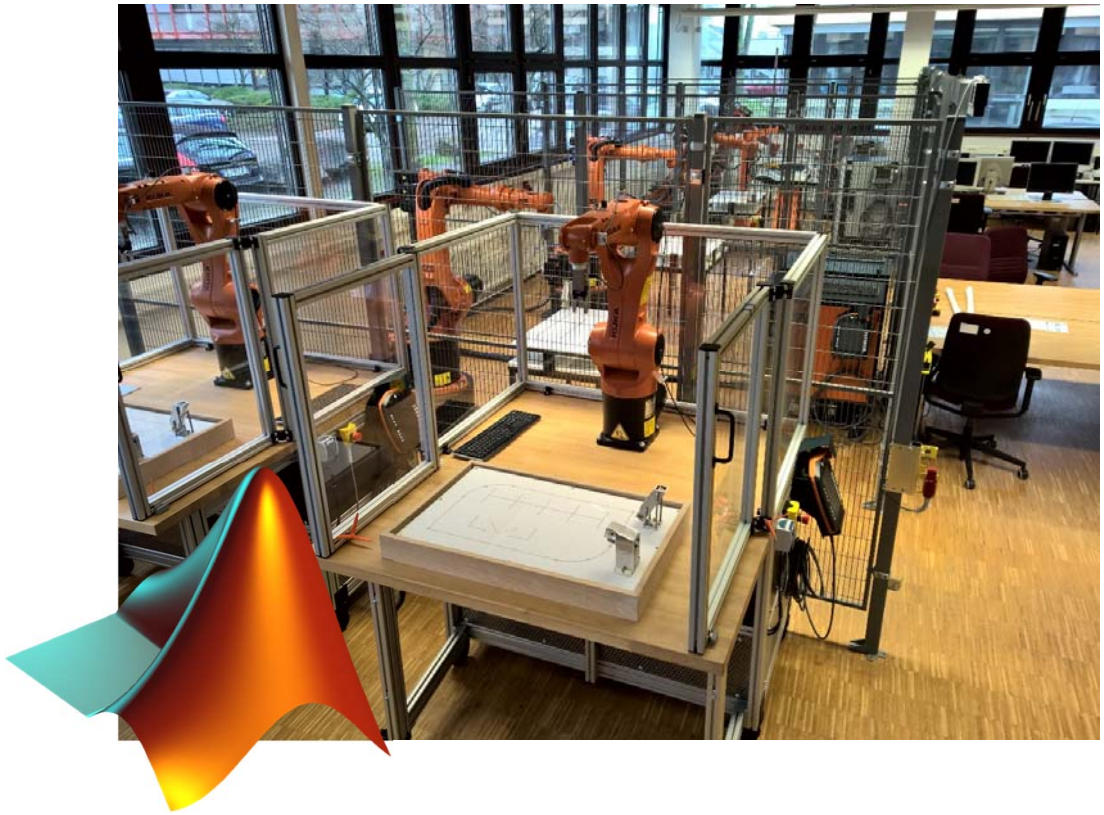
Plattformübergreifende MATLAB/Simulink – Umgebung zur KUKA Roboter Programmierung



Prof. Dr.-Ing. Rolf Biesenbach
Tim Wrütz, M.Sc.
Fachbereich Elektrotechnik & Informatik
Hochschule Bochum
MATLAB EXPO 2019

Hochschule Bochum
Bochum University
of Applied Sciences

Inhalt

- Einleitung
  - Robotersystem
  - Darstellung Industrieroboter
- RoBO-2L
  - V1
  - V2
- Simulink-Umgebung mit mxAutomation
  - mxAutomation
  - Implementierung
  - Einsatzbeispiel
- Fazit

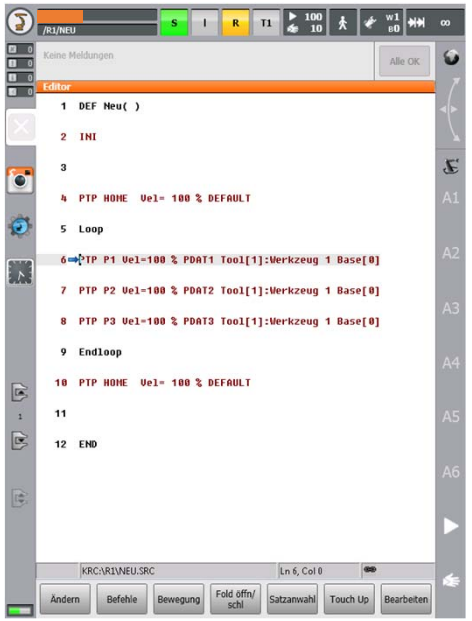
Hochschule Bochum
Bochum University
of Applied Sciences

