Versuche 1 und 2: Aufbau und Erprobung eines Transport- und Fertigungsprozesses

Ruhr-Universität Bochum

Lehrstuhl für Automatisierungstechnik und Prozessinformatik

Prof. Dr.-Ing. Jan Lunze

19. April 2013

Inhaltsverzeichnis

1. Einleitung

Ein typisches Problem der Robotik besteht darin, Objekte mit Hilfe von Industrierobotern zu bewegen oder zu bearbeiten. Thema dieses Praktikumsversuchs ist die Nachbildung eines Lade- und Transportvorgangs. Um die Programmierung zu erleichtern, wird statt eines standardmäßigen industriellen Systems ein Versuchsaufbau der Firma Fischertechnik verwendet.

1.1. Ziel des Versuchs

Ziel des Versuchs ist es, einen Industrieroboter (im Folgenden als "Greifroboter" bezeichnet) so zu programmieren, dass er ein Transportfahrzeug mit mehreren zylinderförmigen Werkstücken belädt. Das Fahrzeug soll sich auf einer vorgegebenen Bahn bewegen und beim Erreichen einer markierten Beladezone anhalten, sodass es durch den Greifroboter beladen werden kann. Nach Abschluss des Beladevorgangs soll das Fahrzeug freigegeben werden und seine Bewegung fortsetzen. Die Synchronisation von Greifroboter und Fahrzeug erfolgt über eine Lichtschranke und einen zusätzlichen Fototransistor.

In dem Praktikum soll die Trajektorie des Greifroboters vorgegeben und mittels eines MATLAB-Programms zur Koordinatentransformation vorverarbeitet werden. Außerdem ist mit Hilfe des Programms ROBOPro die Steuerung des Greifroboters zu implementieren und die Synchronisation zwischen dem Transportfahrzeug und dem Greifroboter zu realisieren.

1.2. Versuchsaufbau

Der Versuchsaufbau besteht aus einem Fahrzeug mit Ladefläche, dem Greifroboter und der angrenzenden Brücke (siehe Abbildung 1). Der Greifroboter kann Werkstücke greifen und diese auf der Ladefläche des Fahrzeugs ablegen, sofern sich dieses im Ladebereich befindet. Mit Hilfe der Brücke wird die Lichtschranke realisiert, die als Schnittstelle zwischen Greifroboter und Fahrzeug dient und den Ladebereich markiert.

1.2.1. Greifroboter

Der Greifroboter ist mit einer Drehachse, zwei Translationsachsen und einem Greifer ausgestattet. An jeder Achse befinden sich ein Endschalter und ein Impulsgeber. Die Zuordnung der Elemente des Greifroboters zu den Ein-/ und Ausgabeports der Steuerung ist in Abbildung 2 dargestellt. Die Bauelemente an der Brücke sind über das zusätzliche I/O-Interface angeschlossen und unten rechts in der Abbildung aufgezeichnet. Die zu den Achsen gehörigen Parameter sind in Tabelle 1 aufgeführt.

Tabelle 1: Parameter des Greifroboters

		Maximaler Verfahrweg	
Achse	Weg pro Impuls	Impulse	Weg
d	0,575 mm	174	100,05 mm
a	0,575 mm	161	92,575 mm
ϕ	$0,77°$	234	180°
Greifer		geschlossen: 30 Impulse	geöffnet: 52 mm

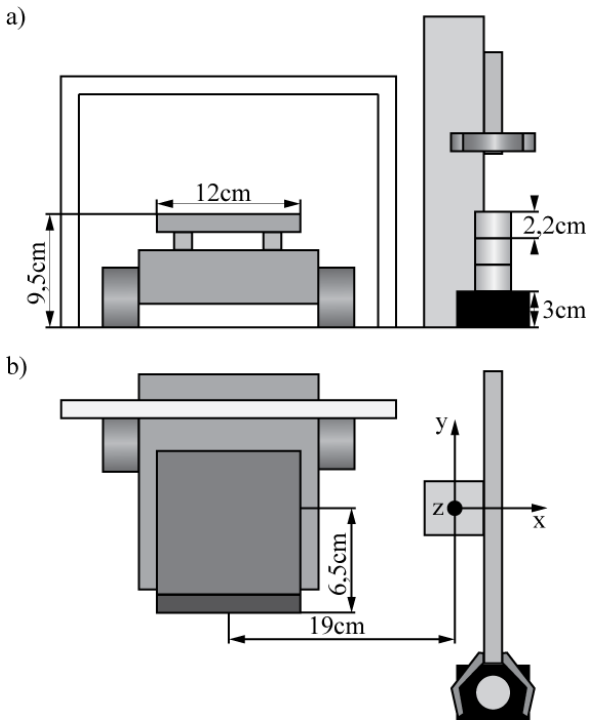


Abbildung 1: Schematische Darstellung des Versuchsaufbaus in Seiten-(a) und Aufsicht(b)

1.2.2. Fahrzeug

Die Interface-Beschaltung des Fahrzeugs ist in Abbildung 3 dargestellt. Der Fototransistor I8 ist an der Seite des Fahrzeugs angebracht und dient als Schnittstelle zum Greifroboter. Die Fototransistoren I4 und I5 befinden sich unter dem Fahrzeug und können folglich genutzt werden, um das Fahrzeug auf der durch eine schwarze Linie vorgegebenen Bahn zu halten.

1.3. Versuchsbeschreibung

Aufgabe des Greifers ist es, das Fahrzeug mit den drei übereinander gestapelten Zylindern zu beladen. Pro Halt des Fahrzeugs soll genau ein Zylinder aufgeladen werden. Nach Starten des Versuchs soll sich der Greifroboter zunächst in einem Wartezustand befinden, während das Fahrzeug der Spur folgt. Bei Unterbrechung der Lichtschranke wird dem Greifroboter signalisiert, dass sich das Fahrzeug im Ladebereich befindet, woraufhin das Fahrzeug angehalten und der Beladevorgang gestartet werden soll. Hierzu beginnt der Greifroboter mit der Abarbeitung einer Liste, wobei die einzelnen Abschnitte der Liste die Trajektorie zum Beladen des Fahrzeugs beinhalten. In diesem Versuch muss die Liste folglich aus drei Abschnitten bestehen, welche jeweils den Beladevorgang für einen Zylinder beschreiben.

Die einzelnen Abschnitte sind durch *Halt*-Markierungen voneinander zu trennen. Nach Erreichen einer *Halt*-Markierung soll der Greifroboter zurück in seine Ausgangsposition fahren und die Lichtschranke kurzzeitig deaktivieren. Durch das nun fehlende Signal an dem Fototransistor kann das Fahrzeug seinen Wartezustand verlassen und das Verfolgen der Spur fortsetzen. Nachdem die drei Zylinder aufgeladen wurden, darf der Greifroboter keine weiteren Aktionen durchführen.

