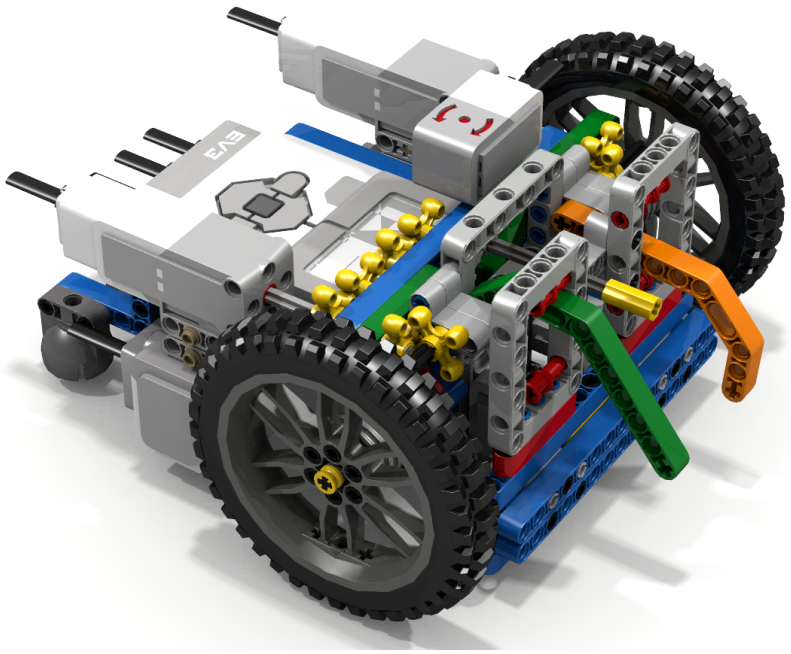
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

EIT FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIONSTECHNIK

LEGO-Praktikum. entwickeln + programmieren + optimieren

Berichte der Studierenden zum Projektseminar Elektrotechnik/Informationstechnik



Eine Schriftenreihe der Otto-von-Guericke-Universität Magdeburg, Fakultät für Elektrotechnik- und Informationstechnik, Institut für Medizintechnik sowie Institut für Elektrische Energiesysteme

Herausgeben von:
Mathias Magdowski, Thomas Schallschmidt, Jörg Petzold und Marius Klahm

Band 8 vom Wintersemester 2024/2025

Inhaltsverzeichnis

Impressum

Herausgeber:	Mathias Magdowski, Thomas Schallschmidt, Jörg Petzold und Marius Klahm Institut für Medizintechnik, Institut für Elektrische Energiesysteme Fakultät für Elektro- und Informationstechnik Otto-von-Guericke-Universität Magdeburg Postfach 4120, 39016 Magdeburg
DOI:	10.24352/UB.OVGU-2025-036
ISSN:	2629-6160
Redaktionsschluss:	Oktober 2025
Seminarzeitraum:	03. Februar – 17. Februar 2025
Bezug:	Open Access, Digitale Hochschulbibliothek Sachsen-Anhalt `http://edoc2.bibliothek.uni-halle.de/`

1. Auflage, Magdeburg, Otto-von-Guericke-Universität, 2025

Erstellung des Sammelbandes mittels LaTeX, `hyperref` und pdfpages

„ Robo Chamäleon “ mit Lego Mindstrom

Mezentsov Tymur, Elektro-und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—- Jährlich findet an der Otto-vonGuericke-Universität in Magdeburg das Projektseminar zur Elektro- und Informationstechnik statt. Im Rahmen der diesjährigen Projektwerkstatt wurde ein spezieller Roboter namens Chamäleon entwickelt. Dieser nutzt einen Farbsensor, um die Farbe der Oberfläche zu erkennen und darauf mit vordefinierten Bewegungen zu reagieren. Für die Konstruktion und den Bau des Roboters wurden LEGO Mindstorms Sets und der Steuercomputer LEGO EV3 verwendet. Die Software wurde mit MATLAB implementiert. In diesem Beitrag werden der Aufbau und die Funktionsweise des Roboters vorgestellt. Außerdem werden Probleme, die während des Entwicklungsprozesses aufgetreten sind, sowie deren Lösungen erläutert.**

***Schlagwörter*—Farberkennung, Roboter, LEGO Mindstorms, Matlab, Programmierung,**

I. Einleitung

DIE Moderne Robotik bietet nicht nur Lösungen für industrielle oder sicherheitsrelevante Anwendungen, sondern auch neue Möglichkeiten der Interaktion und Unterhaltung. Insbesondere biomimetische Roboter, die sich an der Natur orientieren, eröffnen faszinierende Perspektiven für den Alltag. Ein Beispiel für eine solche Entwicklung ist der ChamäleonRoboter, der in diesem Projekt vorgestellt wird. Inspiriert von der Fähigkeit echter Chamäleons (Abbildung 1), ihre Umgebung wahrzunehmen und entsprechend zu reagieren, nutzt dieser Roboter einen Farbsensor, um die Farbe der Oberfläche zu erkennen [1]. Je nach erkannter Farbe führt er verschiedene Bewegungen aus, wodurch er ein dynamisches und interaktives Verhalten zeigt. Diese Art der Robotik verbindet Technik und Spiel, indem sie nicht nur eine präzise Steuerung und Sensorik erfordert, sondern auch ein gewisses Maß an Unvorhersehbarkeit und Reaktionsfähigkeit bietet. Ziel dieses Projekts ist es, eine unterhaltsame und lehrreiche Anwendung von Robotik zu demonstrieren, die sowohl technisches Interesse weckt als auch als interaktive Erfahrung genutzt werden kann.

II. Vorbetrachtungen

In diesem Abschnitt werden die zentralen Werkzeuge dieses Projekts vorgestellt und ihre Funktionen im Zusammenhang mit der Durchführung spezifischer Aufgaben oder der Reaktion auf Eingaben beschrieben.

A. Programmierung von Robotern mit LEGO Mindstorms

Die Programmierung von Robotern erfordert eine Kombination aus mechanischen Bauelementen, elektronischen Komponenten und einer geeigneten Software. Für dieses Projekt wurde das LEGO Mindstorms EV3 Set verwendet, das eine Vielzahl von Sensoren, Motoren und eine benutzerfreundliche

DOI: 10.24352/UB.OVGU-2025-006

Abbildung 1. Chamäleon in freier Wildbahn

Entwicklungsumgebung bietet. Die Möglichkeit, verschiedene Sensoren zu integrieren, macht es besonders geeignet für interaktive Robotikprojekte."

B. Chamäleon-Roboter: Ein interaktives Universitätsprojekt

Im Rahmen des Universitätsprojekts an der Otto-von-Guericke-Universität wurde 2024 ein Chamäleon-Roboter entwickelt. Dieser nutzt einen Farbsensor zur Erkennung der Oberfläche und reagiert darauf mit verschiedenen Bewegungsmustern. Das Ziel war es, ein unterhaltsames, interaktives System zu schaffen, das technische Prinzipien spielerisch vermittelt."

C. MATLAB als Entwicklungsumgebung für die Robotik

Die Software für den Roboter wurde in MATLAB programmiert, einer leistungsfähigen Umgebung für numerische Berechnungen, Datenanalyse und Robotersimulation."

1) Signalverarbeitung und Motorsteuerung in MATLAB: MATLAB bietet zahlreiche Werkzeuge für die Signalverarbeitung, die Steuerung von Motoren sowie die Implementierung von sensor-basierten Algorithmen. In diesem Projekt wurde MATLAB verwendet, um die Sensordaten des Farbsensors zu verarbeiten und entsprechende Bewegungen der Motoren auszuführen."

III. KONSTUKTION UND PROGRAMMIERUNG

A. Aufbau

Das Design des Chamäleon-Roboters ist technisch einfach, aber funktional. Um sicherzustellen, dass der Roboter in der Lage ist, Farben der Oberfläche zu erkennen und darauf mit

spezifischen Bewegungen zu reagieren, sind Sensoren und Motoren erforderlich. Der Farbsensor ermöglicht die Erfassung der Umgebung, während die Motoren verschiedene Reaktionen ausführen, um das Verhalten eines Chamäleons nachzuahmen. Im Folgenden sind die Arten und Eigenschaften der benötigten Motoren und Sensoren aufgeführt. Durch die Kombination aus mechanischen Komponenten, intelligenter Sensorik und MATLAB -Steuerung entstand ein Roboter, der nicht nur auf äußere Reize reagiert, sondern auch ein gewisses Maß an Unvorhersehbarkeit und interaktivem Verhalten aufweist.

1) Motoren: Für den Chamäleon-Roboter werden drei große Motoren verwendet. [2] Einer der Motoren ist für die Vorwärts- und Rückwärtsbewegung des Roboters verantwortlich. Der zweite Motor steuert die Bewegung des Chamäleon-Schwanzes, während der dritte Motor für die Drehung des Förderbands zuständig ist, das die Farbänderung simuliert.

Abbildung 2. großer Motor

2) Farbsensor: Der Chamäleon-Roboter verwendet einen Farbsensor, [3] um die Farbe der Oberfläche zu erkennen und entsprechend darauf zu reagieren. Sobald der Sensor eine der vordefinierten Farben erkennt, aktiviert er die zugehörigen Motoren, um die gewünschte Bewegung auszuführen.

3) Förderband (Farbwechsel-System): Der Chamäleon- Roboter besitzt ein Förderband,(Abbildung 4) das gezielt bestimmte Farben anzeigt. Erkennt der Farbsensor eine vordefinierte Farbe, steuert das System den Motor, der das Förderband um einen festgelegten Winkel dreht. Jede erkannte Farbe löst eine präzise Bewegung aus, sodass der nächste gewünschte Farbbereich sichtbar wird. Dadurch simuliert der Roboter das Farbwechsel-Verhalten eines Chamäleons.

Abbildung 3. Farbsensor

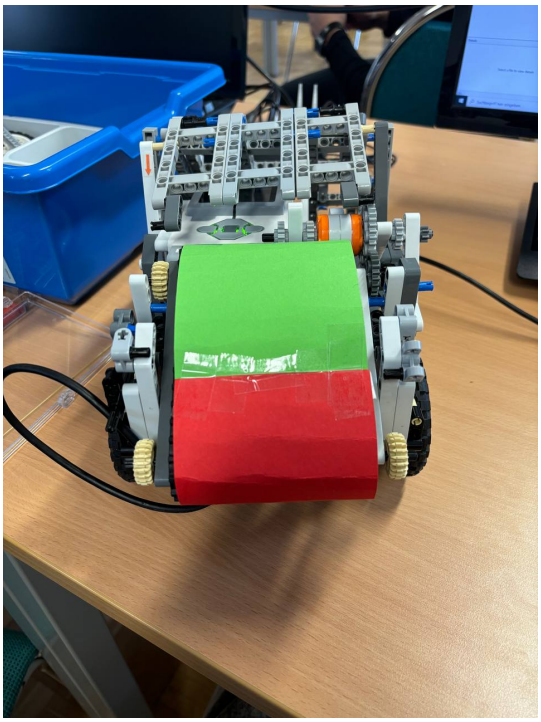


Abbildung 4. Förderband (Farbwechsel -System)

B. Programierung

Der EV3 ist ein Steuerungscomputer mit Anschlüssen für mehrere Sensoren und Motoren sowie USB- und Bluetooth-Schnittstellen. Die Steuerung des EV3-Geräts und der Sensoren/Motoren erfolgt über MATLAB mit dem EV3-Toolbox. (Abbildung 6) Es ist möglich, mit Hilfe dieses Tools die Messwerte der Sensoren abzulesen und die Motoren zu steuern. Eine Kennzeichnung der Anschlüsse mit Buchstaben (für Motoren) und Zahlen (für Sensoren) erleichtert die Identifikation der elektronischen Elemente. In der RWTH Toolbox sind verschiedene Befehle verfügbar, die zur Inbetriebnahme von Sensoren und Motoren genutzt werden können. MATLAB ist eine Plattform für Programmierung und numerische Berechnungen. Wie bei anderen Programmiersprachen enthält es Schleifen wie die While-Schleife, die es ermöglicht, eine Sequenz von Anweisungen wiederholt auszuführen, ohne die entsprechenden Anweisungen mehrmals schreiben zu müssen. Es enthält Anweisungen wie Anweisungen, die bestimmte Teile des Programms nur ausführen, wenn bestimmte Bedingungen erfüllt sind. Die EV3- Einheit empfängt Befehle vom Computer und leitet sie an die Motoren weiter.

C. Probleme

Die Sensoren und Motoren des Chamäleon -Roboters haben begrenzte Funktionen, was zu mehreren Herausforderungen im Design und in der Programmierung führte. Der Farbsensor musste genau positioniert werden, damit er die Farben der

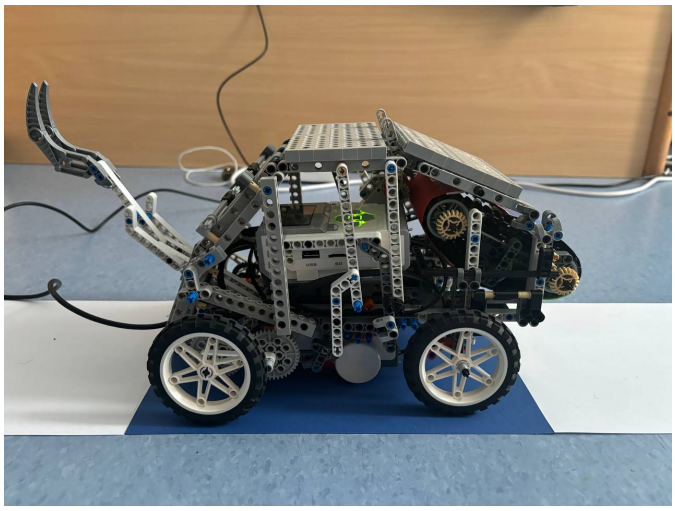


Abbildung 5. fertiger “ Robo Chamäleon ”

```
farbe = brick.ColorSensor(1);
if farbe >= 15  farbe <= 30
brick.motorA.stop();
-420 Grad
brick.motorC.limitValue = 420;
brick.motorC.power = -50;
brick.motorC.smoothStart = 1;
brick.motorC.start();
brick.motorC.waitFor();
pause(4);
elseif farbe >= 38  farbe <= 50 -425 Grad
brick.motorC.limitValue = 425;
brick.motorC.power = -50;
brick.motorC.smoothStart = 1;
brick.motorC.start();
brick.motorC.waitFor();
elseif farbe >= 60 && farbe <= 75 -360 Grad
brick.motorC.limitValue = 360;
brick.motorC.power = -50;
brick.motorC.smoothStart = 1;
brick.motorC.start();
brick.motorC.waitFor();
end
brick.motorA.power = -50;
pause(1);
brick.motorA.power = 50;
pause(1);
```

Abbildung 6. Codekette

Oberfläche zuverlässig erkennen konnte. Anfangs gab es Probleme mit der Farberkennung, da externe Lichtquellen die Messergebnisse beeinflussten. Zudem traten Schwierigkeiten bei der Steuerung der Motoren auf, insbesondere bei der Synchronisation der Bewegungen zwischen dem Förderband und dem Schwanzmechanismus. Durch Anpassungen in der MATLAB -Programmierung und Tests konnten diese Probleme jedoch schrittweise gelöst werden.

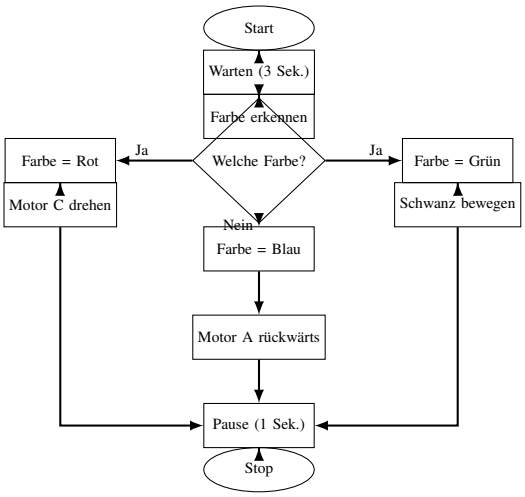


Abbildung 7. Flussdiagramm für den Programmablauf

IV. Ergebnisdiskussion

Die Funktionsfähigkeit des Chamäleon -Roboters wurde erfolgreich bestätigt. (Abbildung 5) Der Mechanismus reagiert zuverlässig auf erkannte Farben, und die Motoren führen die programmierten Bewegungen ohne Verzögerung aus. Die Farberkennung arbeitet stabil, und die Steuerung des Förderbands sowie des Schwanzmechanismus verläuft synchron. Dies zeigt, dass der entwickelte Roboter sich für eine Anwendung in verschiedenen Umgebungen eignet und eine präzise Interaktion mit seiner Umgebung ermöglicht. (Abbildung 7) Das Ziel des Projekts wurde erreicht.

Literaturverzeichnis

[1] BATONA: *Kuriose Fakten über Chamäleons*. http://batona.net/90797-lyubopytnye-fakty-o-hameleonah-21-foto.html. Version: 2019. – Accessed: 2025-03-17

[2] AMAZON.DE: *LEGO MINDSTORMS Education EV3 Servomotor groß*. https://www.amazon.de/EDUCATION-MINDSTORMS%C2%AE-Education-Gro%C3%9Fer-EV3-Servomotor/dp/B00E1QDP4W. Version: 2025. – Accessed: 2025-03-17

[3] BOTLAND.DE: *Lego Mindstorms NXT / EV3 - Farbsensor - Lego*. https://botland.de/eingestellte-produkte/4697-lego-mindstorms-nxt-ev3-farbsensor-lego-9694.html. Version: 2025. – Accessed: 2025-03-17

Robo-Chamäleon

Viktor Rovenskykh, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Während des Projektseminars Elektrotechnik/Informationstechnik 2025 an der Otto-von-Guericke-Universität Magdeburg wurde der Robo-chamaeleon entwickelt, der in der Lage ist, Farben zu erkennen und nachzuahmen. Dieser Roboter wurde mit dem LEGO-Mindstorms-Baukasten konstruiert und nutzt zusätzliche Motoren zur Bewegung mechanischer Teile sowie einen Farbsensor zur Farberkennung.**

Die Programmierung des Roboters erfolgte mit MATLAB [1], das eine Vielzahl integrierter nützlicher Funktionen bietet. Dies ermöglichte es uns, die Programmieraufgabe zu vereinfachen und Zeit bei der Entwicklung und Konstruktion des Roboters zu sparen.

***Schlagwörter*—LEGO Mindstorms, Roboter, Chamäleon, MATLAB, Farberkennung.**

I. Einleitung

In der Welt, in der wir leben, spielt die Robotik eine bedeutende Rolle. Sie erstreckt sich über fast alle Bereiche des menschlichen Lebens – wir nutzen sie sowohl im Alltag als auch in verschiedenen Industriezweigen.

Das Ziel unseres Projekts ist es, das Funktionsprinzip der Robotik zu verstehen, insbesondere die Anwendung eines solchen Elements wie des Farbsensors.

Zur Umsetzung dieses Ziels wurde die Entwicklung des Robo-Chamaeleon gewählt – eines nicht allzu komplexen, aber dennoch sehr interessanten Roboters in Bezug auf sein Funktionsprinzip. In dieser Dokumentation werden alle Phasen der Planung, Konstruktion, Programmierung und Präsentation unseres Projekts detailliert beschrieben.

II. Vorbetrachtungen

Um die Aufgaben des Roboters vollständig zu erfüllen, muss verstanden werden, wie genau die Farberkennung durch den Scanner erfolgt, wie den erkannten Farben nacheifert wird und wie die Bewegung im Raum zur Erkennung verschiedener Farben umgesetzt wird.

A. Farbsensor und Motoren

Um die Farberkennung zu ermöglichen, musste ein Farbsensor (siehe Abbildung 1) angeschlossen werden, und es wurde untersucht, wie genau die Farberkennung abläuft. Es wurde analysiert, in welchem Bereich (zwischen Weiß und Schwarz) sich die vom Scanner ausgegebenen Werte befinden und welche optimale Entfernung zum Objekt erforderlich ist, um die Farben präzise zu identifizieren.

Zudem war es notwendig, sich mit den Motoren vertraut zu machen – insbesondere mit deren Leistung, Abmessungen und den Anschlüssen für die Kabel.

DOI: 10.24352/UB.OVGU-2025-007

Abbildung 1. Farbsensor

Nach dieser Untersuchung wurde eine erste Vorstellung von unserem Roboter machen und schließlich die Grundstruktur entwickeln, an der im weiteren Verlauf alle Komponenten befestigt wurden.

B. Technische Umsetzung und Softwarekomponenten

Die ursprüngliche Idee des Roboters war es, Farben zu erkennen und die gefundenen Farben nachzuahmen. Zur Umsetzung dieser Idee war der Einsatz eines Farbsensors und einer Leuchtdiode (im Folgenden LED) geplant. Allerdings musste die LED-Idee aus folgenden Gründen verworfen werden:

Diese Komponenten waren im bereitgestellten Bausatz nicht enthalten [2] – ein Problem, das sich zwar leicht lösen ließ. Innerhalb unseres zweiwöchigen Projektseminars wäre die Umsetzung dieser Idee sehr anspruchsvoll gewesen, insbesondere im Bereich der Programmierung. Da MATLAB für uns relativ neu war, entschieden wir uns, das Prinzip der „Mimikry“ zu vereinfachen und stattdessen ein drehbares Band mit verschiedenen Farbflächen zu verwenden. Dadurch konnten wir die Funktionalität unseres Roboters beibehalten und uns die Umsetzung erleichtern. Außerdem mussten wir auf einige dekorative Elemente verzichten, da die LEGO-Mindstorms-Bausätze dafür schlichtweg nicht ausgelegt sind. Letztendlich lag unser Hauptfokus darauf, die Funktionsweise verschiedener Komponenten kennenzulernen und nicht darauf, einen optisch ansprechenden Roboter zu entwerfen.

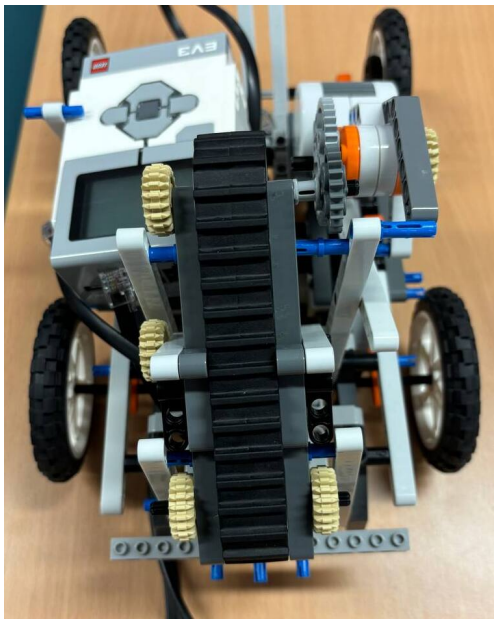


Abbildung 2. Erster Prototyp des Roboters

Abbildung 3. Zweiter Prototyp des Roboters

III. Konstruktion und Realisierung

A. Design und erster Prototyp

Ursprünglich war der Roboter nicht so groß geplant, wie er auf der finalen Präsentation gezeigt wurde. Im Gegenteil, er war ziemlich klein und einfach (siehe Abbildung 2), bestand aus 2 Motoren – einer für die Bewegung durch das Drehen der hinteren Räder und der andere für das Drehen des farbigen Bandes. Der Farbsensor war ebenfalls sofort eingebaut. Aber nach unserer ersten Präsentation, nach der vorgeschlagenen Idee, das farbige Band durch das Hinzufügen eines zweiten drehbaren Bandes zu erweitern, sie beide in einem Abstand von 3-4 cm anzuordnen und sie mit farbigen Streifen zu verbinden, verstanden wir, dass wir den Roboter erweitern mussten.

Zusätzlich fügten wir unserem Roboter einen Schwanz hinzu, der sich mit einem neuen dritten Motor drehte, um unsere Idee ein wenig abwechslungsreicher zu gestalten.

B. Zweiter und dritter (finaler) Prototyp

Nach mehreren Tagen Arbeit begann sich unser zweiter Prototyp des Roboters (siehe Abbildung 3) stark vom ersten zu unterscheiden. Die Größe seiner Basis vergrößerte sich um etwa 25 % in der Breite und um 50 % in der Länge, da wir einfach keinen Platz mehr für die neuen Teile hatten, insbesondere für die neue, verbesserte Version des farbigen Bandes im oberen Teil und für den zusätzlichen dritten Motor im unteren Teil.

Nach dem Hinzufügen neuer Komponenten begann unser Roboter aufgrund des zusätzlichen Gewichts stark zu sinken, und die Räder weigerten sich praktisch, sich zu drehen, was uns zwang, die Fahrwerksstruktur erheblich umzubauen und zu verstärken. Zudem fügten wir dem Roboter einen äußeren Körper für zusätzliche Stabilität hinzu.(In dieser Version des Roboters gab es noch keinen Schwanz.)

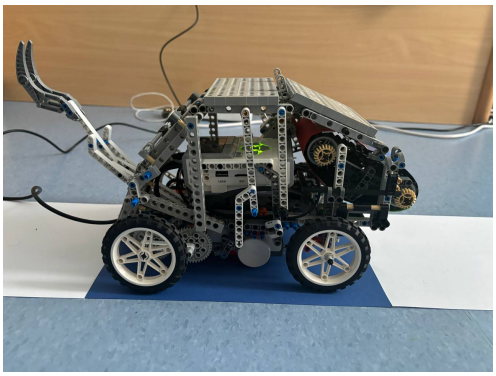


Abbildung 4. Dritter Prototyp des Roboters

In der finalen dritten Version des Roboters (siehe Abbildung 4) gab es keine besonderen Änderungen, aber wir verstärkten die Verbindungen der mechanischen Teile des Roboters, wie zum Beispiel der Zahnräder, damit die Drehbewegungen des Bandes und der hinteren Räder reibungslos und ohne Probleme ablaufen. Wir fügten ihm einen Schwanz hinzu und schlossen ihn an den dritten Motor an, nahmen einige visuelle Änderungen am Gehäuse vor und fügten zusätzliche Halterungen für das Hauptmodul unseres Roboters – den EV3-Controller – hinzu.

C. Programmierung

Da die Konstruktion und der Zusammenbau unseres Roboters ziemlich langwierig und kompliziert war, entschied sich unser Team, einen Teil der Programmierung zu vereinfachen, um rechtzeitig den Zeitrahmen unseres Projektseminars einzuhalten und gleichzeitig die von uns vollständig geplante Funktionalität zu bewahren (siehe Abbildung 5).

Unser Roboter sollte sich geradeaus bewegen, bis er eine bestimmte Farbe (in unserem Fall rot) findet, woraufhin er

anhielt, das farbige Band bis zur Position der roten Farbe drehte und den Schwanz bei einer bestimmten Farbe (in unserem Fall blau) bewegte. Nach der Durchführung dieser Aktion setzte der Roboter seine Bewegung fort, bis er die nächste Farbe fand, und der Zyklus wiederholte sich. Solche Bewegungen führten wir dreimal aus, nach denen der Roboter rückwärts zu seiner Ausgangsposition zurückkehrte.

Das geschriebene Programm haben wir auf dem EV3-Controller ausgeführt, der über eine USB-Verbindung mit dem Computer verbunden war.

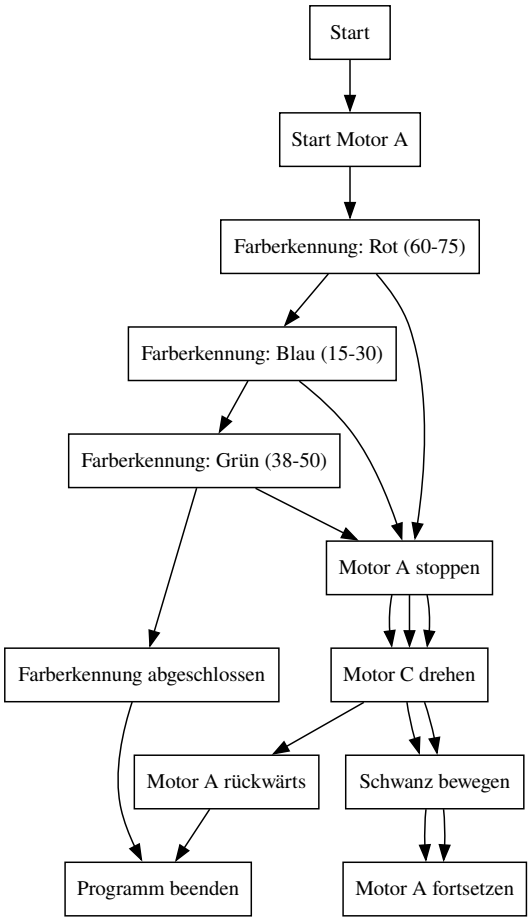


Abbildung 5. Blockdiagramm des Roboters

IV. Ergebnisdiskussion

Nach dem endgültigen Zusammenbau, der Überprüfung aller Komponenten auf ihre korrekte Platzierung und der Überprüfung des Codes wurde der abschließende Test des Roboters durchgeführt, der erfolgreich verlief. Er erkannte alle 3 Farben korrekt, drehte das farbige Band einwandfrei, bewegte den Schwanz ohne Probleme, fuhr geradeaus (möglicherweise mit einer kleinen Abweichung von 1° bis 2°) und kehrte zur Ausgangsposition zurück.

V. Zusammenfassung und Fazit

Nach Ablauf der zweiwöchigen Projektseminar-Periode waren wir mit unserem Ergebnis sehr zufrieden. Wir haben alles erreicht, was wir zu Beginn geplant hatten, und vor allem erfolgreich.

Obwohl wir während der Entwicklung auf Schwierigkeiten stießen, wie etwa einem Mangel an Bauteilen und der Notwendigkeit, den Roboter neu zusammenzubauen, hat uns das nicht davon abgehalten, uns intensiv mit MATLAB und einer Vielzahl von mechanischen und elektrischen Komponenten auseinanderzusetzen

Literaturverzeichnis

[1] Handbuch : *LEGO MINDSTORMS EV3 mit Matlab* https://de.mathworks.com/help/matlab/legomindstormsev3io.html Version: Februar 2025
[2] WIKIPEDIA: *Lego Mindstorms EV3* https://de.wikipedia.org/wiki/Lego_Mindstorms_EV3 Version: Februar 2025

Einparkroboter

Niklas Alsleben, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Das vorliegende Paper beschreibt die Entwicklung eines autonomen Einparkroboters unter Verwendung von LEGO Mindstorms und MATLAB im Rahmen des LEGO Projektpraktikums 2025 an der Otto-von-Guericke-Universität Magdeburg. Der Roboter ist in der Lage, mittels Ultraschallsensoren eine geeignete Parklücke zu erkennen und autonom rechts rückwärts parallel einzuparken. Der Fokus des Papers liegt auf der mechanischen Konstruktion, der Sensorik sowie der softwareseitigen Umsetzung. Herausforderungen und Optimierungsansätze werden ebenfalls diskutiert.**

***Schlagwörter*—Autonomes Parken, LEGO Mindstorms, Ultraschallsensorik, MATLAB, Robotik**

I. Einleitung

AUTONOME Fahrerassistenzsysteme sind ein zentrales Forschungsfeld in der Automobilindustrie. Insbesondere die Entwicklung von Systemen zur automatisierten Durchführung von Einparkmanövern erfährt aufgrund der zunehmenden Fahrzeugautomatisierung sowie der Verdichtung innerstädtischer Parkräume große Aufmerksamkeit. Entsprechende Forschungsprojekte, wie beispielsweise „SynCoPark“ der Technischen Universität Braunschweig, befassen sich mit der Implementierung und Optimierung intelligenter Parksysteme [1]. Im Rahmen des LEGO-Projektpraktikums 2025 an der Otto-von-Guericke-Universität Magdeburg wurde ein autonomer Einparkroboter entwickelt, um die grundlegenden Prinzipien automatisierter Parkvorgänge im Modellmaßstab umzusetzen.

Die gewählte Entwicklungsplattform, LEGO Mindstorms EV3, bietet hierfür eine geeignete Kombination aus Mechanik, sensorischer Erweiterbarkeit und einer kompatiblen Programmierschnittstelle in MATLAB. Dies ermöglicht eine realistische Nachbildung eines echten Personenkraftfahrzeugs. Zielsetzung des Projekts war die Entwicklung eines Roboters, der mithilfe von Ultraschallsensorik eine geeignete Parklücke identifiziert und anschließend autonom rechts rückwärts parallel einparkt. Die Herausforderung bestand insbesondere in der zuverlässigen Objekterkennung, der stabilen Fahrdynamik sowie der präzisen Steuerung von Lenk- und Antriebssystem. Darüber hinaus wurden Parallelen zu realen Fahrzeugsystemen angestrebt, beispielsweise durch das Einbeziehen einer akustischen Einparkhilfe zur Abstandsindikation.

Das Projekt verbindet die praktischen Erfahrungen aus vorangegangenen Tätigkeiten im Kraftfahrzeugbereich mit den im Studium der Elektromobilität erworbenen Fachkenntnissen. Die Umsetzung im Modellmaßstab erlaubt dabei die exemplarische Untersuchung typischer Herausforderungen, die sich auch bei der Realisierung automatisierter Parksysteme in Serienfahrzeugen stellen. Das vorliegende Paper dokumentiert die technischen Grundlagen, den Konstruktions- und Entwicklungsprozess sowie die erarbeiteten Lösungsansätze. Abschließend werden die erzielten Ergebnisse kritisch reflektiert und mögliche Optimierungsmaßnahmen für zukünftige Weiterentwicklungen aufgezeigt

DOI: 10.24352/UB.OVGU-2025-008

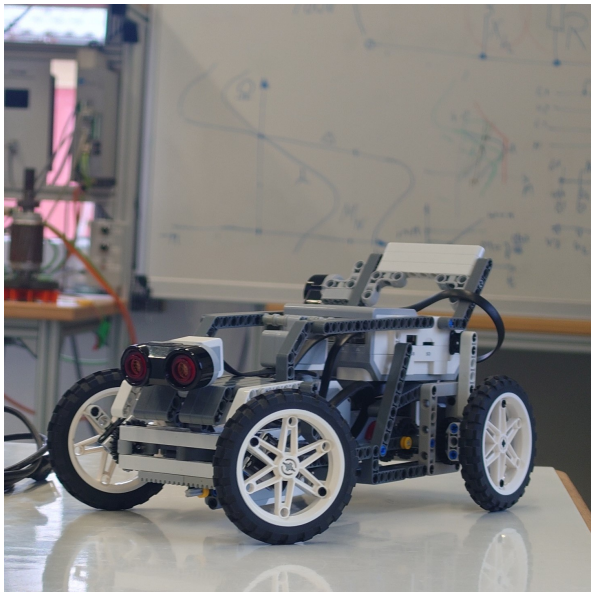


Abbildung 1. Finaler Einparkroboter

II. Vorbetrachtungen

A. Anforderungen an den Einparkroboter

Die grundlegenden Anforderungen an den Roboter bestanden in der Fähigkeit, eine Parklücke autonom zu erkennen, sich präzise in diese einzuordnen und dabei Kollisionen mit Hindernissen zu vermeiden. Das System sollte eine stabile Fahrweise und wiederholbare Ergebnisse liefern, während das Design mithilfe von Elementen wie Motorhaube, Spoiler und Abgasrohren an ein sportliches Auto erinnern sollte. Eine Einparkhilfe sollte ebenfalls verbaut werden, die durch erhöhte Tonfrequenz bei zunehmender Nähe zum rückwärtigen Hindernis ebenfalls Parallelen zum realen Personenkraftfahrzeug aufweist, siehe Abbildung 1.

B. Technologische Grundlagen

LEGO Mindstorms (inklusive des EV3 Controllers) wurde als Basis gewählt, da es eine modulare Bauweise sowie eine intuitive Programmierung mit MATLAB ermöglicht. Drei Ultraschallsensoren (vorne, hinten und rechts) wurden zur Umgebungserkennung eingesetzt, während zwei Motoren für Antrieb und Lenkung verantwortlich waren. Ein Ultraschallsensor arbeitet nach folgendem Prinzip: Er sendet

Ultraschallwellen aus, die von einem Gegenstand reflektiert und zurückgeworfen werden. Im EV3 Ultraschallsensor sind Sender und Empfänger in einem Bauteil enthalten. Somit kann der EV3 Controller den Abstand von Gegenständen oder Hindernissen über das Echo der Schallwellen auswerten. Die Berechnung der Entfernung erfolgt im EV3 Controller, er wertet aus wie lange die Schallwellen brauchen, bis sie wieder am Empfänger angekommen sind. Bei einer Schallgeschwindigkeit von 344 Metern pro Sekunde lässt sich aus der Zeit, die vom Sendevorgang bis zum Empfang vergeht, die Entfernung berechnen. Die Entfernung kann über den Bildschirm des EV3 Controllers in Centimetern ausgegeben werden [2].

III. Umsetzung

A. Mechanische Konstruktion

Der Roboter basiert auf einem stabilen Grundgerüst mit Heckantrieb. Als Antrieb dient der EV3 Large Motor, während die Lenkung über den Medium Motor realisiert wird. Ein Differential wurde integriert, um Drehzahlausgleich zwischen den Rädern bei Kurvenfahrten zu ermöglichen, siehe Abbildung 2. Ursprünglich wurde eine Kardanwellenlenkung getestet, die sich jedoch aufgrund zu hohen Spiels in den Kardangelenken als unpräzise erwies. Außerdem fing die lange Kreuzstange bei großen Kräften an, zu tordieren. Aus diesem Grund wurde die Lenkung überarbeitet und der Medium Motor unter dem Fahrzeug und so nah wie möglich an der Lenkverzahnung montiert, was in Abbildung 3 zu erkennen ist. Infolgedessen mussten zur Garantie von ausreichend Bodenfreiheit die größtmöglichen Räder verbaut werden. Die Modifikationen an der Lenkung und die Montage des EV3 Bricks hatten weiterhin die notwendige Verstärkung der Achsen zur Folge, die präzise Fahrmanöver trotz hohen Fahrzeuggewichts ermöglichte.

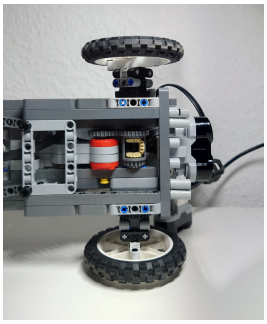


Abbildung 2. Differential

B. Sensorik

Der Roboter ist mit drei Ultraschallsensoren an folgenden Positionen ausgestattet: vorne, hinten und rechts, siehe Abbildung 4. Der rechte Sensor dient der Erkennung von Parklücken, während die vorderen und hinteren Sensoren für die Kollisionsvermeidung genutzt werden. Eine alternative Idee zur Spurhaltung bestand in der Verwendung von zwei seitlichen Ultraschallsensoren, die für permanent ausgeglichene Abstandswerte sorgen und die Lenkung entsprechend ansteuern, sollten die Werte nicht ausgeglichen sein. Auch ein Gyrosensor könnte

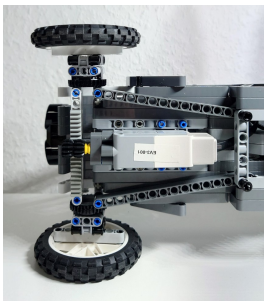


Abbildung 3. überarbeitete Lenkung

in Betracht gezogen werden, der das Fahrzeug auf gerader Spur hält und bei Abweichungen die Lenkung entsprechend gegensteuert. Diese alternativen Möglichkeiten waren jedoch im vorgegebenen Zeitrahmen nicht umsetzbar.

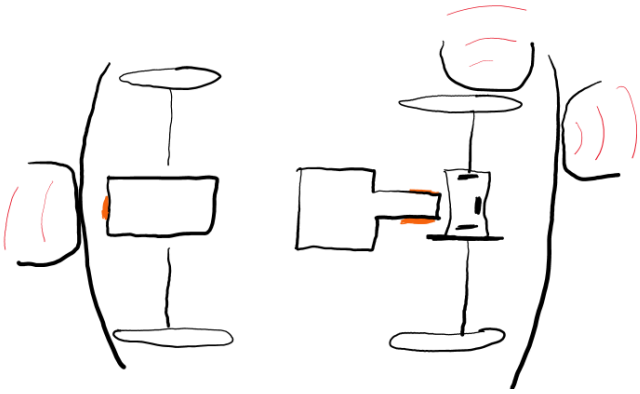


Abbildung 4. Schematischer Aufbau Sensorik

C. Software und Steuerung

Die Steuerung und Programmierung erfolgte mit MATLAB. Grundlegende Hinweise zur MATLAB-Programmierung des EV3 Controllers wurden aus [3] bezogen. Geradeaus- und Rückwärtsbewegungen wurden über eine Geschwindigkeits- und Zeitsteuerung realisiert. Für die Lenkung wurden Grenzwerte definiert, und der Brake-Modus `brake` des EV3-Systems sicherte einen stabilen Lenkwinkel. Die Parklückenerkennung wurde mithilfe des rechten Ultraschallsensors realisiert, der den Abstand zu parkenden Autos misst und so erkennt, wann eine passende Lücke vorliegt. Ist dieser Vorgang abgeschlossen und eine Parklücke gefunden, wird die Parksequenz eingeleitet, welche durch den vorderen und hinteren Sensor unterstützt wird, siehe Abbildung 5.

IV. Ergebnisdiskussion

Die ursprüngliche Idee eines autonom einparkenden Roboters konnte im Rahmen des Projektpraktikums umgesetzt werden. Das finale Endergebnis sowie einige Meilensteine während der Produktreifung können über den Instagram Account `mind.mobility25` eingesehen werden. Während des Entwicklungsprozesses traten verschiedene Herausforderungen auf. Eine geringe Lenkpräzision führte zur Überarbeitung der

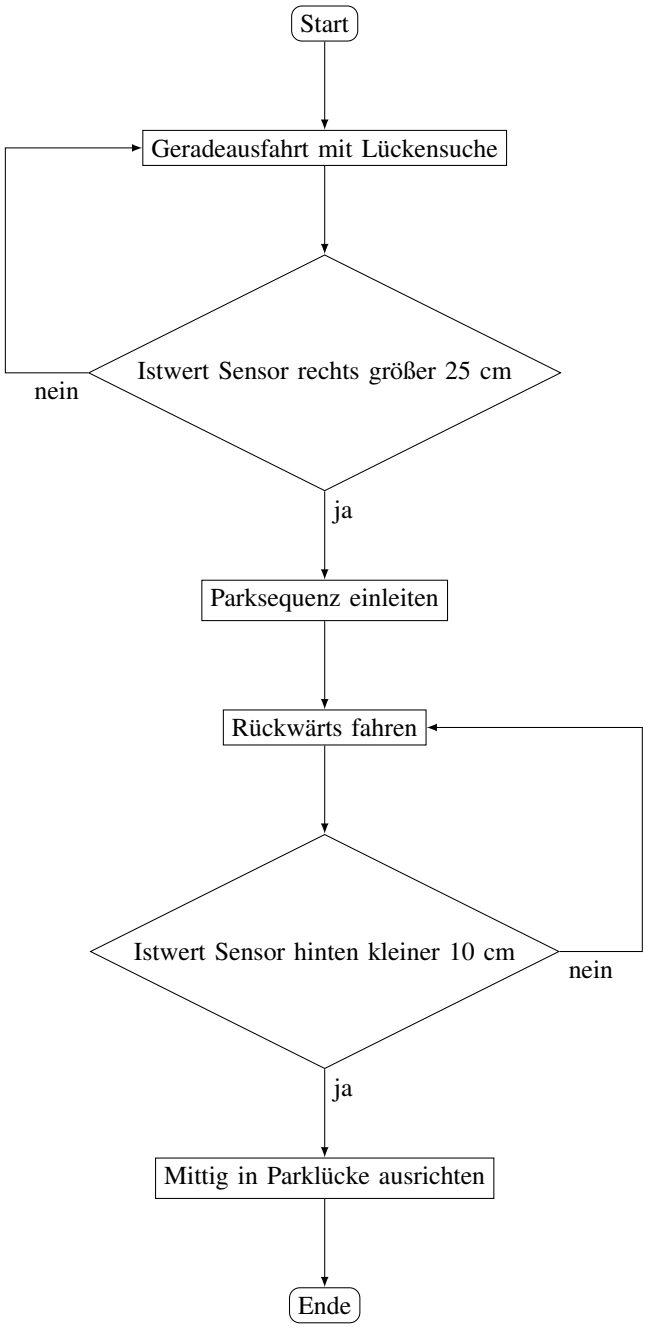


Abbildung 5. Programmablaufplan

Vorder- und Hinterachse. Die Aufhängung wurde angepasst, um Spiel zu reduzieren und das hohe Gewicht des Fahrzeugs tragen zu können, wofür das Programm entsprechend modifiziert werden musste. Zudem beeinflusste die Bodenreibung die Lenkgenauigkeit. Ein geplanter Gyrosensor hätte eine genaue Geradeausfahrt ermöglicht da dieser mit entsprechender Programmierung dafür sorgen würde, dass das Fahrzeug sobald es von seinem Weg abkommt, immer wieder zur Nulllinie zurückkorrigieren würde und somit einen guten Geradeauslauf garantiert hätte. Um trotzdem möglichst gleichbleibende Ergebnisse in den Fahrversuchen zu realisieren, wurde als Untergrund die gummierte Rückseite eines Teppichs gewählt, siehe Abbildung 6. Dieser wurde benötigt, da die schmalen Reifen auf glatten Oberflächen wie der Fußbodenbeschichtung im Labor oder Laminat in anderen Räumlichkeiten dazu neigten, schnell an Traktion zu verlieren. Das hatte zur Folge, dass das Fahrzeug bei eingeschlagener Lenkung im Rückwärtsgang nur noch hinterhergezogen wurde, aber kein Lenken mehr stattfand. Die gummierte Rückseite des Teppichs bot dem Fahrzeug jedoch immer ausreichend Traktion um gute Testergebnisse zu liefern.

V. Zusammenfassung und Fazit

Der erfolgreich entwickelte Einparkroboter zeigt das Potenzial autonomer Parksysteme im Modellmaßstab. Die Kombination aus Mechanik, Sensorik und intelligenter Programmierung ermöglichte ein weitgehend autonomes Einparkverhalten. Künftige Verbesserungen könnten durch präzisere Sensorik, Reifen mit mehr Traktion und eine stabilere Lenkung sowie den Einsatz zusätzlicher Regelungstechniken wie eines Gyrosensors erreicht werden. Ein weiterer Schritt wäre die Integration von KI-gestützten Algorithmen zur adaptiven Optimierung der Einparkstrategie.

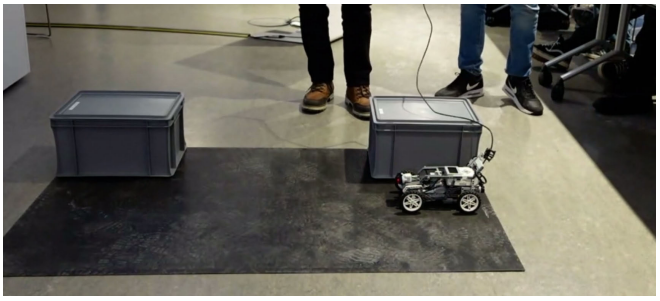


Abbildung 6. Versuchsaufbau

Literaturverzeichnis

[1] NFF Niedersächsisches Forschungszentrum Fahrzeugtechnik : *Intelligentes Parken der Zukunft*. https://www.tu-braunschweig.de/ism/newsdetail/intelligentes-parken-der-zukunft. Version: Juli 2021

[2] *Praxiswissen Ultraschallsensoren Teil 1 Technologie und Funktion im Überblick*. https://www.pepperl-fuchs.com/global/de/24907.htm#:~:text=Der%20Ultraschallsensor%20misst%20den%20zeitlichen%20Abstand%20zwischen%20dem,die%20Schallgeschwindigkeit%20in%20der%20Luft%20rund%20344%20m%2Fs. Version: Januar 2014

[3] RWTH Aachen: *EV3 Toolbox Documentation*. https://git.rwth-aachen.de/mindstorms/ev3-toolbox-matlab/-/blob/master/MindstormsEV3Toolbox.pdf?ref_type=heads. Version: 2017

Der Einparkroboter

Dennis Reß, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Es wurde ein autonom einparkendes Fahrzeug mit LEGO Mindstorms EV3 entwickelt. Während der Umsetzung zeigten sich mehrere Herausforderungen: Die strukturelle Toleranz der LEGO-Bausteine erschwerte die Geradeausfahrt, sodass eine Stabilisierung durch einen Gyrosensor erforderlich wäre. Zudem führte eine instabile Bluetooth-Verbindung zwischen MATLAB und dem EV3 zu Kommunikationsabbrüchen. Trotz dieser Einschränkungen erkannte das System die Parklücken und führte die Einparkvorgänge autonom aus. Verbesserungsmöglichkeiten umfassen die Integration einer Kamera zur präziseren Erkennung und eine stabilere drahtlose Kommunikation.**

***Schlagwörter*—Autonomes Fahren, Einparkassistent, Fahrzeugsteuerung, LEGO Mindstorms, Robotik**

I. Einleitung

Seit der Erfindung des Benz Patent-Motorwagens vor über 200 Jahren ist das Automobil von stetigem Wandel und technischem Fortschritt geprägt [1]. Besonders in den letzten Jahren rückte die autonome Mobilität verstärkt in den Fokus von Forschung und Entwicklung.

Auch in diesem Jahr entwickelten Studierende im Projektseminar ein teilautonomes Fahrzeug, um sich mit den Grundlagen einer solchen Programmierung vertraut zu machen. Dabei standen sowohl die Mechanik als auch die Softwareentwicklung im Mittelpunkt des Projekts. Neben der Implementierung grundlegender Fahrfunktionen wurde insbesondere die sensorbasierte Umgebungswahrnehmung untersucht.

Die folgenden Abschnitte behandeln die Konstruktion des Fahrzeugs, die Programmierung sowie den Versuchsaufbau, bevor die Ergebnisse diskutiert und mögliche Verbesserungen aufgezeigt werden.

II. Vorbetrachtungen

Das parallele Einparken folgt einer strukturierten Vorgehensweise, die sich in mehrere Schritte unterteilen lässt.

Zunächst wird eine geeignete Parklücke ausgewählt, die ausreichend Platz für das Fahrzeug bietet. Anschließend richtet man das eigene Fahrzeug parallel zum parkenden Fahrzeug aus, wobei etwa ein halber Meter Abstand eingehalten werden sollte. Im nächsten Schritt wird das Lenkrad vollständig nach rechts eingeschlagen, während das Fahrzeug langsam rückwärts fährt, bis das hintere Fahrzeug im Sichtfeld erscheint. Danach wird das Lenkrad geradegestellt und das Fahrzeug weiter zurückgesetzt, bis die eigene Fahrzeugfront die hintere Stoßstange des vorderen Autos passiert hat. Nachdem das Lenkrad vollständig nach links eingeschlagen wurde, um in die Parklücke einzufahren, wird das Fahrzeug zentriert und das Lenkrad in Ausgangslage gebracht [2].

DOI: 10.24352/UB.OVGU-2025-009

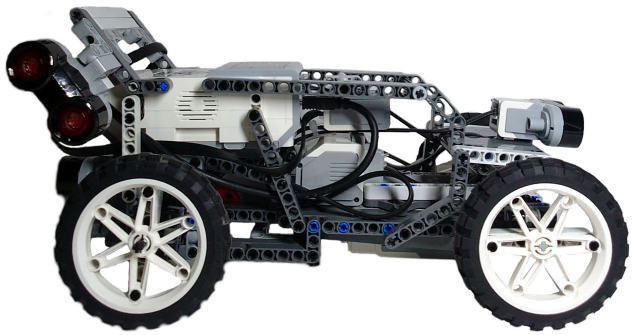


Abbildung 1. Einparkroboter

III. Umsetzung

In diesem Projekt wurde zunächst das Fahrzeug aufgebaut, wobei grundlegende mechanische und elektronische Komponenten integriert wurden. Anschließend wurde die Steuerung programmiert, sodass das Projektziel erreicht werden konnte.

A. Konstruktion

Das Design des Fahrzeugs (siehe Abb. 1) orientiert sich an klassischen Personenkraftwagen statt an typischen Robotermodellen, wodurch die Platzierung der Komponenten anspruchsvoller wurde.

Der Antrieb erfolgt über einen großen Motor, der seine Kraft über ein Differential (siehe Abb. 2) auf die Hinterräder überträgt. So werden unterschiedliche Raddrehzahlen, wie sie bei Kurvenfahrten auftreten, ausgeglichen und ein Durchdrehen der Hinterräder verhindert.

Unterhalb der Vorderachse ist ein mittlerer Motor zur elektrischen Servolenkung montiert (siehe Abb. 4). Die Kraftübertragung erfolgt über zwei Zahnräder, die eine größere Auflagefläche auf der Zahnstange der Lenkung gewährleisten. Aufgrund der Position des Servomotors war es erforderlich, größere Räder zu verbauen, um eine ausreichende Bodenfreiheit sicherzustellen.

In der hinteren Hälfte des Fahrzeugs ist der EV3-Controller fest verbaut. Dieser verfügt über jeweils vier Anschlüsse für Sensoren und Motoren. Die Sensoranschlüsse sind mit einem Gyrosensor sowie drei Ultraschallsensoren belegt, die vorne, hinten und rechts montiert sind. Während die Ultraschallsensoren zur Messung der Abstände des Autos zur Umgebung dienen, ist der Gyrosensor zwar angeschlossen, wurde aber noch nicht in die Funktionalität integriert.

Für die Verbindung zwischen dem EV3 und MATLAB ist zunächst ein Kabel erforderlich. Damit es die Räder nicht

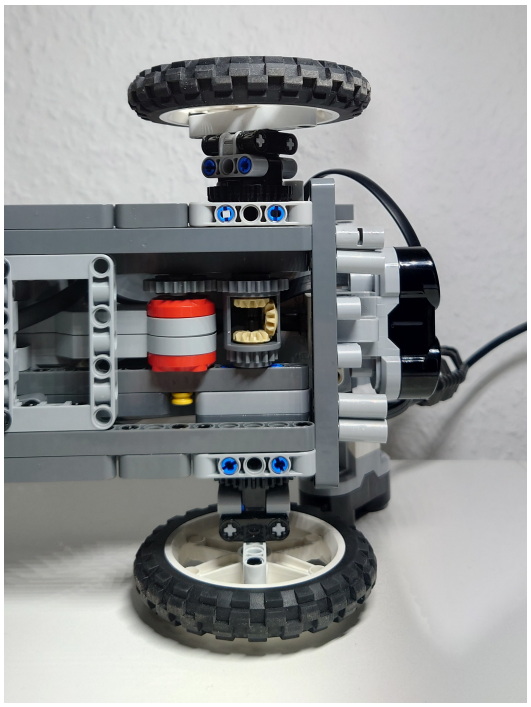


Abbildung 2. Differential der Hinterachse

blockiert, wurde es in einem Käfig oberhalb des Spoilers befestigt.

B. Programmierung

Für das Vor- und Rückwärtsfahren wurden jeweils Funktionen implementiert, die mit einer einzugebenden Geschwindigkeit und Dauer arbeiten. Um ein ungewolltes Ausrollen zu vermeiden, wurde der „brakeMode“ auf „brake“ gesetzt, sodass der Motor am Ende der Fahrt seine Position hält. Diese Einstellung wurde auch bei der Lenkungssteuerung berücksichtigt, um einen eingestellten Lenkwinkel beizubehalten.
Im Gegensatz zu den Antriebsfunktionen arbeiten die beiden Funktionen zur Lenkung nach links und rechts mit einer einstellbaren Motorleistung sowie einer Winkeleingabe. Dies ermöglicht eine Anpassung an verschiedene Fahrbahnbedingungen. Allerdings wirkt sich die Motorbremse sowohl auf die Akkulaufzeit als auch auf die Geräuschentwicklung aus, was als Nachteil in Kauf genommen werden musste.
Für die Ultraschallsensoren an der Vorder- und Rückseite wurde eine Funktion geschrieben, die bei Annäherung an ein Hindernis mit zunehmendem Abstand ein schneller werdendes akustisches Signal ausgibt und den Motor stoppt, bevor es zu einer Kollision kommen könnte.
Erfasst der Ultraschallsensor rechtsseitig einen Wert über $25\,\mathrm{cm}$, wird ein boolescher Wert gesetzt, der das Vorhandensein einer Parklücke speichert. Sobald der Sensor wieder einen Wert unter $25\,\mathrm{cm}$ misst, wird das Ende der Parklücke erkannt und das Fahrzeug veranlasst, anzuhalten.
Auf Basis dieser Funktionen wurde die Programmierung der Parksequenz abgeleitet, wie sie in der Vorbetrachtung und in Abbildung 3 beschrieben wird.

C. Versuchsaufbau

Für die Durchführung von Tests war eine geeignete Testumgebung erforderlich. Diese bestand aus einem Teppich mit gummierter Oberfläche sowie zwei identischen Kisten. Da der Boden des Labors nicht genügend Reibung für die Fahrzeugreifen bot, wurde der Teppich als Fahrfläche gewählt. Zur Sicherstellung konsistenter Testergebnisse wurden die Positionen der Kisten markiert und für jede Versuchsdurchführung beibehalten. Die simulierte Parklücke hatte eine Länge von etwa dem Zweifachen der Fahrzeuglänge.

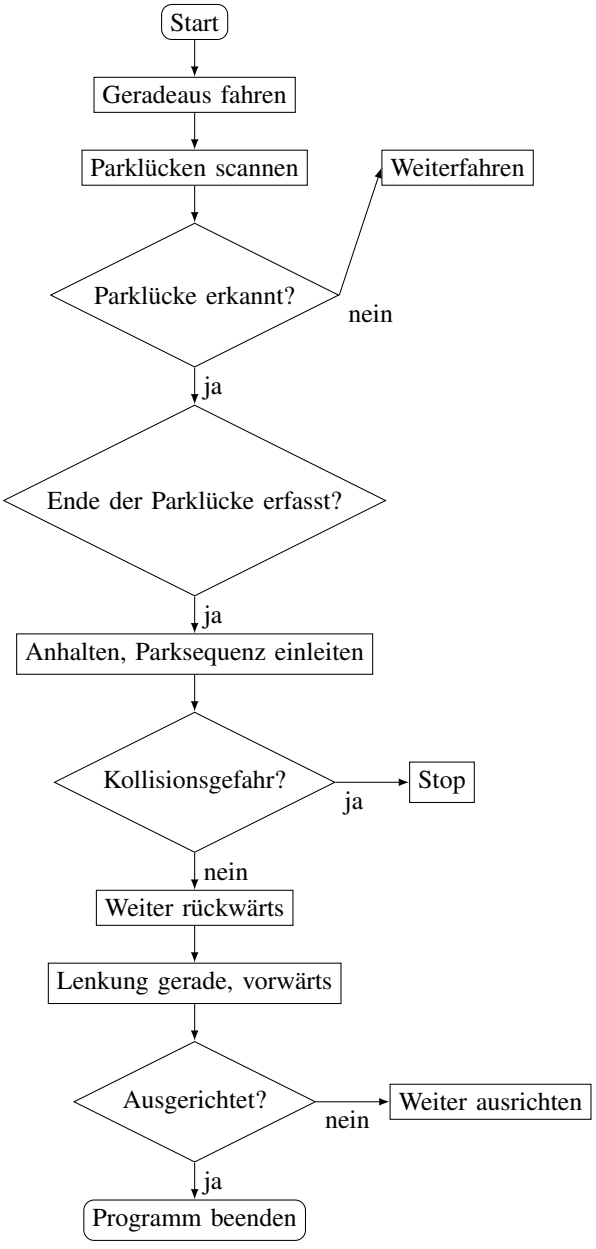


Abbildung 3. Programmablaufplan für das parallele Einparken.

IV. Ergebnisdiskussion / Verbesserungsvorschläge

In der Praxis zeigte sich, dass das Fahrzeug auf längeren Distanzen keine exakte Geradeausfahrt durchführen konnte,

sondern eine leichte Kurvenfahrt vollzog. Trotz der Motorbremse in der Lenkung war eine präzise Geradeausbewegung nicht gewährleistet. Dies ist nicht nur auf die anfängliche Positionierung des Fahrzeugs zurückzuführen, sondern vor allem auf die strukturelle Toleranz der LEGO-Bausteine. Mehrere Anpassungen führten nicht zum gewünschten Ergebnis.

Ein Lösungsansatz bestünde in der Nutzung des im Fahrzeug integrierten Gyrosensors. Dieser könnte zu Beginn der Fahrt einen Referenzwert erfassen und bei Abweichungen entsprechende Lenkbefehle einleiten, um die Kursstabilität zu verbessern. Ein erster Prototyp des Programms wurde bereits entwickelt.

