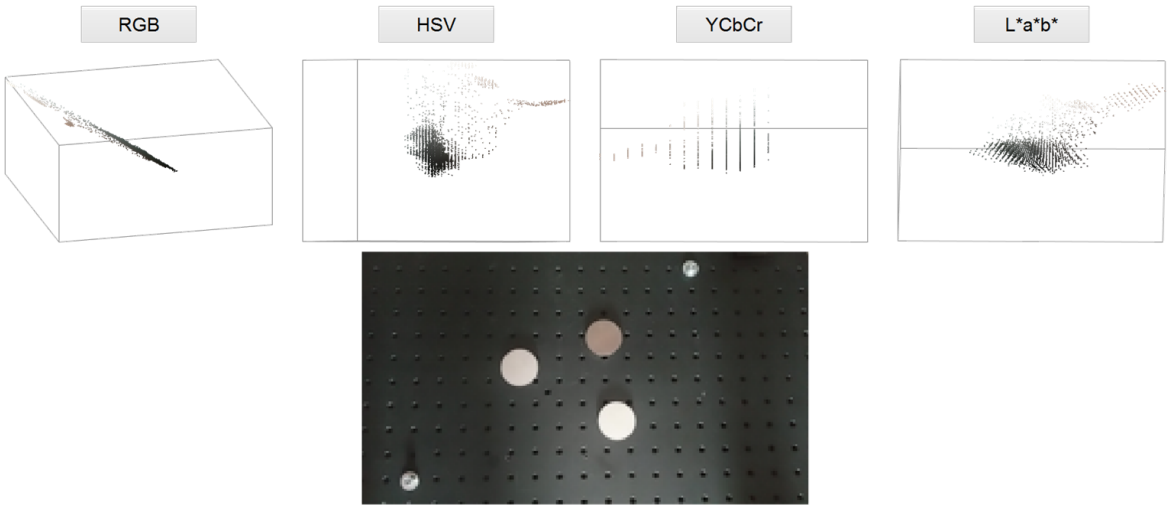


Abbildung 38: Color-Thresholder mit den Farbbereichen.

- **RGB**: In diesem Farbraum wurden die Farben rot, grün und blau definiert [37]. Es können jedoch nicht alle mit dem Auge wahrnehmbaren Farben abgedeckt werden.
- **HSV**: Die Abkürzung steht für Farbton (engl. hue), Sättigung (engl. saturation) und Helligkeit (engl. value). Diese Darstellung wurde für die Bildbearbeitung entwickelt und wird in vielen Graphikprogrammen verwendet [37].
- **YCbCr**: Die Abkürzung steht für die Grundhelligkeit Y sowie Blau-Gelb Chrominanze (Cb) und Rot-Grün Chrominanze (Cr). Dieses Farbmodell wird beispielsweise bei digitalen Videos verwendet.
- **L*a*b***: Dieser Farbraum hat den Vorteil, dass alle wahrnehmbaren Farben dargestellt werden können und ist Gerät unabhängig [37]. Die Abkürzungen stehen für Luminanz L, Grün- bis Rot-Anteil (a) und Blau- bis Gelb-Anteil (b).

Nach der erfolgten Farbfilterung im geeigneten Farbmodell wird das Bild mit der Funktion `im2bw` in ein Schwarzweiß-Bild umgewandelt. Falls sich schwarze Flecken auf den relevanten Kreisflächen befinden, werden diese mit der Funktion `imfill` beseitigt. Der MATLAB® Code dazu ist in dem Programmcode 1 dargestellt.

```
%  Ronde filtern:
        [y,rmat] = RondeFilter(img);

        BWr = im2bw(rmat,.1);
        %imshow(BWr);

        clear_BWr = imfill(BWr,'holes');
        %imshow(clear_BWr);
```

Programmcode 1: MATLAB Codeabschnitt für die Filterung.

Für kreisförmige Objekte befindet sich die Funktion `imfindcircles` [39] in der Toolbox. Die Funktion findet den Mittelpunkt und den Radius von Kreisen auf dem Bild. Dazu wird ein Radiusbereich angegeben. Damit die erkannten Kreise sichtbar werden, wird die Funktion `viscircles` verwendet, siehe Programmcode 2.

```
% Ronde finden
[center1,radius1] = imfindcircles(circles_BWr,[45 55],'Method',...
                    'TwoStage','Sensitivity',sensy_1);
imshow(img);
viscircles(center1cpy, radius1cpy,'EdgeColor','b');
```

Programmcode 2: MATLAB Codeabschnitt für Kreiserkennung.

Wie in der Abbildung 36 zu sehen ist, befinden sich im Bild diagonal zueinander zwei Schrauben. Diese Schrauben definieren den Bereich für die Werkstücke, in dem sie platziert werden können. Die Mittelpunkte von den Schrauben dienen als Bezugspunkt für das Koordinatensystem der Webcam, siehe Programmcode 3.

```
[centerSchraube,radiusSchraube] = imfindcircles(circles_BWscrew,...
    [15 28],'Method','TwoStage','Sensitivity',0.75)
screws = sort(centerSchraube);

%% ------------------ Koordinaten Webcam --------------------
%Webcam Koordinaten
camy =[screws(1, 2), screws(2, 2)];  %[48, 499]
camx =[screws(1, 1), screws(2, 1)];  %[39, 785]

camdx =camx(2)- camx(1);
camdy =camy(2)- camy(1);
```

Programmcode 3: Webcam Koordinaten.

Danach wird das Roboterwerkzeug an die Positionen der Schrauben bewegt und die Positionen in MATLAB® eingetragen. Somit lässt sich eine Transformation angeben, mit der die x und y-Werte vom Bild auf die Roboterkoordinaten umwandelt werden können, siehe Programmcode 4.

```
% Koordinaten-Transformation Ronde
faktorx = deltax/camdx;
faktory = deltay/camdy;

X = center1Int(:,2);
Y = center1Int(:,1);

Xtrans = rox(1)+(X-camx(1))*faktorx;
Ytrans = roy(1)+(Y-camy(1))*faktory;
```

Programmcode 4: Webcam Koordinaten in Roboterkoordinaten.

4.3 Ergebnisse

4.3.1 Farbfilterung

Die in Kapitel 4.2 erwähnten Farbräume wurde zum Filtern der Ronden und färbige Scheiben eingesetzt. In der Abbildung 39 wird gezeigt, dass die HSV-Filterung einen größeren Farbbereich abdeckt, damit die nicht erwünschten Details entfernt werden und daher bevorzugt wird. Die restlichen Farbräume unterscheiden sich beim erzielten Ergebnis kaum.

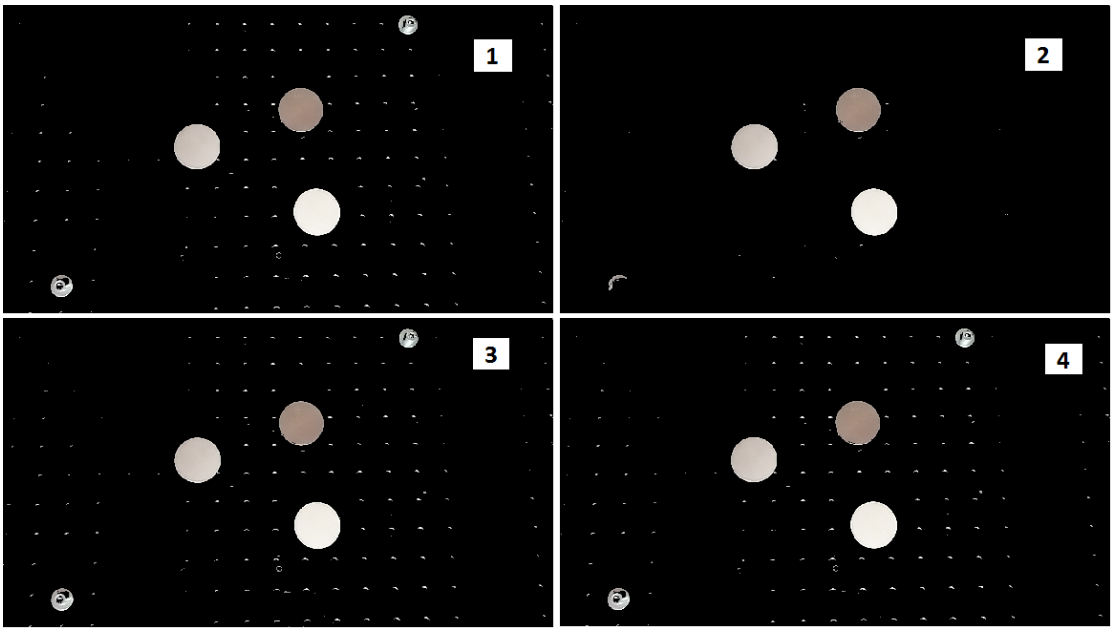


Abbildung 39: Farbfilterung der Ronden mit (1) RGB, (2) HSV, (3) YCbCr und (4) L*a*b*.

Bei den färbigen Test-Scheiben in Abbildung 40 wurden die Farbräume für jede Farbe geprüft. Es zeigt sich, dass die Farbe Blau besser mit dem HSV-Farbraum gefiltert wird, siehe Abbildung 41. Im Gegensatz dazu lassen sich die Farben Rot und Grün mit allen Farbräumen gut darstellen, siehe Abbildung 42 und Abbildung 43.

Abbildung 40: Färbige Scheiben für die Filterung.

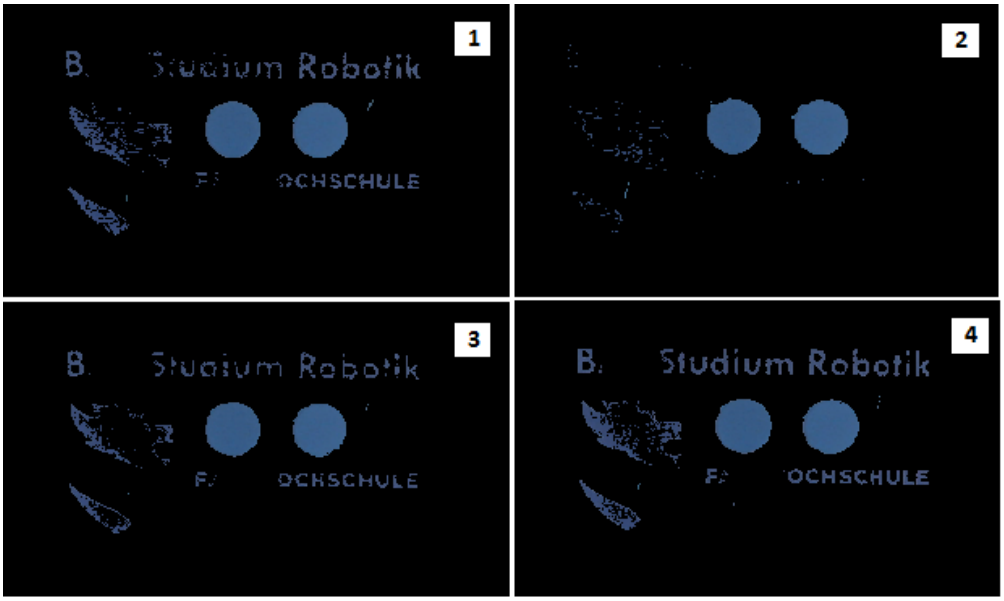


Abbildung 41: Farbfilterung blaue Scheiben mit (1) RGB, (2) HSV, (3) YCbCr und (4) L*a*b*.

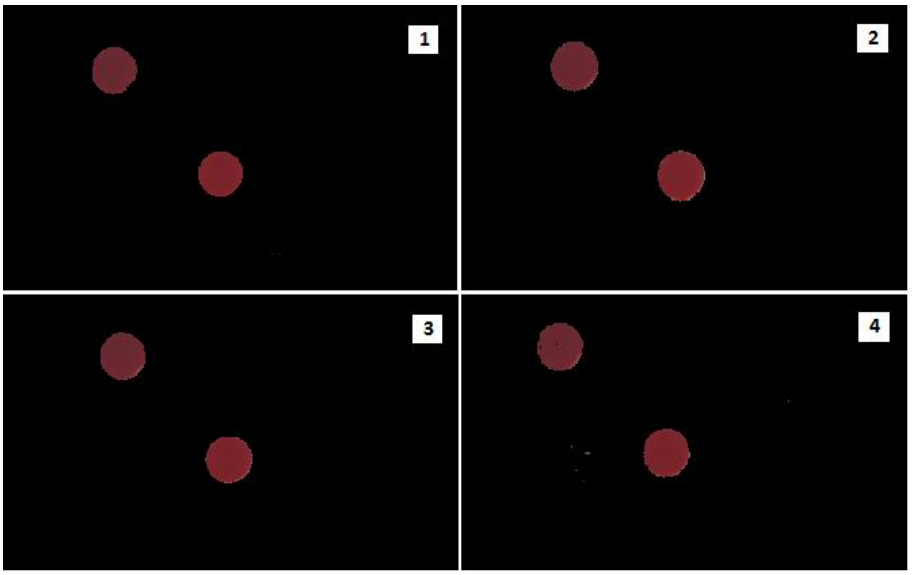


Abbildung 42: Farbfilterung rote Scheiben mit (1) RGB, (2) HSV, (3) YCbCr und (4) L*a*b*.