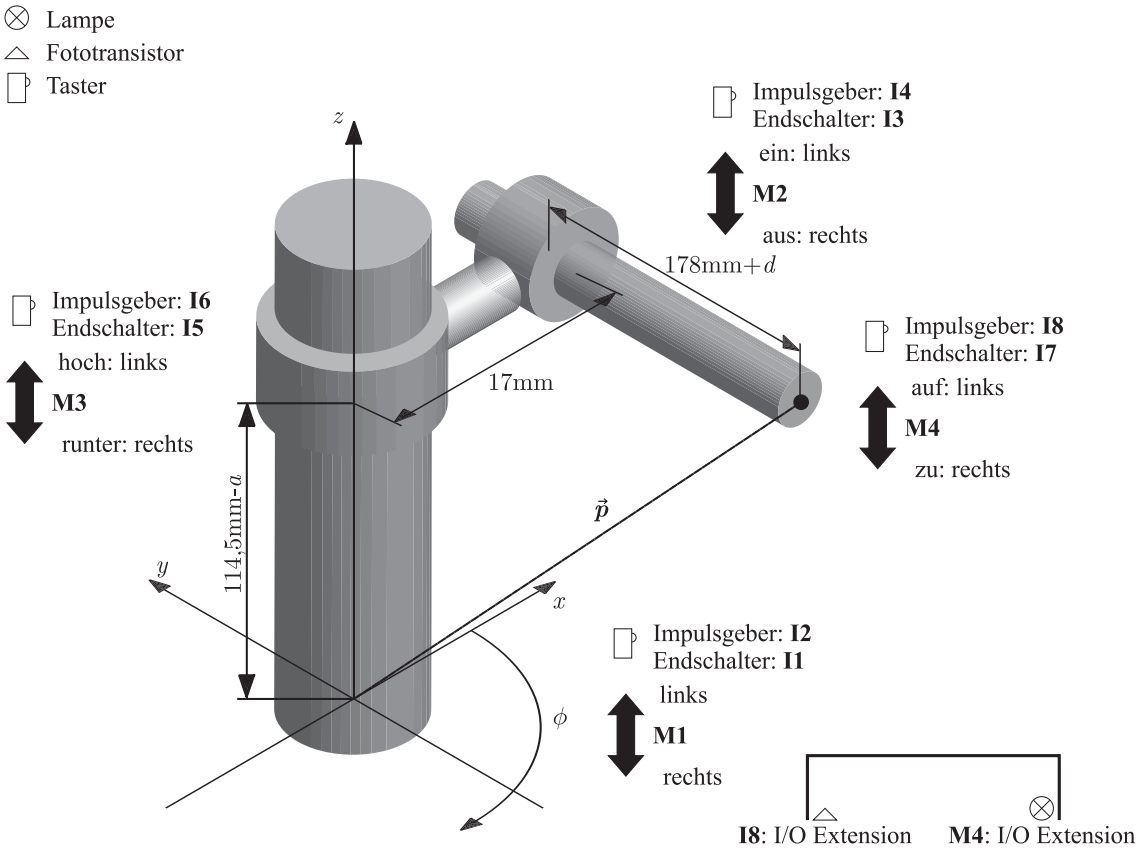


Abbildung 2: Interface-Beschaltung des Greifroboters

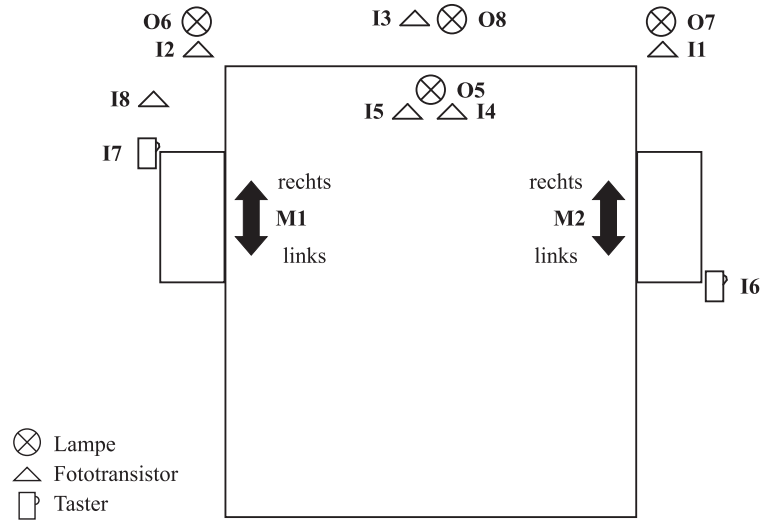


Abbildung 3: Interface-Beschaltung des Fahrzeugs (Aufsicht)

2. Aufgaben zur Vorbereitung des Versuchs

Zur Lösung der Praktikumsaufgaben sind zwei Termine vorgesehen. In Vorbereitung auf den ersten Praktikumstermin muss die gesamte Versuchsbeschreibung durchgearbeitet und verstanden worden sein. Außerdem sind alle Aufgaben in den Abschnitten 2.1 bis 2.3 schriftlich zu lösen. Die Lösung ist bei Praktikumsbeginn vorzuzeigen.

Für den zweiten Termin ist eine schriftliche Aufstellung der bereits bearbeiteten Praktikumsaufgaben, sowie eine Auflistung der noch fehlenden Schritte anzufertigen. Weiterhin müssen die Aufgaben in Abschnitt 2.4 schriftlich gelöst werden.

2.1. Trajektorienplanung

Die folgende Tabelle enthält die Anfahrpunkte zum Aufladen des ersten Werkstücks.

x	y	z	Greifer
17	-178	85	—
—	—	—	Schließen
Referenzpunktfahrt			
-190	0	122	—
—	—	—	Öffnen
Referenzpunktfahrt			

Ergänzen Sie mit Hilfe der schematischen Darstellungen aus Abbildungen 1 und 4 die benötigten Trajektorien (Anfahrpunkte) zum Beladen des Fahrzeugs mit den restlichen Werkstücken. Pro Halt soll immer nur ein Werkstück aufgeladen werden.

2.2. Koordinatentransformation

Um den Greifroboter effektiv programmieren zu können, ist es notwendig, den Zusammenhang zwischen dem kartesischen Weltkoordinatensystem und dem Roboterkoordinatensystem zu kennen. Berechnungsvorschriften zur Bestimmung von solchen Koordinatentransformationen werden in jedem Einführungsbuch zum Thema Robotik in einem der ersten Kapitel behandelt (z.B. in *Robot manipulators*[1]).

Bestimmen Sie

$$\overrightarrow{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = f(a, d, \phi) \text{ und } \begin{pmatrix} a \\ d \\ \phi \end{pmatrix} = g(x, y, z).$$

Die angegeben Werte sind so bemessen, dass der Greifer des realen Roboters das Werkstück mit dem Ortsvektor $\overrightarrow{p}$ direkt greifen kann. Verwenden Sie zur Herleitung die schematische Darstellung des Roboters in Abbildung 4.

2.3. Steuerung des Greifroboters

Die gewünschte Funktionsweise der Steuerung des Greifroboters wird durch das Petrinetz in Abbildung 5 beschrieben. Die Stellen und Transitionen des Petrinetzes sind in Tabelle 2 aufgeführt, die Eingaben in Tabelle 3. Eine Einführung zum Thema Petrinetze ist beispielsweise in dem Buch *Automatisierungstechnik*[2] zu finden.

[1] Paul, Richard P.: *Robot manipulators*. Cambridge, Mass., MIT Press, 1986
[2] Lunze, Jan: *Automatisierungstechnik*. Oldenbourg Wissenschaftsverlag, 2012

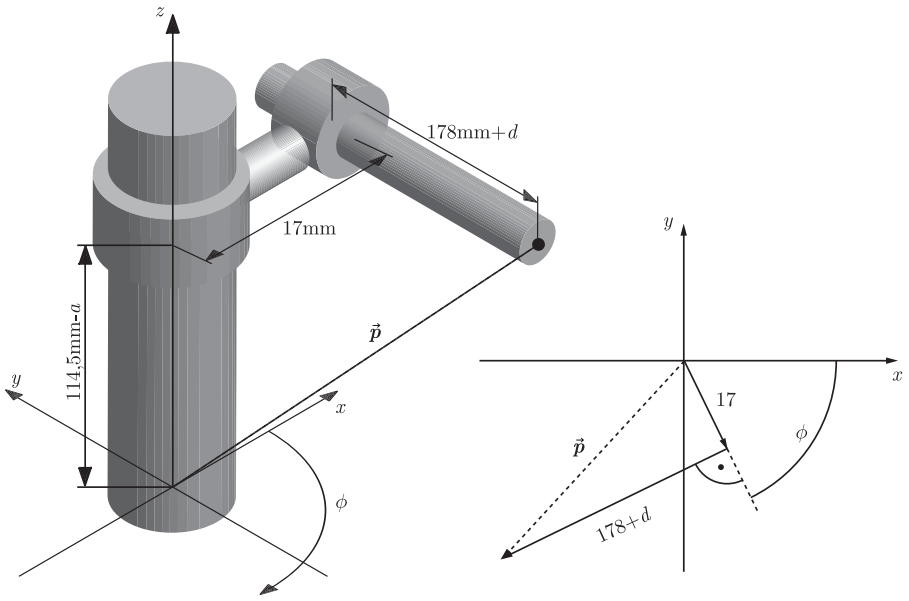


Abbildung 4: Schematische Darstellung des Greifroboters

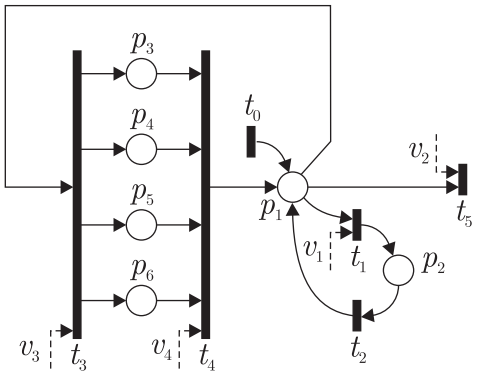


Abbildung 5: Petrinetz zur Beschreibung der Steuerung

Geben Sie die Zustandsraumdarstellung des Petrinetzes in Abbildung 5 an. Bestimmen Sie die Menge der aktivierten Transitionen $\mathcal{T}_{\text{akt}}(\boldsymbol{p}(k), \boldsymbol{v}(k))$, den Transitionsvektor $\boldsymbol{t}(k)$, sowie den Markierungsvektor $\boldsymbol{p}(k)$ für $k = 0, 1, 2$, wenn $v(0) = v_3,\ v(1) = v_4,\ v(2) = v_1$ und $M(0) = p_1$ gilt und interpretieren Sie die Ergebnisse physikalisch.