Robotersystem

- Agilus KR6 R900 (Manipulator)
  - 6 kg Traglast
  - 901 mm Reichweite
  - 0,03 mm Wiederholgenauigkeit
- KRC 4 (Steuerung)
- KRL (Programmiersprache)
  - Bewegungsprogrammierung
  - Logik und Kontrollstrukturen
  - Basisfunktionalität Mathematik

Quelle: KUKA

Quelle: KUKA



Hochschule Bochum
Bochum University
of Applied Sciences

Darstellung Industrieroboter

Hochschule Bochum
Bochum University
of Applied Sciences

Darstellung Industrieroboter

i	α_i	a_i	d_i	θ_i
1	$-90°$	$25\ mm$	$400\ mm$	θ_1
2	$0°$	$455\ mm$	0	$\theta_2 - 90°$
3	$+90°$	$35\ mm$	0	θ_3
4	$-90°$	0	$420\ mm$	θ_4
5	$+90°$	0	0	θ_5
6	$0°$	0	-80	θ_6

Hochschule Bochum
Bochum University
of Applied Sciences

Darstellung Industrieroboter

i	α_i	a_i	d_i	θ_i
1	$-90°$	$25\ mm$	$400\ mm$	θ_1
2	$0°$	$455\ mm$	0	$\theta_2 - 90°$
3	$+90°$	$35\ mm$	0	θ_3
4	$-90°$	0	$420\ mm$	θ_4
5	$+90°$	0	0	θ_5
6	$0°$	0	-80	θ_6

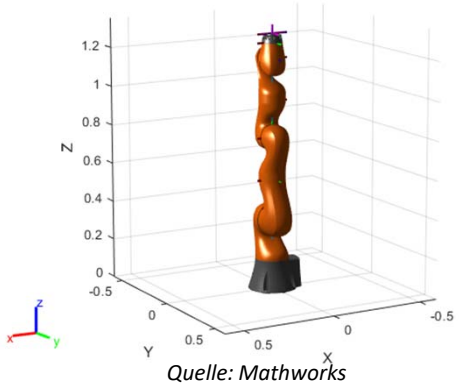
```
[ cos(th_1),  0, -sin(th_1), 25*cos(th_1)]
[ sin(th_1),  0,  cos(th_1), 25*sin(th_1)]
[         0, -1,          0,          400]
[         0,  0,          0,            1]
```

$$ {}^{0}_{6}\boldsymbol{T} = {}^{0}_{1}\boldsymbol{T} \cdot {}^{1}_{2}\boldsymbol{T} \cdot {}^{2}_{3}\boldsymbol{T} \cdot {}^{3}_{4}\boldsymbol{T} \cdot {}^{4}_{5}\boldsymbol{T} \cdot {}^{5}_{6}\boldsymbol{T} $$

Hochschule Bochum
Bochum University
of Applied Sciences

Darstellung Industrieroboter

i	α_i	a_i	d_i	θ_i
1	$-90°$	$25\ mm$	$400\ mm$	θ_1
2	$0°$	$455\ mm$	0	$\theta_2 - 90°$
3	$+90°$	$35\ mm$	0	θ_3
4	$-90°$	0	$420\ mm$	θ_4
5	$+90°$	0	0	θ_5
6	$0°$	0	-80	θ_6



Quelle: Mathworks

```
[ cos(th_1),  0, -sin(th_1), 25*cos(th_1)]
[ sin(th_1),  0,  cos(th_1), 25*sin(th_1)]
[         0, -1,          0,          400]
[         0,  0,          0,            1]
```

$$ {}^{0}_{6}\boldsymbol{T} = {}^{0}_{1}\boldsymbol{T} \cdot {}^{1}_{2}\boldsymbol{T} \cdot {}^{2}_{3}\boldsymbol{T} \cdot {}^{3}_{4}\boldsymbol{T} \cdot {}^{4}_{5}\boldsymbol{T} \cdot {}^{5}_{6}\boldsymbol{T} $$

