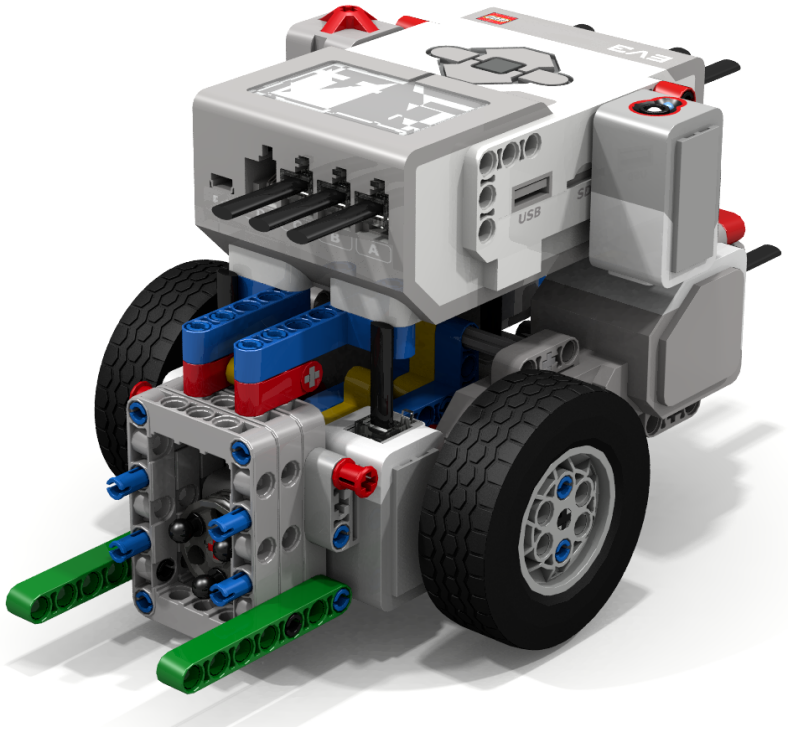
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

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 und Jörg Petzold

Band 7 vom Wintersemester 2023/2024

Inhaltsverzeichnis

Impressum

Herausgeber:	Mathias Magdowski, Thomas Schallschmidt und Thomas Gerlach 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-2024-037
ISSN:	2629-6160
Redaktionsschluss:	May 2024
Seminarzeitraum:	29. Januar – 12. Februar 2023
Bezug:	Open Access, Digitale Hochschulbibliothek Sachsen-Anhalt `http://edoc2.bibliothek.uni-halle.de/`

1. Auflage, Magdeburg, Otto-von-Guericke-Universität, 2024

Erstellung des Sammelbandes mittels LaTeX, `hyperref` und pdfpages

Ballschussmaschine

Kjell Diegeler, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Der in diesem Seminar entwickelte Roboter ist ein fast vollautomatisierter fahrender Ballschussroboter. Die Umsetzung der Konstruktions- und Softwarekonzepte werden im Folgenden erläutert, wie auch die entstandenen Herausforderungen und deren Lösungen. Zum Schluss werden die Verwendungszwecke und mögliche Optimierungen diskutiert.**

***Schlagwörter*—Automatisierung, Ballschuss, LEGO-Mindstorms, Programmierung, Roboter, Sensorik**

I. Einleitung

WÄHREND des Praktikums war es die Aufgabe, einen Roboter zu entwickeln, neben der Hauptaufgabe die Grundlagen in MATLAB zu erlernen und die Sensoren und Motoren von LEGO kennenzulernen. Für die Entwicklung des Roboters ist eine gute räumliche Vorstellung von großem Vorteil.
Stabilität und Balance spielen eine große Rolle beim Entwickeln des Roboters. Bei fehlender Balance besteht die große Gefahr des Umkippens und starke Einschränkungen in der Funktion. Fehlende Stabilität ist allerdings das größere Problem, sollte dies nicht gegeben sein, können sich relevante Bauteile lösen und die Funktion des Roboters nicht gewährleisten. Die Ansteuerung der Sensoren und Motoren kann auch eine kleine Herausforderung darstellen. Die Verwendung eines LEGO-fremden Bauteils, wie zum Beispiel einer Webcam, kann schnell zu einer zeitaufwendigen Herausforderung werden.

II. Vorbetrachtungen

In diesem Abschnitt wird die Konzeptidee des Roboters erläutert, um auf Hinblick der Umsetzung Verständnis zu schaffen. Die Funktionen und benötigten Komponenten werden hierbei genauer vorgestellt.

A. Was soll die Ballschussmaschine können?

Das Konzept des Ballschussroboters ist es, nachdem er gestartet wurde, soll der Roboter warten, bis der Ballschussapparat geladen ist. Daraufhin soll er sich fortbewegen und bei einem vordefinierten Abstand vor dem Ziel halten. Als Nächstes sucht die Maschine den Körper, der als Ziel dient und wenn dieser nicht mittig zum Roboter ausgerichtet ist, richtet sich der Roboter selbständig aus. Ist dies geschehen, erfolgt der Schuss und das Ziel wird getroffen.

B. Die benötigten Komponenten für die Ballschussmaschine

Zur Umsetzung wurde ein LEGO-Mindstorms-Set und eine Webcam zur Verfügung gestellt. Das Set beinhaltet eine verschiedene Anzahl an Motoren, Sensoren und LEGO-Klemmbausteinen, die für die Umsetzung essentiell wichtig sind. Die benötigten Komponenten werden in Abbildung 1 dargestellt. Als Steuereinheit dient der NXT-Stein, welcher sämtliche Motoren und Sensoren steuert und wichtige Daten entnimmt. Der NXT-Baustein ist zu dem auch die Schnittstelle zum PC bzw. Laptop, wo das Programm abläuft. Die Motoren dienen zur Fortbewegung und zur automatischen Ausrichtung des Roboters, als auch zur Steuerung des Sicherungsstiftes des Ballschussapparates. Der Taster wurde dazu eingesetzt, um dem Motor, der den Sicherungsstift steuert, zu signalisieren, wann der Stift ausgefahren werden soll. Der Ultraschallsensor wurde zur Distanzmessung verwendet. Zum Schluss wurde eine Webcam zur Farberkennung verwendet und somit zur Zielsuche.

Abbildung 1. Verwendete Komponenten [1]–[5]

III. Umsetzung

Im Folgenden wird die Umsetzung Schritt für Schritt erläutert. Zuerst wird der Programmablaufplan dargestellt. Danach wird Aufbau und Funktion der Konstruktion erklärt. Zum Schluss wird noch die Funktionsweise der Webcam erläutert.

A. Programmablaufplan

In der Abbildung 2 wird für die Ballschussmaschine der Programmablaufplan dargestellt.

DOI: 10.24352/UB.OVGU-2024-003

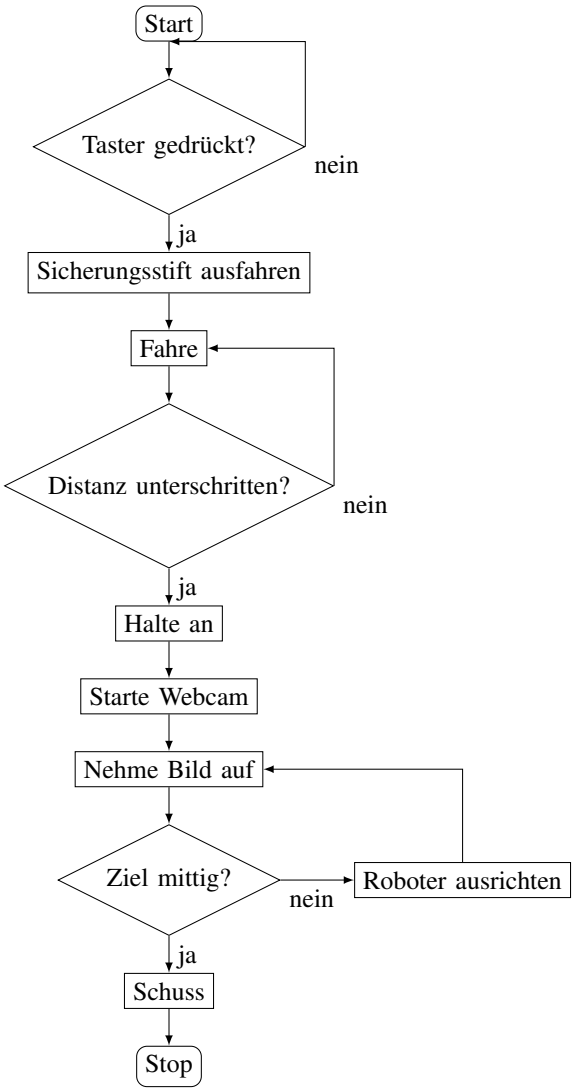


Abbildung 2. Beispielhafter Programmablaufplan zur Erklärung eines Verfolgungsalgorithmus

B. Aufbau und Funktion der Ballschussmaschine

Der Roboter in Abbildung 3 besteht aus einem Fahrwerk, Ballschussapparat, einem Sicherungsstift und einer Webcam. Das Fahrwerk hat die Funktion, den Roboter zu fahren und später den Roboter auszurichten. Hierzu werden zwei von den drei verwendeten Motoren genutzt. Der Ballschussapparat besteht aus einer Schiene, einem Gummi und einer Nachladevorrichtung. Der Sicherungsstift, abgebildet in Abbildung 4, dient hierzu, um die Nachladevorrichtung an seinem Ort zu halten. Dieser wird gesteuert durch einen Tastsensor und einen weiteren Motor. Durch Betätigen des Schalters wird der Motor angesteuert und dieser bewegt sich dann um 180°. Ist der Sicherungsstift ausgefahren, macht das Programm eine Pause und gibt dann den Motoren vom Fahrwerk den Befehl zu starten. Der Ultraschallsensor sorgt ab dann, dass die Ballschussmaschine bei einer vordefinierten Distanz anhält. Dann wird die Webcam gestartet. Die ist dazu da, um ein Bild aufzunehmen und dabei die Farbe Rot herauszufiltern und als Bild auszugeben. Mit den erhaltenen Daten wird nun die Position des Mittelpunktes des Zieles ermittelt und mit dem Mittelpunkt des Bildes abgeglichen. Sollte es zu einer Abweichung kommen, so wird über die Motoren des Fahrwerkes die Position in der jeweiligen Richtung abgeändert und erneut die Bedingungen gecheckt. Wenn die Bedingung erfüllt ist, wird jetzt der Motor des Sicherungsstiftes angesteuert. Dieser dreht sich um 180° und sorgt dafür, dass die Nachladevorrichtung durch das Gummi nach vorne schnallt und somit die Kugel abfeuert. Damit wäre das Programm beendet.

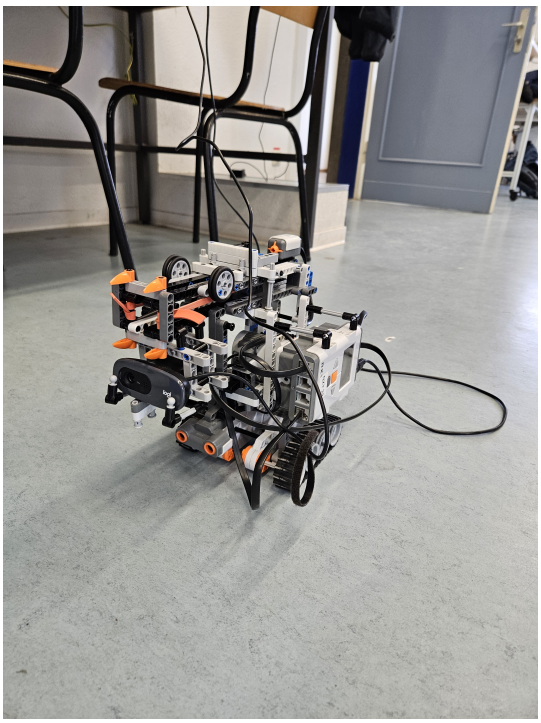


Abbildung 3. fertige Ballschussmaschine

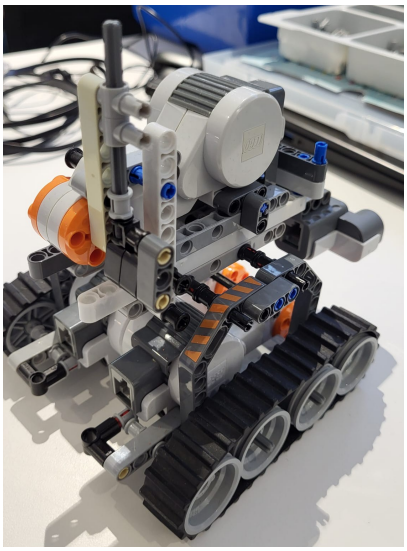


Abbildung 4. Sicherungsstift

C. Benutzung einer Webcam

Die Benutzung einer Webcam stellte sich als kleine Hürde heraus. Der Quellcode wird unten dargestellt. Im ersten Schritt wird die Webcam als Objekt angelegt. Mit dem Befehl `snapshot` wird eine Einmalaufnahme der Webcam gemacht und die Farbwerte in einer 480×640×3 großen Matrix gespeichert. Mit diesen Werten kann man sich auch das Bild ausgeben lassen, mit Skalierung. Hierfür wird der Befehl `imagesc()` verwendet. Als nächsten relevanten Schritt wird eine 0-Maske über die Farbwerte erstellt und mithilfe eines Filters überall, wo die Farbwerte mit der Filterbedingung überschrieben, mit der Zahl 1. Dies kann man sich ebenfalls als Bild ausgeben lassen. Als Nächstes wird über den Suchbefehl `find()` die Koordinaten der ersten und letzten 1 ermittelt. Somit lässt sich dann über die Gleichung 1 die Koordinaten des Mittelpunktes des Zielobjektes ermitteln.

```
cam = webcam;
preview(cam)
img = snapshot(cam);
imagesc(img)
mask = zeros(size(img,1,2));
mask(img(:,:,1)>75 & img(:,:,2)<50 & img(:,:,3)<50) = 1;
figure(2)
imagesc(mask)
[r_a,c_a] = find(mask==1,1,'first');
[r_e,c_e] = find(mask==1,1,'last');
m = c_a + (c_e - c_a)/2
```

$$m = c_a + (c_e - c_a)/2 \tag{1}$$

IV. Diskussion

Natürlich sind während der Umsetzung auch Probleme aufgetreten. Zum einen war die größte Schwierigkeit, die Konstruktion stabil zu gestalten, da das verwendete Gummi sehr viel Spannung erzeugte. So kam es des Öfteren vor, dass bei den ersten Testversuchen die Konstruktion mit zerschoss und man gezwungen war, die Maschine immer stabiler zu gestalten. Dies führte allerdings auch dazu, dass die ganze Konstruktion sehr einseitig belastet war, sodass die Gefahr des Umkippens sehr hoch war.

Dies wurde gelöst, indem man den NXT-Stein seitlich befestigt hat. Desweiteren kam die Frage auf, wie schnell man die Motoren ansprechen kann, ohne Fehler zu generieren, dies konnte nur durch Testen gelöst werden. Die Fixierung des Sicherungsstiftes beim Einfahren gestaltete sich anfangs auch als schwierig, konnte aber durch bauliche Änderungen gesichert werden. Durch die Bauweise sieht die Konstruktion sehr militärisch aus, dies ist natürlich nicht beabsichtigt und kommt zustande, da es sich um ein Kettenfahrzeug handelt. Dies war leider die einfachste Lösung, um den Roboter fahren zu lassen und auch zu drehen.

V. Zusammenfassung und Fazit

Am Ende konnte der Roboter, nachdem er geladen war, vollkommen autonom fahren, sein Ziel suchen und das Ziel mithilfe einer Kugel abschießen. Die Ballschussmaschine ist noch optimierbar. Zum einen könnte man noch die Konstruktion so abändern, dass die Maschine auch sich automatisch nach oben bzw. nach unten ausrichtet. Im Großen und Ganzen wurde das Projektseminar erfolgreich abgeschlossen und die wichtigen Grundlagen der Programmierung mit Matlab wurden mitgenommen.

Literaturverzeichnis

[1] Brickipedia: *NXT-Servomotor*. https://brickipedia.fandom.com/wiki/9842_NXT_Servo_Motor. Version: Februar 2024

[2] Brick-Shop: *NXT-Stein*. https://www.brick-shop.de/Elektrik-4951.html?language=de. Version: Februar 2024

[3] Amazon: *Ultraschallsensor*. https://www.amazon.de/LEGO-NXT-Ultraschallsensor-Ultrasonic-sensor/dp/B000PM8I8O. Version: Februar 2024

[4] LEGO: *Tastsensor*. https://www.lego.com/cdn/cs/set/assets/bltee56bc5628e9b8bf/9843.jpg. Version: Februar 2024

[5] Amazon: *Logitech C270 Webcam*. https://www.amazon.de/Logitech-C270-Webcam-720p-schwarz/dp/B01BGBJ8Y0. Version: Februar 2024

Rund trifft Eckig - Die Ballschussmaschine

Jeremy Fricke, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Durch den anfänglichen Erwerb der theoretischen Grundlagen in der Programmierumgebung MATLAB, war es anschließend möglich einen Roboter, bestehend aus LEGO-Klemmbausteinen, im Rahmen der Aufgabe des Projektseminars Elektrotechnik/Informationstechnik zu konstruieren. Der dafür benötigte Programmcode wurde in MATLAB geschrieben, um mittels des NXTs die verbauten Sensoren und Motoren anzusteuern und auszulesen. Der im Laufe dieses Seminars entwurfene Roboter ist ein fast vollautomatisierter fahrender Ballschussroboter. Anknüpfend wird sich mit den Konstruktions- und Softwarekonzepten auseinandergesetzt und im Zuge dessen die aufgetretenen Probleme, sowie deren Lösungen, geschildert. Abschließend wird auf Verwendungszwecke und mögliche Optimierungen eingegangen.**

***Schlagwörter*—Automatisierung, Ballschuss, LEGO-Mindstorms, Programmierung, Roboter, Sensorik**

I. Einleitung

Im Rahmen des LEGO-Mindstorms war abschließend zum Kennenlernen von MATLAB und LEGO-Sensoren und -Motoren die Aufgabe einen eigenen Roboter zu verwirklichen. Notwendig sind dabei grundlegende Kenntnisse in MATLAB und ein Verständnis für die Funktionsweise der LEGO-Sensoren und -Motoren. Für die Konstruktion des Roboters ist außerdem gutes räumliches Denken von Vorteil, aber nicht zwingend nötig.
Entscheidende Schwerpunkte beim Entwickeln eines Roboters aus LEGO sind beispielweise die Stabilität und Balance der Konstruktion. Bei fehlender Balance kann die Konstruktion leicht umkippen. Wenn die Stabilität nicht ausreichen sollte, können sich je nach Funktionsweise des Roboters tragende Bauteile lösen. In Hinsicht auf die Programmierung kann die Ansteuerung und Justierung von Sensoren und Motoren herausfordernd sein. Bei Verwendung eines nicht für LEGO vorgesehenes Bauteil, wie einer Webcam, kann auch das Probleme hervorbringen. Die Kombination von LEGO-Bauteilen und nicht LEGO-Bauteilen ist ein sehr zeitaufwendiges Hindernis.

II. Vorbetrachtungen

Folgender Abschnitt wird die Idee hinter dem entwickelten Roboter näherbringen. Funktionsweise und verwendete Bauteile werden parallel erklärt.

A. Was soll die Ballschussmaschine können?

Funktionstechnisch soll die Ballschussmaschine nach Starten des Programmcodes warten bis die Schussvorrichtung gespannt ist. Ist diese Bedingung erfüllt, fährt der Roboter vorwärts bis zu einem, im Voraus bestimmten, Abstand zu dem Ziel. Das Ziel wird mittels Farberkenung einer Webcam lokalisiert.

DOI: 10.24352/UB.OVGU-2024-004

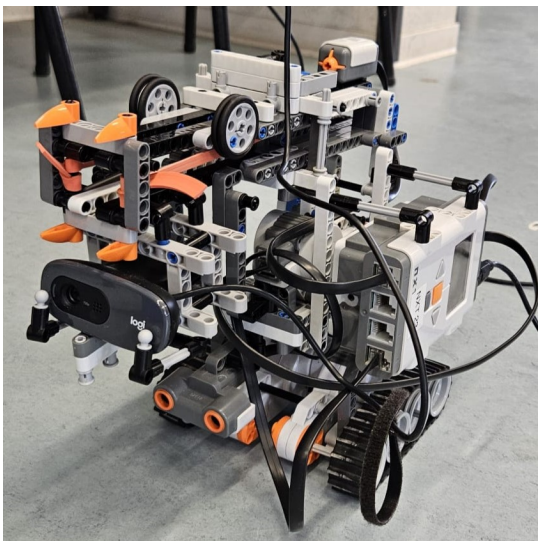


Abbildung 1. Fertige Ballschussmaschine

Sofern das Ziel nicht mittig zum Roboter ausgerichtet ist, korrigiert dieser eigenständig seine Position bis das Ziel mittig vor ihm steht. Nach Abschluss der Positionskorrektur erfolgt der Schuss und das Ziel wird im besten Fall getroffen.

B. Die benötigten Komponenten für die Ballschussmaschine

Zur Verfügung standen ein LEGO-Mindstorms-Set und eine Webcam. Das Set umfasst mehrere Motoren, verschiedenste Sensoren und LEGO-Klemmbausteine, die das Grundgerüst des Roboters darstellen. Der NXT-Stein dient als Schnittstelle zwischen Motoren und Sensoren und dem Programm auf dem Computer. Er ermöglicht die Ansteuerung der Motoren und Sensoren, sowie das Lesen, der von den Sensoren gemessenen Werte.
Die Fortbewegung und Positionskorrektur erfolgen über die Motoren. Außerdem wird auch die Schussvorrichtung über einen Motor gesteuert. Ein Tastsensor wird beim Spannen der Vorrichtung betätigt, wodurch dem Motor signalisiert wird wann er den Sicherungsstift nach oben fahren soll. Die Distanzmessung gelingt durch einen Ultraschallsensor. Die Webcam übernimmt die Farberkennung und die damit einhergehende Zielsuche.

Abbildung 2. Verwendete Komponenten [1]–[5]

III. Umsetzung

Der anschließende Abschnitt verdeutlicht die Realisierung des Roboters Schritt für Schritt.

A. Programmablaufplan

Der Programmablaufplan für die Ballschussmaschine sieht wie folgt aus:

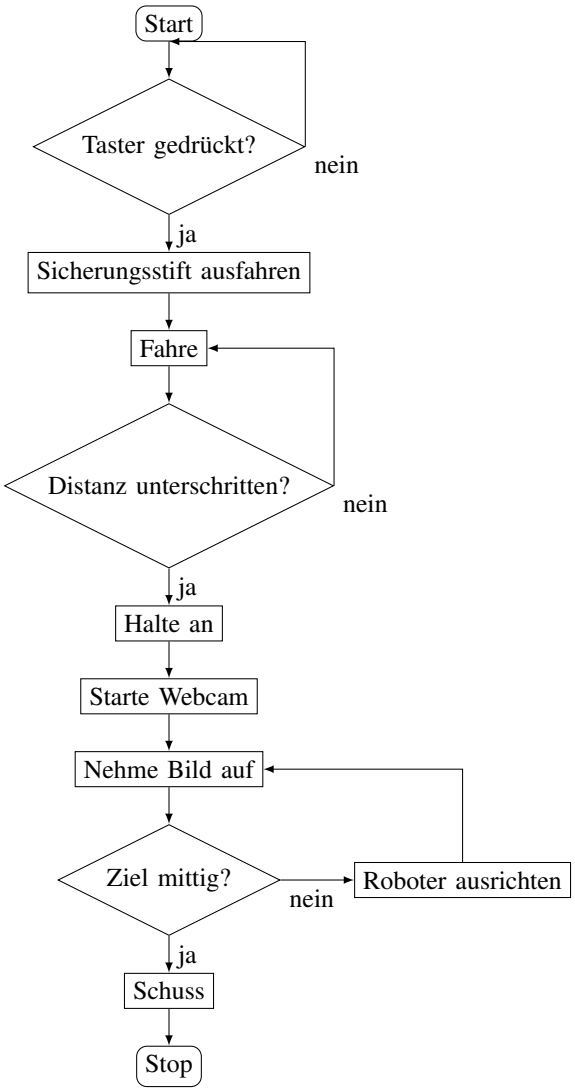


Abbildung 3. Programmablauf der Ballschussmaschine

B. Aufbau und Funktion der Ballschussmaschine

Der in Abbildung 1 Roboter setzt sich aus einem Kettenfahrwerk, einer Schussvorrichtung, dem dazugehörigen Sicherungsstift und einer Webcam zusammen. Das Kettenfahrwerk sorgt für die Fortbewegung und die Positionskorrektur beim Zielen. Zwei von drei Motoren werden hierfür verwendet, wobei jeweils eine Seite von einem Motor bewegt wird. Die Schussvorrichtung umfasst eine Schiene als Ballführung, ein Gummiband und eine Nachladevorrichtung. Der Sicherungsstift aus Abbildung 4 erfüllt diese Funktion.

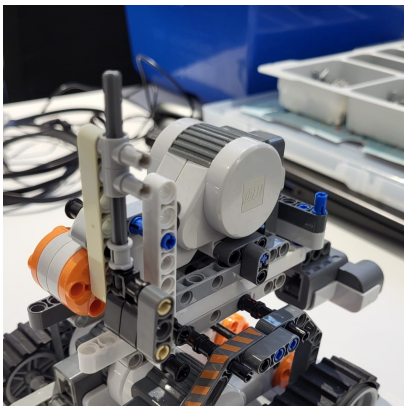


Abbildung 4. Sicherungsstift von Nahem

hält die Nachladevorrichtung an ihrem Platz, wenn er durch den Motor nach oben gefahren wird. Wird der Tastsensor beim Nachladen gedrückt, ist das für den Motor das Signal um den Sicherungsstift nach oben zu fahren. Dafür dreht sich der Motor einmal um 180°.
Der ausgefahrene Sicherungsstift gibt dem Programm das Zeichen, die Motoren des Kettenfahrwerks anzusteuern, um loszufahren. Der Ultraschallsensor achtet auf die Distanz zu dem vor der Ballschussmaschine liegendem Ziel und hält diese an, wenn nur noch ein bestimmter Abstand vorhanden ist. Ist der Roboter stehengeblieben, schaltet sich die Webcam ein. Diese nimmt ein Bild auf und soll mittels Farberkennung die Farbe rot rausfiltern u nd d as e ntstandene B ild ausgeben. Die somit erhaltenen Daten können zur Bestimmung des Mittelpunktes des roten Zieles genutzt werden. Das wird dann mit dem Mittelpunkt des eigentlichen Bildes verglichen. Bei einer Abweichung der Mittelpunkte werden die Motoren des Kettenfahrwerks angesteuert um die Position des Roboters zu korrigieren. Dieser Vorgang wird solange wiederholt bis die Mittelpunkte übereinstimmen. Bei Übereinstimmung wird der Sicherungsstift mit Hilfe des Motors nach unten gezogen. Dadurch schnallt die Nachladevorrichtung mittels des Gummibandes nach vorn und die Kugel wird abgefeuert. Danach ist das Programm beendet.

C. Benutzung einer Webcam

Die Webcam war eine vergleichsweise kleine Hürde. Abbildung 5 zeigt den verwendeten Code. Zuerst wird die Webcam als Objekt angelegt. Der Befehl snapshot macht eine einmalige Aufnahme der Webcam und speichert die Farbwerte in einer 480×640×3-großen Matrix. Mit imagesc() kann das Bild durch

diese Werte mit Skalierung ausgegeben werden. Danach wird eine 0-Maske über die Farbwerte erstellt und mittels Filter überall da wo die Farbwerte mit der Filterbedingung übereinstimmen mit der Zahl eins überschrieben. Auch dieses Bild kann ausgegeben werden. Mit Hilfe des Suchbefehls find() lassen sich die Koordinaten der ersten und letzten einsen ermitteln. Damit können durch die Gleichung 1 die Koordinaten des Mittelpunktes des Zieles bestimmt werden.

$$midpoint_x = col_a + \frac{col_e - col_a}{2} \tag{1}$$

```
cam = webcam;
preview(cam)
img = snapshot(cam);
imagesc(img)
mask = zeros(size(img,1,2));
mask(img(:,:,2)<50 & img(:,:,3)<50) = 1;
figure(2)
imagesc(mask)
[row_a, col_a] = find(mask==1,1,'first');
[row_e, col_e] = find(mask==1,1,'last');
midpoint_x = (col_e - col_a)/2
midpoint_y = (row_a - row_e)/2
```

Abbildung 5. Quellcode der Webcam

IV. Ergebnisdiskussion

Die größte Schwierigkeit lag in der Konstruktion des Roboters. Er musste stabil und doch beweglich sein. Die Schussvorrichtung durfte nicht nach dem Schuss kaputt gehen und die Nachladevorrichtung musste weiterhin das Gummiband spannen können. Diese Anforderungen bedingten viele extra Bauteile in diesem Bereich. Die Lage des Motors für den Sicherungsstift führt zudem noch zu einer leichten Rechtslage der Schussvorrichtung. Um das Ungleichgewicht zu beheben, wurde das NXT, das im Vergleich ein ziemlich hohes Gewicht hat, an der linken Seite befestigt.
Ebenso schwierig war den Sicherungsstift zu fixieren. Anfangs ist dieser aufgrund der Spannung des Gummibandes nicht mit den Motor heruntergefahren. Durch leichte Änderung der Konstruktion am Motor und austauschen des Stiftes gelang aber letztendlich die Fixierung.
Das Kettenfahrwerk kann etwas militärisch wirken, ist allerdings für die Ansteuerung durch nur zwei Motoren die beste Wahl. Es ermöglicht Geradeausfahren und Lenken, was bei einer anderen Bauweise komplizierter und platzaufwendiger geworden wäre.

V. Zusammenfassung und Fazit

Alles in Allem ist die Ballschussmaschine ein funktionierender Roboter. Er erfüllt alle im Voraus gesetzten Ziele: eine automatische Positionskorrektur zum Ziel, fahrend, selbstständiges Nachladen und einen automatischen Schuss bei Farberkennung des Zieles.
Alle aufgetretenen Hürden wurden bestmöglich überwunden. Der Roboter ist stabil und ausbalanciert, die größten aufgekommenen Probleme.
In Zukunft wäre es möglich die Schussvorrichtung auch in der Höhe verstellbar zu machen. Im Moment wird das Zentrieren von dem Kettenfahrwerk übernommen und kann sich somit nur seitlich ausrichten. Durch Höhenverstellbarkeit könnte man eine bessere Zielgenauigkeit erlangen. Allerdings wäre die jetzige Konstruktion nicht für diese Verbesserung ausgelegt, da es andernfalls wieder instabil werden würde.

Literaturverzeichnis

[1] *Brickipedia, NXT-Servomotor*. https://brickipedia.fandom.com/wiki/9842_NXT_Servo_Motor
[2] *Brick-Shop, NXT-Stein*. https://www.brick-shop.de/Elektrik-4951.html?language=de
[3] *Amazon, Ultraschallsensor*. https://www.amazon.de/LEGO-NXT-Ultraschallsensor-Ultrasonic-sensor/dp/B000PM8I8O
[4] *LEGO, Tastsensor*. https://www.lego.com/cdn/cs/set/assets/bltee56bc5628e9b8bf/9843.jpg
[5] *Amazon, Logitech C270 Webcam*. https://www.amazon.de/Logitech-C270-Webcam-720p-schwarz/dp/B01BGBJ8Y0

Sudoku Solver

Zhu Ding, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Rahmen des diesjährigen LEGO Mindstorms Praktikumsprogramms wurde ein Roboter aus LEGO Baukästen gebaut, der mit NXT gesteuert und in MATLAB programmiert wurde und in der Lage war, eigenständig Sudoku-Probleme zu lösen. Der nächste Abschnitt beschreibt den Prozess der Entwicklung des Roboters.**

***Schlagwörter*—Schlagwörter—LEGO Mindstorms, MATLAB, Sudoku, Lichtsensor, Roboter, Projektseminar**

I. Einleitung

IM heutigen digitalen Zeitalter verändert sich die Welt in einem noch nie dagewesenen Tempo, da die Technologie immer weiter voranschreitet und der Einsatz von LEGO—Robotern immer mehr Beachtung und Aufmerksamkeit findet. Lego-Roboter sind nicht mehr nur Spielzeug. LEGO-Roboter können auch Ziele erreichen, die auf den Ideen der Menschen basieren. In diesem Artikel geht es darum, wie Sudoku-Probleme von LEGO-Robotern gelöst werden. Sudoku ist ein klassisches Logikrätsel, das nicht nur unser logisches Denkvermögen trainiert, sondern auch Geduld und Konzentration fördert. Menschen stoßen beim Lösen von Sudoku oft auf eine Menge Probleme, aber jetzt können diese Sudoku-Probleme durch die Entwicklung präziser Algorithmen für LEGO-Roboter gelöst werden, die diese Sudoku-Probleme lösen. Ein System, das Sudoku-Probleme automatisch löst, kann die Probleme, auf die der Spieler stößt, leicht lösen, was dem Spieler nicht nur hilft, Zeit zu sparen, sondern ihm auch ermöglicht, die Lösung des Problems aus einer anderen Perspektive zu entdecken. Deshalb wird in diesem Artikel ausführlich erklärt, wie man Sudoku-Aufgaben mit LEGO-Robotern lösen kann.

II. Vorbetrachtungen

Das Ziel dieses Projekts ist ein LEGO-Roboter, der Sudoku-Probleme löst. Zunächst muss man verstehen, wie man Sudoku-Probleme löst und in der Lage sein, sie in MATLAB Code darzustellen.

A. Das unverzichtbare MATLAB

Wenn man LEGO-Roboter programmieren will, muss man MATLAB beherrschen und benutzen. Das Erlernen einer Programmiersprache erfordert zunächst die Beherrschung der grundlegenden Syntax und Variablendeklarationen sowie der grundlegenden mathematischen Operationen. Darüber hinaus geht es vor allem um das Schreiben und Aufrufen von Funktionen. In diesem Projekt wird zum Beispiel der Motor mit MATLAB-Code gesteuert. Dazu gehören die Initialisierung der Verbindungen, die Angabe des Motoranschlusses und der Parameter sowie das Senden von Steuerbefehlen zum Starten, Stoppen oder Ändern der Bewegung des Motors. Anschließend wird der geschriebene Code in die NXT-Box hochgeladen und ausgeführt. Schließlich werden die erforderlichen Parameter angepasst und optimiert.

DOI: 10.24352/UB.OVGU-2024-005

B. Wie man Sudoku löst

Die Lösung eines Sudoku-Rätsels erfordert die Einhaltung einer Reihe grundlegender Schritte. Zunächst ist das Verständnis der Regeln des Sudoku-Spiels entscheidend, um sicherzustellen, dass jede Zahl in jeder Zeile, jeder Spalte und jedem 3×3-Untergitter (9×9-Netz) nur einmal vorkommt. Anschließend werden die bekannten Zahlen überprüft und logische Schlussfolgerungsmethoden wie die Methode der einzig möglichen Kandidaten und die Ausschlussmethode angewendet, um die leeren Zellen zu füllen. Während des Füllvorgangs wird kontinuierlich ausprobiert und überprüft, um sicherzustellen, dass die eingetragenen Zahlen nicht gegen die Regeln verstoßen oder zu Widersprüchen führen. Diese Schritte werden wiederholt, bis alle leeren Zellen korrekt ausgefüllt sind und das Sudoku-Rätsel gelöst ist.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Abbildung 1. Sudoku-Tabelle

III. Realisierung

A. Aufbau

Das ganze Projekt besteht aus einer NXT-Box, drei Motoren, einem Lichtsensor, einigen Teilen und einigen Antriebsrädern. Die Hauptstruktur besteht aus einem Fahrzeug, das sich hin- und herbewegen kann, und auf dem Fahrzeug befindet sich ein Ausleger, der sich von einer Seite zur anderen drehen kann, und dieser Ausleger kann angehoben und abgesenkt werden.

Abbildung 2 zeigt den anfänglichen Aufbau. Da ein wichtiges Teil fehlte Abbildung 3, wurde versucht, einige Teile als Ersatz selbst herzustellen, aber das Endergebnis ist, dass die gesamte Struktur sehr instabil ist!

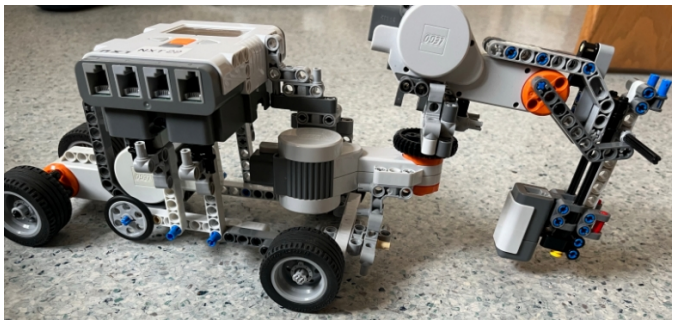


Abbildung 2. ursprünglicher Aufbau des Sudoku Solvers

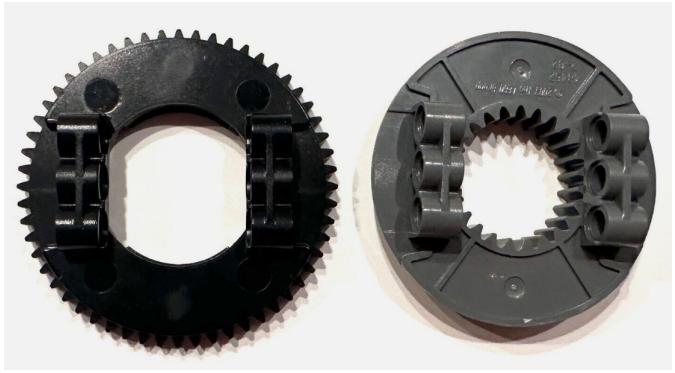


Abbildung 3. Zahnrad

Schließlich wurde dieses wichtige Zahnrad (Abbildung 3) erhalten und es begann, der Aufbau zu modifizieren. Der größte Nachteil der ursprünglichen Konstruktion war die strukturelle Instabilität. Erstens war das Auto darunter kleiner, dann waren die NXT-Box und der Motor schwerer, was dazu führte, dass sich die Kräfte auf das Auto in der Mitte konzentrierten, was es schwierig machte, das Gleichgewicht zu halten. Zweitens wird durch das Fehlen dieses Zahnrad (Abbildung 3) die Kraft auf den Ausleger unausgewogen, was dazu führt, dass der Ausleger als Ganzes nach unten kippt, was das Gleichgewicht des gesamten Roboters beeinträchtigt.

Abbildung 4 zeigt einen verbesserten Aufbau. Es wurde damit begonnen, die Unterseite des Autos zu modifizieren. Das Auto wurde vergrößert, um einen stabilen Boden für den gesamten Roboter zu haben, und es wurden die gegebenen Teile verwendet, um das gesamte Auto stabil zu machen, damit das Gewicht der NXT-Box und der Motoren getragen werden kann. Die unausgewogenen Kräfte auf dem Ausleger wurden auch gelöst, als das Zahnrad verwendet wurde, weil das Zahnrad die Struktur stabiler machte.

B. Durchführung

Steuern Sie zunächst den hinteren Motor, damit das Auto vorwärts fährt. Wenn der Lichtsensor den Sudoku-Tabelle betritt, halten Sie an und beginnen Sie mit dem Scannen. Steuern Sie den vorderen Motor so, dass er sich von links nach rechts dreht und den Ausleger mitbewegt. Jetzt kann der Lichtsensor den gesamten Bereich in der ersten Reihe von links nach rechts abtasten. Dann lässt man den hinteren

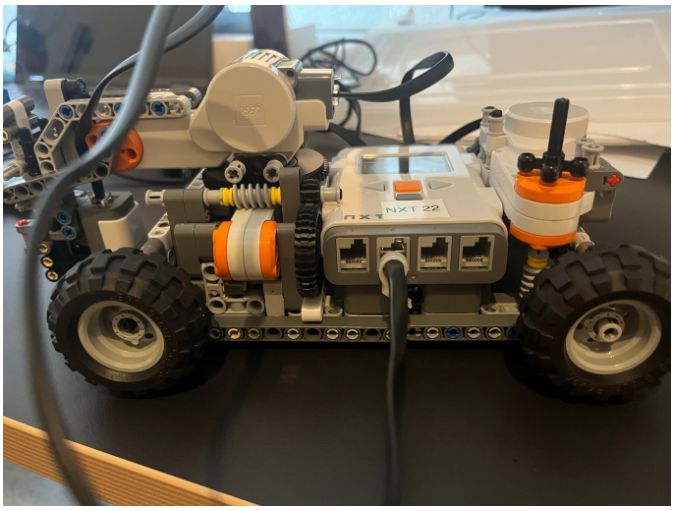


Abbildung 4. verbesserter Aufbau des Sudoku Solvers

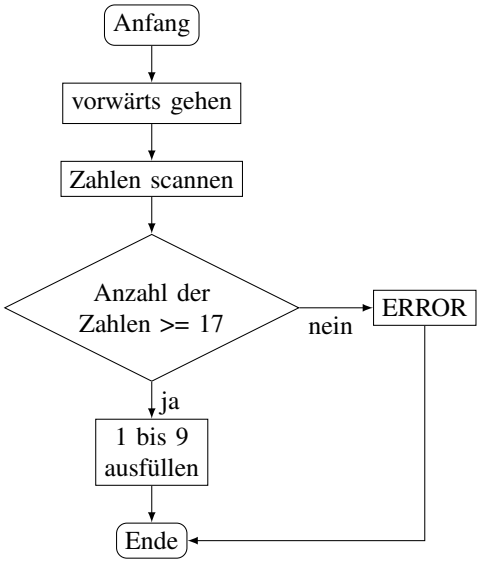


Abbildung 5. Ablaufplan

Motor das Auto ein Stück vorwärts zur zweiten Reihe fahren und wiederholt den vorherigen Vorgang, bis der Roboter alle Zahlen abgetastet hat. Normalerweise gibt es 17 Zahlen in der Sudoku-Tabelle, der Roboter sollte in der Lage sein, alle Zahlen zu scannen, um mit dem Lösen fortzufahren. Wenn die gescannten Zahlen weniger als 17 sind, wird nach dem Ende des Scanvorgangs ”ERROR” auf der NXT-Box angezeigt, um uns zu informieren.

C. Probleme

Oben wurden die 17 Zahlen genannt, die vom Lichtsensor abgetastet werden müssen, um die folgenden Schritte korrekt auszuführen. Eines der Probleme, auf die gestoßen wurde, war, dass der Lichtsensor nicht alle 17 Zahlen scannen konnte. Es wurde festgestellt, dass die Ursache des Problems darin lag, dass nicht genug Licht vorhanden war. Es wurden viele verschiedene Szenarien ausprobiert, und im Grunde genommen

konnte nur eine externe Lichtquelle dazu führen, dass 3 bis 5 Zahlen gescannt wurden. Beim Versuch, eine externe Lichtquelle hinzuzufügen, konnte je nach Lichtquelle der Lichtsensor etwa 10 Zahlen oder mehr scannen. Es wurde versucht, eine Handy-Taschenlampe oder mehrere Handy-Taschenlampen oder eine Schreibtischlampe als Lichtquelle zu verwenden. Nach vielen Versuchen wurde herausgefunden, dass es am besten funktionierte, wenn nur eine Handytaschenlampe als Lichtquelle verwendet wurde, und etwa 15 Nummern gescannt werden konnten, aber immer noch nicht alle. Ein weiteres Problem ist, dass die von uns ausgedruckte Sudoku-Tabelle den Lichtsensor in die Irre führt. Die vertikalen Linien zwischen den Quadraten werden vom Lichtsensor fälschlicherweise als Zahlen 1 interpretiert, was dazu führt, dass die endgültige Anzeige der Tabelle viele Zahlen 1 enthält, was eindeutig gegen die Sudoku-Regeln verstößt und dazu führt, dass der Roboter auch die folgenden Lösungsoperationen nicht richtig ausführen kann.

Abbildung 6. Zahlen scannen

Wie auch auf Abbildung 6 zu sehen ist, wurde eine Lichtquelle von außen hinzugefügt. Als der Scan abgeschlossen war, wurden nur 13 Zahlen auf der NXT-Box angezeigt, was eindeutig nicht den Regeln des Sudoku entspricht.

IV. Ergebnisdiskussion

Nachdem die Vorschläge und Genehmigungen der Professoren eingeholt worden waren, wurden die implementierten Funktionen des Roboters geändert. Der Roboter musste in die Lage versetzt werden, die Zahlen 1 bis 9 ohne Probleme auf ein leeres Blatt Papier zu schreiben, und es wurde auch versucht, "Danke" zu schreiben.

V. Zusammenfassung und Fazit

Zusammenfassend lässt sich sagen, dass während des gesamten Projekts einige Probleme auftraten und nach erfolglosen Versuchen der Rat der Professoren in Anspruch genommen wurde, um das neue Ziel zu erreichen. In diesem Projekt können auch einige Teile gefunden werden, die noch verbessert werden müssen. In Zukunft kann der Lichtsensor durch eine Webcam ersetzt werden, die alle Zahlen deutlicher und einfacher ablesen kann.

Anhang

```
COM_SetDefaultNXT(handle);
Downie = NXTMotor('A','Power',100);
Downie.TachoLimit = 250;
Downie.SendToNXT();
Downie.WaitFor();
Uppie = NXTMotor('A','Power',-100);
Uppie.TachoLimit = 250;
Uppie.SendToNXT();
Uppie.WaitFor();
Leftie = NXTMotor('C','Power',-100);
Leftie.TachoLimit = 80;
Leftie.SendToNXT();
Leftie.WaitFor();
Rightie = NXTMotor('C','Power',100);
Rightie.TachoLimit = 80;
Rightie.SendToNXT();
Rightie.WaitFor();%Steuercode
Downp;
Down;
Down;
Right;
Right;
Up;
Up;
Uppie = NXTMotor('A','Power',-100);
Uppie.TachoLimit = 30;
Uppie.SendToNXT();
Uppie.WaitFor();
Left;
Left;
Leftieyo = NXTMotor('C','Power',100);
Leftieyo.TachoLimit = 10;
Leftieyo.SendToNXT();
Leftieyo.WaitFor();
Upp;
Down;
Downp;
Right;
Right;
Upp;%Beispiel von Acht
```

Der folgende Code zeigt als Beispiel, wie die Zahl Acht geschrieben wird. Die darin enthaltenen Funktionen stellen die Bewegungsfunktionen der Kabine und des Auslegers dar.

Der Grund dafür, dass dieselbe Anweisung zweimal wiederholt werden muss, liegt an der Reibung zwischen dem Stift und dem Papier, und durch mehrere Versuche wurde herausgefunden, wo die Anweisung zweimal wiederholt werden muss.

Literaturverzeichnis

[1] WIKIPEDIA: Regel von Sudoku ,https://en.wikipedia.org/wiki/Sudoku, Version:Februar 2024
[2] Hans Andersson:
Sudoku Solver, https://tiltedtwister.com/sudokudownload.html, Veision:Februar 2024

Robotik trifft Logik: Sudoku Solver

Affaan Sameer Shaikh, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Das Ziel des Projekts bestand in der Konstruktion eines Roboters, der in der Lage ist, ein Sudoku-Rätsel auf einem Blatt Papier zu identifizieren und zu lösen. Das zu lösende Sudoku-Rätsel wird zunächst gescannt, um eine Identifikation des gegebenen Sudoku-Rätsels zu ermöglichen. Im Anschluss werden die Zahlen an MATLAB gesendet, wo das Sudoku mit Hilfe von Kodierung gelöst wird. Schließlich wird die Lösung, die von MATLAB geschickt wurde, vom Roboter geschrieben. Das Projekt wurde in drei Phasen unterteilt: Scannen des Sudoku, Lösen in MATLAB und Schreiben der Lösung. Der Lichtsensor erwies sich als zu schwach, um das Sudoku präzise zu scannen. Das Projekt scheiterte bereits in Phase 1, sodass eine Weiterführung nicht möglich war.**

***Schlagwörter*—Schlagwörter—LEGO Mindstorms, MATLAB, Sudoku, Lichtsensor, Roboter, Projektseminar**

I. Einleitung

DER Sudoku-Solver Roboter beinhaltet eine Vielzahl von Komponenten. Das klassische japanische Rätsel wurde mit großem Eifer gelöst. Allerdings wurden die Schwierigkeiten, die auftreten würden, sowie die zahlreichen Stunden, die für die Programmierung, Fehlerbehebung und ähnliches erforderlich wären, nicht vorhergesehen. Der Roboter verfügt über keinen spezifischen Anwendungsbereich, dennoch stellt das Projekt eine interessante Herausforderung dar. Für das Lösen eines Sudoku-Rätsels wurden etwa zehn Minuten benötigt, während für das Lösen zwanzig Minuten benötigt wurden. Die erste Herausforderung bestand darin, den Roboter so zu bauen, dass sich der Lichtsensor frei um das Sudoku herum bewegen kann. Des Weiteren musste sichergestellt werden, dass sich der Stift komfortabel um das Sudoku herum bewegen lässt. Daher wurde beschlossen, den Stift und das Sudoku in der gleichen Halterung zu befestigen, sodass sie sich gemeinsam bewegen können (Abbildung 1). Der Lichtsensor und der Stift wurden daher zusammen mit Rädern wie bei einem Auto befestigt, sodass sich das System mit einem einzigen Motor vorwärts und rückwärts bewegen lässt. Diese Vorgehensweise wird im weiteren Verlauf detaillierter erläutert.

II. Vorbetrachtungen

Das Projekt bestand in der Konstruktion eines Roboters, der in der Lage ist, Sudokus zu lösen. Daher war es erforderlich, sich mit der Lösungsmethode von Sudokus vertraut zu machen, was mit MATLAB in Angriff genommen wurde.

A. Konzept des Sudokus und Grundlagen

Ein klassisches Sudoku weist ein 9×9-Gitter auf, welches sich leicht in neun 3×3-Gitter unterteilen lässt. Das Hauptziel eines Sudokus besteht darin, das Gitter mit den Zahlen 1 bis 9 so auszufüllen, dass jede Ziffer nur einmal in jeder Zeile und nur einmal in jeder Spalte erscheint. Unter der Voraussetzung, dass der Zeilenindex mit i und der Spaltenindex mit j und $k = 3$ bezeichnet werden, lässt sich folgende Gleichung aufstellen:

$$\forall 1 \leq i \leq k^2, \forall 1 \leq x, j \leq k^2 : S(i,j) = 0 \vee x \neq j \quad (1)$$
$$\rightarrow S(i,j) \neq S(i,x)$$

$$\forall 1 \leq j \leq k^2, \forall 1 \leq x, i \leq k^2 : S(i,j) = 0 \vee x \neq i \quad (2)$$
$$\rightarrow S(i,j) \neq S(x,j)$$

$$\forall 0 \leq i, j \leq k-1, \forall 1 \leq a, b, c, d \leq k : \quad (3)$$
$$S(i \cdot k + a, j \cdot k + b) = 0 \vee (a \neq c \vee b \neq d)$$
$$\rightarrow S(i \cdot k + a, j \cdot k + b) \neq S(i \cdot k + c, j \cdot k + d)$$

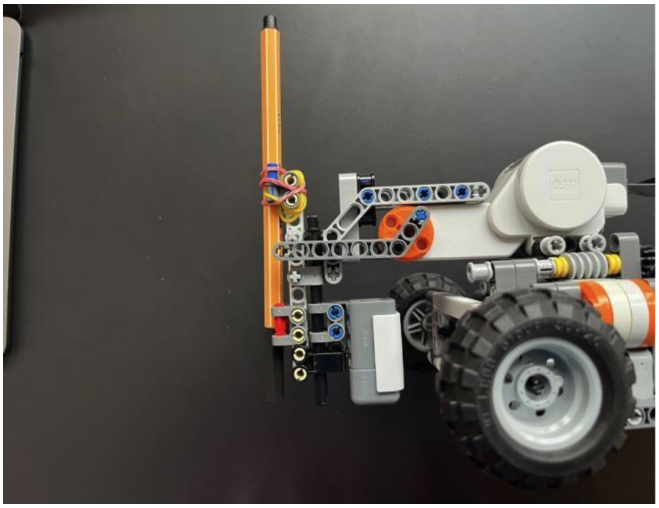


Abbildung 1. Stift und Lichtsensor zusammen

III. Realisierung

A. Aufbau

Der Aufbau besteht aus einem Lichtsensor und einem Motor. Die Bewegung von zwei Rädern ist in der Regel relativ einfach, jedoch gestaltete sich die Befestigung derart, dass sich vier Räder gleichzeitig mit einer stabilen Struktur bewegen konnten. Dies führte zur Idee, einen Roboter mit einem Motor zu bauen. Die Anbringung eines Stiftes und eines Sensors an der Vorderseite ermöglichte die Bewegung, das Scannen und das Schreiben. Des Weiteren wurde die Entscheidung getroffen, einen Motor einzusetzen, der den Stift dreht, sowie

DOI: 10.24352/UB.OVGU-2024-006

einen weiteren Motor, der den Stift hoch und runter bewegt. Dadurch sollte eine einfache Kodierung der beiden Motoren gewährleistet werden. Die zugrunde liegende Idee schien vielversprechend, allerdings konnte sie in der Praxis nicht umgesetzt werden. Der resultierende Roboter (Abbildung 2) wies eine hohe Instabilität und Ineffizienz auf.

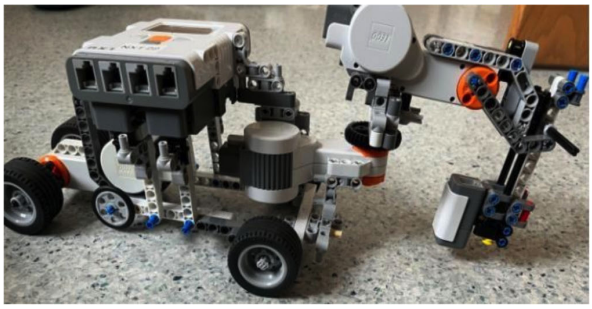


Abbildung 2. Alte Aufbau des Sudoku Solver

In der vorliegenden Version wurde ein größeres Zahnrad verwendet, das über mehr Befestigungen verfügte. Dadurch konnte eines der zuvor genannten Probleme, nämlich die Instabilität, die durch das Gewicht des Lichtsensors verursacht wurde, gelöst werden (Abbildung 3). Mithilfe dieses Zahnrads war es dem Roboter nun möglich, den Stift zu bewegen und auch perfekt zu schreiben.

Abbildung 3. Zahnrad

Um die Stabilität der NXT-Box zu gewährleisten, wurden erstens größere Räder verwendet und zweitens die gesamte Struktur direkt an der NXT-Box befestigt. Auf diese Weise konnte auch das zweite Problem gelöst werden (Abbildung 4). Als Ergebnis wurde ein Roboter erzielt, der sich vorwärts und rückwärts bewegte und dem Stift ermöglichte, sich auf und ab zu bewegen, was das Scannen und Schreiben sehr einfach machte.

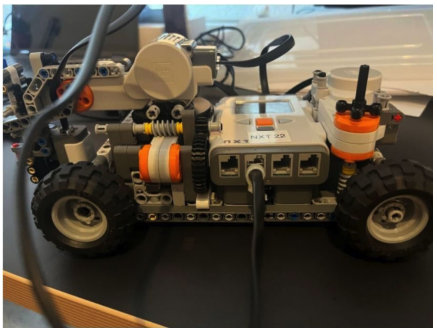


Abbildung 4. Neuer Aufbau des Sudoku Solver

B. *Software*

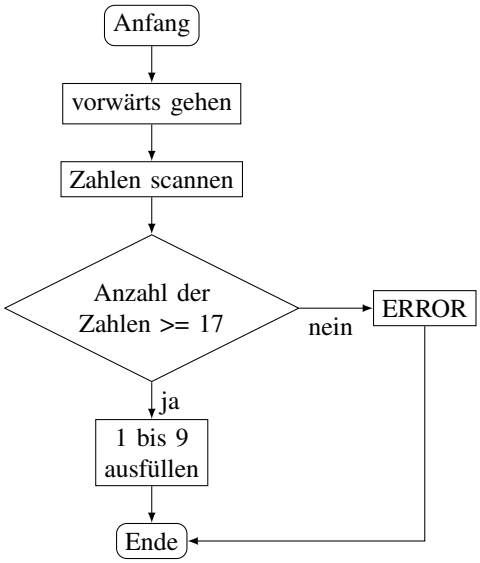


Abbildung 5. Ablaufplan

Die erste Herausforderung bestand also darin, die Zahlen mithilfe eines Lichtsensors zu scannen. Dies stellt eine grobe Erklärung für das dar, was versucht wurde zu erreichen (Abbildung 5).

Abbildung 6. Zahlen Scannen

Die Lösung des Sudokus wird durch einen Algorithmus ermittelt, der nach dem bewährten Backtracking-Prinzip [3] funktioniert. Die Lösung wird durch ein gezieltes Ausprobieren ermittelt, wobei mit der ersten freien Zelle begonnen wird. Es wird nach einer gültigen Ziffer gesucht. Sofern eine gültige Ziffer identifiziert wird, wird der Prozess in der nächsten freien Zelle wiederholt. Andernfalls wird ein Schritt zurückgesetzt und nach einer neuen, gültigen Ziffer gesucht. Die Vorteile dieses Algorithmus liegen in seiner einfachen Rekursivität sowie der Garantie, dass er eine Lösung findet, sofern eine vorhanden ist. Eine weitere Optimierung des Algorithmus kann durch die Berücksichtigung der Anzahl der möglichen Lösungskandidaten in der Abarbeitungsreihenfolge anstelle einer sequenziellen Bearbeitung der freien Zellen erzielt werden. Es wird empfohlen, nach jedem Schritt zu prüfen, welche Zelle die meisten Lösungskandidaten aufweist und diese als nächstes zu bearbeiten. Im Anschluss erfolgt die Bearbeitung der freien Zelle mit der geringsten Anzahl an Lösungskandidaten.

Diese Methode führt in der Regel zu einer erheblichen Reduzierung der Laufzeit. Des Weiteren wird dem Benutzer der aktuelle Stand angezeigt.

C. Probleme

Der Sensor war nicht in der Lage, die Zahlen exakt zu scannen, beispielsweise konnte er den Unterschied zwischen 1 und 7 nicht erkennen, da sie sich sehr ähneln. Zudem wurde nicht immer erkannt, dass eine Zahl überhaupt existierte. Der Code wurde so gestaltet, dass mindestens 17 Zahlen erkannt werden müssen, um mit der Lösung des Sudokus beginnen zu können. Die Existenz dieser 17+ Zahlen wurde vom Sensor nicht einmal erkannt, geschweige denn, dass diese Zahlen identifiziert und dann das Sudoku gelöst wurden. Es wurde in helleren und dunkleren Umgebungen ausprobiert, um zu sehen, ob der Sensor auf diese Weise besser abschnitt (Abbildung 7). Selbst das Handy wurde direkt neben dem Sensor beleuchtet, in der Hoffnung, dass es funktionieren würde, jedoch war leider die Zeit vorbei. Im Rahmen der durchgeführten Untersuchungen wurde ein Code entwickelt, der ein Sudoku lösen kann, sowie ein Roboter, der Zahlen schreiben kann. Aus diesem Grund wurde beschlossen, den Roboter als Abschlussprojekt vorzustellen, der auf Befehl Zahlen schreibt.

Abbildung 7. Versuch mit Licht aus Handy

IV. Ergebnisdiskussion

Nachdem die Vorschläge und Genehmigungen der Professoren eingeholt worden waren, wurden die implementierten Funktionen des Roboters geändert. Der Roboter musste in die Lage versetzt werden, die Zahlen 1 bis 9 ohne Probleme auf ein leeres Blatt Papier zu schreiben, und es wurde auch versucht, ”Danke” zu schreiben.

V. Zusammenfassung und Fazit

Der Code wurde hauptsächlich in 4 Teile aufgeteilt: Vorwärts, Rücken, Links, Rechts, basierend auf der Bewegung des Stifts. Zum Beispiel würde zweimal abwärts/aufwärts eine gerade Linie ergeben, die die Nummer eins ergibt. Ähnlich würde rechts, rücken, links, rechts, rücken, links die Zahl drei ergeben.

Anhang

```
COM_SetDefaultNXT(handle);
Downie = NXTMotor('A','Power',100);
Downie.TachoLimit = 250;
Downie.SendToNXT();
Downie.WaitFor();
Uppie = NXTMotor('A','Power',-100);
Uppie.TachoLimit = 250;
Uppie.SendToNXT();
Uppie.WaitFor();
Leftie = NXTMotor('C','Power',-100);
Leftie.TachoLimit = 80;
Leftie.SendToNXT();
Leftie.WaitFor();
Rightie = NXTMotor('C','Power',100);
Rightie.TachoLimit = 80;
Rightie.SendToNXT();
Rightie.WaitFor();%Steuercode
Downp;
Down;
Down;
Right;
Right;
Up;
Up;
Uppie = NXTMotor('A','Power',-100);
Uppie.TachoLimit = 30;
Uppie.SendToNXT();
Uppie.WaitFor();
Left;
Left;
Leftieyo = NXTMotor('C','Power',100);
Leftieyo.TachoLimit = 10;
Leftieyo.SendToNXT();
Leftieyo.WaitFor();
Upp;
Down;
Downp;
Right;
Right;
Upp;%Beispiel von Acht
```

Der folgende Code zeigt, wie die Zahl Acht geschrieben wird, und die Funktionen darin repräsentieren die Bewegungen der Kabine und des Auslegers. Die Notwendigkeit, eine Anweisung zweimal zu wiederholen, resultiert aus Reibung zwischen Stift und Papier, und durch mehrere Versuche wurde festgestellt, wo dies erforderlich ist.

Literaturverzeichnis

[1] Adriano Parracciani:
One-Motor Car Lego NXT, https://www.pinterest.de/pin/537054324290642123/, Version:Februar 2024

[2] Hans Andersson:
Sudoku Solver, https://tiltedtwister.com/sudokudownload.html, Version:August 2009

[3] Eckart Sußenburger:
Lösungs und Generierungsalgorithmen für Sudoku. Trier: Fachhochschule Trier, Fachbereich Informatik, Bachelor Abschlussarbeit 17.04.2007

Kontaktloser Seifenspender

Mohamed Adil Moussaid, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*— Im Rahmen des Projektseminars Elektrotechnik/Informationstechnik (LEGO Mindstorms) 2024 an der Otto-von-Guericke-Universität Magdeburg wurde ein kontaktloser Seifenspender entwickelt. Dieser Roboter wurde mit dem zur Verfügung gestellten LEGO-NXT-Set und MATLAB konstruiert und programmiert.**

Der Mechanismus des automatischen Seifenspenders basiert im Wesentlichen auf einem Sensor, der die Anwesenheit von Händen erkennt und dann eine bestimmte Menge Seife durch einen Motor abgibt, ohne dass der Benutzer den Spender berühren muss.

Der Entwicklungsprozess dieses Roboters, die damit verbundenen Herausforderungen und die Ergebnisse wurden während des Seminars vorgestellt und seine Funktionalität demonstriert. Dieses Projekt zeigt, wie selbst einfache Aufgaben durch den Einsatz von Technologie automatisiert und verbessert werden können.

***Schlagwörter*— Seifenspender, LEGO Mindstorms, MATLAB, Roboter, Ultraschallsensor, Motor.**

I. Einleitung

In der heutigen Zeit, in der Hygiene und Sauberkeit von größter Bedeutung sind, wird ein Projekt zur Entwicklung eines automatischen Seifenspenders durchgeführt. Die Notwendigkeit für dieses Projekt ergibt sich aus dem Wunsch, die Handhygiene zu verbessern und die Verbreitung von Infektionskrankheiten einzudämmen. Derzeit werden manuelle Seifenspender verwendet, die jedoch eine potenzielle Quelle für die Übertragung von Krankheitserregern darstellen können. Automatische Seifenspender, die auf Bewegungssensoren reagieren, stellen eine fortschrittlichere Lösung dar, die jedoch noch verbessert werden kann. In diesem Projekt soll ein automatischer Seifenspender entwickelt werden, der mit Hilfe eines Ultraschallsensors die Handbewegungen des Benutzers erkennt und eine voreingestellte Menge Seife abgibt. Die Herausforderung besteht darin, genügend Kraft aufzubringen, um die Seifenpumpe zu betätigen. Außerdem muss eine Lösung gefunden werden, die sowohl kostengünstig als auch einfach zu bedienen ist. Durch die Lösung dieser Herausforderungen soll ein Beitrag zur Verbesserung der öffentlichen Gesundheit und Sicherheit geleistet werden. Es wird erwartet, dass dieser spannende Weg weiter beschritten wird.

II. Vorbetrachtungen

Die verwendeten mechanischen LEGO-Komponenten und Sensoren werden kurz vorgestellt.

A. Ultraschall-Sensor

Um den Abstand zwischen der Hand des Bedieners und dem Flaschenkopf zu messen, wird ein Ultraschallsensor verwendet, siehe Abbildung 1. Der Ultraschallsensor verwendet einen speziellen Schallwandler, der das selektive Senden und Empfangen von Schallwellen ermöglicht. Der Schallwandler sendet eine bestimmte Anzahl von Schallwellen aus, die vom zu erfassenden Objekt reflektiert werden. Nach dem Senden der Impulse schaltet der Ultraschallsensor auf Empfangsbetrieb um. Die Zeit bis zum Eintreffen eines eventuellen Echos ist proportional zur Entfernung des Objekts vom Näherungsschalter.

Abbildung 1: LEGO-Ultraschallsensor [3]

B. Schneckengetriebe

Ein Schneckengetriebe ist eine Kombination aus einem Schraubradgetriebe und einem Zahnradgetriebe und besteht aus einer schraubenförmigen Schneckenwelle und einem Zahnrad, dem Schneckenrad. Das Gewinde der Schneckenwelle greift in die Zahnlücken des Schneckenrades, siehe Abbildung 2.

DOI: 10.24352/UB.OVGU-2024-007

.

Abbildung 2: LEGO-Schneckengetriebe [4]

III. KONSTRUKTION UND PROGRAMMIERUNG

Hier wird detailliert beschrieben, wie der Roboter gebaut und programmiert wurde.

A. Aufbau

Der Seifenspender-Roboter ist ein innovatives Projekt, das die Prinzipien der Robotik und Mechanik nutzt, um eine alltägliche Aufgabe zu automatisieren. Der Hauptbestandteil des Roboters ist der NXT-Block, der als Gehirn des Roboters fungiert und die Steuerung der beiden Motoren ermöglicht. Diese Motoren treiben das System an, das den Seifenspender betätigt, siehe Abbildung 3.

Die Konstruktion des Roboters beinhaltet auch die Verwendung eines Ultraschallsensors. Dieser Sensor erkennt die Anwesenheit einer Hand unter dem Spender und aktiviert das System, um Seife auszugeben.

Trotz der sorgfältigen Planung und Konstruktion gab es einige Herausforderungen beim Bau des Roboters. Eine der größten Herausforderungen war das geringe Drehmoment am Pumpenknopf des Seifenspenders. Dieses Problem wurde durch den Einsatz eines Schneckengetriebes gelöst. Ein Schneckengetriebe ist ein spezielles Getriebe, das ein hohes Drehmoment erzeugen kann, was in diesem Fall für die Betätigung des Pumpenknopfes des Seifenspenders notwendig war.

Neben der Verwendung des Schneckengetriebes war es auch wichtig, eine stabile Konstruktion zu bauen. Die Lego-Bausteine boten die nötige Stabilität und Flexibilität, um eine robuste und effiziente Konstruktion zu schaffen.

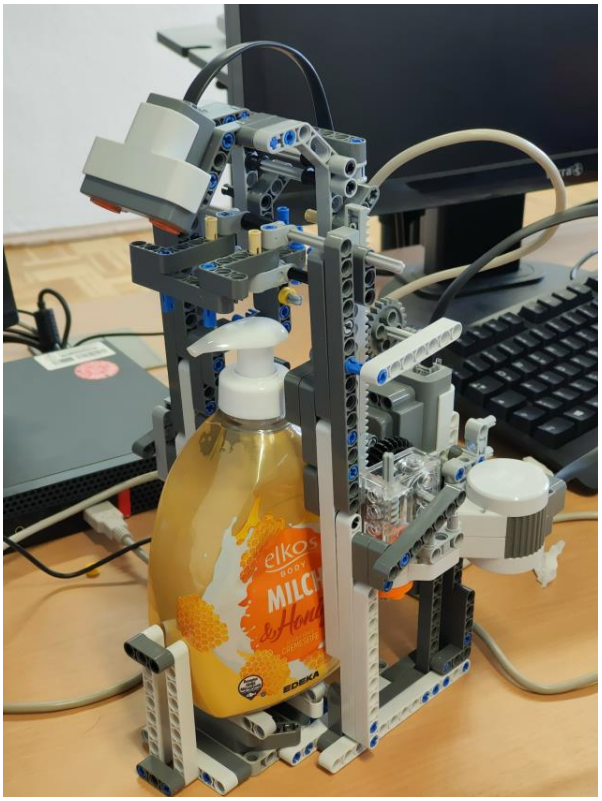


Abbildung 3: Aufbau des automatischen Seifenspenders

B. MATLAB Programm

Die Programmierung eines automatischen Seifenspenders mit MATLAB und LEGO Mindstorms erfordert ein tiefes Verständnis sowohl der Hardware- als auch der Softwarekomponenten.

Bei der Programmierung mit MATLAB ist es wichtig, die spezifischen Befehle und Funktionen zu verstehen, die zur Steuerung des Motors und des Sensors verwendet werden. Matlab bietet eine Vielzahl von Funktionen und Bibliotheken, die speziell für die Interaktion mit Hardware wie Lego Mindstorms entwickelt wurden. Diese Funktionen ermöglichen es dem Programmierer, Befehle an den Motor zu senden, Sensorinformationen zu lesen und auf diese Informationen zu reagieren.

Ein wichtiger Aspekt bei der Programmierung dieses Projekts war die Implementierung der Logik zur Erkennung der Entfernung eines Objekts, z. B. einer Hand. Der Algorithmus musste so entwickelt werden, dass er auf diesen Abstand reagiert, indem er den Motor startet und Seife ausgibt. Dies erforderte eine genaue Kalibrierung des Abstandssensors und eine sorgfältige Programmierung, um sicherzustellen, dass der Sensor korrekt funktioniert.

Außerdem musste der Algorithmus zwischen verschiedenen Zuständen wechseln können, je nachdem, ob ein Objekt erkannt wurde oder nicht. Dies erforderte die Verwendung von Kontrollstrukturen wie Schleifen und bedingten Anweisungen.

Wichtig ist auch, dass die Programmierung eines solchen Systems ein iterativer Prozess ist. Oft muss der Code getestet und angepasst werden, um sicherzustellen, dass er wie erwartet funktioniert. Dies kann besonders schwierig sein, wenn mit Hardware wie Lego Mindstorms gearbeitet wird, da unvorhergesehene Probleme auftreten können, die eine Anpassung des Codes erforderlich machen, siehe Abbildung 4.

