SKDK
IT-ENGINEERING

Technische Hochschule
Brandenburg
University of
Applied Sciences

# Entwicklung einer Logging-Software zur Überwachung und Analyse der Achsdrehmomente von Industrierobotern

## Bachelorarbeit

zur Erlangung des Grades Bachelor of Science
des Fachbereichs Informatik und Medien der
Technischen Hochschule Brandenburg


**Vorgelegt von:** Jan Philipp Seeland
Matr. Nr. 20213364

**1. Betreuerin:** Prof. Dr.-Ing. Angela Pohl
**2. Betreuer:** Dipl.-Ing. Arnardo Schulze


Brandenburg an der Havel, 16. April 2022

# Kurzfassung

In dieser Arbeit werden Achsdrehmomente und Motorlast eines Industrieroboters der Firma FANUC analysiert, um diese über eine Logging-Software zu optimieren. Das Ziel ist es, über die Aufzeichnung der Motordaten, mechanische Belastungen an den Motoren der Robotergelenke zu reduzieren. Durch diesen Ansatz können Bewegungen des Roboters optimiert und Traglasten maximiert werden. Die zur Zeit angebotene Robotersteuerung des Herstellers bietet nur wenig zeitlich aufgelöste Diagnoseoptionen. Die vorgestellte Lösung unterscheidet sich von anderen Lösungen darin, dass keine Verkabelung von zusätzlicher Hardware notwendig ist. Auch muss keine zusätzliche, kostenpflichtige Software installiert werden. Der in dieser Arbeit verfolgte Ansatz implementiert eine webbasierte Logging-Software für einen Client-PC und ein KAREL Programm für einen FANUC Roboter, welches als Schnittstelle dient.
Die Parameter Drehmoment(*Disturbance Torque, Torque Monitor*) und Motorstrom (*OVC Monitor*) konnten periodisch alle 50ms von der Robotersteuerung aufgezeichnet und anschließend zeitlich aufgelöst dargestellt werden. Es wurden Experimente mit verschiedenen Geschwindigkeiten, Achsstellungen und Traglasten durchgeführt. Ein signifikantes Ergebnis konnte bei der Reduzierung der Motorlast unter Verwendung von Beschleunigungsrampen festgestellt werden. Auch wurde der negative Einfluss einer inkorrekt eingestellten Traglast bei der Ansteuerung der Motoren nachvollzogen und aufgezeichnet. In einem Kundenprojekt konnte die Software erfolgreich zur Machbarkeitsanalyse eingesetzt werden.

# Inhaltsverzeichnis

# Abbildungsverzeichnis



# Tabellenverzeichnis



# Listingverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| **API** | Application Programming Interface |
| **CSS** | Cascading Style Sheets |
| **CPU** | Central Processing Unit |
| **CAD** | Computer Aided Design |
| **DOF** | Degree of Freedom |
| **DH** | Denavit-Hartenberg |
| **DOM** | Document Object Model |
| **FTP** | File Transfer Protocol |
| **FIFO** | First In - First Out |
| **HTML** | Hyper Text Markup Language |
| **HTTP** | Hyper Text Transfer Protocol |
| **IP** | Internetprotokoll |
| **JS** | JavaScript |
| **JSON** | JavaScript Object Notation |
| **PTP** | Point to Point |
| **SVG** | Scalable Vector Graphics |
| **SM** | Socket Messaging |
| **SPS** | Speicherprogrammierbare Steuerung |
| **TP** | Teach Pendant |
| **TPP** | Teach Pendant Programm |
| **TCP** | Tool Center Point |
| **URL** | Uniform Resource Locator |

# 1 Einleitung

Im Bereich der Automatisierung wird der Einsatz von Industrierobotern zur Handhabung von Werkstücken, Schweißen, Palettieren, Verpacken, Montieren und Prüfen, um nur einige Bereiche zu nennen, immer relevanter. Damit sollen Mitarbeiter von Routineaufgaben befreit werden und bessere Ergebnisse bei ihren hochspezialisierten Tätigkeiten erzielen [Lie18]. Wie in Abb. 1 zu sehen ist, existiert eine große Produktpalette, um möglichst viele Anwendungsbereiche abdecken zu können. Vom kleinen „Pick-and-Place“ Roboter bis hin zu großen Robotern, welche ganze Autokarossen bewegen können.

Abb. 1: Ausschnitt der verschiedenen FANUC Industrieroboter [1]

Der Bedarf an Robotern ist groß und aus der Industrie nicht mehr wegzudenken [IFR21]. Aber auch für kleine und mittelständige Unternehmen kann sich ihr Einsatz lohnen. Durch intuitive Programmieroberflächen und kollaborative Robotersysteme die keinen Schutzzaum brauchen, kommen immer mehr dieser Systeme zum Einsatz. Jedoch ist die korrekte Dimensionierung der Anlage entscheidend. Ein wichtiger Parameter ist hierbei die maximale Traglast, um keine der Achsen zu überlasten [Lie18]. Dadurch wird ein erhöhter Verschleiß an Motoren und Getriebe vermieden und Wartungsintervalle verlängert. Jedoch ist die Berechnung der Last nicht trivial. Durch den Hebeleffekt, welcher sich über die Roboterachsen aufsummiert und die Fliehkräfte, welche während der Bewegung wirken, kann nicht ausschließlich vom Gewicht der Nutzlast ausgegangen werden. Bei den betrachteten Systemen wird derzeit eine statische Berechnungsformel zur Errechnung der maximalen Traglast angewendet. Problematisch an diesem Ansatz ist, dass weder Achsstellung noch Geschwindigkeit berücksichtigt werden. Dadurch kann nur ein kleiner Teilbereich der Realität abgebildet werden.

[1] Bildquelle: https://www.fanuc.eu/de/de/roboter, letzter Zugriff: 06.04.2022

Das Ziel der Arbeit ist die Programmierung einer Logging-Software und deren experimentelle Evaluation an einem 6-Achsigen Roboter. Anhand der Daten sollen Bewegungen optimiert und Belastungen minimiert werden.

Die Arbeit stellt zunächst einige Grundlagen der Robotik vor. Die Roboterpositionierung und -kinematik bilden dabei die Basis zum Verständnis der Funktionsweise von Industrierobotern.Im Kapitel 3 werden verwandte Arbeiten vorgestellt, welche ähnliche Problemstellungen untersuchen. Das Konzept stellt den Ansatz der Datenaufzeichnung an den 6 Motoren der Robotergelenke vor, welcher in dieser Arbeit implementiert und durchgeführt wird. Dabei wird auf Anforderungen, Datenerhebung und Datenvisualisierung eingegangen. Die entwickelte Logging-Software und dessen Programmierung wird im Kapitel Implementierung detailliert vorgestellt. Dabei wird Aufschluss über die verwendeten Bibliotheken für die browserbasierte Client-Software, die Kommunikation mit der Robotersteuerung und benötigte Schnittstellen gegeben. Zum Schluss der Arbeit werden die Ergebnisse vorgestellt, diskutiert und ein abschließendes Fazit mit Ausblick auf Weiterentwicklungen gegeben.

# 2 Grundlagen Industrieroboter

In diesem Kapitel werden relevante Grundlagen und Begrifflichkeiten der Robotik beschrieben, welche zum Verständnis der Problemstellung beitragen. Es wird näher auf den Roboteraufbau, dessen Positionierung im Raum und die Kinematik von Industrierobotern eingegangen. Beispiele werden anhand eines Roboters der Firma FANUC beschrieben.

Grundsätzlich lässt sich definieren: “Industrieroboter sind universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegungen hinsichtlich Bewegungsabfolge und Wegen bzw. Winkeln frei (d.h. ohne mechanischen bzw. menschlichen Eingriff) programmierbar und gegebenenfalls sensorgeführt sind. Sie sind mit Greifern, Werkzeugen oder anderen Fertigungsmitteln ausrüstbar und können Handhabungs- und/oder Fertigungsaufgaben ausführen“‘[Pro90]. Speziell wird im Rahmen dieser Arbeit mit einem 6-Achs-Roboter, auch Knickarmroboter genannt, gearbeitet. Das bedeutet, dass der Roboter sechs unabhängig voneinander angetriebene Rotationsachsen hat, welche zu einer eindeutigen Bewegung führen. Jedes Robotergelenk verleiht dem Roboter einen sog. Freiheitsgrad F Degree of Freedom (DOF) [Wü18, S. 19 f.]. Der 6-achsige Roboter mit Freiheitsgrad F=6, kann in seinem Arbeitsraum jeden Punkt mit dem Roboterarm in beliebiger Orientierung erreichen, soweit keine Einschränkungen in den Achswinkeln bestehen. Das macht den 6-achsigen Roboter universell geeignet für diverse Automatisierungsaufgaben. Er ist der Standardtyp im Industriebereich [Wü18, S. 21].

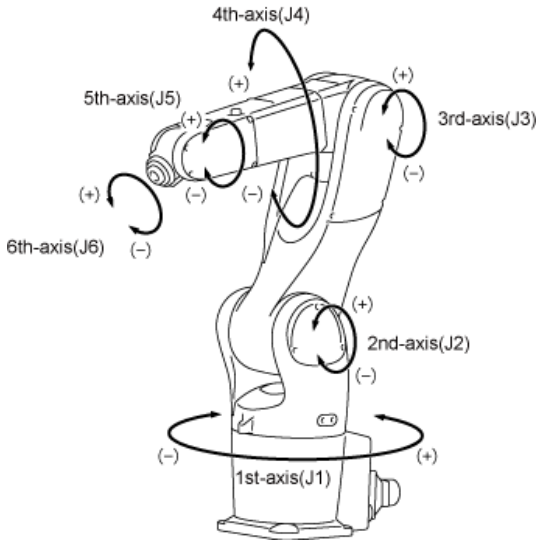


Abb. 2: Schematische Darstellung eines Knickarmroboters mit Beschriftung der Gelenkpositionen [2]

[2]Bildquelle: http://www.olympusftp.com/PROD/User%20Manuals/en/001529.html, letzter Zugriff: 06.04.2022

## 2.1 Roboterpositionierung

Der Endeffektor, auch Effektor oder Tool genannt, beschreibt das Werkzeug am Ende der Kinematischen Kette (Ende des Roboterarms) und wird vom Roboter zum Werkstück geführt [Wü18, S. 19]. Das Werkzeug kann, je nach Einsatzbereich des Roboters, frei gewählt werden. Es wird am Roboterflansch angebracht, an welchem sich zusätzlich Gewinde zum Verschrauben des Werkzeugs befinden. Unter dem Tool Center Point (TCP) versteht man den Werkzeugmittelpunkt. Dieser ist für die Positionierung des Roboters im Raum wichtig und bestimmt den genauen Arbeitspunkt des Werkzeugs, wo der Roboter mit dem Werkstück in Berührung kommt. Das Anlernen des TCPs wird über eine Spitze durchgeführt, an die das Werkzeug mehrmals aus verschiedenen Richtungen vorsichtig herangefahren wird [Wü18, S. 48].

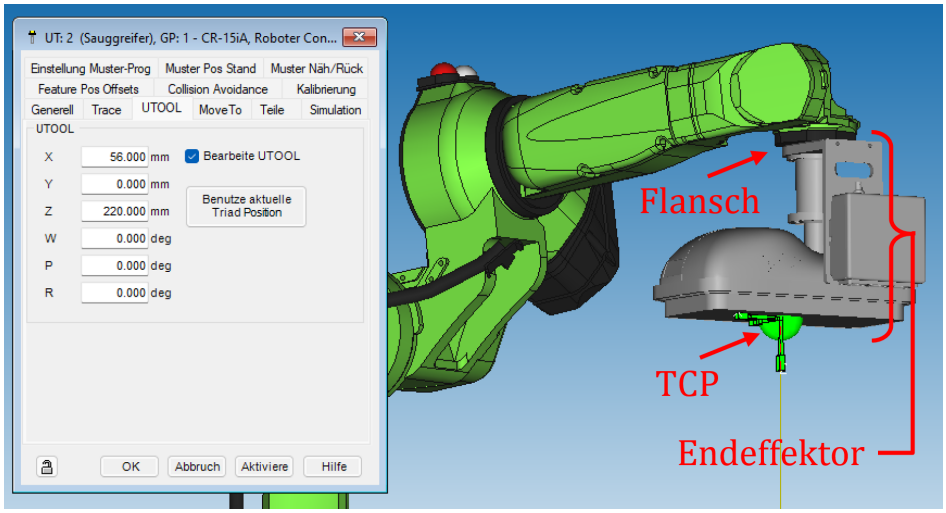


Abb. 3: Aufbau des Endeffektors am Roboter, Vermessung des Werkzeugmittelpunkts in FANUC's Simulatorsoftware ROBOGUIDE

In Fällen, wo keine Messvorrichtung zur Verfügung steht oder das Anfahren aus verschiedenen Richtungen schwierig ist, lässt sich dies genauer und einfacher per Software machen. Die Roboterhersteller bieten daher Simulationsumgebungen an, in denen sich Computer Aided Design (CAD) Dateien importieren lassen. So kann der Greifer dem Robotermodell hinzugefügt werden und der Arbeitspunkt exakt über die Abmessungen des Modells bestimmt werden. Der Simulator kommt auch bei der in Kapitel 2.4 erklärten Offline-Programmierung zum Einsatz.

