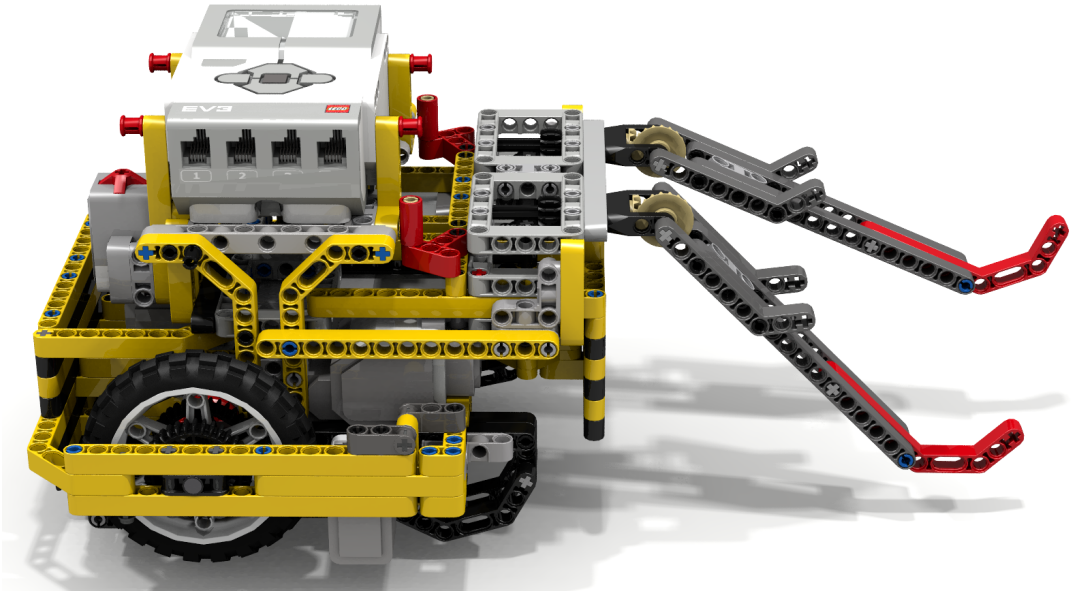
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

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

Herausgeben von: Mathias Magdowski, Thomas Gerlach und Enrico Pannicke

Band 4 vom Wintersemester 2020/2021

Inhaltsverzeichnis

Impressum

Herausgeber:	Mathias Magdowski, Thomas Gerlach und Enrico Pannicke Institut für Medizintechnik Fakultät für Elektro- und Informationstechnik Otto-von-Guericke-Universität Magdeburg Postfach 4120, 39016 Magdeburg
DOI:	10.24352/UB.OVGU-2021-028
ISSN:	2629-6160
Redaktionsschluss:	Mai 2021
Seminarzeitraum:	08.–21. Februar 2021
Bezug:	Open Access, Digitale Hochschulbibliothek Sachsen-Anhalt `http://edoc2.bibliothek.uni-halle.de/`

1. Auflage, Magdeburg, Otto-von-Guericke-Universität, 2021

Erstellung des Sammelbandes mittels LaTeX, `hyperref` und `pdfpages`

Konstruktion und Programmierung eines LEGO-Greifarmes mit automatisierter Objektsuche

Lucius Naumann, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Rahmen dieses Papers wird die Konstruktion und Programmierung eines stationären Greifroboters beschrieben, welcher Tracking- und Greifprozesse durchführen kann. Im Folgenden werden Probleme und Lösungen der Bereiche Hardware und Software präsentiert, speziell die Konstruktion und die Bildverarbeitung/ -analyse in MATLAB auf Grundlage des Blob Detection Bildanalysealgorithmus. Die Problematiken der Gewichtsverteilung und des Feintrackings erfordern das experimentelle Austesten des mit Lego Umsetzbaren. Die finale Version führt die Prozesse aufgrund der Kameraposition halbautomatisch aus, d.h. nach jedem Tracking muss die Kamera manuell entfernt werden. Der optimierte Algorithmus arbeitet mit einer Bildauswertung pro Sekunde und erlaubt schnelles und effektives Tracken. Der Greifroboter kann ein rotes Objekt auf einer festen Kreisbahn um den Mittelpunkt des Roboters suchen, anvisieren, greifen und nach einem akustischen Befehl wieder ablegen.**

***Schlagwörter*—Greifarm, Kamera, Lego, MATLAB, Motor, Suchen**

I. Einleitung

DAS Konzept des entwickelten Roboters ist für den Such- und Greifprozess optimiert. Die Aufgabe des Greifroboters besteht darin, ein rotes Objekt im Raum zu tracken und zu greifen, beziehungsweise nach einem akustischen Befehl das gegriffene Objekt abzulegen. Aufgrund der stationären Position des Greifroboters kann dieser das Zielobjekt greifen, wenn es sich auf einer Kreisbahn um den Mittelpunkt der Montierung befindet. Dabei muss der Radius der Kreisbahn der Länge des Greifarmes entsprechen. Für die Umsetzung dieser Funktionalität ist eine Bewegung des Greifroboters in zwei Freiheitsgraden notwendig. Der erste Freiheitsgrad stellt die Rotation des Greifarmes um eine vertikale Achse durch seinen Mittelpunkt dar. Die Bewegung im zweiten Freiheitsgrad ermöglicht das Heben und Senken des Armes. Neben den Motoren müssen eine externe RGB-Kamera für das Tracking, ein Farblichtsensor für die Lichtanzeige und ein Ultraschallsensor für akustische Befehle in die Konstruktion eingebunden werden. Die Trackingfunktion erfordert einen schnell arbeitenden Bildanalysealgorithmus, mit dem die zuverlässige Identifizierung einer roten Fläche im Bild möglich ist. Anschließend können aus der identifizierten Fläche Informationen über die Lage des Flächenmittelpunktes, d.h. des Objektmittelpunktes abgeleitet werden. Anhand dieses Informationsflusses sind eine schrittweise Motoransteuerung und Annährung an das zu greifende Objekt möglich. Der anschließende Greifprozess beinhaltet das Ansteuern von zwei Motoren, wobei die jeweiligen Winkel- und Leistungswerte von der finalen Konstruktion und dem Gewicht des Zielobjektes abhängig sind. Diese Werte müssen experimentell bestimmt werden.

DOI: 10.24352/UB.OVGU-2021-029

Abbildung 1. Verwendung des Canadarm2 als Stützvorrichtung für den Astronauten Stephen K. Robinson während eines Reparationseinsatzes, 2005. (Quelle: NASA, 2005 August. Web. URL https://en.wikipedia.org/wiki/Mobile Servicing System#/media/File: STS-114 Steve Robinson on Canadarm2.jpg, 19.03.2021)

II. Vorbetrachtung

Als Vorlage für den Greifroboter dient das Space Station Remote Manipulator System, kurz Canadarm2, ein Multifunktionsroboterarm an Bord der Internationalen Raumstation (ISS). Der für den Einsatz im Weltall optimierte Greifarm deckt ein vielfältiges Aufgabenspektrum von Reparationen bis zum Greifen andockender Raumkapseln ab. Der Greifarm arbeitet unter den Bedingungen des Alls hochpräzise und bewerkstelligt auch das Greifen bemannter Raumkapseln, beispielsweise der Dragon Raumkapsel des U.S. Unternehmens SpaceX. In Abbildung 1 ist der Greifarm in Aktion dargestellt.

A. Design

Der Canadarm2 Greifarm besitzt eine Länge von $17\,\mathrm{m}$ und kann bei einer Eigenmasse von $1{,}497\,\mathrm{t}$ maximal $116\,\mathrm{t}$ in der Schwerelosigkeit bewegen. Die Verwendung von Kohlefaser-Thermoplasten garantiert eine hohe Stabilität des Greifarms bei geringer Eigenmasse. Mit sieben Freiheitsgraden ähnelt der Aufbau dem eines menschlichen Armes, wobei jedes Gelenk durch eine zusätzliche Rotationsfreiheit von 540° optimiert ist (Canadian Space Agency., 2019/ Wikimedia Foundation, Inc.,

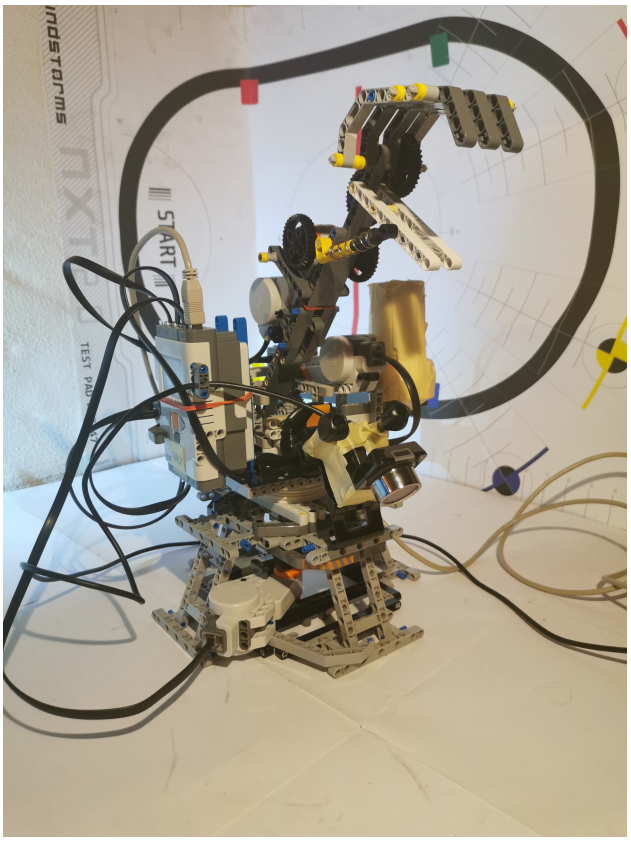


Abbildung 2. Die finale Version des Greifroboters.

2021, Februar 11). Aufgrund des Bauteillimits von 3 Motoren können am Legoroboter maximal 2 Freiheitsgrade abgedeckt werden; der dritte Motor wird für eine motorisierte Greifschaufel eingesetzt. Mit der Bewegung in 2 Freiheitsgraden ist eine Rotationsbewegung um die Basis und das Heben und Senken des Greifarmes realisierbar.

B. *Kamera und Navigation*

Insgesamt sind an dem Canadarm2 Greifarm 4 Farbkameras in unterschiedlichen Perspektiven verbaut, welche die Generierung von Tiefenrauminformationen ermöglichen. Basierend auf diesen Daten kann der Greifarm im Raum manuell navigiert werden. Die sich im Bau befindende Nachfolgerversion Canadarm3 soll basierend auf einer KI selbstständig navigieren und agieren können. Zusätzlich verfügt der Canadarm2 über Trägheitsmoment Sensoren an den Greifspitzen, um einen Tastsinn zu simulieren (CS, 2019). Aufgrund des Vorhandenseins von nur einer Kamera, stehen dem Legoroboter keine Tiefenrauminformationen zur Verfügung. Allerdings kann mit einer Kamera ein Objekt auf einer vorgegebenen Kreisbahn um den Roboter getrackt werden. Die Kreisbahn wird schrittweise abgefahren und Einzelframes analysiert, bis das gesuchte Objekt getrackt ist.

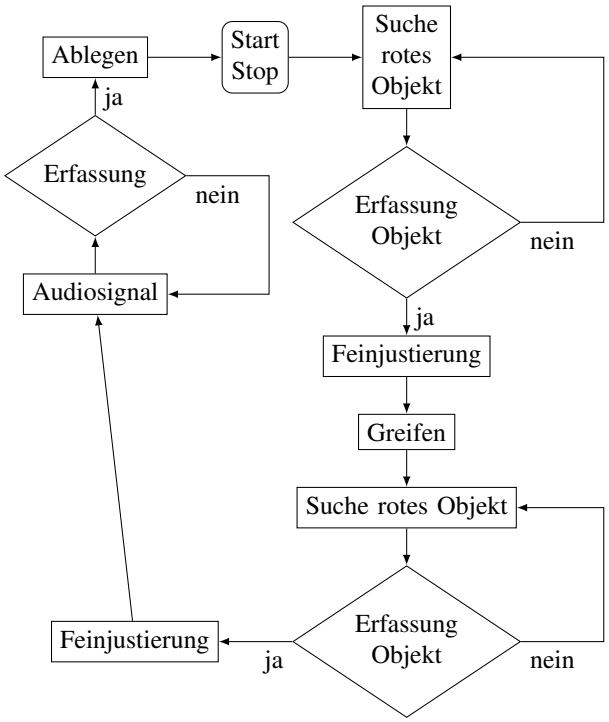


Abbildung 3. Zyklische Darstellung des Track-/ Greifprozesses in einem Programmablaufplan.

III. Hardware- und Softwarelösungen

A. *Konstruktion*

Der Greifroboter ist aus drei Segmenten aufgebaut: der Basis, dem Greifarm und dem Greifer. Die einfach motorisierte Basis ermöglicht die stabile Montierung des Greifarmes sowie die Rotation um die z-Achse. Aufgrund der Ungenauigkeit des Motors muss eine Zahnradübersetzung (beispielsweise 1:3) verbaut werden, damit für möglichst genaues Tracking auch kleine Winkelzahlen angesteuert werden können. Der Greifarm ist einfach motorisiert auf der Basis gelagert, wobei eine unsymmetrische Gewichtsverteilung durch Gegengewichte stabilisiert wird. Ein ebenfalls mit einer Zahnradübersetzung verbundener Motor ermöglicht das Heben und Senken des Greifarms. Auf dem Greifarm sitzt der einfach motorisierte Greifer auf. Um die Gewichtsverteilung zu optimieren und den auf den Greifarm wirkenden Drehmoment zu minimieren, muss der Greifmotor nah an der Aufhängung gelagert werden. Die Distanz zum Greifer kann mit Zahnrädern der Übersetzung 1:1 überbrückt werden. Die Verwendung von Zahnrädern hat den Vorteil, dass sie sich bei ungewollter Bewegung verkeilen und diese blockieren. Diese bauliche Eigenschaft ermöglicht das Anheben eines Objektes, ohne den Motor dauerhaft unter Spannung zu setzen. Die Endversion des stationären Greifroboters ist in Abbildung 2 abgebildet.

B. *Programmablauf und Sensoransteuerung in MATLAB*

Die Softwareschnittstelle zwischen MATLAB und dem NXT Commando Baustein ist eine Bibliothek der RWTH Aachen. Über diese Bibliothek können die drei verbauten Motoren, der Farbsensor (Funktion: Farblichtlampe) und der Schallsensor

(Funktion: Akustischer Befehl für das Ablegen) in MATLAB angesteuert werden. Die Abfolge der Ansteuerungen der einzelnen Sensoren wird in dem Programmablaufplan in Abbildung 3 skizziert. Signifikant ist der zyklische Aufbau des Track-/ Greifprozesses, welcher in Kombination mit der entsprechenden Hardware mehrfach und vollautomatisch ausgeführt werden kann.

C. Bildverarbeitung in MATLAB

Den ersten Abschnitt des Greifprozesses stellt die Suchfunktion eines roten Objektes dar. Dafür rotiert der Greifarm um die z-Achse und sucht auf einer vorgegebenen Kreisbahn nach dem Zielobjekt. Der Radius der Kreisbahn entspricht der Reichweite des Greifarmes. Die Unterteilung der Suchfunktion erfolgt in drei Teilbereiche: Bildgenerierung, Bildauswertung und Tracking. Die Bildgenerierung beschreibt den Prozess der Erstellung eines Einzelframes mit einer externen Kamera. Über die Software Schnittstelle Image Acquisition Toolbox (IATB) kann das Einzelframe in MATLAB genutzt werden (The MathWorks, Inc., Image Acquisition Toolbox.). Anschließend werden in der Bildauswertung aus dem Einzelframe nutzbare Informationen gewonnen. Der verwendete Algorithmus orientiert sich an dem Bildanalysealgorithmus Blob Detection. Hierbei werden gesuchte Regionen innerhalb eines Frames aufgrund spezifischer Eigenschaften identifiziert (Wikimedia Foundation, Inc., 2021, Februar 22.). Das signifikante Alleinstellungsmerkmal des zu greifenden Objektes ist die rote Farbe, über welche das Objekt von anderen Frameregionen abgegrenzt wird. Praktisch wird das RGB-Frame in ein Mono-Frame konvertiert und der R-Kanal extrahiert. Die Verwendung der Computer Vision Toolbox in MATLAB minimiert durch vorgefertigte Analysemethoden den Programmieraufwand (The MathWorks, Inc., Computer Vision Toolbox.). Darauffolgend wird der Objektmittelpunkt errechnet und als Zahlenwertpaar bestehend aus Höhe und Breit in Pixeln in MATLAB nutzbar gemacht. Das Zahlenwertpaar repräsentiert die genaue Position des roten Objektes im Frame.

D. Tracking

Im Prozess des Trackings wird der Greifarm durch Motoransteuerungen des z-Achsen Motors auf das Objekt ausgerichtet. Dafür wird die Differenz zwischen der x-Koordinate des Objektmittelpunktes und der x-Koordinate des Framemittelpunktes berechnet. Nimmt diese Differenz den Wert 0 an, sind beide Mittelpunkte aufeinander ausgerichtet. Aufgrund der parallelen Montierung der Kamera auf der Greifachse, wird somit auch der Greifarm auf das Objekt ausgerichtet. Das Objekttracking ist in zwei aufeinanderfolgende Prozesse unterteilt. Im ersten Prozess wird das Objekt grob getrackt, d.h. der Roboterarm rotiert mit einer Winkelzahl von 20° ohne Feinabstufungen solange, bis das Zielobjekt am Framerand erscheint. Anschließend wird das Objekt im zweiten Prozess fein getrackt, d.h. es findet mit einer feinen Abstufung an Winkelzahlen zwischen 10° und 2° eine Justierung mit einer Genauigkeit von bis zu 50 px statt. Die Genauigkeit beschreibt die tatsächliche Distanz zwischen Objektmittelpunkt und Framemittelpunkt.

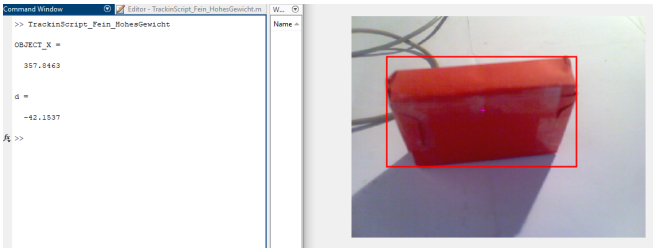


Abbildung 4. Trackinggenauigkeit von 0,1 px bei leichtem Aufbau (Kamera auf Rotationsachse, Aufbaumasse ca. 0,2 kg).

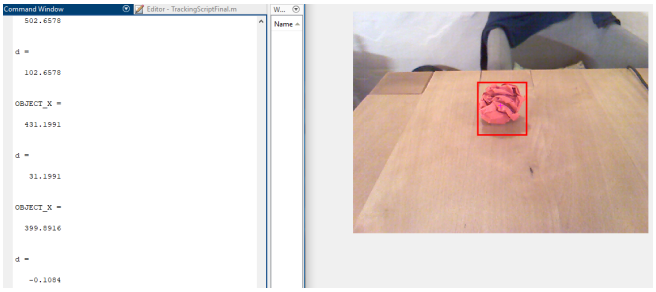


Abbildung 5. Trackinggenauigkeit von durchschnittlich 50 px bei schwerem Aufbau (Greifarm und Kamera auf Rotationsachse, Aufbaumasse ca. 1,5 kg).

Über die Differenzierung zwischen positiver und negativer x-Koordinaten Differenz kann Überdrehung korrigiert werden. Praktisch ist eine Analysegeschwindigkeit von ca. einem Algorithmusdurchlauf pro Sekunde möglich.

E. Optimierung der Trackinggenauigkeit

Die Trackinggenauigkeit ist von der Qualität der Einzelframes und Hardwarefaktoren abhängig. Probleme entstehen typischerweise beim Tracking in den Abendstunden. Aufgrund des Rotstiches trackt der Greifarm auch ungewollte orangene und braune Objekte. Mit Hilfe einer in die IATB integrierten Preview App kann der Rotstich manuell entfernt werden. Kritisch ist ebenfalls die sich bei der Rotation aufbauende Torsion in der Verbindungsachse zwischen z-Achsen Motor und Aufbau. Diese materielle Eigenschaft der Legoachse setzt der Trackinggenauigkeit ein Limit, da Winkelzahlen kleiner als 2° nicht übersetzt werden können. Die Trackinggenauigkeit ist somit abhängig von dem auf der z-Achse lagernden Gewicht und muss experimentell bestimmt werden. Eine Gewichtsoptimierung des Aufbaus erhöht die Trackinggenauigkeit. Die Dimension dieses Problems wird in den Abbildungen 4 und 5 dargestellt.

IV. Ergebnisdiskussion

Die finale Version des Greifroboters kann ein rotes Objekt auf einer festen, von der länge des Greifarmes abhängigen Kreisbahn tracken und begleitet von Lichtsignalen greifen, beziehungsweise nach einem akustischen Befehl ablegen. Somit besitzt der Roboter alle Grundfunktionen, um den Such-/ Greifprozess halbautomatisch umzusetzen. Ein relevantes Hardwareproblem ist die ungleiche und unsymmetrische Gewichtsverteilung des Greifarms auf der Basis. Die Lösung beinhaltet das

experimentelle Ausbalancieren mit Gegengewichten in Form von Metallkugeln und Steinen. Ein weiterer Lösungsansatz ist die Verwendung von Gummizügen, welche allerdings bei zu hohem Gewicht leicht reißen. Des Weiteren ist es schwierig, die ideale Zugstärke mit alltäglichen Gummibändern zu erreichen. Für einen ausbalancierten Roboter kann eine zuverlässige Software entwickelt werden. Der finale Trackingalgorithmus basierend auf dem Bildanalysealgorithmus Blob Detection arbeitet zuverlässig und sehr effektiv. Anfängliche Probleme mit suboptimalen Lichtverhältnissen können durch gezielte Optimierungen behoben werden. Insgesamt weist die finale Version kein ungelöstes Problem auf und bietet viele Erweiterungsmöglichkeiten.

Zusammenfassung und Fazit

Die Konstruktion und Programmierung eines stationären Greifroboters mit automatisiertem Objekttracking erfordert eine aufeinander abgestimmte Planung der Hardware und Software. Der Greifroboter ist aus einer Basis, einem Greifarm und dem Greifer aufgebaut, wobei jedes Segment einfach motorisiert ist. Zwecks einer feineren Ansteuerung und größerer Stabilität sind alle Motoren mit einer Zahnradübersetzung zwischen 1:1 und 1:3 gelagert. Aufgrund einer unsymmetrischen und ungleichen Gewichtsverteilung ist ein ausgeklügeltes System aus Gegengewichten notwendig, um den Greifarm auszubalancieren. Ein schneller und effektiver Bildanalysealgorithmus ermöglicht das Grob- und Feintracking des roten Zielobjektes über eine externe Kamera. Eine erhöhte Trackinggenauigkeit kann durch die manuelle Behebung von ungünstigen Lichtverhältnissen erzielt werden. Allerdings wird der Trackinggenauigkeit aufgrund der Verformbarkeit der Legoverbindungsachse zwischen Basis und Greifarm eine Grenze von ca. 50 px, d.h. 2° Motoransteuerung bei schwerem Aufbau gesetzt. Die finale Version des Greifroboters kann den Prozess nicht vollautomatisch durchführen, da die Kamera im Trackingprozess unter dem Greifarm sitzt und vor dem Greifprozess manuell entfernt werden muss. In einem weiteren Optimierungsschritt kann die Kamera extern, d.h. abseits des gesamten Roboters gelagert werden. In Kombination mit einer zweiten Kamera in einer anderen Perspektive können somit Tiefenrauminformationen gewonnen werden, welche für eine umfangreichere Navigation des Greifarmes verwendet werden können. Ebenfalls kann ein weiteres motorisiertes Gelenk an dem Greifarm eingebaut werden. Dies vergrößert den Bewegungsapparat und ermöglich komplexere Aufgaben. Für diese Anordnung muss eine neue Steuersoftware entwickelt werden, welche mehrere Motoren gegeneinander ansteuern kann. Weiterführende Probleme und Lösungen dieser optimierten Variante fallen in den Fachbereich der inversen Kinematik.

Literaturverzeichnis

[1] The MathWorks, Inc. (o.J.). *Image Acquisition Toolbox.* Abgerufen 04.03.2021, von https://de.mathworks.com/products/image-acquisition.html

[2] The MathWorks, Inc. (o.J.). *Computer Vision Toolbox.* Abgerufen 04.03.2021, von https://de.mathworks.com/products/computer-vision.html

[3] Wikimedia Foundation, Inc. (2021, Februar 22). *Blob detection.* Abgerufen 04.03.2021, von https://en.wikipedia.org/wiki/Blob_detection

[4] Wikimedia Foundation, Inc. (2021, Februar 11). *Canadarm2.* Abgerufen 04.03.2021, von https://de.wikipedia.org/wiki/Canadarm2

[5] Canadian Space Agency. (2019, Mai 20). *Canadarm, Canadarm2, and Canadarm3 – A comparative table.* Abgerufen 04.03.2021, von https://www.asc-csa.gc.ca/eng/iss/canadarm2/canadarm-canadarm2-canadarm3-comparative-table.asp

Kooperatives Roboterkonzept

Sophia Strackeljan, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Projektseminar Elektrotechnik/Informationstechnik 2021 der Otto-von-Guericke-Universität Magdeburg sollte ein Roboter konstruiert werden, der die Funktionen eines automatisierten Greifarmes übernimmt. Die nachfolgende Arbeit beschäftigt sich mit der Realisierung dieses Projektes und zeigt dabei die einzelnen Arbeitsschritte auf. Dazu gehören neben der mechanischen Konstruktion, der Programmierung und der Bildverarbeitung auch entstandene Herausforderungen und entsprechende Lösungen. Am Ende des zweiwöchigen Seminars entstand ein Greifarm, welcher autonom rote Objekte erkennt und anschließend greift beziehungsweise ablegt.**

***Schlagwörter*—Assistenzroboter, Bildverarbeitung, Greifarm, LEGO-Mindstroms, MATLAB**

I. Einleitung

ROBOTER sind in industriellen Produktionsbereichen beispielsweise in der Automobilindustrie seit vielen Jahren unverzichtbar. In den letzten Jahren wurden Roboter aber auch als Assistenzsysteme eingesetzt. Anwendungsbereiche sind hier die Unterstützung bei Montagetätigkeiten, um den Menschen von körperlich belastenden Aufgabenfeldern zu bewahren. Die Anwendungsbereiche sind vielfältig. Roboter werden auch zunehmend im Pflegebereich eingesetzt, um beispielsweise bei Umbettungen von Pflegebedürftigen zu assistieren. Die erreichbaren Positioniergenauigkeiten sind so hoch, dass es aktuell schon Roboter gibt, welche in der Lage sind, Nasenabstriche für Coronatests durchzuführen. Eine gute Übersicht über den Robotereinsatz während der Coronapandemie gibt [1]. Abbildung 1 zeigt einen Roboter, der Material auch in infektiöse Bereiche eines Krankenhauses transportieren kann.

Bei all diesen Anwendungen muss der Roboter mit einem Menschen kooperieren. Er führt also nicht wie bei der Montage eines Pkw feste Bewegungsmuster durch, sondern reagiert auf Situationen, Kommandos und muss Gegenstände unabhängig von der Lage erkennen, greifen und beispielsweise einem Menschen anreichen.

Dies war die Leitidee bei der Realisierung der Praktikumsaufgabe. Auch wenn die Möglichkeiten bei der Umsetzung eines LEGO-Roboters, der hier als Greifarm ausgeführt wird, beschränkt sind, lassen sich doch einige Grundprinzipien von assistierenden Robotern umsetzen. Dies soll im vorliegenden Paper behandelt werden.

II. Vorbetrachtungen

A. Bildverarbeitung

In der Informatik und Elektrotechnik meint der Begriff Bildverarbeitung (Image Processing) das Verarbeiten von

DOI: 10.24352/UB.OVGU-2021-030

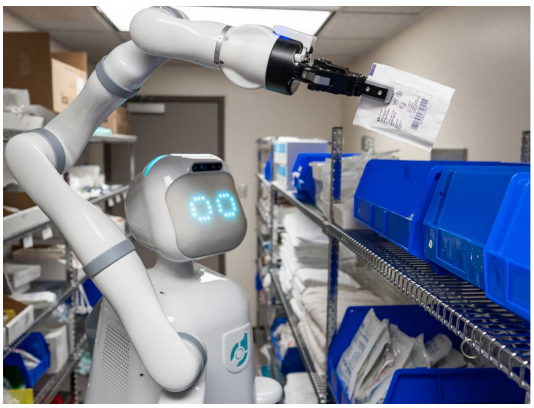


Abbildung 1. Robotereinsatz im Krankenhaus während der Covid-19 Pandemie, Foto: Diligent Robotics aus [1]

bestimmten Signalen, die zusammengesetzt ein Bild darstellen. Image Processing wird mittlerweile in sehr vielen Wissenschaftsbereichen eingesetzt.

Die Software MATLAB stellt selbst eine sogenannte Image Acquisition Toolbox zur Verfügung [2]. Mittels der darin enthaltenen Elemente und Funktionen können externe Kameras mit MATLAB verbunden, die generierten Bilder übertragen und weiterverarbeitet werden.

B. Bildverarbeitung in der heutigen Industrie

Aktuell sind bereits viele Serviceroboter mit einer Bildverarbeitung ausgestattet, da diese es erlaubt, Objekte und sogar Menschen zuverlässig zu erkennen und neue, fremde Situationen verlässlich zu analysieren. Dennoch wird aktuell nur bei einem Fünftel der Maschinen in der Industrie Bildverarbeitung eingesetzt, auch wenn dieses Feld sehr vielversprechende Ergebnisse liefert und sich in den nächsten Jahren definitiv ausbauen wird [3]. Durch die Bildverarbeitung wird die Anpassungsfähigkeit und Variabilität der Maschinen erhöht, die gerade auch in Hinsicht des maschinellen Lernens eine große Rolle spielt. Von besonderem Wert ist dabei natürlich auch die Objekterkennung.

C. Eigene Herangehensweise

Der Roboter sollte mithilfe eines LEGO-Mindstorms-Set gebaut und über den NXT-Stein angesteuert werden, welcher sowohl den Mikrocontroller beinhaltet, als auch über vier Sensoren- und drei Motorenanschlüsse verfügt. Über die Einbindung in eine Toolbox, welche von der RWTH Aachen frei zugänglich bereitgestellt wird, lässt sich der NXT-Stein über die Software MATLAB ansteuern [4]. Mittels verschiedener Befehle ist es so möglich, alle Sensoren und

Motoren anzusprechen und die aktuellen Messwerte auszulesen. Das Lego-Set stellt dabei verschiedene Sensoren zur Verfügung, unter anderem einen Ultraschallsensor für die Distanzmessung, einen Geräuschsensor und einen Farbsensor, welcher zum einen Farben misst und ausgibt und zum anderen als Farblampe dient. Allerdings ist die Funktionalität eines Roboters basierend lediglich auf diesen Sensoren beschränkt. Dies gilt sowohl für die aktive Steuerung des Roboters durch Kommonados, die Erkennung von Objekten und das sichere Greifen und Ablegen nach Aufforderung. Dieses sind allerdings Basisfunktionen eines kooperativen Robotersystems, so dass eine externe Kamera zur Bildverarbeitung integriert wurde.

Das Programm MATLAB erlaubt die Einbindung externe Kameras über die Verbindung eines USB-Ports eines Computers. Auch die Einbindung in den Programmcode ist recht einfach zu realisieren. Genutzt wurde die Philips Webcam SPC1330NC/00 als Farbkamera und die frei verfügbare Toolbox für MATLAB. Die Kamera ist im Winkel verstellbar und erstellt ein 800×600 Pixel großes Bild, dessen Bildmitte folglicherweise bei 400×300 liegt.

Damit stehen alle Elemente zur Verfügung, um die definierte Aufgabenstellung zu erfüllen.

1) Erkennen eines roten Gegenstands (Objekt) unabhängig von dessen Position auf einer Tischplatte, solange sich der Gegenstand in der Reichweite des Roboterarm befindet.
2) Stoppen der Drehbewegung, wenn ein Objekt erkannt und Roboterarm exakt positioniert wird.
3) Absenken des Greifarms und Öffnen des Greifers.
4) Greifen des Objektes und Anheben des Roboterarms.
5) Fortsetzung der Drehbewegung bis zum erneuten Erkennen der Farbe Rot.
6) Absetzen des Objektes nach einem Audiosignal.

III. Realisierung

A. Konstruktion

Für die mechanische Konstruktion des Roboters standen mehrere LEGO-Mindstorms-Sets zur Verfügung inklusive eines NXT-Steines, drei Motoren und verschiedenen Sensoren. Damit mittels der Legosteine ein funktionsfähiger Roboter entstehen kann, ist ein robustes Grundgerüst unentbehrlich. Hierdurch kann später sichergestellt werden, dass der schwere Greifarm mitsamt des NXT-Steins und den weiteren Motoren zuverlässig die Drehbewegung ausführen kann. Im Grundgerüst inkludiert ist der erste Motor, welcher für die Rotation des gesamten Roboter-Greifarmes verantwortlich ist. Oberhalb des Grundgerüstes befinden sich alle beweglichen Komponenten, wobei der Drehkranz die Verbindung zwischen dem fest stehendem Grundgerüst und den rotierenden Komponenten bildet . Zentrales Element ist dabei vor allem der Roboterarm, welcher durch einen Motor mithilfe von 8 Zahnräder angetrieben wird und in der Folge die Senk- und Hebbewegung ermöglicht. Auf dem Roboterarm ist der dritte Motor lokalisiert, welcher über eine erneute Übersetzung das Öffnen und das Schließen des Greifers ermöglicht. Somit werden durch die drei NXT-Motoren je ein Freiheitsgrad bedient und sichergestellt, das eine flexible Bewegung des Roboters möglich ist und in seinem Radius alle möglichen Punkte zum Greifen erreicht werden können (siehe Abbildung 2).

