Masterplan Digitalisierung

Mensch-Roboter-Kollaboration – Robonatives:
Kompetenzzentrum Robotik

Landesinitiative n-21 • www.n-21.de

Landesinitiative n-21:
Schulen in Niedersachsen online

www.n-21.de

Roboter im Unterricht

Mensch-Roboter-Kollaboration im schulischen Einsatz

Kompendium des Kompetenzzentrums Robotik an den Partnerhochschulen:

Leibniz Universität Hannover

HOCHSCHULE OSNABRÜCK
UNIVERSITY OF APPLIED SCIENCES

JADE HOCHSCHULE
Wilhelmshaven Oldenburg Elsfleth

Carl von Ossietzky Universität Oldenburg

Herausgeber:
Landesinitiative n-21: Schulen in Niedersachsen online e. V.
Schiffgraben 27
30159 Hannover
Deutschland

www.n-21.de

Stand: Juni 2023

Layout und Satz: Fabian Icken, Hochschule Osnabrück
Druck: Meinders & Elstermann GmbH & Co. KG

Vorwort

Sehr geehrte Damen und Herren,

der kompetente, kritische, kooperative und gestalterische Umgang mit digitalen Technologien ist zu einer Grundvoraussetzung für die Teilhabe an der Gesellschaft und am Berufsleben geworden. Ein Beispiel für eine solche Technologie ist die Mensch-Roboter-Kollaboration (MRK), die in Handwerk und Industrie zunehmend eingesetzt wird, um die Stärken der beiden Akteure Mensch und Roboter zusammenzuführen. Kollaborative und sensitive Roboter erkennen mit Hilfe ihrer Sensoren einen gefährlichen Kontakt mit Menschen oder Gegenständen und kommen deshalb ohne Schutzzäune oder Lichtschranken aus.

Ziel des Projekts war es, allgemeinbildende und berufsbildende Schulen in Niedersachsen mit modernsten und industrienahen Robotern sowie der notwendigen digitalen Infrastruktur auszustatten, um Lernende angemessen auf die Arbeitswelt in einer Industrie 4.0 vorzubereiten, um eine MINT-Orientierung zu fördern und nicht zuletzt, um an der Robotik orientierte Unterrichtsansätze für den in Niedersachsen derzeit in der Einführung befindlichen Informatikunterricht oder Beiträge zur Einlösung der Kompetenzziele aus dem Strategiepapier der KMK „Bildung in der digitalen Welt“ zu liefern.

Das Projekt mündete in 4 Projektstränge:

- 55 Technologielabore an allgemeinbildenden Schulen
- 7 Innovations- und Zukunftszentren an berufsbildenden Schulen mit den Berufsbereichen Elektrotechnik oder Metalltechnik
- 4 Innovations- und Zukunftszentren an berufsbildenden Schulen mit dem Berufsbereich Gesundheit-Pflege
- Kompetenzzentrum Robotik an den Standorten Hannover, Oldenburg und Osnabrück

Seit August 2021 ist das Kompetenzzentrum Robotik dafür verantwortlich, bedarfsgerechte Qualifizierungen für die Projektlehrkräfte anzubieten, Lernszenarien zu dokumentieren und Unterrichtsmaterialien zu entwickeln. Die Standorte sind allesamt renommierte Institutionen, die in der Forschung und Lehre auf dem Gebiet der Robotik und Didaktik führend sind. Ihre Expertise und ihr Engagement spiegeln sich wider in der Qualität der Lehrerfortbildungen und dieses Kompendiums.

In den Beiträgen des Ihnen hiermit vorliegenden Kompendiums finden Sie neben theoretischen Grundlagen zur Robotik, Endeffektoren, Programmierung und Simulation auch Anleitungen zu verschiedenen Robotertypen. Abgerundet wird das Kompendium mit didaktischen Überlegungen zum Thema Robotik und daraus abgeleiteten Lernsituationen für allgemeinbildende und berufsbildende Schulen.

Das im niedersächsischen Masterplan Digitalisierung (2. Buch, Kap. 2.7) aufgeführte pädagogische Projekt „Mensch-Roboter-Kollaboration – Robonatives“ wurde im Auftrag des Niedersächsischen Kultusministeriums in Projektträgerschaft der Landesinitiative n-21: Schulen in Niedersachsen online e. V. durchgeführt.

Wir hoffen, dass Sie persönlich von den Ergebnissen dieses Projekts im schulischen Einsatz profitieren und wünschen Ihnen eine anregende Lektüre.

Mit herzlichen Grüßen

Michael Sternberg, OStD, Geschäftsführer
Landesinitiative n-21: Schulen in Niedersachsen online e. V.

Projektbeteiligte des Kompetenzzentrums Robotik

Am Standort Hannover

Prof. Dr. Johannes Krugel, Professur für Digitale Bildung und Didaktik der Informatik, Leibniz Universität Hannover
Dr. Torsten Lilge, Wissenschaftlicher Mitarbeiter, Leibniz Universität Hannover
Mavin Becker, Wissenschaftlicher Mitarbeiter, Leibniz Universität Hannover
Max Niklas Bartholdt, Wissenschaftlicher Mitarbeiter, Leibniz Universität Hannover
Levin Stanke, Wissenschaftlicher Mitarbeiter, Leibniz Universität Hannover
Florian Oel, Wissenschaftliche Hilfskraft, Leibniz Universität Hannover
Ina May, Geschäftsführerin, robospace gGmbH
Marvin Bersiner, CTO, robospace gGmbH
Vivien Dos Anjos, Studenten Coach, robospace gGmbH
Abdelaziz Tekaya, Studenten Coach, robospace gGmbH

Am Standort Osnabrück

Prof. Dr. Harald Strating, Professur für Didaktik der Technik, Hochschule Osnabrück
Prof. Dr. Dirk Rokossa, Professur für Handhabungstechnik und Robotik, Hochschule Osnabrück
Rene Egbers, Wissenschaftlicher Mitarbeiter, Hochschule Osnabrück
Fabian Icken, Wissenschaftlicher Mitarbeiter, Hochschule Osnabrück
Marcus Auf der Landwehr, Wissenschaftliche Hilfskraft, Hochschule Osnabrück
Jonas Albersmann, Wissenschaftliche Hilfskraft, Hochschule Osnabrück

Am Standort Oldenburg

Prof. Dr. Peter Röben, Professur für Technik und Technikdidaktik, Carl von Ossietzky Universität Oldenburg
Prof. Dr. Frank Wallhoff, Professur für Assistive Technologien, Jade Hochschule Studienort Oldenburg
Dr. Jan Landherr, Wissenschaftlicher Mitarbeiter, Carl von Ossietzky Universität Oldenburg
Dani Hamade, Wissenschaftlicher Mitarbeiter, Carl von Ossietzky Universität Oldenburg
Yves Korte-Wagner, Wissenschaftlicher Mitarbeiter, Jade Hochschule Studienort Oldenburg
Tobias Neiß-Theuerkauff, Wissenschaftlicher Mitarbeiter, Jade Hochschule Studienort Oldenburg

Inhaltsübersicht

1 Robotik als Unterrichtsthema (didaktisch - curricular)

Inhalt

1.1 Didaktik des Roboters und der Robotik an allgemeinbildenden Schulen

Peter Röben, Jan Landherr, Dani Hamade, Carl von Ossietzky Universität Oldenburg

Die Didaktik geht von der Priorität der Ziele aus: Was will man durch die Behandlung eines Gegenstands erreichen? Im Falle des Roboters haben wir es zunächst mit einem technischen Gegenstand zu tun, der an sich noch keinen Bildungswert besitzt. Damit dieser entwickelt werden kann, müssen die Kontexte, Wirkungen, Funktionsprinzipien des Roboters ermittelt werden. Der Industrieroboter werkelt seit über 60 Jahren in den Werkhallen vornehmlich der Automobilindustrie: Warum kam man lange ohne seine detaillierte Behandlung der Robotik in der *allgemeinbildenden* Schule aus und was macht sie nun zum so dringlichen Thema, dass die Landesregierung Millionen dafür aufbringt, sie nun zum Unterrichtsgegenstand zu machen?
In der *beruflichen* Schule sieht das etwas anders aus. Da Roboter ohne die Betreuung durch menschliche Fachkräfte nicht zu haben sind, gehört das Thema schon länger zum Kanon der beruflichen Schulen.

Im Folgenden wird der Schwerpunkt auf die allgemeinbildenden Schulen gelegt und es wird aus der Perspektive der allgemeinbildenden Technikdidaktik argumentiert. Dies kann aber gerne auch auf die beruflichen Schulen ausgedehnt werden.

1.1.1 Einleitung: Warum jetzt Roboter und Robotik in der Schule?

Unser Projekt führt den Begriff Robonatives im Titel und das verweist in Analogie auf digital Natives auf eine Generation, die mit Robotern aufgewachsen ist. Damit sind aber keineswegs die heute über 60-Jährigen gemeint (der erste Industrieroboter Unimate wurde 1961 in Betrieb genommen), sondern, sondern, Menschen, die Robotern ohne falsche Vorbehalte und Ängste gegenübertreten und mit ihnen zielgerichtet zusammenarbeiten können.

Die häufigste Form des mit Menschen kooperierenden Roboters ist aktuell noch der Industrieroboter, vor allem der sogenannte Cobot. Von ihm allein sind aber keine revolutionären Umbrüche zu erwarten, sondern weitere Schritte auf seiner inzwischen über sechzigjährigen Evolution. Aber sicherlich hat die Stiftung recht, wenn darauf hingewiesen wird, dass der Umkreis der Personen, die mit Robotern arbeitet, sich deutlich vergrößern hat. Arbeiten meint hier beides: sowohl Kooperation mit dem Roboter im Sinne des synchron und asynchron miteinander Arbeitens als auch Programmierung des Roboters, was bislang eher Sache von Spezialisten war, aber nun zur Arbeitsaufgabe von Nicht-Spezialisten wird. Insofern ist es auch völlig legitim, auf einen vorurteilslosen Umgang mit Robotern vorzubereiten.

Die eigentliche Revolution der Roboter wird aber erst noch kommen, allerdings zeichnet sie sich schon in ersten Umrissen ab: Roboter im Straßenbild und im Alltag. Ein Roboter wie z. B. Digit, der als Paketträger konstruiert ist und Pakete aus einem Lager oder von einem autonomen Auslieferungsfahrzeug bis zur Haustür bringen kann, wird inzwischen auf dem Markt angeboten[1]. Spot, ein Roboter der Firma Boston Dynamics, wurde z. B. von der Polizei in Nordrhein-Westfalen angeschafft und kam nach dem dramatischen Großbrand in Essen zum Einsatz[2]. Er wird auch für Routineinspektionen in Industrieanlagen (z. B. Ölplattformen[3]) eingesetzt, die er selbstständig erledigt[4] oder für die Überwachung des Raketenstartgeländes von Space X[5].

Die großen Fortschritte in der Erkennung der realen Umwelt, der Wahrnehmung von Menschen und ihren Gefühlen und der Erzeugung von Sprache mit Hilfe der KI führen fast im monatlichen Rhythmus zu neuen Meldungen über Fortschritte, von denen allerdings viele auch mit Vorsicht zu genießen sind.

1 https://www.theverge.com/2020/1/6/21050322/bipedal-robot-digit-agility-robotics-on-sale-delivery-inspection-ces-2020
2 https://www.feuerwehrmagazin.de/nachrichten/roboter-hund-erkundet-brandruine-113412
3 https://edison.media/digital/kuenstlicher-hund-ueberwacht-oelplattform/25200923/
4 https://www.energy-robotics.com/industries-page
5 https://www.youtube.com/watch?v=aajbFO7xwBM

1.1.2 Bildungsziele

Roboter ziehen Aufmerksamkeit auf sich. Sie sind für viele SuS sehr attraktiv und wenn man ihnen die Möglichkeit bietet, sie für eigene Projekte einzusetzen, werden sie begeistert genutzt. Auch ein Roboter wie z. B. der Dobot, zieht diese Aufmerksamkeit auf sich und dieses Interesse lässt sich für die Bildungsarbeit nutzen.

Um Szenarien für das Thema Roboter vorzubereiten, sollen im Folgenden einige Bildungsziele entwickelt werden. Sie sind nicht im Sinne einer Prioritätenliste zu lesen und auch nicht im Sinne eines vollständig abzuarbeitenden Programms. Vielmehr soll es darum gehen, die verschiedenen Ebenen unterrichtlichen Arbeitens kohärent zu vernetzen und auszurichten. Wie sie in unterrichtlichen Settings aufgegriffen und umgesetzt werden können, wird an konkreten Beispielen demonstriert.

Bildungsziel 1: Aufgeklärter Umgang mit Robotern

Humanoide Roboter, wie z. B. Amica[1], sind auf dem Vormarsch und werden in den nächsten Jahren das Bild des Roboters in der Öffentlichkeit nachhaltig prägen. Und deswegen ist ein zentrales Bildungsziel in Bezug auf Roboter die Unterscheidung zwischen Fake und Wirklichkeit. Gerade weil das Internet nur so überquillt von Fake-Meldungen auch in Bezug auf Roboter, ist es notwendig, dass SuS erlernen, wie man sich ein halbwegs verlässliches Bild über Roboter machen kann: Welche Quellen sind vertrauenswürdig? Wie kann man Videos prüfen? Ist das, was gezeigt wird, mit dem Stand der Technik vereinbar?

Bildungsziel 2: Unterscheidung zwischen Phantasie und Wirklichkeit

Die meisten Roboter, mit denen SuS in Berührung kommen, sind Phantasieprodukte in Filmen, Computerspielen und weiteren Medien. Das bedeutet aber nicht, dass sie für Bildungsaufgaben vernachlässigbar sind. Denn sie prägen Vorstellungen, Ängste und Erwartungen von SuS und ein guter Unterricht sollte dies aufnehmen. Vor diesem Hintergrund empfiehlt es sich, aktuelle Videos über Roboter (reale und phantastische) anzusehen und zu diskutieren. Das Ziel im Unterricht sollte sein, Strategien des Umgangs mit realen und phantastischen Robotern zu entwickeln.
Das Ziel des Unterrichts sollte u. a. sein, dass man auf die Anforderungen an die Robotertechnik zu sprechen kommt. Wie wird es technisch möglich gemacht, dass ein Roboter greifen und laufen kann, Aufgaben erledigt und unvorhergesehene Situationen meistert? Die beste Art, sich davon ein Bild zu machen ist es, selbst einen Roboter zu programmieren und ein Szenario seines Einsatzes zu entwickeln.

Bildungsziel 3: Den Roboter kennenlernen und durchschauen (der Roboter als Lerngegenstand und als Modell)

In der Anfangsphase werden SuS sich genauso wie die LuL mit dem Roboter als Lerngegenstand auseinandersetzen. Auf der einen Seite geht es darum, einen konkreten Roboter (z. B. den Dobot) in Betrieb zu nehmen und ihn dazu zu bringen, dass er das tut, was er tun soll. Dieser konkrete Roboter soll dabei als Modell für Roboter stehen, d. h. was man am konkreten Roboter lernt, lässt sich auch an Robotern der Praxis wiederfinden. Der Dobot z. B. kann dabei als Modell für Industrieroboter stehen, aber nur eingeschränkt für Serviceroboter. Die Frage, was an dem Dobot Modellcharakter hat und was identisch mit echten Robotern ist, ist dabei zu klären.

Der Roboter als Modell

Ein Roboter wie der Dobot hat alle Eigenschaften, die einen echten Industrieroboter auszeichnen. Nach Herstellerangaben wird eine nur geringfügig erweiterte Ausführung als echter Roboter in der Produktion verwendet. Das macht ihn zu einem guten Modell, an dem alle wichtigen Eigenschaften eines Industrieroboters studiert werden können. An ihm kann man vieles über Kinetik, Mechanik, Antrieb, Sensorik, Aktorik und Programmierung sowie Steuerung lernen. Die Übertragung des Gelernten auf echte Industrie-Roboter ist weitgehend möglich, der Dobot ist tatsächlich die „abgespeckte“ Version eines echten Industrieroboters.

1 https://www.golem.de/news/ameca-roboter-angeschaut-der-bislang-am-wenigsten-gruselige-roboter-2201-162256.html

Übertrag und Transfer auf reale Roboter

Die Vielfalt der realen Industrieroboter kann weitestgehend nur durch Medien in den Unterricht geholt werden, z. B. durch Videos und Fotos. Es bietet sich an, eine Exkursion z. B. in ein Unternehmen der Umgebung zu unternehmen, um sie in der Realität kennenzulernen. Durch den immer weiter ausgreifenden Einsatz von Robotern in der Industrie, findet man sie nicht nur in der Automobilindustrie, sondern z.B. Delta-Roboter in der Nahrungsmittelproduktion zum Abpacken von Keksen, Scara-Roboter in der pharmazeutischen Industrie oder auch Gelenkarmroboter in der Logistik in fast allen Industriebereichen.

Mit den medialen Repräsentationen kann ihr Einsatz in der Produktion gut anschaulich gemacht werden, im Internet gibt es dazu sehr viel Material. Aber um sich die Bedeutung der Roboter zu erschließen, müssen SuS sich intensiver mit ihrer Entwicklung beschäftigen, Vergleiche ziehen und Konsequenzen erörtern. Ein solcher Unterricht hätte ökonomische und politische Aspekte zu klären, denn die Automobilindustrie, zumal in Niedersachsen, ist ein immenser politischer Faktor. Um aber die Verwendung von Robotern in der Produktion nicht einfach nur anzusehen, sondern auch zu erschließen, kommt man nicht umhin, sich mit dem Thema Produktionstechnik zumindest in den Grundzügen zu beschäftigen.

Da der Unterricht unter Einbeziehung von Robotern in sehr verschiedenen Fächern stattfinden kann, braucht es ein flexibles Angebot an Materialen, die ggf. nach Anpassung an die eigenen Unterrichtsziele einsetzbar sind.

Beispiele für die verschiedenen möglichen Materialien:
1. Einführungslehrgang in die Inbetriebnahme und den Einsatz des Dobots (ggf. auch anderer Roboter)
2. Thematische Bausteine mit Informationen und Bezügen zum Thema, die in verschiedenen Unterrichten nutzbar sind:
    a. Grundlagen des Roboters (Antrieb, Mechanik, Kinetik, Sensorik, Aktorik, Steuerung und Regelung)
    b. Geschichte des Roboters und seines Einsatzes
    c. Überblick und Systematik von Robotern
    d. Roboter in Fantasie, Literatur, Film und Computerspielen
    e. Beispiele von aktuellen Robotern und solchen, die in den kommenden Jahren in die Alltagswelt vordringen werden (Serviceroboter, Pflegeroboter, Haushaltsroboter)
3. Beispielhafte Unterrichtseinheiten, in denen der Aufbau einer Unterrichtseinheit unter Verwendung von Lehrgängen und thematischen Bausteinen gezeigt wird.

Viele dieser Materialien sind in dem laufenden Projekt bereits realisiert worden.

Bildungsziel 4: Die Technik des Roboters für eigene Ziele einsetzen (der Roboter als Werkzeug)

Komplexe Technik tritt den Schülerinnen und Schülern oft als Black Box gegenüber, in der Informatik gibt es mit dem EVA-Prinzip eine darauf basierende Methode, mit der das komplizierte Innenleben von informationsverarbeitenden Systemen auf das Verhalten von Input zu Output reduziert werden kann. Gleichzeitig verlieren technische Systeme damit einen sinnlich anschaulichen Zugang für das Verstehen, weil Ein- und Ausgabe nur noch in einem abstrakten, rein logischen Verhältnis zueinanderstehen.

Durch das Programmieren des Dobots kann ein wenig Licht in die Blackbox gebracht werden. Wenn erst einmal der Zusammenhang zwischen Sensorik und Aktorik in einem eigenen Programm verwirklicht ist, werden auch mit den Fehlern, die der Dobot damit anfangs zwangsläufig macht, Einsichten in die Robotik und die Erfahrung vermittelt, dass die Robotertechnik kein Buch mit sieben Siegeln ist. Denn die Fehler kann man beheben und nach kurzer Zeit tut der Roboter das, was man von ihm will.

Die Beschäftigung mit Robotern, wie dem Dobot, fordert SuS in Gebieten heraus, die einen wichtigen Teil der technischen Bildung ausmachen: Seine technischen Abläufe können so durchschaut werden, dass eigene Aufgaben mit ihm bearbeitet werden können und er damit als Werkzeug für die eigenen Ziele eingesetzt werden kann. Wenn SuS lernen, den Dobot für ein eigenes Projekt zu programmieren, liefert dies wertvolle Einsichten in das Innere der Blackbox. In diesem handelnden Umgang mit einem Roboter werden Vorstellungen über Möglichkeiten und Grenzen eines Roboters und die Voraussetzungen ihrer Verwirklichung erworben, auf denen weiterer Unterricht zur Aufklärung von Mythen und Darstellung der Wirklichkeit von Robotern anknüpfen kann.

Bildungsziel 5: Wirkungen des Einsatzes von Robotern auf die Gesellschaft erkennen

Roboter entstehen nicht von selbst, sondern werden gemacht. Welche Ziele werden mit ihnen verfolgt? Die Roboter, die bislang die größte Wirkung auf die Gesellschaft hatten, sind Industrieroboter. Ohne sie wären z.B. Autos wesentlich teurer oder sehr viel einfacher. Sie hätten nicht die Qualität, die sie heute haben und die Ausstattungsvielfalt wäre erheblich geringer. In der Produktion von Autos fänden sich sehr viel mehr Arbeitsplätze, die von den Arbeiterinnen und Arbeitern einiges abverlangen und zu Gesundheitsbelastungen führen würden. In der Lackiererei z. B. muss kein Mensch mehr die belastete Luft atmen, da es dort während des Lackierens nur noch Roboter gibt. Die bisherigen Roboter sind klar den ökonomischen Zielen der Unternehmen zuzuordnen, die Roboter im großen Stil zur Rationalisierung der Produktion einsetzen.
Doch das Aufkommen der Serviceroboter verändert diese Situation. Statt in der Produktion findet man bald viele Roboter in Bereichen wie Logistik, Hotels und Restaurants, Museen, Pflegeeinrichtungen und auch im häuslichen Bereich finden. Vereinzelt ist dies jetzt schon der Fall. Sie werden mit den Menschen kommunizieren, teilweise wird das sogar ihre Hauptaufgabe sein, oder Dienstleistungen für ihn erledigen. Das wird viele ethische Fragen aufwerfen, von denen einige auch von großer Bildungsrelevanz für die Schule sein werden. In unseren technischen Seminaren an der Universität wurden lebhafte Diskussionen von den Studierenden darüber geführt, ob Service-Roboter in Pflegeeinrichtungen eingesetzt werden sollen, um Pflegekräfte einzusparen. So wurde beispielsweise die Vorstellung, dass alte Menschen von Robotern gefüttert werden sollen, als inhuman verurteilt.

Übergreifende Bildungsziele

Wie eingangs bemerkt, stellt sich aus der Perspektive der Didaktik auch immer die Frage: Welche Ziele lassen sich mit einem Thema erreichen? Ist es das Ziel, dass Schülerinnen und Schüler am Ende ihrer Schullaufbahn wissen, wie der Dobot funktioniert und wie man ihn programmiert? Es bietet sich an, eine Analogie zu ziehen, z. B. zur Didaktik der Biologie: Müssen Schülerinnen und Schüler notwendigerweise am Mikroskop ausgebildet werden? Die Antwort muss sicherlich „Nein“ lauten, wenn die Notwendigkeit auf den Wert für das spätere Leben bezogen ist, denn nur die wenigsten von ihnen werden später ihren Arbeitsalltag vor dem Mikroskop bestreiten. Doch das Mikroskopieren ist mehr, als die Bedienungselemente des Mikroskops kennenzulernen, seine Anwendung und das Präparieren der Objektträger. Der Blick durchs Objektiv erschließt eine Welt, die vorher nicht wahrgenommen wurde. Gerade, wenn die Proben von SuS selbst gezogen wurden, z. B. aus einem Tümpel in der Nähe der Schule und sie die Probe als Repräsentanten des Tümpels anerkennen, erfährt das Bild im Mikroskop eine Bedeutung, die entsprechende Abbildungen im Schulbuch nicht leisten können. Zusammen mit anspruchsvollen Aufgaben, wie z. B. dem Vergleich unterschiedlicher Proben und der Erkenntnis des Zusammenhangs zwischen naturnahen Tümpeln und einer großen Vielfalt im Mikroskop wird dieses zu einem Werkzeug der Erkenntnis für die SuS.

Auch für den Dobot lassen sich vergleichbare Bildungsziele festmachen. In einer Welt, die von Bits und Bytes beherrscht wird, kommt dem Verständnis über die grundlegenden Funktionsweise des Dobots eine wichtige Funktion für das Verständnis der realen Welt zu. Sein Einsatz als Werkzeug der SuS ermöglicht ihnen ein Verständnis für die reale Roboterwelt.

Der Dobot als Vertreter der Digitalisierung

Dass der Dobot in einem gesellschaftlichen Verhältnis steht, dürfte aus den vorangegangenen Ausführungen deutlich geworden sein. Neben einem aufgeklärt-kritischen Umgang mit der Thematik und dem Erlernen informationstechnologischer Grundlagen am Beispiel des Dobots steht dieser auch als Vertreter der Digitalisierung im Fokus des didaktischen Interesses. Immerhin findet das Projekt „Robonatives“ im Rahmen des Masterplans Digitalisierung statt. Das ist erklärungswürdig, denn für Schülerinnen und Schüler ist erst einmal nicht ersichtlich, was der Dobot mit dem zu tun hat, was sie vielleicht unter Digitalisierung verstehen: das Smartphone, soziale Netzwerke, KI und Smart Home. Der Aspekt der Digitalisierung hat in der Robotik insbesondere im Zusammenhang zur Industrie 4.0 und der vernetzten Produktion eine große Rolle inne.

Digitale Kompetenzen fördern

Digitale Kompetenzen umfassen dem DigComp, dem Europäischen Referenzrahmen für digitale Kompetenzen,[1] nach nicht nur jene Fertigkeiten und Kenntnisse, die man braucht, um z. B. eine Mail zu schreiben oder eine Suchmaschine zu nutzen, sondern u. a. Datenkompetenzen, Kompetenzen für das Gestalten und Erzeugen digitaler Inhalte sowie solche, die mit Sicherheitsaspekten und Problemlösen assoziiert sind. Die KMK hat dies in ihrem Strategiepapier „Bildung in der digitalen Welt“[2] in sechs Kompetenzfelder umgesetzt, von denen hier insbesondere Kompetenzfeld 5 „Problemlösen und Handeln“ sehr gut passt. Die Beschäftigung mit dem Dobot kann diesbezüglich wichtige Impulse geben, um in den verschiedenen Bereichen des Referenzrahmens zu einer Entwicklung bei den Schülerinnen und Schülern zu führen.

1 https://digcomp.enterra.de/europaeischer-referenzrahmen-digcomp.html
2 https://www.kmk.org/fileadmin/Dateien/pdf/PresseUndAktuelles/2018/Digitalstrategie_2017_mit_Weiterbildung.pdf

1.1.3 Robotik im Technikunterricht: Einordnung und curriculare Rahmenvorgaben

Dani Hamade, Carl von Ossietzky Universität Oldenburg

Die Thematisierung der Robotik im Unterricht stellt ein anspruchsvolles Unterfangen dar. Man bedenke alleine die Tatsache, dass die Robotik das gesamte Wissensgebiet der Roboter und ihrer Technik repräsentiert (vgl. Haun, 2013, S. 15). Anders als bei naturwissenschaftlichen Fragen, welche sich mit Kausalitäten beschäftigen, ist die Technik auf die finale Zweckursache ausgerichtet, was es bereits erschwert, einen theoretischen Rahmen als Grundlage für die Unterrichtsplanung auszumachen. In diesem Zusammenhang ist es insbesondere die Vielfalt an Robotern, sowie der anhaltende, technologische Fortschritt auf dem Gebiet der Robotik (vor allem auch in Verbindung mit der künstlichen Intelligenz), welche es zu erschweren scheinen, eine Eingrenzung und somit auch die Implementierung der Robotik unter Berücksichtigung verschiedener Roboterarten in den Unterricht vorzunehmen. Im Hinblick auf Industrieroboter ist mit dem Dobot hingegen ein System gegeben, welches überschaubar ist und sich angemessen in theoretische, aber auch diverse lebensweltbezogene Kontexte setzen lässt. Nicht zu vernachlässigen sind hierbei allerdings die Grenzen, die dem System gesetzt sind, wenn es darum geht, Wirklichkeitsbezüge zu Industrierobotern herzustellen. Es muss also herausgestellt werden, was den Modellcharakter des Dobots ausmacht.

Damit die Frage der Praxis, also des Unterrichts zur Robotik geklärt werden kann, muss man sich allerdings zunächst mit den übergeordneten Zielen auseinandersetzen, die man mit der Integration der Robotik in den Unterricht verfolgen sollte. Das übergeordnete Ziel sollte es, wie in den vorangestellten Abschnitten bereits hervorgehoben, sein, Schülerinnen und Schüler dazu zu befähigen, mit digitalen Systemen, worunter im Kontext des Projektes auch Roboter einzuordnen sind, selbstbestimmt umzugehen. Das Projektziel steht hierbei in Analogie zu der 2016 erstellten „Dagstuhl-Erklärung“, in welcher eine umfassende Betrachtungsweise solcher Systeme für den selbstbestimmten Umgang damit, vorausgesetzt wird (vgl. Gesellschaft für Informatik, 2016, S. 1).

„Bildung in der digitalen vernetzten Welt (kurz: Digitale Bildung) muss aus ***technologischer, gesellschaftlich-kultureller und anwendungsbezogener Perspektive*** *in den Blick genommen werden.“* (Gesellschaft für Informatik, 2016, S. 1).

Die dort erklärten Ziele weisen ebenfalls Synergien mit den fachdidaktischen Perspektiven im Hinblick auf den Technikunterricht auf. Nach dem mehrperspektivischen Verständnis des Technikbegriffs, darf die rein technologische Perspektive auf technische Artefakte und somit auch auf Roboter nicht allein im Mittelpunkt stehen und muss erweitert werden. Vielmehr wird ein anthropozentrisches Bild der Technik deutlich, in welcher Technik in enger Wechselwirkung mit dem Menschen steht. Technikunterricht umfasst demnach viele Perspektiven, was voraussetzt, dass unter anderem auch Ambivalenzen von Technik in den Vordergrund gerückt werden (vgl. Schmayl, Wilkening, 1995, S. 70 ff.). Die Zusammenführung mehrerer Wirkungsdimensionen im Kontext des Technikbegriffes geht demnach mit der Notwendigkeit einher, sowohl die technologische als auch die gesellschaftlich-kulturelle und anwendungsbezogene Perspektive auf Technik zum Gegenstand des Unterrichtes zu machen. Die Überführung der verschiedenen Wirkungsdimensionen von Technik in den Unterricht ist also unabdingbar, damit Schülerinnen und Schüler selbstbestimmt handeln können.

Was bedeutet das nun für die Thematisierung der Robotik? Die einzelnen übergeordneten Perspektiven wurden in den diesem Abschnitt vorangestellten Bezügen zu den Bildungszielen bereits weitestgehend herauskristallisiert. Im Folgenden sollen diese Aspekte in den Vordergrund gerückt und eine systematische Analyse curricularer Gegebenheiten vorgenommen werden, sodass eine Praxistauglichkeit im Kontext der unterrichtlichen Rahmenvorgaben bei gleichzeitiger Berücksichtigung einer mehrperspektivischen Herangehensweise hergestellt werden kann.

Eine Grundvoraussetzung für die Arbeit mit Robotern wie beispielsweise dem Dobot, ist der sichere Umgang mit den Robotik-Systemen. Wird der sichere (und pflegsame) Umgang mit den Systemen im Unterricht konkret vermittelt, so ist im Hinblick auf die Berührungsängste eine Hürde genommen und ein Schritt in Richtung des selbstbestimmten Umgangs geleistet. Dieser Gesichtspunkt ist in den curricularen Vorgaben mit dem ersten Handlungsbereich und dem dort aufgeführten Themenfeld „Sicheres Arbeiten mit Werkzeugen und Maschinen“ abgedeckt (vgl. Niedersächsisches Kultusministerium, 2012, S. 13 f.). Aus allen dort aufgeführten Kompetenzbereichen sind hierbei verschiedene Ziele auf die Arbeit mit Robotern adaptierbar. So sollten Schülerinnen und Schüler im Bereich des Fachwissens die Funktionsteile des Roboters benennen und beschreiben können (z.B. Endeffektoren oder Schrittmotoren), die Handhabung der Roboter beschreiben können (z. B. die Entriegelung der Motoren zur manuellen Führung des Roboterarmes) und schlussendlich auch Sicherheitsregeln benennen können (z.B. Quetschgefahr am Roboterarm) (vgl. ebd.). Im Kompetenzbereich der Erkenntnisgewinnung steht im Zusammenhang zur Arbeitssicherheit, dass Schülerinnen und Schüler Gefahrenpotenziale der Roboter erklären können (z.B. bei der Einbindung des 3D-Druck-Moduls beim Dobot). Gleiches gilt auch für den Kompetenzbereich der Beurteilung und Bewertung, in welchem angestrebt ist, dass sich Schülerinnen und Schüler mit den Sicherheitsregeln der Roboter auseinandersetzen (also keine reine Benennung). (vgl. ebd.). Weiterführend kann im Themenfeld „Planen, Konstruieren und Herstellen“ Bezug zum dort aufgeführten, planvollen Handeln genommen werden (vgl. ebd., S. 14). Besonders vor der Einrichtung aufwendigerer Produktionslinien, in welchen mehrere Roboter zum Einsatz kommen, ist es von großer Bedeutung, dass Schülerinnen und Schüler die Arbeitsabläufe und die dazugehörige Arbeitsorganisation planen.

Ein weiterer Aspekt, welcher in der Auseinandersetzung mit Robotern in den Vordergrund rückt und der technologischen Perspektive unterzuordnen ist, ist der der Antriebstechnik. Will man tiefer in die Technik von Robotern eindringen und sich mit den verschiedenen Antriebssystemen auseinandersetzen, die in der Robotik zum Einsatz kommen, so kann man sich auf den zweiten Handlungsbereich „Energie und Technik“ und das sich darin befindende Themenfeld „Antriebssysteme“ beziehen (vgl. Niedersächsisches Kultusministerium, 2012, S. 17). Auf eine Auflistung einzelner zu verfolgender Ziele aus den drei Kompetenzbereichen wird für diesen Bereich aus platzökonomischen Gründen allerdings verzichtet.

Der Handlungsbereich drei „Information und Kommunikation“ des Curriculums ist jener, welcher für die Auseinandersetzung mit der Robotik im Mittelpunkt steht. Insbesondere stechen hier die Themenfelder „Steuern und Regeln“, „Daten verarbeiten – digitale Schaltkreise“ und „Die Computer automatisieren technische Prozesse“ hervor (vgl. ebd., S. 20). Hier tritt auch die gesellschaftlich-kulturelle Perspektive im Hinblick auf die Robotik in den Vordergrund, nämlich dann, wenn Schülerinnen und Schüler die Auswirkungen der Digitalisierung und computergesteuerter Prozesse auf die Arbeits- und Lebenswelt bewerten und beurteilen (vgl. ebd., S. 21 f.). An dieser Stelle ist auch an die Berufsorientierung als Querschnittsaufgabe aller Fächer zu erinnern. So sollte bei der Diskussion der Auswirkungen der Robotik im Zusammenhang zur Arbeitswelt nicht auf den Bezug zu den Berufen verzichtet werden, bei denen die Robotik eine Rolle spielt und es sollten Möglichkeiten der Aus- und Weiterbildung auf dem Gebiet aufgeführt werden (siehe hierzu zum Beispiel Abschnitt Robotik im berufsbildenden Bereich).

Auch die anwendungsbezogene Perspektive tritt in diesen Handlungsbereichen auf, beispielsweise dann, wenn Schülerinnen und Schüler beschreiben, wie eine Serienfertigung computergestützt eingerichtet werden kann (vgl. ebd., S. 22). Es ließen sich aus diesem Handlungsbereich noch unzählige Kompetenzen im Hinblick auf die technologische Perspektive aufführen, die im direkten Zusammenhang zur Robotik stehen (z. B. die Entwicklung von Programmen zur Steuerung und Regelung oder die Erklärung der Wechselwirkungen zwischen Sensorik und Aktorik), welche hier allerdings nicht alle im Einzelnen aufgeführt werden. Vielmehr sollte mit dieser Analyse der curricularen Vorgaben ein Einblick in mögliche Anknüpfungspunkte für den Technikunterricht unter Berücksichtigung der verschiedenen Wirkungsdimensionen gegeben werden. Zuletzt ist noch anzuführen, dass Roboterarme im Curriculum auch konkret für das Themenfeld „Die Computer automatisieren technische Prozesse“ genannt werden (vgl. ebd., S. 34). Wie sich allerdings gezeigt hat, ist die Robotik an weitaus mehrere Handlungsfelder anknüpfungsfähig. Denkt man alleine an die Gestaltung vieler Roboter, so ließe sich zum Beispiel auch der Handlungsbereich vier mit dem Themenfeld der Bionik adaptieren, da viele Roboter die belebte Natur zum Vorbild haben.

Literatur

Haun, M. (2013): *Handbuch Robotik. Programmieren und Einsatz intelligenter Roboter*. *2. Auflage*. Berlin Heidelberg: Springer Verlag.

Gesellschaft für Informatik (2016): *Bildung in der digital vernetzten Welt. Eine gemeinsame Erklärung der Teilnehmerinnen und Teilnehmer des Seminars auf Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH*. Online abgerufen unter: https://gi.de/fileadmin/GI/Hauptseite/Themen/Dagstuhl-Erkla__rung_2016-03-23.pdf (zuletzt geprüft am: 24.04.2023).

Niedersächsisches Kultusministerium (2012): *Kerncurriculum für die Oberschule Technik*. Online abgerufen unter: https://cuvo.nibis.de/index.php?p=download&upload=25 (zuletzt geprüft am: 24.04.2023).

Schmayl, W., Wilkening, F. (1995): *Technikunterricht. 2. überarbeitete Auflage*. Bad Heilbrunn: Verlag Julius Klinkhardt. Online abgerufen unter: https://dgtb.de/wp-content/uploads/2020/04/Schmayl-Wilkening-Technikunterricht.pdf (zuletzt geprüft am: 24.04.2023).

1.1.4 Anknüpfungspunkte für den Informatikunterricht in Sek. I laut Kerncurriculum

Jan Landherr, Carl von Ossietzky Universität Oldenburg

Janzen et al. (2015) haben in einer Studie im Regierungsbezirk Münster untersucht, weshalb Schülerinnen und Schüler das Fach Informatik nicht wählen und sind hierbei darauf gestoßen, dass andere Fächer attraktiver zu sein scheinen und dass Desinteresse an Computer dazu führt, dass das Fach nicht belegt wird. Peters et al. (2018) vermuten, dass „[...] die curricularen Fragestellungen der Informatik für die Schülerinnen und Schüler nicht ansprechend [sind]". Gleichzeitig legen Studien nahe, dass die Thematisierung von Robotern im Informatikunterricht eine motivierende Wirkung auf die Lernenden haben kann (Peters et al. nennen hier Samuelsen et al. (2009), Çankaya et al. (2017) und Chetty (2015)) und dass das Fach dadurch auch für Schülerinnen und Schüler, die das Programmieren scheuen, attraktiver gestaltet werden könnte. Damit also die Brücke zwischen curricularen Fragestellungen und motivierendem, attraktiv gestaltetem Unterricht durch den Einsatz von Robotern geschlagen werden kann, wird im Folgenden Abschnitt eine curriculare Einordnung der Robotik vorgenommen.

Für den Sekundarbereich I werden im Kerncurriculum für das Fach Informatik in vier Lernfelder die inhalts- und prozessbezogenen Kompetenzen zusammengeführt: *Daten und ihre Spuren, Computerkompetenz, Algorithmisches Problemlösen* und *Automatisierte Prozesse*. Im Folgenden werden diejenigen Kompetenzen zitiert, die sich als anknüpfungsfähig erweisen, um das Thema Robotik im Unterricht zu behandeln, da sich nur in zwei Kompetenzformulierungen ein direkter Hinweis auf das Thema Roboter/Robotik finden lässt.

Der Begriff des Algorithmus ist wesentlicher Bestandteil der Informatik, seine Vermittlung daher ein zentrales Ziel des Informatikunterrichts an allgemein bildenden Schulen. Ausgehend von dieser Leitidee haben Wiesner und Brinda in einer Fallstudie untersucht, inwiefern sich algorithmische Grundstrukturen mithilfe robotischer Systeme im Unterricht vermitteln lassen (Vgl. Wiesner u. Brinda 2007). Besondere Schwierigkeit dabei sei die zeitintensive Einarbeitung, die die Lehrkraft zu tätigen habe, bevor die Schülerinnen und Schüler, z. B. mit komplexen Programmiersprachen einfache algorithmische Abläufe entwerfen und codieren können. Das Ausweichen auf leichte oder grafische Programmiersprachen scheint hierbei eine Möglichkeit zu sein, den Abstraktionsgrad zu verringern. Ebenso scheint es von Vorteil zu sein, einen Praxisbezug und handlungsorientierten Unterricht als Ausgangspunkt zu wählen und reale technische Systeme in den Mittelpunkt zu stellen (Vgl. ebd. 114). Im Lernfeld *Algorithmisches Problemlösen* eignen sich daher folgende Kompetenzaussagen als Bezugspunkte für das Projekt Robonatives:

Die Schülerinnen und Schüler...

- *stellen einen gegebenen Algorithmus in einem Struktogramm dar. (Vertiefung)*
- *interpretieren ein vorgegebenes Struktogramm (Vertiefung)*
- *benennen Anweisung, Sequenz, Schleife und Verzweigung als elementare Kontrollstrukturen (Basis)*
- *entwickeln und implementieren einen Algorithmus in einer grafischen Programmiersprache auf experimentelle Weise (Basis)*
- *entwerfen einen Algorithmus unter zielgerichteter Verwendung der elementaren Kontrollstrukturen (Vertiefung)*

Der Fokus dieses Lernfeldes liegt klar auf der Analyse eines vorgegebenen Problems und dessen Lösung durch einen geeigneten Handlungsablauf. Die Schülerinnen und Schüler sollen hierbei das algorithmische Problemlösen und dessen elementare Prinzipien kennen und anwenden lernen. Dieses Lernfeld abstrahiert noch stark von der konkreten Umsetzung, zum Beispiel durch das Programmieren eines Bewegungsablaufs eines Roboters, sodass hier vielfältige Implementierungsmöglichkeiten denkbar sind.

Für das Projekt Robonatives ist die Verknüpfung des dritten mit dem vierten Lernfeld von Bedeutung, da es hier um *Automatisierte Prozesse* geht. Auf der einen Seite kann das Thema Robotik hier als exemplarischer Vertreter einer Fertigungstechnik behandelt werden:

Die Schülerinnen und Schüler…
- *erläutern Möglichkeiten der Anwendung von robotergestützten Systemen (Basis)*
- *benennen Typen von Sensoren, Aktoren und Verarbeitungskomponenten von technischen Geräten und ordnen sie der Eingabe, Verarbeitung und Ausgabe zu (Basis)*
- *lesen Sensoren aus und steuern Aktoren an (Basis)*
- *beschreiben die einzelnen Schritte beim Ablauf eines automatisierten Prozesses (Vertiefung)*
- *nennen gesellschaftliche Konsequenzen des Einsatzes automatisierter Prozesse, z.B. in der industriellen Produktion (Vertiefung)*

Auf der anderen Seite können die erarbeiteten algorithmischen Lösungsentwürfe in die Praxis überführt werden.

Die Schülerinnen und Schüler…
- *entwickeln einen Algorithmus zur Steuerung eines einfachen Informatiksystems (Vertiefung)*

Keine Berücksichtigung finden Kompetenzaussagen, die die technische Seite robotischer Systeme thematisieren, z. B. Aufbau und Funktionsweise von Aktoren, Sensoren und mechanischen Komponenten. Zudem müssen mehrperspektivische Betrachtungsweisen, etwa zur Rolle von Robotik im industriellen Produktionsprozess oder zu verschiedenen Einsatzgebieten in Anwendungskontexten, ohne Verortung in den Lernfeldern auskommen.

Literatur

Çankaya, S.; Durak, G.; Yünkül, E. (2017): Education on Programming with Robots: Examining Students' Experiences and Views. In: Turkish Online Journal of Qualitative Inquiry 10/2017.

Chetty, J. (2015): The notion of Lego© Mindstorms as a powerful pedagogical tool: Scaffolding learners through computational thinking and computer programming.

Janzen, I.; Thomas, M.; Angélica, Y. (2015): Wahlverhalten zum Schulfach Informatik in der SI - eine Studie im Regierungsbezirk Münster. In: Gallenbacher, J. (Hrsg.), Informatik allgemeinbildend begreifen. Bonn: Gesellschaft für Informatik e.V.. (S. 181-190). https://dl.gi.de/bitstream/handle/20.500.12116/2006/181.pdf?sequence=1&isAllowed=y

Peters, L.; Fahrendorff, N.; Debeye, D.; Alt, D. (2018): Nutzung von Robotern im Informatikunterricht – ein Lösungsvorschlag. In: Becker, M. (Hrsg.), SKILL 2018 - Studierendenkonferenz Informatik. Bonn: Gesellschaft für Informatik e.V.. (S. 119-130).

Samuelsen, D. A. H.; Graven, O. H. (2009): Low Cost Robots as Target System for Students Training Using Java. In: International Journal of Online Engineering.

Wiesner, B.; Brinda, T. (2007): Erfahrungen bei der Vermittlung algorithmischer Grundstrukturen im Informatikunterricht der Realschule mit einem Robotersystem. In: Schubert, S. (Hrsg.), Didaktik der Informatik in Theorie und Praxis – INFOS 2007 – 12. GI-Fachtagung Informatik und Schule. Bonn: Gesellschaft für Informatik e. V.. (S. 113-124).https://cs.emis.de/LNI/Proceedings/Proceedings112/gi-proc-112-010.pdf

1.2 Industrierobotik in der beruflichen Bildung

Harald Strating, Rene Egbers, Jonas Albersmann, Hochschule Osnabrück

1.2.1 Industrierobotik als Wirtschaftsfaktor für Deutschland

Industrielle Robotersysteme sind längst fester Bestandteil moderner Fertigungsstraßen und aus großen Industriezweigen nicht mehr wegzudenken, wie eine Analyse der Statistik[1] der *International Federation of Robotics* über Installationsdaten und operative Bestände von Industrierobotern weltweit verdeutlicht. Schon 1961 setzte die Firma General Motors den *Unimate* der Firma *Unimation* für die einfache Entladung von Spritzgussmaschinen ein. Mittlerweile haben Industrieroboter die Handhabungs- und Automatisierungstechnik komplett revolutioniert. Die weltweite Anzahl an Industrierobotern im operativen Bestand lag im Jahr 2021 - den Daten der *International Federation of Robotics* zufolge - bei 3.477.127 Stück. Der Markt für Industrieroboter wächst weiterhin unaufhaltsam. So wurden im Jahr 2021 fast 520.000 neue Roboter installiert (vgl. Bieller et al. 2022, S. 14.).

Diese globale Expansion basiert nicht zuletzt auf der stetigen Weiterentwicklung der Robotersysteme. Industrieroboter werden immer genauer, schneller und effizienter. Zudem erschließen sich durch moderne Technologien neue und erweiterte Anwendungsfelder für die Robotik. Die kollaborative Industrierobotik ist eines der neueren Anwendungsfelder der Robotik. Ausgefeilte interne Sicherheitssensorik ermöglicht es hier, dass Mensch und Roboter zu gleicher Zeit am gleichen Werkstück arbeiten können und die Sicherheit des Menschen zu jedem Zeitpunkt gewährleistet ist.

Von den ca. 520.000 neu installierten Robotern im Jahr 2021 waren etwa 39.000 kollaborative Industrieroboter (im Vorjahr belief sich diese Zahl auf etwa 26.000 neu installierte kollaborative Industrieroboter) (vgl. Bieller et al. 2022, S. 14). Die Wachstumsrate gegenüber dem Vorjahr 2020 entspricht für die Gesamtzahl an installierten Industrierobotern 31 % und dies trotz der schwierigen Produktionslage durch die Corona Pandemie (vgl. Müller 2022, S. 12). Weltweit bildet Europa nach China den zweitgrößten Absatzmarkt für Industrieroboter mit einem operativen Bestand von 678.706 Industrierobotern (Stand: 2021) und einem Wachstum von etwa 24 % im Jahr 2021. Deutschland bildet die europäische Spitze des Absatzmarktes für Industrieroboter und steht mit 23.777 neu installierten Industrierobotern im Jahr 2021 auf Platz fünf der weltweit am meisten installierten Industrieroboter 2021 hinter China (1), Japan (2), Amerika (3) und Korea (4). Dies entspricht einer Wachstumsrate von 6 % gegenüber dem Vorjahr und damit 5 % der im Jahr 2021 weltweit getätigten Roboterinstallationen (vgl. Bieller et al. 2022, S. 16). Werden diese Zahlen in Relation zu den Bevölkerungszahlen der jeweiligen Länder gesetzt, so wird deutlich, dass Deutschland im internationalen Vergleich eine hohe Roboterdichte aufweist. Die Relevanz der Industrierobotik als Wirtschaftsfaktor für Deutschland ist damit offensichtlich. Vorreiter beim Einsatz von Industrierobotern in der Industrie bilden hierzulande hochautomatisierte Industriezweige wie die Automobilindustrie, der Maschinenbau, die Gummi- und Kunststoffindustrie, die Elektronikindustrie sowie die Pharmaindustrie (vgl. Bieller et al. 2022, S. 27).

1 Die Statistik ist aus dem Jahr 2022 und bezieht sich auf das Geschäftsjahr 2021. Eine aktuellere Statistik aus dem Jahre 2023 ist gegenwärtig noch nicht verfügbar.

1.2.2 Tätigkeits- und Berufsfelder im Zusammenhang mit der Industrierobotik

Die Betrachtung der genannten internationalen Zahlen und Entwicklungen erklärt, dass Industrieroboter alltäglicher Arbeitsgegenstand vieler Personen sind, vorrangig in den oben genannten Industrien. Um die Rolle der Industrierobotik in der beruflichen Bildung einzuordnen, besteht eine zielführende Möglichkeit darin, die Lebensphasen von Industrierobotersystemen und die hierin jeweils auszuführenden Tätigkeiten sowie die Berufe zu betrachten, die für die Erledigung dieser Tätigkeiten zuständig sind (vgl. Schlausch 2017).

Von Industrierobotersystemen wird in diesem Zusammenhang gesprochen, da es sich bei Industrierobotern um unvollständige Maschinen im Sinne der Maschinenrichtlinie 2006/42/EG handelt und Roboter nur in Kombination mit einem Aktor (Endeffektor), einer Applikation und dem zu handhabenden Werkstück eine Maschine im Sinne der Maschinenrichtlinie bilden. Industrieroboter werden demnach als ganzes System, als Robotersystem, in Betrieben installiert.

Schlausch (2017) differenziert sieben Lebensphasen eines Robotersystems und ordnet ihnen jeweils spezifische Tätigkeiten der Facharbeit zu.

1. **Forschung und Entwicklung, Konstruktion und Arbeitsvorbereitung:** Entwicklung, Planung und Design der Roboterapplikation sowie die Vorbereitung der Produktion
2. **Herstellung:** Fertigung und Montage des Robotersystems
3. **Anpassung:** Spezifische Aufgabe der Roboterapplikation einrichten (programmieren)
4. **Roboterintegration:** Inbetriebnahme und Testen des Robotersystems sowie die Optimierung für den Serienlauf
5. **Produktion:** Betrieb und Bedienung des Robotersystems
6. **Instandhaltung:** Reparaturen, Wartungen und Störungsbeseitigungen am Robotersystem
7. **Modernisierung oder Rückbau:** Demontage und Überholung des Robotersystems oder die Entsorgung/das Recycling einzelner Komponenten des Systems

Bevor ein Industrieroboter eingesetzt wird, muss der Bedarf festgestellt und dann ein adäquates Industrierobotersystem entwickelt und konstruiert werden. Hierbei bestehen Entwicklungs-, Planungs- und Konstruktionsaufgaben sowie Aufgaben in der Arbeitsvorbereitung, die von Ingenieurinnen und Ingenieuren in Zusammenarbeit mit Produktionstechnologinnen und Produktionstechnologen ausgeführt werden.

Sind die vorbereitenden Tätigkeiten abgeschlossen, wird das Robotersystem hergestellt. Es muss gefertigt und montiert werden. Ebenfalls wird die spezifische Aufgabe der Roboterapplikation eingerichtet. Ist das Robotersystem montiert und eingerichtet, schließt die Phase der Roboterintegration an, in der die Inbetriebnahme erfolgt und Testläufe des Systems durchgeführt sowie Optimierungen für den Serienanlauf vorgenommen werden. All diese Tätigkeiten erfordern Kenntnisse über die Metall- und Elektrotechnik sowie teilweise über die Informationstechnik. In Betrieben arbeiten hierfür vorwiegend Industriemechanikerinnen und Industriemechaniker, Mechatronikerinnen und Mechatroniker und Elektronikerinnen und Elektroniker für Automatisierungstechnik zusammen. In den Phasen der Anpassung und der Roboterintegration werden je nach Komplexität des Robotersystems und des Ablaufprogramms bei Bedarf ebenfalls Fachinformatikerinnen und Fachinformatiker für Systemintegration involviert.

Nach der Integration des Roboters in sein Robotersystem und der Programmierung des Ablaufprogramms, führt der Roboter vollautomatisiert oder teilautomatisiert, so wie es im kollaborativen Betrieb der Fall ist, seine Tätigkeit aus. Der Betrieb und die Bedienung werden in dieser Phase vorwiegend durch angelernte Produktionsmitarbeiterinnen und Produktionsmitarbeiter oder beispielsweise durch Anlagenführerinnen und Anlagenführer, Industriemechanikerinnen und Industriemechaniker oder Konstruktionsmechanikerinnen und Konstruktionsmechaniker übernommen.

Die Phase der Produktion wird in geregelten Intervallen oder bei Störungen und Defekten immer wieder von Instandhaltungsphasen unterbrochen. In diesen müssen Wartungsarbeiten sowie Störungsbeseitigungen und Reparaturen durchgeführt werden. Diese Tätigkeiten werden abhängig von der Art der Wartung, Störung oder Reparatur durch Industriemechanikerinnen und Industriemechaniker, Mechatronikerinnen und Mechatroniker und Elektronikerinnen und Elektroniker für Automatisierungstechnik oder in Kombination durchgeführt.

Während der Lebenszeit des Robotersystems kann es zu Modernisierungsarbeiten kommen, bei denen es zu Überholungen des Systems oder Änderungen des Produktionsablaufs kommt. Diese Tätigkeiten werden ebenfalls von den drei aufgeführten Fachkräften übernommen. Diese Fachkräfte führen in der Regel auch die Demontage sowie das Recycling und die Entsorgung von Komponenten des Robotersystems nach dem Ablauf der Lebenszeit des Systems durch (vgl. Schlausch 2017, S. 5f.).

Bisher wurden in dieser Beschreibung die gängigsten Ausbildungsberufe genannt, die im Umgang mit den Industrierobotersystemen in der jeweiligen Lebensphase zusammenarbeiten. Je nach Einsatzgebiet können noch weitere Berufsfelder in den verschiedenen Lebensphasen an Robotersystemen tätig sein.

1.2.3 Robotik und ihre Rolle in den Rahmenrichtlinien verschiedener Ausbildungsberufe

Die Ausbildungsberufe der Mechatronikerinnen und Mechatroniker, der Industriemechanikerinnen und Industriemechaniker und der Elektronikerinnen und Elektroniker für Automatisierungstechnik weisen derzeit den wohl größten Bezug zur Industrierobotik auf. Daneben müssen noch die Produktionstechnologinnen und Produktionstechnologen für die Entwicklung, Konstruktion und Arbeitsvorbereitung ein Fachwissen rund um die Industrierobotik nachweisen. In den Phasen der Anpassung und der Roboterintegration können für die Erstellung komplexer Ablaufprogramme oder übergeordnete Steuerungsprogramme ebenfalls Fachinformatikerinnen und Fachinformatiker für Systemintegration herangezogen werden, diese benötigen jedoch kein umfangreiches Fachwissen im Themenfeld der Robotik.

Eine Analyse der Rahmenrichtlinien der Ausbildungsberufe für Mechatronikerinnen und Mechatroniker, Industriemechanikerinnen und Industriemechaniker, Produktionstechnologinnen und Produktionstechnologen und Elektronikerinnen und Elektroniker für Automatisierungstechnik zeigt, dass der Begriff Industrieroboter bisher in den Rahmenrichtlinien von keinem der genannten Ausbildungsberufe auftaucht[1]. Es finden sich eher weitläufig interpretierbare Begriffe der Automatisierungs- und Handhabungstechnik, in denen die beruflichen Handlungskompetenzen gefördert werden sollen. Diese Offenheit der Begrifflichkeiten und Formulierungen soll Schulen eine Anpassungsfähigkeit an die regional eingesetzten Technologien ermöglichen. Unter dem in den Lernfeldern viel benutzten Begriff der Handhabungsautomaten können also verschiedene technische Systeme verstanden und in der Schule behandelt werden. Hierbei können sich die Schulen an die regionalen Gegebenheiten richten. Werden in den Firmen der Umgebung der Schule hautsächlich Industrieroboter in der Handhabung oder anderen Anwendungsfeldern verwendet, so können Industrieroboter Bestandteil des Unterrichts der Lernfelder sein, in denen der Begriff Handhabungsautomaten auftaucht. Werden für diese Tätigkeit regional jedoch eher andere Handhabungsautomaten wie bspw. Vibrationswendelförderer eingesetzt, können diese wiederum Bestandteil des Unterrichts sein. Die begriffliche Offenheit ermöglicht einerseits die Offenheit bei der Behandlung ausgewählter Themen in der Schule, andererseits kann sie jedoch bei der Integration eines ausgewählten Inhalts in den Unterricht auf den ersten Blick als Störfaktor wirken, weil keine explizit passenden Begrifflichkeiten und Formulierungen auftauchen.
In der curricularen Analyse der Rahmenrichtlinien der Ausbildungsberufe für Mechatronikerinnen und Mechatroniker, Industriemechanikerinnen und Industriemechaniker, Produktionstechnologinnen und Produktionstechnologen und Elektronikerinnen und Elektroniker für Automatisierungstechnik sowie des beruflichen Gymnasium Technik, der Grundstufen und der Berufsfachschulen der Fachrichtungen Metall, Mechatronik und Elektronik wurden die entsprechenden Rahmenrichtlinien hinsichtlich der Integration von Inhalten der Robotik in die Lernfelder und Lerngebiete analysiert. Die Analyse zeigt, dass sich Inhalte der Robotik aus fünf Fachperspektiven heraus in den bestehenden Rahmenrichtlinien zuordnen lassen.

Eine erste Möglichkeit der Integration von Inhalten der Robotik in die Lernfelder ergibt sich durch eine Näherung über **steuerungstechnische Inhalte**. Das *Analysieren, Programmieren und Parametrieren steuerungstechnischer Systeme,* die Integration *analoger, digitaler und intelligenter Sensoren und Aktoren sowie die Verarbeitung von Signalen peripherer Geräte und Sensoren* stellen hier oft genannte Inhalte in den Rahmenrichtlinien dar.
Der Industrieroboter lässt sich als steuerungstechnisches System erfassen, welches sich analysieren, programmieren und parametrisieren lässt. Der wichtigste Bestandteil eines Roboters ist der Aktor (Endeffektor). Ohne ihn wäre der Roboter nur eine Aneinanderreihung von Gelenken. Die Verarbeitung vom Endeffektor ausgehender Signale ist also ein Kernbestandteil der Roboterprogrammierung. Ebenso wie die Verarbeitung von Signalen, die von Sensoren und Peripheriegeräten ausgehen, denn diese ermöglichen erst die Kommunikation mit anderen (Teil-)Systemen. Die Näherungsweise durch steuerungstechnische Inhalte bietet vor allem eine Einsicht in die Programmierstruktur sowie den steuerungstechnischen Aufbau von Robotersystemen.

Einen zweiten fachlichen Zugang liefern **fertigungstechnische Inhalte**. In der Industrie ergänzen Industrieroboter zumeist Fertigungsanlagen oder stellen Teilsysteme ganzer Fertigungsstraßen dar. Sie können jedoch auch selbst ein Fertigungssystem bilden. Bei der Näherung durch diese Perspektive finden sich entsprechende Inhalte in der *Strukturierung und Programmierung technischer Abläufe, bei der Integration von Teilsystemen in Ganzsysteme (Fertigungsanlagen, Robotersysteme) sowie bei der Optimierung von Fertigungs-, Montage- und Handhabungssystemen*.

1 Die Analyse der curricularen Vorgaben erfolgt auf der Grundlage der aktuell gültigen Rahmenlehrpläne der genannten Ausbildungsberufe, die auf der Homepage der Kultusministerkonferenz abrufbar sind (www.kmk.org).

Eine dritte Fachperspektive bietet die Suche über **wartungs- und instandhaltungstechnische Inhalte**. Robotersysteme, ihre Aktoren, ihre Sensoren und Peripheriegeräte, aber auch ihre Sicherheitsfunktionen unterliegen als technische (Teil-)Systeme bestimmten Wartungs- und Instandhaltungsintervallen, die durchgeführt werden müssen, um die Funktionsfähigkeit des Systems sowie dessen Sicherheit zu gewährleisten. Instandhaltung und Wartung sind in den Rahmenrichtlinien der Ausbildungsberufe, für die die Robotik in Frage kommt, fast durchgängig Bestandteil der Ausbildung. Robotersysteme oder deren Komponenten können hier durchgängig integriert werden.

Eine vierte Perspektive für die Integration von Inhalten der Robotik bieten **sicherheitstechnische Inhalte**. Robotersysteme unterliegen je nach Tätigkeitsbereich vielfältigen und vielschichtigen Sicherheitsvorgaben. Besonders die kollaborierende Robotik bietet mit ihren Zusatzfunktionen der Kraft- und Leistungsbegrenzung, der Handführung, dem sicherheitsbewerteten überwachten Halt und der Geschwindigkeits- und Anstandsüberwachung ein breites Feld an zusätzlichen Sicherheitseinstellungen zu den sonstigen, wie Sicherheitszäune, Trittmatten usw. Hierfür werden in den curricularen Vorgaben Inhalte wie das *Prüfen, Justieren und Einstellen von Sicherheitseinrichtungen, die Beachtung der Betriebssicherheit und des Gesundheits- und Arbeitsschutzes* sowie das *Messen und Prüfen sicherheitsrelevanter Funktionen* aufgeführt.

Eine letzte durchaus interessante Interpretationsweise liegt in der Anpassung eines Robotersystems an **konstruktive Erweiterungen**. Hierbei werden besonders handhabungstechnische Aspekte zum Lerninhalt behandelt. Ausgehend von einer Problemstellung sollen Komponenten eines Robotersystems erstellt werden, die an den Prozess angepasst und optimiert sind. Bei dieser Betrachtungsweise stehen nicht die Programmierung und die Steuerungstechnik im Fokus, sondern der Prozessablauf, die Problemstellung und das Konstruieren von Komponenten. Dies ermöglicht eine Integration von Robotikinhalten bereits in den ersten Lernfeldern der metalltechnischen Ausbildungsberufe. Inhalte wie die *Erstellung von Teil- oder Ganzzeichnungen und deren zugehörigen Arbeitspläne ausgehend von Problemstellungen, die Auswahl von passenden Werkstoffen unter Berücksichtigung ihrer spezifischen Eigenschaften* sowie die Erstellung und Konstruktion ganzer Baugruppen kommen somit in Betracht.

Es bleibt festzustellen, dass Robotersysteme aus verschiedenen Fachperspektiven in den berufsbildenden Unterricht eingebunden werden können und dass ein vielfältiges Spektrum an verschiedenen metalltechnischen, elektrotechnischen, mechatronischen und informationstechnischen Kenntnissen durch Robotersysteme vermittelt werden können.

1.2.4 Der kollaborative Industrieroboter als Lernmedium in der beruflichen Bildung

Kollaborative Industrieroboter haben eine besondere Eignung als Lernmedium in der beruflichen Bildung. In erster Linie wurden kollaborative Industrieroboter für das Projekt Robonatives vorgesehen, da diese die Zusammenarbeit mit Robotern für Schülerinnen und Schüler zugänglicher machen. Konventioneller Robotikunterricht in der Berufsschule findet aufgrund der Sicherheitsvorgaben und zum Schutz der Schülerinnen und Schüler mit eingehausten Industrierobotern statt. Bei kollaborativen Industrieroboter ist eine Einzäunung des Robotersystems nicht zwingend notwendig. Dadurch wird ermöglicht, dass Schülerinnen und Schüler in direktem Kontakt mit den Robotern arbeiten können. Bevor die kollaborativen Industrieroboter jedoch für den Einsatz im Unterricht genutzt werden können, müssen die Sicherheitsparameter des Systems soweit angepasst werden, dass keine Gefahren beim Betrieb des Roboters für die Schülerinnen und Schüler entstehen kann. Ebenfalls muss das gesamte Robotersystem den Anforderungen für den kollaborativen Betrieb entsprechen. Diese sind in den Normen DIN EN ISO 10218 Teil 1 und 2 festgehalten. Für den Einsatz kollaborativer Industrieroboter als Lernmedium in der Schule ist es ebenfalls empfehlenswert, dass eine Mensch-Roboter-Kollaborationsmessung nach DIN ISO/TS 15066 durchgeführt wurde. Zwingend notwendig ist, dass aus dem Ergebnis der Gefährdungsbeurteilung für die jeweilige Schülerschaft der unbedenkliche Umgang mit den Robotersystemen hervorgeht. Zu keinem Zeitpunkt darf eine Gefahr vom Robotersystem ausgehen. Kann dies nicht gewährleistet werden, sind zusätzliche Schutzmaßnahmen zu treffen, um einen sicheren Umgang mit dem kollaborativen Industrieroboter zu gewährleisten.

Gelingt die sichere Umsetzung des kollaborativen Robotersystems für den kollaborativen Einsatz mit Schülerinnen und Schülern, bietet das System den Vorteil, dass die Schülerinnen und Schüler den Roboterprozess nahbar erfahren können. Dies kann Hemmungen und Ängste vor dieser Art von Technologie mindern und somit das Erreichen der Bildungsziele stützen. Einen der wichtigsten Faktoren für die Integration von kollaborativen Industrierobotern stellt die Sicherheitstechnik dar. Beim konventionellen Industrieroboter ist die Sicherheitstechnik ein Gebiet, welches nur von ausgewählten Berufsbildern im Detail behandelt werden muss. Für die anderen Berufsbilder reichen hierfür grundlegende Informationen. Bei der kollaborativen Robotik verhält sich dies anders. Das Aushängeschild dieser Art von Industrieroboter ist die Sicherheit in der Zusammenarbeit von Mensch und Roboter. Die Sicherheit des Systems wird zu einem zentralen Lernaspekt für alle Schülerinnen und Schüler, die an diesen Systemen arbeiten. Ebenfalls bieten kollaborative Industrieroboter ganz neue Anwendungsfelder für den Unterricht und decken aktuelle, zukunftsweisende gesellschaftliche und ethische Sichtweisen auf die Robotertechnologie auf. Roboter sind nicht mehr nur eigenständig arbeitende Systeme, die aufgrund ihrer ausgehenden Gefahr für den Menschen abgeschirmt von diesem existieren, sie dürfen vielmehr mit dem Menschen zusammen Hand in Hand arbeiten, wobei sie den Menschen bei seiner Arbeit unterstützen und entlasten. Dies macht den Roboter zu einem idealen Lernmedium für die berufsbildende Schule.

1.2.5 Curriculare Analyse relevanter Ausbildungsberufe sowie vollzeitschulischer Bildungsgänge

Diese nachfolgende curriculare Analyse behandelt mit den Ausbildungsberufen für Industriemechanikerinnen und Industriemechaniker, Mechatronikerinnen und Mechatroniker, Produktionstechnologinnen und Produktionstechnologen und Elektronikerinnen und Elektroniker für Automatisierungstechnik nur eine kleine Auswahl an Ausbildungsberufen, für die die Industrierobotik als Unterrichtsthema in Betracht kommt. Ausgewählt wurden diese Berufe, da bei ihnen spezifische Tätigkeits- und Berufsfelder im Zusammenhang mit der Industrierobotik besonders hervortreten. Überdies bietet diese Konstellation der Ausbildungsberufe den Vorteil, dass jeweils Rahmenrichtlinien eines Berufes der Fachrichtungen Metalltechnik, Elektrotechnik und Mechatronik analysiert werden. Die genannten Formulierungen lassen sich jedoch auf andere ähnliche curriculare Inhalte übertragen. Auch wurden nur Lernfelder betrachtet, bei denen die der gegebenen Zielformulierung und Inhalte so zutreffend auf die Robotik ausgelegt sind, dass diese für die Gestaltung umfangreicher Lernsituationen ausreichen. Die Robotik kann also auch in anderen Lernfeldern als den in der nachfolgenden Tabelle genannten, für kleinere Inputs eingesetzt werden.

Neben den Rahmenrichtlinien für die genannten Ausbildungsberufe wurden die curricularen Vorgaben verschiedener vollzeitschulischer Bildungsgänge der jeweiligen Fachrichtungen analysiert. Die Zielformulierungen und Inhalte der Berufsfachschule Metalltechnik überschneiden sich mit den nachfolgend angeführten Inhalten zu den Lernfeldern 1 – 4 der metalltechnischen Berufe. Alle Zielformulierungen und Inhalte der Grundstufe und Berufsfachschule Elektrotechnik sowie Mechatronik sind stark auf einfache elektronische Systeme ausgerichtet und weniger auf komplexe Systeme wie die eines Roboters. Eine Behandlung der Thematik bietet sich daher erst in den höheren Lernfeldern an.

Des Weiteren wurden die Lehrpläne für das Berufliche Gymnasium Technik mit seinen Fachrichtungen Bautechnik, Metalltechnik, Informationstechnik, Mechatronik, Elektrotechnik und Gestaltungs- und Medientechnik analysiert. Für die Fachrichtungen Bautechnik sowie Gestaltungs- und Medientechnik wurden keine speziellen Bezüge zur Robotik festgestellt. Für die Fachrichtung Informationstechnik fokussieren die Richtlinien eher auf andere Inhalte, vergleichbar mit der genannten Grundstufe bzw. Berufsfachschule Elektrotechnik und Mechatronik.

Die Rahmenrichtlinien für die Fachschule Technik wurden hier nicht im Detail analysiert, da die Angaben in den Rahmenrichtlinien technologieoffen formuliert sind und bezüglich der Technologien die ganze Bandbreite von der Montage über die Einrichtung und Wartung bis hin zur Demontage und Entsorgung abdecken. Damit sind alle aufgeführten Optionen zur Integration von Robotersystemen möglich.

In folgender Tabelle sind der Übersichtlichkeit halber die in dieser Analyse betrachteten Ausbildungsberufe und Schulformen mit den Lernfeldern verzeichnet, die für die Thematisierung der Robotik besonders in Frage kommen. Lernfelder, die für die Thematisierung der Robotik angedacht sind, sind mit einem X gekennzeichnet. Lernfelder, die nicht oder nur in geringem Maße für die Thematisierung der Robotik in Betracht kommen, sind nicht markiert.

Ausbildungsberuf und Lernfeld	Industriemechanikerinnen und Industriemechaniker	Mechatronikerinnen und Mechatroniker	Produktionstechnologinnen und Produktionstechnologen	Elektronikerinnen und Elektroniker für Automatisierungstechnik		Berufliches Gymnasium Technik	BG Metalltechnik	BG Elektrotechnik	BG Mechatronik	BG Informationstechnik
LF1						Qualifikationsphase				
LF2	X					SP1	X			X
LF3	X					SP2		X	X	
LF4						$SP3_{(WP)}$				
LF5						$SP4_{(WP)}$			X	
LF6	X	X		X		$SP5_{(WP)}$				
LF7		X		X		SP6				
LF8		X	X							
LF9	X		X							
LF10		X	X	X						
LF11				X						
LF12	X	X	X	X						
LF13	X	X		X						
LF14										
LF15	X									

Tabelle 2: Lernfelder ausgesuchter Ausbildungsberufe, in denen Robotik thematisiert werden kann

Lernfelder 1-4 Metalltechnik

Die Lernfelder 1 – 4 der metalltechnischen Berufe sind als einheitliche metalltechnische Grundstufe inhaltsgleich formuliert. Der Industrieroboter kann insbesondere in den Lernfeldern 2 und 3 als Lernobjekt verwendet werden. Ermöglicht wird die Betrachtung von Prozessen, die vom Industrieroboter ausgeführt werden sollen. Hierfür soll die Applikation durch die Auszubildenden entwickelt und konstruiert werden. Der Fokus liegt hierbei auf handhabungstechnischen Prozessen sowie auf der kreativen Gestaltung von technischen (Teil-)Systemen auf der Basis technischer Problemstellungen. So bieten sich die folgenden Lernfelder wie beschrieben an.

Lernfeld 2: Fertigen von Bauelementen mit Maschinen

Lernfeld 2 stellt ein Lernfeld dar, bei dem sich eine Problemstellung eines robotertechnischen Prozesses anbietet, welche sich durch konstruktive Gestaltung an Applikation oder Endeffektor lösen lässt.
Die Schülerinnen und Schüler können ausgehend von der Problemstellung eigenständig „Teilzeichnungen und die dazugehörigen Arbeitspläne, auch mit Hilfe von Anwendungsprogrammen zum rechnerunterstützten Zeichnen“, erstellen und ändern. Sie können „Werkstoffe unter Berücksichtigung ihrer spezifischen Eigenschaften [auswählen] und […] sie produktbezogen“ der Applikation oder dem Endeffektor zuordnen. Anschließend können die Produkte hergestellt und ausprobiert werden sowie hinsichtlich alternativer Möglichkeiten und Fertigungsverfahren bewertet werden. Arbeits- und Umweltschutz verstehen sich in diesem Lernfeld als zentraler Vermittlungsbestandteil und sollten jederzeit betrachtet werden. Die Schülerinnen und Schüler würden frühzeitig mit Robotersystemen in Kontakt kommen, ohne sie selbst zu programmieren. Jedoch würden sie sich im Kontext der Problemstellung bereits mit dem Bewegungsverhalten sowie den handhabungstechnischen Aspekten von Robotersystemen auseinandersetzen.

Handhabungsspezifische Inhalte wie Ordnungszustände, Positionierungs- und Orientierungsgenauigkeiten usw. können mit fertigungstechnischen Inhalten wie „ISO-Toleranzen“, „Oberflächenangaben“, „Messfehler“ usw. verknüpft werden. Die Auswirkungen der fertigungstechnischen Inhalte auf die handhabungstechnischen Inhalte können durch den Einsatz von Industrierobotersystemen erfahrbar gemacht werden.

Lernfeld 3: Herstellen von einfachen Baugruppen

Das Lernfeld 3 bietet sich als Weiterführung der in Lernfeld 2 erarbeiteten Inhalte an. Die dort beschriebenen Inhalte beziehen sich nun jedoch nicht mehr nur auf ein Bauelement, sondern auf eine Baugruppe, was die Auswahl an Problemstellungen hinsichtlich Applikations- und Endeffektorgestaltung erweitert. Eine Weiterführung der Kopplung handhabungstechnischer Inhalte mit konstruktiven Inhalten ist denkbar. Beispielsweise wären Werkstückmerkmale des zu handhabenden Objektes (Werkstück) kombinierbar mit dem fertigungstechnischen Inhalt des „kraft-, form- und stoffschlüssigen Fügens“.

Lernfeld 4: Warten technischer Systeme

> Die Schülerinnen und Schüler bereiten die Wartung von technischen Systemen insbesondere von Betriebsmitteln vor und ermitteln Einflüsse auf deren Betriebsbereitschaft. Dabei bewerten sie die Bedeutung dieser Instandhaltungsmaßnahme unter den Gesichtspunkten Sicherheit, Verfügbarkeit und Wirtschaftlichkeit.
> Sie lesen Anordnungspläne, Wartungspläne und Anleitungen, auch in englischer Sprache. Die Schülerinnen und Schüler nutzen digitale Informationsquellen.
> Sie planen Wartungsarbeiten und bestimmen die notwendigen Werkzeuge und Hilfsstoffe. Sie wenden die Grundlagen der Elektrotechnik und der Steuerungstechnik an und erklären einfache Schaltpläne in den verschiedenen Gerätetechniken.
> **Inhalte:** Grundbegriffe der Instandhaltung; Wartungspläne; Anordnungspläne; Betriebsanleitungen; Betriebsorganisation; Verschleißursachen, Störungsursachen; Funktionsprüfung; Instandhaltungs- und Ausfallkosten, Störungsfolgen; Schadensanalyse; Größen im elektrischen Stromkreis, Ohmsches Gesetz; Gefahren des elektrischen Stroms, elektrische Sicherheit; Normen und Verordnungen

Industrieroboter müssen wie jedes andere technische System in regelmäßigen Abständen gewartet werden. Besonders die Schutzeinrichtungen und sicherheitstechnischen Funktionen sind zu pflegen. Aufgrund der Komplexität von Industrierobotersystemen und der erforderlichen Sicherheitstechnik wird hier jedoch empfohlen, die Wartung von Industrierobotersystemen erst in den höheren Lernfeldern zu behandeln. Viele der metalltechnischen Ausbildungsberufe weisen hierfür beispielsweise das Lernfeld der Instandhaltung auf.

Industriemechaniker(in)

Rahmenlehrplan für den Ausbildungsberuf Industriemechaniker/Industriemechanikerin (Beschluss der Kultusministerkonferenz vom 25.03.2004 in der Fassung vom 23.02.2018)

Lernfeld 6: Installieren und Inbetriebnehmen steuerungstechnischer Systeme

> Die Schülerinnen und Schüler installieren steuerungstechnische Systeme und nehmen sie in Betrieb.
> **Inhalte**: Sensoren und Aktoren; Betriebsarten; Anlagensicherheit[1]

Das Robotersystem findet hier Anwendung, in dem an ihm als Lernträger verschiedene Aktoren (Endeffektoren) und/oder Sensoren (fotoelektrische Sensoren, Kamerasysteme) installiert, angesteuert und getestet werden können. Hierbei handelt es sich um steuerungstechnische Systeme, die für den Handhabungsprozess relevant sind. Robotersysteme bieten für die Anbindung an diese Komponenten verschiedene Schnittstellen wie z. B. MODBUS oder I/O´s.

Ebenfalls kann ein Robotersystem die geforderten Inhalte Betriebsarten und Anlagensicherheit abbilden. So kommt beispielsweise mit den kollaborativen Industrierobotern eine besondere Betriebsart hinzu, bei der Mensch und Roboter zusammenarbeiten dürfen, und ebenfalls bedienen Industrieroboter gängige Betriebsarten.

Bezüglich der Anlagensicherheit unterliegen Robotersysteme, wie alle Maschinen, gängigen europäischen Sicherheitsrichtlinien und sie können somit als Lernmedium dienen. Interessant wäre in diesem Zusammenhang die Vielfalt der verschiedenen Stufen der Zusammenarbeit, vom Roboter in einer Zelle über die sequenzielle Zusammenarbeit bis hin zur responsiven Zusammenarbeit, bei der Mensch und Roboter in Echtzeit zusammen an einem Werkstück arbeiten. Es gibt 5 Stufen der Zusammenarbeit zwischen Mensch und Roboter und jede dieser Stufen bedingt unterschiedliche sicherheitsrelevante Betrachtungsweisen. Bei der ersten wird die Sicherheitsbetrachtung vollständig auf die Abschottung des Robotersystems gesetzt, während bei der letzten die interne Sensorik so fein justierbar ist, dass Mensch und Roboter Hand in Hand arbeiten können und hierbei jederzeit die Sicherheit der Person gewährleistet ist.

Lernfeld 9: Instandsetzen von technischen Systemen

> Die Schülerinnen und Schüler setzen technische Systeme instand. Sie planen Instandsetzungsmaßnahmen. Sie planen Instandsetzungsmaßnahmen für technische Systeme unter Berücksichtigung betrieblicher und wirtschaftlicher Forderungen. Dazu beschaffen sie die notwendigen technischen Informationen.
> Die Schülerinnen und Schüler demontieren Teilsysteme in Baugruppen und Bauelemente unter Berücksichtigung der jeweiligen Schnittstellen und wählen die erforderlichen Werkzeuge und Hilfsmittel aus.
> Die Schülerinnen und Schüler prüfen die Funktion und bereiten die Abnahme vor.
> Sie planen die fachgerechte Entsorgung der defekten Teile und der verbrauchten Hilfsstoffe.
> Sie wenden die Bestimmungen zur Arbeitssicherheit und zum Umweltschutz an.
> **Inhalte**: Gesamtzeichnungen; Schaltpläne; zustand- und ausfallbedingte Instandsetzung; Verschleiß; Fehleranalyse; Demontage-/Montagepläne

Robotersysteme sind komplexe Handhabungssysteme, die aus verschiedenen Teilsystemen bestehen. Der Industrieroboter selbst stellt im Sinne der Maschinenrichtlinie eine unvollständige Maschine dar (vgl. DGUV 2015, S. 23). Erst in Kombination mit anderen (Teil-)Komponenten bilden sie eine vollständige Maschine im Sinne der Maschinenrichtlinie. Zu diesen (Teil-)Komponenten gehören u.a. Endeffektoren, externe Sicherheitseinrichtungen. Die (Teil-)Systeme sind eng an den Handhabungsprozess gebunden und können/müssen je nach Aufgabe ausgetauscht oder durch andere Komponenten ersetzt werden. Es gibt also viele (Teil-)Systeme, die in Betrieb genommen werden müssen, bzw. verschiedene Lebenserwartungen aufweisen und ersetzt bzw. wieder instandgesetzt werden müssen. Die oben aufgeführten Inhalte können also an Robotersystemen behandelt werden. Ebenfalls kann anhand von Robotersystemen der Sachverhalt unvollständiger Maschinen erläutert werden.

1 Eingerückte Textpassagen sind direkt übernommene Textpassagen aus den im jeweiligen Kapitel oben genannten Rahmenlehrplänen für die Ausbildungsberufe.

Lernfeld 12: Instandhalten von technischen Systemen

Das Robotersystem als technisches System kann hier Betrachtung finden. Roboter besitzen wie andere technische Systeme gewisse Wartungsintervalle, die es zu thematisieren gilt. Bei kollaborierenden Robotersystemen wird empfohlen, die MRK-Messung aufgrund von Verschleiß in bestimmten zeitlichen Intervallen zu wiederholen. Hierdurch wird ein direkter Zusammenhang zwischen Arbeitssicherheit und Instandhaltung deutlich.

Lernfeld 13: Sicherstellen der Betriebsfähigkeit automatisierter Systeme

Robotersysteme sind zumeist in Fertigungsstraßen eingebunden und/oder bilden ein Teilsystem eines übergeordneten Systems. Ein Ausfall des Systems hätte also einen Ausfall der ganzen Fertigungsstraße oder des übergeordneten Prozesses zur Folge. Ein Stillstand kann demnach zu hohen ökonomischen Problemen führen. Die Sicherstellung der Betriebsfähigkeit des Systems ist folglich elementar für die Sicherstellung der Betriebsfähigkeit übergeordneter Systeme. Die Zielformulierungen und Inhalte des Lernfeldes 13 lassen sich demnach in Gänze auf Industrieroboter übertragen.

Lernfeld 15: Optimieren von technischen Systemen

Robotersysteme können als Optimierungsmaßnahme für technische Systeme verwendet werden. Alternativ kann das Robotersystem selbst optimiert werden. So werden Robotersysteme beispielsweise als Optimierung für den Aus- und Einspannprozess an CNC-Drehbänken oder Fräsen verwendet, um die Taktzeit zu erhöhen oder einen kontinuierlichen Materialfluss zu generieren. Auch können manche Prozesse durch Robotersysteme erst abgebildet werden, da diese für den Menschen zu gefährlich wären (bspw. Handhabung giftiger Stoffe). Auch das Robotersystem selbst kann als optimierungsbedürftiges System betrachtet werden. Die Inhalte des Lernfeld 15 sind also ebenfalls vollständig auf Industrieroboter übertragbar.

Mechatroniker(in)

Rahmenlehrplan für den Ausbildungsberuf Mechatroniker/Mechatronikerin (Beschluss der Kultusministerkonferenz vom 30.01.1998 in der Fassung vom 23.02.2018)

Lernfeld 6: Planen und Organisieren von Arbeitsabläufen

> Die Schülerinnen und Schüler beschreiben die betrieblichen Organisationsstrukturen und organisieren die Teamarbeit auch interdisziplinär und nach funktionalen, fertigungstechnischen und ökonomischen Kriterien.
> **Inhalte**: Analyse von Arbeitsabläufen; Ergonomie und vorbeugender Unfallschutz; Darstellungsverfahren von Arbeitsabläufen; Wirtschaftlichkeit, Organisations- und Produktionsabläufe.

Die Betrachtung von Arbeitsabläufen auf ihre Eigenschaften hin kann durch die Implementierung der Untersuchung von Möglichkeiten des Einsatzes kollaborativer Roboter erweitert werden. Das Lernfeld erfasst von technischen und ökonomischen Parametern auch Gesichtspunkte des Gesundheits- und Arbeitsschutzes. Diese - genutzt als Ausgangspunkt für die Untersuchung von Vor- und Nachteilen des Ersetzens des bisherigen Fertigungsprozesses durch einen Roboter - vertiefen das Verständnis der Einsatzspektren und Grenzen der Robotik.

Lernfeld 7: Realisieren mechatronischer Teilsysteme

> Sie kennen Möglichkeiten zur Realisierung von Linear- und Rotationsbewegungen mittels elektrischer, pneumatischer und hydraulischer Komponenten und wenden Kenntnisse über Steuerungen und Regelungen an, um Weg- und Bewegungsrichtung zu beeinflussen.
> Einfache Programmierverfahren werden beherrscht.
> **Inhalte**: Wirkungsweise von Sensoren und Wandlern; Signalverhalten von Sensoren und Wandlern; Programmierung von einfachen Bewegungsabläufen und Steuerungsfunktionen; Prozessdaten auslesen, verarbeiten und interpretieren.

Das Lernfeld kann genutzt werden, um die wichtigsten Grundlagen eines Roboters und eines mechatronischen Systems zu verdeutlichen. Die Robotik kann als Ausgangspunkt für des Erlernen der verschiedenen Bewegungsarten und -richtungen dienen und darüber hinaus auch den zielgerichteten Einsatz von Sensorik im interdisziplinären Zusammenspiel von Roboter und mechatronischen System einführen. Durch einen überschaubar gestalteten Programmieraufwand können Schülerinnen und Schüler den Umgang mit einfachen Programmieraufgaben erlernen.

Lernfeld 8: Design und Erstellen mechatronischer Systeme

> Sie bestimmen die technischen Parameter erforderlicher Schutzeinrichtungen und wählen diese aus.
> Vorschriften des Arbeits- und Gesundheitsschutzes werden von ihnen beachtet.
> **Inhalte**: Funktionsweise, Auswahl und Einstellung von Schutzeinrichtungen; Positionierungsvorgänge, Freiheitsgrade; Programmieren von Bewegungsabläufen und Steuerungsfunktionen.

Das Lernfeld setzt sich unter anderem mit den Vorschriften des Arbeits- und Gesundheitsschutzes auseinander und kann im Kontext des möglichen Einsatzes eines kollaborativen Roboters dazu genutzt werden, unterschiedliche rechtliche und technische Rahmenbedingungen eines solchen Einsatzes genauer zu untersuchen. Die Schülerinnen und Schüler analysieren Gefahrenquellen im Umgang mit der Robotik und bewerten diese kritisch. Sie analysieren die Sicherheitsparameter kollaborativer Industrieroboter und bewerten diese kritisch.

Lernfeld 10: Planen der Montage und Demontage

> Die Schülerinnen und Schüler beherrschen die Planung und Vorbereitung der Montage und Demontage mechatronischer Systeme. Sie erklären den Ablauf der Arbeitsprozesse und können Arbeitsergebnisse beurteilen.
> Sie beziehen bereits in der Vorbereitungsphase Aspekte des Gesundheits- und Arbeitsschutzes in ihre Überlegungen ein. Sie überprüfen Montagebedingungen am Aufstellungsort und berücksichtigen sie.
> **Inhalte**: Betriebliche Montageunterlagen; Sicherheitsmaßnahmen und deren Prüfung; Prüfungen während der Montage.

Wie bei den Industriemechanikerinnen und Industriemechanikern im Lernfeld 9 werden hier Aspekte der Montage und Demontage behandelt, die auf den Industrieroboter übertragbar sind.
Der Aufstellungsort für Roboter spielt im Hinblick auf mögliche Gefahrenquellen für arbeitende Personen im Umfeld eine entscheidende Rolle. Dieser Aspekt muss im Planungs- und Entscheidungsprozess des Einsatzes Betrachtung finden. Mögliche Gefährdungsszenarien im Umfeld des Roboters müssen entweder durch Sperrzonen oder durch Anpassungen des kollaborativen Roboters im Vorfeld getroffen werden.

Lernfeld 11: Inbetriebnahme, Fehlersuche und Instandsetzung

> Die Schülerinnen und Schüler stellen die Gesamtfunktion und die Teilfunktion eines Systems einschl. seiner Schutzeinrichtungen dar. Dazu entnehmen sie Informationen aus technischen Unterlagen.
> Die Schülerinnen und Schüler erläutern die Verfahren zur Inbetriebnahme von mechatronischen Systemen und legen die Vorgehensweise für die Inbetriebnahme eines Gesamtsystems fest.
> Sie nutzen die Möglichkeiten von Diagnosesystemen und interpretieren Funktions- und Fehlerprotokolle. Die Wirksamkeit von Schutzmaßnahmen wird von ihnen überprüft.
> Sie justieren Sensoren und Aktoren, überprüfen Systemparameter und stellen sie ein. Ergebnisse werden in Unterlagen dokumentiert. Sie grenzen Fehler systematisch ein beseitigen Störungen.
> **Inhalte**: Blockschaltbilder, Wirkungs- und Funktionspläne von mechatronischen Systemen; Überprüfung und Einstellung von Sensoren und Aktoren; Systemparameter; BUS Parametrierung; Softwareanwendung; Elektrische und mechanische Schutzmaßnahmen, Schutzvorschriften; Prozessvisualisierunq, Diagnosesysteme, Ferndiagnose; Qualitätssicherungsverfahren; Behebung von Programmfehlern; Einflüsse von mechatronischen Systemen auf ökonomische, ökologische und soziale Bedingungen.

Die Zielformulierungen und Inhalte des Lernfeld 11 der Mechatronikerinnen und Mechatronikern sind hinsichtlich der Inbetriebnahme und Instandsetzung identisch mit denen der Lernfelder 9 und 11 der Industriemechanikerinnen und Industriemechanikern. Die Fehlersuche betreffend, können die Kompetenzen der systematischen Fehleranalyse durch die Nähe von Robotersystemen zu mechatronischen Systemen und den damit einhergehenden Behebungen von Programmfehler durch den Einsatz eines Roboters erlernt werden. Das Überprüfen von Schutzmaßnahmen und -einrichtungen ist auch im Arbeitsfeld der Robotik elementar.

Lernfeld 13: Übergabe von mechatronischen Systemen an Kunden

> Die Schülerinnen und Schüler bereiten Informationen über mechatronische Systeme textlich und grafisch auch in digitaler Form auf und präsentieren sie.
> Sie planen die Einweisung von Betriebs- und Bedienungspersonal in die Anlage und führen diese durch.
> **Inhalt**: Bedienungsanleitungen, Betriebsanleitungen

Besonders Robotersysteme benötigen aufgrund ihrer breitgestreuten Einsatzmöglichkeiten und ihrer oft für das anwendende Bedienungspersonal neuartigen Umgangseigenschaften eine klare und im Vorhinein verständlich strukturierte Einweisung. Das Lernfeld befasst sich mit den Überlegungen der Übergabe von mechatronischen Systemen an Kunden und bietet den Schülerinnen und Schülern die Möglichkeit, sich über das eigene Verständnis hinaus mit den Kompetenzen zum adäquaten Umgang mit Robotersystemen auseinanderzusetzen. Aufgrund der Neuartigkeit des Einsatzes von Robotern in bestimmten Branchen ist die Betrachtung einer geeigneten Einweisung in das System von entscheidenderer Rolle im Kontext der Gefahrenvermeidung.

Produktionstechnolog(in)

Rahmenlehrplan für den Ausbildungsberuf Produktionstechnologe/Produktionstechnologin (Beschluss der Kultusministerkonferenz vom 15.02.2008)

Lernfeld 8: Auftragsanalyse und Projektmanagement

> Die Schülerinnen und Schüler wenden die Methoden des Projektmanagements zur Gestaltung betrieblicher Prozesse an. Dazu analysieren sie betriebliche Aufträge zur Gestaltung von Fertigungs-, Montage-, Handhabungs- oder Logistikprozessen. Sie beschaffen die zur Abwicklung des Projektes erforderlichen Informationen, erfassen die Randbedingungen und erstellen ein Lastenheft.
> Sie nutzen effiziente Verfahren und Methoden zur Planung von Projekten und wenden Projektplanungssoftware an. Sie entwickeln Lösungsalternativen und bewerten diese. Sie planen die Einführung der ausgewählten Lösung in die Produktion und erstellen die erforderlichen Dokumente.
> **Inhalte**: Ablauforganisation; Prozessgliederungsplan

Die Schülerinnen und Schüler sollen in diesem Lernfeld die Methoden des Projektmanagements auf betriebliche Prozesse anwenden und hierfür unter Anderem Handhabungsprozesse analysieren. Ferner sind für diese Prozesse Lösungsalternativen zu suchen. Bestandteil dieses Lernfeldes kann demnach der Vergleich zwischen manuellen Handhabungsprozessen und Handhabungsprozessen mit Industrieroboter sein. Bei den Vergleichstätigkeiten müssen sich die Schülerinnen und Schüler intensiv mit den handhabungstechnischen Aspekten auseinandersetzen.

Lernfeld 9: Einrichten von Handhabungs- und Materialflusssystemen

In Lernfeld 9 treffen alle Zielformulierungen sowie Inhalte auf Robotersysteme als Handhabungssysteme zu. Somit wird eine Vielzahl von Betrachtungsansätzen geboten.
Die Schülerinnen und Schülerkönnen Robotersysteme als „Handhabungssysteme in flexible Fertigungsanlagen integrieren“ sowie „Technische Anforderungen für Robotersysteme und steuerungstechnische Systeme beschreiben“. Zusätzlich sollen die Schülerinnen und Schüler „die Signale der Peripheriegeräte und der Sensoren über Schnittstellen mit der Ablaufsteuerung“ verknüpfen. Sie sollen eigenständig Handhabungssysteme einrichten und die Funktion der Sicherheitseinrichtungen überprüfen. Weitere Punkte wären das Testen und Optimieren von Programmabläufen und eine Projektaufgabe.

Lernfeld 11: Simulieren von Produktionsprozessen

Auch in Lernfeld 11 der Produktionstechnologinnen und Produktionstechnologen treffen alle Zielformulierungen und Inhalte auf den Einsatz von Industrierobotern als Lernmedium zu.
In Lernfeld 11 sollen komplexe Aufgabenstellungen analysiert und Ziele sowie Vorgehensweisen für die Simulation von Gesamt- und Teilprozessen festgelegt werden. Hierbei sind „insbesondere Fertigungs-, Montage-, Handhabungs- und Logistiksysteme sowie Kombinationen dieser Systeme“ zu betrachten. Roboterprozesse können gerade bei komplexen Ablaufprogrammen, der Zusammenarbeit von Industrierobotern und anderen Maschinen oder der Zusammenarbeit mehrerer Industrieroboter in einer Zelle aufwendig und für die Programmiererinnen und Programmierer sehr anspruchsvoll werden. Eine Abhilfe schaffen hierbei Simulationsprogramme für Industrieroboter. Schülerinnen und Schüler können im Laufe dieses Lernfeldes an die Simulation solcher Prozesse herangeführt werden.

Lernfeld 12: Optimieren von Produktionsprozessen

Wie bereits in Lernfeld 9 und Lernfeld 11 treffen auch in Lernfeld 12 alle Zielformulierungen und Inhalte auf den Einsatz von Industrierobotern als Lernmedium in diesem Lernfeld zu.
In Lernfeld 12 gibt es zwei Betrachtungsweisen zum Einsatz von Industrierobotern als Lernmedium. Erstens können Industrieroboterprozesse selbst optimiert werden, beispielsweise durch Taktzeitminimierungen. Des Weiteren können Industrierobotersysteme als Optimierungsoption herangezogen werden, beispielsweise beim Austausch von manuellen Montageprozessen, aufgrund von ökonomischen oder ergonomischen Faktoren, gegen automatisierte mit dem Industrieroboter ausgeführte Montageprozesse. Der Einbezug von kollaborativen Industrierobotern bietet sich hier besonders an. Diese optimieren Handhabungs- und Fertigungsprozesse und werden zur Unterstützung von Maschinen- und Anlagenbedienerinnen und Anlagenbediener eingesetzt.

Elektroniker(in) für Automatisierungstechnik

Rahmenlehrplan für den Ausbildungsberuf Elektroniker für Automatisierungstechnik/ Elektronikerin für Automatisierungstechnik (Beschluss der Kultusministerkonferenz vom 16.05.2003 in der Fassung vom 23.02.2018)

Lernfeld 6: Anlagen analysieren und deren Sicherheit prüfen

> Die Schülerinnen und Schüler bereiten die Prüfung automatisierter Anlagen vor. Dazu analysieren sie Anlagen mit mechanischen, elektrischen, pneumatischen und hydraulischen Komponenten unter Nutzung von Plänen und Dokumentationen auch in audiovisueller und virtueller Form.
> Sie fassen die Anlagenkomponenten zu Funktionseinheiten zusammen, definieren Schnitt-stellen und stellen die Funktionsstruktur von Anlagen grafisch dar. Sie untersuchen arbeits-teilig Signal-, Energie- und Stoffflüsse von Funktionseinheiten sowie deren Komponenten und leiten daraus deren Funktion und deren Übertragungsverhalten ab.
> Die Schülerinnen und Schüler führen Funktionsprüfungen, Sichtprüfungen und Messungen an einzelnen Komponenten und den Anlagen durch, speziell unter den Aspekten Betriebssicherheit und Personenschutz. Sie eignen sich die Handhabung der notwendigen Mess- und Prüfgeräte an und nutzen deren Betriebsanleitungen, auch in englischer Sprache. Sie dokumentieren und präsentieren die Ergebnisse der Prüfungen, erstellen und ändern Pläne auch mit digitalen Medien.
> **Inhalte**: Sensoren, Aktoren; Schnittstellen; Betriebsarten; Redundanz und Diversität; Berührungslos wirkende Schutzeinrichtungen

Der kollaborative Roboter mit seinen vier Sicherheitsfunktionen (Handführung, Geschwindigkeits- und Abstandsüberwachung, sicherheitsgerichteter Stopp und die Kraft- Leistungsbegrenzung) kann hier als zu analysierendes Objekt dienen. Auch seine Funktionsweise kann von den Schülerinnen und Schülern untersucht werden. In diesem Zusammenhang können MRK-Messungen durchgeführt (falls das Equipment vorhanden ist), und/oder die Auswirkungen verschiedener Aktoren am Roboterarm auf deren sicherheitstechnischer Auswirkungen geprüft werden. Verschiedene Sensoren zur Verbesserung der Anwendersicherheit in ein bestehendes System lassen sich integrieren. Hierbei sollten immer die elektrotechnische Implementierung und die Funktionsweisen der verschiedenen Bauteile in den Fokus rücken.

Lernfeld 7: Steuerungen für Anlagen programmieren und realisieren

> Die Schülerinnen und Schüler entwerfen und erstellen normenkonform Steuerungsprogramme auch mit bibliotheksfähigen Funktionen und Funktionsbausteinen. Sie testen und dokumentieren diese.
> Die Schülerinnen und Schüler programmieren Verknüpfungssteuerungen, auch mit Zeit- und Zählfunktionen. Sie entwickeln, testen und dokumentieren lineare und verzweigte Ablaufsteuerungen mit unterschiedlichen Betriebsarten.
> Die Schülerinnen und Schüler programmieren mehrachsige Bewegungsabläufe oder verfahrenstechnische Abläufe.
> Inhalte: Digitale und analoge Signalverarbeitung; Strukturierte Programmierung; Programmiersprachen, auch grafische; Variablendeklaration, Instanzierung, symbolische Adressierung; Anlagensicherheit durch Hardware und Programmierung

Industrieroboter bieten sich hier als programmierbares Objekt an. Sie verfügen zum Teil über bibliotheksfähige Funktionen und Funktionsbausteine. Durch digitale und analoge Signalverarbeitung gibt es Verknüpfungsmöglichkeiten mit anderen Systemen. An ihnen können Variablendeklarationen und Instanzierungen eingeübt und mit realen Produktionsabläufen verknüpft werden. Den Kernbereich von Industrierobotern bilden mehrachsige Bewegungsabläufe, so dass auch hier eine Abbildung möglich ist.

Lernfeld 10: Automatisierungssystem in Betrieb nehmen und übergeben

> Die Schülerinnen und Schüler prüfen, justieren und stellen Sicherheitseinrichtungen ein. Sie beachten dabei die Betriebssicherheit sowie die Vorschriften des Gesundheits- und Arbeitsschutzes.

Zielformulierungen und Inhalte des Lernfeldes 10 gleichen sich hinsichtlich des Einsatzes von Industrierobotern als Lernmedium dem Lernfeld 6 -Installieren und Inbetriebnehmen steuerungstechnischer Systeme- der Industriemechanikerinnen und Industriemechaniker und dem Lernfeld 11 -Inbetriebnahme, Fehlersuche und Instandsetzung- der Mechatronikerinnen und Mechatroniker. Hinzu kommt ein größerer Fokus auf die Sicherheitseinrichtungen des Industrierobotersystems.

Lernfeld 11: Automatisierungssysteme in Stand halten und optimieren

Zielformulierungen und Inhalte entsprechen hinsichtlich der Instandhaltung dem Lernfeld 12 -Instandhalten von technischen Systemen- der Industriemechanikerinnen und Industriemechaniker. Daneben enthält das Lernfeld 11 der Elektronikerinnen und Elektroniker für Automatisierungstechnik Formulierungen darüber, die Instandhaltungsmaßnahmen zu optimieren und Selbstüberwachungen von Steuerungs- und Regelungsprozessen zu integrieren. Diese sind auch übertragbar auf Robotersysteme.

Lernfeld 12: Automatisierungssysteme planen

Zielformulierungen und Inhalte stimmen mit denen des Lernfeldes 6 der Mechatronikerinnen und Mechatroniker und des Lernfeldes 8 der Produktionstechnologinnen und Produktionstechnologen überein. Der Fokus liegt auf der Projektplanung und Entwicklung praktischer Lösungen.

Lernfeld 13: Automatisierungssysteme realisieren

Zielformulierungen und Inhalte gleichen denen der Lernfelder 7 und 13 der Mechatronikerinnen und Mechatroniker.

Berufliches Gymnasium Technik

Rahmenrichtlinien für das Fach Technik im Beruflichen Gymnasium – Technik – Stand: Mai 2008. Erweiterungen: August 2014 und 2016.

Einführungsphase:

In der Einführungsphase kann die Robotik im Lerngebiet **T1: Technische Informationen nutzen und erstellen**, wie in den oben beschriebenen Lernfeldern 1 – 4 der metalltechnischen Ausbildungsberufe behandelt werden.

Qualifikationsphasen: Informationstechnik, Elektrotechnik und Mechatronik

Diese drei Qualifikationsphasen beinhalten alle das Lerngebiet „Technische Prozesse steuern".
Lerngebiet IT1 (verbindlich): Technische Prozesse steuern
Lerngebiet METRO2 (verbindlich): Technische Prozesse steuern
Lerngebiet ET2 (verbindlich): Technische Prozesse steuern

> Die Schülerinnen und Schüler realisieren verbindungs- und speicherprogrammierte Steuerungen unter Berücksichtigung von Standardlösungen. Sie wählen Sensoren und Aktoren aus und binden diese ein.

In den Unterrichtshinweisen ist festgelegt, dass Auswahl und Anbindung von Sensoren und Aktoren nur an wenigen Sensoren und Aktoren exemplarisch durchgeführt werden soll. Hierfür bieten sich Robotersysteme als Lernmedium an, da hier eine (je nach Modell) einfache Integration von Aktoren und Sensoren möglich ist und nach der Integration die Funktionsfähigkeit sowie die Wirkweise an einem realen Modell getestet und überprüft werden kann.

Lerngebiet METRO4 (wahlweise): Handhabungssysteme programmieren und optimieren

> Die Schülerinnen und Schüler programmieren und optimieren Handhabungssysteme. Sie planen die Programmierung von Handhabungssystemen. Dabei stellen sie vollständige Prozesse oder Teilprozesse eines Handhabungssystems durch eine Ablaufbeschreibung dar.
> Sie programmieren und parametrieren Automatisierungskomponenten.
> Sie überprüfen ihre Arbeitsergebnisse und führen systematisch eine Fehlersuche und -korrektur durch.
> Sie beurteilen ihre Lösungen anhand ausgewählter Kriterien und optimieren das Handhabungssystem.

In diesem Lerngebiet werden Roboter explizit als mögliches Handhabungssystem genannt. Hier können alle genannten Inhalte an einem Robotersystem erarbeitet und durchgeführt werden. Ein Schwerpunkt liegt hierbei auf dem Handhabungsprozess.

Qualifikationsphase – Metalltechnik

Lerngebiet MT1 (verbindlich): Technische Produkte gestalten und dimensionieren

> Die Schülerinnen und Schüler wählen Lösungskonzepte für die Gestaltung technischer Produkte aus und begründen ihre Auswahl. Ausgehend vom ausgewählten Lösungskonzept entwickeln sie einen grobmaßstäblichen Entwurf.
> Die Schülerinnen und Schüler arbeiten die Feingestaltung technischer Produkte aus. Sie ermitteln grundlegende Belastungen und führen einfache statische Berechnungen durch. Unter Berücksichtigung von Werkstoffkennwerten dimensionieren sie Einzelteile. Für die notwendigen Berechnungen nutzen sie auch geeignete Software. Die Schülerinnen und Schüler erstellen Teilzeichnungen.

Der Industrieroboter kann hier, wie in den anfangs aufgeführten Lernfeldern 1 – 4 der metalltechnischen Ausbildungsberufe, als Lernmedium verwendet werden. So wird die Betrachtung von Prozessen möglich, die vom Industrieroboter ausgeführt werden sollen. Hierfür können einzelne Komponenten (bspw. Aktoren) entwickelt und durch die Auszubildenden erstellt werden.

Der Fokus liegt dabei auf handhabungstechnischen Prozessen sowie auf der kreativen Gestaltung von technischen (Teil-)Systemen auf der Grundlage technischer Problemstellungen. Der durch die Schülerinnen und Schüler gestaltete Lösungsansatz kann anschließend am realen System erprobt und evaluiert werden.

Literatur

Bieller, S.; Bill, M.; Kraus, W. und Müller, C. (2022): *Market presentation World Robotics 2022 extended version*. Frankfurt am Main, VDMA Services GmbH.

Deutsche Gesetzliche Unfallversicherung e. V. (DGUV) (2015): *DGUV Information 209-074 – Industrieroboter*. Berlin.

DIN EN ISO 10218-1:2012-01: *Industrieroboter – Sicherheitsanforderungen – Teil 1: Roboter (ISO 10218-1:2011)*.

DIN EN ISO 10218-2:2012-06: *Industrieroboter – Sicherheitsanforderungen – Teil 2: Robotersysteme und Integration (ISO 10218-2:2011)*.

DIN ISO/TS 15066:2017-04: *Roboter und Robotikgeräte – Kollaborierende Roboter (ISO/TS 15066:2016)*.

Kultusministerkonferenz (KMK): *Rahmenlehrpläne dualer Ausbildungsberufe*. Downloadbereich. https://www.kmk.org/themen/berufliche-schulen/duale-berufsausbildung/ downloadbereich-rahmenlehrplaene.html (01.05.2023)

Müller, C. (2022): *World Robotics 2022 – Industrial Robots. IFR Statistical Department*. Frankfurt am Main VDMA Services GmbH.

Schlausch, Reiner (2017): *Arbeiten und Lernen an und mit Robotertechnik*. In: lernen und lehren (2017): Schwerpunktthema Robotik. Heft 125 – 32. Jahrgang. Heckner.

2 Grundlagen Robotik

Max Niklas Bartholdt, Florian Oel, Levin Stanke, Leibniz Universität Hannover

Inhalt

2.1 Vorwort

In den folgenden Kapiteln werden Grundlagen und Begrifflichkeiten aus der Robotik erläutert. Ziel ist es hier den Leser mit allgemeinen Hintergrundwissen zum Thema Robotik zu stärken und am Ende das Zusammenspiel der einzelnen Komponenten konzeptionell zu verstehen. Einen Roboter in eine technische Applikation einzubinden erfordert Wissen über Hardware, Funktionsweise und Vorteile bestimmter Modelle. Beispielsweise besitzen die meisten Robotersysteme zwischen drei und sieben bewegliche Achsen. Aber was steckt eigentlich in einer Roboterachse? Und wie viele davon werden benötigt? Was ist der Vorteil einer siebenachsigen Kinematik gegenüber einer sechsachsigen? Was ist überhaupt eine Kinematik? Programmiersprachen, Software und auch Hardware variieren teils stark hinsichtlich der Applikation und der verwendeten Systeme. Ob es sich aber um ein System wie den Dobot Magician oder den Universal UR5 handelt, die Grundlagen der klassischen Robotik sind gesetzt.
Informationen auf einem zugänglichen, wenig mathematischen Niveau werden hier an den Leser herangetragen, um einen Überblick in das Standard-Vokabular der Robotik zu bekommen. Dies erfolgt ohne Anspruch an Vollständigkeit. Für ein tiefgehendes Verständnis sei auf die etablierte und hier teils aufgefasste Fachliteratur verwiesen. Dazu ist am Ende jedes Abschnitts die verwendete Literatur erfasst.

2.2 Roboter: Aufbau und Definitionen

Roboter sind komplexe mechatronische Systeme. Der Begriff Mechatronik setzt sich zusammen aus Mechanik, Elektrotechnik und Informationstechnik. Die Robotik ist folglich ein interdisziplinäres Anwendungs- und Forschungsfeld. Ein Roboter interagiert mit seinem Umfeld mithilfe von Aktoren, Sensoren und einem oder mehreren Prozessrechnern. Häufig werden Elektromotoren genutzt um die Mechanik des Systems in Bewegung zu versetzen, während Kameras, Positions- und Kraftsensoren die Zustände des Roboters und der Umwelt erfasst. Auf Basis dieser Informationen wird auf dem Prozessrechner entschieden, wie die Motoren anzusteuern sind, um ein gewünschtes Verhalten zu erzielen.
Prinzipiell lassen sich Roboter unterschiedlich kategorisieren. Eine mögliche Unterscheidung unterteilt die Systeme in

- Industrieroboter,
- Serviceroboter sowie
- Entertainment - & Edutainmentroboter.

Abbildung 1: Bilder verschiedener Robotersysteme [#Q1]

Während Industrieroboter seit den späten 1980er in Produktionshallen in Fertigung- und Montage genutzt werden, haben Serviceroboter sich bisher in weniger Domänen durchsetzen können. Populäre Beispiele für Service-Roboter sind die häufig in Privathaushalten genutzten Staubsaugerroboter und autonome Rasenmäher. Die Systeme sind hohen Anforderungen an der Autonomie und Sicherheit in weitestgehend unbekannten sowie unstrukturierten Umgebungen ausgesetzt. Der Erfolg der genannten Beispiele lässt sich demnach sicherlich auch auf die relativ strukturierten Eigenschaften eines Wohnzimmers zurückführen. Dementgegen stellt bspw. der Einsatz von Rettungsrobotern in einem Lawinengebiet (wenig Kontraste, schwierige Lichtverhältnisse, keine klaren Begrenzungen, unebener Boden) eine enorme Herausforderung für Forscher und Entwickler dar. Der Dobot Magician, als didaktische, unterhaltende Plattform ist ein gutes Beispiel für einen Edutainmentroboter. Einige Beispiele für Roboter aus dem Service-, Industrie- sowie Forschungsbereich sind in Abbildung 1 zu finden.

Schwerpunkt dieses Kompendiums sind (seriell-kinematische) Industrieroboter und Cobots (kollaborative Roboter). Cobots unterscheiden sich von klassischen Industrierobotern im Wesentlichen durch eine stark reduzierte Masse und entsprechende Sicherheitsanforderungen. Während der klassische Industrieroboter hinter Sicherheitszäunen Fertigungsprozesse durchführt, steht der Cobot im direkten Kontakt mit dem Menschen.
Der prinzipielle Aufbau beider Roboterarten ähnelt sich aber stark. Deswegen werden diese Systeme in den folgenden Abschnitten definiert und ein Grundverständnis über ihren Aufbau vermittelt. Dabei steht kein Produkt im Vordergrund, sondern vielmehr die gemeinsame Grundlage aller seriell-kinematischen Roboter.

Definition Industrieroboter nach VDI-Richtlinie 2860: „Industrieroboter sind universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegungen hinsichtlich Bewegungsfolge und Wegen bzw. Winkeln frei (d. h. ohne mechanischen bzw. menschlichen Eingriff) programmierbar und gegebenenfalls sensorgeführt sind. Sie sind mit Greifern, Werkzeugen oder anderen Fertigungsmitteln ausrüstbar und können Handhabungs- und/oder Fertigungsaufgaben ausführen.“ Ähnliche Definitionen lassen sich in der DIN EN ISO 8373 wiederfinden.