Geplant war die Einbindung des Gyrosensors in zwei Funktionen: Erstens zur Erkennung einer Parklücke während der Geradeausfahrt und zweitens zur präziseren Steuerung während der Einparksequenz mithilfe von Winkeln. Schon bei der ersten Funktion traten jedoch Probleme auf. Das zyklische Abfragen der Sensorwerte dauerte zu lange, sodass das Fahrzeug meist erst eine Fahrzeuglänge hinter dem Ende der Parklücke zum Stillstand kam. Dieses Problem hätte sich theoretisch lösen lassen, indem das Fahrzeug eine bestimmte Strecke zurückgesetzt hätte. In der Realität fährt man jedoch nicht so weit an einer Parklücke vorbei. Abgesehen von der Programmierung fiel zudem auf, dass der Sensorwert in einigen Fällen kontinuierlich anstieg oder sank, obwohl keine Bewegung stattfand. Eine erste Vermutung war, dass der Sensor beim Einschalten des EV3-Controllers in absoluter Ruhe sein musste. Die Einhaltung dieser Bedingung erwies sich jedoch als nicht vollständig zuverlässig. Aufgrund dieser Problematik und des zusätzlichen Zeitmangels wurde die Idee vorerst verworfen.

Eine alternative Lösung stellt der Einbau eines vierten Ultraschallsensors auf der rechten Fahrzeugseite dar. Durch die Messung der seitlichen Abstände mit zwei Sensoren könnten Korrekturen vorgenommen und eine stabilere Geradeausfahrt erreicht werden.

Während der Präsentation zeigte der Einparkroboter neben der unpräzisen Geradeausfahrt auch eine gewisse Unzuverlässigkeit in der Programmsteuerung. Eine mögliche Ursache hierfür sind die Einschränkungen der verbauten LEGO-Ultraschallsensoren. Diese liefern bei einem Abstand von weniger als $5\,\mathrm{cm}$ einen Wert von 255 zurück, was fälschlicherweise signalisiert, dass sich kein Hindernis in der Nähe befindet. Dies führt in der aktuellen Implementierung dazu, dass der Roboter keine Parklücke mehr erkennt und stattdessen auf das Ende der vermeintlichen Lücke wartet. Eine mögliche Lösung wäre die Integration einer Kamera, um die Erkennung der Parklücke sowie umliegender Hindernisse zu verbessern. In Verbindung mit Bildverarbeitungsalgorithmen könnten nicht nur Objekte präziser identifiziert, sondern auch Bodenmarkierungen erkannt werden, wie sie auf vielen Parkplätzen vorkommen. Dadurch ließe sich das System zuverlässiger gestalten und besser an verschiedene Umgebungen anpassen.

Ein letzter praxisnaher Verbesserungsvorschlag wäre eine Bluetooth-Verbindung zwischen MATLAB und dem EV3. Zwar lässt sich eine Verbindung zwischen Laptop und EV3 herstellen, jedoch bricht sie nach etwa zehn Sekunden ständig ab. Dies könnte auf Unterschiede in den Bluetooth-Generationen der beiden Geräte zurückzuführen sein.

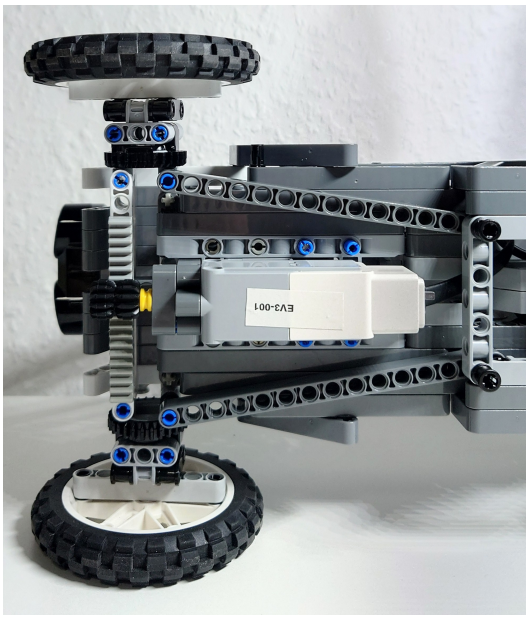


Abbildung 4. Aufbau der Lenkung

V. Zusammenfassung und Fazit

Das Ziel, einen funktionierenden Einparkroboter zu entwickeln, wurde erfolgreich erreicht. Für den Aufbau kamen zwei Motoren, drei Ultraschallsensoren und ein EV3-Controller zum Einsatz. Das zugrunde liegende Programm erkennt eine Parklücke und führt den Einparkvorgang durch, während es gleichzeitig Kollisionen vermeidet. Das Projekt hat gezeigt, dass ein autonomes Einparksystem auch mit einfachen Mitteln wie dem LEGO-Mindstorms Set realisierbar ist.
Dennoch gibt es mehrere Ansätze zur Verbesserung, auf die sich zukünftige Arbeiten konzentrieren könnten, um das System robuster und flexibler für verschiedene Umgebungen zu gestalten. Sollte das Projekt weiterentwickelt werden, sind insbesondere folgende Optimierungen empfehlenswert:

1) Erkennung von Parklücken sowohl auf der linken als auch auf der rechten Seite
2) Einsatz einer Kamera und Bildverarbeitung
3) Implementierung eines Spurhalteassistenten
4) Erhöhung der Zuverlässigkeit der Programmierung

Die bestehende Konstruktion bietet eine solide Basis für diese Weiterentwicklungen. Für Studierende höherer Semester könnte zudem die Implementierung komplexerer Algorithmen aus der Regelungstechnik eine sinnvolle Erweiterung darstellen, um das Einparksystem noch zuverlässiger zu gestalten – insbesondere unter erschwerten Bedingungen.

Literaturverzeichnis

[1] Welt der Wunder: *Die Entwicklung des Autos – von den Anfängen bis heute*. https://www.weltderwunder.de/die-entwicklung-des-autos-von-den-anfaengen-bis-heute/. – Zugriff am: 24. Feb. 2025

[2] Zed Driving School: *How To Parallel Park In 1 Minute*. https://www.youtube.com/watch?v=FjEmxM2PMv8. – Zugriff am: 24. Feb. 2025

Follow-Me Robot

Mingze Ma, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Ziel des Projektseminars Elektrotechnik/Informationstechnik (LEGO Mindstroms) 2025 war es, eine Objektverfolgungsfunktion mit Hilfe eines Roboters aus einem LEGO Baukasten zu realisieren. Die Steuerung des Roboters erfolgt über die Steuerungskomponente NXT, die Datenerfassung über angeschlossene Sensoren und die Programmierung über MATLAB.**

***Schlagwörter*—LEGO Mindstroms, MATLAB, Verfolgen, Abstandsensor, Projektseminar**

I. Einleitung

BEI selbstfolgenden Robotern handelt es sich um intelligente Geräte, die in der Logistik, im Lagerwesen, im Einzelhandel und im Haushalt weit verbreitet sind und durch autonome Aktionen und Objekterkennungstechnologien die Verfolgung von Personen oder Objekten in Echtzeit ermöglichen. Mit der rasanten Entwicklung der Automatisierungstechnik haben diese Roboter ein großes Potenzial zur Verbesserung der Effizienz und zur Verringerung des Personalaufwands. Auch durch die Integration mit anderen Technologien ergeben sich sehr viele Möglichkeiten.

Gleichzeitig ist diese Technologie aber nicht unerreichbar. LEGO ist ein berühmtes Spielzeug, aber im heutigen digitalen Zeitalter können einfache Roboter aus LEGO auch Funktionen ausführen, die schwierig klingen. In diesem Artikel wird zum Beispiel gezeigt, wie die Auto-Following-Technologie mit LEGO-Bauteilen und einfacher Programmierung realisiert werden kann. Dies ist eine relativ einfache Methode, die als Einführung genutzt werden kann, um Menschen zu inspirieren, wie diese Technologie realisiert werden kann und welche Möglichkeiten für die Zukunft bestehen.

II. Vorbetrachtungen

In diesem Abschnitt werden der Zweck des Projekts, die wichtigsten im Projekt verwendeten Komponenten und das für die Programmierung erforderliche MATLAB vorgestellt.

A. Zweck des Projekts

Das Ziel von Tracking-Robotern besteht in erster Linie darin, sich selbständig bewegen zu können. Zweitens müssen sie in der Lage sein, relevante Daten über das verfolgte Objekt zu erhalten, z. B. die Entfernung und die Orientierung. Schließlich werden durch Programm die erfassten Daten verarbeitet, um Befehle zur Verfolgung des Objekts zu generieren. Die dafür erforderliche Komponenten und Software wird im Folgenden beschrieben.

DOI: 10.24352/UB.OVGU-2025-010

B. Erforderliche Komponenten

1) Motor: Die in Abbildung 1 gezeigten Motoren werden im Projekt verwendet, um die Antriebsfunktion zu realisieren. Gleichzeitig ermöglicht der Geschwindigkeitsunterschied zwischen den beiden Motoren das Abbiegen des Roboters.

Abbildung 1: LEGO-Motor [1]

2) Ultraschallsensor: Um das zweite Ziel zu erreichen, nämlich den Abstand zwischen dem Roboter und dem zu messenden Objekt zu ermitteln, ist ein Ultraschallsensor erforderlich, wie in Abbildung 2 dargestellt.

Abbildung 2: LEGO-Ultraschallsensor [2]

C. MATLAB

Um die Komponenten zu steuern und die von den Sensoren erhaltenen Daten zu verarbeiten, ist es notwendig, MATLAB zur Programmierung zu verwenden. Zunächst ist es notwendig, die grundlegende Syntax der MATLAB-Programmiersprache, die Variablendeklarationen und die grundlegenden Funktionsoperationen zu beherrschen. Noch wichtiger ist, dass man Funktionen schreiben und aufrufen kann, damit ist es möglich, den Motor zu steuern und auch die vom Ultraschallsensor gelieferten Daten zu betrachten und zu verarbeiten. Auf dieser Grundlage wird das Programm geschrieben, um die genannten Ziele zu erreichen.

III. HAUPTTEIL

Im Hauptteil werden das Konzept und die Realisierung erläutert.

A. Aufbau

Wie in Abbildung 3 gezeigt, das ganze Projekt besteht aus einer NXT-Box, zwei Motoren, zwei Ultraschallsensor und einige LEGO Beuteile.

Aus einer NXT-Box wurde ein Rahmen gefertigt, der als Hauptkörper des Roboters diente. Zwei Motoren wurden auf beiden Seiten des vorderen Endes des Rahmens montiert, um als Energiequelle für die Fahrenfunktion des Roboters zu dienen. Die Abbiegenfunktion wird durch den Unterschied in der Drehgeschwindigkeit zwischen den beiden Rädern erreicht. Am hinteren Ende ist nur ein nicht motorisiertes Rad angebracht, um den Hauptrahmen zu stützen. Der Vorteil von nur einem Hinterrad ist, dass es dem Verhaltensmuster der Differenziallenkung besser entspricht und den Reibungswiderstand beim Abbiegen vermeidet.

Eine weitere Überlegung für den Hauptrahmen besteht darin, den Schwerpunkt des Roboters zu senken, indem die NXT-Box flach in der Mitte des Hauptrahmens platziert wird. Der unmittelbare Vorteil einer gleichmäßigen Verteilung des Hauptgewichts (d. h. der NXT-Box) besteht darin, dass die Stabilität der Struktur optimiert wird. Dies verhindert eine Überlastung der Komponenten, die den Hauptrahmen mit den Motoren verbinden, was eine präzise Roboterbewegung gewährleistet.

Zwei Ultraschallsensoren sind oberhalb der Vorderseite des Hauptrahmens angebracht, um den Abstand zum zu verfolgenden Objekt zu messen. Der Abstand zwischen den beiden Ultraschallsensoren sollte so groß wie möglich sein, um einen möglichst großen Messbereich zu erhalten. Je größer der Messbereich ist, desto genauer ist die Erkennung, und es werden auch Situationen vermieden, in denen das richtige Objekt nicht verfolgt werden kann.

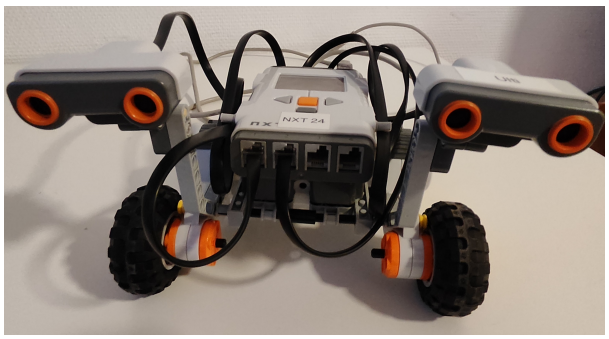


Abbildung 3: ursprünglicher Aufbau des Follow-Me Robot

B. Programmablaufplan

In Abbildung 4 wird ein Programmablaufplan gezeigt. Die Hauptlogik der Follow Me Roboters lässt sich zunächst nachvollziehen.

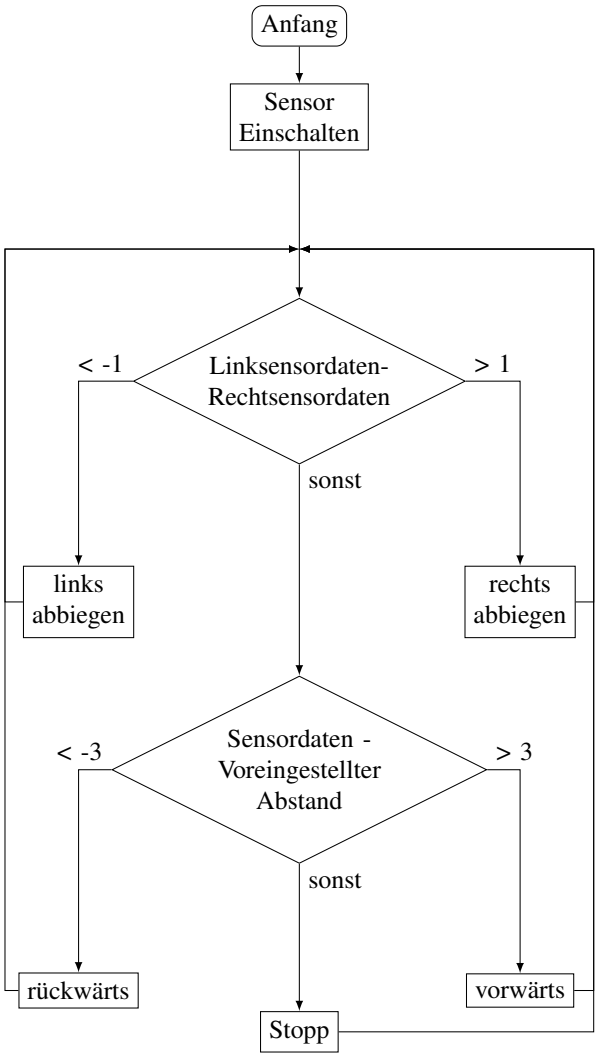


Abbildung 4: Programmablaufplan von Follow Me Robot

C. MATLAB-Programm

Mit Hilfe von Programmablaufplan lässt sich die Logik der Verfolgung von Robotern gut nachvollzogen werden. Als Nächstes muss der Roboter mit MATLAB programmiert werden, um ihn so zu steuern, dass er die gewünschte Funktion erfüllt.

1) Parameter: Zunächst müssen Variablen definiert werden, beispielsweise Zielentfernung (`targetDistance`), Entfernungstoleranz (`toleranz`), Abbiegenentfernung (`AbbiegenDistance`), Fahrgeschwindigkeit (`motorPower`), Abbiegengeschwindigkeit (`AbbiegenPower`) und vor allem die vom Ultraschallsensor in Echtzeit erfassten Entfernungsdaten((`currentDistance_1`) und (`currentDistance_2`)). Daraus kann der Abstandsunterschied (`Diff`) zwischen der linken und rechten Seite berechnet werden, um die Abbiegenanweisungen zu erhalten. Liegt die Differenz innerhalb der Lenkabstandstoleranz, werden die Echtzeit-Abstandsdaten mit dem Sollabstand verglichen und berechnet, um die Solldifferenz zu ermitteln und festzustellen, ob eine Abbiegenanweisung

erforderlich ist.

Alle Parameter müssen mehrmals getestet werden, um einen geeigneten Wert zu ermitteln. Außerdem werden für verschiedene verfolgende Objekte unterschiedliche Parameter benötigt. Zum Beispiel erfordert der Parameter für die Verfolgung einer Person eine größere Entfernungstoleranz als bei der Verfolgung eines flachen Objekts, da die Kleidung relativ stärker gekrümmt ist.

2) Abbiegenanweisung: Die Abbiegenanweisung wird auf die folgende Weise erzeugt und ausgeführt. Wenn der absolute Wert der Differenz zwischen den beiden Entfernungsdaten `(abs(Diff))` größer ist als Summe von Abbiegenentfernung `(AbbiegenDistance)`, wird das Abbiegenprogramm eingegeben. Wenn die Entfernungsdaten der linken Seite größer als die der rechten Seite ist, wird die Geschwindigkeit des linken Motors auf die Abbiegengeschwindigkeit `(AbbiegenPower)` und die der rechten Seite auf den negativen Wert der Abbiegengeschwindigkeit eingestellt. Der umgekehrte Fall trifft ebenfalls zu.

Ist der Abstandsunterschied `(Diff)` kleiner als der Abbiegenentfernung `(AbbiegenDistance)`, so ist folgendes Proframm anzuwenden.

3) Verfolgenanweisung: Die Verfolgenanweisung wird auf die folgende Weise erzeugt und ausgeführt. Wenn die Entfernungsdaten größer als die Summe von Zielentfernung `(targetDistance)` und Entfernungstoleranz `(toleranz)` ist, wird die Geschwindigkeit des beiden Motors auf die Fahrgeschwindigkeit `(motorPower)` eingestellt. Das Gegenteil setzt die Geschwindigkeit des beiden Motors auf einen negativen Wert der Fahrgeschwindigkeit **(motorPower)**.

Liegt die Zielentfernung innerhalb des Entfernungstoleranz `(toleranz)`, wird die Geschwindigkeit beider Motoren auf Null gesetzt, d. h. der Roboter wird zum Stillstand gebracht.

4) Befehlszykluszeit: Alle Befehle werden erteilt, um den Zyklus von Erfassung, Berechnung, Beurteilung und Ausführung von Befehlen wieder aufzunehmen, um eine kontinuierliche Folgefunktion zu erreichen. Nach dem Testen ist 0,5 Sekunden eine angepasste Befehlszykluszeit, die schneller reagieren kann und nicht zu viele Befehle gibt, für deren Ausführung der Roboter keine Zeit hat.

IV. Ergebnisdiskussion

A. Ergebnis

Damit hat der Follow Me Roboter sein Ziel erreicht, nämlich das Objekt zu verfolgen. Dies gilt insbesondere für flache Objekte, bei denen der Roboter in der Lage ist, Entfernungen schnell zu erkennen und Befehle zum Drehen und Folgen auszuführen. Auch das Verfolgen von Personen ist möglich, sofern die Fahrsgeschwindigkeit und die Entfernungstoleranz relativ groß gewählt werden.

B. Problemm - Beschränkung von Ultraschallsensor

1) Einschränkungen bei der Zielerkennung: Das Hauptproblem besteht darin, dass der Follow Me Roboter aufgrund der Beschränkungen der Ultraschallsensoren nicht in der Lage ist, genau zu erkennen, welchem Objekt er folgen soll. Er kann nur einem Objekt folgen, das sich direkt vor ihm befindet.

2) Detektionsbereich und dynamische Verfolgungsgrenzen: Das zweite Problem ist eine Frage der Perspektive. Der Abstand zwischen zwei Ultraschallsensoren ist der Bereich, den der Roboter korrekt erkennen kann. Das bedeutet, dass sich das verfolgte Objekt nicht zu schnell bewegen darf, sonst verliert der Roboter das Ziel. Auch wenn das Objekt unregelmäßig ist, z. B. wenn der Erkennungsbereich der Ultraschallsensoren genau durch die Fußseiten einer Person geht, kann das verfolgte Objekt nicht korrekt erkannt werden.

V. Zusammenfassung und Fazit

Insgesamt hat das Projekt seine erklärten Ziele erreicht. Gleichzeitig kann es als Einstiegspunkt für das Verständnis der Autofollowing-Technologie dienen. Gleichzeitig gibt es noch weitere Probleme zu lösen und weitere Möglichkeiten zu erforschen.

VI. Anhang

Der Quellcode ist im Anhang zu finden [3].

```
% Abbiegen Funkion
while true
% 1. Abstandswerte von beiden Sensoren lesen
    currentDistance_1 = GetUltrasonic(SENSOR_2);
    currentDistance_2 = GetUltrasonic(SENSOR_1);
    fprintf(['[Zustand] Linker Abstand: %.2f cm,
    Rechter Abstand:' ' %.2f cm\n'],
    currentDistance_1, currentDistance_2);

% 2. Entscheidung basierend auf den Abstandswerten
if abs(currentDistance_1 - currentDistance_2) >
       AbbiegenDistance

% 2.1 D_1 > D_2 nach rechts
if currentDistance_1 > currentDistance_2
% Drehe nach rechts
    motorB.Power = AbbiegenPower;
    motorA.Power = -AbbiegenPower;
    motorB.SendToNXT();
    motorA.SendToNXT();
    fprintf('[Bewegung] Drehe nach rechts\n');
else
    % Drehe nach links
    motorB.Power = -AbbiegenPower;
    motorA.Power = AbbiegenPower;
    motorB.SendToNXT();
    motorA.SendToNXT();
    fprintf('[Bewegung] Drehe nach links\n');
end

    % Verfolgen Funktion (je halbe Sekunde)

    else
    % 1. GetDistance
    currentDistance_1 = GetUltrasonic(SENSOR_2);
    currentDistance_2 = GetUltrasonic(SENSOR_1);

    % 2. Motor Steuerung
```

```
if currentDistance_1 > (targetDistance +
            tolerance)
    % 1) forword
    motorB.Power = motorPower;
    motorB.SendToNXT();
    motorA.Power = motorPower;
    motorA.SendToNXT();
    fprintf('[Bewegung] Geht nach vorne\n');

elseif currentDistance_1 < (targetDistance
    - tolerance)
    % 2) back
    motorB.Power = -motorPower;
    motorB.SendToNXT();
    motorA.Power = -motorPower;
    motorA.SendToNXT();
    fprintf('[Bewegung] Geht zurück\n');

else
    % 3) Stopp
    motorB.Power = stoppPower;
    motorB.SendToNXT();
    motorA.Power = stoppPower;
    motorA.SendToNXT();
    fprintf('[Bewegung] Stopp\n');
end
```

LITERATURVERZEICHNIS

[1] *LEGO NXT Motor*. https://forum.arduino.cc/t/lego-nxt-motor-arduino-motor-shield/266470, 2023. – Accessed: 2023-10-10

[2] *LEGO - Ultraschallsensor*. https://www.amazon.de/LEGO-NXT-Ultraschallsensor-Ultrasonic-sensor/dp/B000PM8I8O, 2023. – Accessed: 2023-10-10

[3] UNIVERSITY, RWTH A.: *RWTH Mindstorms NXT Toolbox*. https://www.mindstorms.rwth-aachen.de/de/trac/wiki/Download4.07, 2007. – Accessed: 2023-10-10

Follow-Me-Robot

Xiyue Xu, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Ziel des Projektseminars Elektrotechnik/Informationstechnik (LEGO Mindstroms) 2025 war es, eine Objektverfolgungsfunktion mit Hilfe eines Roboters aus einem LEGO Baukasten zu realisieren. Die Steuerung des Roboters erfolgt über die Steuerungskomponente NXT, die Datenerfassung über angeschlossene Sensoren und die Programmierung über MATLAB.**

***Schlagwörter*—LEGO Mindstroms, MATLAB, Verfolgen, Abstandsensor, Projektseminar**

I. Einleitung

DIE vorliegende Studie beschäftigt sich mit der Entwicklung von Robotersystemen unter Verwendung von LEGO-Bauelementen und der NXT-Steuerungseinheit. Im Forschungsprozess wurden technische Herausforderungen in Bezug auf Roboterstrukturen, Bewegungsmechanismen und Funktionsimplementierung systematisch analysiert. Unser Ziel bestand in der Realisierung eines innovativen LEGO-Roboters mit praktischer Relevanz, der durch kreative Funktionalität und technische Originalität überzeugt.

II. Vorbetrachtungen

In den Vorbetrachtungen werden zunächst bereits existierende Lösungen kurz vorgestellt und mit relevanten Literaturquellen untermauert. Darüber hinaus werden die für die eigene Lösungsfindung angewandten Verfahren erläutert.

A. Forschungszusammenhang

Lego-Robotik nimmt als interdisziplinäres Forschungsfeld eine wichtige Rolle in den Bereichen Bildung und technologische Entwicklung ein.Im Bildungssektor ermöglichen LEGO-Roboter durch ihre niedrigschwellige Zugänglichkeit eine praxisorientierte Vermittlung von MINT-Disziplinen (Mathematik, Informatik, Naturwissenschaften, Technik). Sie fördern nicht nur logisches Denken und kreative Problemlösungskompetenzen, sondern demokratisieren gleichzeitig den Zugang zu robotischen Grundprinzipien – insbesondere durch visuelle Programmierschnittstellen, die auch jüngere Zielgruppen einbeziehen. Parallel dazu dienen LEGO-basierte Prototypen in der Forschungs- und Entwicklungsphase als agiles Testmedium: Forschende können algorithmische Ansätze oder Steuerungskonzepte kosteneffizient und iterativ validieren, ohne auf komplexe Hardware-Infrastrukturen angewiesen zu sein.

B. Forschungsmotivation

Vor dem Hintergrund der LEGO-Robotik-Forschung zielt dieses Projekt auf die Entwicklung eines praxisrelevanten Folgeroboters ab, der Automatisierung, Informatik, Konstruktion und Sensortechnik interdisziplinär vereint. Durch die Integration von Sensorik, Steuerungsalgorithmen und modularer Mechanik wird ein iterativer Lernzyklus von der Konzeption bis zur Optimierung angestrebt. Das Projekt verkörpert die Kernprinzipien der LEGO-Robotik: die Transformation komplexer technologischer Herausforderungen in modular erweiterbare Systeme zur Förderung akademischer und praktischer Kompetenzen.

DOI: 10.24352/UB.OVGU-2025-011

C. Forschungsfragen

Diese Untersuchung konzentriert sich auf drei zentrale Fragestellungen: Erstens die algorithmische Umsetzung einer Steuerungslogik zur Einhaltung eines angemessenen Abstands zum Zielobjekt. Zweitens die physikalische Gestaltung und Implementierung von Lenkmechanismen sowie deren präzise zeitliche Steuerung während der Bewegung. Drittens die strukturelle Optimierung des Roboters, um eine stabile Balance zu gewährleisten und gleichzeitig ästhetische Aspekte zu berücksichtigen. Diese Fragestellungen vereinen Aspekte der Sensorik, Regelungstechnik und mechanischen Konstruktion innerhalb eines interdisziplinären Forschungsrahmens.

D. Forschungslimitationen

Die Studie nutzt LEGO-Bildungssets mit der NXT-Steuereinheit (32-bit ARM7-Mikrocontroller, 48 MHz Taktfrequenz) und einfachen Ultraschallsensoren (Messbereich 0 bis 255 cm, ±3 cm Genauigkeit). Aufgrund der hardwarebedingten Grenzen – begrenzte Rechenleistung (256 kB Flash-Speicher) und verzögerte Sensordatenverarbeitung – sind präzise Echtzeit-Objektverfolgung oder komplexe Regelungsalgorithmen nicht umsetzbar. Die Funktionalität bleibt auf Basisanwendungen mit reduzierter Reaktionsgeschwindigkeit beschränkt. Die in diesem Abschnitt verwendeten Daten beziehen sich auf Angaben aus [1].

III. Konkrete Umsetzung

In diesem Abschnitt werden die konzeptionellen Grundlagen und die praktische Umsetzung in drei Kernbereichen erläutert: Bewegungssteuerung, Code-Implementierung und Struktureller Aufbau.

A. Modulbeschreibung

Wie in Abbildung 1 zu sehen ist, handelt es sich bei dem gezeigten Bauteil um einen LEGO-Servomotor, der zur Steuerung von Drehbewegungen eingesetzt wird.

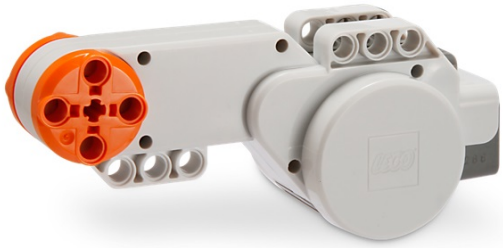


Abbildung 1: LEGO-Motor

Wie in Abbildung 2 dargestellt, erfasst der Ultraschallsensor Objektdistanzen im Bereich von $6\,\mathrm{cm}$ bis $255\,\mathrm{cm}$ und dient als zentrale Komponente der Abstandsregelung.

Abbildung 2: LEGO-Ultraschallsensor

Wie in Abbildung 3 zu sehen ist, Die Kernkomponenten des LEGO NXT umfassen die NXT-Steuereinheit (mit ARM7-Prozessor, Bluetooth-Unterstützung und Anschlüssen für Sensoren und Aktoren), verschiedene Sensoren (wie Tast-, Ultraschall-, Geräusch- und Lichtsensoren) sowie Servomotoren.

Abbildung 3: LEGO - NXT

B. Bewegungssteuerung

IV. Bewegungssteuerung

Das Regelungssystem folgt einem sequenziellen Ablauf: Nach dem Start aktiviert es die Sensoren zur Umgebungserfassung. Durch den Vergleich der Sensordaten (Links-/Rechtssensor) wird die Richtungssteuerung bestimmt: Bei einer Differenz von <-1 erfolgt eine Linkskurve, bei >1 eine Rechtskurve. Anschließend wird der gemessene Abstand mit dem Sollwert verglichen: Werte von <-3 lösen Rückwärtsfahrt aus, Werte von >3 Vorwärtsfahrt. Andernfalls stoppt das System. Dieser Algorithmus gewährleistet eine grundlegende Objektverfolgung unter den hardwarebedingten Echtzeitgrenzen des NXT-Systems.

In Abbildung 4 wird ein Programmablaufplan gezeigt.

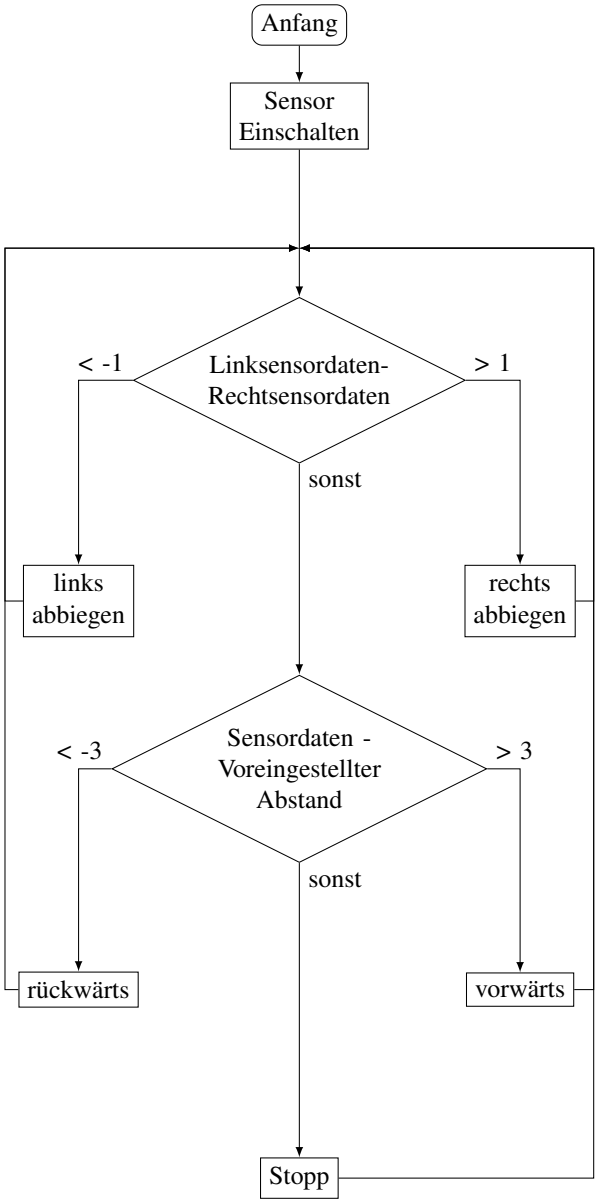


Abbildung 4: Programmablaufplan von Follow Me Robot

A. Code-Implementierung

Die Steuerungslogik wird mittels eines Programmablaufplans strukturiert und in MATLAB implementiert. Zunächst werden zentrale Parameter definiert: die Zielentfernung (`targetDistance`), die Entfernungstoleranz (`toleranz`), der Mindestabstand für Lenkmanöver (`AbbiegenDistance`), die Basisgeschwindigkeit (`motorPower`) sowie die Lenkgeschwindigkeit (`AbbiegenPower`). Die Ultraschallsensoren liefern Echtzeitdaten (`currentDistance_1`, `currentDistance_2`), aus deren Differenz (`Diff`) die Lenkentscheidungen abgeleitet werden.

Die Lenklogik aktiviert sich, wenn der absolute Wert der Sensordifferenz ($|$`Diff`$|$) den Schwellenwert `AbbiegenDistance` überschreitet. Überwiegt der linke Sensormesswert, wird der linke Motor mit `AbbiegenPower` vorwärts und der rechte rückwärts angesteuert – im umgekehrten Fall erfolgt eine spiegelverkehrte Steuerung. Liegt `Diff` innerhalb des Toleranzbereichs, vergleicht das System die gemessene Gesamtdistanz mit `targetDistance`. Überschreitet der Wert die Toleranzgrenze (`targetDistance` $\pm$ `toleranz`), fährt der Roboter vorwärts oder rückwärts; andernfalls stoppt er.

Ein Befehlszyklus von 0,5 Sekunden balanciert Reaktionsgeschwindigkeit und Systemstabilität, angepasst an die begrenzte Rechenleistung des NXT-Systems. Parameter wie `toleranz` werden objektspezifisch kalibriert – beispielsweise erfordert die Verfolgung von Personen höhere Toleranzen aufgrund ungleichmäßiger Oberflächenkonturen.

Der vollständige MATLAB-Code befindet sich im Anhang.

B. Struktureller Aufbau

Der Roboteraufbau orientiert sich an Funktionalität und Implementierungseffizienz. Für die Differentiallenkung sind zwei separat angetriebene Räder mit variabler Geschwindigkeit erforderlich, weshalb die Motoren seitlich angebracht wurden. Die Ultraschallsensoren wurden – basierend auf Vorversuchen – möglichst weit außen positioniert, um präzise Seitendistanzmessungen zu ermöglichen. Zur Gewichtsoptimierung wurde die NXT-Steuereinheit flach im Chassis platziert, wodurch eine stabile Gewichtsverteilung erreicht und Lenkmanöver begünstigt werden. Das folgende Abbildung 5 zeigt das finale Design des Follow-Me-Roboters

Abbildung 5: Aufbau des Follow-Me-Robot

V. Ergebnisdiskussion

A. Ergebnis

Damit hat der Follow-Me-Roboter sein Ziel erreicht, nämlich das Objekt zu verfolgen. Dies gilt insbesondere für flache Objekte, bei denen der Roboter in der Lage ist, Entfernungen schnell zu erkennen und Befehle zum Drehen und Folgen auszuführen. Auch das Verfolgen von Personen ist möglich, sofern die Fahrsgeschwindigkeit und die Entfernungstoleranz relativ groß gewählt werden.

B. Probleme – Beschränkungen des Ultraschallsensors

Hauptproblem: Aufgrund der Einschränkungen der Ultraschallsensoren kann der Follow-Me-Roboter nicht präzise erkennen, welchem Objekt er folgen soll. Er ist lediglich in der Lage, einem Objekt zu folgen, das sich direkt vor ihm befindet.

Erkennungsbereich: Die Perspektive der Sensoren begrenzt die Erkennungsgenauigkeit. Der Abstand zwischen den beiden Ultraschallsensoren definiert den Bereich, in dem der Roboter ein Objekt korrekt erfassen kann. Folglich darf sich das Zielobjekt nicht zu schnell bewegen, da der Roboter sonst das Ziel verliert. Zudem können unregelmäßig geformte Objekte, wie beispielsweise die Fußseiten einer Person, nicht zuverlässig erkannt werden, wenn sie sich im Erkennungsbereich der Sensoren befinden.

VI. Zusammenfassung und Fazit

Insgesamt hat das Projekt seine erklärten Ziele erreicht. Gleichzeitig kann es als Einstiegspunkt für das Verständnis der Autofollowing-Technologie dienen. Gleichzeitig gibt es noch weitere Probleme zu lösen und weitere Möglichkeiten zu erforschen.

VII. ANHANG

```
% Abbiegen Funkion
while true
% 1. Abstandswerte von beiden Sensoren lesen
    currentDistance_1 = GetUltrasonic(SENSOR_2);
    currentDistance_2 = GetUltrasonic(SENSOR_1);
    fprintf(['[Zustand] Linker Abstand: %.2f cm,
    Rechter Abstand:' ' %.2f cm\n'],
    currentDistance_1, currentDistance_2);

% 2. Entscheidung basierend auf den Abstandswerten
if abs(currentDistance_1 - currentDistance_2) >
       AbbiegenDistance

% 2.1 D_1 > D_2 nach rechts
if currentDistance_1 > currentDistance_2
% Drehe nach rechts
    motorB.Power = AbbiegenPower;
    motorA.Power = -AbbiegenPower;
    motorB.SendToNXT();
    motorA.SendToNXT();
    fprintf('[Bewegung] Drehe nach rechts\n');
else
    % Drehe nach links
    motorB.Power = -AbbiegenPower;
    motorA.Power = AbbiegenPower;
    motorB.SendToNXT();
    motorA.SendToNXT();
    fprintf('[Bewegung] Drehe nach links\n');
end

    % Verfolgen Funktion (je halbe Sekunde)

    else
    % 1. GetDistance
    currentDistance_1 = GetUltrasonic(SENSOR_2);
    currentDistance_2 = GetUltrasonic(SENSOR_1);

    % 2. Motor Steuerung
    if currentDistance_1 > (targetDistance +
                tolerance)
        % 1) forword
        motorB.Power = motorPower;
        motorB.SendToNXT();
        motorA.Power = motorPower;
        motorA.SendToNXT();
        fprintf('[Bewegung] Geht nach vorne\n');

    elseif currentDistance_1 < (targetDistance
        - tolerance)
        % 2) back
        motorB.Power = -motorPower;
        motorB.SendToNXT();
        motorA.Power = -motorPower;
        motorA.SendToNXT();
        fprintf('[Bewegung] Geht zurück\n');

    else
    % 3) Stopp
    motorB.Power = stoppPower;
    motorB.SendToNXT();
    motorA.Power = stoppPower;
    motorA.SendToNXT();
    fprintf('[Bewegung] Stopp\n');
end
```

LITERATURVERZEICHNIS

[1] THE LEGO GROUP: *Lego Mindstorms NXT Benutzerhandbuch*. Version 2.0. Billund, Dänemark: The Lego Group, September 2006. https://de.wikipedia.org/wiki/Lego_Mindstorms_NXT. – Offizielles Benutzerhandbuch

„MadLabCrab“ - Die LEGO Krabbe

Till Ehrhardt, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Rahmen des Projektseminares Elektrotechnik/Informationstechnik (LEGO Mindstorms) 2025 an der Otto-von-Guericke-Universität Magdeburg wurde mit Hilfe der zur Verfügung gestellten LEGO-Baukästen ein Roboter konstruiert, welcher in der Lage ist, sich selbständig zu positionieren und die Mitte eines dynamischen Umfeldes zu finden.**
Eine in MATLAB programmierte Software ermöglicht es der MadLabCrab, ein krabbenähnlicher Roboter, sich autonom fortzubewegen. Hierzu erfassen zwei Ultraschallsensoren die Umgebung. Die erfassten Daten werden von einem EV3 Steuerblock an einen PC übertragen, der eine zentrale Position berechnet und die Bewegungsrichtung bestimmt. Diese wird in Form von Steuerbefehlen zurückgesendet.

Die Fortbewegung wird durch acht Füße ermöglicht, die an vier verbundenen Beinkonstruktionen namens Klann-Mechanismus befestigt und von zwei Kurbelwellen sowie Motoren angetrieben werden. Auf die Vielfältigkeit dieser Art von Mechanismen, zu denen auch Joe Klanns Erfindung zählt, wurde durch ein kürzlich veröffentlichtes Video des YouTube-Kanals Veritasium aufmerksam gemacht.

***Schlagwörter*—Krabbe, Klann-Mechanismus, LEGO, MINDSTORMS, Roboter, OVGU**

I. Einleitung

Zu oft wird gesagt, man solle das Rad nicht neu erfinden. Doch genau dieser Gedanke spiegelt sich in der Konzeption und letztendlichen Entwicklung von MadLabCrab (MLC) wider. Gerade bei dieser Art von Projekten ist es wichtig, die gegebene Freiheit zu nutzen und sich von konventionellen Denkweisen zu lösen. Anstatt auf bereits etablierte Lösungen wie den Reifen zurückzugreifen, wurde für MadLabCrab die sogenannte Klann-Mechanismus zur Fortbewegung ausgewählt. Diese ermöglicht es dem Roboter unebenes Terrain zu passieren, wo Räder fehlschlagen würden.

Zudem können Gummireifen leicht durchstochen werden, was bei der Klann-Mechanismus mit ihren Füßen nicht möglich ist. Darüber hinaus kann eine Konstruktion aus einfachen Stangen mit vergleichsweise geringem Aufwand vor Ort repariert werden, während bei radbasierten Fahrzeugen oft das gesamte Gerät stillgelegt und aufgebockt werden muss. Außerdem kann eine Konstruktion aus mehreren Beinen die Last pro Bodenkontaktfläche effizienter reduzieren, insbesondere wenn sie erweiterbar ist. All diese Aspekte machen die MLC zu einer geeigneten Lösung für den Einsatz in z.B. Trümmergebieten, bei der Personenrettung und in abgelegenen Gebieten mit begrenztem Service. Das Konzept „Langsam und stetig gewinnt das Rennen“ wird durch die MadLabCrab veranschaulicht und dient als Idee für diese Art von Fortbewegungsmittel.

Um die Autonomie der Krabbe zu garantieren, wurden zwei Ultraschallsensoren (dargestellt in Abbildung 2) an der höchsten Stelle montiert. Sie messen in entgegengesetzte Richtungen, sodass MLC erkennt, in welche Richtung es sich bewegen muss. Der ideale zeitliche Ablauf einer solchen räumlichen Bewegung wird in Abbildung 1 dargestellt.

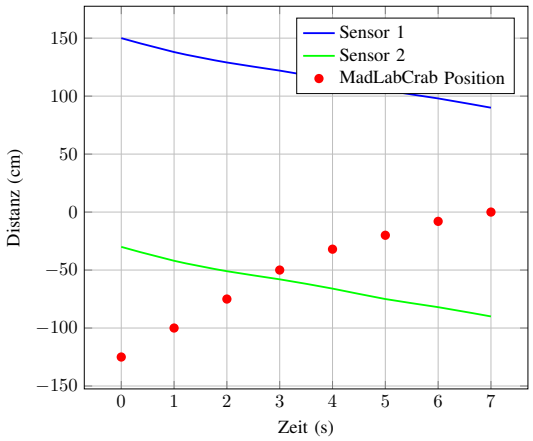


Abbildung 1. Räumliche Bewegung von MadLabCrab

Abbildung 2. NXT Ultraschallsensor [1]

II. Vorbetrachtungen

A. Mechanismus

Joe Klann, der Erfinder des Klann-Mechanismus, entwickelte diesen Mechanismus, um die Effizienz und Vielseitigkeit von Robotern zu steigern. Ein Video des YouTube-Kanals Veritasium, produziert von Derek Muller [2], machte erst vor kurzem auf diese Art von innovativen Mechanismen aufmerksam. Inspiriert von diesem und ähnlichen Projekten im Internet [3]–[5], gelang es der Gruppe 4, den Klann-Mechanismus in eine LEGO-Konstruktion umzuwandeln und erfolgreich für das MadLabCrab-Projekt zu verwenden. In der aktuellen Konstruktion werden vier Beine pro Seite verwendet, anstatt der ursprünglich von dem Klann-Mechanismus vorgesehenen sechs. Diese Veränderung wurde auf Grund der optimierten Stabilität und begrenzten Ressourcen vorgenommen.

B. Anforderungen

Die Anforderungen an MadLabCrab waren, dass es sich selbständig zwischen zwei dynamisch anpassenden Wänden

DOI: 10.24352/UB.OVGU-2025-012

Abbildung 3. Ultraschallsensoren & Not-Aus-Schalter montiert auf EV3

positioniert. Dazu kann es sich entlang einer Achse bewegen und stoppen, wenn die Sensoren beide die gleichen Werte ausgeben. Dies ist auch der Fall, sollte sich MLC in einer zu engen Lücke oder auf einer weiten Fläche, auf der es in beide Richtungen keine Wände erkennt, befinden. In beiden Fällen wurde festgelegt, dass MadLabCrab zum stehen kommt.

C. Komponenten

Zur Distanzerkennung wurden die LEGO-Ultraschallsensoren (siehe Abbildung 2) aus der NXT-Generation verwendet. Es wurde sich für diese Form der Messung entschieden, da die Sensoren performant sind und nur durch wenige äußere Einflüsse beeinträchtigt werden können. Damit eine sichere Bedienung der Krabbe garantiert werden kann, wurden außerdem zwei Knöpfe (einer zu sehen in Abbildung 3) eingebaut. Diese dienen einerseits dazu MLC freizuschalten, sodass der Programm-Timer erst ab diesem Moment gestartet wird, und andererseits auch zur frühzeitigen Beendigung des Programms. Der Not-Aus-Schalter wurde implementiert, um eine Möglichkeit zu bieten, die Bewegungen zu stoppen, wenn diese als gefährlich für den Roboter oder seine Umgebung eingeschätzt wurden.

Die Laufbewegung wurde durch zwei in der Mitte von MadLabCrab platzierte Motoren gesteuert. Zudem herrschte Symmetrie zwischen der linken und rechten Hälfte, sodass die Aufbauten gespiegelt waren. Der EV3 diente als Schnittstelle zwischen den Motoren (siehe Abbildung 10), Sensoren und dem Computer, welcher die Berechnungen anstellte.

III. Entwicklungsprozess

A. Erster Prototyp

Nachdem Gruppe 4 sich mit MATLAB vertraut gemacht hatte und sich auf ein Roboter-Konzept geeinigt hatte, wurden die ersten Testaufbauten begonnen. Um ein Gefühl für die Größenordnung zu bekommen, wurde zunächst eine Grundkonstruktion bestehend aus zwei Motoren und Kurbelwellen, jedoch ohne Füße und Sensoren, errichtet (siehe Abbildung 4, 5). Auf diese Weise konnten erste Bewegungstests durchgeführt werden.

Schnell wurde klar, dass Reibung ein Problem darstellt. Für den Grip der acht Füße wurden Gummistollen eingefügt (siehe Abbildung 6). Diese stellten an den vier Beinsegmenten den Kontakt zum Untergrund her und gewährleisteten somit eine robuste Verbindung.

Klann-Mechanismus Test: Nach vielen Versuchen und Anpassungen fand sich ein Verhältnis von LEGO-Teilen, Stangenlängen und passenden Komponenten, sodass schließlich eine Konstruktion gefunden wurde. Die Beine werden an zwei Punkten mit MLC verbunden: entlang der tragenden Stangen im oberen Bereich und an der kontinuierlich drehenden Kurbelwelle. Durch die Verlinkung der Stangen im Bein bewegt sich der Fuß auf einer vordefinierten Bahn und garantiert so die maximale Kraftübertragung zum Boden.

Abbildung 4. Motor, Kurbelwelle und Zahnräder Version 1

B. Zweiter Prototyp

Der erste MadLabCrab-Prototyp lieferte wertvolle Erkenntnisse, die für die weitere Entwicklung entscheidend waren. Es wurde klar, dass die Stabilität des Roboters, insbesondere im Hinblick auf Größe und Gewichtsverteilung, erhöht werden musste. Zudem offenbarten sich Schwachstellen bei der Kraftübertragung von den Motoren auf die Klann-Mechanik, da die diese zeitweise durchdrehten. Das maximale Drehmoment der LEGO NXT Motoren beträgt $50\,\mathrm{N\,cm}$. Diese Erfahrungen ermöglichten es Gruppe 4, gezielt Verbesserungen vorzunehmen.

Die Anzahl der Träger wurde auf vier durchgehenden erhöht, und es wurden möglichst lange Segmente verwendet (siehe Abbildung 10). Dies führte zu einer erhöhten Stabilität der gesamten Konstruktion, was insbesondere bei der Bewegung von MadLabCrab hilfreich war.

Abbildung 5. LEGO-Kurbelwelle

Abbildung 6. Fußkonstruktion

Die Übersetzungsverhältnisse der Zahnräder wurden von 3:10 auf eine Konstruktion mit den Verhältnissen 3:10 und 1:5 geändert (siehe Abbildung 7). Dies führte zwar zu einem Verlust an Geschwindigkeit, jedoch gewann MLC dadurch erheblich an Kraft. Die zusammengestellte Konstruktion aus Motor, Zahnrädern und Kurbelwelle ist in Abbildung 11 dargestellt.

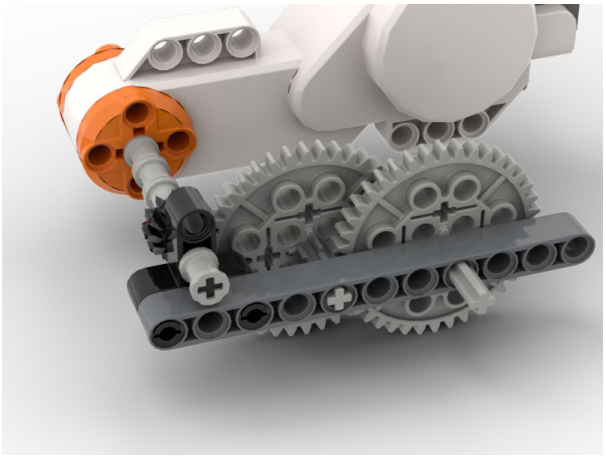


Abbildung 7. Motor und Zahnräder Version 2

C. Finaler Aufbau und Test

Nach zwei Wochen war die Entwicklung nun zur finalen Revision fortgeschritten. MadLabCrab konnte sich selbstständig bewegen und positionieren. Die vorherigen Entwicklungsschritte hatten den krabbenähnlichen Roboter sichtlich verbessert und optimiert (siehe Abbildung 8).

Abbildung 8. Finale Version von MadLabCrab, aufgebockt

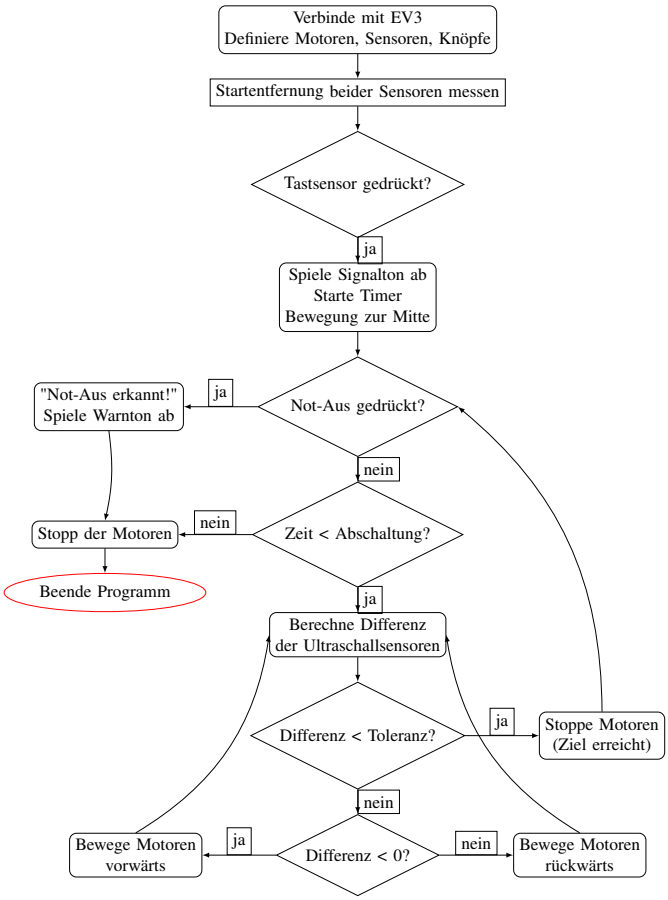
D. Algorithmus

Der Vorgang, wie MadLabCrab entscheidet, in welche Richtung es sich bewegen soll, ist im Programmablaufplan in Abbildung 9 dargestellt. Zunächst wird eine Verbindung zum EV3 hergestellt und die Motoren, Ultraschallsensoren sowie Knöpfe werden definiert. Nun wird gewartet, bis der Start-Tastsensor gedrückt wird.

Sobald der Tastsensor gedrückt wird, ertönt ein Signalton und der Timer beginnt. MadLabCrab startet die Bewegung zur Mitte. Derweil wird kontinuierlich die Differenz der Sensoren Werte errechnet. Liegt diese innerhalb der Toleranz, stoppen die Motoren, da das Ziel erreicht ist. Wenn MLC zu nah an Wand A ist, bewegen sich die Motoren vorwärts, bei zu großer Nähe zu Wand B drehen die Motoren rückwärts.
Wird der Not-Aus-Schalter gedrückt, stoppt der Roboter sofort und gibt ein Warnsignal ab. Andernfalls passt sich MadLabCrab kontinuierlich an die Umgebung an bis zum Ablauf der Zeit.

IV. Ergebnisdiskussion

Mit dem Abschluss der zweiwöchigen Entwicklung ist MadLabCrab nun erfolgreich in der Lage, sich autonom zu positionieren. Der Roboter kann seine Umgebung wahrnehmen und basierend auf diesen Daten handeln. Falls es zu einer Fehlfunktion kommen sollte kann MLC jedoch immer durch den Not-Aus-Schalter zum Stillstand gebracht werden.



Figur 9. Programmablaufplan von MLC

Probleme traten insbesondere bei der Tragekonstruktion auf, da LEGO-Teile, welche aus ABS hergestellt werden, und die Toleranzen zwischen den Steckverbindungen zu Instabilität führten. Dadurch ergab sich unausweichlich eine konkave Form entlang der Stangen.
Auch bei der Verwendung der Ultraschallsensoren zeigten sich Schwachstellen. Diese besitzen einen breiten Messwinkel, wodurch es zu fehlerhaften Werten kommen kann, die auf unerwünschten Hindernissen basieren. Darüber hinaus ist die Reichweite auf 255 cm begrenzt. Bei flachen Winkeln kann es zudem passieren, dass die Schallwellen von einer Oberfläche abgelenkt werden, wodurch diese nicht korrekt zum Sender zurückkehren.
Typischerweise wird der EV3-Baustein auch in die Konstruktion integriert, wobei die Verbindungskabel entsprechend kurz gehalten sind. Die Größe der Motoren und der Klann-Mechanismus führten jedoch dazu, dass die Krabbe relativ breit wurde und die Beine mehrere Zentimeter vom Schwerpunkt entfernt angebracht werden mussten. Aufgrund der beschriebenen Komplikationen wurde der EV3 letztendlich unter MadLabCrab mittig positioniert für einen Trockentest, bei dem sich nicht die Krabbe selbst, sondern die Wände zu bzw. weg von den Sensoren bewegen.

V. Zusammenfassung und Fazit

Das Projekt hat wertvolle Lernerfahrungen im Umgang mit MATLAB und der Robotik gebracht. Alle Teilnehmer wurden gefordert, kreativ und zielbewusst unter Zeitdruck zu entwickeln.
Mögliche Verbesserungen ergaben sich aus den bisherigen Tests, unter anderem durch stabilere Materialien für die Tragstruktur und die Optimierung der Kraftübertragung. Durch gezielte Anpassungen könnten zukünftige Versionen robuster und leistungsfähiger werden. Zusätzlich besteht das Potenzial, diese Art von Mechanismen für größere Maschinen zu skalieren und für diverse Anwendungsbereiche, z. B. die Menschenrettung in schwierigem Terrain, zu nutzen.

Anhang

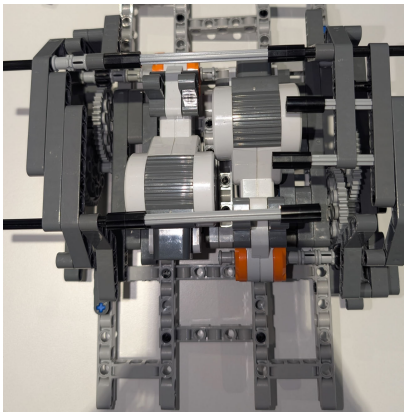


Abbildung 10. Motor-Aufbau ohne Stabilisierungsrahmen

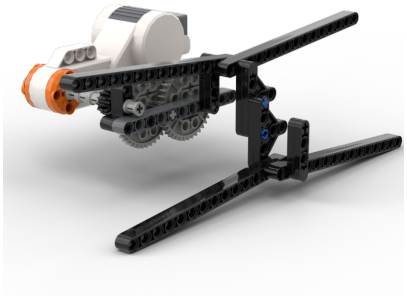


Abbildung 11. Motor, Kurbelwelle und Zahnräder Version 2

Literaturverzeichnis

[1] LEGO: *NXT Ultraschallsensor*. Amazon. https://amzn.eu/d/is9tdVN. Version: 2006. – Abgerufen von Amazon: https://m.media-amazon.com/images/I/71SM4iqillS._AC_SX679_.jpg
[2] Muller, Derek: *34 Years Of Strandbeest Evolution*. https://www.diywalkers.com/klanns-spider-ev3-long-legs.html. Version: 07.12.2024. – YouTube Video
[3] *Klann's Mechanical Spider with Longer Legs (Ver 3)*. https://www.diywalkers.com/klanns-spider-ev3-long-legs.html. Version: n.d.. – Online Projekt
[4] *Klann's Linkage Dimensions*. https://www.diywalkers.com/klanns-linkage-plans.html. Version: n.d.. – Online Projekt
[5] Texas Wiki, University of: *05 - Spider Walking Mechanism*. https://cloud.wikis.utexas.edu/wiki/spaces/RMD/pages/51059372/05+-+Spider+Walking+Mechanism. Version: n.d.. – Online Resource

MadLabCrab
Ein Krabbenroboter

Josua Eric Lohse, Elektro und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

Zusammenfassung - Im Rahmen des LEGO-Projektseminars wurde ein achtbeiniger Roboter entwickelt, der sich krabbenartig seitlich fortbewegen kann. Das Hauptziel des Projektes bestand darin, den Roboter dazu zu befähigen, sich seitlich im Raum zu bewegen und sich selbständig in der Mitte zwischen zwei definierten Punkten zu positionieren. Trotz einiger technischer Herausforderungen konnte ein funktionsfähiger Prototyp erfolgreich umgesetzt werden. Die vorliegende Arbeit dokumentiert die einzelnen Entwicklungsschritte und die technischen Lösungen, die zur Realisierung dieses Roboters führten.

***Schlagwörter*— Autonomer Roboter, Beinrobotik, Klann-Mechanismus, LEGO-Mindstorms, Sensorbasierte Steuerung**

I. Einleitung

DIE Entwicklung von Robotern mit Beinantrieb gewinnt zunehmend an Bedeutung für verschiedene Einsatzbereiche. Ein autonomer Roboter in Form eines Krabbeltiers, der sich nach dem Klann-Mechanismus bewegt, bietet eine vielseitige Lösung. Er kann in der Industrie zur Inspektion mit Kameras eingesetzt werden und in Rettungseinsätzen in unwegsamen Gebieten helfen. Seine spezielle Art der Fortbewegung ermöglicht es ihm, sich gut anzupassen und beweglich zu sein. Diese Arbeit beschreibt den vollständigen Entwicklungsprozess – von der ersten Idee über die technische Umsetzung bis hin zur finalen Realisierung des Prototyps. Neben der Funktionsweise des Roboters werden auch die eingesetzten Mechanismen, Komponenten und Programmieransätze erläutert. Ziel ist es, ein umfassendes Verständnis für die Konstruktion und Steuerung eines autonomen, beinbetriebenen Roboters zu vermitteln.