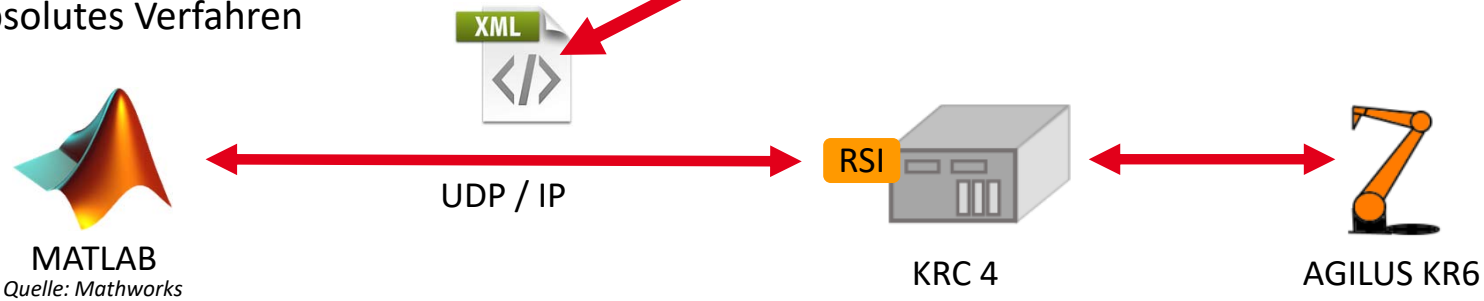
Hochschule Bochum
Bochum University
of Applied Sciences

RoBO-2L V1 *[3]*

- MATLAB Roboterprogrammierung
  - Kein KRL
  - Implementierung komplexer Abläufe
  - Einbindung von MATLAB Toolboxen

- Robot Sensor Interface (RSI)
  - Korrekturbewegungen
  - Geschwindigkeitsvektoren
  - Kein absolutes Verfahren

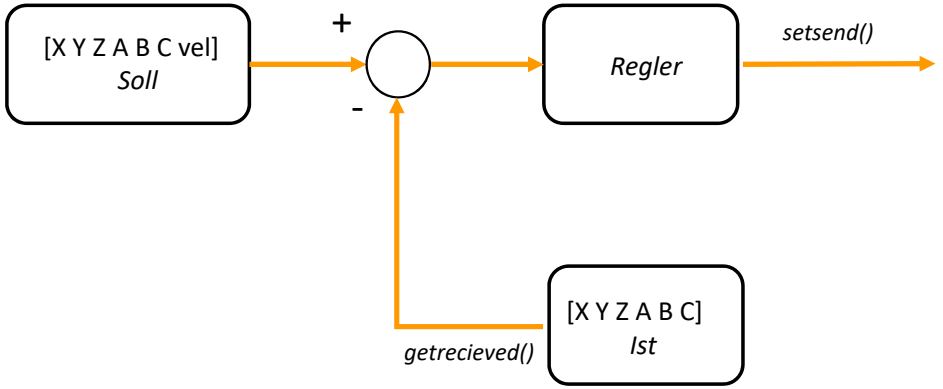
```
1. <Sen Type="ImFree">
2. <EStr></EStr>
3. <Tech T21="1.09" T22="2.08" T23="3.07" T24="4.06" T25="5.05" T26="6.04" T27="7.03" T28="8.02" T29="9.01"
   T210="10.00" />
4. <RKorr X="0" Y="0" Z="0" A="0" B="0" C="0" />
5.  < DiO > 0 </ DiO >
6.  < IPOC >0</IPOC>
```



Hochschule Bochum
Bochum University
of Applied Sciences

Exemplarischer MATLABCODE „LIN“

- Grundkonzept Soll/Ist Vergleich
  - Sortierung nach „Verfahrweg“
  - P-Regler, Verstärkung: 1



- getreceived()
  - Empfängt IST

- setsend()
  - Übermittelt $[\dot{x}\ \dot{y}\ \dot{z}\ \dot{a}\ \dot{b}\ \dot{c}]$

Exemplarischer MATLABCODE „LIN“

- Grundkonzept Soll/Ist Vergleich
  - Sortierung nach „Verfahrweg“
  - P-Regler, Verstärkung: 1

- getreceived()
  - Empfängt IST

- setsend()
  - Übermittelt $[\dot{x}\ \dot{y}\ \dot{z}\ \dot{a}\ \dot{b}\ \dot{c}]$

```
function [done] = lin( varargin )
    if(nargin == 3)
        %Auswertung Punktobjekt + Geschwindigkeit.
    elseif(nargin == 8)
        %Auswertung X,Y,Z,A,B,C,Geschwindigkeit einzeln.
    end

    done = 0;

    while(done == 0)
        XYZABC_IST = getrecieved();
        %Anpassung an KUKA Drehsinn
        if(toA < -90 && isA > 90)
            toA = toA + 360;
        end

        diV = XYZABC_SOLL - XYZABC_IST;
        dV = sort(abs(diV), 'descend');

        direction = vel*diV/(dV(1));
        goal = abs(x)+abs(y)+abs(z)+abs(a)+abs(b)+abs(c);

        if((abs(goal) <= 0.05))
            setsend(0,0,0,0,0,0,G);
            done = 1;
        else
            setsend(direction(1),direction(2),direction(3),direction(4)/2,direction(5)/4,direction(6)/4,G);
        end
    end
end
```