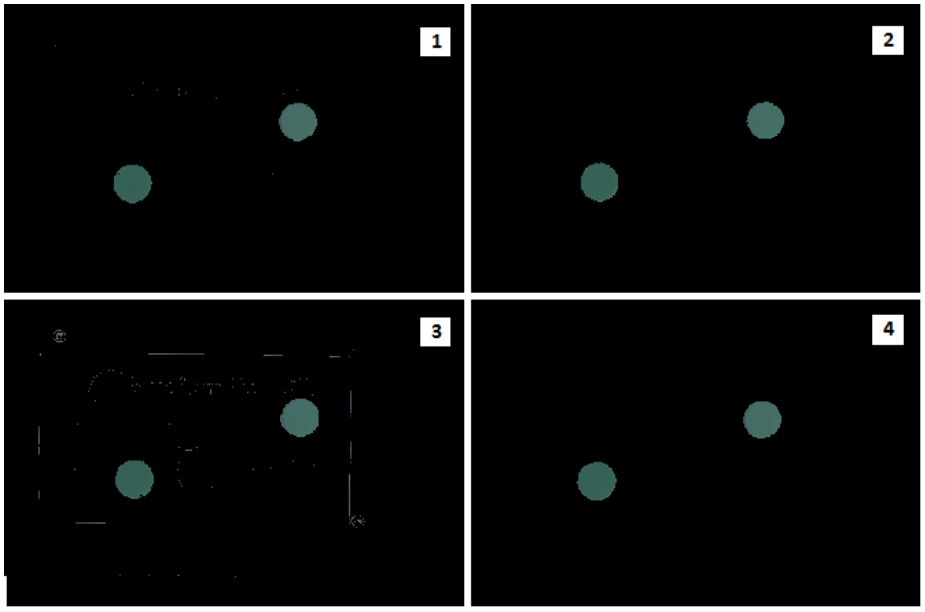


Abbildung 43: Farbfilterung grüne Scheiben mit (1) RGB, (2) HSV, (3) YCbCr und (4) L*a*b*.

4.3.2 Kreiserkennung

Mit dem MATLAB® Code aus der Programmcode 2 lassen sich die Ronden erfolgreich erkennen. In der Abbildung 44 wurden die erkannten Ronden mit der Farbe Rot markiert.

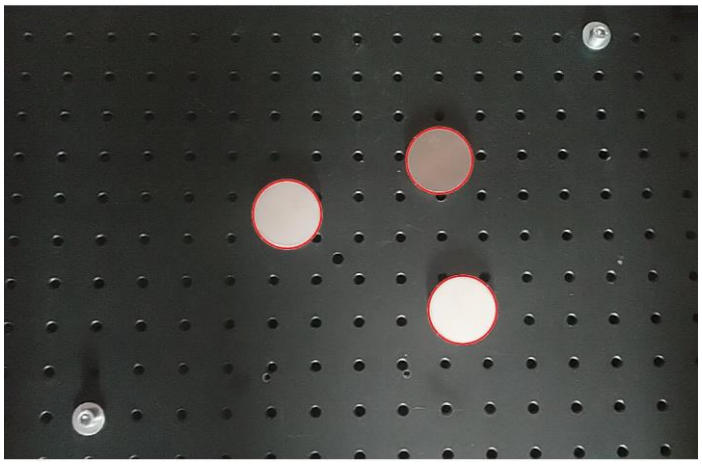


Abbildung 44: Ronden mit roter Markierung.

Mit der Funktion `imfindcircles` wurden zunächst beim größeren Radiusbereich auch kleine Kreise erkannt, siehe Abbildung 45. Nach dem Anpassen der Funktionsparameter auf einen eingeschränkten Radiusbereich wurde die Erkennung korrigiert.

Abbildung 45: Fehler bei der Formerkennung.

Bei den färbigen Scheiben werden alle Farben unterschieden und erkannt, siehe Abbildung 46. Es hat sich herausgestellt, dass die eingestellte Filterung bei anderen Lichtverhältnissen Probleme aufweist.

Abbildung 46: Nach Farben erkannte Scheiben.

Für die Positions-Schrauben wurde eine weitere Kreiserkennung mit angepasstem Radiusbereich durchgeführt, siehe Abbildung 47. Dadurch, dass der Schraubenkopfdurchmesser kleiner ist als von den Ronden, lassen sich die Schrauben von den Ronden in der Funktion `imfindcircles` unterscheiden.

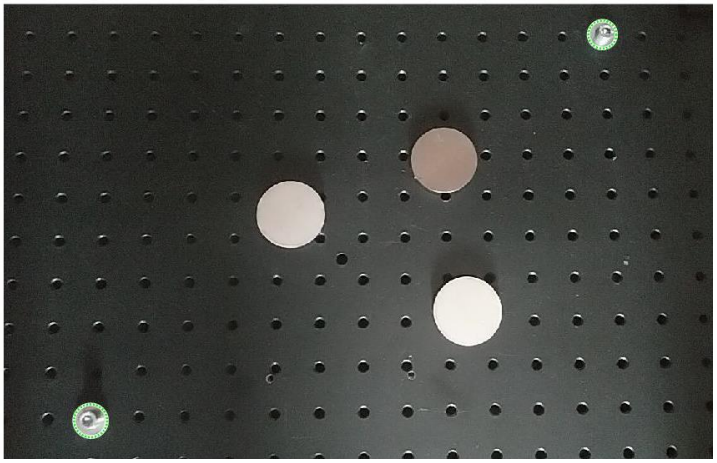


Abbildung 47: Detektierte Schrauben mit Markierung.

5 Roboterprogrammierung der Pick-and-Place Anwendung

Vor der ersten Inbetriebnahme wurden die Bewegungen von dem Delta-Roboter simuliert. Eine Robotersimulation hat den Vorteil, dass die programmierten Anwendungen ohne Abhängigkeit vom „echten“ Roboter getestet werden können. Nach dem Testen wird der Programmcode auf den realen Roboter übertragen. Die Programmierung erfolgt über die eigene Programmiersprache **RAPID** [33]. Zugleich ermöglicht eine Simulation die Analyse von Situationen, die in der realen Welt aufgrund von Kosten- oder Zeitmangel nicht so rasch aufgebaut werden können. Als Programmier- und Simulationsumgebung wurde die Software RobotStudio® [34] von der Fa. ABB verwendet.

5.1 Software Robot Studio®

Die Software RobotStudio® wird von der Fa. ABB bereitgestellt und enthält alle aktuellen ABB Robotermodelle. Somit kann eine graphische Simulation mit dem IRB 360 erstellt werden. In RobotStudio® befindet sich ein sogenannter **Virtual Controller** also eine virtuelle Steuerung mit einer exakten virtuellen Kopie der realen Steuerung. Somit lassen sich Simulationen einfacher an die reale Steuerung anpassen. Auf der virtuellen Steuerung wurde eine Pick-and-Place Anwendung erstellt und ausgeführt. Danach wurde das Programm in die reale Steuerung übertragen. Die Darstellung in Abbildung 48 zeigt den Ablauf von Simulation und realem System.

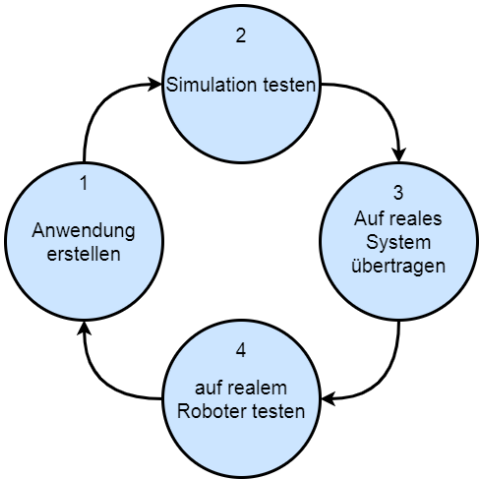


Abbildung 48: Ablauf von Simulation und realem System.

5.2 Virtuelles Modell

Eine Robotersimulation benötigt eine graphische Oberfläche, damit die Bewegungen von dem Roboter sichtbar sind. Die graphischen Darstellungen von dem Delta-Roboter und der Steuerung wurden von der RobotStudio® Bibliothek entnommen, siehe Abbildung 49.

Abbildung 49: Virtuelles Modell des IRB 360 und der Steuerung.

Die Aluminium-Zelle wurde mit den Dimensionen aus der Abbildung 15 in der 3D Design-Software Blender [35] gezeichnet. Des Weiteren wurde die Datei im **stl**-Format exportiert, um die 3D Zeichnung in RobotStudio® zu integrieren. In der Abbildung 50 wird die 3D Darstellung der Zelle in RobotStudio® gezeigt.

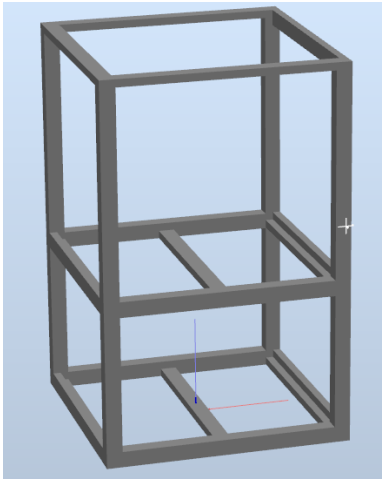


Abbildung 50: 3D Modell der Aluminium-Zelle.

Der Stahl-Rahmen wurde, wie auch die Aluminium-Zelle im Grafik-Tool Blender gezeichnet. Die dazugehörigen Dimensionen von dem Rahmen wurden aus der Abbildung 19 entnommen. In der Abbildung 51 ist der Stahl-Rahmen sichtbar.

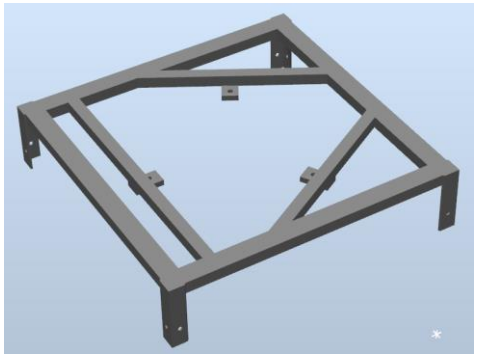


Abbildung 51: 3D Modell des Stahl-Rahmens.

Diese vier Komponenten wurden zu einer virtuellen Roboterzelle zusammengesetzt. Zusätzlich wurde ein einfaches Modell des Schaltschrankes gezeichnet. Des Weiteren wurde ein Modell des Sicherheits-Lichtvorhanges als **stl**-Format heruntergeladen und in das Gesamtmodell eingebaut. Das Vakuumsystem wurde nicht in die visuelle Darstellung einbezogen, da die Komponenten im Gegensatz zu den anderen Modellen relativ klein sind.

Die Abbildung 52 zeigt den gesamten Aufbau als 3D Modell in RobotStudio®.

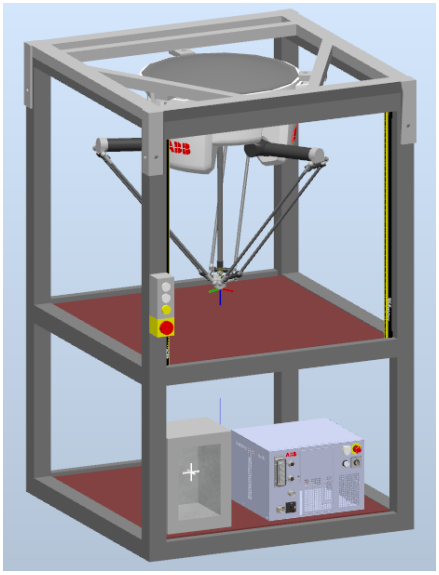


Abbildung 52: 3D Modell der Roboterzelle.

5.3 Konfiguration der Ein- und Ausgänge

Um die externen Komponenten anzusteuern, müssen diese in der Steuerung konfiguriert werden. In Tabelle 3 sind die Komponenten aufgelistet, die als Ein- sowie Ausgang konfiguriert werden müssen.

Tabelle 3: Konfiguration und Bezeichnung der Ein- und Ausgänge.

Baugruppe	Art	Signal
Obere Taste	Eingang	taste1
Schalter	Eingang	schalter1
Obere LED	Ausgang	led1
Schalter LED	Ausgang	led2
Magnetventil	Ausgang	ventil
Förderband	Ausgang	foerderband

Die Signale können im Programmcode verwendet werden, um den Zustand von den Eingängen zu lesen und bei den Ausgängen werden die Zustände gesetzt.

5.4 Pick-and-Place Anwendung

Die Bewegungen des Delta-Roboters wurden mit einer Pick-and-Place Anwendung auf Funktionalität überprüft. Dazu wurden als Werkstücke fünf Edelstahlronden mit je d=40mm und h=4mm von einem **Bereich M** zu einem **Bereich N** befördert. In der Abbildung 53 ist die Anordnung schematisch dargestellt.

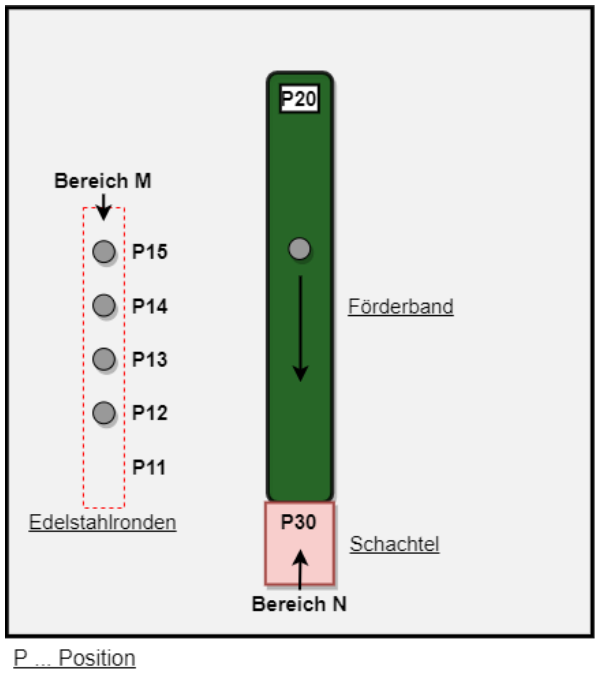


Abbildung 53: Übersicht der Pick-and-Place Anwendung.

Für diese Anwendung wurden zunächst die Positionen der Werkstücke bestimmt. Jede Position beinhaltet eine X, Y und Z Koordinate. Zum Beispiel befindet sich die erste Ronde auf der Position P11. Für alle Positionen werden zwei Z-Werte bestimmt, damit das Werkstück senkrecht aufgehoben wird, siehe Abbildung 54.