Tabelle 2: Stellen und Transitionen der Steuerung

Stellen	
p_1	Listeneintrag auslesen
p_2	Führt zusätzliche Aktionen durch
p_3	Achse a wird verfahren
p_4	Achse d wird verfahren
p_5	Achse ϕ wird verfahren
p_6	Greifer wird verfahren

Transitionen	
t_0	Steuerung wird aufgerufen
t_1	Listeneintrag ausgelesen
t_2	Zusätzliche Aktionen ausgeführt
t_3	Listeneintrag ausgelesen
t_4	Verfahren der Achsen abgeschlossen
t_5	Listeneintrag ausgelesen

Tabelle 3: Eingaben der Steuerung

Eingaben	
v_1	Listeneintrag ist *Halt*-Markierung
v_2	Listeneintrag ist *Ende*-Markierung
v_3	Listeneintrag ist neuer Wegpunkt
v_4	Achsen auf Sollposition

2.4. Synchronisation zwischen Greifroboter und Fahrzeug

Wie zuvor beschrieben, soll das Fahrzeug von dem Greifroboter in jedem Durchgang mit jeweils einem Zylinder beladen werden. Erweitern Sie das in Abbildung 5 angegebene Petrinetz an der Stelle p_2 durch die Beschreibung des Fahrzeugs.

Abbildung 6 zeigt das in ROBOPro zur Verfügung gestellte Basisprogramm des Fahrzeugs. Beschreiben Sie, wie das Programm dafür sorgt, dass das Fahrzeug einer vorgegebenen schwarzen Linie folgt.

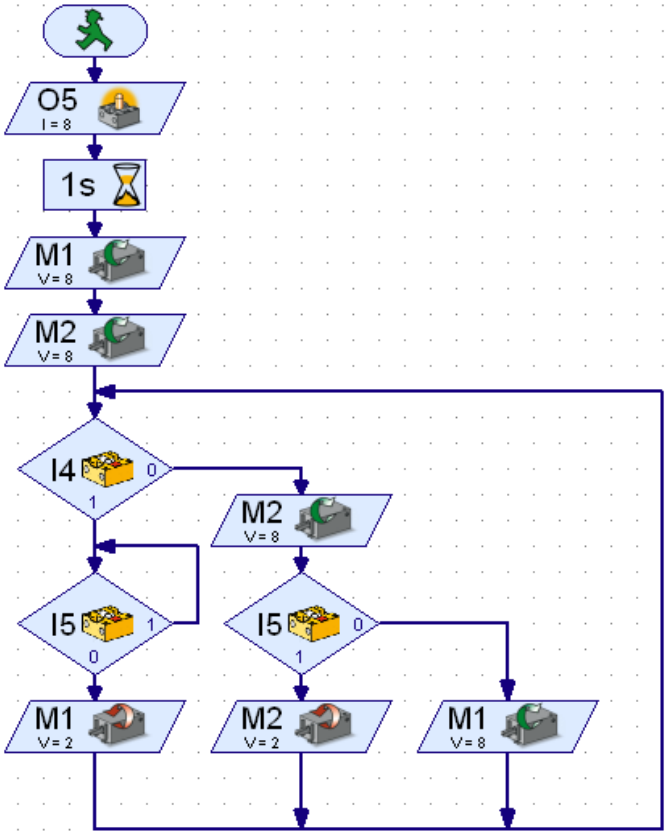


Abbildung 6: Basisprogramm des Fahrzeugs

3. Programmieren in ROBOPro

Die Programmierung der Roboter erfolgt in ROBOPro, einer blockorientierten Entwicklungsumgebung. Eine kostenlose Version von ROBOPro ist auf der Fischtechnik Internetseite `www.fischertechnik.de` verfügbar.

3.1. Einstellungen

Der Zugriff auf Programmelemente ist in ROBOPro durch das eingestellte Level beschränkt. In diesem Praktikum muss in der ROBOPro-Software im Menü Level der Punkt "Level 3: Variablen" ausgewählt werden. Weiterhin muss als Umgebung das "ROBO Interface" benutzt werden. Unter der Schaltfläche Schnittstellenoptionen muss "USB/Bluetooth" als Schnittstelle ausgewählt und ebenfalls der Punkt "ROBO Interface" markiert werden.

3.2. Programmelemente

Alle verfügbaren Elemente sind in der linken Spalte der Bedienoberfläche abgelegt. Durch einfaches Klicken und Ziehen lassen sie sich auf der Arbeitsfläche positionieren. Alle konfigurierbaren Elemente verfügen über ein Eigenschaftsfenster, das durch Rechtsklick auf das Element aufgerufen wird. Jeder Prozess besteht aus einem Fließbild mit einem Start- und einem Stop-Element (Abbildung 7). Das ROBO-Interface auf den Robotern, das die erstellten Programme ausführt, ist in der Lage parallele Prozesse ablaufen zu lassen.

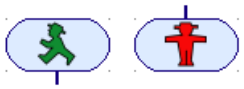


Abbildung 7: Start- und Stop-Element

Eine Übersicht über alle wichtigen Programmelemente sowie ein weiteres Beispielprogramm befinden sich im Anhang. Im Folgenden wird die Verwendung von Unterprogrammen, Variablen und Listen ausführlich behandelt.

3.3. Verwendung von Unterprogrammen und Variablen

Unterprogramme erhöhen die Übersichtlichkeit von Programmen und vermindern bei häufig benötigten Programmelementen den Arbeitsaufwand. Durch die Schaltfläche Neues Unterprogramm erzeugen wird ein neues Unterprogramm in einem neuen Reiter geöffnet. Unterprogramme werden durch die Elemente Unterprogramm-Eingang und Unterprogramm-Ausgang betreten bzw. verlassen (Abbildung 8). In ROBOPro können fertige Unterprogramme unter dem Eintrag Geladene Programme ausgewählt und eingefügt werden. Ein Unterprogramm kann über mehrere Ein- und Ausgänge verfügen. Für jeden Ein- bzw. Ausgang wird ein Anschluss an dem Block des Unterprogramms erzeugt.

Abbildung 8: Unterprogramm-Eingang und Unterprogramm-Ausgang

Prinzipiell gibt es drei Möglichkeiten Unterprogramme aufzubauen, welche in Abbildung 9 dargestellt sind. Die Programme in dem Beispiel schalten die Lampe O1 für 10 s ein und anschließend wieder aus. Ganz links ist die Realisierung ohne Unterprogramme angegeben. Bei UP1 befinden sich das Start- und das Stop-Element im Hauptprogramm.

Das Unterprogramm wird einmal durchlaufen, dann wird das Hauptprogramm weiter abgearbeitet. Das Unterprogramm UP2 kann lediglich aufgerufen werden, da es über keinen Ausgang verfügt. In diesem Beispiel wird es bei Erreichen des Stop-Elements beendet. Es wäre auch die Überführung in eine Schleife möglich, so dass die Lampe bis zum Abschalten den ROBO-Interfaces blinkt. Die dritte Variante ist zu vermeiden. Verwendet man mehrere Unterprogramme dieser Art führt dies zu Problemen in der Ausführung, insbesondere bei Unterprogrammen, die selbst weitere Prozesse starten. **Unterprogramme müssen also über mindestens einen Eingang über den sie aufgerufen werden verfügen und dürfen selbst eigene Prozesse starten.**

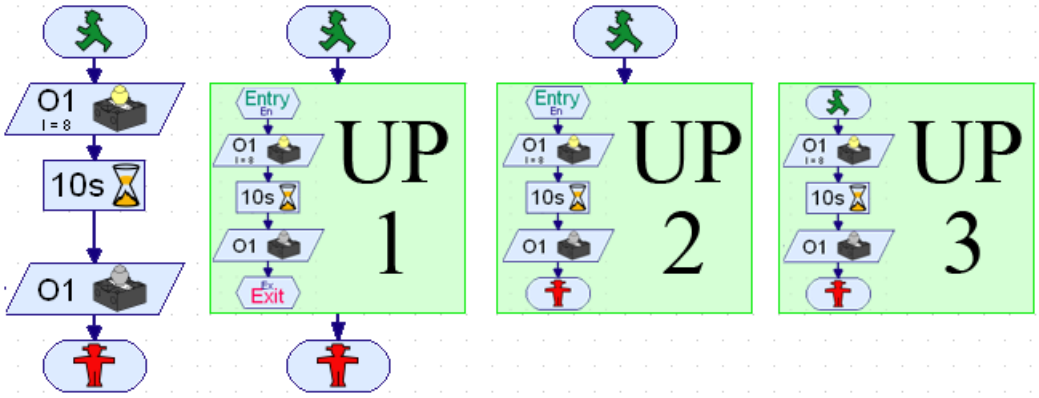


Abbildung 9: Aufbau von Unterprogrammen

Das zuvor erstellte Unterprogramm UP1 lässt eine fest vorgegeben Lampe aufleuchten. Nun wird es derart erweitert, dass es von außen mit einem beliebigen Lampen-Element beschaltet werden kann. Das neue Unterprogramm wird als `Blinken` gespeichert und ist in Abbildung 10 dargestellt.