Die am häufigsten verwendeten Industrieroboter bestehen aus einer serielle Anordnung der Robotergelenke und –glieder. Dies wird als serielle Kinematik bezeichnet. Die hohe Anwendungsvielfalt der seriellen Kinematik liegt vor allem am guten Verhältnis aus Bau- zu Arbeitsraum des Roboters. Damit gemeint ist der benötigte Raum oder die Hallenfläche, die der Aufbau des Systems insgesamt benötigt im Vergleich zu dem Raum in dem der Roboter Aufgaben bearbeiten kann.
Ein Nachteil der seriellen Struktur ist die Fehlerfortpflanzung bei Positionierungenauigkeiten. Eine ungenaue Positionierung im ersten Gelenk wirkt sich beispielsweise auf alle folgende Glieder und Gelenke aus. Dies führt zu einer reduzierten Genauigkeit an der Spitze des Roboters (Endeffektor).

Dahingegen besitzen parallelkinematische Roboter mehrere serielle Strukturen die an einer Endeffektor-Plattform gekoppelt sind. Dadurch können diese Systeme höhere Lasten tragen, schneller Beschleunigen erzielen und sind genauer. Typische Anwendungsfälle sind hier zeitkritische „Pick-and-Place“-Aufgaben und Bearbeitungsprozesse mit erhöhten Prozesskräften (Fräsen, Bohren). Dafür sind parallelkinematische Roboter sperriger und besitzen demnach ein schlechteres Verhältnis aus Arbeitsraum zu Bauraum.

2.2.1 Aufbau eines seriell-kinematischen Roboters

Ein serieller Roboter besteht aus einer Kette von Gliedern und Gelenken. Jedes Glied ist über ein Robotergelenk mit dem nächsten Glied verbunden. Im Gelenk ist ein elektrischer Motor inkl. Getriebe und Sensorik verbaut. Dieser Aufbau ist schematisch für ein Glied und ein Gelenk in Abbildung 4 gezeigt. Die folgenden Abschnitte behandeln die Funktionsweise der Teilkomponenten dieses Aufbaus.

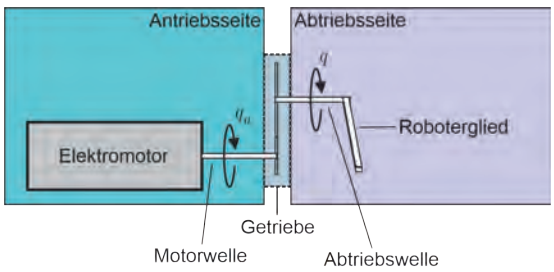


Abbildung 2: Aufbau eines seriell-kinematischen Roboters

2.2.2 Elektromotoren

Um die Robotergelenke bewegen zu können, werden meist Elektromotoren als Antriebe gewählt. Dabei gibt es verschiedene Ausführung mit unterschiedlichen Vor- und Nachteilen.
Schrittmotoren *(engl. stepper motor)*: Bei diesen Motoren kann ohne Sensorik eine schrittweise, diskrete Positionsänderung vorgeben werden, da die Steuerungssignale an feste Positionen des Rotors gekoppelt sind. Sie werden meist für kleine und einfach Roboter genutzt, da es bei zu großen externen Lasten zum Schrittverlust kommt. Der Motor wird dabei auf eine bestimmte Position gesteuert, aber durch die hohe Last, die das interne Magnetfeld des Motors überwindet, erreicht der Motor eine andere, unbekannte Position. Dies kann nur durch zusätzliche Sensorik erkannt werden und stellt den wesentlichen Nachteil dieses Motors da.

Permanent-Magnet Gleichstrommotor: Die rotierende Bewegung eines Permanent-Magnet Gleichstrommotors beruht auf der Wechselwirkung von magnetischen Feldern. Er besteht wie in Abbildung 2 skizziert aus einem feststehende Außenteil, dem sogenannten Stator und einem rotierenden Innenteil, dem Rotor. In dieser skizzierten Ausführung ist der Stator ein Permanentmagnet mit Nord- und Südpol und erzeugt ein statisches Magnetfeld. Der Rotor hingegen ist ein Eisenkern mit elektrischen Spulen, die mit Schleifringkontakten auf der Welle verbunden sind. Fließt über die Kontakte durch die Spulen ein elektrischer Strom, entsteht im Eisenkern ein Magnetfeld. Dadurch dreht sich der gesamte Rotor solange, bis das Magnetfeld im Eisenkern sich vollständig am Stator orientiert hat (Nordpol zu Südpol, Südpol zu Nordpol). Bei diesem Vorgang werden über die Schleifkontakte auf der Welle (Kommutator und Kommutatorbürsten) die Spulen im Rotor umgepolt, sodass der Strom in die entgegengesetzte Richtung fließt. Dadurch setzt der Rotor seine Bewegung kontinuierlich fort.

Bürstenlose Gleichstrommotoren *(engl. BLDC, brushless DC motors)* sind die am häufigsten eingesetzten Motoren in der Robotik. Beim Brushless Motor werden Bürsten und Kommutator ersetzt. Sie haben dementsprechend eine deutlich erhöhte Lebensdauer, da die Bürsten und der Kommutator beim Permanent-Magnet DC Motor durch ständiges Schleifen abnutzen. Allerdings benötigen BLDCs eine komplexe Steuerelektronik auf die an dieser Stelle nicht umfassend eingegangen wird.
BLDCs werden meist in Form von Servomotoren in der Robotik eingesetzt. Aufgrund ihres Aufbaus lassen sich Motorposition und Drehmoment gut regeln. Im Gegensatz zu Schrittmotoren liefern sie ein relativ konstantes Drehmoment und weisen ebenfalls nicht den Nachteil des Schrittverlustes auf.

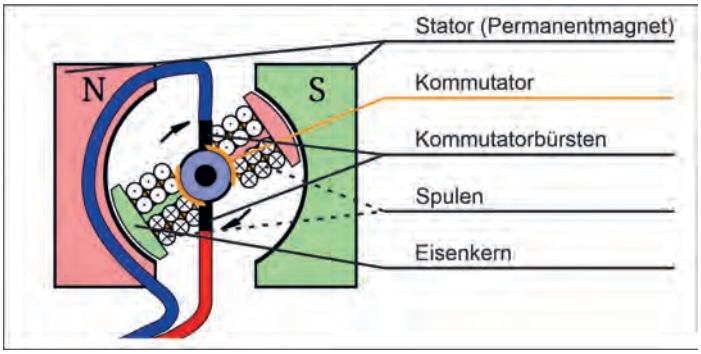


Abbildung 3: Magnetfeldpolung des Rotors in einem Gleichstrommotor mit Permanentmagnetstator (bearbeitet) [#Q2]

2.2.3 Getriebe im Roboter

Die Motorwelle des Elektromotors ist direkt an ein Getriebe gekoppelt. An der Abtriebsseite des Getriebes stellt sich eine verringerte Drehzahl und ein erhöhtes Drehmoment ein. Gängige Getriebeübersetzungen in der Robotik liegen bei 1:100, um die hohen Umdrehungszahlen des Elektromotors zu reduzieren und das Drehmoment an Abtriebswelle deutlich zu erhöhen. Die Übersetzung i des Elektromotors ist definiert als das Verhältnis aus der Drehzahl am Antrieb und der Drehzahl am Abtrieb

$$i = \frac{n_{\text{Antrieb}}}{n_{\text{Abtrieb}}}.$$

Harmonic Drive Getriebe: Eine weitverbreitete Getriebeart in der Robotik ist das in Abbildung 4 dargestellte Harmonic Drive Getriebe. Diese Getriebeart ist eine Schlüsseltechnologie für die kompakte Bauweise von Robotergelenken und Reduktion der Masse in Cobots.

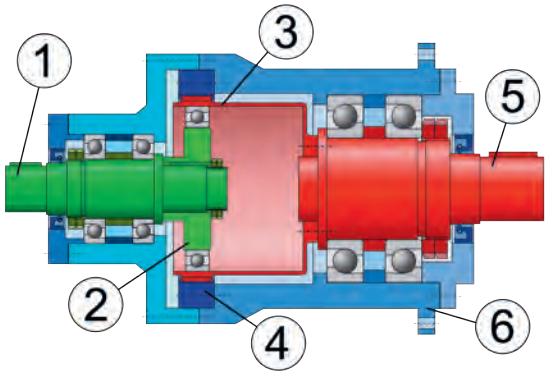


Abbildung 4: Harmonic-Drive Getriebe, Schnitt-/ Seitenansicht [#Q3]

Über den angesteuerten Elektromotor ergibt sich die Eingangsdrehzahl und das Eingangsdrehmoment. Dieses Moment wird auf die elliptische Scheibe (2) übertragen, welche die dünnwandige Stahlbüchse (3) während der Rotation deformiert. Bei der Rotation der elliptischen Scheibe greift die Außenverzahnung der Stahlbüchse somit in die Innenverzahnung des Außenrings (4). Der Außenring ist fixiert, sodass bei einer Umdrehung der Antriebsscheibe (2) die Stahlbüchse (und somit auch der Abtrieb (5)) nur eine kleine Teildrehung durchführt. Das heißt, der Abtrieb dreht sich wesentlich langsamer und entgegengesetzt zum Antrieb. Das Harmonic Drive ist bekannt für die große Übersetzung von 30:1 bis 320:1.

2.2.4 Typische Sensorik für Industrieroboter

Sensoren wandeln eine physikalische Größe (wie zum Beispiel Temperatur, Druck, Feuchtigkeit etc.) in ein elektrisches Signal um, das anschließend messtechnisch weiterverarbeitet wird. Sie sind notwendig, damit im Prozessrechner der Zustand des Roboters, zum Beispiel seine Konfiguration im Raum, korrekt erfasst werden kann. Zu den essenziellen Sensoren in einem seriellen Roboter gehören **Positionssensoren** um die Lage der einzelnen Robotergelenke zu messen, sowie **Kraft-Momenten-Sensoren** um Kontakte mit der Umgebung korrekt zu erfassen. Auch Kameras und Mikrofone sind essenzieller Bestandteil von Robotersystemen, um sich in ihrer Umgebung korrekt zu orientieren und Gegenstände sowie Personen zu erkennen.

2.2.4.1 Lagesensoren

Lagesensoren erfassen durch ein optisches oder magnetisches Prinzip den Rotationswinkel/die Position des Elektromotors. Bei Lagesensoren wird zwischen differenziellen (Inkrementalgeber) und absoluten Sensoren (Resolver) unterschieden. Differentielle Sensoren messen eine schrittweise Änderung der Position. Wird allerdings die Lage benötigt muss zu Beginn der Roboter in eine Referenzposition fahren, um die initiale absolute Position vor Betrieb zu kennen. Absolute Sensoren erfassen zu jedem Zeitpunkt eine eindeutige Position und benötigen deshalb kein Referenzieren bei der Inbetriebnahme des Systems.

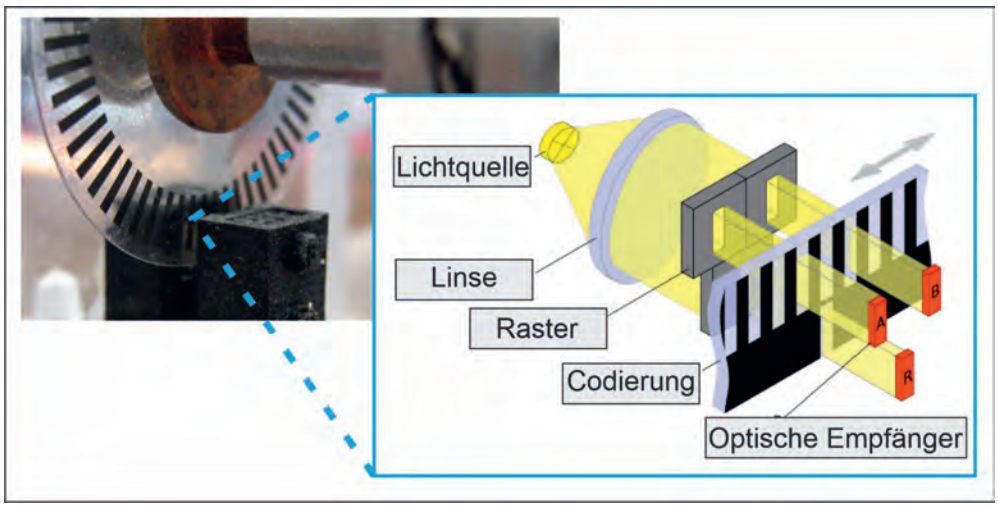


Abbildung 5: Photoelektrische Abtastung (Demonstrationsmodell) [#Q4]

Den inkrementellen Drehgeber (Encoder) gibt es mit magnetischer und **optischer Abtastung**. Bei der optischen Abtastung des inkrementellen Sensors ist eine Codescheibe (Strichcode), wie in Abbildung 5 drehbar gelagert. Eine LED beleuchtet die Codeschreibe, welche strichförmige Löcher aufweist. Ein dahinter gelagerter Empfänger (optoelektronisches Element) wandelt das empfangene Licht in ein elektrisches Signal um. Bei einer Drehung der Scheibe entsteht ein periodisches, binäres helldunkles Signal. Aus diesem kann die Drehrichtung als auch die Geschwindigkeit bestimmt werden.

Bei der **magnetischen Abtastung** ist jede Winkelposition durch einen Feldvektor fest definiert. Der Encoder besitzt einen rotierenden Permanentmagneten der das benötigte magnetische Feld aufbaut. Die Feldvektoren werden von einem Sensorelement in inkrementelle Signale transformiert.

Die **Messung einer absoluten Position** erfolgt durch das einbetten einer eindeutigen Codierung auf der Drehscheibe zwischen Lichtquelle und Optoelektronik. Jede Position der Drehscheibe erzeugt so ein einzigartiges binäres Wort (bspw. 0100 für 4 Sensorsignale: dunkel, hell, dunkel, dunkel). Die Auflösung ist dann bestimmt durch die Wortlänge, also die Anzahl der Bits. Eine 4-Bit-Auflösung ermöglich das Erfassen von 2^4=16 unterschiedlichen Positionen über eine ganze Rotation.

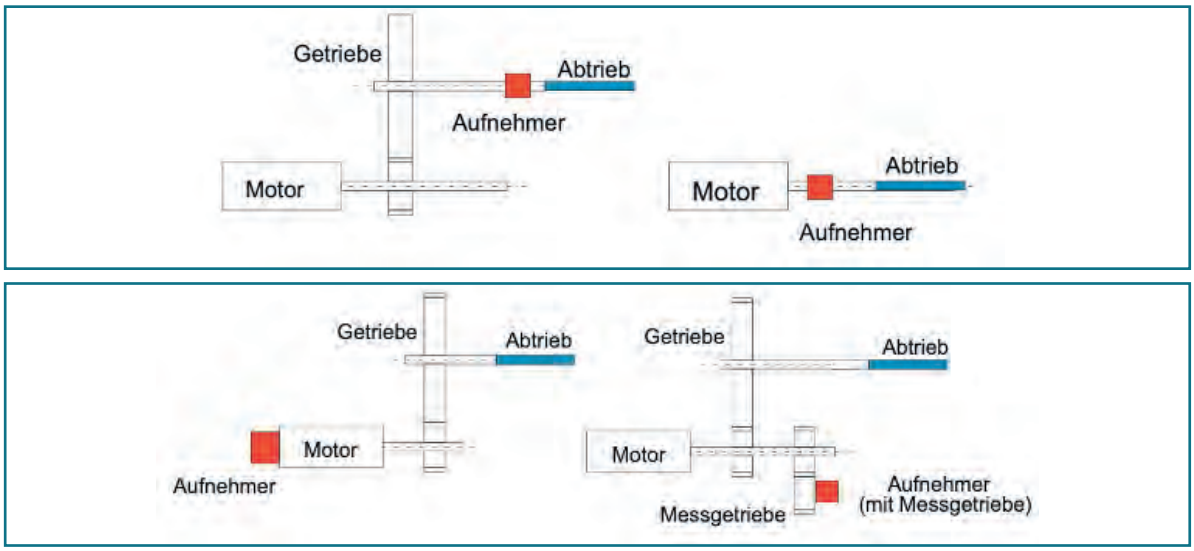


Abbildung 6: Verschiedene Konfigurationen für Lagesensoren

Auch in der Montage des Sensors gibt es einige Unterschiede die in Abbildung 6 dargestellt sind. Bei der direkten Messung wird der Lagesensor des Roboters an der Abtriebsseite des Getriebes angebracht. Durch die kleinen Bewegungen an der Abtriebsseite des Antriebsstranges werden teurere und hochauflösende Messeinheiten benötigt. Deutlich günstiger sind dagegen indirekte Messungen an der Motorseite des Antriebsstranges oder über ein weiteres Getriebe, dass mit der Motorwelle verbunden ist. Hier können günstigere Sensoren verbaut werden, da die Drehzahl sehr viel höher als abtriebseitig ist, aber das Getriebespiel wird so nicht erfasst. Dadurch leidet bei einer Regelung im Gelenkraum die Präzision am End-Effektor.

2.2.4.2 Kraft-Momenten-Sensoren

Neben Lagesensoren sind häufig auch Kraft-Momenten Sensoren (KMS) verbaut. Dies ist vor allem bei Aufgaben mit menschlichem Kontakt notwendig, da über das Erfassen und Lokalisieren von Kräften am Roboter das System eine sichere Reaktionsstrategie ausführen kann. Als primitive Reaktion wäre beispielsweise ein Stop des Systems über die Bremsen denkbar.
KMS messen mechanische Kräfte und Momente indirekt über die Deformation eines flexiblen Körpers, der mit Dehnungsmessstreifen (DMS) versehen ist. DMS sind flache, elektrische Leiterstreifen, die mit einem speziellen Kleber auf deformierbaren Trägerelementen befestigt werden. Bereits bei kleinsten Verformungen werden die Leiter im DMS gestaucht oder gestreckt, wodurch sich der elektrische Widerstand des DMS merklich ändert. Diese Widerstandsänderung wird messelektronisch erfasst und ausgewertet, um auf Kräfte bzw. Momente zurück zu schließen. Eine exemplarische Umsetzung ist in Abbildung 6 gezeigt.

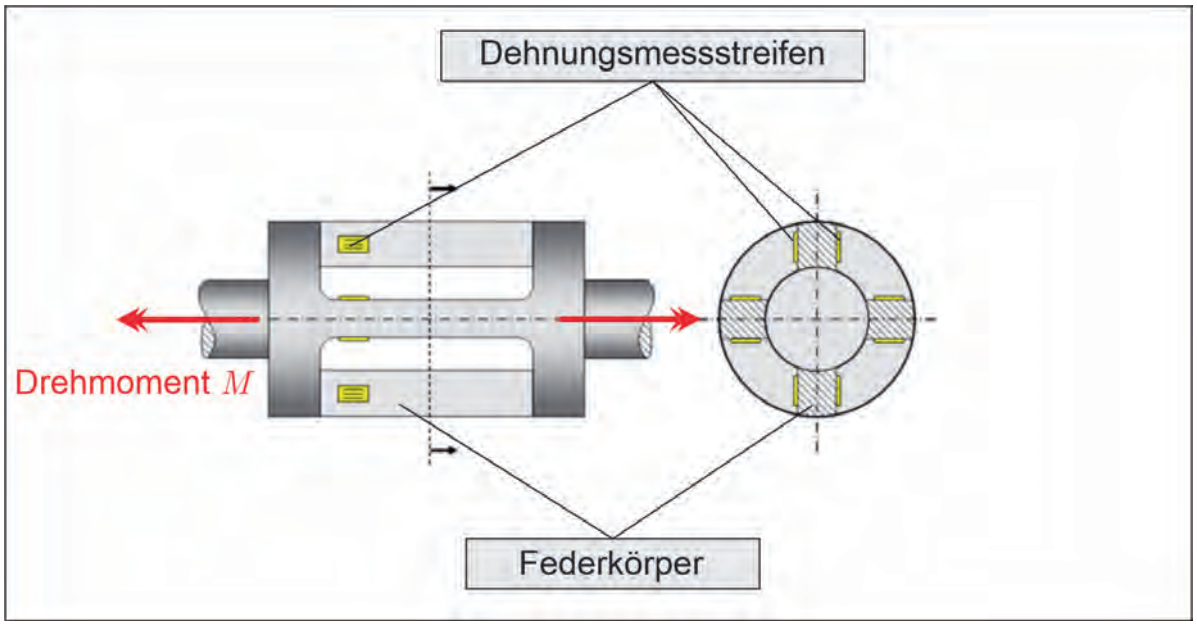


Abbildung 7: Drehmomentenaufnehmer [#Q5]

Um die Sensitivität in den Gelenken des Roboters gegenüber Krafteinwirkungen zu erhöhen, können Drehmomenten-Sensoren am Abtrieb des Getriebes befestigt werden. Teilweise werden Kraft-Momenten-Sensoren auch nahe des End-Effektors montiert, um besonders feinfühlig bei Handhabungs-Arbeiten zu sein.

Eine Alternative zu relativ teuren Kraft-Momenten-Sensoren ist die deutlich günstigere Motorstrom-Messung. Dabei wird die mechanische Kraft nicht direkt gemessen, sondern lediglich über eine Kennlinie aus dem fließenden Motorstrom bestimmt. Eine Voraussetzung ist, dass dieser Zusammenhang zwischen Motorstrom und Drehmoment bekannt ist, bzw. dass eine Kennlinie des Motors vorliegt. Nachteil dieser Methode liegt dabei darin, dass externe Kräfte aufgrund des Messrauschens und der Getriebereibung nicht erkannt werden. Der Roboter erreicht im dem Sinne nicht die gleiche „Feinfühligkeit".

2.2.5 Prozessrechner und Steuerung

Alle Daten werden in der Robotersteuerung verarbeitet. Ein zentraler Hauptrechner im Schaltschrank übernimmt die steuernde Funktion. Für sicherheitsrelevante Tätigkeiten ist ein zweiter Rechner in der Steuereinheit verbaut. Er hat die Aufgabe, dass zum Beispiel bei einem Nothalt alle Bremsen des Roboters bis zu einem gewissen Zeitpunkt nach der Betätigung greifen und den Roboter zum Stillstand bringen. Eine weitere Funktion des Sicherheitsrechners ist auch die Steuerung von Sicherheitskomponenten, welche an die Sicherheitsschnittstelle angeschlossen werden können.

2.2.6 End-Effektoren und Anwendungsgebiete

Je nach Anwendungsgebiet können an dem Arm eines Roboters unterschiedliche Werkzeuge angebracht werden. Bei Pick-and-Place-Aufgaben werden Greifsysteme eingesetzt, welche den Roboter in die Lage versetzt Objekte im Raum zu bewegen. Verschiedene Ausführungen sind je nach Anwendung denkbar und beispielhaft in Abbildung 8 dargestellt. Für empfindlichen Bauteilen kann ein pneumatischer Sauggreifer zum Einsatz kommen, der das Bauteil durch Unterdruck am Greifer fixiert. Bei ferromagnetischen Materialien wie Stahlblechen sind magnetische Greifer sinnvoll. Objekte ohne glatte Oberflächen lassen sich am besten mit Drei-Finger-Greifern oder speziell konstruierten Greifer-Systemen manipulieren. Im Rapid-Prototyping kommt hierbei der 3D-Drucker oft zum Einsatz, um Konzepte für neuartige Greifmechanismen zu testen.

Abbildung 8: Verschiedene Greifer-Systeme [#Q6]

Neben Pick-and-Place oder Montage-Aufgaben mit Greifern, können Roboter noch eine Vielfalt von anderen Tätigkeiten ausführen. Dazu gehören typischerweise

- Schweißen,
- Bohren,
- Verschrauben,
- Messen,
- und Kleben.

Lerninhalte

- Serielle Kinematik in der Robotik
- Aufbau eines Robotergliedes
- Aktoren (Elektromotoren & Getriebe)
- Sensoren (Lage- und Kraftmessung)
- Robotersteuerung (Hardware)
- End-Effektor und Werkzeugarten zur Ausführung einer Tätigkeit

Fragen

- Zeichnen Sie eine serielle Kinematik mit drei Roboterachsen auf.
- Der in der ersten Roboterachse verwendete Elektromotor hat eine Umdrehungszahl von 1000 Umdrehungen pro Minute. Das Harmonic Drive Getriebe eine Übersetzung von 225. Mit welcher Geschwindigkeit (in Umdrehungen pro Minute) dreht sich der Roboterarm?
- Erklären Sie in eigenen Worten die Funktionsweise eines optischen Encoders.

2.3 Freiheitsgrade

Dieses Kapitel bildet die Grundlage für die folgenden Kapitel „Arbeitsraum“ und „Kinematik“. Es wird der Begriff von Freiheitsgraden im Kontext von räumlichen Körpern und kinematischen Ketten erläutert.

2.3.1 Räumliche Freiheitsgrade

Die Anzahl der Freiheitsgrade eines Körpers im Raum bezeichnet die Anzahl der Variablen, die notwendig sind, um die Lage (Position und Orientierung) des Körpers gegenüber einem Bezugssystem beschreiben zu können.

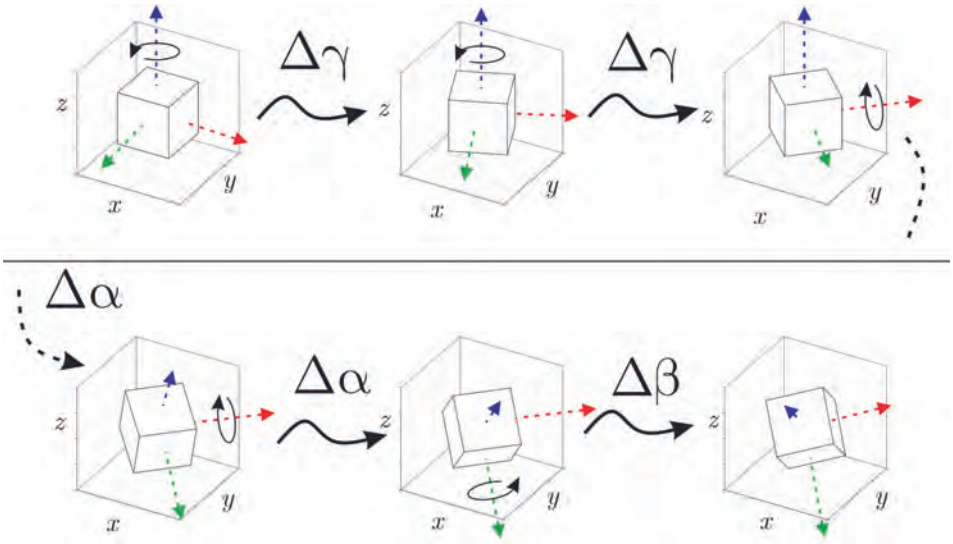


Abbildung 9: Würfel mit mehreren Rotationen

Die Position eines Körpers ist durch drei Komponenten aus unabhängigen Freiheitsgraden festgelegt. Die Position des Würfels in Abbildung 9 ist bspw. durch eine x-, y- und z-Komponente beschrieben und in allen sechs Abbildungen gleich. Um seine Ausrichtung im Raum festzulegen, benötigen wir weitere drei Werte. Daher hat der Würfel insgesamt sechs Freiheitsgrade im Raum. Diese sechs Freiheitsgrade setzen sich aus drei translatorischen (x, y, z) und drei rotatorischen (α, β, γ) Freiheitsgraden zusammen. In der Abbildung wird der Würfel stets um seine eigenen Achsen gedreht, und nicht um die x-, y- und z-Achse des Bezugssystems.
Es gibt also auch unterschiedliche Möglichkeiten die Rotationen eines Körpers zu beschreiben (Drehung um körperfeste Achsen oder raumfeste Achsen). In einigen Bedienungsfeldern von Robotersystemen kann das vom Anwender vorgegeben werden, da je nach Anwendungsfall die eine oder andere Beschreibung von Vorteil ist. Befindet sich der Endeffektor meines Roboters schon sehr nahe seiner Ziel-Lage und nur kleine Korrekturen der Ausrichtung sind gewünscht, ist es intuitiver um die Achsen des Greifers zu drehen, als um die räumlichen Achsen. Plane ich hingegen eine Aufgabe programmatisch ist es einfach die Orientierungen des Endeffektors bezüglich des raumfesten Koordinatensystems vorzugeben.

2.3.2 Freiheitsgrade kinematischer Ketten bei Robotern

Ein Industrieroboter weist in den meisten Fällen sechs Freiheitsgrade auf. Jeder Freiheitsgrad wird durch eine Roboterachse festgelegt. „[Roboter-]Achsen sind geführte, unabhängig voneinander angetriebene Glieder. Entsprechend der Führung unterscheidet man rotatorische Achsen (Drehachsen) und translatorische Achsen [(auch Linearachsen)]“. Eine Roboterachse hat nach dieser Definition dann genau einen (rotatorischen) Freiheitsgrad.

Die Anzahl der unabhängigen Freiheitsgerade am Endeffektor des Roboters wird durch die Summe der Freiheitsgerade aller Roboterachsen in der seriell-verketteten Struktur bestimmt. Ein Roboter mit drei unabhängigen rotatorischen Achsen kann also nur drei Freiheitsgrade am Endeffektor unabhängig voneinander bewegen.

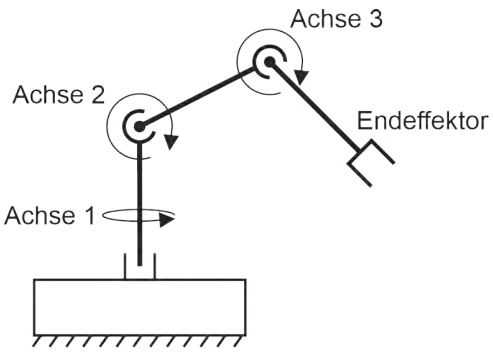


Abbildung 10: Schaubild 3-Achs-Kinematik

Beispiel: Der skizzierte Roboter in Abbildung 10 kann seinen Endeffektor also in drei räumlichen Freiheitsgeraden unabhängig positionieren. Beispielsweise kann die Position im Raum vorgegeben werden, sodass eine eindeutige Konfiguration der Robotergelenke dazu existiert. Voraussetzung dafür ist, dass sich die Position im Arbeitsraum des Roboters befindet.

Lerninhalte

- Definition von räumlichen Freiheitsgraden
- Definition von Freiheitsgraden einer kinematischen Kette

Fragen

- Welche sechs Freiheitsgrade gibt es im Raum?
- Wie berechnet man die Anzahl der Freiheitsgrade eines Roboters?

2.4 Arbeitsraum

Der Arbeitsraum eines Roboters bezieht sich auf den physischen und/oder virtuellen Bereich, in dem der Roboter sich bewegen und seine Aufgaben ausführen kann. Er wird durch die kinematische Struktur des Roboters und den Bewegungsbereich seiner Gelenke und des Endeffektors definiert. Der Arbeitsraum kann durch seine Abmessungen, Form und Zugänglichkeit charakterisiert werden und ist ein wichtiger Aspekt bei der Gestaltung und dem Betrieb eines Robotersystems. Unter Umständen kann auch die Anwendung den Arbeitsraum des Roboters eingrenzen, wenn der Roboter beispielsweise nahe einer Wand steht.

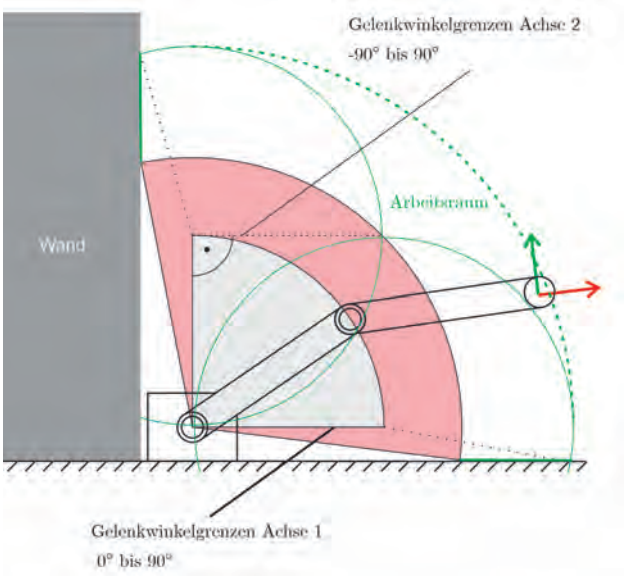


Abbildung 11: Arbeitsraum eines Roboters

Als Lage des Endeffektors ist dessen Position und Orientierung im Raum zu verstehen. Für viele Anwendungen (zum Beispiel beim Bohren) ist die Orientierung des Endeffektors mit vorzugeben. Allerdings ist die Rotation um die Längsachse des Bohrers nicht entscheidend beim Bohrvorgang und kann somit frei gewählt werden.

Neben konstruktionsbedingten Einschränkungen des Arbeitsraumes gibt es noch die sogenannten singulären Gelenkstellungen (auch Konfigurationen) (siehe Abschnitt „Singularitäten“), in denen der Roboter nicht agieren kann.

In Abbildung 11 ist ein planarer Roboter mit zwei Achsen skizziert. Der Roboter steht auf dem Boden neben einer Wand. Achse 1 kann von 0° bis 90° verfahren werden (grauer Bereich). Achse 2 kann von -90° bis 90° Verfahren werden. Dies schränkt den Arbeitsraum durch den minimalen Abstand des Endeffektors zur Basis des Roboters wie skizziert (angedeutete -90° Position für zweite Achse) weiter ein (rötlicher Bereich). Die maximale Reichweite des Roboters ist durch die gestrichelte grüne Linie dargestellt. Diese ist nicht durchgängig bis zur Wand und zum Boden, da vorher die Achslimitierung der ersten Achse erreicht wird. Dazu sind die zwei grünen durchgezogenen Kreise eingezeichnet. Der eingeschlossene grüne Bereich ist der resultierende Arbeitsraum. Für räumliche Systeme mit vielen Freiheitsgraden ist eine vergleichbare Skizze nur rechnergestützt möglich.

Lerninhalte

- Erklärung des Begriffes Arbeitsraum
- Verständnis für die Begriffe Position und Orientierung

Fragen

- Erklären Sie in Ihren eigenen Worten den Arbeitsraum eines Roboters.
- Mit welchen 6 Parametern kann ein Objekt im Raum beschrieben werden?

2.5 Kinematik

In der Kinematik, bezogen auf die Robotik, geht es um geometrische Zusammenhänge zwischen Positionen, Geschwindigkeiten und Beschleunigungen von starren oder flexiblen Körpern. Im speziellen Fall der seriellen Kinematik geht es um verketten Strukturen (Gelenken und Gliedern) und dem Zusammenhang der Gelenkwinkel und der Endeffektor-Lage. Hierbei spielen Kräfte und Momente keine Rolle. Die differenzielle Kinematik schließt für die gleichen Zusammenhänge dann Geschwindigkeiten und Beschleunigungen ein.

2.5.1 Direkte Kinematik

In der Robotik beschreibt das direkte kinematische Problem (DKP) die Position und Orientierung des Endeffektors in Bezug auf die Roboterbasis, die von einer gegebenen Gelenkwinkelkonfiguration eingenommen wird. In seriellen Anordnungen von Robotergelenken kann die DKP durch eine Matrizenmultiplikation der Denavit-Hartenberg-Matrizen berechnet werden. Diese Berechnung kann analytisch und in Echtzeit auf Maschinensteuerungen erfolgen. Da diese Berechnungen komplex sind, wird auf eine detaillierte Erklärung an dieser Stelle verzichtet.

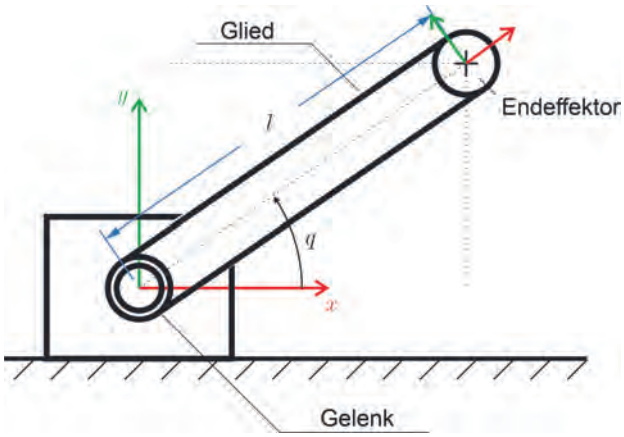


Abbildung 12: Vereinfachte Darstellung eines Robotergelenks

Beispiel: Ein zweidimensionales Beispiel, dargestellt in Abbildung 12 wird verwendet, um die Berechnung anschaulich zu erklären. Durch Anwendung des Sinus oder Cosinus und einer Umstellung zur unbekannten Variable kann aus dem Rotationswinkel des Gelenks die Endeffektor-Koordinate berechnet werden. Die Länge des Robotergliedes l und die Winkelstellung des Robotergelenkes q müssen bekannt sein, um die Endeffektor-Koordinaten (X-, Y-Koordinate) zu berechnen. Mithilfe trigonometrischer Funktionen kann die Endeffektor-Position bestimmt werden. Zunächst sei hier der Zusammenhang über den Sinus gegeben als

$$\sin(q) = \frac{\text{Gegenkathete}}{\text{Hypothenuse}} = \frac{Y - \text{Koordinate des Endeffektors}}{\text{Länge des Robotergliedes}} = \frac{y}{l}.$$

Daraus folgt für die Y-Koordinate des Endeffektors

$$\Rightarrow y = l \cdot \sin(q).$$

Analog folgt über den Cosinus der Wert für die X-Koordinate

$$x = l \cdot \cos(q).$$

Die Orientierung in diesem einfachen Fall entspricht dem Winkel q. Damit ergibt sich die vollständige Kinematik als

$$\boldsymbol{x}_{\mathrm{EE}} = \boldsymbol{f}(q) = \begin{pmatrix} x \\ y \\ \alpha \end{pmatrix} = \begin{pmatrix} l \cdot \cos(q) \\ l \cdot \sin(q) \\ q \end{pmatrix}$$

Dieses zweidimensionale Beispiel soll die Berechnung des direkten kinematischen Problems veranschaulichen. In der Realität sind die kinematischen Zusammenhänge wesentlich komplexer und erfordern die Verwendung von Rotationsmatrizen und Vektoren sowie die Berücksichtigung von mehreren Gelenken und deren Winkelstellungen. Gut zu erkennen ist in diesem Beispiel auch der Arbeitsraum des Roboters. Dieser liegt auf einem Kreis mit dem Radius l. Dabei ist das Bezugssystem am Endeffektor immer um den Winkel α=q gedreht.

2.5.2 Inverse Kinematik

Das inverse kinematische Problem (auch IKP genannt) ist die Frage nach der Gelenkkonfiguration zu einer vorgegebenen Lage des Endeffektors. Gebraucht wird diese bspw. wenn eine Bahn des Endeffektors in der Anwendung in räumlichen Koordinaten geplant wird. Dies ist in der Realität meistens deutlich intuitiver für den Nutzer. Für die Steuerung der Gelenke muss allerdings die für eine vorgegebene Lage notwendige Gelenkwinkel-Konfiguration bestimmt werden.

Diese Berechnung ist deutlich schwieriger als das DKP und ein Grund dafür, warum sich bestimmte kinematische Anordnungen immer wieder bei verschiedenen Robotermodellen wiederholen. Denn für diese Anordnungen lässt sich das IKP besonders gut lösen.

Die inverse Kinematik berechnet die Gelenkstellungen eines Roboters, wenn die Position des Endeffektors bekannt oder vorgeben ist. Es gibt jedoch eine Ausnahme, die sogenannten singulären Stellungen, bei denen die Berechnung der Gelenkstellungen nicht möglich ist. Es gibt keinen allgemeinen Formalismus zur Bestimmung einer Lösung für die inverse Kinematik. Die Lösung muss stattdessen spezifisch für die Geometrie des Roboters gefunden werden, was die Aufstellung der inversen Kinematik komplizierter machen kann.

Bei dem vorangegangenen, einfachen Beispiel mit einer Achse wird zunächst klar, dass nicht für jede beliebige Lage (x, y, α) ein Winkel q existiert. Die vorgegebene Lage muss im Arbeitsraum des Roboters liegen, sonst existiert keine Lösung.

Lerninhalte

- Definition des Begriffes Kinematik
- Unterschied zwischen offener und geschlossener kinematischer Kette
- Verständnis der Unterschiede zwischen direkter und inverser Kinematik

Fragen

- Welche Zusammenhänge beschreibt die Kinematik?
- Der in Abbildung 12 skizzierte Roboter hat die Länge $l = 500\ mm$ und die Achsposition $q = 32{,}76°$. Berechnen Sie die x- und y-Position des Endeffektors bezogen auf das eingezeichnete Koordinatensystems mit der direkten Kinematik.
- Welche Unterschiede gibt es zwischen dem DKP und dem IKP?

2.6 Redundanzen

Wir betrachten hier zwei verschiedene Redundanzen. Die funktionale und die kinematische Redundanz. Die funktionale Redundanz hat einen sicherheitstechnischen Hintergrund, während es sich bei der kinematischen Redundanz um strukturelle Eigenschaften von Roboterkinematiken handelt.

2.6.1 Funktionale Redundanzen

Redundanz bedeutet, dass in einem technischen System eine zusätzliche oder vergleichbare Ressource vorhanden ist, die gleiche oder ähnliche Funktionen wie bereits vorhandene Ressourcen erfüllt. Das Ziel besteht darin, die Ausfall-, Funktions- und Betriebssicherheit zu erhöhen. Wenn beispielsweise ein Sensor ausfällt, kann ein zusätzlich eingebauter Sensor die Betriebssicherheit des Systems aufrechterhalten und einen Ausfall vermeiden. Dieses Konzept wird auch als funktionale Redundanz bezeichnet, da es darauf abzielt, sicherheitstechnische Systeme mehrfach parallel auszulegen, damit im Falle eines Ausfalls einer Komponente die anderen den Dienst weiterhin gewährleisten können. Zusätzlich versucht man, die redundanten Systeme räumlich voneinander zu trennen, um das Risiko einer gemeinsamen Störung zu minimieren. Eine zusätzliche Ressource kann auch von unterschiedlichen Herstellern stammen, um das Risiko eines systematischen Fehlers während der Konstruktion oder Produktion des Bauteils zu vermeiden. Dieses Konzept wird auch als diversitäre Redundanz bezeichnet.

2.6.2 Kinematische Redundanzen

In der Robotik bezieht sich kinematische Redundanz auf die Eigenschaft, dass ein Roboterarm oder -manipulator mehr Freiheitsgrade besitzt als für eine bestimmte Aufgabe erforderlich sind. Mit anderen Worten, der Roboter hat die Fähigkeit, die gleiche Aufgabe auf verschiedene Weise auszuführen, indem er zusätzliche Gelenke oder Achsen nutzt, die nicht unbedingt benötigt werden.

Dies ermöglicht es dem Roboter, eine höhere Flexibilität und Anpassungsfähigkeit zu erreichen, jedoch erfordert es auch eine komplexe Steuerung und Planung, um die Bewegungen des Roboters zu koordinieren und optimieren. In diesem Zusammenhang ist die kinematische Redundanz ein wichtiges Thema in der Forschung, da sie dazu beitragen kann, die Leistungsfähigkeit und Effizienz von Robotersystemen zu verbessern.

Ein Beispiel für Redundanz kann am menschlichen Arm veranschaulicht werden. Wenn man die Hand flach auf den Tisch legt und die Schulter starr hält (das Kugelgelenk darf sich bewegen), kann man den Ellenbogen bewegen. Der Grund dafür sind die insgesamt sieben Gelenke in Schulter, Ellenbogen und Handgelenk. Dadurch entsteht ein zusätzlicher (redundanter) Freiheitsgrad bei der Lagevorgabe der Hand im dreidimensionalen Raum, welcher es ermöglicht, eine Nullraumbewegung durchzuführen. Obwohl sich die kinematische Struktur des Arms bewegt, bleiben die Positionen der Hand (Endeffektor) und des Oberkörpers (Roboterbasis) unverändert.

Bei einer höheren Anzahl an Gelenken fallen Mehrkosten an, da die Harmonic Drive Getriebe und Antriebseinheiten einen hohen Anteil der Materialkosten tragen.

Lerninhalte

- Definition der Redundanz
- Erklärung von Redundanz in Bezug auf die Robotik

Fragen

- Welche Redundanz liegt bei einem Roboter mit 7-Achsen vor?

2.7 Singularitäten

Singularitäten treten in besonderen Stellungen der Roboterachsen auf und führen zu einer Reduktion des Arbeitsraumes des Roboters. Vor allem in der Nähe von singulären Konfigurationen kommt es zu hohen Gelenkgeschwindigkeiten und der Roboter kann in bestimmte Raumrichtungen seine Sensitivität gegenüber Krafteinwirkungen verlieren (Verletzungsrisiko). Dies stellt insbesondere bei kooperierenden Anwendungen, in denen der Mensch ohne Schutzzaun direkt mit dem Roboter zusammenarbeitet, ein hohes Sicherheitsrisiko dar.

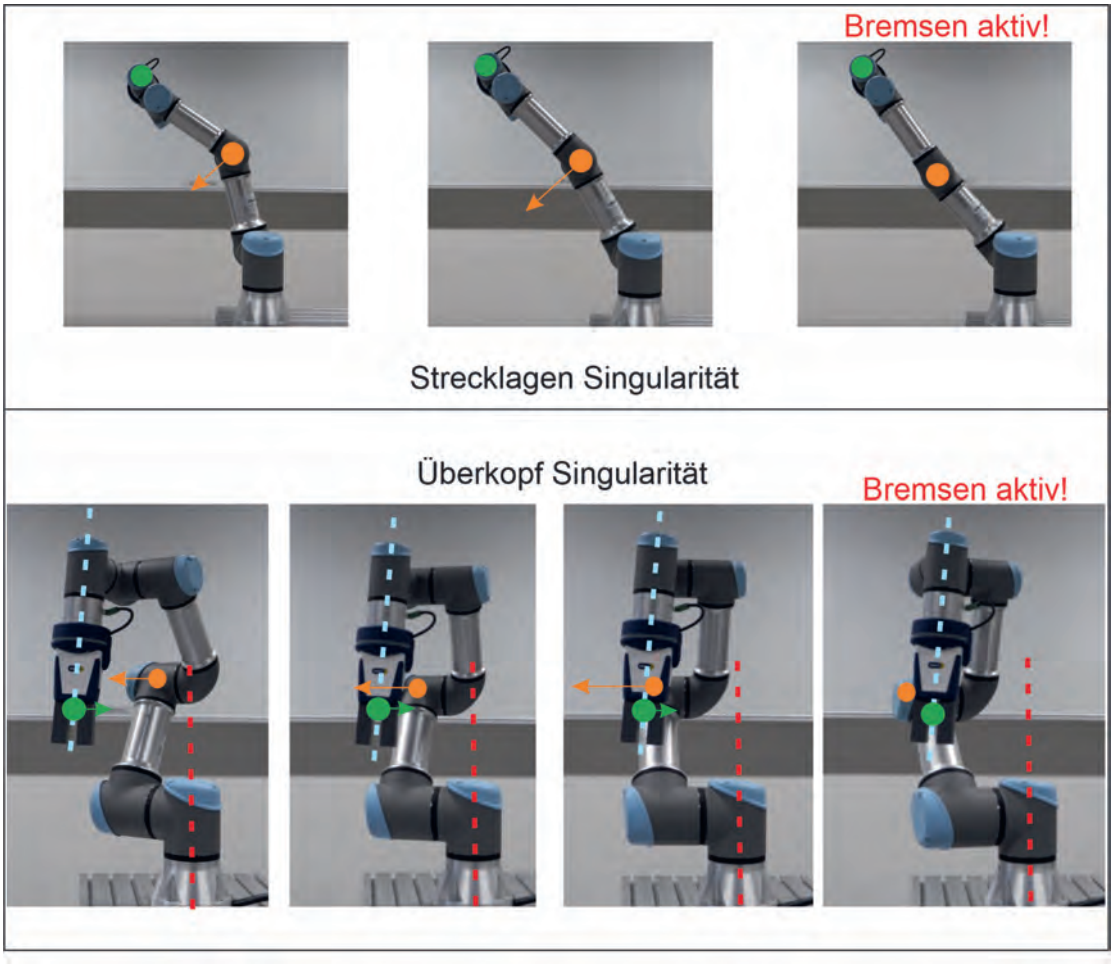


Abbildung 13: Singularitäten bei einem Universal Robots UR3

Beispiel: In Abbildung 13 sind zwei Beispiele für singuläre Konfiguration verdeutlicht. Die Strecklagen-Singularität tritt am Rande des Arbeitsbereiches des Roboters auf, da dort die Achsen zwei und drei den gleichen Winkel aufweisen. Durch die völlige Streckung des Armes wird die Bewegungsmöglichkeit eingeschränkt. Somit ist es nicht möglich in einer lokalen Umgebung, den Endeffektor in radiale Richtung zu bewegen. Ein Freiheitsgrad geht verloren. Eine Kraft, die jetzt in Längsrichtung des Roboters auf den Endeffektor wirkt, während der Roboter in der Strecklage verweilt, kann nicht über die Motorströme und auch nicht über Drehmomenten-Sensoren detektiert werden. Folglich kann der Roboter auch nicht nachgiebig und sicher reagieren. Gut zu erkennen ist hier die schnelle Bewegung der orange markierten Achse bei relativ geringer Bewegung des in grün markierten Endeffektors.

Darunter dargestellt ist die Überkopf-Singularität, bei der die zwei eingezeichneten (gestrichelten) Achsen sich nähern. Bevor sie jedoch übereinanderliegen, löst das System einen Stop aus und zieht die Bremsen an. Kurz vorher kann wieder beobachtet werden, wie sich das orange markierte Gelenk deutlich schneller als der Endeffektor bewegt.

Im Arbeitsraum des Roboters gibt es auch weniger intuitive singuläre Konfigurationen. Diese singulären Stellungen können mathematisch berechnet und softwareseitig erkannt werden. Die Systeme haben unterschiedliche Strategien auf eine solche Konfiguration zu reagieren. Häufig blockieren als Sicherheitsmaßnahme die Bremsen in den Gelenken und der Roboter muss erst wieder freigeschaltet werden.

Der Ursprung ist ein mathematisches Problem. Der Zusammenhang zwischen einer Änderung des Endeffektors und einer Änderung der Gelenkwinkel kann als linearisiertes Gleichungssystem dargestellt werden, welches im Falle einer Singularität nicht lösbar ist. Das bedeutet anschaulich, dass es einen oder mehrere Endeffektor-Freiheitsgrade gibt, die sich durch das Ändern der Gelenkwinkel (in dieser Konfiguration!) nicht mehr beeinflussen lassen. Auf eine vollständige Beschreibung des mathematischen Problems wird an dieser Stelle verzichtet.

Die Robotersteuerung kann somit in der singulären Stellung kein eindeutiges Ergebnis berechnen. Deshalb wird in der Regel nie eine exakte Singularität erreicht. Nur eine Annäherung an diesen Bereich ist möglich. Durch den eingeschränkten Freiheitsgrad in der Nähe einer Singularität kann man auch feststellen, dass die Rotationsgeschwindigkeiten der einzelnen Achsen sichtbar und hörbar ansteigt. Theoretisch würden die Geschwindigkeiten ins Unendliche steigen. Jedoch ist das technisch nicht möglich da die Singularität nie exakt erreicht wird und die Steuerung zuvor abschaltet.

Lerninhalte

- Singularitäten in der Robotik
- Sicherheitsrelevante Aspekte und Folgen einer. singulärer Konfigurationen

Fragen

- Wie erkennt man, dass sich der Roboter in einer singulären Stellung befindet?
- Wie reagieren Systeme auf singuläre Konfigurationen?

2.8 Dynamik

Die Dynamik ist das Teilgebiet der Mechanik, dass sich mit der Wirkung von Kräften erfasst. In der Mechanik wird unter Dynamik die Beschreibung der Bewegung von Körpern in Abhängigkeit von den einwirkenden Kräften verstanden. In diesem Kapitel werden im Vergleich zur Kinematik am Roboter wirkende Kräfte berücksichtigt.

2.8.1 Simulation mithilfe der direkten Dynamik

Um komplexe Systeme analysieren zu können, benötigen Ingenieure, Anlagenplaner und Wissenschaftler digitale Simulationsumgebungen, die die Gegebenheiten korrekt abbilden. Dazu gehört das physikalische Verhalten eines Roboters. Die direkte Dynamik, beschreibt die Gelenkbeschleunigungen $\ddot{q}$, welche aus Antriebskräften τ mit den aktuellen Gelenkwinkeln q und Gelenkgeschwindigkeiten $\dot{q}$ sowie den externen Kräften F am Endeffektor resultieren. Der Zusammenhang wird beschrieben durch eine Funktion f

$$\ddot{\boldsymbol{q}}(t) = \boldsymbol{f}\big(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{\tau}(t), \boldsymbol{F}(t)\big).$$

Anwendung finden derartige Modelle primär in der Robotersimulation. Die direkte Dynamik wird seltener in der Praxis benötigt. Simulationsumgebungen wie MuJoCo (Multiple Joint Contact), Matlab Robotics Toolbox oder Software von Roboterherstellern wie Universal ermöglichen unter anderem die Untersuchung von Roboteranwendungen hinsichtlich der Umsetzbarkeit, Auslegung der Roboter, Produktionszeit und ggf. Aufbau der Peripherie.

2.8.2 Inverse Dynamik zur Berechnung von notwendigen Motorströmen

Im Gegensatz dazu beschreibt die inverse Dynamik, die erforderlichen generalisierten Kräfte und Momente τ für eine gegebene Gelenkbewegung q, $\dot{q}$, $\ddot{q}$ sowie externen Kräften am Endeffektor F. Das bedeutet praktisch, dass dem Roboter ein gewünschtes dynamisches Verhalten vorgegeben wird (zum Beispiel aus einer Bahnplanung) und die Robotersteuerung daraus die Momente der Motoren errechnet, um diese Bewegung korrekt auszuführen. In der Praxis ist dies häufig der Fall, wenn ein System gesteuert wird. Bei jedem Punkt den man dem Roboter über das Bediengerät vorgibt wird in der Regel auch nach der gewünschten Geschwindigkeit und Beschleunigung gefragt. Der funktionale Zusammenhang lautet demnach

$$\boldsymbol{\tau}(t) = \boldsymbol{g}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t), \boldsymbol{\mathcal{F}}(t))$$

2.8.3 Bahnplanung

Gibt der Anwender dem Roboter mehrere Point-to-Point Bewegungen vor, die aus mehreren einzelnen Endeffektor-Lagen bestehen, muss eine Verbindung einzelner Punkte berechnet werden. Dieser Vorgang kann durch eine Interpolation gelöst werden. Dazu gibt es eine Vielzahl an Ansätzen die alle einen zeitlichen Verlauf der Gelenkwinkel, -geschwindigkeiten und –beschleunigungen liefern. Solche Ansätze sind Teil der Bahnplanung. Im Gegensatz zu einer Wegplanung wird hier explizit die Zeit mit einbezogen. Mithilfe der inversen Dynamik können aus den entstandenen Bahnen die zeitlichen Verläufe für die generalisierten Kräfte und Momente berechnet und an die Steuerung übergeben werden. Dies wird im nächsten Kapitel aufgegriffen.

Neben der Point-to-Point Bewegung gibt es noch die Continous-Path-Bewegung die von einer Start- zu einer Endlage über Zwischenlagen interpoliert und den Interpolator, der im Regeltakt der Steuerung Sollwerte für die Bahn an die Motoren liefert.

Trajektorien- und Bahnplanung beschreiben den selben Vorgang.

Lerninhalte

- Was versteht man unter Dynamik?
- Direkte Dynamik
- Inverse Dynamik
- Was ist eine Bahnplanung?

Frage

- Erklären Sie den Unterschied zwischen Kinematik und Dynamik in der Physik.

2.9 Steuerung und Regelung

Ziel dieses abschließenden Abschnitts ist es zu verstehen wie alle bisher vorgestellten Komponenten zusammenwirken, um Aufgaben in der realen Welt zu bearbeiten. Dabei wird anhand eines Beispiels auf den Unterschied zwischen Steuern und Regeln eingegangen.

Der Prozessrechner innerhalb der Steuereinheit eines Roboters ist für die Verarbeitung aller Informationen im System zuständig. Die grafische Nutzeroberfläche (engl. Graphical user interface (GUI)) eines Cobots wie dem Universal Robot, Franka Emikas oder auch Dobots erlauben dem Nutzer mit den im Robote laufenden Programmen zu kommunizieren. Im Hintergrund laufen beispielsweise Zustandsautomaten, Sicherheitsfunktionen und Regler, die das Verhalten des Roboters bestimmen.

2.9.1 Bedienung eines Roboters

Ein Bediener nutzt die vom Hersteller zur Verfügung gestellten Schnittstellen, welche meist in Form einer GUI vorliegen. Hier können je nach Funktionsumfang des Systems bspw. Positionen angelernt (geteached), Soll-Positionen in Raumkoordinaten oder Gelenkwinkeln eingegeben oder blockbasiert Programme geschrieben werden (für genaue Beschreibungen sind in diesem Kompendium eigene Abschnitte vorhanden). Oft erlauben die Systeme auch bis zu einem gewissen Grad die textbasierte Programmierung mit vordefinierten Kommunikationsschnittstellen.

Die Befehle werden über die Nutzerschnittstelle (user interface) an den Prozessrechner übermittelt. Hier sind häufig Zustandsautomaten hinterlegt. Dies sind Programme, die sich in einem bestimmten Zustand befinden (z. B. „warte auf Nutzereingabe“) und durch Ereignisse (z. B. „Nutzerbefehl erkannt“) in einen anderen Zustand (z. B. „werte Nutzereingabe aus“) überführt werden. Aus diesen Zuständen können weitere Programme gestartet werden, die Hintereinander oder Gleichzeitig ausgeführt werden.

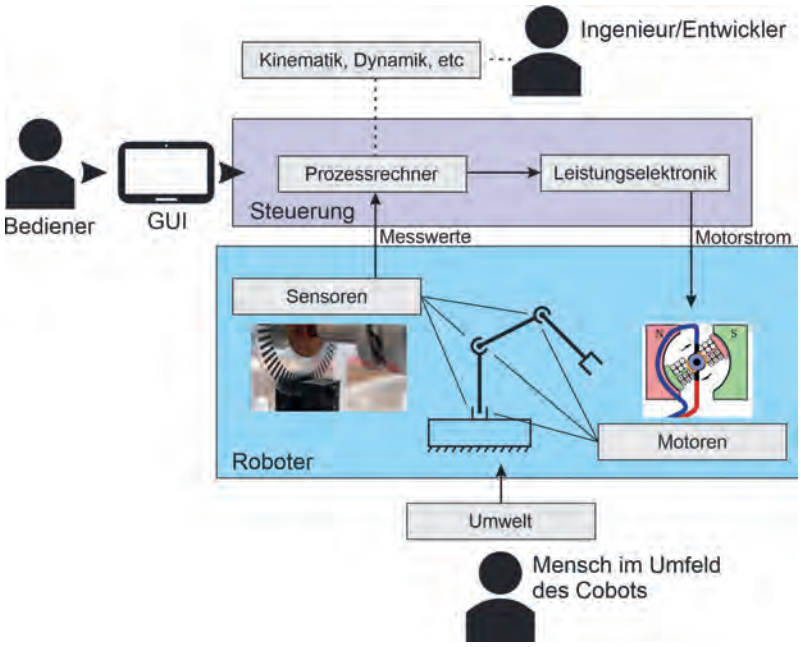


Abbildung 14: Zusammenhang vorheriger Kapitel

2.9.2 Steuerung eines Roboters

Wir folgen beispielhaft einer Nutzereingabe „Beweg den Endeffektor zu Position x“. Auf dem Prozessrechner muss nun berechnet werden, wie der Roboter aus seiner aktuellen zur vorgegeben Position kommt. Mithilfe der inversen Kinematik (siehe Kapitel Kinematik) werden die notwendigen Gelenkwinkel zum Erreichen von Position x berechnet. Anschließend plant der Roboter eine Bahn von den aktuellen Gelenkwinkeln zu den gewünschten (Stichwort: Bahnplanung). Hier gibt es eine Vielzahl von mathematischen Ansätzen zur Interpolation dieser beiden Gelenkwinkel. Ergebnis ist hier ein Bewegungsprofil mit zeitlichen Verläufen der Winkelpositionen, -geschwindigkeiten und -beschleunigungen.

Setzt man diese zeitlichen Verläufe in die inverse Dynamik ein, erhält man für jeden Zeitpunkt ein Antriebsmoment, dass über den Zusammenhang des Getriebes und des Elektromotors in einen Motorstrom umgerechnet wird. Dieser Motorstrom wird jetzt durch die Leistungselektronik gestellt und der Roboter startet seine Bewegung zur Position x.

Was hier beschrieben wird, ist eine klassische modellbasierte Steuerung. Ob der Roboter sein Ziel tatsächlich erreicht, wird mit dieser Bewegungsstrategie nicht überprüft. Es wird voll darauf vertraut, dass die Modelle genau zu dem Roboter passen und das es nicht zur Kollision kommt. Ein solcher Ansatz ist nicht für die Mensch-Roboter-Kollaboration geeignet. Hierzu muss der Roboter taktil sein, um korrekt reagieren zu können.

2.9.3 Regelung eines Roboters

Bei der Regelung wird in Gegensatz zur Steuerung auf Sensoren und Messtechnik zurückgegriffen, um den aktuellen Zustand des Roboters und der Umgebung korrekt zu erfassen. Anschaulich kann man diesen Unterschied gut mit einem Zettel und einem Stift demonstrieren.

Die Norm **DIN IEC 60050-351** definiert die Regelung wie folgt: „Die Regelung bzw. das Regeln ist ein Vorgang, bei dem fortlaufend eine Größe, die Regelgröße, erfasst, mit einer anderen Größe, der Führungsgröße, verglichen und im Sinne einer Angleichung an die Führungsgröße beeinflusst wird. Kennzeichen für das Regeln ist der geschlossene Wirkungsablauf, bei dem die Regelgröße im Wirkungsweg des Regelkreises fortlaufend sich selbst beeinflusst."

Kleines Experiment zur Veranschaulichung von Steuerung und Regelung: Man platziert ein Blattpapier auf dem Tisch und setzt zwei Kreuze mit einem Bleistift. Mit dem Stift in der Hand wird die Spitze auf eins der beiden Kreuze gesetzt. Das Ziel ist jetzt eine Bahn vom ersten Kreuz (der Ist-Position) zum zweiten Kreuz (der Soll-Position) zu ziehen und genau auf dem zweiten Kreuz zu stoppen. Dieser Vorgang wird einmal mit offenen und einmal mit geschlossenen Augen durchgeführt. Um dabei äußere Einflüsse zu simulieren kann eine zweite Person das Blatt bei dem Vorgang leicht bewegen.

Das Ergebnis ist offensichtlich. Wenn wir den Ist-Zustand unseres Armes und des Stifts sehen können, dann wissen wir wohin wir korrigieren müssen. Sind die Augen geschlossen muss sich vollständig auf unsere Einschätzung verlassen werden und bei Störungen liegt diese komplett daneben. Die Augen sind in diesem Beispiel Sensoren, unser Arm der Roboter, der Stift unser End-Effektor, das Papier der Arbeitsraum, und die andere Person eine Störgröße. Die Bahnplanung führen wir im Kopf automatisch durch. Zu jedem Zeitpunkt korrigieren wir unsere Eingabe (Muskelkraft) auf Basis eines Messsignals (Stiftposition über Augen). Diesen Prozess könnte man vereinfacht als Regelung bezeichnen.

Regelungsstrategien von Robotern sind komplex. Beim vorangegangenen Beispiel aus der Steuerung würde der Motorstrom zusätzlich durch eine Auswertung des Fehlers zwischen gemessenen Ist-Gelenkwinkel (aus den Inkrementalgebern) und Soll-Gelenkwinkel (aus der Bahnplanung) beeinflusst werden. Bei der Handführung eines Cobots werden durch das drücken der entsprechenden Schalter spezielle Regelstrategien aktiviert, die das Eigengewicht des Roboters kompensieren (zero-gravity Modus) und auf externe Kräfte durch den Menschen reagieren in dem sich der Roboter aktiv wegbewegt.

Lerninhalte

- Zusammenhang vorangegangener Kapitel
- Erklärung des Begriffes Regelung
- Erklärung des Begriffes Steuerung
- Unterschiede zwischen Steuerung und Regelung

Frage

- Was sind der Unterschied zwischen einer Steuerung und einer Regelung?
- Erklären Sie in Ihren eigenen Worten den Begriff Regelung.

2.10 Programmierung

Für die Programmierung von Robotersystemen gibt es verschiedene Möglichkeiten, die sich je nach Anwendungsgebiet, Hersteller und Vorgehensweise unterscheiden. Für kollaborative Systeme sind dies grundsätzlich die **Offline-Programmierung**, **Online-Programmierung und Handführung**.

Bei der **Online-Programmierung** wird der Roboter in einer vom Hersteller bereitgestellten Sprache programmiert. Die Eingabe findet dabei beispielsweise über ein Tablet statt, welches über eine Datenleitung mit dem System verbunden sein kann. Ein Vorteil dieser Methode ist die schnelle Anpassung von Programmteilen, da potentiell Fehler sehr schnell erkannt werden können. Ein offensichtlicher Nachteil besteht dabei aber in der Bindung an das System, wodurch die Flexibilität in der Programmierung eingeschränkt wird.

Bei der **Offline-Programmierung** wird ein Programm auf einem beliebigen Computer erstellt und anschließend auf den Roboter übertragen. Je nach System kann die Programmierung dabei zum Beispiel in einer vom Hersteller bereitgestellten Simulationsumgebung erfolgen. Darin lassen sich meistens erzeugte Programme an einer Simulation des Roboters testen. Dafür werden häufig **blockbasierte Programmiersprachen** verwendet, welche sich aufgrund ihrer geringen Komplexität insbesondere für jüngere Menschen gut eignen. Dabei werden auf einer grafischen Oberfläche kleine Bausteine oder Blöcke, die jeweils einen eigenen Befehl darstellen, hin- und hergeschoben. Diese Befehle können je nach Anwendungsfall beliebige Funktionen abbilden, wie zum Beispiel die Ausführung einer kleinen Bewegung. Die Abfolge, in der die Blöcke angeordnet werden bestimmt die Reihenfolge, in der die Befehle umgesetzt werden. Manche Bausteine beinhalten außerdem Variablen oder Parameter, über die sich beispielsweise die Geschwindigkeit oder Beschleunigung einstellen lassen.
Eine Alternative zu den blockbasierten Sprachen stellen **textbasierte Sprachen** dar. Im Robotikbereich sind sowohl C/C++ als auch Python weit verbreitet, die auf der einen Seite zwar komplexer, auf der anderen Seite dafür aber deutlich umfangreicher als blockbasierte Sprachen sind. Dadurch ergeben sich mehr Möglichkeiten und die Funktionalitäten der Sprachen wie Schleifen oder „If-Anweisungen“ können frei genutzt werden. Je nach Robotersystem wäre dies bei der Verwendung der entsprechenden blockbasierten Sprachen nicht oder nur teilweise umsetzbar.
Ein Vorteil der Offline-Programmierung im Gegensatz zur Online-Programmierung ist die dadurch vorhandene Flexibilität, da Programme ohne dauerhafte Anbindung an das Robotersystem erstellt und bearbeitet werden können. Allerdings sind im Anschluss an die Übertragung auf das System häufig kleine Anpassungen bei der Position oder der Orientierung notwendig, da die Simulationsumgebung keine vollständige Abbildung der Realität ermöglicht und somit Abweichungen in der Programmierung fast nicht vermieden werden können.

Eine dritte Möglichkeit zur Programmierung besteht in der **Handführung**. Diese beschränkt sich im Wesentlichen auf die Nutzung von leichten, kollaborativen Robotersystemen. Bei der Handführung kann durch das Drücken einer entsprechenden Taste am Roboter manuell eine Bewegung vom Nutzer durchgeführt werden, die im Nachhinein vom Robotersystem selbstständig wiederholt werden kann. Dadurch ist eine Nachbildung von Bewegungen oder einer ganzen Abfolge von Bewegungen möglich. Diese Vorgehensweise ist sehr einfach und anschaulich, wodurch eine schnelle Programmierung und Ausführung eines Programmes erreicht werden kann. Zusätzlich können Fehler verhindert werden, bevor diese überhaupt auftreten. Ein Problem dabei besteht allerdings in den stark eingeschränkten Möglichkeiten, die nur durch eine Online- oder Offlineprogrammierung realisierbar sind.

2.11 Quellen

Albu-Schäffer, A., Haddadin, S., Ott, C., Stemmer, A., Wimböck, T., & Hirzinger, G. (2007). The DLR Lightweight Robot – Design and Control Concepts for Robots in Human Environments. *Industrial Robot: An International Journal*, 34, 376–385.

Craig, J. J. (2004). *Introduction to Robotics: Mechanics and Control*. New, Jersey, USA: Prentice Hall.
Gevatter, H.-J., & Grünhaupt, U. (Hrsg.). (2006). Handbuch der Mess- und Automatisierungstechnik in der Produktion. Springer Berlin Heidelberg. doi:10.1007/3-540-34823-9

Heimann, B., Gerth, W., & Popp, K. (2006). *Mechatronik: Komponenten – Methoden – Beispiele*. München, Deutschland: Carl Hanser Verlag.

Hesse, S., & Schnell, G. (2011). *Sensoren für die Prozess- und Fabrikautomation: Funktion – Ausführung – Anwendung*. Wiesbaden, Deutschland: Vieweg+Teubner Verlag.

Hirzinger, G., Sporer, N., Albu-Schaffer, A., Hdmle, M., Krenn, R., Pascucci, A., & Schedl, M. (2002). DLR's Torque-Controlled Light Weight Robot III – are we Reaching the Technological Limits now? *Proceedings of the 2002 IEEE lnternational Conference on Robotics & Automation*, (S. 1710–1716). Washington.

Khalil, W., & Dombre, E. (2002). *Modeling, Identification & Control of Robots*. New, York, USA: Routledge.
Pfeiffer, F., & Reithmeier, E. (1987). Roboterdynamik. Stuttgart, Deutschland: Teubner Verlag.

Schrüfer, E. (2007). *Elektrische Messtechnik*. München, Deutschland: Carl Hanser Verlag.

Sciavicco, L., & Siciliano, B. (2000). *Modelling and Control of Robot Manipulators*. Berlin, Deutschland: Springer.

Siciliano, B., & Khatib, O. (Hrsg.). (2008). *Springer Handbook of Robotics*. Springer Berlin Heidelberg. doi:10.1007/978-3-540-30301-5

Bildquellen

[#Q1] Abbildung des Honda Roboters: Zoohouse, The Honda ASIMO humanoid robot. 2010. Zugegriffen: 01.05.2023. [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/File:HONDA_ASIMO.jpg [CC-BY-SA-3.0], Abbildung des Roboters Pepper: Tokumeigakarinoaoshima, Service-Roboter Pepper. 2014. Zugegriffen: 01.05.2023. [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/File:SoftBank_pepper.JPG [CC0 1.0]

[#Q2] Standbild aus: MichaelFrey, Animation einer Gleichstrommaschine. 2015. Zugegriffen: 01.05.2023. [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/File:Animation_einer_Gleichstrommaschine.gif [CC-BY-SA-3.0]

[#Q3] Jahobr & Kaboldy, Schnitt durch ein Harmonic-Drive-Getriebe. 2018. Zugegriffen: 01.05.2023. [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/File:Harmonic_drive_cross_Legend.svg [CC-BY-3.0]

[#Q4] Kombination und Beschriftung zweier Bilder: (1) Tycho, Inkrementalgeber mit Gabellichtschranke. 2008. Zugegriffen: 01.05.2023. [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/File:Inkrementalgeber_mit_gabellichtschranke.JPG [CC-BY-SA-3.0], (2) MatthiasDD, Schema eines Glasmaßstabes. 2014. [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/File:Linear_Scale_Scheme.svg [CC-BY-SA-3.0]

[#Q5] Um Drehmomentenpfeile und Beschriftung ergänzt: M. Laible, Käfigläufer als Federkörper (Prinzip). 2007. Zugegriffen: 01.05.2023. [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/File:Torque_Cage.gif#/media/File:Torque_Cage.gif [CC BY-SA 2.5]

[#Q6] Foto Sawyer Roboter: M. Laible, Sawyer Collaborative Robot with a ClickSmart gripper from Rethink Robotics. 2018. Zugegriffen: 01.05.2023. [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/File:ClickSmart_6.jpg#/media/File:ClickSmart_6.jpg [CC BY 4.0]

3 Umgang mit dem Dobot Magician

Inhalt

3.1 Grundlagen zum Dobot Magician

Rene Egbers, Fabian Icken, Marcus Auf der Landwehr, Hochschule Osnabrück

Der *Magician* des chinesischen Herstellers *Dobot* ist ein 4-Achs-Desktop-Roboter, der für die Aus- und Weiterbildung konzipiert wurde. Im Rahmen der Robonatives Förderung haben sich der überwiegende Teil allgemeinbildender Schulen und vereinzelt auch berufsbildende Schulen für die Anschaffung dieser Geräte entschieden. Die Popularität des Magicians lässt sich darauf zurückführen, da es sich bei diesem Gerät um eine vielseitige und im Vergleich zu industriellen Systemen deutlich kostengünstigere Variante handelt, mit der sich viele Inhalte des Unterrichts visualisieren und erlebbar machen lassen. Zur Bezeichnung des *Magicians* wird häufig auch simultan die Bezeichnung *Dobot* verwendet. Neben dem eigentlichen Roboter, bietet der Hersteller umfangreiches Zubehör an. Zusätzlich erworben werden können ein Förderband, eine Linearachse, ein Farbsensor, ein Reflexlichttaster (vom Hersteller als photoelektrischer Sensor bezeichnet), ein Kamerasystem (vom Hersteller als Vision Kit bezeichnet) und ein Arduino-KI-Kit (vom Hersteller als Basic-AI-Kit bezeichnet). Im Lieferumfang eines einzelnen Roboters sind neben einem Parallelbacken- und einem Sauggreifer, einem Kompressor, einem Stifthalter und einigen Schaumstoffwürfeln auch Zubehörteile für das 3D-Drucken mit dem Magician enthalten. Der Roboter unterstützt ein Handhabungsgewicht von maximal 500 g und erreicht eine Wiederholungsgenauigkeit von 0,2 mm.
Die folgende Abbildung zeigt einen Magician mit montiertem Parallelbackengreifer.

Abbildung 1: Dobot Magician mit Parallelbackengreifer

Der Magician ist mit einer Parallelkinematik ausgestattet, was es nur ermöglicht, Punkte „von oben“ anzufahren. Der Endeffektor kann nur um seine z-Achse gedreht werden, jedoch nicht um seine x- oder y-Achse. Die folgenden Darstellungen zeigen neben den Gelenken des Roboters auch die Lage des Basis-Koordinatensystems. Alle Koordinaten im Arbeitsraum des Roboters werden auf diesen Punkt bezogen angegeben. Dieser Nullpunkt liegt zwischen den Antrieben der Parallelkinematik und mittig in der Roboterbasis, kann aber mit dem Endeffektor nicht selbst angefahren werden. Rund um die Basis gibt es einen Sperrbereich der Aufgrund der Kinematik nicht erreichbar ist.

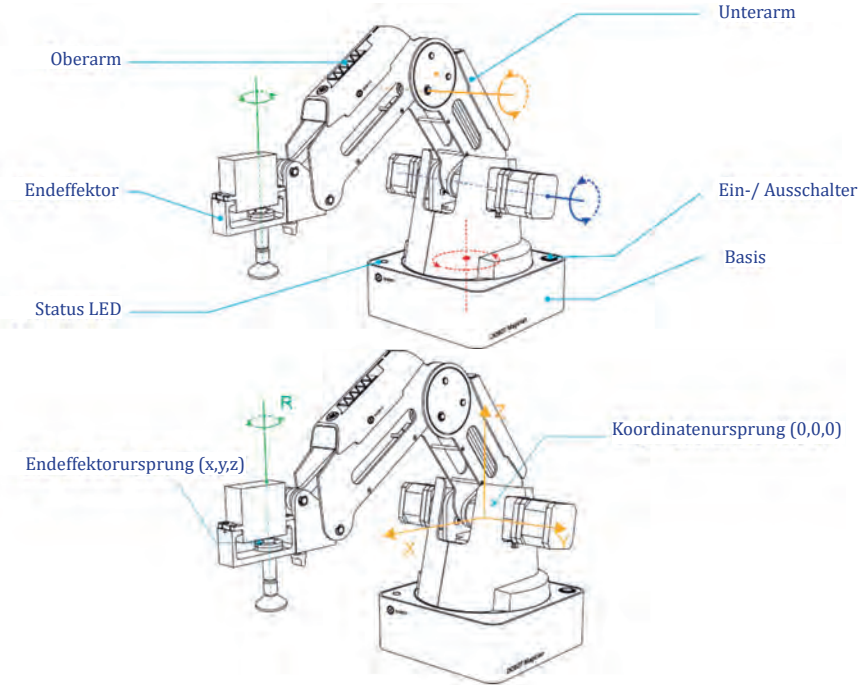


Abbildung 2: Koordinatensysteme und Aufbau des Dobot Magician

Mittlerweile werden drei Computerprogramme zur Programmierung des Magicians angeboten, die von der Webseite des Herstellers kostenfrei heruntergeladen werden können. Das originäre Programm heißt DobotStudio, danach wurde das Programm DobotBlock und vor kurzer Zeit wurde die dritte Software namens DobotLab vorgestellt. Zu Beginn des Projektes war nur DobotStudio verfügbar, weshalb die Lehrkräfte in den Projektschulen ebenfalls im Umgang mit dieser Software geschult wurden und diese Software auch bis heute im Unterricht einsetzen. DobotStudio enthält bis zuletzt kleinere Softwarebugs, die in der Anwendung aber nicht wesentlich stören. Vorteilhaft an DobotStudio ist, dass hier abgesehen vom 3D-Druck alle Funktionen, die der Dobot unterstützt, in einem Programm zusammengeführt sind. Die folgenden Ausführungen beziehen sich immer auf die Software DobotStudio. Ein Vergleich von DobotStudio und dem etwas neuerem Programm DobotBlock kann im Abschnitt „Vergleich der Programmieroptionen“ dieses Werks gefunden werden.
Die folgende Abbildung zeigt den Startbildschirm von DobotStudio.

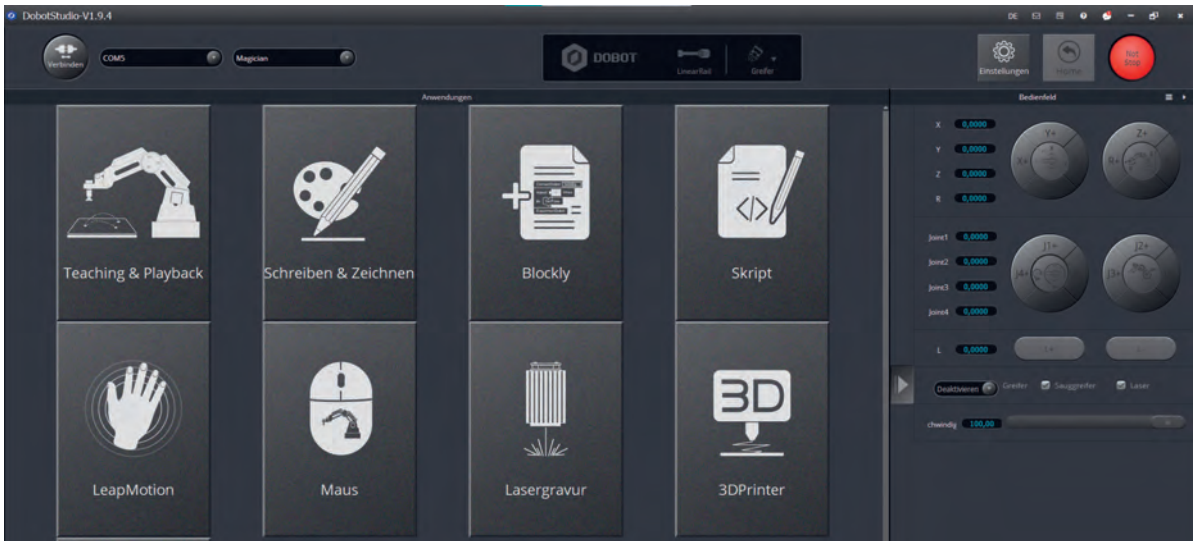


Abbildung 3: Dobot Studio Startbildschirm

In der linken oberen Ecke befinden sich die Schaltflächen, um eine Verbindung zwischen DobotStudio und dem Magician herzustellen. Zunächst muss der COM-Port ausgewählt werden, über den der Roboter mit dem PC verbunden ist. Stehen hier mehere Ports zur Auswahl, kann herausgefunden werden, welcher zum Magician gehört, indem er per USB-Kabel getrennt und wieder verbunden wird und man beobachtet, welcher COM-Port verschwindet und wieder erscheint. Den entsprechenden COM-Port dann auswählen und überprüfen, ob im Drop-down-Menü rechts daneben der Typ „Magician“ eingestellt ist. Anschließend kann über den runden Button links eine Verbindung mit dem Magician hergestellt werden. Konnte eine Verbindung hergestellt werden, wechselt der Button-Text zu „Trennen“ und auf der rechten Seite werden die runden Steuerflächen aktiv. Konnte keine Verbindung hergestellt werden, erscheint eine entsprechende Fehlermeldung.

Abbildung 4: Schaltfläche zur Herstellung einer Verbindung

Ebenfalls möglich ist, dass eine Verbindung aufgebaut werden konnte, jedoch in Anzeige oben in der Mitte eine Fehlermeldung angezeigt wird. Diese kann mithilfe eines Doppelklicks geöffnet werden und nach Beseitigung mittels der Taste „Clear Alarm“ quittiert und gelöscht werden. Wenn der Magician mittels der Freedrive-Taste anschließend aus dem Sperrbereich direkt am Roboter heraus bewegt wurde, sollte hier keine Fehlermeldung mehr auftauchen und die Signalleuchte am Roboter in grüner Farbe dauerhaft leuchten. Ebenfalls in diesem Feld befinden sich die Schaltflächen, um den Endeffektor und, falls angeschlossen, die Linearachse auszuwählen. Nach der Aktivierung ist die Schaltfläche blau hinterlegt.

Abbildung 5: Schaltfläche zur Aktivieren von Endeffektoren und der Linearachse

Neben dem Reiter zur Herstellung einer Verbindung und zur Auswahl eines Endeffektors, befindet sich in der oberen Leiste ein Reiter, auf dem die Einstellung verlinkt sind und auf dem ein Not-Stopp ausgelöst werden kann.

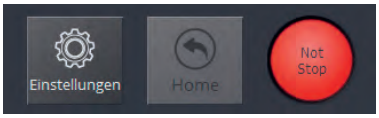


Abbildung 6: Buttons für Einstellungen, Homing und Not-Halt

Zwischen den Einstellungen und dem Not-Stopp befindet sich eine Schaltfläche, die mit „Home“ bezeichnet ist. Durch Betätigung des Home-Buttons wird eine automatische Referenzfahrt gestartet.

Über das Bedienfeld kann der Magician gesteuert werden. Möglich ist eine Bewegung in kartesischen Koordinaten in x-, y- und z-Richtung und die Rotation R des Endeffektors. Alternativ können direkt die Gelenkwinkel Joint1, Joint2, Joint3 und Joint4 des Roboters gesteuert werden. Wenn eine Linearachse angeschlossen ist, kann diese mithilfe der Buttons „L+“ und „L- “ manuell verfahren werden. Mit den Schaltflächen für „Greifer“, „Sauggreifer“ und „Laser“ können, der jeweilige Endeffektor betätigt werden. Ebenfalls lässt sich in diesem Reiter die prozentuale Geschwindigkeit des Roboters einstellen.

Abbildung 7: Buttons für Einstellungen, Homing und Not-Halt

Auf dem Startbildschirm von DobotStudio sind alle Applikationen verknüpft, die mit dem Magician genutzt werden können. Auf die wichtigsten Anwendungen wird hier kurz eingegangen:

1. **Teaching & Playback**: Hierbei handelt es sich um eine Programmierart, bei der Punkte bei einer Handführung eingelernt werden (auch einteachen genannt), die anschließend vom Magician abgefahren werden. An den Punkten können Endeffektorbefehle, z. B. öffnen und schießen des Greifers, hinzugefügt werden. Diese Art der Programmierung wird im nächsten Abschnitt genauer erklärt.
2. **Schreiben & Zeichnen**: Mit dieser Applikation kann der Magician Text schreiben und Bilder zeichnen. Im Teil „Einführung in das Schreiben & Zeichnen“ wird diese Funktion genauer beschrieben.
3. **Blockly**: Die Programmierart Blockly ermöglicht eine blockbasierte Programmierung des Magicians. Im Kapitel „Einführung in Blockly“ wird diese Programmierart genauer beschrieben.
4. **Skript**: Die Programmierart Skript ermöglicht die Programmierung des Magicians mit Python. Im Kapitel „Einführung in Python“ wird diese Programmierart genauer beschrieben.
5. **Lasergravur**: Die Funktion Lasergravur soll das Gravieren von Gegenständen mit dem Dobot ermöglichen. Der Laser und dessen Einfuhr ist jedoch in Deutschland aus Sicherheitsgründen verboten.
6. **3D Printer**: Die 3D-Druck Funktion ermöglicht es, mit dem Dobot 3D zu drucken. Hierfür muss eine andere Firmware auf den Magician geladen werden. Auch wird eine zusätzliche Slicer-Software benötigt. Unterstützt werden der Cura-und der Repetier-Slicer. Eine detaillierte Anleitung ist in den Handbüchern von Dobot enthalten.

3.2 Teaching & Playback

Rene Egbers, Fabian Icken, Marcus Auf der Landwehr, Hochschule Osnabrück

Das Teaching & Playback ermöglicht die Programmierung des Dobot durch das Setzen von Punkten, die der Roboter beim Start des Programms nacheinander abfährt. An den jeweiligen Punkten können dem Dobot noch gewisse Tätigkeiten, wie das Aktivieren und Deaktivieren des Endeffektors zugewiesen werden. Beim Öffnen der Teaching & Playback-Programmierart, erscheint zunächst die folgende Oberfläche:

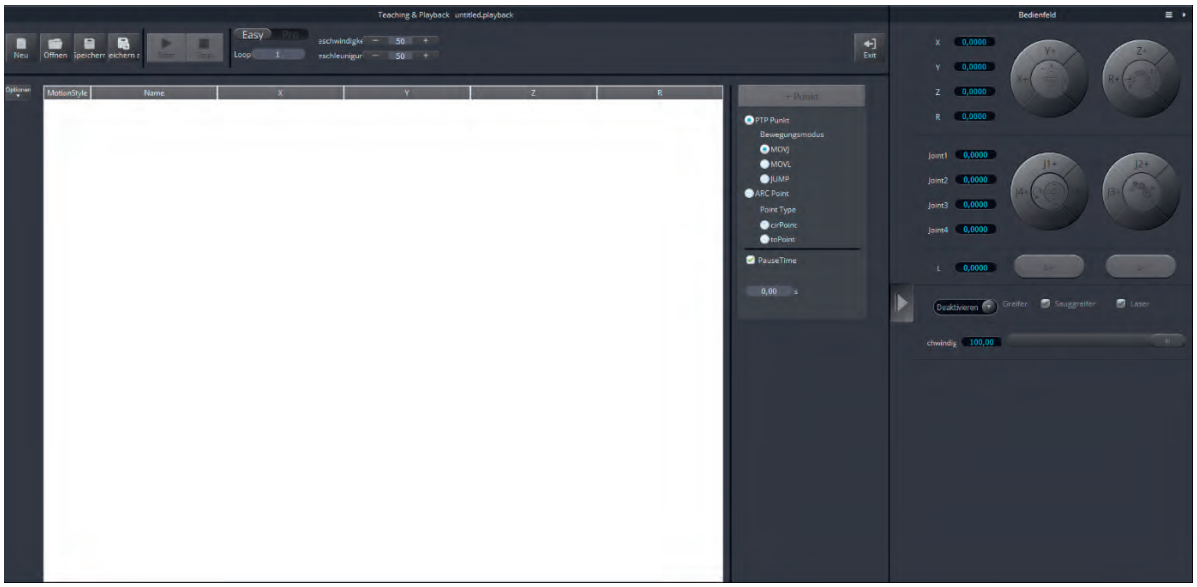


Abbildung 1: Programmieroberfläche der Anwendung Teaching & Playback

Zu sehen ist, dass die Bedienoberfläche an der rechten Seite so bleibt wie im Startbildschirm der Software. Lediglich das Fenster für die Anwendungen ändert sich. Hier erscheint die Teaching & Playback Bedienoberfläche. Diese besteht aus der oberen Leiste mit verschiedensten Funktionen, der rechten Leiste zur Festlegung der Bewegungsart und dem Programmierfeld.

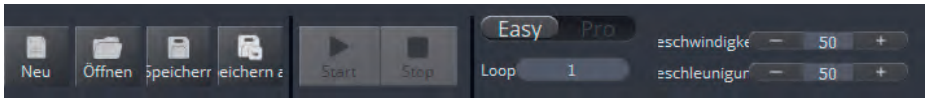


Abbildung 2: Schaltfläche mit verschiedenen Funktionen in der Anwendung Teaching & Playback

In der oberen Leiste tauchen, wie in Abbildung 2 zu sehen, mehrere Funktionen nebeneinander auf. Links sind Optionen verzeichnet, zum Starten neuer Programme, zum Öffnen von Programmen und zum Speichern von Programmen. Daneben sind die Schaltflächen zum Starten und Stoppen des Programmlaufs. Gefolgt von einer Schaltfläche zum Wechsel des Programmiermodus. Hier kann zwischen Easy und Pro unterschieden werden. Unter dieser Schaltfläche befindet sich die Schaltfläche „Loop“, hier kann die Anzahl der Wiederholungen des Programmablaufs festgelegt werden. An der rechten Seite dieser Liste befindet sich eine Schaltfläche, mit der die Geschwindigkeit und Beschleunigung des Magicians in % reguliert werden können.
Die Leiste mit den Bewegungsarten sieht wie folgt aus.

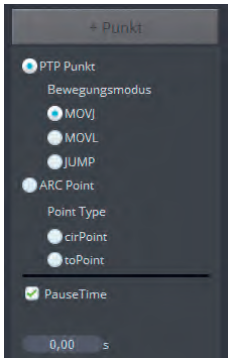


Abbildung 3: Bedienfeld zur Auswahl der Bewegungsart

Hier kann zwischen *PTP Punkt* und *ARC Point* unterschieden werden. Ebenfalls kann eine Pausenzeit (*PauseTime*) an den jeweiligen Punkten festgelegt werden. Unter *PTP Punkt* sind mehrere Bewegungsmodi verzeichnet. Diese werden im Folgenden genauer erörtert.

Bewegungsmodus MOVJ

Der Bewegungsmodus MOVJ bildet eine gängige PTP-Bewegung ab. Bei einer PTP-Bewegung werden die Anfangs- und Endpunkte einprogrammiert, zwischen denen sich der Roboter bewegen soll. Die Steuerung berechnet einmalig die benötigten Gelenkwinkel, die der Roboter zum Erreichen seines Endpunktes anfahren muss und bewegt die Antriebe genau in diese Position. Im industriellen Bereich sind diese Bewegungen schneller als die anderen Bewegungsarten. Es kann jedoch zu einem unvorhersehbaren Bewegungsablauf kommen. Ein Kennzeichen für das Verfahren im MOVJ-Bewegungsmodus bzw. in einer PTP-Bewegung ist eine eher kreisförmige Bewegung des Roboters.

Bewegungsmodus MOVL

Der Bewegungsmodus MOVL bildet hingegen eine gängige Linearbewegung ab. Bei einer Linearbewegung werden wie bei der PTP-Bewegung Anfangs- und Endpunkt einprogrammiert. Die Steuerung versucht nun durch eine Interpolation eine möglichst gerade Linie zwischen den beiden Punkten abzufahren. Hierfür werden Punkte auf der geraden Linie zwischen den Bahnen berechnet und in einem bestimmten Interpolationstakt (dieser liegt im Millisekundenbereich) abgefahren. Hierbei gilt, je geringer der Interpolationstakt, umso genauer fährt der Roboter die Bahn ab. Es entsteht eine annähernd gerade Bahn zwischen dem Anfangs- und dem Endpunkt, die der Roboter abfährt. Abbildung 4 zeigt die schematische Darstellung einer Linearbewegung.

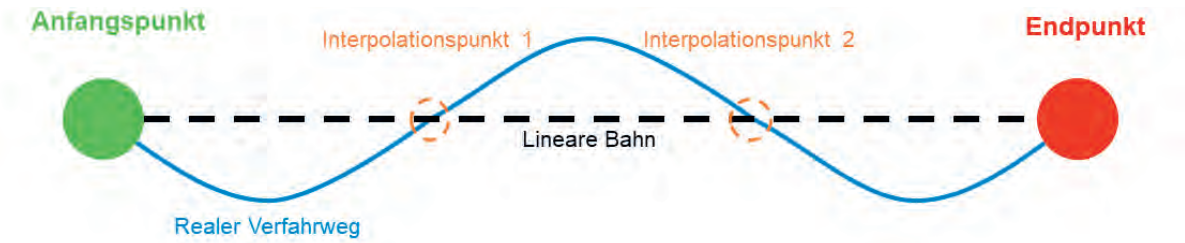


Abbildung 4: Schematische Darstellung einer Linearbewegung

Bewegungsmodus JUMP

Der Bewegungsmodus JUMP ist eine Kombination aus einer PTP-Bewegung und einer Linearbewegung. Es werden, wie bei den beiden anderen Bewegungsmodi, ein Anfangs- und ein Endpunkt einprogrammiert. Der Dobot fährt vom Anfangspunkt eine gewisse vorher eingestellte Distanz Δh (voreingestellt sind ca. 2cm) mit einer Linearbewegung (MOVL) nach oben, fährt mit einer PTP-Bewegung (MOVJ) auf eine Distanz Δh über den Endpunkt und von hier fährt der Roboter wieder mit einer Linearbewegung (MOVL) nach unten. Abbildung 5 zeigt diesen Bewegungsmodus schematisch.

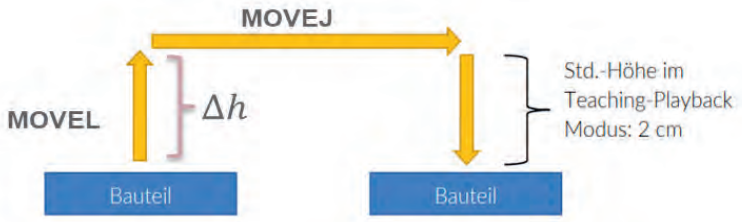


Abbildung 5: Schematische Darstellung einer JUMP-Bewegung

Programmieren mit Teaching & Playback

Für das Programmieren mit Teaching & Playback empfiehlt es sich zuerst eine Referenzfahrt mit „Home“ durchzuführen und den Endpunkt der Referenzfahrt als ersten Programmpunkt festzulegen. Dies gelingt durch das Klicken auf die Schaltfläche „+ Punkt“. Das Ganze sieht dann wie in Abbildung 6 aus. Die Bewegungsart kann für jeden Wegpunkt einzeln festgelegt werden, indem diese unter „MotionStyle“ geändert wird. Wird die Bewegungsart auf der Leiste für die Bewegungsarten eingestellt, so gilt diese übergeordnet für alle nachfolgend einprogrammierten Punkte. Es kann ebenfalls jedem Punkt, wie in Abbildung 6 zu sehen im rechten Feld der Programmieroberfläche ein Befehl zugeordnet werden, mit dem der Endeffektor aktiviert oder deaktiviert wird.

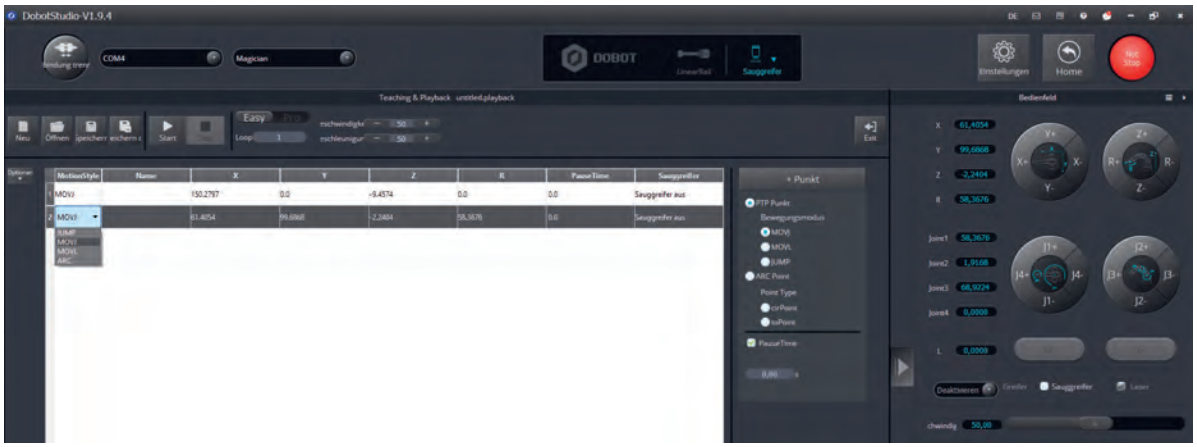


Abbildung 6: Schematische Darstellung einer Linearbewegung

Ist der erste Punkt festgelegt, so kann der Roboter entweder über das Bedienfeld oder über eine Taste, die mit einem Schloss gekennzeichnet ist und sich direkt vorne am Roboterarm befindet, zu einem nächsten Punkt bewegt werden.

Wichtig: Ist im Programm vorgesehen, dass der Endeffektor mit seiner vierten Achse (über R steuerbar) rotiert, so empfiehlt sich die Bewegung des Roboters über das Bedienfeld, da nur hierüber die jeweilige R-Position im übergeordneten Skript hinterlegt wird und so Wiederholgenauigkeit im Programm erreicht werden kann. Wird der Roboter über die Schloss-Taste programmiert, so kommt es hierbei zu Fehlern und die Rotation des Endeffektors ist bei jedem Programmablauf anders ausgerichtet.

Das gesamte Programm wird nun aus einer Aneinanderreihung von Punkten erstellt, die vom Roboter nacheinander abgefahren werden. Jedem Punkt kann ein Befehl für den Endeffektor oder ein Schaltbefehl für einen EIO-Ausgang hinzugefügt werden.

Teaching & Playback bietet eine einfache Art der Programmierung, besonders für die Anfangssituationen, in denen Schülerinnen und Schüler den Roboter kennenlernen. Bei Messebesuchen oder sonstigen Veranstaltungen zeigte sich, dass selbst Grundschüler*innen innerhalb weniger Minuten erste eigene Programme mit dieser Art der Programmierung schreiben können.

Home-Position ändern

Die Teaching & Playback Programmieroberfläche bietet die einzige Möglichkeit, die Home-Position zu ändern, ohne dass die Änderung nach dem Abschalten des Dobot wieder rückgängig gemacht wird. Dies gelingt durch das Einprogrammieren eines Punktes, welcher die neue Home-Position darstellen soll. Ist der Punkt einprogrammiert, so werden durch einen Rechtsklick auf den Punkt die folgenden in Abbildung 7 dargestellten Optionen angezeigt.
Unten ist die Option „Home setzen“ aufgeführt. Mit einem Klick auf diese Option erscheint ein Textfeld, in dem „Home Position erfolgreich gesetzt!“ steht. Jetzt fährt der Roboter am Ende der Referenzfahrt (Home) immer an die neu einprogrammierte Position.

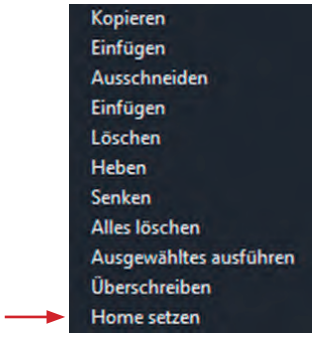


Abbildung 7: „Home setzen“

3.3 Schreiben & Zeichnen

Rene Egbers, Fabian Icken, Marcus Auf der Landwehr, Hochschule Osnabrück

Das Anwendungsfeld des Schreibens & Zeichnen in der Startoberfläche des DobotStudios ermöglicht es, mithilfe des Dobot zu schreiben und Bilder zeichnen zu lassen. Im Folgenden werden die notwendigen Schritte für die „Programmierung“ des Dobot genauer erläutert.

Für diese Anwendung wird der im Lieferumfang des Magicians enthaltene Stifthalter benötigt, der vorne in die Aufnahme für die Endeffektor montiert wird. In den Stifthalter selbst sollte anstelle des mitgelieferten Fineliners ein Bleistift gespannt werden, da der Fineliner bei längerem Kontakt mit dem Papier zu größeren Flecken führt. Bei der Montage ist zu beachten, dass der Stift nicht zu weit nach oben ragt, da er sonst bei einer Konturfahrt dicht an der Basis selbst die Freedrive-Taste des Roboters betätigt. Gleichzeitig sollte der Stift nicht zu hoch eingespannt werden, da sich dieser sonst aufgrund der Nachgiebigkeit des Stiftes Vibrationen ergeben, die zu unsauberen Zeichenergebnissen führen können.

Nach der erfolgreichen Montage von Stift und Stifthalter kann die Applikation Schreiben & Zeichen im Programm DobotSudio gestartet werden. Zu Beginn muss der Stift als Endeffektor ausgewählt werden. Dies erfolgt über das oben mittige Drop-Down-Menü (siehe Pfeil in der folgenden Abbildung).

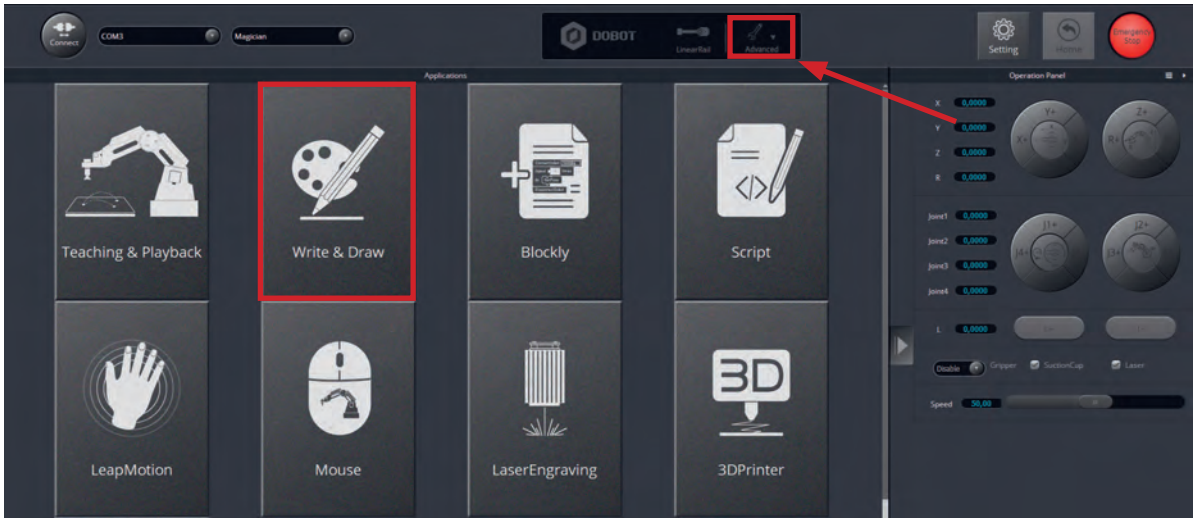


Abbildung 1: Starten der Applikation Schreiben & Zeichnen

Es öffnet sich die in der folgenden Abbildung zu sehende Programmoberfläche. Die aus zwei Halbkreisabschnitten bestehende Darstellung visualisiert den Arbeitsbereich des Magicians, also Punkte, die mit dem Stift erreicht werden können. Wenn der Magician mit der Software verbunden ist, wird die aktuelle Position des Endeffektors in dieser Darstellung mit einem größeren grauen Punkt angezeigt.

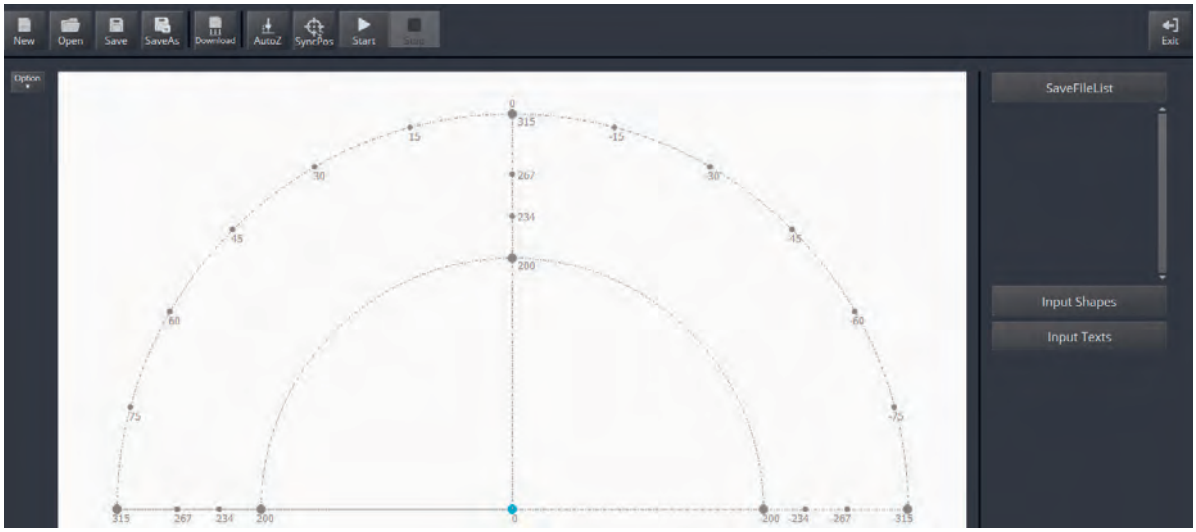


Abbildung 2: Applikation Schreiben & Zeichnen

Bevor mit dem Schreiben & Zeichnen begonnen werden kann, sollte ein Blatt Papier vor dem Magician auf dem Tisch befestigt werden. Für die Praxis kann die Papiergröße DIN-A3 empfohlen werden.

Bilder importieren

Damit der Magician Bilder nachzeichnen kann, müssen diese in die Zeichenfläche importiert werden. Dies erfolgt über die Schaltfläche *Öffnen*. Zu beachten ist, dass nur das Zeichnen von Vektorgrafiken (z. B. SVG-Dateien) unterstützt werden.

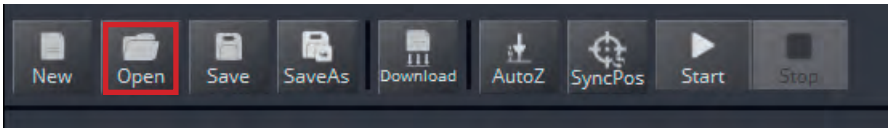


Abbildung 3: Schaltfläche in der Anwendung Schreiben & Zeichnen

Wurde nun ein Bild ausgewählt und geöffnet, erscheint dieses in der Zeichenfläche und kann dort skaliert und verschoben werden. Zu beachten ist, dass das Bild in den Arbeitsbereich, zwischen den beiden Radien, platziert werden muss, ansonsten färbt sich das Bild rot und es kann nicht gezeichnet werden. Alternativ zu den Bildern können auch die Symbole und Bilder aus der Dobot Software verwendet werden, indem auf „Input Shapes“ am rechten Rand geklickt wird und dort die passenden Symbole und Bilder ausgewählt werden.
Sollen andere Bildformate verwendet werden, ist dies nur möglich, indem diese zurvor umgewandelt werden. Dafür bietet DobotStudio die Funktion „Convert Bitmap“ am rechten Rand des Bildschirms. Anschließend kann das Bild mithilfe des Buttons „Plot to Main Scene“ in der Zeichenfläche platziert werden.

Neben Bilder und Symbole kann der Magician auch Text schreiben. Am rechten Bildschirmrand gibt es dafür die Eingabemaske „Input Text“. Durch klicken auf „OK“ wird der Text in die Zeichenfläche eingefügt. Auch hier ist wieder eine Skalierung und Verschiebung der Textelemente möglich. Neben der Schriftart kann der Schriftschnitt über die entsprechenden Buttons verändert werden.

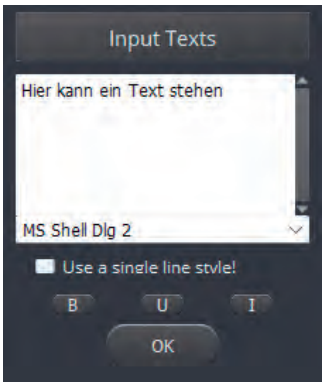


Abbildung 5: Input Text - Schreiben & Zeichnen

Ist der Arbeitsbereich mit den gewünschten Bildern, Symbolen und Texten angereichert worden, kann der Schreibvorgang gestartet werden. Zunächst muss dafür einmalig die Referenzhöhe eingelernt werden. Dafür muss der Roboterarm über die Steuerflächen oder händisch mithilfe der Freedrive-Taste so weit nach unten auf das Papier gefahren wird, sodass die Stiftspitze das Blatt mit einem leichten Druck berührt.

Damit die Referenzhöhe gesetzt wird, muss anschließend die Taste „AutoZ“ betätigt werden. Ist dies geschehen, kann der Schreibvorgang über die Start-Taste gestartet werden. Während des Schreibvorganges kann dieser auch gestoppt oder abgebrochen werden. Zeigt sich, dass die Stiftspitze nicht in allen Bereichen der Zeichenfläche die Papierebene berührt, kann dies durch eine erneute Kalibrierung mithilfe des Auto-Levelling-Tools behoben werden. Eine schrittweise Anleitung hierzu kann in den Einstellungen von DobotStudio gefunden werden.

Abbildung 6: AutoZ und Start in der Anwendung Schreiben & Zeichnen

3.4 Grundlagen in Blockly

Dani Hamade, Jan Landherr, Carl von Ossietzky Universität Oldenburg

3.4.1 Pick & Place mit Variablen

In diesem Abschnitt soll eine kurze Einführung in die blockbasierte Programmierung des Dobots erfolgen. Hierzu kann eine erste Übung durchgeführt werden, in welcher ein Würfel, welcher sich in der Ausgangsposition A befindet, zur Endposition B befördert werden soll (siehe Abbildung 1).

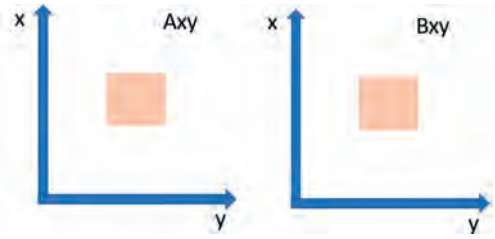


Abbildung 1: Erste Übung: Würfel Axy soll nach Bxy befördert werden

Ein Ansatz, welcher hier verfolgt werden kann, ist die Definition der Positionen durch Variablen. Hier werden die Koordinaten für die Position A beispielsweise mit Aufnahme X, Y und Z definiert. Punkt A wäre somit definiert und es muss lediglich noch ein Ablagepunkt B definiert werden. Auch hierzu werden Variablen definiert, die mit Ablage X, Y und Z beschriftet werden. Für eine bessere Übersichtlichkeit können alle Variablen innerhalb eines Funktionsblocks, welcher mit „Variablen“ beschriftet wird, abgelegt werden. Hierzu kann man wie folgt vorgehen. Man navigiert in den Programmblöcken bei Blockly in DobotStudio auf den Reiter „Funktionen“ und zieht einen einfachen Funktionsblock in das Programm (siehe Abbildung 2).

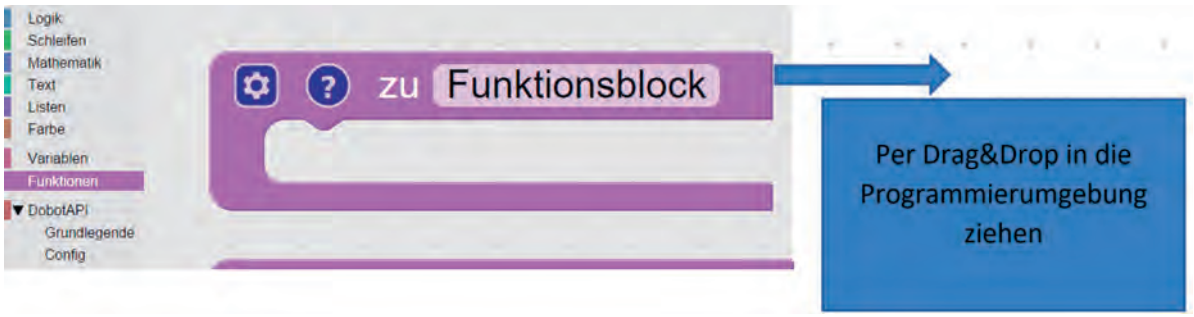


Abbildung 2: Wählen eines Funktionsblocks in DobotStudio Blockly

Anschließend benennt man diesen durch doppeltes Klicken auf das Schriftfeld (Doppelklick auf den Begriff „Funktionsblock“) um in „Variablen“ (siehe Abbildung 3).

Abbildung 3: Umbenennen von Funktionsblöcken

Nachdem dieser Schritt abgeschlossen ist, kann mit der Definition der Variablen begonnen werden. Hierzu wählt man in den Programmblöcken den Reiter „Variablen“ an. Anschließend zieht man einen Block „Schreibe Element“ wieder per Drag&Drop in die Programmieroberfläche (siehe Abbildung 4). Der Befehl „Schreibe Element“ erlaubt es nun, neue Variablen zu definieren.