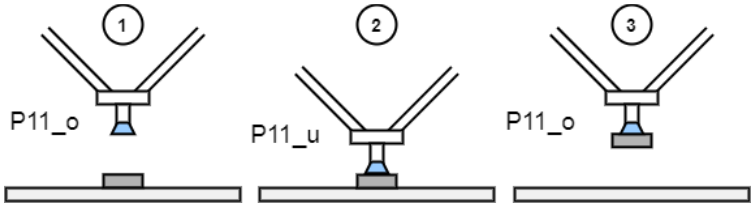


Abbildung 54: Ablauf beim Aufheben einer Ronde.

Die Anwendung kann über die obere Taste auf dem Bedienfeld gestartet werden. Danach wird die Edelstahlronde von dem Delta-Roboter aufgenommen und an die Position **P20** transportiert und abgelegt. Beim Transportieren der Ronde wird die obere LED eingeschalten. Die Ronde wird dann vom Förderband in die Schachtel befördert. Dieser Ablauf wiederholt sich, bis alle fünf Ronden im **Bereich N** in der Schachtel sind. In der Abbildung 55 ist das Flussdiagramm zu der Anwendung ersichtlich.

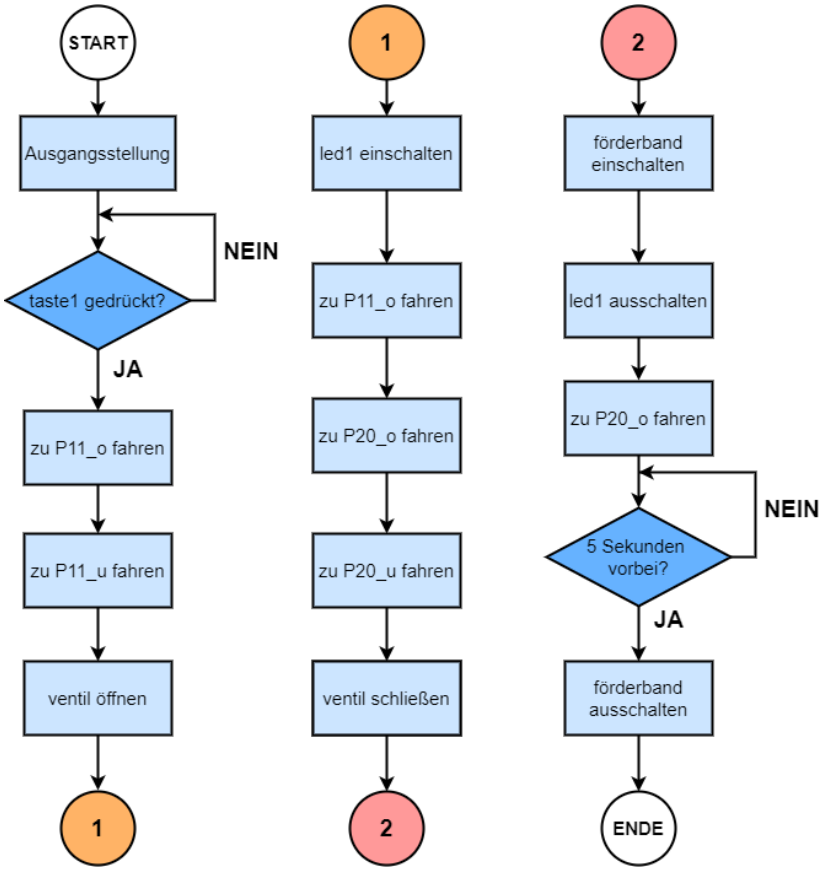


Abbildung 55: Flussdiagramm von einem Ablauf.

5.4.1 Pick-and-Place Programmierung

Im Programmcode wird mit einer **IF**-Abfrage von Signal **taste1** die Anwendung gestartet. Davor werden die benötigten Ausgänge auf null gesetzt und der Delta-Roboter fährt in die Ausgangstellung, siehe Programmcode 5. Danach werden die einzelnen Strecken abgefahren.

```
PROC main()

        SetDO ventil,0;
        SetDO led1,0;
        SetDO foerderband,0;
        Home;

    IF taste1=1 THEN
        Strecke_P11;
        Strecke_P20;
        Strecke_P12;
        Strecke_P20;
        Strecke_P13;
        Strecke_P20;
        Strecke_P14;
        Strecke_P20;
        Strecke_P15;
        Strecke_P20;
        Home;
        WaitTime 5;
        SetDO foerderband,0;
    ENDIF
ENDPROC
```

Programmcode 5: Hauptprogramm in RAPID.

Die Strecken wurden, wie in der Programmcode 6 definiert, in Prozesse eingeteilt. Um den Roboter zu bewegen, wurden zunächst die Positionen aus der Abbildung 53 in eine untere und obere Position unterteilt. Diese Ziele wurden mit dem Befehl `MoveL` ausgeführt, siehe RAPID-Befehlsreferenz von ABB [32]. Der Befehl `MoveL` beinhaltet die Zielposition, die Geschwindigkeit, die Zone, das Werkzeug und das Werkobjekt. Die Zone ist wie in Abbildung 56 ersichtlich, ein Bereich in dem der Roboter einen Bogen mit einem bestimmten Radius um einen Wegpunkt abfährt.

Der Radius wurde in der Programmierung mit **fine** definiert, sodass der Delta-Roboter jede Position genau anfährt.

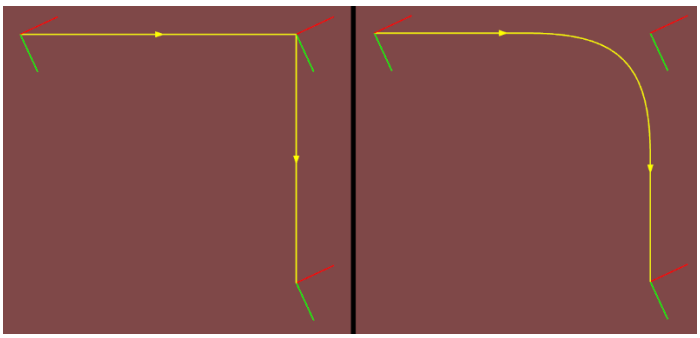


Abbildung 56: Roboter-Bahnkurven ohne Zone „fine“ (links) und mit Zone „z100“ (rechts).

Der Befehl `MoveL` bewegt das Werkzeug linear von der Startposition zur Zielposition. Die Edelstahlronde wurde auf der Position **P11_u** bzw. **P20_u** von dem Vakuumsauger aufgenommen bzw. abgelegt. Dazu wurde das Signal **ventil** High bzw. Low gesetzt. Damit der Transport der Ronde einem Bediener angezeigt wird, wurde das Signal **led1** beim Transport High gesetzt.

```
PROC Strecke_P12()
    MoveL P12_o,v600,fine,Sauger\WObj:=Arbeitsplatte;
    MoveL P12_u,v100,fine,Sauger\WObj:=Arbeitsplatte;
    SetDO ventil,1;
    SetDO led1,1;
    MoveL P12_o,v100,fine,Sauger\WObj:=Arbeitsplatte;
ENDPROC
```

Programmcode 6: Unterprogramm Strecke P12 in RAPID.

Das Unterprogramm **Strecke_P20** lässt den Roboter zum Ablageplatz fahren. Dabei wird an der Position **P20_o** das Förderband gestoppt und der Roboter fährt hinunter auf die Position **P20_u**. An der Position wird das Ventil deaktiviert, sodass die Ronde auf dem Förderband platziert wird, siehe Programmcode 7. Nach der erfolgten Ablage der Ronde wird die obere LED ausgeschalten und das Förderband wieder eingeschalten.

```
PROC Strecke_P20()
    MoveL P20_o,v600,fine,Sauger\WObj:=Arbeitsplatte;
    SetDO foerderband,0;
    MoveL P20_u,v100,fine,Sauger\WObj:=Arbeitsplatte;
    SetDO ventil,0;
    SetDO led1,0;
    SetDO foerderband,1;
    MoveL P20_o,v100,fine,Sauger\WObj:=Arbeitsplatte;
ENDPROC
```

Programmcode 7: Unterprogramm der Strecke P20 in RAPID.

Nach der letzten abgelegten Ronde läuft das Förderband noch fünf Sekunden nach und wird dann ausgeschalten, sodass die letzte Ronde in den Schachtel auf die Position **P30** gelangt. Der Delta-Roboter bewegt sich je nach Zielposition mit unterschiedlicher Geschwindigkeit. Wenn der Abstand zwischen den Positionen kurz ist, bewegt sich der Roboter mit einer Geschwindigkeit v=100mm/s. Bei einem längeren Abstand beträgt die Geschwindigkeit v=600mm/s. Das Förderband hat eine konstante Geschwindigkeit v=150mm/s, wenn es aktiv ist.
Im Anhang befindet sich der ganze RAPID-Programmcode.

5.4.2 Pick-And-Place Simulation

Bei der Simulation kann beobachtet werden, wie der Roboter sich bewegt und wie lange die Simulation dauert. In der Abbildung 57 befindet sich der Delta-Roboter an der Ausgangsstellung (1) und wartet auf ein Signal von der oberen Taste auf dem Bedienfeld.

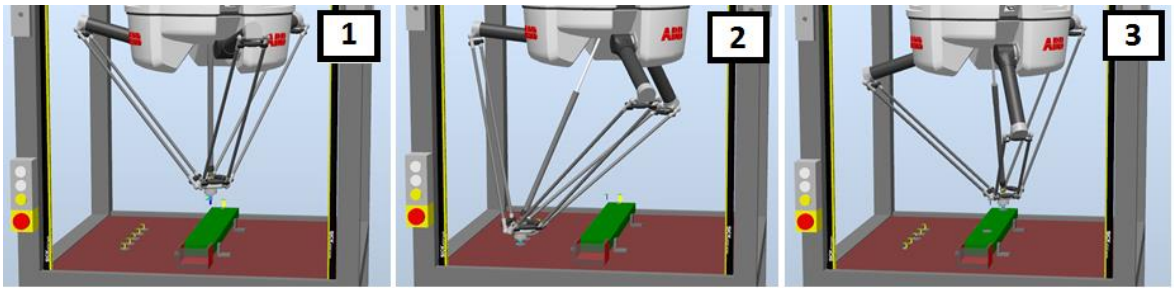


Abbildung 57: Graphische Darstellung vom Simulationsablauf.

Nach dem Signal von der **taste1** hat sich der Delta-Roboter an die Position der ersten Ronde fortbewegt und diese mit dem Sauger aufgehoben (2). Danach hat der Delta-Roboter die Ronde an die Ablageposition **P20_u** transportiert (3).

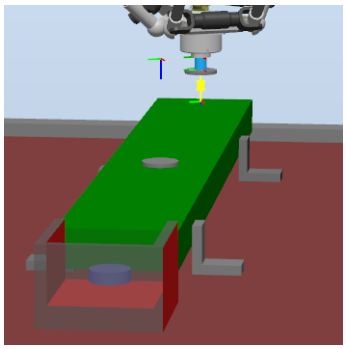


Abbildung 58: Bereich N in der Simulation mit den Ronden.

In der Abbildung 58 wird eine Nahaufnahme der abgelagerten Ronden im **Bereich N** in der Schachtel dargestellt. Nach dem erfolgreichen Testen der Simulation ist der Code bereit um auf die reale Steuerung übertragen zu werden.

Die Simulation wurde auf YouTube hochgeladen und kann dort angesehen werden [1].

5.4.3 Pick-and-Place Taktzeitanalyse

Nach dem die Simulation abgeschlossen war, wurde die Zeit der Bewegung zwischen den Positionen von den Roden bis zum Förderband analysiert. In RobotStudio® lässt sich die Zeit in der Simulation stoppen. Dabei wurden zum Stoppen der Zeit die Positionen **P11_u, P12_u, P13_u, P14_u, P15_u** und **P20_u** als Referenz genommen.

In dem Abbildung 59 wird die benötigte Zeit t=1,632s, um von der Position **P11_u** an die Position **P20_u** zu gelangen, gezeigt.

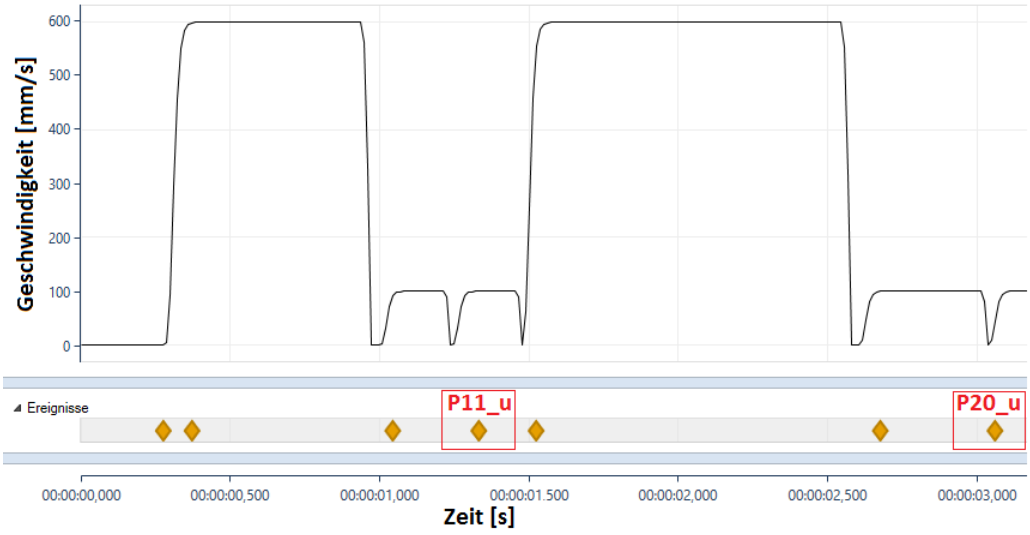


Abbildung 59: Diagramm einer Zeitanalyse der Bewegung von P11_u nach P20_u.

