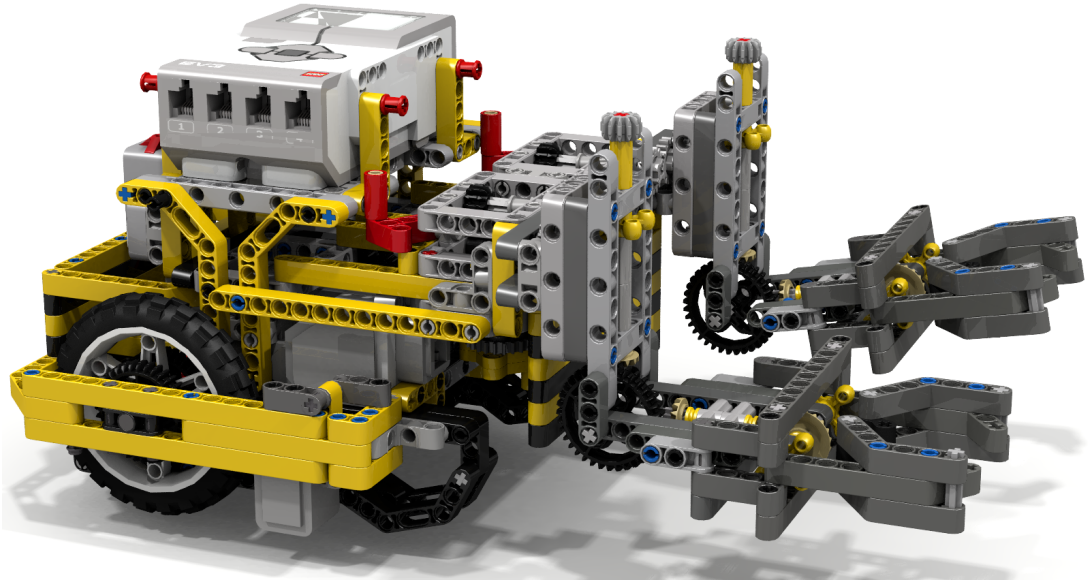
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

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 3 vom Wintersemester 2019/2020

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-2020-050
ISSN:	2629-6160
Redaktionsschluss:	Mai 2020
Seminarzeitraum:	10.–21. Februar 2020
Bezug:	Open Access, Digitale Hochschulbibliothek Sachsen-Anhalt `http://edoc2.bibliothek.uni-halle.de/`

1. Auflage, Magdeburg, Otto-von-Guericke-Universität, 2020

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

Autonomes Parken mit Parklückenerkennung

Jannes Bützer, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Anlässlich des Projektseminars Elektrotechnik/Informationstechnik an der Otto-von-Guericke-Universität Magdeburg sollte ein Roboter entwickelt werden, der die Funktionsweise eines Parkassistenten simuliert. In nachfolgender Arbeit wird veranschaulicht, wie sich dies mithilfe von Lego Mindstorms realisieren ließ. Dabei wird auch grafisch auf die Umsetzung des Roboters eingegangen. Neben Konstruktion und Programmierung erfolgt zudem eine Analyse der dabei aufgetretenen Schwierigkeiten und Probleme. Im Zuge des Projekts ist dabei ein Fahrzeug entstanden, das seine Parklücke selbstständig erkennt und sowohl ein- als auch ausparken kann. Hierbei wird zwischen Längs- und Querparken, sowie zwischen verschiedenen Größen der Parklücke differenziert.**

***Schlagwörter*—Autonomes Fahren, Assistenzsysteme, Parken, Parklückenerkennung, LEGO® MINDSTORMS®, MATLAB**

I. Motivation

DIE Mobilität der Zukunft hat längst begonnen. Auch wenn das vollautonome Fahren gesetzlich noch nicht erlaubt ist, werden moderne Automobile immer eigenständiger. Doch bis sich der Mensch seiner Aufgaben als Fahrzeugführer voll und ganz entledigen und zum reinen Passagier werden kann, sollen unzählige Fahrerassistenzsysteme für Entlastung und Unterstützung sorgen. So verringert beispielsweise der Notbremsassistent, falls ein Hindernis naht, die Geschwindigkeit des Fahrzeuges bis hin zum Stillstand. Der Fernlichtassistent blendet je nach Lichtverhältnissen selbstständig auf oder ab. Neben der Sicherheit im Straßenverkehr bietet die Entwicklung des autonomen Fahrens auch noch einige andere Vorteile. Zum einen lassen sich während der Fahrt wichtige Dinge erledigen, da die Steuerung des Autos nicht mehr übernommen werden muss. Zum anderen können auch Menschen mobil sein, die aufgrund ihres körperlichen oder psychischen Gesundheitszustandes nicht in der Lage sind, ein Fahrzeug eigenständig zu führen.

Aus genannten Gründen war es ersichtlich, sich des Themas Mobilität im Rahmen des zweiwöchigen Projekts anzunehmen und einen Teil des autonomen Fahrens, das autonome Parken, näher zu betrachten. Die Idee war es dementsprechend, ein Fahrzeug zu konstruieren, das eigenständig eine geeignete Parklücke erkennt und nach dessen Begebenheiten autonom ein- und ausparkt. Zur Verfügung standen dabei die Komponenten von Lego Mindstorms mit Motoren und Sensoren, wie z. B. Licht-, Tast- oder Ultraschall-Abstandssensoren. Außerdem der „NXT-Brick“, der die Schnittstelle der Komponenten bildet und mithilfe der Software MATLAB programmiert werden konnte. Eine eigens von der RWTH Aachen entwickelte Toolbox mit NXT-Befehlen diente zur Ansteuerung der Aktorik und Sensorik.

DOI: 10.24352/UB.OVGU-2020-028

II. Vorbetrachtungen

A. Stand der Technik

Parkassistenzsysteme gibt es schon seit mehreren Jahren. So stellte Volkswagen mit dem „Park Assist“ im Jahre 2006 den weltweit ersten Parklenkassistenten mit integrierter Parklückensuche vor. Nach weiteren Entwicklungen beherrscht dieser, neben dem ursprünglichen Längsparken, heute auch das Quer- und Vorwärtsparken [1]. Benötigt werden dafür, zusätzlich zu zwei seitlichen Sensoren, jeweils vier Ultraschallsensoren an Front und Heck des Fahrzeuges [2]. Beschleunigen, bremsen und zwischen den Zügen schalten, muss der Fahrer hingegen noch selbst übernehmen. Ein Modell, in dem das Auto sogar komplett ohne Fahrer auskommt, wurde in Zusammenarbeit von Bosch und Daimler entwickelt [3]. Hierbei fährt das Fahrzeug autonom durch ein Parkhaus und sucht unterdessen einen Parkplatz, während sich der Fahrer schon mit anderen Dingen beschäftigen kann. In gleicher Weise wird das Auto auch wieder ausgeparkt. Den Ein- oder Ausparkvorgang startet der Fahrer per Smartphone-App.

B. Eigene Lösung

Da der NXT nur Anschlussmöglichkeiten für vier Sensoren bietet, konnten lediglich drei Ultraschallsensoren für Abstandsmessungen und ein Farbsensor zur Lenkzeitpunkterkennung zum Einsatz kommen. Die Punkte, die dabei im Vorfeld geklärt werden mussten, waren zum einen die Ermittlung der Größe der Parklücke und zum anderen die unterschiedlichen Parkabläufe bei unterschiedlich großen Parklücken.

Zur Parklückenerkennung bot es sich an, aus der Geschwindigkeit des Fahrzeuges und der zum Passieren der Lücke benötigten Zeit, die Länge der Lücke zu bestimmen. An einen Motor kann allerdings nur die sogenannte Power p übergeben werden. Diese entspricht der am Motor anliegenden Batteriespannung und wird in Prozent angegeben. Um den Zusammenhang zwischen Power und der daraus resultierenden Geschwindigkeit v (in $\frac{\mathrm{cm}}{\mathrm{s}}$) herzustellen, führte man einen Versuch durch. Dazu wurde eine Strecke von einem Meter mit schwarzen Linien abgeklebt und ein Programm zur Zeitmessung zwischen den Linien geschrieben. Die Ergebnisse dieses Versuchs veranschaulicht Tabelle I. Anhand des ermittelten Faktors c und der Gleichung (1) kann folglich aus jeder beliebigen Power (in %) eine Geschwindigkeit (in $\frac{\mathrm{cm}}{\mathrm{s}}$) berechnet werden.

$$v = p \cdot c \tag{1}$$

$p =$ Power (in %)
$c =$ konstanter Faktor (in $\frac{\mathrm{cm}}{\mathrm{s} \cdot \%}$)

Tabelle I
MESSDATEN ZUR GESCHWINDIGKEIT

Power (in %)	Zeit (in s)	Geschwindigkeit (in $\frac{\text{cm}}{\text{s}}$)	Faktor (in $\frac{\text{cm}}{\text{s}\cdot\%}$)
10	21,190	4,71	0,471
20	10,616	9,41	0,470
30	7,066	14,15	0,471
40	5,271	18,97	0,474
60	3,461	28,89	0,481
80	2,642	37,85	0,473
100	2,591	38,59	0,385

Eine weitere Grundlage war die Berücksichtigung unterschiedlicher Längen der Parklücke. In einer großen Lücke kann nicht mit dem gleichem Ablauf eingeparkt werden wie in einer kleinen. In einer sehr engen Parkbox muss außerdem noch ein Zug zum Korrigieren vorgenommen werden. Demnach müssen beim Längsparken drei Größen in Betracht gezogen werden, die nach Konstruktion und Erprobung des Fahrzeuges in das Programm aufgenommen werden konnten. In Folge dieser Betrachtungen lassen sich nun die Algorithmen mithilfe von MATLAB formulieren.

III. REALISIERUNG

Um das autonom parkende Fahrzeug realisieren zu können, müssen zwei Komponenten in Betracht gezogen werden. Zum einen die konstruktive Seite, welche die effektive Anordnung von Aktorik und Sensorik umfasst und dem Vehikel Stabilität verleiht, und zum anderen die programmiertechnische Seite, die das Fahrzeug letztendlich in Bewegung versetzt und steuert. In folgenden Abschnitten wird deshalb genauer auf diese beiden Bestandteile eingegangen.

A. Konstruktion

Bei der Konstruktion des Roboters stehen neben Stabilität und Funktionalität vor allem auch die realitätsnahe Gestaltung des Vehikels im Fokus. Die Wahl fällt daher auf ein Fahrzeug mit vier Rädern, zwei Motoren für die Aktorik, sowie drei Ultraschallsensoren und einem Farbsensor für die Sensorik. Alle Komponenten des Fahrzeuges sind an den Lego-NXT-Brick angeschlossen, der die Steuereinheit des Roboters bildet und gut zugänglich oben auf dem Fahrzeug angebracht ist.

Einer der Motoren fungiert zur Fortbewegung und ist über ein Differentialgetriebe mit den Hinterrädern verbunden. Dieses dient dem Ausgleich der Drehzahlen der beiden angetriebenen Räder in Kurvenfahrten. Dabei dreht sich das kurveninnere Rad, aufgrund seines kleineren Radius zum Kurvenmittelpunkt, langsamer als das äußere Rad. Somit entsteht ein verbessertes Fahrverhalten in engen Kurven, wie es z. B. bei Parkvorgängen der Fall ist. Ein weiterer Motor befindet sich im vorderen Teil des Fahrgestells und betreibt die Lenkung der Vorderräder. Diese ist realitätsgetreu nach dem Ackermann-Lenkprinzip [4] konstruiert. Hierbei sind die Räder nicht über eine Achse miteinander verbunden, sondern besitzen jeweils eine eigene Schwenkachse. Bei Kurvenfahrten wird so das kurveninnere Rad weiter eingeschlagen als das kurvenäußere. Besonders bei großen Lenkwinkeln, wie es auch beim Parken der Fall ist, wird so eine gute Fahrdynamik aufrecht erhalten. Die großen Lenkwinkel werden durch eine optimierte Konstruktion erreicht, die es ermöglicht, von ursprünglich $100°$ auf $160°$ Drehwinkel am Motor zu gelangen.

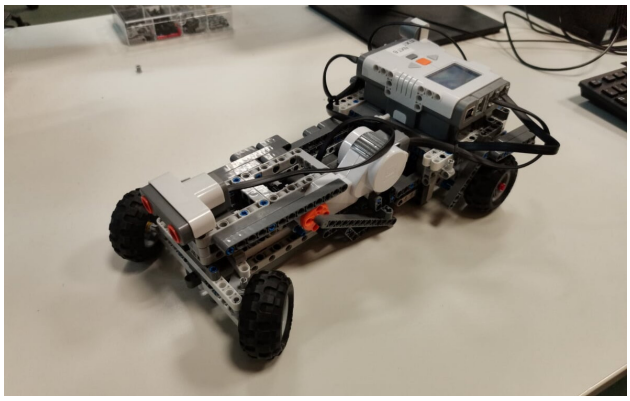


Abbildung 1. Konstruktion des Fahrzeuges

Bei der Anordnung der Sensorik war es das Ziel, alle Komponenten so kompakt wie möglich anzuordnen, die Funktionalität dabei jedoch nicht zu beeinträchtigen. Um die Konstruktion realitätsnah zu gestalten und auch das Einparken in kleinen Parklücken zu ermöglichen, sollte keiner der Sensoren über das Vehikel hinausragen. Aus diesem Grund endet das Fahrgestell, wie in Abbildung 1 dargestellt, genau mit dem vorderen Ultraschallsensor. Dieses Prinzip wird auch auf den hinteren und seitlichen Ultraschallsensor angewandt. Beim seitlichen Sensor wurde zudem noch darauf geachtet, die Befestigung am hinteren Ende vorzunehmen, damit das Heck des Fahrzeugs nach Erkennung einer Parklücke mit dem Ende der Lücke abschließt. Um die Farbmarkierungen zum richtigen Zeitpunkt erkennen zu können, musste auch der Farbsensor im hinteren Teil untergebracht werden. Dieser ist daher an der Innenseite des rechten Hinterrads positioniert.

B. Programmierung

Um das Programm zu beginnen, muss der Nutzer über die grafische Benutzeroberfläche (siehe Abschnitt: III-C) vorab den Wert der Power eingeben und den gewünschten Vorgang starten. Wie bereits in den Vorbetrachtungen erwähnt, wird diese mithilfe der Gleichung (1) in eine Geschwindigkeit (in $\frac{\text{cm}}{\text{s}}$) umgewandelt, damit später die Größe der Parklücke bestimmt werden kann. Die weiteren Programmierungen lassen sich in drei Vorgänge unterteilen. Soll das Fahrzeug einparken, muss es zunächst eine Parklücke finden. Danach folgt der eigentliche Parkvorgang, während zum Schluss noch ausgeparkt werden kann. Alle Abläufe werden im Folgenden näher erläutert.

1) Parklückenerkennung: Während sich das Fahrzeug vorwärts bewegt, misst der seitlich angebrachte Ultraschallsensor dauerhaft den Abstand zur Wand, welche die bereits parkenden Autos darstellt. Tritt hierbei eine große Differenz der Abstandswerte nach oben auf, so wird dieser Zeitpunkt gespeichert. Werden die Abstände sprunghaft kleiner, wird erneut ein Zeitpunkt im Speicher hinterlegt. Aus der Differenz beider Zeitpunkte wird nun die zum Passieren der Lücke

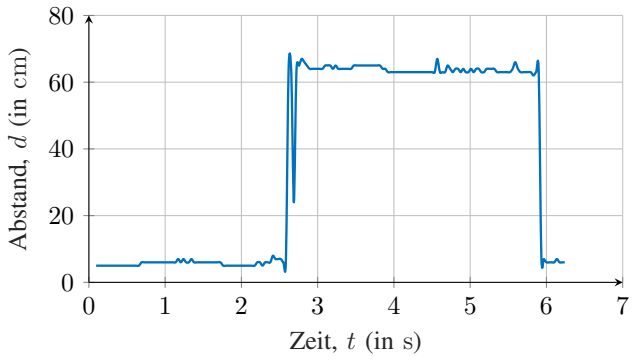


Abbildung 2. Zeitverlauf des Abstandes des Fahrzeuges zur Wand, gemessen durch den seitlichen Ultraschallsensor

benötigte Zeit bestimmt. Erkennbar ist dieser Zusammenhang in Abbildung 2. Die Zeitpunkte des sprunghaften Verhaltens des Graphen signalisieren jeweils den Anfang und das Ende der Lücke.

Aus Geschwindigkeit und Zeit wird folglich die Länge der Lücke ermittelt. Sollte diese größer sein als die Eigenlänge bzw. Eigenbreite des Fahrzeugs (zzgl. einer Toleranz von $10\,\mathrm{cm}$), erkennt das Fahrzeug diese Lücke als Parklücke an und stoppt zunächst seine Bewegung nach vorn.

2) Einparken: Unmittelbar nach der Parklückenerkennung wird mit dem gewählten Einparkvorgang (längs bzw. quer) begonnen. Der Algorithmus des Querparkens ist in Abbildung 3 dargestellt. Im Folgenden wird nur der Ablauf des Längsparkens erläutert.

Hier entscheidet das Fahrzeug zwischen drei unterschiedlichen Parklücken. Alle Einparkvorgänge beginnen mit dem Rückwärtsfahren auf vollem Lenkeinschlag nach rechts. Sobald der Farbsensor die rote Markierung erkennt, erfolgt der Volleinschlag der Lenkung nach links. Für die beiden kleinen Parklücken wird der hintere Ultraschallsensor verwendet, um soweit wie möglich nach hinten zu fahren und zu stoppen, bevor die hintere Wand erreicht wird. Sollte die Parklücke sehr eng sein, wird noch ein Zug zum Korrigieren benötigt. Dabei wird erneut rechts eingeschlagen und vorwärts gefahren. Damit das Fahrzeug zum Schluss ordnungsgemäß in der Lücke steht, werden die beiden Ultraschallsensoren an Front und Heck verwendet. Diese werten die Abstände nach vorn und hinten aus und es wird gegebenenfalls vor- bzw. zurückgefahren, bis die beiden Abstände übereinstimmen. Bei großen Parklücken wird der vordere Ultraschallsensor genutzt, um die Bewegung zu stoppen, sobald das Fahrzeug gerade in der Lücke steht. Letzten Endes wird so weit wie möglich vorgefahren und der Einparkvorgang ist beendet.

3) Ausparken: Das Ausparken erfolgt auf Basis der Einparkvorgänge, deren Abläufe umgekehrt und teilweise spezifiziert sind. Auch hier werden die Farbmarkierungen genutzt, um den genauen Zeitpunkt der Lenkeinschläge zu bestimmen. Das Fahrzeug wurde so programmiert, dass es seine momentane Parksituation selbstständig erkennt und so den korrekten Ausparkvorgang autonom einleitet. Die Unterscheidung der Fälle erfolgt durch Auswertung von Abständen, die mittels Ultraschallsensoren an Front und Heck gemessen werden.

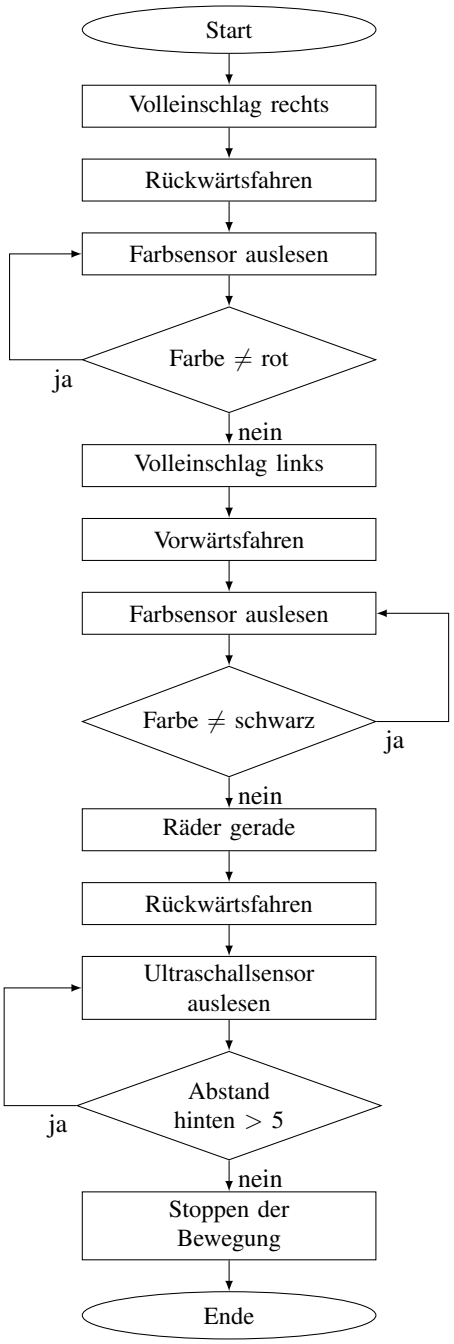


Abbildung 3. Programmablaufplan des Einparkalgorithmus zum Quereinparken

C. Grafische Benutzeroberfläche

Zur Interaktion des Fahrzeuges mit seiner Umwelt wurde eine grafische Benutzeroberfläche (GUI, englisch: graphical user interface) entwickelt, die es dem Anwender ermöglicht, durch einfache Bedienung mit dem Fahrzeug zu kommunizieren. Durch die Anordnung zweier Knöpfe auf der Bedienoberfläche kann der Nutzer entscheiden, ob er das Fahrzeug ein- oder ausparken möchte. Soll das Fahrzeug eine Parklücke suchen und einparken, ist es mittels eines PopUp-Menüs möglich, zwischen einem der beiden Einparkvorgänge auszuwählen. Für den Ausparkvorgang muss allerdings keine Auswahl im PopUp-

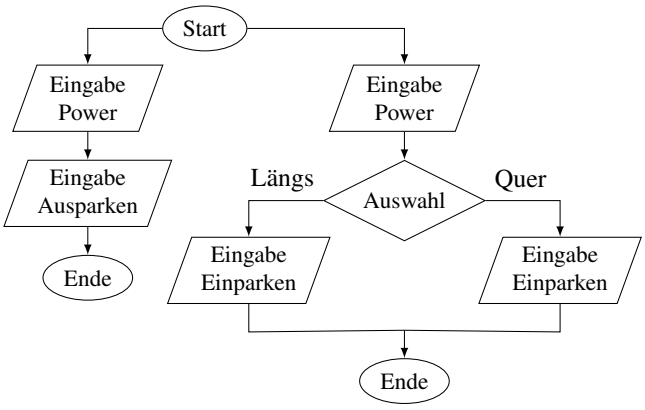


Abbildung 4. Programmablaufplan der grafischen Benutzeroberfläche

Menü getroffen werden, da das Fahrzeug selbstständig erkennt in welcher Parklücke es sich augenblicklich befindet. Durch einen Schieberegler kann zudem noch die Geschwindigkeit eingestellt werden, mit der das Auto eine Parklücke sucht. Hierbei ist darauf zu achten, dass sowohl das eigentliche Ein- als auch Ausparken mit halber Geschwindigkeit ablaufen. Zum Schluss muss noch der entsprechende Knopf zum Ein- oder Ausparken betätigt werden und das Fahrzeug beginnt mit dem gewählten Ablauf. Der Programmablauf der Benutzeroberfläche ist in Abbildung 4 dargestellt.

IV. Ergebnisdiskussion

Das Ergebnis des Projekts ist ein Roboter, der eigenständig Parklücken erkennt und autonom ein- sowie ausparkt. Die Ermittlung der Größe der Parklücke funktioniert mit einer geringen Abweichung von $1\,\mathrm{cm}$ bis $2\,\mathrm{cm}$ sehr genau und wurde daher ohne weitere Verbesserungen umgesetzt. Die Parkvorgänge stellen allerdings nur eine Simulation des eingangs erwähnten Parkassistenten dar. Die Gründe dafür werden im Folgenden aufgezeigt.

Aufgrund der teilweise instabilen Legoteile erwies sich die Befestigung der Räder als sehr schwierig und die Lenkung bekam somit zu viel Bewegungsfreiraum. Des Weiteren zeigten sich die Ultraschallsensoren in der Abstandsmessung ziemlich ungenau und ermittelten keine Abstände unter $5\,\mathrm{cm}$. Diese beiden Faktoren führten dazu, dass ein Algorithmus, der den Roboter auf einem gleichmäßigen Abstand zur Wand hält, nicht umsetzbar war. Ein gerades Vorwärtsfahren des Fahrzeuges konnte daher nicht realisiert werden. Der Roboter beginnt jedes erneute Parken mit einer unterschiedlichen, seitlichen Entfernung zur Wand, was dazu führt, dass er manchmal schief in der Parklücke steht oder beim Parken sogar an der Wand anstößt. Außerdem besteht das Problem, dass die Parkvorgänge immer noch an einen speziellen Modellaufbau gebunden sind. Es werden Farbmarkierungen benötigt, die den richtigen Zeitpunkt zum Lenkeinschlag signalisieren und mit festen Abständen am Boden befestigt sind. Dementsprechend ist das Ergebnis eine unter den Möglichkeiten von Lego Mindstorms sinnvolle Lösung, die aber nicht der Realität entspricht.

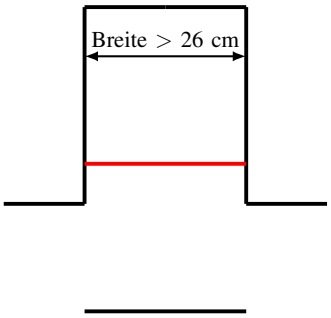


Abbildung 5. Darstellung einer Parklücke zum Quereinparken inkl. der benötigten Farbmarkierungen

V. Zusammenfassung und Fazit

Das anfangs erwähnte Ziel, einen Roboter zu konstruieren, der eine geeignete Parklücke erkennt und autonom ein- sowie ausparkt, wurde vollständig in die Realität umgesetzt. Dabei wurden neben zwei Motoren für Lenkung und Antrieb vier Sensoren genutzt, die die Umgebung des Fahrzeuges scannen. Drei Ultraschallsensoren sind für Abstandsmessungen verantwortlich, wobei ein seitlich angebrachter Sensor zur Vermessung der Parklücke dient. Zudem wird ein Farbsensor verwendet, um den exakten Zeitpunkt der Lenkbewegungen zu erfassen.

Zur zukünftigen Entwicklung gehören die Verbesserung der genannten Probleme und weitere Optimierungen. So ließe sich zusätzlich noch das Vorwärtsparken, sowohl parallel als auch quer zur Fahrbahn, realisieren. Weiterhin muss speziell beim Ausparken auf sich nahende Hindernisse, wie z. B. den fließenden Verkehr, geachtet werden. Das Fahrzeug wäre dementsprechend so zu programmieren, dass seine momentane Bewegung unterbrochen wird, wenn Hindernisse erkennbar sind.

Anhang

Die Abbildung 5 zeigt die Gestaltung einer Parklücke zum Quereinparken inklusive der benötigten Farbmarkierungen zur Lenkzeitpunkterkennung. Des Weiteren wurden im Rahmen der Abschlusspräsentationen des Projektseminars Kurzfilme der diesjährigen Projekte erstellt. Die Playlist [5] enthält mehrere Videos, die den Ablauf der Parkvorgänge darstellen.

Literaturverzeichnis

[1] *Einparken leicht gemacht: ein Jahrzehnt „Park Assist.* https://www.volkswagen-newsroom.com/de/pressemitteilungen/einparken-leicht-gemacht-ein-jahrzehnt-park-assist-1574. Version: 5/3/2020

[2] Volkswagen Service Training VSQ-1: SSP 389 Der Parklenkassistent. (2007). https://www.motor-talk.de/forum/aktion/Attachment.html?attachmentId=677825

[3] *Fahrerlos geparkt. Automated Valet Parking*. https://www.daimler.com/innovation/case/autonomous/fahrerlos-geparkt.html. Version: 5/3/2020

[4] *VolksBot*. https://www.volksbot.de/ackermann-de.php. Version: 1/5/2020

[5] Magdowski, Mathias: *Lego-Praktikum 2020*. https://www.youtube.com/playlist?list=PLWCaO6Bpqy-5xCwNJblRE4hnuXa2PTu0M. Version: 5/3/2020

Autonomes Einparken im Modellversuch

Vincent Siermann, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Während des Projektseminars „Elektrotechnik/Informationstechnik“ der Otto-von-Guericke-Universität Magdeburg wurde ein Modellversuch zum Parklenkassistenten durchgeführt. Unter Benutzung von „Lego-Mindstorms“ konnten die technischen Abläufe eines autonom ein- und ausparkenden Fahrzeuges demonstriert und erweitert werden. Über die Vermessung einer Parklücke ist das entwickelte System in der Lage, eine passende Lücke zu erkennen. Die über einen Algorithmus ausgeführte Messung wird im Anschluss mit der Fahrzeuggröße verglichen. Zur Umsetzung der Lenkbefehle des Ein- und Ausparkvorganges dienen definierte Fahrbahnmarkierungen.**

***Schlagwörter*—Assistenzsystem, Ausparken, Autonom, Einparken, Lego-Mindstorms, Parklückenerkennung**

I. Einleitung

EIN Fahrzeug besitzt heutzutage immer häufiger Assistenzsysteme. Die *Park Distance Control* (PDC) gehört heute nahezu zur Serienausstattung eines PKWs. Selbst Systeme wie Spurhalteassisent, Spurwechselassistent oder Intelligente Tempomaten gehören zum „Must-have“ bei einem Neufahrzeug. Diese sollen dem Fahrer assistieren und in bestimmten Situationen das Führen des Fahrzeuges erleichtern. Das autonome Fahren ist heutzutage in der Forschung und Entwicklung an oberste Stelle geraten und kann eine enorme Unterstützung für die Fahrzeugführer darstellen. Autonom fahrende Fahrzeuge sollten nicht nur selbstständig fahren, sondern auch in verschiedenen Alltagssituationen autonom Ein- und Ausparken können. Der Parklenkassistent erleichtert schon heute das Einparken, sodass der Fahrer nur die Geschwindigkeit des Fahrzeuges regulieren muss.
Innerhalb des Projektseminars „Elektrotechnik/Informationstechnik“ der Otto-von-Guericke-Universität Magdeburg war es das Ziel, einen autonom ein- und aus parkenden Roboter zu entwickeln. Die Konstruktion sollte einem PKW nachempfunden werden. Für die Konstruktion wurde ein „Lego-Mindstorms“ Bausatz verwendet. Den Kern bildet hier der „NXT-Baustein“. Dort lassen sich bis zu 4 Sensoren und 3 Motoren anschließen. Durch die „RWTH - Mindstorms NXT Toolbox“ der RWTH Aachen [1] ist es möglich, eine Kommunikation zwischen NXT und MATLAB herzustellen. Der Roboter sollte selbständig erkennen, ob die Größe einer Parklücke für das Fahrzeug geeignet ist. Je nach Größe der Parklücke müssen verschiedene Szenarien berücksichtigt werden. Während des Parkvorgangs sollte der Roboter mit einer konstanten Geschwindigkeit einparken und zusätzlich beim Ausparken selbständig erkennen, ob das Fahrzeug zuvor quer oder längs eingeparkt wurde.

DOI: 10.24352/UB.OVGU-2020-029

II. Vorbetrachtungen

A. Stand der Technik

Der Parklenkassistent ist nun seit mehr als 10 Jahren in Kraftfahrzeugen erhältlich. Er basiert auf der *Park Distance Control* (PDC), dieser misst die Distanz zu vorderen oder hinteren Gegenständen. Zusätzlich werden dem Fahrer durch akustische und/oder optische Signale Informationen zur Distanz zum erkannten Hindernis übermittelt. Bei dem PDC-System muss der Fahrer noch selbstständig die Lenkvorgänge durchführen. Der Parklenkassistent hingegen übernimmt die Lenkvorgänge und der Fahrzeugführer muss nur noch die Geschwindigkeit beim Einparken regulieren. Der Parklenkassistent der „Volkswagen AG“ muss vor dem Einparkvorgang aktiviert werden. Sobald das System aktiv ist, wird dem Fahrzeugführer empfohlen, mit einer Geschwindigkeit von unter $30\,\frac{\mathrm{km}}{\mathrm{h}}$ an Parklücken vorbeizufahren. Währenddessen sucht das System eine passende Parklücke. Sollte die Geschwindigkeit von $30\,\frac{\mathrm{km}}{\mathrm{h}}$ überschritten werden, schaltet sich das System ab. Erkennt das System eine Lücke, wird mit dem Einparkvorgang begonnen. Dazu wird dem Fahrer signalisiert anzuhalten und den Rückwärtsgang einzulegen. Während des Einparkvorganges darf eine Geschwindigkeit von $7\,\frac{\mathrm{km}}{\mathrm{h}}$ nicht überschritten werden [2].

B. Geschwindigkeitsmessung

Tabelle I
Messdaten zur Geschwindigkeit

Power (in %)	Zeit (in s)	Geschwindigkeit (in $\frac{\mathrm{cm}}{\mathrm{s}}$)	Faktor (in $\frac{\mathrm{cm}}{\mathrm{s}\cdot\%}$)
10	21,190	4,71	0,471
20	10,616	9,41	0,470
30	7,066	14,15	0,471
40	5,271	18,97	0,474
60	3,461	28,89	0,481
80	2,642	37,85	0,473
100	2,591	38,59	0,385

Bei den „Lego-Mindstorms“-Motoren kann die Leistung nur in % der Batteriespannung angegeben werden, was mit „Power“ bezeichnet wird. Um die Geschwindigkeit des Fahrzeuges zu ermitteln, wurde ein Versuch durchgeführt. Dazu erfolgte eine Messung der Zeit, die das Fahrzeug benötigt, um einen Meter ($100\,\mathrm{cm}$) Strecke zurückzulegen. Anhand dieser Werte konnte die Geschwindigkeit (in $\frac{\mathrm{cm}}{\mathrm{s}}$) ermittelt werden. Indem die berechneten Geschwindigkeiten durch die eingegebene Power (in %) dividiert werden, ergab sich ein Faktor (in $\frac{\mathrm{cm}}{\mathrm{s}\cdot\%}$) mit dem man jede Powereingabe in eine Geschwindigkeit umrechnen kann. Aus der Tabelle I können die für dieses Fahrzeug ermittelten Faktoren entnommen werden.

C. Lenkgeometrie

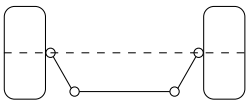


Abbildung 1. Lenktrapez

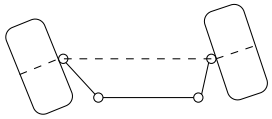


Abbildung 2. Lenktrapez bei nach links eingeschlagener Lenkung

Um ein problemloses Abrollen der Räder bei einer Kurvenfahrt zu ermöglichen, spielt die Lenkgeometrie eine große Rolle. Da der Aufbau in diesem Modellversuch nah an einem realen Fahrzeug liegen soll, wurde die Lenkung nach dem Lenktrapez (Abbildung 1) aufgebaut. Das kurveninnere Rad muss beim Fahren einer Kurve einen kleineren Radius zurücklegen als das kurvenäußere Rad. Hierzu muss die Lenkung einen unterschiedlichen Einschlagwinkel der Räder ermöglichen. Das kurveninnere Rad muss demzufolge mehr einschlagen als das kurvenäußere Rad (Abbildung 2) [3].

III. Aufbau und Funktionsabläufe

A. Aufbau des Fahrzeuges

Abbildung 3. Farbsensor

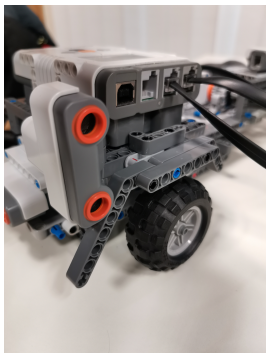


Abbildung 4. Seitlicher Ultraschallsensor

1) Sensorik: Durch drei Ultraschallsensoren und einen Farbsensor wird die Abfolge des Ein- und Ausparkens umgesetzt. Die Ultraschallsensoren werden zur Abstandsmessung benutzt, wobei der erste an der Front angebracht ist. An dieser Position dient er der Überwachung des Abstandes zu Objekten, welche sich vor dem Roboter befinden und löst ggf. ein Stoppen des Bewegungsablaufes aus. Der am Heck angebrachte zweite Ultraschallsensor ist dem bereits beschriebenen Sensor in seiner Funktion äquivalent, außer, dass er den Abstand zu Objekten hinter dem Roboter ermittelt. Ein weiterer, dritter Ultraschallsensor misst den Abstand zur rechten Seite, wodurch die Größe der Parklücke ermittelt wird. Um den Zeitpunkt des Umlenkens während des Einparkvorgangs zu finden, wird ein Farbsensor benutzt.
Die genaueren Funktionen der Sensoren werden in den nachfolgenden Abschnitten näher erläutert.

2) Aktorik: Um das Fahren und Lenken zu realisieren, wurden zwei Motoren integriert. Motor A treibt über ein Differenzial die Hinterachse an, um vor- und zurück zu fahren. Motor B ist für die Lenkung während des Ein- und Ausparkens zuständig. Die „Lego-Mindstorms“-Motoren lassen sich so ansteuern, dass sie nur eine bestimmte Anzahl an Umdrehungen durchführen. Dies wird über einen Befehl „TachoLimit“ festgelegt. Hierbei wird ein Winkel in Grad eingegeben um den sich der Motor drehen soll. Ein Winkel von $360°$ entspricht einer vollen Umdrehung. Bei der hier konstruierten Lenkung ist es möglich, $140°$ nach links oder rechts einzuschlagen. In diesem Fall und allen folgenden Erläuterungen wird vom Drehwinkel des Motors ausgegangen und nicht vom eigentlichen Lenkeinschlag des Roboters. Außerdem ist sie einem realen Fahrzeug nachempfunden und nach dem Lenktrapez (siehe Abbildung 1) aufgebaut.

B. Parklückenerkennung

Der Roboter sollte während des Vorbeifahrens an parkenden Gegenständen selbstständig eine Lücke erkennen und ggf. einparken. In diesem Abschnitt wird genauer auf den Ablauf zur Erkennung einer Lücke eingegangen.
Realisiert wird die Abstandsmessung über einen seitlich angebrachten Ultraschallsensor (Abbildung 4), welcher kontinuierlich beim Vorbeifahren den Abstand misst. Die Messung wird in Abbildung 5 dargestellt.

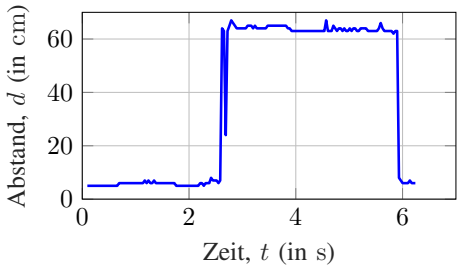


Abbildung 5. Abstandsmessung durch den seitlichen Ultraschallsensor beim Passieren einer Lücke.

Dort ist veranschaulicht wie der Sensor beim Passieren eines Gegenstandes kleine Abstandswerte misst und die Abstandswerte ansteigen, sobald die Lücke beginnt. An dem Punkt wo der Ultraschallsensor einen deutlich größeren Abstandswert misst, wird der erste *clock*-Befehl gesetzt (ermöglicht es, das Datum und die exakte Uhrzeit unter MATLAB auf ms genau abzuspeichern). Die genaue Größe des Abstandes ist für diesen Versuch nebensächlich, wichtig dabei ist ein klarer Unterschied zwischen den Abstandswerten. Sobald also der Anfang einer Lücke erkannt wird, fährt das Fahrzeug weiter und der Ultraschallsensor misst einen großen Abstand (Abbildung 5 Bereich von $2{,}3\,\mathrm{s} < t\ (\mathrm{in\ s}) < 5{,}9\,\mathrm{s}$). Wird nun vom Ultraschallsensor wieder einen Abstand der kleiner als $10\,\mathrm{cm}$ gemessen, startet ein zweiter *clock*-Befehl. Aus der Differenz der Anfangs- und Endzeit kann nun die Zeit ermittelt werden, die von Anfang bis zum Ende der Lücke verstrichen ist. Das geschieht über den in MATLAB definierten *etime*-Befehl. Aus dieser Zeit und mit der Geschwindigkeit des Fahrzeuges, kann nun die

Größe der Parklücke ermittelt werden, da die Geschwindigkeit konstant gehalten wird. Dies erfolgt über das physikalische Gesetz der gleichförmigen Bewegung $v = \frac{s}{t}$. Grundsätzlich wird die Größe der Parklücke beim Längseinparken mit der Fahrzeuglänge verglichen und beim Quereinparken mit der Fahrzeugbreite. Wie die Geschwindigkeit dieses Roboters mit „Lego-Mindstorm“-Motor ermittelt wurde, zeigt Punkt B der Vorbetrachtung (*II-B*).

C. Einparkvorgang

Der Einparkvorgang erfolgt direkt nachdem eine Parklücke identifiziert wurde. Dieser beginnt mit einem definierten Rechtseinschlag (hier ein Winkel von $130°$ zum Quer- und $120°$ zum Längseinparken). Ist dies abgeschlossen wird mit der Rückwärtsfahrt begonnen. Hier kommt der Farbsensor zum Einsatz (Abbildung 3), welcher am Heck des Fahrzeuges angebracht wurde.

1) Quereinparken: Der weitere Ablauf wird am Beispiel des Querparkens näher erläutert. Der Programmablaufplan des Quereinparkens wird in Abbildung 6 gezeigt. Der Farbsensor ermittelt beim Rückwärtsfahren kontinuierlich die Farbe des Untergrundes. Erkennt er eine Markierung (hier: rote Linie), wird die Rückwartsfahrt beendet und der Motor B schlägt die Lenkung auf vollen Linkseinschlag (entspricht dem nach rechts eingeschlagenen Winkel vom $130°$ + einen Winkel von $150°$). Nun beginnt das Fahrzeug nach vorn zu fahren und der Farbsensor ermittelt erneut die Farbe des Untergrundes. Hat dieser nun eine weitere Markierung (hier: schwarze Linie) erkannt, wird die Bewegung nach vorn gestoppt und die Lenkung gerade gestellt. Der Farbsensor wird ab diesem Zeitpunkt nicht mehr benötigt und geschlossen. Das Fahrzeug fährt nach der geraden Ausrichtung der Lenkung rückwärts, nun wird über den am Heck angebrachten Ultraschallsensor die Distanz nach hinten ermittelt. Sollte die Distanz den Wert von $5\,\mathrm{cm}$ erreichen, wird die Rückwärts-Fahrt beendet und der Parkvorgang damit abgeschlossen.

2) Längseinparken: Die Abfolge des Parkens bis zur Erkennung der roten Linie entspricht der Abfolge des Quereinparkens. Allerdings wird ein anderer Lenkeinschlag verwendet. Nach dem Erkennen der roten Linie wird die Lenkung voll nach links eingeschlagen (entspricht dem nach rechts eingeschlagenen Winkel von $120°$ + einen Winkel von $160°$) aber anders als beim Quereinparken weiter zurück gefahren. Des Weiteren müssen beim Längseinparken drei Fälle unterschieden werden, welche nach der Größe der Parklücke definiert sind.

Fall 1: Die Parklücke ist kleiner als $50\,\mathrm{cm}$. Diese Lücke ist für den entwickelten Roboter so klein, sodass in der Lücke die Position korrigiert werden muss um gerade zu stehen. Beim zurück fahren mit vollem Linkseinschlag wird der hintere Ultraschallsensor ausgelesen und solange gefahren bis dieser einen Wert von $5\,\mathrm{cm}$ misst. Anschließend steht der Roboter noch nicht gerade ausgerichtet in der Parklücke. Die Lenkung wird daher voll nach rechts eingeschlagen (entspricht dem nach links einschlagenen Winkel von $140°$ + einen Winkel von$140°$) und der Roboter fährt nach vorn bis der vordere Ultraschallsensor einen Wert von $5\,\mathrm{cm}$ misst. Die Räder werden gerade ausgerichtet und ein Ausgleichsvorgang beginnt. Der

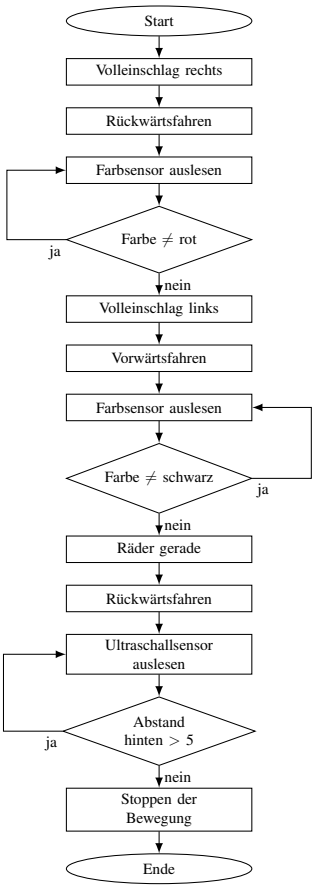


Abbildung 6. Programmablaufplan zum Algorithmus des Quereinparkens.

Ausgleichsvorgang sieht vor, dass der Roboter zum vorderen und hinteren Objekt den selben Abstand aufweist. Dabei messen die Ultraschallsensoren vorn und hinten kontinuierlich den Abstand. Es wird so lange rückwärts oder vorwärts gefahren, bis beide Sensoren den gleichen Wert aufweisen.

Fall 2: Die Parklücke ist größer als $50\,\mathrm{cm}$, aber kleiner als $55\,\mathrm{cm}$. Das Einparken erfolgt wie im ersten Fall, allerdings kann auf das Korrigieren innerhalb der Lücke verzichten werden. Der Roboter steht nach dem Zurückfahren mit vollem Linkseinschlag gerade sobald der hintere Sensor einen Wert von $5\,\mathrm{cm}$ misst. Danach werden die Räder gerade ausgerichtet und der Ausgleichsvorgang beginnt.

Fall 3: Die Parklücke ist größer als $55\,\mathrm{cm}$. Das Einparken erfolgt auch hier ähnlich zu den ersten beiden Fällen. Allerdings muss beim Zurückfahren mit vollem Linkseinschlag der vordere Ultraschallsensor zum stoppen der Bewegung eingesetzt werden, da der hintere Sensor aufgrund der Größe der Parklücke, immer noch große Werte misst, obwohl das Fahrzeug schon gerade ausgerichtet ist. Dies wäre auch der Fall wenn die Parklücke nur nach vorn eine Begrenzung aufweist. Die Bewegung wird bei einem Wert des vorderen Sensors von $21\,\mathrm{cm}$ gestoppt. Danach startet nicht wie in den anderen Fällen ein Ausgleichsvorgang, sondern das Fahrzeug fährt an das vordere Objekt bis auf einen Abstand von $10\,\mathrm{cm}$ heran.

D. Ausparkvorgang

Grundsätzlich werden beim Ausparkvorgang die Abläufe der einzelnen Einparkvorgänge umgekehrt. Das Fahrzeug erkennt allerdings selbständig, ob es quer oder längs in einer Parklücke steht. Dies wird über die gemessenen Abstände der drei Ultraschallsensoren festgelegt. Für das Längsausparken ist es wichtig, die drei Fälle vom Einparken zu berücksichtigen. Die Messungen erfolgen hierbei nur über den hinteren Ultraschallsensor. Der Roboter steht durch den Ausgleichsvorgang mittig in der Lücke. Misst dieser Sensor nun einen Abstand von unter $10\,\mathrm{cm}$, wird die kleinsten Lücke identifiziert (*III-C2 Fall 1*). Wenn der Abstand zwischen $10\,\mathrm{cm}$ und $15\,\mathrm{cm}$ liegt, steht der Roboter in der Lücke wie im Fall 2 beschrieben (*III-C2 Fall 2*). Sollte nur der Abstand einen Wert von mehr als $15\,\mathrm{cm}$ betragen, ist das Objekt nach hinten weit entfernt oder nicht vorhanden, in dieser Weise im Fall 3 (*III-C2 Fall 3*) zu sehen.

E. Eingabe in die GUI

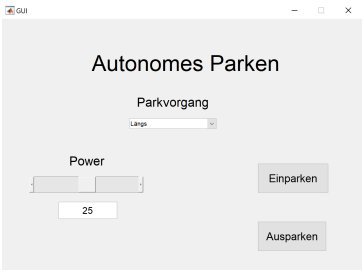


Abbildung 7. Eingabeoberfläche der GUI

In MATLAB gibt es die Möglichkeit eine grafische Benutzeroberfläche (*graphical user interfaces* auch „GUI“ genannt) zu entwerfen. „[Diese] ermöglichen die Point-and-Click-Steuerung von Softwareanwendungen“ [4] und können über das MATLAB Add-On „GUIDE“ entworfen werden.

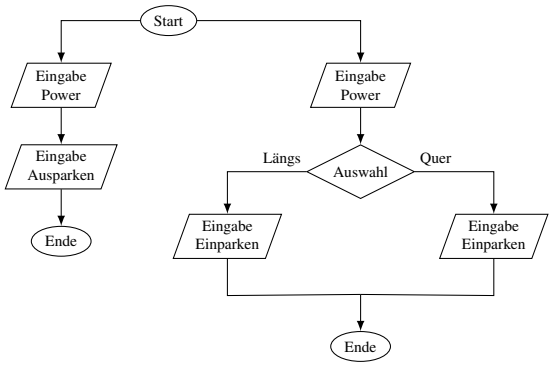


Abbildung 8. Programmablaufplan zur Eingabe in die graphische Benutzeroberfläche (GUI)

Hier wird eine GUI verwendet, um die Befehle zum Ein- und Ausparken nicht einzeln eingeben zu müssen. Wird die GUI (Abbildung 7) aufgerufen, kann mittels eines Pop-Up-Menüs ausgewählt werden ob der Einparkvorgang längs oder quer durchgeführt werden soll. Über einen Schieberegler kann die Power des Motors ausgewählt werden. Das Programm wird mit den Schaltflächen „Ein- und Ausparken“ gestartet. In der Abbildung 8 ist der Ablauf der Eingabe dargestellt. Zum Ausparken ist die Eingabe über das Pop-Up-Menü nicht erforderlich.

IV. Ergebnisdiskussion

Die Anfangs gesetzten Ziele wurden erfolgreich umgesetzt und es wurde ein Roboter entwickelt, welcher in der Lage ist, autonom ein- und auszuparken. Problematisch beim Aufbau ist, dass keine Spurführung vorgesehen ist. Das Fahrzeug muss vor jedem Vorgang exakt ausgerichtet werden, damit es gerade an Lücken vorbei fährt. Zusätzlich konnte bei den ersten Versuchen der erforderliche Lenkradius nich erreicht werden. Problematisch wurde dies beim Längseinparken, weshalb anfangs nur in Lücken größer $50\,\mathrm{cm}$ eingeparkt werden konnte. Dies wurde durch einen Umbau der ursprünglichen Lenkung korrigiert. Der resultierende erhöhte Lenkeinschlag ermöglichte es auch in Lücken unter $50\,\mathrm{cm}$ einzuparken. Ein weiterer Nachteil sind die Fahrbahnmarkierungen, welche als Hilfestellung zur Steuerung der Lenkung erforderlich sind, um einen erfolgreichen Parkvorgang zu gewährleisten.

V. Zusammenfassung und Fazit

Der entwickelte Roboter kann anhand von Messdaten durch Ultraschallsensoren einen Ein- und Ausparkvorgang steuern. Dabei erkennt er durch Vermessung die Größe einer Parklücke. Zusätzlich plant er selbstständig anhand dieser Daten wie der Vorgang des Parkens am effizientesten umzusetzten ist. Beim Ausparken erkennt er durch die Abstandswerte der Ultraschallsensoren, ob er längs oder quer zur Fahrbahn ausgerichtet ist.

In Zukunft sollten Lösungsansätze entwickelt werden, wie ohne Farbahnmarkierungen gearbeitet werden kann. Des weiteren ist es sehr wichtig, dass dieses System mit einem autonomen Fahrsystem verbunden wird. Dies würde das Problem beseitigen, dass der Roboter keine Spurführung zum fahren hat. Somit könnte man problemlos überall vorbeifahren und der Roboter würde seine Fahrtrichtung nicht verlieren. Die Lenkung funktioniert für diesen Modellversuch angemessen präzise. Allerdings sollte diese dennoch überarbeitet werden. Möglicherweise können Alternativen zu Legobauteilen getestet werden, um in Zukunft genauere Lenkbefehle umsetzen zu können.

Anhang

Im Rahmen der Abschlusspräsentation des Projektseminars wurde ein Video gedreht, welches die Funktion des Einparkroboters zeigt: https://www.youtube.com/watch?v=eopOMXVead0.