II. Vorbetrachtungen

Die Idee, die zur Entwicklung des Roboters führte, wird im Folgenden näher beleuchtet. Die Funktionsweise und die verwendeten Bauteile werden dabei in einem parallelen Prozess erläutert, um ein umfassendes Verständnis zu gewährleisten

A. Was soll der Roboter können?

In erster Linie soll der Roboter eigenständig ohne weitere Stützhilfen sich in einer geraden Linie bewegen. Zwei Ultraschallsensoren messen dann die Entfernungen zu einer beweglichen Wand rechts und links vom Roboter. Sollte der Roboter sich näher an der rechten Wand befinden werden die Motoren aktiviert und er geht in Richtung der linken Wand. Sollte der Roboter sich näher an der linken Wand befinden, drehen sich die Motoren in die entgegengesetzte Richtung und er bewegt sich nach rechts. Wenn der Roboter sich in der Mitte der Wände positioniert hat, ertönt ein Signalton und der Roboter schaltet sich ab. Sollte der Roboter eine Fehlfunktion haben oder beschädigt werden, kann außerdem ein zusätzlicher Knopf gedrückt werden und der Roboter schaltet sich ab.

Abbildung 1: Finale Roboterversion

B. Welche Komponenten wurden benötigt?

Die zentrale Steuerung des Roboters wird durch den LEGO-Mindstorms EV3-Baustein realisiert. Dieser Baustein stellt eine Schnittstelle zwischen einem Computer und den weiteren Roboterelementen dar. Über eine Verbindung zum Computer werden Befehle aus MATLAB-Code an die andern Bauelemente weitergeleitet, während Sensordaten, wie zum Beispiel Abstandsmessungen, zur weiteren Verarbeitung an den Computer zurückgesendet werden. Zur Realisierung der Fortbewegung mittels acht Beinen wurden zwei leistungsstarke LEGO-Mindstorms-EV3-Motoren (95658) implementiert. Diese Motoren gewährleisten die erforderliche Drehmomentleistung, um die Gravitationskräfte, die durch das Eigengewicht des

DOI: 10.24352/UB.OVGU-2025-013

Roboters entstehen, zu überwinden. Des Weiteren wurden zwei LEGO-Mindstorms Ultraschallsensoren (9846) zur Erfassung von Umgebungsdaten eingesetzt. Diese Sensoren ermöglichen die simultane Distanzmessung zu beiden Seiten des Roboters mit einem maximalen Messbereich von 255 cm. Zusätzlich verfügt der Roboter über zwei Tastsensoren. Ein Sensor dient als Initialisierungseinheit zur Aktivierung des Roboters, während der zweite Sensor als Not-Aus-Schalter konfiguriert ist, um im Bedarfsfall eine sofortige Systemabschaltung zu ermöglichen.

III. Realisierung

In diesem Abschnitt werden die einzelnen Entwicklungsschritte detailliert beschrieben, die zur erfolgreichen Realisierung einer funktionsfähigen Roboterversion führten.

A. *Programmablaufplan und Code*

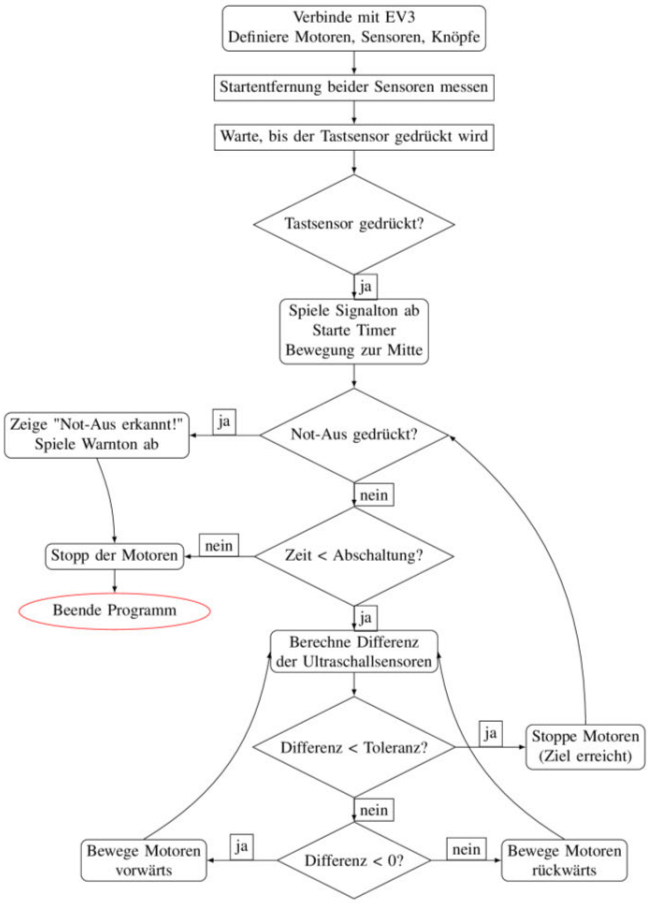


Abbildung 2: Programmablaufplan für MLC

In Abb. 2 ist zu sehen, dass nach der Initialisierung, bei der die Verbindung zum EV3-Baustein hergestellt, die Motoren, Sensoren und Tasten definiert und die Startentfernungen der Ultraschallsensoren gemessen werden, wartet der Roboter auf ein Startsignal durch den Tastsensor. Sobald dieses Signal eintrifft, gibt der Roboter ein akustisches Signal aus, gibt der Roboter ein akustisches Signal aus, startet einen internen Timer und beginnt mit der Bewegung zur Mitte, wobei er die Motoren entsprechend ansteuert. Während dieser Bewegung überprüft der Roboter kontinuierlich, ob der Notausschalter betätigt wurde. Ist dies der Fall stoppt der Roboter sofort die Motoren und gibt ein akustisches Warnsignal aus. Sollte dieser Knopf allerdings nicht betätigt werden, überprüft der Roboter, ob die vorgegebene Zeit abgelaufen ist. Sollte die Zeit abgelaufen sein und der Roboter befindet sich noch nicht in der Mitte läuft er trotzdem weiter, bis er in dem mittigen Toleranzbereich angekommen ist und schaltet sich danach aus. Wenn der Toleranzbereich vor Ablauf der Zeit erreicht wurde, bleibt er dort stehen und überprüft weiter den Abstand der Wände, bis der Timer runtergelaufen ist und er sich abschaltet

B. *Wie funktioniert der Laufmechanismus*

Für die Fortbewegung des Roboters wurde der Klann-Mechanismus [1] gewählt – ein speziell konstruiertes Koppelstangengetriebe, das eine beinartige Bewegung ermöglicht. Dieser Mechanismus besteht aus mehreren starr miteinander verbundenen Stangen, die über Drehpunkte präzise aufeinander abgestimmt sind und durch eine Kurbelwelle (Abb. 3) in Bewegung gesetzt werden. Die dadurch entstehende Mechanik sorgt dafür, dass sich die Füße des Roboters in der Vorwärtsbewegung nahezu geradlinig über den Boden bewegen, während sie in der Rückwärtsbewegung angehoben werden. Dadurch entsteht ein stabiler, gleichmäßiger Gang, der sich besonders für unebenes Terrain eignet. Die Konstruktion des Klann-Mechanismus erlaubt eine relativ flache und kontrollierte Fortbewegung, die der natürlichen Bewegung von Krabbeltieren wie Krabben ähnelt. Dies verleiht dem Roboter eine hohe Standfestigkeit und eine effiziente Laufbewegung, ohne dass zusätzliche Räder oder herkömmliche Beine erforderlich sind.

Abbildung 3: Kurbelwelle

C. *Weitere nennenswerte Entwicklungsschritte*

Ein entscheidender Fortschritt in der Entwicklung des Motorblocks und der Kraftübertragung war die Optimierung der Zahnradübersetzung zwischen den Motoren und der Kurbelwelle. Während der ersten Testphase zeigte sich, dass der

Roboter nicht über ausreichend Antriebskraft verfügte, um sein Eigengewicht zu bewegen. Eine Erhöhung der Anzahl der Motoren hätte zwar die Leistung verbessert, hätte jedoch auch zu einer unerwünschten Gewichtszunahme geführt. Daher war eine alternative, effizientere Lösung erforderlich. Die finale Umsetzung bestand aus einer Zahnraduntersetzung von 3:10, gefolgt von einer weiteren Übersetzung im Verhältnis 1:5 (Abb. 4). Diese Modifikation führte zwar zu einer Reduzierung der Drehzahl der Kurbelwelle, erhöhte jedoch gleichzeitig das Drehmoment erheblich. Durch diese Steigerung der Drehkraft konnte der Roboter schließlich sein Eigengewicht überwinden und die gewünschte Leistung zuverlässig erbringen.

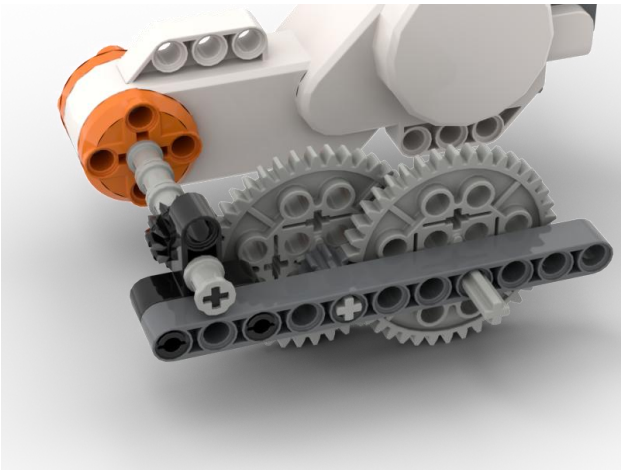


Abbildung 4: Getriebe und Zahnradübersetzung

Ein letzter, wesentlicher Entwicklungsschritt war die Konstruktion der Beine, die mehrere Anforderungen erfüllen mussten. Sie durften weder zu schwer, noch zu instabil sein, um das Eigengewicht zu tragen. Zudem war ein optimales Verhältnis zwischen Starrheit und Beweglichkeit erforderlich. Das finale Design wurde auf Basis eines bestehenden Robotermodells [2] entwickelt, und ist in Abb.5 dargestellt.

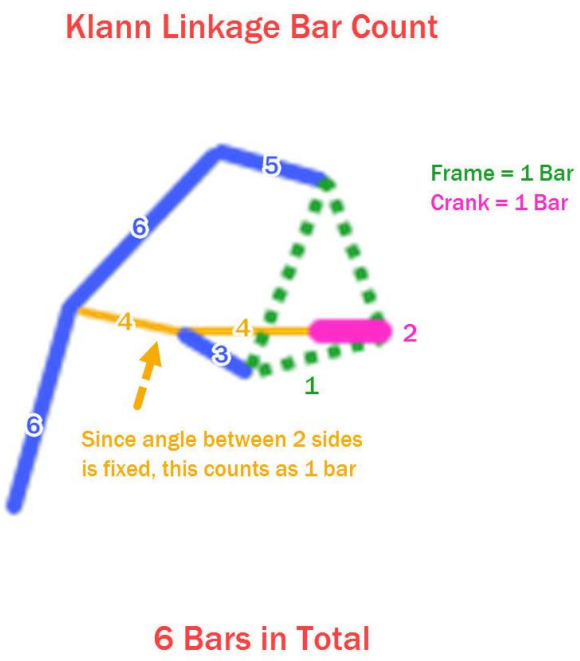


Abbildung 5 Beindesign[2]

IV. Ergebnisdiskussion

Im Rahmen des zweiwöchigen LEGO-Projektseminars konnte ein voll funktionsfähiger Roboter entwickelt werden (Abb. 1). Trotz einiger Herausforderungen während der Demonstration gelang es, das angestrebte Ziel einer zentrierten Positionierung im Raum erfolgreich zu veranschaulichen. Ein ungelöstes Problem stellte jedoch das gelegentliche Lösen der Kurbelwelle von der Getriebebox dar, was durch die hohe mechanische Belastung verursacht wurde. Dieses Problem resultierte aus der begrenzten Klemmkraft der LEGO-Bauteile, die für derart hohe Kräfte nicht ausgelegt sind.

V. Zusammenfassung und Fazit

Das Projektseminar zeigte, dass es möglich ist, mit dem LEGO-Klemmbausteinsystem einen funktionsfähigen, autonom bewegenden Roboter zu realisieren. Die Nutzung des Klann-Mechanismus für die Fortbewegung erwies sich als effektive Lösung, auch wenn einige mechanische Herausforderungen bewältigt werden mussten. Während der Demonstration konnte das Hauptziel des Projekts erfolgreich veranschaulicht werden. Für zukünftige Verbesserungen könnten insbesondere eine stabilere Befestigung der Kurbelwelle (Abb. 3) sowie eine optimierte Gewichtsverteilung in Betracht gezogen werden, um die Leistung und Zuverlässigkeit weiter zu steigern. Insgesamt bietet die entwickelte Lösung eine solide Grundlage für weiterführende Arbeiten in diesem Bereich.

Literaturverzeichnis

[1] Google Patents: Walking device
https://patents.google.com/patent/US6260862B1/en
[2] DIY Walkers Klann's Linkage Dimension
https://www.diywalkers.com/klanns-linkage-plans.html

Gyroskopisch Gesteuerter Roboter-Arm „Project MoJo“

Moritz Heucke, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Das hier dokumentierte Projekt entstand im Zuge des Projektseminars Elektrotechnik/Informationstechnik an der Otto-von-Guericke-Universität Magdeburg. Es handelt sich um einen dreiachsigen Roboter-Arm aus LEGO-Bausteinen, der durch menschliche Eingabe gesteuert wird. Als Eingabemethode dient ein Control-Stick, in dem Gyroskopsensoren verbaut sind. Ziel ist es, dass der Arm stets die räumliche Ausrichtung dieses Control-Sticks ansteuert und so die menschliche Bewegung reproduziert. Diese Arbeit befasst sich mit dem Aufbau und der Funktionsweise sowie Erfolgen und Limitierungen des Projektes.**

***Schlagwörter*—EV3, Gyroskop, LEGO, MATLAB, Roboter**

I. Einleitung

EGAL ob in der Forschung, Medizin oder Katastrophenhilfe, von Menschenhand ferngesteuerte Robotik findet zahlreiche Anwendungszwecke. Diese Technologie vermindert Gesundheitsrisiken, kann die Präzision menschlicher Bewegungen verbessern und körperlich belastende Arbeit erleichtern. Im Gegensatz zu autonomen Systemen wird menschliches Urteilsvermögen mit maschineller Kraft und Präzision kombiniert. Auf dieser Idee basiert das hier vorgestellte Projekt. Der Roboter soll durch die gyroskopische Steuerung möglichst intuitiv und natürlich bedient werden können. Dabei tun sich Herausforderungen bei der Konstruktion und Programmierung auf. Denn der Arm soll einen möglichst großen Bewegungsraum haben und die Drehung des Control-Sticks im Raum muss akkurat erfasst werden.

II. Vorbetrachtungen

Im Folgenden werden wichtige Informationen bezüglich genutzter Bauteile gegeben. Ziel ist es, die Funktionsweise des Roboters verständlicher zu machen.

A. Entwicklungsplattform

Als Kontrolleinheit wird der LEGO Mindstorms EV3 verwendet (Abb. 1). Dieser wird mit der Programmiersprache MATLAB in der MATLAB-Entwicklungsumgebung programmiert. Hierzu wird eine von der RWTH Aachen zur Verfügung gestellte Toolbox [1] verwendet, welche die Kommunikation zum EV3 über MATLAB ermöglicht und hilfreiche Kontrollobjekte für Motoren und Sensoren bietet. Auch vereinfacht die Programmiersprache den Umgang mit Vektoren und Matrizen, was für dieses Projekt von Nutzen ist. Der EV3 verfügt im Vergleich zum Vorgängermodell NXT über einen weiteren Motor-Port und eine höhere Rechenleistung [2]. Besonders letzterer Vorteil wird gebraucht, da die Ausführungsgeschwindigkeit des Skripts möglichst hoch sein muss.

DOI: 10.24352/UB.OVGU-2025-014

B. Gyroskopsensoren

Zur Erfassung von Drehbewegungen werden LEGO-Gyroskope genutzt (Abb. 2). Diese können Rotation jeweils nur um eine Achse messen, weshalb im Control-Stick drei senkrecht zueinanderstehende Sensoren verbaut sind. Dies ermöglicht die Wahrnehmung von Drehungen um alle möglichen im Raum liegenden Achsen. Des Weiteren lassen sich sowohl absolute Winkel als auch Winkelgeschwindigkeiten auslesen. Da zur Umsetzung der Positionserfassung möglichst präzise Winkeländerungen $\Delta\varphi$ gemessen werden müssen, werden in diesem Projekt die Winkelgeschwindigkeiten ω in kleinen Zeitintervallen gemessen und mit der entsprechenden Zeitspanne multipliziert: $\Delta\varphi = \omega \cdot \Delta t$. Dadurch erfolgt eine deutlich genauere Bestimmung der Winkeldifferenzen, weil die absolute Winkelmessung nur ganzzahlige Werte ausgibt. Auch ist zu beachten, dass die Gyroskope beim Einschalten nicht bewegt werden dürfen, da sie sonst falsch kalibriert sind.

Abbildung 1: LEGO Mindstorms EV3 [3]

Abbildung 2: LEGO Mindstorms Gyroskop [4]

III. Aufbau und Funktionsweise

Das Projekt setzt sich aus Hardware- und Softwarekomponenten zusammen. In diesem Abschnitt wird auf die bauliche und programmiertechnische Umsetzung eingegangen.

A. Konstruktion

Die baulichen Hauptmerkmale umfassen den Control-Stick, den LEGO EV3, den Roboter-Arm mit drei Motoren sowie einen Tastsensor als Stopp-Knopf, der am unteren Gerüst des Roboters befestigt ist (Abb. 3, 4). Die Steuerung erfolgt über den Control-Stick, in dessen unterem Teil drei Gyroskope verbaut und über LEGO-Datenkabel mit dem EV3 verbunden sind. Der obere Teil dient als Griff und ein Standfuß erleichtert die Kalibrierung der Sensoren. Die von den Gyroskopen erfassten Winkelgeschwindigkeiten werden vom EV3 ausgelesen und durch ein MATLAB-Skript verarbeitet.
Der Arm besteht aus einem rotierenden Basissegment, auf dem ein weiteres Segment angebracht ist, das um eine orthogonale Achse rotiert. Ein zusätzliches Gelenk verbindet das mittlere Segment mit einer Plattform, auf der Sensoren oder andere Instrumente installiert werden können. Die Rotationen der unteren beiden Gelenke bewirken eine Ausrichtung des Arms in die Richtung, in die der Control-Stick zeigt. Das obere Gelenk ermöglicht eine zusätzliche Drehung der Plattform um ihre eigene Achse, wodurch die exakte Orientierung des Control-Sticks im Raum nachgebildet wird. Die gesamte Konstruktion ist auf einem stabilen Gerüst montiert, das für einen sicheren Stand sorgt. An Arm und Stick ist je ein blauer Pin angebracht, mit dem die richtige Startorientierung sichergestellt werden kann.

B. MATLAB-Skript

Nach der Initialisierung der Motor- und Sensorobjekte sowie der Definition globaler Variablen wird eine While-Schleife ausgeführt, die erst durch das Betätigen des Stopp-Knopfs beendet wird. Innerhalb dieser Schleife werden kontinuierlich Sensordaten erfasst und daraus die Soll-Winkel für die Motoren berechnet. Anschließend erfolgt die Ansteuerung der Motoren gemäß den berechneten Werten. Beim Drücken des Stopp-Knopfs kehren die Motoren in ihre Ausgangsposition zurück. Abbildung 5 zeigt eine schematische Darstellung des Skripts.

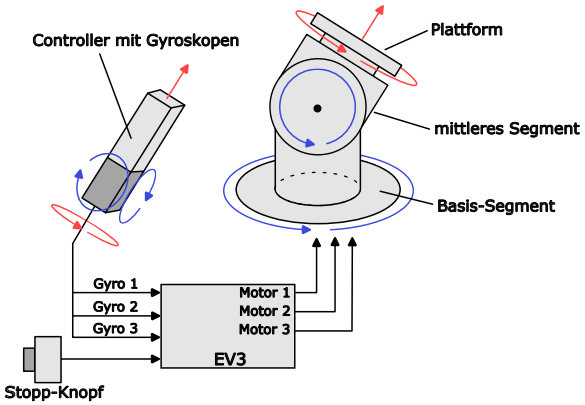


Abbildung 3: Aufbau des Roboters

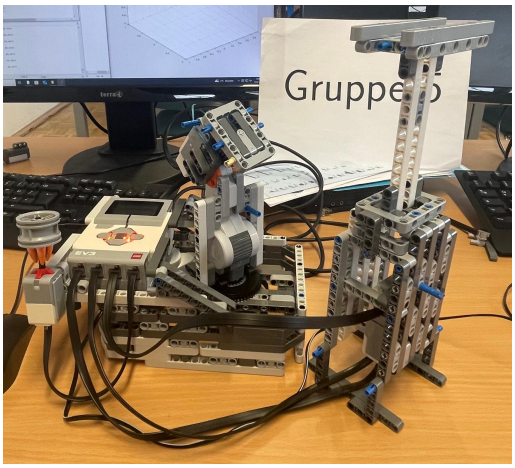


Abbildung 4: Finale Umsetzung des Roboters

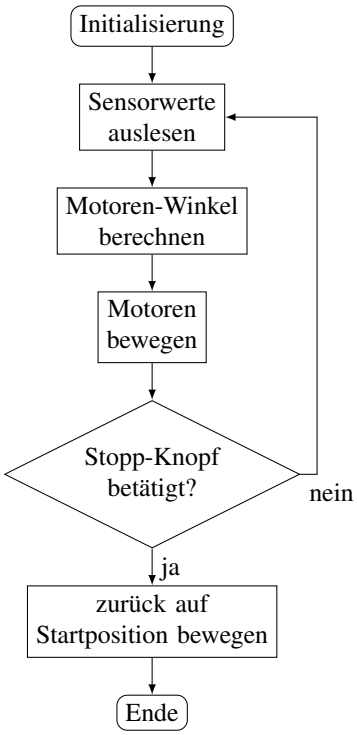


Abbildung 5: Programmablaufplan des MATLAB-Skripts

C. Erfassung der Ausrichtung des Control-Sticks

Eine zentrale Herausforderung in der Entwicklung besteht darin, dass die Gyrodaten nicht direkt zur Steuerung der Motoren verwendet werden können. Der Grund hierfür liegt in der sich ständig verändernden Orientierung der Rotationsachsen der Gyroskope während der Bewegung des Control-Sticks. Jede Gyroskopachse kann frei im Raum gedreht werden. Im Gegensatz dazu kann das Basissegment des Arms nur um eine räumlich feste Achse rotieren, während die nachfolgenden Segmente jeweils um eine weitere bewegliche Achse rotieren. Lediglich das obere Plattformsegment kann direkt über den Gyroskopwert des entsprechenden Sensors gesteuert werden, da beide Drehachsen parallel zueinander ausgerichtet sind. Dieser

Zusammenhang ist in Abbildung 3 farblich dargestellt. Zur Lösung dieses Problems wird ein System aus drei orthogonalen Einheitsvektoren aufgestellt, welche die Gyroskopachsen repräsentieren. Einer dieser Vektoren dient als Richtungsvektor, an dem sich der Roboter-Arm ausrichtet. In den Abbildungen 3 und 6 ist dieser rot gefärbt. Während der Bewegung des Control-Sticks müssen die Vektoren so transformiert werden, dass sie stets die Ausrichtung der Gyrosachsen übernehmen. Die Basisvektoren werden in zwei Matrizen $[\mathbf{V}]$, $[[\mathbf{V}_t]] \in \mathbb{R}^{3\times3}$ als Spaltenvektoren gespeichert. Die Matrix $[\mathbf{V}]$ enthält die Ausgangsvektoren vor der Transformation und $[\mathbf{V}_t]$ enthält die transformierten Vektoren. Vor der Rotation sind $[\mathbf{V}]$ und $[\mathbf{V}_t]$ identisch. Nach der Messung der drei Gyroskopwinkel wird jeder der Spaltenvektoren in $[\mathbf{V}_t]$ nacheinander um alle Spaltenvektoren in $[\mathbf{V}]$ um die gemessenen Winkel φ rotiert. Hierfür eignet sich die „Rodrigues' rotation formula" (1), mit der ein beliebiger Raumvektor um eine definierte Achse und einen bestimmten Winkel rotiert werden kann [5].

$$\vec{v}_{\text{rot}} = \vec{v}\cos\varphi + (\vec{k}\times\vec{v})\sin\varphi + \vec{k}(\vec{k}\cdot\vec{v})(1-\cos\varphi) \qquad (1)$$

$\vec{v}_{\text{rot}}$	rotierter Vektor
$\vec{v}$	zu rotierender Vektor
$\vec{k}$	Rotationsachse (Einheitsvektor)

Sowohl $\vec{v}$ als auch $\vec{k}$ sind Spaltenvektoren von $[\mathbf{V}]$. Die Berechnung der Gesamtrotation eines Vektors erfolgt durch die aufeinanderfolgende Anwendung von Teilrotationen. Diese Teilrotationen sind die von den Gyroskopen gemessenen Winkeldifferenzen $\varphi_{x,y,z}$. Diese Herangehensweise basiert auf der kommutativen Additivität von Winkelgeschwindigkeiten. Das Prinzip wird in Quelle [6] näher erläutert.

$$\begin{aligned}
\vec{\omega}_{\text{ges}} &= \vec{\omega}_x + \vec{\omega}_y + \vec{\omega}_z \\
\vec{e}_r\frac{\mathrm{d}\varphi_{\text{ges}}}{\mathrm{d}t} &= \vec{e}_x\frac{\mathrm{d}\varphi_x}{\mathrm{d}t} + \vec{e}_y\frac{\mathrm{d}\varphi_y}{\mathrm{d}t} + \vec{e}_z\frac{\mathrm{d}\varphi_z}{\mathrm{d}t} \\
\vec{e}_r\,\mathrm{d}\varphi_{\text{ges}} &= \vec{e}_x\,\mathrm{d}\varphi_x + \vec{e}_y\,\mathrm{d}\varphi_y + \vec{e}_z\,\mathrm{d}\varphi_z
\end{aligned} \qquad (2)$$

$\vec{\omega}_{\text{ges}}$	Gesamtwinkelgeschwindigkeit
$\vec{\omega}_{x,y,z}$	Teilgeschwindigkeiten um Gyro-Achsen
$\vec{e}_r$	Einheitsvektor parallel zur Rotationsachse
$\vec{e}_{x,y,z}$	Einheitsvektoren parallel zu Gyro-Achsen

Da die Addition von Teildrehungen zu einer Gesamtdrehung nur für differenzielle Winkel exakt ist, müssen möglichst kleine Winkelveränderungen $\Delta\varphi$ erfasst werden, um eine präzise Steuerung des Roboter-Arms zu gewährleisten. Nach der Transformation werden die Spaltenvektoren aus $[\mathbf{V}_t]$ in $[\mathbf{V}]$ übernommen und bilden die Rotationsachsen für die nächste Transformation. Der Prozess ist schematisch in Abbildung 6 dargestellt. Zuletzt wird der zuvor festgelegte Richtungsvektor in Kugelkoordinaten (r, α, β) umgewandelt. Die Winkel α und β können nämlich direkt von den Motoren angesteuert werden, damit der Arm die Ausrichtung des Vektors übernimmt.

D. Steuerung der Motoren

Zum Antrieb des Roboters werden Servomotoren genutzt, was bedeutet, dass ihre aktuelle Position jederzeit ausgelesen werden kann. Außerdem ist die Eigenschaft „speedRegulation" aktiviert, wodurch die Geschwindigkeit direkt durch die Eigenschaft „power" kontrolliert werden kann [1]. Die Bewegung eines Motors auf eine spezielle Position wurde folgendermaßen umgesetzt. Der jeweilige Motor wird so lange in die entsprechende Richtung bewegt, bis die angestrebte Position erreicht ist. Dann wird er gestoppt. Unter diesen Voraussetzungen ist festzustellen, dass die LEGO-Motoren, gerade bei höheren Drehzahlen, nicht an der exakten Position zum Stehen kommen. In Folge wird die Zielposition leicht übertreten.
Dieses Problem löst eine Funktion im Code, welche die Drehzahl und Drehrichtung eines Motors abhängig von der Differenz zwischen der angestrebten und aktuellen Position bestimmt. Bei Annäherung an die angestrebte Position wird dabei die Geschwindigkeit verringert. Die Funktion nimmt verschiedene Parameter auf, die das Verhalten der Motoren beeinflussen. Hierzu zählen der anzusteuernde Winkel und der Toleranzbereich, also die Distanz, ab der die Drehzahl verringert wird. Auch die Normalgeschwindigkeit außerhalb dieses Bereiches und die Minimalgeschwindigkeit, auf die der Motor verlangsamt wird, wenn die Positionsdifferenz gegen Null geht. Aus verschiedenen Testläufen stellen sich die folgenden Parameterwerte als geeignet heraus.

Toleranz:	25 Grad
Maximalgeschwindigkeit („power"):	5
Minimalgeschwindigkeit („power"):	0

Die Geschwindigkeitskurve unter diesen Parametern ist in Abbildung 7 festgehalten. Bei diesem Ansatz zeigt sich in der Praxis aber auch eine nennenswerte Auffälligkeit. Aufgrund der geringen Minimal- und Maximalgeschwindigkeit ist die berechnete Geschwindigkeit in einem Differenzbereich von mehreren Grad zu gering, um von den LEGO-Motoren umgesetzt zu werden. Dies hat zur Folge, dass die Motoren innerhalb dieses Bereiches stillstehen und sich nach Verlassen recht ruckartig in Bewegung setzen. Dies verringert die Genauigkeit und Reaktivität des Arms, aber filtert gleichzeitig kleine, ungewollte Bewegungen heraus.

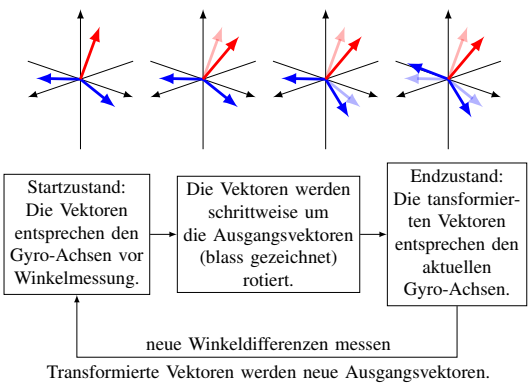


Abbildung 6: exemplarischer Ablauf der Vektortransformation

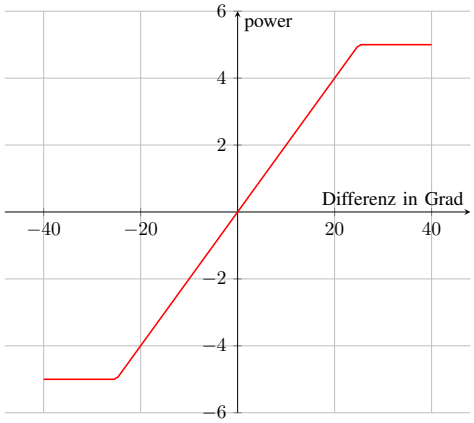


Abbildung 7: Motorverhalten abhängig von der Nähe zur Zielposition

IV. Ergebnisdiskussion

Die Umsetzung des Konzeptes ist gelungen. Der im Rahmen des Projektseminars entwickelte Roboter erfüllt seine vorgesehene Funktion, die Orientierung des Control-Sticks im Raum anzusteuern. Aber es zeigen sich funktionale Limitierungen, die durch die LEGO-Bauteile bedingt sind. Ein zentrales Problem besteht in der Genauigkeit der Positionserfassung des Control-Sticks. Die Ausführungsgeschwindigkeit des MATLAB-Skripts auf dem EV3 ist zu gering, als dass eine exakte Positionserfassung über Zeiträume mehrerer Minuten möglich ist. Sind die gemessenen Winkeldifferenzen größer, so ist die Transformation der Vektoren ungenauer. Bei zu großen Zeitabständen zwischen den Messungen und zu schnellen Bewegungen des Control-Sticks, kann dieser Effekt bereits nach Sekunden spürbar werden. Daher sind langsame und gleichmäßige Bewegungen des Control-Sticks notwendig. Weitere Problemstellen sind die Latenz, mit welcher der Roboter auf Eingaben reagiert und Einschränkungen in der Beweglichkeit durch die kurzen und unflexiblen Kabel. Durch Optimierungen in der Konstruktion und Programmierung funktioniert der Roboter dennoch in ausreichendem Maße. Das Ergebnis zeigt, dass das Konzept intuitiv bedienbarer Robotik umsetzbar ist und bietet einen Anhaltspunkt für weitere Entwicklung.

V. Zusammenfassung und Fazit

Der Roboter-Arm verfügt über 3 Bewegungsachsen und ist so in der Lage jegliche Orientierung des Control-Sticks nachzustellen. Die Steuerung erfolgt über einen LEGO EV3, der ein MATLAB-Skript ausführt. Hierbei werden Sensordaten eingelesen, verarbeitet und schließlich in Bewegungsanweisungen umgesetzt. Eine eindeutige Erfassung der Positionierung wird durch Transformation von Vektoren erzielt. Die Motoren werden durch eine Kontrollfunktion gesteuert, welche die Zielgenauigkeit erhöht. Auch wenn die LEGO-Bauteile Begrenzungen in der Leistungsfähigkeit mit sich bringen, erfüllt der Roboter seine Anforderungen und stellt eine Möglichkeit menschlich gesteuerter Robotik dar. Weitere Entwicklungsschritte können die Verwendung einer leistungsstärkeren Steuereinheit und eine kabellose Datenübertragung der Sensordaten umfassen. Der Roboter ist als eine Basis konzipiert, die durch verschiedenste Instrumente erweitert werden kann. Denkbar sind etwa eine Kamera und andere Sensorik oder eine Greifzange. Weitergedacht kann diese Technologie Menschen helfen, ihnen unzugängliche Räume zu erkunden, wie die Tiefsee oder Objekte im Weltraum. Sie ermöglicht es, Handlungen in großen Entfernungen auszuführen, die Erfahrung und Urteilsvermögen eines Menschen erfordern. Somit bietet das hier vorgestellte Projekt nicht nur eine technische Lösung, sondern öffnet auch Perspektiven für weiterführende Anwendungen an der Schnittstelle zwischen Mensch und Roboter.

Literaturverzeichnis

[1] Atorf, L. ; Sondermann, B. ; Stadtmann, T. ; Rossmann, J.: *RWTH - Mindstorms EV3 Toolbox*. https://git.rwth-aachen.de/mindstorms/ev3-toolbox-matlab. Version: 2018

[2] Wikipedia, the free encyclopedia: *Lego Mindstorms EV3*. https://en.wikipedia.org/wiki/Lego_Mindstorms_EV3. Version: Februar 2025

[3] Vetalru: *Lego mindstorms ev3*, Oktober 2013. https://commons.wikimedia.org/w/index.php?curid=28877049. – CC BY-SA 3.0, Bild unverändert, Abruf: 26.02.2025

[4] Jim McTurbo: *EV3 Gyro Sensor*, Juni 2020. https://commons.wikimedia.org/w/index.php?curid=91295885. – CC BY-SA 4.0, Bild unverändert, Abruf: 26.02.2025

[5] Wikipedia, the free encyclopedia: *Rodrigues' rotation formula*. https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula. Version: Februar 2025

[6] Wikipedia, the free encyclopedia: *Winkelgeschwindigkeit*. https://de.wikipedia.org/wiki/Winkelgeschwindigkeit. Version: Februar 2025

Projekt MoJo

Joe-Marco Sperling, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In dieser Arbeit wird die Zusammenarbeit zwischen Mensch und Maschine untersucht. Mit theoretischem Wissen aus den Anfängen des Seminars wird ein Roboterarm gebaut und programmiert, dessen Aufgabe es ist, einem Control-Stick zu folgen und dabei möglichst präzise zu sein. Die Umsetzung dieses Projekts gelingt mit Hilfe der LEGO-Mindstorms-Klemmbausteine und dem Programm MATLAB, in dem der Code für den Roboterarm geschrieben wird. Der Arm folgt hierbei den am Control-Stick angebrachten Gyro-Sensoren von LEGO. Diese Sensoren senden Daten an den Code, der daraufhin die Motoren am Arm steuert. Im weiteren Verlauf der Arbeit werden die auftretenden Probleme und deren Lösungen detailliert erläutert. Zudem werden exemplarische Code-Schnipsel vorgestellt, die die Funktionsweise des Systems verdeutlichen und zur Lösung der Herausforderungen beitragen.**

***Schlagwörter*—Gyro-Senoren, LEGO-Mindstorms, MATLAB, Mensch-Maschine Interaktion, Przisionssteuerung, Robotik.**

I. Einleitung

IM Verlauf des LEGO-Mindstorms-Projekt-Seminars war die Aufgabe, einen Roboter zu entwickeln. Unter vielen Ideen wird ein Roboterarm ausgewählt, der einem Gyroskop folgt. Der Arm, der in Abb. 1 zu sehen ist, kann beispielsweise verwendet werden, um einen Raum zu scannen, wenn eine Kamera daran befestigt ist, oder den Raum zu erhellen, wenn eine Taschenlampe angebracht wird. Damit der Arm funktioniert, müssen die Gyroskope ausgelesen und in Daten umgewandelt werden, die von den Motoren ausgeführt werden können. Dies geschieht in Echtzeit. Damit der Arm immer vom gleichen Ausgangspunkt startet, integriert man einen Kill-Switch, der das Programm abbricht und auf die Nullposition zurücksetzt. Zusätzlich muss man die Ungenauigkeiten, die durch die LEGO-Mindstorms-Bauteile entstehen, berücksichtigen und den Code entsprechend anpassen, obwohl auf Präzision großer Wert gelegt wird.

II. Vorbetrachtungen

In der Vorbetrachtung wird detailliert auf die auftretenden Probleme eingegangen. Zudem werden die Lösungen für diese Herausforderungen im Detail erläutert, um ein besseres Verständnis der Vorgehensweise zu ermöglichen.

A. Genauigkeit der Motoren

Die LEGO-Mindstorms Bausteine haben ein Problem mit der Genauigkeit, welches sowohl das Auslesen der Sensoren als auch deren Limitierungen betrifft, aber auch die Ausführung der Motoren betrifft. Die Motoren drehen sich dank des Brake-Modus „Brake“ bereits relativ genau, jedoch war die Präzision nicht ausreichend, da die Zahnräder einen Großteil

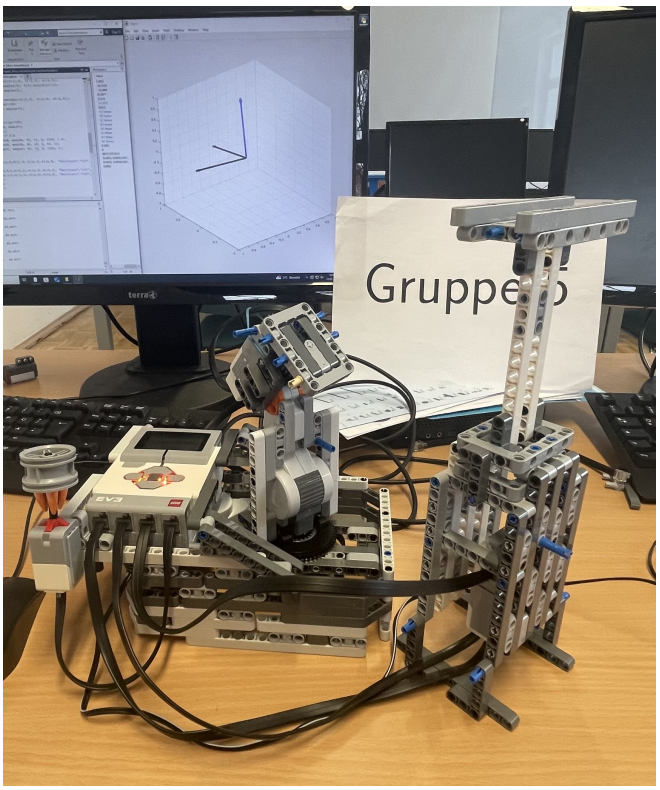


Abbildung 1. Fertiger Roboter mit fertigem Control-Stick

der Ungenauigkeit ausmachten. Aus diesem Grund wurde die Funktion der SpeedControl geschrieben, die den Motor verlangsamt, je näher man dem Zielwinkel kommt, das sieht man in der Abb. 2

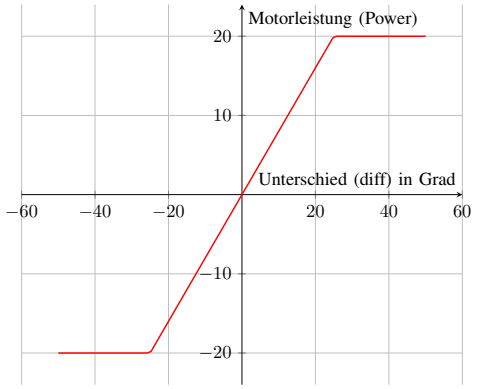


Abbildung 2. Diagramm der Motorleistung in Abhängigkeit von der Entfernung in grad.

DOI: 10.24352/UB.OVGU-2025-015

B. Genauigkeit der Sensoren

Wie in II-A erwähnt, gibt es auch Probleme mit den Sensoren. Die LEGO-Gyrosensoren haben das Problem, dass sie nur eine Rotationsachse wahrnehmen und anfangen können zu driften. Das größte Problem ist jedoch die Genauigkeit der Gradzahl, die sie ausgeben, da diese nur ganze Zahlen sein können. Für diese Probleme wurden jedoch schnell Lösungen gefunden. Wenn es nur eine Achse gibt, muss man einfach für jede Achse einen Sensor verwenden. Das Driften wurde durch die Bewegung des Sensors beim Einschalten verursacht, daher musste dieser nur stillgehalten werden. Für das letzte Problem wurden die Gyrosensoren auf Drehgeschwindigkeit umgeschaltet. Dies ermöglicht eine feinere und schnellere Anpassung der Motoren, um den Arm in Echtzeit zu steuern, ohne dass große Fehler durch Trägheit oder ungenaue Winkelmessungen auftreten.

C. Vektoren um Vektoren rotieren

Man rotiert Vektoren, um die Orientierung des Roboterarms basierend auf den Gyrosensorwerten zu berechnen. Da der Arm sich im 3D-Raum bewegt und die Sensoren die Bewegung entlang einer Achse messen, übersetzt man die Gyroskopdaten in Vektorbewegungen, um die Ausrichtung des Arms zu bestimmen. Diese Rotation ist notwendig, um die komplexen Bewegungen des Arms korrekt nachzuvollziehen und Fehler durch ungenaue Sensorwerte zu minimieren. Die Rotation der Vektoren ermöglicht es uns, die aktuelle Orientierung des Arms in Echtzeit zu berechnen und die Motoren präzise anzupassen. Die rotateVectors-Funktion verarbeitet die Gyroskopdaten, um die Armbewegung exakt zu steuern. Das ist in Abb. 3 zu sehen. Für die Transformation der Vektoren wird die „Rodrigues' Rotation Formula" verwendet, die im Abschnitt III-C gezeigt und erklärt wird. Ohne diese Formel wäre die Umsetzung nicht möglich gewesen.

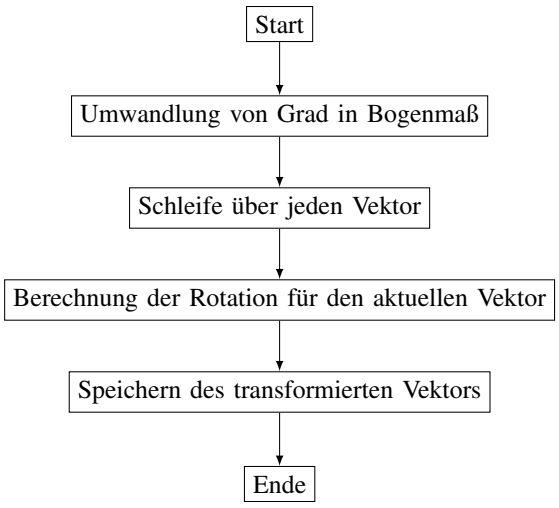


Abbildung 3. Programmablaufplan der Vektor Transformation

III. Umsetztung

A. Funktionsweise im Ganzen

Im Code werden die Werte der Gyrosensoren, die sich im Controllstick befinden, genutzt, um die Drehbewegung des Roboterarms in Echtzeit zu messen. Diese Messwerte werden in Rotationswinkel umgerechnet und dienen dazu, die aktuelle Ausrichtung des Arms zu bestimmen. Die Motoren werden dann entsprechend den berechneten Winkelabweichungen angepasst, um die gewünschte Position zu erreichen. Die Funktion SpeedControl sorgt dabei für eine präzise Steuerung der Motorleistung, indem sie den Unterschied zwischen dem aktuellen und dem Zielwinkel berücksichtigt.

B. Programmablaufplan

In Abb. 4 wird die Hauptschleife des Programms dargestellt und zeigt auch, was passiert, wenn die Abbruchbedingung erfüllt ist. Zusätzlich wird in diesem Diagramm visualisiert, welche spezifischen Schritte unternommen werden, um den Ablauf des Programms zu steuern und zu beenden.

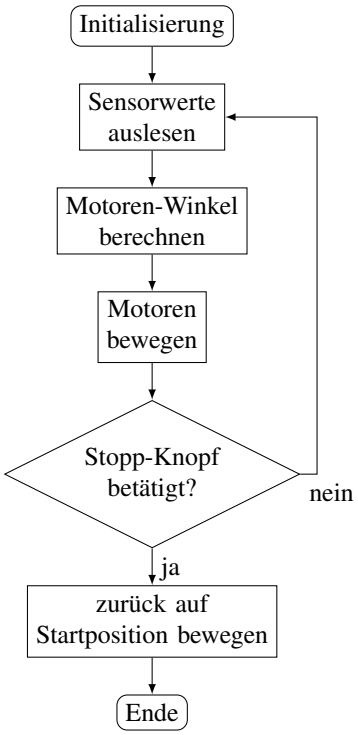


Abbildung 4. Programmablaufplan MATLAB-Skript

C. Gleichungen

Im Folgenden wird die Gleichung gezeigt und erklärt, die diesem Roboter ermöglicht haben seine Aufgaben auszuführen.

$$\mathbf{v}_{\text{neu}} = \mathbf{v}\cos\theta + (\mathbf{k}\times\mathbf{v})\sin\theta + \mathbf{k}(\mathbf{k}\cdot\mathbf{v})(1-\cos\theta)$$

hierbei gilt
$\mathbf{v}$ ist der ursprüngliche Vektor.
$\mathbf{k}$ ist der Einheitsvektor, der die Rotationsachse beschreibt.
θ ist der Winkel, um den der Vektor rotiert.
Die Formel ist unter der Quelle [1] zu finden

IV. Ergebnisdiskussion

Am Ende konnte die Idee erfolgreich umgesetzt werden und der Roboterarm funktionierte mit der gewünschten Funktionalität. Zwar brachten die LEGO Mindstorms-Bauteile einige Einschränkungen hinsichtlich Präzision und Rechenleistung mit sich, doch ließen sich diese durch kreative Lösungsansätze gut überwinden. Mit sorgfältiger Planung und der Anpassung der vorhandenen Ressourcen konnte die Leistung des Roboterarms optimiert werden.

In Abb. 5 ist ein früher Entwurf des Roboters zu sehen, der als Ausgangspunkt für die weitere Entwicklung dient. Dieser Entwurf stellt die Grundlage für den fertigen Roboter dar, dessen endgültige Version in Abb. 1 gezeigt wird. Während der Entwurf noch einige grobe Aspekte und Anpassungen aufwies, wurde er im Laufe der Zeit weiter verfeinert und optimiert, sodass er schließlich die gewünschte Funktionalität und Präzision erreichte.

Für den Control-Stick wurden zwei Varianten entwickelt, die in Abb. 6 zu sehen sind: eine mit einem großen Control-Stick und eine mit zwei kleinen Sticks. In Variante eins ist der Control Stick frei beweglich und enthält alle drei Gyrosensoren. In Variante zwei hingegen ist jeder Stick nur mit einem Gyrosensor ausgestattet, der jeweils nur eine Achse kontrolliert, ähnlich wie bei Controllern für Drohnen oder RC-Autos.

Variante zwei brachte jedoch Probleme mit sich, wie etwa eine mangelnde Kontrolle durch nicht wahrgenommene Bewegungen oder das Fehlen der dritten Achse, weshalb wir uns letztlich für Variante eins entschieden. Es zeigte sich, dass durch die richtige Wahl der Sensoren und Steuerungsstrategien selbst mit den begrenzten Möglichkeiten der LEGO Mindstorms-Technologie eine sehr leistungsfähige und präzise Steuerung des Arms möglich war. Dies beweist, dass auch innerhalb der Einschränkungen eines solchen Systems innovative Lösungen gefunden werden können, die diese überwinden.

V. Zusammenfassung und Fazit

Am Ende wurde ein voll funktionsfähiger Roboterarm entwickelt, der genau die beabsichtigte Aufgabe erfüllt. Es wurden alle Hürden überwunden und clever gelöst.
Um den Roboterarm zu erweitern, gibt es noch einige Ideen. Wenn man den Arm mit einer Webcam und Rädern ausstatten, könnte er einen Raum nach etwas absuchen, und der Control-Stick würde wegfallen. Eine weitere mögliche Weiterentwicklung wäre die Integration eines LiDAR-Sensors, um den Arm in die Lage zu versetzen, die Umgebung präzise zu scannen und Hindernisse zu erkennen. Dadurch könnte der Roboter autonom navigieren und Aufgaben in komplexeren Umgebungen ausführen. Allerdings wäre dies aufgrund der Limitationen der LEGO-Mindstorms-Bauteile, wie beispielsweise der begrenzten Rechenleistung und der fehlenden Sensoren für präzisere Bewegungen und Umwelterkennung, momentan nicht vollständig umsetzbar. Die beste Möglichkeit, den LiDAR-Sensor zu imitieren, wäre die Verwendung eines Ultraschallsensors, jedoch wäre auch dieser für komplexe Aufgaben aufgrund der eingeschränkten Präzision und Reichweite möglicherweise nicht ausreichend.

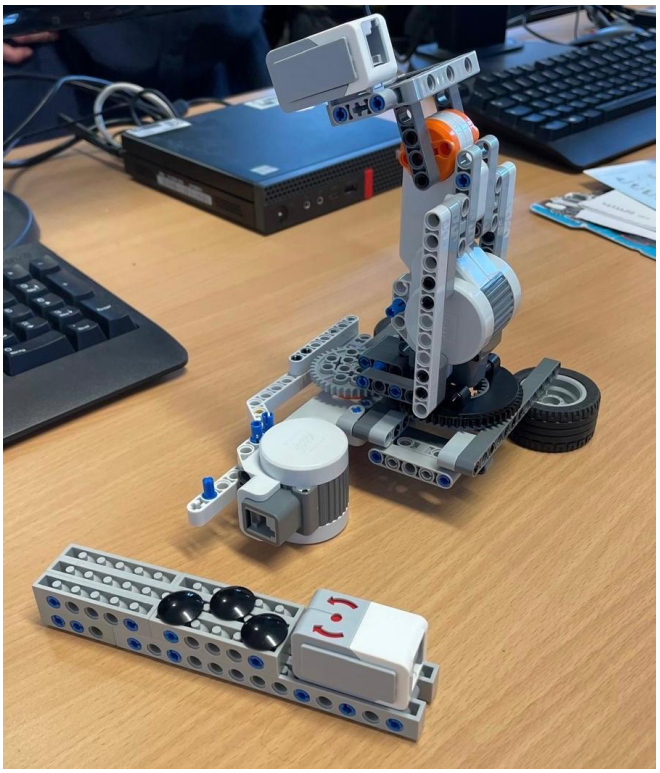


Abbildung 5. Früher Entwurf des Roboters und Control-Sticks

Abbildung 6. Zwei Varianten des Control-Sticks: Links Variante 2 und Rechts Variante 1

Literaturverzeichnis

[1] Wikipedia, the free encyclopedia: *Rodrigues' rotation formula*. https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula. Version: Februar 2025

Einen Abdruck hinterlassen – LEGO Drucker

Marvin Adam, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

Im Rahmen des LEGO-Praktikums 2025 der Otto-von-Guericke-Universität Magdeburg wurde ein 3-Farben-Drucker aus LEGO-Komponenten entwickelt. Ziel war es, einen funktionsfähigen, Drucker zu entwerfen und zu bauen. Dieser kann mit einem speziellen Farbauswahlmechanismus in drei Farben drucken. Die Steuerung erfolgte über MATLAB mithilfe der LEGO Mindstorms Toolbox. Damit war es möglich die verschiedenen Motoren und Sensoren der LEGO Mindstorms Sets anzusteuern und zu benutzen. Herausforderungen im Bau und der Software wurden analysiert und gelöst. Der Drucker ermöglicht mit seinen stabilen und aneinander angepassten Komponenten das solide Drucken mit Filzstiften. Die Aufgabe verbindet wichtige ingenieurswissenschaftliche Disziplinen, wie die praktische Arbeit und den theoretischen Grundlagen.

anspruchsvoll, Leistung, LEGO-Mindstorms, praktisch, zeitaufwendig

I. Einleitung

LEGO Mindstorms ist eine Produktreihe von LEGO. Im Mittelpunkt steht immer noch das klassische, kreative Bauen mit Klemmbausteinen, allerdings erweitert mit den Möglichkeiten eines programmierbaren Steins, dem EV3 und einigen Sensoren und Motoren. Dieser EV3 ist in der Lage bis zu je vier Motoren und Sensoren anzusteuern. So werden die kreativen Möglichkeiten denkbar vervielfacht und Programmieren kann näher und greifbarer beigebracht werden [1].

Innerhalb des Projektseminars Elektrotechnik/Informationstechnik (LEGO Mindstorms) wurde die Aufgabe gestellt ein Gerät, mit mehr oder weniger praktischer Anwendung, zu bauen und anschließend in MATLAB zu programmieren. Genutzt wurden dafür LEGO Mindstorms Basis- und Bonus-Sets. Programmiert wurde mithilfe der LEGO Mindstorms Toolbox in MATLAB. So war es möglich die verschiedenen Motoren und Sensoren zu benutzen. Angefangen hat das Seminar mit einem Einstiegskurs in MATLAB. Anschließend gab es einige Aufgaben, um mit der Toolbox umgehen zu lernen. Einfaches benutzen der Sensoren und Motoren in Kombination mit LEGO-Steinen gehörte dazu. Hier steht das Projekt „Drucker“ im Mittelpunkt. Dieser hatte, innerhalb der Begrenzungen von LEGO, einen praktischen Nutzen. Er war sowohl nicht einfach zu bauen als auch schwer zu programmieren. Die große Herausforderung bestand darin alles in dem gegeben Zeitrahmen zu bewältigen. Zusätzlich sollte auch ein Social Media Account erstellt werden. Ob Instagram, LinkedIn oder ähnliches war dabei freigestellt. Auf dem Instagram Kanal „`legoovgu256`“ [2] kann man verschiedene Tagesaufgaben, lustige Memes und den Entwicklungsvorgang sehen und nachvollziehen.

II. Bestandteile des Druckers

Der Drucker besteht aus vielen einzelnen Komponenten. Damit sind nicht die einzelnen LEGO-Steine gemeint, sondern viele verschiedene Einheiten von Teilen, die zusammen dieses Gerät ergeben. Im gröbsten kann man diese in Basis und Kopf aufteilen.

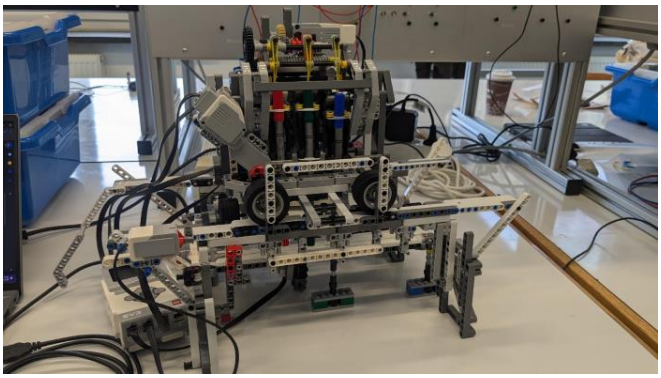


Abbildung 1: fertiger Drucker

A. Basis

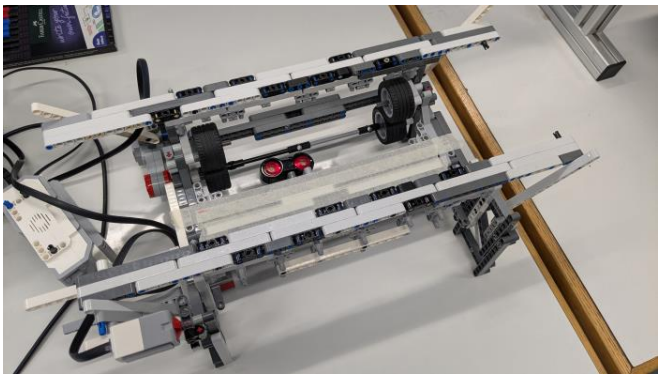


Abbildung 2: Die Basis

Die Basis ist das stabile Fundament. Auf ihr sitzt der spätere Druckkopf. Sie besteht aus zwei Rahmenkomponenten, die über zwei „Querstreben“ miteinander verbunden sind und stabil

DOI: 10.24352/UB.OVGU-2025-016

zusammengehalten werden. Die beiden Streben wurden einmal hinten und einmal vorne/mittig angebracht. Die mittige Strebe hat dabei gleich mehrere Funktionen. Einmal dient sie der Stabilität, anderseits auch ist sie auch etwas breiter, da sie als Ablage und Gegenkraft für das Papier dient, welches an dieser Stelle bedruckt wird. Dieses Gerüst ist so hoch, dass in entsprechender Höhe zwei Schienen quer sitzen, auf denen der Druckkopf liegt. Diese bieten zusätzliche Stabilität. Da sie zu von einem Rahmen zum anderen gespannt werden, gab es das Problem, dass sie in der Mitte durchhingen. Dieses wurde mit ausreichend Verstärkung und Verbindungen minimiert. Die größte Herausforderung in Bezug auf die Basis bestand ohnehin darin für ausreichend Stabilität zu sorgen. Aufgrund der Eigenschaften von LEGO und dem sehr massiven Druckkopf, war es sehr schwer den Zusammenhalt der Steine zu gewährleisten und mögliches wackeln und spiel zu minimieren.

Zur Basis gehören nicht nur klassische LEGO-Bausteine, sondern auch die speziellen LEGO-Mindstorms-Motoren und Sensoren. Einige finden sich auch in der Basis. Unter der hinteren Strebe ist eine Achse eingebaut, die ebenfalls quer liegt und an beiden Enden mit je einen Reifen verbunden ist. Diese Achse ist an einen Motor angeschlossen. Über diesen Rädern sind, mit minimalsten Abstand, aber noch ausreichender Berührungsfläche, weitere bewegliche Räder angebracht. Wenn der Motor die Achse bewegt, bewegen sich jetzt die beiden unteren Räder. Diese bewegen, aufgrund der Reibung, die oberen beiden Räder. So funktioniert der Papiereinzug. Aufgrund des speziellen Abstandes der Räder mussten die beiden Rahmenkomponenten mehrmals angepasst werden und in sehr spezieller Form gebracht werden. Dieser Mechanismus hat einerseits die Aufgabe das Papier zu fixieren und andersseits hat es in Kombination mit den anderen Sensor der Basis und dem Kopf die Position der zu bedruckenden Stelle zu erreichen, halten und anzupassen. Der Sensor, der in der Basis verbaut ist, ist ein Ultraschallsensor. Dieser ist kurz vor der mittleren Strebe eingebaut und an ihr befestigt. Er zeigt nach oben und misst die Entfernung des nächstgelegenen Objektes. Die meiste Zeit ist das die Decke vom Raum. Schiebt der Motor das Blatt in Richtung Druckfläche erkennt der Sensor wenn das Blatt vor ihm liegt, da jetzt der gemessene Abstand kleiner ist. Nach kurzer wird der Motor angehalten. Ursprünglich war dafür ein Farbsensor vorgesehen, dieser hat allerdings nicht die gewünschte Wirkung gezeigt und somit wurde sich für den Ultraschallsensor entschieden.

Mit dieser Basis ist es möglich, dass alles überhaupt erst funktionieren kann.