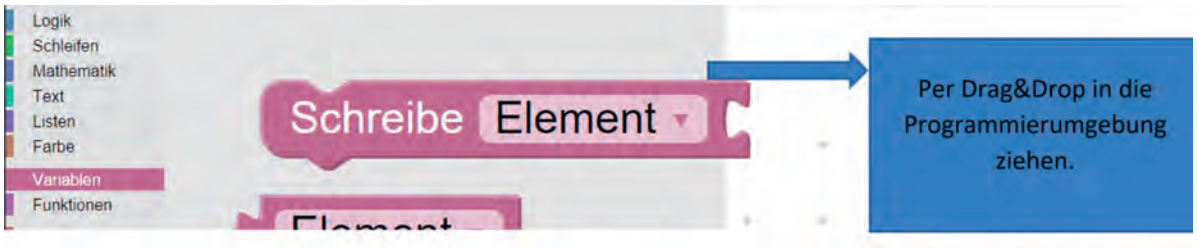


Abbildung 4: Wählen eines Variablenblocks

Man zieht den Block „Schreibe Element“ hierzu nun in den Funktionsblock „Variablen“, um mit der Definition von Variablen zu beginnen (siehe Abbildung 5). Damit die Variable nun individuell definiert werden kann, öffnet man das Dropdown Menü bei dem Block „Schreibe Variable“ und klickt auf „Neue Variable…“ (siehe Abbildung 6).

Abbildung 5: Definition neuer Variablen

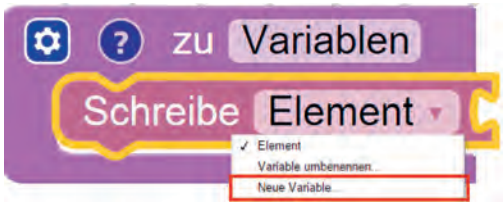


Abbildung 6: Anlegen neuer Variablen

Es öffnet sich nun ein Schriftfeld, in dem man der Variable eine Bezeichnung geben kann. Für die Aufgabenstellung werden insgesamt sechs Variablen benötigt, nämlich für jede Koordinate eine (Aufnahme X, Y und Z und Ablage X, Y und Z). Die erste Variable wird demnach „AufnahmeX“ benannt.

Durch Drücken auf „OK“ wird die Variable in der Ansicht umbenannt (siehe Abbildung 7).

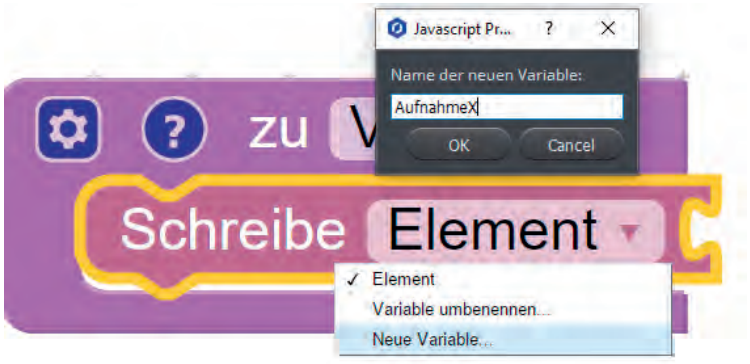


Abbildung 7: Benennen von Variablen

Es öffnet sich nun ein Schriftfeld, in dem man der Variable eine Bezeichnung geben kann. Für die Aufgabenstellung werden insgesamt sechs Variablen benötigt, für jede Koordinate eine (Aufnahme X, Y und Z und Ablage X, Y und Z). Die erste Variable wird demnach „AufnahmeX“ benannt.
Durch Drücken auf „OK“ wird die Variable in der Ansicht umbenannt (siehe Abbildung 7).

Diesen Schritt wiederholt man nun für alle notwendigen Variablen. Hierzu kann man entweder wieder über den Programmblock „Variablen“ neue Bausteine in die Funktion ziehen (siehe Abbildung 4) oder über Copy&Paste die bereits definierte Variable kopieren und einfügen (hierzu die Variable anwählen, sodass diese gelb aufleuchtet, dann Strg &C und Strg&V.
HINWEIS: Die eingefügte Variable muss über das DropDown Menü wieder überschrieben werden!). So sieht es aus, wenn alle sechs Variablen definiert sind:

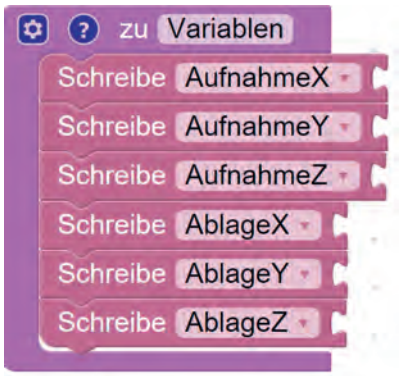


Abbildung 8: Ansicht Funktionsblock Variablen

Den einzelnen Variablen müssen nun noch entsprechende Koordinaten (Werte) zugeordnet werden. Hierzu wählt man bei den Programmblöcken den Reiter „Mathematik“ aus und zieht eine einfache Zahl (siehe Abbildung 10) per Drag&Drop an die einzelnen Variablen im Funktionsblock (Hinweis: Die Zahlenblöcke müssen an die einzelnen Variablen „andocken“). Diesen Vorgang wiederholt man für alle sechs Variablen. Ist dieser Schritt erfolgt, kann man nun die passenden Werte der Koordinaten übertragen. Hierzu bewegt man den Dobot-Arm zu der entsprechenden Position (Aufnahme X, Y und Z bilden in der Aufgabenstellung den Punkt A). Man drückt und hält hierzu den Entriegelungsknopf am Arm des Dobots und führt ihn an die Würfeloberfläche. Anschließend kann man die Koordinaten am Steuerkreuz auf der rechten Seite des Bildschirms ablesen und übertragen (siehe Abbildung 9). Den Vorgang wiederholt man für die Ablagekoordinaten, sodass allen Variablen Werte zugeordnet werden können (siehe Abbildung 11).

Abbildung 9: Ablesen der Koordinaten am Steuerkreuz

Abbildung 10: Ablesen der Koordinaten am Steuerkreuz

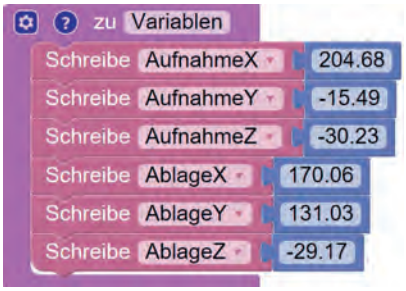


Abbildung 11: Zuweisung aller Koordinaten

Sind nun alle Variablen definiert, so kann mit der Programmierung des Bewegungsablaufs begonnen werden. Hierzu wird zunächst wieder ein Funktionsblock angelegt, welcher mit „Bewegung“ benannt wird (Der Vorgang wurde in den Abbildungen 2-3 beschrieben).

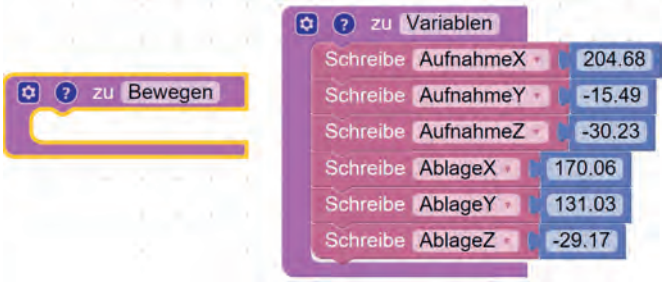


Abbildung 12: Anlegen des Funktionsblocks für die Bewegungsabfolge

Für die Stapel- und Ablagelogik hat sich der Jump-Befehl als vorteilhaft erwiesen, weshalb dieser für die Bewegungsabläufe genutzt wird. Hierzu klickt man in den Programmblöcken auf die DobotAPI und dort auf „Antrag“. Unter Antrag wählt man dann den „Jump To“ Befehl aus und zieht diesen per Drag&Drop in den neuen Funktionsblock „Bewegen“ (siehe Abbildung 13).

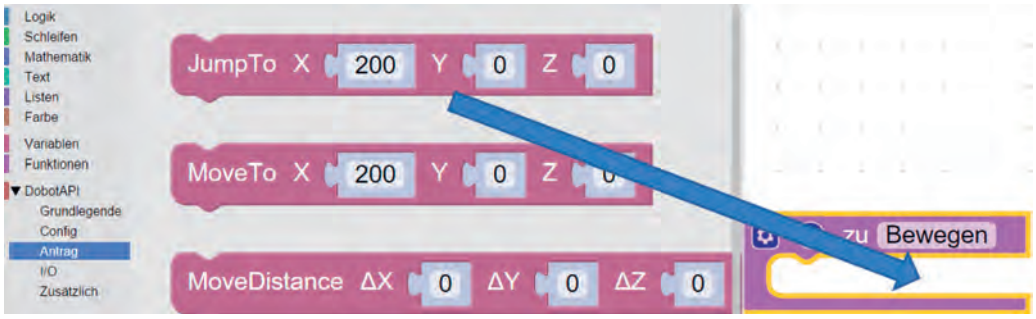


Abbildung 13: Auswahl der Bewegungsart

Jetzt muss man sich Gedanken über die Fahrabfolge machen. Der Dobot soll in diesem Fall zunächst den Aufnahmepunkt A anfahren. Die Koordinaten des Aufnahmepunktes wurden unter dem Funktionsblock „Variablen“ bereits definiert und müssen nun noch auf den Jump To Befehl übertragen werden. Hierzu geht man in den Programmblöcken auf den Reiter Variablen und zieht die entsprechende Variable (hier AufnahmeX, AufnahmeY und AufnahmeZ) in die einzelnen Felder des Jump To Befehls (siehe Abbildung 14-15).

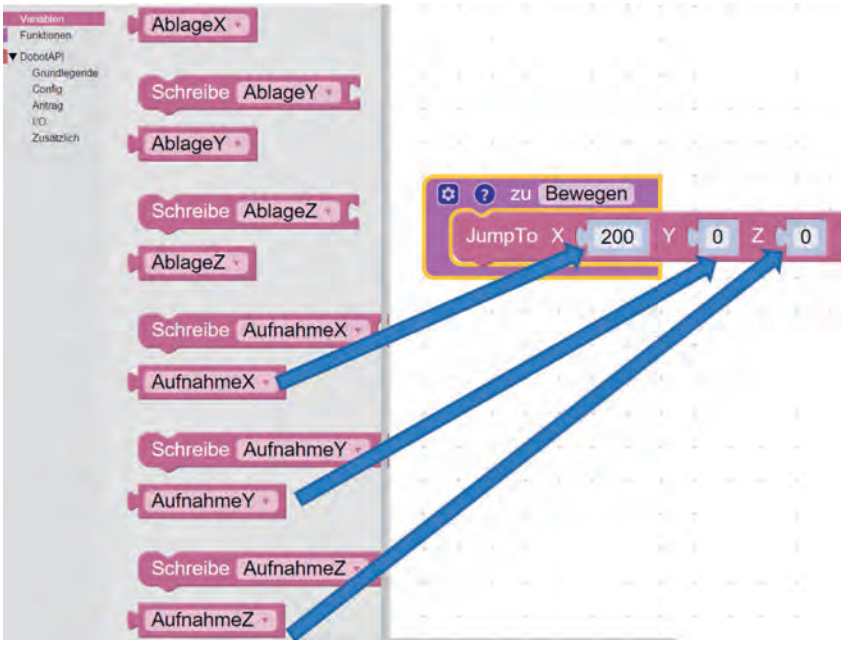


Abbildung 14: Zuweisung Variablen zu Punkten

Sind alle Variablen vollständig hinzugefügt, ergibt sich folgende Ansicht:

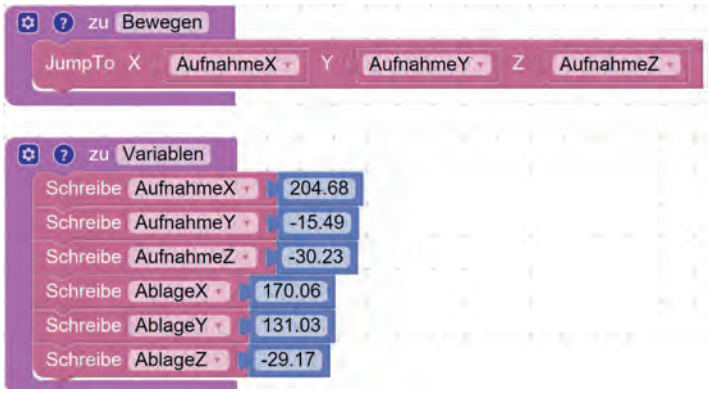


Abbildung 15: Vollständige Variablenzuweisung in einem Jump Befehl

Nun muss die Ablagelogik weiter durchdacht werden. Der Dobot würde sich jetzt an der Aufnahmeposition befinden. Dort soll der Dobot mithilfe des Sauggreifers einen Würfel heranziehen. Hierzu wird in der DobotAPI unter „Antrag" der Befehl Sauggreifer AN/AUS gewählt und per Drag&Drop unter den Jump To Befehl in unserer Bewegungsfunktion gesetzt (siehe Abbildung 16-17).

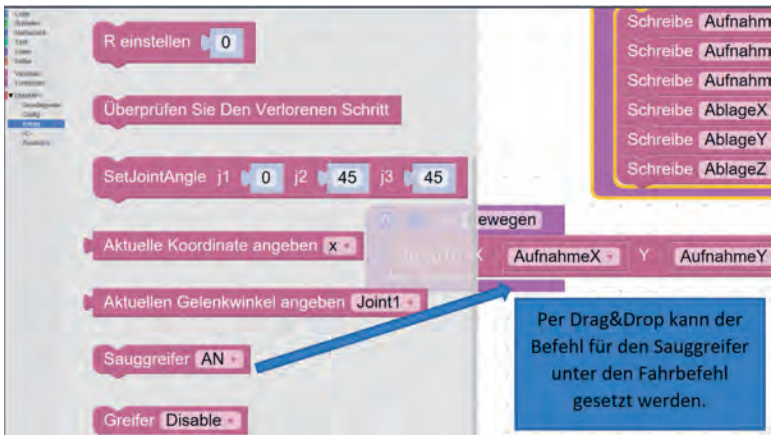


Abbildung 16: Implementation des Saugnapfes

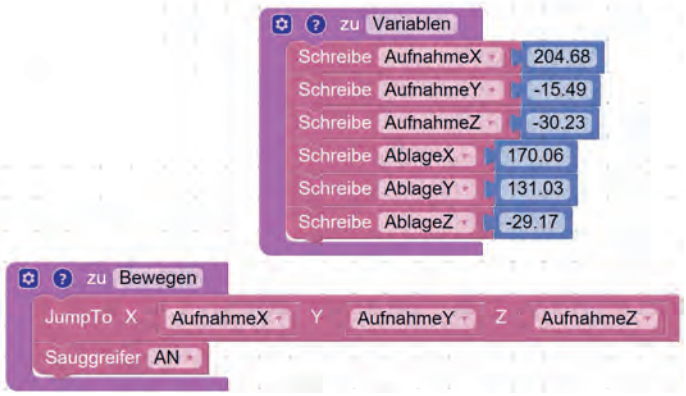


Abbildung 17: Einschalten des Saugnapfes

Nachdem der Sauggreifer eingeschaltet ist, soll der Dobot den Würfel zur Ablageposition bewegen. Hierzu wird wieder ein Jump To Befehl benötigt und unterhalb des Sauggreifer-Befehls im Funktionsblock positioniert. Die Koordinaten werden wieder durch die Variablen definiert, nur, dass dieses Mal die Ablagekoordinaten gewählt werden müssen (AblageX, AblageY und AblageZ). Das Ergebnis ist in Abbildung 18 zu sehen.

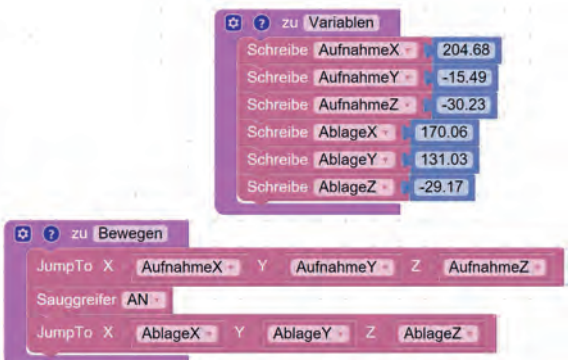


Abbildung 18: Einfügen der Ablageposition

HINWEIS: Wenn man sich viel Arbeit ersparen möchte, kann man den ersten Jump To Befehl (Aufnahmeposition) anklicken, kopieren (Strg&C) und über Strg&V wieder einfügen. Hier muss nur beachtet werden, dass die Variablen geändert werden müssen. Hierzu kann das Dropdown Menü geöffnet und die richtigen Variablen ausgewählt werden.
Der Dobot hat den Würfel nun zur Ablageposition befördert, muss diesen allerdings noch loslassen. Hierzu wird erneut der Sauggreifer Befehl implementiert, allerdings wird dieser dieses Mal ausgeschaltet. Damit der Sauggreifer ausgeschaltet werden kann, muss man im Dropdown Menü von „AN“ zu „AUS“ wechseln (siehe Abbildung 19).

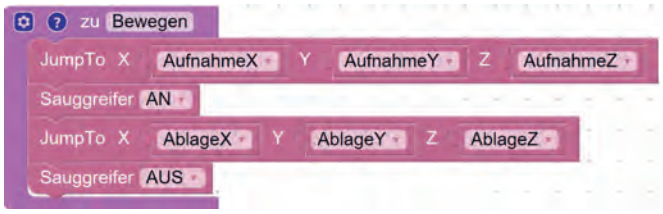


Abbildung 19: Ausschalten des Saugnapfes

Das (fast) fertige Programm für die erste Übung sieht schlussendlich folgendermaßen aus:

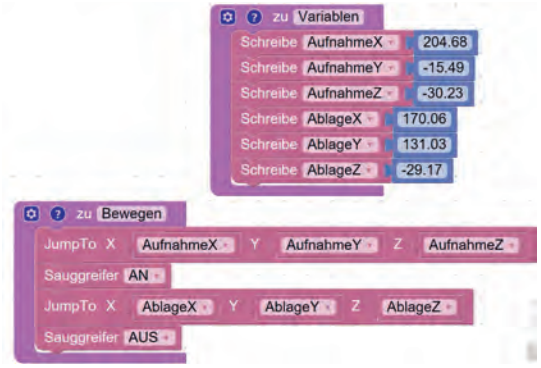


Abbildung 20: (Fast) Fertiger Sketch

Warum fast?

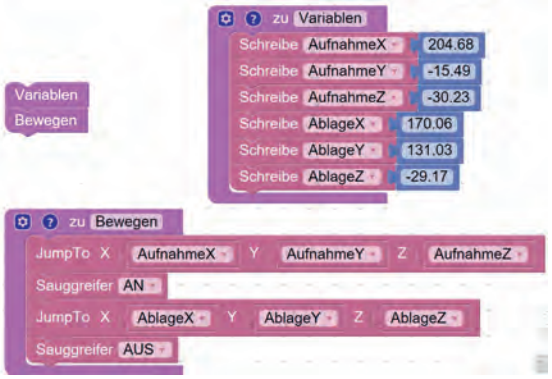


Abbildung 21: Abruf der Unterprogramme

Die Programme wurden hier in Funktionsblöcken definiert. Diese Funktionsblöcke stellen Unterprogramme dar, die noch im Hauptprogramm zu berücksichtigen sind. Damit dies erfolgen kann, gehen Sie in den Programmblöcken wieder auf den Reiter „Funktionen“. Dort erscheinen nun die zwei soeben bearbeiteten Unterprogramme „Variablen“ und „Bewegen“. Diese zieht man per Drag&Drop in die Programmieroberfläche und reiht sie aneinander (siehe Abbildung 21). Nun führt der Dobot (sobald der Start Button betätigt wurde) die beiden Unterprogramme „Variablen“ und „Bewegung“ aus.

3.4.2 Ablagelogiken

In der zweiten Übung sollen drei vertikal nebeneinander ausgerichtete Würfel um eine X-Komponente verschoben werden. Hierzu ist eine sogenannte Ablagelogik erforderlich, welche im Folgenden beschrieben wird.

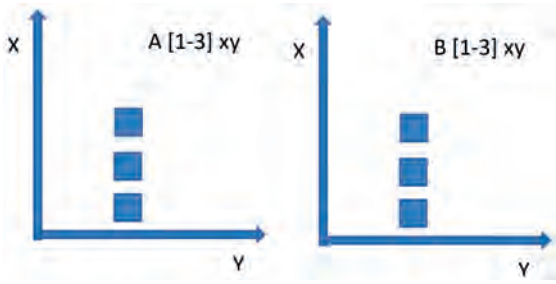


Abbildung 22: Übung zur Ablagelogik

Zunächst ist zu sagen, dass die grundlegenden Variablen aus Übung eins beibehalten werden können. Die Aufnahme- und Ablagepositionen sollen in dieser Übung nicht über das stumpfe Hinzufügen neuer Punkte definiert werden. Vielmehr soll es darum gehen, eine Ablagelogik zu entwickeln, mit der es möglich wird, die Ausgangsvariablen mit jedem Durchlauf zu verändern. Insgesamt sind in dieser Übung drei Würfel zu bewegen, weshalb der Einsatz einer Schleife sinnvoll erscheint. Damit eine Schleife hinzugefügt werden kann, geht man unter den Programmblöcken auf den Reiter „Schleifen“. Wir haben insgesamt drei Würfel, die bewegt werden müssen, weshalb eine Schleife mit n-facher Wiederholung gewählt wird (wie oft wiederholt werden soll, ist individuell anpassbar, indem man den Wert verändert). Wir ziehen die Schleife per Drag&Drop in unseren Funktionsblock aus Übung 1 und ändern den Wert in eine drei, sodass die Schleife drei Mal durchgeführt wird (siehe Abbildung 23).

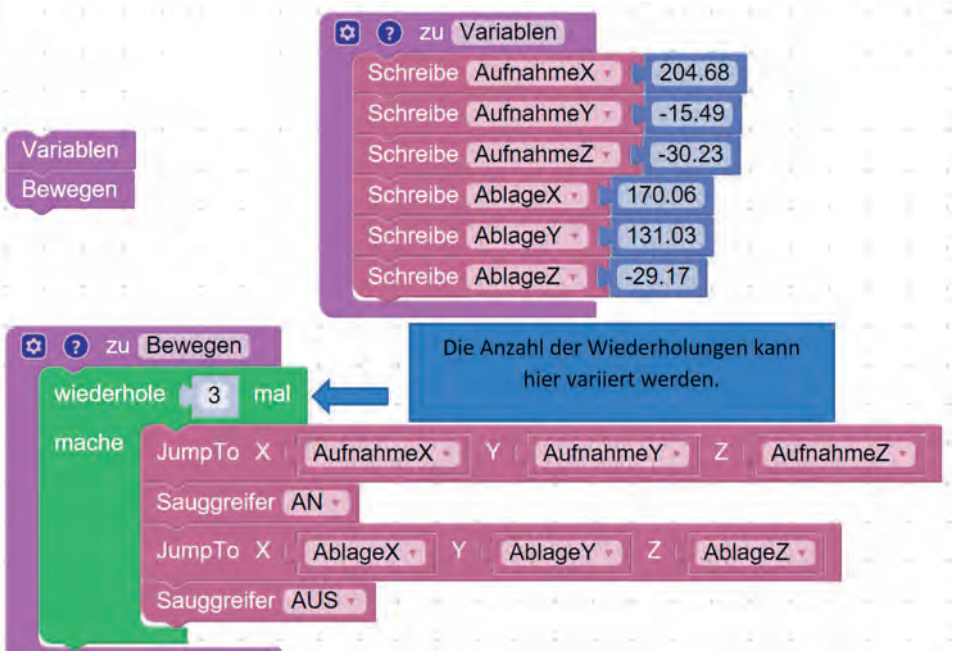


Abbildung 23: Implementation einer Schleife

Würde man den Befehl jetzt ausführen, so würde der Dobot allerdings immer wieder an die gleichen Positionen fahren. Der Dobot soll aber mit jedem Durchgang um einen bestimmten Wert in der X- Achse variieren. Der Startpunkt ist hierbei der rosafarbene Würfel aus der Aufgabenstellung (siehe Abbildung 22). Das bedeutet, dass der Dobot die Aufnahme- und Ablageposition um einen bestimmten X-Wert erhöhen muss, sobald die Schleife durchlaufen ist. Der Verschiebungswert ergibt sich hierbei aus einer gesamten Würfelbreite (wenn dazwischen eine Lücke gelassen wurde, muss die Differenz durch das Anfahren von zwei Positionen bestimmt werden). Damit die Positionen nun neu berechnet werden können, muss im Funktionsblock „Variablen" eine neue Variable mit dem Namen „VerschiebungX" angelegt werden, welche mit dem Wert der Verschiebung versehen wird (siehe Abbildung 24).

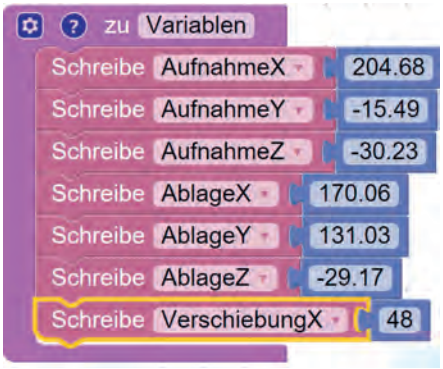


Abbildung 24: Hinzufügen der Verschiebung

Damit der Dobot die Verschiebung nun im Ablauf berücksichtigen kann, müssen die Aufnahme- und Ablagepositionen X innerhalb der Schleife um den Wert der VerschiebungX erweitert werden. Hierzu wird in den Programmblöcken unter dem Reiter „Mathematik" die Summenfunktion implementiert (siehe Abbildung 25).

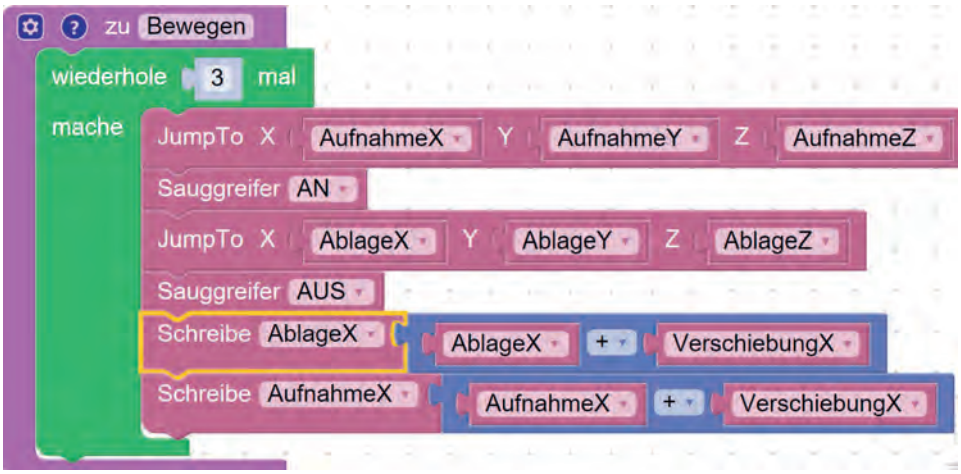


Abbildung 25: Implementierung der Summenfunktion zur Ablagelogik

In einer dritten Übung sollen nun auch wieder drei Würfel bewegt werden. Die drei nebeneinanderliegenden Würfel sollen dieses Mal allerdings aufeinandergestapelt werden (siehe Abbildung 26).

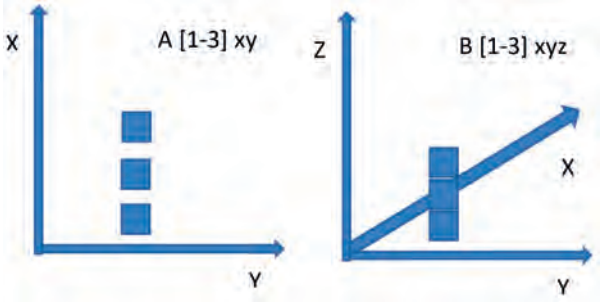


Abbildung 26: Übung 3: Aufeinanderstapeln

Für die Aufnahme der Würfel kann wieder dieselbe Logik verwendet werden wie in Übung zwei, da die Würfel wieder vertikal nebeneinander liegen. Die Ablagelogik hingegen verändert sich. Es ändert sich nicht mehr der AblageX-Wert, sondern die Z-Komponente, da übereinandergestapelt werden soll. Hierzu wird zunächst erneut eine neue Variable angelegt, welche die Höhendifferenz angibt (hierzu entweder die Würfelhöhe messen oder zwei Würfel übereinanderstapeln und die Differenz berechnen). Die Variable wird „VerschiebungZ“ genannt (siehe Abbildung 28). Die Variable „VerschiebungX“ kann bestehen bleiben, da diese für die Aufnahmeposition wieder benötigt wird.

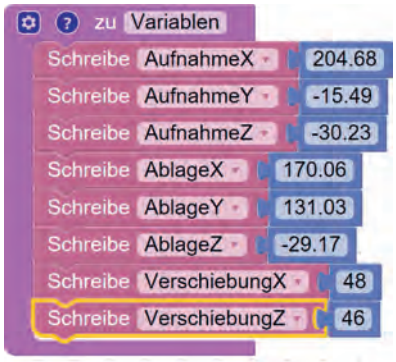


Abbildung 27: Definition der Verschiebung

Anstelle der Änderung der „AblageX“ im Bewegungsablauf verändert sich die „AblageZ“. Hierzu im Dropdown Menü die Elemente verändern, sodass sich folgende Lösung ergibt:

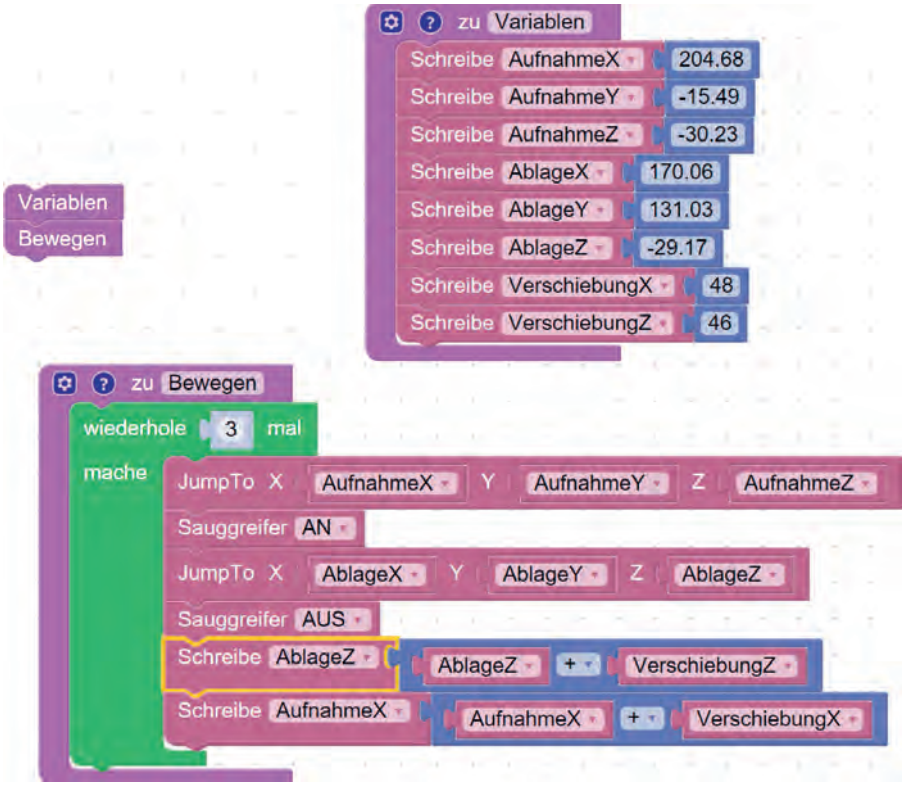


Abbildung 28: Lösung Übung 3

3.5 Lichtschranke, Farbsensor und Sortierung

Dani Hamade, Carl von Ossietzky Universität Oldenburg

3.5.1 Einbindung des Förderbandes

Die erste Übung besteht hier darin, das Förderband ordnungsgemäß anzuschließen und einen Sketch zu entwickeln, mit dem das Förderband auf Funktion überprüft werden kann. Das Förderband wird hier an den Anschluss „STEPPER1“ angeschlossen. Nun folgt die Funktionsprüfung. Zunächst kann man hierzu eine Funktion anlegen, die den Namen „Funktionsüberprüfung Förderband“ (siehe Abbildung 29) erhält (wie man Funktionen anlegt, kann in Kapitel eins nachgelesen werden).

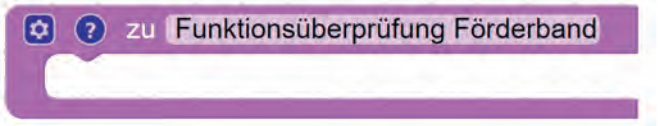


Abbildung 29: Funktion zur Funktionsüberprüfung des Förderbandes

Ein Ansatz, der nun zur Funktionsüberprüfung verfolgt werden kann, ist über eine Endlosschleife. Hierzu nimmt man den Block „wiederhole solange“ aus Blockly (DOBOTSTUDIO) unter „Schleifen“ und setzt ihn unter die Funktion (siehe Abbildung 30).

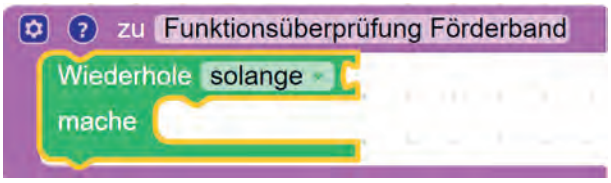


Abbildung 30: Implementation der Schleife in die Funktion

Der Ausdruck „wiederhole solange“ (in Python „while“) muss, damit daraus eine Endlosschleife wird, an eine Bedingung geknüpft werden. Hier kommt der boolesche Ausdruck „wahr“ (in Python „true“) zum Einsatz (bei Blockly zu finden unter „Logik“). Verknüpft man diese beiden Bestandteile, so erhält man eine Endlosschleife (siehe Abbildung 31).

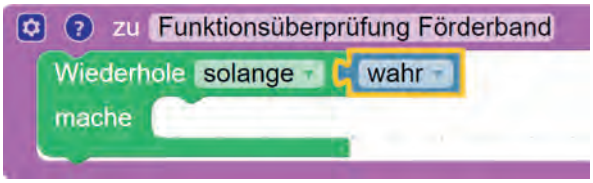


Abbildung 31: Implementation Endlosschleife

Der nächste Schritt ist, dass das Förderband eingebettet wird. Mit dem hier gewählten Ansatz wird das Förderband permanent laufen, sodass man die Funktionsweise überprüfen kann. Die Befehle für das Förderband (und auch andere Peripherie) sind in der DOBOTAPI unter „Zusätzlich“ zu finden. Wichtig ist, dass der richtige Anschluss für den Schrittmotor des Förderbandes im Auswahlmenü des Befehls gewählt wird. Im Fall von Abbildung 32 ist der Schrittmotor an den Anschluss „Stepper 1“ des DOBOTs angeschlossen (siehe Abbildung 32). Zusätzlich dazu lässt sich auch die Geschwindigkeit des Schrittmotors einstellen (Einheit mm/s). Zu beachten ist hierbei, dass die maximale Geschwindigkeit mit 120 mm/s angegeben ist. Die Erfahrung hat gezeigt, dass es ab 100 mm/s bereits zu Problemen kommen kann.
Zum Schluss muss die Funktion nur noch ausgeführt werden (siehe Abbildung 33):

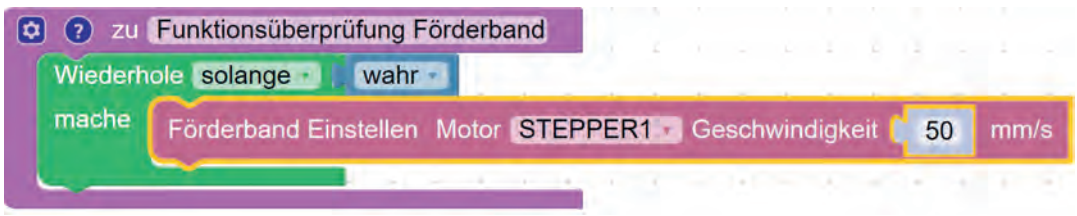


Abbildung 32: Einbettung des Förderbandes in die Endlosschleife

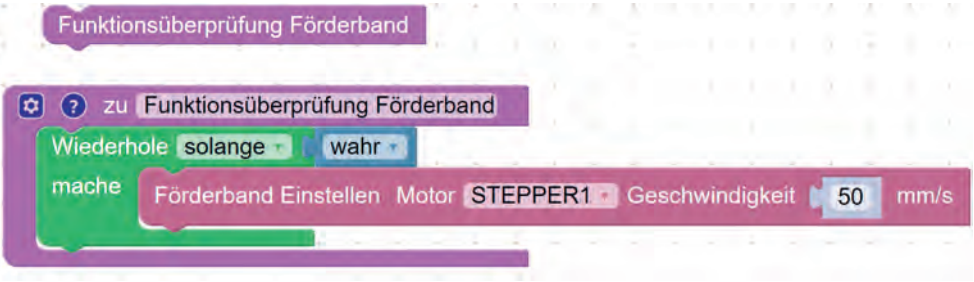


Abbildung 33: Fertiger Sketch zu Übung 1a

In einer weiteren Übung soll nun ein Sketch entwickelt werden, bei dem das Förderband in einem Rhythmus von fünf Sekunden ein- und ausgeschaltet wird. Zu beachten ist hierbei, dass dies in einer Dauerschleife stattfindet. Der Ansatz zur Umsetzung einer Dauerschleife wurde in der Lösung zur ersten Übung mit dem Förderband bereits dargelegt und kann auf diese Übung übertragen werden (siehe Abbildung 34).

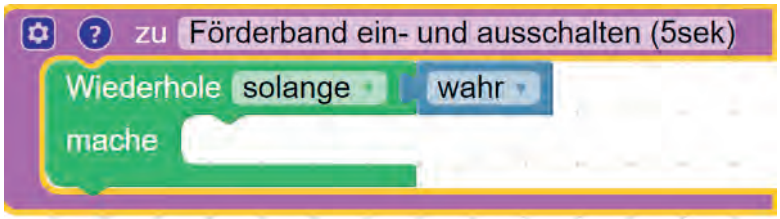


Abbildung 34: Dauerschleife für den Rythmus aus Übung 1b

Wie das Förderband angesprochen werden kann, ist aus der ersten Übung abzuleiten. Dieses Mal muss aber zusätzlich dazu eine Verzögerungszeit implementiert werden. Damit der Förderprozess pausiert werden kann, ist die Geschwindigkeit auf 0 mm/s einzustellen (siehe Abbildung 35).

Abbildung 35: Lösung Übung 1b

Nun soll der Dobot Würfel auf dem Förderband ablegen, welche dann vom Förderband abtransportiert werden. Hierzu sind zunächst die Positionen für die Aufnahme und die Ablage als Variablen zu definieren (siehe Abbildung 36). Zusätzlich dazu, muss für diese Übung eine Aufnahmelogik eingebettet werden, weshalb es sinnvoll ist, zusätzlich die Breite eines Würfels zu definieren (oder die Würfelhöhe, je nach Aufnahmelogik). Falls die Vorgehensweise nicht mehr bekannt ist, so kann in Kapitel eins nachgeschaut werden.

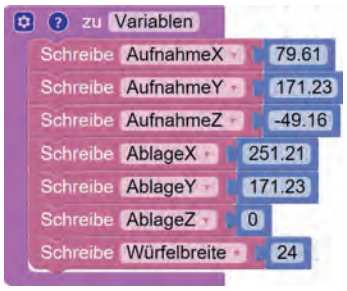


Abbildung 36: Positionsdefinitionen (individuell anpassen!)

Eine Endlosschleife wie in den Übungen zuvor würde bei der Aufnahmelogik zu Komplikationen führen, da der Arbeitsbereich des DOBOTs an seine Grenzen stoßen würde. Aus diesem Grund ist eine Schleife mit vorab definierter Wiederholungsanzahl zu wählen (in diesem Fall sollen drei Durchläufe stattfinden). Mit Einbettung der Auf- und Abnahme der Würfel sowie der Ablagelogik ergibt sich der in Abbildung 37 abgebildete Programmaufbau (siehe Abbildung 37). Zunächst fährt der DOBOT zu der Aufnahmeposition mittels JUMP-Befehl und schaltet dort den Sauggreifer ein, um einen Würfel aufzunehmen. Anschließend springt der DOBOT mit angesaugtem Würfel auf das Förderband. Damit es beim Anfahren des Förderbandes zu keiner Kollision mit dem Saugnapf und dem Würfel kommt, fährt der DOBOT um 10mm nach oben (Δz=10). Der Würfel ist abgelegt und das Förderband kann losfahren, was mit dem nächsten Befehl geschieht (die gewählte Geschwindigkeit beträgt hier 50mm/s). Das Förderband fährt hier insgesamt fünf Sekunden und hält dann wieder an. Als nächstes ist nur noch die Aufnahmelogik zu implementieren. In diesem Fall sind die Würfel horizontal nebeneinander gelagert, sodass die Y-Koordinate der Aufnahmeposition um eine Würfelbreite erweitert werden muss. Nach Durchführung der drei Durchläufe sind dementsprechend alle drei nebeneinander gelegten Würfel nacheinander abtransportiert (hier ein Video zum Programmablauf: https://youtube.com/shorts/74nZLB-nO4U?feature=share).

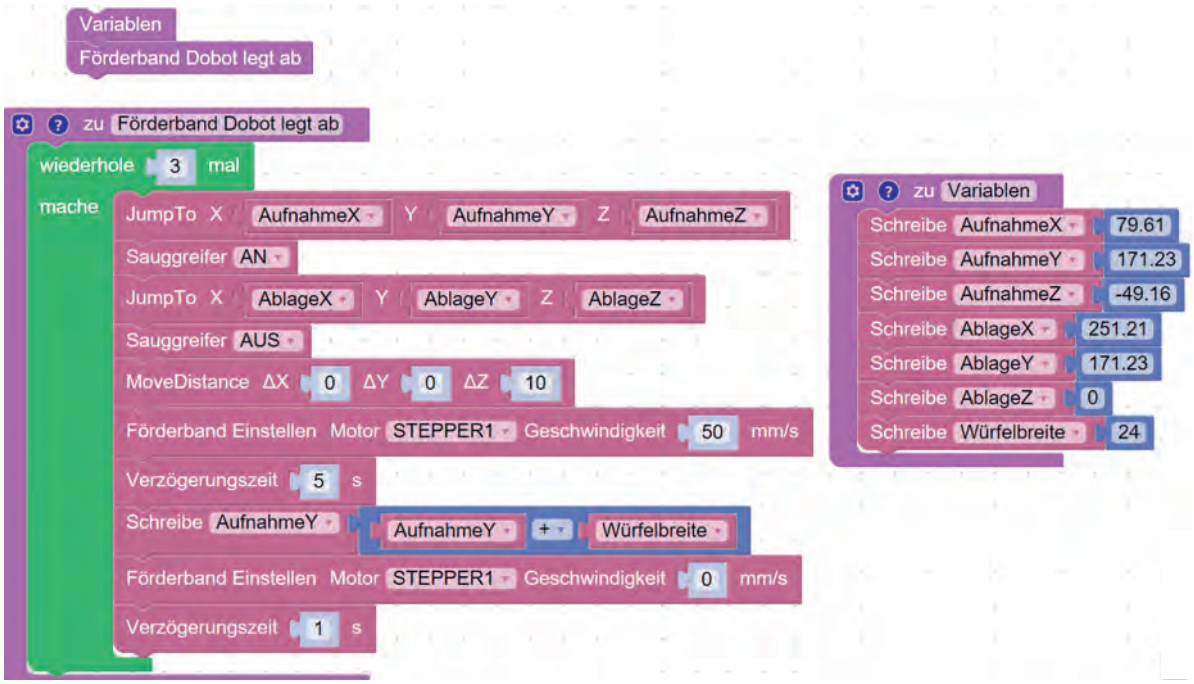


Abbildung 37: Lösung für die Übung 1c mit Aufnahmelogik

3.5.2 Einbindung der Lichtschranke

In diesem Abschnitt geht es um den ordnungsgemäßen Anschluss der Lichtschranke. In den seitlichen Schienen des Förderbandes befinden sich Gewindeplatten mit einem M6 Innengewinde (siehe Abbildung 38).

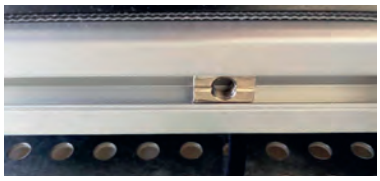


Abbildung 38: Verstellbare Gewindeplatte am Förderband

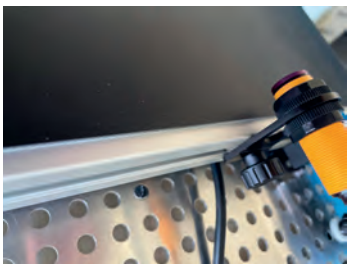


Abbildung 39: Befestigung der Lichtschranke an der Gewindeplatte des Förderbandes

Abbildung 40: Anschluss Lichtschranke DOBOT

An diesen kann die Lichtschranke mit einer M6 Schraube befestigt werden (siehe Abbildung 39). Nach ordnungsgemäßer Befestigung kann die Lichtschranke an den DOBOT angeschlossen werden (richtigen Port beachten! Siehe Abbildung 40).

In der Übung soll die Lichtschranke nun auf Funktion überprüft werden. Hierzu soll eine Printausgabe bei Blockly erstellt werden, aus der hervorgeht, ob ein Objekt erkannt wird oder nicht. Ein Ansatz ist hier, dass wieder eine Funktion mit dem Titel „Printausgabe Lichtschranke“ erstellt wird. In dieser kann die Lichtschranke zur besseren Übersicht zunächst als Variable definiert werden. Die entsprechenden Befehle für die Lichtschranke sind in der DOBOT API unter „Zusätzlich“ zu finden. Die Funktion soll fortlaufend überprüft werden, weshalb wieder eine Endlosschleife gesetzt wird (siehe Abbildung 41).

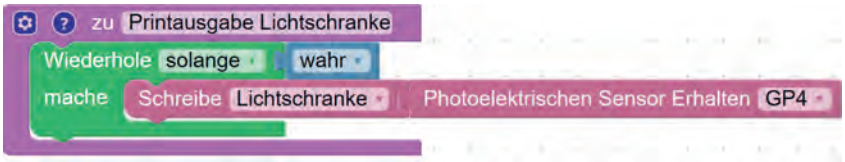


Abbildung 41: Definition der Lichtschranke als Variable (Auswahl des richtigen Ports beachten!)

Abbildung 42: Einschalten der Lichtschranke

Nun muss die Lichtschranke initiiert werden, was durch das Einschalten dieser innerhalb des Sketches geschieht. Hierzu ist der Befehl wie in Abbildung 42 zu ergänzen. Auch hier ist wieder auf die richtige Version und die korrekte Auswahl des Ports zu achten. Das Ziel ist es nun, eine Printausgabe zu erhalten, WENN ein Objekt erkannt wird. ANSONSTEN soll eine Printausgabe erfolgen, aus der hervorgeht, dass kein Objekt erkannt wird.

Hier bietet sich, wie aus der Betonung der Bedingungen schon hervorgeht, eine WENN DANN SONST Bedingung an. Diese ist bei Blockly unter „LOGIK“ zu finden. Die SONST Bedingung lässt sich über das Zahnrad hinzufügen (siehe Abbildung 43).

Abbildung 43: Implementation der Bedingungen

Damit die Printausgabe erfolgen kann, sind nun nur noch die Bedingungen zu definieren. Die Lichtschranke gibt einen Integer-Wert zurück, weshalb die Bedingung sein muss, dass WENN ein Objekt erkannt wird, die Variable Lichtschranke gleich eins sein muss. Über den Reiter „Text“ lässt sich dann eine Ausgabe implementieren, in die geschrieben werden kann „Objekt erkannt“ (siehe Abbildung 44). Wird kein Objekt erkannt, so soll in diesem Beispiel folgende Ausgabe erfolgen: „kein Objekt“. Der fertige Sketch zur Funktionsüberprüfung nimmt somit folgende Gestalt an:

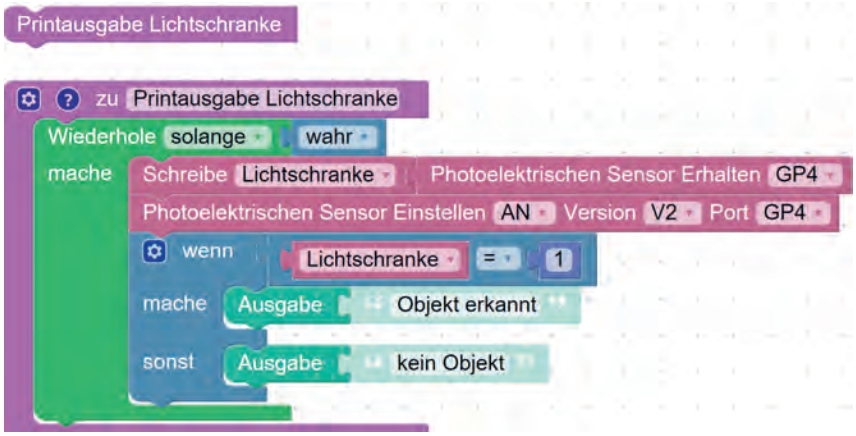


Abbildung 44: Fertiger Sketch zur Funktionsüberprüfung der Lichtschranke

Die dazugehörige Printausgabe erscheint nach dem Starten des Programmes im Protokoll auf der rechten Seite (siehe Abbildung 45).

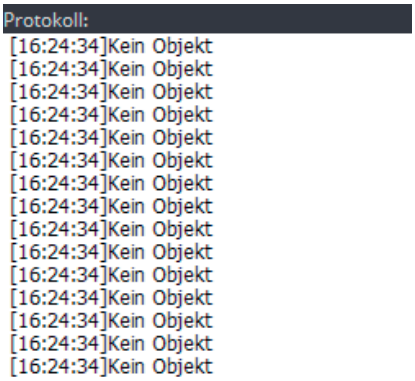


Abbildung 45: Protokoll zur Printausgabe der Objekterkennung

Nun, da die Lichtschranke funktioniert soll eine weitere Übung folgen, in welcher auch das Förderband eingebunden wird. In dieser Übung soll der zuvor erstellte Sketch angewendet werden, sodass das Förderband für zwei Sekunden stoppt, sobald ein Würfel erkannt wurde und diesen nach Ablauf der zwei Sekunden abtransportiert. Der Grundaufbau aus der Übung zur Lichtschranke kann übernommen werden und um die Befehle zur Ansteuerung des Förderbandes ergänzt werden. Der hier verfolgte Ansatz (siehe Abbildung 46) ist, dass die Geschwindigkeit des Förderbandes für zwei Sekunden auf 0 mm/s gesetzt wird, sobald ein Objekt erkannt wird. Anschließend soll das Förderband mit gewohnter Geschwindigkeit von 50 mm/s weiterfahren. Wird kein Objekt erkannt (also SONST), so soll das Förderband einfach permanent mit 50mm/s fahren. Wichtig ist hierbei allerdings, dass hier ebenfalls eine Verzögerungszeit von zwei Sekunden angegeben wird, da das Förderband sonst „stottern" würde, weil der Würfel über die gesamte Breite immer wieder neu als Objekt erkannt wird.

Eine weitere Möglichkeit das „Stottern" zu umgehen ist, dass eine zusätzliche Schleife eingebunden wird, in der die Aussage getroffen wird, dass „nichts" gemacht werden soll, solange ein Objekt erkannt wird. Auf diese Weise fährt das Förderband nicht permanent weiter, während der Würfel von der Lichtschranke erkannt wird (siehe Abbildung 19).

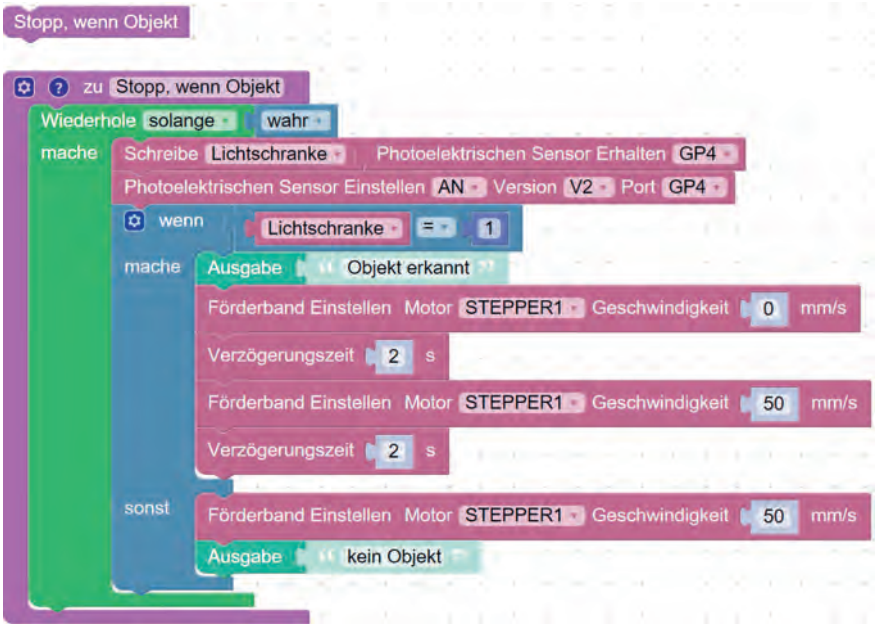


Abbildung 46: Sketch zu Übung 2c, Lösungsweg 1

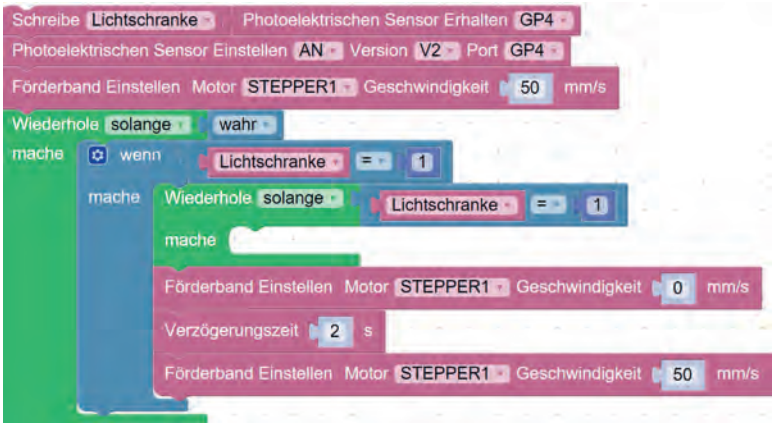


Abbildung 47: Sketch zu Übung 2c, Lösungsweg 2

Aufbauend auf der Übung soll nun der DOBOT eingebettet werden. Dieser soll nämlich nach der Objekterkennung (Würfel) eingreifen und das erkannte Objekt in einer Box lagern. Hierzu werden wieder sowohl Aufnahmeposition als auch Ablageposition als Variablen definiert. WENN die Lichtschranke nun ein Objekt erkennt, dann soll der DOBOT zur Aufnahmeposition fahren und den Sauggreifer einschalten. Nachdem der Würfel aufgenommen wurde, fährt der DOBOT zur Ablageposition und schaltet den Sauggreifer darüber ab. Wird kein Würfel erkannt (also SONST), so kann das Förderband weiter fördern. Es ergibt der in Abbildung 48 abgebildete Sketch.

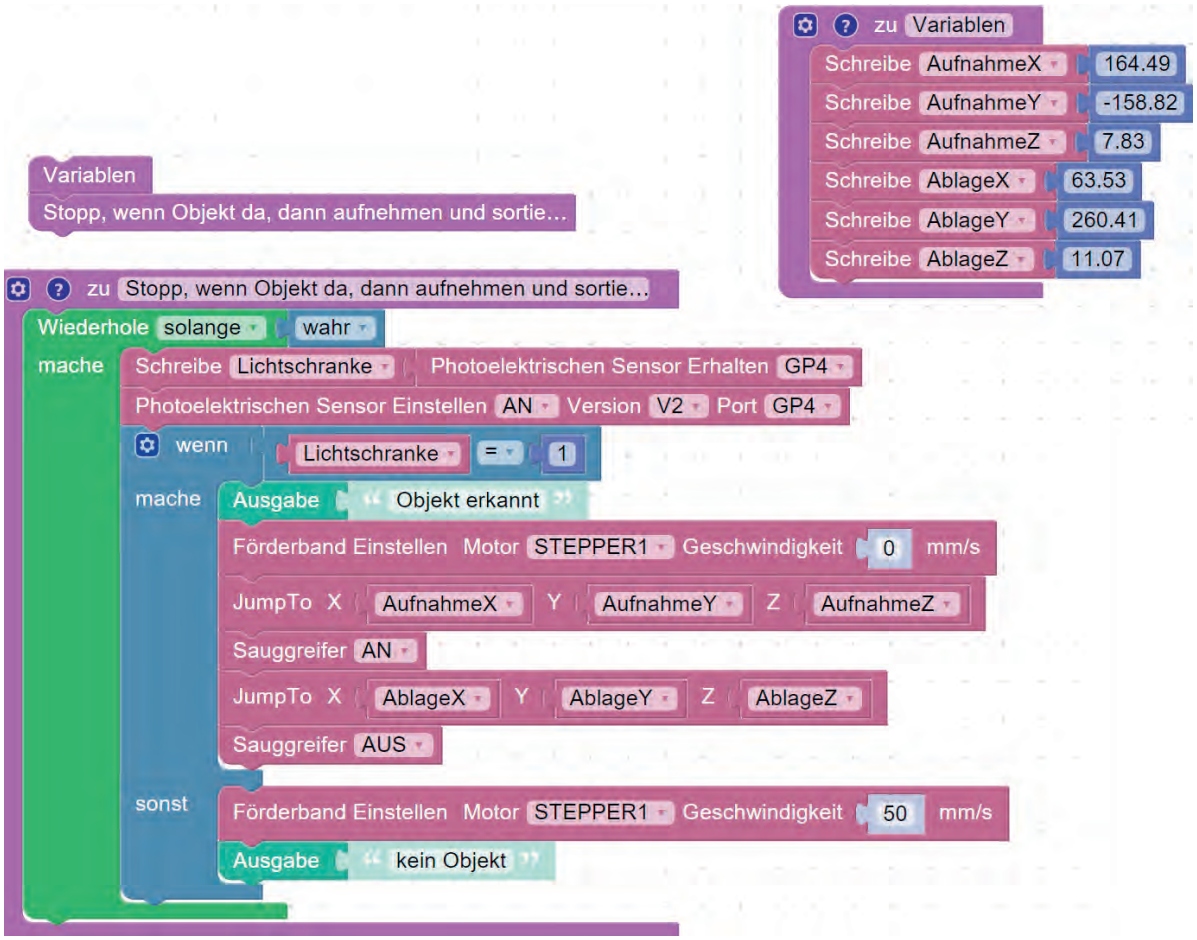


Abbildung 48: Fertiger Sketch zu Übung 3

3.5.3 Farbsensor

Nach erfolgreichem Aufbau des Farbsensors (Port GP2), soll in dieser Übung ein Sketch entwickelt werden, bei dem die vom Farbsensor erkannten Farben in einer Printausgabe angezeigt werden. Wie eine Printausgabe erstellt werden kann, ist aus den vorherigen Übungen zu entnehmen. Was in dieser Übung neu hinzukommt, sind die Befehle zur Ansteuerung des Farbsensors. Diese sind in der DOBOT-API unter „Zusätzlich“ zu finden (siehe Abbildung 49).

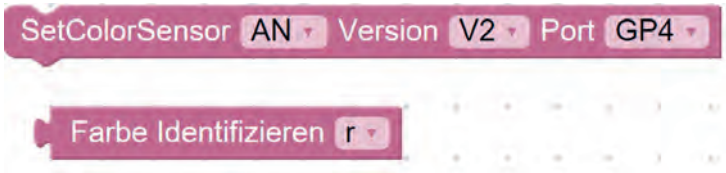


Abbildung 49: Befehle zur Steuerung des Farbsensors

Man beachte, dass der Farbsensor zunächst aktiviert, die richtige Version ausgewählt und der Port definiert werden muss (analog zur Lichtschranke). Der Befehl „Farbe identifizieren“ ist der für das Programm ausschlaggebende, da hier der Abruf zur Farberkennung gestartet wird. Die verschiedenen Farben (RGB) können über das Auswahlmenü variiert werden (siehe Abbildung 49). Damit die Farberkennung nun gestartet werden kann, ist wieder eine Endlosschleife zu implementieren, sodass die Farberkennung dauerhaft durchläuft. Anschließend müssen die Farbwerte abgefragt werden (Integer) und eine Printausgabe gesetzt werden, WENN eine Farbe erkannt wurde (siehe Abbildung 50).

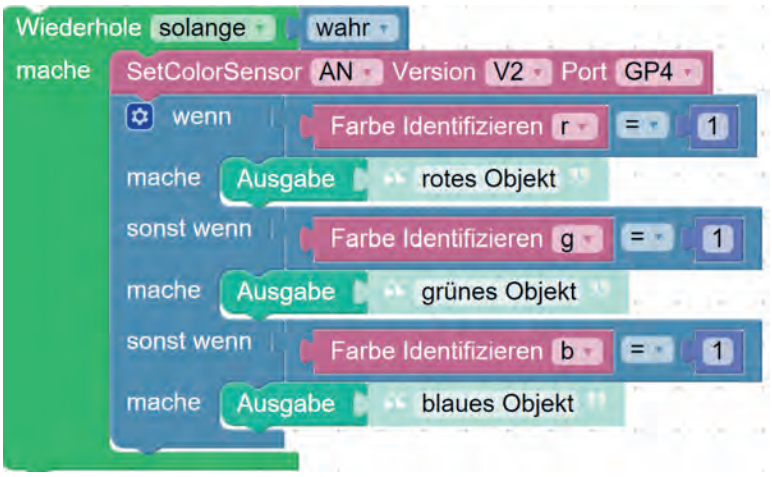


Abbildung 50: Printausgabe Farberkennung

Hat die Printausgabe funktioniert, kann nun eine Sortierung stattfinden. In dem Programm sollen Förderband, Lichtschranke, DOBOT und Farbsensor zusammenwirken. Zunächst sollte das Förderband stoppen, wenn ein Objekt erkannt wurde. Dieses Objekt sollte vom DOBOT aufgenommen und über den Farbsensor gelegt werden. Anschließend soll im Programm eine Farberkennung stattfinden, sodass der DOBOT die Würfel entsprechend der Farbe sortieren kann. Hierbei sollten alle roten Würfel in einer gesonderten Box und alle andersfarbigen in einer gemeinsamen Box gelagert werden.

In dem hier verfolgten Ansatz werden zunächst unter der Funktion „Variablen“ wieder alle Positionen definiert (siehe Abbildung 51). Zusätzlich dazu wird dem „Farbe identifizieren“ Befehl eine jeweilige Farbe zugeschrieben. In der Funktion zur Farbsortierung werden dann innerhalb einer Endlosschleife die Befehle so zusammengeführt, dass eine Farberkennung erfolgen kann.
Zunächst muss aber das Förderband gestoppt werden, sobald ein Würfel erkannt wurde. Anschließend wird der Würfel aufgenommen und über den Farbsensor gefahren. Je nach Farbe wird dann entsprechend sortiert.
Möchte man nun, dass jede Farbe eine Ablagebox erhält, muss lediglich eine neue Position zu den Variablen hinzugefügt werden. Es bietet sich an, die Position BOXGB (also Box grün blau) umzubenennen in BOXG (also Box für grüne Würfel). Anschließend kann eine neue Position für die blauen Würfel, also BOXB hinzugefügt werden (eine Position beinhaltet immer drei Variablen für die Koordinaten X;Y;Z). Zusätzlich zu den Positionsdefinitionen müssen die Positionen innerhalb der Funktion zur Farbsortierung angepasst werden (siehe Abbildung 52).

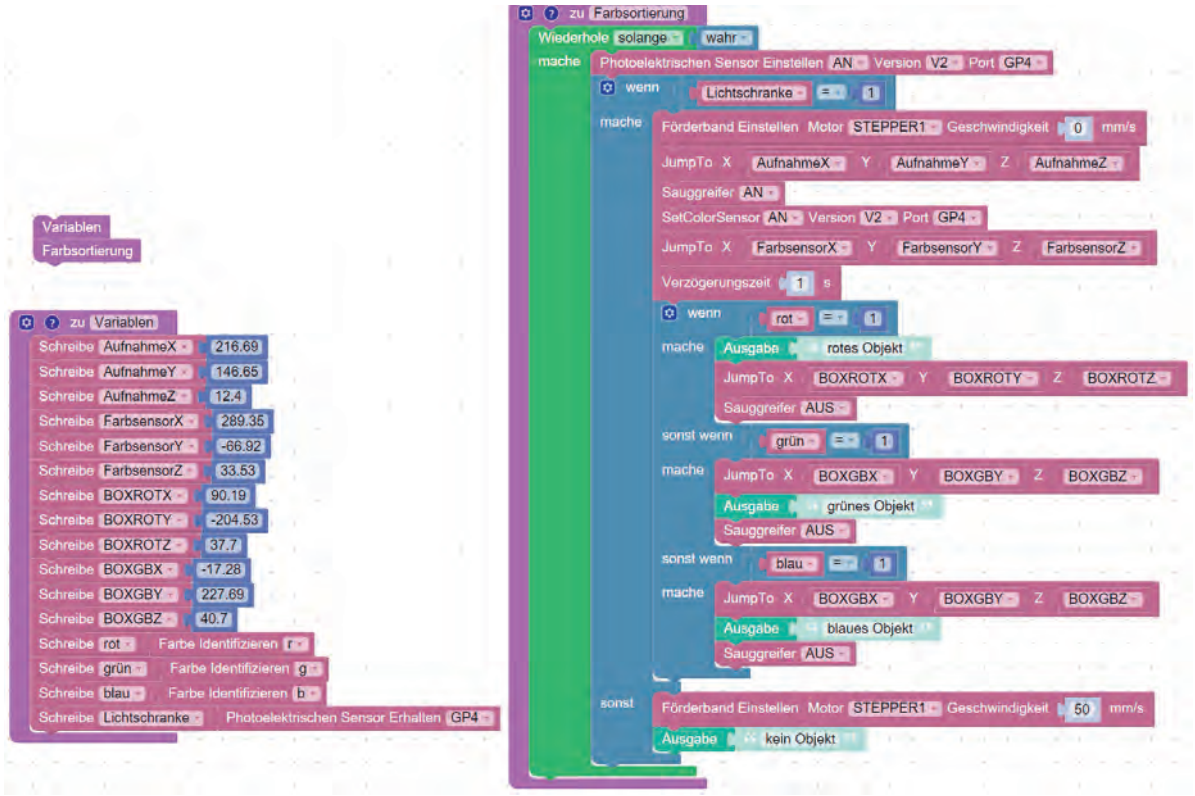


Abbildung 51: Lösung zu Übung 5

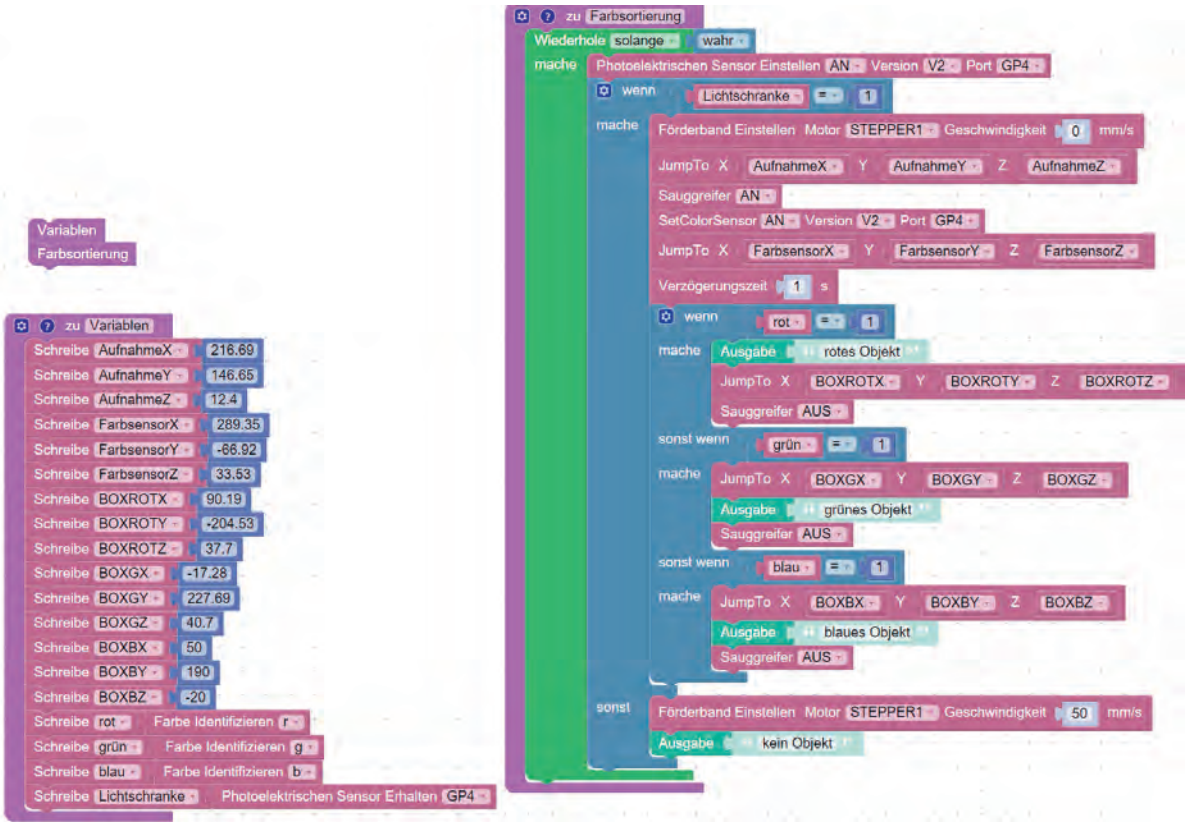


Abbildung 52: Lösung zu Übung 6

3.6 Linearachse

Dani Hamade, Carl von Ossietzky Universität Oldenburg

Die Linearachse ist eine Komponente, welche häufig in der Industrie, z. B. zur Bestückung von Maschinen mit Bauteilen, in Lagerwirtschaftssystemen, Produktionslinien etc., eingesetzt wird. Der Grund dafür ist, dass die Linearachse den Arbeitsbereich erheblich vergrößern kann und das Verfahren großer Differenzen dadurch mit hoher Geschwindigkeit automatisiert werden können, welches wiederum in flexibleren Produktionslösungen resultiert. Zu den Nachteilen zählt das Gewicht, da es zumeist zu schweren Konstruktionen führt und zusätzlich erhöht jede ergänzte Komponente die Fehleranfälligkeit, sodass auch die Linearachse für Störungen verantwortlich sein kann.

3.6.1 Verbindung mit der Linearachse

Bei Verwendung der Linearachse sind jeweils eigene Anschlüsse zu verwenden, welche sich unterhalb der Platte befinden, auf welcher der Dobot montiert werden muss. So muss der gelbe Stecker in die „Stepper2"-Buchse und der grüne in „GP2", welche zusammen mit der eigenen Stromversorgung ebenfalls aus der Schleppkette jeweils am Sockel des Dobots angeschlossen werden müssen. Die Linearachse selbst benötigt eine Stromversorgung mit einem Netzteil am äußeren Ende der Linearachse, von welchem ebenfalls ein USB-Kabel für den Rechner verläuft. Die Auswahl der Linearachse erfolgt oben im Fenster von DobotStudio, wie auch schon beim Sauggreifer, wo es nach Auswahl blau angezeigt wird.

3.6.2 Steuerung der Linearachse

Die Linearachse wird mit dem Block *SetLinearRail*, welcher in Abbildung 53 dargestellt ist, initialisiert.

Abbildung 53: Initialisierung der Linearachse

Die Definition von Geschwindigkeit und Beschleunigung findet mithilfe des Blockes *SetLinearRailSpeed* (siehe Abbildung 54) statt. Die höchstmögliche Geschwindigkeit der Linearachse liegt bei 150 mm/s und die Beschleunigung bei 150 mm/s^2.

Abbildung 54: Einstellen der Geschwindigkeit und Beschleunigung

Um zu bestimmen, zu welcher Position der Dobot entlang der Linearachse fahren soll, wird der Block *MoveLinearRailTo* verwendet (siehe Abbildung 55). Diese Koordinate beschreibt die absolute Position des Dobots auf der Linearachse in Millimeter, sodass der Dobot sich nicht bewegen wird, falls er bereits an der definierten Stelle steht. Der Bewegungsbereich (in mm) erstreckt sich von 0 bis 1000. Zusätzlich kann die Steuerung des Dobots auf der Linearachse manuell im Operation-Panel (via L+ und L-) erfolgen.

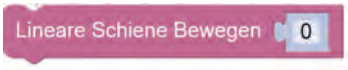


Abbildung 55: Befehl zur Bewegung der Linearachse

Im Fall, dass diese Buttons im Operation-Panel nicht dargestellt werden, muss unter dem Liste-Symbol rechts in der Ecke des Operation Panels ein Haken unter „Linear Rail control" gesetzt werden. Wie schon zuvor, ist ein Homing des Dobots sofort nach Verbindung mit der Linearachse durchzuführen, bei welchem der Dobot sowohl Start- als auch Endposition anfährt, damit keine Koordinationsschwierigkeiten auftreten. Ein Programm zum Abfahren des Start- und Endpunktes könnte z. B. so aussehen, wie in Abbildung 56 dargestellt.

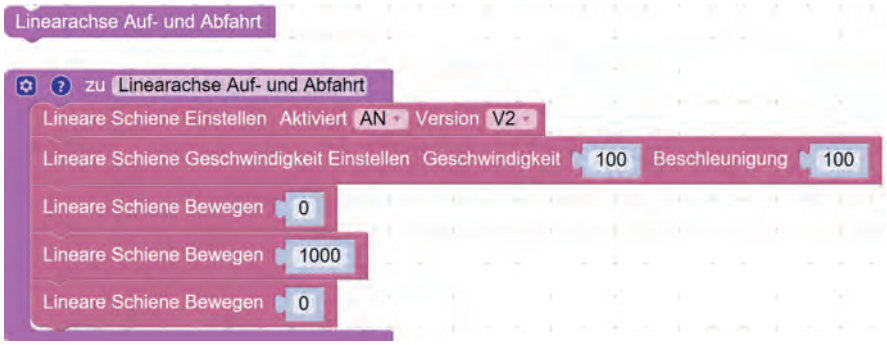


Abbildung 56: Beispiel zum Abfahren von Start- und Endposition

3.6.3 Interaktion mit dem Dobot und der Linearachse

Die Linearachse lässt sich nun mit dem Dobot problemlos ansteuern, mit dem Vorteil, dass sich die Bewegung um einen Freiheitsgrad und damit um eine Variable erhöht hat. Falls nun zum Beispiel neun Würfel (3 x 3) um einen Meter versetzt werden sollen (siehe Abbildung 57), lässt sich dies auf simple Weise realisieren.

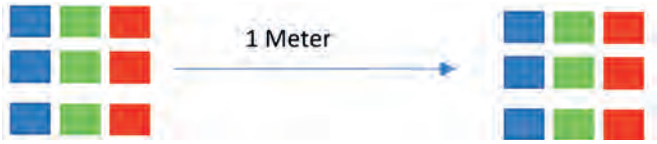


Abbildung 57: Übung Würfelmatrix mit Linearachse

Durch die Bewegung der Linearachse ist die Entfernung von 1 m bereits gewährleistet, wenn die *MoveLinearRail*-Blöcke auf 0 und 1000 jeweils gesetzt werden. Eine Änderung der Dobot-Positionskoordinaten ist damit nicht zwangsläufig notwendig. Eine Stellung, in die der Dobot zurückkehrt, wenn er sowohl aufgenommen als auch abgelegt hat, ist sinnvoll, um bei der Fahrt über die Linearachse eine Kollision mit Hindernissen, wie z. B. anderen Würfeln, zu verhindern. Außerdem findet die Initialisierung der Würfelbreite statt, deren Bedeutung im Folgenden näher erklärt wird. Die Variablen-Funktion kann folgendermaßen aussehen (siehe Abbildung 58).

Abbildung 58: Funktion der Variablen

Nun können die zwei Aufgaben des Dobots (Aufnahme und Ablage) definiert werden. Die X-, Y- und Z-Koordinaten sind bei beiden gleich, während sich die Linearachsen-Koordinate und die Sauggreifer-Funktion (on/off) ändern. Die Abbildungen 59 und 60 stellen die beiden Prozesse jeweils dar. Bei der Z-Position wird ein höherer Wert angegeben als eigentlich zutreffend und dann via *MoveDistance* angenähert, um den Würfel von oben abzusetzen und zu verhindern, dass bereits platzierte Würfel verschoben werden.

Ein direkter *MoveTo*-Befehl hätte dagegen eine Bewegung in gerader Linie zum Bestimmungsort veranlasst. In dem eigentlichen Skript können diese Komponenten zusammengefasst werden. Ein direkter *MoveTo*-Befehl hätte dagegen eine Bewegung in gerader Linie zum Bestimmungsort veranlasst. In dem eigentlichen Skript können diese Komponenten zusammengefasst werden.

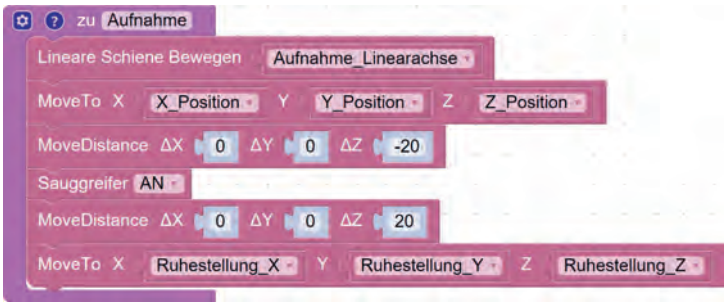


Abbildung 59: Funktion der Aufnahmeposition und der Bewegung

Mithilfe des Variablen-Aufrufs wird, wie zuvor, sichergestellt, dass die Variablen zugreifbar sind, woraufhin die Linearachse, zusätzlich zur Geschwindigkeit und Beschleunigung, initialisiert wird. Mit dem Aufbau, wie in Abbildung 5 zu sehen, und der hier getätigten Programmstruktur müssen die Variablen jeweils neu angepasst werden. In der Skizze stehen jeweils drei Würfel nebeneinander, d. h. man kann eine Schleife erstellen, die dreimal eine Handlung ausführt und nach jedem Schleifendurchlauf die X-Positionskoordinate um den Würfelbreite-Wert ändert. Daraufhin sollte in diesem Fall erneut eine Koordinate geändert werden, die Y-Koordinate.

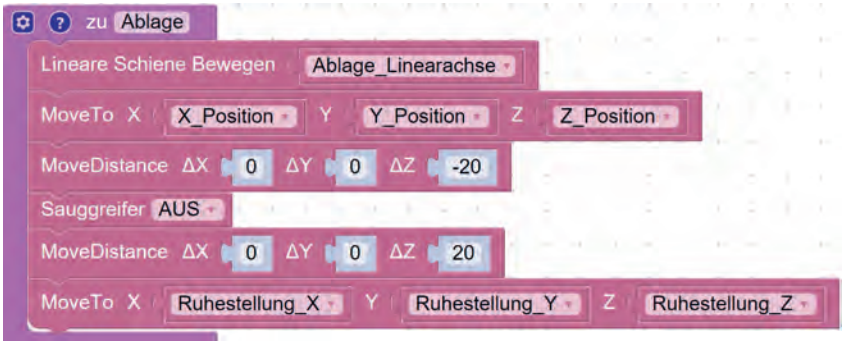


Abbildung 60: Funktion der Ablageposition und Bewegung

Bei direkt aneinander liegenden Würfeln wäre der nächste Würfel um eine Würfelbreite in Y-Richtung entfernt. Allerdings würde die X-Positionskoordinate lediglich für einen weiteren Schleifendurchlauf stimmen, da sie weiterhin um die Würfelbreite erhöht wird. Zur Verhinderung dessen wird die X-Koordinate also wieder auf den Initialwert gesetzt.

Der gesamte Prozess soll dreimal durchgeführt werden, weshalb sich wieder eine repeat-Schleife anbietet. Das Resultat des beschriebenen Vorgangs zum (Haupt-)Skript ist in Abbildung 61 dargestellt.

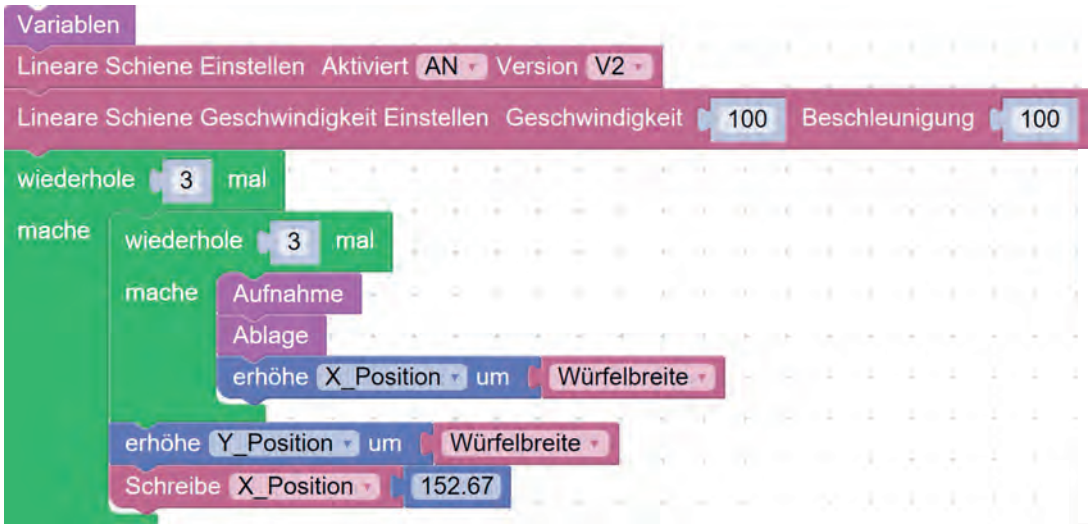


Abbildung 61: Möglicher Lösungsweg im Hauptskript

Die Abbildung 62 zeigt dagegen das vollständige Programm. Es handelt sich dabei lediglich, wie zuvor, um ein Lösungsbeispiel.

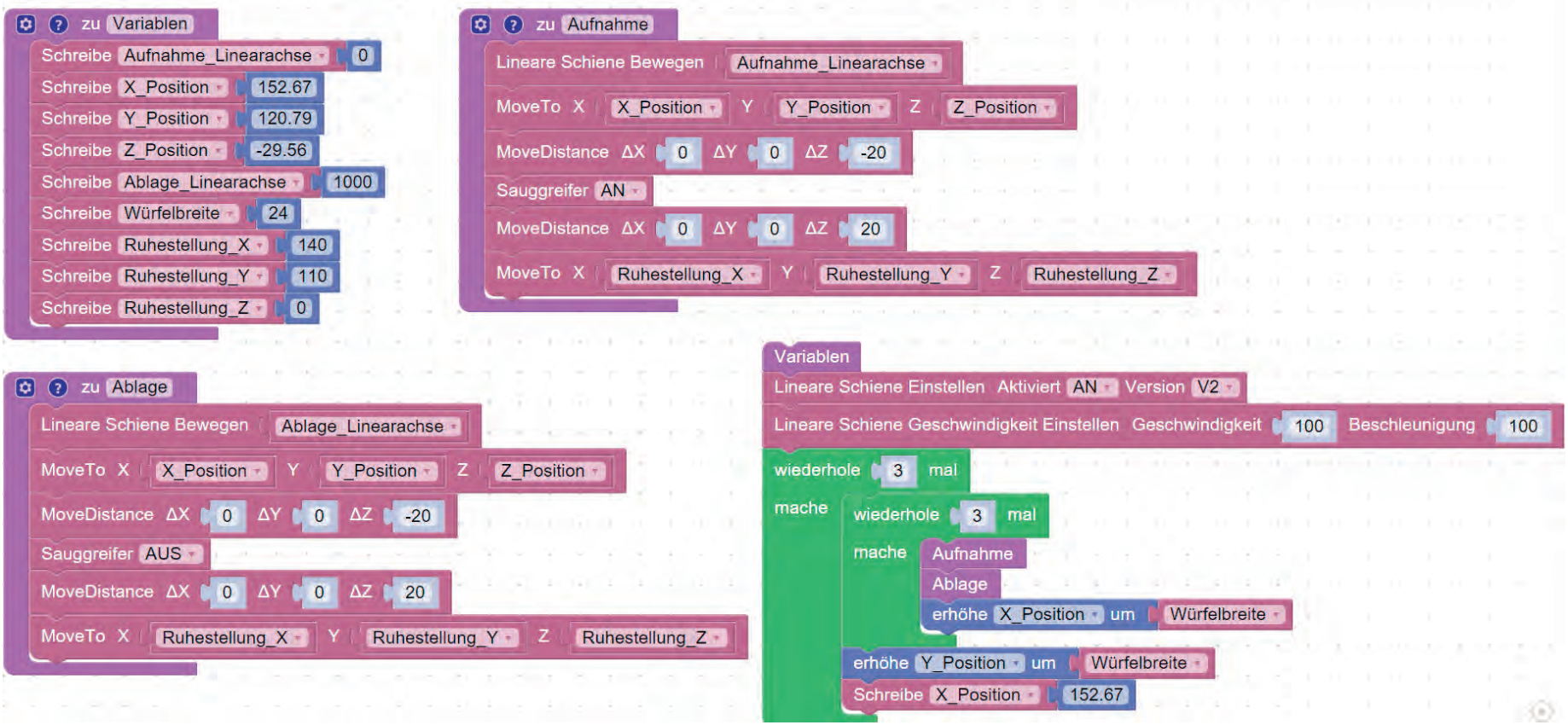


Abbildung 62: Musterlösung zum Übungsbeispiel

3.7 Nutzung der IO-Ports zur Vernetzung

3.7.1 Vernetzung mittels Taster

Jan Landherr, Dani Hamade, Carl von Ossietzky Universität Oldenburg

In der Industrie werden Roboter aus unterschiedlichen Gründen von Hand gesteuert. Auf der einen Seite kann zum Beispiel die Sicherheit von dem Personal besser gewährleistet werden, wenn der Roboter sich erst nach einem Tastendruck durch einen Anwender in Bewegung setzen kann. Auf der anderen Seite können so kompliziertere Prozesse besser zeitlich abgepasst werden, d. h. nachdem ein Roboter seine Tätigkeit beendet hat und nun für einen weiteren Arbeitsschritt einem anderen Roboter Raum geben muss, kann ein Taster an einem anderen Ort platziert werden, damit der vorige Roboter dem nächsten seinen Startzeitpunkt signalisiert. Auf diese Weise können Schäden durch Kollisionen vermieden werden. Kommunikation zwischen Robotern wird in den kommenden Abschnitten im Detail erklärt. Dieses Unterkapitel dient lediglich einem kurzen Ausblick auf den Anschluss eines Tasters an den Dobot Magician. In der NBC ist eine Bauanleitung für eine Tasterlösung enthalten, mit der zwei Taster in Verbindung mit zwei Dobots genutzt werden können.

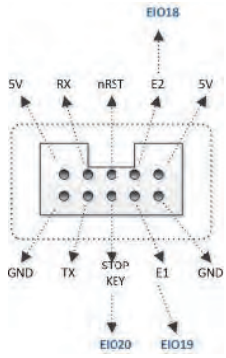
3.7.2 Vernetzung mittels EIO-Schnittstelle

Steffen Dreier, Evangelisches Gymnasium Nordhorn

Signale werden in der Digitaltechnik mit 0 ("LOW") oder 1 ("HIGH") kodiert. Diese Kodierung können wir z. B. mit einem Spannungslevel von 0 V oder 3,3 V realisieren. Dazu müssen die Schnittstellen entsprechend die Level-Input (Spannung wird gelesen) bzw. Level-Output (Spannung wird gesendet) Funktionalität beherrschen.

Achtung: Viele Mikrokontroller sind sehr empfindlich hinsichtlich der erlaubten Spannung. Eine zu hohe Spannung zerstört den Mikrokontroller irreversibel.

Für den den DOBOT Magician werden in der Dokumentation (Dobot Magician V2 User Guide V2.0.pdf) im Kapitel 4.3 Multiplexed I/O Interface Description die Eingabe/Ausgabe-Schnittstellen beschrieben. Die meisten Pins haben mehrfache Funktionsbelegungen. Im folgend vorgestellten Beispiel benutzen wir die Pins EIO18 und EIO19 sowie einen GND-Pin als Rückleiter aus dem "Communication Interface" auf der Rückseite des Dobot Magician. In der "I/O addressing"-Tabelle kann man erkennen, dass der PIN EIO18 den Level Output Modus jedoch nicht Level Input beherrscht. Der PIN EIO19 hingegen beherrscht nur den Level Input und nicht den Level Output. Ganz wichtig ist, dass beide PINs die gleiche "Voltage" haben! In diesem Fall also 3,3 V.



I/O Port	Spannung	Ausgang	PWM	Eingang	ADC
18	3,3 V	X	-	-	-
19	3,3 V	-	-	X	-
20	3,3 V	-	-	X	-

Ansicht und Beschreibung zum "Communication Interface" auf der Rückseite des Dobot Magician

3.7.2.1 Test mit einem DOBOT Magician

Mit Hilfe von einfachen female-DUPONT-Kabel werden die PINs EIO18 und EIO19 verbunden. Ein Kabel zum Verbinden der GND-PINs ist bei nur einem einzigen DOBOT Magician nicht notwendig, da alle GND PINs intern verbunden sind.

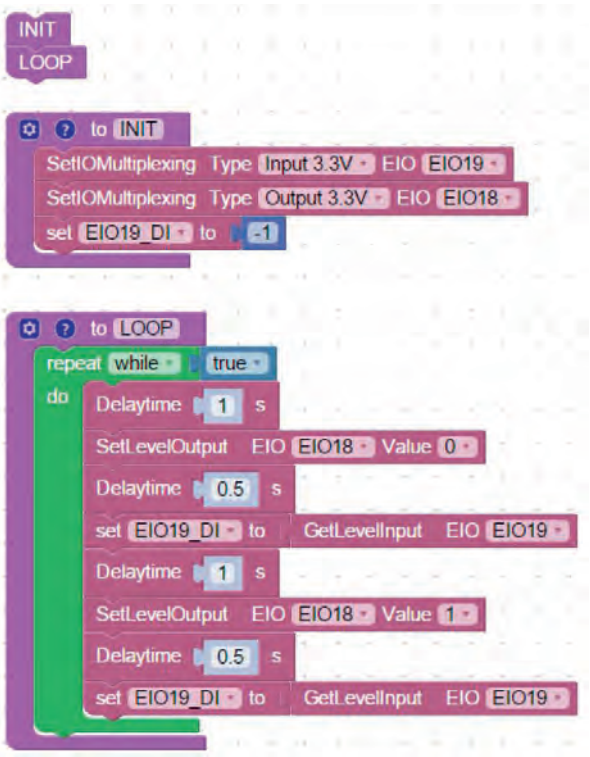
Achtung: Um Beschädigungen zu vermeiden, empfiehlt es sich, den DOBOT Magician auszuschalten und erst dann die PINs mit Kabeln zu verbinden.

Zum Testen der Output-Input-Verbindung ist in Abbildung 3 ein DobotBlock Programm abgebildet. Dabei wird zunächst in der Funktion "INIT" eine Variable (hier: EIO19_DI) mit dem Wert "-1" initialisiert  . Später sollen die Werte 0 und 1 zur Kontrolle auf der Scratch-Bühne der Dobot-Block Programmieroberfläche visualisiert werden. Weiter werden mit dem Block  der Port EIO19 für den Modus IOFunctionD**I** und der Port EIO18 für den Modus IOFunctionD**O** initialisiert.

Achtung: Wird der Modus nicht entsprechend der Hardwareleistungsfähigkeit der "I/O addressing Tabelle" initiiert, können Schäden am Roboter entstehen!

Im LOOP wird der Output-Port "EIO18" auf "LOW"  und dann "HIGH"  gesetzt und jedes Mal zur Kontrolle entsprechend der Input-Port "EIO19" in die Variable "EIO19_ID" ausgelesen  . Die "wait" Blöcke  sind entweder dafür da den Level-Wechsel nach dem "Set digital Output Port..."  gesichert abzuwarten oder für den Anwender sichtbar zu machen (nach der Zuweisung an die Variable "EIO19_DI"). Als Ergebnis sollte die Variable "EIO19_ID" im Abstand von 1,5 sec abwechselnd den Inhalt "0" und "1" anzeigen.

Umsetzung in DobotBlock für einen DOBOT Magician



Umsetzung in DobotStudio Blockly

3.7.2.2 Test mit zwei DOBOT Magician

Die beiden DOBOT Magician Roboter "DobotSender" und "DobotEmpfänger" werden so verkabelt, dass der PIN "EIO18" des "DobotSender" mit dem PIN "EIO19" des DobotEmpfänger" verbunden wird. Zusätzlich müssen ein GND PIN des "DobotSender" mit einem GND PIN des "DobotEmpfänger" mit einem Kabel verbunden werden. Im "Communication Interface" stehen dafür gleich zwei PINs zur Auswahl.

Achtung: Um Beschädigungen zu vermeiden, empfiehlt es sich, den DOBOT Magician auszuschalten und erst dann die PINs mit Kabeln zu verbinden.

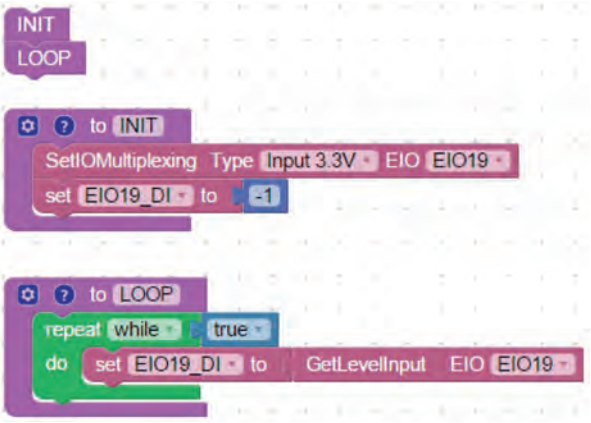
Verbindungskabel zwischen den beiden DOBOT Magician an die PINs des "Communication Interface"

Hinweis: Natürlich ist es möglich mit einem zusätzlichen Kabel den "DobotEmpfänger" Port EIO18 mit dem "DobotSender" Port EIO19 zu verbinden, um auch vom "DobotEmpfänger" Signale in der Gegenrichtung an den "DobotSender" zu schicken.

Zum Testen der Output-Input-Verbindung zwischen den beiden DOBOTs wird das DobotBlock Programm aus dem Beispiel mit nur einem DOBOT Roboter auf zwei Programme aufgeteilt.

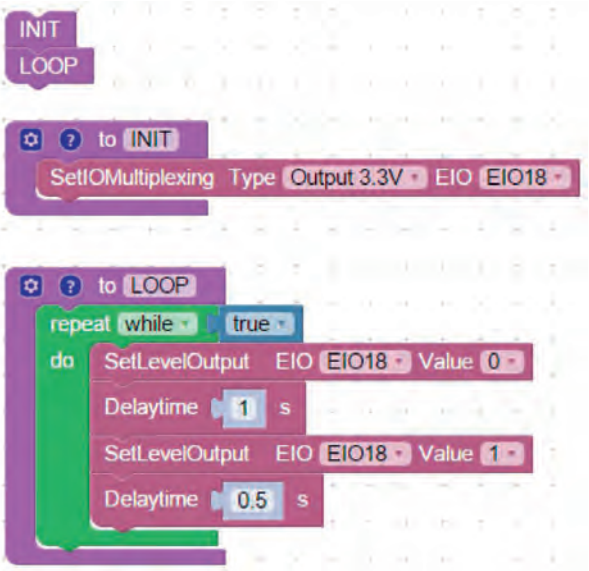
Achtung! Wird der Modus nicht entsprechend der Hardwareleistungsfähigkeit der "I/O addressing Tabelle" initiiert, können Schäden am Roboter entstehen!

Umsetzung in DobotBlock für den "DobotEmpfänger"



Umsetzung in DobotStudio Blockly für den "DobotEmpfänger"

Umsetzung in DobotBlock für den "DobotSender"



Umsetzung in DobotStudio Blockly für den "DobotSender"

Nachdem zuerst das Programm des Empfängers und anschließend das Programm des Senders gestartet wurde, wird im Abstand von einer Sekunde abwechselnd der Inhalt "0" und "1" in der Variable "EIO19_DI" auf der Scratch Bühne angezeigt.

3.7.3 Einführung in Python

Dani Hamade, Carl von Ossietzky Universität Oldenburg

Python ist eine höhere Programmiersprache, die sich in den vergangenen Jahren einer wachsenden Beliebtheit erfreut. Das hat zur Folge, dass ebenfalls die Anzahl an Anwendungsgebieten stetig zunimmt, sodass sie sich für beinahe jeden Zweck einsetzen lässt, sei es Webdesign, Datenbanken, Maschinelles Lernen uvm. Zur persönlichen Anwendung vieler Applikationen sind pythoninterne Bibliotheken verfügbar, die für den Gebrauch lediglich in die Umgebung importiert werden müssen, von wo sie aus frei einsetzbar sind. So ist der Dobot ebenfalls via Python steuerbar. Grundsätzlich gibt es eine firmeneigene Dobot-Python-API (*Application Programming Interface*), die sich allerdings nicht als sonderlich einsteigerfreundlich erweist, sodass eine eigene API entwickelt wurde. Diese basiert allein auf Python und muss zur Verwendung lediglich heruntergeladen und in der individuellen Python-Umgebung installiert werden (siehe Installationsinstruktionen). Die hier vorgestellten Befehle sind mithilfe dieser API ausführbar und werden im näheren erläutert. Im Gegensatz zu anderen Programmiersprachen arbeitet Python statt mit geschweiften Klammern mit Einrückungen, welches unter anderem die Lesbarkeit zur Folge hat, für die Python so geschätzt wird. Jede Missachtung von Einrückungen wird von Python mit konsequenter Fehlerausgabe bestraft, sodass bei der Programmierung besonders darauf geachtet werden muss.

3.7.3.1 Import der Python-API und erste Anwendung

Prinzipiell müssen benötigte Bibliotheken importiert werden, welches der Übersichtlichkeit wegen zumeist am Anfang eines Python-Programms geschieht. Für den ersten Gebrauch des Dobots sind nur die Bibliotheken „sys“ und die API „pyDobot“ von Belang:

```
import sys
import pyDobot
```

Die sys-Bibliothek dient dazu, den aktuellen Pfad in der Dateistruktur um einen weiteren Ordner „src“ zu erweitern. Der Grund dafür ist, dass die API dort gelagert ist, auf welche natürlich der Zugriff benötigt wird. Mit folgendem Befehl wird dies realisiert:

```
sys.path.insert(0, "src")
```

Nun kann die eigentliche Verbindung von Python zum Dobot aufgebaut werden:

```
dBot = Dobot("COM3")
```

Mittels dieses Befehles wird -für den Python-Kenner- ein Objekt namens „Dobot“ erzeugt. Jeder Anwender, dem dieser Ausdruck nichts sagt, muss sich damit nicht weiter beschäftigen, solange der Eingabeparameter hinter dem „Dobot“-Aufruf mit der aktuellen Verbindung übereinstimmt. Bei Windows-Benutzern wird dies unter dem „Geräte-Manager“ > „Anschlüsse (COM & LPT)“ > „Silicon Labs CP210x [...]“ angezeigt. Der wichtige Teil davon, für den in die Anführungszeichen gesetzten Eingabeparameter, ist in der Klammer „COM[...]“ mit der entsprechenden Zahl zu finden. Bei UNIX-basierten Systemen, wie Mac und Linux, sind die Verbindungen unter „/dev/[...]“ gelistet, wobei die USB-Verbindung an sich als Name auftritt, beispielsweise:

```
dBot = Dobot("/dev/ttyUSB0")
```

Um die Verbindung zu lösen, wird (Betriebssystem-unabhängig) folgender Befehl verwendet:

```
dBot.close()
```

3.7.3.2 Pick-and-Place in Python

Als erstes Beispiel könnte ein Programm erstellt werden, um den Dobot ein Objekt aufheben und an einem anderen Ort wieder ablegen zu lassen. Um das angeschlossene Werkzeug benutzen zu können, muss es im Programm zunächst initialisiert werden:

```
dBot.setEndEffector(Dobot.ENDEFFECTOR_SUCTIONCUP)
```

Mit diesem Aufruf wird der Sauggreifer als angeschlossenes Werkzeug erklärt, wodurch der Dobot die Koordinaten weiß, wann mit dem Ende des Sauggreifer die Untersatz-Oberfläche erreicht ist.
Der Sauggreifer selbst lässt sich mit dem folgenden Aufruf steuern:

```
dBot.suctionCupOn(True)
```

Der Eingabeparameter True ist eine boolean-Variable und sorgt für die Einschaltung des Sauggreifers, während False den Sauggreifer ausschaltet.
Mithilfe von einigen Variablendeklarationen und dem jump-Befehl lässt sich der Dobot in einer JUMP-Bewegung zum gewünschten Ort transportieren:

```
dBot.jumpTo(X_Aufnahme, Y_Aufnahme, Z_Aufnahme, R_Aufnahme)
```

Das vollständige Programm könnte dann beispielsweise so aussehen:

```
import sys
import pyDobot

sys.path.insert(0, "src")

def Pick_Place():
    dBot = Dobot("COM3")
    dBot.setEndEffector(Dobot.ENDEFFECTOR_SUCTIONCUP)

    X_Aufnahme = 211.57
    Y_Aufnahme = -157.48
    Z_Aufnahme = -43.92
    R_Aufnahme = -36.66

    X_Ablage = 230.12
    Y_Ablage = 62.61
    Z_Ablage = -44.53
    R_Ablage = 15.22

    dBot.jumpTo(X_Aufnahme, Y_Aufnahme, Z_Aufnahme, R_Aufnahme)
    dBot.suctionCupOn(True)
    dBot.jumpTo(X_Ablage, Y_Ablage, Z_Ablage, R_Ablage)
    dBot.suctionCupOn(False)
    dBot.moveRel(0.0,0.0,10.00)
    dBot.close()
```

In diesem Beispiel wurde das Programm in eine Funktion eingebettet, kenntlich gemacht durch das def. Ähnlich wie bei Blockly werden die Funktionen nicht durch ihre bloße Existenz aufgerufen. Der Quellcode-Absatz zeigt lediglich nur eine Funktionsdeklaration. Der Aufruf von der Funktion könnte durch den uneingerückten Funktionsnamen samt Parameter stattfinden:

```
Pick_Place()
```

Wobei die elegantere Methode darin besteht eine if-Kondition aufzustellen, welche fragt, ob das Programm selbst aufgerufen wurde oder durch ein anderes Programm. Bei manchen Programmen möchte man möglicherweise nur einige Funktionen eines Skripts übernehmen. Bei einem Import würde das Programm aber auch jede Zeile, die nicht Teil einer Funktion ist, mit ausführen.

Um dies zu verhindern, wird folgendes hinzugefügt:

```
if __name__== '__main__':
  Pick_Place()
```

Es gibt bei den Bewegungsbefehlen neben der oben gezeigten noch die Möglichkeit das über eine API-interne Variable zu realisieren, nämlich über *DobotPose([x, y, z, r])*. Dadurch kann das Programm ein wenig kürzer und damit ebenfalls übersichtlicher gestaltet werden. Um die Parameter allein und nicht das ganze Objekt zu übergeben, muss bei dem jump-Aufruf ein Asterisk (*) vorgesetzt werden:

```
pose = DobotPose([230.12, 62.61, -44.53, 15.22])
dBot.jumpTo(*pose)
```

Der Einfachheit halber wurde im Folgenden mit den Standardvariablen statt mit dem *DobotPose([])*-Objekt gearbeitet.

3.7.3.3 Schleifen in Python

Jeden Arbeitsschritt in einem Programm einzeln zu schreiben, ist nicht nur müßig, sondern verfehlt auch den Sinn des Programmierens vollständig, da es die Arbeit erleichtern soll. Wenn also regelmäßige Anpassungen vorgenommen werden müssen, lässt es sich mittels einer Schleife zusammenfassen, solange die Bedingungen klar formulierbar sind. Es gibt generell (in jeder Programmiersprache) *while*- und *for*-Schleifen. While bezeichnet den Schleifenprozess, welcher solange durchgeführt ist, bis ein Schlusskriterium (dem while-Aufruf anhängend) erreicht wurde:

```
zaehler = 0

while kriterium < zaehler:
  […]
  zaehler = zaehler + 1
```

Die letzte Zeile lässt sich sogar verkürzen:

```
zaehler += 1
```

Auf diese Weise wird der Zähler automatisch um einen Wert erhöht. Dies lässt sich ebenso auf Multiplikation, Subtraktion und Division anwenden.
Die andere Schleife ist die *for*-Schleife, in welchem die Zählervariable automatisch angepasst wird. In Python wird das Start- und Abbruchkriterium bei einer simplen Zählung mit *range([Start,]Ende)* realisiert:

```
for zaehler in range(0,3):
  […]
```

In einem Beispiel, in welchem der Dobot einen Turm aus drei nebeneinander liegenden Schaumstoff-Würfeln bauen soll, könnte das zum Beispiel so aussehen:

```
import time

def Turmbau():
    dBot = Dobot("COM3")
    dBot.setEndEffector(Dobot.ENDEFFECTOR_SUCTIONCUP)

    X_Aufnahme = 211.57
    Y_Aufnahme = -157.48
    Z_Aufnahme = -43.92
    R_Aufnahme = -36.66
    Wuerfelbreite = 24.5

    X_Ablage = 230.12
```

```
    Y_Ablage = 62.61
    Z_Ablage = -44.53
    R_Ablage = 15.22

    for i in range(3):
        dBot.jumpTo(X_Aufnahme, Y_Aufnahme,  Z_Aufnahme, R_Aufnahme)
        dBot.suctionCupOn(True)
        dBot.jumpTo(X_Ablage, Y_Ablage, Z_Ablage, R_Ablage)
        dBot.suctionCupOn(False)
        dBot.moveRel(0.0,0.0,10.0)
        time.sleep(0.5)
        Y_Aufnahme += Wuerfelbreite
        Z_Ablage += Wuerfelbreite
    dBot.close()

if __name__ == '__main__':
  Turmbau()
```

Dieses Programm wurde wieder in Form einer Funktion zur besseren Übersichtlichkeit definiert, welches in der sogenannten „main"-Funktion aufgerufen wird. Hier wurde nun für eine Verzögerung, ähnlich zu dem „Delaytime" in Blockly eingebaut. Da Python keine eigene Funktion dafür besitzt, wird die Bibliothek *time* importiert und aufgerufen (*time.sleep([...])*).
Um das Dobot-Objekt nicht wiederholt aufrufen zu müssen, kann das Objekt bei Funktionsaufruf als Parameter übergeben werden. Dafür muss es in der Funktion als Eingabeparameter in die Klammer nach dem Funktionsnamen definiert und bei Aufruf (durch die main) eingesetzt werden:

```
def Turmbau(Dobot: dobot):
    dobot.setEndEffector(Dobot.ENDEFFECTOR_SUCTIONCUP)

    X_Aufnahme = 211.57
    Y_Aufnahme = -157.48
    Z_Aufnahme = -43.92
    R_Aufnahme = -36.66
    Wuerfelbreite = 24.5

    X_Ablage = 230.12
    Y_Ablage = 62.61
    Z_Ablage = -44.53
    R_Ablage = 15.22

    for i in range (3):
        dobot.jumpTo(X_Aufnahme, Y_Aufnahme,  Z_Aufnahme, R_Aufnahme)
        dobot.suctionCupOn(True)
        dobot.jumpTo(X_Ablage, Y_Ablage, Z_Ablage, R_Ablage)
        dobot.suctionCupOn(False)
        dobot.moveRel(0.0,0.0,10.0)
        time.sleep(0.5)
        Y_Aufnahme += Wuerfelbreite
        Z_Ablage += Wuerfelbreite

if __name__ == '__main__':
      dBot = Dobot("COM3")
      Turmbau(dBot)
      dBot.close()
```

Auf diese Weise kann das *Dobot*-Objekt für jede Funktion verwendet werden, ohne einzeln erzeugt werden zu müssen. Dass das Objekt in der *Turmbau()*-Funktion anders heißt, ist möglich, da es bei Funktionsdeklaration in der Klammer entsprechend definiert wurde. Es muss nicht zwangsläufig *dBot* heißen. Bei der Funktion fällt aber noch das *Dobot:* auf. Dies ist optional und dient dazu, in den IDEs die automatische Vervollständigung zu ermöglichen, welche andernfalls nicht funktionieren würde. Der Funktion wird damit vermittelt, dass es sich bei der genannten Variable um das *Dobot*-Objekt handelt, worauf sich wiederum die automatische Vervollständigung bezieht.

3.7.3.4 Nutzung des Förderbandes und des RGB-Farbsensors

Mithilfe der Python-Dobot-API lässt sich auch das angeschlossene Förderband steuern. Es muss nach Erzeugung des Dobot-Objektes initialisiert werden:

```
dBot.setConveyorPort(DobotPort.STEPPER1)
```

Die Einstellung der Geschwindigkeit (in mm/s) ist ähnlich wie in Blockly:

```
dBot.setConveyorSpeed(50)
```

Nun muss eine Verzögerung und das Setzen der Geschwindigkeit auf 0 folgen. Insgesamt könnte dies so aussehen:

```
dBot.setConveyorSpeed(50)
time.sleep(6.0)
dBot.setConveyorSpeed(0)
```

Einen Dobot nacheinander drei Würfel aufnehmen und auf das Förderband setzen zu lassen, welches sich daraufhin in Bewegung setzt, könnte, wie folgt, aussehen:

```
def Foerderband():
    dBot = Dobot("COM4")
    dBot.setEndEffector(Dobot.ENDEFFECTOR_SUCTIONCUP)
    dBot.setConveyorPort(DobotPort.STEPPER1)

    X_Aufnahme = 224.28
    Y_Aufnahme = -31.18
    Z_Aufnahme = -50.94
    X_Ablage = -81.39
    Y_Ablage = -213.91
    Z_Ablage = 8.59

    numLoops = 3
    Wuerfelbreite = 24.5

    for k in range(0, numLoops):
        dBot.jumpTo(X_Aufnahme, Y_Aufnahme, Z_Aufnahme)
        dBot.suctionCupOn(True)
        dBot.jumpTo(X_Ablage, Y_Ablage, Z_Ablage)
        dBot.suctionCupOn(False)

        dBot.setConveyorSpeed(50)
        time.sleep(6.0)
        dBot.setConveyorSpeed(0)
        Y_Aufnahme += Wuerfelbreite

    dBot.close()
```

Ähnlich ist es beim Farbsensor, welcher zunächst initialisiert werden muss:

```
dBot.setColorSensor(True, DobotPort.GP2)
```

Die Farbabfrage geschieht ähnlich wie bei Blocky:

```
dBot.getColor()
```

Die einzelnen Farben werden mit einem Punkt abgetrennt in einer if-Anweisung abgefragt:

```
if dBot.getColor().g:   ODER  if dBot.getColor().g == True:
```

Die if-Anweisung ist der in Blockly ähnlich. Auf das if folgt die zu prüfende Variable und die Prüfbedingung wird danach angehängt. Prüfbedingungen können „<“ (kleiner als), „>“ (größer als), „<=“ (kleiner oder gleich wie), „>=“ (größer oder gleich wie), „!=“ (ungleich) oder eben auch „==“ (gleich) sein. Um jeweils auf eine bestimmte Bedingung zu prüfen, kann auf ein „if“ ein „elif“ folgen: Wenn es nicht die „if“-Bedingung erfüllt, dann evt. die „elif“-Bedingung. Im Fall, dass keine der Bedingungen zutrifft, kann die Befehlskette nach „else“ die entsprechenden Aktionen einleiten.
Es gibt ebenfalls die Möglichkeit auf mehrere Bedingungen zu prüfen. Falls rot oder grün erkannt werden, führe etwas aus:

```
if dBot.getColor().g or dBot.getColor().r:
```

Dies wird entsprechend mit einem or zwischen zwei Konditionen realisiert. Falls zwei Bedingungen zutreffen müssen, müsste das or durch ein and ersetzt werden. Bei Farbsortierung sind mehrere Bedingungen vielleicht nicht so sinnvoll, allerdings ist es vorteilhaft diese Methode kennengelernt zu haben. Wenn nun ein Dobot, wie bei einer Produktionskette, durch das Förderband transportierte Würfel, aufnehmen, analysieren und den Farben entsprechend sortieren soll, könnte das folgendermaßen aussehen.

```
def Aufgabe4_2_3():
    dBot = Dobot("COM4")
    logging.basicConfig(level=logging.INFO)
    dBot.setEndEffector(Dobot.ENDEFFECTOR_SUCTIONCUP)
    dBot.clearAlarms()

    X_Aufnahme = -29.50
    Y_Aufnahme = -195.02
    Z_Aufnahme = 9.72
    R_Aufnahme = -98.60

    X_RGB = 44.82
    Y_RGB = -283.02
    Z_RGB = 35.92
    R_RGB = 15.22

    X_Rot = 182.12
    Y_Rot = -169.60
    Z_Rot = -41.43

    X_Blau = 225.59
    Y_Blau = 37.97
    Z_Blau = -50.35

    X_Gruen = 209.97
    Y_Gruen = -64.49
    Z_Gruen = -50.77

    X_Farblos = -29.50
    Y_Farblos = -195.02
    Z_Farblos = 9.72

    for i in range(3):
        time.sleep(5.0)
        dBot.jumpTo(X_Aufnahme, Y_Aufnahme,  Z_Aufnahme, R_Aufnahme)
        dBot.suctionCupOn(True)
        dBot.jumpTo(X_RGB, Y_RGB, Z_RGB, R_RGB)

        time.sleep(2.0)

        dBot.setColorSensor(True, DobotPort.GP2)
        if dBot.getColor().g:
            print("Gruen")
            dBot.jumpTo(X_Gruen, Y_Gruen,  Z_Gruen)
```

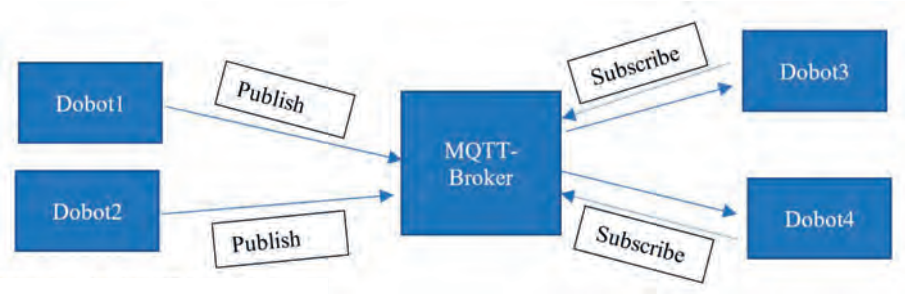
```
        elif dBot.getColor().r:
            print("Rot")
            dBot.jumpTo(X_Rot, Y_Rot,  Z_Rot)
        elif dBot.getColor().b:
            print("Blau")
            dBot.jumpTo(X_Blau, Y_Blau,  Z_Blau)
        else:
            print("Weder rot noch blau noch gruen")
            dBot.jumpTo(X_Farblos, Y_Farblos, Z_Farblos)

        dBot.suctionCupOn(False)
        dBot.moveRel(0.0,0.0,10.0)

dBot.close()
```

3.7.3.5 Vernetzte Produktion mit Python (MQTT)

Damit man nun mehrere Dobots miteinander vernetzten kann, ohne sie direkt miteinander verdrahten zu müssen wie bei dem Optokoppler oder dem Taster, kann man standardisierte Nachrichtenprotokolle wie MQTT zur Machine to Machine Kommunikation (M2M) nutzen. Diese Funktion wurde in der Python API für den Dobot implementiert. MQTT ist hierbei ein Standard-Messaging-Protokoll für das Internet of Things (IoT).