Literaturverzeichnis

[1] RWTH Aachen (Hrsg.): *RWTH - Mindstorms NXT Toolbox*. https://www.mindstorms.rwth-aachen.de/trac/wiki/Download4.07

[2] Volkswagen AG (Hrsg.): *Der Parklenkassistent: Konstuktion und Funktion: Selbststudienprogramm 389*. https://www.motor-talk.de/forum/aktion/Attachment.html?attachmentId=677825. Version: 2007

[3] *Fachkunde Kraftfahrzeugtechnik*. 30., neubearbeitete Auflage. Haan-Gruiten : Verlag Europa-Lehrmittel, 2013 (Europa-Fachbuchreihe für Kraftfahrzeugtechnik). – ISBN 978–3–8085–2240–0

[4] Math Works (Hrsg.): *Matlab GUI*. https://de.mathworks.com/discovery/matlab-gui.html

Emi.mem – Der Memoryroboter

Julian Fürtig, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Wie bereits seit 2013, fand auch in diesem Jahr 2020 wieder das LEGO Mindstorms Praktikum statt. In diesem sollten die teilnehmenden Studenten eigene Ideen entwickeln und diese mithilfe von LEGO-Mindstorms und MATLAB umsetzen. Ziel war es, nach zwei Wochen bei der Abschlusspräsentation einen funktionierenden Roboter vorzustellen. In diesem Paper wird der Memoryroboter behandelt, der auch ein Ergebnis des diesjährigen Praktikums ist. Ziel des Roboters war das Erkennen von zwölf Memorykarten und danach das richtige zuordnen der Paare. Obwohl einige Probleme aufgetreten waren, war es möglich, am Ende der zwei Wochen einen funktionierenden Roboter zu präsentieren.**

***Schlagwörter*—Emi.mem, LEGO, MATLAB, Memoryroboter, NXT, Otto-von-Guericke-Universität Magdeburg**

I. Einleitung

DER Memoryroboter-Emi.mem sollte in der Lage sein, ein Memoryfeld von 4×3 Feldern zu erfassen und die zusammengehörigen Paare finden. Das selbst gesteckte Ziel war es, einen Roboter zu entwerfen, der einen Nutzen hat. So war es mit Hilfe des Emi.mem möglich, ohne Kontakt zu anderen Menschen Memory zu spielen, oder einfach dem Roboter beim Spielen zuzuschauen. Somit stand das Ziel des Projekts fest. Um diese Idee umzusetzen, standen ein NXT-Baustein der als „Kopf“ funktioniert und auf dem der geschriebene Programmcode „läuft“ zur Verfügung. Weiterhin war es möglich, bis zu drei Motoren und vier Sensoren an den NXT-Baustein anzuschließen. Der Roboter sollte selbstständig die Felder anfahren, die Farben erkennen und diese dann speichern. Anschließend sollte er nacheinander die sechs Farben angeben und die dazugehörigen Paare zeigen. Es sollten zwei Spielmodi über ein GUI wählbar sein, ein Einzelspieler-Modus und dann einen für Memory üblichen Zwei-Spieler-Modus. Im Zwei-Spieler-Modus sollte es möglich sein, gegen den Roboter zu spielen.

II. Vorbetrachtungen

Zur Umsetzung des Roboters wurden zwei Farbsensoren, ein Tastsensor und drei Motoren benötigt. Als Grundlage diente ein Baukasten der LEGO-NXT-Serie und ein weiterer mit LEGO-Technik Teilen. Als erstes war es notwendig, sich mit der Funktion der Sensoren und der Motoren vertraut zu machen. Damit wird sich der nachfolgende Abschnitt beschäftigen.

A. Der Memoryroboter

Der Memoryroboter orientiert sich zur Paarsuche an Farbmarkierungen außerhalb des Feldes, um die richtigen Positionen zu erreichen. Emi.mem beginnt damit, das gesamte Feld einzuscannen, um sämtliche Karten zu erfassen. Anschließend fährt er die Paare einzeln an, um zu zeigen, wo sich diese befinden. Nachdem der Roboter wieder die Ausgangsposition erreicht hat, wird die Farbe via Sprache ausgegeben. Ein Zwei-Spieler-Modus ist ebenfalls vorhanden, um das Memory-Spiel nachzuempfinden.

DOI: 10.24352/UB.OVGU-2020-030

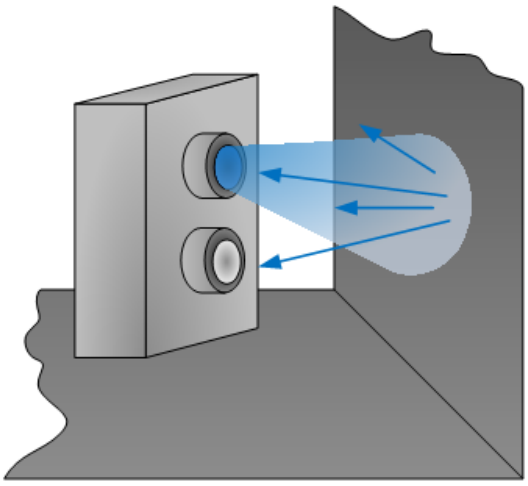


Abbildung 1. Prinzip des Farbsensors

B. Der Farbsensor

Der Farbsensor wird verwendet, um verschiedene Farben zu erkennen. Mit dem RGB-Sensor von LEGO ist es möglich, sechs verschiedene Farben zu erkennen und die entsprechende Farbe als Zeichenkette (String) in MATLAB ausgeben zu lassen. Der Sensor kann durch Anstrahlen der zu erkennenden Oberfläche mit Licht in den Farben Rot, Grün und Blau anhand der Reflexionen die Farbe der Oberfläche ermitteln. Beispielsweise wird Weiß durch die gleich starke Reflexion der drei Farben ermittelt, während Schwarz durch geringe Reflexion der drei Farben bestimmt wird. Wenn nur eine der drei Farben zurückgeworfen wird hat die Karte die entsprechende Farbe. Gelb wird durch eine gleichstarke Reflexion von rot und grün ermittelt, blau wird dabei nicht zurückgeworfen. Die Funktionsweise des Farbsensors ist in Abbildung 1 erkennbar.

C. Bedienung des Roboters

Der Roboter sollte mithilfe eines Tastsensors gestartet werden und nach dem Einlesen der Karten den „Spielvorgang“ starten. Der Programmablauf wurde am Ende mit einem GUI (Graphical User Interface) gestartet und nicht mit dem Tastsensor, da vorher noch der Spielmodus ausgewählt werden muss. Das eigentliche Spiel wurde aber weiterhin über den Tastsensor gestartet.

III. Hauptteil

A. Erste Schritte

Die erste Überlegung war, einen Roboter nach dem Prinzip eines Portalkrans zu bauen, der sich direkt über dem Memoryfeld befindet. Die Umsetzung dieser Idee war aber nicht möglich, da das senkrechte Herablassen des Farbsensors auf die Memorykarten ähnlich des Kranhakens mit LEGO schwierig umsetzbar war. Deshalb wurde auf die aktuelle Lösung zurückgegriffen, den Roboter neben dem Memoryfeld zu platzieren. Nachdem die Position des Roboters feststand, wurde mit ersten Versuchen begonnen, den Roboter neben dem Feld zu bewegen. Dazu wurde ein Grundrahmen aufgebaut, an dem vier Räder befestigt wurden. Zunächst wurde nur eine Achse mit einem außenliegenden Motor angetrieben. Dieses Chassis fuhr entlang der in Abbildung 3 sichtbaren Orientierungsmarken. Auf dieses Chassis wurde dann eine um 90° gedrehte Schienenkonstruktion aus Zahnschienen gesetzt, um die Armbasis entlang der „Feldspalten“ zu bewegen und so die Erreichbarkeit jedes Feldes zu gewährleisten. Auch die obere Basis wurde mithilfe eines eigenen Motors bewegt. Die erste Konstruktion sah vor, einen beweglichen Arm zu verwenden, um auch die hintere Feldreihe zu erreichen. Dies ist in Abbildung 5 dargestellt. Parallel zur mechanischen Umsetzung des Roboters wurde sich mit der Funktionsweise der Farbsensoren vertraut gemacht. Auch das Ansteuern der Motoren und das Abfragen des Tastsensors wurden ausprobiert.

B. Probleme

Die erste Version des Roboters war noch nicht perfekt und benötigte noch einiges an Optimierungen. Der Roboter war zum einen nicht sehr standsicher, zum anderen war durch den beweglichen Arm die Treffsicherheit des Farbsensors auf die Karten nicht immer gegeben. Des Weiteren gab es mehrere Probleme beim Anfahren der unteren Einheit. Eine weitere Herausforderung war, dass der obere Wagen bei ausgefahrenem Arm zum nach-vorn-Kippen neigte. Noch eine Schwierigkeit war, dass der Roboter die „Feldzeilen“ nicht genau genug anfuhr. Weitere Probleme waren, dass die Farbsensoren einen sehr geringen Abstand zur erfassenden Oberfläche und auch sehr große Farbflächen benötigten, damit die Messung zuverlässig funktionierte. Des Weiteren durfte das Umgebungslicht nicht zu hell sein, damit der Farbsensor die richtige Farbe erkannte.

C. Problemlösung

Um das Problem der Standsicherheit zu lösen, wurde der außen montierte Motor des Chassis nach innen verlegt und der NXT auf eine Seite über die Räder des Chassis verschoben, sodass der Schwerpunkt zwischen die Achsen rutschte und der Roboter nicht mehr zum Kippen neigte. Das Kippen des oberen Wagens wurde durch den Umbau von einem beweglichen Arm auf einen festen Arm unterbunden. Um dies zu realisieren, musste die Zahnschienenkonstruktion verlängert werden und der zuvor für das Ausklappen des Arms verantwortliche Motor wurde nun als Gegengewicht für den verlängerten Arm verwendet. Die Treffsicherheit des Farbsensors wurde durch den Wegfall der Armgelenke deutlich

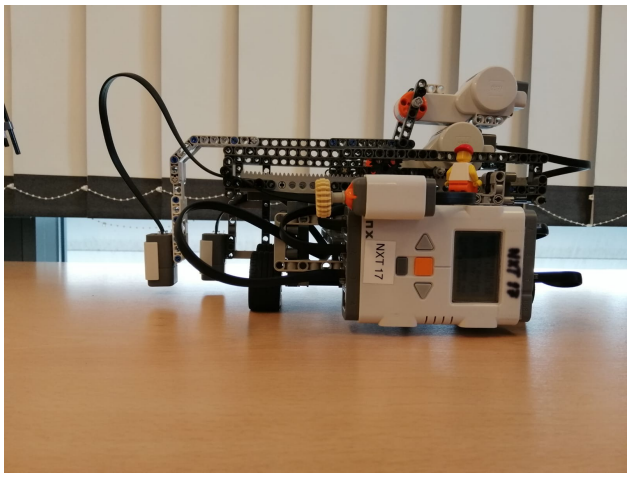


Abbildung 2. Rückansicht des Roboters, gut erkennbar sind die beiden Farbsensoren

verbessert. Ein Problem, welches die steifen Kabel verursachten, war, dass der obere Wagen „entgleiste“. Dieses Problem konnte durch das Anbringen von seitlichen Führungsschienen meist unterbunden werden. Ein Umkippen beim Anfahren konnte ebenfalls durch das Verschieben des Motors in das Chassis sowie durch das Antreiben aller vier Räder gelöst werden. Des Weiteren wurde die Kraft des Motors reduziert und durch den „Allradantrieb“ auch besser verteilt. Die Genauigkeit des Anfahrens der „Feldzeilen“ wurde durch das Anbringen eines zweiten Farbsensors realisiert. Dieser orientierte sich an den Markierungen neben dem Feld. Die rote Markierung legte den Startpunkt des Roboters fest, die schwarzen Markierungen jeweils eine Zeile. Dies stellt Abbildung 3 dar. Der geringe Abstand zur Oberfläche war ein Problem, da sich der Arm mit zunehmender Länge etwas nach unten bog und somit auf den Karten auflag, wodurch er kein Licht „einfangen“ und so nicht mehr messen konnte, weshalb die Höhe des Sensors sehr wichtig war. Das Ausrichten des zweiten Farbsensors zur Orientierung war auch ein Problem, da die Orientierungsmarken zu schmal waren, um durch den Sensor zuverlässig erkannt zu werden. In der finalen Version sind diese Marken noch breiter als in der Abbildung 3, um zu verhindern, dass der Roboter eine Markierung übersah und einfach weiterfuhr. Das hätte zur Folge, dass der gesamte Ablauf neu gestartet und der Roboter wieder manuell ausgerichtet werden müsste. Der Aufbau des Roboters ist in Abbildung 4 zu erkennen.

D. Programm

Nachdem sich das Team mit der Funktionsweise und Ansteuerung der Sensoren und Motoren vertraut gemacht hatte, wurde begonnen, die gewünschten Bewegungen in Quellcode umzusetzen. Als erstes wurde der Vorgang des Einlesens programmiert. Dazu wurden for-Schleifen verwendet, um alle Felder erfassen zu können. Damit der Roboter die einzelnen „Feldzeilen“ richtig anfuhr, wurden die Farbmarkierungen am Rand des Memoryfeldes mit eingebunden. Der Fahrmotor wurde solange bewegt, bis der zweite Farbsensor die schwarze Markierung erkannt hatte. Damit der Farbsensor beim erneuten

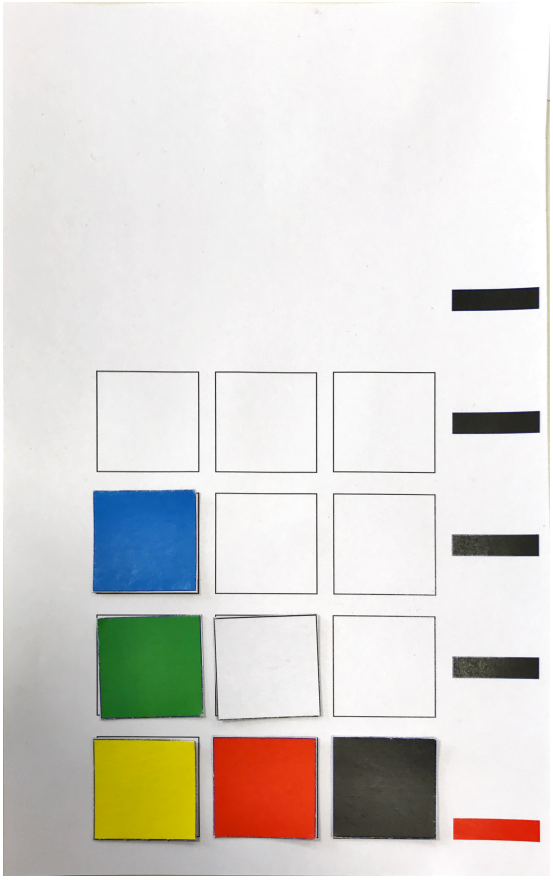


Abbildung 3. Memoryfeld mit Orientierungsmarken

Losfahren nicht nochmal die schwarze Markierung erkannte, wurde eine kurze Pause in das Programm mittels Pause-Befehl eingebaut. Erst nach wenigen Sekunden wurde das Programm fortgesetzt und so das doppelte Erfassen der Markierungen verhindert. Das Einlesen der Karten ist in Abbildung 4 zu sehen. Um einen Startpunkt für das Spiel zu erhalten, wurde die rote Markierung angebracht. Diese Markierung wurde erst nach dem Erfassen des gesamten Feldes benötigt, um das Programm zu vereinfachen, da das Zählen von schwarzen Markierungen einen höheren Programmieraufwand zur Folge gehabt hätte. Das Anfahren der Felder in die andere Richtung auf der zweiten Ebene wurde dann über Laufzeiten des Motors realisiert. Der Motor lief abhängig von dem anzufahrenden Feld eine längere Zeit oder eben eine kürzere. Das Einlesen der Felder erfolgt über ein Array (Feld) welches dem Memoryfeld nachempfunden war.
Nach dem Beenden des Einlesevorgangs begann der Roboter damit, die Felder sequentiell miteinander zu vergleichen. Die erste Memorykarte, in Abbildung 3 die schwarze Karte, wurde mit den restlichen Karten verglichen. Sobald der Roboter die zugehörige Karte gefunden hatte, startete er den zweiten Teil des Programms. In diesem Teil wurde zuerst jedem Feld zugeordnet, wie sich der Roboter bewegen muss. Dies wurde mittels Koordinaten des Arrays realisiert. Jeder Arraykoordinate wurde ein eindeutiger Weg zugeordnet, sodass es möglich war, jedes Feld unabhängig anzufahren. Bei der nächsten Farbe wurde

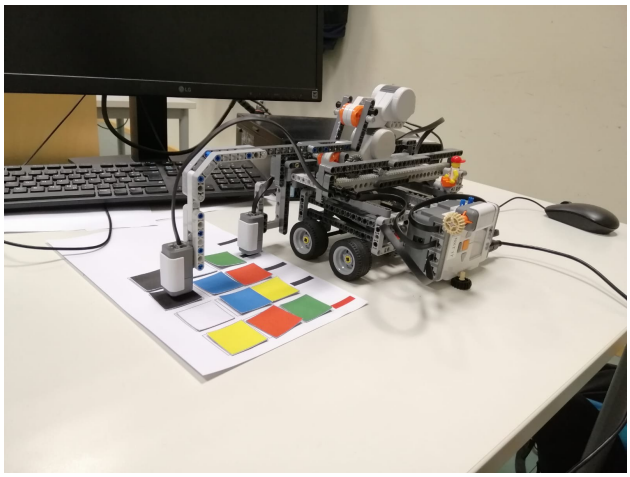


Abbildung 4. Roboter in Aktion

das erste Feld übersprungen und mit dem zweiten Feld weiter gemacht. In der Abbildung 3 ist das die rote Memorykarte. Jetzt verglich er wieder diese Karte mit den anderen und wenn er die dazu passende Karte gefunden hatte, wurden wieder die beiden Felder angefahren und zum Schluss die gefundene Farbe sprachlich ausgegeben. Die Sprachausgabe wurde realisiert, indem zuvor die Farben eingesprochen und anschließen über den im NXT verbauten Lautsprecher ausgegeben wurden.
Der Zweispieler-Modus begann damit, dass der Roboter erst das Memoryfeld einlas. In dieser Zeit konnte auch der menschliche Spieler sich das Memoryfeld merken. Dieser durfte dann beginnen und so lange weiter spielen, bis er einen Fehler beging. Erst dann war der Roboter an der Reihe. Dieser fand jetzt mit einer Wahrscheinlichkeit von 60% ein passendes Paar. Dies war notwendig, da es sonst unmöglich wäre, gegen der Roboter zu gewinnen, da dieser eigentlich nicht „vergessen“ kann. Die Umsetzung erfolgte mittels Zufallsgenerator, welcher mit einer voreingestellten Wahrscheinlichkeit von 60% zu einem Erfolg des Roboters führte, mit einer Wahrscheinlichkeit von 40% zu einem Misserfolg und damit zur Ausgabe einer Fehlermeldung. Die Eingabe des realen Spielers erfolgte im GUI, dort war das Memoryfeld dargestellt. Der Spieler musste dort seine gefundenen Felder markieren, damit der Roboter nicht versuchte, diese Felder zu suchen.

IV. Ergebnisdiskussion

Abschließend, war das Projekt Memoryroboter ein Erfolg. Die schwerwiegendsten Probleme konnten behoben werden und auch der Zweispieler-Modus konnte zum Schluss noch erfolgreich getestet werden. Eine Einzelspieler-Runde dauerte knapp 6 Minuten, welche auf die konstruktionsbedingt langsamen Bewegungen zurückzuführen sind. Ein ungelöstes Problem blieb, dass das Ausrichten des Roboters am Feld nicht immer exakt genug war, sodass er sich nicht parallel zum Feld bewegte, wodurch gelegentlich ein Neustart des Programms erforderlich war. Um dies zu verhindern, war es manchmal notwendig, die Position des Memoryfeldes zu korrigieren, da sonst der Vorgang unnötig lange dauerte. Das Umdrehen der Karten zu realisieren war nicht möglich, da dazu noch mindestens

zwei weitere Motoren nötig gewesen wären. Die Anzahl der Motoren war aber durch den NXT-Baustein auf maximal drei beschränkt.

V. Zusammenfassung und Fazit

Der Memoryroboter-Emi.mem war in der Lage alleine, aber auch gegen einen anderen Spieler zu spielen. Im Ein-Spieler-Modus benötigte der Roboter etwa 6 Minuten für ein Spiel, während im Zwei-Spieler-Modus, abhängig von der Eingabegeschwindigkeit des menschliches Mitspielers, deutlich mehr Zeit erforderlich war. Zukünftig wäre es möglich, mit Hilfe eines zweiten NXT das Umdrehen der Karten zu ermöglichen, sodass es dem realen Memoryspiel näher käme. Des Weiteren wäre es möglich, den Farbsensor durch ein Webcam zu ersetzen, und so nicht nur Farben, sondern auch Motive zu erkennen, und die Paare zu finden.

Anhang

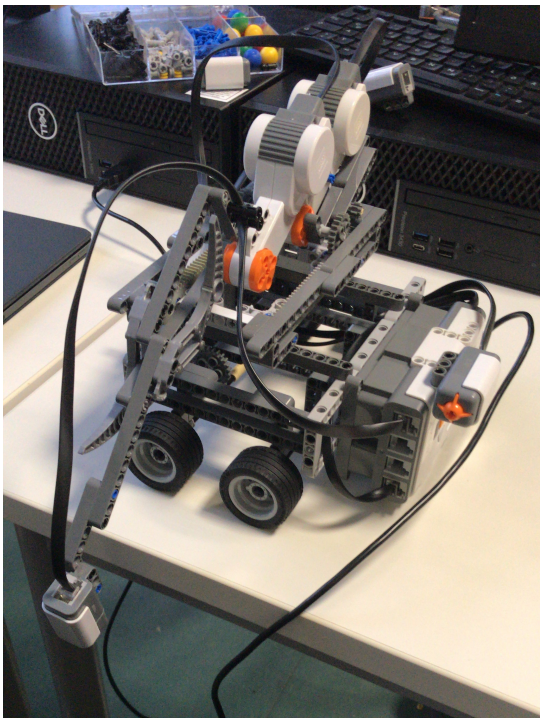


Abbildung 5. Erster Prototyp mit beweglichem Arm und kurzem oberen Aufbau

Literaturverzeichnis

Abbildung 1 - Felix Bangemann: Vortrag Sensoren.https://elearning.ovgu.de/mod/resource/view.php?id=44971. Februar 2020

Emi.mem - Ein Roboter, Der Memory Spielt

Nikita Lorber, EIT
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Dieses Paper beschreibt die Arbeit an einem Roboter, der im Rahmen des Projektseminars Elektrotechnik/Informationstechnik 2020 an der Otto-von-Guericke-Universität Magdeburg erstellt wurde. Seine Aufgabe war es, Memory auf einem Feld aus 3×4 Karten zu spielen. Genutzt wurde ein Set vom LEGO Mindstorms NXT und die Programmieroberfläche MATLAB. Es wird genauer auf den Bau des Roboters und die Programmierung eingegangen.**

***Schlagwörter*—LEGO, MATLAB, Memory, Mindstorms, NXT, Roboter**

I. Einleitung

MEMORY ist ein bekanntes Gesellschaftsspiel. Man spielt es nach dem Pairs-Prinzip, in welchem Paare gleicher, verdeckt aufliegender Kärtchen durch Aufdecken im Wechsel der Spieler erkannt werden müssen [1]. Es gibt allerdings auch Adaptionen (besonders in digitalen Versionen), in welchen man in den ersten Sekunden alle Kärtchen aufgedeckt sehen kann, womit es mehr zu einem Merkspiel wird. Diese variante eignet sich sehr gut zur Programmierung eines Roboters.

II. Vorbetrachtung

Um ein Memoryspiel mit den eingeschränkten Möglichkeiten des LEGO-Roboters umsetzen zu können, musste das Spielfeld ein wenig eingeschränkt werden: Die Anzahl der Karten wurde von 66 auf 12 reduziert, und anstelle von verschiedenen Motiven sind die Karten jeweils einfarbig (rot, gelb, grün, blau, weiß, schwarz).

Der Roboter besteht aus einem programmierbaren Stein (genannt Brick), welcher 7 anschlüsse hat: 3 für Motoren und 4 für eine Auswahl an Sensoren (u.a. Farbsensor, Tastsensor, Ultraschallsensor). Für das Memoryspiel sind besonders die Motoren und ein Farbsensor von Bedeutung.

III. Bau und Programmierung des Roboters

A. Aufbau

Der Roboter besteht aus 2 Teilen: Der untere Teil bildet die Basis, die sich anhand eines Motors, welcher mit einem Allradgetriebe 4 Räder bewegt, nach links und rechts fahren kann. Zusätzlich befinden sich an dieser Basis auch der Brick, ein Tastsensor (welcher dem Starten Programmes dient) und ein Farbsensor, welcher anhand von einer roten und vier schwrzen Markierungen dem Roboter eine genaue Orientierung ermöglicht. Der obere Teil besteht aus einem Arm, an welchem ein Farbsensor zur Erkennunng der Karten angebracht ist. Dieser Arm kann mithilfe eines Motors auf Schienen, welche auf der Basis angebracht sind, vorwärts und rückwärts fahren. Zusätzlich ist an der Rückseite des Armes ein zweiter Motor angebracht, welcher allerdings lediglich als Gegengewicht des sehr frontlastigen Arms dient. Die Spielfläche ist ein bedrucktes Blatt Papier mit 12 Feldern (3 Felder lang, 4 Felder breit), welche jeweils mit einem Kärtchen bedeckt werden. Des Weiteren befinden sich dort ebenfalls die rote und die schwarzen Markierungen für den unteren Farbsensor.

B. Programmierung

1) Grundidee: Der Roboter soll zunächst der Reihenfolge nach alle Kärtchen einlesen. Nachdem er dann zur Ausgangsposition zurückgekehrt ist, zeigt er auf jeweils zwei Kärtchen und signalisiert, dass er die dazugehörige Farbe erkannt hat. Wenn er alle 6 Paare gefunden hat, soll er zur Ausgangsposition zurückkehren.

2) Herangehensweise: In der Programmierumgebung MATLAB ist die Programmierung mithilfe von Matrizen sehr anschaulich. Daher wurde das Spielfeld in einer 3×4 Matrix dargestellt. Zunächst wurde jedem Element der Matrix eine Position zugeordnet, and welcher sich der Farbsensor befinden musste, um das jeweilige Feld auf der Spielfläche einlesen bzw. anzeigen zu können. Um dies möglichst genau zu erzielen sind dem Motor, welcher den Arm vor und zurück bewegt, für jede der drei Zeilen ein Winkel zugeordnet. Für die genaue Positionierung zu den Spalten werden Markierungsstreifen auf der Spielfläche genutzt: Der rote Streifen markiert die Ausgangesposition, die 4 schwarzen Streifen markieren jeweils die Spalten (welche mithilfe eines Zählers genau erkannt werden).

Jeder Farbe, die der Farbsensor erkennen kann, ist eine Zahl zugeordnet. Somit kann beim Einlesen der Kärtchen die Matrix mit den jeweiligen Zahlen ausgefüllt werden, um somit die Reihenfolge festzulegen, in welcher später die einzelnen Positionen auf der Spielfläche abgefahren werden. Um sicher zu gehen, dass der Roboter die Farbe einlesen konnte (wird durch variierende Lichtverhältnisse erschwert), gibt er immer wenn er die Farbe eines Kärtchens erkannt hat einen Signalton ab. Er gibt ebenfalls einen Signalton ab, wenn er die Paare anzeigt, um dies genauer erkennbar zu machen. Jedes mal, wenn er ein Paar angezeigt hat spielt er außerdem eine Sprachdatei ab, die der jeweiligen Farbe entspricht (z.B. „Rot gefunden“). Anschließend fährt der Roboter zur Ausgamgsposition zurück und beendet das Programm.

Zusätzlich wurde die Funktion eines 2 Spieler Modus hinzugefügt. Mithilfe einer GUI kann man vor Start des Programmes zwischen 1 Spieler Modus (Roboter löst das Spiel alleine) und 2 Spieler Modus (Mensch gegen Roboter) wählen. In diesem 2 Spieler Modus liest der Roboter zunächst wieder alle Kärtchen ein. Darauf hin wartet der Roboter, sodass die Kärtchen umgedreht werden können und der Spieler seinen

DOI: 10.24352/UB.OVGU-2020-031

ersten Zug machen kann. Ist der Spieler fertig mit seinem Zug, gibt er in der GUI in einer Matrix an, welche Kärtchen er gefunden hat. Dann kann der Spieler anhand der GUI dem Roboter signalisieren, dass er nun am Zug ist. Für jedes Paar, welches der Roboter nun auswählen kann, besteht eine Chance von 40%, dass der Roboter kein Paar findet (er vergisst gewissermaßen die Paare, sodass der Modus dem Spieler eine Chance gibt). Der Roboter ist am Zug, bis er entweder die Paare vergessen hat (danach ist der Spieler wieder am Zug), oder bis alle Paare gefunden wurden.

IV. Ergebnisdiskussion

Das Ergebnis dieses Projekts ist ein Roboter, der eine vereinfachte Version des Memoryspiels spielen kann, und zusätzlich nach Bedarf gegen einen Menschen spielen kann.

Es sind während der Bearbeitung des Roboters ein Paar Probleme aufgetreten, die allerdings vorwiegend physischer Natur waren: Ungleichgewicht des Roboters durch den frontlastigen Arm, instabile Bauweise des Roboters und lichtbedingte Schwierigkeiten bei der Farberkennung waren die Hauptprobleme, welche allerdings alle größtenteils gelöst wurden.

V. Zusammenfassung und Fazit

Im Rahmen dieses Projektes ist ein gut funktionierender Roboter entstanden, welcher in der Lage ist, allein oder gegen einen anderen Spieler Memory zu spielen.

In Zukunft könnte das Programm noch ein wenig abgeändert werden, sodass der Roboter nicht bevor er spielt alee Kärtchen einlesen muss, sondern dass er dies während er spielt durchführt, sodass er näher an die ursprüngliche Spielweise von Memory herankommt. Dies würde zusätzlich die Notwendigkeit des zufälligen Vergessens der Paare im 2 Spieler Modus entfernen.

Anhang

Literaturverzeichnis

[1] Wikipedia, the free encyclopedia: *Memory (Spiel)*. https://de.wikipedia.org/wiki/Memory_(Spiel). Version: März 2020

’Emi.mem’ – MemoryBot

Tim Jan Müller, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In diesem Paper soll es spezifisch um die Schwierigkeiten und Herausforderungen gehen, die es bei der Umsetzung der Idee gab, einen Memory spielenden Roboter zu entwerfen, der sowohl allein, als auch als Spielpartner fungieren soll. Diesbezüglich werden im folgenden sowohl verworfene Ideen und Konzepte, als auch das abgeschlossene Modell vorgestellt und detailliert erläutert.**

***Schlagwörter*—FEIT, LEGO-Mindstorm, Memory, Roboter**

I. Einleitung

IN vielen Bereichen des alltäglichen Lebens werden Prozesse durch autonom fungierende Robotoren erleichtert. Ob zu Hause, wie Staubsaug- und Kochroboter (z.B. ThermoMix), oder in der Industrie, als Hilfe bei der Fließbandarbeit, maschinelle Hilfsmittel sind nicht mehr wegzudenken. Jedoch steckt hinter diesen meist kompakten Anlagen eine Menge Programmier- und Konstruktionsarbeit.
Um den Studenten der FEIT dieses Thema näher zu bringen, findet in unserer Universität seit 8 Jahren das LegoMindstorms-Projekt statt. Die Studenten können sich hier am Entwurf, der Konstruktion und der Programmierung eines eigenen Roboters testen. Die einzige Einschränkung war dabei, dass dieser aus dem vorhandenen Material des Lego-Mindstorm-Set bestehen soll. Der Zeitraum, in dem das Projekt stattfand, lag bei 2 Wochen, wobei die effektive Arbeitszeit eine Woche war, da zuerst in das Thema eingeleitet und auf eventuelle Herausforderung bestmöglich vorbereitet wurde.
Im Rahmen des Projektes hat sich so eine Projektgruppe, bestehend aus Nikita Lornik, Julian Fürtig und Tim Müller, gebildet, welche sich für die Konstruktion eines Memory-Roboters entschied. Die klare Zielsetzung lag dabei darauf, ein benutzerfreundliches Interface zu schaffen, mit dem es jeden möglich sein sollte mit ’Emi.mem’ zu interagieren. Außerdem sollte der Roboter logisch und fehlerfrei agieren können. Die Spielidee von Memory ist wahrscheinlich allgemein bekannt, jedoch wird im Folgenden noch einmal das etwas abgewandelte Spielkonzept erläutert. Der Roboter soll die Plättchen einmal einscannen und sich die Farben mit ihren Positionen merken. Im Folgenden werden die Plättchen umgedreht und der Roboter bzw. der Spieler (je nach Spielmodus) drehen Plättchen um. Im Fall des Solo-Modus, in dem der Roboter allein mustergültig Paare findet und zuordnet, wird er alle Paare der Reihe nach aufzeigen. Beim Spielen im Spieler-Modus soll der Roboter Plättchen ”vergessen”, so dass ein echter Spieler auch eine Chance auf den Sieg hat. Der Rest verläuft wie beim normalen Memoryspiel. Zur Entwicklung eines Konzepts wurden verschiedene Ideen in Betracht gezogen, welche folgend erläutert werden.

DOI: 10.24352/UB.OVGU-2020-032

II. Vorbetrachtungen

Zu einem funktionierenden Roboter gehören mehrer Kompenten. Zum einen natürlich das Design bzw. die Kontruktion desgleichen, und zum anderen das äußere Umfeld, Zubehör, sowie die Programmierung und dazugehörige Methoden. Diesbezüglich sind einige Grundideen aufgekommen.

A. Material

Als Material wurde das Lego-Mindstorms-Set zur Verfügung gestellt. Dieses beinhaltet einen NXT-Controller, der als Rechner fungiert, Motoren, Tast-, Farb-, Helligkeits-, Ultraschall-, und Tonsensoren, sowie diverse Legosteine, aus denen der Roboter gebaut werden soll. Es durften jedoch auch weitere Hilfsmittel, wie z.B. eine Webcam verbaut werden. Es gibt aber auch Einschränkungen. So darf zum Beispiel nur ein NXT-Controller verwendet werden, welcher wiederum nur 3 Motoren und 4 Sensoren ansteuern kann. Zusammengefasst sind durch die große Vielfalt an Sensoren und den immerhin 3 Motoren eine relativ breite Varietät an Möglichkeiten, einen Roboter zu entwickeln gegeben.

B. Struktur - Hallenkran

Als Grundstruktur für ’Emi.mem’ wurde ein Hallenkran als Vorbild genommen.

Die Funktionsweise eines solchen ist effektiv wie simpel. Er besitzt zwei bewegliche Lager die senkrecht aufeinander stehen und dem Kran so ermöglichen, die komplette Horizontalbewegung abzudecken. Er kann dadurch nach links/rechts und nach vorne/hinten fahren. Des Weiteren besitzt ein Hallenkran, wie in Abbildung 1 abgebildet, einen herabfahrbaren Haken. So kann dieser mit einfachen Mitteln jeden Punkt in einem 3-dimensionalen Koordinatensystem anfahren. Darüber hinaus ist es spezifisch auf das Memoryspiel bezogen vorteilhaft, da dieses symmetrisch angeordnet ist und der Kran so nicht schräg fahren muss, was die Bedienung um so simpler macht. Jedoch bringt dieses Design auch Nachteile mit sich, da es dem Kran nur schwer möglich ist Kärtchen umzudrehen, weil alle drei, für dieses Projekt verfügbaren, Motoren bereits ausgenutzt waren. Außerdem würde ein solches Gerüst den freien Zugang zum Memoryfeld blockieren, weshalb diese Idee insgesamt noch einige Verbesserungen benötigte.

C. Das ’optimale’ Memoryfeld

Unabhängig von der Ausführung des Designs des Roboters muss auch das Design des Memoryfeldes bestmöglich gewählt werden. Da der Roboter an das Prinzip des normalen Memorys angelehnt sein soll, musste das Spielfeld auch dementsprechend nicht wirklich verändert werden. Damit es dem Zuschauer

Abbildung 1. Modell eines Hallenkrans [1]

nicht zu langweilig wird, und der Roboter auch eine Weile beschäftigt ist, wurde die Größe des Spielfeldes auf 3×4 bzw. 4×4 festgelegt. Außerdem ist auch eine Anzahl von 6 bzw. 8 zu findenden Paaren eine angemessene Menge. Die Wahl der Markierung der Karten war keine Herausforderung, da es durch die vorgegeben Farbsensoren nicht viel Auswahl gab, außer die Karten verschiedenfarbig zu gestalten. Da die Sensoren nur 6 Farben als Wortlaut ausgeben können (rot, blau, gelb, grün, schwarz, weiß), wurde sich letztendlich dazu entschieden zwölf Kärtchen und dementsprechend 6 Paare einzubauen. Die logische Schlussfolgerung, war die Verwendung des 3×4 Memoryfeldes.

D. Programmentwicklung

Die software-technische Umsetzung dieses Projektes sollte in MATLAB stattfinden. Die Programmiersprache ist vergleichbar mit C, jedoch etwas leichter verständlich und vor allem geeigneter um Grafiken auszugeben oder ein Graphical User Interface, auch kurz GUI, zu gestalten.
Die Grundidee für die Programmierung ist, den Code möglichst einfach und verständlich zu schreiben. 'Emi.mem' sollte zwei Spielmodi durchführen können. Im Solo-Modus war generell geplant, dass er mit dem Scannen anfangen und die Farben entsprechend in einer Matrix speichern, diese später wieder auslesen, Paare finden, diese zuordenen und aufzeigen können soll. Die Verwirklichung des Spieler-Modus wurde sich so vorgestellt, dass der Roboter beim Auslesen der Farben zu einer gewissen Wahrscheinlichkeit Felder auslässt und diese so "vergessen"werden.
Ziel war außerdem die Entwicklung eines übersichtlichen und verständlichen GUI, welches dem Spieler die Spielregeln aufzeigt und eine leichte Bedienung ermöglicht.
Mit diesen Grundideen als Voraussetzungen, wurde das Projekt umgesetzt.

III. Umsetzung

In diesem Abschnitt wird auf den Zielpunkt des Projektes und auf Zwischenschritte und -überlegungen desselbigen eingegangen.

A. Aufbau des Roboters

Wie bereits erwähnt, wird der Hallenkran als Vorbild für das Design unseres Roboters genommen. In Abbildung 3 lässt sich erkennen, dass 'Emi.mem' aus zwei Ebenen besteht.

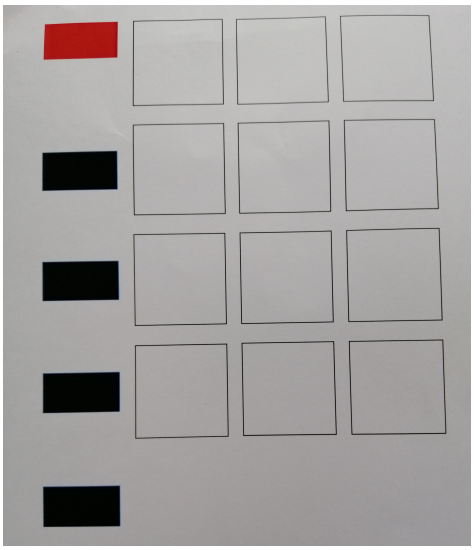


Abbildung 2. Design des Memoryfeldes

Die untere Ebene des Roboters ist auf vier Gummireifen gelagert, welche ihn horizontal nach links und rechts entlang der in Abbildung 2 erkennbaren Rot-Schwarz-Makierung fahren lassen. Dabei sind die Räder als Allradantrieb geschalten, um zu verhindern, dass ein Rad hinterherhinkt und der Roboter so zur Seite abdriftet. Dieser Punkt hatte zunächst einige Schwierigkeiten bereitet, da zu diesem Zeitpunkt kein Motor für eine Lenkung zu Verfügung stand. Durch den neuen Antrieb konnte dieses Problem jedoch relativ simpel gelöst werden.
Auf der zweiten Ebene ist der feste Arm mit Hilfe von 4 Zahnrädern aufgelagert. Dieser lässt sich so auf dieser Ebene horizontal nach vorne und hinten bewegen. Hier wurde sich für Zahnräder entschieden, um den Roboter auf Schienen fahren lassen zu können, damit so, wie bereits beschrieben, das abdriften verhindert werden kann. Am Ende des Armes befindet sich der zu bewegende Farbsensor.
Zu Beginn des Projektes, wurde sich dazu entschieden, diesen Arm beweglich anzubringen, sodass 'Emi.mem' ähnlich wie ein Hallenkran, auch eine vertikal Bewegung durchführen kann. Jedoch hatte dieser keine wirkliche Daseinsberechtigung und hat nur zu weiteren Problemen geführt, weshalb zum Abschluss des Projektes, entschieden wurde, diesen zu demontieren. Das führte wiederum dazu, dass nun wieder einen Motor frei zur Verfügung stand. Da jedoch der Arm bei dem Befahren der zweiten Ebene noch sehr locker gelagert war, wurde der nun übrige Motor einfach als Gegengewicht verwendet.
In Abbildung 3 kann noch ein weiterer Farbsensor auf der linken Seite, einen Tastsensor zu 'Emi.mem's' rechten und den NXT-Controller, an dem der Tastsensor befestigt ist, erkannt werden. Der zweite Farbsensor auf der linken Seite des Roboters, ist dafür zuständig, die Markierungen in Abbildung 2 zu erkennen. Dies erklärt auch, warum diese versetzt zum Memoryfeld angebracht sind. Der Tastsensor war nur für die Präsentation des Solo-Modus von Bedeutung, um den Roboter nach jedem gefundenen Paar anhalten zu lassen und wieder starten zu können. Der NXT-Controller ist an der Seite des Roboters angebracht, um das Gleichgewicht des Konstrukts

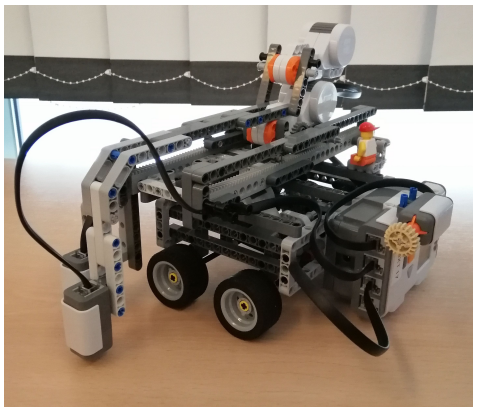


Abbildung 3. ’Emi.mem’

nicht zu sehr zu gefährden. Die Funktion des Controllers ist es, die eingelesenen Daten zu sammeln und den Austausch mit dem Computer zu tätigen.

B. Programmablauf

Der Roboter ist, wie bereits erwähnt, auf 2 Spielmodi ausgelegt, wobei die ersten beiden Schritte komplementär ablaufen.
Zunächst steht ’Emi.mem’ auf dem roten Feld, welches als Startpunkt fungiert. Dann fährt er den ersten schwarzen Balken an und scannt die 3 Felder der entsprechenden Reihe ein. Dies wiederholt sich solange, bis er den vierten schwarzen Balken angefahren hat. Im Folgenden fährt er bis zum roten Startpunkt zurück. Damit ist der Scanvorgang beendet und die Farben werden alle in einer Matrix gespeichert.

Nun beginnen sich die beiden Spielmopdi zu unterscheiden. Im Solo-Modus liest der Roboter die Daten der Matrix einzeln aus und sucht nach Paaren. Sobald er ein Paar gefunden hat, werden beide Felder angefahren und durch einen Signalton aufgezeigt. Im Anschluss wird noch eine Sound-Datei mit der entsprechenden Farbe abgespielt. Dies wird solange wiederholt bis 6 Paare gefunden wurden. Danach begibt sich der Roboter zum Startpunkt zurück und ist wieder einsatz- bereit.
Im Spieler-Modus ist nach Emi.mem’s Scanvorgang zunächst der Spieler am Zug. Nachdem die Karten umgedreht wurden, darf dieser nun versuchen Paare zu finden. Findet er kein Paar mehr, müssen die bereits gefundenen Paare, aus Sicht der Schwarz-Rot-Markierung, in das GUI übertragen werden. So weiß der Roboter, welche Paare noch zu finden sind, ohne nochmal das gesamte Feld scannen zu müssen. Nun geht ’Emi.mem’, wie bereits im Solo-Modus, alle übrigen Felder der Reihe nach durch und überprüft auf Paare. Jedoch findet die Überprüfung nur unter einer gewissen Wahrscheinlichkeit statt. Das bedeutet, dass manche Felder beim Überprüfen der Matrix ausgelassen werden und der Roboter diese sozusagen ”vergisst”. Falls nach diesem Schritt noch nicht alle Felder gefunden wurden, ist der Spieler wieder am Zug und der Vorgang wiederholt sich.
In Abbildung 4 ist der, so eben vorgestellte, Solo-Modus noch einmal veranschaulich dargestellt.

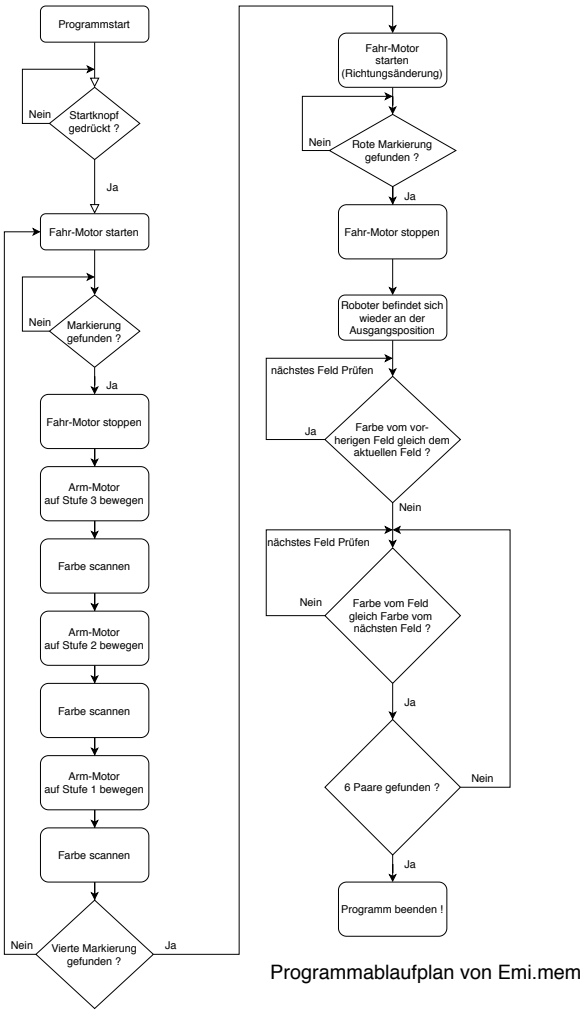


Abbildung 4. PAP des Solo-Modus

C. GUI - Graphical User Interface

Zu guter Letzt wird noch auf den Aufbau und die Funktionsweise des entworfenen GUI eingegangen.

Es ist erkennbar, dass sich im oberen Bereich des Bildschirmes (Abbildung 5) ein Drop-Down-Menü befindet, in welchem der zu benutzende Spielmodus auszuwählen ist. Etwas darunter befindet sich ein großer, weißer Kasten. Nach der Auswahl des Spielmodus werden hier die Spielregeln und die Funktionsweise des GUI aufgelistet. Nun beginnt der eigentlich spannende Teil. Wird der Solo-Modus ausgewählt, so muss nur auf ’Spiel starten...’ gedrückt werden, und ’Emi.mem’ beginnt seinen Zug. Ist jedoch der Spieler-Modus ausgewählt, so muss zunächst der Knopf ’Emi.mem ist am Zug’ betätigt werden. Der Roboter startet nun das Scannen des Spielfeldes und hält dann wieder an. Nachdem der Spieler nun seinen Zug erledigt hat, muss er gefundene Pärchen in die Matrix, über dem vorher gedrückten Knopf, eintragen. Dies muss aus der Sichtweise des Roboters heraus geschehen. Danach muss ’Übernehmen’ geklickt werden um die Änderungen zu speichern. Mit einem erneuten Klick auf ’Emi.mem ist am Zug’ startet der Roboter seinen Durchlauf.

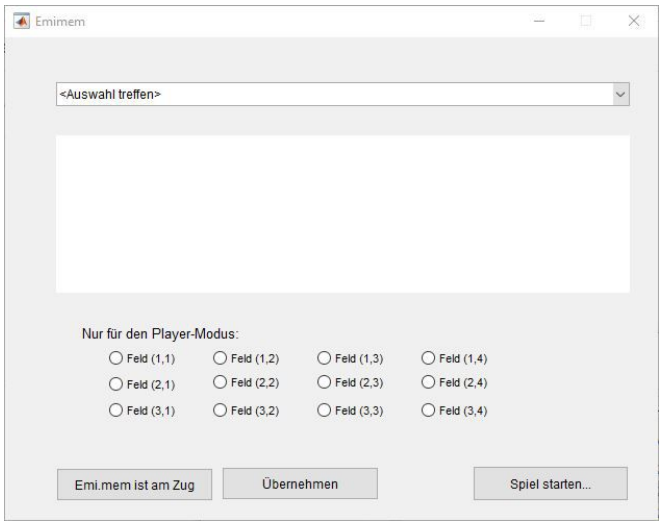


Abbildung 5. Graphical User Interface

Nun geht das Spiel, wie in Abschnitt B. beschrieben, weiter.

IV. Ergebnisdiskussion

Letztendlich ist der Roboter voll funktionsfähig und erfüllt seinen Zweck. Trotz kleinerer Hindernisse, wie einigen Programmierschwierigkeiten und der allgemeinen Stabilität an manchen Stellen, war es möglich den Roboter zu entwerfen und die zu Beginn konstruierte Idee in die Tat umzusetzen. Jedoch gibt es trotzdem kleinere Mängel, die bis jetzt noch nicht komplett behoben werden konnten.
Ein solcher ist zum Beispiel der Farbsensor. Dieser funktioniert unter bestimmten Lichtverhältnissen nur bedingt gut, worauf leider nur relativ wenig Einfluss genommen werden konnte. Als einzige Gegenmaßnahme dafür wurden die Farbsensoren so nah wie möglich an der Oberfläche angebracht, um die optimale Farbaufnahme zu ermöglichen. Da der Roboter jedoch essentiell auf die beiden Farbsensoren zählt, kann das gesamte Programm, durch einen Fehler beim Scannen der Farben, nicht mehr richtig funktionieren.
Ein weiteres Problem ist das Spurhalten des Roboters. Es konnte zwar die obere Ebene konstant zum geradeaus fahren gebracht werden, jedoch kann es beim längeren Benutzen oder ungenauem Ausrichten der unteren Räder, schnell zum abdriften kommen. Trotz einiger Versuche, ist es nicht gelungen hier eine geeignete Schiene zu entwerfen, da diese meist mit dem zweiten Farbsensor kollidiert ist. Jedoch kann das Problem manuell, durch ein leichtes Drehen des Memoryfeldes, behoben werden.
Ein letztes Problem ist die lange Scanzeit, die man mit einer Webcam hätte verkürzen können. Der Grund für die Entscheidung der Verwendung eines Farbsensors statt Webcam, ist größtenteils dem Projektnamen geschuldet. Da sich an den im Set enthaltenen Materialien orientiert wurde lag hier die Idee, die von Lego gestellten Farbsensoren zu verwenden, nicht allzu fern, auch wenn dieser Weg etwas unpraktischer und umständlicher ist.

V. Zusammenfassung und Fazit

Insgesamt ist die Entwicklung des Memoryroboters sehr zufriedenstellend abgelaufen. Das Projekt bereitete viel Freude, insbesondere durch die Möglichkeit eigene Ideen realisieren zu können. Das Projekt wurde innerhalb der gegebenen Zeit umgesetzt, und der Bau- und Programmiervorgang konnten, ohne große Vorkenntnisse des Teams zu diesem Thema, erfolgreich zum Abschluss gebracht werden.
Hätte mehr Zeit zur Verfügung gestanden, so hätten auch noch weitere Ideen in den Roboter eingebracht werden können. Es könnte beispielsweise, der dritten Motor für eine Lenkung verwendet werden, die das Abdriften des Roboters verhindert. Des Weiteren wäre auch eine Implementierung verschiedener Schwierigkeitsgrade für den Spieler-Modus möglich. Außerdem könnte ein ganz neuer Spielmodus hinzugefügt werden, in dem, wie beim echten Memory, gespielt wird, ohne die Plättchen vorher zu Scannen und sich der Roboter per Trial-and-Error die Pärchen sucht. Hätte man mehr Motoren und mehr Anschlüsse, also dementsprechend mehr NXT-Controller, könnte man auch eine Apparatur bauen, die es dem Roboter ermöglicht, die Plättchen selbstständig umzudrehen.
Wie man sehen kann, sind die Grenzen eines Memoryroboters unendlich. Man kann das Konzept auch auf andere Spiele übertragen und 'Emi.mem' so zu einem 'Universal-Gegner' in jeglichem Spiel verwandeln. Jedoch ging es in diesem Projekt nicht darum, eine Maschine zu entwickeln, welche die Anwendung von Robotoren im Allgemeinen neu definiert, sondern nur darum, sowohl den Studenten, als auch jedem der dies liest, zu zeigen, dass die Automatisierung, die zur Zeit in jedem Lebensbereich stattfindet, nichts übermenschliches ist und trotzdem grenzenlose Möglichkeiten offenbart.

