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.

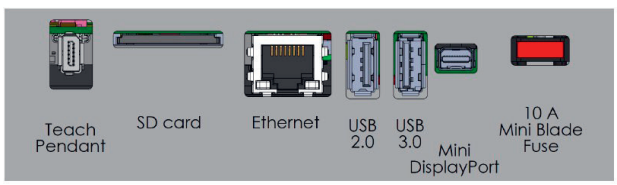
Nr.	Element
1	Startknopf
2	Not-Aus-Schalter
3	USB-Anschluss
4	3PE Knöpfe oder Freedrive Knopf

4.1.1.3 Controller

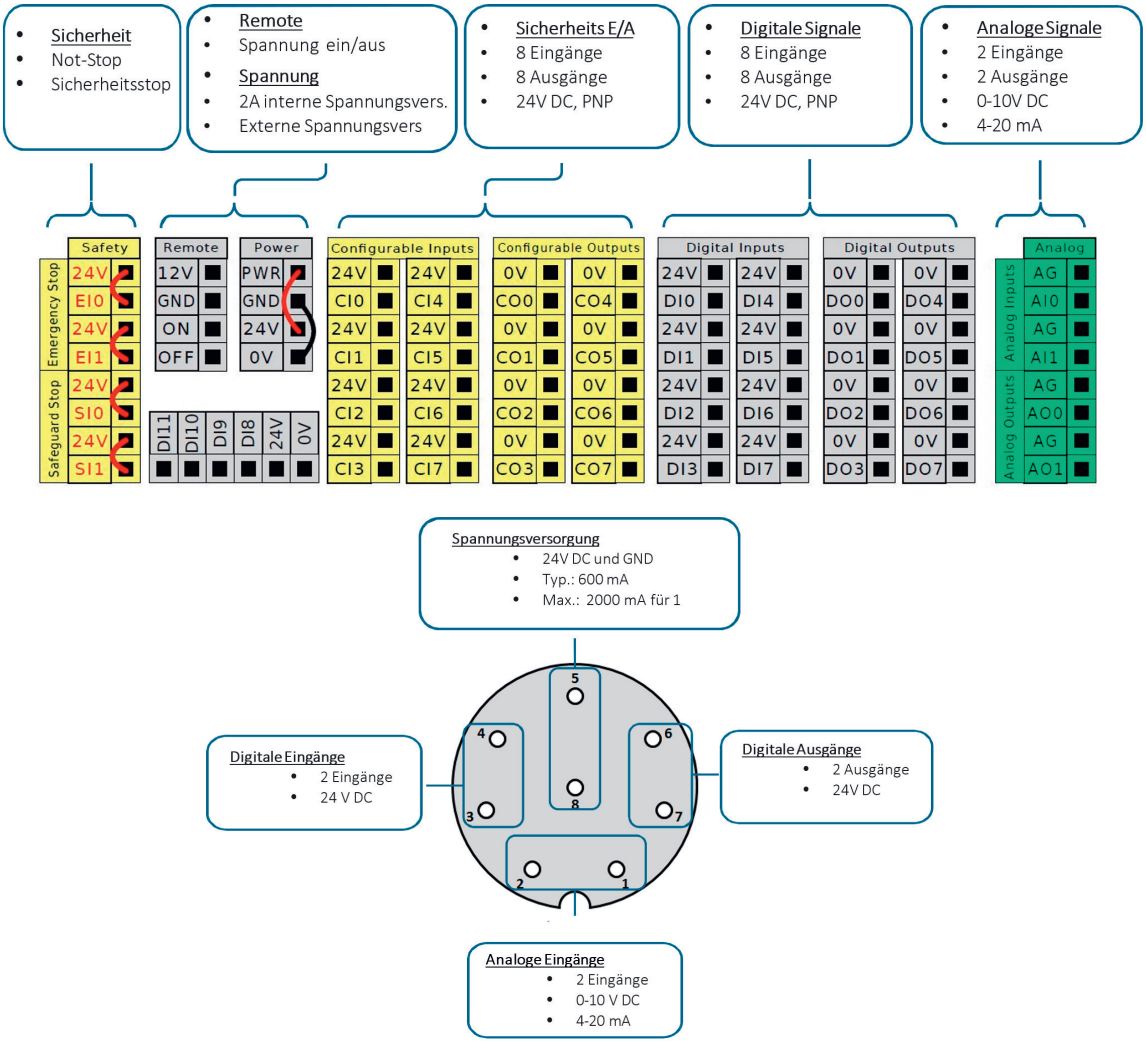
Der Controller des UR ist das zentrale Steuerungs- und Rechenelement des Robotersystems. Er koordiniert und leitet alle relevanten Informationen, wie Sensordaten und Bewegungsanweisungen, um die gewünschten Roboteraktionen auszuführen.
Die Anschlüsse eines UR-Controllers variieren je nach Modell, aber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

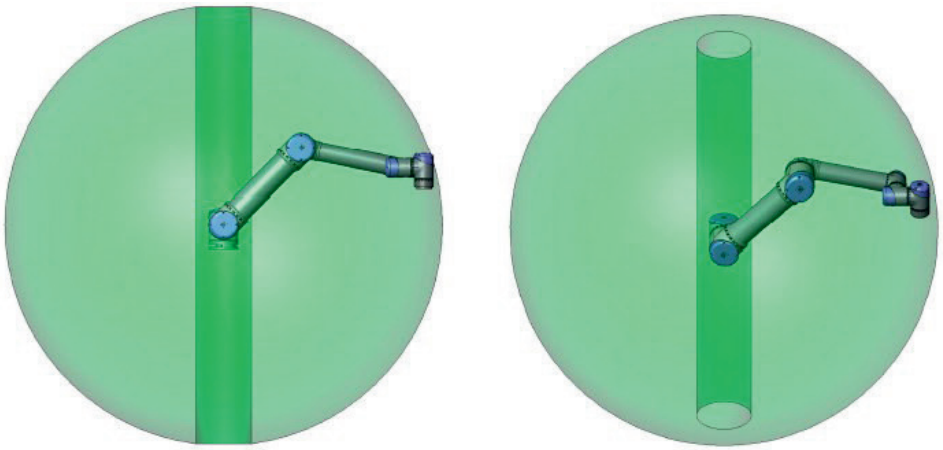


In der folgenden Abbildung sind sowohl Controller-E/A als auch der Werkzeuganschluss dargestellt.



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.

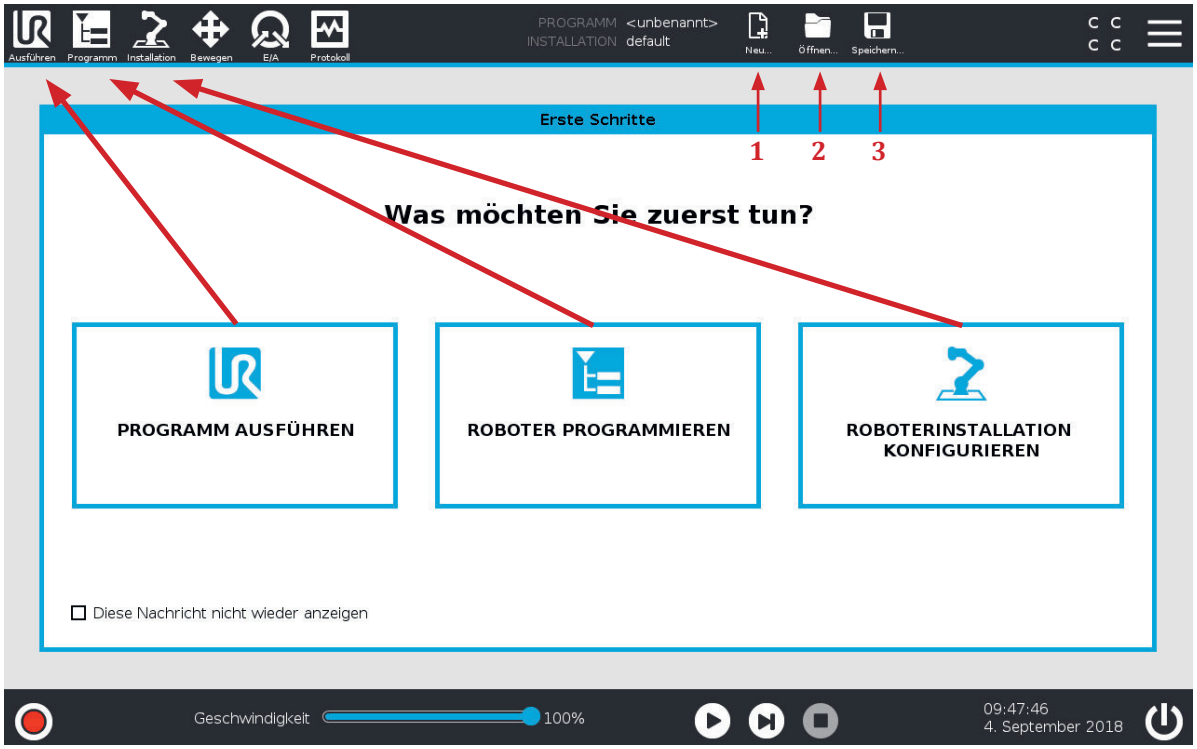


Front Geneigt

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.



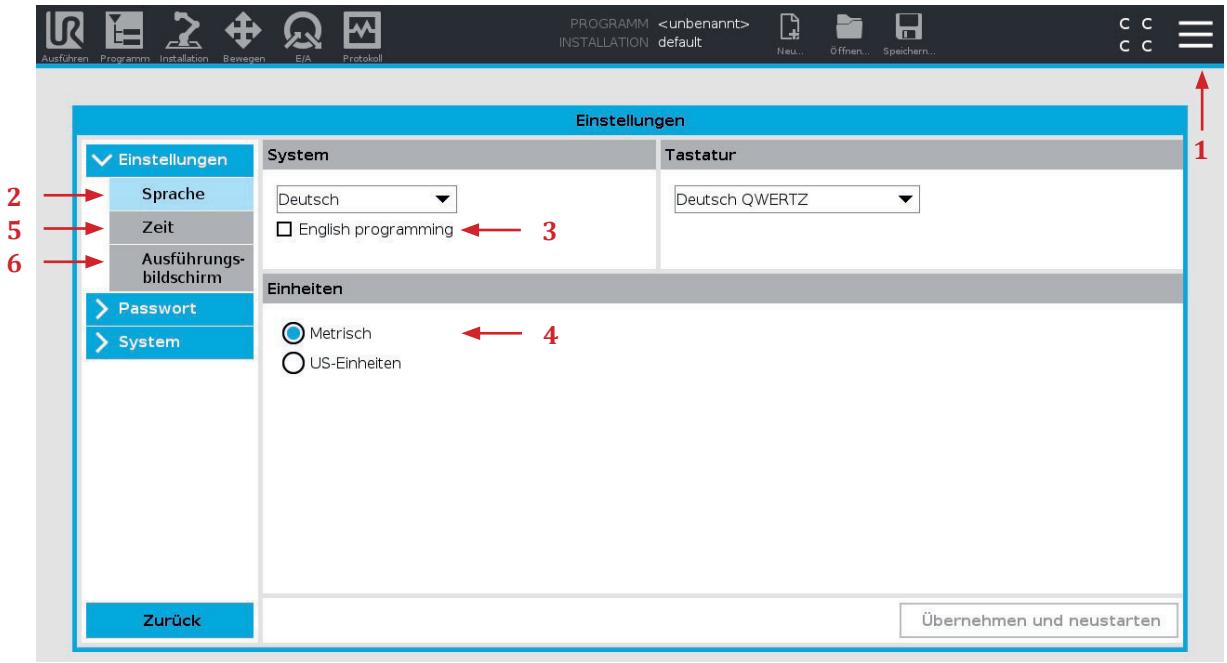
Nach dem Starten des Roboters wird der Startbildschirm angezeigt. Der Bildschirm bietet folgende Auswahloptionen an, welche sich auch in der dauerhaft eingeblendeten Statusleiste wiederfinden.