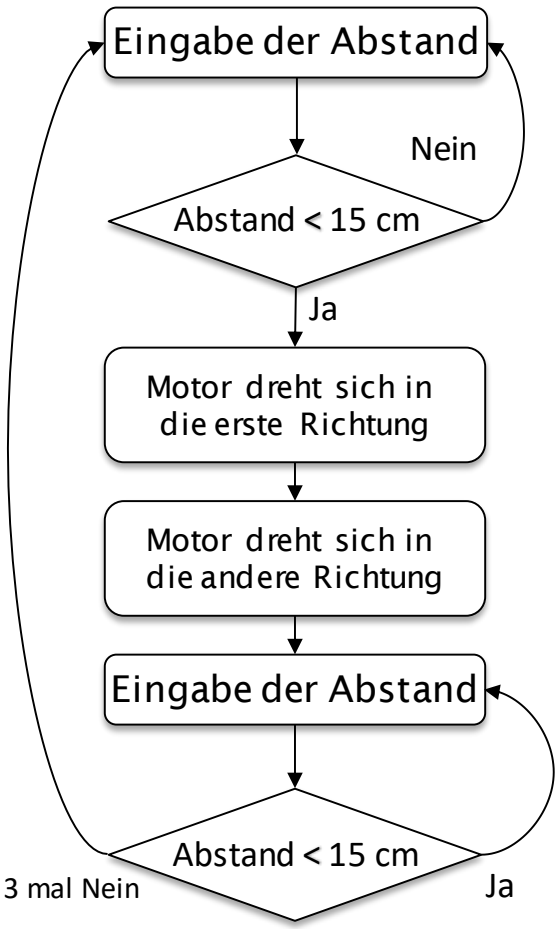


Abbildung 4: Algorithmus des Roboters

IV. Ergebnisdiskussion

Das Projektziel, einen automatischen Seifenspender zu entwickeln, wurde erfolgreich erreicht. Der Spender gibt Seife korrekt aus, wenn eine Hand vorhanden ist, und wartet, bis die Hand entfernt wird, bevor der gesamte Vorgang von neuem beginnt. Dies wurde durch Ergänzungen und Korrekturen des Codes erreicht, die zufällige Eingabefehler berücksichtigen.

Der integrierte Ultraschallsensor hat sich als effektives und zuverlässiges Mittel zur Verbesserung der Funktionalität des automatischen Seifenspenders erwiesen. Er versorgt das Programm mit genauen und zuverlässigen Messwerten. Es gab jedoch Herausforderungen bei der Implementierung, wie die Feinabstimmung des Sensors und die korrekte Programmierung des Systems.

Ein weiterer bemerkenswerter Aspekt des Projekts ist seine Flexibilität. Die Konstruktion kann für andere Zwecke modifiziert werden, z. B. für die Ausgabe von Shampoo, und ist nicht auf Seife beschränkt.

Insgesamt ermöglicht der automatische Seifenspender den Benutzern nun, Seife ohne physischen Kontakt mit dem Spender zu dosieren. Dies ist besonders in Umgebungen mit vielen Benutzern von Vorteil. Trotz einiger Herausforderungen bei der Umsetzung wurde das Projektziel erreicht und hat Potenzial für weitere Anwendungen über die Seifenspende hinaus.

V. Zusammenfassung und Fazit

Insgesamt ist der Seifenspender-Roboter ein hervorragendes Beispiel für die Anwendung von Robotik und Mechanik zur Lösung alltäglicher Probleme. Trotz einiger Herausforderungen während der Konstruktion gelang es dem Team, durch kreatives Denken und technisches Know-how eine funktionierende Lösung zu entwickeln.

Literaturverzeichnis

[1] Baumer: Ultraschall-Näherungsschalter (Funktionsweise) https://www.baumer.com/de/de/service-support/funktionsweise/funktionsweise-und-technologie-von-ultraschallsensoren/a/Know-how_Function_Ultrasonic-sensors

[2] Wikipedia (Schneckengetriebe): https://de.wikipedia.org/wiki/Schneckengetriebe

[3] Amazon (Abbildung 1): https://www.amazon.de/LEGO-NXT-Ultraschallsensor-Ultrasonic-sensor/dp/B000PM8I8O

[4] Steinpalast (Abbildung 2): https://www.steinpalast.eu/en/1-x-lego-brick-trans-clear-technic-gearbox-2-x-4-x-3-1/3/black-technic-gear-worm-screw-long/axle/technic-gear-24-tooth-3648-24505-4716-6588-32239

Kontaktloser Seifenspender

Felix Müller, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Abstract*— Das Ziel des diesjährigen Projektseminars Elektrotechnik und Informationstechnik LEGO Mindstorms war es einen Roboter aus LEGO-Teilen zu Bauen und mit MATLAB zu Programmieren. In diesem Rahmen wurde ein kontaktloser Seifenspender gebaut, welcher in der Lage ist, ohne physischen Kontakt Seife auszugeben. In diesem Paper werden folglich Aufbau, Funktionsweise und Probleme mit deren Lösungsansätzen dargestellt.**

***Schlagwörter*—Getriebe, Kontaktloser Seifenspender, LEGO Mindstorms, MATLAB, Motor, Ultraschallsensor**

I. Einleitung

HEUTZUTAGE ist der globale Fokus auf Hygiene noch nie so potent gewesen, wie in den Jahren zuvor. Überall sind Keime und dreckige Partikel, welche für Krankheiten sorgen können. Darum wurde eine hohe Priorität auf das regelmäßige Waschen der Hände gelegt. Allerdings benötigt man dafür nicht nur Wasser, sondern auch Seife. Wenn man dann jedoch mit seiner dreckigen Hand den Seifenspender betätigt, verteilt man die Keime an alle anderen Personen, die nach einem den Seifenspender anfassen. Darum wurde der Kontaktlose Seifenspender entwickelt, damit man Zugriff auf Seife hat ohne Verunreinigungen zu verteilen, ähnlich wie bei einem kontaktlosen Wasserhahn.

II. Vorbetrachtungen

Bei diesem Projekt waren die Bedenken von Anfang an, ob die LEGO-Konstruktion genug Stabilität leisten kann und die Motoren genug Kraft aufbringen können, um den Seifenspender zu Bedienen.

A. Inspiration

Vor dem Start des Bauprozesses wurde die Idee zu einem Roboter, der kontaktlos Seife spenden kann von einem Roboter aus dem vorherigen Projektseminar LEGO Mindstorms 2022/23 gezogen [1]. Der Plan war diesen zu verbessern, indem er es schafft, schneller die Seife auszugeben ähnlich wie bei einem richtigen automatischem Seifenspender. Als es dann in den Bauprozess ging, wurde der grobe Aufbau des Roboters ebenso von [2] inspiriert. Allerdings wurde schnell festgestellt, dass die benötigte Kraft den Seifenspender hinunter zudrücken zu gering ist, weswegen Modifikationen vorgenommen werden mussten.

B. Ultraschallsensor

Um das Vorhandensein der Hand an dem Kontaktlosem Seifenspender zu messen wird ein Ultraschallsensor (Abbildung 1) benutzt. Dieser wurde über dem Seifenspenderkopf befestigt, um die Entfernung zum nächstgelegenen Objekt darunter zu messen. Dazu sendet der Sensor Ultraschallsignale ab, welche am nächsten Objekt zurück in Richtung des Sensors abgeprallt werden. Der Ultraschallsensor misst dann die Zeit, die das Ultraschallsignal gebraucht hat, um die Entfernung zu einem Objekt zu messen. Daher erkennt der kontaktlose Seifenspender, ob sich eine Hand darunter befindet, indem dieser eine kürzere Entfernung zum Boden misst.

Abbildung 1: LEGO-Ultaschallsensor [3]

C. Motoren und Getriebe

Um die Seife aus dem Seifenspender zu bekommen, muss eine relativ starke Kraft aufgebracht werden, um den Kopf nach unten zu drücken. Allerdings sind die Motoren dafür zu schwach, wie nach einem Prototyp festgestellt wurde. Deswegen wurden verschieden Getriebearten getestet, um die Kraft zu verstärken. Zuerst wurde ein großes Zahnrad an ein kleines geschlossen. Dadurch wird die abgegebene Kraft mit dem Verhältnis der Zahnradgrößen multipliziert und somit verstärkt.

$$M_{zu} = \frac{z_1}{z_2} \cdot M_{Ab} \quad (1)$$

z-Anzahl der Zähne der Zahnräder

Allerdings hat die entstehende Kraft immer noch nicht ganz gereicht, um den Seifenspender jedes Mal zu drücken. Außerdem hat die Stabilität des ganzen Konstruktes bei diesem Prototyp unter diesem Getriebe gelitten. Darum wurde sich schlussendlich für ein Schneckengetriebe entschieden, welches noch mehr Kraft aufbringen konnte und zu dem noch sehr stabil war.

III. Umsetzung

In diesem Teil wird der genaue Aufbau des Roboters und die Umsetzung des Programmiertechnischen Teils eingegangen.

DOI: 10.24352/UB.OVGU-2024-008

A. Aufbau

Der Roboter, wie in Abbildung 2 zu sehen, besteht aus 4 elektrischen Bauteilen. Das erste Bauteil ist der NXT-Block, der den eingegebenen Code an die anderen Teile weitergibt. Dann enthält er zwei Motoren, die die Kraft erzeugen, um den Seifenspender zu drücken. Außerdem befindet sich auch noch ein Ultraschallsensor, um die Hand zu erkennen, in dem Roboter. Zudem wurden noch andere normale LEGO-Teile in der Konstruktion benutzt.
Der normale Seifenspender ist von allen vier Seiten mit LEGO-Teilen fixiert und kann sich daher während des Spendeprozesses nicht wegbewegen. Außerdem kann man ihn leicht austauschen, indem man ihn nach vorne neigt, das vordere LEGO-Konstrukt zur Seite bewegt und dann den Seifenspender entnimmt.
Auf der Rückseite befindet sich der NXT-Block, welcher mit Kabeln mit den Motoren und dem Ultraschallsensor verbunden wurde. Dieser sorgt auch dafür, dass der Seifenspender nicht nach hinten wegkippen kann.
An den beiden Seiten wurde mit LEGO-Teilen nach oben gebaut. Dort befinden sich auch die Schneckengetriebe und die Motoren. Die Motoren spenden die eingehende Kraft in das Schneckengetriebe. Die dann vervielfältigte Kraft wird an ein Zahnrad übergeben, welches durch seine Drehbewegung Zähne, die in einer Schiene entlanglaufen, nach unten und oben verschieben kann. Wenn diese Konstruktion aus Zähnen von beiden Seiten mit beiden Motoren nach unten gedrückt wird, kann der Spendeprozess durchgeführt werden, indem die sich bewegende Konstruktion auf den Kopf des Spenders wirkt.
An den beiden Seiten befestigt, befindet sich über dem Seifenspender der Ultraschallsensor, der fest verankert ist und sich nicht bewegen kann.

Abbildung 2: Kontaktloser Seifenspender

B. Schneckengetriebe

In der finalen Konstruktion des Kontaktlosen Seifenspenders wurde sich für ein Schneckengetriebe als Getriebeart entschieden, welches in Abbildung 3 zu sehen ist. Der Grund war der erhebliche Kraftaufwand, der benötigt wurde für das Spenden der Seife. Außerdem bietet das Schneckengetriebe eine gute Stabilität, dass sich die Getriebeteile nicht viel bewegen können und daher den Prozess nicht behindern können. Im Grundlegenden besteht das Getriebe nur aus einer Schraube und einem Zahnrad, welche nah aneinandergesteckt sind, sodass sie sich gegenseitig bewegen können. Wenn ein Motor an der zylindrischen Schraube in dem Schneckengetriebe dreht, fungiert diese dabei, wie ein kleines Zahnrad. Dadurch wird die Kraft durch die unterschiedlichen Größen der Zahnräder deutlich verstärkt. Das größere Zahnrad ist mit einem anderen Zahnrad gekoppelt, welches dann für die Bewegung an der eigentlichen Konstruktion sorgt.
Außerdem ist durch die Verwendung des Schneckengetriebes weniger Platz verbraucht wurden, was zum einen zu einer besseren Funktionsweise und Aufbau führt und zum anderen besser aussieht.

Abbildung 3: Schneckengetriebe [4]

C. Programmierung

Der Code in dem Roboter ist simpel. Der Ultraschallsensor misst dauerhaft den Abstand zum Boden. Solange der Abstand größer als 15 cm bleibt, passiert auch nichts und der Ultraschallsensor misst weiter. Sollte sich dieser verringern auf unter 15 cm, weil eine Hand den Abstand verkleinert hat, dann fangen die Motoren an sich zu bewegen. Dabei bewegt sich eine Konstruktion nach unten, welche auf den Kopf des Seifenspenders drückt. Wenn dieser komplett eingedrückt ist warten die Motoren kurz und drehen sich in die andere Richtung, um die Konstruktion wieder nach oben in die Ausgangsposition zu bewegen.
Nachdem der eigentliche Code implementiert wurde, gab es noch einen Verbesserungsvorschlag, welcher umgesetzt wurde. Dabei ging es um die Tatsache das der Spender dauerhaft spendet sollte ein Objekt sich darunter befinden oder falls der Ultraschallsensor falsche Werte misst. Darum wurde im Code eingefügt das nach der Seifenspende der Abstand nochmal gemessen wird und erst wenn der Abstand sich für kurze Zeit

größer als 15 cm hält, darf der Kontaktlose Spender wieder Seife spenden.

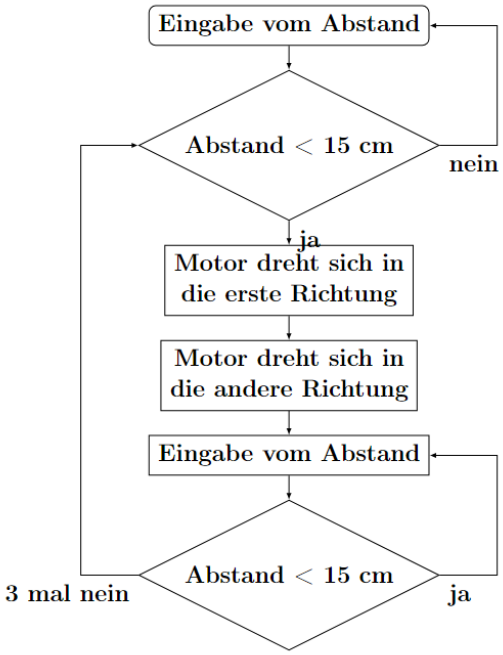


Abbildung 4: Flussdiagramm von dem Code

IV. Ergebnisdiskussion

Das Ergebnis des Projektes war wie am Anfang erwartet ein funktioneller Kontaktloser Seifenspender, der mithilfe eines Ultraschallsensors und Motoren funktioniert. Einige Probleme, die während des Prozesses aufgekommen sind konnten am Schluss behoben werden. Das Problem mit den Getrieben wurde schon ausführlich erklärt, aber ein anderes Problem war die Stabilität des Konstruktes. Da es sich immer noch um ein LEGO-Roboter handelt sind alle Teile sehr biegsam und nicht wirklich stabil. Es kam oft dazu, dass sich Teile verbogen haben, anstatt den Seifenspenderkopf nach unten zu drücken. Ein anderes Problem war es die beweglichen Zähne, die an den Zahnrädern gekoppelt sind, in eine Art Schiene zu stecken, damit diese sich nur nach oben und unten bewegen können und diese sich nicht drehen.

V. Zusammenfassung und Fazit

Das Projekt war auf jeden Fall erfolgreich. Es wurde das umgesetzt, was am Anfang erwartet wurde. Ein Kontaktloser Seifenspender wurde gebaut und der geschriebene Code funktioniert einwandfrei. Es gab mehrere Prototypen, welche immer umgebaut wurden, um die Fehler zu beseitigen. Nebenbei wurde überlegt, wie man den Roboter noch besser machen kann. Man könnte natürlich auch andere Flüssigkeiten ausgeben lassen und zum Beispiel einen automatischen Saucenspender mit diesem Konzept erstellen. Eine möglichkeit der Verbesserung wäre möglicherweise den NXT Block so gut wie möglich einzubauen, um die komplette Konstruktion wasserdicht zu machen, da es beim Händewaschen öfter zu Wasserspritzern kommen kann.

Literaturverzeichnis

[1] Mathias Magdowski. (2023, April). LEGO-Praktikum. *Journal*. *volume 6*, pages 9-15. Available: https://doi.org/10.24352/UB.OVGU-2023-018

[2] Artem 16. (2020, April). EV3 Dispenser [45544]+[45560]. Video. Available: https://youtu.be/jPahgubKM40

[3] Bild Ultraschallsensor: https://www.amazon.de/LEGO-NXT-Ultraschallsensor-Ultrasonic-sensor/dp/B000PM8I8O

[4] Bild Schneckengetriebe: https://www.steinpalast.eu/en/1-x-lego-brick-trans-clear-technic-gearbox-2-x-4-x-3-1/3/black-technic-gear-worm-screw-long/axle/technic-gear-24-tooth-3648-24505-4716-6588-32239

Morsen mit dem El-Mo-Apparat

Mika Schäfer, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Der elektrische Morseapparat (kurz: El-Mo-Apparat) beschreibt einen Roboter, der ein ursprüngliches Morsegerät aus NXT-Bausteinen darstellt. Die Programmierung erfolgt durch MATLAB und erfüllt einen essentiellen Teil für die Funktionalität des Apparats. Der El-Mo-Apparat kann einen Morsecode-Streifen einlesen und diesen in Form von Buchstaben auf einem Bildschirm wiedergeben. Dabei werden zwei verschiedene Versionen thematisiert, bei denen alle ursprünglichen Problematiken behoben werden konnten, sodass beide Apparate einwandfrei funktionieren. Die beiden Versionen bestehen aus einem Tast- beziehungsweise einem Lichtsensor, sodass hier das Morsen mit zwei verschiedenen Anwendungen möglich ist: durch Helligkeitsstufen (elektrische Signale) und das Drücken eines Tasters (mechanische Signale).**

***Schlagwörter*—Lichtsensor, MATLAB, Morsecode, Roboter, Taster**

I. Einleitung

In der heutigen Zeit kennen viele diese Nachrichtentechnik nicht mehr: das Morsen. Somit wurde sich hier mit einer veralteten Praktik auseinandergesetzt und diese mit einer modernen Technik betrieben. Der Morsecode, benannt nach seinem Erfinder Samuel Morse [1], ist eine zeitlose und zugleich faszinierende Form der Kommunikation. Diese beruht auf einer einfachen, aber effektiven Methode: der Übertragung von Informationen mithilfe von kurzen und langen Lichtsignalen oder Tönen.

Die Geschichte des Morsecodes reicht zurück bis ins 19. Jahrhundert, als Samuel Morse dieses Kommunikationssystem entwickelte. Dabei wurden Nachrichten über weite Entfernungen insbesondere über Telegrafenleitungen übermittelt. Was einst als innovative Technologie begann, entwickelte sich zu einem unverzichtbaren Werkzeug der menschliche Kommunikation. Das Besondere des Morsecodes liegt in seiner Einfachheit und seiner Universalität. Durch die Verwendung von nur zwei Signalen, dem Punkt und dem Strich, können Buchstaben, Zahlen und sogar Sonderzeichen codiert und decodiert werden. Diese grundlegenden Elemente bilden das Alphabet des Morsecodes, das es ermöglicht, komplexe Nachrichten ohne die Notwendigkeit einer gesprochenen oder geschriebenen Sprache zu übermitteln.

Im Laufe der Geschichte hat der Morsecode eine bemerkenswerte Vielseitigkeit erreicht. Von seiner Verwendung in der Schifffahrt und im Militär bis hin zu seinen Einsatzmöglichkeiten in Rettungssituationen und im Amateurfunk hat der Morsecode zahlreiche Anwendungen gefunden. Selbst in Zeiten modernster Technologie und digitaler Kommunikation bleibt der Morsecode relevant und wird von vielen als wichtiger Bestandteil des Erbes der Telekommunikation betrachtet.

DOI: 10.24352/UB.OVGU-2024-009

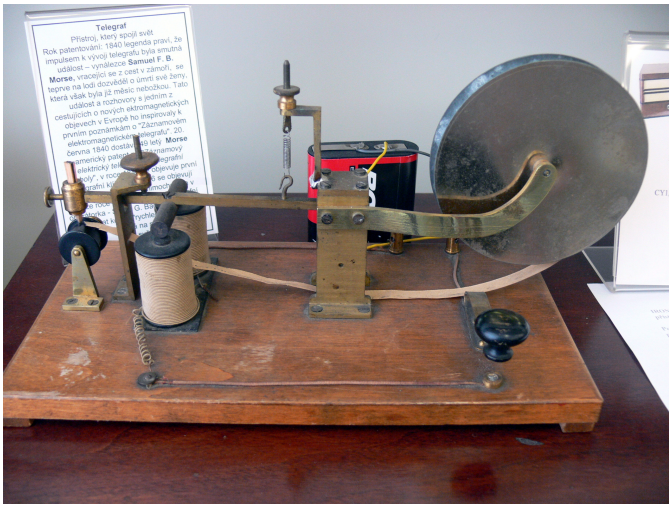


Abbildung 1. Beispiel eines Morsegerätes [2]

II. Vorbetrachtungen

A. Elektromagnetischer Morseapparat

Die Funktionsweise des elektromagnetischen Morseapparats beruht auf dem Prinzip der elektromagnetischen Induktion. Durch das Drücken eines Hebels wird ein Stromkreis geschlossen, sodass Strom durch einen Elektromagneten fließt und ein magnetisches Feld erzeugt wird. Der Elektromagnet zieht einen sogenannten Anker, ein Stück leitendes Metall, an und schließt einen Schaltkontakt. Dadurch wird ein elektrischer Impuls erzeugt, der als Signal verwendet wird [3].

Eine weitere Möglichkeit besteht darin, einen Stift mit dem Anker zu verbinden, der dadurch auf dem Papier angehoben und heruntergelassen wird.

Das Morsealphabet wird durch kurze und lange Impulse dargestellt, die als Punkte und Striche bezeichnet werden. Eine bestimmte Abfolge von Punkten und/oder Strichen stellt dabei immer einen Buchstaben, eine Zahl oder ein Sonderzeichen dar (beispielsweise wird das A durch “Punkt Strich” beschrieben). Der Morseapparat wird bedient, indem der Hebel gedrückt und losgelassen wird, um kurze oder lange Impulse (Punkte beziehungsweise Striche) zu erzeugen. Kurze oder lange Impulse entstehen, indem der Hebel eine kurze bzw. lange Zeit gedrückt wird.

Aus der Kombination von Punkten und Strichen wird die empfangene Nachricht interpretiert. Der elektrische Impuls wird auf der Empfängerseite in akustische oder visuelle Signale umgewandelt, die dann vom Empfänger entschlüsselt werden können.

B. Welche Sensoren eignen sich für einen Morseapparat?

Da auf Basis von LEGO NXT-Mindstorms gearbeitet wurde, ist es schwierig, einen richtigen Stromkreis zu schließen. Aus diesem Grund wurden zwei verschiedene Morseapparate gebaut. Einer dient dem Übersetzen von Buchstaben aus mechanischen Signalen (durch einen Tastsensor). Der zweite übersetzt elektrische Signale (durch einen Lichtsensor) ebenfalls in Buchstaben.

Für die erste Version des El-Mo-Apparats wurden zwei NXT-Tastsensoren verwendet. Diese können entweder in einem gedrückten oder nicht gedrückten Zustand vorliegen. Der Tastsensor liefert entweder eine „1“ (gedrückt) oder eine „0“ (nicht gedrückt) an den verbundenen Computer.

Bei der zweiten Version des El-Mo-Apparats ist es schwieriger, die ursprüngliche Funktion des Morseapparats zu übertragen. Er besteht aus einem NXT-Lichtsensor und einem LEGO-NXT-Motor. Der Lichtsensor erkennt verschiedene Helligkeitsstufen und hat zwei Modi. Im ersten Modus nutzt der Sensor die Umgebungshelligkeit, um eine Helligkeitsstufe zu erkennen. Im zweiten Modus schaltet der Lichtsensor sein eigenes Licht ein und verwendet so nur seine eigene Lichtquelle. Der LEGO-Motor kann um einen bestimmten Winkel oder kontinuierlich gedreht werden. Darüber hinaus können die Geschwindigkeit und das Verhalten beim Anhalten (stoppen oder ausrollen) eingestellt werden.

C. Weitere NXT-Bausteine

Ein weiterer essentieller Baustein ist der NXT-Stein. Er ist in der Lage, sowohl Informationen vom Computer und NXT-Bausteinen zu empfangen, als auch selbst Informationen zu senden. Des Weiteren kann er Töne abspielen.

III. Konstruktion und Programmierung

Zur Verwirklichung des eigenen Morseapparats wurden, wie oben bereits geschrieben, insgesamt zwei Versionen des El-Mo-Apparats konstruiert.

A. El-Mo-Apparat (Version 1)

Die erste Version war der erste Schritt in der Entwicklung eines Morsecode-Apparats. Dieser Roboter bietet die Möglichkeit einer langsamen Eingewöhnung in die Programmierung mit MATLAB.

1) Konstruktion: Um die Konstruktion möglichst einfach zu gestalten, wurde ein echtes Morsegerät als Inspiration genutzt. In Abbildung 2 wird gezeigt, dass der erste Tastsensor unter einer „Wippe“ aus LEGO platziert ist. Mit deren Hilfe kann der Taster gedrückt oder nicht gedrückt werden. Dieser stellt den Morsetaster dar.

Der zweite Sensor ist ebenfalls mit einem Kabel an dem NXT-Stein angeschlossen und dient der Initialisierung. Aus diesem Grund wird dieser als Initialisierungstaster bezeichnet.

2) Programmierung: Das Programm der ersten Version des El-Mo-Apparats funktioniert auf einer einfachen Basis. Zunächst muss der NXT-Stein ständig prüfen, ob der Initialisierungstaster gedrückt ist. Ist dieser nicht gedrückt, kann auch keine Eingabe mit der Morsetaste erfolgen. Wird nun

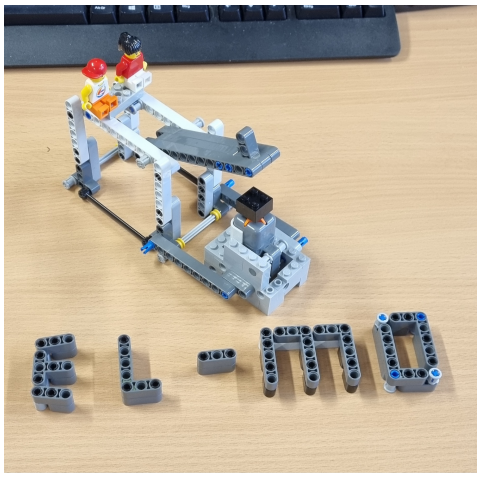


Abbildung 2. El-Mo-Apparat (Version 1)

der Initialisierungstaster gedrückt, erfolgt die Eingabe mit dem Morsetaster, die durch Töne und Textausgaben am NXT-Stein und Computer sichtbar werden. Je nachdem, wie lange der Taster gedrückt wird, wird ein Wert in der Eingabetabelle gespeichert.

In dieser Tabelle werden die Daten abgelegt, die das Programm später benötigt, um einen Buchstaben auf dem Bildschirm auszugeben. Zu beachten ist, dass die Codierung und die Ausgabe von Buchstaben beschränkt ist. Da ein Buchstabe in Morsesprache nur aus maximal vier Zeichen, d.h. Punkten und/oder Strichen besteht, geht das Programm nach der achten Zahl, die es in der Eingabetabelle speichert, zur Auswertung der Tabelle über, wo diese in eine Funktion verarbeitet werden. Nach der Auswertung gibt das Programm den Buchstaben auf dem Bildschirm aus, siehe Abbildung 3.

B. El-Mo-Apparat (Version 2)

Im zweiten Schritt der Entwicklung wurde die zweite Version des El-Mo-Apparats entworfen. Dieser soll nun einen eigens entworfenen Morsecode-Streifen mit einem Lichtsensor auswerten.

1) Der Morsecodestreifen: Der Morsestreifen, siehe Abbildung 4, besteht aus einer schwarzen und weißen Abfolge. Die kürzeren Abschnitte sind immer einen Zentimeter lang, die längeren genau zwei Zentimeter. Die kurzen schwarzen Stellen werden vom El-Mo-Apparat als Punkte, die langen als Striche in dem Morsealphabet interpretiert. Die kurzen weißen Stellen entsprechen Pausen, die zwischen den einzelnen Zeichen (Punkten und Strichen) benötigt werden, damit später ein Buchstabe ausgegeben werden kann. Ohne die langen weißen Abschnitte können die Buchstaben nicht voneinander getrennt abgelesen werden.

2) Konstruktion: Für die zweite Konstruktion wurde ein NXT-Lichtsensor und ein NXT-Motor verwendet, wie in Abbildung 5 gezeigt wird. Der Motor dreht ein Rad, das den Morsecode-Streifen auf der dafür vorgesehenen Bahn unter dem Lichtsensor durchführt. Um Probleme beim Auslesen des Streifens zu vermeiden, bewegt sich der Motor in bestimmten Abständen vor und zurück, um den Lichtsensor immer genau

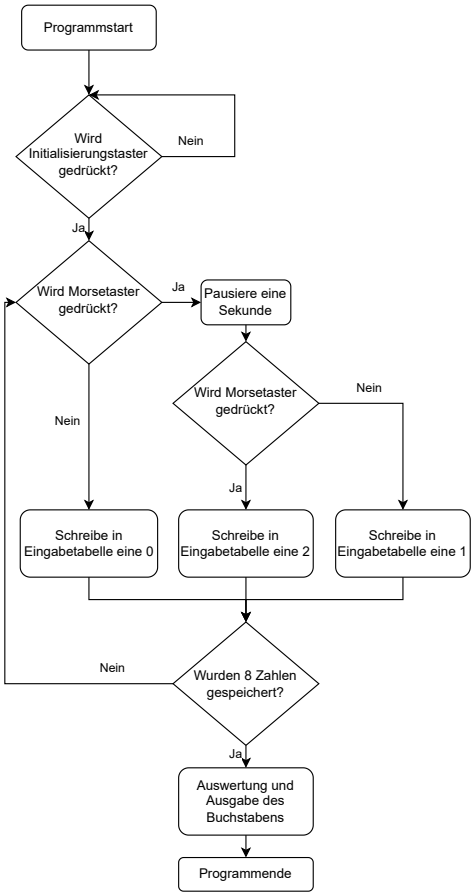


Abbildung 3. Programmablaufplan des El-Mo-Apparates (Version 1)

Abbildung 4. Zwei Morsecode-Streifen (Oberer: „O B T“, Unterer: „X O R“)

in der Mitte der hellen und dunklen Stellen zu positionieren, sodass fehlerfrei abgelesen werden kann. Der El-Mo-Apparat (Version 2) verfügt über eine Zahnradübersetzung, die die Genauigkeit beim Einziehen des Streifens verbessert.

Der Lichtsensor ist senkrecht zur Bahn und somit zum Morsecode-Streifen angebracht. Er wird im 2. Modus, wie im Abschnitt „Welche Sensoren eignen sich für einen Morseapparat?“ bereits erklärt, betrieben. Um das Umgebungslicht so gut wie möglich fernzuhalten, ist im El-Mo-Apparat noch eine Abschirmung aus LEGO-Steinen um den Lichtsensor gebaut.

3) Programmierung: Der Programmablaufplan des El-Mo-Apparats ist in Abbildung 6 dargestellt. Das Grundkonzept der zweiten Version besteht darin, dass der Morsecode-Streifen immer um eine bestimmte Strecke eingezogen wird. Daraufhin überprüft der Lichtsensor, ob sich unter diesem ein heller oder dunkler Abschnitt befindet. Um dies mit höchster Präzision zu gewährleisten, ist am Anfang des Programms eine Einspannungssequenz erforderlich. Der Morsecode-Streifen wird so

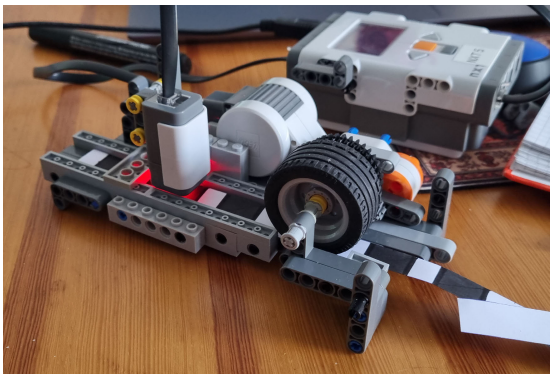


Abbildung 5. El-Mo-Apparat (Version 2)

weit unter den Lichtsensor gefahren, dass dieser sich über dem Anfang des Streifens befindet. Wenn sich der Lichtsensor über einem hellen oder dunklen Abschnitt befindet und das dazugehörige Signal an den Computer weitergeleitet hat, untersucht er die nächste Stelle. Die folgende Überprüfung entscheidet, ob sich auf dem Morsecode-Streifen eine kurze oder lange Stelle befindet. Abhängig von der Länge und der Helligkeit (hell oder dunkel) wird ein Wert in der Eingabetabelle gespeichert.

Das Programm hat zwei Möglichkeiten, den Einlesevorgang zu beenden. Entweder wurden bereits acht Werte eingespeichert oder der Lichtsensor hat dreimal hintereinander einen hellen Abschnitt erkannt. In beiden Fällen wird die Abbruchsequenz eingeleitet, die den Morsecode-Streifen aus der Konstruktion herausfährt. Anschließend wird die Eingabetabelle mit den gespeicherten Werten ausgewertet und ein einzelner Buchstabe ausgegeben.

C. Auswertung der Eingaben und Ausgabe des Buchstabens

Die Auswertung der Eingabetabelle und die Ausgabe des Buchstabens erfolgt in einer externen Funktion, die am Ende des Programms aufgerufen wird. Zunächst wird die erste gespeicherte Zahl mit 10^6 multipliziert. Anschließend wird die nächste Stelle in der Tabelle mit einer Zehnerpotenz, deren Exponent immer um eins verringert wird, multipliziert und auf die erste Zahl addiert. Beispielsweise ist ein „X“ die Zahl 2010102 (lang schwarz, kurz weiß, kurz schwarz, kurz weiß, kurz schwarz und so weiter). Dies geschieht entweder, bis eine „-1“ (ein langer weißer Abschnitt) in der Tabelle steht oder sieben Zahlen ausgewertet und aufeinander addiert wurden. Die letzte und achte Stelle in der Tabelle ist immer eine „-1“, sofern sie nicht bereits zuvor aufgetreten ist.

IV. Ergebnisdiskussion

A. Endergebnis

Beide Versionen des El-Mo-Apparats funktionieren nach mehreren Problemlösungen und Tests einwandfrei. Auch wenn das ursprüngliche Morsegerät mit elektromagnetischer Induktion betrieben wird und dies mit den LEGO-Bausteinen nicht nachahmbar ist, wurde durch die zwei verschiedenen Versionen dennoch ein funktionierendes und gutes Ergebnis erreicht.

Ursprünglich war die Planung, dass mit dem El-Mo-Apparat ganze Wörter und sogar Sätze ein- und ausgegeben werden

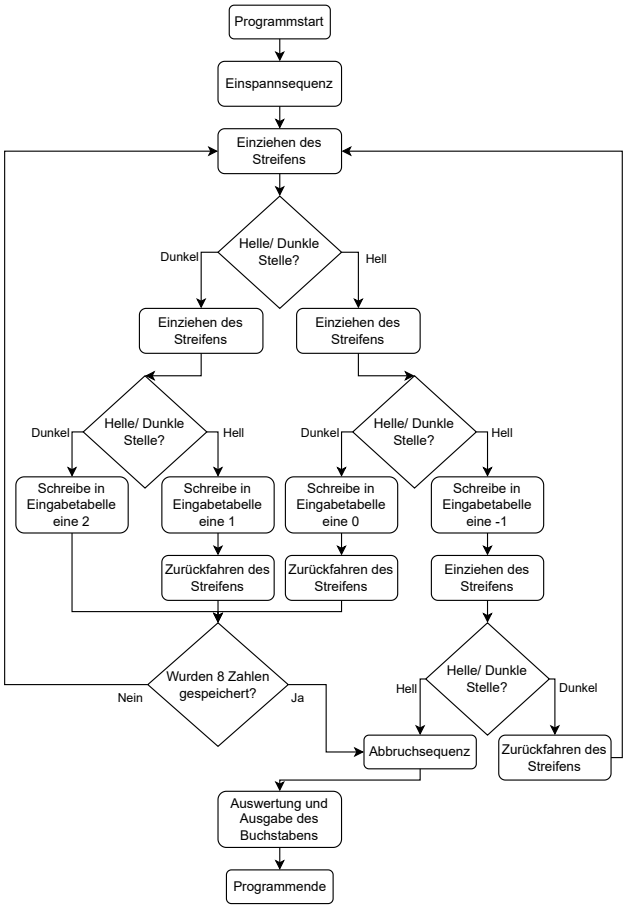


Abbildung 6. Programmablaufplan des El-Mo-Apparates (Version 2)

können. Da es aber schwierig ist, mit den Sensoren eine Abbruchsequenz bzw. ein Leerzeichen zu erzeugen, wurde diese Idee verworfen.

Die erste Version des Apparats mit dem Tastsensor erzeugt Signale, die an den Computer weitergeleitet und dort durch MATLAB in Buchstaben umgewandelt werden.

Die zweite Version liest die verschiedenen unterschiedlich hellen bzw. dunklen Abschnitte fehlerfrei ein und codiert daraus den dazugehörigen Buchstaben des Morsealphabets.

B. Probleme

Ein immer wiederkehrendes Problem bei der ersten Version sind die Pausen, die das Programm zwischen den einzelnen Tastendrücken machen soll. Einerseits sollen die Pausen so lange wie möglich sein, damit jemand, der sich mit der Morsesprache nicht gut auskennt oder den El-Mo-Apparat zum ersten Mal benutzt, genügend Zeit hat. Andererseits sollten die Pausen so kurz wie möglich gehalten werden, damit keine große Zeitspanne benötigt wird, um ein neues Zeichen eingeben zu können (siehe Abbildung 3).

Die meisten Probleme verursacht jedoch die Umsetzung der zweiten Version. Hierbei gab es gleich mehrere Problematiken, die auf verschiedenen Ebenen stattfanden. Zunächst geht eine Störung von externen Lichtquellen aus, die den Lichtsensor beeinflussen und damit die Ergebnisse bzw. die zu messenden Abschnitte verfälschen. Die Behebung des Problems bestand, wie oben beschrieben, durch eine kleine Abschirmung, die um den Lichtsensor angebracht wurde.

Des Weiteren muss der Morsecode-Streifen sehr präzise sein, weil schon Abweichungen von Millimetern zu Veränderungen und falschen Messwerten führen können.

Die größte Herausforderung bestand bei dem Drehwinkel des NXT-Motors. Der Drehwinkel des Motors ist sehr klein und damit stark fehleranfällig. Mit den LEGO-Bausteinen ist es kaum bis gar nicht möglich, konstant denselben kleinen Winkel bei jeder Drehung zu erreichen. Ein konstanter Winkel ist wichtig, damit der Morsecode-Streifen gleichmäßig und kontinuierlich um einen Zentimeter eingezogen wird. Somit waren zu Beginn die übertragenen Werte in der Eingabetabelle durch falsches Ablesen des Lichtsensors inkorrekt. Um dieses Problem zu lösen, wurde eine Zahnradübersetzung eingebaut. Dadurch kann ein größerer, nahezu konstanter Drehwinkel des Motors, bei gleichbleibendem Einzug des Morsecode-Streifens, erreicht werden. Dadurch wird die Fehleranfälligkeit reduziert und ein konstantes Einziehen ist gewährleistet.

V. Zusammenfassung und Fazit

Bei diesem Projekt wurde ein Morsegerät aus LEGO-NXT-Bausteinen nachgebaut und neu programmiert. Dabei entstanden zwei Versionen des El-Mo-Apparats, die jeweils unterschiedliche Eingabe-Mechanismen verfolgen. Bei der ersten Version wird ein Tastsensor genutzt, um so mit dem Drücken des Sensors Morsezeichen zu erstellen. Die zweite Version besteht aus einem Lichtsensor, der aufgrund von Helligkeitsstufen verschiedene Abschnitte erkennt und diese anschließend an einen Computer weiterleitet, sodass aus den abgelesenen Werten Buchstaben entstehen können. Dabei entstanden verschiedene Probleme, die allerdings durch verschiedene Lösungsansätze behoben werden konnten. Insgesamt wurde das Ziel erreicht, einen bzw. zwei fähige Morseapparate zu erstellen. Zukünftig könnten die beiden Versionen der El-Mo-Apparate so verbessert werden, dass sie sowohl ganze Sätze als auch Ziffern einlesen und auch ausgeben können. Des Weiteren könnte ein zweiter El-Mo-Apparat gebaut werden, der eine Verbindung per Bluetooth zu einem anderen Apparat hergestellt. Somit könnte eine direkte Kommunikation zwischen zwei El-Mo-Apparaten ermöglicht werden. Zusammenfassend lässt sich sagen, dass das Praktikum eine sehr spannende und lehrreiche Zeit war. Durch die vielen Herausforderungen waren Spaß, Verzweiflung und Lernerfahrungen eng miteinander verbunden.

Literaturverzeichnis

[1] Wikipedia, the free encyclopedia: *Morsecode*. https://de.wikipedia.org/wiki/Morsecode

[2] Sauber, Wolfgang: *Tastsensor*. https://commons.wikimedia.org/wiki/File:Höritz_Museum_-_Morsegerät.jpg. Version: Februar 2009

[3] Strecker, Karl: *Der Morseapparat, In: Die Telegraphentechnik: Ein Leitfaden für Post- und Telegraphenbeamte*. Berlin, Heidelberg: Springer Berlin Heidelberg, November 1917. https://doi.org/10.1007/978-3-642-92301-2_9

Der elektrische Morseapparat, kurz: El-Mo

Julian Sebastian Wenzel, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In dem folgenden Dokument werden die Planung, die Konstruktion und der Bau, sowie die MATLAB-Programmierung eines neu gedachten Morsegerätes dargelegt. Im Folgenden werden zwei Geräte vorgestellt, da es zwei Konstruktionen unterschiedlicher Machart und Programmierteile gibt. Grundsätzlich funktionieren beide Geräte derart, dass sie äußere Signale erhalten, auswerten und diese decodiert über den Bildschirm ausgeben.**
Das erste der beiden Geräte wurde über zwei Taster angesteuert und verglich deren Werte. Das zweite, welches der tatsächliche El-Mo-Apparat –im Folgenden nur noch als „El-Mo“ bezeichnet– ist, las einen Papierstreifen mit einer Strichcodierung ein.

***Schlagwörter*—Codierung, Elektrotechnik, LEGO Mindstorms, Morsen, Projektseminar**

I. Einleitung

DIE heutzutage vorherrschende Meinung ist, dass in der Welt der Nachrichtenkommunikation der Morsecode, seiner extensiven Natur wegen, keinen Platz mehr hat. Doch sollte er deswegen ganz in Vergessenheit geraten? Immerhin ist er nicht digital derart abzufangen, wie es zum Beispiel E-Mail und SMS sind, kann also eine gewisse Art der digitalen Sicherheit bieten. Noch dazu kommt, dass der Morsecode keine ausgedehnten Apparaturen und Geräte benötigt, kann also in Situationen Verwendung finden, in denen zum Beispiel Platz eine große Rolle spielt.
Um also diese Art der Kommunikation am Leben zu erhalten, wird im Rahmen des Projektseminars ein Prototyp für eine neue Möglichkeit des Einlesenst des Morsecodes entwickelt. Diese wäre, dass die zu übermittelnde Botschaft nicht per Kontaktschließung eines Stromkreises verschlüsselt wird, sondern — einem Barcode ähnlich – von einem Papierstreifen automatisch abgelesen werden kann. Offensichtlich ist dafür ein Zusammenspiel von Sensorik und Motoren von Nöten. Die Herausforderung liegt nun darin, die Präzision in dieser Zusammenarbeit zu gewährleisten.

II. Vorbetrachtungen

A. Morsecode

Der erste Schritt für dieses Projekt war, die Funktionsweise des Morsens zu verstehen. Grundlegend für das Morsen ist, dass „zwei verschiedene Zustände (wie etwa Ton oder kein Ton) eindeutig und in der zeitlichen Länge variierbar dargestellt werden können. Dieses Übertragungsverfahren nennt man Morsetelegrafie.“ [1] Daraus folgt, dass mindestens zwei Werte zu beachten sind, sollte dieses Verfahren verwendet werden. Theoretisch könnten Zusatzsignale vereinbart werden, da es hier keine Beschränkungen gibt. Betrachtet man nun das Morsealphabet, so fällt auf, dass die einzelnen Buchstaben nach ihrer Häufigkeit im englischen Sprachgebrauch codiert sind. Demzufolge sind „E“ und „T“ am simpelsten zu versenden, da sie jeweils nur aus einem Zeichen bestehen.

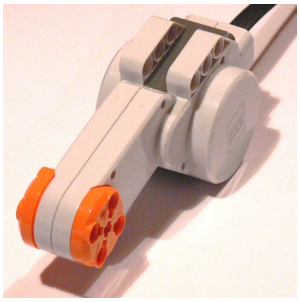
DOI: 10.24352/UB.OVGU-2024-010

B. Erprobung der Umsetzungsmöglichkeiten

Der zweite Schritt dieses Projektes bestand darin, zu testen, ob sich die Idee erst einmal mit einer sensorlosen Konstruktion umsetzen ließe, und nur auf User-Input angewiesen wäre. Als hier die Problemstellen und deren Lösungen identifiziert waren, konnte in die nächste Phase übergegangen werden. Nach dem Bekanntmachen mit der Programmierumgebung Matlab und des LEGO-Mindstorms-Sets, wurde es klar, dass die Programmierung den größten Teil der Arbeit ausmachen würde. So wurden zuerst Tests zur richtigen Verwendung von Lichtsensoren (siehe Abbildung 1a),Tastern (siehe Abbildung 1b) und Motoren (siehe Abbildung 1c) durchgeführt, so dass die spätere Verwendung darauf fußen konnte.

(a) Lichtsensor

(b) Tastsensor



(c) Servomotor

Abbildung 1. Verwendete Sensoren und Motoren im Verlauf des Projekts von LEGO Mindstorms

III. Umsetzung

A. Prototypentwicklung

Es wurde also zuerst ein durch Taster angesteuerter Morseapparat gebaut. In dessen erster Entwicklungsphase wurde versucht, die Eingabesignale, derer es zwei gab, durch nur

einen einzigen Taster zu erreichen. Die Unterscheidung der langen und kurzen Signale sollte durch eine Stoppuhrfunktion erreicht werden, sodass einzig die Länge des Signals ausschlaggebend sein sollte. Da dies scheiterte, wurde der Ansatz der Vergleichsmessung verfolgt; ein weiterer Taster musste betätigt werden, damit die Signale aufgenommen werden konnten. Mit dieser Methodik wurden erfolgreich die ersten Buchstaben übermittelt.

B. Programmierung

Um die Werte der Taster abzuspeichern, wurde ein Array aus Einsen erstellt. Kamen nun Signale über die Taster ein, unterschieden sich diese im Programm in ihren Werten. Bei langen Signalen wurde im Array eine Eins addiert, bei kurzen änderte sich nichts, und die Trennungen zwischen diesen beiden wurde mit einer 0 dargestellt. Der Ablauf einer Programmschleife, wird in der Abbildung 2 verdeutlicht:

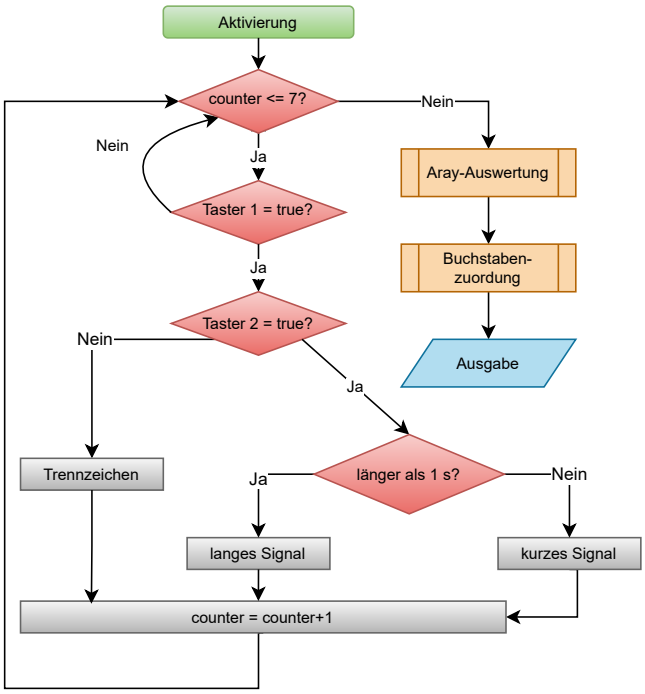


Abbildung 2. Flussdiagramm für den Prototypen von El-Mo

Wie zu sehen ist, wird in Abbildung 2 auch die Entschlüsselung der Array-Werte dargestellt. Diese erfolgte über eine for-Schleife, in der eine Variable rekursiv mit Werten über eine Summation befüllt wurde. Im folgenden wird die dafür zuständige Gleichung erläutert, dabei wird die Variable mit dem Wert als „W“ bezeichnet und der Array-Wert bei der i-ten Iteration „A(i)“:

$$W = W + A(i) \cdot 10^{(7-i)} \tag{1}$$

Um es etwas zu verdeutlichen, hier ein Beispiel: Hat man den Buchstaben „N“ gemorst, steht in dem Array die Zahlenfolge 2,0,1,0,0,0,0. Durch die Anweisung wird „W“ zuerst auf $2 \cdot 10^6$ gesetzt. In dem dritten Durchlauf wird dann dazu noch $1 \cdot 10^4$ addiert, so dass das Ergebnis 2010000 wäre. Diese Zahl wäre der Stellvertreter für den Buchstaben „N“. Danach wird in einer switch-Anweisung dieser Zahl der korrekte Buchstabe zugeordnet und auf der MATLAB-Oberfläche ausgegeben.

C. Mechanische Aufgaben

Der eigentliche El-Mo, hatte zwei große mechanische Hürden zu meistern. Zum einen erfolgte die Signaleingabe nun nicht mehr über Taster, sondern über Papierstreifen, welche mit einer Art Barcode beschrieben waren. Doch damit El-Mo mit diesen arbeiten konnte, mussten die einzulesenden Streifen sehr genau gezeichnet sein. Zum anderen hat der Automatismus zum Einziehen des Streifens vorgeschriebene Eckpunkte einzuhalten. Die Breite der Segmente des Codes auf dem Papier, wurde in Zentimeterschritten gemessen. Darum musste auch El-Mo in diesen Schritten arbeiten können. Durch eine begrenzte Auswahl an Radgrößen musste also die Programmierung dafür sorgen, dass der Motor dieses Rad im richtigen Takt um die richtige Winkelzahl bewegt. Durch Messungen und anschließende Rechnungen wurde es irgendwann klar, dass der Motor sich um 24° zu drehen hat, damit El-Mo seine Aufgabe zufriedenstellend lösen konnte.
Erschwernd zu der Präzesions-Problematik beim Einlesen des Strichcodes kam, dass der Papierstreifen sicher und ohne zu verrutschen eingezogen werden konnte. Um dies zu gewähleisten, wurde die Gummierung des Rades mit verbaut. Daraus entstand jedoch ein sehr hoher Reibungswiderstand zwischen Gummi und Unterlage. Infolge dessen hatte der Motor immense Schwierigkeiten damit, die nötige Leistung aufzubringen, den Streifen um die richtige Distanz zu bewegen. Dieses Problem wurde mit einer Zahnradübersetzung im Verhältnis von 1:6 gelöst. So konnte die Motorsleistung um das Sechsfache gesteigert werden, womit der Reibungswiderstand ausgeglichen werden konnte.

D. Codierungs Normierung

Damit El-Mo mit der Strichcodierung arbeiten konnte, mussten bestimmte Normen erstellt werden. Zum einen galt es, festzulegen, dass der Beginn jedes Strichcodes mit einem weißen, einen Zentimeter breiten Segments beginnt. Das spielt bei dem Einziehen des Streifens eine wichtige Rolle, da dieser Prozess mit der Änderung der gemessenen Helligkeit arbeitet (s. Abschnitt E: „El-Mo´s Programm“). Zum anderen musste klar festgelegt werden, wie breit die Segmentierung der Streifen sein durfte. Da dies willkürlich gewählt werden konnte, wurde beschlossen, dass die Länge der Signale mit Zentimetereinheiten arbeiten soll. Einen Überblick zu den Normierungen findet sich in Tabelle 1 „Übersicht zur Codierung der Papierstreifen“. Wie diese in der Praxis umgestzt wurden, ist in Abbildung 3 zu sehen.

Tabelle 1
Übersicht zur Codierung der Papierstreifen

Signalart	Farbe	Länge in cm	Array-Wert
langes Signal	Schwarz	2	2
kurzes Signal	Schwarz	1	1
Trennzeichen	Weiß	1	0

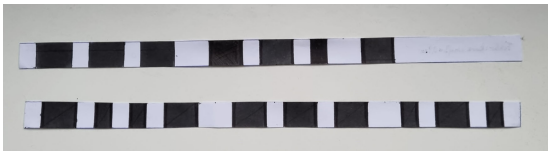


Abbildung 3. Beispiel eines für El-Mo lesbaren Strichcodes
oben: S, G, T
unten: X, O, R

E. Grundsätze der Funktionsweise von El-Mo

Viele Lösungsansätze, die bei der Bearbeitung des Prototypen entwickelt wurden, konnten tatsächlich in die Programmierung von El-Mo übertragen werden. Einzig die Aufgabe des automatischen Einziehens ist gänzlich neu gewesen. Nach dem Positionieren des Streifens wird der Motor aktiviert und das Papier wird Richtung Sensor geschoben. In einer Kontrollschleife wird geprüft, ob der Sensor etwas Helles detektiert. Ist dies der Fall, so wird der Motor kurz gestoppt. Anschließend wird der Streifen so platziert, dass der Sensor über der $5\,\mathrm{cm}$-Marke, vom Beginn des Streifens gemessen, schwebt.
Von hier an wird der Papierstreifen um die Länge von einem Zentimeter eingezogen, die Helligkeitsstufe des Segments geprüft und mit der Helligkeitsstufe des zuvor eingelesenen Segments verglichen. Sind diese identisch (Schwarz-Schwarz), so wird ein langes Signal detektiert und eine „2“ in das Array eingetragen. Sind sie nicht identisch – entweder Schwarz-Weiß oder Weiß-Schwarz – so wird im ersten Fall ein kurzes Signal eingelesen und eine „1“ in das Array eingetragen und im zweiten Fall ein Trennzeichen, bei dem eine „0“ in das Array geschrieben wird.
Zum Schluss wird der Streifen erneut positioniert, damit der nächste Vergleich beginnen kann. Ist die maximale Anzahl von Zeichen erreicht, wird die Prüfschleife beendet. Das Auslesen des Arrays und die Buchstabenzuordnung konnte eins zu eins aus dem Prototypen übernommen werden.

IV. Ergebnisdiskussion

Das Ergebnis von zwei Wochen intensiver Arbeit lässt sich wie folgt zusammenfassen: Der Apparat erkennt problemlos die Segmente des Codes und gibt den entschlüsselten Buchstaben korrekt aus. Eine Momentaufnahme dieses Prozesses ist in Abbildung 4 zu sehen.

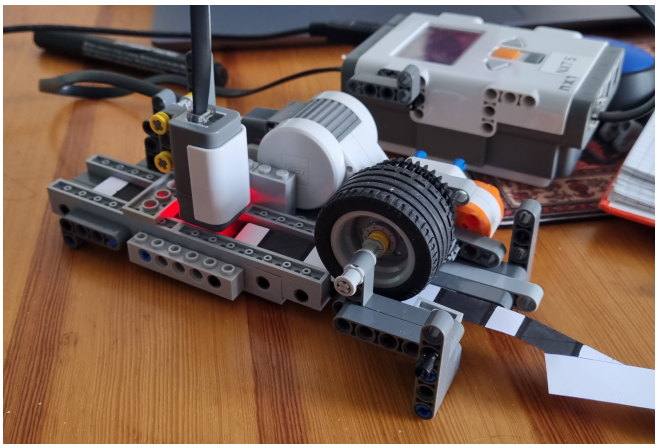


Abbildung 4. El-Mo mit Lichtsensor, während ein Streifen eingelesen wird. Hinter dem Rad ist die Zahnradübersetzung zu erkennen.

Die größten unerwarteten Herausforderungen traten weniger im Programmierteil als in mechanischen Ungenauigkeiten zutage. Bis kurz vor der Fertigstellung zog der Motor immer etwas zu viel des Papiers ein, was nach einigen Segmenten zum falschen Einlesen des Codes führte. Letztendlich bestand die Lösung darin, zu erkennen, dass die Fehlerkorrektur über das Anpassen der Gradzahl, um die sich der Motor zu drehen hatte, erfolgen konnte, sowie in der Erkenntnis, dass eine Unterstützung mit einer Zahnradübersetzung nötig war.

V. Zusammenfassung und Fazit

Abschließend ist zu sagen, dass unter den gegebenen Bedingungen die erbrachte Leistung optimal war, wobei auch über Potential zur Verbesserung nachgedacht wurde. Sollte El-Mo zu einem späteren Zeitpunkt weiterentwickelt werden, so wäre es überlegenswert, zu versuchen, die Ausgabe der Buchstaben nicht über MATLAB erfolgen zu lassen. Vielleicht bietet sich hier eine elegantere Lösung, möglicherweise sogar das Koppeln mehrerer NXT, die über Bluetooth mit einander verbunden sind.

VI. Literaturverzeichnis

[1] Morsecode, Wikipedia, 2024.02.14, 13:18 Uhr
https://de.wikipedia.org/wiki/Morsecode

VII. Anhang

Für ein besseres Verständnis der Unterschiede zwischen dem Code des Prototypen und El-Mo lässt sich folgendes sagen: Vom Prinzip her ähneln sich die beiden, da es in beiden gleichende Abschnitte für die Signalarten gibt, mit dem Unterschied, wie diese eingelesen werden. El-Mo jedoch arbeitet auch mit einem Motor und hat sehr viel mehr Vergleichsschleifen, die zu einem viel größeren Code-Gebilde führen, als es beim Prototypen der Fall ist. Dies ist für den visuell Ausgelegten möglicherweise besser mit Hilfe der Abbildung 5 nachzuvollziehen

Da es wahrscheinlich seltsam erscheinen mag, warum ein Flussdiagramm vom Prototypen aber nicht von El-Mo im Dokument auftaucht, so ist dieses als Abbildung 5 hier zu finden.

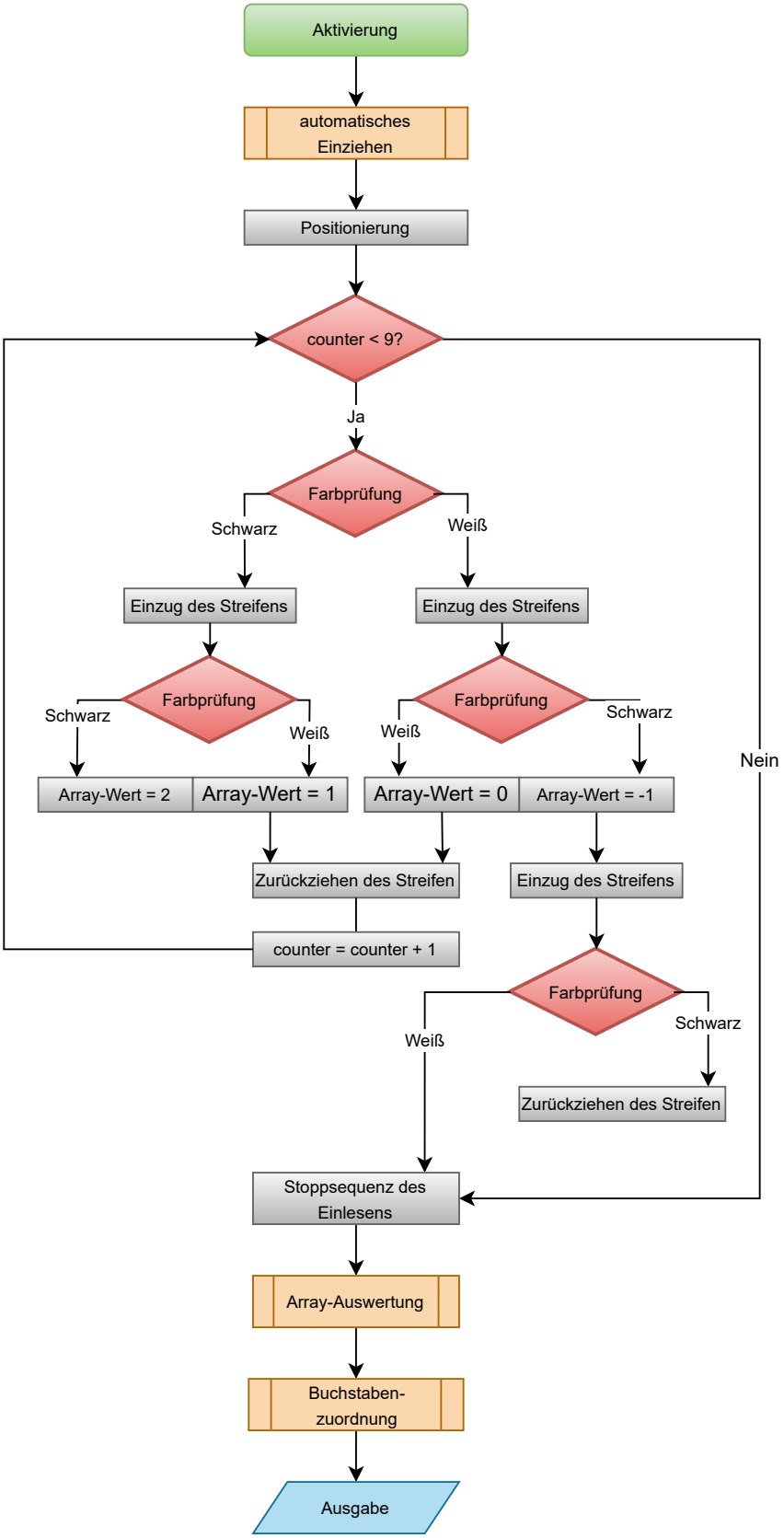


Abbildung 5. Flussdiagramm des Codes von El-Mo

Oszilloskop-Plotter

Ahmad Alhamid, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*— Das Projektseminar Elektrotechnik und Informationstechnik wird jedes Jahr an der Otto-von-Guericke-Universität durchgeführt. In diesem Jahr wurde ein Oszilloskop-Plotter als Kunstprojekt angefertigt. Dieses Projekt hat keinen technischen Nutzen und löst kein Problem. Es ist ein Kunstprojekt und diese Kunst nutzt das Wissen über Elektrizität, in dem eine Person wie eine Etch a sketch zeichnen kann, aber mit Hilfe von Spannungsänderung in Potentialmeter**

***Schlagwörter* Potentiometer, MATLAB, Osziloskop, Gleichstrommaschine, LEGO Mindstorms, Spannung.**

I. Einleitung

Das LEGO Mindstorms-Oszilloskop-Plotter-Projekt ist eine Demonstration der Verwendung von LEGO Mindstorms Robotik, um das klassische Etch a Sketch-Spielzeug nachzubauen. Das Projekt beinhaltet die Verwendung von zwei Potentiometern, die mit einem Oszilloskop verbunden sind, wobei jedes LEGO Motor die Bewegung eines Potentiometer steuert. Die mit Zahnrädern ausgestatteten Motoren bewegen die Potentiometer auf eine Weise, die dem Malmechanismus eines Etch-A-Sketch ähnelt.

II. Vorbetrachtungen

Für das Projekt werden LEGO-Mindstorms-Komponenten wie Motoren, Getriebe, LEGO-Teile und ein programmierbarer Steuerungscomputer verwendet. Mit Potentiometern wird die Position eines Stiftes Rotation auf der x- und der anderer y-Achse gesteuert, um die Bewegung eines Etch a Sketchs zu imitieren. Während sich die Motoren drehen, passen sie die Positionen der Potentiometer an und ermöglichen so das Zeichnen von Formen und Mustern auf dem Oszilloskop, indem sie die Spannungsänderung im Potenzialmesser messen. Das Projekt kombiniert Elemente der Robotik, Elektronik und Programmierung, um die nostalgische Erfahrung der Verwendung eines Etch-A-Sketchs in einem modernen Kontext nachzubilden.

Die Integration eines Oszilloskops ermöglicht es, die elektrischen Signale in Echtzeit zu visualisieren und das Verhalten des Systems zu analysieren. Dies hilft, ein tieferes Verständnis der grundlegenden physikalischen Prinzipien zu erlangen und ermöglicht die Weiterentwicklung der technischen Fähigkeiten.

A. *Motivation*

Die Motivation war, dass ein kleines Projekt bereits gemacht wurde und in den sozialen Medien von Dr. Mathias Magdowski hochgeladen wurde und die Potentiometer manuell mit der Hand gedreht wurden und die Idee war, dies mit einigen NXT-Komponenten und Programmcodes zu automatisieren. Dies wird helfen, das Projekt zu nutzen, um viel stabilere und komplexere Zeichnungen zu zeichnen .

B. *LEGO NXT*

LEGO Mindstorms NXT stellt eine vielseitige Robotikplattform dar, konzipiert und entwickelt von der renommierten LEGO-Gruppe. Das NXT-System besteht aus einer Reihe von Komponenten, die es den Benutzern ermöglichen, komplexe robotische Systeme zu erstellen. Mit einem Fokus auf technische Präzision bietet das LEGO eine visuelle Programmierumgebung, die es den Nutzern ermöglicht, maßgeschneiderte Programme zu erstellen. Dies geschieht durch das Verbinden von Blöcken, die eine Vielzahl von Aktionen und Steuerungsmechanismen repräsentieren [1]. Diese Blöcke ermöglichen es die Funktionalität des Roboters in einem intuitive Format zu konfigurieren und anzupassen.

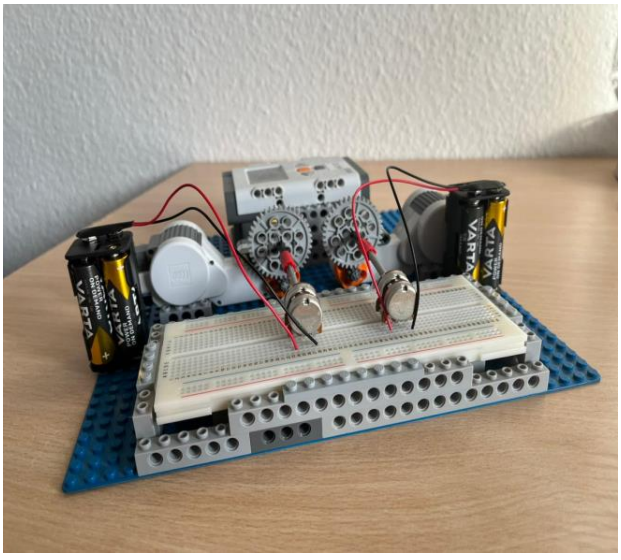


Abbildung 1 Aufbau des Oszilloskop-Plotters

Abbildung 1 zeigt, wie der Hardware-Teil des Projekts aufgebaut und miteinander verbunden ist.

DOI: 10.24352/UB.OVGU-2024-011

III. KONSTRUKTION UND REALISIERUNG

Um dieses Projekt zu realisieren, werden einige mechanische Teile benötigt, die aus der LEGO-Mindstroms-Box stammten. Die verwendeten Hardwarekomponenten umfassten verschiedene Zahnräder, zwei Motoren und andere Teile zur Verbindung aller Komponenten. Zusätzlich werden auch einige externe Komponenten wie ein Breadboard, Batterien und ein Potentiometer bezogen. Außerdem wird ein Oszilloskop um die Zeichnung darzustellen benutzt.

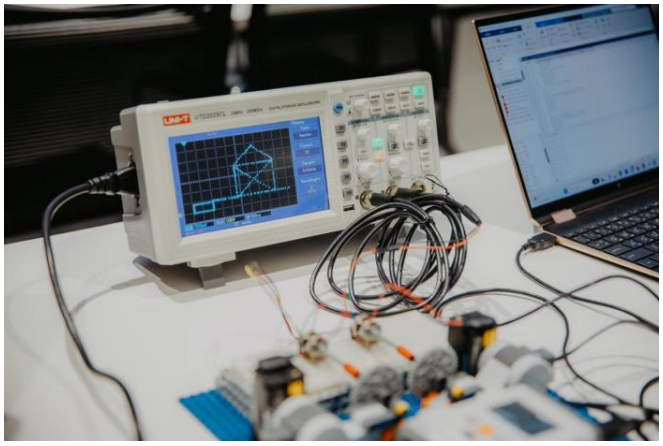


Abbildung 2 Vom Oszilloskop-Plotter gezeichnetes "Haus des Nikolaus"

A. Zahnräder

Ein 8-Zahn-Zahnrad für jeden Motor und ein 40-Zahn-Zahnrad für jeden Potentiometer wurden verwendet. Dies reduzierte die Rotation. Dadurch verringert sich das Übersetzungsverhältnis, da das Potentiometer nicht eine Komplette Drehung von 360 Grad hat. Das Oszilloskop wurde verwendet, um die Spannungsänderung in jedem Potentiometer zu messen.

B. Motoren

Die NXT-Motoren aus dem LEGO-Mindstorms-Set wird verwendet. Diese Motoren sind speziell für LEGO-Roboter mit 9 Volt-Gleichstrommaschine entwickelt und bieten eine gute Leistung und Präzision [2].

C. Oszilloskop

Das Oszilloskop wurde verwendet, um die Spannungsänderung in jedem Potentiometer zu messen.
ein Potentiometer ist an Kanal 1 und das andere ist an Kanal 2 angeschlossen.
Die Einstellungen des Oszilloskops wurden so konfiguriert, dass Kanal 1 auf der x-Achse und Kanal 2 auf der y-Achse dargestellt werden.

D. Programmierung

Um das Projekt zu programmieren, wurde MATLAB mit der RWTH-Toolbox verwendet. Die RWTH Toolbox stellt verschiedene Befehlezur Verfügung [3]. Mit diesen werden die Sensoren und Motoren in Betrieb genommen. Das Programm wird in MATLAB geschrieben und an den programmierbaren Computer NXT gesendet. Zuerst wurde das Projekt gebaut und alle Komponenten wurden fixiert Wie in der Abbildung 2 dargestellt, dann wurde das Programm geschrieben. Im Programm wurde zunächst ein Code geschrieben, um den Potentiometer in den Anfangszustand zu setzen, da der Potentiometer keine freie Rotation hat. Der Code wurde als Anweisung an jeden Motor geschrieben, sich um einen bestimmten Winkel zu drehen, bis der gemessene Startpunkt erreicht ist. Dann wurde eine weitere Funktion geschrieben, um beide Potentiometer in einen bestimmten Punkt, genannt Startzustand im Oszilloskop, zu bewegen. Dann wurden beide Funktionen dem Hauptcode hinzugefügt, der den Code für die zu zeichnende Figur enthält. Die zu zeichnende Figur wird als Grafikfunktion in MATLAB geschrieben, wobei ein Motor den Befehl erhält, sich zu drehen, um einen bestimmten Winkel zu zeichnen, und sich zusammen dreht, um diagonale Linien zu zeichnen.