Literaturverzeichnis

[1] ABUS - Mehr bewegen: *Hallenkrane/Laufkrane*. https://www.abus-kransysteme.de/krane/laufkrane. Version: März 2020

Farberkennungsgesteuerte Ballschussmaschine

Fabian Meyer, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Auch in diesem Jahr wurden an der Otto-von-Guericke-Universität Magdeburg Studenten des ersten Semesters des Studiengangs Elektro- und Informationstechnik im Projektseminar LEGO Mindstorms damit betraut, einen Roboter zu konstruieren und in MATLAB zu programmieren, der selbständig eine Aufgabe bewältigen soll. In diesem Paper wird die Entwicklung, Konstruktion und Programmierung eines Roboters beschrieben, welcher in der Lage ist, durch Zielerkennung farblich markierte Objekte im Raum zu erkennen, diese anzusteuern und letztendlich auch mit einer Ballschussmaschine zu beschießen.**

***Schlagwörter*—Ballschussanlage, Fahrzeug, LEGO Mindstorms, Roboter, Zielerkennung**

I. Einleitung

DIE Erkennung von Objekten durch Computer, ob von markierten oder unmarkierten spielt eine große Rolle in vielen Bereichen unserer modernen Welt. Ob in medial sehr wirksamen, und allgemein stark umstrittenen Bereichen wie der militären Anwendung, oder in, dem Ottonormalverbraucher eher verschlossenen Gebieten, wie der zivilen Logistik, solche Erkennungssysteme spielen vielerorts eine wichtige Rolle.

II. Vorbetrachtungen

A. LEGO Mindstorms

Erst einmal sollten die zur Verfügung stehenden Bauteile ein wenig genauer angesehen werden. Das LEGO Mindstorms Education Set besteht aus dem NXT-Brick, also dem zentralen Baustein des Roboters, welcher vier Sensoren-Eingänge und drei Motoren-Ausgänge zur Verwendung zur Verfügung stellt. Zum Anschließen sind fünf verschiedene Sensoren verfügbar: Ein einfacher, binärer Tastsensor, ein Geräuschsensor, ein Farbsensor, ein Lichtsensor und ein Ultraschallsensor [1]. Daraus, bis zu drei interaktiven Servomotoren und normalen LEGO-Technik-Bauteilen sollte ein Roboter konstruiert werden.

B. MATLAB

Für die Programmierung des Roboters wurde MATLAB verwendet, eine Software zur Lösung von mathematischen und ingenieurtechnischen Problemstellungen. Entwickelt wird MATLAB von MathWorks, welches laut eigener Angabe „der führende Entwickler von Software für mathematische Berechnungen für Ingenieure und Wissenschaftler“ [2] sei. Für die Kommunikation zwischen MATLAB und dem NXT wird die dazu bestimmte Software des Lehrstuhls für Bildverarbeitung der RWTH Aachen verwendet [3].

DOI: 10.24352/UB.OVGU-2020-033

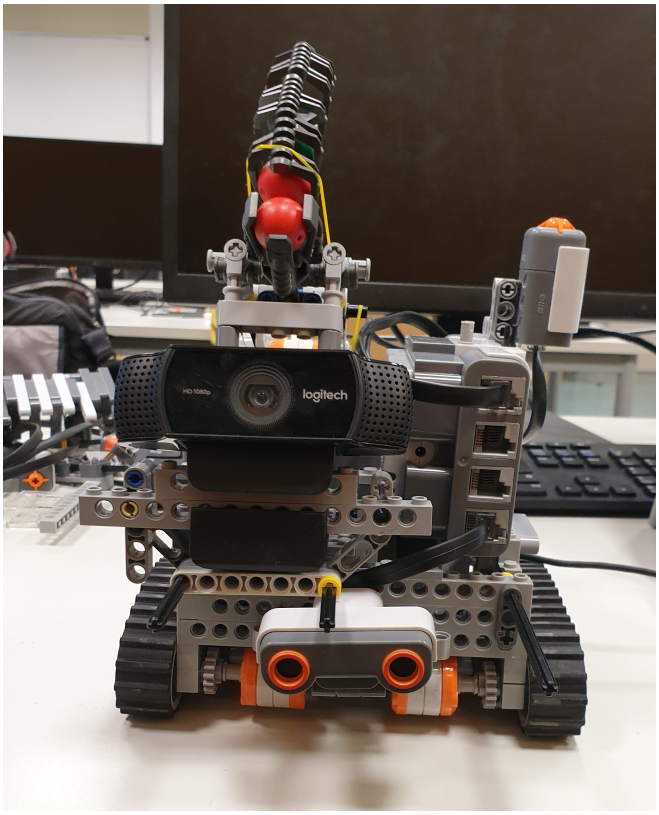


Abbildung 1. Frontansicht des Roboters

III. Konstruktion und Programmierung

A. Zwei Funktionsmodi als Grundidee

Die Idee zu den grundlegenden Funktionsmodi des Roboters war von beginn an, dass der Roboter über einen autonomen und einen manuellen Modus verfügen sollte. Dabei war das Problem zu überwinden, dass der LEGO-NXT-Baustein nur 4 Sensor-Eingänge hat, welche bei der erdachten Fernsteuerungs-Architektur bereits alle belegt wären, und uns somit keine für den autonomen Modus mehr übrig wären. Dazu wurde ein System entwickelt, wobei die Kabel der Fernbedienung für den manuellen Modus erst angesteckt werden müssten, um diesen zu benutzen. Dies wurde so realisiert, dass bei Programmstart entweder ein bestimmter und bereits initialisierter Tastsensor der Fernbedienung für den manuellen Modus, oder ein bereits am Roboter fest verbauter Sensor, welcher den autonomen Modus startet gedrückt werden müssen. Der für den autonomen Modus benötigte Ultraschall-Sensor und der zusätzliche Tastsensor am Roboter sind bei Initialisierung bereits verbaut und verbunden, und letzterer muss ausgelöst

werden, um den autonomen Modus zu starten. Dies ist auch im Programmablaufplan in Abbildung 6 dargestellt.

B. Autonomer Modus

Im autonomen Modus macht der Roboter immer wieder Bilder mit seiner Webcam, welche über USB an einen Computer angeschlossen ist und analysiert diese dann auf rote Objekte. Findet er keins, dreht er sich ein Stück weg und fährt vorwärts, wobei er auch fortwährend Bilder aufnimmt und bei Objektfund sofort anhält. Stößt der Roboter auf eine Wand, welche er mit seinem Ultraschallsensor erkennt, fährt er ein Stück zurück, dreht sich von ihr weg und setzt seine Fahrt fort. Sobald er ein Ziel in der Kamera gefunden hat, hält er sofort an und richtet sich durch feine Motoransteuerungen darauf aus. Ist er ausgerichtet, passt er seine Distanz zu Ziel mit Hilfe des Ultraschallsensors an. Sobald dies erfolgt ist, schießt er auf das Ziel und nimmt danach ein Bild auf. Ist das Ziel nicht mehr zu sehen, also umgeklappt, dreht der Roboter sich zum nächsten Ziel. Sollte das Ziel nicht von der Bildfläche verschwinden, also nicht getroffen worden sein, was bei der materialbedingten inkonstanz der Schüsse passieren kann, wird er solange schießen, bis das Ziel getroffen wurde.

C. Wahl des Antriebssystems

Die Wahl des Antriebssystems fiel sehr schnell auf einen auf ketten basierten Antrieb, da er es dem Roboter ermöglicht, auch unebenes Terrain mit Leichtigkeit zu überwinden und sich auf der Stelle zu drehen. Zusätzlich besteht bei einem eher kleineren Fahrzeug wie diesem die Möglichkeit Allrad-Kettenantriebe, also Kettenantriebe bei denen alle Räder Antriebsräder sind und keine Laufräder vorkommen, zu verwenden, da die auftretende Reibung eher zu vernachlässigen ist. Dies verleiht dem Roboter die Möglichkeit sich auch mit nicht montierten oder während des Betriebs abgefallenen Rädern problemlos fortzubewegen. Eine Seite des Allrad-Getriebes ist in Abbildung 2 dargestellt.

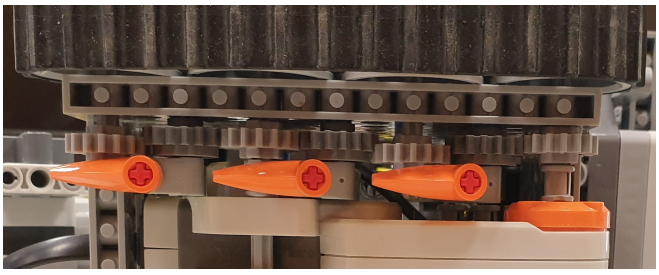


Abbildung 2. Ansicht einer Seite des Allrad-Ketten-Getriebes

D. Funktion der Webcam

Die Farberkennung mit der Webcam funktioniert so, dass die Webcam ein Bild aufnimmt und nur die Rotwerte des RBG-Bildes als Matrix ausgibt. Dabei entsteht eine Matrix vom Format der Auflösung der Webcam, welche aus den Rotwerten von 0 bis 255 besteht. Dies ist beispielhaft in Abbildung 3 dargestellt. Das Bild wird nun erst gefiltert um unnötiges Rauschen zu vermeiden. Dabei erhält jeder Pixel im

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 255 & 255 & 255 & 255 & 0 \\ 0 & 255 & 255 & 255 & 255 & 0 \\ 0 & 255 & 255 & 255 & 255 & 0 \\ 0 & 255 & 255 & 255 & 255 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Abbildung 3. Beispielhafte Darstellung eines perfekt roten Quadrates, aufgenommen in 6×6-Auflösung

gefilterten Bild den durchschnittlichen Wert der neun Pixel im 3×3-Bereich um ihn herum und sich selbst im ursprünglichen Bild. Daraufhin wird nach Farbwerten gefiltert. Dabei wird jeder Wert, der das 0.18-fache der maximalen Licht-/Farbstärke überschreitet zu einer Eins, und die, die das nicht tun zu einer Null. Dies ist in Abbildung 4 veranschaulicht, außerdem ist der Unterschied zwischen Abbildung 3 und Abbildung 4 zu betrachten. Vor dem eigentlichen Kenntlichmachen der

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Abbildung 4. Umwandlung auf Nullen und Einsen

roten Objekte werden noch Objekte, welche aus weniger als 300 Pixeln bestehen entfernt, damit vom Computer nicht kleinste rote Objekte fälschlicherweise als Ziele erkannt werden. Danach kann mit dem Markieren begonnen werden. Dafür wird um das rote Objekt eine sog. „Bounding-Box“ erzeugt, also einfach ein Kasten um das Objekt. Im Zentrum dieser „Bounding-Box“ wird daraufhin ein sog. „Centroid“ markiert, ein Objekt, welches in der Mitte des roten Objektes liegt. Dies ist in Abbildung 5 veranschaulicht. Die Koordinaten des „Centroids“ in Kamerabild verwendet der Roboter daraufhin, um sich auf das Ziel auszurichten.

Abbildung 5. Eine „Bounding-Box“ und ein „Centroid“ um ein rotes Objekt

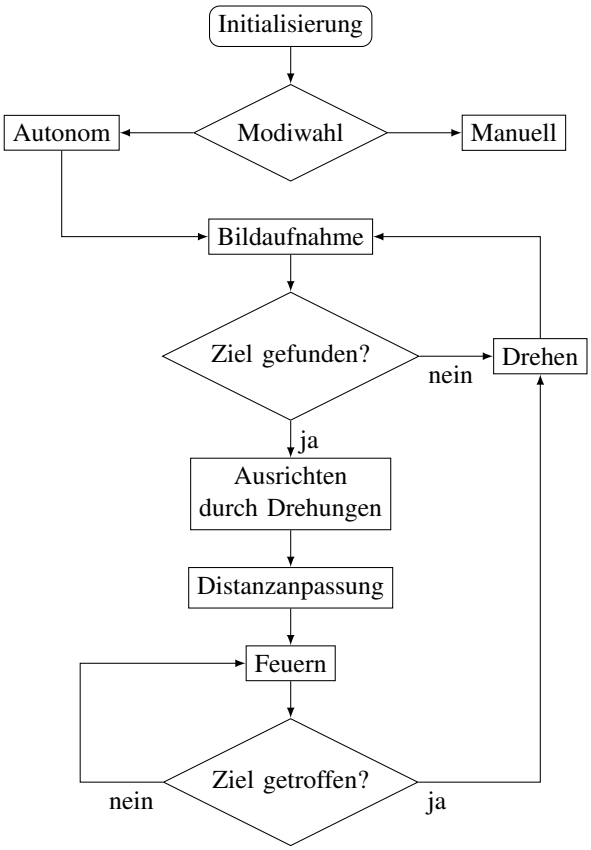


Abbildung 6. Programmablaufplan des Roboterprogrammes

E. Die Ballschussanlage

Die Ballschussanlage stellte eine größere Herausforderung dar als eigentlich gedacht. Beim Herausdrücken der Bälle aus der Abschussvorrichtung musste nämlich mehr Kraft aufgewandt werden, als zuerst angenommen. Sobald die Vorrichtung konstruiert war, stellt sich ein zweites Problem dar: Durch die große Kraft, die durch den Motor aufgewandt werden musste, verbog sich das Plastikgestell so sehr, dass kein Abschuss zu Stande kam. Dies wurde mit Gummibändern behoben, welche so gespannt wurden, dass sie die Biegungen so weit einschränkten, dass Abschüsse möglich waren. Eine Seitenansicht der fertigen Ballschussanlage ist in Abbildung 7 zu sehen.

IV. Problemstellungen

A. Limitationen des NXT

Der NXT, also das zentrale Steuer- und Empfangsmodul für die Motoren und Sensoren bietet eine Limitationen. Dazu gehören die begrenzte Anzahl an Sensor-Eingängen, nämlich vier Stück. Die Anzahl der Motor-Ausgänge ist mit drei Stück noch stärker begrenzt. Dies schränkte einige Ideen, die entwickelt wurden stark ein. Zusätzlich streikt der NXT dann, wenn er zu viele Befehle zu schnell bekommt, was beim wiederholten Abfragen von einigen Sensoren sehr schnell passiert.

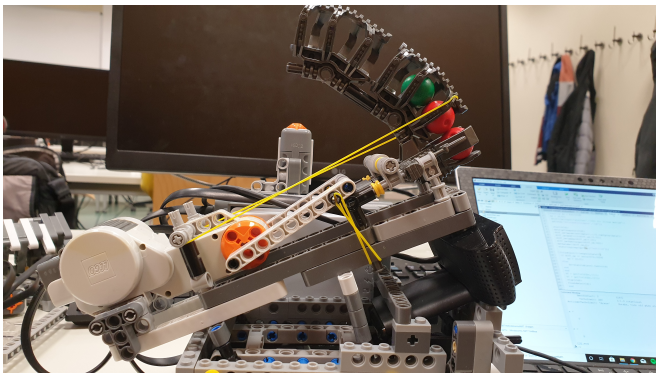


Abbildung 7. Seitenansicht der Ballschussanlage

B. Ungenauigkeit der Sensoren

Die Sensoren lassen bei manchen Aspekten der Genauigkeit etwas zu wünschen übrig. Zum Beispiel muss der Ultraschall-Sensor genau senkrecht positioniert werden um die Entfernung zuverlässig und akkurat auslesen zu können. Sobald eine Rundung oder eine Ecke anvisiert wird, verliert Entfernungsmessung an Zuverlässigkeit.

V. Andere Versionen und verworfene Ideen

A. Position der Kamera

In älteren Versionen des Roboters fand sich die Webcam noch neben der Ballschussmaschine. Dadurch entstand jedoch eine Verschiebung der Schusslinie, da sich der Roboter mit Hilfe der Kamera ausrichtet, jedoch, wenn die Kamera zentriert ist, immer an Ziel vorbei schoss. Deshalb befindet sich die Kamera in der finalen Version direkt unter der Kamera. So richtet der Roboter sich perfekt aus.

B. Höhenabhänige Distanz

Außerdem war eine Funktion geplant, bei welcher der Roboter aus der Entfernung zu Ziel und dem Winkel, in dem es sich in der Kamera befindet, die Höhe des Ziels berechnet, und so entsprechend seine Distanz dynamisch anpasst. Die Unzuverlässigkeit des Ultraschallsensors zwang uns jedoch dazu, diese Idee zu verwerfen.

C. Lenkung durch Zu- und Abkopplung der Ketten

Zusätzlich stand die Idee im Raum, die Lenkung des Roboters so zu realisieren, dass mit einem kupplungsähnlichen LEGO-Technik-Teil eine Seite der Ketten abgeschaltet werden könnte. So hätte man zum Treiben der Ketten nur noch einen Motor benötigt. Allerdings hätten dann für die Schaltung der Kupplung ein Motor verwendet werden müssen und Drehmoment der Ketten wäre verloren gegangen, da sie nur noch mit einem Motor betrieben werden würden. Deshalb fiel Entscheidung auf die simple Konstruktion mit einem Motor pro Seite. Außerdem kann so beim Lenken eine Seite vorwärts und die andere rückwärts drehen, anstatt eine Seite im Leerlauf zu haben. Dies verkleinert den Wendekreis noch weiter, nämlich auf die Länge des Fahrzeugs.

VI. Ergebnisdiskussion

Am Ende dieses Projektes steht ein Roboter, dargestellt in Abbildung 1, welcher zuverlässig rot markierte Objekte anvisiert und trifft. Auch das Bewegen im Raum funktioniert bei richtigen Bedingungen zuverlässig und gut. Diese richtigen Bedingungen sind bei der Bewegung im Raum relativ weit gefasst, wobei ein komplexerer Aufbau, wie zum Beispiel eine Lücke in der aufgebauten Wand an falscher Stelle ausreicht, um den Ultraschallsensor in die Irre zu führen. Die Funktion des Anvisierens ist allerdings einfacher zu stören. Trägt jemand im Raum ein rotes T-shirt wird derjenige anstatt des eigentlichen Ziels vom Roboter beschossen.

VII. Zusammenfassung und Fazit

Wir haben also einen Roboter entworfen, welcher zuverlässig markierte Objekte im Raum findet, ansteuert und beschießt. Die Realisierung dessen mit einer Webcam lässt Platz für relativ einfache und schnell Änderungen. So könnte zum Beispiel mit einer einfachen Änderung im Code die gesuchte Farbe geändert, oder mit der Einbettung von Computer-Vision-Bibliotheken komplexe Bilder erkannt werden, um die Nützlichkeit des Roboters in immer neuen Anwendungsgebieten zu gewährleisten. Auch könnten bei neueren Versionen des LEGO-Mindstorms-Bausatzes neue Funktionen des Roboters möglich werden. Bei neueren Sensoren könnte auf komplett neue Größen der realen Welt reagiert werden. Ebenfalls könnten bei hinzugefügten zusätzlichen Motor-Ausgängen weitere Funktionen hinzukommen, wie zum Beispiel das Heben und Senken der Abschussvorrichtung, oder ein Arm, welcher die getroffenen Ziele selbst wieder aufstellt. Der Vielfalt an möglichen zukünftigen Entwicklungsschritten und Versionen ist also nahezu keine Grenze gesetzt.

Literaturverzeichnis

[1] Bastgen, Peter, http://www.bastgen.de/schule/Projektkurs/robotik%209/NXT%20User%20Guide/assets/languages/german/print_all/content/9797_LME_UserGuide.pdf, 05.03.2020

[2] Mathworks, https://de.mathworks.com/, 05.03.2020

[3] RWTH Aachen University, Lehrstuhl für Bildverarbeitung, https://www.mindstorms.rwth-aachen.de/trac, 05.03.2020

LEGO Mindstorms Ballschussmaschine

Jonah Adrian Walther, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im folgenden Paper geht es um die Planung, Konstruktion und Programmierung eines autonomen Ballschussroboters der mittels Farb- und Distanzerkennung auf farbige Ziele schießen kann. In allen Bereichen sind einige Probleme aufgetreten die ebenfalls erläuter werden und zum Großteil behoben wurden, weiterhin werden die Anwendungsbereiche aufgezeigt.**

***Schlagwörter*—Ballschussmaschine, Distanzerkennung, Farberkennung, LEGO Mindstorm, Otto-von-Guericke-Universität, Projektseminar, Roboter**

I. Einleitung

WIE auch schon in vergangenen Jahren ist auch im Jahre 2020 wieder das LEGO Mindstorms Projektseminar der Otto-von-Guericke-Universität abgehalten worden, um den Studenten einen Einblick in die Welt der Robotik zu verschaffen. In dem zweiwöchigem Seminar sind viele Studenten in Gruppen mit der Programmiersprache MATLAB vertraut gemacht worden, um dann mit dem LEGO Mindstorms Set einen Roboter zu erschaffen, der eine Aufgabe bewältigen kann. Heutzutage gehören Roboter auf unserer Welt fast überall zum Leben dazu. Sie helfen uns alltägliche Probleme zu überstehen, helfen uns bei der Arbeit und fordern uns heraus unsere Leistungen zu verbessern. Roboter sind schneller, präziser und können Orte erreichen, die für den Menschen unerreichbar scheinen. Die Entwicklung ist noch lange nicht am Ende, vor allem in der Industrie ist der Roboter gern gesehen, da jeder Schritt genaustens programmiert werden kann, er schwere Objekte mit Leichtigkeit bewegen lässt und immer auf den Punkt genau arbeitet, nimmt er hier eine der größten Rollen ein. Aber auch im Sport hat er sich hervorgetan, beim Trainieren von Muskeln und zum Kraftaufbau wird er gerne eingesetzt. Um die Koordination, Schnelligkeit und Reaktionsgeschwindigkeit zu verbessern nutzen viele Sportler im Rückschlagsport (z. B. Tennis) sogenannte ”Ballschussanlagen”, das sind Roboter die verschiedenste Arten von Bällen abschießen können. Sie simulieren dem Sportler einen Gegenspieler, wenn keiner vorhanden ist. Im militärischen Bereich haben sich Drohnen weit nach vorn gearbeitet, das sind führerlose Fahrzeuge, welche zu Land, zu Wasser oder in der Luft agieren können, um zum Beispiel ein Gelände zu Kartagoriesieren. Eine Kamera zeichnet Bilder der Umgebung auf und eine künstliche Intelligenz erarbeitet daraus eine Karte, die für den Menschen anschaulich ist. Unser Ballschussroboter verbindet den militärisch und sportlich genannten Bereich und bietet einem ein autonomes Fahrzeug welches mittels einer Kamera die Umgebung erfasst und Bälle auf Ziele schießt

DOI: 10.24352/UB.OVGU-2020-034

II. Vorbetrachtungen

Um die Funktionen des Roboters besser erklären zu können, müssen zuerst die Funktionen des Lego Mindstorm-NXT-2.0 dem Kernelement des Systems erläutert werden, sowie die extra implementierte Webcam.

A. NXT

Der NXT-Legostein ist eine programmierbare Steuereinheit die 4 Eingänge für Sensoren und 3 Ausgänge für Motoren unterstützt. Bei unserem Roboter wurde der Ultraschallsensor und der Tastsensor verbaut. Der Tastsensor befindet sich in Abbildung 1 ganz links und funktioniert nach dem gleichen Prinzip wie ein herkömmlicher Taster, solange er gedrückt wird, gibt er ein Signal aus.

Abbildung 1. NXT 2.0 https://jaxenter.de/lego-mindstorms-nxt-2662

B. Ultraschallsensor

Der Ultraschallsensor des LEGO-Mindstroms-2.0-Set funktioniert nach einem Echo-Prinzip. Wie in Abbildung 2 zu sehen, wird von einem Sender ein Schallimpuls ausgesendet der an Objekten reflektiert wird und vom Empfänger ausgelesen werden kann. Die Zeit welche zwischen dem Senden und Empfangen vergeht, dient dabei als Referenz für die Distanzberechnung. Der Ultraschallsensor des LEGO-Mindstroms-2.0-Sets kann Objekte die bis zu $2{,}5\,\mathrm{m}$ entfernt sind bestimmen, die Distanz gibt dieser dabei in cm aus.

C. Webcam

Die Implementierte Webcam ist eine Logitech C920 HD Webcam mit einer Bildauflösung von 1920*1080 Pixeln, welche über einen USB Port angeschlossen werden kann. Zusätzlich ist die Linse im Winkel verstellbar.

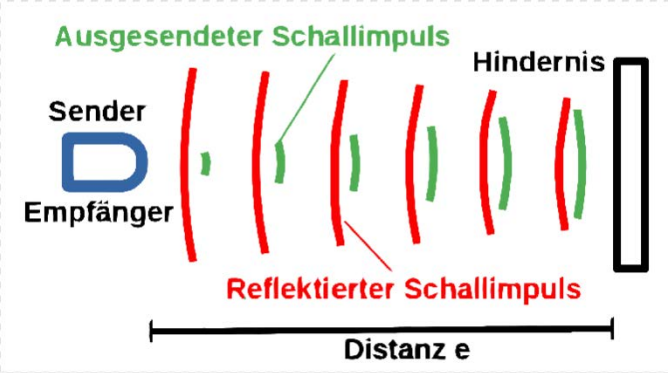


Abbildung 2. Funktion des Ultraschallsensor https://www.technik-unterrichten.de/Robotik/Ultraschallsensor/Ultraschallsensor.php

Abbildung 3. Logitech Webcam C920 HD https://www.amazon.de/Logitech-C920-HD-Pro-Webcam-Videogespräche-Videoaufnahmen-Full-HD-Stereo-Mikrofonen/dp/B006A2Q81M

III. Hauptteil

A. Mechanische Konstruktion

Für die Konstruktion standen den Gruppen mehrere Lego-Mindstorms-Baukastensets zur Verfügung, außerdem standen mehrere Sensoren zur freien Verfügung. Mit dem Baukasten wurde zuerst eine stabile Grundkonstruktion errichtet, welche das Gewicht des NXT-Legosteins standhält. An diese Grundkonstruktion wurde ein Antrieb an der Unterseite angebracht, wofür zwei Motoren verwendet wurden. Die Motoren treiben jeweils eine Zahnradreihe an, die für einen Kettenantrieb verwendet werden. Oberhalb der Grundkonstruktion wurde der NXT-Legostein befestigt und eine Abschussvorrichtung für Legobälle angebracht, die Abschussvorrichtung wurde zusätzlich mit einem Motor versehen. An der Vorderseite ist der Ultraschallsensor angebracht worden und an der rechten

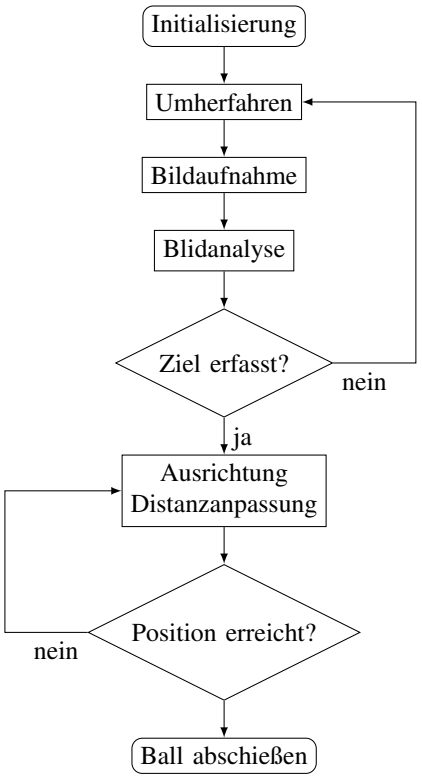


Abbildung 4. Programmablaufplan zur Erklärung des MatLab Program

Seite der Tastsensor. Oberhalb des Ultraschallsensors wurde eine kleine Halterung zum Fixieren der Webcam positioniert, sowie die Webcam selbst.

B. Programablauf

Anhand des Programmablaufplans ist eine grobe Angabe des Programms in MATLAB und dient zur Visualisierung für Personen, die sich mit der Programmiersprache nicht auseinandergesetzt haben.

C. Bildaufnahme und Analyse

Wie in Abbildung 4 gezeigt wird ein Bild von der Webcam aufgenommen und dann analysiert. Das von der Webcam aufgenommene Bild wird in MATLAB in einer Matrix gespeichert, welche die Auflösung der Kamera als Größe fest legt (1920*1080 also 2 073 600 stellen in der Matrix). Jeder aufgenommene Pixel, wird dabei in einem Array gespeichert, welches dann im Matrix-Element implementiert wird. Der Pixel im Array ist als RGB-Code gespeichert, der RGB-Code steht für die Grundfarben Rot, Grün und Blau im RGB-Farbraum. Der RGB-Farbraum lässt sich wie einen Würfel beschreiben in dem alle möglichen Farben in einen drei dimensionalen kartesischen Koordinatensystem von 0 bis 255 aufgelistet sind. Die Koordinaten bestimmen dabei dem Rot-, Grün- und Blauanteile. Die Matrix in MATLAB wird im nächsten schritt untersucht, jedes Element der gespeicherten Arrays wird auf bestimmte Werte geprüft die in dem Farbwürfel im roten Bereich liegen. Der darauf folgende Schritt ändert jedes Array, welche den roten Bereich nicht erreichen zu einem Array voller

Nullen und alle Arrays die den roten Bereich erreichen zu einem Array voller Einsen. Alle Arrays werden danach in den Durchschnittswert ihres Inhaltes umgeschrieben und in einer neuen Matrix gespeichert. Diese neue Matrix wird im nächsten schritt auf Nullen und Einsen untersucht und umrahmt alle Gruppen von Einsen mit einem Kasten. Innerhalb des Kastens kann ein Zentrierungspunkt gesetzt werden, um den Mittelpunkt des roten Objektes zu definieren, anhand des Mittelpunktes richtet sich der Roboter dann aus Abbildung 5

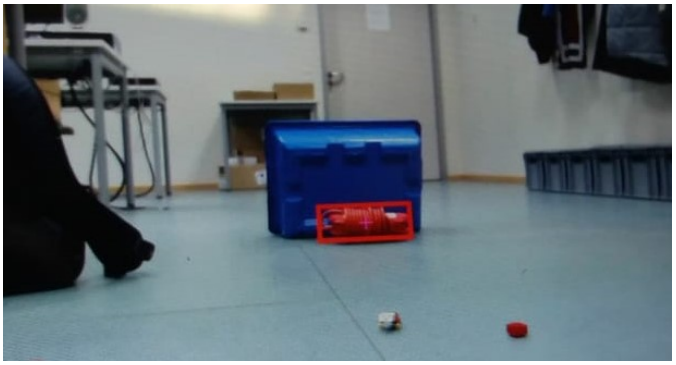


Abbildung 5. Bildhafte Umsetztung der Matrixanalyse

D. Umherfahren

Außerdem wird in Abbildung 4 gezeigt, dass sich das Programm in einer Schleife befindet, welche so lange abläuft wie kein Ziel erkannt wurde. Dabei fährt der Roboter gerade aus und untersucht mit dem Ultraschallsensor den Abstand zu einem Objekt vor ihm, zum Beispiel einer Wand. Währenddessen erzeugt er jede Sekunde ein Bild mit der Webcam zum Analysieren und solange auf diesem Bild kein Zielobjekt zu erkennen ist, fährt er weiter gerade aus. Sollte er kein ziel erkennen und sich einem Hindernis auf unter $20\,\mathrm{cm}$ Distanz befinden, stoppt der Roboter und dreht sich um ca. $90°$ zur Seite, danach macht er mit seinem Ablauf weiter.

E. Ausrichtung und Distanzanpassung

Zusätzlich zeigt die Abbildung 4 eine zweite Schleife welche sich um die Position des Roboters zu seinem Ziel kümmert. Dabei nutzt er den Mittelpunkt des roten Objektes in seiner Bild Matrix, um die horizontale Ausrichtung zu bestimmen. Da sich das Kamerabild direkt proportional zur Matrixgröße verhält, dreht sich der Roboter mittels Antreiben des Kettenantriebes in unterschiedliche Richtungen auf der Stelle und das Objekt in der Mitte des Bildes zu positionieren. Danach ermittelt der Roboter mittels dem Ultraschallsensor den Abstand zu seinem Ziel und passt die Entfernung auf seine Schussreichweite an.

IV. Ergebnisdiskussion

Während des Baus und der Programmierung sind viele Probleme aufgetreten und es gibt auch einige die nicht gelöst werden konnten. Zuerst hat die mechanische Konstruktion große Fehler verursacht, dabei hat sich die Ballabschussvorrichtung durch den enormen Druck, der aufgewendet werden musste, um den Ball abzufeuern sehr instabil angefühlt und sich oft mal selbst auseinander genommen hat. Gelöst wurde die fehlerhafte Abschussvorrichtung dann mit Gummibändern als Fixierung, was den Großteil des Problems behoben hat. Leider entstand dabei sehr viel Stress aus mechanischer Sicht auf die Bauteile, was diese wohl nicht auf ewig überstehen werden. Das nächste Problem welches aufgetreten ist, war die Position des Roboters zu seinem Ziel. Da sich die Ausrichtung zur Mitte der Matrix bewegt, muss ein bestimmter Bereich festgelegt werden, da die Motoren nicht präzise genug arbeiten, dabei haben wir den Bereich vorerst zu klein gewählt, wodurch sich der Roboter immer um den Mittelpunkt rumgedreht hat und nie zur Ausrichtung der Distanz gekommt ist. Ein Problem welches nicht gelöst werden konnte ist, dass der Abschusswinkel konstant ist und somit die Objekte zum Abschießen auf derselben Höhe aufgestellt werden müssen da sich eine variable Höhe aus programmier und technischer Sicht aktuell nicht lösen ließ. Mit einem weiteren Motor wäre dies möglich gewesen allerdings ist keinen Platz mehr für einen weiteren Motor vorhanden, weshalb dieses Lösung leider weg fiel. In programmtechnischer Hinsicht war das Programm zu Beginn recht unordentlich, bestimmte Bereiche waren überflüssig und Fehler haben viele Probleme gemacht. Dieses Problem wurde durch Unterprogramme gelöst, welche für sich ordentlich und fehlerfrei sind und dann nur noch in einem Hauptprogramm zusammengeführt werden. Das letzte Problem, welches aufgetreten war, ist wenn mehrere Objekte auf dem Bild sind. Der Roboter hat dann das Objekt, welches am weitesten links ist fokussiert und sich ausgerichtet aber dann abgebrochen da sich ja noch eines auf der rechten Seite gefunden hat. Durch eine simple Programmanweisung wurde ein Objekt favorisiert und zuerst angesteuert.

A. Ergebnis

Am Ende des Projektes ist ein Roboter vorgestellt worden, welcher sich von allein in einem begrenzten Raum fortbewegt und auf rote Ziele feuern kann.

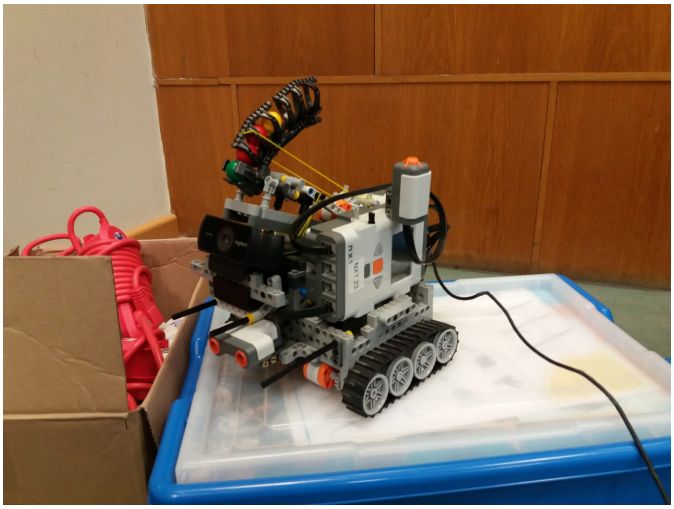


Abbildung 6. Finale Konstruktion

V. Zusammenfassung und Fazit

Zusammengefasst wurde ein funktionstüchtiger Lego-Roboter gebaut, welcher die Aufgabe farbige Ziele abzuschießen, erfolgreich durchführen kann. Dabei kann er allein ein begrenzten Bereich erkunden und von allein die Ziele ausfindig machen. Sich nach den Zielen zu orientieren und zu beschießen erfolgt über die Webcam und die Abschussvorrichtung. In Zukunft können noch weitere Verbesserungen vorgenommen werden, welche zum einen die Präzision verbessern könnten. Mittels eines Algorithmus soll der begrenzte Bereich nach einem Muster abgefahren werde und nicht mehr dem Zufall überlassen sein. Außerdem wäre eine Umstrukturierung der Abschussvorrichtung notwendig, um die Bauteile zu entlasten und eine allgemeine Halterung die jede Art von Webcam halten und nicht ausschließlich die aus Abbildung 3.

Anhang

Hier ist ein Video welches zum Verständis der Bildanalyse: https://www.youtube.com/watch?v=W9gsIOgFFo0

VI. Literaturverzeichnis

[1] Wikipedia, RGB-Fabraum, https://de.wikipedia.org/wiki/RGB-Farbraum

Fußballroboter – Ein LEGOMindstorms Projekt

Fabio Näther, Studiengang
Otto-von-Guericke-Universität Magdeburg

Mithilfe der LEGO Mindstorms Baureihe sollte während des Praktikums ein Roboter oder Anlage gebaut werden, welche in der Lage sein sollte, eine selbformulierte Aufgabe eigenständig zu bewältigen. Verfügbar waren dafür ein programierbarer Legostein NXT Sowie verschiedene Sensoren und Motoren. Ebenfalls gehörten LEGO Technik und normale LEGO-Bausteine zu den verbaubaren Teilen. Limitierende Grenzen waren dabei durch die maximale Anzahl an Motoren, der NXT verfügt über 3 Motoranschlüsse, und die maximale Anzahl an Sensoren, nämlich 4 Sensoranschlüsse gegeben. Die Programme wurden dabei über die Programiersoftware MATLAB erstellt und anschließend über USB-Kabel oder Bluetooth übertragen.

I. Eigene Aufgabenstellung

Ziel des Roboters war es einen Ball innerhalb eines begrenzten Spielfeldes zu erkennen und eigenständig in ein Tor zu bewegen. Um diese Aufgabe lösen zu können, musste der Roboter in der Lage sein, seine eigene Position im Spielfeld als auch die Position des Balls zu erkennen. Er musste außerdem den Ball sicher aufnehmen und wieder ablegen können. Zudem musste er in der Lage sein, den Ball in das Tor zu bewegen. Um diese Aufgaben lösen zu können, musste der Roboter einen Antrieb besitzen, durch den er das Spielfeld befahren konnte. Dafür war es notwendig, eine Steuerung für eben jenen Antrieb zu programmieren. Für die Aufnahme des Balls war es erforderlich, dass der Roboter sich zum Ball bewegt und diesen greift. Um das Tor zu erzielen, musste der Roboter in der Lage sein, sich zum Tor auszurichten und den Ball in der richtigen Position abzulegen.

II. Bau des Roboters

A. Bau des fahrbaren Chassi

Um die Aufgabe des Antriebes zu bewältigen, wurde entschieden, ein Kettenfahrzeug zu bauen und dafür 2 der 3 möglichen Motoren zu benutzen. Nach Tests mit Ketten- und Radfahrzeugen stellte sich heraus, dass eine genaue Drehung des Fahrzeuges nur möglich wäre, wenn das Radfahrzeug über eine Lenkung verfügen würde. Das einfache Abstellen eines Motors bei einem Radfahrzeug führte zu großen und ungenauen Drehungen. Da dies jedoch ein Hauptbestandteil der Erfassung des Balls werden würde, war es nur mithilfe des dritten Motors möglich, eine solche Genauigkeit zu erzeugen. Da jedoch dieser Motor bereits für das Öffnen und Schließen des Greifers benötigt wurde, war ein Radfahrzeug keine Option. Aufgrund dieser Probleme wurde sich für ein Kettenfahrzeug entschieden, welches 2 Ketten besaß. Dabei wurde jeder dieser Ketten von einem eigenen der beiden für diese Aufgabe abgestellten Motoren bewegt. Diese wurden direkt durch eine Achse mit dem Laufrad der jeweiligen Kette verbunden. Beide Motoren wurden miteinander verbunden, um die nötige Stabilität zu erzeugen. Dies geschah durch verschieden längliche LEGO Steine und Stangen. Auf den beiden Motoren, welche die Ketten antrieben wurde der NXT befestigt und die beiden Motoren wurden durch Kabel an den Motoranschlüssen des NXT angeschlossen. Diese Anordnung ermöglichte es durch das Drehen einer Kette in Fahrtrichtung und eine in die Entgegengesetzte, eine perfekte Drehung auf der Stelle durchzuführen. Nach einigen Test mit dieser Konfiguration stellte sich heraus, dass die beiden Motoren für eine Drehung auf der Stelle nicht nur entgegengesetzte Richtungen sondern auch mit verschieden Geschwindigkeiten und Laufzeiten aktiviert werden mussten. Das Verhältnis dieser Faktoren wurde in einigen Test herausgefunden und stellte somit eine genaue Drehung sicher, welche für spätere Aufgaben unabdingbar war.

B. Bau des Greifers und der Ballerkennung

Um die Aufgabe des Greifen und Ablassen des Balls zu ermöglichen, war der Bau eines Greifers nötig. Dieser sollte in der Lage sein den Ball sicher zu erkennen und zu greifen. Außerdem war es nötig in anschließend wieder sicher abzulassen. Die Bewegung dieses Greifers übernahm der dritte Motor, welcher liegend über dem NXT befestigt wurde. Durch eine Achse wurde eine der in Fahrtrichtung linke Greiferarm direkt angetrieben. Der zweite Greiferarm wurde durch drei in Reihe liegende Zahnräder bewerkstelligt. Durch das Verwenden von einer ungeraden Anzahl an Zahnräder wurde es möglich die beiden Greiferarme mit nur einem Motor zu öffnen und zu schließen. Auch dies war entscheidend, da nur ein Motor zur Verfügung stand. Der Motor wurde über ein Kabel am dritten Motoranschluss am NXT angeschlossen. Der Ball sollte mit drei senkrecht nach unten gerichteten Farbsensoren erkannt werden. Ziel dabei war es, aus der Bewegung heraus den Ball zu erkennen und anzuhalten um den Ball zu greifen. Die drei Farbsensoren wurden gleichmäßig über der Spannweite des Greifers verteilt und an die ersten drei Sensorenanschlüsse am NXT durch Kabel angeschlossen. Eben jene begannen bei Fahrtbeginn den Boden unter ihnen zu scannen und die

DOI: 10.24352/UB.OVGU-2020-035

Farbe zu speichern. Dies taten sie alle 20 ms. Der Roboter bewegte sich bei diesem Teil des Programms innerhalb einer Schleife welche die Farbwerte der Sensoren prüfte. Sollte einer der Sensoren einen roten Wert aufnehmen

würde die Schleife beendet werden und der Roboter stoppen. Anschließend sollte der Teil des Programms ablaufen, welcher den Greifer sich schließen lies. Nach einigen Test wurde das Problem deutlich, dass der Stopp des Roboters verzögert sein würde und dadurch der Ball berührt werden würde. Dies führte dazu, dass der Ball in Bewegung gebracht wurde und nicht mehr vom sich schließenden Geifer gesichert werden konnte. Für die Lösung dieses Problems stoppte der Roboter beim Erkennen eines roten Objektes nicht sofort ab sondern bewegte sich in langsam Tempo weiter gerade aus. Dadurch wurde der sich vom Roboter entfernende Ball gesichert.

C. Erkennen des Roboters und des Balls

Um den Roboter und den Ball auf dem Spielfeld zu erkennen bot sich die Nutzung einer Webcam an. Diese wurde über dem Spielfeld platziert und fotografierte das Spielfeld aus einer Höhe von circa 1.70 m. Idee dabei war es das gesamte Spielfeld in kleine Quadrate aufzuteilen. Diese würden das Abmaß der Auflösung der Kamera haben. Das Programm sollte dabei jedes einzelne Quadrat nach dessen Farbzusammensetzung überprüfen. Dafür wurden die RGB Werte in drei, der Auflösung der Kamera entsprechende , Tabellen eingetragen. Da unser Ball rot war, hat er alle von ihm eingenommenen Felder nur mit einem rot Wert beschrieben. Mithilfe einer Gleichung wurden von der Kamera fehlerhaft, jedoch immer nur mit kleinen rot Werten, versehende Felder ignoriert. Hatte das Programm die Werte erhalten suchte es nach Werten, welche es berücksichtigen sollte und rechnete den Mittelpunkt dieser Felder aus. Der Wert dieses Feldes in der Tabelle, also sein Spalten- und Zeilenwert wurden dabei an zwei Variablen übergeben. Somit war die Position des Balls im Spielfeld erkannt. Um die Position des Roboters zu erkennen wurde dieser mit einem grünen und einem blauen Punkt versehen. Der grüne Punkt wurde dabei direkt über dem Drehpunkt des Fahrzeuges platziert(Erklärung folgt). Der blaue Punkt wurde in der Längsachse des Fahrzeuges in Fahrtrichtung vor dem grünen Punkt platziert. Vom selben Teil des Programms wurden jeweils zwei weiteren Variablen die Werte des grünen und des blauen Punktes übergeben.

D. Ausrichtung des Roboters zum Ball

Um nach der Erkennung des Balls und des Roboters diesen Auszurichten wurden die in Punkt C. gespeicherten Variablen verwendet. Idee war es durch das Berechnen des Winkel des Roboters und es Roboters zum Ball einen Winkel zu berechnen, welcher verwendet werden konnte um sich zum Ball zu drehen. Dafür wurde zuerst der Winkel des Roboters berechnet. Dies geschah mithilfe der atan2 Funktion und den gespeicherten Variablen. Da Programm rechnete einen Winkel zwischen dem grüne und dem blauen Punkt aus. Dieser lag im Wertebereich zwischen 180° und -180°. Anschließend wurde der Winkel zwischen dem grünen Punkt und dem Ball mit der gleichen Funktion berechnet. Die auszuführende Drehung des Roboters wurde anschließend durch die Subtraktion des Roboter Winkel vom Winkel zum Ball. Der resultierende Winkel war der zu drehende Winkel. Durch den Wertebereich von 180° bis -180° war es möglich eine zu drehende Richtung anzugeben. Mithilfe einer If-Else-Schleife entschied das Programm nach links oder nach recht zu drehen. War der Winkel größer als 0 drehte der Roboter sich nach links. War der Winkel negativ drehte er sich nach links. Der grüne Punkt wurde dabei exakt über dem Drehpunkt platziert um das berechnen des Winkels zu vereinfachen und die anschließende Drehung genauer zu gestalten. Dadurch wurde ein am Ball vorbei fahren, verhindert.

E. Ausrichtung des Roboters zum Abschlusspunkt

Um den Roboter in eine Position zu bringen, welche sich vor dem Tor befindet, wurde das in Punkt D. erläuterte Verfahren erneut angewandt. Der einzige Unterschied hierbei war, das ein neues Bild und Positionsvariablen erstellt wurden. Dies geschah wie in Punkt C. Beschrieben. Die Werte des Balls wurden dabei ersetzt durch einen festgelegten Abschlusspunkt('Elfmeterpunkt'). Nachdem der zu drehende Winkel berechnet war wurde die Fahrtstrecke bestimmt. Dafür wurde ein Vektor vom grünen Punkt zum festgelegten Punkt berechnet. Anschließend errechnete das Programm die Länge dieses Vektors und gab dem Roboter die Anweisung die zu fahrende Strecke zurückzulegen. Nach dem sich die Motoren dich notwendige Strecke gedreht hatten blieb der Roboter stehen.

F. Ausrichtung des Roboters zum Tor

Das gleiche in E. erläuterte Verfahren wurde erneut angewandt. Einziger Unterschied hierbei war, dass der Abschlusspunkt mit seinen Variablen durch einen Torpunkt ersetzt wurde. Auch bewegt sich der Roboter nach dem Ausrichten zum Punkt nicht auf ihn zu.

G. Torschuss

Der Roboter öffnet den Greifer und fährt eine gewisse Strecke rückwärts. Anschließend fährt er mit maximaler Geschwindigkeit ein wenig weiter als der Anlauf gewesen ist und trifft damit den Ball. Der Ball bewegt sich durch das Anstoßen durch den Roboter in Richtung des Tores.

III. Umsetzung des Projektes

Die im Bau des Roboters beschriebenen Punkte fanden bei der Umsetzung der Idee nacheinander statt. Dabei traten immer wieder Probleme auf, welche teilweise durch das anpassen einzelner Werte und erneutes Testen aber auch durch komplettes abändern von Abläufen, behoben wurden. Durch das Erstellen einzelner Unterprogramme war es immer wieder möglich einzelne Teile des Programms zu testen und zu verbessern. Nachdem die einzelnen Unterprogramme funktionierten und in einem einzelnen Programm zusammengefasst waren, liefen sie wie in folgendem PAP zu sehen, ab.

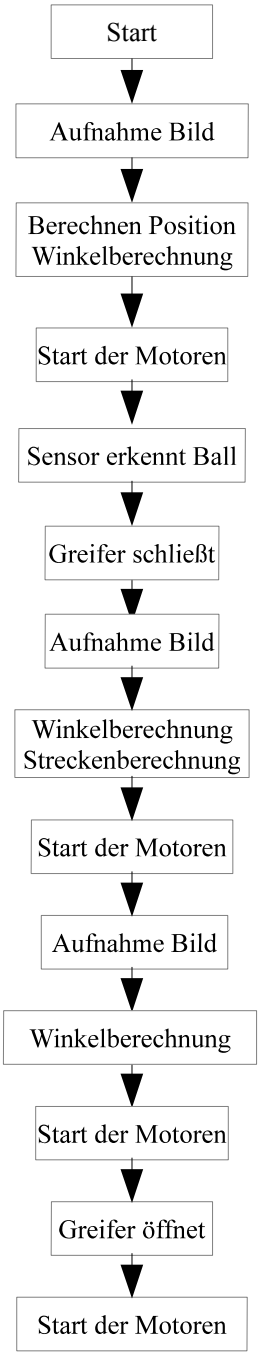


Abbildung 1: Programmablaufplan

Nach dem Start des Programms wird es zuerst über Bluetooth an den NXT übertragen. Nachdem dies abgeschlossen ist wird das erste Bild mit der Webcam geschossen. Nachdem dies geschehen ist setzt das Programm bei dem in Punkt C. geschilderten Ablauf ein. Dabei speichert es die entsprechenden Variablen und geht in den Punkt D. Über. Hat sich der Roboter ausgerichtet beginnt er sich mit halber Geschwindigkeit nach vorne zu bewegen. Und beginnt mit der in Punkt B. beschriebenen Suche nach dem Ball. Ist dies geschehen wird erneut ein Bild geschossen und der Roboter beginnt mit dem in Punkt E. beschriebenem Programm. Das Programm überschreibt dabei die noch vorhandenen alten Werte der Variablen. Es wird danach erneut ein Bild geschossen und der Punkt F. Wird ausgeführt. Anschließend folgt der finale Teil mit dem Punkt G..

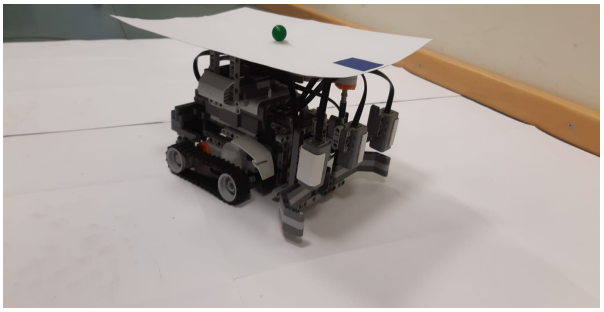


Abbildung 2: Aufbau Roboter mit Punkten

In Abbildung 2 ist der fertige Roboter mit dem grünen Punkt und dem blauen Punkt zusehen. Die angebrachte weiße Pappe soll dabei Farbige stellen auf einzelnen Bauteilen des Roboters verdecken und so die Positionsbestimmung genauer machen. Im vorderen Bereich des Roboters sind die beiden Greifarme und die drei Farbsensoren zu erkennen.