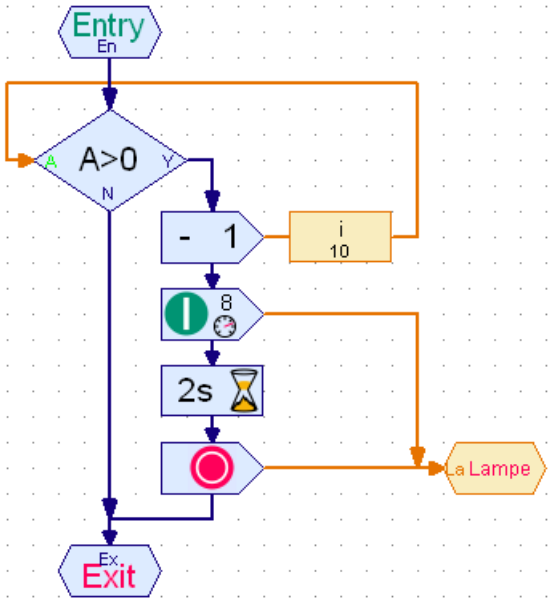


Abbildung 10: Unterprogramm `Blinken`

Zum Einstellen der Lichtstärke werden die Elemente Ein und Aus verwendet (Abbildung 11). In ihrer Funktionsweise entsprechen sie einer Konstanten mit dem Wertebereich 0 (für eine ausgeschaltete Lampe) bis 8 (für die maximale Leuchtstärke). Die Leuchtstärke kann über das Eigenschaftsfenster des Ein-Elements eingestellt werden.

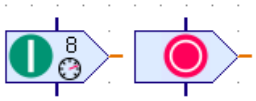


Abbildung 11: Programmelemente Ein und Aus

Als Schnittstelle zum Hauptprogramm benötigt das Unterprogramm einen Unterprogramm-Befehlsausgang, der in Abbildung 10 mit "Lampe" bezeichnet wird. Zusätzlich wird eine globale Variable i eingeführt, mit deren Hilfe die Zahl der Durchläufe festgelegt wird. Sie wird in jedem Durchgang dekrementiert. Sobald ihr Wert gleich 0 ist, führt das Unterprogramm keine Aktionen mehr durch.

Variablen werden in ROBOPro über ihren Namen aufgerufen. Verwendet man zwei verschiedene Variablen-Elemente mit demselben Namen, so verweisen sie immer auf die gleiche Speicherzelle, auch wenn die Elemente keine logische Verbindung besitzen. Globale Variablen werden bei Programmstart mit ihren Anfangswerten initialisiert und sind bis zum Programmende gültig.

Abbildung 12 zeigt das Hauptprogramm, von dem das Unterprogramm `Blinken` aufgerufen wird. Zwei Exemplare von `Blinken` werden in eine Schleife eingebunden und verwendet, um zwei Lampen anzusteuern. Hierzu wird jeweils ein Lampenausgang-Element eingebunden. Diese und alle weiteren E/A-Elemente befinden sich unter dem Eintrag Eingänge/Ausgänge.

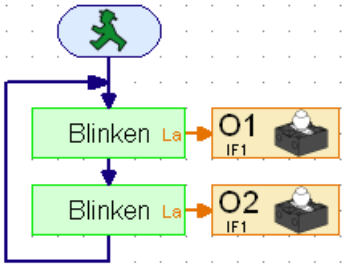


Abbildung 12: Hauptprogramm

In dem angegebenen Programm greifen beide Exemplare von `Blinken` auf die gleiche Variablen i bzw. auf die gleiche Speicherzelle zu. Das bedeutet, dass pro Schleifenumlauf i um zwei dekrementiert wird. Mit Hilfe von lokalen Variablen kann jedem Exemplar eine eigene Variable i zur Verfügung gestellt werden. Im Gegensatz zu globalen Variablen existieren lokale Variablen nur während der Laufzeit des Unterprogramms und verweisen, auch bei gleichen Namen, auf eine eigene Speicherzelle. Der Wert der Variablen bleibt bei Verlassen des Unterprogramms nicht erhalten. Wird das Unterprogramm erneut betreten, so wird die Variable mit ihrem Startwert initialisiert. Werte kann man unter Verwendung der Elemente Unterprogramm-Befehlseingang und Unterprogramm-Befehlsausgang an Unterprogramme übergeben und herausführen (Abbildung 13).

Abbildung 13: Unterprogramm-Befehlseingang und -ausgang

Abbildung 14 zeigt die veränderte Version des Unterprogramms `Blinken` und die erweiterte Version des Hauptprogramms. Die Variable i wurde in diesem Beispiel ausgelagert und das Unterprogramm wurde um einen Ein- und Ausgang erweitert. An dieser Stelle

wäre es auch möglich, in dem Hauptprogramm jedem Exemplar des Unterprogramms eine eigene globale Variable zuzuweisen.

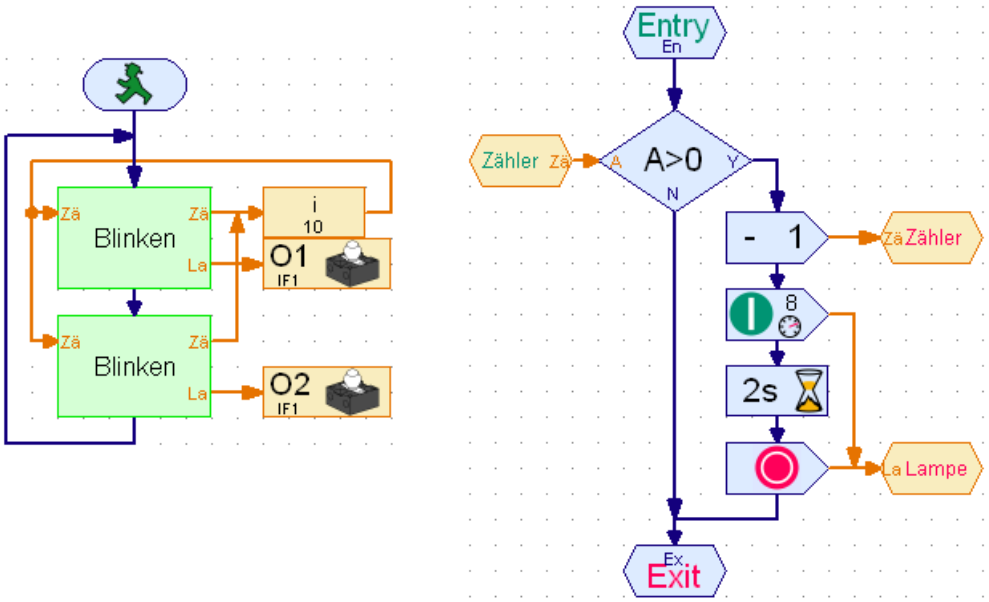


Abbildung 14: Hauptprogramm mit der ausgelagerten Variablen i (links) und das modifiziertes Unterprogramm `Blinken` (rechts)

3.4. Arbeiten mit Listen

Listen stellen eindimensionale Arrays dar und sind in ROBOPro mit zwei Ein- und zwei Ausgängen versehen. Ein Listenelement ist in Abbildung 15 dargestellt. Der Eingang I (Index) nimmt den aktuellen Zeigerwert auf. Mit Hilfe des Zeigers wird das gewünschte Listenelement ausgewählt. Über den Eingang S (Schreiben) wird das aktuell ausgewählte Element beschrieben und über R (Rücklesen) ausgegeben. In dem Eigenschaftsfenster der Liste kann eine externe .csv Datei geladen werden. Bei .csv Dateien handelt es sich um Textdateien, die eine oder mehrere Spalten mit Daten beinhalten. Als Separator zwischen den Spalten können ., ; oder : dienen. ROBOPro ist nur in der Lage die erste Spalte einer .csv Datei auszulesen, daher kann man sich im Folgenden darauf beschränken ausschließlich Listen mit einer Spalte zu erstellen.