Die restlichen Zeitaufnahmen der Positionen wurden in die Tabelle 4 eingetragen.

Tabelle 4: Zeiten zwischen den Positionen.

Startposition	Zielposition	Zeit – Zone „fine“	Zeit – Zone „z100“
Ausgangsstellung	P11_u	0,792s	0,756s
P11_u	P20_u	1,632s	1,440s
P20_u	P12_u	1,923s	1,824s
P12_u	P20_u	1,440s	1,344s
P20_u	P13_u	1,827s	1,728s
P13_u	P20_u	1,344s	1,248s
P20_u	P14_u	1,731s	1,632s
P14_u	P20_u	1,248s	1,152s
P20_u	P15_u	1,635s	1,536s
P15_u	P20_u	1,152s	1,056s
		+5s	+5s
	Gesamtzeit:	**19,724s**	**18,716s**

Im Programmcode wurden die Zonen von **fine** auf **z100** geändert. Danach wurde die Zeit der einzelnen Strecken wieder gestoppt.

Der Unterschied zwischen den Zeiten in Tabelle 4 zeigt, dass die Zone einen großen Einfluss auf die Zeit hat. In einer Produktion könnte die Zone verwendet werden, um die Produktionszeit anzupassen.

Die gesamte Simulation mit der Zone **fine** dauerte t=19,724s und mit der Zone **z100** dauerte sie t=18,716s.

5.5 Socket Verbindung

Nach der getesteten Simulation kann eine Socket Verbindung erstellt werden, um die Daten der Werkstück-Koordinaten der Objekterkennung von MATLAB® in Echtzeit in die Robotersteuerung zu übertragen.

Ein Socket ist an eine Portnummer von dem Computer verbunden [40]. Für eine Verbindung zwischen zwei Computern wird ein Server und ein Client benötigt. In unserem Fall ist der Server die Roboter-Steuerung und der Client wird in MATLAB® erstellt. Danach wartet der Server bis der Client eine Verbindungsanforderung sendet, siehe Abbildung 60.

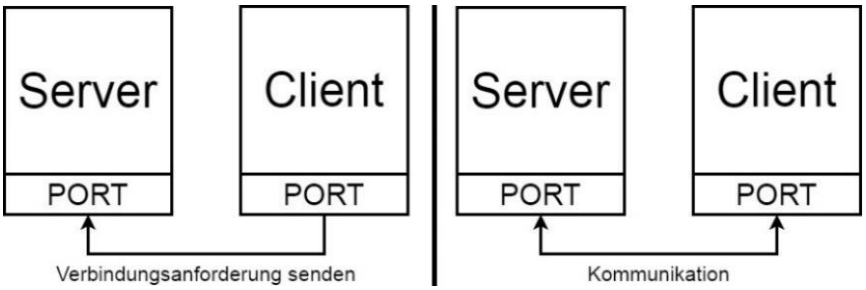


Abbildung 60: Socket Kommunikation.

Für die Verbindung wird der Hostname und die Portnummer benötigt. In dem Programmcode 8 ist der funktionelle Syntax der Socket-Kommunikation in RAPID und MATLAB® Script zu dargestellt mit dem die Positionsdaten der Werkstücke an die Robotersteuerung gesendet werden.

```
PROC Socket_Verbindung()
    !Kommunikation erstellen
    SocketCreate server;
    SocketBind server, "127.0.0.1",55000;
    SocketListen server;
    SocketAccept server, client;

    !Daten senden
    SocketSend client, \Str:="starten";

    !Daten empfangen
    SocketReceive client,\RawData:=data \Time:=WAIT_MAX;
    UnpackRawBytes data,1,msg,\ASCII:=15;

    !Kommunikation schließen
    SocketClose server;
ENDPROC
```

```
%ip-Adresse: 10.20.21.1
%Portnummer: 55000
tc = tcpip('10.20.21.1',55000);
%Kommunikation öffnen.
fopen(tc);

%Daten vom Server empfangen.
message= fread(tc);

%Daten an den Server senden.
fwrite(tc,msg);

%Kommunikation schließen
fclose(tc);
```

Programmcode 8: Socket-Kommunikation: Server in RAPID (links) und Client in MATLAB® (rechts).

6 Zusammenfassung

In dieser Arbeit wurde eine Roboterzelle für den Delta-Roboter IRB 360 erfolgreich aufgebaut. Die Zelle ist stabil und der Roboter kann in Betrieb genommen werden. Der Lichtvorhang und die Not-Aus Taste stellen die erforderlichen Sicherheitsfunktionen dar. Mit dem Vakuumsystem lassen sich leichte Werkstücke mit glatter Oberfläche wie die Edelstahlronden als auch die bunten Scheiben aufheben. Nach kurzer Wartezeit zum Ansaugen der Werkstücke, kann der Roboter das Werkstück aufnehmen und die gewünschte Bewegung hin zur Ablageposition am Förderband ausführen. Das Förderband läuft mit der voreingestellten Geschwindigkeit die manuell im Schaltschrank geändert werden kann.

Zur sicheren Aufnahme der Werkstücke wurde eine Objekterkennung mittels sowohl einer monochromen Cognex Kamera als auch einer einfachen Webcam erfolgreich umgesetzt. Während die industriellen Cognex-Kamera bereits integrierte Objekterkennung ermöglich, wurde die Webcam mittels spezieller MATLAB® Funktionen zur Bildbearbeitung und Objekterkennung genutzt. Mit dem **Color Thresholder** ist die effiziente Farbfilterung der Bilder in unterschiedlichen Farbräumen möglich. Weiters wurde die Objekterkennung aufgrund geometrischer Formen durchgeführt. Hier zeigte sich, dass die jeweiligen Parameter dieser Funktion je nach Werkstück angepasst werden müssen beziehungsweise für andere Formen weiter adaptiert werden müssen.

Die gesamte Roboterzelle wurde für die erfolgte Simulation komplett in 3D virtualisiert . Dadurch wurde eine visuelle Darstellung der Bewegungsabläufe vom Delta-Roboter IRB 360 in der Roboterzelle mit RobotStudio® erfolgreich erstellt. In der Simulation wurde eine Pick-and-Place Anwendung durchgeführt. Bei der Simulation wurde außerdem eine Taktzeitanalyse für einen vordefinierten Bewegungsablauf durchgeführt. Dabei wurden unterschiedliche Einstellungen der Zonen der Wegpunkte analysiert. Dadurch konnte die Bearbeitungszeit um insgesamt t=1,006s verkürzt werden.

Literaturverzeichnis

[1] M. Algan, “Pick-and-Place Simulation des IRB 360 Flexpickers von ABB,” Simulationsvideo zur Bachelor-Arbeit, FH Wr. Neustadt, 2020. [online] (available: 19.Mai 2020) https://www.youtube.com/watch?v=12U0iN5cQhc

[2] FANUC, „M-2iA/3SL“. [online] (available: 21.März 2020) https://www.fanuc.eu/at/de

[3] Omron, „Hornet“. [online] (available: 25.März 2020) https://industrial.omron.at/de/home

[4] igus, „Technische Dokumentation drylin Delta Roboter“. [online] (available: 25. März 2020) https://www.igus.at

[5] Stäubli, „FAST picker TP80“. [online] (available: 25. März 2020) https://www.staubli.com/de-at/

[6] ABB, „IRB 1200“. [online] (available: 25. März 2020) https://new.abb.com/at

[7] A. Markis, H. Montenegro, „Sicherheit in der Mensch-Roboter-Kollaboration“, Frauenhofer Austria, Wien, 2016. [online] (available: 21.Mai 2020) https://www.fraunhofer.at/content/dam/austria/documents/WhitePaperTUEV/White%20Paper_Sicherheit_MRK_Ausgabe%201.pdf

[8] KUKA, „Sensitive robotics _LBR iiwa“. [online] (available: 29. März 2020) https://www.kuka.com/de-at

[9] ABB, „IRB 360 FlexPicker“. [online] (available: 29. März 2020) https://new.abb.com/at

[10] J.M. Sebastian, A. Traslosheros, “Parallel Robot High Speed Object Tracking”, Universidad Politecnica de Madrid, Spanien, 2007. [online] (available: 30. März 2020) https://www.researchgate.net/publication/221472655_Parallel_Robot_High_Speed_Object_Tracking

[11] K. Daniel, R. Annika, “Simulation and design of an orientation mechanism for assembly systems”, Leibniz Universität Hannover, Institute of Assemby Technology, Garbsen, Germany, 2015. [online] (available: 30. März 2020) https://www-sciencedirect-com.wn.idm.oclc.org/science/article/pii/S2212827116004042

[12] S. Sarangpure, K. Chaudhari and Y. Salame, "Design and Implementation of Low-Cost Closed Loop Pick and Place Assembly," 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2019. [online] (available: 30. März 2020) https://ieeexplore-ieee-org.wn.idm.oclc.org/document/8987743

[13] A. Rouhollahi, M. Azmoun and M. T. Masouleh, „Experimental study on the visual servoing of a 4-DOF parallel robot for pick-and-place purpose," 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), Kerman, 2018. [online] (available: 1.April 2020) https://ieeexplore-ieee-org.wn.idm.oclc.org/document/8336617

[14] Qt Group, „About Qt“. [online] (available: 21. Mai 2020) https://wiki.qt.io/About_Qt

[15] L. Angel, J. Viola, „Fractional order PID for tracking control of a parallel robotic manipulator type delta”, Universidad Pontificia Bolivariana, Colombia, 2017. [online] (available: 1.April 2020) https://www-sciencedirect-com.wn.idm.oclc.org/science/article/pii/S0019057818301563

[16] ABB, „Produkthandbuch IRB 360“. [online] (available: 9.April 2020) https://new.abb.com/at

[17] ABB, „Produkthandbuch IRC5 Compact“. [online] (available: 9.April 2020) https://new.abb.com/at

[18] ABB, „Bedienungsanleitung IRC5 mit FlexPendant“. [online] (available: 9.April 2020) https://new.abb.com/at

[19] SICK Sensor Intelligence, „C4C-SA18030A10000, C4C-EA18030A10000 deTec Sicherheits-Lichtvorhänge”. [online] (available: 10.April 2020) https://www.sick.com

[20] SICK Sensor Intelligence, „UE48-2OS2D2 Sicherheitsschaltgeräte“. [online] (available: 9.April 2020) https://www.sick.com

[21] EATON, „Datenblatt – M22-PV/KC11/IY“. [online] (available: 10.April 2020) https://www.eaton.com

[22] Vetter Kleinförderband, „FR-40-120”. [online] (available: 11.April 2020) https://www.vetter-band.de

[23] Vetter Kleinförderband, „VSW-11”. [online] (available: 11.April 2020) https://www.vetter-band.de

[24] SMC, „3 Port Solenoid ValveDirect Operated Poppet Type“. [online] (available: 13.April 2020) https://www.smc.eu/de-at

[25] KNF, „MEMBRAN-VAKUUMPUMPE LABOPORT® N 816.3 KT.18”. https://www.knf.de

[26] Festo, „Suction cups VAS/VASB”. [online] (available: 13.April 2020) https://www.festo.com

[27] RS-Components, „Pneumatik-Verteiler”. [online] (available: 13.April 2020) https://at.rs-online.com

[28] ABB, „Product specification Integrated Vision“. [online] (available: 14.April 2020) https://new.abb.com

[29] ABB, „Integrated Vision Kamerageführte Robotertechnik für die Industrie“. [online] (available: 14.April 2020) https://new.abb.com/at

[30] ABB, „Anwendungshandbuch Conveyor Tracking“. [online] (available: 14.April 2020) https://new.abb.com/at

[31] ABB, „Conveyor Tracking Module Product Management, ABB Robotics“. [online] (available: 14.April 2020) https://new.abb.com

[32] ABB, „System pro E control Gemini. Low voltage insulating switchboards “. [online] (available: 16.April 2020) https://new.abb.com

[33] ABB, „Technisches Referenzhandbuch – RAPID Instruktionen, Funktionen und Datentypen“. [online] (available: 25.April 2020) https://new.abb.com/at

[34] ABB, „Bedienungsanleitung RobotStudio“. [online] (available: 25.April 2020) https://new.abb.com/at

[35] Blender, „The Software“. [online] (available: 25.April 2020) https://www.blender.org

[36] Logitech, „Getting started with Logitech® HD Webcam C615”. [online] (available: 10. Mai 2020) https://www.logitech.com

[37] M. Rauter, „Bildverarbeitung und Objekterkennung“, Vorlesungsunterlagen, FH Wiener Neustadt, 2020.