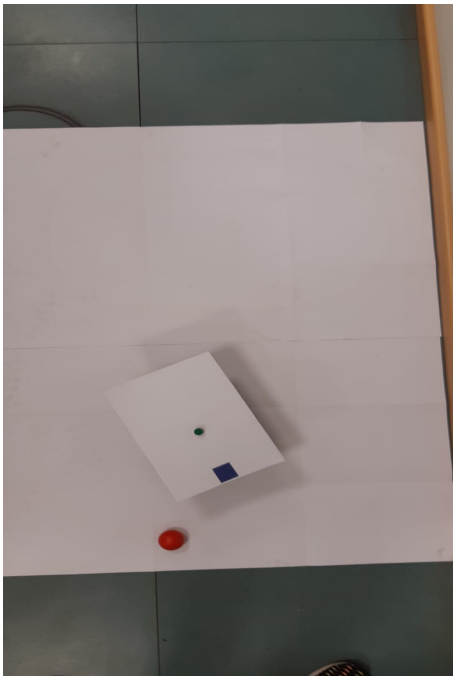


Abbildung 3: Sicht der Kamera

In der Abbildung 3 ist ein aufgenommenes Bild der Kamera zu sehen. Anhand eines solchen Bildes berechnet das Programm die Mittelpunkte des grüne und blauen Punktes und des Balls. Und übernimmt diese Werte als Variablen für die späterer Winkelberechnung.

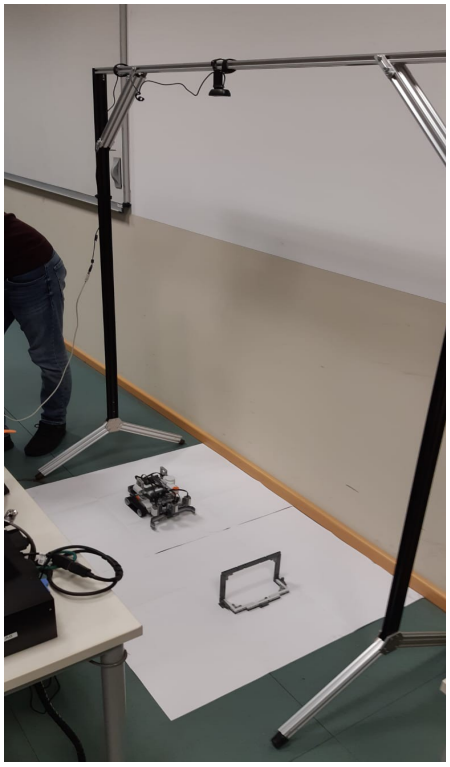


Abbildung 4: Gesamter Aufbau

In Abbildung 4 ist der gesamte Aufbau zu sehen. Die Kamera ist am zu sehenden Stativ befestigt, welches das gesamte Spielfeld überspannt. Sie ist über ein USB-Kabel mit dem Computer auf, welchem das Programm läuft, verbunden. Es ist außerdem der Roboter und das Tor zu sehen. Das Spielfeld ist hierbei in etwa durch die weißen Planen markiert.

IV. Auswertung des Projektes

Als Ergebnis des Projektes ist ein Roboter zu betrachte, welcher selbständig in der Lage ist seine Position und die eines Ball in einem Spielfeld zu erkennen. Sich zu diesem auszurichten und ihn aufzunehmen. Sich anschließend zu einem Tor auszurichten und dieses im Anschluss zu treffen. Zudem gehören zu diesem Projekt das Tor und die notwendige Kamera. Beim Bau dieses Roboters traten verschiedene Probleme auf. Die genaue Steuerung des Chassis erwies sich als anfängliches Problem. Zudem gestaltete sich das Greifen des Balls mit dem Greifer als Problem, welches zu großen teilen gelöst werden konnte. Die Winkelberechnung wurde durch die Verwendung der atan2-Funktion im Vergleich zur davor verwendeten Berechnung durch zwei Vektoren deutlich verbessert. Die genaue Ansteuerung des Abschlusspunkt gestaltet sich schwierig da eine genaue Längeneinchätzung mit den Motoren nur schwer möglich ist. Ein weiteres Problem tritt auf wenn der Ball nah am Rand des Spielfeld liegt. Durch den Ablauf des Programms dem Ball nach erkennen noch ein Stück hinterher zu fahren kann es passieren, dass beim folgenden Bild Teile des Roboters mit den Erkennungspunkten nicht mehr im Spielfeld stehen.

V. Zusammenfassung und Fazit

Innerhalb des Praktikums entstand ein Roboter, welcher in der Lage ist ein Tor eigenständig zu erzielen. Dafür wurden grundlegende Programmierabläufe in MATLAB und die Funktion von LEGO Mindstorms erlernt. Außerdem wurden mehrere Vorträge gehalten und diese Facharbeit verfasst. Für das Projekt fehlend sind noch genauere Verbesserung des Roboters um die Trefferwahrscheinlichkeit zu erhöhen. Zudem wäre es vorstellbar Hindernisse zu erkennen und zu um fahren. Auch wäre es möglich, sich den Weg des Roboters in einem Koordinatensystem zur besseren Auswertung auszugeben.

VI. Quellen

Für dieses Projekt benutzten wird die Matlab Hilfefunktion als auch den sehr nützlichen Rat unserer Betreuer.

Fußballroboter mit Lego-Mindstorms

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

***Zusammenfassung*—In diesem Paper geht es um die Entwicklung und Konstruktion eines Fußballroboters mit Lego-Mindstorms und MATLAB. Dieser soll beliebig mit einem Ball in einem Spielfeld platziert werden und dann selbstständig den Ball suchen und ein Tor erzielen. Dazu muss das Bild einer Webcam, die das Spielfeld erfasst, ausgewertet werden. Die Orientierung mit Hilfe von Punkten auf Roboter und Spielfeld und auch der Aufbau des Roboters wird dargestellt, wobei insbesondere der Greifmechanismus betrachtet wird.**

***Schlagwörter*—Lego-Mindstorms, Fußballroboter, Projektseminar, Bildauswertung, Vektoren**

I. Einleitung

DER Ausbau der Automatisierung schreitet immer weiter voran. In der Industrie werden Roboter in fast jedem Zweig genutzt. Sie bauen Autos oder transportieren Gegenstände von A nach B. Aber auch in privaten Haushalten schreitet die Automatisierung immer weiter voran. Staubsaugerroboter saugen das Haus selbstständig und finden nach Abschluss der Arbeit selbstständig ihre Ladestation. Auch Rasenmähroboter finden ihre Ladestation selbstständig. Sie bewegen sich in ihren vom Besitzer gesteckten Grenzen, mähen dort und entleeren sich auch ohne Einfluss eines Menschen. Diese Entwicklung wird sich voraussichtlich fortsetzen und ausweiten. Im Rahmen des Projektseminars wurde mit Hilfe von Lego-Mindstorms ein Roboter entwickelt, der mit einer Webcam das Spielfeld erkennt, analysiert und anschließend berechnet, welche Bewegungen gemacht werdnen müssen, um schlussendlich ein Tor zu erzielen.

II. Vorbetrachtungen

Um einen Algorithmus zur Orientierung zu entwickeln, werden Ansätze eines bereits vorhandenen Programms zur Farberkennung verwendet.

A. Fußball

Die englische Fußballlegende Gary Lineker sagte einst: ”Football is a simple game; 22 men chase a ball for 90 minutes and at the end, the Germans always win. „Fußball ist ein einfaches Spiel: 22 Männer jagen 90 Minuten lang einem Ball nach, und am Ende gewinnen immer die Deutschen.“ In diesem Projektseminar geht es weder darum mit 22 Spielern zu spielen, noch gegeneinander zu spielen. Aber einfach ist das Spiel dennoch. Das Ziel ist weiterhin das Schießen von Toren. Allerdings wird hier nur mit einem Roboter auf ein Tor gespielt.

DOI: 10.24352/UB.OVGU-2020-036

B. Bewegung auf dem Spielfeld

Da sowohl Ball als auch Roboter beliebig platzierbar sein soll, wird der Ball nicht immer gerade vor dem Roboter liegen. Der Roboter muss sich also zum Ball orientieren können. Man könnte das ganze Spielfeld systematisch abfahren, um den Ball zu finden. Allerdings kann das unter Umständen sehr lange dauern. Deshalb soll mit Orientierungspunkten gearbeitet werden. Einer der Punkte, die sich auf dem Roboter befinden, sollte dabei der Hauptorientierungspunkt sein. Von diesem sollen Vektoren zum zweiten Roboterpunkt und zum Ball berechnet werden. Zwischen diesen Vektoren soll der Winkel berechnet werden. Mit diesem Winkel soll sich der Roboter orientieren und sich dann zum Ball ausrichten. Dieser Weg ist wesentlich effizienter als das systematische Abfahren des Spielfeldes. Da der Roboter auch nicht in jedem Fall gerade vor dem Tor stehen wird, müssen zwei weitere Punkte mit den Koordinaten festgelegt werden. Darüber können wieder Vektoren bestimmt werden, die durch Winkelberechnung zur Orientierung genutzt werden.

C. Farberkennung

Zur Farberkennung muss zuerst ein Bild mit Hilfe einer Webcam aufgenommen werden. Die Kamera setzt das Bild im RGB-Farbraum zusammen. Die jeweiligen Werte des Rot-, Grün- oder Blauanteils müssen dann extrahiert werden. Aus jedem Farbanteil muss nun der Mittelpunkt bestimmt werden. Wenn man nun ein relativ farbneutrales Spielfeld nutzt, kann man sich durch einen farbigen Ball und Fabrmarkierungen auf dem Roboter relativ einfach im Bild beziehungsweise auf dem Spielfeld orientieren und bewegen.

III. Realisierung

A. Konstruktion

Als erstes wurden zwei Lego-Mindstorms-Motoren zur Steuerung des Roboters verbaut. Diese sollten jeweils eine Antriebskette antreiben, um eine bestmögliche Beweglichkeit sicherzustellen. So kann sich der Roboter auch auf der Stelle drehen. Der Greifer wird über einen weiteren Motor gesteuert. Dieser befindet sich an der Oberseite des Roboters und ist mit Achsen und Zahnrädern mit dem Greifer verbunden. Der Greifer befindet sich an der Vorderseite des Roboters und wurde so konstruiert, dass der Spielball in die Greiferarme passt ohne zu fest zu sitzen. Zunächst gab es nur einen Farbsensor zur Ballerkennung in der Mitte des Greifers, wie in Abbildung 1 gezeigt. Wenn der Ball aber nun nicht mittig in die Greiferarme gelangt, erkennt der Sensor keinen Ball und der Roboter sucht den Ball weiter, obwohl er ihn schon hat. Deshalb wurden zwei weitere Farbsensoren, jeweils einer auf jeder Seite des Greifers,

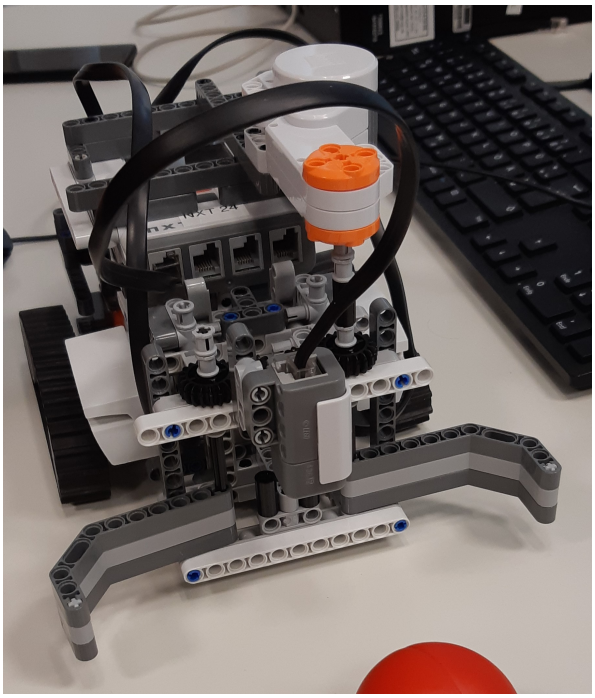


Abbildung 1. Greifer mit einem Farbsensor

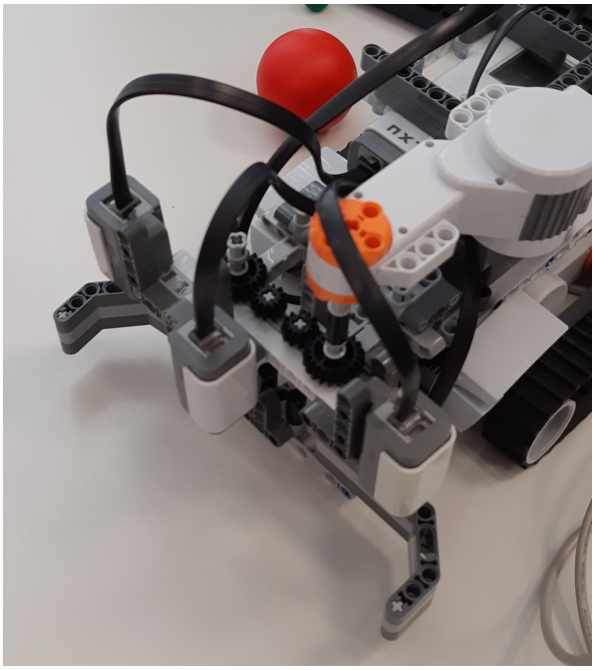


Abbildung 2. Verbesserter Greifer mit 3 Sensoren

angebaut, wie in Abbildung 2 zu sehen ist. So nimmt der Roboter den Ball auch wahr, wenn dieser nicht genau mittig im Greifer liegt. Zuletzt wurde eine weiße Pappabdeckung auf den Roboter geklebt, um jegliche Farbstörungen für die Webcam zu verhindern und gleichzeitig zwei Orientierungspunkte zu markieren.

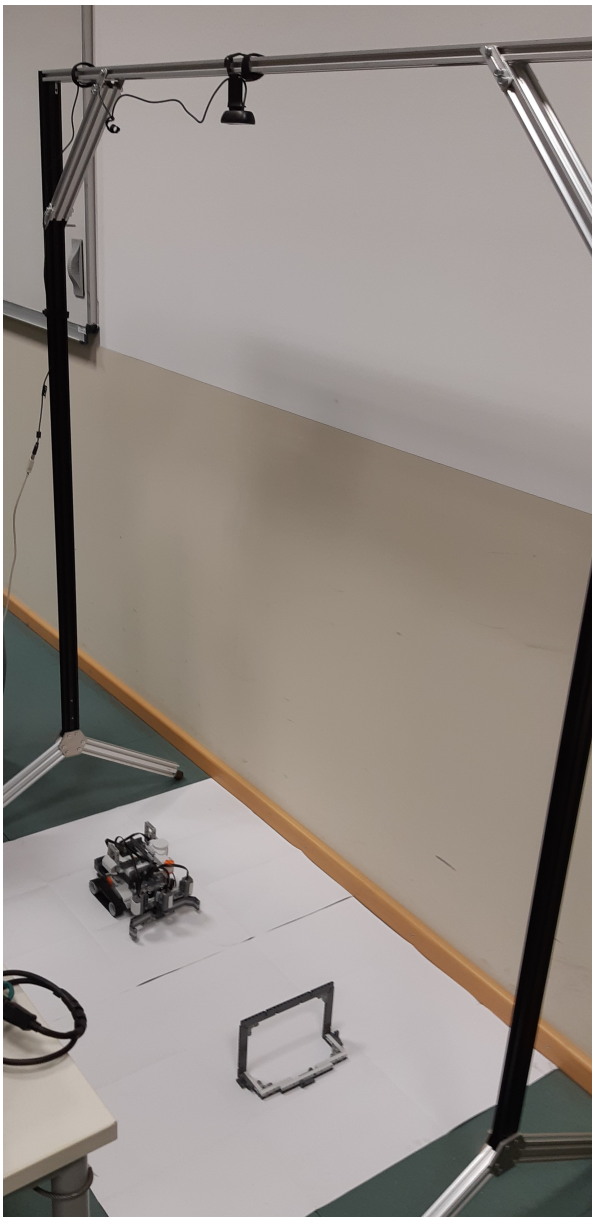


Abbildung 3. Stativ mit Webcam über dem Spielfeld

B. Spielfeld und Webcam

Die Webcam zur Spielfelderkennung wurde an einem Gestell aus Aluminiumprofilen befestigt. So kann sie direkt von oben auf das Spielfeld gucken. Da die Kamera aber so viel wie möglich von Spielfeld sehen soll, muss es ca. $126\,\mathrm{cm}$ lang und $89\,\mathrm{cm}$ breit sein. An einer der schmaleren Spielfeldseiten befindet sich mittig ein Tor.

C. Farberkennung

Wenn die Kamera ein Bild aufnimmt, werden drei Matrizen erschaffen, die so groß sind wie die Auflösung der Kamera. Jede Matrix gibt entweder den Rot-, Grün- oder Blauanteil des Bildes an. Es werden nun die Farbanteile berechnet und in jeweils eine Matrix gepackt. Aus diesen Matrizen werden die Pixel herausgesucht, die einen genügend hohen Anteil der jeweiligen

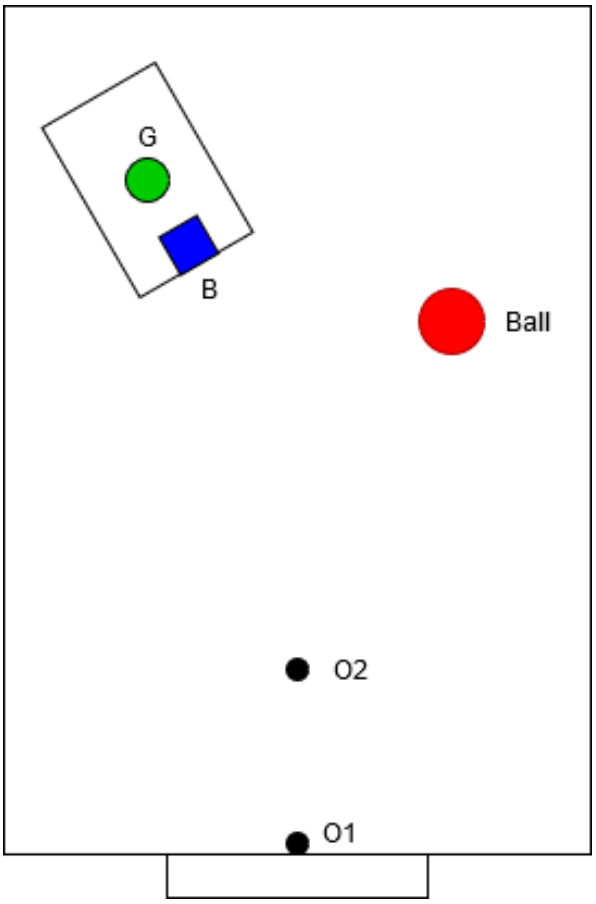


Abbildung 4. Übersicht der Orientierungspunkte

Farbe haben und erneut in drei Matrizen überführt. Nun wird für jeden Farbanteil der Mittelpunkt berechnet. Da das Spielfeld neutral ist, sollten die Farbmittelpunkte den Mittelpunkten der Orientierungshilfen entsprechen.

D. Orientierung und Bewegung

Es gibt insgesamt fünf Orientierungspunkte: Den roten Ball (R), den grünen Punkt auf der Drehachse des Roboters (G), einen blauen Punkt über dem Greifer (B) und zwei programminternen Punkten. Einer in der Mitte des Tores (O1) und einen vor dem Tor (O2). Der grüne Punkt ist der Hauptorientierungspunkt. Von diesem werden alle Vektoren berechnet. Bei der ersten Aufnahme werden die Vektoren GB und GR gebildet und der Winkel mit Hilfe der atan2-Funktion. Mit dieser Funktion kann der Roboter erkennen, ob der Ball links oder rechts liegt, da diese Funktion einen Wertebereich von 180° bis -180° hat. Wenn der Winkel positiv ist, muss er sich nach rechts drehen, wenn nicht, nach links. Bei Foto 2 wird der Winkel zwischen den Vektoren GB und GO1 und bei Foto 3 der Winkel zwischen GB und G01 berechnet. Der Roboter muss sich dann um diesen jeweils berechneten Winkel drehen.

E. Gesamtes Programm

Der Ablaufplan des gesamten Programms ist in Abbildung 5 dargestellt. Nachdem die Kamera, das Spielfeld und das Tor ausgerichtet sind, können Roboter und Ball beliebig auf dem Spielfeld platziert werden. Dann muss das Programm nur noch gestartet werden und der Roboter sucht den Ball selbstständig orientiert sich und versucht dann ein Tor zu schießen. Wenn die dritte Aufnahme gemacht wurde und der Roboter sich zum Tor gedreht hat, öffnet er den Greifer, nimmt Anlauf und fährt dann gegen den Ball, der ins Tor rollen sollte.

IV. Ergebnisdiskussion

In diesem Projektseminar konnte ein Roboter entwickelt werden, der beliebig in einem Spielfeld platziert werden kann und sich ohne weitere Einflüsse orientieren kann, um schlussendlich ein Tor zu erzielen. Ein Problem stellte der Greifer dar, da der Ball nicht immer mittig auf den Roboter traf und so der Farbsensor nicht ausgelöst wurde. Dies wurde behoben, indem zwei weitere Farbsensoren angebracht wurden, sodass der Ball an jeder Stelle des Greifers erkannt wurde. Weitere Schwierigkeiten bereitete uns die korrekte Drehung, da zur Winkelberechnung die Kosinusfunktion benutzt wurde. Dieses Problem wurde durch die Verwendung der atan2-Funktion behoben. Ein weiteres Problem stellten der Ball und das Spielfeld dar. Diese waren zu uneben, sodass der Ball beim ablegen wegrollte und so nicht perfekt vor dem Tor liegen blieb. Dieses Problem konnte aus zeitlichen Gründen nicht gelöst werden, weshalb der Roboter nur in etwa 75 Prozent der Fälle das Tor trifft. Die Ungenauigkeiten der Motoren von Lego-Mindstorms kann allerdings nicht beseitigt werden, weshalb immer eine Chance besteht, dass eine zu ungenaue Bewegung stattfindet und der Ball nicht im Tor landet. Aus zeitlichen Gründen jubelt der Roboter, egal ob der Ball im Tor ist oder nicht.

V. Zusammenfassung und Fazit

Während des Projektseminars wurde ein Roboter konstruiert und entwickelt, der „Fußball spielt“. Das Programm wurde so realisiert, dass der Ball und der Roboter beliebig auf dem Spielfeld platziert werden können und das Tor trotzdem mit hoher Wahrscheinlichkeit erzielt wird. Dies wurde mit Hilfe einer Webcam geschafft, die auf das ganze Spielfeld von oben blickt und mit Orientierungspunkten weiß, wie sich der Roboter bewegen muss, um ein Tor zu erzielen. In Zukunft kann zu diesem Projekt noch ein grafische Benutzeroberfläche entwickelt werden, mit der man zum Beispiel die Seite auswählen könnte, auf der das Tor erzielt werden soll. Außerdem kann das Wegrollen des Balles aufgrund von Unebenheiten reduziert werden, indem der Ball beim Anlauf mitgeführt wird und der Greifer erst beim Schuss geöffnet wird. Um den Roboter nur jubeln zu lassen, wenn er ein Tor erzielt hat, müsste man die Webcam noch eine Aufnahme machen lassen und das Programm entsprechend ergänzen.

Literatur

[1] Mathworks, https://de.mathworks.com/help/supportpkg/raspberrypiio/ref/track-a-green-ball.html (Abruf: 20.03.2020)

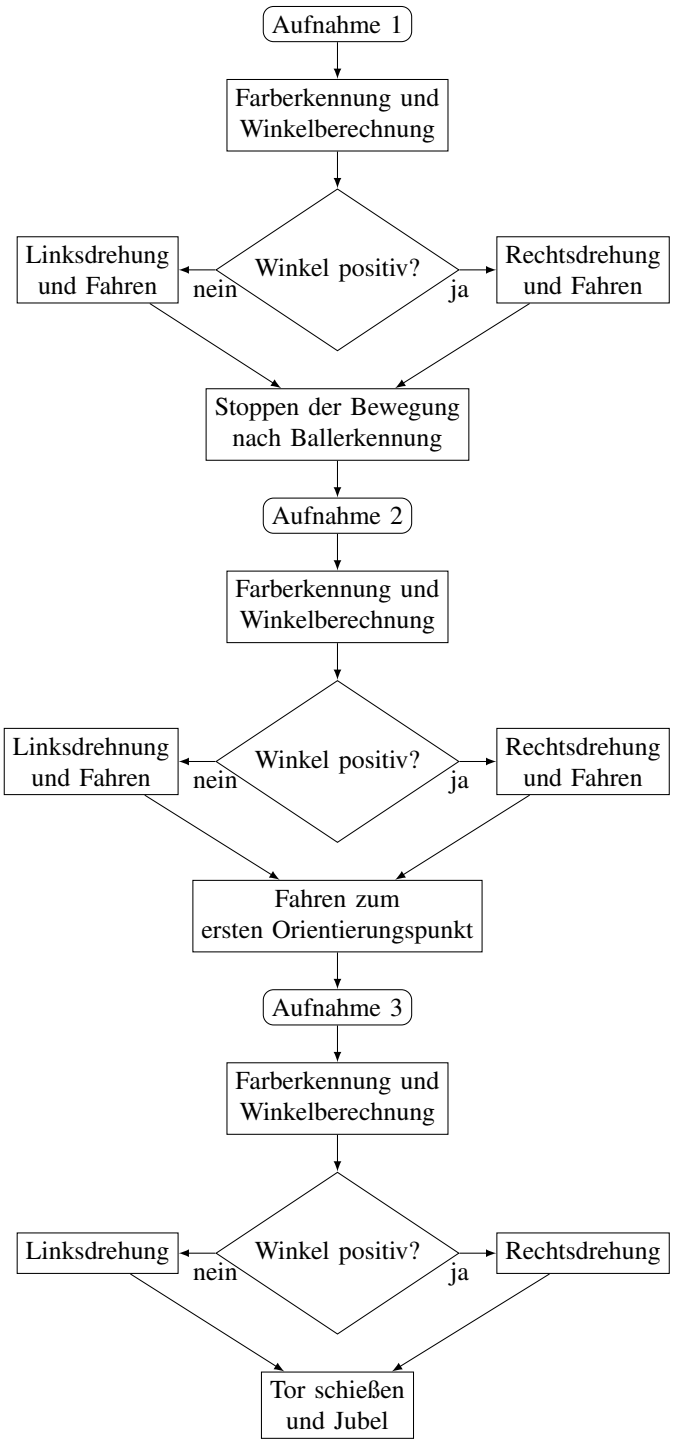


Abbildung 5. Gesamter Programmablaufplan

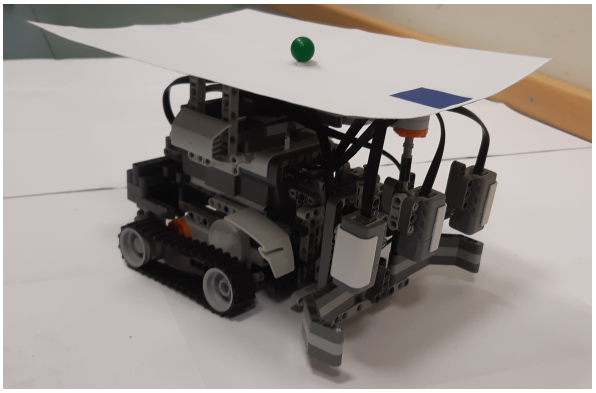


Abbildung 6. Fertiger Roboter mit Abdeckung

Verteilzentrum für Bestellungen

Friedrich Schneider, Elektrotechnik/Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Heutzutage werden Pakete in einem Verteilzentrum mithilfe eines Kippmechanismus sortiert. Das Verteilzentrum für Bestellungen, mit dem sich in diesem Paper auseinandergesetzt wird, soll den Ablauf und die grundlegende Funktionsweise in einem Verteilzentrum aufzeigen. Dabei wird hauptsächlich auf die Planung, die technische Umsetzung, die Programmierung und die aufgetretenen Probleme eingegangen. Zum Abschluss des Seminars wurde ein funktionstüchtiger Roboter präsentiert, der Farbkugeln anhand einer Bestellung verteilt.**

***Schlagwörter*—LEGO Mindstorms, NXT, OVGU, Otto-von-Guericke Universität, Projektseminar, Verteilzentrum**

I. Einleitung

ROBOTER und automatisierte Prozesse sind aus unserem heutigen Leben kaum noch wegzudenken. Gerade in der Logistik-Branche werden sie in immer größeren Maße eingesetzt, denn die Anzahl der Pakete, die im Durchschnitt täglich versendet werden, ist in den letzten Jahren deutlich gestiegen. Rund 700 000 Pakete pro Tag werden allein in den Paketzentren der schweizerischen Post verarbeitet [1].
Auf Grundlage eines Verteilzentrums, wie es Versanddienstleister wie Amazon, die Post oder Hermes betreiben, kam die Idee dies aus LEGO in vereinfachter Form nachzubauen. Es soll also möglich sein eine Bestellung aufzugeben und der LEGO Roboter soll erkennen, wo der bestellte Artikel liegt, um ihn in den passenden Bestellkorb zu transportieren. Mit diesem Ziel wurde das LEGO Verteilzentrum beim diesjährigen Projektseminar zwischen dem 11. Februar und dem 24. Februar 2020 entwickelt und gebaut.

II. Stand der Technik / Grundlagen

Im Folgenden wird zunächst auf die Funktionsweise eines Paket-Verteilzentrums eingegangen und dann anhand einer kurzen Erläuterung zu ähnlichen Projekten auf eine mögliche Umsetzung mit LEGO geschlossen.

A. Vorbild

Was ist eigentlich ein Verteilzentrum?

> Laut der DHL Tochterfirma Salodoo!, sorgt es dafür, dass Waren zwischen Unternehmen und Kunden möglichst schnell und effizient transportiert werden. Das Verteilzentrum, welches auch unter dem Begriff Verteillager bekannt sei, sorge dafür das Unternehmen im weltweiten Handel konkurrenzfähig bleiben, da es für einen optimalen Warenfluss verantwortlich sei. [2]

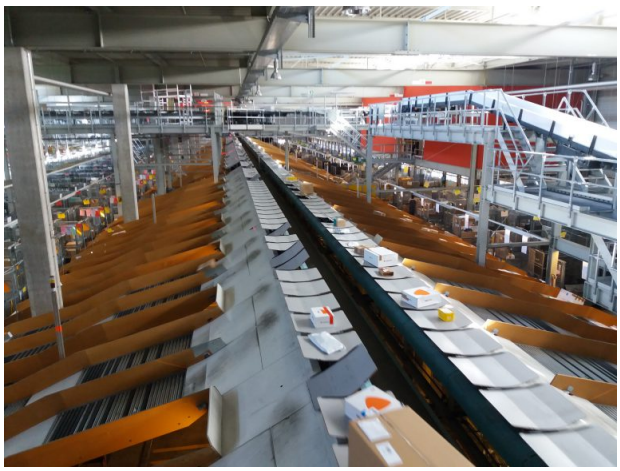


Abbildung 1. Einblick in ein Schweizer Post-Verteilzentrum [4]
Mittig: graue Kippschalen-Sortierer
Links und Rechts: Speicherrutschen

In einem automatisierten Verteilzentrum, auch Mechanisierte Zustellbasis (MechZB) genannt, werden zunächst die per Lkw angelieferten Pakte per Teleskopförderband entladen und anschließend die Adressen und Barcodes der Pakete am Band mit Hilfe eines stationären Scanner automatisch gelesen. Anschließend werden sie automatisch per Kippschalen-Sortierer (siehe Abb. 1) „nach einzelnen Zustellbezirken sortiert und in Speicherrutschen-Endstellen gesammelt, die direkt vor dem Beladetor liegen, an dem das Zustellfahrzeug vom Zusteller geladen wird.“ [3]

B. Ähnliche Roboter

Bei der Projektfindung wurden mehrere Roboter betrachtet, die eine ähnliche Funktionsweise aufwiesen. Dabei handelte es sich hauptsächlich um „Farbsortierer“. Diese sind grundsätzlich so aufgebaut, dass sie einen Farbsensor nutzen, um farbige Steine oder Kugel zu erkennen. Für die Beförderung werden häufig Transportbänder genutzt. Somit bot es sich an, dass Farbkugeln Produkte bzw. Pakete darstellen sollten und das zur Unterscheidung statt Barcodes die Farbe der Kugel genutzt werden.
Die Vorbetrachtung war sehr hilfreich, da sie Lösungsideen für mögliche Probleme lieferte: so lässt sich z. B. die Übersetzung der Motoren mit Zahnradkonstruktionen anpassen und somit die Genauigkeit erhöhen.

DOI: 10.24352/UB.OVGU-2020-037

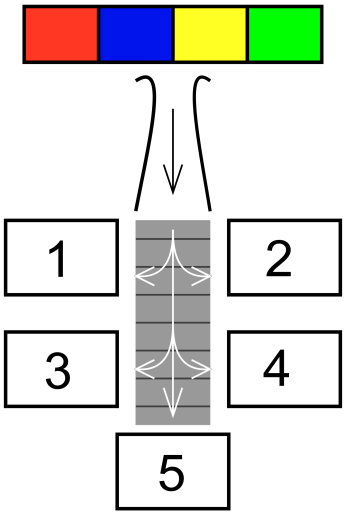


Abbildung 2. Skizze von der grundlegenden Idee

III. Umsetzung

Auf Basis eines Farbsortier-Roboters kam die Idee, ein automatisches Verteilzentrum zu konstruieren. Dazu standen ein LEGO Mindstorms Baukasten, ein NXT-Controller, 3 Motoren, sowie Farb-, Licht-, Tast-, und Ultraschallsensoren zur Verfügung. [5]
Die Zielstellung war es, dass der Roboter Bestellungen mit einer bestimmten Anzahl von farbigen Kugeln entgegen nimmt. Deren Position sollte zunächst automatisch ermittelt werden und dann in das der Bestellnummer zugehörige Fach transportiert werden. Dazu wurde eine Konzeptskizze erstellt, die in Abbildung 2 zu sehen ist.

A. Mechanischer Aufbau

Der Motor C (siehe Abb. 3) wurde zunächst auf einer Grundplatte befestigt und eine möglichst niedrige Übersetzung aus Zahnrädern konstruiert. Darauf aufbauend erfolgte der Bau von drei Fächern für die Aufbewahrung der Kugeln und ein Förderband für den Transport in den entsprechenden Bestellkorb. Die anfängliche Idee das Konstrukt aus Fächern und Förderband oberhalb des Motor C zu befestigen, stellte sich als viel zu instabil dar. Es wurde eine andere Lösung gefunden, die in Abbildung 3 und 4 zu sehen ist. Zunächst wurde der Schwerpunkt der Konstruktion ermittelt und aus LEGO-Steinen zwei Stützpfeiler gebaut. Den Abschluss der Pfeiler bildete jeweils ein Schneckengetriebe. Diese waren über eine entsprechenden Drehachse verbunden, um die schwebende Konstruktion mithilfe des Motors C kippbar zu machen (siehe Abb. 4).
Die Fächer wurden jeweils mit einer Öffnung versehen, die das Rausrollen der Kugeln ermöglichen sollte. Damit immer nur aus einem Fach eine Kugel rollt, kam eine Konstruktion aus mehreren Loch- und Zahnstangen zum Einsatz. Diese wurde über den Motor A (siehe Abb. 3) mit Hilfe eines Schneckengetriebes beweglich gemacht, sodass sich die Fächer öffnen und schließen lassen. Des Weiteren wurde an dem Öffnungsmechanismus der Farbsensor mit einer Stange oberhalb der Fächer befestigt. Um

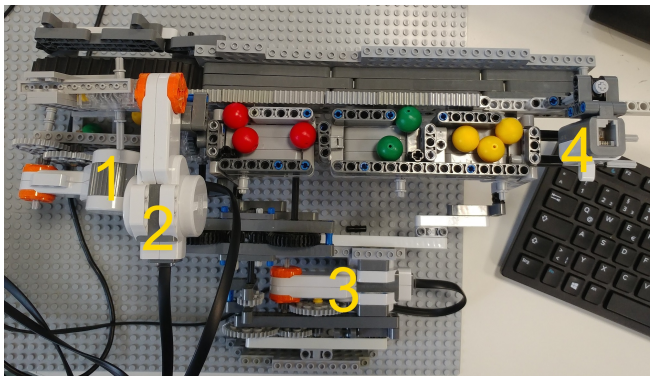


Abbildung 3. Roboter von oben
1 - Motor B (Band bewegen)
2 - Motor A (Fächer öffnen bzw. schließen)
3 - Motor C (vor oder zurück kippen)
4 - Farbsensor

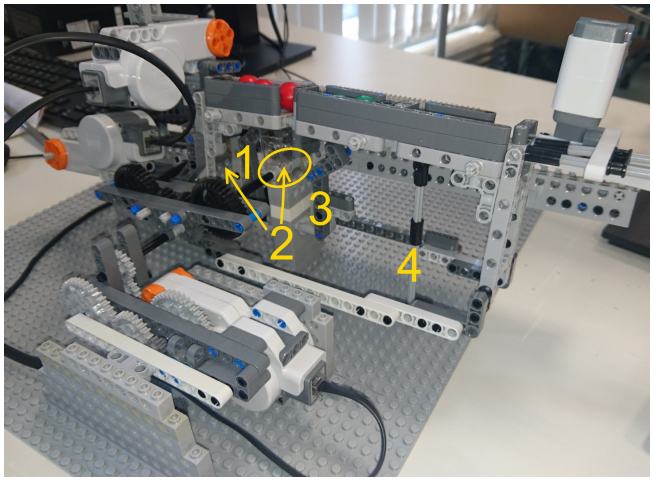


Abbildung 4. Roboter von der Seite
1,3,4 - Stäbe um die Böden hoch zu drücken
2 - Schneckengetriebe für die Kippbewegung

die Kugeln dann auf ein Förderband rollen zu lassen, wurden Lochstangen zu einer Art Rutsche zusammen verbunden und an den Fächern befestigt. Der Motor B (siehe Abb. 3) war für die Bewegung des Förderbandes zuständig, welches nahe genug an der Rutsche befestigt werden musste, damit die Kugeln auch transportiert werden. Mit einer Bande wurde erreicht, das die Kugeln nur in einer Richtung vom Förderband rollen können, wie in Abbildung 6 zu sehen ist. Nachdem der Prototyp fertiggestellt war, wurde mit dem Programmieren begonnen.

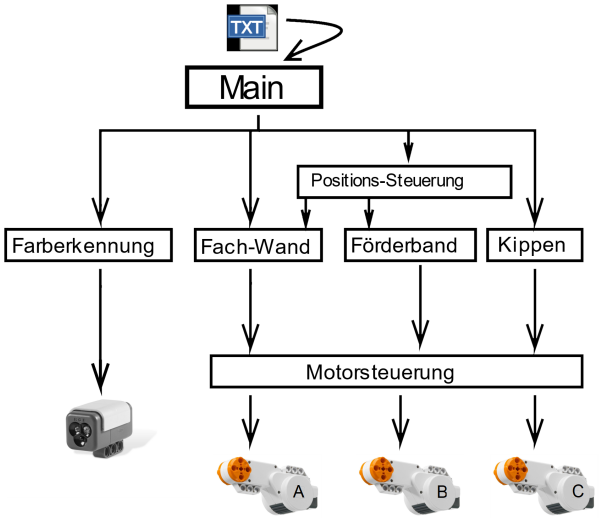


Abbildung 5. Grundlegender Funktionsablauf

B. Programmierung

Mithilfe der NXT Toolbox [6], welche von der RWTH Aachen entwickelt wurde, konnten die an den NXT-Controller angeschlossenen Motoren und Sensoren über MATLAB [7] ausgelesen und gesteuert werden.
Das Programm bestand aus einer Hauptfunktion und mehren Unterfunktionen, die gegenseitig mit den benötigten Übergabeparametern kommunizierten (siehe Abb. 5).
Zu Beginn fuhr der Farbsensor alle Fächer ab und ermittelte die Farbe der Kugeln. Diese wird dann in einer Tabelle, bzw. genauer einem Struct, mit der entsprechenden Fachnummer gespeichert. Für die Positionierung des Sensors wurde die Zeit ermittelt, wie lange der Motor A bei einer bestimmten Leistung drehen muss, bis der Sensor an der entsprechenden Position ist. Nach diesem „Stoppuhr-Prinzip“ wurde bei den anderen Motoren auch vorgegangen. Somit ergaben sich für Motor A fünf Positionen, drei zum Öffnen der Fächer, eine zwischen Fach eins und zwei, sowie eine zwischen Fach zwei und drei. Der Motor B hatte vier Positionen, die jeweils der Bestellnummer entsprachen. Für Motor C gab es nur zwei Positionen, zum einen nach vorn kippen, sodass die Kugel aus dem Fach rollt, und nach hinten kippen, sodass die Kugel vom Band in den Bestellkorb rollt (siehe Abb. 6).

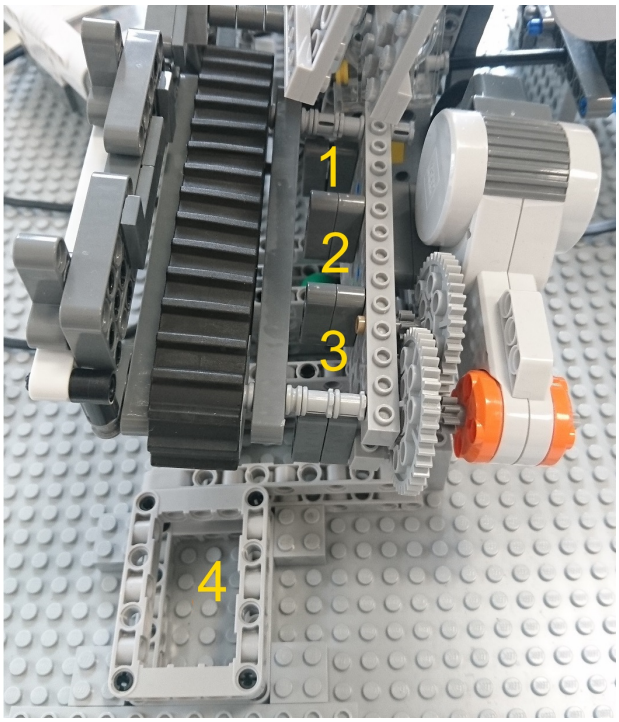


Abbildung 6. Band mit den Bestellkörben
1 - Position 1 für Bestellung 1 (= 1 s)
2 - Position 2 für Bestellung 2 (= 4 s)
3 - Position 3 für Bestellung 3 (= 7 s)
4 - Position 4 für Bestellung 4 (= 10 s)

Tabelle I
BEISPIEL FÜR EINE BESTELLUNG

Bestellnr.	Anz_Farbe_Rot	Anz_Farbe_Gruen	Anz_Farbe_Gelb
1	2	1	0
2	1	2	1
3	1	1	0
4	0	1	3

C. Die Bestellung

Für die Bestellung wurde ein einfache und praktische Lösung gefunden. Diese bestand darin, dass die Bestellung in einer normalen Textdatei eingegeben wurde. Das Programm las diese Datei dann ein und speicherte die Werte in eine Tabelle, wie es beispielhaft in Tabelle I zu sehen ist. Mit einer for-Schleife wurde dann jede Spalte durchlaufen und entsprechend der bestellten Anzahl, der Auslieferungsprozess für jede Farbe wiederholt gestartet.

IV. Ergebnisdiskussion

Zum Abschluss des Projektes konnte ein funktionstüchtiger Roboter präsentiert werden, der allerdings noch ein paar Probleme aufwies. Zum einen war es notwendig, das sich der Öffnungsmechanismus und die Neigung des Roboters in der richtigen Ausgangsposition befinden mussten, bevor das Programm aufgerufen werden konnte. Des Weiteren konnte in der begrenzten Zeit keine Lösung gefunden werden, die es gänzlich verhindert, dass immer nur eine Kugel aus dem Fach rollt. Dazu wäre ein zusätzlicher Motor notwendig gewesen, der eine weitere Wand versetzt zur vorhanden bewegt oder man hätte die Fächer nochmal grundsätzlich verändern müssen. Die zur Verfügung gestellten Kugeln stellten sich auch als etwas problematisch dar, da sie nicht vollkommen rund, sondern zwei Abflachungen aufwiesen, die dafür sorgten, dass die Kugel einfach liegen blieb, obwohl der Roboter ausreichend gekippt war. Auch der Öffnungsmechanismus stellte sich als nicht ganz zuverlässig heraus, da durch die Schwerkraft bedingt der Kontakt zwischen Zahnrad und Zahnstangen nicht immer ausreichend war. Positiv hervorzuheben ist der Kippmechanismus, durch den ein Motor eingespart werden konnte, der durch eine ausreichende Übersetzung zwar recht langsam, aber zuverlässig funktionierte.

V. Zusammenfassung und Fazit

Während des Projektseminars LEGO Mindstorms wurde erfolgreich ein einfaches Modell eines Paket-Verteilzentrums entwickelt, dessen Funktionsweise in den Grundzügen dem Vorbild recht nahe kommt. Natürlich ist es längst nicht so komplex, aber vorausgesetzt man hätte mehr Zeit und Teile zur Verfügung gehabt, dann wäre es z. B. durch Einsatz einer Webcam möglich gewesen, Bar- bzw. QR-Codes einzulesen und darauf basierend eine Verteilung vorzunehmen. Die Bestellung wäre dann auch über eine GUI realisiert wurden. Des Weiteren bot der Kippmechanismus auch noch Optimierungspotential, wenn man durch Erhöhung der Übersetzung das Kippen beschleunigt hätte.

Literaturverzeichnis

[1] HSR Hochschule für Technik Rapperswil: HSR forscht am sich selbst überwachenden Paketzentrum. In: *HSR Magazin* (2018), Nr. 1, 34–35. https://leanbi.ch/wp-content/uploads/2018/07/leanpredict_HSR_Magazin_1.2018.pdf

[2] *Was ist ein Verteilzentrum? Definitionen & Erklärungen — Saloodo!* https://www.saloodo.com/de/logistik-lexikon/verteilzentrum/. Version: 05.03.2020

[3] Wikipedia (Hrsg.): *Mechanisierte Zustellbasis*. https://de.wikipedia.org/w/index.php?title=Mechanisierte_Zustellbasis&oldid=194704237. Version: 05.03.2020

[4] *Predictive Maintenance im Schweizer Post-Verteilzentrum mit LeanPredict - LeanBI*. https://leanbi.ch/blog/predictive-maintenance-im-schweizer-post-verteilzentrum-mit-leanpredict/. Version: 04.03.2020

[5] *MINDSTORMS® — Themenwelten — Offizieller LEGO® Shop DE*. https://www.lego.com/de-de/themes/mindstorms. Version: 06.03.2020

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

[7] *MATLAB - MathWorks - MATLAB & Simulink*. https://de.mathworks.com/products/matlab.html. Version: 2019

Bau eines Verteilzentrums mit LEGO Mindstorms

Joachim Wanke, B. Sc. Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Kurzfassung*—Modernes Einkaufen erfolgt immer öfter per Knopfdruck. Der Online-Handel boomt. Um Kosten zu senken und die Effizienz zu steigern, bietet sich für Unternehmen eine Automatisierung ihrer Logistik an. Im Rahmen eines Projektseminars an der Otto-von-Guericke-Universität Magdeburg wurde ein Modell für die Automatisierung von Verteilzentren entwickelt. Die Konstruktion erfolgte mit LEGO Mindstorms. In das Programm der Anlage gibt man mehrere Bestellungen ein. Die Anlage erkennt die Produkte, wählt sie aus und verteilt deren Exemplare richtig auf die Bestellungen. Dieses Paper erläutert die mechanische Konstruktion, die Programmierung mittels mehrerer verknüpfter Funktionen und den Ablauf der Verteilung. Resultat ist eine funktionstüchtige Anlage, die größtenteils problemlos funktioniert, aber noch Verbesserungspotential hat.**

***Schlagwörter*—Automatisierung, LEGO, Logistik, Mindstorms, Onlinehandel, Verteilzentrum**

I. Einleitung

EFFIZIENT zu arbeiten, ist das Ziel jedes Unternehmens, das wettbewerbsfähig bleiben will. Um im Preiskampf mit anderen Anbietern zu bestehen, sparen Unternehmen, wo immer sie können. Eine Möglichkeit zur Kostenreduzierung liegt in der Automatisierung der Logistik. Eine zentrale Rolle nehmen dabei Verteilzentren ein. Hier werden angebotene Artikel auf ihre Bestellungen verteilt.

Eine der heute boomenden Branchen ist der Onlinehandel. Hier ergibt sich eine besonders einfache Möglichkeit, Prozesse zu automatisieren. Die Branche ist sehr dynamisch und wächst stabil [1], daher muss viel neue Infrastruktur entstehen. Bei deren Aufbau können von Anfang an technische Innovationen umgesetzt werden. Verteilzentren bieten sich deshalb besonders gut für die Automatisierung an.

Auf automatisierte Verteilzentren setzt beispielsweise Amazon. Bisher stellen Mitarbeiter die Bestellungen zusammen, in dem sie die bestellte Ware manuell einsammeln. Der Konzern will jedoch expandieren und plant neue Verteilzentren. Dort soll der Betrieb automatisiert werden. Amazon senkt so Kosten und steigert gleichzeitig das Tempo. [2] Ein automatisiertes Verteilzentrum erleichtert eine Erweiterung der Lager, man kommt mit wenigen Arbeitskräften aus und ist flexibler. [3]

II. Vorbetrachtungen

A. Idee

Ziel des Projekts war der Bau eines Modells für ein solches automatisiertes Verteilzentrum. Grundlegend soll es verschiedenfarbige LEGO-Plastikkugeln sortieren. Jede Fabe entspricht je einem Produkt. Die Kugeln sind demnach die Produktexemplare. Zu Anfang sollen diese nach Produkten (Farbe) sortiert in Aufbewahrungsfächern liegen. Am Ende soll jede Bestellung fertig zusammengestellt in je einem Bestellkorb liegen.

Das Verteilzentrum muss dazu verschiedene Aufgaben ausführen. Dazu gehört die Produkterkennung, umgesetzt durch den Farbsensor. Das richtige Produkt muss ausgewählt werden, zu den Bestellkörben transportiert und dort in den richtigen Bestellkorb einsortiert werden. Die Auswahl übernimmt im Modell die Fach-Wand, der Transport erfolgt über die Rutsche und die Einsortierung erfolgt über die Laufzeit des Förderbandes.

Zur konkreten Umsetzung des Projekts stand der LEGO-Mindstorms-Baukasten zur Verfügung. Zur Auswahl standen Sensoren für Farberkennung, Licht-/Helligkeitserkennung, Geräuscherkennung, Ultraschall und ein Tastsensor. Kernstück ist der programmierbare NXT-Stein. Er besitzt drei Ausgänge für Motoren und vier Eingänge für Sensoren (vgl. [4] S. 20).

B. Konzept

Das Verteilzentrum sollte in der Lage sein, 4 verschiedene Produkte in jeweils variabler Stückzahl auf 5 Bestellungen zu verteilen. Zu Anfang sollten Exemplare jedes Produktes (d. h. Farbkugeln gleicher Farbe) in jeweils einem einzelnen Fach liegen. Die Anlage sollte nun ein Produkt nach dem anderen auswählen, und einzeln Exemplare aus ihren Fächern entnehmen. Auf einer Rutsche sollten diese nun auf ein Förderband rollen. Das Förderband sollte die Produktexemplare zur richtigen Position bringen, sodass die Anlage sie in die richtigen Bestellkörbe bringen konnte. Die Eingabe von bis zu 5 Bestellungen gleichzeitig sollte über ein Graphical User Interface (GUI) erfolgen. Aus Gründen der Effizienz werden zunächst alle bestellten Exemplare des ersten Produktes, dann des zweiten etc. verteilt.

Um den Aufwand der Anlage zu reduzieren, wurde im Lauf des Baus die Anzahl der Bestellkörbe von fünf auf vier und die Anzahl Produkte von vier auf drei verringert.