- **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

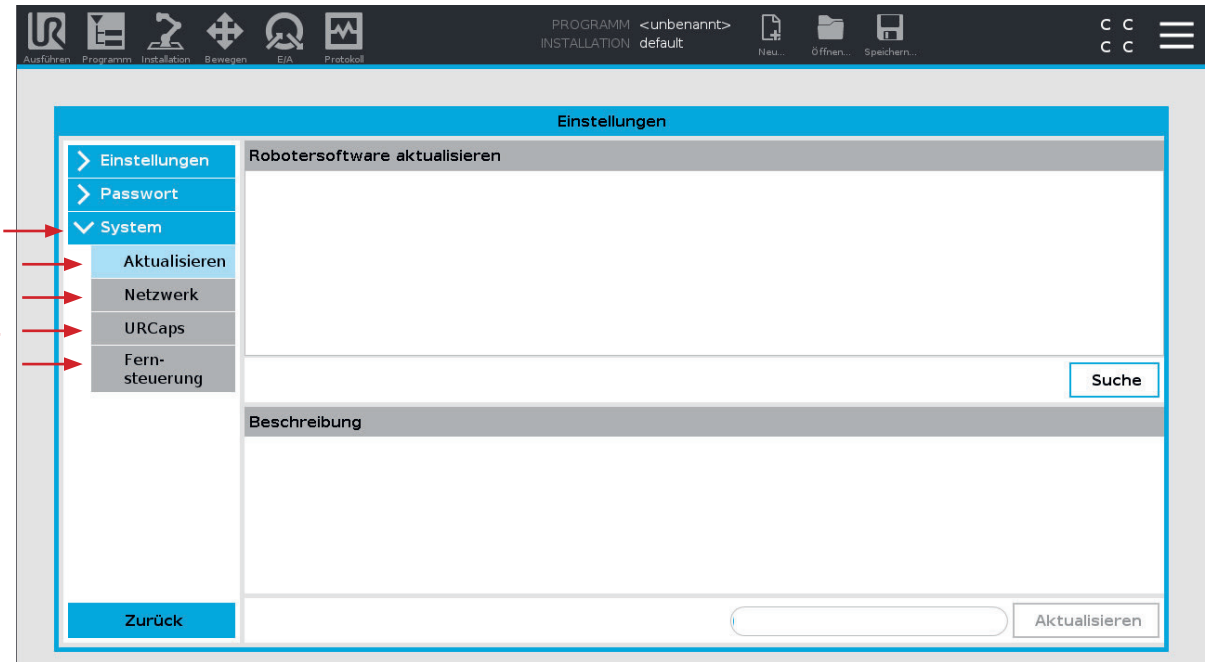


Nr.	Modul	Beschreibung
1	**Hamburger Menü**	Dient zum Aufrufen der PolyScope Hilfe, Infos und Einstellungen
2	**Sprachen**	Auswahl der Sprache (23 Sprachen)
3	**English programming**	Programmier-Befehle in englischer Sprache
4	**Einheit**	Auswahl der Einheit (Metrisch oder U.S.)
5	**Zeit**	Auswahl des 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

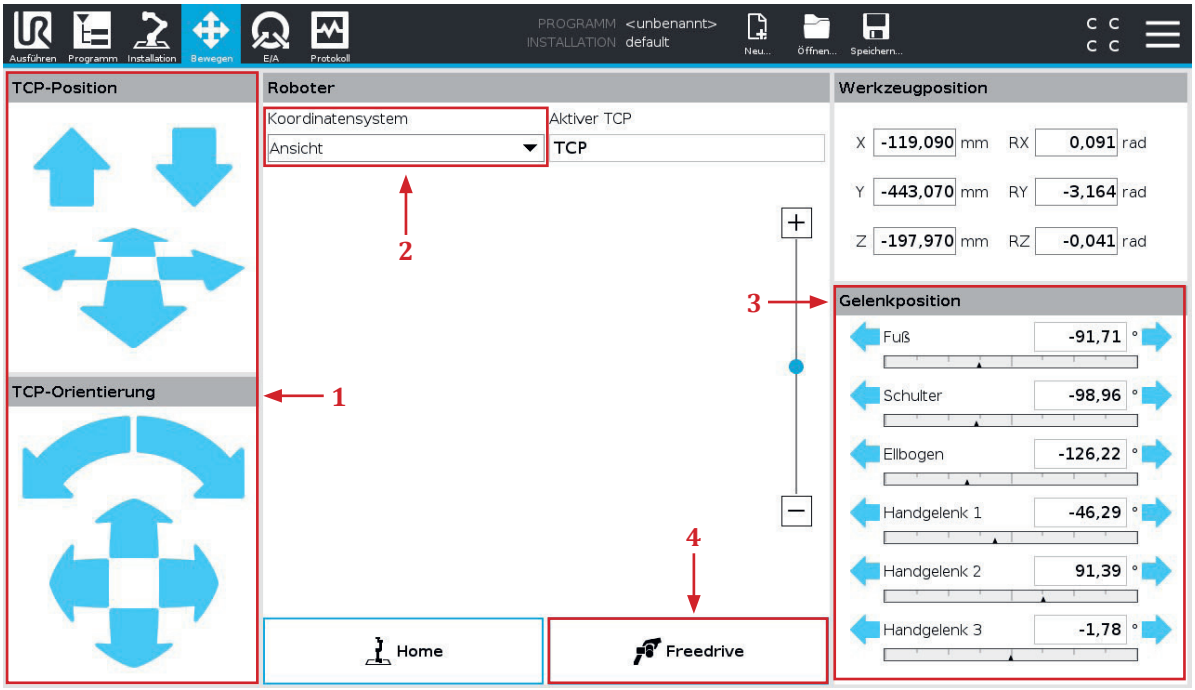
4.1.2.3 Tab System



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

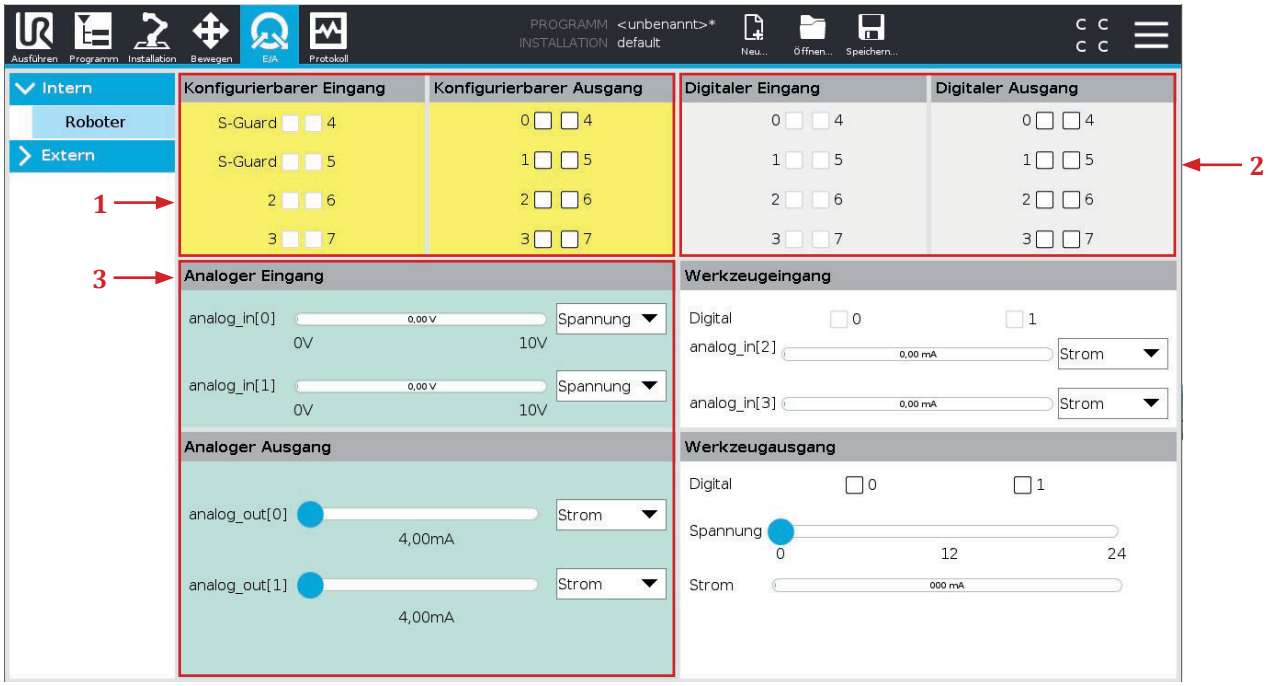
4.1.3 Register „Bewegen“

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“

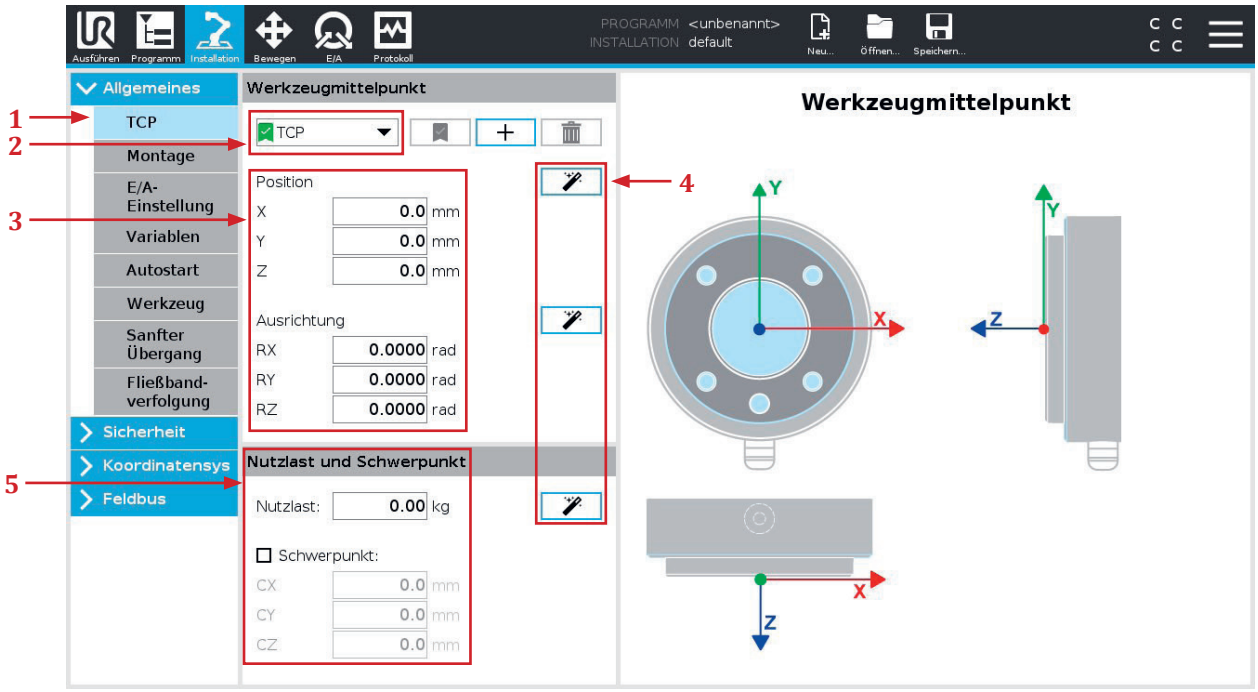


In diesem Bildschirm können die E/A-Signale überwacht und eingestellt werden. Der Bildschirm zeigt den aktuellen Zustand der Ein- und Ausgänge an, auch während der Programmausführung. Werden während der Ausführung des Programms Änderungen vorgenommen, so stoppt das Programm. Wenn ein Programm stoppt, 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.

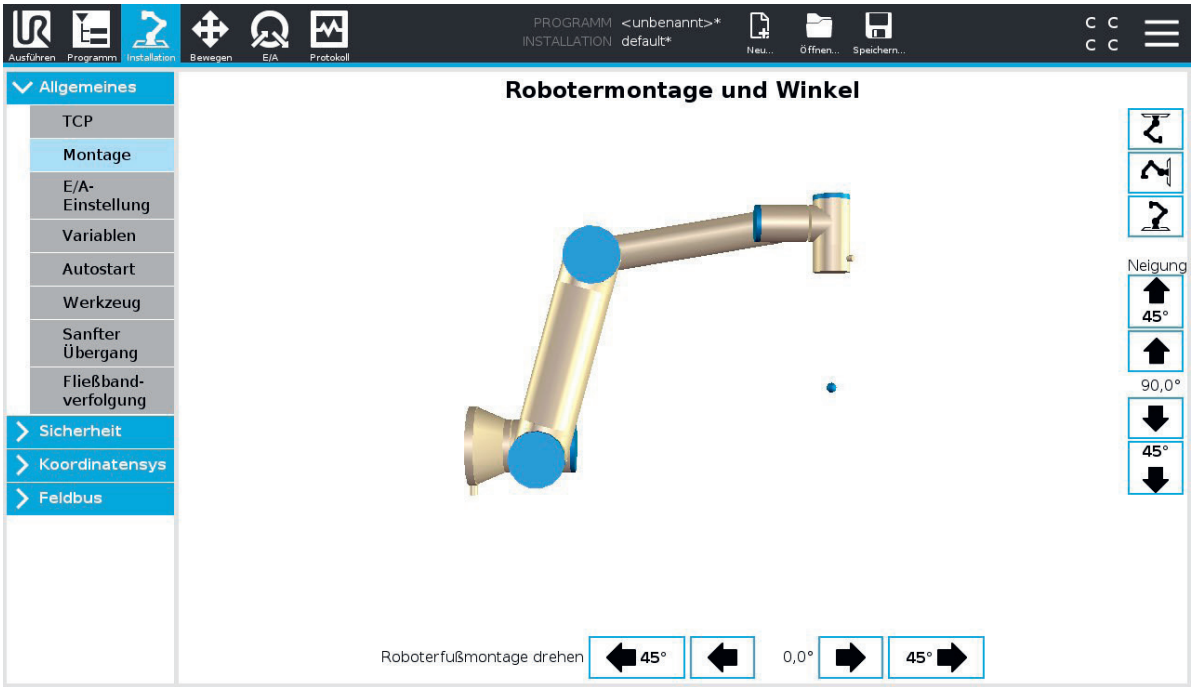


Nr.	Modul	Beschreibung
1	**TCP**	Der Tool-Center-Point (TCP) ist der Punkt am Ende des Roboterarms, an dem sich das Werkzeug oder die Greifvorrichtung befindet.
2	**Dropdown-Menü-TCP**	Es kann mehr als ein TCP eingestellt werden
3	**Position/Ausrichtung**	Einstellung der XYZ- bzw. RXRYRZ-Werte gem. Darstellung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