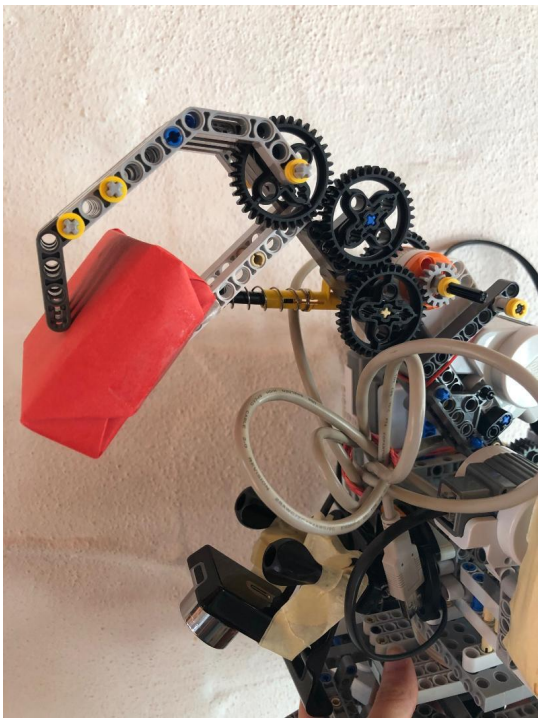


Abbildung 2. Greifarm mit aufgenommenem Objekt und Kamera

Das Gewicht des Greifarms stellte sich als großes konstruktives Problem heraus. Der gesamte rotierende Teil lässt sich nur über eine sehr weiche Welle an den Motor des Grundgerüstes anbinden (siehe Abbildung 3). Aus diesem Grund wurde versucht, den Greifarm möglichst gewichtssparend und mit einer tiefen Schwerpunktslage zu konstruieren. Mehrere Gegengewichte sorgen für einen Ausgleich zum schweren Gewicht des NXT-Steins und des Greifarmes. Des Weiteren wurde ein Farbsensor eingebaut, welcher bei Heb- und Senkbewegung ein grünes Licht und beim Greifprozess ein rotes Licht auswirft. Der Geräuschsensor wurde am hinteren Teil des Roboters angebaut und ist in der Lage, die Intensität von Geräuschen zu messen.

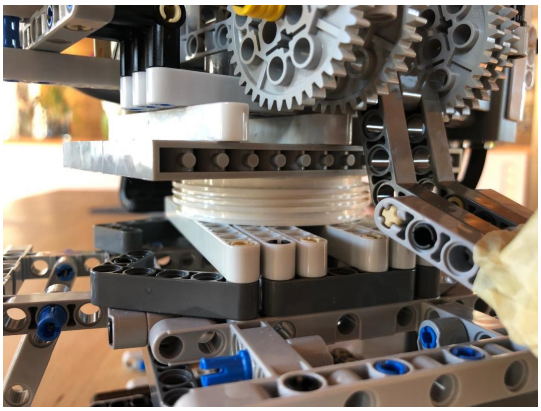


Abbildung 3. Grundgerüst (unten) und Drehkranz

Die Kamera wurde möglichst tief oberhalb des rotierenden Drehkranz montiert und lag somit unterhalb des Greifarmes. Diese Anordnung wurde gewählt, weil die externe Kamera einen sperrigen Aufbau hat und ein Anbringen der Kamera direkt am Greifarm die Schwerpunktlage noch ungünstiger gestaltet hätte. Dies führt zu einem geringfügigen Handlingsproblem, da nun mechanisch kein voll automatisierter Prozess möglich war und die Ansteuerung von Bildverarbeitung und Motorbewegung separat erfolgen musste.

B. Programmablauf

Der gesamte Ablaufplan des Codes ist in Abbildung 4 dargestellt und gliedert sich in grob drei Schritte. Als erstes wird das Tracking des roten Gegenstandes gestartet, welches mithilfe der Kamera und entsprechender Bildverarbeitung nach einem roten Objekt sucht. Wurde dies erfolgreich durchgeführt, startet die zweite Sequenz des Programms, und zwar die Feineinstellung auf das rote Objekt, am Ende dessen schlussendlich ein Greif- beziehungsweise Ablege-Prozess steht.

Durch die von der RWTH Aachen zur Verfügung gestellte NXT-Toolbox, konnten die an den NXT-Stein angeschlossenen Motoren und Sensoren über das Programm MATLAB angesteuert und ausgelesen werden.

C. Bildaufnahme und Bildanalyse

Die Bildverarbeitung wurde mithilfe sogenannter BLOBs (Binary Large Objects) durchgeführt. Dabei werden mehrere nebeneinanderliegende Pixel mit ähnlichen beziehungsweise gleichen Helligkeits- oder Farbwerten zu einer größeren Gruppe zusammengefasst [5]. Dieses Verfahren funktioniert lediglich, wenn das zu erkennende Objekt diese Eigenschaften in geschlossener Form aufweist, weshalb es sich anbot, als Objekt eine Box auszuwählen. Die Bildverarbeitung analysiert jedes aufgenommene Bild auf die Farbe Rot, genauer gesagt auf einen roten Gegenstand. Dabei wird das Farbbild in ein monochromes Bild umgewandelt und anschließend der Rotkanal extrahiert, um rote Objekte identifizieren zu können. Um den entsprechenden Gegenstand wird dann eine sogenannte Bounding Box erstellt und der Mittelpunkt mithilfe eines Fadenkreuzes fixiert (siehe Abbildung 5). Wird im aufgenommenen Bildausschnitt kein Objekt gefunden, dreht der Motor den Greifarm und die Kamera. Ein neuer Bildausschnitt wird erfasst und anschließend analysiert. Dies erfolgt solange, bis ein rotes Objekt im Frame erkannt wird, woraufhin das Skript eine 1 ausgibt, abbricht und die Feinjustierung startet. Dabei wird mit kleineren Gradzahlen bei der Umdrehung gearbeitet und ständig der Abstand zwischen der vertikalen Bildmitte und dem gefundenen Objekt gemessen. Da es sich bei dem Bildausschnitt um ein 800×600 Pixel großes Bild handelt, liegt der Bildmittelpunkt bei 400×300 Pixeln. Der Gegenstand muss allerdings nicht genau auf der Bildmitte liegen, sondern lediglich auf der vertikalen Achse, die durch den Bildmittelpunkt geht. Deshalb wird der ständige Vergleich von dieser Achse mithilfe der x-Koordinaten solange vollzogen, bis die Koordinaten im Toleranzbereich liegen. Liegen die Werte im Minusbereich, dreht sich der Motor nach rechts, handelt es sich um einen positiven Abstand, erfolgt die Drehung nach links. Ist die

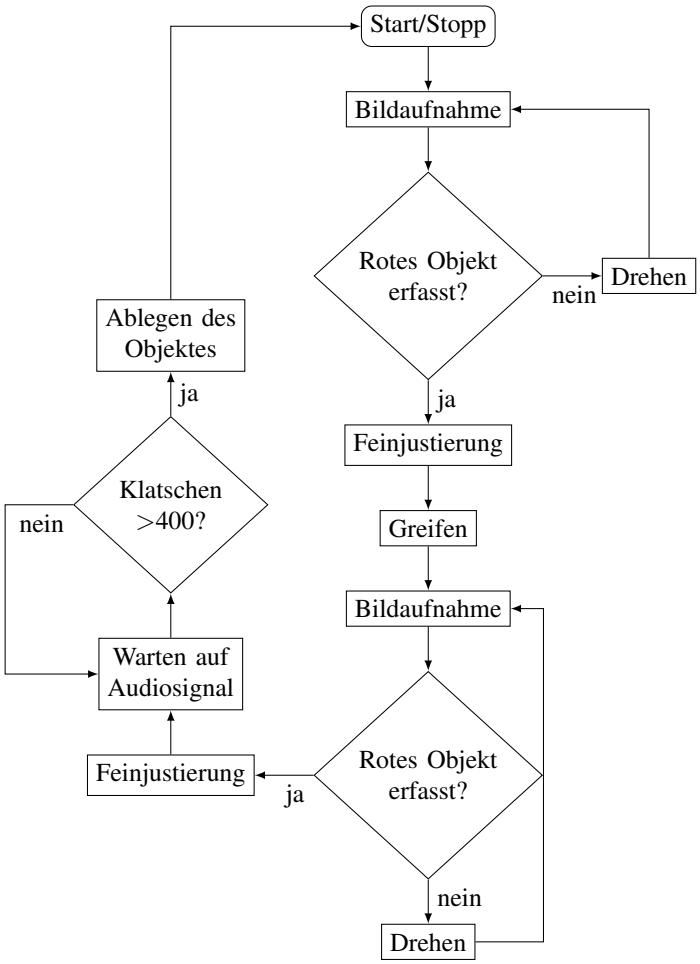


Abbildung 4. Programmablaufplan

genaue Ausrichtung erfolgt und der Abstand liegt im vorher definierten Toleranzbereich, bricht auch dieses Skript ab und startet die Motoransteuerung für den Greifprozess.

Für den Fall, dass die Bildverarbeitung zwei rote Objekte erkennt, wird kein Abstand gemessen und der Roboter dreht sich solange weiter, bis er nur einen Gegenstand im Bildausschnitt identifiziert.

Abbildung 5. Bildverarbeitung: Erkennung des zugreifenden Objektes

IV. Ergebnisdiskussion

Am Ende des Projektseminares konnte der Roboter die definierte Aufgabenstellung eigenständig erfüllen. Der Greifarm wurde so konstruiert und programmiert, dass er mittels einer integrierten Kamera und anschließender Bildverarbeitung in MATLAB seine Umgebung nach roten Objekten abscannen kann und sich solange neu ausrichtet, bis er einen entsprechenden Gegenstand in seinem Bildausschnitt detektiert. In der Folge wird die Feinjustierung gestartet und der Greifarm richtet sich mittig auf das Objekt aus, um es anschließend greifen zu können. Nach einer weiteren Analyse der Umgebung wird der Körper auf der nächsten rot markierten Stelle nach einem lauten Klatschen abgelegt. Die Bildverarbeitung arbeitet dabei präzise und kann bis auf den vorgegebenen Toleranzbereich genau die Position des Objektes feststellen. Je schwerer der Roboter allerdings ist, desto schwieriger wird eine genaue Ausrichtung, verschuldet durch die Ungenauigkeit der NXT-Motoren. Es war jedoch möglich, die Genauigkeit auf bis zu $-0{,}1$ Pixel zu programmieren, als nur die Kamera auf dem Grundgerüst montiert war. Mit dem schweren Greifarm lag der Toleranzbereich bei +/- 50 Pixeln. Deshalb bewegte sich der Greifarm teilweise etwas ruppig auf dem Grundgerüst, konnte aber trotz dessen eine Ausrichtung auf das rote Objekt sicherstellen. Des Weiteren muss ein neutraler Hintergrund gewählt werden, da ansonsten auch für den Menschen nicht sichtlich rote Umfelder als solche erkannt werden. Aufgrund der mechanischen Konstruktion mussten die Programmabläufe in zwei Schritten erfolgen und die Kamera zwischenzeitlich demontiert werden, da ansonsten ein Senken beziehungsweise Heben des Greifarmes nicht möglich gewesen wäre.

Weiteren Funktionen des Roboters standen unter anderem die Ungenauigkeiten der NXT-Sensoren und die Bearbeitungszeit im Wege. Hilfreich für einen kooperative arbeitenden Roboter wäre es gewesen, zusätzlich den Ultraschallsensor, welcher Distanzen bis zu $2{,}55\,\mathrm{m}$ messen kann, zu verwenden, da somit weitere Funktionen abgedeckt hätten werden können. Dieser Sensor arbeitete allerdings nicht immer zuverlässig und ist somit bei einem autonomen Roboter nicht zu empfehlen. Hinzukommt, dass durch die Bildverarbeitung eine viel genauere Ausrichtung möglich ist, welche nahezu immer präzise arbeiten kann. Des Weiteren stand die Idee im Raum, den Assistenzroboter per simpler Spracherkennung zu steuern. Allerdings reagiert der Geräuschsensor entsprechend seiner Funktion nur auf unterschiedliche Lautstärken. Eine Differenzierung zwischen verschiedenen Befehlen ist daher nur über den Geräuschpegel und verschiedene Silbenbetonungen möglich. Über die Lautstärke z. B. Pegelwerte lassen sich allerdings keine Kommandos zuverlässig an den Roboter übermitteln. Bei der Drehung des Roboters oder der Änderung des Abstandes zwischen der Bedienerin und dem Sensor ändert sich die Lautstärke, unabhängig vom eigentlichen Kommando. **??** zeigt die Werte des Geräuschsensors beim Sprechen des Kommandos „Wally Start" (ca. bei $20\,\mathrm{s}$) und „Stopp" (ca. bei $35\,\mathrm{s}$).

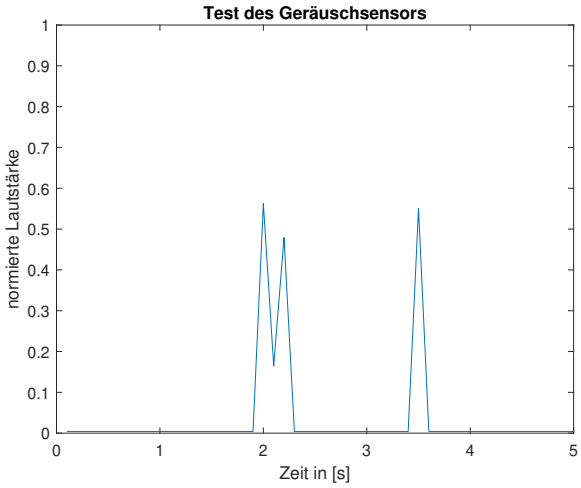


Abbildung 6. Auslesung des Geräuschsensors: 'Wally Start, Stopp'

V. Zusammenfassung und Fazit

Zusammenfassend lässt sich festhalten, dass im Projektseminar Elektrotechnik/Informationstechnik 2021 ein Roboter in Form eines Greifarmes gebaut werden konnte, welcher mithilfe einer externen Kamera seine Umgebung analysieren kann. Der Greifarm ist in der Lage, durch entsprechende Bildverarbeitung rote Objekt in seiner Umgebung zu suchen und sich auf diesen Gegenstand mittig auszurichten. Dadurch ist es dem Greifarm möglich, diverse rote Körper aufzugreifen und durch erneute Analyse der Umgebung an einem zweiten roten Platz abzulegen. Das Abstellen des roten Gegenstandes erfolgt erst durch ein Audiosignal, welches in diesem Projekt in Form eines Klatschens erfolgte. In Zukunft können noch einige Verbesserungen vorgenommen werden, wie zum Beispiel die Platzierung der Kamera, wodurch es im finalen Ablauf leider nur zur einer Vorführung der einzelnen Programmsequenzen und nicht eines voll automatisierten Programmablaufes kommen konnte. Weiterhin wäre es ein Wunsch gewesen, neben der Kamera am Greifarm eine weitere beliebig im Raum aufzustellen und somit die Kameraperspektive zu ändern und eine präzisere Lokalisierung der Gegenstände zu ermöglichen.

Durch das bereits gelungene Einbinden der Bildverarbeitung wäre es zukünftig leicht realisierbar, z. B. die Farbe der gesuchten Gegenstände zu ändern oder sogar eine Gesichtserkennung in den Programmablauf einzubinden. Durch sehr vielfältige Anwendungsmöglichkeiten stehen dem hier gebauten Roboter noch sehr viele verschiedene Funktionen und Einsatzgebiete offen.

Literaturverzeichnis

[1] Erico Guizzo, Randi K.: *How Robots Became Essential Workers in the COVID-19 Response*. Version: 30.09.2020. https://spectrum.ieee.org/robotics/medical-robots/how-robots-became-essential-workers-in-the-covid19-response

[2] MATHWORKS: *Image Acquisition Toolbox*. https://www.mathworks.com/products/image-acquisition.html

[3] Richard Bormann, Dr.-Ing. Werner K.: *Mit Bildverarbeitung schauen Roboter genau hin*. https://www.elektrotechnik.vogel.de/mit-bildverarbeitung-schauen-roboter-genau-hin-a-640290/. Version: 05.09.2017

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

[5] *Binary Large Object*. Version: Abgerufen am 24.02.2021. https://de.wikipedia.org/wiki/Binary_Large_Object

Autonom fahrender Aufklärungsroboter

Felix Alexander Kolodziej, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Abstract*—Wir haben uns ihm Rahmen des Legopraktikums 2021 dem Thema der maschinellen Unterstützung in menschenfeindlichen Gefahrensituationen angenommen und uns dafür entschieden einen autonom fahrenden Aufklärungsroboter zu entwickeln, der in der realen Anwendung beispielsweise in einem eingestürzten Gebäude eingesetzt werden könnte. In dem folgenden Paper geht es um die Konstruktion und den programmiertechnischen Hintergrund eines solchen Roboters, der in der Lage ist, nach der autonomen Absolvierung eines Parkours, eine Grundlage für eine Einsatzplanung in diesem unbekannten Gebiet auszugeben und somit eine Koordination eines Einsatzes zu ermöglichen, ohne das sich zuvor ein Mensch dieser Gefahr aussetzen musste.**

***Schlagwörter*—Aufklärungsroboter, autonomes Fahren, Bergung, Kettenfahrzeug, Lego-Mindstorms, NXT .**

I. Einleitung

Autonome Systeme begleiten heutzutage jeden im Alltag, ob es die automatisch regelnde Heizung ist oder die Jalousien, die sich runterfahren, wenn es dunkel wird. Vor Allem spielt diese neue Technologie eine große Rolle in der Fahrzeugentwicklung, von den ersten Einparkhilfen bis zu vollständig autonom fahrenden Autos. Diese Assistenzsysteme sind heutzutage nicht mehr wegzudenken. Allerding können autonome Roboter nicht nur als Alltagsbereicherung eingesetzt werden. Es ist auch sinnvoll diese in Situationen anzuwenden, die für Menschen zu gefährlich wären. So würde ein autonom fahrender Bergungsroboter verhindern, dass Menschen sich in ein eingestürztes Gebäude begeben müssen, wobei sie in Lebensgefahr geraten könnten. Der autonome Roboter unterstützt hierbei den Menschen und verhindert, dass sich dieser einer Gefahrensituation aussetzen muss.
Das Ziel unseres Projektes war es eine entsprechende Gefahrensituation zu simulieren und einen Roboter zu entwickeln, der in der Lage ist einen Parkour autonom zu absolvieren und ortsfesten Hindernissen auszuweichen. Außerdem soll er in der Lage sein in dem Parkour Opferstellen und Gefahrenstellen zu erkennen und entsprechend darauf reagieren. Den Gefahrenstellen soll der Roboter ausweichen und bei Opferstellen soll der Roboter mit einem Ton reagieren. Im Anschluss sollte der Roboter dann eine Karte von dem Parkour erstellen, sodass man diesen im Nachhinein nachvollziehen kann.

DOI: 10.24352/UB.OVGU-2021-033

II. Vorbetrachtungen

A. Grundtypen von Bergungsrobotern

In der realen Anwendung von Bergungsrobotern unterscheidet man zwei Grundtypen. Zum einen Aufklärungsroboter die zur Aufklärung beschädigter bzw. einsturzgefährdeter Strukturen dienen. Dabei liegt der Fokus auf der Inspizierung der Umgebung, um ein Lagebild zu erstellen mit dem die Einsatzplanung und Koordination effektiv umgesetzt werden kann. Zum anderen gibt es die nächst anspruchsvollere Stufe, den Bergungsroboter. Dabei liegt der Fokus auf der Bergung von Opfern, also dem autonomen Transport von Verletzten aus dem Gefahrenbereich. Bei dieser Technologie arbeitet man mittlerweile an Robotern, die in der Lage sind direkt vor Ort erste Hilfe zu leisten, bevor die Bergung erfolgt. In dem Projekt wurde sich für die erste Anwendung, also für die Realisierung eines Aufklärungsroboters entschieden.

B. Umsetzung

Um die reale Anwendung für das Projekt zu realisieren soll sich der Roboter autonom durch den Parkour bewegen, dabei soll er sich immer an der rechten Wand orientieren um so letztendlich wieder am Start des Parkours anzukommen. Über Sensoren wird also die ganze Zeit der Abstand zu den entsprechenden Wänden bestimmt an denen sich orientiert wird. Entsprechend muss der Roboter dann auf diese Abstandswerte reagieren. Dabei soll der Roboter bei zu großen oder kleinen Werten seine Bewegung ausgleichen, in dem er sich auf der Stelle dreht. Dies realisieren wir, indem sich die beiden Räder, die die Ketten antreiben, mit der selben Leistung in verschiedene Richtungen drehen, solange bis der Abstandswert wieder vernünftig ist. Einen sehr kleinen Abstandswert soll der Roboter als Linksabbiegung interpretieren und sich entsprechend drehen. Einen sehr großen Abstandswert soll der Roboter als eine Rechtsabbiegung interpretieren, dabei reicht es nicht sich an der Wand zu orientieren, da er sonst dort verkantet, es ist also wichtig zuvor nochmal ein Stück nach vorne zu setzen. Um auf spezielle Stellen im Parkour zu reagieren arbeiten wir mit einem Farbsensor, der diese farbig gekennzeichneten Stellen erkennt und entsprechend eine Reaktion hervorruft.

C. Fuzzylogik

In der normalen Logik unterscheidet man zwischen zwei Zuständen, 0 als falsch und 1 als wahr. Diese beiden Fälle reichen allerdings nicht aus um eine autonome Bewältigung des Parkours umzusetzen, denn es könnte lediglich zwischen einer unpassenden Entfernung und einer passenden Entfernung unterschieden werden wobei der Roboter nicht weiß wie er auf die unpassende Entfernung reagieren soll, da diese sowohl zu weit als auch zu kurz sein könnte. Oder er erkennt nur ob er zu

weit oder zu kurz entfernt ist, dann wäre er allerdings ununterbrochen damit beschäftigt seine Position auszugleichen und hat keinen Modus, in dem die passende Entfernung zur Weiterfahrt erreicht wurde. Die Lösung für diese Problematik ist die Fuzzylogik, diese wurde zur Modellierung von Unschärfen entwickelt. Grundlage der Fuzzylogik sind dabei die unscharfen Mengen. Das sind Mengen, die nicht nur durch ihre Elemente definiert sind, sondern über den Grad ihrer Zugehörigkeit zu dieser Menge.
Über eine Zugehörigkeitsfunktion wird jedem Element der Definitionsmenge ein Wert aus dem Intervall [0,1] zugeordnet, welcher den Zugehörigkeitsgrad des Elements zur unscharfen Menge angibt. Damit wird jedes Element zum Element jeder unscharfen Menge, aber mit jeweils unterschiedlichen, entsprechend zugeordneten Zugehörigkeitsgraden für die jeweilige Teilmenge. Verknüpft mit logischen Operationen ist man nun in der Lage, durch die Bildung von Schnittmengen und Vereinigungsmengen, nicht nur die logischen Ränder, sondern auch Zwischenwerte zu betrachten. Auf unser Beispiel bezogen kann man also nicht nur zu große oder zu kleine Abstände berücksichtigen, sondern auch entsprechend auf ein bisschen zu kurzen Abstand oder einen sehr kurzen Abstand angemessen und verschieden reagieren. Dies ist für die Absolvierung des Parkours sehr wichtig, da man unterscheiden muss ob gerade nur eine leichte Abweichung bei einer geraden Fahrt korrigiert werden soll oder eine Abbiegung erfolgen müsste. Wir unterscheiden später zwischen 5 verschiedenen Fällen. Der erste Fall tritt ein, wenn der Roboter stark zu weit nach rechts orientiert ist und erfordert eine entsprechend starke Linksdrehung. Der zweite Fall tritt ein, wenn der Roboter ein bisschen zu weit nach rechts orientiert ist und erfordert dabei nur eine leichte Linksdrehung. Der dritte Fall ist die anzustrebende Ausrichtung, dabei ist der Roboter weder zu weit nach links, noch nach rechts orientiert, sodass er einfach geradeaus fahren soll. Die anderen beiden Fälle entsprechen einer Orientierung die viel zu weit nach links ist oder nur ein bisschen zu weit nach links und entsprechend erfolgt analog zu Fall eins und zwei eine starke oder eine leichte Rechtsdrehung.

III. Realisierung

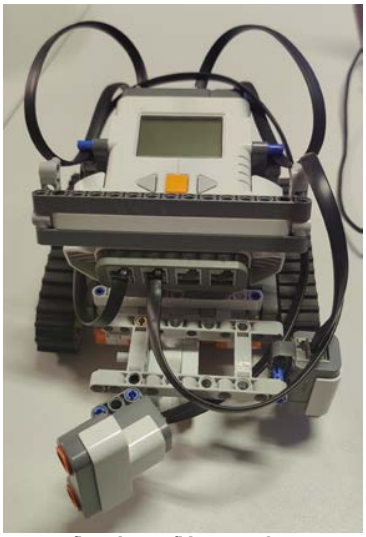


Abbildung 1: Aufbau des Aufklärungsroboters mit Sensoren

A. Aufbau

Das Grundgerüst besteht aus zwei verbundenen NXT-Motoren, auf denen der NXT-Stein befästigt wurde. Dabei war es wichtig, dass die Konstruktion stabil ist und keine wesentlich größere Fläche einnimmt, als der NXT-Stein, da so bei der Fahrt und bei der Drehung weniger Raum benötigt wird und der Roboter auch schmalere Stellen des Parkours absolvieren kann. Es wurde auch darauf geachtet die Kabel möglichst im Inneren oder direkt am Rand des NXT-Roboters langzuführen, sodass auch dadurch keine Behinderungen auftreten. Wie in Abbildung 1 erkennbar sind vorne an einem Gerüst die verwendeten Sensoren angebracht. Zum einen ein Ultraschallsensor, der im 45° Winkel nach vorne ausgerichtet ist und für die Abstandsmessung benötigt wird und ein Farbsensor, der unmittelbar vor der linken Kette angebracht wurde und nach unten gerichtet ist um dort Farbwerte zu erkennen.
Oberhalb des NXT-Steins befindet sich eine Halterung, in der ein Smartphone zur Videoaufnahme befästigt werden kann. Der Antrieb des Roboters erfolgt über zwei Ketten, die jeweils an der vorderen Achse von einem NXT-Motor angetrieben werden. Dabei laufen die Motoren getrennt voneinander, sodass eine Drehung der beiden Ketten in jeweils verschiedene Richtungen möglich ist und sich der Roboter somit auf der Stelle drehen kann. Zudem ist ein Kettenfahrzeug in der Lage verschiedene Untergrundtypen zu befahren und auch kleinere Unebenheiten im Boden zu bewältigen.

B. autonomes Fahren

Die autonome Absolvierung eines Parkours durch unseren Roboter haben wir mit Hilfe des Ultraschallsensors realisiert. Dabei orientiert sich der Roboter grundsätzlich an der rechten Wand und soll mit Hilfe permanenter Abstandsmessung eine

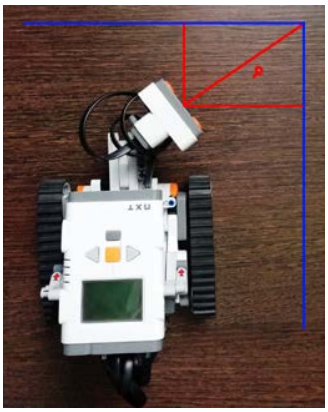


Abbildung 2: Ausrichtung des Ultraschallsensors

schräge Ausrichtung korrigieren. Diese tritt auf durch Abweichungen in den Bewegungen der Motoren und Ungenauigkeiten bei der Drehungen des Roboters. Durch eine Ausrichtung des Ultraschallsensors in einem 45° Winkel vor dem Fahrzeug, wie man in Abbildung 2 sehen kann, ist es möglich sowohl den Abstand zur rechten Wand, als auch den Abstand zu einer vorderen Wand zu bestimmen und entsprechend eine Kollision mit beiden zu verhindern. Ebenso ist dies wichtig, damit der notwendige Übergang zu einer neuen „Orientierungswand“ erfolgen kann. Denn bei dem Drehungsvorgang nach links ist nun die Wand, die vor dem Roboter liegt, die

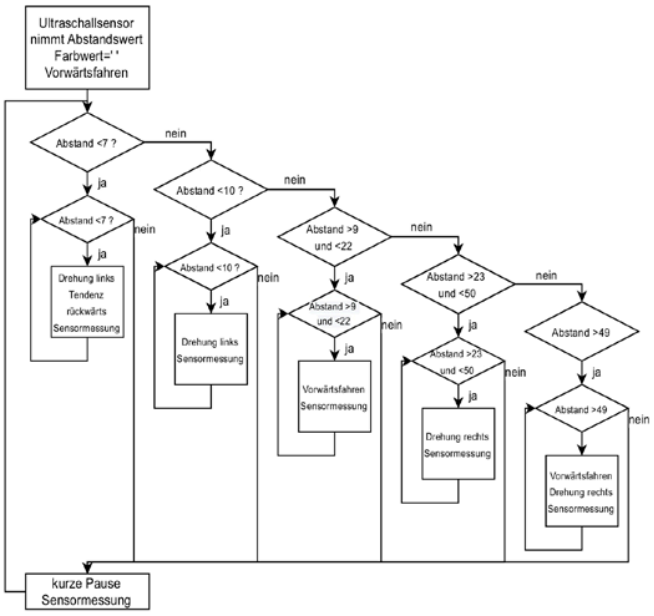


Abbildung 3: Ablaufplan der Fallunterscheidung für Abstandswerte

neue rechte Wand an der er sich bei seiner Ausrichtung orientieren soll. Somit kann der Roboter mit nur einem Ultraschallsensor Abbiegungen im Parkour bewältigen.

Durch die schon angesprochenen Ungenauigkeiten der Motoren ist es wichtig, dass sich der Roboter während der Fahrt neu ausrichtet. Dabei haben wir 5 Fälle eingeführt, die sich durch unterschiedliche Abstandswerte unterscheiden und eine entsprechende Reaktion des Roboters hervorrufen, sodass dieser mit der zuvor beschriebenen Fuzzylogik den Parkour autonom absolviert. Dabei entsprechen die logischen Ränder einer Ausrichtung zu weit links oder einer Ausrichtung zu weit rechts. In Abbildung 3 ist der entsprechende Programmablaufplan dargestellt. Mit Hilfe des Ultraschallsensors wird nach jedem Teilschritt ein Sensorwert aufgenommen, der den Abstand zu der entsprechenden Wand darstellt.

Der erste Fall tritt ein, wenn der Abstandswert kleiner als 7 ist, dieser Fall soll eine Kollision mit einer Wand frontal oder seitlich verhindern, also eine Orientierung, die zu weit nach rechts ist, ausgleichen. Der Roboter führt eine Linksdrehung durch bei der er sich leicht rückwärts bewegt, indem sich der linke Motor stärker dreht, solange bis der Abstand wieder größer als 7 ist, dabei entfernt er sich von der Wand, bleibt aber trotzdem bei einer Orientierung an der rechten Wand und vollführt, falls notwendig eine Drehung bei einer Abbiegung. Der zweite Fall tritt bei Abstandswerten kleiner als 10 auf, dieser dient der Korrektur von Abweichungen, bei denen der Roboter leicht zu weit nach rechts ausgerichtet ist, entsprechend erfolgt eine leichte Linksdrehung auf der Stelle, bis der optimale Abstand erreicht wird. Dieser wird durch den dritten Fall realisiert, bei einem Abstandswert zwischen 9 und 22 soll der Roboter geradeaus fahren, da dies einer ungefähr parallel zur Wand ausgerichteten Bewegung entspricht, welche angestrebt wird. Der vierte Fall tritt ein bei einem Abstand zwischen 23 und 50, was einer leicht zu großen Abweichung nach links entspricht. Entsprechend führt der Roboter eine leichte Rechtsdrehung auf der Stelle durch, bis der Abstand wieder im optimalen Bereich liegt. Der fünfte Fall entspricht einer Orientierung, die stark zu weit nach links ist. Das tritt meistens bei Abbiegungen nach rechts auf, dabei soll der Roboter sich solange auf der Stelle nach rechts drehen, bis er wieder ein reche Wand findet, an der er sich orientieren kann, allerdings war es notwendig, das er zuerst ein Stück nach vorne setzt, damit er nicht an der Ecke verkanntet. Nach Abschluss des Vorgangs folgt er wieder der rechten Wand. Wie in Abbildung 3 erkennbar erfolgt in jeder Schleife nach jedem Zwischenschritt eine Sensormessung, sodass der Roboter so genau wie möglich auf seine aktuelle Situation reagieren kann und so den Parkour autonom absolviert.

C. Farberkennung

Der Roboter soll über die Absolvierung des Parkours hinaus auch bestimmte Stellen in dem Parkour erkennen und auf diese reagieren. Dabei soll zwischen roten und grünen Flächen unterschieden werden. Eine grüne Fläche stellt dabei eine Opferstelle dar, bei der ein bestimmter Ton von dem Roboter erzeugt wird. Eine rote Fläche stellt eine Gefahrenstelle dar, welche umfahren werden soll und bei der ein anderer Ton erzeugt wird. In Abbildung 4 sieht man den Programmablaufplan für diese Fallunterscheidung der Farbwerte. Jeder Fall der vorherigen Fallunterscheidung für die Abstandswerte wurde um diesen Programmteil ergänzt, sodass zu jedem Abstandswert auch ein Farbwert des Untergrundes aufgenommen wurde, auf den der Roboter gegebenenfalls reagiert.