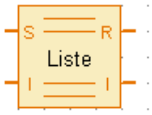


Abbildung 15: Listenelement

Abbildung 16 zeigt ein Beispiel zur Verwendung eines Listenelements. Das Unterprogramm `Blinken` lässt die Lampe O1 In-mal aufleuchten. Die Variable index nimmt den Listenzeiger auf und wird pro Schleifenumlauf um eins inkrementiert.

Die Liste ist mit den in Tabelle 4 angegeben Werten gefüllt. Das Programm arbeitet schrittweise die Liste ab und übergibt den aktuellen Wert jeweils an die Verzweigung A $>$ 0 und an das Unterprogramm `Blinken`. Die Variable index wird mit dem Wert 0 initialisiert. Das Unterprogramm lässt die Lampe, gemäß der Liste, im ersten Durchgang 20-mal, im zweiten Durchgang 15-mal usw. aufleuchten. Die 0 wird in diesem Beispiel als

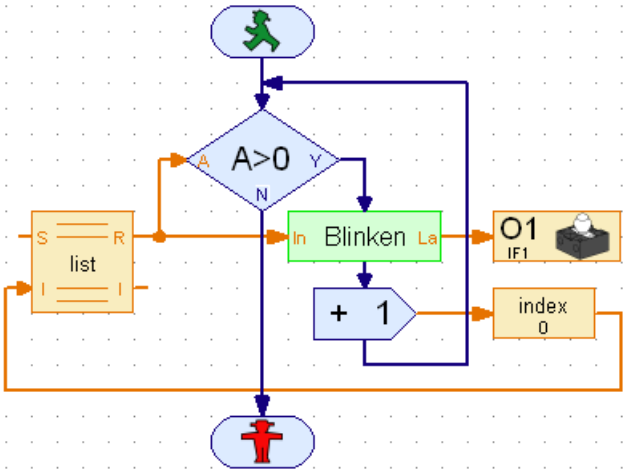


Abbildung 16: Einfache Listenverarbeitung

Ende-Markierung verwendet, d.h., ist das Programm an der 0 angelangt, soll es beendet werden. **Da ROBOPro den letzten Wert einer Liste nicht korrekt ausgibt, wurde eine zusätzliche 0 angefügt.**

Tabelle 4: .csv Datei

20
15
10
5
0
0

4. Praktikumsaufgaben

Das Praktikum ist in drei Aufgaben unterteilt, in denen ein Programm zur Koordinatentransformation, die Robotersteuerung und die Synchronisation zwischen Greifroboter und Fahrzeug realisiert werden sollen.

Auf Grund der eingeschränkten numerischen Fähigkeiten von ROBOPro ist es zweckmäßig, die Koordinatentransformation in MATLAB durchzuführen. Weiterhin erleichtert ein solches Programm die Eingabe von umfangreicheren Trajektorien, welche in ROBOPro einzeln gesetzt werden müssten. Das MATLAB-Programm muss .csv Dateien erstellen und abspeichern. Diese Dateien sollen während der Laufzeit von ROBOPro geladen und interpretiert werden.

Denken Sie daran, alle neu erstellten Programmteile auf Ihre Funktionsfähigkeit zu testen, bevor Sie mit der Bearbeitung eines anderen Aufgabenteils fortfahren. Nur durch dieses Vorgehen ist eine spätere Fehlersuche sowie der erfolgreiche Abschluss des Versuchs in einem angemessenen Zeitrahmen möglich.

4.1. Vorarbeiten

Überlegen Sie in der Gruppe, welche Spezifikationen die Programme erfüllen müssen, beispielsweise wie *Halt*-Markierungen dargestellt und verarbeitet werden oder wie eine Referenzpunktfahrt durchgeführt wird.

Es wird angeraten, sich vor Beginn der Bearbeitung mit dem Verhalten der einzelnen Bauteile vertraut zu machen. Überprüfen Sie hierzu die in Abbildung 2 und Abbildung 3 (Seite 5) dargestellte Beschaltung des Greifers und des Fahrzeugs mit Hilfe der Funktion Interface testen in ROBOPro.

4.2. MATLAB-Programmierung

Dieser Aufgabenteil befasst sich mit der Programmierung eines MATLAB-Programms zur Koordinatentransformation und zur Überprüfung der Einhaltung des Arbeitsbereichs. Es steht Ihnen die Funktion `ftsave.m` zur Verfügung, mit deren Hilfe die Spalten einer übergebenen Matrix getrennt in je eine .csv Datei gespeichert werden. Das Blockschaltbild der geforderten Arbeitsweise des Programms ist in Abbildung 17 zu sehen.

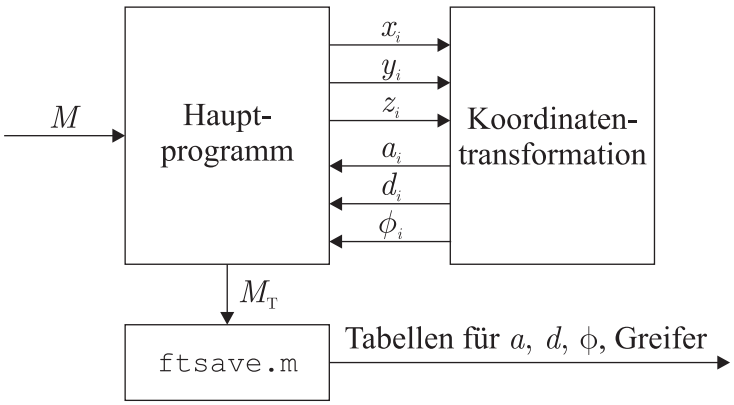


Abbildung 17: Blockschaltbild des MATLAB-Programms

Ziel ist es, aus einer gegebenen Matrix M die transformierte Matrix M_T zu erhalten. Die Matrix M muss die Anfahrtspunkte in kartesischen Koordinaten in der Form

$$M = \begin{pmatrix} x_0 & y_0 & z_0 & \text{Greifer}_0 \\ \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & z_n & \text{Greifer}_n \end{pmatrix}$$

beinhalten.

Die transformierten Werte werden in der Matrix M_T abgelegt, welche die Form

$$M_T = \begin{pmatrix} a_0 & d_0 & \phi_0 & \text{Greifer}_0 \\ \vdots & \vdots & \vdots & \vdots \\ a_n & d_n & \phi_n & \text{Greifer}_n \end{pmatrix}$$

haben soll.

4.2.1. Koordinatentransformation

Erstellen Sie in MATLAB eine Funktion, die aus den Komponenten des Ortsvektors $\overrightarrow{\boldsymbol{p}}$ die Werte für a, d und ϕ berechnet. Verwenden Sie dazu Ihre Ergebnisse aus den Vorbereitungsaufgaben. Berücksichtigen Sie, dass die in ROBOPro zu realisierende Steuerung nur natürliche Zahlen verarbeiten kann. Überprüfen Sie die Funktionalität der Funktion anhand folgender Tabelle:

x	y	z
17	-178	$114,5$
-245	-13	$46,5$
$-178,7$	$-148,1$	$46,5$
$-193,8$	120	$14,5$

$\longrightarrow$

a	d	ϕ
0	0	0
68	67	91
68	54	55
100	49	126

4.2.2. Hauptprogramm

Das zu erstellende Programm soll Wegpunkte und zusätzliche Steuerbefehle aus einer Matrix M erhalten und verarbeiten. Zur Umrechnung der Koordinaten soll die zuvor erstellte Funktion verwendet werden. Im Hauptprogramm muss sichergestellt werden, dass der Roboter durch die berechneten Verfahrwege nicht seinen Arbeitsraum, der durch die Parameter in Tabelle 1 begrenzt ist, verlässt. Die Ergebnismatrix M_T ist mit der Funktion `ftsave.m` zu exportieren. Der Aufruf von `ftsave` erfolgt durch
`ftsave(matrix,{'Name_Spalte1','Name_Spalte 2',···})`.