B. Druckkopf

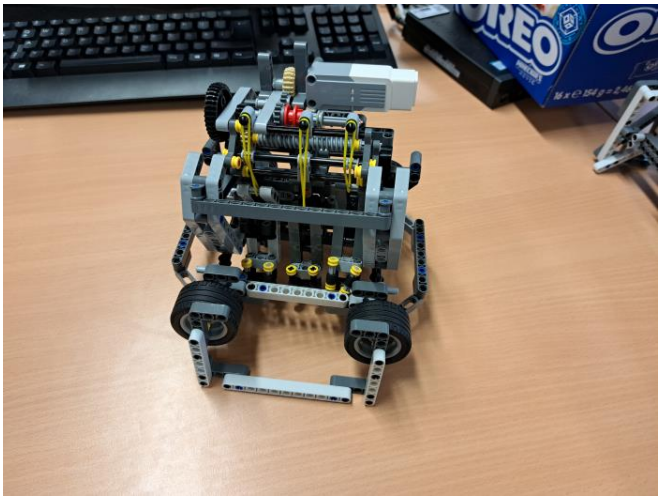


Abbildung 3: Druckkopf ohne Stifte

Der Druckkopf ist, wenn auch optisch kleiner, deutlich kompakter und etwas schwerer als die Basis. Hier ist der größte Teil der Motoren eingebaut. Auch wenn er nicht so aussieht ist auch er eine zusammenfuhr vieler einzelner Komponenten, die zusammen und perfekt abgestimmt, in der Lage sind drei verschiedene Farben auf das Papier zu bringen.

Der Teil, welcher auf den Schienen sitzt ist der Schlitten. Er ist ein Gerüst mit vier Rädern und zwei Motoren. Er wird mit den Rädern auf die beiden parallelen Schienen gestellt. Ähnlich wie schon bei der Basis sind zwei der Räder über eine Achse mit einem Motor verbunden. Dieser Motor ist der Antrieb. Er bewegt den ganzen Druckkopf, um die richtige Position der Stifte einzustellen. Der zweite Motor hebt den beweglich befestigten Rest des Kopfes hoch und runter. Das hat einmal den Sinn den Kopf und damit auch die Stifte hochzuhalten, wenn sie nicht gebraucht werden oder sich der Kopf zur nächsten Position bewegt, sodass sie nicht versehentlich einen Abdruck auf dem Papier hinterlassen. Der Motor hat auch die umgekehrte Funktion den Kopf in der richtigen Situation herunterzulassen, damit die Stifte an der richtigen Stelle den Pixel drucken können.

Der restliche Druck Kopf hat die Stifte und den Stiftauswahlmechanismus. Die Stifte, in diesem Fall Filzstifte, liegen an ihrer entsprechenden Stelle an einer LEGO Stange. Mithilfe von Gummibändern und einen weiteren Gummiteil von LEGO werden sie festgemacht und auf eine passende Höhe eingestellt. Diese drei Vorrichtungen sind ebenfalls beweglich, mit etwas weniger elastischen Gummi an eine Achse angebracht. Im Normalfall würde bei dieser Bauweise kein Stift einen Abdruck auf dem Papier hinterlassen, da sie so auch mit dem Motor zum Herunterlassen zu hoch wären. Hier kommt der letzte Motor, den wir noch nutzen können, ins Spiel. Über der Achse mit den Stifthalterungen ist eine weitere Achse eingebaut. Diese Achse ist über ihre volle Länge mit aneinander anschließenden Spiralen bestückt. Diese Achse ist an den vierten, und aus Platzgründen kleineren, Motor angebracht.

Zusätzlich wird in das Konstrukt ein dickeres Zahnrad im 90 Grad Winkel, mit verlängerten und nach unten zeigenden Kolben, gedrückt. Dreht sich jetzt der Motor dreht sich die ganze Achse mit den Spiralen und schiebt das Zahnrad in die entsprechende Richtung. Der sich bewegende Kolben drückt die gewünschte Halterung nach unten. Der heruntergedrückte Stift ist so tief genug, um einen Abdruck zu hinterlassen, während die anderen beiden Stifte dafür zu hoch liegen. All diese Komponenten zusammen haben dafür gesorgt, dass der Kopf in der Lage ist seine Aufgabe zu erfüllen.

III. Programmierung

Ein weiterer wichtiger Teil war natürlich die Programmierung. Diese war nicht einfach und ist sicherlich nicht perfekt, was in der relativ kurzen Zeit, auch nicht machbar ist. Der Code ist objektorientiert und gut strukturiert.

Ohne zu spezifisch zu werden funktioniert das Programm wie zu sehen ist in Abbildung 4.

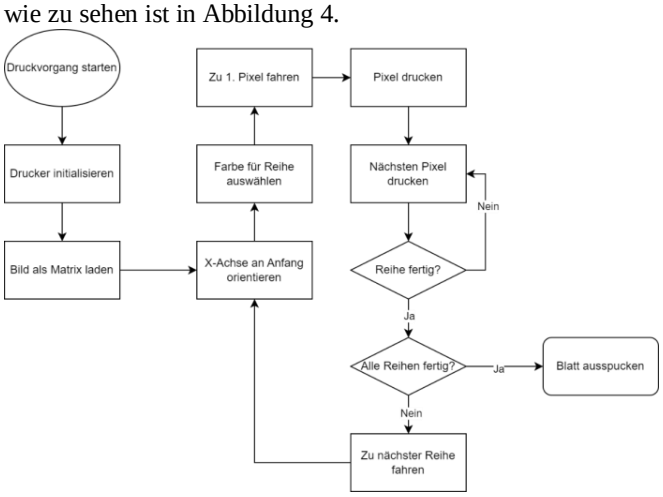


Abbildung 4: Programmablaufplan

Man muss zuerst ein Bild auswählen oder erstellen. Dieses muss eine 32×32-Pixel-Rastergrafik sein. Der Drucker wird initialisiert. Dafür gibt es eine festgelegte Reihenfolge von Aktionen, die ausgeführt werden, um sicherzugehen, dass alles auf der Ausgangsposition ist und funktioniert. Dazu gehört das Fahren des Kopfes in Richtung EV3 bis er gegen einen Knopf drückt, wodurch er einmal in die andere Richtung und wieder zurück fährt. Dann ist er an der richtigen Stelle. Der Stiftauswahlmechanismus wird auf ähnliche Art in Position gebracht. Hier handelt es sich nicht um einen physischen Knopf, der gedrückt wird, sondern um einen Farbsensor. Beim Initialisieren fährt der Kolben in Richtung Sensor. Sobald der Sensor die Farbänderung sieht fährt der Kolben in die entgegengesetzte Richtung und wieder zurück. Damit steht der Kolben an der Ausgangsposition und es wurde getestet, ob die Stifthalterungen halten oder sich der Kolben irgendwie verklemmt.

Nun wird das Bild mithilfe eines passenden MATLAB-Befehls als Matrix geladen. An der Stelle musste aufgrund eines zu hohen Zeitaufwandes festgelegt werden, dass es nur eine Farbe pro Reihe geben soll und kann. Ständiges wechseln der Stifte würde zu viel Zeit in Anspruch nehmen. Theoretisch wäre es möglich das zu machen, es ist aber im Code nicht implementiert.

Nun wird ausgehend vom beschriebenen Ausgangspunkt die Farbe für die Reihe ausgewählt. Der Kopf fährt jetzt von zu jeden einzelnen Pixel und entscheidet, ob ein Punkt gesetzt werden muss. Wenn ja wird dieser gesetzt, ansonsten hält er trotzdem kurz an, bewegt sich aber nicht nach unten. Ist der Punkt gesetzt oder nicht wird zur nächsten Position gefahren und erneut entschieden. Wenn alle Pixel gesetzt sind ist die Reihe fertig. Sollte das nicht die letzte Reihe sein, beginnt der ganze Prozess von vorne. Der Kopf nimmt die Ausgangsposition ein, das Blatt wird eine Zeile hochgeschoben und so weiter. Sollten alle Reihen fertig sein wird der Vorgang beendet und der Rest vom Papier wird durchgezogen.

IV. Probleme der Konstruktion

Keine Konstruktion ist von Fehlern befreit, so auch diese. Hier liegen viele in den Limitierungen von LEGO.

Einmal ist da die grundsätzlich fehlende absolute Stabilität von LEGO-Steinen. Größtenteils hat die Basis mit diesem Problem zu kämpfen. Trotz vielen Verbindungen ist es fast unmöglich „LEGO vom Wackeln zu befreien“. Wenn sich der massive Kopf bewegt ist dieses Problem zu sehen. Auch wenn dies aufgrund der Konstruktionsart ein sehr kleines Problem ist, ist es doch erwähnenswert. Ähnlich ist es bei den Schienen. Diese müssen über große Distanzen aufgespannt liegen. Das führt dazu, dass sie in der Mitte dazu neigen durchzuhängen. Durch verschiedene Verstärkungen und die erhöhte Dicke konnte auch dieses Problem minimiert werden, bleibt aber bestehen.

Die kompakte Bauart des Kopfes sorgt dafür, dass mögliche oder nötige Veränderungen nur schwer zu beheben sind, da fast alles einmal auseinander gebaut werden müsste. Auch er ist nicht von LEGOs Limitationen befreit. Das größte Problem ist die Kabellänge. Aufgrund der vorgegebenen Kabel von LEGO ist es nicht möglich, dass der Kopf einmal die komplette Schienenlänge fahren kann. Der EV3 konnte an keiner passenden Stelle angebracht werden und stand deshalb seitlich nah an einem Ende der Schiene. Es ist daher nicht möglich die für den Kopf die andere Schienenseite zu erreichen, ohne dass das Kabel abbricht.

Es ist auch nicht möglich in mehr als drei Farben zu drucken, da es keine LEGO Achse gibt, die lang genug ist, um noch mehr Spiralen aneinander zu setzen

Ein weiteres Problem ist der hohe Zeitaufwand des Druckvorgangs. Um den einfachen Text „RIP Fax“ und dazu noch eine kleine Zeichnung zu drucken brauchte das Gerät ungefähr 16 Minuten. Zur praktischen Anwendung ist der Drucker nicht geeignet.

Die Konstruktionsweise des Kolbens im Stiftauswahlmechanismus ist nicht sehr stabil. Es kann passieren, dass dieser rausspringt und der Mechanismus nicht mehr funktioniert.

V. Zusammenfassung

Die gestellte Aufgabe wurde erfolgreich umgesetzt. Der entwickelte Drucker erfüllt seine Funktion und bietet trotz allem eine praktikable Anwendung innerhalb der Möglichkeiten von LEGO Mindstorms. Die Konstruktion ist ausreichend stabil. Die Basis trägt das Gewicht des Druckkopfs zuverlässig und gewährleistet dessen präzise Bewegung. Der kompakte Druckkopf verfügt über einen effizienten Mechanismus zum Wechseln der Stifte, der in den meisten Fällen reibungslos funktioniert. Die größte Herausforderung stellte die Programmierung dar, die jedoch elegant gelöst wurde. Der entwickelte Algorithmus sorgt für einen strukturierten Ablauf und gewährleistet eine weitgehend fehlerfreie Funktionsweise.

Anhang

Hier, in Abbildung 6 ist ein kleiner Ausschnitt des Codes. Er soll einmal die Ordnung verdeutlichen. Er zeigt die Struktur des Programmes und den Anfang des Initialisierungsablaufes.

```
classdef Printer

    properties
        % Store reference to the EV3 toolbox API here
        brick = EV3()

        % Stores the currently selected color 1..3 or 0 if not initialized
        selectedColor = 0

        % Is 1 if one pen is currently down at the paper
        penDown = 1

        % Matrix to convert a tacho vector into a mm vector
        matTMM = [1,0,0,1]; %TODO Calculate & fill values
    end

    methods (Access = public)
        % Initialize internal stuff (e.g. USB setup) to be able to later
        % send commands to the printer
        function initialize(obj)
            % Cleanup old connections
            obj.brick.killAllConnections();

            % Old connections should be closed, but check just to make
sure
            % and reconnect
            if ~obj.brick.isConnected
                obj.brick.connect('usb');
            end
        end

        % Printer will turn motors until sensor detects sheet of paper
        % TODO: Handle paper already there (-> sensor) when func-
tion was
        % called
        function insertPaper(obj)
            obj.brick.motorA.setProperties('power', 50, 'brakeMode',
'Brake');
            obj.brick.motorA.start();
            obj.brick.sensor1.mode = DeviceMode.UltraSonic.DistCM;
            while obj.brick.sensor1.value > 8
            end
            obj.brick.motorA.stop();

            %TODO Insert paper a little more to move it to start position
```

Abbildung 6: Ausschnitt des Codes

[1] siehe Wikipedia: https://de.wikipedia.org/wiki/Lego_Mindstorms, abgerufen am 26.02.2025
[2] siehe Instagram, https://www.instagram.com/legoovgu256

Dreifarbiger LEGO®-Drucker

Finn Karstens, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Drucker sind aus dem modernen Leben kaum noch wegzudenken. Sie übertragen digitale Inhalte auf Papier und müssen dabei präzise und zuverlässig funktionieren. Naheliegend ist die Idee, eine ähnliche Maschine mithilfe der LEGO®-Mindstorms®-Plattform zu entwickeln. Als zusätzliche Herausforderung und Weiterentwicklung existierender LEGO®-Drucker wurde hier eine Farbwechselfunktion eingebaut, welche während des Druckes zwischen drei Farben wechseln kann. Die vorliegende Arbeit beschäftigt sich mit der Entwicklung eines solchen Prototypen. Sie dokumentiert wesentliche Design-Entscheidungen, stellt Programm-Abläufe dar und setzt sich mit der finalen Konstruktion auseinander. Dabei werden Herausforderungen wie Stabilität und Motoransteuerung weiter erläutert und mögliche Lösungen dargestellt.**

***Schlagwörter*—Drucker, Farbwechsel, Motoransteuerung, Rastergrafik, Vektorgrafik**

I. Einleitung

DRUCKER gehören wohl mitunter zu den meist gehassten Maschinen in den Büros dieser Welt. Unzuverlässigkeit, Papierstaus und Probleme mit der Tinte sind häufige Unannehmlichkeiten [1]. Aber wie kompliziert ist es wirklich, beliebige digitale Inhalte von einem Computer auf ein Blatt Papier zu übertragen?
Das Ziel, in unter zwei Wochen ein Gerät aus Klemmbausteinen zu entwickeln, das vergleichbar mit kommerziellen Lösungen ist, wäre etwas hochgesteckt. Daher wurden einige Vereinfachungen festgelegt, die ebenfalls im Einklang mit den technischen Möglichkeiten der vorgeschriebenen LEGO®-Mindstorms®-Plattform stehen.

1) Sehr stark reduzierte Auflösung
2) Keine Mischung beliebiger Farben
3) Software-Umgebung auf MATLAB beschränkt
4) Keine Scan-Funktion

II. Vorbetrachtungen

Die Idee, einen Drucker aus LEGO®-Steinen zu bauen, ist schon seit den Anfängen der LEGO®-Mindstorms®-Produktreihe vorhanden [2]. In bekannten Online-Portalen gibt es diverse Videos und andere Inhalte, in denen Leute ihre Ansätze und Umsetzungen präsentieren, z. B. [3], [4].
Hier war der Anspruch, einen Schritt weiter zu gehen, um so viel wie möglich in der gegebenen Zeit aus den Materialien herauszuholen.

A. Bereits existierende Umsetzungen

Im Internet existieren bereits viele Videos und Bauanleitungen zu LEGO®-Druckern. Dabei gibt es hauptsächlich zwei verschiedene Design-Konzepte.

DOI: 10.24352/UB.OVGU-2025-017

1) Roboter-Arm: Diese Variante basiert auf einer Art Greifarm, der sich zu jedem beliebigen Punkt einer Ebene bewegen kann. Dies geschieht in einer 2D-Ebene mit einem zusätzlichen Motor für das Heben und Senken des Stifts. Der Ansatz ist besonders gut geeignet für das Zeichnen von runden Formen, da dies der natürlichen Bewegung der Motoren entspricht. Allerdings kommt es zu erheblichen Hebelwirkungen, falls das Bild, das gezeichnet werden soll, größer als ein bestimmter Bereich ist. Dies hat negative Auswirkungen auf die Präzision und das Endergebnis.

2) X-Y-Drucker: Dieser Ansatz entspricht einem handelsüblichen Drucker am ehesten. Hierbei wird ein Blatt durch einen Papiereinzug in y-Richtung hin und her bewegt und ein Druckkopf bringt die Farbe in x-Richtung am jeweiligen Punkt auf das Papier.

Letztendlich wurde als Grundlage der x-y-Drucker gewählt, da er einem kommerziellen Gerät am ehesten entspricht und sich verhältnismäßig einfach ansteuern lässt. Durch diese Wahl konnte der Fokus auf andere Probleme, wie Farbwechsler und Stabilität, gerichtet werden.

B. Zusätzliche Herausforderungen

Nahezu alle der bereits existierenden LEGO®-Drucker haben eine Gemeinsamkeit: Sie können lediglich eine Farbe darstellen. Da nur 3 Motoren für die Funktionalität eines einfachen Druckers benötigt werden, steht bei der Verwendung eines LEGO® Mindstorms® EV3s mit 4 Motor-Anschlüssen noch ein weiterer Motor für die Umsetzung eines Farbwechsel-Mechanismus zur Verfügung. Dieser Mechanismus muss also mit einem Motor auskommen, aber auch robust und mit möglichst geringen Toleranzen funktionieren, da sonst die Positionierung des Stiftes unzuverlässig ist. Gerade bei der Verwendung von Klemmbausteinen sind Toleranzen und Spiel im Mechanismus ein besonders häufiges Problem, das spezielle Bautechniken zur Stabilisierung erfordert [5].
Ein weiterer Faktor, der die Entwicklung beeinflusst, ist die Verwendung von Vektor- oder Rastergrafiken. Die Umsetzung des Druckers ist hier bewusst allgemein gehalten, sodass sowohl das Drucken von Vektor- als auch Rastergrafiken möglich sein soll.

III. Umsetzung

Der Prozess, einen Drucker zu bauen und zu programmieren, wurde in mehrere Unterschritte eingeteilt. Einerseits wurde beim Bau des Druckers zwischen Basis und Druckkopf unterschieden. Die Ergebnisse der parallelen Arbeit wurden dann später zusammengeführt. Andererseits wurde erst mit der Programmierung begonnen, als bereits ein erster Prototyp

fertiggestellt war. Dadurch konnte dieser dann validiert und im Nachhinein noch etwas weiter angepasst werden.

A. Basis

Die Basis bildet das Grundgerüst des Druckers. Sie muss stabil sein und darf nicht schwingen, damit der Druckkopf auf den dafür vorgesehenen Schienen fahren kann, ohne dass es dabei zur ungewollten Verschiebung der Position der Stifte kommt. Außerdem verfügt sie über einen Motor, der für den Papiereinzug und die Bewegung entlang der y-Achse zuständig ist. Wie in Abbildung 1 zu sehen, ist ein zusätzlicher Ultraschall-Sensor vorhanden, um zu erkennen, ob ein Blatt erfolgreich eingezogen wurde. Dadurch kann das Papier in die richtige Position zum Drucken gebracht werden und es wird verhindert, dass der Drucker sich selbst bedruckt.

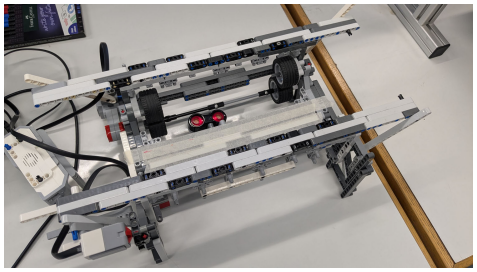


Abbildung 1. Konstruktion der Basis des Druckers

Ein zusätzlicher Drucksensor ist für die Erkennung der Nullposition der x-Achse zuständig und wird für die Kalibrierung benutzt. Diese erfolgt wie in Abbildung 2 dargestellt. Dabei fährt der Drucker von einer beliebigen Startposition aus immer in die Nullposition und dann die maximal mögliche x-Strecke und wieder zurück zur Nullposition. Sollte es dabei zu einem Problem kommen, wird dies noch vor Beginn des eigentlichen Druckes erkannt und kann behoben werden.
Durch die Kalibrierung wird sichergestellt, dass der Drucker nachfolgend immer in gleicher Ausgangslage auf der x-Achse ist. Dies funktioniert trotz des Spiels der Klemmbausteine verhältnismäßig zuverlässig.

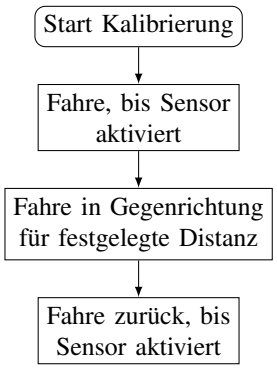


Abbildung 2. Programmablaufplan zur Kalibrierung einer Achse

B. Druckkopf

Das zentrale Element des Druckkopfes ist der Farbwechsler. Dieser wird in Abschnitt III-C detailliert betrachtet. Dazu kommt ein Hebe-Senk-Mechanismus, der die Höhe aller Stifte steuert. Somit wird sichergestellt, wann der aktive Stift aufdrückt oder nicht. Ein weiterer Motor steuert die Position des Druckkopfes entlang der x-Achse.
Abbildung 3 zeigt den fertigen Druckkopf.

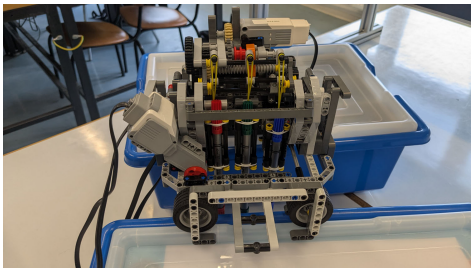


Abbildung 3. Konstruktion des Druckkopfes des Druckers

C. Farbwechsler

Das Merkmal, das den hier entwickelten Drucker von bereits existierenden Konstruktionen unterscheidet, ist der Farbwechsler. Dieser Mechanismus sorgt dafür, dass ein Stift von dreien als primärer Stift ausgewählt werden kann.
Dabei wird ein kleiner Schlitten durch eine Schneckenschraube an die jeweilige Position des Stiftes bewegt. Dies sorgt dafür, dass der Stift an der jeweiligen Stelle des Schlittens nach vorne und gleichzeitig nach unten gedrückt wird. An den Positionen, an denen der Schlitten nicht ist, klappen die Stifte automatisch wieder nach oben und hinten.

Damit der Drucker immer weiß, wo sich der Schlitten befindet, durchläuft auch der Farbwechsler zu Beginn des Druckes eine Kalibrierungsroutine. Diese verläuft analog zu Abbildung 2 und der in Abschnitt III-A beschriebenen Kalibrierung der x-Achse. Durch das Ausführen dieses Ablaufs wird sichergestellt, dass der Mechanismus funktionsfähig ist und sich in Nullposition befindet.

Die Wahl der Stifte ist eine weitere wichtige Entscheidung. Hier wurde sich für Filzstifte entschieden, da dort die Linienqualität weniger stark vom exakten Winkel abhängt, in dem der Stift aufkommt. Darüber hinaus ist die Spitze auch etwas flexibler, was Toleranzen, die durch die LEGO®-Konstruktion entstehen, wieder ausgleicht.

D. Codeaufbau in MATLAB

Den Kern der Software des Druckers bildet die EV3 Toolbox für MATLAB der RWTH Aachen. Diese erlaubt es, dem EV3 Befehle über eine aktive USB-Verbindung zu senden und dadurch Motoren und Sensoren anzusprechen. Der nötige Code zur Ansteuerung wurde in Funktionen geschachtelt und diese wiederum zu einer eigenen Klasse verbunden. Dies hat den Vorteil, dass sinnhafte Code-Abschnitte einfach wiederverwendet werden können und die konkrete Steuerung des Druckers an einem anderen, übersichtlicheren Ort stattfinden kann. Zu diesem Zweck wurde der Live-Editor von MATLAB verwendet. Dort können simple Code-Segmente erstellt werden, die dann Funktionen der Drucker-Klasse aufrufen. So ist es z. B. möglich, eine Sektion zu erstellen, die den Drucker initialisiert und eine

Linie druckt. Ein weiterer Vorteil dieser Struktur liegt in der Erstellung von Drucker-Skripten. Dies findet bei kommerziellen Druckern ebenfalls in Form von PostScript Anwendung [6]. Durch die Erstellung solcher Skripte gibt es einen einfachen Weg, die primitiven Funktionen, die durch die Drucker-Klasse bereit gestellt werden, zu komplexeren Abläufen zu vereinen. Ein Beispiel ist das Drucken eines Quadrats. Dafür werden lediglich Funktionen zum Heben und Senken des Druckkopfes, Farbauswahl und das Fahren entlang einer geradlinigen Bahn benötigt. Diese werden durch die Drucker-Klasse bereitgestellt.

E. Drucken einer Rastergrafik

Das Drucken von Rastergrafiken ist mit diesem Drucker im Vergleich zu Vektorgrafiken einfacher, da beide Achsen getrennt voneinander an die richtige Position fahren können. Dies vereinfacht die Motorsteuerung.
Dafür wird die gewünschte Grafik in MATLAB als Matrix geladen und richtig orientiert. Danach fährt der Drucker jeden Pixel, sprich jede Position der Matrix, welche auf das Blatt übertragen wird, ab und platziert dort einen Farbpunkt, wo der Farbwert des Pixels es vorsieht. Dies wird detailliert durch den Ablaufplan 4 beschrieben.
Anzumerken ist, dass die Initialisierung auch die Kalibrierungen beinhaltet, die in Abbildung 2 dargestellt und in den Abschnitten III-A und III-C näher erläutert werden. Ein wohldefinierter Ausgangszustand wird dadurch zu Beginn eines Druckes sichergestellt.

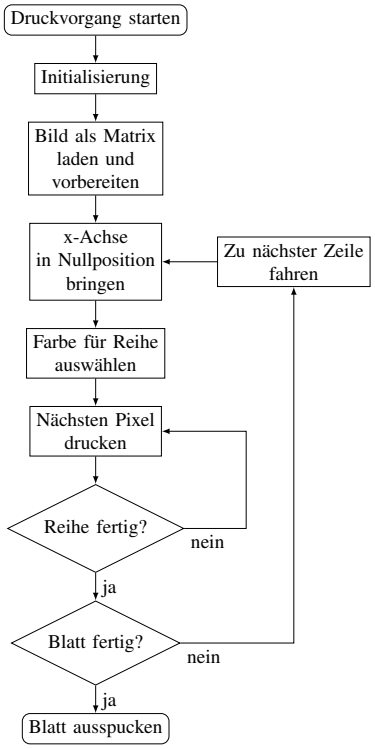


Abbildung 4. Programmablaufplan zum Drucken einer Rastergrafik

F. Drucken primitiver Formen

Neben dem Verarbeiten einer Rastergrafik ermöglicht die Verwendung eines Stiftes auch das Zeichen von Elementen einer Vektorgrafik. Die primitivste Form neben dem Punkt ist die Linie. Anstatt einen vollwertigen Parser für Vektorgrafiken in MATLAB zu schreiben, wurde zunächst nur die Implementierung dieser Linienfunktion in Betracht gezogen.
Für die genaue Kontrolle der Motoren ist eine erweiterte Motoransteuerung nötig. Diese wird genauer in Abschnitt III-G behandelt. In der Zeit des Seminars konnte keine Funktion zum Zeichnen von Geraden mit beliebigen Winkeln mehr fertigstellt werden. Die aktuelle Implementierung umfasst das Zeichnen von 0°-, 45°- und 90°-Linien. Damit können bereits Quadrate, Rechtecke und weitere primitive Formen mit 45°- und 90°-Winkeln gezeichnet werden. Außerdem wäre eine Darstellung von Zahlen denkbar, die einer Sieben-Segment-Anzeige gleicht. Jedoch konnte dies auch aus zeitlichen Gründen nicht mehr fertiggestellt werden.

G. Motoransteuerung

Bei der Motorsteuerung müssen einige Eigenheiten der verwendeten LEGO®-Mindstorms®-Plattform beachtet werden, um optimale Ergebnisse zu erzielen.

1) Brake-Mode: Der Brake-Mode definiert das Verhalten der Motoren, nachdem sie gestoppt wurden. Im Falle des Druckers ist es essenziell, dass die Motoren nach dem Stoppen ihre Position halten. Das ist zwar mit erhöhtem Energieaufwand verbunden, sorgt aber dafür, dass z. B. die Position des Druckkopfes nicht verfälscht wird. Deshalb muss der Brake-Mode auf „brake“ gestellt werden.

2) Synchrone Ansteuerung: Um eine gleichmäßige und vorhersehbare Bewegung der Motoren zu erhalten, ist es nicht ausreichend, beide Motoren für x- und y-Achse nacheinander mit jeweils einem Befehl zu starten. Durch die Latenz zwischen zwei Befehlen drehen beide Motoren nicht zur exakt selben Zeit und bewegen sich daher nicht so wie gewünscht. Deshalb müssen beide Motoren mittels Motorsteuerung synchronisiert werden. Die Firmware des EV3s stellt dafür eine Funktion zur Verfügung, die zwei Motoren mit unterschiedlicher Geschwindigkeit synchron ansteuern kann [7].
Diese Funktion benötigt einen zusätzlichen Parameter, `turnRatio`. Die Implementierung dieser Ansteuerung funktioniert dabei folgendermaßen: Der Hauptmotor bekommt Geschwindigkeit, Anzahl an Umdrehungen, den Nebenmotor und `turnRatio` als Parameter. So lange sich der Hauptmotor dreht, wird sich der Nebenmotor ebenfalls drehen und nimmt dabei die Geschwindigkeit `turnRatio` an. Zusätzlich kann durch Vorzeichen und Addition/Subtraktion von 100 die Richtung beider Motoren gesteuert werden.
Bei einem symmetrischen Aufbau der Motoren ist diese Variante gut geeignet, um eine einfache Lenkung zu realisieren. Da im Fall des Druckers der Aufbau aus Platzgründen nicht symmetrisch ist und eine Zahnrad-Übersetzung vorhanden ist, erschwert dies allerdings die Motoransteuerung. Drehen sich beide Motoren gleich schnell und viel, entsteht im Normalfall eine 45°-Linie. Um dieses Ergebnis hier zu erreichen, muss die entsprechende `turnRatio` mit gegebener Formel (2) errechnet werden. Das benötigte Übersetzungsverhältnis (1) wurde mit einem Online-Tool berechnet [8].

$$r_x = 0{,}6 \tag{1}$$

$$t = (1 - r_x) \cdot 100 \tag{2}$$

Zur Umrechnung von der Distanz in mm in Tacho-Einheiten, mit denen die EV3 Toolbox arbeitet, wurde im MATLAB-Stil eine Umrechnungs-Matrix verwendet. Der Vektor mit Maßangaben in mm wird dabei durch eine Matrix-Multiplikation mit der Umrechnungsmatrix A (4) in Grad-Motorumdrehung umgerechnet. Die Konstante (3a) gibt den Durchmesser des verwendeten LEGO®-Reifens an [9]. Die experimentell bestimmte Kompression der Gummi-Reifen durch Eigengewicht wird durch (3b) und (3c) berücksichtigt. Der resultierende Vektor enthält die nötigen Motorumdrehungen in Grad, mit denen die angegebene Strecke trotz Berücksichtigung der speziellen Motoransteuerung zurückgelegt werden kann.

$$d = 43{,}2\,\mathrm{mm} \tag{3a}$$
$$c_x = 0{,}955\,\mathrm{mm} \tag{3b}$$
$$c_y = 0{,}0\,\mathrm{mm} \tag{3c}$$

$$A = \begin{pmatrix} \frac{360^\circ}{\pi \cdot (d - c_x)} \cdot \frac{1}{r_x} & 0 \\ 0 & \frac{360^\circ}{\pi \cdot (d - c_y)} \end{pmatrix} \tag{4}$$

Um grobe Ungenauigkeiten und stärkere Schwingungen zu vermeiden, wurde nach verschiedenen Tests die Leistung der Motoren vorerst auf 35% begrenzt.

IV. Ergebnisdiskussion

Trotz der verhältnismäßig kurzen Entwicklungszeit hat der Drucker brauchbare Ergebnisse produziert, die zusätzlich durch Zeitraffer-Aufnahmen festgehalten werden konnten. Die Ursachen für die momentanen Limitierungen sind Zeitmangel und die LEGO®-Mindstorms®-Plattform selbst. Mit etwas mehr Zeit und Budget würden sich Probleme wie Kabellänge und Stabilität bzw. Toleranzen der Konstruktion sicherlich noch weiter optimieren lassen. Ein weiteres Problem ist die Fixierung des Blattes. Durch das momentan lose Aufliegen entsteht ein kleiner Luftspalt zwischen Blatt und Basis, der beim Aufdrücken des Stifts für eine leichte Vorwärtsbewegung sorgt. Dies könnte durch eine Weiterentwicklung der Basis verbessert werden.
Außerdem könnte eine Verbesserung des Farbwechslers nicht nur die Zuverlässigkeit beim Wechseln der Farbe erhöhen, sondern auch die Höhe der Stifte fixieren. Dadurch könnte sichergestellt werden, dass alle Farben gleich stark aufdrücken, was momentan nicht der Fall ist. Verbesserungen des Codes könnten zudem das Zeichnen von Geraden mit beliebigem Winkel sowie von runden Formen ermöglichen. Auch könnte die Druckzeit noch durch einige Optimierungen bei der Bewegung verringert werden. Ein weiterer Drucksensor am anderen Ende der Schiene könnte die Maßgenauigkeit während des Druckes noch erhöhen. Da die Gesamtkonstruktion jedoch eine gewisse Größe erreicht hat, war die verfügbare Kabellänge nicht ausreichend.

V. Theoretische Weiterentwicklung

Mit mehr Zeit und Ressourcen könnte man zunächst die in Abschnitt IV angesprochenen Verbesserungen durchführen. Wie in Abschnitt III-F bereits erwähnt, arbeiten kommerzielle Drucker hauptsächlich mit PostScript [6]. Dies und ein Vektorformat (z. B. SVG) könnten unterstützt werden und somit eine deutlich bessere Anbindung an bereits etablierte Technologien ermöglichen. Das Zeichnen von runden Formen ist zwar prinzipiell möglich, setzt aber die Steuerung der Beschleunigung der Motoren voraus. Dies ist nicht in der Firmware für den EV3 vorgesehen, könnte sich aber möglicherweise durch schrittweises Ändern der Geschwindigkeit simulieren lassen. Zusätzlich könnte neben einer fortgeschritteneren Motorsteuerung auch noch eine fortgeschrittenere Bildverarbeitung implementiert werden. Diese wäre dann in der Lage, Rastergrafiken runter zu skalieren und Verarbeitung von Pixelwerten zu ermöglichen, die irgendwo zwischen den Farben der vorhandenen Stiften liegen.

VI. Zusammenfassung und Fazit

Trotz all der offenen Verbesserungsmöglichkeiten wurde die vorhandene Zeit sehr effektiv genutzt und es konnte ein brauchbares Ergebnis erzielt werden. Bei der Konstruktion und Programmierung einer Drucker-ähnlichen Maschine müssen viele Faktoren berücksichtigt und ein nicht unerheblicher Anteil der Zeit für das Testen eingeplant werden. Durch leichte Hardware-Anpassungen und einige Nachbesserungen am Code könnte aus der finalen Konstruktion noch viel mehr herausgeholt werden, als in der zur Verfügung stehenden Zeit möglich war. Es zeigt sich, dass es nicht unbedingt schwer ist, einfache Inhalte vom Computer auf ein Blatt zu übertragen. Der Teufel liegt jedoch im Detail und je brauchbarer das Ergebnis sein soll, desto mehr Aufwand fließt in die Optimierung von Mechanismen und des Codes.

Literaturverzeichnis

[1] David Lee: Drucker: Ist der Ruf erst ruiniert, wirbt es sich ganz ungeniert. https://www.galaxus.de/de/page/drucker-ist-der-ruf-erst-ruiniert-wirbt-es-sich-ganz-ungeniert-30891. – Abgerufen: 26.02.2024

[2] Spillerrec: *Lego RCX Plotter*. YouTube. https://www.youtube.com/watch?v=IHMdCttb1Qo. – Abgerufen: 26.02.2024

[3] JK Brickworks: *LEGO Printer*. YouTube. https://www.youtube.com/watch?v=dHmgaLgFRGM. – Abgerufen: 26.02.2024

[4] Laurens Valk: *LEGO MINDSTORMS NXT 2.0 Plotter / Printer*. YouTube. https://www.youtube.com/watch?v=tNp62EfaJ60. – Abgerufen: 26.02.2024

[5] Unbekannt: *Tolerances, accuracies and their sensible limits*. LEGOISM.INFO. https://legoism.blogspot.com/2017/01/tolerances-accuracies-and-their.html. – Abgerufen: 26.02.2024

[6] Wikipedia: *PostScript*. https://de.wikipedia.org/wiki/PostScript. – Abgerufen: 26.02.2024

[7] mindboards: *LEGO MINDSTORMS EV3 source code*. GitHub. https://github.com/mindboards/ev3sources/blob/78ebaf5b6f8fe31cc17aa5dce0f8e4916a4fc072/lms2012/c_output/source/c_output.c#L71. – Abgerufen: 26.02.2024

[8] Unbekannt: *LEGO Gear Ratio Calculator*. https://gears.sariel.pl/. – Abgerufen: 26.02.2024

[9] Unbekannt: *LEGO Wheels Chart*. http://www.wheels.sariel.pl/. – Abgerufen: 26.02.2024

Roboter zum Eierfärben

Kein Ersatz für den Osterhasen, aber für präzises und kreatives Eierfärben!

Korchunova Oleksandra, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Rahmen eines spannenden LEGO-Mindstorms-Projekts im Fachbereich Elektrotechnik und Informationstechnik im Jahr 2025 wurde ein Roboter entwickelt, der Ostereier automatisiert bemalen kann. Neben der konzeptionellen Planung erfolgte auch die Umsetzung eines präzisen Programmcodes, der eine flexible und effiziente Steuerung ermöglicht.**

***Schlagwörter*—LEGO-Mindstorms, MATLAB, Präsentation, Präzision, Roboter.**

I. Einleitung

Beim herkömmlichen Färben von Ostereiern bedarf es großer Geduld und einer ruhigen Hand, um gleichmäßige und ansprechende Muster zu erzielen. Häufig führt die manuelle Technik zu ungleichmäßigen Farbaufträgen, insbesondere bei der Verwendung mehrerer Farben.

Der vorgestellte Eierbemalungsroboter automatisiert diesen Prozess und ermöglicht so eine gleichmäßige sowie kreative Gestaltung. Durch die Minimierung von Farbfehlern und die Vereinfachung von Mustervariationen eröffnet das System neue Möglichkeiten für künstlerische Designs.

Die robuste Konstruktion des Roboters garantiert, dass das Ei sicher fixiert und gleichmäßig rotiert wird, während der bewegliche Pinselhalter auf einer vorgegebenen Bahn präzise Linien und Muster aufträgt. Optional können integrierte Sensoren den Farbauftrag überwachen und optimieren, was zu konsistenten und ästhetisch ansprechenden Ergebnissen führt.

II. Vorbetrachtungen

Im Vorfeld wurden grundlegende Aspekte für die Umsetzung des Projekts analysiert:

- Einsatz von LEGO Mindstorms als Plattform für die Roboterentwicklung.
- Verwendung von Motoren zur präzisen Steuerung der Farbapplikation.
- Integration unterschiedlicher Muster durch gezielte Bewegungsabläufe.
- Herausforderungen bei der fixen Positionierung der Eier.

A. Aufgaben des Roboters

Eine präzise Definition der vom Roboter zu erfüllenden Aufgaben ist essenziell, um zu verhindern, dass vielversprechende Konzepte in endlosen Optimierungsroutinen stecken bleiben. Deshalb wurde ein solides konzeptionelles Fundament geschaffen, auf dem später die Programmierung und der Aufbau mit LEGO-Komponenten basierten.

DOI: 10.24352/UB.OVGU-2025-018

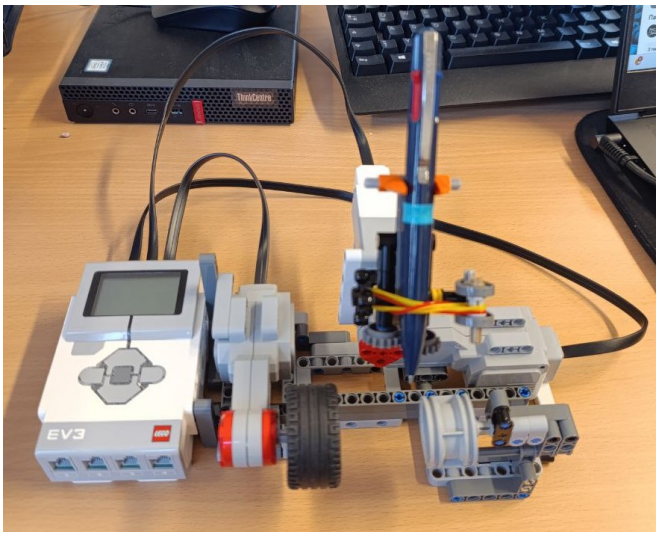


Abbildung 1. Roboter zum Eierfärben

B. Funktionsprinzip der LEGO-Komponenten

Vor dem Start der Konstruktion lag der Schwerpunkt auf der Untersuchung der Arbeitsweise der EV3-Motoren (siehe Abbildung 1). Besonderes Augenmerk wurde dabei auf den Umgang mit dem Pinsel und die Möglichkeit der Integration von Farbsensoren gelegt, da diese maßgeblich zur Qualität der Bemalung beitragen.

Diese Voruntersuchungen ermöglichten es, die Potenziale des LEGO-Systems realistisch einzuschätzen und lieferten gleichzeitig wichtige Erkenntnisse für die spätere Softwaresteuerung. So konnte ein stabiler und gut strukturierter Quellcode entwickelt werden.

C. Technische Umsetzung und Softwarekomponenten

Obwohl zahlreiche Designkonzepte entwickelt wurden, stießen erste Ansätze aufgrund der Einschränkungen in MATLAB und der mechanischen Herausforderungen beim Zusammenbau der LEGO-Elemente auf Schwierigkeiten. Diese Faktoren reduzierten zunächst die Funktionalität des Roboters. Zudem führte die Implementierung in MATLAB zu unerwarteten Problemen, die wiederholte Anpassungen des Codes erforderten.

III. Konstruktion

A. Design

Der Eierbemalungsroboter basiert auf dem EV3-Controller (siehe Abbildung 2), der als zentrale Steuereinheit fungiert und

Abbildung 2. EV3-Controller

Abbildung 3. Großer Motor

die unterschiedlichen Bewegungsabläufe koordiniert. Zur präzisen und gleichmäßigen Farbapplikation werden drei separate Motoren eingesetzt:

- Motor A: Zuständig für die Rotation des Eis [2]. Eine gleichmäßige Drehung sorgt für einen durchgängigen Farbauftrag ohne unregelmäßige Verläufe. Geschwindigkeit und Drehrichtung können an das gewünschte Muster angepasst werden (siehe Abbildung 3).
- Motor C: Steuert die horizontale Bewegung des Pinsels [2], wodurch unterschiedliche Muster erzeugt werden können (siehe Abbildung 3).
- Motor B: Regelt die vertikale Bewegung des Pinsels, was eine differenzierte Farbapplikation in variablen Höhen erlaubt. Je nach Design kann der Pinsel sanft oder mit mehr Druck aufgetragen werden, um unterschiedliche Intensitäten zu erzielen (siehe Abbildung 4).

B. Herausforderungen und Lösungsansätze

Zu Beginn des Projekts wurde der Roboter mit einer drehbaren Halterung für das Ei entworfen, um eine gleichmäßige Rotation und eine exakte Farbapplikation zu gewährleisten (siehe Abbildung 1). Allerdings traten in den ersten Tests diverse Probleme auf:

- Stabilität des Eis: Das Ei verrutschte während der Drehbewegung, was zu ungleichmäßigen Mustern führte. Dieses Problem konnte durch eine verbesserte Fixierung gelöst werden.
- Präzision der Motoren: Erste Tests zeigten ungenaue Bewegungen, wodurch die Muster unsauber wurden. Eine Feinjustierung der Steuerparameter in MATLAB erhöhte die Genauigkeit und ermöglichte eine gleichmäßige Farbverteilung.

Abbildung 4. Kleiner Motor

- Synchronisation der Motoren: Für die Optimierung der Muster war eine exakte Abstimmung der Motoren notwendig. Durch mehrere Testphasen und Anpassungen der Bewegungsabläufe konnte eine präzise Synchronisation erreicht werden (siehe Abbildung 5).

Diese Optimierungen führten zu einer verbesserten Stabilität der Konstruktion, reduzierten Vibrationen und einer insgesamt gesteigerten Leistung des Roboters.

C. Programmierung in MATLAB [1]

Für die Steuerung des Roboters wurden mehrere Malmodi implementiert, um eine vielseitige und präzise Gestaltung der Eier zu ermöglichen. Die Befehle werden über ein Kabel an den EV3-Controller übertragen, da Versuche mit Bluetooth aufgrund instabiler Verbindungen nicht erfolgreich waren.

Der Code ist modular aufgebaut und umfasst einzelne Blöcke, die die Bewegungen der drei Motoren steuern. Dabei werden Parameter wie Geschwindigkeit, Drehrichtung und Bewegungsdauer festgelegt. Zwei Hauptmodi wurden programmiert:

1) Gleichmäßige Farbapplikation: Das Ei rotiert kontinuierlich, während der Pinsel konstant Farbe aufträgt.
2) Gestreifte Farbgebung: Der Pinsel wird in definierten Intervallen abgesenkt und angehoben, um ein gestreiftes Muster zu erzeugen.

Die Steuerung erfolgt mithilfe einer `for`-Schleife, die die fortlaufende Bewegung der Motoren sicherstellt. Zudem wird eine `if`-Bedingung verwendet, um den Farbauftrag zu kontrollieren und im Fehlerfall den Betrieb zu unterbrechen, um eine mangelhafte Bemalung zu vermeiden.

Der modulare Aufbau des Quellcodes ermöglicht es, Muster flexibel anzupassen und zukünftige Funktionen problemlos zu integrieren.

```
motorEgg = motorB;
motorBrush = motorA;
motorMove = motorC;
```

```
disp('Wählen Sie einen Malmodus:');
disp('1 - Einheitliche Farbgebung');
disp('2 - Gestreifte Färbung');
mode = input('Geben Sie 1, 2 ein: ');
if mode == 1
disp('Gleichmäßiges
Färben beginnt...');
motorBrush.setProperties
('power',20,'limitValue', 145);
start(motorBrush);
pause(1);
for i = 1:50
powerValue = 57;
motorEgg.setProperties
('power',-25);
start(motorEgg);
motorMove.brakeMode='brake';
motorMove.setProperties
('power',30,'limitValue',
powerValue);
start(motorMove);
motorMove.waitFor;
motorMove.setProperties
('power',-30,'limitValue', powerValue);
start(motorMove);
motorMove.waitFor;
stop(motorEgg);
end
motorBrush.setProperties
('power',-20);
start(motorBrush);
stop(motorBrush);
stop(motorMove);
elseif mode == 2
for i = 1:13
motorEgg.setProperties
('power',-30);
start(motorEgg);
motorBrush.setProperties
('power',10,'limitValue', 140);
start(motorBrush);
pause(2.5);
stop(motorBrush);
motorBrush.setProperties
('power',-10);
start(motorBrush);
pause(0.5);
motorMove.setProperties
('power',10,'limitValue', 4);
start(motorMove);
pause(0.5);
stop(motorEgg);
end
stop(motorBrush);
stop(motorMove);
stop(motorEgg);
disp('Falsche Wahl!');
end
disp('Programm ist abgeschlossen.');
```

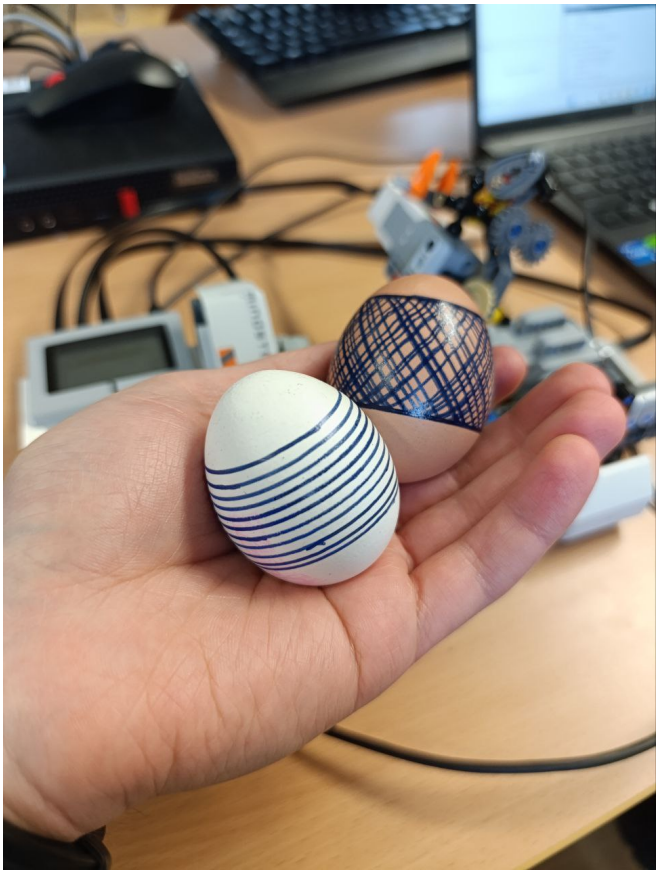


Abbildung 5. Bemalte Eier

IV. Diskussion der Ergebnisse

Nach zahlreichen Testläufen mit verschiedenen Mustern und Farbvarianten konnte bestätigt werden, dass der Roboter die vorgegebenen Anforderungen erfüllt. Die einzige Einschränkung lag in der kabelgebundenen Verbindung zum Laptop, die für die MATLAB-Steuerung notwendig ist. Dennoch arbeitete die Motorsteuerung zuverlässig und ermöglichte eine präzise Ausführung unterschiedlicher Maltechniken.

Insgesamt zeigte sich, dass der Roboter stabil und effizient arbeitet. Kleinere Software- und Mechanikanpassungen könnten zukünftig die Präzision der Farbapplikation weiter optimieren (siehe Abbildung 5).

V. Zusammenfassung und Fazit

Das Projekt erreichte nicht nur das Ziel, einen funktionierenden Roboter zu entwickeln, sondern lieferte auch wertvolle Einblicke in die Verbindung von Mechanik, Programmierung und Design. Die präzise Steuerung ermöglichte einen gleichmäßigen Farbauftrag, und künftige Erweiterungen könnten die Effizienz und Funktionalität weiter steigern.

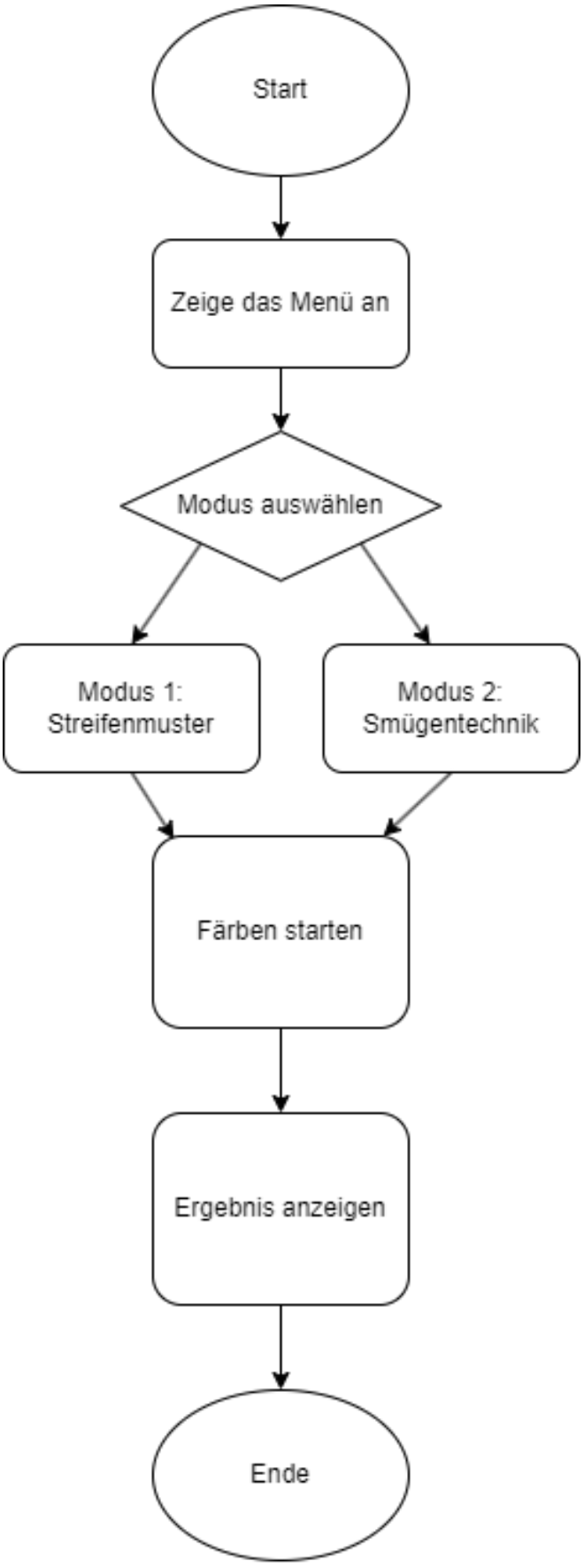


Abbildung 6. Programmablaufplan

LITERATURVERZEICHNIS

[1] MATHWORKS: *MATLAB*. Verfügbar unter: https://de.mathworks.com/products/matlab.html Version: 2024
[2] MINDSTORMS NXT: *NXT Motor*. Verfügbar unter: https://mindstormsnxt.blogspot.com/2006/08/closer-look-at-nxt-motors.html Version: 2006
[3] MATHWORKS: *MATLAB Syntax*. Verfügbar unter: https://de.mathworks.com/help/matlab/ref/function.html Version: 2024
[4] LEGO® MINDSTORMS® EV3: . Verfügbar unter: https://education.lego.com/en-us/product-resources/mindstorms-ev3/downloads/building-instructions/ Version: 2025
[5] *Lego Mindstorms EV3*. Verfügbar unter: https://www.academia.edu/42810872/ Jahr: 2020

Roboter zum Eierfärben

Wird keinen Osterhase ersetzen, aber das Eierfärben präziser und kreativer machen!

Nebrytov Maksym, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Rahmen des innovativen LEGO-Mindstorms-Projekts im Studiengang Elektrotechnik und Informationstechnik im Jahr 2025 wurde ein autonomer Roboter entwickelt, der in der Lage ist, Ostereier automatisch und individuell zu bemalen. Die Idee zu diesem Projekt entstand aus der Überlegung, ein kreatives, gleichzeitig technisch herausforderndes und praxisnahes Vorhaben umzusetzen, das sowohl mechanische Präzision als auch intelligente Steuerung vereint. Der Gedanke, traditionelle Handarbeit durch Automatisierung kreativ zu interpretieren, führte schließlich zur Entwicklung eines Ostereier-Bemalroboters.**

Neben der Konzeption des Roboters wurde ein präziser Programmcode implementiert, um eine effiziente und flexible Steuerung zu gewährleisten.

Die Einführung in das Seminar begann mit der Erläuterung dieses Vorhabens, gefolgt von einer fundierten Einführung in die Grundlagen der Programmierung mit MATLAB. Dies bildete das unerlässliche Rüstzeug, um die Studierenden mit den essenziellen Kenntnissen und Fähigkeiten auszustatten, die für die Umsetzung ihres Roboterprojekts unabdingbar waren. Der Lernprozesser streckte sich über verschiedene Phasen, die von der Theorie bis zur praktischen Umsetzung reichten.

***Schlagwörter*—LEGO-Mindstorms, MATLAB, Präsentation, Präzision, Roboter.**

I. Einleitung

Beim traditionellen Ostereierfärben sind Geduld und Präzision benötigt, um schöne Muster zu erzeugen. Die Qualität des Ergebnisses hängt von der ruhigen Hand und der gleichmäßigen Farbverteilung ab. Gleichzeitig kann es schwierig sein, gleichmäßige Muster zu erzeugen, insbesondere wenn mehrere Farben verwendet werden.

Mit dem Eierfärbe-Roboter lässt sich dieser Prozess automatisieren, wodurch eine gleichmäßige und kreative Gestaltung der Eier ermöglicht wird. Das System reduziert Farbfehler und erleichtert das Experimentieren mit verschiedenen Mustern.

Der Roboter ist mit einer stabilen Basis ausgestattet, die das Ei sicher hält und rotiert. Gleichzeitig bewegt sich der Pinselhalter entlang einer programmierten Bahn, um präzise Linien oder Muster aufzutragen. Ein integrierter Sensor kann dabei helfen, Farbaufträge zu kontrollieren und gegebenenfalls anzupassen. Dadurch wird ein gleichmäßiges und ästhetisches Ergebnis erzielt.

II. Vorbetrachtungen

Einsatz von LEGO Mindstorms für die Robotik-Umsetzung. Nutzung von Motoren zur präzisen Steuerung der Farbauftragung. Integration verschiedener Muster durch gezielte Bewegungsabläufe. Herausforderungen in der exakten Fixierung der Eier.

DOI: 10.24352/UB.OVGU-2025-019

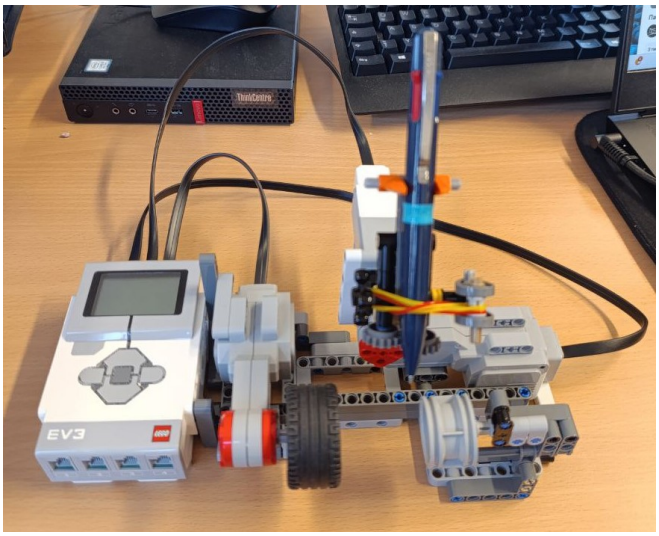


Abbildung 1. Roboter zum Eierfärben

A. Roboteraufgaben

Eine klare Definition der Aufgaben, die der Roboter ausführen soll, ist entscheidend, da Unklarheiten oder Unsicherheiten dazu führen können, dass eine vielversprechende Idee in endlosen Optimierungsversuchen stecken bleibt. Deshalb wurde ein solides konzeptionelles Fundament entwickelt, auf dem anschließend die Programmierung und der Aufbau mit LEGO-Elementen basieren.

B. Funktionsweise der Lego Komponentn

Bevor mit der Konstruktion des Eierfärbe-Roboters begonnen wurde, stand zunächst die Untersuchung der Funktionsweise der EV3-Motoren (siehe Abbildung 1) im Fokus.

Besonderes Augenmerk galt zudem der Handhabung des Pinsels und der möglichen Integration von Sensoren zur Farbkontrolle, da diese eine entscheidende Rolle für die Qualität der Bemalung spielen.

Durch diese Voruntersuchungen konnten die Möglichkeiten des LEGO-Systems für die weitere Umsetzung besser eingeschätzt werden. Gleichzeitig ermöglichte dieses Wissen ein tieferes Verständnis der Softwaresteuerung, was sich später als hilfreich erwies, um einen stabilen und gut strukturierten Quellcode zu entwickeln.

C. Technische Umsetzung und Softwarekomponenten

Trotz zahlreicher Designkonzepte wurde das Projekt anfangs durch die Grenzen von MATLAB und die strukturellen

Abbildung 2. EV3-Controller

Abbildung 3. Großer Motor

Herausforderungen beim Zusammenbau der LEGO-Elemente eingeschränkt. Diese Faktoren begrenzten auch die Funktionalität des Roboters. Zusätzlich stellte die Implementierung in MATLAB das Team vor unerwartete Schwierigkeiten, da bestimmte Funktionen Anpassungen im Quellcode erforderlich machten, was zu mehreren Überarbeitungen führte.