4.1.5.1 TCP-Konfigurationsassistent

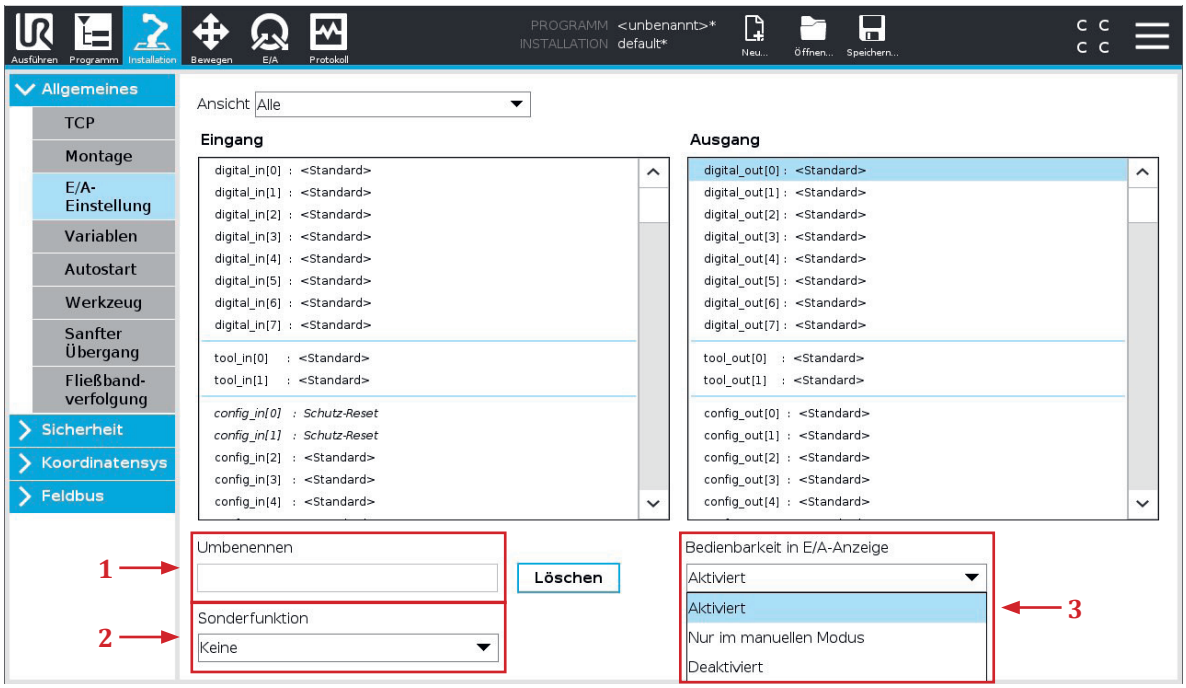
Der Universal Robot Assistent zur Einstellung der Nutzlast und des Schwerpunkts ist hilfreich, da er den Einrichtungsprozess für den Roboter vereinfacht und automatisiert.
Durch die Verwendung des Universal Robot Assistenten zur Einstellung der Nutzlast und des Schwerpunkts kann der Einrichtungsprozess 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.

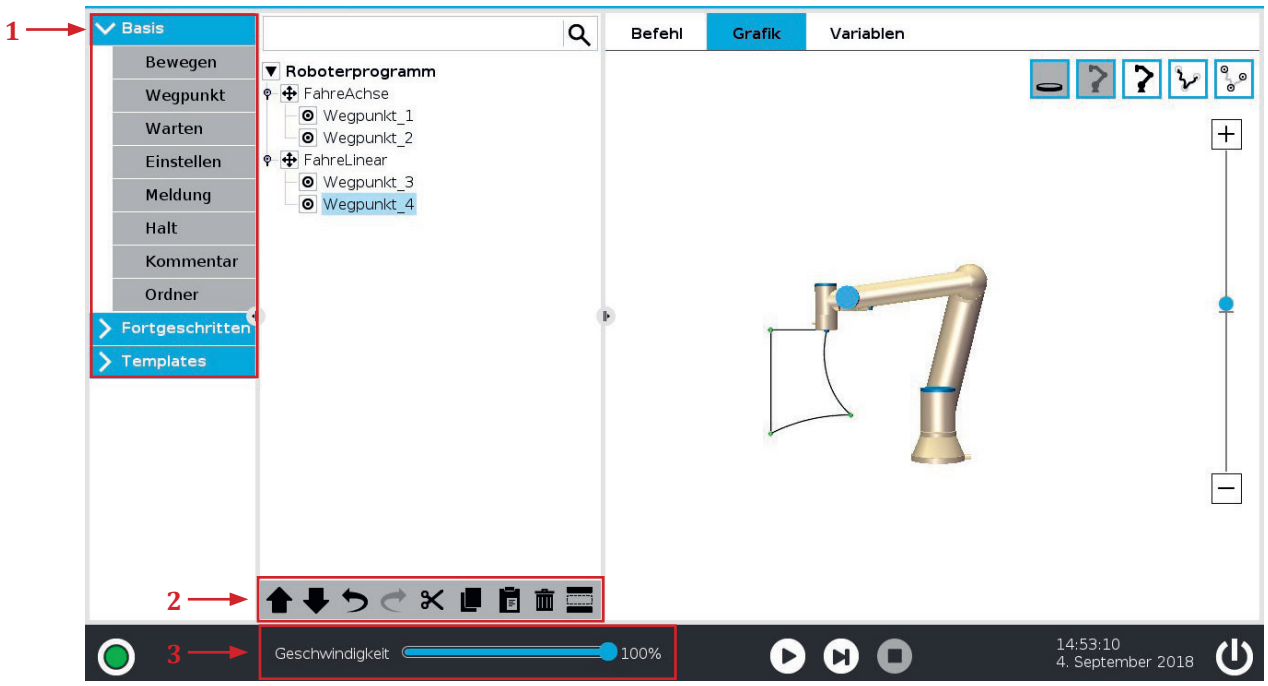
4.1.5.3 E/A-Einstellungen



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.

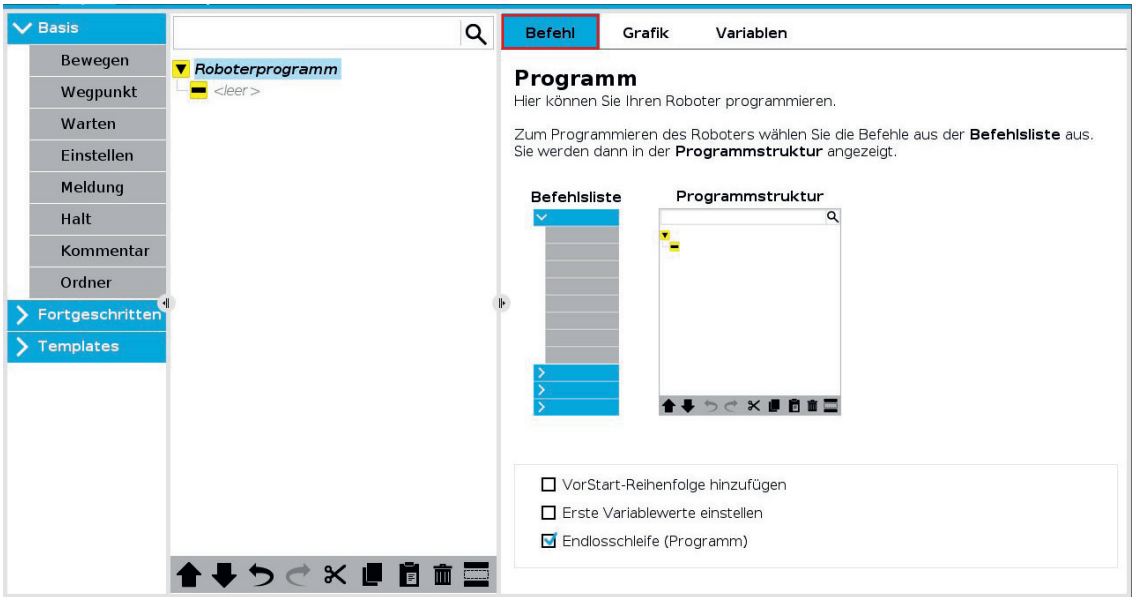
4.1.6 Register „Programm“

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.

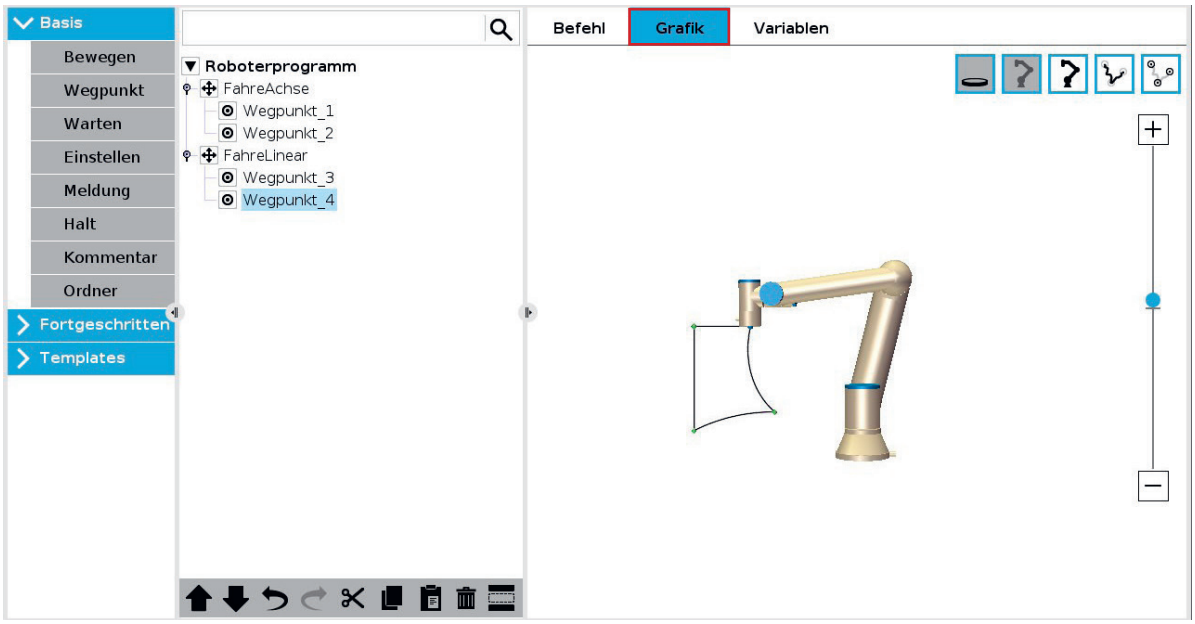
4.1.6.1 Tab Befehl



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.

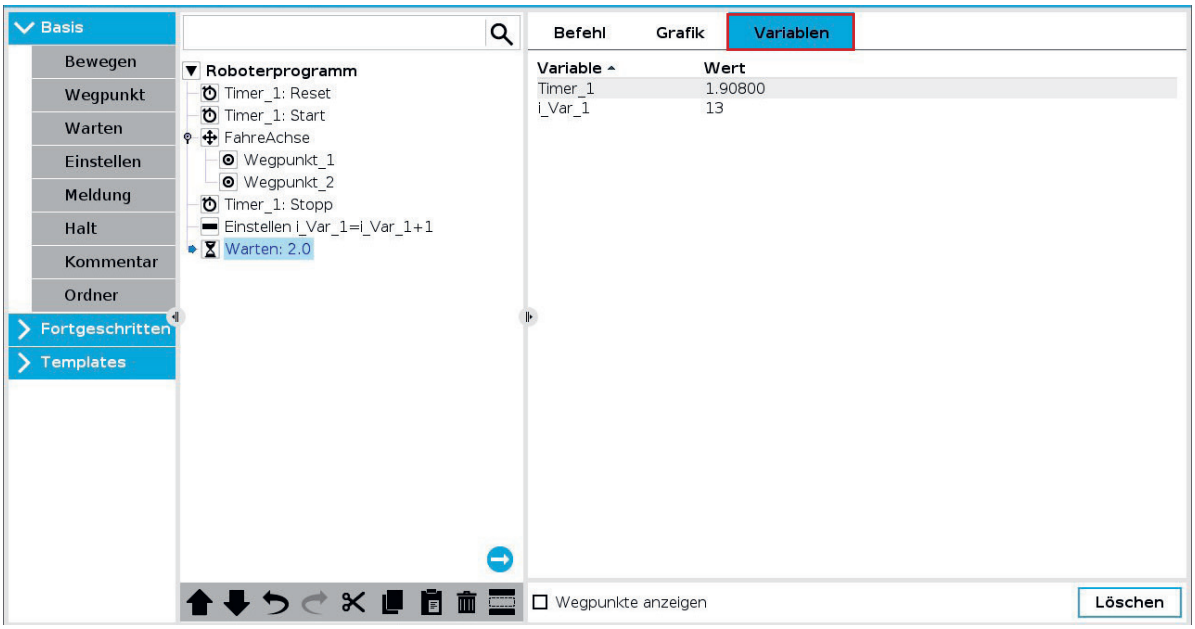
4.1.6.2 Tab Grafik

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.