IV. PROBLEME UND ERGEBNISDISKUSSION

Während der zwei Wochen der Arbeit an diesem Projekt traten viele Probleme auf, aber das schwierigste Problem bestand darin, die Motoren so zu programmieren, dass sie diagonale Linien zeichnen konnten. Beide Motoren mussten gleichzeitig und mit gleicher Geschwindigkeit rotieren, was sehr komplex war, um sie richtig einzustellen und zu konfigurieren, um perfekte diagonale Linien zu erhalten. und der Grund dafür war, dass die Motoren mit einem externen Bauteil an den Potentiometern befestigt waren und es eine kleine Toleranz gab, so dass der Motor Millisekunden vor dem Potentiometer läuft. Nach vielen Versuchen und harter Arbeit wurde das Problem jedoch gelöst, und es gelang, perfekte Diagonalen zu erzeugen. Ursprünglich sollte das Projekt dazu dienen, ein Nikolaushaus zu zeichnen, aber am Ende gelang es, viele verschiedene Figuren und Kreise zu zeichnen Wie in der Abbildung 3 und 4 dargestellt, was mit der Hand manuell sehr schwierig wäre.

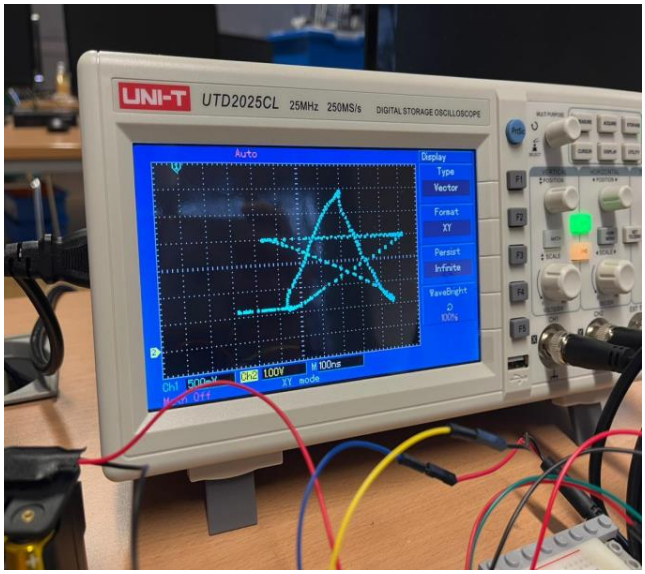


Abbildung 3 Zeichnung eines Sterns

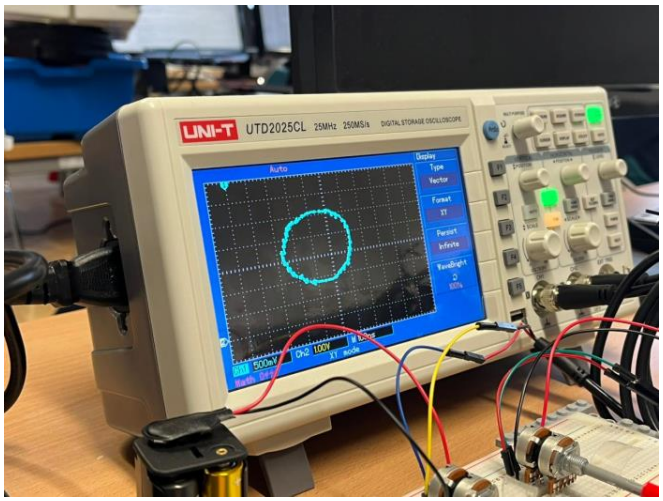


Abbildung 4 Zeichnung eines Kreis

V. Zusammenfassung

Zusammenfassend kann festgestellt werden, dass die Ziel des Projekts zufriedenstellend erreicht wurde. Hätte das Projektseminar länger als zwei Wochen gedauert wäre is möglich eine größere Version dieses Projekts gebaut und komplexere Zeichnungen kodiert hätte. Abschließend kann man sagen, dass man in diesem Projekt viel über MATLAB Programmierung und Spannungen lernt. Außerdem ist zu erwähnen, dass das LEGO-Mindstorm-Projektseminar eine gute Möglichkeit ist, Studierende mit der Programmierung in MATLAB vertraut zu machen.

Literaturverzeichnis

[1] LEGO® 9V Technic Motors compared characteristics. https://www.philohome.com/motors/motorcomp.htm
[2] WIKIPEDIA, Steckplatine (https://de.wikipedia.org/wiki/Steckplatine) Version: Januar 2024
[3] RWTH -Mindstorms NXT Toolbox (https://www.mathworks.com/matlabcentral/fileexchange/18646-rwth-mindstorms-nxt-toolbox) Version: Februar 2023

Anhang

So wurde der in Abbildung 2 dargestellte Prozess in MATLAB programmiert:

```
 clear all
handle=COM_OpenNXT();
COM_SetDefaultNXT(handle);
% program

OBJA=NXTMotor(MOTOR_A);
OBJB=NXTMotor(MOTOR_B);

% Motoren auf Anfang
disp('Motor A auf Anfang:')
anfangswertA=motor_an_anfang(OBJA,-10);
endwertA=1500+anfangswertA;
disp('Motor B auf Anfang:')
anfangswertB=motor_an_anfang(OBJB,10);
endwertB=anfangswertB-1460;

%load('endwerte.mat')

% Endwerte etwas nach "innen" verschieben
anfangformA=anfangswertA+900;
anfangformB=anfangswertB-900;
endformA=endwertA-50;
endformB=endwertB+50;

% % Haus des Nikolaus
form_x=[0,3,3,5,7,7,3,7,3,10];
form_y=[0,0,3,4.5,3,0,3,3,0,0];

% x-Werte für Motor B umrechnen
m=(endformB-anfangformB)/12;
n=anfangformB;
motorB_pos=m*form_x+n;

% y-Werte für Motor A umrechnen
m=(endformA-anfangformA)/6;
n=anfangformA;
motorA_pos=m*form_y+n;

figure(2)
plot(motorB_pos,motorA_pos)

% Motorpower
power=60;
OBJA.SpeedRegulation=1;
OBJB.SpeedRegulation=1;

OB-
JA=motor_an_position(OBJA,anfangformA,power);
OBJA.SendToNXT;OBJA.WaitFor;
OBJB=motor_an_position(OBJB,anfangformB,power
);OBJB.SendToNXT;OBJB.WaitFor;

% Schleife über alle Punkte
for n=1:length(form_x)
    tacho_limitA=motorA_pos(n)-
OBJA.ReadFromNXT.Position;
    tacho_limitB=motorB_pos(n)-
OBJB.ReadFromNXT.Position;
 % Power berechnen und auf 100 begrenzen

powerA=max([5,min([100,round(156.5902/149.523
6*abs(tacho_limitA/tacho_limitB)*power)])]);
    pow-
erB=max([5,min([100,round(149.5236/156.5902*a
bs(tacho_limitB/tacho_limitA)*power)])]);
    disp(['Power A: ',num2str(powerA)])
    disp(['Power B: ',num2str(powerB)])
    if powerA>powerB
        powerA=power;
    else
        powerB=power;
    end
```

Elektronische Kunst: Oszilloskop-Plotter in Aktion

Yeva Makarova, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Die folgende Arbeit beschreibt den Entwicklungsprozess eines Oszilloskop-Plotters. Der Prozess wurde so automatisiert, dass der Roboter mit Hilfe von zwei in MATLAB programmierten Motoren jede im x-y-Koordinatensystem darstellbare Figur auf dem Bildschirm des Oszilloskops zeichnen kann.**

***Schlagwörter*—Kunst, LEGO Mindstorms, MATLAB, Oszilloskop-Plotter, Projektseminar.**

I. Einleitung

OSZILLOSKOPE werden zum Messen, Speichern und Aufzeichnen von elektrischen Signaldaten verwendet. Normalerweise werden sie in Laboratorien sowie in der Forschung und Entwicklung zur Überwachung oder Messung und Untersuchung elektrischer Signale eingesetzt. Aber wer sagt, dass sie nicht auch in der Kunst eingesetzt werden können?

In einem speziellen Zweikanal-Oszilloskopmodus, dem so genannten x-y-Modus, wird das Signal des einen Kanals zur Ablenkung des Strahls entlang der horizontalen x-Achse und das Signal des anderen Kanals zur Ablenkung des Strahls entlang der vertikalen y-Achse verwendet.

In diesem Artikel wird beschrieben, wie man mit Hilfe von zwei Potentiometern ein bestimmtes Bild auf dem Bildschirm des Oszilloskops darstellen kann. Das Prinzip des Zeichnens ist das gleiche wie bei dem bekannten Kinderspiel ”Etch A Sketch”: Durch Bewegen des linken Potentiometers wird eine horizontale Linie gezeichnet, durch Bewegen des rechten Potentiometers eine vertikale Linie. Durch Einstellen der Geschwindigkeit und der Drehrichtung der Potentiometer kann jede beliebige Form gezeichnet werden.

II. Vorbetrachtungen

A. Idee und Beweggrund

Am Anfang stand die Idee, einen Roboter zu entwickeln, der auf dem Bildschirm eines Oszilloskops beliebige Figuren zeichnen kann, insbesondere das Nikolaushaus. Da das Oszilloskop zum Zeichnen im x-y-Modus mit zwei Potentiometern verwendet werden kann, können diese Potentiometer auch mit Motoren verbunden werden, die auf bestimmte Weise programmiert werden können. Dadurch sollte das Bild auf dem Oszilloskop von wesentlich besserer Qualität sein, als wenn es von Hand gezeichnet wird. Wenn es genug Zeit übrig wäre, könnte auch versucht werden, andere Formen zu zeichnen, z. B. einen Kreis.

DOI: 10.24352/UB.OVGU-2024-012

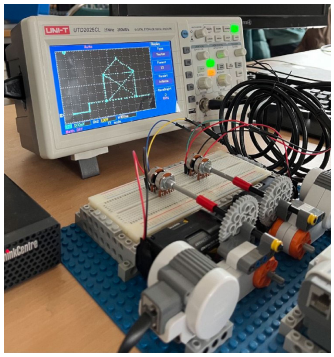


Abbildung 1. Komplette Roboterkonstruktion in Verbindung mit dem Oszilloskop

B. Bestandteile

Die Konstruktion des Roboters besteht aus folgenden Elementen (siehe Abbildung 1):

1) LEGO-Bauplatte
2) LEGO-Motor 2x
3) 8-er Zahnrad 2x
4) 40-er Zahnrad 2x
5) Potentiometer 2x
6) Batterie 2x
7) Breadboard
8) Drähte
9) LEGO-NXT-Stein
10) Oszilloskop
11) Andere LEGO-Bauteile

C. Code-Aufgabe

Ursprünglich sollte der Code die folgende Abfolge von Aktionen ausführen:

1) Kalibrierung der Motoren
2) Grenzen nach innen verschieben
3) Darstellung der Figur am Computer
4) Starten der Motoren und Zeichnen der Figur auf dem Bildschirm des Oszilloskops

III. Technische Umsetzung

In diesem Abschnitt wird die technische Realisierung des Roboters beschrieben, die aus zwei Hauptteilen besteht: der Konstruktion und dem Programmcode. Ein vereinfachtes Programmflussdiagramm ist ebenfalls in Abbildung 4 dargestellt.

A. Aufbau

Die Konstruktion des Oszilloskop-Plotters besteht aus zwei Motoren, die jeweils über zwei Zahnräder mit einem der Potentiometer verbunden sind. Diese Zahnräder regeln gleichzeitig die Drehgeschwindigkeit der Potentiometer (siehe Abbildung 2). Das erste Zahnrad ist direkt mit dem Motor verbunden und hat daher die gleiche Geschwindigkeit. Das zweite Zahnrad befindet sich neben dem ersten Zahnrad und ist mit einem Potentiometer verbunden. Das Übersetzungsverhältnis zwischen den beiden Zahnrädern beträgt 5:1, d. h. das zweite Zahnrad dreht sich fünfmal langsamer. Das bedeutet, dass sich das Potentiometer langsamer dreht und somit besser gesteuert werden kann. Alle Zahnräder sind fest mit dem Motor verbunden.

Die Batterien mit einer Spannung von jeweils 6 V werden mit Hilfe eines Breadboards parallel zu den Potentiometern angeschlossen. Bei der Installation der Batterien ist darauf zu achten, dass keine Kurzschlüsse entstehen.

Die Drähte, die den Aufbau mit dem Oszilloskop verbinden, werden in ähnlicher Weise angeschlossen, jedoch so, dass der Widerstand des Potentiometers halbiert wird (siehe Abbildung 3). Das erste Potentiometer (Abb. 1 links) ist für die x-Achse, das zweite Potentiometer (Abb. 1 rechts) ist für die y-Achse zuständig. Beim Verstellen der Potentiometer ändert sich die Spannung. Das Oszilloskop misst die Spannung und zeigt sie als Diagramm auf dem Bildschirm im x-y-Modus an.

Die gesamte Konstruktion, einschließlich des LEGO-NXT-Steins und des Breadboards, wird auf der LEGO-Bauplatte befestigt.

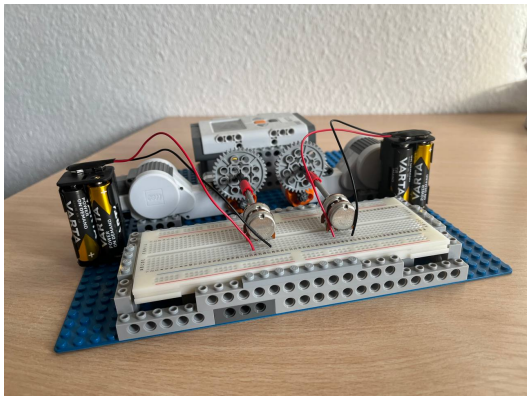


Abbildung 2. Konstruktion des Roboters, Ansicht von hinten

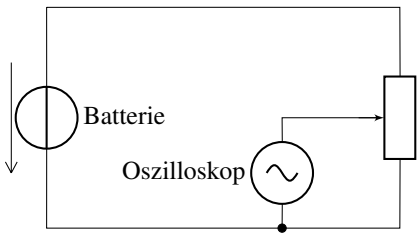


Abbildung 3. Elektrische Auslegung des Roboters

B. Programm

Zuerst muss das Programm die Anfangs- und Endpunkte kalibrieren, auf die die Potentiometer maximal eingestellt werden können. Dazu müssen die Motoren so programmiert werden, dass sie zunächst in eine Richtung bis zum Anschlag fahren und dann anhalten, ohne dass die Potentiometer aus dem Breadboard herausspringen. Dann müssen die Motoren ihre Position bestimmen und die Daten über den LEGO-NXT-Stein an den Computer senden und das Gleiche in die andere Richtung wiederholen.

Da die ermittelten Start- und Endpositionen der Motoren bei jedem Programmstart andere Koordinaten haben und eine vollständige Kalibrierung bei jedem Programmstart sehr zeitaufwendig ist, bestand die Notwendigkeit, diese Positionen schnell und allgemein zu ermitteln. Nach mehreren Kalibrierungsversuchen wurde festgestellt, dass der maximale Drehmomentbereich von Motor A bei etwa 1500 Drehmomenteinheiten und der von Motor B bei 1460 Drehmomenteinheiten liegt. Mit diesen Daten kalibriert der Code die Endpunkte wie folgt: Beide Motoren bewegen die Potentiometer nach rechts bis zum Anschlag und stoppen dann; danach werden die Endpunkte berechnet und im Code gespeichert.

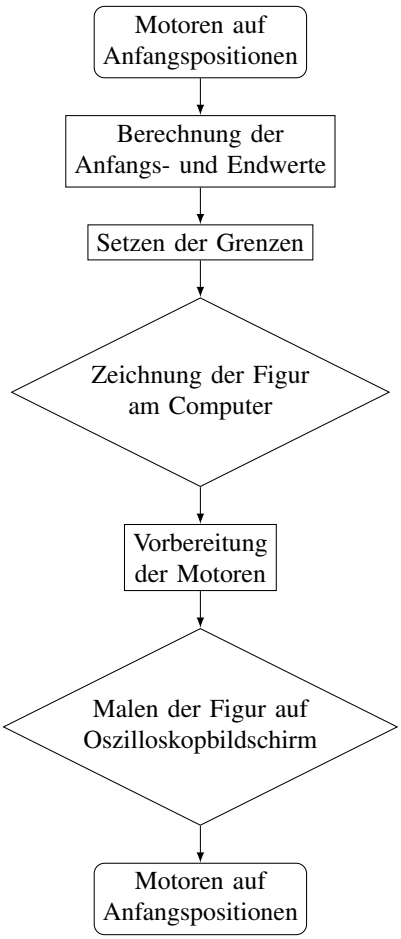


Abbildung 4. Vereinfachtes Flussdiagramm des Programms

Im nächsten Schritt wurde versucht, die erste Figur zu zeichnen, was jedoch nicht gelang. Es stellte sich heraus,

dass es Bereiche auf der Koordinatenebene des Oszilloskops gab, in denen die Figur nicht linear gezeichnet wurde. Wurde z. B. das Potentiometer in einem ausreichend großen Winkel gedreht, so wurde zunächst eine sehr kurze gerade Linie auf dem Oszilloskop gezeichnet. Dann, nach einer kleinen Drehung desselben Potentiometers, wurde eine um ein Vielfaches längere Linie gezeichnet, und dann, nach einer langen Drehung, wieder eine kurze Linie. Nach Untersuchung des Verhaltens der Oszilloskopkurve wurde ein linearer Bereich ermittelt, in dem das Bildverhalten vorhersagbar ist. Daher wird bei der Ausführung des Codes nach der Kalibrierung der Potentiometer der Arbeitsbereich des Koordinatensystems berechnet und seine Grenzen festgelegt. Anschließend wird der Lichtpunkt, mit dem die Figur auf dem Oszilloskop gezeichnet wird, an den Anfang des zulässigen linearen Bereichs verschoben.

Der nächste Schritt besteht darin, eine beliebige Form auszuwählen und sie im Koordinatensystem der x-Achse mit Maximalwerten von 6 in y-Richtung und 10 in x-Richtung zu definieren. Die Form wird dann in das Koordinatensystem des Oszilloskop-Plotters umgewandelt. MATLAB zeichnet dann die definierte Form zur Überprüfung auf den Computer und startet die Motoren.

Da die Motoren spiegelbildlich zueinander montiert wurden, musste der Softwarecode in gewisser Weise angepasst werden. Das heißt, um eine gerade Diagonale z. B. in einem Winkel von 45 Grad zu zeichnen, mussten die Motoren die gleiche Geschwindigkeit haben, aber in entgegengesetzter Richtung. Es gibt noch ein weiteres Problem: Um Diagonalen zu erzeugen, die nicht im Winkel von 45 Grad verlaufen, müssen die Motoren unterschiedliche Geschwindigkeiten haben, die voneinander und von der Position abhängen, von der sie kommen und zu der sie gehen sollen (siehe die Lösung im Anhang).

Sobald die gewünschte Figur auf dem Bildschirm des Oszilloskops angezeigt wird, setzt das Programm die Motoren auf ihre Startpunkte und ist bereit für den nächsten Lauf.

IV. Ergebnisdiskussion

Das Ergebnis ist, dass nicht nur die Figur des Nikolaushauses (Abb. 5), wie ursprünglich geplant, sondern auch ein Stern (Abb. 6a) und ein Kreis (Abb. 6b) dargestellt werden können, was ein unbestreitbarer Erfolg ist. Dies zeigt, dass der Oszilloskop-Plotter in der Lage ist, jede im x-y-Koordinatensystem angegebene Form darzustellen.

Es gibt aber auch Probleme, die noch nicht vollständig gelöst sind. An einigen Stellen drehen sich die Motoren nicht vollständig, was dazu führen kann, dass die Figur nicht zusammenhängend ist. Und obwohl wir dieses Problem zunächst lösen konnten, indem wir die Motoren so programmierten, dass sie eine bestimmte Strecke über die eingestellte Position hinaus drehen, hat sich herausgestellt, dass sich die erforderliche Strecke bei jedem Einschalten des LEGO-NXT-Steins ändert.

V. Zusammenfassung und Fazit

In zwei Wochen wurde viel Arbeit geleistet und die meisten Fehler wurden behoben. Der Oszilloskop-Plotter funktioniert recht gut und ist bereits in der Lage, einige Formen recht schön und flüssig zu zeichnen. Mit mehr Zeit wäre es möglich gewesen, einige der Zeichnungen zu verbessern, indem der oben beschriebene Fehler behoben worden wäre.

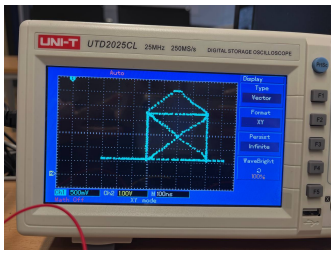
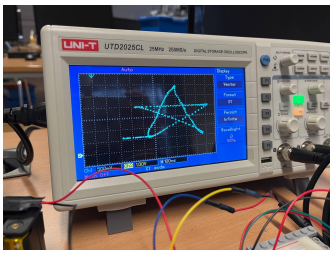
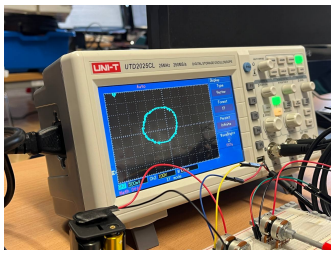


Abbildung 5. Haus des Nikolaus



(a) Stern



(b) Kreis

Abbildung 6. Andere Ergebnisse

Anhang

Einstellung der Motordrehzahl in Abhängigkeit von der Anfangs- und Endposition für das Zeichnen von Diagonalen:

```
powerA=max([5,min([100,round
(156.5902/149.5236*abs(tacho
_limitA/tacho_limitB)*power)])]);
powerB=max([5,min([100,round
(149.5236/156.5902*abs(tacho
_limitB/tacho_limitA)*power)])]);

disp(['Power A: ',num2str(powerA)])
disp(['Power B: ',num2str(powerB)])

if powerA>powerB
  powerA=power;
else
  powerB=power;
  end
```

Literaturverzeichnis

[1] Wikipedia: Oszilloskop, https://de.wikipedia.org/wiki/Oszilloskop Version: Januar 2024
[2] Wikipedia: Etch A Sketch, https://en.wikipedia.org/wiki/Etch_A_Sketch Version: Januar 2024

Flunky-Ball-Roboter

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

***Zusammenfassung*—Im Rahmen des LEGO-Mindstorms-Workshop-Projekts wurde ein Flanky-Ball-Roboter gebaut. Der Roboter basiert auf NXT3 und besteht aus Motoren, einem Farbsensor und LEGO-Teilen. Der Code wurde in Matlab erstellt. Der Roboter kann von zwei oder mehr Personen gespielt werden, kann aber auch alleine gespielt werden.**

***Schlagwörter*—LEGO Mindstorms, NXT, Projektseminar, Flunky-Ball-Roboter, Spielroboter.**

I. Einleitung

In der heutigen Welt hat jeder das Recht, sich nach einem harten Arbeitstag zu entspannen und mit Freunden Spaß zu haben. Es gibt viele Spiele, die dazu dienen, Menschen zu unterhalten und Zeit miteinander zu verbringen. Bei solchen Spielen ist es nicht wichtig, ob man gewinnt oder verliert, da sie alle dazu da sind, Freundschaften zu schließen. Allerdings gibt es auch Spiele, bei denen Gewinnen oder Verlieren direkte Auswirkungen auf die Gesundheit haben kann. Es handelt sich um das Bier-Pong. Diejenigen, die schon einmal Zeit mit Freunden bei dieser Aktivität verbracht haben, werden in diesen Worten sicherlich einen Widerhall finden: Je öfter der Ball das Ziel trifft, desto gesunder ist der Werfer.

Der Flunky-Ball-Roboter wurde entwickelt, um die Genauigkeit von Tischtennisballwürfen zu verbessern. Der Ball und der zu treffende Becher entsprechen den Dimensionen des Spiels. Die Bewegungen des Roboters im Raum und die Drehung des Bechers sollen das Treffen erschweren.

II. Vorbetrachtungen

A. Spielregeln

Um wettbewerbsfähig zu sein, wurden zunächst die Spielregeln entwickelt. Ein Farbsensor ermöglicht das Spielen in Teams. Das Spiel wird von einem blauen und einem roten Team mit jeweils farblich passenden Bällen gespielt. Die Spieler befinden sich auf gegenüberliegenden Seiten und bewegen sich jeweils einen Meter vom Roboter entfernt. Wenn der Ball den Becher trifft, erkennt der Sensor die Farbe und gibt ein Signal aus, das für jedes Team unterschiedlich ist. Danach endet das Spiel und man kann von vorne beginnen.

Der Roboter ist so programmiert, dass er sich innerhalb eines $0{,}8\,\mathrm{m}$ bis $0{,}8\,\mathrm{m}$ großen Quadrats auf einer vorgegebenen Bahn bewegt. Der Wettbewerb dauert $60\,\mathrm{s}$. Wenn in dieser Zeit niemand den Ball wirft, beginnt das Spiel von vorne.

B. Konzeptuelle Feinheiten

Das gesamte Projekt wurde durch LEGO Mindstorms ermöglicht. Obwohl es eine Chance für die Phantasie bot, hatte es auch einige Nachteile. Die Zuverlässigkeit der Kunststoffmaterialien machte mehrere Umgestaltungen erforderlich. Außerdem gab es Einschränkungen in Form von drei Motoren und mehreren Sensoren (siehe Abb. 1). Der Mangel an Teilen erforderte die Zusammenarbeit mit anderen Teams, was die Kooperation vertiefte. Obwohl der Farbsensor nicht perfekt ist und die Farbe nicht immer erkennen konnte, wurde dennoch eine Lösung gefunden. Diese soll im Folgenden bestätigt werden.

Abbildung 1. Aussehen des Spielroboters

C. Software

Der Code wurde in MATLAB geschrieben und war dank des von den Kursleitern zur Verfügung gestellten Materials verständlich. Wir konnten unsere im Informatikunterricht erlernten Programmierkenntnisse erfolgreich anwenden. Es war auch nützlich, die im Programmierunterricht erworbenen Kenntnisse anzuwenden. Der Computer wurde über das USB-B-Kabel an den Smartblock angeschlossen. Nach der Arbeit mit MATLAB stellte sich heraus, dass das Programm selbst nicht ideal war. Eine Schwierigkeit bestand darin, dass einige Elemente, insbesondere der Farbsensor, nicht konstant betrieben werden konnten. Die Lösung bestand darin, den Sensor von Bewegung zu Bewegung zu aktivieren, anstatt ihn kontinuierlich zu betreiben.

DOI: 10.24352/UB.OVGU-2024-013

Abbildung 2. Erste Ansicht des raupenbasierten Roboters

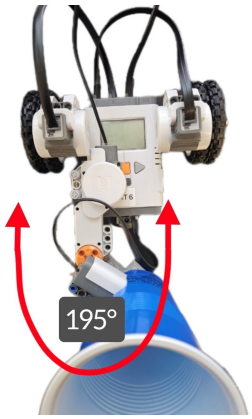


Abbildung 3. Die Mobilität des Roboters

III. Projektrealisierung

A. Anfängliches Projekt und Verbesserung

Zu Beginn wurde beschlossen, den Roboter auf einer Raupenkettenbasis aufzubauen, um ihn stabiler zu machen (siehe Abb. 2). Allerdings hatte dies den gegenteiligen Effekt: Die Raupen rollten nach innen und die Krümmung des Roboters war sogar mit bloßem Auge erkennbar. Zudem wurde ein grundlegender Fehler festgestellt. Aufgrund der Besonderheiten bei der Arbeit mit Lego ist nicht sofort klar, wie der intelligente Stein befestigt werden soll. Die Schwierigkeit besteht darin, dass er schwer ist und sich durch die beweglichen Teile hindurchdrückt.

Nach Überlegungen wurden zwei wichtige Entscheidungen getroffen: Erstens sollte alles um den intelligenten Stein herum gebaut werden, da er der monolithischste und stärkste Teil ist. Zweitens ist es notwendig, auf eine Radbasis umzusteigen. Obwohl dadurch der Kontakt mit der Oberfläche verringert wird, erhält der Roboter aufgrund der fehlenden Drehmomentübertragung Geschwindigkeit und Mobilität, was für das Projekt notwendig ist.

B. Technisches Teil

Das gesamte Projekt ist mobil dank drei Motoren, für die jeweils ein eigener Code geschrieben wurde. Die Motoren A und B wurden für die Bewegung verwendet. Jeder Motor war nur für ein Rad zuständig, so dass der Roboter sich auf der Stelle drehen und wenden konnte. Zur Stabilisierung wurde ein Element verwendet, das einem Lager ähnelte. Es war am vorderen Teil des Roboters angebracht und ermöglichte auch Drehbewegungen, da es sich in seiner Hülle in alle Richtungen drehen konnte. Der Roboter blieb auch bei maximaler Leistung und einem scharfen Start stabil und behielt seine Position bei. Um eine bessere Haftung auf der Oberfläche zu gewährleisten, wurden zwei Räder auf jeder Seite verwendet. Es wird jedoch empfohlen, den Roboter nur auf einer ebenen und harten Oberfläche zu verwenden.

Motor C ist für die Drehung des Bechers verantwortlich. Aufgrund der geringen Größe des Roboters wurde beschlossen, die Drehung bei 195° zu belassen. Die Bewegung des Bechers beginnt an der äußersten rechten Position und setzt sich fort durch eine Drehung um 195° gegen den Uhrzeigersinn, gefolgt von einer Drehung um 195° im Uhrzeigersinn und so weiter während des gesamten Spiels. (siehe Abb. 3) Durch die Bewegung des Bechers wird es viel schwieriger, das Ziel zu treffen. Tests haben gezeigt, dass die Genauigkeit des Balls um 25 % abnimmt, wenn der Becher bewegt wird. Es gibt 4 Verbindungskabel, die die Motoren und den Sensor mit dem Gerät verbinden.

C. Farbsensor-Anwendung

Ein Farbsensor ist am Becher befestigt. Wenn der Ball auf den Becher trifft, erkennt der Sensor die Farbe des Balls genau, da der Abstand zwischen Sensor und Ball nur wenige Millimeter beträgt. Ein roter Ball wird mit 800 Hz signalisiert, ein blauer mit 300 Hz. So kann man erkennen, welche Mannschaft den Ball geworfen hat und somit das Spiel gewonnen hat. Die Tondauer beträgt 760 Millisekunden, was ausreichend ist, um den Sieger deutlich zu hören, ohne dass der Ton störend wirkt.

Das Hauptproblem besteht darin, dass der vorhandene Sensor nicht in der Lage ist, den Inhalt des Bechers kontinuierlich abzutasten. Er kann nur ein- und ausgeschaltet werden, aber nicht die ganze Zeit eingeschaltet bleiben. Die Lösung des Problems war nicht der ständige Betrieb des Sensors, sondern das ständige Ein- und Ausschalten des Sensors. Der Sensor wird aktiviert, wenn der Roboter eine Bewegung ausführt, wie z.B. Drehen, Anhalten oder Rotieren.

```
OpenNXT2Color(port, 'FULL')
        color = GetNXT2Color(port)
        if strcmp(color, 'BLUE')
            NXT_PlayTone(300, 760)
            StopMotor('all','off')
            break
        elseif strcmp(color, 'RED')
             NXT_PlayTone(800, 760)
             StopMotor('all','off')
             break
        end
```

Diese Lösung hat einen Nachteil: Der Sensor funktioniert nur, wenn sich die Bewegung ändert, nicht während der Bewegung selbst. Wenn der Ball also während der Vorwärtsbewegung geworfen wird, erreicht der Roboter erst den Wendepunkt, bevor das Spiel endet. Die beste Änderung am Matlab-Programm wäre die Möglichkeit, mehrere verschiedene Codes parallel zu schreiben und auszuführen. Das bedeutet, dass ein Code für die Bewegung des Roboters und ein anderer für die Steuerung der Sensoren zuständig ist. Dies würde es ermöglichen, den bestehenden Code zu vereinfachen und unsere Idee mit dem kontinuierlichen Abtasten des Becherinhalts zu realisieren.

D. Programm

Der Gesamtcode besteht aus Zyklen. Zunächst beschreibt der Roboter ein Quadrat und führt dann innerhalb dieses Quadrats weitere Aktionen aus. Anschließend werden alle Aktionen im Hauptzyklus wiederholt.

Wie bereits erwähnt, dreht sich der Becher kontinuierlich. Dies wird durch eine von NXT gelesene Zeile ermöglicht. Sie prüft, ob der Motor C läuft und schaltet ihn gegebenenfalls wieder ein, damit die Drehung in die entgegengesetzte Richtung geht und der Becher stets in Bewegung bleibt.

```
OBJC = NXTMotor('C','Power',10)
x=0
while x < 2
    i=0
while i < 4
    if ~OBJC.ReadFromNXT().IsRunning
        OBJC.Power = -OBJC.Power;
        OBJC.TachoLimit = 180;
        OBJC.SendToNXT()
        OBJA = NXTMotor('A', 'Power', 50)
        OBJB = NXTMotor ('B', 'Power',50)
        OBJA.TachoLimit = 600;
        OBJB.TachoLimit = 600;
        OBJA.SendToNXT()
        OBJB.SendToNXT()
        OBJA.WaitFor()
        OBJB.WaitFor()
```

E. Bluetooth-Anwendungspotenzial

Viele Betriebsprobleme sind auf die Länge des Kabels zurückzuführen. Der Spielroboter ist sehr mobil und die Länge des Kabels schränkt sein Potenzial stark ein. Nachdem es möglich war, den Roboter über Bluetooth zu verbinden, zeigten Tests überraschende Ergebnisse. Das Spiel war interessant und der Roboter konnte sich frei bewegen. Jedoch hat der NXT den Nachteil, dass die Bluetooth-Verbindung nach drei Versuchen vollständig verloren geht und nur noch ein kurzes Kabel als Ausweg bleibt. Trotzdem wurde bei den kurzen 3 Versuchen über Bluetooth festgestellt, dass es eine Verzögerung zwischen dem Senden der vom Sensor gelesenen Daten an den Computer und dem Stoppen des Spiels gibt. Diese Verzögerung entspricht der Verzögerung vor dem Start des Projekts über Bluetooth. Es handelt sich um einen Fehler von MATLAB, der leider nur durch eine Änderung des Programms selbst behoben werden kann. Das Potenzial der Verwendung von Bluetooth in dem Projekt ist jedoch enorm.

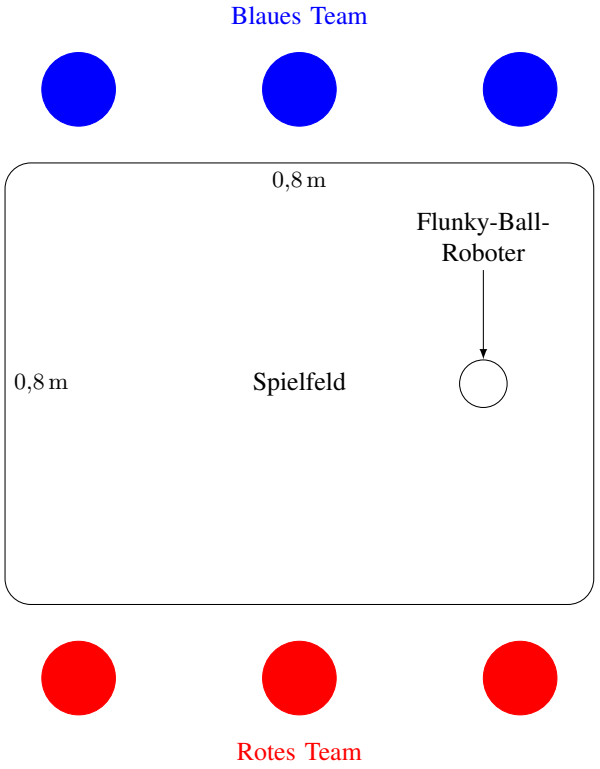


Abbildung 4. Das Spielschema "Precision Shooter".

F. Precision Shooter

Precision Shooter ist der Name des ersten Spiels, das für diesen Roboter entwickelt wurde. Es ist ein lustiges Spiel für einen angenehmen Zeitvertreib in Gesellschaft. Für das Spiel benötigt man zwei Personen und die gleiche Anzahl Bälle in zwei Farben (siehe Abb. 4). Der Roboter steht in der Mitte des Raumes und die Spieler bewegen sich 2 Meter von ihm entfernt Nachdem der Roboter gestartet wurde, haben die Spieler 1 Minute Zeit, um den Ball in den Korb zu werfen. Die Mannschaft, die den Ball zuerst in den Korb wirft, gewinnt. Schafft es keine Mannschaft, den Ball in der vorgegebenen Zeit in den Korb zu werfen, beginnt das Spiel von vorne. Es ist auch möglich, die Spielmechanik des Roboters auf verschiedene Weise zu verändern. Zum Beispiel kann man die Anzahl der Teams auf 3, 4 oder mehr erhöhen, indem man dem Code nur 2 Zeilen hinzufügt.

Natürlich ist es auch möglich, Precision Shooter alleine zu spielen. Dazu stellt der Spieler den Roboter in die Mitte des Raumes, entfernt sich 2 Meter und versucht dann, den Ball in den Becher zu werfen. Dieses Spiel ist nicht nur für Erwachsene, sondern vor allem auch für Kinder geeignet, da es die Koordination und Zielgenauigkeit schult.

G. Flunky-Ball – neue Interpretation

Der Roboter ist auch ideal, um Kindern und Erwachsenen das alte Spiel Flunky Ball beizubringen. Zwei Mannschaften

stehen sich in einem Abstand von ca. 3-4 Metern gegenüber. Die Anzahl der Spieler pro Team ist nicht standardisiert, in der Regel haben die Teams zwischen 3 und 8 Spieler. Vor jedem Spieler auf dem Boden steht eine Flasche mit Wasser, Limonade oder einer anderen Flüssigkeit. Zwischen den beiden Mannschaften steht ein Roboter in der Mitte des Spielfeldes. Beide Mannschaften haben gleich viele Bälle und werfen sie abwechselnd. Trifft eine Mannschaft, muss ein Spieler seine Flasche austrinken, dann den Ball aus dem Becher holen und das Spiel geht weiter. Nach einem Treffer trinkt der nächste Spieler. Wenn eine Mannschaft alle Getränke ausgetrunken hat, hat sie das Spiel gewonnen.

IV. Ergebnisdiskussion

Das Endergebnis des Projekts wurde erfolgreich getestet und bei der Abschlusspräsentation vorgestellt. Das vollständige Video des Spiels ist auf dem YouTube-Kanal von Kursleiter Mathias Magdowski zu finden [1]. Das Spiel dauert eine Minute und bietet die Möglichkeit, es mehrmals zu spielen, und der Roboter entwickelt Wurf-Fähigkeiten. Die Möglichkeiten, den Code und das Projekt in Zukunft zu nutzen, sind vielfältig. Leider gibt es in MATLAB eine Reihe von Problemen, die es nicht ermöglichen, das Potenzial des Roboters voll auszuschöpfen. Um viele derzeit ungenutzte Ideen zu verwirklichen, können die folgenden Änderungen in MATLAB vorgenommen werden:

1) Paralleles Schreiben von Code für mehrere Elemente gleichzeitig. Am Beispiel des Farbsensors wird die Notwendigkeit des kontinuierlichen Betriebs bestimmter Mechanismen.
2) Verbesserung der Bluetooth-Verbindung. Die aktuelle Bluetooth-Verbindung ist instabil und funktioniert nicht richtig. Eine Lösung dieses Problems kann den Anwendungsbereich der Technologie, einschließlich des entwickelten Roboters, erheblich erweitern.

V. Zusammenfassung und Fazit

Insgesamt haben die Leiter und Teilnehmer des Kurses in den zwei Wochen ein enormes Arbeitspensum bewältigt. Alle Probleme wurden gelöst und das Endergebnis ist erfreulich und inspirierend. Die Otto-von-Guericke-Universität sollte solche Kurse häufiger anbieten, da sie die Fähigkeiten der Teilnehmer tatsächlich fördern und ihnen die Möglichkeit zur Selbstverwirklichung bieten.

Literaturverzeichnis

[1] Mathias Magdowski: *YouTube.*
https://youtu.be/CqgnKNTeBPU?t=17910 Version: Februar 2024
[2] WIKIPEDIA:*Flunkyball*
https://de.wikipedia.org/wiki/Flunkyball Version: Februar 2024

Flunky-Ball-Roboter

Wird nicht in der Lage sein, Flunky-Ball zu spielen, wird aber dabei helfen, Genauigkeit zu entwickeln.

Hennadii Shypunov, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Rahmen des Projektseminares Elektrotechnik/Informationstechnik (LEGO Mindstorms) 2024 an der Otto-von-Guericke-Universität Magdeburg wurde mit Hilfe des LEGO-Baukastens ein Roboter konstruiert, der durch seine Bewegungen die Präzision menschlicher Würfe verbessern könnte.**

Eine in MATLAB entwickelte Software ermöglicht es dem Roboter, sich entlang einer vorgegebenen Route zu bewegen. Diese Bewegung unterstützt die Verbesserung der Präzision bei menschlichen Ballwürfen, indem sie ein dynamisches Trainingsszenario bietet. Zudem verfügt der Roboter über eine Selbstabschaltfunktion, um Energie zu sparen.

Zum Abschluss des Projekts wurde der entwickelte Flunky-Ball-Roboter in einer Präsentation vorgestellt und seine Funktionalität demonstriert.

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

I. Einleitung

UM einen Ball präzise zu werfen, muss eine Person mehr als 30 Muskeln einsetzen. Die Genauigkeit des Wurfs hängt von der Koordination und der Reaktion des Werfers ab. Gleichzeitig haben die Menschen zu Hause nicht immer die Möglichkeit, Bälle in voller Größe zu werfen, und der nächste geeignete Ort kann weit von ihrem Wohnort entfernt sein.

Durch das Üben mit Flunkyball-Robotern kann man Wurfgenauigkeit, Ausdauer und die Fähigkeit, geduldig zu sein, entwickeln, ohne das Haus zu verlassen. In diesem Fall ist das Risiko, Möbel zu beschädigen oder sich zu verletzen, minimal.

Der Roboter verfügt über einen stabilen Radstand, der einen Korb mit Sensor entlang einer vorgegebenen Strecke bewegt. Zusätzlich zur Bewegung der Basis selbst dreht sich auch der Sensorbehälter um 180°, was den Vorgang etwas erschwert. Ein im Korb integrierter Sensor ermöglicht es mehreren Personen, abwechselnd leichte Plastikbälle in verschiedenen Farben zu werfen, was das Spiel interessanter macht. Wenn der Ball nicht geworfen wird, bleibt der Roboter am Ende der Strecke stehen. Dadurch wird sichergestellt, dass der Roboter nicht mit Objekten außerhalb seines Bewegungsbereichs kollidiert.

II. Vorbetrachtungen

Damit ein Roboter die ihm übertragenen Aufgaben erfolgreich ausführen kann, muss man genau verstehen, welche Möglichkeiten dafür zur Verfügung stehen, und auch sicherstellen, dass die Idee als Ganzes umgesetzt werden kann.

DOI: 10.24352/UB.OVGU-2024-014

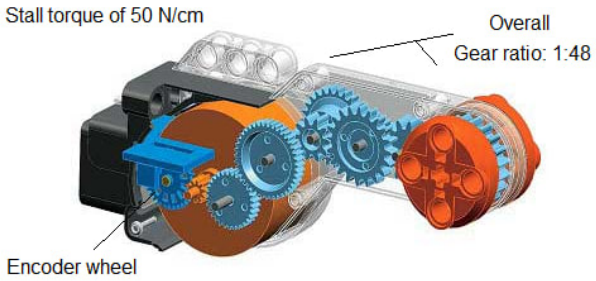


Abbildung 1. Innenansicht des Motor [1]

A. Einführung in Motoren und Sensoren

Bevor mit dem Zusammenbau des ersten Modells begonnen wurde, wurde beschlossen, die Funktionsweise der NXT-Motoren (siehe Abbildung 1) zu untersuchen und herauszufinden, welche Geschwindigkeiten sie erreichen können. Auch dem Farbsensor wurde besondere Aufmerksamkeit gewidmet, da er ein sehr wichtiges und fragiles Element des Projekts war. Dank einer so kleinen Studie war es möglich zu verstehen, welche Möglichkeiten uns der LEGO-Konstrukteur für die weitere Arbeit damit bietet. Dieses Wissen ermöglichte auch ein tieferes Verständnis der Softwarekomponente des Projekts, was es später ermöglichte, zuverlässigen und verständlichen Quelltext zu erstellen.

B. Roboteraufgaben

Es ist wichtig, die Aufgaben, die der Roboter ausführen muss, klar zu definieren, da Unklarheiten oder Unsicherheiten dazu führen können, dass eine zunächst erfolgreiche Idee in einer Reihe von Versuchen, sie zu verbessern, stecken bleibt. Daher wurde beschlossen, ein starkes konzeptionelles Grundgerüst zu erstellen und diesem dann Quelltext und LEGO-Elemente hinzuzufügen.

C. Technische Umsetzung und Softwarekomponenten

Trotz der Vielzahl an Designideen waren die Möglichkeiten des Projekts zunächst durch die Fähigkeiten von MATLAB und die spezifischen Herausforderungen beim Zusammenbau von LEGO-Strukturelementen begrenzt. Diese Faktoren schränkten auch die Fähigkeiten des Roboters ein. Außerdem ergaben sich während der Programmierung des Roboters Herausforderungen durch spezielle Funktionen in MATLAB, was das Team dazu zwang, den Quelltext mehrmals zu ändern.

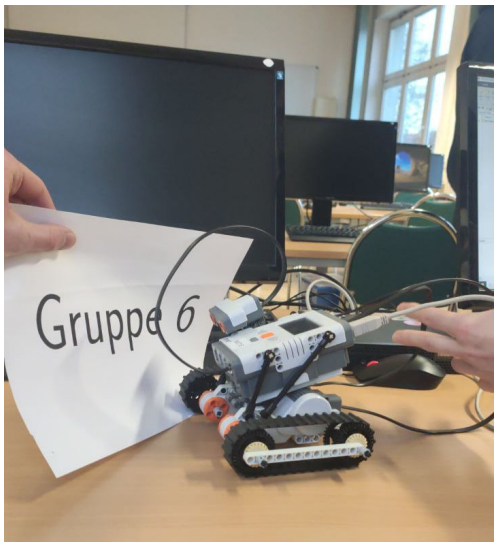


Abbildung 2. Erster Prototyp auf Raupenbasis

III. Konstruktion und Realisierung

A. Design und erster Prototyp

Ursprünglich war der Roboter auf Raupenbasis konzipiert, wobei zur Erhöhung der Stabilität ein Stein mit leichter vertikaler Abweichung befestigt wurde (siehe Abbildung 2). Anschließend wurde am Stein selbst eine zusätzliche Halterung angebracht, die den dritten Motor hielt, der für die Bewegung des Korbes mit dem Sensor zuständig ist. Unmittelbar nach dem Zusammenbau der Struktur wurden Probleme mit der Spannung der Raupen festgestellt, wodurch die Befestigungen, die die Rollen hielten, geschwächt wurden. Leider gelang es dem Roboter nach den ersten Tests nicht, ausreichend Geschwindigkeit zu entwickeln, und später stellte sich heraus, dass die gesamte Struktur nach einer kurzen Bewegung schwächer wurde (siehe Abbildung 2).

Aus diesem Grund wurde beschlossen, die Basis des Roboters zu ändern und zwei Motoren anzubringen, die dafür verantwortlich waren, den Roboter direkt zum Stein zu bewegen, sowie einen dritten, der den Korb mit dem Sensor drehte. Dadurch konnte die Anzahl der Verbindungen reduziert werden, was die Struktur stabiler machte und auch den Einfluss von Vibrationen durch Elektromotoren verringerte (siehe Abbildung 3).

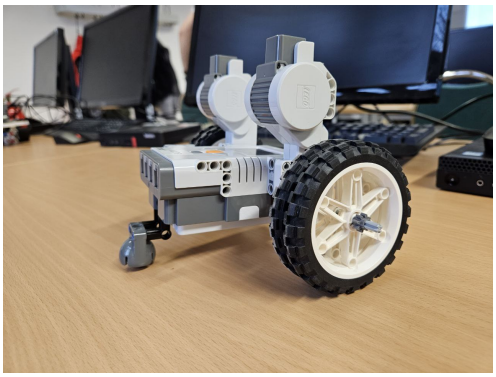


Abbildung 3. Verbesserte Version mit Radstand

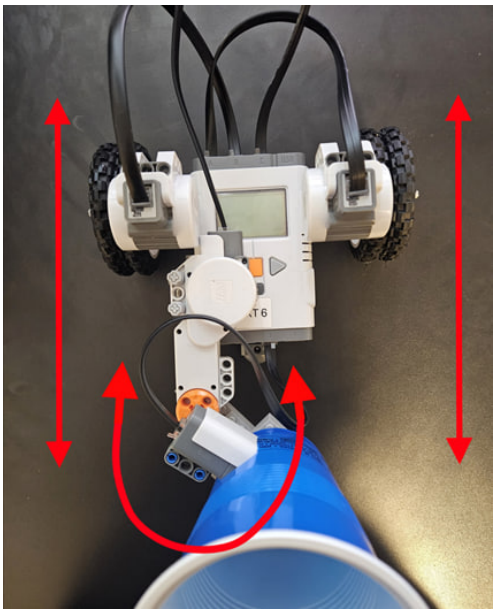


Abbildung 4. Zwei Motoren bewegen die Basis des Roboters und der dritte dient zum Bewegen des Sensorkorbs

B. Modul mit Sensor zur Farberkennung

Neben dem Radstand ist ein weiterer wichtiger Teil des Roboters der Farbsensor, dessen Aufgabe es ist, die Farbe des Balls zu lesen, der in den Korb fällt. Da alle Sensoren direkt mit dem Stein verbunden sein müssen, ist die Möglichkeit, sie um eine volle Umdrehung zu drehen, eingeschränkt. Aus diesem Grund wurde ein Drehwinkel von $180°$ gewählt (siehe Abbildung 4). Da die Teilnahme von zwei Personen oder zwei Teams geplant war, liest der Sensor beliebige Farben, aber erst wenn Blau oder Rot erkannt wird, stoppt der Roboter. Durch die sichere Montage des Korbes mit dem Sensor konnte sichergestellt werden, dass dieser auch bei einem direkten Aufprall auf den Sensor an Ort und Stelle bleibt und bereit ist, die Farben der auftreffenden Bälle abzulesen. Lediglich bei einem Element des gesamten Roboters wurde kein Material verwendet, das in einem LEGO-Set enthalten wäre. Nämlich ein Plastikbecher, der als Korb fungiert, in den der Spieler den Ball werfen muss.

C. Quelltext in MATLAB

Beim Schreiben des Quelltextes wurde beschlossen, dem Roboter eine klare Bewegungsroute zu geben. Dadurch konnte sich der Roboter in der Mitte eines $0{,}8\,\mathrm{m}$ bis $0{,}8\,\mathrm{m}$ Quadrats bewegen. Das einzige, was den Roboter und den Computer verband, war das Kabel, ohne das es unmöglich war, Befehle von MATLAB zu übertragen. Ein Versuch, Bluetooth zu verwenden, wurde ebenfalls unternommen, führte jedoch zu keinem positiven Ergebnis. Es konnte keine Verbindung zum drahtlosen Netzwerk durch den Roboter hergestellt werden, daher blieb ein Kabel die einzige Option. Der Quelltext besteht aus Elementen, die denen in der Abbildung 5 ähneln. Jedes

Element gibt die Richtung, Geschwindigkeit und Distanz an, die der Roboter zurücklegen muss.

Der gesamte Quelltext besteht aus kleinen Abschnitten, die angeben, in welche Richtung, mit welcher Geschwindigkeit sich die Räder drehen und wie sich der Sensorkorb dreht. Es wurde eine while-Schleife verwendet. Es war notwendig, eine Route in Form eines Quadrats anzulegen. Zusätzlich wurde `"if ~OBJC.ReadFromNXT().IsRunning"` verwendet, um den Sensorkorb zu bewegen, bei Stopps und Wendemanövern. Darüber hinaus wurde `"if strcmp(color, 'BLUE')"` und `"elseif strcmp(color, 'RED')"` verwendet, um Bedingungen zu schaffen, unter denen der Roboter anhält, wenn der Ball den Korb trifft, und ihn scannt. Außerdem wurden dem Quelltext Zeilen hinzugefügt, die einen Piepton mit verschiedenen Tönen abspielten, um zwischen blauen und roten Balltreffern zu unterscheiden (siehe Abbildung 5).

Es ist zu beachten, dass die Bewegung des Roboters nicht mit dem endet, was im folgenden Quelltextelement angegeben ist. Der Roboter dreht sich auch auf der Stelle, bewegt sich geradlinig in verschiedene Richtungen und mit unterschiedlicher Geschwindigkeit und führt auch andere Manöver aus. Aufgrund seiner Größe ist es nicht möglich, den gesamten Quelltext einzufügen.

```
while i < 4
if ~OBJC.ReadFromNXT().IsRunning
        OBJC.Power = -OBJC.Power;
        OBJC.TachoLimit = 180;
        OBJC.SendToNXT()

        OBJA = NXTMotor('A', 'Power', 50)
        OBJB = NXTMotor ('B', 'Power',50)
        OBJA.TachoLimit = 600;
        OBJB.TachoLimit = 600;
        OBJA.SendToNXT()
        OBJB.SendToNXT()
        OBJA.WaitFor()
        OBJB.WaitFor()

        OpenNXT2Color(port, 'FULL')
        color = GetNXT2Color(port)
        if strcmp(color, 'BLUE')
           NXT_PlayTone(300, 760)
           StopMotor('all','off')
           break
        elseif strcmp(color, 'RED')
            NXT_PlayTone(800, 760)
            StopMotor('all','off')
            break
        end
```

Abbildung 5. Flunkyball-Roboter

IV. Ergebnisdiskussion

Nachdem die Programmierung und Montage abgeschlossen waren, wurde der Roboter getestet. Im Korbbefestigungselement wurde eine strukturelle Schwachstelle identifiziert und anschließend behoben.

Nach mehreren Tests mit Leuten aus anderen Teams wurde entschieden, dass der Roboter alle Anforderungen erfüllte. Seine Bewegung wurde nur durch das mit dem Laptop verbundene Kabel begrenzt, und der Farbsensor bestimmte zuverlässig die Farbe des Balls, woraufhin sich der Roboter abschaltete.

V. Zusammenfassung und Fazit

Zusammenfassend lässt sich sagen, dass das endgültige Design des Geräts alle seine Ziele erreicht hat. Durch das Studium von MATLAB konnten wichtige Kenntnisse erworben werden, die in Folgeprojekten eine wichtige Rolle spielen werden. Der Roboter hat Entwicklungspotenzial. Zukünftig besteht Möglichkeit, den Bewegungsweg des Roboters im Code nicht mehr strikt vorzuschreiben, sondern Abstandssensoren hinzuzufügen, entlang derer sich der Roboter im Raum bewegt und bei Bedarf selbstständig Hindernisse ausweicht.

Literaturverzeichnis

[1] Philippe Hurbain: *NXT® motor internals.*
https://www.philohome.com/nxtmotor/motor1-3.jpg Version: Januar 2023
[2] WIKIPEDIA:*Flunkyball*
https://de.wikipedia.org/wiki/Flunkyball Version: Februar 2024

Tastaturroboter

Vladyslav Shkliarslyi, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Rahmen des Projektseminars Elektrotechnik/Informationstechnik (LEGO-Mindstorms) 2024 an der Otto-von-Guericke-Universität Magdeburg wurde den Studierenden zunächst das Ziel, einen Roboter zu erstellen, erläutert, dann die Grundlagen der Programmierung mit MATLAB vermittelt und schließlich die LEGO-NXT-Sets zur Verfügung gestellt.**

***Schlagwörter*—LEGO Mindsorm, MATLAB, NXT-Farbsensor, Tastatur-Roboter, Textverarbeitung**

I. Einleitung

Gleichzeitig mit dem Fortschritt der Technologie wächst auch die menschliche Faulheit, und zwar schneller als je zuvor. Um das Wachstum der Faulheit zu unterstützen, wurde ein Tastaturroboter entwickelt, der in der Lage ist, eine vorgegebene Tastenkombination auf der Tastatur einzugeben. Es kann oft vorkommen, dass man nach einem anstrengenden Tag überhaupt nicht in der Stimmung ist, auf eine Nachricht zu antworten, die aber trotz der Müdigkeit beantwortet werden muss. In diesem Moment der extremen Müdigkeit ist man immer so verzweifelt, dass der Rest des Tages den Bach runtergehen kann. Genau hier kommt der Tastaturroboter ins Spiel, der eine vorgegebene Tastenkombination eingeben würde. Für die Entwicklung des Roboters wurden drei NXT-Motoren und ein NXT-Farbsensor verwendet. Die Motoren ermöglichen die Bewegung entlang der drei Achsen. Damit der Roboter während der Fahrt die Tasten in der richtigen Reihenfolge drücken kann, gibt es verschiedene Farben auf der Tastatur, Jedem Buchstaben ist eine Farbe zugeordnet, die diesen Buchstaben auf der Tastatur repräsentiert.

Abbildung 1. NXT-Farbsensor

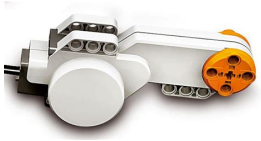


Abbildung 2. NXT-Motor

DOI: 10.24352/UB.OVGU-2024-015

II. Vorbetrachtungen

In diesem Abschnitt werden die Funktionen und Eigenschaften der für die Roboterentwicklung notwendigen Komponenten aus Sicht der Programmierung vorgestellt.

A. NXT-Motor

In der von MATLAB bereitgestellten Bibliothek, die zur Programmierung des Roboters verwendet wurde, stellt der NXT-Motor eine Struktur dar, deren Variablen die Eigenschaften des Notors repräsentieren. Bei der Programmierung des Tastaturroboters waren Eigenschaften wie `Tacholimit` und `Power` am relevantesten. Die Variable `Tacholimit` ist dafür verantwortlich, um wie viele Grad sich der Motor drehen kann. Der Wert dieser Variablen wird in Bogenmaß dargestellt, z. B. entspricht der Wert 360 einer vollen Umdrehung des Motors. Die Variable `Power` ist nicht nur für die Leistung verantwortlich, mit der der Motor arbeitet, sondern auch für die Richtung, in die sich der Motor dreht. Wenn Power ein negatives Vorzeichen hat, dann ist das die gleiche Leistung wie wenn Power ein positives Vorzeichen hat, aber sie dreht sich in die entgegengesetzte Richtung.

B. NXT-Farbsensor

Der NXT-Farbsensor kann insgesamt 13 Farben erkennen. Nachdem der Sensor eine Farbe erkannt hat, gibt er einen Rückgabewert in Form eines Arrays zurück. Dieses Array kann wiederum mit der von MATLAB bereitgestellten Funktion `strcmp()` mit einem String verglichen werden. Die Funktion gibt als Ergebnis des Vergleichs einen booleschen Wert zurück, wodurch sie für unendliche Schleifen geeignet ist, da man den Rückgabewert als Operation für den Ausgang der Schleife verwenden kann.

III. Hauptteil

A. Aufbau

Der Roboter bewegt sich auf vier Rädern, die von einem Motor angetrieben werden, der sich unter der Haube in Abbildung 3 befindet. Der Roboter kann nur in zwei Richtungen fahren, vorwärts und rückwärts, aber das ist auch genügend. Auf den Rädern befindet sich ein Gestell, auf dem eine NXT-Steuerung und eine weitere Konstruktion angebracht sind. Diese weitere Konstruktion enthält zwei Motoren, von denen einer dazu dient, den Druckteil, der auf der Abbildung 5 zu sehen ist, über die Tastatur zu schieben, und der zweite dazu dient, den Druckteil in die Tastatur hinein zu bewegen.

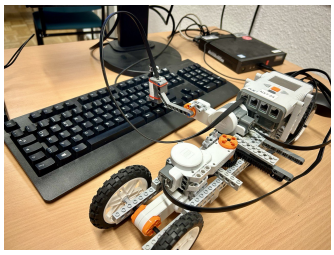


Abbildung 3. Tastaturroboter

Abbildung 4. Sicht von oben

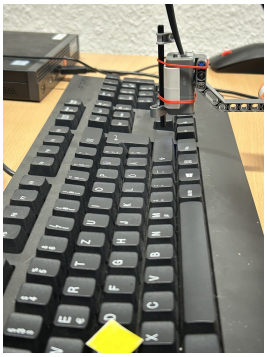


Abbildung 5. der drückende Teil

IV. Funktionsweise

In diesem Teil werden die drei genutzten Motoren des Verständnis halber solcherweise unterteilt:

Motor 1 - bewegt den Roboter entlang der Tastatur
Motor 2 - schiebt den drückenden Teil über die Tastatur
Motor 3 - drückt den drückenden Teil in die Tastatur hinein

A. Fahrt

Vor dem Start schiebt der Roboter den vorderen Teil, an dem der Farbsensor befestigt ist, durch den Antrieb von Motor 2 über die Tastatur um eine vordefinierte Länge aus. Mit dem ausgefahrenen Farbsensor fährt der Roboter entlang der Tastatur. Der Farbsensor analysiert in jedem Moment die Farbe, die sich vor ihm befindet. Wenn die Farbe erkannt wird, hält der Roboter an. Wenn die richtige Farbe gedrückt wurde, fährt der Roboter zur nächsten Farbe. Wenn nicht, wird die ursprüngliche Länge, um die der Farbsensor herausgezogen wird, etwas verkürzt und beim dritten Versuch vergrößert. Es ist zu beachten, dass die Tastenkombination vor der Fahrt festgelegt werden muss.

Listing 1. Definition der Struktur

```
typedef struct {
    char y;// Abstand zur Y-Achse
    int nummer;// Reihenfolgenummer
}Buchstabe;
typedef struct {
    Buchstabe p;
    Buchstabe o;
    Buchstabe w;
    Buchstabe e;
    Buchstabe r;
    Buchstabe o;
    Buchstabe f;
    Buchstabe f;
}Word;
```

V. Software

Wie bereits erwähnt, muss die Tastenkombination im Voraus definiert werden. Dazu wird eine Struktur verwendet, die mehrere Strukturen enthält. Da MATLAB eine seltsame Art hat, Strukturen zu definieren, wird in Listing 1 ein Beispiel mit der Definition derselben Struktur, jedoch in der Sprache C, gezeigt. Für den Code wurde ein rekursiver Ansatz gewählt. In der Main-Funktion werden zunächst drei Motoren und ein Farbsensor sowie die Struktur, die das Wort repräsentiert, initialisiert. Danach wird die rekursive Funktion `typing` aufgerufen, die in Listing 2 zu sehen ist. Die Funktion nimmt 6 Werte als Parameter an, nämlich 3 Motoren, eine Struktur, die aktuelle Farbe und den Rückgabewert, den eine andere Funktion zurückgibt. Die aktuelle Farbe ist eine Zahl. Durch diese Zahl versteht der Roboter, welche Farbe er suchen soll. Die Anzahl der Farben ist vorher festgelegt, so dass sie als Steuerung für den Ausgang der Rekursion dient. Der letzte Parameter `Z` ist der Rückgabewert, der angibt, ob die gewünschte Taste gedrückt wurde. Wenn dieser Wert nicht gleich 1 ist, bedeutet dies, dass die gewünschte Taste nicht gedrückt wurde, und der Roboter muss das Tacholimit des zweiten Motors entweder vergrößern oder verkleinern. Die Funktion `some`, die auch in Abbildung 8 zu sehen ist, übernimmt die Vorbereitung der Motoren und die Bestimmung der Farbe, die der Roboter suchen soll. Am Ende der Vorbereitung übergibt die Funktion `some` die Kontrolle an eine andere Funktion, in der eine Endlosschleife abläuft, während der der Farbsensor arbeitet. Wenn der Farbsensor erfolgreich die gewünschte Farbe findet, wird die Schleife beendet, der dritte Motor angetrieben und die Taste gedrückt. Nachdem die Taste gedrückt wurde, dreht sich der dritte Motor bis zum Ausgangszustand. Danach kehrt die Kontrolle an die Stelle zurück, an der die letzte Funktion in der Funktion

Listing 2. Funktion typing

```
function [] = typing(Letters,
             current_color,
             motor1,
             motor2,
             motor3,z)
    if current_color > x
        return
    end
    retval = some(Letters,
    current_color,
    motor1,
    motor2,
    motor3,z);
    if retval ~= 1
        typing(Letters,
        current_color,
        motor1,
        motor2,
        motor3,0);
    else
        typing(Letters,
        current_color + 1,
        motor1,
        motor2,
        motor3,1)
    end
end
```

`some` aufgerufen wurde. Dort kehrt alles in den ursprünglichen Zustand zurück und die Funktion `some` übergibt bei Erfolg 1 an die Funktion `typing`. Dann wird in der gleichen Funktion `typing` die gleiche `typing` aufgerufen, die jetzt aber eine andere Farbe als die gesuchte Farbe hat. So wird die Funktion `typing` rekursiv aufgerufen, bis die aktuelle Farbe größer wird als die Steuerung für den Ausgang der Funktion.

VI. Ergebnisdiskussion

Das Ergebnis ist ein funktionsfähiger Tastaturroboter, der in der Lage ist, eine vorgegebene Kombination von Buchstaben mit hoher Genauigkeit einzugeben. Es gibt natürlich noch viele weitere Möglichkeiten, die Konstruktion und Funktionalität des Roboters zu verbessern, aber diese wurden aufgrund der begrenzten Zeit nicht berücksichtigt.

VII. Zusammenfassung und Fazit

Das Ziel des Projekts war nicht nur, einen Roboter mit einer bestimmten Funktionalität zu entwickeln, sondern auch praktische Erfahrungen zu sammeln, die Arbeitsumgebung auszuprobieren, die Herangehensweise an nicht triviale Probleme zu entwickeln, in einem Team zu arbeiten, zu lernen, wichtige Entscheidungen selbst zu treffen und viele andere praktische Kenntnisse zu erwerben.

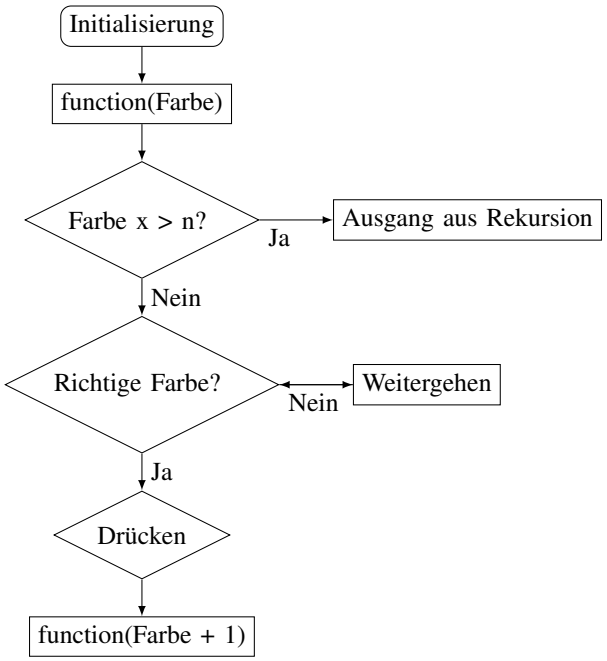


Abbildung 6. Schema des Code-Algorithmus

Literaturverzeichnis

[1]*MATLAB: Dokumentation*.
https://de.mathworks.com/help/matlab/ref/function.html
Version:2023

[2]*Brickwiki: NXT Components*.
https://brickwiki.org/wiki/NXT
Version:2013

[3] *MINDSTORMSNXT: User manual Lego*.
https://www.manualslib.com/manual/726722/Lego-Mindstorms-Nxt.html#manual

Tastatur-Roboter

Mykhailo Zahorodniuk, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Rahmen des faszinierenden Projektseminars LEGO Mindstorms im Bereich Elektrotechnik-Informationstechnik im Jahr 2024 an der renommierten Otto-von-Guericke-Universität Magdeburg wurde ein Roboter entwickelt und umgesetzt, der in der Lage ist, Texte über eine Tastatur einzugeben. Erfunden wurde nicht nur das richtige Konzept, sondern auch der Programmcode, der für die Effizienz sorgen soll.**

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 Lernprozess erstreckte sich über verschiedene Phasen, die von der Theorie bis zur praktischen Umsetzung reichten.

***Schlagwörter*—LEGO Mindsorm, MATLAB, NXT-Farbsensor, Tastatur-Roboter, Textverarbeitung**

I. Einleitung

IM heutigen Streben nach ständiger Automatisierung und Optimierung des täglichen Lebens ist das Projectseminar ein Vorreiter einer innovativen Lösung zur Verbesserung der Tastatureingabe. Mit dem unaufhaltsamen Fortschritt der Technologie manifestiert sich parallel dazu auch die rasant zunehmende Trägheit der Menschheit. Diese Entwicklung vollzieht sich in einem noch nie dagewesenen Tempo. Als Lösung zur Erleichterung der täglichen Routine wurde ein innovativer Tastaturroboter entwickelt, der in der Lage ist, eine vorgegebene Tastenkombination auf der Tastatur zu tippen. Nachdem er die exakte Tastenkombination programmiert hat, ist der Roboter in der Lage, in jedem Moment der Faulheit oder in Momenten der Ermüdung die erforderlichen Textzeilen mit Präzision zu reproduzieren.

Drei NXT-Motoren [1] und ein NXT-Farbsensor werden verwendet, um diesen Roboter zu bauen. Die Motoren ermöglichen die Bewegung in drei Achsen. Erster Motor wurde auf ein Fahrgestell mit vier Rädern gesetzt. Der zweite Motor bewegt den Push-Pull-Ausleger (siehe Abbildung 1) auf die Höhe der Tastatur. Und der dritte Motor ist für den Tastendruckmechanismus zuständig. Damit der Roboter in der Lage ist, die Tasten in der richtigen Reihenfolge zu drücken, während er sich bewegt, werden den richtigen Tasten auf der Tastatur verschiedene Farben zugeordnet, die vom Farbsensor des NXT auf der Tastatur erkannt werden.

Nach der Zuordnung der Farben zu den richtigen Tasten ist der Roboter in der Lage, die richtige Taste zu finden. Diese Kombination aus Technologie und Farbcodierung ermöglicht es dem Tastaturroboter, dem Benutzer zu helfen und in Momenten extremer Ermüdung immer zur Stelle zu sein. Dies steigert nicht nur die Effizienz, sondern löst auch die Probleme des modernen Lebens, indem die Technologie in den Alltag integriert wird.