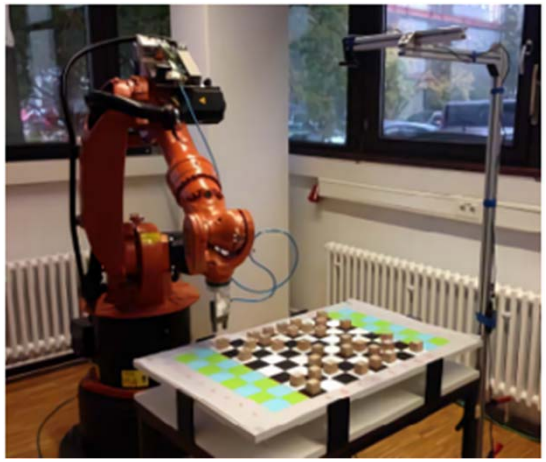
Hochschule Bochum
Bochum University
of Applied Sciences

Roboterschach

- **Computer Vision Toolbox**
  - **Erkennen von SURF Features**
  - **Fusion von Bildern**
  - **Transformation**

- Stockfish Chess Engine *[4]*
  - Open Source Chess KI
  - C++

- RoBO-2L
  - Steuerung IR

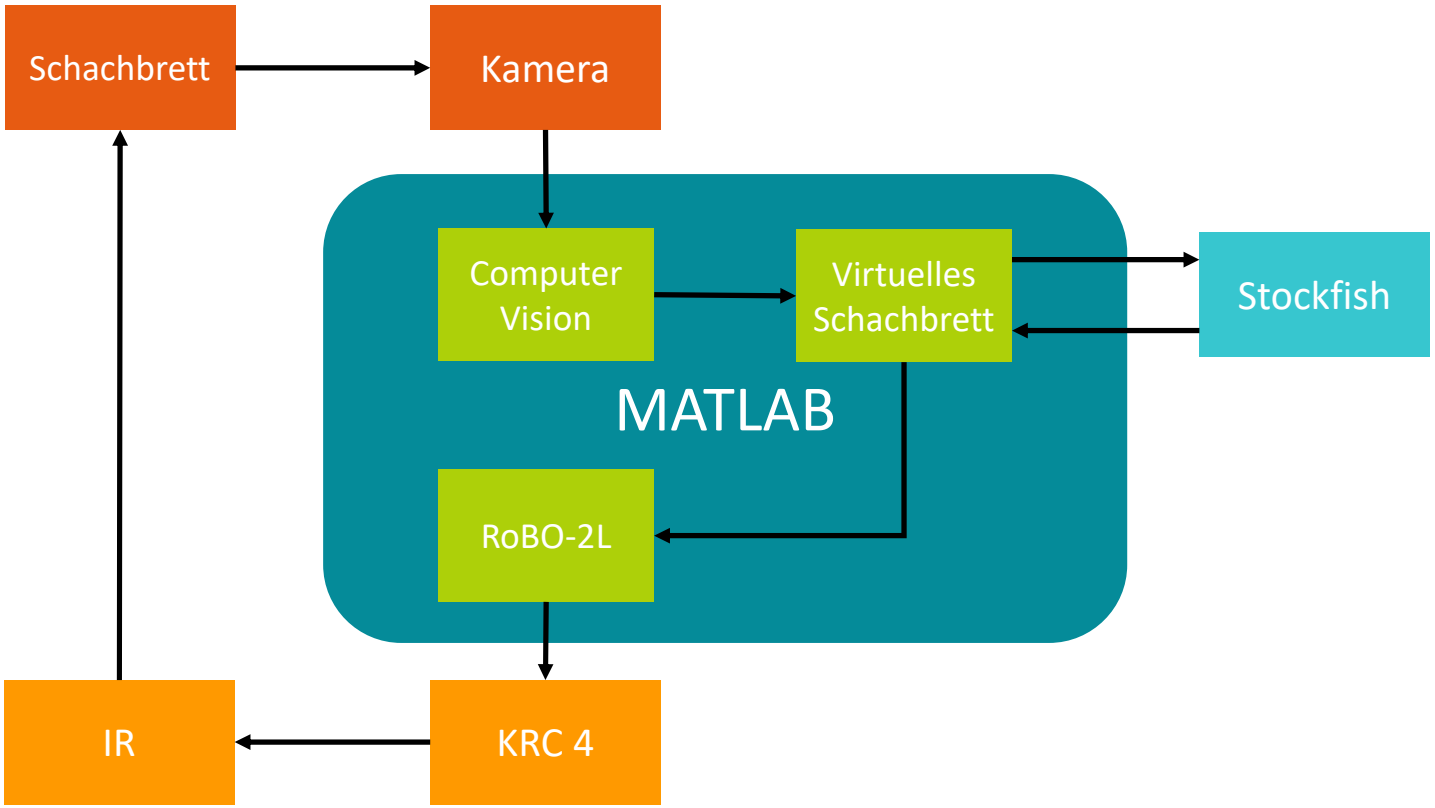


[5]