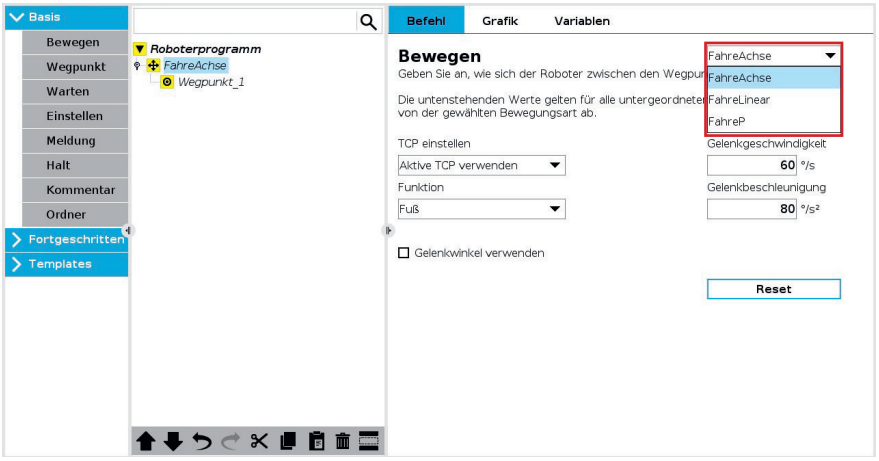
4.1.6.3 Tab Variablen

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

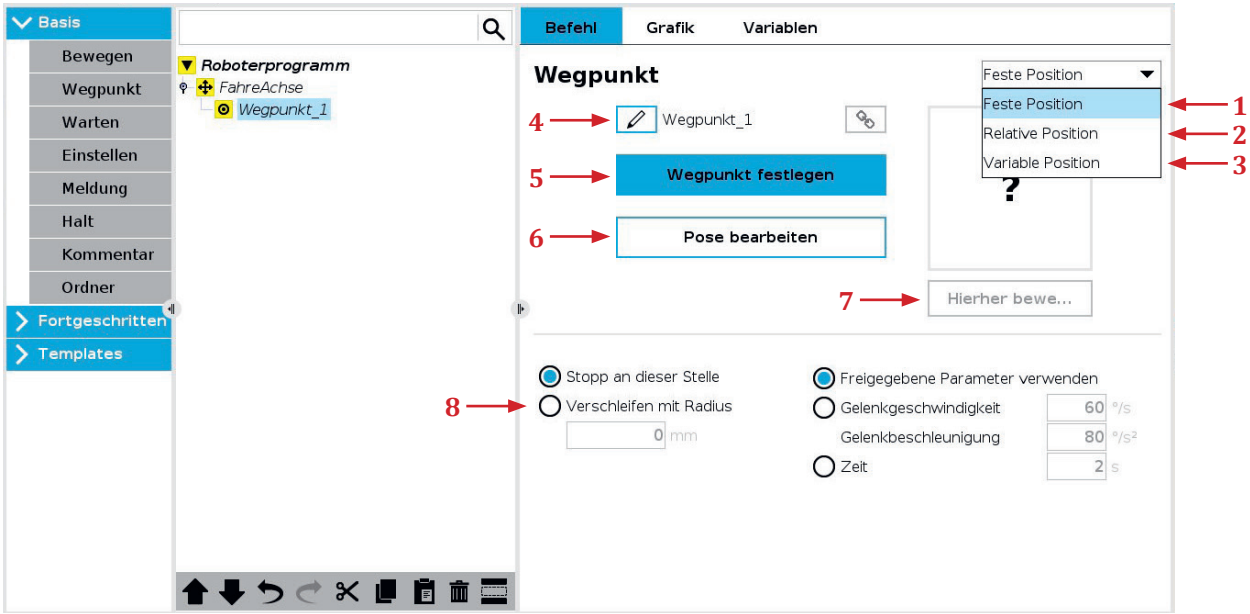
4.1.7.1 Bewegen



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.

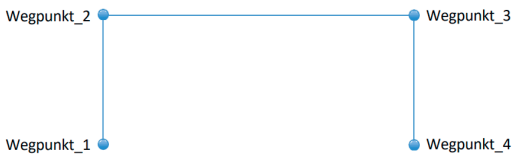
4.1.7.2 Wegpunkt



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.

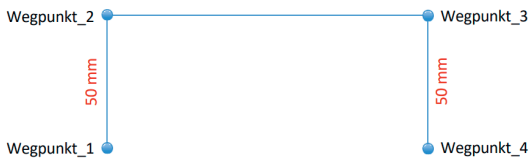
4.1.7.3 Ausprobieren: Erste Wegpunkte

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.



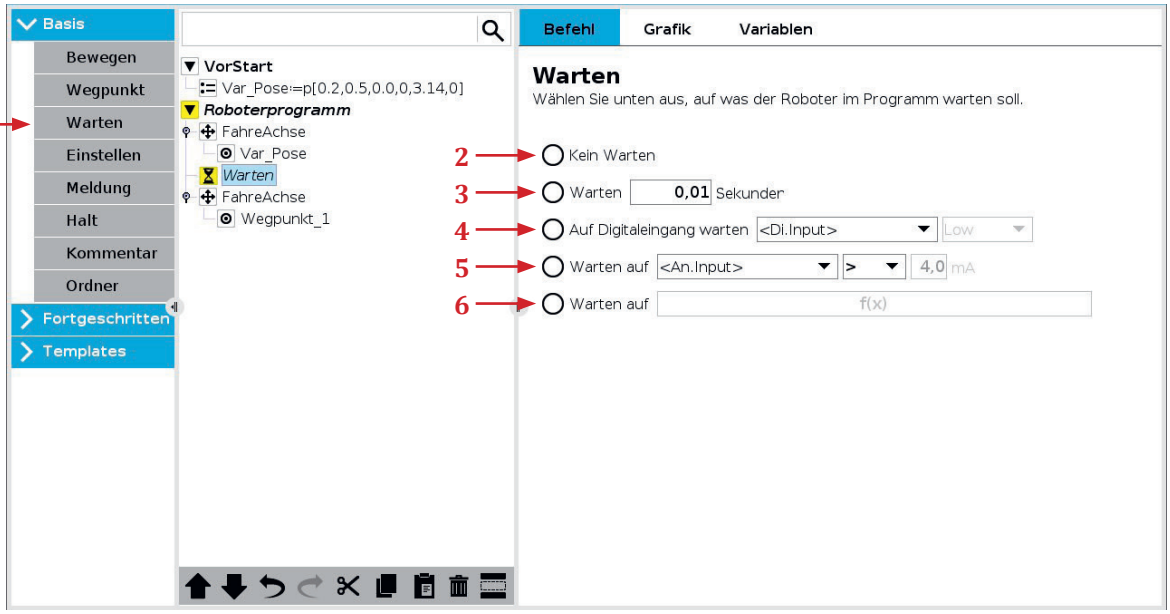
4.1.7.4 Ausprobieren: Definierte Wegstrecken

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.

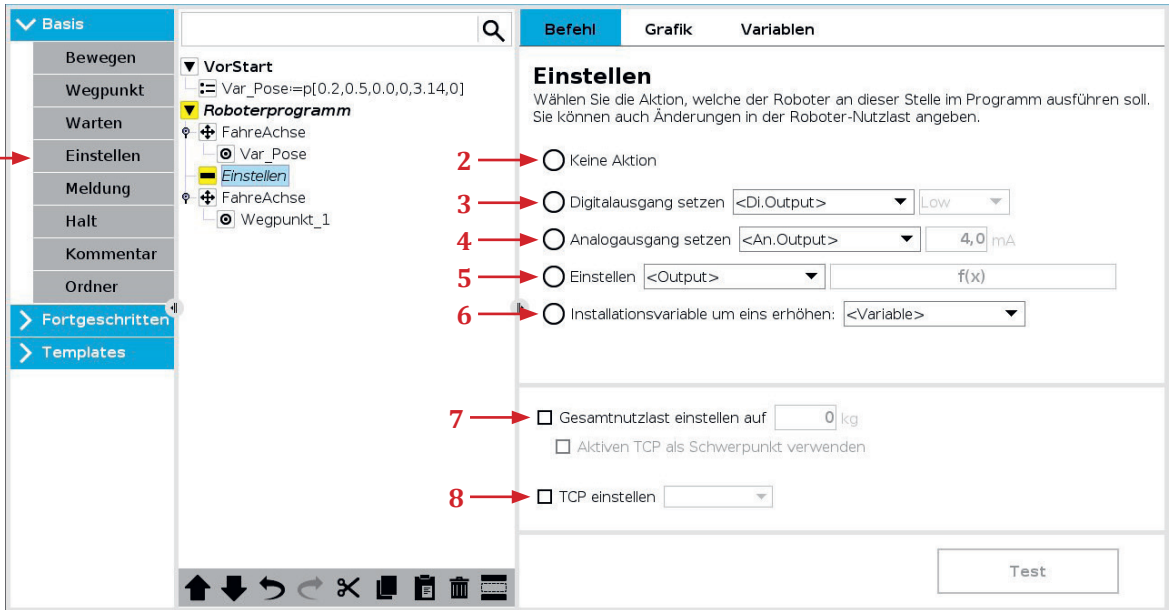
4.1.7.5 Befehl: Warten



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

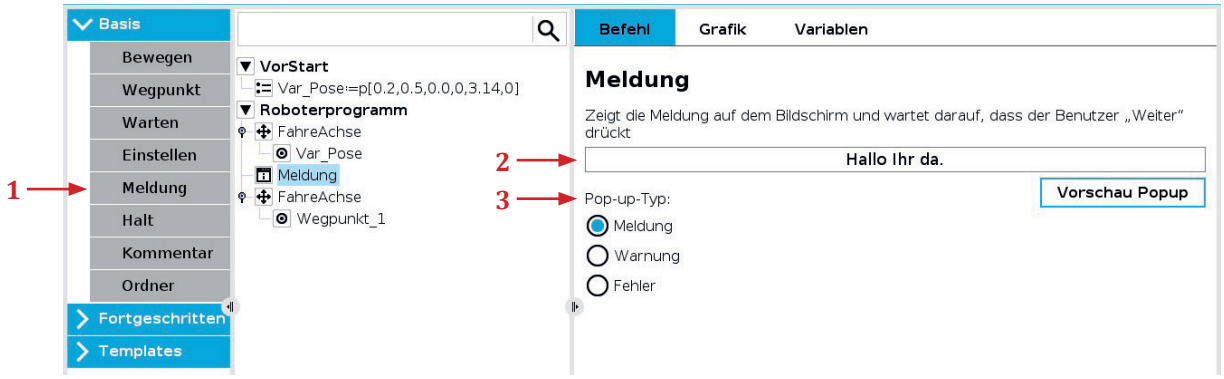
4.1.7.6 Befehl: Einstellen



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



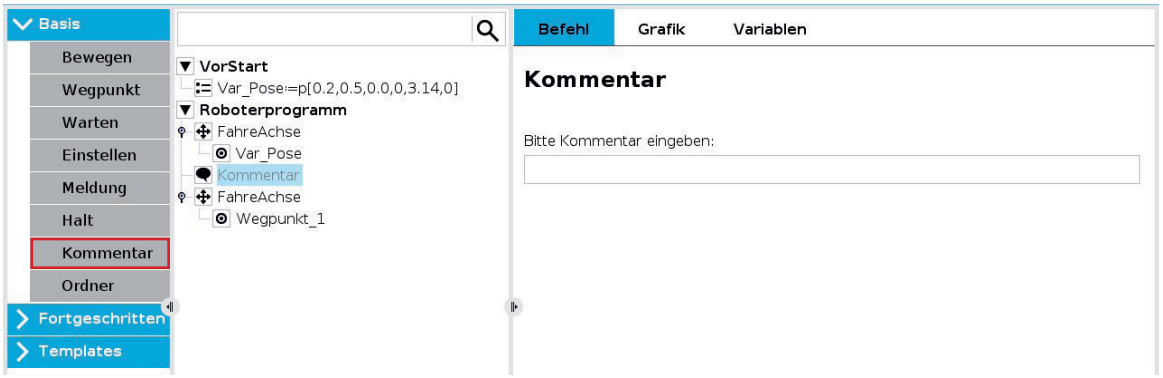
Meldung wir in Form eines Popups angezeigt. Programmablauf wird pausiert, bis Nutzerinteraktion erfolgt hat.

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

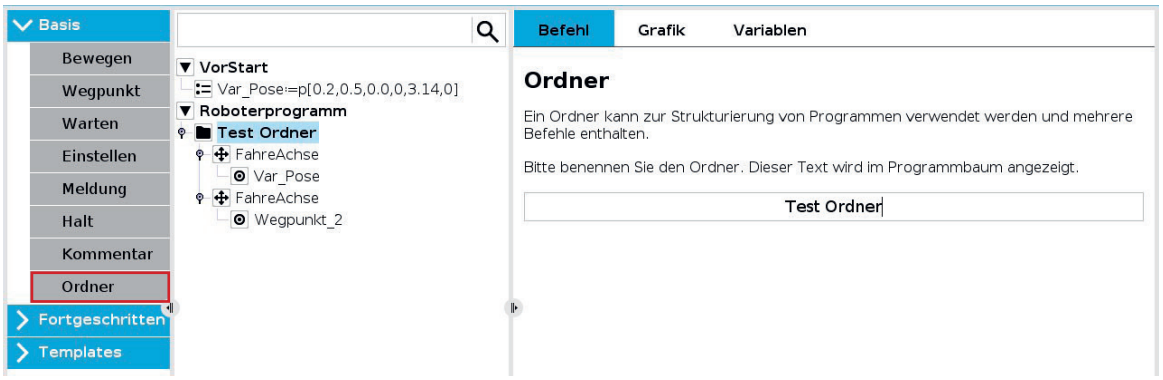
Die Ausführung des Programms wird an dieser Stelle beendet.

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

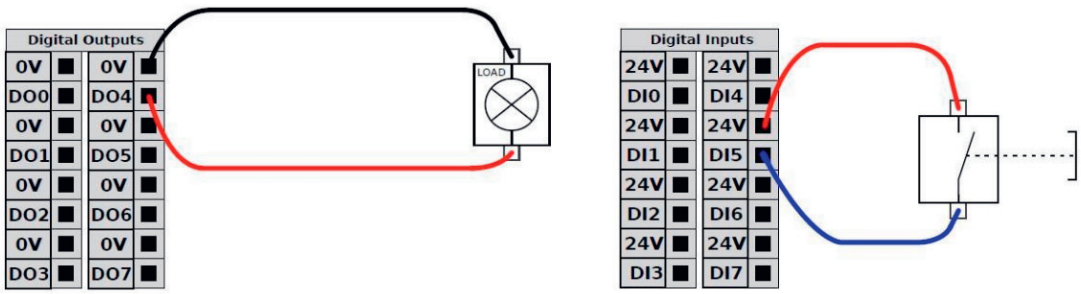


Ein Ordner kann zur Organisation und zur besseren Lesbarkeit des Programms eingesetzt werden. Ordner sind rein visuell und haben keine Auswirkungen auf das Programm und seine Ausführung.