Nach der Vermessung des Werkzeugs ist der Arbeitspunkt bestimmt, an dem das Werkstück manipuliert wird. Da der Roboter weder das Gewicht des Werkzeugs noch des Werkstücks kennt, muss das am Roboterarm lastende Gewicht, auch *Payload* genannt, korrekt vom Programmierer über das Handprogrammiergerät gesetzt werden. Der Roboter berechnet daraus, welche Kräfte an den einzelnen Achsen wirken und regelt daraufhin die Motorströme und Schwellwerte für die Kollisionserkennung.

Der Programmierer kann den Roboter in verschiedenen Roboter-Koordinatensystemen verfahren lassen und Bewegungspfade in diesen speichern. Sie bestimmen, wie sich der TCP im Raum positionieren lässt.

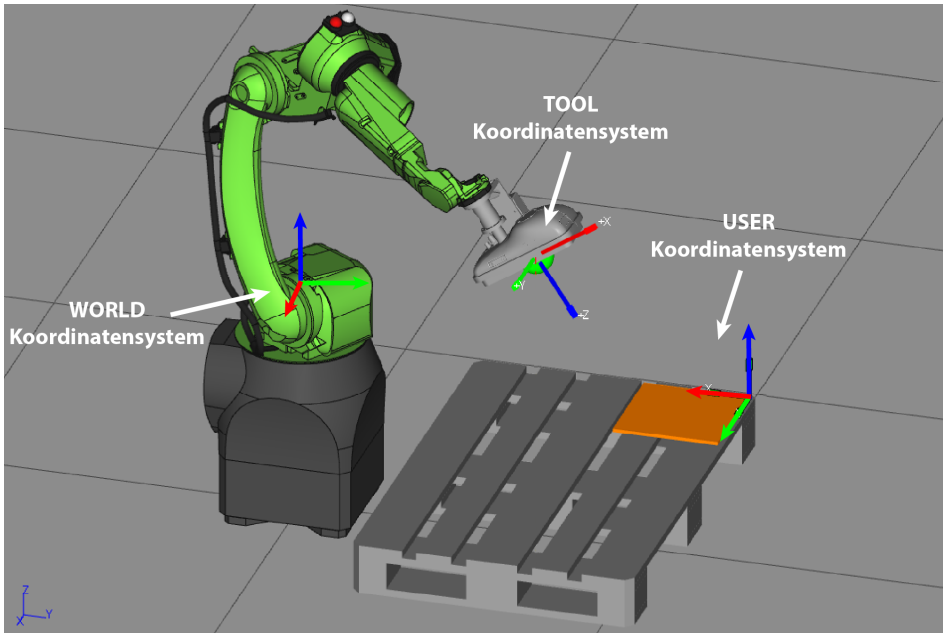


Abb. 4: Roboter Koordinatensysteme WORLD, TOOL und USER

Der Ursprung des WORLD Koordinatensystems liegt immer im Fuß des Roboters und hat die Koordinaten (0,0,0). Dort liegt auch das ROBROOT Koordinatensystem. Diese sollten deckungsgleich zueinander sein. Der Ursprung des TOOL Koordinatensystems legt der TCP fest. Standardmäßig liegt der Ursprung im Flansch des Roboters und wird von dort aus, je nach Tool transformiert. Das USER Koordinatensystem kann verwendet werden, um die Position des Werkstücks zu beschreiben. Der Nullpunkt ist eine Transformation ausgehend vom Ursprung des WORLD Koordinatensystems.

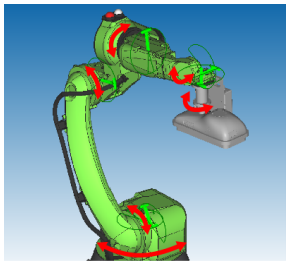

Abb. 5: JOINT Koordinatensystem

Das JOINT oder Gelenkkoordinatensystem erlaubt das Positionieren des Roboters durch Drehung einzelner Achsen in positiver/negativer Richtung.

## 2.2 Bewegungsarten

Wenn ein Koordinatensystem festgelegt ist, kann zwischen zwei Bewegungsarten gewählt werden. Der Point to Point (PTP) Steuerung und der Bahnsteuerung.
Bei der PTP Steuerung wird ein Anfangs- und ein Endpunkt definiert. Alle Achsen synchronisieren ihre Geschwindigkeit mit der langsamsten Achse, sodass alle Achsen ihre Bewegung gleichzeitig beenden im Endpunkt. Dies nennt man auch Synchrone PTP-Steuerung. Die Bewegungsbahn wird nicht vorgegeben. Das Bedeutet, dass die Bahn auf der sich der Roboter bewegt, nicht eindeutig ist [Tea]. In den meisten Fällen resultiert aus der Bewegung der Achse eine bogenartige Bewegung. Wenn Bewegungen jedoch exakt programmiert werden müssen, also der Bahnverlauf und die Geschwindigkeit von Bedeutung sind (z.B. Beim Auftragen einer Klebemasse entlang eines Pfades), kommt die Bahnsteuerung zum Einsatz. Bei dieser Bewegungsart müssen die Motoren mit hoher Präzision gesteuert werden. Dies erfordert von der Robotersteuerung viel mehr Rechenleistung, da mehrere Bahnstützpunkte berechnet werden, um Kreis-, Linear- oder Polynombahnen abfahren zu können [Tea].

## 2.3 Roboterkinematik

Die Roboterkinematik beschreibt die Zusammenhänge zwischen dem Raum in Roboterkoordinaten (Gelenkkoordinaten) und dem Raum in Weltkoordinaten (Kartesischer Raum), ausgehend von der Lage des Endeffektors [Asf20]. Das kinematische Modell des Roboterarms wird idealisiert als sog. Offene Kinematische Kette beschrieben. Die Achsen werden als starre Körper betrachtet und die Gelenke als dessen Verbindungen [Bar98, S. 287]. Da wir Menschen in kartesischen Koordinaten (XYZ) denken und arbeiten, sollte der Endeffektor auch in dieser Form positionierbar sein. Die Schwierigkeit besteht darin, die Zusammenhänge zwischen den Gelenkwinkeln und den Koordinaten im Raum zu bestimmen, um den Roboterarm in den Koordinatensystemen aus Kapitel 2.1 positionieren zu können.

Um die Position und Orientierung des TCPs beschreiben zu können, muss eine mathematische Beziehung zwischen Basiskoordinatensystem und allen Koordinatensystemen der Zwischensegmente bestehen. Das bedeutet, dass bei n-achsigen Robotern die Beziehungen von $n + 1$ Koordinatensystemen (Basiskoordinatensystem + Anzahl der Achsen) beschrieben werden müssen [Bar98, S. 289]. Diese Beziehungen werden mittels Denavit-Hartenberg (DH)-Konventionen errechnet. Diese stellen eine besonders effiziente Art dar, die Beziehungen zwischen zwei benachbarten Gelenken in Translation und Rotation zu beschreiben [Asf20].

- Roboterposition in Gelenkkoordinaten:

  J1: 110,710 ° J2: -7,008 ° J3: 7,529 °
  J4: 0,290 ° J5: -96,766 ° J6: -110,695 °

- Roboterposition umgerechnet in kartesische Koordinaten:

  X: -179,290 mm Y: 626.519 mm Z: 625.286 mm
  W: -179.184 ° P: 0.000 ° R: -0.020 °

Diese Umrechnung wird als *kinematische Transformation* bezeichnet. Bei der Umrechnung von Gelenkkoordinaten in kartesische Koordinaten spricht man von *kinematischer Vorwärtstransformation*, bei der Umrechnung von kartesischen Koordinaten in Gelenkkoordinaten spricht man von *kinematischer Rückwärtstransformation* oder *inverser kinematischer Transformation* [Asf20].

## 2.4 Roboterprogrammierung

Um den Roboter in die gewünschten Positionen zu bewegen, muss der Robotersteuerung ein Verhalten angelernt werden. Dieses nennt man programmieren. Grundlegend lässt sich zwischen Online- und Offline-Programmierung unterscheiden.
Bei der Online-Programmierung wird der Roboter im eingeschalteten Zustand über das Handprogrammiergerät oder auch Teach Pendant (TP), mit stark reduzierter Geschwindigkeit verfahren. Mit dem TP können alle Bewegungsfunktionen des Roboters direkt gesteuert werden. Eine schematische Abbildung des Handprogrammiergeräts ist in Bild 7 zu sehen. Der Programmierer führt nun mit dem TP die gewünschte Bewegungsabfolge aus. Diese wird dann in Form von Punkten mit Bewegungsart und Geschwindigkeit im Roboterprogramm gespeichert [Wü18, S. 40]. Ist die vom Roboter auszuführende Bewegungsabfolge komplex oder sensorgesteuert, empfiehlt sich die Offline-Programmierung. Hier können (wie in Kapitel 2.1 erwähnt) CAD Dateien benutzt werden, um in Simulationsprogrammen die Roboterzelle nachzubilden. Es können Sicherheits- und Sperrzonen angelegt werden oder ein grafischer Editor zum Schreiben der Roboterprogramme benutzt werden.

Neben der Programmierung der Bewegungsabfolge über das TP, lässt sich der Roboter über eine Herstellerspezifische Sprache für komplexere Programme programmieren. Bei Robotern der Firma FANUC kommt die proprietäre Programmiersprache *KAREL Command Language* (kurz *KAREL*) zum Einsatz. *KAREL* ist eine Mischung aus Hochsprachen wie Pascal und PL/1 und Funktionen für Roboteranwendungen. Der Compiler über setzt den *KAREL* Code in für den FANUC Roboter lesbaren proprietären *p-code* [Ltdc, S. 40].

## 2.5 Roboterkonnektivität

Mit steigendem Automatisierungsgrad erhöht sich auch die Anzahl der Geräte, welche untereinander verdrahtet werden müssen. Um diesem Aufwand in der industriellen Kommunikation entgegen zu wirken, wurden die Geräte parallel mit der Regel- und Steuertechnik verdrahtet und nicht untereinander. Daher entstanden diverse Feldbus Systeme, welche in einem Master-Slave-Betrieb mit den Feldgeräten, z.B. Motoren, Schalter, Sensoren oder Lampen kommunizieren [Kun22]. Auch die zeitliche Abstimmung, wann welches Programm z.B. auf einem Roboter ausgeführt werden soll und wann Feldgeräte wegen einer Störung gestoppt werden müssen wird überwacht und geregelt. Diese Aufgabe übernehmen sog. Speicherprogrammierbare Steuerungen (SPS). In einer Fabrik, wo viele Prozesse voneinander abhängig sind, ist diese enge Vernetzung wichtig, um mehrere Automatisierungsschritte miteinander verknüpfen zu können.

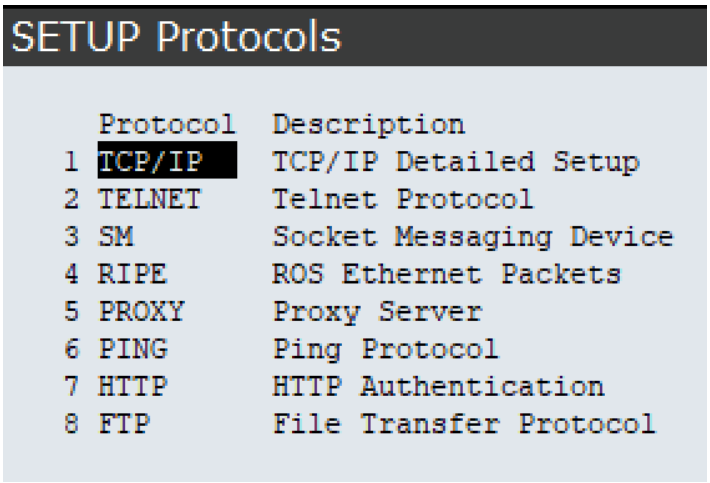


Abb. 6: Netzwerkprotokolle der Robotersteuerung

Die Robotersteuerung stellt für die Kommunikation verschiedenste Schnittstellen zur Verfügung, wie in Abbildung 6 zu sehen. Für die Programmierung des Roboters sind folgende Protokolle nützlich: Hyper Text Transfer Protocol (HTTP), um z.B. den Roboterstatus oder das TP über einen Browser aufzurufen oder File Transfer Protocol (FTP), um Dateien einfach auf den Roboter zu übertragen. Aber auch einfache digitale und analoge Signale können von der Steuerung verarbeitet werden. Um unabhängig vom Hersteller mit verschiedensten Geräten kommunizieren zu können, stehen viele Schnittstellen zur Verfügung.