[38] Wikipedia, „YCbCr-Farbmodell“. [online] (available: 8.Mai 2020) https://de.wikipedia.org/wiki/YCbCr-Farbmodell

[39] MATLAB®, Image Processing Toolbox. [online] (available: 8.Mai 2020) https://de.mathworks.com/help/images/index.html?s_tid=CRUX_lftnav

[40] Oracle®, „What Is a Socket?“. [online] (available: 8.Mai 2020) https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html

Formelzeichen

Bezeichnung	Symbol	Einheit
Breite	b	[m]
Durchmesser	d	[m]
Wahrscheinlichkeit	e	[1]
Kraft	F	[N]
Höhe	h	[m]
Länge	l	[m]
Traglast	m	[kg]
Gesamtmasse	M	[kg]
Picks pro Minute	p	[min^{-1}]
Radius	r	[m]
Distanz	s	[m]
Zeit	t	[s]
Spannung	U	[V]
Geschwindigkeit	v	[m/s]
Genauigkeit	w	[m]
Auflösung	B	[px]

Anhang

```
MODULE Module1
    !Alle Positionen für die Anwendung
    CONST robtarget Ausgangstellung:=[[0,0,-
1740],[0,0.976,0.216,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget P11_o:=[[150,-325,-
1900],[0,0.976,0.216,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget P11_u:=[[150,-325,-
1921],[0,0.976,0.216,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget P12_o:=[[50,-325,-
1900],[0,0.976,0.216,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget P12_u:=[[50,-325,-
1921],[0,0.976,0.216,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget P13_o:=[[-50,-325,-
1900],[0,0.976,0.216,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget P13_u:=[[-50,-325,-
1921],[0,0.976,0.216,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget P14_o:=[[-150,-325,-
1900],[0,0.976,0.216,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget P14_u:=[[-150,-325,-
1921],[0,0.976,0.216,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget P15_o:=[[-250,-325,-
1900],[0,0.976,0.216,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget P15_u:=[[-250,-325,-
1921],[0,0.976,0.216,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget P20_o:=[[-400,0,-
1810],[0,0.976,0.216,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget P20_u:=[[-400,0,-
1850],[0,0.976,0.216,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    !*********************************************************
    !
    ! Modul:  Programmcode
    !
    ! Beschreibung:
    !   Dies ist eine Pick-And-Place Anwendung für
Demonstrationszwecke.
```

```
    !
    ! Autor: Mustafa Algan
    !
    ! Version: 1.0
    !
    !*************************************************************

    !*************************************************************
    ! Prozedur main
    !
    !   Dies ist die Startroutine 'main', die den Ablauf der Task
steuert.
    !*************************************************************

    VAR socketdev server;
    VAR socketdev client;
    VAR string msg;
    VAR rawbytes data;

    PROC main()
        !setzt alle benötigten Ausgänge 0
        SetDO ventil,0;
        SetDO led1,0;
        SetDO foerderband,0;
        !Roboter geht in die Ausgangstellung.
        Home;

        !Path_10;               !Test Bahn - fährt alle Positionen
einzeln ab.

        IF taste1=1 THEN
            !Wenn die Taste1 betätigt wird, beginnt die Anwendung
            !Roboter fährt die Strecken ab.
            Strecke_P11;
            Strecke_P20;
            Strecke_P12;
            Strecke_P20;
            Strecke_P13;
            Strecke_P20;
            Strecke_P14;
            Strecke_P20;
            Strecke_P15;
            Strecke_P20;
            Home;
            !System wartet 5 Sekunden, damit die letzte Ronde in
die Schatel gelangt.
            WaitTime 5;
            !Foerderband ausschalten.
```

```
            SetDO foerderband,0;
        ENDIF
    ENDPROC

    !Position der Ausgangsstellung.
    PROC Home()
        MoveL
Ausgangstellung,V200,fine,Sauger\WObj:=Arbeitsplatte;
    ENDPROC

    !Position der ersten Ronde bis zur fünften Ronde P11-P15
    !wurden in einzelne Untergrogramme geschrieben.
    PROC Strecke_P11()
        MoveL P11_o,v600,fine,Sauger\WObj:=Arbeitsplatte;
        MoveL P11_u,v100,fine,Sauger\WObj:=Arbeitsplatte;
        SetDO ventil,1;
        !Ventil wird eingeschalten.
        SetDO led1,1;
        !LED1 wird eingeschalten.
        MoveL P11_o,v100,fine,Sauger\WObj:=Arbeitsplatte;
    ENDPROC

    PROC Strecke_P12()
        MoveL P12_o,v600,fine,Sauger\WObj:=Arbeitsplatte;
        MoveL P12_u,v100,fine,Sauger\WObj:=Arbeitsplatte;
        SetDO ventil,1;
        SetDO led1,1;
        MoveL P12_o,v100,fine,Sauger\WObj:=Arbeitsplatte;
    ENDPROC

    PROC Strecke_P13()
        MoveL P13_o,v600,fine,Sauger\WObj:=Arbeitsplatte;
        MoveL P13_u,v100,fine,Sauger\WObj:=Arbeitsplatte;
        SetDO ventil,1;
        SetDO led1,1;
        MoveL P13_o,v100,fine,Sauger\WObj:=Arbeitsplatte;
    ENDPROC

    PROC Strecke_P14()
        MoveL P14_o,v600,fine,Sauger\WObj:=Arbeitsplatte;
        MoveL P14_u,v100,fine,Sauger\WObj:=Arbeitsplatte;
        SetDO ventil,1;
        SetDO led1,1;
        MoveL P14_o,v100,fine,Sauger\WObj:=Arbeitsplatte;
    ENDPROC

    PROC Strecke_P15()
        MoveL P15_o,v600,fine,Sauger\WObj:=Arbeitsplatte;
```

```
    MoveL P15_u,v100,fine,Sauger\WObj:=Arbeitsplatte;
    SetDO ventil,1;
    SetDO led1,1;
    MoveL P15_o,v100,fine,Sauger\WObj:=Arbeitsplatte;
ENDPROC

!Ablageposition auf dem Foerderband.
PROC Strecke_P20()
    MoveL P20_o,v600,fine,Sauger\WObj:=Arbeitsplatte;
    !Foerderband ausschalten.
    SetDO foerderband,0;
    MoveL P20_u,v100,fine,Sauger\WObj:=Arbeitsplatte;
    !Ventil wird bei jedem Ablegen ausgeschalten.
    SetDO ventil,0;
    !LED1 wird nach der Transportation ausgeschalten.
    SetDO led1,0;
    !Foerderband einschalten.
    SetDO foerderband,1;
    MoveL P20_o,v100,fine,Sauger\WObj:=Arbeitsplatte;
ENDPROC

PROC Socket_Verbindung()
    !Kommunikation erstellen.
    SocketCreate server;
    SocketBind server,"127.0.0.1",55000;
    SocketListen server;
    SocketAccept server,client;

    !Daten senden.
    SocketSend client,\Str:="starten";

    !Daten empfangen.
    SocketReceive client,\RawData:=data\Time:=WAIT_MAX;
    UnpackRawBytes data,1,msg,\ASCII:=15;

    !Kommunikation schließen.
    SocketClose server;
ENDPROC
!Testlauf von allen Positionen.
PROC Path_10()
    MoveL P11_o,v600,z100,Sauger\WObj:=Arbeitsplatte;
    MoveL P11_u,v100,z100,Sauger\WObj:=Arbeitsplatte;
    MoveL P11_o,v100,z100,Sauger\WObj:=Arbeitsplatte;
    MoveL P20_o,v600,z100,Sauger\WObj:=Arbeitsplatte;
    MoveL P20_u,v100,z100,Sauger\WObj:=Arbeitsplatte;
    MoveL P20_o,v100,z100,Sauger\WObj:=Arbeitsplatte;
    MoveL P12_o,v600,z100,Sauger\WObj:=Arbeitsplatte;
    MoveL P12_u,v100,z100,Sauger\WObj:=Arbeitsplatte;
```

```
        MoveL P12_o,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P20_o,v600,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P20_u,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P20_o,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P13_o,v600,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P13_u,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P13_o,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P20_o,v600,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P20_u,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P20_o,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P14_o,v600,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P14_u,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P14_o,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P20_o,v600,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P20_u,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P20_o,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P15_o,v600,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P15_u,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P15_o,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P20_o,v600,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P20_u,v100,z100,Sauger\WObj:=Arbeitsplatte;
        MoveL P20_o,v100,z100,Sauger\WObj:=Arbeitsplatte;
    ENDPROC
ENDMODULE
```

Anhang 1: Programmcode - Pick-and-Place Anwendung in RAPID.

```
clc; clear all; close all;
filename1 = ...
'Z:\_FH Wiener Neustadt\BAC2\Delta-Roboter_Zelle_-
_Files\04_Dobot_Studio';
filename2 = ...
'Z:\_FH Wiener Neustadt\BAC2\Bildbearbeitung\BAC_MATLAB\hier.txt';

sensy_1 = 0.80;
sensy_2 = 0.85;

%----------------------WEBCAM----------------------------%
%
% % webcamlist            //zeigt alle angeschlossenen Webcams.
% cam = webcam('USB Camera');
% preview(cam);
% %camzoom(2)
% pause(1);
% cam.Resolution = '800x600';
% img3 = snapshot(cam);
% closePreview(cam);
% %imshow(img);
% %cam                    //Webcam Eigenschaften anzeigen.

%Zum Testen Bilder benutzt.
img1 = imread('img1.jpg');
img2 = imread('Test1.bmp');

imshow(img1);imshow(img2);
%% ---------------------BILDBEARBEITUNG----------------------%

%  Ronde filtern:
        [y,rmat] = RondeFilter(img1);

        BWr = im2bw(rmat,.1);
        %imshow(BWr);

        clear_BWr = imfill(BWr,'holes');
        %imshow(clear_BWr);

        se=strel('disk',10);
        circles_BWr =imopen(clear_BWr,se);
        %imshow(circles_BWr);

        figure(2);
        imshowpair(clear_BWr,circles_BWr,'montage');

% Färbige Scheiben:
        [y,rot] = rot1(img2);
        [y,gruen] = gruen1(img2);
        [y,blau] = blau1(img2);

        BWrot = im2bw(rot,.1);
        BWgruen = im2bw(gruen,.1);
        BWblau = im2bw(blau,.1);
        %imshow(BWr);

        clear_BWrot = imfill(BWrot,'holes');
```

```
        clear_BWgruen = imfill(BWgruen,'holes');
        clear_BWblau = imfill(BWblau,'holes');
        %imshow(clear_BWrot);

        se=strel('disk',5);
        circles_BWrot =imopen(clear_BWrot,se);
        se=strel('disk',5);
        circles_BWgruen =imopen(clear_BWgruen,se);
        se=strel('disk',10);
        circles_BWblau =imopen(clear_BWblau,se);
        figure(2);
        imshowpair(clear_BWgruen,circles_BWgruen,'montage');
%% --------------------OBJEKTE ERKENNEN-----------------------%

% Ronde finden
[center1,radius1] = imfindcircles(circles_BWr,[45 55],'Method',...
                    'TwoStage','Sensitivity',sensy_1);

c1 = isempty(center1);
if c1==1
    center1=[0 0];
    center1cpy = [0 0];
    radius1cpy = 0;
else
    center1cpy = center1(:,:);
    radius1cpy = radius1(:);
end
center1Int = round(center1,0);

figure(5);
imshow(img1);
viscircles(center1cpy, radius1cpy,'EdgeColor','r');

% färbige Scheiben finden
[centerRot,radiusRot] = imfindcircles(circles_BWrot,[10 100],...
    'Method','TwoStage','Sensitivity',sensy_1);
[centerGruen,radiusGruen] = imfindcircles(circles_BWgruen,[10 100],...
    'Method','TwoStage','Sensitivity',sensy_1);
[centerBlau,radiusBlau] = imfindcircles(circles_BWblau,[10 100],...
    'Method','TwoStage','Sensitivity',sensy_1);

figure(6);
imshow(img2);
viscircles(centerRot, radiusRot,'EdgeColor','b');
viscircles(centerGruen, radiusGruen,'EdgeColor','r');
viscircles(centerBlau, radiusBlau,'EdgeColor','g');
%%------------------- Schrauben suche ---------------------

        [y,bmat] = Schrauben(img1);

        BWscrew = im2bw(bmat,.1);

        clear_BWscrew = imfill(BWscrew,'holes');

        se=strel('disk',2);
        circles_BWscrew =imopen(clear_BWscrew,se);
```

```
        % imshow(circles_BWscrew);

        [centerSchraube,radiusSchraube] = imfindcircles(...
            circles_BWscrew,[15 28],'Method','TwoStage',...
            'Sensitivity',0.75)
        screws = sort(centerSchraube);

        centerSchraubeCpy = centerSchraube(:,:);
        radiusSchraubeCpy = radiusSchraube(:);
        imshow(img1);
        viscircles(centerSchraubeCpy, radiusSchraubeCpy,'EdgeColor',...
            'g','LineStyle',':');

%% ------------------- Koordinaten Webcam --------------------
%Webcam Koordinaten
camy =[screws(1, 2), screws(2, 2)];  %[48, 499]
camx =[screws(1, 1), screws(2, 1)];  %[39, 785]

camdx =camx(2)- camx(1);
camdy =camy(2)- camy(1);

% Roboter-Koordinaten
rox=[ -150, 150];
roy=[-400, -167];

deltax=rox(2)-rox(1);
deltay=roy(2)-roy(1);

faktorx = deltax/camdx;
faktory = deltay/camdy;

%% Koordinaten-Transformation Ronde
% Koordinaten-Transformation Ronde
faktorx = deltax/camdx;
faktory = deltay/camdy;

X = center1Int(:,2);                    % Koordinaten X
Y = center1Int(:,1);                    % Koordinaten Y

Xtrans = rox(1)+(X-camx(1))*faktorx;    %+186+(X-46)*faktorx;
Ytrans = roy(1)+(Y-camy(1))*faktory;    %-114+(Y-48)*faktory;
Koor=[Xtrans;  Ytrans];                    % Ausgabe Roboter-Koordinaten

n=size(center1 );
n=n(1);

X_Y=[Koor(1:n),Koor(n+1:2*n)];

%ip-Adresse: 10.20.21.1
%Portnummer: 55000
tc = tcpip('10.20.21.1',55000);
%Kommunikation öffnen.
fopen(tc);

%Daten vom Server empfangen.
message= fread(tc);
```

```
%Daten an den Server senden.
fwrite(tc,Koor);

%Kommunikation schließen
fclose(tc);

%end
% File-Save zur Kontrolle
fileID = fopen(filename2, 'w');       % File öffnen
fileSpec = '%2d\n';                  % File Format
fprintf(fileID,fileSpec,Koor);         % File schreiben
fclose(fileID);                      % File schließen
```

Anhang 2: Programmcode - Objekterkennung Main in MATLAB.

```
function [BW,maskedRGBImage] = createMask(RGB)
%createMask  Threshold RGB image using auto-generated code from
colorThresholder app.
%  [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using
%  auto-generated code from the colorThresholder app. The colorspace and
%  range for each channel of the colorspace were set within the app. The
%  segmentation mask is returned in BW, and a composite of the mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 12-May-2020
%------------------------------------------------------


% Convert RGB image to chosen color space
I = rgb2hsv(RGB);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.567;
channel1Max = 0.600;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.469;
channel2Max = 0.718;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.277;
channel3Max = 0.590;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

% Initialize output masked image based on input image.
maskedRGBImage = RGB;

% Set background pixels where BW is false to zero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

end
```