III. Konstruktion

A. Design

Der Eierfärbe-Roboter basiert auf dem EV3-Controller (siehe Abbildung 2) , der als zentrale Steuereinheit fungiert und die verschiedenen Bewegungsabläufe koordiniert. Damit die Farbmuster präzise und gleichmäßig auf das Ei aufgetragen werden, kommen drei Motoren zum Einsatz, die spezifische Aufgaben übernehmen:

Motor A: Verantwortlich für die Drehung des Eis [2] . Durch eine gleichmäßige Rotation wird sichergestellt, dass die Farbe lückenlos aufgetragen wird und keine ungleichmäßigen Farbverläufe entstehen. Die Geschwindigkeit und Richtung der Drehung können je nach gewünschtem Muster variieren (siehe Abbildung 3).

Motor C: Regelt die horizontale Bewegung des Pinsels [2] . Durch diese kontrollierte Seitwärtsbewegung können verschiedene Muster auf das Ei Oberfläche aufgetragen werden. (siehe Abbildung 3).

Motor B: Steuert die vertikale Bewegung des Pinsels. Dies ermöglicht eine kontrollierte Farbauftragung in unterschiedlichen Höhen des Eis. Abhängig vom gewählten Design kann der Pinsel sanft oder mit stärkerem Druck aufgesetzt werden, um verschiedene Intensitäten der Farbauftragung zu erzielen (siehe Abbildung 4).

Abbildung 4. Kleiner Motor

B. Herausforderungen und Lösungen

Zu Beginn wurde der Roboter mit einer rotierenden Halterung für das Ei entworfen, die eine gleichmäßige Rotation ermöglichen und eine präzise Farbauftragung gewährleisten sollte (siehe Abbildung 1). Allerdings traten während der ersten Tests mehrere Herausforderungen auf.

Ein zentrales Problem war die Stabilität des Eis. In der ursprünglichen Konstruktion verrutschte das Ei während der Drehbewegung, was zu ungleichmäßigen Farbmustern führte. Durch eine verbesserte Fixierung konnte dieses Problem behoben werden.

Ein weiteres Hindernis war die Präzision der Motoren. Erste Tests zeigten, dass die Bewegungen nicht genau genug waren, wodurch die Muster unsauber wurden. Durch Feinjustierungen der Steuerparameter in MATLAB konnte die Genauigkeit optimiert und eine gleichmäßige Farbauftragung erreicht werden.

Zudem stellte sich heraus, dass die Optimierung der Muster eine genauere Synchronisation der Motoren erforderte. Manche Designs konnten anfangs nicht präzise umgesetzt werden. Durch eine Reihe von Testphasen und Anpassungen der Bewegungsabläufe gelang es, eine verbesserte Abstimmung der Motoren zu erreichen und die gewünschten Muster exakt zu reproduzieren (siehe Abbildung 5).

Durch diese Anpassungen wurde die Stabilität der Konstruktion verbessert, unerwünschte Vibrationen reduziert und die Gesamtleistung des Roboters erheblich optimiert.

C. Coding in MATLAB

Bei der Programmierung des Roboters wurde beschlossen, verschiedene Färbemodi zu implementieren, um eine präzise und vielseitige Gestaltung der Eier zu ermöglichen. Die Steuerung erfolgt über MATLAB [1] , wobei die Befehle über ein Kabel an den EV3-Controller gesendet werden. Ein Versuch, Bluetooth zu verwenden, scheiterte, da keine stabile Verbindung hergestellt werden konnte. Daher blieb die kabelgebundene Lösung die einzige praktikable Option.

Der Quelltext ist modular aufgebaut und besteht aus einzelnen Befehlsblöcken, die die Bewegungen der drei Motoren steuern. Jeder Abschnitt definiert die Geschwindigkeit, Drehrichtung und Dauer der Motorbewegungen. Dabei wurden verschiedene Färbemodi programmiert:

Gleichmäßiges Färben: Das Ei dreht sich kontinuierlich, während der Pinsel gleichmäßig Farbe aufträgt. Streifenmuster: Der Pinsel wird in regelmäßigen Abständen gesenkt und angehoben, um ein gestreiftes Muster zu erzeugen.

Die Steuerung erfolgt durch eine for-Schleife, die die kontinuierliche Bewegung der Motoren sicherstellt. Zusätzlich wurde eine if-Bedingung implementiert, um den Farbauftrag zu kontrollieren. Falls das Ei nicht korrekt rotiert oder der Pinsel eine unerwartete Position erreicht, stoppt der Roboter automatisch, um eine fehlerhafte Bemalung zu vermeiden.Die Funktionsweise des Programmes (siehe Abbildung 6) .

Der gesamte Quelltext ist darauf ausgelegt, eine präzise Steuerung der Bewegungen zu gewährleisten und gleichzeitig genügend Flexibilität für zukünftige Erweiterungen zu bieten. Die modulare Struktur ermöglicht eine einfache Anpassung der Muster und die Integration neuer Funktionen.

```
motorEgg = motorB;
motorBrush = motorA;
motorMove = motorC;


disp('Wählen Sie einen Malmodus:');
disp('1 - Einheitliche Farbgebung');
disp('2 - Gestreifte Färbung');

mode = input('Geben Sie 1, 2 ein: ');

if mode == 1
disp('Gleichmäßiges
Färben beginnt...');
motorBrush.setProperties
('power',20,'limitValue', 145);
start(motorBrush);
pause(1);

for i = 1:50
powerValue = 57;
motorEgg.setProperties
('power',-25);
start(motorEgg);
motorMove.brakeMode='brake';
motorMove.setProperties
('power',30,'limitValue',
powerValue);
start(motorMove);
motorMove.waitFor;
motorMove.setProperties
('power',-30,'limitValue', powerValue);
start(motorMove);
motorMove.waitFor;
stop(motorEgg);
end

motorBrush.setProperties
('power',-20);
start(motorBrush);
stop(motorBrush);
stop(motorMove);

elseif mode == 2

for i = 1:13
motorEgg.setProperties
('power',-30);
start(motorEgg);
motorBrush.setProperties
('power',10,'limitValue', 140);
start(motorBrush);
pause(2.5);
stop(motorBrush);
 motorBrush.setProperties
  ('power',-10);
 start(motorBrush);
pause(0.5);
 motorMove.setProperties
('power',10,'limitValue', 4);
 start(motorMove);
pause(0.5);
stop(motorEgg);
end

 stop(motorBrush);
stop(motorMove);
stop(motorEgg);
disp('Falsche Wahl!');
end

disp('Programm ist abgeschlossen.');
```

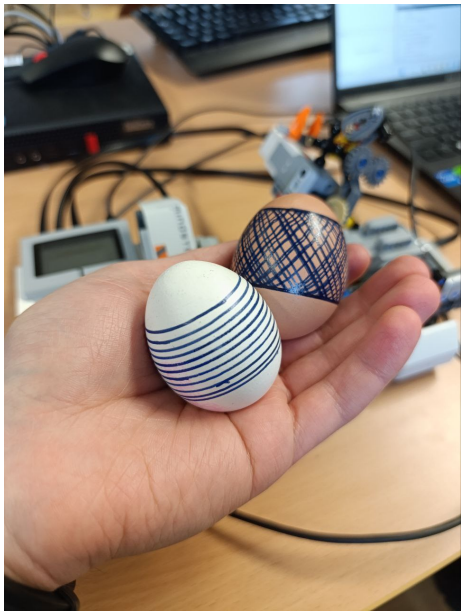


Abbildung 5. Bemalte Eier

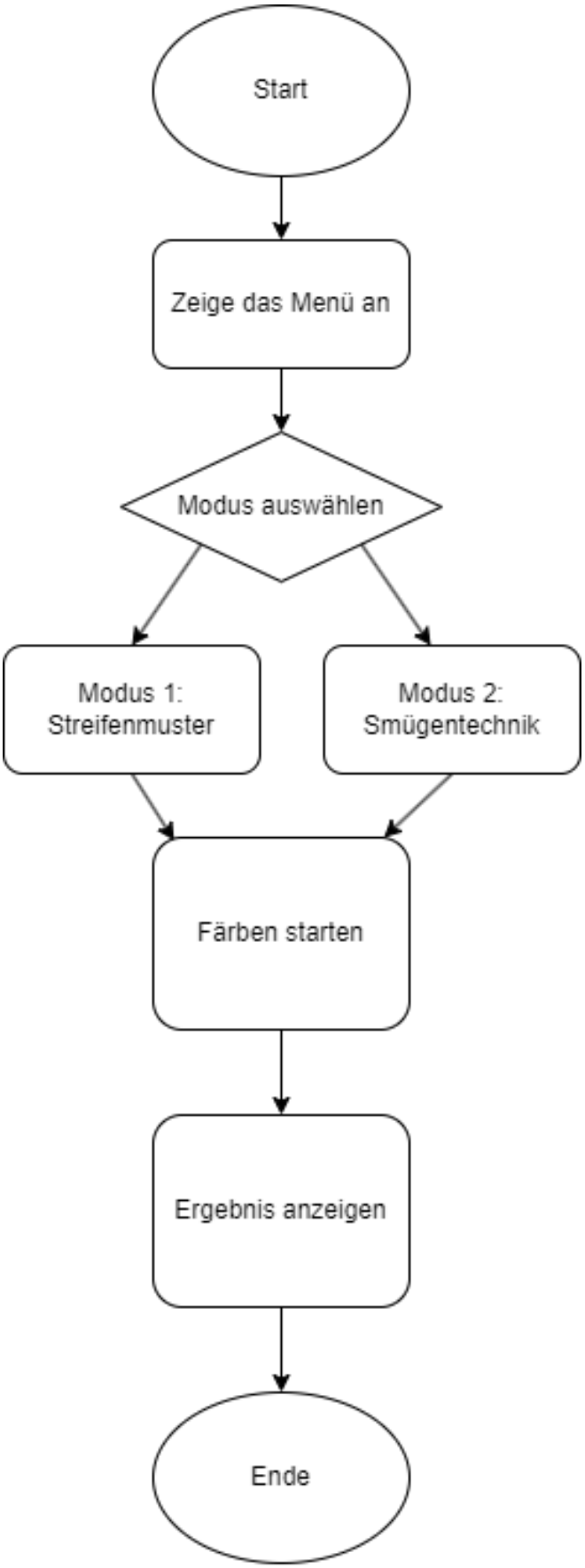


Abbildung 6. Programmablaufplan

IV. Ergebnisdiskussion

Nach mehreren Tests mit unterschiedlichen Mustern und Farben wurde bestätigt, dass der Roboter die gestellten Anforderungen erfüllte. Die einzige Einschränkung bestand in der Verbindung zum Laptop über ein Kabel, das für die Steuerung über MATLAB notwendig war. Trotzdem funktionierte die Motorsteuerung zuverlässig, und der Roboter konnte verschiedene Maltechniken präzise ausführen.

Insgesamt wurde der Roboter als stabil und effizient bewertet, wobei kleinere Verbesserungen in der Software und Mechanik das Potenzial für eine noch präzisere Farbgestaltung bieten (siehe Abbildung 5) .

V. Zusammenfassung und Fazit

Das Projekt erreichte nicht nur das Ziel, einen funktionierenden Roboter zu konstruieren, sondern ermöglichte auch wertvolle Einblicke in die Kombination von Mechanik, Programmierung und Design. Die präzise Steuerung ermöglicht eine gleichmäßige Farbauftragung, und zukünftige Erweiterungen könnten die Effizienz und Funktionalität weiter verbessern.

Literaturverzeichnis

[1] MATHWORKS: *MATLAB*. Verfügbar unter: https://de.mathworks.com/products/matlab.html Version: 2024

[2] MINDSTORMS NXT: *NXT Motor*. Verfügbar unter: https://mindstormsnxt.blogspot.com/2006/08/closer-look-at-nxt-motors.html Version: 2006

Die Maus

Daniel Anders, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Der in diesem Seminar entwickelte Projektroboter „Die Maus“ befasst sich mit der Entwicklung eines autonomen Fahrzeugs, das mithilfe klassischer Bildverarbeitung einen Ausgang in einem Raum erkennt und diesen verlässt, ähnlicher einer Maus, die auf der Suche nach einem Mauseloch ist. Die Konstruktion basiert auf LEGO Mindstorms, einem EV3-Baustein als Steuereinheit und einer Webcam zur Bildaufnahme. Die Bildverarbeitung erfolgt in MATLAB unter Verwendung der ComputerVisionToolbox, wobei Kantenerkennung und Polygonerkennung zur Detektion des Ausgangs eingesetzt werden. Erste Tests mit einer Konstruktion auf vier Rädern zeigten Schwierigkeiten bei Drehbewegungen, weshalb eine optimierte Version mit drei Omniwheels entwickelt wurde. Die Maus bewegt sich gezielt auf den Ausgang zu und korrigiert ihre Position anhand einer kontinuierlichen Bildauswertung. Ein Ultraschallsensor unterstützt das System, indem er erkennt, ob die Maus den Raum verlassen hat. Erste Tests bestätigen die Funktionsfähigkeit des Systems, obwohl Herausforderungen wie falsch-positive Erkennungen Optimierungspotenzial aufzeigen.**

***Schlagwörter*—Bilderkennung, ComputerVisionToolbox, EV3, MATLAB, Omniwheels, Polygonerkennung**

I. Einleitung

DEN Autofahrern der heutigen Welt werden durch digitale Computersysteme zunehmend durch Brems-, Spurhalte- und Parkassistenten unterstützt, bis hin zum autonomen Fahren. Hierfür werden Sensoren und Kameras in Echtzeitdatenverarbeitungssysteme integriert, die nicht nur strenge Echtzeitkriterien einhalten müssen, sondern auch mit hoher Zuverlässigkeit und geringer Fehlerrate funktionieren müssen. Fortschritte im Bereich der künstlichen Intelligenz verbesserten Erkennungsrate von komplexen Objekten in Kamerasystemen, lösten allerdings traditionelle Bildverarbeitung wie Kantenerkennungen aufgrund ihrer Schnelligkeit und Einfachheit für klar definierte Probleme nicht vollständig ab [1]. Klassische Einparkassistenten setzen auf Ultraschallsensoren und Rückfahrkameras, die die Aufgabe des Einparkens zwar unterstützen, allerdings diese letztendlich dem Menschen überlassen. Aus der Idee ein Fahrzeug zu entwickeln, welches selbstständig in eine Garage einparken kann entwickelte sich eine damit verwandte Idee der "Maus", die eigenständig mittels klassischer Bildverarbeitung einen rechteckigen Ausgang in einer Wand findet und so einen Raum verlassen kann.

II. Vorbetrachtungen

A. Welche Aufgaben soll die Maus erfüllen können?

Die Maus soll als Fahrzeug sich frei in einem Raum bewegen können und eigenständig durch Rotation um die eigene Achse sich so lange drehen, bis ein Ausgang im Blickfeld der Kamera erkannt wurde. Hierfür muss es sich mit und gegen den Uhrzeigersinn drehen können, sowie in der Lage sein Vorwärtsbewegungen durchzuführen. Die Maus soll selbst auf den Mittelpunkt eines außen schwarz markierten Ausgangs in kleinen Schritten zufahren können und in der Lage sein, nach jedem Schritt nach Bedarf sich durch Rotation des Fahrzeugs sich eigenständig neu auszurichten. Die Maus soll zusätzlich in der Lage sein unabhängig von der Kamera zu erkennen, dass es den Raum verlässt, der die Markierungen des Ausgangs zu diesem Zeitpunkt sich nicht mehr im Blickfeld befinden.

B. Welche Komponenten werden benötigt?

Für die Konstruktion des Fahrzeugs wurde ein umfangreicher LEGO-Mindstorms-Baukasten sowie ein EV3-Baustein zu Verfügung gestellt. Der EV3 ist dabei der zentrale Steuercomputer des Fahrzeugs, der die drei Motoren und einen Ultraschallsensor ansteuert. Ein Smartphone stellt die Webcam des Systems dar und wird mit Gummibändern an der Vorderseite des Fahrzeugs befestigt.

C. Wie erkennt die Maus den Ausgang?

Die am Fahrzeug befestigte Webcam liefert Bilddaten, welche aufgrund des hohen Berechnungsaufwands nicht lokal auf dem Fahrzeug, sondern in einer MATLAB-Umgebung mittels der als Add-on installierten ComputerVisionToolbox verarbeitet werden. Der Ausgang befindet sich in einem rechtwinkligen Raum und muss einen starken Kontrast zu den Wänden aufweisen. In der selbstgebauten Testumgebung wurde daher weiße Wände für eine zuverlässigere Erkennung des Konturen des rechteckigen und mit schwarzem Klebeband umrandeten Ausgangs verwendet. Ein nach oben gerichteter Ultraschallsensor prüft, ob die sich die Decke in unmittelbarer Nähe befindet, um zu verhindern, dass die Maus beim Verlassen des Raums durch Drehungen erneut nach dem Ausgang sucht, sondern weiterhin vorwärts fährt um den Raum vollständig zu verlassen.

III. Fahrzeugkonstruktion und Softwareentwicklung

A. Konstruktion

Die erste Konstruktion des Fahrzeugs (siehe Abbildung 1) basierte auf insgesamt vier Rädern, wovon die vorderen zwei einfachen Räder durch jeweils einen großen LEGO-Mindstorms Motor angetrieben wurden und die hinteren Omniwheels dem Fahrtverlauf gefolgt sind. An dieser Konstruktion wurde bereits ein Ultraschallsensor angebracht und eine mit Gummibändern eingespannte USB-Webcam lieferte die Daten zur Bilderkennung. Das per USB ferngesteuerte Fahrzeug fuhr stabil und es konnten mit dieser ursprünglichen Konstruktion

DOI: 10.24352/UB.OVGU-2025-020

bereits erste Erfolge bei der Bilderkennung notiert werden. Mit zunehmendem Entwicklungsverlauf wurde jedoch klar, dass vor allem die Suche nach dem Ausgang durch Drehung um die eigene Achse sich als eine Herausforderung herausstellte. Abgesehen von dem großen Drehradius erhöhte das Mitziehen der Allseitenräder in einigen Fällen die Bewegungsungenauigkeit des Fahrzeugs erheblich, weshalb schlussendlich ein neuer Ansatz gewählt wurde. Für die zweite und finale Konstruktion

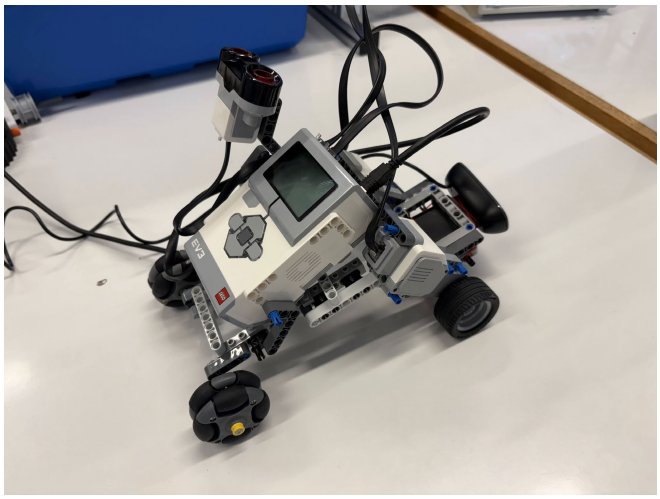


Abbildung 1. Erste Konstruktion auf vier Rädern mit zwei Motoren und USB Webcam

wurden drei in Stern angeordnete kleine LEGO-Mindstorm Motoren eingesetzt (siehe Abbildung 3). Die drei Omniwheels stehen soweit mit den LEGO Bauteilen umsetzbar nahezu 120^o zueinander versetzt und bieten weiterhin Fahrstabilität. Eine einfache Y-Anordnung der Motoren hat leider nicht gereicht, da den mittig sitzende EV3-Kern mit seiner Gewichtskraft auf das Fahrgestell einwirkt und diese leicht verbiegt, wodurch die Omniwheels nicht idealen Kontakt mit dem Untergrund haben. Um dies zu verhindern, wurde ein Dreieck zur Erhöhung der Stabilität hinzugefügt (siehe Abbildung2). Das neue Design hat den Vorteil, dass Drehung mit und gegen den Uhrzeigersinn durch Ansteuerung der Motoren in die selbe Drehrichtung bei gleicher Leistungszufuhr möglich ist und dabei keine Räder schleifen müssen.

Die Maus ist in der Lage sich vorwärts, aber auch in zwei weitere Richtungen seitwärts zu bewegen, in dem jeweils zwei Motoren mit der selben Leistung in entgegengesetzte Drehrichtungen betrieben werden, sodass zwei Omniwheels eine translatorische Bewegung hervorrufen und das dritte Omniwheel mitgezogen wird. Der Ultraschallsensor wurde seitlich an den EV3 angebracht und ist an die Decke gerichtet. An der Vorderseite wurde ein möglichst mittig positionierte Smartphone Kamera als kabellos verbundene Webcam eingesetzt, welche mit Gummibändern an der LEGO Konstruktion befestigt wurde. Somit ist der EV3 nun die einzige mit dem Hostcomputer via Kabel verbundene Komponente.

B. Funktionsweise und Software

Der Ablauf des in MATLAB entwickelten Programms der Maus ist in der Abbildung 7 vereinfacht dargestellt. Der EV3

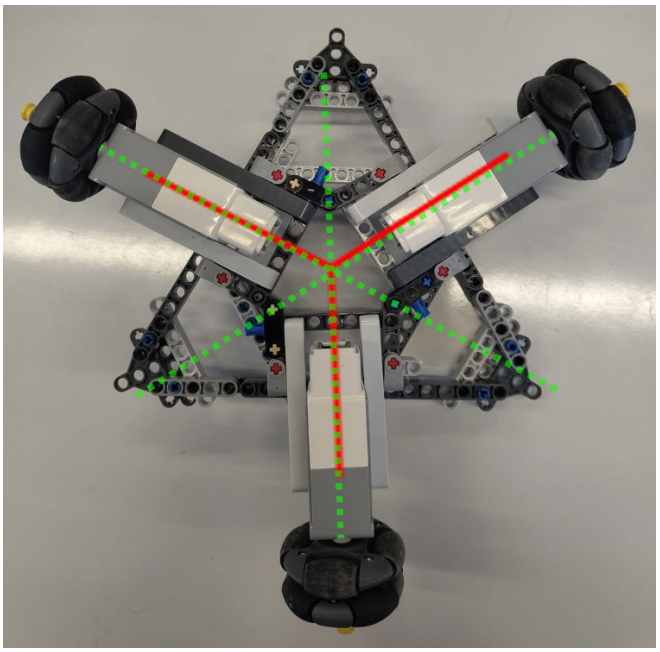


Abbildung 2. Unterseite der zweiten Konstruktion, Idealer 120^o Versatz (rot), Verlängerte Motor-Rad-Achsen (grün gepunktet)

wurde mittels der EV3-Toolbox [2] an MATLAB angebunden und ermöglichte eine Fernsteuerung des Fahrzeugs per USB. Eine auf dem Smartphone installierte App teilte die Kamera im lokalen WLAN mit dem Hostcomputer, auf dem der geteilte Kamerastream wie eine gewöhnliche Webcam abgegriffen werden kann. In MATLAB wurde die ComputerVisionToolbox als Add-On installiert und verwendet, um sowohl ein Webcam Objekt zu erstellen, als auch an den Snapshots dieses Objektes dann die Kanten und Polygonerkennung durchzuführen. Ein Snapshot steht dabei für den aktuell verfügbaren Frame, der in MATLAB als eine $3 \times 1280 \times 720$ große Matrix gespeichert wird.

Das RGB codierte Farbbild wird mit dem `rgb2gray()` Befehl zuerst in eine Graustufenmatrix , ähnlich der Darstellung in Abbildung 4 umgewandelt, damit anhand dieser dann die Kantenerkennung durchgeführt werden kann. Für die Kantenerkennung wurde der `edge()` Befehl mit einem Schwellwert von 0.7 sowie Canny als Erkennungsmethode verwendet [3], wobei die ermittelten Kanten des Ausgangs sich in Abbildung 5 dargestellt sind. Die tatsächliche Polygonerkennung übernimmt `regionprops()` mit der Konfiguration aus dem in dem Moment bereitgestellten schwarz-weiß Bild die Bounding Box [4] sowie Fläche der gefundenen Regionen zu ermitteln. Bereits hier können anhand dieser Daten ein Großteil der gefundenen Objekte gefiltert werde, da zu kleine oder große Rechtecke sowie Objekte mit unrealistischen Seitenverhältnis mit hoher Sicherheit verworfen werden können.

Als Ausgang interpretiert wird aus den verbleibenden Regionen das Rechteck, das nach einer Maximum-Suche den größten Flächeninhalt aufweist. Die Bounding Box dieser Fläche sowie dessen geometrischer Mittelpunkt sind in Abbildung 6 dargestellt. Sollte keine Region gefunden werden oder in Frage kommen, prüft der Ultraschallsensor, ob sich eine

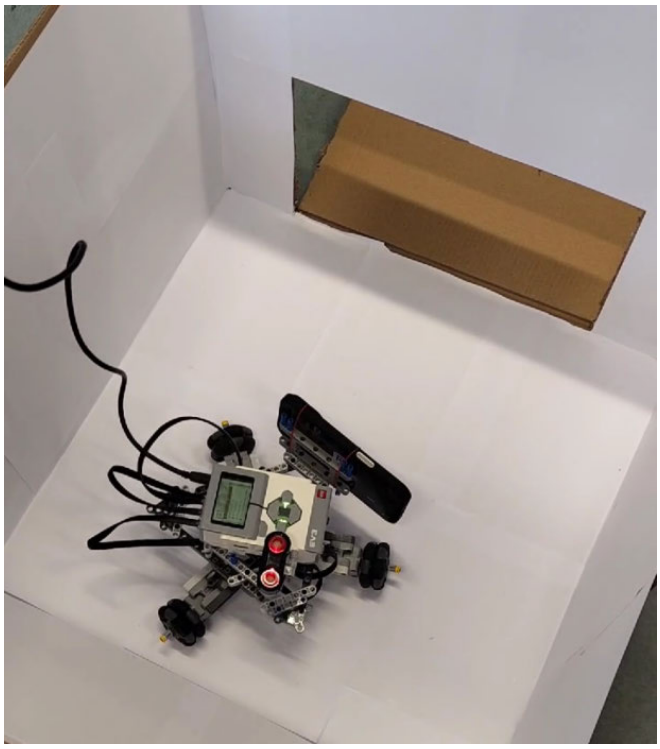


Abbildung 3. Zweite Konstruktion auf drei Rädern mit drei Motoren und Smartphone als Webcam, Fahrzeug in gebauter Testumgebung noch ohne schwarz umrandetem Ausgang

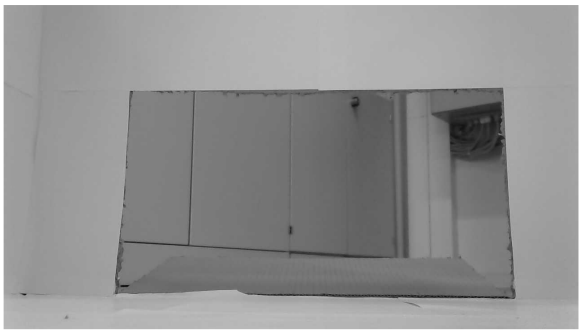


Abbildung 4. Graustufenbild des Ausgangs nach $rgb2gray()$ Befehl

Decke unmittelbar über dem Fahrzeug befindet, da dies darauf hindeutet, dass die Maus in dem Moment den Raum bereits verlässt und weiter gerade aus fahren muss. Sollte dies nicht der Fall sein, führt die Maus eine kleine Rotation gegen den Uhrzeigersinn um ihre eigene Achse durch, um weiterhin im Raum nach einem Ausgang zu suchen. Sobald ein gefundenes Rechteck als potentieller Ausgang erkannt wurde, wird aus der Bounding Box der geometrische Mittelpunkt ermittelt, anhand dessen Position drei Fälle unterschieden werden. Befindet sich der Mittelpunkt auch im Zentrum des Kamerablickfeldes, so wird davon ausgegangen, dass der Ausgang frontal vor der Maus liegt und es wird eine Vorwärtsbewegung durchgeführt. Als Zentrum gewertet wird ein Mittelpunkt, dessen X-Koordinate sich höchstens $20\,\%$ links oder rechts vom

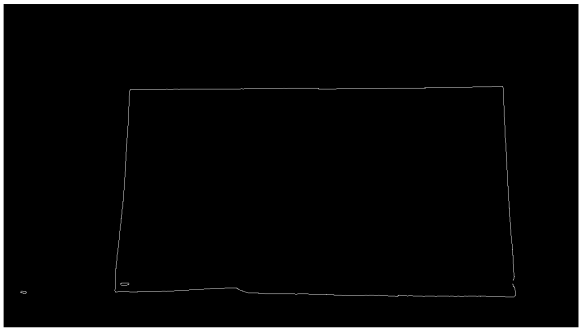


Abbildung 5. Ausgang nach Kantenerkennung mit Canny-Methode und Schwellwert von 0,7

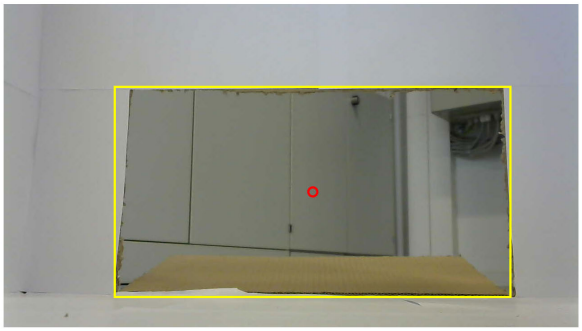


Abbildung 6. Ausgang mit Bounding Box (gelb) und geometrischer Mitte (rot)

Mittelpunkt des Frames befindet, wobei hier ein Kameraoffset an beide Bereichsgrenzen aufaddiert wird, um den negativen Einfluss der nicht vollständig mittig positionierten Kamera zu verringern. Befindet sich der Mittelpunkt des Ausgangs im rechten oder linken Bereich des Blickfelds, so werden durch kleine entgegengesetzte Drehbewegung die Maus auf diesen gefundenen Ausgang ausgerichtet. Da in jedem Durchlauf die nächste Bewegung neu berechnet wird, regelt die Maus bei Bedarf ihre Blickrichtung eigenständig auf den Ausgang, wodurch auch Ungenauigkeiten der Motoren weniger zu Gewicht fallen. Diese werden angesteuert, in dem zuerst bei allen Motoren der gewünschte Drehzahlwert (`TachoValue`) sowie deren Leistung konfiguriert, daraufhin alle Motoren gestartet werden und auf die Beendigung der Bewegung gewartet wird.

IV. Ergebnisdiskussion

Im Allgemeinen hat sich in Tests gezeigt, dass sowohl die vorgestellte Idee der Ausgangserkennung als auch das Verlassen des Raumes mit der finalen Konstruktion funktionieren. Die Maus findet den Ausgang oft innerhalb von Sekunden und bewegt sich entsprechend auf ihn zu. Auch die eigene Ausrichtung der Maus auf den Ausgang und dessen stetige Korrektur bei Annäherung lassen sich beobachten. Die Idee das Fahrzeug kabellos zu betreiben funktionierte zwar bei der Webcam, allerdings nicht für den EV3-Baustein, da dieser Bluetooth Verbindungsprobleme aufwies. Es wird vermutet, dass die

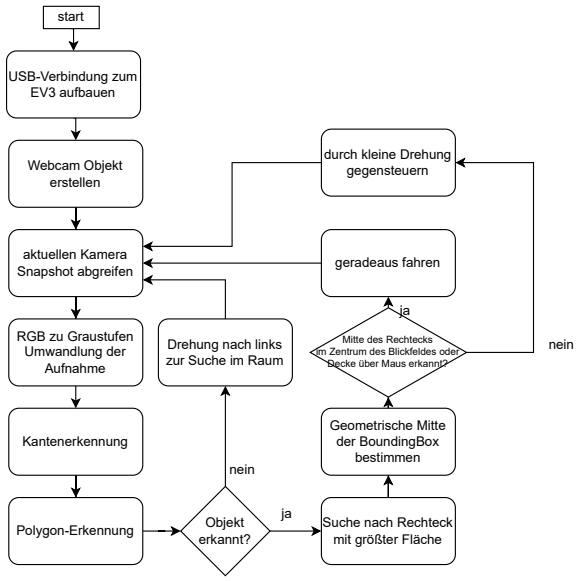


Abbildung 7. Vereinfachtes Ablaufdiagramm

höchste vom EV3 unterstützte Bluetooth-Version 2.1 [5] die Ursache für die Verbindungsprobleme mit heutiger moderner Hardware ist. Der Einsatz der Smartphone-Kamera als Webcam verbesserte die Wahrnehmen des Ausgangs signifikant und ermöglichte dies auch unter schwierigeren Lichtverhältnissen. Auch wenn der Ausgang mit hoher Zuverlässigkeit gefunden wird, sind falsch positive Erkennungen der Maus das größte Problem, da diese die Maus vom vorgesehen Weg abbringen und die Maus in diesen Fällen oft mit Seitenwänden kollidiert. Dieses Problem ist besonders präsent, wenn die Maus sich unmittelbar vor dem Ausgang befindet und das Kamerablickfeld überwiegend die Außenumgebung wahrnimmt.

Eine selbstgebaute Testumgebung mit weißen Wänden sowie einem schwarz markierten Ausgang sowie das feinjustieren der Kantenerkennungsparameter und Filterkriterien hält diese Fehlerkennungen in Grenzen, kann diese allerdings nicht vollständig verhindern. Hinzu kommt Bewegungsunschärfe der Kamera, dessen Einfluss jedoch minimiert wird, in dem das Fahrzeug nach jeder ausgeführten Bewegung für 300 Millisekunden ruht, um die Schwingungen an der Webcam ausklingen zu lassen. Es wurde beobachtet, dass die Maus häufiger beim Verlassen der Testumgebung mit den Seitenwänden kollidierte, was sich auf eine breite Fahrzeugkonstruktion der Maus, den physikalischen Kamera-Offset des Smartphones und zu groben Bewegungen erklären lässt. Korrekturen der Bereichsgrenzen des mittleren Bereichs im Blickfeld sowie eine Verringerung der zurückgelegten Strecke bei Vorwärtsbewegungen verringerte nicht nur die Anzahl der Kollisionen mit der Wand, sondern erhöhte zusätzlich die Erkennungsrate der Decke am Ausgang durch den Ultraschallsensor.

V. Zusammenfassung und Fazit

Das Projekt „Die Maus“ befasst sich mit der Entwicklung eines autonomen Fahrzeugs, welches mithilfe klassischer Bildverarbeitung eigenständig einen Ausgang in einem Raum erkennt und diesen verlässt. Das Fahrzeug basiert auf LEGO Mindstorms und wird von einem EV3-Baustein gesteuert, während eine Webcam Bilddaten liefert, die in MATLAB und der ComputerVisionToolbox verarbeitet werden. Eine Kombination aus Kantenerkennung und Polygonerkennung hilft dem System bei der Suche nach einem rechteckigen Ausgang und richtet sich auf diesen aus. Die erste Konstruktion mit vier Rädern erwies sich als nicht ideal, da Drehungen um die eigene Achse viel Raum benötigten und sich als zu ungenau herausstellten, weshalb eine zweite Version mit drei omnidirektionalen Rädern entwickelt wurde, die präzisere Drehbewegungen ermöglichte.

Während der Tests zeigte sich, dass das System den Ausgang zuverlässig findet, jedoch durch falsch-positive Erkennungen und Bewegungsunschärfen beeinträchtigt wurde, die durch Anpassungen minimiert werden konnten. Ein Ultraschallsensor unterstützt zudem das Verlassen des Raumes, indem erkannt wird, ob sich eine Decke über der Maus befindet. Das Projekt zeigt, dass autonome Navigation mittels klassischer Bildverarbeitung realisierbar ist, jedoch ein Bedarf nach weiteren Optimierungen in der Erkennungsgenauigkeit besteht. Als Fazit lässt sich festhalten, dass sich mit dem EV3 und einer MATLAB-Umgebung bereits mit wenigen Aktoren und Sensoren komplexere Systeme und Roboter realisieren lassen, die in der Lage sind reale Probleme zu lösen.

Literaturverzeichnis

[1] evoVIU: *Klassische Bildverarbeitung vs Bildverarbeitung mit KI*. https://www.evoviu.de/klassische-bv-vs-bv-mit-ki/
[2] RWTH Aachen: *GITLAB: ev3-toolbox-matlab*. https://git.rwth-aachen.de/mindstorms/ev3-toolbox-matlab
[3] MathWorks: *MATLAB Documentation, edge*. https://de.mathworks.com/help/images/ref/edge.html
[4] Ultralytics: *Bounding Box*. https://www.ultralytics.com/de/glossary/bounding-box
[5] EV3DEV: *EV3 Bluetooth Module*. https://www.ev3dev.org/docs/kernel-hackers-notebook/ev3-bluetooth/

Die Maus

Annika Kristin Ollesch, Elektro-und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Das LEGO-Praktikum 2025 hat als Ziel einen funktionsfähigen Roboter zu bauen, der ein bestimmtes gesetztes Problem löst. In diesem Paper wird das Projekt die Maus genauer betrachtet und Ziele, Funktionsweise sowie Probleme bei der Entwicklung beleuchtet. Die Maus fungiert dabei wie eine echte Maus, die mithilfe von Omniwheels und Bildverarbeitung, ein Loch in einem Raum sucht durch das sie hindurch fahren kann.**

***Schlagwörter*—Omniwheels, Bildverarbeitung, Wegfindung, ComputerVisionToolbox, MATLAB**

I. Einleitung

In der heutigen Zeit arbeiten viele Systeme mit Bildverarbeitung. Durch immer größere Fortschritte wird die künstliche Intelligenz immer weiter entwickelt. Gerade was Systeme wie Brems-, Spurhalte- und Parkassistenzen angeht ist vieles möglich. Um diese Technik zu ermöglichen müssen präzise arbeitende Sensoren und Kameras zum Einsatz kommen. Dafür muss besonders auf eine Echtzeit-Bildverarbeitung und hohe Erkennungsrate von gesuchten Objekten geachtet werden.
Aus diesem Wissen entstand das Projekt „Die Maus“, welche ohne menschliche Hilfe durch reine Bilderkennung einen Ausgang in einem Raum finden soll. Dafür werden auf oben genannte ähnliche bekannte Konzepte zurückgegriffen.

II. Vorbetrachtungen

A. Ziel der Maus

Die Maus soll sich in einem Raum mit vier Wänden bewegen können. Sie soll in dem Raum der aus vier Wänden besteht ein Loch in einer der Wände finden. Dafür dreht sich die Maus 360° um ihre eigene Achse und hält Ausschau nach dem Loch. Findet sie das Loch, fährt sie darauf zu und richtet sich immer wieder passend darauf aus. Dieser Vorgang wird solange durch geführt bis sie durch das Loch in der Wand gefahren ist.

B. Benötigte Bauteile

Für die Konstruktion wurde ein LEGO-Mindstorms Baukasten genutzt. Zentraler Punkt der Konstruktion ist der Steuercomputer, welcher die Motoren und Sensoren ansteuert. Für die Maus wurde ein EV3 verwendet. Um die Bewegungen, wie eine 360° Drehung, auszuführen, welche als Ziel der Maus aufgeführt wurden, werden 3 Omniwheels verwendet. Sie werden in einem Dreieck angeordnet und mit drei kleinen Motoren angesteuert. Um zu erkennen, ob die Maus durch ein Loch gefahren ist, wird ein Ultraschallsensor verwendet. Dieser erkennt durch das aussenden kontinuierlicher Ultraschallsignale Hindernisse. Um zu erkennen, dass die Maus durch das Loch gefahren ist, wurde er an die Decke gerichtet.

DOI: 10.24352/UB.OVGU-2025-021

Der wichtigste Teil der Konstruktion ist die Webcam, welche im späteren verlauf durch ein Smartphone ersetzt wurde und an der Vorderseite des Fahrzeuges angebracht ist. Die Webcam bzw. das Smartphone ist mit zwei Gummibändern an der Konstruktion befestigt.

C. Funktion des Programms

Damit die Maus das Loch findet, wird das Bild der Webcam auf dem Laptop verarbeitet. Um brauchbare Daten aus den gewonnen Bildern zu bekommen, wird in der MATLAB Umgebung mit dem Add-on ComputerVisionToolbox gearbeitet. Das Fahrzeug muss mittels der Kamera ein Rechteck und dessen Lage im Blickfeld erkennen und darauf hin entsprechend reagieren. Um dies zu ermöglichen muss ein starker Schwarz-Weiß-Kontrast zwischen dem Raum und dem Rechteck gegeben sein. Das Fahrzeug wurde dementsprechend in einer weißen Testumgebung mit einem schwarz umrandeten Rechteck getestet.
Die Motoren werden so angesteuert, dass das Fahrzeug entweder rotiert, sich seitwärts oder gerade vorwärts bewegt.
Der nach oben gerichtete Ultraschallsensor misst den Abstand zu der tatsächlichen Raumdecke und erkennt beim Durchfahren des Loches eine verringerte Deckenhöhe. Daraufhin fährt die Maus gerade weiter um den Raum gerade durch das Loch zu verlassen. Um dies zu testen, wurde in der Testumgebung das Stück der niedrigen Decke verbreitert.

III. Entwicklungsprozess

A. Konstruktion mit zwei Omniwheels

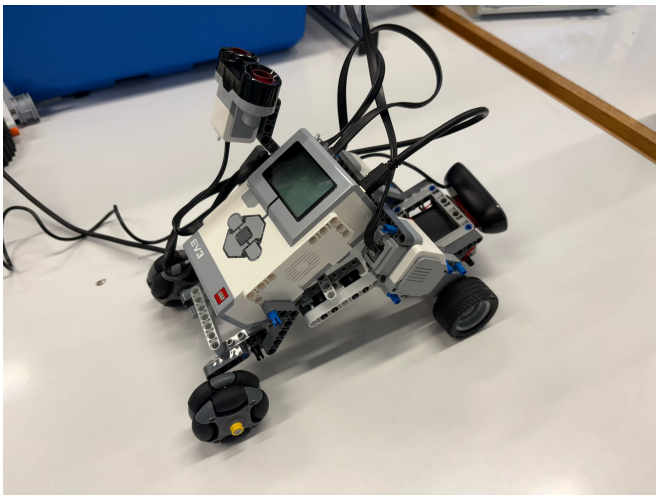


Abbildung 1: Erste Konstruktion mit zwei Omniwheels und USB-Webcam

In der ersten Idee der Konstruktion (Abbildung 1) wurde der EV3-Baustein auf ein rechteckiges Grundgerüst gesetzt. An der Konstruktion wurden zwei große Motoren angebracht die nach vorne ausgerichtet waren. Sie haben den EV3 in eine Schräglage gebracht. An den Motoren wurden normale Räder angebracht.
Schon in der ersten Konstruktion kam der Gedanke an die Omniwheels auf, da die Idee der Maus eine hohe Beweglichkeit des Fahrzeugs benötigt. Als Schlussfolgerung dieser Gedanken wurden am hinteren teil der Konstruktion zwei Omiwheels angebracht.
In dem ersten Teil der Entwicklung wurden außerdem wichtige Schritte zur weiteren Entwicklung getätigt. Die Grundidee den Ausgang in einer Wand mithilfe von Bildverarbeitung zu erkennen kristallisierte sich recht schnell. Für die Umsetzung dieser Vorstellung wurde vorerst eine Webcam verwendet. Dafür wurde sie in eine Halterung aus LEGO mit Gummibändern eingespannt und anschließend auf die Vorderseite der schon bestehenden Konstruktion gebaut.
Der Ultraschallsensor, welcher die Deckenhöhe messen soll, wurde in der ersten Konstruktion an dem EV3 in Richtung Decke befestigt.
Die Konstruktion wurde durch Ansteuern der Motor auf ihre Funktionalität getestet, jedoch musste schnell festgestellt werden, dass eine Rotation zwar möglich ist, aber ein genaueres ausrichten eher schwer fällt da keine wirklichen reinen Seitwärtsbewegungen getätigt werden konnten. Eine weitere Schwäche der Konstruktion war das Kabel der Webcam, welches fixiert werden musste und sich in der Rotationsbewegung immer verdrehte.

B. Konstruktion mit drei Omniwheels

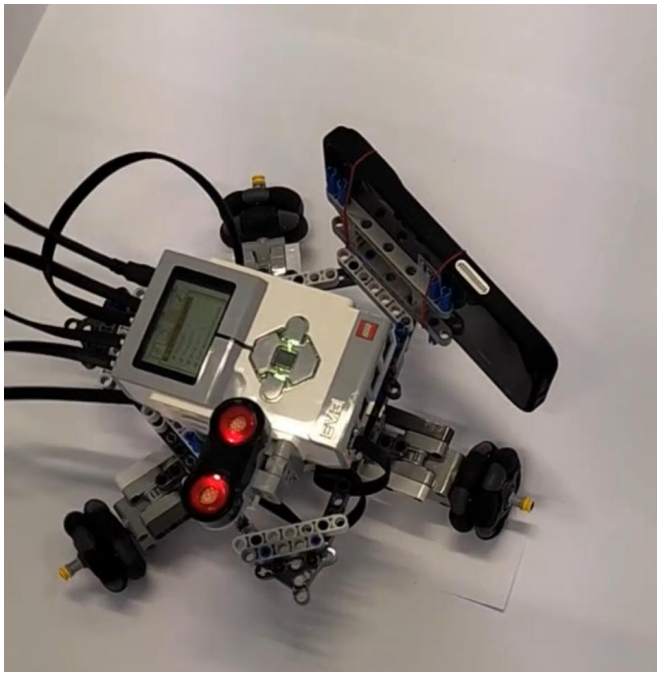


Abbildung 2: Finale Konstruktion der Maus

Da die erste Konstruktion einige neue Herausforderungen und Fragen aufwarf, wurde eine neue Konstruktion entwickelt. Die zweite Konstruktion (Abbildung 2) unterscheidet sich grundlegend von der ersten Konstruktion.
Um mehr Beweglichkeit zu gewährleisten, die in der ersten Konstruktion zu wünschen übrig ließ, kam die Frage nach 4 Omniwheels auf. Schnell wurde, durch eine Onlinerecherche über die Anordnung von Omniwheels, jedoch festgestellt das 3 Omniwheels ausreichend sind.
Daraufhin wurde ein neues Grundgerüst aus LEGO gebaut das die Form eines Dreiecks hat. Die zwei großen Motoren wurden durch drei kleine Motoren ersetzt, um einen kompakten Bau des Fahrzeugs zu ermöglichen. Die Motoren wurden von unten an die Konstruktion so befestigt, dass sie sich in der Mitte fast berühren (siehe Abbildung 3). An den Motoren wurden dann die Omniwheels angebracht.

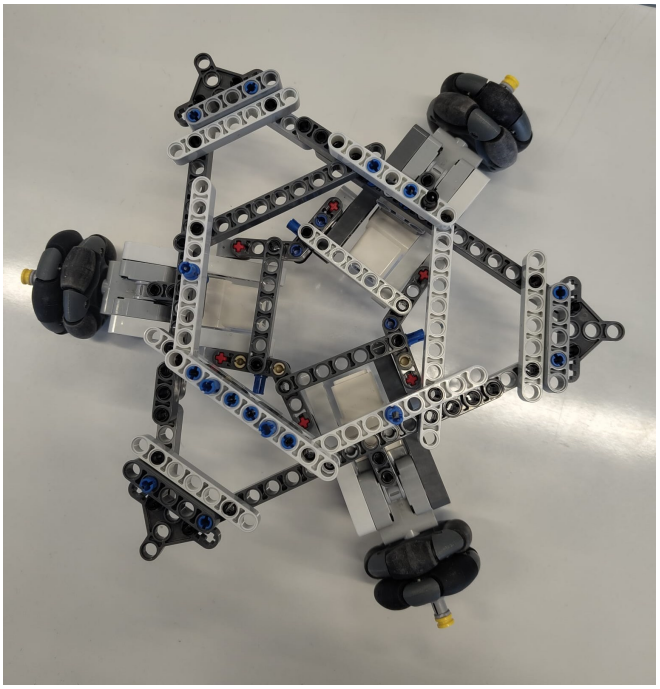


Abbildung 3: Draufsicht auf die Konstruktion mit drei Omniwheels

Eine Schwierigkeit der Omiwheels ist dabei direkt aufgekommen. Diese müssen senkrecht auf den Boden aufkommen, damit ihre Funktionsweise der kleinen seitwärts rollenden Räder nicht beeinträchtigt ist. Stehen sie nicht richtig auf dem Boden liegt das Rad mit dem nicht rollenden Plastikteil der Innenseite auf und bewegt sich nicht. Um dieses Problem zu lösen sind die Motoren untereinander nochmals verbunden. So wird die Stabilität erhöht und sie sinken nicht so sehr in der Mitte des Dreiecks ab.
Ein weiteres Problem mit der Anordnung der 3 Motoren ist, dass die Motoren durch die gerade Anzahl der Seitenlänge der Dreieckskonstruktion nicht ganz mittig von jeder Seite angebracht werden konnte. Sie stehen als nicht im gewollten 120°-Versatz zueinander (siehe Abbildung 4). Dies stellte zum Glück in den Tests kein Problem der Funktionalität dar.

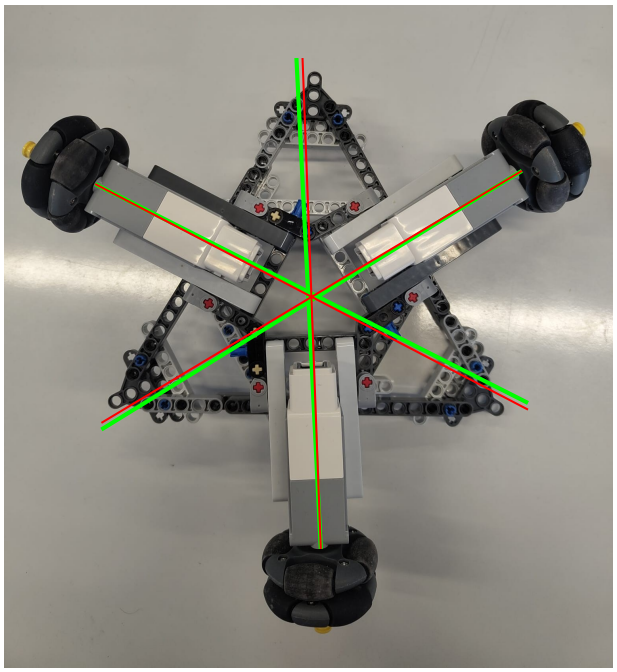


Abbildung 4: Unterseite der zweiten Konstruktion, Idealer 120°-Versatz (rot), Verlängerte Motor-Rad-Achsen (grün)

Die Ultraschallsensor wurde wie auch in der vorherigen Konstruktion wieder am EV3 nach oben gerichtet angebracht. In der ersten Konstruktion ist das Kabel der Webcam negativ aufgefallen, da es ständig im Weg war. Da man in MATLAB auch eine mit WLAN verbundene App verwenden kann ist die Webcam aus der Konstruktion entfallen und stattdessen kam die Idee der Benutzung des Smartphones zur Bildaufzeichnung. Dafür wurde die Halterung der Webcam verworfen und ein LEGO-Bauteil an einer Ecke des Fahrzeuges befestigt. Das Smartphone wird dann mit Gummibändern an diesem Bauteil befestigt.

C. Funktionsweise und Software

Um den Ablauf des Programmes einfach zu verdeutlichen wurde ein Programmablaufplan erstellt (siehe Abbildung 5). Der verwendete EV3 wurde mit der EV3-Toolbox [1] per USB an MATLAB angeschlossen. Das Bild, welches verarbeitet werden soll, wird mit einer auf dem Smartphone installierten App aufgenommen und per WLAN mit dem verwendeten Laptop geteilt. In MATLAB wird diese wie eine normal angeschlossene USB Kamera gesehen.
Zur Bildverarbeitung in MATLAB wurde das Add-on ComputerVisionToolbox auf dem verwendeten Laptop installiert. Um ein Rechteck in einer weißen Wand zu erkennen werden einzelne Snapshots erstellt, welche in MATLAB als eine Matrix gespeichert werden. An dieser wird Kanten und Polygonerkennung durchgeführt. Das Farbbild wird in ein Schwarz-Weiß Bild (Abbildung 6) umgewandelt um den Kontrast höher zu machen und damit das Rechteck mit der Kantenerkennung besser zu erkennen (Abbildung 7).

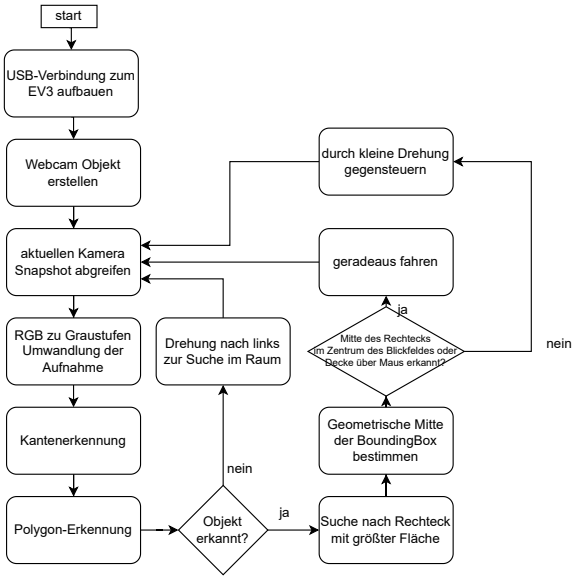


Abbildung 5: Vereinfachtes Ablaufdiagramm

Anschließend wird eine Bounding Box erstellt (Abbildung 8). Die Größe der gefundenen Fläche wird ermittelt. Durch Abgleich der Größe und Seitenverhältnissen können zu kleine oder zu große Rechtecke und sonstige erkannte Objekte heraus gefiltert werden.
Um das Problem der Echtzeit Bildverarbeitung zu lösen, stoppt das Fahrzeug in regelmäßigen Abständen, um ein klares Bild aufnehmen zu können. Das verbleibende Rechteck ist das Loch in der Wand.

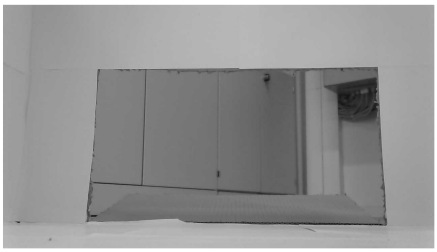


Abbildung 6: Schwarz-Weiß-Bild

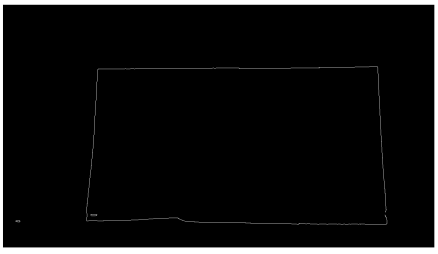


Abbildung 7: Kantenerkennung

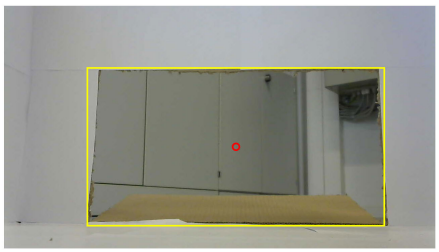


Abbildung 8: Bounding Box

Um die Erkennung umzusetzen muss das Fahrzeug sich vorerst einmal um sich selbst drehen, um den gesamten Raum abzusuchen. Diese Drehung um sich selbst wird durch Ansteuerung in MATLAB der 3 Omniwheels in die gleiche Richtung bei gleicher Leistungszufuhr ermöglicht. Es erfolgt eine Rotation. Durch Rotation wird er immer wieder passend auf das Loch ausgerichtet. Um schließlich auf das Loch gerade zuzufahren muss die Vorwärtsbewegung durch geführt werden. Dies erfolgt durch die Ansteuerung von zwei Motoren in die selbe Richtung.
Befindet sich nun die gefundene Bounding Box nach der Rotation im Sichtbereich der Maus, wird der Mittelpunkt dieser ermittelt. Der Mittelpunkt wurde mit einem Offset berechnet, da die Smartphonekamera nicht mittig auf dem Fahrzeug montiert ist. Wurde keine passende Bounding Box gefunden, dreht sich das Fahrzeug weiter bis es eine findet. Anhand des Mittelpunktes treten drei Möglichkeiten des weiteren vorgehen auf. Befindet sich der Mittelpunkt zentral vor der Maus, fährt sie gerade hindurch. Befindet er sich im rechten oder linken Blickfeld der Kamera wird durch die Rotations- und Vorwärtsbewegung so lange korrigiert, bis die Kamera den Mittelpunkt wieder zentral im Blickfeld hat und das Fahrzeug gerade fahren kann.
Um nicht erneut nach einem Rechteck zu suchen bzw. eine Rotation auszuführen wenn die Maus durch das Loch ist, wird der Ultraschallsensor verwendet. Der Ultraschallsensor misst die Deckenhöhe und erkennt beim Durchfahren der Wand eine deutlich verringerte Deckenhöhe als die übliche Raumdecke. Sollte der Ultraschallsensor diese verringerte Deckenhöhe wahrnehmen, hat die Maus den Auftrag weiter gerade durch das Loch hindurch zu fahren, anstatt eine Rotation auszuführen, weil keine Bounding Box mehr auffindbar ist.

IV. Ergebnisdiskussion

Das Fahrzeug konnte sich am Ende eigenständig aus dem Raum bewegen.

Es wurde jedoch in einer eigens dafür gebauten Testumgebung getestet, die weiße Wände und einen klar markierten Ausgang hatte, um den schwarz-weiß Kontrast möglichst groß zu machen.
Das Problem welches trotz der Testumgebung aufkam, waren unterschiedliche Lichtverhältnisse. Diese beeinflussen das Bild der Kamera und damit auch die Bildverarbeitung.
Ein weiteres Problem war die Umgebung hinter dem Loch der Testumgebung. Wird in der Umgebung ein anderer Gegenstand als Rechteck erkannt, fährt das Fahrzeug nicht frontal auf den Ausgang zu, sondern schräg und bleibt an der Seitenwand hängen. Dies passiert vor allem, wenn sich die Kamera schon sehr nah am Loch befindet und vermehrt die Außenumgebung wahrnimmt.
Das Fahrzeug ruht in regelmäßigen Abständen um die Bewegungsunschärfe der Kamera möglichst gering zu halten. Die Kamera ist durch die vorherige Bewegung jedoch immer noch in leichter Bewegung weshalb trotzdem leichte Bildunschärfe entsteht.
Das Fahrzeug wurde außerdem zwar durch die Smartphonekamera mit einem Kabel weniger an dem Laptop verbunden, jedoch war es nicht möglich die Maus komplett Kabellos zu gestalten. Die Bluetooth-Versionen des EV3-Bausteines und des Laptops lagen zu weit auseinander um ein Verbinden möglich zu machen.

V. Zusammenfassung und Fazit

Abschließend kann man sagen, dass die Maus ein recht erfolgreiches Projekt war, welches mit mehr Zeit und mehr Budget die Möglichkeit hätte in einem größeren Maß, zum Beispiel dem Einparken von Autos in Garagen, oder im kleineren Maß durch ein kleines Gerät, welches für den Menschen unerreichbare Orte eigenständig erkunden kann.
Was in kurzer Zeit mit der Maus umgesetzt werden konnte war nur ein kleiner praktischer Einblick in was möglich wäre. Natürlich hat das Projekt noch Schwachstellen vor allem in der Bildverarbeitung.
Das Projekt war eine gute Möglichkeit über zwei Wochen zu testen, was mit Kreativität und Wissbegierde in MATLAB und mit einem LEGO Baukasten alles geht.

Literaturverzeichnis

[1] RWTH Aachen: *GITLAB: ev3-toolbox-matlab*. https://git.rwth-aachen.de/mindstorms/ev3-toolbox-matlab

„Mindcub4r“ mit Lego Mindstorms

Anton Kresan, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Jedes Jahr findet bei Otto-von-Guericke Magdeburg ein Design - Workshop statt, bei dem Studierende der Ingenieurwissenschaften einen Roboter aus Lego entwerfen.**

Im Rahmen des diesjährigen Projekts wurde ein spezieller Mechanismus entwickelt – der „Mindcub4r“. Dabei handelt es sich um einen Roboter, der gezielte Manipulationen am Rubik's Cube durchführt.

Dieses Projekt zielt darauf ab, zu untersuchen, wie der Roboter mit komplexeren Manipulationen umgehen kann. Durch präzise programmierte Bewegungsabläufe, insbesondere den Pif-Paf-Algorithmus, wird getestet, wie stabil und genau der Roboter die geplanten Sequenzen ausführt.Für die Gestaltung kamen Lego -Mindstorms-Sets und der EV3 -Steuerungscomputer zum Einsatz. Dieser Artikel beschreibt die Struktur und Funktionen des Mechanismus. Auch die bei der Gestaltung aufgetretenen Probleme und deren Lösungen wurden erwähnt.