4.2.3. Speicherung der Trajektorie

Bringen Sie die in den Vorbereitungsaufgaben bestimmte Trajektorie in die für das Programm benötigte Form und transformieren und exportieren Sie die Datensätze mit dem zuvor erstellten Programm.

4.3. Steuerungsprogrammierung

In diesem Aufgabenteil wird schrittweise die Steuerung für den Greifroboter in ROBOPro erstellt. Jede Achse ist mit einem Antrieb, Endschalter und Impulsgeber ausgestattet.

Auf Grund des ähnlichen Aufbaus aller Achsen ist es zweckmäßig, ein Unterprogramm zu erstellen, das sich von außen beschalten und somit für die Steuerung jeder Achse verwenden lässt. Weiterhin soll eine Listenverarbeitung implementiert werden, welche die in MATLAB erzeugten .csv-Dateien auswertet. Die Steuerung soll so strukturiert sein, dass sie durch das in Abbildung 5 angegebene Petrinetz beschrieben wird.

4.3.1. Steuerungsmodul für einen Antrieb

Ziel dieses Aufgabenteils ist es, ein Modul zur Steuerung eines Antriebs zu realisieren und zu testen. Das Modul soll die Istposition der angeschlossenen Achse auf die gewünschte Sollposition bringen. Hierzu ruft es das Unterprogramm `Lageregler` auf, welches für das Verfahren der Achsen zuständig ist. Ihnen wird die Datei `SteuerungA.rpp` zur Verfügung gestellt, in der bereits die benötigten Elementen zur Steuerung der a-Achse verschaltet sind.

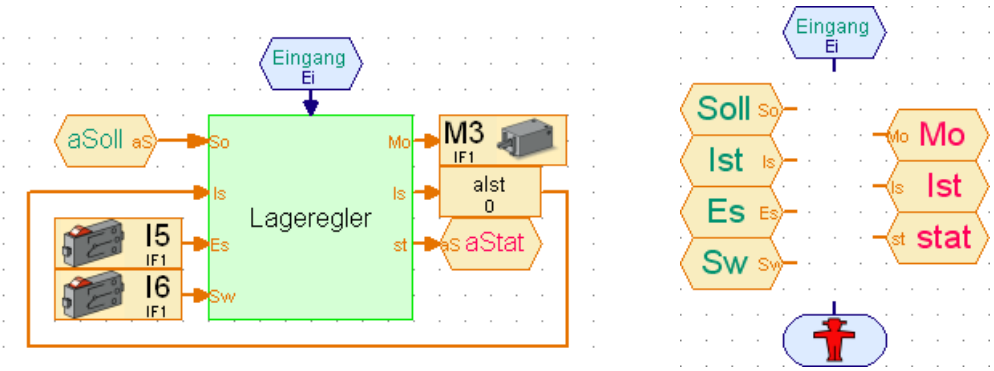


Abbildung 18: Unterprogramm `a` (links) und Unterprogramm `Lageregler` (rechts)

Der für Sie relevante Teil der Datei besteht aus dem Unterprogramm `Lageregler`, dem Unterprogramm `a` und einem `Hauptprogramm`. Alle anderen Unterprogramme dienen nur zum Testen Ihrer Implementierung und sollen nicht weiter verwendet werden. Unterprogramm `a` ist links in Abbildung 18 dargestellt und steuert das Verfahren auf der a-Achse. Die Ausgänge des aufgerufenen Unterprogramms `Lageregler` werden mit Mo, Ist und stat bezeichnet. Der Ausgang Mo ist mit einem Motorausgang-Block beschaltet und spricht den Motor M3 der z-Achse direkt an. Die an den Ist-Ausgang geschaltete Variable alst speichert den aktuellen Istwert.

Mit Hilfe des Ausgangs aStat besteht die Möglichkeit, eine Größe aus dem Unterprogramm `a` herauszuführen. Denkbar wären beispielsweise eine 0 bei Regelabweichung und eine 1 bei Erreichen des Sollwerts. Bei I5 handelt es sich um den Endschalter und bei I6 um den Impulsgeber der a-Achse (vgl. Abbildung 2). Über den Unterprogramm-Befehlseingang aSoll wird dem Unterprogramm `a` von außen ein Sollwert zugewiesen.

Aufgabe ist es, das Unterprogramm `Lageregler`, welches rechts in Abbildung 18 dargestellt ist, fertigzustellen. Nach Aufruf soll es den Motor am Ausgang Mo so lange verfahren bis der Sollwert erreicht ist und sich dann beenden.

Überprüfen Sie die Funktionsweise dieses von Ihnen erstellten Programmabschnittes unter Verwendung des `Hauptprogramms`. Es besteht aus mehreren Abschnitten, welche die Funktionalität des entworfenen Unterprogramms überprüfen. Starten Sie das Programm im Onlinemodus und folgen Sie den Anweisungen.

4.3.2. Einbindung aller Antriebe

Erzeugen Sie für die Steuerung der übrigen Achsen d und ϕ und des Greifers jeweils ein Modul, welches das zuvor erstellte Unterprogramm `Lageregler`, sowie die benötigten

Schnittstellen und Variablen beinhaltet.

4.3.3. Steuerung des Greifroboters

Erstellen Sie ein Unterprogramm, das die in MATLAB erzeugten .csv Dateien verarbeiten kann und für jeden Anfahrpunkt die zuvor erstellten Steuerungsmodule aufruft. Das Unterprogramm sollte über zwei Ausgänge verfügen. Ein Beispiel mit entsprechenden Ausgängen ist in Abbildung 19 gegeben. Der erste Ausgang, im Bild mit en bezeichnet, soll erreicht werden, wenn die Liste komplett abgearbeitet wurde, der zweite, mit ha bezeichnete Ausgang, wenn im Programm eine *Halt*-Markierung erkannt wurde.

Abbildung 19: Steuerungsmodul für den Greifroboter

Abbildung 20 zeigt die gewünschte Verknüpfung der Listen mit den zuvor erstellten Antriebsreglern. In jedem Schritt werden von den Listen die Sollwerte an der Stelle index ausgegeben und von den Antriebsregelungen verarbeitet. Hierzu müssen die Steuerungsmodule zyklisch für jeden Listeneintrag aufgerufen werden. Sobald alle Achsen ihre Sollposition erreicht haben, soll am Ausgang status eine 1 anliegen.

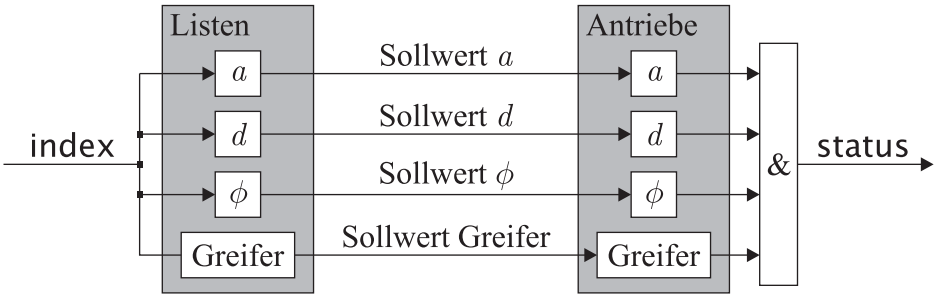


Abbildung 20: Blockschaltbild der Listenverarbeitung

Überprüfen Sie abschließend, ob die von Ihnen vorgegebenen und mit dem MATLAB-Programm transformierten Trajektorien zu einem korrekten Verhalten des Greifroboters führen und korrigieren Sie sie gegebenenfalls.

4.4. Synchronisation zwischen Greifroboter und Fahrzeug

In dieser Aufgabe soll die Beladung des Fahrzeugs durch den Greifroboter sowie die dazu benötigte Synchronisation zwischen den beiden Systemen realisiert werden. Für die Steuerung des Greifroboters ist ein übergeordnetes Programm zu erstellen, in welchem das zuvor implementierte Steuerungsmodul aufgerufen wird. Außerdem ist ein Warte-Zustand einzuführen, der verlassen wird, sobald sich das Fahrzeug in der Beladezone befindet. So lange Werkstücke vorhanden sind, soll der Greifroboter bei jeder Einfahrt des Fahrzeugs ein Werkstück aufladen. Nach Abschluss eines Beladevorgangs soll der Greifroboter in den Warte-Zustand zurückkehren.