DOI: 10.24352/UB.OVGU-2024-016

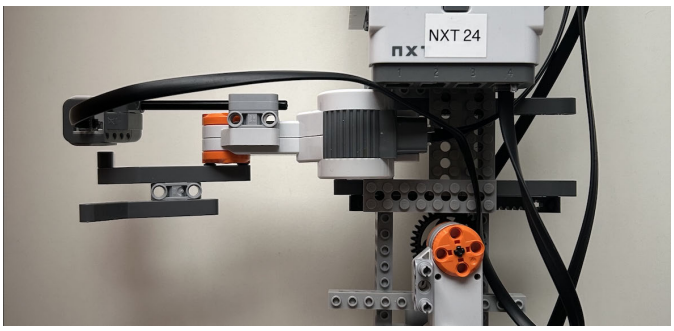


Abbildung 1: Push-Pull-Ausleger

II. Vorbetrachtungen

A. NXT-Motor

Im Rahmen der von MATLAB [2] vorgeschlagenen Bibliothek, die für die Programmierung des Roboters verwendet wurde, stellt der NXT-Motor [3] (siehe Abbildung 2) eine umfassende Struktur dar, deren Variablen die essenziellen Eigenschaften des Motors repräsentieren. Die präzise Steuerung des Motors erfolgt durch die gezielte Veränderung dieser Variablen. Bei der Entwicklung des Tastaturroboters waren insbesondere Eigenschaften wie das Tacholimit und die Power von entscheidender Bedeutung.

Die Tacholimit-Variable spielt eine zentrale Rolle, indem sie festlegt, um wie viele Grad sich der Motor drehen kann. Hierbei wird der Wert dieser Variable im Bogenmaß ausgedrückt, was beispielsweise die erfolgreiche Navigation der Tasten in der richtigen Reihenfolge ermöglicht. Dabei entspricht 360° Grad einer vollständigen Drehung des Motors.

Die Leistungsgröße ist nicht nur für die Leistung verantwortlich, mit der der Motor arbeitet, sondern beeinflusst auch die Drehrichtung des Motors. Wenn der obigen Variablen ein negativer Wert zugewiesen wird, kehrt sie die Bewegungsrichtung um. Im Gegenteil, ein positiver Wert ermöglicht eine Bewegung in die andere Richtung. Diese Logik bietet die Möglichkeit, die Bewegungsrichtung des Roboters präzise und gezielt anzupassen.

B. NXT-Farbsensor

Der NXT-Farbsensor [4] (siehe Abbildung 3) demonstriert seine Fähigkeit, insgesamt 13 verschiedene Farben zu erkennen (siehe Abbildung 1 im Anhang). Sobald der Sensor eine Farbe identifiziert hat, gibt er den Rückgabewert als Array zurück, das wiederum bequem mit einer Zeichenkette verglichen werden

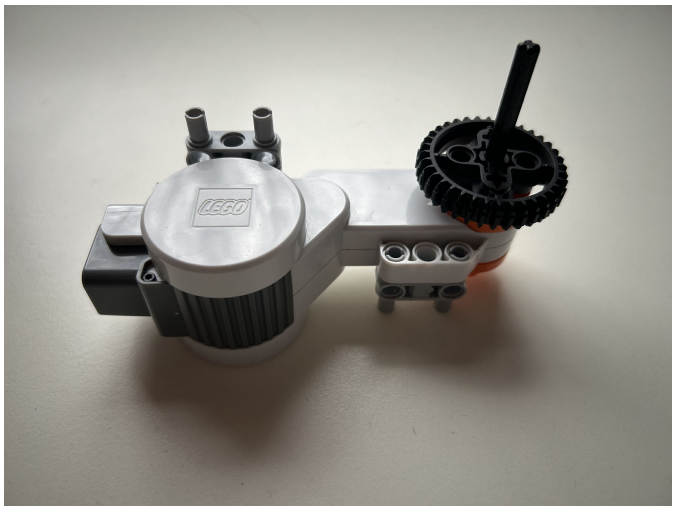


Abbildung 2: NXT-Motor

Abbildung 3: NXT-Farbsensor

kann. Der Vergleich des Werts ergibt eine logische Antwort von 1 oder 0. Diese Werte können sowohl für den Vergleich als auch für das Verlassen des Zyklus verwendet werden. Dies stellt eine elegante Möglichkeit dar, den Sensor im Dauerbetrieb für eine effektive Überwachung der Farberkennung einzusetzen.

III. Entwicklungsprozess

A. Konstruktion

Um die Konstruktion zu realisieren, war es erforderlich, alle drei Motoren effizient zu nutzen und somit eine Bewegung in drei Achsen zu ermöglichen. Das Design umfasste die Entwicklung eines Teils auf dem Chassis, um entlang der Tastatur zu manövrieren. Dieser Aspekt gewährleistet horizontale Mobilität und ermöglicht es dem System, alle Bereiche der Tastatur zu erfassen und zu navigieren.

Für die vertikale Bewegung auf der Tastatur wurde ein Teil unter Verwendung des zweiten Motors entwickelt. Dieser Motor setzt einen Zahnradmechanismus in Gang, der seinerseits entlang der Führungsschienen bewegt wird und so eine präzise und

Abbildung 4: Drei-Achsen-Konstruktion

reibungslose Bewegung auf verschiedenen Ebenen der Tastatur ermöglicht. Dieser Ansatz zur vertikalen Bewegung ergänzt die horizontalen Fähigkeiten der Konstruktion und schafft so eine vollständige dreidimensionale Bewegungsfreiheit.

Der dritte Motor im System erfüllt eine wichtige Funktion – er agiert als Druckmechanismus. Dieser Motor ist für die Umsetzung des Texteingabeprozesses verantwortlich und bietet präzisen und steuerbaren Druck beim Betätigen der Taste. Dies gewährleistet eine hohe Eingabetreue und zusätzliche Flexibilität bei der Anpassung der Systemparameter an die spezifischen Anforderungen des Benutzers.

Somit gewährleistet die Integration der drei Motoren in die Konstruktion umfassende Mobilität in drei Dimensionen (siehe Abbildung 4). Dieses sorgfältig durchdachte Design ermöglicht eine effiziente Interaktion der Motoren, um maximale Funktionalität und Optimierung der Texteingabe über die Tastatur zu erreichen.

B. Programmierung

Aufgrund der besonderen Art der Strukturdefinition in MATLAB ist es wichtig, die später zu setzende Tastenkombination im Voraus zu definieren (siehe Abbildung 2 im Anhang). Für die Implementierung des Codes wurde ein rekursiver Ansatz gewählt. Die Hauptfunktion beginnt mit der Initialisierung der drei Motoren, des Farbsensors und einer Struktur, die die Art des zu schreibenden Wortes widerspiegelt. Dann wird eine rekursive Funktion [5] aufgerufen (siehe Abbildung 3 im Anhang). Diese Funktion benötigt sechs Parameter, darunter

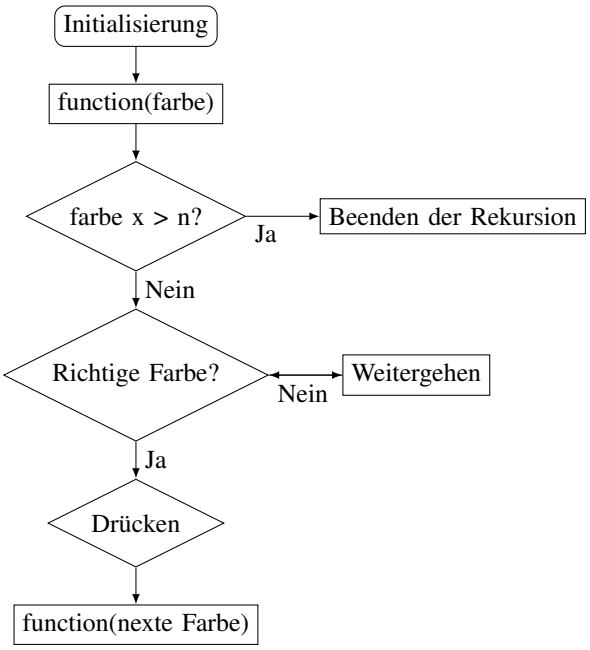


Abbildung 5: Schema des Code-Algorithmus

die drei Mechanismen, die Struktur, die aktuelle Farbe und einen Rückgabewert, der von einer nachfolgenden Funktion geliefert wird.

Die aktuelle Farbe wird durch eine Zahl dargestellt, sodass der Roboter feststellen kann, nach welcher Farbe er suchen muss. Die Anzahl der Farben ist vordefiniert und dient als Voraussetzung für das Verlassen der Rekursion. Der letzte Parameter, der Rückgabewert, gibt Auskunft darüber, ob die Zieltaste gedrückt wurde. Ist dieser Wert ungleich 1, bedeutet dies, dass die Zieltaste nicht gedrückt wurde. In einem solchen Fall muss der Roboter die Geschwindigkeit des zweiten Motors entweder erhöhen oder verringern.

Die Funktion bereitet die Motoren vor und bestimmt die Farbe, die der Roboter für die Suche verwenden soll. Sobald die Vorbereitung abgeschlossen ist, übergibt die Funktion die Kontrolle an eine andere Funktion, die eine Endlosschleife startet (siehe Abbildung 5), in der der Farbsensor aktiv ist. Wenn der Farbsensor erfolgreich die gewünschte Farbe erkennt, wird die Schleife unterbrochen und der dritte Motor tritt in Aktion und drückt die Taste. Wenn die Taste gedrückt wird, kehrt der dritte Motor in seine Ausgangsposition zurück. Die Steuerung kehrt dann zu dem Punkt zurück, an dem diese letzte Aktion in der Funktion eingeleitet wurde. Alle Elemente kehren in ihren ursprünglichen Zustand zurück, und wenn die Funktion erfolgreich abgeschlossen wurde, übergibt sie den Befehl zum Weitergehen an die Funktion, die den Text eingibt. Innerhalb derselben Funktion wird die Funktion erneut aufgerufen, allerdings mit einer anderen Farbe als der zuvor gefundenen. Dieser rekursive Aufruf wird so lange wiederholt, bis die aktuelle Farbe mit der zuvor angegebenen Bedingung für das Beenden der Funktion übereinstimmt.

Abbildung 6: Tastatur-Roboter

IV. Ergebnisdiskussion

Das Ergebnis ist ein Roboter, der Tasten mit ausreichender Geschwindigkeit und Präzision drücken kann. Während des Betriebs verfügt der Roboter über eine bemerkenswerte Fähigkeit zur Selbstkorrektur, was bedeutet, dass die Wahrscheinlichkeit, dass er die richtige Taste nicht erreicht, minimal ist.

Die Leistung des Roboters zeigt sich besonders in seiner Fähigkeit, Text effizient einzugeben. Die präzise Erkennung der Tastenpositionen durch spezielle Sensoren und die ständige Überprüfung der Position des Roboters durch einen Farbsensor gewährleisten eine außergewöhnliche Genauigkeit und Zuverlässigkeit während des gesamten Prozesses.

Der Roboter analysiert nicht nur die Oberfläche der Tastatur mithilfe spezieller Sensoren, sondern prüft auch während der Aufgabe kontinuierlich seine Position mit dem Farbsensor. Dieser Mechanismus ermöglicht es dem Roboter, seine Po-

sition automatisch anzupassen, was Genauigkeit und relative Zuverlässigkeit bei der Texteingabe gewährleistet.

Die beiden Hauptfunktionen – die genaue Erkennung der Tastenpositionen und die automatische Überprüfung der Roboterposition – arbeiten zusammen, um eine automatische Texteingabe zu ermöglichen.

V. Fazit

Das Projekt erreichte nicht nur das Ziel, einen funktionsfähigen Roboter zu schaffen, sondern sammelte auch wertvolle praktische Erfahrungen. Zusammenarbeit, kreatives Denken und die Umsetzung von Ideen waren ebenfalls wesentliche Bestandteile dieses Projekts. Das Ergebnis ist ein ausgezeichnet realisierter Roboter, der seine gesteckten Ziele erfolgreich erfüllen kann. Dieses Projekt eröffnet nicht nur vielversprechende Perspektiven für die Weiterentwicklung der Idee, sondern verdeutlicht auch das erhebliche Potenzial für künftige Verbesserungen.

Anhang

```
%   The color index values roughly
   correspond to the following
%   table (when using modes 0 and 1):
%            0 = black
%            1 = violet
%            2 = purple
%            3 = blue
%            4 = green
%            5 = lime
%            6 = yellow
%            7 = orange
%            8 = red
%            9 = crimson
%            10 = magenta
%            11 to 16 = pastels
%            17 = white
```

Abbildung 1: Farben, die der Sensor erkennt

```
Enter.x = 1;
Enter.y = 1;
WW.x = 315;
WW.y = 2;
NN.x = 175;
NN.y = 0;
global Letters;
Letters.enter = Enter;
Letters.W = WW;
Letters.N = NN;
```

Abbildung 2: Vereinfachtes Beispiel für eine Struktur

```
if current_color > 9
    return
end
    retval = some(Letters, current_color,
        motor1, motor2, motor3, z);
if retval ~= 1
    typing(Letters, current_color, motor1
       , motor2, motor3, 0)
else
    typing(Letters, current_color + 1,
       motor1, motor2, motor3, 1)
end
```

Abbildung 3: Rekursive Funktion

Literatur

[1] BRICKWIKI: *NXT Components*. https://brickwiki.org/wiki/NXT. Version: 2013

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

[3] MINDSTORMSNXT: *NXT Motor*. https://mindstormsnxt.blogspot.com/2006/08/closer-look-at-nxt-motors.html. Version: 2006

[4] LEGOPEDIA: *NXT Farbsensor 9694*. https://lego.fandom.com/de/wiki/NXT_Farbsensor_9694. Version: 2012

[5] MATHWORKS: *MATLAB Syntax*. https://de.mathworks.com/help/matlab/ref/function.html. Version: 2023

Tic-Tac-Toe-Spieler

Stanislav Panzhar, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

Zusammenfassung **- Im Rahmen eines jährlich stattfindenden Technikseminars wurde ein lang ersehntes Projekt realisiert, das zuvor aufgrund begrenzter Ressourcen nicht realisierbar schien: der Bau eines Tic-Tac-Toe Roboters. Mit Hilfe von LEGO-Mindstorms-Bauteilen, einem Controller und einer Kamera wurde ein funktionsfähiger Prototyp entwickelt und getestet. Der Roboter wurde entwickelt, um als interaktiver Spieler in einem Tic-Tac-Toe Spiel zu agieren. Im Laufe des Projekts wurde klar, dass der Roboter mit den Verbesserungen und Anpassungen ein großes Potenzial hat. In den folgenden Abschnitten werden die technischen Spezifikationen des Projekts detailliert beschrieben und Vorschläge für mögliche Anwendungen und Weiterentwicklungen gemacht.**

I. Einleitung

In der Welt, in der Technologie und Kreativität Hand in Hand gehen, entstehen oft faszinierende Innovationen oft aus den einfachsten Materialien. Ein solches Beispiel liegt in der Verbindung von LEGO und Robotik, die nicht nur Kinder begeistert, sondern auch die Köpfe von Studenten inspiriert. Die Idee, aus Legosteinen einen Roboter zu erschaffen, mag zunächst wie ein Spaß erscheinen, doch hinter dieser scheinbaren Simplizität verbirgt sich oft eine erstaunliche Komplexität.

II. Vorbetrachtungen

Die Konzeption beinhaltet, dass der Roboter das Spielfeld für Tic-Tac-Toe erstellt, den ersten Zug ausführt und dann zurückfährt, um dem menschlichen Spieler die Möglichkeit zu geben, seinen Zug zu machen. Nachdem der Spieler seinen Zug gemacht hat, bewegt sich der Roboter zurück und analysiert sowohl seine eigenen Züge als auch die des Spielers. Auf der Grundlage dieser Analyse platziert der Roboter dann seine eigenen Markierungen in dem vom Algorithmus ausgewählten Bereich des Spielfelds.

Um dieses Konzept umsetzen zu können, benötigt der Roboter einen Mechanismus zum Erstellen des Spielfeldes, eine Kamera zur Erkennung der Position der Spielsteine des Spielers und einen Algorithmus zur Analyse der Spielzüge. Der Roboter muss also in der Lage sein, seine eigenen Spielzüge zu planen und auszuführen.

III. Grundlegende Strukturelemente

Der Roboter wurde von drei Motoren angetrieben. Ein Motor befand sich im hinteren Teil des Roboters und steuerte die Vorwärts- und Rückwärtsbewegung des Roboters. Der zweite Motor befand sich vorne im unteren Bereich und war für die Bewegung der oberen Konstruktion des Roboters nach links und rechts verantwortlich. Der dritte Motor befand sich ganz oben auf dem Roboter. Dieser Motor bewegte den Stift nach oben und unten. Mit Hilfe dieser drei Motoren konnte der Roboter das richtige Spielfeld zeichnen.

Abbildung 1. Konstruktion des Roboters

Für die Steuerung der Motoren wurde ein NXT-Controller mit dem Akku eingesetzt, der sämtliche Komponenten des Roboters einschließlich der Motoren und des Hauptsteuerblocks mit Strom versorgte. Anschließend wurden alle Komponenten so programmiert, dass sie bestimmte Funktionen ausführten. Dabei wurden die Motoren entsprechend den Spielzügen des Roboters und des menschlichen Spielers programmiert. Die Kamera hatte die Aufgabe, die Positionen der Spielsteine zu erfassen und

DOI: 10.24352/UB.OVGU-2024-017

relevante Informationen zu sammeln. Der Hauptsteuerblock (Controller) koordinierte dann alle Aktionen des Roboters nach einem vordefinierten Algorithmus. Die Programmierung des NXT-Controllers ermöglichte eine präzise Steuerung der Motoren und der Kamera entsprechend den Anforderungen des Tic-Tac-Toe-Spiels. Es wurden Algorithmen entwickelt, um die Bewegungen des Roboters zu koordinieren, die Positionen der Spielsteine zu erkennen, Spielzüge zu analysieren und entsprechend zu reagieren. Durch diese Programmierung konnte der Roboter effektiv als autonomer Spieler agieren und eine anspruchsvolle Spielumgebung bieten.

Abbildung 2. NXT-Hauptteil

Ursprünglich wurden zwei Sensoren verwendet - Positions- und Farbsensor. Der Positionssensor hatte die Aufgabe, die Bewegungen des Roboters zu koordinieren, indem er den Raum vor ihm scannte. Der Farbsensor, der über dem Spielfeld positioniert war, sollte die Züge des Roboters und des Spielers erkennen und die Informationen an den Hauptblock weiterleiten. Es gab jedoch ein Problem damit, da der Lichtsensor das Spielfeld unzuverlässig und langsam erkannte. Aus diesem Grund wurde beschlossen, beide Sensoren durch eine Kamera zu ersetzen und diese mit MATLAB zu programmieren. Diese Änderung ermöglichte eine effizientere Erkennung und Verarbeitung der Spielfeldinformationen.

Für die Kamera wurde entschieden, dass der Roboter das Spielfeld und seine eigenen Züge mit einem schwarzen Stift zeichnet, während der Spieler einen roten Stift verwendet. Diese Wahl erleichterte es der Kamera, genau zu erkennen, wo ein Zug gemacht wurde, da sie die Farben Schwarz und Rot klar unterscheiden konnte. Diese klare Unterscheidung ermöglichte es der Kamera, präzise die Positionen der Spielsteine zu erfassen und dem Hauptblock genaue Informationen über den Spielverlauf zu übermitteln. Dieses innovative Konzept trug wesentlich dazu bei, die Leistung und Genauigkeit des Roboters beim Spielen von Tic-Tac-Toe deutlich zu verbessern.

Abbildung 3. Verwendete Kamera «Logitech»

IV. Programmierung

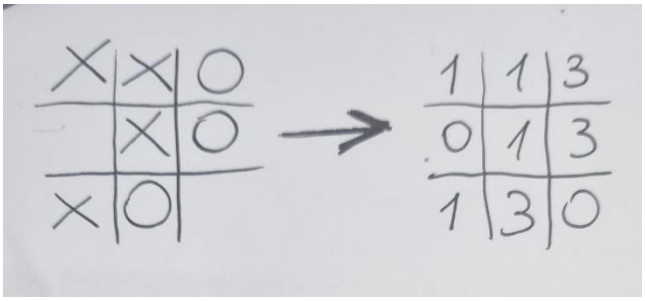


Abbildung 4. Idee der Analyse

Der erste Schritt bestand darin, die Motoren so zu programmieren, dass sie das Spielfeld zeichnen konnten. Dazu wurde eine Befehlsfolge verwendet, wobei die Bezeichnungen der Motoren, ihre Aktionen und die Koordinaten (z. B. die Länge der gezeichneten Linien oder die Position des Stifts auf dem Roboter) variiert wurden. Diese Programmierung stellte sicher, dass eine klare und deutliche Darstellung des Spielfeldes möglich war. Die Koordinaten wurden sorgfältig festgelegt, um sicherzustellen, dass das Spielfeld korrekt und symmetrisch gezeichnet wurde. Abschnitt des verwendeten Codes:

```
[ motor = NXTMotor('A', 'Power',
50, 'TachoLimit', 200);

motor.SendToNXT();

motor.WaitFor(); ]
```

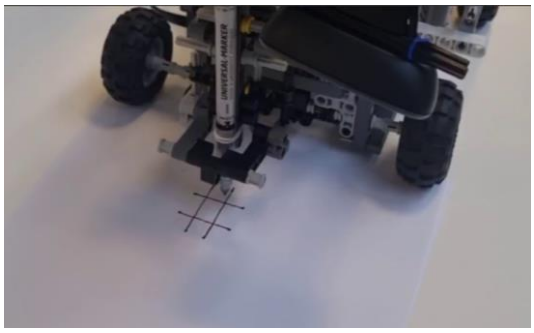


Abbildung 5. Der Roboter zeichnet das Spielfeld.

Im zweiten Schritt erfolgte die Programmierung der Kamera, um das Spielfeld zu erkennen. Nachdem der Spieler seinen Zug gemacht hatte, nahm die Kamera ein Foto auf an und analysierte es, um die Position des Spielerzugs zu bestimmen. Dies erfolgte durch den Einsatz von Farberkennungsalgorithmen zur Identifizierung der roten Farbe des Spielers. Nachdem die Position des Zuges des Spielers erkannt war, musste der Roboter programmiert werden, um zu entscheiden, in welchem von neun möglichen Bereichen sich der Zug befand. Dadurch konnte der Roboter seinen eigenen Zug planen und ausführen, basierend auf dem erkannten Spielzustand und der Strategie des Algorithmus.

Die optimale Lösung bestand darin, das aufgenommene Foto in 9 Bereiche zu zerlegen und zu bestimmen, wo sich der Spielerzug befindet. Aufgrund der unvollkommenen Genauigkeit der Zeichnung des Spielfeldes war es jedoch nicht einfach, genaue Koordinaten zu bestimmen

Die beschriebenen Schritte zeigen, dass der Roboter regelmäßig Schwierigkeiten hatte, die genaue Position des Spielerzuges zu erkennen.

V. Idee für die Logik des Roboters

Leider konnte die innovative Lösung für die Roboterlogik aufgrund von Schwierigkeiten mit der Kamera nicht vollständig genutzt werden. Die Grundidee bestand darin, das Spielfeld als 3×3-Matrix in MATLAB abzubilden, wobei leere Felder mit 0, Spielerzüge mit 3 und Roboterzüge mit 1 gekennzeichnet wurden. Dann wurden bestimmte Kombinationen gesucht, bei denen die Summe entlang der Diagonalen, Spalten oder Zeilen bestimmte Werte erreichte. Zum Beispiel könnte eine Summe von 6 bedeuten, dass zwei Nullen in einer bestimmten Zeile stehen und der Roboter daher ein Kreuz machen muss, um den Spieler am Gewinnen zu hindern.

Literaturverzeichnis

[1] MathWorks Help Center, 1994-2024

https://de.mathworks.com/help/imaq/imaqdevice

[2] Jeremy Hausotter:

How to Win Tic Tac Toe Like a Champion, Januar 2020

https://books.google.de/books/about/How_to_Win_Tic_Tac_Toe_Like_a_Champion.html?id=gxNuzQEACAAJ&redir_esc=y

[3] Abbildung 2: Lego Mindstorms NXT Wikipedia https://commons.m.wikimedia.org/wiki/File:Lego_mindstorms_nxt_main_brick.jpg

[4] Abbildung 3: Logitech-Website https://www.logitech.com/de-at/search.html?q=c270

Tic-Tac-Toe-Spieler

Denys Malovanyi, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

Zusammenfassung **- Während eines jährlichen Technik-Seminars wurde ein lang ersehntes Projekt realisiert, das aufgrund begrenzter Ressourcen zuvor als nicht durchführbar erschien: die Konstruktion eines Tic-Tac-Toe-Roboters. Mithilfe von LEGO Mindstorms-Bauteilen, einem Controller und einer Kamera, wird ein funktionsfähiger Prototyp entwickelt und getestet. Der Roboter wurde entworfen, um als interaktiver Spieler in Tic-Tac-Toe Spiel zu agieren. Im Laufe des Projekts wurde klar, dass der Roboter mit den Verbesserungen und Anpassungen großes Potenzial hat. Die nachfolgenden Abschnitte bieten eine detaillierte Darstellung der technischen Spezifikationen des Projekts sowie die Vorschläge über seine möglichen Anwendungen und Entwicklungsmöglichkeiten.**

I. Einleitung

Während des LEGO Mindstorms-Praktikums war es eine große Herausforderung, den engen Zeitrahmen für das Projektmanagement zu bewältigen und gleichzeitig die Vorbereitung auf Klausuren zu meistern. Diese Prüfungen nahmen einen erheblichen Teil der ohnehin schon begrenzten Zeit in Anspruch und erforderten eine effiziente Zeitplanung, um sowohl das Projekt als auch die akademischen Verpflichtungen erfolgreich zu erfüllen. Trotz dieser zusätzlichen Belastungen war es möglich, fokussiert zu bleiben und die Ressourcen optimal zu nutzen. Jede verfügbare Minute wurde effizient genutzt und kontinuierlich auf die wichtigsten Aspekte des Projekts konzentriert, was entscheidend für den Erfolg war. Letztendlich führte diese Herangehensweise dazu, dass das Roboterprojekt innerhalb von nur zwei Wochen erfolgreich abgeschlossen werden konnte.

II. Vorbetrachtungen

Die Grundidee besteht darin, dass der Roboter zunächst das Spielfeld für Tic-Tac-Toe zeichnet, einen ersten Zug ausführt und dann vorübergehend zur Seite fährt, um dem menschlichen Spieler die Gelegenheit zu geben, seinen Zug zu machen. Sobald der Spieler seinen Zug gemacht hat, kehrt der Roboter zurück und übernimmt die Analyse sowohl seiner eigenen Züge als auch der des Spielers. Auf Basis dieser Analyse setzt der Roboter dann seine eigenen Spielsteine (Kreuze) in den vom Algorithmus ausgewählten Bereich des Spielfelds.

Um diese ambitionierte Idee erfolgreich umzusetzen, sind einige wichtige Komponenten und Funktionen erforderlich. Zunächst benötigt der Roboter eine spezielle Vorrichtung oder Mechanik, um das Spielbrett präzise zu zeichnen. Darüber hinaus sind Sensoren von entscheidender Bedeutung, um die Position der Spielsteine des menschlichen Spielers zu erkennen. Ein hochentwickelter Algorithmus ist erforderlich, um die Spielzüge zu analysieren und dem Roboter eine Strategie für seine eigenen Züge vorzuschlagen.

Es ist wichtig, dass der Roboter nicht nur in der Lage ist, die Spielzüge zu erkennen und zu analysieren, sondern auch über die Fähigkeit verfügt, eigene Spielzüge zu planen und auszuführen. Diese Fähigkeit erfordert eine komplexe Steuerungslogik und präzise Bewegungen des Roboters. Darüber hinaus könnte eine intuitive Benutzeroberfläche für den menschlichen Spieler von Vorteil sein, um den Spielstand leicht verfolgen zu können und eine reibungslose Interaktion zwischen Mensch und Roboter zu gewährleisten.
Insgesamt erfordert die Umsetzung dieser Idee eine umfassende Planung und Entwicklung, bei der verschiedene technologische Komponenten nahtlos zusammenarbeiten müssen, um ein reibungsloses Spielerlebnis zu gewährleisten. Es ist wichtig, dass der Roboter nicht nur technisch zuverlässig ist, sondern auch eine benutzerfreundliche Erfahrung bietet, um das Spiel für alle Beteiligten unterhaltsam zu gestalten kann.

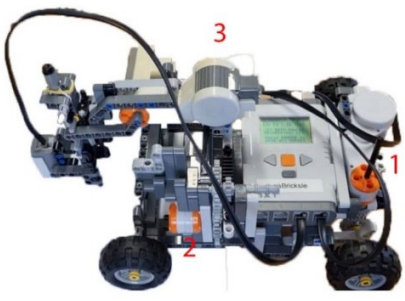


Abbildung 1. Konstruktion des Roboters.

DOI: 10.24352/UB.OVGU-2024-018

III. Grundlegende Strukturelemente

Die Bewegung des Roboters wurde durch drei Motoren realisiert. Erster Motor (Abbildung 1, Motor 1) befand sich hinten am Roboter und steuerte das Vorwärts- und Rückwärtsfahren. Der zweite Motor (Abbildung 1, Motor 2) befand sich vorne unten am Roboter und war für die Bewegung des Stifts zuständig, der das Spielfeld zeichnete, sowie für seitliche Bewegungen des Roboters nach rechts und links. Der dritte Motor (Abbildung 1, Motor 3) bewegt den Stift, der sich oben auf der Roboterstruktur befand, wurde durch einen Mechanismus angetrieben, der es ermöglichte, den Stift anzuheben und abzusenken. Damit konnte der Roboter die Spielfiguren setzen.

Abbildung 2. NXT Hauptteil; Quelle [3]

Zur Steuerung der Motoren wurde ein NXT-Controller (Abbildung 2) verwendet, der alle Hauptkomponenten des Roboters, einschließlich Sensoren und Motoren, mit Strom versorgte. Die Energieversorgung wurde durch einen Akku sichergestellt. Danach wurden alle Komponenten so programmiert, dass sie spezifische Aufgaben erfüllten.

Abbildung 3. Links - Ultraschallsensor; Rechts - Farbensensor; Quelle [4]

Zu Beginn wurden zwei Sensoren verwendet - ein Positionssensor (Abbildung 3; links) und ein Farbsensor (Abbildung 3; rechts). Der Positionssensor sollte die Bewegungen des Roboters koordinieren, indem er den Raum vor ihm scannte, während der Farbsensor, der über dem Marker hing, die Züge des Roboters und des Spielers erkennen und die Informationen an den Hauptblock übertragen sollte. Hier traten jedoch Probleme auf, da der Sensor, der über dem Marker hing, das Spielfeld schlecht und langsam erkannte und es schwierig war, ihn zu programmieren. Daher wurde beschlossen, diese beiden Sensoren durch eine herkömmliche Kamera zu ersetzen und sie durch MATLAB zu programmieren, um eine effektivere Erkennung und Verarbeitung der Spielfeldinformationen zu ermöglichen. Die Kamera erwies sich als weitaus zuverlässiger und genauer bei der Erkennung der Positionen und Farben der Spielsteine, was zu einer verbesserten Leistungsfähigkeit und schnelleren Reaktionszeiten des Roboters führte. Durch die Verwendung von MATLAB konnte eine detaillierte Bildverarbeitung implementiert werden, die es dem Roboter ermöglichte, die Spielsituation in Echtzeit zu analysieren und entsprechend zu reagieren. Dies stellte sicher, dass der Roboter in der Lage war, die Spielzüge sowohl des menschlichen Spielers als auch seine eigenen präzise zu verfolgen und umzusetzen, wodurch die Gesamtfunktionalität des Systems erheblich verbessert wurde.

Die hohe Genauigkeit der Kamera und die fortgeschrittenen Bildverarbeitungsfähigkeiten von MATLAB verbesserten die Systemleistung erheblich. Die Integration von MATLAB ermöglichte eine präzisere Analyse der Bilddaten, was zu effizienterer Erkennung und Verarbeitung führte. Dadurch konnte der Roboter die Spielsituation stets korrekt erfassen und angemessen darauf reagieren, was zu einer optimierten und leistungsfähigeren Lösung führte.

Abbildung 4. Die verwendete Kamera «Logitech»; Quelle [5]

Für die Kamera (Abbildung 4) wurde die Entscheidung getroffen, dass der Roboter das Spielfeld und seine eigenen

Züge mit einem schwarzen Marker zeichnete, während der Spieler mit einem roten Marker spielte. Dies erleichterte es der Kamera, zu erkennen, wo genau ein Zug gemacht wurde, da sie die Kontrastfarben Schwarz und Rot klar unterscheiden konnte. Dadurch konnte die Kamera präziser die Positionen der Spielsteine erfassen und dem Hauptblock genaue Informationen über den Spielverlauf übermitteln. Dieses innovative Konzept trug dazu bei, die Leistung und Genauigkeit des Roboters beim Spielen von Tic-Tac-Toe deutlich zu verbessern.

IV. Programmierung

Der Entwicklungsprozess begann mit der Programmierung der Motoren, um das Spielfeld (Abbildung 5) zu zeichnen. Dabei wurde dieselbe Sequenz von Befehlen verwendet, wobei lediglich die Motorenbezeichnungen, ihre Aktionen und die Koordinaten variierten (wie z.B. die Länge der gezeichneten Linien oder die Position des Markers auf dem Roboter). Diese konsistente Programmierung ermöglichte eine einheitliche und präzise Darstellung des Spielfelds. Die Koordinaten wurden festgelegt, um sicherzustellen, dass das Spielfeld korrekt und symmetrisch gezeichnet wurde und dass die Bewegungen des Roboters reibungslos verliefen. Die genaue Festlegung der Koordinaten spielte eine entscheidende Rolle dabei, dass der Roboter die Linien in der richtigen Länge und an den richtigen Positionen zog. Um dies zu gewährleisten, wurde eine Reihe von Tests und Kalibrierungen durchgeführt, um sicherzustellen, dass die gezeichneten Linien genau den gewünschten Spezifikationen entsprachen. Diese präzise Vorgehensweise sorgte dafür, dass das Spielfeld nicht nur ästhetisch ansprechend, sondern auch funktional für das Tic-Tac-Toe-Spiel geeignet war. Darüber hinaus wurden die Motoren so programmiert, dass ihre Bewegungen geschmeidig und kontrolliert abliefen, wodurch ein gleichmäßiger und fehlerfreier Betrieb ermöglicht wurde. Diese detaillierte Programmierung und sorgfältige Kalibrierung führten letztendlich zu einem hochpräzisen und zuverlässigen System, das in der Lage war, das Spielfeld exakt so zu zeichnen, wie es für das erfolgreiche Spielen des Tic-Tac-Toe-Spiels erforderlich war.Abschnitt des verwendeten Codes (Quelle [1]):

```
[ motor = NXTMotor('A', 'Power', 50, 'TachoLimit', 200)
motor.SendToNXT();
motor.WaitFor(); ]
```

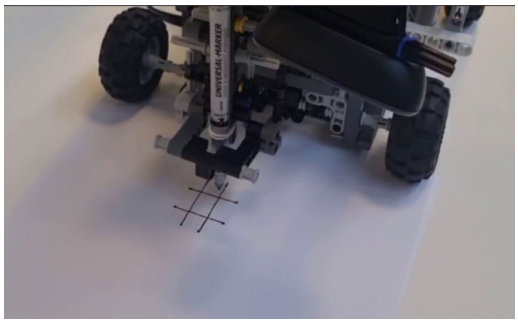


Abbildung 5. Der Roboter zeichnet das Spielfeld.

Im zweiten Abschnitt der Entwicklung richteten geht es über die Implementierung eines Bildverarbeitungsmoduls zur automatisierten Erfassung und Analyse der Spielzüge. Nachdem ein Spieler seinen Zug gemacht hatte, wurde ein Foto des Spielfelds von der Kamera aufgenommen. Dieses Bild wurde dann mithilfe spezifischer Farberkennungsalgorithmen verarbeitet, um die Position des Spielerzugs zu identifizieren. Die Algorithmen waren darauf ausgelegt, die charakteristische rote Farbe des Spielers zu erkennen und die Position des Zuges präzise zu bestimmen.

Im Anschluss an die Identifizierung des Spielerzugs durch die Bildverarbeitungssoftware wurde der Roboter programmiert, um den Spielzustand zu analysieren und seinen eigenen Zug strategisch zu planen. Dabei war es entscheidend, den Spielbereich, in dem sich der Spielerzug befand, präzise zu bestimmen, um eine optimale Reaktion zu gewährleisten. Dieser Prozess erforderte eine sorgfältige Koordination zwischen der Kamera, der Bildverarbeitungssoftware und den Steuerungsalgorithmen des Roboters.

Im Verlauf der Entwicklung traten immer wieder Herausforderungen auf. Insbesondere die exakte Bestimmung der Spielerzugposition stellte sich als komplex heraus, da das Spielfeld nicht immer perfekt gezeichnet war und kleine Abweichungen auftreten konnten. Darüber hinaus war die Platzierung der Kamera auf dem Roboter von entscheidender Bedeutung, um die Genauigkeit der Bildaufnahmen zu maximieren und potenzielle Fehlerquellen zu minimieren.

Insgesamt zeigt dieser Abschnitt deutlich die technologischen Herausforderungen und die strategische Planung, die bei der Entwicklung eines automatisierten Systems zur Spielzugerkennung und -reaktion bewältigt werden mussten. Trotz der begrenzten Ressourcen und des Zeitdrucks arbeitete das Team hart daran, eine zuverlässige Lösung zu entwickeln, die die Anforderungen des Projekts erfüllte.

V. Idee für Logik des Roboters

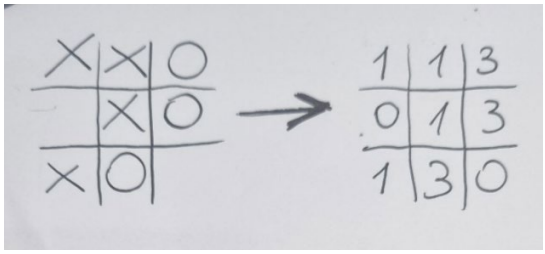


Abbildung 6. Die Idee des Logiks; Quelle [2]

Bedauerlicherweise blieb die innovative Lösung zur Steuerung der Spiellogik des Roboters ungenutzt, da Schwierigkeiten mit der Kamera die Implementierung behinderten. Das zugrunde liegende Konzept war äußerst vielversprechend: Es

sah vor, das Spielfeld in Form einer 3×3-Matrix (Abbildung 6) in der Programmiersprache MATLAB abzubilden.
Dabei wurden leere Felder als 0, Spielerzüge als 3 und Roboterzüge als 1 markiert. Anschließend sollte nach bestimmten Kombinationen gesucht werden, bei denen die Summe entlang der Diagonalen, Spalten und Zeilen bestimmte Werte erreichte. Diese Summen würden wiederum die Strategie des Roboters bestimmen.
Beispielsweise könnte eine Summe von 6 bedeuten, dass sich zwei Nullen in einer bestimmten Linie befinden, und demzufolge müsste der Roboter einen Zug setzen, um zu verhindern, dass der Spieler gewinnt. Trotz des vielversprechenden Ansatzes konnte diese ausgefeilte Logik aufgrund des Zeitmangels nicht in die Praxis umgesetzt werden.

LITERATURVERZEICHNIS

[1] MathWorks Help Center, 1994-2024
https://de.mathworks.com/help/imaq/imaqdevice
[2] Jeremy Hausotter:
How to Win Tic Tac Toe Like a Champion, Januar 2020
https://books.google.de/books/about/How_to_Win_Tic_Tac_Toe_Like_a_Champion.html?id=gxNuzQEACAAJ&redir_esc=y
[3] Abbildung 2: Lego Mindstorms NXT Wikipedia
https://commons.m.wikimedia.org/wiki/File:Lego_mindstorms_nxt_main_brick.jpg
[4] Abbildung 3: Farbensensor
https://lego.fandom.com/de/wiki/Farbsensor_10286
Ultraschallsensor
https://wiki.hshl.de/wiki/index.php/Ultraschall_mit_Matlab/Simulink
[5] Abbildung 4: Logitech-Website
https://www.logitech.com/de-at/search.html?q=c270

Einparkroboter

Jannik Findeklee, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im folgendem Paper geht es über den Prozess der Entwicklung und die Funktion des LEGO-NXTs bis hin zum fertigen Produkt eines Einparkroboters. Dieser kann mithilfe von Ultraschallsensoren selbstständig Parklücken aufspüren und mittels eines Algorithmus in diese rückwärts einparken. Es werden sowohl Probleme als auch Errungenschaften näher beleuchtet, die zum Entstehen von dem Endresultat geführt haben. Am Ende folgen noch weitere Verbesserung Möglichkeiten, die zum Verbessern der Funktionen des Einparkroboters führen könnten.**

***Schlagwörter*—Autonom, Einparken, LEGO Mindstorms, Otto-von-Guericke-Universität, Projektseminar, Ultraschallsensor**

I. Einleitung

IM Jahr 2024 ist die Welt der Robotik und Technologie immer weiter vorangeschritten. Passend zu diesem Fortschritt der Menschheit findet auch dieses Jahr wieder das Projektseminar LEGO Mindstorm statt, um den Studierenden der Otto-von-Guericke-Universität die Welt der Robotik und des autonomen Lebens näher zu bringen. Innerhalb eines zweiwöchigen Praktikums entwickelten 17 verschiedene Gruppen unterschiedlichste Ideen und bauten diese mit Hilfe von LEGO-NXT. Zu Beginn des Praktikums wurden alle mit dem Programm MATLAB vertraut gemacht. Kurz darauf bekam jede Gruppe ihr eigenes LEGO-Mindstorm-Set mit den unterschiedlichsten LEGO-Teilen wie Motoren, Ultraschallsensoren, Tastern oder auch Farb- und Lichtsensoren. Mit Hilfe dieser Sets hatten alle Schülerinnen und Schüler die Möglichkeit, ihren eigenen LEGO-Roboter zu bauen und zu programmieren. Roboter sind heutzutage für jeden Menschen unentbehrlich, sei es zum Beispiel ein Handy oder eben ein Einparkassistent, sie alle sind erst durch den Fortschritt der immer weiter voranschreitenden Technik möglich geworden. Außerdem können Roboter die Arbeit des Menschen viel effizienter, genauer und länger erledigen, weshalb sie bereits in vielen verschiedenen Bereichen eingesetzt werden. Zum Beispiel in der Automobilindustrie, wo Roboter mehrere hundert Kilogramm schwere Autos selbstständig bewegen, was ein Mensch niemals schaffen würde. Außerdem ist fast der gesamte Prozess der Autoherstellung durch verschiedene Roboter automatisiert, was vor einigen Jahrzehnten noch unvorstellbar war. In vielen neuen Automodellen werden immer bessere Assistenzsysteme angeboten, wie zum Beispiel der Einparkassistent, der den kompletten Einparkvorgang übernimmt. Der Begriff Assistent ist hier nicht ganz zutreffend, da es sich nicht mehr um Assistenz, sondern um Automatisierung handelt, weshalb der Roboter auch als Einparkroboter bezeichnet wird.

DOI: 10.24352/UB.OVGU-2024-019

II. Vorbetrachtungen

A. NXT

Der NXT ist ein programmierbarer Teil des LEGO Mindstorm Sets, mit dem ein Roboter gebaut und programmiert werden kann. Er wird zu Bildungszwecken eingesetzt und ermöglicht es jedem, seinen eigenen Roboter zu programmieren. Der NXT hat vier Eingänge für Sensoren und drei Ausgänge für Motoren. Er kann sowohl über Kabel als auch über Bluetooth angeschlossen werden, was die Steuerung auch über größere Entfernungen ermöglicht. Mit dem NXT sind Aktionen und Reaktionen möglich, z. B. ertönt nach dem Drücken einer Taste ein Signal oder ein Motor beginnt sich zu drehen.

Abbildung 1. NXT 2.0

B. Ultraschallsensor

Das Prinzip der Ultraschall-Sensoren ist im Wesentlichen immer gleich (Abbildung 2):

1) Erzeugung von Schallwellen: Der Ultraschallsensor sendet hochfrequente Schallwellen aus, die für das menschliche Ohr nicht hörbar sind. Diese Schallwellen breiten sich im Raum aus. In Abbildung 2 rot dargestellt (Echo).
2) Reflexion der Schallwellen: Treffen die Schallwellen auf einen Gegenstand, wird ein Teil der Schallwellen von diesem reflektiert. Je nach Oberfläche und Beschaffenheit des Objektes wird ein Teil der Schallwellen zurück zum Sensor reflektiert, während der Rest absorbiert oder gestreut wird.
3) Laufzeitmessung: Der Ultraschallsensor misst die Zeit, die die reflektierten Schallwellen benötigen, um zum Sensor zurückzukehren. In Abbildung 2 ist dies grün dargestellt (Trig). Da die Schallgeschwindigkeit bekannt ist (ca. 343 Meter pro Sekunde bei Raumtemperatur), kann der Sensor aus der Laufzeit berechnen, wie weit das Objekt entfernt ist.
4) Umrechnung in Entfernung: Durch die Messung der Laufzeit und die Kenntnis der Schallgeschwindigkeit

kann der Sensor die Entfernung zum Objekt berechnen. Diese Entfernung wird dann in einer für den Benutzer verständlichen Form in Zentimetern ausgegeben.

C. Tastsensor

Der Tastsensor, in Abbildung 1 unten links zweimal zu sehen, funktioniert folgendermaßen. Es wird dauerhaft ein Impuls vom NXT zum Tastsensor gesendet, der für eine Nachfrage des Taster Zustandes sorgt. Der Taster-Zustand gibt an, ob der Taster gedrückt wird oder nicht. Hierbei gibt der Sensor eine 0 für ungedrückten Zustand, und eine 1 für den gedrückten Zustand zurück, der dann an den NXT zurückgesendet wird.

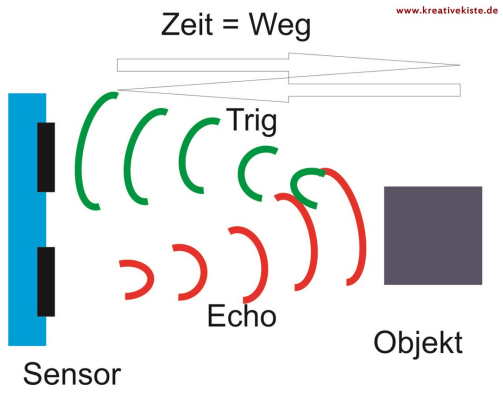


Abbildung 2. Funktion des Ultraschallsensor

III. Umsetzung

A. Verworfenes Konzept

Das erste Konzept sollte sich vom realistischen Einparkassistenten stark unterscheiden. Hierbei besaß das Fahrzeug acht anstelle der normalen vier Räder. Von diesen Rädern waren jeweils vier nach vorne beziehungsweise nach hinten, und vier nach rechts beziehungsweise links gerichtet (Abbildung 3). Das sorgte dafür, dass das Fahrzeug nicht lenken konnte, aber dafür in deutlich kleinere Parklücken einfahren konnte.
Der Ablauf des Einparkvorganges ging wie folgt. Als erstes waren die Rechts-Links-Räder in der Luft, bis das Fahrzeug direkt neben einer Parklücke stand. Danach drückte ein Motor die Rechts-Links-Räder nach unten, sodass die Vor-Zurück-Räder in der Luft waren. Hierbei entstanden die ersten Probleme, da die Vor-Zurück-Räder auch den Rahmen des Fahrzeugs hielten und somit deutlich zu schwer für den Motor beziehungsweise für die Zahnräder wahren. Das sorgte dafür, dass immer ein oder zwei Räder auf dem Boden schliffen und somit dafür sorgten, dass das Fahrzeug nicht gerade in die Parklücke fuhr. Nach Stundenlanger Fehlerbehebung und keinem besseren Resultat wurde dieses Konzept verworfen und ein realistischerer und mechanisch simplerer Einparkroboter entwickelt.

B. Mechanische Konstruktion

Der Einparkroboter verwendete zwei Motoren und zwei Sensoren. Ein Motor diente zum Antrieb der Vorwärts- und

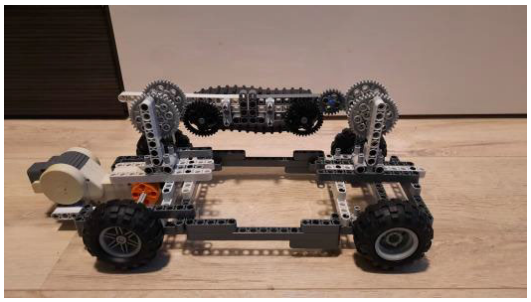


Abbildung 3. Konstruktion des verworfenen Konzepts

Rückwärtsbewegung, während der andere Motor die Servolenkung bewegte. Es wurden ein Ultraschallsensor und ein Tastsensor verwendet. Der Ultraschallsensor (Abbildung 4 rechts) war an der rechten Seite des Fahrzeugs angebracht und zeigte in Richtung der parkenden Autos. Der Tastsensor (Abbildung 4 links) war am Heck des Fahrzeugs angebracht und zeigte nach hinten. Der NXT wurde in der Mitte montiert, um den Schwerpunkt in die Mitte des Fahrzeugs zu verlegen, was für das Manövrieren von Vorteil war.

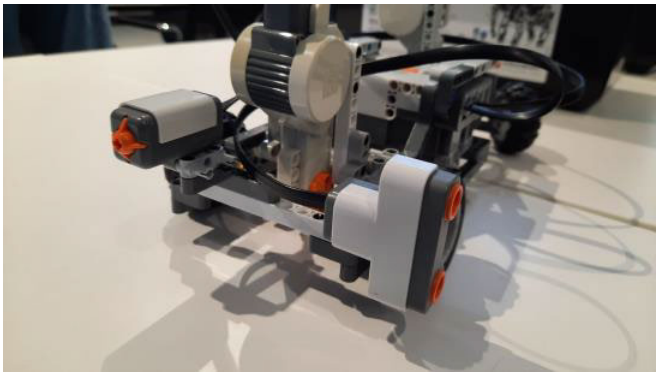


Abbildung 4. Sensoren an der Rückseite

C. Servolenkung (Abbildung 5)

Der Einparkroboter arbeitet mit einer Servolenkung, um den Einparkvorgang durchzuführen. Diese bestand aus einem Motor, einem Zahnrad und einer Zahnstange. Der Motor drehte das Zahnrad, das wiederum die Zahnstange nach links und rechts bewegte. Diese Konstruktion ermöglichte einen Lenkeinschlag von 150°.

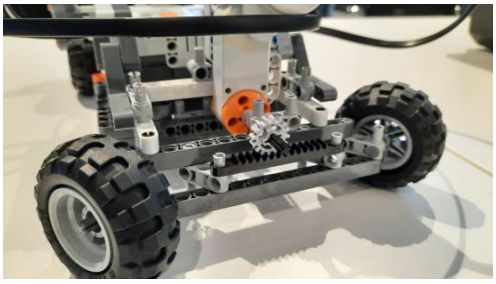


Abbildung 5. Servolenkung

D. Programmablauf

Abbildung 6 zeigt ein vereinfachtes Flussdiagramm des Quelltextes, das den Vorgang des Einparkens beschreibt.

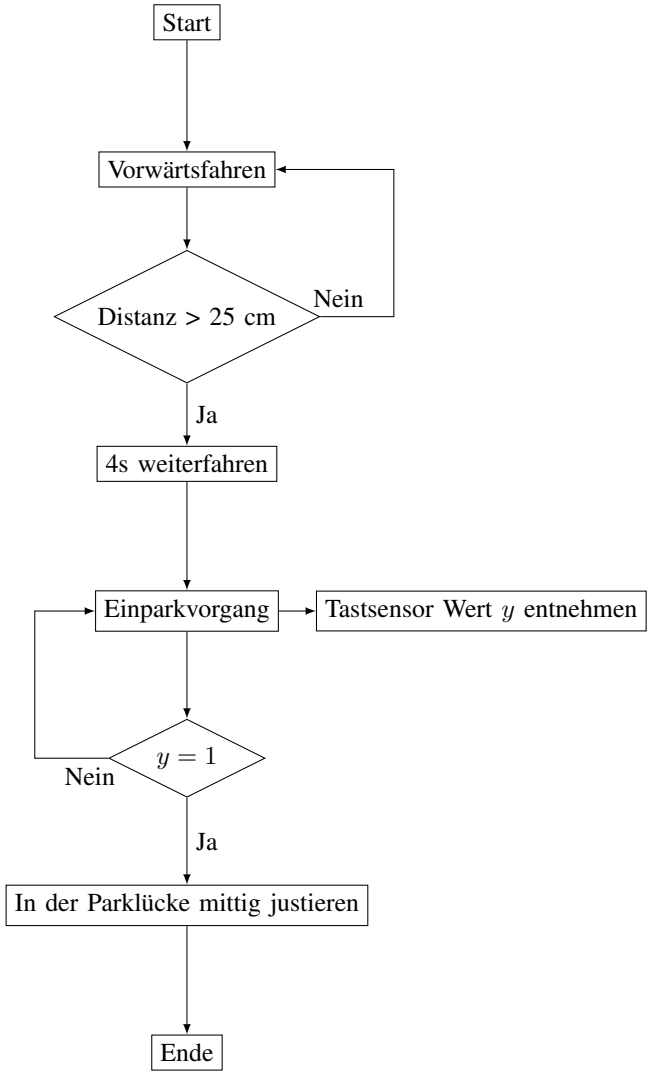


Abbildung 6. Konstruktion des verworfenen Konzepts

IV. Ergebnisdiskussion

Beim Bau und der Programmierung traten Probleme auf, die teilweise umgangen werden mussten, anstatt sie zu lösen. Das erste Konzept war sehr zeitaufwendig und nahm kostbare Zeit vom Bau des Endprodukts in Anspruch. Daher musste das Endprodukt simpel bleiben, um keine komplizierten Probleme aufzuwerfen. Das zweite Konzept, beziehungsweise das Endkonzept, war deutlich einfacher umzusetzen und hat in diesem Aspekt keine Probleme aufgeworfen. Ein Problem im Quellcode war, dass der Taster nur beim Rückwärtsfahren als Notaus funktioniert hat. Daher wurde er umgewandelt, um das Fahrzeug bei Kollision zu stoppen. Es wäre sinnvoller gewesen, dies mit einem Ultraschallsensor zu lösen, da dieser das Fahrzeug kurz vor einer Kollision stoppen würde. Das ist besonders bei einem Einparkroboter von Vorteil. Außerdem erkennt das Fahrzeug nicht, wenn die Lücke zu klein ist und versucht, in jeden noch so kleinen Spalt einzuparken. Das Endprodukt enthält jedoch die zuvor genannten Eigenschaften und entspricht somit unseren Erwartungen größtenteils. Der Einparkroboter kann nämlich selbstständig fahren, Parklücken aufspüren und einparken.

V. Zusammenfassung und Fazit

Am Ende des zweiwöchigen Projektseminars wurde ein funktionsfähiger LEGO-Roboter gebaut. Dieser kann die zuvor genannten Aufgaben des Einparkroboters lösen. Er findet selbstständig eine Parklücke, platziert sich einparkbereit und parkt ein. Zur Verbesserung könnten mehrere Ultraschallsensoren beitragen, die das Einparken auf der linken Seite in Fahrtrichtung ermöglichen. Durch größere Änderungen am Quelltext wäre es möglich gewesen, einen Spurhalteassistenten und die Erkennung von engen Lücken zu integrieren. Außerdem sollte der Notstopp-Taster durch einen Ultraschallsensor ersetzt werden, um den Einparkvorgang sicherer und zuverlässiger zu gestalten. So wäre das Einparken in unterschiedlichste Parklücken möglich. Das Wichtigste wäre jedoch eine bessere Zeitplanung gewesen.

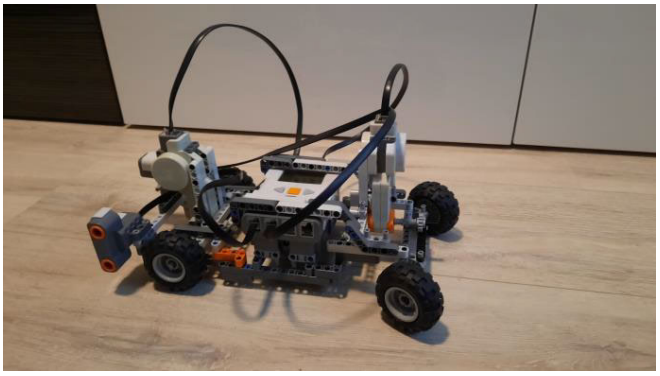


Abbildung 7. Finale Konstruktion

VI. Bildverzeichnis

[1] Wikimedia, Abbildung 1. NXT 2.0 https://commons.wikimedia.org/w/index.php?curid=19070424 (Abruf: 03.06.2024)

[2] Kreativkiste, Abbildung 2.Funktion des Ultraschallsensor https://www.kreativekiste.de/ultraschall-abstandsmessung-einparkhilfe-arduino (Abruf: 21.02.2024)

Einparkroboter

Daksh Verma, Mechatronik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In dem vorliegenden Paper wird der Entwicklungsprozess und die Funktionsweise des LEGO-NXT bis hin zur Fertigstellung eines Einparkroboters behandelt. Dieser ist in der Lage, Parklücken eigenständig mithilfe von Ultraschallsensoren zu erkennen und mithilfe eines Algorithmus rückwärts einzuparken. Es werden sowohl Herausforderungen als auch Erfolge beleuchtet, die zur Entstehung des Endprodukts beigetragen haben. Zum Schluss werden weitere Möglichkeiten zur Verbesserung diskutiert, die die Funktionen des Einparkroboters weiter optimieren könnten.**

***Schlagwörter*—Autonom, Einparken, LEGO Mindstorm, Otto-von-Guericke-Universität, Projektseminar, Ultraschallsensor**

I. Einleitung

IM Jahr 2024 hat sich die Robotik und technologische Entwicklung signifikant weiterentwickelt. An der Otto-von-Guericke-Universität wird erneut das LEGO-Mindstorms-Projektseminar angeboten, um den Studierenden Einblicke in die Robotik und das autonome Leben zu vermitteln. Während eines zweiwöchigen Workshops entwickelten 17 Teams eine Vielfalt an Ideen, die sie mit LEGO NXT umsetzten. Alle Teilnehmenden wurden zunächst in die Benutzung von MATLAB eingeführt. Danach erhielt jedes Team ein eigenes LEGO-Mindstorms-Set, ausgestattet mit einer breiten Palette an LEGO-Komponenten wie Motoren, Ultraschallsensoren, Schaltern sowie Farb- und Lichtsensoren. Diese Kits ermöglichten es den Teilnehmerinnen und Teilnehmern, individuelle LEGO-Roboter zu konstruieren und zu programmieren. In der heutigen Zeit sind Roboter aus dem menschlichen Alltag nicht mehr wegzudenken, ob als Mobiltelefon oder Einparkhilfe – all dies wurde durch den kontinuierlichen technologischen Fortschritt möglich. Darüber hinaus können Roboter menschliche Arbeit deutlich effizienter, präziser und über längere Zeiträume hinweg ausführen, was ihren Einsatz in zahlreichen Branchen begründet. Ein Beispiel hierfür ist die Automobilindustrie, in der Roboter Autos, die mehrere hundert Kilogramm wiegen, autonom bewegen können – eine Aufgabe, die für Menschen unerreichbar ist. Fast der gesamte Herstellungsprozess von Autos ist heutzutage durch den Einsatz verschiedener Roboter automatisiert, eine Vorstellung, die vor einigen Jahrzehnten noch undenkbar war. Zudem bieten viele der neuesten Fahrzeugmodel zunehmend fortgeschrittenere Assistenzsysteme wie zum Beispiel den Einparkassistent an, der den kompletten Einparkvorgang übernimmt. Die Bezeichnung Assistent trifft in diesem Kontext nicht ganz zu, denn es handelt sich eher um Automatisierung als um Unterstützung. Deshalb wäre die Benennung als Einparkroboter passender.

DOI: 10.24352/UB.OVGU-2024-020

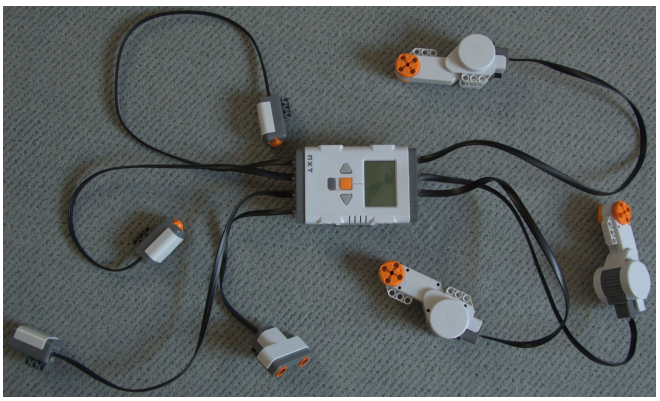


Abbildung 1. LEGO-Mindstorms-NXT-2.0-Stein mit über Kabel angeschlossenen Sensoren [1]

II. Vorbetrachtungen

A. NXT

Der NXT, ein programmierbares Element des LEGO-Mindstorms-Sets, ermöglicht das Bauen und Programmieren eigener Roboter. Hauptsächlich zu Lehrzwecken gedacht, bietet der NXT die Möglichkeit, individuelle Roboter zu erschaffen. Er verfügt über vier Sensoranschlüsse und drei Anschlüsse für Motoren. Die Verbindung kann sowohl über Kabel als auch Bluetooth erfolgen, was Fernsteuerung über weite Strecken erlaubt. Der NXT macht verschiedene Aktionen und Reaktionen möglich; so kann beispielsweise das Drücken eines Knopfes ein akustisches Signal auslösen oder einen Motor in Bewegung setzen.

B. Ultraschallsensor

Das Prinzip der Ultraschallsensoren ist immer gleich (siehe Abbildung 1 unten mitte-links):

1) Schallwellenerzeugung: Der Ultraschallsensor sendet hochfrequente Schallwellen aus, die für das menschliche Ohr nicht wahrnehmbar sind. Diese Schallwellen breiten sich im Raum aus und werden in Abbildung 2 als Sendeimpuls dargestellt.
2) Reflexion der Schallwellen: Wenn die Schallwellen auf ein Objekt treffen, wird ein Teil von ihnen vom Objekt reflektiert. Je nach Oberfläche und Beschaffenheit des Objekts wird ein Teil der Schallwellen zurück zum Sensor reflektiert, während der Rest absorbiert oder gestreut wird.
3) Zeitmessung der Schallwellen: Der Ultraschallsensor ermittelt, wie lange die reflektierten Schallwellen brauchen, um wieder beim Sensor anzukommen. Dies ist in Abbildung 2 als Echo gekennzeichnet. Mit der bekannten

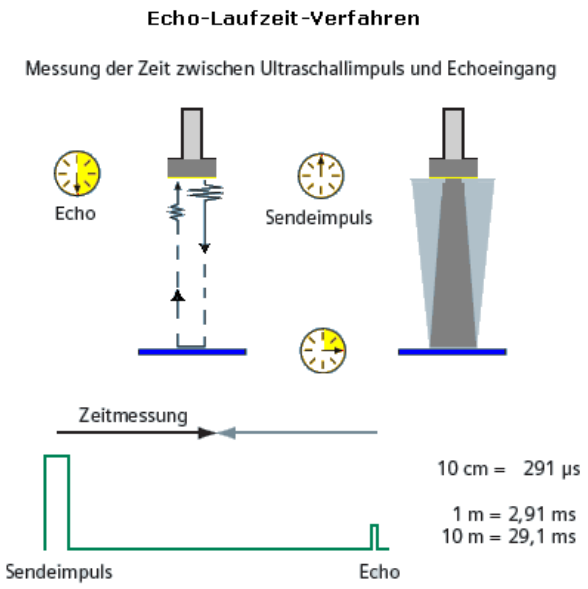


Abbildung 2. Funktion des Ultraschallsensors [2]

Schallgeschwindigkeit von etwa 343 Metern pro Sekunde bei Raumtemperatur lässt sich aus der gemessenen Zeit die Entfernung zum Objekt bestimmen.

4) Berechnung der Distanz: Mit der erfassten Laufzeit und dem Wissen über die Schallgeschwindigkeit ist es möglich, die Entfernung zum Gegenstand zu bestimmen. Die ermittelte Distanz wird anschließend in Zentimetern dargestellt, sodass sie für den Anwender nachvollziehbar ist.

C. Tastsensor

Die Funktionsweise des Tastsensors (Abbildung 1 links) ist wie folgt: Der NXT (Abbildung 1 in der Mitte) sendet kontinuierlich einen Impuls an den Tastsensor, um den aktuellen Zustand des Tasters abzufragen. Der Zustand des Tasters zeigt an, ob der Taster betätigt ist oder nicht. Dabei liefert der Sensor den Wert 0, wenn der Taster nicht gedrückt ist, und den Wert 1, wenn der Taster gedrückt ist. Diese Information wird dann an den NXT zurück übermittelt.

III. Konstruktion

A. Verworfenes Konzept

Das erste Konzept sollte sich stark vom realistischen Einparkassistenten unterscheiden. Dabei hatte das Fahrzeug nicht die üblichen vier Räder, sondern insgesamt acht. Von diesen Rädern waren jeweils vier nach vorne und vier nach hinten ausgerichtet (siehe Abbildung 4), sowie vier nach rechts und vier nach links (siehe Abbildung 3). Dies ermöglichte es dem Fahrzeug, nicht zu lenken, aber dafür in deutlich kleinere Parklücken einzufahren.

Der Ablauf des Einparkvorgangs war wie folgt: Zuerst waren die Räder für die seitliche Bewegung in der Luft, bis das Fahrzeug direkt neben einer Parklücke stand. Dann senkte

Abbildung 3. Rechts-Links-Rädermodul

Abbildung 4. Vor-Zurück-Rädermodul

ein Motor die seitlichen Räder ab, während die Räder für die Vorwärts- und Rückwärtsbewegung angehoben wurden. Dabei traten die ersten Probleme auf, da die Räder für die Vorwärts- und Rückwärtsbewegung auch den Rahmen des Fahrzeugs stützten und somit zu schwer für den Motor oder die Zahnräder waren. Dies führte dazu, dass immer ein oder zwei Räder auf dem Boden schliffen und das Fahrzeug nicht gerade in die Parklücke fuhr. Nach versuchter Fehlerbehebung ohne Ergebnisverbesserung wurde entschieden, einen realistischer umsetzbaren und mechanisch einfacheren Einparkroboter zu realisieren.

B. Mechanische Konstruktion

Der Einparkroboter verwendete zwei Motoren und zwei Sensoren. Einer der Motoren diente dem Antrieb für Vorwärts- und Rückwärtsfahrten, während der andere Motor für die Servolenkung zuständig war. Es wurden ein Ultraschallsensor und ein Tastsensor eingesetzt. Der Ultraschallsensor (siehe Abbildung 5 rechts) war an der rechten Seite des Fahrzeugs montiert und zeigte in Richtung der parkenden Autos. Der Tastsensor (siehe Abbildung 5 links) befand sich an der Rückseite und war nach hinten gerichtet. Der NXT war zentral angebracht, was den Masseschwerpunkt in die Mitte des Fahrzeugs legte und somit zu einer besseren Manövrierfähigkeit beitrug.

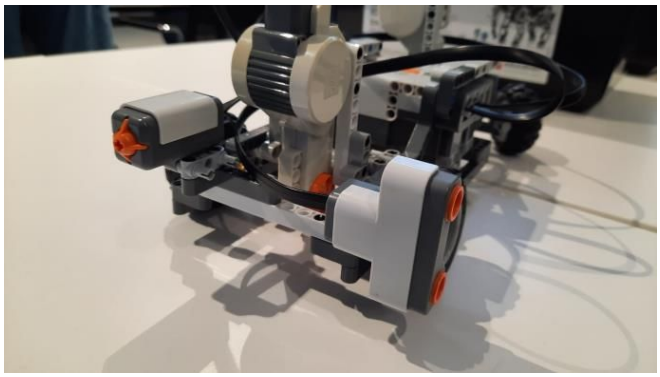


Abbildung 5. Sensoren an der Rückseite

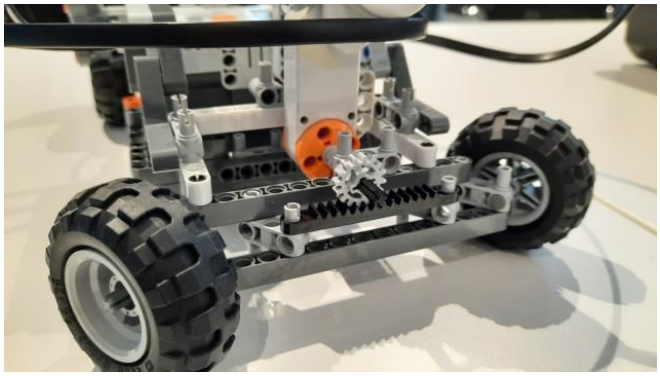


Abbildung 6. Servolenkung

C. Servolenkung

Der Einparkroboter nutzt eine Servolenkung (siehe Abbildung 6), um den Einparkvorgang durchzuführen. Diese bestand aus einem Motor, einem Zahnrad und einer Zahnstange. Der Motor drehte das Zahnrad, welches wiederum die Zahnstange nach links und rechts bewegte. Dank dieser Konstruktion konnte ein Lenkwinkel von $150°$ erreicht werden, was ein sehr gutes Manövrieren ermöglichte.

D. Programmablauf

Das Flussdiagramm in Abbildung 7 stellt den Quelltext vereinfacht dar und beschreibt den Ablauf, den der Einparkroboter durchläuft, um das Einparken zu ermöglichen.