> *„Das Messaging-Protokoll MQTT (Message Queueing Telemetry Transport) kommt vor allem im Bereich des Internet Of Things (IOT) zum Einsatz. Ziel bei der Entwicklung dieses Protokolls war das Funktionieren in unzuverlässigen Netzwerken mit geringer Bandbreite mit Geräten, die nur über eingeschränkte Ressourcen (wie geringem Speicher) verfügen und verhältnismäßig kurze Nachrichten übertragen." (Abts, 2022, S. 187).*

Grundlegend beinhaltet die MQTT-Struktur sogenannte Publisher, einen Broker und Subscirber (siehe Abbildung 68). Die Publisher sind in unserem Anwendungsfall Dobots, die Nachrichten versenden. Einer oder mehrere andere Dobots können diese Nachrichten als Subscirber, welche durch den MQTT Broker von den Publishern entkoppelt sind, empfangen. Die Nachrichten, die vom Publisher versendet werden gehen an ein bestimmtes Topic (z.B. Farberkennung). Je nach dem, für welches Topic die Subscriber angemeldet sind, werden sie über den Broker an diese weitergeleitet.



Als Broker verwenden wir in diesem Fall den Open Source MQTT-Broker „Eclipse Mosquitto". Eine detaillierte Installationsanleitung erhält man hier: https://mosquitto.org/download.
Jeder Dobot der nun eine Verbindung zum Broker aufbaut, ist ein Client mit einer bestimmten ID. Diese ClientID sollte sorgfältig gewählt werden, sodass die Dobots hinterher gut voneinander zu unterscheiden sind. Im Python Code wird dies wie folgt festgelegt (hier „myDobot"):

```
# intern
import sys
import time
# own
sys.path.insert(0, "src")        # Diese Zeile ist nur notwendig, wenn
pyDobot nicht installiert wurde.
from pyDobot import DobotMqtt, DobotPort, DobotIOFunction
```

```
if __name__ == "__main__":
    dBot = DobotMqtt(
        mqtt_host       = "localhost",
        mqtt_auth       = ("mqttuser", "mqttpasswd"),
        mqtt_client_id  = "myDobot",
        serial_port     = "COM3"
    )
```

Zunächst werden oben die notwendigen Bibliotheken importiert. Im Hauptprogramm beginnen dann direkt schon die Definitionen für die MQTT-Struktur. Wir arbeiten in diesem Beispiel zunächst über einen lokalen Host (also der Laptop als Broker). Zur Autorisierung kann man bei Mosquitto nun ein Passwort und einen Username festlegen (hier „mqttuser“ und „mqttpasswd“). Die Client ID ist diejenige, welche auch im Broker angezeigt wird (hier myDobot). Zum Schluss muss man nur noch den Port definieren (der große Vorteil ist nun, dass man beliebig viele Dobots (z. B. über einen USB HUB) an einen Laptop anschließen kann). Als Erweiterung kann man hier auch einen Raspberry PI als Broker schalten, sodass der Laptop nicht zwangsweise permanent bei den Dobots stehen muss. Man kann die Dobots dann über WLAN und den Raspberry steuern. Anhand eines Beispiels soll nun gezeigt werden, wie der Nachrichtenaustausch abläuft. Die Problemstellung ist, dass zwei Dobots, die in einer Fertigungslinie sehr nah aneinander stehen, ein Homing bei Schichtbeginn ausführen sollen. Da die Dobots allerdings so nah beieinanderstehen, müssen sie das Homing zur Kollisionsvermeidung nacheinander durchführen. Dobot1 soll in diesem Fall also eine Referenzfahrt durchführen und erst nach Abschluss eine Nachricht an den Broker übermitteln, dass er fertig ist, sodass der zweite Dobot (Subscriber) starten kann. Es ergibt sich folgender Code:

Für **Dobot1**:

```
import sys
import time
# own
sys.path.insert(0, "src")            # Diese Zeile ist nur notwendig,
wenn pyDobot nicht installiert wurde.
from pyDobot import DobotMqtt, DobotPort, DobotIOFunction

if __name__ == "__main__":
    dBot = DobotMqtt(
        mqtt_host       = "localhost",
        mqtt_auth       = ("mqttuser", "mqttpasswd"),
        mqtt_client_id  = "Dobot1",
        serial_port     = "COM3"
    )

    dBot.clearAlarms()
    dBot.enableRemoteCtrl(True)
    dBot.home()
    dBot.sendMqttMsg ("fertig")
    dBot.close()
    sys.exit()
```

Für **Dobot2**:

```
# intern
import sys
import time
# own
sys.path.insert(0, "src")            # Diese Zeile ist nur notwendig,
wenn pyDobot nicht installiert wurde.
from pyDobot import DobotMqtt, DobotPort, DobotIOFunction

if __name__ == "__main__":
    dBot = DobotMqtt(
        mqtt_host       = "localhost",
        mqtt_auth       = ("mqttuser", "mqttpasswd"),
```

```
    mqtt_client_id  = "Dobot2",
    serial_port     = "COM3"
)

dBot.clearAlarms()
dBot.enableRemoteCtrl(True)
msg=dBot.getMqttMsg()
if msg ['text'] == "oki":
    dBot.home()

dBot.close()
sys.exit()
```

Nun könnte man beliebig viele Nachrichten zwischen den Dobots austauschen (z. B. könnte Dobot2 nach dem Homing den Arbeitsprozess initiieren, indem er wieder eine Nachricht sendet, die Dobot1 empfängt). Außerdem ist es möglich, nur auf Nachrichten eines bestimmten Absenders zu reagieren (durch Ergänzung eines „from“ in der Bedingung). Will man, dass die Dobots zu einem bestimmten „Topic“ (also Oberthema) eine Handlung ausführen, so kann man „publishen“ und „subscriben“. Ein Beispieltopic wäre hier die Farberkennung. Der Publisher müsste folgende Daten veröffentlichen:

```
dBot.mqttPubMsg("Farbe",dBot.getColor())
```

Der andere Dobot muss diese Daten entsprechend subscirben:

```
dBot.mqttSubMsg("Farbe",dBot.getColor())
```

Literatur

Abts, D. (2022): *Masterkurs Client/Server-Programmierung mit Java. Anwendungen entwickeln mit Standard-Technologien. 6. Auflage*. Wiesbaden: Springer Fachmedien GmbH.

3.8 Vergleich der Programmieroptionen, Möglichkeiten in DobotLab

Marvin Bersiner, robospace gGmbH

In diesem Abschnitt wird die Software DobotLab kurz vorgestellt. Im Anschluss gibt es eine Auflistung der Fehler und die jeweiligen Work-Arounds der drei bestehenden Dobot Softwareanwendungen DobotStudio, DobotBlock und DobotLab. Abschließend werden noch die Vorteile bei der Nutzung von DobotLab als Hauptsoftwareanwendung aufgezeigt.

3.8.1 Allgemeine Vorstellung DobotLab

DobotLab ist eine integrierte Software Plattform, die webbasiert via https://dobotlab.dobot.cc erreicht oder desktopbasiert als Programm auf dem Laptop oder PC installiert werden kann. Das folgende Bild zeigt den Startbildschirm von DobotLab. Getestete Software, sowie weitere Schulungsunterlagen sind unter https://robospace.de/dobot-schulung erreichbar. Die aktuellste Software und ein User Guide sind über die offizielle Website von Dobot https://dobot.cc unter Support > Download Center verfügbar.

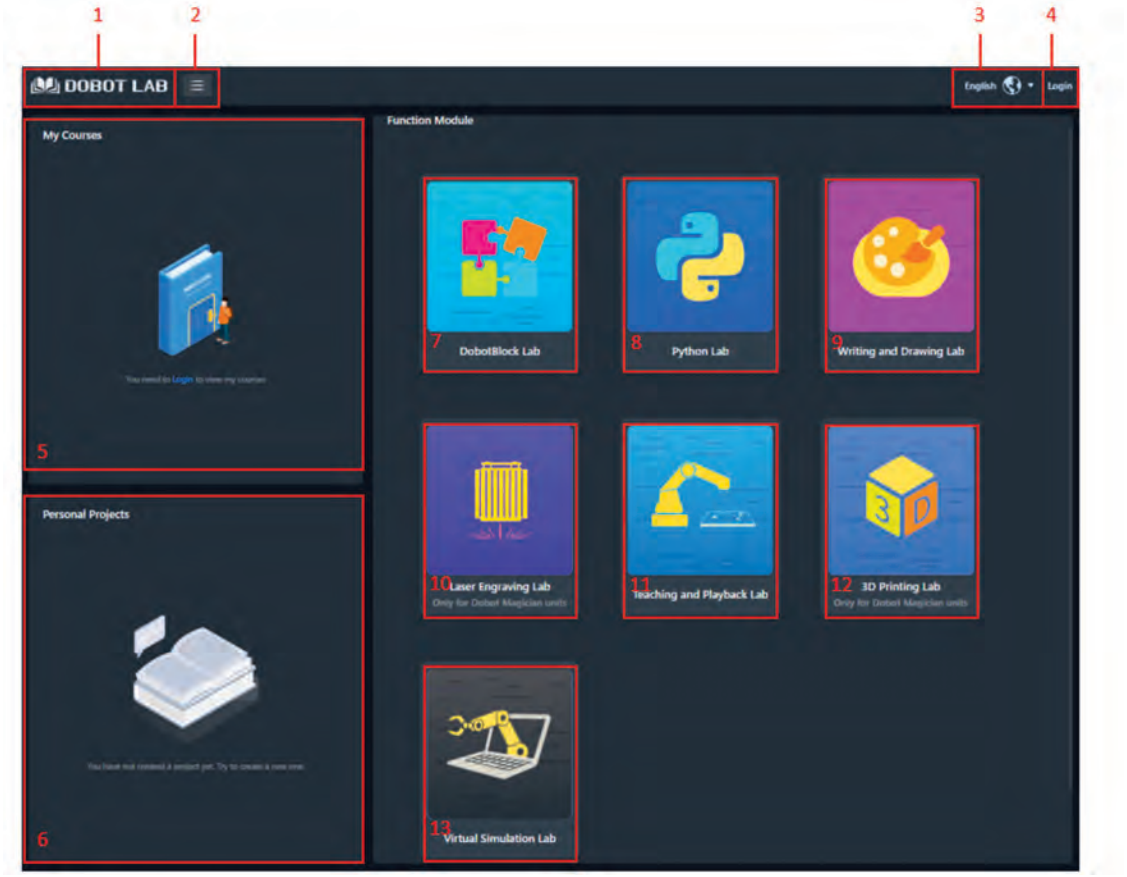


Abbildung 1: Übersicht DobotLab

Nr.	Modul	Beschreibung
1	Startseite	Klicken, um zur DobotLab Startseite zu gelangen
2	Menü	Currency: Anleitung zur Nutzung von DobotLab Help: Anzeigen und Download von Dokumenten für DobotLab Feedback: Feedback über DobotLab weitergeben About: Informationen über DobotLab
3	Sprache	Auswahl der Sprache
4	Login	Einloggen mit dem eigenen Account
5	Meine Kurse	Angebot von frei-zugänglichen Online-Kursen
6	Persönliche Arbeiten	Anzeigen und bearbeiten von gespeicherten Projekten
7	DobotBlock Lab	Bedienung des Dobot durch blockbasiertes Programmieren

Nr.	Modul	Beschreibung
8	Python Lab	Bedienung des Dobot durch skriptbasiertes Programmieren
9	Writing and Drawing Lab	Bewegung des Dobot um zu Schreiben oder zu Malen
10	Laser Engraving Lab	Steuerung des Dobot zum Gravieren eines Bitmap-Bildes mit einem Laser
11	Teaching and Playback Lab	Den Dobot anlernen, wie er sich bewegen soll. Aufzeichnung der Bewegung, Dobot führt die aufgezeichnete Bewegung aus.
12	3D Printing Lab	Bedienung des Dobot Magician um 3D zu drucken
13	Virtual Simulation Lab	Simulation von Roboterbewegungen durch Programmieren innerhalb einer virtuellen Umgebung und eines Roboter Modells

Im Folgenden wird nur auf Aspekte eingegangen, die sich zu den Softwareanwendungen DobotStudio oder DobotBlock unterscheiden.

3.8.2 Python Lab – Skriptbasierte Programmierumgebung

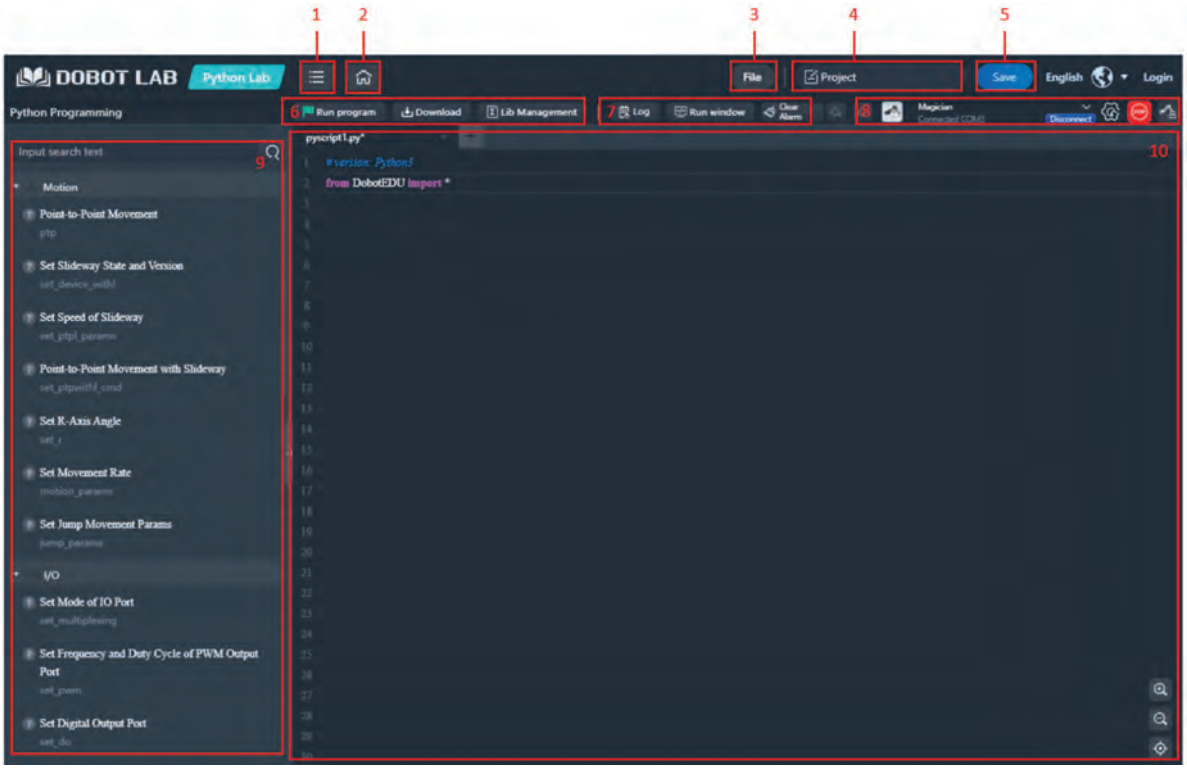


Abbildung 2: Startseite PythonLab

Nr.	Modul	Beschreibung
1	Menü	Currency: Anleitung zur Nutzung von DobotLab Help: Anzeigen und Download von Dokumenten für DobotLab Feedback: Feedback über DobotLab weitergeben About: Informationen über DobotLab
2	Home	Klicken, um zur DobotLab Startseite zu gelangen
3	Datei	Beinhaltet die Funktionen: Neue Datei, Öffnen, Speichern als, Hochladen von Lokal,..
4	Projekt Name	Anzeige des aktuellen Projektnamens
5	Speichern	Aktuelles Projekt speichern

Nr.	Modul	Beschreibung
6	Programm Kontrolle	„Run program“: Klicken um das aktuelle Programm im Code Bereich zu starten „Download“: Downloaden des aktuellen Programms auf den Speicher des Roboters „Lib Management“: Installieren von Python Erweiterungs-Bibliotheken. Danach können die Bibliotheks-Funktionen aufgerufen werden.
7	Lauf Informationen	Log: Alarm Infos werden angezeigt. Clear Alarm: Um die Fehlermeldungen zu löschen. Run window: Zeigt die aktuelle Laufzeit Aktivitäten
8	Geräte Kontrolle	Connect: Auswahl eines Gerätes und Aufbauen einer Verbindung mit dem Gerät Stop: Drücken, um im „Notfall“ das Programm zu stoppen Roboter-Symbol: Kontrolle/Bewegung des Dobot Magician
9	Kommando Liste	Anzeige der zur Verfügung stehenden Befehle. Doppelklick auf die Befehle, damit sie im Code Bereich erscheinen. Beim Klicken auf das Fragezeichen erscheint die Dokumentation zu den jeweiligen Befehlen.
10	Code Bereich	Bearbeitung von Programmen mithilfe von Python

Anmerkung: Bei der Verwendung von externen Geräten wie Sensoren oder dem Förderband sollte beim Verbinden mit dem Roboter „Magic Box + Magician“ ausgewählt werden. Erst dann werden in der linken Spalte alle notwendigen Befehle für die externen Geräte angezeigt.

3.8.3 Writing and Drawing

Die folgende Abbildung zeigt die Startseite des Writing and Drawing Programms. In dem rot markierten Feld kann der Text eingegeben werden, der von dem Dobot Magician gezeichnet werden soll. Der Text oder das Bild, welches gemalt werden soll, sollte sich in dem eingezeichneten Bereich befinden, ansonsten kann der Roboter die Zeichnung nicht abfahren, da er sich außerhalb seiner Achsgrenzen befindet.

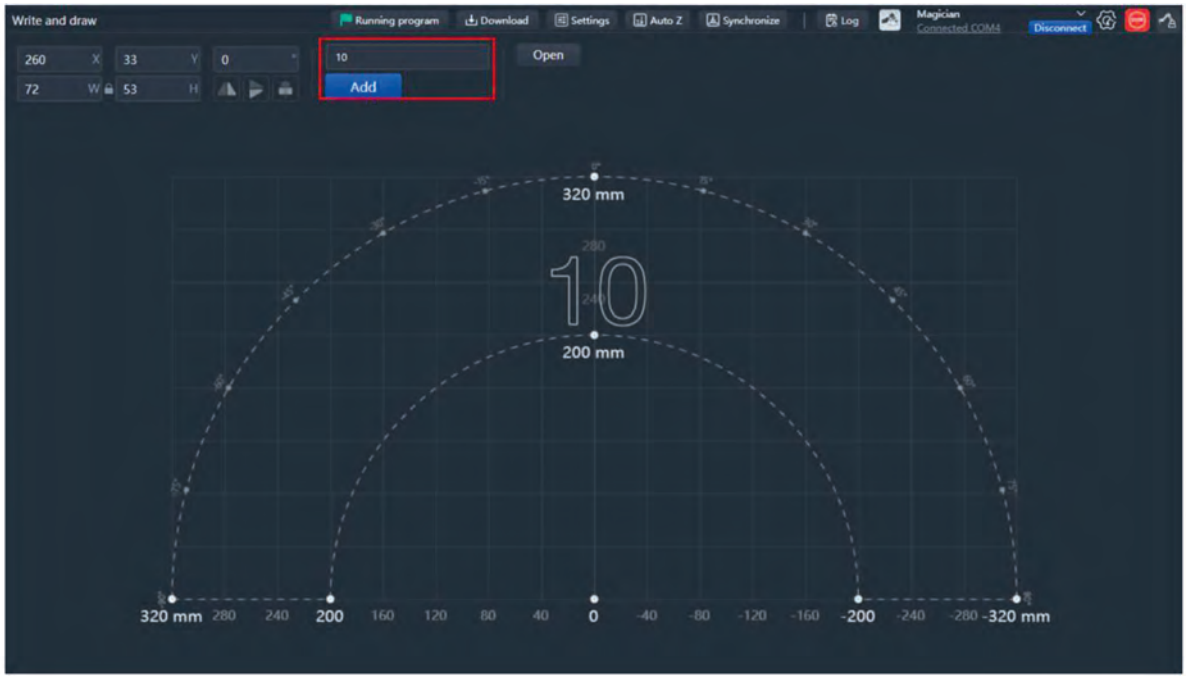


Abbildung 3: Startseite Write and Draw

Unter „Settings“ kann vor der Benutzung des Stiftes die Stiftanhebe-Höhe eingegeben werden. Diese liegt standardmäßig bei 20 mm. Bevor die Zeichnung abgefahren wird, muss der Stift einmal am Papier angesetzt werden. Wenn dies geschehen ist, kann der Button „Auto Z“ gedrückt werden, um die aktuelle Z-Position zu speichern.

Mithilfe der Linearachse können auch längere Texte oder Zeichnungen angefertigt werden. Hierfür muss zunächst der Roboter auf der Linearachse befestigt werden. Bei Klicken des Roboter-Symbol (rechts oben) öffnet sich das „Arm Control Panel“. In diesem muss für das Aktivieren der Linearachse bei „Rail“ das Häkchen gesetzt werden. Nun ist die Linearachse aktiv und der Arbeitsraum des Roboters hat sich verändert. Es können längere Grafiken dargestellt werden.

3.8.4 Fehlerliste der Hardware Dobot Magician und der Softwareanwendungen DobotStudio, DobotBlock und DobotLab

Fehler	**Tritt auf**	**Work-Around**
Hardware		
Ausschalter schaltet den Roboter nicht aus	Wenn der Roboter abstürzt	Stromkabel trennen
Stepper Motoren des Förderbands bewegen sich nicht korrekt	Wenn man zu hohe oder zu niedrigere Geschwindigkeiten einstellt	Mit einem Wert im Bereich von 50 mm/s starten und sich langsam an die Grenzen tasten
DobotStudio: Allgemein		
Fehlermeldung auf Chinesisch	Wenn der falsche Port beim Verbinden ausgewählt wird	Richtigen Port auswählen
Notaus funktioniert nicht	Wenn der Roboter abstürzt/aufhängt/nicht reagiert	Stromkabel trennen
DobotStudio: Teach and Playback		
Spalte für das Werkzeug / Linearachse wird nicht angezeigt	Wenn das Tool ausgewählt wird und vorher keins (oder der Stift) ausgewählt war	Neuen Punkt erstellen
Spalten/Positionen haben keinen Wert für die Werkzeugspalte	Wenn Positionen erstellt wurden und danach ein anderes Werkzeug ausgewählt wird	Doppelklick auf das entsprechende Feld und Zustand auswählen
DobotStudio: Blockly		
Alle Blöcke werden ausgeführt	Wenn man Blöcke vom Hauptstrang „abzieht“ und irgendwo in der Oberfläche positioniert	Blöcke einzeln deaktivieren
Man kann nicht mehrere Blöcke auf einmal deaktivieren		
Übersetzung von „Movements“ als „Antrag“		
Greifer dreht sich ungewollt	Wenn man den Springe zu/Jump to Block verwendet	
Es gibt keine Möglichkeit einer Joint Bewegung mit kartesischen Koordinaten	Wenn man Rechtsklick -> Add Point verwendet	
Förderband läuft nicht	Bei zu hohen Werten	Geschwindigkeitsrampe programmieren
„Photoelektrischer Sensor Erhalten“ gibt keinen booleschen Wert, sondern eine Zahl zurück		In einer Variable zwischenspeichern

Fehler	Tritt auf	Work-Around
Sensorausgangsblöcke sind nicht in jedem optisch passenden Block verwendbar		In einer Variable zwischenspeichern
Dataentypen können optisch nicht unterschieden werden		Ausprobieren, ob der Block an den Eingang andockt
Man kann die Linearachse nicht gleichzeitig mit dem Arm bewegen		
Man kann die Linearachse nicht mit einer vorgegebenen Geschwindigkeit steuern		
DobotBlock		
Linearachse kann nicht verwendet werden	Wenn der Geschwindigkeitsblock der Linearachse verwendet wird	Anderen Block verwenden oder DobotBlock v1.6.0-beta9 benutzen
Block zu Manipulation der Geschwindigkeit und Beschleunigung funktioniert nicht mehr	Bei Verwendung von DobotBlock v1.6.1-beta.4	DobotBlock v1.6.0-beta9 benutzen
Linearachse und Sensoren funktionieren nicht	Wenn mehrere Roboter in einem Programm verwendet werden	Alle Sensoren und Linearachsen an dem Bot anbringen, von welchem aus das Programm gestartet wird
DobotLab: Allgemein		
Kann Wifi-Modul verbinden, aber im Nachgang nicht mehr finden		Nutze DobotStudio, damit findest du den Roboter im Netz
In Python Lab: Kein Befehl *get_endeffektor*		
In Python Lab: Kein Befehl für die Arc-Bewegung		
DobotBlock Lab: Knopf zum Aufrufen des Python-Codes nicht immer verfügbar	Wenn man sich beim Öffnen von DobotLab mit dem Internet verbunden ist	DobotLab ohne Internet öffnen, dann erscheint der Python Button

3.8.5 Vorteile bei der Nutzung von DobotLab

Die folgende Auflistung zeigt eine Übersicht der Vorteile von DobotLab gegenüber der Nutzung von DobotBlock oder DobotStudio.

- Als Desktop und Web Anwendung nutzbar
- Teaching, Blockbasierte und Textbasierte Programmierung in einem Programm möglich
- Simulierte Umgebung vorhanden
- Viele Fehler aus den Programmen DobotStudio und DobotBlock wurden behoben
  - Bspw. können nun bei der Ansteuerung von zwei oder mehr Robotern die Sensoren und externen Komponenten an beliebigen Robotern angeschlossen werden.
- Bessere Dokumentation innerhalb des skriptbasierten Programmierumgebung

4 Umgang mit den Cobots

4.1 Universal Robots

Marvin Bersiner, Abdelaziz Tekaya, robospace gGmbH

Inhalt

4.1.1 Einführung

In diesem Kapitel werden wir uns mit den Grundlagen der Programmierung von Universal Robots der e-Serie befassen. Die e-Serie ist eine fortschrittliche Generation von kollaborativen Roboterarmen, die für ihre einfache Handhabung, Flexibilität und Sicherheit bekannt sind. Die e-Serie umfasst Modelle wie UR3e, UR5e und UR10e, die jeweils unterschiedliche Traglasten und Reichweiten bieten, um den Anforderungen verschiedener industrieller Anwendungen gerecht zu werden.
Als kleiner Hinweis vorweg soll hier an dieser Stelle die kostenlose Universal Robots Academy (https://academy.universal-robots.com/de/kostenloses-e-learning/e-learning-fur-die-e-series/) hervorgehoben werden. Gerade vor dem Erstkontakt mit dem Roboter oder auch zur Wiederholung einzelner Inhalte ist diese kostenlose interaktive Onlineschulung jedem ausnahmslos zu empfehlen. Aber nun zurück zum eigentlichen Inhalt des Kapitels.
Wir beginnen mit einer Einführung in die Universal Robots Software, die zur Programmierung und Steuerung von e-Series-Robotern verwendet wird. Die Software verfügt über eine benutzerfreundliche grafische Oberfläche, die es auch Personen ohne Programmierkenntnisse ermöglicht, komplexe Roboteranwendungen zu erstellen. Wir werden uns mit den grundlegenden Funktionen der Software vertraut machen und lernen, wie man den Roboterarm durch die verschiedenen Bewegungsmodi steuert.
Anschließend werden wir uns auf die Programmierung von e-Series-Robotern konzentrieren, indem wir die grundlegenden Programmierstrukturen und -befehle vorstellen, die in der Universal Robots Software verwendet werden. Wir werden auch den Umgang mit Variablen, Schleifen und bedingten Anweisungen erläutern, um komplexere Roboteraktionen und Entscheidungen zu ermöglichen.
Darüber hinaus werden wir uns mit dem Anschluss von externen Geräten, wie Sensoren und Greifern, an den Roboterarm beschäftigen und lernen, wie man sie in die Programmierung einbindet, um den Roboter noch vielseitiger und effizienter zu gestalten.
Schließlich werden wir uns mit den Sicherheitsaspekten der Programmierung von e-Series-Robotern auseinandersetzen. Wir werden die verschiedenen Sicherheitsfunktionen und -einstellungen, die in der Universal Robots Software verfügbar sind, erkunden und erläutern, wie man sie effektiv einsetzt, um eine sichere und produktive Zusammenarbeit zwischen Menschen und Robotern zu gewährleisten.

4.1.1.1 Der Roboterarm

Der UR besteht aus Roboterarm, Teach Pendant (TP) und Controller. Der e-Serie Roboterarm ist ein Knickarm-Roboter und verfügt über 6 Achsen und 3-Phasen AC Servomotoren mit einer Bewegungsfreiheit von +/- 360°. Die folgende Darstellung zeigt die modulare Gelenkkonfiguration der 6-achsigen UR-Cobots im Vergleich.

4.1.1.2 Teach Pendant (TP)

Das Teach Pendant ist ein tragbares Bedien- und Programmiergerät, das hauptsächlich in der Industrieautomation und Robotik eingesetzt wird. Es ermöglicht dem Bediener, die Bewegungen von Industrierobotern manuell zu steuern, zu programmieren und zu testen. Durch die Verwendung eines Teach Pendants kann der Bediener sicher und effizient mit dem Roboter interagieren, um dessen Bewegungen genau zu definieren und anschließend für wiederholbare, automatisierte Aufgaben zu speichern. In folgender Abbildung sind die unterschiedlichen Bestandteile des TP dargestellt.

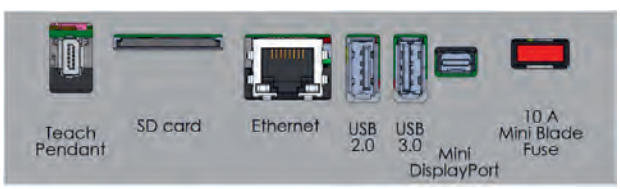
Nr.	Element
1	Startknopf
2	Not-Aus-Schalter
3	USB-Anschluss
4	3PE Knöpfe oder Freedrive Knopf

4.1.1.3 Controller

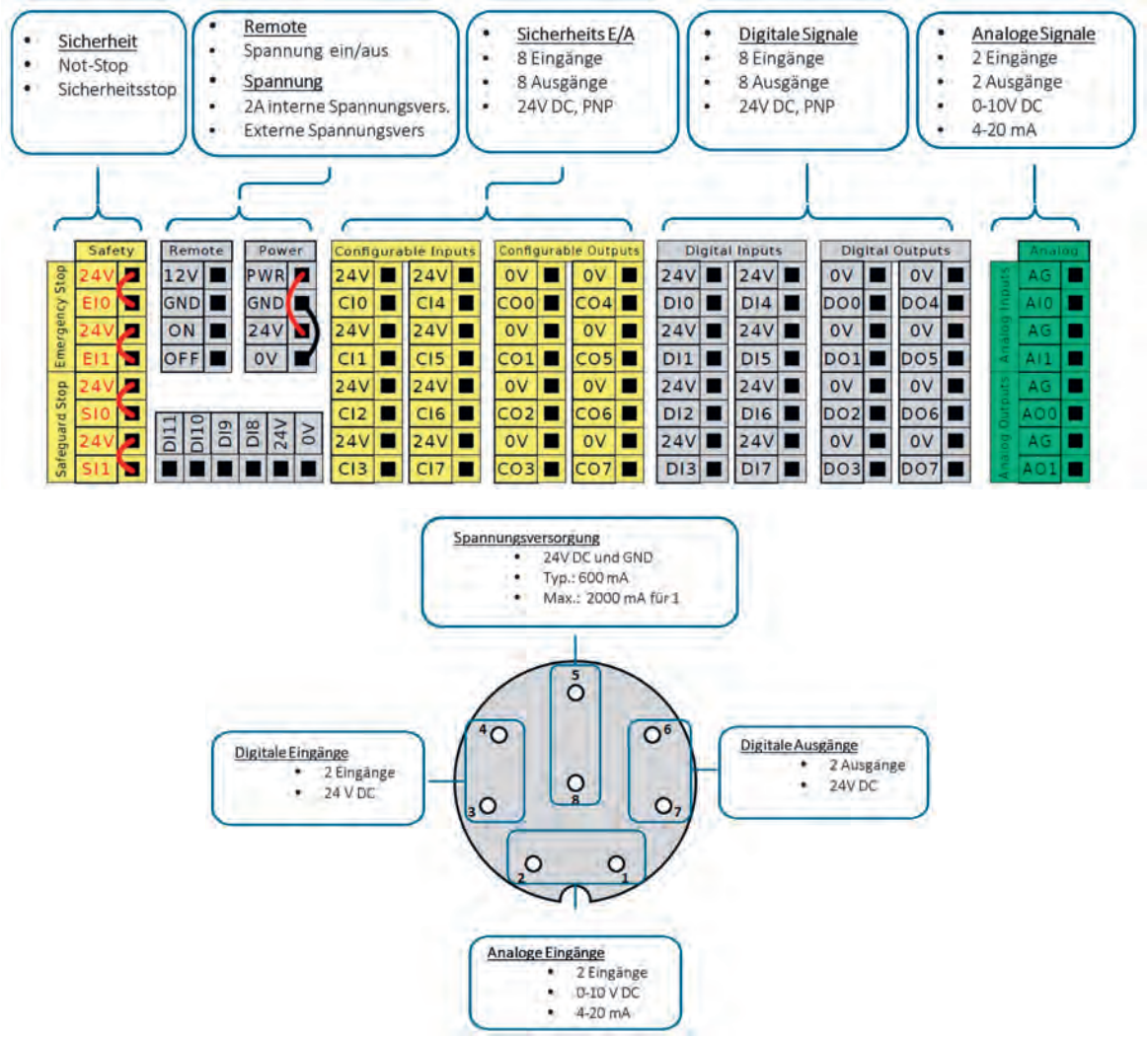
Der Controller des UR ist das zentrale Steuerungs- und Rechenelement des Robotersystems. Er koordiniert und leitet alle relevanten Informationen, wie Sensordaten und Bewegungsanweisungen, um die gewünschten Roboteraktionen auszuführen.
Die Anschlüsse eines UR-Controllers variieren je nach Modell, aber im Allgemeinen sind folgende Anschlüsse vorhanden:

Externe Anschlüsse:
- Stromanschluss 220/110 V AC
- Roboterarmanschluss

Interne Anschlüsse:
- Teach Pendant
- SD-Card
- Ethernet
- USB
- Mini DisplayPort
- Controller-E/A

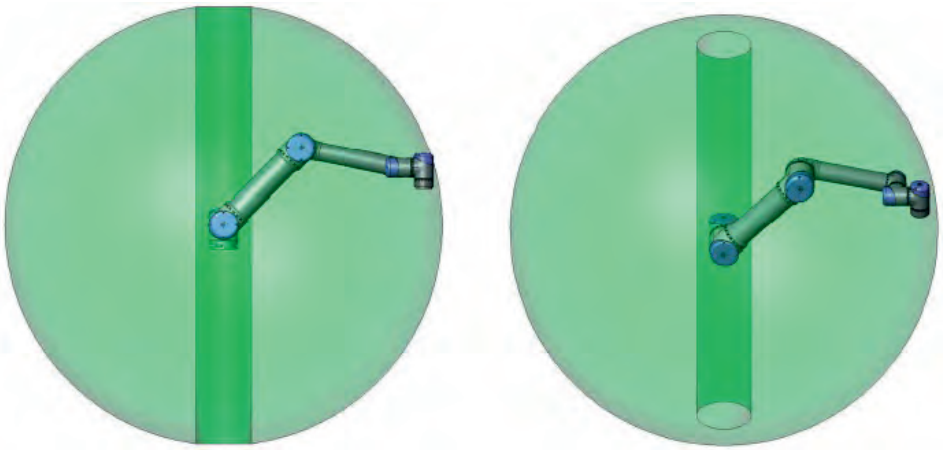


In der folgenden Abbildung sind sowohl Controller-E/A als auch der Werkzeuganschluss dargestellt.



4.1.1.4 Arbeitsbereich des Roboterarms (am Beispiel des UR5e)

Der Universal Robot UR5e ist ein kollaborativer Industrieroboterarm mit einer Traglast von 5 kg. Der Arbeitsraum eines UR5e bezieht sich auf den physischen Bereich, in dem der Roboterarm seine Aufgaben ausführen kann.
Der Arbeitsraum des UR5e wird durch seinen horizontalen und vertikalen Arbeitsbereich definiert. Der UR5e hat einen maximalen horizontalen Arbeitsbereich von bis zu 850 mm, gemessen vom Mittelpunkt der Basis bis zur Spitze des Handgelenks des Roboters. Im vertikaler Arbeitsbereich kann der UR5e seine Endeffektoren von der Basis bis zu einer Höhe von etwa 850 mm anheben.

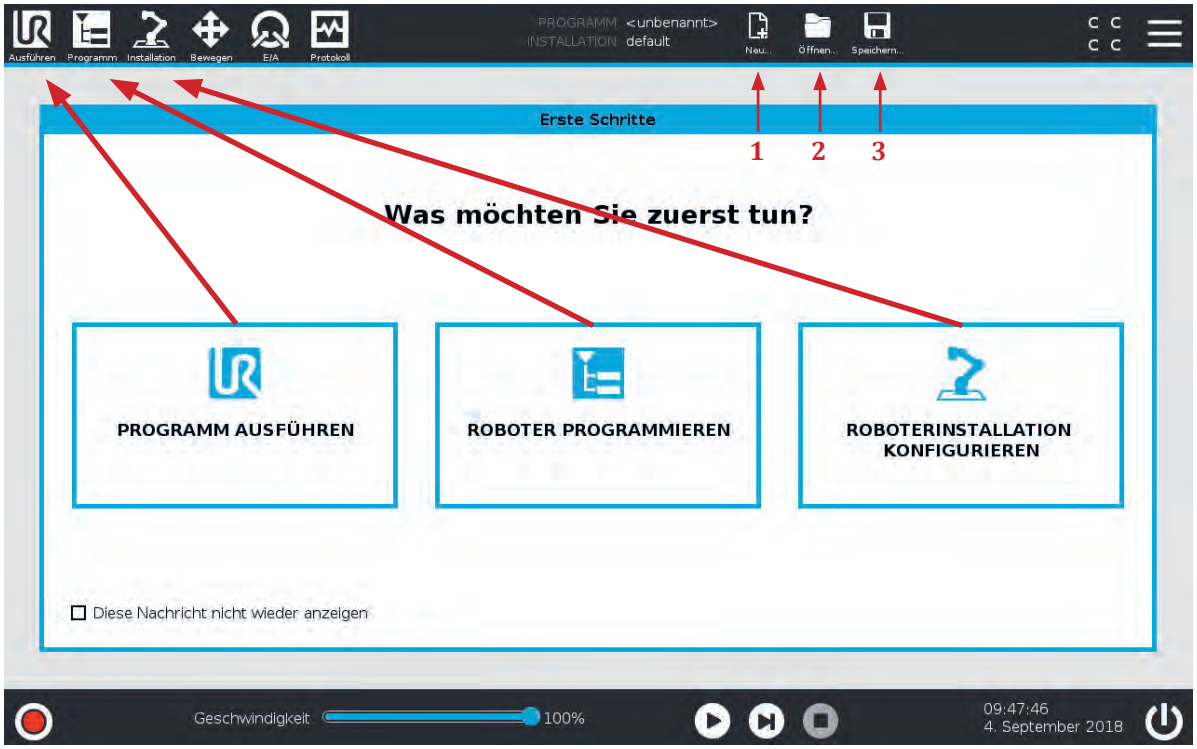


Front Geneigt

Die Begrenzungen des Arbeitsraums sind hauptsächlich auf die mechanische Konstruktion und die Reichweite des Roboterarms zurückzuführen. Dabei spielen Faktoren wie die Anzahl der Gelenke, die Länge der Segmente und die Drehwinkel der Gelenke eine entscheidende Rolle. Es ist wichtig, den Arbeitsraum des UR5e bei der Planung von Roboteranwendungen zu berücksichtigen, um sicherzustellen, dass der Roboter alle erforderlichen Aufgaben innerhalb seiner Reichweite ausführen kann.
Neben den physischen Begrenzungen des Arbeitsraums können auch externe Faktoren wie die Umgebung, in der der Roboter eingesetzt wird, und die Anwesenheit von Hindernissen den Arbeitsraum einschränken. Bei der Planung und Integration von Robotern sollten diese Faktoren berücksichtigt werden, um sicherzustellen, dass der Roboter effizient und sicher arbeiten kann.

4.1.1.5 Einführung in PolyScope

PolyScope ist die grafische Benutzerschnittstelle (GUI), durch die Sie den Roboter bedienen, vorhandene Roboterprogramme ausführen oder einfach neue Programme erstellen können. PolyScope wird durch den Touch-Screen am Steuergerät bedient.



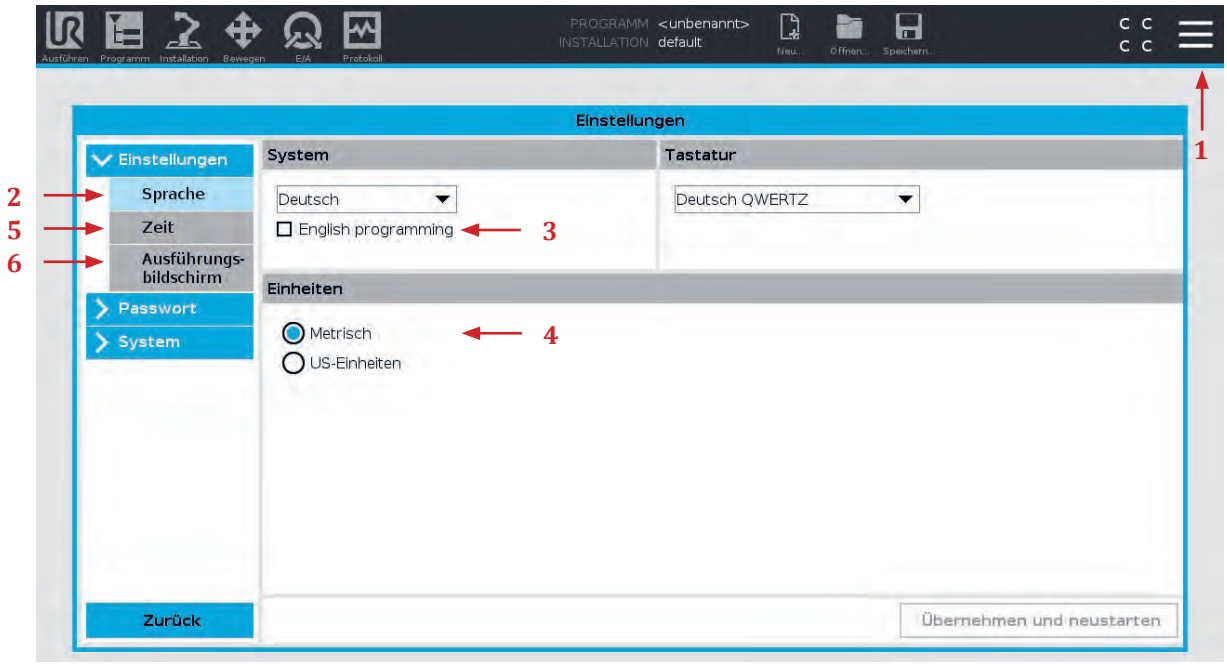
Nach dem Starten des Roboters wird der Startbildschirm angezeigt. Der Bildschirm bietet folgende Auswahloptionen an, welche sich auch in der dauerhaft eingeblendeten Statusleiste wiederfinden.

- **Programm ausführen**: Wählen Sie ein auszuführendes Programm. Dies ist die einfachste Art der Bedienung des Roboters, erfordert jedoch ein bereits erstelltes geeignetes Programm.
- **Roboter programmieren**: Ändern Sie ein Programm oder erstellen Sie ein neues Programm.
- **Roboterinstallation konfigurieren**: Konfigurieren Sie den Roboter, Endeffektor, Sicherheitseinstellungen, aktualisieren Sie die Software, usw.

Nr.	Modul	Beschreibung
1	**Neu**	Dient zum Erstellen eines neuen Programms und/oder einer Installation
2	**Öffnen**	Dient zum Laden eines Programms und/oder einer Installation
3	**Speichern**	bietet drei Möglichkeiten: • **Alles speichern**, um das aktuelle Programm und die Installation direkt zu speichern • **Programm speichern als**, um den Namen und das Verzeichnis für das neue Programm zu ändern • **Installation speichern als**, um den Namen und das Verzeichnis für die neue Installation zu ändern

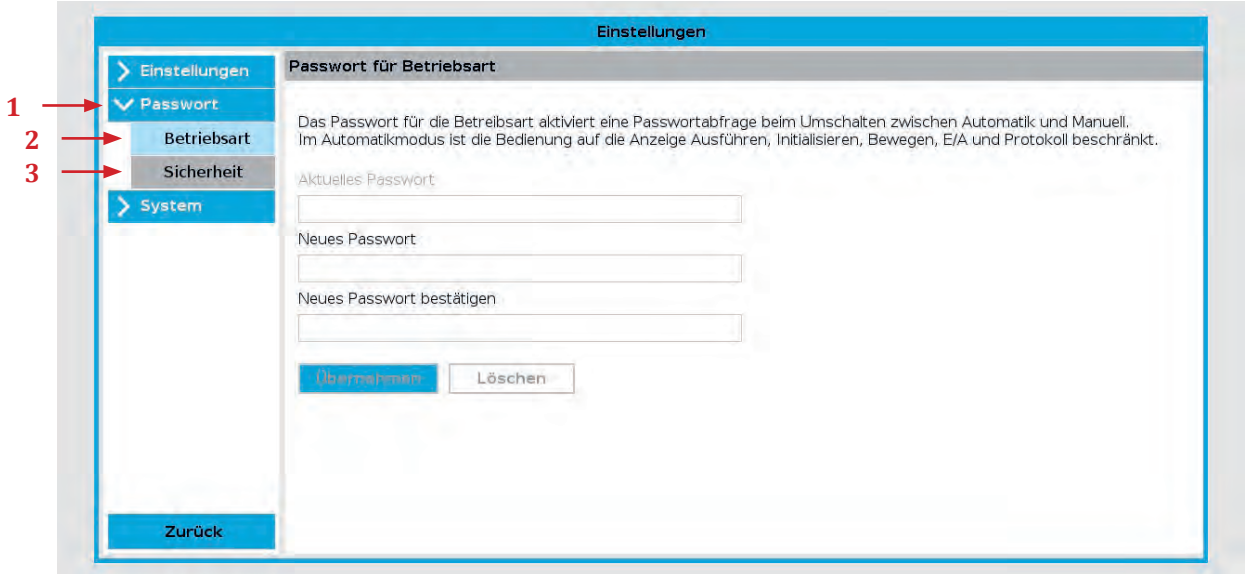
4.1.2 Einstellungen

4.1.2.1 Tab Einstellungen



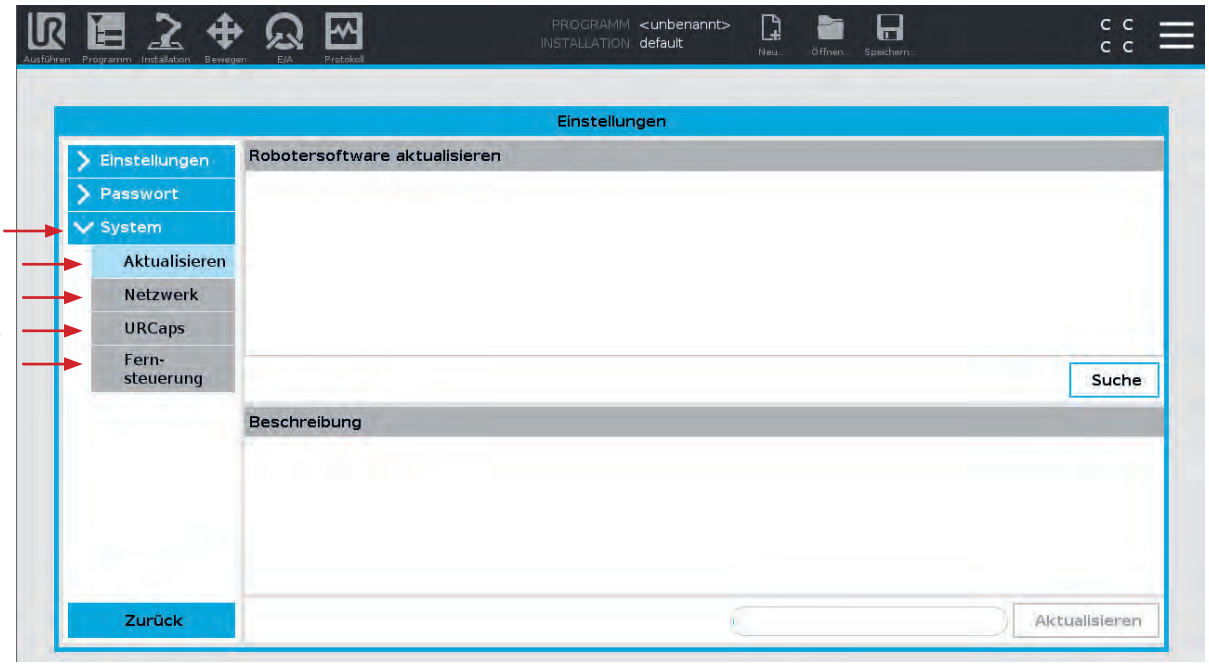
Nr.	Modul	Beschreibung
1	**Hamburger Menü**	Dient zum Aufrufen der PolyScope Hilfe, Infos und Einstellungen
2	**Sprachen**	Auswahl der Sprache (23 Sprachen)
3	**English programming**	Programmier-Befehle in englischer Sprache
4	**Einheit**	Auswahl der Einheit (Metrisch oder U.S.)
5	**Zeit**	Auswahl des Zeit- bzw. Datumsformats
6	**Ausführungsbildschirm**	Einblenden / Ausblenden Geschwindigkeitsreglers im Ausführungs-bildschirm

4.1.2.2 Tab Passwort



Nr.	Modul	Beschreibung
1	**Passwort**	Dient zur Änderung von den Passwörtern
2	**Betriebsart**	Erforderlich, um den Zugang zum Programmbereich zu begrenzen
3	**Sicherheit**	Erforderlich, um die Sicherheitseinstellungen anzupassen

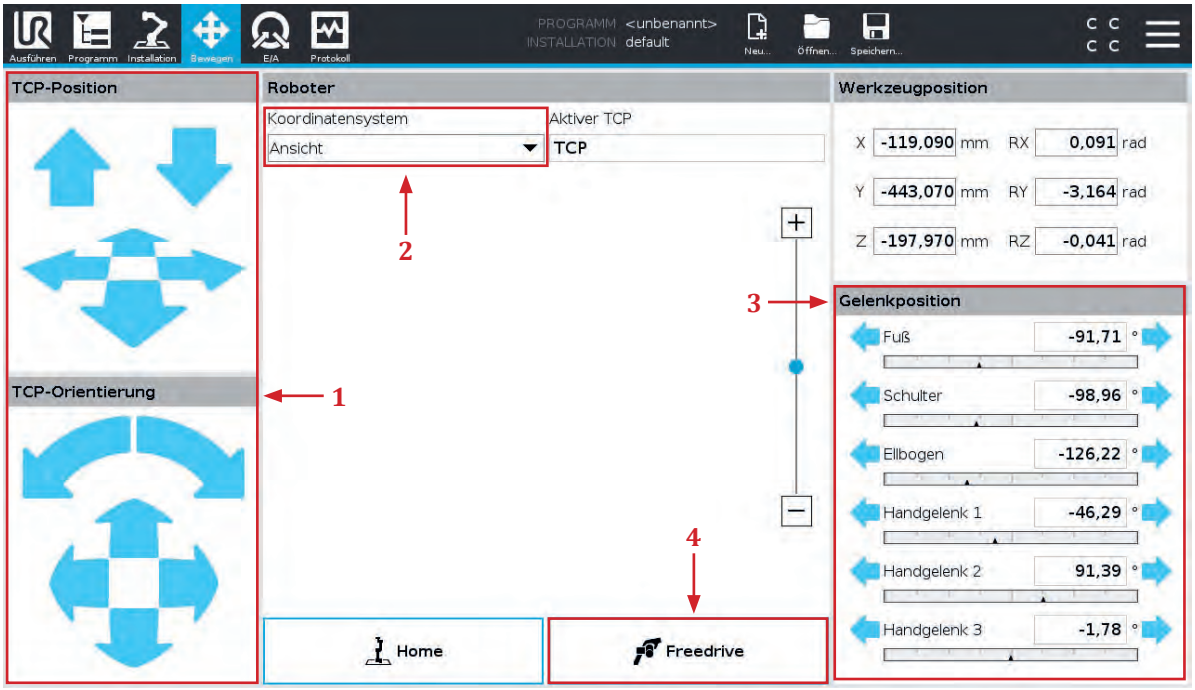
4.1.2.3 Tab System



Nr.	Modul	Beschreibung
1	**System**	Unter System findet man die Systemeinstellungen
2	**Aktualisieren**	Dient zur Aktualisierung der Robotersoftware, die auf der UR Support Seite gratis zu Verfügung stehen
3	**Netzwerk**	Netzwerkeinstellungen (IP-Adresse, Subnetzmaske, Standard-Gateway und DNS-Server)
4	**URCaps**	Dient zu Installation von Drittanbieter Softwarepaketen (Plugins)
5	**Fernsteuerung**	Erlaubt externe Ansteuerung bspw. über das Netzwerk

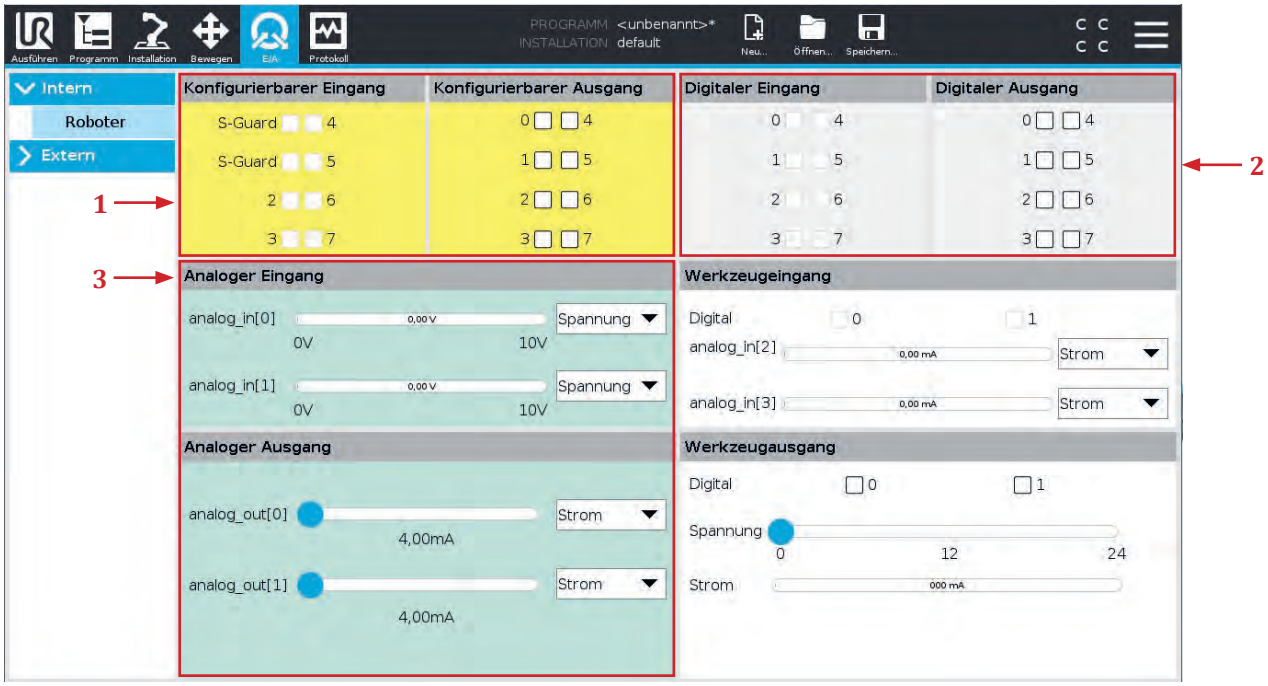
4.1.3 Register „Bewegen“

Mit diesem Bildschirm lässt sich der Roboter bewegen. Entweder durch Versetzung/Drehung des Roboterwerkzeuges oder durch Bewegung der einzelnen Robotergelenke.



Nr.	Modul	Beschreibung
1	**TCP-Position/TCP-Orientierung**	Bedienfeld dient zum Bewegen des TCP in kartesischen Koordinaten, Änderung der TCP-Orientierung (Drehung), Farbänderung der Knöpfe in Achsfarben mit Beschriftung durch Auswahl von nicht-Ansicht Koordinatensystemen
2	**Koordinatensystem**	Auswahl des aktiven Koordinatensystems. Beeinflusst die Bewegungsrichtung der TCP-Position. Auswahlmöglichkeiten: Ansicht, Basis und Werkzeug/Tool
3	**Gelenkpositionen**	Dient zur Bewegung der einzelnen Gelenken in Grad. Stellt aktive Gelenkbegrenzungen aus den Sicherheitseinstellungen dar.
4	**Freedrive**	Dient zur Aktivierung des Freedrive-Modus

4.1.4 Register „E/A“

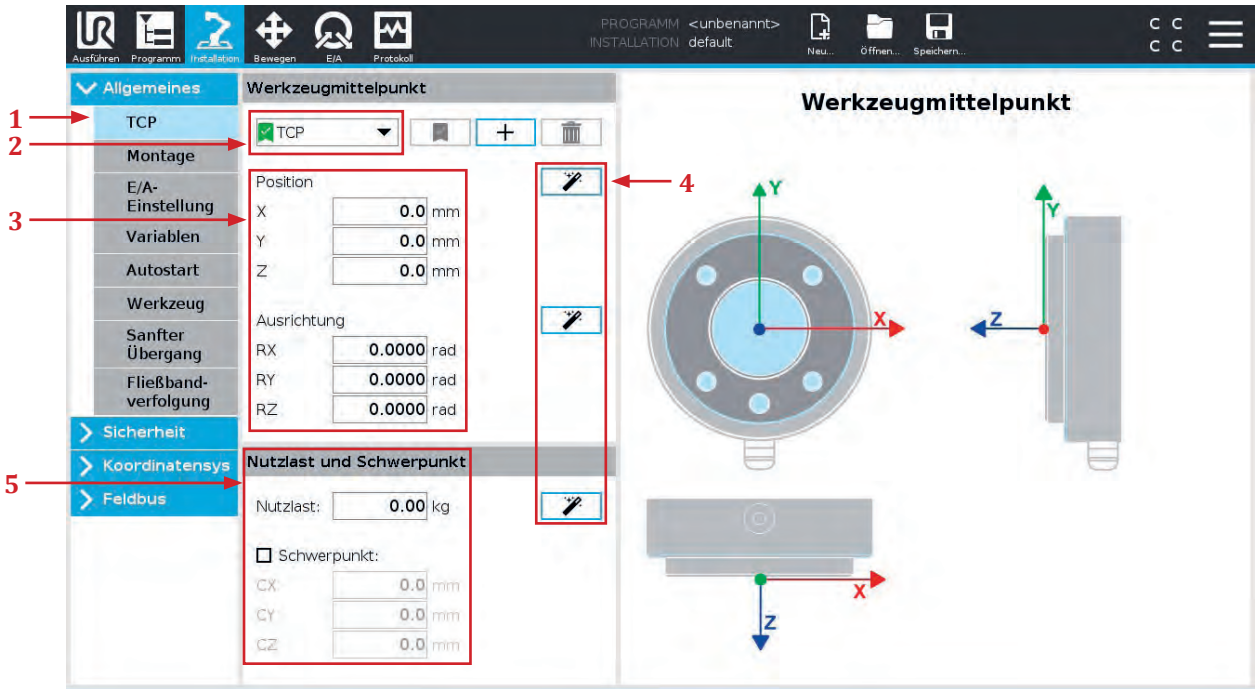


In diesem Bildschirm können die E/A-Signale überwacht und eingestellt werden. Der Bildschirm zeigt den aktuellen Zustand der Ein- und Ausgänge an, auch während der Programmausführung. Werden während der Ausführung des Programms Änderungen vorgenommen, so stoppt das Programm. Wenn ein Programm stoppt, behalten alle Ausgangssignale ihren Status bei. Die Signale werden auf dem Bildschirm wird mit 10 Hz aktualisiert, sodass ein sehr schnelles Signal eventuell nicht richtig angezeigt wird.

Nr.	Modul	Beschreibung
1	**Konfigurierbare E/A**	Konfigurierbare E/A können für spezielle Sicherheitseinstellungen reserviert werden. Reservierte E/A tragen den Namen der Sicherheitsfunktion statt des Standardnamens oder eines benutzerdefinierten Namens. Konfigurierbare Ausgänge, die für Sicherheitseinstellungen reserviert sind, können nicht bedient werden und werden nur angezeigt.
2	**Digitale E/A**	Die digitalen E/A verwenden 24 V und boolesche Ein- und Ausgabewerte (1/0). Die aktuellen Werte werden für mögliche spätere Neustarts des Controllers bei der Speicherung eines Programms gespeichert. Digitale Ausgänge lassen sich über Tippen schalten.
3	**Analoge E/A**	Die analogen E/A können entweder auf mit Strom [4-20 mA] oder Spannung [0-10 V] verwendet werden. Der Unterschied zu digitalen E/A ist, dass der Zustand eines analogen E/A einen Wertebereich und nicht nur einen der beiden booleschen Zustände abdeckt. Die aktuellen Werte werden für mögliche spätere Neustarts des Controllers bei der Speicherung eines Programms gespeichert. Auch hier lassen sich über Tippen die Ausgänge steuern.

4.1.5 Register „Installation“

Die Installation definiert die Roboterkonfiguration, dies umfasst neben vielen Einstellungen das verwendete Werkzeug, die Montage des Roboters, wie auch die elektrischen Verbindungen zu anderen Geräten. Diese Einstellungen können durch die verschiedenen Bildschirme unter der Registerkarte Installation festgelegt werden. Es ist möglich, mehr als eine Installationsdatei für den Roboter zu speichern.



Nr.	Modul	Beschreibung
1	**TCP**	Der Tool-Center-Point (TCP) ist der Punkt am Ende des Roboterarms, an dem sich das Werkzeug oder die Greifvorrichtung befindet.
2	**Dropdown-Menü-TCP**	Es kann mehr als ein TCP eingestellt werden
3	**Position/Ausrichtung**	Einstellung der XYZ- bzw. RXRYRZ-Werte gem. Darstellung
4	**Konfigurationsassistenten**	Mehrschrittige Assistenten zur einfachen Einstellung des TCP-Position, TCP-Ausrichtung und Nutzlast/Schwerpunktes
5	**Nutzlast und Schwerpunkt**	Dient zur Einstellung der Nutzlast bzw. Schwerpunktes des Werkzeugs

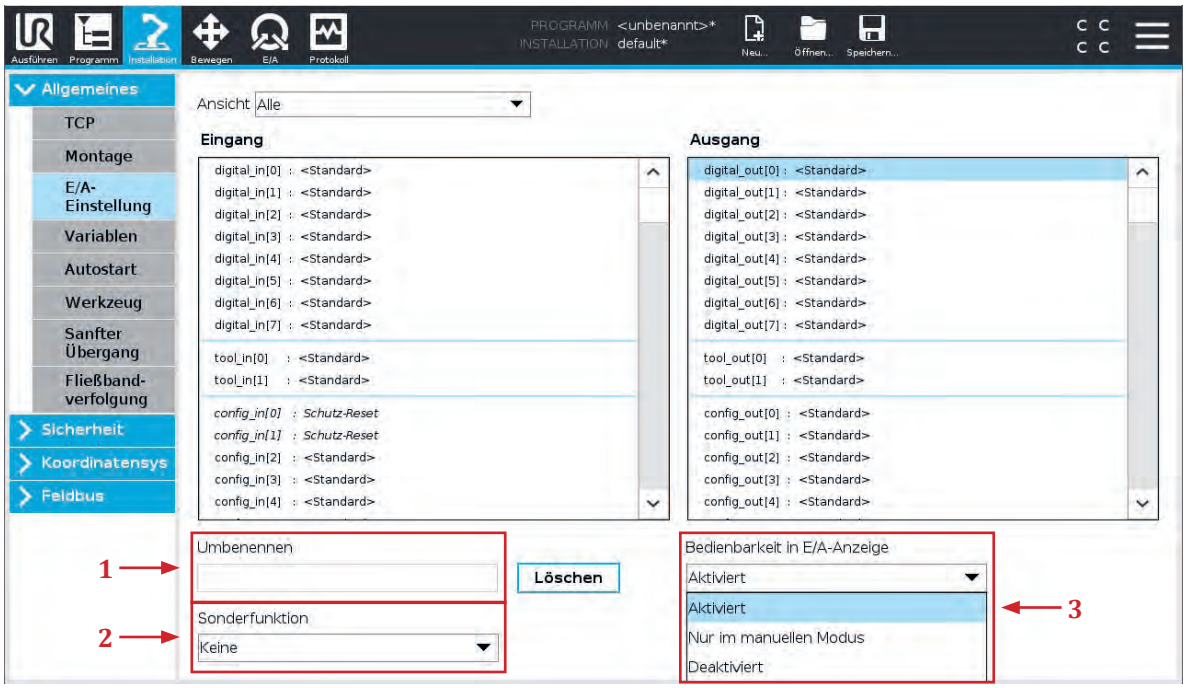
4.1.5.1 TCP-Konfigurationsassistent

Der Universal Robot Assistent zur Einstellung der Nutzlast und des Schwerpunkts ist hilfreich, da er den Einrichtungsprozess für den Roboter vereinfacht und automatisiert.
Durch die Verwendung des Universal Robot Assistenten zur Einstellung der Nutzlast und des Schwerpunkts kann der Einrichtungsprozess schneller und genauer durchgeführt werden, was zu einer höheren Produktivität und Qualität führt. Der Assistent führt den Benutzer durch eine Reihe von Schritten, um die genaue Position und Masse der Nutzlast zu bestimmen, indem mehrere Punkte angefahren und gemessen werden. Darüber hinaus kann die Verwendung des Assistenten die Wahrscheinlichkeit von Fehlern reduzieren, die durch eine manuelle Einstellung der Nutzlast und des Schwerpunkts verursacht werden können.

4.1.5.2 Montage

Sollte die Robotermontageposition von der klassischen Ausrichtung abweichen, muss in diesem Dialogfeld die Montageposition angegeben werden. Eine falsche Montageausrichtung kann zu ungenauer Bewegung, unsicherer Positionierung und sogar Schäden an der Umgebung oder dem Werkstück führen.

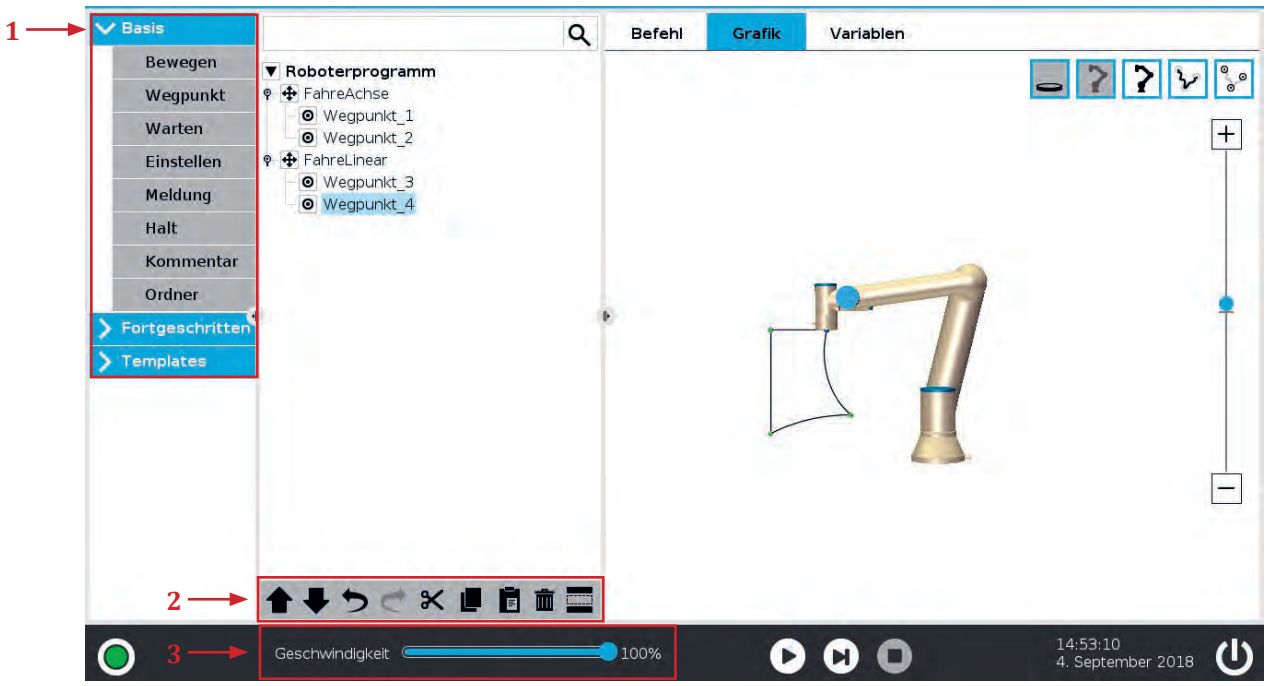
4.1.5.3 E/A-Einstellungen



Nr.	Modul	Beschreibung
1	**Umbenennen**	Eingangs- und Ausgangssignale können Namen gegeben werden. Dieser wird bei der Verwendung angezeigt
2	**Sonderfunktion**	Ermöglicht die Verarbeitung von Eingangs- und Ausgangssignalen durch vordefinierte Funktionen. Funktionen für Eingänge: • Programm starten • Programm stoppen • Programm pausieren • Freedrive (Bspw. mit einem Schalter) Funktionen für Ausgänge: • Low/High wenn Programm nicht aktiv • High, wenn aktiv-Low, wenn gestoppt • Low bei ungeplantem Stopp(, ansonsten High) • Kontinuierlicher Takt wenn Programm aktiv
3	**Bedienbarkeit in E/A-Anzeige**	Über diese Einstellung kann eingeschränkt werden, dass die Ausgänge über das Bedienfeld manuell über das TP gesetzt werden können. So kann bspw. verhindert werden, dass der Nutzer in den Programmablauf durch manuelles Setzten von Ausgängen eingreift.

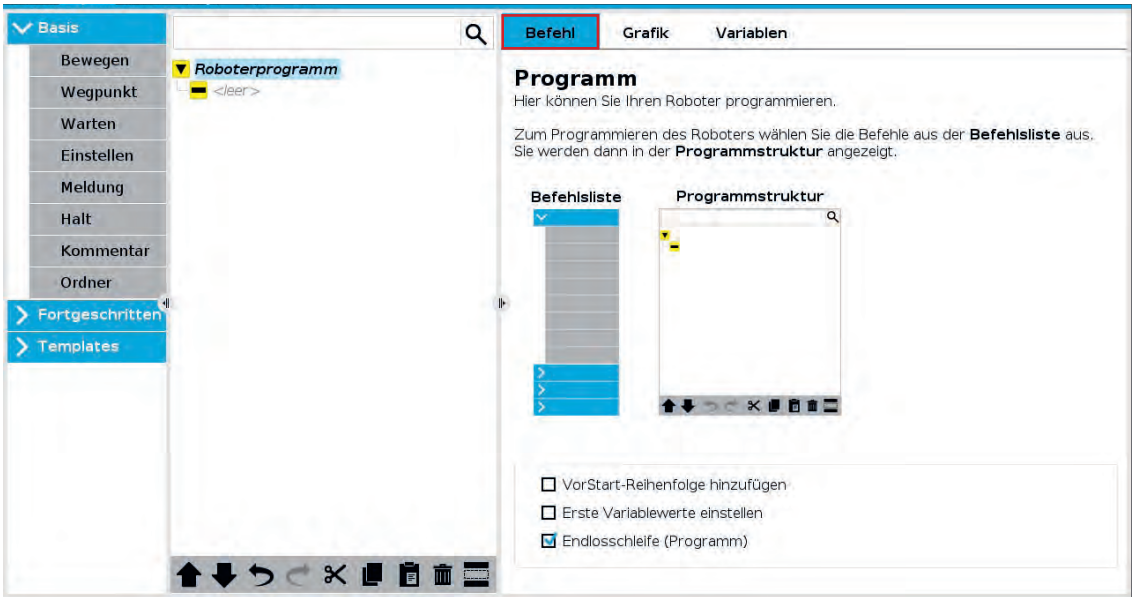
4.1.6 Register „Programm“

Der Register „Programm“ dient zur Erstellung von Roboterprogrammen. Auf der linken Seite finden sich Programmierbefehle, auf der rechten Seite sind mehrere Tabs dargestellt, die im Folgenden erklärt werden.



Nr.	Modul	Beschreibung
1	**Programmierbefehl**	Beinhaltet alle Befehle im PolyScope unterteilt in **Basis**, **Fortgeschritten** und **Templates/Assistenten**
2	**Bearbeitungsbereich**	Dient zu Bearbeitung und Organisierung des Programmbaumes. (Verschieben (hoch, runter), Rückgängig, Wiederherstellen, Ausschneiden, Kopieren, Einfügen, Löschen, Auskommentieren)
3	**Dashboard**	Dient zur Ausführung des Programms und Einstellung der Geschwindigkeit. Auch schrittweise ausführen, Pausieren und Stoppen ist möglich.

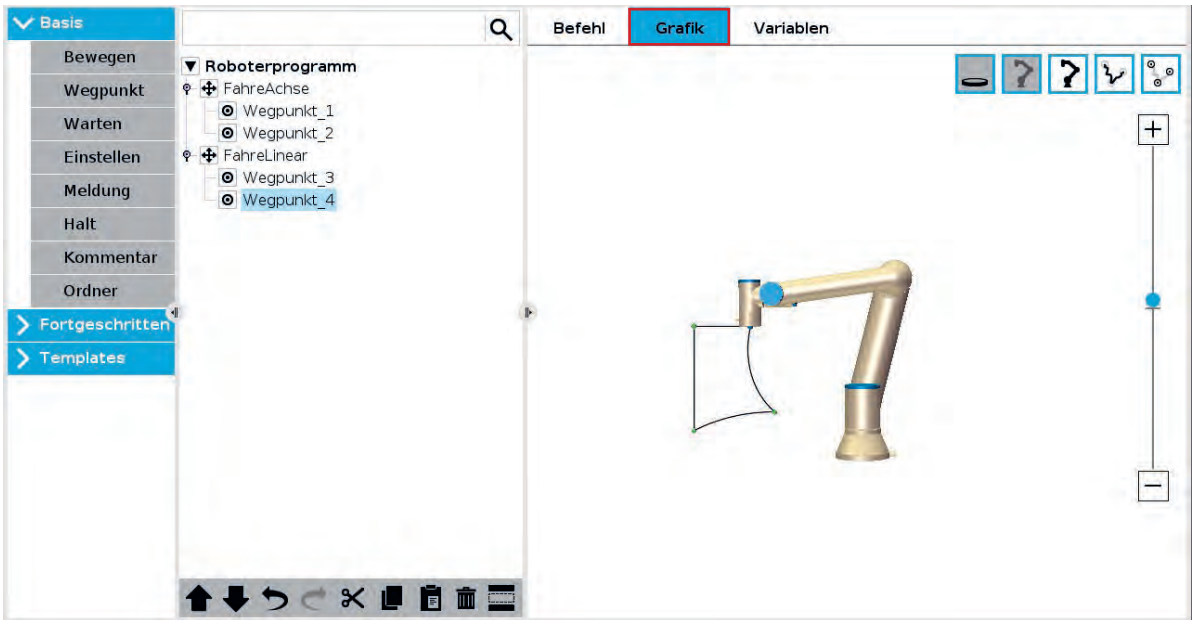
4.1.6.1 Tab Befehl



Der in der Regel vorausgewählte Tab **Befehl** befindet sich auf der rechten Seite im Register Programm. Durch Tippen kann zwischen Befehl, Grafik und Variablen gewechselt werden. Im Tab Befehl können die Inhalte und Einstellungen des jeweils ausgewählten Programmknoten konfiguriert werden. Grundsätzlich ist eine leere Programmstruktur nicht erlaubt. Programme mit falsch oder nicht konfigurierten Programmknoten können nicht ausgeführt werden. Ungültige Programmknoten werden gelb hervorgehoben.

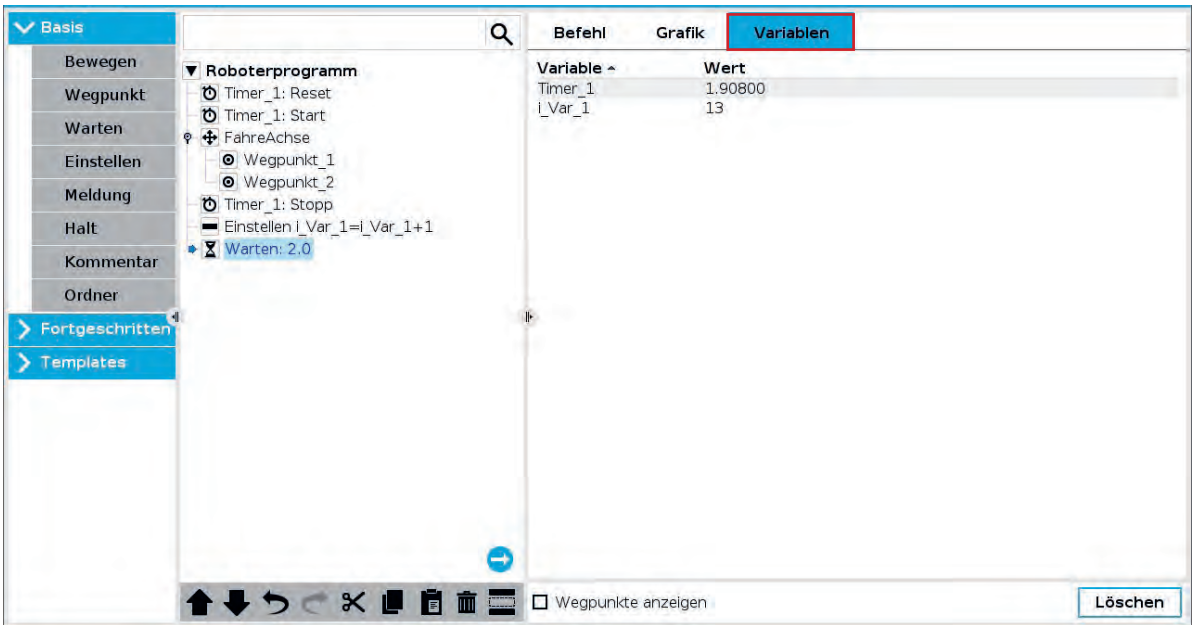
4.1.6.2 Tab Grafik

Der Tab Grafik bietet eine grafische Darstellung des aktuellen Roboterprogramms. Der Weg des TCP wird in einer 3D-Ansicht gezeigt, mit schwarzen Bewegungssegmenten und grünen Übergangssegmenten (Übergänge zwischen den Bewegungssegmenten). Die grünen Punkte zeigen die Positionen der Wegepunkte im Programm. Über die Auswahlfelder im oberen rechten Bereich lassen sich Teile der 3D Darstellung Ein- und Ausblenden.



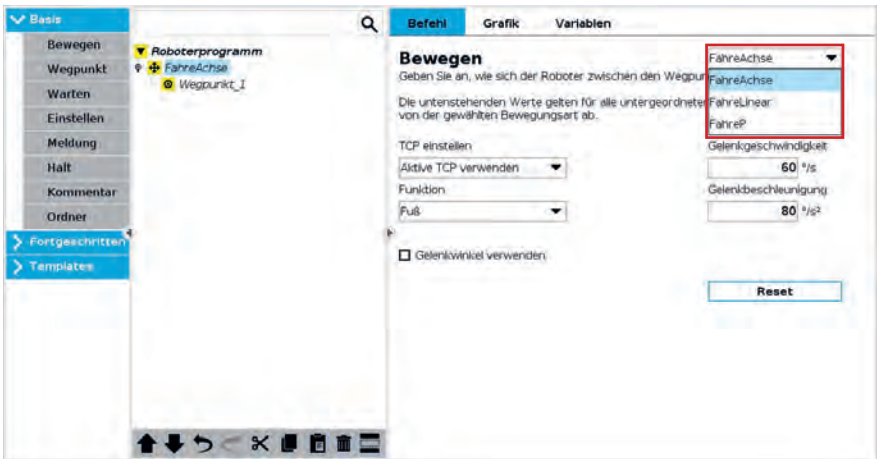
4.1.6.3 Tab Variablen

Im Tab Variablen lassen sich Variablen, auch während der Programmausführung, mit ihren aktuellen Werten anzeigen. Mehr zu Variablen im Kapitel Fortgeschrittene Befehle.



4.1.7 Grundbefehle

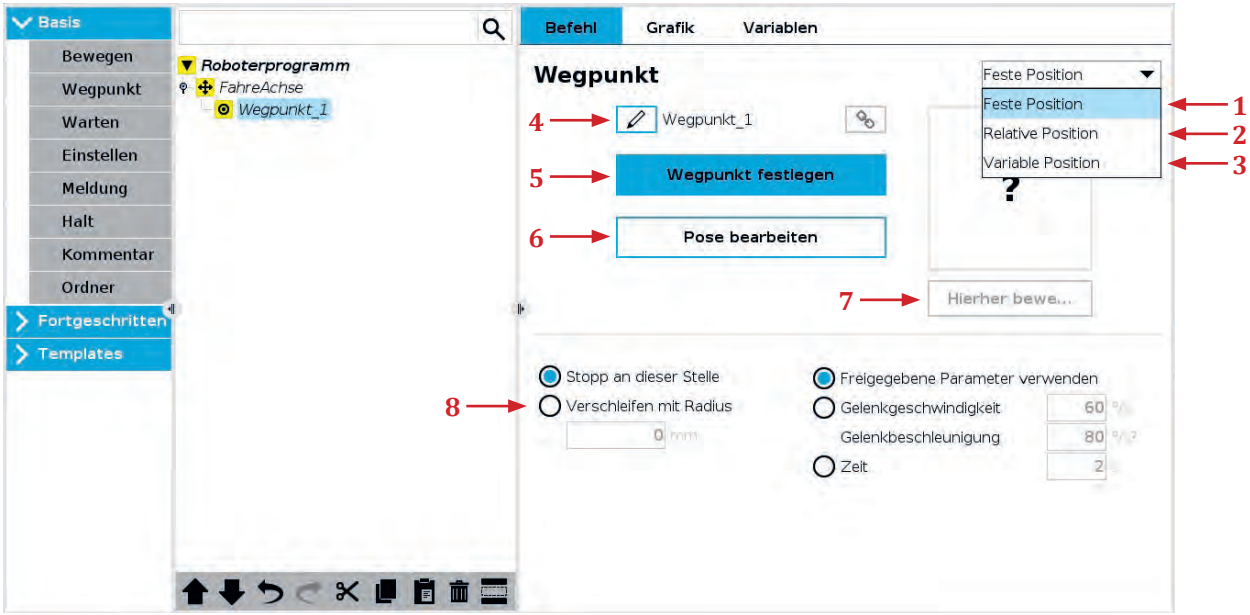
4.1.7.1 Bewegen



Über den Befehl Bewegen kann eine Roboterbewegung zum Programm hinzugefügt werden. Wird dieser Befehl ausgewählt, wird automatisch ein Bewegen-Programmknoten mit einem Wegpunkt erzeugt. Hinweis: Die für Bewegungen konfigurierten Beschleunigungen und Geschwindigkeiten können beliebig gesetzt werden, sind aber in jedem Fall durch die aktuellen Sicherheitseinstellungen limitiert.
Folgende drei **Bewegungsarten** stehen zur Auswahl:

- **FahreAchse** führt eine Gelenkwinkelbewegung aus. Jedes Gelenk des Roboters erhält eine Zielstellung, die von jedem Gelenk mit der maximal zulässigen Beschleunigung und Geschwindigkeit angefahren werden. Die führt in der Regel zu einer gekrümmten Trajektorie und ist die schnellste Bewegungsart.
- **FahreLinear** sorgt für eine lineare Bewegung des Werkzeugmittelpunkts (TCP) zwischen Wegpunkten. Dies bedeutet, dass für jedes Gelenk kontinuierlich Zwischenschritte interpoliert werden.
- **FahreP** bewegt den TCP auch linear jedoch zusätzlich bei konstanter Geschwindigkeit. Anwendungsbereiche sind hier bspw. Kleben oder Schweißen.
- **Kreisbewegung (MoveC)** kann zu einem FahreP hinzugefügt werden, um eine Kreisbewegung zu erzeugen. Der Roboter beginnt die Bewegung von seiner aktuellen Position oder seinem Startpunkt aus, bewegt sich durch einen auf der Kreisbahn definierten Durchgangspunkt und einen Endpunkt, der die Kreisbewegung vollendet.

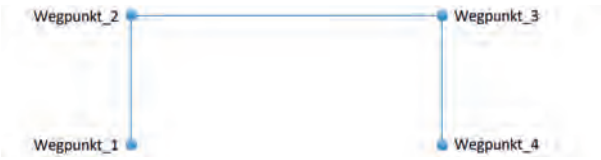
4.1.7.2 Wegpunkt



Nr.	Modul	Beschreibung
1	**Feste Position**	Ein Wegepunkt mit einer fixen Position wird gespeichert. Bspw. indem der Roboterarm physisch in die entsprechende Position bewegt wird.
2	**Relative Position**	Eine relative Position wird durch zwei feste Positionen definiert. Eine Bewegung mit einer relativen Position bewegt den Roboterarm um die Differenz der beiden festen Positionen. Immer ausgehend von der aktuellen Position des Roboters. Damit lassen sich Bewegungen wie z.B. „Zwei Zentimeter nach links“ ausführen.
3	**Variable Position**	Ein Wegpunkt, der durch eine Pose-Variable definiert ist. Hiermit kann bspw. innerhalb des Programms die Position berechnet werden. Eine Pose-Variable folgt folgender Synatx: p[x,y,z,rx,ry,rz]
4	**Punkt umbenennen**	Wegepunkte erhalten automatisch einen Namen. Der Name kann durch den Benutzer geändert werden.
5	**Wegpunkt festlegen**	Dient zu Festlegung des angelernten Punktes
6	**Pose bearbeiten**	Position anpassen, POSE-Editor
7	**Hierher bewegen**	Bewegt den Roboter zu dieser Position
8	**Verschleifen**	Verschleifradien ermöglichen an Wegpunkten in einer kontinuierlichen Kurvenbewegung vorbeizufahren, ohne die Geschwindigkeit zu verringern oder anzuhalten. Allerdings sind Verschleifradien nicht in allen Situationen sinnvoll oder anwendbar, beispielsweise wenn der Roboter ein Werkstück senkrecht aufnehmen soll, da in diesem Fall ein Verschleifradius die Applikation unbrauchbar machen würde.

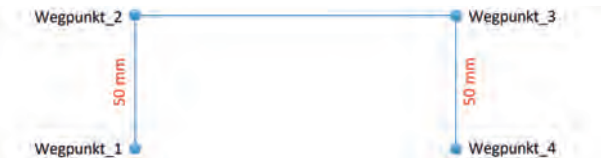
4.1.7.3 Ausprobieren: Erste Wegpunkte

Aufgabe: Entnehme ein Objekt aus einem Regal und lege es auf einen Platz im Arbeitsraum des Roboters. Wie viel Wegpunkte werden benötigt? Welche Bewegungsart eignet sich am besten?
Lösung: Es werden 4 Wegpunkte benötigt. Für das erste und letzte Wegstück eignet sich eine Linearbewegung am besten, für das mittlere Wegstück kann seine Gelenkwinkelbewegung verwendet werden. Für dieses unspezifische Beispiel gibt es jedoch kein klares richtig oder falsch. Wichtig ist die Vor- und Nachteile der Bewegungsarten zu betrachten.



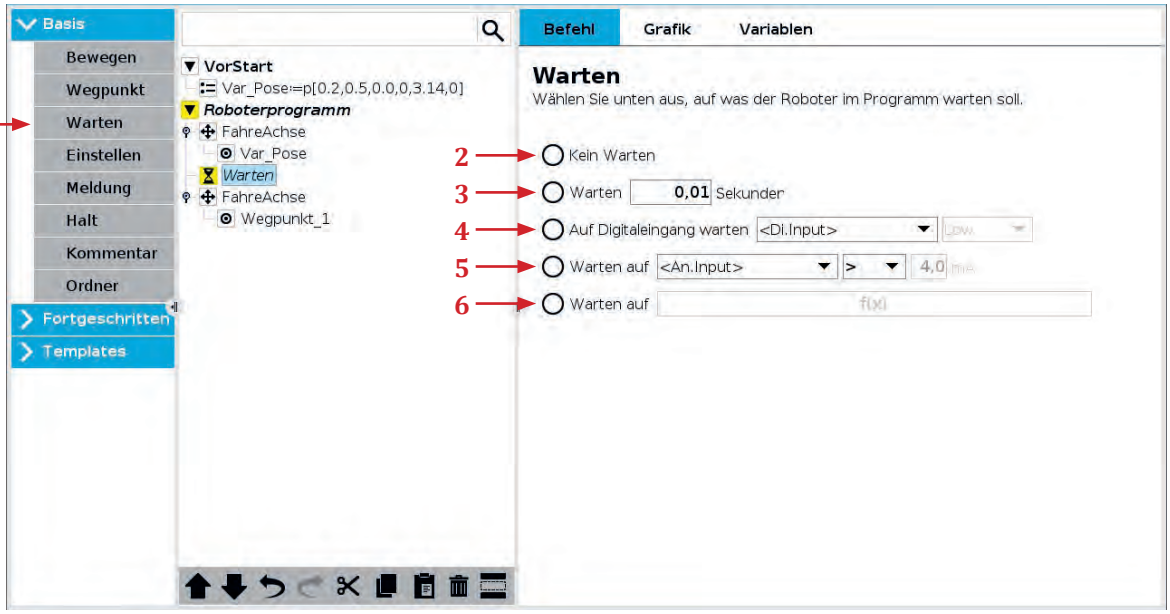
4.1.7.4 Ausprobieren: Definierte Wegstrecken

Aufgabe: Entnehme ein Objekt aus dem Regal und lege es auf einen Platz im Arbeitsraum des Roboters. Verwende einen Verschleifradius von 20 mm für Wegpunkt 2 und 3. Verwende eine Aufnahmehöhe von 50 mm für Wegpunkt 2 und 3, benutze dafür den POSE-Editor.



Hinweis: Der POSE-Editor ist über den Knopf Pose bearbeiten bei markiertem Wegpunkt erreichbar. Über das Verwenden der Plus und Minus Knöpfe können definierte Wegstrecken oder Winkel zur aktuellen Position hinzugefügt oder entfernt werden. Wichtig: Wähle dabei das korrekte Koordinatensystem. Das Koordinatensystem Ansicht entspricht der 3D-Darstellung des Roboters und verändert beim Drehen der 3D-Ansicht die Koordinaten entsprechend der Darstellung. Für definierte Verfahrbewegungen ist ein festes Koordinatensystem empfehlenswert.

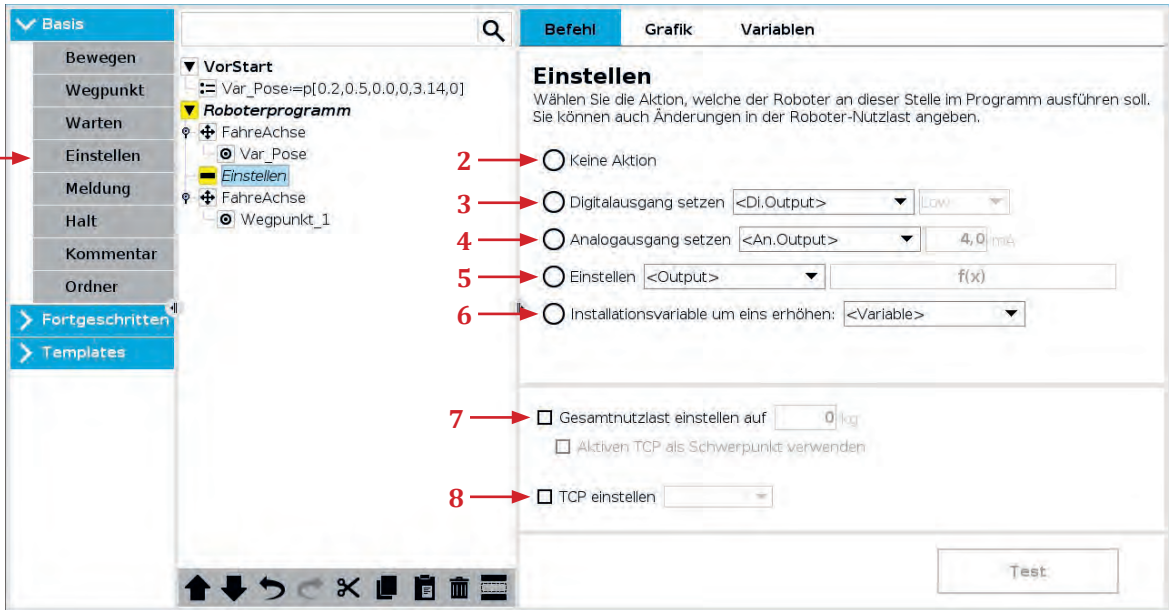
4.1.7.5 Befehl: Warten



Das Programm verbleibt auf dem Warten Befehl. Je nach Konfiguration für eine gewisse Zeit oder bis zum Eintreten eines Ereignisses.

Nr.	Modul	Beschreibung
1	**Warten**	Befehl hinzufügen
2	**Kein Warten**	Es wird nicht gewartet
3	**Warten**	Warten um X Sekunden
4	**Auf Digitaleingang warten**	Auf einen Digitaleingang, der aus dem Dropdown-Menü ausgewählt wird, warten
5	**Warten auf (Analogeingang)**	Auf einen Analogeingang, der aus dem Dropdown-Menü ausgewählt wird, warten
6	**Warten auf (Funktion)**	Warten auf selbst definierte Funktion. Komplexe Logik möglich

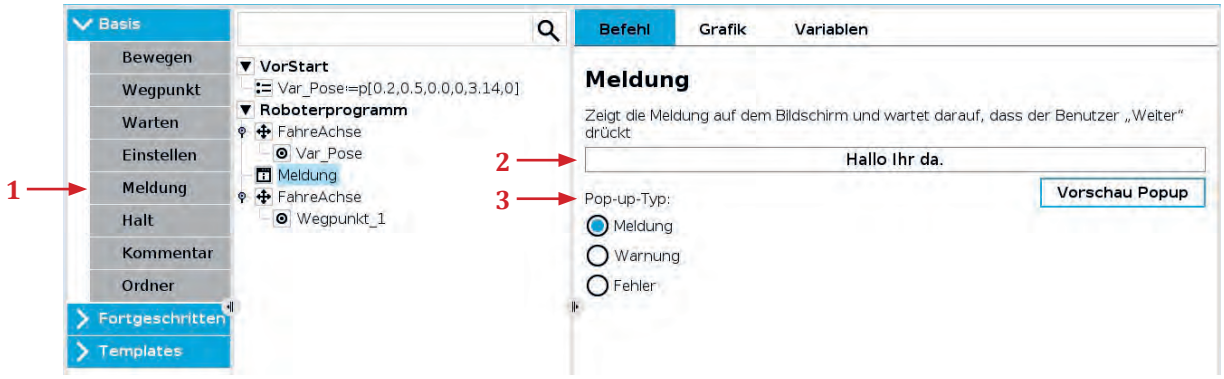
4.1.7.6 Befehl: Einstellen



Mit dem Einstellen-Befehl können unterschiedliche Aktionen ausgeführt werden.

Nr.	Modul	Beschreibung
1	**Einstellen**	Befehl hinzufügen
2	**Keine Aktion**	Befehl hat keine Funktion
3	**Digitalausgang setzen**	Einen Digitalausgang ansteuern (Low / High)
4	**Analogausgang setzen**	Einen Analogausgang setzen
5	**Einstellen**	Einem Digital- bzw. Analogausgang über eine Funktion setzten
6	**Installationsvariable um eins erhöhen**	Dient dazu eine Installationsvariable als Zähler zu verwenden, indem man diese um eins erhöht.
7	**Gesamtnutzlast einstellen auf**	Dient zur Einstellung der aktuellen Nutzlast bspw. nach dem Aufheben eines Werkstücks.
8	**TCP einstellen**	Änderung des TCP nach bspw. Werkzeugwechsel

4.1.7.7 Befehl: Meldung



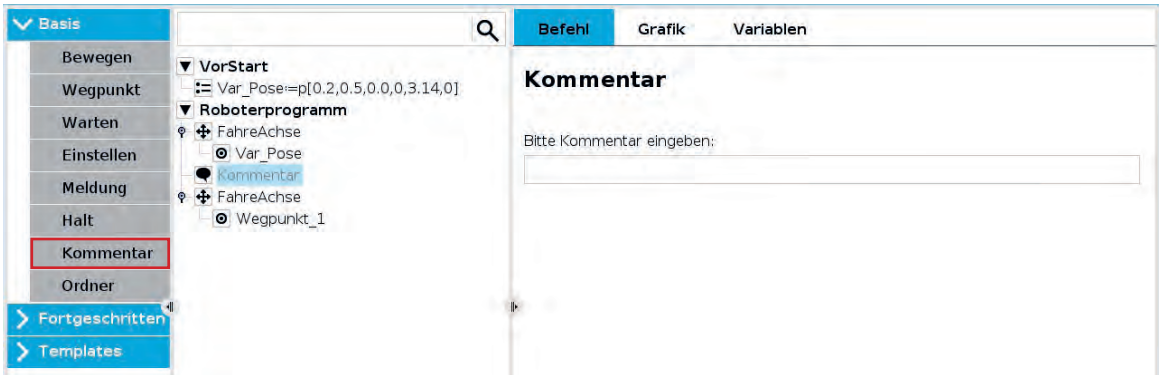
Meldung wir in Form eines Popups angezeigt. Programmablauf wird pausiert, bis Nutzerinteraktion erfolgt hat.

Nr.	Modul	Beschreibung
1	**Meldung**	Befehl hinzufügen
2	**Textfeld**	Meldungstext
3	**Pop-up-Typ**	Icon, welches neben der Meldung angezeigt wird (Meldung (blaues i), Warnung (gelbes Warndreieck) und Fehler (rotes X)

4.1.7.8 Befehl: Halt

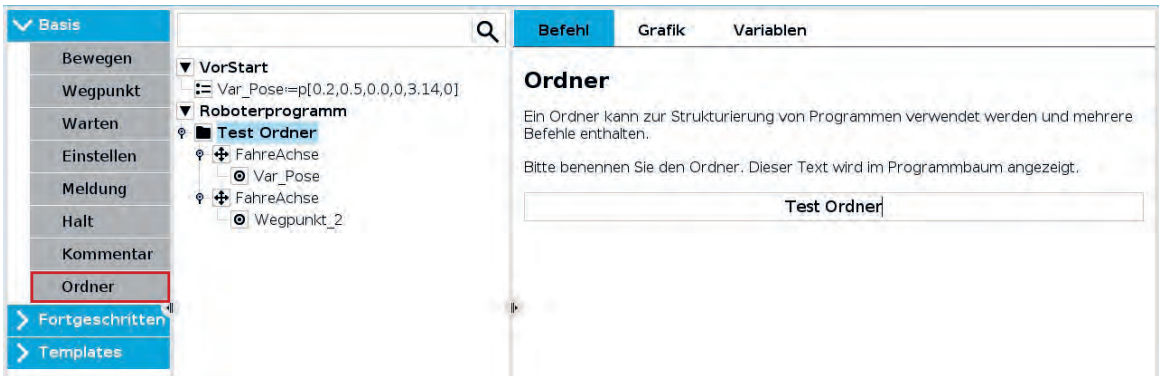
Die Ausführung des Programms wird an dieser Stelle beendet.

4.1.7.9 Befehl: Kommentar



Hier erhält der Programmierer die Möglichkeit, das Programm durch einen Kommentar zu ergänzen. Dieser hat auf die Ausführung des Programms keinerlei Auswirkung.

4.1.7.10 Befehl: Ordner

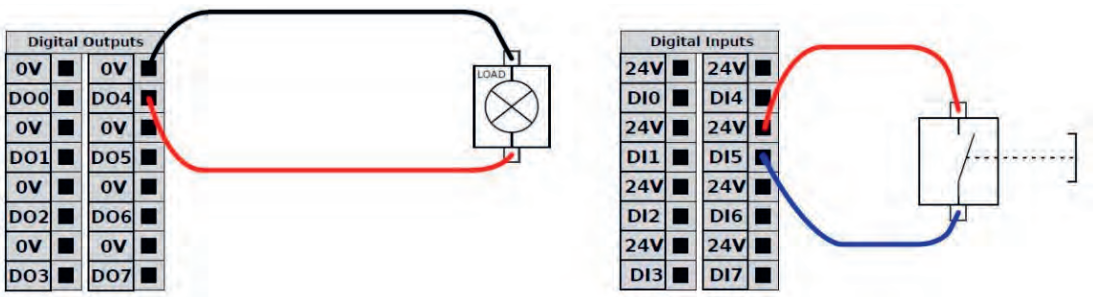


Ein Ordner kann zur Organisation und zur besseren Lesbarkeit des Programms eingesetzt werden. Ordner sind rein visuell und haben keine Auswirkungen auf das Programm und seine Ausführung.

4.1.7.11 Ausprobieren: Einfaches Programm mit E/A-Signalen

Vorbereitung:

- Benötigtes Material: Lampe und Taster (In der Standard UR Schulungszelle vorgesehen), andernfalls extern anschließen
- Benenne *digital_out[4]* zu "Lampe" und *digital_in[5]* zu "Taster"
- Verbinde die Lampe mit Digital Output 4 im Steuergerät, wie unten gezeigt
- Verbinde den Taster mit Digital Input 5 im Steuergerät, wie unten gezeigt



Aufgabe:
Erstelle die Aufgabe in folgenden drei Ordnern.

Ordner - Trigger
- Erstelle einen Ordner mit der Bezeichnung ”Trigger”
- Füge einen Warten-Befehl ein und warte hier auf das Signal des Tasters
- Sobald der Taster betätigt wurde, schalte die Lampe ein

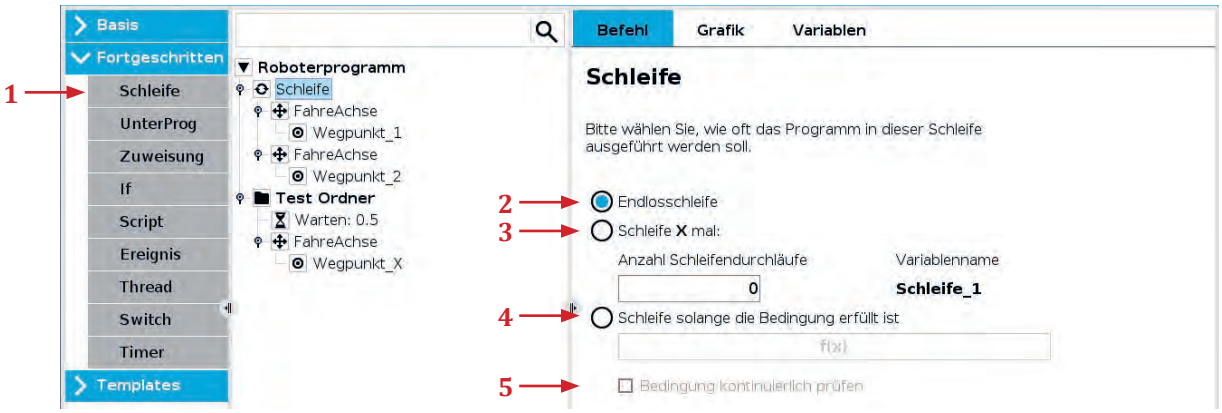
Ordner - PickUp
- Erstelle einen Ordner mit der Bezeichnung ”PickUp”
- Fahre in diesem Ordner auf WP1 und dann linear nach unten auf WP2
- An WP2 Greifer schließen und Teil aufnehmen dann wieder auf WP1

Ordner - Place
- Erstelle einen Ordner mit der Bezeichnung ”Place”
- Fahre auf WP 3 und dann linear nach unten auf WP4
- Öffne den Greifer und lege das Werkstück an der Position ab, dann zurück auf WP3
- Schalte die Lampe wieder aus

Musterlösung zum Download unter robospace.de/ur oder alternativ in der niedersächsischen Bildungscloud.

4.1.8 Fortgeschrittene Befehle

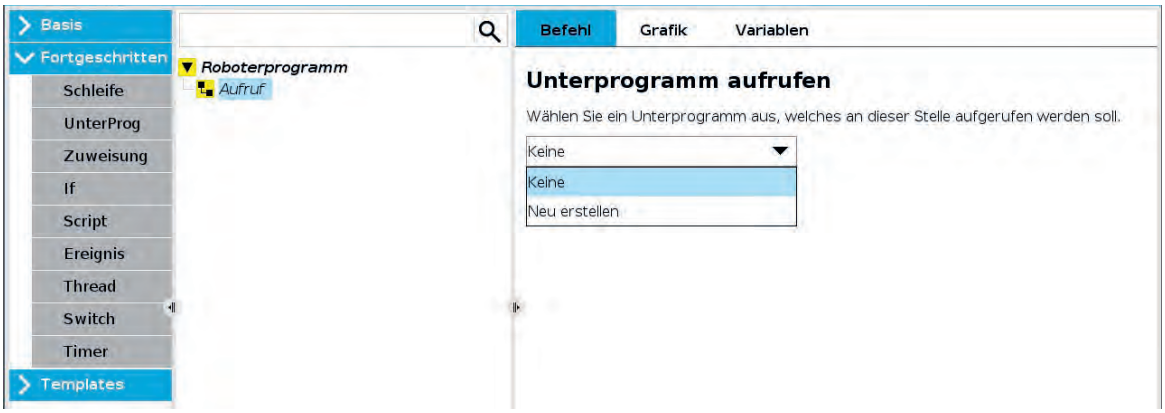
4.1.8.1 Befehl: Schleife



Eine Schleife in der Programmierung ist eine Struktur, die es ermöglicht, bestimmte Anweisungen oder Codeblöcke wiederholt auszuführen, solange eine festgelegte Bedingung erfüllt ist. Dadurch können repetitive Aufgaben effizienter durchgeführt werden, ohne den Code mehrfach schreiben zu müssen.

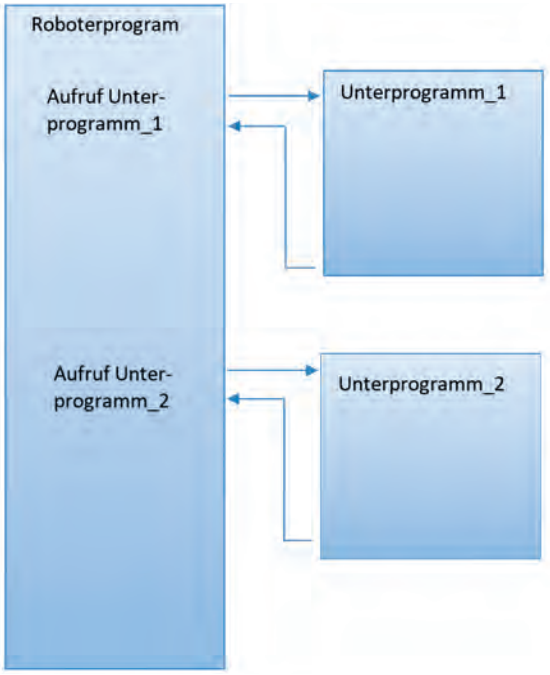
Nr.	Modul	Beschreibung
1	**Schleifen**	Befehl hinzufügen
2	**Endlosschleifen**	Die Schleife wir endlos ausgeführt. Hinweis: Manuelle Abbruchbedingung nötig
3	**Schleife X mal**	Schleife wird um die angegebene Anzahl wiederholt
4	**Schleife, solange die Bedingung erfüllt ist**	Schleife wird wiederholt, bis die Bedingung aus dem Funktionsfeld erfüllt ist
5	**Bedingung kontinuierlich prüfen**	Entgegen der Schleifenlogik aus herkömmlichen Programmiersprachen ist es durch Auswahl dieses Feldes möglich die Abbruchbedingung der Schleife dauerhaft überwachen zu lassen. So wird nicht mehr nur nach jedem vollständigen Schleifendurchlauf die Bedingung geprüft, sondern dauerhaft. Dies hat zur Folge, dass eine Schleife in dem Zeitpunkt verlassen wird, in dem die Bedingung erfüllt wird. Andernfalls würde der aktuelle Schleifendurchlauf zunächst beendet werden.

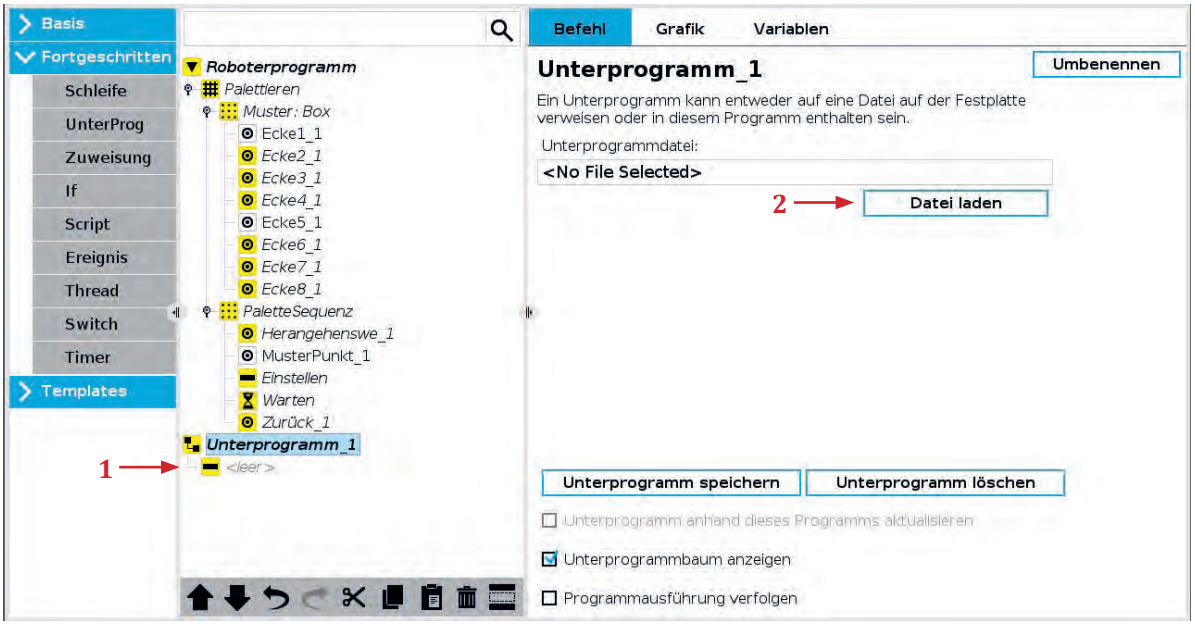
4.1.8.2 Befehl: Unterprogramme



Durch den Befehl Unterprogramm lassen sich bereits gespeicherte Programme in das aktuelle Programm importieren. Ähnlich wie bei Ordnern ist einer der Vorteile, dass sich das Programm dadurch organisieren lässt. Ein Unterscheidungsmerkmal im Vergleich zu Ordnern ist, dass Unterprogramme auf dem Dateisystem des Roboters gespeichert werden können und müssen.