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

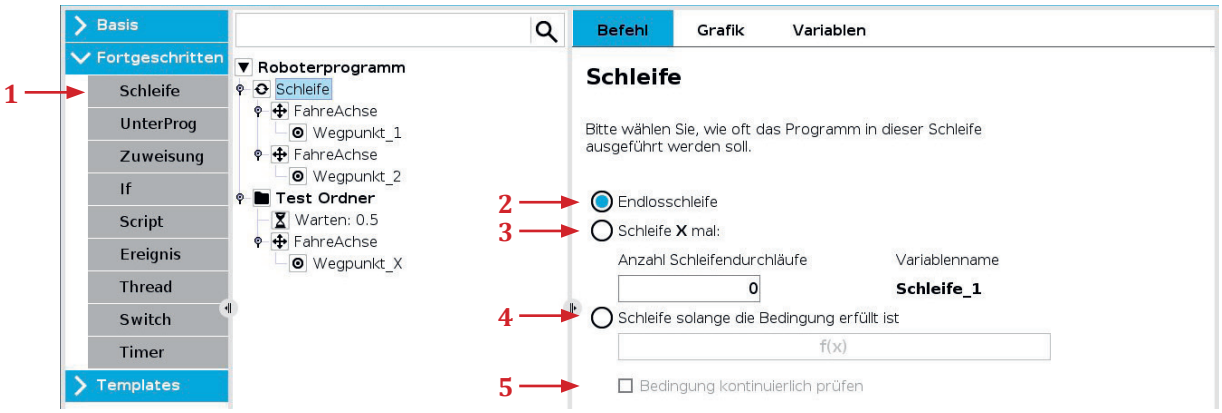
Ordner - PickUp
- Erstelle einen Ordner mit der Bezeichnung ”PickUp”
- Fahre in diesem Ordner auf WP1 und dann linear nach unten auf WP2
- An WP2 Greifer schließen und Teil aufnehmen dann wieder auf WP1

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

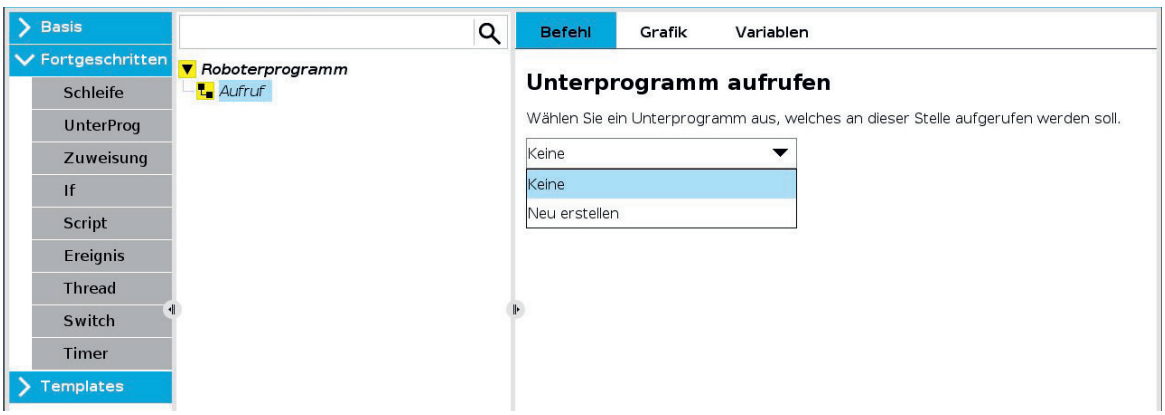
4.1.8.1 Befehl: Schleife



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.

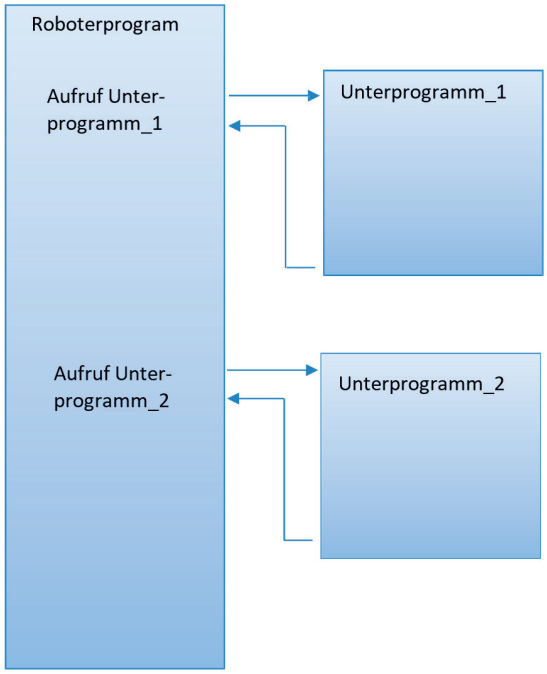
Nr.	Modul	Beschreibung
1	**Schleifen**	Befehl hinzufügen
2	**Endlosschleifen**	Die Schleife wir endlos ausgeführt. Hinweis: Manuelle Abbruchbedingung nötig
3	**Schleife X mal**	Schleife wird um die angegebene Anzahl wiederholt
4	**Schleife, solange die Bedingung erfüllt ist**	Schleife wird wiederholt, 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.

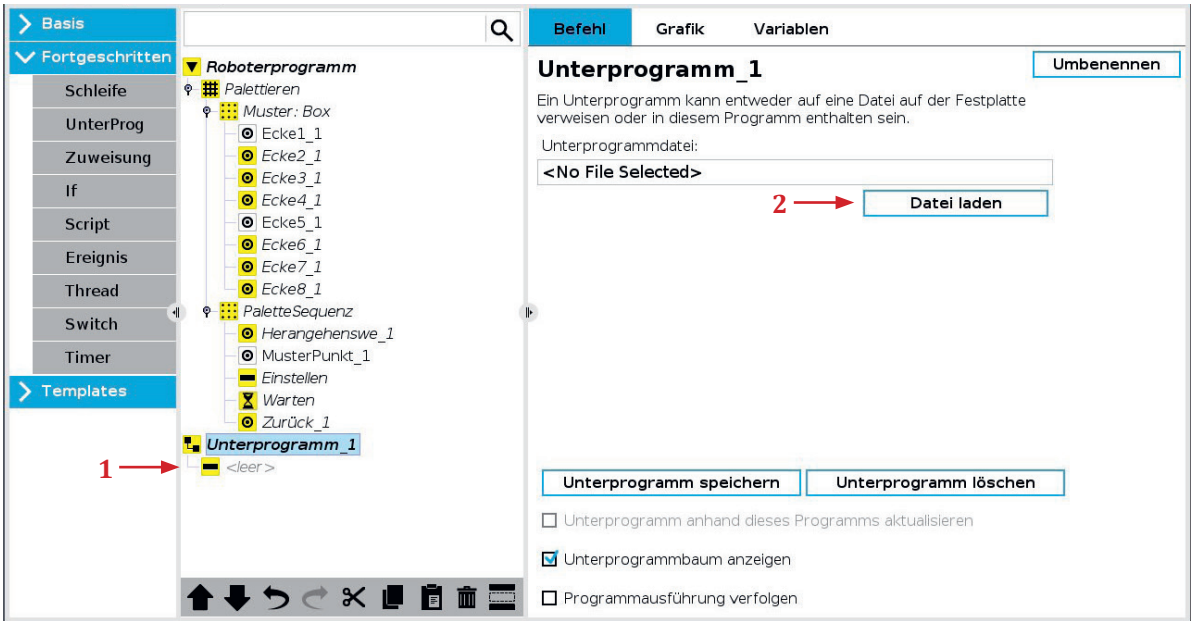
4.1.8.2 Befehl: Unterprogramme



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.

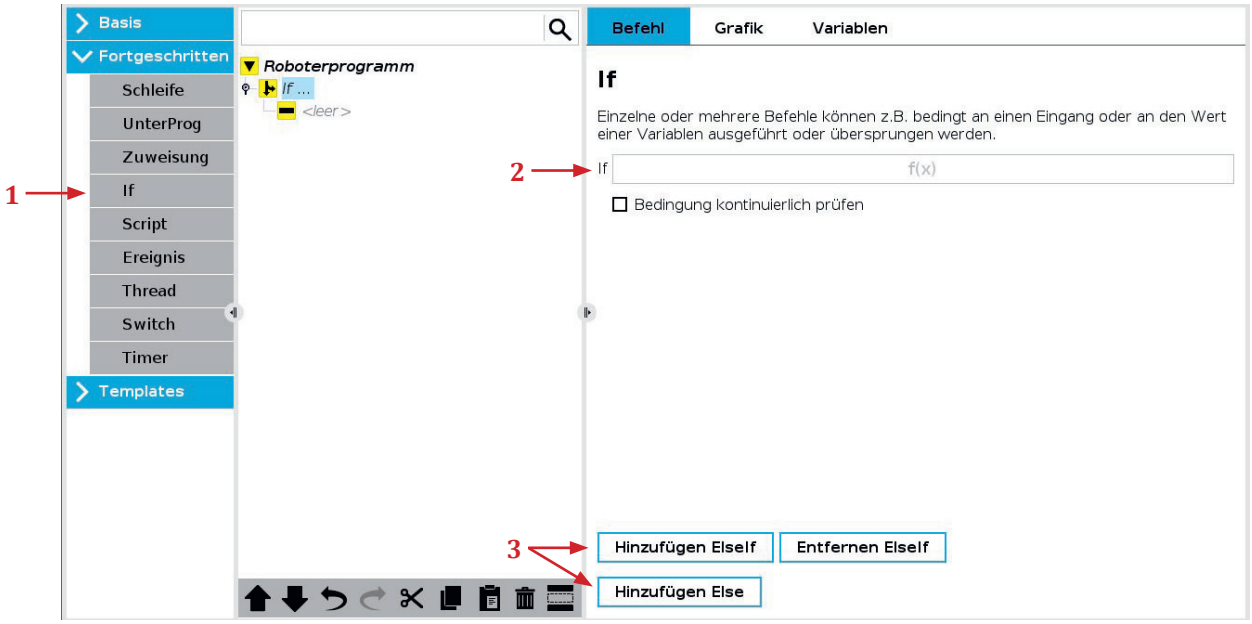
So wird es möglich Programmteile in unterschiedlichen anderen Programmen erneut zu verwenden. Der Aufruf eines Unterprogramms innerhalb eines Unterprogramms ist nicht möglich. Die Ausführung des Unterprogramms erfolgt sequenziell, das heißt das Hauptprogramm wird für das Unterprogramm verlassen und danach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

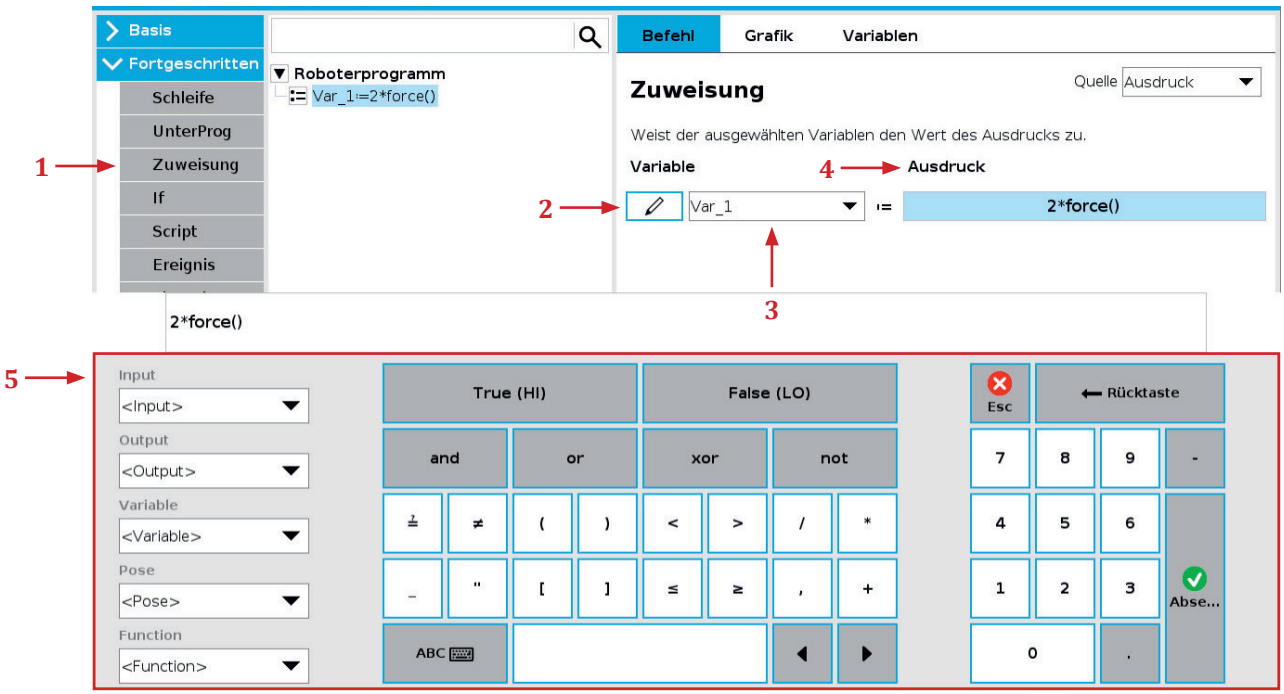
Arten von Variablen:

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

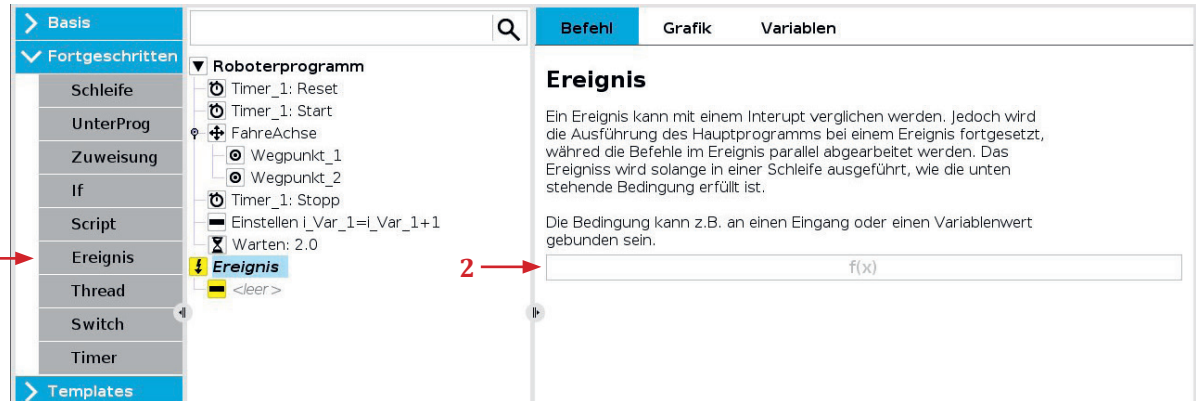
Benötigtes Material: 1x Schalter/Taster
Aufgabe: Erstelle ein Programm welches Installationsvariablen verwendet um Programmdurchläufe zu zählen

- 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

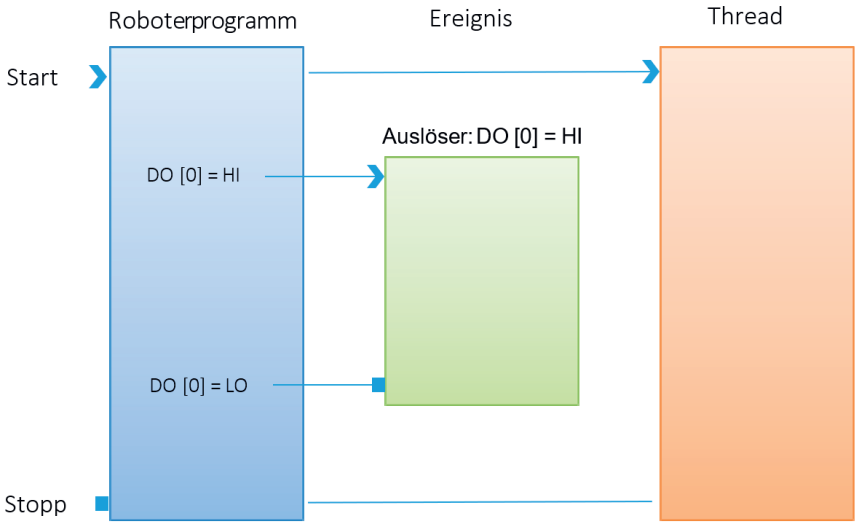
Ein Ereignis definiert einen Programmteil, der nur ausgeführt wird, wenn die im Ereignis definierte Bedingung erfüllt wird. Die Besonderheit eines Ereignisses ist, dass es zeitgleich also parallel zum Hauptprogramm in einer Endlosschleife ausgeführt wird, solange die Bedingung erfüllt ist.



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

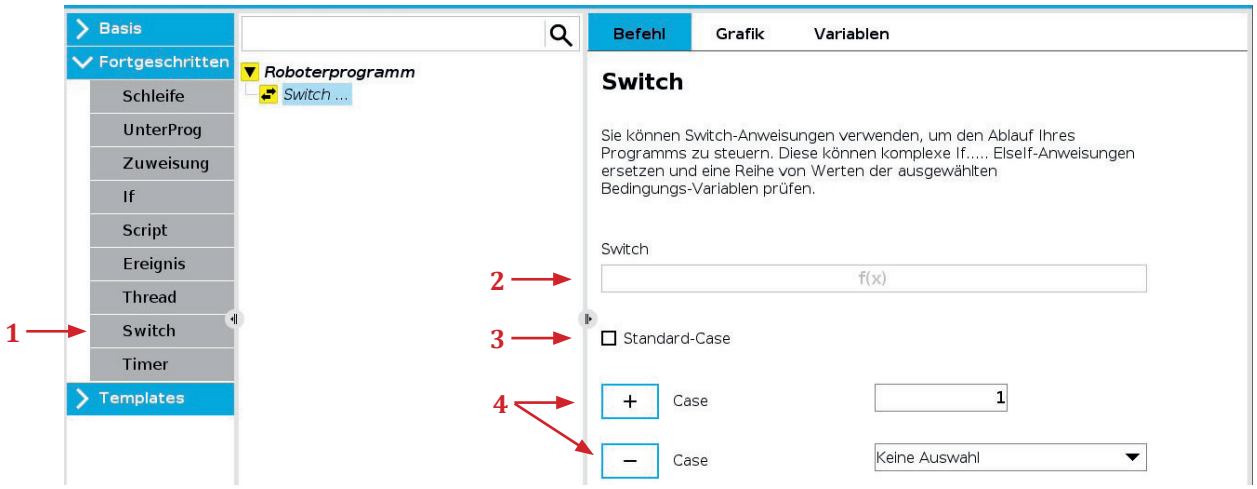
4.1.8.9 Befehl: Thread

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

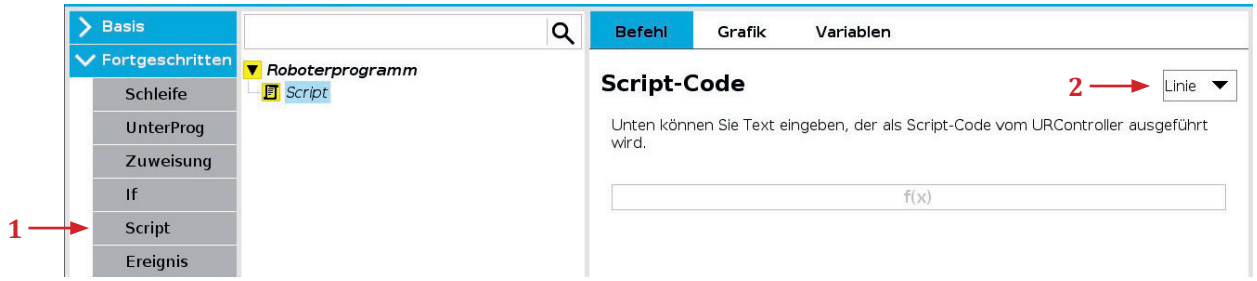
"Switch" ist eine Kontrollstruktur in der Programmierung, die im Vergleich zu "If"-Anweisungen eine oft übersichtlichere Alternative bietet, zwischen verschiedenen Fällen oder Bedingungen zu wechseln und den entsprechenden Programmteil auszuführen, basierend auf dem Wert einer bestimmten Variable oder 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.



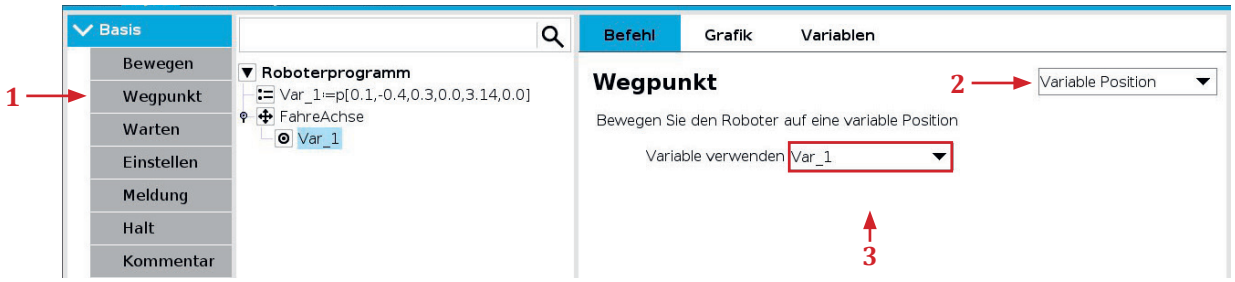
Nr.	Modul	Beschreibung
1	**Script**	Befehl hinzufügen
2	**Option-Auswahl**	Wechsel zwischen „Linie" und „File"

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]]

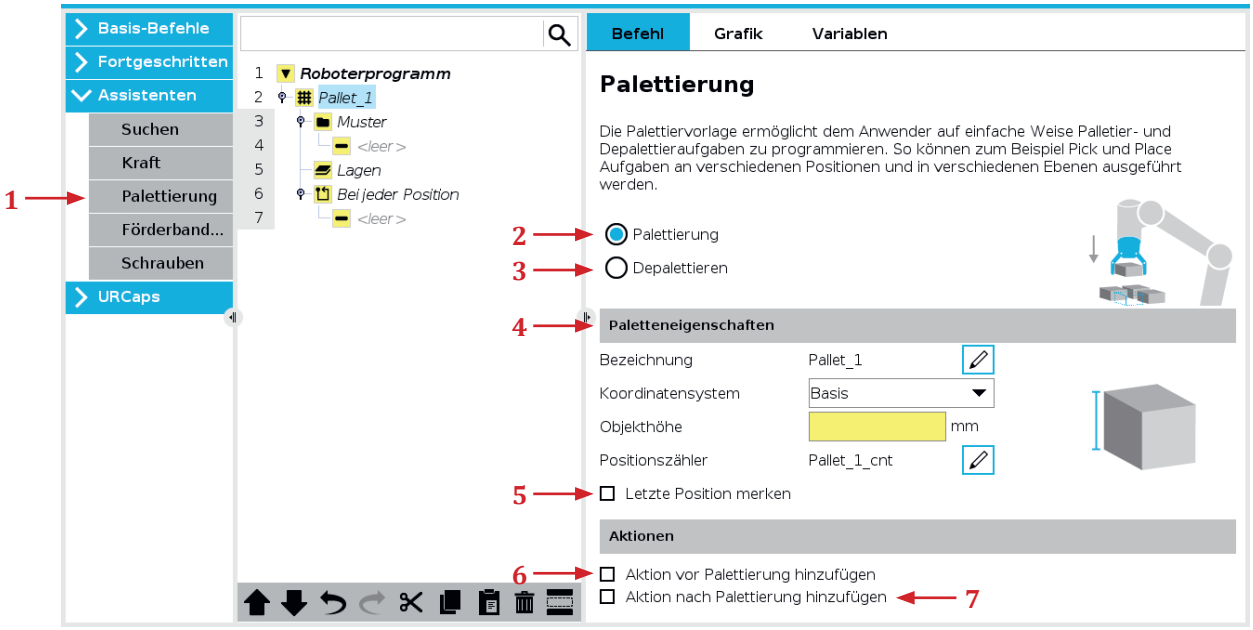
Erklärung:
Zunächst wird die aktuelle Position über *get_actual_tcp_pose()* ausgelesen und über eine Zuweisung in die Variable aktuelle_pos gespeichert. Anschließend wird wieder über einen Zuweisungs-Befehl die Variable neue_pos gespeichert. Hierbei muss die korrekte Syntax einer Pose Variable beachtet 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.

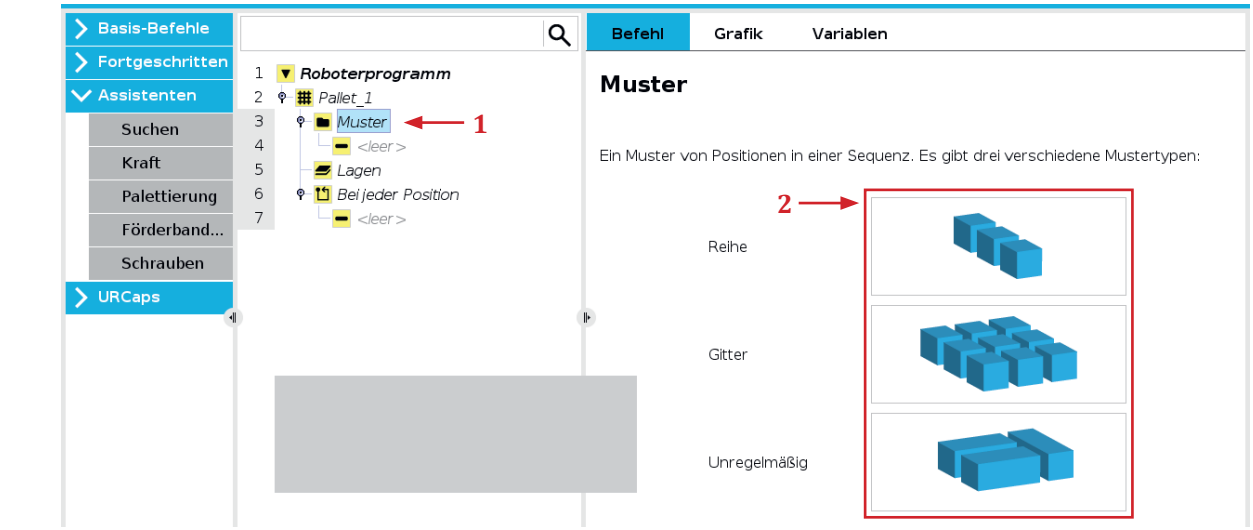
4.1.9 Assistenten

4.1.9.1 Palettierung

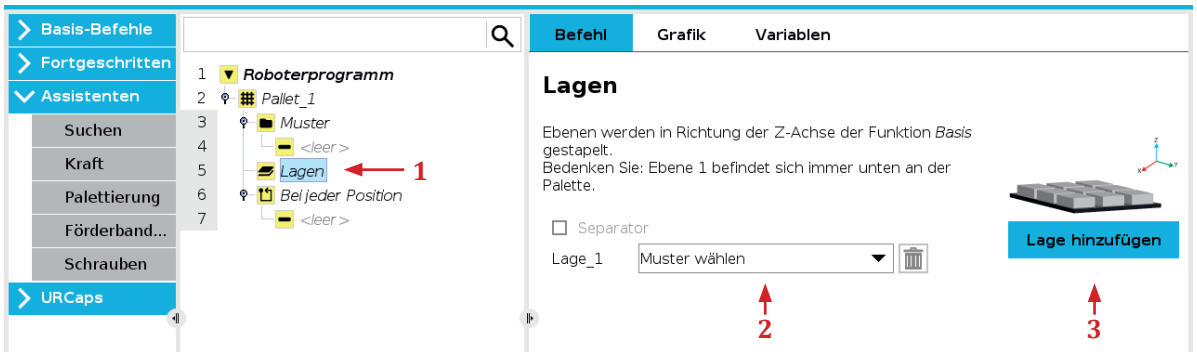
Der Palettierungsassistent unterstützt bei der Palettierung und Depalettierung. Es ist möglich mehrere Lagen und unterschiedliche Muster anzuwenden.