III. Hauptteil

A. Mechanische Realisierung

Zunächst musste eine geeignete Konstruktion geschaffen werden. Dafür wird auf das Prinzip der Schwerkraft zurückgegriffen. So bewegen sich die Produktexemplare im Laufe der Sortierung immer weiter nach unten. Die Aufbewahrungsfächer liegen demnach oben. Sie sind entlang der Rutsche angeordnet, auf der die Produktexemplare zum Anfang des Förderbandes rutschen. Die Neigung der Rutsche ist gering, um unterschiedliche Geschwindigkeiten zu verhindern. Beim Auftreffen auf das Förderband wird das Produktexemplar durch hervorstehende Rillen gestoppt und in Position gehalten. Das Förderband fährt nun das Produktexemplar in eine Position direkt über dem richtigen Bestellkorb. Danach werden die

DOI: 10.24352/UB.OVGU-2020-038

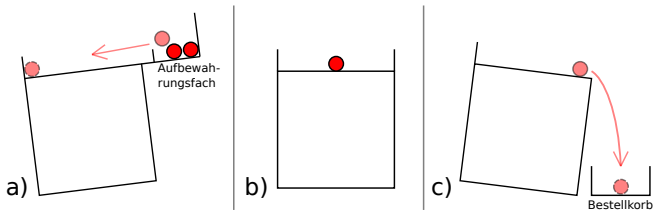


Abbildung 1. Kippmechanismus: a) Anlage nach vorne gekippt, b) Anlage nicht gekippt, c) Anlage nach hinten gekippt.

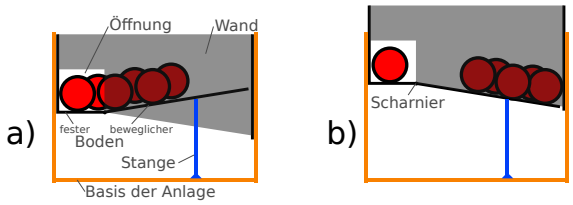


Abbildung 2. Vereinzelungs-Mechanimus a) bei nach vorne gekippter Anlage, b) bei nach hinten gekippter Anlage.

Produktexemplare seitlich vom Band geschoben und fallen so in den richtigen Bestellkorb.

Eine Hürde bei der Umsetzung bestand darin, dass der NXT-Baustein nur drei Ausgänge für Motoren hat (vgl. [4] S. 20). Also ging es darum, mit nur drei Motoren die Auswahl des Faches, Bewegung des Förderbandes, Transport aus dem Fach auf die Rutsche und Herunterschieben in den richtigen Bestellkorb zu realisieren. Wir entschlossen uns dazu, die zwei letztgenannten Aufgaben zusammenzufassen. Dazu ist die gesamte Vorrichtung in ihrer Längsachse kippbar. Nach vorne gekippt, fälllt ein Produktexemplar bei geöffnetem Aufbewahrungsfach auf die Rutsche (vgl. Abbildung 1). Seitlich an der Rutsche und am Förderband ist einseitig eine Wand angebracht. Sie hindert das Produktexemplar daran, aus der Sortieranlage zu fallen. Das Produktexemplar rutscht zum Förderband. Nun fährt das Produktexemplar auf dem Förderband bis über den richtigen Bestellkorb. Die Anlage kippt in die andere Richtung. Auf der nun unten liegenden Seite ist keine Wand angebracht. So kann das Produktexemplar in den richtigen Bestellkorb fallen (vgl. Abbildung 1).

Der Farbsensor ist mit einem Arm an der Fach-Wand befestigt, sodass er sich über den Kugeln befindet und deren Farbe scannen kann.

Um ein Produktexemplar auf die Rutsche zu lassen, muss das richtige Aufbewahrungsfach geöffnet werden. Die Fächer sind an einer Stelle der an der Rutsche liegenden Seite offen. Zwischen dieser Öffnung und der Rutsche befindet sich eine bewegliche Fach-Wand. Sie hat in ihrer Mitte genau eine Öffnung, durch die ein Produktexemplar passt. Per Motor kann diese Fach-Wand so positioniert werden, dass sich ihre Öffnung genau an der Öffnung eines Aufbewahrungsfaches befindet. Somit kann ein Produktexemplar durch beide Öffnungen auf die Rutsche fallen. Sollen alle Fächer geschlossen sein, wird die Fach-Wand so positioniert, dass sich ihre Öffnung mit keiner der Fachöffnungen überschneidet.

Um sicherzustellen, dass nur eine Kugel gleichzeitig ausgegeben wird, ist der Boden jedes Aufbewahrungsfach teilweise beweglich (siehe Abb. 2). Auf den festen Teil des Bodens passt genau eine Kugel. An der Basis der Anlage (orange) ragt eine Stange bis zum Boden des Faches auf (blau eingezeichnet). Kippt die Anlage nach vorne (Abb. 2a), senkt sich der Boden des Aufbewahrungsfachs auf die Stange. Diese drückt daher den Boden des Faches nach oben. So rollen alle Kugeln im Fach in Richtung Öffnung. Kippt die Anlage nach hinten (Abb. 2b), dann hebt sich der Boden von der Stange und klappt so nach unten. Alle Kugeln bis auf eine (die genau auf den festen Teil des Bodens passt) rollen von der Öffnung weg. Da im Zustand (b) das Fach geöffnet wird, kann nur eine Kugel gleichzeitig ausgegeben werden.

B. Programmierung

Die gesamte Programmierung erfolgte in der Programmiersprache MATLAB. Gesteuert wird das Verteilzentrums über mehrere Funktionen, die sich gegenseitig und teilweise rekursiv aufrufen. An oberster Stelle steht dabei die Main-Funktion.

Für die Bewegung des Kippens, des Förderbandes und der Fach-Wand sind entlang der Bewegungsrichtung Positionen festgelegt. Jede Position ist für unterschiedliche Zwecke nützlich. Der Kippmechanismus kennt zwei, die Fach-Wand 5 und das Förderband 4 Positionen. Die Positionen der Fach-Wand und ihr Zweck ist in Abbildung 3 dargestellt.

Die Bewegung zu den entsprechenden Positionen wird mit je einer Positionssteuerung realisiert. Die Main-Funktion übergibt als Parameter nur die aktuelle Position der Fach-Wand und die Zielposition. In der Postionssteuerung sind die Zeiten gespeichert, die der Motor laufen muss, um von jeder Position in die benachbarte zu gelangen. Die Positionssteuerung übergibt zunächst einen Befehl an die Motorsteuerung, den Motor mit voller Geschwindigkeit in die Richtung laufen zu lassen, in der die Zielposition liegt. Dann wird der Programmablauf für die Zeit unterbrochen, die der Motor zur Bewegung in die benachbarte Position braucht. Anschließend wird der Motorsteuerung der Befehl zum Anhalten des Motors übergeben. Der Motor hält an und die Fach-Wand befindet sich in der benachbarten Position. Ist dies die Zielposition, beendet sich die Positionssteuerung. Falls nicht, ruft sich die Positionssteuerung rekursiv auf, um die nächste Position in Richtung Zielposition zu erreichen. Sie ändert die Startposition allerdings zu jener, in der sich die Fach-Wand gerade befindet.

Das Kippen der Anlage erfolgt ebenfalls indirekt. Die Main-Funktion gibt den Befehl dazu an eine eigene Positionssteuerung. Diese ist nach dem gleichen Schema wie die Fach-Wand-Positionssteuerung aufgebaut. Jedoch gibt es hier nur zwei Positionen (siehe Abbildung 1): nach hinten (a) und nach vorne (c) gekippt. Daher wird nur ein Parameter übergeben: die gewünschte Kippposition.

Zur Bewegung des Förderbandes übergibt die Main-Funktion die Bestellnummer an die Band-Funktion. In der Band-Funktion ist gespeichert, wie lange sich das Band bewegen muss, um das Produktexemplar bis über den richtigen Bestellkorb zu fahren. Die Band-Funktion gibt den Befehl zum Start des Motors an die Motorsteuerung. Dann wird das Programm für den entsprechenden Zeitraum unterbrochen. Daraufhin wird der

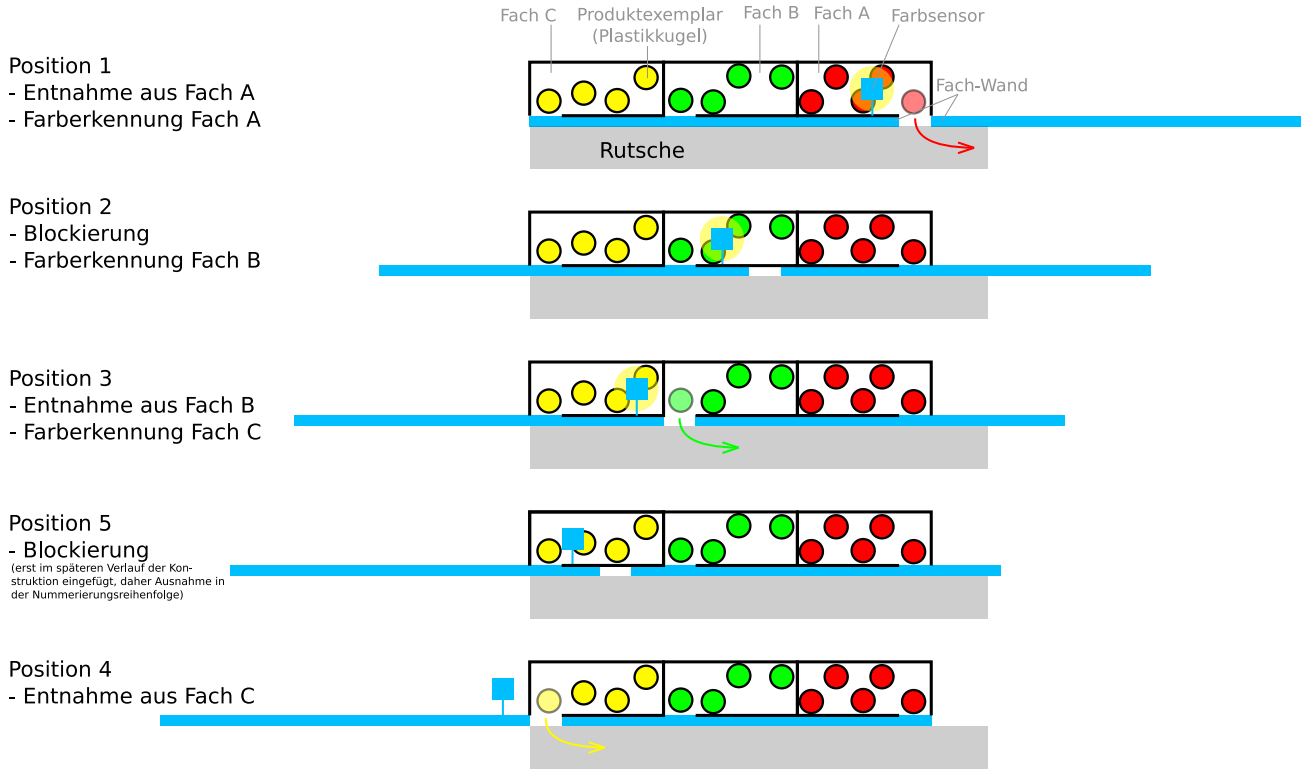


Abbildung 3. Positionen der Fach-Wand

Befehl zum Stopp des Motors an die Motorsteuerung gegeben. Das Band hält an und das Produktexemplar befindet sich über dem richtigen Bestellkorb.

Die Motorsteuerung ist für alle Motoren gleich und erhält als Parameter den Port des anzusteuernden Motors am NXT und die Batterieleistung in Prozent, mit der sich der Motor drehen soll. Dies ist bei voller Geschwindigkeit ±100 (je nach Bewegungsrichtung) und bei Stopp des Motors 0. Die Motorsteuerung führt danach die NXT-spezifischen Befehle aus, die den Motor wie gewünscht ansteuern. Sie hat also nur den Zweck, diese Codepassagen aus der Positionssteuerung auszulagern und den Code so übersichtlicher zu machen.

Die ursprünglich geplante GUI wurde nicht realisiert. Die Bestellungen werden stattdessen in ein TXT-Dokument eingegegeben, dieses gespeichert und von der Main-Funktion eingelesen.

C. *Programmablauf*

Zu Anfang des Programms sollen die Farben erkannt werden. Der Farbsensor ist mechanisch an der Fach-Wand angebracht. Daher wird zunächst die Fach-Wand-Funktion mehrmals aufgerufen, um den Farbsensor nacheinander über alle Fächer zu bringen und so alle Farben zu erkennen. Die Information, welche Farbe in welchem Fach liegt, wird in einem Array gespeichert.

Als nächstes liest die Main-Funktion direkt das TXT-Dokument ein, in dem die Bestelldaten stehen (siehe Abb. 4). Beginnend mit der ersten Zeile wird zunächst die Bestellnummer (erster Eintrag) ausgelesen. Dann wird der zweite

```
Bestellnr. Anz_Farbe_Rot Anz_Farbe_Gruen Anz_Farbe_Gelb
2 1 2 1
1 0 2 0
4 1 0 2
3 1 0 1
```

Abbildung 4. TXT-Dokument, das die Main-Funktion einliest. Die oberste Zeile gibt an, welche Spalte welche Information enthält.

Eintrag abgefragt, der die Anzahl bestellter roter Kugeln angibt. So viele Male, wie rote Kugeln bestellt werden, gibt die Sortieranlage nun rote Kugeln in den Korb mit der Bestellnummer. Das Programm fährt nun mit der zweiten, dann mit der dritten etc. Zeile fort und verteilt so zunächst die roten Kugeln. Sind alle Zeilen, also alle Bestellungen, durchgegangen, folgt die Verteilung der grünen Kugeln. Die Vorgehensweise gleicht der Verteilung der roten Kugeln, nur wird nun der dritte statt dem zweiten Eintrag in der jeweiligen Zeile abgefragt. Dieser gibt die Anzahl bestellter grüner Kugeln an. Nach der Verteilung der grünen Kugeln folgt nach gleichem Prinzip die Verteilung der gelben und dann das Programmende.

D. *Prozess: Ausgabe eines Produktexemplars*

Die Ausgabe der Produktexemplare funktioniert nach folgendem Prinzip: Zu Anfang ist die Anlage nach vorne gekippt. Die Fach-Wand fährt in eine Position, bei der keine Kugeln aus ihren Fächern rollen können. Dies ist Position 2, falls aus Fach A oder B eine Kugel entnommen werden soll; oder Position 5,

falls als nächstes Fach C geöffnet werden soll (siehe Abb. 3). Die Anlage kippt nun nach hinten, die Kugeln werden von der Fach-Wand noch blockiert, sodass sie nicht auf die Rutsche rollen. Je nach benötigtem Fach fährt die Fach-Wand auf die Positionen 1 (Fach A), 2 (Fach B) oder 4 (Fach C) und öffnet so das jeweilige Fach (siehe Abb. 3). Eine Kugel rollt aus dem Fach auf die Rutsche und danach auf das Förderband. Das Förderband setzt sich in Bewegung und fährt die Kugel über den richtigen Bestellkorb. Die Anlage kippt nach vorne. So rollt die Kugel vom Band und fällt in den richtigen Korb.

IV. Ergebnisdiskussion

Aus unserer Arbeit ist eine Sortieranlage entstanden, die ihre Anforderungen in großen Teilen erfüllt. Die zu erfüllenden Aufgaben Auswahl, Transport und Einsortierung werden gelöst. Eine farblich erkannte, aus dem Fach ausgeworfene Kugel erreicht den richtigen Bestellkorb zuverlässig.

A. Bestehende Probleme

Ein seltenes, aber vorhandenes Problem besteht mit der Form der Kugeln aus dem Mindstorms-Kasten. Sie sind an einer Seite leicht abgeflacht, sodass sie manchmal liegen blieben, statt zu rollen.

In Einzelfällen erkannte der Farbsensor schwarz statt der richtigen Farbe. Mit Justierungen an seiner Befestigung und anderer Platzierung der Exemplare im Fach konnte dieses Problem oft umgangen werden.

Probleme macht auch der Vereinzelungs-Mechanismus, der schwierig umzusetzen ist. Aufgrund des knappen Zeitplans des Projektes gelang es nicht mehr, den Mechanismus zuverlässig zu konstruieren. Bei Fach A war der Platz zu eng, um zwischen der Kippachse und dem Fachboden den Mechanismus einzubauen. Bei den anderen Fächern war der Unterschied zwischen nach vorne und nach hinten gekippter Anlage so gering, dass der Boden sich nicht ausreichend hob und senkte.

B. Optimierungsmöglichkeiten

Die Eingabe könnte optimiert werden, indem ein GUI eingebaut würde. Dies wäre optisch ansprechender als das TXT-Dokument.

Bei Testläufen ergaben sich Ungenauigkeiten der Motoren bei niedriger Drehzahl. Zur Bewegung der Anlagenkomponenten sind jedoch niedrige Drehzahlen vonnöten. Zudem benötigt das Kippen der Anlage aufgrund ihres Gewichtes eine erhebliche Kraft. Daher wurden Zahnrad-Übersetzungen eingebaut, die die Drehzahl der Motoren, dadurch aber auch die Geschwindigkeit der Sortierung, verringern. Alle Motoren werden immer mit maximaler Drehzahl angesteuert, um die Effekte „Genauigkeit“ und „Kraft“ zu maximieren. Das hat jedoch zur Folge, dass immer nur ein Motor gleichzeitig laufen kann, da er mit voller Batterieleistung angesteuert wird. Dies verringerte die Geschwindigkeit der Verteilung weiter. Hier besteht noch Optimierungsbedarf, den wir noch in der Projektzeit erkannten. Aufgrund des begrenzten Zeitplans scheiterte jedoch eine Neukonstruktion der Zahnradübersetzung. Diese wäre zudem zwangsweise verbunden mit einer Umprogrammierung der Anlage aufgrund veränderter Motorlaufzeiten (sieht Abschnitt „Programmierung“).

V. Zusammenfassung und Fazit

Insgesamt genügt das Resultat seinem Anspruch. Es ist gelungen, ein Modell eines automatischen Verteilzentrums nach dem anfänglichen Konzept zu konstruieren. Die Anlage sortiert die korrekte Zahl Produktexemplare in die richtigen Bestellkörbe. Durch den Kippmechanismus wurde die Einschränkung von nur 3 Motoren umgangen. Die Probleme, Kugeln nicht einzeln ausgeben zu können und seltene Fehlfunktionen der Kugeln und des Farbsensors bleiben jedoch bestehen. Die Anlage hat noch Optimierungspotential bei der Sortierungsgeschwindigkeit. Optisch könnte die Anlage durch ein GUI ansprechender werden.

Literaturverzeichnis

[1] IBI RESEARCH AN DER UNIVERSITÄT REGENSBURG GMBH: *Prognose: E-Commerce-Anteil am Einzelhandelsumsatz wird bis 2025 nochmals deutlich steigen.* https://ibi.de/aktuelle-meldungen/prognose-e-commerce-anteil-am-einzelhandelsumsatz-wird-bis-2025-nochmals-deutlich-s Version: Januar 2020

[2] HOFER, Joachim: *Amazon automatisiert Lager - Wenn das Regal Räder bekommt.* https://www.handelsblatt.com/unternehmen/logistik-spezial/amazon-automatisiert-lager-wenn-das-regal-raeder-bekommt/20057656-all.html. Version: Juli 2017

[3] SWISSLOG HOLDING AG: *Absolut Vodka, Schweden: Flexibel erweiterbare Automatisierung im Verteilzentrum.* https://www.swisslog.com/de-de/fallstudien-und-downloads/fallstudien-referenzen-kundenprojekte/2016/07/absolut

[4] THE LEGO GROUP: *LEGO® MINDSTORMS® Bedienungsanleitung.* https://www.lego.com/biassets/bi/4589649.pdf. Version: 2009

Anhang

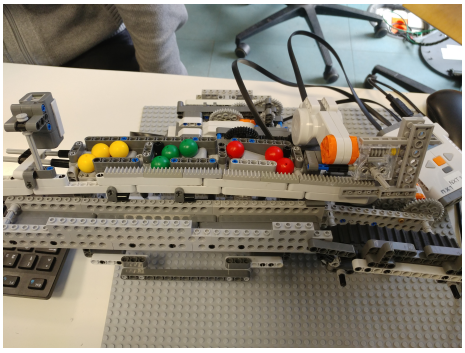


Abbildung 5. Seitliche Ansicht der fertigen Anlage. In der Mitte des Bildes die Fach-Wand (weiß mit aufgesetzten grauen Zähnen).

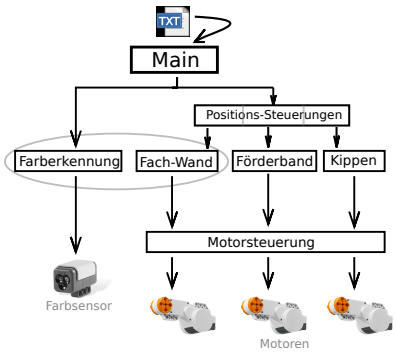


Abbildung 6. Diagramm der verwendeten Funktionen.

oaBricksle
Der knuffige LEGO-Roboter

Olivia Ley, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Ziel des LEGO-Mindstorms-Projektseminars war es, einen LEGO-Roboter aus dem NXT-Bausatz zu bauen. Dazu den programmierbare LEGO-'Stein', im Original 'Brick', mit MATLAB zu programmieren. Ein Zweck des Roboters war nicht vorgegeben und so entstand oaBricksle, der knuffige aber sinnlose (und auf Dauer etwas nervige) LEGO-Roboter.**

Knuffig und niedlich sollte er sein, da es in der Welt einfach zu wenig Knuffigkeit gibt. Nichts produktives zu können ist ein weiteres, häufiges Merkmal knuffiger Dinge wie z.B. junger Katzen und daher ein wichtiger Aspekt für die letztendliche Knuffigkeit.

Von da aus war es auch nur noch ein kleiner Schritt bis zur Nervigkeit und nun existiert das oaBricksle, der knuffige LEGO-Roboter, welcher fröhlich piepend durch den Raum fährt, Aufmerksamkeit verlangt und allein gelassen bald schlechte Laune bekommt und dann anfängt 'rumzuheulen'.

Natürlich freut er sich auch, wenn man ihn streichelt, er schnurrt dann sogar.

***Schlagwörter*—LEGO-Mindstorms, Roboter, niedlich, sinnlos, NXT**

I. Einleitung

DAS oaBricksle findet vor allem in privaten Haushalten Anwendung, aber auch in sozialen Umfeldern wie Kindergärten oder Altenheimen könnte er durchaus eine Bereicherung des Umfelds darstellen.

So soll er den Alltag seiner Besitzer erfreulicher und angenehmer machen, eine Bezugsperson bis zu einem gewissen Grad positiv emotional beeinflussen können und mehr Knuffigkeit in die Welt tragen. Es gibt bis zu einem Punkt bereits künstliche Intelligenz, die uns das Leben vereinfacht und angenehmer gestaltet, doch häufig werden auch Zweifel laut, ob wir nicht zu bequem werden, wenn wir so vieles den neuen Technologien überlassen.

Das oaBricksle schafft diesen Vorwurf durch seinen beständigen Hunger nach Aufmerksamkeit beiseite und verbessert mit seiner Knuffigkeit auch das Leben seines Besitzers. So ermöglicht er eine positive emotionale Bindung nicht nur für Personen, die sich wegen Allergien, der Hausordnung oder anderen Gründen kein Haustier anschaffen können, sondern auch für die, die einfach keine Tiere, dafür aber Knuffigkeit mögen.

An Knuffigkeit und schönen Dingen fehlt es dieser Welt und den Nachrichten an erfreulichen Tagesthemen. Die Inspiration zum oaBricksle fand sich im in Abbildung 1 dargestellten Webcomic.

DOI: 10.24352/UB.OVGU-2020-039

Abbildung 1. Die Idee, mit dem Roomba als Vorbild [1]

Der dort gezeigte Roboter ist an den Saugroboter Roomba angelehnt. So nach Aufmerksamkeit zu suchen, wurde zu einem Hauptzweck des oaBricksle.

Einem Wesen so einfach und schnell eine Freude bereiten zu können, löst positive Gefühle aus, welche wiederum wichtig für die Gesundheit sind. Denn wer sich gut fühlt, lebt damit auch ein kleines bisschen gesünder.

II. Vorbetrachtungen

In den Vorbetrachtungen werden die bereits vorhandenen Lösungen kurz vorgestellt und mit einer Literaturquelle belegt. Ebenso werden für die eigene Lösung genutzte Verfahren erklärt.

A. LEGO-Mindstorms

LEGO-Mindstorms ist eine programmierbare Reihe der LEGO Serie, die die Bausteine des LEGO-Technik Bausatzes verwendet.

Außerdem sind ein programmierbarer LEGO-Stein, 'Brick' im Englischen, Sensoren und Motoren sowie Kabel Teil des Systems.

Über eine eigene Umgebung von LEGO kann der Brick dann mittels Codebausteinen programmiert werden. Hier wurde die zweite Generation, der NXT genutzt.

Durch die Einbindung einer von der RTWH Aachen zur freien Verfügung gestellten Bibliothek kann der Brick auch über die Software MATLAB angesprochen und gesteuert werden.

Mittels entsprechender Befehle können damit die Sensoren abgefragt und Motoren angesprochen werden, wie es bei der originalen Software möglich ist.

B. Roombas und die Amöbenstrategie

Roomba ist der geschützte Name der Saugroboter der Firma iRobot [2].

Da vor allem das Prinzip der Funktion von Saugrobotern, beziehungsweise deren Fortbewegung durch die Wohnung, und weniger die Marke als Grundlage diente, ist dies also nur ein bekannteres Beispiel mit hohem Erkennungswert.

Im nachfolgenden Text wird der Name daher etwas weitgreifender benutzt.

Um einen Saugroboter durch die Wohnung zu leiten werden unterschiedliche und unterschiedlich effiziente Strategien genutzt [3]. Häufig werden mehrere Strategien kombiniert, um auch schwierigere Situationen simpel lösen zu können und Festfahren zu vermeiden.

So gibt es Strategien, die bestimmte vorgegebene Wege abarbeiten lassen, zum Beispiel die Mäander-Strategie [4].

Bei der Wandverfolgung wird wiederum eng mit der Sensorik zusammen gearbeitet, um den richtigen Weg zu finden [5].

Um solche Wege zu vereinfachen und den Ablauf effizienter zu gestalten, werden häufig gleich Karten der Räume und Hindernisse erstellt, auf die bei der nächsten Fahrt wieder zugegriffen werden kann.

Allerdings braucht ein knuffiger Roboter keine Effizienz.

Deshalb wurde hierfür auf das Prinzip der Zufallsfahrt zurückgegriffen. Die Zufallsfahrt wird in zwei Varianten unterschieden: der zeitabhängigen Random-Strategie [6] und der Amöben-Strategie [7].

Bei beiden wird beim Eintreten festgelegter Ereignisse ein zufälliger Richtungswechsel eingeleitet.

Beide Varianten nutzen einen Frontschild zur Kollisionserkennung, der die vordere Hälfte des Roboters abdeckt. Bei der Random-Strategie wird das Ereignis außerdem beim Erreichen einer zuvor festgelegten, maximalen Fahrtzeit ausgelöst.

Die Amöben-Strategie erhielt ihren Namen über das Muster der Spur die bei längeren Laufzeiten entsteht, welche bei der Draufsicht dem Aussehen einer Amöbe ähnelt.

III. Aufbau und Funktionen

Das Konzept der etwas nervigen, aber knuffigen Verhaltensweise wurde stufenweise aufgebaut und implementiert. Damit stehen auch, im Rahmen der technischen Umsetzbarkeit, beliebige Erweiterungen offen.

A. Fahren

Die grundlegende Eigenschaft des Konzepts war die eigene Fortbewegung im Raum, die etwa der eines simplen Staubsaugeroboters ähneln sollte.

Er sollte sich im Raum bewegen können und bei Zusammenstoß mit einem Hindernis, auch 'andotzen', die Richtung ändern.

Zum Vorwärtsfahren mussten beide Ketten, auf denen sich oaBricksle bewegen sollte, mit der gleichen, positiven Motorleistung betrieben werden. Eine negative Motorleistung lässt die Elektromotoren lediglich mit entsprechend angegebener Leistung rückwärts drehen.

Drehen wurde durch gegenläufige Ketten realisiert. Im Programmcode werden dafür die Ketten einmal mit der positiven und einmal mit der negativen Motorleistung angetrieben. Rückwärtsfahren war ohne Wenden zu müssen mit entsprechend negativer Motorleistung möglich.

Um Hängenbleiben an Hindernissen zu vermeiden, fährt er zunächst zurück, hält an und wendet dann. Das Halten zwischendurch ist erforderlich, um nicht zu viele Anweisungen zeitgleich an die Motoren zu senden und sie damit zu überfordern.

Andotzen ist die Kollision des Bumpers, der Frontschild des oaBricksle, mit einem Hindernis, wie einer Wand, und das damit verbundene Auslösen des Tasters.

Statt des Tasters welcher durch den damit verbundenen Bumper die gesamte Vorderseite abdeckt wäre auch ein Ultraschallsensor möglich gewesen. Die Möglichkeit wurde nicht weiter verfolgt, da die Ultraschallsensoren nur einen stark beschränkten Winkel und eine geringe Höhe erfassen und zudem nicht exakt genug arbeiten.

Anstatt das Umfahren von Objekten und Wegkommen von Wänden dann noch zu optimieren, wurde der Winkel, um den sich der Roboter neu ausrichtet, um einen Zufallsfaktor erweitert. Dieses etwas verschusselt und hilflos anmutende Verhalten ist häufig bei knuffigen Wesen anzutreffen.

B. Piepen / Akustische Rückmeldung

Es gibt wenig Dinge, die als niedlicher empfunden werden als ein maunzendes Kätzchen, ein freudig bellender Welpe, oder ein glücklich piepender Roboter - wie in diesem Fall.

Außerdem ist es eine hervorragende Möglichkeit, den Roboter zurückgeben zu lassen, was er gerade tut, ohne auf die Ausgabe in der Befehlszeile achten zu müssen.

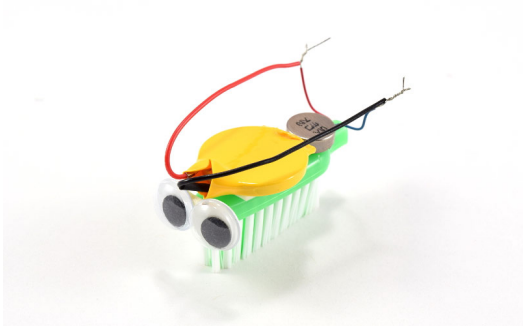


Abbildung 2. Ein Bristlebot oder Zahnbürstenroboter [8]

Vom Konzept her sollte er also ab und zu ein fröhliches Pfeifen von sich geben. Über die Angabe von Frequenz und Länge sowie Pausen dazwischen wurden einige eingängige, längere und kürzere Melodien umgesetzt und getestet.

Dabei schnitten kürzere, eigens geschriebene Stücke deutlich besser ab, da sie keine bereits bestehenden, negativen Assoziationen wach rufen.

Neben dem Grundzustand des normalen Fahrens, der von einem fröhlichen Dreiton begleitet wurde, gab es auch ein tieferes Brummen, das etwa einem Brummeln oder Schnurren (dazu später noch mehr) entsprach.

Um dem Ganzen einen nicht ganz erzwungenen Charakter zu verleihen, hätte nun wieder ein Algorhythmus geschrieben werden können.

Da aber einerseits Auslöser für solche Reaktionen nicht ganz klar sind und andererseits schwieriger sowie zeitaufwändiger umzusetzen wären, wurde auch hier wieder eine Zufallsfunktion dafür implementiert.

C. *Schnurren / pwrr-o-Mat*

Viele Bestandteile von oaBricksle, für die der pwrr-o-Mat ein exzellentes Beispiel darstellt, haben etwas verwirrende Namen. Diese sind ebenso wichtiger Bestandteil des Konzeptes.

So lässt sich der programmierbare NXT-'Brick' auch über Bluetooth mit einem PC verbinden und kann dann angesteuert werden.

In Heimarbeit beim Basteln stellt das noch kein Problem dar. Allerdings wird man bei einer größer konzipierten Veranstaltung wie dieser schnell erkennen, dass einheitlich benannte Geräte fatal hinsichtlich des Wiedererkennungswertes für einzelne Geräte sind.

Kurz: Welcher NXT, bei dem Bluetooth aktiviert war, war nun der Richtige?

Über die hauseigene Software von LEGO ließ sich der Brick dann am Computer über Kabel umbennen.

Im Sinne der knuffigen Namensgebung sollte der Brick dann den Namen Bricksley tragen, was sich jedoch als ein Zeichen zu viel erwies. Für den Namen waren nur acht Zeichen vorgesehen.

Und da der Name 'Bricksle' bereits etwas nach bayrischem Dialekt klang, wurde als Variable für die Verbindung dann 'oaBricksle' genommen. Abgesehen vom Nutzen als Namen des Roboters hat die Bezeichnung also tatsächlich auch einen Nutzen im Code.

Ursprünglich war geplant, das Schnurren, also eine Vibration über das Grundprinzip der Funktionsweise eines einfachen Roboters auf Borsten zu nutzen.

Diese Roboter werden in der simpelsten Ausführung auch Bristlebot oder Zahnbürstenroboter genannt.

Sie bestehen dabei lediglich aus dem abgeschnittenen Kopf einer Zahnbürste, einer Knopfzelle und einem Vibrationsmotor. Ein Beispiel ist in Abbildung 2 zu sehen, es wurden außerdem Verzierungen angebracht.

Dabei kann die Vibration auch durch einen beliebigen anderen Motor erzeugt werden. Dafür müssen lediglich die Umdrehungen pro Minute der Rotationsachse hoch genug sein und mit einen asymmetrisch daran platzierten Schwungstück eine Unwucht erzeugt werden, die dann die Vibration hervorruft.

Dieser häufig unerwünschte Effekt zerlegt allerdings über einen längeren Zeitraum ziemlich zuverlässig starre Systeme.

Ein nicht zum LEGO Bauset gehörender Motor war samt Batterien und Schwungstück vorhanden.

Das Einschalten und Einbauen erwies sich allerdings als nur schwer umsetzbar und es standen noch viele Zahnräder und ein dritter Motor zur Verfügung.

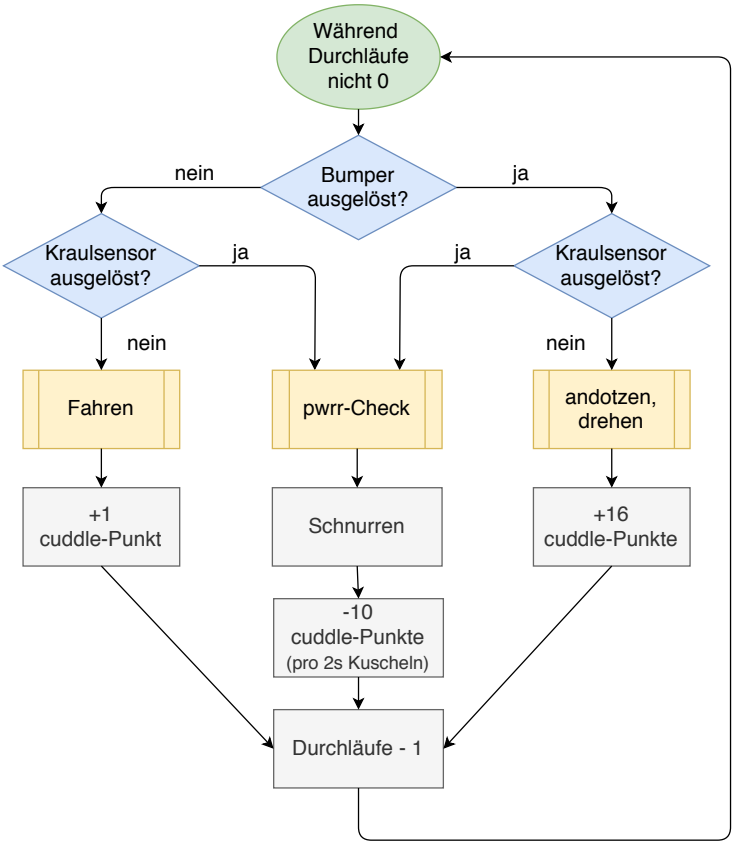


Abbildung 3. Programmablaufplan zur Unterscheidung von Fahren, Kollision und Kraulen

Summa summarum dreht beim Schnurren jetzt ein etwas wackliger Propeller schnell (jedoch ohne viel Kraft dahinter) und bringt den oaBricksle fühl- und hörbar zum Schnurren.

D. Cuddle-Meter / die Persönlichkeit

Zum Konzept des knuffigen Roboters gehört auch, dass er wütend wird, beziehungsweise anfängt 'rumzuheulen'. Dies wird ausgelöst durch Vernachlässigung oder wenn ihm schlimme Dinge passieren, wie zum Beispiel die Kollision mit einer Wand. Diese Grundaggressivität ließ sich am Besten über eine Geschwindigkeitserhöhung realisieren.

Allerdings brauchte es dazu ein Gedächtnis, hier das Cuddle-Meter, das den Kuschelwert angibt und damit ein Maß für die Unzufriedenheit ist. Das Cuddle-Meter ist jeweils die Gesamtsumme der cuddle-Punkte am Ende eines Durchlaufs.

Die Anzahl der Durchläufe wird beim Start des Programms festgelegt.

Nach jedem erfolgten Durchlauf wird die Anzahl der verbleibenden Durchläufe um eins verkleinert, bis das Programm schließlich bei null Durchläufen endet. Damit wird die gesamte Laufzeit begrenzt. Zu Testzwecken wurden üblicherweise 300 Durchläufe genutzt.

Für jeden Durchlauf wird dabei zunächst entschieden, ob oaBricksle gerade gegen ein Hindernis gefahren ist oder nicht. Nach jedem per Kollision erreichten Hindernis wendet oaBricksle, piept enttäuscht und das Cuddle-Meter wird für diese Runde um 16 Punkte erhöht.

Wenn kein Hindernis gefunden wurde, fährt er eine Runde normal weiter und das cuddle-Meter wird um einen Punkt hochgezählt. Die Beträge, um die das cuddle-Meter verändert wird, sind willkürlich gewählt, sie stellten sich nach mehreren Testläufen als geeignet heraus.

Um eine höhere Priorität zu erreichen wird der Kraulsensor, wie in Abbildung 3 zu sehen ist, erst nach dem Bumper abgefragt.

Statt beide Sensoren nacheinander abzufragen sind so die Reaktionszeiten bei gleichzeitig weniger Durchläufen kürzer. Der Code benötigt dann weniger Ressourcen.

Wird der Kraulsensor nicht ausgelöst, werden die normalen Aktionen weiter ausgeführt. Andernfalls wird die Funktion pwrr-Check aufgerufen. Diese aktiviert den pwrr-o-Mat, den Schnurrmotor von oaBricksle und sorgt für gelegentliches zufriedenes Brummeln.

Außerdem sorgt diese Funktion dafür, dass cuddle-Punkte für das Kraulen vom cuddle-Meter abgezogen werden. Für je zwei Sekunden Kraulen sinkt es um zehn Punkte. oaBricksle wird wieder zufriedener.

E. Aufregen / furity

Anhand des cuddle-Meters lässt sich erkennen, wie aufgeregt oaBricksle zu welchem Zeitpunkt ist. Um das auch äußerlich umzusetzen, berechnet die Funktion furity die momentane Geschwindigkeit anhand des cuddle-Meters.

Dafür wird ausgewertet, in welchem Bereich die cuddle-Punkte sich gerade befinden und die Grundgeschwindigkeit wird entsprechend angepasst. Damit wird oaBricksle dann nach mehrmaliger Kollision mit Hindernissen schneller werden, durchquert damit wiederum schneller den Raum und gelangt dadurch schneller zum nächsten Hindernis.

Durch die nächste Kollision steigt das cuddle-Meter wieder, oaBricksle wird noch schneller und aufgeregter und in überschaubarer Zeit ist er bei der maximalen Geschwindigkeit angekommen. Ein Kreislauf, von dem er nur zeitweise durch Kraulen abgebracht werden kann.

IV. Zusammenfassung und Fazit

Aus einer Idee, viel LEGO, Code und Tee entstand in den zwei Wochen des Seminars die erste voll funktionsfähige Version des knuffigen Roboters oaBricksle. Er kann sich eigenständig durch einen Raum bewegen, Hindernisse umfahren und einfache Interaktionen mit Menschen durchführen sowie seine momentane Laune ausdrücken.

Verbesserung ist hinsichtlich der knuffigen Optik ebenso notwendig, wie bessere und ausgefeiltere Sensorik. Dadurch lassen sich dann noch mehr Interaktionsmöglichkeiten einbinden.

Die im LEGO-Bausatz enthaltene Sensorik waren doch sehr begrenzt und simpel gehalten.

Beispielsweise könnte die Stimmung besser über ein Display angezeigt, eine Sprach- und Bilderkennung für Unterhaltungen und ein Belohnungssystem eingefügt und mehr Speicherplatz für komplexere Sprachausgaben eingefügt werden.

Das noch recht simple Verhalten lässt Raum für weitere Feinheiten und komplexere, ja sogar unterschiedliche Persönlichkeiten.

Vielleicht hängt ja der eine oaBricksle der nächsten Generation gerne in staubigen Ecken ab, während ein anderer gerne Kissen erklimmt und ein weiterer sich vielleicht hungrig über die auf dem Fußboden liegenden Krümel der letzten Mahlzeit her macht?

Der Kreativität sind ja bekanntlich keine Grenzen gesetzt, und was sich vielleicht doch alles umsetzen lässt, wird sich zeigen.

Begeisterte Nachfragen gab es bereits einige.

Anhang

Literaturverzeichnis

[1] STARFLEETRAMBO ON TUMBLR: *Webcomic: ok but imagine a roomba...* https://starfleetrambo.tumblr.com/post/171767831093/starfleetrambo-starfleetrambo-ok-but-imagine-a. Version: März 2018

[2] IROBOT GERMANY GMBH: *Roomba Saugroboter*. https://www.irobot.de/roomba. Version: März 2020

[3] U.FRITZ: *Saugroboter auf Zufallsfahrt*. http://saugrobot.de/bewegung-zufallsfahrt.php. Version: März 2005-2012

[4] U.FRITZ: *Mäander-Fahrstrategie*. http://saugrobot.de/saugroboter-maeander.php. Version: April 2005-2012

[5] U.FRITZ: *Wandverfolgung*. http://saugrobot.de/an-der-wand-entlang.php. Version: April 2005-2012

[6] U.FRITZ: *Zufalls-Fahrstrategie*. http://saugrobot.de/random-fahrstrategie.php. Version: März 2005-2012

[7] U.FRITZ: *Amöben-Fahrstrategie*. http://saugrobot.de/amoeben-fahrstrategie.php. Version: März 2005-2012

[8] SCIENCE BUDDIES: *bristlebot-googly-eyes*. https://cdn.sciencebuddies.org/Files/7937/15/bristlebot-googly-eyes.jpg. Version: März 2020

oaBricksle – Ein knuffiger Roboter

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

***Zusammenfassung*—In der Woche vom 17.02. bis 13.02. wurde oaBricksle erschaffen. Dieser LEGO Mindstorms Roboter wurde mit MATlab programmiert und soll die knuffige Seite von Robotern zeigen. Er findet seinen eigenen Weg und verlangt nach einer gewissen Zeit nach Aufmerksamkeit. OaBricksle wurde als etwas tolpatschig programmiert und macht auf sich aufmerksam, wenn er gestreichelt werden möchte.**

***Schlagwörter*—Knuffigkeit, Lego Mindstorms, Roboter, Roomba, Schülerpraktikum**

UM dem Defizit von Knuffigkeit entgegen zu wirken war es das Ziel einen Roboter zu entwickeln, der niedlich und tolpatschig ist. Dieser Roboter soll sich ähnlich wie ein Haustier verhalten und Knuffigkeit in den Alltag von vielen Menschen bringen, jedoch auf eine neue Art und Weise. Diese Entwicklung soll sich in der Wohnung des Besitzers bewegen und mit den Anwesenden auf eine knuffige Art und Weise inteagieren können. So soll er nach Aufmerksamkeit verlangen und Ärger machen, wenn er diese nicht bekommt. Der Roboter wurde nach der Vorlage eines Roombas konstruiert und sein Verhalten wurde durch einen Comic aus dem Internet inspiriert [1].

I. Einleitung

A. Inspiration

Bei der Ideenfindung gab es 2 größere Einflüsse, die dieses Projekt beeinflussten. Einerseits den im Juli 2018, von James Vincent, veröffentlichten Artikel „This sun-chasing robot looks after the plant on its head“ [2], andererseits durch einen Webcomic [1]. So kamen die Idee, eines elektronischen Begleiters, der im Gegensatz zu vielen anderen Robotern, sich knuffig und liebenswürdig verhält, zustande.

B. LEGO Mindstorms

LEGO Mindstorms ist ein Teilgebiet von LEGO Technik. Es dreht sich alles um den NXT, einen etwas größeren LEGO-Stein. Der NXT ist ein Microcontroller der 3 Slots für Motoren und 4 Slots für Sensoren besitzt. Er kann über Bluetooth angesteuert und programmiert werden. Außerdem ist es möglich über den NXT Sounds auszugeben. Während des Projekts wurde der NXT mit MATLAB, anstatt der LEGO Software, programmiert und es wurde mit den gegeben LEGO Technik Bausteinen gearbeitet.

C. Der Name oaBricksle

Bei der Recherche nach Ideen für dieses Projekt, im Rahmen des LEGO Praktikums, fiel auf, dass im Englischen der NXT der LEGO Mindstorms „Brick“ genannt wird, weil dieser einem Ziegelstein sehr ähnlich aussieht. Bei dem späteren Versuch den Nxt über Bluetooth anzusteuern wurde festgestellt, dass alle NXT als „NXT“ bezeichnet und somit nicht voneinander unterscheidbar, waren. Um den NXT in Zukunft von den Anderen auseinanderhalten zu können sollte er zu „Bricksley“umbenannt werden, in Anlehnung an „Brick“. Jedoch ist dieser Name einen Buchstaben zu lang und es passte nur „Bricksle“ in das Namensfeld. Es wurde eine bayrische Tendenz erkannt und seitdem heißt der Roboter „oaBricksle“.

II. Hauptteil

Die 2 Wochen des Praktikums starteten mit einer Einführung in MATLAB. Mit diesem Programm sollte dann auch das Projekt programmiert werden. Die Projekte wurden in Zweiergruppen umgesetzt. Die Projektgruppe des „oaBricksle“ bestand aus Olivia Ley und Tim Werner. Olivia Ley übernahm größtenteils die Programmierung, während Tim Werner den Roboter gebaut hat. So entstand oaBricksle, ein Roomba-ähnlicher Roboter der auf eine gewisse Weise mit seiner Umwelt inteagieren und dabei einen Gefühlszustand wiederspiegeln, wodurch er auf eine niedliche Weise den Alltag verbessern soll.

A. Sensorik

An der Front des oaBricksle befindet sich ein Kollisionserkennungsmodul. Dieses besteht aus einer Platte, die, bei Kontakt mit einer Wand, einen Taster betätigt. Ein weiterer Taster befindet sich an der oberen Seite des oaBricksle. Dieser ist für die Erkennung des Kraulens zuständig. Zum Kraulen wird eine Deckplatte oberhalb des Roboters eingedrückt.

B. Aktorik

Die Aktorik beim oaBricksle ist der wichtigste Teil des Roboters, da diese für Knuffigkeit zuständig ist. Jedoch durch die Begrenztheit des NXT, mit nur 3 Slots für Motoren, muss diese auch simpel sein. Am NXT sind alle drei Motoren verbaut und es können Sounds ausgegeben werden. Es ist je ein Motor pro Kette verbaut. Der dritte Motor funktioniert als „Schnurrmotor“. Die Motoren an den Ketten sind dafür zuständig, dass sich beide Ketten unabhängig voneinander bewegen können und somit eine Bewegung in alle Richtungen, als auch eine Drehung ohne Wendekreis, möglich ist. Der „Schnurrmotor“ ist mit einer Unwucht ausgestattet. Dies sorgt dafür dass, wenn er aktiv ist, der gesamte Roboter vibriert und somit ein Gefühl simuliert wird, welches dem Schnurren einer Katze nahe kommt.

C. Programmierung

Das Grundgerüst des Programms ist eine Schleife. Diese Schleife hat einen bestimmten Wert an Durchläufen der jedes mal vor dem Start des Programmes angegeben werden muss.

DOI: 10.24352/UB.OVGU-2020-040

Dieser Wert gibt an wie häufig die Schleife durchlaufen wird und bestimmt somit wie lange sich oaBricksle durch die Gegend bewegt. Während oaBricksle aktiv ist fährt er solange vorwärts bis er entweder gestreichelt wird oder eine Kollision stattfindet. Bei einer Kollision fährt er kurz Rückwärts und dreht sich in eine zufällig ausgewählte Richtung in einem zufällig ausgewählten Winkel. Die Winkel und die Richtung sind zufällig, weil oaBricksle tolpatschig wirken soll, so ist es möglich dass er sich in einer Ecke lange aufhält oder immer wieder gegen das gleiche Hinderniss fährt. Nachdem oaBricksle sich gedreht hat, fährt er weiter gerade aus. Falls er gestreichelt wird, werden alle momentanigen Aktionen abgebrochen und er fängt an zu „Schnurren“, d.h. der Schnurrmotor fängt an sich zu drehen.

D. *Cuddle-Meter*

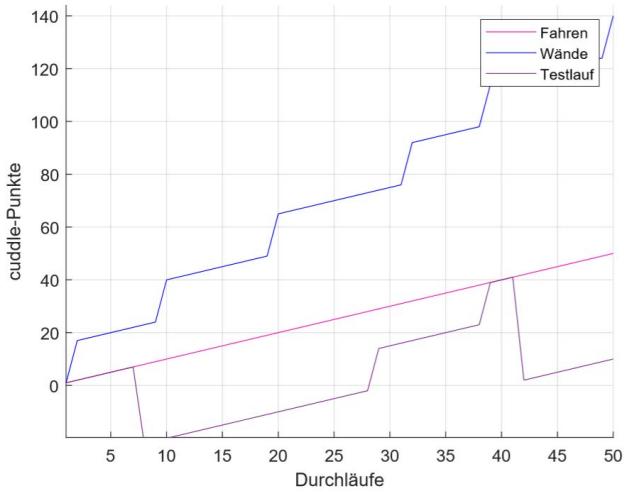


Abbildung 1. Cuddle-Meter Testläufe

Um einen „Gefühlszustand“ wiederspiegeln zu können gibt es das „Cuddle - Meter“. Je nach Stand des „Cuddle - Meters“ wird oaBricksle wüternder oder ruhiger. Der Wert des „Cuddle - Meters“ steigt mit der Zeit konstant langsam an, kommt es jedoch zu einer Kollision so hüpft der Wert ein größeres Stück. Damit der Wert wieder sinkt muss man oaBricksle kraulen. Das „Cuddle - Meter“ hat direkten Einfluss auf das Programm, indem die Geschwindigkeit mit der sich der Roboter bewegt mit dem Wert des „Cuddle - Meters“ verändert. Bei einem höhren Wert bewegt sich oaBricksle schneller. Dies führt zu verschiedenen Effekten zum Beispiel ist es möglich dass vorherige Hindernisse einfach vom Roboter weggeschoben werden, da die LEGO Sensoren nicht schnell genug reagieren können. Außerdem sind die Motoren viel lauter durch die höhere Geschwindigkeit und deuten so einen wütenderen oaBricksle an. Beim Kraulen sinkt der Wert des „Cuddle - Meters“, wodurch er sich wieder langsamer bewegt. Es ist möglich dass, das „Cuddle - Meter“ einen negativen Wert erreicht. Sollte dies der Fall sein, bewegt sich oaBricksle bis zu einem Minimum lansgamer als normal.