Abbildung 21 zeigt das zur Verfügung gestellte Basisprogramm des Fahrzeugs (`Basisprogramm_Transport.rpp`), wobei der rot markierte Bereich eine Eingriffsmöglichkeit darstellt. Das Programm des Fahrzeugs ist ebenfalls um einen Warte-Zustand zu erweitern, der betreten werden soll, sobald das Fahrzeug die Beladestation erreicht hat. Nach Abschluss des Beladevorgangs soll das Fahrzeug zur Spurensuche zurückkehren. Zum Erkennen der

Beladestation kann der Fototransistor I8 verwendet werden, der auf die Lichtschranke an der Brücke des Greifroboters reagiert.

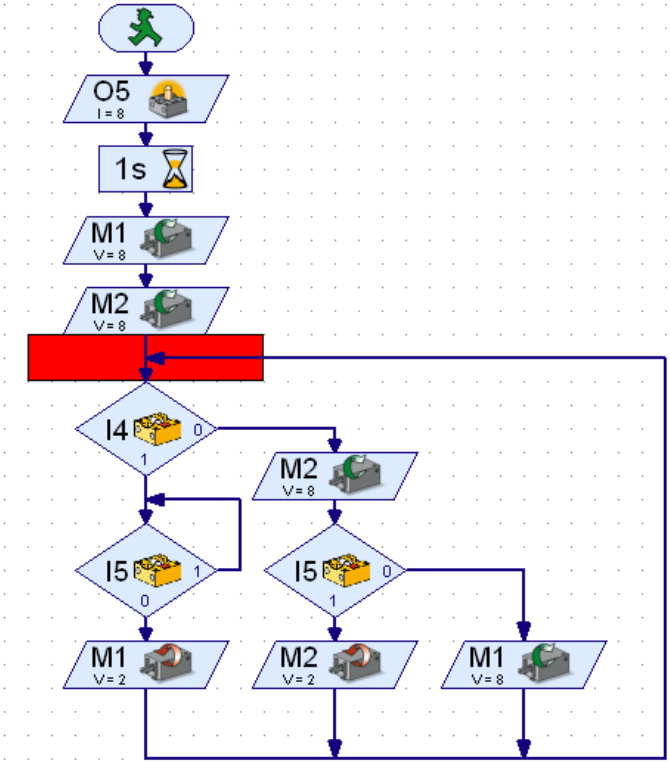


Abbildung 21: Basisprogramm des Fahrzeugs

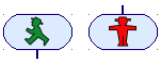
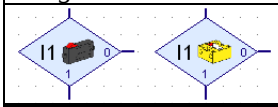
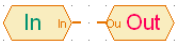
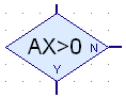
5. Aufgaben zur Nachbereitung des Versuchs

Zur Nachbereitung des Versuchs ist ein schriftliches Versuchsprotokoll anzufertigen, in dem alle während des Versuchs gewonnenen Erkenntnisse festgehalten werden. Beachten Sie dazu die Hinweise auf der Internetseite der Veranstaltung.

Beantworten Sie im Rahmen Ihres Berichts die folgenden Fragen:

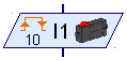
- Wie sieht die Matrix `MT` aus, die Ihr MATLAB-Programm in die .csv Dateien speichert, wenn es eine Matrix `M` mit den Werten aus der linken Tabelle in Abschnitt 4.2.1 verarbeitet? Ergänzen Sie die vierte Spalte (mit den Werten für den Greifer) entsprechend den von Ihnen festgelegten Werten.
- Wie haben Sie *Halt*- und *Ende*-Markierungen codiert?
- Wurde die Steuerung tatsächlich exakt gemäß dem Petrinetz in Abbildung 5 implementiert?
  - Wenn ja, welche Stellen/Transitionen entsprechen welchen Programmteilen?
  - Wenn nein, wie sieht das Petrinetz zu der von Ihnen implementierten Steuerung aus?
- Gab es grundlegende Probleme bei der Durchführung des Versuchs?
  - Was war die Ursache für diese Probleme?
  - Wie konnten sie behoben werden?

A. ROBOPro Programmelemente

Start und Ende	
Programmelemente / Grundelemente	
	Die Elemente dienen zum Starten und Beenden eines Threads.
Verzweigung Digital	
Programmelemente / Grundelemente	
	Verzweigungen überprüfen den angegebenen digitalen Eingang auf 0 oder 1. Die Ausgänge 0 und 1 sowie das Bild können unter *Eigenschaften* geändert werden.
Unterprogramm-Ein-* und *Ausgang	
Programmelemente / Unterprogramm I/O	
	Die Elemente dienen als Schnittstelle zu übergeordneten Programmen und können unter *Eigenschaften* mit einem Namen versehen werden.
Unterprogramm-Befehlsein-* und *-ausgang	
Programmelemente / Unterprogramm I/O	
	Die Elemente dienen als Schnittstelle für Daten zu übergeordneten Programmen und können unter *Eigenschaften* mit einem Namen versehen werden.
Verzweigung Analog	
Programmelemente / Grundelemente	
	Verzweigungen vergleichen den angegebenen analogen Eingang mit der angegeben Bedingung. Die Ausgänge Y und N können unter *Eigenschaften* geändert werden. Es stehen die Operatoren <, >, ≤, ≥ und <> zur Verfügung.
Wartezeit	
Programmelemente / Grundelemente	
	Das Element realisiert eine Verzögerung. Es sind Eingaben mit zwei Nachkommastellen zulässig. Als Einheit lassen sich Stunden, Minuten und Sekunden einstellen.
Motorausgang	
Programmelemente / Grundelemente	
	Dieses Element steuert die Motorausgänge direkt an. Die Geschwindigkeit ist als ganze Zahl zwischen 0 und 8 anzugeben.
Lampenausgang	
Programmelemente / Grundelemente	
	Dieses Element steuert die Lampen-Ausgänge für gegen Masse geschaltete Lampen direkt an. Die Leuchtkraft ist als ganze Zahl zwischen 0 und 8 anzugeben.
Warten auf Eingang	
Programmelemente / Grundelemente	
	Das Element unterbricht den Programmfluss bis am angegebenen digitalen Eingang die eingestellte Bedingung erfüllt ist. Einstellbar ist 1, 0, steigende und fallende Flanke.

Impulszähler

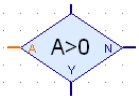
Programmelemente / Grundelemente



Das Element unterbricht den Programmfluss bis eine vorgegebene Menge von Impulsen am angegebenen digitalen Eingang registriert wurden. Der Impulszähler reagiert auf steigende oder fallende Flanke oder auf beide.

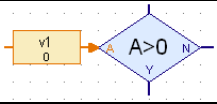
Verzweigung (mit Dateneingang)

Programmelemente / Verzweigung, Warten, …



Verzweigungen vergleichen den an *A* angelegten Wert mit der angegeben Bedingung. Die Ausgänge Y und N können unter *Eigenschaften* geändert werden. Es stehen die Operatoren <, >, ≤, ≥ und <> zur Verfügung.

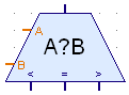
Beispiel:



Der Wert der Variablen *v1* muss dem Ausdruck *v1*>0 genügen.

Vergleich

Programmelemente / Verzweigung, Warten, …



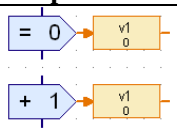
Das Element vergleicht die zwei an *A* und *B* angelegten Werte und überprüft dabei die Ausdrücke *A*<*B*, *A*=*B* und *A*>*B*.

Globale* und *lokale Variable

Programmelemente / Variablen, Timer, …

Das Element erzeugt eine Variable. Über das Eigenschaftsfenster lässt sich der Name und der Typ der Variable, global oder lokal, angeben.

Beispiel:



Bei Erreichen des Zuweisungsoperators *=0* wird der Variablen v1 der Wert 0 zugewiesen.
Bei Erreichen des Zuweisungsoperators *+1* wird der Wert der Variablen v1 um eins inkrementiert.

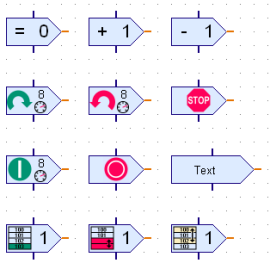
Liste

Programmelemente / Variablen, Timer, …

Das Element *Liste* erzeugt realisiert ein eindimensionales Array. Am Eingang *I* wird die Zeigervariable übergeben, welche die aktuelle Position in der Liste angibt. Über *S* wird ein Wert an die aktuelle Stelle geschrieben und über *R* ausgelesen.