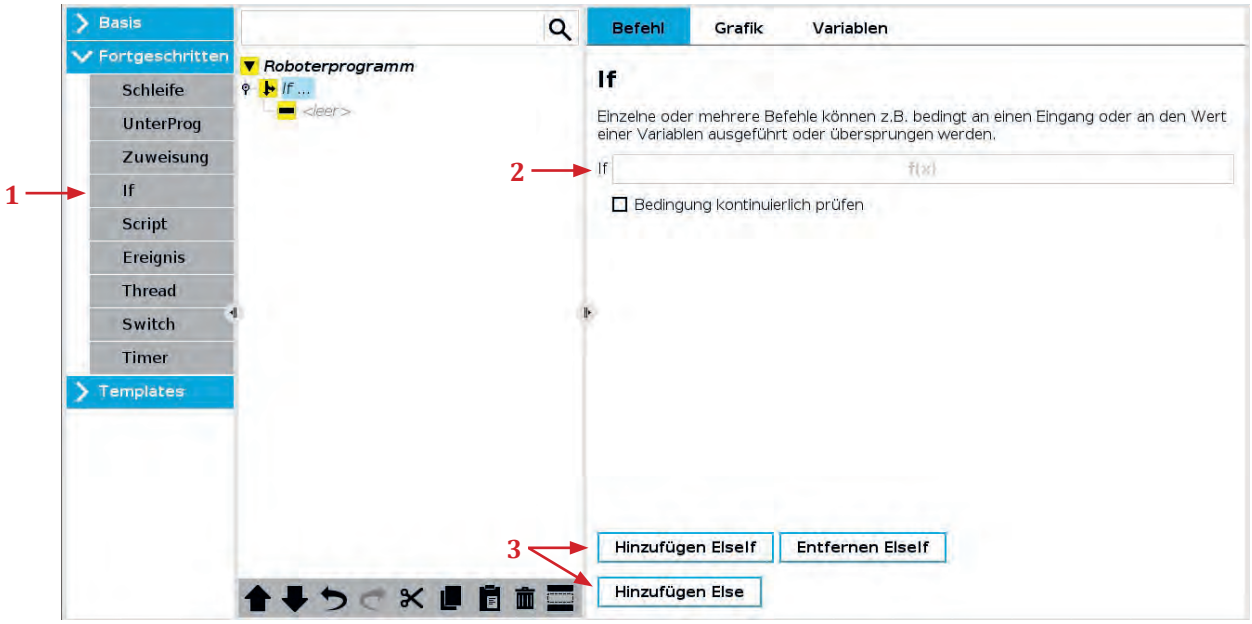
So wird es möglich Programmteile in unterschiedlichen anderen Programmen erneut zu verwenden. Der Aufruf eines Unterprogramms innerhalb eines Unterprogramms ist nicht möglich. Die Ausführung des Unterprogramms erfolgt sequenziell, das heißt das Hauptprogramm wird für das Unterprogramm verlassen und danach fortgesetzt. Folgende Grafik zeigt den Aufruf eines Unterprogramms.





Nr.	Modul	Beschreibung
1	**Leer**	Unterprogramm konfigurieren
2	**Datei laden**	Existierendes Programm laden

4.1.8.3 Befehl: Verzweigung (If...else)



"If-else" ist eine bedingte Anweisung in der Programmierung, die den Programmfluss abhängig von einer bestimmten Bedingung steuert. Wird die Bedingung erfüllt, wird der "if"-Block ausgeführt, andernfalls wird der "else"-Block ausgeführt.

Nr.	Modul	Beschreibung
1	**If-Befehl**	Befehl hinzufügen
2	**Ausdruckeditor**	Hier wird die Bedingung definiert. Die Bedingung muss so gestaltet werden, dass entweder wahr oder falsch als Ergebnis entsteht.
3	**Hinzufügen ...**	Hiermit können beliebig viele "ElseIf" hinzugefügt werden. "ElseIf" ist eine erweiterte bedingte Anweisung, die nach einem "If" und vor einem "Else" verwendet wird, um zusätzliche Bedingungen zu prüfen und den Programmfluss entsprechend zu steuern. Dazu ist auch das Hinzufügen eines Else möglich.

4.1.8.4 Ausprobieren: Unterprogramme, Schleifen und Bedingungen

Benötigtes Material: 2x Schalter (angeschlossen an die E/A), 1x Lampe (angeschlossen an die E/A), in der Standard UR Schulungszelle bereits vorhanden
Aufgabe:

- Schließe alle Hardware an die E/A an und benenne die Geräte korrekt (Schalter 1, Schalter 2, Lampe)
- Erstelle ein einfaches Programm, welches mithilfe einer Schleife die Lampe 5-mal aufleuchten lässt.
- 0,5 Sekunden an, dann für 0,5 Sekunden aus
- Speichere das Programm als *Lampe_Flash*, es wird später als Unterprogramm verwendet
- Erstelle ein neues leeres Programm
- Erstelle 3 Wegpunkte mit beliebigen Positionen (WP1, WP2, WP3)
- An WP1 wird das Programm gestartet
- Warte hier bis der Schalter 1 betätigt wird
- Wenn der Schalter 1 betätigt wurde, rufe das Unterprogramm Lampe_Flash auf
- Danach fahre zu WP2 und warte dort 1 Sekunde
- Wenn Schalter 2 nach dieser Wartezeit True ist → Bewegung zurück zu WP1
- Wenn Schalter 2 nach dieser Wartezeit False ist → Bewegung zu WP3 und dann zurück zu WP1
- Wird während des Programms der Schalter 1 auf False gesetzt soll das Programm sofort stoppen

Lösungstipp: Das sofortige Stoppen beinhaltet ein „Bedingung kontinuierlich Prüfen“. Welche Befehle könnten sich hierfür eignen?
Musterlösung zum Download unter robospace.de/ur oder in der niedersächsischen Bildungscloud.

4.1.8.5 Variablen

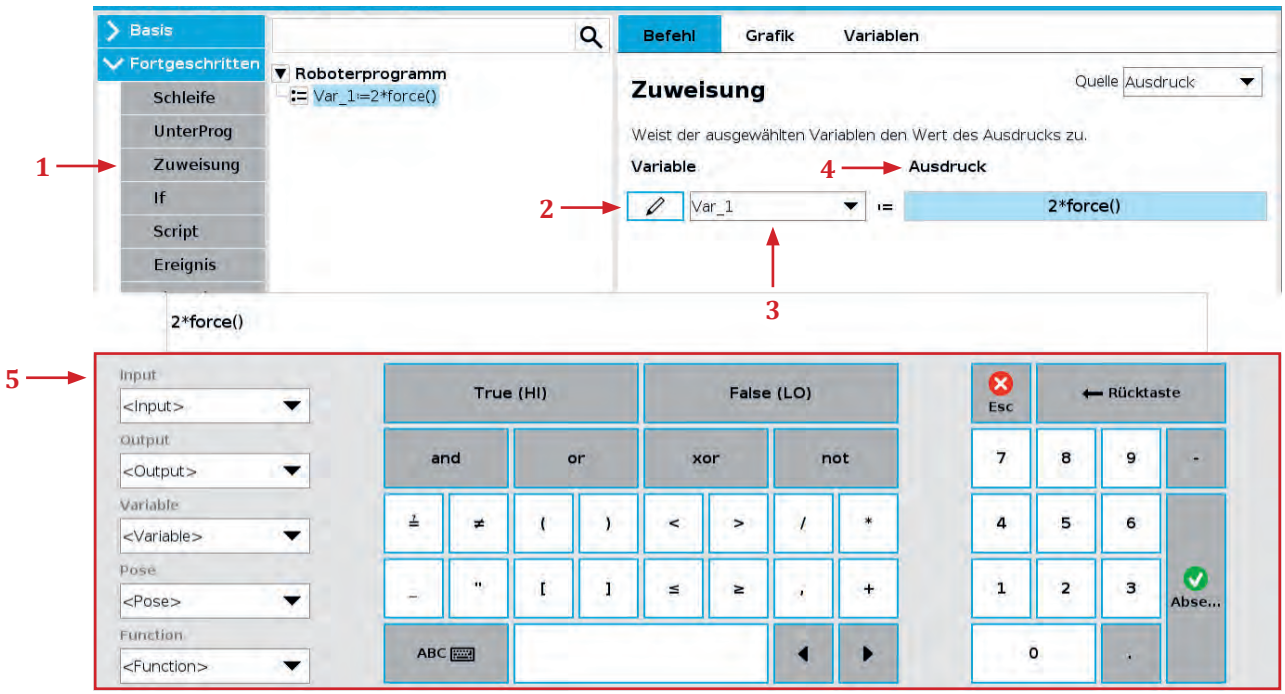
Arten von Variablen:

Typ	Wert
boolean	True/False
integer	Ganzzahl (16 Bit) z. B. 3
floating point	Realzahl z. B. 3,75
string	ASCII Zeichen (Text)
pose	Positions variable p[x,y,z,rx,ry,rz] (Struct)
list	Reihe von Variablen (Array)

Gültigkeitsbereich von Variablen:

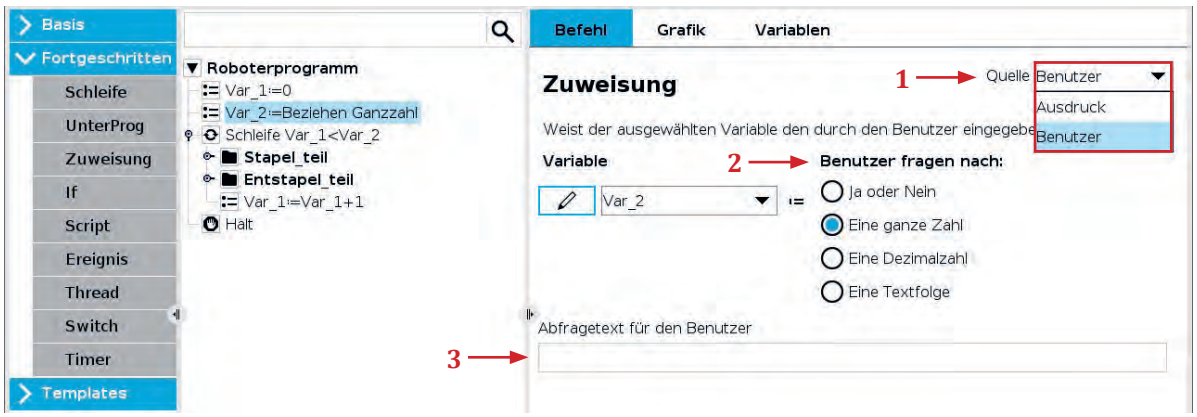
- Local-Variablen (lokale Variablen)
  - Werden im Programm deklariert
  - Zugänglich nur vom gleichen Programm
  - Wert wird beim Ausschalten gelöscht
- Global-Variablen (globale Variablen)
  - Deklariert in der Installationsdatei
  - Zugänglich von mehreren Programmen welche die gleiche Installationsdatei verwenden
  - Wert wird auf der Festplatte gespeichert (Kein Datenverlust beim Ausschalten)

4.1.8.6 Befehl: Zuweisung



Nr.	Modul	Beschreibung
1	**Zuweisung**	Befehl hinzufügen (Variablenwerte setzen oder Nutzereingabe abfragen)
2	**Zauberstab**	Variablennamen bearbeiten
3	**Variable**	Auswahl einer Variable
4	**Ausdruck**	Dient dazu, der Variable einen Wert bzw. eine Funktion zuzuweisen
5	**Ausdruckseditor**	Der Ausdruckseditor ist bei der Eingabe in jedem Funktionsfeld verfügbar. Er erleichtert durch zusätzliche Eingabemöglichkeiten die Definition eines Ausdrucks. Neben Zahlen finden sich hier auch Eingänge, Ausgänge, Variablen, Posen, eine Auswahl an UR-Script Funktionen und logische Operatoren.

Für eine Benutzereingabe muss die Quelle auf Benutzer gesetzt werden.



Nr.	Modul	Beschreibung
1	**Quelle**	Quelle „Benutzer“ wählen
2	**Benutzer fragen nach**	Dient der Bestimmung der Variablenart, die abgefragt wird.
3	**Abfragetext für den Benutzer**	Bietet die Möglichkeit, eine Nachricht für den Nutzer hinzuzufügen.

4.1.8.7 Ausprobieren: Variablen verwenden

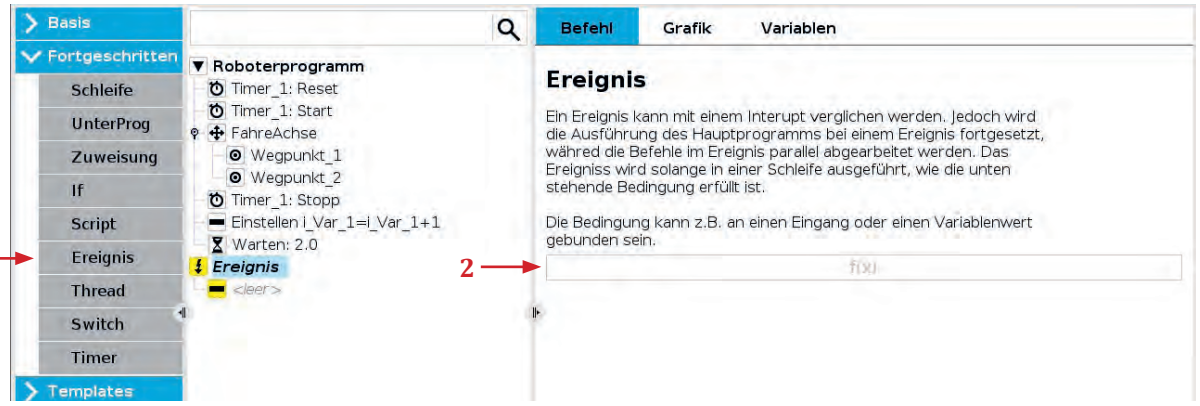
Benötigtes Material: 1x Schalter/Taster
Aufgabe: Erstelle ein Programm welches Installationsvariablen verwendet um Programmdurchläufe zu zählen

- Erstelle eine Installationsvariable, nenne sie „count“ und weise ihr den Wert „0“ zu
- Erstelle ein einfaches Programm welches sich zwischen Wegpunkt_1 und Wegpunkt_2 bewegt
- Erhöhe die Zählervariable nach jeder Bewegung.
- Wenn die Zählervariable 10 erreicht, reinige Werkzeug (bewege zu Wegpunkt_3)
- Wenn die Zählervariable 20 erreicht, zeige Meldung „Wechsle Tray”
- Setze das Programm nur fort, wenn die Zählervariable über einen Eingang (bspw. Schalter) zurückgesetzt wurde
- Überprüfe den Variablenwert von „count“ über den Variablen Tab während der Programmlaufzeit
- Stoppe das Programm an einem beliebigen Zeitpunkt, merke dir den aktuellen Wert und starte den Roboter neu, Überprüfe den Variablenwert

Musterlösung zum Download unter robospace.de/ur oder in der niedersächsischen Bildungscloud.

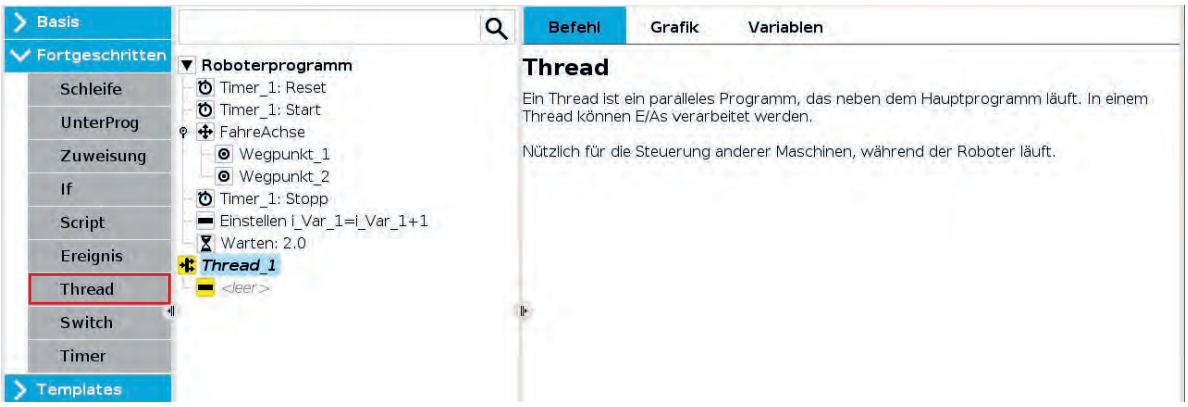
4.1.8.8 Befehl: Ereignis

Ein Ereignis definiert einen Programmteil, der nur ausgeführt wird, wenn die im Ereignis definierte Bedingung erfüllt wird. Die Besonderheit eines Ereignisses ist, dass es zeitgleich also parallel zum Hauptprogramm in einer Endlosschleife ausgeführt wird, solange die Bedingung erfüllt ist.

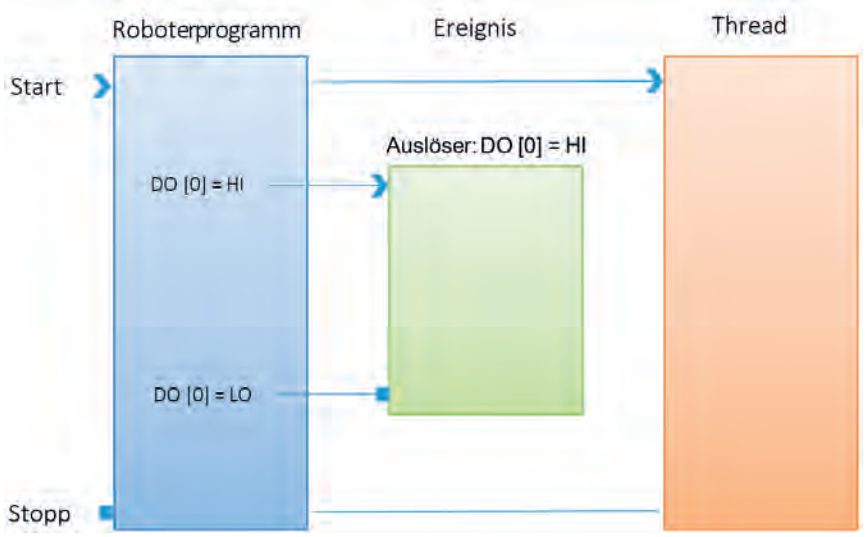


Nr.	Modul	Beschreibung
1	**Ereignis**	Befehl hinzufügen
2	**Eingabefeld**	Bedingung, die das Ereignis auslöst und ggf. aktiv hält

4.1.8.9 Befehl: Thread

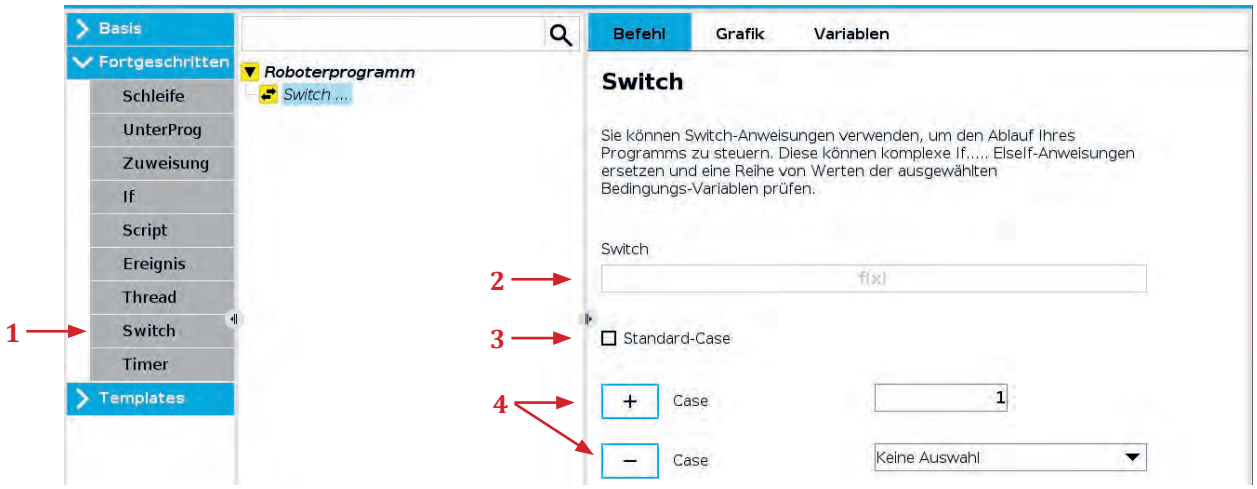


Ein Thread ist ein paralleler Prozess zum Roboterprogramm. Er kann zur Steuerung einer externen Maschine, unabhängig vom Roboterarm, eingesetzt werden (z.B. ein Förderband mit Lichtschranken). Mithilfe von Variablen und Ausgangssignalen kann ein Thread mit dem Roboterprogramm kommunizieren. Der Thread startet immer mit dem Hauptroboterprogramm und wird in einer Endlosschleife ausgeführt. Stoppt das Hauptprogramm wird auch die Ausführung des Thread beendet. Die folgenden Grafik stellt grafisch den Zusammenhang zwischen Roboterprogramm, Ereignis und Thread dar.



4.1.8.10 Befehl: Switch

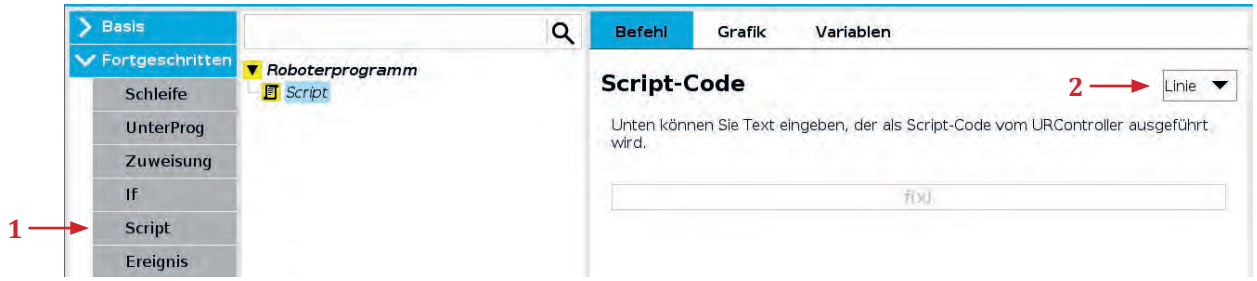
"Switch" ist eine Kontrollstruktur in der Programmierung, die im Vergleich zu "If"-Anweisungen eine oft übersichtlichere Alternative bietet, zwischen verschiedenen Fällen oder Bedingungen zu wechseln und den entsprechenden Programmteil auszuführen, basierend auf dem Wert einer bestimmten Variable oder eines Ausdrucks. Während "If"-Anweisungen mehrere Bedingungen nacheinander prüfen, erlaubt "Switch" eine klarere Struktur, insbesondere wenn es viele mögliche Bedingungen gibt. Jedes "Switch" kann mehrere Cases sowie einen Default Case haben.



Nr.	Modul	Beschreibung
1	**Switch**	Befehl hinzufügen
2	**Eingabefeld**	Hier wird der Ausdruckseditor verwendet, um die Bedingung zu definieren
3	**Standard-Case**	Dient der Definition eines Standard-Cases
4	**+ und -**	Dient zum Hinzufügen und löschen von Cases

4.1.8.11 Befehl: Script

Der Befehl Script ermöglicht den Zugriff auf die Echtzeit-Scriptsprache, die vom Roboter-Controller ausgeführt wird und richtet sich ausschließlich an erfahrene Benutzer. Anleitungen finden Sie im Skripthandbuch auf der Support-Webseite (http://www.universal-robots.com/support). Mithilfe der „File"-Option oben rechts können Benutzer Skript-Programmdateien importieren, wodurch umfangreiche und komplexe Skript-Programme in Kombination mit der benutzerfreundlichen Programmierung von PolyScope genutzt werden können. Mit der Option „Linie" kann eine Zeile Script über den Ausdruckseditor eingegeben werden. Grundsätzlich ist UR-Script an die Programmiersprache Python angelehnt.
Hinweis: Variablen, die im Script definiert wurden, werden nicht im Variablen-Fenster angezeigt. Wegepunkte, die in Script definiert sind, werden nicht im Grafikfenster angezeigt.

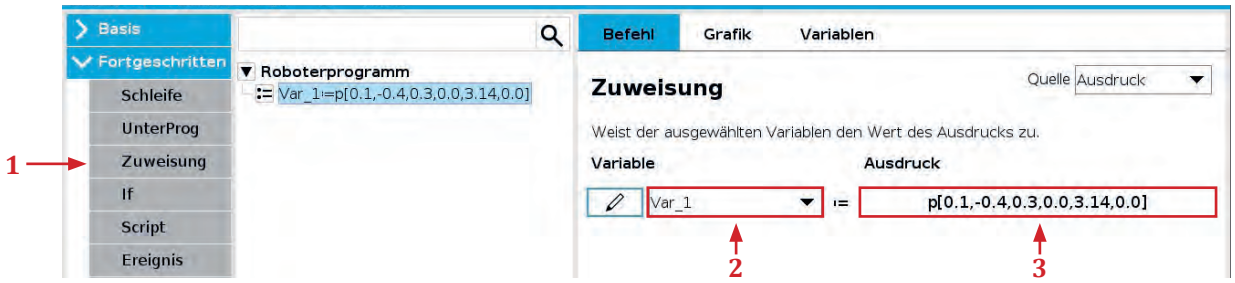


Nr.	Modul	Beschreibung
1	**Script**	Befehl hinzufügen
2	**Option-Auswahl**	Wechsel zwischen „Linie" und „File"

4.1.8.12 Pose Variable

Pose-Variablen sind spezielle Variablen, die Informationen über die Position und Orientierung eines Wegpunkts im Raum speichern. Sie bestehen aus sechs Werten (x, y, z, rx, ry, rz), die jeweils die Koordinaten im kartesischen Raum (x, y, z) und die Rotationswinkel um die Achsen (rx, ry, rz) repräsentieren. Für die Koordinaten im kartesischen Raum wird die Einheit Meter verwendet. Die Rotationswinkel werden in Radiant angegeben. Folgende Syntax wird für Pose Variablen verwendet.

Durch die Verwendung von eckigen Klammern ist der Zugriff auf einzelne Elemente einer Pose Variable möglich. So kann über *pose_var[0]* auf den Wert es x-Elements und über *pose_var[5]* auf den Wert des rz-Elements zugegriffen werden. Wichtig: Es wird bei 0 angefangen zu zählen.
Um eine Pose Variable zu definieren wird der Zuweisung-Befehl (1) benötigt. Nachdem die Variable mit einem Namen versehen wurde (2), wir bei Ausdruck (3) die oben gezeigte Syntax eingetragen. Wichtig: Richtige Einheit für die Werte verwenden.



Die definierte Pose Variable kann nun als Wegpunkt verwendet werden. Hierfür muss zunächst eine Bewegung/Wegpunkt (1) hinzugefügt oder ausgewählt werden. Anschließend muss im oberen rechten Menu „Variable Position“ (2) ausgewählt werden. Zuletzt muss die Variable (3), die die Pose enthält ausgewählt werden. Hinweis: Wenn der erste Wegepunkt im Programm variabel oder relativ ist, wird der Autostart unterdrückt.

Den Inhalt (aktuelle Position) einer Pose Variable zu ermitteln oder zu verarbeiten kann eine Herausforderung darstellen. Hierfür können jedoch Script-Funktionen zur Hilfe eingesetzt werden. Mit *get_actual_tcp_pose()* kann bspw. die aktuelle TCP-Position ausgelesen und direkt in eine Variable gespeichert werden.
In folgender Tabelle ist eine kleine Auswahl an Funktionen dargestellt. Weitere Funktionen finden sich in der UR-Script Dokumentation.

UR-Script Funktion	**Beschreibung**
get_actual_tcp_pose()	Gibt die aktuelle TCP-Position zurück
get_actual_tcp_speed()	Gibt die aktuelle Geschwindigkeit des TCP als Vektor zurück
get_inverse_kin()	Inverse Kinematik – Berechnet aus einer Pose die Gelenkkoordinaten
get_target_tcp_pose()	Gibt die aktuelle Zielposition des TCP zurück
get_target_tcp_speed()	Gibt die aktuelle Zielgeschwindigkeit des TCP als Vektor zurück
interpolate_pose(p_from, p_to, alpha)	Lineare Interpolation zwischen zwei Punkten
pose_add(p_1, p_2)	Addiert zwei Pose-Variablen
pose_dist(p_from, p_to)	Gibt den Abstand zwischen zwei Positionen in Metern zurück
pose_inv(p_from)	Gibt die Inverse einer Pose –Variable zurück
pose_sub(p_to, p_from)	Subtrahiert zwei Pose-Variablen voneinander
pose_trans(p_from, p_to)	Transformiert eine Pose in ein anderes Koordinatensystem

4.1.8.13 Ausprobieren: Speichern und Verändern einer Pose Variable

Aufgabe:
Speichere die aktuelle Position in die Variable „aktuelle_pos“ und erhöhe die Z-Höhe um 400 mm. Speichere das Ergebnis in „neue_pos“.

Lösung:
Zuweisung: aktuelle_pos = get_actual_tcp_pose()
Zuweisung: neue_pos = p[aktuelle_pos[0], aktuelle_pos[1], aktuelle_pos[2]+0.4, aktuelle_pos[3], aktuelle_pos[4], aktuelle_pos[5]]

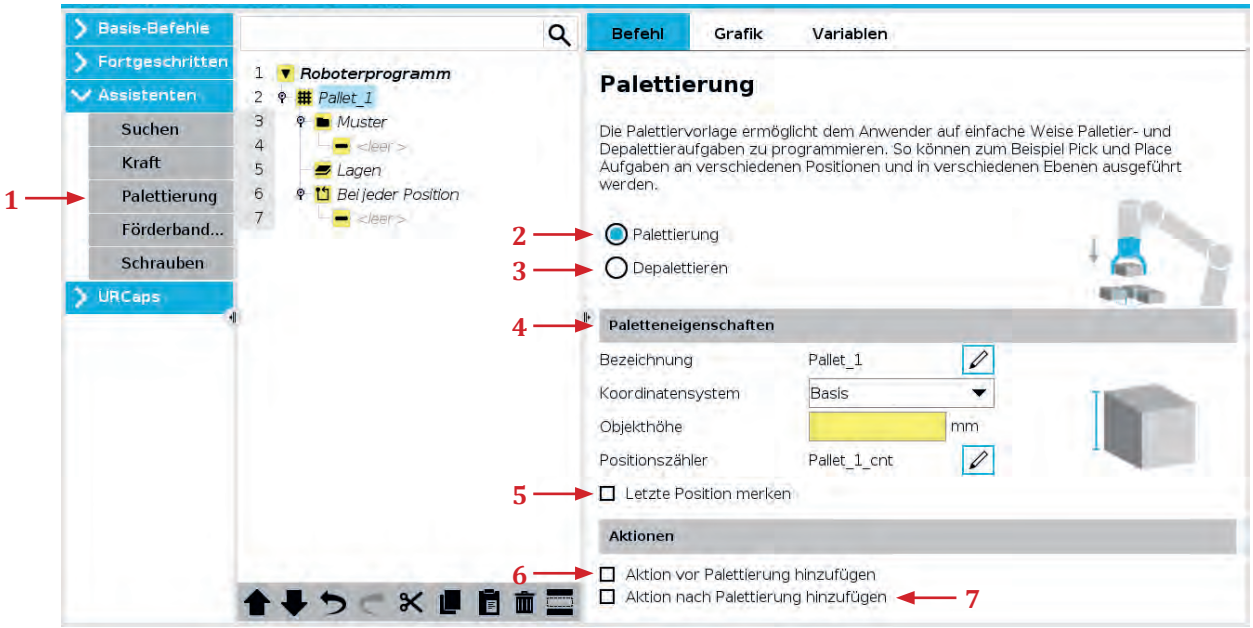
Erklärung:
Zunächst wird die aktuelle Position über *get_actual_tcp_pose()* ausgelesen und über eine Zuweisung in die Variable aktuelle_pos gespeichert. Anschließend wird wieder über einen Zuweisungs-Befehl die Variable neue_pos gespeichert. Hierbei muss die korrekte Syntax einer Pose Variable beachtet werden p[x,y,z,rx,ry,rz]. Über eckige Klammern werden die Werte der entsprechenden Elemente aus aktuelle_pos in neue_pos übernommen. An der dritten Stelle der neuen Pose Variable wird das Element mit 0.4 addiert um einen positiven Versatz von 400 mm mit einzubringen.

Hinweis:
Dezimalzahlen müssen mit einem Punkt geschrieben werden.

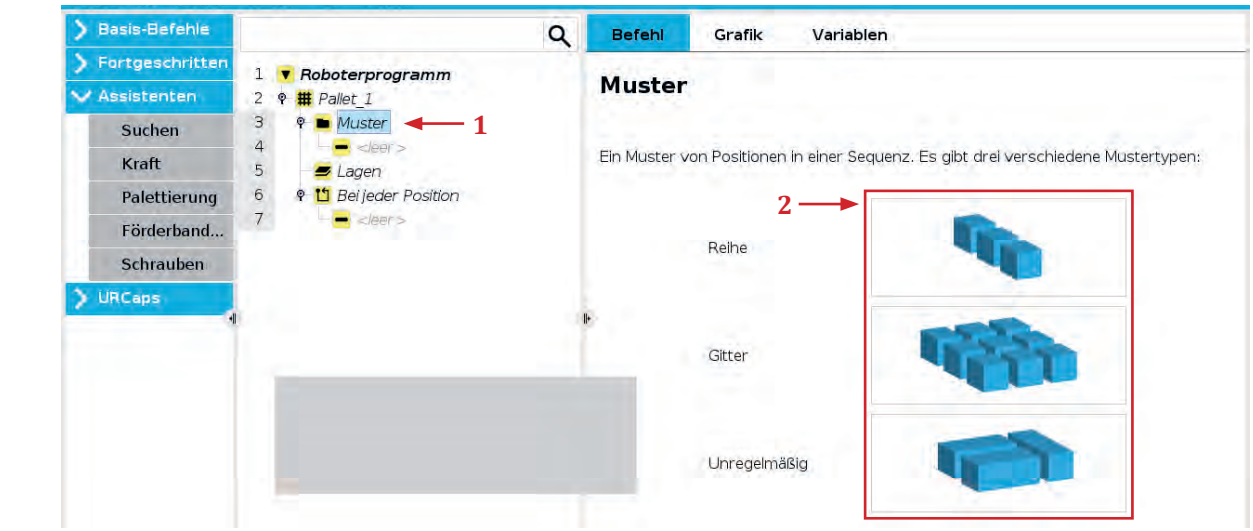
4.1.9 Assistenten

4.1.9.1 Palettierung

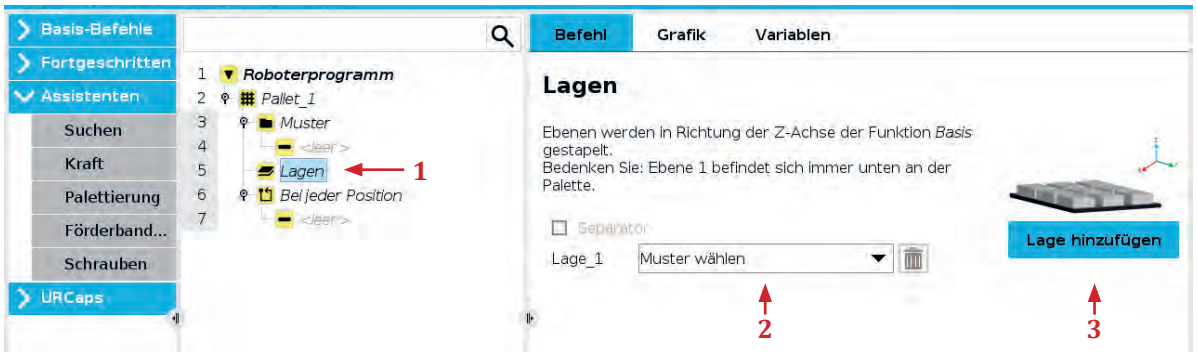
Der Palettierungsassistent unterstützt bei der Palettierung und Depalettierung. Es ist möglich mehrere Lagen und unterschiedliche Muster anzuwenden.



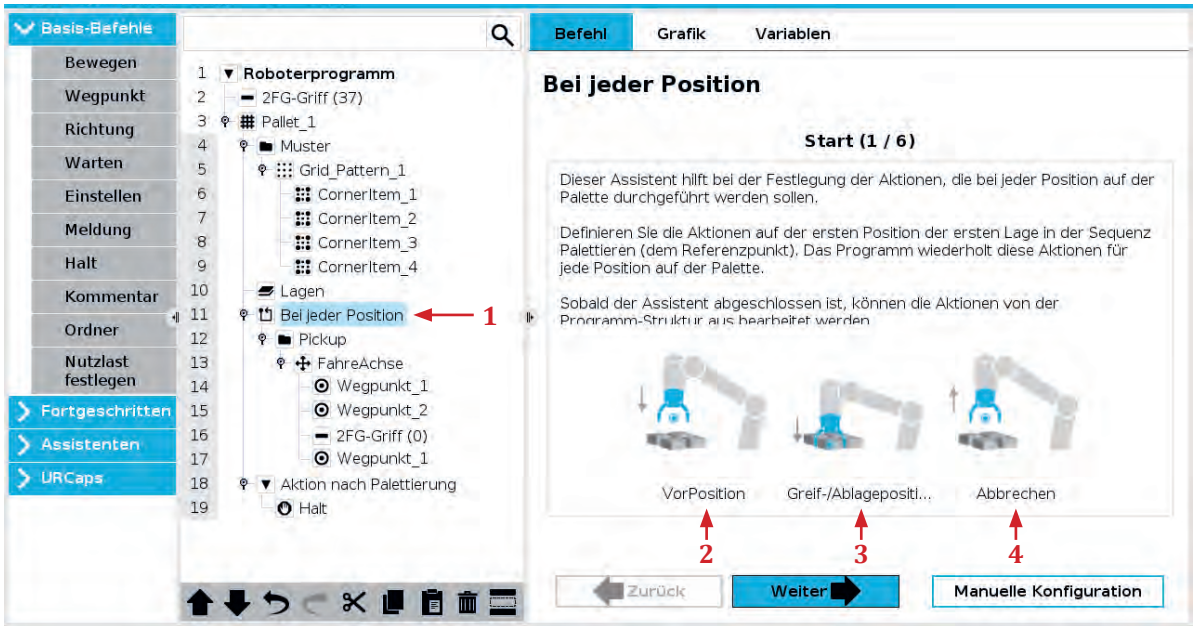
Nr.	Modul	Beschreibung
1	**Palettierung**	Befehl hinzufügen
2	**Palettierung**	Elemente auf einer Palette hinzufügen
3	**Depalettierung**	Elemente von einer Palette entfernen
4	**Paletteneigenschaften**	Setzen der Bezeichnung des verwendeten Koordinatensystems, der Objekthöhe und des Names des Positionszählers.
5	**Letzte Position merken**	Fortfahren bei der zuletzt gespeicherten Position
6	**Aktion vor Palettierung hinzufügen**	Programmknoten, der vor dem Start der Palettierung ausgeführt wird, hinzufügen.
7	**Aktion nach Palettierung hinzufügen**	Programmknoten, der nach dem Abschluss der Palettierung ausgeführt wird, hinzufügen. Hier kann es hilfreich sein einen Halt Befehl einzufügen damit der Stapelvorgang nach Erfolg beendet wird.



Nr.	Modul	Beschreibung
1	**Muster**	Antippen, um das Muster festzulegen
2	**Reihe/Gitter/ Unregelmäßig**	Muster auswählen, um dem Roboter lagenspezifische Positionen (z. B. Start- und Endpunkte, Gitterecken und/oder Anzahl der Elemente) anzulernen.



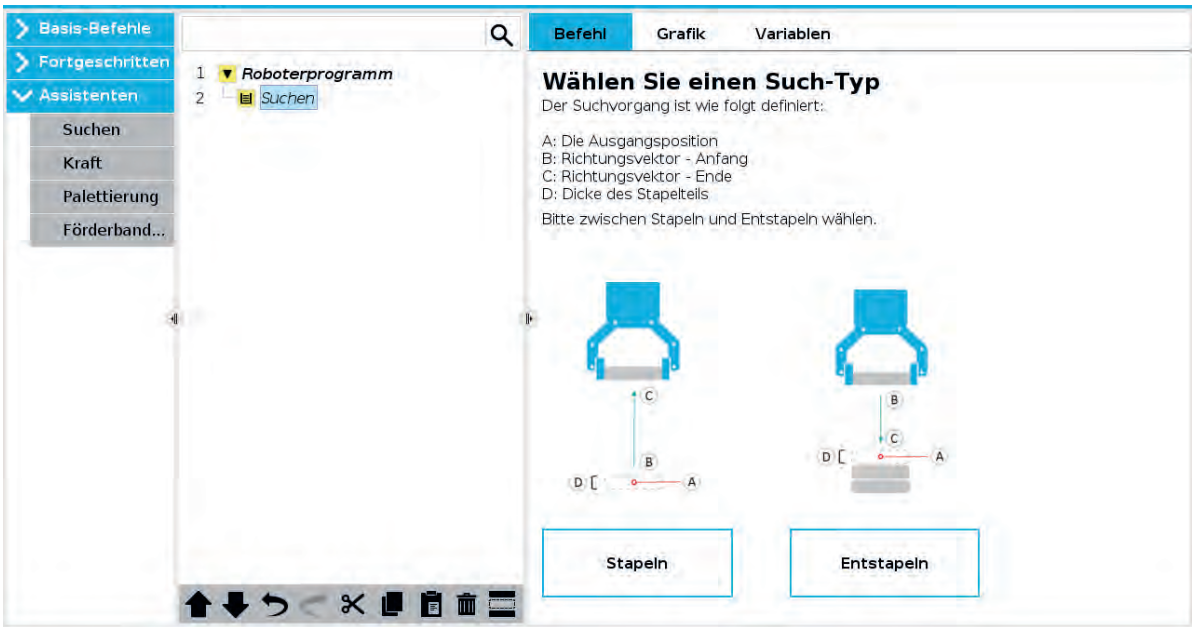
Nr.	Modul	Beschreibung
1	**Lagen**	Konfiguration der Palettierungsschichten
2	**Muster wählen**	Zuordnung eines Palettierungsmustern zu einer Schicht, Unterschiedliche Muster pro Schicht möglich.
3	**Lage hinzufügen**	Die gewünschte Anzahl an Palettierungsschichten hinzufügen



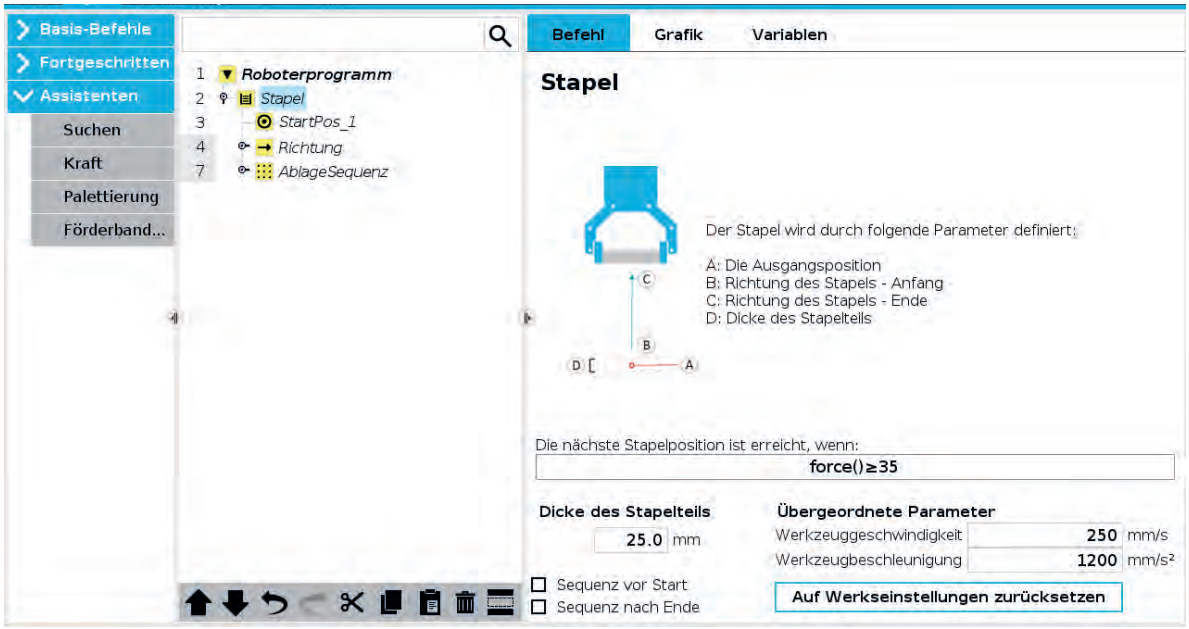
Nr.	Modul	Beschreibung
1	**Bei jeder Position**	Definition der Bewegungsroutine, die bei jeder Position ausgeführt wird. Bspw. das Greifen und kollisionsfreie Einfügen in die aktuelle Palettierungsposition. Ein mehrschrittiger Assistent hilft dabei die Positionen zu erzeugen.
2	**Vorposition**	Wegpunkt, der vor dem Eintauchen in die Greif-/Ablageposition angefahren wird. Dient der Kollisionsvermeidung.
3	**Greif-/Ablageposition**	Wegpunkt, der die Position definiert in der das Objekt aufgenommen bzw. abgelegt wird.
4	**Abbrechen**	Wegpunkt, der wie auch die Vorposition zur Kollisionsvermeidung dient und nach dem Greifen bzw. Ablegen angefahren wird.

4.1.9.2 Stapeln

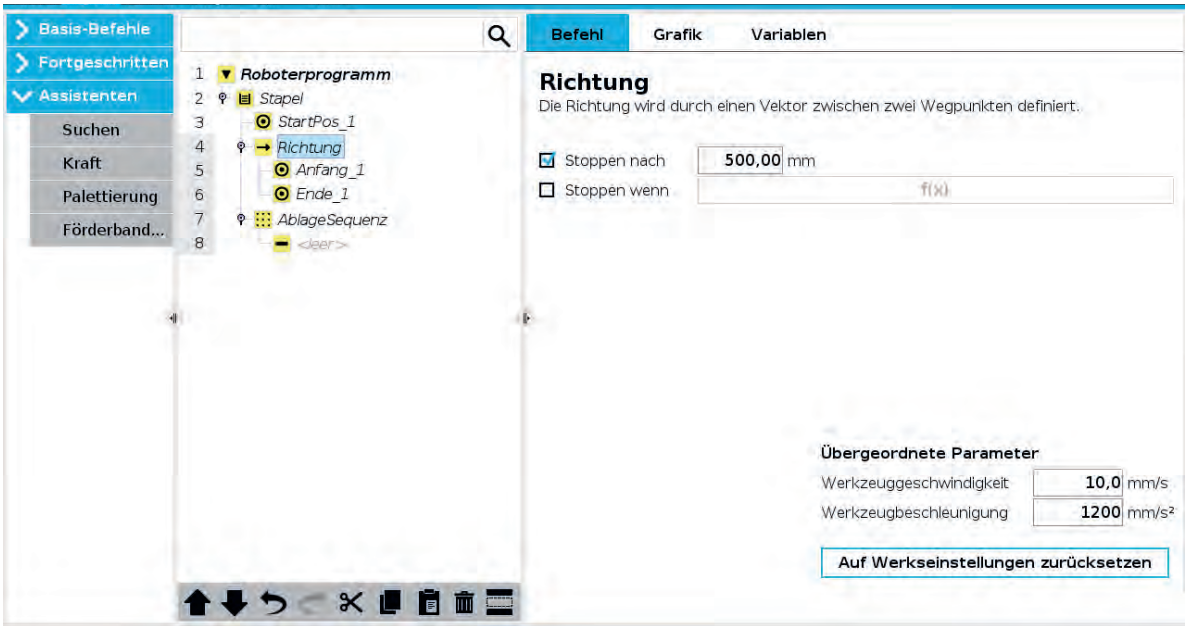
Der Stapel bzw. Entstapelassistent befindet sich im Reiter Suchen. Der Assistent vereinfacht die Aufgabe des Stapelns bzw. Entstapelns.



Es ist möglich Objekte allein anhand ihrer Dicke zu Stapeln oder die Ablageposition mithilfe einer definierten Kraft zu ermitteln. Sollte ein Stapeln nach Kraft gewünscht sein, ist es empfehlenswert die Bauteildicke etwas größer als gemessen anzugeben, damit die Kraftsuchfahrt erfolgreich durchgeführt werden kann. Dazu muss im Feld Die nächste Stapelposition ist erreicht, wenn: bspw. *force()≥35* eingegeben werden. Diese Bedingung verwendet den UR-Script Befehl *force()*, der die am Tool Flange anliegende Kraft in N ausgibt. Sofer diese im gegebenen Beispiel 35 N erreicht oder überschreitet wird die Bedingung wahr und der Kraftsuchvorgang für die Stapelposition ist abgeschlossen.
Hinweis: Es kann sein, dass je nach Roboter eine größere Kraftschwelle verwendet werden muss. Bemerkbar macht sich dieser Umstand dadurch, dass die Kraftsuchfahrt zu früh abgebrochen oder erst gar nicht durchgeführt wird. Tipp: Die Kraft in einem Thread in eine Variable speichern und über den Variablen-Tab auslesen, um ein Gefühl für die Werte zu erlangen.

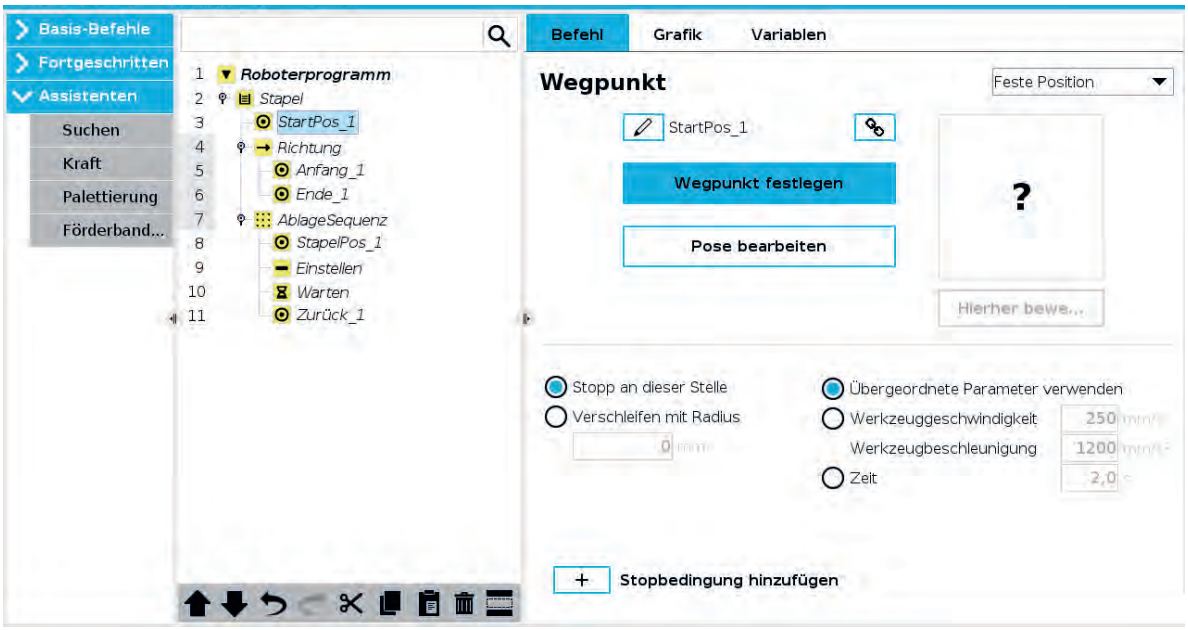


Um den Stapelvorgang vollständig zu konfigurieren muss im Knoten Richtung eine maximale Stapelhöhe oder Abbruchbedingung festgelegt werden.

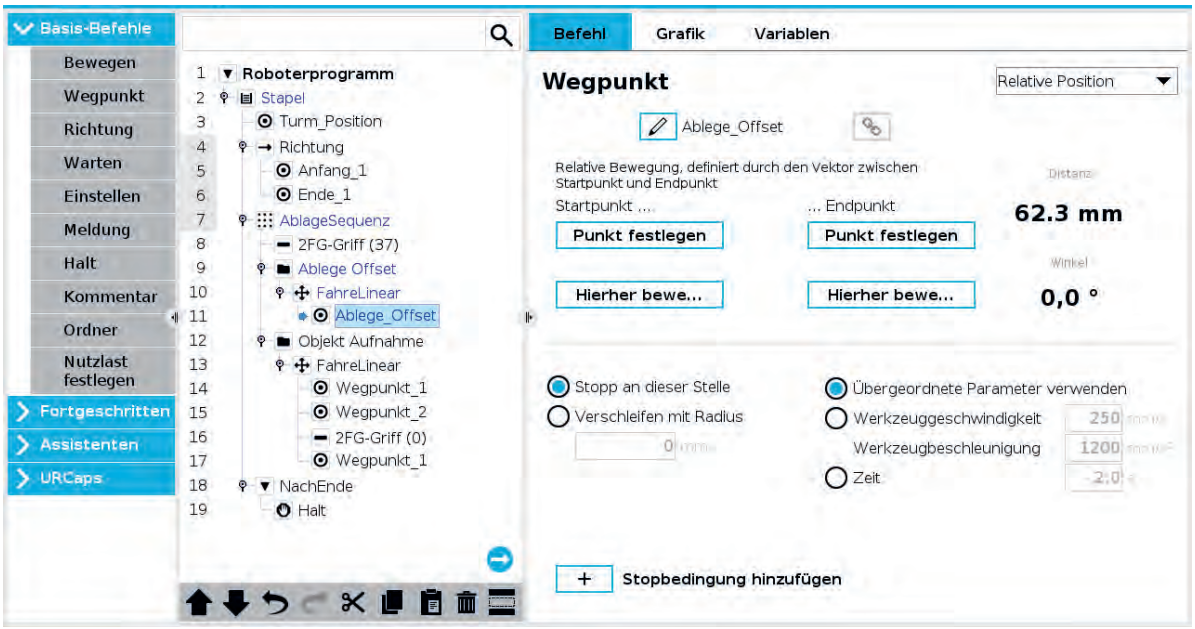


4.1.9.2.1 Beispielprogramm Stapeln

Da die Benutzung des Assistenten in der Vergangenheit zu Problemen geführt hat, findet sich im Folgenden ein Beispielprogramm, dass alle nötigen Teilschritte zum Stapeln beinhaltet.
Folgende Grafik zeigt die durch den Assistenten erzeugte leere Vorlage. Für unser Beispiel wird StartPos_1 in Turm_Position umbenannt, da dies die unterste Steinposition für den zu stapelnden Turm definiert.
Unter Richtung müssen für Anfang_1 und Ende_1 zwei Wegpunkte definiert werden. Die absolute Position der beiden Punkte hat keine Relevanz, da lediglich die Differenz der beiden Wegpunkte die Stapelrichtung definiert (ähnlich einer relativen Position).



Im Knoten *AblageSequenz* werden alle 4 Elemente gelöscht. Anschließend werden im Knoten *AblageSequenz* zwei Ordner erstellt, die *Ablege Offset* und *Objekt Aufnahme* heißen.



Der Inhalt des Knotens *AblageSequenz* beinhaltet die folgenden Bestandteile.

Modul	Beschreibung
2GF-Griff	Öffnen des Greifers (Gedanklich wird hier nach dem erfolgreichen Ablegen eines Bauteils gestartet.)
Ablege Offset	Beinhaltet eine relative Bewegung, die den Endeffektor nach dem Öffnen des Greifers in eine sichere Position (nach oben) verfährt, um Kollisionen mit dem Stapel zu vermeiden.
Objekt Aufnahme	Beinhaltet Wegpunkte zum Anfahren des individuell definierten Aufnahmepunkts und einen Befehl, um den Greifer zu schließen.
Nach Ende	Hält das Programm nach Erreichen der Zielstapelhöhe an.

Hinweis: Sollte der Knoten Ablagesequenz weiterhin gelb bleiben, wechseln Sie im Befehl Tab der *AblageSequenz* zu einem beliebigen anderen Wegpunkt als Referenzposition.

4.1.9.3 Kraft

Der Kraftassistent beim Universal Robot ermöglicht kraftbasierte Steuerung und passt Roboterbewegungen an externe Kräfte an. Zum Beispiel kann der Roboter beim Polieren von Oberflächen gleichmäßigen Druck auf gekrümmte Oberflächen ausüben oder in Montageprozessen empfindliche Teile präzise zusammenfügen. Beim Kraftassistenten kann zwischen den Konfigurationsoptionen Einfach, Rahmen, Punkt und Bewegung ausgewählt werden. Über die gewählte Option kann bestimmt werden in welche Achse oder Rotation der Roboter nachgiebig wird. Der Roboterarm kann Kraft in drei Achsrichtungen und Drehmoment um diese drei Achsen messen und definiert aufbringen. Als Startpunkt für den Kraftassistenten ist Modul 11 der UR Academy (https://academy.universal-robots.com/de/kostenloses-e-learning/e-learning-fur-die-e-series/e-series-pro-track/) ein sehr hilfreicher Anlaufpunkt. Durch das interaktive Training mit passenden Animationen wird der Assistent bestmöglich erklärt.

4.1.10 Konfigurierbare Sicherheit

Bei kollaborierenden Robotern ist es wichtig, eine Risikobewertung durch den Integrator durchzuführen, sobald der Roboter in einer Applikation eingebunden wurde. Die konfigurierbaren Sicherheitseinstellungen helfen bei dieser Risikobewertung.

4.1.10.1 Sicherheitspasswort

Die Sicherheitseinstellungen von UR sind passwortgeschützt, damit es nicht jedem Anwender möglich ist die Einstellungen zu ändern. Die Konfigurierbarkeit ermöglicht es, dass die Sicherheit für jede Applikation individuell angepasst werden kann. So kann sichergestellt werden, dass Mitarbeiter und Peripheriegeräte sich nicht verletzen bzw. beschädigt werden. Um die Sicherheitsparameter konfigurieren zu können muss zunächst unter: Installation → Sicherheit am unteren Bildschirmrand das gültige Passwort eingegeben werden. Auf Werkseinstellungen hat der Roboter kein Passwort gesetzt, dieses muss vor der ersten Verwendung der Sicherheitskonfigurationen durch den Nutzer gesetzt werden. Die Änderung des Passworts ist über Hamburger Menü → Einstellungen → Sicherheit möglich.

4.1.10.2 Sicherheitsprüfsumme

Die Prüfsumme ist zu jeder Zeit rechts oben neben dem Hamburger Menü zu finden. Befindet sich der Roboter in Werkseinstellungen ist die Prüfsumme CCCC. In der Prüfsumme sind die Sicherheitseinstellungen des Roboters kodiert. Damit kann der Anwender auf einfache Weise überprüfen, ob sich der Roboter die erwartete Sicherheitskonfiguration angewandt hat.

4.1.10.3 Roboter-Limits

Im Tab Roboter-Limits der Sicherheitseinstellungen können die sicherheitsrelevanten Roboterbegrenzungen gesetzt werden. Dies kann über **Werksvoreinstellungen** in Form von Vorgaben geschehen oder manuell sofern **Detaillierte Einstellungen** ausgewählt wird. Bei den Voreinstellungen kann zwischen stark eingeschränkt und schwach eingeschränkt in 4 Stufen gewählt werden. Die aktuellen Grenzen werden hierbei in der Tabelle auf der Einstellungsseite angezeigt. Wichtig: Die Roboter-Limits sind Obergrenzen, die der Roboter in keinem Fall überschreitet, jedoch nicht unbedingt erreicht oder erreichen kann. Grund für das nicht erreichen können die Grenzen in den Bewegungsbefehlen oder die Roboterhardware sein, die bspw. für eine maximal zulässige Leistung von 1000 W nicht spezifiziert ist. Es sind Einstellungen für Normal und Reduziert vorhanden auf die später eingegangen wird. Um die geänderten Einstellungen zu speichern, muss unten **„Übernehmen“** gedrückt werden. Damit werden die Einstellungen bestätigt.

4.1.10.4 Betriebsmodi

Der Roboter besitzt zwei verschiedene Betriebsmodi, in denen er sich befinden kann. Ohne zusätzliche Konfiguration befindet sich der Roboter im **normalen Modus**. Durch Sicherheitseinstellungen kann der Roboter auch in den **reduzierten Modus** versetzt werden. Dabei verfährt der Roboter mit den in der Sicherheitseinstellung definierten eingeschränkten Limits. Der reduzierte Modus kann durch Verwendung eines Sicherheitseingangs oder Sicherheitsebene ausgelöst werden. Beide Varianten werden in folgenden Kapiteln erklärt. Tipp: Ob sich der Roboter im reduzierten Modus oder normalem Modus befindet, lässt sich zu jeder Zeit unten links neben der runden Statusanzeige erkennen.

4.1.10.5 Gelenkgrenzen

Im Tab Gelenkgrenzen der Sicherheitseinstellungen können im Bereich **Positionsbereich** die minimalen und maximalen Gelenkwinkel für jedes einzelne Gelenk eingestellt werden. Dies kann bspw. zur Kollisionsvermeidung verwendet werden. Im Bereich **maximale Geschwindigkeit** können die maximal zulässigen Geschwindigkeiten für jedes Gelenk festgelegt werden. Für beide Bereiche können Grenzen für den normaler Modus und den reduzierten Modus festgelegt werden.

4.1.10.6 Sicherheitsebenen

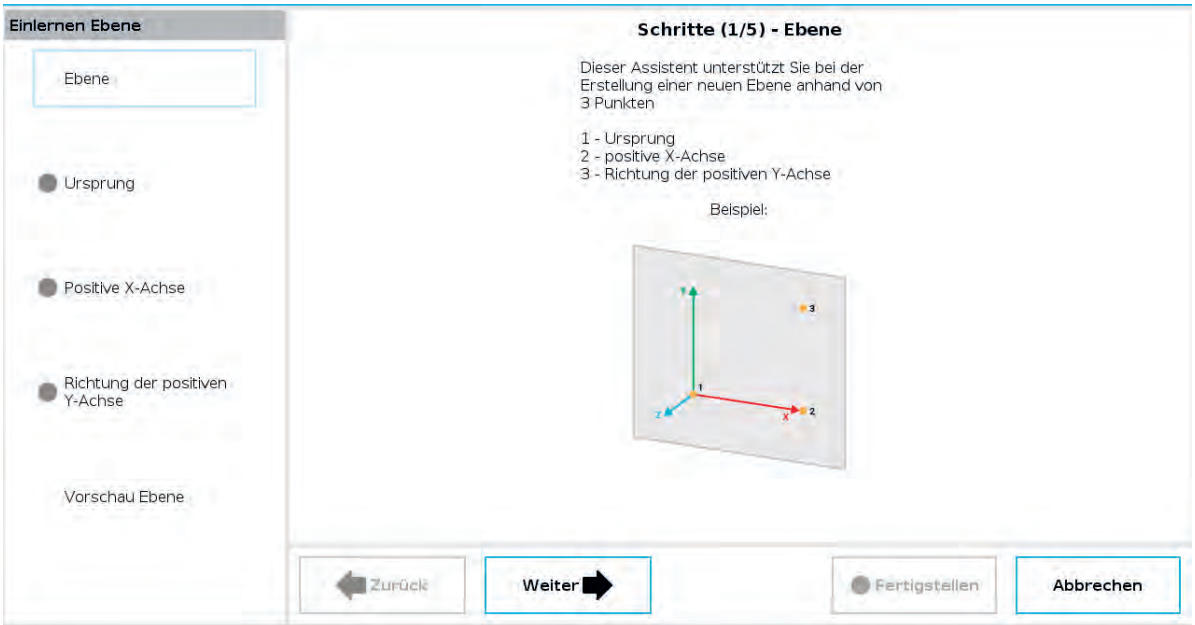
Sicherheitsebenen ermöglichen die Definition von Sicherheitszonen im Arbeitsraum des Roboters, die bei Betreten oder Verlassen unterschiedliche Sicherheitsmaßnahmen, wie bspw. den reduzierten Modus, auslösen. Folgende Sicherheitsmaßnahmen können beim Durchstoßen einer Ebene konfiguriert werden:

Symbol	Sicherheitsmodus	Verhalten
	Deaktiviert	Inaktiv
	Normal	Agiert als „strenge Begrenzung“ im normalen Modus
	Reduziert	Agiert als „strenge Begrenzung“ nur im reduzierten Modus
	Normal & Reduziert	Agiert als „strenge Begrenzung“ in beiden Modi
	Auslöser reduzierter Modus	Roboter schaltet auf reduzierten Modus um, wenn Tool Flange in die Ebene eindringt

Eine strenge Begrenzung führt dazu, dass der Roboter im jeweiligen Betriebsmodus den definierten Sicherheitsbereich nicht mehr betreten darf. Wird der Roboter in den Sicherheitsbereich bewegt stellt dies eine Verletzung der Sicherheitskonfiguration dar und die Bewegung des Roboters wird gemeinsam mit einer Fehlermeldung gestoppt. Der Roboter kann nun im **Wiederherstellungsmodus** aus dem Sicherheitsbereich geführt werden.
Tipp: Sollte sich der Roboter aufgrund der Verletzung einer Sicherheitseinstellung nicht starten lassen, dann muss die Sicherheitseinstellung vor dem Start des Roboters entfernt werden. Alternativ kann probiert werden den Roboter mithilfe des **Backdrive** in einen zulässigen Bereich zu bewegen. Der Backdrive lässt sich durch Drücken des Freedrive Knopfes aktivieren, wenn sich der Roboter im gelben Leerlauf Zustand befindet. Hierbei fällt das Bewegen deutlich schwerer, da der Roboter ohne Motorunterstützung bewegt wird.
Neben einer strengen Begrenzung lässt sich auch das Auslösen des reduzierten Modus definieren.

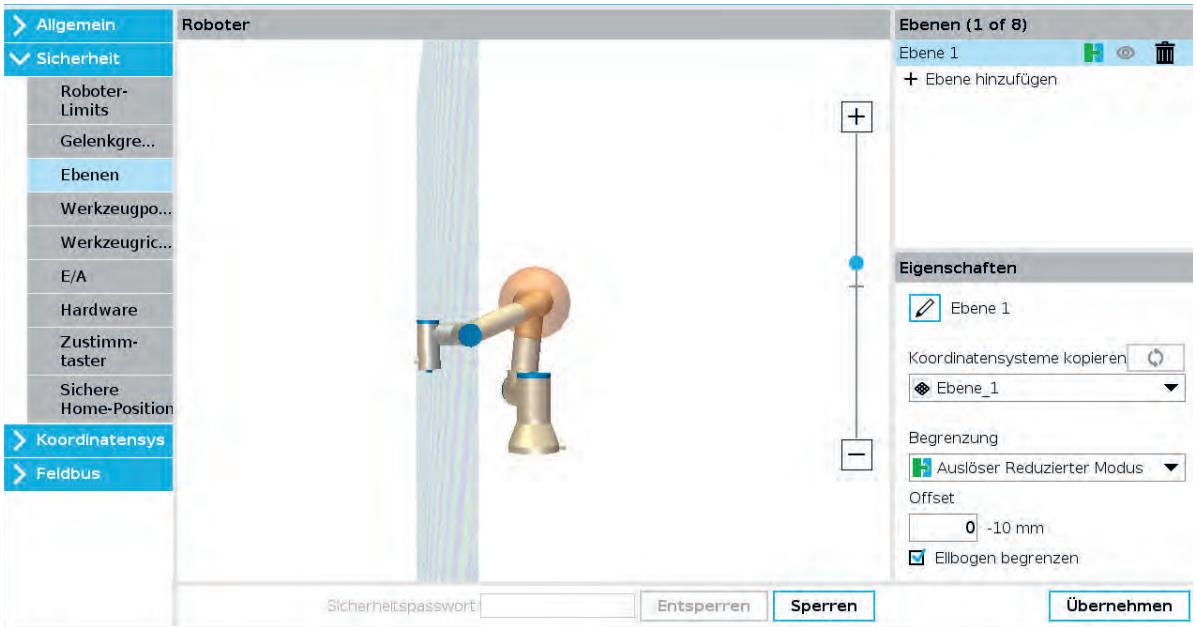
4.1.10.6.1 Definition einer Sicherheitsebene

Um eine Sicherheitsebene konfigurieren zu können, muss im ersten Schritt eine neue Ebene unter Installation → Koordinatensysteme → Ebene erstellt werden. Durch Tippen auf Ebene wird direkt eine neues gelbes Ebenenelement erzeugt. Über das Stift Symbol kann dieser Ebene ein Name gegeben werden. Anschließend muss über den Knopf **„Diese Ebene anlernen“** die Ebene mithilfe eines Konfigurationsassistenen im Raum definiert werden. Die Ebene wird durch die drei orangenen nummerierten Punkte aufgespannt, die nacheinander angefahren werden.



4.1.10.6.2 Konfiguration einer Sicherheitsebene

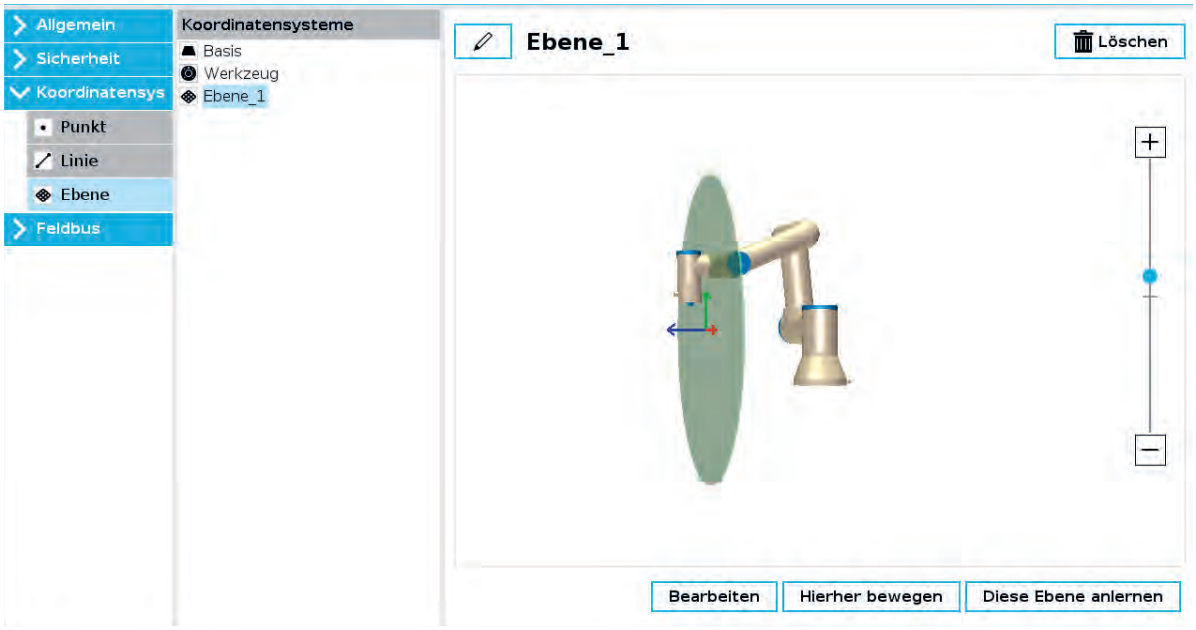
Ist die Erstellung der Ebene abgeschlossen kann diese für die Konfiguration einer Sicherheitsebene verwendet werden. Dazu muss der Tab Ebenen unter Installation → Sicherheit → Ebenen aufgerufen werden. Nach dem Eingeben des Sicherheitspassworts und dem Entsperren des Roboters kann im oberen rechten Bereich durch Tippen auf das Plus Symbol eine neue Ebene hinzugefügt werden.



Durch den Stift kann der Sicherheitsebene ein Name gegeben werden. Im Dropdown Menu Koordinatensysteme kopieren muss die gewünschte zuvor definierte Ebene ausgewählt werden und der dazu gewünschte Begrenzungsmodus. Über Offset lässt sich die Ebene entlang der Z-Achse in beide Richtungen verschieben. Die Auswahloption Ellenbogen begrenzen wird visuell durch eine Kugel um den Ellenbogen des Roboters dargestellt und führt dazu, dass nicht nur der Tool Flange des Roboters, sondern auch der Ellenbogen einen Sicherheitsmodus auslösen kann.

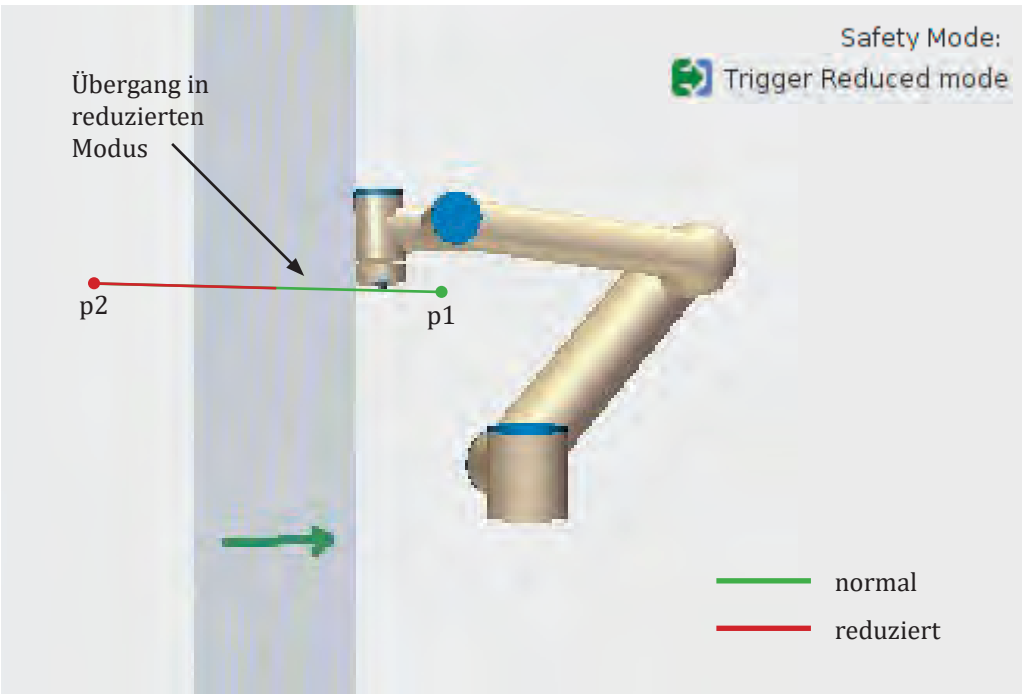
4.1.10.6.3 Wirkrichtung einer Sicherheitsebene

Die Orientierung der Ebene ist entscheidend für den Effekt der Sicherheitsmodi. Die positive Z-Richtung bestimmt die Wirkrichtung der Ebene.



Im Beispiel aus dem Screenshot ist der reduzierte Modus als Sicherheitsmodus für die Ebene definiert. Dieser wird nach Durchstoßen der Ebene in positiver Z-Richtung ausgelöst, wie sich unten links anhand der Statusanzeige erkennen lässt. Wichtig: Ist die positive Z-Richtung der Ebene zur falschen Seite orientiert lässt sich über Bearbeiten die Ebene um 180 Grad drehen. Hierbei ist wichtig, dass nach dem Tippen auf Bearbeiten das Koordinatensystem der Ebene ausgewählt wird und für den Rotationsvektor die Einheit Grad. Danach lässt sich die Ebene durch eine Drehung um 180 Grad um RX oder RY drehen. Es darf im Anschluss nicht vergessen werden die Ebene in den Sicherheitseinstellungen mit der gedrehten Ebene zu aktualisieren. Hierfür muss unter Sicherheit → Ebenen das Aktualisierungssymbol mit dem kleinen gelben Warndreieck gedrückt werden. Anschließend verschwindet das Warndreieck wieder und zeigt an, dass die Kopie der Ebene aktuell ist.

4.1.10.7 Ausprobieren: Reduzierten Modus mit einer Sicherheitsebene auslösen



1. Erstelle eine Ebene im Tab Koordinatensysteme > Ebene
2. Achte auf die Orientierung der Z-Achse
3. Konfiguriere mit dieser Ebene eine Sicherheitsebene im Tab Sicherheit > Ebenen, die bei Durchstoßen den Reduzierten Modus auslöst
4. Führe den Roboter im Freedrive durch die Ebene und überprüfe mit der Statusanzeigen unten Links in welchem Modus sich der Roboter befindet
5. Wechsel den Sicherheitsmodus der Sicherheitsebene und prüfe die Funktion
6. Drehe ggf. die Wirkrichtung der Ebene, dabei das Aktualisieren nicht vergessen

4.1.10.8 Werkzeugposition

Der Kontaktpunkt mit einer Sicherheitsebene und Auslösepunkt der Sicherheitsmodi ist der Tool Flange oder je nach Konfiguration auch der Ellenbogen. Das angebrachte Werkzeug des Roboters wird ohne zusätzliche Konfiguration im Tab Werkzeugposition nicht berücksichtigt. Um das Werkzeug schematisch abzubilden können bis zu drei virtuelle Kugeln mit unterschiedlichen Durchmessern und Ursprüngen definiert werden.

4.1.10.9 Werkzeugrichtung

Die Neigung des Werkzeugs kann zwischen 5 und 181 Grad im Bezug zu einem Koordinatensystem eingeschränkt werden. Die aktuelle Einstellung wird visuell durch einen Kegel dargestellt. Die Einstellung kann bspw. verwendet werden, um zu verhindern, dass ein Laser in Augen gerichtet werden kann.

4.1.10.10 Sicherheits E/A

Sicherheitsfunktion:
Sicherheitsfunktionen können den konfigurierten E/A's zugewiesen werden.
Alle Funktionen sind redundant. Dies bedeutet, dass es zwei Signale für jede Funktion gibt.
Konfigurierbare E/A's sind:
- Digitale Eingänge
- Digitale Ausgänge

Sicherheits-Eingangssignale:
Bei den Sicherheitseingängen gibt es die vier folgenden Auswahlmöglichkeiten:
- **Not-Stopp**: Für externe Verbindung eines Not-Aus-Schalters oder einer Sicherheits-SPS
- **Reduzierter Modus**:
  - LOW: Roboter arbeitet im normalen Modus
  - HIGH: Roboter arbeitet im reduzierten Modus
- **Schutz-Reset**: Wenn hier ein Kontakt hardwaremäßig angeschlossen ist, so kann mit dieser Einstellung die Sicherheit des Roboters resettet werden.
- **3-Stufenschalter**: Ermöglicht zwischen drei Betriebsarten zu wechseln: 1) vollautomatischer Betrieb, 2) manuelle Steuerung mit reduzierter Geschwindigkeit und 3) Not-Aus-Funktion zur sofortigen Anlagenabschaltung.

Sicherheits-Ausgangssignale:
Bei den Sicherheitsausgängen gibt es die folgenden fünf Auswahlmöglichkeiten:
- **System-Notabschaltung**
  - HIGH: Normaler Modus
  - LOW: Not-Stopp
- **Roboter bewegt sich**
  - HIGH: Roboter bewegt sich nicht
  - LOW: Roboter bewegt sich
- **Roboter stoppt nicht**
  - HIGH: Roboter soll stoppen, Signal ist HIGH bis der Roboter gestoppt ist
  - LOW: keine Stopp-Aufforderung
- **Reduzierter Modus**
  - HIGH: Normaler Modus
  - LOW: Reduzierter Modus
- **Nicht Reduzierter Modus**
  - Antivalent zum reduzierten Modus

Weiter Schulungsunterlagen und Musterlösungen zum Download unter robospace.de/ur oder in der niedersächsischen Bildungscloud.

4.2 Mitsubishi Melfa Assita

Dirk Rokossa, Hochschule Osnabrück

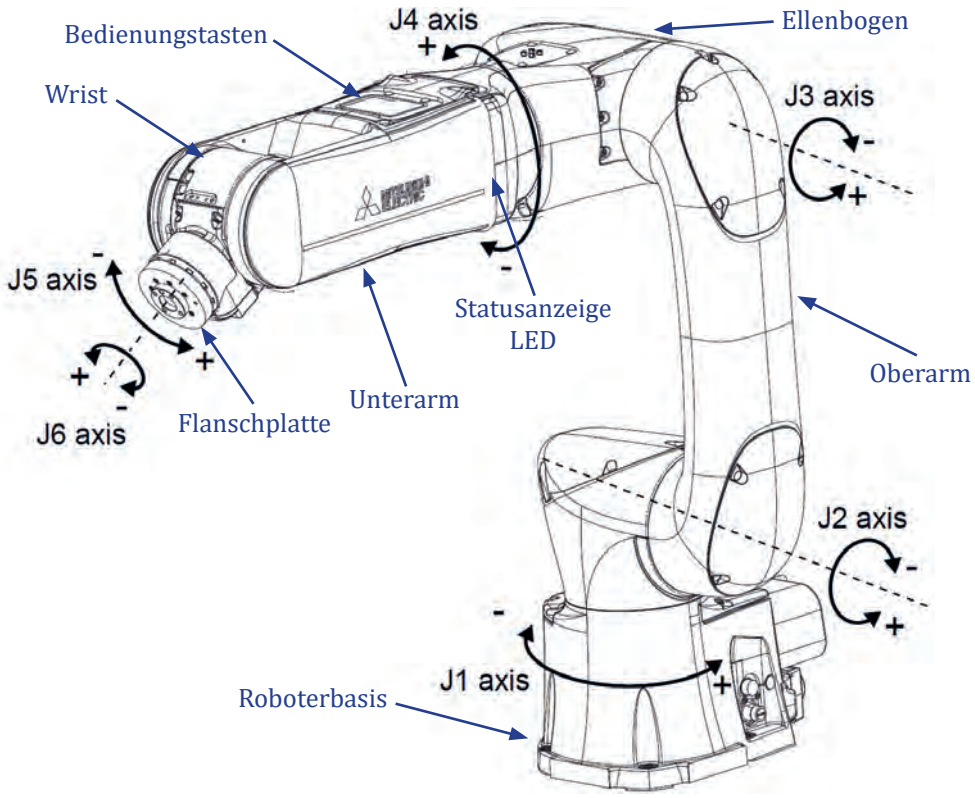
Inhalt

4.2.1 Einleitung

Dieses Dokument soll auf einfachem Weg die grundsätzlichen Elemente zur Bedienung und Programmierung des Mitsubishi-Roboters vom Typ RV-5AS-D nahebringen. Häufig wird der Roboter auch mit dem Namen Assista bezeichnet. In dieser Kurzanleitung werden beide Bezeichnungen synonym verwendet. Bevor Sie starten können, ist es unumgänglich sich mit den Sicherheitseinrichtungen im Umfeld des Roboterarbeitsplatzes und deren Bedienung zu beschäftigen. Hierzu existieren ggf. zusätzliche Dokumente, die das Starten und Betreiben der Sicherheitseinrichtungen erklären und die Besonderheiten erläutern.

4.2.2 Der Mitsubishi RV-5AS-D

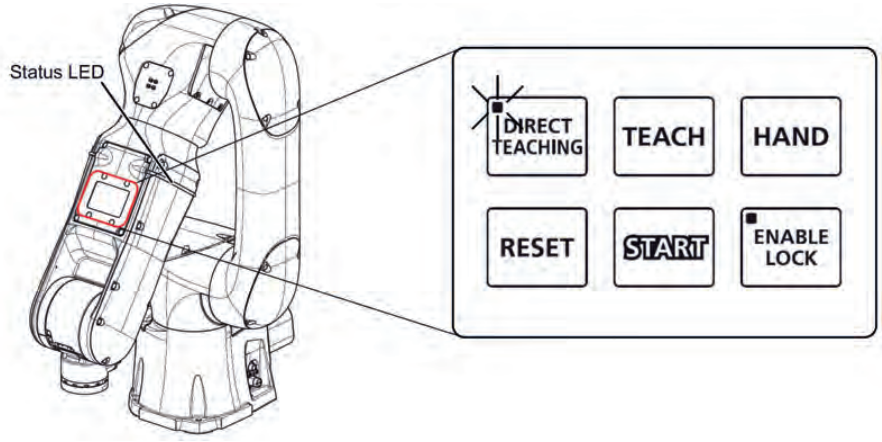
Der Mitsubishi-Roboter RV-5AS-D ist ein 6-Achsen Knickarmroboter, der in industriellen Anwendungen universell einsetzbar ist. Als kollaborierender Roboter (→ Cobot) darf er innerhalb einer Applikation auch ohne klassische Schutzeinhausungen betrieben werden. In diesem Fall sind nach einer entsprechenden Risikobeurteilung mögliche Kontaktszenarien zwischen Roboter bzw. Roboterwerkzeug und Bedienperson zu prüfen und geeignete Schutzmaßnahmen gemäß den geltenden Regeln zur Auslegung kollaborierender Roboterarbeitszellen durchzuführen und zu dokumentieren.



Das Robotersystem besteht im praktischen Einsatz in der Regel aus dem eigentlichen Roboterarm, der Robotersteuerung, dem Handbediengerät, der Hand/Automatik-Umschaltbox und der Stromversorgungsbox mit dem Hauptschalter. Zur Programmierung am PC wird ein Windows-Betriebssystem benötigt, auf dem die Programme *RT VisualBox* und *RT Toolbox3* installiert sind.

4.2.2.1 Bedienungstasten am Roboterarm

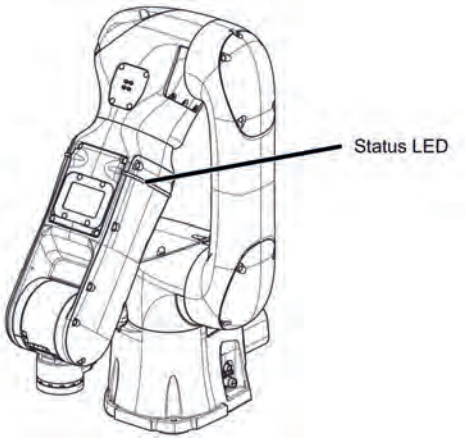
Die Bedientasten befinden sich auf der Oberseite des Roboter-Unterarms.



Taste	Beschreibung
	Halten Sie die Taste mindestens 2 Sekunden lang gedrückt, um das direkte Teachen ein- bzw. auszuschalten. Drücken Sie einmal, um den Direkt-Teaching-Modus zu ändern. Die Betriebsmodi schalten in der folgenden Reihenfolge um: Gelenkmodus → Translationsmodus → Rotationsmodus → Gelenkmodus Das LED-Verhalten hat folgende Bedeutung: Aus: Direktes Teachen gestoppt An: Direktes Teachen ist eingeschaltet (Gelenkmodus) Schnelles Blinken: Direktes Teachen ist eingeschaltet (Translationsmodus) Langsames Blinken: Direktes Teachen ist eingeschaltet (Rotationsmodus) Siehe auch Abschnitt 4.1.1.
	Drücken Sie einmal, um die aktuelle Position einzulernen (teachen). Jedes Mal, wenn diese Taste gedrückt wird, wird eine Position geteached. Das Programm RT VisualBox ist erforderlich, um mit dieser Taste Posen zu teachen. Siehe auch Abschnitt 4.1.2.
	Halten Sie die Taste mindestens 2 Sekunden lang gedrückt, um das angeflanschte Werkzeug auszurichten. Informationen zur Ausrichtung des Werkzeugs finden Sie auch im Abschnitt 3.2.5. Drücken Sie einmal, um das Werkzeug zu öffnen/zu schließen.
	Wenn Fehler auftreten: Einmal drücken, um Fehler zurückzusetzen. Wenn Programme unterbrochen sind: Drücken Sie einmal, um das Programm zurückzusetzen.
	Wenn der Roboter im Kollaborationsmodus betrieben wird und die Zusammen¬arbeit beendet ist: Halten Sie die Taste mindestens 2 Sekunden lang gedrückt, um das Programm von Anfang an auszuführen. Wenn der Kollaborationsbetrieb unterbrochen wurde, halten Sie die Taste mindestens 2 Sekunden lang gedrückt, um das Programm ab dem aktuellen Befehl neu zu starten. Verwenden Sie das Programm RT VisualBox, um zu prüfen, ob der Kollaborationsbetrieb gestoppt oder unterbrochen wurde.
	Einmal drücken, um die Sperre zu aktivieren/deaktivieren. Die Übernahme der Betriebsrechte über diese Taste verhindert, dass andere Geräte den Roboter bedienen/bewegen. Das Verhalten der LED ist wie folgt. ...Ein: Betriebsrechte erworben ...Aus: Betriebsrechte aufgegeben Siehe hierzu auch Abschnitt 4.

4.2.2.2 Statusanzeige-LED

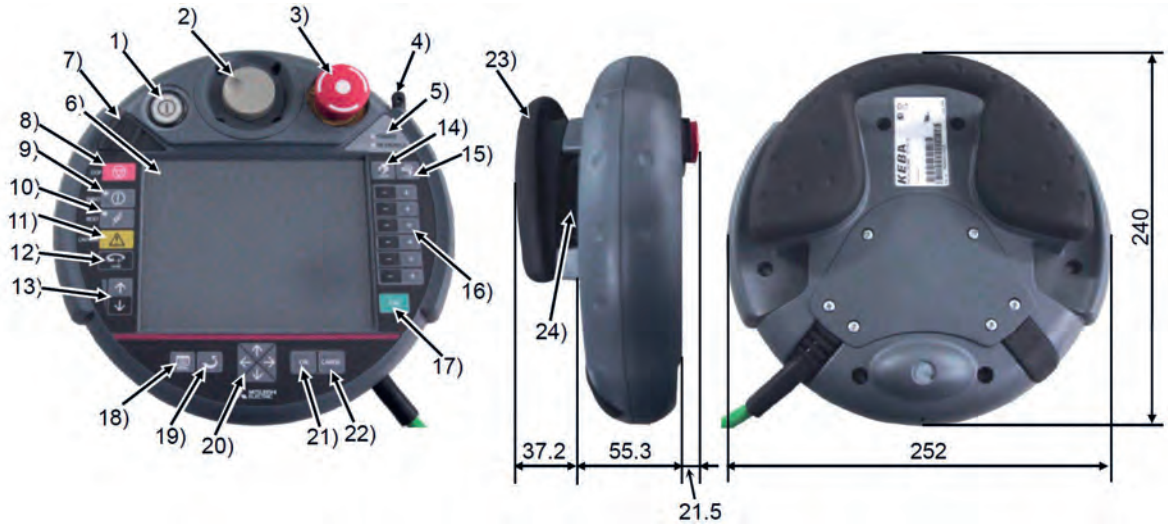
Die Statusanzeige-LED befindet sich auf dem Unterarm des Roboters. Über diese Anzeige kann direkt der aktuelle Status des Roboters erkannt werden.



Farbe	Bedeutung	
	leuchtet	blinkt
rot	Low-Level-Fehler	High-Level-Fehler
gelb	Warnung (Betrieb mit niedriger Geschwindigkeit/Standardbetrieb)	Warnung (Betrieb mit hoher Geschwindigkeit)
blau	Angehalten (Betrieb mit niedriger Geschwindigkeit/Standardbetrieb)	Angehalten (Hochgeschwindigkeitsbetrieb)
grün	Kooperative Betriebsart (Langsamlauf)	Kollaborative Betriebsart (Standardbetrieb)
weiß	---	Hochgeschwindigkeitsbetrieb
hellblau	Motoren (Servos) ausgeschaltet	Neustart der Steuerung

4.2.2.3 Handbediengerät R56TB

Das Handbediengerät dient – wie bei jedem ‚klassischen' Industrieroboter – zum Erstellen, Bearbeiten und Steuern eines Roboterprogramms, zum Einlernen der Bearbeitungsposen oder zum Ausführen des Tippbetriebs. Das Handbediengerät verfügt über eine grafische Touchscreen-Benutzeroberfläche (GUI), die eine einfache Bedienung ermöglicht. Darüber hinaus ist der 3-Positionen-Freigabeschalter (Totmann-Schalter) für den sicheren Gebrauch integriert.



1) TEACH Taste	Mit diesem Umschalter können Sie den Betrieb der Tasten auf dem Handbediengerät aktivieren oder deaktivieren. Die Lampe (weiß) leuchtet während der Aktivierung.
2) Drehrad	Bewegung des Cursors und Menü-Auswahl
3) Not-Aus-Schalter	Der Roboter wird im Notzustand angehalten. Die Motoren werden ausgeschaltet. Drehen Sie den Schalter nach rechts zum Quittieren.
4) Eingabestift (im Gehäuse integriert)	Der Stift, mit dem der Touchscreen bedient wird.
5) LED für die Stromversorgung, LED für die Freigabe des Handbediengerätes	Die POWER-LED leuchtet während der Versorgung mit Strom. Die TB ENABLE-LED leuchtet während der Freigabe und zeigt an, dass das Handbediengerät verwendet werden kann.
6) Touchscreen	Tippen Sie direkt auf den Bildschirm oder nutzen Sie den Eingabestift.
7) USB-Anschluss	Buchse für einen USB-Speicherstick.
8) STOP Taste	Hierdurch wird der Roboter sofort angehalten. Das Motoren schalten sich aber nicht aus.
9) SERVO Taste	Hierdurch wird die Stromversorgung der Motoren bei gleichzeitig betätigtem Freigabeschalter eingeschaltet. Die LED (grün) leuchtet während die Motoren den Status ON haben.
10) RESET Taste	Diese Taste setzt einen aufgetretenen Fehlerzustand zurück.
11) CAUTION Taste	Wenn diese Taste im Tippbetrieb gedrückt wird, kann der Endschalter aufgehoben werden. Drücken Sie außerdem diese Taste, wenn Sie die Bremse lösen wollen.
12) HOME Taste	Rückkehr zur zuvor eingestellten Home-Position.
13) OVRD Taste	Damit wird die Override-Bewegungsgeschwindigkeit verändert.
14) HAND Taste	Anzeige des Bildschirms für die Werkzeugeinstellungen.
15) JOG Taste	Anzeige des Bildschirms für den Tippbetrieb.
16) +/- Tasten	Diese Tasten funktionieren entsprechend der gewählten Funktion
17) EXE Taste	Bewegen des Roboters, z. B. zur Ausrichtung des Werkzeugs. Siehe Abschnitt 3.2.5.
18) MENU Taste	Anzeige des Menübildschirms.
19) RETURN Taste	Schließen der einzelnen Bildschirme.
20) Pfeiltasten	Hiermit kann der Cursor bewegt werden.
21) OK Taste	Bestätigung der jeweiligen Bildschirmoperation
22) CANCEL Taste	Abbrachen der jeweiligen Bildschirmoperation
23) Griff	Zum Festhalten des Handbediengerätes.
24) Freigabeschalter	Wenn dieser Schalter bei aktiviertem Handbediengerät losgelassen oder mit Kraft gedrückt wird, schalten sich die Motoren des Roboters direkt aus. Um den Roboter im Tippbetrieb o.ä. zu bewegen, drücken Sie den Schalter leicht und halten Sie ihn gedrückt. Freigabeschalter befindet sich an jedem der beiden Griffe.

4.2.2.4 Einschalten des Roboters

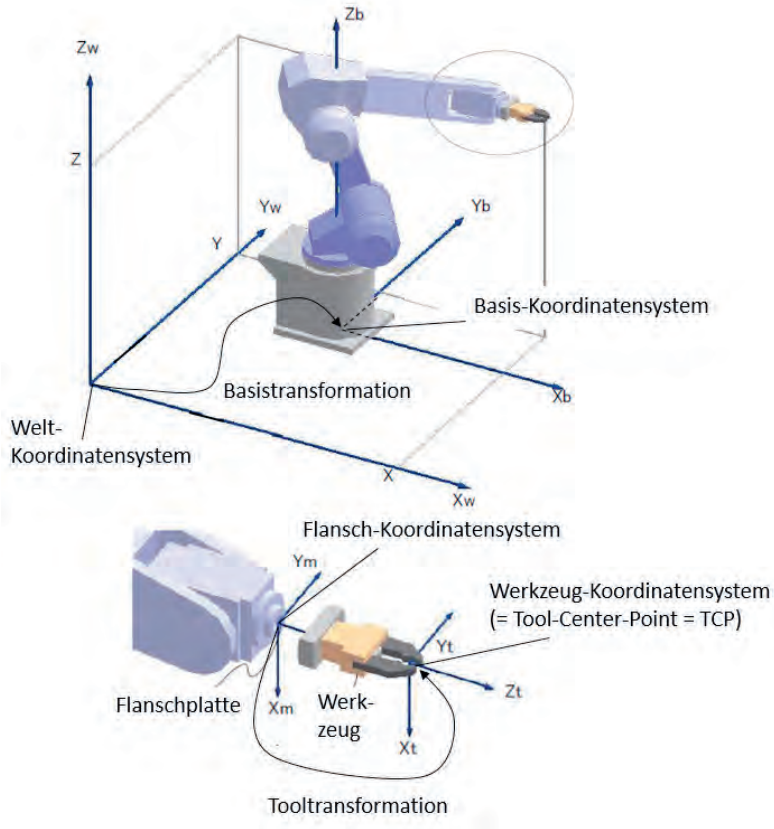
Vergewissern Sie sich zunächst, dass sich keine Hindernisse, wie z. B. Werkzeuge, im Arbeitsbereich des Roboters befinden, bevor Sie den Roboter einschalten. Drehen Sie dann den Hauptschalter an der Stromversorgungsbox auf die Stellung ON. Dadurch wird die Steuerung des Roboters gestartet, was einige Sekunden dauert. Nach erfolgreichem Start der Steuerung leuchtet die Statusanzeige-LED blau (siehe Abschnitt 2.2). Der Roboter ist nun betriebsbereit. Starten Sie auch den PC am Roboterarbeitsplatz. Abhängig von der Art, wie Sie den Roboter bedienen und programmieren wollen, werden zusätzlich auf dem PC installierte Programme (RT VisualBox und RT Toolbox3) benötigt.

4.2.2.5 Ausschalten des Roboters

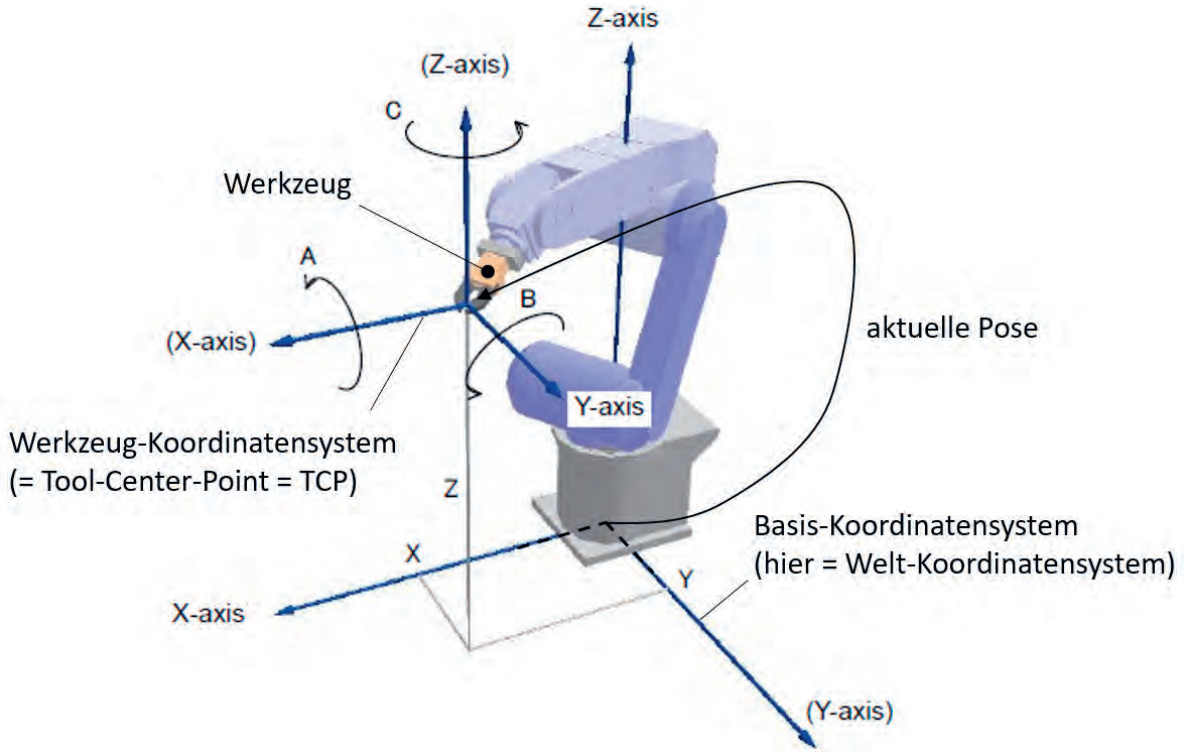
Beim Ausschalten des Roboters darf der Roboterarm nicht in Bewegung sein und sich kein Programm in der Ausführung befinden. Bringen Sie den Roboter vor dem Ausschalten in eine Stellung, so dass der Arbeitsplatz des Roboters frei zugänglich ist.
Der Roboter wird dann durch Drehen des Hauptschalters an der Stromversorgungsbox auf die Stellung OFF ausgeschaltet.

4.2.2.6 Koordinatensysteme und Posendefinition

Zur Bedienung und Programmierung des Roboters lassen sich vier Koordinatensysteme am Roboter und in seiner Arbeitsumgebung definieren:
Das **Welt-Koordinatensystem** ist der Standard für die Anzeige der aktuellen Pose (Position und Orientierung) des Roboters.
Das **Basis-Koordinatensystem** ist auf der Befestigungsebene des Roboters fest mit dem Robotersockel verbunden. Die X-Achse zeigt vorne aus dem Robotersockel heraus. Da die Basistransformation standardmäßig auf Null (= keine Transformation) gesetzt ist, stimmt das Weltkoordinatensystem mit dem Basiskoordinatensystem überein!
Das **Flansch-Koordinatensystem** liegt vorne am Roboter in der Mitte der Flanschplatte. Die Z-Achse zeigt dabei aus der Flanschplatte heraus.
Das **Werkzeug-Koordinatensystem** sollte abhängig vom jeweils montierten Endeffektor definiert werden (= Tooltransformation). Ohne eine Neudefinition des Werkzeug-Koordinatensystems ist es identisch mit dem Flansch-Koordinatensystem (= keine Tooltransformation).



Die aktuelle Stellung des Roboters (→ die Roboterpose) kann über die 6 Gelenkwinkelwerte (J1 bis J6) der 6 Roboterachsen definiert werden. Alternative (und i.d.R. häufiger genutzt) können kartesische Koordinaten relativ zu einem Bezugskoordinatensystem verwendet werden. Das Bezugskoordinatensystem für die Roboterpose ist das Welt-Koordinatensystem, was – wie schon auf der vorherigen Seite beschrieben – häufig das Basis-Koordinatensystem des Roboters ist.



Bei den 6 kartesischen Koordinaten einer Pose geben die X-, Y- und Z-Werte die Position des Ursprungs des Werkzeug-Koordinatensystems (→ Tool-Center-Point, TCP) an. Die Orientierung des Werkzeug-Koordinatensystems wird beim Mitsubishi-Roboter über die Winkel A, B und C definiert:
A → Drehwinkel (in Grad) um die X-Achse des Bezugskoordinatensystems
B → Drehwinkel (in Grad) um die Y-Achse des Bezugskoordinatensystems
C → Drehwinkel (in Grad) um die Z-Achse des Bezugskoordinatensystems

Die kartesische Definition einer Pose im Arbeitsraum des Roboters sieht beispielhaft also wie folgt aus:

Name	X	Y	Z	A	B	C
Bsp_Pose	300.000	0.000	500.000	180.000	45.000	90.000

Hierbei ist zu beachten, dass die Rotationen um die A-, B- und C-Winkel sequenziell betrachtet werden müssen. Die Drehungen finden dabei immer bezogen auf das feststehende Bezugskoordinatensystem (z. B. das Basis-Koordinatensystem) statt. Dies entspricht der Orientierungs-Notation gemäß Roll-Pitch-Yaw (und nicht Eulerwinkeln).
Zum Vergleich: Bei Kuka-Robotern werden auch die Winkel A, B und C verwendet. Dort sind es aber Eulerwinkel und die Drehwinkel beziehen sich auf das jeweils mitgedrehte Koordinatensystem.

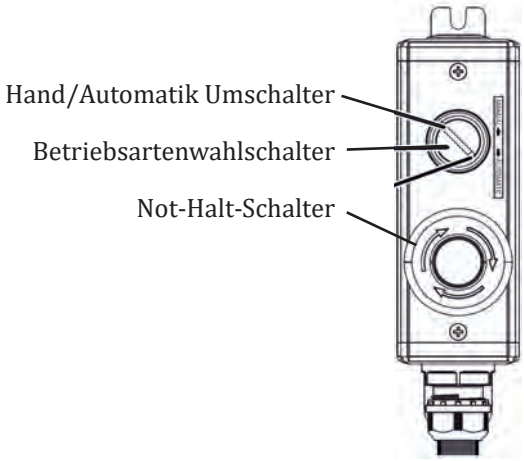
4.2.3 Bewegen des Roboters

Eine Bewegung des Roboters in seinen Achsen kann auf unterschiedliche Weisen erfolgen. Am effizientesten kann eine Bewegung direkt durch das Anfassen des Roboterarms oder über das Handbediengerät erfolgen. Für eine grobe Ausrichtung der einzelnen Robotergelenke innerhalb des Roboterarbeitsraums ist die direkte Bewegung des Roboterarms durchaus geeignet und intuitiv. Eine zielgenaue Ausrichtung z. B. eines montierten Werkzeugs in der Arbeitsumgebung ist so aber mitunter zu ungenau. Mit dem Handbediengerät gelingt dies über den Tippbetrieb i.d.R. genauer.

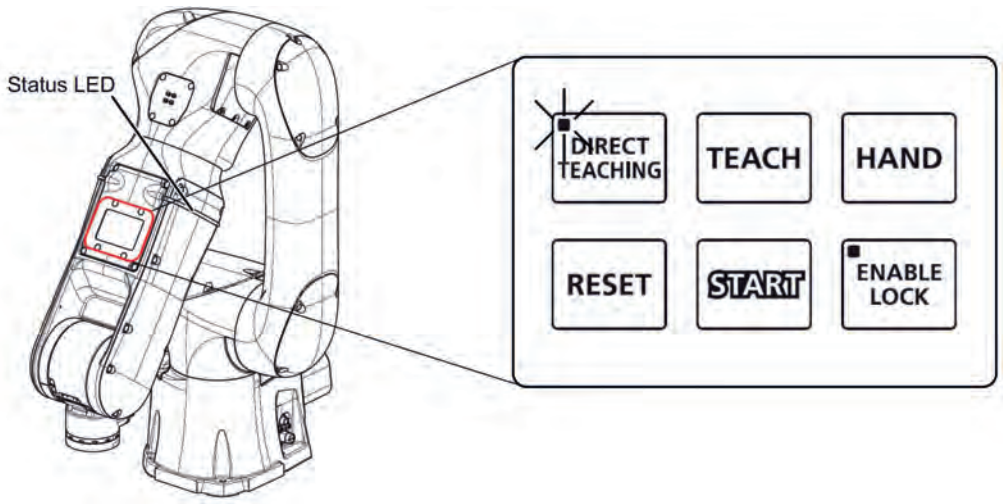
4.2.3.1 Bewegen des Roboters von Hand

In diesem Abschnitt wird erklärt, wie Sie den Roboterarm mit Hilfe der direkten Teach-Funktion beispielhaft in eine gut zugängliche Pose bringen, in der Sie ein Werkzeug am Roboterflansch montieren können. Sollte beim Bewegen des Roboterarms ein Fehler auftritt, drücken Sie die Taste [RESET] auf dem Bedienfeld des Roboterarms, um den Fehler zurückzusetzen.

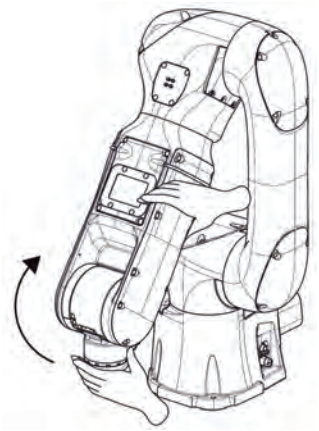
1. Stellen Sie den Betriebsartenschalter an der Hand/Automatik-Umschaltbox auf die Betriebsart AUTOMATIK und ziehen Sie ggf. den Schlüssel des Betriebsartenschalters ab.



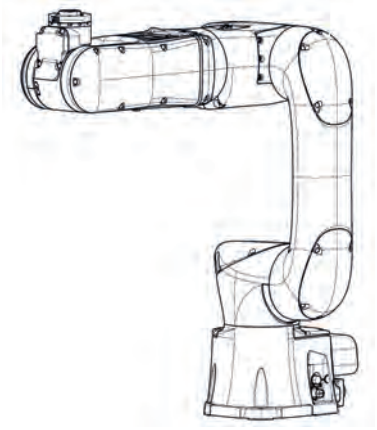
2. Halten Sie die Taste [DIRECT TEACHING] auf dem Bedienfeld des Roboterarms mindestens 2 Sekunden lang gedrückt. Lassen Sie die Taste los, wenn die LED nicht mehr blinkt.
3. Überprüfen Sie, ob die LED der Taste [DIRECT TEACHING] leuchtet oder die LED der Statusanzeige grün blinkt.



4. Halten Sie den Roboterarm mit der Hand fest, und richten Sie die Handmontagefläche nach oben.



5. Bringen Sie den Roboterarm in die unten gezeigte Position und halten Sie dann die Taste [DIRECT TEACHING] auf dem Bedienfeld des Roboterarms auf dem Bedienfeld des Roboterarms mindestens 2 Sekunden lang gedrückt.



6. Vergewissern Sie sich, dass die LED der Taste [DIRECT TEACHING] erloschen ist oder die LED der Statusanzeige von grün auf blau gewechselt hat.

4.2.3.2 Bewegen des Roboters mit dem Handbediengerät

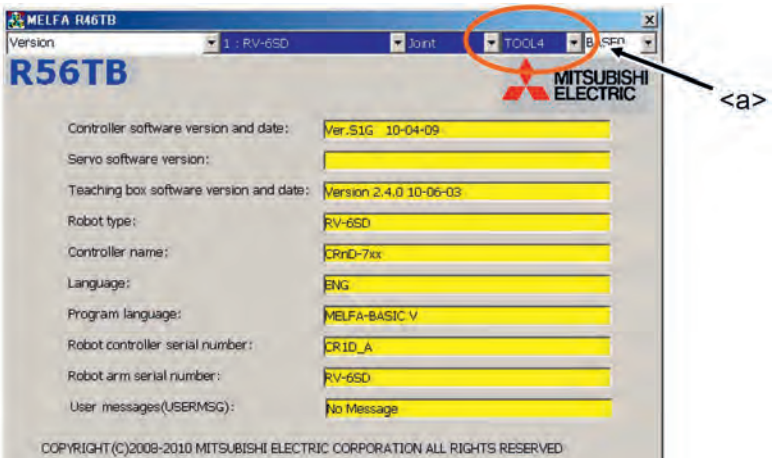
Eine Bewegung des Roboterarms mit dem Handbediengerät setzt ein Verständnis der Bewegungsmöglichkeiten der einzelnen Roboterachsen voraus (siehe Abschnitt 2). Ebenso sollte die Lage des Basis-Koordinatensystems und des Werkzeug-Koordinatensystems bekannt sein (siehe Abschnitt 2.6). Für die Bewegung der Roboterachsen sollte jeweils die Bewegungsgeschwindigkeit angepasst werden.

4.2.3.2.1 Einstellen der Geschwindigkeit

Drücken Sie die Taste [OVRD (oberer Pfeil)] oder [OVRD (unterer Pfeil)] (→ <a>) auf dem Handbediengerät und ändern Sie so die Geschwindigkeit für die nachfolgenden Bewegungen des Roboterarms. Zu Auswahl stehen:
LOW – HIGH – 3% - 5% - 10% - 30% - 50% - 70% - 100%
Die geänderte Geschwindigkeit wird im Kombinationsfeld unten links auf dem Bildschirm des Handbediengerätes angezeigt (→ <b>).
Die Auswahl dieses Kombinationsfeldes ermöglicht ebenfalls die Einstellung der Geschwindigkeit.
Die Werte für LOW und HIGH sind fest vorgegeben.

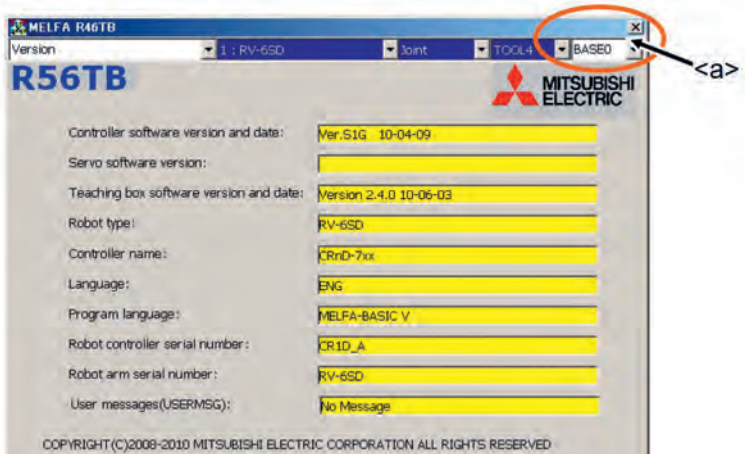
4.2.3.2.2 Auswahl eines Werkzeug-Koordinatensystems

Das für die nachfolgenden Bewegungen des Roboterarms zu betrachtende Werkzeug-Koordinatensystem TOOL0 – TOOL* und damit auch die Werkzeugdateneinstellung MEXTL1 – MEXTL* kann über das über ein Kombinationsfeld (→ <a>) ausgewählt werden. TOOL0 bezeichnet dabei i.d.R. das Koordinatensystem direkt an der Flanschplatte des Roboterarms. Es kann verwendet werden, wenn kein Werkzeug am Roboterarm montiert ist oder die Einstellungen für das montierte Werkzeug unbekannt sind.