Sollte der Farbsensor einen roten Untergrund feststellen beginnt der Roboter mit einer Linksdrehung. Dabei bewegt er sich auch leicht rückwärts um nicht auf die Gefahrenstelle zu fahren. Diese Rückwärtsbewegung wird realisiert, indem sich der linke Motor stärker dreht als der rechte Motor. Diese Linksdrehung hält solange an, bis der Farbwert nicht mehr rot ist, dann geht der Roboter wieder in den gewöhnlichen Modus über und orientiert sich an der Wand, was dazu führt, dass er sich wieder nach rechts dreht, bis wieder eine rote Fläche gescannt wird. Der Roboter arbeitet sich also durch Linksdrehungen, die in Folge der Farberkennung verursacht werden und Rechtsdrehungen, die in Folge der Abstandserkennung verursacht werden, an dem roten Hindernis vorbei, bis er dieses Überwunden hat und normal den Parkour weiter absolvieren kann.

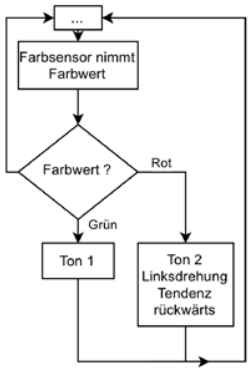


Abbildung 4: Ablaufplan der Reaktion auf farbige Stellen

D. Videobegleitung

In die Halterung, die oben auf dem NXT Stein angebracht wurde, kann nun eine Kamera oder Smartphone befestigt

werden, sodass die Fahrt durch den Parkour aufgezeichnet werden kann und man so den Parkour visuell und die speziellen Stellen audiovisuell nachvollziehen kann. Durch diese Aufnahme kann im Anschluss eine Einsatzplanung sehr koordiniert und spezifisch für den vorliegenden Parkour umgesetzt werden.

IV. Ergebnisdiskussion

Am Ende unseres Projektes ist ein Kettenfahrzeug entstanden, welches sich fortbewegen kann und in der Lage ist sich auf der Stelle zu drehen. Der Roboter konnte einen Parkour autonom und flüssig bewältigen. Zudem werden Gefahrenstellen umfahren und man hört einen Ton, wenn diese erkannt werden. Auch Opferstellen können erkannt werden und der Roboter reagiert entsprechend mit einem anderen Ton. Wir waren zeitlich bedingt nicht mehr in der Lage näher auf die Erstellung einer Karte am Ende des Projektes einzugehen, allerdings ermöglichte uns die Erweiterung des Aufbaus um eine Kamera die Aufzeichnung der Fahrt, sodass der Parkour trotzdem audiovisuell nachvollzogen werden kann.

Während der Arbeit an dem Projekt sind einige Probleme aufgetreten, so gab es anfangs oft Ungenauigkeiten bei der Drehung des Roboters oder eine verzögerte Reaktion auf die Wände, sodass wir wieder auf eine Kabelverbindung umsteigen mussten, damit mehr Messwerte aufgenommen werden können und die Drehungen genauer absolviert werden können. Außerdem mussten wir auch den Parkour ein bisschen anpassen, da es an einigen Stellen Probleme bei der Absolvierung von rechten Winkeln gab, da über den Ultraschallsensor der Wechsel zu einer neuen Orientierungswand nicht gelungen ist. Dort reichte allerdings eine leichte Anschrägung der entsprechenden Wand. Zuletzt gab es dann ein Problem mit Streuwerten, die durch Ungenauigkeiten des Ultraschallsensors hervorgerufen wurden und zu einer fehlerhaften Reaktion des Roboters führen konnten, dieses Problem wurde allerdings gelöst in dem wir den Ultraschallsensor vertikal, statt horizontal an dem Roboter angebracht haben, sodass diese Streuwerte nichtmehr auftraten. Das größte Problem bei der Bewältigung des Projektes waren die Ungenauigkeiten der Motordrehung, sodass wir auch das Nachverfolgen des Roboters im Parkour und das erstellen einer Karte damit nicht realisieren konnten, da bei teilweise gleichen Motordrehzahlen eine unterschiedlich weite Drehung absolviert wurde, sodass ständig ein Ausgleich durch Sensorwerte erfolgen musste und ein Orientierung im Parkour nicht mehr möglich war.

V. Zusammenfassung und Fazit

Im Rahmen des Projektes wollten wir einen Roboter bauen, der in der Lage ist autonom einen Parkour zu bewältigen, spezielle Stellen darin zu erkennen und den Parkour auf einer Karte darzustellen, sodass mit der Ausgabe des Roboters ein Einsatz in dem unbekannten Parkour geplant und bewältigt werden kann. Es ist uns gelungen ein autonom fahrendes Kettenfahrzeug zu konstruieren, welches der praktischen Anwendung eines Aufklärungsroboters schon sehr nahe kommt. Durch die letztendlich entstehende Videoaufnahme des Parkours, mit entsprechender akustischer Begleitung an speziell zu erkennenden Stellen, bietet der Roboter auch eine Grundlage um in einer realen Anwendung einen Einsatz zu planen und koordinieren, da er einen guten ersten Überblick über einen unbekannten Parkour bietet, ohne das ein Mensch sich der Gefahrensituation aussetzen muss. Natürlich funktioniert diese Anwendung nur für starke Vereinfachungen in einem Parkour und ist in der Form nicht nützlich in der realen Anwendung, da man dort nicht auf Farbwerte für Gefahrenstellen zurückgreifen kann und der Parkour sicherlich Stellen enthält, die unser Roboter nicht absolvieren kann.

Es bestehen einige Verbesserungsmöglichkeit, so hätte man das Projekt auch grundsätzlich über ein Tracking mit einer Webcam über dem Parkour umsetzen können und müsste so nicht auf Sensorwerte der Motoren zurückgreifen um sich zu orientieren, außerdem wäre auch die Verwendung von mehr als einem Ultraschallsensor sinnvoll um den Parkour flüssiger und besser absolvieren zu können ohne Probleme an entsprechenden Kanten zu haben. Eine mögliche Erweiterung unseres Konzeptes des Aufklärungsroboters wäre auch wie in der realen Anwendung die nächst höhere Stufe eines Bergungsroboters, der dann auch in der Lage ist mit Hilfe eines Greifarms Objekte aus dem Parkour zu ziehen. Das wäre natürlich deutlich praktischer in der realen Anwendung und würde verhindern, dass sich Menschen überhaupt in die Gefahrensituation begeben müssten.

Literaturverzeichnis

[1] Mathworks, Matlab https://de.mathworks.com/products/matlab.html – [Abgerufen am 2. März 2021]

[2] Mathworks, RWTH Aachen - Mindstorms NXT Toolbox. https://de.mathworks.com/matlabcentral/fileexchange/18646-rwth-mindstorms-nxt-toolbox – [Abgerufen am 2. März 2021]

[3] Wikipedia, the free encyclopedia: Fuzzylogik https://de.wikipedia.org/wiki/Fuzzylogik#Fuzzy-Set-Theorie – [Abgerufen am 3. März 2021]

[4] google: Fachforum autonome Systeme https://www.acatech.de/publikation/fachforum-autonome-systeme-chancen-und-risiken-fuer-wirtschaft-wissenschaft-und-gesellschaft-abschlussbericht/ – [Abgerufen am 3. März 2021]

Autonom fahrender Aufklärungsroboter

Gia Bao Truong, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Abstract*— Roboter werden eingesetzt, um Menschen in allen Bereichen des Lebens zu helfen und Probleme zu lösen. In Situationen in denen das Leben eines Menschen gefährdet werden könnte, soll ein Roboter diese Aktion übernehmen, um das Risiko zu verringern. Im Rahmen des Projektseminars 2021 haben wir uns zur Aufgabe gesetzt einen Aufklärungsroboter aus Lego zu entwickeln, der in realen Szenarien eingesetzt werden kann um eine gefährliche Umgebung abzusuchen. In dem folgendem Paper geht es um die schrittweise Entwicklung eines Roboters, der eingesetzt wird um autonom einen Parkour zu bewältigen und währenddessen Informationen über diesen zu sammeln, welche eine Bergungsmission in dem zuvor unerforschten Gebiet zu ermöglichen. Der Einsatz des Roboters kann in einer realen Situation die Gefährdung von menschlichem Leben vorbeugen.**

I. Einleitung

Technologie wird in einem rasantem Tempo weiterentwickelt und soll das menschliche Leben immer mehr vereinfachen. In allen Bereichen des Alltags findet man Maschinen, die Arbeit übernehmen können, ob es die Kaffeemaschine ist, die auf Knopfdruck einen heißen Kaffee in die Tasse füllt, oder Drucker, die innerhalb von Sekunden ganze Seiten mit Text füllen können. Autonome Systeme finden heutzutage immer mehr Gebrauch in Fahrzeugen. Dabei können sie helfen mit Kameras und Sensoren sicherer einzuparken, schalten automatisch die Scheinwerfer oder Scheibenwischer an wenn sie benötigt werden, oder übernehmen sogar komplett das Fahren. Solche Assistenzsysteme haben schon vielen Leuten Zeit und Nerven gespart. Autonome Systeme werden jedoch nicht nur benutzt um den Alltag zu vereinfachen, sondern auch um dem Verlust von menschlichem Leben zu umgehen. Ein Bergungsroboter kann dabei helfen in einem Gefahrengebiet einen besseren Überblick zu schaffen um somit die darauf folgenden Handlungen besser planen zu können. Dadurch verhindert der Roboter, dass sich ein Mensch in Gefahr begeben muss, um dieses Gebiet zu erkunden, wodurch auch zusätzlich Zeit gespart werden, welche bei der Rettung von verletzten Personen wichtig ist.
Das Ziel war einen Roboter zu konstruieren, der selbstständig einen Parkour durchfahren soll. Dabei muss er Hindernissen ausweichen, bestimmte Stellen als Gefahr erkennen und umgehen und bei der Opferstelle ein akustisches Signal abgeben. Währenddessen sollen Informationen gesammelt werden, um im Anschluss eine Karte vom Parkour erstellen zu können. Hierbei soll der Roboter aus dem zur Verfügung gestelltem Lego-Mindstorm-Set gebaut und mit dem Programm Matlab angesteuert werden.

DOI: 10.24352/UB.OVGU-2021-034

II. Vorbetrachtungen

A. Grundtypen von Bergungsrobotern

Man unterscheidet zwischen zwei Typen von Bergungsrobotern. Der erste Typ sind die Aufklärungsroboter. Sie dienen der Aufklärung von gefährlichen Umgebungen mit dem Ziel einen möglichst genauen Überblick über die Umgebung mit ihren Gefahren zu erhalten. Dadurch soll eine darauf folgende Operation in diesem Gebiet effektiver geplant werden können. Der zweite und komplexere Typ sind die Bergungsroboter. Sie sollen zusätzlich zum ersten Typen auch verletzte Personen aus diesem Gebiet transportieren können.
Aufgrund der begrenzten Zeit und unserem geringem Vorwissen haben wir uns für den ersten Typen entschieden.

B. Umsetzung

Unser Roboter soll autonom durch einen aufgebauten Parkour bewegen. Dabei soll er sich immer an der Wand rechts von ihm entlangfahren bis er an das Ende des Parkours gelangt. Es wurde sich bei dem Roboter für ein Kettenfahrzeug mit zwei Ketten, die jeweils von einem Motor angetrieben werden entschieden, weil somit Drehungen leichter zu realisieren sind. Über einen Ultraschallsensor, der den Abstand misst, soll der Roboter die Information, darüber wie weit der Roboter von der Wand rechts von ihm entfern ist, bekommen und sich mit dem Ziel den Abstand möglichst konstant zu halten entsprechen bewegen. Hierbei müssen wir vor dem Start einen Abstand definieren und wenn der gemessene Abstand größer ist als der vordefinierte, dann soll der Roboter sich näher an die Wand bewegen und wenn der gemessene Abstand kleiner ist als der vordefinierte Abstand, dann soll der Roboter sich von der Wand wegbewegen. Der Abstandausgleich wird durch das Drehen des Roboters, welches durch das Antreiben der Ketten in unterschiedliche Richtungen realisiert wird, ermöglicht. Um zu verhindern, dass sich der Roboter die ganze Zeit nur hin und her dreht gibt es einen Toleranzbereich in dem der Abstand sein darf und in dem der Roboter geradeaus fährt.

C. Fuzzy Logic

In der einfachen Logik wird nur zwischen den zwei Zuständen 0 und 1 beziehungsweise falsch und wahr unterschieden. Für das Bewältigen des Parkours in einer moderaten Zeit reichen zwei Zustände allerdings nicht aus. Entweder könnte der Roboter nur auf einen passenden und unpassenden Abstand, oder auf einen zu kleinen oder zu großen Abstand reagieren. Im ersten Fall fehlt dem Roboter die Information, ob er beim unpassenden Abstand sich der Wand nähern oder sich von der

Wand entfernen soll. Im zweiten Fall würde der Roboter sich nur nach links oder rechts drehen und um den gewählten Abstand schwanken ohne dabei vorwärts zu kommen. Die Fuzzylogik kann zwischen mehr als nur zwei Zuständen unterscheiden und wurde entwickelt um mit Unschärfen zu arbeiten. Übertragen auf unseren Roboter kann der Roboter nun auf mehr als nur zwei Fälle reagieren. Ohne die Fuzzylogik konnte er nur auf einen zu großen, oder einen zu kleinen Abstand regieren. Mit der Fuzzylogik ist es ihm möglich einen Abstand in >>viel zu groß<<, >>ein wenig zu groß<<, >>passend<<, >>ein wenig zu klein<<, und „zu klein“ zu unterteilen und dem entsprechend einzuschätzen wie er zu reagieren hat. Dies ist wichtig, weil der Roboter nun weiß, ob er seine Fahrt nur leicht korrigieren muss, oder sogar komplett abbiegen muss. Entsprechend zum Abstand kann der Roboter schließlich in beide Richtungen seinen Abstand nur leicht korrigieren, in beide Richtungen abbiegen und bei einem optimalen Abstand geradeaus fahren.

III. Realisierung

A. Aufbau

Der Roboter ist so aufgebaut, dass zwei Lego NXT Motoren direkt nebeneinander befestigt wurden. Die Motoren treiben jeweils eine Kette an und können unabhängig voneinander angesteuert werden. Die Unabhängigkeit beider Ketten wird benötigt um Drehungen im Uhrzeigersinn oder gegen den Uhrzeigersinn zu realisieren. Wenn der Roboter sich im Uhrzeigersinn drehen soll, dann dreht der Motor auf der linken Seite nach vorne und der Motor auf der rechten Seite nach hinten. Bei einer Drehung gegen den Uhrzeigersinn drehen sich die Motoren dementsprechend anders herum. Auf diesen Motoren wird der NXT Stein gesetzt. Der NXT Stein ist die Schnittstelle zwischen dem Roboter und dem Computer. Per USB Kabel oder Bluetooth kann der Computer dann mit den NXT Stein kommunizieren. Dadurch erreichen wir eine möglichst kleine Grundfläche, wodurch Drehungen auf kleinem Raum möglich sind. Das ist wichtig, weil der Roboter so durch schmalere Stellen des Parkours kommen kann und auch nicht viel Fläche einnimmt. In Abbildung 1 ist erkennbar, dass zusätzlich zu den Motoren noch zwei weitere Sensoren verwendet werden. Diese sind vor den Motoren und dem NXT Stein angebracht. Der Ultraschallsensor ist ganz vorne in einem 45 Grad Winkel angebracht. Dieser 45 Grad Winkel ermöglich die Abstandsmessung nach vorne sowohl als auch nach rechts. Dadurch kann der Roboter den Abstand nach rechts halten und gelichzeitig davor geschützt werden mit der Wand, auf die er sich zubewegt zu kollidieren. Der Farbsensor ist genau vor der linken Kette platziert und nach unten gerichtet um später Farben zu erkennen. Die Kabel, die den NXT Stein mit den Motoren und Sensoren verbinden wurden so platziert, dass sie die Bewegung des Roboters nicht einschränken und die Messungen der Sensoren nicht beeinträchtigen oder verfälschen. Zusätzlich gibt es eine Halterung, in die ein Smartphone eingesetzt werden kann um die Fahrt aus Fahrersicht zu filmen.

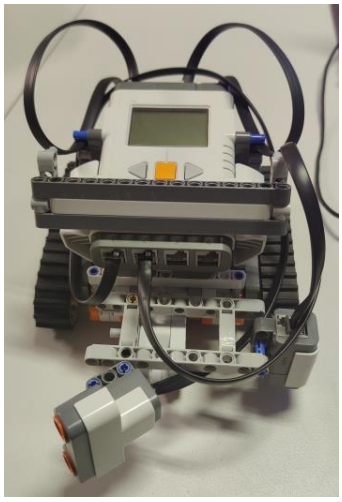


Abbildung 1:Roboter mit Smartphone-Halterung.

B. Autonomes Fahren

Grundsätzlich wird das autonome Bewältigen des Parkours dadurch geschafft, dass der Roboter sich mit Hilfe des Ultraschallsensors an der rechten Wand hält. Dazu wird permanent der Abstand gemessen und ausgewertet. Dadurch wird gleichzeitig eine schräge Ausrichtung korrigiert. Schräge Ausrichtungen kommen dadurch zustande, dass die Motoren sich nicht exakt gleich verhalten und deswegen unterschiedlich schnell beschleunigen und abbremsen. Dadurch kommen ungenaue Drehungen zustande welche ausgeglichen werden müssen. Wie in Abbildung 2 erkennbar ist, sorgt die 45 Grad Ausrichtung des Ultraschallsensors an

der Front dafür, dass der Abstand zur rechten Wand und zur vorderen Wand gemessen werden kann. Wenn eine 90 Grad

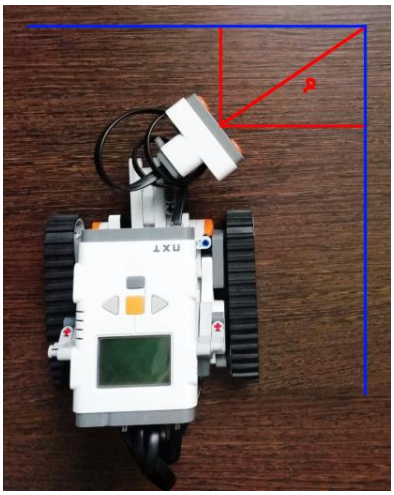


Abbildung 2: Abstandsmessung nach rechts und vorne durch 45 Grad Ausrichtung des Ultraschallsensors

Ecke nach links kommt kann der Roboter nun die Wand vor sich zu seiner rechten Wand machen indem er sich gegen den Uhrzeigesinn dreht. Danach orientiert er sich wieder an der von ihm rechten Wand. Dadurch kann Roboter nun Abbiegen ohne gegen eine Wand zu fahren. Durch die Ungenauigkeiten der Motoren muss der Roboter sich während der Fahrt ständig neu ausrichten. Dabei gibt es, wie in der Fuzzylogik beschrie-

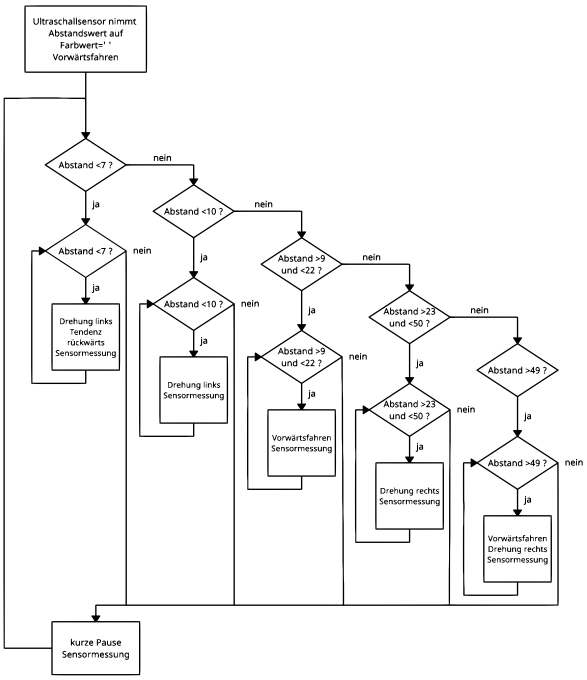


Abbildung 3: Programmablaufplan zum Algorithmus zur Abstandserkennnung.

ben, 5 verschiedene Fälle, die durch die unterschiedlichen Abstandswertbereiche zustande kommen und dem entsprechend unterschiedliche Reaktionen hervorrufen damit der Roboter sich autonom durch den Parkour bewegen kann. Im ersten Fall ist der Abstand kleiner als 7. In diesem Fall dreht der Roboter sich gegen der Uhrzeigersinn und bewegt sich dabei leicht nach hinten. Dabei orientiert er sich immer noch an der rechten Wand. Durch diesen Falls wird verhindert, dass der Roboter gegen eine Wand fährt. In Fall zwei ist der gemessene Abstand zwischen 7 und 10. Dieser Fall tritt ein, wenn der Roboter bei der Fahrt etwas zu nah an die Wand heranfährt. In diesem Fall führt der Roboter eine Drehung auf der Stelle gegen den Uhrzeigersinn aus bis der optimale Abstand wieder erreicht ist. In dem Fall in dem der Roboter sich im optimalen Abstand zur Wand befindet ist der gemessene Abstand zwischen 9 und 22. Bei einem Abstand zwischen 9 und 22 steht der Roboter relativ parallel zur rechten Wand und soll geradeaus fahren indem beide Motoren mit gleicher Leistung nach vorne drehen. In Fall 4 ist der gemessene Abstand zwischen 23 und 50. Das entspricht einer leichten Auslenkung nach links und führt zu einer Drehung auf der Stelle im Uhrzeigersinn bis der Abstand wieder im optimalen Bereich ist. Im fünften Fall ist der gemessene Abstand größer als 49. Das tritt ein wenn eine Abbiegung nach rechts geschehen soll. Dafür muss der Roboter allerdings zuerst ein kleines Stück nach vorne fahren um nicht mit der Ecke zu verkanten. Danach Dreht er sich so lange im Uhrzeigersinn bis er wieder eine Wand findet, an der er sich orientieren kann. Dieser Algorithmus ist in Abbildung 3 beschrieben. Hier erkennt man auch, dass nach jeder Aktion der Ultraschallsensor wieder einen Abstand misst.

C. Farberkennung

Der Roboter soll nun neben dem Durchfahren des aufgestellten Parkours auch noch farbig gekennzeichnete Flächen erkennen und auf diese je nach Farbe reagieren. Hierbei soll zwischen roten Gefahrenflächen und grünen Opferflächen unterschieden werden. Wenn der Roboter eine grüne Opferfläche erkennt, soll er ein Signal in Form von einem Geräusch abgeben. Dieses Geräusch kann der NXT-Stein mit dem integrierten Lautsprecher erzeugen. Eine rote Fläche soll als Gefahrenfläche erkannt werden. In einer realen Anwendung wäre dies ein Loch, welchem der Roboter ausweichen muss. In Abbildung 4 ist der Programmablaufplan für die Farbmessung dargestellt. Der Algorithmus für die Abstandsmessung mit dem Ultraschallsensor wurde in den Algorithmus für die Farbmessung eingefügt, sodass der Roboter zusätzlich zur permanenten Abstandsmessung auch eine permanente Farbmessung durchführt. Wenn der Roboter eine Rote Farbfläche erkennt bleibt er zunächst stehen. Danach dreht er sich nach links, wobei er sich gleichzeitig etwas nach hinten bewegt. Das wird erreicht indem der linke Motor mit einer höheren Leistung nach hinten dreht als der rechte Motor sich nach vorne dreht. Dies wird so lange wiederholt bis er keine rote Farbe mehr erkennt. Danach orientiert er sich wieder an dem vom Ultraschallsensor gemessenen Abstand, was dazu führt, dass der Roboter sich nach vorne rechts bewegt weil der Abstand viel zu groß ist. Dadurch erkennt er wieder die rote Fläche und fängt wieder an sich nach links zu drehen. Dadurch bewegt der Roboter sich also durch ständige Rechtsdrehungen durch die Abstandsmessung und ständigen Linksdrehungen durch die Farbmessung langsam aber sicher an der roten Gefahrenfläche vorbei bis er wieder eine Wand vor sich erkennt an der er sich orientiert.

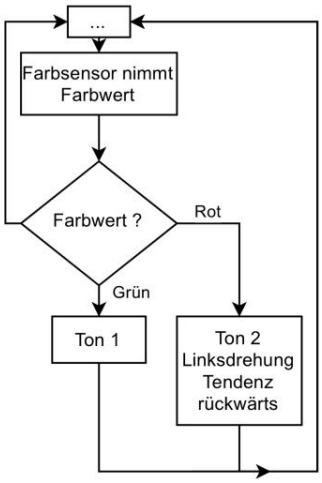


Abbildung 4:Programmablaufplan zum Algorithmus zur Farberkennnung.

D. Videobegleitung

Wie im Aufbau erwähnt gibt es eine Halterung für ein Smartphone, welches die Fahrt aufnehmen soll. Das hat den Zweck, dass der Parkour im Nachhinein auch durch ein Video nachvollzogen werden kann. Das hat in einer realen Anwendung den Vorteil, dass ein darauf folgender Einsatz in diesem Gebiet besser geplant werden kann.

IV. Ergebnisdiskussion

Nach dem zweiwöchigen Legopraktikum ist es uns gelungen ein Kettenfahrzeug aus Lego zu bauen welches durch Drehungen autonom einen Parkour in annehmbarer Zeit bewältigen kann. Zudem erkennt unser Aufklärungsroboter Gefahrenstellen, denen er ausweichen kann und Opferstellen an dem er ein akustisches Signal von sich gibt. Aufgrund der begrenzten Zeit ist es uns nicht gelungen eine automatische Kartenerstellung zu implementieren. Dahingegen haben wir aber eine Möglichkeit gefunden den Ablauf per Video festzuhalten, sodass die Fahrt später nachvollzogen werden kann. Während dem Arbeiten und Testen an dem Roboter sind uns allerdings auch einige Probleme aufgetreten. Durch die Verbindung mit Bluetooth waren wir zwar nicht mehr auf einen engen Raum gebunden, haben allerdings durch die Latenz immer wieder Ungenauigkeiten bei Drehungen erkannt. Dadurch mussten wir wieder auf Kabelverbindungen umsteigen. Nicht nur der Roboter, sondern auch der Parkour hatte Probleme. An 90-Grad Ecken, an denen der Roboter sich nach links drehen soll blieb er öfters hängen. Das ließ sich beheben indem der Winkel etwas vergrößert wurde. Da die Sensoren nicht dafür gedacht sind zu jeder Zeit sehr genaue wert zu liefern und im Endeffekt auch nur Spielzeug sind, traten manchmal unerklärliche Werte auf, die den Roboter etwas durcheinander brachten. Das konnten wir beheben indem wir den Ultraschallsensor nicht horizontal sondern vertikal verbauten. Das größte Problem waren jedoch die Ungenauigkeiten in den Motoren. Öfters drehten sich die Motoren weiter als sie sollten bei gleichen Befehlen. Dadurch musste durch die Sensoren immer die Lage ausgeglichen werden, was ein Orientieren im Parkour nicht möglich gemacht hat. Somit mussten wir schließlich auch das Erstellen der Karte verzichten.

V. Zusammenfassung und Fazit

Wir haben von Anfang an geplant einen Aufklärungsroboter zu entwickeln, der durch einen Parkour fahren kann ohne dabei die Wände zu berühren und zudem auch noch verschiedene Stellen erkennt und schließlich eine Karte erstellt. Hierzu haben wir uns reale Aufkläreungroboter als Beispiel genommen und uns an ihnen orientiert. Wir haben es geschafft ein Kettenfahrzeug zu bauen, welches autonom durch einen unbekannten Parkour fahren konnte und dies durch Videoaufnahme aufzeichnen konnte. Dabei hat das Fahrzeug Farben erkannt und hat dem entsprechend Töne an den dafür gedachten Stellen gemacht. Damit haben wir er geschafft einen Roboter zu bauen, der an Stelle von Menschen ein ungekanntes potenziell gefährliches Gebiet erforscht um somit Rettungsmissionen zu ermöglichen, ohne, dass sich dafür ein Mensch vorher in Gefahr begeben musste. Es ist zwar immer noch eine stark vereinfachte Variante von echten Aufklärungsrobotern, die mehr Herausforderungen haben als nur gerade Wände und farblich gekennzeichnete Flächen auf dem Boden. Es bestehen also noch sehr viele Verbesserungsmöglichkeiten. Mit mehr Ultraschallsensoren wäre es sicherlich möglich gewesen die Umgebung besser zu erfassen und den Roboter besser durch den Parkour zu manövrieren.

Literaturverzeichnis

[1] Alexander Behrens (2021). RWTH - Mindstorms NXT Toolbox (https://www.mathworks.com/matlabcentral/fileexchange/18646-rwth-mindstorms-nxt-toolbox), MATLAB Central File Exchange. Retrieved March 28, 2021.

[2] Wikipedia, The Free Encyclopedia – FuzzyLogik. https://de.wikipedia.org/wiki/Fuzzylogik - 28.03.2021

Sortierbobby – Der Sortierroboter

Sa'ed Abushawish, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Schon seit 2013 findet jedes Jahr das Projektseminar der Fakultät Elektrotechnik/Informationstechnik statt. Ziel ist, einen Roboter mithlife der LEGO-Bausteine zu bauen und mittels MATLAB zu programmieren. Roboter spielen heutzutage eine große Rolle im Leben. Sie können viele Aufgaben übernemen, die die Menschen wegen ihrer Gefahr wie z.B. Entschärfen von Bomben nicht erledigen können. Dieses Paper befasst sich mit einem Roboter, der die Farben von Objekten erkennt und an die richtige Stelle bringt (je nach Farbe). Außerdem wird noch erwähnt, wie der Roboter gebaut und anschließend programmiert wurde.**

***Schlagwörter*—Farbsortierer, Lego Mindstorms, NXT, OVGU, Projektseminar**

I. Einleitung

Roboter gibt es schon überall auf der Welt. Sie werden überall eingesetzt, wo sie auch gebraucht werden. Sie sind heute nicht mehr wegzudenken und werden auch immer mehr zum Einsatz kommen. In der Zukunft denkt man sich, dass die Welt von Robotern besetzt wird und deshalb verfallen die Menschen in Angst und Bange. Doch wenn man sich mit den Robotern auseinandersetzt, so werden die Ängste schnell zur Nebensache.

Es existieren viele Vorteile vom Roboter. Roboter erledigen die Arbeit schneller als die Menschen, sind billiger anzuschaffen und verursachen keine Fehler.

Eines der Roboter die eingesetzt werden, sind Sortierroboter. Sortierroboter sind dafür zuständig, dass verschiedene Objekte schnell an die richtigen Plätze gebracht werden. Beispielsweise findet man Sortierroboter heutzutage in der Indusrtie, wo z.B. rote Deckel oder blaue Deckel zu der richtigen Flasche gebracht werden.

Deswegen wurde entschieden, einen Sortierroboter im Zuge des Projektseminars zu entwicklen mit dem Namen „Sortierbobby".

II. Aufbau und Funktionsweise des Roboters

Hier wird diskutiert, wie den Roboter aufgebaut und programmiert wird. Außerdem wird die Funktionsweise des Sortierroboters erklärt.

A. Aufbau

Der Roboter besteht aus 3 Motoren, zahlreichen LEGO-Steinen, einem NXT-Gerät, einem Arm und einem Farbsensor.

Die drei Motoren haben zahlreiche Aufgaben. Motor C ist für das Greifen vom Ball zuständig. Der Greifer (siehe Abbildung 5) ist mit dem Arm verbunden und ist so konstruiert, dass er die Bälle richtig greift, damit die Bälle während der Bewegung nicht herausfallen. Motor B hingegen hebt den Arm nach oben oder nach unten an, nachdem der Greifer den Ball schon gegriffen hat. Einer zusätlicher Motor (Motor A) wurde dafür gebraucht, dass sich der Arm immer 90° vom Startpunkt nach rechts oder nach links dreht, wo die Behälter sind. Die Motoren sind in der Abbildung 2 dargestellt.

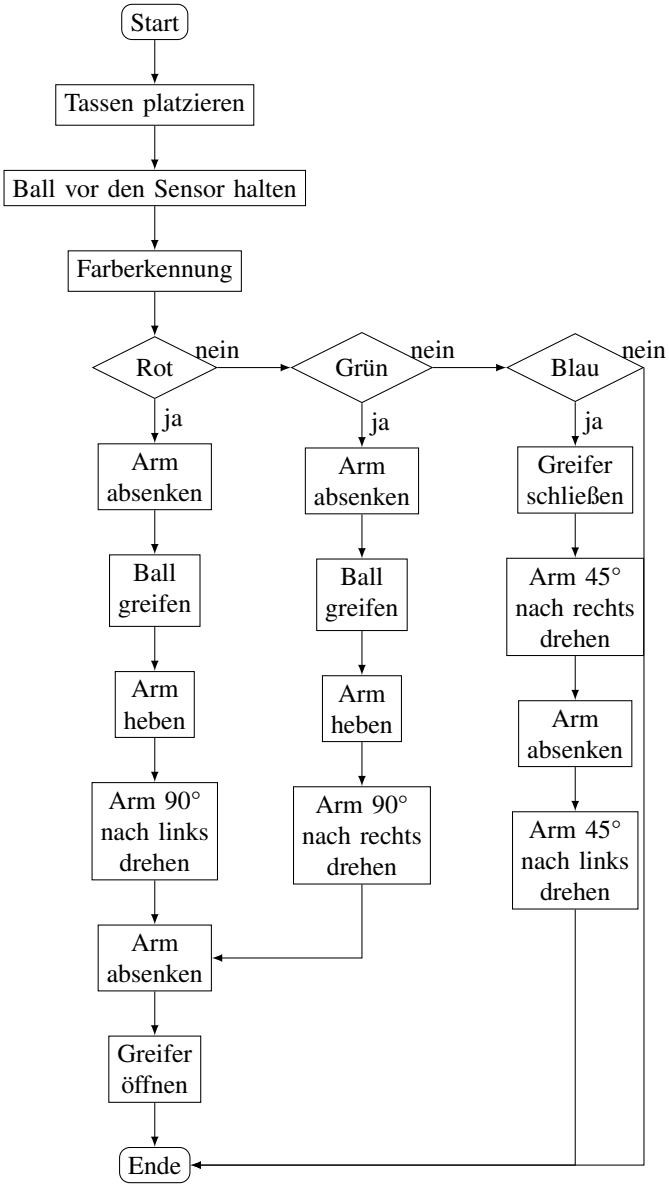


Abbildung 1. Programmablaufplan

Der Arm muss die passende Länge haben, damit er bei der Bewegung nach unten, nach Oben, nach rechts und nach Links nicht gegen den Rest des Konstrukts stößt. Der Arm des Roboters wird in der Abbildung 3 abgebildet.