Hochschule Bochum
Bochum University
of Applied Sciences

Roboterschach

- Computer Vision Toolbox
  - Erkennen von SURF Features
  - Fusion von Bildern
  - Transformation
- Stockfish Chess Engine
  - Open Source Chess KI
  - C++
- RoBO-2L
  - Steuerung IR



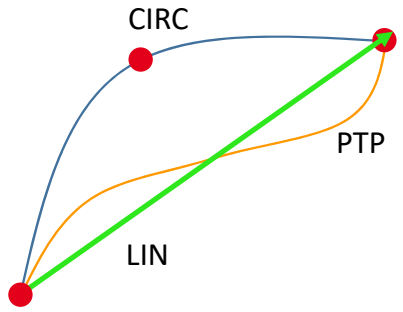
Hochschule Bochum
Bochum University
of Applied Sciences

RoBo-2L V2 [6]

- **V1**: Verfahren mittels $[\dot{x}\ \dot{y}\ \dot{z}\ \dot{a}\ \dot{b}\ \dot{c}]$ funktioniert,
  wichtige PTP Bewegung jedoch nicht ohne weiteres möglich.

- In der Praxis gab es Stabilitätsprobleme der
  Orientierungsparameter $[B\ C]$

- Einfache Implementierung, da keine Kinematik notwendig.

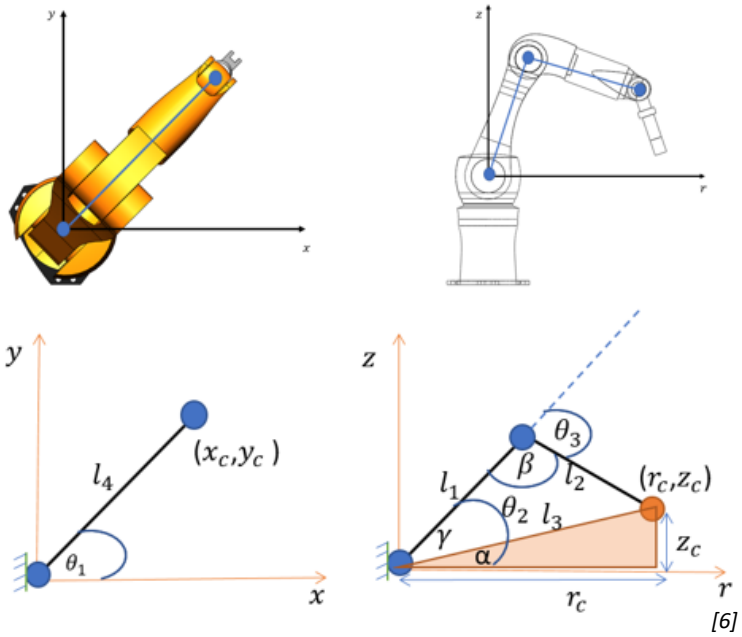


- **Lösung:** Wechsel von $[\dot{x}\ \dot{y}\ \dot{z}\ \dot{a}\ \dot{b}\ \dot{c}]$ auf $[\dot{j}_1\ \dot{j}_2\ \dot{j}_3\ \dot{j}_4\ \dot{j}_5\ \dot{j}_6]$
  - Voraussetzung: Vorwärts- und Inverse Kinematik

Hochschule Bochum
Bochum University
of Applied Sciences

RoBo-2L V2

- Vorwärtstransformation
  - **Matrizen**: DH Parameter
- Inverse Kinematik (Kinematische Entkopplung)
  - Rotation um Y mittels **roty(-90)**
    - Anpassung Werkzeugkkordinaten -> Flansch
  - Geometrische Lösung von$[\theta_1\ \theta_2\ \theta_3]$
  - Bestimmen von$[\theta_4\ \theta_5\ \theta_6]$ über Koeffizientenvergleich mit der Vorwärtstransformation
- Alternativ: numerische Lösung mit **MATLAB**
  - **BFGS** und **Levenberg-Marquardt**