***Schlagwörter*—Rubik´s Cube, LEGO Mindstorms, MATLAB, Robotik, Pif-Paf-Algorithmus.**

I. Einleitung

DIE Manipulation von Objekten durch autonome Roboter ist ein wichtiger Bereich der modernen Robotik. Der Rubik´s Cube stellt dabei eine ideale Testplattform dar, da er sowohl präzise motorische Bewegungen als auch eine exakte Steuerung erfordert.

Das Ziel dieses Projekts war die Entwicklung eines Roboters, der kontrollierte Manipulationen am Rubik´s Cube ausführen kann. Hierbei wurde der Pif-Paf-Algorithmus (Abbildung 1), als zentrale Bewegungssequenz implementiert . Diese Bewegung ist eine der fundamentalen Sequenzen zur Lösung des Würfels und dient als Basis für viele fortgeschrittene Algorithmen. Durch den Einsatz von LEGO Mindstorms EV3 und MATLAB wurde eine effiziente Steuerung des Roboters realisiert, die eine präzise und wiederholbare Ausführung der Bewegungen gewährleistet.

II. Vorbetrachtungen

In diesem Abschnitt werden die zentralen Werkzeuge dieses Projekts vorgestellt und ihre Funktionen im Zusammenhang mit der Durchführung spezifischer Aufgaben oder der Reaktion auf Eingaben beschrieben.

A. Programmierungumgebung

Die Technologie basiert auf LEGO -Bausteinen, elektronischen Komponenten und spezialisierter Software, die zur Entwicklung und Steuerung autonomer Roboter eingesetzt wird.

Im Rahmen des Projekts wurden zwei mögliche Steuerungssysteme in Betracht gezogen: LEGO Mindstorms EV3 und dessen Vorgängermodell NXT. Nach einer umfassenden

DOI: 10.24352/UB.OVGU-2025-023

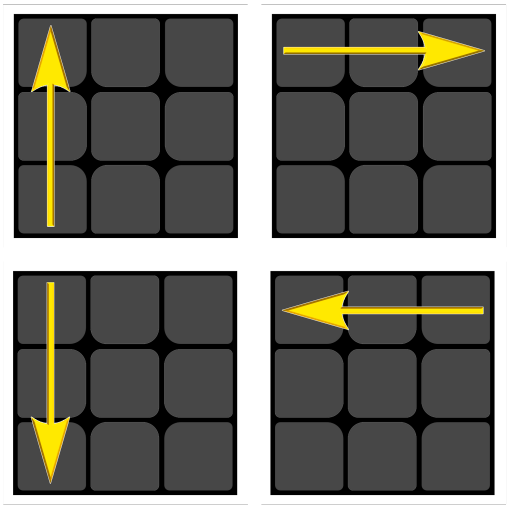


Abbildung 1. Pif-Paf-Algorithmus [1]

Analyse der technischen Spezifikationen und Anforderungen des Projekts fiel die Wahl auf EV3, da es über eine höhere Rechenleistung, präzisere Motorsteuerung und erweiterte Sensorik verfügt. Die Motorik ist präziser und weist eine höhere Auflösung bei Winkelbewegungen auf, was fließende Bewegungen und Drehungen ohne Fehler ermöglicht. Diese Eigenschaften sind entscheidend für die exakte und wiederholbare Durchführung der Manipulationen am Rubik´s Cube.

B. MATLAB als Entwicklungsumgebung

Die Wahl der Programmiersprache spielt eine entscheidende Rolle für die effiziente Steuerung und Entwicklung von Robotersystemen. Für das vorliegende Projekt wurde MATLAB als Hauptprogrammiersprache gewählt, da es eine direkte und leistungsstarke Schnittstelle zur Steuerung des LEGO Mindstorms EV3 bietet.

MATLAB bietet leistungsstarke Funktionen für Vektorrechnung, Matrizenoperationen und Algorithmenoptimierung, die nicht nur bei der Umsetzung präziser Bewegungen von Vorteil sind, sondern auch für die exakte Steuerung eines Roboters und die Speicherung von Daten über den Zustand des Rubik´s Cubes unerlässlich sind. Darüber hinaus ermöglicht MATLAB die separate Programmierung von Sensoren und Motoren, wodurch der Funktionsumfang des Roboters flexibler gestaltet und die Implementierung der Steuerungsalgorithmen erheblich vereinfacht wird.

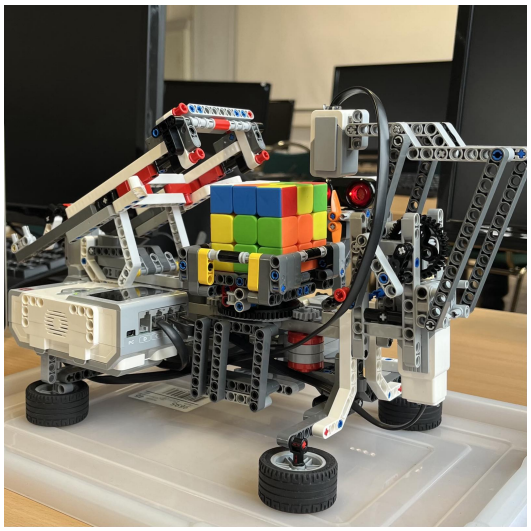


Abbildung 2. Mindcub4r [2]

III. Konstuktion und Programmierung

A. Aufbau

Die Konstruktion des Roboters Mindcub4r basiert auf einer durchdachten und zuverlässigen mechanischen Struktur, die eine präzise Steuerung aller Komponenten gewährleistet.

Die Anordnung der Motoren und Sensoren wurde so gestaltet, dass gegenseitige Beeinträchtigungen vermieden werden, um einen effizienten und reibungslosen Bewegungsablauf zu gewährleisten. Zudem wurden vier zentrale Mechanismen entwickelt, um die vollständige Funktionalität des Roboters sicherzustellen und eine präzise Manipulation des Rubik's Cubes zu ermöglichen. (Abbildung 2).

Im Zentrum befindet sich eine Plattform, auf der Rubik´s Cube fixiert und gedreht wird. Links wurde ein Greifmechanismus integriert, der die anspruchsvollsten Manipulationen ausführt, darunter das Umklappen und Fixieren des Würfels. Rechts befindet sich ein ausfahrbarer Mechanismus mit einem Farbsensor, der zur Farbenerkennung genutzt wird. Ergänzend wurde auf der Rückseite ein Ultraschallsensor platziert, der die Positionserkennung unterstützt. Diese kompakte und stabile Konstruktion ermöglicht eine präzise Steuerung und effiziente Manipulation des Würfels.

1) Motoren: Wie bereits zuvor erwähnt, spielen die Motoren eine zentrale Rolle in der Funktionsweise des Roboters. Zwei große Motoren (Abbildung 3) sind für die Rotation der Plattform sowie die Bewegungen des Greifmechanismus verantwortlich . Ein mittlerer Motor (Abbildung 4) steuert den ausfahrbaren Mechanismus mit dem Farbsensor . Diese gezielte Ansteuerung der Motoren gewährleistet eine präzise und zuverlässige Durchführung der Manipulationsprozesse.

2) Ultraschallsensor Sensor: Die gesamte Programmsequenz startet erst, wenn der Ultraschallsensor (Abbildung 5) bestätigt, dass der Rubik´s Cube korrekt auf der Plattform liegt . Er fungiert als zentrales Kontrollsystem, das die Startbedingung überwacht und erst nach erfolgreicher Erkennung die Manipulationsabläufe freigibt. Dadurch wird sichergestellt, dass alle Bewegungen präzise und fehlerfrei ausgeführt werden.

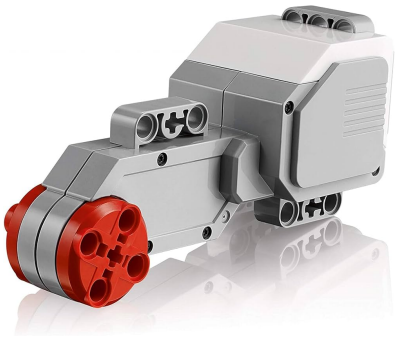


Abbildung 3. Großer Motor [3]

Abbildung 4. Mittlerer Motor [4]

3) Farbsensor: Der Farbsensor (Abbildung 6) fungiert als Informationsüberträger, der sämtliche Daten über die einzelnen Flächen des Rubik-Würfels an den Mikrocontroller weiterleitet . Basierend auf diesen erfassten Informationen und unter Anwendung des im Code implementierten Algorithmus führt der Roboter präzise Manipulationen am Würfel durch. Diese Datenverarbeitung ermöglicht eine gezielte Steuerung der Bewegungsabläufe und gewährleistet eine exakte Umsetzung der programmierten Sequenz.

B. Programmierung

Die korrekte und präzise Funktionsweise des Roboters hängt nicht nur von seiner mechanischen Konstruktion, sondern auch von einer strukturierten und fehlerfreien Programmierung ab. Um eine klare und systematische Steuerung zu gewährleisten,

Abbildung 5. Ultraschallsensor [5]

Abbildung 6. Farbsensor [6]

wurde der Code für jede Bewegung jedes Mechanismus separat programmiert. Besondere Steuerungsalgorithmen wurden für die Sensoren entwickelt, um eine präzise Datenerfassung und Verarbeitung zu ermöglichen. Ein herausragendes Beispiel ist der Farbsensor, dessen erfasste Daten in sechs 6×6-Matrizen überführt werden. Diese Matrizen repräsentieren die sechs Seiten des Rubik´s Cubes, wobei jeder einzelne Würfelstein eine spezifische Position innerhalb der Matrix einnimmt. Die gescannten Farbinformationen werden in der MATLABKonsole visualisiert, wodurch eine genaue Analyse und Weiterverarbeitung der Daten ermöglicht wird. Unten ist der Piff-Paff-Algorithmus. Jede Zeile des Algorithmus ist eine separate Manipulation, für die ein separater Code geschrieben wird (Abbildung 7).

Aus dem Algorithmus ist ersichtlich, dass nur 2 Motoren beteiligt sind. Beispielsweise hat Motor B separate Codes vom Algorithmus.Die Programme „turnWhenHold“ (Abbildung 8) und „turnLeftWhenHold“ sind dafür zuständig, die Plattform mit dem Zauberwürfel um jeweils 90 Grad nach rechts bzw. links zu scrollen.Man kann auch feststellen, dass im Programmalgorithmus Motor B viermal vorkommt. Das bedeutet, dass die Ränder so oft nach rechts oder links gescrollt werden müssen, um einen Piff Paff zu erhalten.

Programme mit Motor C sind ausschließlich für die Bewegungen der Greifzange zuständig. Das Programm „hold“ sorgt dafür, dass die Greifzange den Würfel fixiert. Das Programm „flipAfterHold“ dreht den Zauberwürfel wiederum mit umgekehrter Geschwindigkeit in eine andere Position. Diese Programme werden mit dem Programm für die Plattform kombiniert. Dadurch wird der Zauberwürfel fixiert, die Plattform dreht die Fläche, und anschließend kippt die Greifzange den Zauberwürfel um. Das Programm „flipOver“ (Abbildung 9) führt die Befehle „hold“ und „flipAfterHold“ zusammen aus.

```
clear all;
ev3 = legoev3('usb');
motorC = motor(ev3, 'C');
motorB = motor(ev3, 'B');
for i = 1:6
hold(motorC);
turnWhenHold(motorB);
flipAfterHold(motorC);
hold(motorC);
turnWhenHold(motorB);
flipAfterHold(motorC);
flipOver(motorC);
flipOver(motorC);
hold(motorC);
turnLeftWhenHold(motorB);
flipAfterHold(motorC);
hold(motorC);
turnLeftWhenHold(motorB);
flipAfterHold(motorC);
flipOver(motorC);
flipOver(motorC);
end
```

Abbildung 7. Algorithmus

```
function turnWhenHold(motorB)
pause(0.2);
resetRotation(motorB);
motorB.Speed = -49.5;
start(motorB);
while abs(readRotation(motorB)) < 14/16 * 360
pause(0.4);
end
motorB.Speed = 0;
pause(0.3); stop(motorB);
resetRotation(motorB);
motorB.Speed = 40;
start(motorB);
while abs(readRotation(motorB)) < 14/16*77.5
pause(0.18);
motorB.Speed = 0;
pause(0.1);
stop(motorB);
pause(1);
end
```

Abbildung 8. turnWhenHold

C. Probleme

Die Entwicklung eines Roboters zum Lösen des Rubik's Cube brachte unerwartete technische Herausforderungen mit sich. Obwohl der Roboter über alle notwendigen Mechanismen zur Manipulation des Würfels verfügte, traten erste Schwierigkeiten bei der Farberkennung auf. Bei Tests mit zwei Farbsensoren stellte sich heraus, dass vier von sechs Farben fälschlicherweise als Blau erkannt wurden. Versuche, die Farberkennung durch Anpassung der Wertebereiche zu verbessern, führten nur zu begrenzten Erfolgen: Während das Scannen einzelner der neun Felder auf einer Würfelseite akzeptable Ergebnisse lieferte, versagte der Sensor bei rotierender Plattform in der korrekten Identifizierung aller Farben.

Um dieses Problem zu umgehen, wurde beschlossen, den Roboter so zu programmieren, dass er standardisierte algorithmische Sequenzen, bekannt als "Pif-Paf", ausführt, ohne auf vollständige Farberkennung angewiesen zu sein.

Zusätzlich wurde der Roboter mit einem ausfahrbaren Mechanismus mit Farbsensor ausgestattet, der ihm ermöglicht,

```
function flipOver(motorC)
motorC.Speed = -30.5;
motorC.start();
pause(2);
motorC.stop();
motorC.Speed = 27;
motorC.start();
pause(1.5);
motorC.stop();
end
```

Abbildung 9. flipOver

seine Funktionalität zu erhalten und seine Fähigkeiten zu demonstrieren.

Ein weiteres Konstruktionsproblem betraf die Plattform. Diese war zunächst auf drei kleinen Zahnrädern montiert, erwies sich jedoch als mechanisch instabil und unzuverlässig. Zur Verbesserung der Funktionssicherheit wurde die ursprüngliche Anordnung durch eine Kombination aus einem großen und zwei kleineren Zahnrädern ersetzt. Nach Umsetzung dieser Modifikation zeigte das System eine deutlich gesteigerte Zuverlässigkeit, Präzision und Betriebsstabilität.

IV. Ergebnisdiskussion

Zusammenfassend lässt sich feststellen, dass der Roboter erfolgreich in der Lage ist, die “Pif-Paf”-Algorithmen eigenständig auszuführen. Für das vollständige Lösen des Rubik’s Cube ist jedoch eine präzise Farberkennung jeder Seite erforderlich. Obwohl die aktuellen Farbsensoren gewisse Einschränkungen aufweisen, bietet dies eine hervorragende Gelegenheit für zukünftige Optimierungen. Mit gezielten Verbesserungen im Farberkennungssystem kann der Roboter das Potenzial entfalten, den Rubik’s Cube vollständig autonom zu lösen.

Literaturverzeichnis

[1] Cubcorner, Dzen: *Rubic’s Cube 3*3*3*. https://dzen.ru/a/YioqZz1XWGFx5YWy. Version: 2018. – Accessed: 2025-03-13

[2] Instagram: *Mindcub4r*. https://www.instagram.com/ovgu_projektseminar?igsh=cWJkeG9pdzdsa3Jo. Version: 2025. – Accessed: 2025-03-13

[3] Amazon: *LEGO MINDSTORMS Education EV3 Servomotor groß*. https://www.amazon.de/EDUCATION-MINDSTORMS%C2%AE-Education-Gro%C3%9Fer-EV3-Servomotor/dp/B00E1QDP4W. Version: 2025. – Accessed: 2025-03-13

[4] Amazon: *LEGO MINDSTORMS Education EV3 Medium Servomotor*. https://www.amazon.de/LEGO-MINDSTORMS-Education-Servomotor-45503/dp/B00E1Q5NBU. Version: 2025. – Accessed: 2025-03-13

[5] Amazon: *LEGO® MINDSTORMS® Education EV3 Ultraschallsensor*. https://www.amazon.de/LEGO-MINDSTORMS-Education-Ultraschallsensor-45504/dp/B00E1PTRAE. Version: 2025. – Accessed: 2025-03-13

[6] Amazon: *LEGO MINDSTORMS Education EV3 Farbsensor*. https://www.amazon.de/LEGO-MINDSTORMS-Education-Farbsensor-45506/dp/B00E1QNH2W. Version: 2025. – Accessed: 2025-03-13

LEGO Mindstorms Roboter-Cardsdealer

Hlieb Khramov, Studiengang
Otto-von-Guericke-Universität Magdeburg

Ein Dokument, das das Projekt, dessen Grundidee und Funktionalität beschreibt sowie zentrale Entscheidungen während der Entwicklung des Roboters darlegt. Darüber hinaus enthält es eine Übersicht über die wesentlichen Herausforderungen bei der Konstruktion des Roboters und der Programmierung. Ergänzend sind die verwendeten Quellen und Programme aufgeführt, die im Rahmen der Projekterstellung zum Einsatz kamen.

I. Einleitung

DIESES Projekt ist ein Roboter zum Kartenspielen, der mit MATLAB-Programmen und dem LEGO Mindstorm-Konstruktor erstellt wurde. Die Idee dieses Roboters besteht darin, den Prozess des Kartenausteilens zu beschleunigen und außerdem Fehler beim Kartenausteilen zu minimieren, wenn beispielsweise einem der Spieler eine Karte mehr ausgeteilt wird. Das Ziel war es außerdem, einen Roboter mit veränderlichen Einstellungen zur Kartenverteilung zu entwickeln, der bei der Auswahl der Spielerzahl, der Kartenzahl und der Spielart selbst flexibler ist. Für dieses Projekt wurden MATLAB-Software, LEGO TECHNIK- und LEGO Mindstorm-Teile, während der Arbeit am Projekt gewonnenes Wissen sowie die Idee des Roboters selbst benötigt.

II. Vorbetrachtungen

Die Erstellung des Projekts wurde in drei Phasen unterteilt: Die erste Phase ist die Idee, die zweite Phase ist die Konstruktion, die dritte Phase ist die Programmierung.

A. *Idee*

Ursprünglich bestand die Idee darin, einen Roboter zur Steuerung eines Telefons zu bauen, der durch die Feeds sozialer Netzwerke scrollen und Fotos liken oder disliken sollte. Diese Idee wurde jedoch verworfen, da der Roboter nur über wenige Funktionen verfügte und Fotos nur anhand der Farbe erkennen konnte. Danach wollten wir zusammen einer anderen Gruppe und mache ein Projekt mit zwei schnell bewegende Roboter, wie bei einem Rennen. Auch diese Idee war schlecht, da die Maximalgeschwindigkeit aller Motoren gleich ist und daher auch die Geschwindigkeit der Roboter gleich sein wird. Am Ende entschied man sich, den eigenen Roboter zu bauen, der die Karten an die Spieler verteilt. Das Gute an diesem Roboter ist, dass er über zahlreiche Funktionen verfügt, wie etwa eine 270-Grad-Drehung, um allen Spielern Karten zu geben, eine Schaltfläche, um einem Spieler eine separate Karte zu geben, und auch die Möglichkeit, die Anzahl der Karten und Spieler zu ändern.

B. *Konstruktion*

Der Bau des Roboters brachte viele Herausforderungen mit sich, beispielsweise die Entwicklung eines Mechanismus zur Kartenausgabe. Der Mechanismus muss guten Kontakt mit der Karte haben, um nur eine Karte pro Motorumdrehung zu produzieren. Außerdem gab es ein Problem mit dem Gewicht des Turms: Er war zu schwer und konnte sich nicht um 270 Grad drehen. Auch die Kabel vom Motor zum Hauptbedienfeld des Roboters störten und mussten außen verlegt werden, da im Körper des Roboters nur wenig Platz für sie vorhanden war.

C. *Programmierung*

Beim Programmieren des Roboters stieß man auf nur ein Problem – das Fehlen einiger Plugins für den korrekten Betrieb des Roboters und seiner Sensoren mit dem MATLAB-Programm.

III. Konstruktion und Programmierung

Im Hauptteil werden das Konstruktion und die Realisierung erläutert.

Abbildung 1 zeigt den im Rahmen dieses Projekts hergestellten Roboter.

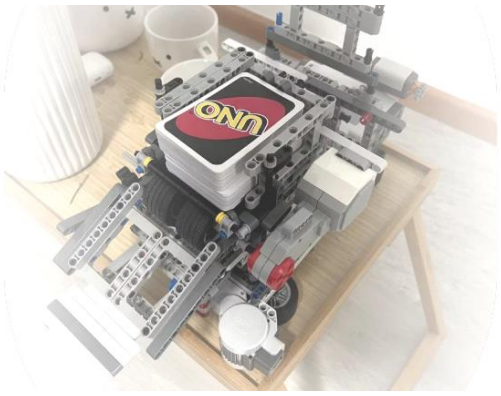


Abbildung 1: Roboter

A. *Konstruktion*

Die Abbildung 2, 3 und 4, zeigen den Mechanismus zur Ausgabe der Roboterkarten von verschiedenen Seiten.

DOI: 10.24352/UB.OVGU-2025-024

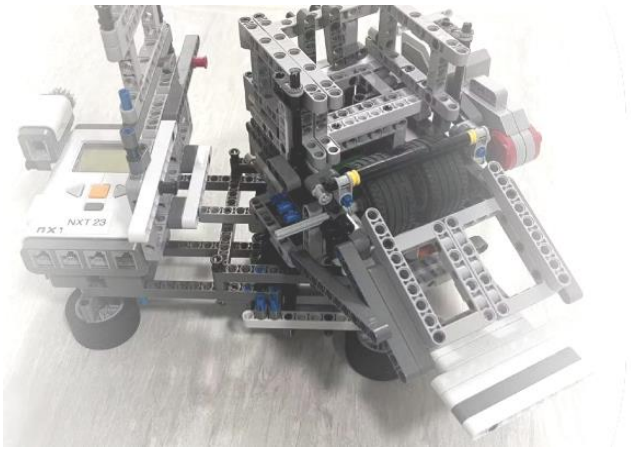


Abbildung 2: Blick auf den Mechanismus aus einem Winkel

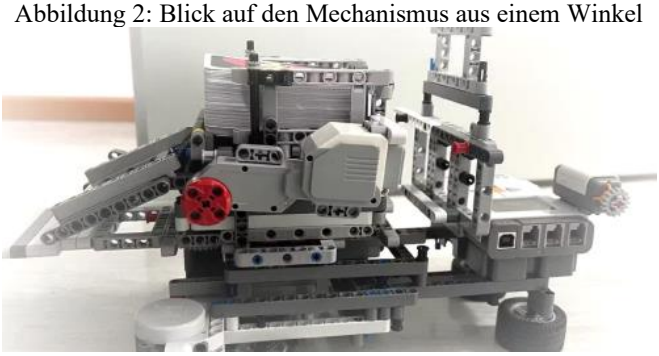


Abbildung 3: Blick auf den Mechanismus von der Seite

Abbildung 4: Blick auf den Mechanismus von oben

Um diesen Mechanismus zu implementieren, waren zwei Motoren erforderlich: einer zum Drehen des Turms, der zweite für den Kartenausgabemechanismus. Im Turm selbst haben wir einen Mechanismus zur Kartenausgabe gebaut, wofür wir ein Paar Lego-Räder verwendet haben. Sie waren ideal, um festen Kontakt mit der Karte herzustellen. Zusätzlich wurde für die Karten eine Hülle angefertigt, damit diese nicht herausfallen oder sich bewegen können, da hierdurch auch die Kartenausgabe beeinträchtigt wird.

B. Programmierung

Abbildung 5 zeigt ein vereinfachtes Flussdiagramm des Robotercodes, das den Start, die Prüfung und die Ausführung der Hauptfunktionen des Roboters beschreibt.

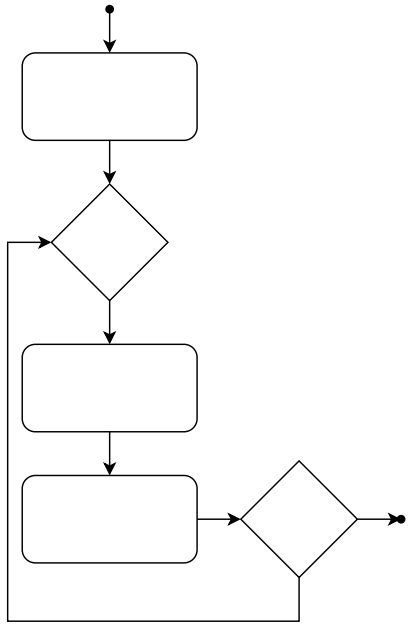


Abbildung 5: Flussdiagramm

Ein Beispiel unseres Hauptcodes für den Kartenausgabemechanismus, erstellt mit der For-Schleife.

```
for j = 1:a
  for k = 1
    motorA.Power = 40;
    if c == 1
      motorA.TachoLimit = 450;
    elseif c == 2
      motorA.TachoLimit = 350 * c;
    elseif c == 3
      motorA.TachoLimit = 300 * c;
    else
      motorA.TachoLimit = 265 * c;
    end

    motorA.SendToNXT();
    motorA.WaitFor(c);
    motorA.Stop('off');

    motorA.Power = -50;
    cotorA.TachoLimit = 500;
    motorA.SendToNXT();
    motorA.WaitFor(1);
    motorA.Stop('off');
  end

  motorB.Power = -15;
  motorB.TachoLimit = round(angle_per_player);
  motorB.SendToNXT();
  motorB.WaitFor(1);
  motorB.Stop('off');
end
```

Außerdem ist hier der Code für den Sensor zur Kartenausgabe nach Zyklusende auf dem Knopf vorhanden.

```
while true
 sensorState = GetSwitch(SENSOR_4);
if sensorState == 1
   press_time = tic; % Запоминаем момент нажатия

   % Ждём, пока кнопка удерживается
   while GetSwitch(SENSOR_4) == 1
      elapsed_time = toc(press_time);
      if elapsed_time > 1
         break;
      end
   end

   if elapsed_time > 0.3

      disp('Double!');
      motorA.Power = 90;
      motorA.TachoLimit = 310 * 2;
      motorA.SendToNXT();
      motorA.WaitFor(1.5);
      motorA.Stop('off');

      motorA.Power = -100;
      motorA.TachoLimit = 200;
      motorA.SendToNXT();
      motorA.WaitFor(1);
      motorA.Stop('off');
   else
      % Если кнопка удерживалась менее 1 секунды – выдаём 1 карту
      disp('One card!');
      motorA.Power = 100;
      motorA.TachoLimit = 360;
      motorA.SendToNXT();
      motorA.WaitFor(1);
      motorA.Stop('off');

      motorA.Power = -100;
      motorA.TachoLimit = 200;
      motorA.SendToNXT();
      motorA.WaitFor(1);
      motorA.Stop('off');
   end

   % Ждём, пока пользователь отпустит кнопку
   while GetSwitch(SENSOR_4) == 1
   end
end
```

IV. Ergebnisdiskussion

Das Endergebnis dieses Projekts ist die einwandfreie Bedienung aller oben beschriebenen Funktionen des Roboters sowie die hochwertige Montage von Mechanismen für die weitere Nutzung des Roboters. Während der Arbeit an dem Projekt sind uns viele Fehler und Probleme begegnet, wie beispielsweise der schwere Roboterturm, der Mechanismus zur Kartenausgabe, der Platzmangel für das USB-Kabel usw. Auch bei der Installation von Plugins auf dem Computer und beim Arbeiten mit Batterien gab es Probleme und man musste sparsam mit ihnen umgehen, da der Roboter viel Energie verbrauchte und die Batterien schnell leer waren.

V. Zusammenfassung und Fazi

Durch die Konstruktion und Programmierung des Roboters konnten Kenntnisse im Umgang mit LEGO-Komponenten, im Entwurf komplexer mechanischer Strukturen sowie in der Anwendung von MATLAB zur Programmierung vertieft werden. Darüber hinaus wurde der Entwicklungsprozess durch fortlaufendes Testen und Optimieren der Mechanismen unterstützt.

Zur weiteren Verbesserung des Roboters bietet es sich an, eine Mobilitätsfunktion durch den Einbau eines Fahrwerks zu integrieren, um die Einsatzmöglichkeiten zu erweitern. So könnte der Roboter beispielsweise auf einem Tisch navigieren und dabei eigenständig Spielkarten austeilen, etwa bei einem Pokerspiel. Auch eine Kartenmischfunktion ließe sich umsetzen, was jedoch den Einsatz von zwei zusätzlichen Motoren erfordern würde.

Literaturverzeichnis

[1] Meets MINDSTORMS: How to Control LEGO NXT Robots Using MATLAB for Educational Purposes
[2] Required Equipment - MATLAB LEGO Mindstorms NXT 2.0 Instruction Manual
[3] Projektseminar Elektrotechnik/Informationstechnik (LEGO Mindstorms)
[4] Video mit dem Titel LEGO Card Shuffler and Dealer

LEGO Mindstorms Roboter-Cardsdealer

Volodymyr Kornev, Elektrotechnik und Informationstechnikg
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Zusammenfassung—Im Rahmen des Projekts wurde ein LEGO Mindstorms Roboter entwickelt, der das Austeilen von Spielkarten automatisiert. Der „Cardsdealer"-Roboter ermöglicht eine fehlerfreie und schnelle Kartenverteilung mit anpassbaren Parametern wie Spieleranzahl und Kartenzahl. Die Konstruktion stellte verschiedene Herausforderungen dar, insbesondere die Entwicklung eines zuverlässigen Kartenausgabemechanismus und die Integration der Komponenten innerhalb der begrenzten Struktur. Die Programmierung erfolgte in MATLAB, wobei einige technische Hürden, wie fehlende Plugins, überwunden werden mussten. Das Projekt demonstriert die Machbarkeit eines präzisen und flexiblen Kartenausteil-Roboters und eröffnet Potenzial für zukünftige Erweiterungen, etwa eine Mobilitätsfunktion oder eine Kartenmischoption.**

***Schlagwörter*—LEGO Mindstorms, MATLAB, Robotik, Kartenverteiler, Mechanik, Automatisierung.**

I. Einleitung

DIEses Projekt ist ein Kartenspiel-Roboter, entwickelt mit MATLAB-Programmen und dem LEGO Mindstorms-Baukasten. Ziel ist es, das Kartenausteilen zu beschleunigen und Fehler zu vermeiden, etwa das falsche Verteilen der Karten. Der Roboter bietet anpassbare Einstellungen für Spieleranzahl, Kartenzahl und Spielart. Zur Umsetzung wurden MATLAB-Software, LEGO Technic- und LEGO Mindstorms-Teile sowie das während der Entwicklung gewonnene Wissen genutzt.

II. Vorbetrachtungen

Die Entwicklung des Projekts erfolgte in drei Phasen: die Ideenfindung, die Konstruktion und die Programmierung.

A. Idee

Ursprünglich wurde die Idee verfolgt, einen Roboter zu entwickeln, der ein Telefon steuert, durch soziale Netzwerke scrollt und Fotos likt oder dislikt. Diese Idee wurde jedoch verworfen, da der Roboter nur wenige Funktionen hatte und Bilder nur anhand der Farbe erkennen konnte. Anschließend wurde in Betracht gezogen, ein Projekt mit zwei schnellen Robotern zu realisieren. Auch dieser Ansatz erwies sich als unpraktikabel, da die maximale Geschwindigkeit aller Motoren identisch ist und die Roboter somit gleich schnell wären. Schließlich fiel die Entscheidung auf die Entwicklung eines Roboters, der Karten an Spieler verteilt. Sein Vorteil liegt in den vielseitigen Funktionen, darunter eine 270-Grad-Drehung für die Kartenverteilung, eine Schaltfläche zum separaten Austeilen von Karten und die Möglichkeit, die Anzahl der Spieler und Karten anzupassen.

DOI: 10.24352/UB.OVGU-2025-025

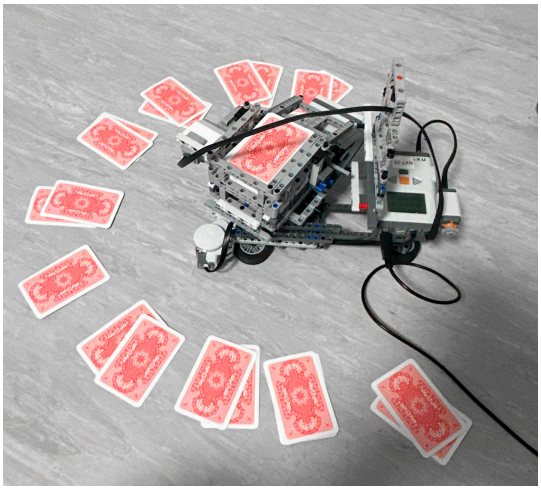


Abbildung 1. Reales Foto: Roboter-Kartenverteiler

B. Konstruktion

Der Bau des Roboters stellte mehrere Herausforderungen dar, insbesondere die Entwicklung eines zuverlässigen Kartenausgabemechanismus. Ein LEGO-Rad mit Gummireifen sollte die Karte greifen und durch Drehung ausgeben. Ein weiteres Problem war das hohe Gewicht des Turms, da sich darauf eine massive Konstruktion, ein Kartenstapel und ein Motor befanden. Dies führte zu Ungenauigkeiten bei der 270-Grad-Drehung, weshalb der Drehmechanismus verstärkt werden musste, was viele LEGO-Teile erforderte. Zudem war im Inneren des Roboters wenig Platz für die Kabel, sodass sie extern verlegt werden mussten.

Abbildung 2. LEGO-Rad für die Kartenausgabe

C. Programmierung

Bei der Programmierung des Roboters trat lediglich ein Problem auf – das Fehlen einiger Plugins, die für den korrekten Betrieb des Roboters und seiner Sensoren mit dem MATLAB-Programm erforderlich sind. Abgesehen davon gab es keine besonderen Schwierigkeiten bei der Umsetzung des Programmcodes, da der Algorithmus dieses Roboters relativ einfach ist. Ganz anders verhielt es sich jedoch mit der Konstruktion, die deutlich mehr Zeit in Anspruch nahm.

III. Konstruktion und Programmierung

Die Konstruktion des Roboters wurde so entwickelt, dass sie eine stabile Funktion des Kartenverteilungsmechanismus gewährleistet. Der Hauptbestandteil des Roboters ist die Basisplattform, auf der sich der Drehmotor für den Turm sowie der NXT-Controller befinden. Ursprünglich war eine 360-Grad-Drehung geplant, jedoch wurde diese in der Praxis auf 270 Grad begrenzt. Diese Entscheidung erwies sich als Vorteil, da sie Probleme mit den Kabeln verhinderte und eine präzisere Kartenverteilung ermöglichte. [1]

Der Turm bildet das zentrale mechanische Element des Roboters. Er hat eine Box-Konstruktion, die speziell an die Größe eines UNO-Kartenstapels 4 angepasst wurde. Dies gewährleistet eine sichere Lagerung und kontrollierte Ausgabe der Karten. Unter den Karten befinden sich zwei LEGO-Räder, die mit einem seitlich montierten Motor verbunden sind.

Ein wichtiges Designelement war ein zweites Radpaar, das nicht mit dem Motor verbunden ist, aber eine entscheidende Funktion erfüllt: Es verhindert die versehentliche Ausgabe mehrerer Karten gleichzeitig. Die Karte befand sich zwischen den beiden Rädern und dem rotierenden Mechanismus, was die Genauigkeit der Kartenausgabe erheblich verbesserte. Diese Lösung reduzierte Fehler bei der Kartenausgabe auf null.

Der Programmablauf [2]war relativ einfach, da die Hauptarbeit in der Konstruktion lag. Die Software ermöglichte es dem Benutzer, die Anzahl der Spieler und die Anzahl der Karten pro Spieler festzulegen. Anschließend wurde der Drehwinkel des Turms unter Berücksichtigung des Übersetzungsverhältnisses berechnet. Der Turm bewegte sich zuerst in die Nullposition, bevor die Kartenausgabe begann.

Eine effektive programmiertechnische Lösung war die automatische Korrektur zur Vermeidung der Ausgabe einer zusätzlichen Karte. Nach der Kartenverteilung für einen Spieler führte der Motor eine kurze Rückdrehung durch, um eine möglicherweise versehentlich ausgegebene Karte zurück in das Magazin zu befördern.

Nach Abschluss der Verteilung kehrte der Turm automatisch in die Zentralposition zurück.

Auf der Basisplattform des Roboters wurde ein Tastsensor installiert, der eine manuelle Kartenausgabe ermöglichte:

Kurzes Drücken – Ausgabe einer Karte. Langes Drücken – Ausgabe von zwei Karten. Dies macht den Roboter vielseitig einsetzbar und erlaubt sowohl eine automatische als auch manuelle Steuerung.

In Abbildung 3 ist das Bild des fertig entwickelten Roboters dargestellt.

Der folgende Code zeigt die Implementierung der Steuerung für den LEGO-Roboter in MATLAB. In Abbildung 3 ist das Bild des fertig entwickelten Roboters dargestellt.

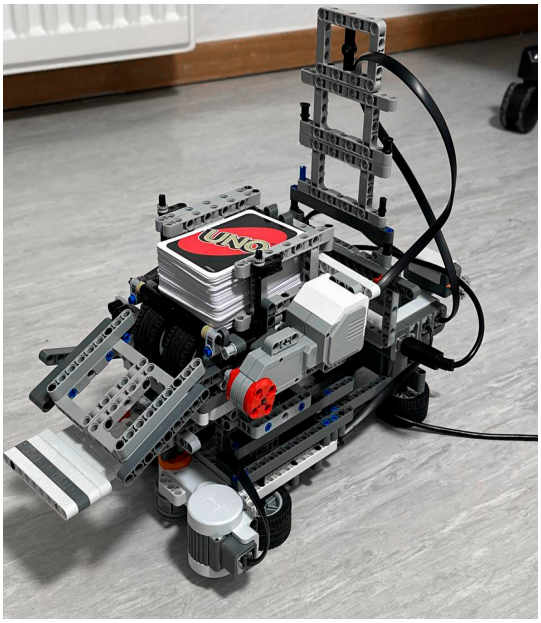


Abbildung 3. Der entwickelte LEGO-Roboter

Abbildung 4. UNO-Kartenspiel

```
a = input('Enter the number of players: ');
c = input('Enter the number of cards per player: ');

angle_per_player = round(270 / gear_ratio) / a;
gear_ratio = 0.666;
```

Listing 1. MATLAB-Code: Teilung des Drehwinkels auf die Anzahl der Spieler

IV. Ergebnisdiskussion

Als Endergebnis wurde ein voll funktionsfähiger Roboter entwickelt, der über eine interessante und zuverlässige Programmfunktionalität verfügt. Das System arbeitet fehlerfrei, da das Hauptziel darin bestand, mögliche Fehler auf ein Minimum zu reduzieren und eine zuverlässige Ausführung der Aufgaben sicherzustellen. [3]

V. Zusammenfassung und Fazit

Zusammenfassend lässt sich sagen, dass das Projekt erfolgreich war und seine Umsetzung nicht besonders aufwendig war. Der Roboter verfügt über alle ursprünglich geplanten Funktionen und sogar über zusätzliche Features. Eine mögliche Verbesserung wäre die Optimierung der Kartenverteilung: Anstatt alle Karten nacheinander an einen Spieler auszugeben und dann zum nächsten zu wechseln, könnte der Roboter die Karten in Runden verteilen – also jeweils eine Karte pro Spieler, bis alle Karten ausgegeben sind. Dies würde zu einer gleichmäßigeren Verteilung führen und die Effizienz des Mischvorgangs verbessern. [4]

Literaturverzeichnis

[1] MathWorks: *Meets MINDSTORMS: How to Control LEGO NXT Robots Using MATLAB for Educational Purposes*. https://de.mathworks.com/videos/matlab-meets-mindstorms-how-to-control-lego-nxt-robots-using-matlab-for-educational-purposes-90418.html. Version: 2025. – Accessed: 18 March 2025

[2] *Required Equipment - MATLAB LEGO Mindstorms NXT 2.0 Instruction Manual*. : *Required Equipment - MATLAB LEGO Mindstorms NXT 2.0 Instruction Manual*, 2025. https://www.manualslib.com/manual/3545144/Matlab-Lego-Mindstorms-Nxt-2-0.html?page=5#manual. – Accessed: 2025-03-19

[3] INSTAGRAM: *LEGO Mindstorms NXT - Instagram Page*. https://www.instagram.com/legomindstormnxtgrup10/. Version: 2025. – Accessed: 2025-03-19

[4] Elektrotechnik/Informationstechnik, Projektseminar: *LEGO Mindstorms Seminar*. https://elearning.ovgu.de/course/view.php?id=178. Version: 2025. – Accessed: 2025-03-19

Self-Balancing Robot on a Ball

Der Roboter mit der Gelegenheit einer weiteren innovativen Entwicklung, der die Menschheit zukünftig assistieren kann.

Artem Hrach, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung* — Die Hauptidee dieses Projekt war die Entwiklung eines Roboters, der während des Projektseminares für Elektrotechnik/Informationstechnik 2025 aus Legobauteile(LEGO-Mindstorms) gebaut wurde und auf einem Ball balancieren kann, um weitere technische Entwicklungen in der Robotik zu fördern und den Menschen zu helfen.**

Das Programm, das dieser Roboter in Bewegung setzen konnte, wurde in MATLAB geschrieben. Mit Hilfe des Programms kann Der Roboter jeden seiner nächsten Schritte erfassen, was besonders wichtig ist, da seine Funktion schnelle berechnung benötigt, um ein Umkippen zu verhindern.

Zum Abschluss des Projekts wurde ein Prototyp vom Roboter in einer Präsentation vorgestellt, bei der seine Funktionen in Aktion demonstriert wurden.

***Schlagwörter* — LEGO-Mindstorms, MATLAB, Präsentation, Prototyp, Roboter.**

I. Einleitung

DEr Balancier-Roboter erfordert eine präzise Steuerung und die Koordination mehrerer Motoren, um das Gleichgewicht aufrechtzuerhalten. Um stabil auf einer Kugel zu balancieren, muss er kontinuierlich seine Position und Neigung prüfen und entsprechende Korrekturen vornehmen. Dies erfordert eine schnelle Reaktionsfähigkeit und eine schwerige Regelungstechnik.

Gleichzeitig haben viele Menschen zu Hause nicht die Möglichkeit, komplexe Konzepte wie Gleichgewicht und Stabilität praktisch zu erforschen. Der Roboter ermöglicht es, diese Prinzipien anschaulich zu erleben, ohne teure Laborgeräte zu benötigen.

Falls eine Neigung erkannt wird, regelt er sich durch gezielte Motorbewegungen, um das Gleichgewicht wiederherzustellen. Dank dieses Mechanismus kann er auf einer Kugel balancieren, ohne umzukippen. Dies macht ihn nicht nur zu einem spannenden Demonstrationsobjekt für technische und physikalische Experimente, sondern auch zu einem unterhaltsamen Gerät für interaktive Anwendungen. Erster Versuch einen Roboter zu testen, ist auf der Abbildung 1 demonstriert.

II. Vorbetrachtung

Um einen Roboter zu bauen und programmieren, bräuchte man zuerst die ganze Idee zu verstehen. Dabei auch was einem für die Ziele in Verfügung steht, um alles ordentlich zusammenzusetzen, mit die Möglichkeiten des LEGO-Mindstorm-Sets.

Sie können über Lego Mindstorms EV3 mehr auf Wikipedia [2] entdecken.

DOI: 10.24352/UB.OVGU-2025-026

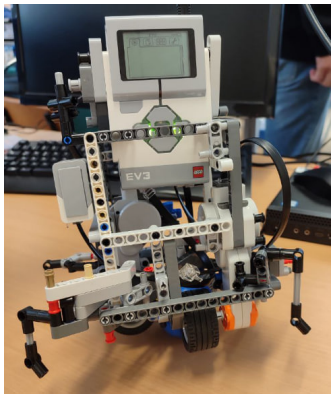


Abbildung 1. Komplett gebauter Roboter mit den Stützungstangen

A. Aufgabe des Roboters

Die Hauptfunktion des Roboters besteht natürlich darin, auf einem Ball balancieren, ohne übermäßige bewegung auszuführen. Falls dennoch solche unerwünliche Bewegung auftritt, dann bringen die Motoren den Roboter zurück ins Gleichgewicht, um diese Neigung zu vermeiden. Nachdem wir die wichtigste aufgabe erreicht haben, können wir andere aufgaben dazu hinzufügen. Beispielsweise, könnte der Roboter, mit Hilfe eines Tabletts Gegenstände auf dem „Kopf“ transportieren oder einfach einige Objekte für bestimmte Zeit halten.

B. Bestandteile

Die Teile, von dennen der Roboter gebastelt wurde, sind:
Legobauteile
2 Motoren
2 Gyrosensoren
2 Räder
EV3-Stein
Kabel inkl.(PC-USB verbindung)

Mehr info über die Teile [1]

C. Funktionsweise des Roboters

Der Balancier-Roboter funktioniert durch eine präzise Kombination aus Sensorik, Regelungstechnik und Antriebssystemen, um dauerhaft auf einer Kugel zu balancieren. Seine Hauptkomponenten sind:

1. Gyrosensoren zur Lageerkennung: Der Roboter verwendet zwei Gyrosensoren, um kontinuierlich die Neigung und Rotationsbewegung zu messen, was hilft ihm die Situation zu verstehen. Falls eine Abweichung vom Gleichgewichtszustand erkannt wird, sendet das System sofort Signale an die Motorsteuerung, die dann ihn zürückziehen sollen.

2. Regelungssystem: Der wichtigste Teil von Befehlsgebung ist ein PID-Controller (Proportional-Integral-Diverative-Regler) verarbeitet die Sensordaten und berechnet in Echtzeit die notwendigen Korrekturen, um das Gleichgewicht wiederherzustellen. Durch schnelles Nachjustieren der Motoren kann der Roboter auch auf kleinen Neigungen reagieren und stabil bleiben.

3. Motoren und Antriebsmechanismus: Der Roboter ist mit mehreren Motoren ausgestattet, die unabhängig voneinander gesteuert werden. Sie bewegen sich in verschiedene Richtungen, um das Gleichgewicht zu halten, indem sie kleine Anpassungen auf der Kugel vornehmen. Je nach Konstruktion können dies omnidirektionale Räder oder spezielle Rollen sein, die eine präzise Bewegung ermöglichen.

III. Technische Umsetzung

Der Roboter ist ein Prototyp, weil er noch nicht fertiggestellt wurde, und kann nicht seine wichtigste Funktion richtig umsetzen. Es gab Herausforderungen auf unserem Weg, sowohl mit der Konstruktion als auch mit der Programmierung in MATLAB Software.

A. Konstruktion des Roboters

Der Roboter wurde solche Weise gebastelt, dass er richtige Gewichtsverteilung hat. Seine Form errinert an ein Quadrat bzw. länglicher kubischer Körper, was hilft bei der Verteilung und somit kann man die bauteile, wie Motoren und Sensoren in bestimmte Positionen aufsetzen. Die Räder befestigten sich auch in specifischen Positionen, damit wenn der Roboter in verschiedenen Richtingen neigt, kann ein einziger Motor je vom bestimmten Winkel sich drehen und gekippte Seite zurück in die Anfangsposition zu bringen. Abbildung 2 zeigt besseren Anblick der Motoren.

B. Motoren und Gyrosensoren

1. Motoren. Sie sorgen für gezielte Bewegungen, um das Gleichgewicht auf dem Ball zu halten. In diesem Fall werden Servomotoren verwendet. Großer EV3-Servomotor ist ein leistungsstarker Motor mit eingebautem Rotationssensor, der eine Genauigkeit von einem Grad aufweist.

1. Gyrosensoren. Er misst die Drehgeschwindigkeit und Neigung des Roboters. Wenn der Roboter sich zur Seite neigt, erkennt der Gyrosensor diese Bewegung und sendet ein Signal an den EV3-Stein Controller. Dieser berechnet die nötige Korrektur und gibt den Motoren Anweisungen, um das Gleichgewicht wiederherzustellen.

Die Kombination aus Gyrosensoren und Motorsteuerung ermöglicht es dem Roboter, schnell auf Veränderungen zu reagieren und stabil auf dem Ball zu balancieren.

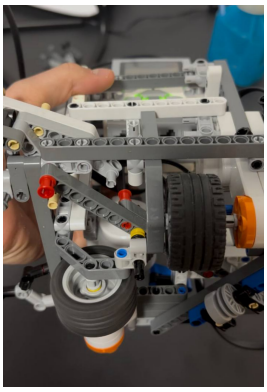


Abbildung 2. Konstruktion (Anblick von unten)

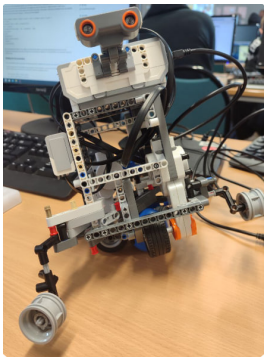


Abbildung 3. Problem mit dem Schwerpunkt

C. Probleme und Herausforderungen

Unsere größte Herausforderung war in MATLAB programmieren. Natürlich war dies auch die Hauptidee des Projekts - Code schreiben zu können und konzept des Programms zu verstehen. Danach überlegten wir uns, wie man den Roboter in die Balance bringen könnte. Dabei half uns der PID-Controller, jedoch mussten wir auch die richtigen k_{i},k_{p} und k_{d} Werte zu finden. Ein weiteres Problem gab es mit der Konstruktion, insbesondere mit dem EV3-Stein, da er zu schwer war und dadurch der Schwerpunkt gestört wurde (Abbildung 3). Um dies Problem zu vermeiden, schoben wir ihn so weit nach unten wie möglich - und das gelang uns. Das größte und problematischste Herausforderung war jedoch, dass die Gyrosensoren nicht richtig funktionieren. Nähmlich die Werte steigen bei jedem Testlauf an. Deshalb konnte man keine verlässlichen Anfangswerte bestimmen. Beispielsweise, wenn Anfangswert 0° und 1° ist, dann zeigt der Sensor nach mehreren Tests bereits 2° und -4° an. Das lässt sich auf dargestellten Diagramm auch gut anzeigen (Abbildung 4).

IV. Programmierung in MATLAB

Der Code soll so funktionieren, dass wenn der Roboter sich in eine beliebige richtung kippt, dann werden ausgemessenen Werte der beiden Gyrosensoren mit Hilfe eines Befehls ausgelesen und die Motoren in Bewegung setzen. Welcher Motor seiner Drehung startet, hängt vom Neigungswinkel ab. Beispielsweise, wenn der Roboter nach vorne neigt, sodass der Wert von des ersten Gyrosensors positiv und der des zweiten

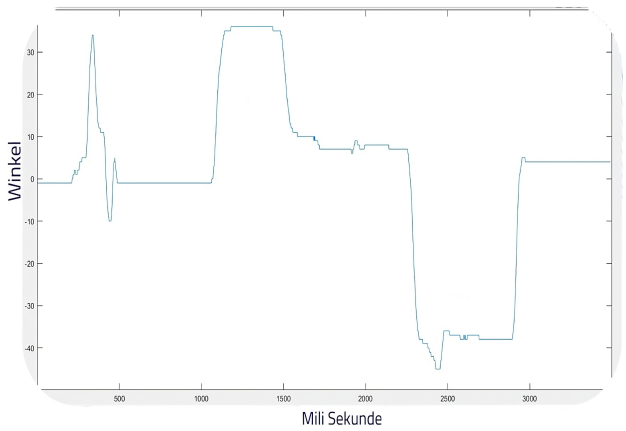


Abbildung 4. Diagramm der wechselnden Werte.

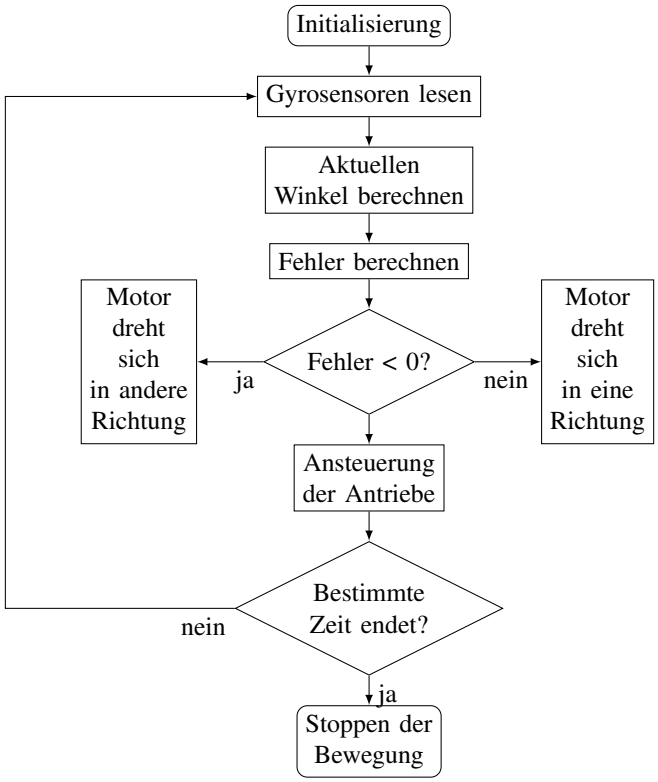


Abbildung 5. Verfolgungsalgorithmus des Programms für Balancier-Roboter

negativ ist, dann dreht sich nur Frontmotor, da in diese Situation kein Backmotor benötigt wird. Genauer kann man auf der Abbildung 5 sehen. Der Kern des Programms ist der PID-Controller. Er hat dafür auch 3 Werte.

1) Der Proportional-Anteil k_p: Reagiert direkt auf die aktuelle Abweichung vom Sollwert (z. B. gewünschter Neigungswinkel ist gleich 0°). Je größer Kp, desto stärker reagiert das System auf eine Abweichung. Ist Kp jedoch zu hoch, kann es zu Schwingungen oder Instabilität kommen.

2) Der Integral-Anteil k_i: Summiert kleine Abweichungen über die Zeit auf. Korrigiert systematische Fehler (Offset), die der Proportional-Anteil allein nicht ausgleichen kann. Ein zu hoher Ki-Wert kann zu Überschwingungen oder träghem Verhalten führen.

3) Der Differential-Anteil k_d: Betrachtet, wie schnell sich der Fehler ändert. Dämpft abrupte Bewegungen und hilft, frühzeitig gegenzusteuern. Ist Kd zu hoch, kann das System nervös oder ruckartig reagieren.

Der PID-Regler vergleicht ständig die tatsächliche Neigung mit der idealerweise 0°-Neigung. Basierend auf k_p, k_i und k_d berechnet er das Korrektursignal für die Motoren. So kann der Roboter selbst bei leichten Stößen oder unebenen Untergründen schnell reagieren und im Gleichgewicht bleiben.

V. Ergebnisdiskussion

Nachdem alles erledigt war, wurde der Roboter getestet. Es war auch eindeutig klar, das der Code funktioniert und die Motoren drehen sich relativ zum Winkelswert. Aber natürlich konnte das nicht lang gut läufen, denn es gibt das Problem mit Winkel und zwar mit einer Fehlerberechnung. Eine Lösung für dieses Problem ist noch nicht gefunden, aber uns ist eine Idee eingefallen. Man kann den Code direkt auf dem EV3-Stein übertragen und dann noch einige Teste machen. Es könnte natürlich sein, dass MATLAB kann nicht so schnell rechnen oder die Gyrosensoren übertragen ihre Werte nicht so schnell wie es bedarf.

VI. Zusammenfassung und Fazit

Der Roboter ist ein beeindruckendes Beispiel dafür, wie Technik und Mathematik zusammenwirken, um autonome Systeme stabil und reaktionsschnell zu gestalten. Man bekam neue Erfahrung mit dem MATLAB-Software, was auch in der Zukunft nützlich und wertvoll für weitere Projekte sein werden. Ein sehr spannendes Projekt, das einem beibring, wie man Probleme löst und gleichzeitig hilft, moderne Kenntnissen in der Roboterprogramierung zu erwerben. Diese Erkenntnisse sind nicht nur für den Bereich der Robotik relevant, sondern bieten auch einen praxisnahen Einblick in moderne Steuerungstechniken. Zum Beispiel, Mobile Transportmittel: Ähnliche Prinzipien wie bei selbstbalancierenden Segways werden genutzt, um kompakte, wendige Fahrzeuge zu entwickeln. A

Anhang

Nur kurze Anhang, wie das Programm PID berechnet:

```
error1 = desiredAngle1 - currentAngle1;
error2 = desiredAngle2 - currentAngle2;
integral1 = integral1 + error1 * elTime;
integral2 = integral2 + error2 * elTime;
der1 = (error1 - prevError1) / elTime;
der2 = (error2 - prevError2) / elTime;
```

Literaturverzeichnis

[1] . *EV3-Mindstorm Roboterteile*. 2025. URL: https://de.scribd.com/document/422734718/RobotParts-grade10.

[2] Wikipedia contributors. *Lego Mindstorms EV3 — Wikipedia, The Free Encyclopedia*. März 2025. URL: https://en.wikipedia.org/wiki/Lego_Mindstorms_EV3.

Selbstbalancierender Roboter auf einem Ball

Kseniia Shustikova, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Rahmen des LEGO-Mindstorms-Seminars an der Otto-von-Guericke-Universität Magdeburg wurde ein selbstbalancierender Roboter entwickelt, der auf einer Kugel statt auf Beinen oder Rädern stabilisiert wird.**

Zur Erfassung der Neigung in X- und Y-Richtung werden zwei Gyrosensoren verwendet. Ein Algorithmus verarbeitet diese Sensordaten und steuert zwei Motoren so, dass der Roboter aktiv Korrekturen vornimmt, um das Gleichgewicht auf der Kugel zu halten. Die Programmierung erfolgte mit MATLAB.

Am Ende des Projektseminars wurde der Roboter den Teilnehmern vorgestellt.

***Schlagwörter*—LEGO-Mindstorm, MATLAB, OVGU, Projektseminar, Roboter.**

I. Einleitung

DER selbstbalancierende Roboter könnte dank seiner Fähigkeit, das Gleichgewicht auf einem Ball zu halten, ein großartiger persönlicher Assistent sein. Da er sehr flexibel und stabil ist, kann er kleine oder große Gegenstände tragen (je nach Größe des Roboters) und bei einigen Aufgaben helfen. Aus denselben Gründen könnte er auch in Krankenhäusern eingesetzt werden, und wenn ein paar zusätzliche Sensoren und Kameras hinzufügen wurden, könnte er seinen Platz im Sicherheitsbereich finden.

Der Roboter hat 2 Gyrosensoren, die den Winkel an der X- und Y-Achse messen. Außerdem hat er 2 große Motoren, die zwei Räder steuern. Mit dem richtigen Algorithmus werden die, von den Gyrosensoren gelesenen, Daten mit mehreren anderen Daten multipliziert und dann als Geschwindigkeit an die Motoren gesendet. Durch das Beschleunigen und Verlangsamen der Motoren in die eine oder andere Richtung muss sich der Roboter auf der Kugel aufrecht halten und seine Stabilität bewahren [1].

In diesem Artikel wurde daher beschrieben werden, wie dies erreicht werden kann, welche Herausforderungen auf dem Weg dorthin auftreten können und welche Lösungsmöglichkeiten es gibt.

II. Vorbetrachtungen

A. Die Idee und Prinzip

Die Idee, einen Roboter zu bauen, der Menschen helfen kann, gibt es schon seit dem ersten Tag des Projektseminars. Die Fähigkeit, sich frei zu bewegen und stabil zu sein, egal, welche Hindernisse es gibt, dürfte also das erste und wesentliche Merkmal gewesen sein. Um diese freie Bewegung zu erreichen, sollte der Roboter keine Räder oder Beine haben, da er sonst nicht so leichtgängig ist, wie er sein sollte.

Das Funktionsprinzip basiert des Roboters auf einem einziegen Ball, die als bewegliche Basis dient. Der Roboter balanciert auf einem Ball und nutzt es als Antriebselement. Eingebaute Gyrosensoren messen die Neigungs- und Drehwinkel des Roboters in Echtzeit, so dass die aktuelle Position des Roboters immer bekannt ist. Diese Sensordaten werden in einen Steuerungsalgorithmus eingespeist, der mithilfe eines PID-Reglers kontinuierlich berechnet, wie die Motoren arbeiten müssen, um das Gleichgewicht zu halten.