# 3 Verwandte Arbeiten

Es besteht ein allgemeines Interesse an der Überwachung der mechanischen Einheiten von Industrierobotern, um den Übergang von der reaktiven zur zustandsorientierten Instandhaltung zu schaffen.
Folgende Arbeiten werden dazu vorgestellt:

| **Features** | **Anwendung** |
|---|---|
| FANUC AIServoMonitor [Ltd21a] | Auslesen und Überwachen der Motordaten über das Netzwerk, zur präventiven Wartung, nicht-invasiv, externer Server notwendig, für CNC Steuerungen konzipiert |
| Sabry, A.H. et al. 2020 [SNSAAK20] | Fehlerdetektion über Stromverbrauchsmessung des Roboters, zur präventiven Wartung, invasiv, proprietärer Messaufbau, Logging Datenrate 123 |
| Paes, K. et al. 2014 [PDE+14] | Messung der Motorströme zur Energieoptimierung der Pfade, zur Optimierung, nicht-invasiv, spezielle Messhardware, Logging Datenrate 3,2GHz |

Tabelle 1: Übersicht der diskutierten Arbeiten

## 3.1 FANUC AIServoMonitor

In [Ltd21a] wird der Aufbau und die Funktion einer Software zur Datenvisualisierung der Firma FANUC für CNC Maschinen beschrieben. Es handelt sich um eine webbasierte Software mit dem Namen *AIServoMonitor*. Diese kommt in Großproduktionen zum Einsatz, um Ausfälle von FANUC CNC Motoren frühzeitig zu erkennen. Der Anwender wird über Störungen benachrichtigt, welche als Graph visualisiert werden. Ein Referenzdatensatz mit Drehmomentdaten wird mithilfe von mehreren Ausführungszyklen computergesteuert erstellt. Bei Störung der Motoren werden die Abweichungen der Drehmomentdaten von den Referenzwerten erkannt [Ltd21a]. Zur Überwachung der mechanischen Komponenten wird keine zusätzliche Hardware an der Maschinensteuerung benötigt. Softwareseitig wird ein *FANUC OPC-Server*[3] in Verbindung mit der Überwachungssoftware *FANUC MT-LINKi Operation Management Software* zur Datensammlung benötigt, welche auf einem externen Gerät installiert werden muss. Große Firmen nutzen diese zur Überwachung von mehreren hundert Maschinen im Firmennetzwerk. Die Daten lassen sich in vollem Umfang zur zustandsorientierten Instandhaltung von Servo/Spindel Motoren auf FANUC's CNC Steuerungen auslesen.

[3]Kommunikationsstandard für industrielle Automatisierung. FANUC OPC-Server ist eine Windows Anwendungssoftware, welche das Kommunikationsprotokoll zwischen OPC und FOCAS konvertiert. Geräte mit OPC-Client-Funktion können über diese Software mit FANUC CNC-Anlagen kommunizieren [Ltda].

Für Roboter Controller sind nur die Signale zur Programmsteuerung und -überwachung auslesbar [Ltd21b].

## 3.2 Strommessung zur Motorfehlerdetektion

In der Publikation [SNSAAK20] stellen die Autoren eine *Fehlerdetektierung und Diagnose von Industrierobotern über Modellierung des Stromverbrauchs* dar.
Die Forschung untersucht den direkten Einfluss, den ein mechanische Fehler auf die benötigte gesamt Energie des Roboters hat. An einem 6-achsigen Industrieroboter wird unter der Verwendung von proprietärer Hardware, bestehend aus Funkmodulen und Stromsensoren der Gesamtstromverbrauch des Manipulators[4] gemessen. Der Stromverbrauch in Watt und die Gelenkposition in Grad werden zeitlich aufgelöst in einem Graphen dargestellt. Durch den eigens entwickelten Messaufbau konnte eine Datenrate von 123 Datenpunkten pro Sekunde erreicht werden. Die Erkennung von Abweichungen bei der Stromaufnahme geschieht über ein Neuronales Netz, welches initial auf fehlerfreien Daten trainiert wurde. Dadurch konnten erfolgreich Einzel- und Mehrfachfehler des Manipulators erkannt werden.

## 3.3 Energieoptimierung durch effiziente Pfade

Die Publikation [PDE+14], mit dem Titel *Energie effiziente Pfade für einen ABB Industrieroboter* verfolgt einen nicht-invasiven Ansatz zur Aufzeichnung der Motorströme, um den Energieverbrauch der Motoren zu optimieren. Dabei wurden Stromklemmen jeweils an die Stromleitungen der Motoren gelegt und mit einem Chauvin Arnoux 8335 Netz-Analysegerät gemessen und aufgezeichnet. Das Messgerät hat eine feste Datenrate von 3200Hz und kann maximal 80 Sekunden lang die Ströme aufzeichnen, bevor der Speicher voll ist. Dazu wurde synchronisiert die Gelenkposition mit 100Hz über die Robotersteuerung ausgelesen. Die Werte wurden mit einem in MatLab erstellten, parametrierten, dynamischen Robotermodell mit einer Optimierungsroutine für die ausgeführten Pfade verglichen. Die Simulationsergebnisse kommen auf eine Zeit und Energieersparnis von 5% gegenüber dem Bewegungsplaner des Roboterherstellers. Außerdem wurden Messungen im Roboterstillstand durchgeführt, wobei das Anziehen der Bremsen im Vergleich zum benötigten Haltestrom im Stillstand, die größte Energieersparnis darstellte.

[4] Beschreibt in der Robotik ein Gerät, welches die physikalische Interaktion mit der Umgebung ermöglicht, in diesem Fall die mechanische Einheit des Roboters. (Quelle: https://de.wikipedia.org/wiki/Manipulator, letzter Zugriff: 13.04.2022)

# 4 Konzept

Die Bachelorarbeit wird am Automatisierungsunternehmen SKDK IT-Engineering GmbH geschrieben. Das Unternehmen hat seinen Sitz in Berlin Spandau und konzipiert und programmiert Automatisierungslösungen nach Kundenwunsch. Im Bereich der Roboterprogrammierung verfügt das Unternehmen über langjährige Erfahrungen im Umgang mit Robotersteuerungen der Firma FANUC.
Ziel der Arbeit ist die Erstellung und Evaluation einer Logging-Software. Den Anwendern sollen Werte über die Belastung des Roboters zeitlich aufgelöst dargestellt werden können. Zur Evaluation der Software werden Experimente durchgeführt, um den Einfluss der Parameter auf die jeweiligen Achsen quantifizieren zu können. Dazu gehört die Analyse der Achsdrehmomente und des Motorstroms.

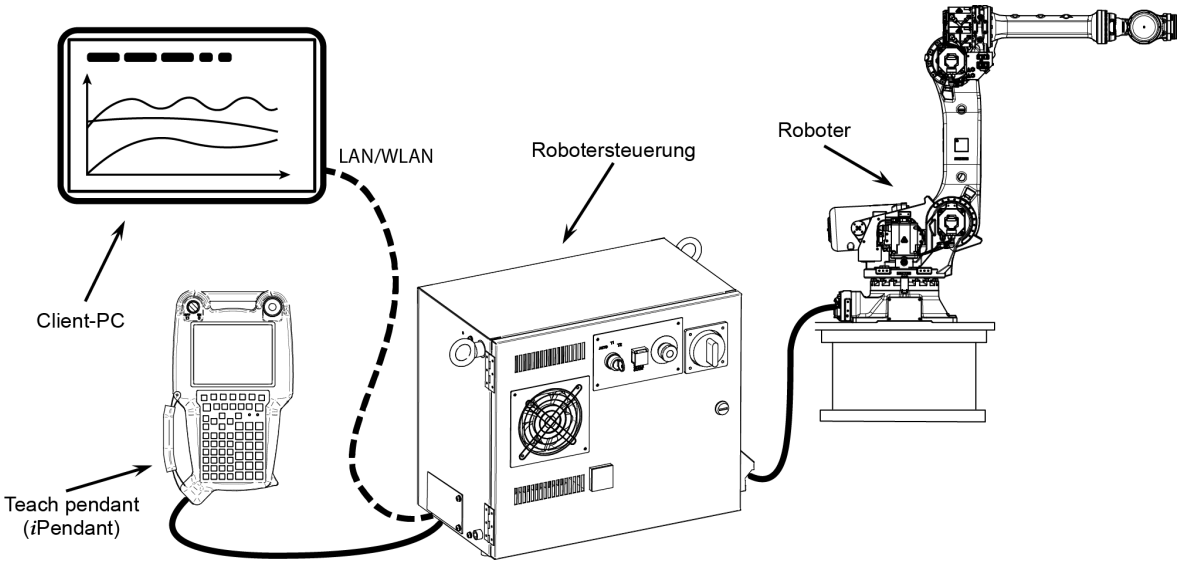


Abb. 7: Schematische Darstellung des Robotersystems, modifiziert nach [Ltdd]

Die Software soll aus zwei Teilen bestehen. Einer Schnittstelle auf dem Roboter und einer Web-Anwendung zur Visualisierung der Daten auf einem Client-PC. Durch die begrenzten Hardwareressourcen in der Robotersteuerung muss die Kommunikation effizient erfolgen und die Daten müssen auf dem externen Gerät gespeichert werden. Die Benutzeroberfläche soll übersichtlich aufgebaut und intuitiv bedienbar sein. Die resultierenden Daten sollen Auskunft über die Belastung der Achsen geben, um den Anwender Aufschlüsse über den Belastungszustand des Roboters zu geben. So sollen die Daten zur Optimierung der Pfade, Bestimmung von maximal möglichen Traglasten oder Entlastung der Mechanik beitragen können.

# 5 Anforderungen

Im folgenden Abschnitt wird näher auf die funktionale und nicht-funktionalen Anforderungen der einzelnen Komponenten eingegangen. Es wird evaluiert, welche Funktionen zielführend sind und wo Kompromisse eingegangen werden müssen.

## 5.1 Roboterhardware

Um den Einsatz der Software für den Benutzer zu erleichtern, soll so wenig wie möglich am Roboter geändert werden müssen. Im Gegensatz zu anderen Lösungen (vgl. [SNSAAK20]), soll daher kein zusätzliches Hardware Modul angeschlossen werden müssen. Damit fällt auch keine zusätzliche Verkabelung an. Die Kompatibilität soll auf FANUC Robotersteuerungen der Serie R-30 ausführbar sein.

Da in dieser Arbeit mit einem kommerziellen System gearbeitet wird, sind nicht alle Informationen öffentlich zugänglich. Daher kann keine exakte Aussage über die Leistung der Robotersteuerungen getroffen werden und ob diese für den Anwendungszweck leistungsfähig genug sind. Außerdem variiert die Hardwareausstattung je nach Steuerung. Im Wartungshandbuch zur Robotersteuerung R-30iB wird ein gesonderter Kontrollschaltkreis für das Benutzer Bedienfeld (operator's panel) beschrieben. Die Berechnung der prozesskritischen Motorbewegungen werden von der Haupt Central Processing Unit (CPU) gesteuert [Ltdd, S. 10]. Trotzdem wird vor Leistungseinbußen an der Netzwerkschnittstelle bei zu vielen zyklischen Dateiübertragungen gewarnt und empfohlen, bei Leistungseinbrüchen das Intervall vor der nächsten Übertragung zu vergrößern, sodass Dateianfragen nicht parallel, sondern sequentiell an die Steuerung geschickt werden [Ltdb, S. 361].

## 5.2 Roboter Schnittstelle

Für die Verbindung mit der Logging-Software sollen vorhandene Schnittstellen des Roboters genutzt werden. Es wird davon ausgegangen, dass die Netzwerkschnittstelle des Roboters eingerichtet und benutzbar ist. Auch sollen keine speziellen Kommunikationsprotokolle, welche für einen Aufpreis herstellerseitig nachgerüstet werden können, eingesetzt werden. Die damit verbundenen Kosten würden den Einsatzbereich der Software sonst erheblich einschränken. Mit Verwendung der Standardfunktionen soll die Schnittstelle unabhängig von der Softwareausstattung auf allen Steuerungen ausführbar sein. Eine weitere Einschränkung die beachtet werden muss, ist dass keine Register, in denen Daten zur Bewegungsabfolge gespeichert werden, belegt werden dürfen.

## 5.3 Logging-Software

Um nicht auf bestimmte Endgeräte beschränkt zu sein, soll die Software möglichst plattformunabhängig funktionieren. Hier ist eine browserbasierte Implementierung naheliegend. Die Mindestanforderung sollte ein sog. “moderner“ Browser sein welcher auf Hyper Text Markup Language (HTML) Version 5, Cascading Style Sheets (CSS) Version 3 und JavaScript (JS) Version ES6 basiert (Firefox, Chrome, Safari, IE9 aufwärts). Da die Daten über die Netzwerkschnittstelle des Roboters übertragen werden sollen, ist eine gewisse Latenz Aufgrund der Netzwerkprotokolle nicht vermeidbar. Ziel sollte es jedoch sein, eine Abtastrate von mindestens 20hz (20 Datenpunkte pro Sekunde) zu erreichen. Eine gröbere Auflösung ist nicht zielführend, da ein zu hoher Abstand zwischen den einzelnen Datenpunkten zu einer Verfälschung der Daten führt. Leistungsspitzen, welche nur für einen Bruchteil einer Sekunde auftreten, müssen messbar bleiben. Die Datenaufzeichnung darf jedoch nicht die Robotersteuerung überlasten und dessen Reaktionsgeschwindigkeit beeinträchtigen. Die Daten sollen nach der Aufzeichnung direkt in der Software gespeichert werden, um Datenverlust zu vermeiden. Dadurch können die Analysen auch in komplexeren Tabellenkalkulationsprogrammen, wie z.B. Excel überführbar sein. Die Funktion, die Aufzeichnungen in einem universellen Dateiformat exportieren zu können, soll gegeben sein.