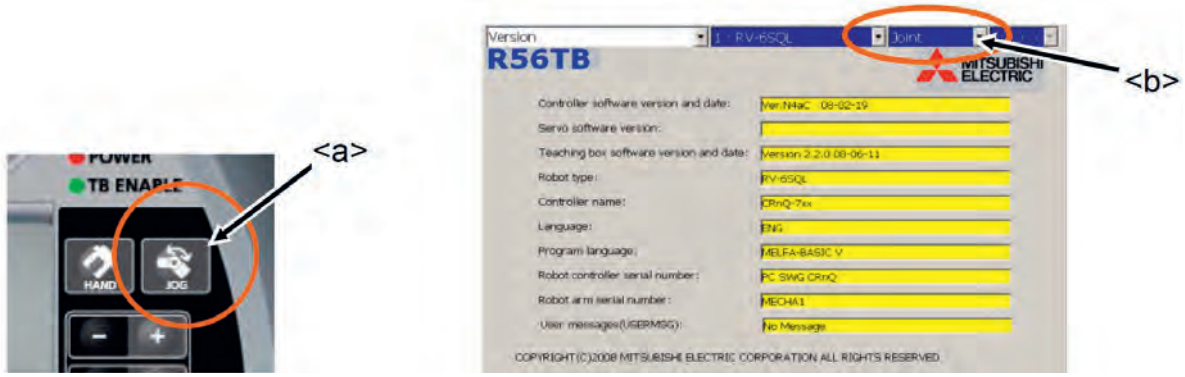
4.2.3.2.3 Auswahl eines Basis-Koordinatensystems

Prinzipiell ist es möglich, auch das zu verwendende Basis-Koordinatensystem zu verändern. Das Basis-Koordinatensystem kann über das Kombinationsfeld in der oberen rechten Ecke des Bildschirms ausgewählt werden (→ <a>). Der Standardfall ist, dass BASE0 als Basis-Koordinatensystem gewählt wird. Dies ist dann das Koordinatensystem im Sockel des Roboterarms.



4.2.3.2.4 Bewegungsarten

Der Roboterarm kann in unterschiedlicher Weise im Tippbetrieb bewegt werden. Es gibt die Modi JOINT, XYZ, TOOL, 3-Achsen XYZ, Cylinder und WORK.
Drücken Sie die [JOG]-Taste (→ <a>) am Handbediengerät mehrmals nacheinander, um zwischen den verschiedenen Modi umzuschalten. Der Tippbetrieb-Modus wird in folgender Reihenfolge geändert:
JOINT -> XYZ -> TOOL -> 3-Achsen XYZ -> Zylinder -> WORK
Der aktuelle Modus wird in der Combo-Box (→ <b>) angezeigt.



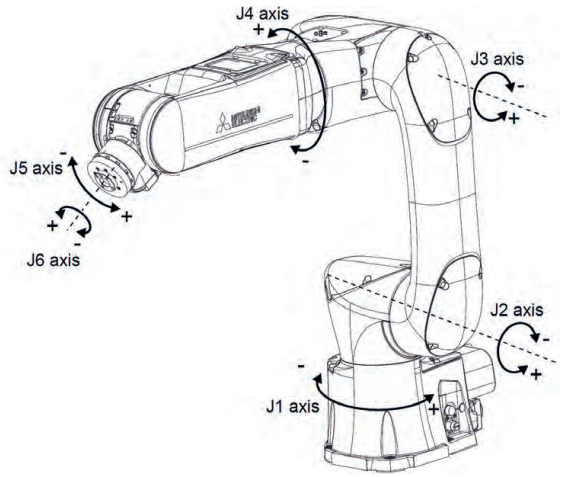
Der Tippbetrieb kann auch über dieses Kombinationsfeld ausgewählt werden.
Für eine Bewegung des Roboterarms müssen generell die Motoren des Roboters eingeschaltet werden und es muss der Freigabeschalter (Totmann-Schalter) (→ <c>) betätigt werden. Die Motoren schalten sich dann ein, wenn die Taste [SERVO] (→ <d>) gedrückt wird. Wenn die Motoren des Roboters eingeschaltet sind, leuchtet die LED (grün) der Taste [SERVO] auf.
Durch Loslassen oder stärkeres Drücken des Freigabeschalters während des Tippbetriebs wird direkt die Stromversorgung der Motoren unterbrochen und der Roboter hält unmittelbar an.

Enable switch

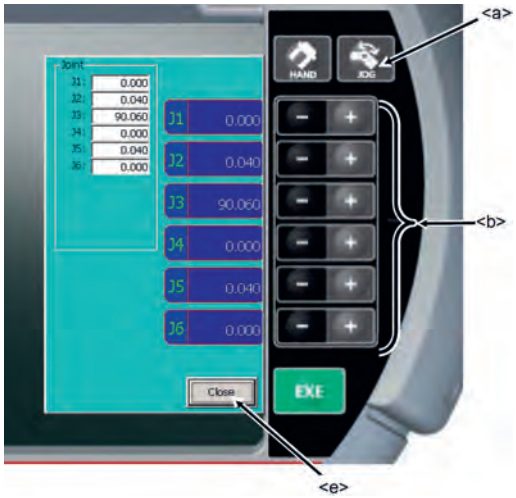
Servo switch

JOINT-Tippbetrieb

Im JOINT-Tippbetrieb können Sie die einzelnen Achsen des Roboters unabhängig voneinander bewegen.

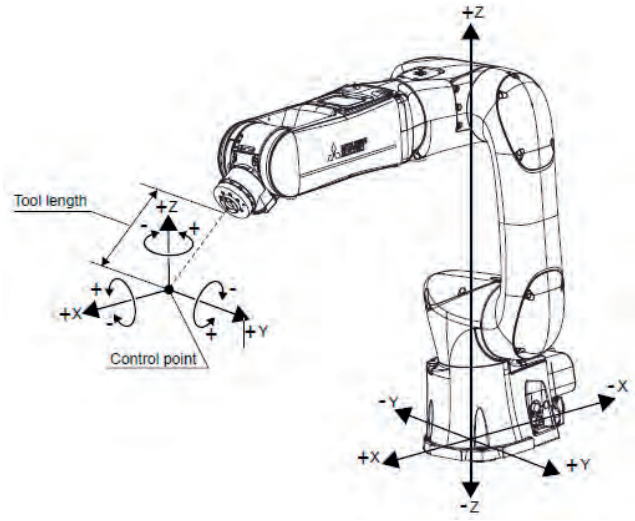


Drücken Sie die [JOG]-Taste (→ <a>) auf dem Handbediengerät so oft, bis der JOG-Betriebsbildschirm im Modus JOINT angezeigt wird.

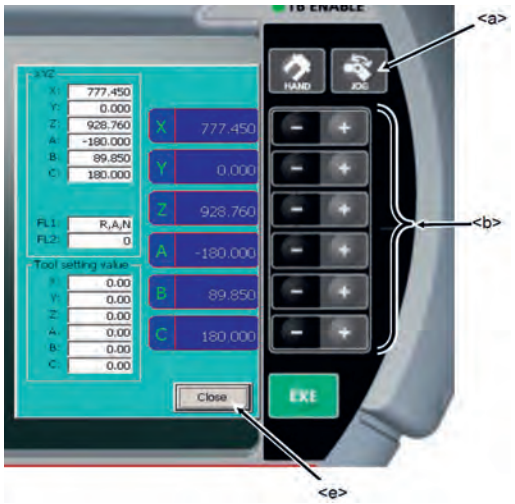


XYZ-Tippbetrieb

Im XYZ-Tippbetrieb können Sie den Roboter im kartesischen Raum entlang der Koordinatenachsen des Basis-Koordinatensystems bewegen.



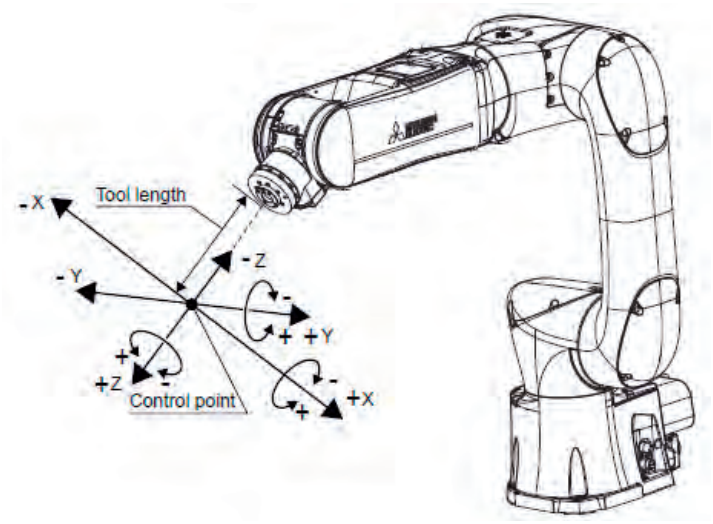
Drücken Sie die [JOG]-Taste (→ <a>) auf dem Handbediengerät so oft, bis der JOG-Betriebsbildschirm im Modus XYZ angezeigt wird.



Drücken Sie die Tasten (+ oder -) der gewünschten Bewegungsrichtung. Die Roboterachsen bewegen sich dadurch koordiniert und das Werkzeug (soweit ein Werkzeug am Flansch befestigt wurde) bewegt sich in die jeweilige Richtung, solange wie die Taste gedrückt wird.
Um den Tippbetrieb zu verlassen, drücken Sie auf die Schaltfläche [Schließen] (→ <e>) auf dem Bildschirm und schließen Sie damit den JOG-Betriebsbildschirm.

TOOL-Tippbetrieb

Im TOOL-Tippbetrieb können Sie den Roboter im kartesischen Raum entlang der Koordinatenachsen des aktuellen Werkzeug-Koordinatensystems bewegen.



Drücken Sie die [JOG]-Taste (→ <a>) auf dem Handbediengerät so oft, bis der JOG-Betriebsbildschirm im Modus TOOL angezeigt wird.

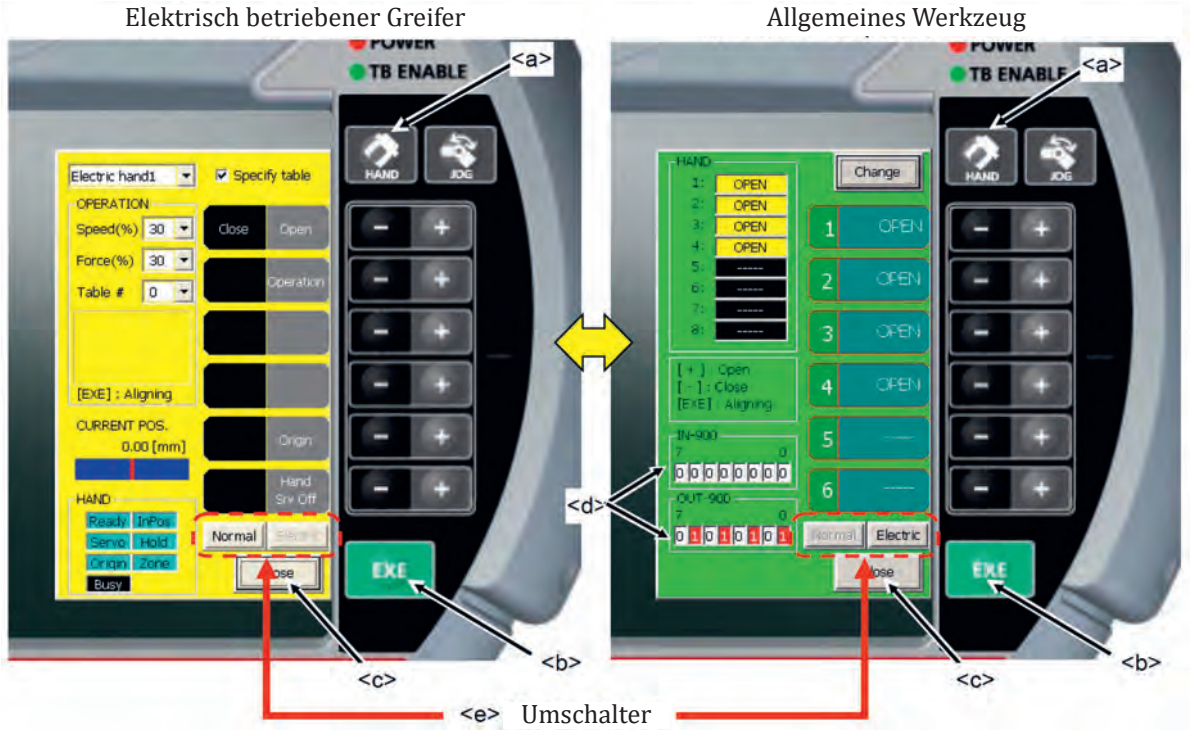
Drücken Sie die Tasten (+ oder -) der gewünschten Bewegungsrichtung. Die Roboterachsen bewegen sich dadurch koordiniert und das Werkzeug (soweit ein Werkzeug am Flansch befestigt wurde) bewegt sich in die jeweilige Richtung, solange wie die Taste gedrückt wird.
Um den Tippbetrieb zu verlassen, drücken Sie auf die Schaltfläche [Schließen] (→ <e>) auf dem Bildschirm und schließen Sie damit den JOG-Betriebsbildschirm.

3-Achsen-XYZ- / Zylinder- / WORK-Tippbetrieb

Diese drei Tippbetrieb-Modi funktionieren analog zu den zuvor beschriebenen Modi – sie bieten aber nicht grundsätzlich bessere Möglichkeiten für die Bewegung des Roboterarms. Bei Bedarf und für weitere Informationen hierzu bitte in das entsprechende originale Mitsubishi-Handbuch schauen.

4.2.3.2.5 Ausrichten des Werkzeugs

Die Ausrichtung des am Roboter befestigten Werkzeugs kann 90 Grad-Schritten relativ zum Basis-Koordinatensystem des Roboters geändert werden. Diese Funktion bewegt den Roboter (und damit das Werkzeug) in eine Pose, in der die A-, B- und C-Komponenten auf die nächstgelegenen 90-Grad-Werte.
Wenn die Werkzeugkoordinaten durch den Befehl TOOL oder die entsprechenden Parameter angegeben sind, wird das Werkzeug den angegebenen Werkzeugkoordinatenachsen ausgerichtet. Wenn die Werkzeugkoordinaten nicht angegeben sind, wird Flanschplatte entsprechend ausgerichtet.
Drücken Sie die Taste [HAND] (→ <a>) auf dem Handbediengerät und zeigen Sie damit den Werkzeugbedienungsbildschirm an.



Beim Ausrichten des Werkzeugs muss sich die Servo-Stromversorgung im EIN-Zustand befinden und der Freigabeschalter am Handbediengerät muss weiterhin gedrückt bleiben. Wenn der Freigabeschalter gedrückt ist und die Taste [EXE] (→ <b>) gedrückt wird, bewegt sich der Roboter in Richtung der Ausrichtungspose. Während der Bewegung des Roboters wechselt das Symbol unten links auf dem Handbediengerät zwischen und wechselt nach Erreichen der Ausrichtung auf das Symbol .

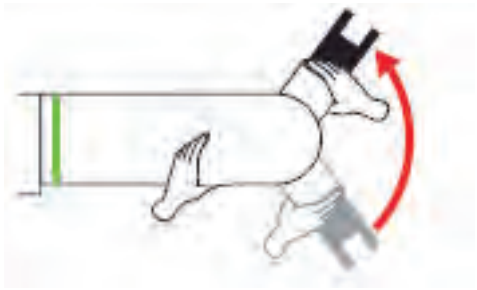
Wenn der Vorgang abgeschlossen ist, tippen Sie auf die Schaltfläche [Schließen] (→ <c>) auf dem Bildschirm und schließen Sie den Werkzeugbedienungsbildschirm.

4.2.4 Programmierung des Roboters

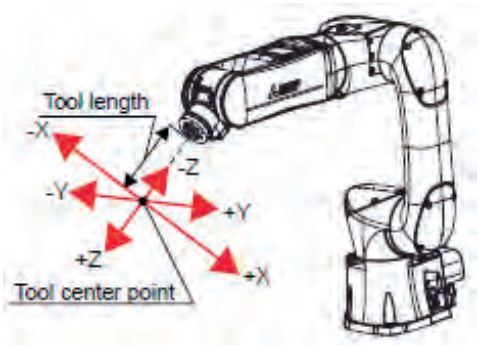
Der Mitsubishi-Roboter kann auf unterschiedliche Weisen programmiert werden. Dazu können z. B. die Bedientasten am Roboterarm, die Software *RT VisualBox* oder das Simulationsprogramm *RT Toolbox3* verwendet werden. Es kann allerdings kann nur eine dieser Möglichkeiten zu einer Zeit verwendet werden. Dadurch werden dann auch die "Betriebsrechte" zur Steuerung des Roboters an den Bediener vergeben. Es ist unbedingt darauf zu achten, dass andere Personen dann keine Betriebsrechte erhalten können, auch wenn Sie sich in der Nähe des Roboters aufhalten, während er in Betrieb ist.

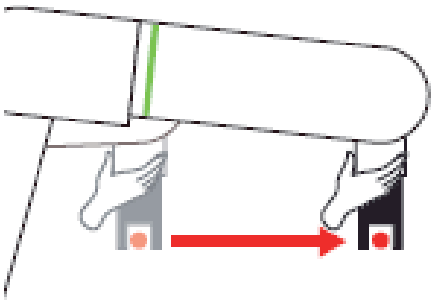
4.2.4.1 Direktes Teachen des Roboters

Es gibt drei Modi beim direkten Teachen.
Gelenkmodus:
Die Achsen können einzeln oder gemeinsam in die gewünschte Position gebracht werden. Zu beachten: Wenn die J4-Achse und die J6-Achse aufeinander ausgerichtet sind (→ Handachsen-Singularität), kann die J4-Achse rotieren, während die J6-Achse bewegt wird.

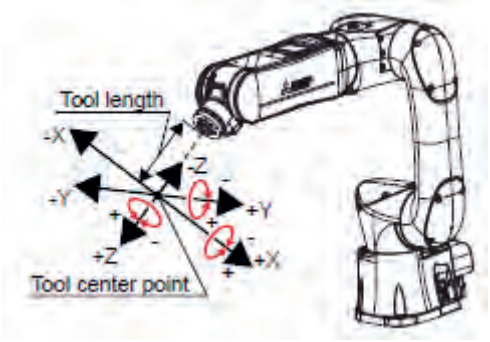


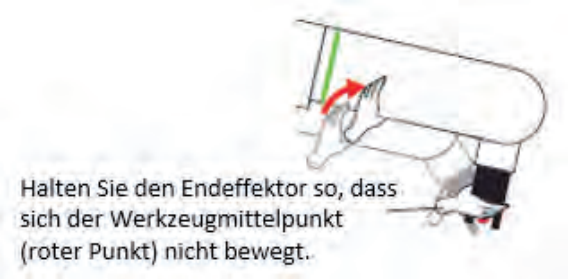
Translationsmodus:
Der Roboterarm kann geradlinig relativ zum Werkzeugmittelpunkt in die Koordinatenrichtungen des Werkzeugkoordinatensystems bewegt werden.





Rotationsmodus:
Der Roboterarm kann um den Werkzeugmittelpunkt im Werkzeugkoordinatensystem gedreht werden.





Zu beachten: Wenn sich der Werkzeugmittelpunkt (roter Punkt) selbst translatorisch bewegt, tritt ein Drehmomentfehler (H221n) auf.

4.2.4.1.1 Direktes Teachen mit den Bedientasten

Gehen Sie wie folgt vor:

1. Stellen Sie den Betriebsartenschalter an der Hand/Automatik-Umschaltbox auf die Betriebs¬art AUTOMATIK
2. Halten Sie die Taste [DIRECT TEACHING] mindestens 2 Sekunden lang gedrückt, um das di-rekte Teachen zu aktivieren. Prüfen Sie, ob die LED der Taste [DIRECT TEACHING] leuchtet oder die Statusanzeige-LED grün leuchtet oder blinkt.
3. Halten Sie den Roboterarm direkt mit Ihren Händen und bewegen Sie ihn in die gewünschte Pose.
4. Drücken Sie die Taste [DIRECT TEACHING] mehrfach nacheinander, um den Teach-Modus zu wechseln. Der Modus wechselt in der folgenden Reihenfolge: Gelenkmodus → Translationsmodus → Rotationsmodus → Gelenkmodus
5. Drücken Sie die [TEACH]-Taste einmal, um die aktuelle Pose einzulernen. Auf dieser Weise wird die aktuelle Pose an RT VisualBox übertragen.
6. Halten Sie die Taste [DIRECT TEACHING] erneut für mindestens 2 Sekunden lang gedrückt, um das direkte Teachen zu deaktivieren. Prüfen Sie, ob die LED der Taste [DIRECT TEACHING] erlischt oder die LED der Statusanzeige von grün auf blau wechselt.

   Das LED-Verhalten der Taste [DIRECT TEACHING] ist wie folgt zu deuten:
   - Aus: Direktes Teachen gestoppt
   - An: Direktes Teachen ist eingeschaltet (Gelenkmodus)
   - Schnelles Blinken: Direktes Teachen ist eingeschaltet (Translationsmodus)
   - Langsames Blinken: Direktes Teachen ist eingeschaltet (Rotationsmodus)

Zu beachten: Das Programm RT VisualBox ist erforderlich, um Positionen zu teachen.

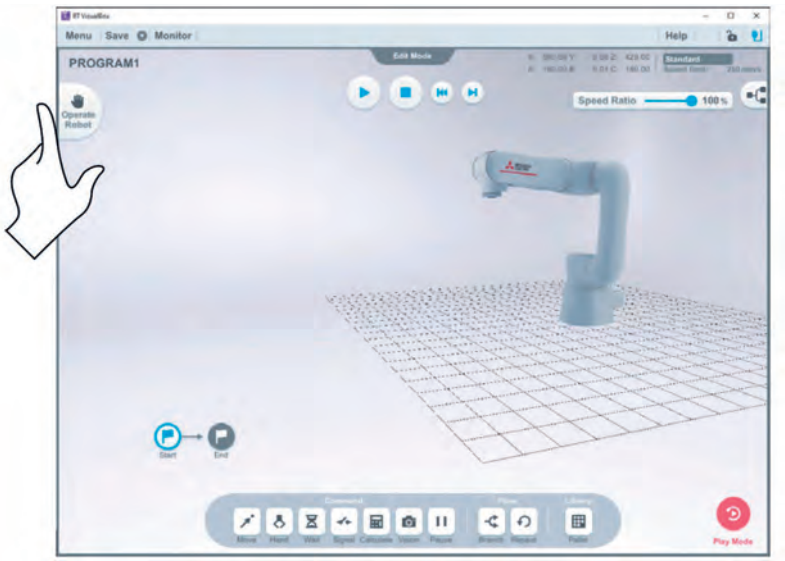
Das direkte Teachen wird unter den folgenden Bedingungen ausgeschaltet:

- Wenn der Roboterarm nach dem Einschalten des direkten Teachens eine bestimmte Zeit lang nicht bewegt wird (der Anfangswert beträgt 60 Sekunden). Um die Zeit bis zum automatischen Abschalten des direkten-Teachens zu ändern, ändern Sie den Parameter DTTMR (siehe hierzu in den originalen Mitsubishi-Handbüchern nach).
- Wenn Stoppsignale eingegeben werden
- Wenn ein Fehler auftritt.
- Wenn die Kommunikation zum Programm RT VisualBox für 30 Sekunden unterbrochen wird.

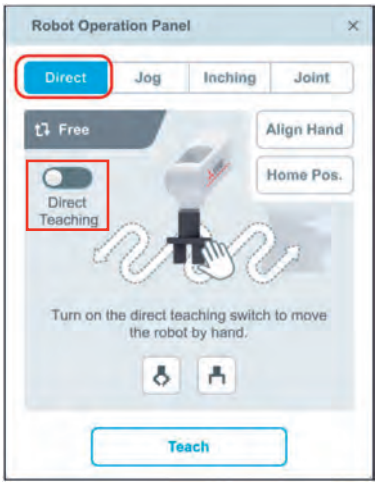
4.2.4.1.2 Direktes Teachen in RT VisualBox

Gehen Sie wie folgt vor:

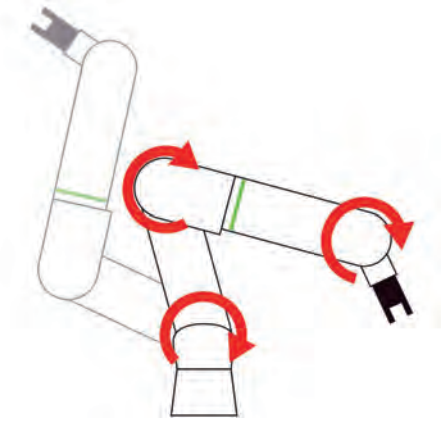
1. Stellen Sie den Betriebsartenschalter an der Hand/Automatik-Umschaltbox auf die Betriebs¬art AUTOMATIK
2. Starten Sie das Programm RT VisualBox am PC.
3. Tippen/Klicken Sie auf die Schaltfläche [Operate Robot], um das Bedienfeld für den Roboter anzuzeigen.



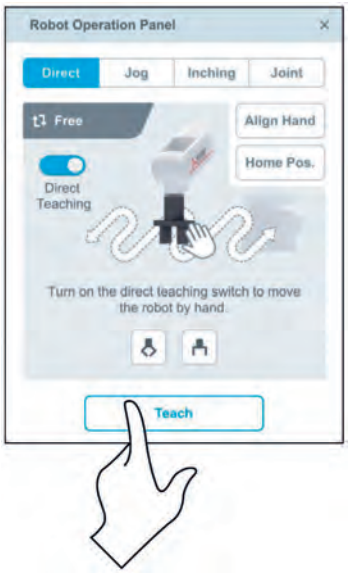
4. Wählen Sie im Bedienfeld des Roboters die Registerkarte [Direct].
5. Tippen/klicken Sie auf den Schalter [Direct Teaching] im Bedienfeld des Roboters, um das direkte Teachen zu aktivieren. Das direkte Teachen bleibt aktiviert, bis der Schalter [Direct Teaching] wieder ausgeschaltet wird.



6. Halten Sie den Roboterarm direkt mit Ihren Händen und bewegen Sie ihn in die gewünschte Pose.



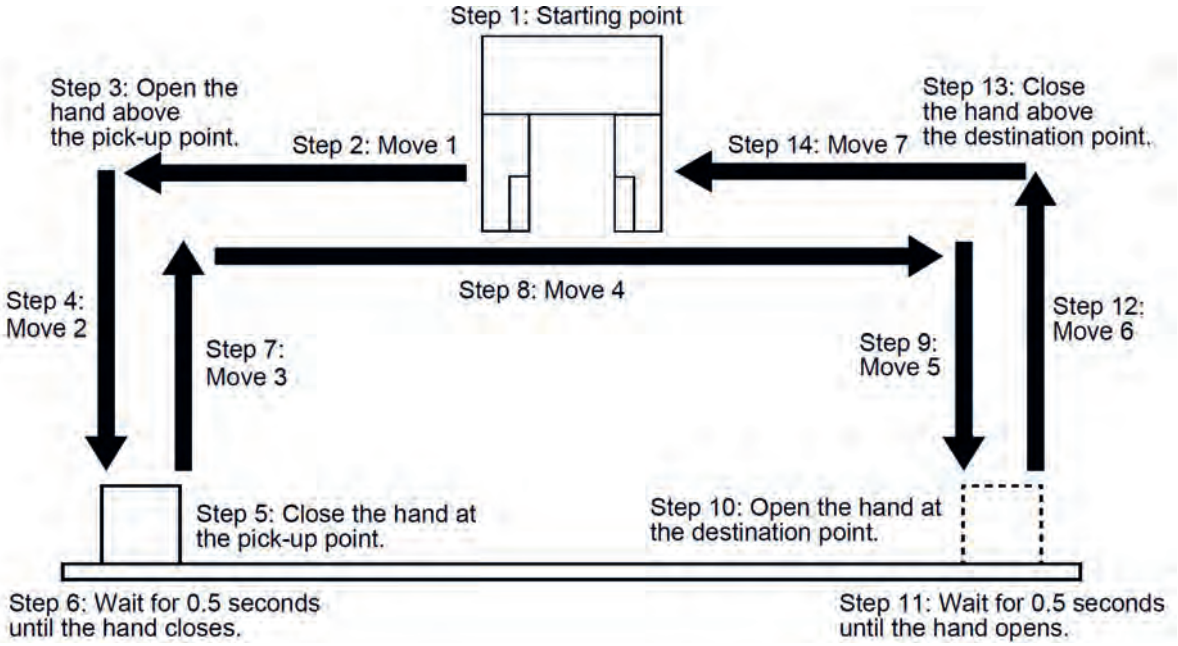
7. Tippen/Klicken Sie auf die Schaltfläche [Teach] im Bedienfeld des Roboters, um die aktuelle Pose zu teachen. Das Programm RT VisualBox ist erforderlich, um Posen über diese Schaltfläche zu teachen.



8. Tippen/klicken Sie erneut auf den Schalter [Direct Teaching] im Bedienfeld des Roboters, um das direkte Teachen zu deaktivieren.

4.2.4.2 Ein Roboterprogramm mit RT VisualBox erstellen/editieren

In diesem Abschnitt wird erklärt, wie Sie ein Roboterprogramm mit der Bedienersoftware RT VisualBox erstellen und bearbeiten können.
Die folgende Abbildung zeigt einen Überblick über das hier beispielhaft zu erstellende Programm.



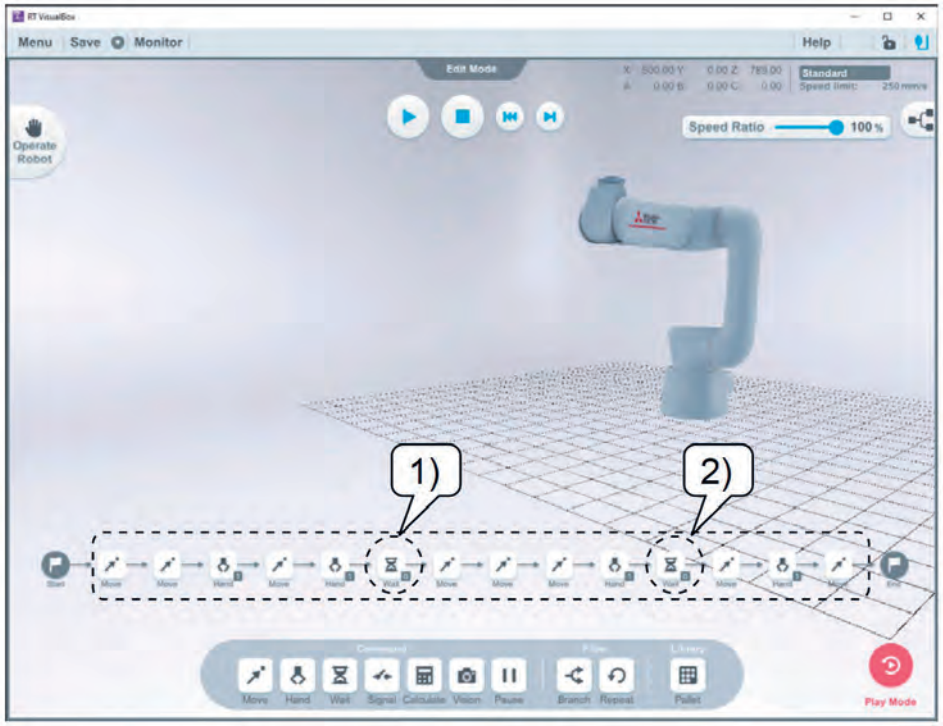
Gehen Sie wie folgt vor:

1. Ziehen Sie einen [Move]-Block in den Bereich zwischen Start und Ende, um eine Startpose festzulegen.

2. Ziehen Sie einen weiteren [Move]-Block in den Bereich zwischen dem in Schritt 1 platzierten Block und Ende, um die Pose oberhalb des Aufnahmepunkts festzulegen.

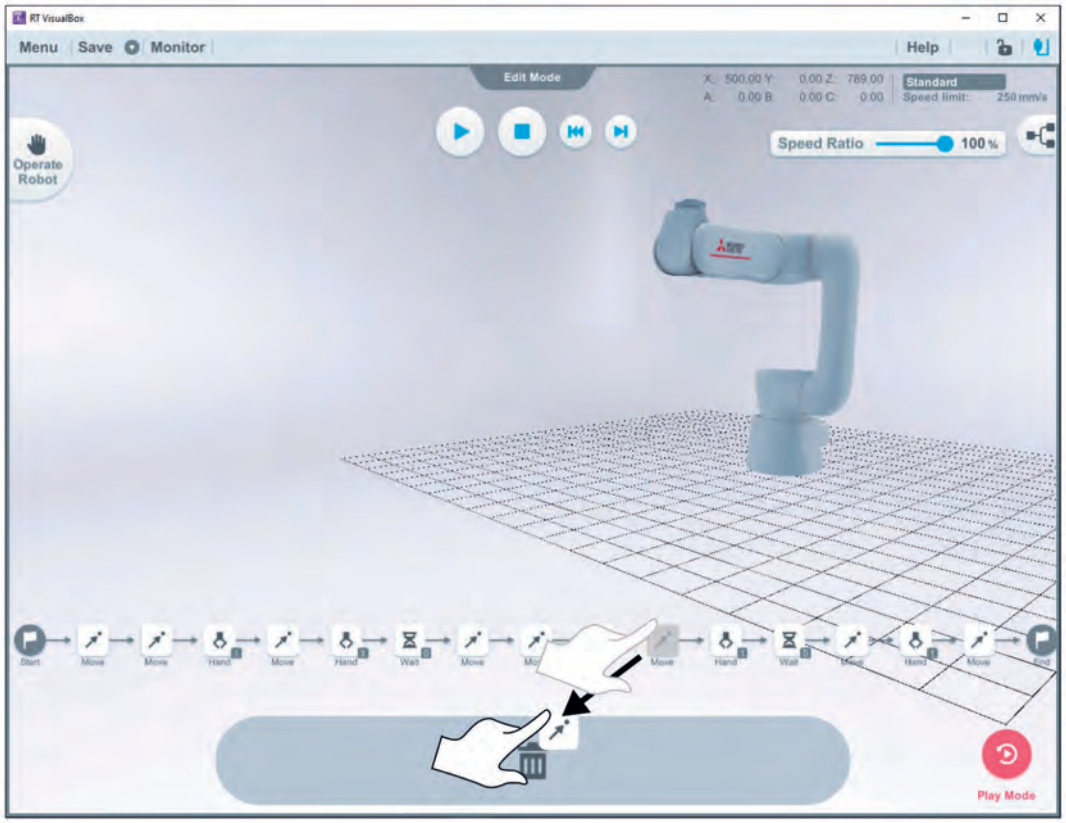
3. Ziehen Sie einen [Hand]-Block in den Bereich zwischen dem in Schritt 2 platzierten Block und End, um die Position für das Öffnen für das Öffnen der Hand über dem Aufnahmepunkt festzulegen.

4. Wiederholen Sie für die Programmschritte 4 bis 14 die zuvor beschriebenen Schritte, um das Programm wie in der Abbildung zu erstellen.



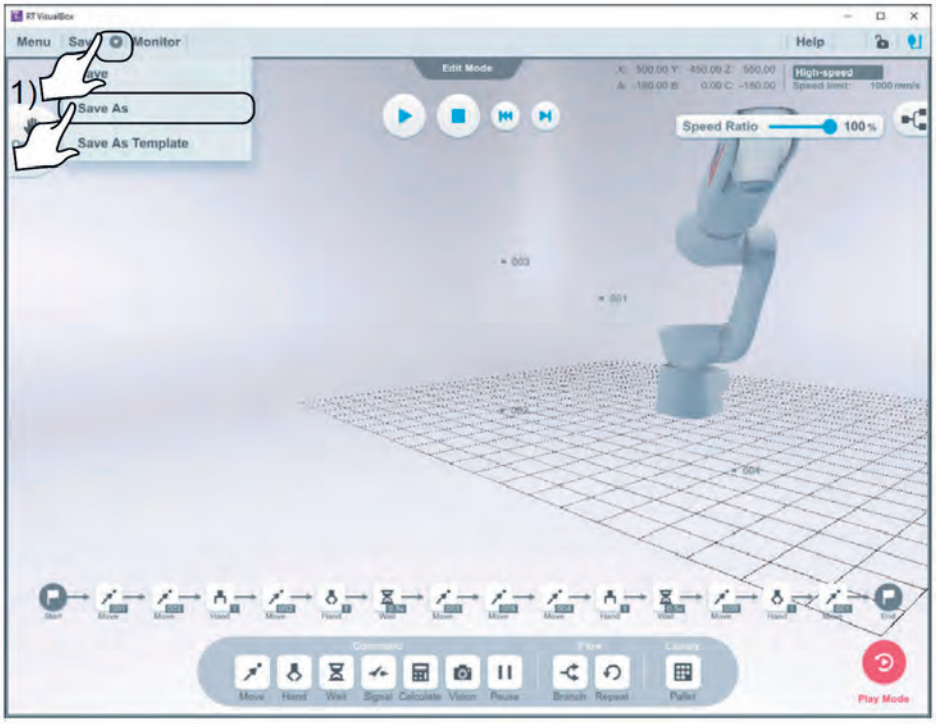
5. Setzen Sie einen [Wait]-Block an die mit den Nummern 1 und 2 markierten Stellen.

Zum Löschen unnötiger Blöcke gehen Sie wie folgt vor:
Wählen Sie den zu löschenden Programmblock im Programm aus und ziehen Sie ihn nach unten. Es erscheint dann dort ein Mülleimer. Um den Block zu löschen, bewegen Sie ihn in den Mülleimer und lassen ihn los.

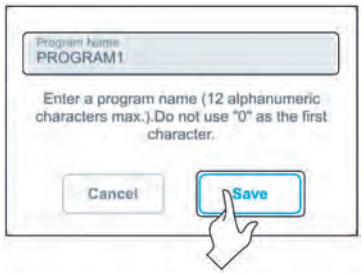


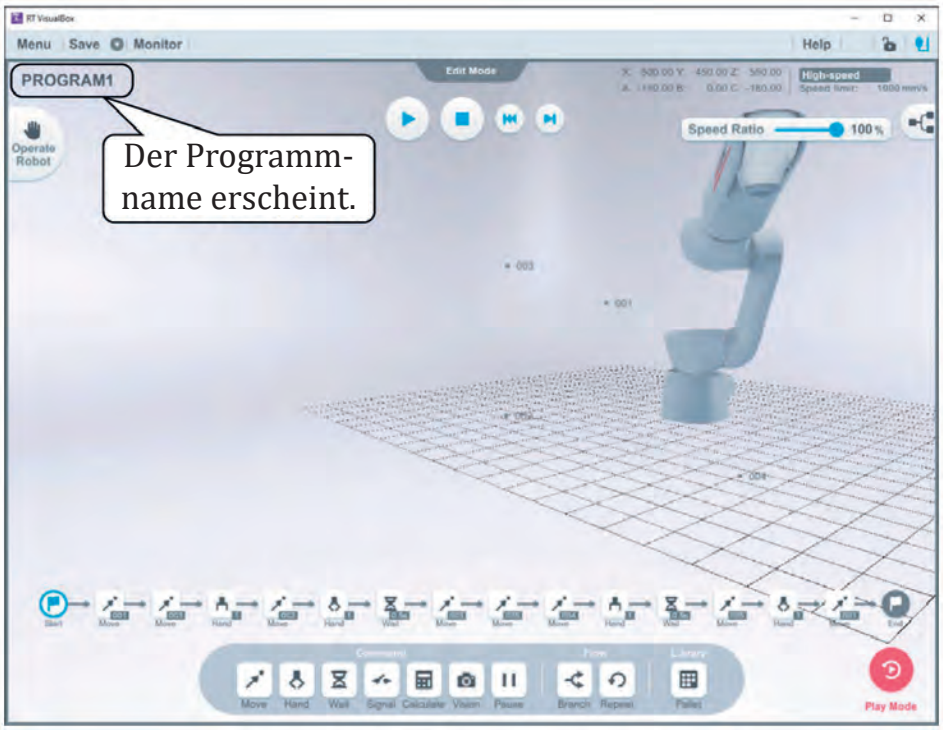
Zum Speichern eines in RT VisualBox erstellen Programms gehen Sie wie folgt vor:

1. Tippen/klicken Sie auf die Schaltfläche rechts neben [Speichern] (→ 1).



2. Wählen Sie [Save As].
3. Geben Sie einen Namen in das Feld ‚Program Name' ein.
4. Nach dem Speichern des Programms erscheint der Programmname in der oberen linken Ecke des Bildschirms.





Zu beachten: Die Daten werden direkt in der Robotersteuerung gespeichert.

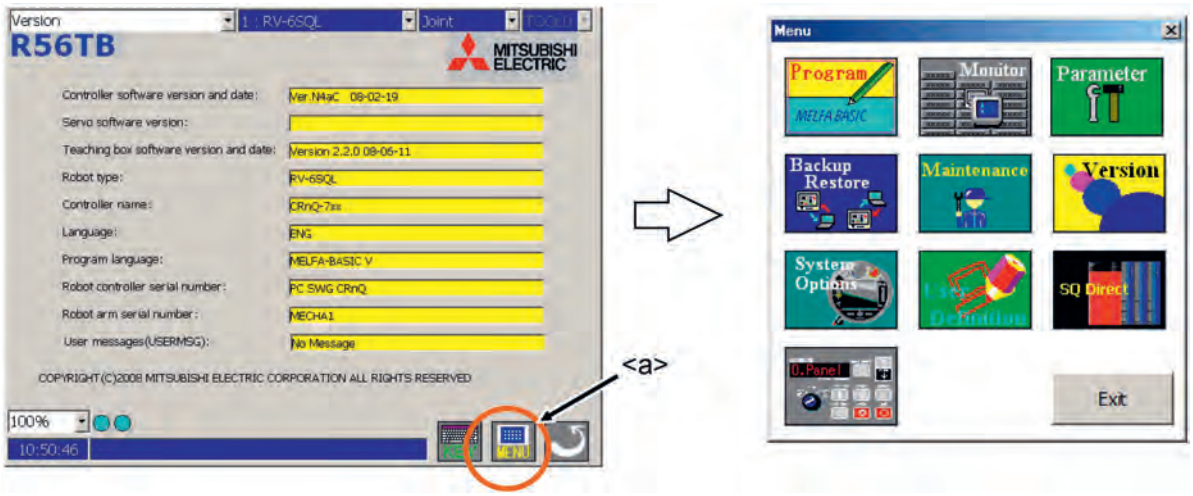
4.2.4.3 Ein Roboterprogramm mit dem Handbediengerät erstellen/editieren

Auch mit dem Handbediengerät können Roboterprogramme erstellt, editiert und verwaltet werden.

4.2.4.3.1 Erstellen eines neuen Roboterprogramms

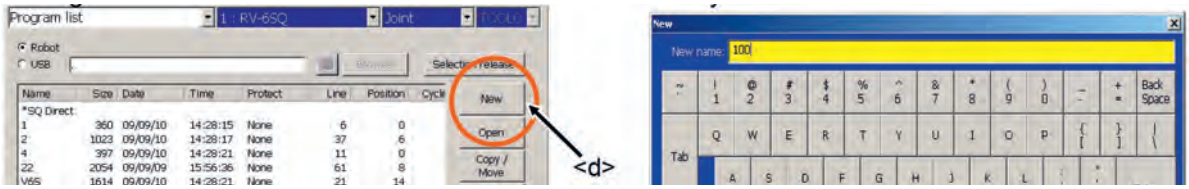
Gehen Sie wie folgt vor:

1. Tippen Sie auf dem Bildschirm des Handbediengerätes auf die Taste [MENU] (→ <a>), um das Menüfeld anzuzeigen.

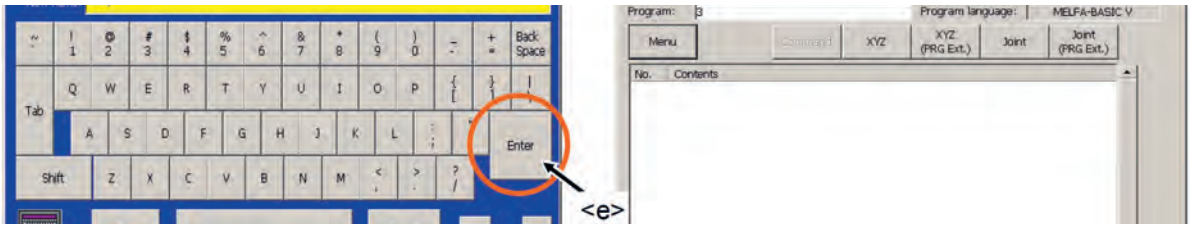


2. Tippen Sie auf die Schaltfläche [Program] (→ <b>) und eine Programmliste wird angezeigt. Wenn zu diesem Zeitpunkt ein neues Programm für eine Robotersteuerung erstellt wird, ist das Optionsfeld [Roboter] (→ <c>) aktiviert. Wenn das Programm neu auf dem USB-Speicher erstellt wird, ist die Optionsschaltfläche [USB] (→ <c>) zu aktivieren.

3. Tippen Sie auf die Schaltfläche [New] (→ <d>) und eine Tastatur wird angezeigt.



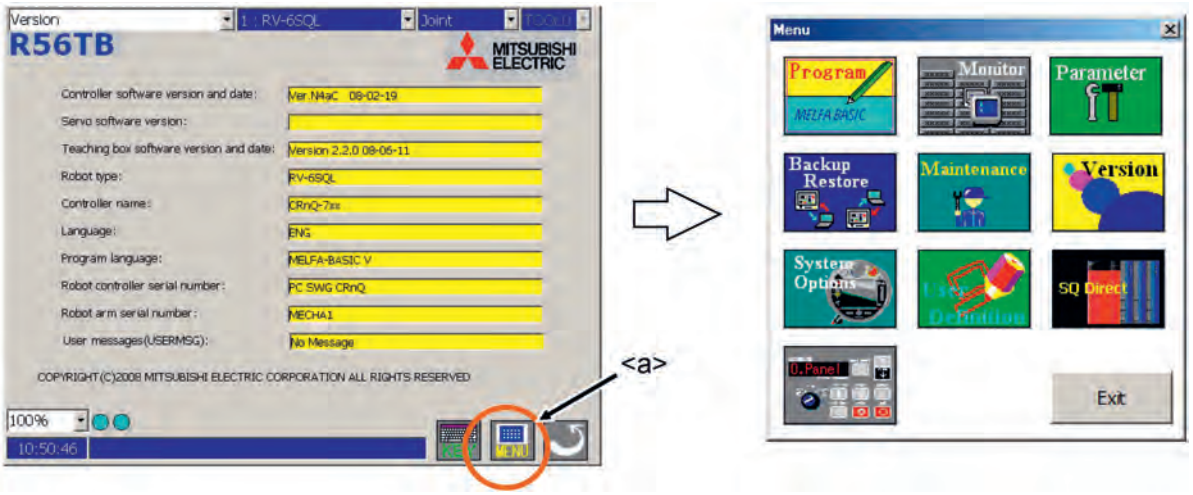
4. Wenn ein Programmname über die Tastatur des Bildschirms eingegeben und die [Enter]-Taste (→ <e>) gedrückt wird, wird der leere Bearbeitungsbildschirm zur Erstellung des neuen Roboterprogramms angezeigt. Der Programmname wird oben angezeigt.



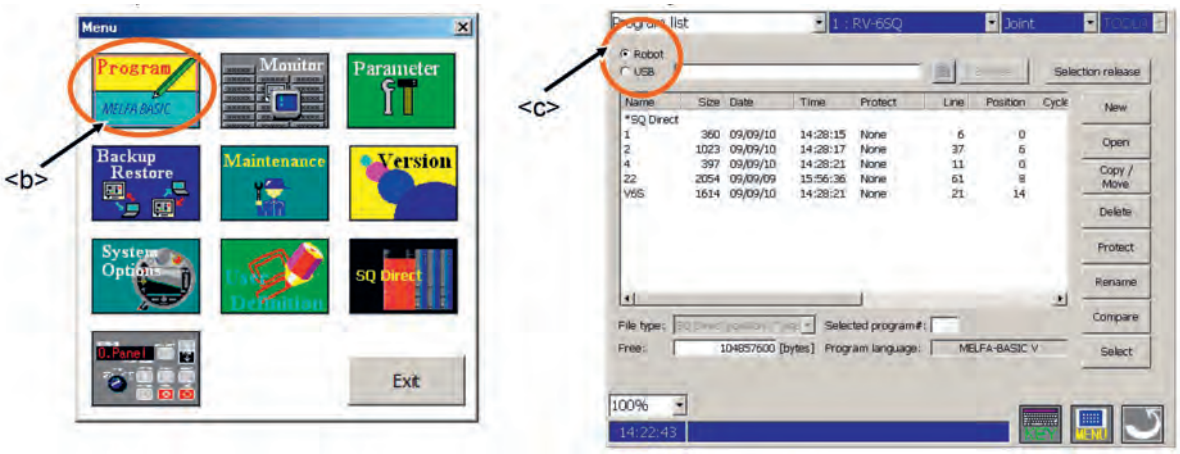
4.2.4.3.2 Öffnen eines existierenden Roboterprogramms

Gehen Sie wie folgt vor:

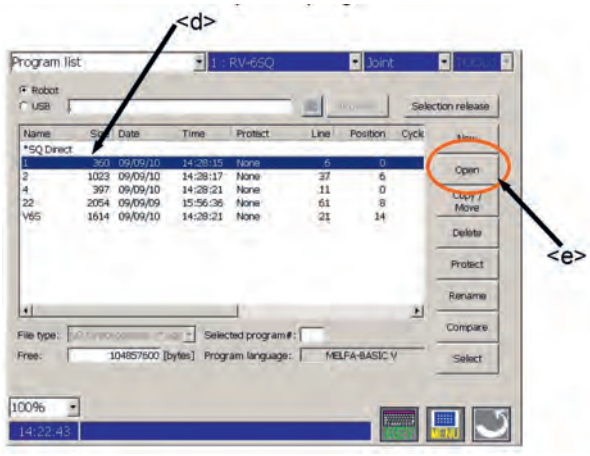
1. Tippen Sie auf dem Bildschirm des Handbediengerätes auf die Taste [MENU] (→ <a>), um das Menüfeld anzuzeigen.



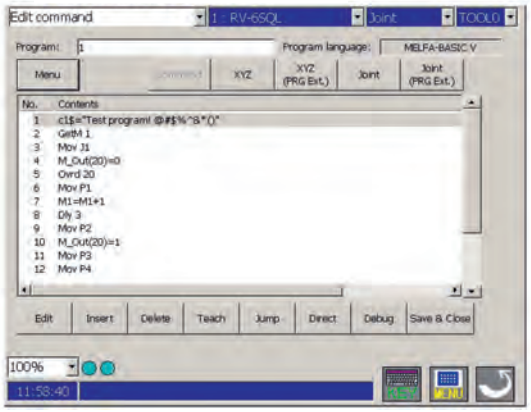
2. Tippen Sie auf die Schaltfläche [Program] (→ <b>) und die Programmliste wird angezeigt.



3. Wählen Sie das zu öffnende Programm aus (→ <d>) und tippen Sie auf die Schaltfläche [Open] (→ <e>). Der Bildschirm zur Befehlsbearbeitung wird angezeigt.

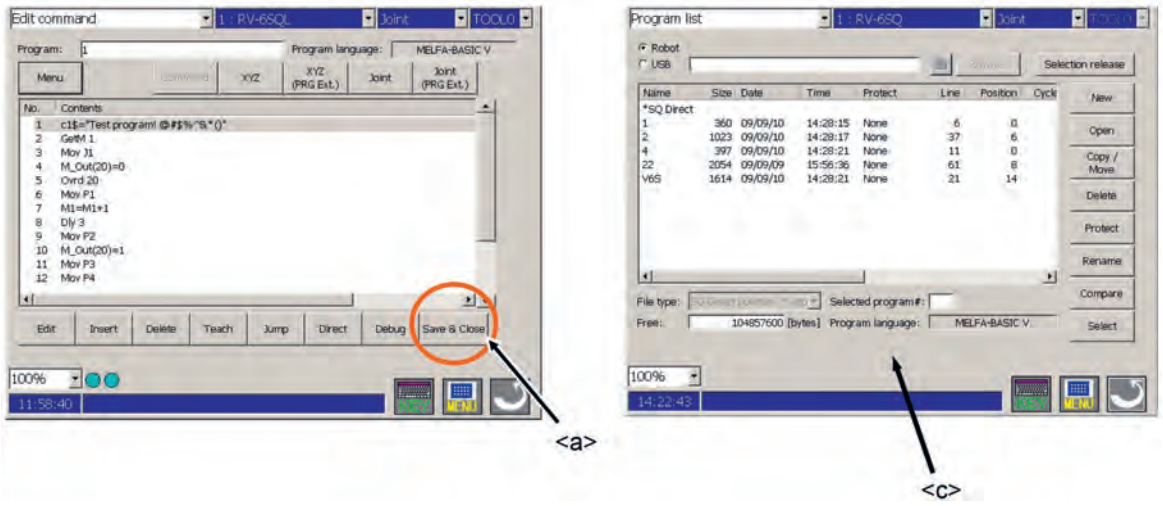


Befehlsbearbeitung



4.2.4.3.3 Speichern eines Roboterprogramms

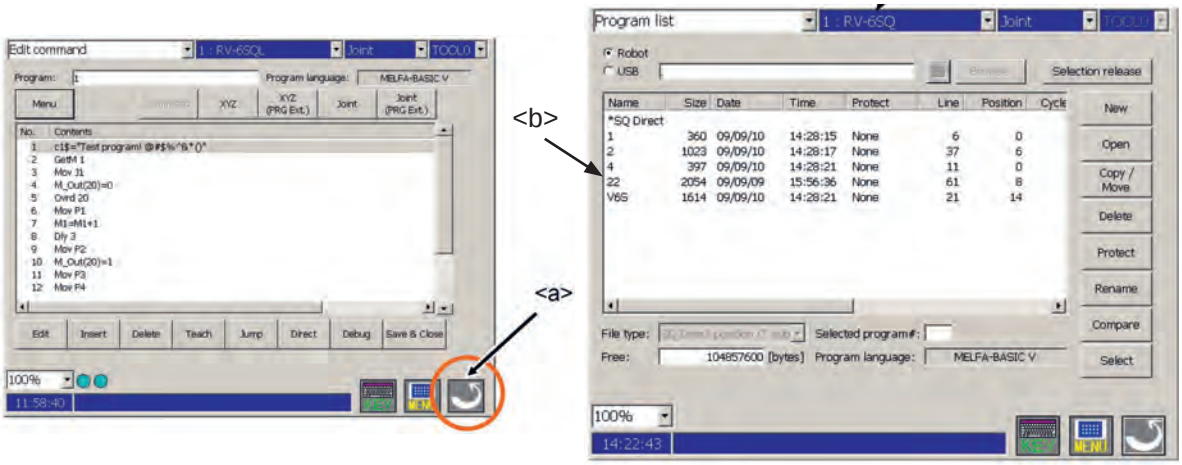
1. Tippen Sie auf die Schaltfläche [Save & Close] (→ <a>) auf dem Bildschirm.



Während des Speichervorgangs wird ein Fortschrittsbalken angezeigt. Nach Abschluss des Speichervorgangs kehrt der Bildschirm zur Programmliste zurück (→ <c>).

4.2.4.3.4 Schließen des Bearbeitungsbildschirms

Tippen Sie auf die [RETURN] Schaltfläche (→ <a>) in der Ecke unten rechts des Handbediengerätes. Es wird dann wieder die Programmliste angezeigt (→ <b>).

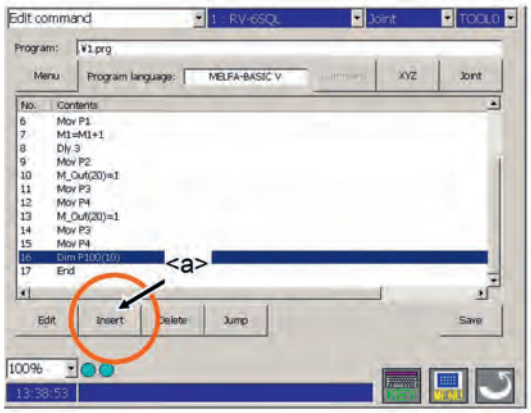


Wenn zuvor Programmbefehle und/oder Positionsvariablen bearbeitet wurden erscheint eine Sicherheitsabfrage, die bestätigt werden muss. Die Änderungen an dem Programm werden dann nicht gespeichert.

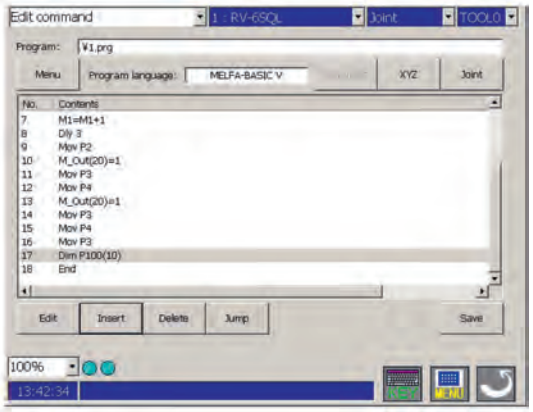
4.2.4.3.5 Eingabe einer Befehlszeile

Der Vorgang des Hinzufügens einer Codezeile zu einem neuen oder bestehenden Programm wird auf dem Bildschirm für die Befehlsbearbeitung durchgeführt.

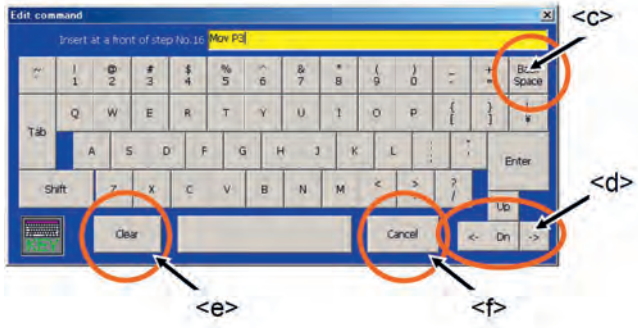
Gehen Sie wie folgt vor:
Durch Tippen auf die Schaltfläche [Insert] (→ <a>) auf dem Bildschirm zur Befehlsbearbeitung wird die Tastatur angezeigt.

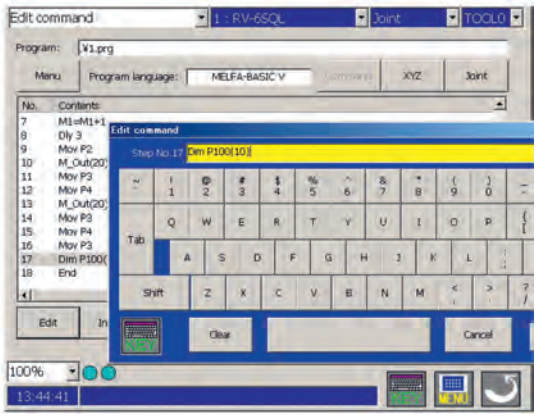


Geben Sie die Codezeile über die Tastatur ein und tippen Sie auf die Taste [Enter] (→ <b>).
Die eingegebene Codezeile wird in der Reihenfolge der Zeilennummer hinzugefügt. Wenn die Taste [Cancel] (→ <f>) angetippt wird, wird die Tastatur geschlossen.



Wenn das eingegebene Zeichen korrigiert werden soll, löschen Sie es mit der [Back Space]-Taste. (→ <c>). Bewegen Sie den Cursor mit den Cursortasten [<-] und [->] (→ <d>) auf die rechte Seite des fehlerhaften Zeichens und tippen Sie auf die Taste [Leertaste]. Wenn die Taste [Clear] (→ <e>) der Tastatur angetippt wird, kann der Programmbefehl gelöscht werden und die Zeichen kann erneut eingegeben werden. Außerdem wird durch Tippen auf die Taste [Cancel] (→ <f>) die Eingabe beendet.

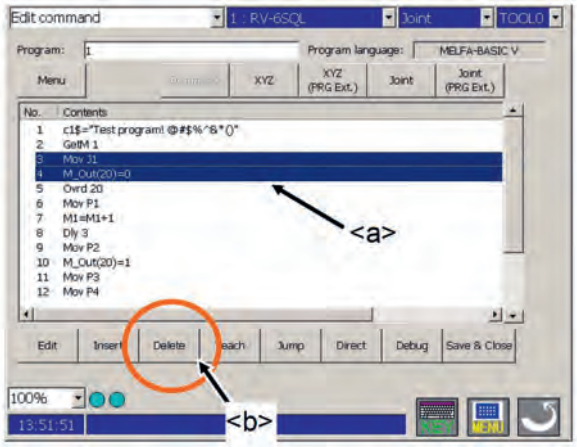


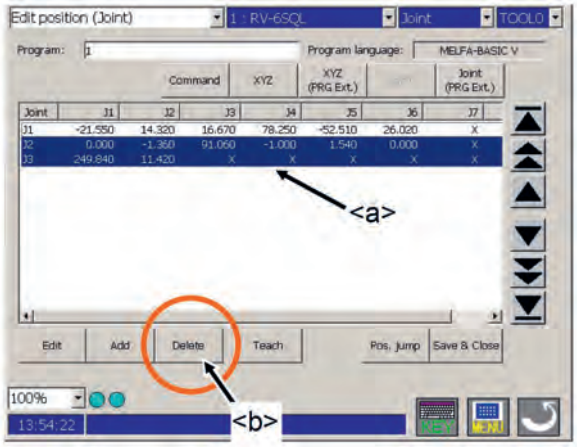


4.2.4.3.6 Löschen einer Befehlszeile

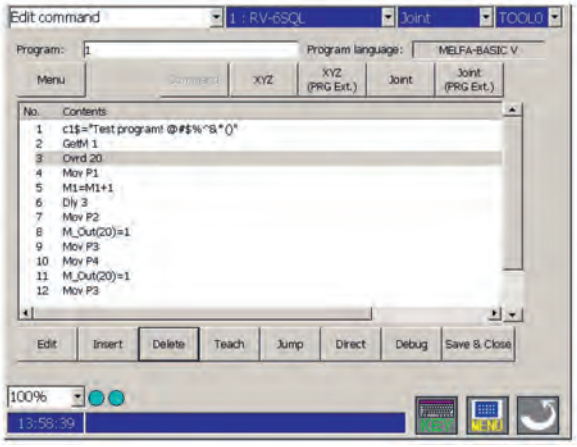
Der Vorgang zum Löschen einer Codezeile oder einer Positionsvariablen verläuft wie folgt:

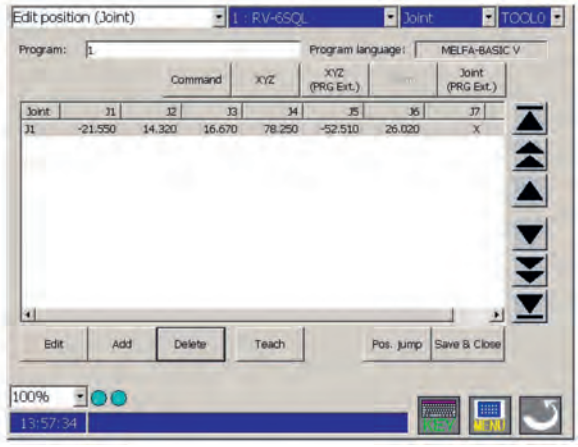
1. Wählen Sie die zu löschende Zeile oder die Positionsvariable aus. Es können auch mehrere Zeilen markiert werden (→ <a>)



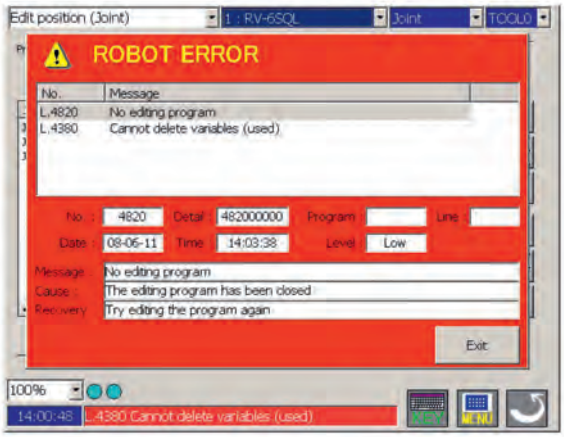


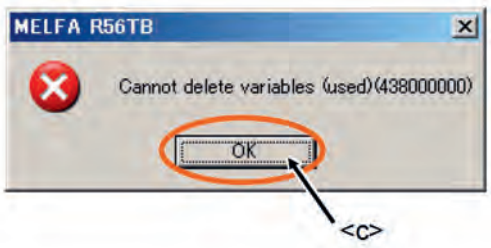
2. Tippen Sie auf die Schaltfläche [Delete] (→ <b>). Die ausgewählte Codezeile oder die Positionsvariable wird gelöscht.





Hinweis: Es tritt ein Fehler auf, wenn eine Positionsvariable gelöscht werden soll, die derzeit in einer Codezeile verwendet wird und sich das dazu gehörende Programm in der Robotersteuerung bearbeiten wird. Drücken Sie die Taste [RESET] auf dem Handbediengerät und schließen Sie die Meldung mit der Taste [OK]. (→ <c>)

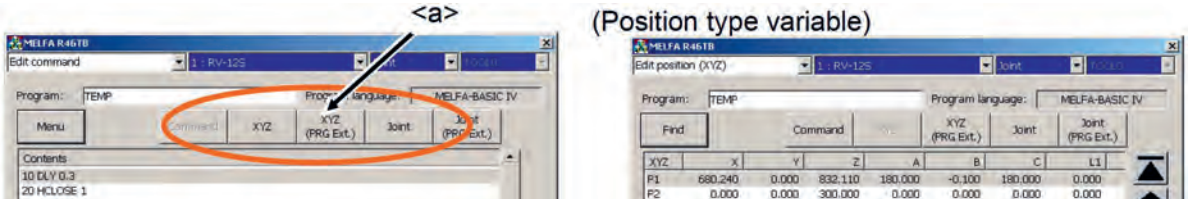




4.2.4.4 Posenvariablen mit dem Handbediengerät erstellen/editieren

Bearbeiten Sie die Roboter-Posen auf dem Bildschirm zur Posenbearbeitung.
Drücken Sie die [XYZ], [XYZ (Global)], [Joint] oder [Joint (Global)] Schaltfläche (→ <a>). Der Bildschirm zur Bearbeitung der Position wird angezeigt.
Dabei sind:

[XYZ]	Variablen mit kartesischer Posendefinition
[XYZ (Global)]	Variablen mit kartesischer Posendefinition (Systemvariable)
[Joint]	Variablen mit Gelenkwinkel-Pose
[Joint (Global)]	Variablen mit Gelenkwinkel-Pose (Systemvariable)

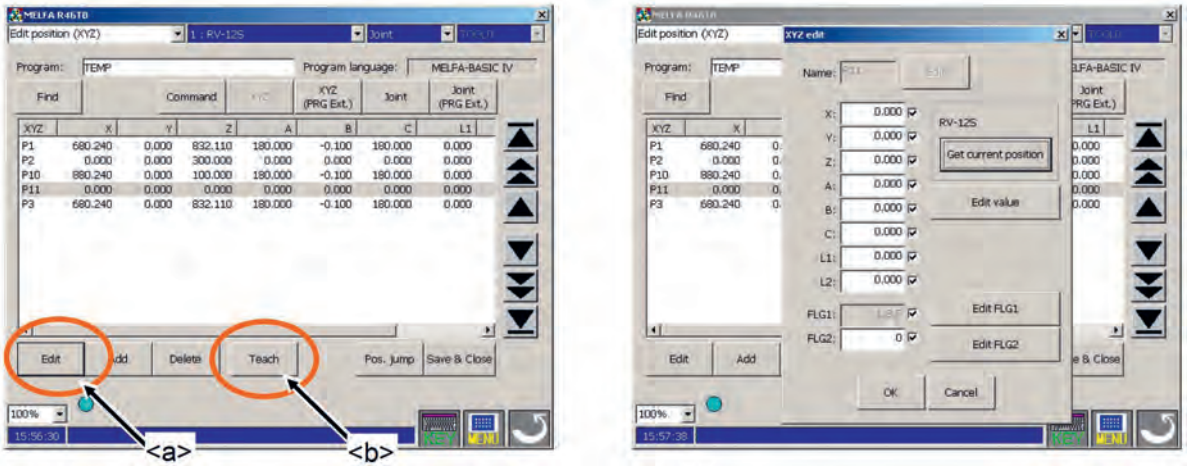


4.2.4.4.1 Registrierung der aktuellen Roboter-Pose

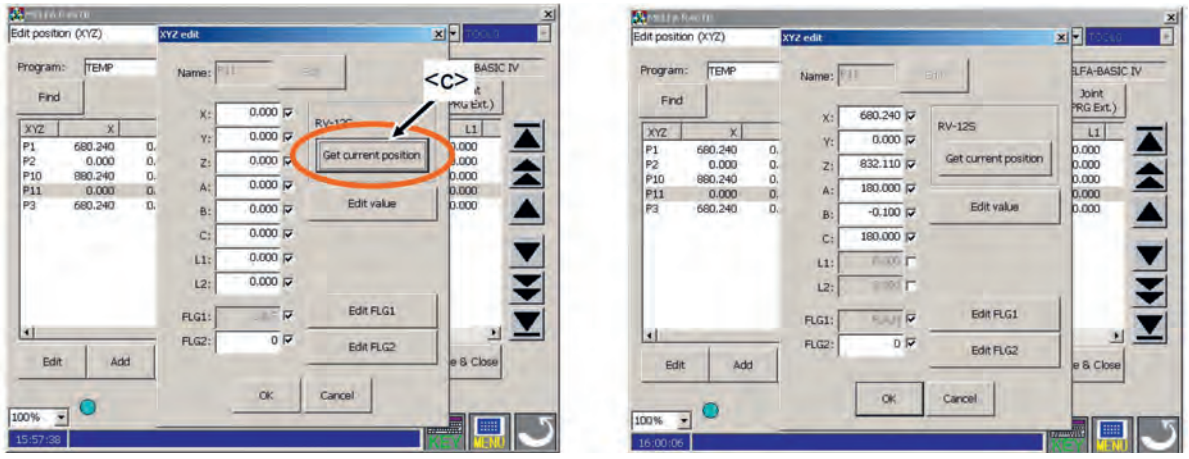
Das Verfahren zur Erfassung der aktuellen Pose des Roboters verläuft wie folgt:

1. Bewegen Sie den Roboter im Tippbetrieb in die einzulernende Pose.
2. Wählen Sie die einzulernende Variable aus und tippen Sie auf die Schaltfläche [Teach] (→ <b>).

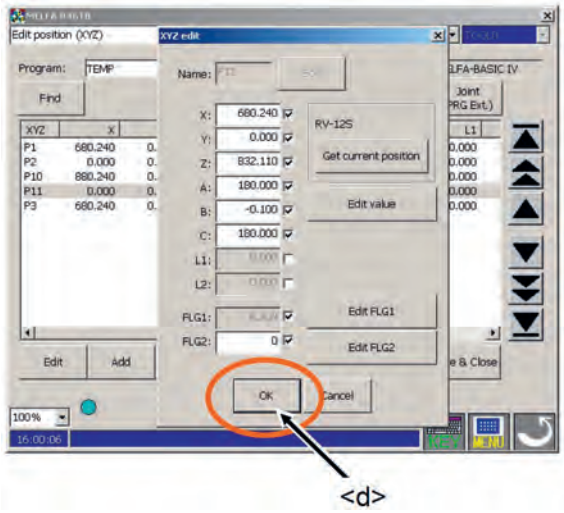
Die aktuelle Pose des Roboters kann dann eingelernt werden. Wenn Sie nach dem Bestätigen der aktuellen Posedaten Änderungen durchführen möchten, tippen Sie auf die Schaltfläche [Edit] (→ <a>).
Es wird der Bildschirm zur Bearbeitung der Posenwerte angezeigt.

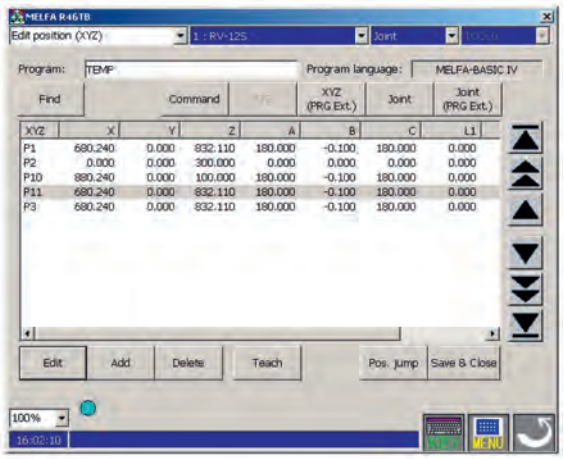


3. Wenn die Schaltfläche [Get current position] (→ <c>) auf dem Datenbearbeitungsbildschirm gedrückt wird, werden die aktuellen Koordinatenwerte des Roboters übernommen.



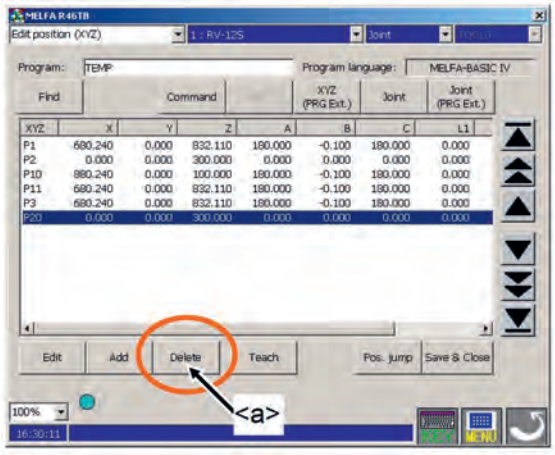
4. Tippen Sie auf die Schaltfläche [OK] (→ <d>) und registrieren Sie so die Posen.

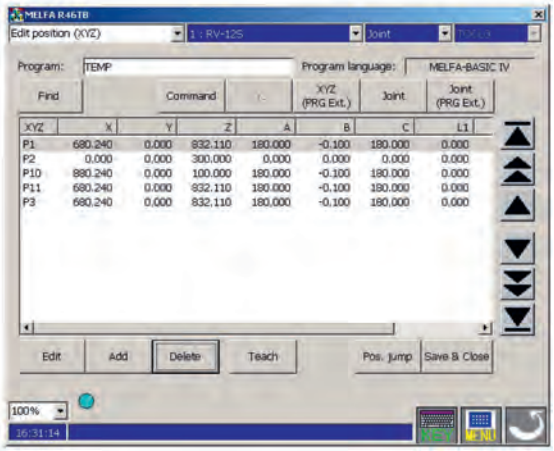




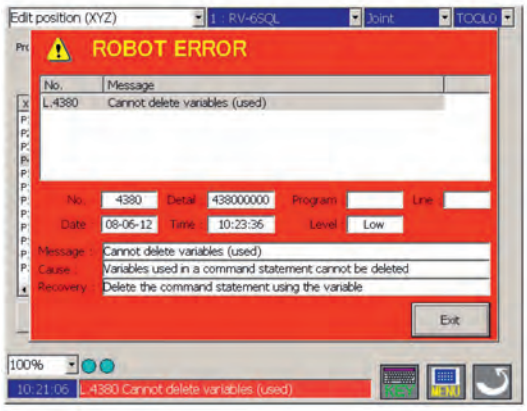
4.2.4.4.2 Löschen einer Posenvariablen

Nachfolgend wird das Löschen von Posendaten erläutert. Beachten Sie: Eine aktuell in einer Befehlszeile verwendete Pose kann nicht gelöscht werden.
Wählen Sie die zu löschende Variable und tippen Sie auf die Schaltfläche [Delete] (→ <a>). Die ausgewählte Posenvariable wird gelöscht.





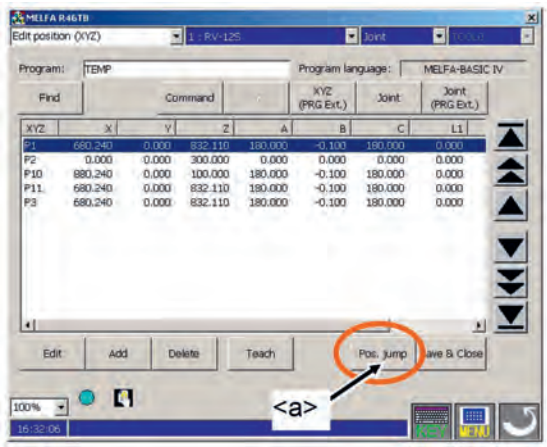
Die derzeit von der Befehlszeile verwendete Posenvariable kann nicht gelöscht werden. Es erscheint ein Fehler. Drücken Sie die Taste [RESET] (→ <b>), um den Fehler zu löschen.

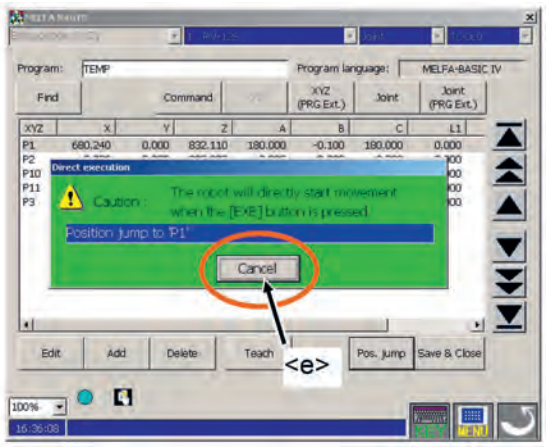


4.2.4.4.3 Überprüfen einer Posenvariablen (→ Posen-Sprung)

Der Roboter kann zu der eingelernten Pose gefahren und so die Pose überprüft werden. Die Bewegungsart des Roboters entspricht dabei dem aktuell eingestellten Tippbetrieb.
Gehen Sie wie folgt vor:

1. Bewegen Sie den Roboter zuvor im Tippbetrieb in eine sichere Position.
2. Wählen Sie die Posenvariable aus und tippen Sie auf die Schaltfläche [Pos. jump] (→ <a>). Der Bestätigungsbildschirm wird angezeigt. Tippen Sie auf die Taste [Abbrechen] (→ <e >), wenn Sie den Vorgang abbrechen möchten.





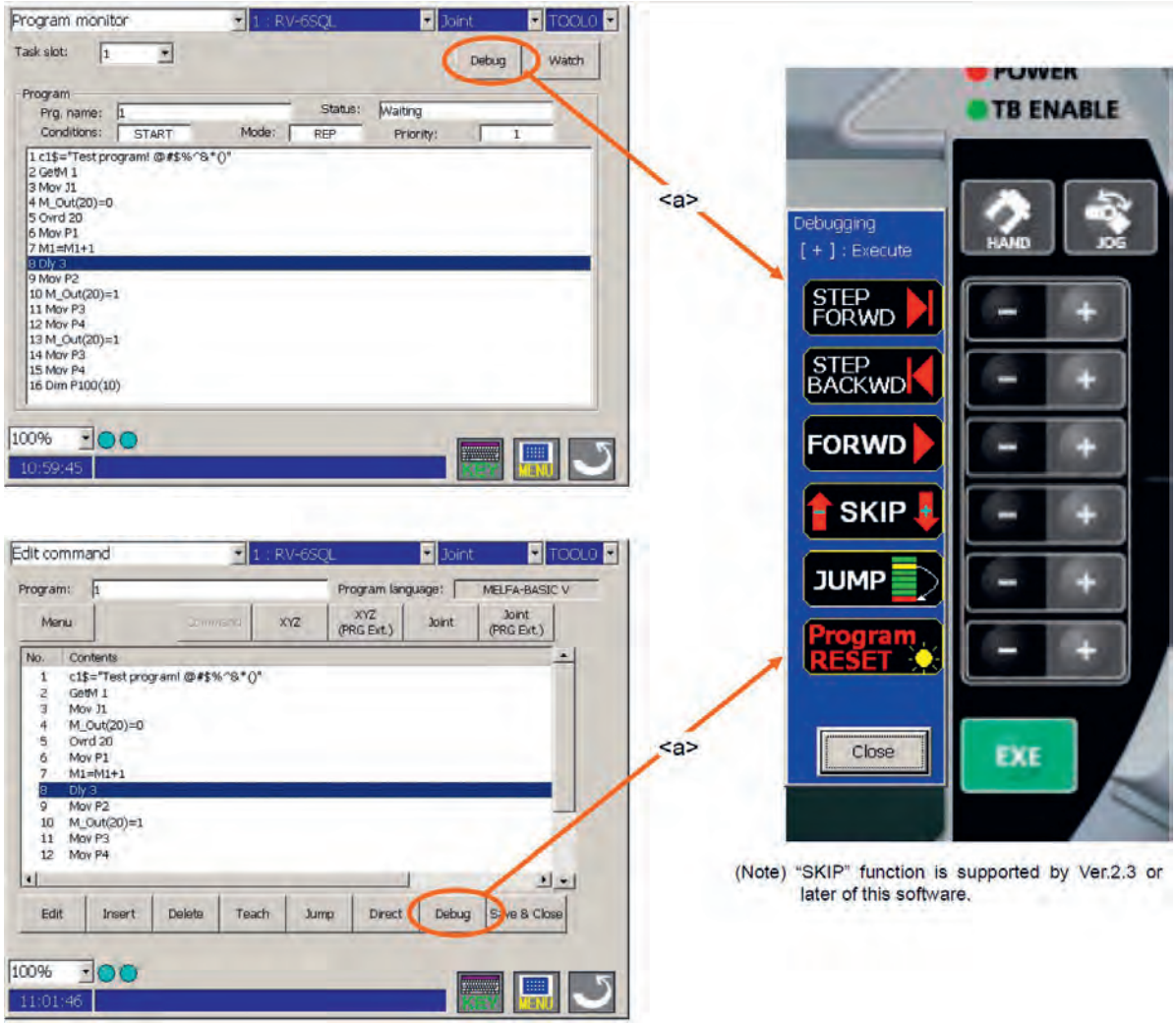
3. Im anderen Fall: Betätigen Sie den Freigabeschalter (→ <c>) und schalten Sie die Motoren im Roboter durch Drücken der [SERVO] (→ <b>). Wenn die Motoren eingeschaltet sind, leuchtet in der Taste [SERVO] die LED (grün) auf.
4. Nur wenn der Freigabeschalter gedrückt ist und die [EXE]-Taste (→ <d>) gedrückt wird, bewegt sich der Roboter. Während der Bewegung wechselt das Symbol unten links auf dem Handbediengerät zwischen und und wechselt bei Erreichen der Pose auf das Symbol .

4.2.4.5 Überprüfen eines erstellten Programms (Debugging)

Beim Debugging wird getestet, ob das erstellte Programm korrekt funktioniert und es können Fehler korrigiert werden. Dies kann mit Hilfe der Debugging-Funktion am Handbediengerät durchgeführt werden. Führen Sie das Debugging immer nach der Erstellung eines Roboterprogramms durch und überprüfen Sie so, ob das Programm fehlerfrei läuft.

4.2.4.5.1 Starten des Schrittbetriebs zum Debugging

Das Debugging-Fenster wird über das Fenster zur Bearbeitung von Roboterprogrammen oder zur Anzeige von Programmen aufgerufen. Tippen Sie in einem dieser Fenster auf die Schaltfläche "Debug" (→ <a>) und das Fenster "Debuggen" erscheint rechts auf dem Handbediengerät. Zur Bedienung drücken Sie die Taste "+" oder "-" in der entsprechenden Zeile.



(Note) "SKIP" function is supported by Ver.2.3 or later of this software.

4.2.4.5.2 Schrittbetrieb zum Debugging

Beim Schrittbetrieb wird das Programm Zeile für Zeile ausgeführt. Die Arbeitsgeschwindigkeit ist langsam und der Roboter hält nach jeder Zeile an, so dass die Programmzeile und die aktuelle Pose des Roboters bestätigt werden können.
Während der Bewegung des Roboters wechselt das Symbol unten links auf dem Handbediengerät zwischen und und wechselt bei Erreichen der Pose durch den Roboter auf das Symbol .
Drücken Sie den [TEACH]-Schalter (→ <a>) auf dem Handbediengerät und vergewissern Sie sich, dass der [TEACH]-Schalter und die [TB ENABLE]-Lampen leuchten. Führen Sie dann den Schrittbetrieb aus.

Führen Sie die folgenden Operationen aus, um das Programm zu durchlaufen und den Roboter zu bewegen. Während der gesamten Zeit muss der Freigabeschalter am Handbediengerät leicht ge¬drückt werden und die Motoren des Roboters müssen eingeschaltet sein.

Schritt vorwärts
Drücken Sie kontinuierlich die Taste "+", die dem "STEP FORWD" entspricht. Das Programm wird zeilenweise in Vorwärtsrichtung abgearbeitet. Wenn die "+"-Taste während eines Schrittes losge¬lassen wird, wird der Programmablauf in der betreffenden Zeile gestoppt.
Die Taste "-" hat hier keine Funktion.

Schritt rückwärts
Drücken Sie kontinuierlich die Taste "+", die dem "STEP BACKWD" entspricht. Das Programm wird zeilenweise in Rückwärtsrichtung abgearbeitet. Wenn die "+"-Taste während eines Schrittes los¬gelassen wird, wird der Programmablauf in der betreffenden Zeile gestoppt.
Die Funktion kann nur für die Interpolationsbefehle verwendet werden. Beachten Sie auch, dass nur bis zu 4 Zeilen zurückgegangen werden kann.
Die Taste "-" hat hier keine Funktion.

Kontinuierliche Ausführung
Drücken Sie kontinuierlich die Taste "+", die dem "FORWD" entspricht. Dadurch wird das Pro-gramm ab der aktuellen Zeile fortlaufend ausgeführt.
Die Taste "-" hat hier keine Funktion.

4.2.5 Übersicht der MELFA-BASIC-Befehle

In den folgenden Tabellen sind die MELFA-BASIC-Befehle in anwendungsspezifische Gruppen zusammengefasst. Die Reihenfolge wird durch die Verwendungshäufigkeit der Programmierbefehle bestimmt.

4.2.5.1 Befehle zur Bewegungssteuerung

Befehl		**Funktion**
Mov	(Move)	Bewegung mit Gelenk-Interpolation
Mvs	(Move S)	Bewegung mit Linear-Interpolation
Mvr	(Move R)	Kreis-Interpolation
Mvr2	(Move R2)	Kreis-Interpolation
Mvr3	(Move R3)	Kreis-Interpolation
Mvc	(Move C)	Kreis-Interpolation
Mva	(Move Arch)	Bewegung mit Bogen-Interpolation
MvSpl	(Move Spline)	Bewegung mit Spline-Interpolation
MvTune	(Move Tune)	Bewegungsmodus
Ovrd	(Override)	Übersteuerung
Spd	(Speed)	Geschwindigkeit festlegen
JOvrd	(J override)	Übersteuerung Gelenk-Interpolation
Cnt	(Continous)	Roboterbewegung überschleifen
Accel	(Accelerate)	Beschleunigung und Verzögerung einstellen
ColChk	(Col Check)	Kollisionsüberwachung aktivieren
CavChk On	(CavChk On)	Anti-Kollisions-Funktion aktivieren
Cmp Jnt	(Compliance Joint)	Achsenweichheit im Gelenkkoordinatensystem aktivieren
Cmp Pos	(Compliance Posture)	Achsenweichheit im XYZ-Koordinatensystem aktivieren
Cmp Tool	(Compliance Tool)	Achsenweichheit im Werkzeugkoordinatensystem aktivieren
Cmp Off	(Compliance OFF)	Achsenweichheit deaktivieren
CmpG	(Compliance Gain)	Achsenweichheit einstellen
Oadl	(Optimum Acceleration/ Deceleration)	Optimale Beschleunigung/Abbremsung
Mxt	(Move External)	Externe Steuerung
LoadSet	(Load set)	Hand- und Werkstückbedingung einstellen
Prec	(Precision)	Verfahrweggenauigkeit erhöhen
Torq	(Torque)	Drehmomentgrenze definieren
JRC	(Joint Roll Change)	Gelenkposition verändern
Fine	(Fine)	Feinpositionierung
Fine J	(Fine Joint)	Feinpositionierung bei Gelenk-Interpolation
Fine P	(Fine Pause)	Feinpositionierung über geradlinigen Abstand
Servo	(Servo)	Servo ein-/ausschalten
Wth	(With)	Anweisung hinzufügen
WthIf	(With If)	Anweisung hinzufügen, wenn ...
EMvs	(E Move S)	Linear-Interpolation entlang des Werkstück-Koordinatensystems
EMvc	(E Move C)	Kreis-Interpolation entlang des Werkstück-Koordinatensystems

Befehl		Funktion
EMvr	(E Move R)	Kreis-Interpolation entlang des Werkstück-Koordinatensystems
EMvr2	(E Move R2)	Kreis-Interpolation entlang des Werkstück-Koordinatensystems
EMvr3	(E Move R3)	Kreis-Interpolation entlang des Werkstück-Koordinatensystems

4.2.5.2 Befehle zur Programmsteuerung

Befehl		Funktion
Rem	(Remarks)	Kommentar
If Then Else EndIf	(If Then Else)	Bedingte Verzweigung
Select Case	(Select case)	Prozess ausführen
GoTo	(Go To)	Sprung zu einer Marke
GoSub	(Go Subroutine)	Sprung zu einem Unterprogramm
Reset Err	(Reset Error)	Fehler zurücksetzen
CallP	(Call P)	Programm aufrufen
FPrm	(FPRM)	Parameter definieren
Dly	(Delay)	Verzögerung einstellen
Hlt	(Halt)	Programmablauf stoppen
End	(End)	Programmende
On GoSub	(ON Go Subroutine)	Sprung zu einem Unterprogramm
On GoTo	(On Go To)	Programmverzweigung
For-Next	(For-next)	Programmschleife
While WEnd	(While End)	Programmschleife
Open	(Open)	Datei oder Kommunikationsleitung öffnen
Print #	(Print)	Daten übertragen
Input #	(Input)	Daten einlesen
Close	(Close)	Datei oder Kommunikationsleitung schließen
ColChk	(Col Check)	Kollisionsüberwachung aktivieren
On Com GoSub	(ON Communication Go Subroutine)	Sprung zu einem Unterprogramm
Com On	(Communication ON)	Kommunikations-Interrupt freigeben
Com Off	(Communication OFF)	Kommunikations-Interrupt sperren
Com Stop	(Communication Stop)	Kommunikations-Interrupt stoppen
HOpen/HClose	(Hand open/Hand close)	Handzustand festlegen
Error	(Error)	Fehler generieren
Skip	(Skip)	Sprung zum nächsten Programmschritt
Wait	(Wait)	Wartestatus definieren
Clr	(Clear)	Löschen

4.2.5.3 Definitionsbefehle

Befehl		**Funktion**
Dim	(Dim)	Dimension einer Feldvariablen definieren
Def Plt	(Define pallet)	Palette definieren
Plt	(Pallet)	Koordinaten für Palette berechnen
Def Act	(Define Act)	Interrupt-Prozess definieren
Act	(Act)	Interrupt freigeben/sperren
Def Arch	(Define Arch)	Bogen definieren
Def Jnt	(Define Joint)	Gelenkvariable definieren
Def Pos	(Define Position)	Positionsvariable definieren
Def Inte/Long/ Float/Double	(Define Integer/Long/ Float/ Double)	Numerische Variable definieren
Def Char	(Define Character)	Zeichenkettenvariable definieren
Def IO	(Define IO)	Ein-/Ausgangsvariable definieren
Def Fn	(Define function)	Funktion definieren
Title	(Title)	Programmtitel festlegen
Base	(Base)	Basis
Tool	(Tool)	Werkzeug-Konvertierungsdaten
SetCalFrm	(Set Calibration Frame)	Koordinatensysteme für eine Transformation festlegen

4.2.6 Ein Beispielprogramm

Die nachfolgenden Programme (1 x Hauptprogramm und 2 x Unterprogramm) zeigen beispielhaft die Realisierung einer Pick&Place-Bewegung mit dem Mitsubishi Assista. Es geht dabei um die Montage einer Leitungsgruppe mit zwei Steckern auf einem Reflektor der VW Tiguan Heckleuchte. Der gesamte Ablauf kann auch (dann aber unter Verwendung eines UR5e Roboters) unter folgendem Link betrachtet werden: https://www.youtube.com/watch?v=COJ5Dchc498
Es wird jeweils eine Leitungsgruppe mit Hilfe eines Greifers gegriffen und dann nacheinander zunächst der erste Stecker und dann der zweite Stecker auf dem Reflektor montiert. Über eine FOR-Schleife wird der Vorgang insgesamt 5-mal durchgeführt.

Zeile	Befehl	Funktion
Das Hauptprogramm:		
1	'Kabelmontage fuer Tiguan-Reflektor	Kommentarzeile
2	Dim Pose_Kabel(5)	Definition von 2 Posen-Arrays
3	Dim Pose_Stecker(2)	
4	Mvel_schnell = 500 '100mm/sec	Definition von Geschwindigkeitskonstanten
5	Mvel_langsam = 50 '20mm/sec	
6	'Home-Pose langsam anfahren	
7	JOvrd 20	
8	Mov Pose_Home	PTP zur Home-Pose fahren
9	'Greifer oeffnen	
10	HOpen 1	Beide Greifer öffnen
11	HOpen 2	
12	'Schleife ueber 5 Kabel in 5 Reflektoren	
13	For M1 = 1 To 5	For-Schleife
14	'Warte auf Start	
15	Wait M_In(1) = 1	
16	'Hole Kabel	
17	CallP "PICK", Pose_Kabel(M1), Mvel_schnell, Mvel_langsam	Unterprogrammaufruf mit Parameterübergabe
18	'Zwischen-Pose in der Mitte anfahren	
19	Spd Mvel_schnell	
20	Cnt 1	Roboterbewegung überschleifen
21	Mvs Pose_Zwischen	
22	'Setze Stecker 1	
23	CallP "PLACE", 1, Pose_Stecker(1), Mvel_schnell, Mvel_langsam	Unterprogrammaufruf mit Parameterübergabe
24	'Zwischen-Pose ueber Reflektor anfahren	
25	Spd Mvel_schnell	
26	Cnt 1	
27	Mvs Pose_Reflektor	
28	'Setze Stecker 2	
29	CallP "PLACE", 2, Pose_Stecker(2), Mvel_schnell, Mvel_langsam	Unterprogrammaufruf mit Parameterübergabe
30	'Zwischen-Pose in der Mitte anfahren	
31	Spd Mvel_schnell	
32	Cnt 1	

Zeile	Befehl	Funktion
33	Mvs Pose_Zwischen	
34	Next M1	Nächster Schleifendurchlauf
35	'Home-Pose langsam anfahren	
36	JOvrd 50	
37	Mov Pose_Home	Zur Home-Position fahren
Pose_Kabel(1)=(+500.00,+400.00,+225.00,+180.00,+0.00,-180.00)(7,0)		Posenliste
Pose_Kabel(2)=(+500.00,+350.00,+225.00,+180.00,+0.00,-180.00)(7,0)		
Pose_Kabel(3)=(+500.00,+300.00,+225.00,+180.00,+0.00,-180.00)(7,0)		
Pose_Kabel(4)=(+500.00,+250.00,+225.00,+180.00,+0.00,-180.00)(7,0)		
Pose_Kabel(5)=(+500.00,+200.00,+225.00,+180.00,+0.00,-180.00)(7,0)		
Pose_Stecker(1)=(+500.00,-200.00,+300.00,+180.00,+0.00,-180.00)(7,0)		
Pose_Stecker(2)=(+500.00,-250.00,+300.00,+180.00,+0.00,-180.00)(7,0)		
Pose_Home=(+300.00,+0.00,+600.00,+180.00,+0.00,-180.00)(7,0)		
Pose_Zwischen=(+300.00,+0.00,+500.00,+180.00,+0.00,-180.00)(7,0)		
Pose_Reflektor=(+500.00,-225.00,+400.00,+180.00,+0.00,-180.00)(7,0)		
Das Pick-Unterprogramm:		
1	'Aufnehmen eines Kabels	
2	FPrm Pose_Pick, Mvel_fast, Mvel_slow	Definition der übergebenen Parameter
3	Spd Mvel_fast	
4	Cnt 1	Vor-Pose überschliffen anfahren
5	Mvs Pose_Pick,-50 'Approach-Pose anfahren	
6	Spd Mvel_slow	
7	Cnt 0	Pose konkret (nicht überschliffen) anfahren
8	Mvs Pose_Pick 'Pick-Pose anfahren	
9	HClose 1	Beide Greifer schließen
10	HClose 2	
11	Dly 0.5	Wartezeit
12	Cnt 1	
13	Mvs ,-50	Wegfahrbewegung
14	End	
Pose_Pick=(+500.00,+200.00,+225.00,+180.00,+0.00,-180.00)(7,0)		Pose
Das Place-Unterprogramm:		
1	'Aufstecken eines Steckers	
2	FPrm Mnum_Greifer, Pose_Place, Mvel_fast, Mvel_slow	
3	Spd Mvel_fast	
4	Cnt 1	
5	Mvs Pose_Place,-50 'Approach-Pose anfahren	

Zeile	Befehl	Funktion
6	Spd Mvel_slow	
7	Cnt 0	
8	Mvs Pose_Place 'Place-Pose anfahren	
9	HOpen Mnum_Greifer 'Greifer oeffnen	
10	Dly 0.5	
11	Cnt 1	
12	Mvs ,-50 'Depart-Pose anfahren	
13	End	
Pose_Place=(+500.00,-250.00,+300.00,+180.00,+0.00,-180.00)(7,0)		Pose

4.2.7 Quellen und weiterführende Informationen

Die Bilder in diesem Dokument wurden entweder im Labor Handhabungstechnik und Robotik der Hochschule Osnabrück selbst erstellt oder sind aus Handbüchern des Herstellers entnommen worden. Grundlage der Informationen aus diesem Dokument, sind die Handbücher zum Mitsubishi RV-5AS-D. Es stellt somit eine Zusammenfassung für den schulischen Betrieb da. Weiterführende Informationen sind den nachfolgenden Handbüchern zu entnehmen, die hier gleichermaßen als Quellenangabe dienen.

Handbuchname	Beschreibung	Dokument-Nr.
Collaborative Robot Safety Manual	Um die Sicherheit der Roboterbenutzer zu gewährleisten, enthält dieses Handbuch Informationen über allgemeine Vorsichtsmaßnahmen und Sicherheitsmaßnahmen, die bei der Handhabung des Roboters oder der Entwicklung und Konstruktion von Robotersystemen Systeme. Lesen Sie zuerst dieses Handbuch.	BFP-A3733
Hello ASSISTA Quick Set-up Guide	Beschreibt die Vorgehensweise beim Auspacken, der Installation, der Programmierung mit RT VisualBox und Betrieb des Roboters.	BFP-A3715
Standard Specifications	Enthält Informationen über Standardproduktspezifikationen, Optionen und Wartungsteile. Sie enthält außerdem Informationen über Sicherheit und technische Vorsichtsmaßnahmen bei der Einführung des Roboters in einer neuen Umgebung.	BFP-A3727
Robot Arm Setup and Maintenance	Erläutert die Anforderungen vor dem Betrieb des Roboters (Auspacken, Transport, Installation und Betriebskontrollen) und wie die Wartung und Inspektion durchgeführt werden.	BFP-A3729
Controller Setup and Maintenance	Erläutert die Schritte, die vor der Verwendung der Robotersteuerung durchgeführt werden müssen (Auspacken, Transport und Installation). Es enthält auch Informationen zur Wartung und Inspektion.	BFP-A3731
Collaborative Robot: Detailed explanations of functions and operations	Bietet Informationen über Funktionen, die speziell für kollaborative Roboter gelten.	BFP-A3735
Detailed explanations of functions and operations	Enthält Informationen über Funktionen und Betriebsmethoden, über die Verwendung von MELFA-BASIC Befehlen in Programmen verwendet werden, wie man externe Ein-/Ausgabegeräte anschließt und wie man Parameter einstellt. Die FR-Serie wird als Beispiel für die Erläuterung verwendet. Informationen zu nicht unterstützten Funktionen finden Sie in der separaten Betriebsanleitung "Collaborative Robot: Detailed explanations of functions and operations"	BFP-A3478
Troubleshooting	Informiert über die Ursachen und Lösungen von Fehlern, die beim Betrieb des Roboters auftreten können.	BFP-A3480
Tracking Function	Enthält Informationen zu den Spezifikationen, Funktionen und der Verwendung der Förderbandverfolgungsfunktion.	BFP-A3520
GOT Direct Connection Extended Function	Beschreibt die Datenkonfiguration des Speichers zwischen dem GOT und dem Roboter, die Überwachung und die Betriebsverfahren.	BFP-A3546
Ethernet Function	Erklärt, wie man mit einem Computer über Ethernet unter Verwendung von TCP/IP-Protokollen kommuniziert.	BFP-A3379

4.3 Franka Panda

Marvin Becker, Leibniz Universität Hannover

Inhalt

4.3.1 Einleitung

Das Franka Control Interface (FCI) ermöglicht eine schnelle und direkte Low-Level-Verbindung mit dem Franka Emika Robotern und dessen Greifern. Das FCI liefert den aktuellen Status des Roboters und ermöglicht seine direkte Steuerung mit einem externen, über Ethernet angeschlossenen Workstation-PC. Durch die Verwendung von *libfranka*, einer Open-Source-C++-Schnittstelle, können Echtzeit-Steuerwerte mit 1 kHz über verschiedene Schnittstellen gesendet werden. Darüber hinaus verbindet *franka_ros* die Franka Emika Forschungsroboter mit dem gesamten ROS-Ökosystem. Es integriert *libfranka* in ROS Control.
Da die direkte Steuerung der Franka Roboter über das FCI recht komplex ist und die Nutzung für typische Robotik Anwenderaufgaben (z.B. Pick-Place Aufgaben) einen erheblichen Mehraufwand erfordert, wurde im Rahmen des Kompetenzzentrums das RF Robot Control Framework entwickelt. Dabei handelt es sich um ein Programmier-Framework, welches *franka_ros* und *ros_control* integriert und eine grafische Benutzeroberfläche bereitstellt, um zusätzliche nützliche Funktionen anzubieten, wie z. B. das Einlernen von Aufgabenpunkten und Gelenkkonfigurationen, einfaches Starten von Programmen, Zurücksetzen von Fehlern und viele weitere. Die Komponenten sind, wie bei ROS üblich, für die Ausführung auf mehreren Maschinen ausgelegt. Bei diesem Setup wird erwartet, dass *franka_ros* und *ros_control* auf einem Steuerungs-PC mit Echtzeit-Kernel laufen, der direkt mit dem Franka-Master-Controller verbunden ist. Das Framework ist so konzipiert, dass es die Entwicklung entweder direkt auf dem Steuerrechner oder auf separaten Entwicklerrechnern erlaubt, die über Ethernet mit dem Steuerrechner verbunden sind und keine installierte Echtzeit haben müssen.

Das Framework unterstützt derzeit drei Arten von Bewegungen, die als Klassen definiert sind
- MoveCartesian (RFLIN): Erzeugt lineare Bewegungen von einem kartesischen Punkt zu einem anderen.
- MoveJoint (RFPTP): Erzeugt Bewegungen direkt von einem Punkt zu einem anderen im Joint-Raum
- MoveToContact (RFMTC): Erzeugt lineare Bewegungen in eine bestimmte Richtung, bis eine entgegengesetzte Kraft überschritten wird.

Innerhalb dieser Klassen gibt es eine Reihe von Funktionen, die mit der gewünschten Bewegungsart ausgeführt werden können. So kann sich der Roboter z.B. zu einem gewünschten Punkt bewegen, oder mit einer Relativbewegung in eine bestimmte Richtung von einem Punkt aus. Mehr über die Klassen und Funktionen finden Sie in der Doxygen-Dokumentation.
Dieses Framework unterstützt derzeit den Franka Emika Panda Roboter, eine Simulation mit dem CoppeliaSim Simulator und bietet begrenzte Unterstützung für Universal Robots. Beachten Sie jedoch, dass sich die folgende Dokumentation auf die Verwendung mit einem Franka Emika-Roboter konzentriert.

4.3.2 Übersicht

Das Framework wird in einem Github Repository bereitgestellt und ist unter folgendem Link erreichbar:
https://gitlab.com/roboterfabrik1/rf_robot_framework
Um die Installation zu erleichtern wird das Framework in einem Docker Container bereitgestellt. Dabei handelt es sich um eine isolierte Umgebung, die eine Anwendung und alle ihre Abhängigkeiten enthält, die für deren Ausführung benötigt werden. Container sind somit ähnlich wie virtuelle Maschinen, aber im Gegensatz zu virtuellen Maschinen teilen sie sich den Kernel des Host-Betriebssystems, was sie effizienter und ressourcenschonender macht.

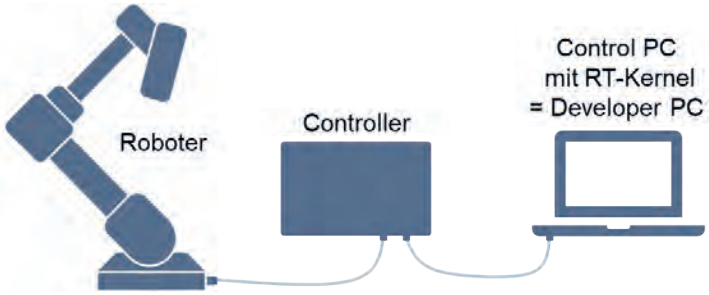
Das Repository beinhaltet:
- ein kompiliertes Docker Image mit passenden Abhängigkeiten für Franka Roboter mit der Desk Version 4.2.1. Es beinhaltet: Ubuntu 20.04, ROS Noetic, libfranka Version: 0.9.0 (Version vom 28.03.2022), franka_ros Version: 0.9.0 (Version vom 24.08.2022), Simulationsumgebung: CoppeliaSim Education Version 4.3.0 Rev 12, Universal Robot ROS Driver Version von https://github.com/UniversalRobots/Universal_Robots_ROS_Driver (Version vom 02.09.2022)
- das dazugehörige Dockerfile um das Image selbst zu kompilieren (→ so können eigene Programme hinzugefügt werden oder falls notwendig die Abhängigkeiten für neue/alte Franka Desk-Versionen angepasst werden)
- Skripte um das Framework zu installieren inklusive: aller benötigten Abhängigkeiten, Docker, VS Code mit Erweiterungen

Verbindungsmöglichkeiten mit dem Franka Roboter

Leider ist für die Nutzung der FCI Schnittstelle und damit auch für die Nutzung des RF Robot Frameworks die Installation eines Real-Time Kernels nötig. Die Installation eines Real-Time Kernels ist recht aufwendig und dauert vergleichsweise lang (je nach PC ca. 1-2 Stunden, siehe auch Kapitel „Installation Real-Time Kernel“). Deshalb wurde das RF Framework so konzipiert, dass es mit verschiedene Verbindungsmöglichkeiten mit dem Franka Roboter zur Verfügung stellt.

Folgende Verbindungsmöglichkeiten gibt es:

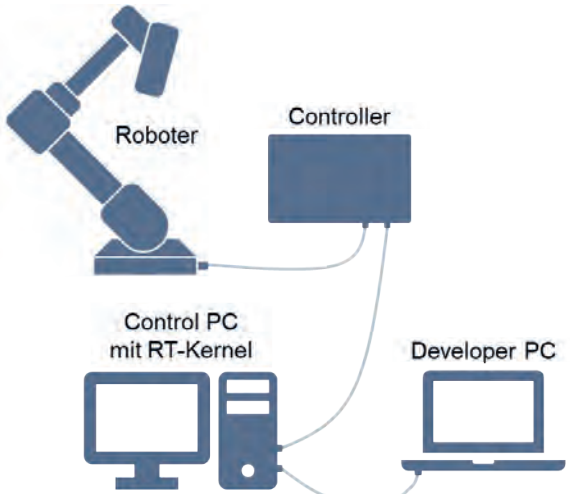
Setup 1



Bei diesem Setup wird nur ein einzelner PC benötigt, welcher gleichzeitig zur Programmierung (= Developer PC) und Ansteuerung (= Control PC) des Franka Roboters genutzt wird. Der PC, welcher einen installierten Real-Time Kernel benötigt, ist direkt mit dem Mastercontroller des Franka Roboters verbunden.

Vorteile:
- Setup einfacher
- Nur ein PC benötigt
- Control PC benötigt nur einen Netzwerkanschluss

Setup 2



Beim zweiten Setup wird ein Control PC mit installiertem Real-Time Kernel direkt mit dem Mastercontroller des Franka Roboters verbunden (via Ethernet). Der Programmcode und das Starten von Programmen erfolgt jedoch auf einem (oder mehreren) Developer PCs, welche wiederum mit dem Control PC verbunden sind (via Ethernet).ad asd d

Vorteile:
- Developer PCs benötigen keinen Realtime Kernel → einfache Installation
- Einfacher Wechsel des Developer PCs möglich
- Anschluss mehrerer Developer PCs möglich und somit auch gut parallel von mehreren Gruppen nutzbar

Hinweise zur Auswahl des Control PCs

- Die benötigten Linux Realtime Kernel unterstützen keine NVIDIA Grafikkarten. Installieren Sie also bitte den Realtime Kernel nicht auf einem System mit NVIDIA Grafikkarte. Die Installation könnte das System komplett unbrauchbar machen und eine Wiederherstellung ist recht aufwendig.
- Auf dem PC muss das Betriebssystem Ubuntu laufen, optimalerweise in der Version 20.04. Es existiert zwar schon die Version 22.04, allerdings sind hier erst wenige Packages getestet und optimiert worden, so dass wir insbesondere im Hinblick auf die zukünftige Nutzung mit ROS Ubuntu 20.04 empfehlen. Ubuntu 18.04 sollte auch noch funktionieren, wurde von uns aber nicht getestet.
- Generell ist eine gute Netzwerkverbindung zwischen PC und Roboter nötig. Deshalb ist eine gute Netzwerkkarte von Vorteil. Lange Ethernetkabel oder Verbindungen über Switches o.ä. sollten auch vermieden werden.
- Falls ein zusätzlicher Developer PC genutzt werden soll, ist es von Vorteil, wenn der Control PC mehrere Netzwerkanschlüsse hat (aber nicht zwingend erforderlich). Diese können bei normalen Desktop PCs einfach und kostengünstig nachgerüstet werden. Alternativ kann auch ein USB-zu-Ethernet- Adapter genutzt werden.

Voraussetzung - Real-Time Kernel Installation

Für die Nutzung von FCI wird ein Real-Time Kernel benötigt damit die versendeten Daten und Befehle zum Roboter priorisiert behandelt werden. Dadurch ist eine hohe Updaterate des Roboters möglich. Ein Real-Time Kernel wird nur auf dem Control PC benötigt. Die Installation auf einem Developer PC ist nicht notwendig.
Die Installation und Konfiguration des Realtime Kernel ist recht aufwendig und wird etwas Zeit in Anspruch nehmen. Die Installation erfolgt durch Eingabe der nachfolgenden Befehle in ein Terminal.
Die Installationsanleitung ist größtenteils von hier übernommen: https://frankaemika.github.io/docs/

Achtung: Nvidia Treiber werden von dem Real-Time Kernel nicht unterstützt.!
→ Nicht auf einem Rechner mit Nvidia Grafikkarte installieren!

Verwendung des Terminal unter Ubuntu

In Linux bzw. Ubuntu kann das Terminal für praktisch alles genutzt werden. Wir werden damit die erforderlichen Pakete installieren, später Programmcode kompilieren und ausführen.

Kurze Übersicht der Befehle im Terminal:
- Terminal öffnen: Im Hauptmenü nach „terminal“ suchen oder **Strg + Alt + T**
- Im Terminal ausgeführten Befehl/Programm abbrechen: **Strg + C**
- Copy/Kopieren: **Strg + Shift + C**
- Paste/Einfügen: **Strg + Shift + V**
- In bestimmtes Verzeichnis wechseln: **cd** z. B. cd /directory/src/
- In Verzeichnis darüber wechseln: **cd ..** (zwei Punkte)
- Befehl ausführen: **Enter**
- Bei erstmaliger Eingabe von **sudo** (ähnlich zu „als Administrator ausführen“ in Windows) muss das Benutzerpasswort eingeben werden
- Bei (Teil-)Eingabe eines Befehls und dann Drücken von **Tab** wird dieser automatisch vervollständigt

Installationsanleitung Real-Time Kernel

- Öffne ein Terminal: Über die Suche „terminal“ oder per Tastenkombination **Strg + Alt + T**
- Danach folgende Befehle (graue Kästen) markieren, kopieren (**Strg + C**), im Terminal-Fenster einfügen (**Strg + Shift + V**) und mit Enter ausführen:
- Zuerst installieren wir ein paar Pakete, die zur Installation des Kernels benötigt werden:

```
sudo apt-get install build-essential bc curl ca-certificates gnupg2 libssl-dev
lsb-release libelf-dev bison flex dwarves zstd libncurses-dev
```

- Im nächsten Schritt müssen die zu installierenden Dateien heruntergeladen werden. Dazu erstellen wir uns zunächst einen Ordner im Download- Verzeichnis und wechseln in das neue Verzeichnis:

```
mkdir ~/Downloads/kernel_files
cd ~/Downloads/kernel_files/
```

- Jetzt müssen wir uns für eine Kernel Version entscheiden. Um die aktuelle genutzte Kernel Version herauszufinden kann man folgenden Befehl nutzen:

```
uname -r
```

- Es wird empfohlen eine Version zu nutzen, die der aktuellen Kernel Version am nächsten ist. Eine Liste aller verfügbaren Realtime Kernel Patches ist unter folgendem Link zu finden: https://www.kernel.org/pub/linux/kernel/projects/rt/
- Es wird empfohlen eine Version zu nutzen, die der aktuellen Kernel Version am nächsten ist. Eine Liste aller verfügbaren Realtime Kernel Patches ist unter folgendem Link zu finden: https://www.kernel.org/pub/linux/kernel/projects/rt/ sdfsd

Beispiel:

```
uname -r
```

liefert: 5.15.0-46-generic → Auf https://www.kernel.org/pub/linux/kernel/projects/rt/ finden wir einen Realtime Kernel Patch zu der Version 5.15.55 und wählen deshalb diese Version.