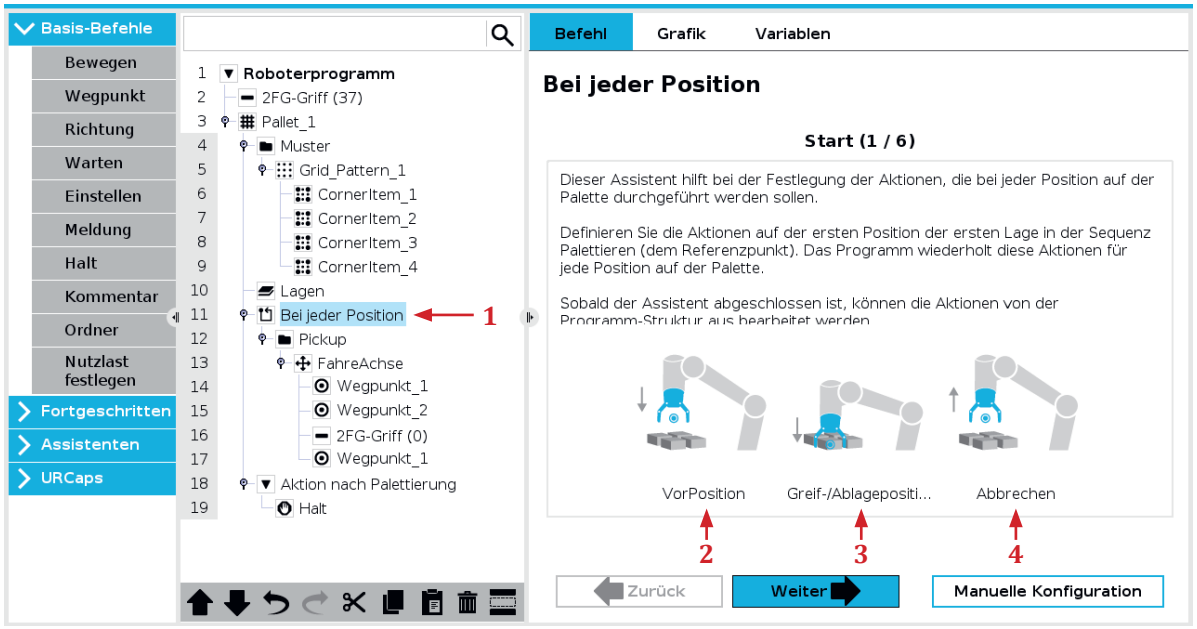
Nr.	Modul	Beschreibung
1	**Palettierung**	Befehl hinzufügen
2	**Palettierung**	Elemente auf einer Palette hinzufügen
3	**Depalettierung**	Elemente von einer Palette entfernen
4	**Paletteneigenschaften**	Setzen der Bezeichnung des verwendeten Koordinatensystems, der Objekthöhe und des Names des Positionszählers.
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.



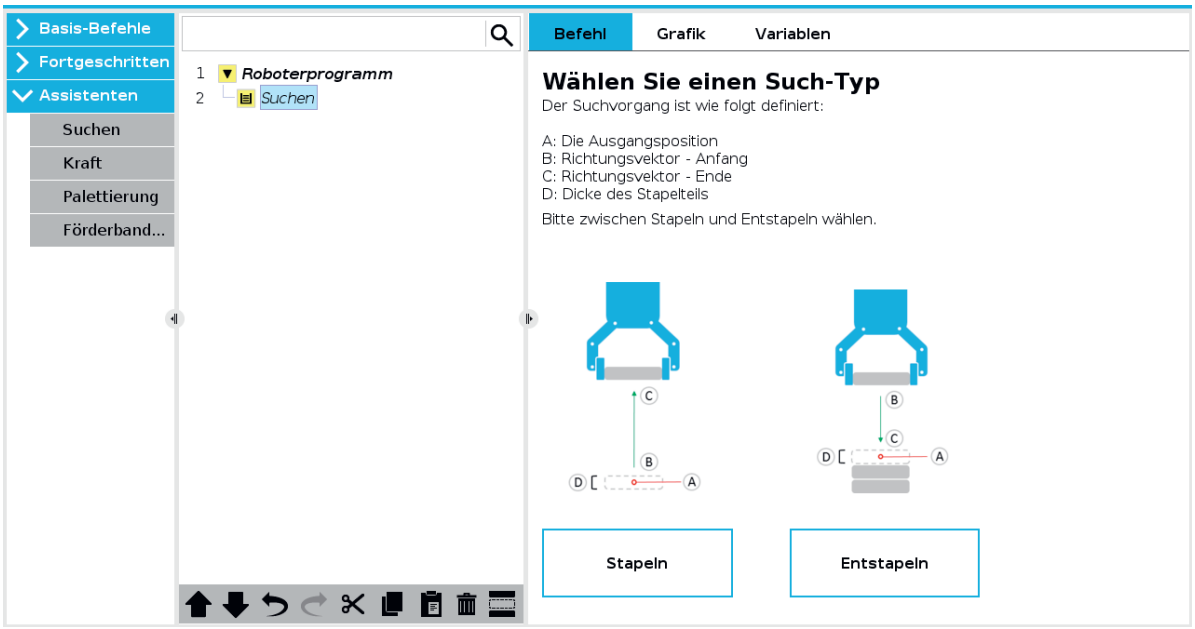
Nr.	Modul	Beschreibung
1	**Lagen**	Konfiguration der Palettierungsschichten
2	**Muster wählen**	Zuordnung eines Palettierungsmustern zu einer Schicht, Unterschiedliche Muster pro Schicht möglich.
3	**Lage hinzufügen**	Die gewünschte Anzahl an Palettierungsschichten hinzufügen



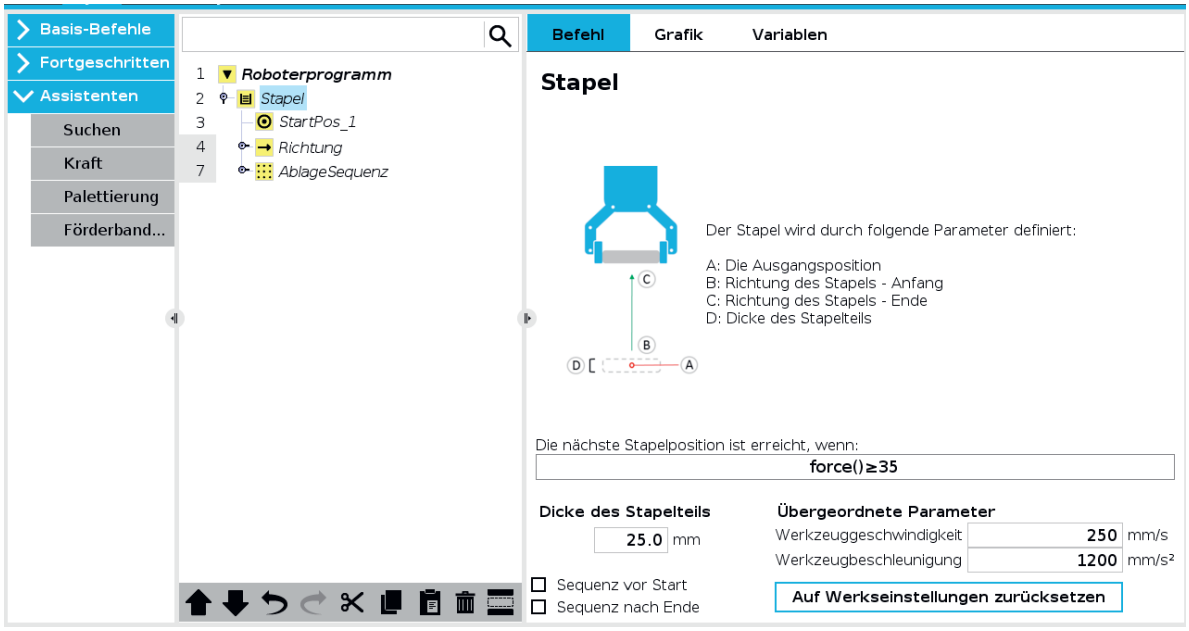
Nr.	Modul	Beschreibung
1	**Bei jeder Position**	Definition der Bewegungsroutine, die bei jeder Position ausgeführt wird. Bspw. das Greifen und kollisionsfreie Einfügen in die aktuelle Palettierungsposition. Ein mehrschrittiger Assistent hilft dabei die Positionen zu erzeugen.
2	**Vorposition**	Wegpunkt, 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

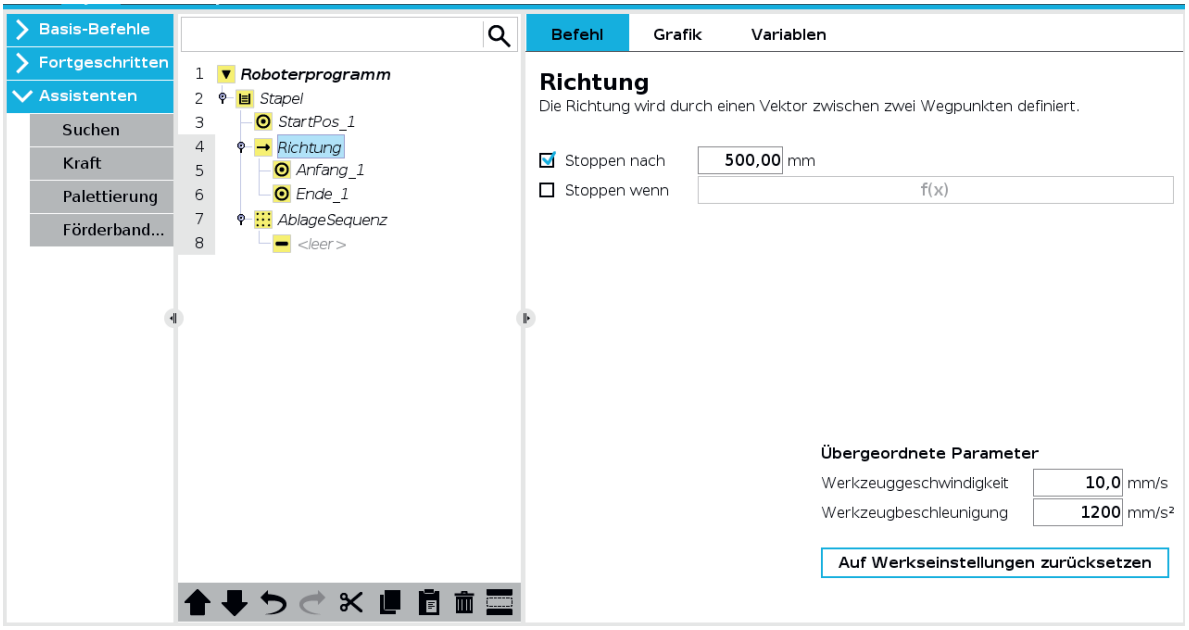
Der Stapel bzw. Entstapelassistent befindet sich im Reiter Suchen. Der Assistent vereinfacht die Aufgabe des Stapelns bzw. Entstapelns.



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.

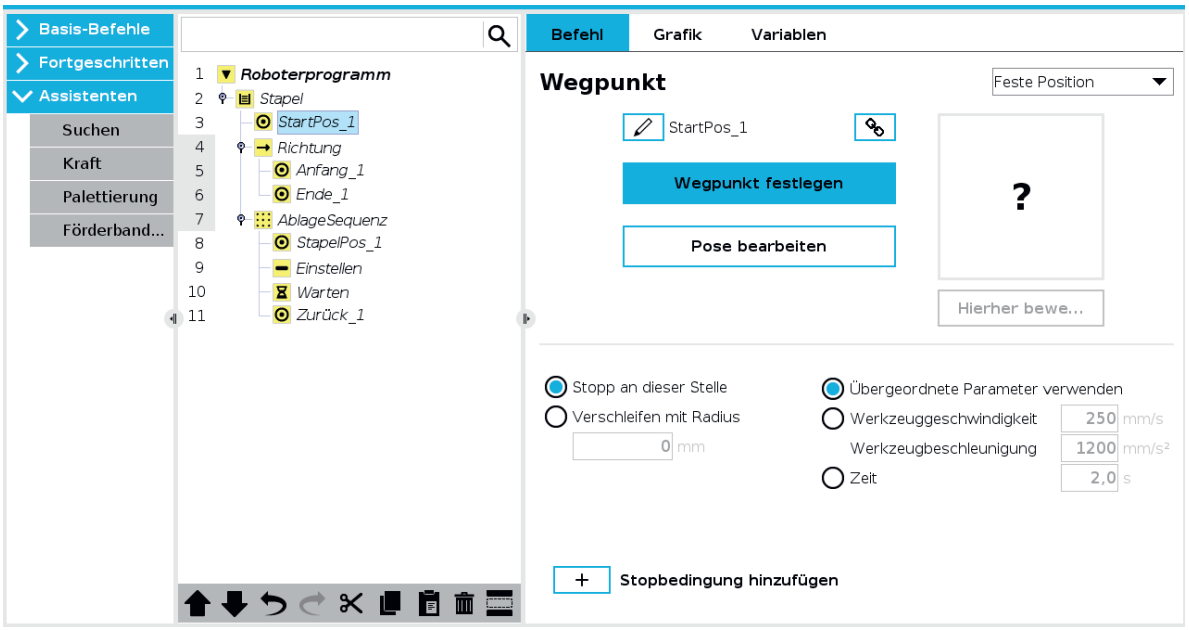


Um den Stapelvorgang vollständig zu konfigurieren muss im Knoten Richtung eine maximale Stapelhöhe oder Abbruchbedingung festgelegt werden.