Anhang 3: Programmcode - Farbfilterung für Blau in MATLAB.

```
function [BW,maskedRGBImage] = gruen1(RGB)
%createMask  Threshold RGB image using auto-generated code from
colorThresholder app.
%  [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using
%  auto-generated code from the colorThresholder app. The colorspace and
%  range for each channel of the colorspace were set within the app. The
%  segmentation mask is returned in BW, and a composite of the mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 31-Jul-2018
%------------------------------------------------------


% Convert RGB image to chosen color space
I = rgb2hsv(RGB);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.238;
channel1Max = 0.540;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.070;
channel2Max = 1.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 1.000;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

% Initialize output masked image based on input image.
maskedRGBImage = RGB;

% Set background pixels where BW is false to zero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

end
```

Anhang 4: Programmcode - Farbfilterung für Grün in MATLAB.

```
function [BW,maskedRGBImage] = createMask(RGB)
%createMask  Threshold RGB image using auto-generated code from
colorThresholder app.
%  [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using
%  auto-generated code from the colorThresholder app. The colorspace and
%  range for each channel of the colorspace were set within the app. The
%  segmentation mask is returned in BW, and a composite of the mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 12-May-2020
%------------------------------------------------------


% Convert RGB image to chosen color space
I = rgb2hsv(RGB);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.937;
channel1Max = 0.084;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.277;
channel2Max = 1.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 1.000;

% Create mask based on chosen histogram thresholds
sliderBW = ( (I(:,:,1) >= channel1Min) | (I(:,:,1) <= channel1Max) ) &
...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

% Initialize output masked image based on input image.
maskedRGBImage = RGB;

% Set background pixels where BW is false to zero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

end
```

Anhang 5: Programmcode - Farbfilterung für Rot in MATLAB.

```
function [BW,maskedRGBImage] = createMask(RGB)
%createMask  Threshold RGB image using auto-generated code from
colorThresholder app.
%  [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using
%  auto-generated code from the colorThresholder app. The colorspace and
%  range for each channel of the colorspace were set within the app. The
%  segmentation mask is returned in BW, and a composite of the mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 11-May-2020
%------------------------------------------------------


% Convert RGB image to chosen color space
I = rgb2ycbcr(RGB);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 128.000;
channel1Max = 255.000;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.000;
channel2Max = 255.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 111.000;
channel3Max = 149.000;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

% Initialize output masked image based on input image.
maskedRGBImage = RGB;

% Set background pixels where BW is false to zero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

end
```

Anhang 6: Programmcode - Schrauben-Filterung in MATLAB.

```
function [BW,maskedRGBImage] = createMask(RGB)
%createMask  Threshold RGB image using auto-generated code from
colorThresholder app.
%  [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using
%  auto-generated code from the colorThresholder app. The colorspace and
%  range for each channel of the colorspace were set within the app. The
%  segmentation mask is returned in BW, and a composite of the mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 11-May-2020
%------------------------------------------------------


% Convert RGB image to chosen color space
I = rgb2hsv(RGB);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.000;
channel1Max = 1.000;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.000;
channel2Max = 1.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.503;
channel3Max = 1.000;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

% Initialize output masked image based on input image.
maskedRGBImage = RGB;

% Set background pixels where BW is false to zero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

end
```

Anhang 7: Programmcode - Ronden-Filterung in MATLAB.

ROBOTICS

IRB 360 FlexPicker™
Industrieroboter

Anwendungsbereiche

Montage
Materialhandhabung
Pick & Place
Verpacken

Seit mehr als 15 Jahren ist der IRB 360 FlexPicker der führende Industrieroboter in der Pick-&-Place-Technologie. Der Roboter verfügt über eine herausragende Bewegungssteuerung mit kürzesten Zykluszeiten, höchster Genauigkeit und hoher Handhabungskapazität. Die PickMaster™-Software wurde zur einfachen Bedienung entwickelt und erleichtert die Integration des Roboters.

Der IRB 360 FlexPicker ist in verschiedenen Ausführungen mit Handhabungskapazitäten zwischen 1 kg und 8 kg verfügbar. Jede Ausführungsvariante ist für schnelle Pick- & Place-Anwendungen sowie für Pack-Aufgaben optimiert.

In der Edelstahl-Ausführung mit Schutzart IP69K ist der IRB 360 besonders für den Einsatz in der Lebensmittelindustrie geeignet. Er lässt sich mit industriellen Reinigungsmitteln und Hochdruck-Heißwasser reinigen. Für einen Korrosionsschutz verfügt der IRB 360 außerdem über glatte, leicht zu reinigende Oberflächen und schmiermittelfreie Gelenke.

Die PickMaster-Software für das Einrichten einer Applikation hat sich inzwischen zu einer wertvollen Unterstützung für Integratoren und Anwender des IRB 360 entwickelt. Sie vereinfacht die Konfiguration der Bildverarbeitung und bietet die erforderlichen Werkzeuge für effiziente Pick-&-Place-Anwendungen im Hochgeschwindigkeitsbereich.

Die zuverlässige, seit vielen Jahren bewährte Steuerung IRC5 ist ebenfalls integraler Bestandteil der FlexPicker-Roboterlösung. Mit den Funktionen TrueMove™ und QuickMove™ stellt die IRC5 höchstmögliche Bahngeschwindigkeit und Bahngenauigkeit sicher und ermöglicht damit dem Roboter, schnell laufenden Förderbändern mit extremer Genauigkeit synchron zu folgen. Die IRC5 ist auch in einer platzsparenden Einbauvariante zur einfachen Integration in vorhandene Anlagen und Produktionslinien erhältlich.

Eigenschaften und Vorteile

- höchste Geschwindigkeit und Flexibilität
- leistungsstark mit bis zu 8 kg Handhabungskapazität
- hygienisches Design für Anwendungen im Nassbereich
- überragende Präzision beim Conveyor Tracking
- integrierte Software zur Bildverarbeitung
- integrierte Steuerung von getakteten Förderern

Spezifikation			
Roboterversion	**Handhabungs-kapazität**	**Durchmesser Arbeitsbereich**	**Anzahl der Achsen**
IRB 360-1/800	1 kg	800 mm	4
IRB 360-1/1130	1 kg	1130 mm	3/4
IRB 360-3/1130	3 kg	1130 mm	3/4
IRB 360-1/1600	1 kg	1600 mm	4
IRB 360-6/1600	6 kg	1600 mm	4
IRB 360-8/1130	8 kg	1130 mm	4

Zusatzlast:	Am Ober- und Unterarm können Zusatzlasten von jeweils 350 g montiert werden.
Schutzart / Ausführung:*	IP54 / Standard, IP54 / Reinraum**, IP54 / Reinraum, Edelstahl**, IP67 / Nassreinigung, IP69K / Edelstahl, Nassreinigung
Montageart:	hängend
Integrierte Anwenderschnittstelle:	12 Signale (50 V, 250 mA)
Integrierte Vakuumversorgung:	max. 7 bar / max. Vakuum 0,75 bar

* alle Varianten werden standardmäßig mit lebensmittelverträglicher Schmierung (NSF H1) ausgeliefert
** IPA-zertifiziert, Reinraumklasse 5

Leistung	
Positionswiederholgenauigkeit:	0,1 mm
Winkelwiederholgenauigkeit:	
Standard / Edelstahl, 4 Achsen	0,4°
Nassreinigung, 4 Achsen	1,5°

Zykluszeit***					
25/305/25 (mm)	**0,1 kg**	**1 kg**	**3 kg**	**6 kg**	**8 kg**
IRB 360-1/1130	0,30	0,36			
IRB 360-3/1130	0,40	0,40	0,52		
IRB 360-8/1130		0,38	0,42		0,60
IRB 360-1/1600	0,35	0,40			
IRB 360-6/1600		0,43	0,48	0,60	
90/400/90 (mm)	**0,1 kg**	**1 kg**	**3 kg**	**6 kg**	**8 kg**
IRB 360-1/1130	0,44	0,51			
IRB 360-3/1130	0,60	0,60	0,75		
IRB 360-8/1130		0,55	0,65		0,92
IRB 360-1/1600	0,50	0,54			
IRB 360-6/1600		0,57	0,63	0,80	

*** Typische Zykluszeiten für Standard-Varianten.
Die Zykluszeiten für den IRB 360-1/800 sind unterschiedlich.

Conveyor Tracking****	
Geschwindigkeit, konstant [mm/s]	**Wiederholgenauigkeit [mm]**
200	1,0
350–750	1,5
800–1400	5,0
Start/Stop Conveyor [mm/s]	**Wiederholgenauigkeit [mm]**
500 (Start/Stopp in 0,2 Sek.)	3,5
Indexing Conveyor Control	**Wiederholgenauigkeit [mm]**
3,5 g Beschleunigung/Abbremsung	2

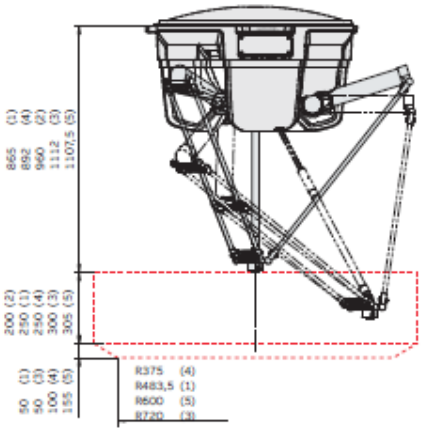
**** Die Leistungsdaten zum Conveyor Tracking wurden unter Produktionsbedingungen mit dem IRB 360-1/1130 und der Software PickMaster™ ermittelt. Die Werte können in Relation zu den Anforderungen der aktuellen Applikation, abhängig von der Geschwindigkeit und Beschleunigung des Roboters, variieren.

Elektrische Anschlüsse	
Netzspannung:	200–600 V, 60 Hz
Leistungsaufnahme bei max Last: (Typischer Pick- & Place-Zyklus mit 1 kg Nutzlast)	0,477 kW

Gewicht	
Gewicht:	120 kg 145 kg (nur Edelstahl-Ausführung)

Betriebsbedingungen	
Umgebungstemperatur:	±0 °C bis +45 °C
Relative Luftfeuchtigkeit:	max. 95 %
Geräuschpegel:	< 70 dB (A)
Emission:	EMC/EMI-abgeschirmt

Arbeitsbereich



(1) IRB 360-1/1130 und IRB 360-3/1130
(2) IRB 360-1/800
(3) IRB 360-1/1600
(4) IRB 360-8/1130
(5) IRB 360-6/1600

ABB Automation GmbH
Unternehmensbereich Robotics
Grüner Weg 6
D-61169 Friedberg
Phone: +49 60 31 85-0
Fax: +49 60 31 85-297
E-Mail: robotics@de.abb.com

www.abb.de/robotics

RO/80082 DE_H Mai 2017

Anhang 8: Datenblatt - Delta-Roboter IRB 360.

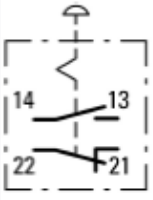
DATENBLATT - M22-PV/KC11/IY

Gehäuse, NOT-HALT-/NOT-AUS-Tasten, Pilzform, 38 mm, unbeleuchtet, Zugentriegelung, 1 Ö, 1 S, Schraubanschluss, rot, gelb

Powering Business Worldwide

Typ	**M22-PV/KC11/IY**
Katalog Nr.	**216525**
Alternate Catalog No.	**M22-PV-KC11-IYQ**

Lieferprogramm

Sortiment			RMQ-Titan
Grundfunktion			Gehäuse NOT-HALT-/NOT-AUS-Tasten
Einzelgerät/Komplettgerät			Komplettgerät
Bauform			Pilzform
Durchmesser	∅	mm	38
Beleuchtung			unbeleuchtet
Prüfzeichen			DGUV Test ET 16107 Sicherheit geprüft tested safety
			Zugentriegelung
Anschlussart			Schraubanschluss
Beschreibung			überlistungssicher nach ISO 13850/EN 418
Farbe			
Pilzstößel			rot
			
Gehäusedeckel			gelb
Schutzart			IP66, IP69
Anbindung an SmartWire-DT			nein
Kontaktbestückung			
Ö = Öffner			1 Ö
S = Schließer			1 S
Hinweis			= Sicherheitsfunktion, durch Zwangsöffnung nach IEC/EN 60947-5-1
Weg des Bedienteils und Betätigungskraft nach DIN EN 60947-5-1, K.5.4.1			
Zwangsöffnungsweg	mm		4.8
maximaler Weg	mm		5.7
Mindestkraft für Zwangsöffnung	N		20
Schaltzeichen			

Anhang 9: Datenblatt - Not-Aus-Taste.

Vakuumsauger
VASB-15-1/8-PUR-B
Teilenummer: 1395671

FESTO

Datenblatt

Merkmal	Werte
Nennweite	3 mm
Sauger-Durchmesser	15 mm
Sauger-Volumen	0,83 cm3
Wirksamer Saugdurchmesser	12,4 mm
Anschlusslage	oben
Einbaulage	beliebig
Saugerform	rund Faltenbalg 1,5-fach
Betriebsdruck	-0,95 ... 0 bar
Nennbetriebsdruck	-0,7 bar
Betriebsmedium	atmosphärische Luft in Anlehnung an ISO 8573-1:2010 [7:-:-]
Korrosionsbeständigkeitsklasse KBK	2
Lebensmittelunbedenklichkeit	gemäß Herstellererklärung
Umgebungstemperatur	-20 ... 60 °C
Haltekraft bei Nennbetriebsdruck	8,5 N
Produktgewicht	3 g
Befestigungsart	über Vakuumanschluss
Vakuumanschluss	G1/8
Farbe	blau
Shore-Härte	60 +/- 5
Werkstoffinformation Einschraubzapfen	Messing
Werkstoffhinweis	RoHS konform
Werkstoffinformation Sauger	PUR

Anhang 10: Datenblatt - Vakuumsauger.