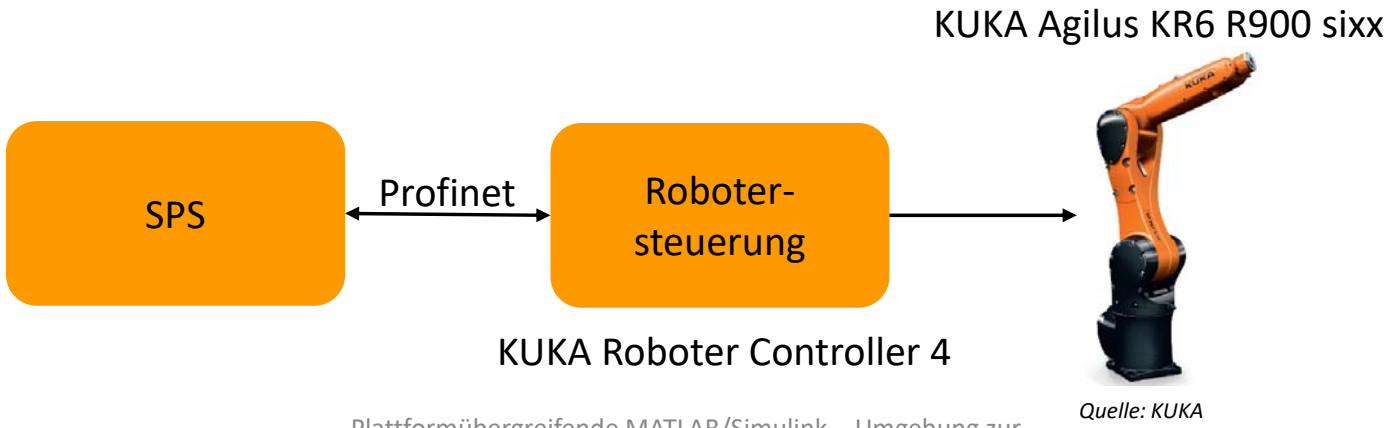


[6]

Hochschule Bochum
Bochum University
of Applied Sciences

mxAutomation

- Softwarepaket von KUKA für SPS basierte Steuerung
- Absolutes Verfahren möglich
- Implementierung eines Interpreters auf der SPS um Bewegungsabläufe zur Laufzeit übertragen zu können.

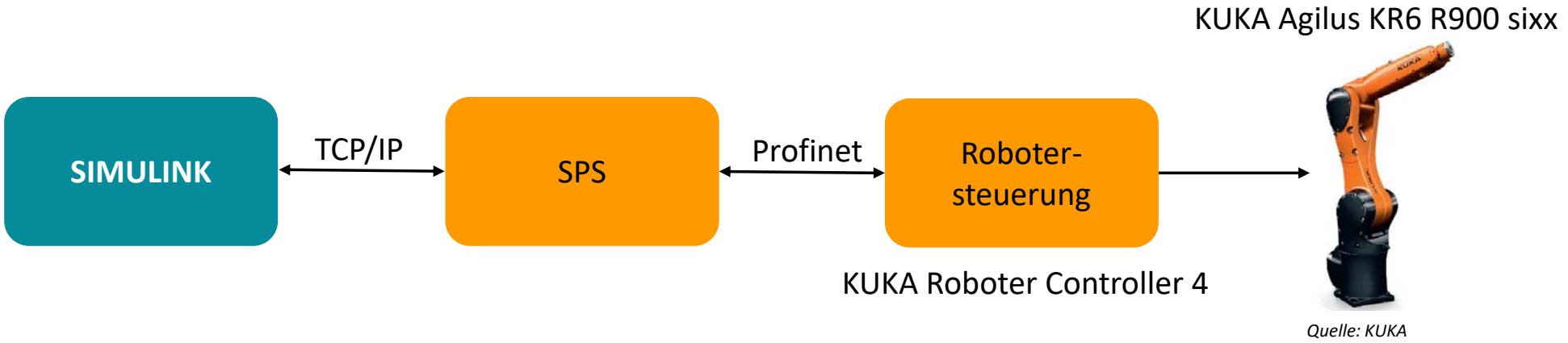


Quelle: KUKA

Simulink-Umgebung mit mxAutomation

Implementierung

- Schnittstelle zwischen mxAutomation und MATLAB/Simulink
- Orientierung am **Embedded Coder**