IV. Ergebnisdiskussion

Beim Bau und der Programmierung traten verschiedene Probleme auf, die teilweise umgangen werden mussten, anstatt sie vollständig zu lösen. Das erste Konzept erwies sich als sehr zeitaufwendig und beanspruchte daher wertvolle Zeit für den Bau des Endprodukts. Aus diesem Grund musste das Endprodukt einfach gehalten werden, um keine komplexen Probleme zu verursachen. Das zweite Konzept bzw. das Endkonzept (siehe Abbildung 8) war hingegen deutlich einfacher umzusetzen und stellte in diesem Bereich keine Probleme dar. Ein Problem im Quellcode war jedoch, dass der Tastsensor ursprünglich nur als Notaus gedacht war, jedoch nur beim Rückwärtsfahren funktionierte und daher umprogrammiert

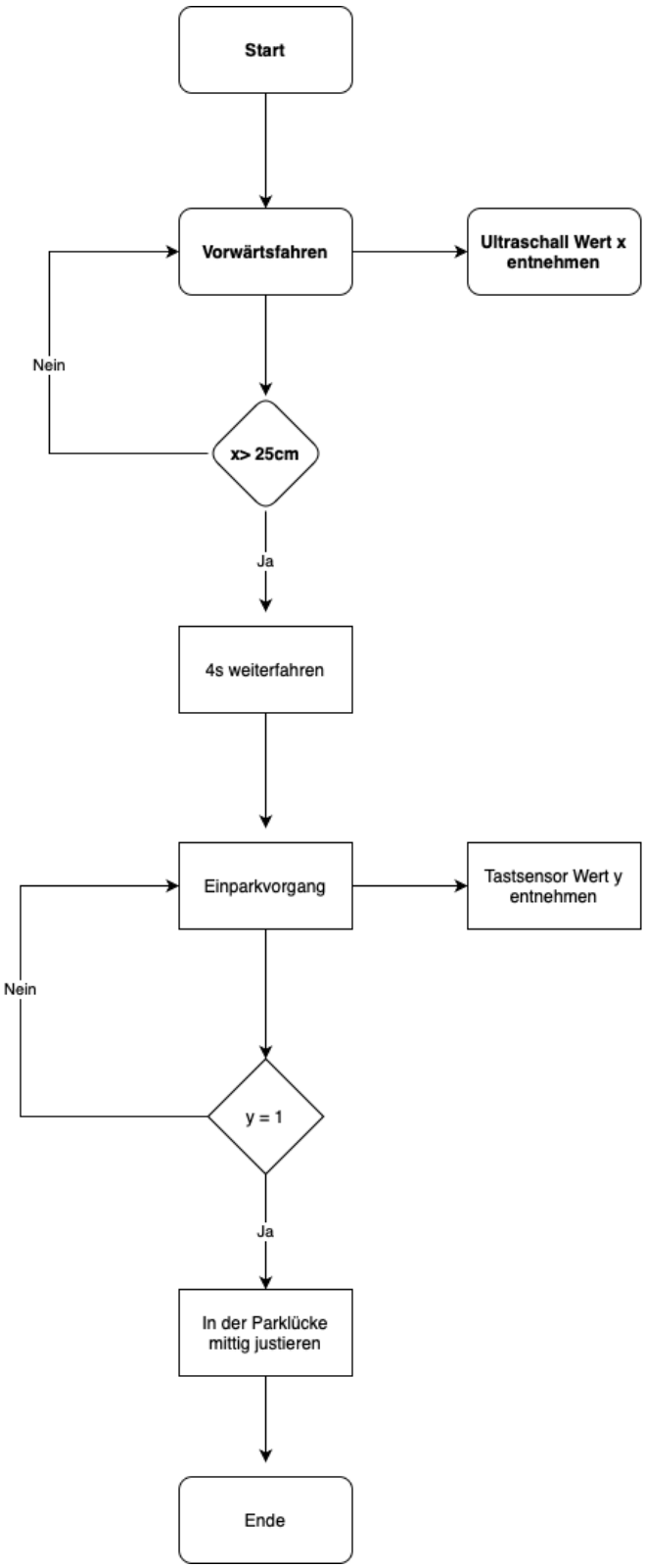


Abbildung 7. Flussdiagramm des Einparkvorgangs

Abbildung 8. Finale Konstruktion

werden musste, um das Fahrzeug bei einer Kollision zu stoppen. Dies hätte besser mit einem Ultraschallsensor gelöst werden sollen, da dieser das Fahrzeug kurz vor einer Kollision stoppen würde, was für einen Einparkroboter sinnvoller wäre. Außerdem erkennt das Fahrzeug nicht, wenn die Parklücke zu klein ist, und versucht in jede noch so kleine Lücke einzuparken. Trotzdem erfüllt das Endprodukt größtenteils die gestellten Anforderungen und entspricht somit unseren Erwartungen. Der Einparkroboter ist in der Lage, autonom zu fahren, Parklücken eigenständig zu erkennen und eigenständig einzuparken.

V. Zusammenfassung und Fazit

Am Ende des zweiwöchigen Projektseminars wurde erfolgreich ein funktionsfähiger LEGO-Roboter gebaut, der die zuvor genannten Aufgaben des Einparkroboters lösen kann. Er ist in der Lage, eigenständig eine Parklücke zu finden, sich in eine Einparkposition zu bringen und schließlich eigenständig in die Parklücke einzuparken. Zur Verbesserung des Systems könnten mehrere Ultraschallsensoren beitragen, die unter anderem das Einparken auf der linken Seite in Fahrtrichtung ermöglichen würden. Durch umfangreichere Änderungen im Quellcode wäre auch die Implementierung eines Spurhalteassistenten sowie die Erkennung zu kleiner Lücken möglich gewesen. Außerdem sollte der Notstopp-Taster durch einen Ultraschallsensor ersetzt werden, um den Einparkvorgang sicherer und zuverlässiger zu gestalten, sodass das Einparken in verschiedenste Parklücken möglich ist. Das Wichtigste wäre jedoch eine verbesserte Zeitplanung gewesen.

Literaturverzeichnis

[1] Tiger10: *Der Lego Mindstorms NXT 2.0-Stein mit über Kabel angeschlossenen Sensoren.* https://commons.wikimedia.org/w/index.php?curid=19070425. Version: April 2012. – via CC BY-SA 3.0

[2] Alfaomega: *Prinzip Ultrallschall*. https://commons.wikimedia.org/w/index.php?curid=32123333. Version: September 2008. – via CC BY-SA 3.0

Kugelsortiererroboter

Radwan El-Maalem, ETIT
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Der LEGO-Kugelsortiererroboter verwendet LEGO-Bauteile und wird von Sensoren und Motoren gesteuert, um die kleinen Kugeln nach ihrer Farbe zu sortieren und sie in die entsprechenden Behälter zu legen. Die Entwicklungsphasen umfassen den Aufbau der mechanischen Struktur, die Integration von Sensoren zur Farberkennung und die Programmierung eines Steuerungsalgorithmus für den Sortiervorgang. Bei der Entwicklung des Kugelsortiererroboters sind einige Design- und Programmierprobleme aufgetreten, die in diesem Artikel näher beschrieben werden.**

***Schlagwörter*—Farbsensor, kugelsortiererroboter, LEGO-Mindstorms, Projektseminar, Roboter**

I. Einleitung

DIE Roboter können heutzutage verschiedene und mehrere Aufgaben gleichzeitig ausführen und sich autonom bewegen. Mit zunehmender Industrialisierung können Roboter die Arbeit von Menschen ersetzen und werden immer wichtiger und unverzichtbarer. Sie können nicht nur das Leben vereinfachen sondern auch die Produktivität in verschiedenen Branchen steigern.
Sortierroboter werden in vielen Branchen und Anwendungsbereichen eingesetzt. Sie sortieren Objekte nach bestimmten Kriterien.
Ein effizientes Beispiel für den Einsatz von Sortierrobotern in der Recyclingindustrie, wo sie produktiver als Menschen arbeiten können, ist in Abbildung 1 dargestellt. Sie können verschiedene Materialien wie Papier, Plastik, Glas und Metalle voneinander trennen, um effizient recyceln zu können. Dabei muss zum Beispiel nicht nur nach Materialart, sondern oft auch nach Farbe oder Form unterschieden werden.

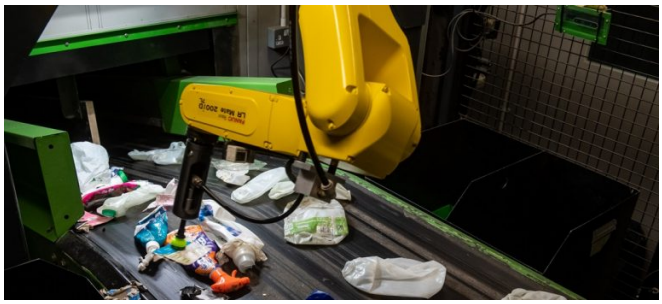


Abbildung 1. Abfallsortierroboter [1]

II. Vorbetrachtungen

Der Kugelsortiererroboter ist im Wesentlichen aufgebaut aus einem Farbsensor und Motoren, die für die Sortierung erforderlich sind.

DOI: 10.24352/UB.OVGU-2024-021

A. kugelsortiererroboter Prinzip

Das Ziel oder die Idee besteht darin, einen Roboter zu entwickeln, der Objekte beispielsweise Kugeln basierend auf ihrer Farbe sortiert. Im ersten Schritt identifiziert ein Farbsensor die Farbe jeder Kugel danach wird die Kugel in einen entsprechenden Behälter transportiert, der zu ihrer Farbe passt. Dieser Prozess setzt sich fort, bis entweder alle Kugeln sortiert sind oder der Benutzer entscheidet, das Programm zu beenden. Dieser Roboter basiert im Prinzip auf einem einfachen Prinzip der Farberkennung.

B. Farbsensor

Bei diesem Teil des Roboters handelt es sich um einen wichtigen Teil der Projektidee. Seine Funktionalität besteht darin, vier Farben zu erkennen: Blau, Rot, Gelb und Grün. Sobald der Roboter die Farbe von Kugeln erkennt, wird die Kugel in entsprechende Behälter transportiert.

Um Farben zu erkennen, verwendet der RGB-Sensor ein spezielles Verfahren. Mit den Farben Rot, Blau und Grün beleuchtet der Sensor zunächst das zu erkennende Objekt. Anschließend erkennt er die Farben, die von dem Objekt reflektiert werden. Wird zum Beispiel mehr grünes Licht reflektiert, so erkennt der Sensor, dass es sich um ein grünes Objekt handelt. Das gleiche Prinzip wird auch für die Farben Rot und Blau angewendet. Zusätzlich ist der Farbsensor in der Lage, Gelb bei gleichmäßiger Reflexion von Rot und Grün, aber ohne Blau zu erkennen.

C. Einordnung der Kugeln

Die Kugeln befinden sich auf einer festen Rampe und sind zufällig angeordnet. Durch die Schwerkraft rollen sie automatisch nach unten bis zum Farbsensor, wo ihre Farbe erkannt wird. Anschließend werden die Kugeln von Motor A weiter nach unten geschoben und dann mithilfe vom Motor B in die entsprechenden Behälter abfallen.

III. Konstruktion und Ablauf des Kugelsortierer-Roboter

Im Folgenden soll die Realisierung des kugelsortiererroboters von der Konstruktion über die Funktionsweise bis zur Programmierung erläutert werden.

A. Aufbau

Für den Bau des Roboters stehen zwei große LEGO-Kästen zur Verfügung. Diese enthalten mehrere LEGO Mindstorms Sets, einschließlich eines NXT-Steins, drei Motoren und

verschiedene Sensoren.

Der Roboter soll so konstruiert, dass er eine hohe Stabilität aufweist, obwohl er aus leichten LEGO-Bausteinen besteht (siehe Abbildung 4).

Der Roboter besteht aus zwei Teilen: einem festen und einem beweglichen Teil. Der bewegliche Teil, wie die untere Rampe oder Sortierrampe, ist im Design integriert, um die Kugeln nach der Farberkennung zu den jeweiligen Behältern zu transportieren. Sie ist so konzipiert, dass sie durch einen Motor B vollständige Drehungen um $360°$ ausführen kann, um die Kugeln präzise zu verteilen.

Die Antriebskraft für die Bewegungen des Roboters wird von LEGO-Motoren geliefert. Diese Motoren sind präzise mit Matlab gesteuert.

B. Umsetzung

Zur Realisierung des kugelsortiererroboter werden in diesem Fall zwei Motoren, ein Farbsensor, das NXT-Gerät und verschiedene LEGO-Bausteine benötigt.
Der Roboter ist modular aus LEGO-Bausteine aufgebaut, was den Aufbau und die Fehlersuche vereinfacht und jedes Segment des Roboters ist so konzipiert, dass es spezifische Aufgaben erfüllt.

Der Arbeitsablauf gliedert sich wie folgt: Motor A schiebt die Kugeln weiter nach unten, nachdem der Farbsensor ihre Farbe erkannt hat. Motor B dreht die Sortierrampe bis zu $360°$, um die Kugeln in die entsprechenden Behälter zu transportieren.

Die Steuerung des Roboters erfolgt über MATLAB. Um die Motoren zu verwenden, müssen sie zunächst korrekt konfiguriert werden. Auf dem NXT-Gerät sind die Anschlüsse mit Buchstaben gekennzeichnet, die in MATLAB verwendet werden, um die Motoren richtig zu definieren. In diesem Fall sind Motor A und der Motor B festgelegt. Die Motoren müssen so eingestellt werden, dass die gewünschte Kraft und Drehung konfiguriert sind. Hierfür stehen die Befehle `MotorX.Power` und `MotorX.TachoLimit` zur Verfügung.

C. Funktionsweise

Im ersten Schritt beginnt der Prozess damit, dass die Kugeln auf eine feste Rampe abrollen, bis sie den Farbsensor erreichen. Dieser ist darauf programmiert, die Farben der Kugeln präzise zu identifizieren und entsprechende Signale zu generieren.

Nach der Farberkennung gibt der Farbsensor die Information an Motor B weiter, der für die Ausrichtung der Sortierrampe verantwortlich ist. Motor B dreht die Sortierrampe in die Position, die dem erkannten Farbwert der Kugel entspricht, sodass jede Kugel in der richtigen Position in die entsprechende Sammelstation gelangt.

Sobald die Rampe ausgerichtet ist, aktiviert sich Motor A, um die Kugel in den jeweiligen Behälter zu stoßen, danach kehrt Motor A in seine Startposition zurück, bereit für die nächste Kugel. Nachdem die Kugel erfolgreich sortiert wurde, fährt auch Motor B die Sortierrampe wieder in ihre Anfangsposition zurück, die als Position 0 definiert ist.

Bei Erkennung einer grünen Kugel positioniert Motor B die Rampe um $90°$, sodass die Kugel in den grünen Behälter geleitet wird. Im Falle einer roten Kugel erfolgt eine Drehung um $-280°$, um die Kugel in den roten Behälter zu befördern. Erkennt der Sensor eine blaue Kugel, wird die Rampe um $-90°$ justiert, damit die Kugel ihren Weg in den blauen Behälter findet. Wenn der Farbsensor Gelb ausgibt, dreht Motor B die Sortierrampe um $280°$. Nach dieser Bewegung schiebt Motor A die blaue Kugel, die dann direkt in den blauen Behälter fällt.

D. Programablaufplan

Ein Beispiel für einen Programmablaufplan zur Erläuterung des ist in Abbildung 2 dargestellt

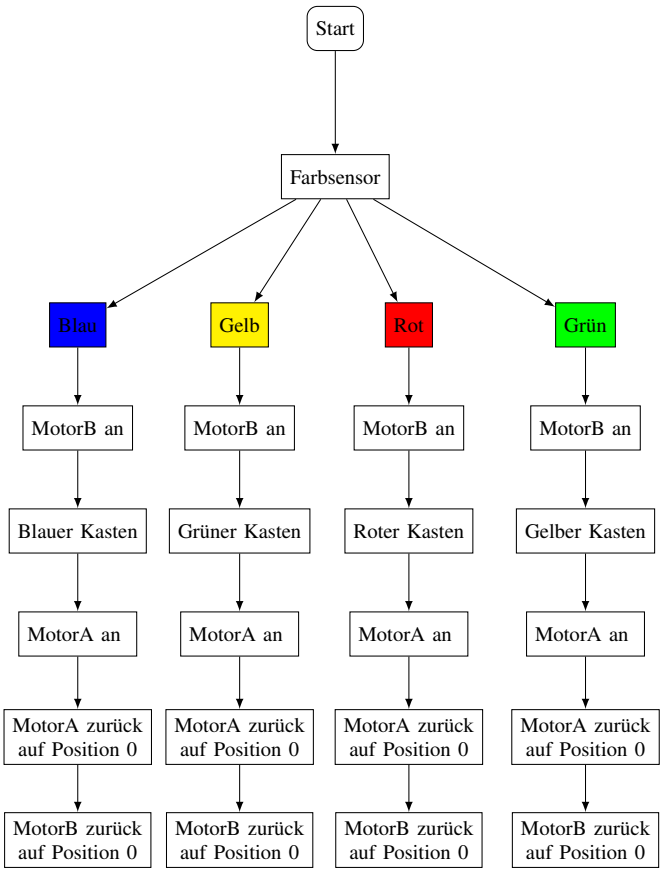


Abbildung 2. Programmablaufplan des Farbsortierroboters

IV. Ergebnisdiskussion

Nach Abschluss von zwei Wochen intensiver Arbeit im Rahmen des Projektseminars konnte ein erfolgreicher Kugelsortiererroboter entwickelt werden, der Kugeln basierend auf ihrer Farbe sortiert, wie in Abbildung 3 zu sehen ist. Um die Leistungsfähigkeit des Roboters zu verbessern. Es wurden verschiedene Tests durchgeführt, darunter auch die Messung der Transportdauer einer Kugel vom Startpunkt bis zu ihrem entsprechenden Behälter. Die Ergebnisse zeigten einen Durchschnittswert von 4 Sekunden pro Kugel.

Es wurde auch festgestellt, dass die Sortiergeschwindigkeit weiter gesteigert werden könnte, durch eine Erhöhung der

Power der Motoren A und B. Bei einem weiteren Test wurde die Genauigkeit der Sortierung überprüft. Hierbei wurden jeweils 5 Kugeln jeder Farbe verwendet, die vom Roboter sortiert wurden. Die Auswertung ergab, dass die Zuordnung von grünen, roten und blauen Kugeln zu $100\,\%$ korrekt erfolgte, während die Fehlerquote bei gelben Kugeln bei $98\,\%$ lag.

V. Zusammenfassung und Fazit

Der Kugelsortiererroboter erfüllte seine Funktion gut. Mit mehr als zwei Wochen Zeit hätte man eine grafische Benutzeroberfläche (GUI) für den Roboter entwickeln können, die nicht nur die Anzahl der Kugeln pro Behälter anzeigt, sondern auch die Implementierung neuer Funktionen ermöglicht hätte.

Literaturverzeichnis

[1] Recycleye bringt KI-gesteuerte Abfallsortierroboter nach Deutschland, 18.05.2022, https://recyclingportal.eu/Archive/73020, Version = 1.03.2024

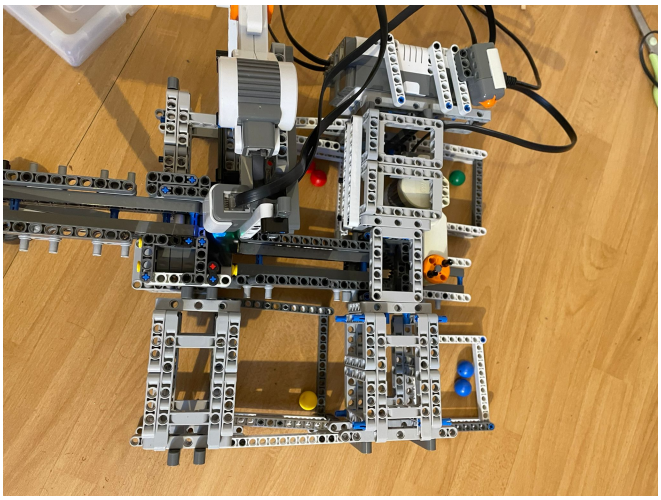


Abbildung 3. Kugelsortiererroboter, Ansicht von oben

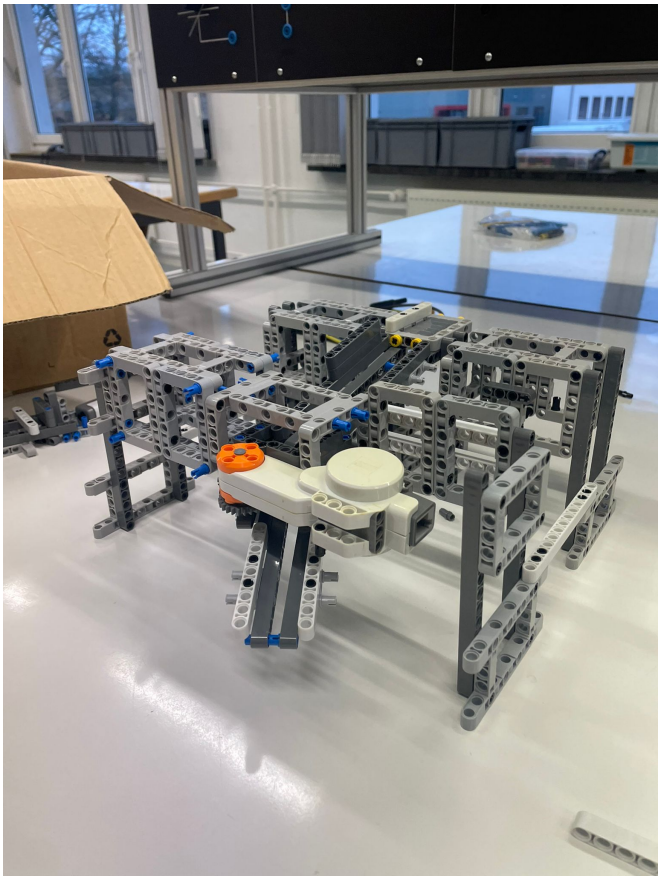


Abbildung 4. Kugelsortiererroboter, Ansicht von vorne

LABrat – Ein Roboter, der den Weg kennt

Max Bachran, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Das diesjährige LEGO-Praktikum 2024 widmete sich der Aufgabe aus LEGO und Matlab-Programmierung einen funktionsfähigen Roboter zu bauen, der gezielte Problemstellungen überwinden kann. Dabei wurde in diesem Paper der autonom fahrende Labyrinthroboter und seine Entwicklung betrachtet. Mit Omni-Wheels, die das Fahren in alle Richtungen ermöglichen und Ultraschallsensoren zur Distanzmessung wurde ein intelligentes Fahrzeug gebaut, das sich durch ein Labyrinth bewegt und den Ausgang findet.**

***Schlagwörter*—Self-driving, Hinderniserkennung, Labyrinth, Omni-Wheels, Wegfindung**

I. Einleitung

DIE heutige Welt wird zunehmend stärker vernetzt und digitalisiert. Dies ist in vielen Bereichen des Lebens deutlich spürbar, auch in der Automobilbranche. Dort wird seit langem an selbstfahrenden Fahrzeugen geforscht und erste Schritte, wie zum Beispiel Parkassistent oder automatische Notbremssysteme, sind schon längst für den Verbraucher erhältlich. Der Mensch wird in diesen Bereichen zunehmend von der Technik unterstützt und entlastet. Auch die vollständige Automatisierung des Fahrzeugs scheint nicht allzu weit in der Zukunft zu liegen. Die Entwicklung dieser Systeme ist im Fokus vieler und es wird aus vielen Richtungen geforscht. Aufgrund der potenziell verheerenden Konsequenzen von Fehlern für Verkehrsteilnehmer sind die sicherheitstechnischen Bedenken für viele Menschen in Bezug auf selbstfahrende Autos noch zu groß und Vertrauen zu gering. Sollten jedoch autonom fahrende Fahrzeuge zuverlässiger sein als menschliche Fahrer, könnten sie in Zukunft auf den Markt kommen und den Straßenverkehr sicherer und effizienter gestalten.
Auf Basis dieser Überlegungen entstand die Idee, bereits heute ein Fahrzeug zu entwickeln, das selbständig fahren kann und auf äußere Umstände reagiert, damit schon heute ein Blick ins Morgen geworfen werden kann.

II. Vorbetrachtungen

A. Welche Funktionen muss das Fahrzeug beherrschen?

Das Fahrzeug muss grundlegend in der Lage sein, sich fortzubewegen. Es sollte daher zu Vorwärts-, Rückwärts- und Seitwärtsbewegung in der Lage sein. Da das Labyrinth mit rechten Winkeln konstruiert wurde, ist diese Fahrimplementation theoretisch ausreichend. Nur für kleineres Ausrichten muss sich das Fahrzeug drehen können.
Darüber hinaus muss es in der Lage sein, auf Hindernisse zu reagieren und daher die Ultraschallsensoren ansprechen zu können. Nun muss auf die Rückgabe der Sensoren auch reagiert werden. Eine Logik muss implementiert werden, die auf die Hindernisse reagiert und es dem Fahrzeug ermöglicht, den richtigen Weg zu finden.

DOI: 10.24352/UB.OVGU-2024-022

B. Welche Komponenten werden dafür benötigt?

Für die Umsetzung des Projekts wurde ein LEGO-Bausatz bereitgestellt, der alle benötigten Komponenten für die Konstruktion enthielt. Dabei spielten einige Bauelemente eine zentrale Rolle. Der Kern der Konstruktion war der Steuercomputer, der für die Ansteuerung der weiteren Komponenten zuständig war. Zu Beginn wurde ein NXT-Baustein verwendet, der später durch einen EV3-Baustein ersetzt wurde. An diesen Baustein waren insgesamt vier Motoren und vier Ultraschallsensoren angeschlossen.
Die Ultraschallsensoren dienten der Hinderniserkennung, indem sie kontinuierlich Ultraschallsignale aussenden und deren Reflexionen an der Oberfläche von Hindernissen die Distanz zum nächsten gerade vor ihnen liegenden Hindernis ermittelten. Vier Motoren wurden als Antrieb für die vier verwendeten Räder eingesetzt. Dadurch war es möglich, alle vier Räder gleichzeitig oder separat anzusteuern. Bei den Rädern handelte es sich um spezielle Omni-Wheels [1], bei denen die Lauffläche teilweise mit Rollen ausgestattet war, deren Drehachsen im rechten Winkel zur Drehachse des jeweiligen Hauptrades lagen. Diese ermöglichten dem Fahrzeug das seitwärts Fahren, ohne dass eine Lenkbewegung erforderlich war. Alle Teile wurden mithilfe von LEGO-Steinen miteinander verbunden, um so eine stabile Konstruktion als Ganzes zu gewährleisten.

III. Umsetzung

A. Konstruktion 1

Die anfängliche Idee, das Fahrzeug in die Höhe zu bauen, um die Breite zu minimieren und somit das Labyrinth nicht zu vergrößern, erwies sich als Herausforderung. Der NXT-Baustein als Steuercomputer wurde zentral platziert, mit je einem Motor an der Längs- und Querseite, wobei die Drehachsen unterhalb des Bausteins positioniert waren. Jeder Motor sollte zwei Omni-Wheels über eine verlängerte Achse antreiben.
Allerdings stieß diese Konfiguration schnell auf Probleme. Durch die Verwendung von nur zwei Motoren kam es zu einer ungleichmäßigen Lastverteilung, was die Stabilität des Fahrzeugs beeinträchtigte. Zudem führte die Anordnung der Räder auf gleicher Höhe dazu, dass die Radachsen miteinander kollidierten. Angesichts dieser Schwierigkeiten wurde dieser Ansatz rasch verworfen, und es war klar, dass eine neue Lösung gefunden werden musste, um das Fahrzeug stabiler und funktionaler zu gestalten.

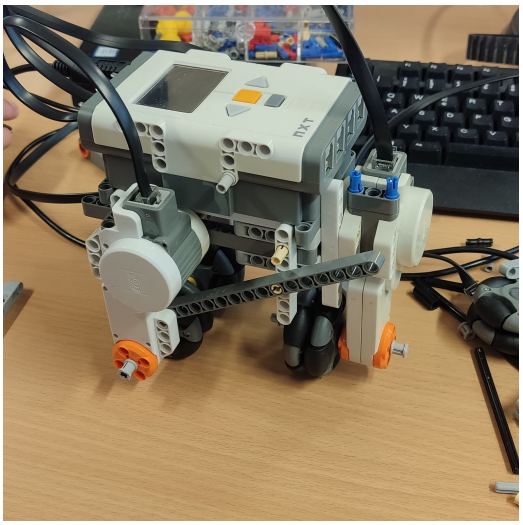


Abbildung 1. Konstruktion 2 – LABrat mit drei Motoren & 3 Rädern

B. Konstruktion 2

Aufgrund der Erkenntnisse aus der ersten Konstruktion wurden bei der zweiten Verbesserungen vorgenommen. Die Motoren wurden weiterhin senkrecht nach unten aufgestellt, wobei am NXT-Baustein zwei Motoren an der Längsseite und ein Motor am Heck an der Querseite angebracht wurden. Wie in Abbildung 1 zu sehen ist, betrieb jeder Motor ein Omni-Wheel. Diese Anordnung führte zu einer verbesserten Lastverteilung durch den dritten Motor und das dritte Rad.
Trotz dieser Verbesserungen traten beim Vorwärts- und Rückwärtsfahren horizontale Kräfte auf, die durch die Trägheit der seitlichen Motoren verursacht wurden. Diese verschoben sich während der Fahrt in die entgegengesetzte Richtung. Besonders beim seitlichen Fahren wurde dieses Phänomen beim hinteren Motor deutlich. Während es möglich war, die Motoren an der Längsseite zu stabilisieren, gestaltete sich dies beim hinteren Motor problematisch. Die auftretenden Kräfte konnten nicht vollständig unterbunden werden, obwohl rudimentäre Fahrtests mit Vorwärts- und Rückwärtsbewegungen erfolgreich durchgeführt wurden.
Weiterhin war die fehlende gemeinsame Drehachse zwischen den drei Motoren-Rad-Kombinationen ein Problem. Aufgrund der Art der LEGO-Steine war es äußerst schwierig, alle Motoren symmetrisch zueinander auszurichten. Dies führte zu unerwünschten Lenkbewegungen beim seitlichen Fahren und machte eine weitere Entwicklung dieser Konfiguration unpraktikabel.

C. Konstruktion 3

Die dritte Konstruktion war auch das endgültige Fahrzeug. Hierbei wurde nun der NXT-Baustein durch einen EV3-Baustein ersetzt. Bei diesem konnten vorteilhafter Weise jetzt vier Motoren und dadurch vier Räder angesteuert werden. An einer Stützkonstruktion wurden sowohl der EV3-Steuercomputer als auch die Motoren befestigt, sodass sich die Drehachsen unterhalb des EV3-Bausteins befanden. Jeder Motor wurde mit einem passenden Omni-Wheel ausgestattet. Das Stützkonstrukt

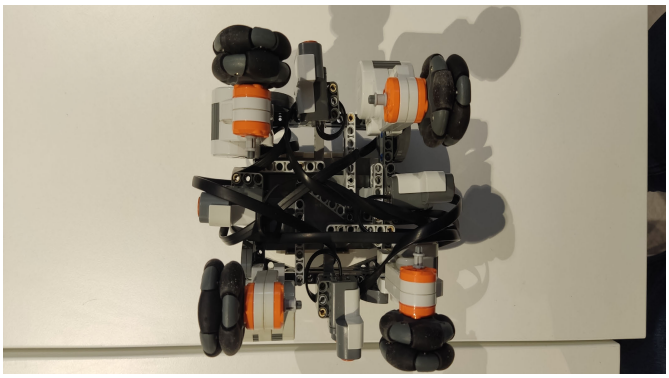


Abbildung 2. Unterseite des Fahrzeuge mit Kennzeichnung der Drehachsen

fungierte als eine Art Chassis und stabilisierte das Gesamtkonstrukt in gewissem Maße, während es auch verschiedene Anknüpfungspunkte für weitere Komponenten bot.
Die Motoren waren versetzt zueinander orientiert, wobei jeweils zwei Räder parallel zur Längsseite und zwei Räder parallel zur Querseite des EV3-Steuercomputers ausgerichtet waren, wie in Abbildung 2 zusehen ist. Durch die Symmetrie der Motoren zueinander entstand eine gemeinsame Drehachse, was ungewollte Lenkbewegungen verhinderte. Das Fahren in allen relevanten Richtungen wurde dadurch deutlich verbessert im Vergleich zu den vorherigen Konstruktionen.

Die Verwendung des EV3-Bausteins erforderte eine Anpassung des bereits geschriebenen Codes, da die Programmierung hier objektorientiert erfolgte. Der Code konnte aber problemlos migriert werden und erste Fahrtests erfolgten ohne Probleme. Anschließend wurden die vier Ultraschallsensoren angebracht, je zwei an der Längs- und Querseite des Fahrzeugs in der Mitte. Auch hier verliefen die ersten Funktionstests erfolgreich.
Mit der Fertigstellung des in Abbildung 3 zusehenden Prototypen verlagerte sich der Fokus auf das Verfeinern des MATLAB-Codes. Während der Entwicklung der finalen Konstruktion wurden kontinuierlich Funktionstests durchgeführt und der Code aus den vorherigen Revisionen an das EV3-konforme Format angepasst.

D. Funktionsweise

In der Abbildung 4 wird der Programmablauf des Fahrzeugs vereinfacht dargestellt. Das Fahrzeug befindet sich zum Start bereits im Labyrinth und muss nun herausfinden.
Dabei hält sich das Fahrzeug stets an der rechten Wand, um garantiert den Ausgang zu finden. Da das Fahrzeug mit vier Ultraschallsensoren zur Distanzmessung ausgestattet ist, kann es die Wände in alle Richtungen kontinuierlich erfassen.
Zuerst wird das Programm initialisiert. Dabei muss beachtet werden, dass verschiedene Werte korrekt eingestellt werden, damit z.B. das Fahrzeug rechtzeitig auf Hindernisse reagiert. Zu groß und das Fahrzeug bleibt zu weit von Wänden entfernt und kann möglicherweise einen potenziellen Gang nicht erkennen. Zu klein und das Fahrzeug bremst nicht früh genug.
Dann folgt die Hauptschleife des Programms. In dieser werden die Sensoren abgefragt. Wenn ein kritischer Schwellwert unterschritten wird, weiß das Fahrzeug, dass es kurz davor

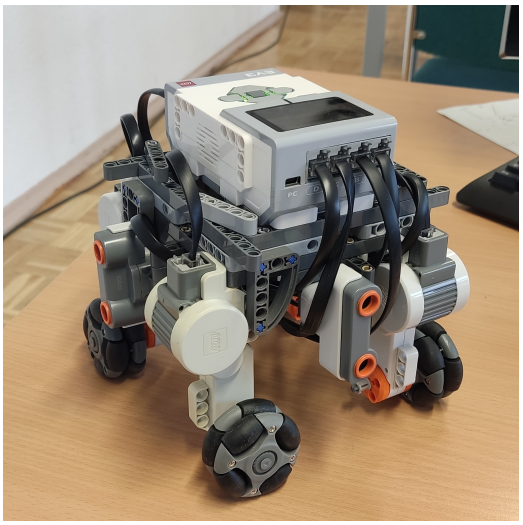


Abbildung 3. Konstruktion 3 – Finaler Prototyp der LABrat

ist in ein Hindernis zu fahren und stoppt alle Motoren. Danach überprüft es die Parallelität zur Wand durch eine Differenzberechnung zwischen den letzten Werten und den Aktuellen der seitlichen Sensorik. Bei einer Differenz muss nun das Fahrzeug die Motoren stoppen und sich ein kleines Stück drehen.
Als nächster Schritt wird die optimale Fahrtrichtung neu berechnet, um auf neu auftretende Abzweigungen reagieren zu können. Aufgrund der Prämisse sich rechts zu halten, haben effektiv nur neu auftretende Gänge rechts vom Fahrzeug einen Einfluss auf das Fahrverhalten. In diesem Fall muss das Fahrzeug vor dem Abbiegen noch etwas weiter fahren, da die Sensorik mittig vom Fahrzeug angebracht ist und bei einem sofortigen Richtungswechsel der hintere Teil des Fahrzeugs in die Wand fahren würde. Daher wird ein Moment gewartet, bevor die Motoren auch hier gestoppt werden. Es muss hier aber auch beachtet werden, dass nach diesem Richtungswechsel das Fahrzeug noch in der Kreuzung steht und soeben in Fahrtrichtung rechts der Ausgangsrichtung entspricht. Daher ist es nötig einen erneuten Richtungswechsel zu unterbinden und erst wieder zu erlauben, nachdem die Kreuzung verlassen wurde und sich das Fahrzeug im neuen Gang befindet.
Potenziell muss das Fahrzeug inzwischen die Motoren neu konfigurieren, da sich die Richtung des Fahrzeugs ändern soll oder wieder starten, weil sie gestoppt wurden. Die Motoren werden, falls sie noch laufen, gestoppt, rekonfiguriert und dann neu gestartet.
Die Motoren sind so konfiguriert, dass sie sich bis zu einem Stoppsignal drehen, um unnötige Start- und Stoppsequenzen zu vermeiden, da bei diesen sich das Fahrzeug drehen kann. Um zusätzliche Drehungen zu vermeiden, werden die Motoren aus diesem Grund auch synchron gestartet.
Da in jedem Durchlauf auch die Richtung neu berechnet wurde, fährt das Fahrzeug immer in die optimale Richtung. Solange der Ausgang nicht erreicht wurde, läuft das Programm in dieser Schleife ab. Nach Erreichen des Ausgangs kann das Fahrzeug nun ruhen.

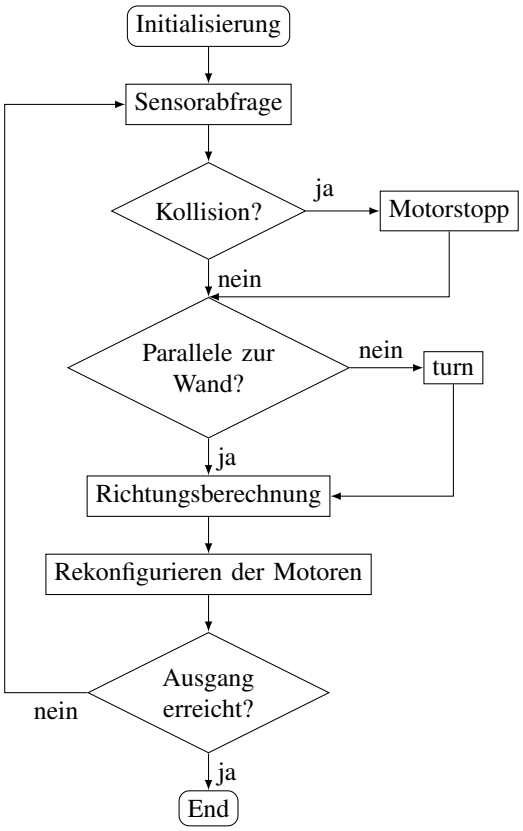


Abbildung 4. Beispielhafter Programmablaufplan zur Erklärung eines Verfolgungsalgorithmus

IV. Ergebnisdiskussion

Es hat sich gezeigt, dass die Idee des Labyrinthroboters prinzipiell funktioniert. Bewegung und Algorithmik werden im Kern ausgeführt, wie angedacht. Allerdings gibt es noch ungelöste Probleme in der Skalierung. Auf kleineren Teststrecken zeigt das Fahrzeug ein zuverlässiges Fahrverhalten. Aber je größer das Labyrinth ist, desto häufiger treten Fehler auf. Drei Probleme konnten dabei charakterisiert werden.
Das Erste ist die Umgebung. Der Boden und seine Beschaffenheit beeinflussen, wenn auch nur geringfügig, das Fahrverhalten des Fahrzeugs.
Zudem ist die aktuelle Konstruktion, genauer gesagt die LEGO-Bausteine, problematisch. Es konnte festgestellt werden, dass horizontale Kräfte auf das Fahrzeug wirken, wenn dieses sich vorwärts, rückwärts oder seitwärts bewegt. Diese Kraft hat Auswirkungen auf die Motoren und ungewollte Bewegungen, welche die Richtung des Fahrzeugs beeinflussten. Ein Drift nach links oder rechts der Fahrtrichtungen war zu beobachten. Auch wenn versucht wurde diesen auszugleichen, hat dies nur mittelmäßig funktioniert.
Des Weiteren sind die Ultraschallsensoren nur in der Lage gerade vor ihnen zu messen. Das Fahrzeug kann nur erkennen, ob Hindernisse direkt vor den Sensoren sind. Daher entstehen sehr große tote Winkel, in denen das Fahrzeug blind ist. Daher kommt es in Extremfällen zu Kollisionen. Durch präzise Parameterbestimmung kann dieses Problem minimiert werden, allerdings sind diese nur für eine Teststrecke optimiert und

führen auf neuen Strecken erneut zu Problemen.

V. Zusammenfassung und Fazit

Zusammenfassend lässt sich sagen, dass kreative Projekte mit LEGO und MATLAB verwirklicht werden können. Die Implementation von verschiedenen Funktionen zum Fahren, zur Hinderniserkennung und Algorithmik lässt sich schnell lernen und umsetzen. Selbst ein selbständig fahrendes Fahrzeug kann entwickelt werden.
Festzuhalten ist, dass jedes Projekt stets mit einem realisierbaren Konzept angegangen werden sollte, im kleinen angefangen werden muss und dann nach erfolgreichen Tests skaliert werden kann.

In Bezug auf den Labyrinthroboter kann gesagt werden, dass die Konstruktion noch Schwachstellen beinhaltet, die eine höhere Stabilität und Berechenbarkeit zurückhalten. Auch ist der Code sehr Parameter und Labyrinth abhängig, weshalb entweder mehr Sensoren oder eine andere Herangehensweise in Betracht gezogen werden sollten.

Literaturverzeichnis

[1] https://de.wikipedia.org/wiki/Allseitenrad

LABrat – Ein Roboter, der weiß, wo es langgeht

Nils-Aaron Hildebrandt, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Dieses Paper beschäftigt sich mit der Entwicklung eines Fahrzeugs, das durch motorbetriebene Omni-Wheels in alle Richtungen fahren konnte. Durch den Einsatz von Ultraschallsensoren und Anwendung eines geeigneten Algorithmus wurde das eigenständige Finden des Ausgangs eines Labyrinths gewährleistet. Zur Realisierung dieses Projekts stand ein LEGO-Bausatz und MATLAB als Programmiersprache zur Verfügung.**

***Schlagwörter*—Automatisierung, Hinderniserkennung, Labyrinth, Omni-Wheels, Wegfindung**

I. Einleitung

In unserer heutigen globalen Welt werden sämtliche Lebensbereiche immer stärker vernetzt, technisiert, digitalisiert oder sogar automatisiert. Auch im Bereich der Automobilindustrie ist dieser Trend längst angekommen und es wird aktiv geforscht und entwickelt, um Fahrzeuge intelligenter zu gestalten. Seien es klimatechnische Faktoren, sicherheitsrelevante Aspekte oder schlichtweg der Komfort des Fahrzeugführers — in all diesen Bereichen soll der Mensch durch aktive Entscheidungen von verschiedensten Systemen unterstützt werden. Vor allem im Bereich des klassischen Pkw ist auffällig, wie die Themen Assistenzsysteme und autonomes Fahren in den letzten Jahren bis heute an Popularität dazugewonnen haben. Solche intelligenten Fahrzeuge bevölkern zunehmend unsere Straßen und die Grenzen menschlicher und maschineller Intelligenz verschwimmen untereinander.
Die Entwicklung von Systemen, die eigenständig aktive Entscheidungen treffen, steht im Mittelpunkt der Interessen. Es muss sichergestellt werden, dass unter allen Umständen, seien sie noch so minimal, das Fahrzeug stets richtig handelt und der allgemeine Straßenverkehr nicht behindert oder sogar gefährdet wird. Das Leib und Wohl von Mensch, Tier und Natur muss gewahrt werden. Demnach ist akribisches, fehlerfreies Kreieren und Testen solcher Systeme von allerhöchster Relevanz. Werden solche Systeme als zuverlässig eingestuft, so kann der Straßenverkehr sicherer, effizienter und sogar nachhaltiger gestaltet werden, wovon letztendlich alle profitieren. Aus der Tatsache heraus, dass intelligente und autonome Fahrzeugsysteme die Zukunft sind, entstand die Idee ein Fahrzeug zu erschaffen, das selbstständig Hindernisse erkennt und eigenständig entscheidet, wie diese Hindernisse umgangen werden können.

II. Vorbetrachtungen

Hinderniserkennung ist bereits in zahlreichen Automodellen verschiedenster Hersteller vorhanden. Dabei wird nicht eine einzige Methode verwendet. Vielmehr ist es ein Zusammenspiel aus verschiedensten Assistenzsystemen. So wird die Ultraschallsensorik vor allem für die Hinderniserkennung im Nahbereich verwendet. Hierbei ist das prominenteste Beispiel das Einpark-Assistenzsystem mit Ultraschallsensoren an der Front und am Heck des Fahrzeugs. Beispiele können bei BMW [1] oder Audi [2] gefunden werden.

A. Welche Funktionen musste das Fahrzeug beherrschen?

Grundsätzlich musste das Fahrzeug in der Lage sein, sich fortzubewegen. Dabei lag das Augenmerk auf eine Vorwärts-, Rückwärts- und Seitwärtsbewegung, da das Labyrinth so konzipiert wurde, dass sich alle Gänge im rechten Winkel zueinander befanden. Doch eine reine Bewegungsimplementation war nicht ausreichend. Es war wichtig, dass das Fahrzeug auch bei bestimmten Bedingungen überhaupt anhält und dass dies auch rechtzeitig geschah. Um dies zu gewährleisten, war es erforderlich, dass Hindernisse erfolgreich als solche registriert wurden. Nach erfolgreicher Erkennung der Hindernisse musste eine Logik vorherrschend sein, die entschied, wie und wohin das Fahrzeug als Nächstes fahren sollte.

B. Welche Komponenten wurden dafür benötigt?

Für die Realisierung des Projekts stand ein LEGO-Bausatz zur Verfügung, der sämtliche Komponenten der Konstruktion beinhaltete. Dabei waren einige Bauelemente von besonderer Bedeutung. Das Herzstück der Konstruktion ist der Steuercomputer, der das Ansteuern weiterer Komponenten ermöglichte. Zu Beginn wurde ein NXT-Baustein verwendet, der im weiteren Verlauf durch einen EV3-Baustein ersetzt wurde. An diesen Baustein waren insgesamt vier Motoren und vier Ultraschallsensoren angeschlossen. Die Sensorik diente der Hinderniserkennung, bei der permanent ein Ultraschallsignal gesendet und dessen Reflexion an der Oberfläche eines Hindernisses detektiert wurde.
Die vier Motoren dienten als Antrieb für die vier verwendeten Räder. Dadurch wurde gewährleistet, dass alle vier Räder gleichermaßen von der Software angesprochen werden konnten. Bei den Rädern handelte es sich um eine besondere Version. Hier wurden sogenannte Omni-Wheels verwendet. Dies waren Räder, bei denen die Lauffläche teilweise mit Rollen ausgestattet war, deren Drehachsen im rechten Winkel zur Drehachse des jeweiligen Hauptrades liegen. Dadurch konnte seitlich gefahren werden, ohne eine Lenkbewegung auszuführen. Sämtliche Bauteile wurden durch vielfache LEGO-Bausteine miteinander verbunden, sodass ein stabiles Gesamtkonstrukt entstand. Neben der Hardware bedarf es auch an Software, was durch die Verwendung eines selbstgeschriebenen MATLAB-Programmes sichergestellt wurde. In diesem Programm befand sich sämtliche Logik, die zur Ansteuerung der Baukomponenten benötigt wurde.

DOI: 10.24352/UB.OVGU-2024-023

III. Entwicklungsprozess

Der Entwicklungsprozess ließ sich in zwei große Bereiche einteilen: Das Bauen eines geeigneten Konstrukts und das Etablieren eines funktionalen MATLAB-Codes, um sämtliche Funktionen zu erfüllen und zu testen. Zum Thema der Konstruktion lassen sich drei Phasen einteilen. Innerhalb jeder Phase existierte ein mehr oder weniger brauchbarer Prototyp.

A. Konstruktion 1

Um ein Fahrzeug zu bauen, das eigenständig den Ausgang eines Labyrinths findet, war eine stabile Konstruktion der Ausgangspunkt des Erfolgs. Die Grundidee bestand darin, das Fahrzeug eher in die Höhe statt in die Breite zu bauen, da sonst das Labyrinth mit zunehmender Breite des Fahrzeuges größer dimensioniert werden musste. Das Kernelement war der NXT-Baustein als Steuercomputer. An der Längs- und an der Querseite des Bausteins wurde jeweils ein Motor angebracht. Die Motoren waren so ausgerichtet, dass deren Drehachse sich unterhalb des NXT-Bausteines befanden. Jeder Motor sollte jeweils 2 Omni-Wheels über je eine verlängerte Achse antreiben. Diese Vorstellung stellte sich recht schnell als eine unpraktische Lösung heraus. Aufgrund der Verwendung von nur zwei Motoren wurde eine ungleichmäßige Lastverteilung und damit Stabilitätseinbußen vorprogrammiert. Ebenfalls kollidierten die Radachsen miteinander, da sich die Räder aller auf gleicher Höhe befanden sollten. Dahingehend wurde dieser Konstruktionsversuch schnell verworfen.

B. Konstruktion 2

Nach den Erkenntnissen der ersten Konstruktion konnten Verbesserungen an der zweiten vorgenommen werden. Die Motoren wurden weiterhin nach unten ausgerichtet. Am NXT-Baustein wurden zwei Motoren an der Längsseite und ein Motor an der Querseite am Heck angebracht. Jeder Motor trieb ein Omni-Wheel an (siehe Abb. 1). Aufgrund des dritten Motors und des dritten Rades konnte die Lastverteilung deutlich verbessert werden. Allerdings wurden horizontale Kräfte beim Vorwärts- bzw. Rückwärtsfahren festgestellt. Wegen der Trägheit der seitlichen Motoren verschoben sich diese beim Vorwärts- bzw. Rückwärtsfahren in die entsprechende entgegengesetzte Richtung. Beim hinteren Motor trat das Phänomen beim seitlichen Fahren auf.
Während die an der Längsflanke angebrachten Motoren stabilisiert werden konnten, gestaltete sich dies beim hinteren Motor problematisch. Die auftretenden horizontalen Kräfte konnten nicht vollständig unterbunden werden. Allerdings waren rudimentäre Fahrtests mit dem Testen der Vorwärts- und Rückwärtsbewegung erfolgreich. Ein weiteres Problem führte zur Beendigung der weiteren Entwicklung an der zweiten Revision der LABrat. Zwischen den drei Motoren-Radkombinationen konnte keine gemeinsame Drehachse hergestellt werden. Baubedingt gestaltete es sich äußerst schwer, alle Motoren symmetrisch zueinander auszurichten. Das Resultat waren ungewollte Lenkbewegungen beim seitlichen Fahren.

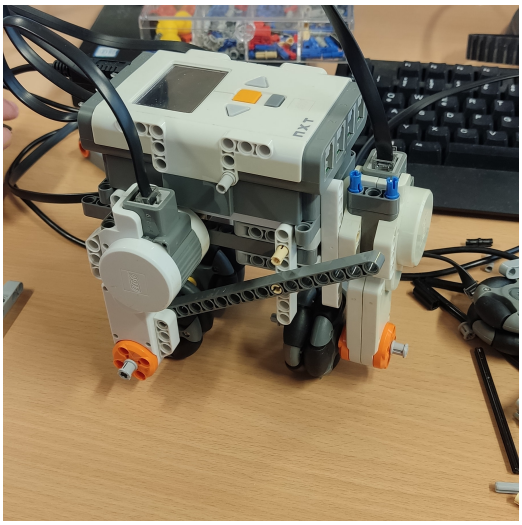


Abbildung 1. Konstruktion 2 – LABrat mit drei Motoren & 3 Rädern

C. Konstruktion 3

Mit der dritten Revision der LABrat wurde der finale Prototyp fertiggestellt. Allerdings musste der NXT-Baustein durch einen EV3-Baustein ersetzt werden. Der Vorteil bestand darin, dass nun vier Motoren und dadurch vier Räder angesteuert werden konnten. Infolgedessen wurde eine Stützkonstruktion entwickelt, auf die der EV3-Steuercomputer sich anbringen ließ. An dieser Stützkonstruktion wurden sämtliche Motoren befestigt und so ausgerichtet, dass deren Drehachsen sich unterhalb des EV3 befanden. Jeder Motor wurde mit einem passenden Omni-Wheel bestückt. Das Stützkonstrukt diente als eine Art Chassis und stabilisierte die Gesamtkonstruktion zu einem gewissen Grad und bot verschiedene Anknüpfungspunkte für weitere Komponenten.
Die Motoren waren versetzt zueinander orientiert. Dies bedeutet, dass zwei Räder parallel zur Längsseite und zwei Räder parallel zur Querseite des EV3-Steuercomputers ausgerichtet waren. Trotz der versetzten Orientierung wurden die Motoren symmetrisch zueinander aufgebaut (siehe Abb. 2). Dadurch entstand eine gemeinsame Drehachse für alle Motoren-Radkombinationen. Ungewollte Lenkbewegungen wurden dadurch verhindert und das Fahren in allen relevanten Richtungen erfolgte deutlich besser als bei den vorherigen Konstruktionen.

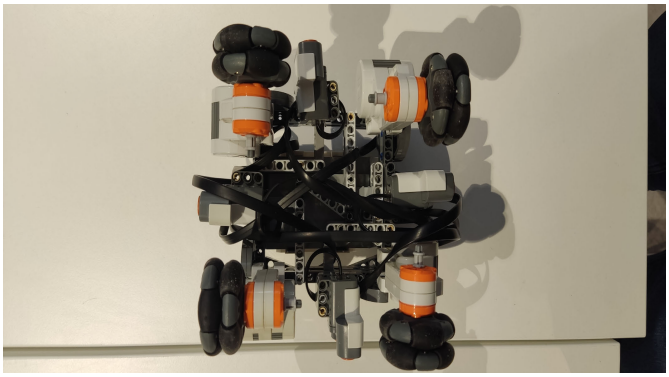


Abbildung 2. Unterseite des Fahrzeuge mit symmetrischer Motorenanordnung

Die Verwendung des EV3-Bausteines verlangte eine Anpassung des bereits geschriebenen Codes, da hier die Programmierung objektorientiert erfolgte. Dies stellte aber keine großen Probleme dar und das Fahren konnte erfolgreich getestet werden. Es folgte das Anbringen der vier Ultraschallsensoren, wobei je zwei an der Längs- und Querseite mittig angebracht wurden. Auch hier waren erste Funktionstests erfolgreich. Durch die Fertigstellung des Prototyps (siehe Abb. 3) verschob sich die Konzentration auf das Verfeinern des MATLAB-Codes. Bereits während der Entwicklung der finalen Konstruktion wurden immer wieder Funktionstests durchgeführt und der aus den vorherigen Revisionen geschriebene Code an das EV3-konforme Format adaptiert.

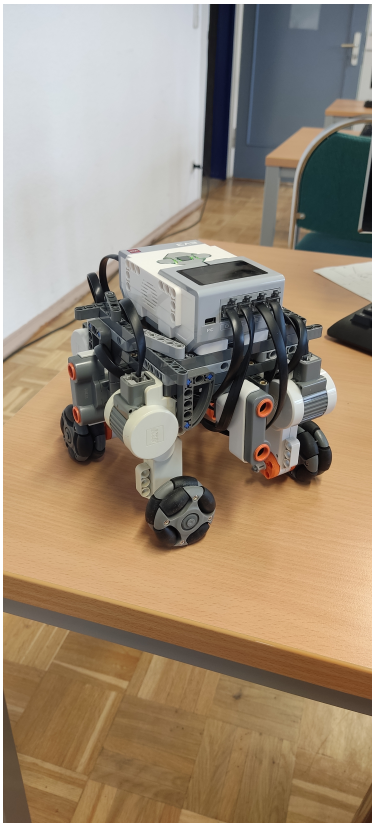


Abbildung 3. Konstruktion 3 – Finaler Prototyp der LABrat

D. Funktionsweise

Mithilfe von MATLAB konnten die angedachten Fahrzeugfunktionen umgesetzt werden. In Abb. 4 ist der Programmablauf des Fahrzeugs vereinfacht dargestellt. Zu Beginn befand sich das Fahrzeug bereits im Labyrinth. Das Ziel war es, den einzigen Ausgang dieses Labyrinths zu finden. Die Grundidee war, dass das Fahrzeug sich stets an der rechten Wand orientierte. Denn unter dieser Bedingung war der Ausgang des Labyrinths stets auffindbar, auch wenn dieser Algorithmus recht rudimentär wirkte. Die ausgestatteten Ultraschallsensoren sendeten kontinuierlich ein Signal aus, welches reflektiert und wieder vom Fahrzeug erfasst wurde. Somit wurde permanent die Entfernung zu potenziellen Hindernissen gemessen und diese ausgewertet. Konkret bedeutete dies, dass nach der Initialisierung des Programms alle Sensoren abgefragt wurden. Die Kunst bestand darin, passende Distanzwerte in das Programm als Referenz einzupflegen, damit das Fahrzeug adäquat auf die Gegebenheiten im Labyrinth reagieren konnte. Wurden die Schwellenwerte für die Distanzen zu einem Hindernis zu groß gewählt, so blieb das Fahrzeug zu weit von den Labyrinthwänden entfernt und erkannte möglicherweise andere potenzielle Labyrinthgänge nicht eindeutig. Kleinere Schwellenwerte hatten zur Folge, dass das Fahrzeug gegen die Hindernisse fuhr, da zu spät auf diese reagiert und abgebremst wurde.

Nach erfolgreicher Abfrage aller Sensoren und beim Unterschreiten eines kritischen Schwellenwertes der Distanz war dem Fahrzeug bekannt, dass eine unmittelbare Kollision mit einem Hindernis drohte. Infolgedessen wurden alle Motoren gestoppt. Wurde der Schwellenwert nicht unterschritten, so blieben die Motoren intakt. Danach erfolgte eine Überprüfung auf Parallelität bezogen auf die Labyrinthwände. Dies geschah über eine Differenzberechnung zwischen dem letzten und dem aktuellen Abstandswert der seitlichen Ultraschallsensoren. Trat eine Differenz auf, so wurden die Motoren gestoppt, falls dies noch nicht geschehen war. Die LABrat drehte sich nachfolgend geringfügig. Die Ursache liegt darin begründet, dass trotz des Fahrens in geraden Richtungen manchmal ein unwillkürlicher Drift seitwärts auftreten konnte. Um dem entgegenzuwirken und um das Fahrzeug parallel zu den Wänden auszurichten, wurde diese minimale Drehung implementiert.

Im weiteren Verlauf wurde die optimale Fahrtrichtung neu berechnet. Dies war relevant, um auf neu auftretende Abzweigungen reagieren zu können. Die Grundannahme, dass das Fahrzeug sich rechts halten sollte, bedeutete, dass nur neu hinzukommende Gänge rechts vom Fahrzeug das Fahrverhalten beeinflussten. In solchen Fällen musste das Fahrzeug vor dem Richtungswechsel eine kurze Strecke geradeaus fahren. Durch die mittig am Fahrzeug angebrachte Sensorik hätte ein plötzlicher Richtungswechsel dazu führen können, dass der hintere Teil des Fahrzeugs gegen eine Wand gestoßen wäre. Um dies zu verhindern, wurde eine Retentionszeit eingepflegt, die dafür sorgte, dass das Stoppen der Motoren etwas verzögerte. Nach einem Richtungswechsel befand sich das Fahrzeug noch in der Kreuzung und die vorhandene Ausrichtung entsprach der neuen Fahrtrichtung. Das hatte wiederum zur Folge, dass ein weiterer ungewollter Richtungswechsel unterbunden werden musste, bis das Fahrzeug sich aus der Kreuzung entfernt hatte. Potenziell musste das Fahrzeug inzwischen die Motoren neu konfigurieren, da sich die Richtung des Fahrzeugs leicht ändern hätte können (leichter Drift möglich).

Auch ein Neustart der Motoren war durchaus möglich, sofern sie vorher gestoppt wurden. Bei aktiven Motoren wurden diese gestoppt, neu konfiguriert und erneut gestartet. Um unnötige Start- und Stoppvorgänge zu minimieren und damit das Risiko von unerwünschten Drehungen zu reduzieren, waren die Motoren so eingestellt, dass sie bis zum Erreichen eines Stoppsignals weiter drehten. Aus diesem Grund wurden die Motoren synchron gestartet. Durch die Neuberechnung der Richtung in jedem Schleifen-Durchlauf fuhr das Fahrzeug stets in die optimale Richtung. Die Schleife wurde so lange iteriert, bis der Ausgang des Labyrinths gefunden wurde. Nach der Flucht aus dem Labyrinth konnte die LABrat inaktiviert werden.

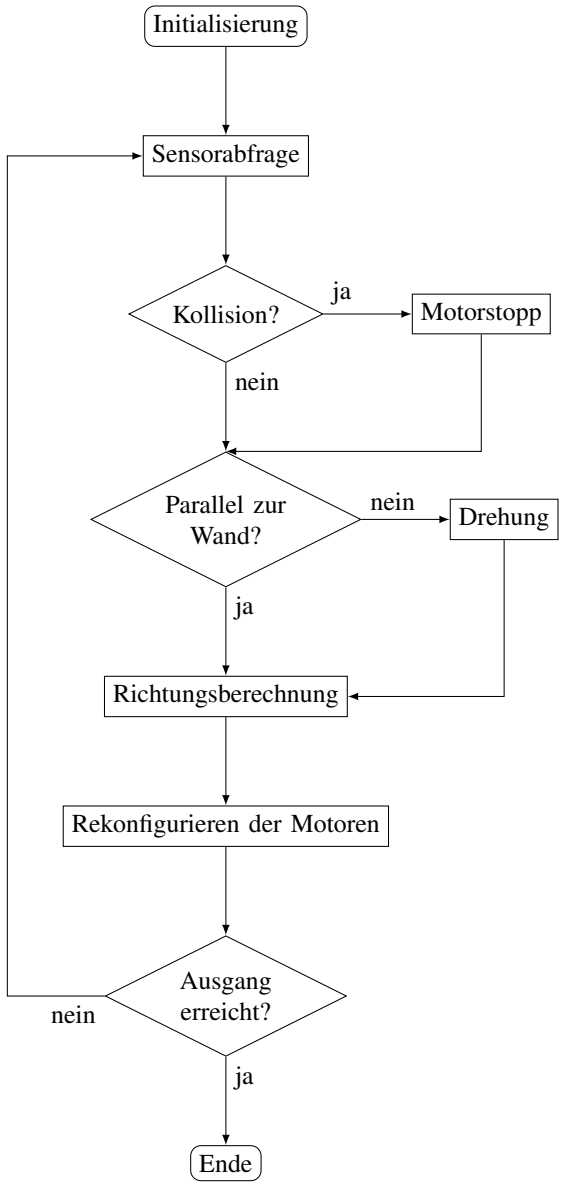


Abbildung 4. Programmablaufplan des finalen Prototyps

IV. Ergebnisdiskussion

Es zeigt sich, dass das konzipierte Fahrzeug grundsätzlich funktionierte. Die Bewegung als auch der Auswahlalgorithmus wurden im Kern so ausgeführt, wie es angedacht war. Allerdings gab es Probleme bei der Skalierung. In einem kleinen Maßstab funktionierte das Fahrzeug häufig zuverlässig, allerdings traten Komplikationen auf, je größer das Labyrinth gestaltet war. Es ließen sich drei Problemstellen feststellen. Die erste ist die Umgebung. Es konnte nicht vollständig kalkuliert werden, wie sich die Bodenbeschaffenheit auf das Fahrverhalten ausübte. Allerdings kann davon ausgegangen werden, dass dieses Problem im Vergleich zu den anderen beiden vermutlich eher marginaler Natur war.

Das zweite Problem war die Konstruktion und die LEGO-Bausteine insgesamt. Es kann festgehalten werden, dass horizontale Kräfte auf das Fahrzeug wirkten, wenn dieses sich vorwärts, rückwärts oder seitwärts bewegte. Diese Kraft wirkte natürlich auch auf die nach unten gerichteten Motoren, die sich wiederum aufgrund ihrer Trägheit ein wenig in die entgegengesetzte Fahrtrichtung verschoben hatten. Dadurch kam es zu ungewollten Bewegungen des Fahrzeuges. So entstand ein Drift nach links oder rechts, wenn das Fahrzeug nur geradeaus fahren sollte.
Das dritte Problem waren die toten Winkel der vier Ultraschallsensoren. Diese waren im rechten Winkel zueinander angeordnet. Das bedeutete aber auch, dass zwischen zwei Ultraschallsensoren ein Raum entstand, der nicht aktiv gemessen wurde. Befand sich dort ein Objekt und wich das Fahrzeug leicht von seinem Kurs ab, konnten Kollisionen entstehen, die nicht auftreten sollten.

V. Zusammenfassung und Fazit

Es lässt sich zusammenfassen, dass mithilfe von LEGO und MATLAB kreative Projekte ermöglicht werden können. Die Idee, ein Fahrzeug zu entwickeln, welches selbstständig einen bestimmten Weg abfährt, kann verwirklicht werden. Eigenschaften wie Fahren, Hinderniserkennung und die Implementation eines Algorithmus lassen sich gut in der Realität umsetzen. Es kann festgehalten werden, dass ein Projekt stets mit einem Konzept angegangen werden sollte, bei dem zu Beginn im kleinen Maßstab gearbeitet wird und dann ein Hochskalieren erfolgen kann. Konkret bedeutet das für die LABrat, dass die Konstruktion nochmals genauer betrachtet werden muss, um die Stabilität des Fahrzeuges auf ein neues Niveau zu heben. Auch müsste überlegt werden, wie mit den toten Winkeln zwischen den Ultraschallsensoren verfahren werden kann. Mehr als vier Ultraschallsensoren sind aufgrund der EV3-Bauweise nicht möglich. Vielleicht wäre ein anderer Ansatz eine optische Erkennung von Objekten und Hindernissen mithilfe einer Kamera. Ferner muss bedacht werden, dass das Fahrzeug lediglich Anweisungen erledigte. Der LABrat war nicht bekannt, wo sie sich im Labyrinth befand. Wenn eine Art Kartografierung etabliert werden kann, dann wäre das Durchfahren des Labyrinths vermutlich deutlich weniger fehleranfällig.

Literaturverzeichnis

[1] BMW: *Sensoren in Autos: Sinnesorgane der Assistenz-Systeme*. 2021. – https://www.bmw.com/de/innovation/sensoren-im-auto.html
zuletzt aufgerufen am 23. Februar 2024, 19:54

[2] Audi: *Fahrerassistenzsysteme: Audi Q3 Sportback*. 2019. – https://t1p.de/ngofo
zuletzt aufgerufen am 23. Februar 2024, 19:54

Multifunktionaler Roboter

Buchniev Danyil, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In diesem Beitrag wird das Projekt 'Multifunktionsroboter' vorgestellt. Der Roboter verfügt über zwei Hauptmodi, nämlich die Lichtsuche und die Fernbedienung, sowie vier Untermodi, die Vorwärts-, Rückwärts-, Rechts- und Linksbewegungen ermöglichen. Während des Projekts wurden Lego EV3 und Arduino Mega 2560 verwendet. Dank dieser Entwicklung kann der Roboter mittels IR-Steuerung in vier Richtungen gesteuert werden.**

***Schlagwörter*—Arduino, Farbsensor, Infrarotsensoren (IR), LEGO Mindstorms EV3, Lichtsensor, Robotersteuerung, Schrittmotor**

I. Einleitung

Im Zuge des modernen epochalen Übergangs zur Industrie 4.0 und dem aktiven Einsatz von Robotik in verschiedenen Branchen sind die Entwicklung und Implementierung multifunktionaler Robotersysteme zu einem integralen Bestandteil des technologischen Fortschritts geworden. In diesem Zusammenhang ist die Forschung an einem Roboter auf Basis des EV3-Boards von Lego Mindstorms von großer Bedeutung. Dieser Roboter ist ein bemerkenswertes Beispiel für eine innovative Synthese aus Technik und Programmierung mit erheblichem Potenzial zur Verbesserung von Produktionsprozessen, Steigerung der Effizienz und Einführung automatisierter Systeme in verschiedenen Branchen.

II. Vorbetrachtungen

Ursprünglich war geplant, den Roboter mithilfe eines Farbsensors und einer RGB-LED zu steuern. Hierbei sollte die LED durch einen Arduino-Mikrocontroller gesteuert werden, welcher mit einem Infrarotempfänger ausgestattet ist, um Signale von der Fernbedienung zu empfangen. Der LEGO EV3 sollte die Farbe der LED (Rot, Blau, Grün) interpretieren. Leider konnte der Farbsensor die Strahlen unserer LED nicht erkennen. Für die Stromversorgung des Arduino Mega 2560 wird empfohlen, ein Leistungselement wie die 9 V-Krona-Batterie zu verwenden.

A. Fernsteuersystem

Das Handsteuersystem (siehe Abbildung 1) ist ein zentraler Aspekt der multifunktionalen Robotersteuerung. Es ermöglicht dem Bediener, den Roboter präzise in alle Richtungen zu bewegen, indem er ein drahtloses Steuergerät wie ein Pult verwendet. Der Roboter kann sich in alle vier Hauptrichtungen bewegen: vorwärts, rückwärts, nach links und nach rechts. Durch eine geschickte Integration von Motorsteuerungen und drahtloser Signalübertragung wird es ermöglicht, dass der Bediener den Roboter durch den Raum steuern kann, indem er die gewünschte Bewegungsrichtung präzise auswählt und das Handsteuersystem intuitiv bedient.

DOI: 10.24352/UB.OVGU-2024-024

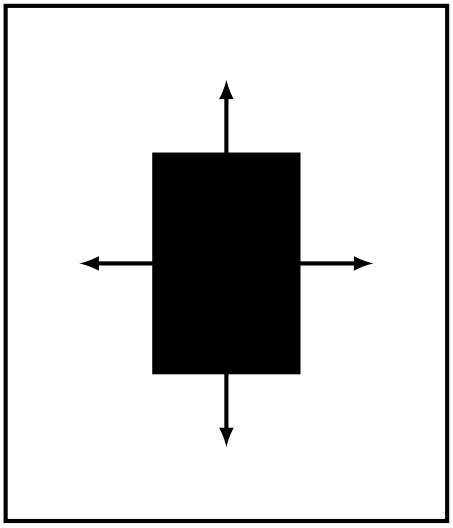


Abbildung 1: Vier Seiten zum Fahren

Es ergeben sich verschiedene Anwendungsszenarien und Nutzen:

- Präzise Steuerung: Das Handsteuersystem ermöglicht eine äußerst präzise Manövrierfähigkeit, was insbesondere in engen Räumen oder bei der präzisen Anfahrt spezifischer Positionen von großem Nutzen ist.
- Das Handsteuersystem ermöglicht eine schnelle und flexible Steuerung durch den Bediener, um auf unvorhersehbare Ereignisse in dynamischen Umgebungen zu reagieren. Wir sind überzeugt, dass dieser System eine schnelle Reaktion auf Veränderungen ermöglicht

Das Handsteuersystem erweitert die Einsatzmöglichkeiten des Roboters insgesamt. Es ermöglicht eine direkte und unmittelbare Steuerung durch den Bediener und deckt somit eine breite Palette von Anwendungsszenarien ab.

B. Lichtsuchmodus

Der Lichtsuchmodus ist eine Hauptfunktion des Roboters (siehe Abbildung 2). Er erlaubt es dem Roboter, Lichtquellen autonom zu verfolgen. Sobald eine Lichtquelle erkannt wird, bewegt sich der Roboter selbstständig in deren Richtung. Der Prozess erfolgt durch die Auswertung der Daten des Lichtsensors, welcher die Lichtintensität misst. Dadurch ist es dem Roboter möglich, auf Veränderungen in der Umgebung zu reagieren.

Anwendungszenarien und Nutzen:

- Der Roboter kann in unzugänglichen oder gefährlichen Umgebungen eingesetzt werden, in denen menschlicher Zugang schwierig ist. Der Roboter kann in unzugänglichen

oder gefährlichen Umgebungen eingesetzt werden, in denen menschlicher Zugang schwierig ist. Der Roboter kann in unzugänglichen oder gefährlichen Umgebungen eingesetzt werden, in denen menschlicher Zugang schwierig ist. Mit dem Lichtsuchmodus kann er autonome Lichtquellen verfolgen und Umgebungsdaten sammeln

- Unterstützung bei Such- und Rettungseinsätzen: Wir sind überzeugt, dass der Lichtsuchmodus eine wertvolle Ergänzung für Such- und Rettungseinsätze darstellt. Der Lichtsuchmodus des Roboters kann in begrenzter Sicht, Überlebende oder Gefahrenquellen autonom lokalisieren und verfolgen.