DOI: 10.24352/UB.OVGU-2021-035

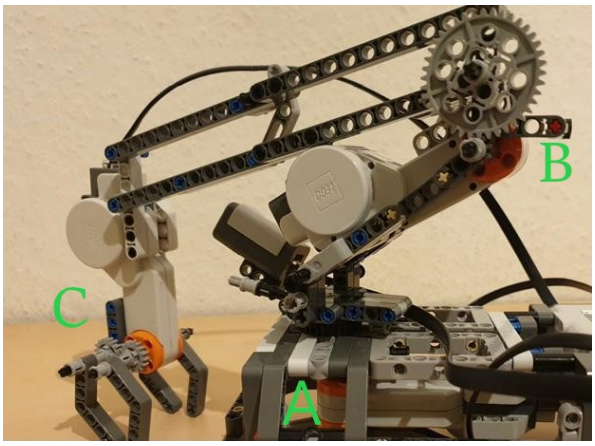


Abbildung 2. Motoren

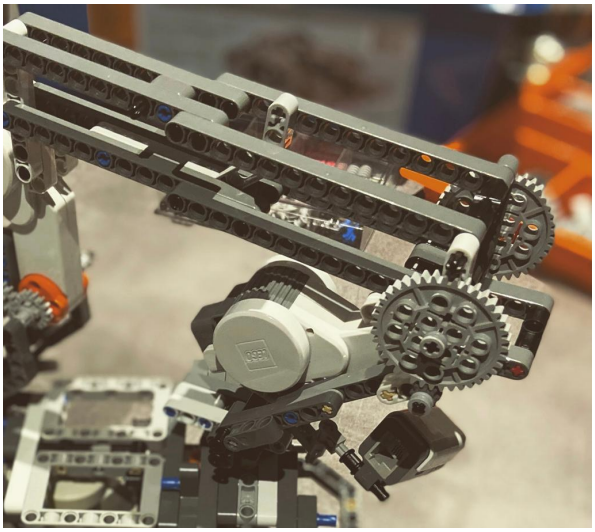


Abbildung 3. Arm des Roboters

Abbildung 4. Farbsesnor

Der Robtoer wurde aus LEGO-Steinen zusammengebaut. Zur Steuerung des Roboters kam das NXT-Gerät zum Einsatz. Dieses Gerät dient dazu, die Befehle vom Computer zu empfangen und dann an die Motoren weiterzuleiten. Desweitern wurde noch einen Farbsensor eingesetzt.

Farbsensor kam auch zum Einsatz. Der Sensor ist dafür zuständig, die Farben von Bälle zu erkennen. Man sieht den Farbsensor in der Abbildung 4.

B. Programmablauf

Ein beispielhafter Programmablaufplan zur Erklärung einer Prozedur oder einer Routine ist in Abbildung 1 dargestellt.

Die Kabel sind alle angeschlossen, und das NXT-Gerät ist schon an. Das Programm wird gestartet, in dem auf -RUN- in MATLAB-Programm geklickt wird. Per Hand bringt man die Bälle vor dem Farbsensor, sodass der Roboter die Farben gut erkennt. Zwischen 3 Farben (Rot, Grün und Blau) muss unterschieden werden. Ist die erkannte Farbe Rot, dann bewegt sich der Arm erstmal nach unten, greift den Ball mit dem Greifer und hebt den Ball nach oben an. Dann bewegt der Arm

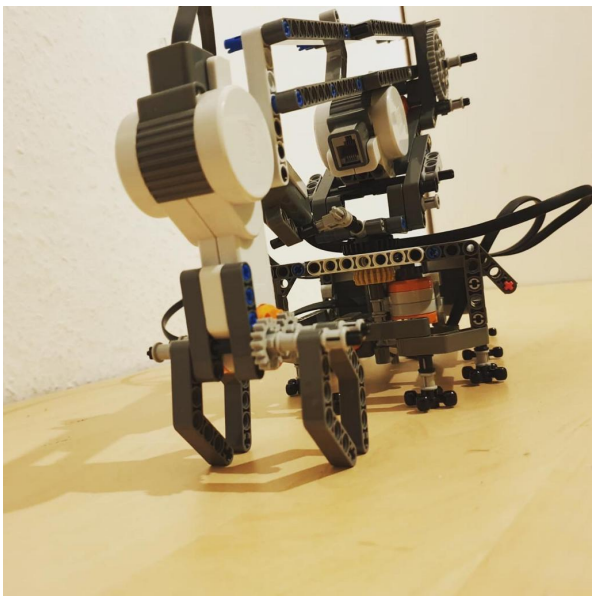


Abbildung 5. Greifarm des Roboters

sich dann um 90° nach links und lässt den Ball in die rote Tasse fallen. Am Ende kehrt der Arm an den Nullpunkt (bzw. 0°) zurück. Falls aber die erkannte Farbe Grün ist, wiederholt sich dann der ganze Ablauf, aber diesmal bewegt sich der Arm um 90° nach rechts und wirft den Ball in den grünen Behälter. Sollte die Farbe des Balls aber Blau sein, dann dreht sich der Arm um 45° nach rechts und bewegt sich nach unten, und dann stößt den Ball nach links weg.

III. Probleme

Auf jeden Fall sind viele Probleme aufgetaucht. Zum einen ist die Ungenauigkeit des Farbsensors ein Problem. Es kann sein, dass beim Scannen der Bälle die Farben falsch gelesen werden. Das hat zur Folge, dass entweder die Bälle an die falsche Stelle gebracht wird, oder gar keine Bewegung stattfindet. Man kann dieses Problem vermeiden, in dem man ein Farbsensor mit hoher Genauigkeit verwendet.

Zum anderen ist die Länge der Kabel zu kurz. Die Kabel waren nicht lang genug, sodass sich der Arm ohne Probleme dreht. Aufgrund dieses Problems wurde die Bewegung des Roboters um 90° nach rechts bzw. nach links beschränkt. Der Roboter kann sich z.B. nicht um 180° drehen.

Ein weiteres Problem tauchte bei der Verbindung zwischen dem NXT-Gerät und dem Laptop auf, da es versucht wurde, die Verbindung per Bluetooth und nicht mit dem Kabel herzustellen. Das Problem war so, dass die Verbindung zwischen dem Gerät und dem Laptop manchmal verloren wurde.

Es wird empfohlen, dass man die Verbindung zwischen dem NXT-Gerät und dem Laptop immer mit dem Kabel herstellt. Außerdem wäre es besser, wenn man lange Kabel statt der kurzen Kabel nutzt, damit man keine Limitationen für die Bewegung seines Roboters bekommt.

IV. Anwendungsbereiche

Sortierroboter kann man in verschiedenen Bereichen einsetzen, z.B. Zuhause. Es ist schon deutlich, dass der Müll in Deutschland nach Material getrennt wird. Mülltrennung kann eine Aufgabe für den Roboter sein, dass er die Müllsäcke nach Farben sortiert.

Man kann die Idee weiterentwickeln und Militärisch einsetzen. Er kann z.B. nach Bomben suchen und dann sie entschärfen.

Weiterentwicklung kann dazu führen, dass im Krankenhaus auch Roboter zum Einsatz kommen, in dem der Roboter die saubere Sachen sortiert, ohne direkter Kontakt mit Menschen zu haben, und somit vermeidet man die Krankheiten.

V. Ergebnisdiskussion

Das Ziel dieses Praktikums wurde erreicht. Der Roboter kann die farbigen Bälle erkennen, und dann sie greifen und sie ans richtige Platz bringen.
Ein Problem ergab sich durch den Farbsensor. Der Farbsesnor hatte das Problem, dass er nur in bestimmten Lichtverhältnissen gut arbeitet. Der Ball, bei dem die Farbe durch den Farbsesnor erkannt werden musste, musste an den Farbsesnor so nah wie möglich gebracht werden. Falls der Ball nicht nah genug an dem Farbsesnor war oder das Licht nicht zu hell, so

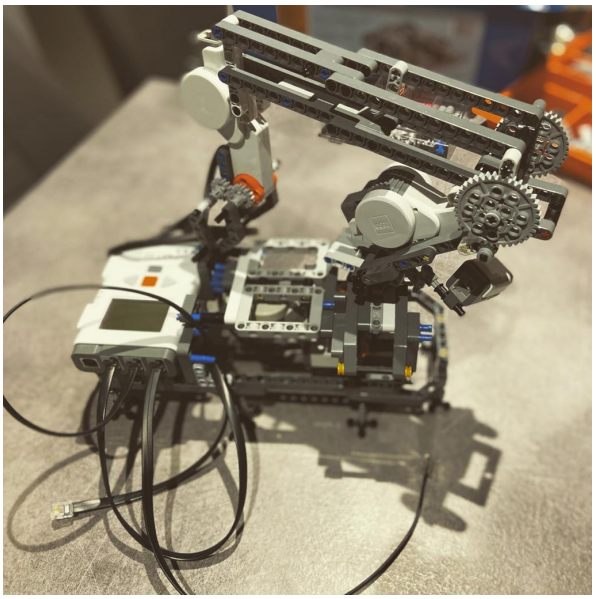


Abbildung 6. Basis des Roboters

fumktionierte der Farbsesnor nicht und das führt zu einem Fehler im Programm und es musste alles neugestartet werden. Ein weiteres Problem waren die Bauteile. Mehr Bauteile könnte einen besseren Aufbau ermöglichen. Das Problem lag nicht nur an dem Farbsesnor, sondern auch an mangelnde Bauteile.

Ein weiterer Farbsesnor hätte es möglich gemacht, auch die Behälter ebenfalls analysieren zu können.

VI. Zusammenfassung und Fazit

Während des Projektseminars Lego Mindstorms wurde erfolgreich ein Roboter entwickelt, der seine Aufgaben verwirklicht hat. Und das Zeil wurde erfolgreich abgeschlossen. Eine neue Programmiersprache zu lernen, ist heutzutage sehr wichtig, damit man die Welt weiterentwickeln kann.

Die Sortierroboter werden schon viel eingesetzt und ich hoffe das sie noch mehr zum Einsatz kommen wie z.B bei Abfall im Wasser könnten die Roboter nicht nur den Menachen einen Vorteil sein, sondern auch der Natur.

Literaturverzeichnis

[1] A. Herz, "Die Vor- und Nachteile der Automation durch Roboter,"
https://wandelbots.com/die-vor-und-nachteile-der-automation-durch-roboter

ANHANG

Der folgende Programmcode, der mittels MATLAB geschrieben wurde, zeigt, wie man den Roboter in Bewegung gesetzt hat.

```
handle = COM_OpenNXTEx('Bluetooth','001653193FC4','bluetooth.ini')
port = SENSOR_3;
OpenNXT2Color(port,'FULL',handle);
pause(3)
color = GetNXT2Color(port,handle);
display('go');
pause(5)
if strcmp(color,'GREEN')
    motorA = NXTMotor('A','Power',20,'TachoLimit',350)
    motorA.SendToNXT(handle);
    pause(3)
    motorB = NXTMotor('B','Power',-20,'TachoLimit',20)
    motorB.SendToNXT(handle);
    pause(3)
    motorA = NXTMotor('A','Power',-20,'TachoLimit',350)
    motorA.SendToNXT(handle);
    pause(3)
    motorC = NXTMotor('C','Power',20,'TachoLimit',300)
    motorC.SendToNXT(handle);
    pause(3)
    motorB = NXTMotor('B','Power',20,'TachoLimit',20)
    motorB.SendToNXT(handle);
    pause(3)
    motorC = NXTMotor('C','Power',-20,'TachoLimit',300)
    motorC.SendToNXT(handle);
elseif strcmp(color,'RED')
    motorA = NXTMotor('A','Power',20,'TachoLimit',350)
    motorA.SendToNXT(handle);
    pause(3)
    motorB = NXTMotor('B','Power',-20,'TachoLimit',20)
    motorB.SendToNXT(handle);
    pause(3)
    motorA = NXTMotor('A','Power',-20,'TachoLimit',350)
    motorA.SendToNXT(handle);
    pause(3)
    motorC = NXTMotor('C','Power',-20,'TachoLimit',200)
    motorC.SendToNXT(handle);
    pause(2)
    motorB = NXTMotor('B','Power',20,'TachoLimit',20)
    motorB.SendToNXT(handle);
    pause(3)
    motorC = NXTMotor('C','Power',20,'TachoLimit',200)
    motorC.SendToNXT(handle);
else
    motorB = NXTMotor('B','Power',-20,'TachoLimit',20)
    motorB.SendToNXT(handle);
    pause(2)
    motorC = NXTMotor('C','Power',20,'TachoLimit',200)
    motorC.SendToNXT(handle);
    pause(2)
    motorA = NXTMotor('A','Power',20,'TachoLimit',330)
    motorA.SendToNXT(handle);
    pause(3)
    motorC = NXTMotor('C','Power',-100,'TachoLimit',350)
    motorC.SendToNXT(handle);
    pause(2)
    motorC = NXTMotor('C','Power',20,'TachoLimit',150)
    motorC.SendToNXT(handle);
    pause(2)
    motorA = NXTMotor('A','Power',-20,'TachoLimit',330)
    motorA.SendToNXT(handle);
    pause(2)
    motorB = NXTMotor('B','Power',20,'TachoLimit',20)
    motorB.SendToNXT(handle);
    pause(2)
end
CloseSensor(port,handle);}
```

Sortierbobby – Der Sortierroboter

Saad Al-Hamid, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Seit 2013 findet jährlich das Projektseminar der Fakultät Elektrotechnik/Informationstechnik statt, welches die Aufgabe stellt, den Studierenden einen Roboter mithilfe der Bausteine bauen zu lassen und entsprechend auch zu simulieren. Die Idee war, im Projektseminar einen Sortierroboter zu entwickeln. In dem nachfolgenden Artikel wird veranschaulicht, wie sich der Sortierroboter mithilfe der Lego Bausteine, den dazugehörigen Geräten und MATLAB realisieren lässt. Im Zuge des Projekts ist ein Roboter entwickelt worden, der Bälle entsprechend ihrer Farbe sortiert.**

***Schlagwörter*—Farberkennung, LEGO® MINDSTORMS®, MATLAB, NXT-Gerät, Sortierroboter**

I. Motivation

In der heutigen Welt werden viele Arbeiten durch von Menschen erfundene Roboter erledigt, welches es unmöglich macht, sie wegzudenken. Die Roboter ermöglichen vieles, was die Menschen nicht schaffen können. Viele Menschen fürchten sich vor Robotern, doch wenn man sich mit den Robotern und ihrer Programmierung beschäftigt, so lassen sich mehr Vorteile als Nachteile herausfiltern. Die Roboter ermöglichen eine erhöhte Gesamtprdokuktivität, sind flexibler und agiler, ermöglichen einen geringeren Kostenfaktor, haben eine hohe Präzision und werden nie müde.
Ein Typ von Robtern, welcher von Menschen eingesetzt wird, ist der Sortierroboter. Der Sortierroboter hat, wie der Name schon sagt, die Aufgabe, Objekte an ihre richtige Stelle zu bringen. Der Sortierroboter wird heute überall auf der Welt eingesetzt. Ein Beispiel, wo Sortierroboter eingesetzt werden, sind Kliniken. In den Kliniken werden sie eingesetzt, um Essenswagen oder Medikamentenwagen zu transportieren und zu sortieren.
Deshalb war das Ziel im Projektseminar, einen Sortierroboter zu entwickeln mit den Namen „Sortierbobby“. Die Aufgabe von „Sortierbobby“ ist die Sortierung beliebiger Objekte.

II. Vorbetrachtungen

Die Idee bestand darin, dass ein Sortierroboter gebaut wird. Um dies zu verwirklichen, wurden drei Motoren, einen Farbsesnor, das NXT-Gerät und verschiedene LEGO®-Bausteine gebraucht.

A. Motor

Es wurde überlegt, drei Motoren in den Roboter einzubauen. Der erste Motor (A) ist dafür zuständig, dass sich der Greifarm horizontal, also nach links und rechts bewegen kann. Der zweite Motor (B) hat die Aufgabe, den Greifarm nach unten und oben zu bewegen. Der dritte Motor (C) hat die Aufgabe, die jeweilgen Objekte zu greifen. Die Motoren sind in Abbildung 1 dargestellt.

DOI: 10.24352/UB.OVGU-2021-036

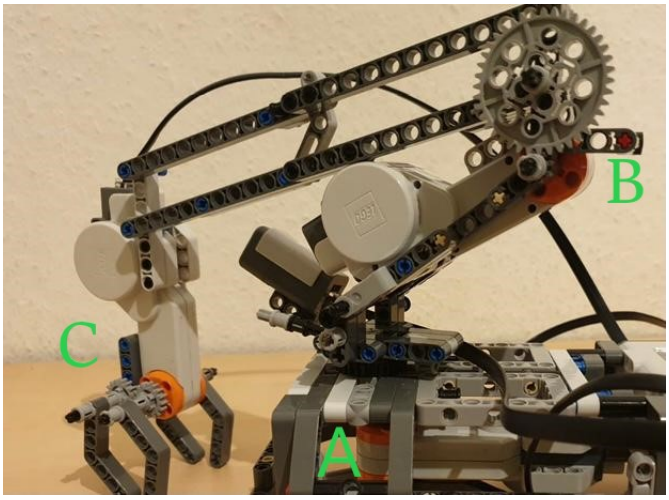


Abbildung 1. Position der Motoren A, B und C

B. Farbsensor

Der Farbsensor erkennt die Farbe des jeweiligen Objekts. Die Objekte waren drei unterschiedlich gefärbte Bälle. Der erste Ball ist rot. Der Farbsensor soll diese Farbe erkennnen und dann soll der Ball an die richtige Stelle gebracht werden. Der zweite Ball ist blau. Nach dem Erkennen der blauen Kugel vom Sensor soll sich der Greifarm schließen und dann 45° nach rechts bewegen. Danach bewegt sich der Arm nach unten und schlägt den Ball mit dem geschlossenen Greifarm seitlich an. Der Roboter weigert sich sozusagen, den Ball aufzunehmen. Der dritte Ball mit grüner Farbe soll nach dem Erkennen mithilfe des Farbsensors ebenfalls an die richtige Stelle gebracht werden. Der Farbsensor ist in Abbildung 2 zu sehen.

C. NXT-Gerät und LEGO®-Bausteine

Die LEGO®-Bausteine waren der Schlüssel zu allem. Ohne die Bausteine wäre es nicht möglich gewesen, den Roboter zu bauen. Dennoch waren die meisten Probleme wegen der Bausteine. Zum Beispiel hat sich der Roboter nicht bewegt. Obwohl man dachte, das alles richtig war und schon mit dem Programmieren begonnen wurde, funktionierte er nicht richtig. Es wurde von vorne wieder angefangen und schleißlich nach vielen gescheiterten Versuchen gelang es, den Roboter problemlos zu bewegen.
Das NXT-Gerät agierte als Übersetzer zwischen dem programmierten Skript in MATLAB und dem Roboter.

Abbildung 2. Farbsensor

III. Hauptteil

A. Konstruktion

Der Sortierroboter wurde aus 3 Teilen aufgebaut. Der unterste Teil des Roboters war die Befestigung am Boden, damit sich der Greifarm ohne Probleme bewegen kann, ohne dabei zur Seite oder nach vorne und hinten zu kippen. Die Basis des Roboters ist in Abbildung 3 zu sehen. In der Basis war Motor A eingebaut (siehe Abbildung 1).
Der zweite Teil des Roboters war der Arm. Er sollte so konstruiert werden, dass er den Greifer nach unten und oben bewegen kann. Er sollte die passende Länge haben, damit er beim Anheben der Bälle nicht gegen den Rest des Konstrukts schlägt. Die Bewegung des Arms wurde durch Motor B realisiert (siehe Abbildung 1). Den Arm selbst zeigt Abbildung 4.
Der letzte Teil des Roboters war der Greifer. Er ist dafür zuständig, die Objekte zu greifen und anzuheben. Die Objekte, die er anhebt, sollen während der Bewegung des Arms nicht herausfallen. Motor C gewährleistet das Anheben und Greifen der Objekte. Den Greifer zeigt Abbildung 5.

B. Programmablauf

Der Roboter kann sich nicht von selbst bewegen. Dem Farbsensor mussten die farbigen Bälle hingehalten werden. Wenn dieser die Ballfarbe erkannt hat, so wurden die Bälle unter dem Greifer hingestellt.

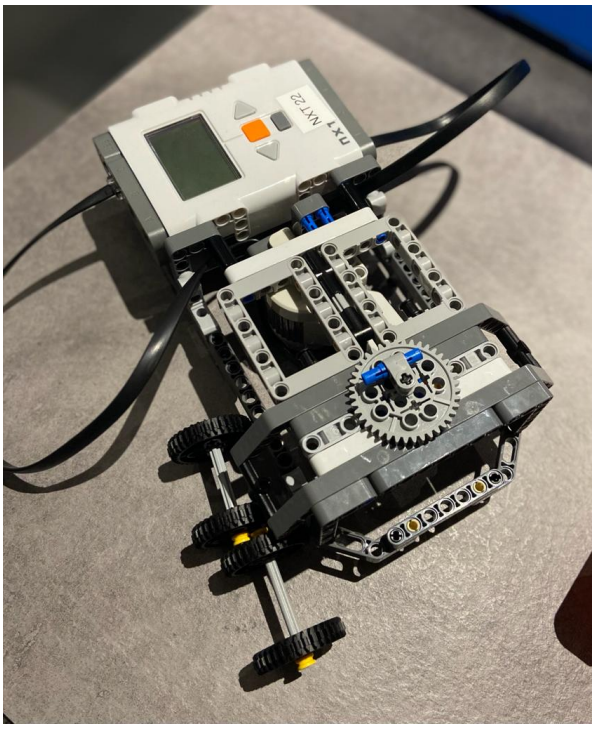


Abbildung 3. Basis des Roboters

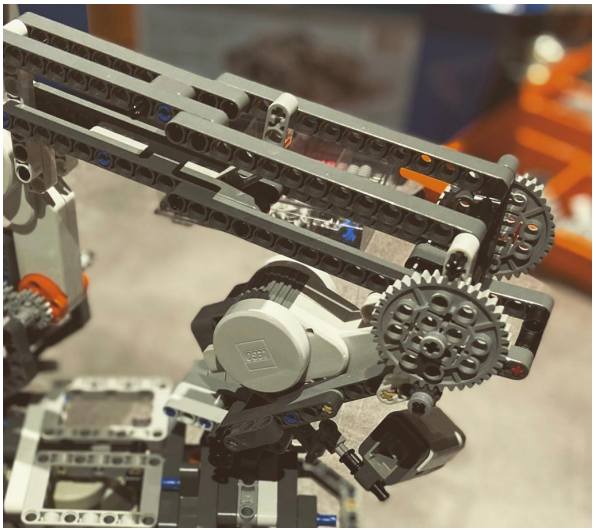


Abbildung 4. Arm des Roboters

Nach dem Erkennen des roten Balls bewegt sich der Arm (Motor B) nach unten, um dann mit dem Greifer (Motor C) den Ball aufzunehmen. Danach bewegt der Arm den Ball nach oben. Danach bewegt sich der Greifarm 90° nach links (realisiert durch Motor A). Eine bereitgestellte Tasse dient dazu, den Ball aufzufangen, nachdem der Arm sich etwas nach unten bewegt hat und der Greifer öffnet, damit der Ball in die Tasse fällt.
Beim blauen Ball schließt sich zuerst der Greifer, bevor der Arm 45° nach rechts bewegt wird. Danach bewegt sich der Arm nach unten, um dann mit Schwung den blauen Ball wegzuschlagen.

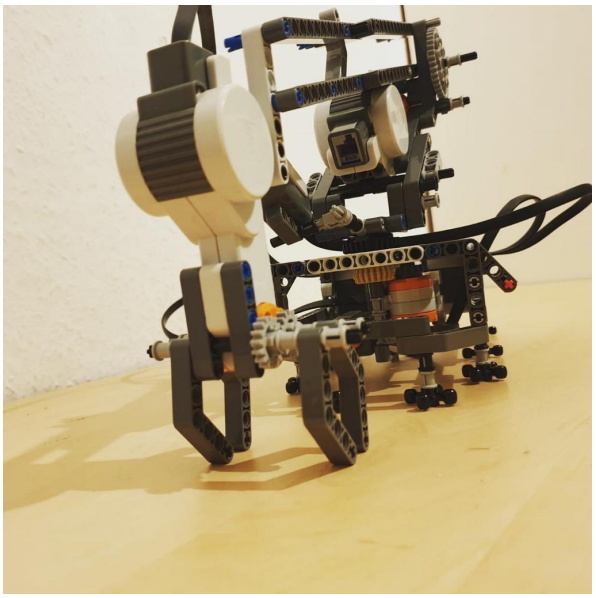


Abbildung 5. Greifer des Roboters

Wenn der Farbsensor den grünen Ball erkennt, so soll der gleiche Ablauf wie beim roten Ball ausgeführt werden; mit dem Unterschied, dass die Tasse 90° rechts vom Roboter abgestellt wird.
Ein beispielhafter Programmablaufplan zur Erklärung einer Prozedur oder einer Routine ist in Abbildung 6 dargestellt.
Im Anhang befindet sich der Programmiercode in Matlab, womit sich die definierten Bewegungen vom Roboter realisieren lässt.

IV. Ergebnisdiskussion

Das anfangs gesetzte Ziel, einen Roboter zu entwickeln, welcher in der Lage ist, verschiedenfarbige Objekte zu erkennen und entsprechend zu sortieren, wurde erfolgreich umgesetzt. Das Projekt wurde innerhalb der vorgegebenen Zeit umgesetzt, auch ohne große Vorkentnisse zm diesem Thema. Eine Verbesserungsmöglichkeit wäre, dass man nicht die Bälle vor den Farbsensor halten muss, sondern dass der Farbsensor selber die farbigen Bälle analysiert. Das würde gehen, wenn mehr LEGO®-Bauteile und auch ein weiterer Farbsensor vorhanden wären. Aufgrund der pandemiebedingten Einschränkungen musste der Roboter zuhause gebaut werden, was die Lage erschwerte. Auch der Kontakt zu anderen Studierenden war eingeschränkt. Der Kontakt wäre wichtig gewesen, um sich gegenseitig helfen zu können, falls etwas nicht klappt oder wenn Bauteile fehlen.
Ein weiteres Problem war der Farbsensor. Er funktionierte nur bei bestimmten Lichtverhältnissen. Das hatte zur Folge, dass die Bälle so nah wie möglich an den Farbsensor gehalten werden mussten, damit eine optimale Aufnahme erzielt werden konnte. Wenn der Ball nicht vom Farbsensor erkannt wurde, konnte das Programm nicht ausgeführt werden.

V. Zusammenfassung und Fazit

Abschließend kann man sagen, dass die Entwicklung des Sortierroboters zufriedenstellend abgelaufen ist. Im Projekt gab

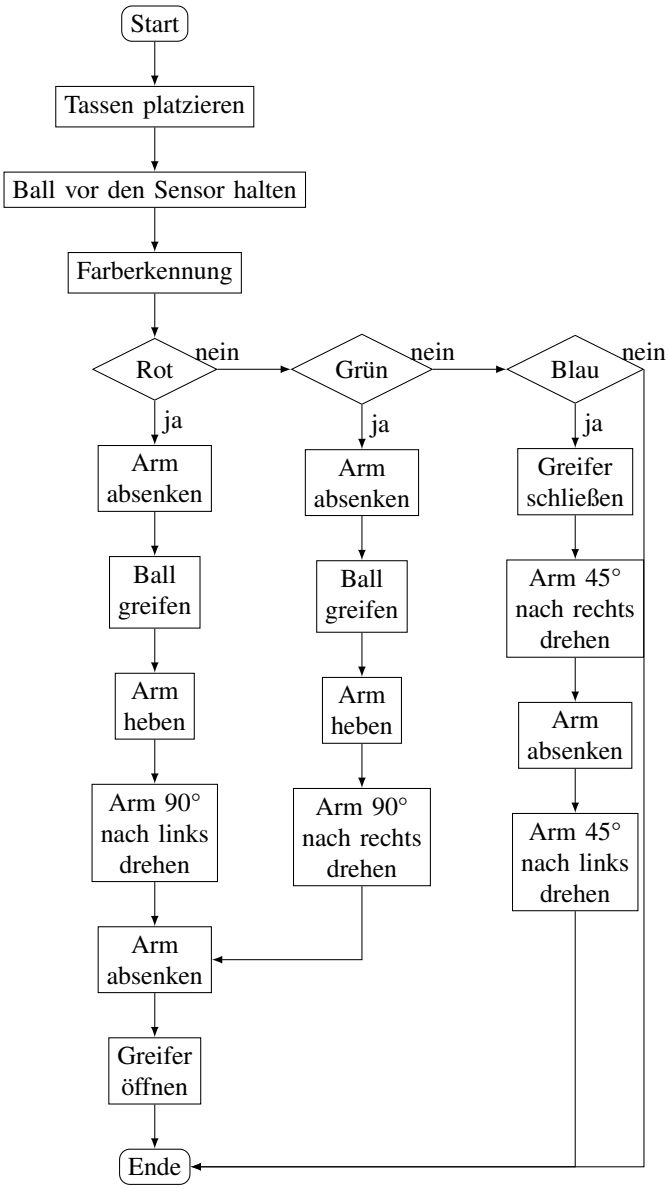


Abbildung 6. Programmablauf

es Herausforderungen, die wir bewältigen mussten. Das Projekt erfordert Grundkentnisse in MATLAB und die Studierenden sollten eigene Ideen integrieren können.
Der Sortierroboter wird schon in vielen Bereichen eingesetzt, da er die Arbeit erleichtert und schneller die Aufgaben erledigt. Beispielsweise kann der Sortierroboter zukünftig zum Sortieren des Mülls verwendet werden, was nicht nur den Menschen einen Vorteil bringt, sondern auch der Natur.

Literaturverzeichnis

[1] A. Herz, ”Die Vor- und Nachteile der Automation durch Roboter,”
https://wandelbots.com/die-vor-und-nachteile-der-automation-durch-roboter/

ANHANG

Die Demonstration des Roboters wurde aufgenommen und in YouTube eingestellt:
https://www.youtube.com/watch?v=Jbd8RcFAnXA

Im folgenden wird der Programmcode gezeigt, mit welchem der Roboter in Bewegung gesetzt wurde:

```
handle = COM_OpenNXTEx('Bluetooth','001653193FC4','bluetooth.ini')
port = SENSOR_3;
OpenNXT2Color(port,'FULL',handle);
pause(3)
color = GetNXT2Color(port,handle);
display('go');
pause(5)
if strcmp(color,'GREEN')
    motorA = NXTMotor('A','Power',20,'TachoLimit',350)
    motorA.SendToNXT(handle);
    pause(3)
    motorB = NXTMotor('B','Power',-20,'TachoLimit',20)
    motorB.SendToNXT(handle);
    pause(3)
    motorA = NXTMotor('A','Power',-20,'TachoLimit',350)
    motorA.SendToNXT(handle);
    pause(3)
    motorC = NXTMotor('C','Power',20,'TachoLimit',300)
    motorC.SendToNXT(handle);
    pause(3)
    motorB = NXTMotor('B','Power',20,'TachoLimit',20)
    motorB.SendToNXT(handle);
    pause(3)
    motorC = NXTMotor('C','Power',-20,'TachoLimit',300)
    motorC.SendToNXT(handle);
elseif strcmp(color,'RED')
    motorA = NXTMotor('A','Power',20,'TachoLimit',350)
    motorA.SendToNXT(handle);
    pause(3)
    motorB = NXTMotor('B','Power',-20,'TachoLimit',20)
    motorB.SendToNXT(handle);
    pause(3)
    motorA = NXTMotor('A','Power',-20,'TachoLimit',350)
    motorA.SendToNXT(handle);
    pause(3)
    motorC = NXTMotor('C','Power',-20,'TachoLimit',200)
    motorC.SendToNXT(handle);
    pause(2)
    motorB = NXTMotor('B','Power',20,'TachoLimit',20)
    motorB.SendToNXT(handle);
    pause(3)
    motorC = NXTMotor('C','Power',20,'TachoLimit',200)
    motorC.SendToNXT(handle);
else
    motorB = NXTMotor('B','Power',-20,'TachoLimit',20)
    motorB.SendToNXT(handle);
    pause(2)
    motorC = NXTMotor('C','Power',20,'TachoLimit',200)
    motorC.SendToNXT(handle);
    pause(2)
    motorA = NXTMotor('A','Power',20,'TachoLimit',330)
    motorA.SendToNXT(handle);
    pause(3)
    motorC = NXTMotor('C','Power',-100,'TachoLimit',350)
    motorC.SendToNXT(handle);
    pause(2)
    motorC = NXTMotor('C','Power',20,'TachoLimit',150)
    motorC.SendToNXT(handle);
    pause(2)
    motorA = NXTMotor('A','Power',-20,'TachoLimit',330)
    motorA.SendToNXT(handle);
    pause(2)
    motorB = NXTMotor('B','Power',20,'TachoLimit',20)
    motorB.SendToNXT(handle);
    pause(2)
end
CloseSensor(port,handle);}
```

Aufbau eines Sortierroboters mit Lego Mindstorms

Thomas Bruckner, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

Abstract— Das Interesse von jungen Menschen auf Wissenschaft zu lenken gestaltet sich schwierig. Dennoch ist Forschung nötig, um den Alltag zu vereinfachen. Einfache jedoch zeitintensive Aufgaben werden oft mit Hilfe von technischen, programmierten Geräten erledigt. In diesem Artikel werden Erfahrungen im Lego Mindstorm NXT Robotics Project der Fakultät für Elektrotechnik und Informationstechnologie der Otto-von-Guericke-Universität Magdeburg dargestellt. Diese beziehen sich auf die Erstellung eines Sortierroboters.