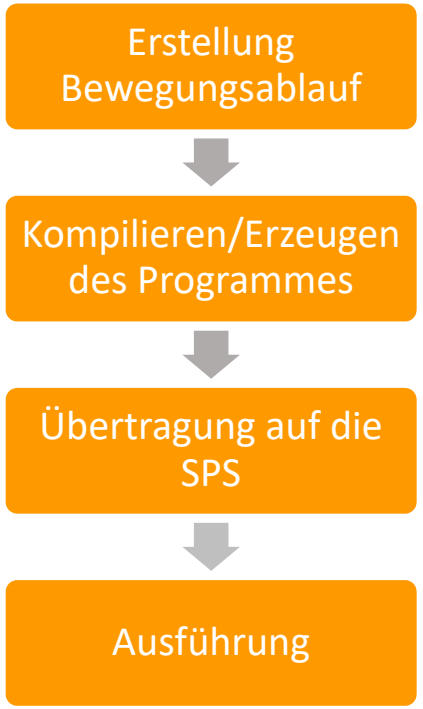


Quelle: KUKA

Simulink-Umgebung mit mxAutomation

Implementierung

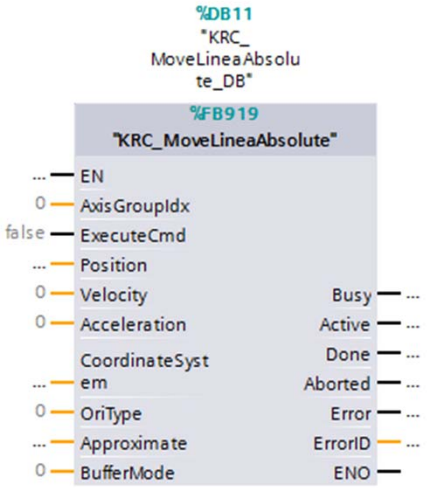
- Schnittstelle zwischen mxAutomation und MATLAB/Simulink
- Orientierung am **Embedded Coder**
- Programmierung zwischen *START* und *ENDE* Baustein
- **Fokus:** Nutzerfreundlichkeit



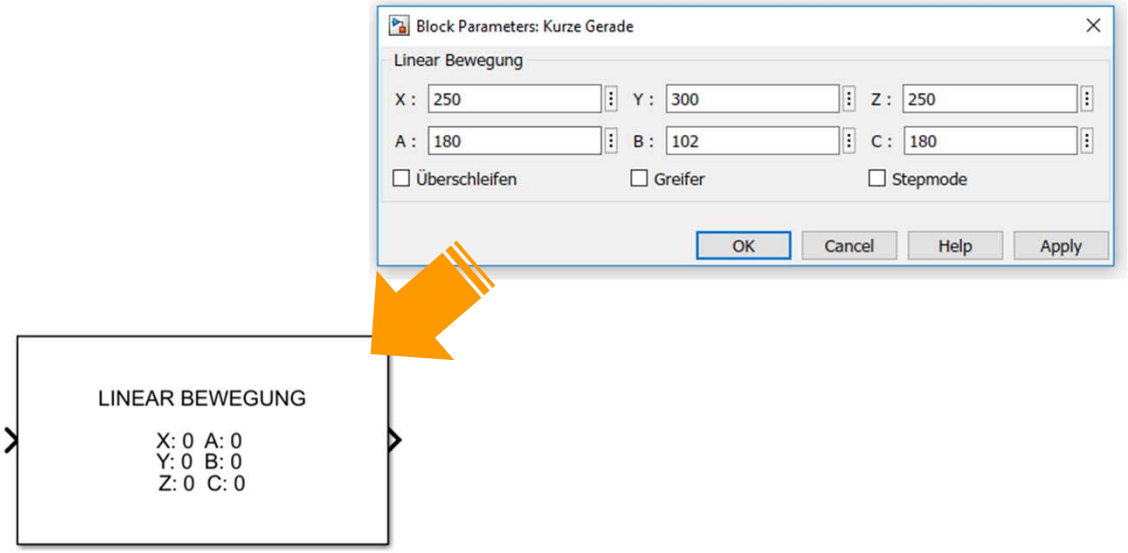
Hochschule Bochum
Bochum University
of Applied Sciences