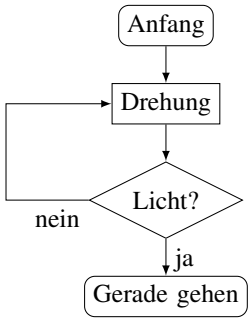


Abbildung 2: Programmablaufplan

C. Zusammenwirken des Motors und der Farbscheibe.

Für dieses Projekt wurden ein Schrittmotor (siehe Abbildung 3) und eine Farbscheibe (siehe Abbildung 4) als Verbindungsglieder zwischen LEGO und Arduino eingesetzt.

Durch die enge Koordination zwischen Motor und Farbscheibe kann der Roboter basierend auf Farberkennung präzise gesteuert werden. Die Farbscheibe ist mit verschiedenen Farben ausgestattet und rotiert in Verbindung mit dem Motor, wodurch unterschiedliche Farbsignale an den Farbsensor übermittelt werden. Der Farbsensor wertet die Signale aus und gibt präzise Steuerbefehle an den Roboter weiter.

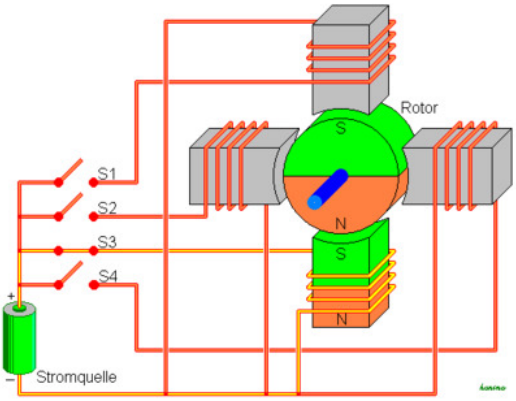


Abbildung 3: Schrittmotor [1]

III. Aufbau

Unser Projekt konzentriert sich auf die Entwicklung und Umsetzung eines vielseitigen Roboters, der durch die EV3 - Stein (siehe Abbildung 5) von Lego Mindstorms und Arduino

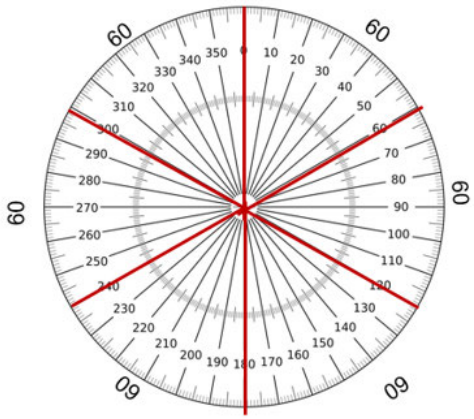


Abbildung 4: Kreis der Farbscheibe

Mega 2560 (siehe Abbildung 6) gesteuert wird. In dieser Hauptsektion werden die Kernelemente unseres Konzepts erläutert und die Umsetzung in der Realität beschrieben. Es wird ein Einblick in den Prozess der Schaffung dieses multifunktionalen Roboters gegeben.

Abbildung 5: EV3 [2]

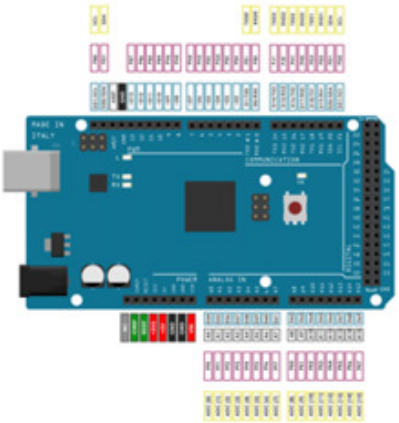


Abbildung 6: Arduino [3]

A. Idee und Konzept:

Unsere Ausgangsidee war die Entwicklung eines Roboters, der nicht nur über eine Fernbedienung gesteuert werden kann, sondern auch autonom auf Lichtquellen reagiert. Der Kernmechanismus besteht darin, dass der Roboter nach Lichtquellen sucht, sich zu ihnen ausrichtet und ihnen folgt, wenn das Licht ausreichend intensiv ist. Allerdings konnte der Farbsensor (siehe Abbildung 7a aufgrund reflektierender Lichtstrahlen die Farbveränderungen vor dem EV3-Sensor nicht zuverlässig erkennen. Zunächst musste die Steuerung der RGB-LEDs des Roboters implementiert werden. Die Farbveränderungen, die durch einen drehbaren Farbkreis (siehe Abbildung 7b gesteuert werden, beeinflussen die Bewegungsrichtung und den Modus des Roboters. Dadurch entsteht ein vielseitiger Einsatzbereich, sei es durch manuelle Steuerung oder autonomes Verhalten aufgrund von Lichtveränderungen. Im Folgenden werden unsere Realisierung dieses Konzepts gezeigt (siehe Tabelle I, Abbildung 8a und Abbildung 8b).

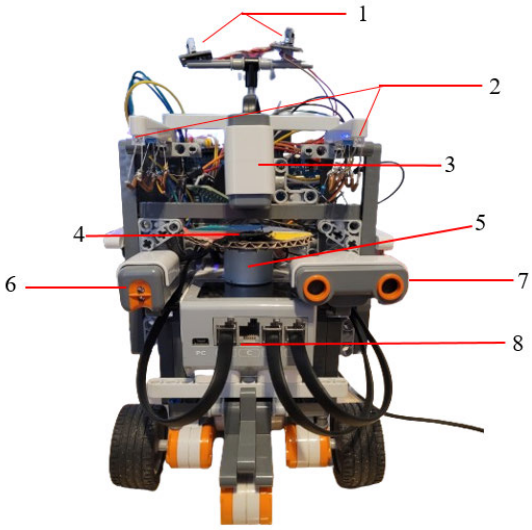
(a)

(b)

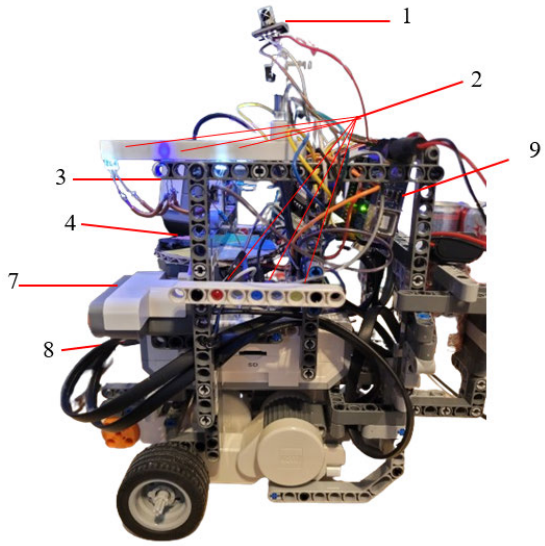
Abbildung 7: Farbsensor und Farbkreis

№	Bezeichnung
1	Infrarotsensoren (IR)
2	LEDs
3	Farbsensor
4	Farbkreis
5	Schrittmotor
6	Lichtsensor
7	Ultraschallsensor
8	LEGO EV-3
9	Arduino Mega 2560

Tabelle I: Passende Nummern und Details



(a) Vorderseite



(b) Seite nansicht des Roboters

Abbildung 8: Aussehen

B. Hardware und Mechanik:

Die Konstruktion umfasst zwei Rädermotoren sowie drei Sensoren: einen Annäherungssensor, einen Lichtsensor und einen Farbsensor. Zusätzlich gibt es 9 Lichtdioden, um eine Bewegung klar zu zeigen. Die Umsetzung beinhaltet auch die Integration eines Schrittmotors, der über eine Arduino-Platine gesteuert wird. Die Stromversorgung erfolgt durch drei parallel geschaltete Batterien (siehe Abbildung 9) mit der Gesamtkapazität = 1,5 Ah (siehe Formel 1), die jedem Arduino-Element (siehe Abbildung 10) die notwendige Energie liefern. Dies ermöglicht eine effiziente und mobile Stromversorgung.

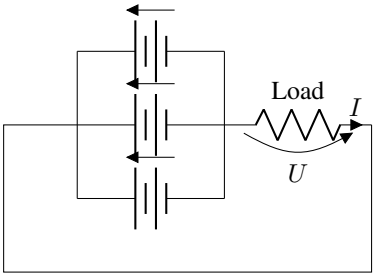


Abbildung 9: Batterieverteilung [5]

$$Cgesamt = \sum_{i=1}^{n} Ci = 0{,}5\,\text{Ah} + 0{,}5\,\text{Ah} + 0{,}5\,\text{Ah} = 1{,}5\,\text{Ah} \tag{1}$$

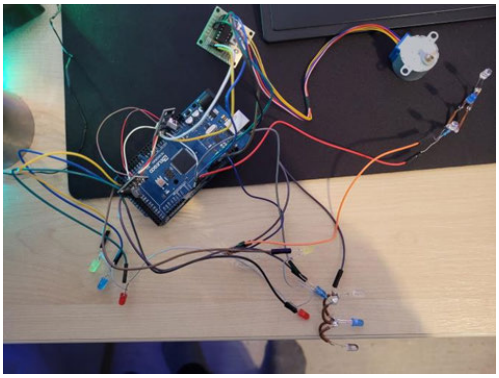


Abbildung 10: Arduino und Teile

C. Software und Programmierung:

Die Programmierung wurde sowohl in MatLab als auch in der Programmiersprache C für die Arduino-Platine durchgeführt. Die Kommunikation zwischen dem Roboter und der Fernbedienung erfolgt über Infrarotsignale, die von einer Antenne empfangen werden. Bestimmte Bewegungsrichtungen werden durch Farbänderungen vor dem Farbsensor ausgelöst, wobei der schwarze Farbzustand den Ruhezustand des Roboters markiert (siehe Abbildung 11).

IV. Ergebnisdiskussion

Der multifunktionale Roboter wurde erfolgreich entwickelt und realisiert. Er wird durch die EV3-Platine von Lego Mindstorms gesteuert und verfügt über vielseitige Steuerungsoptionen. Dazu gehört die Möglichkeit, per Fernbedienung bewegt zu werden und autonom auf Lichtquellen zu reagieren. Die Integration von Hardwarekomponenten wie Motoren und Sensoren sowie einer Arduino-Platine ermöglicht eine reibungslose Zusammenarbeit und vielfältige Funktionalität. Eine sorgfältige Softwareprogrammierung ist von großer Bedeutung.

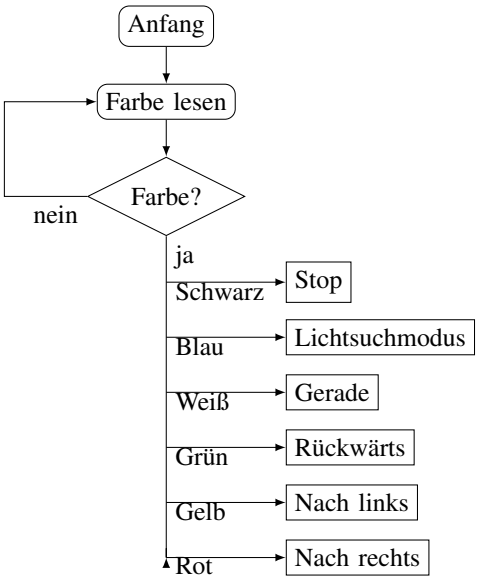


Abbildung 11: Programmablaufplan

V. Zusammenfassung und Fazit

Das Projekt hat wertvolle Lernerfahrungen in den Bereichen Robotik, Elektronik und Programmierung auf zwei Programmiersprachen gebracht. Die Herausforderungen während des Prozesses haben zu einem besseren Verständnis der Wechselwirkungen zwischen Hardwarekomponenten und Softwaresteuerung geführt.

Literatur

[1] Wikipedia. *https://de.wikipedia.org/wiki/Schrittmotor*. 2023.
[2] Brickmerge. *LEGO® Mindstorms 45500 Intelligenter EV3-Stein*. https://www.brickmerge.de/45500-1_lego-mindstorms-intelligenter-ev3-stein
[3] ElectronicsHub. *Arduino Mega Pinout | Arduino Mega 2560 Layout, Specifications*. https://www.electronicshub.org/arduino-mega-pinout/
[4] Steinelager. *LEGO® Mindstorms MINDSTORMS® NXT Farbsensor 9694*. https://www.steinelager.de/de/set/9694-1/mindstorms-nxt-farbsensor
[5] DigiKey. *Reihen- und Parallelschaltung von Batterien*. https://www.digikey.de/de/blog/series-and-parallel-battery-circuits
[6] Vishay. *DataIR Receiver Modules for Remote Control Systems*. Vishay Semiconductors, 2017.

NXT-Roboter mit Fernbedienung

Danylo Tsoi, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Projektseminar Elektrotechnik und Informati- onstechnik wurde ein innovativer Roboter entwickelt, der vielsei- tige Funktionen bietet. Mithilfe von Arduino-Programmierung wurde der Roboter so konzipiert, dass er ferngesteuert werden kann und in der Lage ist, Lichtquellen zu erkennen und ihnen zu folgen. Die Hauptziele waren die Steuerung über eine Fernbedie- nung und die autonome Lichtverfolgung. Dabei wurde ein Schritt- motor für präzise Bewegungen integriert. Die Verwendung von LEDs ermöglichte eine effektive visuelle Signalisierung. Der Inf- rarotport wurde für die drahtlose Steuerung über eine Fernbedie- nung implementiert. Eine wichtige Designüberlegung war die si- chere und zuverlässige Stromversorgung, die durch verschiedene Energiequellen gewährleistet wurde.**

***Schlagwörter*—Arduino-Programmierung, Farbsensor, Infrarotsensoren, LEGO Mindstorms EV3, Lichtsensor, Robotersteuerung, Schrittmotor.**

I. Einleitung

Roboter sind heutzutage unverzichtbare Komponenten in verschiedenen Bereichen der Industrie. Sie automatisieren Pro- zesse und ermöglichen eine schnelle und präzise Verarbeitung von Massenwaren. Besonders im Bereich der Logistik spielen automatisierte Transportsysteme eine wichtige Rolle, um die Arbeitseffizienz zu steigern und den Materialfluss zu beschleu- nigen. In diesem Projekt wurde ein NXT-Roboter mit Fernbe- dienung entwickelt, der mithilfe der LEGO Mindstorms EV3- Platine gesteuert wird. Dieser Roboter verfügt über die Fähig- keit, sich mithilfe von Sensoren in seiner Umgebung zu orien- tieren und auf Lichtquellen zu reagieren. Im Folgenden wird die Konstruktion und Funktionsweise dieses Roboters genauer be- schrieben

II. Vorbetrachtungen

Die erste Idee umfassten die Implementierung eines ferngesteuerten LED vor dem Sensor, um ihn ferngesteuert zu schalten. Hierbei war geplant, die LED mithilfe eines Arduino-Mikrocontrollers zu steuern, der mit einem Infrarotempfänger ausgestattet ist, um Signale von der Fernbedienung zu empfangen (siehe Abbildung 1). Der Sensor von LEGO Mindstorms EV3 sollte die Lichtintensität messen und entsprechend auf die Beleuchtung reagieren. Darüber hinaus sollte der Roboter auf Ketten basieren, um eine bessere Traktion und Manövrierfähigkeit auf verschiedenen Oberflächen zu gewährleisten. Alle Komponenten des Roboters sollten so konzipiert sein, dass sie effizient von einer einzigen Batterie betrieben werden können, um die Mobilität des Roboters zu maximieren und die Anzahl der benötigten Energiequellen zu minimieren.

DOI: 10.24352/UB.OVGU-2024-025

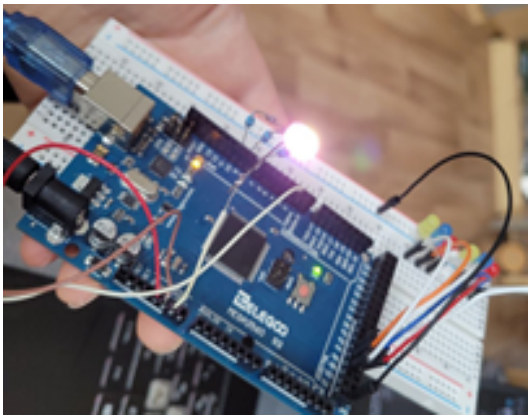


Abbildung 1: RGB-LED mit dem Arduino-Mikrocontroller

III. Entwicklungsprozess

A. Aufbau

Der Roboter (siehe Abbildung 2) besteht aus einem LEGO-Rahmen, auf dem verschiedene Komponenten montiert sind, darunter Sensoren, Motoren und eine Arduino-Steuerung. Zentrale Elemente sind die Motoren für die Bewegung des Roboters, die Tastsensoren zur Interaktion mit der Umgebung sowie die Webcam zur Farberkennung. Die Konstruktion wurde so entworfen, dass sie kompakt und stabil ist, um die ordnungsgemäße Funktion des Roboters zu gewährleisten.

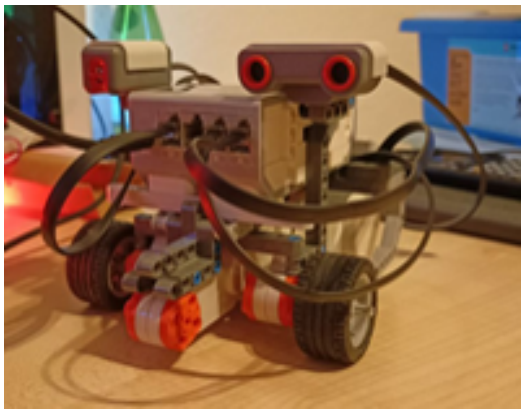


Abbildung 2: Roboter ohne Arduino-Board

B. Konstruktions-Entwicklung

LEGO-Gestell: Der Aufbau des Roboters begann mit der Montage des LEGO-Rahmens, wobei darauf geachtet wurde, dass genügend Platz für alle Komponenten vorhanden ist und die Konstruktion stabil ist. Die Anordnung der Motoren, Sen-

soren und der Webcam wurde sorgfältig geplant, um eine effiziente Funktionsweise sicherzustellen.

Elektronik: Die elektronischen Komponenten, einschließlich der Arduino-Steuerung (siehe Abbildung 3), wurden mit dem LEGO-Rahmen verbunden und so positioniert, dass sie leicht zugänglich und miteinander verbunden sind. Die Verkabelung wurde ordentlich verlegt, um Störungen zu vermeiden und eine zuverlässige Stromversorgung sicherzustellen.

Mechanik: Die Bewegungsmechanik des Roboters wurde durch die Montage der Motoren und Räder realisiert, wobei da- rauf geachtet wurde, dass der Roboter sich reibungslos bewegen kann und gleichzeitig stabil bleibt. Die Integration des drehba- ren Rads zur Orientierung des Roboters wurde ebenfalls be- rücksichtigt.

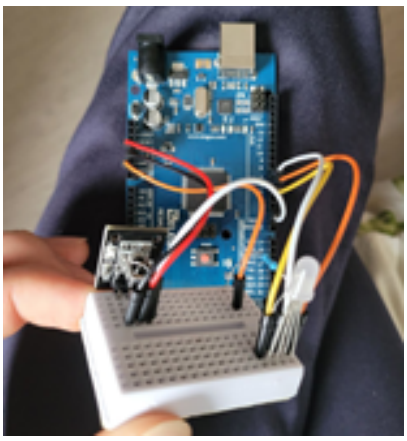


Abbildung 3: Arduino-Steuerung

C. Programmierungs-Entwicklung

Die Funktionsweise des Roboters wurde durch die Programmierung der Arduino-Steuerung realisiert. Dies umfasst die Steuerung der Motoren für die Bewegung des Roboters, die Verarbeitung von Eingaben von den Tastsensoren zur Interaktion mit der Umgebung, die Verarbeitung von Bildern von der Webcam zur Farberkennung sowie die Implementierung von Algorithmen zur Entscheidungsfindung und Steuerung des Roboters. Allerdings musste auch MATLAB-Code implementiert werden, um den Roboter vollständig funktionsfähig zu machen. Die Logik dafür ist auf die Abbildung 5 im Anhang A dargestellt.

D. Probleme

Während des Entwicklungsprozesses traten verschiedene Probleme auf, darunter Herausforderungen bei der Integration der elektronischen Komponenten, Kalibrierung der Sensoren und Feinabstimmung der Bewegungsmechanik. Diese Probleme wurden durch iterative Tests und Anpassungen der Konstruktion und Programmierung gelöst.

E. Ausbaumöglichkeiten

Der Roboter bietet zahlreiche Möglichkeiten zur Weiterentwicklung, darunter die Integration zusätzlicher Sensoren für erweiterte Funktionalitäten, die Implementierung fortschrittlicherer Bildverarbeitungsalgorithmen für eine präzisere Farberkennung und die Erweiterung der Steuerungsoptionen für eine vielseitigere Interaktion mit der Umgebung.

IV. Ergebnissdiskussion

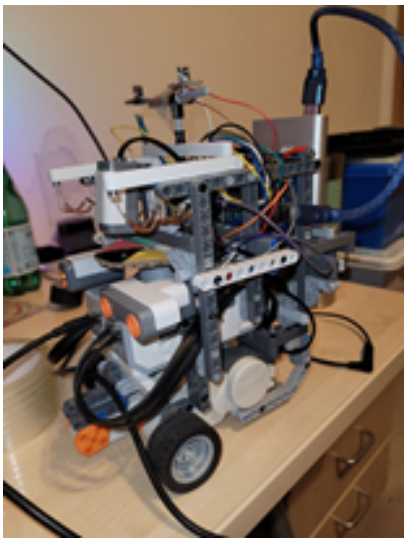


Abbildung 4: NXT-Roboter

Am Ende des Projekts wurde ein funktionsfähiger Roboter entwickelt (siehe Abbildung 4), der sich mit Hilfe von Infrarotsensoren, einem Schrittmotor und einem Farbsensor bewegen und interagieren kann. Der Roboter kann verschiedene Bewegungen ausführen, z. B. vorwärts und rückwärts gehen und sich nach links und rechts drehen. Außerdem kann er auf Lichtquellen reagieren und ihnen folgen, was eine autonome Navigation ermöglicht. Während des Projekts traten jedoch einige Probleme auf. Insbesondere die Integration des MATLAB-Codes erwies sich als Herausforderung, da eine nahtlose Kom- munikation zwischen dem Arduino-Programm und dem MATLAB-Code erforderlich war. Dies erforderte eine gründliche Fehlersuche und Anpassung, um sicherzustellen, dass beide Systeme reibungslos zusammenarbeiten. Ein weiteres Problem bestand darin, die Genauigkeit der Farberkennung zu verbessern, insbesondere bei schwierigen Lichtverhältnissen oder bei der Erkennung bestimmter Farben wie Braun oder Schwarz. Dies erforderte eine Feinabstimmung der Sensorempfindlichkeit und der Bildverarbeitungsalgorithmen. Trotz dieser Herausforderungen war das Endergebnis des Projekts zufriedenstellend, da ein funktionsfähiger und vielseitiger Roboter entwickelt wurde, der die gestellten Anforderungen erfüllt und eine solide Grundlage für zukünftige Entwicklungen in diesem Bereich bietet.

V. Zusammenfassung und Fazit

Die Zusammenfassung dieses Projekts zeigt, dass ein funktionsfähiger Roboter entwickelt wurde, der in der Lage ist, verschiedene Bewegungen auszuführen und mit seiner Umgebung zu interagieren. Trotz einiger Herausforderungen bei der In-

tegration von MATLAB und der Verbesserung der Farberkennung konnte das Projekt erfolgreich abgeschlossen werden. Für zukünftige Entwicklungen könnten weitere Verbesserungen an der Farberkennung vorgenommen werden, um die Genauigkeit und Zuverlässigkeit des Roboters weiter zu erhöhen. Außerdem könnten zusätzliche Funktionen wie die Implementierung weiterer Sensoren oder die Erweiterung der Bewegungsfähigkeit des Roboters in Betracht gezogen werden.

VI. Anhang A

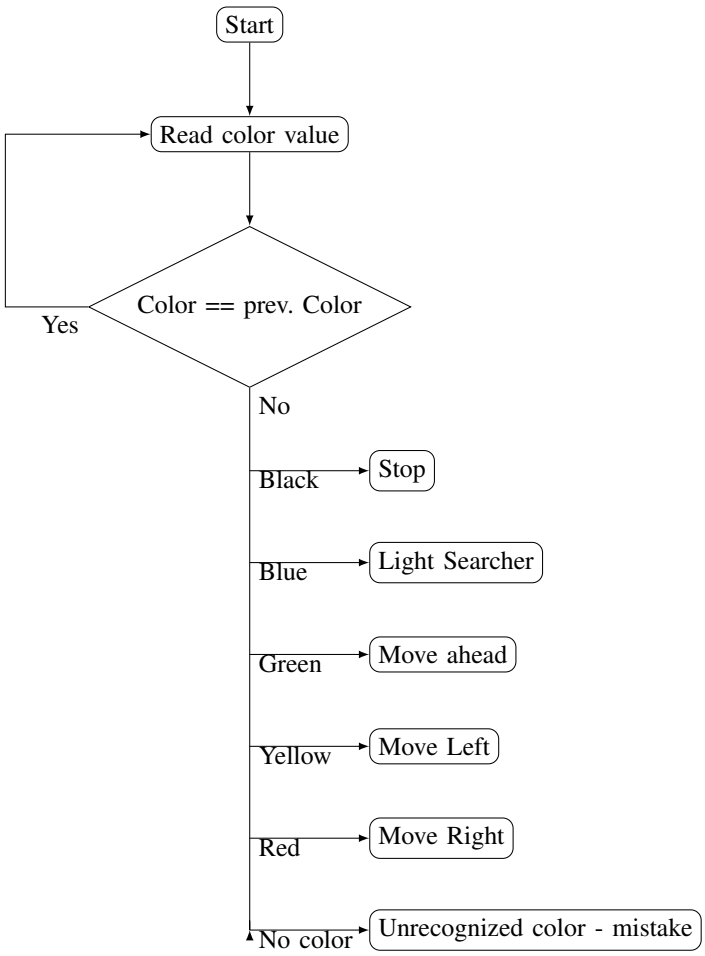


Abbildung 5: While-Loop Logik

Literatur

[1] Builderdude35. *Master the LEGO color sensor*.https://builderdude35.com/2021/01/28/master-the-color-sensor-lego-mindstorms-robot-inventor/. Januar 2021.

[2] Arduino. (Last revision: Februar 2024). *Arduino and Stepper Motor Con- figurations. Electronics*. https://docs.arduino.cc/learn/electronics/stepper-motors/

[3] Arduino. (Last revision: Februar 2024). *Blink. Basics*. https://docs.arduino.cc/built-in-examples/basics/Blink/

[4] Debashis Das. *Interfacing IR Sensor Module with Arduino. Arduino*.https://circuitdigest.com/microcontroller-projects/interfacing-ir-sensor-module-with-arduino. März 2022.

Der Tic-Tac-Toe-Robo

Frodo Hinze, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Rahmen des Projektseminars Elektrotechnik/Informationstechnik wurde ein Roboter entwickelt, der gegen einen menschlichen Spieler im Spiel Tic Tac Toe antreten kann. Anstelle von Kreuzen und Kreisen verwendet der Roboter rote und grüne Kugeln. Der Spieler kann eine Kugel auf dem Spielfeld platzieren und den Roboter mit einem Knopfdruck zum Zug auffordern. Der Roboter erfasst das Spielfeld mit einem Farbsensor, indem er das Spielfeld dreht und den Arm an dem der Farbsensor befestigt ist vor und zurück bewegt. Anschließend wird der bestmögliche Zug für den Roboter mit Hilfe des Min-Max-Algorithmus berechnet. Der Zug wird mit einem Kugelauswerfer, der sich ebenfalls am Arm befindet, ausgeführt. Die Steuerung erfolgt durch einen EV3-Controller, der die Befehle eines MATLAB-Skripts über die USB-Schnittstelle von einem Laptop empfängt und die Befehle ausführt.**

***Schlagwörter*—EV3, Min-Max-Algorithmus, Spiel, Tic Tac Toe**

I. Einleitung

SPIELEND lernen ist ein Grundsatz der Wissensvermittlung. Eben dieser Grundsatz wurde auch im vorliegenden Projekt angewendet. Als Spiel wurde das allgemein bekannte Spiel Tic Tac Toe, ausgewählt. Das Ziel bestand darin, einen Roboter zu konstruieren und zu programmieren, der gegen einen menschlichen Spieler Tic Tac Toe spielen kann.

II. Vorbertrachtung

Im Rahmen des zweiwöchigen Projektseminars bestand der Arbeitsauftrag darin, einen LEGO-Roboter zu konstruieren und mit MATLAB zu programmieren. Der Roboter sollte in der Lage sein, das Spiel „Tic Tac Toe“ zu spielen.

Der Plan sah vor, dass der Spieler eine farbige Kugel auf das Spielfeld legt. Anschließend sollte der Roboter das Spielfeld einlesen und seinen bestmöglichen Zug berechnen. Außerdem sollte der Roboter in der Lage sein, selbstständig eine Kugel auf das Spielfeld zu legen, um so einen natürlichen Spielfluss zu ermöglichen.

In der Abbildung 1 sind die wesentlichen Bestandteile des Tic-Tac-Toe-Robos (im Folgenden T3R) bereits gut zu erkennen. Das Spielfeld ist in der Skizze oberhalb des Motors M1 zu sehen. Dieser soll das Spielfeld drehen, damit die Kugeln platziert werden können. Ein fahrbarer Arm (M2) mit einem Farbsensor (S1) kann dann das Spielfeld einlesen. Anschließend kann ein Kugelauswurfmechanismus eine Kugel für den Roboter auf dem Spielfeld abwerfen. Um dem Roboter zu signalisieren, dass der Spieler seinen Zug gespielt hat, war ein Drucksensor (S2) vorgesehen, der als Knopf fungieren sollte. Ein weiterer Drucksensor (S3) hatte die Aufgabe, das Spielfeld so zu drehen, dass der Spieler auch an die Felder gelangen konnte, die andernfalls vom Farbsensor und dem Kugelauswurf verdeckt worden wären.

DOI: 10.24352/UB.OVGU-2024-026

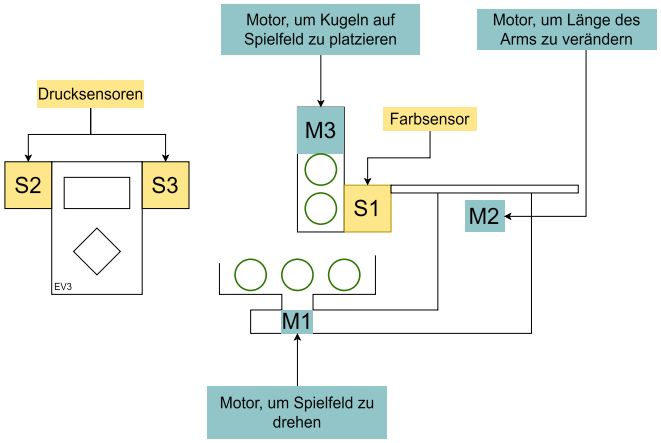


Abbildung 1. Arbeitsskizze des Roboters aus früher Entwicklungsstufe

Die Steuerung erfolgt über einen LEGO Mindstorms EV3-Controller, der MATLAB-Skripte ausführt. Der Zugriff auf den Controller erfolgt über die von der RWTH Aachen erstellte Bibliothek „RWTH Mindstorms EV3 Toolbox“, welche über das e-Learning-System der Universität zur Verfügung gestellt wurde. Zunächst fühlte sich die Verwendung des EV3 wie ein Nachteil an, da viele andere Gruppen einen NXT-Controller verwendeten. Jedoch zeige sich schnell, dass zur Ansteuerung des EV3 keine zusätzlichen Treiber verwendet werden mussten. Der Controller musste lediglich mit einem Laptop via USB verbunden werden. Die einzige Schwierigkeit bestand darin, dass die Verbindung ab und zu zurückgesetzt werden musste; nachdem dieser Schritt Einzug in jedes Skript fand, stellte auch das kein Problem mehr da.

Die Logik des T3R basiert auf dem Min-Max-Algorithmus. Eine Implementierung dieses Algorithmus' in Python existiert auf YouTube [1] und wurde für das Programm des Roboters entsprechend in MATLAB umgesetzt. Anschließend mussten noch einige kleine Änderungen implementiert werden, um das Zusammenspiel mit den restlichen Funktionen zu gewährleisten.

III. Aufbau & Programmierung

Im Folgenden wird der Aufbau des T3R erläutert.

A. Hardware

1) Spielfeld: Ein wichtiger Teil des Roboters ist das Spielfeld. Auf ihm soll die eigentliche Interaktion mit dem Spieler stattfinden. Um dies zu ermöglichen, wurde der Roboter mit einem Spielfeld der Größe 3×3 ausgestattet, das auf den Durchmesser der verwendeten LEGO-Kugeln abgestimmt ist. Nach einigen Tests wurde der Spieltisch mit weißen Papierschnipseln ergänzt, da dies die Erkennung des Spielfeldes stark

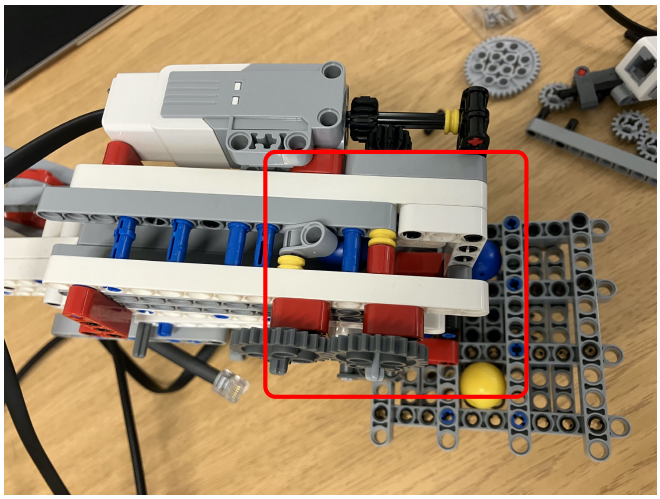


Abbildung 2. Kugelauswurf am Roboter

vereinfachte und die Zuverlässigkeit der Spielfelderkennung deutlich verbesserte. Die Platte, auf der sich das Spielfeld befindet, wird von einem großen Motor angetrieben. Dadurch ist es möglich, den Teller zu drehen.

2) Arm: Wie in Abbildung 1 zu sehen ist, wird das Spielfeld von einem Ausleger überragt. Dieser wird von einem großen Motor vor und zurück bewegt. Weitere Freiheitsgrade sind hier nicht notwendig, da durch das Drehen des Spielfeldes und das Vor- und Zurückbewegen des Arms bereits alle Positionen erreichbar sind. Am Arm befinden sich zwei weitere wichtige Komponenten des Roboters: der Farbsensor und der Auswurfmechanismus. Beide sind in der Abbildung 3 oberhalb des Spielfeldes gut zu erkennen.

3) Farbsensor: Der Farbsensor befindet sich auf der linken Seite vor dem Auswurfmechanismus. Die Aufgabe des Farbsensors ist es, den Zustand des Spielfeldes zu erkennen. In Tests zeigt sich, dass der Farbsensor rote und grüne Kugeln zuverlässiger erkennt als andere Farben. Dafür ist es jedoch notwendig, dass der Farbsensor sehr dicht über dem Spielfeld platziert wird. Unterhalb des Arms wurden zwei kleine Stifte angebracht, die eine genauere Justierung der Armhöhe und damit des Farbsensors ermöglichen.

4) Kugelauswurf: Das zweite Element des Arms ist weniger anfällig für Höhenunterschiede als der Farbsensor. Der Kugelauswurfmechanismus ermöglicht es dem Roboter, Kugeln auf dem Spielfeld zu platzieren. Hierfür wurde ein Kanal konstruiert, in den vor Spielbeginn maximal fünf Kugeln gelegt werden. Über einen versetzten Klappmechanismus werden die Kugeln dann einzeln auf das Spielfeld geworfen. Dieser Mechanismus ist in Abbildung 2 gut zu erkennen.

5) Spielsteuerung: Für die Interaktion mit dem Spieler sind seitlich zwei Taster angebracht. Ein Taster wird betätigt, wenn der Spieler seinen Zug beendet hat und der Roboter an der Reihe ist. Der zweite Taster dreht den Spieltisch um 90 Grad. Dadurch kann der Spieler die Kugeln auch unter Felder legen, die sonst durch den Farbsensor oder den Kugelauswurf verdeckt wären.

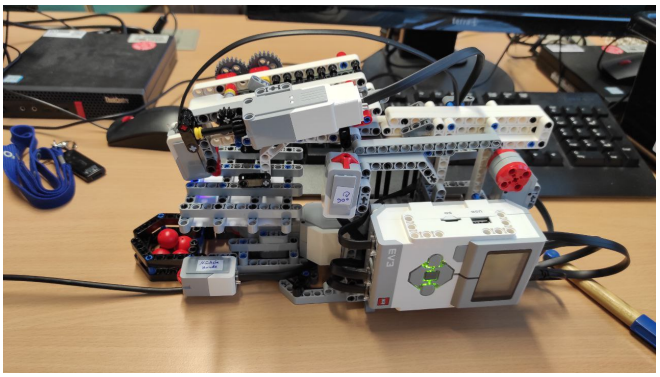


Abbildung 3. Seitenansicht des fertigen Roboters

B. Software

Die Software des T3R besteht aus einem Hauptskript und neun verschiedenen Funktionen. Um Probleme mit der Verbindung zwischen dem Computer und dem LEGO-Programmierstein während der Entwicklung des Roboters zu vermeiden, wurde das Zurücksetzen der Verbindung ein Teil der Initialisierung. Wenn ein Spieler den Drucktaster betätigt, der das Ende seines Zuges anzeigt, werden die Motoren für die Steuerung des Arms und des Tellers zurückgesetzt.

Im Anschluss wird eine Funktion zum Einlesen des Spielfeldes aufgerufen. Diese Funktion steuert den Arm und den Teller so an, dass der Farbsensor jede Position auf dem Spielfeld erreichen kann. Dazu werden erneut zwei Funktionen verwendet, um den Quelltext möglichst kurz zu halten. Die ausgelesenen Werte werden anschließend in einer 3×3-Matrix gespeichert. Die vom Farbsensor verwendeten Werte werden dabei einfach übernommen. Dabei wird der Wert 5 für Rot und der Wert 3 für Grün verwendet.

Damit der Roboter weiß, wie er spielen muss, um zu gewinnen, müssen die Spielregeln von Tic Tic Toe in eine Programm umgesetzt werden. Glücklicherweise gibt es dafür bereits einen Algorithmus [1], so dass dieser nur noch für MATLAB angepasst werden musste. Der hier verwendete Min-Max-Algorithmus ist ein rekursiver Algorithmus. Aus diesem Grund ist der in Abbildung 4 dargestellte Ablauf nicht korrekt. Der Min-Max-Algorithmus ermittelt alle möglichen Spielausgänge und entscheidet sich aufgrund dieser Analyse für eine von vier Varianten. Entweder hat einer der beiden Spieler gewonnen; wenn dies der Roboter ist, dreht er das Spielfeld dreimal. Bei einem Unentschieden piept der Tic Tac Toe Roboter dreimal. Tritt keiner dieser Fälle ein, wird die nächste Kugel des Roboters platziert. Dazu wird die vom Min-Max-Algorithmus ermittelte Position an eine Funktion übergeben, die diesen Schritt ausführt.

IV. Ergebnisdiskussion

Das Ergebnis des zweiwöchigen Seminars ist ein gut funktionierender Roboter. Dieser ist in der Lage, gegen einen Menschen Tic Tac Toe zu spielen.

Während der Konstruktion traten naturgemäß verschiedene Schwierigkeiten auf, die es zu überwinden galt. Einige konnten gelöst werden, andere bleiben offen.

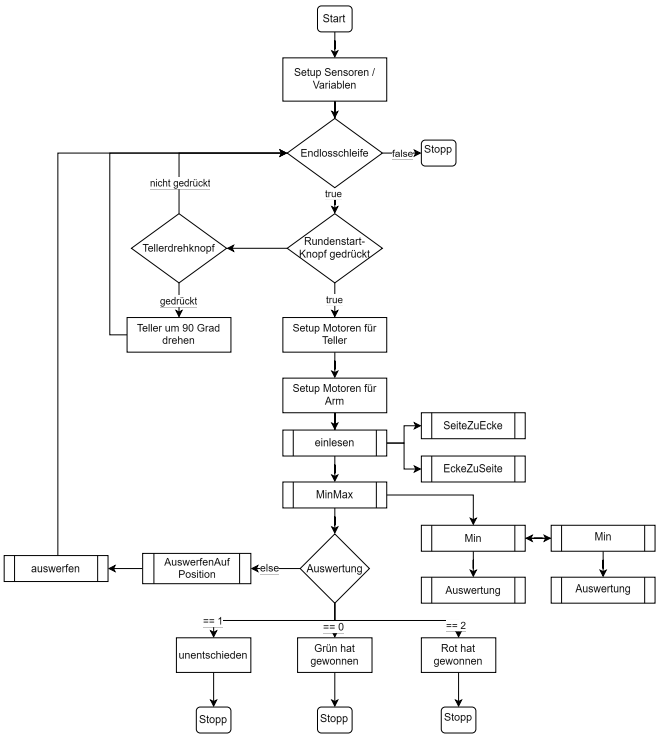


Abbildung 4. Programmablaufplan

1) Motoren: Die Motoren sind eine der wichtigsten Komponenten des T3R. Umso ärgerlicher war es, dass es immer wieder zu Problemen mit der Schrittgenauigkeit der Motoren kam. Dies führte letzten Endes zu längeren Wartezeiten beim Einlesen des Spielfeldes, aber auch zu einer höheren Fehlerquote beim Auswerfen der Kugeln. Um dieses Problem zu beheben, wurden zwei Maßnahmen ergriffen: Zum einen wurde die Bremse aktiviert, zum anderen wurden die Motoren vor dem Start eines Unterprogramms immer zurückgesetzt. Dies führte zu besseren Ergebnissen, jedoch blieb eine Restungenauigkeit der Motoren bis zum Schluss bestehen.

2) Farbsensor: Am Farbsensor traten während der Entwicklung immer wieder Zuverlässigkeitsprobleme auf. Häufig handelte es sich dabei um die Erkennung leerer Felder oder um die Erkennung von Kugeln, die nicht optimal unter dem Sensor positioniert waren. Das zweite Problem könnte nur durch eine Verbesserung der Genauigkeit der Platte gelöst werden. Das erste Problem konnte gelöst werden, indem ein weißes Blatt Papier auf den Boden des Spielfeldes gelegt wurde. Die Farbe Weiß wurde eindeutig erkannt; weitere größere Probleme traten danach nicht mehr auf.

Ein anderes Problem ergab sich aus der Höhe des Farbsensors. Wird der Sensor zu hoch über dem Spielfeld angebracht, sinkt die Anzahl der korrekt erkannten Kugeln drastisch. Wird der Sensor zu weit unten angebracht, kann es zu einem Kontakt zwischen Spielfeld und Sensor kommen. Dadurch werden die Bewegungen des Arms und der Platte ungenau und die Erkennung beziehungsweise das Setzen der Kugeln funktioniert nicht mehr korrekt oder dauert deutlich länger. Ein Beispiel für eine sehr schnelle und gute Farberkennung zeigt ein Zwischenstand des Projektes auf Instagram [2]. Um

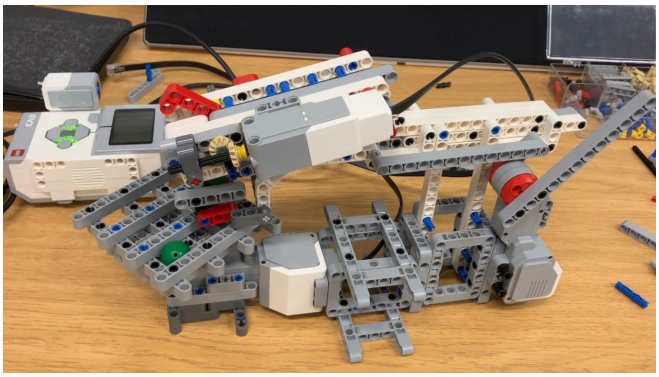


Abbildung 5. Alte Konstruktion des Kugelauswurfs. Mehrer Kugeln können gleichzeitig den Auswurf verlassen

eine möglichst gute geeignete Höhe zu finden, wurden an der Lauffläche zwischen Arm und Stützkonstruktion des Roboters zwei kleine verschiebbare Stifte angebracht, mit denen die Höhe variiert werden kann.

3) Kugelaufwurf: Bei der Konstruktion des Kugelauswurfes trat das Problem auf, dass zwei Kugeln gleichzeitig den Auswurf verließen. Versuche, dieses Problem durch eine schnellere Ansteuerung der Motoren zu lösen, schlugen fehl. Der Grund dafür war, dass nur ein L-Stück verwendet wurde, um die Kugeln am Verlassen des Kugellagers zu hindern. Dieses rote L-Stück ist in Abbildung 5 sehr gut zu erkennen. Die Konstruktion wurde ersetzt, so dass ein Auswerfen von mehreren Kugeln bereits mechanisch unmöglich war (siehe Abbildung 2). Am Ende bestand jedoch noch immer das Problem, dass sich manchmal Kugeln im Auswurfkanal verklemmten, da sie Unebenheiten auf der Kugeloberfläche aufwiesen. Komplett runde Kugeln wären hierfür eine geeignete Lösungsmöglichkeit.

V. Zusammenfassung und Fazit

Ziel des Projekts war es, einen Roboter zu bauen, gegen den ein Mensch Tic Tac Toe spielen kann. Dieses Ziel wurde erreicht. Wie bereits in der Ergebnisdiskussion erwähnt, gibt es noch viele Möglichkeiten, den Roboter weiterzuentwickeln. So wäre z. B. ein Programm denkbar, das erkennt, ob ein Spieler die Position einer Kugel nachträglich verändert hat. Dieses Programm könnte auch die Zuverlässigkeit des Kugelauswurfs überprüfen und daraus Maßnahmen für das Spiel ableiten.

Literaturverzeichnis

[1] Nopper, Tobias: *Implementierung des MinMax-Algorithmus für Tic Tac Toe | Wie spielen Computer?* 2020 https://www.youtube.com/watch?v=ODgPsXssUvk

[2] Gruppenvorstellung2024: *k. T.* 2024 https://www.instagram.com/gruppenvorstellung2024/reel/C3BOWyRNzey/

Der Tic-Tac-Toe-Robo

Karsten Schulz, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In den letzten Jahren etablierten sich viele Spiele, die man gegen einen Computer spielen kann. Der hier entwickelte Tic-Tac-Toe-Robo stellt eine weitere Möglichkeit dar. Unter der Verwendung von LEGO® zur Konstruktion und LEGO® Mindstorms® EV3 zur Ansteuerung wird eine weitere Umsetzungsvariante dargestellt. Durch die Programmierung mit MATLAB ergaben sich weitere vielfältige Optionen den Spielverlauf gegen den Robo-Gegenspieler interessanter zu gestalten. Die vorliegende Arbeit zeigt zunächst die bekannteste Standardumsetzung des Tic-Tac-Toe-Spiels.**

***Schlagwörter*—Farberkennung, Min-Max-Algorithmus, Roboter, Strategiespiel, Tic-Tac-Toe**

I. Einleitung

DIE Geschichte des Tic-Tac-Toe-Spiels ist in seiner Grundidee bereits auf die Zeit um 1300 v. Chr. zurückzuführen. Zwar war das Spielfeld zu dieser Zeit noch anders aufgebaut, dennoch sind die einfachen Spielregeln bis heute gleich geblieben. Da man besonders gerne Kieselsteine als Spielfiguren im Mittelalter nahm, etablierte sich der Name „Drei Kieselsteine gleichzeitig". Das Tic-Tac-Toe-Spiel, wie wir es heutzutage kennen, entstand Mitte des 19. Jahrhunderts. Durch die einfachen Spielstrukturen entstand im Jahr 1952 das erste Tic-Tac-Toe-Computerspiel mit dem Namen „OXO". Es war eines der ersten Computerspiele, bei dem der Mensch und der Computer gegeneinander spielten [1].
Nach diesem Prinzip funktioniert auch der Tic-Tac-Toe-Robo. Er funktionierte nicht über einen Bildschirm, sondern als Roboter. Er zeichnet nicht wie beim klassischem Tic-Tac-Toe das Spielfeld und seine Spielzeichen, wie es mit Zeichenroboter umgesetzt wird. Er besteht aus einem festen LEGO-Spielfeld und erkennt bzw. setzt selbstständig Spielsteine. Abbildung 1 zeigt diese vollständig verwirklichte Idee. Diese Umsetzung wird im folgenden erklärt.

II. Vorbetrachtungen

Als Spielsteine für den Roboter bieten verschiedenfarbige Kugeln mehrere Vorteile. Zu einem können sie vom LEGO-Farbsensor erkannt werden. Dadurch sollte der Farbsensor in der später folgenden Konstruktion das Spielfeld einlesen. Bei den ersten Farberkennungsversuchen wurde festgestellt, dass manche Farben nicht immer korrekt vom Sensor erkannt wurden. Dies war durch die unterschiedlichen Distanzen zwischen Sensor und Kugel, Umgebungslicht und der leicht glänzenden Oberfläche der Kugel begründet. Am sichersten erkannte der Sensor die Farben rot und grün. Aus diesem Grund wurden sie die Spielsteinfarben, wobei die grünen

DOI: 10.24352/UB.OVGU-2024-027

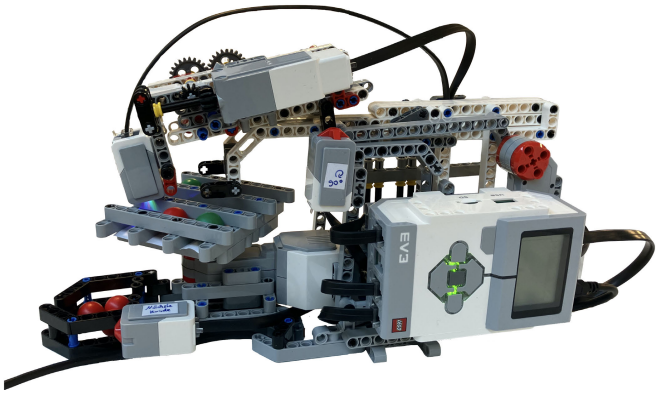


Abbildung 1. Fertiger Tic-Tac-Toe-Robo beim Einlesen des Spielfeldes

Kugeln vom Roboter und die roten vom Spieler gesetzt werden. Ein weiterer Vorteil der Kugeln zeigte sich in ihrer Form, sodass man sie ins Spielfeld fallen lassen konnte und sie die richtige Position im Feld einnehmen. Der größte Vorteil liegt jedoch im Prinzip des Auswerfens der Kugeln, da kein Greifarm benötigt wird, der die Spielsteine legen müsste.
Es war lediglich eine Lösung zu überlegen, wie man das Einlese- und Auswurfprinzip der Kugeln miteinander verknüpft und der Roboter dabei gewinnorientiert spielt. Dies realisierte der Min-Max-Algorithmus.

III. Umsetzung

Bei der Umsetzung wurde der Roboter in einzelnen Schritten aufgebaut, in denen immer neue Teilfunktionen integriert wurden. In diesem Zusammenhang wurden kleine Teilprogramme zur Ansteuerung entwickelt, die im finalen Schritt auf den gesamten Roboter angepasst wurden. Im folgenden wird der Aufbau in seinen einzelnen Schritten beschrieben. Dabei wird zwischen Konstruktion und Programmierung unterschieden.

A. Konstruktion

Die Grundvoraussetzung für den Roboter war das Spielfeld, in dem alle neun Kugeln gut saßen. Das erste Spielfeld war durch LEGO-Pins abgesteckt, wobei jede Kugel eine Fläche von ungefähr $1{,}2\,\mathrm{cm}^2$ Platz hatte. Beim Fallen der Kugeln stellte sicher heraus, dass die einzelnen Felder nicht ausreichend tief waren. Daraufhin wurde das Spielfeld erweitert und Querverstrebungen eingebaut, sodass für jede Kugel die Fläche auf $2{,}25\,\mathrm{cm}^2$ vergrößert wurde (siehe Abb. 2). Als nächstes sollte der Farbsensor über das Spielfeld zu jeder Kugel gelangen. Dies wurde mithilfe zweier Motoren realisiert. Ein Motor drehte das Spielfeld und der andere Motor ließ

einen Arm vor und zurück fahren, an dem der Farbsensor befestigt war (siehe Abb. 2). Durch die Kombination von Spielfeld drehen und vor- bzw. zurückfahren des Sensors, ist es möglich jedes Einzelfeld im Spielfeld zu erreichen.

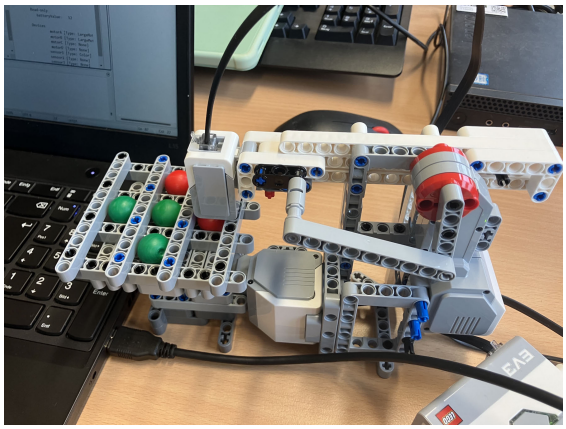


Abbildung 2. Farbsensor bestimmt Kugelfarben des gesamten Spielfelds

Nach der erfolgreichen Umsetzung des Einlesevorgangs erfolgte die Realisierung des Kugelauswurfs. Dabei saß dieser leicht geneigt auf dem Arm, an dem vorher der Farbsensor befestigt war (siehe Abb. 3). Die Neigung war notwendig, damit die nachfolgenden Kugeln aufrutschen konnten. Das Fassungsvermögen war auf fünf Kugeln ausgelegt, da dies die notwendige Anzahl an Kugeln pro Spiel ist. Der Kugelauswurf öffnete bzw. schloss sich durch ein Schließstück, das durch einen Motor betätigt wurde (siehe Abb. 3, Schließstück ist der rote Winkel).

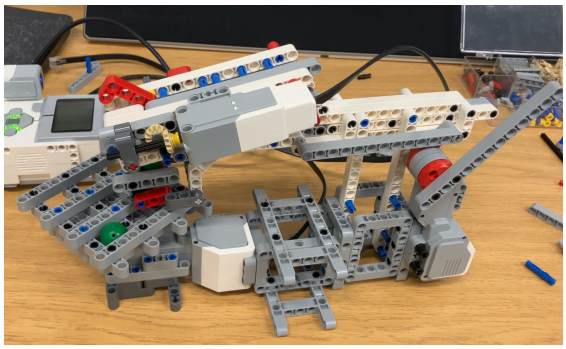


Abbildung 3. Kugelauswurf erreicht jede Spielfeldposition, Auswurfmechanismus durch Schließstück (rot) ohne Stopper

Da das Auswerfen einer Kugel notwendig war, wurde der Öffnungs- und Schließprozess schnell umgesetzt. Die hohe Schließgeschwindigkeit hatte zur Folge, dass sich eine Kugel verklemmte und sich der Winkel des Schließstücks verstellte. Dadurch fielen in den nachfolgenden Versuchen zwei Kugeln hinaus. Um dies zu vermeiden, wurde ein zusätzlicher Stopper verbaut. Dieser klinkte sich zwischen der ersten und zweiten Kugel ein, sobald sich das Schließstück öffnete. Dadurch stoppte die zweite Kugel, sodass lediglich die erste hinausfallen konnte. Der Stopper ist durch Zahnräder mit dem Schließstück verbunden. Dies hat zur Folge, dass der Stopper wieder hoch fuhr, sobald der Kugelauswurf geschlossen war. Im Anschluss rutschten die restlichen Kugeln wieder auf und der Mechanismus konnte erneut starten.
Das Gewicht von Motor und Kugelauswurf belastete den Arm stärker als der Farbsensor. Aus diesem Grund war eine Verstärkung des Arms, seiner Führung und die Unterkonstruktion des Roboters notwendig. Dabei wurde der Abstand zwischen Spielfeld und der Konstruktion, auf der die Armführung sitzt, erweitert. Dies lag an der Länge des Kugelauswurfs. Dadurch konnte er jede Position im Spielfeld erreichen.
Auch mit den Verstärkungen war ein Höhenunterschied am Arm festzustellen. Dies war in einer Toleranz zwischen Arm und Führung begründet, da der Arm etwas dünner als der Abstand zwischen der Führung war. Durch das Einbringen von zusätzlichen Stützen, die das vordere Ende der Führung erhöhten, konnte der Arm zu einer Parallele zum Spielfeld gerichtet werden (siehe Abb. 1). Am meisten profitierte der Farbsensor von dieser neuen Ausrichtung, der im Anschluss vor dem Kugelauswurf angebaut wurde. Somit ist die Distanz zwischen Kugel und Sensor immer gleich, sodass benannte Fehler bei der Farberkennung (erste Farberkennungsversuche in Abschnitt II) vermieden wurden.
Da es teilweise beim Treffen der Felder zu Fehlpositionen kam, wurde um die Fallstrecke eine Bande errichtet, die ein Wegrollen der Kugeln verhinderte. Im Weiteren ergänzten zwei Taster den Roboter. Der untere Taster diente als Start-Befehl für den Roboter zum Einlesen, Auswurfposition berechnen und Auswerfen der Kugeln. Der obere Taster ermöglichte das Spielfeld um 90° zu drehen (siehe Abb. 1). Dies war notwendig, da das vordere Einzelfeld vom Farbsensor verdeckt war. Dies erleichterte das Legen sowie das Herausnehmen der Kugeln. Im finalen Schritt wurde der Roboter durch ein Behältnis für die roten Kugel ergänzt und optisch angepasst, wie beispielsweise die Kabelführung.

B. Programmierung

Im folgenden wird der Aufbau des Programms erklärt. Die Erklärung wird vereinfacht durch den Programmablaufplan in Abbildung 4 dargestellt.
Alle eigenständigen Abläufe des Roboters wurden in einzelne Funktionen untergliedert, wie beispielsweise das Auswerfen einer Kugel. Diese sind im Hauptprogramm miteinander verknüpft. Als ersten Schritt wird im Hauptprogramm die Steuereinheit EV3 neu initialisiert. Dadurch konnten Fehlfunktionen mit der Winkelbegrenzung und Motorbremse vermieden werden. Im nächsten Schritt wurden für die Funktionen alle Winkel und Geschwindigkeiten der Motoren festgelegt. Das erfolgt über Variablen, da sie universell einsetzbar sind und Fehler reduzieren, bei den Anpassungen dieser Werte. Andernfalls wäre diese Anpassung in jeder einzelnen Funktion notwendig. Nachdem alle Variablen durch Werte bestimmt und der Farbsensor initialisiert ist, konnte eine 3×3-Matrix erstellt wurden. Dies entspricht der Einsmatrix, da die 1 im Späteren für ein nicht belegtes Feld steht. Die Matrix stellt das Spielfeld dar.

Durch das betätigen des unteren Tasters, wurde der Roboter gestartet. Folgend wurde jeder Motor gestoppt um Fehler zu vermeiden. Mögliche Fehler zeigten sich bei ersten Versuchen, indem ein Motor nicht seine Drehwinkelbegrenzung erreichte. Dabei versuchte der Motor sie weiterhin zu erreichen, auch wenn er mechanisch gehindert wird und stoppte somit nie. Da jedoch auf den Stillstand des Motors gewartet wird, führte dies zum Stillstand des gesamten Programms. Im Anschluss wird für die folgenden Spielrunden in allen Motoren die Motorbremse und der Drehwinkel begrenzt. Das war notwendig, da sich sonst die Ausgangsposition in jeder Spielrunde änderte.

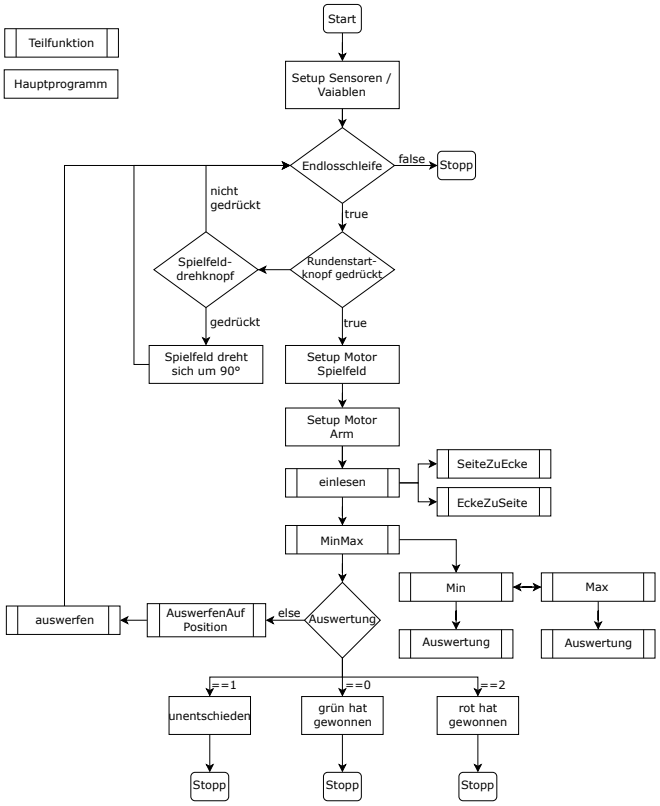


Abbildung 4. Programmablaufplan

Im Anschluss startet der Roboter mit dem Einleseprozess, der durch eine Funktion beschrieben ist. Sobald der Farbsensor eine Kugelfarbe im Spielfeld entdeckte, überschrieb er die erwähnte 3×3-Matrix mit der ermittelten Konstellation. Dabei nahm die Stelle in der Matrix den Farbsensorwert an, die der Kugelfarbe entspricht. Somit ist es möglich jeden Feldzustand durch eine Zahl zu beschrieben. Hierbei entsprach die Kugelfarbe grün dem Wert 3 und die rote Kugel dem Wert 5. Ein leeres Feld entsprach dem Farbsensorwert 1 (grau). Die Einlesefunktion wurde durch zwei weitere Funktionen ergänzt, die das Abfahren von einer Kanten- zu einer Eckposition (SeiteZuEcke) und von einer Eck- zu einer Kantenposition (EckeZuSeite) beinhalten.
Nach dem erstellen einer Matrix, die dem Spielstand entsprach, musste ein Algorithmus entstehen, der die Auswurfposition berechnet. Dabei war eine kurzzeitige Überlegung, alle möglichen Spielausgänge einzuprogrammieren. Nach einer kurzen Aufwandsabschätzung wurde diese Idee verworfen, da es $9! = 362880$ mögliche Spielverläufe gibt. Auch wenn die möglichen Spielverläufe durch Nutzung der Symmetrie und ein frühes Gewinnen reduziert werden, wäre die Zeit zu knapp gewesen, um alle möglichen Pfade zu programmieren. Eine weitere Möglichkeit bot das Addieren der einzelnen Zeilen, Spalten und Diagonalen, um im Anschluss einen Algorithmus dafür zu programmieren. Allerdings gab es bei allen probierten Zahlenkombinationen, bei mindestens zwei unterschiedlichen Szenarien die gleichen Ergebnisse. Ein Beispiel hierfür ist, mit den bekannten Zahlen 1, 3 und 5, die zwei Kombinationen $3 + 3 + 3 = 9$ (grün + grün + grün) und $1 + 3 + 5 = 9$ (leeres Feld + grün + rot). Nach einer Recherche bietet sich der Ansatz des Min-Max-Algorithmus an. Zur detaillierten Einarbeitung waren zwei Videos von Tobias Nopper [2],[3] hilfreich.

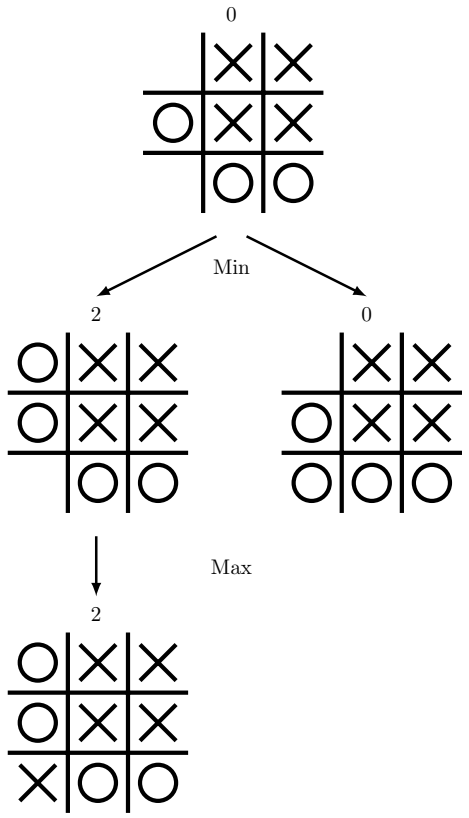


Abbildung 5. Min-Max-Algorithmus anhand eines Beispiels

Der Min-Max-Algorithmus wird mithilfe zweier rekursiver Funktionen umgesetzt. Wenn der Roboter gewinnt, wird dies durch ein Minimalwert beschrieben und wenn der Spieler gewinnt durch einen Maximalwert. Gewinnt keiner, wird ein Wert zwischen Minimum und Maximum festgelegt. Die Funktionen werden immer abwechselnd abgerufen und geben der anderen Funktion ihren Maximal- bzw. Minimalwert weiter. Als Beispiel dient Abbildung 5. Hierbei gibt es zwei Möglichkeiten. Zu einem kann „O" gewinnen, wobei sich der Minimalwert 0 ergibt. Zum anderen kann „X"

gewinnen, wobei sich der Maximalwert 2 ergibt. Hier wird eine Minimalauswertung für das obere Feld bestimmt, da „O" am Spielzug ist. Weil 0 kleiner als 2 ist, wird für das obere Feld die 0 weitergegeben. Somit sagt die 0 aus, dass „O" im nächsten Spielzug gewinnt. Über diese Methode werden alle möglichen Pfade berechnet und ihre Gewinnchancen bestimmt. Damit der Roboter seine Auswurfposition bestimmen kann, wird der Minimalwert abgefragt. Dabei gibt die Funktion alle Positionen aus, die einen Minimalwert anstreben. Hierbei wird der Roboter die letzte Position verwenden, da sie die vorherigen Positionen überschreibt. Aus diesem Grund unterscheidet der Algorithmus nicht wie schnell man den Sieg erreicht. Das bedeutet, dass der Roboter mit einem Spielzug gewinnen könnte, jedoch woanders hinsetzt und später gewinnt.
Nachdem eine günstige Position gefunden wurde, richtete sich der Kugelauswurf über diese Feldpostion aus. Diese Ausrichtung ist in einer eigenen Funktion beschrieben. Sie ordnet jeder berechneten Postion die Ausrichtung des Kugelauswufs zu. Wenn der Kugelauswurf die richtige Position erreicht hat, startet die Auswerfenfunktion. Im finalen Schritt wurde mithilfe der Stellmotoren die Ausgangsposition angestrebt.
Das Teilprogramm, dass durch den Tasters ausgelöst wurde ist damit abgeschlossen. Nun konnte der Spieler setzten oder den anderen (oberen) Taster für die Spielfeld-Drehung um 90° drücken. Durch das wiederholte drücken des unteren Taster startet das Teilprogramm erneut. Stellt der Roboter beim Einlesevorgang fest, dass er gewinnt, gibt er eine akustische Meldung ab und dreht dabei das Spielfeld drei mal. Wenn er verliert bzw. keiner gewinnt, gibt er andere akustische Meldungen aus.

IV. Ergebnisdiskussion / Verbesserungsvorschläge

Die Motoren, die für die Drehung des Spielfeldes und der Vor- bzw. Zurückbewegung des Armes zuständig sind, haben eine mechanische Toleranz. Dies hat zur Folge, dass das Spielfeld und der Arm für ein reibungslosen Ablauf perfekt zueinander ausgerichtet sein müsste. Die Fehleranfälligkeit des Farbsensors ist höher als die vom Kugelauswurf, da er eine Bande hat. Aus diesem Grund wäre eine Selbstkalibrierung beispielsweise durch farbigen Linien vorteilhaft. Alternativ könnte auch eine Bilderkennung mittels Kamera verwendet werden. Dabei wird kontinuierlich das ganze Feld erfasst. Durch diese Erkennung könnte der Roboter auch ein Schummeln des Spielers oder einen Fehler beim Auswurf der Kugeln feststellen.
Wenn der Roboter setzt und damit gewinnt, wäre es eine Verbesserung, wenn er direkt in den Gewinnmodus übergeht, ohne nochmal das Spielfeld einzulesen. Eine weitere Optimierung erfolgt, wenn der Roboter mit so wenig wie nötigen Spielzügen gewinnt. Im Spiel selbst traten keine weiteren Probleme auf. Eine letzte Optimierung ist durch die Nutzung von MATLAB möglich, da es seine eigene `Minimax` Funktion hat. Durch ihr bräuchte man lediglich diese eine Funktion anstelle von zwei rekursive Einzelfunktionen.

V. Zusammenfassung / Fazit

Die Erwartungen, einen Roboter zu bauen, der gegen einen menschlichen Spieler gewinnorientiert Tic-Tac-Toe spielt, wurde erfolgreich umgesetzt. Es wurden für den Roboter drei Motoren, ein Farbsensor, zwei Taster und als Steuereinheit der EV3 verwendet. Die Hauptbestandteile des Programms sind das Spielfeld einzulesen, auszuwerten und eine Position zum Setzten zu berechnen und schlussendlich genau in diese Position die Kugel auszuwerfen.
Für zukünftige Tic-Tac-Toe-Spiele, die über einen ähnlichen Aufbau verfügen, sind folgende Optimierungen empfehlenswert:

1) eine Selbstkalibrierung oder Bildauswertung mithilfe einer Kamera
2) so wenig Züge wie nötig zum Gewinnen
3) Erkennung falls der Spieler schummelt oder die Kugel falsch fällt
4) die eigene MATLAB-Funktion `Minimax` verwenden
5) Gewinnen erkennen ohne ein erneutes Einlesen

Auch für zukünftige Strategiespiele ist der Min-Max-Algorithmus zu empfehlen, da er sehr komplizierte Verzweigungen von Pfaden erleichtert sowie die Gewinnchancen und mögliche Setztposition bestimmt.