***Schlagwörter*— Lego Mindstorm NXT, MatLab, Programmierung, Robot, Sortierung.**

I. EINLEITUNG

Eine dieser alltäglichen, zeitintensiven Aufgaben ist das Sortieren. Anwendung findet es in jedem Gebiet vom Strukturieren von Dokumenten bis hin zum Recycling. Eine gängige Situation in der das Sortieren scheitert ist die Müllentsorgung. Oftmals werden Abfälle, ohne Rücksicht auf die Umwelt zu nehmen, in Parks oder Besucherzentren nicht sachgerecht weggeworfen. Straßenverschmutzungen kosten Deutschland jährlich 800 000 000 Millionen Euro. Aus Bequemlichkeit wird das Recycling von vielen Menschen nicht mehr ernstgenommen. Kann diese Aufgabe von einem Roboter übernommen werden?
Während des NXT Seminars wurde ein Sortierroboter entworfen. Hierfür wurde der Lego Mindstorm NXT Roboter und das dazugehörige MatLab Programm verwendet. Das ursprüngliche Ziel war es den Roboter die Farben der Papierbälle erkennen zu lassen und diese in unterschiedliche Boxen zu legen. Außerdem sollte die Anzahl der zugeteilten Objekte pro Box angegeben werden. Das Ergebnis ist ein Sortierroboter, der lediglich Papierbälle findet und sie mit einem Greifarm in eine Schachtel legt. Das Paper zeigt die Umsetzung des Projekts, Limitationen und Funktionsweisen auf. Dabei wird es auch zusammen mit entsprechenden Ergebnissen vorgestellt.

II. VORBETRACHTUNGEN

A. MatLab

MatLab steht für „Matrizen Laboratory“ und ist ein Programm, welches mit Matrizen rechnet. Die Programmiersprache ist proprietär, sodass es variabel für verschiedene Geräte genutzt werden kann. Es wird zum Lösen von mathematischen und technischen Problemen verwendet, womit es ein sehr großes Anwendungsgebiet umfasst. Seine Software ist plattformunabhängig und somit vielseitig aufrufbar.

B. Lego Mindstorms NXT

Abbildung 1: Der RGB Sensor
Abbildung 2: Farbsensor innen

Lego Mindstorms NXT ist ein Steuerungscomputer, der diverse Anschlüsse für Sensoren und Aktoren hat. Er hat einen ARM Prozessor. Auf dem Computer befinden sich drei Anschlüsse für Motoren, unterhalb vier weitere für die gegebenen Ultraschall-, Farb-, Tast-, und Tonsensoren. Für das Projekt wurden drei Motoren genutzt, was auch der maximal möglichen Anzahl entspricht.
Der Sensor, der ursprünglich verwendet wurde, war der Lego Mindstorms NXT-Farbsensor, mit dem der Roboter nicht nur zwischen schwarz und weiß, sondern auch zwischen verschiedenen hellen und pastellfarbenen Tönen unterscheiden konnte. Er verwendet drei verschiedenfarbige Leuchtdioden (LEDs), um die Zieloberfläche zu beleuchten und die Intensität jeder von der Oberfläche reflektierten Farbe zu messen. Unter Verwendung der relativen Intensität der jeweiligen Farbreflexion berechnet der Farbsensor eine Farbnummer, die an das NXT-Programm zurückgegeben wird.
Mit dem Ultraschallsensor kann der Roboter Objekte finden, Hindernissen ausweichen, Entfernungen messen und

DOI: 10.24352/UB.OVGU-2021-037

Bewegungen erkennen. Er verwendet das gleiche wissenschaftliche Prinzip wie Fledermäuse: Er misst die Entfernung, indem er die Zeit berechnet, die eine Schallwelle benötigt, um ein Objekt zu treffen und als ein Echo wieder zurückkehren. Der Ultraschallsensor misst den Abstand in Zentimetern und Zoll. Er kann 0 bis 2,5 Meter mit einer Genauigkeit von +/- 3 cm messen.

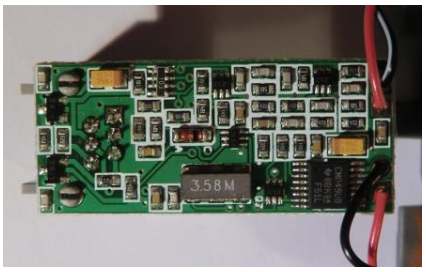


Abbildung 3: Der Ultraschallsensor
Abbildung 4: Ultraschallsensor innen

III. Hauptteil

An den Roboter wurden 3 Motoren angeschlossen. Der erste ist für die 240°-Mobilität des Arms mit der Klaue verantwortlich. Ein zweiter Motor, so dass der Arm sich nach der gegebenen Höhe einstellt und sich zu erkannten Objekten bewegt. Der dritte und letzte Motor war für das Öffnen und Schließen der Klaue zuständig, um nach den Objekten, in diesem Fall farbige Papierbälle, greifen zu können. Der Ultraschallsensor wurde ebenfalls angeschlossen, um die Papierbälle in seiner Umgebung zu erkennen. Da es das Ziel war, die Bälle farblich in unterschiedliche Boxen sortieren zu lassen, kam der RGB-Sensor dazu, welcher die Farben erkennen musste. Zudem sollten die Objekte nach der „Entsorgung“ gezählt werden, sodass beispielsweise nach dem Weglegen des ersten Balls die Zahl „eins“ vom Computer ausgesprochen wird.

Nachdem alles an den Computer angeschlossen und die Aufgabe entsprechend programmiert wurde, kam es zu einem Versuchsdurchlauf. Hierfür wurden rote, grüne und blaue Papierbälle um den Roboter gelegt, die als solche erkannt und in verschiedene Boxen gelegt werden sollten. Allerdings erkannte der Farbsensor keine Farben, da die Funktion zu stark abhängig von Einflüssen, wie Licht, behindert wurde. So wurde bei MatLab nur die Farbe „schwarz“ ausgegeben. Nach mehreren Versuchen, das Problem zu beheben kam es zu einer neuen Zielsetzung, da die Farben nicht erkannt und somit auch nicht sortiert werden konnten. Jetzt soll der Roboter Papierbälle unabhängig von der Farbe in eine Box legen. Da der Ultraschallsensor funktioniert, hat der Greifarm bestehend aus den drei Motoren sich zu den Objekten bewegt, die Höhe an die Größe der Bälle angepasst und mit der Klaue nach ihnen gegriffen. Danach wurden die Papierbälle nach Programm in die Box gelegt.

IV. Ergebnisdiskussion

Am Ende konnte der Roboter drei willkürlich verteilte Papierbälle in seiner Umgebung auffinden, weglegen und zählen.

Sie waren in der Nähe des Roboters und einer Box, in die der Roboterarm die Kugeln legen sollte. Dies ging mit Hilfe des Ultraschallsensors von links nach rechts, welcher nach einem Objekt suchte, bis eines gefunden wurde. Nachdem das Objekt gefunden worden war, wurde der Arm mit der Klaue in einer idealen Höhe positioniert, um dann den Papierball zu nehmen und ihn zur Box zu lenken. Nachdem der Roboter die 3 Bälle bereits an ihrem Platz abgelegt hatte, suchte er zum letzten Mal, ob sich ein Objekt in seiner Nähe befand, und als er nichts fand, gab er mithilfe eines an den Roboter angeschlossenen Computers die genaue Anzahl von Bällen in der Box an. Es wurde ein Roboter erstellt, der eine unterschiedliche Anzahl von Objekten verteilen und zählen kann.

Das größte Problem stellte der RGB-Sensor dar. Seine Präzision war so mangelhaft, dass die Zielsetzung verändert werden musste. Außerdem sind Papierbälle eher elastisch als kompakt und wurden teilweise so stark von der Klaue verformt, dass sie nicht mehr gegriffen werden konnten oder aus ihr rausrutschten.

Manchmal hat die Klaue nur nach einem Papierzipfel gegriffen, weil der Ultraschallsensor nicht nach dem größten Volumen sucht, sondern nur die Oberfläche des Objekts erkennt. Desweiteren stellte die Box für ihn auch ein Objekt dar, sodass die Klaue einige Male versuchte diese aufzunehmen.

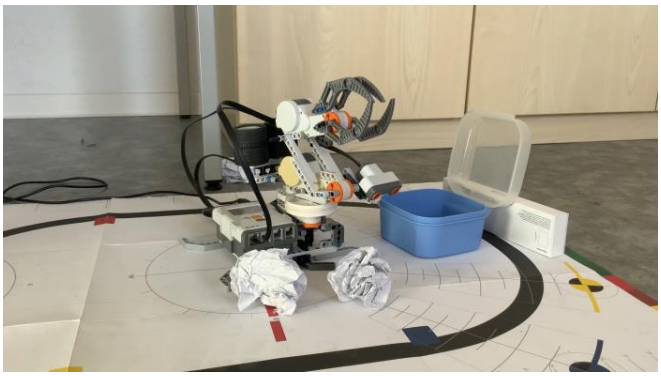


Abbildung 5: Fertiger Roboter mit Greifarm

V. Zusammenfassung

Trotz einiger Probleme und der nicht ganz vollständigen Erreichung des ursprünglichen Ziels, kann das abgeschlossene Projekt auf einige alltägliche und einfache Aufgaben übertragen werden. Mit präziseren Sensoren, könnte man noch komplexere Probleme lösen und die Anwendung im Alltag ausweiten. Trotzdem kann ein bereits so einfaches Programm bei Aufgaben wie der Müllentsorgung behilflich sein.

LITERATURVERZEICHNIS

https://www.welt.de/debatte/kommentare/article197197431/Abfall-in-Parks-und-Strassen-Vermuellung-der-Staedte-ist-typisch-deutsch.html
Entnommen am 20.03.2021 10:05 Uhr

https://de.wikipedia.org/wiki/Lego_Mindstorms_NXT
Entnommen am 20.03.2021 10:28 Uhr

http://www.matlab.rwth-aachen.de/index.php?id=14
Entnommen am 20.03.2021 11:43 Uhr

Abbildung 1: https://www.steinelager.de/de/set/9694-1/mindstorms-nxt-farbsensor
Entnommen am 03.03.2021 12:31 Uhr

Abbildung 2: http://botbench.com/blog/2011/09/16/exposed-lego-colour-sensor/
Entnommen am 03.03.2021 10:36 Uhr

Abbildung 3: https://www.generationrobots.com/de/401180-ultraschallsensor-lego-mindstorms-nxt.html
Entnommen am 03.03.2021 20:15 Uhr

Abbildung 4: http://botbench.com/blog/2011/09/21/exposed-lego-ultrasonic-sensor/
Entnommen am 03.03.2021 14:20 Uhr

Abbildung 5: Fotografiert am 18.02.21 um 23:07 Uhr

LEGO Mindstorms Votingbot

Sebastian Thielecke, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Auch in diesem Jahr 2021 wurde das Legopraktikum trotz komplizierter Situation durchgeführt. Dabei wurde einigen Studierenden der Fachschaft für Elektro- und Informationstechnik die Möglichkeit geboten allein oder als Gruppe eine Idee für einen Roboter zu entwickeln und diese umzusetzen. Dieser Roboter sollte aus LEGO gebaut werden und eine oder mehrere Aufgaben autonom ausführen können. In diesem Paper wird auf die Konstruktion, Programmierung und Entwicklung eines Roboters eingegangen, welcher durch einen Ultraschallsensor Objekte erkennen, diese aufnehmen, in einen Behälter ablegen und anschließend die Anzahl der Objekte, die er sortiert hat per Sprachausgabe ausgeben konnte.**

***Schlagwörter*—Greifarm, Lego Mindstorms, Roboter, sortieren, Sprachausgabe**

I. Einleitung

DIE korrekte und automatische Sortierung und auch Auszählung unterschiedlicher Objekte spielt heutzutage in vielen Teilen unseres Lebens eine große Rolle und wird von Tag zu Tag immer bedeutender. Einige aktuelle Anwendungen sind zum Beispiel bereits existente Wahlroboter, Geldzählmaschinen aber auch Anlagen zur Sortierung von Post und Paketen. Diese Roboter bzw. Geräte dienten auch als Inspiration für den hier thematisierten „Votingbot“. Dieser sollte es schaffen automatisch einen Gegenstand zu erkennen diesen präzise zu greifen und dann wieder automatisiert in einen bereitgestellten Behälter abzulegen. Danach sollte der Vorgang wiederholt werden, bis keine Objekte mehr vor ihm liegen und der Roboter sollte dann per Sprachausgabe wiedergeben wie viele Gegenstände einsortiert wurden. Alle diese Schritte sollten automatisch ausgeführt werden.

II. Vorbetrachtungen

Die Bauteile, die benötigt wurden, um den Roboter zu konstruieren wurden den Studierenden von der Uni in Form eines „Lego-Mindstorms- Kastens“ bereitgestellt. Dieser bestand aus einem NXT-Baustein, drei Servomotoren, unterschiedlichen Verbindungstücken und vier verschiedenen Sensoren (Tast-, Ultraschall-, Licht- und Farbsensor). Der eben erwähnte NXT-Baustein besaß insgesamt sieben Ports von denen drei für die Motoren und 4 für Sensoren gedacht waren [1]. Um diese anzusteuern wurden Befehle benötigt die extern von einem Computer versendet werden mussten. Diese Befehle konnten entweder per Bluetooth oder Kabel an den NXT-Stein geschickt werden.

DOI: 10.24352/UB.OVGU-2021-038

A. MATLAB

Um den Roboter anzusteuern bzw. zu programmieren wurde eine Software namens „MATLAB“ genutzt. Diese wurde von „MathWorks“ entwickelt, um Daten zu analysieren, Algorithmen zu konzipieren und mathematische Modelle zu erarbeiten [2]. Damit eine reibungslose Kommunikation zwischen dem NXT-Baustein und MATLAB entstehen konnte wurde außerdem die dafür entworfene MATLAB-Bibliothek der RWTH Aachen verwendet.

B. Ultraschallsensor

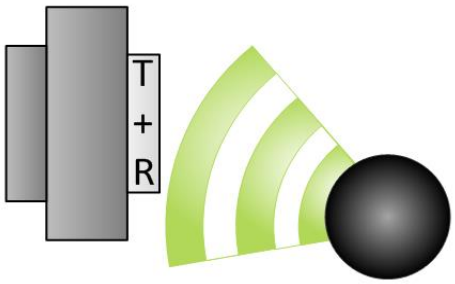


Abbildung 1. Funktionsweise des Ultraschallsensors

Wie zu Beginn des Papers bereits erwähnt, wurde in dem hier thematisierten Projekt ein Ultraschallsensor verwendet, um zu ermitteln wo sich die Objekte bzw. der Behälter zum ablegen dieser befindet. Mit einem solchen Ultraschallsensor kann im Optimalfall die Entfernung, Annäherung, Existenz, der Füllstand und der Durchmesser von Objekten erfasst werden [3]. Dies geschieht dadurch, dass der Sensor wie in Abbildung 1 zu sehen Schallwellen aussendet und das Echo dieser Schalwellen wieder auffängt. Dann kann der Sensor auf Grundlage des Winkels und der Zeit die das Echo braucht, um zurückzukehren die einzelnen Werte ermitteln. Leider ist der in diesem Projekt verwendete Sensor jedoch auf Abstände und Bewegungen beschränkt gewesen die maximal 2,5 Meter entfernt von diesem auftreten durften [1]. Wie bei vielen anderen Messgeräten können auch bei Ultraschallsensoren Ungenauigkeiten bzw. Probleme auftreten. Diese Probleme können auf drei Hauptinitiatoren zurückgeführt werden. Zum einen könnte das Objekt auf das die Schallwellen treffen zu stark geneigt sein und somit das Echo vom Sensor weg reflektieren. Andererseits könnte es auch passieren, dass das Echo zu sehr streut, da der Gegenstand eine

zu raue Oberfläche besitzt. Im Fall, dass mehr als nur 1 Sensor gleichzeitig verwendet wird kann es außerdem auftreten, dass die Sensoren sich gegenseitig mit ihren Echos bzw. Schallwellen überlagern und somit ein verfälschtes Ergebnis liefern [3].

III. Konstruktion und Verwirklichung der Idee

A. *Ideenfindung/Grundidee*

Zu Beginn des Projekts stand vor allem die Ideenfindung im Mittelpunkt. Aus diesem Grund wurden zuerst drei unterschiedliche Ideen erarbeitet auf die nun genauer eingegangen wird. Der erste Gedanke war, dass man etwas konstruieren könnte bei dem der Farbsensor implementiert werden kann, da dieser eine Vielzahl an Anwendungs-möglichkeiten bereitstellen würde. Daraufhin wurde zuerst an einen Art „Ampelbot“ gedacht, welcher erkennen könnte ob die Farben rot oder grün angezeigt werden und dann per Sprachausgabe Befehle bzw. Aufforderungen von sich gibt wie „Stopp! Sie müssen anhalten“ für rot oder „Sie dürfen nun weiter gehen“ bei grün. Diese Idee wurde jedoch recht schnell wieder verworfen da sie recht trivial war und man dementsprechend lieber etwas Komplizierteres umsetzen wollte. Im weiteren Verlauf stand auch die Überlegung im Raum einen Basketball-Roboter zu bauen der immer in den Korb trifft. Allerdings musste auch hier akzeptiert werden das diese Überlegung nicht durchzusetzen wäre auf Grund der zeitlichen und technischen Begrenztheit des Praktikums. Der letzte Gedanke der dann schließlich auch verwirklicht werden sollte, war ein Modell eines Wahlroboters bzw. Votingbots, welcher verschiedenfarbige Gegenstände mittels Farbsensor erkennen kann, diese im Anschluss der Farbe nach in Behälter einsortieren und dann die Anzahl der Bälle pro Gefäß per Sprachausgabe wiedergeben sollte. Zum Anfang gab es auch die Idee diesen umherfahren zu lassen was jedoch nicht möglich war da maximal 3 Motoren verwendet werden konnten und diese für die Steuerung des Arms benötigt wurden.

B. *Konstruktion*

Als entschieden war was für eine Idee realisiert werden sollte ging es auch direkt an die Konstruktion und den Bau des Modells. Der erste Prototyp des Roboters besaß eine Grundfläche aus „Legoarmen“, die für einen stabilen Stand sorgen sollten (siehe Abb. 2). Diese Grundfläche wurde in 2 Hälften aufgeteilt. Eine dieser Hälften war für den NXT-Baustein vorgesehen und auf der andern sollte das Konstrukt des Greifarms stehen (siehe Abb. 2). Der eben erwähnte Greifarm sollte durch drei Motoren gesteuert werden von denen einer für die Rotation des Armes verwendet wurde, ein weiterer für die Auf- und Abbewegung und der letzte für das öffnen und schließen der Kralle. Bereits beim Bau wurde allerdings deutlich, dass der Greifarm ziemlich groß und dementsprechend auch schwer werden würde, weshalb es nötig war Gegengewichte einzubauen, um die Stabilität des Armes zu gewährleisten. Die eingebauten Gegengewichte bestanden wie in Abbildung 2 zu sehen ist aus 4 großen Lego-Rädern die am hinteren Ende des Arms befestigt wurden. Des Weiteren wurde die Kralle mit der die Gegenstände gegriffen werden sollten aus einem oberen Teil der sich nicht bewegen ließ und einem unteren der geöffnet und geschlossen werden konnte konstruiert, wodurch sich größere Objekte wie z.B. Papierkugeln, gut festhalten ließen. Der beim Abschnitt der Ideenfindung erwähnte Farbsensor wurde zu Beginn neben der Kralle befestigt (siehe Abb. 2.1). Leider musste dieser jedoch nach einigen Tests gegen einen Ultraschallsensor ausgetauscht werden der etwas weiter unten neben dem mittleren Motor angebracht wurde (siehe Abb. 1). Auf die speziellen Gründe die für die Änderung des Sensors verantwortlich waren, wird im späteren Verlauf des Papers genauer eingegangen.

Abbildung 2. (1) Roboter mit Farbsensor Frontansicht(links), (2) Seitenansicht des Roboters mit Farbsensor (rechts)

C. *Funktionsweise des „Votingbots“*

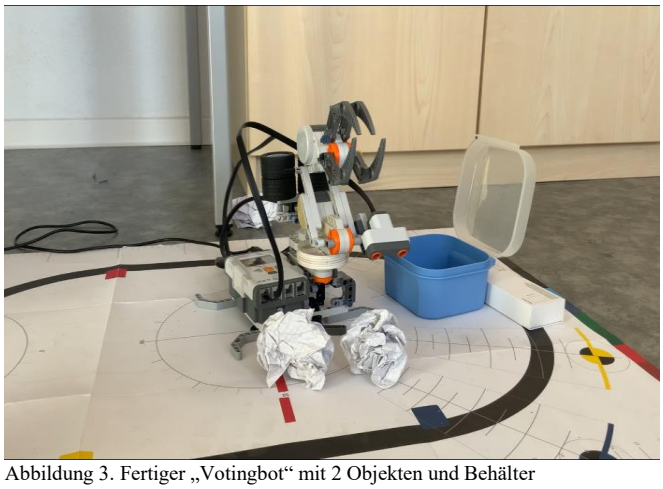


Abbildung 3. Fertiger „Votingbot“ mit 2 Objekten und Behälter

Der Greifarm des Votingbots, zu sehen in Abbildung 3, musste zu Beginn des Vorgangs in eine Grundposition gebracht werden von der aus er in eine bestimmte Richtung rotierte. Diese Rotation führte der Arm solange aus bis der Roboter durch den Ultraschallsensor eine Erhöhung bzw. Änderung der Entfernung messen konnte. Daraufhin wurde eine vorher festgelegte Korrekturbewegung initialisiert, da der Sensor, wie in Abbildung 3 zu sehen ist, nicht genau in der Mitte angebracht wurde. Dann griff er nach dem Gegenstand und drehte sich in die entgegengesetzte Richtung bis er mit dem Sensor wieder eine Veränderung des Abstandes wahrnehmen konnte. Dieses zweite Signal

bedeutete für den Roboter das er erneut eine Korrektur-Bewegung ausführen und dann den Gegenstand in den Behälter ablegen sollte. Im Anschluss fuhr der Arm wieder in die andere Richtung und der Prozess begann von erneut. Wenn der Sensor keinen Gegenstand mehr erfassen konnte, gab er per Sprachausgabe über den Computer aus wie viele Objekte er einsortiert hatte.

D. Sprachausgabe

Ein wichtiger Bestandteil des Projekts war die Sprachausgabe per MATLAB, welche verwendet wurde, um die Anzahl der einsortierten Objekte wiederzugeben. Damit all das funktionieren konnte mussten die verschiedenen Töne zuerst eingesprochen und dann wie in Abbildung 6 zu sehen auf unterschiedlichen Variablen gespeichert werden. Außerdem konnte man, wie in Abbildung 4 zu erkennen ist, die eingesprochenen Töne auch in einem Diagramm visualisieren.

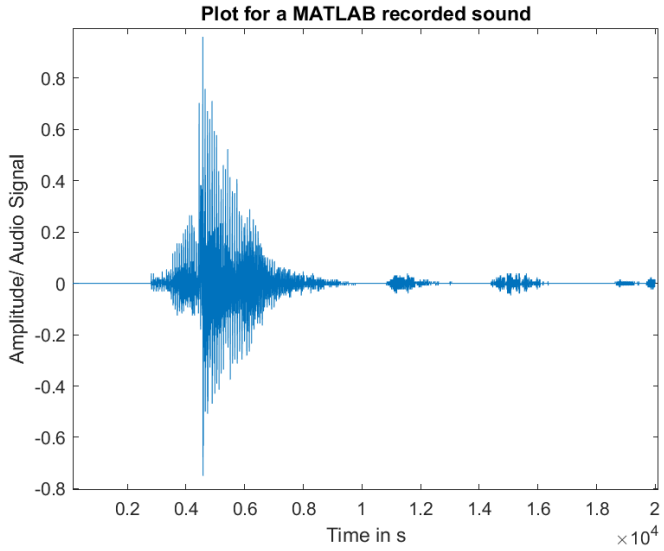


Abbildung 4. Ein per MATLAB selbstaufgenommener Sound

Um zum Schluss die passende Menge an Gegenständen zu benennen wurde die Anzahl der Schleifendurchläufe des Sortiervorgangs ausgelesen und danach diese den Variablen für die aufgenommenen Töne zugeordnet. Die Aufnahme und Ausgabe dieser Töne erfolgte per MATLAB über das Mikrofon und die Lautsprecher des Computers, da der NXT-Stein vermutlich eine schlechtere Audioqualität gehabt hätte.

E. Probleme

Im Verlauf des Projekts musste sich vielen Problemen gestellt werden, auf die nun genauer eingegangen werden soll. Die größte Hürde bestand darin die Objekte korrekt zu erkennen. Deshalb konnte auch leider die anfängliche Idee eines Farbsensors nicht umgesetzt werden und musste später durch einen Ultraschallsensor ersetzt werden. Der Grund dafür war das der Farbsensor nur sehr intensive Farben erkennen konnte und diese auch nur aus einer Entfernung von maximal 3-4 Centimetern. Außerdem war der Sensor zu unverlässlich, um ihn für das Projekt zu nutzen, da dieser bei mehreren Messungen hintereinander immer wieder Schwarz anstatt der eigentlichen Farbe ausgab. Des Weiteren stellte auch die Konstruktion der Kralle und die Wahl der zu transportierenden Gegenstände eine schwierige Aufgabe dar, da mit dieser zuerst kleinere Objekte bewegt werden sollten diese jedoch andauernd aus dem Griff des Arms herausfielen und somit eine nahezu fehlerfreie Sortierung der Gegenstände nicht realisierbar war. Ein anderes Problem in Bezug auf die Konstruktion waren die Kabel die von den Motoren bzw. den Sensoren zum NXT-Baustein verliefen. Dies lag daran, dass die erwähnten Kabel oft bei der Rotation des Roboters aufgewickelt wurden und ab einem gewissen Dehnungspunkt dann gegen die Motoren des Roboters arbeiteten. Das führte dazu, dass der Greifarm gelegentlich nach einer ausgeführten Bewegung durch die Zugkraft der Kabel etwas zurückgedreht wurde und deshalb der Gegenstand nicht korrekt abgelegt oder aufgenommen werden konnte. Auch das präzise Greifen der Objekte war zu Beginn eines der Probleme, da der Sensor etwas seitlich angebracht wurde und daher die gewünschten Gegenstände und Behälter entweder zu früh oder zu spät erkannt wurden was zur Folge hatte, dass der Arm oft danebengriff.

F. Problemlösung

Um das im vorhergehenden Abschnitt erwähnte Problem bezüglich der Kabel zu lösen wurde sich überlegt, dass der Roboter sich nicht komplett um seine eigene Achse drehen müsste, wenn er zuerst in eine Richtung und im Anschluss in die entgegengesetzte fahren könnte. Dadurch wurde erreicht, dass die Kabel die zu Beginn aufgewickelt wurden, wieder abgewickelt werden konnten. Leider ließ sich die Situation dennoch nicht komplett lösen, da gelegentlich trotzdem noch Komplikationen auftraten, welche allerdings nur durch längere Kabel hätten vermieden werden können. In Bezug auf die Problematik mit der Kralle bzw. den Objekten die diese greifen sollte, wurde im Team beschlossen das eine Neukonstruktion der Kralle zu Zeitaufwendig wäre und man deshalb die zu transportierenden Objekte einfach vergrößern könnte. Aus diesem Grund wurde entschieden, dass Papierkugeln, ähnlich zu der aus Abbildung 5, verwendet werden sollten.
Die Schwierigkeit die bei der präzisen Auf- und Ablage der Gegenstände aufgetreten war, ließ sich auch recht trivial lösen indem gemessen wurde wie weit der Greifarm neben das Objekt gegriffen hat und dann eine Korrigierbewegung einprogrammiert wurde, welche den Arm um die gleich Gradzahl in die andere Richtung drehen ließ.

Abbildung 5. Bild einer Papierkugel (ähnlich der im Projekt verwendeten)

IV. Ergebnisdiskussion

Zum Ende des Projekts wurde das Ziel erreicht einen Roboter zu konstruieren der Papierkugeln präzise aufnehmen und ablegen konnte. Auch das Erkennen der Kugeln per Ultraschallsensor lief nahezu reibungsfrei ab. Damit dies funktionieren konnte mussten diese allerdings vorerst von einer Person in einem Viertel- bis Halbkreis um den Roboter angeordnet werden. Des Weiteren konnte das Problem mit den Kabeln leider nicht komplett gelöst werden was zu einer höheren Fehlerquote führte.

V. Zusammenfassung und Fazit

Zusammenfassend muss gesagt werden das es geschafft wurde einen funktionstüchtigen Roboterarm zu entwickeln der von selbst einzelne Gegenstände erkennen, diese Aufnehmen, wieder gezielt an einem bestimmten Ort Ablegen und danach die Anzahl der Objekte per Sound ausgeben konnte.
Wenn mehr Zeit und Mittel zur Verfügung gestanden hätten, wäre vermutlich noch die Kralle des Roboters verbessert worden damit diese auch kleinere Objekte präziser hätte greifen können. Mit einem NXT-Baustein der mehr Anschlüsse für Motoren hat wäre es außerdem durchführbar gewesen den Bot umherfahren zu lassen. Zusätzlich könnte dieser auch, wie in der anfänglichen Idee beschrieben, mit einem (verbesserten) Farbsensor versehen werden, wodurch eine Sortierung der Gegenstände nach Farbe und Größe realisierbar gewesen wäre. Den Möglichkeiten und der Vielfalt von entsprechenden Verbesserungen bzw. Ergänzungen sind demnach nahezu keine Grenzen gesetzt.

Anhang

```
app.recObj=audiorecorder;

               disp('Start speaking.')
recordblocking(app.recObj, 2.5);
disp('End of Recording.');
app.y=getaudiodata(app.recObj);
plot(app.y);


%sound(app.y)
```

Abbildung 6. Kurzer Ausschnitt des Quelltextes zum Aufnehmen und Plotten eines Sounds

Abbildungsverzeichnis

Abbildung 1 – Bangemann, Felix: Vortag Sensoren, https://elearning.ovgu.de/mod/resource/view.php?id=44971, 02.03.2021
Abbildung 5 – Henke, Detlef, „Meine Papierkugel“, http://www.detlef-henke.de/meine-papierkugel/, 04.03.2021

Literaturverzeichnis

[1] Bastgen, Peter, “Lego Mindstorms Education- Bedienungsanleitung”, http://www.bastgen.de/schule/Projektkurs/robotik%209/NXT%20User%20Guide/assets/languages/german/print_all/content/9797_LME_UserGuide.pdf, 01.03.2021
[2] MathWorks, https://de.mathworks.com/, 01.03.2021
[3] Bangemann, Felix: Vortrag Sensoren, https://elearning.ovgu.de/mod/resource/view.php?id=44971, 02.03.2021

Mobiler Solartracker mit LEGO Mindstorms

Lennart Thies Christian Brehmer, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Anlässlich des Projektseminars Elektrotechnik/Informationstechnik an der Otto-von-Guericke-Universität wurde ein Roboter entwickelt, welcher in der Lage ist eine Ebene abzufahren und sich in dieser zum Licht hin auszurichten. In folgender Ausarbeitung wird die Realisierung mithilfe von LEGO Mindstorms und MATLAB sowie die Ansteuerung über eine grafische Benutzeroberfläche näher betrachtet. Des weiteren wird auf die entstandenen Probleme und deren Lösung eingegangen. Zuletzt werden mögliche Weiterentwicklungen und praxisnahe Einsatzgebiete diskutiert.**

***Schlagwörter*—Elektromobilität, LEGO, MATLAB, Solar, Solartracker, Zukunft**

I. Motivation

WÄHREND die Autoindustrie das Zeitalter des Verbrennungsmotors hinter sich lässt und Milliarden von Euro in neue elektrisch betriebene Automobile investiert [1] stellt sich die Frage wie die Menschheit den neuen Bedarf an Energie decken kann. In Deutschland wurden 2020 zwar schon 46% der gesamten Stromversorung aus erneuerbaren Rescourcen gewonnen [2], allerdings ist dies noch unzureichend um den immer näher kommenden irrevisiblen Klimawandel aufzuhalten. Eine mögliche Lösung ist die verstärkte Förderung von Solarenergie. Dafür müssten zuerst die momentanen Probleme dieser alternativen Energiegewinnung gelöst werden [3]. Unter anderem kann nur während des Tages die Sonne als brauchbare Energiequelle benutzt werden. Dies ist nicht ideal, da der größte Energieverbrauch am Abend und in der Nacht auftritt. Nicht bewegliche Solaranmodule verringern zudem ihre Effizienz im Laufe des Tages, da sich der Einfallswinkel der Sonne ändert. Moderne Solaranlagen besitzen daher sogenannte Solartracker (siehe Abschnitt II-A), welche diese zum Stand der Sonne hin ausrichten. Der Roboter soll diese weiterentwickeln und eine weitere Ebene an Mobilität zu solchen Solartrackern hinzufügen. Somit könnte das Problem der sich ändernden Wetterlage und die damit auftretenden Veränderungen der Lichtverhältnisse, welche den Ertrag an Energie drastisch verringern, gelöst werden. Um dies zu realisieren, soll sich der Roboter nicht nur stationär zur Sonne hin ausrichten, sondern auch eine Ebene abfahren, um so aus schattigen Stellen zu entkommen können.