## 5.4 Datenvisualisierung

Die Visualisierung der Daten ist das Kernelement der Software. Die von der Schnittstelle ausgelesenen Rohdaten müssen interpretiert und zeitlich aufgelöst dargestellt werden. Die einzelnen Achsen sollen kombiniert mit unterschiedlichen Farben in einem Liniendiagramm gezeichnet werden. Einzelne Achsen ausgeblendet werden können, sodass der Benutzer wählen kann welche Achsen dargestellt werden und welche ausgeblendet werden sollen. Im Gegensatz zu der Visualisierung der Echtzeit Daten soll bei der Darstellung der Aufzeichnungen in Ausschnitte herein gezoomt werden können. Der Anwender soll auch durch Bewegen der Maus über die Datenreihen die aktuellen Werte angezeigt bekommen. Zudem sollen für die Auswertung interessante Punkte, wie z.B. Extremwerte, angezeigt werden. Das Endgerät, welches die Daten entgegen nimmt, muss genug Leistung haben um die in 5.3 beschriebenen Softwareanforderungen verzögerungsfrei ausführen zu können.

# 6 Implementierung

In diesem Kapitel wird die Implementierung der Logging-Software auf Client und Roboterseite beschrieben. Dazu gehört das Auslesen und Verarbeiten der Roboterdaten, die Einbindung von Bibliotheken zur Visualisierung und Erklärung von Funktionen.

## 6.1 Kommunikation mit der Robotersteuerung

Grundsätzlich müssen 3 unterschiedliche Teile des Systems betrachtet werden:
Die Robotersteuerung, das Handprogrammiergerät TP und der Roboter (siehe 7).
Die Robotersteuerung ist die zentrale Einheit des Systems. Auf ihr werden alle Berechnungen zur Routenplanung ausgeführt und alle Schnittstellen für den Bediener und für die Netzwerkteilnehmer bereitgestellt. So auch der Webserver, welcher interaktive HTML Seiten anzeigt, mit denen der Roboter gesteuert werden kann. Das (TP) verbindet sich ebenfalls über eine IP Adresse mit dem Roboterserver und zeigt die Menüstruktur an und sendet Benutzereingaben, z.B. welche Knöpfe gedrückt wurden, an die Steuerung zurück. Das TP fungiert lediglich als ein Thin-Client [5], welcher alle Daten (Roboteroptionen, Menü, Programmieroberfläche,...) über Web-Anfragen von der Steuerung gesendet bekommt.
Ein Großteil der Daten, auf die der Roboter angewiesen ist, finden sich in den Systemvariablen. Diese sind für den Anwender zugänglich und bieten diverse Einstellungen und Systemwerte. Manche Variablen haben nur einen Wert oder eine Zeichenkette gespeichert, andere verbergen ganze Datenstrukturen. In solchen Strukturen werden meist mehrere Variablen, welche zu einem größeren Softwaremodul des Roboters gehören, zusammengefasst.

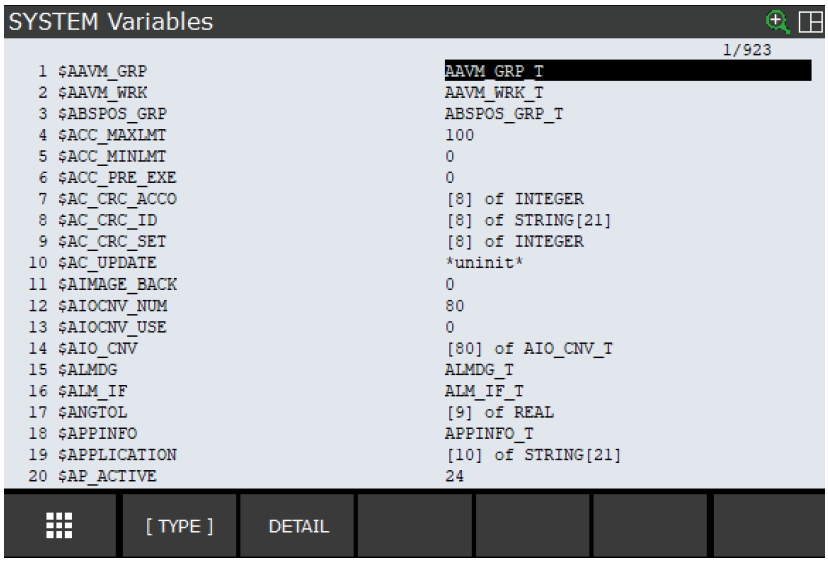


Abb. 8: TP Bildschirm der Systemvariablen, eigene Aufnahme aus Roboter Simulatorumgebung

[5] Leistungsschwacher PC, welcher mit einem Server verbunden ist und dessen Ressourcen nutzt [al.22]

Auf diese Daten kann auch programmatisch zugegriffen werden. Der Zugriff erfolgt dabei mit dem Punktoperator nach folgendem Schema:

$MOR_GRP[1].$CUR_TORQUE[1]

Hierbei ist *MOR_GRP* der Name der Struktur und *CUR_TORQUE* der Name des Arrays mit den momentanen Drehmomentwerten des Roboters.

Ein Konzept zur Ansteuerung des Roboters bestand darin, die Socket Messaging (SM) Funktionalität des Roboters zu benutzen. Diese Netzwerkschnittstelle baut eine verbindungsorientierte Kommunikation, aufsetzend auf dem Internetprotokoll Transmission Control Protocol (kurz TCP) auf. Der Socket ist dabei der Verbindungsendpunkt, welcher vom Programm wie eine gewöhnliche Datei beschrieben und gelesen werden kann. Durch diesen Kommunikationskanal können Daten effizient gesendet und empfangen werden [PP17].
Bei dem Versuch, die SM Funktionalität zu implementieren, musste jedoch festgestellt werden, dass die Option zwar im Menu jedes Roboters aufgeführt wird, die freie Verfügung über die Funktion jedoch eine kostenpflichtige Softwareoption ist. So konnte nach einiger Recherche in Erfahrung gebracht werden, dass die Software Option *R636 (Socket Messaging)* lediglich Daten für spezielle Anzeigen auf dem TP bereitstellt. Nur die Option *R648 (User Socket Msg)* hätte die benötigten Optionen zur Nutzung von SM für den Programmierer freigegeben. Da die Anforderung, keine zusätzlichen Softwaremodule einzusetzen dadurch nicht erfüllt ist, wurde sich gegen die Implementation der Socket Messaging Funktionalität entschieden. Zur Abhilfe wurde die in Kapitel 6.1.3 behandelte PIPE-Ausgabe zur Datenübertragung genutzt.

### 6.1.1 Roboter Web-API Reverse-Engineering

Eine Web-Application Programming Interface (API), im deutschen auch Programmierschnittstelle genannt, ist grundsätzlich ein Satz an Befehlen und Funktionen, mit denen der Programmierer mit Software oder einem externen System interagieren können. [6]
Da die API des Webservers auf der Robotersteuerung nur für die von FANUC programmierten Werkzeuge konzipiert ist, liegt keine offizielle Dokumentation vor. Jedoch hat die API Zugriff auf alle Daten der Steuerung, womit die Nutzung dieser Schnittstelle nicht im Vorhinein ausgeschlossen werden sollte. Durch die fehlende Dokumentation mussten Serveranfragen mitgeschnitten und deren Funktionsweise rekonstruiert werden.

Viele Funktionen werden auf Seiten des Clients ausgeführt, um Hardware Ressourcen auf der Robotersteuerung einzusparen. Dies ermöglicht, sich über einen Webbrowser

[6] https://www.talend.com/de/resources/was-ist-eine-api/

mit der Steuerung zu verbinden, die Oberfläche des TP aufzurufen und über die im Browser integrierten Entwicklertools übertragene Dateien und Web-Anfragen einzusehen. Unter dem Reiter *Netzwerk* und Request-Typ *fetch* lässt sich die Anfrage-URL einsehen. Beispielhaft ist in Abbildung 9 eine solche Uniform Resource Locator (URL) kommentiert dargestellt:

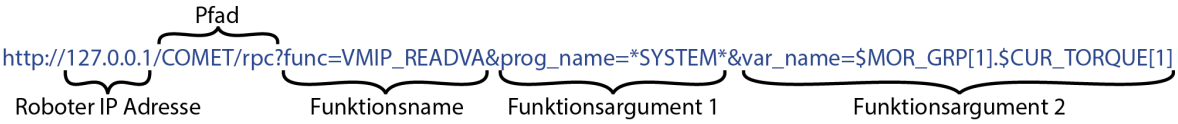


Abb. 9: Beispiel einer Anfrage-URL für einen *Remote Procedure Call*

Auf diese Weise war es möglich, die Struktur der Anfragen nachzubauen und über den eigenen JavaScript Code, Anfragen an die Robotersteuerung zu schicken.

```
function getSystemVar(sysVarName) {
  fetch(url + 'COMET/rpc?func=VMIP_READVA&prog_name=*SYSTEM*&
        var_name=' + sysVarName, {
    cache: 'no-store',
  })
  .then(res => res.json())
  .then(data => {
      /* do something with the data */
  }).catch(err => alert('Failed to Fetch SystemVar: ' + err))
}
```

Listing 1: Beispiel eines Web Requests über JavaScript an den Roboterserver

### 6.1.2 Grenzen der Web-API

Der Flaschenhals der Web-API besteht darin, dass nur einzelne Variablen ausgelesen werden können. Der aktuelle Belastungswert muss für alle 6 Achsen möglichst zeitgleich ausgelesen werden und jeder Achswert steht in einer eigenen Variable. Der “fetch“ Befehl in JavaScript läuft asynchron. D.h. Der Client, auf welchem die Logging-Software ausgeführt wird, schickt alle 50ms sechs Web-Requests in einem sehr kurzen Zeitfenster los. Da diese Anfragen nicht alle schnell genug vom Roboter Web-Server verarbeitet werden können, kommt es an dieser Stelle zu Leistungseinbußen. Die Steuerung reagiert dabei sehr träge oder friert komplett ein. Wie in Kapitel 5.1 beschrieben, sollen Prozesse, die die Reaktionszeit des Systems einschränken vermieden werden. Mit einem Zeitintervall, welches deutlich über 100ms liegt und mit einer Totzeit vor abschicken des nächsten Web-Requests, können Daten übertragen werden, aber die angestrebte Datenrate von 20Hz kann nicht erreicht werden. Daher kann dieser spezielle Web-Request nicht zur Aufzeichnung der Roboterdaten benutzt werden. Jedoch lässt sich

der Befehl dennoch für Daten verwenden, die einmalig oder weniger frequentiert von der Steuerung abgerufen werden müssen.

### 6.1.3 PIPE Ausgabe über KAREL

Die Robotersteuerung unterstützt zusätzlich das Beschreiben von *File Pipes*. Eine Pipe wird zum Puffern oder dem temporären Speichern von Daten benutzt, meistens um Daten zu einer anderen Anwendung zu übertragen. Sie arbeitet nach dem First In - First Out (FIFO) Prinzip. D.h. Daten "fließen" in das Datenrohr hinein und "fallen" an dessen Ende wieder heraus. Dieses Konzept wird auch unter UNIX/Linux und Windows benutzt [Ltdc, S. 200]. Auf der Robotersteuerung ist die Pipe als virtueller Datenträger auf dem schnellen, aber volatilen Systemspeicher unter dem Verzeichnis *PIP:* ansprechbar. Der Speicherzugriff ist dadurch sehr effizient. Die Größe der Pipe ist begrenzt und liegt standardmäßig bei 8 Kilobytes. Des weiteren ist die Pipe als Ringpuffer implementiert. Daher dürfen die Daten nicht die eingestellte Pipe Größe überschreiten, da sonst bei einem Überlauf der vordere Speicherbereich überschrieben wird [Ltdc, S. 200]. Für den Pufferinhalt werden jedoch maximal 56 Byte benötigt, weswegen die Begrenzung nicht erreicht wird.

Die Dateien werden mithilfe einer *KAREL* Routine an die Pipe übertragen. KAREL ist die eigens von FANUC entwickelte Hochsprache, wie im Kapitel 2.4 bereits beschrieben. Die Routine wird im Hintergrund ausgeführt und wird parallel zu einem (Teach Pendant Programm (TPP)) gestartet. Da die *KAREL* Routine lediglich eine Schnittstelle zur Pipe ist und keine Bewegungen ausführen soll, muss das Flag *%NOLOCKGROUP* gesetzt sein. Damit wird sichergestellt, dass dem Programm keine Bewegungsgruppe zugeordnet ist keine Roboterachsen blockiert sind. Nach Initialisierung der benötigten Variablen wird alle 50ms eine WHILE-Schleife durchlaufen. Damit kann die angestrebte Datenrate von 20Hz erreicht werden.