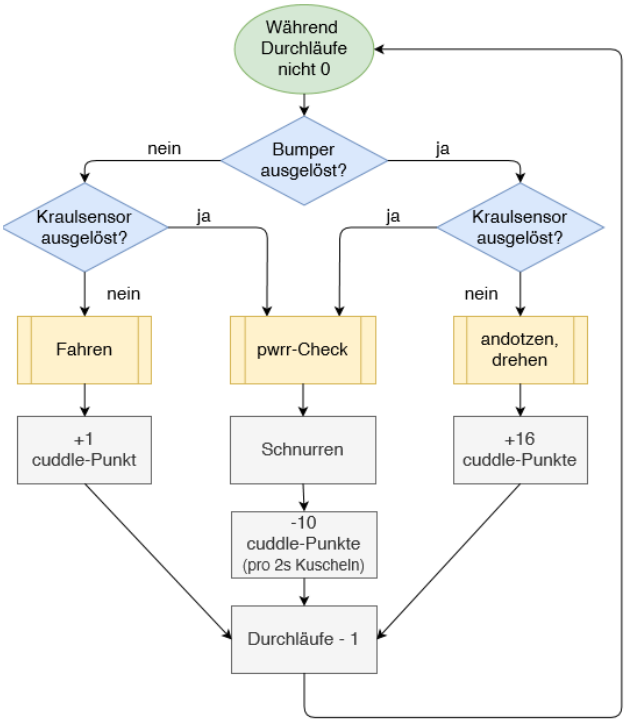


Abbildung 2. Ablaufplan

III. Zusammenfassung und Fazit

In den zwei Wochen des Projekts wurde der erste Prototyp des oaBricksle angefertigt. Er kann sich problemlos durch ebenes Gebiet bewegen und einfache Interaktionen mit der Umwelt sind möglich. Jedoch muss an der Niedlichkeit noch gearbeitet werden, noch ist oaBricksle nicht mehr als ein Kasten auf Ketten. Außerdem fehlen noch weitere Interaktionsmöglichkeiten mit Menschen, die aufgrund der Begrenztheit des NXT, hinsichtlich der Anzahl von Sensoren und Aktoren, nicht umgesetzt werden konnten. Wichtig sind also noch ein niedlicheres Aussehen, leisere Motoren, ein besserer Microcontroller und eine eventuelle grafische Ausgabe des Gefühlszustandes. Da oaBricksle bisher auch noch nichts Produktives leisten kann, soll er in Zukunft noch mit einem Staubsauger austattbar sein, oder eine Pflanze tragen können wie [2] „Hexa“ von der Firma Vincross.

Literaturverzeichnis

[1] Starfleetrambo: *no title*. https://starfleetrambo.tumblr.com/post/171767831093/starfleetrambo-starfleetrambo-ok-but-imagine-a. Version: March 2012

[2] Vincent, James: *This sun-chasing robot looks after the plant on its head*. https://www.theverge.com/2018/7/12/17563688/robot-plant-hybrid-hexa-vincross-succulent. Version: July 2018

Anhang

Abbildung 3. Webcomic von Starfleetrambo

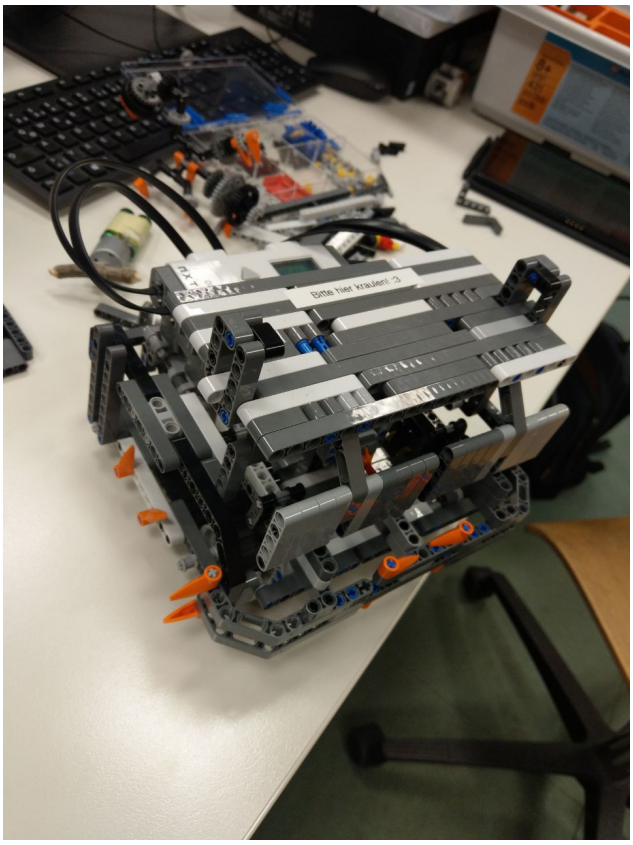


Abbildung 4. Aussehen des ersten oaBricksle Prototyps

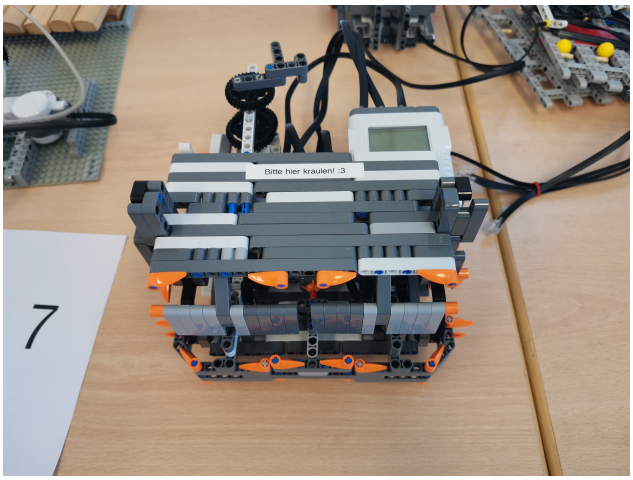


Abbildung 5. oaBricksle

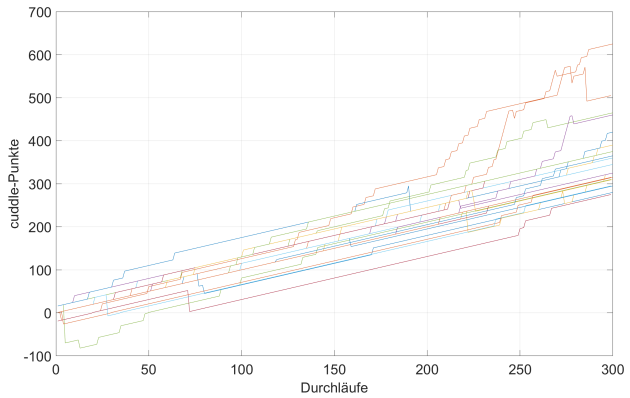


Abbildung 6. Testlaufbeispiele

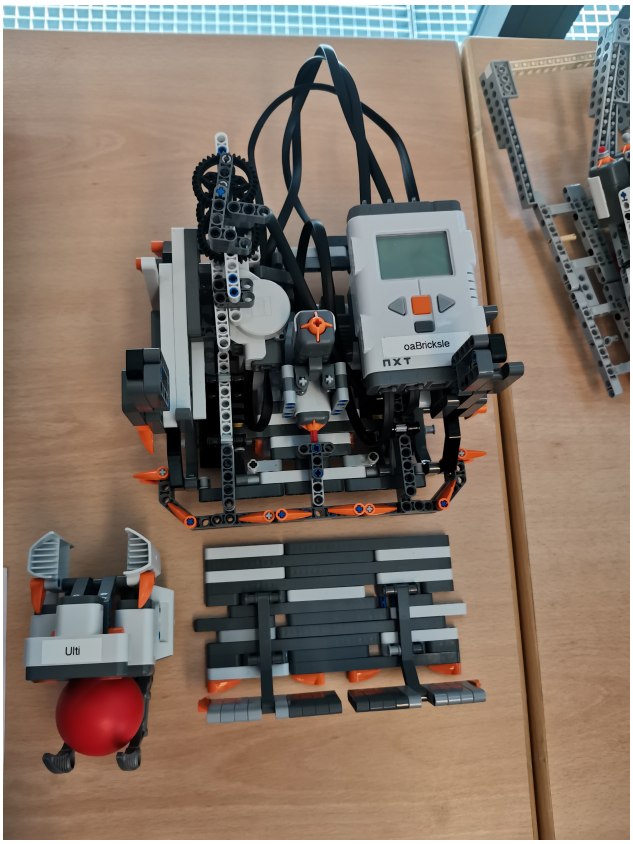


Abbildung 7. Kraulsensor (nach oben gerichtet), Kollisionssensor (parallel zum Boden) und Schnurrmotor (links neben Kraulsensor)

Der Xylophon-Spieler

Duc Manh Pham, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In diesem Projekt geht es um den Bau eines Roboters, dessen Aufgabe es war, Melodien zu spielen. Am Ende des Projektseminars ist es gelungen, ein Roboter mithilfe Lego-Bausteine zu bauen. Der in MATLAB programmierte Roboter war in der Lage, einfache Melodien auf einem Xylophon fließend zu spielen.**

***Schlagwörter*—Lego Mindstorms, MATLAB, Melodien, Roboter, Xylophon.**

I. Einleitung

MENSCHEN haben Musik und Musikinstrumente seit tausenden Jahren gespielt. In Prinzip kann jeder Gegenstand, der Töne oder auch nur Geräusche hervorbringt, als Musikinstrument dienen. [1]

Das Xylophon gehört zu den klassischen Musikinstrumente. In vielen außereuropäischen Musikkulturen wie Asien und Afrika nehmen Xylophone eine wichtige Stellung ein. Vom klassischen Xylophon wurden zahlreiche Arten von Xylophonen weltweit verbreitet. Zum Beispiel: das Balophon in Westafrika, die Marimba in Guatemala, das Trogxylophon pattala in Myanmar, usw. [2]

Roboter und Musikinstrumente haben einen Zusammenhang. Die sind alle menschliche Erfindungen. Man kann einen Roboter konstruieren, um einem bestimmten Zweck zu dienen. Wenn der Zweck ist, Musik zu machen, sollte das nicht unmöglich sein. Trotzdem tauchten viele Probleme während des Baus auf. Der erste Prototyp entstand nach einer Woche Arbeit (siehe Abbildung 1). Dabei sind einige Anforderungen entstanden.

1. Änderung der Positionierung: Der Roboter stand auf dem Tisch ohne feste Position. Der Abstand zwischen Roboter und Xylophon war nicht nahezu konstant. Außerdem wurde der Schläger noch nicht mit dem Oberteil des Roboters befestigt. Es war erforderlich, dieses Problem zu lösen, sonst war der Roboter nicht funktionsfähig.

2. Zuordnung der Tonhöhe zur Winkelposition des Motors: Die Abweichung war relativ groß. Die Töne konnten nicht richtig gespielt werden. Die Ansteuerung war unstabil und ungenau. Der Grund dafür war der NXT-Motor. Der hat bei jedem Befehl etwas falsch durchgeführt. Zum Beispiel: Der Motor sollte sich 90 Grad umdrehen, drehte sich aber einmal um 88 Grad, ein anderes Mal um 92 Grad. Es führte dazu, dass die falschen Klangstäbe geschlagen wurden.

3. Stabilität: Der ganze Roboter hat stark gewackelt, wenn er schnell gedreht wurde. Die zwei Motoren vibrierten viel bei der Ausführung. Man muss den Roboter umbauen, um das Wackeln zu minimieren.

4. Der Roboter sollte nicht nur einzelne Noten abspielen, sondern auch beispielsweise eine ganze Melodie darstellen. Das war eine Anforderung des Programmierens, weil der Roboter einige Prinzipien wie Tonlängen und Tonhöhen verstehen musste.

DOI: 10.24352/UB.OVGU-2020-041

Abbildung 1. Erster Prototyp

II. Vorbetrachtungen

A. Positionierung

Die Probleme mit der Positionierung, die beim ersten Prototyp entstanden sind, wurden in der letzten Version gelöst. Das ganze System von Xylophon und dem Roboterarm wurde mit Lego-Bausteinen auf einer großen flachen Platte festgemacht. Die angemessene Position des Xylophons wurde markiert. Dadurch wurde der Abstand nahezu konstant gehalten. Der Oberteil des Roboterarms wurde neu gebaut, damit sich der Schläger zusammen mit dem Arm bewegte. Das heißt, der Schläger dreht mit gleichem Winkel wie der Roboterarm. Das verhindert, dass sich der Schläger manchmal nicht mit bewegt oder wackelt.

B. Zuordnung der Tonhöhen

Bei der ersten Version konnte der Roboter schon einzelne Note spielen. Aber der Versuch eine Oktave vorzuführen ist noch nicht gelungen, weil die Abweichung nach jeder Drehung zunimmt. Wenn der Arm von C zu C drehen und bei jeder Note einmal schlagen sollte, drehte er von C bis E in der nächsten Tonleiter. Außerdem konnte der Roboter nicht wissen, wo er gerade war. Bei jeder Ausführung musste er zurück zu C per Hand getrieben werden.

Es gab in der letzten Version einige Verbesserungen. Die Genauigkeit stieg deutlich im Vergleich mit dem ersten Prototyp. Der Arm konnte schon zu irgendeinem Klangstab kommen, egal wo er vorher war. Zu Beginn jeder Ausführung ist eine Initialisierung erforderlich, aber sie erfolgt mit drei einfachen Tasten.

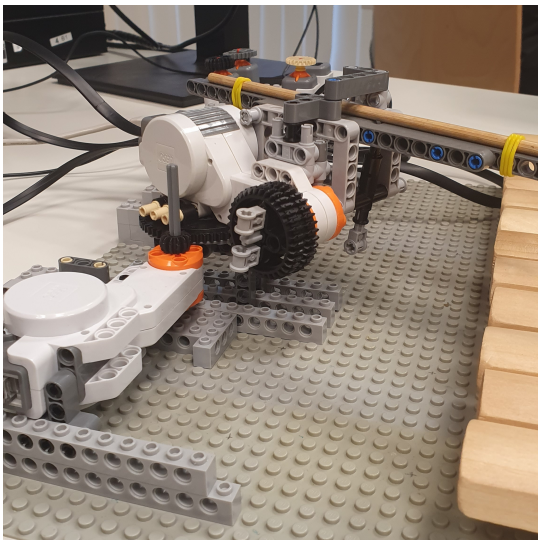


Abbildung 2. Umbau mit einem Drehkranz

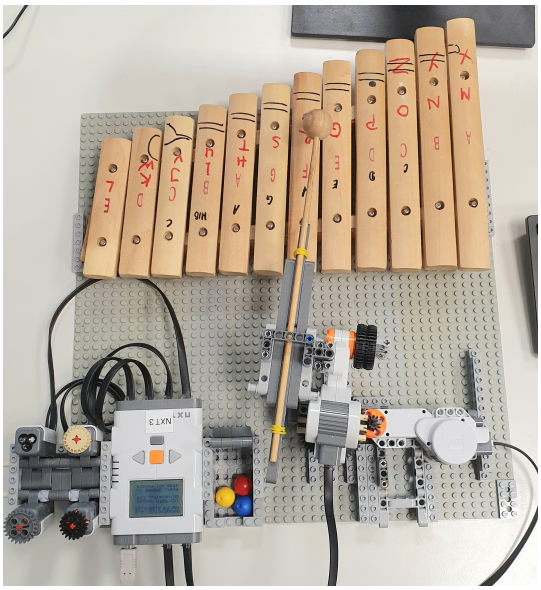


Abbildung 3. Letzte Version

C. Stabilität

Dank eines Videos über ein ähnliches Projekt [3]und der Hilfe des Betreuers konnte der Aufbau des Roboters verbessert werden. Dadurch stieg auch die Stabilität. Ein Drehkranz wurde zwischen zwei Motoren eingefügt, um das Übersetzungsverhältnis zu vergrößern. Das heißt, der in horizontale Richtung bewegende Motor muss um einen größeren Winkel drehen, um das Ziel zu erreichen. Die Abweichung nach mehreren Drehungen wurden dadurch verkleinert (siehe Abbildung 2).

D. Programmieren

Das Programm zur Bedienung des Roboters wurde in Matlab geschrieben. Es wurde dann im Gerät Mindstorms NXT 2.0 übertragen und ausgeführt. Eine zusätzliche Benutzeroberfläche wurde auch programmiert, damit der Roboter einfacher bedient wurde. Ganze Melodien darzustellen war für ihn machbar, da ein Farbsensor angebaut wurde. Je nach Erkennung eine Farbe wurde die entsprechende Melodie abgespielt. Es gab drei verschiedene Melodien zur Auswahl.

Es ist gelungen, die Prinzipien der Musik im Programm mitzubringen. Nicht nur Noten, sondern auch Tonhöhe und ihren Winkelpositionen wurden berücksichtigt und definiert. Eine Korrektur wurde auch hinzugefügt, um einige Fehler bei der Ausführung zu verringern.

III. Realisierung

A. Technischer Aufbau

Wie in Abbildung 3 zu sehen ist, handelt es sich um drei Bereiche. Im oberen Bereich befindet sich ein Xylophon. Das ist ein aus Holz ganz normales Xylophon zum Spielen. Es gibt zwölf Tonhöhen zur Verfügung, aber der Roboterarm konnte nur von C bis C in der nächsten Tonleiter eintreffen. Der Grund dafür ist, damit der Schläger in eine Umlaufbahn fährt, die ähnlich wie ein Kreis ist. Außerdem besteht es einen Vorteil, dass der Winkel zwischen zwei nebeneinander Tonhöhe immer einen bestimmten Wert hat. Die Position des Xylophons wurde mit einigen Lego-Bausteine markiert, damit der Wert nicht verändert wurde.

Im unteren linken Bereich sind Plätze für das NXT-Gerät, einen kleinen Kasten mit drei Kugeln und ein Bedienfeld. Das NXT-Gerät wurde mit zwei NXT-Motoren und vier Sensoren verbunden. Alle Programme zur Vorführung des Roboters laufen in einem Rechner. Die Signale oder Befehle wurden dann über ein USB-Kabel zum NXT-Gerät gesendet. Danach wurde alles ausgeführt. Der kleine Kasten war da, um die farbige Kugeln zu lagern und um den linken Bereich auf der Platte zu befestigen. Es gibt im Bedienfeld vier Sensoren: drei Tastsensoren und ein Farbsensor. Der Farbsensor dient dazu, dass man verschiedene Melodien für den Roboter auswählen kann, welche er dann abspielt. Drei Tastsensoren sind für den Handantrieb da. Wird die graue (schwarze) Taste gedrückt, bewegt sich der Arm nach links (rechts). Wird die gelbe Taste gedrückt, dreht der in vertikale Richtung bewegende Motor. Dieses Bedienfeld dient zur Initialisierung. Zu Beginn jeder Ausführung muss der Roboterarm zur Klangplatte C ausgerichtet werden. Der Kugel auf dem Schläger muss genau mittig auf C angeordnet werden. Der Stab, der mit dem zweiten Motor verknüpft wurde, muss senkrecht nach oben zeigen. Je genauer diese Schritte durchgeführt werden, desto besser wird das Ergebnis. Um die Initialisierung zu beenden, werden die graue Taste und die schwarze Taste gleichzeitig gedrückt.

Im rechten Bereich platziert sich der Roboter. Er wurde nach mehreren Prototypen verbessert. Solche Probleme wie Wackeln und Erschütterung kommen selten vor, da er ziemlich fest auf der Platte steht. Der Roboter basiert auf zwei Motoren. Der einzige Zweck ist, dass der Schläger so frei wie möglich in horizontale und vertikale Richtung gedreht werden kann. Beim ersten Prototyp wurden zwei Motoren direkt miteinander verbunden. Wird der Wert von „Tacho Limit“ gleich 10 im Matlab-Programm gespeichert, wird der Schläger genau 10

Grad gedreht. Da der Abstand zwischen zwei Tonhöhen relativ klein ist, passierte es oft so, dass falsche Klangstäbe geschlagen wurden und die Melodie verdorben wurde. Dieses Problem wurde großenteils gelöst, indem man ein System mit zwei Zahnräder gebaut hat. Das größere Zahnrad, nämlich der Drehkranz, wurde sich an dem zweiten Motor angelagert. Das kleinere Zahnrad fügte man an den ersten Motor über einen Stab an. Das hat eine Wirkung, dass der Schläger deutlich kleinerer Winkel drehen lässt, als den Drehwinkel des ersten Motors. Dadurch wird die Chance, dass falsche Klangstäbe geschlagen werden, geringer. Zusammenfassend lässt sich sagen, dass die Spiel-Genauigkeit besser wird.

B. Programmablauf

In der Abbildung 4 wird gezeigt, wie das Hauptprogrammm geschrieben wurde. Zuerst muss der Schalter umgelegt werden. Dann ist eine Initialisierung erforderlich. Das macht man mit Tasten am Bedienfeld. Wenn die Initialisierung erfolgt, kommt danach die Songauswahl. Wird eine Farbkugel am Farbsensor gebracht, beendet die Auswahl. Eine Nachricht wird auf dem Bildschirm angezeigt und die dazugehörige Melodie wird abgespielt.

Die Abbildung 5 legt den Ablaufplan des Programm dar, welches den Farbsensor regelt. Hier wurden Switch-Verzweigungen verwendet, die dann entschieden, welche Melodie zur Vorführung gewählt wurde. Aus zeitlichen Gründen konnten nur drei Melodien programmiert werden. Rote Kugel war für den Song „Summ summ summ“, blau für „Alle meine Entchen“ und gelb für „Twinkle twinkle little star“.

C. Logische Steuerung

Es ist gelungen, die Ideen bei der Kick-Off-Präsentation im Programm mitzunehmen. Die Töne und Tonlängen wurden in zwei Felder gespeichert (siehe Anhang). Der Abstand zwischen zwei nebeneinander Klangstäbe ist 11 Grad. Mit Faktor drei des Übersetzungsverhältnisses ist der Abstand der Töne verdreifacht (33 Grad). Jede einfache Melodie enthält Viertelnote, halbe Note und ganze Note. Die Dauer der Viertelnote ist 45 ms, halbe Note ist zweimal Viertelnote und ganze Note ist gleich zweimal halbe Note. Durch Winkelpositionen des Schlägers wurden Töne definiert. Eine Korrektur wurde auch hinzugefügt. Wenn der Schläger einen langen Weg fahren muss, wie beispielsweise die Länge von vier Klangstäben, dann muss er noch länger als üblich fahren, damit er das Ziel erreicht. Die Korrektur ist von Melodie zu Melodie unterschiedlich. Das Programm wurde in einer for-Schleife gemacht, weil es viele Vorteile bot. Für unterschiedlichen Melodien wird die selbe for-Schleife verwendet. Nur das Ton-Feld und das Tönlänge-Feld muss geändert werden. Das spart Zeit beim Programmieren, weil kein ganz neues Programm geschrieben werden muss. Die Länge der Melodie spielt keine große Rolle mehr. Das Programm wird nicht länger und übersichtlich. MATLAB ist ein Interpreter. Je kürzer die Übersetzung dauert, desto schneller wird die Ausführung. Deswegen konnte der Roboter noch früher in Bewegung setzen.

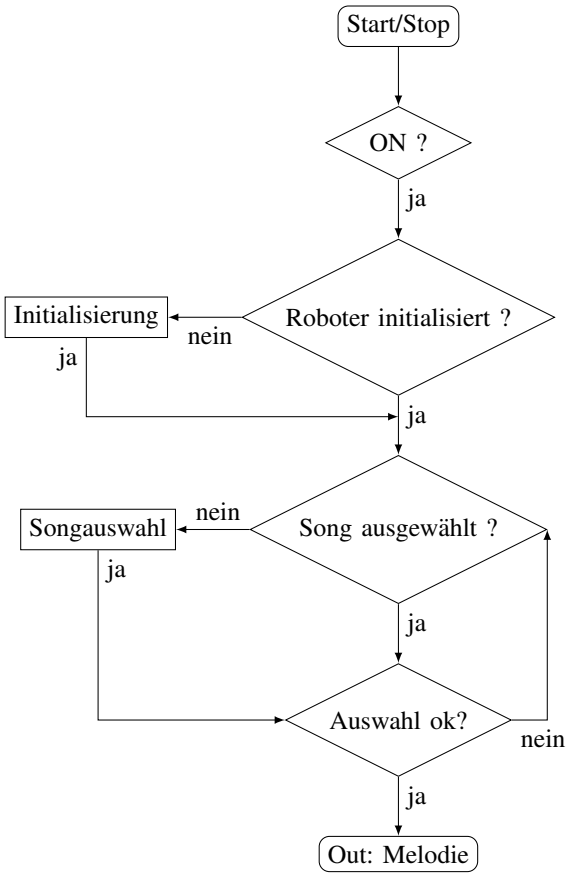


Abbildung 4. Programmablaufplan für die Benutzeroberfläche

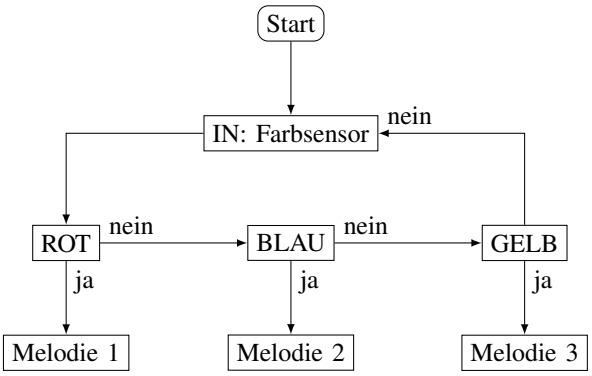


Abbildung 5. Programmablaufplan für Auswahl der Melodien durch Farbsensor

IV. Ergebnisdiskussion

Wie bei der Abschlusspräsentation oder in Youtube [4] zu gucken ist, funktionierte der Roboter tatsächlich schon. Es ist auch erkennbar, dass welche Melodie er gerade darstellt. Trotzdem gibt es auch Probleme, die man nicht vermeiden kann. Die Durchschnittsgeschwindigkeit war noch langsam, im Vergleich mit einem normalen Spieler. Es klingelte nicht schön wie es sein soll. Die Melodien wurden extra gewählt, damit sich der Roboter am wenigsten bemühen muss. Aber das Ergebnis war nicht so toll wie erwartet. Komplexe Melodien sind folglich für den Roboter nicht spielbar. Noch ein Problem

tauchte bei der Vorführung auf, als er gerade eine Melodie spielte, gab das NXT-Gerät ein Ton von sich. Das passierte, weil zu viele Befehle auf einmal gekommen sind. Das Programm muss im Prinzip noch optimiert werden. Die Benutzeroberfläche funktionierte nicht einwandfrei. Bei jeder Ausführung konnte nur eine Melodie ausgewählt werden. Als man die Wahl ändern wollte, funktionierte das Programm nicht mehr.

V. Zusammenfassung und Fazit

Als Fazit lässt sich sagen, dass das Projekt schon in die Richtung fuhr. Mit Maschinen Musik zu machen ist auf jeden Fall eine interessante Idee. Nach der zweiwöchigen Arbeit wurde ein Produkt bereitgestellt und präsentiert. Aber das Endprodukt hätte nicht entstehen können, wenn es keine Mühen von Beteiligten und Hilfen von Betreuers gäbe. Es ist allerdings erforderlich, Grundkenntnisse über MATLAB zu haben und kreativ zu sein. Der Roboter hat noch Potential, bestehende Probleme zu lösen und besser, menschlicher zu spielen. Ein weiterer Motor könnte hinzugefügt werden, um die Leistung und die Geschwindigkeit zu steigern. Das Programm könnte verkürzt und logischer strukturiert werden. Die Anwendungsgebiet könnte in einem Konzert oder in einer Musikschule liegen, aber der Weg bis dahin ist noch so weit.

Anhang

Im Anhang wird der Quelltext des Programms, der den Roboter regelt, um Musik zu machen, gezeigt. Die Aufklärung des Programms befindet sich im Unterabschnitt III.C Logische Steuerung.

```
l = 33; %Abstand von einer Tonhöhe
v = 445/1000; %Viertelnote
h = 2 * v; % Halbenote
g = 2 * h; %ganze Note
tonlaenge = [v,v,v,v,h,h,v,v,v,v,g,v,v,v,v,g,v,v,v,v,h,h,v,v,v,v,g];
toene = [0,1,1,1,1,0,1,0,0,0,-1,1,0,0,0,-1,-1,0,0,0,-1,0,-1,0,0,0,-1];
%Melodie-Schleife
for i = 1:length(toene)

        %Anfahren der Töne
        if (toene(i) == 0) %gleicher Ton
         %nichts passiert
        elseif(toene(i) < 0) %tieferer Ton
             motor_h.Power=100 * -1;
             motor_h.TachoLimit= abs(toene(i)) * l - 1;
             motor_h.SendToNXT();
        else
             motor_h.Power=100; %höherer Ton
             motor_h.TachoLimit= abs(toene(i)) * l;
             motor_h.SendToNXT();
        end
        % Spielen der Töne
        pause(745/1000);
        motor_v.Power=100;
        motor_v.TachoLimit=180;
        motor_v.SendToNXT();
        pause(tonlaenge(i));
end
end
```

Literaturverzeichnis

[1] Wikipedia: *Musikinstrument*, Dezember 2019. http://https://https://de.wikipedia.org/wiki/Musikinstrument

[2] Wikipedia: *Xylophon*, Februar 2020. http://https://de.wikipedia.org/wiki/Xylophon

[3] Kanal Technew Robot, Youtube: *Lego EV3 robot plays music on a glockenspiel xylophone*, https://www.youtube.com/watch?v=-FYDRMKozoc

[4] Kanal Mathias Magdowski, Youtube: *Xylophon-Roboter aus dem Lego-Praktikum 2020 an der OVGU Magdeburg spielt "Twinkle Little Star"*, https://www.youtube.com/watch?v=zFOnkqJ9IaQ

Der Xylophon-Spielende Roboter

Viviane Wolters, Digital Engineering
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Zuge des Projektseminares „Elektrotechnik/Informationstechnik (LEGO Mindstorms)“ der Otto-von-Guericke-Universität Magdeburg wird es den Studenten ermöglicht, innerhalb von zwei Wochen, mit Hilfe von LEGO-Mindstroms-NXT-Bausätzen, eigene Projekte zu realisieren. Diese Arbeit dient als Bericht über eines der Projekte aus diesem Seminar - dem xylophon-spielenden Roboter. Dabei wird auf den mechatronischen Aufbau sowie die Steuerung des Roboters eingegangen. Während des Projektes aufgetretene und nicht zu lösende Probleme, wie die Spielgenauigkeit und die geringe Spielgeschwindigkeit des Roboters, werden außerdem beleuchtet.**

***Schlagwörter*—Xylophon, Roboter, mechanische Musikinstrumente, selbstspielende Musikinstrumente, LEGO Mindstroms**

I. Einleitung

ZEITGLEICH zur Entwicklung der Musikinstrumente wuchs der Wunsch des Menschen, diese selbstspielend zu gestalten. Antrieb für diesen Wunsch war das ureigene Bedürfnis des Menschen nach Musik. Wer zu Zeiten, in denen es weder Radios, CDs oder Internet gab, den Wunsch hatte Musik zu hören, musste entweder selbst ein Instrument spielen oder einen Konzertsaal besuchen. Somit galt das Hören von Musik lange als Privileg der adligen und gut bürgerlichen Bevölkerung. [1]

Die ältesten noch erhaltenen selbstspielenden Instrumente (auch mechanische Musikinstrumente genannt), sind Glockenspiele in Monumentaluhren, welche erstmals im 14. Jahrhundert in Kirchen zum Einsatz kamen. Durch die steigenden technischen und musikalischen Möglichkeiten entwickelten sich die mechanischen Musikinstrumente immer weiter, sodass Anfang des 19. Jahrhunderts sogenannte Musikmaschinisten sogar selbstspielende Orchester konstruierten. Mit der Entwicklung des Grammophons und dem Rundfunk gerieten die mechanischen Musikinstrumente jedoch langsam in Vergessenheit. [2]

Durch die heutzutage immer mehr fortschreitende Digitalisierung und die stetig steigende Anzahl neuer Erkenntnisse im Forschungsbereich der Künstlichen Intelligenz besteht die Chance das selbstspielende Musikinstrumente in Zukunft wieder mehr Aufmerksamkeit erhalten. So wurde beispielsweise bereits ein marimba-spielender Roboter entwickelt, dem es mit Hilfe von Big-Data-Analysen und Deep Learning möglich ist, eigene harmonische Kompositionen zu kreieren und vorzutragen. [3]

In dieser Arbeit wird, angelehnt an die Idee der selbstspielenden Musikinstrumente, die Entwicklung eines xylophonspielenden Roboters beschrieben. Auch wenn dieser Roboter keine künstliche Intelligenz besitzt, so wurde sich dennoch das Ziel gesetzt einen Roboter zu erschaffen, welcher selbständig mehrere einfache Melodien auf einem Xylophon spielen kann. In den folgenden Abschnitten werden zunächst die, für dieses Projekt notwendigen, musikalischen Aspekte erläutert. Im Hauptteil dieses Berichtes wird auf den mechatronischen Aufbau und den verwendeten Algorithmus zur Ansteuerung des Roboters eingegangen. Abschließend werden die aufgetretenen Probleme sowie zukünftig möglichen Verbesserungen und Erweiterungen des Roboters beleuchtet.

DOI: 10.24352/UB.OVGU-2020-042

II. Vorbetrachtungen

Ein selbstspielendes Musikinstrument besteht im wesentlichen aus drei verschiedenen Komponenten: Dem Instrument (hier: das Xylophon), dem Antrieb (hier: der Roboterarm) und dem Toninformationsträger (hier: der Programmcode). Aufgrund dessen, dass die Informationen des zu spielenden Musikstückes in der Steuerung des Roboters hinterlegt und entsprechend verarbeitet werden müssen, ist für die Umsetzung des Projektes ein grundlegendes musikalisches Verständnis notwendig. Die wichtigsten musikalischen Parameter werden im Folgenden näher betrachtet.

A. Musikalische Parameter einer Komposition

Ein Musikstück (Komposition) besteht aus vielen verschiedenen Komponenten. Die wichtigsten zwei dieser Komponenten sind die Melodik und die Rhythmik.
Die Melodik beschreibt die Abfolge der Tonhöhen einer Komposition auch Melodie genannt. [4] Die wichtigsten musikalischen Parameter eines Melodieverlaufes sind die Richtung und die Tonabstände (Intervalle). Die wichtigsten Intervalle sind in Abbildung 1 dargestellt.

Die Rhythmik beschreibt die Abfolge von Noten- bzw. Pausenwerten. Diese Parameter geben Auskunft über die Tondauer einzelner Noten bzw. die Pausenzeiten an. [4] Diese beiden Zeiten stehen immer im Verhältnis zu der Anzahl der im Musikstück enthaltenen Noten bzw. Pausen pro Takt und der allgemeinen Spielgeschwindigkeit der Komposition.
Abbildung 2 zeigt die in diesem Projekt verwendeten Noten- und Pausenwerte und deren Relation zueinander.

B. Das Xylophon

Das in diesem Projekt verwendete Xylophon besitzt zwölf Klangplatten mit einem Tonumfang von a bis e” (vgl. Abbildung 4). Klangplatten für chromatische Zwischentöne wie beispielsweise das Fis oder Cis besitzt das Xylophon nicht, sodass nur Musikstücke welche in C-Dur geschrieben sind, gespielt werden können. Bei der Melodieauswahl für dieses Projekt ist somit zu beachten, dass diese mit den vorhandenen Tönen spielbar sind. Abbildung 3 zeigt eine C-Dur Tonleiter und den in diesem Projekt verwendeten Tonumfang.

Abstand	Beispiel	Name	deutsch
vom 1. zum 1. Ton		Prime	der erste
vom 1. zum 2. Ton		Sekunde	der zweite
vom 1. zum 3. Ton		Terz	der dritte
vom 1. zum 4. Ton		Quarte	der vierte
vom 1. zum 5. Ton		Quinte	der fünfte
vom 1. zum 6. Ton		Sexte	der sechste
vom 1. zum 7. Ton		Septime	der siebente
vom 1. zum 8. Ton		Oktave	der achte

Abbildung 1. die wichtigsten Intervalle eines Musikstückes [5]

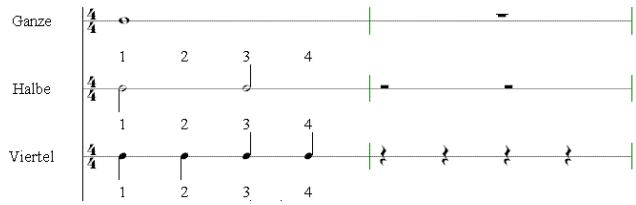


Abbildung 2. Für dieses Projekt relevante Noten- und Pausenwerte [6]

III. Realisierung

Für den xylophon-spielenden Roboter werden Bauteile von mehreren LEGO-Mindstroms-NXT-Bausätzen, eine Grundplatte von LEGO Classic und ein kleines Xylophon mit Spielstab benötigt. Die Abbildung 4 zeigt den Aufbau des Xylophon-Roboters, welcher im Folgenden näher erläutert wird.

A. Mechanik

Für den Roboterarm werden zwei Motoren benötigt, welche für die vertikale und horizontale Bewegung des Spielstabes zuständig sind.

Der Motor für die Horizontalbewegung (hier: $Motor_H$) wird fest auf die Grundplatte montiert und bewegt mit Hilfe eines kleinen Zahnrades einen größeren Drehkranz auf dem der

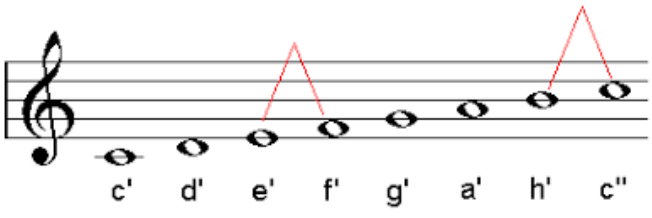


Abbildung 3. C-Dur Tonleiter [7]

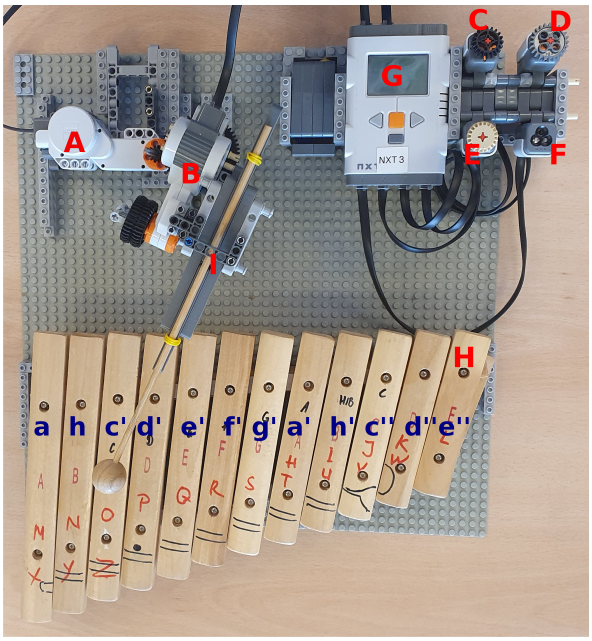


Abbildung 4. Aufbau Xylophon Roboter,
A: $Motor_H$ (horizontale Bewegung), B: $Motor_V$ (vertikale Bewegung), C: $Tastsensor_{S1}$ (Spielstab nach rechts), D: $Tastsensor_{S2}$ (Spielstab nach links), E: $Tastsensor_{S3}$ (Spielstab nach oben/unten), F: Farbsensor (Melodieauswahl), G: NXT-Steuerungsgerät, H: Xylophon, I: Spielstabhalterung

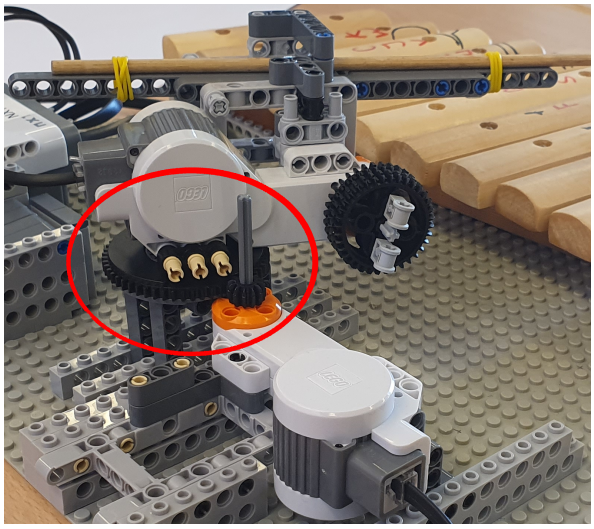


Abbildung 5. Drehkranz und Zahnrad für die horizontale Bewegung des Roboters

zweite Motor (hier: $Motor_V$) montiert ist (siehe Abbildung 5). Die Ansteuerung über die Zahnräder vergrößert das Übersetzungsverhältnis der Motorumdrehung zur Spielstabbewegung und verbessert die Genauigkeit der Ansteuerung der Klangplatten des Xylophons.

Der $Motor_V$ bewegt eine Rotationsachse, an welcher ein kleiner Stift befestigt ist. Vertikal ausgerichtet, hält dieser Stift den, darüber befestigten, Spielstab in seiner höchsten Position. Wird der $Motor_V$ angesteuert so rotiert dieser Stift und lässt den Spielstab nach unten schnellen. Pro Motorumdrehung könnten demnach zwei Töne auf dem Xylophon erzeugt werden.

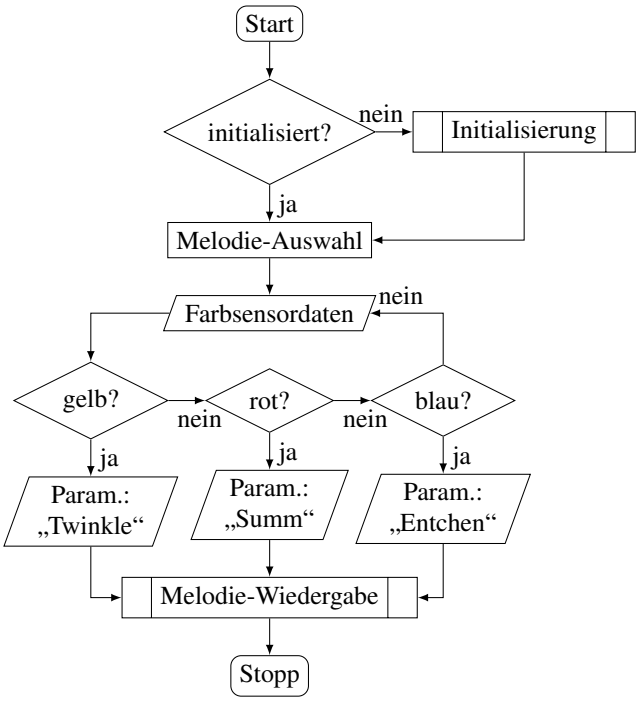


Abbildung 6. Programmablaufplan des Xylophon-Roboters

B. Elektronik

Drei Tastsensoren sowie ein Farbsensor bilden das Bedienfeld des Roboters. Die Tastsensoren dienen zur Handansteuerung des Roboterarms, um die Startposition des Spielstabes initialisieren zu können. Mit Hilfe des Farbsensors werden verschiedene Melodien ausgewählt.

C. Steuerung

1) Programmablauf: Das Roboterprogramm besteht aus drei grundlegenden Prozessen: der Initialisierung, der Melodie-Auswahl und der Melodie-Wiedergabe. Wie diese Prozesse zusammenarbeiten wird in der Abbildung 6 anhand eines vereinfachten Programmablaufplanes dargestellt.

Begonnen wird mit der Initialisierung der Startposition. Dabei wird der Spielstab mittels des $Tastsensor_{S1}$ und $Tastsensor_{S2}$ über der Klangplatte für die Note c' des Xylophons positioniert und mit Hilfe des $Tastsensor_{S3}$ die höchste Position des Stabes angefahren. Durch gleichzeitiges Betätigen von $Tastsensor_{S1}$ und $Tastsensor_{S2}$ wird die Motorposition von $Motor_H$ auf 0 gesetzt, die Initialisierung quittiert und abgeschlossen.

Nach der Initialisierung kann die abzuspielende Melodie gewählt werden, indem die entsprechende Farbe vor dem Farbsensor gehalten wird. Der Roboter startet nach dem Erkennen der Farbe automatisch mit dem Spielen der zugehörigen Melodie. Die Tabelle I zeigt die Zuordnung der Farben zu den im Zuge des Projektes programmierten Kompositionen.

2) Melodie-Algorithmus: Wie bereits in den Vorbetrachtungen erwähnt, sind die wichtigsten Parameter eines Musikstückes die verschiedenen Intervalle und die Richtung des Melodieverlaufes sowie die verschiedenen Noten- bzw. Pausenwerte. Zum Spielen des Xylophons müssen diese Parameter dementsprechend definiert werden. Anschließend können diese in einem Algorithmus verarbeitet und die Motoren des Roboters entsprechend angesteuert werden.

Tabelle I
AUSWÄHLBARE MELODIEN UND DEREN FARBZUORDNUNG

Farbe	Melodie
blau	Alle meine Entchen
rot	Summ summ summ
gelb	Twinkle twinkle little star

Die musikalischen Parameter werden in zwei separaten Arrays gespeichert. Das erste Array besteht aus den einzelnen Intervallen der Melodie, kombiniert mit einem entsprechenden Vorzeichen, um die Richtung des Melodieverlaufes beschreiben zu können. Das zweite Array besteht aus den Tonwerten des Musikstückes. Einfachheitshalber wurden existierende Pausenwerte auf die im Melodieverlauf vorherigen Notenwerte addiert, sodass zwischen diesen beiden Parametern nicht unterschieden werden muss.
Neben den beiden Arrays muss außerdem der Drehwinkel definiert werden welcher vom Motor gefahren werden muss, um das Sekunden-Intervall (siehe Abbildung 1) realisieren zu können. Dieses wurde im Zuge des Projektes experimentell ermittelt. Alle weiteren Intervalle können auf dessen Grundlage errechnet werden.

Der eigentliche Algorithmus (vgl. Algorithmus 1) besteht aus einer For-Schleife, die durch die beiden Parameter-Arrays iteriert:

In jedem Durchgang wird zunächst eine Fallunterscheidung durchgeführt. Ist das Intervall 0, bewegt sich der $Motor_H$ nicht. Ist das Intervall kleiner als 0 (negatives Intervall) wird ein tieferer Ton als der vorherige angesteuert. Ist das Intervall größer als 0 wird dementsprechend ein höherer Ton angesteuert.

Die anzufahrenden Winkelpositionen des Motors ergeben sich hierbei aus dem Produkt des im Array hinterlegten Intervall und dem experimentell ermittelten Drehwinkel für das Sekunden-Intervall. Bei Intervallen größer als Vier muss noch eine kleine Korrektur an der anzufahrenden Winkelposition vorgenommen werden, da in diesen Fällen Abweichungen von der Soll-Position des Spielstabes zu beobachten waren. Die hier zu addierende Konstante muss, wie der Drehwinkel, experimentell ermittelt werden.

Nach der Fallunterscheidung und dem Ansteuern des $Motor_H$ muss eine kurze Pause erfolgen. Würde dies nicht geschehen, käme es zu Störungen, da die Signale zu schnell an das Steuergerät gesendet würden. Die Länge dieser Pause muss ebenfalls experimentell bestimmt werden.

Nach der kleinen Pause wird der $Motor_V$ angesteuert und die entsprechende Klangplatte angeschlagen. anschließend muss wieder eine Pause erfolgen, zum einen um erneuert zu garantieren, dass die Signale nicht zu schnell an das Steuergerät gesendet werden und zum anderen, um die verschiedenen Notenwerte zu realisieren. Auch hierbei muss experimentell

ermittelt werden, welche Pausenzeit mindestens einzuhalten ist. Dieser Wert definiert dann die Länge einer Viertelnote. Die anderen Notenwerte können danach entsprechend berechnet werden (siehe Abbildung 2).

Wenn die Parameter-Arrays vollständig von der For-Schleife durchlaufen sind, ist das Ende des Musikstückes erreicht und das Programm wird beendet.

Algorithm 1 Melodie-Wiedergabe (Pseudocode)

```
for i ← Laenge der Arrays do
    if Intervall(i) == 0 then
        nichts passiert
    else if abs(Intervall(i)) >= 4 then
        M_H Power ← 100
        M_H TachoLimi. ← abs(Intervall(i)) * Drehw. + c
        SendToNXT
    else if Intervall < 0 then
        M_H Power ← 100 * −1
        M_H TachoLim. ← abs(Intervall(i)) * Drehw.
        SendToNXT
    else
        M_H Power ← 100
        M_H TachoLim. ← abs(Intervall(i)) * Drehw.
        SendToNXT
    pause(minval)
    M_V Power ← 100
    M_V TachoLim. ← 180
    SendToNXT
    pause(Notenwert(i))
```

IV. Ergebnisdiskussion

Es ist gelungen einen Roboter zu konstruieren, welcher einfache Melodien auf einem Xylophon wiedergeben kann. Dabei sind verschiedene Probleme aufgetreten, von denen nicht alle zufriedenstellend gelöst wurden konnten.

A. Genauigkeit der Ansteuerung

Immer mal wieder treten Abweichungen bezüglich der Soll-Position des Spielstabes auf, mit der Folge, dass falsche Töne angespielt werden.
Bei einem früheren Prototypen war es noch nicht einmal möglich, die einfache C-Dur Tonleiter fehlerfrei zu spielen. Eine Vergrößerung des Übersetzungsverhältnis von $Motor_H$ zur Spielstabbewegung durch Einsatz eines Drehkranzes, anstelle der zuvor erprobten direkten Montage des $Motor_V$ auf den $Motor_H$ (vgl. Abschnitt III.A Mechanik), konnte der Ungenauigkeit entgegen wirken.
Trotz alledem kommt es von Zeit zu Zeit zu ungewollten Abweichungen. Diese Abweichungen können unter anderem durch eine ungenaue Initialisierung (menschlicher Fehler) oder durch ein Verändern der Position des Xylophons hervorgerufen werden. Abhilfe würde hier eine automatisierte Initialisierung und eine verbesserte Fixierung des Xylophons verschaffen.

B. Geschwindigkeit des Roboters

Der Xylophon-Roboter kann Melodien nicht beliebig schnell abspielen. Gelangen die Signale zur Ansteuerung der Motoren zu schnell zum Steuergerät, führt dies zu einer Störung. Das Spielen von Notenwerten kleiner als eine Viertelnote ist somit nicht möglich ohne eine Reduzierung der gesamt Geschwindigkeit, womit sich die Melodie verzerrt anhören würde. Um dieses Problem zu lösen, könnte ein weiterer Roboterarm installiert werden, sodass der Arbeitsraum zwischen den Robotern aufgeteilt werden kann. Dies könnte sich ebenfalls positiv auf die Spielgenauigkeit auswirken. Um dies umsetzen zu können, müsste jedoch ein zweites Steuergerät zum Einsatz kommen, da der eingesetzte NXT-Baustein nur drei Motoren ansteuern kann.

C. Komplexität der Melodien

Aufgrund des eingeschränkten Arbeitsbereiches des Roboterarmes, den Limits bezüglich der Spielgeschwindigkeit und des eingeschränkten Tonumfangs des Xylophons, können nur sehr einfache Melodien vom Roboter gespielt werden. Ein Zusammenschalten von mehreren Robotern mit unterschiedlichen Xylophonen wäre hierfür die Lösung.

V. Zusammenfassung und Fazit

Im Zuge dieser Arbeit wurde die Realisierung eines xylophon-spielenden Roboters von der Konstruktion bis hin zur Programmierung betrachtet und aufgetretene Probleme diskutiert. Mit dem entstandenen Roboter ist es möglich, einfache Melodien wie z.B. „Alle meine Entchen“ abzuspielen. Eine Garantie, dass dies fehlerfrei geschieht, gibt es allerdings nicht. Eine Verbesserung der Initialisierung könnte hierbei Abhilfe verschaffen.

An die technisch möglichen Grenzen gelangt der Roboter bezüglich komplexerer und vor allem schnellerer Musikstücke. Eine mögliches, auf den hier vorgestellten Roboter aufbauendes, Projekt könnte daraus bestehen, dieser Problematik mit Hilfe eines weiteren Roboterarmes entgegen zu wirken.

Trotz aller zukünftig möglichen Verbesserungen bleibt ein Roboter jedoch ein Roboter, welcher nur eine bestimmte zeitliche Abfolge von Tönen abspielen und nicht wie ein Mensch die eigenen Gefühle und Emotionen in die Interpretation des Musikstücks einfließen lassen kann.