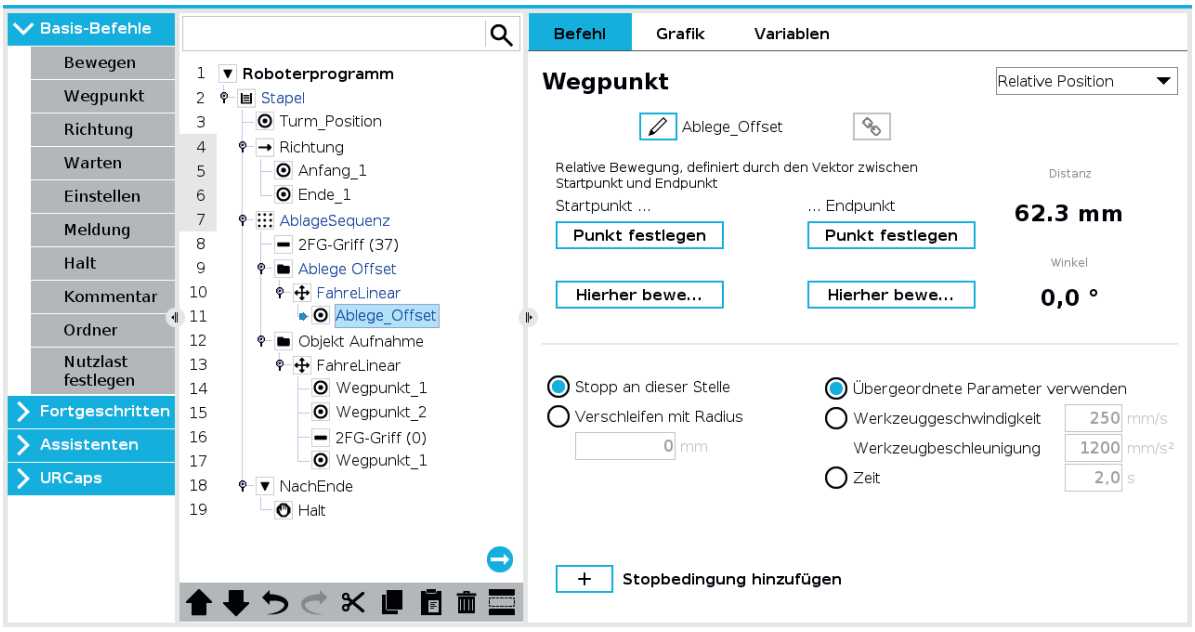


4.1.9.2.1 Beispielprogramm Stapeln

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

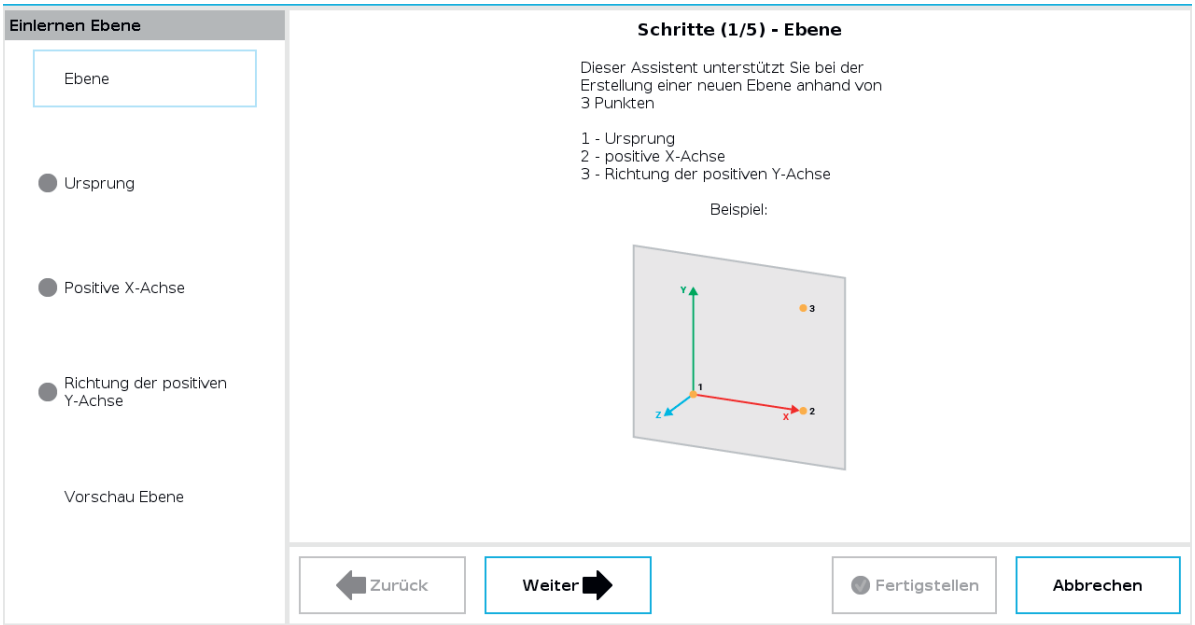
Sicherheitsebenen ermöglichen die Definition von Sicherheitszonen im Arbeitsraum des Roboters, die bei Betreten oder Verlassen unterschiedliche Sicherheitsmaßnahmen, wie bspw. den reduzierten Modus, auslösen. Folgende Sicherheitsmaßnahmen können beim Durchstoßen einer Ebene konfiguriert werden:

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.

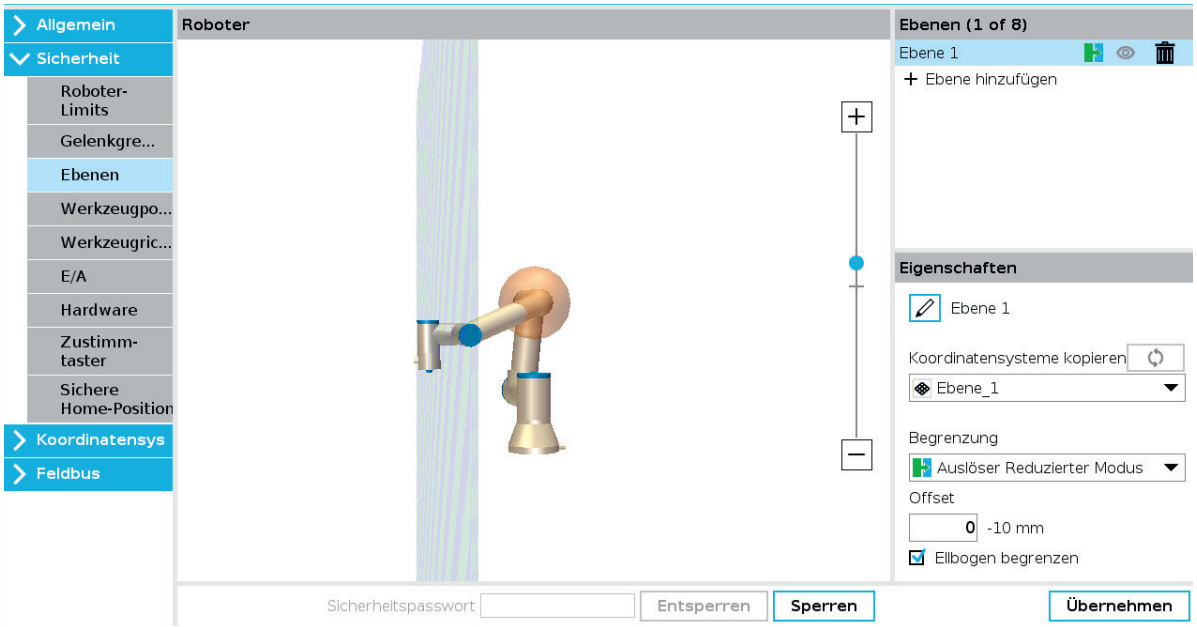
4.1.10.6.1 Definition einer Sicherheitsebene

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

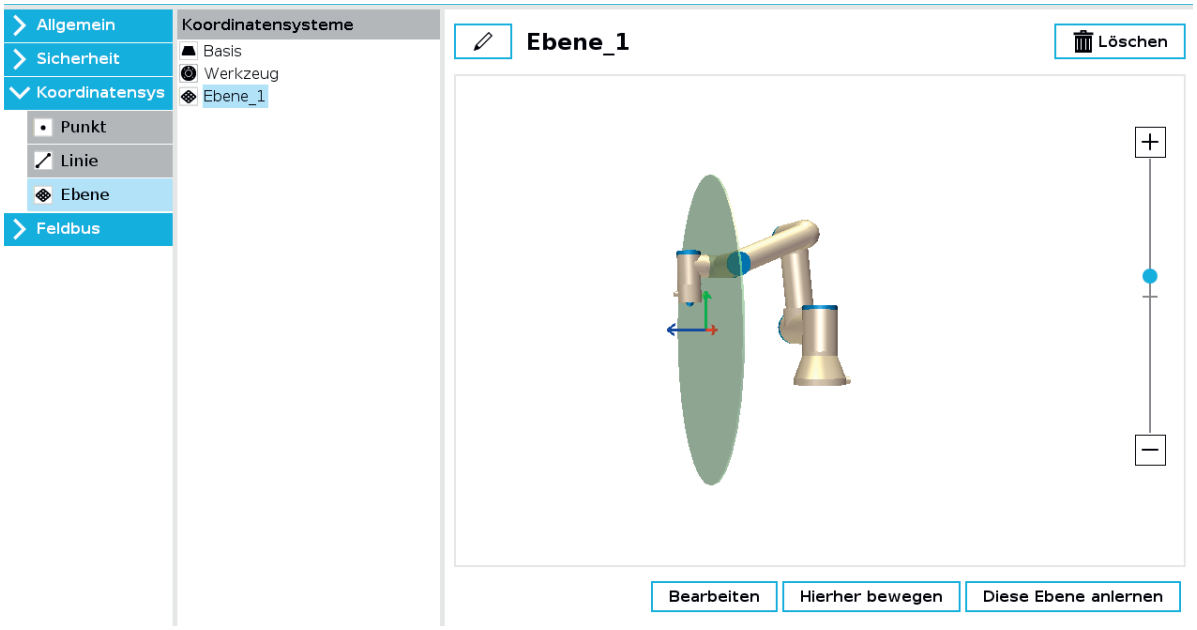
Ist die Erstellung der Ebene abgeschlossen kann diese für die Konfiguration einer Sicherheitsebene verwendet werden. Dazu muss der Tab Ebenen unter Installation → Sicherheit → Ebenen aufgerufen werden. Nach dem Eingeben des Sicherheitspassworts und dem Entsperren des Roboters kann im oberen rechten Bereich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

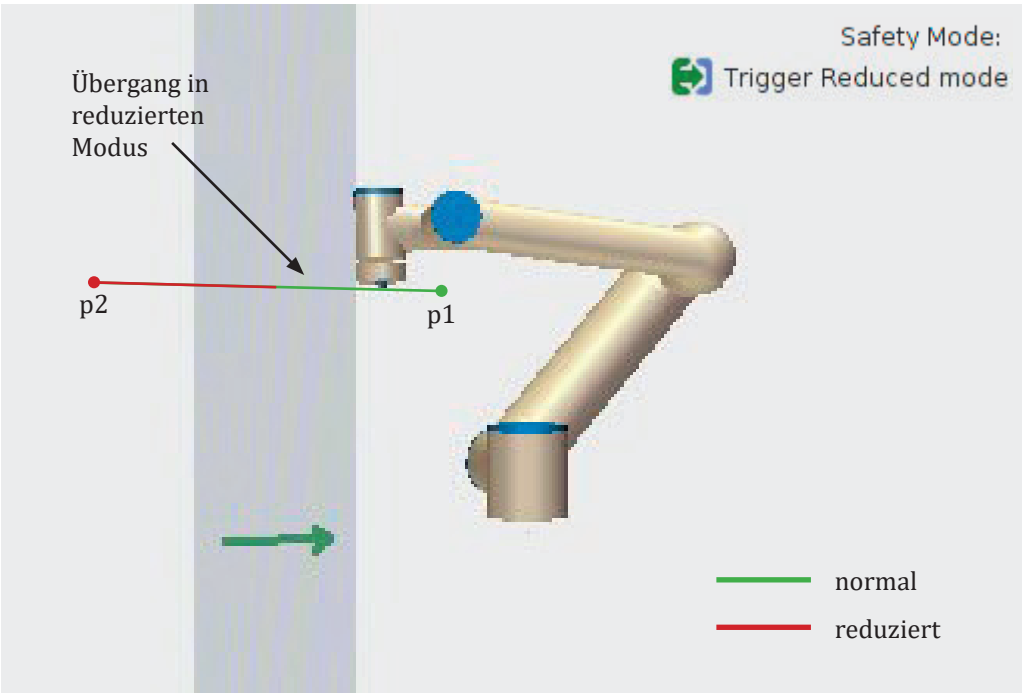
4.1.10.6.3 Wirkrichtung einer Sicherheitsebene

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.