Literaturverzeichnis

[1] Marcus, M.: Tic Tac Toe-Geschichte: Drei In Folge Im Wandel Der Zeit, URL: *https://www.coolmathgames.com/de/blog/tic-tac-toe-geschichte-drei-in-folge-im-wandel-der-zeit* (Stand: 2024-02-16)

[2] Nopper, T.: Computer spielen: Wie eigentlich?: Der Min-Max-Algorithmus, URL: *https://www.youtube.com/watch?v=3ufcmCpKb6w* (Stand: 2024-02-07)

[3] Nopper, T.: Computer spielen: Wie eigentlich?: MinMax implementieren für Tic Tac Toe, URL: *https://www.youtube.com/watch?v=ODgPsXssUvk* (Stand: 2024-02-07)

LEGO-Gießroboter

Ihor Azovskyi, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Rahmen dieses Projekts wurde ein funktionsfähiger Gießroboter entwickelt, der mithilfe LEGO-Mindstorms-Sets gebaut wurde. Ziel war es, einen Roboter zu konstruieren, der das automatisierte Gießen von Getränken ermöglicht. Die Konstruktion umfasste drei Hauptkomponenten: das Fahrzeug, den Flaschenhalter mit Ausgleich und die Sensoren. Trotz technischer Schwierigkeiten wurde das Projekt erfolgreich abgeschlossen und demonstrierte die Möglichkeiten von LEGO Mindstorms im Bereich der Robotik und Automatisierung.**

***Schlagwörter*—Automatisierung, Gießroboter, LEGO Mindstorms, MATLAB-Programmierung, Sensoren, LEGO-Praktikum**

I. Einleitung

In der heutigen Gesellschaft ist der Fokus oft übermäßig auf Arbeit, Leistung, Produktivität und ähnliches gerichtet. Die ständige Beschäftigung und der hohe Stresslevel lassen den Menschen kaum noch Zeit für sich selbst und für die Ausübung von Aktivitäten, die Freude bereiten. Um diesem Trend entgegenzuwirken und das Leben zu erleichtern, sind zahlreiche Forschungsprojekte im Gange, die zur Entwicklung spezieller Geräte führen. Eine solche innovative Entwicklung wurde im Rahmen eines Projektseminars realisiert, bei dem ein Gießroboter mithilfe eines LEGO-Mindstorms-Sets und EV3-Kontrollsteins (Abbildung 1) konstruiert wurde. Dieser Roboter ist dazu bestimmt, den Alltag der Menschen zu erleichtern, indem er ihnen beim Trinken assistiert und somit den Prozess vereinfacht.

Abbildung 1. EV3-Kontrollstein [1]

II. Vorbetrachtungen

Im folgenden Abschnitt wird der zentrale Gedanke des Konzepts ausführlich erklärt, um ein umfassendes Verständnis für die Umsetzung zu vermitteln. Besonderes Augenmerk wird dabei auf die erforderlichen Bauteile gelegt, die für die Realisierung des Gießroboters notwendig sind.

DOI: 10.24352/UB.OVGU-2024-028

A. Fahrzeug

Zunächst war es erforderlich, ein Fahrzeug zu konstruieren, das in der Lage ist, die Flasche zu einem Glas zu transportieren. Dies stellte sich jedoch nicht als größeres Problem dar, da bereits ein Standardfahrzeug aus "Mindstorms Education" [2] zur Verfügung stand. Diese Wahl wurde getroffen, da dieses Fahrzeug grundlegende Funktionen wie Fahren, Drehen und die Verfolgung des Winkels aufweist. Für diese Aufgabe werden zwei große Motoren, zwei Ultraschallsensoren, einen Tastsensor und einen Gyrosensor benötigt, die auf der Abbildung 2 betrachtet werden können. Diese Komponenten sind entscheidend für die Funktionalität des Fahrzeugs und ermöglichen es, weitere Aufgaben wie das Erfassen von Hindernissen mittels Ultraschallsensoren, das Erkennen von Berührungen durch den Tastsensor und die präzise Ausrichtung mithilfe des Gyrosensors auszuführen. Durch die Kombination dieser Sensoren und Motoren ist das Fahrzeug in der Lage, die Flasche sicher zum Ziel zu transportieren und dabei Hindernisse zu umgehen oder ihnen auszuweichen.

B. Flaschenhalter und Ausgleich

Die zweite Aufgabe bestand darin, einen Flaschenhalter und einen entsprechenden Ausgleich zu konstruieren. Eine Herausforderung bestand zunächst darin, die optimale Position für den Flaschenhalter zu finden, um sicherzustellen, dass die Flasche sich auf der idealen Höhe befindet. Hierfür wurden verschiedene Entwürfe ausprobiert, und schließlich wurde ein Design ausgewählt, das es ermöglicht, den Winkel der Flasche einfach anzupassen. Der Ausgleich wurde bewusst weit nach hinten verlagert, um eine maximale Effizienz zu gewährleisten und das Gleichgewicht des Fahrzeugs während des Transports zu erhalten.

Abbildung 2. LEGO-Mindstorms-Komponenten [2-5]

III. Umsetzung

Im folgenden Abschnitt wird die Umsetzung des Gießroboters ausführlich diskutiert. Dabei wird zunächst eine Beschreibung des Aufbaus des Roboters gegeben, gefolgt von einer detaillierten Funktionsbeschreibung.

A. Aufbau und Komponente

Der Gießroboter in Abbildungen 3 und 4 besteht aus drei Hauptkomponenten, die jeweils unterschiedliche Funktionen erfüllen. Der erste Teil ist der eigentliche Roboter, das Fahrzeug, das für die Fortbewegung des gesamten Systems verantwortlich ist. Dieser Roboter ist mit einem Gyrosensor ausgestattet, um während der Fahrt Abweichungen zu messen und gegebenenfalls die Richtung anzupassen.

Der zweite Teil umfasst den Flaschenhalter und den Ausgleich. Diese Konstruktion ist äußerst robust und verfügt über einen hinteren Ausgleich, um eine optimale Stabilität zu gewährleisten, da das Fahrzeug aufgrund äußerer Störungen von seinem geraden Weg abweichen kann. Es gibt auch die Möglichkeit, den Winkel des Flaschenhalters anzupassen, um Flaschen verschiedener Größen aufnehmen zu können. Zusätzlich ist ein Gummiband vorhanden, um die Flasche noch sicherer zu halten. Ein kleiner Motor im Flaschenhalter wird beim Stoppen des Roboters aktiviert, um die Flasche präzise zu öffnen. Dies geschieht durch die horizontale Bewegung des Motors, die durch zwei Zahnräder in eine vertikale Bewegung umgewandelt wird. Durch das Einstechen von Zahnstochern in die Folie wird die Flasche geöffnet.

Der dritte Teil besteht aus zwei Ultraschallsensoren und einem Tastsensor. Diese Sensoren dienen dazu, die Entfernung zu einem Glas zu messen, und der Tastsensor erfasst, ob der Roboter mit einem Glas kollidiert ist. Die Ultraschallsensoren sind so positioniert, dass sie eine ausreichende Distanz voneinander haben, um zwischen verschiedenen Größen von Gläsern zu unterscheiden.

Das gesamte System ist über Kabel mit einem Laptop verbunden, der das Programm zur Steuerung des Roboters ausführt.

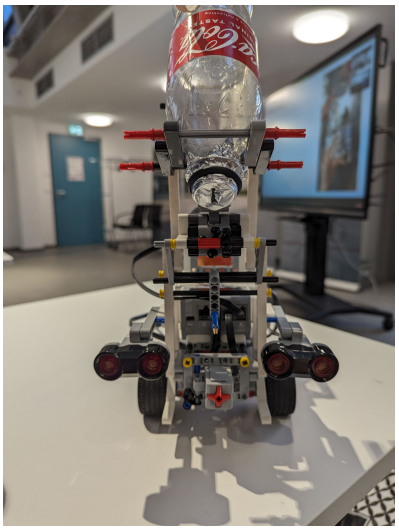


Abbildung 3. Roboter von vorn

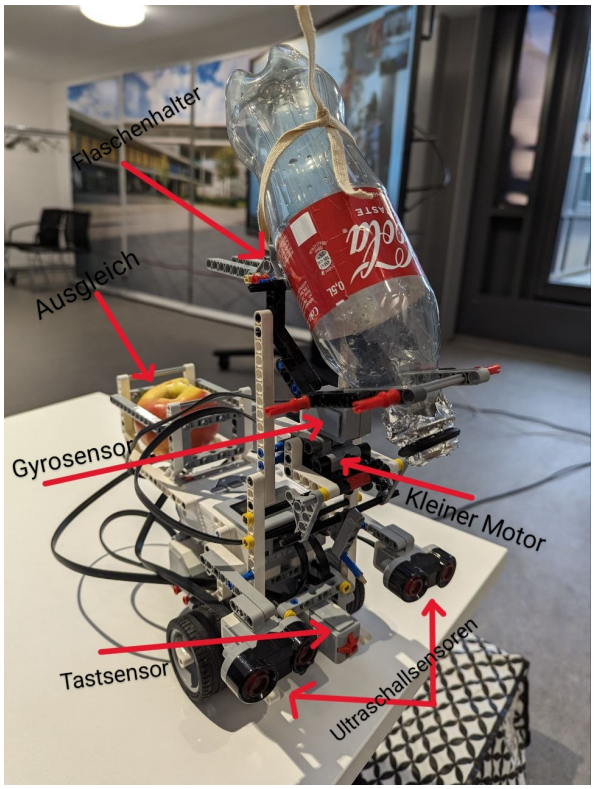


Abbildung 4. Aufbau

B. Funktionsweise

Zu Beginn des Programms wird das EV3-Modul initialisiert, falls dies noch nicht geschehen ist. Dies umfasst das Herstellen einer Verbindung zum Modul über USB und die Zuweisung der Motoren (B und C) sowie der Sensoren (1, 2, 3 und 4) für die Verwendung im Programm. Das sieht man im folgenden Programmierbeispiel:

```
if not(exist('brick', 'var'))
    brick = EV3();
    brick.connect('usb');
end
```

Die Motoren B und C werden entsprechend konfiguriert, um eine bestimmte Leistung, Geschwindigkeitsregelung und Begrenzungswerte für die Drehbewegungen festzulegen. Diese Einstellungen ermöglichen eine präzise Steuerung der Bewegungen des Roboters während des Gießvorgangs. Das sieht man im folgenden Programmierbeispiel:

```
motorB = brick.motorB;
motorC = brick.motorC;

motorB.setProperties('power',40,'limitValue',0);
motorC.setProperties('power',40,'limitValue',0);
```

Die Funktionsweise des Gießroboters wird im folgenden Programmablaufplan (PAP) veranschaulicht, der in Abbildung 5 grafisch dargestellt ist.

In der Hauptschleife des Programms wird kontinuierlich überprüft, ob der Stoppsensor aktiviert ist. Solange dieser nicht aktiviert ist, überwacht der Roboter die Sensoren 1 und 4, um Hindernisse zu erkennen und entsprechend zu reagieren.

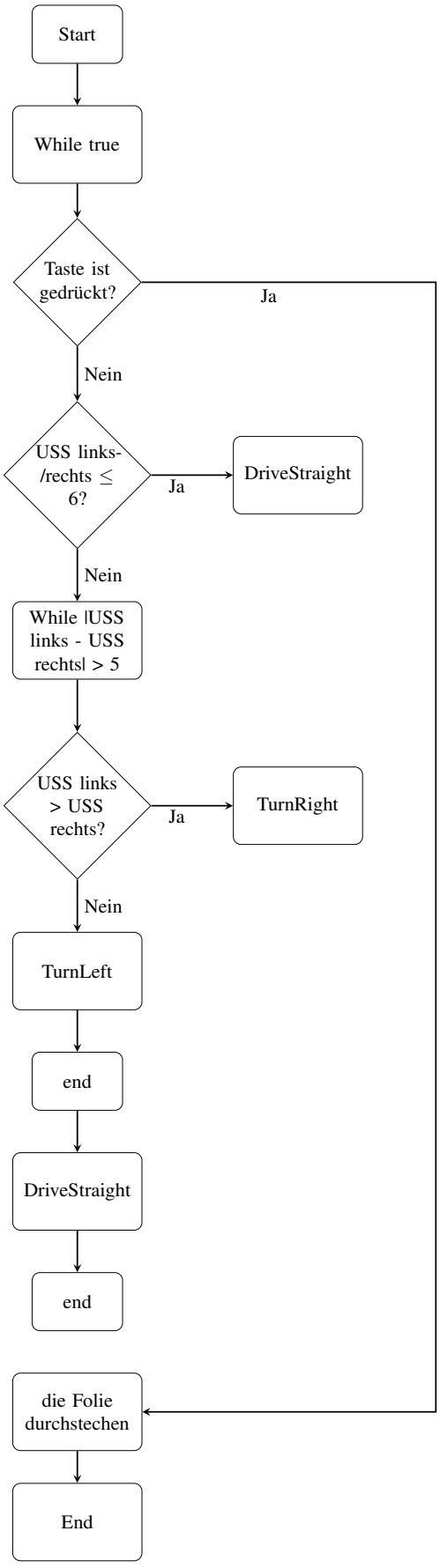


Abbildung 5. Programmablaufplan

Wenn der Roboter feststellt, dass er von seinem Kurs abweicht, führt er eine Korrektur durch, indem er entweder nach links oder rechts lenkt, um den ursprünglichen Fahrweg beizubehalten. Dies geschieht unter Berücksichtigung der aktuellen Werte der Sensoren und der Gyro-Sensordaten, um eine präzise Steuerung zu gewährleisten.

Sobald der Stoppsensor aktiviert wird, stoppt der Roboter den Gießvorgang und führt gegebenenfalls weitere Bewegungen aus, um die Flasche zu positionieren oder einen Signalton abzugeben.

Am Ende des Programms führt der Roboter einige zusätzliche Bewegungen aus, wie das Hin- und Herbewegen des Motors A, um sicherzustellen, dass die Flüssigkeit vollständig ausgegossen wurde.

Zusätzlich zu den Hauptfunktionen des Programms gibt es spezielle Funktionen wie `"Turn"` und `"driveStraight"`, die verwendet werden, um den Roboter zu lenken und geradeaus zu fahren, basierend auf den Werten der Sensoren und der aktuellen Gyro-Sensordaten.

Insgesamt arbeitet der Gießroboter kontinuierlich daran, die Flasche sicher zu positionieren und die Flüssigkeit präzise auszugießen, während er Hindernisse erkennt und umfährt, um einen reibungslosen Ablauf des Gießvorgangs zu gewährleisten.

IV. Ergebnisdiskussion

Als Ergebnis entstand ein funktionsfähiger Roboter, der in der Lage ist, eine Reihe von Befehlen auszuführen. Die größte Herausforderung bestand darin, die Ultraschallsensoren so zu programmieren, dass sie das Zielglas zuverlässig erkennen können und nach der Identifizierung nicht mehr von anderen Objekten gestört werden. Ein weiteres Problem bestand darin, dass als die Entfernung zwischen dem Roboter und dem Glas sehr gering wird, etwa 5 cm, dann das dazu führte, dass die Sensoren möglicherweise ein weiteres Objekt erkannten. Aus diesem Grund wurde ein Tastsensor eingebaut, der bei mechanischem Kontakt das Fahrzeug stoppt, um Kollisionen zu vermeiden. Es war auch wichtig sicherzustellen, dass das Glas selbst ausreichend schwer ist, damit der Tastsensor beim Stoßen gedrückt wird und nicht das Glas weggeschoben wird.

V. Zusammenfassung und Fazit

Die Entwicklung eines Gießroboters mithilfe von LEGO-Mindstorms-Sets war ein aufschlussreiches Projekt, das verschiedene technische Herausforderungen mit sich brachte. Ziel war es, einen Roboter zu konstruieren, der in der Lage ist, das Gießen von Getränken zu automatisieren. Der Roboter wurde in drei Hauptkomponenten unterteilt: das Fahrzeug, den Flaschenhalter mit Ausgleich und die Sensoren. Während des Projekts wurden zahlreiche Konstruktions- und Programmieraspekte berücksichtigt. Die Programmierung der Ultraschallsensoren stellte eine besondere Herausforderung dar. Die Konstruktion des Flaschenhalters und des Ausgleichs erforderte sorgfältige Überlegungen zur Stabilität und Anpassungsfähigkeit an verschiedene Flaschengrößen. Die Integration eines Motors zur Öffnung der Flasche erwies sich ebenfalls als wichtiger Schritt, um den Gießvorgang zu automatisieren.

Insgesamt war das Projekt eine erfolgreiche Demonstration der Fähigkeiten von LEGO Mindstorms im Bereich der Robotik und Automatisierung. Für künftige Optimierungen besteht die Möglichkeit, die Konstruktion weiter zu verbessern. Ein Ansatz wäre beispielsweise, den Flaschenhalter so zu gestalten, dass er biegbar ist. Dadurch könnte dieselbe Flasche mehrmals verwendet werden, was die Effizienz des Geräts erhöhen würde.

LITERATURVERZEICHNIS

1) **Amazon:** EV3 Intelligent brick https://www.amazon.de/-/en/45500-EV3-Intelligent-brick/dp/B00E1P3ACK
2) **Amazon:** EV3 Servo Motor Large https://www.amazon.de/-/en/MINDSTORMS-Education-Servo-Motor-Large/dp/B00E1QDP4W
3) **Amazon:** EV3 Ultraschallsensor https://www.amazon.de/-/en/7645504/dp/B00E1PTRAE
4) **Amazon:** EV3 Touch Sensor https://www.amazon.de/-/en/45507-EV3-Touch-Sensor/dp/B00E1PRQ48
5) **Amazon:** EV3 Gyro Sensor https://www.amazon.de/-/en/45505-EV3-Gyro-Sensor/dp/B00E1QLPXK

Gießroboter

Oleksii Bidnyi, ETIT
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Die LEGO Mindstorms Serie, ein gewaltiger Sprung in der Lernrobotertechnik, ermöglicht es Enthusiasten seit langem, eine Reihe genialer automatisierter Konstruktionen zu bauen. Unter diesen Innovationen ist der Gießroboter ein Beispiel für die Verschmelzung von praktischem Nutzen und Freude an der Kreation. Dieses Roboterwunder, das aus dem vielseitigen Mindstorms-Bausatz entwickelt wurde, ist mit einer Reihe von Sensoren und programmierbaren Bausteinen ausgestattet. Ausgelöst durch einen ausgeklügelten Algorithmus steuert er ein bestimmtes Glas mit Präzision an. Sobald es positioniert ist, aktiviert es einen Mechanismus, um Wasser auszugießen und ein erfrischendes Getränk zu servieren. Dieses Projekt zeigt nicht nur das Potenzial von Mindstorms, sondern ist auch ein Beweis für den menschlichen Erfindungsreichtum bei der Nutzung von Technologie für alltägliche Zwecke.**

***Schlagwörter*—Automatisierung, Gießroboter, LEGO Mindstorms, MATLAB-Programmierung, Sensoren, LEGO Praktikum.**

I. Einleitung

In einer Zeit, in der die menschliche Arbeit immer mehr an Maschinen ausgelagert wird, ist der Gießroboter (siehe Abbildung 1) der Gipfel der komfortorientierten Innovation. Dieses Gerät, das sorgfältig aus einer Reihe von Zahnrädern, Sensoren und programmierbaren Elementen zusammengesetzt ist, wurde entwickelt, um die äußerst komplexe Aufgabe des Einfüllens von Wasser in ein Glas auszuführen – eine Aufgabe, von der man einst dachte, dass sie selbst von einem minimal motivierten Menschen bewältigt werden kann.

Mit klinischer Effizienz navigiert der Gießroboter zu seinem Ziel, führt den Guss mit Präzision aus und befreit so seine menschlichen Vorgesetzten von der Sisyphusarbeit, einen Krug zu heben. Diese Maschine arbeitet mit stoischer Gelassenheit, ein stiller Wächter, der sich um die Trinkbedürfnisse einer Spezies kümmert, die solch triviale körperliche Anstrengungen in den Bereich der Robotik verlagert. Der Gießroboter ist ein Monument menschlicher Genialität – oder vielleicht ein Spiegel unserer Trägheit, der ein Zeitalter widerspiegelt, in dem selbst der einfache Akt des Ausgießens von Wasser als zu mühsam für die empfindlichen Sinne des Menschen angesehen wird.

II. Vorbetrachtungen

A. Funktionsfähigkeiten des Gießroboters

Der Roboter kann die Entfernung zum Ziel bestimmen und feststellen, in welcher Richtung sich das Ziel relativ zu ihm befindet. Der Roboter überwacht diese Indikatoren ständig, um sicherzustellen, dass er sich auf dem richtigen Weg befindet. Der Roboter erkennt auch, wenn er sich einem Glas nähert, und sobald er sich diesem nähert, wird der Ausgießmechanismus aktiviert.

DOI: 10.24352/UB.OVGU-2024-029

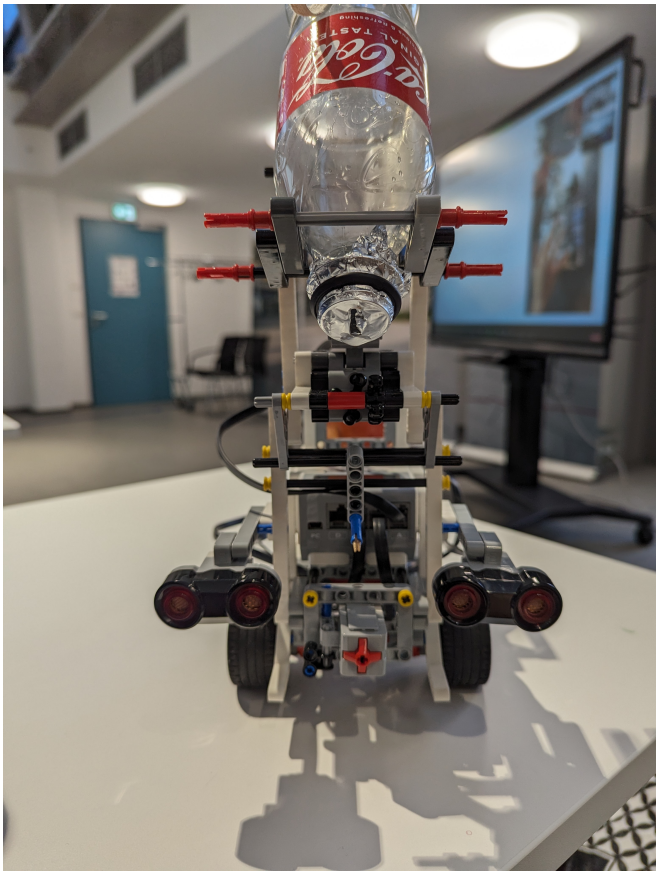


Abbildung 1. Gießroboter, Känozoikum, 2024 AD

B. Anwendung

Der Ausgießroboter ist ein vielseitiges Werkzeug mit einer Vielzahl von Anwendungsmöglichkeiten in unterschiedlichen Bereichen. Zu Hause kann er vor allem für Menschen mit eingeschränkter Mobilität eine unschätzbare Hilfe sein, indem er einfache Aufgaben wie das Einfüllen eines Glases Wasser oder die Zubereitung von Getränken übernimmt und so zu mehr Unabhängigkeit und Lebensqualität beiträgt.

In der belebten Atmosphäre von Bars und Restaurants könnte dieser Roboter den Getränkeservice revolutionieren. Er kann die monotone Aufgabe des Einschenkens von Getränken übernehmen, so dass das Personal seine Aufmerksamkeit auf kompliziertere Service-Details und die Interaktion mit den Kunden richten kann, was das Gesamterlebnis im Restaurant verbessern könnte.

Auch Einrichtungen des Gesundheitswesens und der Altenpflege könnten von der Einführung des Gießroboters profitieren. Er würde dafür sorgen, dass Patienten und Bewohner ständig

Zugang zu Wasser haben, was die Flüssigkeitszufuhr und das Wohlbefinden fördert, während sich das Pflegepersonal auf wichtigere Aspekte der Pflege konzentrieren kann.

Büros und Geschäftsräume könnten durch die Integration des Roboters in ihre Ausstattung einen Anstieg der Effizienz und der Mitarbeiterzufriedenheit verzeichnen. Pausenräume und Besprechungsräume, die mit einem solchen Gerät ausgestattet sind, würden den Mitarbeitern den Komfort von Erfrischungen bieten, ohne ihren Arbeitsablauf zu unterbrechen.

Darüber hinaus eignet sich der Roboter aufgrund seiner Präzision für spezielle Umgebungen, in denen eine sorgfältige Mengenkontrolle und die Vermeidung von Verunreinigungen von entscheidender Bedeutung sind, wie z. B. in wissenschaftlichen Labors oder bei sensiblen Fertigungsprozessen. Die präzisen Gießfähigkeiten des Roboters könnten das Risiko menschlicher Fehler bei diesen kritischen Aufgaben erheblich verringern.

Die Anpassungsfähigkeit des Gießroboters macht ihn zu einer wertvollen Ergänzung für jede Umgebung, indem er die Funktionalität verbessert und einen Hauch von modernem technologischem Komfort in die täglichen Routinen einbringt.

C. Klimaschutz

Der Gießroboter wurde unter dem Aspekt der Klimaneutralität entwickelt und spiegelt das Engagement für Nachhaltigkeit wider. Er wird elektrisch betrieben und kann aus erneuerbaren Energiequellen wie Solar- oder Windenergie gespeist werden, was mit dem Ethos eines geringen CO2-Fußabdrucks in Einklang steht.

Darüber hinaus verringert die Effizienz des Roboters bei der Ausführung seiner Aufgaben die Wahrscheinlichkeit einer Wasserverschwendung, was für einen verantwortungsvollen Umgang mit den Wasserressourcen von entscheidender Bedeutung ist.

Durch die Kombination der Prinzipien von Energieeffizienz, nachhaltiger Materialnutzung und Abfallvermeidung ist der Gießroboter ein Beispiel für klimabewusste Innovation im Bereich der Bildungs- und Funktionsrobotik.

III. Arbeitsablauf

A. Struktur des Gießroboters

Das architektonische Design des Gießroboters besteht aus einer Reihe von elektromechanischen Komponenten, die systematisch angeordnet sind, um einen autonomen Betrieb zu ermöglichen (siehe Abbildung 2). Das Teilsystem für die Fortbewegung wird von einem Elektromotoren-Duo (siehe Abbildung 3) angetrieben, das die nötige kinetische Energie für die Translationsbewegung über ebene Flächen liefert.

Die Stabilisierung und die Winkelausrichtung werden von einem eingebauten Gyrosensor (siehe Abbildung 4) gesteuert, der die genaue Einhaltung der programmierten Flugbahn gewährleistet. Dieser Sensor erkennt jede Abweichung von der horizontalen Achse und ermöglicht Korrekturen in Echtzeit, um die Position des Roboters aufrechtzuerhalten. Die

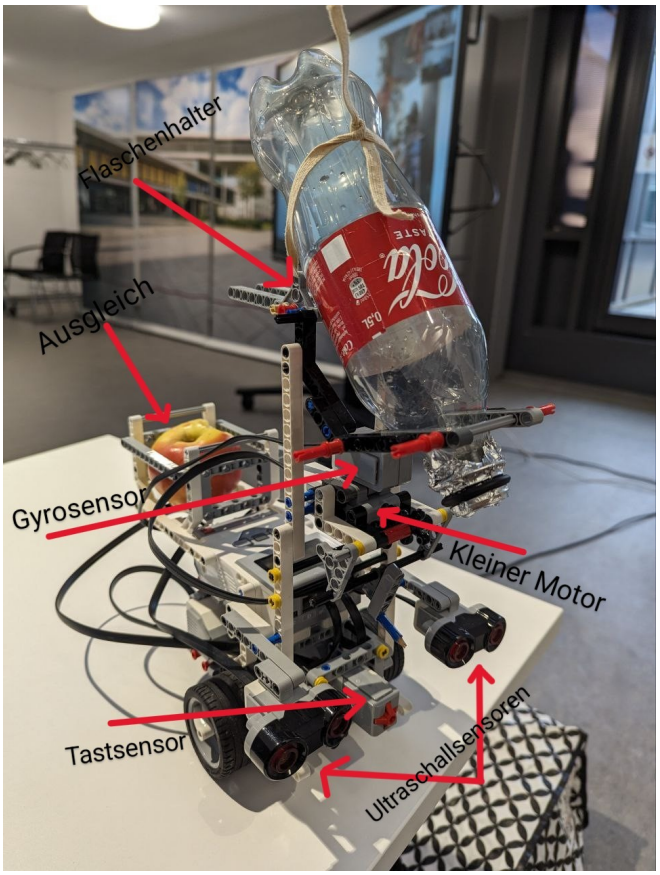


Abbildung 2. Struktur

Abbildung 3. Elektromotor

Abbildung 4. Gyrosensor

Annäherung an Objekte in der Umgebung wird durch ein Paar Ultraschallsensoren (siehe Abbildung 5) erreicht. Diese Ultraschallsensoren arbeiten nach dem Prinzip der Echo-Ortung, indem sie Schallimpulse aussenden und das Intervall bis zur Rückkehr des Echos messen, um die Nähe zu potenziellen Hindernissen, insbesondere dem Zielbehälter für die Flüssigkeit,

Abbildung 5. Ultraschallsensor

zu berechnen.

Bei erfolgreicher Annäherung an das Ziel wird der Tastsensor (siehe Abbildung 6) eingeschaltet. Dieser Kontaktsensor dient als Endpunktbestätigung, die anzeigt, dass der Roboter die für die Flüssigkeitsabgabe erforderliche Nähe erreicht hat. Die Aktivierung des Sensors bedeutet, dass der mechanische Ablauf des Ausgießens beginnen kann.

Abbildung 6. Tastsensor

Der Flüssigkeitsausgabemechanismus ist durch eine umgedrehte Wasserflasche gekennzeichnet, deren Öffnung mit einer Folienmembran hermetisch verschlossen ist, um ein vorzeitiges Austreten des Wassers zu verhindern. Ein kleiner, in das System integrierter Elektromotor ist dafür verantwortlich, die Folie auf ein Signal der Steuereinheit hin zu durchstoßen und so die kontrollierte Abgabe von Wasser einzuleiten.

Im Wesentlichen verkörpert die strukturelle Zusammensetzung des Roboters ein ausgeklügeltes Zusammenspiel von sensomotorischen Elementen, von denen jedes eine bestimmte Rolle innerhalb des Betriebsparadigmas erfüllt und die so orchestriert sind, dass sie die Aufgabe des autonomen Auffindens eines Glases und der präzisen Ausführung des Ausgießens erfüllen. Alle Sensoren funktionieren als ein einziger Mechanismus, der von der EV3-Steuereinheit gesteuert wird (siehe Abbildung 7).

Abbildung 7. EV3-Steuereinheit

B. Betriebsalgorithmen

Der Betriebsalgorithmus (siehe Abbildung 8) des Gießroboters ist ein systematisches Verfahren, das die genaue

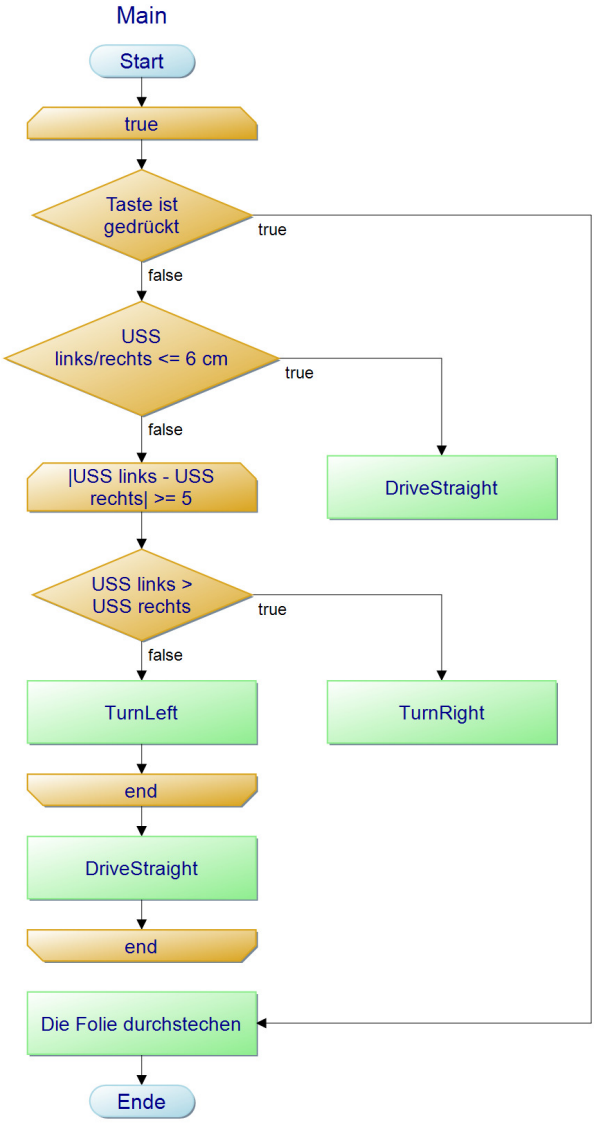


Abbildung 8. Allgemeiner Algorithmus

Ausführung der Aufgabe gewährleistet. Der Prozess beginnt damit, dass der Robot prüft, ob eine Tastsensor gedrückt wurde. Wenn der Knopf gedrückt ist, ist der Roboter bereits am Ziel. Danach beginnt der Roboter seine beiden seitlich angebrachten Ultraschallsensoren nutzt, um kontinuierlich den Abstand in Zentimetern zu Glass.

Der Algorithmus verarbeitet die Entfernungsdaten von beiden Ultraschallsensoren und vergleicht die Messwerte, um die Ausrichtung des Roboters relativ zum Glas zu bestimmen. Ein kürzerer Abstandswert von einem Sensor deutet darauf hin, dass sich der Roboter auf dieser Seite näher am Ziel befindet. Das Steuerungssystem des Roboters befiehlt dann eine leichte Drehung in Richtung des näheren Sensors, um sich auf das Glas auszurichten.

Der Roboter bewegt sich iterativ in kleinen Schritten vorwärts und hält dabei inne, um die Abstände erneut zu messen und zu vergleichen. Wenn der Unterschied zwischen den

Sensormesswerten weniger als 5 cm beträgt, schließt der Algorithmus daraus, dass der Roboter korrekt auf die Zielbahn ausgerichtet ist. Trotzdem nimmt der Roboter auf der Grundlage des Sensorfeedbacks weiterhin Mikroanpassungen an seinem Kurs vor, um diese Ausrichtung beizubehalten.

Gleichzeitig spielt der Gyrosensor eine entscheidende Rolle bei der Sicherstellung einer linearen und stabilen Bewegung des Roboters. Er erkennt jede Winkelabweichung von der beabsichtigten geraden Bahn, wie z. B. Driften oder Schräglage, und das Steuerungssystem gleicht diese Abweichungen entsprechend aus, um der Bewegungspfad des Roboters neu auszurichten.

Diese Schleife aus Erfassen, Bewegen und Korrigieren wird so lange fortgesetzt, bis der Tastsensor aktiviert wird und anzeigt, dass der Roboter das Glas erreicht hat. Daraufhin startet der Algorithmus die Ausgießsequenz. Das Signal des Geschmackssensors aktiviert einen kleinen Elektromotor, der einen Getriebemechanismus in Gang setzt, der mit einem Zahnstocher oder einem ähnlichen scharfen Gegenstand verbunden ist. Dieses Werkzeug durchstößt die Folie der Wasserflasche, so dass das Wasser in das darunter liegende Glas fließen kann.

IV. Ergebnisdiskussion

Die Analyse der Leistung des Gießroboters bestätigt die Präzision der sensorgesteuerten Navigation und der algorithmischen Steuerung. Die Ultraschallsensoren lenkten den Roboter effektiv, während der Gyrosensor für genaue Fahrwege sorgte. Der Tastsensor zeigte zuverlässig das Erreichen des Ziels an und ermöglichte so die erfolgreiche Aktivierung des Wasserspendermechanismus. Die Ergebnisse bestätigen die Fähigkeit des Roboters, Ausgießaufgaben effizient und genau zu automatisieren, und zeigen das Potenzial für ähnliche Roboteranwendungen im täglichen Leben auf.

V. Zusammenfassung und Fazit

Zusammenfassend lässt sich sagen, dass der Gießroboter einen bedeutenden Fortschritt im Bereich der Haushaltsrobotik darstellt, da er hochentwickelte Sensortechnologie mit fortschrittlichen Steuerungsalgorithmen kombiniert, um alltägliche Aufgaben autonom zu erledigen. Die erfolgreiche Ausführung seiner Gießfunktion veranschaulicht nicht nur die operative Effizienz des Roboters, sondern auch seinen potenziellen Nutzen in verschiedenen Bereichen, von der Unterstützung von Menschen mit Mobilitätseinschränkungen bis hin zur Verbesserung der Serviceeffizienz im Gastgewerbe. Die präzise Leistung und die Anpassungsfähigkeit des Roboters deuten auf eine Zukunft hin, in der sich eine solche Automatisierung nahtlos in das tägliche Leben integrieren lässt. Sie bietet Komfort und Unterstützung und ist gleichzeitig ein Beispiel für den innovativen Einsatz von Technologie im privaten und beruflichen Umfeld. Das Problem beim derzeitigen Stand der Roboterentwicklung ist, dass der Roboter nicht erkennen kann, was ein Glas ist und was nicht. Der Roboter weiß nicht genau, worauf er zusteuert, sondern hofft nur, dass es sich bei diesem Objekt um ein Glas handelt, so dass der Betriebsraum des Roboters die Abwesenheit anderer Objekte erfordert. Mit der Einführung moderner Technologie kann der Roboter jedoch erheblich verbessert werden. Zum Beispiel mit einer Kamera und KI.

Literatur

[1] Amazon Touch sensor: https://www.amazon.de/-/en/45507-EV3-Touch-Sensor/dp/B00E1PRQ48
[2] Amazon EV3: https://www.amazon.de/-/en/45500-EV3-Intelligent-brick/dp/B00E1P3ACK
[3] Amazon Ultrasonic sensor: https://www.amazon.de/-/en/7645504/dp/B00E1PTRAE
[4] Amazon Gyrosensor: https://www.amazon.de/-/en/45505-EV3-Gyro-Sensor/dp/B00E1QLPXK
[5] Amazon Large Electric Motor: https://www.amazon.de/-/en/MINDSTORMS-Education-Servo-Motor-Large/dp/B00E1QDP4W
[6] List of sensors: medium.com/kidstronics/lego-sensors-touch-n-color-edbe0f6642fhttps://www.amazon.de/-/en/45500-EV3-Intelligent-brick/dp/B00E1P3ACK

The Maze Escaper

Mohamed Ahmed, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Dieses Projekt an der Otto-von-Guericke-Universität in Verbindung mit dem Projektseminar Elektrotechnik/Informationstechnik soll die Bedeutung von autonomen intelligenten Autos in unserem Leben hervorheben, indem ein vereinfachtes Projekt vorgestellt wird. Das Auto verlässt sich auf die Erkennung von Farben und Entfernungen, um selbst zu fahren und dabei Hindernissen und Kollisionen auszuweichen.**

***Schlagwörter*—EV3, Farbesensor, LEGO-Mindstorms, Maze, Selbstfahrend, Ultraschalsensor**

I. Einleitung

DIE Wissenschaft hat in jüngster Zeit einen weiten Weg zurückgelegt. Mit der Entwicklung der Wissenschaft haben sich auch viele Luxusgüter vermehrt. Ein solches Gut ist das selbstfahrende Auto. Selbstfahrende Autos, wie in Abbildung 1 gezeigt wird [1], hatten neben dem Luxus noch weitere Vorteile. Sie schonen die Umwelt, indem sie die Kohlenstoffemissionen minimieren. Intelligente selbstfahrende Autos werden mit Strom betrieben. Auch aus verkehrstechnischer Sicht ist es von Vorteil, da es menschliche Fehler minimiert, die zu einem großen Verlust an Eigentum und Menschenleben führen. Außerdem sind sie in der Lage, miteinander zu kommunizieren, um Staus zu erkennen und zu umfahren.

Abbildung 1: selbstfahrend

II. Vorbetrachtungen

In diesem Teil des Artikels wird das Projekt vorgestellt und erläutert. Auch die wichtigsten verwendeten Teile werden erwähnt. LEGO Mindstorms ist eine von der LEGO-Gruppe entwickelte Robotik Plattform. Diese Plattform bietet eine einzigartige Erfahrung für alle Altersgruppen. Sie ermöglicht es dem Benutzer, seinen eigenen Roboter frei zu bauen und zu programmieren. Es werden LEGO-Komponenten, Sensoren, Motoren und programmierbare Blöcke verwendet, die als Gehirn des Roboters fungieren. In diesem Projekt wird die EV3-Serie verwendet. Sie wurde 2013 veröffentlicht und ist die dritte Generation der LEGO-Mindstorms Serie. EV3 steht für "`Evolution 3`". Die EV3-Serie bietet viele Verbesserungen in Bezug auf die Anzahl der Ein- und Ausgänge, die Leistung und die Technologie wie Kommunikation, Verarbeitungsleistung, Speicher und Verbesserungen bei Motoren und Sensoren. Darüber hinaus macht das verbesserte Design mit neuen Farben und Installation einfacher und besser [2].

III. Bausteine und Funktionsprinzip

A. Aufbau:

Für den Bau dieses Fahrzeugs wurden die folgenden Teile aus dem LEGO Mindstorms-Bausatz verwendet (siehe Abbildung 6).

- Ultraschallsensor

Der Ultraschallsensor kann die Entfernung zwischen sich und einem beliebigen Objekt vor ihm messen, wie in Abbildung 2 gezeigt wird [3], und ist so programmiert, dass er Befehle zum Anhalten des Fahrzeugs gibt, wenn er ein Objekt in einem Abstand von weniger als 8 cm vor sich wahrnimmt. Wenn der Abstand größer als 8 cm ist, gibt er dem Wagen den Befehl, sich wieder in Bewegung zu setzen.

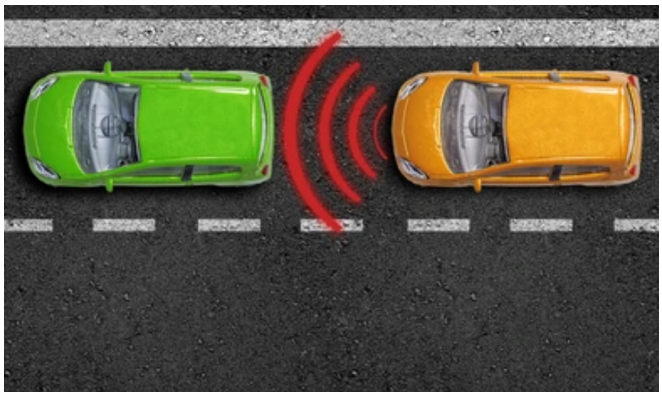


Abbildung 2: Abstandssensor Diagramm

- Motor

Die Motoren werden ausschließlich zum Drehen verwendet, und ihre Geschwindigkeit, Dauer und Drehrichtung werden durch Programmierung gesteuert, wobei die Programmiersprache MATLAB verwendet wurde. In diesem Projekt wurden zwei Motoren parallel installiert. Sie bewegen sich mit einer

DOI: 10.24352/UB.OVGU-2024-030

konstanten Geschwindigkeit miteinander, um das Fahrzeug vorwärts oder rückwärts zu bewegen. Sie drehen sich entgegengesetzt zueinander, um das Fahrzeug nach links oder rechts zu abbiegen.

- Farbsensor

Ein Farbsensor kann mehrere Farben erkennen. In dem Projekt wurden nur vier davon verwendet: Weiß, Schwarz, Rot und Grün. Jede Farbe gab der Software einen anderen Befehl. Die Software startet, wenn die Farbe Grün erkannt wird. Das Fahrzeug biegt nach rechts ab, wenn Weiß erkannt wird, und nach links, wenn Schwarz erkannt wird. Das Fahrzeug hält an, wenn die Farbe Rot erkannt wird.

- Tastsensor

Dies ist der Notrufknopf, der das Fahrzeug jederzeit stoppt, wenn er gedrückt wird. Er wird normalerweise verwendet, wenn die Dinge außer Kontrolle geraten.

B. Programm

1.Programmierung

Die Motoren und Sensoren werden gesteuert, indem sie an das EV3 angeschlossen werden. Sie wird mit der Programmiersprache MATLAB über die Datei `"EV3-toolbox-MATLAB-master"` programmiert. Die Datei "MindstormsEV3Toolbox.pdf" wird auch als Code-Referenz verwendet. Sie wurde für die Programmierung dieses Projekts verwendet. Sie enthält einige Beispielcodes für die EV3-Version und eine Erklärung [4]. Mit ihr können viele Projekte mit vielen Ideen programmiert werden. Diese PowerPoint-Datei enthält alle Codes, die für die Auswahl der Drehrichtung, Geschwindigkeit und Dauer der Drehung der Motoren verwendet werden, und die Funktion `"SpeedRegulation"` kann ebenfalls hinzugefügt werden. Es ist auch möglich, dem Lichtsensor zu befehlen, statt der Lichtintensität die Farbe zu erfassen, wie bereits in diesem Artikel als Farbsensor erwähnt. Es wird auch gezeigt, wie man den Ultraschallsensor so einstellt, dass er die Entfernung vor ihm erkennt.

```
while true
   distance = brick.sensor1.value;
   disp(['distance=',num2str(distance)]);
   if distance > 8
      if not(brick.motorA.isRunning)
         brick.motorB.setProperties('speedRegulation','off');
         brick.motorA.setProperties('debug','off','power',20,
          'limitValue',0,'speedRegulation','off');
         brick.motorB.setProperties('debug','off','power',20,
          'limitValue',0,'speedRegulation','off');
         brick.motorA.power=20;
         brick.motorA.syncedStart(brick.motorB);
      end
      disp('fahren')
   else
      disp('stoppen')
      brick.motorA.stop;brick.motorB.stop;
      for n=1:5
         color(n) = brick.sensor2.value;
         pause(0.05)
      end
```

Abbildung 3: Kurzer Ausschnitt des Quelltextes

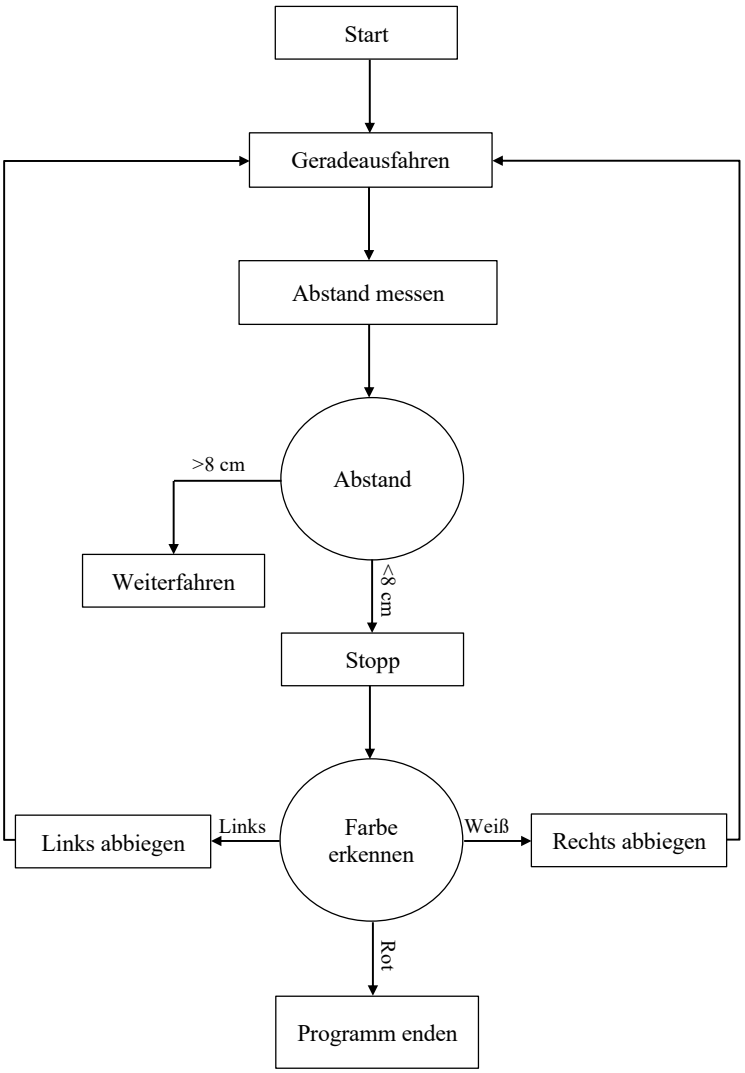


Abbildung 4: Programmablauf

2.Programmablauf

Der Wagen setzt sich in Bewegung, wenn die Schaltfläche `"Run"` in MATLAB gedrückt wird, und bewegt sich mit einer festgelegten und geregelten Geschwindigkeit vorwärts, wenn sich kein Objekt 8 cm vor ihm befindet. Erscheint ein Objekt weniger als 8 cm vor dem Wagen, hält der Wagen an und die Ultraschallsensorfunktion wird vorübergehend beendet; zur Veranschaulichung wird ein kurzer Ausschnitt des Quellcodes gezeigt (siehe Abbildung 3). Der Wagen verlässt sich dann auf den Farbsensor, um zu wenden. Wenn die Farbe Weiß erkannt wird, fährt der Wagen leicht zurück und dreht sich dann nach rechts. Wird Schwarz erkannt, kehrt der Wagen leicht um und dreht sich dann nach links. Der Grund, warum der Wagen rückwärtsfährt, ist, dass das Fahrzeug beim Anhalten leicht verzögert wird, wenn ein Objekt vor ihm erkannt wird. Anschließend fährt es wieder vorwärts, nachdem es je nach Farbe nach rechts oder links abgebogen ist. Das Fahrzeug hält an, wenn Rot erkannt wird, und das Programm endet (siehe Abbildung 4 und

5). Für den Fall, dass die Dinge außer Kontrolle geraten oder der Kontrolleur das Fahrzeug anhalten möchte, gibt es eine Notfalltaste, die Sensortaste.

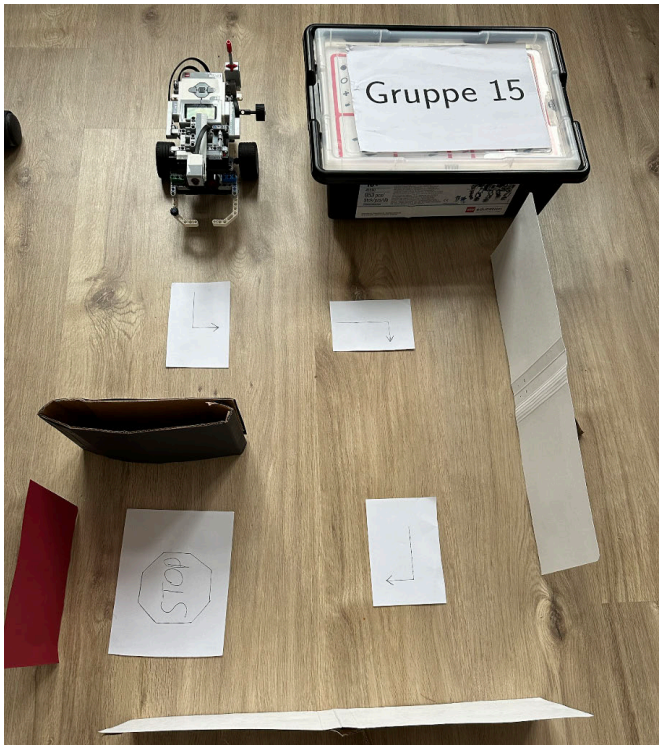


Abbildung 5: Beispiel für die lauf Plan

IV. Ergebnisdikussion

In diesem Projekt wurde die LEGO-Mindstorms Technologie verwendet. Es wurde mit MATLAB programmiert. Es ist ein vereinfachtes Beispiel und ein Miniaturmodell der vorzustellenden Idee. Dieses Projekt wurde erfolgreich umgesetzt. Es ist auch für die Zukunft skalierbar.

V. Zusammenfassung und Fazit

Am Ende des LEGO-Praktikums wurde der Maze Escaper programmiert und zusammengebaut. Der Maze Escaper ist ein einfaches Projekt, um die Idee eines intelligenten Autos auf einfache Art und Weise vorzustellen. Die meisten der für das Projekt gesetzten Ziele wurden erfolgreich erreicht. Es war eine sehr nützliche und angenehme Erfahrung und eine gute Gelegenheit, die Programmiersprache MATLAB zu erlernen, und nicht nur das, sondern auch zu lernen, wie man dieses Wissen anwendet, um Roboter zu entwickeln, die mehrere Funktionen ausführen.

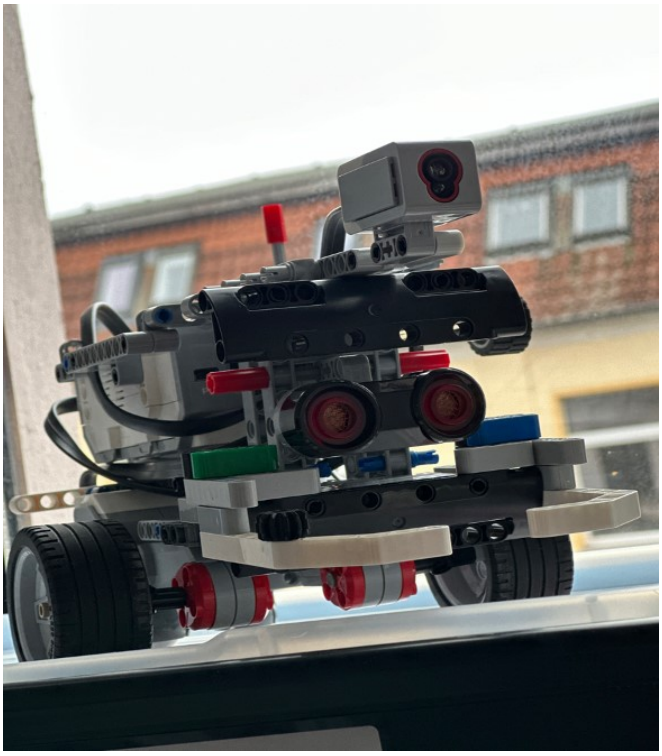


Abbildung 6: aufgebauter Wagen

Literaturverzeichnis

[1] https://www.techradar.com/news/self-driving-cars (Stand: Februar 2024)

[2] Wikipedia, The Free Encyclopedia: LEGO-Mindstorms EV3 (https://en.wikipedia.org/wiki/Lego_Mindstorms_EV3) (Version: 29.10.2023)

[3] https://www.dreamstime.com/asphalt-two-cars-road-distance-sensor-emergence-break-assistant-asphalt-cars-road-distance-image134395421 (Stand: Februar 2024)

[4] Alexander Behrens. (2020, Januar). Mindstorms EV3 Toolbox Documentation. [Online]. Available e-mail: behrens@lfb.rwth-aachen.de. pages Available: (www.mindstorms.rwth-aachen.de)

Selbstfahrendes Auto

Tayseer Khshainy, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Das Projektseminar der Fakultät für Elektro-und Informationstechnik der Otto-von-Guericke-Universität Magdeburg wird jährlich verrichtet. Diesmal wurde ein selbstfahrendes Auto erstellt. Die Herstellung des Autos wurde auf Basis vom LEGO-Mindstorms-Sets erfolgt, die der Nachfolger des NXT-Sets. Im Rahmen der hier vorgestellten Arbeit wurde die Ansätze für ein selbstfahrendes Auto umgesetzt. Da wurde ich auch erwähnet, wie der Bau und Algorithmen der Funktionsprinzip des Autos ist. Darüber hinaus wird einige Schwierigkeiten bei der Konstruktion des Autos und sowie auf deren Lösungsansätze eingegangen.**

***Schlagwörter*—EV3, Elektroauto, Farb, Ultraschall und Tastsensor, Matlab**

I. Einleitung

Autonome Fahrsysteme (AFS) haben in den letzten Jahren eine enorme Entwicklung durchgemacht. Elektroautos tragen durch die Senkung der CO2 Emission zum Umweltschutz bei. Dies erklärt, warum die meisten Automobilhersteller heute diese Technologie mit Unterstützung einiger Stellen, die die Verbreitung von Elektroautos fördern, aktiv entwickeln. Statistiken zu Verkehrsunfällen: Laut Unfallstatistik gab es im Jahr 2022 ca. 289 672 Verkehrsunfälle mit Verletzten [1]. Angesichts der zunehmenden Zahl von Verkehrsunfällen deuten die Anzeichen darauf hin, dass selbstfahrende Autos eine effektive Lösung zur Senkung der Unfallraten bieten könnten, da auch der Fahrer seine Zeit für andere Aktivitäten nutzen, wie Lesen oder sogar Entspannen, wie in Abbildung 1 dargestellt. Dies kann Müdigkeit reduzieren und die Konzentration auf die Straße verbessern, wenn ein Eingreifen erforderlich ist.

Abbildung 1: Hände frei beim Fahren [2]

II. Vorbetrachtungen

In diesem Papier wird ein selbstfahrendes Auto vorgestellt, wie ich im Projektseminar das Auto aufbaute, welche Motoren, Sensorik und Hardware-Implementierungsumgebung verwendet wurde und die Funktionsprinzip erklärt.

III. Bausteine und Funktionsprinzip

A. Aufbau:

- Struktur

In den ersten drei Tagen des Projekts wurde die Konstruktion des Autos gebaut. Beim Bau wurden Probleme erfahren, besonderes bei der Auswahl einer ausgewogenen Form, damit das Fahrzeug ausbalanciert ist und nicht umfällt, insbesondere da das Fahrzeug nur zwei Räder vorne hat und hinten zwei Kugelstücke hinzugefügte, um das Gleichgewicht zum Unterstützen, insbesondere beim Abbiegen.

Allgemeine über LEGO-EV3 und deren Sensoren:

- LEGO-EV3

Das LEGO-Mindstorms-EV3 ist das Nachfolgemodell des LEGO-Mindstorms NXT und wurde 2013 eingeführt. Das EV3-Gerät enthält ein Mikro-SD-Kartenleser, ein USB-Anschluss und vier Motoranschlüsse, mit der und deren Sensoren ein Roboter gebaut werden können, (siehe Abbildung 2).

Abbildung 2: EV3-Gerät und dessen Anschlüsse [3]

- Motor

Die an den Seiten des Autos Motoren dienen in der Regel dazu, sich zu beschleunigen und die Bewegungsrichtung zu ändern. Dies geschieht durch die Steuerung der Geschwindigkeit und Richtung der Bewegung des Autos. Die beiden Räder drehen sich gleichzeitig in entgegengesetzten Richtungen. Der erste Motor dreht sich vorwärts und zweite wird rückwärts

DOI: 10.24352/UB.OVGU-2024-031

gedreht. So wird die Bewegungsrichtung rechts oder links geändert.

- Farbsensor

Ein Farbsensor kann die Farben von Oberfläche der Objekte erkennen, es kann von Farbsensor idealer Zustand sieben Farben gemessen werden, aber wegen der Umgebungslicht und der Beschaffenheit der Objekte erkennt der Farbsensor nicht alle Farben, deshalb wurde den Farbsensor nach vorne verschoben, wie in Abbildung 3, damit er die Farben besser erkennen kann. Er misst die Wellenlänge des Lichtes, des von einem Objekt reflektiert wurde und vergleich die Anweisungen mit einer internen Datenbank.

- Tastsensor

Der Tastsensor befinden sich hinten am Ende des Autos. Unter Verwendung des Tastsensors wurde das Programm beendet und das Fahrzeug gestoppt, (siehe Abbildung 3).

- Ultraschallsensor

Er verwendet Schallwellen, um die Entfernung zu Objekten zu messen. Er sendet Schallwellen aus und misst die Entfernung von Objekten. Der Sensor basierend auf die Zeit, die es dauert, bis die Echos von Objekten zurückprallen. Anschließend kann die Entfernung dazwischen berechnet werden.

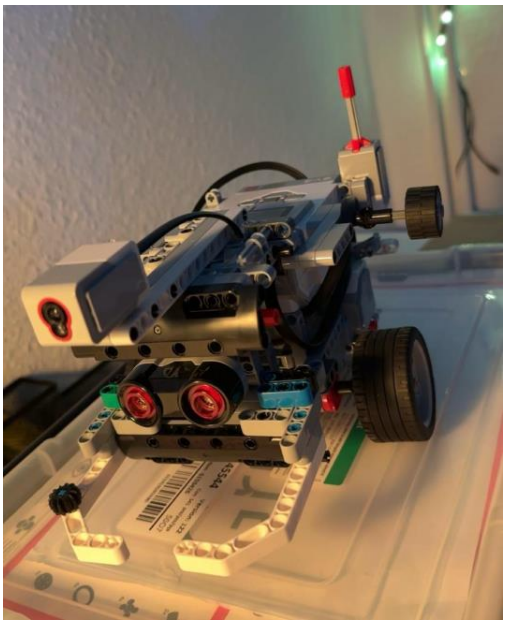


Abbildung 3: Frontsicht

B. *Programm*

1) *Programmierung*

Die Steuerung der Motoren und Sensoren erfolgt durch deren Anbindung an den EV3. Für die Programmierung wird die Sprache MATLAB verwendet, unterstützt durch das "EV3-toolbox-MATLAB-master". Als zusätzliche Programmierressource dient die Datei „MindstormsEV3Toolbox.pdf“, die Beispielcodes und Erklärungen für die EV3-Version bietet und für

Abbildung 3: Frontsicht

dieses Projekt genutzt wurde [4]. Mit dieser Dokumentation lassen sich zahlreiche Projekte umsetzen. Die dazugehörige Power-Point-Datei enthält alle notwendigen Codes zur Steuerung der Richtung, Geschwindigkeit und Dauer der Motorbewegungen. Auch die Funktion „SpeedRegulation“ kann integriert werden. Zusätzliche kann der Farbsensor so konfiguriert werden, dass er Farben statt Lichtintensität erfasst, wie bereits im Artikel beschrieben. Ferner wird erklärt, wie der Ultraschallsensor eingestellt wird, um Entfernung zu messen.
So wurde der Prozess, der in Abbildung 4 dargestellt ist, in MATLAB programmiert:

```
while true
   distance = brick.sensor1.value;
   disp(['distance=',num2str(distance)]);
   if distance > 8
      if not(brick.motorA.isRunning)

brick.motorB.setProperties('speedRegulation','off');

brick.motorA.setProperties('debug','off','power',20,
'limitValue',0,'speedRegulation','off');

brick.motorB.setProperties('debug','off','power',20,
'limitValue',0,'speedRegulation','off');
         brick.motorA.power=20;
         brick.motorA.syncedStart(brick.motorB);
      end
      disp('fahren')
   else
      disp('stoppen')
      brick.motorA.stop;brick.motorB.stop;
      for n=1:5
         color(n) = brick.sensor2.value;
         pause(0.05)
      end
```

Abbildung 4: Kurzer Ausschnitt des Quelltextes

2) *Programmierung der Funktionsweise*

Der Wagen beginnt sich zu bewegen, sobald die „Run“-Taste in MATLAB betätigt wird. Da fährt das Auto geradeaus, solange nichts vor ihm in einem Abstand von 8 cm ist und durch die Programmierung der Ultraschallsensor sendet er, wenn sich das Objekt weniger als 8 cm vor ihm befindet, Befehl an das Gerät, um das Auto anzuschalten, dann scannt der Farbsensor die vordere Farbe, um Befehl des Vorgehens vorzugeben. Diese sind das Anhalten bei einer roten Farbe. Zurückzugehen und links abzubiegen bei einer weißen Farbe oder zurückzugehen und rechts abzubiegen bei einer Schwarzen Farbe. Das Fahrzeug kommt zum Stillstand und das Programm wird beendet, sobald die Farbe Rot wahrgenommen wird. Zudem ist eine Notfalltaste, der Tastsensor, vorhanden für Situation, in denen die Kontrolle verloren geht oder der Bediener das Fahrzeug stoppen möchte, (siehe Abbildung 5.)

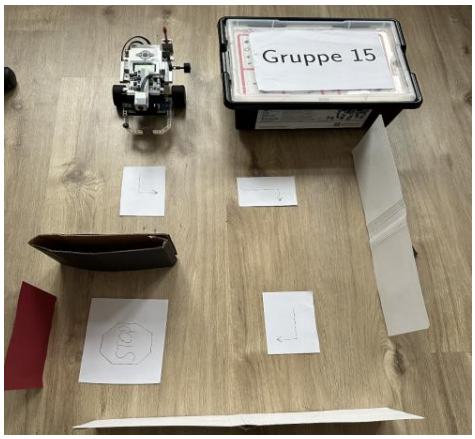


Abbildung 5: Fahrzeug mit Teststrecke

3) Laufplandiagramm

Die Auswertung der Sensorik und das Ansteuern von Motorik wird, wie in Abbildung 6 Laufplandiagramm gezeigt, erfolgen.

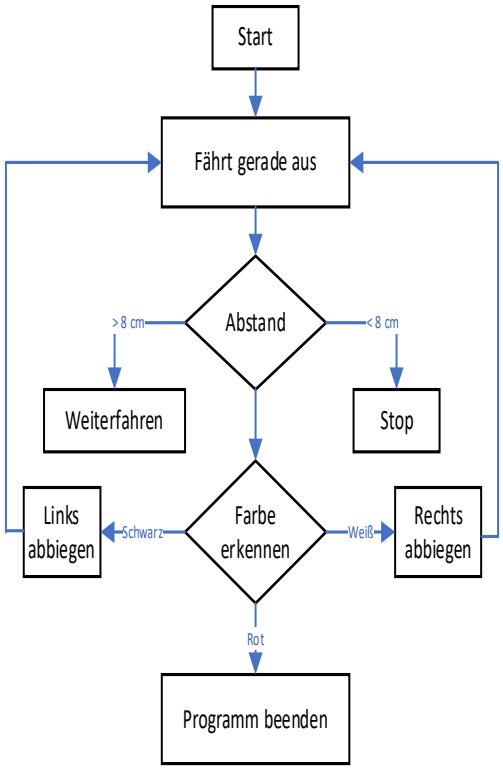


Abbildung 6: Programmablauf

IV. Ergebnisdiskussion

Ziel dieses Projekt war es, ein selbstfahrendes Auto zu bauen, indem es mit MATLAB programmiert wurde. Dieses Projekt wurde erfolgreich abgeschlossen, aber es gibt natürlich noch Verbesserungsmöglichkeiten für die Zukunft. Dazu gehört die Integration von Sensoren, die die Farben deutlicher und von fernerem Abstand erkennen können, sowie die Installation einer intelligenten Kamera zur Erkennung in 360°-Aussicht und von Verkehrszeichen und die Optimierung der Stabilität des Autos, die eine erhöhte Manövrierfähig des Fahrzeugs umfasst. Zusätzlich kann das Auto programmiert werden, dass es in der Lage ist in beliebigen Parkplatz zu parkieren.

V. Zusammenfassung und Fazit

Am Ende dieses LEGO-Praktikums wurde endlich ein selbstfahrendes Auto hergestellt, das wie in der Abbildung 7 aussieht, so dass der Farbsensor zur Richtungsbestimmung und einem Ultraschallsensor zur Abstandsmessung ist. Es handelte sich um eine äußerst hilfreiche und erfreuliche Erfahrung sowie eine ausgezeichnete Gelegenheit, die Programmiersprache MATLAB zu erlernen. Darüber hinaus bot sie die Chance, zu verstehen, wie man ein Roboter zu entwickeln, die vielfältige Aufgaben übernehmen können.

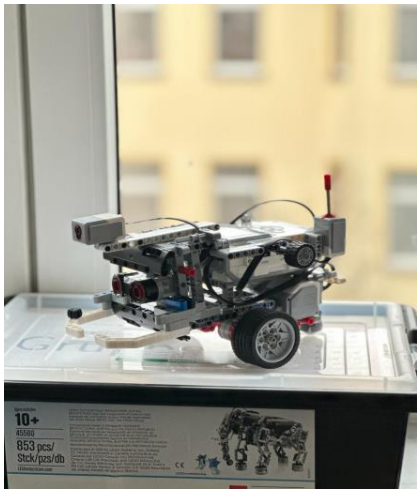


Abbildung 7: Endgültiger Aufbau

Literaturverzeichnis

[1] *Statistiches Bundesamt Straßenverkehrsunfälle nach Unfallkategorie (Personenschaden), Ortslage.* (https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Verkehrsunfaelle/Tabellen/polizeilich-erfasste-unfaelle.html) (Stand: Februar 2024)

[2] Süddeutsche Zeitung, *Wenn Computer Autofahrer ablösen,* (https://www.sueddeutsche.de/auto/autonomes-fahren-wenn-computer-den-menschen-abloesen-1.2831833) (Version: 29.10.2023)

[3] Wikimedia Commons, LEGO-EV3 (https://commons.wikimedia.org/w/index.php?curid=28877049) (Version: 6.10.2013)

[4] Alexander Behrens. (2020, Januar). Mindstorms EV3 Toolbox Documentation. [Online]. Available e-mail: behrens@lfb.rwth-aachen.de. pages Available: (www.mindstorms.rwth-aachen.de)

Skorpion Puzzle - Aus Spielzeug, Spielzeug bauen

Tim Kasten, ETIT
Otto-von-Guericke-Universität Magdeburg

***Abstract*—In einem Projektseminar für Elektrotechnik und Informationstechnik erhielten die Studierenden zwei Bauteilsets aus der LEGO®-Mindstorms® Produktreihe. Ihre Aufgabe bestand darin, einen funktionsfähigen Roboter zu entwickeln und die dazugehörige Software mithilfe von MATLAB zu programmieren. Im Zuge dieses Projekts wurde ein Roboter konstruiert, der eine innovative Art des Puzzelns ermöglichte. Dieser Roboter fokussiert sich nicht darauf, Bilder zusammenzusetzen, sondern auf das Erkennen von Melodien. Die Lieder wurden in einzelne Abschnitte zerlegt, wobei jedem Abschnitt eine bestimmte Farbe zugeordnet wurde. Der Roboter navigiert über diese farbigen Segmente, und wenn sie in der richtigen Reihenfolge angeordnet sind, spielt der Roboter das Lied korrekt ab. Auf diese Weise verfolgt der Roboter einen alternativen Ansatz im Vergleich zu herkömmlichen Puzzles, der interaktiver und spannender ist.**

Schlagwörter—LEGO® Mindstorms®, Musik, Puzzle, Roboter, Spielzeug.

I. Einleitung

In den letzten Jahren hat die Integration von Lernrobotern und vereinfachten Programmierumgebungen in Bildungseinrichtungen einen deutlichen Aufschwung erfahren. Durch den Einsatz von Roboterkits wie LEGO®-Mindstorms® wird es für Schülerinnen, Schüler und Studierende immer einfacher, erste Schritte in der Welt der Robotik zu unternehmen. Im Rahmen eines Projektseminars Elektrotechnik/Informationstechnik wurde ein Roboter entwickelt, der auf den LEGO®-Mindstorms®-EV3 Bausätzen basiert. Ziel war es, einen Roboter zu entwerfen, der Farben erkennen und je nach Farbe bestimmte Abschnitte bekannter Melodien abspielen kann. Dabei wurden Melodien aus verschiedenen Bereichen wie Videospielen mit „Zelda“, der Filmindustrie mit Star Wars und klassischer Musik mit „Für Elise“ ausgewählt. Die Realisierung erfolgte durch die Programmierung des EV3 mit MATLAB, dass über eine Schnittstelle der RWTH Aachen[1] die Steuerung verschiedener Sensoren und Motoren ermöglicht. Dieser Roboter bietet eine innovative Herangehensweise an das Puzzle-Konzept, indem er Melodieteile anstelle von Bildteilen verwendet. Auf diese Weise können die Nutzerinnen und Nutzer durch die Anordnung verschiedener Farbkarten eine vollständige Melodie erstellen. Zur Unterstützung des Farbkartenwechsels verfügt der Roboter über eine Fahrautomatik und eine Start-Stopp-Automatik, die die beiden Motoren für die Fortbewegung des Roboters steuern. Das Hauptziel dieses Projekts war es, Studierenden grundlegende Kenntnisse in den Bereichen Robotik, Programmierung und Projektmanagement zu vermitteln. Dazu wurde den Teilnehmenden die in Zweiergruppen arbeiteten, zwei Kästen LEGO® sowie die Möglichkeit, den gesamten Prozess von der Ideenfindung über die Umsetzung bis hin zur abschließenden Präsentation eigenständig zu gestalten, gegeben.