Über die KAREL Built-in Routine *GET_VAR* werden die Systemvariablen zur Überwachung der einzelnen Achsen ausgelesen. Beispielhaft ist in Listing 2 die Syntax zum Auslesen des momentan anliegenden Störmoment (engl. Disturbance Torque) zu sehen. Der an den Robotermotoren anliegende Störmoment überwacht, ob eine Abnormalität der Position durch das am Endeffektor festgehaltene Werkstück vorliegt. Wenn der gemessene Wert der Abnormalitäts-Beurteilungsvorrichtung den Grenzwert überschreitet, stoppt die Steuerung die Bewegungsausführung [Fur].

```
1 GET_VAR(1,'*SYSTEM*','$MISC[1].$HPD_TRQ[1]',tj1,status)
```

Listing 2: Funktionsaufruf zum Auslesen einer Systemvariablen

Zum Speichern der Werte in einer *LogDatei*, wird über die Built-in Routine *PIPE_CONFIG* die Datei *torque.dat* erstellt. Über einen Schreibbefehl werden die ausgelesenen Variablen in einem Komma separierten Format in die Datei geschrieben. Zur eindeutigen Identifikation enthält jeder Datensatz eine ID (*sampleCounter*). Diese Datei wird von der Client Software im Browser interpretiert. Da die Übertragungszeit über das Netzwerk variiert und der verwendete JS Timer eine hohe Ungenauigkeit hat (vgl. Kapitel 6.2.2), kommt zusätzlich ein Ringpuffer zum Einsatz, sodass die *LogDatei* immer 2 Datensätze hält. Dadurch geht, wenn ein Web-Request verspätet losgeschickt wird, kein Datensatz verloren.

```
1  46, -0.60423, 0.411975, 1.18099, 0.535567, -0.137325, -1.73029; 47,
     -0.65423, 0.4105, 2.11099, 1.9674, -0.573321, 1.4307;
```

Listing 3: Beispiel eines Datensatzes, welcher über die Pipe übertragen wird

Am Ende der Routine wird noch der *sampleCounter* erhöht, damit überprüft werden kann, ob kein Datensatz verpasst wurde. Nach einer Verzögerung von 50 Millisekunden beginnt der nächste Schleifendurchlauf. Dieser Intervall kann angepasst werden, sollte aber nicht zu klein gewählt werden, um Leistungseinbußen der Roboter Software zu verhindern.

## 6.2 Client Software

Die Client Software stellt die Bedienoberfläche für den Benutzer. Sie wird als Webseite über einen Browser aufgerufen und bedarf keiner weiteren externen Software. In diesem Abschnitt wird vorgestellt, aus welchen Komponenten die Software besteht und welche Besonderheiten diese aufweist. Mithilfe von Code Ausschnitten werden interessante Stellen erläutert. Grundlegend lässt sich die Software in drei Teilbereiche gliedern. Dem Frontend, welches das Layout, die Bedienelemente und dessen Darstellung enthält. Der Datenzugriffsschicht, welches die Verbindung mit der Robotersteuerung herstellt und die Belastungsdaten verarbeitet und der Datenvisualisierung, welche die vorbereiteten Daten in Form von 2 Graphen visualisiert.


Abb. 10: Ansicht einer aufgezeichneten Belastungskurve in der Client Software

### 6.2.1 Frontend

Alle Bedienelemente befinden sich in der Hauptdatei *LiveLog.html*. Die HTML Datei inkludiert, die für die Datenhaltung und Datenvisualisierung benötigten Dateien. Außerdem wird ein auf Bootstrap basierendes CSS verwendet. Bootstrap ist ein frei verfügbares Framework (z. Dt. Rahmenstruktur), zur Erstellung von responsiven Webseiten. Zusätzlich werden Designbeispiele, sog. *Themes* zur Verfügung gestellt [Ott22]. Durch die Verwendung eines Bootstrap Themes wird dem Programmierer viel Arbeit abgenommen, eine visuell ansprechende Webseite zu erstellen.
Unter Berücksichtigung der an die Software gestellten Anforderungen gilt es folgende Elemente auf der Webseite anzuordnen:

- Schaltfläche zum Verbinden des Roboters über die KAREL Schnittstelle
- Schaltfläche zum Starten/Stoppen einer Aufzeichnung
- Schaltfläche zur Auswahl und Anzeige der Aufzeichnungen
- Schaltfläche zum Ein-/Ausblenden der einzelnen Roboterachsen
- Graph zur Echtzeitvisualisierung der Roboterdaten
- Graph zur Anzeige der Aufzeichnungen
- Anzeige von Extrem- bzw. Grenzwerten

Der wichtigste Aspekt war, eine sinnvolle Aufteilung für die Graphen und dessen Bedienelemente zu finden. Daher wurde die Website in zwei Bereiche eingeteilt. Der Linke Teil hält die verschiedenen Visualisierungen. Diese sind in jeweils eigenen Schaltflächen

zwischen denen gewechselt werden kann (Tabs) untergebracht. Der Vorteil ist, dass immer nur eine Visualisierung aktiv ist. So lässt sich die Ansicht schnell ändern und in Zukunft um weitere Visualisierungen einfach erweitern. Die Bedienelemente, welche den rechten Teil der Website einnehmen, verändern jeweils die aktive Visualisierung. Diese globale Schaltfläche dient zum Ein- und Ausblenden individueller Achsen. Sobald eine Visualisierung angezeigt wird, synchronisieren die Schaltflächen ihre Farbe zu denen der angezeigten Graphen. Damit kann der Benutzer, die Kurven eindeutig den Roboterachsen zuordnen.

Der erste Tab mit dem Namen *LiveLog* blendet die Schaltflächen zum Starten der Verbindung über die IP-Adresse [7] des Roboters und zum Starten einer Aufzeichnung ein. Darunter findet sich der dazugehörige Graph. Der zweite Tab *SaveLog* zeigt lokal gespeicherte Aufzeichnungen (Logs) mit dessen jeweiligem Aufzeichnungsdatum an. Zwei kleine Schaltflächen neben jedem Log ermöglichen das Löschen oder Herunterladen der Datei im *.csv* Format. Nach Auswahl einer Aufzeichnung wird diese unter den Schaltflächen angezeigt.

### 6.2.2 Datenzugriffsschicht

Im Hintergrund gibt die Datei *dataHandler.js* den Schaltflächen ihre Funktion. Um das Projekt schlank und übersichtlich zu halten, wurde nur mit reinem JS programmiert. Dadurch ist die Datenhaltung und -verarbeitung von keinen Bibliotheken abhängig, was Wartung, Kompatibilität und zukünftige Aktualisierungen einfacher gestaltet. Lediglich die Datenvisualisierung ist auf eine JS-Bibliothek angewiesen.

Um das *LiveLog* angezeigt zu bekommen, muss die IP-Adresse des Roboters angegeben werden und anschließend auf den Knopf *Roboter Verbinden* gedrückt werden. Da es sich hier um eine Echtzeit Ansicht handelt, wird der Graph über ein Intervall stetig aktualisiert. Dies geschieht im gleichen Abstand, wie Daten auf die Pipe über die KAREL Schnittstelle geschrieben werden. Hier lässt sich ein Timer Event einsetzen, um das gewünschte Verhalten zu erzielen. Während der Implementierung ist jedoch aufgefallen, dass der JS Timer nicht exakt ist und Intervalle nicht konstant eingehalten werden. Die Ausführung von Timern in JS ist von der CPU Last und Energiespareinstellungen des unterliegenden Betriebsystems abhängig [WHA22]. Infolge dessen, werden in der Pipe Ausgabe (vgl. Kapitel 6.1.3) jeweils zwei Datensätze übertragen, um keinen Datensatz durch die inkonsistente Ausführung der Intervalle zu verlieren. Der simplifizierte Codeausschnitt zeigt, wie die JS-Datei als Controller zwischen den Document Object Model (DOM) Elementen der HTML Seite und den Funktionen des Graphen fungiert:

[7] Individuelle Adresse eines (in diesem Fall) lokalen Gerätes im Netzwerk, welches Daten über das Internetprotokoll (IP) sendet [Lab22].

```
function checkConnection(){
  // überprüfen, ob die Verbindung gestartet oder gestoppt wurde
  if (document.getElementById('liveLog').checked) {
    // Roboter IP global setzten
    robotIP = document.getElementById('robotIP').value;
    // Intervall starten
    timerID = window.setInterval(refreshChart, 50);
  }else{
    // Intervall stoppen
    window.clearInterval(timerID);
  }
}
```

Listing 4: Starten/Stoppen des echtzeit Graphen

Die Darstellung des *SaveLogs* ist bedingt durch zwei Nutzereingaben. Zum einen müssen eine oder mehrere Aufnahmen vorhanden sein. Während eines aktiv laufenden *LiveLogs* muss zum Starten der Knopf *Aufzeichnung starten* gedrückt werden, sodass dieser rot aufleuchtet. Erneutes Drücken beendet die Aufnahme. Jetzt kann in den rechten Tab *SaveLog* gewechselt werden. Dort befindet sich die Aufnahme in Form eines grauen Knopfes, mit dem Zeitstempel des Aufnahmebeginns. Wird dieser vom Benutzer gedrückt, erscheint der aufgezeichnete Graph.

Die Daten der beiden Graphen werden jeweils in einem für die Datenvisualisierungsbibliothek lesbaren JavaScript Object Notation (JSON) Format gespeichert. Zusätzlich wird die aktuelle Position des Roboters alle 200ms über die im Kapitel 6.1.1 beschriebene Web-API ausgelesen und abgespeichert. Dadurch kann die Position des Roboters mit den Belastungswerten in Relation gesetzt werden. Zwecks besserer Übersichtlichkeit werden nur die letzten 10 Aufnahmen gespeichert. Es ist anzumerken, dass die Aufnahmen im *localStorage* vom Browser abgelegt werden. Dadurch bleiben die Aufnahmen persistent, auch nach Schließen des Browsers gespeichert. Diese Funktionalität gibt dem Benutzer genügend Zeit und Sicherheit, die Daten über den herunterladen Knopf neben der Aufnahme als *.csv* Datei an einem beliebigen Ort auf der Festplatte zu sichern.

### 6.2.3 Datenvisualisierung

Zur Datenvisualisierung kommt die JavaScript Bibliothek *D3* zum Einsatz. *D3*, kurz für *Data Driven Documents* hilft dem Programmierer Daten zum Leben zu erwecken. *D3* stützt sich auf den vollen Funktionsumfang der Web Standards HTML, Scalable Vector Graphics (SVG), CSS und JS, wodurch die Bibliothek eigenständig auf allen modernen Browsern auch mit großen Datenmengen sehr schnell und effizient ausgeführt wird. Kombiniert mit zahlreichen Visualisierungskomponenten und Datengetriebener

DOM Manipulation können interaktive Diagramme, Karten und Grafiken erstellt werden [Bos21]. Die Entscheidung, *D3* zu verwenden fiel einerseits auf die Kompaktheit der Bibliothek, da nur eine JS Datei inkludiert werden muss und zum anderen auf die hohe Flexibilität Visualisierungen zu erstellen.

Grundsätzlich setzt sich eine Visualisierung in *D3* aus verschiedenen Bestandteilen zusammen. Auf ein * * Objekt im DOM, welches mit einer ID ansprechbar ist, wird ein SVG mit definierter Höhe und Breite angefügt. Dieses SVG fungiert als „Leinwand“ auf der *D3* Elemente gezeichnet werden können. Mit dem Container Element *<g>* (engl. g-Tag) lassen sich verschiedene Elemente auf einem SVG gruppieren. So wird jeweils eine Gruppe für die Achsen des Diagramms erstellt, eine Gruppe für den Bereich in dem die Graphen gezeichnet werden und eine für die Graphen selbst. Wie in Abbildung 11 zu sehen, ist unter dem Hauptdiagramm noch eine Übersicht des gesamten Graphens zu sehen. Dieser Bereich nennt sich *brush* und ermöglicht, einen Ausschnitt des Graphen mit der Maus zu vergrößern und markiert die Stelle des angezeigten Ausschnittes.


Abb. 11: Ausschnitt einer Aufzeichnung, um zwei Achsen im Detail miteinander vergleichen zu können

Simple Kurvendiagramme mit einem Graphen sind mit relativ wenig Code in *D3* umsetzbar. Wenn jedoch mehrere Graphen in einem Diagramm dargestellt werden müssen, welche sich in kurzen Abständen aktualisieren, müssen mehrere Aspekte beachtet werden:

- anstelle eines statischen Datenarrays muss ein Puffer verwendet werden
- der Graph muss sich nach Links verschieben bei jedem neuen Wert
- die X-Achse muss anhand der auftretenden Maximalwerte mit skaliert werden
- die Y-Achse (Zeit) muss sich mit dem Graphen mit verschieben
- eine Update Routine muss periodisch die neuen Daten laden

Für die Datenhaltungsprobleme können bekannte Methoden verwendet werden. Anstelle eines statischen Arrays bietet sich eine Warteschlange mit definierter Länge an, welche nach dem FIFO-Prinzip arbeitet. Mit den nativ in JavaScript verfügbaren Befehlen *push* und *shift* können Daten jeweils am Ende des Arrays angefügt und am Anfang gelöscht werden. Wie im Listing 4 gezeigt, lässt sich der Funktion *window.setInterval()* eine Update Routine und ein Intervall übergeben, welche anschließend bis zum Löschen des Intervalls mit *window.clearInterval()* periodisch ausgeführt wird. Das Verschieben und Skalieren des Graphen und der Achsen passiert ebenfalls in der Update Routine des Echtzeit Graphen. Dafür müssen die Elemente über Funktionen der *D3* Bibliothek, um die Anzahl an Pixel nach links verschoben werden, um die sich der Graph und die Achsen bewegt haben. Intern werden die Elemente nicht verschoben, sondern neu auf das SVG gezeichnet und zwischen den alten und neuen Elementen überblendet. Dem Benutzer wird im Browserfenster eine flüssige Darstellung des sich bewegenden Graphen angezeigt.