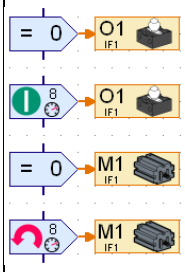
Befehls-Element

Programmelemente / Befehle



Befehl-Elemente dienen zum Steuern von Lampen und Motoren sowie als Zuweisungsoperator für Variablen und Listen. Über das Eigenschaftenfenster sind folgende Befehle möglich: Zuweisen, Plus, Minus, Rechts, Links, Stopp, Ein, Aus, Text, Wert Anhängen, Wert(e) entfernen, Werte vertauschen.

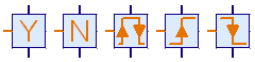
Beispiel:



Der Zuweisungsoperator *=0* legt eine 0 an den Lampenausgang an und schaltet sie damit ab. Er hat die gleiche Wirkung wie *Aus.*
Der Zuweisungsoperator *Ein*=8 schaltet die Lampe O1 mit einer Lichtstärke von 8 ein.
Der Motor M1 wird durch den Zuweisungsoperator *=0* abgeschaltet. *=0* entspricht dem Befehl *Stopp*.
Durch den Befehl *Links=8* dreht der Motor links herum mit einer Geschwindigkeit von 8 (richtiger Anschluss vorausgesetzt). Der Befehl entspricht der Zuweisung *=-8*. Der Befehl *Rechts* entspricht *=8.*

Warten auf…

Programmelemente / Verzweigung, Warten, …



Die Elemente unterbrechen den Programmfluss bis durch den digitalen Eingang eine der folgenden einstellbaren Bedingungen erfüllt ist. Einstellbar sind: *Warten auf 1*, *0*, *steigende Flanke*, *fallende Flanke*, *Flanke*.

Beispiel:

Das Element unterbricht den Programmfluss solange bis an der Photodiode an I1 eine *1* anliegt.

Digitaler Eingang

Programmelemente / Eingänge, Ausgänge

Das Element *Digitaler Eingang* dient als direkte Schnittstelle zu den digitalen Anschlüssen. Im Eigenschaftsfenster lassen sich ein digitaler Eingang und ein Piktogramm auswählen.

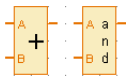
Motor*- und *Lampenausgang

Programmelemente / Eingänge, Ausgänge

Das Element spricht die Motorausgänge direkt an. Das *Motor*-Element dient zur Ansteuerung der Motoren. Das *Lampen*-Element wird zum Ansteuern von gegen Masse geschalteten Lampen verwendet.

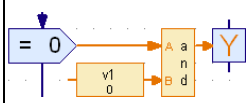
Operator

Programmelemente / Operatoren



Das Element *Operator* dient zum Verknüpfen von Datenquellen (Variablen, Zuweisungsoperatoren und digitalen Eingängen). Mit Hilfe des Eigenschaftsfensters lassen sich 1 bis 24 Eingänge einstellen. Folgende Operationen werden zur Verfügung gestellt: +, -, *, /, =, < >, <=, <, >=, >, and, or, nicht, AND, OR, NOT, XOR, SHL, SHR, ASR.

Beispiel:



Der Programmfluss des rechten Threads wird durch *Warte auf 1* unterbrochen. Erst mit Erreichen des *Befehls-Elements =1* wird der logische Ausdruck ausgewertet und der rechte Thread weiterverarbeitet.

B. Ergänzendes Beispiel

In diesem Beispiel wird erläutert, wie der Greifer mit Hilfe des Endschalters und Impulsgebers auf der a-Achse verfahren werden kann. Abbildung 22 zeigt die a-Achse in der Seitenansicht. Die Achse ist am oberen Ende mit einem Endschalter ausgestattet, der an I5 angeschlossen ist. Der Impulsgeber befindet sich an der Welle und ist an I6 angeschlossen.

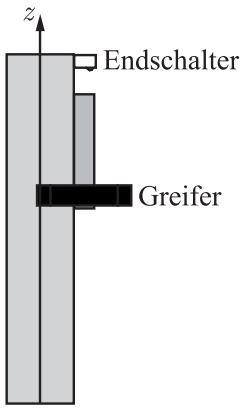


Abbildung 22: a-Achse des Greifroboters

In Abbildung 23 ist eine mögliche Realisierung für die Achssteuerung in ROBOPro dargestellt. Das Programm, das Ihnen als Datei `LRBeispiel.rpp` zur Verfügung gestellt wird, ermöglicht eine Referenzfahrt und das Anfahren von in der Variablen Soll angegebenen Punkten. Die Referenzfahrt wird benötigt, um das inkrementelle Wegmesssystem einzustellen. Hierzu fährt der Greifer bis zum Endschalter, welcher die Position 0 repräsentiert. Durch die Abfrage A=1 (Feld 1 in Abbildung 23) wird bei einem Sollwert von 0 automatisch eine Referenzfahrt durchgeführt. Feld 2 beschreibt den genauen Ablauf der Referenzfahrt. Mit Hilfe des Motorelements bewegt sich der Greifer in positive z-Richtung. Die logische Abfrage Warten auf 1 unterbricht den Programmablauf bis der Endschalter I5 auslöst. Bei Erreichen des Referenzpunktes wird die Variable aist auf 0 gesetzt und der Motor gestoppt. Werden Werte ungleich 0 eingestellt, so erfolgt die Bearbeitung über den Vergleich in Feld 3. Durch die angeschlossenen Variablen wird der Ausdruck A?B überführt in aist?Soll. Gilt aist<Soll, so wird Feld 4 ausgeführt und der Greifer wird nach unten bewegt. Dabei wird jeder Impuls von I6 gezählt und auf die Variable aist addiert. Bei aist>Soll wird der Greifer entsprechend nach oben bewegt und aist dekrementiert. Sobald der Sollwert mit aist=Soll erreicht ist, wird der Motor abgeschaltet.

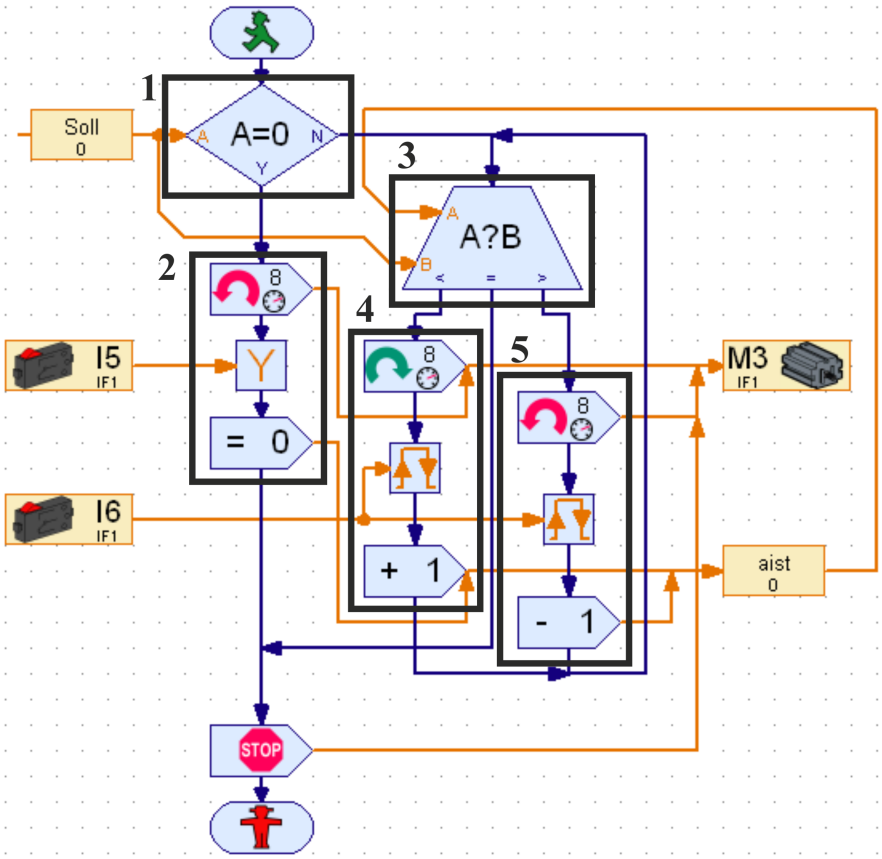


Abbildung 23: Beispielprogramm (`LRBeispiel.rpp`) zum Verfahren der a-Achse