- Jetzt können die Dateien mit curl heruntergeladen werden

```
curl -SLO https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.15.55.tar.xz
curl -SLO https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.15.55.tar.sign
curl -SLO https://www.kernel.org/pub/linux/kernel/projects/rt/5.15/older/patch-5.15.55-rt48.patch.xz
curl -SLO https://www.kernel.org/pub/linux/kernel/projects/rt/5.15/older/patch-5.15.55-rt48.patch.sign
```

- Für eine andere Version müssen in den vorherigen vier Befehlen einfach nur die rot markierten Nummern ausgetauscht werden.
- Die heruntergeladenen Dateien liegen als tar-Datei vor. Wir entpacken diese mit:

```
xz -d *.xz
tar xf linux-*.tar
```

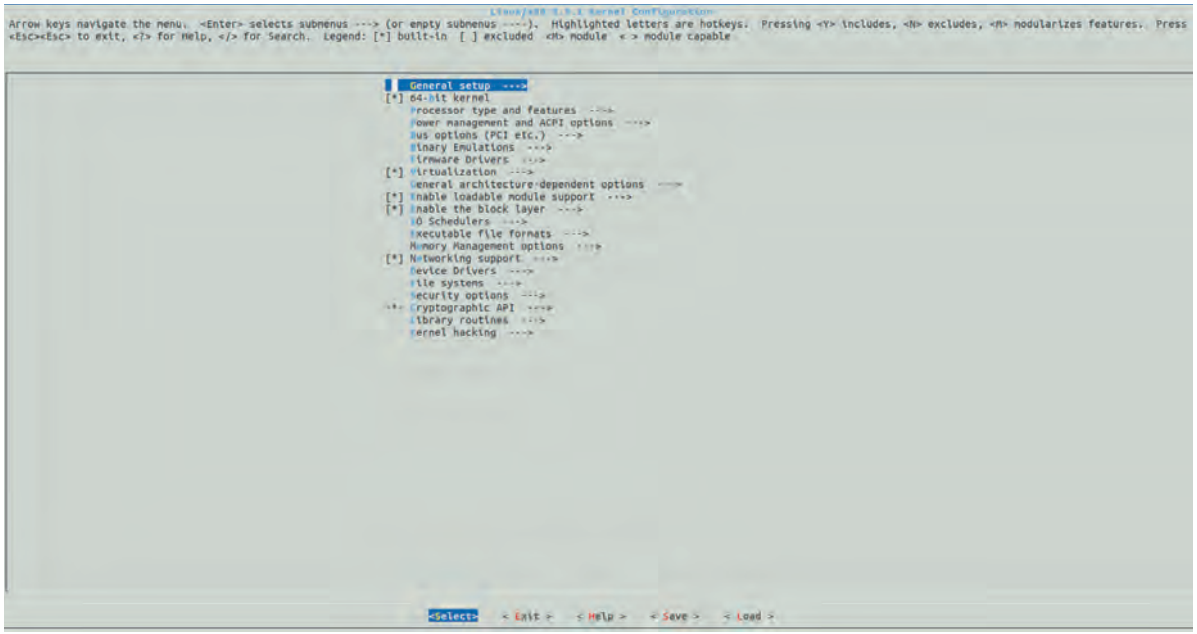
- Und wechseln dann in das entpackte Verzeichnis:

```
cd linux-*/
```

- Danach muss der heruntergeladene Kernel konfiguriert werden:

```
patch -p1 < ../patch-*.patch
cp -v /boot/config-$(uname -r) .config
make olddefconfig
make menuconfig
```

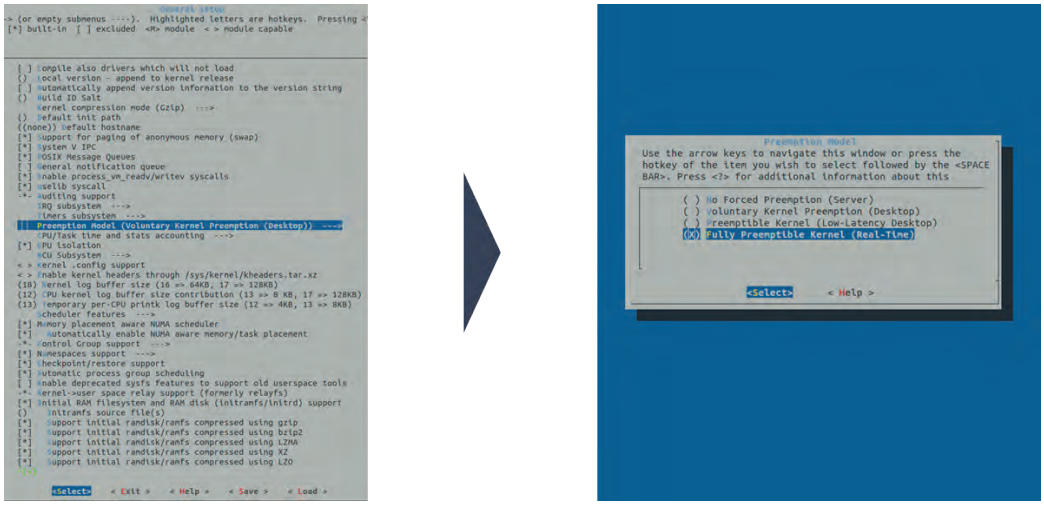
Mit dem letzten Befehl öffnet sich ein grafisches Menü, in dem zwei Einstellungen vorgenommen werden müssen. Navigiert wird mit den Pfeiltasten. Bestätigt mit **Enter** und zurück mit **Esc + Esc** (analog zum Doppelklick mit der Maus).



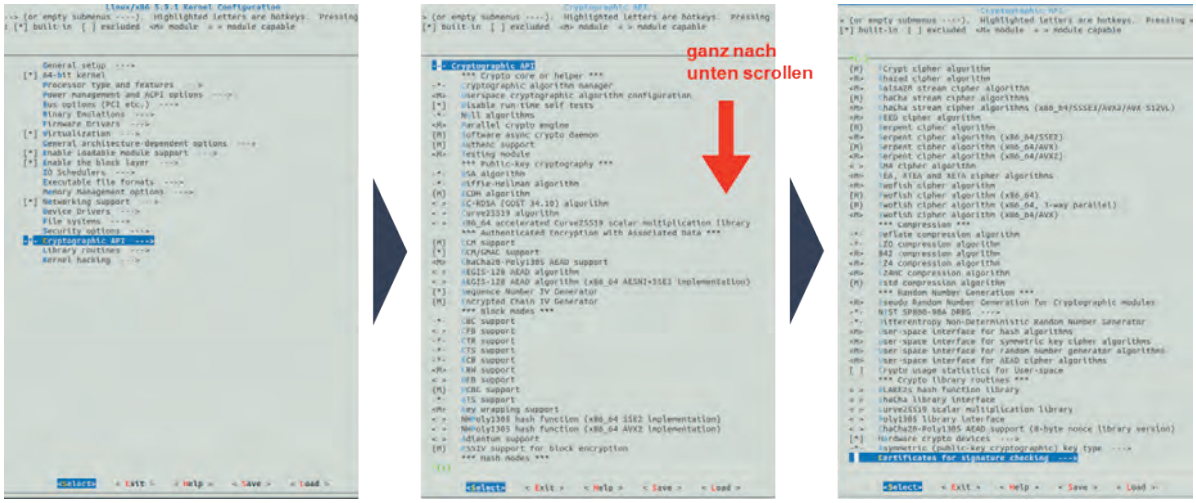
Die folgenden Einstellungen müssen angepasst werden:

- General setup → Preemption Model

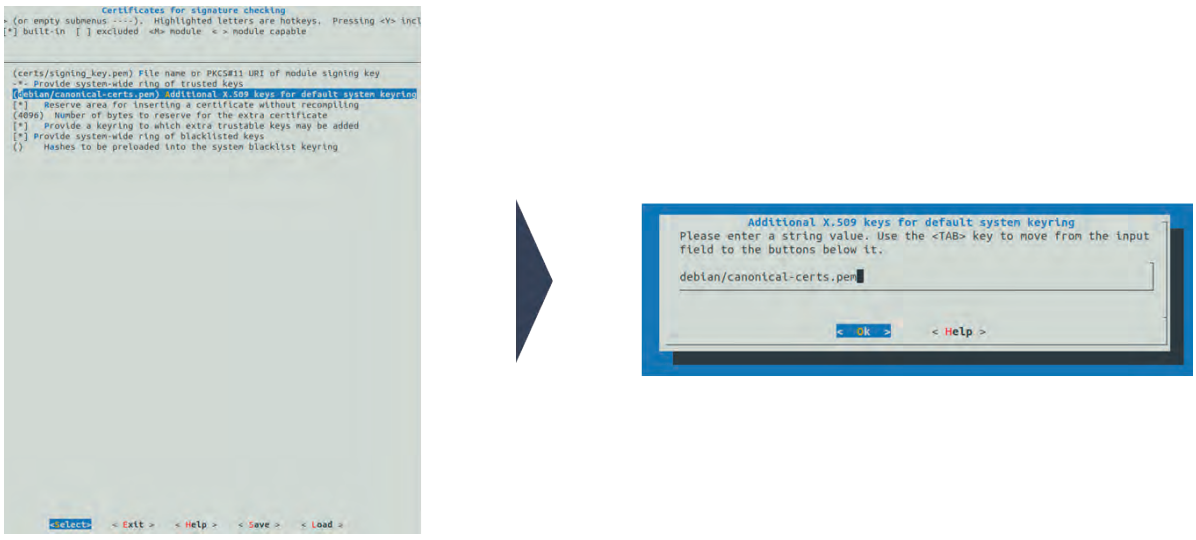
Hier auswählen: Fully Preemptible Kernel (Realtime)



- Cryptographic API → Certificates for signature checking



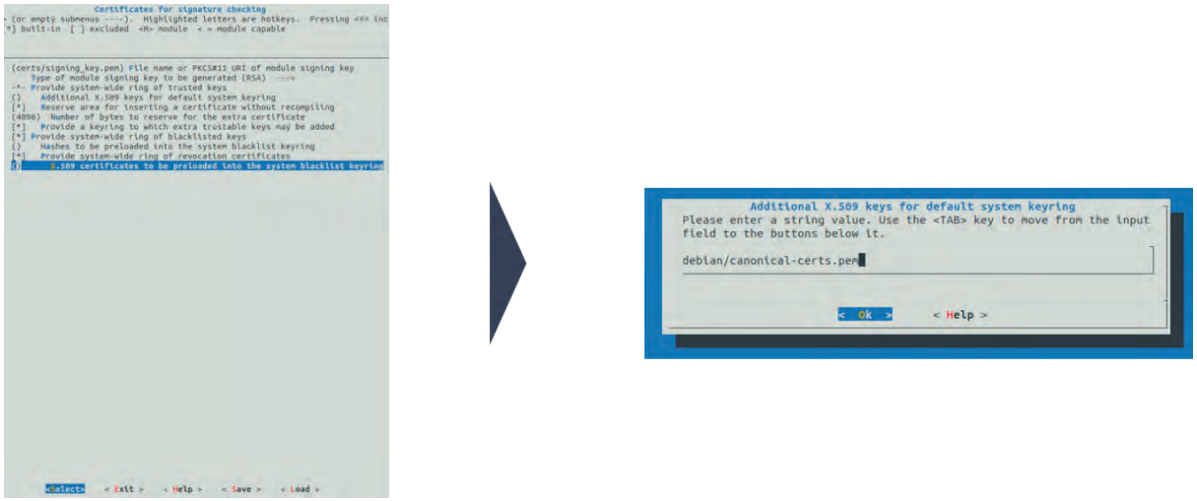
- Cryptographic API → Certificates for signature checking (ganz nach unten scrollen) → Provide system-wide ring of trusted keys → Additional X.509 keys for default system keyring



Wenn der Punkt ausgewählt wird, erscheint eine Texteingabezeile. Der dort stehende Text muss gelöscht werden. Danach bestätigen.

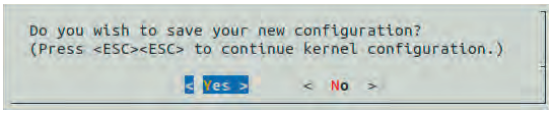
Optional: Je nach Kernelversion gibt es hier noch eine zusätzliche Einstellung:

- Cryptographic API → Certificates for signature checking (**ganz nach unten scrollen**) → Provide system-wide ring of revocation certificates → X.509 certificates to be preloaded into the system blacklist keyring



Wenn der Punkt ausgewählt wird, erscheint eine Texteingabezeile. Der dort stehende Text muss **gelöscht** werden. Danach bestätigen.

- Wurden alle Optionen gesetzt, muss das Menü mit doppelten Drücken von **Esc** beendet werden. Die Konfiguration muss anschließend gespeichert werden:



- Im Anschluss muss der Kernel kompiliert werden. Der Prozess kann eine längere Zeit dauern (**bis zu ca. einer Stunde!**). Solange im Terminal eine Ausgabe durchläuft ist alles in Ordnung. Es muss dann lediglich gewartet werden.
- ACHTUNG: Der Rechner darf dabei nicht in den Energiesparmodus wechseln und sollte auch nicht den Bildschirm ausschalten. Deshalb am besten vorher folgende Einstellungen in Ubuntu vornehmen:
- Settings → Power → Power Saving → Blank Screen: Never
- Settings → Power → Suspend & Power Button → Automatic Suspend: Off
- Kompilierung des Kernels:

```
make -j20
```

- Nach dem Kompilieren wird der kompilierte Kernel installiert:

```
sudo make modules_install -j20
sudo make install -j20
```

- Die Einstellungen des Bootloaders müssen danach geupdated werden, damit der neue Kernel beim Neustart auch angezeigt wird:

```
sudo update-grub
```

- Jetzt muss das System neu gestartet werden.

```
sudo reboot
```

- Nach dem Neustart sollte im Bootmenü (GRUB) unter „Advanced Options for Ubuntu“ die installierte Kernelversion auswählbar sein. In unserem Beispiel wäre das: Ubuntu, mit Linux 5.15.55-rt48
- Nach Auswahl wird der Kernel gestartet. Je nach Einstellung und System kann dies aber auch automatisch erfolgen und GRUB wird nicht benötigt.
- Nachdem das System gebootet ist, kann geprüft werden, ob die richtige Kernel Version installiert ist. Dazu im Terminal eingeben:

```
uname –a
```

- Der angezeigte Text sollte den String **PREEMPT RT** und die Versions- Nummer **5.15.55** enthalten.

Beispielausgabe:
Linux irtpc058 5.15.55-rt48 #1 SMP PREEMPT_RT Thu Sep 1 08:07:49 CEST 2022 x86_64 x86_64 x86_64 GNU/Linux

- Falls die Ausgabe **PREEMPT RT** enthält, können die nächste 4 Schritte übersprungen werden

- Falls der neue Realtime Kernel nicht gestartet wird und beim Starten des PCs das Grub Bootmenu nicht erscheint, muss folgender Befehl ausgeführt werden:

```
sudo gedit /etc/default/grub
```

- Im sich öffnenden Fenster müssen die folgenden Variablen geändert/hinzugefügt werden:

```
GRUB_DEFAULT=saved
GRUB_SAVEDEFAULT=true
GRUB_TIMEOUT_STYLE=menu
GRUB_TIMEOUT=10
```

- Anschließend folgenden Befehl ausführen:

```
sudo update-grub
```

- PC neustarten und im GRUB Bootmenu den Kernel ändern wie oben beschrieben.
- Abschließend müssen dem eigenen Benutzer noch die Rechte für die Verwendung der Realtime Berechtigungen gegeben werden:

```
sudo addgroup realtime
sudo usermod -a -G realtime $(whoami)
```

- Und in der Datei **/etc/security/limits.conf** Parameter gesetzt werden:
- Datei im Texteditor öffnen mit:

```
sudo gedit /etc/security/limits.conf
```

- Am Ende der Datei dann Folgendes anfügen:

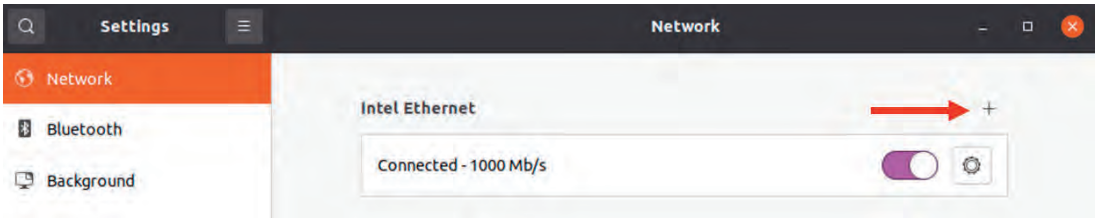
```
@realtime soft rtprio 99
@realtime soft priority 99
@realtime soft memlock 102400
@realtime hard rtprio 99
@realtime hard priority 99
@realtime hard memlock 102400
```

Damit ist die Installation des Real-Time Kernels abgeschlossen.

4.3.3 RF Robot Control Framework - Installation

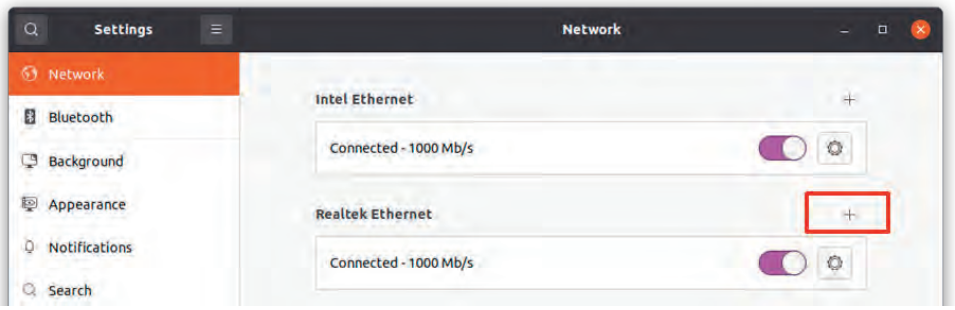
4.3.3.1 Vorbereitung – Einrichten der Netzwerkverbindung

Die Einrichtung der Netzwerkverbindung kann gegebenenfalls übersprungen werden. Es ist jedoch wichtig, dass dem Franka Roboter eine statische IP Adresse zugewiesen wird (standardmäßig wird die IP Adresse über DHCP bezogen).
Um die (Erst-)Konfigurationsschritte zu öffnen, muss ein Bediengerät (PC mit Firefox o. Chromium) über ein Ethernet-Kabel an den X5-Anschluss an der Basis des Franka Roboters (nicht am Controller!) angeschlossen werden. Die Ethernet-Verbindung des Bediengeräts muss so konfiguriert sein, dass die IP-Adresse automatisch über DHCP bezogen wird (**Settings → Network**).



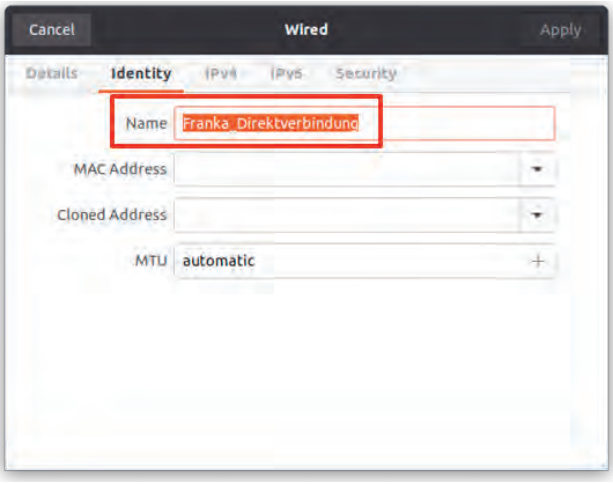
Durch Klicken auf das + wird ein neues Profil angelegt (Standardeinstellungen beibehalten, d. h. **IPv4 Method** auf **Automatic (DHCP)** eingestellt lassen).
Sobald der Roboter eingeschaltet ist, wird dem Schnittstellengerät eine automatische IP-Adresse zugewiesen. Nun kann die URL **robot.franka.de** in die Adresszeile eines Webbrowsers (Firefox o. Chromium) eingeben und mit Enter bestätigt werden. Als nächstes kann die Konfiguration des Roboters vorgenommen werden:

- Falls die Erstkonfiguration noch nicht stattgefunden hat, sollte dafür das Handbuch des Roboters gelesen werden.
- Andernfalls melden wir uns mit **Username** und **Password** an, um die Oberfläche von Desk zu öffnen.
- Nun navigiere zu den **Settings** des Roboters.
- Klicke in der linken Leiste auf **Network**.
- Unter Shop Floor network können die folgenden Einstellungen vorgenommen werden:
  - statische IP-Adresse des Roboters (hier: 192.168.1.11)
  - Netzmaske (hier: 255.255.255.0)
- Klicke zum Abschluss auf Apply
- Schließlich kann der PC über das Ethernet-Kabel mit dem Ethernet-Anschluss des Master Controllers verbunden werden
- Weitere Informationen:
  https://frankaemika.github.io/docs/getting_started.html#control-network-configuration
- Gehe zu **Settings → Network** und klicke auf das **+** rechts von Ethernet, um eine neue Ethernet-Verbindung hinzuzufügen

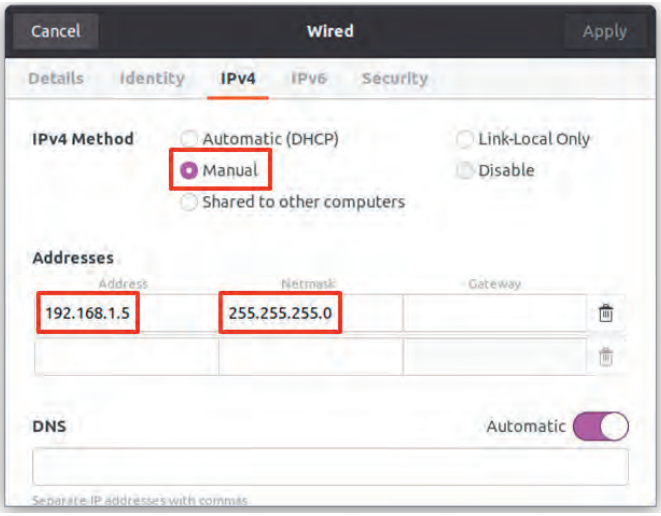


Hinweis: Falls der Control PC über zwei Netzwerkkarten verfügt, muss hier die **Netzwerkkarte** ausgewählt werden, die für die **Direktverbindung zum Roboter** verwendet werden soll.
Die andere **Netzwerkkarte** dient dem Control PC als **Zugang zum Netzwerk** bzw. Internet.

- Unter **Identity** kann der Name der neuen Verbindung festgelegt werden:



- Wechsle zu **IPv4** und schalte die Methode auf **Manual** um:



- Vergib eine statische IPv4 Adresse in der gleichen IP-Range wie Franka:
  - z. B. Franka IP = 192.168.1.11
  - → Rechner IP = 192.168.1.5
- Hinweis: Hier ist mit „Rechner IP“ die IP-Adresse der Direktverbindung zwischen PC und Roboter gemeint. Nicht die IP-Adresse des Rechners im Netzwerk!
- Setze die Netzmaske wie beim Franka Roboter (z. B. 255.255.255.0)
- Teste die Verbindung indem du die IP vom Franka Roboter (hier: 192.168.1.11) in die Adresszeile von einem Webbrowser eingibst

4.3.3.2 Vorbereitung - PC

Öffne ein Terminal: über die Suche „Terminal“, oder per Tastenkombination **Strg + Alt + T**.
Gib nacheinander die folgenden Befehle in das neue Terminal ein:

- Git (Software zur verteilten Versionsverwaltung von Dateien) installieren:

```
sudo apt install git
```

- Erstelle einen neuen Arbeitsbereich:

```
mkdir -p ~/catkin_ws/src
```

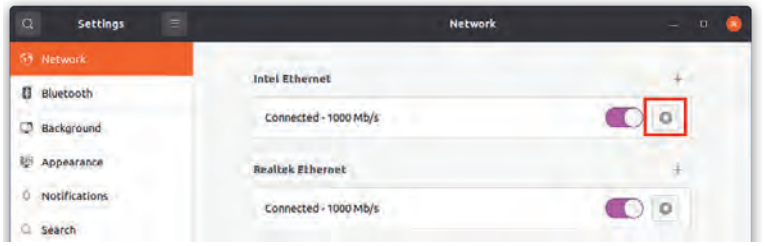
- Wechsle das Verzeichnis:

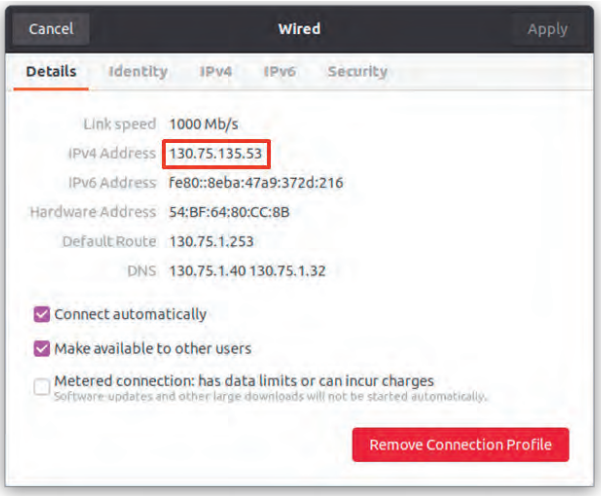
```
cd ~/catkin_ws/src
```

- Klone das Framework Repository:

```
git clone -b main https://gitlab.uni-hannover.de/roboterfabrik/rf_robot_framework.git .
```

- Nun müssen wir die **IP-Adresse des Rechners** herausfinden, mit der sich der Rechner im Netzwerk befindet. In dieser Anleitung besitzt der Control PC **zwei Ethernet-Karten**. Eine stellt die **Direktverbindung zum Roboter** her und die andere Karte dient nur **Verbindung mit dem Netzwerk** über eine statische IP-Adresse.
- Um die **IP-Adresse des Control PC zum Netzwerk** herauszufinden, gehen wir zu den **Settings** und **Network** und klicken auf das Zahnrad:



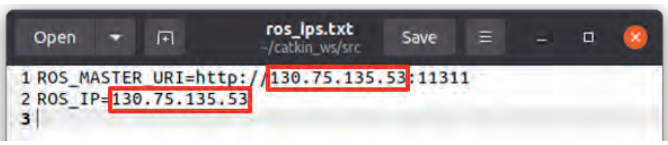


- Ändere die ROS_MASTER_URI und die ROS_IP auf dem Developer PC in der Datei ros_ips.txt:

```
gedit ./ros_ips.txt
```

- Durch den Befehl öffnet sich ein Fenster, in dem folgende Einstellungen gesetzt werden müssen:

```
ROS_MASTER_URI = http://<IP vom Control PC>:11311
ROS_IP = <IP vom Developer PC>
```

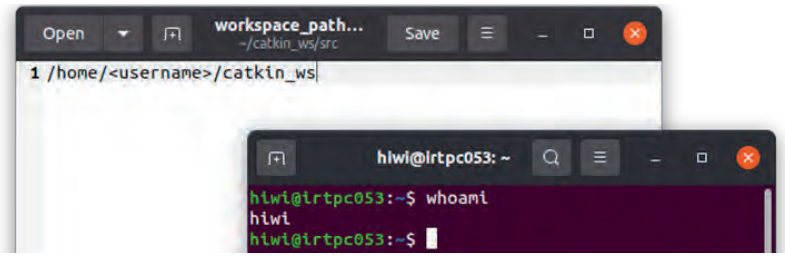


Hinweise:

- Mit „IP vom Control PC“ ist die IP-Adresse gemeint, mit der sich der Computer im Netzwerk befindet. Nicht gemeint ist die IP der „Franka_Direktverbindung“!
- Die dritte Zeile der Datei ros_ips.txt muss vorhanden und leer sein!
- Bei **Setup 2** ist der Control-PC auch der Developer PC, weshalb **beide IPs gleich** sind!

- Ändere den Pfad zum Catkin Workspace in der Datei workspace_path.txt:
  - Es muss nur der Nutzername geändert werden. Dazu am besten folgenden Befehl in einem Terminal ausführen:

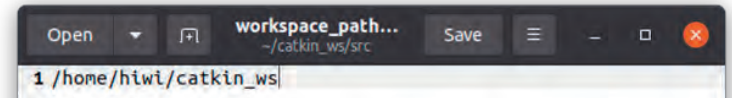
```
whoami
```



In dem dargestellten **Beispiel** liefert der Befehl whoami den Benutzernamen bzw. username **hiwi**.

- Anschließend die Ausgabe kopieren und den Platzhalter <username> in der workspace_path.txt Datei durch den Benutzernamen ersetzen:

```
gedit ./workspace_path.txt
```



Setup 1

- Führe folgende Skripte auf dem Developer PC (=Control PC) aus:

```
./host_preparation.sh
./kernel_preparation.sh
```

Setup 2

- Führe folgendes Skript auf dem Developer PC aus:

```
./host_preparation.sh
```

- Führe folgende Skripte auf dem Control PC aus:

```
./host_preparation.sh
./kernel_preparation.sh
```

Anschließend in beiden Fällen (Setup 1, Setup 2) Neustart des PCs durchführen.

- Jetzt muss nur noch das kompilierte Docker Image heruntergeladen werden:

```
cd ~/catkin_ws/src
./pull_docker_image.sh
```

Nun ist die PC Vorbereitung abgeschlossen.

Anpassung des Docker Containers

Nur wenn kein Zugriff auf das GitLab Docker Repository möglich ist oder das Container Image selbst gebaut werden soll, kann folgender Befehl ausgeführt werden:

```
contbuild
```

Dieser Befehl sollte nach Möglichkeit vermieden werden, es sei denn er ist ausdrücklich gewünscht. Der Download des Docker Images ist unbedingt zu bevorzugen. Wenn das Docker Image via `contbuild` neu erstellt wird, werden auch die neusten Versionen von *libfranka* und *franka_ros* installiert, was möglicherweise zu Inkompatibilitäten führen kann. Eine Möglichkeit für die Installation bestimmter Versionen ist aber als Kommentar im Dockerfile angegeben.

4.3.4 RF Robot Control Framework - Startup

Dieses Kapitel führt durch den Prozess der Inbetriebnahme des Roboters und die Steuerung durch das RF Robot Control Framework.
Das Tutorial geht von Setup 1 aus, d.h., der PC sollte direkt mit dem Roboter Controller verbunden sein und der Echtzeit-Kernel sollte laufen (Control PC = Developer PC).

1. Starte den Franka Controller
2. Öffne einen Browser
3. Gib die IP des Roboters in die Suchleiste des Webbrowsers ein. Dadurch wird das Franka Desk Interface geöffnet. Es wird zum Ent- und Verriegeln der Gelenke und zum Aktivieren des Franka Control Interface (FCI) verwendet.
4. In Desk: Entriegle die Gelenke. Der Roboter wird sich leicht bewegen. Wenn die Gelenke entriegelt sind, wechselt die Farbe der Schnittstelle von gelb zu weiß.
5. Aktiviere das FCI, indem du im Menü auf der rechten Seite auf **Activate FCI** klickst. Das Browserfenster kann anschließend minimiert oder geschlossen werden.

Hinweis: Es empfiehlt sich die Installation des Terminal-Multiplexers **Terminator**, der es ermöglicht, mehrere Terminals innerhalb eines einzigen Fensters zu benutzen. Das Fenster lässt sich mit der Tastenkombination **Strg + Shift + O** horizontal bzw. mit **Strg + Shift + E** vertikal teilen.
Die Installation von Terminator erfolgt mit dem folgenden Befehl in einem Terminal:

```
sudo apt install terminator
```

6. Öffne ein Terminal auf dem Control-PC (nachfolgend **Control-Terminal**)
7. Starte einen Docker Container mit Echtzeitfähigkeit auf dem Control-PC. Führe dazu folgenden Befehl im **Control-Terminal** aus:

```
contkernel
```

8. Starte die Control Node für die Verbindung zu Franka via FCI mit folgendem Befehl in dem Control-Terminal:

```
roslaunch rf_robot_control Panda.launch robot_ip:=<Roboter IP> load_gripper:=true
```

- Stelle sicher, dass die richtige IP-Adresse des Roboters für den Platzhalter <Roboter IP> im obigen Befehl eingegeben wurde. In unserem Beispiel lautet die Roboter IP **192.168.1.11**
- Damit dieser Befehl funktioniert, müssen die Robotergelenke bei der Befehlseingabe entriegelt sein.

Hinweis: Möchte man eine Node im Terminal beenden, funktioniert das über die Tastenkombination **Strg + C**. Das Beenden eines Docker Containers geht über **Strg + D** im Terminal.

9. Öffne ein neues Terminal auf dem Developer-PC (nachfolgend Developer-Terminal). Im Terminator geht das z. B. über **Strg + Shift + E**, indem man das aktuelle Fenster vertikal teilt.
10. Starte hier einen Docker Container auf dem Developer-PC. Führe dazu folgenden Befehl in dem Developer-Terminal aus:

```
cont
```

Dieser Schritt startet den Docker Container und baut den gesamten Arbeitsbereich auf, einschließlich der installierten Pakete.
Achtung: Falls weitere Geräte (z. B. USB-Kamera) genutzt werden sollen, müssen diese mit dem PC verbunden sein, bevor der Container gestartet wird.
11. Um das Projekt zu kompilieren, muss in dem Developer-Terminal das Folgende ausgeführt werden:

```
cd ~/catkin_ws
catkin build
```

Info: Falls Setup 1 verwendet wird (Developer-PC = Control-PC), müssen die Befehle trotzdem in zwei verschiedenen Terminals ausgeführt werden.

12. Jetzt kann mit der Entwicklung begonnen werden und Franka über ein Graphical User Interface (GUI) angesteuert werden. Um das GUI zu starten, führe folgenden Befehl in dem Developer-Terminal aus:

```
rosrun rf_robot_control node
```

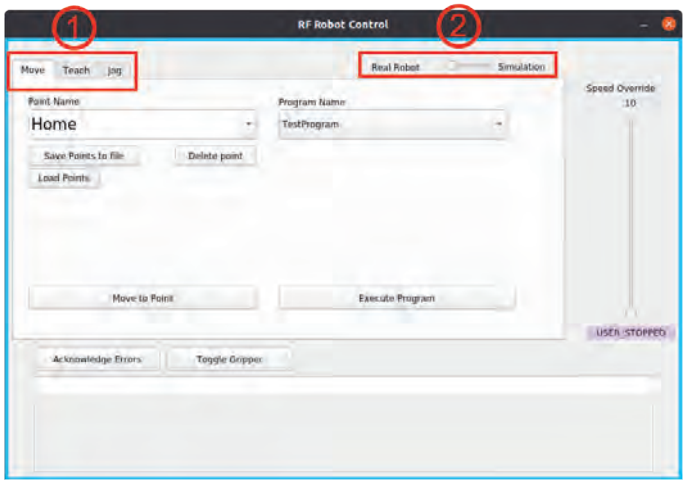
Die Benutzeroberfläche bzw. GUI von RF Robot Control wird geöffnet. Hier können Punkte angelernt bzw. „geteached“ werden und das eigene Programm ausgeführt werden. Falls Änderungen an einem Programm vorgenommen wurden, muss das Projekt neu kompiliert werden. Dazu muss das GUI geschlossen und die Schritte 11 und 12 erneut ausgeführt werden. Sobald wir in VS unseren Code anfangen zu schreiben, ist es komfortabler die Schritte 11 und 12 über ein Terminal im Fenster von VS auszuführen.
Hinweis: Wenn sich das Programm nicht wie gewünscht verhält, ist es wahrscheinlich, dass das Projekt nicht neu kompiliert wurden. Vor dem Kompilieren sollte der Code zunächst gespeichert werden. Die Funktionen des Frameworks werden im nächsten Abschnitt beschrieben.

4.3.5 RF Robot Control Framework - Nutzung

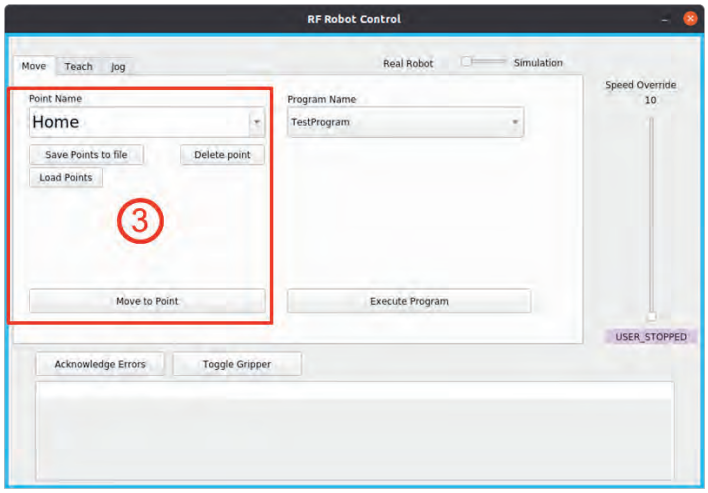
4.3.5.1 Graphical User Interface (GUI)

Im Folgenden werden die Funktionen des GUIs von **RF Robot Control** beschrieben:

1. Registerkarten zum Wechseln zwischen den Dialogfeldern **Move**, **Teach** und **Jog**:
   - **Move** ist der Modus, in dem der Roboter Bewegungen ausführt. Entweder beim Anfahren von zuvor „geteachten" Punkten oder zum Ausführen eines erstellen Programms
   - **Teach** ist der Modus zum Einspeichern bzw. „Teachen" von Punkten
   - **Jog** zum Variieren der Gelenkwinkel des 7-achsigen Roboters (Info: funktioniert nur im Modus Simulation! (siehe (2))
2. Schiebeschalter zum Wechseln zwischen den Modi **Real Robot** (blau Fensterumrandung) und **Simulation** (grüne Fensterumrandung). Auf die Simulationsumgebung wird im Kapitel „RF Robot Control Framework – Simulationsumgebung" noch genauer eingegangen.

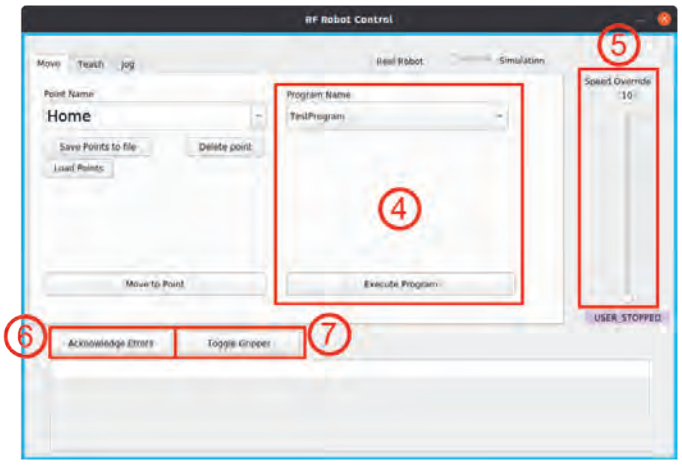


3. Dialog der gespeicherten Punkte:
   - **Point Name**: Dropdown-Liste der zuvor „geteachten" Punkte
   - Button **Save Points to file** zum Speichern der Punkte in einer Datei (Ort: rf_robot_control/config/points.yaml)
   - Button **Load Points** zum Öffnen von gespeicherten Punkten (yaml-Datei)
   - Button **Delete point** um einzelne Punkte aus der Dropdown-Liste zu löschen
   - Button **Move to Point** um Roboter zu aus Dropdown-Liste ausgewählten, zuvor eingespeicherten Punkt zu bewegen

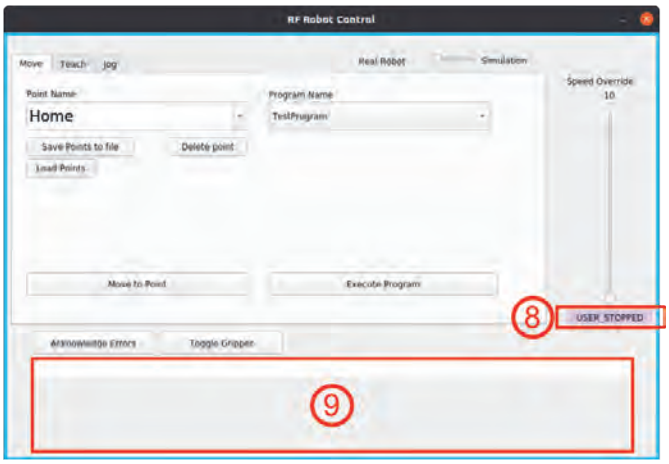


Hinweis: Die Punkte werden standardmäßig in der Datei points.yaml gespeichert. Es ist nicht notwendig nach dem „Teachen" und Löschen von Punkten, die Punkte in der Datei points.yaml abzuspeichern. Das geschieht bereits automatisch. Ebenso wird die Datei points.yaml geladen, sobald das GUI geöffnet wird.

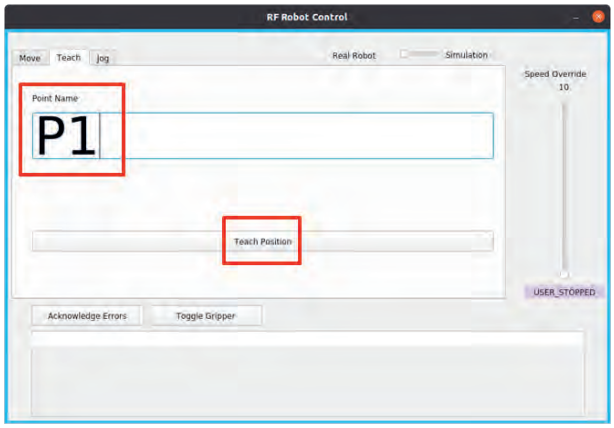
4. Dialog zum Aufrufen und Ausführen des Programms:
   - **Program Name**: Dropdown-Liste zum Auswählen des Programms, das ausgeführt werden soll (Standardprogramm: TestProgram)
   - Button **Execute Program** zum Ausführen des zuvor ausgewählten Programms
5. Schieberegler **Speed Override** zur Geschwindigkeitsübersteuerung des Roboters in Prozent
6. Button **Acknowledge Errors** zum Quittieren von Fehlermeldungen in der Anzeige
7. Button **Toggle Gripper** zum Öffnen und Schließen des Greifers (engl. gripper)



8. Statusanzeige des Roboters (versch. Modi):
   - **USER_STOPPED**: User Stop Button ist gedrückt, Roboter bewegt sich nicht
   - **IDLE**: User Stop Button ist gelöst, Roboter ist untätig (engl. idle)
   - **MOVE**: User Stop Button ist gelöst, Roboter bewegt sich
   - **GUIDING**: User Stop Button ist gedrückt, Roboter wird manuell geführt
   - **REFLEX**: Fehler bei Programmausführung
9. Anzeige von Informationen, Fehler- und Warnmeldungen:



Registerkarte Teach zum Einspeichern von Punkten:

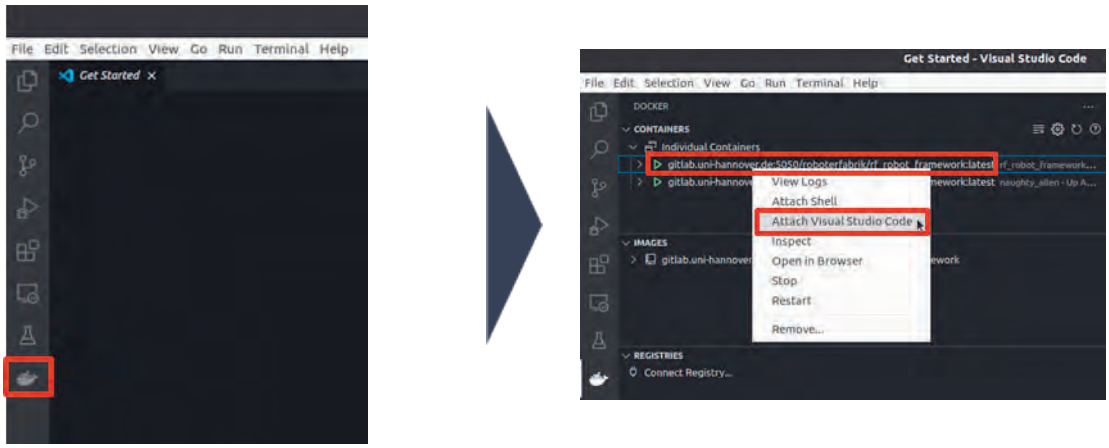


Nachdem der **User Stop** des Roboters gelöst wurde, kann der Roboter bei gleichzeitiger Betätigung des **Guiding Tasters** und **Zustimmungsschalters** manuell bewegt werden, um eine neuen Punkt zu „teachen".
Dazu bewegt man den Roboter zunächst zu dem gewünschten Punkt. Dann kann man dem Punkt unter **Point Name** einen Namen vergeben (z. B. P1) und mit einem Klick auf den Button **Teach Position** abspeichern.

4.3.5.2 Programmierung mit Visual Studio Code

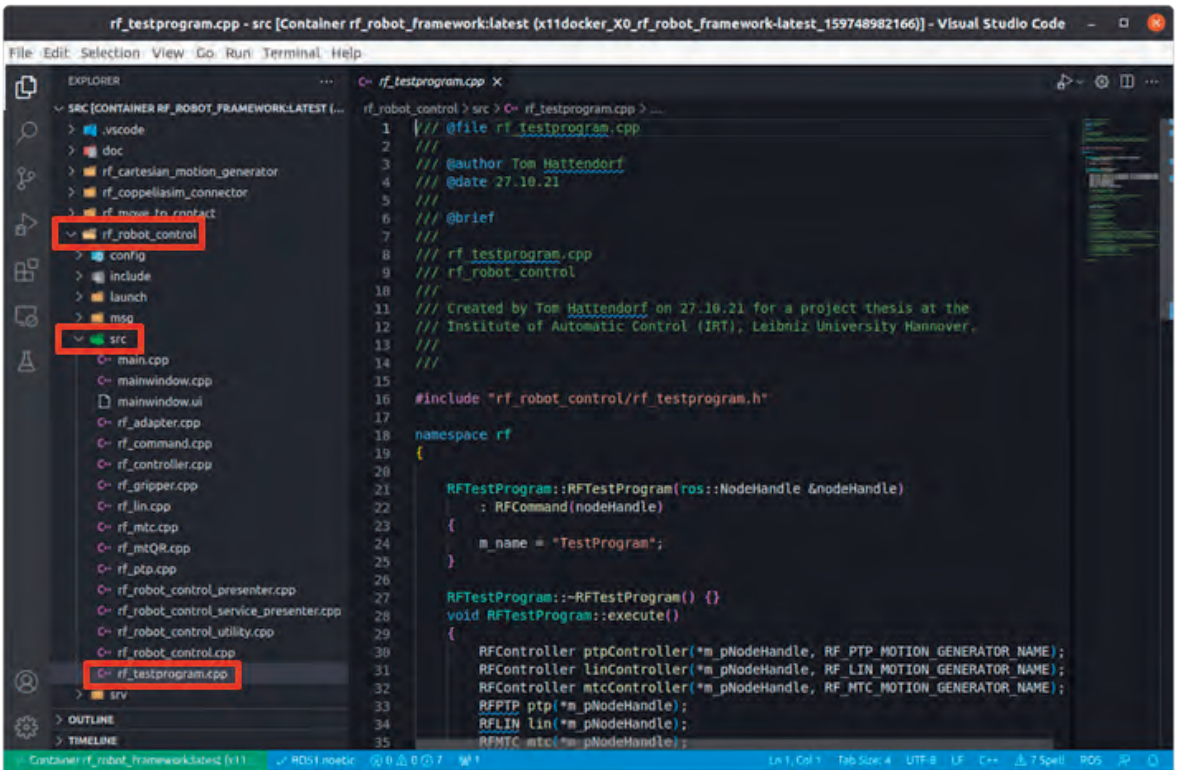
Im Folgenden wird beispielhaft anhand eines Testprogramms beschrieben wie man eigene Programme mit dem RF Robot Control Framework mithilfe des Editors Visual Studio Code schreibt. Dafür gibt es im Repository schon das Beispielprogramm „TestProgram".
Um den Code eines Programms zu editieren bzw. um ein eigenes Programm zu schreiben, ist es am einfachsten den Arbeitsbereich des Docker Containers in Visual Studio Code zu laden. Dazu sind die folgenden Schritte nötig.

- Öffne **VS Code** und klicke mit der linken Maustaste im Menü auf der linken Seite auf die Docker Erweiterung. Sobald der PC-Container gestartet ist, ist er auf der linken oberen Seite unter der Kategorie **Containers** zu sehen.
- Klicke mit der rechten Maustaste auf den ersten Eintrag **gitlab.uni-hannover.de:5050/roboterfabrik/rf_robot_framework:latest** und dann auf **Attach Visual Studio Code**.

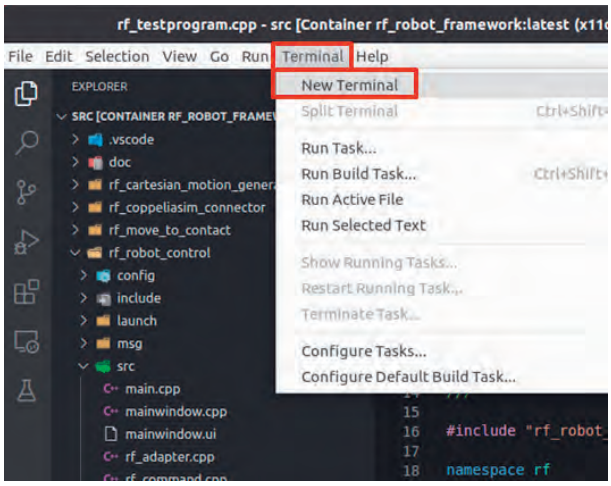


Es öffnet sich ein neues Fenster von VS Code mit dem Arbeitsverzeichnis des Ordners catkin_ws.
Das ursprüngliche VS Code Fenster kann nun geschlossen werden.

- Nun können wir anfangen unseren Code in VS Code zu schreiben. Dazu navigieren wir durch die Verzeichnisstruktur zu der Datei rf_testprogram.cpp.
- Die Datei enthält den C++-Code des Programms „TestProgram", welches über das GUI RF Robot Control ausgeführt werden kann.

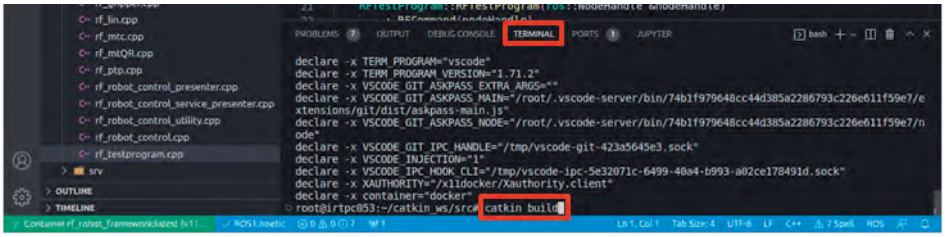


- Sobald wir in VS Code arbeiten, bietet es sich an, in einem einzigen Fenster zu arbeiten. Dort können wir den Code kompilieren und das GUI öffnen bzw. den Code ausführen.
- Dazu können wir in VS Code ein neues Terminal öffnen (Terminal → New Terminal).



- Sobald der Code in der Datei rf_testprogram.cpp zu Ende editiert und gespeichert (**Strg + S**) wurde, kann der Code in dem neuen Terminal mit dem folgenden Befehl kompiliert werden:

```
Catkin build
```



- Nach dem Kompilieren des Codes, kann das GUI über den folgenden Befehl geöffnet werden

```
rosrun rf_robot_control node
```

- Über das GUI lässt sich schließlich der Code ausführen und die Bewegung des Roboters kann gestartet werden.

4.3.5.3 Doxygen Dokumentation

Für Informationen zu den verschiedenen Klassen und Befehlen des Frameworks kann die **Doxygen Dokumentation** im Webbrowser gelesen werden:

1. Öffne ein neues Terminal mit **Strg + Alt + T** und gibt den folgenden Befehl ein, um in das Verzeichnis der Dokumentation zu wechseln und um das zip-Archiv der Dokumentation zu extrahieren:

```
cd ~/catkin_ws/src/doc && unzip doxygen_output.zip
```

2. Gib in ein Terminal den nächsten Befehl ein, um die Doxygen Dokumentation z.B. im Firefox Browser zu öffnen (alternativ kann man die Datei über den Dateimanager öffnen):

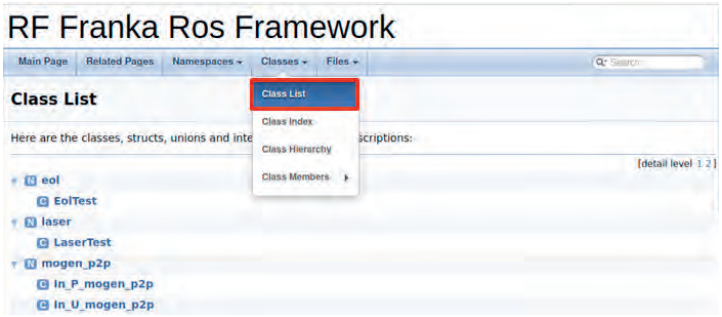
```
firefox ~/catkin_ws/src/doc/doxygen_output/html/index.html
```

3. In dem sich öffnenden Fenster kann die Dokumentation nun gelesen werden.
   Aktuell unterstützt das Framework drei verschiedene Bewegungsarten (Klassen):
   - MoveCartesian (**RFLIN**): erzeugt lineare Bewegungen von einem kartesischen Punkt zu einem anderen
   - MoveJoint (**RFPTP**): erzeugt Bewegungen direkt von einem Punkt zu einem anderen im Gelenkraum (engl. joint space)
   - MoveToContact (**RFMTC**): ein linearer Punkt bewegt sich in eine bestimmte Richtung bis eine entgegengesetzte Kraft überschritten wird

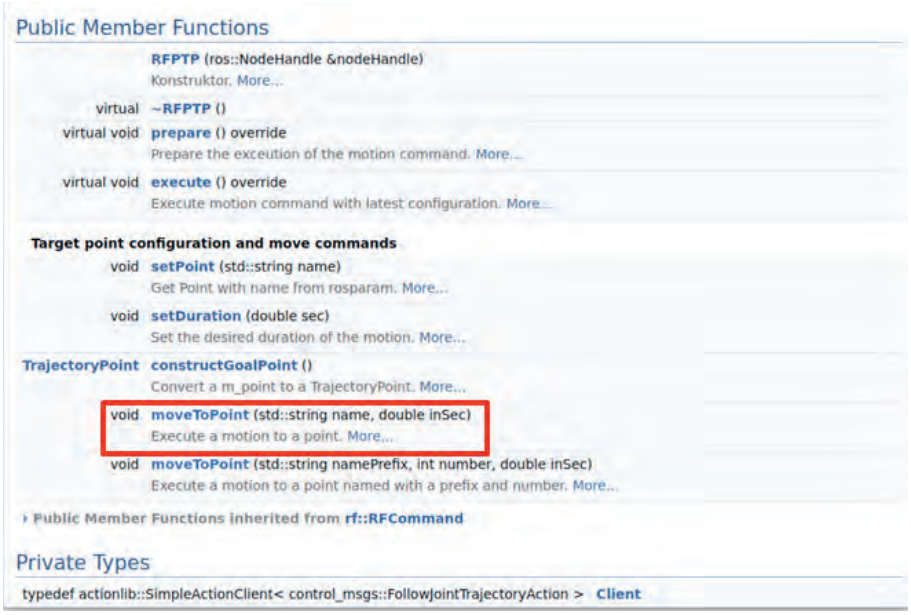
Neben den Klassen der Bewegungsarten (**RFLIN**, **RFPTP**, **RFMTC**) ist die Klasse des Greifers (**RFGRIPPER**) von besonderer Bedeutung bei der Programmierung von Roboteranwendungen.

In der Doxygen Dokumentation des Frameworks können die verschiedenen **Klassen** genauer studiert werden. Dort kann man sich die möglichen **Befehle** samt der erforderlichen **Parametrisierung** anschauen. Im Folgenden wird am **Beispiel** der Klasse **RFPTP** gezeigt, wie man durch die Doxygen Dokumentation navigiert, um die **Befehle** und **Parameter** genauer zu verstehen:

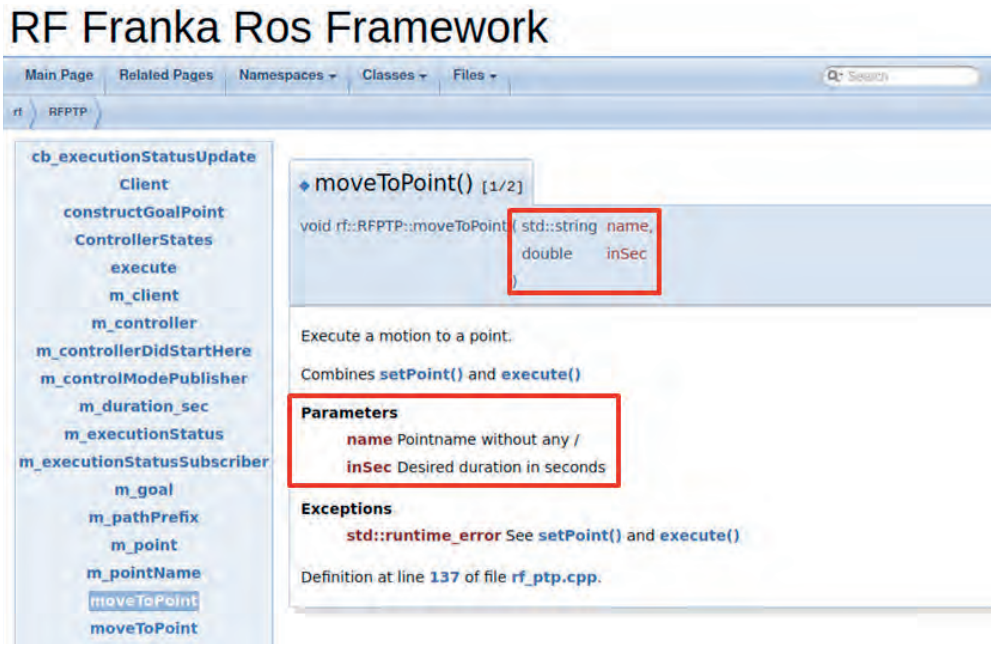
1. Auf der Startseite bzw. **Main Page** der Doxygen Dokumentation kann man in der Navigationsleiste die Maus auf „**Classes**“ bewegen und auf „**Class List**“ klicken:



2. Auf der Seite zu den verschiedenen Klassen (Class List) kann man zu RFPTP navigieren, um nähere Informationen darüber zu erhalten, welche Befehle die Klasse bereitstellt:



4. Nun kann man genauere Informationen zum Befehl moveToPoint() erhalten:
   - Parameter: Reihenfolge + Datentypen
   - Kurzbeschreibung der Parameter

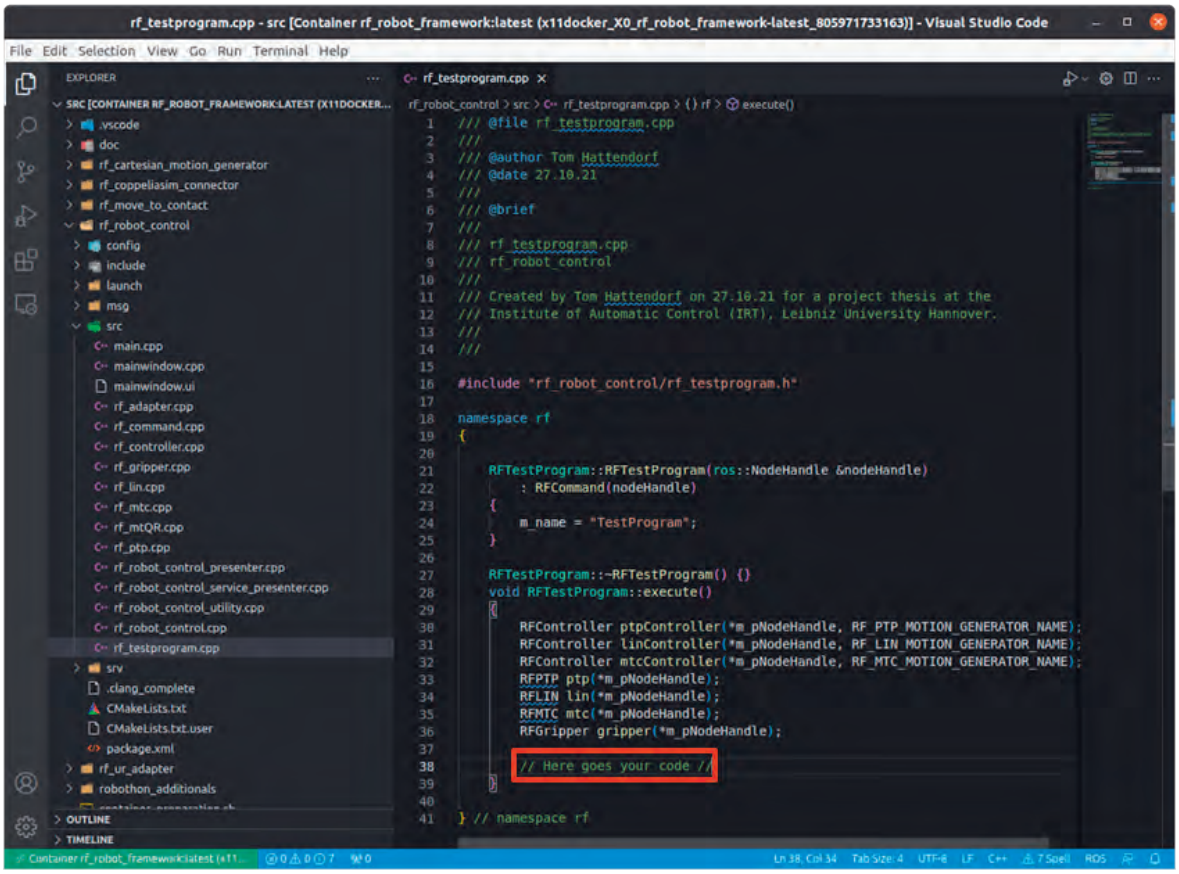


Weitere wichtige Klassen bzw. Befehle sind:

- **RFGRIPPER**
  - gripper.open()
  - gripper.close()
  - gripper.grasp()
- **RFLIN**
  - lin.setSpeed()
  - lin.moveRelativeToPoint()
- **RFPTP**
  - ptp.moveToPoint()
- **RFMTC**
  - mtc.moveToContact()

4.3.5.4 Programmierbeispiel

Als nächsten thematisieren wir anhand eines **Beispiels** die Programmierung mit Hilfe des Frameworks. Dazu arbeiten wir in **Visual Studio Code** in der Datei rf_testprogram.cpp. Das Programm lädt bereits alle notwendigen Controller für die drei Bewegungsarten sowie für den Gripper. Die Stelle an welcher der eigene Code eingefügt werden sollte, ist im folgenden Bild in Rot markiert. Der Rest der Datei kann zunächst unverändert bleiben.



Beim Programmieren stehen zwei wichtige **Hilfsmittel** zur Verfügung:

- Doxygen Dokumentation
- Auto-Vervollständigung (IntelliSense in Visual Studio Code)

IntelliSense kann in VS Code mit der Taste **Tab** aufgerufen werden. Dadurch wird intelligente Autovervollständigung und Hilfe bei Parametern ausgelöst.

4.3.5.4.1 Beispielaufgabe

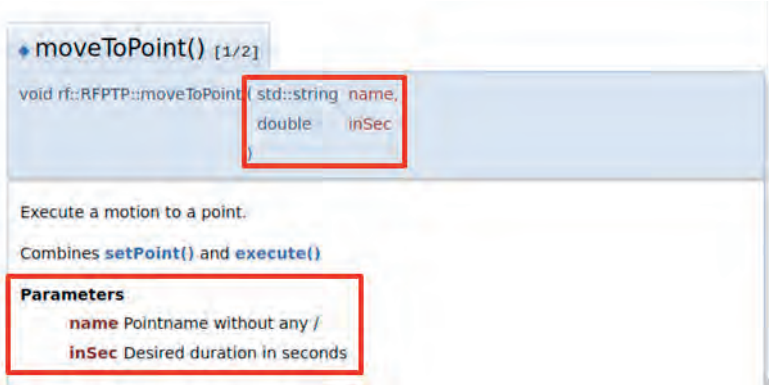
Schreibe ein einfaches Programm zur Durchführung der folgenden Teilaufgaben.
Wir gehen davon aus, dass die Punkte P1, P2 und Home bereits in einem vorherigen Schritt über das GUI RF Robot Control eingespeichert bzw. „geteached“ wurden.

Teilaufgaben:

1. Bewege den Roboter mit einer Dauer von 2 Sekunden mit der Bewegungsart PTP (PointToPoint) zu Punkt P1 und öffne den Greifer.
2. Bewege den Roboter linear von P1 mit einer maximalen Geschwindigkeit von 0,1 m/s um 5 cm nach unten.
3. Nimm einen Würfel mit einer Kantenlänge von 1 cm auf.
4. Bewege den Roboter linear mit der Standardgeschwindigkeit zu Punkt P2.
5. Lege das Objekt an Punkt P2 ab.
6. Bewege den Roboter mit PTP zurück zum Punkt Home.

4.3.5.4.2 Lösung Beispielaufgabe

Teilaufgabe 1: Bewege den Roboter mit einer Dauer von 2 Sekunden mit der Bewegungsart PTP (Point-ToPoint) zu Punkt P1 und öffne den Greifer.

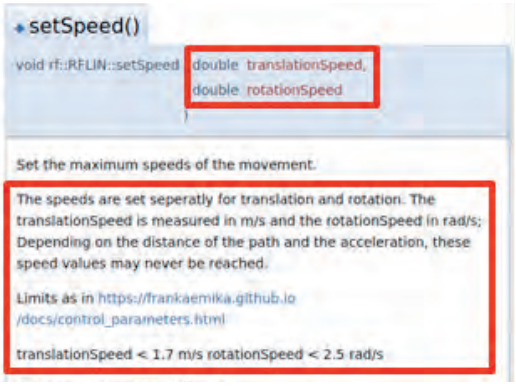


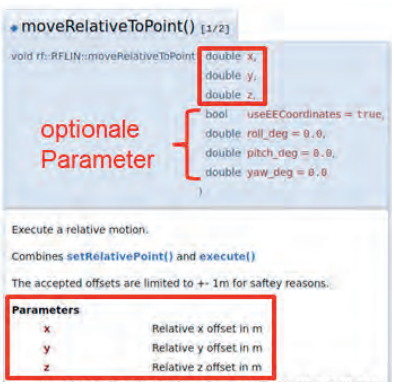

```
27    RFTestProgram::~RFTestProgram() {}
28    void RFTestProgram::execute()
29    {
30        RFController ptpController(*m_pNodeHandle, RF_PTP_MOTION_GENERATOR_NAME);
31        RFController linController(*m_pNodeHandle, RF_LIN_MOTION_GENERATOR_NAME);
32        RFController mtcController(*m_pNodeHandle, RF_MTC_MOTION_GENERATOR_NAME);
33        RFPTP ptp(*m_pNodeHandle);
34        RFLIN lin(*m_pNodeHandle);
35        RFMTC mtc(*m_pNodeHandle);
36        RFGripper gripper(*m_pNodeHandle);
37
38        // Task 1:
39        // PTP motion is in general faster than a linear motion
40        ptp.moveToPoint("P1", 2);
41        gripper.open();
42    }
43
44    } // namespace rf
```

Hinweise:

- Vor dem Kompilieren speichern!
- Am Ende eines Befehls Semikolon nicht vergessen!
- Code ein- und auskommentieren funktioniert mit **Strg + Shift + 7**

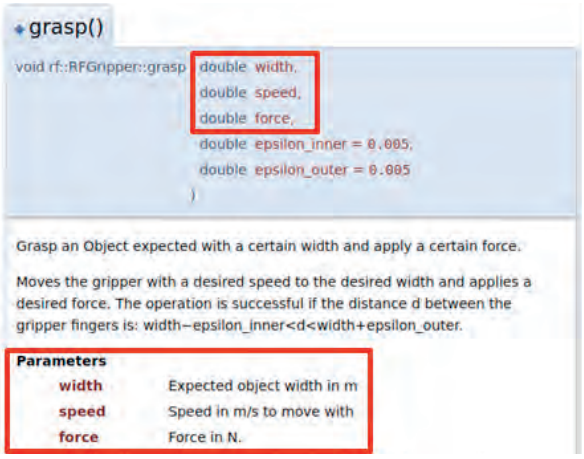
Teilaufgabe 2: Bewege den Roboter linear von P1 mit einer maximalen Geschwindigkeit von 0,1 m/s um 5 cm nach unten.






```
38        // Task 1:
39        // PTP motion is in general faster than a linear motion
40        ptp.moveToPoint("P1", 2);
41        gripper.open();
42
43        // Task 2:
44        // Set the desired speed for translation. Rotation set as default value.
45        // Execute a relative motion.
46        lin.setSpeed(0.1, 0.5);
47        lin.moveRelativeToPoint(0, 0, 0.05);
48    }
```

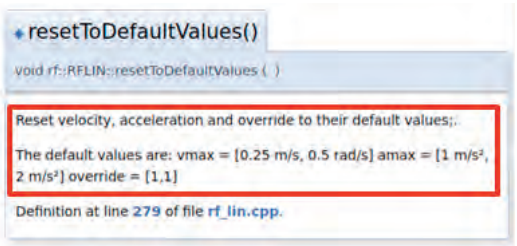
Teilaufgabe 3: Nimm einen Würfel mit einer Kantenlänge von 1 cm auf.




```
48
49      // Task 3:
50      // Grasp the object.
51      // gripper.close() could also be possible, but we know the exact object size.
52      gripper.grasp(0.01, 0.05, 20);
53  }
```

Hinweis: Das Programm bricht ab, wenn sich der Würfel nicht an dem Punkt befindet, wo er aufgenommen bzw. gegriffen werden soll.

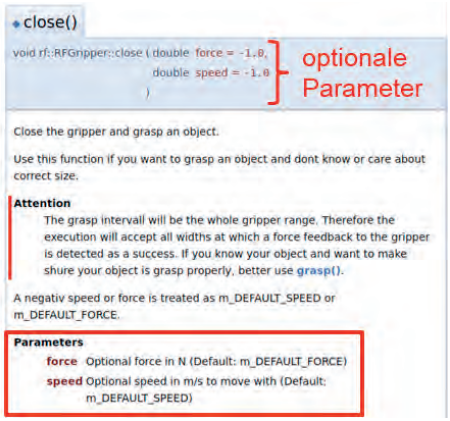
Teilaufgabe 4: Bewege den Roboter linear mit der Standardgeschwindigkeit zu Punkt P2.




```
53
54      // Task 4:
55      // The speed from Task 2 is still set. Therefore, we have to reset it.
56      lin.resetToDefaultValues();
57      lin.moveToPoint("P2");
58  }
```

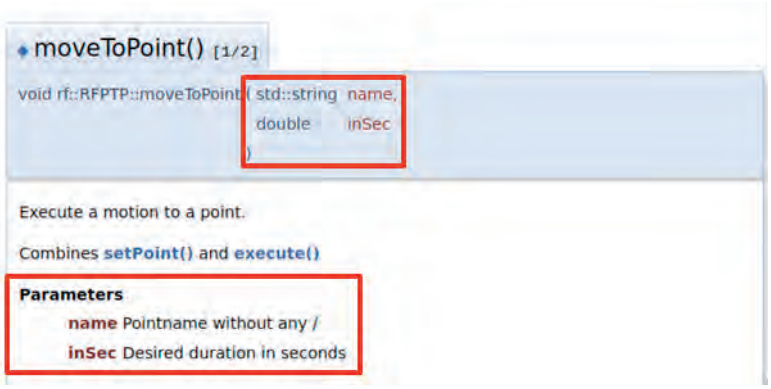
Hinweis: Das Programm bricht ab, wenn sich der Würfel nicht an dem Punkt befindet, wo er aufgenommen bzw. gegriffen werden soll.

Teilaufgabe 5: Lege das Objekt an Punkt P2 ab.



```
58
59      // Task 5:
60      // Simply open the gripper with default speed.
61      gripper.open();
62  }
```

Teilaufgabe 6: Bewege den Roboter mit PTP zurück zum Punkt Home.




```
62
63      // Task 6:
64      ptp.moveToPoint("Home", 2);
65  }
```

4.3.5.4.3 Exception Handling

Nachdem wir unser erstes Programm erstellt haben, können wir uns mit der Behandlung von Ausnahmen (engl. **exceptions handling**) auseinandersetzen. Bei der Arbeit mit echten Robotern können **unvorhersehbare Dinge** passieren, daher lösen viele Befehle, wenn sie nicht erfolgreich sind, Ausnahmen aus.

Beispiel: Nehmen wir an, du willst einen Stein vom Tisch greifen und ihn auf einen anderen legen. Dafür würdest du den Befehl `gripper.grasp()` verwenden. Einer der Parameter ist die erwartete Objektbreite. Unter normalen Umständen kann das Objekt problemlos gegriffen werden. Was passiert jedoch, wenn man vergisst, einen Stein an der gewünschten Stelle zu platzieren?
In diesem Fall wird von `gripper.grasp()` eine **Ausnahme geworfen**, weil der Greifer weit mehr als die eingestellte Objektbreite geschlossen wird. Das ist eigentlich eine gute Sache, weil etwas schiefgelaufen ist. Diese Ausnahme (exception) kann mit einem **try-catch-Block** behandelt werden.

In der Programmiersprache C++ sind **try-catch-Blöcke** eine Art und Weise Ausnahmen zu behandeln. Alles, was schief gehen kann, wird im try-catch-Block stehen. Wenn etwas eine Ausnahme auslöst, kann der **catch-Block** das Ausnahmeobjekt abfangen und entsprechend darauf reagieren. Andernfalls wird der Code im catch-Block niemals ausgeführt werden. Die meisten Ausnahmen im Framework sind std::runtime_error.
Wenn irgendwo eine Ausnahme ausgelöst wird, geht das Programm auf dem Aufrufstapel (engl. **call stack**) nach oben, um einen geeigneten catch-Block zu finden. Wenn kein Block gefunden wird, wird das Programm beendet. Mit anderen Worten: Wenn man eine Ausnahme nicht abfängt, wird das Programm nicht fortgesetzt. Aus Sicherheitsgründen ist das eigentlich eine gute Sache.
Nun schauen wir uns ein **Beispiel** an: Der nachfolgende Codeschnipsel ist die Grundroutine zum Greifen eines Steines.

Codeschnipsel (**ohne** try-catch-Block):

```
38      lin.moveToPoint("Pickup");
39
40      gripper.grasp(0.015, 0.05, 20);
41
42      lin.moveToPoint("Place");
43      gripper.open();
44      lin.moveToPoint("Home");
45  }
```

Befindet sich kein Stein am Punkt Pickup, löst gripper.grasp() eine unabgefangene Ausnahme aus. Der Roboter wird einfach am Punkt Pickup mit geschlossenem Greifer verharren. Die Bewegungen zu den Punkten Place und Home werden nicht ausgeführt.

Codeschnipsel (**mit** try-catch-Block):

```
38      lin.moveToPoint("Pickup");
39
40      try
41      {
42          gripper.grasp(0.015, 0.05, 20);
43      }
44      catch (const std::runtime_error &e)
45      {
46          ROS_ERROR_STREAM("Gripper grasp failed. Aborting and moving back to Home: " << e.what());
47          gripper.open();
48          lin.moveToPoint("Home");
49          throw;
50      }
51
52      lin.moveToPoint("Place");
```

Befindet sich kein Stein am Punkt Pickup, löst `gripper.grasp()` eine Ausnahme aus, die im catch-Block abgefangen wird. Als Reaktion wird der Benutzer darüber informiert, dass der Vorgang fehlgeschlagen ist. Der Roboter öffnet wieder den Greifer und bewegt sich zurück nach Home.

Bitte beachte, dass die Ausnahme am Ende des catch-Blocks erneut ausgelöst wird. Das liegt daran, dass die Funktion `execute()` ihren Zweck nicht erfüllt hat und daher ebenfalls fehlgeschlagen ist.

Allerdings befindet sich der Roboter nun in einem definierten Zustand (Home, Greifer offen).

4.3.6 RF Robot Control Framework – Simulationsumgebung

CoppeliaSim, früher bekannt als V-REP, ist ein **Robotersimulator**, der in Industrie, Bildung und Forschung eingesetzt wird. Innerhalb des RF Robot Control Frameworks kann er genutzt werden um Code zunächst in einer Simulationsumgebung zu testen bevor ein echter Roboter benötigt wird. In der aktuellen Version funktionieren jedoch noch keine Befehle, die Kraft- oder Momentmessung benötigen (wie z. B. MoveToContact) und es gibt noch weitere kleinere Einschränkungen, die am Ende des Kapitels zusammengefasst sind.
Um den Robotersimulator zu starten, muss zunächst der folgende Befehl in ein **Developer Terminal** eingegeben werden, um den Developer Container zu starten:

```
cont
```

Öffne nun VS Code und wähle über die Docker Erweiterung in der linken Leiste den Container **rf_robot_framework:latest** mit einem Rechtklick aus und klicke auf **Attach Visual Studio Code**. Das ursprüngliche VS Code-Fenster kann geschlossen werden.
Nun kann mit der Entwicklung des Codes begonnen werden wie im Kapitel „Programmierung mit Visual Studio Code" beschrieben wurde. Um das Projekt zu kompilieren, kannst du in einem Terminal innerhalb von VS Code (Terminal → New Terminal) den nächsten Befehl eingeben:

```
catkin build
```

Durch einen **Rechtsklick auf das Terminal**, welches du innerhalb von VS Code geöffnet hast, und anschließendem Klick auf **Split Terminal** kannst du ein **weiteres Terminal** innerhalb von VS Code öffnen.

Möchtest du nun **CoppeliaSim starten**, gib den nachstehenden Befehl ein:

```
~/coppelia_sim/coppeliaSim.sh
```

Die Simulationsumgebung ist noch leer bzw. enthält keinen zu simulierenden Roboter.
Für die Verbindung mit dem RF Robot Control Framework ist es notwendig die **BlueZero** Schnittstelle von CoppeliaSim zu nutzen. Dafür kann der **BlueZero Server** wie folgend beschrieben gestartet werden:

- Klicke auf **Modules → Connectivity → B0 remote API server** und bestätige im sich öffnenden Fenster mit **Yes**

Nun können die **Roboter-Modelle** für die Simulation in CoppeliaSim ausgewählt werden.
Hinweis: Dafür bitte nicht die Modelle aus CoppeliaSim direkt auswählen, sondern die Modelle welche im Repository mitgeliefert werden. Ansonsten ist nicht garantiert, dass die Simulation einwandfrei funktioniert.
Es kann zwischen den Modellen des Franka Emika Panda und den Universal Roboter Modellen UR3 und UR5 gewählt werden.
Klicke auf **File → Load model...** und klicke dich zu dem Pfad der Modelle durch:

```
/root/catkin_ws/src/rf_coppeliasim_connector/models
```

Dort kann z.B. der Franka Emika Panda mit Greifern (engl. gripper) ausgewählt werden.
Das Modell des Franka Emika Panda mit Greifern sollte nun erfolgreich in die Simulationsumgebung von CoppeliaSim geladen worden sein.

Wir wechseln wieder zu **VS Code** und öffnen ein **weiteres Terminal** (Rechtsklick → Split Terminal). Wir sollten nun 3 Terminals innerhalb von VS Code offen haben:

1. Terminal zum Kompilieren des Codes via catkin build
2. Terminal zum Starten von CoppeliaSim via ~/coppelia_sim/coppeliaSim.sh
3. Terminal zum Starten des Hardware Interface und Controller Managers.

Das **Hardware Interface** und den **Control Manager** starten wir nun in unserem 3. Terminal mit dem folgenden Befehl (Info: Dabei wird auch der sog. roscore gestartet):

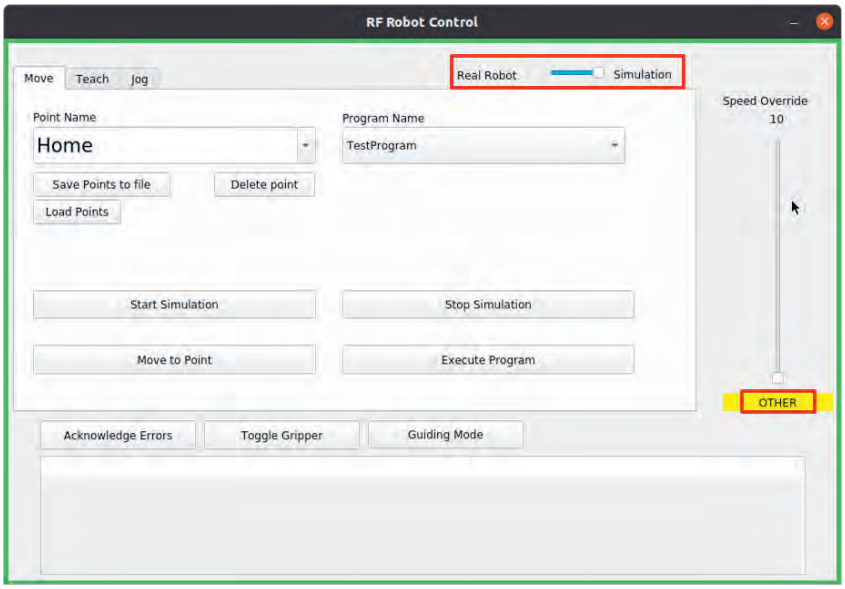
```
roslaunch rf_coppelia_interface panda.launch
```

Hinweis: Sollte zuvor ein Universal Roboter (Modell UR3 oder UR5) ausgewählt worden sein, muss der letzte Teil des Befehls universal.launch lauten.

Um nun das **GUI** von **RF Robot Control** zu öffnen, geben wir in ein 4. Terminal (Rechtsklick → Split Terminal) innerhalb von VS Code diesen Befehl ein:

```
rosrun rf_robot_control node
```

Sobald das GUI von RF Robot Control in der **Statusanzeige** den Modus **OTHER** (gelb) anzeigt, kann die Simulation gesteuert werden. Zunächst muss der Schieberegler von **Real Robot** auf **Simulation** umgestellt werden:

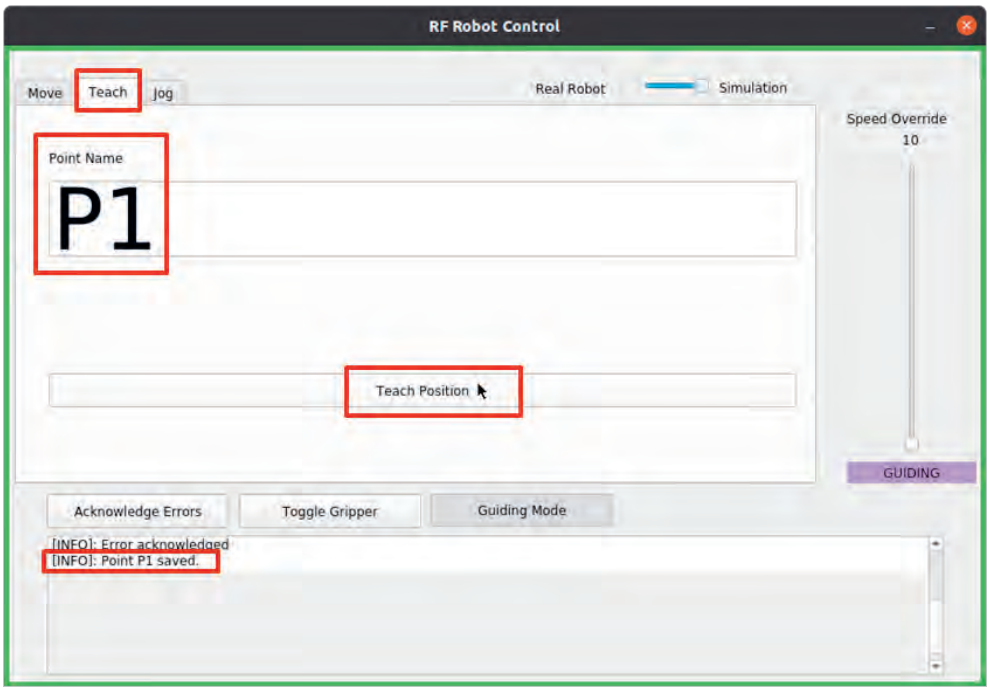


Die Benutzeroberfläche ist analog zum Modus Real Robot aufgebaut. Im Modus Simulation können wir:

- Punkte "teachen"
- den Roboter manuell bewegen bzw. „joggen"
- eingespeicherte Punkte via Move to Point anfahren
- Programme ausführen

- Wollen wir nun **Punkte** virtuell in der Simulationsumgebung „**teachen**", dann können wir dazu die Registerkarte **Jog** aufrufen und anschließend auf den Button **Guiding Mode** klicken. Die Statusanzeige sollte nun den Modus **GUIDING** (violett) anzeigen.
- Durch Variation der **Schieberegler** für die verschiedenen Gelenke (engl. joints) können wir den Roboter **manuell führen** (engl. to guide). Die Gelenke sind ausgehend von der Roboterbasis bis zum Endeffektor von 1 bis 7 durchnummeriert.

- Das Speichern der Punkte erfolgt genau wie beim echten Roboter. Durch Klicken auf die Registerkarte **Teach** kommen wir in den Dialog zum Einspeichern des zuvor angefahrenen Punktes.
- Nach der Namensvergabe im Feld **Point Name** klicken wir auf den Button **Teach Position**.



In der Anzeige erhalten wir die Information, dass der Punkt P1 erfolgreich eingespeichert wurde. Schließlich können wir weitere Punkte (P2, P3 etc.) durch einen Wechsel zwischen den Registerkarten **Jog** und **Teach** speichern. Sind wir mit dem Einspeichern unserer Punkte fertig, können wir das GUI von RF Robot Control schließen und zu VS Code wechseln.
In VS Code können wir die zuvor gespeicherten Punkte für unseren **Code** verwenden.

Die Simulationsumgebung hat aktuell noch folgende Einschränkungen:

- Der Roboter verfügt über keine Nachgiebigkeit (engl. compliance) in bestimmte Raumrichtungen.
- Funktionen wie MoveToContact oder das Durchführen von Aufgaben mit definierten Kräften fehlen.
- Außerdem ist das Greifen mit einer bestimmten Kraft nicht unterstützt. Jedoch können wir den Greifer öffnen und schließen und so ein Objekt greifen.

Nachdem wir unseren Code geschrieben, gespeichert und kompiliert haben, können wir erneut das **GUI RF Robot Control** öffnen.
Dort wählen wir über den Schieberegler **Simulation** aus und klicken auf den Button **Start Simulation**. Das zuvor geschriebene Programm kann nun durch einen Klick auf den Button **Execute Program** gestartet und ausgeführt werden.
In **CoppeliaSim** kann der Code durch die Bewegungen des Roboters nachvollzogen werden.

5 Beispiele für Lernsituationen

Inhalt

Im Verlauf des Projektes Robonatives sind an den Projektschulen sowie in zahlreichen gemeinsamen Fortbildungen und Workshops an den Standorten des Kompetenzzentrums eine Vielzahl von Unterrichtsentwicklungen, Lernsituationen, Lehr-/ Lernmaterialien usw. entstanden. Diese Projektergebnisse werden über die Niedersächsische Bildungscloud allen interessierten Schulen und Lehrkräften zur Verfügung gestellt.

Beispielhaft für die entwickelten Unterrichtsmaterialien werden im Folgenden ausgewählte Lernsituationen unter Angabe der beteiligten Lehrkräfte beschrieben.

Robotertyp	Titel	Beteiligte Lehrkräfte
Dobot Magician Anfänger	Containerschiff / LKW nach Dänemark	Jens Haufe, Peter Osing, August-Benninghaus-Schule Ankum Dirk Hehemann, Bertha-von-Suttner-Oberschule Osnabrück Jakob Hensel, Schule am Roten Berg Hasbergen Robert Ritter, Integrierte Gesamtschule Bramsche
Dobot Magician Fortgeschrittene	Waschmaschinen-produktionsanlage	Marco Düvelmeyer, Sören Meiners, Thomas-Morus-Schule Osnabrück Gerrit Herbers, Schulzentrum Lohne Eva Lienemann, Oberschule am Sonnenhügel Osnabrück
Universal Robots	Sortierung von Spritzgussteilen mithilfe von Kamerasystemen	Georg Janssen, Christian Schmidchen, Torsten Tjarks, Hauke Fremy, Martin Siemens, Berufsbildende Schulen II Emden
Mitsubishi Melfa Assista	Montage Lagerbock	Ingo Siebels, Hochschule Osnabrück Rene Egbers, Hochschule Osnabrück
Franka Emika	Wiederholgenauigkeit von Robotern	Malte Ohlsen, Integrierte Gesamtschule Seevetal

5.1 Dobot Magician Anfänger

UE / LS	Containerschiff / LWK nach Dänemark
Jg. + Fach / Lernfeld	Allgemeinbildende Schule Sek. I , 8. - 9. Klasse + Technik / Automatisierte Prozesse
Anzahl Unterrichtsstunden	ca. 4 Doppelstunden
Raum / Geräte / Lernträger	Dobot Magician, Sauggreifer, Kompressor, Lochplatten, Paletten, Containerwürfel, Schiff / LKW, Dübel / Schrauben / 3D-Stift für Arbeitsbereich, Dobot Studio
Handlungsziel/ -produkt	Folgende Handlungsergebnisse sollten erreicht werden: • Der Arbeitsbereich des Krans ist mit den Begrenzungspins abgesteckt. Die SuS erstellen ein Foto des Arbeitsbereiches. Sie dokumentieren auf dem Arbeitsblatt den Arbeitsbereich (AB Arbeitsbereich). • Mehrere Container werden aus dem Palettenlager links in das Hochregallager rechts transportiert. • Das Containerschiff fährt einen Zielhafen an und ist mit drei Boxen bestückt. • Das Containerschiff fährt mehrere Zielhäfen an und die drei Boxen und 1 Container sind sinnvoll auf das Schiff geladen. Bei Anfahrt der Zielhäfen werden die Container zeitsparend verladen. • Es liegt eine Dokumentation/ Präsentation vor. Inhalt: Fotos der Ergebnisse, Programmablauf, Kurzbeschreibung der Lösungswege.
Beschreibung	Der Kranführer Manfred hat seine theoretische Ausbildung erfolgreich absolviert. Der Hamburger Hafen ist sein neuer Arbeitsplatz und Manfred soll sich in den nächsten Tagen im Bereich des Güterumschlages einarbeiten. Er wird seinen Kran und den vorhandenen Arbeitsbereich genau kennen lernen. Dazu gehört, dass der Arbeitsbereich des Krans, welcher in der Lage ist automatisierte Arbeitsabläufe auszuführen, absteckt. Danach wird Manfred erste Container mit sensiblen Waren aus dem Lagerbestand auf bereitgestellte Paletten laden. Nachdem ihm dies erfolgreich gelungen ist, soll Manfred sein erstes Containerschiff/ seinen ersten LKW beladen. Dieser soll zwei Container nach Antwerpen bringen. Der zweite Auftrag beinhaltet mehrere Ziele, so dass Manfred die Container sinnvoll verladen muss und auch entscheiden muss, ob der Transport per Schiff oder LKW günstiger ist.
Kompetenzen	**Berufliche Handlungskompetenzen:** • Berufliche Handlungskompetenzen: • Berufliche Orientierung: Anwendungsgebiete der Automatisierung erläutern • Komplexe Arbeitsabläufe strukturieren • Gestalten, Beurteilen und Optimieren von Abfertigungsprozessen • Positionierungsvorgänge analysieren Die SuS sind am Ende der Lernsituation in der Lage, den Arbeitsbereich eines Dobot Magician zu kennzeichnen und für die weitere Arbeit zu nutzen. Sie können bei zukünftigen Lernsituationen mit anderen Robotern einschätzen und erkennen, unter welchen Bedingungen Aufgaben umgesetzt werden können und, dass die Arbeitsbereiche begrenzt sind. Darüber hinaus sind die SuS in der Lage, passende Bewegungsarten sinnvoll auszuwählen, damit der Transport von Containern etc. reibungslos gelingt. Die SuS führen einfache Dispositionsaufgaben aus: Sie legen die Reihenfolge für die Verladung der Boxen und Container sinnvoll fest, so dass alle Boxen/ Container am Zielhafen direkt entnommen werden können.

UE / LS	Containerschiff / LWK nach Dänemark
Kompetenzen	Die SuS sind in der Lage, ihre Arbeitsschritte und Ergebnisse in geeigneter Form zu dokumentieren und zu präsentieren. Sie verdeutlichen ihre Lösungswege und begründen Entscheidungen. **Fachkompetenz:** • Die SuS beschreiben eine automatisierte Prozesssteuerung. • Sie beschreiben die Auswirkungen prozessgesteuerter Produktionen. • Die SuS benennen automatisierte Prozesse aus ihrer Lebenswelt. • Die SuS beschreiben die einzelnen Schritte beim Ablauf eines automatisierten Prozesses. **Methoden-, Lernkompetenz und kommunikative Kompetenz:** • Die SuS beschreiben Möglichkeiten der Anwendung von robotergestützten Systemen. • Die SuS erläutern gesellschaftliche Konsequenzen des Einsatzes automatisierter Prozesse z. B. in der industriellen Lagerwirtschaft. • Die SuS verbessern technische Lösungen hinsichtlich gegebener Anforderungen. • Die SuS planen, konstruieren und erstellen das Modell einer automatisierten Abfertigung in der Lagerwirtschaft. **Personal- und Sozialkompetenz:** • Die SuS diskutieren technische Lösungen. • Die SuS beurteilen ihre Lösungen und optimieren gemeinsam ihre Arbeitsergebnisse. • Die SuS strukturieren ihre Arbeit und fügen ihre Ergebnisse zusammen.
Voraussetzungen	Lernsituation 1: Einführung / Grundlagenwissen, Lernsituation 2: Schreiben und Zeichnen, Teach & Playback

Unterrichtsverlaufsplan

Handlungsschritte (eine vollständige Handlung)	Unterrichtsverlauf	Materialien und Hinweise (Roboter, Computer, ABs, Filme, ...)
Analysieren	SuS lesen den Arbeitsauftrag aufmerksam, im Plenum werden Eckpunkte besprochen und ggf. das benötigte Material gelistet/ genannt.	• Arbeitsblatt • Power Point Präsentation • Tippblatt: QR Code • Dobots, Steckbretter, Container
Informieren	Erarbeitung aller Grundlagen, die (auch als theoretischer Hintergrund) zur Auftragsbearbeitung erforderlich sind.	• Arbeitsblatt • Power Point Präsentation • Tippblatt: QR Code • Dobots, Steckbretter, Container
Planen / Entscheiden	SuS entwickeln einen sinnvollen Arbeitsablauf. Sie legen die Reihenfolge der Container fest.	• Dobots, Steckbretter, Container • ggf. Ablaufplan
Durchführen	praktische Durchführung oder Simulieren bzw. „Durchspielen“ der Arbeitsplanung.	• Dobots, Steckbretter, Container
Kontrollieren / Bewerten	Testlauf, dann Präsentation der Gruppenergebnisse, Vergleich der Ergebnisse, kurze Diskussion	• Dobots, Steckbretter, Container
Reflektieren	Diskussion, ggf. Evaluationsbogen	• Unterrichtsgespräch

Arbeitsblatt 1

Containertransport nach Dänemark

Manfred hat einen neuen Job! Er ist als Kranführer beim Hamburger Hafen angestellt und hat seine theoretische Ausbildung erfolgreich abgeschlossen.

Aufgabe 1:

Nun geht es darum, dass seinen neuen Kran kennenlernt. Er muss wissen, welchen Arbeitsbereich der Kran hat.
Dazu fährt er mit seinem Kran manuell den Arbeitsbereich ab. Wenn er den Arbeitsbereich verlässt, leuchtet eine rote LED im Kran auf. Am Boden hilft ihm ein Kollege, indem er den Arbeitsbereich mit Hütchen absteckt.

Kennzeichnet den Standort des Roboters und den Arbeitsbereich auf dem zweiten Arbeitsblatt.

Aufgabe 2:

Nun darf Manfred endlich richtig loslegen. Seine Aufgabe besteht darin, einen Container aus dem Lager auf eine Palette zu verladen.

Aufgabe 3:

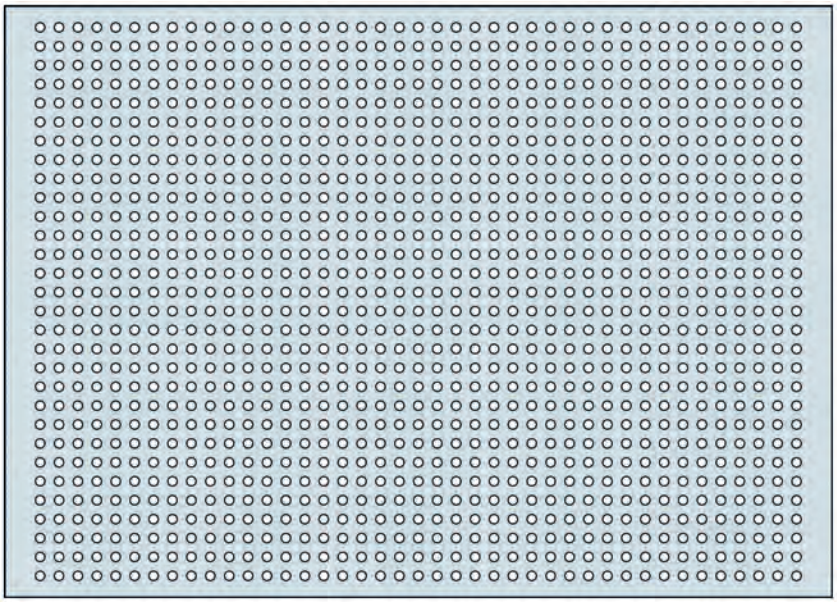
Nun soll Manfred sein erstes Schiff beladen. Das Schiff fährt von Hamburg über Esbjerg nach Kopenhagen. In Esbjerg sollen zwei kleine Container abgeladen werden. Die restlichen Container (ein kleiner und ein großer) sollen in Kopenhagen entladen werden.

Tipp: Überlege dir einen sinnvollen Ablauf des Beladens.

Präsentiert euren Mitschülerinnen und Mitschülern euer Ergebnis.

Arbeitsblatt 2

Arbeitsbereich:



5.2 Dobot Magician Fortgeschrittene

UE / LS	Waschmaschinenproduktionsanlage
Jg. + Fach / Lernfeld	9/10 + Informatik Oberschule/ Sek 1, Lernfeld „Algorithmisches Problemlösen“
Anzahl Unterrichtsstunden	3 Doppelstunden (270 Minuten)
Raum / Geräte / Lernträger	Je Gruppe: Dobot Magician (2x), Förderband (1x), fotoelektrischer Sensor, Endeffektor Sauggreifer, Endeffektor Parallelbackengreifer mit Gabel, Lochplatte mit Koordinaten, PC oder Laptop, Palette (2x) , Waschmaschinenmodell (2x) Zwei Lehrkräfte, Raum mit mittlerem Konferenzbereich, Techniklabor (ausgestattet, Arbeitsplätze), 2 S. pro Dobot, Sortiersystem (für Materialien)
Handlungsziel/ -produkt	Die Waschmaschinen werden durch Roboter auf Paletten auf dem Förderband zusammengesetzt. Das Förderband bewegt die Paletten zum zweiten Roboter, der die volle Palette entnimmt und gegen eine leere Palette tauscht. Als Endergebnis soll eine automatisierte Produktionsanlage realisiert werden.
Hinweise	Das Programmierniveau ist hoch!
Beschreibung	Die Firma Waschfix möchte ihre Produktion der Waschmaschine automatisieren. Aufgrund von Fachkräftemangel und hohen Produktionskosten soll ein möglichst hoher Automatisierungsgrad erreicht werden. Um dies zu erreichen, sollen Roboter für den Prozess eingesetzt werden.
Foto / Abbildung	https://www.youtube.com/watch?v=HuTdz5GumnU
Kompetenzen	**Basis: (P 1.1-2 P 2 P 3.1-2 P 4.1 I 2.1-2 I 3.2)** • SuS strukturieren Handlungsabläufe in logische Teileinheiten. • SuS benennen Anweisung, Sequenz, Schleife und Verzweigung als elementare Kontrollstrukturen. • SuS entwickeln und implementieren einen Algorithmus in einer grafischen Programmiersprache auf experimentelle Weise. • SuS überprüfen, ob eine Implementierung die Problemstellung löst. **Vertiefung: (P 3.3 I 2.3)** • SuS verwenden Variablen und Wertzuweisungen in einfachen Algorithmen. **Ergänzung: (P 1.3 P 4.2 I 1.1)** • SuS zerlegen einen komplexeren Algorithmus in mehrere Operationen, um z. B. Teillösungen wiederzuverwenden.
Voraussetzungen	Blockly, Schleifen, Pick and Place (MoveJ, MoveL, Jump), Dobot Studio, Sauggreifer, Parallelbackengreifer mit Gabel

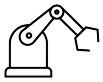
Unterrichtsverlaufsplan

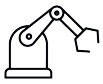
Handlungsschritte (eine vollständige Handlung)	Unterrichtsverlauf	Materialien und Hinweise (Roboter, Computer, ABs, Filme, ...)
Analysieren	Einstieg: L präsentiert das Einstiegsszenario eines fiktives Waschmaschinenherstellers. Im Anschluss wird das Beispiel-Video zur Produktionsanlage gezeigt.	

Handlungsschritte (eine vollständige Handlung)	Unterrichtsverlauf	Materialien und Hinweise (Roboter, Computer, ABs, Filme, ...)
Analysieren	Die SuS analysieren das Video im Hinblick auf die einzelnen Arbeitsabläufe in der Produktion. Klärung des Auftrages: 7 Schritte • Roboter 1: Zusammenbau der Waschmaschine auf einer Palette, die auf dem Förderband bereitsteht (Differenzierung schwerer – Palette mit 2 Waschmaschinen beladen, Differenzierung leichter – zusammengebaute Waschmaschine aufladen.) • Roboter 1: Förderband bewegen mithilfe der Geschwindigkeit/Distanz-Einstellung, Verzögerungszeit beginnt • Roboter 2: Photoelektrischer Sensor erkennt Palette. • Roboter 2: entnimmt beladene Palette und stellt sie ab. • Roboter 2: nimmt leere Palette und stellt sie auf das Förderband. • Roboter 1: Verzögerungszeit endet, bewegt das Förderband. • Roboter 1: beginnt neu zu beladen.	Film, AB mit Lernsituation, Dobots (2x), Förderband (1x), fotoelektrischer Sensor (1x), Paletten (2x), Saugnapf (1x), Parallelbackengreifer mit Gabel (1x), Waschmaschine (2x), Lochbrett mit Koordinaten (2x) AB für die Schritte (Praxis) und Funktionen (Theorie) Ablauf anhand Bildkarten sortieren (Screenshots der Videos)
Informieren	Erarbeitung Theorie: Hintergrundwissen „Funktionen“, „Förderband“, „Photoelektrischer Sensor“ • Die SuS informieren sich über den Verwendung und den Aufbau einer „Funktion“. Fachbegriffe: Funktion, Funktionsaufruf, Anweisung ggf. Funktionsrumpf (C++), • Die SuS informieren sich über die Verwendung und den Aufbau des Förderbandes „Geschwindigkeit und Distanz“ • Die SuS informieren sich über den „Photoelektrischen Sensor“ • Die Fachbegriffe „Schleife“, „Variable“ werden in einer kurzen Murmelphase wiederholt. Erarbeitung Praxis: Wiederholung in einer kurzen Murmelphase die Bewegungsarten, die Endeffektoren und den Aufbau von Blockly	Arbeitsblatt: Informationstexte oder Lehrervortrag Hinweis: Theorie und Praxis sind auf zwei verschiedenen Seiten der Arbeitsblätter. Es ist sinnvoll, die Arbeitsblätter nicht nacheinander zu bearbeiten, sondern zwischen Theorie und Praxis abzuwechseln.
Planen / Entscheiden	• Die SuS planen im Arbeitsblatt „Ablaufplan“ (Tabelle mit Arbeitsschritten) die einzelnen Arbeitsschritte, die für die Fertigung des Produktes notwendig sind. • Sie markieren Schritte, die von den Robotern gelöst werden müssen. • Differenzierung 1: SuS, die mit der Programmierung Schwierigkeiten haben, können sich auf einen Dobot konzentrieren mithilfe von „Pick and Place“ die Waschmaschine zusammensetzen. • Differenzierung 3: Die vollständige, mit Klebeband zusammengeklebte Waschmaschine wird auf die Palette gesetzt – Fokus „Förderband“. • Differenzierung 4: Die SuS konzentrieren sich auf die Aktivierung des Förderbandes mithilfe einer Funktion.	Hinweis: Das Förderband könnte auch mit dem Photoelektrischen Sensor gestoppt werden, hier entsteht jedoch eine Fehleranfälligkeit, da das fahrende Objekt nicht immer an der gleichen Stelle anhält. Die Verwendung der Geschwindigkeit/Distanz-Blöcke ist empfehlenswert.

Handlungsschritte (eine vollständige Handlung)	Unterrichtsverlauf	Materialien und Hinweise (Roboter, Computer, ABs, Filme, ...)
Durchführen	Die SuS simulieren den Zusammenbau und das Versenden sowie die Entnahme der befüllten Paletten mit Blockly (siehe Differenzierungsmöglichkeiten oben).	Dobots (2x), Förderband (1x), photoelektrischer Sensor (1x), Paletten (2x), Saugnapf (1x), Parallelbackengreifer mit Gabel (1x), Waschmaschine (2x), Lochbrett mit Koordinaten (2x) Hinweis: Nicht mit dem Freedrive Koordinaten setzen, dann wird die R-Achse verändert, fehleranfällig, deswegen immer mit dem Bedienfeld Koordinatenpunke setzen.
Kontrollieren / Bewerten	1. Kontrolle: A: zusammengebaute Waschmaschine befindet sich auf der Palette B: Förderband bewegt sich C: Sensor wird aktiviert D: Palette wird entnommen und wieder neu aufgeladen E: Förderband bewegt sich 2. Optional: Struktogramm-Programmierung	
Reflektieren	Mögliche Alternativen, (Unterrichts-) Prozessbewertung, Frage: Was würde ich beim nächsten Mal anders machen? Weiterarbeit im Hinblick auf die Erweiterung der Fertigungsstraße, Ausarbeiten einer Produktionsanlage – Eine Verbindung zum Verschiffen der Produkte (siehe LM2).	

Arbeitsblatt 1

Ziel der Stunden - Praxis	
	Wir setzen automatisiert und wiederholbar die Einzelteile einer Waschmaschine auf eine Palette zusammen, bewegen die Palette über ein Förderband, entladen das Förderband und beladen es neu (Produktionsstraße).

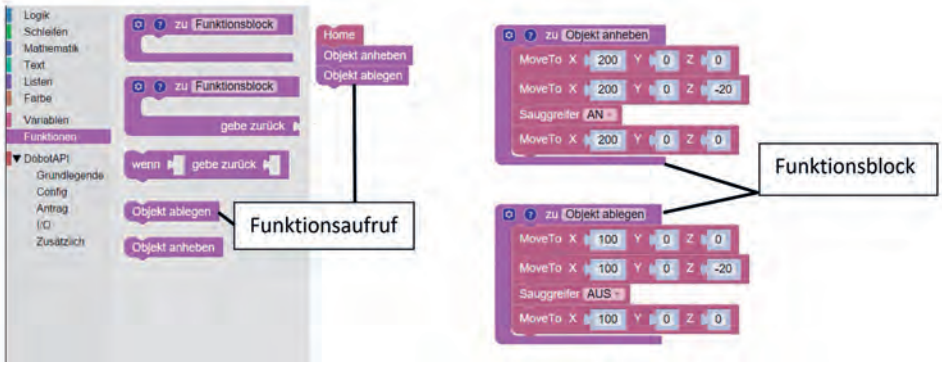
Die Firma Waschfix möchte ihre Produktion der Waschmaschine automatisieren. Aufgrund von Fachkräftemangel und hohen Produktionskosten soll ein möglichst hoher Automatisierungsgrad erreicht werden. Um dies zu erreichen, sollen Roboter für den Prozess eingesetzt werden.	
	**Aufgabe 1:** Betrachtet das Beispielvideo zur Produktion. Füllt die Lücken mit sinnvollen Arbeitsschritten. **Hinweis:** 2x Förderband aktivieren (Zwei Wege), 1x Photoelektrischer Sensor erkennt …

Schritte in der Produktion			
Roboter 1	Home	**Roboter 2**	Home
1.	Startposition anfahren	1.	Startposition anfahren
2.		Warte bis:	
3.			
4.			
1.	Startposition anfahren	3.	
Verzögerungszeit		4.	
		1.	Startposition anfahren
		6.	
		7.	
5.		1.	Startposition anfahren

	**Aufgabe 2A:** Programmiert die Arbeitsschritte von Roboter 1. Jede Zahl ist eine eigene Funktion. Das Förderband wird bewegt. **Aufgabe 2B:** Programmiert die Arbeitsschritte von Roboter 2. Jede Zahl ist eine eigene Funktion. Der Photoelektrische Sensor wird aktiviert. **Aufgabe 2C:** Verbindet die Programme, indem ihr die Verzögerungszeiten anpasst. **Zusatz-Aufgabe:** Überlegt euch, wie die Programme dauerhaft laufen könnten.

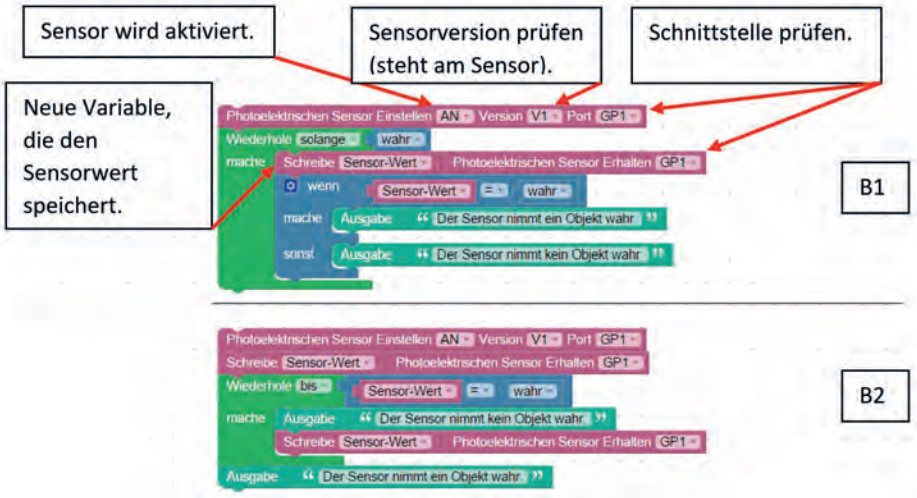
Arbeitsblatt 2

Ziel der Stunden - Theorie	
	Wir kennen die Fachbegriffe „Funktion“, „Förderband“ und „photoelektrischer Sensor“.

	**Aufgabe 1:** Lies den Informationstext „Funktion“ durch und erkläre anschließend deinem Partner den Fachbegriff. Verwende dabei die Wörter: „Funktion“, „Funktionsaufruf“, „Anwendungen“, „Funktionsblock“.
	**Eine Funktion** ist eine Art Unterprogramm. Eine Funktion zu verwenden ist sehr sinnvoll, da sie im Verlauf eines Programmes immer wieder durch den „Funktionsaufruf“ neu gestartet werden kann. Somit spart sich ein Entwickler unnötige Arbeitszeit, wenn bestimmte Aufgaben an verschiedenen Stellen wiederholt werden müssen. Außerdem können Funktionen ein Programm strukturieren, da einzelne Arbeitsabläufe durch einen eigenen Namen gekennzeichnet werden. Um Fehler schneller in einem Progamm zu finden, ist das hilfreich. Einer Funktion geben wir immer einen sinnvollen Namen, der in Kurzform die Aufgabe oder den Inhalt des Unterprogramms beschreibt. Anschließend wird der Funktionsblock (oder Funktionsrumpf) programmiert. Das heißt, du programmierst in die Funktion alle nötigen Anweisungen deiner klar abgegrenzten Aufgabe hinein. Der Funktionsblock (Funktionsrumpf) enthält alles, was dieses Unterprogramm für dich tun soll. Möchtest du nun im Verlauf deines Programmes diese Funktion aufrufen, verwendest du den Funktionsaufruf, also den von dir ausgesuchten Namen dieser Funktion. In Blockly-Programmierung sieht das z. B. so aus: 

Arbeitsblatt 3

Aufgabe 2: Erkläre die Funktion eines „photoelektrischen Sensors“. Gehe auch auf das „EVA-Prinzip“ ein. Recherchiere dazu selbstständig. Nenne die Quelle deiner Information mit Link.



Zwei Beispiele, wie der Sensor in ein Programm eingebunden werden könnte. Es gibt noch viele weitere Möglichkeiten.

B1:
Der photoelektrische Sensor wird an der passenden Schnittstelle aktiviert.