II. Vorbetrachtungen

Im folgendem Abschnitt wird der aktuelle Stand moderner Solartracker kurz vorgestellt sowie ein kleiner Einblick in die verschiedenen LEGO-Mindstorms-Varianten gewährleistet.

A. Solartracker

Um maximale Effizienz moderner Solaranlagen zu erhalten, werden Solartracker verwendet. [4] Diese sind in der Lage ein Objekt, wie z. B. ein Solarmodul, sehr genau zur Sonne hin auszurichten, damit der Einfallswinkel zwischen Solarplatte und dem Sonnenlicht möglich gering ist. Somit soll der Energieverlust sehr stark verringert werden. Man unterscheidet weithin zwischen einachsigen und zweiachsigen Systemen, wobei erstere aufgrund ihres Preises weiter verbreitet sind. Ein einfaches Beispiel ist in Abbildung 1 zu sehen. Obwohl die zweiachsigen System effektiver sind, machen die höheren Produktionskosten eine kommerzielle Nutzung unattraktiver.

Abbildung 1. Solartracker von Suntactics [5]

B. LEGO Mindstorms

LEGO Mindstorms ist eine programmierbare Produktreihe des dänischen Spielwarenherstellers LEGO, um erste Erfahrungen in dem Bereich der Robotik zu erhalten [6]. Es können mithilfe eines zentralen Computers verschieden Sensoren und Motoren angesteuert werden. Angefangen hat die Produktreihe im Jahr 1998 mit dem RCX-Baustein, welcher 2006 von dem leistungsstärkeren NXT-Baustein ersetzt wurde, welchen in Abbildung 2 zu sehen ist. Die aktuellste Variante, der EV3-Baustein, kam 2013 auf dem Markt. Die verschieden Systeme unterscheiden sich in der Leistungsstärke ihrer eingebauten Prozessoren sowie Speichern. Die Anzahl kompatibler Sensoren und Motoren hat sich über die Jahre ebenfalls erhöht [7]. Beide Bausteine besitzen ein auf Linux basierendes Betriebssystem. Wobei der EV3-Baustein standardmäßig mithilfe von Python angesteuert wird, wurde für den NXT-Baustein eine eigene Programmierumgebung namens „NXT-G“ angeboten, welche auf LabView basiert [8]. Es werden auch andere Programmiersprachen wie MATLAB unterstützt. Allerdings benötigt man hierfür z. B. die von der RTWH Aachen entwickelten Bibliothek [9].

III. Realisierung

In diesem Abschnitt wird auf die Realisierung des Roboters eingegangen. Hierfür wird zuerst ein kurzer Überblick der

DOI: 10.24352/UB.OVGU-2021-039

Abbildung 2. Lego-Mindstorms NXT-Baustein [10]

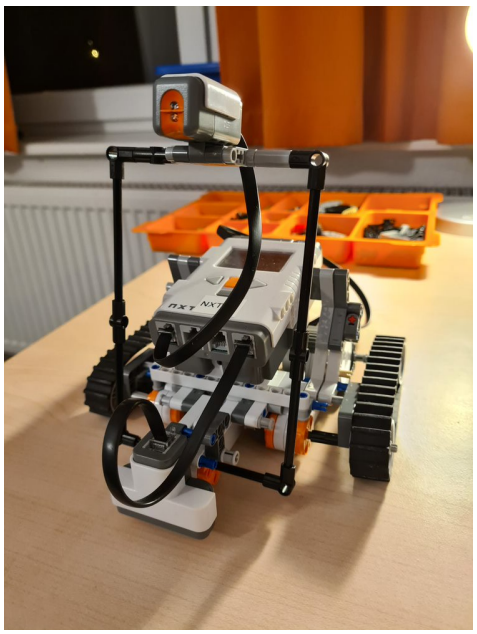


Abbildung 3. Konstruktion des Roboters

Konstruktion betrachtet. Danach werden die automatisierten Prozesse vorgestellt. Zuletzt wird die grafische Benutzeroberfläche erläutert und deren Funktion im Detail erklärt.

A. *Konstruktion*

Bei der Konstruktion des Roboters, welche bei Abbildung 3 zu sehen ist, wurde vor allem auf Funktionalität geachtet. Daher werden zur Fortbewegung zwei Ketten verwendet, welche jeweils von einem Motor angetrieben werden und somit separat ansteuerbar sind. Dadurch ist es dem Roboter möglich sich auf der Stelle zu drehen. Die Konstruktion besitzt zudem einen dritten Motor, welcher über zwei Stangen den Lichtsensor hoch und runter bewegen kann. Besagter Motor wurde zentral an der Vorderseite des Roboters angebracht, um ein eventuelles Anschließen eines Solarmodules zu simulieren. An der Vorderseite wurde ebenfalls zentral ein Ultraschallsensor befestigt. Dieser zeigt in einem 90° Winkel nach unten und soll zur Kantenerkennung dienen. Gesteuert wird der Roboter über den zentral mittig angebrachten NXT-Baustein.

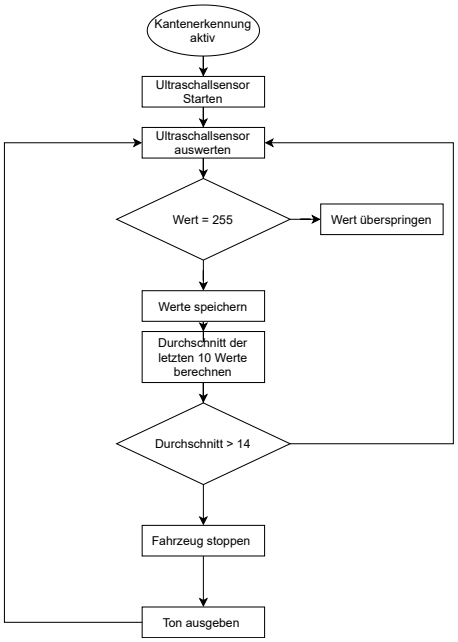


Abbildung 4. Programmablaufplan der Kantenerkennung

B. *Programmierung*

1) Kantenerkennung: Der Roboter verfügt über eine Kantenerkennung, wie in Abbildung 4 beschrieben, welche mithilfe eines Ultraschallsensors realisiert wurde. Hierfür misst der Sensor konstant die Entfernung zum Untergrund und speichert die Werte in einem Array ab. Sobald die ersten zehn Werte abgespeichert wurden, wird der Durchschnitt aus diesen berechnet, um Ungenauigkeiten des Sensors auszugleichen. Aus dem selben Grund wird auch eine Fallunterscheidung bei dem Maximalwert vorgenommen, welche diesen in der Berechnung überspringt. Wenn vom Roboter eine Kante erkannt wird, werden die Motoren, welche die Ketten ansteuern, umgehend gestoppt und es wird ein Ton abgespielt. Dieser wird solange abgespielt bis keine Kante mehr erkannt wird.

2) Lichtwert plotten: Eine weitere Funktion des Roboters ist es, mithilfe des Lichtsensors Werte zu messen, zu verwerten und auszugeben. Letzteres erfolgt über ein Koordinatensystem. Damit das Plotten des Funktionsgraphen in Echtzeit erfolgen kann, müssen die Lichtwerte in einem Array gespeichert und mit jeder Ausführung des Skripts um einen Wert erweitert werden.

3) Reset des Lichtsensors: Der Lichtsensor des Roboters wird beim Start auf einen bestimmte Anfangsposition gebracht, damit dieser nicht vor der Konstruktion sitzt und die Stabilität während des Fahrens beeinflusst. Hierfür wird der Sensor solange hochgefahren bis die Drehzahl des Motors 0 beträgt. Dieses wurde realisiert, indem sowohl die Position des Motors an zwei verschieden Zeitpunkten, als auch die dafür benötigte Zeit gemessen und daraus die Drehzahl berechnet wurde. Um Ungenauigkeiten zu vermeiden, wird auch hier der Durchschnittswert betrachtet. Sobald die ausgerechnete Drehzahl den Schwellenwert erreicht hat, wird das Tacholimit auf 1 gesetzt und der Motor wird gestoppt. Das wird gemacht, da

die Motoren sonst ihn einer Endlosschleife gefangen wären, was die weitere Ansteuerung unmöglich machen würde.

4) Automatisches Ausrichten auf der Stelle: Der Roboter ist in der Lage sich automatisch stationär zum Licht auszurichten. Hierfür dreht er sich einmal um 360° und misst dabei die Lichtintensität mithilfe des Lichtsensors. Gleichzeitig wird die Postion der Motoren ausgewertet, welche die Ketten antreiben. Die Werte werden in Arrays gespeichert und nachdem der Roboter seine erste Drehung vollendet hat dreht er sich zu der Position zurück, an der der höchste Lichtwert gemessen wurde.

5) Automatisches Abfahren einer Ebene: Beim automatischen Abfahren einer Ebene fährt der Roboter zuerst vorwärts bis er eine Kante erkennt (siehe Abschnitt III-B1). Sobald dies geschehen ist, fährt der Roboter kurz rückwärts und dreht sich um 180°. Gleichzeitig setzt er einen internen Zähler hoch, welcher für die Anzahl erkannter Kanten steht. Nun fährt der Roboter wieder vorwärts bis er eine weitere Kante erkennt. Allerdings werden jetzt mithilfe des Lichtsensors die Lichtwerte und die dazugehörige Postionen in einem Array gespeichert. Sobald eine weitere Kante erreicht wurde, schließt sich der Lichtsensor und der Roboter dreht sich wieder um 180°. Anschließend fährt er zu der Postion zurück, an der er den höchsten Lichtwert gemessen hat und dreht sich dort um 90°. Daraufhin wiederholt der Roboter die selben Schritte, um auch hier die Postion mit der größten Lichtintensität zu finden. Wenn diese gefunden wurde, dreht sich der Roboter allerdings nicht mehr um 90° sondern führt die Funktion zur automatischen Ausrichtung auf der Stelle aus (siehe Abschnitt III-B4). Zuletzt wird der Lichtsensor nochmals separat ausgerichtet, indem er von unten nach oben bewegt wird. Der passenden Programmablaufplan für das automatische Abfahren einer Ebene ist bei Abbildung 5 zu sehen.

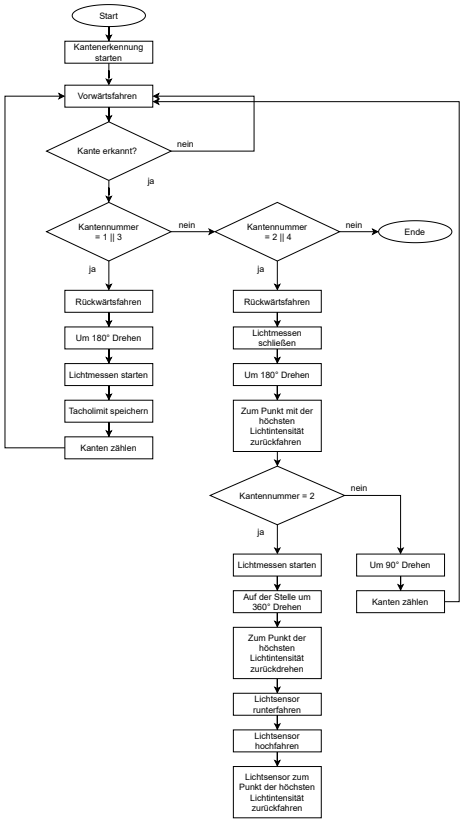


Abbildung 5. Programmablaufplan des automatischen Abfahren einer Ebene

C. Grafische Benutzeroberfläche

Der Roboter kann komplett über die in Abbildung 6 zu sehenden Benutzeroberfläche gesteuert werden, welche mithilfe des in MATLAB vorhandenem App Designer erstellt wurde. Links oben kann der Roboter manuell gesteuert werden. Hierbei kann nicht nur die Bewegung des gesamten Roboters gesteuert werden, sondern auch die Ausrichtung des Lichtsensors. Oben rechts befinden sich Eingabefelder, welche benutzt werden um den Motoren-Parameter wie z. B. den gewünschten Winkel zu übermitteln. Damit diese nicht bei jedem Start erneut eingegeben werden müssen, wurden Startwerte festgelegt. Unter den Knöpfen befinden sich die Ansteuermöglichkeiten für die automatisierten Bewegungsabläufe. Dort kann neben dem automatischen Absuchen der Ebenen (siehe Abschnitt III-B5) auch die Zurücksetzung des Lichtsensors (siehe Abschnitt III-B3) oder das automatische Ausrichten auf der Stelle (siehe Abschnitt III-B4) angesteuert werden. Darunter können links die Drehzahl der Motoren und rechts die vom Lichtsensor gemessenen Werte in Echtzeit graphisch ausgewertet werden. Um dies zu steuern, befindet sich unter jedem Koordinatensystem ein Schalter. Zuletzt kann die Kantenerkennung (siehe Abschnitt III-B1) mittels eines Zustandsknopfes bedient werden.

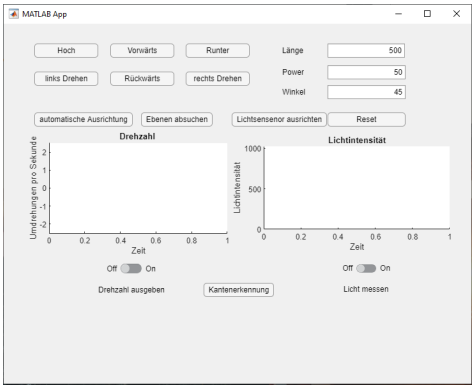


Abbildung 6. GUI zur Steuerung des Roboters

IV. Ergebnisdiskussion

Am Ende des Projektes ist es dem Roboter möglich, eine Ebene wie z. B. einen Tisch abzufahren, ohne von dort herunter zu fallen. Dabei kann er die Lichtintensität mithilfe eines Lichtsensors messen und sich die Position auf dem Tisch merken, an der der Maximalwert gemessen wurde und sich automatisch dorthin zurückbewegen. So ist es ihm möglich sich z. B. automatisch aus einer schattigen Stelle herauszubewegen. Wie in diesem Video [11] zu sehen, fährt der Roboter nicht sehr gerade. Dies liegt zum einem an der Konstruktion, da die Ketten nicht perfekt symmetrisch sind und der NXT-Baustein nicht schnell genug die MATLAB-Befehle bearbeiten kann,

zum anderen daran, dass die Drehung über die Eingabe eines bestimmten Tacholimits erfolgt, welches durch Ausprobieren nur bedingt gut einen genauen Drehwinkel beschreiben kann. Eine Möglichkeit dies zu lösen wäre die Nutzung eines Gyrosensors, um genauere Daten zum Drehwinkel zu erhalten. Ein weiteres Problem ist die Kabelverbindung von Roboter zu Computer. Diese kann theoretisch durch eine Bluetooth-Verbindung ausgetauscht werden. Allerdings führt die dadurch erhöhte Latenz dazu, dass wichtige Funktionen wie die Kantenerkennung (siehe Abschnitt III-B1) nicht funktionieren, da die Daten der Sensoren zu lange brauchen, um vom NXT-Baustein zum Computer gesendet zu werden und der Computer diese zwar auswerten, aber erst zu spät neue Befehle zurück senden kann. Des weiteren ist es dem Roboter nicht möglich, gleichzeitig Funktionsgraphen zu plotten und die Kantenerkennung aktiv zu haben, da beide Funktion mithilfe von Endlosschleifen realisiert wurden und daher nicht parallel ausgeführt werden können. Bei dem automatischem Abfahren einer Ebene ist das kein Problem, da diese Funktion in die der Kantenerkennung (siehe Abschnitt III-B1) mit integriert ist und somit mithilfe von Verschachtelungen in der Schleife mit ausgeführt wird. Eine mögliche Lösung könnte hier die parallel Computing Toolbox [12] oder die Nutzung eines Interrupt-Befehls sein.

V. Zusammenfassung und Fazit

Während des Projektseminars LEGO Mindstorms wurde erfolgreich ein Roboter konstruiert und programmiert, welcher in der Lage ist, automatisch in einer Ebene den hellsten Punkt zu suchen und sich dorthin zubewegen. Im Zuge dessen traten vor allem durch technischen Limitationen der Sensorik und des NXT-Bausteins Probleme auf, welche nur bedingt gelöst wurden konnten. Des weiteren kann man den Roboter über eine grafische Benutzeroberfläche ansteuern. In dieser sind nicht nur die automatisierten Prozesse verfügbar sondern auch die manuelle Ansteuerung sowie die Auslesung der Drehzahl der verschieden Motoren und die vom Lichtsensor gemessene Lichtintensität mittels Funktionsgraphen in Echtzeit. In Zukunft könnte im Rahmen unter dem Lichtsensor ein Solarmodul angebracht werden, welches Elektrizität in einen Energiespeicher senden und so z. B. den Roboter selbst laden könnte.

Anhang

Im Rahmen des Projektseminars wurde eine Playlist erstellt, welche die Funktionen aller entstandenen Roboter darstellt [13].

Quellenverzeichnis

[1] Germis, Carsten: *VW nimmt das Rennen mit Tesla in der Elektromobilität auf*. 2020 https://www.faz.net/aktuell/wirtschaft/auto-verkehr/vw-investiert-35-milliarden-euro-in-elektromobilitaet-17051526.html

[2] Energie, Bundesministerium für Wirtschaft u.: *Erneuerbare Energien*. 2021 https://www.bmwi.de/Redaktion/DE/Dossier/erneuerbare-energien.html

[3] Follett, Andrew: *Top 11 problems for wind and solar*. 2015 https://www.bmwi.de/Redaktion/DE/Dossier/erneuerbare-energien.html

[4] Wikipedia: *Solar tracker*. 2021 https://en.wikipedia.org/wiki/Solar_tracker

[5] Adsala: *Suntastics Solar tracker*. 2016 https://commons.wikimedia.org/wiki/File:Suntactics_solar_tracker.jpg

[6] LEGO: *LEGO MINDSTORMS*. 2021 https://education.lego.com/de-de/products/lego-mindstorms-education-ev3-set/5003400#ev3-set

[7] Wikipedia: *Lego Mindstorms*. 2021 https://de.wikipedia.org/wiki/Lego_Mindstorms

[8] Wikipedia: *Lego Mindstorms NXT*. 2021 https://de.wikipedia.org/wiki/Lego_Mindstorms_NXT

[9] RWTH: *Mindstorms NXT Toolbox*. 2019 https://www.mindstorms.rwth-aachen.de/

[10] Löwenstein, Bernhard: *LEGO MINDSTORMS NXT*. 2013 https://jaxenter.de/lego-mindstorms-nxt-2662

[11] Magdowski, Mathias: *Lichtfinder aus dem Lego-Praktikum 2021 an der OVGU Magdeburg sucht die hellste Stelle einer Fläche*. 2021 https://www.youtube.com/watch?v=9ruyJ5D16nY

[12] TheMathWorks: *Parallel Computing Toolbox*. 2021 https://de.mathworks.com/products/parallel-computing.html

[13] Magdowski, Mathias: *Lego-Praktikum 2021*. 2021 https://www.youtube.com/watch?v=RAflL1oAXsM&list=PLWCaO6Bpqy-7nJr45PQG828J6OdIsfCIi

Kettenfahrzeug mit automatischem Lichtnachführungssystem

Felix Grimm, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

Kurzfassung— Im Rahmen des Projektseminars Elektrotechnik/Informationstechnik soll ein Fahrzeug mit automatischer, sensorgesteuerter, zweiachsiger Nachführung konstruiert werden, dass sich damit selbständig in Richtung der Sonne ausrichten kann. Dies ermöglicht einer Photovoltaikanlage, beispielsweise auf einem Rover, eine möglichst effiziente Stromerzeugung. In dieser Arbeit wird die Idee zur Umsetzung eines solchen Systems sowie die Realisierung mit Hilfe eines über MATLAB angesteuerten LEGO-Mindstorms-Sets beschrieben. Dabei ein Kettenfahrzeug entstanden, das sich sowohl auf der Stelle in die hellste Richtung ausrichten als auch den hellsten Ort auf einer Ebene selbständig ausfindig machen kann.

***Schlagwörter*— Automatisierung, LEGO® MINDSTORMS®, MATLAB, Nachführungssysteme, Photovoltaik.**

I. Einleitung

PHOTOVOLTAIKANLAGEN spielen schon heute als erneuerbare Energiequelle eine wichtige Rolle bei der Stromerzeugung. Laut [1] stellen sie mit einer installierten Leistung von knapp 50 Gigawatt alleine in Deutschland den zweitgrößten Anteil bei der Stromerzeugung durch erneuerbare Energielieferungstechnologien dar. Die Installation von Photovoltaikanlagen als Privatperson ist zudem die einfachste und kostengünstigste Möglichkeit selbst Strom zu erzeugen und bietet gerade in Kombination mit einem eigenen Elektrofahrzeug viele Vorteile, sodass das Aufgreifen dieser Thematik in Bezug auf den Studiengang *Elektromobilität* durchaus Sinn ergibt. Elektrofahrzeuge können durch eingebaute Solarzellen ihre Reichweite signifikant erhöhen und die Kohlendioxid-Bilanz deutlich verbessern. Aber auch in anderen Anwendungsgebieten, wie bei Rovern oder Sonden, sind automatisierte Nachführungssysteme essentiell, da hier die Solarenergie oft die einzige Möglichkeit darstellt, diese über einen langen Zeitraum mit Strom zu versorgen. Außerdem ist eine manuelle Steuerung aufgrund der hohen Übertragungsverzögerungen hier oft nicht mehr möglich oder sinnvoll, sodass automatisierte Prozesse die manuelle Steuerung ablösen. Dementsprechend war es das Ziel dieses Projektes, ein Fahrzeug zu konstruieren, dass einerseits eigenständig den hellsten Ort auf einer Ebene suchen kann, um beispielsweise aus dem Schatten zu fahren, und sich andererseits auf der Stelle optimal ausrichten kann, ähnlich wie sich Sonnenblumen und andere Pflanzen in der Natur verhalten.

II. Vorbetrachtungen

Unter astronomischer Nachführung versteht man in der Regel einen mechanischen Ausgleich der Erdrotation um die eigene Achse. In Bezug auf Photovoltaikanlagen bedeutet das, dass die Solarzellen stets optimal, also meistens senkrecht zur Sonne ausgerichtet sein sollten.

A. *Stand der Technik*

Fest montierte Solaranlagen ohne Nachführungssystem sind zwar weit verbreitet, doch auch Solaranlagen mit Nachführsystemen sind keine Seltenheit, da sie einige Vorteile bieten. So können laut [2] nachgeführte Anlagen den Ertrag um etwa 35 Prozent steigern. Bei solchen Nachführungssystemen unterscheidet man einerseits zwischen sensorgesteuerter und astronomischer Nachführung und andererseits zwischen einachsiger und zweiachsiger Nachführung. Einachsige Nachführungssysteme können sich lediglich um die horizontale oder vertikale Achse drehen, während zweiachsige sich um beide Achsen drehen können. Bei einer astronomischen Nachführung ist die Bewegung der Anlagen bereits fest einprogrammiert, da der Verlauf der Sonne im Verlauf des Tages sowie die Veränderung im Laufe des Jahres bekannt ist. Somit richten sich diese Solaranlagen immer senkrecht zur Sonne aus, obwohl diese Position nicht immer optimal ist. Hierin liegt der Vorteil von sensorgesteuerten Nachführungssystemen. Diese können nämlich die optimale Position durch Helligkeitsmessungen herausfinden und so die Anlagen dementsprechend ausrichten. Das ist vor allem an bewölkten Tagen von großem Vorteil. An Tagen mit stark wechselnder Bewölkung kann so der Ertrag gemäß [3] um bis zu 70 Prozent gesteigert werden.

B. *Idee zur Umsetzung einer sensorgesteuerten Nachführung*

Im Folgenden wird eine recht einfache Möglichkeit zur Umsetzung einer sensorgesteuerten, zweiachsigen Nachführung auf einem Kettenfahrzeug beschrieben. Die Drehung um zwei Achsen lässt sich sehr einfach realisieren. Für die Drehung um die vertikale Achse kann man einfach das Kettenfahrzeug an sich auf der Stelle drehen. Dafür werden zwei Motoren benötigt, die jeweils eine Kette antreiben. Ein dritter Motor kann schließlich noch die Rotation um die horizontale Achse ermöglichen. Als Sensoren lassen sich für eine Drehachse zwei Fotowiderstände verwenden, deren Widerstand mit zunehmender Helligkeit geringer wird. Die beiden Widerstände müssen dabei

DOI: 10.24352/UB.OVGU-2021-040

auf einer zur Drehachse senkrechten Ebene liegen (siehe Abbildung 1). Für zwei Achsen benötigt man also entweder vier Fotowiderstände oder zwei Fotowiderstände, die sich um 90° drehen können. Zeigen die Fotowiderstände in die gleiche Richtung, so kann man anhand der Widerstandsdifferenz feststellen, ob sie auf den hellsten Punkt zeigen. In diesem Fall sollte die Differenz beider Widerstände gleich Null sein. Natürlich würde das auch für den dunkelsten Punkt gelten, sodass man überprüfen muss, ob die Helligkeit einen gewissen Schwellenwert überschreitet. Ist die Differenz ungleich Null, so muss die Solarzelle solange in die Richtung des Sensors, der die höhere Lichtintensität misst, gedreht werden, bis beide die gleiche Intensität messen. Bei der Verwendung von vier Sensoren kann die Drehung in beide Achsen parallel stattfinden, bei zwei Sensoren nur nacheinander. Alternativ lässt sich die Erkennung des hellsten Punktes aber auch mit nur einem Lichtsensor realisieren. Dabei werden nacheinander, während des Drehens aufgenommene Helligkeitswerte miteinander verglichen. So lässt sich bestimmen, ob eine Drehung in eine bestimmte Richtung zu einer Erhöhung oder einer Senkung der Helligkeit führt. Ein einzelner Sensor kann jedoch keine genaue Aussage über die Ausrichtung im Stand treffen.

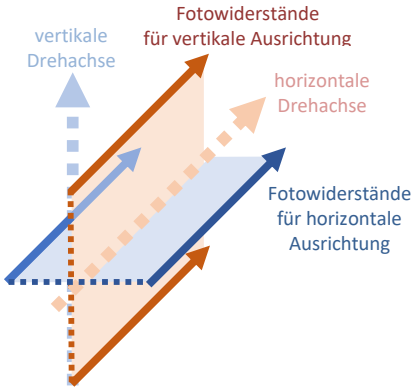


Abbildung 1: Anordnung von vier Fotowiderständen für ein zweiachsiges Nachführungssystem

III. Eigene Umsetzung der Idee

Im Rahmen des Projektseminars war es das Ziel, ein Kettenfahrzeug mit einem zweiachsigen Nachführungssystem zu konstruieren. Als Konstruktionsgrundlage diente das LEGO-Mindstorms-Set mit dem NXT als programmierbarem Steuerungscomputer. Zur Ansteuerung des NXT wurde die Programmiersoftware MATLAB in Verbindung mit einer an der RWTH Aachen erstellten Toolbox verwendet.

A. Realisierung der Idee mit LEGO-Mindstorms

Die Grundidee des zweiachsigen Nachführungssystem ist mithilfe des LEGO-Baukastens gut umzusetzen. Allerdings gilt dies nur für eine sensorgesteuerte Lösung mit einem einzigen Sensor, da im Baukasten lediglich ein Lichtsensor enthalten ist. Außerdem würden vier Sensoren alle Anschlüsse am NXT beanspruchen, sodass keine weiteren Sensoren hätten verbaut werden können. Für eine Lösung aus zwei Sensoren wären insgesamt mehr als drei Motoren nötig gewesen. Ein NXT hat jedoch nur die Möglichkeit, drei Motoren gleichzeitig anzuschließen, sodass schließlich die Lösung mit einem einzigen Lichtsensor am sinnvollsten erschien.

Zudem sollte das Fahrzeug auch noch auf einer Ebene die hellste Stelle finden können, um so beispielsweise aus dem Schatten zu fahren. Für dieses Ziel ist eine automatische Kantenerkennung zur Ermittlung der Ebenengrenzen, z. B. der eines Tisches, nötig.

Zusätzlich zur automatisierten Ausrichtung sollte das Fahrzeug komplett manuell steuerbar sein, um notfalls eingreifen zu können. Um sowohl die Automatisierung als auch die manuelle Steuerung zu integrieren, empfiehlt sich das Erstellen einer grafischen Benutzeroberfläche (GUI).

B. Konstruktion des Fahrzeuges

Die Konstruktion des Fahrzeuges ist recht einfach. Zwei Motoren treiben zwei Räder an, die über eine Kette mit zwei weiteren Rädern verbunden sind. Der Aufbau erfolgte symmetrisch, um zu verhindern, dass bei gleicher Leistung der Motoren die Drehzahlen variieren. Eine dynamische Leistungsanpassung der Motoren an eine bestimmte Drehzahl wäre zwar möglich gewesen, würde aber aufgrund der recht unpräzisen LEGO-Motoren ziemlich ungenau sein. Vorne am Fahrzeug befindet sich ein Ultraschallsensor, der fest verbaut stets senkrecht nach unten zeigt. Dieser dient dem Kantenerkennungsalgorithmus und ermöglicht es so dem Fahrzeug, eine Ebene selbständig abzufahren. Der Lichtsensor sitzt auf einem rechteckigen Gestell, welches eine potenzielle Solarzelle darstellen soll. Ein dritter Motor ist dafür zuständig, diesen Rahmen mitsamt dem Lichtsensor vertikal zu drehen. Da der Lichtsensor auf diesem Gestell angebracht ist, dreht er sich nicht nur um die horizontale Achse, sondern bewegt sich außerdem auf einer Kreisbahn etwas in vertikaler Richtung. Die Bewegung des Lichtsensors ist also an die Bewegung der Solarzelle gebunden, was etwaige Verschattungen des Sensors durch die Solarzellen, gerade bei niedrigem Sonnenstand, verhindern kann.

Die fertige Konstruktion des Kettenfahrzeuges mit den Sensoren ist in Abbildung 2 zu sehen.

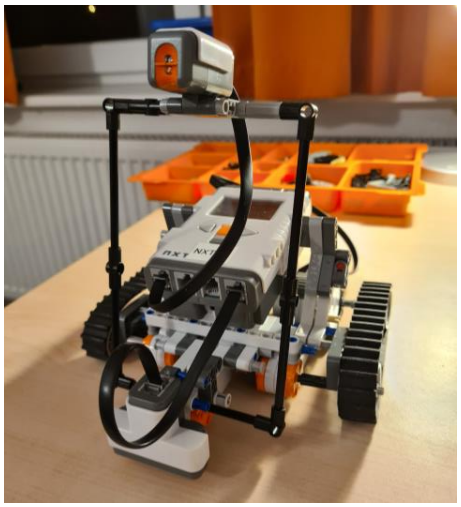


Abbildung 2: Konstruktion des Kettenfahrzeuges

C. *Grafische Benutzeroberfläche*

Mit Hilfe der grafischen Benutzeroberfläche (siehe Abbildung 3) lässt sich das Fahrzeug komplett manuell steuern und die automatischen Ausrichtungsfunktionen können per Knopfdruck aktiviert werden.

Für die manuelle Steuerung gibt es sechs Befehlsknöpfe, mit denen man das Fahrzeug geradeaus und rückwärts fahren sowie nach links oder rechts drehen lassen kann. Rechts davon sind drei Eingabefenster zur Anpassung der Parameter für die manuelle Steuerung. Die *Länge* gibt den Motoren ein bestimmtes Tacholimit für das Geradeausfahren und das Rückwärtsfahren vor, der *Winkel* gibt an, wie weit sich das Fahrzeug nach links oder rechts drehen soll (in Gradzahlen) und die *Power* gibt die Leistung der Motoren von 0 bis 100 Prozent an. Während Power und Tacholimit direkte Parameter sind, die an den NXT und an die Motoren übermittelt werden können, wird das übergebene Tacholimit eines bestimmten Winkels für jeden Motor vorher erst durch eine Funktion berechnet und dann erst übergeben.

Unterhalb der Kontrollknöpfe befinden sich zwei Graphen, die während der manuellen Steuerung die Drehzahl aller drei Motoren oder die Lichtintensität im zeitlichen Verlauf in Echtzeit darstellen können. Während die Drehzahldarstellung hauptsächlich aus Debugging-Gründen implementiert wurde, kann die grafische Darstellung der Lichtintensität auch genutzt werden, um manuell eine optimale Position des Fahrzeuges zu finden. Das Darstellen der beiden Graphen kann jeweils durch einen *Switch-Schalter* unterhalb der Graphen aktiviert oder deaktiviert werden. Der darunter befindliche Kantenerkennungsknopf ist ein *Toggle-Knopf* und aktiviert bzw. deaktiviert die automatische Kantenerkennung durch den Ultraschallsensor.

Die drei linken Knöpfe oberhalb der Graphen dienen der Aktivierung der einzelnen automatischen Ausrichtungsfunktionen. Der rechts davon befindliche Reset-Knopf setzt die vertikale Position des Lichtsensors (und der Solarzelle) wieder zurück.