3 Port Solenoid Valve
Direct Operated Poppet Type

New

CE

RoHS

Power consumption

4 W Standard type
(Existing product: 4.8 W)

1.8 W Energy-saving type
(Existing product: 2 W)

Vacuum applications

–101.2 kPa

A single valve with various valve functions

(Universal porting type)

N.C. valve	N.O. valve
Divider valve	Selector valve

etc.

Low concentration ozone resistant
Rubber seal material: HNBR for main valve

Mounting dimensions are **interchangeable** with existing product

Body ported type

Manifold type

Series VT307

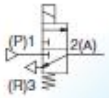
SMC

CAT.EUS11-107A-UK

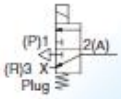
■ A variety of valve options

■ Application examples

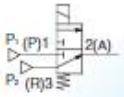
❶ Blow-off valve



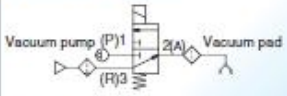
❷ Pressure release valve



❸ Selector valve

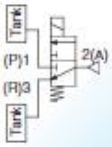


❹ Valve for vacuum

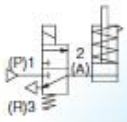


Vacuum releasing air
Atmospheric pressure or micro pressure

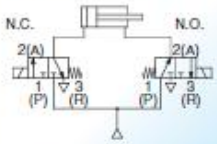
❺ Divider valve



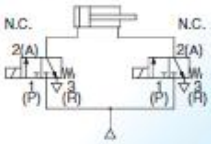
❻ Single-acting cylinder drive



❼ Double-acting cylinder drive



❽ Double-acting cylinder drive (Exhaust centre)



3 Port Solenoid Valve, Universal Porting Type Variations

Poppet type	Direct operated poppet type			Pilot poppet type
Series	New VT307	VT317	VT325	VP300/500/700
Cv (P⟷A)	0.19	0.62	1.4	0.8 to 3.6

⚠Caution

Make sure that dust and/or other foreign materials do not enter the valve from the unused port (e.g. exhaust port).

Standard Specifications

Type of actuation	Direct operated type 2 position single solenoid
Fluid	Air
Operating pressure range	0 to 1 MPa (High-pressure type), 0 to 0.7 MPa (Standard type)
Ambient and fluid temperature	−10 to 50°C (No freezing)
Response time Note 1)	20 ms or less (at 0.5 MPa)
Max. operating frequency	10 Hz
Lubrication	Not required (Use turbine oil Class 1 ISO VG32, if lubricated.)
Manual override	Non-locking push type
Mounting orientation	Unrestricted
Impact/Vibration resistance Note 2)	150/50 m/s²
Enclosure	Dustproof

Electrical entry			DIN terminal
Coil rated voltage [V]	**AC (50/60 Hz)**		24, 48, 100, 110, 200, 220, 240
	DC		6, 12, 24, 32, 48, 100
Allowable voltage fluctuation			−15 to +10% of rated voltage
Apparent power Note 3) Note 4)	**AC**	**Inrush**	12.7 VA (50 Hz), 10.7 VA (60 Hz)
		Holding	7.6 VA (50 Hz), 5.4 VA (60 Hz)
Power consumption Note 3) Note 4)	**DC**		Without indicator light: 4 W, With indicator light: 4.2 W
Light/Surge voltage suppressor	**AC**		Varistor, LED
	DC		Diode, LED

Note 1) Based on dynamic performance test, JIS B 8374-1981. (Coil temperature: 20°C, at rated voltage, without surge voltage suppressor)

Note 2) Impact resistance: No malfunction occurred when it is tested with a drop tester in the axial direction and at the right angles to the main valve and armature in both energized and de-energized states every once for each condition. (Values at the initial period)

Vibration resistance: No malfunction occurred in a one-sweep test between 45 and 1000 Hz. Test was performed at both energized and de-energized states in the axial direction and at the right angles to the main valve and armature. (Values at the initial period)

Note 3) At rated voltage

Note 4) The value is different for continuous duty type (VT307E), and energy-saving type (VT307Y/W). Refer to "Valve Options" shown below.

Flow-rate Characteristics

Valve model	Port size	Flow-rate characteristics															
		1 → 2 (P → A)				2 → 3 (A → R)				3 → 2 (R → A)				2 → 1 (A → P)			
		C [dm³/(s·bar)]	b	Cv	Q [L/min] (ANR) Note 2)	C [dm³/(s·bar)]	b	Cv	Q [L/min] (ANR) Note 2)	C [dm³/(s·bar)]	b	Cv	Q [L/min] (ANR) Note 2)	C [dm³/(s·bar)]	b	Cv	Q [L/min] (ANR) Note 2)
VT307	1/8	0.71	0.35	0.18	187	0.68	0.27	0.17	170	0.65	0.36	0.17	172	0.63	0.35	0.17	166
VT307V (Vacuum spec. type)																	
VT307E (Continuous duty type)		0.41	0.26	0.10	102	0.44	0.35	0.11	116	0.48	0.27	0.12	120	0.35	0.33	0.10	91
VT307Y (Energy-saving type)																	
VT307W (Energy-saving, Vacuum spec. type)																	
VT307	1/4	0.71	0.31	0.19	182	0.71	0.25	0.17	175	0.68	0.33	0.17	176	0.71	0.26	0.18	176
VT307V (Vacuum spec. type)																	
VT307E (Continuous duty type)		0.49	0.20	0.12	117	0.44	0.34	0.11	115	0.48	0.17	0.12	113	0.46	0.28	0.11	116
VT307Y (Energy-saving type)																	
VT307W (Energy-saving, Vacuum spec. type)																	

Note 1) Values for a single valve unit. It is not applicable to the manifold. Refer to the manifold specifications on page 5.

Note 2) These valves have been calculated according to ISO6358 and indicate the flow rate under standard conditions with an inlet pressure of 0.6 MPa (relative pressure) and a pressure drop of 0.1 MPa.

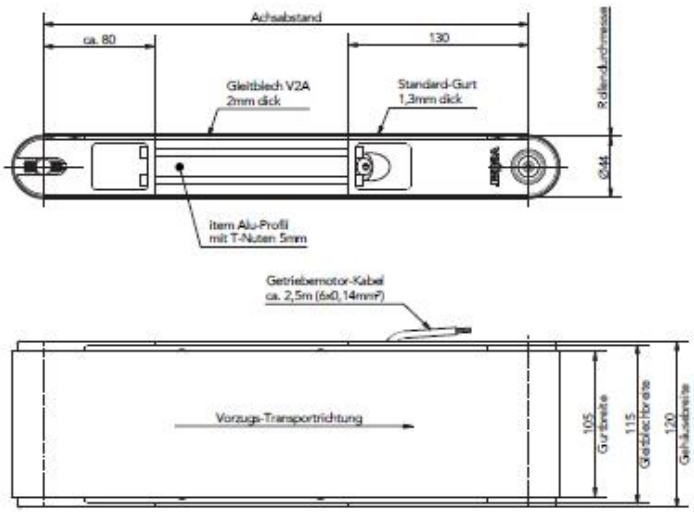
Anhang 11: Datenblatt - Elektromagnetventil.

Innenliegender Antrieb

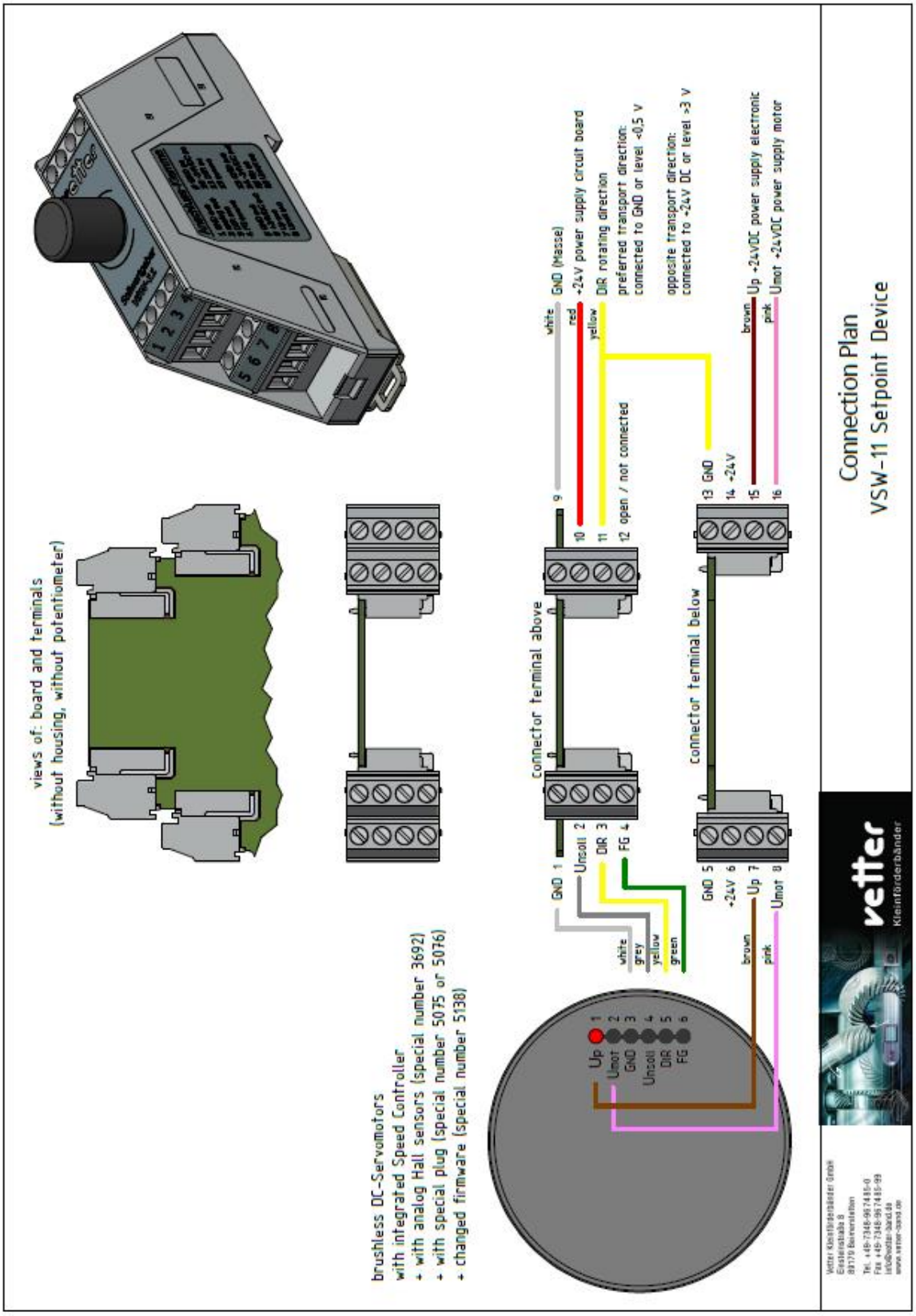
FR-40-120

Gehäusehöhe	40 mm
Gehäusebreite	120 mm
Gurtbreite	105 mm

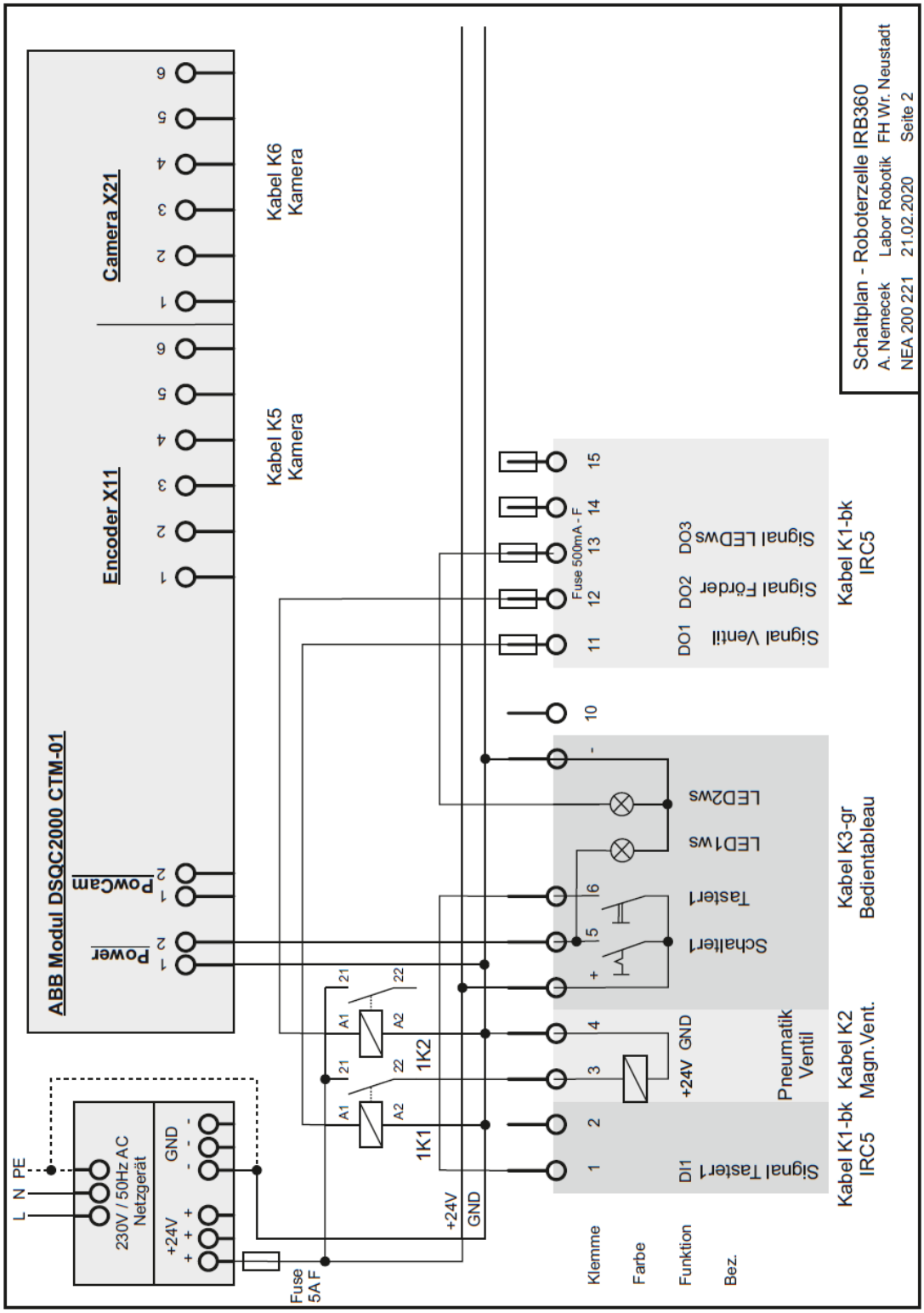
Achsabstände	in mm (kurzfristig lieferbar): 350, 400, 450, 500, 600, 700, 800, 900, 1.000, 1.200, 1.400, 1.600 Sonder-Achsabstände auf Anfrage
Geschwindigkeit / Last	2 - 10 m/min bei max. 50 N oder 0,5 - 3 m/min bei max. 100 N Geschwindigkeit über Analogsignal 0-10 V regelbar: GND oder 0 - 0,3 V: Festdrehzahl programmiert 0,5 - 0,9 V: Motor Stopp 1 - 10 V: Geschwindigkeit regelbar von min. bis max. bis 11 V: Drehzahlsteigerung bis zu 110 % der Nenngeschwindigkeit mit einem Vetter Sollwertgeber
Transportgurt	grün, FDA, antistatisch (weitere auf Anfrage)
Erweiterungen	- Gleitunterlagen (z.B. U-Rinnen) - Messerkante (anderer Gurt) - Wellenstummel - Encoderzapfen - Förderbänder gekoppelt (zweireihig) - Sollwertgeber VR-24-SW-11 - Sollwertgeber VSW-11
Versorgungsspannung	24 V DC