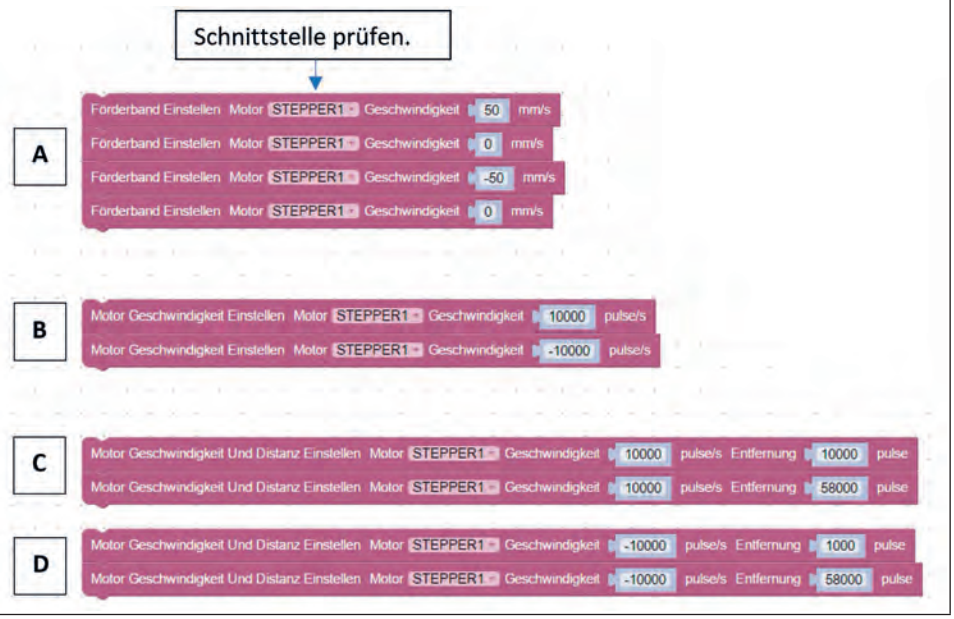
In der Endlosschleife wird eine Variable (Sensor-Wert) erstellt, die den Sensor-Wert speichert. Die Bedingung, ob die Variable einen Sensorwert erhalten hat (= wahr), wird geprüft und gibt in der Verzweigung den passenden Satz aus.

B2:
Der photoelektrische Sensor wird an der passenden Schnittstelle aktiviert. Es wird die Variable (Sensor-Wert) erstellt, die den Sensor-Wert speichert.

Die While-Schleife prüft, ob ein Sensor-Wert vorhanden ist:

- Wenn es keinen Sensor-Wert gibt, dann informiert sie darüber und überschreibt die Variable mit einem neuen Sensor-Wert.
- Wenn es einen Sensor-Wert gibt, bricht die While-Schleife ab und informiert, dass ein Sensor-Wert erhalten wurde.

Arbeitsblatt 4

	Aufgabe 3a: Recherchiere die Dateninformationen zum Förderband (Dobot), gehe auf die Tragfähigkeit, Förderstrecke, Beschleunigung und Geschwindigkeit ein. **Aufgabe 3b:** Probiert die vier unterschiedlichen Programmblöcke zum Förderband unten aus (A, B, C, D). • Beschreibe A, B, C und D kurz. Achte besonders auf die Minus- und Pluswerte, sowie die Maßeinheiten.
	

5.3 Universal Robots

UE / LS	Sortierung von Spritzgussteilen mithilfe von Kamerasystemen
Curricularer Bezug	Die Lernsituation kann u.a. in folgenden Ausbildungsberufen eingesetzt werden: • Industriemechaniker:innen • Produktionstechnolog:innen • Elektroniker:in für Automatisierungstechnik
Zeitumfang	10 Unterrichtsstunden
Lernfeld	**Industriemechaniker:innen:** • LF 6: Installieren und Inbetriebnehmen steuerungstechnischer Systeme • LF 13: Sicherstellen der Betriebsfähigkeit automatisierter Systeme Inhalte: Steuerung; Regelung; Programmierbare Steuerungen; Betriebsarten; Ablaufsprache, Funktionsbausteinsprache; flexible Handhabungssysteme; Schnittstellen; Sicherheitseinrichtungen • LF 15: Optimieren von technischen Systemen MRK als Optimierungsmöglichkeit von technischen Systemen **Produktionstechnolog:innen:** • LF 7: Strukturieren und Programmieren von technischen Abläufen • LF 9: Einrichten von Handhabungs- und Materialflusssystemen Die SuS können Robotersysteme als „Handhabungssysteme in flexible Fertigungsanlagen integrieren". „Technische Anforderungen für Robotersysteme und steuerungstechnische Systeme beschreiben". Ebenfalls sollen SuS „die Signale der Peripheriegeräte und der Sensoren über Schnittstellen mit der Ablaufsteuerung" verknüpfen. Des Weiteren sollen die SuS eigenständig Handhabungssysteme einrichten…" • LF 10: Simulieren von Produktionsprozessen • LF 11: Optimieren von Produktionsprozessen **Elektroniker:in für Automatisierungstechnik:** • LF 7: Steuerungen für Anlagen programmieren und realisieren • LF 8: Antriebssysteme auswählen und integrieren • LF 10: Automatisierungssystem in Betrieb nehmen und übergeben
Notwendige Hard- und Software	• UR3 (E-Series/ CB-Series) • 2x Förderband mit 4x Photoelektrischer Sensor • Bauteile (verschiedene würfelförmige Bauteile in den Farben rot, blau und grün; personalisierte Aufkleber) • Kamerasystem (Robotiq Wrist Kamera)
Voraussetzungen	Die SuS sind im Umgang mit den Robotern vertraut und haben bereits erste Programme geschrieben und auch die Förderbänder samt Sensorik implementiert.
Schulische Entscheidungen	Diese Lernsituation bildet den Einstieg in das an allen unseren Cobots verbaute Kamerasystem. Durch die Nutzung von würfelförmigen Bauteilen, wird die Handhabung, insbesondere das Greifen und Ablegen sehr vereinfacht. Auch das Einlernen des Kamerasystems ist bei geometrisch simplen Bauteilen einfacher umzusetzen. Bei stärkeren Lerngruppen bzw. einer tiefergehenden Behandlung des Themas ist eine Kooperation mit dem Gebiet der CAD-Anwendung in Verbindung mit 3D-Druck denkbar. So könnten unsere Technischen Produktdesigner:innen komplexere Bauteile konstruieren und an unserem schuleigenen 3D-Drucker drucken. Der regionale Bezug kann damit weiter ausgebaut und angepasst werden. Die Handhabung und das Einlernen der Kamera wird bei komplexeren Bauteilen deutlich erschwert. Für die elektronische Identifikation der Bauteile können Aufkleber mit QR-Codes verwendet werden. Die Aufgabenstellung eignet sich für eine Bewertung der Projektarbeit (idealerweise in Zweier-Teams). Die Inhalte können zusätzlich in eine schriftliche Leistungsüberprüfung einfließen.
Hinweise	Diese Lernsituation eignet sich für eine individuelle Anpassung an die Fähigkeiten der SuS, da sie modular strukturiert ist und in ihren Details veränderbar.

UE / LS	Sortierung von Spritzgussteilen mithilfe von Kamerasystemen
Ausgangssituation / Einstiegsszenario	Sie arbeiten bei einem Hersteller von Spitzgussbauteilen für die Automobilindustrie. Dort sind Sie als Spezialist für kollaborative Robotikanwendungen eingestellt. Ihre Firma hat einen neuen Auftrag vom hiesigen VW-Werk erhalten, bei dem würfelförmige Gehäuse für Steckkontakte für Elektroautos in drei Farben hergestellt werden. Die Bauteile werden auf drei unterschiedlichen Maschinen durch Spritzgießen hergestellt. Da die Firma Platz sparen muss, sind die jeweiligen Fertigteilauswürfe der drei Maschinen alle an das gleiche Förderband angebunden, so dass am Ende des Förderbandes eine Kiste mit den Bauteilen in drei verschiedenen Farben steht. In einen Verpackungskarton müssen jedoch immer 3 grüne Würfel, 3 rote Würfel und 3 blaue Würfel eingelegt werden. Zudem muss das blaue Gehäuse noch mit einem personalisierten Aufkleber (QR-Code) versehen werden. Ihr Chef kam nun auf die Idee, hierfür einen kollaborativen Roboter zum Sortieren einzusetzen, da dieser es ermöglicht, dass ein Arbeiter mit dem Roboter zusammenarbeiten kann und die personalisierten Aufkleber auf die blauen Gehäuse kleben darf. Er hat für diesen Anwendungszweck einen UR3-Roboter mit einem Kamerasystem gekauft und bittet Sie nun, das Roboterprogramm für den genannten Anwendungszweck zu schreiben. Die erforderlichen Bedienungsanleitungen bekommen sie ausgehändigt.
Foto / Abbildung	

Handlungsergebnis

Die SuS können ein automatisiert ablaufendes Roboterprogramm vorweisen, bei dem zuerst auf eine Fläche eine beliebe Anzahl verschiedener Bauteile in den Farben rot, blau und grün liegen und diese vom UR3 unter Zuhilfenahme des Kamerasystems sortiert werden (LZ1).
Die Gehäuse werden in eine jeweils für die Farbe vorgesehenen Karton gelegt, wozu auch eine Palettierung erforderlich ist. Ist dieser voll, wird er vom Roboter auf ein Förderband gelegt und weitertransportiert (LZ2). Dabei muss darauf geachtet werden, dass die blauen Gehäuse vor dem Sortieren an eine bestimmte Position fahren (LZ3), an der ein Werker einen Sticker aufklebt und das Aufkleben mit einer Berührung des Roboterarmes (Kraft- Momentensensorik) quittiert (LZ4). Ist das Aufkleben quittiert, dann sortiert der Roboter das Bauteil ein. Ist ein Karton voll (mit 3 roten, 3 grünen und 3 blauen Würfeln), dann gibt der Roboter eine Meldung aus (LZ6).

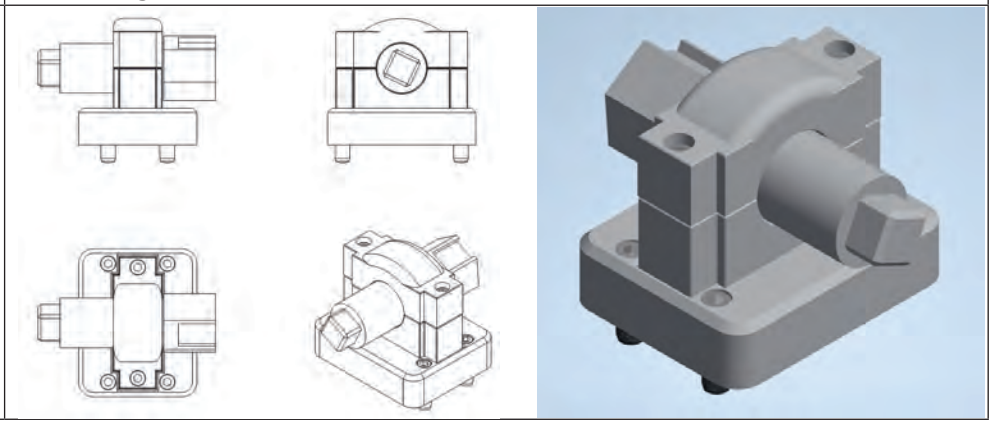
Berufliche Handlungskompetenzen

Die SuS sind am Ende der Lernsituation in der Lage, …
- ein Projekt eigenverantwortliche zu organisieren und durchzuführen.
- sich in einer Arbeitsgruppe zu organisieren.
- sich Inhalte allein durch die Nutzung von Bedienungsanleitungen zu erarbeiten.
- eine Kameraanwendung in ein Roboterprogramm zu integrieren.
- ein Roboterprogramm nach Kundenauftrag zu konzipieren.

Fachkompetenz	Methoden-, Lernkompetenz und kommunikative Kompetenz	Personal- und Sozialkompetenz
Die SuS integrieren Handhabungssysteme in flexible Fertigungsanlagen, dazu analysieren sie die erforderlichen Prozessabläufe. Die SuS entwickeln einen Programmstrukturplan und erstellen das Programm. Sie verknüpfen die Signale der Peripheriegeräte und der Sensoren. Die SuS testen und optimieren den Programmablauf.	Die SuS sind mit dem Roboter vertraut und haben bereits mehrere Programme nach aufbauendem Schwierigkeitsgrad erstellt. In dieser Lernsituation bekommen sie zur Hilfestellung somit nur noch die Bedienungsanleitungen ausgehändigt. Die Aufgabenstellung lässt Spielraum für die konkrete Umsetzung. Die SuS werden somit aufgefordert, die Problemstellung zielgerichtet und strategisch im Team umzusetzen. Für die Bewertung zur Umsetzung der Aufgabenstellung können von der Lehrkraft im Vorfeld konkrete Kriterien vorgegeben werden, wie z.B. die Funktionssicherheit, Anpassungsfähigkeit oder Optimierungsmöglichkeit des erstellten Programms.	Diese Lernsituation ist als eine Projektarbeit ausgelegt und somit spielt die eigenverantwortliche Arbeit eine große Rolle. Demnach werden hinsichtlich der Personal- und Sozialkompetenz folgende Eigenschaften gefördert: • Selbstständigkeit, durch die eigene Übertragung der Inhalte auf die Handlungssituation und die eigenständige Projektabwicklung. • Teamfähigkeit, durch die Abwicklung der Handlungssituation als Gruppe. • Arbeitsbereitschaft und Verantwortungsbewusstsein, durch die zugeteilte Gruppenaufgabe.

Zeit	Handlungsschritt (vollständige Handlung)	Grober Verlauf der Lernsituation	Medien / Materialien / Hinweise
2 Std.	Analysieren und Informieren	Die SuS analysieren die Anforderungen an die Aufgabenstellung. Es erfolgt eine Klärung und Präzisierung der Aufgabenstellung. Die SuS erstellen eine vollständige Anforderungsliste für diesen Auftrag. Die SuS informieren sich mit Hilfe der Bedienungsanleitungen über die Funktionsweise des Kamerasystems.	Bedienungsanleitung UR3 Cobots Bedienungsanleitung Kamerasystem E-Learning Kurs der UR-Academy
2 Std.	Planen/Entscheiden	Die SuS planen die Programmierung durch Zerlegen der Gesamtaufgabe in mehrere Teilaufgaben. Zunächst skizzieren sie einen groben Programmablauf .	Programmablauf als Pseudocode auf Papier oder Tablet
4 Std.	Durchführen	Die SuS erstellen das Programm direkt über das Bedienpanel der Cobots.	UR3-Schulungsstationen
1 Std.	Kontrollieren/Bewerten	Die SuS beurteilen und bewerten ihre Programmierung und vergleichen diese auch untereinander.	
1 Std.	Reflektieren	Die SuS reflektieren ihre Lösungen unter Berücksichtigung der Kundenzufriedenheit und der Prozesssicherheit. Sie reflektieren zudem ihre Vorgehensweise bei der Umsetzung der Programmieraufgabe.	

5.4 Mitsubishi Melfa Assista

UE / LS	Montage Lagerbock
Curricularer Bezug	Einsetzbar in unterschiedlichen Ausbildungsberufen, da die Lernsituation eine Einführung in die Robotik abbildet und vielseitig einsetzbar ist.
Zeitumfang	ca. 18 Unterrichtsstunden
Lernfeld	Exemplarisch sind hier die Lernfelder des Ausbildungsberufes für Industriemechaniker und Industriemechanikerinnen aufgelistet. **LF 6**: Installieren und Inbetriebnehmen steuerungstechnischer Systeme Zielformulierung: „Die Schülerinnen und Schüler installieren steuerungstechnische Systeme und nehmen sie in Betrieb“ Inhalte: Sensoren und Aktoren; Betriebsarten; Anlagensicherheit **LF 13**: Sicherstellen der Betriebsfähigkeit automatisierter Systeme Inhalte: Steuerung; Regelung; Programmierbare Steuerungen; Betriebsarten; Ablaufsprache, Funktionsbausteinsprache; flexible Handhabungssysteme; Schnittstellen; Sicherheitseinrichtungen **LF 15**: Optimieren von technischen Systemen MRK als Optimierungsmöglichkeit von technischen Systemen
Notwendige Hard- und Software	Mitsubishi MELFA Assista RV-5AS-D, Laptop, Software RT-ToolBox3, 3D-Druckteile der Lernsituation (Lagerunterschale, Lageroberschale, Welle, Lagerbock Basis und die Bauteilträger der jeweiligen Bauteile)
Voraussetzungen	Diese Lernsituation stellt die Einstiegslernsituation für die Robotik dar, daher werden keine Voraussetzungen an die Schülerinnen und Schüler gestellt.
Ausgangssituation / Einstiegsszenario	Sie sind im Unternehmen GK Sondermaschinenbau beschäftigt. Zu Ihren Aufgaben zählen neben der Produktionsüberwachung auch die Anpassung und Einrichtung von Maschinen auf neue Produkte. Ihr Unternehmen ist im Sondermaschinenbau tätig und fertigt daher im Tagesgeschäft hauptsächlich Kleinserienteile. Neue Aufträge, Umrüstungen, Programmierung und Neueinrichtungen für Werkstücke erfordern viel Flexibilität Ihrer Firma und der Fachkräfte. Trotz der häufigen Umrüstungen fertigen Sie mit der Unterstützung von moderner Robotertechnik, um wettbewerbsfähig in der Produktion zu sein. Damit diese Wettbewerbsfähigkeit durch eine unkomplizierte Zusammenarbeit zwischen Mensch und Roboter erhalten bleibt, wurden neue kollaborative Robotersysteme angeschafft. Diese sollen die Produktion zum einen flexibler unterstützen können und zum anderen neue Möglichkeiten für die Programmierung sowie Einrichtung bieten. Damit soll Zeit bei der Programmierung eingespart werden. Durch die Einführung wird eine Einarbeitung in das neue System notwendig. Ihre Aufgabe als Fachkraft ist es nun, die Automatisierung weiterer Fertigungsprozesse mit den neuen Robotersystemen durchzuführen. Als ersten Kundenauftrag erhalten Sie die kollaborative Automatisierung der Montage eines Lagerbocks. Ihre Firma nutzt einen neuen thermoplastischen Kunststoff mit Kohlefaseranteil, der keine Lagerbuchse benötigt. In den Lagerbock muss eine Welle eingesetzt werden, bevor die obere Schale des Lagerbocks verschraubt wird. Die Abbildung zeigt eine Darstellung des fertig montierten Lagerbocks. Der Roboter soll alle Tätigkeiten bis auf das Verschrauben des Lagebocks übernehmen, um die Produktionsmitarbeiter*innen zu entlasten.
Foto / Abbildung	

UE / LS	Montage Lagerbock
Handlungsergebnis	
Am Ende der Lernsituation sind die SuS in der Lage, ein Roboterprogramm zu schreiben, welches den Fertigungsprozess nach Kundenwunsch abdeckt. Hierbei verwenden sie gängige Bewegungsarten und Programmierfunktionen von Industrierobotern. Sie gestalten den Arbeitsplatz entsprechend der Mensch-Roboter-Kollaboration und sind in der Lage, Roboteranwendungen hinsichtlich der Bedienergesundheit zu reflektieren.	
Berufliche Handlungskompetenz	
Die SuS sind am Ende der Lernsituation in der Lage … • grundlegende Industrieroboterprogramme zu schreiben. • den Mitsubishi MELFA Assista zu bedienen. • Roboteranwendungen hinsichtlich ihrer MRK Tauglichkeit zu analysieren. • Sicherheitsrelevante Aspekte im Umgang mit Robotersystemen umzusetzen. • Handhabungsprozesse zu planen und umzusetzen.	

Fachkompetenz	**Methoden-, Lernkompetenz und kommunikative Kompetenz**	**Personal- und Sozialkompetenz**

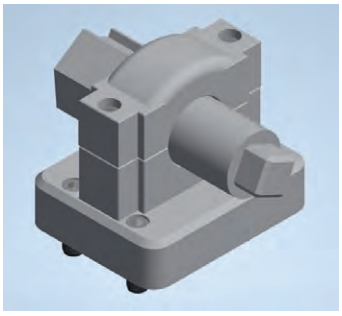
Die SuS lernen ...
• ... Grundlagen der kollaborativen Robotik, indem sie ...
  • ... Formen der MRK unterscheiden
  • ... Sicherheitsvorkehrungen für kollaborative Robotik anwenden
  • ... Aufbau von Robotern beschreiben
  • ... Arbeitsräume beschreiben
  • ... Koordinatensysteme der Robotik erklären und zielorientiert auswählen
  • ... einen kollaborativen Roboter steuern
• ... einen Montageablauf / eine Programmierung zu entwickeln, indem sie ...
  • ... Ordnungszustände bestimmen
  • ... Handhabungsfunktionen verwenden
  • ... Handhabungsprozesse planen
  • ... Programmbefehle lösungsorientiert auswählen
• ... einen kollaborativen Roboter zu programmieren, indem sie ...
  • ... Bewegungspunkte per Teach-in Verfahren einspeichern
  • ...ausgewählte Programmbefehle programmieren
• ... die eigene und andere Programmierungen zu bewerten, indem sie ...
  • ... Lösungen nach wirtschaftlichen Aspekten bewerten / optimieren
	Die SuS lernen ...
	• ... in Partner- und Gruppenarbeit
	• ... eigenständig Probleme zu lösen
		In dieser Lernsituation folgen Handlungsanweisungen hauptsächlich durch thematischen Input, der von den SuS in Gruppen auf die Handlungssituation übertragen werden muss. Demnach werden hinsichtlich der Personal- und Sozialkompetenz folgende Eigenschaften gefördert:
		• Selbstständigkeit, durch die eigene Übertragung der Inhalte auf die Handlungssituation.
		• Teamfähigkeit, durch die Abwicklung der Handlungssituation als Gruppe.
		• Arbeitsbereitschaft und Verantwortungsbewusstsein, durch die zugeteilte Gruppenaufgabe.
		• Soziale Verantwortung, durch den Einbezug gesundheitlicher Förderung von Arbeitskräften durch die Implementierung eines kollaborativen Robotiksystems.

Zeit	Handlungs-schritt (vollständige Handlung)	Grober Verlauf der Lernsituation	Medien / Materialien / Hinweise
90 min (2)	Analysieren	Einstieg durch einen stummen Impuls in Kombination mit einer Murmelphase mit dem Begriff „kollaborative Robotik“ an der Tafel. Das Ziel hierbei ist die Aktivierung von Vorwissen zum Thema. → Ergebnissicherung durch das gemeinsame Erstellen einer Mindmap an der Tafel. Die Lehrkraft gibt Ausblick über die nächsten Unterrichtsstunden und teilt die ersten beiden Seiten des Einstiegs-AB aus. Anschließend wird der Handlungsauftrag von den SuS analysiert und besprochen. SuS werden in drei unterschiedliche Gruppen mittels der World-Café-Methode eingeteilt. Daraufhin wird die dritte Seite ausgeteilt und die Gruppentische sind in einzelne Fachbereiche (Kollaborativer Roboter, Handhabungsobjekt Lagerblock und Werkzeuge & Peripherie) eingeteilt. SuS sollen die über die Fachbereiche diskutieren und wie sich zur Lösung des Kundenauftrag beitragen. Ergebnisse werden auf einer Flipchart zusammengetragen. Die Ergebnisse werden vorgestellt und im Plenum diskutiert.	Tafel, Flipchart Einstiegsarbeitsblatt: Montage Lagerbock
360min (8)	Orientieren/ Informieren	Die SuS erarbeiten die nachfolgenden Themenbereiche mithilfe von Arbeitsblättern. Die Erarbeitungsform wird hierbei zwischen den Themenbereichen variiert (siehe jew. Arbeitsblätter). • Formen der MRK unterscheiden • Sicherheitsvorkehrungen für kollaborative Robotik anwenden • Aufbau von Robotern beschreiben • Arbeitsräume beschreiben • Koordinatensysteme der Robotik erklären und anwenden • Steuerung von kollaborativen Robotern	Arbeitsblätter: • Kollaborative Robotik • Herstellerinformationen • Sicherheitsunterweisungen Teil 1 – 3, Biomechanische Grenzwerte • Koordinatensystem und das Verfahren des Roboters Software (RT-ToolBox 3) Kollaborative Roboter Beamer, Dokumentenkamera, Computer, Roboterprogramme
135min (3)	Planen/Entscheiden	Die SuS lernen anhand von Aufgaben handhabungstechnische Grundlagen, Elementarfunktionen des Mitsubishi und die Bewegungsarten kennen, um mit diesen Informationen ein Roboterprogramm zu planen, welches die Lagerböcke zusammensetzt. Darüber hinaus entscheiden sich die SuS für eine bestimmte Bewegungsart. SuS lernen durch Nachmachen die ersten Schritte der RT-ToolBox3. Zunächst wird in der Simulationsumgebung ein Testprogramm programmiert und anschließend wird in die Online-Programmierung umgeschaltet. Die Lehrkraft kontrolliert alle Testprogramme. SuS einigen sich mit der Lehrkraft auf eine Montagereihenfolge bevor sie mit der eigentlichen Programmierung des Roboter anfangen, um den Kundenauftrag zu lösen.	Arbeitsblätter: • Handhabungstechnische Grund-lagen • Zeichnungen zum Lagerbock-Arbeitsplatz • Bewegungsarten und Program-mierung Software (RT-ToolBox 3) Kollaborative Roboter Beamer, Dokumentenkamera Computer, Roboterprogramme

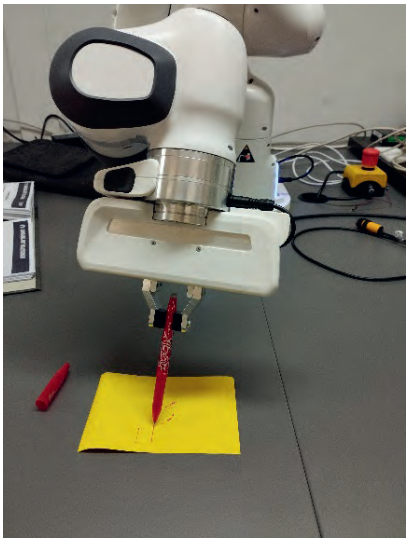
Zeit	Handlungsschritt (vollständige Handlung)	Grober Verlauf der Lernsituation	Medien / Materialien / Hinweise
180min (4)	Durchführen	SuS entscheiden sich für einen Lösungsweg und programmieren den kollaborativen Roboter eigenverantwortlich, indem verschiedene kleinere Aufgabenstellungen abgearbeitet werden. Es werden ausgewählte Programmbefehle programmiert und Bewegungspunkte eingeteacht, um den Kundenauftrag zu lösen.	Arbeitsblatt: Bewegungsarten und Programmierung Software (RT-ToolBox 3) Kollaborative Roboter Beamer, Dokumentenkamera Computer, Roboterprogramme
45min (1)	Bewerten / Reflektieren	Sobald die SuS die vorletzte Aufgabe (Nr. 5) absolviert haben, also das Lösen des Kundenauftrags, nimmt die Lehrkraft die Programmierung ab und testet diese auf Funktion. Ist der Kundenauftrag gelöst, erhalten die SuS eine weitere Aufgabe, um den Prozessablauf zu optimieren. Dazu werden das Überschleifen, optimierte Bewegungsabläufe, Abstände und Verfahrwege eingeführt sowie auf Geschwindigkeiten der Achse verwiesen. Ziel dabei ist es, die Programmzeit zu optimieren. Dafür stoppen die SuS die Laufzeit ihrer Programme und diese werden an der Tafel dokumentiert. Vier verschiedene Programmierungen der SuS werden im Plenum besprochen und es wird über weitere Optimierungen diskutiert. Dabei wird Bezug zur Wirtschaftlichkeit genommen.	Arbeitsblatt: Bewegungsarten und Programmierung Software (RT-ToolBox 3) Kollaborative Roboter Beamer, Dokumentenkamera Computer, Roboterprogramme

Arbeitsblatt 1 - Montage Lagerbock GK Sondermaschinenbau

Sie sind im Unternehmen GK Sondermaschinenbau beschäftigt. Zu Ihren Aufgaben zählt neben der Produktionsüberwachung auch die Anpassung und Einrichtung von Maschinen auf neue Produkte. Ihr Unternehmen ist im Sondermaschinenbau tätig und fertigt daher im Tagesgeschäft hauptsächlich Kleinserienteile. Neue Aufträge, Umrüstungen, Programmierung und Neueinrichtungen für Werkstücke fordern viel Flexibilität Ihrer Firma und der Fachkräfte. Trotz der häufigen Umrüstungen fertigen Sie mit der Unterstützung von moderner Robotertechnik um wettbewerbsfähig in der Produktion zu sein. Damit diese Wettbewerbsfähigkeit erhalten bleibt, wurden neue kollaborative Robotersysteme angeschafft. Diese sollen die Produktion zum einen flexibler unterstützen können und zum anderen neue Möglichkeiten für die Programmierung sowie Einrichtung bieten. Damit soll Zeit bei der Programmierung eingespart werden. Durch die Einführung wird eine Einarbeitung in das neue System notwendig. Ihre Aufgabe als Fachkraft ist es nun die Automatisierung weiterer Fertigungsprozesse mit den neuen Robotersystemen durchzuführen. Als ersten Kundenauftrag erhalten Sie die kollaborative Automatisierung der Montage eines Lagerbocks aus dem 3D-Druck. Ihre Firma nutzt einen neuen thermoplastischen Kunststoff mit Kohlefaseranteil, der keine Lagerbuchse benötigt. In den Lagerbock muss eine Welle eingesetzt werden, bevor die obere Schale des Lagerbocks verschraubt wird. Die unten stehende Abbildung zeigt eine Darstellung des fertig montierten Lagerbocks.



5.5 Franka Emika

UE / LS	Wiederholgenauigkeit von Robotern
Jg. + Fach / Lernfeld	10 + Wahlpflicht Robotik
Anzahl Unterrichtsstunden	2 (exklusive Vorbereitung und Einführung in die Benutzeroberfläche)
Raum / Geräte / Lernträger	Franka Emika Robot / Niryo NED Roboter (oder andere kollaborative Roboter) beschreibbare Unterlage (evtl. fixierbar) / Stifte / greifbare Gegenstände
Handlungsziel/ -produkt	Wiederholgenauigkeit – Ausprobieren und thematisieren
Hinweise	Die Wiederholgenauigkeit von kollaborativen Robotern ist wichtig, weil sie die Qualität und Effizienz der Prozesse beeinflusst, die sie mit Menschen teilen. Kollaborative Roboter können Aufgaben übernehmen, die für Menschen anstrengend, monoton oder ergonomisch ungünstig sind. Dabei müssen sie jedoch präzise und zuverlässig arbeiten können, ohne die Sicherheit oder das Wohlbefinden der Menschen zu gefährden. Eine große Wiederholgenauigkeit ist vor allem für Anwendungen notwendig, die eine hohe Genauigkeit oder eine feinfühlige Manipulation erfordern. Zum Beispiel: • Montage: Kollaborative Roboter können bei der Montage von kleinen oder empfindlichen Teilen helfen, wie zum Beispiel in der Elektronik- oder Automobilindustrie. Die Wiederholgenauigkeit ermöglicht es ihnen, die Teile korrekt zu positionieren und zu befestigen. • Maschinenbeschickung und Teileinspektion: Kollaborative Roboter können Maschinen mit Werkstücken beladen oder fertige Produkte auswerfen und prüfen. Die Wiederholgenauigkeit sorgt dafür, dass die Maschinen optimal genutzt werden und die Qualitätsstandards eingehalten werden. • Kraftgesteuerte Anwendungen: Kollaborative Roboter können auch Kräfte auf Objekte ausüben oder erfassen, wie zum Beispiel beim Schleifen, Polieren oder Bohren. Die Wiederholgenauigkeit ermöglicht es ihnen, die richtige Kraft für das jeweilige Material anzuwenden und zu regulieren. Der Franka Emika Robot hat eine Wiederholgenauigkeit von +/- 0,1 mm. Der Niryo NED hat eine Wiederholgenauigkeit von circa +/- 0,5 mm.
Beschreibung	In dieser Lernsituation sollen die SuS eigenständig herausfinden, wie groß die Wiederholgenauigkeit der Roboter ist. Dazu erstellen sie eine „Versuchsreihe“ und überprüfen, ob verschiedene Faktoren die Wiederholgenauigkeit beeinflussen.
Foto / Abbildung	

UE / LS	Wiederholgenauigkeit von Robotern
Kompetenzen	**Prozessbezogene Kompetenzen:** Die SuS planen und führen eine Messreihe durch (möglicherweise unter Festhalten von Variablen), sie dokumentieren, vergleichen und interpretieren Ergebnisse. **Inhaltsbezogene Kompetenzen:** Die SuS verstehen das Konzept der Wiederholgenauigkeit von kollaborativen Robotern und beurteilen ihre Relevanz für verschiedene Anwendungsbereiche. Die SuS verstehen, dass in Technik und Naturwissenschaften Messergebnisse immer mit Fehlerangaben versehen sind, um die Genauigkeit der Messung anzugeben. Die SuS programmieren einen reproduzierbaren Bewegungsablauf eines kollaborativen Roboters.
Inhaltlich-methodische Voraussetzungen	Die SuS haben grundlegende Erfahrungen mit der Franka Benutzeroberfläche (Desk) gemacht und können grundlegende Bewegungen (kartesische Bewegungen, Greifer öffnen/schließen) sowie Schleifen programmieren. Die SuS haben grundlegende Erfahrungen mit den Niryo Ned Robotern gemacht und können Bewegungen und Greifbewegungen programmieren. Hierzu können die Niryo Minikurse von der Robokind Stiftung (https://robokind.de/lehrgaenge/) als Einstieg genutzt werden.

Unterrichtsverlaufsplan

Handlungsschritte (eine vollständige Handlung)	Unterrichtsverlauf	Materialien und Hinweise (Roboter, Computer, ABs, Filme, ...)
Vorbereitung	Erklären des Konzepts der Wiederholgenauigkeit und warum es für kollaborative Roboter wichtig ist. Zeigen von Beispielen für Anwendungen von kollaborativen Robotern in verschiedenen Branchen.	An dieser Stelle sollte, je nach Lerngruppe, die Bedeutung von Fehlerangaben (z. B. +/- 5cm) in den Naturwissenschaften thematisiert werden. Auch, dass in den Naturwissenschaften Messwerte grundsätzlich immer mit Fehlern behaftet sind und diese immer mit angegeben werden, könnte aufgegriffen werden.
Programmierung und Aufnehmen einer oder mehrere Versuchsreihen.	Die SuS arbeiten in Gruppen mit jeweils einem Roboter. Sie programmieren die Roboter, um eine einfache Aufgabe auszuführen, die es ermöglicht Abweichungen von der Präzision zu messen (z.B. einen Gegenstand greifen und platzieren, etwas mit einem Stift zeichnen...). Sie messen die Abweichung der Position des Roboters bei mehreren Durchläufen. Leistungsstärkere Gruppen können auch die Variable „Wiederholgenauigkeit“ unter Veränderung anderer Variablen ermitteln, wie z.B. Beschleunigung, Geschwindigkeit, das Bewegen von schwereren Gegenständen (bis zu dem erlaubten Maximalgewicht) und ob die Wiederholgenauigkeit von der Anzahl der in der Bewegung genutzten Achsen abhängt.	Maximallast Niryo: 0,3 kg Maximallast Franka: 3 kg Um die Wiederholgenauigkeit zu überprüfen, muss eine beschreibbare Unterlage auf der Arbeitsfläche so befestigt werden, dass sie nicht verrutscht.

Handlungs-schritte (eine vollständige Handlung)	Unterrichtsverlauf	Materialien und Hinweise (Roboter, Computer, ABs, Filme, ...)
Auswertung und Reflexion	Die Ergebnisse der Gruppen werden miteinander verglichen, z.B. im Plenum. Abschließend könnten die SuS, je nach Interesse und Fähigkeiten der Lerngruppe, eine kurze Präsentation vorbereiten, einen soge-nannten sales pitch, um einen der Roboter anzupreisen. Dazu können die SuS auf die spec sheets der jeweiligen Roboter schauen und sich Features suchen, die positiv sind und welche, die eher unvorteilhaft sind (Franka → Preis, Niryo → nicht robust genug für längere Aufgaben oder präzise Anwendungen). Hier können kreative Gruppen versuchen die Vorteile besonders hervorzuheben und die Nachteile umzudeuten oder als weniger bedeutsam darzustellen (siehe Beispiel sales pitch)	Material: Data sheets für beide Roboter als Infor-mation für die SuS: Franka Emika Robot: https://download.franka.de/documents/100010_Product%20Manual%20Franka%20Emika%20Robot_10.21_DE.pdf S.34 Niryo NED: https://docs.niryo.com/product/ned/v4.0.0/en/source/hardware/technical_specifications.html

Beispiel: Sales Pitch Franka Emika

Der Franka Emika Robot ist ein kollaborativer Roboter mit 7 Freiheitsgraden, der von der deutschen Firma FRANKA EMIKA entwickelt wurde. Der Franka Roboter ist ein vielseitiges und benutzerfreundliches Werkzeug für Industrie, Forschung und Bildung. Er verfügt über eine hohe Präzision, Sensibilität und Geschicklichkeit, die es ihm ermöglicht, eine Vielzahl von Aufgaben zu erledigen, wie z.B.: Maschinenbeladung, Kleben, Verpacken, Automatische Montage, Prüfung, oder Schweißen. Der Franka Roboter ist einfach zu installieren und zu bedienen. Er kann über eine intuitive Benutzeroberfläche namens “Desk” gesteuert werden, die modulare App-Bausteine für die Programmierung bietet. Er kann auch mit externen Sensoren und Software verbunden werden. Der Franka Roboter ist nicht nur leistungsfähig, sondern auch sicher und zertifiziert für den Einsatz in industriellen Umgebungen. Er hat Drehmomentsensoren an allen sieben Achsen, die ihm eine feine Anpassung an die Umgebung ermöglichen. Er kann sogar vor einem Luftballon anhalten, ohne ihn zu zerplatzen. Er hat auch eine geringe Verletzungsgefahr für Menschen, da er seine Kraft und Geschwindigkeit je nach Situation regulieren kann. Der Franka Roboter ist eine Investition, die sich schnell auszahlt. Er hat einen niedrigen Preis im Vergleich zu anderen kollaborativen Robotern auf dem Markt. Er kostet nur circa 25.000 €. Er hat auch einen geringen Energieverbrauch und Wartungsaufwand. Er kann helfen, die Produktivität, Qualität und Flexibilität zu steigern und Kosten zu senken.

Beispiel: Sales Pitch Niryo NED

Der Niryo NED Roboter ist ein kollaborativer Roboterarm mit 6 Freiheitsgraden, der von der französischen Firma Niryo entwickelt wurde. Der NED Roboter ist ein ideales Werkzeug für Bildung und Forschung im Bereich der Industrie 4.0. Er ist preiswert und erschwinglich. Er kostet nur circa 3000 Euro. Das ist viel günstiger als andere kollaborative Roboter auf dem Markt. Er ist einfach zu installieren und zu bedienen. Er hat eine intuitive Benutzeroberfläche namens Niryo Studio, die es ermöglicht, Programme mit Drag-and-Drop zu erstellen. Er kann auch mit verschiedener Software wie Python, ROS und Blockly programmiert werden. Er ist vielseitig und anpassbar. Er kann verschiedene Aufgaben erledigen, wie z.B. Pick-and-Place. Er hat ein EasyConnect System, das es ermöglicht, verschiedene Werkzeuge wie Greifer oder Kameras zu wechseln. Er kann auch mit einem Förderband oder einem Vision Set ausgestattet werden, um seine Interaktion mit der Umgebung zu verbessern. Er ist offen und erweiterbar. Das bedeutet, dass man Zugang zu vielen Bibliotheken und Ressourcen hat, um eigene Programme zu erstellen und zu verbessern. Man kann auch eigene Werkzeuge oder Sensoren hinzufügen oder den Roboter mit einem 3D-Drucker modifizieren.
Der NED Roboter ist nicht nur leistungsstark, sondern auch sicher und zertifiziert für den Einsatz in Bildungs- und Forschungsumgebungen.
Der NED Roboter hat eine Präzision und eine Wiederholgenauigkeit von 0,5 mm, was für die meisten Bildungs- und Forschungsprojekte ausreichend ist. Er hat eine Reichweite von 440 mm, was ihm eine gute Abdeckung seines Arbeitsbereichs gibt. Er ist nicht für den industriellen Einsatz konzipiert, aber er kann helfen, industrielle Szenarien zu prototypisieren und zu simulieren.

6 Robotik in der Pflege

Inhalt

6.1 Bemerkungen zur unterrichtlichen Behandlung von Robotik in der Pflegeausbildung

Jan Landherr, Carl von Ossietzky Universität Oldenburg

Der zunehmende Einsatz von technischen Innovationen im pflegerischen Handeln wird Arbeitsprozesse und Arbeitsinhalte nachhaltig verändern. Dies wirkt sich auch auf die Qualifikationsanforderungen im gesamten Pflegebildungsbereich aus. Der berufliche Umgang und die professionelle Interaktion mit neuen und innovativen Pflegetechnologien setzt die angemessene Qualifikation von Pflegefachpersonen heraus. Neben den Qualifikationswegen durch Fort- und Weiterbildungen müssen die Voraussetzungen für den Umgang mit diesen Technologien bereits in der Ausbildung gelegt werden. Die Einführung der neuen Ausbildungs- und Prüfungsverordnung für die Pflegeberufe auf Grundlage des Pflegeberufegesetzes regelt hierbei die Ausbildungsinhalte und -strukturen und formuliert Kompetenzen für die Zwischen- und Abschlussprüfungen und kann daher als Anforderungskatalog an die Auszubildenden verstanden werden.

Bei genauer Betrachtung der Kompetenzen fällt auf, dass innovative und digitale Technologien in der Ausbildungs- und Prüfungsverordnung nur in geringem Maße berücksichtigt wurden. Weiter gehen die Rahmenlehrpläne der Fachkommission nach §53 des Pflegeberufegesetzes, in denen zum Beispiel die Nutzung digitaler Informations- und Kommunikationstechnologien für den Wissenserwerb, digitale Pflegedokumentationssysteme. digitale Notfall-Informations- und Frühwarnsysteme, digitale Assistenzsysteme, Smart-Home-Technik und digitale Hilfsmittel thematisiert werden. Robotische Systeme finden hingegen keine Berücksichtigung, weshalb die Implementierung jener Themen und deren Verortung in den schuleigenen Curricula den Pflegeschulen überlassen wird. Wippich und Klein (2022, S. 824) fassen die Lage so zusammen: „Trotz des offensichtlichen Bedarfs fehlen bis dato pflegewissenschaftlich und pädagogisch legitimierte Unterrichtsinhalte und Unterrichtskonzepte zum Thema ‚Robotik in der Pflege'.

Herausfordernd ist hierbei nicht nur das Einräumen eines adäquaten zeitlichen Rahmens und die Verortung in die eigenen Lehrpläne, sondern auch das Einarbeiten in ein neues Thema, dessen inhaltliche Seite sehr stark technisch und erst einmal weniger pflegerisch geprägt ist. Dies kann zu Berührungsängsten und Unsicherheiten führen, sowohl auf Seiten der Lernenden als auch bei den Lehrenden.

Aufgabe der Lehrkräfte ist es, Kontexte zu schaffen, in denen Roboter unterrichtlich behandelt und entsprechend den curricularen Einheiten eingesetzt werden können. Während Roboter wie Pepper von der Konstruktion und Funktionalität her leicht in pflegerische Settings integriert werden können, ist die Nutzung kollaborativer Roboter(-arme) eine durchaus nicht zu unterschätzende Aufgabe: Ursprünglich nicht für die Arbeit im körpernahen Bereich vulnerabler Personen entwickelt, ergeben sich für den Unterricht dadurch besondere ethische, rechtliche, soziale und ökonomische Implikationen. Das Ziel der unterrichtlichen Thematisierung sollte daher sein:

> "Um Auszubildende dazu zu befähigen, aktuelle Entwicklungen von Robotik in der Pflege einschätzen, reflektiert, umsetzen und ggf. mitgestalten zu können, ist es erforderlich, relevante Unterrichtsinhalte zu ermitteln, die grundlegend zur Erlangung dieser Fähigkeiten beitragen." (vgl. Wippich und Klein, 2022, S. 824)

Um die Entwicklungen in der Robotik, der erste der genannten Punkte, richtig einschätzen zu können, kann im Unterricht die Geschichte der Robotik thematisiert werden: Seit wann gibt es Roboter, wo wurden sie eingesetzt und weshalb treten sie bisher vor allem in bestimmten industriellen Bereichen auf? Zudem sollten die Auszubildenden etwas über verschiedene robotische Systeme erfahren und welche Kategorien von Robotern es gibt: Gleichen sich alle „Pflegeroboter" oder gibt es für bestimmte Einsatzzwecke besondere Roboter? Was unterscheidet sie? In dieser Frage steckt bereits ein antizipatorisches Moment, wenn die Auszubildenden er- und begründen, für welche pflegerischen Tätigkeiten welche Funktionen und technische Ausführungen benötigt werden. So ist für die Lagerung von zu Pflegenden Personen ein anderer Endeffektor vonnöten als für die Unterstützung bei der Nahrungsaufnahme. Damit zusammen hängen weitere technische Fragestellungen: Woher „weiß" ein Roboter, dass er es mit einem Menschen zu tun hat und wo sich dessen Körper im Raum befindet? Die Thematisierung von Sensoren und Lösungen zur Kollisionsvermeidung soll Auszubildenden ein möglichst realistische Bild von Robotern liefern, die nicht, wie in Filmen oder Videospielen, denkende oder womöglich fühlende und intelligente Systeme sind, sondern Werkzeuge, die vom Menschen programmiert und für bestimmte Aufgaben hin konstruiert und gefertigt werden müssen.
Mit dem Wissen über unterschiedliche Roboterarten, Endeffektoren, verschiedene Sensoren und Bewegungsarten lässt sich bereits reflektieren, welchen Restriktionen der Einsatz von Robotern in pflegerischen Settings unterworfen sind.

Dieses Kompendium möchte diesen Umstand daher zum Anlass nehmen, interessierten Pflege-Lehrkräften Informationen und Anregungen für den eigenen Unterricht zu geben. Neben der zusammenfassenden Darstellung der internationalen Normen für den Einsatz von Assistenzrobotern werden auch die technischen Anforderungen an Assistenzroboter erläutert. Zudem werden verschiedene Systeme von unterschiedlichen Herstellern präsentiert.

6.2 Modell zur didaktischen Strukturierung von Lerneinheiten im Kontext der Pflegerobotik

Dani Hamade, Carl von Ossietzky Universität Oldenburg

Die Thematisierung von Robotern in der Pflege bedingt, wie zuvor bereits angedeutet, dass eine multiperspektivische Auseinandersetzung mit der Thematik erfolgt. Man interagiert in einem interdisziplinären Handlungsfeld, in welchem es neben ökonomischen und soziokulturellen Bedingungsfaktoren viele weitere Perspektiven zu berücksichtigen gilt. Aus konstruktivistischer Sicht macht es hierbei Sinn, sich mit den subjektiven Theorien von Auszubildenden auseinanderzusetzen, um eine didaktische Strukturierung der Unterrichtsinhalte vorzunehmen. So rücken Wippich und Klein in ihrer Untersuchung (2022) die subjektiven Theorien von Auszubildenden im Hinblick auf professionelle Pflege, Anwendungsbereiche robotischer Systeme und ethische Aspekte der Pflegerobotik in den Vordergrund, um Rückschlüsse auf unterrichtsrelevante Inhalte ziehen zu können (vgl. Wippich und Klein, 2022, S. 825 ff.).

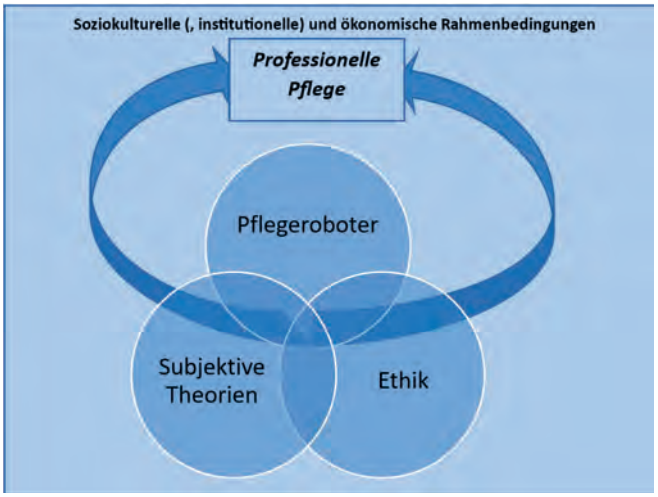


Abbildung 1: Modell zur didaktischen Strukturierung im Kontext der Pflegerobotik. In Anlehnung an die Dimensionen subjektiver Theorien von Auszubildenden nach Wippich und Klein (2022). [eigenes Werk]

Diese drei Perspektiven sollen in dem hier konstruierten Modell in den Vordergrund gerückt werden (s. Abbildung 1).

Bei den drei Perspektiven handelt es sich um unterrichtsrelevante Spannungsfelder, welche dazu genutzt werden können, die Pflegerobotik als Unterrichtsgegenstand didaktisch zu rekonstruieren. Der Perspektiven der subjektiven Theorien liegt hierbei eine zentrale Rolle inne, da diese in ständiger Wechselwirkung mit den anderen Perspektiven steht und diese maßgebend beeinflusst. So müssen subjektive Theorien im Kontext ethischer Fragestellungen in die Unterrichtsgestaltung ebenso einbezogen werden wie die Vorstellungen der Auszubildenden im Kontext des technologisch Machbaren.
Für die didaktische Strukturierung von Lernsituationen ist hierbei zunächst die Frage nach dem Verständnis von **professioneller Pflege** in den Vordergrund zu rücken. Die Pflegearbeit ist ein komplexes Unterfangen, welche insbesondere dadurch gekennzeichnet ist, dass sie äußerst situativ ist. Als ein zentrales Element professioneller Pflege kann damit einhergehend beispielsweise das hermeneutische-analytische Fallverstehen gesehen werden, welches die komplexe Fähigkeit des situativen Verstehens in Pflegesituationen bezeichnet (vgl. Dütthorn, Hülsken-Giesler und Pechuel, 2018, S. 86). Nimmt man diesen zentralen Aspekt professionellen Handelns in der Pflege und stellt ihn den einzelnen Perspektiven im Kontext der Pflegerobotik gegenüber, so ergeben sich diskussionswürdige Fragestellungen für den Unterricht. Die Schnittmengen der einzelnen Perspektiven sind hierbei als Spannungsfelder zu verstehen, müssen aber auch im Hinblick auf Synergien analysiert werden, um sinnstiftende Rückschlüsse für die didaktische Strukturierung von Lernsituationen ziehen zu können.

Bezogen auf Pflegeroboter ist hierbei zunächst die Frage der subjektiven Theorien zu klären. Welche Vorstellungen haben die Auszubildenden in der Pflege im Hinblick auf die Möglichkeiten und Limitationen von Robotern im Kontext des situativen und explorativen pflegerischen Handelns? Diese subjektiven Theorien sind der theoretischen Auseinandersetzung mit Robotern in der Pflege gegenüberzustellen, sodass Schnittmengen aber auch Dissonanzen ausgemacht und thematisiert werden können. Blickt man den subjektiven Theorien gegenüberstellend auf die Robotik und das Leistungsspektrum von Robotern, so ergeben sich zum Beispiel mögliche unterrichtsrelevante Fragestellungen im Hinblick auf die Machbarkeit im Kontext von Echtzeitanforderungen respektive im Zusammenhang zur Möglichkeit des Emotionsempfindens („Deep Acting vs. Surface Acting") von Robotern (vgl. Afflerbach, 2021, S. 15). Weitere unterrichtsrelevante Kontexte ergeben sich beispielsweise durch die Möglichkeiten und Limitationen im Hinblick auf die sogenannte „Lernfähigkeit" von Robotern. Hier sind insbesondere die Kontexte „Roboterlernen in der sozialen Welt" (z. B. Körpersprache, Dialogfähigkeit und Aufmerksamkeit), „Roboterlernen in der physikalischen Welt" (z. B. „[…] Regelung, Sensorik, Echtzeitanforderungen, Sicherheit, Bewegungsplanung und Energiemanagement." (Steil, 2019, S. 21)) und „Roboterlernen in der virtuellen Welt" (z. B. Deep Learning Verfahren und damit einhergehende Limitationen durch Datenmengen) von Bedeutung (siehe hierzu z.B. (Steil, 2019, S. 20 f.; Mainzer und Mainzer, 2016, S. 7 f.; Zech, 2020, S. 29)). Neben der rein technologischen Betrachtung im Zusammenhang zum Leistungsspektrum von Robotern ist aber auch darauf hinzuweisen, inwiefern solche Systeme in der Praxis bereits Einsatz finden, um damit einhergehend zu beleuchten, welche ökonomischen Einflussfaktoren hierbei ebenfalls zum Tragen kommen (s. hierzu z. B. Wahl, Mombaur und Schubert, 2021, S. 64).
Hat man ein Bewusstsein dafür geschaffen, was auf dem Gebiet der Pflegerobotik technisch möglich ist, so kann man sich mit der ethischen Perspektive als bewertende Perspektive im Hinblick auf das professionelle pflegerische Handeln auseinandersetzen. Auch hier sind subjektive Theorien zu berücksichtigen. Welche ethischen Aspekte stellen die Auszubildenden im Kontext der Robotik in den Vordergrund und wie kann man an diese anknüpfen, um einen ethisch verantwortlichen Umgang mit Robotern in der Pflege zu diskutieren? Wippich und Klein (2022) haben im Zuge ihrer Untersuchung folgende Schwerpunkte auf Seiten der subjektiven Theorien von Auszubildenden im ethischen Zusammenhang herausgestellt:

„
1. Beziehungsgestaltung durch robotische Systeme
2. Roboter als mögliche Interaktionspartner und Informationsgeber
3. Beschäftigungs- und Aktivierungsangebote durch Robotik
4. Übernahme von Körperpflege durch Robotik
5. Unterstützung für zu Pflegende, Angehörige und Pflegende
6. Sicherheit des Patienten
7. Datenschutz
8. Ersatz von Pflegepersonal durch Robotik
9. Entwicklung im und Auswirkungen auf den Pflegeberuf

" (Wippich, Klein, 2022, S. 829)

Es zeigt sich, dass sich hier durchaus Schnittmengen mit der Auseinandersetzung der technologischen Perspektive auf Pflegeroboter ergeben. Insbesondere die Aspekte der Sicherheit, Roboter als Interaktionspartner oder Beziehungsgestaltung durch robotische Systeme docken unmittelbar an die vorherige Auseinandersetzung mit der technischen Machbarkeit an.

Als geeignete Anknüpfungspunkte an diese subjektiven Theorien schlagen Wippich und Klein im Zusammenhang zur unterrichtlichen Auseinandersetzung den Bezug zur Stellungnahme des Deutschen Ethikrates im Hinblick auf Pflegeroboter und zu dem MEESTAR-Modell her, welche hier ebenfalls als geeignet angesehen werden (vgl. Wippich und Klein, 2022, S. 830).
Diese Schnittstellen zwischen ethischer und technologischer Perspektive sind auf das hermeneutisch-analytische Fallverstehen pflegerischen Handelns als zentrales Element zurück zu beziehen, um das Wirkungsgeflecht aus den verschiedenen Perspektiven zu einer professionsbezogenen Einheit zu schließen. Es zeigt sich, dass die Wirkungsdimension eng miteinander zusammenhängen und sich diskussionswürdige Spannungsfelder während der Durchleuchtung der einzelnen Perspektiven in Verbindung mit der Auseinandersetzung im Kontext subjektiver Theorien von Auszubildenden ergeben können. Das Modell bietet demnach eine Möglichkeit, die Pflegerobotik im Kontext der Unterrichtsgestaltung, bei gleichzeitiger Berücksichtigung der Prinzipien professionellen pflegerischen Handelns, aus konstruktivistischer Sichtweise sinnstiftend und vor allem umfassend zu integrieren.

6.3 Technische Grundlagen der Pflegerobotik

Yves Korte-Wagner, Tobias Neiß-Theuerkauff, Frank Wallhoff, Jade Hochschule Studienort Oldenburg

Das Thema Robotik in der Pflege unterliegt gegenüber der gewöhnlichen, industriell genutzten Robotik zusätzlichen Herausforderungen. Die dort eingesetzten Systeme sollen unmittelbar mit Menschen interagieren. Unabhängig davon, ob es sich bei dem interagierenden Menschen um die pflegende oder um die gepflegte Person bzw. Patientin oder Patient handelt. Es gibt also mindestens zwei zu berücksichtigende Anwenderebenen an die das robotische Assistenzsystem angepasst werden muss. Akzeptanz, Ethik und Sicherheit spielen dabei eine große Rolle. Der demografische Wandel, der Fachkräftemangel und die besonderen Arbeitsbedingungen in der Pflege sind nur einige Gründe, warum Robotik in der Pflege unterstützen kann (vgl. Wallhoff, Vox und Theuerkauff, 2019).

Ein technisches Assistenzsystem kann dabei viele verschiedene Formen annehmen. Je nach Aufgabe, Umgebung und Umsetzung des Assistenzsystems, müssen auch entsprechende Sicherheitsfunktionen vorhanden sein, die eine Gefährdung des Menschen im besten Fall ausschließen oder minimieren. Daher unterliegt jedes Assistenzsystem einer Risikobeurteilung. Welche Risiken entstehen können und unter welchen Bedingungen eine Sicherheitsfunktion entwickelt und getestet werden muss, wird von den internationalen Normen vorgegeben.

Die Anforderungen und verfügbaren Technologien zeigen den anwendenden Personen auch die Grenzen der Assistenzroboter auf. Das Wissen über diese Limitierungen hilft das Verhalten eines Systems zu verstehen und auf unvorhersehbares Verhalten richtig zu reagieren. Bei ungünstigen Situationen könnte es sonst zu Gefährdungen kommen, was vermieden werden muss. In den folgenden Kapiteln werden verschiedene Sicherheitstechniken betrachtet und vorgestellt, die in Assistenzrobotern typischerweise wiederzufinden sind.

6.3.1 Internationale Normen

Die Sicherheitskonzepte von Maschinen, Robotern oder Assistenzsystemen unterliegen, wie es auch bei anderen Maschinen oder Geräten der Fall ist, den internationalen Normen. Angefangen von der allgemeinen Maschinensicherheit, der Sicherheitsanforderung von Industrierobotern, Flurförderfahrzeugen und Servicerobotern, bis hin zum sichereren Betrieb von kollaborierenden Robotern und Assistenzsystemen.

In Abbildung 2 ist eine Übersicht der relevanten Normen für den Bereich Robotik dargestellt. Zu sehen ist, dass es sich bei einigen DIN-Normen um Ableitungen bzw. Ergänzungen einer vorigen Norm handelt. Bekanntermaßen wird im Bereich der Pflege kein Industrieroboter verwendet. Dennoch ist es sinnvoll, die Ergänzungen und Ableitungen der allgemein gültigen Normen für Industrieroboter zu betrachten, da viele der Sicherheitsanforderungen auch für Assistenzsysteme gelten.

Die Normen werden vor finaler Herausgabe bereits als Entwurf zur Verfügung gestellt. Finalisierte Normen können weiterentwickelt und Ergänzt werden, was den Jahreszahlen hinter der Normbezeichnung zu entnehmen ist.

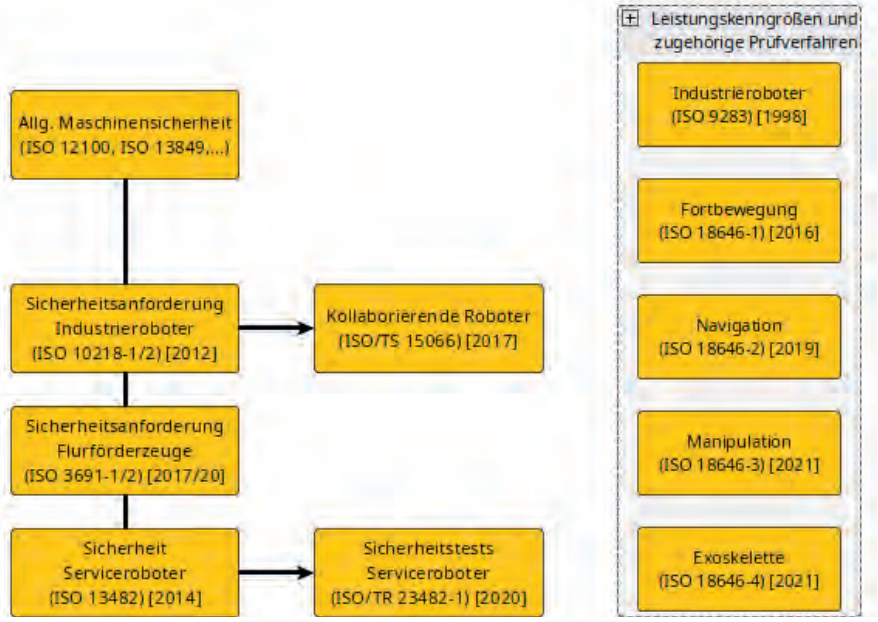


Abbildung 2: Übersicht der relevanten Normen im Bereich Robotik [eigenes Werk]

Im Folgenden werden zunächst die allgemeinen Sicherheitstechniken aus Sicht der internationalen Normen betrachtet, die für Industrieroboter und Assistenzsysteme gelten. Der besondere Fall des kollaborierenden Betriebs wird danach im Abschnitt *Mensch-Roboter-Kollaboration* genauer betrachtet.

6.3.1.1 Mögliche Gefährdungen durch Assistenzroboter

Die Norm DIN ISO 13482 befasst sich mit den Sicherheitsanforderungen und mit den ausgehenden Gefährdungen von Assistenzrobotern (vgl. DIN e.V. (Hrsg.), 2014).

Energiespeicherung und Versorgung
Bei batteriebetriebenen Assistenzrobotern sollte stets die Betriebsanleitung herangezogen werden. Bspw. sollte der Anwender sich bewusst machen, ob das System bei vollem Akku von der Ladestation genommen werden muss oder ob die Akkus in irgendeiner Weise gewartet werden müssen. Es existieren auch Assistenzroboter, die zum Laden in einen bestimmten Betriebsmodus geschaltet werden müssen. Die Ladekontakte am Roboter sowie an der Ladestation sollten vor unbeabsichtigtem Kontakt mit dem Menschen geschützt sein. Auch die Art in welcher Umgebung (Luftfeuchtigkeit, Raumtemperatur, drinnen oder draußen) geladen werden darf, könnte eingeschränkt sein. Im Bezug auf die Pflege sollte der Ladevorgang nicht in einem Aufenthaltsraum stattfinden, sondern besser in einem separaten Raum, der nur befugtem Personal zugänglich ist. Es ist davon auszugehen, dass die gepflegten Personen bzw. Patienten keine Sicherheitseinweisung erhalten haben und Gefährdungen durch falsche Bedienung hervorgerufen werden könnten.

Einschalten und Wiederaufnahme des Normalbetriebs
Ebenfalls sollte sich bewusst gemacht werden, was beim Einschalten des Assistenzroboters in Bezug auf eine mögliche Anfahrtsbewegung passiert, oder auch wie sich das System bei Rückstellung eines Sicherheitsstopps oder Not-Halte-Stopp verhält. Der/die Anwender*innen dürfen nicht von plötzlichen Bewegung überrascht werden.

Form des Roboters
Je nach Einsatzgebiet des Roboters müssen scharfe und spitze Kanten vermieden werden. Je nach Konstruktion können gerade bei beweglichen Teilen Quetschungen o.ä. auftreten. Dies lässt sich nicht immer zuverlässig bei der Konstruktion vermeiden.

Emissionen und fehlende Wahrnehmung durch den Menschen
Der Assistenzroboter darf im Betrieb nicht so laut werden, dass Personen im Umfeld beeinträchtigt werden. Andererseits sollte bspw. ein Transportsystem akustisch auf sich aufmerksam machen, damit es von anderen Menschen wahrgenommen werden kann und die Gefahr von möglichen Zusammenstößen verringert wird.

Lichtsignale können bei Personen mit eingeschränktem Hörvermögen zusätzlich hilfreich sein. Hier muss im Rahmen der Pflege ein besonderes Bewusstsein dafür entwickelt werden, welche Personen mit dem Assistenzroboter in Kontakt treten und wie diese den Roboter wahrnehmen.

Umgebungsbedingungen
Nicht jeder Assistenzroboter wird für alle prinzipiell möglichen Umgebungsbedingungen entwickelt. Staubige oder sandige Umgebungen könnten Luftschlitze verstopfen oder Lüfter blockieren. Auch der Transport oder das Hantieren mit Flüssigkeiten kann zu Gefährdungen führen, sollte der Assistenzroboter dafür nicht geeignet sein. Es muss im Voraus festgelegt werden, unter welchen Umgebungsbedingungen der Roboter agieren soll und ob der Roboter dafür geeignet ist. Auch Hinweise in der Betriebsanleitung im Bezug auf Reinigung und Wartung sollten zu Kenntnis genommen werden.

Lokalisierungs- und Navigationsfehler
Ein mobiler Assistenzroboter nimmt seine Umgebung technisch nicht unbedingt so war, wie man es intuitiv erwartet. Je nach Einsatzort muss überprüft werden, ob das System auch dafür ausgelegt ist, um zu funktionieren. Als Beispiel sollte vor dem Betrieb eines Systems in einem Obergeschoss überprüft werden, wie das System mit Treppen umgeht bzw. ob es diese überhaupt erkennt, um ein mögliches Herabfallen zu vermeiden. Im Kapitel *Self Location and Mapping (SLAM)* wird etwas genauer auf die Techniken zur Wahrnehmung der Umgebung eingegangen.

Falsche autonome Entscheidungen
Es kann nicht ausgeschlossen werden, dass ein autonomes System falsche Entscheidungen trifft. Objekte könnten falsch identifiziert oder Personen nicht erkannt werden. Bei einer Objektidentifizierung wird meistens auf künstliche Intelligenz (siehe *Mustererkennung & Künstliche Intelligenz*) zurückgegriffen. Es ist abhängig davon, wie gut die Algorithmen trainiert worden sind. Sicherheitsrelevante Objekte werden daher auch meist gesondert, über optische Marker (Bar-Code, QR-Code o.ä.), markiert, um eine erfolgreiche Identifizierung sicher zu stellen. Auch bei mobilen Systemen, die autonom durch den Raum navigieren, stellt sich die Frage, wie sie auf Hindernisse reagieren. Es gibt die Möglichkeit anzuhalten ohne das Ziel zu erreichen oder alternativ ein Ausweichmanöver zu vollziehen. Je nach System kann Letzteres aufgrund von rutschigen oder unebenen Böden zu Gefährdungen führen. Welche Entscheidung das System bei bestimmten Ereignissen trifft, sollte in den Benutzerinformationen wiederzufinden sein.

6.3.1.2 Risikobeurteilung und Leistung einer Sicherheitstechnik

Je nach Risiko einer möglichen Gefährdung unterteilen diese Stufen den Zuständigkeitsbereich der Sicherheitstechnik. Wie in Kapitel Verschiedene Arten von Systemen dargestellt wird, können Assistenzsysteme viele verschiedene Formen haben, sodass es stark abhängig von der Aufgabe und der Umgebung ist, ob eine Sicherheitstechnik bereits bei der Konstruktion oder erst bei der Anwendung vor Ort notwendig ist. Im niedrigschwelligen Fall kann eine Sicherheitstechnik auch aus Warnhinweisen bestehen.

1. **Unmittelbare Sicherheitstechnik**: Es handelt sich dabei um Sicherheitstechniken, die bereits bei der Konstruktion des Roboters angewendet werden, um Gefährdungen zu beseitigen oder möglich stark einzuschränken.
2. **Mittelbare Sicherheitstechnik**: Dies sind Sicherheitstechniken, die in der Arbeitsumgebung angewendet werden. Das können z.B. Schutzkabinen oder Lichtschranken sein. Die mittelbaren Sicherheitstechniken müssen angewendet werden, wenn nicht alle Gefahren bereits bei der Konstruktion beseitigt werden konnten.
3. **Hinweisende Sicherheitstechnik**: Hier wird der Anwender über mögliche Restrisiken hingewiesen. Dies kann durch Hinweise in der Betriebsanleitung, durch Anbringung von Hinweisschildern oder durch optische bzw. akustische Warnsignale umgesetzt werden.

Wenn eine Gefährdung nicht direkt bei der Konstruktion des Systems minimiert werden kann, so muss die nächste Stufe angewendet werden, um die Gefährdung durch ergänzende Schutzmaßnahmen weiter zu verringern. Dieses kann zum Beispiel durch externe Peripheriegeräte im Einsatzgebiet des Systems erfolgen. In der letzten Stufe muss der Benutzer über die verbleibenden Risiken informiert werden. Gerade die hinweisenden Sicherheitstechniken sollten den anwendenden Personen des Systems vertraut sein, um zu entscheiden, ob die verbleibenden Restrisiken zumutbar sind.

Sicherheitstechniken müssen verlässlich funktionieren und ausfallsicher sein. Die Norm EN ISO 13849-1 definiert hierfür fünf sog. *Performance-Level* (PL) von *a* bis *e*. Die PL sind durchschnittliche Zeiten, in der

eine Sicherheitstechnik ausfallen könnte ($MTTF_D$[1]) (vgl. DIN e.V. (Hrsg.), 2016a, S. 16).
Um die Ausfallsicherheit einer Sicherheitstechnik zu erhöhen, können z.B. redundante Geber (Sensoren) eingesetzt werden. Das heißt, dass eine physikalische Größe von zwei unabhängigen Sensoren gemessen wird. Die Werte der Sensoren werden verglichen und müssen übereinstimmen, um die Plausibilität der Werte sicherzustellen.

Zur Klärung der Fragestellung, welche Sicherheitstechnik welchen Performance-Level (Beitrag zur Risikominimierung) besitzen muss (PL_r[2]), wurde in den internationalen Normen ein Risikograph definiert (Abbildung 3), mit dem ein Risiko nach dem Schweregrad des Schadens (S), der Schadenshäufigkeit (F) und der Schadensvermeidung (P) einem Performance-Level zugewiesen werden kann (vgl. Maier, 2016, S. 168; DIN e.V. (Hrsg.), 2016a, S. 60 ff.).

1 Mean Time to Failure (danger): mittlere Zeit bis zu einem gefahrbringenden Ausfall
2 Performence-Level required

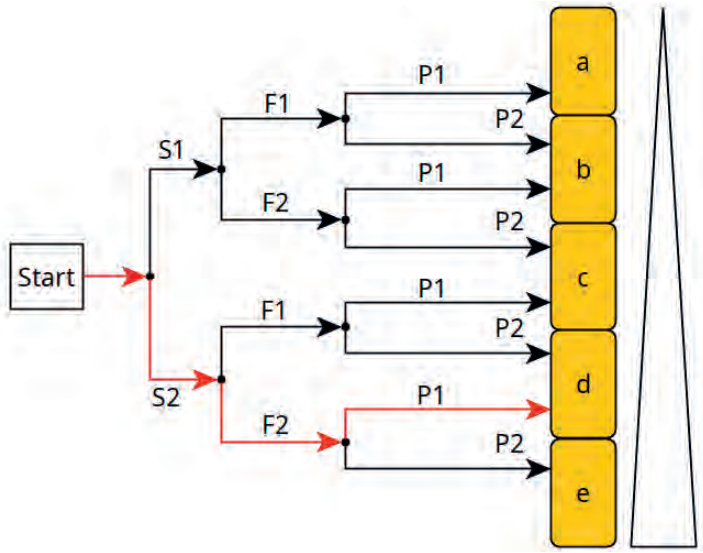


Abbildung 3: Risikograph nach DIN13849-1 zur Ermittlung des PL_r
Der Pfeil rechts gibt eine abnehmende Schwere des Risikos von unten nach oben an.

Parameter	Beschreibung
S1	leichte, reversible Verletzungen
S2	schwere, irreversible Verletzungen oder Tod
F1	selten bis weniger häufig und/oder die Zeit der Gefährdungsexposition ist kurz
F2	häufig bis dauernd und/oder die Zeit der Gefährdungsexposition ist lang
P1	Vermeidung möglich, unter bestimmten Bedingungen
P2	Vermeidung kaum möglich

In der Abbildung 3 ist als Beispiel ein Pfad eingezeichnet: S2 → F2 → P1. Ein Risiko kann eine schwere irreversible Verletzung hervorrufen, dessen Eintrittswahrscheinlichkeit häufig ist und nur unter bestimmten Bedingungen vermieden werden kann. Die Zuverlässigkeit der Sicherheitstechnik muss dem PL_d entsprechen.

6.3.1.3 Das Stillsetzen von Systemen nach Stopp-Kategorien

In den Normen sind ebenfalls verschiedene Haltevorgänge nach Kategorien festgelegt. Dabei ist es wichtig zu wissen, wann bzw. wie eine Maschine anhält und wann und unter welchen Umständen eine Maschine die Arbeit wieder aufnimmt. Ein unerwarteter Anlauf einer Maschine kann wiederum zu einer Gefährdung führen. Grundsätzlich gibt es zwei Arten von Haltevorgängen: beim Auslösen einer Sicherheitsfunktionen (Sicherheitshalt) oder beim Betätigen eines Not-Aus-Schalters (Not-Halt).

In der DIN EN ISO 13850 werden drei Stopp-Kategorien definiert (vgl. DIN e.V. (Hrsg.), 2016b, S. 12 ff.):

Kategorie 0
Bei der Stoppfunktion nach der Kategorie 0 wird die gesamte Energiezufuhr zu den Antriebselementen sofort unterbrochen. Man spricht von einem *ungesteuerten Stillsetzen*. Ungesteuert, da beim Unterbrechen der Energiezufuhr nicht mehr aktiv gebremst werden kann. Je nach Geschwindigkeit, Last und Trägheit trudelt der Roboter aus. Dabei ist es egal, ob es sich bei der Bewegung um einen Roboter-Arm oder ein fahrendes Transportsystem handelt.

Kategorie 1
Gegenüber dem ungesteuerten Stillsetzen der Kategorie 0 wird bei der Kategorie 1 ein *gesteuertes Stillsetzen* angewendet. Das heißt, dass die Energiezufuhr zu den Aktoren erst nach Erreichen der Ruhelage getrennt wird. Dies ist sinnvoll, wenn das System aktiv gebremst werden soll, um den Anhaltweg zu verkürzen.

Kategorie 2
Das System wird, wie in Kategorie 1, gesteuert stillgesetzt. Die Energiezufuhr zu den Antrieben bleibt im Anschluss jedoch bestehen.

Welche Stopp-Kategorie angewendet werden soll, hängt in erster Linie davon ab, wie sich potentielle Risiken minimieren lassen. Ein Stopp nach Kategorie 0 kann gefährlich sein, wenn ein Roboter-Arm bspw. zu dem Zeitpunkt der Aktierung etwas transportiert. Die Abschaltung der Energiezufuhr kann bewirken, dass ein Gegenstand fallengelassen wird oder der gesamte Arm erschlafft und herunterfällt. Es darf also durch die Trennung der Energiezufuhr keine zusätzliche Gefährdung entstehen. Andererseits kann ein Stopp nach Kategorie 0 auch gewünscht sein. In der Industrie können die Roboter so mit verschiedensten Werkzeugen ausgestattet werden. Angefangen vom einfachen Greifer bis hin zu Sägen, Bohrmaschinen oder Schweißgeräten. Ein Kategorie-0-Stopp kann daher erforderlich sein, um die Werkzeuge stromlos zu schalten. Auch wenn eine Person vom Roboter eingequetscht werden sollte, ist es notwendig, dass die Motoren keine weitere Kraft mehr erzeugen und blockierende Getriebe freigeschaltet werden. Die Stopp-Kategorien sind zwar aus industriellen Anwendungsfeldern entstanden. Sie gelten aber auch für alle anderen Assistenzsysteme (vgl. DIN e.V. (Hrsg.), 2012, S. 15f).

Not-Halt
Der Not-Halt wird beim Auslösen des Not-Halt-Schalters manuell vom Benutzer ausgelöst. Der Schalter muss immer gut erreichbar sein und wird in der Regel nur im Notfall betätigt. Die Rückstellung des Schalters kann nur manuell durch den Benutzer erfolgen, bis zur manuellen Rückstellung des Schalters ist die Maschine nicht einsatzbereit. Die Maschine muss nach Stopp-Kategorie 0 oder 1 halten.

Sicherheitshalt
Ein Sicherheitshalt kann manuell oder automatisch, falls eine Sicherheitsfunktion anschlägt, ausgelöst werden. Die Rückstellung geschieht manuell oder automatisch. Die Maschine muss mind. eine Stopfunktion nach Stopp-Kategorie 0 oder 1 besitzen. Sie kann auch nach Stopp-Kategorie 2 halten, solange der Stillstand überwacht wird (siehe Kapitel *Sicherheitsbewerteter überwachter Halt*).

6.3.1.4 Mensch-Roboter-Kollaboration

Eine Maschine kann nur genau vorgegebene Abläufe vollautomatisiert wiederholt durchführen. Doch existieren Aufgaben, die nur durch einen Mensch mit seinen speziellen motorischen Fähigkeiten gelöst werden können. Der Mensch kann auch auf unvorhersehbare Situation intuitiv und situationsbezogen reagieren. Einer Maschine hingegen muss genau und in kleinen Schritten ein Ablaufprozess einprogrammiert werden. Die Wiederholgenauigkeit, Kraft, Geschwindigkeit und Präzision eines Roboters sollen mit der Flexibilität, Intelligenz und Kreativität des Menschen kombiniert werden. Der Mensch und die Maschine besitzen einen gemeinsamen Arbeitsraum bzw. stehen im direkten Kontakt miteinander.

Da die direkte Zusammenarbeit zwischen Mensch und Maschine immer mehr zunimmt, müssen auch die Normen an neue mögliche Gefährdungen und Sicherheitstechniken angepasst werden. Nicht-Kollaborative Industrieroboter können durch Kabinen oder Schutzzäune vom Menschen isoliert werden. Dieser Schutz fällt bei der Mensch-Roboter-Kollaboration weg.

Seit 2006 haben die internationalen Normen das Thema kollaborierender Roboter aufgegriffen und in der Norm DIN EN ISO 10218-1/2 erstmals Anforderungen festgelegt. Die Norm wurde mehrmals erneuert. Ebenfalls wurde ergänzend eine technische Spezifikation entworfen (EN ISO/TS 15066), in der u.a. auch zulässige Kräfte zwischen Mensch und Maschine definiert wurden, um echte Kollaboration zu ermöglichen.

Welche Sicherheitstechniken am besten angewendet werden können, hängt stark von der Art der Interaktion mit dem Roboter zusammen (Maier, 2016, S. 179ff; vgl. Bendel und Daimler und Benz Stiftung, 2018, S. 6).

Ko-Existenz
Bedeutet, dass Mensch und Roboter getrennt von einander arbeiten und nicht ständig im Kontakt stehen.

Kooperation
Mensch und Roboter verfolgen ein gemeinsames Ziel. Aktionen und Arbeitsvorgänge von Roboter und Mensch sind klar voneinander getrennt.

Kollaboration
Mensch und Roboter sind im ständigen Kontakt. Sie bearbeiten gemeinsam eine Aufgabe. Hier sollen die Vorteile eines Menschen und einer Maschine kombiniert werden.
Im weiteren Verlauf wird kurz auf die Anforderung für einen kollaborierenden Betrieb eingegangen.

Sicherheitsbewerteter überwachter Halt
Ein sicherheitsbewerteter überwachter Halt ist eine Funktion, bei der der Roboter anhalten muss, sobald bspw. eine Person in den Aktionsraum des Roboters eindringt, wenn es sich um die Interaktionsform der Ko-Existenz handelt. Alternativ kann erst die Geschwindigkeit bis zum Stopp reduziert werden (Stopp-Kategorie 2), welches einem Sicherheitshalt entspricht. Die Überwachung des Kollaborationsraumes kann mit Hilfe von Kameras oder Lichtschranken umgesetzt werden. Dabei muss die Sicherheitstechnik ständig prüfen, ob das System wirklich anhält und stehenbleibt. Sollte die entsprechende Sicherheitstechnik zur Überwachung des Stillstands feststellen, dass das System nicht still steht, so muss die nächste Stopp-Kategorie angewendet werden und das System stromlos schalten. Verlässt die Person den Arbeitsraum wieder, so kann der Roboter automatisch seine Aufgaben wieder aufnehmen. (vgl. DIN e.V. (Hrsg.), 2012, S. 21f)

Handführung
Die Handführung eines Roboters wird meist angewendet, um dem Roboter Bewegungen beizubringen. Es muss dafür eine Vorrichtung am Endeffektor (Werkzeug am Ende eine Roboterarms) angebracht sein, die auch mit einem Not-Halt und Zustimmschalter ausgestattet ist. Durch die Betätigung des Zustimmschalters lässt sich der Roboter bewegen. Die Bewegungen werden vom Roboter aufgenommen und können anschließend abgespielt werden (Teach-and-Playback). Ein gutes Beispiel dafür ist das System *Robert der Firma KUKA*.

Geschwindigkeits- und Abstandsüberwachung
Die Verwendung der Geschwindigkeits- und Abstandsüberwachung sieht vor, einen festgelegten Abstand und eine Geschwindigkeit zu der Bedienperson einzuhalten. Sobald die festgelegte Geschwindigkeit überschritten oder der Abstand unterschritten wird, muss ein Sicherheitshalt durchgeführt werden.

Leistungs- und Kraftbegrenzung
Die Leistungs- und Kraftbegrenzung (PFL[1]) ist eine Technik, in der die von den Aktoren aufgenommene Energie begrenzt wird. Das heißt, dass auftretende Kräfte einen Grenzwert nicht überschreiten dürfen. Eine Kraftbegrenzung kann passiv, über einen federnden Greifer oder eine Polsterung, umgesetzt werden. Bei einer aktiven Kraftbegrenzung werden Kraft- und Drehmomentsensoren eingesetzt.

Die Grenzwerte wurden vom Fraunhofer-Institut für Fabrikbetrieb und -automatisierung (IFF) experimentell ermittelt und in die technische Spezifikation für kollaborierende Roboter (DIN ISO/TS 15066) übernommen. Dafür wurden relevante Punkte am Körper definiert (Abb. X). Eine Messgerät übte mit steigender Kraft Druck auf die Körperpunkt aus. Der Proband konnte über eine Fernbedienung das Gerät bei der Schwelle zum Schmerzempfing anhalten. Es wurden verschiedene Personengruppen (männlich, weiblich) und mit unterschiedlichen Body-Mass-Indexen für die Untersuchung herangezogen (vgl. Behrens und Pliske, 2019).

1 Power and Force Limiting (PFL)

Das entstandene Körpermodell hat aber auch seine Grenzen. Es wird davon ausgegangen, dass die Kontaktfläche zwischen Mensch und Maschine unelastisch und eben ist. Die Kontaktfläche hatte eine Größe von 1,4 x 1,4 cm. Bei einer kleineren oder anderen geometrisch geformten Kontaktfläche können die Grenzwerte abweichen, weil die über die Kontaktfläche auf den Körper ausgeübte Kraft anders verteilt sein kann. Die ermittelten Werte sollen als Richtwerte für die Risikobeurteilung im Sinne der Normen dienen (vgl. DIN e.V. (Hrsg.), 2017, S. 30ff).

Es ist daher zu empfehlen, solche Mensch-Maschinen-Kontakte am eigenen Körper zu testen und die Zumutbarkeit unter realen Bedingungen selbst zu entscheiden. Dies wird an dem Umstand klar, dass ein spitzer Gegenstand schmerzhafter ist als ein stumpfer Gegenstand bei gleicher Kraft, weil das Verhältnis von Kraft zu Fläche größer ist.

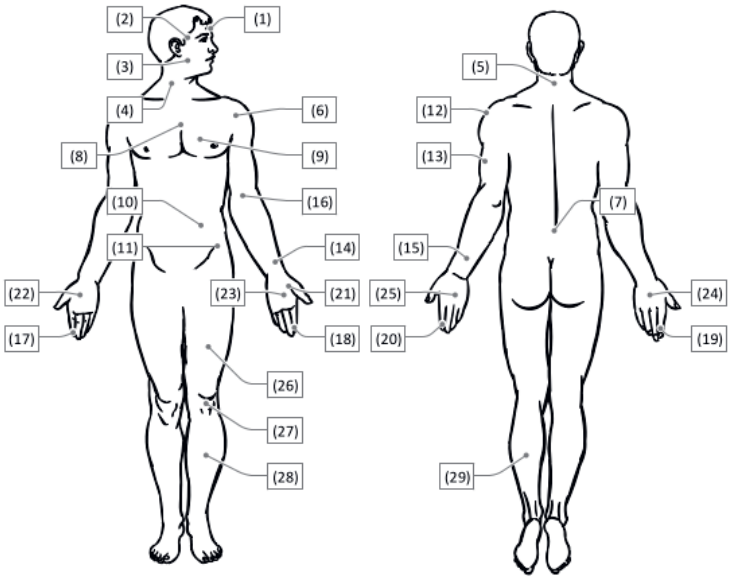


Abbildung 4: Punkte am Körper zur Ermittlung der biomechanischen Schmerzgrenzen (Behrens und Pliske, 2019, S. 6)

6.3.2 Technische Herausforderungen an Assistenzrobotern

Um zu verstehen, wieso ein Roboter bestimmte Handlungen durchführt oder auf bestimmte Situationen reagiert, sollte bei der Verwendung eines Assistenzroboters geklärt werden, wie dieser die Umgebung wahrnimmt und verarbeitet. Gerade bei Assistenzrobotern, die zum Teil autonom handeln oder navigieren, macht es durchaus Sinn, "die Welt durch die Augen eines Roboters" zu sehen und nachzuvollziehen, auf welchen Daten seine Entscheidungen beruhen.

Um selbständig Aufgaben zu übernehmen, muss ein System über diverse Sensorik verfügen und die Daten interpretieren. Sensoren, wie Kameras, Mikrofone oder Laserscanner, überwachen die Umwelt. Es muss dabei berücksichtigt werden, dass es sich bei den Daten für den Roboter um reine Messdaten und Zahlen handelt (Winkel, Längen, Bildpunkte oder Spannungen), die über einen Computer und mathematische Verfahren gedeutet werden müssen.

Wie aber kann ein Roboter, z.B. aus den Bildpunkten, ein Gesicht oder irgendein anderes Objekt erkennen? Die Antwort lautet, dass ein Roboter dies mit Methoden der so genannten Mustererkennung als Teilgebiet der künstlichen Intelligenz kann, bei der gewisse Merkmale und Muster (sog. Features) aus einem Bild extrahiert und mit bereits vorhandenen Merkmalen vergleichen werden. Das heißt auch, dass der Roboter auf Modelle, Fakten oder Daten zurückgreifen muss, die ihm zuvor in irgendeiner Form einprogrammiert wurden. Auch das selbständige Navigieren durch Räume bedarf der Wahrnehmung und Interpretation der Umgebung, um Routen zu planen und Ziele zu erreichen (siehe *SLAM*). Die Eigenschaften solcher Systeme ergänzen sich immer mehr zu einem kognitiven Verhalten.

Ein kognitives System ist ein System, das in der Lage ist, komplexe Aufgaben auszuführen, die normalerweise menschliche kognitive Fähigkeiten erfordern, wie z. B. Sprachverarbeitung, visuelle Wahrnehmung, Entscheidungsfindung und Problemlösung. Dabei werden meist *Mustererkennung & künstliche Intelligenz* eingesetzt, um die Aufgaben zu lösen. Eine Schematische Darstellung ist in Abbildung 5 zu sehen.

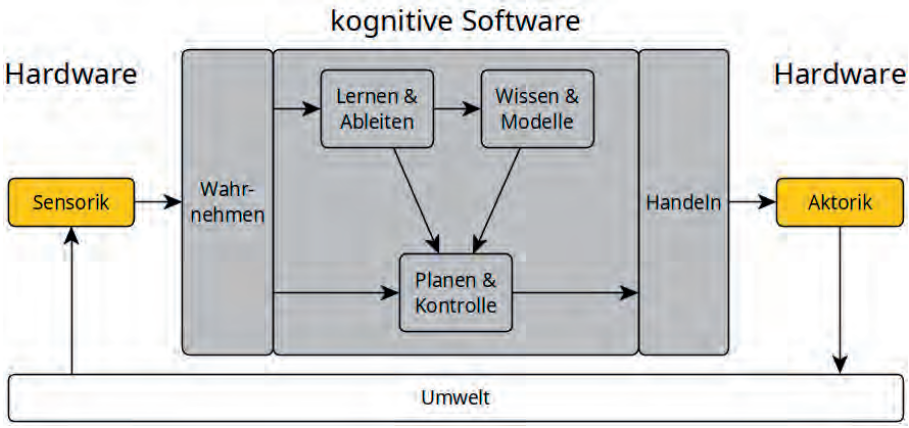


Abbildung 5: Schematische Darstellung eines kognitiven Systems [eigenes Werk]

6.3.2.1 Self Location And Mapping (SLAM)

Das "Simultaneous Localization and Mapping" (SLAM) ist ein Verfahren, bei dem es darum geht, eine Karte der Umgebung zu erstellen, und gleichzeitig die Position innerhalb dieser Karte zu bestimmen. Das Verfahren ist eine Grundvoraussetzung für das autonome Navigieren. In den meisten Fällen wird die Umgebung optisch, z.B. über ein sog. Light Detection and Ranging Sensor (Lidar), erfasst (Abb. X). Lidar-Systeme messen die Entfernung zu einem Punkt im Raum, z.B.indem ein rotierender Spiegel einen Laserstrahl seitlich ablenkt, nicht nach unten oder oben (2D-Abtastung). Trifft der Strahl auf ein Objekt, so wird der Strahl reflektiert. Das Lidar-System misst dabei die vergangene Zeit zwischen dem Aussenden des Strahls und der Reflektion. Dadurch entsteht ein schmales Abbild vom Raum. Lidar-Systeme führen diesen Prozess mehrere Male pro Sekunde durch (Abbildungen 6 und 7).

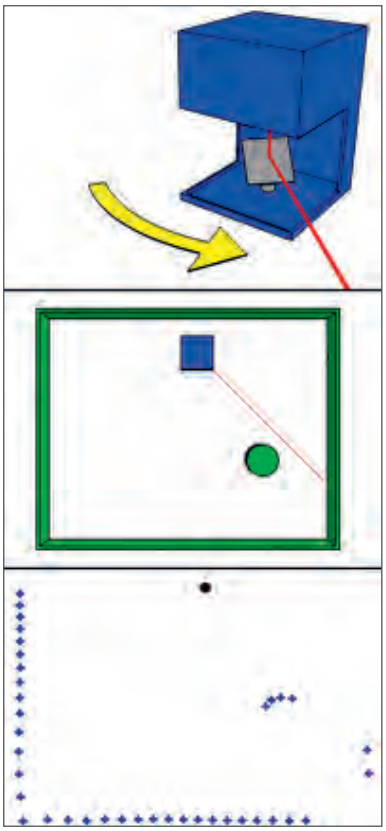


Abbildung 6: Darstellung der 2D-Abtastung eines Lidars[2]

Abbildung 7: Ein Lidar in einem mobilen Roboter von der Firma SICK[1]

1 S. Winkvist, Mobile robot with a LIDAR Sensor. 2008. Zugegriffen: 21. April 2023. [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/File:LIDAR_equipped_mobile_robot.jpg

2 Mike1024, Animation der 2D Abtastung LIDAR. 2008. Zugegriffen: 29. März 2023. [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/File:LIDAR-scanned-SICK-LMS-animation.gif

Bewegt sich ein Roboter durch den Raum, müssen die einzelnen Aufnahmen des Lidar von einem Prozessor zu einer Karte zusammengesetzt werden. Dazu muss das System bestimmen, wie weit sich der Roboter bewegt hat. Bei radbetriebenen Plattformen kann, bspw. über ein Inkrementalgeber (Verweis noch oben vom gesamten Kompendium), die Drehung der Räder nachvollzogen und Rückschlüsse auf die geänderte Position berechnet werden, die sog. Odometrie.

Rutschige oder unebene Böden können die Odometrie verfälschen. Es gibt Materialien, die Licht gar nicht (Glas) bis vollständig (Spiegel) reflektieren oder brechen. Bodentiefe Fenster könnten fälschlich als Durchgang wahrgenommen werden. Auch Reflektionen können die Messung verfälschen. Meist sind diese Scanner nahe am Boden angebracht und würden von einem Tisch oder Stuhl nur die Beine als Punkte auf der Karte erkennen, nicht aber die höher gelegene Tischkante. Ebenfalls problematisch ist das Ausschalten und Versetzen des Systems. Hier müssen die Position und Orientierung auf der Karte wieder korrigiert werden. Ein Sensor ist zudem nie zu 100 % präzise und besitzt eine nicht zu vermeidende Ungenauigkeit.

Zusammengefasst gleicht das SLAM Verfahren einer Schätzung über den Raum und die Position in diesem. Es gibt verschiedene SLAM Algorithmen um eine möglichst genaue Karte und die Position darin zu berechnen. Die Karten werden meist statisch hinterlegt und müssen zuvor angelernt werden (Offline SLAM). Bei Änderungen in der Umgebung sollte die Karte stets aktualisiert werden, um Fehler in der Navigation zu verringern (siehe *Mögliche Gefährdungen*). Beim Online SLAM hingegen wird die Karte und die Position permanent bei der Fahrt aktualisiert.

6.3.2.2 Mustererkennung & künstliche Intelligenz

Objekt-, Sprach- oder Gesichtserkennung sind technische Entwicklungen, die den Aufgabenbereich eines Assistenzroboter erweitern. Die Interaktion mit Systemen über Sprache lassen andere und intuitive Möglichkeiten zu. Diese Techniken können unter dem Begriff Künstliche Intelligenz (KI), oder Artifical Intelligence (AI), zusammengefasst werden. Dabei wird versucht, Maschinen, durch das Beobachten und Interpretieren der Umgebung, autonomes Handeln zu ermöglichen: Maschinen sollen intelligent auf neue, unbekannte Situationen reagieren, ohne dass eine situationsspezifische Programmierung vorgenommen werden muss.

Doch wann ist eine Maschine “intelligent”? Festzustellen ist, dass der Begriff der Intelligenz nicht genau definiert ist. Künstliche Intelligenzen sind daher als mathematische Modelle und Algorithmen zu verstehen, die statistische Zusammenhänge in Daten finden und vergleichen (Lämmel und Cleve, 2020, S. 9; vgl. Bünte, 2022, S. 85).

Je nach zu verarbeitenden Daten gibt es verschiedene mathematische Modelle, die angewendet werden können, um Zusammenhänge zu beschreiben. Klassische Beispiele dafür wären die sogenannten *Support Vector Maschines* (SVM), *k-nearest Neighbor Algorithmen* (kNN), *Decision Trees* oder *Markov-Modelle*. Je nach oben genannten Anwendungsfall ist das eine oder andere Modell besser geeignet. Vorher der Anwendungsphase muss jedoch jedes Model trainiert werden. Ein Modell stellt nur ein Grundgerüst dar, das mit Informationen und Daten gefüllt werden muss. Die Qualität der Daten, mit denen ein Modell trainiert wird, ist entscheidend für die Aussagegenauigkeit der KI. In der Informatik wird dieser Umstand als *Garbage In, Garbage Out* (GIGO) bezeichnet und beschreibt im übertragene Sinne: “Wenn (Daten-)Müll in ein Programm importiert wird, kann auch nur (Daten-)Müll herauskommen”. Das Problem hierbei ist, dass eine Maschine nicht entscheiden kann, was Müll ist und was nicht, was jedoch bei der Entwicklung berücksichtigt wird.

Wie ein Modell trainiert werden kann, lässt sich grob in drei Verfahren einteilen:

Supervised learning
Das überwachte Lernen beschreibt den Vorgang, bei dem einem Modell Daten eingespielt werden und zusätzlich mittgeteilt wird, um welche Daten es sich dabei handelt. Bspw. wird dem Modell ein Bild von einer Katze gezeigt und gleichzeitig mitgeteilt, das es sich um eine Katze handelt. Den Daten wird somit ein sog. Label mitgegeben.

Unsupervised learning
Bei dem Verfahren werden die Daten ohne Label verwendet. Das Modell muss selber Strukturen in den Daten erkennen (Clustering).

Reinforcement learning
Beim Reinforcement learning (RF-Learn) werden demgegenüber keine Daten für das Training benötigt. Hier lernt das Model durch "Ausprobieren", durch Erfolg und Misserfolg. Ähnlich wie ein Mensch das Fahrradfahren erlernt, so lernt das Modell beim RF-learning durch Interaktionen mit der Umgebung. Diese Lernmethode wird z. B. bei Videopielen angewendet, in denen gegen ein Computer gespielt werden kann. Beim Lernvorgang spielt der Computer das Spiel und probiert eine große Zahl an Möglichkeiten aus, um das Ziel zu erreichen. Dazu muss das Modell erst einmal herausfinden, wie z. B. die Spielfigur bewegt wird. Eine KI versucht Korrelationen zwischen den unbekannten Eingangsdaten und dem bereits antrainierten "Wissen" zu finden. Korrelation bedeutet allerdings nicht immer auch Kausalität. Ein einfaches Beispiel: Wird das Wort *Kuh* in einer Bildersuchmaschine eingegeben, werden viele Kühe entweder auf grünen Wiesen oder im Stall angezeigt. Würde man nun diese Bilder verwenden und eine KI damit trainieren, so würde bspw. eine Kuh am Strand nicht erkannt werden. Die Korrelation besteht darin, dass eine Kuh fast immer auf einer grünen Wiese steht. Für den Menschen ist klar, das die Kuh an sich nichts mit dem Hintergrund zu tun hat, dem KI-Modell hingegen nicht.
In dem Beispiel müssten die Trainingsdaten für eine erfolgreiche Erkennung stattdessen so gewählt werden, das möglichst viele unterschiedliche Kühe hinter verschiedenen Hintergründe stehen, um der Korrelation, dass Kühe immer auf grünen Wiesen stehen, zu entgehen. Auch Anweisungen über Sprache kann für eine KI zur Herausforderung werden. Es gibt sehr viele Varianten über Sprache, einen Sachverhalt darzustellen, wodurch eine große Anzahl an Variationen möglich ist. Zudem entscheiden Kontext, Mimik und Gestik sowie Lautstärke und Betonung, wie etwas Gesagtes gemeint ist.

Neuronale Netze

Mit den häufig eingesetzten Neuronalen Netzen (NN) als einem weiteren Mustererkenner wird versucht, ein generisches, also nicht auf ein spezifisches Problem zugeschnittenes Modell für auftretende Probleme zu erstellen. Im menschlichen Gehirn wird die Lösung der o.g. Problemstellung auf eines einzigen physiologischen Bausteins zurückgefürt — das Neuron. Über ein mathematisches Modell wird das Funktionsprinzip eines Neuron nachgebildet (Abbildung 8). Damit es für komplexe Aufgaben funktioniert, müssen, wie in einem Gehirn, eine sehr hohe Anzahl dieser Neuronen zusammengeschaltet werden, sodass die Lernphasen des NN eine sehr hohe Rechenleistung erfordern. Ebenfalls werden sehr viele Daten zum Trainieren des NN benötigt.

NN können in Schichten eingeteilt werden. Die erste Schicht, bei der die Daten eingehen, wird Input-Layer und die letzte Schicht als Output-Layer bezeichnet. Die Schichten zwischen Input- und Output-Layer werden Hidden-Layer genannt. Sobald ein Netzwerk mehr als einen Hidden-Layer hat, wird es als DeepNN bzw. tiefes neuronales Netzwerk bezeichnet.

In der Abbildung 8 ist ein einfaches mathematisches Modell eines künstlichen Neurons dargestellt. Unterschiedliche Parameter, wie Gewichtung, Schwellwert usw., werden über den Lernprozess mithilfe verschiedener Verfahren justiert. Der Ausgang des Neurons wird wieder mit einem Eingang eines oder mehrerer Neuronen verbunden, wie in Abbildung 9 zu sehen ist.

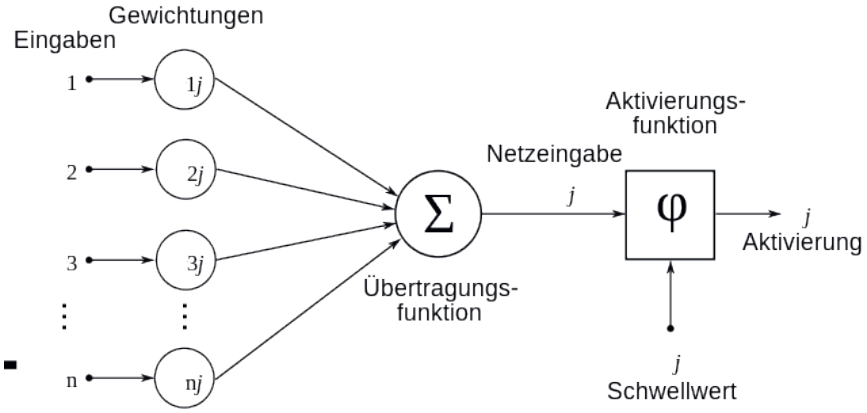


Abbildung 8: Darstellung eines Neuron-Modells[1]

1 Perhelion, Schematische Darstellung eines künstlichen Neurons mit dem Index j. 2010. Zugegriffen: 11. April 2023. [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/File:NeuronModel_deutsch.svg

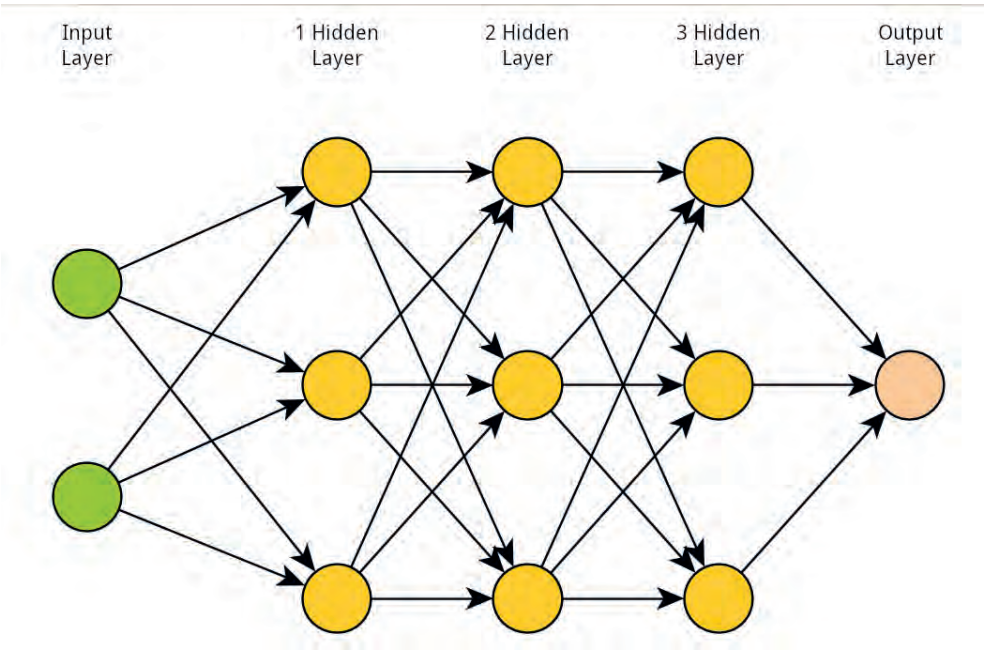


Abbildung 9: Darstellung eines NN [eigenes Werk]

6.3.3 Verschiedene Arten von Robotiksystemen

Viele Menschen denken bei dem Begriff Robotik oft zunächst an eine Maschine, die sich auf zwei Beinen bewegt, oder an einen Roboter-Arm, der etwas bearbeitet oder montiert. Dabei umfasst Robotik eine deutlich größere Spannweite an Systemen, wie beispielsweise Transportsysteme oder auch Exoskelette. Sie sollen mühselige oder sich oft wiederholende Aufgaben möglichst automatisch erledigen, um den Menschen ihre Arbeit oder ihren Alltag zu erleichtern. Systeme, die im Alltag unterstützende Aufgaben verrichten, werden als Assistenzsysteme bezeichnet, dazu zählen mobile Assistenzroboter, bewegungsunterstützende Roboter und Personenbeförderungsroboter für eine oder mehrere Personen.
Im Folgenden werden beispielhaft einige Projekte kurz vorgestellt, die sich mit Assistenzsystemen im Bereich der Pflege beschäftigen, um darzustellen, welche Formen und Funktionen Assistenzroboter haben können.

6.3.3.1 SeRoDi

Das Projekt SeRoDi ("Servicerobotik für personenbezogene Dienstleistungen") war ein Verbundprojekt, das in der Zeit von 2014 bis 2018 durchgeführt wurde. Das Ziel des Projekts war es herauszufinden, welche Auswirkung der Einsatz verschiedener Serviceroboter auf das Personal in stationären Einrichtungen hat.

Dabei entstand ein *intelligenter Pflegewagen*, der über ein Smartphone gerufen werden kann. Dem Pflegewagen wurde zuvor die Anzahl der Utensilien beim Befüllen mitgeteilt und alle entnommenen Utensilien werden gescannt. Der Wagen registriert daher, wann welche Materialien zu Neige gehen und fährt nach Freigabe des Pflegepersonals zur Auffüllstation. Weite Laufwege und das Führen einer Dokumentation über die verbrachten Materialien soll der Pflegewagen den Pfleger*innen abnehmen. Der Pflegewagen ist modular aufgebaut und kann für den Einsatz im Pflegeheim (Transport von Müll oder Wäsche) oder im Krankenhaus (Transport von Verbandsmaterialien etc.) angepasst werden (Christian Schiller u. a., 2019).

6.3.3.2 KUKA Robert

Der Roboter Robert der Firma KUKA (Abbildung 10) ist ein sehr gutes Beispiel für einen handgeführten Roboter (s. *Handführung*). Er wird für die Mobilisierung von Patientinnen und Patienten eingesetzt, bei denen Gelenke und Muskeln nach einer Operation bewegt werden müssen. Dafür wird eine Manschette um das zu bewegende Körperteil gewickelt und an einem Rotoberarm befestigt.

Die Pflegekraft kann nun über einen Griff am Roboterarm eine Bewegung durchführen und diese anschließend beliebig oft vom Roboter wiederholen lassen.
Der Roboter selbst ist mobil und kann von einer/einem Patienten*innen zum anderen geschoben werden. Ziel ist es, die Rehabilitationsmaßnahmen zu automatisieren und den Pflegekräften mehr Zeit für andere Aufgaben zu verschaffen (vgl. KUKA Group, ohne Datum).

6.3.3.2 RoboBed

Ein etwas anderes System ist das RotoBed. Gegenüber einem gewöhnlichen Pflegebett mit Motoren zum Aufrichten, kann dieses Bett sich über ein Kopfdruck zu einem Stuhl umformen. Dabei dreht sich das Bett um 90 Grad und unterstützt die Nutzer*innen beim Ein- und Aussteigen. Das Bett kann in den eigenen vier Wänden verwendet werden, um einen möglichst langen Verbleib der Pflege empfangenden Person im eigenen Haushalt zu gewährleisten und so die Lebensqualität und Autonomie zu erhöhen. Nutzer*innen sind so nicht mehr auf die Hilfe von Angehörigen oder Pflegekräften angewiesen (vgl. RotoBed, ohne Datum).

6.3.3.3 Pepper

Pepper wurde im Jahre 2014 von den Firmen Aldebaran und Softbanks entwickelt und ist ein ca. 1,20 m großer humanoider Roboter (Abbildung 11). Im Bereich der Pflege wird Pepper eingesetzt, um ältere Menschen oder Patient*innen zu unterhalten, ihnen Gesellschaft zu leisten und einfache Aufgaben, wie das Erinnern an die Medikamenteneinnahme, zu übernehmen. Über zahlreiche Sensoren ist es möglich, mit Pepper zu interagieren, sei es über Sprache, Gesten oder Gesichtsausdrücke. Ein Tablet an der Vorderseite erlaubt zudem weitere Interaktionsmöglichkeiten (vgl. „Pepper the Robot: All the Figures | Aldebaran“, 2015; „Pepper - ROBOTS: Your Guide to the World of Robotics“, ohne Datum).

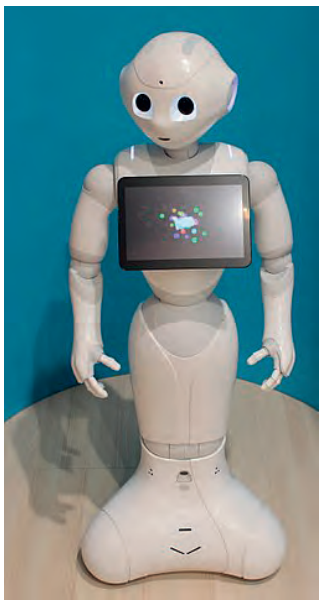


Abbildung 11: Der humanoider Roboter Pepper[1]

1 Xavier Caré, Aldebarans Pepper-Roboter bei Innorobo in Lyon im Jahre 2015. 2015. Zugegriffen: 20. April 2023. [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/File:Innorobo_2015_-Aldebaran-_Pepper.JPG

6.4 Quellen

Afflerbach, T. (2021) *Serviceroboter: Digitalisierung von Dienstleistungen Aus Kunden-, Mitarbeiter-Und Managementperspektive.* Springer Fachmedien Wiesbaden GmbH.

Behrens, R. und Pliske, G. (2019) *Human-Robot Collaboration: Partial Supplementary Examination [of Pain Thresholds] for Their Suitability for Inclusion in Publications of the DGUV and Standardization.* FF-FP 0430. Magdeburg: Fraunhofer IFF. Verfügbar unter: https://www.dguv.de/ifa/forschung/projektverzeichnis/ff-fp0430.jsp (Zugegriffen: 9 März 2023).

Bendel, O. und Daimler und Benz Stiftung (Hrsg.) (2018) Pflegeroboter. Wiesbaden: Springer Gabler (OPEN).

Bünte, C. (2022) „Künstliche Intelligenz Ein Überblick über die aktuelle und zukünftige Bedeutung von KI in der Wirtschaft und im Gesundheitswesen in Europa", in M.A. Pfannstiel (Hrsg.) *Künstliche Intelligenz im Gesundheitswesen.* Wiesbaden: Springer Fachmedien Wiesbaden, S. 81–100. Verfügbar unter: https://doi.org/10.1007/978-3-658-33597-7_3.

Christian Schiller u. a. (2019) „Servicerobotik bei personenbezogenen Dienstleistungen". Verfügbar unter: https://www.ipa.fraunhofer.de/content/dam/ipa/de/documents/Projekte/Forschungsprojekte/Abschlussbrosch%C3%BCre%20Servicerobotik%20in%20der%20Pflege.pdf (Zugegriffen: 28 April 2023).

DIN e.V. (Hrsg.) (2012) „DIN EN ISO 10218-1". Beuth-Verlag Berlin.

DIN e.V. (Hrsg.) (2014) „DIN EN ISO 13482". Beuth-Verlag Berlin.

DIN e.V. (Hrsg.) (2016a) „DIN EN ISO 13849-1". Beuth-Verlag Berlin.

DIN e.V. (Hrsg.) (2016b) „DIN EN ISO 13850". Beuth-Verlag Berlin.

DIN e.V. (Hrsg.) (2017) „DIN ISO/TS 15066". Beuth-Verlag Berlin.

Dütthorn, N., Hülsken-Giesler, M. und Pechuel, R. (2018) „Game Based Learning in NursingDidaktische Und Technische Perspektiven Zum Lernen in Authentischen, Digitalen Fallsimulationen", *Digitale Transformation von Dienstleistungen im Gesundheitswesen IV: Impulse für die Pflegeorganisation*, S. 83–101.
KUKA Group (ohne Datum) „ROBERT Rehab Robot Life Science Robotics Patient Nurse 2". Verfügbar unter: https://assets.kuka.com/share/3E22A888-791C-4527-9AF103634EA222D2/?mediaId=2679DC01-5E01-4F85-ADBB51449291845B (Zugegriffen: 13 April 2023).

Lämmel, U. und Cleve, J. (2020) *Künstliche Intelligenz: Wissensverarbeitung - neuronale Netze.* 5., überarbeitete Auflage. München: Hanser.

Maier, H. (2016) *Grundlagen der Robotik*. Berlin Offenbach: VDE Verlag GmbH (Lehrbuch Studium).

Mainzer, K. und Mainzer (2016) *Künstliche Intelligenz-Wann Übernehmen Die Maschinen?* Springer.

„Pepper - ROBOTS: Your Guide to the World of Robotics" (ohne Datum). Verfügbar unter: https://robots.ieee.org/robots/pepper/ (Zugegriffen: 3 Mai 2023).

„Pepper the Robot: All the Figures | Aldebaran" (2015). Verfügbar unter: https://web.archive.org/web/20150910140352/https://www.aldebaran.com/en/a-robots/pepper/more-about-pepper (Zugegriffen: 3 Mai 2023).

RotoBed (ohne Datum) „RotoBed Home Brochure". Verfügbar unter: https://rotobed.com/en/wp-content/uploads/2020/08/RotoBed%C2%AEHome-US-Brochure.pdf (Zugegriffen: 26 April 2023).

Steil, J.J. (2019) „Roboterlernen Ohne Grenzen? Lernende Roboter Und Ethische Fragen", *Roboter in der Gesellschaft: Technische Möglichkeiten und menschliche Verantwortung*, S. 15–33.

Wahl, H.-W., Mombaur, K. und Schubert, A. (2021) „Robotik Und Altenpflege: Freund Oder Feind?“, *Pflegezeitschrift*, 74(11), S. 62–66.

Wallhoff, F., Vox, J.P. und Theuerkauff, T. (2019) „Assistenz- und Servicerobotik die Gestaltung der Mensch-Maschine-Schnittstelle als Grundlage des Anwendungserfolgs“, in R. Haring (Hrsg.) Gesundheit digital. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 99–122. Verfügbar unter: https://doi.org/10.1007/978-3-662-57611-3_7.

Wippich, A. und Klein, M. (2022)„„Robotik Und KI in Der Pflege “als Lerneinheit in Der Generalistischen Pflegeausbildung, Bedarf Und Pflegerische Wirklichkeit“, in Mario A. Pfannstiel (Hrsg.) *Künstliche Intelligenz Im Gesundheitswesen: Entwicklungen, Beispiele Und Perspektiven.* Wiesbaden: Springer, S. 821–833. Verfügbar unter: https://doi.org/10.1007/978-3-658-33597-7.

Zech, H. (2020) „Risiken Digitaler Systeme: Robotik, Lernfähigkeit Und Vernetzung Als Aktuelle Herausforderungen Für Das Recht“, *Weizenbaum Series* [Preprint]. Verfügbar unter: https://doi.org/10.34669/WI.WS/2.

Notizen