Implementierung

- **Fokus:** Nutzerfreundlichkeit



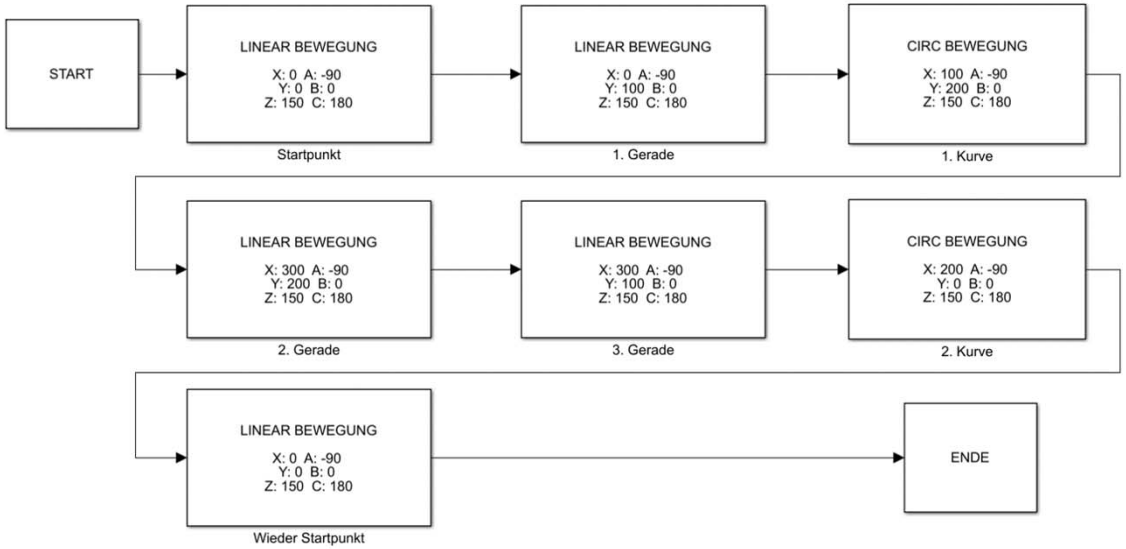
Quelle: Siemens



Hochschule Bochum
Bochum University
of Applied Sciences

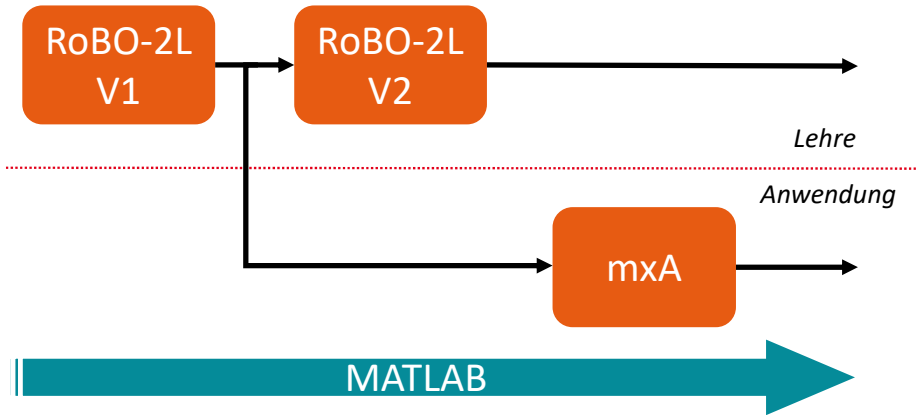
Programmierung, internationale Kooperationen

- Simulink Programmierung aus der Philadelphia University, Jordanien, heraus auf einen Roboter an der Hochschule Bochum.



Fazit

- MATLAB Simulation unterstützt Tests
- MATLAB ermöglicht einfache Integration von Computer Vision
- MATLAB unterstützt maßgebliche Berechnungen
- Typische Bewegungen LIN, CIRC und P2P aus Matlab heraus



- Die noch vergleichsweise junge Robotic System Toolbox, bietet mit ROS Integration und Kinematik Algorithmen Optionen um das Projekt zu erweitern.

Hochschule Bochum
Bochum University
of Applied Sciences

Vielen Dank für Ihre Aufmerksamkeit

Prof. Dr.-Ing. Rolf Biesenbach
Tim Wrütz, M.Sc.
Fachbereich Elektrotechnik & Informatik
Hochschule Bochum
MATLAB EXPO 2019

Hochschule Bochum
Bochum University
of Applied Sciences

Quellen:

[1] KCT: a MATLAB toolbox for motion control of KUKA robot manipulators, F. Chinello, S. Scheggi, F. Morbidi, D. Prattichizzo, University of Siena, IEEE International Conference on Robotics and Automation, May 2010

[2] KUKA-KRL-Toolbox for Matlab® and Scilab, G. Maletzki, M. Christern, A. Schmidt, T. Pawletta, P. Dünow, University of Applied Sciences Wismar, 2015 Online: [http://www.mb.hs-wismar.de/cea/Kuka_KRL_Tbx/KukaKrlTbx.htm13]

[3] Golz J., Biesenbach R., Implementation of an Autonomous Chess Playing Industrial Robot, IEEE 16th International Conference on Research and Education in Mechatronics REM2015, Proceedings, ISBN 978-3-945728-01-7, Bochum, Germany, November 2015

[4] Stockfish – Open – Source Chess Engine, T. Romstad, M. Costalbam and J. Kiisk, Online:[www.stockfishchess.org], 2015

[5] Implementation of an autonomous chess playing industrial robot, with the RoBO-2L Interface. J. Golz, A. Bergmann, R. Biesenbach, University of Applied Sciences Bochum, The 2nd International Conference on Engineering Science and Innovative Technology (ESIT 2016), Thailand, April 2016

[6] Development of a PTP Movement of a Industrial Robot via MATLAB by deriving its Kinematics and Integration in an Offline Programming Tool, K. Enien, T. Wrütz, R. Biesenbach, IEEE 19th International Conference on Research and Education in Mechatronics (REM2018), Delft, Netherlands, June 2018