# 7 Durchführung und Ergebnisse

Die Messreihen zeigen den Einfluss verschiedener Achsstellungen und Robotergeschwindigkeiten auf die Parameter Motorstördrehmoment (*Disturbance Torque*), Motordrehmoment (*Torque Monitor*) und Motorstrom (*OVC Monitor*). Alle Aufzeichnungen wurden am FANUC M10-iD Industrieroboter mit einer R-30iB Plus Steuerung durchgeführt.
Es wurde mit zwei Belastungen gemessen. Mit 4kg, dem Eigengewicht des Greifers am Roboter und mit 12,9kg bei dem der Greifer ein zusätzliches 8,9kg schweres Objekt trug. Neben Gewichten wurde in Experiment 7.4 ein definierter Drehmoment auf die sechste Achse des Roboters gegeben. Dazu wurde ein Drehmomentschlüssel am Flansch angebracht und die Motorlast unter verschiedenen Drehmomenten und Achsstellungen aufgezeichnet.

## 7.1 Einfluss der Geschwindigkeit

Eine weitere Kenngröße die mit der Logging-Software erfasst wurde, ist die Abhängigkeit der Belastung der Motoren im Verhältnis zu der Geschwindigkeit des Roboters. Zur Messung wurde der 4kg schwere Endeffektor als Traglast am Roboter angebracht. Der Roboterarm wurde in einer normalen Position (nicht gestaucht oder gestreckt) verfahren und führte zwischen zwei Punkten eine lineare, horizontale Bewegung aus.
Es wurden insgesamt 3 verschiedene Geschwindigkeiten aufgezeichnet: 1000 $mm/s$ ($3,6$ $km/h$), 500 $mm/s$ ($1,8$ $km/h$) und 1000 $mm/s$ mit Beschleunigungsrampen. Beim Verfahren mit Rampen wird die angegebene Geschwindigkeit später erreicht. Vor allem bei kurzen Strecken muss dies berücksichtigt werden, da der Roboter in diesem Fall die 1000 $mm/s$ nicht erreichen würde. In dieser Messreihe wurde eine Rampe von $ACC25$ (25% Beschleunigung) benutzt. FANUC gibt die für die Motoren zulässigen Werte auf einer Skala von 0% - 100% an. Bei Erreichen der 100% geht der Roboter in einen Fehlerzustand und zieht die in den Motoren verbauten Bremsen an, um die Motoren vor Überlast zu schützen.

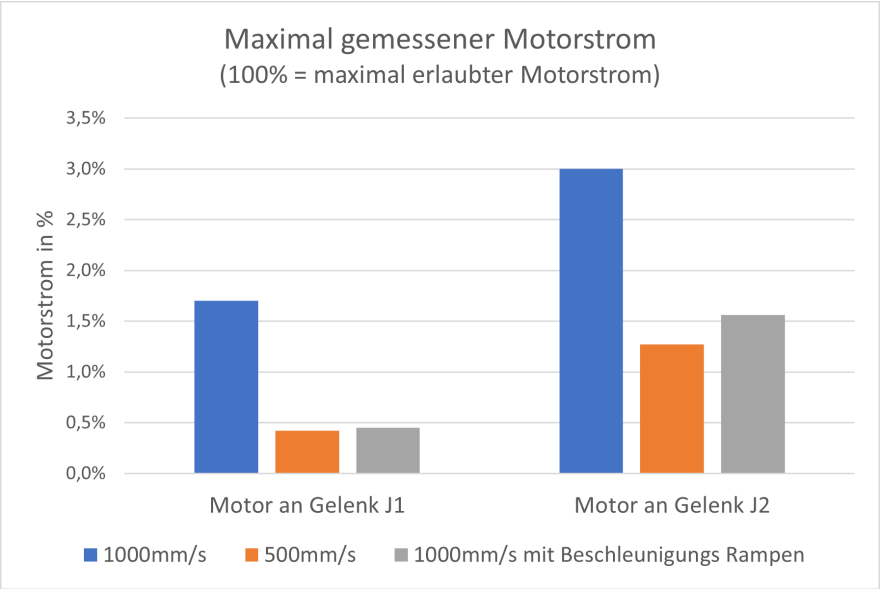


Abb. 12: Einfluss der Geschwindigkeit und Beschleunigung auf den Motorstrom an den Motoren der Gelenke J1 und J2 bei einer horizontalen Bewegung

Bei einer Geschwindigkeit von 1000 $mm/s$ liegt der gemessene, maximale Motorstromwert an Gelenk J1 bei 1,7%. Beim Verfahren mit halbierter Geschwindigkeit fällt der Maximalwert auf 0,42%. Unter Einsatz von Beschleunigungsrampen bei 1000 $mm/s$ liegt der Wert bei 0,45%. Bei einer Geschwindigkeit von 1000 $mm/s$ liegt der gemessene, maximale Motorstromwert an Gelenk J2 bei 3%. Beim Verfahren mit halbierter Geschwindigkeit fällt der Maximalwert auf 1,27%. Unter Einsatz von Beschleunigungsrampen bei 1000 $mm/s$ liegt der Wert bei 1,56%. Die Motoren an Gelenk J1 und J2 sind ausgelegt auf eine maximale Leistungsaufnahme von 2500 Watt.

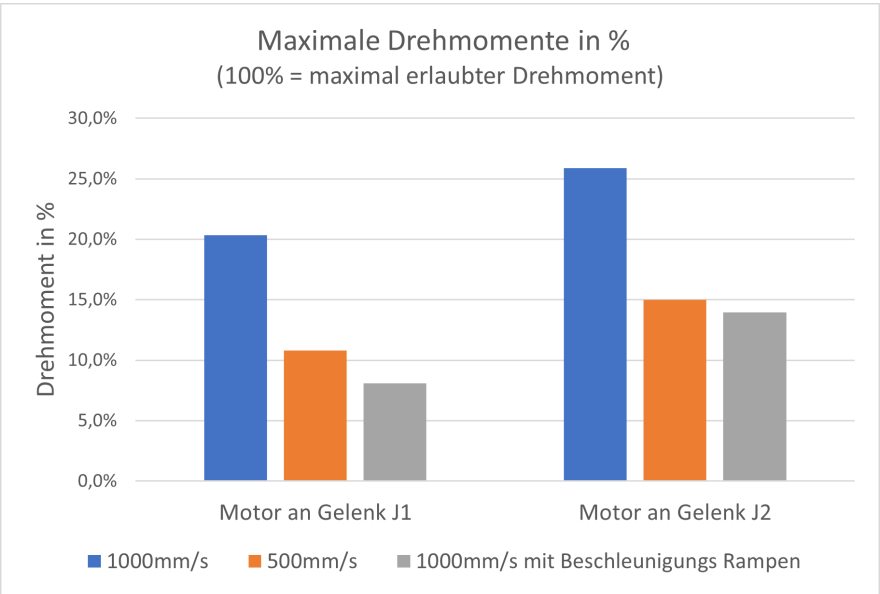


Abb. 13: Einfluss der Geschwindigkeit und Beschleunigung auf den Drehmoment an den Motoren der Gelenke J1 und J2 bei einer horizontalen Bewegung

Unter den gleichen Versuchsbedingungen zeigt Abbildung 13 den maximal gemessenen Drehmoment an Gelenk J1 und J2. Bei einer Geschwindigkeit von 1000 $mm/s$ liegt der gemessene, maximale Drehmoment an Gelenk J1 bei 20,3%. Beim Verfahren mit halbierter Geschwindigkeit fällt der Maximalwert auf 10,9%. Unter Einsatz von Beschleunigungsrampen bei 1000 $mm/s$ liegt der Wert bei 8,1%. An Gelenk J2 liegt bei einer Geschwindigkeit von 1000 $mm/s$ der maximal gemessenen Drehmoment bei 25,9%. Beim Verfahren mit halbierter Geschwindigkeit fällt der Maximalwert auf 15%. Unter Einsatz von Beschleunigungsrampen bei 1000 $mm/s$ liegt der Wert bei 14%.

## 7.2 Messreihe zur Gelenkstellung

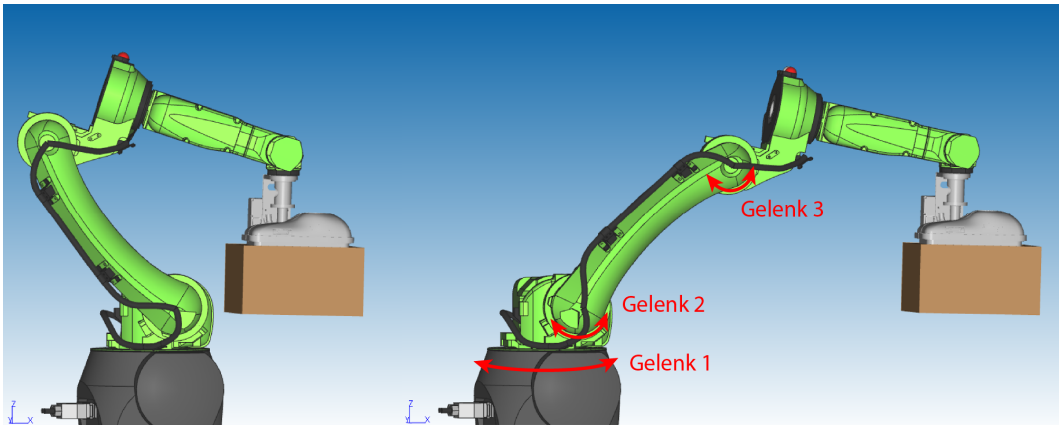


Abb. 14: Darstellung der Achsstellungen

Zur Messung des Einflusses der Gelenkstellung auf die Motoren, wurde der 4kg schwere Endeffektor des Roboters für die erste Messung in die Nähe des Roboternullpunkts verfahren und für die zweite Messung der Arm nahezu komplett gestreckt, wie in Abb. 14 zu sehen ist. Diese Position wurde solange gehalten, bis sich die Motorlast auf einen konstanten Wert eingepegelt hat und anschließend aufgezeichnet. Da die Achsen an den Gelenken 2 und 3 in Flucht mit dem Lastarm stehen, liegen die Belastungsänderungen dieser Gelenke im Fokus.

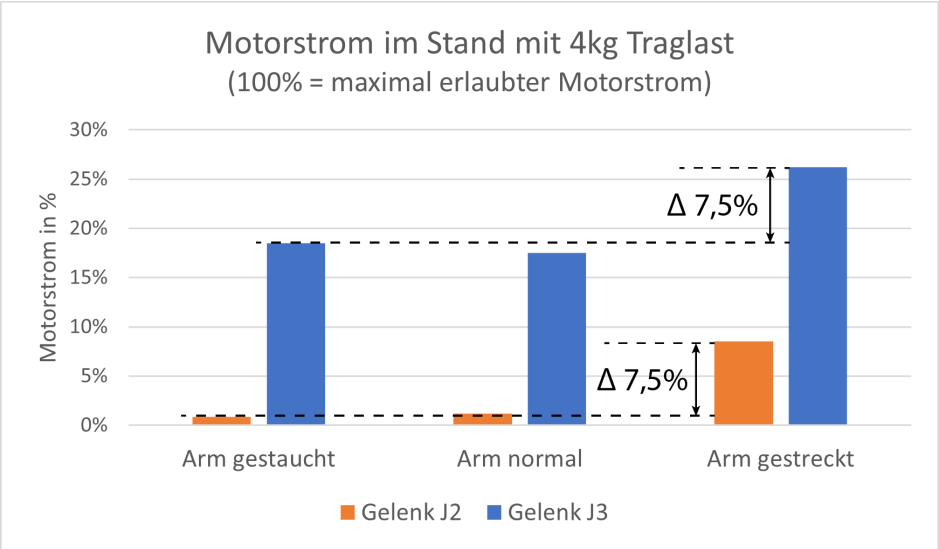


Abb. 15: Säulendiagramm mit Einfluss der Achsstellungen auf die Motorlast

Das Diagramm in Abb. 15 zeigt die ausgewerteten Daten der einzelnen Gelenkpositionen. Zusätzlich wurde noch eine mittlere Position gemessen, in welcher der Arm weder gestaucht, noch gestreckt ist. Gelenk J2 (orange) hat in der gestauchten und normalen Position eine durchschnittliche Belastung von 1%. In der gestreckten Position erhöht sich der Wert auf 8,5%. Dies entspricht einem Delta von 7,5%. Ein ähnliches Verhältnis liegt an Gelenk J3 (blau) vor. Die Motorlast steigt von 18,5% auf 26% an ($\Delta$ 7, 5%).

## 7.3 Relevanz der Traglasteinstellung

Mit dieser Fragestellung sollte untersucht werden, welche Auswirkung eine am TP falsch eingestellte Traglast, auf die Leistungsaufnahme und Ansteuerung der Motoren hat.