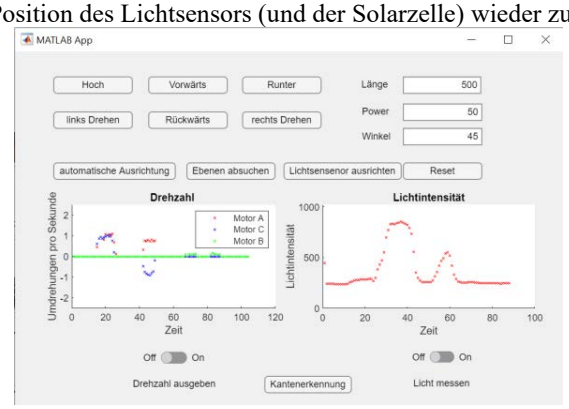


Abbildung 3: Grafische Benutzeroberfläche

D. *Programmablauf der automatischen Ausrichtung*

Der erste Schritt bei der optimalen, automatischen Ausrichtung ist das Absuchen einer Ebene. Der entsprechende Algorithmus ist auch der komplexeste. Da es viel zu lange dauern würde, jede Position auf einer Eben abzufahren, fährt das Fahrzeug lediglich auf zwei senkrecht zueinander verlaufenden Strecken. Zu Beginn fährt das Fahrzeug von einer beliebigen Position aus geradeaus, bis es eine Kante erkennt. Die Kantenerkennung erfolgt über den Ultraschallsensor. Aus zehn Messwerten wird der Durchschnitt gebildet, um zufällige Fehler zu vermeiden. Messwerte von 255, dem maximalen Distanzwert, werden dabei sogar komplett ignoriert, da sie verhältnismäßig häufig auftraten und selbst den Durchschnitt von zehn Messwerten stark verfälschten. Überschreitet der Durchschnittswert einen gewissen Schwellenwert, stoppt das Fahrzeug sofort, fährt etwas zurück und dreht sich um 180°. Nun bekommt das Fahrzeug wieder den Befehl geradeaus zu fahren und der Lichtsensor wird geöffnet. In regelmäßigen Abständen wird der Lichtwert in einem Array abgespeichert. Gleichzeitig wird auch die Motorposition, d. h. die Drehungsanzahl in Grad seit Beginn, in einem separaten Array abgespeichert. Erreicht das Fahrzeug nun erneut eine Kante, so hält es wieder an, fährt ein wenig zurück und dreht sich um 180°. Außerdem wird die Bestimmung von Licht- und Positionswerten gestoppt. Jetzt wird der Index des maximalen Lichtwertes ermittelt. Berechnet man nun die Differenz aus dem letzten Positionswert mit dem Positionsindex mit demselben Index des maximalen Lichtwertes, so erhält man das Tacholimit bis zu der Position, an der der hellste Lichtwert gemessen wurde. Bis zu dieser Stelle fährt das Fahrzeug nun und dreht sich dort um 90° nach rechts. Nun fährt es wieder bis zu einer Kante, wendet dort und misst, während es wieder bis zur nächsten Kante geradeausfährt, das Licht und speichert die Motorposition ab. Genau wie beim ersten Mal werden die Arrays ausgewertet und die Rückkehrposition bestimmt. Dort angekommen hat das Fahrzeug eine vermutlich optimale Position auf der Ebene gefunden, wo es nun stehen bleiben kann. Diese Methode ist natürlich nicht perfekt und kann, gerade bei mehreren Lichtquellen, durchaus fehleranfällig sein. Allerdings stellt sie einen guten Kompromiss aus Zuverlässigkeit und Zeitaufwand dar.

Angekommen an der optimalen Position auf der Ebene kann nun die horizontale Ausrichtung beginnen. Dafür dreht sich das Fahrzeug einmal um 360° und misst dabei wieder in regelmäßigen Abständen das Licht und speichert die Werte in einem Array ab. Auch die Motorpositionen werden wieder parallel abgespeichert. Nach der Drehung werden die Arrays erneut ausgewertet und das Fahrzeug dreht sich bis zur hellsten Stelle zurück.

Nun folgt schlussendlich noch die vertikale Ausrichtung des Lichtsensors. Das Prinzip ist wieder das gleiche wie zuvor. Das Gestell wird zusammen mit dem Lichtsensor nach unten geneigt, während der Lichtsensor in regelmäßigen Abständen Werte aufnimmt und abspeichert. Gleichzeitig wird wieder die Position des Motors abgespeichert, sodass sich der Motor erneut soweit nach oben bewegt, bis der Punkt erreicht wird, an dem der Sensor die größte Lichtintensität gemessen hat. Die Schwierigkeit bei der vertikalen Ausrichtung lag darin, dass sich der Motor beim Neigen des Lichtsensors um einen bestimmten Winkel dreht. Dabei wird davon ausgegangen, dass der Ausgangspunkt des Lichtsensors soweit wie möglich nach hinten gedreht ist. Würde er weiter vorne starten, so würde das Kabel des Lichtsensors irgendwann nicht mehr lang genug sein. Der Motor würde sich also nicht weiterdrehen können, weil das Kabel ihn daran hindert. Daher muss der Lichtsensor vorher stets zurückgesetzt werden. Dies geschieht jedes Mal beim Start des Programms oder manuell durch das Drücken des *Reset-Knopfes*, falls man den Lichtsensor vorher manuell bewegt hatte. Beim Zurücksetzten der vertikalen Position wird der Motor solange in eine Richtung gedreht, bis die Drehzahl des Motors Null ist. Die Drehzahl ist genau dann Null, wenn die

Bewegung des Lichtsensors durch das Zusammenstoßen mit dem NXT verhindert wird. Die Bestimmung der Drehzahl erfolgt über die zeitliche Veränderung der ausgelesenen Motorpositionen.

Der gesamte Ablauf der automatischen Positionsfindung auf einer Ebene ist vereinfacht noch einmal in einem Programmablaufplan im Anhang zu finden.

IV. Ergebnisdiskussion

Während des Projektseminars ist es gelungen ein Fahrzeug zu konstruieren, dass sich sowohl auf der Stelle optimal zur Sonne ausrichten als auch auf der Ebene die optimale Position finden kann. Dieses zweiachsige, automatische Ausrichtungssystem wird vollständig mit nur einem Lichtsensor realisiert. Alternativ lässt sich die Helligkeit auch in Echtzeit grafisch darstellen, während das Fahrzeug manuell gesteuert wird.

Natürlich kamen während des Projektes auch Probleme auf, hauptsächlich bei der Programmierung. Die Ungenauigkeit des Ultraschallsensors führte zu Beginn häufig zu einer fälschlichen Kantenerkennung. Erst durch die Lösung, den Durchschnittswert zu bilden und die Ausgabe höchstmöglicher Distanzwerte zu ignorieren, funktionierte die Kantenerkennung sehr zuverlässig. Die Ansteuerung des NXT über eine Bluetooth-Verbindung stellte sich als weniger vorteilhaft als gedacht heraus. Zwar konnte eine Verbindung hergestellt und Befehle übermittelt werden, jedoch sorgte eine recht hohe Verzögerung dafür, dass Kanten zu spät erkannt wurden und Motoren nach Erreichen des Tacholimits zu spät gestoppt wurden, sodass die Drehung um einen bestimmten Winkel stark variierte. Auch mit einer Verbindung über USB war die Winkelgenauigkeit oft nicht sehr präzise. Selbst beim Geradausfahren über längere Strecken fiel auf, dass das Fahrzeug nicht perfekt gerade fuhr.

V. Zusammenfassung und Fazit

Das entstandene Kettenfahrzeug kann sich, ähnlich wie moderne, sensorgesteuerte, zweiachsige Nachführungssysteme, optimal zur Sonne bzw. zum hellsten Punkt hin ausrichten. Weiterhin kann das Fahrzeug automatisch den hellsten Ort auf einer Ebene finden. Erreicht wird dies durch einen einzelnen Lichtsensor, der während der Bewegung Lichtwerte in einem Array zusammen mit den entsprechenden Motorpositionen abspeichert. Dadurch kann das Fahrzeug anschließend zu der Position der hellsten gemessenen Lichtintensität zurückkehren. Die automatische Kantenerkennung für das Absuchen einer Ebene erfolgt mit Hilfe des Ultraschallsensors, der stets den Abstand zum Boden misst.

In Zukunft könnte die automatische Ausrichtung ohne den Zwang realisiert werden, immer alle Strecken komplett abfahren bzw. sich immer vollständig drehen zu müssen. Wenn beispielsweise ein Helligkeitswert gemessen wird, der schon die maximale Lichtintensität darstellt, könnten nachfolgende Bewegungen abgebrochen werden. Zudem könnte ein Abbruch nachfolgender Bewegungen schon dann erfolgen, wenn die Helligkeitswerte mit zunehmender Bewegung sinken. Durch die Verwendung mehrerer Lichtsensoren wäre es außerdem möglich abzufragen, ob die aktuelle Ausrichtung optimal ist, sodass das Fahrzeug automatisch eine neue Korrektur der Ausrichtung nur dann vornimmt, wenn es nötig ist. Des Weiteren ermöglichen mehrere Sensoren eine schnellere automatische Ausrichtung, da so die Drehung um zwei Achsen gleichzeitig ermöglicht wird. Durch die Verwendung eines Gyrosensors könnte auch die Genauigkeit beim Drehen um bestimmte Winkel sowie beim Geradeausfahren erhöht werden.

Anhang

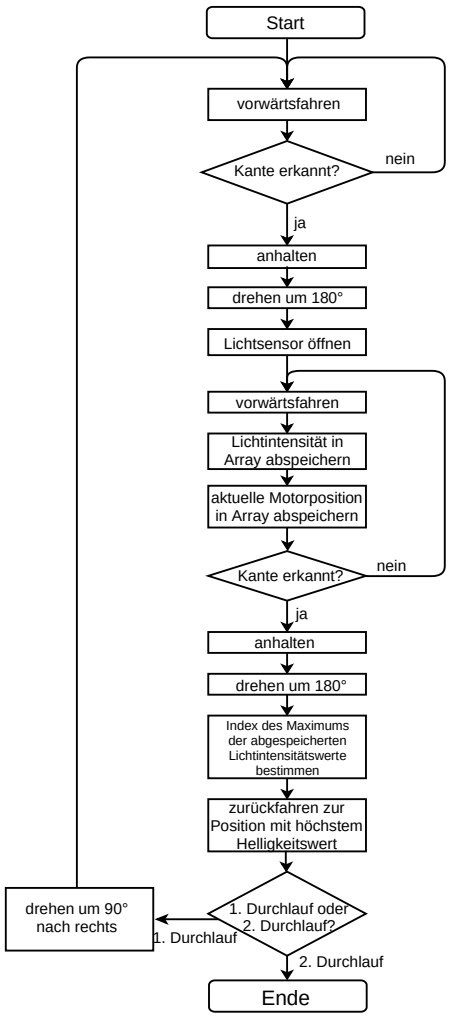


Abbildung 4: Programmablaufplan zur automatischen Ebenenabsuchung

Literaturverzeichnis

[1] Umweltbundesamt, „Erneuerbare Energien in Zahlen," 18. Dezember 2020. [Online]. Available: https://www.umweltbundesamt.de/themen/klima-energie/erneuerbare-energien/erneuerbare-energien-in-zahlen#uberblick. [Zugriff am 2. März 2021].

[2] C. Märtel, „Nachführungssystem für Solar," 9. November 2020. [Online]. Available: https://www.solaranlagen-portal.com/solar/solares-bauen/nachfuehrsystem#:~:text=Damit%20durch%20Solar%20ein%20optimaler,die%20Effektivit%C3%A4t%20der%20Anlage%20steigt.. [Zugriff am 2. März 2021].

[3] C. Märtel, „Vor- und Nachteile verschieder Nachführungssysteme im Vergleich," 9. November 2020. [Online]. Available: https://www.photovoltaik-web.de/photovoltaik/dacheignung/solar-tracker-nachfuehrung-nachfuehrsysteme. [Zugriff am 2. März 2021].

Printi - der süße, zutrauliche Drucker Roboter zum Kopieren von Bildern

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

***Zusammenfassung*—Im Rahmen des jährlichen Projektseminars „Elektrotechnik und Informationstechnik“ der Otto-von-Guericke-Universität wurde ein Fotokopierer für kontrastreiche Schwarz-Weiß-Bilder entwickelt, der mithilfe des NXT-Blockes in der Lage ist Kopien, anzufertigen. In der nachfolgenden Arbeit wird auf die Umsetzung des Roboters genauer eingegangen. Betrachtet werden dabei die Konstruktion, wichtige Abschnitte der Programmierung und die dabei entstandenen Probleme.**

***Schlagwörter*—Kopierer, LEGO-Mindstorms®, Otto-von-Guericke-Universität, Plotter, Printi, Scanner**

I. Motivation

PAPIERDRUCKER sind heutzutage in fast jedem Haushalt und Büro, zu finden. Sie bilden eine der wichtigsten Schnittstellen zwischen digitaler und realer Welt. Die Faszination für solch alltägliche Produkte - auch in der einfachen Ausführung des Papierdruckes in der Ebene - bilden die Grundlage für dieses Projekt. Ziel ist es, einen Kopierer zu bauen, der es schafft, Bilder zu scannen und anschließend zu drucken. Das Projekt soll zeigen, wie komplex unser technisches Umfeld zu Hause ist, um mehr Menschen für die Technik zu begeistern und ein technisches Verständnis von Druckern zu schaffen.

II. Vorbetrachtung

A. Stand der Technik

Drucker findet man in nahezu jedem Anwendungsbereich und auch in dementsprechend vielfältigen Ausführungen. Die drei bekanntesten Arten sind: Laserdrucker, Tintenstrahldrucker und der 3D-Drucker. Laser- und Tintenstrahldrucker haben ihren Anwendungsbereich in der Ebene, um unterschiedliche Arten von Papieren und Plastikfolien zu bedrucken, wohingegen der 3D-Drucker, losgelöst von der Ebene, dreidimensionale Körper aus unterschiedlichen Werkstoffen produzieren kann. Auch hier ist der Einsatzbereich sehr differenziert: Von kleinen technischen Bauteilen, bis hin zu ganzen Häusern.
Laserdrucker werden oftmals mit anderen Arten von Drucken verwechselt, die die Trockentransferelektrographie zum Drucken verwenden. Das Verfahren unterscheidet sich hauptsächlich in der Art und Weise, wie das Bild auf die Trommel übertragen wird. Die Trockentransferelektrographie funktioniert so, dass über die Photonen des Lichtes bestimmte Teile einer drehenden Walze negative Landung verlieren. An diesen positiv geladen Stellen haftet der negativ geladene Toner. Dieses entstandene Abbild auf der Trommel wird anschließend auf das vorher positiv geladene Papier übertragen. Zwei heiße Walzen schmelzen dann den Toner und fixieren die Farbe auf dem Blatt [1].
Der Tintenstrahldrucker überträgt mithilfe unterschiedlicher Arten von Druckköpfen und mit unterschiedlichen Verfahren unter Druck kleine Mengen flüssiger Tinte auf das Blatt, welches an dem Druckkopf vorbeigeführt wird. In den meisten Fällen wird hier das Blatt in y-Richtung bewegt und der Druckkopf wandert in x-Richtung über das Papier. Aufgrund der vielseitigen Methoden, Tinte in unterschiedlichen Mengen und Verläufen auf das Blatt zu übertragen, eignen sich Tintenstrahldrucker besonders für das Drucken von Grafiken und Bildern [2].
Eine ähnliche Art des Druckens sind die sogenannten Plotter. Plotter bewegen sich mithilfe mechanischer Bauteile über das Blatt und können sowohl in runden Formen, als auch in geraden Linien über das Papier wandern und so ein Bild nachzeichnen. Wie bei den Tintenstrahldruckern gibt es auch hier Verfahren, bei denen das Blatt in eine Richtung bewegt wird und sodass der Drucker sich nur auf der anderen Achse zu bewegen brauch. Die Farbe kann hier, beispielsweise wie bei einem Stiftplotter, mit Hilfe des Auf- und Absetzens des Stiftes auf das Blatt übertragen werden. Die Nachteile der beiden letzten Verfahren im Gegensatz zur Trockentransferelektrographie sind sowohl die Geschwindigkeit als auch der Verschleiß.

B. Eigene Lösung

Eine Lösung zur Nachahmung moderner Drucker ist mit LEGO® nahezu unmöglich realisierbar. Die Trockentransferelektrographie und der Tintenstrahldrucker sind daher nicht für dieses Projekt geeignet. Stiftplotter sind die Art Drucker, die mit den vorhandenen Arbeitsmitteln umsetzbar sind. Dabei entstanden ist ein Stiftplotter, welcher mithilfe eines Farbsensors eine Bildvorlage einscannen kann und nach anschließendem händischen wechseln des Blattes, das eingescannte Bild reproduziert. Der Mechanismus des Scanners ist nicht wie bei modernen Drucken losgelöst von dem Druckmechanismus, sondern hängt hier an derselben Brücke, die auch für den Prozess des Druckens zuständig ist.

III. Realisierung

A. Konstruktion

Printi ist ein Stiftplotter mit fixiertem Blatt. Er Besteht dabei aus einem Grundrahmen, in dessen Mitte genug Platz für ein A4 Blatt besteht. Auf dem rechteckigem Grundrahmen bewegt sich eine Brücke, die dazu dient, das Blatt von oben nach unten abzufahren. Der Gesamtaufbau mit Grundrahmen und einem

DOI: 10.24352/UB.OVGU-2021-041

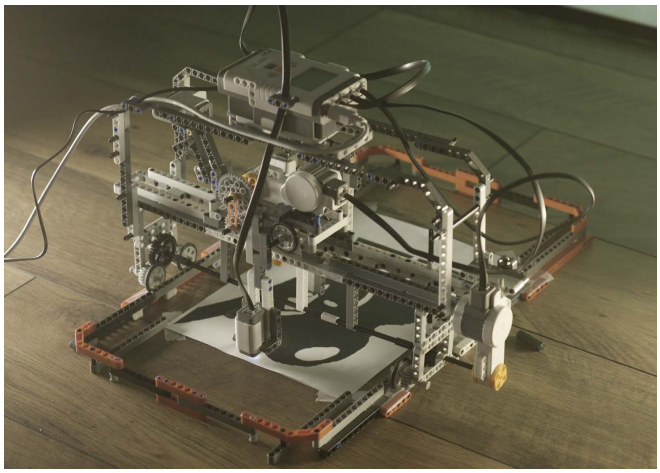


Abbildung 1. Printi

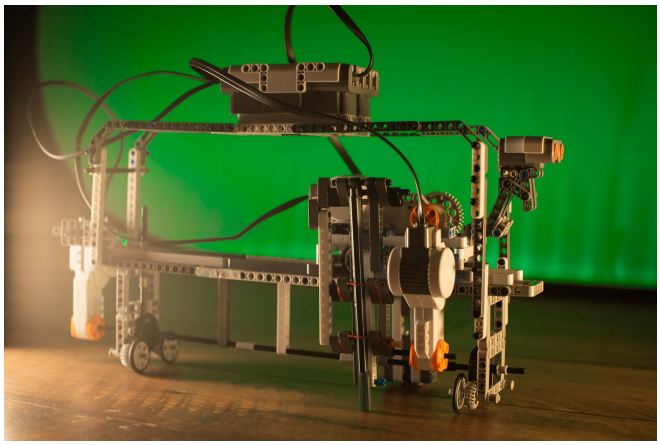


Abbildung 2. Die Brücke

fixierten Blatt sind in Abbildung 1 zu sehen.
Die Brücke, in Abbildung 2 dargestellt, enthält die gesamte Technik des Druckers. Unten sind auf beiden Seiten der Brücke jeweils zwei Räder angebracht, welche durch eine durchgehende Welle mit dem selben Motor angetrieben sind. Das Ziel des Allradantriebs ist es, ein mögliches verkanten der schweren Brücke zu vermeiden.
In der Mitte der Brücke sitzt die Katze. Sie ist ebenfalls motorisiert und kann das Blatt von links nach rechts abfahren. Die Katze sitzt wie ein umgedrehtes U über der Brücke. Auf der einen Seite hängt der Farbsensor fest angebracht herunter (links in Abbildung 3). Auf der anderen Seite hängt die Befestigung für den Stift (rechts in Abbildung 3). Der Stift ist mit einem Motor beweglich montiert, sodass er bei einem schwarzen Wert aufsetzen und bei einem weißen Wert anheben kann.
Ganz oben auf der Brücke befindet sich eine Art Bogen, welcher der Stabilität dient und Platz für den NXT-Stein schafft. Aufgrund der begrenzten Kabellänge und der Gewichtsverteilung ist das der bestmögliche Platz für den NXT-Stein. Printi ist somit in der Lage, sich in x-y-Richtung (in einer Ebene) über das Blatt zu bewegen und über die Bewegung des Stiftes in z-Richtung das Setzen der Farbe zu steuern.

Der angebrachte Ultraschallsensor an der Katze hat keine Bedeutung für den Vorgang des Scannens oder Druckens. Er hat lediglich die Aufgabe, dem Drucker seinen Namen zu verleihen und süß auszusehen.

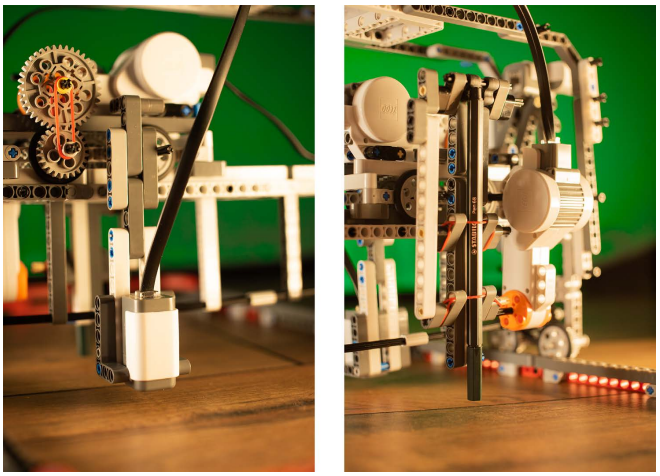


Abbildung 3. Linke und Rechte Seite der Katze, an den der Farbsensor und der Stift befestigt sind

B. Programmierung

Der Prozess des Kopierens besteht aus zwei Schritten: Zum einen das Einscannen und zum anderen das Drucken (in diesem Fall eher Plotten). Beide Algorithmen müssen die Brücke und die Katze ansteuern, damit sie sich bewegt und das Blatt abfährt. Damit der Computer weiß, an welchen Stellen er schwarz und nicht schwarz zeichnen muss, ist das gesamte Blatt in eine Raster Grafik unterteilt, d.h. der Computer fasst an jeder Position den aufgenommenen Farbwert als einen Pixel zusammen. Die Werte werden dann in einer Matrix (hier 68×22) gesammelt. Die Matrix stellt dann eine grobe Darstellung des einzuscannenden Bildes dar. Mithilfe der Matrix setzt der Plotter dann je nach Werteintrag an den einzelnen Positionen einen Strich oder nicht. Im Folgenden werden die beiden Schritte noch detaillierter beschrieben – beginnend mit dem Vorgang des Scannens.
Beim Scannen ist das Ziel, das Blatt abzufahren und währenddessen das Bild einzuscannen. Unnötige Wege sind hier bei beiden Algorithmen zu vermeiden, daher bewegen sich die Brücke und die Katze in einer rechteckigen Sinuskurve über das Papier. Hierbei fährt die Katze einmal von links nach rechts. Anschließend bewegt sich die Brücke eine Position weiter und die Katze fährt wieder zurück auf ihre Ausgangsposition. Zuletzt bewegt sich die Brücke wieder eine Position weiter. Dieser Algorithmus des Abfahrens ist die Grundlage für die hier vorliegende Schleife. Vor jeder Fortbewegung der Katze ruft sie den aktuellen Wert des Farbsensors ab und speichert diesen an der jeweiligen Position in der Matrix. Der Algorithmus, der hier in einzelnen Teilen näher erläutert wird, ist in ausführlicher Form in Abbildung 4 mittels eines Programmablaufplans dargestellt.
Bei genauerer Betrachtung des Programmablaufplans in Abbildung 4 ist auffällig, dass drei Variablen benötigt werden, um

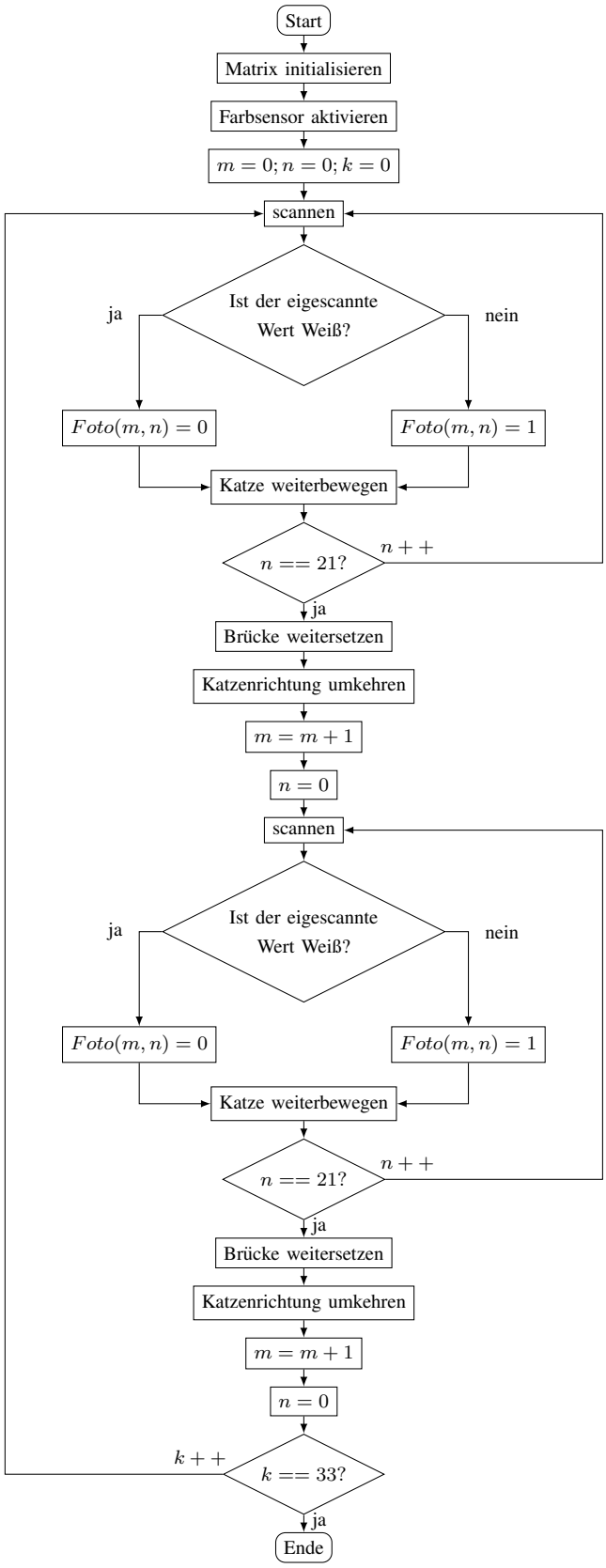


Abbildung 4. Programmablaufplan des Einscannens

die Werte in die Matrix einschreiben zu können und die Brücke und Katze weiterzubewegen. Die Variablen m und n geben das aktuelle Element in der Matrix an. Die Katze bewegt sich mit einer for-Schleife mit der Laufvariable n vorwärts und nutzt dieselbe Variable, um die aktuelle Spalte des Elements in der Matrix anzugeben. Bei der Brücke ist das in nicht umsetzbar, da in einem Schleifendurchlauf die Brücke zweimal bewegt wird (siehe Abb. 4). Aus diesem Grund würde nur jede zweite Zeile beschrieben und bei jedem Durchlauf jeweils einmal überschrieben werden. Die Lösung ist m einzeln zu zählen, um die aktuelle Zeile des Elements zu erhalten, während k die Durchläufe der for-Schleife zählt.

Eine weitere Anpassung des Algorithmus auf die Gegebenheiten der LEGO®-Komponenten war aufgrund der ungenauen Farberkennung des Farbsensors zwingend notwendig. Eine reine Abfrage der beiden Werte Schwarz (=1) und Weiß (=0) war nicht realisierbar, weil die genauen Farbwerte des kontrastreichen Schwarz-Weiß-Bildes nicht erkannt wurden. Deutliche Verbesserungen waren erkennbar, als die Abfrage nicht mehr lautete: „Ist der eingescannte Wert Schwarz? Wenn ja, dann setze eine 1 in die Matrix, andernfalls eine 0“, sondern „Ist der eingescannte Wert nicht Weiß? Wenn Ja, dann trage eine 1 in die Matrix, andernfalls eine 0“.

Der Algorithmus des Druckens (Plottens) ist nahe dem des Einscannens. Die Brücke fährt den gleichen Weg erneut ab, nur, dass in diesem Algorithmus die Matrix ausgelesen wird. Ist in der aktuellen Zelle der Position eine 1, setzt er den Stift auf und bei einer 0 hebt er den Stift an. Um zu verhindern, dass Printi keinen Schritt doppelt macht und somit unregelmäßig zeichnet, muss eine Positionsabfrage des Stiftes eingeführt werden. Die Abfrage ist genauer in Abbildung 5 beschrieben. Sie ist so ausgelegt, dass sie nur die Veränderungen betrachtet, da die Abfrage, ob der Stift so bleiben kann, überflüssig ist, wenn nur die Zustandsänderung für eine Aktion von Bedeutung ist. Die Variable s steht für die Position des Stiftes. Hat s den Wert 1, dann ist der Stift aufgesetzt, 0 wenn er angehoben ist. Die Einführung der Variable s vereinfacht den Prozess des Abfragens, da keine Positionswerte des Motors abgefragt und ausgewertet werden müssen.

IV. Ergebnisdiskussion

Printi erfüllt die gestellten Erwartungen und schafft es, kontrastreiche Bilder zu kopieren und anschließend zu reproduzieren. Ein vollautomatischer Kopierer ist es jedoch nicht, da die Konstruktion kein automatisches Wechseln des Blattes und keine automatische feine Justierung des Stiftes zulässt. Trotz der Grobmotorik und der unzuverlässigen Farberkennung lässt sich das Ergebnis sehen. In Abbildung 6 ist ein Ergebnis des Druckers zu finden. Ganz links im Bild sieht man das Original - Lena - Bild [3]. Da dieses Bild zu fein für den Farbsensor ist, musste das Bild für den Drucker vorbereitet werden. Das kontrastreiche Bild, das Printi verarbeiten kann, ist in der Mitte in Abb. 6 zu erkennen. Rechts sieht man dann das Replikat, welches Printi angefertigt hat.

Der hier verwendete Algorithmus ist nicht effizient gestaltet. Er schafft es, ein Bild zuverlässig zu scannen und wieder zu drucken, dennoch braucht er für diesen Vorgang ca. 50

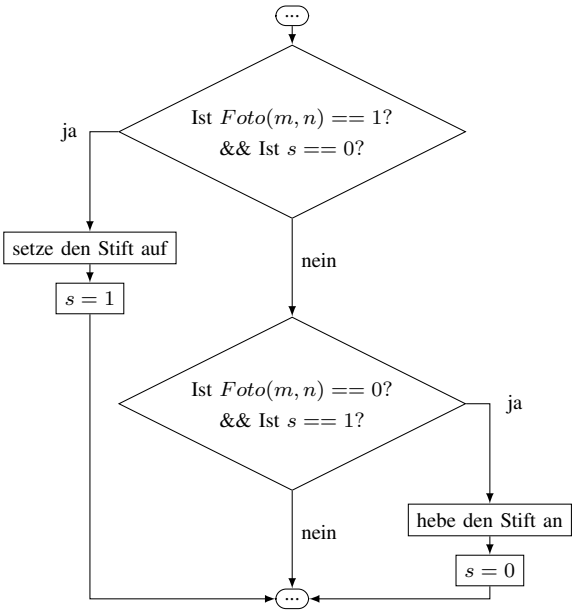


Abbildung 5. Programmablaufplan der Positionsbestimmung des Stiftes

Abbildung 6. Vergleich des Ausgangsbildes mit Printis Replikat

Minuten. Eine bessere Auswertung der Matrix könnte den Prozess des Druckens beschleunigen, dabei müsste aber auch der Algorithmus des Einschreibens verändert werden, da der Algorithmus kein exaktes Abbild in der Matrix schafft, sondern jede zweite Zeile spiegelverkehrt ist. Mit einer Anpassung des Algorithmus ist das Problem aber leicht zu beheben.
Die Resultate von Printi sind nicht mit modernen Laser- oder Tintendruckern vergleichbar, aber es zeigt in Grundzügen die Arbeit eines Druckers und veranschaulicht das Prinzip und den Algorithmus eines alltäglichen Produktes, das essentiell für die Verbindung zwischen realer und digitaler Welt ist.

V. Zusammenfassung

Das anfangs erwähnte Ziel, einen Kopierer zu konstruieren, der Bilder einscannen und diese dann reproduzieren kann, wurde vollständig erfüllt. Zum Einsatz kamen dabei drei Motoren, die sowohl die Brücke als auch die Katze in x-y-Richtung über das Blatt bewegen lassen und den angebrachten in z-Richtung Stift auf- und absetzen können. Für den Prozess des Scannens wurde ein festangebrachter Farbsensor genutzt.
Zur zukünftigen Entwicklung gehören die Verbesserung der Algorithmen des Scannens und Druckens, um eine deutliches Zeitersparnis und die Entwicklung eines Systems zur Auswertung der eingescannten Matrix zu erreichen. Zudem könnte es zukünftig möglich sein, vom Computer Bilder zu drucken und auf den Computer zu scannen.