II. Vorbetrachtungen

A. *LEGO®-Mindstorms®*

LEGO®-Mindstorms® ist eine innovative Produktlinie von LEGO®, die gezielt entwickelt wurde, um das Bauen und Programmieren von Robotern zu erleichtern. Sie dient nicht nur Bildungszwecken, sondern auch der grundlegenden Entwicklung von Robotertechnologien. Die verschiedenen Kits enthalten eine Vielzahl modularer, Sensoren Motoren sowie die Typischen Legosteine, die es den Benutzern ermöglichen, ihre eigenen Roboter nach individuellen Vorstellungen zu gestalten und zu programmieren. Durch die Nutzung verschiedener Gerätegenerationen mit unterschiedlichen Sensoren können Anwender die für ihre spezifischen Anforderungen am besten geeignete Technologie wählen.

In diesem Projekt wurde die EV3-Version von LEGO®-Mindstorms® eingesetzt. Die Programmierung erfolgt in der Regel über eine benutzerfreundliche grafische Programmiersprache, die von LEGO® entwickelt wurde und besonders für Anfänger*innen zugänglich ist. Jedoch wurde im Rahmen dieses Projekts eine Schnittstelle der RWTH Aachen verwendet, um den Funktionsumfang zu erweitern. Diese Schnittstelle ermöglicht es, mithilfe von MATLAB-Code die Sensoren und Motoren des Roboters präzise anzusteuern und komplexere Projekte umzusetzen, die über die Möglichkeiten der einfachen „LEGO-Programmiersprache“ hinausgehen.

B. *Auswahl der Melodien*

Aufgrund der begrenzten Leistungsfähigkeit des Lautsprechers und der Verarbeitungsgeschwindigkeit des Mindstorms-Prozessors ist es von entscheidender Bedeutung, Melodien sorgfältig auszuwählen. Diese müssen einerseits allgemein bekannt sein, um von allen potenziellen Benutzern wiedererkannt zu werden. Andererseits ist zu beachten, dass der EV3-Brick nur in der Lage ist, einen Ton zur gleichen Zeit abzuspielen. Dadurch können komplexere Tonüberlagerungen nicht wiedergegeben werden. Daher ist die Melodienauswahl auf bekannte und einfache Melodien beschränkt, die auch mit längeren

DOI: 10.24352/UB.OVGU-2024-032

Tönen klar erkennbar sind. Als Ergebnis wurde sich für Melodien aus Star Wars, Zelda und für Das Lied „Für Elise" entschieden.

Es ist jedoch wichtig zu anzumerken, dass diese Songauswahl nachträglich erweitert werden kann.

C. *Wiedergabe von Melodien*

Um Tonausgaben auf dem EV3 zu ermöglichen, muss eine spezifische Anweisung an den Brick gesendet werden. Diese Anweisung enthält Informationen über die Frequenz eines bestimmten Tons, seine Dauer sowie die gewünschte Lautstärke. Um ganze Melodien abzuspielen, muss die Melodie zuerst in Frequenzen und Tonlängen zerlegt werden, damit sie anschließend schrittweise an den Brick gesendet werden kann. Hierfür können Umwandlungstabellen verwendet werden, die entsprechende Frequenzen für die verschiedenen Töne bereitstellen.

D. *Probleme mit den Sensoren*

Die Sensoren, die für diesen Roboter erforderlich sind, bestehen aus zwei Tastsensoren und zwei Farbsensoren. Die Tastsensoren haben die Funktion, entweder anzuzeigen, ob sie im aktuellen Moment gedrückt sind, oder die Anzahl der bisherigen Betätigungen zu zählen. Letzteres wurde im Programm verwendet, um mit einem der Taster zwischen den Melodien zu wechseln und mit Hilfe des anderen Tasters die Wiedergabe sowie den Motor zu stoppen.

Bei der Umsetzung der Projektidee mussten insbesondere die Ungenauigkeiten der Farbsensoren berücksichtigt werden, da sie häufig unzuverlässige Ergebnisse lieferten. Insbesondere bei schlechten Lichtverhältnissen wurden Farben oft falsch erkannt. Um diesem Problem vorzubeugen, wurden zwei Sensoren eingebaut, die im Programm miteinander abgeglichen wurden.

Außerdem war ursprünglich geplant, einen Gyrosensor zu verwenden, um die Lenkung des Roboters zu überwachen. Dieser Sensor sollte einerseits den Drehwinkel und andererseits die Drehgeschwindigkeit messen. Allerdings wurde diese Idee aufgegeben, da der Gyrosensor fehlerhafte Werte lieferte und insgesamt ungenau war.

III. Der Code

Das Programm basiert auf einem sich nach jeder Iteration wiederholenden Schema, das im folgenden Ablaufdiagramm (Abb. 1) veranschaulicht wird. Zunächst muss der Benutzer mithilfe eines Tasters eine der drei Melodien (Star Wars, Für Elise, Zelda) auswählen. Da der Tastsensor auf zwei Modi beschränkt ist - einem Modus, in dem er nur während des Drückens ein Signal ausgibt, und einem Modus, in dem die absolute Anzahl der Betätigungen gezählt wird - muss zunächst in den absoluten Zählmodus gewechselt werden. Anschließend ermöglicht das Programm durch die Berechnung des Modulos `n = mod(tastsensorcount, 3)` einen Wechsel zwischen drei Modi, die jeweils einer Melodie entsprechen:

$$n = \begin{cases} 0, & \text{Starwars} \\ 1, & \text{Zelda} \\ 2, & \text{Für Elise} \end{cases}$$

Mithilfe von if-Statements wird jeder dieser Modi einer Melodie zugeordnet, die im weiteren Verlauf des Programms verwendet wird. Anschließend werden die beiden verwendeten Farbsensoren ausgelesen und dreimal abgeglichen. Falls die Werte der Sensoren bei den drei Abgleichen nicht übereinstimmen, wird der Wert eines der beiden Sensoren genutzt. Basierend darauf wird ein Teil der Melodie abgespielt, der durch Switch-Statements an die Farbausgabe der Farbsensoren gekoppelt ist. Eine typische Ausgabe wäre "blaue Farbe", die dann beispielsweise "case blue" und den entsprechenden Melodieausschnitt abspielt.

Solange die Melodie abgespielt wird, stoppt der Roboter, und erst wenn eine Melodie beendet ist, wird die Fahrt fortgesetzt. Sobald der Farbsensor eine neue Farbe erkennt, wird der dazu korrespondierende Melodieabschnitt abgespielt, und der Roboter stoppt erneut. Dieser Vorgang wiederholt sich, bis ein zweiter Taster, der nach dem gleichen Funktionsprinzip wie der Taster zum Wechseln der Melodie funktioniert, ein Stopsignal auslöst und die Melodie beendet wird, während der Motor in den Bremsmodus wechselt.

Um das Programm nachträglich erweiterbar zu gestalten, wurden die einzelnen Melodien in separate Funktionen integriert. Dadurch ist es möglich, zusätzliche Melodien hinzuzufügen, solange sie das erforderliche Format für die einzelnen Töne einhalten:

```
"tone(Frequenz (in Hz), Lautstärke (in %), Länge (in ms))"
```

Um zu erkennen, ob der Roboter weiterfahren kann oder die Melodie noch zu Ende gespielt werden muss, wurde außerdem eine Funktion implementiert, die überprüft, ob er sich noch auf derselben Farbe befindet oder bereits auf einer neuen Farbe steht und deshalb stoppt, bis die Melodie vollständig abgespielt wurde.

IV. Konstruktion

Neben der Implementierung der Software war es von entscheidender Bedeutung, ein robustes Gerüst zu konstruieren, das sowohl mehrere Sensoren als auch zwei Motoren sicher halten konnte. In der ersten Version des Roboters wurde auf Ketten als Antrieb gesetzt sodass die Lenkung mittels einer klassischen „Panzersteuerrung“ realisiert werden konnte (Abb. 2), jedoch obwohl die Legoteile hervorragend für ihren Zweck als Spielzeug sind, sind sie ungeeignet für mechanisch präzise Aufgaben. Dies führte dazu, dass der Roboter nicht geradeaus fahren konnte und zufällig nach rechts oder links abwich. Aus diesem Grund wurden die Ketten durch vier Räder ersetzt, was es dem Roboter ermöglichte, präziser zu fahren und vorerst auf eine Lenkung verzichtet (Abb. 3).

Neben den unerlässlichen Komponenten war auch das Design ein zentraler Aspekt der Entwicklung. Ein ikonisches

Design steigert den Wiedererkennungswert erheblich und ist somit entscheidend für ein marktfähiges Produkt. Außerdem bieten die Optikelemente des Puzzles zusätzliche Stabilität und Griffigkeit trotz dessen das diese modular und leicht demontierbar sind. Letzteres war besonders Hilfreich bei der Fehlerbehebung und Weiterentwicklung.

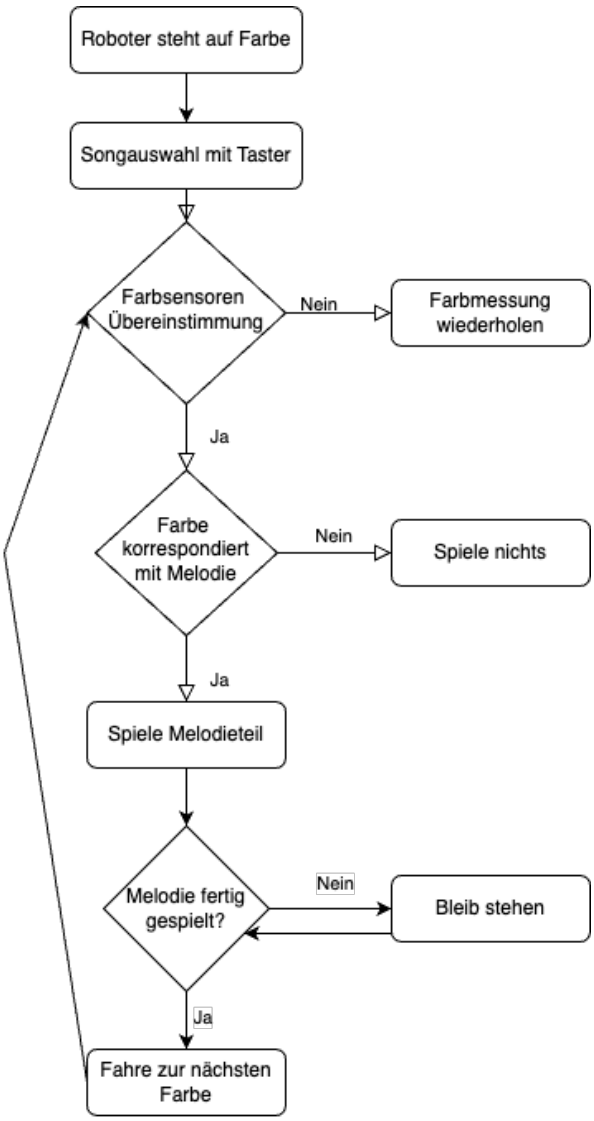


Abb. 1 Ablaufplan

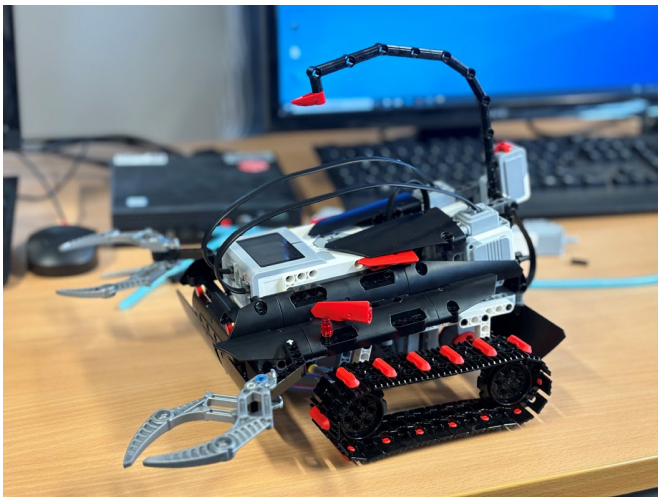


Abb. 2 Mark 2

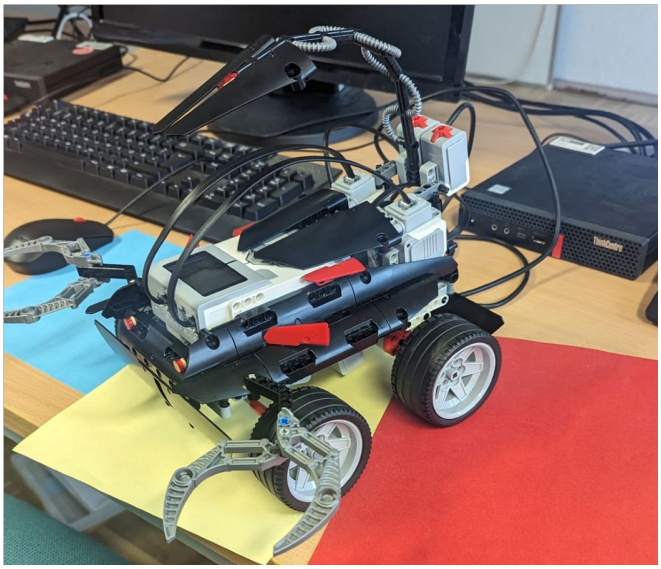


Abb. 3 Mark 6 / Finale Version

V. Perspektivische Erweiterungen

Aktuell sind drei Melodien integriert, jedoch kann diese Anzahl unbegrenzt erweitert werden, sofern für jede weitere Melodie Funktionen implementiert werden, die dem gleichen Schema folgen wie die bereits integrierten Melodien. Der begrenzende Faktor liegt dabei in der Steuerung zwischen den Melodien. Aufgrund der eingeschränkten Nutzbarkeit der Taster, wie bereits in den Vorüberlegungen erläutert, ist es nur komfortabel, eine geringe Anzahl an Melodien zu nutzen. Bei jedem Wechsel muss man durch verschiedene Melodien navigieren, ohne eine spezifische Melodie direkt auswählen zu können.

Eine weitere Möglichkeit zur Erweiterung besteht darin, einen externen Lautsprecher anzubringen. Dieser bietet nicht nur eine bessere Tonqualität als der im Steuerungsmodul vorhandene, sondern kann auch komplexere Tonabfolgen mit geringerer Latenz abspielen. Dadurch würde sich ein größeres Repertoire an Melodien eröffnen.

Für die weitere Implementierung von Melodien sollte außerdem eine Zusatzfunktion im Programm vorhanden sein, die es ermöglicht, aus fertigen Songs konvertierte und abspielbare Melodien zu erstellen. Obwohl dies eine aufwendige Erweiterung wäre, ist sie unumgänglich, um eine breitere Auswahl an Melodien zu bieten.

Durch eine Umgestaltung der Steuerung wäre es zukünftig auch möglich, weitere Fahrmodi einzuführen. Diese könnten beispielsweise eine Änderung der Fahrtrichtung in Korrespondenz mit der ausgelesenen Farbe ermöglichen.

VI. Fazit

LEGO®-Mindstorms® bietet eine Vielzahl kompatibler Klemmbausteine, hervorragen für den Einstieg in die Welt der Roboterentwicklung. Dank der intuitiven Anwendung, ist der Entwicklung eigener Projekte, nur durch die begrenzte Flexibilität der Klemmbausteine im Einzelnen Grenzen gesetzt. Die größten Herausforderungen ergeben sich jedoch aus den Sensoren, die sowohl in ihrer Genauigkeit als auch in ihren Ausgabemodi begrenzt sind. Dadurch sind komplexere Projektideen zwar eingeschränkt, aber durch die Integration zusätzlicher Mikrocontroller, Sensoren oder ähnlicher Komponenten durchaus realisierbar.

Die Einbindung von MATLAB funktioniert grundsätzlich gut, jedoch treten häufig Konnektivitätsprobleme auf, was zu einer erheblichen Zeitinvestition in die Fehlerbehebung während der Entwicklungsphase führt. Die größten Herausforderungen bei der Entwicklung dieses Roboters waren einerseits die Gewährleistung einer zuverlässigen Steuerung und andererseits der Umgang mit den Sensoren.

Vom Ansatz her war das Projekt ideal, um einen Einstieg in die Programmierung zu bieten, da eine Vielzahl grundlegender Programmierkonzepte implementiert werden musste. Zusätzlich forderte das Projekt gewisse Kenntnisse in Mechanik und Konstruktion um die gewünschten Roboter zu bauen, was über dem Rand des Projekts hinaus durchaus Interessant für die Zukunft ist.

VII. Literaturverzeichnis

[1] Lehrstuhl für Bildverarbeitung RTWH Aachen, [Online]. Available: https://www.lfb.rwth-aachen.de/de/academics/lectures/mindstorms/ [Zugriff am 08.03.2024].

Entwicklung eines Soundpuzzleroboters

Matthes Schaefer, Medizintechnik
Otto-von-Guericke-Universität Magdeburg

***Abstract*— Im Zuge des Projektseminar Elektrotechnik/Informationstechnik wurde ein Roboter auf Basis der LEGO Mindstorms Bausätze gebaut. Dabei wurde die Version EV3 genutzt, um mit Hilfe von Matlab einen funktionstüchtigen Roboter zu bauen, welcher in der Lage ist Farben zu scannen und abhängig von der Farbe bestimmte Abschnitte bekannter Melodien zu spielen. Durch die Aneinanderreihung verschiedener Farbkarten kann somit eine vollständige Melodie gespielt werden. Zum Wechsel der Farbkarten nutzt der Roboter eine Fahrautomatik, sowie eine Start-Stoppautomatik. Das Ziel des Roboters ist es eine neuartige Form des Puzzelns zu ermöglichen, die nicht auf der Anordnung von Bildteilen, sondern auf der Anordnung von Melodieteilen basiert.**

Schlagwörter: Matlab, Roboter, Farbsensor, Sound

I. Einleitung

Die Integration von Lernrobotern, sowie vereinfachten Programmierumgebungen in Bildungsumgebungen hat in den letzten Jahren stark zugenommen. Die Verwendung von Roboterkits wie LEGO Mindstorms bietet eine praktische Möglichkeit, Studenten oder Schülern eine Einführung in die Welt der Robotik zu ermöglichen. In diesem Zusammenhang wurde im Rahmen des Projektseminars Elektrotechnik/Informationstechnik ein Roboter entwickelt, der auf Basis der LEGO Mindstorms EV3 Bausätze konstruiert wurde. Dabei war das Ziel, einen funktionsfähigen Roboter zu entwerfen, der in der Lage ist, Farben zu erkennen und abhängig von der erkannten Farbe bestimmte Abschnitte bekannter Melodien abzuspielen. Die genutzten Melodien, wurden aus bekannten Spielen wie Zelda, den Star Wars Filmen, sowie aus der Klassik gewählt. Die Umsetzung dieses Konzepts erfolgte durch die Programmierung des EV3 mit Matlab, welches durch die von der RWTH Aachen entwickelten Schnittstelle verschiedene Sensoren und Motoren ansteuern kann. Der Roboter soll eine innovative Herangehensweise an das Puzzle-Konzept bieten, indem er Melodieteile anstelle von Bildteilen verwendet. Dies ermöglicht es den Benutzer*innen, durch die Anordnung verschiedener Farbkarten eine vollständige Melodie zu erzeugen. Um den Wechsel zwischen den Farbkarten zu ermöglichen, ist der Roboter mit einer Fahrautomatik sowie einer Start-Stoppautomatik ausgestattet, welche die zwei Motoren, die zum Fahren des Roboters nötig sind, ansteuern. Das Hauptziel dieses Projekts war es Student*innen Grundlagen der Robotik, der Programmierung, sowie Projektmanagementkompetenzen zu vermitteln, indem in Form von zweier Gruppen der komplette Prozess von Ideenfindung, über Umsetzung bis Hin zur finalen Präsentation freie Hand gelassen wurde.

II. Vorbetrachtungen

A. LEGO Mindstorms

LEGO Mindstorms ist eine Produktreihe von LEGO, die speziell für den Bau und die Programmierung von Robotern entwickelt wurde. Sie bietet eine Plattform für Bildungszwecke, sowie für die rudimentäre Entwicklung von Robotern. Die Kits enthalten modulare Bausteine, Sensoren und Motoren, die es Benutzern ermöglichen, Roboter nach ihren eigenen Vorstellungen zu konstruieren und zu programmieren. Dabei gibt es verschiedene Gerätegenerationen, welche verschiedene Sensoren nutzen. In diesem Projekt wurde die Version EV3 genutzt. Die Programmierung erfolgt typischerweise über eine grafische von LEGO selbst entwickelte Programmiersprache, die eine einfache und zugängliche Schnittstelle bieten, insbesondere für Anfänger*innen. Um den Funktionsumfang zu erweitern, wurde in diesem Projekt eine von der RWTH Aachen [1] entwickelte Schnittstelle genutzt, die es ermöglicht mit Hilfe von Matlab Code Sensoren, sowie Motoren des Roboters anzusteuern und auszulesen um somit komplexe Projekte, welche über die eigentlichen Fähigkeiten der grafischen Programmiersprache hinausgehen zu ermöglichen.

B. Auswahl der Melodien

Aufgrund der eingeschränkten Fähigkeiten des Lautsprechers, was sowohl die Verarbeitungsgeschwindigkeit des Mindstorm Prozessors angeht als auch die Fähigkeit Töne zu erzeugen, ist es essenziell Melodien zu wählen, die zum einen bekannt sind und somit für alle potentiellen Benutzer*innen wiedererkennbar sind. Des Weiteren kann der Brick des EV3 lediglich einen Ton gleichzeitig abspielen, wodurch komplexere Überlagerungen von Melodien nicht mit abspielbar wären. Damit ist man in der Auswahl der Melodien auf bekannte, simple Melodien begrenzt, die auch mit längeren Tönen wiedererkennbar sind. Dabei wurde sich für die Melodien von Star Wars, Zelda und Für Elise entschieden. Diese Songauswahl ist im Nachhinein erweiterbar.

C. Wiedergabe von Melodien

Um auf dem EV3 Tonausgaben zu ermöglichen, muss eine Anweisung an den Brick geschickt werden, welche eine Frequenz für einen bestimmten Ton enthält, die Länge des Tons, sowie die Lautstärke, in der der Ton ausgegeben werden soll. Für die Wiedergabe ganzer Melodien muss daher vorher eine Melodie in Frequenzen, sowie Tonlänge zerlegt werden, um

DOI: 10.24352/UB.OVGU-2024-033

im folgenden Ton für Ton an den Brick gesendet zu werden. Dafür können Umwandlungstabellen genutzt werden, die korrespondierende Frequenzen zu den Tönen liefern.

D. Ungenauigkeiten von Sensoren

Die Sensoren, welche für diesen Roboter benötigt werden, bestehen aus zwei Tastsensoren, sowie zwei Farbsensoren. Die Tastsensoren sind darauf beschränkt auszugeben, ob sie im aktuellen Moment gedrückt werden, oder wie oft sie bereits gedrückt wurden. Zweiteres wurde im Programm genutzt, um mit einem der Taster die Melodien zu wechseln und mit Hilfe des anderen Tasters die Wiedergabe, sowie den Motor zu stoppen. In der Umsetzung der Projektidee waren vor allem die Ungenauigkeiten der Farbsensoren zu berücksichtigen, da diese häufig unzuverlässig in ihren Angaben waren. Besonders bei schlechter Beleuchtung, wurden Farben falsch erkannt. Um diesem Umstand vorzubeugen, wurden zwei Sensoren eingebaut, welche im Programm miteinander abgeglichen werden.

III. SOFTWAREENTWICKLUNG

Das Programm basiert auf einem Sich nach jeder Iteration wiederholendem Schema, welches im folgenden Ablaufdiagramm (Abb. 1) veranschaulicht wird.
Als erstes muss durch den Nutzer mit Hilfe eines Tasters eine der drei Melodien (Star Wars, Für Elise, Zelda) gewählt werden. Da der Tastsensor auf zwei Modi begrenz ist, einem Modus, in dem er nur während des drückens ein Signal ausgibt und einen Modus, in dem die absolute Anzahl des Drückens gezählt wird, musste man als erstes in den absoluten Zählmodus wechseln. Davon ausgehend wurde in dem Programm durch die Berechnung des Modulos
$n = mod(tastercount, 3)$ ein Wechsel zwischen drei Modi ermöglicht, welche jeweils einer Melodie entsprechen.

$$n = \begin{cases} 0, & \text{Starwars} \\ 1, & \text{Zelda} \\ 2, & \text{Füre Elise} \end{cases}$$

Mit Hilfe von if Statements wurde jedem der drei möglichen Varianten von n eine Melodie zugewiesen, welche im Folgenden genutzt wird. Danach wurden die zwei verwendeten Farbsensoren ausgelesen und dreimal abgeglichen, falls sie in den drei Abgleichen nicht übereinstimmen, wurde der Wert eines der beiden Sensoren genutzt. Um darauf basierend einen Teil der Melodie abzuspielen, welche durch Switch Statements an die Farbausgabe der Farbsensoren gekoppelt sind. Eine typische Ausgabe wäre $blue$, welche dann z.B. $case\ blue$ und den damit verbundenen Melodieausschnitt abspielt. Solange die Melodie gespielt wird, stoppt der Roboter, erst wenn eine Melodie beendet wird, wird die Fahrt fortgesetzt. Sobald der Farbsensor eine neue Farbe erkennt, wird der dazu korrespondierende Melodieabschnitt abgespielt und der Roboter stoppt. Das wiederholt sich bis man mit Hilfe eines zweiten Tasters, der auf dem gleichen Funktionsprinzip wie der Taster zum Melodiewechsel basiert, ein Stopsignal ausgelöst wird und die Melodie beendet wird und der Motor in den Bremsmodus wechselt.

$$m = \begin{cases} 0, & \text{Roboter fährt, Melodie wird gespielt} \\ 1, & \text{Roboter bremst, Melodie stoppt.} \end{cases}$$

Damit das Programm im Nachhinein erweiterbar ist wurde es so aufgebaut, dass die einzelnen Melodien in eigene Funktionen platziert wurden. Dadurch ist es möglich im nachhinein weitere Melodien einzupflegen, solange sie die benötigte Formatierung für die einzelne Töne befolgen:
$tone(Frequenz\ (in\ Hz), Lautstärke\ (in\ \%), Länge\ (in\ ms)$
Damit der Roboter erkennt, ob er weiterfahren kann, oder die Melodie noch zu Ende spielen muss wurde zudem eine Funktion implementiert, die Überprüft, ob er noch auf derselben Farbe steht, oder sich bereits auf einer neuen Farbe befindet und deshalb stoppt bis die Melodie zu Ende gespielt wurde.

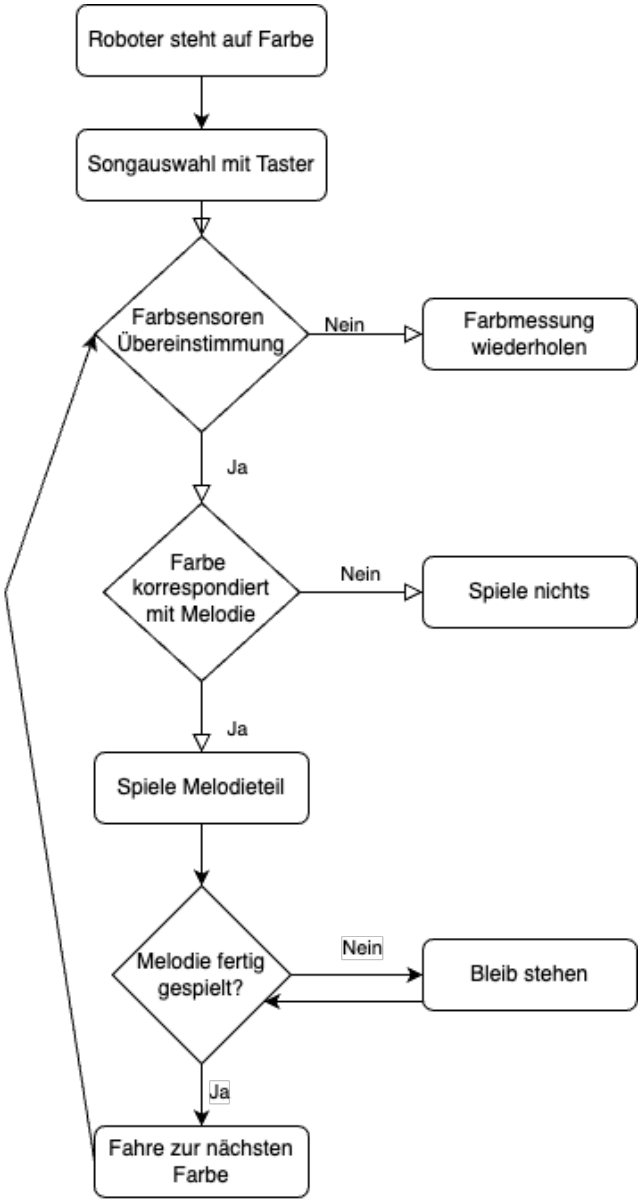


Abbildung 1 Ablaufdiagramm

IV. Grundkonstruktion

Neben der softwareseitigen Umsetzung war es essentiell ein Gerüst zu bauen, dass sowohl zwei Sensoren als auch zwei Motoren halten kann. Nachdem in der ersten Variante des Roboters eine Differentialsteuerung gescheitert ist, weil die verbauten Teile zu Ungenauigkeiten in der Steuerung führten, wodurch der Roboter nicht in der Lage war geradeaus zu fahren und zufällig nach recht oder links abgewichen ist. Daher wurden die Ketten durch vier Räder ersetzt, was es ermöglichte, geradeaus zu fahren. Neben den essenziellen Bestandteilen war auch das Design ein großer Interessenspunkt in der Entwicklung, da ein ikonisches Design zu einem hohen Wiedererkennungswert führt und dadurch auch für ein marktreifes Produkt entscheidend ist.

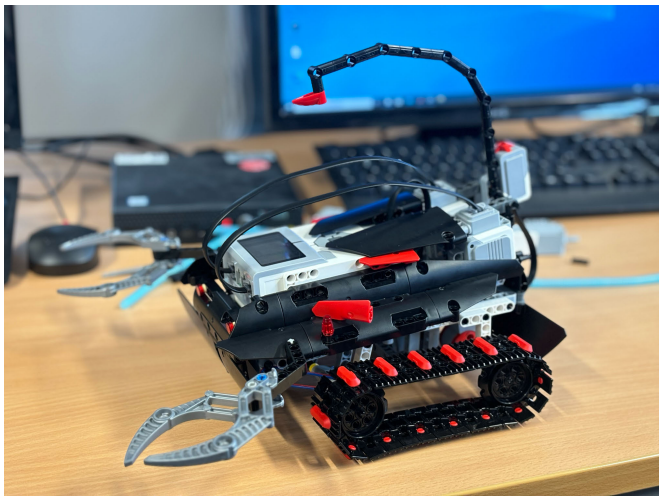


Abbildung 2 Erste Iteration

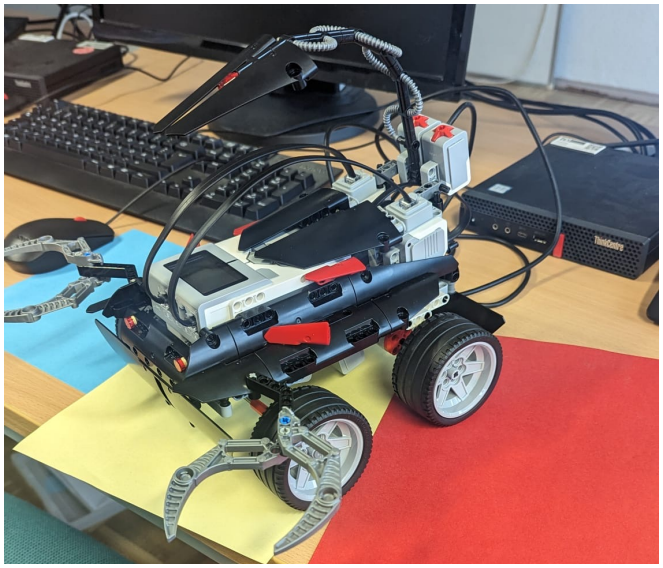


Abbildung 3 Finale Variante

V. Perspektivische Erweiterungen

Aktuell sind drei Melodien eingepflegt, das kann unbegrenzt erweitert werden, unter der Bedingung, dass man für jede weitere Funktionen implementiert, die demselben Schema, wie die bereits implementierten Melodien entsprechen. Der begrenzende Faktor wäre dabei die Steuerung zwischen den Melodien, da diese durch die eingeschränkte Nutzbarkeit der Taster, wie bereits in den Vorüberlegungen aufgeführt, nur für eine geringe Anzahl an Melodien komfortabel zu nutzen ist, da man mit jedem Wechsel durch verschiedene Melodien wechseln muss und keine spezifische Melodie heraussuchen kann. Eine weitere Möglichkeit zur Erweiterung wäre der Anbau eines Lautsprechers, der zum einen eine bessere Tonqualität hat, als der in dem Steuerungsmodul vorhandene und auch komplexere Tonabfolgen mit geringerer Latenz abspielen kann. Das würde ein weiteres Feld an Melodien ermöglichen. Zur weiteren Implementierung von Melodien sollte zudem eine Zusatzfunktion im Programm gegeben sein, die es ermöglicht aus fertigen Songs umgewandelte, abspielbare Melodien zu fertigen. Das wäre eine sehr aufwendige Erweiterung, die für eine größere Melodieauswahl jedoch unumgänglich wäre. Durch einen Umbau der Steuerung wäre es in Zukunft auch möglich andere Fahrmodi einzubauen, die beispielsweise eine Änderung der Fahrtrichtung in Korrespondenz mit der ausgelesenen Farbe ermöglichen.

VI. Zusammenfassung und Fazit

Die Nutzung des LEGO Mindstorms bietet einen guten Einstieg in die Roboterentwicklung. Durch die große Anzahl an kompatiblen Klemmbausteinen sind der Umsetzung verschiedenster Projekte kaum Grenzen gesetzt. Die größten Limitationen stellen die Sensoren dar, da diese zum einen nicht genau genug sind und zum anderen nur stark begrenzt in ihren Ausgabemodi sind. Daher sind komplexere Projektideen eingeschränkt, jedoch durch die Nutzung weiterer Mikrocontroller, Sensoren oder ähnlichem möglich. Die Einbindung von Matlab funktioniert gut, wenn auch häufig mit Konnektivitätsproblemen, wodurch in der Entwicklung viel Zeit für Troubleshooting eingeplant werden muss. Die größten Herausforderungen in der Entwicklung dieses Roboters, war zum einen die zuverlässige Steuerung, zum anderen der Umgang mit den Sensoren. Von der Projektidee her war das Projekt ideal um einen Einstieg in die Programmierung zu bieten, da verschiedenste grundlegende Programmierkonzepte implementiert werden mussten.

Literaturverzeichnis

[1] https://www.lfb.rwth-aachen.de/de/academics/lectures/mindstorms/
veröffentlicht von der RWTH Aachen (aufgerufen am 22.02.2024)

Manipulator

Vladyslav Karkishko, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Abstract* — Diese Forschungsarbeit zielt darauf ab, einen Roboter-Manipulator zu entwickeln, zu bauen und zu programmieren, der mit dem Steuerungsblock LEGO Mindstorms EV3 interagieren kann. Die Hauptaufgabe besteht darin, ein System zu schaffen, das in der Lage ist, Objekte mithilfe eines Sensors zu erkennen und sie mithilfe von Motoren zu bewegen. Der Softwareteil des Projekts wurde unter Verwendung der Programmiersprache MATLAB entwickelt.**

Schlagwörter — EV3, LEGO Mindstorms, mechanische Krallen, Roboter-Manipulator

I. Einleitung

Ein beladener und bewegter Roboter-Manipulator (Abbildung 1) stellt eine Lösung für eine Vielzahl von Anwendungen dar. Seine Funktionalität kann in der Logistik, der Fertigung, der Lagerhaltung und anderen Bereichen, in denen Automatisierung und effizienter Güterumschlag von größter Bedeutung sind, gefordert werden. Die Hauptidee war das Heben und Bewegen von Lasten. Es wurde jedoch Wert daraufgelegt, die Vielseitigkeit des Roboters zu erhöhen. Das war das Hauptproblem, denn die LEGO-Teile erlaubten es nicht, vielseitige Lasten zu heben, also war es das Ziel, über LEGO hinauszugehen. Durch die Entwicklung von Modifikationen und Verbesserungen wurde das Spektrum der Aufgaben, die erfolgreich von unserem Roboter ausgeführt werden können, erheblich erweitert.

Abbildung 1: Roboter-Manipulator

II. Vorbetrachtungen

Der gesamte Mechanismus der mechanischen Krallen basiert auf Sechsecken, so dass die Last mit Hilfe eines mittelgroßen Motors angehoben und abgesenkt wird. Der Roboter hat mit nur einem Sensor zur Objekterkennung, dem Touchsensor (Taste) korrekt funktioniert. Insgesamt werden drei Motoren und zwei Sensoren verwendet. Der mittlere Motor ist für das Aufnehmen und Laden des Objekts zuständig, während die beiden größeren Motoren für die Bewegung des Roboters verantwortlich sind.

A. Grundprinzip des mechanischen Krallenmechanismus

Die Idee, mechanische Krallen (Abbildung 2) zu bauen, beschränkte sich auf LEGO-Teile. In diesem Zusammenhang tauchte die Frage auf, an welchem Teil der Antriebseinheit sie befestigt werden sollten und welche Art von Last der Manipulator heben sollte. Diese Überlegungen führten zu weiteren Betrachtungen hinsichtlich der Konstruktion und Funktionalität des Systems. Es wurde dabei diskutiert, wie die Krallen effizient in das Gesamtsystem integriert werden könnten, um eine optimale Leistung zu gewährleisten. Darüber hinaus wurden verschiedene Möglichkeiten in Betracht gezogen, um die Belastbarkeit und Stabilität des Manipulators zu verbessern. Insgesamt erforderte die Umsetzung der Idee eine sorgfältige Planung und Abwägung verschiedener Faktoren, um ein funktionales und zuverlässiges System zu entwickeln.

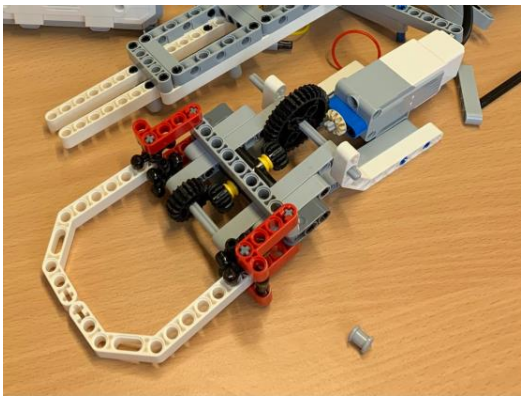


Abbildung 2: Mechanische Krallen

B. Sensortechnologie im Roboterdesign

Die Wahl der Sensoren war einer der wichtigsten Punkte, da der Ultraschallsensor (Abbildung 3) nicht verwendet werden

DOI: 10.24352/UB.OVGU-2024-034

konnte, weil die Krallen einen Schatten auf die Sensoren warfen und er nicht korrekt arbeiten konnte, und der Touchsensor konnte eine leichte Last nicht erkennen.

Abbildung 3: Ultraschallsensor

C. *Motoren im Einsatz*

Die Entscheidung bezüglich der Motorwahl für die mechanischen Krallen war entscheidend für das potenzielle Gewicht der zu hebende Last. Gleichzeitig war es nicht möglich, jeden Mechanismus am Antriebsteil des Roboters anzubringen. Aus diesem Grund wurde sich für einen Motor mittlerer Größe (Abbildung 4) entschieden.

Abbildung 4: Mittelmotor

III. ENTWICKLUNGSPROZESS

A. *Konstruktion*

Die mechanischen Krallen (Abbildung 2) setzen sich aus einem Mittelmotor zusammen, der ein ausgeklügeltes Getriebesystem antreibt. Dieses System ermöglicht eine präzise Bewegung der Krallen um 170 Grad nach oben oder unten sowie das Öffnen und Schließen zu einem genau definierten Zeitpunkt. Durch das implementierte Getriebesystem wird die Energieeffizienz des Roboters erheblich verbessert. Im Vergleich zu einem zweimotorigen Krallen, bei der ein Motor für das Anheben und der zweite für das Öffnen und Schließen zuständig ist, entfällt hier die Notwendigkeit, den zweiten Motor anzuheben. Dies trägt dazu bei, den Energieaufwand bei der Bewältigung des zu hebenden Gewichts zu optimieren. Eine Herausforderung bestand dennoch darin, dass mechanische Krallen mit nur einem Motor eine präzise Justierung des Getriebes erfordern, um die Öffnung im exakten Moment zu gewährleisten.

Als Lasterkennungssensor (Abbildung 5) wurde der Touchsensor gewählt, da er am stabilsten ist.

Abbildung 5: Lasterkennungssensor

Um die Vielseitigkeit des Roboter-Manipulators zu erhöhen, wurde ein Schwenkmechanismus integriert. Hierbei wurde der Roboter in zwei Komponenten aufgeteilt. Der erste Teil beherbergt den EV3-Stein, den Ultraschallsensor (Abbildung 3) sowie zwei große Motoren. Der zweite Teil umfasst die mechanischen Krallen und den speziellen Behälter. Beide Teile sind mithilfe von Zahnrädern miteinander verbunden, wodurch der zweite Teil des Roboters zu einem Anhänger des ersten Teils wird.

B. *Software*

Die Funktionalität des Robotermanipulators wurde mit MATLAB implementiert. Der Algorithmus (Abbildung 7) beginnt mit der Einstellung der Motorleistung und dem Starten der Motoren, um den Roboter vorwärtszubewegen. Dann tritt der Algorithmus in eine Endlosschleife ein, die bis zur Aktivierung des Touchsensors andauert. Wenn der Touchsensor aktiviert wird, stoppen die Motoren, und der mittlere Motor, der zum Anheben der Krallen um einen bestimmten Winkel verwendet wird, wird gestartet. Anschließend erfolgt eine Änderung der Motorenrichtung und ihr Start mit einer anderen Leistung, um die Krallen in die Ausgangsposition zurückzubringen. Danach werden die Bewegungsmotoren mit neuen Parametern wiederhergestellt. Wenn die Entfernung vom Ultraschallsensor zum Hindernis weniger als 50 beträgt, stoppen die großen Motoren, und der Roboter dreht sich, um das Hindernis zu vermeiden. Am Ende werden die Motorleistungen auf positive Werte eingestellt, dann wartet der Roboter 3 Sekunden, und beide Motoren stoppen. Ein Programmablaufplan zur Erklärung eines Verfolgungsalgorithmus ist in Abbildung 6 dargestellt.

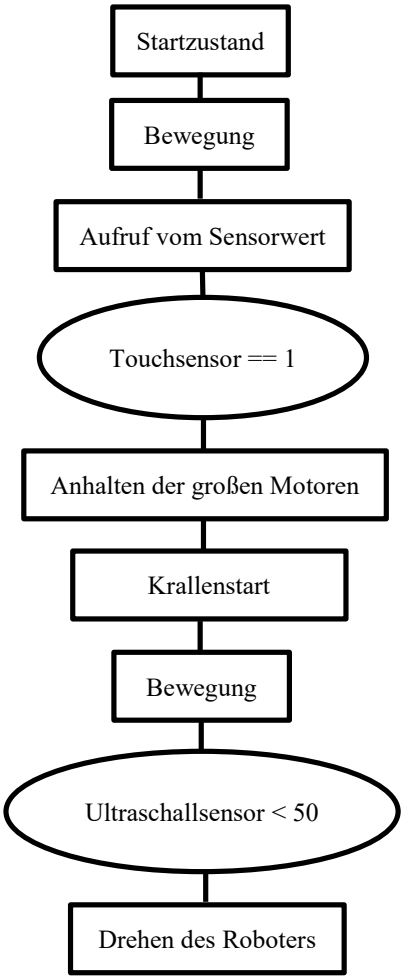


Abbildung 6: Programmablaufplan zur Erklärung eines Verfolgungsalgorithmus

IV. Ergebnisdiskussion

Nach den Tests bewies der Roboter seine Leistungsfähigkeit trotz der physikalischen Einschränkungen, die mit LEGO-Steinen verbunden sind.

Das Problem, dass MATLAB nur eine Aktion auf einmal ausführen kann, wurde durch die Implementierung von waitFor() beseitigt. Außerdem wurden alle unnötigen Teile entfernt, was die Leistung der Maschine erheblich verbesserte.

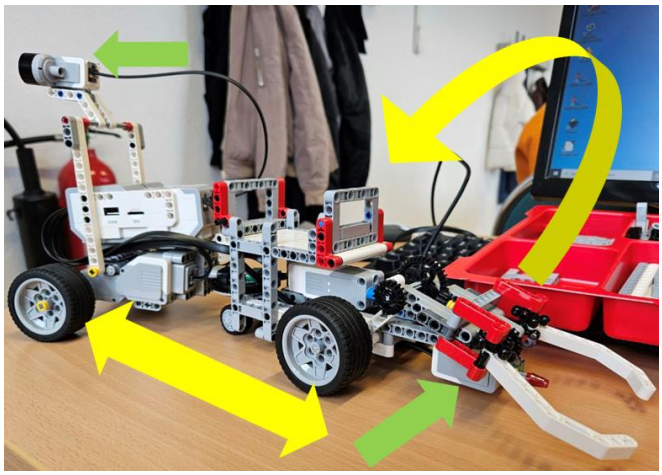


Abbildung 7: Roboter-Manipulator-Algorithmus

V. Zusammenfassung und Fazit

Zusammenfassend kann man sagen, dass alle Ziele erreicht wurden. Eine Bluetooth-Verbindung wurde ebenfalls hinzugefügt, so dass der Roboter Funktionen aus der Ferne ausführen kann. In Zukunft kann das automatische Fahrsystem verbessert und der Entladevorgang weiterentwickelt werden. Zum Beispiel könnte ein stationärer Manipulator gebaut werden, den der Robotermanipulator automatisch durch Lichtemission findet und unter ihn fährt.

Literaturverzeichnis

[1] LfB - RWTH Aachen, “Mindstorms EV3 Toolbox”, Documentation, Release v1.0, Jan 27, 2020.

„Manipulator“ mit LEGO Mindstroms

Danylo Korzh, Elektrotechnik und Informationstechnik

Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung* - Jährlich findet an der Otto-von-Guericke-Universität in Magdeburg das Projektseminar zur Elektro - und Informationstechnik statt. Im Rahmen der diesjährigen Projektwerkstatt wurde ein spezieller Mechanismus namens 'Manipulator' entwickelt. Dieser hilft, das Verletzungsrisiko bei der Arbeit zu verringern und den Prozess zu automatisieren. Für die Konstruktion und den Bau des Mechanismus 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 Mechanismus vorgestellt. Außerdem werden Probleme, die während des Entwurfsprozesses aufgetreten sind, und deren Lösungen erörtert.**

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

I. Einleitung

Die Fortschritte in der Robotertechnik bieten uns die Möglichkeit, effektiv auf eine Vielzahl von Herausforderungen zu reagieren, sei es in der Industrie, in der Medizin oder im Bereich der öffentlichen Sicherheit. Die Geschichte der Liquidator-Roboter (*Abbildung 1*) von Tschernobyl zeigt deutlich, wie Robotik und Automatisierung menschliches Leben retten und schützen können.

In Anbetracht der aktuellen Entwicklungen und der wachsenden Notwendigkeit sicherer und multifunktionaler Lösungen erscheint die Weiterentwicklung von Robotern mit verschiedenen Fähigkeiten als unausweichlich. Diese Technologie verspricht nicht nur eine erhöhte Effizienz und Sicherheit, sondern auch eine Reduzierung menschlicher Risiken in gefährlichen Umgebungen. Letztendlich liegt es an uns, diese Technologie verantwortungsbewusst zu nutzen und sicherzustellen, dass sie zum Wohl der Gesellschaft eingesetzt wird. Mit einer sorgfältigen Planung und einem klaren ethischen Rahmen können wir die Vorteile der Robotik maximieren und gleichzeitig potenzielle Risiken minimieren.

II. Vorbetrachtungen

Dieser Abschnitt fasst die wichtigsten in diesem Projekt verwendeten Tools zusammen und beschreibt ihre Funktionen in Bezug auf die Durchführung bestimmter Aufgaben oder die Reaktion auf Eingabedaten.

Programmieren von Robotern. Die Technologie besteht aus LEGO-Baukastenelementen, elektronischen Komponenten und Software für den Bau und die individuelle Anpassung von Robotern. Die neueste Version von EV3 bietet eine große Auswahl an Sensoren und Motoren sowie eine grafische Software zur einfachen Programmierung. Im Rahmen des Universitätsprojekts Otto von Guericke 2024 wurde mit EV3 ein Roboter entworfen und gebaut,

Abbildung 1: Liquidationsroboter in Tschernobyl [1]

der dank einer intuitiven und skalierbaren Plattform Aufgaben stark vereinfacht und die Entwicklung von Maschinen beschleunigt. Die neueste Version von Mindstorms EV3 bietet eine größere Funktionalität und Benutzerfreundlichkeit als sein NXT-Vorgänger, was es besonders für technische und wissenschaftliche Herausforderungen wie den Bau und die Programmierung eines Manipulators attraktiv macht.

MATLAB (Matrix Laboratory) ist eine Programmiersprache und Entwicklungsumgebung, die von MathWorks entwickelt wurde. Es bietet Möglichkeiten für numerische Berechnungen, Datenvisualisierung, Simulation und Analyse. MATLAB verfügt über Werkzeuge zur Matrixmanipulation, Signalverarbeitung und Bildverarbeitung sowie über Funktionen zur Lösung mathematischer und technischer Probleme. MATLAB kann bei der Entwicklung von Robotern für verschiedene Aufgaben eingesetzt werden, wie beispielsweise die Entwicklung und den Test von Steuerungsalgorithmen, die Simulation der Roboterdynamik, die Analyse von Sensordaten oder den Entwurf eines Videoüberwachungssystems. Es ist daher von Vorteil, Grundkenntnisse der Programmierung in MATLAB zu besitzen, um den Mechanismus erfolgreich zu bauen.

III. Konstruktion und Programmierung

A. Aufbau

Das Design des "Manipulators" (*Abbildung 5*) ist technisch einfach. Um jedoch sicherzustellen, dass ein Mechanismus in der Lage ist, Lasten zu erkennen, zu erfassen und zu manövrieren, sind Motoren und Sensoren erforderlich, um bestimmte Operationen zu erkennen und durchzuführen. Im Folgenden sind die Arten und Eigenschaftender benötigten Motoren und Sensoren aufgeführt.

DOI: 10.24352/UB.OVGU-2024-035

- *Motoren*

Drei Motoren werden benötigt, um einen Mechanismus zu bauen, von denen zwei große Motoren und ein mittlerer Motor (*Abbildung 2*) sind. Zwei große Motoren sind für das Hin- und Herfahren und das Abbiegen verantwortlich. Der mittlere Motor ist verantwortlich für das Anheben der Klaue.

Abbildung 2: großer Motor und mittlerer Motor [2–3]

- *Touch Sensor*

Abbildung 4: Touch Sensor

Der Knopf (*Abbildung 4*), der in den vorderen Teil des Mechanismus unter dem Greifer eingebaut ist, wird verwendet, um die Last zu bestimmen, die angehoben werden muss. Wenn der Knopf gedrückt wird, schließen sich die Greifer und die Last wird nach oben bewegt.

- *Ultraschallsensor*

Der Ultraschallsensor eignet sich zur Messung von Entfernungen und Abständen zu anderen Objekten. Dazu wird ein Ultraschallsignal ausgesendet und dessen Echo empfangen. Um den Abstand zum Objekt messen zu können, muss die Laufzeit des Signals gemessen und daraus der Abstand berechnet werden. Auf diese Weise wird ein Hindernis erkannt.

B. Programmierung

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 der EV3-Toolbox. 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 (*Abbildung 6*).

Beispiele für die Verwendung von MATLAB sind: `'brick.motorA.power = -30'`, um die Motorleistung zu regeln, oder `'GetUltrasonic'`, um die Entfernung mithilfe eines Ultraschallsensors zu berechnen. 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 **`if`**-Anweisungen, die bestimmte Teile des Programms nur ausführen, wenn bestimmte Bedingungen erfüllt sind. Die EV3-Einheit empfängt

Abbildung 5: fertiger "Manipulator"

programmierten Skript und einem Roboter. Das Programm wird durch Drücken der Run-Taste in MATLAB gestartet.

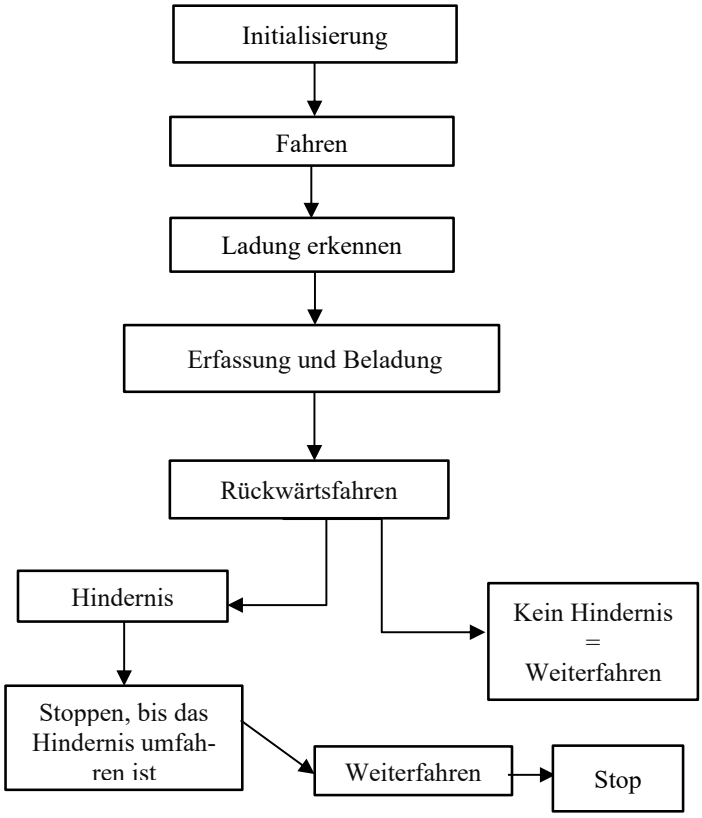


Abbildung 6: Programmablaufplan

Befehle vom Computer und leitet sie an die Motoren weiter. Sie fungiert als Übersetzer zwischen einem in MATLAB

Für ein anschauliches Beispiel, wie die Codekette und der Programmablaufplan im MATLAB (*Abbildung 7*) angeordnet sind.

```
% Check sensor++++++++
if brick.sensor1.value == 1
% stop motorA
brick.motorA.stop();
brick.motorD.stop();
%Motor B wird für kurze Zeit
entsprechend angesteuert.
brick.motorB.limitValue = 290;
brick.motorB.power = 50;
brick.motorB.smoothStart=1;
brick.motorB.start();
brick.motorB.waitFor();
pause(2);
% Verzögerung, um mehrfache
Tastenaktivierungen zu vermei-
```

Abbildung 7: Codekette

Die Sensoren im Design haben nur begrenzte Funktionen, und das Design des Roboters musste ständig angepasst werden, um sicherzustellen, dass alle Sensoren und Motoren ordnungsgemäß funktionieren. Der Ultraschallsensor befand sich zunächst unter der Klaue, die die Last aufnahm und anhob, aber es stellte sich heraus, dass der Sensor zu niedrig positioniert war und nur die Klaue oder den Tisch selbst erkennen konnte, aber nicht die benötigte Ladung. Es gab auch Probleme mit der Programmierung, aber sie wurden durch Erfahrung mit der Software gelöst.

IV. Ergebnisdiskussion

Die Erreichung des gesetzten Ziels wird bestätigt. Der Mechanismus funktioniert ohne Ausfälle; Sensoren und Motoren arbeiten synchron und präzise gemäß der Befehlssequenz. Ein Mechanismus, der eine solch zuverlässige Leistung aufweist, zeigt das Potenzial für eine erfolgreiche Integration in verschiedene Umgebungen und Anwendungen in der realen Welt. Das Ziel des Projekts wurde erreicht.

Literaturverzeichnis

[1] ***Liquidationsroboter in Tschernobyl*** (https://pkitis.tltsu.ru/?p=1038) Version 26 April 2022

[2] **Amazon:** EV3 Großer Motor https://www.amazon.com/Lego-Mindstorms-Large-Servo-Motor/dp/B00E1QDP4W

[3] **Amazon:** EV3 Mittlerer Motor https://www.amazon.de/LEGO-MINDSTORMS-Education Servomotor-45503/dp/B00E1Q5NBU

[4] **Amazon:** EV3 Touch Sensor https://raisingrobots.com/product/touch-sensor/

[5] ***Projekt Instagram:*** https://www.instagram.com/izzy17_projekt2024?igsh=MXRhcTk1Y3hrOGZ6eQ==

Manipulator

Mykyta Samiliak, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Abstract* — Diese Arbeit beschäftigt sich mit der Planung, dem Design und schließlich der Programmierung eines Robotermanipulators, der mit einem LEGO-Mindstorms-EV3-Controller interagieren kann. Das Hauptziel ist es, ein Objekt mit einem Berührungssensor zu erkennen und es mit Hilfe von Motoren und Greifern zu bewegen. Das Programm wurde mit MATLAB geschrieben.**

***Schlagwörter* — LEGO Mindstorms, EV3, Projektseminar, Manipulator, Roboter, Greifer.**

I. Einleitung

Seit langer Zeit verbessern und entwickeln die Menschen Maschinen, die sie bei ihrer harten Arbeit unterstützen. Diese Maschinen werden in der Industrie eingesetzt und helfen, Zeit, Geld und menschliche Energie zu sparen. Heutzutage sind Roboter ein unverzichtbarer Bestandteil der Industrie. Sie ermöglichen die Automatisierung von Prozessen, was eine schnelle und präzise Verarbeitung von Massenprodukten ohne Fehler gewährleistet. Eine besonders wichtige Rolle spielen dabei innerbetriebliche Transportmittel wie Pick-up-Trucks (siehe Abbildung 1). Sie steigern nicht nur die Effizienz der Arbeit, sondern beschleunigen auch den Materialfluss innerhalb des Unternehmens. Zur Erreichung dieses Ziels werden verschiedene automatisierte Transportsysteme mit Sensoren eingesetzt. Diese Systeme sind in der Lage, Waren und Güter eigenständig zu organisieren und an die benötigten Orte innerhalb des Unternehmens zu liefern. Somit nutzen Pick-up-Trucks, ähnlich wie andere robotergesteuerte Transportmittel, modernste Technologien, um ihre Aufgaben in der Industrie effizient zu erfüllen.

Abbildung 1[1]: Manipulator Auto

II. Vorbetrachtungen

Ziel des Projekts ist die Entwicklung eines Roboters, der Objekte autonom laden kann. Der Roboter soll Objekte mit Hilfe eines hexagonalen Lademechanismus aufnehmen und aufnehmen und transportieren können. Die Motivation hinter diesem Projekt ist Aufgaben im industriellen Umfeld zu automatisieren, um die Effizienz zu steigern zu steigern und menschliche Arbeitskräfte zu entlasten. Durch die Verwendung eines Objekterkennung und drei Motoren - ein kleiner Motor zum Greifen und zwei große Motoren für die Bewegung - soll eine kostengünstige und dennoch soll eine kostengünstige und dennoch effektive Lösung erreicht werden. Der Lademechanismus ist speziell für das seitliche Anheben von Objekten ausgelegt. was die Vielseitigkeit des Roboters erhöht und die Integration in verschiedene Arbeitsumgebungen ermöglicht.

A. Beschreibung des Designprozesses

Die ursprüngliche Idee war, einen Manipulator zu entwickeln und ihn auf die Maschine zu setzen. Aber wie ein Mann sagte: "Man kann nicht reinstopfen, was man nicht reinstopfen kann". Daher werden in diesem Abschnitt die Ergebnisse der Versuche vorgestellt und anschließend der unabhängig gewählte Lösungsansatz präsentiert.

Wie bereits erwähnt, funktionierte alles mit Zahnrädern, und das Hauptproblem bei der einfachen Konstruktion war die Größe der Klauen und die Frage, wo sie angebracht werden sollten.

B. Eigener Lösungsansatz

Um die Funktion und die Anwendung des Roboters zu optimieren, muss die Anzahl der Teile, aus denen die Klauen bestehen, reduziert, die Motoren richtig positioniert und die Griffigkeit verbessert werden.

III. Konstruktion

Dieser Abschnitt beschreibt die Entwicklung eines Roboters Schritt für Schritt vom Prototyp bis zum Endprodukt.

A. Mechanismus zum Greifen und Laden

Im Prinzip gibt es viele Möglichkeiten, das Greifen von Objekten zu realisieren. Es war jedoch wichtig, dass der Mechanismus nicht zu schwer ist, so dass der kleine Motor ihn bei etwa 180 Grad anheben kann. Wichtig war auch, dass er einen ausreichend großen Gegenstand aufnehmen kann. Zu diesem Zweck wurde beschlossen, einen Mechanismus zu

DOI: 10.24352/UB.OVGU-2024-036

verwenden, der das Objekt ergreift und es dank eines Getriebes (siehe Abbildung 2) anhebt.

In Abbildung 2 ist auch deutlich zu erkennen, dass der gesamte Mechanismus zum Greifen von Gegenständen durch einen kleinen Motor angetrieben wird. Das große schwarze Zahnrad ist für das senkrechte Anheben des Objekts zuständig. Kleinere Zahnräder sind für das Schließen der Klauen und für das Öffnen der Klauen bei der Rückkehr in die Ausgangsposition zuständig.

Die Bewegung der Klauen war ungenau und kehrte nicht in die Ausgangsposition zurück, was darauf zurückzuführen ist, dass die EV3-Einheit nur eine Codezeile auf einmal verarbeiten kann und die Befehle im MATLAB-Programm als separate Funktionen geschrieben wurden.

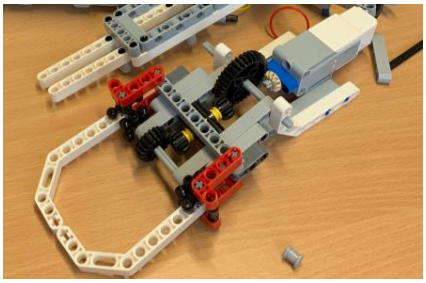


Abbildung 2: Mechanische Krallen

B. *Sensor für Objekterkennung*

Der erste Sensor, mit dem die Maschine erkennt, wann sich ein Objekt im Erfassungsbereich befindet, war ein Ultraschallsensor. Er war unter den Klauen angebracht, so dass er sie ständig sah und sehr ungenau war. Es wurde dann beschlossen, einen Farberkennungssensor zu verwenden. Sein Problem war, dass er Farben erst ab einer Entfernung von 1 cm oder weniger erkannte. Da diese Entfernung sehr kurz ist, konnten die Krallen das Objekt nicht erfassen. Schließlich wurde beschlossen, eine Taste mit einer Verlängerung zu verwenden, durch deren Betätigung der Transportvorgang gestartet wurde. Abbildung 3 zeigt ein Beispiel.

Abbildung 3: Sensor für die Objekterkennung

C. *Finale Version.*

Abbildung 4 zeigt die letzte und endgültige Version des Pickup-Tracks. Nach erfolgreicher Installation der Klauen an der Maschine wurde ein spezieller Behälter gebaut, in den das Objekt gelegt wurde. Zwei Probleme waren noch zu lösen: erstens die Installation des schwenkbaren Teils. Mechanisch war es schwierig, es zu installieren, da die gesamte Struktur ziemlich schwer und nicht drehbar war. Zweitens galt es, einen MATLAB-Code zu schreiben, da es Probleme beim Greifen des Objekts gab. Dies wurde durch die Verwendung spezieller Funktionen Abbildung 5 aus der Mindstorms EV3 Toolbox Dokumentation erreicht [1].

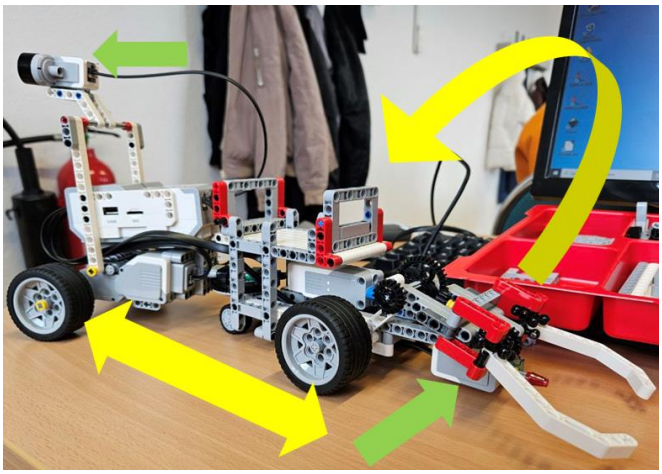


Abbildung 4: Schema der Objektaufhebung

Außerdem verfügte die endgültige Version über einen Ultraschallsensor, der für das Drehen zuständig war. Wenn der Abstand weniger als 50 cm betrug, wendete das Auto sanft nach rechts, bis der Abstand auf 50 cm erhöht wurde. Wenn der Knopf berührt wurde, schloss der kleine Motor die Klauen, die das Objekt ergriffen und sie um 170 Grad horizontal nach oben anhoben. Am Scheitelpunkt öffneten sie sich, der Gegenstand fiel in die Zelle, und eine Sekunde später kehrten sie in ihre Ausgangsposition zurück.

```
brick.motorB.smoothStart = 1;
    brick.motorB.start();
```

Abbildung 5: SmoothStart, um das Objekt zu greifen.

IV. Software

In diesem Abschnitt wird die MATLAB-Software kurz beschrieben. Die verschiedenen vom Gerät ausgeführten Operationen sind in den folgenden MATLAB-Funktionen kodiert (siehe Abbildung 6).

V. Ergebnisdiskussion

Nach zahlreichen Funktionstests erwies sich der Roboter als erfolgreich. Allerdings führte die Rechtsdrehfunktion des Roboters zu einer ruckartigen bogenförmigen Bewegung anstelle der erwarteten gleichmäßigen Drehung. Darüber hinaus stellten die physikalischen Grenzen der LEGO-Steine und des EV3-Blocks während des gesamten Entwicklungsprozesses ein ständiges Hindernis dar.

Auch dank der Smoothstart-Funktion waren die Klauen nicht scharf und griffen das Objekt sanft. Das Problem, dass MATLAB jeweils nur eine Aktion ausführen kann, wurde durch die Funktion `waitFor()` behoben. Dabei wurden alle überflüssigen Teile entfernt, was die Maschine erheblich erleichterte.

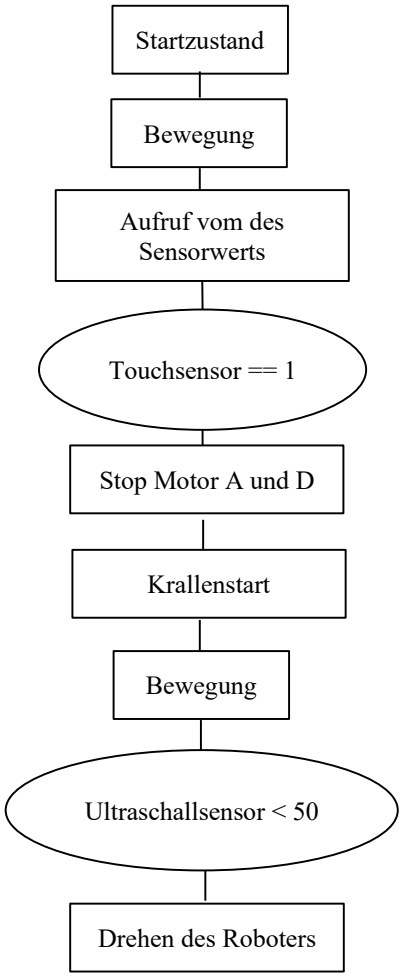


Abbildung 6: Programmablaufplan

VI. Zusammenfassung und Fazit

Zusammenfassend lässt sich sagen, dass alle Ziele mit Hilfe der MATLAB-Programmierung erreicht wurden. Die Bluetooth-Verbindung wurde ebenfalls hinzugefügt, aber um Probleme zu vermeiden, wurde die Verbindung über Kabel verwendet. In Zukunft ist es möglich, den rotierenden Teil zu verbessern und auch den Entladevorgang zu gestalten. Die Idee war, einen Magneten in das Objekt einzufügen und einen zweiten großen Magneten als separate Station zu errichten, bei dessen Annäherung das Objekt nach oben gezogen wird und das Auto entladen wird.

Anhang

```
brick.motorA.power = -30;
brick.motorD.power = -30;
brick.motorA.start;
brick.motorD.start;
% Richtung den MotorB;
motorB direction = 1;
```

Abbildung 7: Fahrtrichtung vor dem Erwerb des Ziels

```
Brick.motorB.limitValue = 170;
Brick.motorB.power = 50;

Brick.motorB.smoothStart = 1;
Brick.motorB.start();
%brick.motorB.waitFor();
Pause(2);
%Verzögerung, um mehrfache Taste
Brick.motorB.power = -60;
Brick.motorB.start();
Brick.motorB.waitFor();
Pause(1);
%Zuruck motorA
```

Abbildung 8: Der Prozess des Anhebens des Objekts und der Rückführung der Klauen in ihre ursprüngliche Position

```
if brick.sensor4.value () < 50
  brick.motora.stop () ;
  brick.motorD.stop ();
  brick.motorD.power = - 97;
  brick.motorA.power = 100;
  brick.motorA.start ();
  brick.motorD.start ();
  pause (10) ;
  brick.motorA.stop ();
  brick.motord.stop ();
  break
end
```

Abbildung 9: Drehfunktion, in einer bestimmten Entfernung

Literaturverzeichnis

[1] Der Manipulator Auto ist ein Spezialfahrzeug für das Be- und Entladen sowie den Transport von Fahrzeugen, URL: https://goo.su/6dwbh

[2] RWTH Aachen, “Mindstorms EV3 Toolbox”, Documentation, Release v1.0, Jan 27, 2020, URL : https://git.rwth-aachen.de/mindstorms/ev3-toolbox-matlab/-/blob/master/MindstormsEV3Toolbox.pdf?ref_type=heads