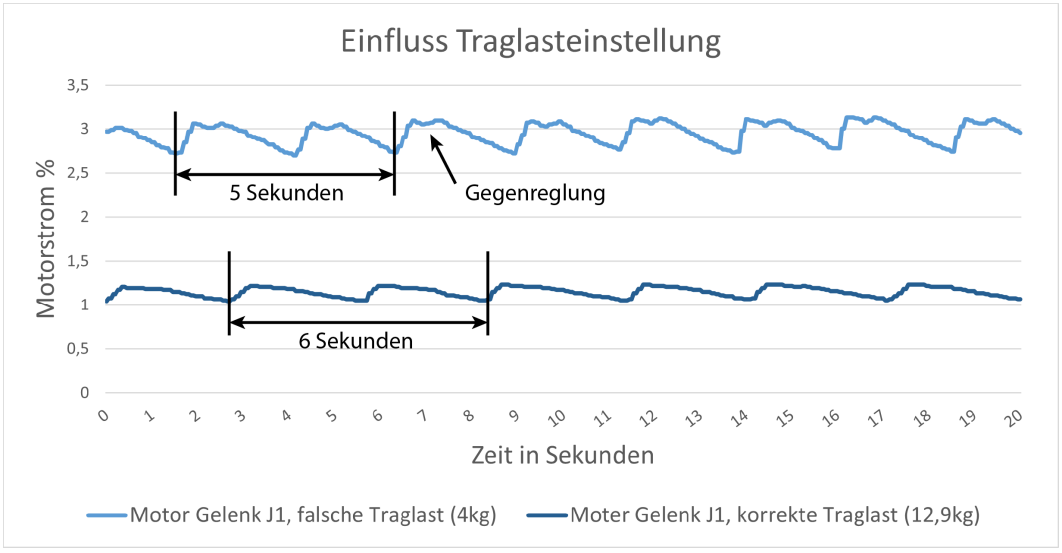


Abb. 16: Einfluss der Traglasteinstellung auf Motor an Gelenk J1

Abbildung 16 zeigt die Aufzeichnung des Motorstroms einer für Industrieroboter typische *pick and place* Bewegung. Dabei führt der Roboter zwei vertikale Bewegungen am Anfang und am Ende aus, die mit einer horizontalen Bewegung verbunden werden. Am 4kg schweren Endeffektor hängt eine Last von 8,9kg, sodass das Gesamtgewicht 12,9kg beträgt. Durchgeführt wurde die Messung mit der Maximalgeschwindigkeit des Roboters von 2000 $mm/s$ unter Verwendung von Rampen mit 75% Beschleunigung ($ACC75$) benutzt. Die Abbildung zeigt den Vergleich zwischen dem Motorstromprofil bei korrekt eingestellter Traglast und bei inkorrekt eingestellter Traglast. Mit korrekter Traglasteinstellung (dunkelblau) ist ein steiler Anstieg der Motorlast während der horizontalen Bewegung erkennbar. Danach sinkt der Motorstrom linear ab während Gelenk J1 stillsteht. Mit inkorrekter Traglasteinstellung (hellblau) ist ein noch steilerer Anstieg bei Ausführung der horizontalen Bewegung zu sehen. Danach fällt die Motorlast für ca. 200 $ms$ ab und steigt noch einmal leicht, bevor diese nach weiteren 200ms linear abfällt. Bei korrekter Traglast ist der Motorstrom geringer und die Zykluszeit für eine gesamte Bewegung verlängert sich von 5 auf 6 Sekunden.

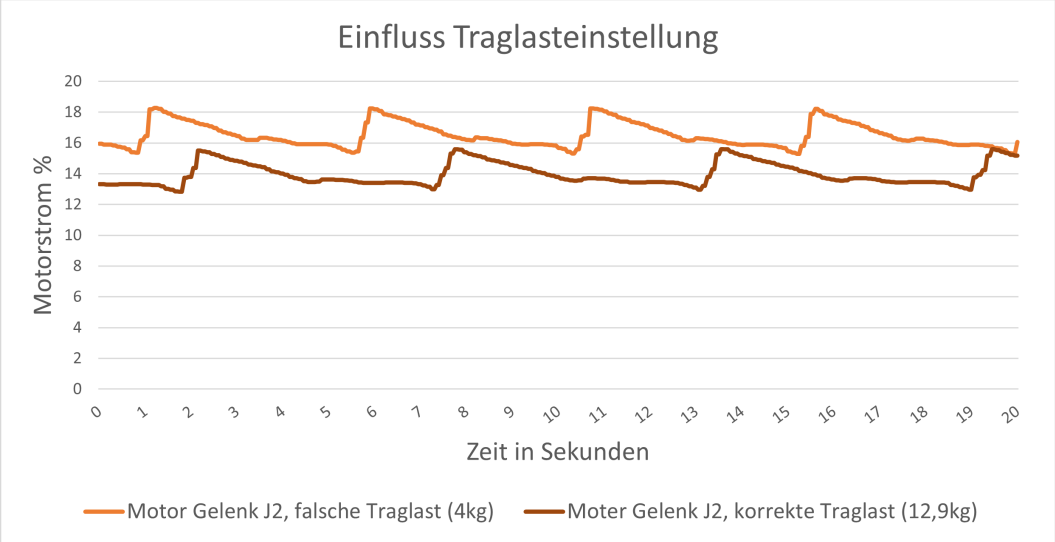


Abb. 17: Einfluss der Traglasteinstellung auf Motor an Gelenk J2

An Gelenk J2 wurden die Kurven in Abbildung 17 aufgenommen. Hier zeichnet sich die veränderte Zykluszeit stark ab. Wie in Abbildung 16 ist der Motorstrom bei korrekter Traglasteinstellung niedriger.

## 7.4 Machbarkeitsanalyse Verschraubroboter

Als Praxiseinsatz ließ sich die Software bei einem Kunden der SKDK GmbH einsetzen, welcher um eine Machbarkeitsanalyse gebeten hatte. Es sollte die Frage beantwortet werden, ob für die Verschraubung eines Werkstücks mit einem Drehmoment von 90 $Nm$ der kollaborative Roboter *CR-15iA* eingesetzt werden kann. Daraufhin wurde ein Messaufbau vorbereitet, welcher den Drehimpuls des Schraubers simulieren sollte.

Mit einem Drehmomentschlüssel wurde eine definierte Last in Nm an dem Endeffektor simuliert. Im Anhang zeigt Abbildung 18 zwei Messreihen mit jeweils unterschiedlichen Achsstellungen.
In der ersten Messreihe wurde eine für die Achsen ungünstige Position (siehe Abb. 18 linker Graph) gewählt, sodass der schwächste Motor die höchste Last tragen musste. In diesem Fall löste der Leistungsschutz des Motors bei 65 $Nm$ aus. In einer günstigeren Achsstellung (siehe Abb. 18 rechter Graph), in welcher Achse 4 und 6 in Flucht stehen und sich die Kraft auf 2 Motoren aufteilt, löste der Schutz erst bei 78 $Nm$ am schwächsten Motor aus. Die aufgezeichneten Daten zeigen, dass sogar bei optimaler Achsstellung die 90 $Nm$ nicht erreicht werden. Das in den Graphen zu sehende “Tal„ ist experimentell bedingt durch erneutes Ansetzten des Drehmomentschlüssels und hat keine Relevanz für die Fragestellung.

# 8 Diskussion

Die Logging-Software wurde entwickelt, um den Einfluss der Parameter Gelenkstellung, Geschwindigkeit, Beschleunigung und Traglast eines Roboters der Firma FANUC (Oshino, Japan) zu untersuchen. Die Software kann den Motorstrom und Motordrehmoment aufzeichnen und wiedergeben. Der Vorteil ist, dass im Gegensatz zu der bisher statischen Grenzwertberechnung über die aufgezeichneten Ergebnisse, Grenzwerte in Relation zu Geschwindigkeit und Beschleunigung gemessen werden können. Zusätzlich bietet die Software eine benutzerfreundliche Bedieneroberfläche. Evaluiert wurde die Logging-Software an einer FANUC R-30iB Plus Steuerung. Die Robotersteuerung entspricht der neusten Hardwaregeneration und ist der Standard für alle Robotermodelle von FANUC. Um die Funktionalität der Software zu testen, wurden verschiedene Messungen durchgeführt.
Neu ist, dass durch die Logging-Software die Reduktion des Motorstroms quantifiziert werden konnte. Bei horizontaler Bewegung an Gelenk 1 reduzierte sich der maximale Motorstrom bei halber Geschwindigkeit auf ein drittel. An Gelenk 2 reduzierte sich der Motorstrom um 50%. Der Einsatz von Beschleunigungsrampen bei maximaler Geschwindigkeit zeigte die gleiche Auswirkung auf den Motorstrom, wie das Halbieren der Geschwindigkeit. Während der Messung konnte beobachtet werden, dass nach Einstellen einer sanfteren Beschleunigung (Rampe), die Verankerung des Roboters und angrenzende Strukturen spürbar weniger vibrierten bzw. erschütterten. In einem weiteren Schritt wurde das Drehmoment an den Gelenken gemessen, um die Belastung auf Getriebe und Motoren zu beurteilen. Mit Beschleunigungsrampen konnten an den Gelenken 1 und 2 die niedrigsten Werte aufgezeichnet werden. Dies zeigt, dass nicht nur die Motorlast durch die Rampen reduziert wurde, sondern auch die auf die gesamte Mechanik wirkenden Kräfte. An Gelenken die nur Haltearbeit leisten mussten, kam es zu keiner signifikanten Reduktion der Kräfte.

Entgegen der Erwartungen, konnte bei Betrachtung des gesamten Bewegungszyklus mit Rampen keine Senkung des Energieverbrauchs festgestellt werden. Beim Aufaddieren der Motorströme wurde ein ca. 40% höherer Energieverbrauch beim Verfahren mit Rampen ausgerechnet, im Gegensatz zu der Bewegung ohne Rampen bei 1000 $mm/s$. Der Verbrauch bei 500 $mm/s$ war zu 60% höher. Daraus lässt sich folgern, dass der Verbrauch maßgeblich durch die für die Bewegungsausführung benötigte Zeit bedingt ist. Auch in der Publikation [CSK11] wurde aufgezeichnet, dass langsamere Bewegungen nicht zwangsläufig am effizientesten sind und der Gesamtenergieverbrauch stark vom Robotermodell abhängig ist.

Die Aufzeichnung in Abschnitt 7.2 wurde im Stand durchgeführt und liefert eine Aussage zum maximal benötigten Haltemoment. Kurze Bewegungen mit gestrecktem Arm

können von den Motoren kompensiert werden. Bei langen, kontinuierlichen Bewegungsanweisungen in Achsstellungen mit gestrecktem Arm, erhöht sich die Motorlast an den Gelenken J2 und J3 um 7,5%. Diese zusätzliche Belastung kann durch eine vorteilhaftere Achsstellung vermieden werden.

Herstellerseitig wird empfohlen, vor allem bei hohen Gewichten, die Traglast exakt einzustellen, damit Pfade, Beschleunigungen und Sicherheitsfunktion ordnungsgemäß vom Roboter berechnet werden können [Rob19]. Auch auf dem Handprogrammiergerät wird dem Programmierer die Warnung gegeben, dass das Ändern der Traglast Auswirkungen auf den Pfad und die Zykluszeit hat. Mit der Logging-Software konnten die Auswirkungen auf den Motorstrom bei einer falsch eingestellten Traglast untersucht werden. Die aufgezeichneten Daten zeigten einen direkten Einfluss auf den Motorstrom und die Zykluszeit. Bei korrekter Traglasteinstellung ist der Motorstrom geringer und die Zykluszeit verlängert sich. Bei inkorrekter Traglasteinstellung werden die Rampen nicht optimal berechnet. Dadurch muss der Motor den Trägheitsmoment der Last kompensieren. Dieses Nachregeln zeigt sich über das nicht lineare Verhalten des Motorstroms.

Durch die Experimente konnte gezeigt werden, dass die programmierte Logging-Software zur Prozessoptimierung geeignet ist. Die verwandten Arbeiten (siehe Kapitel 3) verfolgen unterschiedliche Ansätze zur Aufzeichnung der Motorstöme.
Paes et al. [PDE+14] ist vergleichbar bezüglich des nicht-invasiven, “Standalone„ Ansatzes zur Aufzeichnung der Motorstöme. Hierfür wird eine externe Messhardware über Stromklemmen an der Motorverdrahtung, ohne Umbau des Roboters angebracht. Motorströme können über einen kurzen Zeitraum von 80 Sekunden in einer hohen Auflösung von 3200Hz gemessen werden. Die Datenrate der Logging-Software ist mit 20Hz deutlich niedriger und es können keine rohen Spannungswerte ausgelesen werden, dafür lassen sich beliebige Werte der Robotersteuerung anzeigen und über längere Zeiträume aufzeichnen. Über die Aufzeichnung hinaus hat die Arbeitsgruppe eine Software zur Optimierung der Bewegungsplanung geschrieben. Dieser Prozess führte zu einer Einsparung von 5% der benötigten Zeit und Energie. Die Logging-Software konnte ähnliche Ergebnisse experimentell bestätigen.

Einen anderen Ansatz verfolgte die Arbeitsgruppe Sabry, A.H. et al. [SNSAAK20]. Zur Datensammlung wurde eine eigene Messstation entwickelt, welche drahtlos mit einem Empfänger über ZigBee kommuniziert. Die Lösung lässt sich nicht einfach rekonstruieren, oder während des Betriebs nachrüsten. Die in der vorliegenden Arbeit entwickelte Software benötigt keine spezielle Mess- und Kommunikationshardware, da über LAN/WLAN mit der Robotersteuerung kommuniziert wird. Bei Sabry wird der gesamte Stromverbrauch über einen Widerstand gemessen, welcher in Reihe mit der Stromversorgung des Roboterarms eingesetzt werden muss. Zusätzlich benutzt die Ar-

beitsgruppe ein Neuronales Netz, welches Abweichungen im Gesamtstromverbrauch einem fehlerhaften Gelenk zuordnet. Die Logging Software dagegen zeichnet den Strom direkt für jeden einzelnen Motor auf, kommt jedoch nur auf ein sechstel der Auflösung.