Literaturverzeichnis

[1] LEIFIPHYSIK, JOACHIM HERTZ STIFTUNG: *Fotokopierer und Laserdrucker*, März 2021. https://www.leifiphysik.de/elektrizitaetslehre/ladungen-felder-mittelstufe/ausblick/fotokopierer-und-laserdrucker
[2] LEIFIPHYSIK, JOACHIM HERTZ STIFTUNG: *Tintenstrahldrucker*. https://www.leifiphysik.de/elektrizitaetslehre/ladungen-felder-mittelstufe/ausblick/tintenstrahldrucker. Version: März 2021
[3] SOUTHERN CALIFORNIA, Signal USC University o. ; IMAGE PROCESSING INSTITUTE, Ming Hsieh Department of Electrical E.: *Girl (Lena, or Lenna), 512x512 pixels, 768 kB, Color (24 bits/pixel)*. http://sipi.usc.edu/database/database.php?volume=misc&image=12#top. Version: 1972

Reinigungsroboter im Modellversuch

Kevin Bursian, Elektrotechnik/Informationstechnik

Otto-von-Guericke-Universität Magdeburg

***Abstract*-In diesem Paper geht es um die Entwicklung eines autonomen Reinigungsroboters, welcher im Rahmen des Projektseminars Elektrotechnik/Informationstechnik 2021 entstand. Im Folgenden wird auf die Konstruktion, die Programmierung, aber auch auf die Probleme, die während der Arbeit aufgetreten sind, eingegangen.**

***Schlagwörter-LEGO Mindstorms, NXT,* Reinigungsroboter, Ultraschallsensor**

I. Einleitung

AUCH in diesem Jahr fand trotz der Corona-Pandemie das „LEGO Mindstorms Projektseminar“ an der Otto-von-Guericke-Universität Magdeburg statt. In diesem Seminar der Fakultät für Elektro- und Informationstechnik bekamen die Studierenden die Möglichkeit, mit Hilfe einer Hardware von LEGO Mindstorms die MATLAB-Software, die Programmiersprache MATLAB und deren Anwendung zu erkunden. Die Robotik gewinnt im heutigen Leben immer mehr an Bedeutung. In der Industrie erleichtern Roboter den Menschen die Arbeit oder ersetzen diese sogar. Roboter übernehmen Aufgaben, die für Menschen zu gefährlich oder unmöglich sind. Sei es bei der Kampfmittelbeseitigung, oder der Erkundung des Weltraumes. Auch im Haushalt etablieren sich autonom arbeitende Maschinen immer mehr.Sei es der Mähroboter im Garten, oder der autonome Staubsauger in der Wohnung. In diesem Zusammenhang hatten die Studierenden die Aufgabe, einen Roboter aus den LEGO-Mindstorms-Sets zu bauen, und diesen im Nachhinein so zu programmieren, dass er eine zuvor gewählte Tätigkeit ausführen kann.

II. Vorbetrachtung

A. Stand der Technik

Im Jahre 2001 brachte die Firma Electrolux den ersten Saugroboter auf den Markt. In den vergangenen 20 Jahren wurde die Technik dieser Geräte immer weiter entwickelt. Während die ersten Geräte nur Staub saugen konnten, gibt es heute schon Varianten mit Bürsten zur Teppichreinigung, oder mit Wischfunktion für glatte Böden. Auch die Orientierungsmöglichkeiten und die Navigation der Geräte wurden verbessert. Zu Beginn der Entwicklung wurden einfache Mechanismen verwendet, welche ohne Elektronik auskamen, um den Roboter bei Berührung mit einem Hindernis wenden zu lassen. Heutzutage sammeln bei hochwertigen Geräten Ultraschall- und Infrarotsensoren Informationen über den Raum und der Roboter speichert diese als eine Art Landkarte zur Orientierung ab. [1]

Abbildung 1: Electrolux Trilobite [2]

B. Lösung mit LEGO Mindstorms

Da es mit LEGO Mindstorms nicht möglich ist, eine Vorrichtung mit Saugwirkung zu bauen, wurde die Aufgabe des Roboters auf das Staubwischen mit Staubtüchern beschränkt. Somit war dieser nur in der Lage, glatte Böden zu reinigen. Außerdem kam ein Ultraschallsensor zur Erkennung von Hindernissen zum Einsatz.

III. Umsetzung

Für die Umsetzung sind zwei Punkte zu betrachten. Zum Einen die Konstruktion und zum Anderen die Programmierung.

A. Konstruktion des Roboters

Beim Bau des Roboters stand die Funktionalität im Vordergrund. Als Basis diente ein Kettenfahrzeug mit zwei Ketten. Jede Kette wurde von einem Motor angetrieben, wodurch im Gegensatz zu Fahrzeugen mit Rädern das Wenden auf der Stelle möglich war. Ein dritter Motor wurde als Antrieb für die Konstruktion, die zum Staubwischen verwendet wurde, genutzt. Diese Konstruktion bestand aus einer Achse, auf der ein Zahnrad befestigt war. Durch die Löcher in diesem Zahnrad wurden, wie in Abbildung 3 zu sehen, Streifen von Staubtüchern gezogen. Damit waren alle drei Motorausgänge des NXT-Bausteins belegt. Ein Ultraschallsensor, welcher am vordersten Punkt des Roboters befestigt wurde, diente der Erkennung von Hindernissen.

DOI: 10.24352/UB.OVGU-2021-042

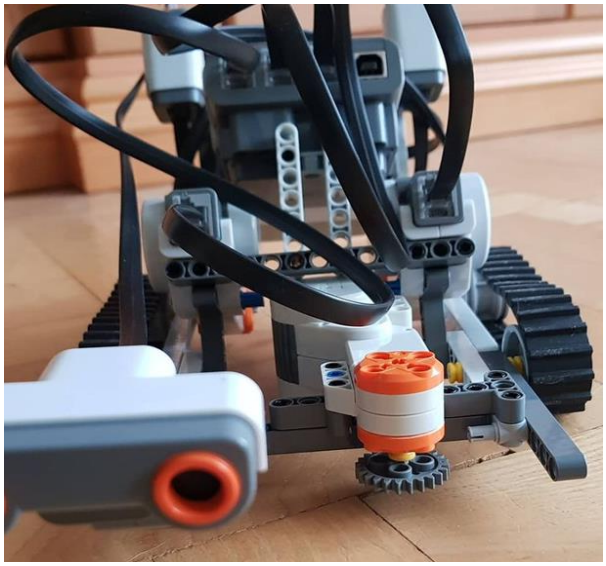


Abbildung 2: Roboter

Abbildung 3: Werkzeug zur Reinigung

B. Programmierung

Im Programm wurde eine while-Schleife verwendet, welche die gemessene Distanz des Ultraschallsensors abfragt. Ist diese größer als 30 Zentimeter, so werden die beiden Antriebsmotoren synchron angesteuert und der Roboter fährt vorwärts. Währenddessen wird weiterhin die Distanz abgefragt. Wird diese kleiner als 30 Zentimeter, so dreht sich der Roboter um 90° nach rechts, fährt ca. 30 Zentimeter, und dreht sich dann nochmals um 90° nach rechts, um in die nächste Bahn zu gelangen. Stößt er auf das nächste Hindernis, erfolgt der selbe Vorgang erneut mit Linksdrehungen. Eine for-Schleife begrenzt die Anzahl der Durchläufe. Damit kann die Laufzeit des Roboters an die zu reinigende Fläche angepasst werden.

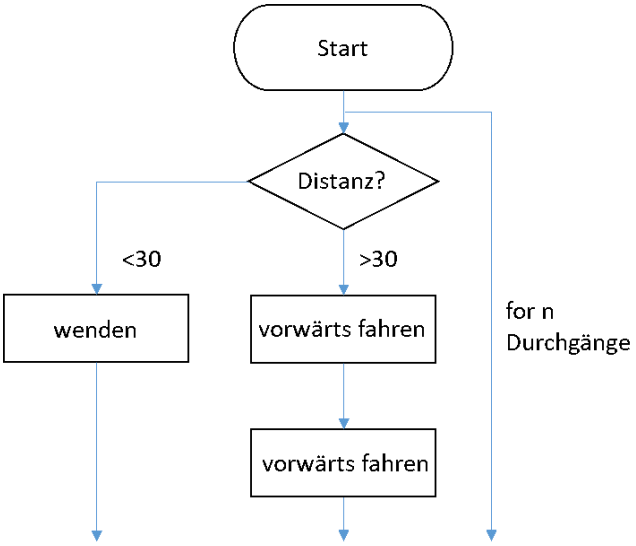


Abbildung 4: Ausschnitt aus dem Programmablaufplan

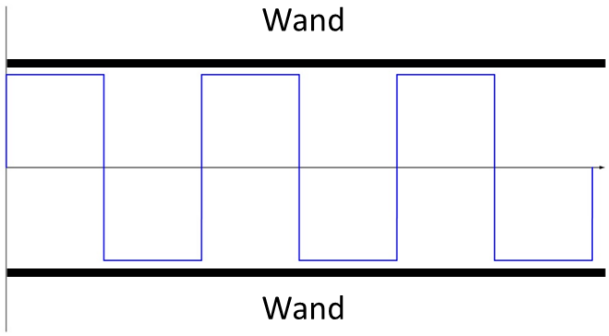


Abbildung 5: Bahn des Roboters

IV. Probleme und Ergebnis

A. Aufgetretene Probleme

Man bekommt während der Arbeit zu spüren, dass es sich bei den LEGO Mindstorms Sets lediglich um technisches Spielzeug und nicht um qualitativ hochwertige Elektronik handelt. Der Ultraschallsensor lieferte teilweise ungenaue Messergebnisse. Teilweise wurde der Sensor nicht angesteuert, was sowohl ein Hardware-, als auch ein Softwareproblem sein kann. Wahrscheinlich hätte dieses Problem mit kurzen Pausen im Programm behoben werden können. Zudem trat auch ein Verschleiß einiger Bauteile auf. Überdehnte Fahrketten zum Beispiel, waren dafür verantwortlich, dass trotz synchroner Motoransteuerung und gleicher Umdrehungszahl, Abweichungen von der Fahrbahn entstanden.

B. Ergebnis

Am Ende des zweiwöchigen Projektes war als Ergebnis ein technisch funktionierender Roboter (Abbildung 2), der jedoch in der Programmierung noch einige Schwachstellen hatte, vorhanden. Er reinigte die Strecke, die er fuhr. Allerdings wurde der Programmablauf nach einer Bahn und dem Umsetzen auf eine andere unterbrochen und es wurde nicht die gesamte Fläche nach dem Muster, wie es in Abbildung 4 zu sehen ist, abgefahren.

V. Zusammenfassung und Fazit

Im Projektseminar erhielt man gute Einblicke in den Umgang mit MATLAB und man lernte aus den vorhandenen technischen Möglichkeiten das Beste zu entnehmen. Es entstand ein funktionierender Roboter, der seine Aufgabe erfüllte. Mit einem überarbeiteten Programm und eventuell einem vierten Motor, was bei dem NXT-Stein nicht möglich ist, hätte man den Roboter noch effizienter arbeiten lassen können. Mit hochwertigeren Bauteilen hätte man auch genauere Steuermöglichkeiten und eine höhere Genauigkeit bei dem Einsatz des Roboters erzielen können.

Literaturverzeichnis

[1] Wikipedia, Staubsaugerroboter, https://de.wikipedia.org/wiki/Staubsaugerroboter , 14.01.2021, Autorenschaft

[2] http://saugrobot.de/electrolux-trilobite-2.php, Saugrbot.de, Robotexpert UG Münsterstr. 336 40470 Düsseldorf

KALIMB0T ein Melodieroboter

Kevin Tom Robert Heine, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Bei einem 2-wöchigen Projekt, entstand der KALIMB0T. Gebaut wurde er mithilfe eines LEGO-Mindstorms- Sets, aus über 650 Teilen. Bewegung erhielt der KALIMB0T durch Matlab und ist so in der Lage einige Melodien auf der Kalimba zu spielen.**

***Schlagwörter*—Kalimba, KALIMB0T, LEGO Mindstorms, LEGO NXT, Musikautomat, Roboter**

I. Motivation

DIE Musik ist seit vielen Jahren ein wichtiger Begleiter der Menschheit. Bereits vor 35.000 Jahren, wurden die ersten Instrumente erfunden [1]. Sogenannte Musikautomaten spiegeln hierbei den Drang der Menschen nach stetiger Weiterentwicklung wieder. Bei diesen Musikautomaten handelt es sich um selbstspielende Musikinstrumente oder Vorrichtungen und Maschinen, welche die herkömmlichen Instrumente ohne einen Spieler spielen. Die ältesten selbstspielenden Instrumente wurden schon im Mittelalter angefertigt und bereits zu Beginn des 19. Jahrhunderts existierten selbstspielende Orchester. [2] In der heutigen Zeit macht es die Technik und ihre Computer, sowie Roboter denkbar einfach eine Maschine zu entwickeln, die in der Lage ist, ein Instrument eigenständig zu spielen. Doch wie leicht ist es mit einem LEGO-Mindstorm-Set, einen Roboter zu entwickeln der ein Instrument spielen kann? Um dieses Vorhaben mit den begrenzten Möglichkeiten dieses Sets zu realisieren, musste ein einfaches Instrument gewählt werden.

Abbildung 1. Kalimba

II. Vorbetrachtungen

Die Wahl fiel auf eine Kalimba (s. Abbildung 1). Hierbei handelt es sich um ein afrikanisches Instrument mit 17 metallischen Klangzungen, die mit den Daumen angespielt werden. [3] Um den Roboter zu konstruieren muss erstmal die Funktionsweise der Kalimba geklärt werden. Durch das Anspielen der Klangzungen mit dem Daumennagel, werden diese in Vibration versetzt, welche wiederrum in dem Klangkörper in einen Ton umgewandelt werden. Dies geschieht mit einer nach unten gerichteten Wischbewegung. Die Klangzungen sind in C-Dur gestimmt, wobei die mittlere Zunge ein C ist. Die darauffolgenden Noten, sind immer abwechselnd links und rechts nach außen hin verteilt. Um die Kalimba nun mit einem Lego-NXT-Roboter spielen zu können, muss der Raum um die Klangzungen abgedeckt werden. Da es für drei NXT-Motoren zu anspruchsvoll ist, die flüssige Bewegung des Daumens beim Anspielen nachzuahmen, muss eine andere Anspielmöglichkeit gefunden werden. Die Klangzungen dürfen dabei nicht zu stark nach unten gedrückt werden, um eine Beschädigung derer zu vermeiden. Es kommen dabei zwei Möglichkeiten in Frage. Die erste wäre eine sehr dünne Holzspitze, hier würde sich zum Beispiel eine Zahnstocherspitze eignen. Diese Spitze rutscht durch eine leichte Druckausübung, an der glatten Oberfläche der Klangzungen ab und bringt diese zum Schwingen. Eine andere Möglichkeit wäre eine flexible Plastikspitze, welche sich auch wieder durch die Ausübung von Druck, so biegt, dass sie frontal abrutscht. Theoretisch könnte man nun an den 3 Motoren, jeweils eine Plastikspitze befestigen und diese vor einer Klangzunge befestigen. Jedoch würden auf diese Weise nur drei Noten anspielbar sein. Drei Noten ergeben aber noch kein Lied, um nun mehr Noten spielbar zu machen, kommen drei Roboter oder Maschinen in Frage.

A. Stiftwalze

Hierbei handelt es sich um das Prinzip der Spieldose (s. Abbildung 2), genauer um eine Walzenspieldose. Eine mit Spitzen bestückte Walze, dreht sich entlang einer Reihe von Tonzungen. Dasselbe Prinzip könnte man bei der Kalimba anwenden, in dem man zwei solcher Walzen jeweils diagonal entlang der Klangzungenreihen anbringt. Diese Walzen würden dann von zwei Motoren zum Drehen gebracht werden. Jedoch ist diese Methode sehr umständlich, da eine Walze nur ein Lied spielen könnte.

B. Faltarm

Inspiriert durch einen Zeichen-Roboter [5], steuern zwei Motoren zwei Arme, die durch Gelenke miteinander verbunden sind. Auf diese Weise, kann eine zweidimensionale Ebene abgefahren werden. Durch eine auf und ab Bewegung dieses Faltarmes, werden dann die Klangzungen angespielt.

C. Schienenbasiert

Durch zwei Achsen, kann man eine 2-dimensionale Ebene waagerecht zu den Klangzungen abdecken. Eine Möglichkeit,

DOI: 10.24352/UB.OVGU-2021-043

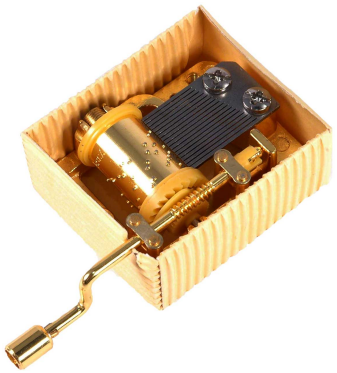


Abbildung 2. Spieldose [4]

diese Achsen zu realisieren, ist mittels Schienen. Über zwei parallele Schienen, die die x-Achse bilden, führt eine Art Brücke, welche als eine weitere Schiene fungiert und damit die Y-Achse darstellt. Die Zungen können dann mit einem Arm, an dem ein Motor befestigt ist, angespielt werden.

III. Der erste Prototyp

Der erste Prototyp (s. Abbildung 3) wurde nach dem Prinzip des Faltarmes konstruiert. An einem Rechteck, wird sowohl Links, als auch Rechts ein Motor befestigt. An diesen beiden Motoren, ist ein nach außen gehender Arm befestigt, welcher in einem 90° Winkel nach innen bewegt werden kann. An jedem dieser Arme, sind mit einem Gelenk weitere Arme befestigt, die am Ende wieder durch ein Gelenk miteinander verbunden sind wie in Abbildung 3 dargestellt. Die Verbindung der zwei Arme, kann so durch die Veränderung der Winkel an den Motoren, entlang der Klangzungen bewegt werden. An der Verbindungstelle wird eine Plastikspitze befestigt, sodass die Zungen angespielt werden können. Das ganze Konstrukt wird so, als Wippe auf einen Standfuß gestellt. Durch einen dritten Motor, kann die Wippe nun auf und ab bewegt werden und die Klangzungen anspielen. Damit sie aber nicht angespielt werden, während sich der Arm bewegt, geschieht dies entweder oberhalb oder unterhalb der Klangzungen.

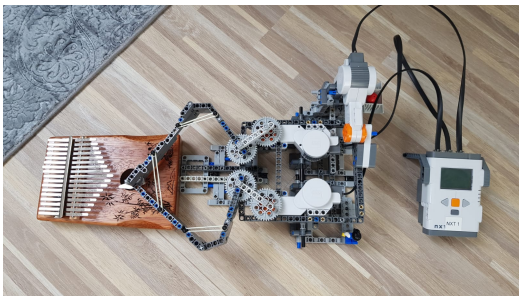


Abbildung 3. Der erste Prototyp

A. Probleme des Prototyps

Um verschiedene Noten anspielen zu können, muss man sehr präzise sein. Dies war auch das größte Problem des Faltarm-Prinzips. Die beiden Motoren, welche die zwei Arme bewegten, hatten etwas Spiel beim Stillstand, sodass sie sich nach der Bewegung, noch um ein Paar weitere Millimeter bewegen konnten. Das Problem konnte durch Gummibänder, die die Gelenke zusammenziehen, minimiert werden. Jedoch war der Prototyp nach einigen Tests immer noch zu unpräzise, sodass er abgebaut werden musste und die Idee des schienenbasierten Roboter konstruiert wurde.

IV. Realisierung

A. Aufbau/Funktionsweise

Um die Ebene der Klangzungen abzudecken, sind zwei Achsen nötig. Die x-Achse wird durch zwei parallel zueinanderstehende Schienen realisiert. Die beiden Schienen sind durch eine Bodenfläche miteinander verbunden. Entlang der Schienen, sind Zahnstangen befestigt. Über die beiden Schienen, fährt eine Art Brücke entlang. An der einen Seite der Brücke ist ein Motor befestigt, der eine Achse dreht. Zudem ist ein Zahnrad an der Achse befestigt, welches über die Zahnstangen der Schiene dreht und damit die Brücke entlang dieser verschieben kann. Damit das Gewicht der Brücke, einfacher an den Schienen entlang verschoben werden kann, verläuft die Achse, an der der Motor befestigt ist, einmal unter der Brücke durch und ermöglicht so, dass auf beiden Schienen, je ein Zahnrad entlanglaufen kann. Die Brücke selbst fungiert ebenfalls als Schiene und stellt somit die Y-Achse da. Auf der Y-Achse fährt nun eine Art Arm entlang, der einem Kran ähnelt. Das Prinzip des Verschiebens entlang der Y-Achse, ist dasselbe wie bei der X-Achse. Entlang der Schiene ist seitlich ein Motor befestigt, welcher eine Achse mit einem Zahnrad entlang der Zahnstange dreht. Somit bewegt sich der ganze Arm entlang der Y-Achse. An dem einen Ende des Arms, ist ein weiterer Motor befestigt, welcher einen Hebel mit einer daran befestigten Plastikspitze nach oben und unten bewegen kann. Am anderen Ende des Armes, ist ein Gegengewicht angebracht. So kann nun der Bereich der Klangzungen, durch die zwei Achsen abgedeckt werden. Durch den Hebel am Arm, können die Zungen angespielt werden. Um die Probleme des ersten Prototyps zu vermeiden, war die Stabilität der Konstruktion von hoher Priorität. Durch möglichst viele Verstrebungen und eine hohe Anzahl an Teilen wird versucht mehr Stabilität zu gewährleisten.

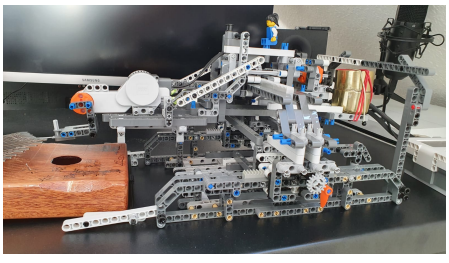


Abbildung 4. Seitenansicht des KALIMB0Ts

B. Positionierung

Der letzte wichtige Punkt vor der Programmierung ist die Positionierung. Die Kalimba selbst wird mit den Klangzungen,

parallel zur X-Achse vor den KALIMB0T gestellt (s. Abbildung 4). Damit die Kalimba immer an dieselbe Stelle vor dem Roboter gestellt wird, existiert ein Gestell am KALIMB0T, welches die Kalimba von drei Seiten umfasst und festhält. Durch das Achsenprinzip des Roboters, ist es möglich den Bereich der Klangzungen als Koordinatensystem darzustellen. Die Position 0/0 beschreibt die äußeren Punkte der Schienen. Diese Position kann per Hand durch Fixpunkte an den Schienen exakt eingestellt werden. Dies ist wichtig, damit der Roboter beim Öffnen des Programms an dieser Position steht. Die Koordinaten der einzelnen Klangzungen, müssen nun millimetergenau abgemessen werden und dann in die Rotationszahl des Motors umgerechnet werden. Doch vorher muss der Umrechnungsfaktor bestimmt werden. Die Rotationszahl des Motors, beträgt für eine Umdrehung 360. Nun muss der Roboter eine Umdrehung ablaufen und die zurückgelegte Strecke muss millimetergenau gemessen werden. Nun kann man ausrechnen, wie groß die Rotationszahl für einen Centimeter zurückgelegte Strecke ist. Mit diesem Faktor kann man nun die Koordinaten der Klangzungen in Rotationszahlen berechnen. Die Realität zeigt jedoch, dass bei der Größendimension, die Messungen mit einem herkömmlichen Lineal zu ungenau sind. Die genauen Koordinaten müssen also mithilfe eines Testprogramms nachjustiert werden. Der KALIMB0T selbst, kann durch die Länge der Y-Achse nur 11 der 17 Töne spielen.

C. Programm

1) GUI des Programms: Die GUI des KALIMB0Ts ist sehr simpel gehalten (s. Abbildung 5). Durch mehrere Buttons, wird die Form der Klangzungen nachgestellt. Dabei steht jeder Button für die Note der Zunge an dieser Position. Beim Drücken eines Buttons wird die Note, die er darstellt auch angespielt. Beim anspielen einer Note färbt sich der jeweilige Button in der GUI für kurze Zeit rot, um dem Benutzer zu signalisieren, welche Note gespielt wird. Links unterhalb der Noten Button befinden sich noch zwei weitere Button. Der Button „Auf Null setzen“ fährt den Arm auf die Position 0/0 des Koordinatensystems, also den Ausgangspunkt, zurück. Und der Button „Start“ setzt die internen Koordinaten auf die Position 0/0. Links neben den beiden Button, befindet sich dann die Songauswahl, welche die vorprogrammierten Songs spielt.

2) Programm: Das Programm ist ebenfalls simpel gehalten, wobei jede Note ihre eigene Funktion hat, um angespielt zu werden. Diese Funktionen, werden dann in den jeweiligen Buttons benutzt um die Noten zu spielen. Um die Funktionsweise des KALIMB0Ts zu erklären ist es sinnvoll, sich die Funktion einer Note anzugucken. In der Abbildung 6, ist der Programmablaufplan einer solchen Funktion dargestellt. Die Positionen der beiden Achsen, sind globale Variablen und somit in der Funktion einsehbar. Dadurch, dass mit der X und der Y Position dasselbe geschieht, kann man es nur für eine erklären. Die Funktion liest die alte Position, welche nun von der neuen Position, die von der gewünschten Note subtrahiert wird. Dabei kommt der Abstand der zwei Positionen heraus. Diesen muss der Motor nun abfahren. Dabei muss aber drauf geachtet werden, dass das Tacholimit welches man dem Motor weitergibt, positiv sein muss. Ist die Zahl des

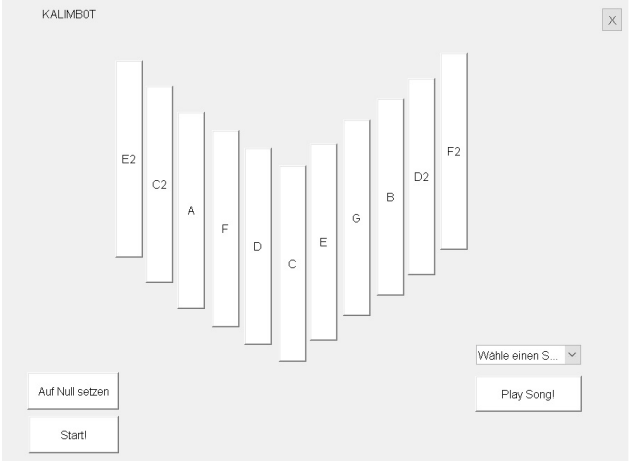


Abbildung 5. Die GUI

Abstandes jedoch negativ, wird die Power des Motors negativ festgelegt und nicht das Tacholimit. Wenn dies nun für beide Koordinaten gemacht wurde, fährt der Arm an die gewünschte Position. Dort angekommen bewegt sich der Hebel am Ende des Arms ruckartig nach oben um die Klangzunge anzuspielen. Danach fährt der Hebel wieder nach unten. Damit die Noten aber nicht zweimal angespielt werden, fährt die X-Achse vor dem Herunterfahren des Hebels ein kleines Stück zurück und nach dem Herunterfahren wieder vor. Am Ende der Funktion werden noch die globalen Variablen der X und Y Position mit der neuen Position überschrieben. Die eben erklärte Funktion ist so für jede der Noten erstellt. Der einzige Unterschied sind die Zahlenwerte der Positionen.

3) Ein Lied spielen: Durch die vorgeschriebenen Funktionen für jede Note wird die Aufgabe ein Lied zu spielen stark vereinfacht. Jedoch ist es immer noch sehr kompliziert und mühsam. Das größte Problem hier, sind die richtigen zeitlichen Abstände der Noten zu treffen. Um dies zu realisieren ist es sinnvoll, die Noten für das gewünschte Lied durch die Funktionen zu schreiben. Dabei kann man nun die zeitlichen Abstände der gespielten Noten mit dem tic - toc - Befehl von Matlab zu messen. Außerdem muss man auch die zeitlichen Abstände der Noten im Lied messen. Mit diesen Werten kann man nun hochrechnen, wie man im Programm die Pausen setzen muss. Dabei schaut man sich den kürzesten Abstand der Noten im Lied an und die dazugehörige Zeit, die der Roboter braucht um von der einen Note zur anderen zu fahren. Diesen Abstand nimmt man jetzt und multipliziert ihn jeweils mit allen zeitlichen Abständen aus dem Lied. Nun hat man die zeitlichen Abstände für die Noten in der Geschwindigkeit des KALIMB0Ts. Um die Pausen im Programm auszurechnen, muss man nun nur noch die Zeiten, die der Kalimbot zum Fahren zwischen den zwei Noten benötigt, von den ausgerechneten Abständen abziehen. Das Ergebnis ist nun die jeweilige Pausenzeit, die das Programm einlegen muss. Falls bei dieser Subtraktion die Pausenzeit negativ ist, müssen die Abstände neu berechnet werden.

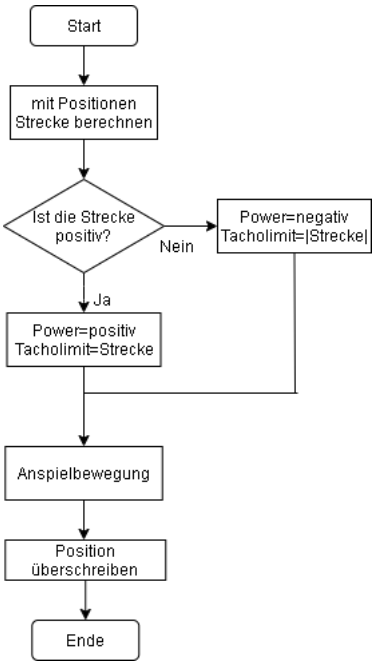


Abbildung 6. Programmablaufplan für eine Note

V. Ergebnisdiskussion

Es wurde ein Roboter konstruiert und programmiert, welcher dazu in der Lage ist eine Kalimba zu spielen. Jedoch ist die Ausführung nicht wie erwünscht. Obwohl das anspielen von einzelnen Noten perfekt funktioniert und ansehbar ist, ist das Spielen eines Liedes eher misslungen. Um die Positionen der Noten präzise zu erreichen muss dies recht langsam geschehen. Um eine zwölf Sekunden lange Melodie in den richtigen zeitlichen Abständen zu spielen braucht der KALIMB0T ganze 5 Minuten. Das größte Problem stellt dabei der Aufbau der Kalimba da. Um zum Beispiel zwei schnell aufeinander folgende Noten zu spielen Braucht der KALIMB0T manchmal lange, da sie von der Position her weit auseinander stehen. Um die Melodie nun richtig zu hören, muss man ein aufgenommenes Video in ungefähr 16-facher Geschwindigkeit abspielen. Ein Problem dabei ist, dass so der Klang der Kalimba extrem verzerrt wird. Ein weiteres Problem des KALIMB0TS ist die Genauigkeit. Um die nur wenige Millimeter von einander entfernten Klangzungen anständig anspielen zu können, ist eine hohe Präzision gefragt, welche nur schwer durch ein LEGO-Mindstorm-Set zu verwirklichen ist. Der sonstige Aufbau des Programms und der GUI ist jedoch gut gelungen und den Vorstellungen entsprechend.

VI. Fazit

Die Frage für dieses Projekt war, ob es möglich ist eine Kalimba mithilfe eines LEGO-Mindstorm- Sets zu spielen. Die Antwort darauf ist, dass es durchaus möglich ist wie es der funktionierende KALIMB0T zeigt. Um einzelne Noten zu spielen ist es auch ein anschauliches Projekt. Wenn man jedoch ein Lied spielen möchte, ist es durch die Trägheit des KALIMB0Ts nicht sinnvoll das LEGO-Set dazu zu benutzen eine Kalimba zu spielen.

Anhang

```
function playE2()
global handle MotorA MotorB MotorC posx posy
if posx==0
    posx=153
else
    posx=153-posx
end

if posy==0
    posy=60
else
    posy=60-posy
end

if posx>0
    MotorA.Power=20;
    MotorA.TachoLimit=posx;
    MotorA.ActionAtTachoLimit = 'HOLDBRAKE';
else
    MotorA.Power=-20;
    MotorA.TachoLimit=abs(posx);
    MotorA.ActionAtTachoLimit = 'HOLDBRAKE';
end

if posy>0
    MotorB.Power=20;
    MotorB.TachoLimit=posy;
    MotorB.ActionAtTachoLimit = 'HOLDBRAKE';
else
    MotorB.Power=-20;
    MotorB.TachoLimit=abs(posy);
    MotorB.ActionAtTachoLimit = 'HOLDBRAKE';
end

MotorC.Power=-20;
MotorC.TachoLimit=50;
MotorA.SendToNXT(handle);
MotorB.SendToNXT(handle);
MotorB.WaitFor();
pause(0.5)
MotorC.SendToNXT(handle);
MotorC.WaitFor();
MotorA.Power=-10;
MotorA.TachoLimit=40;
MotorA.SendToNXT(handle);
MotorA.WaitFor()
MotorC.Power=20;
MotorC.TachoLimit=50;
MotorC.SendToNXT(handle);
MotorC.WaitFor();
MotorA.Power=10;
MotorA.TachoLimit=40;
MotorA.SendToNXT(handle);
MotorA.WaitFor()
posx=153
posy=60
```

Abbildung 7. Quelltext einer Notenfunktion

Literaturverzeichnis

[1] Wikipedia, the free encyclopedia: *Musik*. https://de.wikipedia.org/wiki/Musik. Version: März 2021

[2] Wikipedia, the free encyclopedia: *Mechanischer Musikautomat*. https://de.wikipedia.org/wiki/Mechanischer_Musikautomat. Version: März 2021

[3] Wikipedia, the free encyclopedia: *Lamellophon (Kalimba)*. https://de.wikipedia.org/wiki/Lamellophon. Version: März 2021

[4] Bild: *Spieldose*. https://experimentis-shop.de/media/image/32/35/1e/AK01030-Spieldose-musikdose.jpg. Version: März 2021

[5] Avenel, Christophe: *EV3 Print3rbot*. https://www.youtube.com/watch?v=9pjpQoZoW6E&t=3s. Version: März 2021