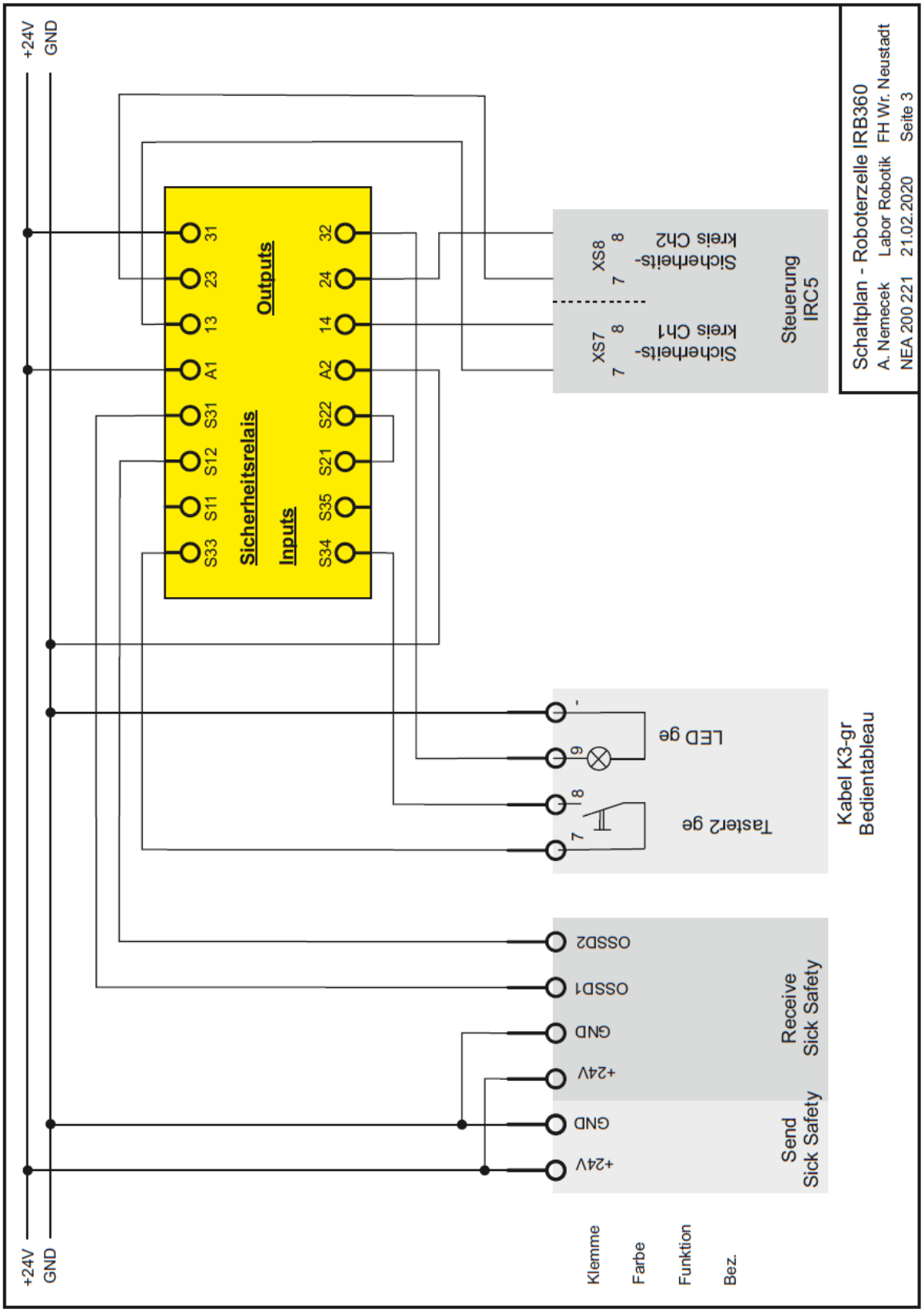
Anhang 12: Datenblatt - Vetter Förderband Typ FR-40-120.



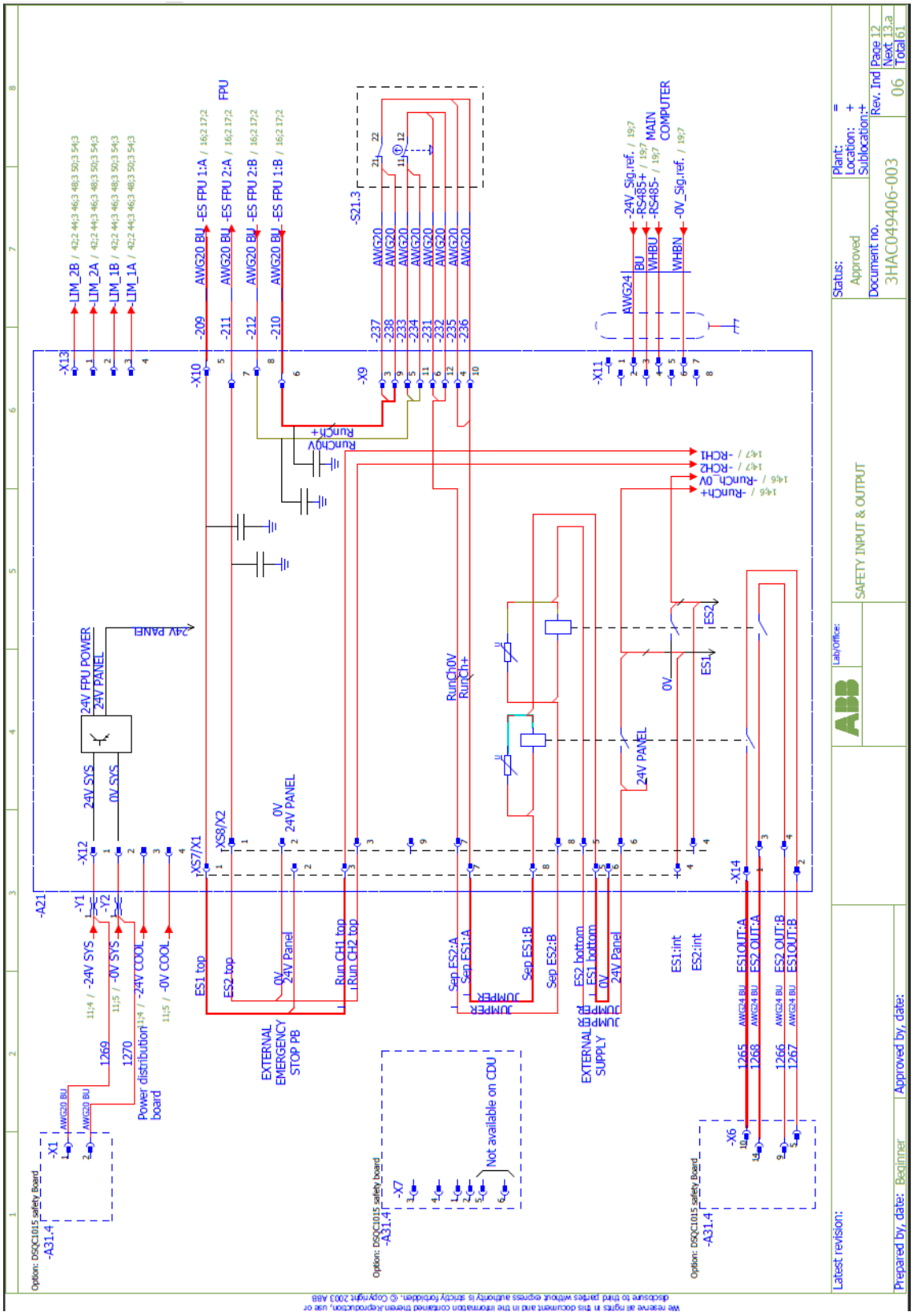
Anhang 13: Anschlussplan - Potentiometer.



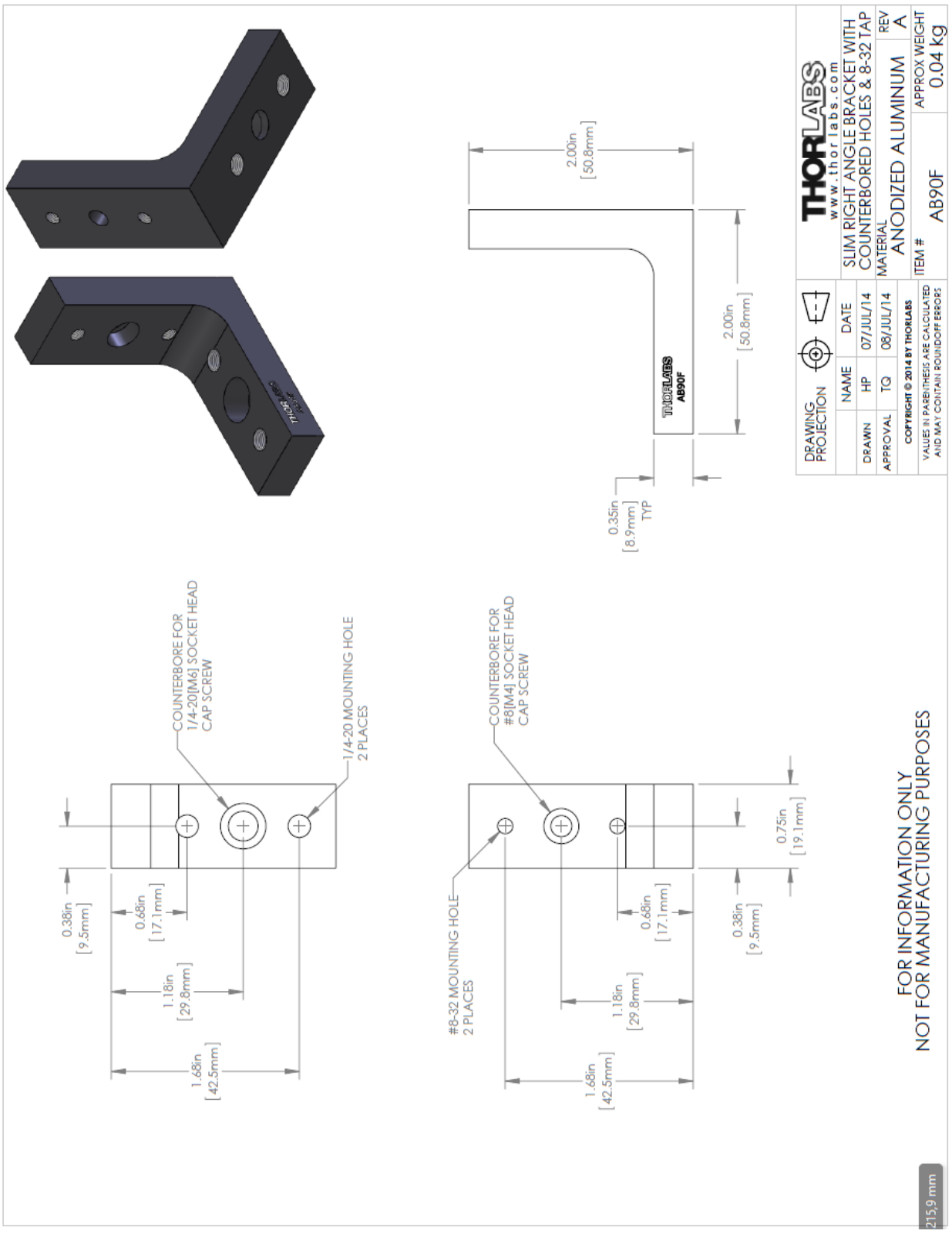
Anhang 14: Anschlussplan - Schaltschrank.



Anhang 15: Anschlussplan - Sicherheitskreis.



Anhang 16: Schaltplan - Sicherheitskreis der IRC5 Steuerung.



Anhang 17: Technische Zeichnung - Winkel.