Die Datenvisualisierung der vorliegenden Arbeit wurde als Webapplikation mit einer benutzerfreundlichen Oberfläche implementiert. Dieser Ansatz der Visualisierung findet auch Anwendung in der FANUC Software AIServoMonitor [Ltd21a]. Die beiden Systeme unterscheiden sich jedoch in ihrer Anwendung, da die FANUC Software auf die Überwachung von FANUC CNC Maschinen ausgelegt ist. Die Logging-Software erweitert diese Funktionalität auf FANUC Robotersteuerungen.

Das Gesamtsystem aus Datenlogging und Datenvisualisierung wurde einer Machbarkeitsanalyse unterzogen und demonstriert den wirtschaftlichen Nutzen der Software. Es sollte die Frage beantwortet werden, ob die Leistung eines Robotersystems für die Verschraubung eines Werkstücks mit 90Nm ausreicht. Da der Roboter während der Verschraubung des Werkstücks steht, spielen Trägheitsmomente und Beschleunigung in diesem Anwendungsfall keine Rolle. Es kann nicht die Formel des Herstellers zur Berechnung der Machbarkeit genutzt werden, weil sich die Berechnungen des Herstellers auf das Verfahren des Roboters mit maximaler Geschwindigkeit beziehen. Die Logging-Software konnte hingegen die Kraftverteilung der jeweiligen Achsen im Stand aufzeichnen. Anschließend konnte analysiert werden, ob und wo Grenzwerte überschritten wurden. In diesem Fall musste festgestellt werden, dass das System für diesen speziellen Anwendungseinsatz nicht leistungsfähig genug ist. Belastungsspitzen bis 78Nm konnte die Mechanik mit optimal gewählter Achsstellung aushalten, um jedoch zuverlässig mit 90Nm arbeiten zu können muss ein anderes, leistungsstärkeres Robotersystem eingesetzt werden.

# 9 Fazit

In dieser Arbeit wurde eine Logging-Software erfolgreich implementiert und zur Prozessoptimierung eingesetzt. Die Daten werden über einen nicht-invasiven Ansatz, d. h. ohne Änderung der Hardware, von der Robotersteuerung ausgelesen und in einer Web-Applikation visualisiert. Es konnte die Reduktion der Motorlast durch den Einsatz von Beschleunigungsrampen gezeigt werden. Zusätzlich konnte die Belastung der Motoren bei extremen Achsstellungen über die Logging-Software quantifiziert werden. Die Daten zeigen, dass sich eine am Handprogrammiergerät inkorrekt eingestellte Traglast, direkt auf die Leistungaufnahme und Ansteuerung der Motoren auswirkt. Zuletzt fand die Software erfolgreich Anwendung in einem praxisbezogenen Projekt zur Evaluierung der Leistungsfähigkeit eines 6-Achs-Roboters.
Zukünftig können weitere empirische Forschungen an aussagekräftigen Systemvariablen, z.B. Temperatur oder Positionsfehler der Motoren, mithilfe der Software durchgeführt werden. Auch Untersuchungen zu verschiedenen Bewegungspfaden sind Gegenstand zukünftiger Arbeiten. Die Software ließe sich durch einen Automatismus erweitern, der computergesteuert Soll- und Ist- Werte überwacht. Damit könnte die Software zusätzlich zur Wartung von Robotersystemen eingesetzt werden.

# Literaturverzeichnis


[al.22] AL., Unicard ic; e.: *Thin Client*. https://de.wikipedia.org/wiki/Thin_Client. Version: 2022, Abruf: 09. März 2022

[Asf20] ASFOUR, Fabian Tamim; P. Tamim; Paus: *Einführung in die Robotik [Vorlesungsfolien]*. http://dx.doi.org/10.5445/DIVA/2019-C44. Version: 2019/2020, Abruf: 02. Feb. 2022

[Bar98] BARTENSCHLAGER, Hans; Schmidt G. Jörg; Hebel H. Jörg; Hebel: *Handhabungstechnik mit Robotertechnik*. Springer Verlag, 1998. http://dx.doi.org/https://doi.org/10.1007/978-3-663-12166-4. http://dx.doi.org/https://doi.org/10.1007/978-3-663-12166-4

[Bos21] BOSTOCK, Mike: *Data-Driven Documents - Overview*. https://d3js.org/. Version: 2021, Abruf: 16. März 2022

[CSK11] CHEMNITZ, Moritz ; SCHRECK, Gerhard ; KRÜGER, Jörg: Analyzing energy consumption of industrial robots. In: *ETFA2011*, 2011, S. 1–4

[Fur] FURUYA, Yamanashi J. Yoshitake: *Robotersystem, das eine Abnormalität eines festgehaltenen Werkstücks beurteilt, und Abnormalitäts-Beurteilungsverfahren*. https://patents.google.com/patent/DE102016003639A1

[IFR21] IFR: *Robot Sales Rise Again*. https://ifr.org/ifr-press-releases/news/robot-sales-rise-again. Version: 2021, Abruf: 13. April. 2022

[Kun22] KUNBUS.DE: *Feldbus Grundlagen*. https://www.kunbus.de/feldbus-grundlagen.html. Version: 2022, Abruf: 05. März 2022

[Lab22] LAB, Kaspersky: *IP-Adresse – Definition und Erläuterung*. https://www.kaspersky.de/resource-center/definitions/what-is-an-ip-address. Version: 2022, Abruf: 15. März 2022

[Lie18] LIEBL, Micha; Luft C. Stefan; Seidel S. Stefan; Seidel: *ERP-Systeme im Wandel der Digitalisierung*. https://betrieb-machen.de/nachgelesen_robotik/. Version: 2018, Abruf: 4. April. 2022

[Ltda] LTD., FANUC: *OPC Server*. https://www.fanuc.eu/de/de/cnc/connectivity/opc-server, Abruf: 25. März 2022

[Ltdb] LTD., FANUC: *R-30+B/ R-30+B Mate CONTROLLER Optional Function OPERATOR'S MANUAL*. Version 5

[Ltdc] LTD., FANUC: *R-30iA/Mate, R-30iB/MATE, and R-30iB Plus/Mate Plus/Compact Plus Controller KAREL Reference Manual*. Rev. L

[Ltdd] LTD., FANUC: *R-30iB/R-30iB Plus CONTROLLER MAINTENANCE MANUAL*. Version 9

[Ltd21a] LTD., FANUC: *AI Servo Monitor - Detect abnormalities and predict failures*. Broschüre. https://emo2021.fanuc.eu/wp-content/uploads/2021/09/FANUC-at-EMO-AI-Servo-Motor.pdf. Version: 2021, Abruf: 25. März 2022

[Ltd21b] Ltd., FANUC: *FANUC MT-LINKi - Operational Management Software for Factory IoT*. Broschüre. https://www.fanuc.co.jp/en/product/catalog/pdf/cnc/MT-LINKi(E)-01a.pdf. Version: 2021, Abruf: 25. März 2022

[Ott22] Otto, Jacob; et a. Mark; Thornton T. Mark; Thornton: *Bootstrap - Getting Started*. https://getbootstrap.com/docs/5.1/getting-started/introduction/. Version: 2022, Abruf: 15. März 2022

[PDE+14] Paes, Koen ; Dewulf, Wim ; Elst, Karel V. ; Kellens, Karel ; Slaets, Peter: Energy Efficient Trajectories for an Industrial ABB Robot. In: *Procedia CIRP* 15 (2014), 105-110. http://dx.doi.org/https://doi.org/10.1016/j.procir.2014.06.043. – DOI https://doi.org/10.1016/j.procir.2014.06.043. – ISSN 2212–8271. – 21st CIRP Conference on Life Cycle Engineering

[PP17] Prof. Plate, Jürgen: *Grundlagen Netzwerkprogrammierung - Die Socket-Netzwerkschnittstelle*. http://www.netzmafia.de/skripten/inetprog/inetprog1.html. Version: 2017, Abruf: 12. März 2022

[Pro90] Produktionstechnik, VDI-Gesellschaft: *VDI-Richtlinie 2860, Montage- und Handhabungstechnik*. Beuth-Verlag, 1990

[Rob19] Robotics, One: *TO ACC OR NOT TO ACC*. https://www.onerobotics.com/posts/2013/to-acc-or-not-to-acc/. Version: 2019, Abruf: 31. März 2022

[SNSAAK20] Sabry, Ahmad H. ; Nordin, Farah H. ; Sabry, Ameer H. ; Abidin Ab Kadir, Mohd Z.: Fault Detection and Diagnosis of Industrial Robot Based on Power Consumption Modeling. In: *IEEE Transactions on Industrial Electronics* 67 (2020), Nr. 9, S. 7929–7940. http://dx.doi.org/10.1109/TIE.2019.2931511. – DOI 10.1109/TIE.2019.2931511

[Tea] Team induux: *Robotersteuerung*. https://wiki.induux.de/Robotersteuerung, Abruf: 02. Feb. 2022

[WHA22] WHATWG: *HTML Living Standard - 8.6 Timers*. https://html.spec.whatwg.org/multipage/timers-and-user-prompts.html#timers. Version: 2022, Abruf: 3. April. 2022

[Wü18] Wüst, Prof. Dr. K.: *Grundlagen der Robotik*. 2018

**Eidesstattliche Erklärung**

Brandenburg, den 16. April 2022

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt wurde.

Ort, Datum

Unterschrift
*(Jan Philipp Seeland)*

## Anhang

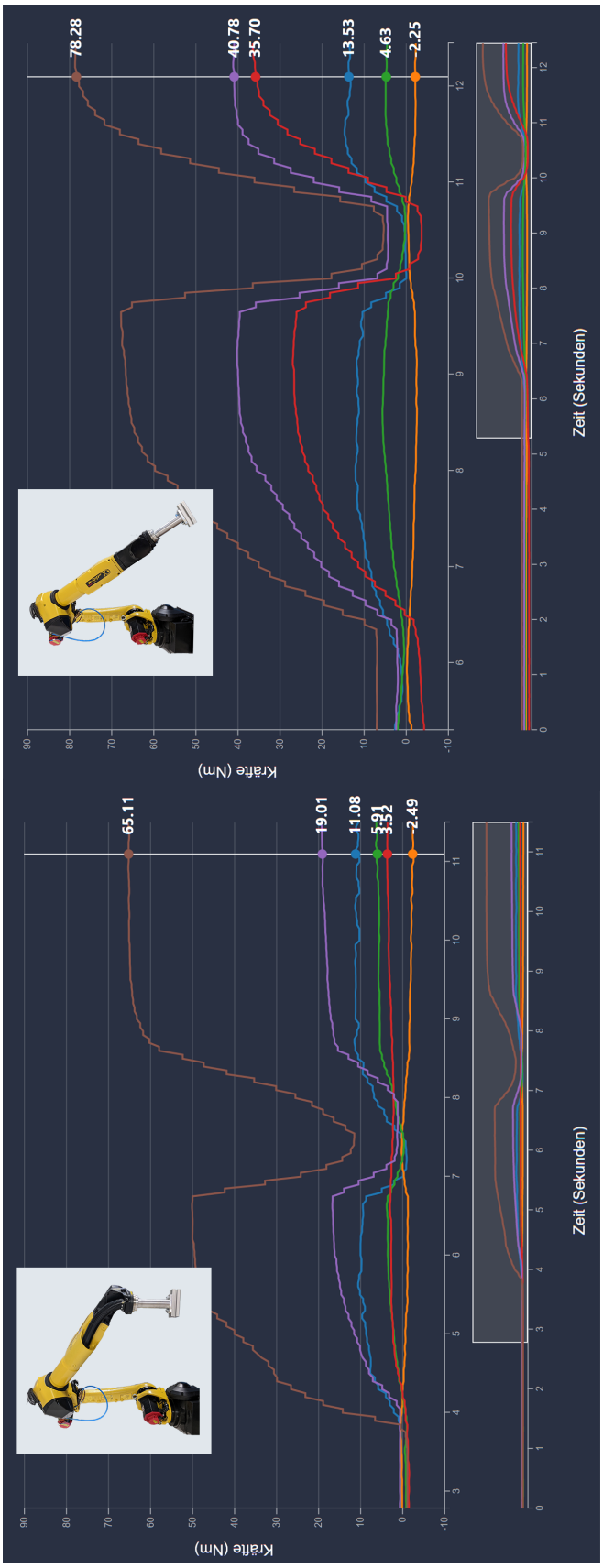


Abb. 18: Aufzeichnung der auf die Gelenke wirkenden Kräfte in Nm. Links: ungünstige Position der 6 Achsen, Rechts: Gelenk 4 und 6 in Flucht (Gelenk 1 = blau, Gelenk 2 = orange, Gelenk 3 = grün, Gelenk 4 = rot, Gelenk 5 = lila, Gelenk 6 = braun)