Literaturverzeichnis

[1] Surber, Hansjürg: *Mechanische Musikinstrumente, Bedeutung, Funktionsweise und soziale Stellung*. http://www.musikautomaten-ungarn.eu/de/geschichtemechanischemusik.html. Version: 2017

[2] Tenten, Walter and GSMeV: *Mechanische Musikinstrumente*. https://www.musica-mechanica.de/pages/de/geschichte.php. Version: November 2015

[3] Gerstel, Sebastian: *Xylophon-Roboter komponiert und spielt seine eigene Musik*. https://www.elektronikpraxis.vogel.de/xylophon-roboter-komponiert-und-spielt-seine-eigene-musik-a-616930/. Version: Juni 2017

[4] Köppl, Birgit: *Musik-Analyse: Wichtige musikalische Parameter*. https://www.gym-raubling.de/medien/Unterricht/Musik/musik_analyse_klasse_10.pdf

[5] Kaiser, Ullrich: *Intervalle bestimmen*. https://musikanalyse.net/tutorials/intervalle-bestimmen/. Version: 2018

[6] Kaiser-Kaplaner, Johannes: *Notenwerte (Klangdauer)*. https://www.musiklehre.at/1_002.htm. Version: 2019

[7] Kaiser-Kaplaner, Johannes: *Diatonik - Durtonleiter*. https://www.musiklehre.at/9_004.htm. Version: 2019

Pfandflaschenautomat

Konstantin Bredenfeld, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In dieser Arbeit wird der Bau und die Funktionsweise eines Pfandflaschenautomates dokumentiert. Dieser wurde im Rahmen des zweiwöchigen Projektseminars Elektrotechnik/Informationstechnik im Wintersemester 2020 gebaut. Hierfür wurden ein programmierbarer NXT-Baustein und LEGO-Technik-Bauteile verwendet. Hinzu kommen zwei Motoren,drei Ultraschallsensoren, eine Webcam für die Erkennung des Barcodes mit Hilfe der Zebra Crossing Library und ein Taster. Programmiert wurde der NXT-Baustein mit MATLAB. Der Transport sowie die Rotation der Flasche und die Erkennung des Barcodes auf der Pfandflasche wurden erfolgreich realisiert.**

***Schlagwörter*—Barcode, LEGO Projektseminar, Mindstorms, NXT, Pfandflasche, Pfandflaschenautomat**

I. Einleitung

Seit Januar 2003 gilt in Deutschland eine Pfandpflicht für Getränkeverpackungen. Das System wurde eingeführt, um die Wiederverwendung von Flaschen zu ermöglichen. Die Rücknahme von Getränkeverpackungen wie Einweg-, Mehrwegplastik-, oder Glasflaschen erfolgt durch den Pfandflaschenautomat. Dieser ist verantwortlich für die Erkennung und Sortierung von Pfandflaschen. Einwegflaschen werden sofort zerkleinert wohingegen Mehrwegplastik- und Glasflaschen separiert werden. In einem zweiwöchigen Projektseminar im Wintersemester 2019/20 wurde an der OVGU Magdeburg in Zusammenarbeit mit Robin Kürbis ein Pfandflaschenautomat gebaut und programmiert. Im Seminar wurden Grundlagen der Programmierung in MATLAB und vom NXT-Baustein vermittelt. Das Ziel der Arbeit war es, einen Pfandflaschenautomaten zu bauen, welcher Flaschen transportiert, rotiert und den Barcode erkennt, um zu ermitteln, ob es sich um eine Pfandflasche handelt. Anschließend soll der Automat die Flasche abtransportieren oder dem Benutzer wieder zukommen lassen. Wenn der Benutzer keine Flaschen mehr abgeben möchte, kann dieser den Vorgang über einen Knopf abbrechen. Als Grundlage für den Pfandflaschenautomat dienten der programmierbare NXT-Baustein und LEGO-Technik-Bausteine. Programmiert wurde in MATLAB [1] und mithilfe eines Toolkit der RWTH Aachen [2]. Mit diesem ist ein direkter Zugriff auf die Sensoren und Motoren von LEGO möglich. Die Erkennung der Flasche erfolgt über das Erfassen des Barcodes mit einer Kamera und einer zusätzlichen Bibliothek [3].

II. Vorbetrachtungen

A. Prinzip

Das Prinzip eines Pfandflaschenautomatens ist auf den ersten Blick relativ simpel. Jedoch werden mehrere Schritte durchgeführt, bevor eine Pfandflasche sortiert wird. Zur Erkennung einer Pfandflasche sind eine Vielzahl an Komponenten vonnöten. Dazu zählen: Laufband, Rotationsmechanismus mithilfe einer Rolle, Gewichtssensor, Kamera und Laserscanner. Das Laufband transportiert die Flasche vom Eingang ins Innere des Leergutautomaten. Dort angekommen wird mit einem Laserscanner das Pfandsymbol und der Barcode erfasst. Der Strichcode wird mit einer Datenbank verglichen und es wird geprüft, ob es sich um eine Pfandflasche handelt. Anschließend wird über die Kamera die Kontur der Flasche ermittelt. Entspricht die Kontur der Flasche derjenigen aus der Datenbank, wird die Flasche angenommen. So wird auch verhindert, das verformte Flaschen angenommen werden. Zusätzlich ist ein Gewichtssensor verbaut, um nur nahezu leere Flaschen anzunehmen. Wird der Barcode der Getränkeverpackung nicht im ersten Moment erkannt, wird diese durch den Rotationsmechanismus gedreht. Der Barcode enthält darüberhinaus Informationen über die Art der Pfandflasche. Einwegflaschen werden in einem Kompaktor zerkleinert und in Behälter verstaut. Hingegen werden Mehrwegflaschen wieder in Kästen per Hand einsortiert. Der Aufrichter ist für das Aufstellen der Mehrwegflasche verantwortlich. Diese funktioniert nur, wenn die Flasche mit dem Boden als erstes in den Leergutautomaten gesteckt wird. Möchte der Kunde keine weiteren Pfandflaschen einwerfen, kann sich dieser abschließend vor Ort einen Pfandbon ausdrucken lassen.

B. Stand der Technik

Die neuesten Modelle der Leergutautomaten besitzen 360° Scanner. Dabei wird die Flasche bereits beim Einwurf in den Automaten gescannt. Die Flasche darf aber nicht zu schnell eingeworfen werden, da sonst der 360° Scanner keine Möglichkeit hat, sie zu erkennen. Mit diesem Verfahren wird der Rotationsmechanismus überflüssig und daher eine höhere Durchlaufzeit erreicht. Desweiteren gibt es Systeme, bei denen die Einsortierung von Mehrwegflaschen in Kästen automatisiert erfolgt. Eine zusätzliche Arbeitskraft für das Einordnen ist so nicht mehr nötig.

C. Barcode

Der Barcode ist eine Abfolge von hintereinander gereihten schwarzen (1) und weißen (0) Strichen. Es gibt eine Vielzahl von Barcodes. Die populärsten sind der QR-Code und die ISBN. In der Lebensmittelindustrie werden vorrangig zwei Arten von Strichcodes verwendet: EAN-8 und EAN-13. Sie bestehen aus 8 bzw. 13 Ziffern. Eine Ziffer wird durch 6 schwarze oder weiße Striche dargestellt. EAN steht für Europäische Artikel-Nummer. Die EAN-8 wird nur an Artikel vergeben, bei denen eine EAN-13 mehr als 25 % der Frontfläche in

DOI: 10.24352/UB.OVGU-2020-043

Abbildung 1. EAN-13-Barcode [5]

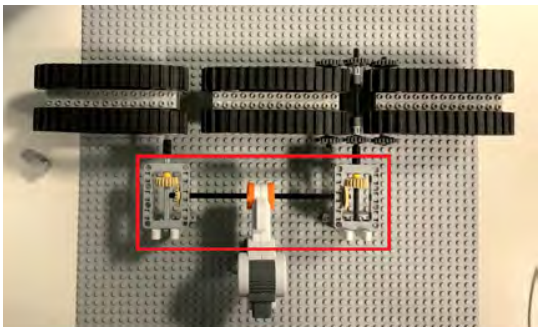


Abbildung 2. Laufband mit Zahnradkonstruktion

Anspruch nehmen würde. Hinzu kommt, dass es durch die geringeren Variationsmöglichkeiten der Zahlenkombinationen weniger EAN-8 als EAN-13 gibt. Daher werden EAN-8 seltener vergeben. Links und rechts besitzt die EAN jeweils ein Start- und Stoppzeichen. Dieses ist bei jeder EAN gleich und besteht aus abwechselnd zwei schwarzen und einem weißen Strich (1 0 1). Damit wird dem Auslesegerät der Anfang und das Ende vom Barcode vermittelt. Vor beziehungsweise nach dem Start- oder Stoppzeichen ist eine Hellzone ohne Störzeichen für die richtige Erkennung des Barcodes nötig. Zusätzlich gibt es ein Trennzeichen in der Mitte des Barcodes. Dieser besteht aus drei weißen und zwei schwarzen Strichen (0 1 0 1 0). Zusätzlich besitzt die EAN an letzter Stelle eine Prüfziffer zur Selbstkontrolle der Richtigkeit des Barcodes. Man multipliziert die Ziffern abwechselnd mit 1 und 3 bis auf die letzte Ziffer (links mit 1 anfangen) und summiert alle Teilwerte. Die Prüfziffer ergibt sich aus der Differenz der Summe zum nächsten Vielfachen von 10. Die ersten zwei bis drei Ziffern des Barcodes stehen für die Länderkennung. Dabei handelt es sich aber nicht um das Herkunftsland des Produktes, sondern ausschließlich um das Land, in welchem der Barcode vergeben wurde. [4]

III. Konstruktion und Programmierung des Pfandflaschenautomatens

Als Grundlage für den Bau des Roboters stand der LEGO-Mindstorms-Baukasten zur Verfügung. Dieser beinhaltet den programmierbaren NXT-Baustein, Ultraschallsensoren, Taster, Farbsensoren, Geräuschsensoren und weitere Lego Technik Bausteine. Der NXT wurde mit der MATLAB Version 2019a programmiert. Der NXT-Baustein besitzt drei Ausgänge, vier Eingänge und einen USB-Anschluss, über den der Datenaustausch und die Programmierung erfolgt. Das Ziel war es, mit dieser Ausgangsposition eine vereinfachte Form des Pfandflaschenautomates zu bauen. Dieser Automat nimmt ausschließlich nahezu leere Plastikflaschen mit einer maximalen Füllmenge von 0,5 l an. Während des Seminars wurde der Automat gebaut und programmiert.

A. Aktorik

Für den Transport der Pfandflasche vom Einwurf bis zum Ende des Pfandflaschenautomaten wurde eine Kettenkonstruktion verwendet. Diese bietet Vorteile gegenüber einem System mit Rollen. Die Getränkeverpackung muss nicht exakt auf die Kettenkonstruktion gelegt werden, sondern kann auch leicht geworfen werden, wie es auch in der Realität ist. Das System transportiert hierbei die Pfandflasche fehlerfrei ab. Bei Rollen kann es zu Komplikationen führen, wenn sich eine Flasche zwischen den Rädern verhakt. Das Laufband bestand anfangs nur aus zwei später jedoch aus drei Kettenpaaren. Mit dieser Anpassung konnte das Laufband eine größere Strecke erreichen, um die Pfandflasche bis zur Barcodeerkennung zu transportieren. Die Kettenpaare wurden über eine Welle und zwei Zahnradkonstruktion angesteuert (siehe rote Makierung Abb. 2). So konnte nur mit einem Motor (Motor A) gleichzeitig drei Kettenpaare angesteuert werden.

Der zweite Motor (Motor B) wurde im Rotationsmechanismus für die Getränkeverpackung verbaut. Anfangs stand die Überlegung im Raum, den Mechanismus über mehrere Räder zu realisieren, welche anschließend seitlich am Laufband hochfahren, um die Flasche zu rotieren. Es wurde sich jedoch gegen diese Idee entschieden, da sich bei zahlreichen Tests diese Methode als sehr fehleranfällig herausstellte. Zusätzliche hätte ein zweiter Motor für die horizontale Bewegungsrichtung verbaut werden müssen.

Schlussendlich wurde sich für eine ähnliche Methode entschieden. Dabei ist der Motor fest neben dem Laufband verbaut und über eine Welle mit zwei Querstreben verbunden (siehe blaue Makierung Abb. 3). Beim Hineinfahren der Flasche stehen diese horizontal, damit diese nicht die Flasche behindern. Auf der gegenüberliegenden Seite des Rotationsmechanismus sind lose Rollen auf einer Welle angebracht (siehe rote Markierung Abb. 3). Sie ermöglichen eine komplikationsfreie Rotation. Soll es nun zu einer Rotation kommen, drehen sich die Querstreben, so dass die Flasche ebenfalls in eine Rotationsbewegung um ihre eigene Achse kommt.

B. Sensorik

Um die Position der Pfandflasche zu ermitteln, sind insgesamt drei Ultraschallsensoren verbaut. Der erste Ultraschallsensor (Sensor 1) ist direkt am Anfang des Pfandflaschenautomates mit einem Abstand von 6 cm zum Laufband angebracht. Der Grund dafür liegt in der Messweise des Sensors. Dieser gibt erst ab ca. 5 cm wahrheitsgemäße Werte für die Distanz aus. Zusätzlich ist eine Pappe auf der gegenüberliegende Seite des Laufbandes angebracht. Wenn keine Flasche im Laufband ist, gibt so der Sensor konstante Werte aus, da an der planen Fläche die Ultraschallwellen reflektiert werden. Zuvor

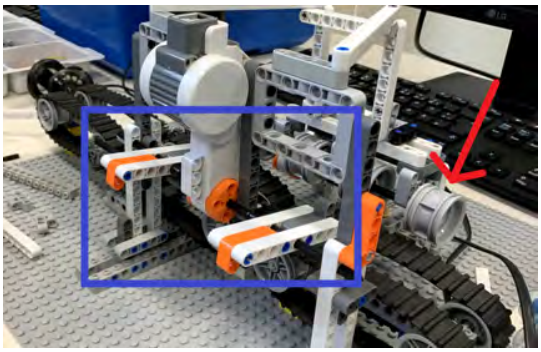


Abbildung 3. Finaler Roationsmechanismus

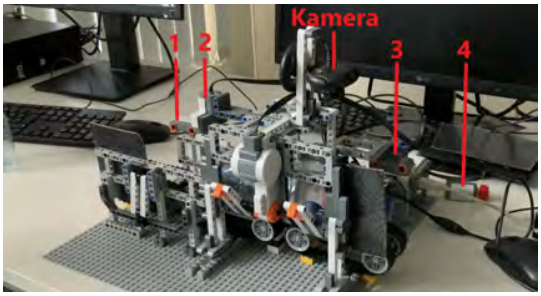


Abbildung 4. Bild von Taster, Ultraschallsensoren und Kamera

war ausschließlich eine Lego-Konstruktion verbaut, die die Reflexion nur teilweise ermöglichte. Die Funktion des Sensors ist, zu prüfen, ob eine Pfandflasche in den Leergutautomaten gelegt wurde. Der zweite Ultraschallsensor (Sensor 2) ist mittig oberhalb des Laufbandes platziert. Seine Aufgabe ist es, zu erkennen, ob die Flasche das Kameramodul erreicht hat. Wenn die Getränkeverpackung richtig liegt, gibt der Sensor die Distanz vom Laufband bis zum Ultraschallsensor zurück. Davor kann nur der Abstand vom Sensor bis zur Flasche gemessen werden.

Der letzte Ultraschallsensor (Sensor 3) befindet sich am Ende des Laufbandes. Dieser dient als Absicherung für den zweiten Sensor, damit die Flasche genau unter dem Kameramodul liegt. Hier ist wie beim ersten Ultraschallsensor eine Pappe verbaut und der Abstand beträgt ebenfalls 6 cm vom Laufband bis zum Sensor. Als letztes ist ein Tastsensor (Sensor 4) am Leergutautomaten verbaut. Dieser symbolisiert den Knopf zum Ausdrucken des Pfandbons und beendet das Programm. Für die Barcode-Erkennung wurde eine Webcam oberhalb des Laufbandes angebracht. Die Kamera ist über ein USB-Kabel an den Computer angeschlossen auf dem ebenfalls das Programm läuft. Darüber hinaus besitzt die Webcam sechs LEDs um, die Flasche auszuleuchten. Der Fokus lässt sich manuell an der Linse der Kamera einstellen.

C. Programm

Der NXT Baustein wurde über ein USB-Kabel mit MATLAB 2019a programmiert. Das MATLAB-Script nutzt mehrere Funktionen für die Ansteuerung der Motoren und das Auslesen der Ultraschallsensoren. Zusätzlich wird mithilfe der Zebra Crossing Library das Kamerabild ausgewertet und

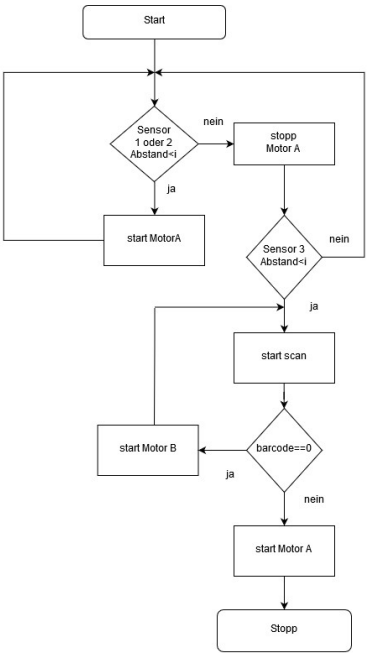


Abbildung 5. Programmablaufplan

nach EAN-8/EAN-13 Barcodes gesucht. In Abbildung 5 ist ein vereinfachtes Schema des Programms zu sehen. Das erste Ziel des Programm ist es, die Pfandflasche bis zur Kamera zu transportieren. Dafür werden nacheinander die Ultraschallsensoren ausgelesen, um zu ermitteln, wo sich die Flasche befindet. Solange Sensor 1 oder 2 eine Flasche erkennt, bewegt sich das Laufband durch den Motor A. Wenn die Sensoren nicht die Bedingung erfüllen, wird das Laufband angehalten und der Sensor 3 ausgelesen. In diesem Fall befindet sich erst gar keine Flasche im Laufband oder wurde bereits zum Kameramodul transportiert. Wird die Bedingung vom Sensor 3 nicht erfüllt, beginnt das Programm von vorne.

Sollte sich eine Flasche unter der Webcam befinden, ist die Bedingung vom Sensor 3 erfüllt. Nun wird ein Bild von der Flasche aufgenommen und mit Hilfe der Zebra Crossing Library decodiert und auf Barcodes analysiert. Wird kein Strichcode erkannt, geht der Leergutautomat davon aus, dass die Flasche nicht richtig unter der Kamera liegt und steuert Motor B an, um die Getränkeverpackung zu rotieren. Anschließend wiederholt sich der Vorgang insgesamt sechsmal bis andernfalls die Flasche wieder ausgefahren wird. Durch Testen hat sich gezeigt, dass der Automat nach der sechsten Umdrehung die Möglichkeit hatte, das Etikett von der gesamten Flasche auszuwerten. Sollte ein Barcode erkannt werden, gibt die Funktion getbarcode.m den Strichcode aus (siehe rote Markierung Abb. 6). Dieser wird anschließend dem Benutzer ausgegeben und mit bereits bekannten Flaschen abgeglichen. Kommt es zu einer Übereinstimmung mit einer eingespeicherten Flasche, wird auch der Name ausgegeben. Anschließend wird Motor A über eine Funktion angesteuert, so dass die Flasche abtransportiert wird. Das Programm wird beendet, wenn der Benutzer den Sensor 4 betätigt. Andernfalls fängt das Programm wieder von vorne an.

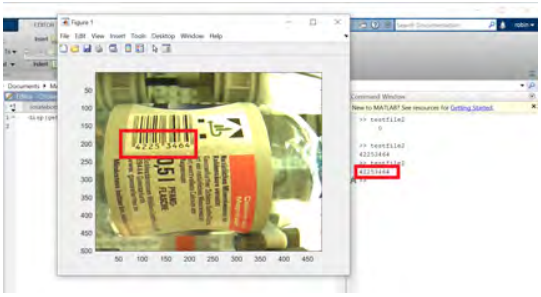


Abbildung 6. Barcodeerkennung

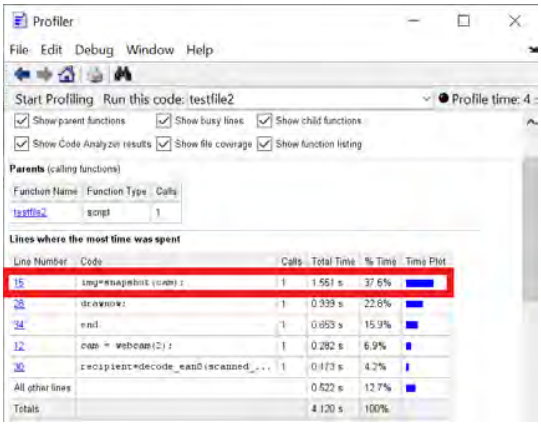


Abbildung 7. Codeanalyse mit Profiler

IV. Ergebnisdiskussion

Der Pfandflaschenautomat war am Ende des Seminars in der Lage, den Barcode einer Pfandflasche zu decodieren, die Getränkeverpackung zu transportieren und zu rotieren. Dabei gibt es jedoch die Einschränkung, das der Benutzer die Flasche mit dem Verschluss als erstes in der Automaten legen muss. Weiterhin geht der Leergutautomat bei der Erkennung eines Barcodes davon aus, dass es sich zweifelsfrei um eine Pfandflasche handeln muss. Das Pfandsymbol wird nicht erkannt, sondern nur der Barcode. Bei der Konstruktion des Automaten traten Schwierigkeiten bei der Realisierung des Rotationsmechanismus auf. Die wohl größte Hürde war das Erkennen des Barcodes. Dies wurde durch eine zusätzliche Bibliothek ermöglicht. Ein weiteres Problem war die hohe Laufzeit der Barcodeerkennung. Dieses lag vorrangig im Fotografieren der Getränkeverpackung (siehe rote Markierung Abb. 7). Dieses Problem wurde gelöst, indem das Kamerabild die ganze Zeit ausgelesen wurde und nicht erst wenn ein Foto geschossen werden soll.

Hinzu kamen Messungenauigkeiten durch die Ultraschallsensoren (siehe Abb. 8). Ein weiterer Faktor war die zylindrische Form der Pfandflasche. Dadurch wurden die Ultraschallwellen nicht optimal zurückgestrahlt und die Werte verfälscht. So kam es zu Werten die weit über den möglichen Wertebereich gingen. Dies ist im Diagramm bei 2 und 2,5 s zu sehen. Infolgedessen stoppt die Flasche selten nicht genau unter der Kamera und macht in diesem Fall eine Barcodeerkennung nicht immer möglich.

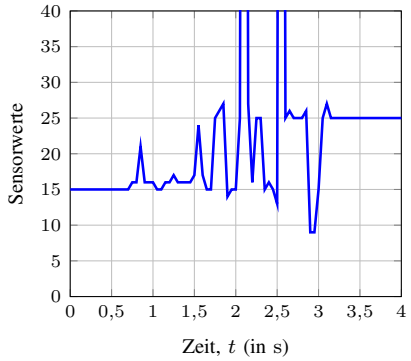


Abbildung 8. Zeitlicher Verlauf des Ultraschallsensor 3 beim Herreinfahren der Flasche in den Leergutautomaten

Das letzte Problem, welches nur provisorisch gelöst werden konnte, ist die Verfälschung des Kamerabildes durch Lichteinstrahlung. Das Etikett reflektiert Lichteinstrahlung so, dass die Barcodeerkennung nicht erfolgreich funktioniert. Um dies zu verhindern, musste der Raum abgedunkelt oder eine Pappe oberhalb der Kamera angebracht werden.

V. Zusammenfassung und Fazit

Das Ziel, eine vereinfachte Form des Pfandflaschenautomaten zu bauen, welcher Pfandflaschen transportiert, rotiert und diese durch einen Barcode erkennt, wurde erreicht. Ein Großteil der aufgetretenen Probleme konnte erfolgreich gelöst werden. Die übrig gebliebenen Probleme, wie das Weiterfahren der Flasche, könnten durch genauere Ultraschallsensoren oder eine zusätzliche Auswertung des Kamerabildes bzgl. der Konturen der Flasche gelöst werden. Eine mögliche Verbesserung des Systems wären zwei weitere Kameras, die ebenfalls das Etikett scannen. Damit wäre eine höhere Erfassungsquote von Barcodes und eine schnellere Laufzeit des Automaten möglich. Ein Einsatz in einem Supermarkt wäre nicht sinnvoll, jedoch erfüllt der Pfandflaschenautomat seine Zielsetzung in diesem Rahmen.

Literaturverzeichnis

[1] Mathworks: *MATLAB*. https://de.mathworks.com/products/matlab.html. – [Abgerufen am 6. März 2020]

[2] Mathworks: *RWTH Aachen - Minestorms NXT Toolbox*. https://de.mathworks.com/matlabcentral/fileexchange/18646-rwth-mindstorms-nxt-toolbox. – [Abgerufen am 6. März 2020]

[3] Google: *ZXing (Zebra Crossing) barcode scanning library for Java, Android*. https://github.com/zxing/zxing/. – [Abgerufen am 6. März 2020]

[4] Gruber, Gerald: *EAN/UPC*. https://www.gs1.at/strichcodes-rfid/strichcodes/eanupc.html. – [Abgerufen am 19. März 2020]

[5] Sakurambo: *Barcode*. https://de.wikipedia.org/wiki/Strichcode. – [Abgerufen am 19. März 2020]

Der Pfandflaschenautomat

Robin Kürbis, Elektrotechnik/Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Dieses Paper setzt sich mit eben diesem Pfandflaschenautomat auseinander. Der Roboter soll eine Pfandflasche zu einer Kamera transportieren, um den Barcode der Flasche zu erkennen und dann gegebenenfalls bei Erkennung die Flasche weiter zu transportieren oder sie wieder auszugeben. Nach einigen Schwierigkeiten ist es der Gruppe gelungen, einen Roboter mit diesen Funktionen fertig zu stellen**

***Schlagwörter*—Lego, Pfandflaschenautomat, Programierung, Projektseminar, Roboter,**

I. Einleitung

WIE in jedem Jahr fand das Lego Mindstorms Projektseminar der Otto von Guericke Universität statt. Studenten beschäftigten sich in Gruppen mit der Programmierung und Konstruktion von Robotern aus Lego. Der Pfandflaschenautomat ist eines dieser Projekte. Viele Maschinen begleiten und vereinfachen den menschlichen Alltag und der Pfandflaschenautomat ist eine davon. Er ermöglicht, so schnell wie möglich Pfandflaschen abzugeben. Damit hilft er nicht nur im Alltag, sondern trägt zum Umweltschutz bei, der heute wichtiger denn je ist. Ein solcher Roboter hilft bei der Erkennung der zum Recycling brauchbaren Flaschen. Das bedeutet, es muss ein Roboter gebaut werden, der erkennt, wenn eine Flasche eingeworfen wird und prüft, ob es sich um eine Pfandflasche handelt. Der Roboter soll mithilfe eines Laufbandes die Flaschen transportieren können, um dann mittels Kamera zu versuchen, einen Barcode zu erkennen, um dann selbst zu entscheiden, ob es sich um eine Pfandflasche handelt oder nicht.

II. Vorbetrachtungen

Zur Erkennung eines Barcodes benötigt der Roboter eine Kamera und zusätzlich Ultraschallsensoren, welche die Position der Flasche auf dem Laufband ermitteln sollen.

Abbildung 1. Beispiel für einen handelsüblichen Barcode (siehe Literaturverzeichnis [8])

DOI: 10.24352/UB.OVGU-2020-044

A. Funktionsweise Barcode

Ein Barcode besteht grundsätzlich aus schwarzen und weißen Streifen, den Guard Bars, die in bestimmten Verhältnissen zueinander stehen (siehe Abbildund1). Das bedeutet, dass verschiedene Schwarz-Weiß Verhältnisse verschiedene Informationen wiederspiegeln. Barcodes besitzen Right Guard Bars, Left Guard Bars und Middle Guard Bars, die den Messbereich festlegen. Es gibt verschiedene Barcodes, einige bestehen aus 13 Ziffern und andere aus nur acht. Eine Ziffer wird dabei immer durch das Verhältnis von sieben Streifen dargestellt.

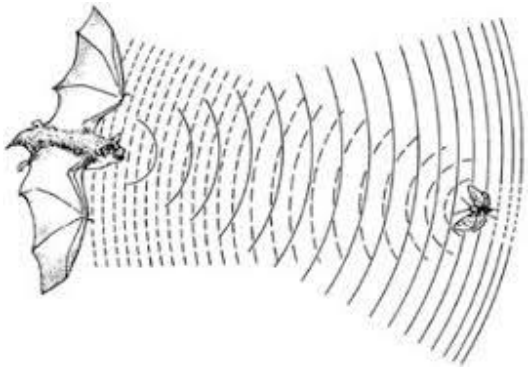


Abbildung 2. Ultarschallsensor aus der Natur (siehe Literaturverzeichnis [7])

B. Ultraschallsensoren

Ultraschallsensoren können Abstände zu beliebigen Gegenständen messen. Die Ultraschallsensoren funktionieren wie bei Fledermäusen in der Natur (siehe Abbildung 2). Der Sensor sendet eine Schallwelle aus, welche nach Auftreffen auf einen Gegenstand reflektiert wird. In Abhängigkeit von der Zeit der Wiederkehr wird der Abstand bestimmt.

C. Kamera

Mit einer integrierten Kamera erfolgt die Aufnahme von Fotos, hiernach die programmgestützte Barcodeerkennung.

D. Heutige Technik

Heutige Pfandflaschenautomaten verfügen über einen Laser, der die Flaschen scannt. Sie nehmen gescannte Flaschen auf und zerkleinern sie. Wie jeder wahrscheinlich schon persönlich erfahren hat, sind diese Automaten auch nicht fehlerfrei und geben Fehlermeldungen aus, obwohl eine Pfandflasche eingesteckt wurde. Die Erkennbarkeit kann schon durch leichte Verformungen eingeschränkt sein. Sie werden entweder in die richtige Position durch von unten kommende Rollen gedreht. Neuere Modelle verfügen über einen Laser, der 360 Grad abscannen kann.

III. Hauptteil

A. Konstruktion

Der Roboter verfügt über zwei Motoren. Motor A dient zur Ansteuerung des Laufbandes, welches die eingeworfenen Flaschen zur Kamera transportiert, um sie dann je nach Ausleseergebnis vorn oder hinten auszuwerfen. Die einzelnen Laufbänder sind durch eine Zahnrandkonstruktion jeweils nur durch einen Motor ansteuerbar. Der Motor muss je nach Situation das Laufband in zwei verschiedene Richtungen antreiben. Motor B dient zur Ansteuerung eines Rotationsmechanismus, welcher die Flasche im Falle einer Nichterkennung rotiert, um ein Ausleseergebnis erzielen zu können. Der Rotationsmechanismus besteht aus zwei Stangen, welche durch die Rotation unter die Flasche geraten und sie dadurch anheben. Auf der anderen Seite sind bewegliche Räder angebracht, die sich dadurch mitdrehen und so die Flasche Stück für Stück um ihre eigene Achse rotiert. Eine Führung sorgt dafür, dass die Flaschen eingeworfen werden können und beim Transport nicht vom Laufband fallen.

B. Sensorik

Der Aufbau besteht aus drei Ultraschallsensoren und einer Webcam. Der erste der drei Ultraschallsensoren ist auf den Anfang des Laufbandes gerichtet und dient zur Erkennung, ob eine Flasche eingeworfen wird. Er befindet sich seitlich am Aufbau. Der zweite Sensor befindet sich über dem Laufband und misst von oben. Dieser Sensor bindet sich in der Mitte des Laufbandes und soll ebenso erkennen, ob sich eine Flasche auf dem Laufband befindet oder nicht. Der dritte Ultraschallsensor dient genau wie die anderen beiden zur Erkennung, aber er hat noch eine andere Aufgabe. Sobald er eine Flasche erkennt, soll das Laufband gestoppt werden, um die Flasche in die richtige Position für den letzten Sensor zu bringen. Auch dieser Sensor befindet sich seitlich am Aufbau. Die Kamera befindet sich über dem Laufband und macht Fotos von oben. Die Fotos sollen dann zur Auslesung des Barcodes dienen. Auf die genaue Funktionsweise wird im nächsten Abschnitt eingegangen.

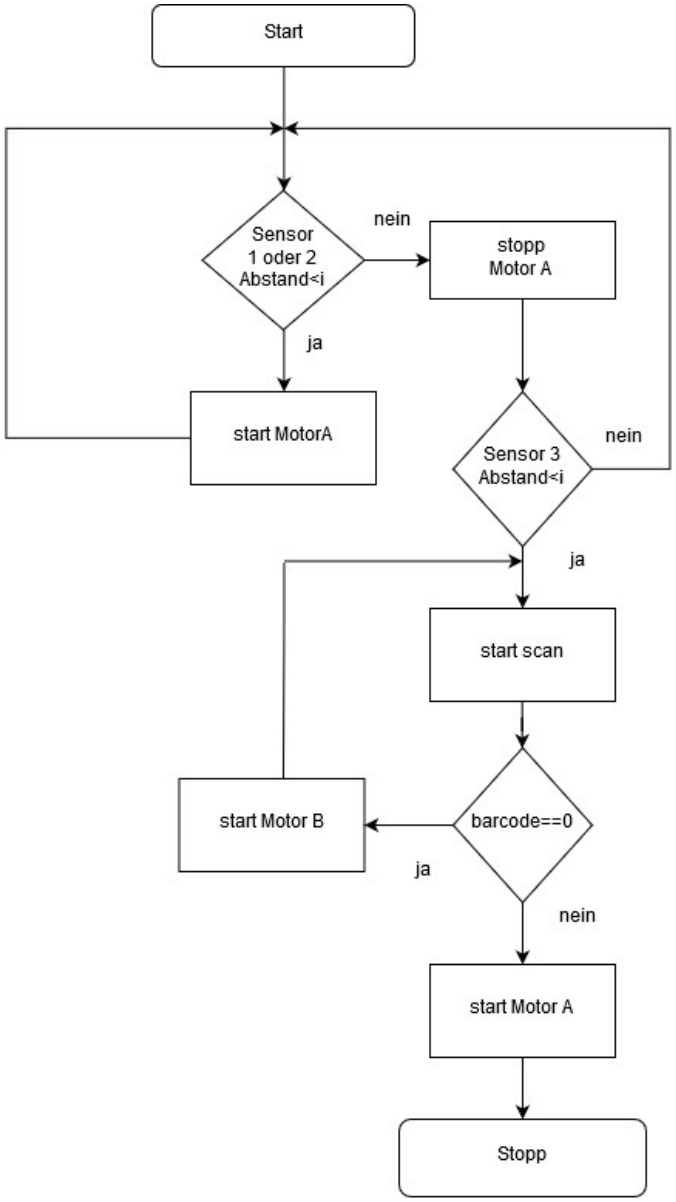


Abbildung 3. Programmablaufplan

C. Funktionsweise des Programmes

Die Programmierung des Roboters erfolgte über MATLAB. Für die Barcode-Erkennung wurde jedoch eine Bibliothek aus Java hinzugezogen. Sobald das Programm gestartet wird, beginnen die Ultraschallsensoren zu messen. Wenn der Messewert unter einen bestimmten Wert fällt, startet Motor A, der das Laufband antreibt (siehe Abbildung 3). Antriebskraft (Power) und Antriebsgeschwindigkeit wurden in einer Funktion festgelegt. Sobald die Messwerte nicht mehr unter diesem Wert liegen stoppt Motor A wieder. Dieses Verfahren läuft solange, bis die Messwerte des Ultraschallsensors am Ende des Laufbandes unter einen bestimmten Wert fallen. Ist das der Fall, stoppt Motor A, ein Zähler wird einen Wert hochgesetzt und die Kamera beginnt ein Foto aufzunehmen. Sobald das geschehen ist, beginnt ein Programm den Barcode auszulesen. Gelingt das, wird der Zähler wieder auf Null gesetzt, der Barcode wird ausgegeben und Motor A gestartet, welcher die Flasche hinten auswirft. Nun kann eine neue Flasche eingeworfen werden. Falls der Barcode nicht beim ersten Mal ausgelesen werden kann, wird Motor B gestartet, der den Rotationsmechanismus antreibt. Antriebskraft (Power), Antriebsgeschwindigkeit und Umdrehungen für Motor B wurden wieder in einer Funktion festgelegt. Die Flasche soll dadurch so in Position gebracht werden, dass der Barcode nach oben zeigt und ausgelesen werden kann. Funktioniert dies wieder nicht, wird der Zähler wieder um einen Wert hochgesetzt und Motor B wieder gestartet. Dieser Vorgang wird bis zu einer bestimmten Zählerzahl durchgeführt, sollte dann noch kein Auslesevorgang erfolgreich gewesen sein, wird Motor A gestartet und somit die Flasche vorne wieder ausgeben. Der Zähler wird wieder auf Null gesetzt und es kann eine neue Flasche eingeworfen werden. Durch die Einbindung von zwei Scanprogrammen funktioniert es sowohl für achtziffrige (EN8) und dreizehnziffrige (EN13) Barcodes.

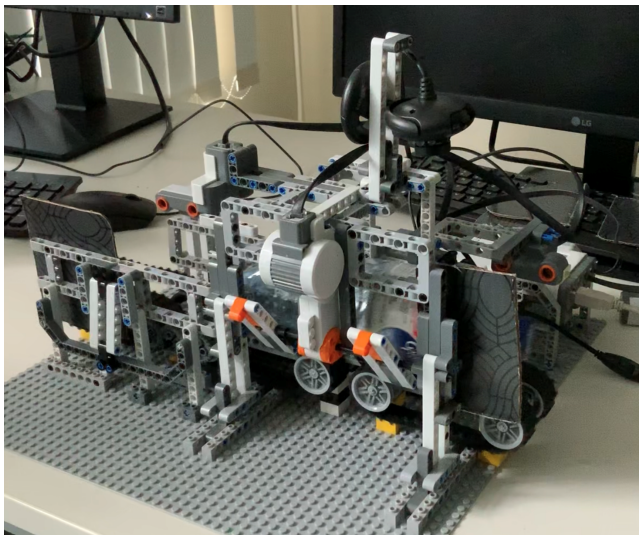


Abbildung 4. Pfandflaschenautomat

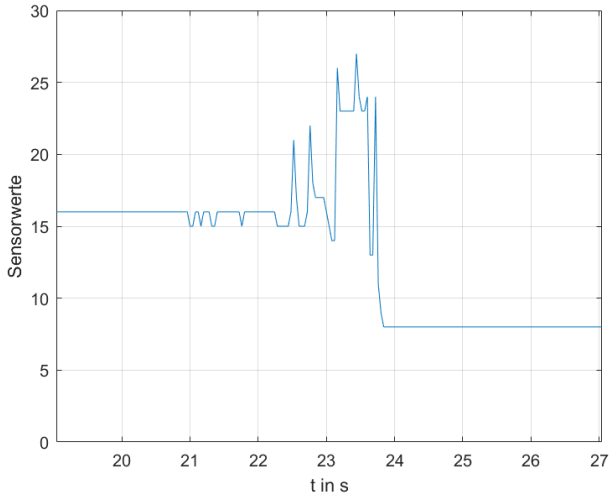


Abbildung 5. Ankunft einer Flasche am hinteren Ultraschallsensor

IV. Herausforderungen

Die Lichtverhältnisse, die es immer wieder erschwert haben, den Barcode zu erkennen, stellten eine der Herausforderderungen in der Entwicklung des Pfandflaschenautomaten dar. Durch den Lichteinfall kam es wiederholt zu Reflektionen auf dem Barcode, die die Barcodeerkennung verhindert haben. Deshalb musste des Öfteren mit einer Abdeckung gearbeitet werden, die den Lichteinfall auf das Nötigste reduzierte. Des weiteren führten Messungenauigkeiten der Ultraschallsensoren zu Problemen. Aufgrund der gewölbten Oberfläche der Flaschen wurden die Schallwellen unterschiedlich zurückgeworfen und so falsche Messwerte ausgeben. Um den Ungenauigkeiten nachzukommen, wurden Wände aus Pappe gegenüber den Sensoren angebracht (siehe Abblidung 4). Die Ausschläge im Diagramm zeigen deutlich die Schwankungen der Messwerte bei der Ankunft einer Flasche (siehe Abbildung 5). Die Konstanten wiederum zeigen den Abstand zur Wand und nach den Änderung den Abstand zur Flasche. Es wird deutlich, dass der Sensor bei der Ankunft der Flasche unter einen Wert fällt (siehe Abbildung 5) und bei Messungenauigkeiten Motor A zu früh oder zu spät stoppt.

V. Ergebnisdiskussion

Am Ende des Seminars konnte ein funktionsfähiger Roboter vor den anderen Teilnehmern des Seminars präsentiert werden. Er war in der Lage, nach Einwurf der Flasche diese zu transpotieren, einen Barcode zu scannen und zu entscheiden ob ein Code erkannt wurde und je nach Ergebnis die Flasche aus zu geben. Außerdem war es möglich, die Flasche nicht nur behutsam einzulegen, sondern sie einzuwerfen. Es spielte dabei keine Rolle, ob der Barcode dabei nach oben ausgerichtet wurde, denn der Rotationsmechanismus ermöglichte es, die Flasche in die Richtige Position zu bringen. Natürlich lief nicht alles reibunglos, es traten immer wieder einige Probleme auf. Das größte Problem war der Lichteinfall, der teilweise für die Kamera zur Erkennunglosigkeit des Barcodes geführt hat, dadurch wurde die Flasche einfach wieder vorne ausgegeben. Es kam auch gelegentlich dazu, dass der Rotationsmechanismus, der eigentlich die Flasche in Position bringen sollte, das Gegenteil tat und der Barcode aus dem Erkennungsbereich geriet. Der Sensor, der bei Erkennung einer Abstandsänderung dazu führte, dass das Laufband stoppte, um die Flasche in der richtigen Position zu halten, führte durch Messungenauigkeiten teilweise zum Gegenteil. Ein selten auftretendes Problem war, dass die Flasche während des Transportes hängenblieb und somit nicht zum Auslesepunkt gelangte. Aufgrund der Größe unseres Roboters ist er nur in der Lage, kleine Flaschen aufzunehmen.

VI. Zusammenfassung und Fazit

Der Roboter kann Barcodes von Flaschen scannen, die eingeworfen werden. Die Arbeit an dem Projekt hat gezeigt, was man schon aus LEGO und mit ein weinig Programmierarbeit aufbauen kann. Sie hat außerdem die Arbeit zwischen den einzelnen Gruppenmitglieder gestärkt und hat ihnen durch Spaß an dem Projekt eine neue Programmiersprache näher gebracht. Es bietet vor allem eine Abwechslung zum üblichen Alltag an der Uni. Mögliche Verbesserungen für den Roboter wären die Minimierung der Laufzeit des Programmes, denn es dauert seine Zeit, bis ein Foto aufgenommen und ausgewertet wurde. Es wäre außerdem möglich, ein GUI zu erstellen, welches dem Benutzer am Ende seiner Flaschenabgabe anzeigt, welchen Betrag er erhalten würde. Von Vorteil wäre auch eine tunnelartige Konstruktion, wie bei echten Pfandflaschenautomaten, welche den Lichteifall begrenzt. Man könnte den ganzen Aufbau in einem größeren Format konstruieren, um auch große Flaschen scannen zu können. Dafür wäre aber eine viele größere Menge an Legoteilen erforderlich. Der Aufbau lässt noch Spielraum für einige Verbesserungen.

Literaturverzeichnis

[1] Institute of Imaging Computer Vision: *Mindstorms NXT Toolbox*. https://www.mindstorms.rwth-aachen.de/trac/wiki/Download4.07. Version: März 2020

[2] WIKIPEDIA, THE FREE ENCYCLOPEDIA: *Strichcode*. https://de.wikipedia.org/wiki/Strichcode. Version: März 2020
[3] MATHWORKS: *Barcode Recognition*. https://www.mathworks.com/help/vision/examples/barcode-recognition.html. Version: März 2020
[4] STUPIDEDIA, DIE SINNFREIE ENZYKLOPÄDIE: *Pfandflaschenautomat*. https://www.stupidedia.org/stupi/Pfandflaschenautomat. Version: März 2020
[5] MATHWORKS: *Barcode Reader*. https://www.mathworks.com/matlabcentral/fileexchange/31727barcode-reader. Version: März 2020
[6] MATHWORKS: *Computer Vision Toolbox*. https://www.mathworks.com/help/vision/examples/barcoderecognition.html. Version: März 2020
[7] SPEKTRUM.DE: *Echoorientierung*. https://www.spektrum.de/lexikon/biologie-kompakt/echoorientierung/3359. Version: März 2020
[8] TEC-IT: *FAQ Strichcode Parameter*. https://www.tec-it.com/de/support/faq/barcode/bar-code-config/Default.aspx. Version: März 2020
[9] ENGEL AUTOMATEN+TECHNIK: *Rücknahmeautomaten für Pfand-Leergut*. https://www.engeltec.de/produkte/ruecknahmeautomaten/index.php. Version: März 2020

Sortierroboter

Rifat Alkabariti, ETIT
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Technologie ist die Zukunft und wir bereiten uns darauf vor. Die Lego-Roboter können heutzutage viele Aufgaben tätigen, deswegen sollte dieser Bereich allen Studenten beigebracht werden. Unsere Universität hat mir die Chance und die Motivation gegeben, einen Roboter zu bauen und herauszufinden, wie es gemacht wird. Mit Hilfe eines Colleges wurde einen Sortier-Roboter gebaut, der die Objekte je nach Farbe des Objekts an der richtigen Stelle sortiert.**

I. Einleitung

In diesem Artikel wird es diskutiert, wie der Roboter mechanisch gebaut wurde und wie er programmiert wurde. Es wird auch auf die konfrontierten Herausforderungen eingehen, und wie sie gelöst wurden. Am Anfang musste man lernen, was MATLAB ist, und wie es funktioniert und wie man einen NXT damit programmiert. Danach wurde es beschlossen, einen Sortier-Roboter zu bauen.

II. Aufbau des Roboters

Drei Motoren wurden für die Bewegung verwendet, einen (Motor A) zum Greifen der Objekte und einen (Motor B) zum Anheben und einen (Motor C) zum Drehen des Arms an die richtige Stelle. Ein Farbsensor wurde auch gebraucht, um die Farbe der Objekte zu erkennen.

Für den Aufbau wurden Legosteine, die die Flexibilität der Aufbau ermöglicht, verwendet. In der Basis befindet sich der Farbsensor, sowie der erste Motor (Motor A), der für die Rotation des Arms zuständig ist. Für das Heben und Senken der Klaue wurde ein weiterer Motor (Motor B) direkt über der Drehachse befestigt. Da durch den langen Arm eine große Hebelwirkung der Klaue auf den Motor entsteht, war es auch hier nötig, eine Übersetzung mit Zahnrädern zu verwenden. Die Klaue wurde so am Arm befestigt, dass sie immer vertikal zum Boden steht. Eine Schraube greift in die zwei Zahnräder der Gabeln, die dann eine entgegengesetzte Drehung vollziehen und sich damit zueinander oder voneinander bewegen. Dadurch werden das Öffnen und Schließen der Kralle realisiert.

III. Programmablauf

Der Roboter wurde hergestellt, um die Objekte in Abhängigkeit von der Farbe zu sortieren. Zu Beginn wird das Objekt in den Anfangspunkt eingefügt. Der Sensor erkennt die Farbe des Objekts und gibt den Motoren den Startbefehl. Das Objekt wird durch Motor A gegriffen und durch Motor C zum seinem bestimmten Platz gebracht. Motor C dreht sich 90° für ein blaues Objekt, 120° für ein rotes Objekt und 150° für grünes Objekt. Der Programmiercode für diesen Roboter

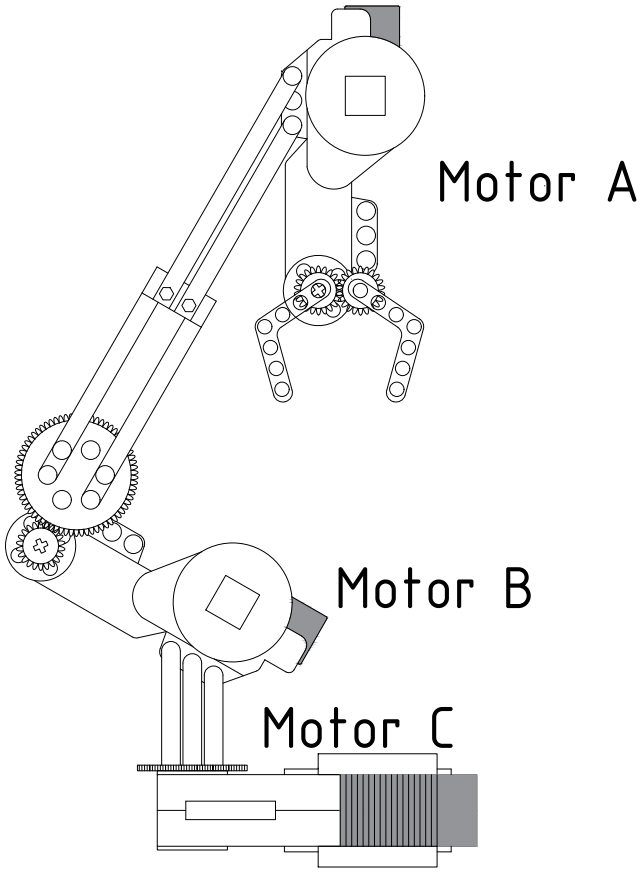


Abbildung 1. Aufbau des Armes

hängt aufgrund der häufigen Wiederholung hauptsächlich von der Reihenfolge (while) und (if) ab.

Es wurde auch ein User-Interface verwendet, um den Roboter manuell and automatisch zu steuern. Die obigen Schaltflächen geben Befehle zum Sortieren der Objekte und die seitlichen Befehle zum Sortieren der Schüsseln, in der die Objekte sortiert werden. In der Mitte befindet sich die automatische Taste, mit der der Roboter den Programmplan automatisch starten kann.

Um die Motordrehung zu steuern, wurden einige Befehle so geschrieben, dass sich der Motor drehen kann, bis er nicht mehr kippen kann, und der Motor eine weitere Drehung erhält, wenn der Rotationswinkel für beide Male gleich ist, dann wird der Motor diesen Winkel als seine Tacho-Grenze betrachten.

IV. Probleme

Die wichtigsten mechanischen Probleme waren die Stabilität und die Genauigkeit. Der Arm am Anfang zeigte nicht seine beste Leistung, weil der Greifmotor zu schwer war, so dass der

DOI: 10.24352/UB.OVGU-2020-045

Arm so geändert wurde, dass er kürzer und näher an der Basis war. Außerdem wurden zwei Gummibänder an den Seiten der Basis angebracht, damit sich der Robot nicht bewegt, wenn sich der Arm selbst dreht. Es wurden auch Gummibänder an der Klaue angebracht, um ein Herunterfallen des Objekts zu vermeiden und die Objekte besser zu greifen.

Auch die Kabel waren zu kurz, so dass sich der Arm nicht frei und weit drehen konnte. Um dieses Problem zu lösen, wurde der NXT vertikal mit der Basis verbunden, sodass die Eingänge des Motorkabels näher am Motor liegen.

Aufgrund der technischen Realisierung des Sensors, muss er sich wenig Zentimeter über dem Objekt befinden, um die Farbe zu erfassen.

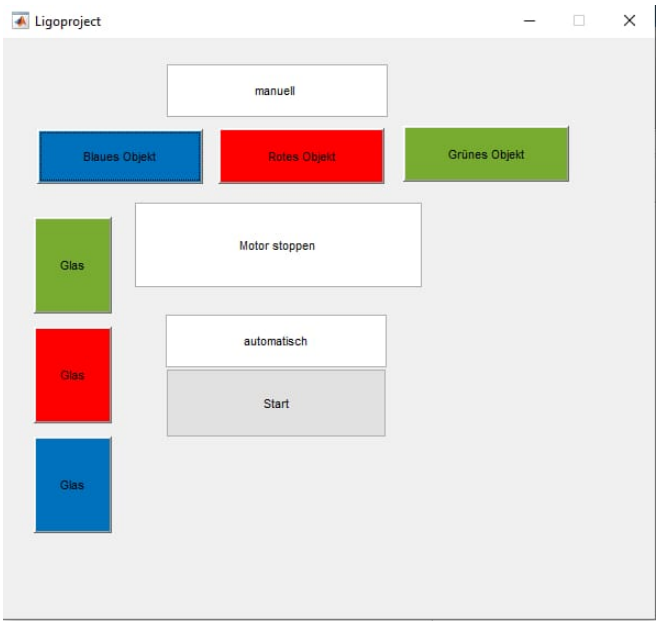


Abbildung 2. User-Interface

V. Ergebnisdikussion

Der Roboter war in der Lage, den selbst gewählten Anforderungen zu entsprechen: Er konnte die Farbe der Objekte selbstständig erfassen und war damit flexibel in der Ausgangssituation, die Objekte wurden in der Farbe fast immer richtig erfasst, wobei Ausnahmen nur durch die Ungenauigkeit der verbauten Sensoren auftraten und die Objekte in den dafür vorgesehen Schüssel gelegt werden konnten. Auch beim Legen gab es selten Probleme, da sich die Motoren nicht immer um den denselben Winkel gedreht haben.