DOI: 10.24352/UB.OVGU-2025-027

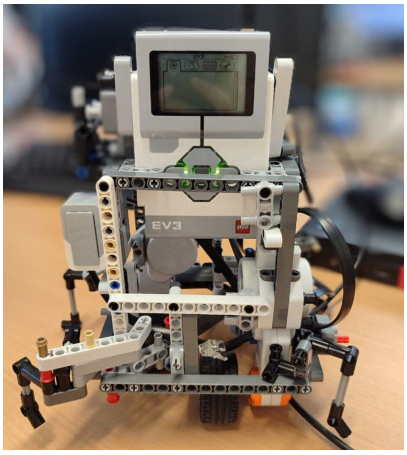


Abbildung 1. Fertige Konstruktion des Roboters

Die Motoren sind in der Lage, die Leistung dynamisch in Abhängigkeit vom Winkel und dessen Vorzeichen anzupassen. Dies ermöglicht es dem Roboter, stabil zu bleiben und sich in jede gewünschte Richtung zu bewegen.

B. Roboterteile

Die Teile, die zum Bau des Roboters verwendet wurden (siehe Abbildung 1):

1) Gyrosensoren 2×
2) Große LEGO-Motoren 2×
3) EV3-Brick
4) Batterie
5) Kabel
6) Verschiedene LEGO-Teile

C. Gyrosensoren

Die Gyrosensoren sind für den selbstbalancierenden Roboter auf einem Ball unerlässlich, da sie die Winkel und Neigungswinkel pro Sekunde auslesen und so präzise Informationen über den aktuellen Neigungszustand liefern. Dieser besondere Roboter ist mit zwei Gyrosensoren ausgestattet, erster befindet sich auf der linken Seite und misst kontinuierlich die Winkelveränderungen entlang der Y-Achse, während der andere Sensor auf der

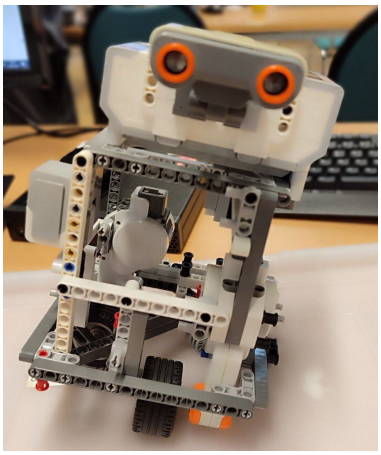


Abbildung 2. Der erste Entwurf

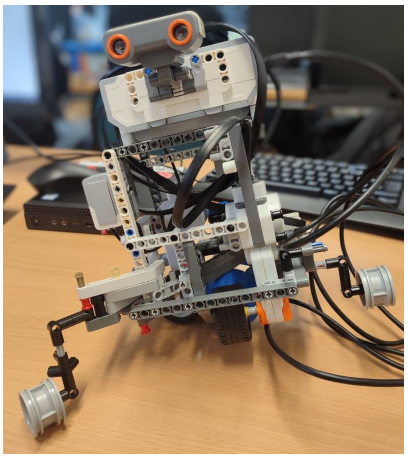


Abbildung 3. Konstruktion mit zusätzlichen Rädern

Rückseite die Winkelveränderungen entlang der X-Achse erfasst.

Diese Sensoren arbeiten mit hoher Aktualisierungsrate und Genauigkeit, was entscheidend ist, damit der Roboter schnell auf Veränderungen reagieren kann. Ihre Hauptaufgabe besteht also darin, die relevanten Daten in Echtzeit zu erfassen, die anschließend in einem Steuerungsalgorithmus – in diesem Fall PID-Algorithmus – korrekt berechnet und an die Motoren weitergeleitet werden. Die Motoren nutzen diese Informationen, um kontinuierlich kleine Korrekturen vorzunehmen, wodurch der Roboter immer wieder in die aufrechte Position zurückgeführt wird und somit stabil bleibt.

III. Hauptteil

A. Aufbau

Der Bau des Roboters wurde durch ein Handbuch mit detaillierten Anweisungen erleichtert, jedoch gab es noch einige Probleme, die eine Anpassung der Konstruktion erforderlich machten [2].

Das Hauptproblem bei der Entwicklung des robotergestützten Auswuchtsystems war die Herausforderung, das gewünschte Gewicht zu erreichen. Der EV3-Stein erwies sich als zu schwer. Bei der anfänglichen Konstruktion wurde er oben auf dem Roboter positioniert, was dazu führte, dass er bei den leichtesten Bewegungen herunterfiel (siehe Abbildung 2). Trotzdem wurde zunächst versucht, zusätzliche Räder anzubringen, um eine bessere Stabilität zu erreichen, ohne dabei auf die Verlagerung des Schwerpunkts zu achten (siehe Abbildung 3).

Nach einer Überprüfung der Konstruktion des Roboters wurde beschlossen, den Massenschwerpunkt (den EV3-Brick) so niedrig wie möglich zu platzieren, um die Stabilität zu erhalten (siehe Abbildung 1). Außerdem die zusätzlichen Räder wurden entfernt und stattdessen nur kleine Beine hinzugefügt, die dem Roboter bei Tests Halt geben.

B. Sensorik und Regelung

Wie bereits erwähnt, werden die Gyroskope verwendet, um die Winkel entlang der X- und Y-Achse zu überwachen. Diese Daten werden dann verarbeitet und an die Motoren

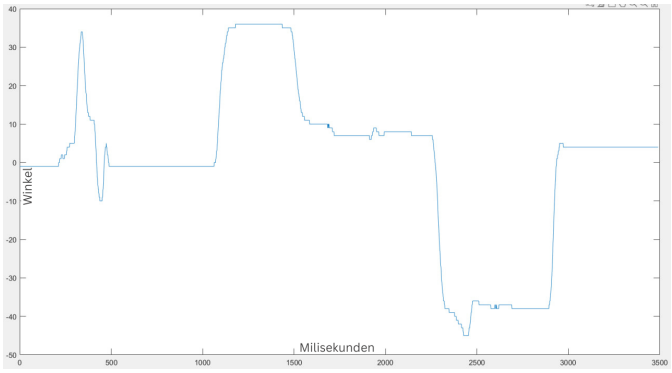


Abbildung 4. Grafik, die den Gyrosensor-Fehler anzeigt

gesendet. Der PID-Algorithmus ist das geeignete Werkzeug zur Verarbeitung dieser Daten [3].

Der PID-Algorithmus verwendet die Daten von Gyrosensoren, um das Gleichgewicht zu halten, indem er den aktuellen Winkel mit der gewünschten aufrechten Position vergleicht. Der proportionale Teil befasst sich mit dem unmittelbaren Fehler, der integrale Teil berücksichtigt die akkumulierten Fehler der Vergangenheit und der derivative Teil antizipiert plötzliche Änderungen. Die Summe dieser Terme bildet das Steuersignal, das den Ball in die entgegengesetzte Richtung jeder Neigung bewegt, wodurch eine kontinuierliche Rückkopplungsschleife entsteht, um den Roboter aufrecht zu halten.

Die folgende Formel veranschaulicht den Prozess, bei dem die PID-Werte zur Berechnung des Endergebnisses verwendet werden, das in diesem Fall die Leistung der Motoren ist.

$$\text{MotorPower} = K_p \cdot e + K_i \cdot \int e\,dt + K_d \cdot \frac{de}{dt} \qquad (1)$$

Allerdings traten bei jedem Test Fehler auf. Insbesondere die Gyroskopsensoren lieferten ungeeignete Daten. Wie in der Grafik dargestellt, ergibt die Ausgangsposition des Roboters (aufrecht) einen Winkel von 0. Dies ist die richtige Ausrichtung. Nach der Neupositionierung sollte der Winkel wieder auf Null zurückgehen. Dies ist jedoch nicht der Fall; stattdessen erhöht er (siehe Abbildung 4).

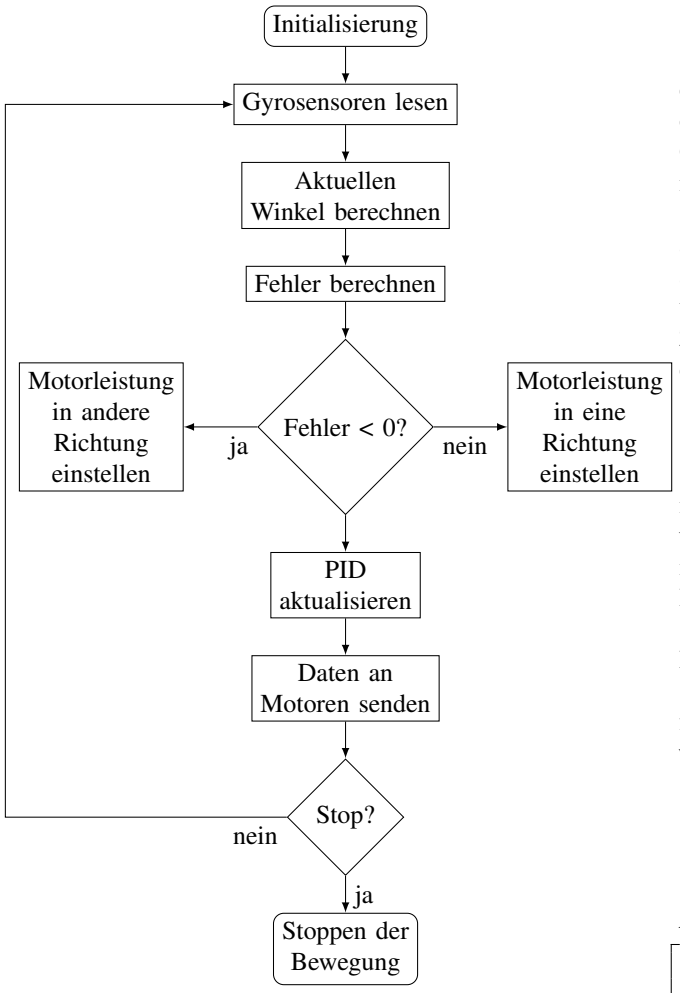


Abbildung 5. Flussdiagramm, das den Code darstellt

C. Funktionsweise und Stabilisierung

Zuerst initialisiert der Roboter alle notwendigen Komponenten, darunter seinen EV3-Brick, die Gyrosensoren und die Motoren. Dann werden die Daten der Gyrosensoren ausgelesen, die Informationen über die aktuelle Neigung liefern. Die Differenz zwischen dem gewünschten aufrechten Winkel und dem empfangenen Winkel wird als Fehler bezeichnet. Nach der Bestimmung des Vorzeichens des Fehlers aktualisiert das System den PID-Regler mit diesen neuen Informationen. Der PID-Algorithmus berechnet die erforderliche Korrektur auf der Grundlage von Proportional-, Integral- und Derivativtermen und erzeugt eine Steuerausgabe, die dann an die Motoren gesendet wird. Dieser aktualisierte Befehl veranlasst die Motoren, ihre Leistung so zu ändern, dass der Roboter wieder in eine aufrechte Position gebracht wird, und dieser

Prozess wird kontinuierlich fortgesetzt, solange der Roboter aktiv ist. In jeder Phase liest das System Sensordaten aus und berechnet den Winkel neu. Wenn eine Stopp-Bedingung erreicht wird - der Roboter hält an und es werden keine Daten mehr ausgelesen - kann kein Gleichgewicht mehr erreicht werden.

IV. Ergebnisdiskussion

Am Ende des Baus und der Programmierung wurde also dieses Ergebnis erhalten: die Räder in der Lage, ihre Geschwindigkeit und Drehrichtung auf der Grundlage der Daten von Gyrosensoren zu ändern, und genau - in einem positiven oder negativen Winkel, auf der x- oder y-Achse.

Der Roboter kann sich immer noch nicht auf einem Ball aufrecht halten, obwohl er durchaus versucht, seine Position zu ändern, um näher an die Oberseite des Balls heranzukommen und auf ihm zu bleiben. Dies liegt sowohl an der begrenzten Zeit für die Entwicklung als auch an ungenauen Sensordaten der Gyrosensoren - ein Problem, das bislang ungelöst blieb.

V. Zusammenfassung und Fazit

In zwei Wochen wurde viel Arbeit geleistet, auch wenn noch nicht alle Probleme gelöst sind. Durch diese Praxis konnte ich viele neue Kenntnisse und Fähigkeiten erwerben, wie die Arbeit mit Matlab und verschiedenen Sensoren, Motoren, Teamarbeit, Problemlösung und kritisches Denken. Auch wenn nicht alles so reibungslos verlief wie erwartet, sind die gewonnenen Erfahrungen von unschätzbarem Wert.

Und mit mehr Zeit wäre es möglich gewesen, die Ursache der fehlerhaften Sensordaten zu identifizieren und zu optimieren, wie häufig die Daten aktualisiert werden müssen, um eine stabile Balance des Roboters zu gewährleisten.

Anhang

Ein Beispiel für die Anordnung der Motorleistung in Abhängigkeit von einem positiven oder negativen Winkel.

```
if rate1 > rate2 && rate1 > 0 && rate2 < 0
    motorPower = -(Kp * error + Ki * integral
        + Kd * derivative);
    motorA.power = max(min(motorPower, 100),
        -100);
    motorB.power = 0;
end
if rate1 < rate2 && rate1 < 0 && rate2 > 0
    motorPower = (Kp * error + Ki * integral +
         Kd * derivative);
    motorB.power = max(min(motorPower, 100),
        -100);
    motorA.power = 1;
end
```

Literaturverzeichnis

[1] Wikipedia: *Ballbot*. : Wikipedia, Februar 2025. https://en.wikipedia.org/wiki/Ballbot

[2] Yamamoto, Yorihisa: *NXT Ballbot (Self-Balancing Robot On A Ball) Controller Design*. Version 1.0.0.0. : MATLAB, Februar 2025. https://ch.mathworks.com/matlabcentral/fileexchange/23931-nxt-ballbot-self-balancing-robot-on-a-ball-controller-design

[3] Wikipedia: *Proportional–integral–derivative controller*, Februar 2025. https://en.wikipedia.org/wiki/Proportional%E2%80%93integral%E2%80%93derivative_controller

Zeichenroboter

Mohamed Ammar Abdalla Hassan, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In diesem Artikel werden die Inspiration für die Projektidee, die Bausteine des Projekts, der mechanische Aufbau und die Programmierung des Roboters, die Herausforderungen, die sich während des Projekts ergaben, und die Mittel, die zur Lösung der einzelnen Probleme eingesetzt wurden, sowie die Ergebnisse und das Potenzial von Zeichenrobotern in der Zukunft beschrieben.**

***Schlagwörter*—Schultermotor, Ellbogenmotor, Stiftmotor, glückliches Emoji, trauriges Emoji .**

I. Einleitung

UNSERE heutige Welt wächst rasant, und täglich werden verschiedene Roboter gebaut, um unterschiedliche Aufgaben zu erfüllen. Das LEGO-Seminar war eine Gelegenheit, in diese Welt einzutauchen und sich die Aspekte dieser Welt vorzustellen. Das Seminar bot auch eine großartige Chance zu lernen, wie man in Matlab programmiert und wie man die mit NXT-Bausteinen gebauten LEGO-Roboter steuert.

Der Roboter, der im LEGO-Praktikum 2025 gebaut wurde, heißt Zeichenroboter. Wie der Name schon sagt, handelt es sich um einen Roboterarm, der dafür zuständig ist, Formen wie Kreise usw. zu zeichnen. Der Roboter wurde hauptsächlich für Unterhaltungszwecke gebaut, kann aber auch für andere Zwecke verwendet werden. Der Roboter kann z.B. ein Spielgerät werden, mit dem sich Kinder die Zeit vertreiben können usw.

II. Vorbetrachtungen

In diesem Abschnitt wird erklärt, woher die Idee für das Projekt stammt und welche Teile und Bausteine für den Zusammenbau der Roboterkunst benötigt werden.

A. Inspiration

Die Idee für den Roboter wurde durch Videos und Bilder aus Internet inspiriert. Die erste Idee war, den Roboterarm in der Lage zu machen, Alphabete zu schreiben, aber aufgrund einiger Schwierigkeiten, die später in diesem Artikel erklärt werden, war es klug, nach etwas anderem zu suchen, das einfacher zu zeichnen war. So wurde beschlossen, den Roboterarm stattdessen Emojis zeichnen zu lassen, insbesondere glückliche und traurige Emojis. Abbildung 1 war eines der Bilder, die die Idee zu diesem Projekt inspirierten.

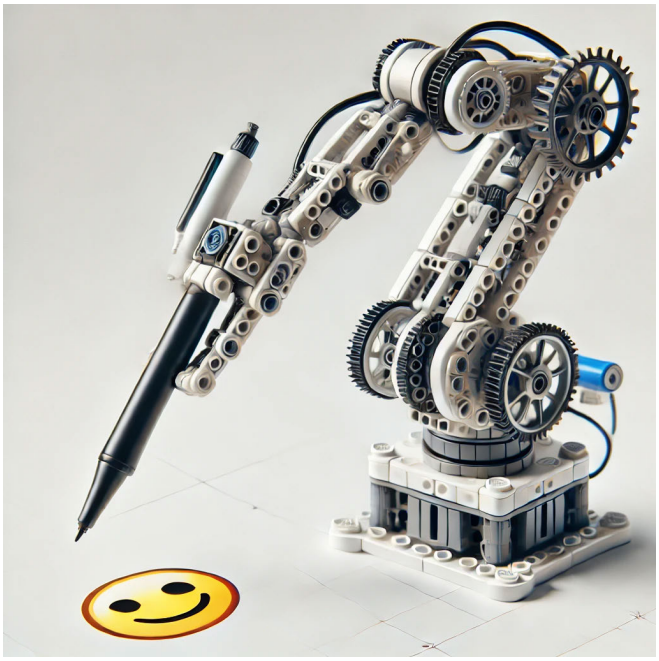


Abbildung 1. Inspiration [1]

B. Bausteine

Für dieses Projekt wurden NXT-Motoren, LEGO-Blöcke und Zahnräder verwendet. Der Roboterarm steht auf einer großen Plattform, um Stabilität zu gewährleisten. Die Bausteine und Komponenten des Projekts sind in Abbildung 2 dargestellt.

Abbildung 2. Verwendete Komponenten [2]–[5]

DOI: 10.24352/UB.OVGU-2025-029

III. Realisierung

In diesem Abschnitt werden der Aufbau des Roboters und die zur Steuerung des Roboters verwendeten Funktionen erläutert.

A. Konstruktion

Der Roboter besteht aus drei großen Motoren und zwei Hauptzahnrädern. Der erste Motor ist der „Schultermotor“, der es ermöglicht, den Arm vor und zurück zu bewegen, der zweite ist der „Stiftmotor“, der es ermöglicht, den Stift zu heben und zu senken, während der letzte wichtige Motor der „Ellbogenmotor“ heißt, an den der Stift direkt angeschlossen ist und der dafür verantwortlich ist, den Stift zu bewegen, um auf dem Papier zu zeichnen. Der Schultermotor ist mit der Plattform verbunden, der Stiftmotor ist mit dem großen Zahnrad verbunden, das wiederum von einem kleinen Zahnrad gedreht wird, das mit dem Schultermotor verbunden ist, und der Ellbogenmotor ist direkt mit dem Stiftmotor verbunden. Der Zusammenbau der Komponenten und die Kennzeichnung der einzelnen Teile sind in Abbildung 3 dargestellt.

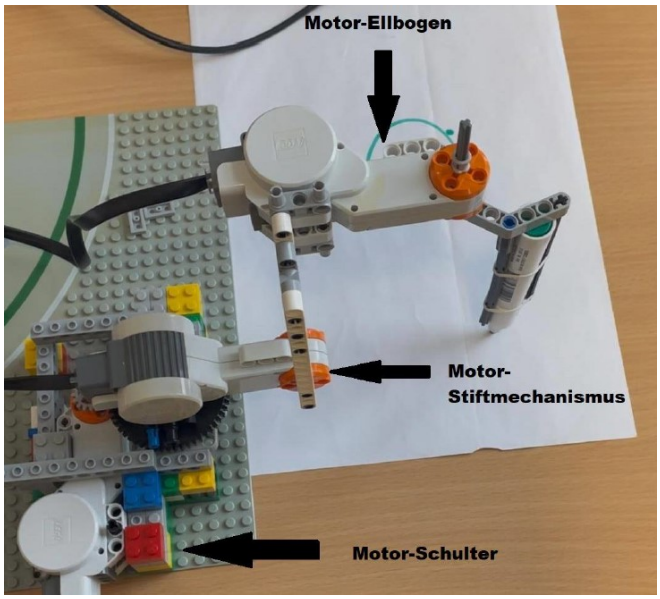


Abbildung 3. Konstruktion

B. Programmierung

Um ein fröhliches und ein trauriges Emoji zu zeichnen, wurden zwei separate Funktionen benötigt. Die erste Funktion heißt „draw_happy()-Funktion“ für fröhlich und die zweite Funktion heißt „draw_sad()-Funktion“ für traurig. Abbildung 4 demonstriert den Ablauf des Programms und die Reihenfolge, in der die Teile der Emoji gemalt wurden.

Beide Funktionen beginnen mit dem Zeichnen des Gesichts, indem ein Vollkreis mit der 360-Grad-Drehung des Motors gezeichnet wird. Dann wird der Stift mit Hilfe des Stiftmotors angehoben und mit Hilfe des Schultermotors zu dem Punkt verschoben, an dem der Mund gezeichnet wird. Bei der draw_happy()-Funktion wird ein Lächeln gezeichnet, während bei der draw_sad()-Funktion ein trauriger Mund gezeichnet

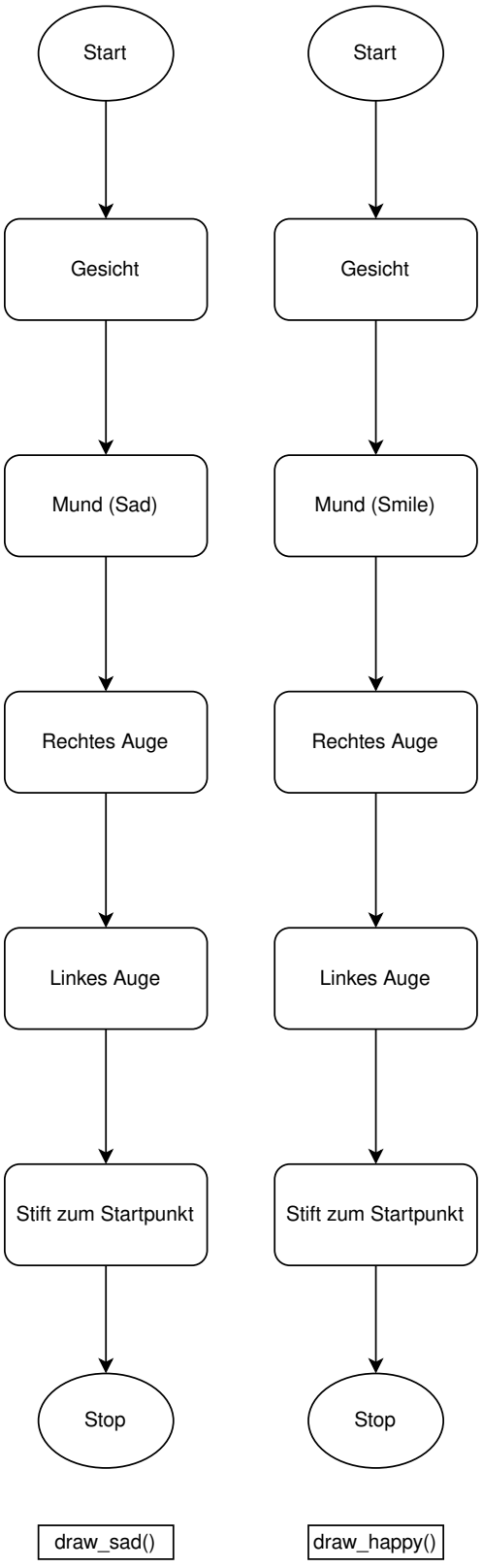


Abbildung 4. Programablauf und Funktionsschritte

wird. Nach dem Zeichnen des Mundes wird der Stift wieder angehoben. Der nächste Schritt ist das Zeichnen der Augen. Der

Stift wird auf die Position des rechten Auges verschoben. Dann wird der Stift aufgesetzt, um das rechte Auge zu zeichnen. Der gleiche Vorgang wird für das linke Auge durchgeführt. Zum Schluss wird der Stift wieder in die Anfangsposition gebracht und die Funktion ist nun beendet.

IV. Ergebnisdiskussion

In diesem Abschnitt werden die Herausforderungen und ihre Lösungen vorgestellt und wie die endgültige Version des Roboters erreicht wurde.

A. Herausforderungen

Die folgenden sind einige der Schwierigkeiten, die bei der Entwicklung des Roboters auftraten:

1) **Begrenzte Motorbewegungen:**
   Die ursprüngliche Idee war, wie bereits erwähnt, einen Roboter zu entwickeln, der Alphabete schreiben kann, aber aufgrund der begrenzten Bewegung der Motoren, die nur kreisförmige Bewegungen zuließen, war es eine wirklich schwierige Aufgabe, gerade Linien zu zeichnen, um Alphabete wie A, T und Z zu schreiben. Um eine gerade Linie zu zeichnen, musste das Konzept der inversen Kinematik verwendet werden. Die Umsetzung dieses Konzepts war aufgrund der komplexen mathematischen Gleichungen recht schwierig. Die Idee war also, etwas zu zeichnen, das einfacher zu programmieren war. So entstand die Idee, Emojis zu zeichnen, die nur kreisförmige Bewegungen der Motoren erfordern würden.

2) **Kabellänge:**
   Die zweite Idee war, um das System herum zu zeichnen, z. B. einen großen gewellten Kreis, und andere Bilder um das System herum zu zeichnen. Die Länge der Kabel, die die LEGO-Motoren mit dem NXT-Brick verbinden, war recht kurz. Dies schränkte die Bewegung des Stifts ein und verhinderte eine 360-Grad-Bewegung. Es war also notwendig, sich auf eine Seite und einen festen Punkt zu konzentrieren. Daher war das Zeichnen auf einem Blatt Papier, das neben dem Roboter platziert wird, die ideale Lösung.

3) **Aufbau einer stabilen Systemstruktur:**
   Obwohl die Struktur des Projekts relativ einfach aussieht, Mechanisch gesehen war es eine Herausforderung, die drei großen Motoren zusammenzufügen und sie irgendwie zusammenzuhalten, um das Gewicht des Stiftes zu tragen. Das Gewicht der Motoren zusätzlich zum Gewicht des Stifts war eine schwierige Aufgabe. Um dieses Problem zu lösen, mussten mehrere Prototypen gebaut und getestet werden, um die bestmögliche Struktur zu finden.

B. Roboter-Prototypen

Im Folgenden sind einige der Prototypen aufgeführt, die im Laufe des Projekts bis zur endgültigen Version gebaut wurden.

- *Allererster Prototyp:*
  Im ersten Prototyp (siehe Abbildung 5) wurden die Motoren zusammengebaut, um die Funktionalität des Roboters zu testen und die ideale Struktur zu finden.

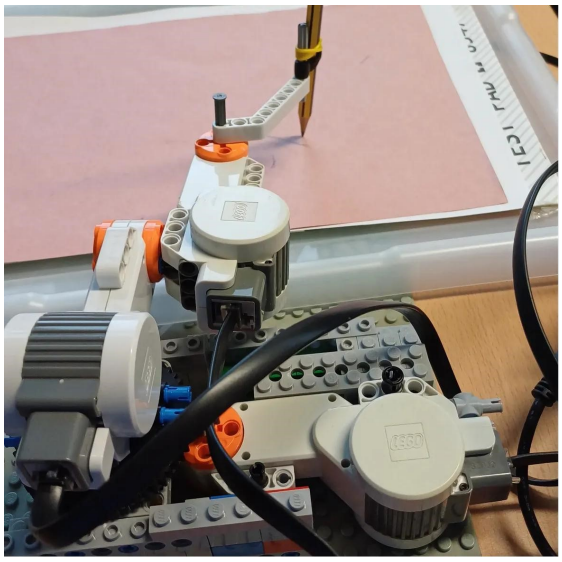


Abbildung 5. Erster Prototyp

- *Zweiter Prototyp:*
  Bei der zweiten Variante, die in Abbildung 6 dargestellt wurde, wurde die gesamte Plattform angehoben und auf eine höher gelegene Fläche gestellt.

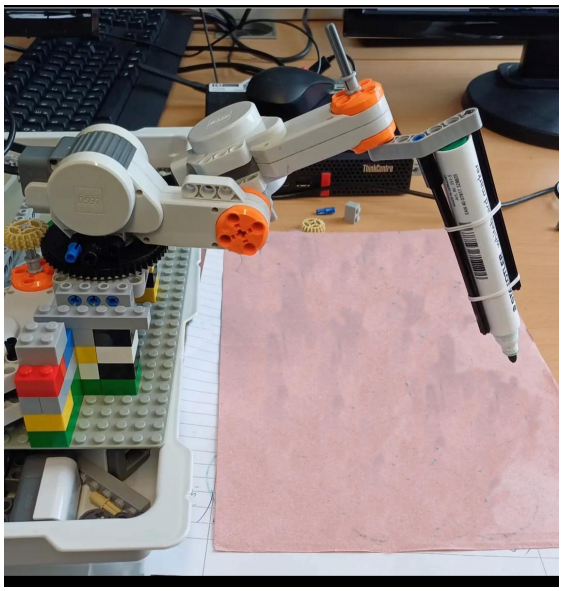


Abbildung 6. Zweiter Prototyp

- *Finale Struktur:*
  In der endgültigen Version, wie in Abbildung 7 dargestellt, wurde die Plattform nach hinten abgesenkt, aber nur der Ellbogenmotor wurde angehoben, wobei eine Verlängerung verwendet wurde, die den Ellbogenmotor mit dem Stiftmotor verbindet. Dies geschah, um eine präzise Auf-

und Abwärtsbewegung des Stifts zu ermöglichen, aber auch, um das System zu stützen und für mehr Stabilität zu sorgen.

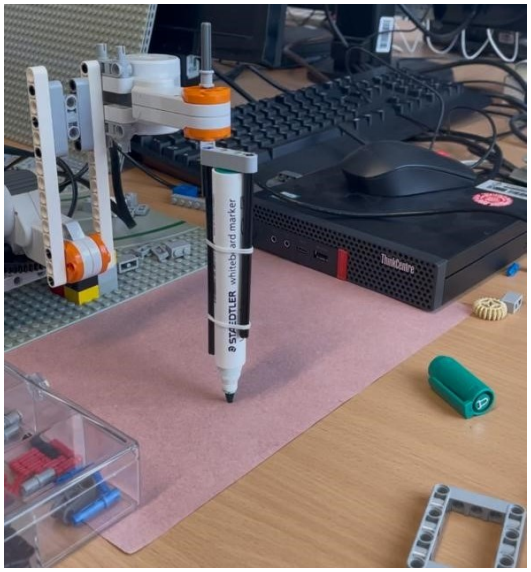


Abbildung 7. Endergebnis

V. Zusammenfassung und Fazit

Nach all den Umbauten und Programmierungen und nachdem fast alle Probleme gelöst waren, war das Endergebnis recht zufriedenstellend. Der Roboter konnte nicht nur traurige Emoji zeichnen, sondern auch fröhliche Emoji, um Freude auf die Gesichter der Menschen zu bringen. Natürlich ist der Zeichenroboter noch stark verbesserungsfähig. Der Roboter kann so in der Zukunft verbessert werden, dass er nicht nur Alphabete, sondern auch Wörter und sogar Absätze schreiben kann!

Literaturverzeichnis

[1] CHATGPT: *Das Inspirationsbild wurde mit AI und genau mit ChatGPT erstellt*. 2025

[2] BUY SITES: *NXT motor*. https://goodshksk.click/product_details/23980594.html. Version: Januar 2021

[3] AMAZON: *LEGO Block*. https://www.amazon.ae/TTEHGB-TOY-Building-Compatible-Creative/dp/B0BM4S64H9?th=1. Version: März 2025

[4] THE FAMILY BRICK: *Großzahnrad und Platform*. https://thefamilybrick.com/lego-mocs/motorizing-the-lego-christmas-tree/. Version: Oktober 2020

[5] TOYPRO: *Kleinzahnrad*. https://www.toypro.com/en/product/5832/gear-8-tooth-type-2/dark-bluish-gray. Version: März 2025

„Roboter-Manipulator“Automatische Gabelstapler mit LEGO-Mindstroms

Yehor Bykov, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

Zusammenfassung - Ein jährliches Projektseminar über Elektrotechnik und Informationstechnologie und die Auswirkungen auf andere Fachbereiche. In diesem Jahr handelt es sich bei dem Projekt um einen speziellen Mechanismus namens „Robotermanipulator“, einen automatischen Gabelstapler. Dadurch wird die Verletzungsgefahr verringert, während der Arbeitsaufwand reduziert und der Prozess automatisiert wird. Der Mechanismus wurde von LEGO entwickelt und hergestellt.

Mindstorms-Bausätze und der LEGO-NXT-Steuercomputer wurden verwendet.

Die Programmierung wurde mit MATLAB durchgeführt. In diesem Artikel wird der Aufbau und die Funktionsweise des Mechanismus vorgestellt. Außerdem werden Probleme, die während des Entwurfsprozesses auftraten, aufgezeigt und deren Lösungen diskutiert.

Schlüsselwörter: Automatisierung, Sensor, Roboter, Matlab, Code

I. IDEE

Die Fortschritte in der Robotik haben es uns ermöglicht, eine Vielzahl von Problemen zu lösen, sei es in der Industrie oder im Bereich der öffentlichen Sicherheit. Sicherheit. Das erste Bild zeigt die Unsicherheit nicht automatisierter Handlungen Dies veranschaulicht, wie Robotik und Automatisierung Aufgaben ohne menschliches Eingreifen reibungslos ausführen können. (Abbildung 1) Wir alle wissen, dass Innovationen wie automatisierte Maschinen zu einem notwendigen Sicherheitssystem in verschiedenen Arten von Robotern werden. Diese Technologie verspricht nicht nur mehr Effizienz und Sicherheit, sondern auch ein geringeres Risiko für Menschen in gefährlichen Umgebungen.Mit sorgfältiger Robotik und Sicherheit können wir die vollständige Automatisierung von Maschinen erreichen und potenzielle Risiken minimieren.

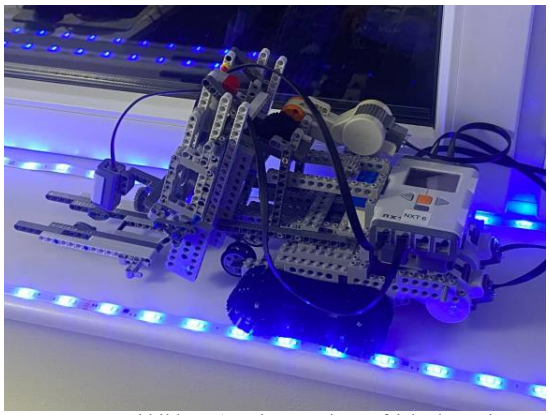


Abbildung 1: Seiten- und Draufsicht des Roboters

II. KONSTRUKTION

A.Struktur des automatischen „Roboter-Manipulator“-Systems Die Implementierung des Laders ist nicht sehr schwierig. Um jedoch die automatische Fähigkeit eines Mechanismus zu gewährleisten, die Ladung zu lokalisieren und zu bewegen, sind Motoren und Sensoren erforderlich, die bestimmte Arten und Spezifikationen erkennen und ausführen. Die Arten und Eigenschaften der erforderlichen Elemente sind nachstehend aufgeführt (Abbildung 2).

B. Motoren

Für den Aufbau des Roboters werden drei Motoren benötigt, zwei davon für die Fortbewegung. Für die Fortbewegung wurde ein Raupenfahrwerk gewählt, das für diese Zwecke am besten geeignet ist, da es Stabilität und gute Manövrierfähigkeit auf engstem Raum gewährleistet und auch die Manövrierfähigkeit ein wichtiger Punkt bei diesem Roboter ist. Der Einsatz eines reinen Hinterradantriebs mit zwei Motoren ermöglichte mehr Kraft und Kontrolle an schwer zugänglichen Stellen.

C. Dritter Motor für den Hubmechanismus

Ziel war es, einen reibungslosen Betrieb des Motors unabhängig vom Gewicht der Last, die von der Plattform gehoben wird, zu gewährleisten. Um einen stabilen Betrieb des Systems zu gewährleisten, entschied man sich für die Verwendung von kleinen Übersetzungsverhältnissen mit Hilfe von Zahnrädern, deren Drehung die Zahnstange auf der Plattform selbst bewegt. Dies ermöglicht eine präzise Ausführung der Hebevorgänge. Die Plattform selbst wurde so konstruiert, dass die Last sicher darauf befestigt werden konnte. Die Schalung wurde mit zwei Gabelzinken und speziellen Querträgern ausgeführt, um ein Herabfallen der Last zu verhindern.

D. Eigenschaften des Roboters

Aufgrund des hohen Gewichts der Maschine und der großen Höhe der Plattform wurde beschlossen, alle Teile, einschließlich des NXT-Geräts, auf der Rückseite des Rahmens zu platzieren, der dann als Gegengewicht dient und verhindert, dass der Roboter aufgrund des Gewichts der angehobenen Last nach vorne kippt. Um dieses Ergebnis zu erreichen, mussten die Plattform und der Rahmen komplett neu konstruiert werden.

DOI: 10.24352/UB.OVGU -2025 -030

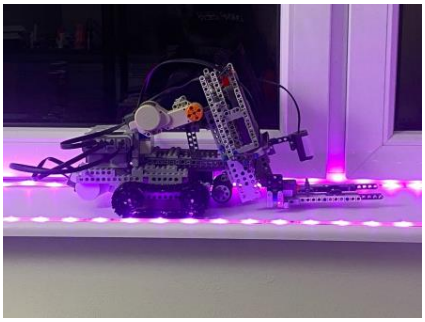


Abbildung 2: Seitenansicht des Roboters

E. Strukturelle Probleme und deren Lösungen

Ein Hauptproblem bei der Konstruktion war die Unzuverlässigkeit des Systems aufgrund der Verwendung von Lego-Würfeln. Sie verbogen sich ein wenig, was zu Spiel im System zwischen den Teilen führte. Insbesondere beim Mechanismus zum Anheben der Plattform rutschten die Zähne der Zahnräder durch und konnten die Last nicht anheben.

Die Lösung des Problems bestand darin, die Konstruktion zu verbessern und die Plattform um 25 bis 30 Grad nach hinten zu kippen, so dass die Last im Verhältnis zur Hubbewegung auf den Kontaktpunkt zwischen den Zahnrädern und der Zahnstange drückt.

III. Automatisierung

A.Automatisierungsmöglichkeiten für Lego-Roboter

Die Lego-NXT-Steuereinheit wird zur Programmierung und Automatisierung des Roboters verwendet. Zur Realisierung der Funktionalität eines Gabelstaplerroboters kommen Drucksensoren zum Einsatz, um das Vorhandensein einer Last auf der Plattform zu erkennen. Zusätzlich wird ein Farbsensor integriert, der die Sortierung der Paletten ermöglicht.

B.Tastsensor

Das Lego-NXT-System verfügt über einen Tastsensor, der an der Hebebühne befestigt ist. Berührungssensor Der Berührungssensor ist an der Vorderseite der Plattform angebracht, um Objekte auf der Plattform zu erkennen. Er diente als Stopptaste für die Lokomotiven und zeigte an, wenn sich ein Fremdkörper auf der Plattform befand.

C.Farbsensor

Eine weitere Aufgabe des Roboters besteht darin, die Last an einen definierten Ort zu transportieren. Zur Sortierung der Paletten wird ein Farberkennungssystem eingesetzt. Farbsensor Der Farbsensor befand sich oben an der Vorderseite der Plattform, um die Klasse des Objekts zu scannen und einen Algorithmus auszulösen, der das Objekt an die gewünschte Position bewegt (Beispiel: Scannen von QR-Codes).

D. Probleme und Lösungen

Das Hauptproblem bei der Installation der Sensoren war folgendes: Der Berührungssensor musste so positioniert werden, dass er zwar auf das Objekt reagierte, aber nicht mit ihm kollidierte, weshalb die Plattform verlängert und der Aktionsradius des Sensors vergrößert wurde. Das Problem des Lichtsensors war sein Aktionsradius (nicht mehr als 2 bis 3 Zentimeter), der Probleme bei der Positionierung verursachte, aber nach mehreren Messungen wurde der Sensor so positioniert, dass er die Farbe des Objekts abtasten konnte, ohne es zu berühren.

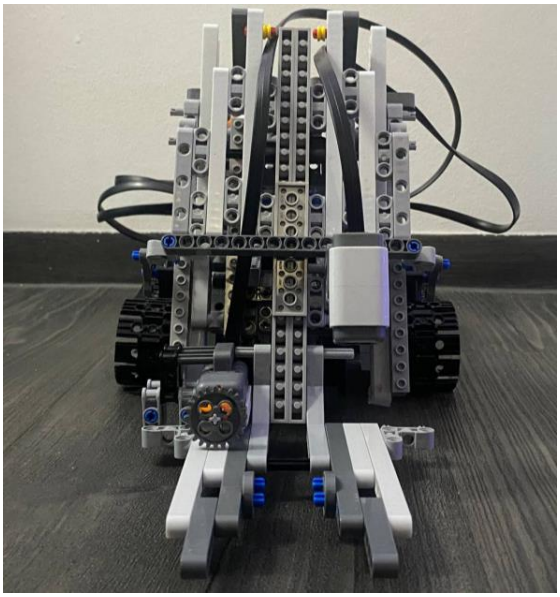


Abbildung 3: Vorderansicht des Roboters 1

IV. Roboter-Algorithmus

Zu Beginn demonstriert der Roboteralgorithmus alle seine Funktionen und Eigenschaften.

A .Algorithmus

Der Betrieb des Roboters erfolgt nach folgendem Algorithmus:

1. Start an der Ausgangsposition
2. Rückwärtsfahren
3. Bewegung zur Palette 1 (rot auf dem Foto)
4. Stoppen der Antriebsmotoren und Anheben der Palette um 20 Grad nach Drücken des Sensors
5. Farbpalettenscannen
6. Richtung „Lager“ drehen
7. Platzieren Sie das Tablett auf einem dafür vorgesehenen Regal (z. B. auf dem obersten Regal)
8. Rückwärtsbewegung
9. Fahrt zur Palette 2 (grün auf dem Foto)
10. Stoppen der Antriebsmotoren und Anheben der Palette um 20 Grad nach Drücken des Sensors
11. Farbscan der Palette
12. Drehung in Richtung „Lager“
13. Platzieren Sie das Tablett auf einem dafür vorgesehenen Regal (z. B. auf dem untersten Regal)
14. Abschluss des Algorithmus

A. Zusätzliche Gebäude

Um zu demonstrieren, wie der Roboter funktioniert

Auf der Lego-Plattform werden verschiedene zusätzliche Strukturen verwendet. Die wichtigsten sind natürlich die Paletten und das „Warehouse“. Auf dem Regal auf zwei Ebenen befinden sich farbige Schilder, die verdeutlichen, wo die Paletten hingestellt werden müssen. Die Paletten selbst wurden für unsere Art von Plattform entworfen und sind farblich gekennzeichnet. Außerdem sind sie mit farbigen Elementen versehen, so dass der Farbsensor den erforderlichen Algorithmus starten kann. Jede Palette steht an ihrem Platz und es gibt spezielle Plätze für ihre Position auf dem Regal (Abbildung 4). Außerdem haben wir beschlossen, als Ausgangspunkt einen improvisierten Parkplatz für die Gabeln der Plattform zu bauen.

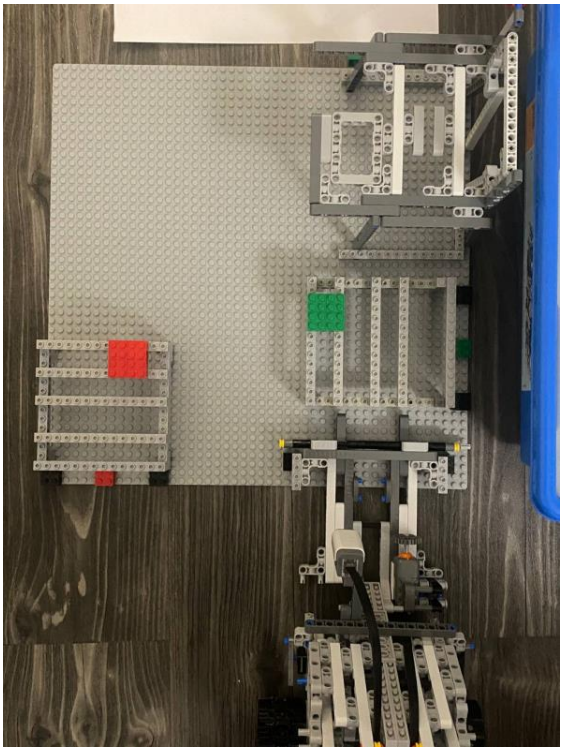


Abbildung 4: Ansicht des „Ladeterminals“ von oben

V. PROGRAMMIERUNG

A. Der NXT-Roboter wird mit MATLAB unter Verwendung der optionalen RWTH Mindstorms NXT Toolbox-Bibliothek programmiert. Diese Bibliothek enthält alle notwendigen Elemente, um den Roboter zu testen und den Algorithmus auszuführen. Diese Bibliothek stellt dem Programmierer alle Funktionen zur Verfügung, die es ihm ermöglichen, den gewünschten Algorithmus auf einfache Weise mit dem Lego NXT System zu realisieren und den Roboter effizient zu steuern.

B. Ein Beispiel für einen Teil des Programmiercodes des Roboters ist in (Abbildung 5) dargestellt:

Dies ist ein Beispiel für den Code, mit dem der Roboter so programmiert wird, dass er die angegebenen Befehle ausführen kann

```
state = 0;
lastButtonState = -1;
while true
   currentButtonState = GetSwitch(SENSOR_1);
     if currentButtonState == 1 && lastButtonState == 0
      if state == 0
         mMoveA.Power = MOVE_MOTOR_POWER;
         mMoveB.Power = MOVE_MOTOR_POWER;
         mMoveA.SendToNXT();
         mMoveB.SendToNXT();
         state = 1;
      elseif state == 1
         mMoveA.Stop('brake');
         mMoveB.Stop('brake');
         mLiftC.Power = LIFT_MOTOR_POWER;
         mLiftC.TachoLimit = DEGREES_PER_CM;
         mLiftC.SendToNXT();
         mLiftC.WaitFor();
         mMoveA.Power = ROTATION_POWER;
         mMoveB.Power = ROTATION_POWER;
         mMoveA.TachoLimit = 540;
 mMoveB.TachoLimit = 540;
```

Abbildung 5: Teil des Programmiercodes

C. Zusammenfassung:

Der Code, den ich im Beispiel geschrieben habe, zeigt die Elemtarfunktionen und Aktionen, die der Roboter ausführen kann

VI. LITERATURVERZEICHNIS

[1] Lego Mindstorm Wikipedia
https://en.wikipedia.org/wiki/Lego_Mindstorms\

[2] NXT Großer Motor
https://brickscout.com/de/products/detail/4057296793389/537%2087/electric-motor-nxt

[3] NXT Touch Sensor
https://www.steinpalast.eu/en/1-x-lego-brick-dark-bluishgray-electric-sensor-touch-nxt-8527-4296929-9843-53793

[4] RWTH - Mindstorms NXT Toolbox
https://de.mathworks.com/matlabcentral/fileexchange/18646-rwth-mindstorms-nxt-toolbox

[5] Roboter-Konstruktionen mit Lego Mindstorms NXT
https://www.generationrobots.com/de/content/60-roboter-konstruktionen-mit-mindstorms-nxt-v2

[6] Navigation von Wegpunkten mit einem Lego MINDSTORMS NXT Roboter
https://de.mathworks.com/matlabcentral/fileexchange/42835-navigating-waypoints-with-a-lego-mindstorms-nxt-robot

[7] NXT Farbsensor
https://botland.de/eingestellte-produkte/4697-legomindstormsnxt-ev3-farbsensor-lego-9694.html

[8] Projekt Instagram
https://www.instagram.com/2y_lego_projekt_seminar.g13/

Automatischer Gabelstapler

Yehor Popovych, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

Im Rahmen des Design-Workshops an der Otto-von-Guericke-Universität Magdeburg wurde ein automatisierter Gabelstapler entwickelt. Ziel des Projekts ist die Optimierung des innerbetrieblichen Transports durch eine autonome Lösung.

Zur Umsetzung kamen LEGO-Mindstorms-Bausätze und der NXT-Steuerungscomputer zum Einsatz. Dieser Artikel beschreibt den Aufbau, die Funktionen sowie die wichtigsten Herausforderungen und Lösungsansätze.

***Schlagwörter* – Berührungssensoren, Elektromotoren, Farbsensor, Gabelstapler, LEGO, MATLAB.**

I. Idee

In diesem Projekt wird ein vollautomatischer Gabelstapler aus LEGO-Teilen konstruiert. Für die vollständige Automatisierung werden LEGO-Mindstorms-NXT-Komponenten verwendet, darunter Farbsensoren, Berührungssensoren und Elektromotoren. Durch den Einsatz dieser Sensoren kann der Roboter das Vorhandensein einer Last auf der Plattform erkennen, die Farbe der Palette erfassen und sie an einen bestimmten Ort im „Lager“ transportieren. Das Hauptmerkmal des Systems ist die Fähigkeit, vergleichsweise schwere Objekte, wie beispielsweise ein iPhone, anzuheben. Beim Entwurf des Designs wurden keine externen Prototypen aus dem Internet verwendet.

Abbildung 1: Seiten- und Draufsicht des Roboters

II. Konstruktion

Das Design wurde unabhängig und ohne die Verwendung von Prototypen Dritter entwickelt (Abb. 1).

A. Fahrantrieb

Bei der Konstruktion des Fahrantriebs müssen die spezifischen Anforderungen des Roboters berücksichtigt werden, insbesondere die Fähigkeit, sich in alle Richtungen zu bewegen, auf engem Raum zu manövrieren und auf der Stelle zu drehen. Für diese Anforderungen ist ein Raupenfahrwerk am besten geeignet, da es eine hohe Wendigkeit und Stabilität gewährleistet.

Der Antrieb erfolgt über zwei Elektromotoren, jeweils einen pro Raupe. Diese Lösung verbessert die Manövrierfähigkeit sowie die Geländegängigkeit auf begrenztem Raum. Zudem erhöht der Einsatz von zwei Motoren die Leistungsdichte des Roboters, sodass er auch unter Last höhere Geschwindigkeiten erreichen kann.

B. Hubantrieb

Der Hubantrieb muss so ausgelegt sein, dass er das Heben schwerer Gegenstände ermöglicht. Zu diesem Zweck wird ein einzelner Elektromotor verwendet, der ein Getriebe antreibt. Um das Übersetzungsverhältnis zu optimieren und die Reibung zu erhöhen, kommen zwei Zahnradpaare zum Einsatz, die durch ihre Rotation die an der Plattform befestigte Zahnstange vertikal bewegen.

Die gesamte Konstruktion ist horizontal geneigt, um die Hubkapazität am oberen Endpunkt zu maximieren. Durch diese Neigung wird das Gewicht der Plattform auf die Zahnräder übertragen, was die Stabilität und Effizienz des Hebevorgangs verbessert.

C. Plattform

Die Plattform muss so konstruiert sein, dass sie das Anheben von Paletten und deren Transport zu den Regalen ermöglicht. Zu diesem Zweck wurde eine Bauweise mit zwei Gabelzinken und darauf befindlichen Befestigungslaschen entwickelt. Diese Konstruktion erlaubt es, verschiedene Lasten sicher anzuheben, zu fixieren und zu den vorgesehenen Regalen zu transportieren.

D. Konstruktionsmerkmale

Die Hauptsteuereinheit sowie drei Elektromotoren wurden an die Rückseite des Roboters verlegt, um ein Gegengewicht zu erzeugen (Abb. 2). Diese Anordnung trägt zur Stabilisierung des Roboters bei und verbessert seine Balance während des Betriebs.

DOI: 10.24352/UB.OVGU-2025-031

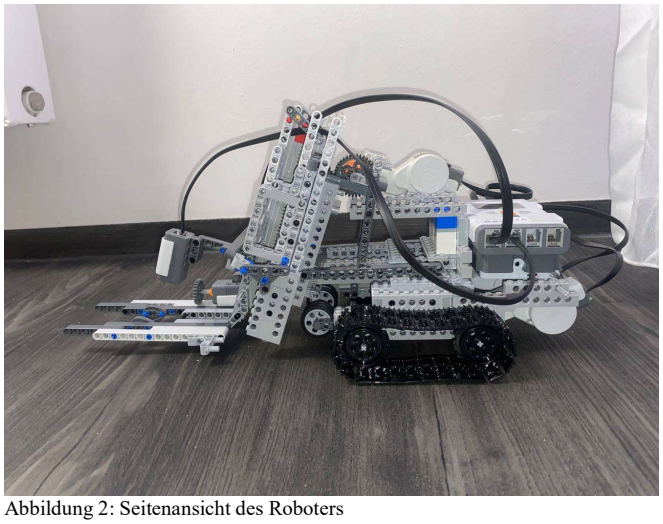


Abbildung 2: Seitenansicht des Roboters

E. Strukturelle Probleme und deren Lösungen

Ein zentrales Problem bei der Konstruktion stellte das Durchrutschen der Ritzel und Zahnräder im Hubmechanismus dar. Aufgrund der unzureichenden Stabilität des Hebemechanismus sowie des Spiels zwischen den Lego-Teilen hatte der Roboter Schwierigkeiten, schwere Objekte anzuheben.

Zur Lösung dieses Problems wurde der Hubmechanismus nach hinten geneigt, sodass das Gewicht der Last im Verhältnis zur Hubbewegung auf den Kontaktpunkt zwischen Zahnrädern und Zahnstange drückt. Zusätzlich wurde die gesamte Rahmenstruktur verstärkt und stabilisiert, um unerwünschte Spielräume zu minimieren und die Gesamtstabilität zu verbessern.

III. AUTOMATISIERUNG

A. Automatisierungsmöglichkeiten für Lego-Roboter

Die Lego-NXT-Steuereinheit wird zur Programmierung und Automatisierung des Roboters verwendet. Zur Realisierung der Funktionalität eines Gabelstaplerroboters kommen Drucksensoren zum Einsatz, um das Vorhandensein einer Last auf der Plattform zu erkennen. Zusätzlich wird ein Farbsensor integriert, der die Sortierung der Paletten ermöglicht.

B. Tastsensor

Das Lego-NXT-System verfügt über einen Tastsensor, der an der Hebebühne befestigt ist. Beim Annähern des Roboters an eine Palette wird eine Taste betätigt, wodurch die nachfolgenden Schritte im Algorithmus aktiviert werden. Auf diese Weise erfolgt die Erkennung einer Last auf der Plattform vollständig automatisiert.

C. Farbsensor

Eine weitere Aufgabe des Roboters besteht darin, die Last an einen definierten Ort zu transportieren. Zur Sortierung der Paletten wird ein Farberkennungssystem eingesetzt. Der Farbsensor des Lego-NXT-Systems erfasst die Farbe der Palette und steuert den Transport an den entsprechenden Zielort.

Die Farbmarkierung auf der Palette ist so positioniert, dass sie vom Sensor zuverlässig erfasst werden kann. Beispielsweise wird eine blaue Palette dem unteren Regal zugewiesen, während eine rote Palette im oberen Regal platziert wird.

D. Probleme und Lösungen

Ein zentrales Problem bei der Installation der Sensoren ist der begrenzte Erfassungsradius des Farbsensors. Experimentelle Untersuchungen haben gezeigt, dass der Sensorstrahl in einem Abstand von etwa 2 Zentimetern zuverlässig arbeitet (Abb. 3). Diese Einschränkung stellt besondere Anforderungen an die Konstruktion. Daher muss der Montageort des Sensors auf Grundlage der Abmessungen der Palette sowie der Position der Farbmarkierung präzise berechnet werden, um eine korrekte Farberkennung zu gewährleisten.

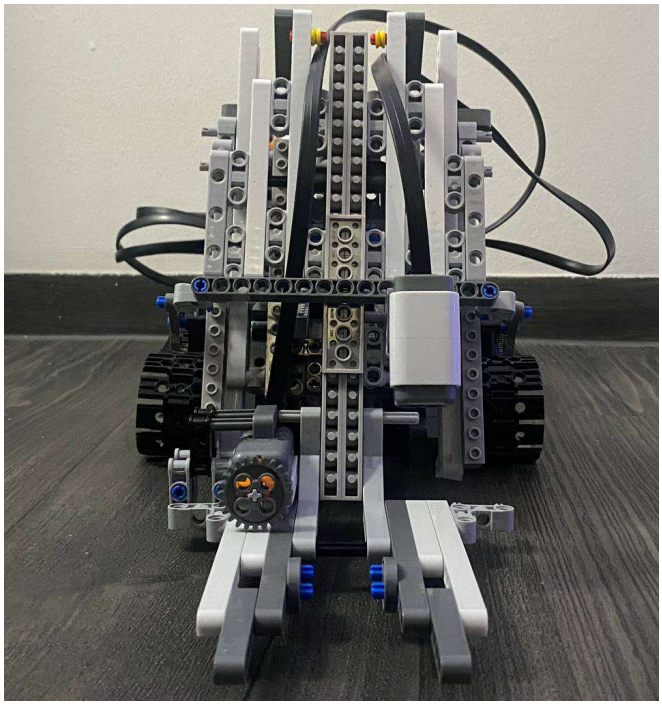


Abbildung 3: Vorderansicht des Roboters 1

IV. ROBOTER-ALGORITHMUS

Zu Beginn führt der Algorithmus des Roboters eine Demonstration aller seiner Funktionen und Eigenschaften durch.

A. Algorithmus

Der Ablauf des Roboters folgt dem folgenden Algorithmus:

1. Start an der Ausgangsposition
2. Rückwärtsfahrt
3. Annäherung an die erste Palette (rot auf dem Foto)
4. Anheben der Palette nach Betätigung des Sensors
5. Farbscan der Palette
6. Drehung in Richtung „Lager“

7. Platzierung der Palette auf dem vorgesehenen Regal (z. B. oberstes Regal)
8. Fahrt zur zweiten Palette (grün auf dem Foto)
9. Anheben der Palette nach Betätigung des Sensors
10. Farbscan der Palette
11. Drehung in Richtung „Lager“
12. Platzierung der Palette auf dem vorgesehenen Regal (z. B. unterstes Regal)
13. Abschluss des Algorithmus

B. Zusätzliche Gebäude

Zur Demonstration der Funktionsweise des Roboters werden verschiedene zusätzliche Strukturen auf der Lego-Plattform eingesetzt. Die wichtigste davon sind die Paletten, die speziell an die Größe der Gabelzinken des Gabelstaplers angepasst wurden und auf denen beliebige Objekte mit geeigneten Abmessungen platziert werden können. Jede Palette ist mit einer farbigen Markierung versehen, die für die Sortierung erforderlich ist.

Die Paletten werden an definierten Positionen aufgestellt, den sogenannten „Ladeterminals“ (Abb. 4). Zudem gibt es ein Regallager, in dem die Paletten entsprechend ihrer Farbmarkierung entweder im oberen oder unteren Bereich einsortiert werden. Darüber hinaus befindet sich auf der Plattform eine Startstation des Roboters, von der aus der gesamte Algorithmus initiiert wird.

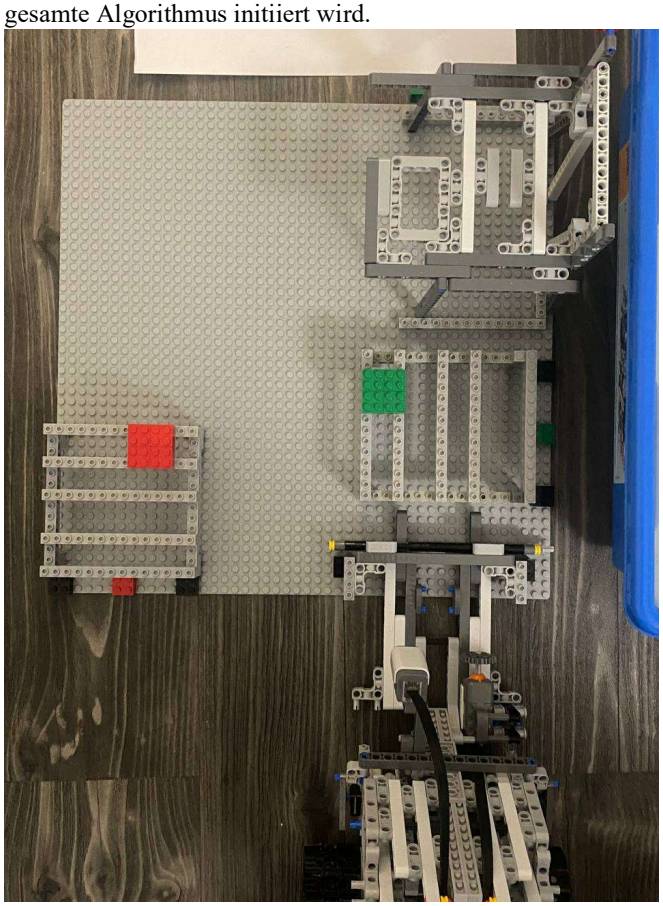


Abbildung 4: Ansicht des „Ladeterminal“ von oben

V. PROGRAMMIERUNG

Die Programmierung des Roboters erfolgt mithilfe der Software MATLAB unter Verwendung der zusätzlichen Bibliothek RWTH Mindstorms NXT Toolbox. Diese Bibliothek ermöglicht die Integration der Programmierumgebung mit dem Lego-NXT-System und erlaubt eine effiziente Steuerung des Roboters.

Im Folgenden wird ein Beispiel für einen Teil des Programmiercodes des Roboters vorgestellt (Abb. 5):

```
state = 0;

lastButtonState = -1;

while true

  currentButtonState = GetSwitch(SENSOR_1);

   if currentButtonState == 1 && lastButtonState == 0

    if state == 0

      mMoveA.Power = MOVE_MOTOR_POWER;

      mMoveB.Power = MOVE_MOTOR_POWER;

      mMoveA.SendToNXT();

      mMoveB.SendToNXT();

      state = 1;

    elseif state == 1

      mMoveA.Stop('brake');

      mMoveB.Stop('brake');

      mLiftC.Power = LIFT_MOTOR_POWER;

      mLiftC.TachoLimit = DEGREES_PER_CM;

      mLiftC.SendToNXT();

      mLiftC.WaitFor();

      mMoveA.Power = ROTATION_POWER;

      mMoveB.Power = ROTATION_POWER;

      mMoveA.TachoLimit = 540;

  mMoveB.TachoLimit = 540;
```

Abbildung 5: Teil des Programmiercodes

VI. LITERATURVERZEICHNIS

[1] Lego Mindstorm Wikipedia
https://en.wikipedia.org/wiki/Lego_Mindstorms
[2] RWTH - Mindstorms NXT Toolbox
https://de.mathworks.com/matlabcentral/fileexchange/18646-rwth-mindstorms-nxt-toolbox
[3] Roboter-Konstruktionen mit Lego Mindstorms NXT
https://www.generationrobots.com/de/content/60-roboter-konstruktionen-mit-mindstorms-nxt-v2
[4] Navigation von Wegpunkten mit einem Lego MINDSTORMS NXT Roboter
https://de.mathworks.com/matlabcentral/fileexchange/42835-navigating-waypoints-with-a-lego-mindstorms-nxt-robot

„Golfroboter“ mit Lego Mindstroms

Voloshyn Ivan, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

Abstract — An der Otto-von-Guericke-Universität Magdeburg findet jährlich ein Projektseminar für Elektrotechnik und Informationstechnik statt. Im Rahmen des aktuellen Seminars wurde ein Golf-Roboter entwickelt, der zur Automatisierung bestimmter Trainingsprozesse im Golfsport konzipiert wurde.
Für den Bau des Roboters wurden LEGO Mindstorms EV3-Komponenten verwendet, die eine flexible und modulare Konstruktion ermöglichen. Die Steuerung und Programmierung erfolgten mit MATLAB, wodurch eine präzise Verarbeitung der Sensordaten und eine effiziente Bewegungssteuerung realisiert wurden.

***Schlagwörter* — LEGO, Mindstorms, Golf-Roboter, MATLAB, Programm, Automatisierung.**

I. Einleitung

Die moderne Entwicklung der Robotik eröffnet immer mehr Möglichkeiten zur Automatisierung verschiedener Prozesse – von der industriellen Produktion und der Medizin bis hin zur öffentlichen Sicherheit. Immer häufiger finden robotergestützte Systeme Anwendung im Sport, wo sie dazu beitragen, das Training effizienter zu gestalten und Sportler von monotonen oder komplexen Abläufen zu entlasten. Einer der entscheidenden Faktoren für ein erfolgreiches Training ist die Wiederholbarkeit der Bewegungen, deren Erreichung jedoch eine hohe Konzentration und zahlreiche Wiederholungen erfordert.

Im Rahmen der Entwicklung des Golf-Roboters (Abb. 1) wurden folgende Schlüsselanforderungen an das System definiert:

- **Autonome Erkennung des Balls** mithilfe von Sensoren,
- **Präzise Annäherung an den Ball** durch ein Bewegungssteuerungssystem,
- **Einfache Konstruktion und Programmierung**, die eine leichte Erweiterung der Funktionalität ermöglicht.

Der Einsatz von Sensoren, automatisierten Steuerungsalgorithmen und adaptiven Technologien kann die Effizienz des Trainingsprozesses erheblich steigern.

Es ist jedoch wichtig, dass die Einführung solcher Technologien bewusst und ethisch verantwortungsvoll erfolgt, damit sie den größtmöglichen Nutzen bringen.
In diesem Bericht werden zunächst die theoretischen Grundlagen des Projekts erläutert, gefolgt von einer detaillierten Beschreibung der Konstruktion und Programmierung des Roboters. Abschließend werden die Ergebnisse analysiert und mögliche Verbesserungsansätze für das System vorgestellt

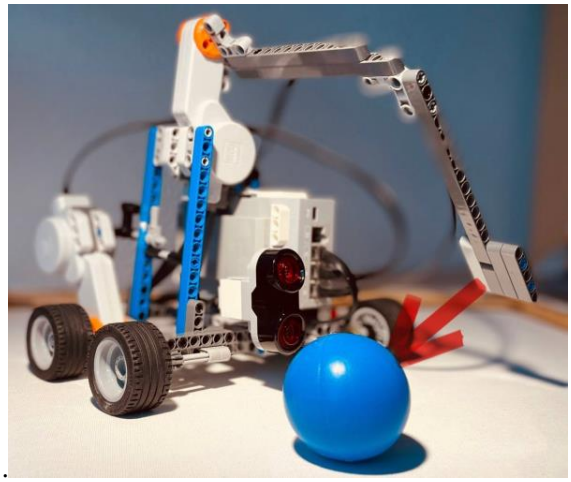


Abbildung 1: Golfroboter [1]

II. Vorbetrachtungen

Dieser Abschnitt bietet einen Überblick über die im Projekt verwendeten Technologien und beschreibt deren Funktionalität im Kontext der gestellten Aufgaben und der Verarbeitung von Sensordaten.

Das in diesem Projekt verwendete System basiert auf LEGO-Mindstorms-EV3-Komponenten, elektronischen Modulen und entsprechender Software, was eine flexible Konstruktion und individuelle Anpassung des Roboters ermöglicht. Die aktuelle Version EV3 verfügt über eine breite Auswahl an Sensoren und Motoren sowie eine benutzerfreundliche Programmierumgebung, die sie besonders für die Entwicklung autonomer Systeme geeignet macht.

DOI: 10.24352/UB.OVGU-2025-032

Im Vergleich zur vorherigen Version NXT bietet das Mindstorms-EV3-System erweiterte Funktionalitäten und eine verbesserte Lesbarkeit des Codes. Dadurch eignet es sich insbesondere für anspruchsvolle technische und wissenschaftliche Projekte, wie die Entwicklung und Programmierung autonomer Systeme.

III. Konstruktion und Programmierung

A. Aufbau

Die Entwicklung und der Bau des Golf-Roboters wurden in kurzer Zeit abgeschlossen, wobei die eigentliche Programmierung einen erheblichen Zeitaufwand erforderte, da sie in MATLAB umgesetzt wurde. Während des Konstruktionsprozesses kam ein Ultraschallsensor zum Einsatz, um den Golfball zuverlässig zu erkennen und darauf zu reagieren. Zur Fortbewegung des Roboters wurden zwei EV3-Motoren verwendet, die mit den Hinterrädern verbunden sind.

a) EV3-Motor

Abbildung 2: EV3-Motor [2]

Der LEGO-EV3-Motor (Abb 2) ermöglicht präzise Bewegungen und eine zuverlässige Steuerung des Roboters. Er ist mit einer eingebauten Sensorik zur Erfassung des Drehwinkels und der Geschwindigkeit ausgestattet, wodurch eine exakte Bewegungsausführung gewährleistet wird.

b) EV3-Ultraschallsensor

Abbildung 3: EV3-Ultraschallsensor [3]

Der EV3-Ultraschallsensor (Abb 3) misst Abstände zu Objekten mithilfe von Ultraschallwellen. Er wurde in diesem Projekt eingesetzt, um Hindernisse zu erkennen und den Ball auf dem Tisch zu lokalisieren. Diese Sensorik ermöglicht eine autonome Navigation und eine genaue Positionierung des Roboters.

Dieser Abschnitt fasst die wichtigsten Hardware-Komponenten zusammen, die für die Erkennung des Balls, die Bewegung des Roboters und die Verarbeitung der Sensordaten verwendet wurden.

B. Programmierung

Für die Umsetzung des Projekts wurden LEGO-Bauteile, elektronische Komponenten und spezielle Software verwendet, um eine flexible Konstruktion und individuelle Anpassung des Roboters zu ermöglichen. Das aktuelle EV3-System bietet eine Vielzahl an Sensoren, Motoren und eine intuitive Programmierumgebung, wodurch die Entwicklung effizient realisiert werden konnte. Im Rahmen des Universitätsprojekts „Otto von Guericke 2025“ wurde mit dieser Technologie ein autonomer Roboter erfolgreich konstruiert. Im Vergleich zur vorherigen NXT-Generation zeichnet sich LEGO Mindstorms EV3 durch erweiterte Funktionalitäten und eine verbesserte Benutzerfreundlichkeit aus. Dies macht es zu einer idealen Plattform für komplexe und anspruchsvolle Robotik-Projekte.

Eine der Entwicklungsphasen war die Planung des Algorithmus des Roboters, der in Form eines Flussdiagramms dargestellt wurde (Abb 4). Das Flussdiagramm stellt die Logik der Aufgabenausführung visuell dar. Dieses grafische Werkzeug spielt eine wichtige Rolle bei der Strukturierung des Programmcodes und erleichtert die Umsetzung des Algorithmus

Die Programmierung des Roboters erfolgte in MATLAB (Matrix Laboratory) – einer von MathWorks entwickelten Umgebung, die leistungsstarke numerische Berechnungen, Simulationen und Datenvisualisierungen ermöglicht. In der Robotik spielt MATLAB eine wesentliche Rolle, insbesondere bei der Entwicklung von Steuerungsalgorithmen, der Simulation von Roboterdynamiken und der Verarbeitung von Sensordaten. Daher sind Grundkenntnisse in MATLAB besonders wertvoll, um solche Systeme erfolgreich umzusetzen.

C. Problemen

Die größten Herausforderungen ergaben sich im Zusammenhang mit dem Sensor, dessen korrekte Funktion eine wiederholte Anpassung des Codes erforderte. Um eine optimale Leistung des Roboters zu gewährleisten, musste der Code insgesamt viermal erheblich überarbeitet werden. Zusätzlich gab es kleinere Schwierigkeiten mit den beiden Motoren, die für die Fortbewegung des Roboters verantwortlich sind. Insbesondere die

Geschwindigkeitssteuerung erwies sich als problematisch. Dieses Problem konnte jedoch durch gezielte Anpassungen im Code erfolgreich behoben werden.

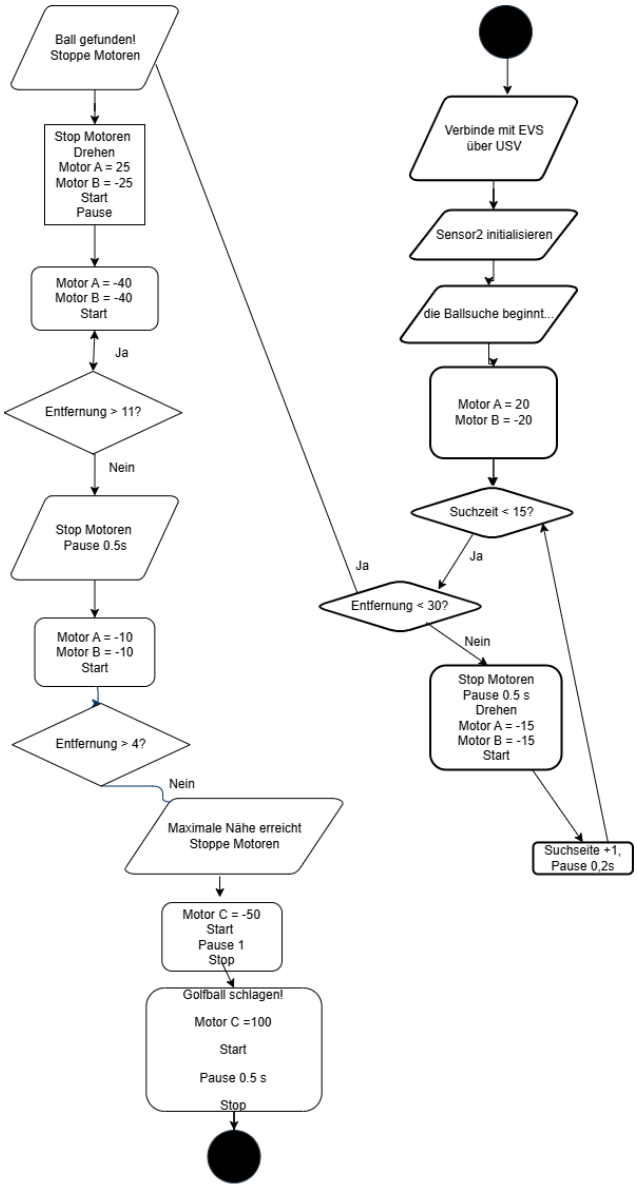


Abbildung 4: Programmablaufplan

IV. Ergebnisdiskussion

Im Rahmen des Projekts konnte das gesetzte Ziel erfolgreich erreicht werden: Der Roboter funktioniert stabil, und sowohl die Sensoren als auch die Motoren reagieren präzise und synchron auf Steuerbefehle. Die durchgeführten Codeanpassungen haben dazu beigetragen, eine zuverlässige Leistung sicherzustellen.
Diese hohe Funktionssicherheit zeigt das Potenzial für eine vielseitige Anwendung in unterschiedlichen Umgebungen und Szenarien.

V. Zusammenfassung und Fazit

Der entwickelte Golfroboter wurde als System zur Erkennung und zum wiederholbaren Schlagen eines Golfballs konzipiert. Während der Entwicklung kamen LEGO Mindstorms EV3, verschiedene Sensoren und MATLAB zum Einsatz, wobei der Fokus auf einer stabilen Mechanik und einer zuverlässigen Steuerung lag.

Während der Entwicklung traten Herausforderungen bei der Kalibrierung der Sensoren und der exakten Übertragung von Steuerbefehlen auf. Durch eine iterative Anpassung des Codes konnten jedoch diese Probleme behoben und die Funktionsfähigkeit des Roboters optimiert werden.

Das Projektziel wurde erfolgreich erreicht: Der Roboter lokalisiert den Ball zuverlässig, bewegt sich gezielt in dessen Richtung und führt einen stabilen Schlag aus. Dies bestätigt die Praxistauglichkeit des Systems. In Zukunft könnten die Mechanik verbessert, das Sensorsystem erweitert und detailliertere Analysen der Schlagtechnik integriert werden, um den Roboter als fortschrittliches Trainingsgerät für Golfspieler weiterzuentwickeln.

Literaturverzeichnis

[1] Projekt Instagram – Gruppe 14

[2] LEGO MINDSTORMS EV3 Servomotor, Amazon.

[3] *LEGO MINDSTORMS Education EV3 Ultraschallsensor*

Mykola Hnatushenko, Elektro-und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

“Golfroboter” mit Lego Mindstroms

Zusammenfassung – Jährlich findet an der Otto-von-Guericke-Universität Magdeburg das Projektseminar zur Elektro- und Informationstechnik statt. Im Rahmen der diesjährigen Projektwerkstatt wurde ein spezieller „Golf-Roboter“ entwickelt, der dazu dient, bestimmte Übungsabläufe im Golfsport zu automatisieren und dieBelastung für den Anwender zu reduzieren. Für den Aufbau des Roboters wurden LEGO-Mindstorms-Sets sowie der Steuercomputer LEGO EV3 verwendet.. Die Software wurde mithilfe von MATLAB realisiert. In diesem Beitrag wurden sowohl die Konstruktion als auch die Funktionsweise des „Golf-Roboters“ vorgestellt. Darüber hinaus wurden technische Herausforderungen beleuchtet, die während des Entwicklungsprozesses auftraten, und Lösungen erläutert, mit denen diese Schwierigkeiten erfolgreich behoben wurden.

***Schlagwörter*— Automatisierung, Golf-Roboter, MATLAB, Mindstorms, Robotik, Sensortechnik**

I. EINLEITUNG

Die modernen Fortschritte in der Robotik ermöglichen eine effektive Bewältigung eines breiten Spektrums an Aufgaben – von der industriellen Fertigung und der Medizin bis hin zum Bereich der öffentlichen Sicherheit. Doch finden robotische Systeme auch im Sport Anwendung, wo sie dazu beitragen, die Qualität des Trainings zu erhöhen, das Verletzungsrisiko zu verringern und Sportler von monotonen oder komplexen Abläufen zu entlasten. Ein anschauliches Beispiel stellt der sogenannte „Golf-Roboter“ (Abb. 1) dar, der routinemäßige Elemente des Trainingszyklus übernimmt und dabei eine hohe Genauigkeit sowie Wiederholbarkeit der Bewegungsabläufe gewährleistet.

Angesichts der steigenden Anforderungen an Sicherheit und Multifunktionalität scheint eine stetige Weiterentwicklung solcher Systeme unvermeidlich. Eine präzise Messdatenerfassung, automatisierte Analysen und die Möglichkeit einer fein abgestimmten Trainingssteuerung versprechen einen bedeutenden Beitrag zur Weiterentwicklung des modernen Sports. Ein verantwortungsvoller Umgang mit neuen Technologien ist dabei von entscheidender Bedeutung: Nur durch sorgfältige Planung und die Einhaltung klarer ethischer Standards kann sichergestellt werden, dass sie zum Wohle der Gesellschaft eingesetzt werden. Indem wir uns an diese Grundsätze halten, können wir das Potenzial der Robotik – einschließlich innovativer Lösungen wie dem „Golf-Roboter“ – voll ausschöpfen und mögliche Risiken auf ein Minimum reduzieren.

Abbildung 1: Golfroboter [1]

II. VORBETRACHTUNGEN

Dieser Abschnitt gibt einen Überblick über die wichtigsten in diesem Projekt eingesetzten Werkzeuge und deren Funktionen im Hinblick auf die Ausführung spezieller Aufgaben und die Verarbeitung von Eingaben.

Roboterprogrammierung**.** Das hierbei genutzte System setzt sich aus LEGO-Bauteilen, elektronischen Komponenten und passender Software zur Konstruktion und individuellen Anpassung von Robotern zusammen. Die aktuelle EV3-Generation umfasst eine umfangreiche Auswahl an Sensoren und Motoren sowie eine benutzerfreundliche grafische Programmierumgebung. Im Rahmen des Universitätsprojekts „Otto von Guericke 2025“ wurde mithilfe dieser EV3-Technologie ein Roboter konzipiert und realisiert, wobei die Umsetzung von Aufgaben durch die intuitive und flexible Plattform deutlich vereinfacht und der Entwicklungsprozess beschleunigt wurde. Gegenüber dem NXT-Modell zeichnet sich das neueste Mindstorms-EV3-System durch einen erweiterten Funktionsumfang und gesteigerte Bedienerfreundlichkeit aus, was es ideal für anspruchsvolle technische und wissenschaftliche Vorhaben wie etwa den Bau und die Programmierung eines Manipulators gemacht hat.

III. KONSTUKTION UND PROGRAMMIERUNG

A. Aufbau Der Entwurf und die Konstruktion des Golfroboters dauerten nicht allzu lange, während die Programmierung selbst als ein zeitaufwändiger Prozess angesehen wurde, der die Programmierung in Matlab erforderte. Auch während des Konstruktionsprozesses wurde ein Entfernungssensor verwendet, um den Ball korrekt zu suchen und anzuspielen. Für den Antrieb des Roboters wurden zwei große Motoren verwendet, die mit den Hinterrädern verbunden waren. Der verwendete EV3-Motor ermöglicht präzise Bewegungen durch integrierte Sensorik zur Erfassung von Drehwinkel und Geschwindigkeit. Im Projekt wurde der

DOI: 10.24352/UB.OVGU-2025-033

Motor nicht nur für die Fortbewegung eingesetzt, sondern auch zur Steuerung des Schlages. Besonders herausfordernd war die Synchronisation beider Antriebsmotoren, die durch gezielte Codeanpassungen erreicht wurde. Der EV3-Ultraschallsensor misst mithilfe von Ultraschallwellen den Abstand zu Objekten. In diesem Projekt wurde er zur Erkennung des Golfballs verwendet. Dabei musste berücksichtigt werden, dass der Sensor eine minimale Distanz zur Erfassung benötigt. Deshalb wurde eine Bewegung in Richtung des Balls eingeplant, bis die Sensorreichweite eine eindeutige Erkennung ermöglichte.

Abbildung 2: EV3-Motor [2]

Abbildung 3: EV3-Ultraschallsensor [3]

A. Roboterprogrammierung.

Zum Einsatz kommen LEGO-Bauteile, elektronische Komponenten und Software, um Roboter individuell zu bauen und anzupassen. Die aktuelle EV3-Generation umfasst eine umfangreiche Auswahl an Sensoren und Motoren sowie eine benutzerfreundliche grafische Programmierumgebung. Im Universitätsprojekt „Otto von Guericke 2025“ konnte so ein Roboter effizient entwickelt werden. Gegenüber NXT bietet Mindstorms EV3 mehr Funktionen und bessere Bedienbarkeit, was es für anspruchsvolle Projekte besonders geeignet macht.

MATLAB (Matrix Laboratory), eine von MathWorks entwickelte Umgebung, ermöglicht numerische Berechnungen, Simulationen und Datenvisualisierungen. Gerade in der Robotik spielt MATLAB eine zentrale Rolle, etwa bei der Entwicklung von Steuerungsalgorithmen, der Simulation von Roboterdynamik oder der Auswertung von Sensordaten. Grundkenntnisse in MATLAB sind daher sehr hilfreich, um solche Mechanismen erfolgreich zu realisieren

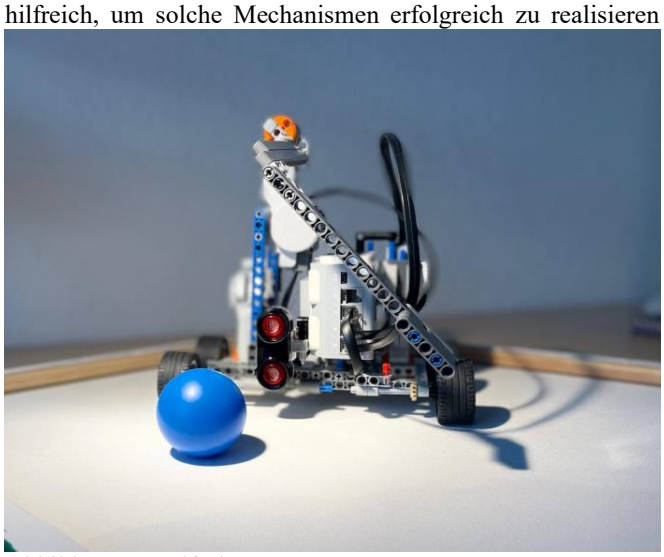


Abbildung 4: Golfroboter

Abbildung 5: Programmablaufplan

B. Probleme

B. Roboterprogrammierung / Programmablaufplan (PAB)
Das Herzstück des Projekts bildete der detaillierte Programmablaufplan (PAB), der in MATLAB entwickelt und

umgesetzt wurde. Der PAB beschreibt die logische Abfolge der Aktionen des Roboters und wurde so konzipiert, dass er autonom auf Umweltveränderungen reagieren kann. Der Ablaufplan lässt sich wie folgt zusammenfassen:

1. Initialisierung aller Sensoren und Motoren.
2. Aktive Suche nach einem Objekt mittels Ultraschallsensor – der Roboter dreht sich schrittweise links und rechts, bis ein Objekt in Reichweite erkannt wird.
3. Sobald ein Objekt erkannt wird, fährt der Roboter darauf zu.
4. In näherer Distanz (ca. 6 cm) erfolgt ein Stopp und eine Verifizierung des Objekts über die Farberkennung (optional).
5. Wenn das Objekt als Golfball erkannt wird, löst der Roboter den Schlagmechanismus aus.
6. Andernfalls wird der Suchvorgang erneut gestartet.

Der gesamte Ablauf wurde mithilfe von Stateflow-Diagrammen in MATLAB modelliert und über Simulink implementiert. Besonderes Augenmerk lag auf der robusten Fehlerbehandlung, beispielsweise beim Ausfall des Sensorsignals oder bei unvollständiger Erkennung. Mehrere Testläufe wurden durchgeführt, um die Stabilität der Zustandsübergänge zu gewährleisten.

IV. Zusammenfassung und Fazit

Der Golfroboter wurde als System für präzise, wiederholbare Schläge auf den Golfball konzipiert. Während der Entwicklung wurden LEGO Mindstorms EV3, verschiedene Sensoren und MATLAB eingesetzt, wobei besonderes Augenmerk auf das optimale Design und die Algorithmen zur Schlagkraftberechnung gelegt wurde. Die Hauptschwierigkeiten betrafen die zuverlässige Befehlsübermittlung und die Kalibrierung der Sensoren; jedoch ermöglichte eine iterative Vorgehensweise die Behebung von Messungenauigkeiten. Letztendlich wird der Ball vom Roboter mit ausreichender Präzision geschlagen und es wird zuverlässig auf Veränderungen in seiner Umgebung reagiert – ein deutlicher Beleg für das Erreichen des Hauptziels. Zukünftig wird geplant, die Mechanik weiter zu verbessern, das Sensorsystem auszubauen und eine erweiterte Schlaganalyse zu integrieren, damit der Roboter sowohl von Amateuren als auch von Profis als vollwertiges Trainingsgerät genutzt werden kann.

Literaturverzeichnis

[1] https://golf-event.ru/2022/12/02/golf-robot/

[2] LEGO MINDSTORMS EV3 Servomotor, Amazon.

[3]https://wiki.hshl.de/wiki/index.php/Objekterkennung_mit_rotierenden_Ultraschall_mit_Matlab/Simulink_und_EV3

„Roboterlader“ mit LEGO Mindstorms

Volodymyr Drovnikov, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

Jedes Jahr findet an der Otto-von-Guericke-Universität Magdeburg ein Design-Workshop statt, bei dem Studierende der Ingenieurwissenschaften einen Roboter aus LEGO entwerfen. Im Rahmen des diesjährigen Projekts habe ich einen besonderen Mechanismus geschaffen, ich bin „Roboterlader“. Dieses Projekt trägt dazu bei, das Risiko von Verletzungen und Schäden am Arbeitsplatz, insbesondere in Lagerhallen, zu verringern. Für die Gestaltung kamen LEGO-Mindstorms-Sets und der NXT-Steuerungscomputer zum Einsatz. Dieser Artikel beschreibt die Struktur und Funktionen des Mechanismus. Auch die bei der Gestaltung aufgetretenen Probleme und deren Lösungen wurden erwähnt.

Schlagwörter — Hinderniserkennung, Ultraschall, Roboter, MATLAB, Programm.

I. Einleitung

Die rasanten Fortschritte in der Robotik eröffnen neue Möglichkeiten, um auf globale Herausforderungen zu reagieren – sei es in der Industrie, der Medizin oder der öffentlichen Sicherheit. Besonders eindrucksvoll zeigte sich dies nach der Katastrophe in Fukushima, als spezialisierte Roboter eingesetzt wurden, um die hochgradig verstrahlten Gebiete zu erkunden und Informationen zu sammeln (Abb. 1). Ohne ihr Eingreifen wären unzählige Menschenleben in Gefahr gewesen.

Angesichts der zunehmenden Risiken in Krisensituationen – von Naturkatastrophen bis hin zu industriellen Unfällen – ist die Weiterentwicklung dieser Technologie unerlässlich. Intelligente, vielseitige Roboter können nicht nur Effizienz und Sicherheit erhöhen, sondern auch das Risiko für Menschen in gefährlichen Umgebungen erheblich reduzieren.

Doch mit der steigenden Bedeutung der Robotik wächst auch die Verantwortung. Es reicht nicht aus, neue Technologien nur zu entwickeln – wir müssen sicherstellen, dass sie ethisch vertretbar und im Sinne der Gesellschaft eingesetzt werden. Eine klare Regulierung und durchdachte Strategien sind erforderlich, um die Potenziale der Robotik zu maximieren und gleichzeitig Risiken zu minimieren.

Die Zukunft gehört den Maschinen – doch es liegt in unseren Händen, sie klug und verantwortungsbewusst zu nutzen.

II. Vorbetrachtungen

In diesem Abschnitt werden die zentralen Werkzeuge dieses Projekts vorgestellt und ihre Funktionen im Zusammenhang mit der Durchführung spezifischer Aufgaben oder der Reaktion auf Eingaben beschrieben.

Roboterprogrammierung. Die Technologie basiert auf LEGO-Bausteinen, elektronischen Komponenten und spezieller

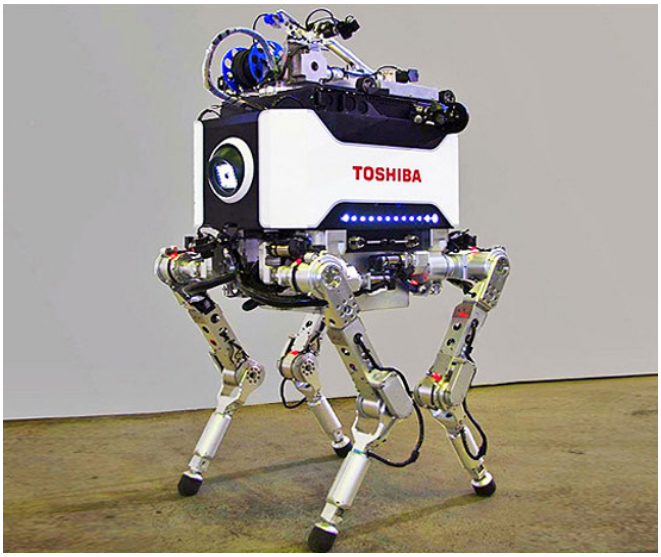


Abbildung 1: Tetrapode-Roboter in Fukushima [1]

Software zur Entwicklung und Steuerung von Robotern. Die aktuelle Version von NXT bietet eine breite Auswahl an Sensoren und Motoren sowie eine benutzerfreundliche grafische Programmierumgebung.

Im Rahmen des Otto-von-Guericke-Universitätsprojekts 2025 wurde ein Roboter auf Basis von NXT entwickelt und gebaut. Diese Plattform erleichtert die Umsetzung technischer Konzepte erheblich und beschleunigt den Entwicklungsprozess dank ihrer intuitiven und skalierbaren Struktur. Die neueste Version von Mindstorms NXT bietet eine erweiterte Funktionalität und noch höhere Benutzerfreundlichkeit im Vergleich zu ihrem Vorgänger. Dadurch eignet sie sich besonders für wissenschaftliche und technische Anwendungen, etwa die Entwicklung und Programmierung eines Manipulator-Roboters.

MATLAB als Entwicklungsumgebung. MATLAB (Matrix Laboratory) ist eine von MathWorks entwickelte Programmiersprache und Entwicklungsumgebung. Sie ermöglicht numerische Berechnungen, Datenvisualisierung, Simulationen und Analysen. MATLAB bietet leistungsstarke Werkzeuge für die Matrixmanipulation, Signal- und Bildverarbeitung sowie für mathematische und ingenieurwissenschaftliche Berechnungen.

In der Robotik kann MATLAB vielseitig eingesetzt werden – von der Entwicklung und Erprobung von Steuerungsalgorithmen über die Simulation der Roboterdynamik bis hin zur Sensordatenanalyse oder dem Entwurf eines

DOI: 10.24352/UB.OVGU-2025-034

Überwachungssystems. Daher sind grundlegende Programmierkenntnisse in MATLAB besonders wertvoll für die erfolgreiche Umsetzung technischer Mechanismen.

III. Roboterentwicklung

A. *Aufbau*

Die Konstruktion eines Roboters ist technisch simpel. Doch damit das System effektiv arbeiten kann, muss es nicht nur äußere Reize wahrnehmen, sondern auch auf diese reagieren und sich entsprechend bewegen. Dafür spielen Motoren und Sensoren eine entscheidende Rolle: Sie ermöglichen es dem Roboter, Lasten zu manövrieren, präzise Bewegungen auszuführen und auf verschiedene Umweltbedingungen zu reagieren. Im Folgenden werden die wichtigsten Sensoren und Motoren sowie ihre spezifischen Eigenschaften aufgeführt, die für eine optimale Funktionalität des Roboters erforderlich sind.

- *Motoren*

Für den Bau des Mechanismus werden drei große Motoren benötigt (Abb.2). Zwei davon steuern die Vorwärts- und Rückwärtsbewegung sowie das Abbiegen des Roboters, während der dritte Motor für das Anheben der Klaue verantwortlich ist. Diese Kombination ermöglicht eine präzise Steuerung und flexible Bewegungen, wodurch der Roboter effizient auf verschiedene Aufgaben reagieren kann.

Abbildung 2: großer Motor [2]

- *Farben Sensor*

Um eine präzise Steuerung zu gewährleisten, sind zwei Farbsensoren (Abb. 3) symmetrisch an der Unterseite des Roboters angebracht. Ihre Hauptaufgabe besteht darin, die Farbe Schwarz zu erkennen und somit die Position des Roboters auf der Strecke zu überwachen. Sobald einer der Sensoren eine schwarze Linie erfasst, passt der Roboter automatisch seine Bewegung an, um auf Kurs zu bleiben. Dieses intelligente Steuerungssystem ermöglicht eine zuverlässige Navigation und verhindert ungewollte Abweichungen.

- *Ultraschallsensor*

Ein moderner Roboter muss nicht nur Objekte erkennen, sondern auch intelligent darauf reagieren. Dafür kommt ein Ultraschallsensor (Abb.4) zum Einsatz, der die Distanz zu Boxen misst und ihre Position bestimmt. Nach der ersten Messung erfasst der Sensor eine zweite Distanz, vergleicht beide Werte und entscheidet, ob eine Anpassung notwendig ist. Falls die Höhenunterschiede ausgeglichen werden müssen, greift die Klaue ein und positioniert die Box präzise auf das gewünschte Niveau. Diese Technologie ermöglicht eine effiziente und automatisierte Objekthandhabung, die besonders in logistischen und industriellen Anwendungen von großem Vorteil ist.

Abbildung 3: Farbsensor [2]

B. Programmierung

Die Verbindung von moderner Programmierung und innovativer Technik spielt eine entscheidende Rolle in der Robotik. Der NXT-Controller fungiert als Steuerzentrale eines Roboters und ermöglicht die präzise Ansteuerung von Motoren und Sensoren über verschiedene Schnittstellen, darunter USB. Gesteuert wird das System über MATLAB, eine leistungsstarke Plattform für numerische Berechnungen und automatisierte Prozesse.

Abbildung 4: Ultraschallsensor [2]

Mit dem NXT-Toolkit lassen sich Sensordaten erfassen und Motorbewegungen exakt steuern. Eine klare Kennzeichnung der Anschlüsse – Buchstaben für Motoren, Zahlen für Sensoren – erleichtert dabei die Konfiguration. Die NXT-Toolbox stellt eine Vielzahl von Befehlen zur Verfügung, die eine intuitive Programmierung ermöglichen: So steuert der Befehl „*motorA.setPower(-30)*“ die Motorleistung, während „*SensorUltrasonic.getDistance()*“ die Distanzmessung mit dem Ultraschallsensor durchführt.

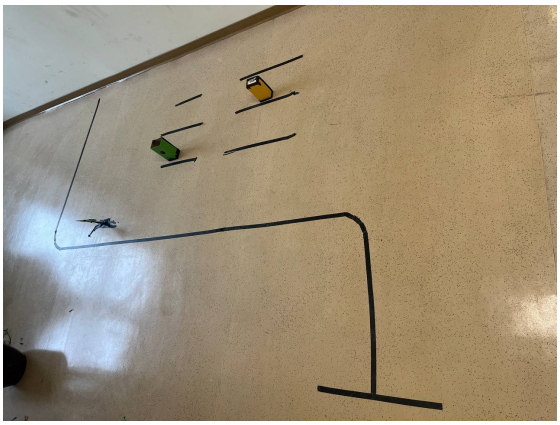


Abbildung 5: Fahrfeld des „Roboterladers“

MATLAB bietet neben klassischen Programmierstrukturen wie Schleifen und Bedingungen eine direkte Schnittstelle zur Hardware. Der NXT-Block empfängt Befehle vom Computer und überträgt sie an den Roboter, der daraufhin die gewünschten Aktionen ausführt. Durch einfaches Drücken der „Run“-Taste startet der gesamte Prozess – eine effiziente Lösung für den schnellen Einsatz in technischen und wissenschaftlichen Projekten.

```
COM_CloseNXT all;
h = COM_OpenNXT();
COM_SetDefaultNXT(h);
OpenLight(SENSOR_1, 'ACTIVE');
OpenLight(SENSOR_2, 'ACTIVE');
motorLeft = MOTOR_B;
motorRight = MOTOR_C;
while true
   leftValue = GetLight(SENSOR_1);  ка
   rightValue = GetLight(SENSOR_2);

   if leftValue >= 500 && leftValue <= 600
      NXT_SetOutputState(motorLeft, -30, 0, 'Brake', 0, 0, 0);
      NXT_SetOutputState(motorRight, 30, 0, 'Brake', 0, 0, 0);
   else
      NXT_SetOutputState(motorLeft, 30, 0, 'Brake', 0, 0, 0);
      NXT_SetOutputState(motorRight, 30, 0, 'Brake', 0, 0, 0);
   end
   pause(0.1);
end
CloseSensor(SENSOR_1);
CloseSensor(SENSOR_2);
COM_CloseNXT(h);
```

Abbildung 6: Codekette

In (Abb. 6) wird ein Beispielcode präsentiert, der zur Anpassung des Roboters dient, wenn der linke Sensor mit einer schwarzen Linie in Kontakt kommt.

C. Probleme

Während der Entwicklung des Roboters wurden viele Probleme festgestellt. Eines der interessantesten soll hier hervorgehoben werden: Das erste Problem bestand darin, dass ein EV3-Lichtsensor an den NXT angeschlossen wurde, was dazu führte, dass falsche Werte geliefert wurden. Dieses Problem konnte jedoch leicht behoben werden, indem ein für den NXT geeigneter Sensor verwendet und ausgetauscht wurde.

Als der Roboter bereits in der Lage war, der schwarzen Linie zu folgen (Abb. 5), stellte sich die Herausforderung, zwei Kästchen zu erfassen, ohne unnötige Werte in die Variablen einzutragen. Ursprünglich wurde die Idee verfolgt, eine fahnenförmige Identifikationsmarkierung anzubringen, um die Entfernung festzulegen, bis zu der ein Codeblock ausgeführt werden sollte, um die Position der Kästchen zu bestimmen. Jedoch wurde festgestellt, dass der Roboter nicht erkannte, dass dasselbe Kästchen mehrfach erfasst wurde. Aus diesem Grund wurde die Idee der Markierung verworfen.

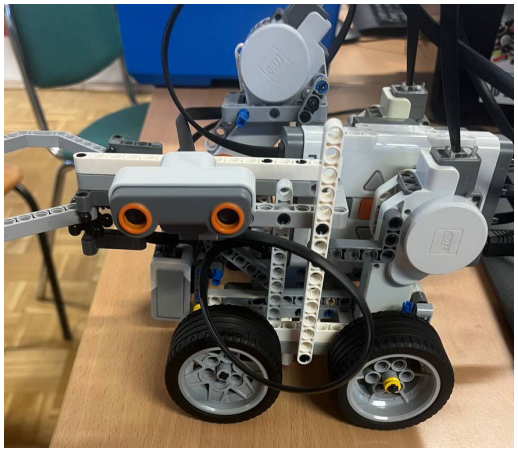


Abbildung 7: fertiger „Roboterlader“

Dieses Problem wurde gelöst, indem die Entfernung nur dann gemessen wurde, wenn sie unter einem bestimmten Wert lag. Die Messung wurde dann gestoppt, bis der Roboter den Raum zwischen den Kästchen erreichte und erneut einen Wert über dem in der Bedingung definierten Schwellenwert registrierte.

Auch bei den Farbsensoren traten Probleme auf, da der verwendete Sensortyp den Farberkennungsmodus nicht unterstützte. Daher wurde eine Analyse der Lichtreflexionen verschiedener Farben durchgeführt, um diese zur Bestimmung der jeweiligen Farbe heranzuziehen.

IV. Ergebnisdiskussion

Die Erreichung des gesetzten Ziels wurde bestätigt. Der Mechanismus (Abb. 7) funktioniert fehlerfrei; die Sensoren und Motoren arbeiten synchron und präzise gemäß der Befehlssequenz. Ein Mechanismus, der eine solche zuverlässige Leistung zeigt, besitzt das Potenzial für eine erfolgreiche Integration in verschiedene Umgebungen und reale Anwendungen. Das Ziel des Projekts wurde erfolgreich erreicht.

Literaturverzeichnis

[1] ***Liquidationsroboter in Fukushima***
https://www.zeit.de/wissen/2011-04/roboter-katastrophen

[2] ***LEGO NXT***

https://www.lego.com/cdn/product-assets/product.bi.core.pdf/4520734.pdf

[3] ***WIKIPEDIA: LEGO NXT***
https://de.wikipedia.org/wiki/Lego_Mindstorms_NXT

[4] ***SOFTWARE LEGO NXT***
https://www.lego.com/de-de/service/help/mindstorms_nxt/mindstorms_nxt/lego-mindstorms-nxt-software-downloads-kA009000001dck4CAA

[5] ***Projekt Instagram:***
https://www.instagram.com/legomindstormsgruppe15wise2025?

Roboterlader

Hlib Horbachov, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Jedes Jahr veranstaltet die Otto-von-Guericke-Universität Magdeburg eine Projektwerkstatt für Elektro- und Informationstechnik. Im Rahmen des diesjährigen Seminars wurde ein spezieller Mechanismus, ein so genannter „Roboterlader", entwickelt. Er hilft beim Transport von Lasten, bewegt sich selbstständig auf einer schwarzen Linie und parkt sich auch selbstständig in einer von drei vordefinierten Positionen ein. Für den Entwurf und die Entwicklung des Mechanismus wurden LEGO Mindstorms Sets und ein LEGO NXT Steuercomputer verwendet. Die Software wurde mit MATLAB implementiert. In diesem Beitrag werden der Aufbau und die Funktionsweise des Mechanismus vorgestellt. Darüber hinaus werden die Probleme, die während des Entwurfsprozesses aufgetreten sind, und die Möglichkeiten zu ihrer Lösung analysiert.**

***Schlagwörter*—Hinderniserkennung, Ultraschall, Lichtsensor, Roboter, Matlab, Programm**

I. Einleitung

DIE moderne Robotik gibt uns die Möglichkeit, verschiedene alltägliche Aufgaben effektiv zu automatisieren, sei es in Unternehmen, im Haushalt oder in der Medizin. Im Jahr 2020, während der COVID-19-Pandemie, erlebte der Online-Handel einen Aufschwung, und viele Unternehmen sahen sich mit dem Problem verspäteter Lieferungen konfrontiert. Amazons Lagerhäuser haben die Situation jedoch mit autonomen Gabelstaplerrobotern gerettet, die in Logistikzentren auf der ganzen Welt eingesetzt werden [1].

Diese von Kiva Systems [2] (2012 von Amazon gekauft) entwickelten Roboter (Abbilding 1) sind in der Lage, schnell die richtigen Waren zu finden, sie den Mitarbeitern zum Verpacken zu übergeben und die Lagerprozesse zu optimieren. Dank dieser Roboter konnte Amazon die Geschwindigkeit der Auftragsabwicklung verdoppeln und Millionen von Menschen helfen, ihre Weihnachtsgeschenke rechtzeitig zu erhalten.

Dieser Fall zeigt, wie Robotik nicht nur die Effizienz steigern, sondern auch den Menschen echte Vorteile bringen kann, insbesondere in schwierigen Zeiten!

II. Vorbetrachtungen

In diesem Abschnitt werden die wichtigsten Werkzeuge vorgestellt, die in diesem Projekt verwendet werden, sowie ihre Funktionen zur Durchführung bestimmter Aufgaben und zur Verarbeitung von Eingabedaten.

Roboterprogrammierung. Die Technologie basiert auf der Verwendung von LEGO Bausteinen, elektronischen Modulen und Software zur Erstellung und Konfiguration von Robotersystemen. In diesem Projekt wurde LEGO NXT verwendet, das trotz einer neueren Version von EV3 eine robuste und funktionelle Entwicklungsplattform darstellt. NXT bietet eine breite Palette von Sensoren und Motoren sowie eine visuelle

DOI: 10.24352/UB.OVGU-2025-035

Abbildung 1. Amazon Laderoboter

Programmierung und ist damit ein praktisches Werkzeug für die Konzeption und Realisierung von Roboterlösungen. Im Rahmen des Universitätsprojekts Otto von Guericke 2024 wurde die LEGO-NXT-Plattform zur Entwicklung eines Roboters verwendet, der in der Lage ist, Aufgaben effizient auszuführen, den Konstruktionsprozess zu beschleunigen und den Maschinenbau zu vereinfachen.

MATLAB (Matrix Laboratory) ist eine von MathWorks entwickelte Softwareumgebung und Programmiersprache, die Funktionen für numerische Analyse, Modellierung und Datenverarbeitung bietet [3]. MATLAB enthält Werkzeuge für die Matrix-, Signal- und Bildverarbeitung und ist daher für die Analyse und Steuerung von Robotersystemen sehr nützlich.

III. Konstuktion und Programmierung

A. Aufbau

Das Design des Roboters zeichnet sich durch technische Einfachheit aus. Um effektiv arbeiten zu können, muss das System jedoch nicht nur externe Signale wahrnehmen, sondern auch in geeigneter Weise darauf reagieren, um Bewegungen zu ermöglichen. Motoren und Sensoren spielen dabei eine Schlüsselrolle, denn sie ermöglichen es dem Roboter, Lasten zu kontrollieren, präzise Manipulationen durchzuführen und sich an Veränderungen in der Umgebung anzupassen. Im Folgenden werden die wichtigsten Sensoren und Motoren und ihre Eigenschaften vorgestellt, die für eine optimale Roboterleistung erforderlich sind. Für den Zusammenbau des Mechanismus sind drei große Motoren erforderlich. Zwei davon steuern die Vorwärts- und Rückwärtsbewegung sowie die Drehung des Roboters, während der dritte Motor für die

Funktion der Klaue verantwortlich ist. Diese Kombination sorgt für präzise Manövrierbarkeit und hohe Anpassungsfähigkeit, so dass der Roboter seine Aufgaben effizient erfüllen kann. Nachstehend finden Sie ein Blockdiagramm (Abbildung 2 der Roboterlogik.

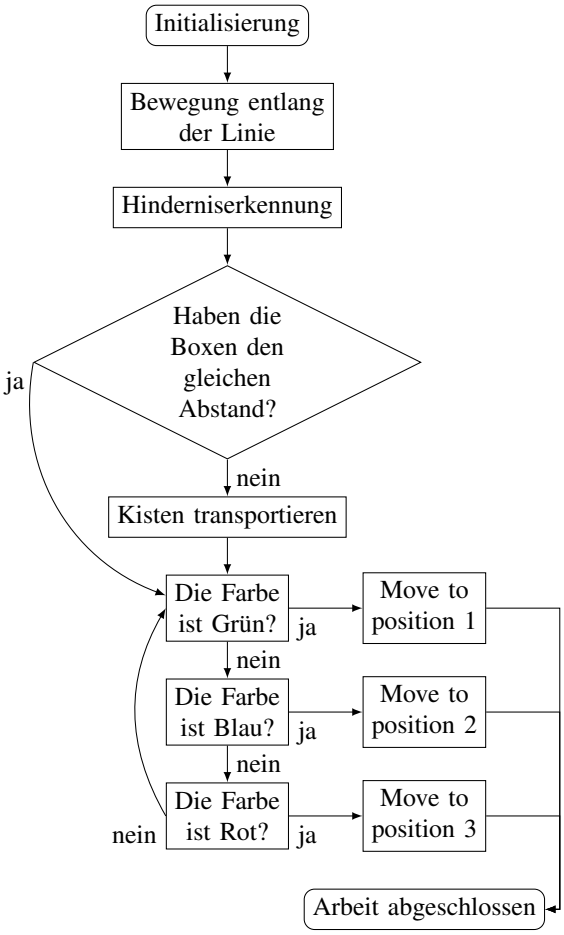


Abbildung 2. Blockdiagramm des Roboterbetriebs

B. Motoren

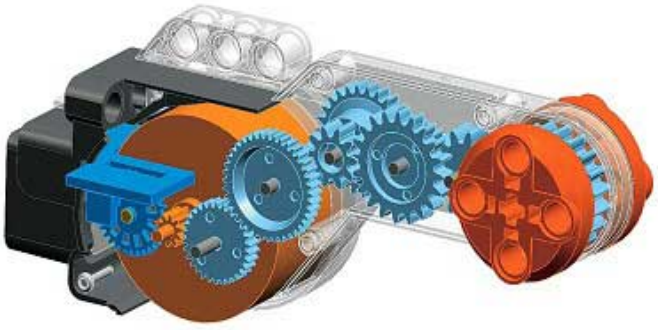


Abbildung 3. großer Motor

Für den Zusammenbau des Mechanismus sind drei große Motoren (Abbildung 3) erforderlich. Wie in [4] dargestellt, zeigt das Bild die interne Struktur des NXT-Motors. Zwei davon steuern die Vorwärts- und Rückwärtsbewegung sowie die Drehung des Roboters, während der dritte Motor für die Funktion der Klaue verantwortlich ist. Diese Kombination sorgt für präzise Manövrierbarkeit und hohe Anpassungsfähigkeit, so dass der Roboter seine Aufgaben effizient erfüllen kann.

C. Lichtsensoren

Abbildung 4. Lichtsensor

Zwei Lichtsensoren (Abbildung 4) sind symmetrisch am unteren Teil des Roboters angebracht, um seine Bewegung präzise zu steuern. Ihre Hauptfunktion besteht darin, eine schwarze Linie zu erkennen, die dazu beiträgt, den Roboter auf einer vorgegebenen Bahn zu halten. Wenn einer der Sensoren eine schwarze Linie erkennt, passt der Roboter automatisch seine Richtung an, um auf dem Kurs zu bleiben. Dieses Kontrollsystem gewährleistet eine stabile Navigation und verhindert Abweichungen von der Route.

D. Ultraschallsensor Sensor

Abbildung 5. Ultraschallsensor

Moderne Roboter erkennen nicht nur Objekte, sondern interagieren auch gekonnt mit ihnen. Ein Ultraschallsensor (Abbildung 5) wird eingesetzt, um den Abstand zu den Kisten zu bestimmen und sie genau zu positionieren. Er misst zunächst den Abstand zur ersten Kiste, dann zur zweiten, analysiert die Daten und entscheidet, ob eine Anpassung notwendig ist. Wenn die Abstände nicht übereinstimmen, aktiviert der Roboter eine Klaue, die die zweite Kiste sanft an den richtigen Platz bewegt. Dank dieses Verfahrens werden die Objekte mit hoher Genauigkeit platziert, was die Effizienz der automatisierten Frachtbewegung deutlich erhöht. Diese Technologie findet breite Anwendung in der Logistik und in Produktionslinien, in denen Schnelligkeit und Präzision gefragt sind. Darüber hinaus minimiert dieses System das menschliche Eingreifen, was die Fehlerwahrscheinlichkeit verringert und die Arbeitsabläufe optimiert.

E. Programmierung

Der Roboter basiert auf dem NXT-Controller; er ermöglicht es dem Roboter, alle oben genannten Geräte in einem System zu kombinieren. Zur Programmierung habe ich eine Bibliothek mit Funktionen für den Roboter verwendet, die RWTH Mindstorms NXT Toolbox. Mit seiner Hilfe konnte ich viele Funktionen für den Roboter erstellen, um verschiedene Aufgaben auszuführen, und diese zu einem vollständigen Skript kombinieren, um mit der zugewiesenen Aufgabe zu arbeiten. Unten können Sie den Roboterlader auf dem Arbeitsfeld beobachten (Abbildung 6)

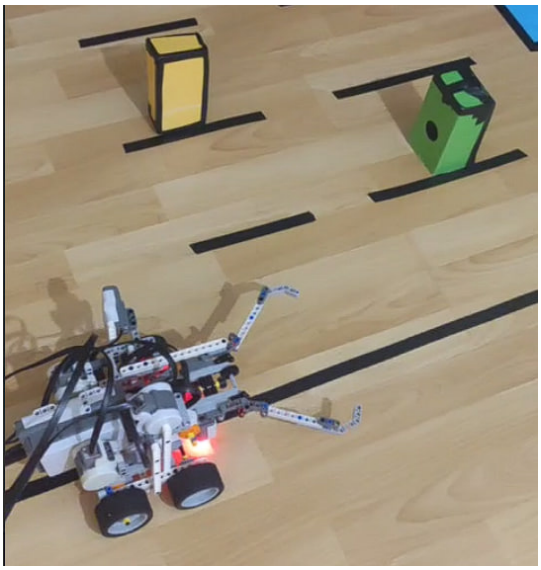


Abbildung 6. Roboterlader verrichtet Arbeit

Im Folgenden werden einige Funktionen vorgestellt, die zur Vermeidung von Wiederholungen im Code entwickelt wurden.

```
function moveDirekt(motorLeft,motorRight,speed)
    motorLeft.Power = -speed;
    motorRight.Power = -speed;
    motorLeft.SendToNXT();
    motorRight.SendToNXT();
end
```

Abbildung 7. moveDirekt Function

Diese Funktion (Abbildung 7) fährt die Räder des Roboters mit der eingestellten Geschwindigkeit vorwärts.

```
function correctLeft(motorLeft, motorRight)
    motorLeft.Power = 0;
    motorRight.Power = -10;
    motorLeft.SendToNXT();
    motorRight.SendToNXT();
    fprintf('CorrectLeft');
    pause(0.5);
    moveForward(motorLeft, motorRight, 20);
end
```

Abbildung 8. correctLeft Function

Diese Funktion (Abbildung 8) dient dazu, den Roboter nach links auszurichten, wenn der linke Sensor die schwarze Linie berührt. Während seiner Ausführung korrigiert das rechte Rad die Position des Roboters relativ zur Linie.

```
function correctRight(motorLeft, motorRight)
    motorRight.Power = 0;
    motorLeft.Power = -10;
    motorLeft.SendToNXT();
    motorRight.SendToNXT();
    fprintf('CorrectRight');
    pause(0.5);
    moveForward(motorLeft, motorRight, 20);
end
```

Abbildung 9. correctRight Function

Die Spiegelfunktion (Abbildung 9) , die ausgeführt wird, wenn der rechte Sensor die schwarze Linie trifft.

```
COM_CloseNXT all;
h = COM_OpenNXT();
COM_SetDefaultNXT(h);

motorLeft = NXTMotor('C');
motorRight = NXTMotor('B');

OpenUltrasonic(SENSOR_1);
OpenLight(SENSOR_4, 'ACTIVE');
OpenLight(SENSOR_3, 'ACTIVE');

while true

    lightValueRight = GetLight(SENSOR_3);
    lightValueLeft = GetLight(SENSOR_4);

    if lightValueLeft < 450 && lightValueRight >= 450
        fprintf('Left\n');
        correctLeft(motorLeft, motorRight);
    elseif lightValueLeft >= 450 && lightValueRight <
         450
        fprintf('Right\n');
        correctRight(motorLeft, motorRight);
    elseif lightValueLeft <= 500 && lightValueRight <=
         500
        moveForward(motorLeft, motorRight, 20

        pause(1);
        stopMotors(motorLeft, motorRight);
        pause(2);
```

```
        if (lightValueLeft > 530 && lightValueLeft <
             630) || (lightValueRight > 530 &&
             lightValueRight < 650) % Red
            fprintf('Red\n');
                moveForward(motorLeft, motorRight, -20)
                     ;

            pause(13);
            stopMotors(motorLeft, motorRight);



        end
        if (lightValueLeft > 350 && lightValueLeft <
             430) || (lightValueRight > 350 &&
             lightValueRight < 430)% blau
            fprintf('Blue\n');
                moveForward(motorLeft, motorRight, -20)
                     ;

            pause(10.5);
            stopMotors(motorLeft, motorRight);
        end
        if (lightValueRight > 470 && lightValueRight <
             500) || (lightValueLeft > 470 &&
             lightValueLeft < 500) % green
            fprintf('green\n');
                moveForward(motorLeft, motorRight, -20)
                     ;
            pause(8.3);
            stopMotors(motorLeft, motorRight);
        end

        CloseSensor(SENSOR_1);
        CloseSensor(SENSOR_3);
        CloseSensor(SENSOR_4);

        COM_CloseNXT(h);

        error('Program␣finished.');
    else
        fprintf('Forward\n');
        moveForward(motorLeft, motorRight, 20);
    end
end
end
```

Abbildung 10. moveLinie Script

Ein Codeblock (Abbildung 10) aus dem vollständigen Roboterskript, der es dem Roboter ermöglicht, entlang der schwarzen Linie zu fahren sowie die Farbe und das entsprechende Verhalten des Roboters zu bestimmen.

IV. Probleme

Während der Entwicklung des Roboters traten verschiedene Probleme auf. Eines der interessantesten war die Verwendung eines EV3-Lichtsensors im NXT, was zu falschen Messwerten führte. Dieses Problem konnte jedoch leicht gelöst werden, indem man einen für den NXT geeigneten Sensor verwendete und diesen ersetzte.

Sobald der Roboter in der Lage war, der schwarzen Linie zu folgen, stand er vor der Aufgabe, zwei Kisten zu zählen, ohne unnötige Werte in Variablen einzugeben. Zunächst wurde die Verwendung einer Identifikationsmarkierung in Form einer Flagge in Betracht gezogen, die den Abstand angibt, in dem der Codeblock ausgeführt werden sollte, um die Position der Kisten zu fixieren. Es stellte sich jedoch heraus, dass der Roboter dieselbe Box mehrmals erkennt, so dass diese Methode ungeeignet war. Stattdessen wurde das Problem dadurch gelöst, dass die Abstandserkennung nur aktiviert wurde, wenn der gemessene Wert unter einem bestimmten Schwellenwert lag. Die Messung wurde dann so lange gestoppt, bis der Roboter den Raum zwischen den Kisten durchquerte und erneut einen Abstand unterhalb des festgelegten Schwellenwerts erkannte.

Probleme gab es auch mit den Farbsensoren, da die verwendeten Sensoren den Farberkennungsmodus nicht unterstützten. Um die Farbe zu bestimmen, wurde der Bereich der Lichtreflexion verschiedener Farben analysiert und zur Identifizierung der Farben verwendet.

V. Ergebnisdiskussion

Der Roboterlader hat die Aufgabe erfolgreich abgeschlossen. Die Sensoren ermittelten Farbe und Entfernung zu den Kisten korrekt und die Motoren bewegten die Ladung reibungslos und stellten sie im dafür vorgesehenen Bereich ab. Der Algorithmus arbeitete exakt gemäß der angegebenen Befehlsfolge und gewährleistete so die Zuverlässigkeit und Effizienz des Betriebs. Das System ist bereit für die Integration in reale Logistikprozesse. Das Projektziel wurde erfolgreich erreicht.

Literaturverzeichnis

[1] Habr: *Artikel über Roboterlader in Amazon-Lagern während der Covid-19-Pandemie*. https://habr.com/ru/companies/pochtoy/articles/493094/. Version: 2020. – Zugriff am 13. März 2025

[2] Wikipedia: *Amazon Robotics*. https://en.wikipedia.org/wiki/Amazon_Robotics. Version: 2025. – Zugriff am 13. März 2025

[3] MathWorks: *MATLAB zur Robotersteuerung – Dokumentation zur Arbeit mit NXT über MATLAB, Einrichten von Sensoren und Motoren*. https://www.mathworks.com/solutions/robotics.html. Version: n.d.. – Zugriff am 13. März 2025

[4] Philo: *NXT Motor Analysis – Bildquelle*. https://www.philohome.com/nxtmotor/nxtmotor.htm. Version: n.d.. – Zugriff am 13. März 2025