VI. Anwendungsmöglichkeiten

Dieser Roboter soll die aufwendige Arbeit des Aufräumens/sortieren übernehmen, womit Zeit gespart wird, die in andere Beschäftigungen gesteckt werden kann. Er ist vielfältig einsetzbar und kann verschiedene Prozesse im Alltag übernehmen, wie das Sortieren der Wäsche oder Kinderspielzeugen. So kann schnell und einfach Chaos minimiert werden und Wichtiges vom Unwichtigen getrennt werden. Dieser Roboter kann in zu vielen Bereichen eingesetzt werden, zum Beispiel kann es die zweite Hand eines Technikers sein, die ihm hilft, die kleinen Objekte an ihren Plätzen zu sortieren, oder um die Müllsäcke nach ihren Farben zu sortieren.

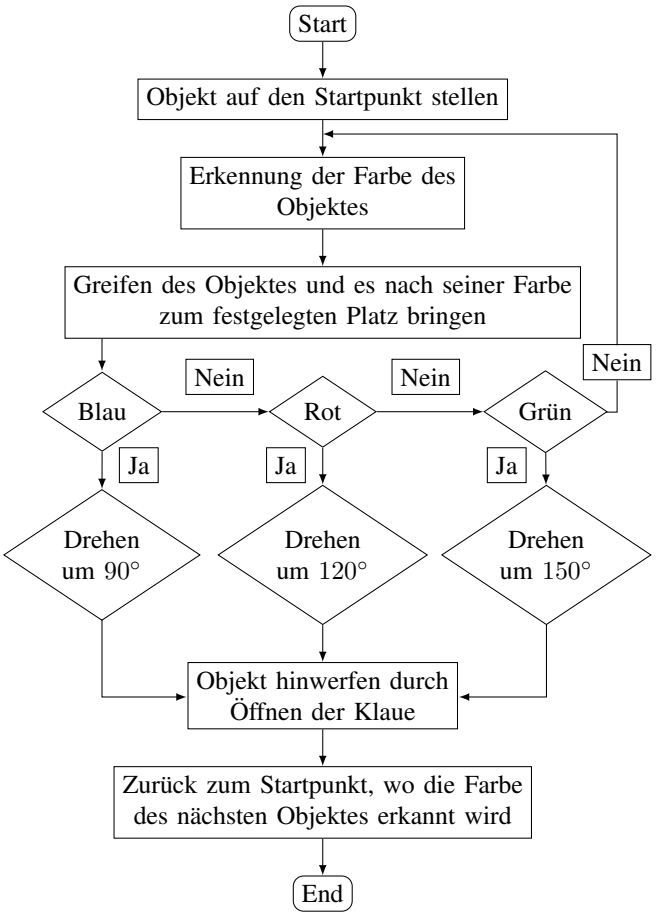


Abbildung 3. Programm-Ablauf-Plan

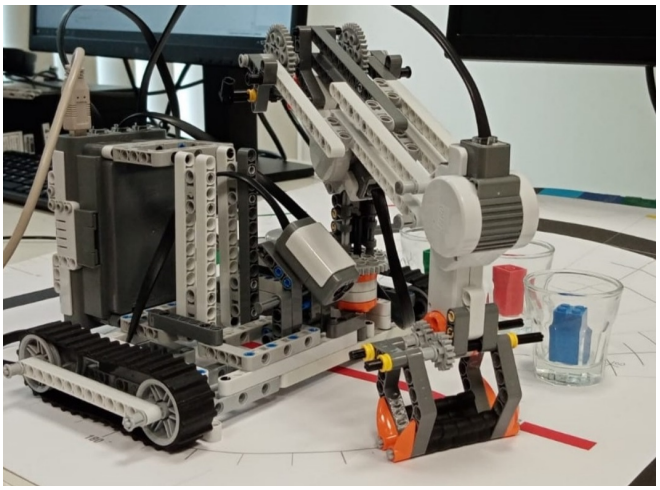


Abbildung 4. Sortier-Roboter-Aufbau

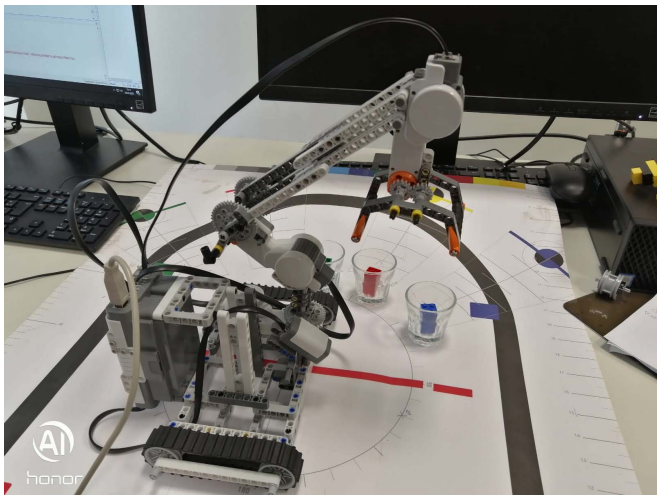


Abbildung 5. Sortier-Roboter-Aufbau

VII. Zusammenfassung Und Fazit

Durch das Projekt wurde eine neue Programmiersprache gelehrt,und durch die geförderte Zusammenarbeit zwischen den Kommilitonen lassen wir uns gegenseitig bei Problemen helfen. Wenn das Projektseminar länger als zwei Wochen könnte man eine komplexere Maschine bauen, auch wenn es manchmal die Menge an Legoteilen nicht zulässt. Den Farbsortiere könnte man in größere Dimensionen übertragen und ihn aus richtigen Bestandteilen der Industrie bauen umso auch andere Gegenstände zu scannen und zu sortieren. Lego -Mindstorms ist eine gute Möglichkeit um Kinder und Jugendliche das Programmieren beizubringen und sie auf einen Weg bringen, der im späteren Leben noch sehr nützlich wird .

Literatur

[1] Bruno Siciliano, Oussama Khatib: Springer Handbook of Robotics. Springer-Verlag, Berlin 2008, ISBN 978-3-540-23957-4, https://de.wikipedia.org/wiki/Robotik.

[2] FEIT OvGU (2019- February)[Online], Unterlagen zum LEGO Mindstorms Praktikum in der FEIT

Sortierroboter

Haytham N. H. Darawish, ETIT
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In diesem Paper handelt es sich um die Entwicklung und Konstruktion eines Sortierroboters mit LEGO Mindstorms. Im Rahmen des Seminarprojektes wurde versucht, einen Sortierroboter zu bauen, der Objekte anhand der Farbe von einem Punkt zum anderen bringt. Der Aufbau, die Funktionsweise sowie das Programieren werden in diesem Paper erklärt. Es wird außerdem auf die Probleme, die aufgetaucht sind, eingegangen. Abschließend werden die möglichen Anwendungsbereiche und das Endergebnis des Projektes vorgestellt.**

I. Einleitung

Mit der Entwicklung der Technologie sind wir in einer Zeit angekommen, in der das Nutzen von programmierten Robotern erheblich angestiegen ist, wodurch der Mensch als Arbeitskraft durch diese Robeter nach und nach ersetzt wird, was dazu geführt hat und führen wird, dass manche Arbeitsstellen vollständig von Robotern besetzt werden, wodurch die Weiterentwicklung und das Ausbauen von Robotern in deren Fähigkeiten und Eigenschaften nötig werden. Der in diesem Paper vorgestellte Roboter ist ein sehr gutes und realistisches Beispiel, denn dieses Gerät kann Objekte rund um die Uhr bewegen und verfrachten, ohne einen höheren Lohn anzufordern, was die Einstellung von Menschen unnötig macht, was viele Unternehmen bevorzugen. Er muss nur von Menschen gebaut und programiert werden.

II. Vorbetrachtungen

Nachdem die Grundlagen von MATLAB gelernt wurden, war die Aufgabe für jede Gruppe, die zwei bis drei Personen hat, eine Idee zu finden, an der gearbeitet wird. Jede Gruppe hat eine Lego-Mindstorms-Kiste, die ganz viele Bauteile für ganz unterschiedliche Roboter bietet, bekommen. Die Lego-Mindstorms-Kiste enthält drei Gleichstrommotoren, Tastsensor, Ultraschallsensor, Lichtsensor, Farbsensor, Räder, Verbindungsstangen usw. sowie den NXT-Baustein. Der NXT-Baustein bildet das zentrale Bauelement jedes Roboters und ist dessen "Gehirn". Mit dem NXT-Baustein können alle zu steuernden Bauteile (Gleichstrommotoren, Tastsensor, Ultraschallsensor, Lichtsensor, Farbsensor) des Roboters per Kabel verbunden und gesteuert werden. Dafür werden in MATLAB Programme am Computer erstellt und über USB-Kabel auf den NXT-Baustein übertragen [1]. Bevor man sich auf ein Thema festgelegt hat, waren einige Aufgaben zu lösen, die zur Ansteuerung von NXT gehören. Nachdem diese Aufgaben bearbeitet wurden, kam die Zeit ein Thema für das Projekt zu finden. Die Idee war, einen Sortierroboter zu entwickeln. Dafür wurde überlegt, was für Bauteile für den Aufbau benötigt werden. Angefangen wurde mit der Konstruktion des Roboters. Für die Konstruktion des Roboters wurden von den steuernden Bauteilen der Lego-Mindstorms-Kiste drei Motoren und ein Farbsensor verwendet. Bevor angefangen wurde, ein Programm zu schreiben, mit dem NXT-Baustein programmiert wird, wurde ein Programmablauf erstellt (siehe Abbildung 3).

DOI: 10.24352/UB.OVGU-2020-046

III. Konstruktion des Roboters

Der Roboter weist drei Bauteile auf, die Klaue, den Arm und die Basis. In Abbildung 1 ist eine schematische Darstellung des Roboters gezeigt.

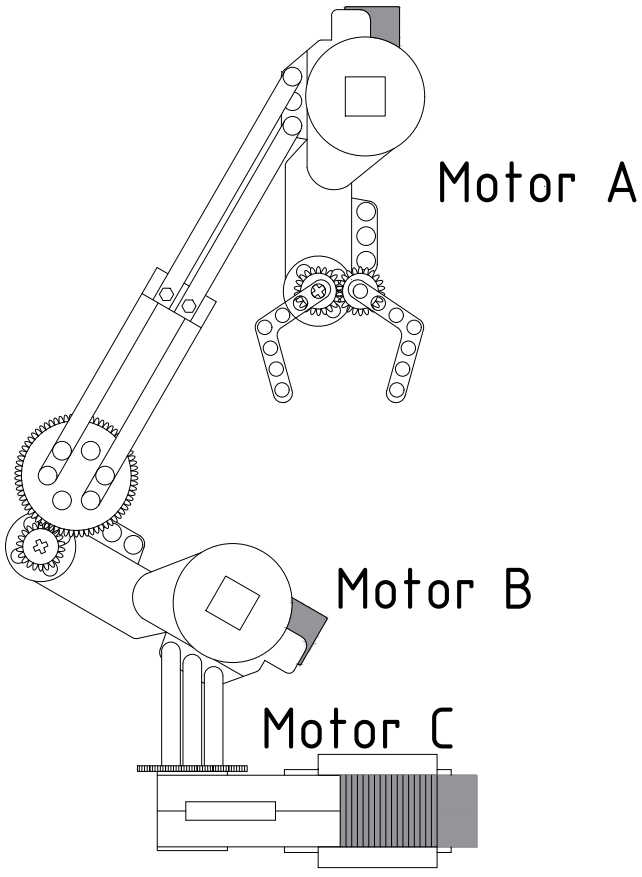


Abbildung 1. Einfache schematische Darstellung des Roboters

1) Klaue
   Die Klaue ist das einfachste Bauteil des Roboters. Sie besteht grundsätzlich aus einem DC-Motor (Motor A in Abbildung 1) und einem einstufigen Stirnradgetriebe. Das einstufige Stirnradgetriebe besteht aus zwei Wellen. Das erste Zahnrad ist direkt mit der Motorwelle verbunden. Das zweite Zahnrad sitzt auf der anderen

Welle und ist zu dem ersten Zahnrad parallel gerichtet. Die zwei Zahnräder greifen also ineinander. Bei Drehung des Motors A dreht sich auch das mit der Motorwelle verbundene Zahnrad. Damit werden die Drehzahl und das Drehmoment des ersten Zahnrades auf das zweite Zahnrad übertragen. Außerdem ändert sich der Drehsinn der Zahnräder. Rotiert also bspw. das mit der Motorwelle verbundene Zahnrad gegen den Uhrzeigersinn, so wird sich das mit ihm verzahnte Zahnrad im Uhrzeigersinn drehen [2]. Damit lässt sich die Klaue durch Steuerung des Motors A öffnen und schließen.

2) Arm
Der Arm ist das zweite Bauteil des Roboters. Er liegt zwischen der Klaue und der Basis des Roboters und besteht aus einem Motor (Motor B Abbildung 1) und zwei Stirnradgetrieben. An die Motorwelle sind zwei kleine Zahnräder direkt angeschlossen. Die Zentralhand ist mit zwei größeren Zahnrädern verbunden, die zu den mit der Motorwelle verbundenen kleinen Zahnrädern parallel sind. Wie bei der Klaue übertragen sich hier bei Drehung des Motors B die Drehzahl und das Drehmoment der kleinen Zahnräder auf die größeren Zahnräder. Die Zentralhand lässt sich damit durch Steuerung des Motors B nach oben oder nach unten bewegen.

3) Basis
Der Hauptteil der Basis ist der Motor C, der für die Kreisbewegung von dem Arm und der Klaue zuständig ist. Weil der Arm und die Klaue schwer wiegen, war die Basis das schwerste Bauteil bei dem Aufbau des Roboters. Der Arm wurde direkt an die Motorwelle (Motor C) angechlossen. Somit entsprach die Güte des Prozesses nicht 100% und folglich konnte das Objekt nicht immer direkt ins Glas geworfen werden. Der Roboter hat in einem von sechs Fällen das Ziel verfehlt. Zur Stabilitätssteigerung wurden zwei Gummiketten an der Basis fixiert.

IV. Programmablauf

Um die Steuerung per Mausklick ausführen zu können, wurde in MATLAB das in Abbildung 2 gezeigte Graphical User Interface erstellt, sodass man keine Befehle eingeben muss. Die erstellte Graphical User Interface verfügt über zwei Optionen, die ”Manue” und ”Automatik” genannt werden. Für die manuelle Steuerung wurde für jedes einzelne Objekt ein Programm geschrieben, sodass man überprüfen kann, ob alle Objekte richtig plaziert sind. Das Gleiche gilt auch für die Gläser. Nachdem es überpruft wird, dass alle Objekte bei der manuelen Ausführung das Ziel getroffen haben, kommt der Einsatz des Farbsensors. Mithilfe des Farbsensors wurde eine Fallunterscheidung erstellt, sodass der Roboter automatisch die Objekte nach deren Farben sortiert.

Erst vor dem Starten des Programms wurden, wie schon erwähnt, die drei Gläser manuell durch das Userinterface an die richtige Stelle platziert. Das Objekt muss nach dem Starten

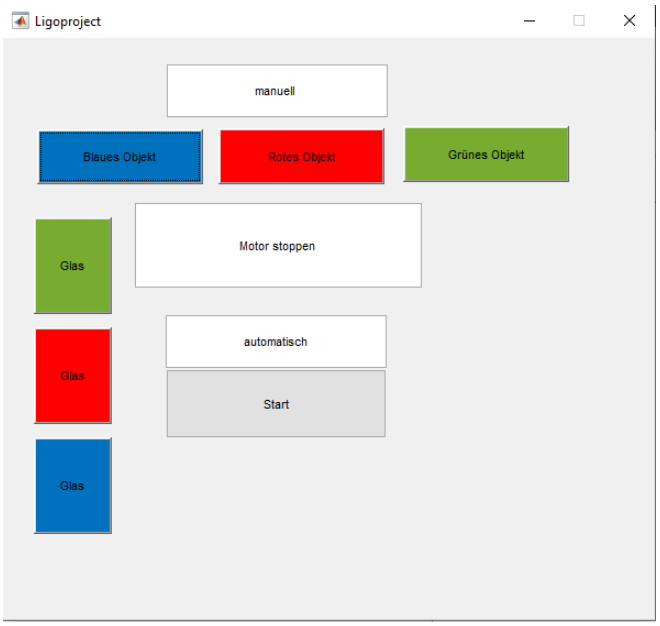


Abbildung 2. Graphical User Interface

des Programmes auf den Startpunkt, da wo der Farbsensor festgelegt ist, gestellt werden. Die Farbe wird zunächst von dem Farbsensor erkannt. Nach der Erkennung der Farbe des Objektes, wird es gegriffen und zu dem festgelegten Platz gebracht. Nun ist eine Fallunterscheidung notwendig. Ist die Farbe des Objektes Blau, dreht sich der Motor C um 90 Grad. Ist die Farbe des Objektes Rot, dreht sich der Motor C um 120 Grad. Ist die Farbe des Objektes Grün, dreht sich der Motor C um 150 Grad. Nachdem der Motor C den Arm zum festgelegten Platz gedreht hat, wird das Objekt durch Öffnen der Klaue geworfen. Dannach dreht sich der Arm wieder zum Startpunkt zurück, wo die Farbe des nächsten Objektes erkannt wird. Der Vorgang wird solange wiederholt, bis das Programm beendet wird.

V. Probleme

Weil nur ein an der vorderseite angelegter Farbsensor verwendet wird, muss das Objekt auf den Startpunkt gestellt werden. Das heißt, dass der Bewegungsumlauf in Bezug auf den Startpunkt ist. Dieses Problem wurde gelöst, indem die 3 Glässer durch das User Interface vor dem Starten auf den entsprechenden Stellen platziert werden. Ein weiteres Problem ist die Genauigkeit des Farbsensors. Um zu vermeiden, dass die Farbe des Objektes nicht richtig erkannt wird, muss das Objekt nah genug am Sensor sein. Das dritte Problem ist die Stabilität bei Bewegung und Greifen.

VI. Verbesserung

Um das erste Problem, dass der Bewegungsumlauf in Bezug auf den Startpunkt ist, lösen zu können, muss der Aufbau des Roboters und der Programablauf noch erweitert werden. Die umzusetzende Idee, die nicht geschafft wurde, war, dass der Roboter erst die Gläser sucht, in die die verschiedenen

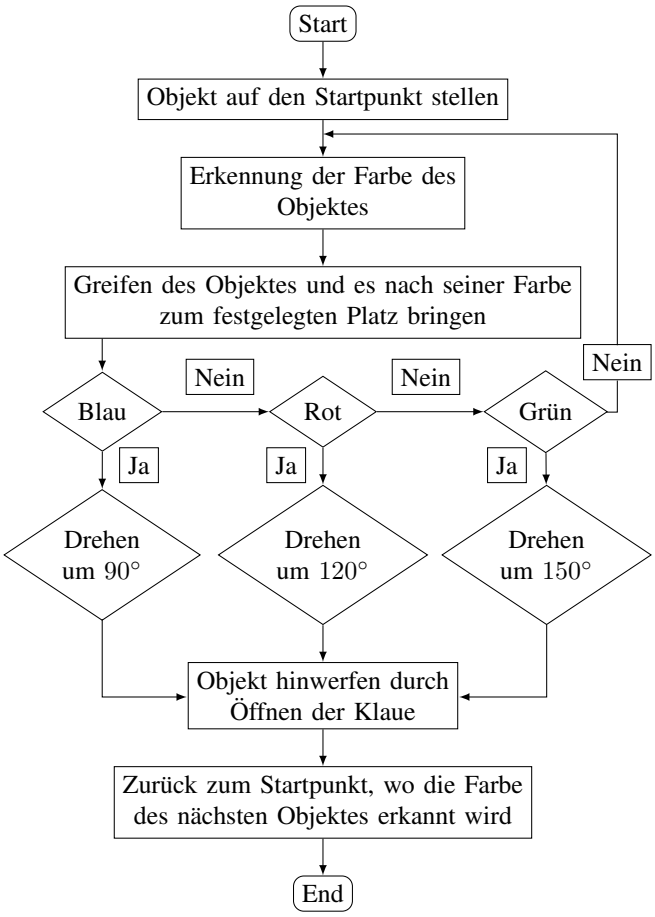


Abbildung 3. Programmablauf für die automatische Steurung

Objekte geworfen werden. Dafür muss man weitere Sensoren verwenden.

Um die Genauigkeit des Farbsensors zu verbessern, kann man statt nur einen besser zwei Sensoren verwenden und ihre Werten vergleichen. Falls die Beiden den selben Wert ausgeben, stellt man fest, dass die Farbe des Objektes sicher erkannt wurde.

Für die Stabilitätsteigerung können mechanische Änderungen am Roboter notwendig sein.

VII. Einsatzgebiete

Die Sortierroboter können in zahlreichen Bereiche verwendet werden, vor allem für routinierte Aufgaben, bei denen man die gleichen Aufgaben wiederholen muss.

Im Folgenden werden einige der Einsatzgebiete genannt.

- Landwirtschaft
  In der Landwirtschaft wird der Mechanismus des Roboters prinzipiell in Mähmaschinen eingesetzt, womit der Vorgang des Erntens durch Sensoren automatisiert wird.

- Lagersortierung
  Um den Prozess der Lagersortierung und Kommissionierung der Produkte zu optimieren, kommen solche Roboter zum Einsatz.

VIII. Endergebniss

Es wurde im Rahmen dieses Seminarprojektes geschafft, einen Sortierroboter zu entwickeln, der Objekte (kleine farbige Legosteine) durch einen Farbsensor erkennt und sie an die richtige Stelle bringt, wo mehrere Behälter (Gläser) platziert und für die drei verschiedenen Farben gedacht sind. Der Roboter hat also trotz der schon genannten Probleme seine zu erwartende Aufgabe erfüllt. Dennoch muss die Konstruktion des Roboters noch verbessert werden, da der Roboter noch nicht genug stabil ist. Mit Verwendung von mehreren und besseren Sesoren kann der Roboter die Farben ohne menschliches Eingreifen (den Legostein näher an den Sensor bringen) erkennen.

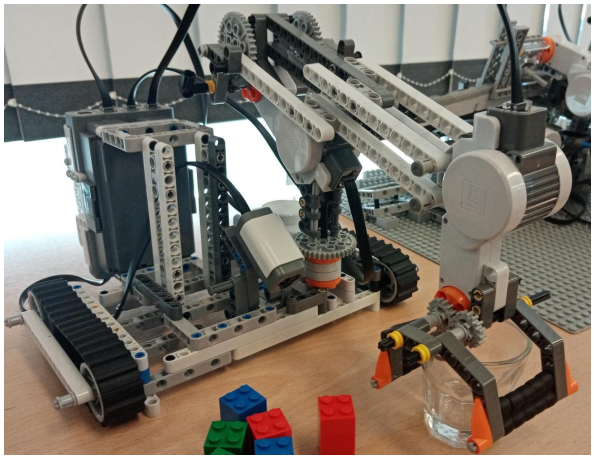


Abbildung 4. Sortierroboter von vorne

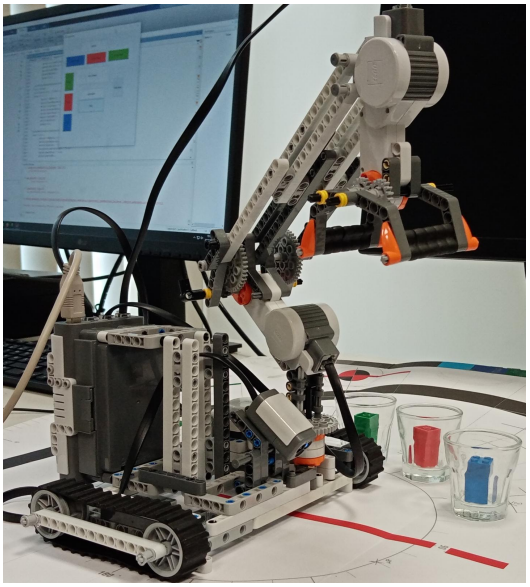


Abbildung 5. Sortierroboter von der Seite

LITERATUR

[1] zdi Netzwerk DU.MINT Duisburg Niederrhein: LEGO Mindstorms EV3-Roboter-Sets, https://www.uni-due.de/du-mint/lego-mindstorms, Eingesehen am 01.03.2019

[2] TEC-Science: Getriebetechnik, Grundlagen, Funktionsweise, https://www.tec-science.com/de/getriebetechnik/grundlagen/funktionsweise/, Eingesehen am 02.08.2018.

ANHANG

Während der Demonstration wurde ein Video aufgenommen und bei YouTube hochgeladen:

https://www.youtube.com/watch?v=s7ahCUFZtL0

Im Folgenden ist ein Beispielcode zum Schließen der Klaue gezeigt. Es wurde eine 1xn-Matrix, die im folgenden Beispielcode "Anglesss" genannt wurde, erstellt. In dieser Matrix werden die Motorwinkel registriert. Sobald die Klaue das Objekt greift, werden in der erstellten Matrix die gleichen Winkelwerte registriert, also der Winkel bleibt konstant. In diesem Fall wird die while-Schleifen mithilfe der break-Anweisung, die in Verbindung mit der if-Abfrage steht, ohne weiteres verlassen. Als nächstes wird der Motor gestoppt. Dann dreht sich der Motor nur um 1 Grad, um sicher zu stellen, dass das Objekt von der Klaue gegriffen ist.

```
COM_CloseNXT('all');
handle = COM_OpenNXT();
COM_SetDefaultNXT(handle);

disp('Klaueschliessen')
        motorA = NXTMotor('A', 'Power',-50);
        motorA.SpeedRegulation = false;
        motorA.SmoothStart = true;
        motorA.SendToNXT();
n=0;
while 1
    n=n+1;
    data = motorA.ReadFromNXT();
    Anglesss(n)=data.TachoCount;
    if n>3
        if Anglesss(end)==Anglesss(end-3)
            break;
        end
    end
    pause(0.05)
end
disp('Motor stoppen')
        motorA.Power=0;
        motorA.SendToNXT();
disp('Klaue noch mal etwas schliessen')
        motorA.Power=-20;
        motorA.TachoLimit = 1;
        motorA.ActionAtTachoLimit = 'Holdbrake';
        motorA.SendToNXT();
        motorA.WaitFor()
disp('Halten')
pause(0.5)

COM_CloseNXT(handle);
```

Erkundungsroboter

Omar Altarabeen,Elektrotechnik
Otto-von-Guericke-Universität Magdeburg

I. Einleitung

IN den letzten Jahren hat die Nutzung von persönlichen Assistenten erheblich zugenommen und die Wirkung auf Roboter grundlegend verändert. Das Projekt behandelt als Problemstellung Roboter, die sich insbesondere mit dem Aspekt der künstlichen Intelligenz auseinandersetzt. Das Projekt hat zum Ziel den Forschungsstand zum Thema künstliche Intelligenz zusammenzufassen und neue Erkenntnisse in Hinblick auf den Aspekte der Bewegung zu liefern. In dieser Arbeit wäre die Methode A* - Algorithmus, der schwer zu programmieren war, maßgeblich für die Untersuchung von Themenbereich der intelligente Bewegung. Während dieser Arbeit wird der Roboterbau und die Programmierung, die den größten Teil der Arbeit hatten, behandelt.

II. Herausforderungen

Die Idee war zunächst, einen Roboter zu bauen, der durch sein Labyrinth laufen konnte. Durch Bewegungssensoren, die es dem Roboter ermöglichen, den richtigen Weg zu kennen. Dort gab es jedoch ein Problem, bei dem der Roboter den Endpunkt kennt.
Zu diesem Zeitpunkt wurde die Idee des Roboters geändert. Und die Idee wurde zu einem intelligenten Roboter, der sich innerhalb eines bestimm- ten Ortes von einem Startpunkt zu einem Endpunkt bewegen und die Hindernisse kennen und überwinden kann.

Aufgrund dieses Algorithmus, der viel Zeit in Anspruch nahm und nicht richtig programmiert wurde. kann die Idee nicht angewendet werden. wegen der kurzen Restzeit. wurde die Entscheidung getroffen, den Roboter auf einfache Weise zu programmieren, um die gewünschte Idee zu kommunizieren.

III. Hauptteil

Der Bau eines Roboters war der einfachste Teil des Projekts. Zuerst wurde eine quadratische Basis gebaut. Dann wurden vier Räder an beiden Seiten der Basis angebracht. Zwei Motoren sind an der Basis angebracht und an den Vorderrädern befestigt. Um die Bewegung des Roboters zu erleichtern, sind die Vorderräder über Zahnräder an den Hinterrädern befestigt. Dann wurde der UltraSonic Sensor vor dem Roboter platziert. Damit er die vor ihm liegenden Hindernisse erkennen kann.
Der Programmierteil war am schwierigsten. Nachdem den Algorithmus nicht geschrieben werden kann. Ein 7×7 mesh grid wurde erstellt. Da allen Seiten die Nummer -1 zugewiesen ist. Dies sind die Grenzen der Region, innerhalb derer sich der Roboter bewegen kann.
Dann wurde die Entfernung, die der Roboter in einem Zyklus zurücklegte, durch diese Gleichung berechnet.

$$\text{Circumference of Wheel} = 3.14 \times \text{Diameter} \quad (1)$$

$$\text{Distance} = \text{Circumference} \times \text{Rotations} \quad (2)$$

Jetzt kann Sie die Länge der quadratischen Seite und die Gesamtfläche berechnet werden, innerhalb derer sich der Roboter bewegen kann. Der Roboter ist so programmiert, dass er sich in drei Richtungen bewegt. Der Endpunkt sollte entweder 0 °, 45 ° oder 90 ° vom Startpunkt entfernt sein.

Hier ist der Code für die Karte.

```
Map(1,:) = -1;
Map(7,:) = -1;
Map(:,7) = -1;
Map(:,1) = -1;
Start = [2,2];
Goal  = [x,y];
Map(2,2) = 1;
Map(x,y) = 1;
disp(Map)
disp(Start)
cd = Goal - Start;
disp(cd)
path = sqrt((cd(1))^2 + (cd(2))^2);
disp('Path = ')
disp(path)
```

Danach wurde die Karte mit der Bewegung des Roboters verknüpft. Der UltraSonic Sensor liest

DOI: 10.24352/UB.OVGU-2020-047

```
- 1.  - 1.  - 1.  - 1.  - 1.  - 1.  - 1.
- 1.    1.    0.    0.    0.    0.  - 1.
- 1.    0.    0.    0.    0.    0.  - 1.
- 1.    0.    0.    0.    0.    0.  - 1.
- 1.    0.    0.    0.    0.    0.  - 1.
- 1.    0.    0.    0.    0.    0.  - 1.
- 1.  - 1.  - 1.  - 1.  - 1.  - 1.  - 1.
```

Abbildung 1. Beispiel für die Karte

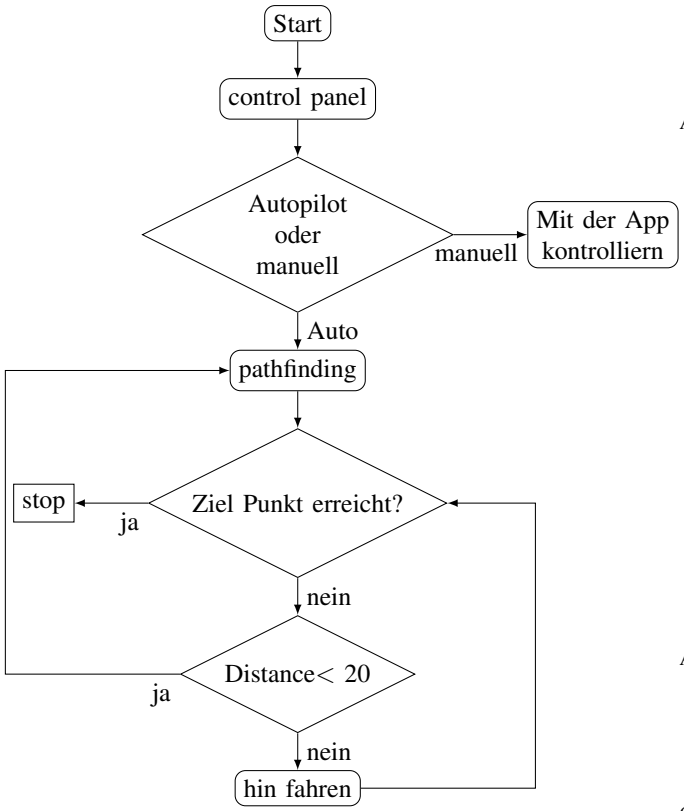


Abbildung 2. Programmablaufplan (PAP)

immer den Abstand vor dem Roboter. Für den Fall, dass der Abstand weniger als 20 cm betrug. Dies bedeutet, dass ein Hindernis vor ihm liegt. Er wird sich umdrehen. Dann bewegt es sich wieder in Richtung Endpunkt. Wenn der Roboter den Endpunkt erreicht, stoppt er.

Zusätzlich wurde ein User interface erstellt, in der der Benutzer den Endpunkt festlegen kann.Die manuelle Steuerung des Roboters wurde ebenfalls durchgeführt. Der Benutzer kann die Motorleistung steuern, nach links oder rechts abbiegen, nach vorne fahren oder stoppen. Nur zur Sicherheit, bei manueller Steuerung, liest der UltraSonic Sensor den Abstand vor dem Roboter. Und wenn es weniger als 20 cm ist. Der Roboter stoppt

Abbildung 3. Erkundungsroboter

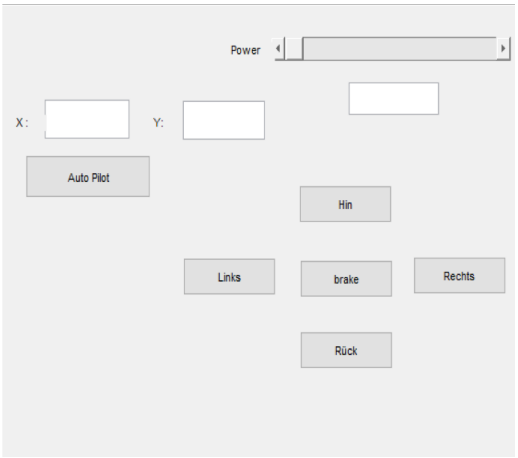


Abbildung 4. Das User Interface

IV. ZUKÜNFTIGE ENTWICKLUNGEN

Am Ende kann sich jedes Projekt entwickeln und eine Zukunft haben. Hier werden die möglichen Ergänzungen geschrieben, bis dieser Roboter entwickelt ist und wo er verwendet werden kann. Wenn der richtige Algorithmus verwendet wird. Einige Sensoren können hinzugefügt werden.

- Ein Intelligenter Prozessor.
- Ein Kamera, damit er der Umwelt um ihn herum erkennen kann.
- Ein Tonsensor zum Hören von Sprachbefehlen.

Mit alle diesen Ergänzungen können wir einen persönlichen Assistenten entwickeln, der wie ein Haustier aussieht und sich im Haus bewegen und alle Smart-Home-Geräte steuern kann.

LITERATURVERZEICHNIS

[1] James Floyd Kelly. *LEGO MINDSTORMS NXT-G Programming Guide*. Apress, 2010. https://books.google.de/books?id=IAU4QHrP1WsC

Der Erkundungsroboter

Ben Duwanoff, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Abstract*—Dieses Paper enthält das Ergebnis des Projektseminars Elektrotechnik/Informationstechnik (LEGO Mindstorms), dessen Ziel es ist einen Erkundungsroboter zu bauen. Dabei wurde ein NXT-Set der LEGO Mindstorms Serie und MATLAB als Programmiergrundlage verwendet. Zudem wurden für die Programmierung Bug-Algorithmen und der A*-Algorithmus erforscht sowie eine eigene Lösung eines Zielfindungsalgorithmus aufgestellt und ein Roboter mit diesen grundlegenden Funktionen konstruiert sowie sein autonomes Fahrverhalten im Zusammenhang getestet und ausgewertet.**

***Schlagwörter*—autonomes Fahren, Erkundung, Fahrzeug, LEGO Mindstorms NTX, Suchalgorithmus**

I. Vision und Antrieb

Die Erkundung der Welt ist ein urmenschliches Bedürfnis. Jedoch kommt der Mensch immer wieder an seine physikalischen Grenzen, die ersehnten Orte zu erreichen, seien es die Tiefen der Meere, die höchsten Berge oder fremde Planeten. Viele dieser Orte konnte sich der Mensch mit Hilfe von Technik zugänglich machen. Der Erkundungsroboter soll dabei das Problem lösen, dass eine direkte Steuerung nicht immer möglich oder gewollt ist, sowie die Aufzeichnung und Übertragung der Messdaten gewährleisten. Das Projekt beschäftigt sich zunächst mit der Bewegung, Steuerung und Orientierung eines solchen Fahrzeugs. Es soll entweder über eine Benutzeroberfläche ferngesteuert oder über eine Zielkoordinate eigenständig auf den Weg geschickt werden und dabei eine Kollision mit Hindernissen vermeiden.

II. Grundlagen und Nachforschungen

A. LEGO Mindstorms NTX Serie

In diesem Projekt wurde ein NTX-Baustein der LEGO Mindstorms Serie verwendet. Im NTX-Grundset sind ein NXT-Stein, drei Elektromotoren mit Rotationssensor, zwei Tastsensoren, einen Ultraschallsensor, ein Farbsensor bzw. Lichtsensor, ein Schallsensor, sieben RJ12 Kabelverbindungen, ein USB-Kabel sowie einige LEGO-Technic-Elemente, Bauanleitungen, eine Testfläche und NXT-G als integrierte Entwicklerumgebung enthalten. Diese NTX-Steuerungseinheit besitzt Anschlüsse für vier Sensoren und drei Elektromotoren. Angesteuert wird der NXT-Stein über ein USB A zu USB B Kabel, mit dessen Hilfe man entweder den Roboter von einem Computer aus befehlen oder ein Programm auf den Roboter übertragen kann, welches dann mit Hilfe der Tasten und der Menüführung auf dem NTX-Stein direkt ausgeführt wird. Die Sensoren und Motoren werden durch RJ12-Kabelverbindung mit den NTX-Stein an den entsprechenden Anschlüssen verbunden [1]. Um das Programm zu realisieren, wurde MATLAB als Programmiersprache sowie als Compiler des Programms verwendet. Da MATLAB selbst jedoch keine Schnittstelle zur Steuerung von LEGO-Mindstorms-NTX-Bausteinen besitzt, diente die MATLAB-Bibliothek, die die RWTH Aachen bereits entwickelt hat und diese freundlicher Weise auf ihrer Seite [2] frei zur Verfügung stellt, als Solche.

B. Pfadsuchalgorithmen

Da der Roboter einen Weg zum Ziel finden soll, standen zur Wegfindung zunächst ein Bug-Algorithmus oder der A*-Algorithmus zur Auswahl. Der Bug-Algorithmus benötigt im Gegensatz zum A*-Algorithmus neben seinem Start und Ziel keine globalen Kenntnisse, da er sich an einem einfachen Verhalten orientiert. Er folgt einer geraden Linie zum Ziel und falls er auf ein Hindernis trifft, fährt er diesem nach rechts oder nach links entlang, um dann hinter dem Hindernis in einer geraden Linie Richtung Ziel weiter zu kommen. Es gibt zwei Varianten diesen zu realisieren, den Bug 1 und den Bug 2 Algorithmus. Bei dem Bug 1 Algorithmus fährt der Roboter einmal um das gesamte Hindernis herum und merkt sich zu jeder Position auf dem Weg den Abstand und die Koordinaten bis er am ersten Kontaktpunkt mit dem Hindernis wieder angekommen ist. Dann findet er den Punkt mit dem kürzesten Abstand zum Ziel und fährt noch einmal am Hindernis bis zu diesem Punkt entlang. Danach bleibt er wieder auf der Geraden aufs Ziel zu, bis er ankommt oder ein weiteres Hindernis erkennt. Der Bug2-Algorithmus verkürzt die Umfahrung des Hindernisses, da dieser nicht das ganze Objekt umrundet, sondern nur solange am Hindernis entlang fährt, bis er wieder auf die Gerade trifft, die am Anfang zum Ziel berechnet wurde. Dieser schlägt zeitlich dadurch in fast allen Fällen den Bug1-Algorithmus. Die Bug-Algorithmen benötigen sehr genaue Sensoren, da es durch Fehlermessungen zu Kollisionen oder Irrfahrten des Roboters kommt [3]. Der A*-Algorithmus benötigt im Gegensatz dazu alle globalen Informationen seines Zielgebietes, wodurch er auch ohne Sensoren den kürzesten Weg zum Ziel finden würde. Jedoch sollten zur Sicherheit einige Sensoren angebracht werden, damit der Roboter bei einer Fehlfunktion zum Start zurückkehren kann. Bei dem A*-Algorithmus berechnet man zunächst die Abstände aller möglichen Positionen im Zielgebiet zum Ziel. Danach markiert man die Positionen der Hindernisse und eröffnet mindestens zwei Listen. Die sogenannte offene Liste enthält alle noch nicht abgearbeiteten Positionen. Sie

DOI: 10.24352/UB.OVGU-2020-048

beinhaltet am Anfang nur den Startpunkt oder ist je nach spezieller Struktur leer und wird danach vom Algorithmus mit den zu bearbeitenden Positionen befüllt. Die zweite Liste ist die sogenannte geschlossene Liste. In diese werden alle Hindernisse mit ihrem Status als solche eingetragen. Man kann theoretisch auch eine eigene Liste für Hindernisse anlegen.

Nun beginnt der Algorithmus in einer Schleife mit der Position, welche in der offenen Liste steht und die geringsten Gesamtkosten zum Ziel hat. Diese setzen sich zusammen aus den Wegkosten, welche die kürzeste Strecke vom Start zu der entsprechenden Position darstellen und Restkosten, welche der direkte Abstand von der Position zum Ziel festlegt. Am Anfang beginnt der Algorithmus deshalb mit der Startposition. Danach berechnet er alle Kosten der um den fokussierten Punkt liegenden Positionen. Wobei sich die Wegkosten eines Nachbarn aus den Wegkosten des fokussierten Punktes und der Strecke zwischen diesen beiden Punkten errechnet. Zusätzlich werden die Koordinaten des fokussierten Punktes den Daten des Nachbarn angehangen, diese werden auch als Eltern bezeichnet. Danach wird geprüft, ob einer der Nachbarn das Ziel oder ein Hindernis darstellt oder bereits in einer der Listen vorhanden ist. Wenn dies nicht der Fall ist, werden die Datensätze der betroffenen Positionen der offenen Liste hinzugefügt. Falls der Nachbar bereits vorhanden ist, wird geprüft, ob die neu berechneten Wegkosten kleiner als die bereits vorhandenen sind. Sollte das der Fall sein, werden die Datensätze des Nachbarn aus allen Listen entfernt und in der offenen Liste wird der neue Datensatz dieser Position eingetragen. Ist der Nachbar ein Hindernis wird er ignoriert und sollte er das Ziel sein, wird sein Datensatz zusammen mit dem der fokussierten Position in die geschlossene Liste verschoben und aus den anderen Listen entfernt. Danach wird die Schleife abgebrochen. Sonst geht es weiter, bis alle Nachbarn abgearbeitet sind. Nun wird der Datensatz, welcher am Anfang noch der Start ist, in die geschlossene Liste verschoben und aus der offenen Liste entfernt. Danach beginnt die Schleife wieder von vorn mit der Suche eines neuen fokussierten Punktes aus der offenen Liste.

Sollte die Schleife durch das Ziel abgebrochen werden, kann man über die Koordinaten der Eltern von Punkt zu Punkt bis zum Start zurückfahren und erhält so die kürzeste Strecke zwischen Start und Ziel. Die Datensätze der Positionen der kürzesten Strecke kann man dann noch in einer eigenen Liste speichern, um den kürzesten Weg nicht zweimal berechnen zu müssen [4],[5].

III. Realisierung des Erkundungsroborters

A. *Konstuktion und technischer Aufbau*

Der Erkundungsroboter besteht aus einem LEGO Mindstorms NTX-Stein, welcher oben am Konstrukt angebracht ist. Darunter befinden sich zwei Elektromotoren aus dem LEGO-Mindstorms-Set. Jeder Motor steuert dabei die Vorderräder der jeweiligen Seite an. Die Hinterräder sind auch Antriebsräder, werden jedoch über ein Getriebe zum jeweiligen Vorderrad betrieben. Dadurch benötigt der Roboter keine Lenkung, sondern wird durch die Geschwindigkeit der beiden Motoren gesteuert. Durch die eingebauten Rotationssensoren kann so die Fahrstrecke oder der Drehwinkel für das Fahren bestimmt werden.

Um die Fahrweite für die Programmierung zu kalibrieren, wurde der Roboter testweise mit verschiedenen Motordrehweiten gefahren bis ein Drehwinkel von 360 Grad gefunden wurde, damit der Roboter 20 cm nach vorn fährt. Um vorwärts oder rückwärts zu fahren, müssen sich hier beide Motoren in die gleiche Richtung bewegen. Um im Programm leichter die Position des Roboters bestimmen zu können, kann sich er nur auf der Stelle drehen, jedoch keine Kurven fahren. Deshalb drehen sich die Motoren beim Ausrichten des Roboters gleich schnell, jedoch in unterschiedliche Richtungen. Dabei wurde auch hier die Drehweite der Motoren für die einzelnen Drehungen über Testläufe kalibriert, sodass sich zum Ausrichten nach 90 Grad der vorwärts drehende Motor um 380 Grad und der rückwärts drehende Motor um 100 Grad drehen musste. Da die Motoren sich proportional zur Ausrichtung bewegen müssen, lassen sich die Drehweiten der Motoren für andere Winkel durch das Verhältnis der Winkel zum Kalibrierungswinkel als

Abbildung 1: Konstruktion des Roboters

Vorfaktor bestimmen.

Zudem besitzt der Roboter noch einen nach vorn zeigenden Ultraschallsensor, um Hindernisse auf dem Weg entdecken zu können. Bei der Abfrage des Sensorwertes gibt der Ultraschallsensor automatisch den Abstand des davor liegenden Objektes in Zentimetern aus. Da der Sensor einen 8 Bit Wert zurückgibt, können die Abstände dabei nur bis zu 255 cm betragen. Sollte 256 cm ausgegeben werden, übersteigt der Abstand den Messbereich. Wenn man von vorn auf den Sensor schaut, befinden sich dort zwei Öffnungen. In der rechten Öffnung befindet sich der Ultraschallgenerator und in der Linken der Ultraschallsensor. Dadurch kann es bei zu kleinem Abstand zu Fehlermessungen kommen, weshalb Sensor nur für Werte über 6 cm verwendet werden kann.

B. *Steuerung und Programmierung*

Die Robotersteuerung besteht aus zwei Programmabschnitten: der manuellen Steuerung und der automatischen Zielfindung. Beide werden durch die gleiche grafische Benutzeroberfläche gesteuert. Bei der manuellen Steuerung besitzt der Roboter auf der Benutzeroberfläche einen Geschwindigkeitsregler und vier Bewegungsregler, welche aus einem Vorwärtsknopf, einem Rückwärtsknopf, jeweils einem Knopf, zur Rechts- und Linksdrehung um 45 Grad bestehen, sowie einen Bremsknopf, um den Roboter anzuhalten.

Wenn man den Vorwärts- oder Rückwärtsknopf drückt, fährt der Roboter solange in die entsprechende Richtung, wie man

den Knopf gerückt hält. Zudem hat der manuelle Modus einen Auffahrschutz einprogrammiert, der automatisch die Vorwärtsbewegung abbricht, sollte der Roboter einem Objekt zu nahe kommen. Dies wird durch eine Bewegungsabbruchsbedingung realisiert, die bei einem zu geringen Sensorwert des Ultraschallsensors initialisiert wird. Dadurch werden alle Motoren des Roboters angehalten. Danach kann die manuelle Kontrolle mit der grafischen Benutzeroberfläche den Roboter wieder übernehmen.

Für das automatische Finden des Ziels wurde eine Anlehnung an den Bug-Algorithmus verwendet, der aufgrund des zeitlichen Aufwandes stark vereinfacht wurde. So erkennt der Roboter Hindernisse, fährt jedoch an diesen nicht entlang, sondern initialisiert einen vorgegeben Ausweichpfad. Deshalb muss das Hindernis standardisiert sein und die Strecke der Umgehung für jede Art von Hindernis angepasst werden. Bei der automatischen Zielfindung gibt man dem Roboter die Zielkoordinaten, wobei diese immer in Bezug auf den Roboter übergeben werden müssen, da für ihn selbst keine Koordinaten übergeben werden. Für diesen Zweck wurden auf der Benutzeroberfläche zwei Eingabefelder bereitgestellt, welche einmal die x- und die y-Koordinate des Ziels verlangen. Andere Koordinatenformen wie Polarkoordinaten sind zudem nicht möglich. Wenn man nun auf den „Automatisch“-Knopf unterhalb der Eingabefelder drückt, entscheidet sich der Roboter zunächst für einen der drei folgenden Fälle:

1. Das Ziel befindet sich ausschließlich in x-Richtung oder in ausschließlich y-Richtung oder das Ziel befindet sich zwischen der x- und der y-Richtung. Sollte sich das Ziel nur in x-Richtung befinden, fährt der Roboter gerade aus auf das Ziel zu. Alle 20 Zentimeter erhöht er dann seine eigene x-Position, die am Programmstart null ist, um eins bis sie gleich der Zielkoordinate ist. Dann wird das automatische Programm beendet und der Roboter wartet auf eine Eingabe aus der grafischen Benutzeroberfläche. Sollte sich ein Hindernis auf dem Weg befinden, so erkennt der Roboter dieses mit Hilfe des Ultraschallsensors. Dabei misst der Sensor vor und während der Bewegung den Abstand vor sich. Sollte nun der Abstand zu gering werden, also unter 20 cm sein, bedeutet das, dass der Roboter den nächsten Streckenabschnitt nicht ohne Kollision mit dem Hindernis fahren kann.

   Deshalb stoppen zunächst alle Motoren, wodurch der Roboter anhält, um dann einen Ausweichpfad zu initialisieren, mit dem das Hindernis umfahren wird. Zuerst dreht sich der Roboter dazu um 90 Grad nach links, fährt dann 20 cm nach vorn und dreht sich um 90 Grad nach rechts. Danach fährt der Roboter 40 cm nach vorn, um das Hindernis mit Sicherheit umfahren zu haben und dreht sich erneut um 90 Grad nach rechts. Nun fährt der Roboter wieder 20 cm vor und dreht sich um 90 Grad nach links. Anschließend wird noch die Position des Roboters korrigiert, die durch den Ausweichpfad verursacht wurde und der ursprüngliche Algorithmus übernimmt wieder die Steuerung.
2. Bei der Zielführung in y-Richtung richtet sich der Roboter aus und fährt mit einem ähnlichen Algorithmus wie dem für die x-Richtung zum Ziel. Dabei wird der y-Wert statt dem x-Wert alle 20 cm Fahrstrecke erhöht. Sollte hier der Roboter auf ein Hindernis stoßen, so wird der Ausweichpfad gespiegelt zu dem bei der x-Richtung ausgeführt, damit der Roboter nicht aus dem Kartenbereich herausfährt. Er dreht sich also statt nach links nach rechts und umgekehrt.
3. Falls beide Werte der Zielkoordinate ungleich null sind, errechnet der Roboter zunächst den Winkel in dem das Ziel zu ihm liegt und dreht sich dann auf der Stelle Richtung Ziel, um anschließend dorthin zu fahren. Dabei wird immer alle 20 cm Fahrstrecke der x- und y-Werte des Roboters um Eins erhöht. Wenn er sich am Ziel befindet, hält er an und beendet den Algorithmus. Sollte der Roboter auf dem Weg dorthin mit dem Ultraschallsensor ein Hindernis entdecken, so initialisiert er einen Ausweich-pfad, welcher den Roboter zunächst um 45 Grad nach links drehen und dann um 20 cm nach vorn fahren lässt. Danach fährt der Roboter weiter und erhöht seine x- und y-Position alle 20 cm. Der x-Wert der Zielposition wurde um Eins verringert, damit der Roboter nicht am Ziel vorbeifährt. Jedoch endete der Roboter bei den Versuchsläufen hinter dem Ziel, daher dreht er sich nun als Abschlusssequenz um 135 Grad und fährt noch einmal um 20 cm nach vorne, hält an und beendet den Algorithmus.

Nun kann man das gesamte Programm beenden oder dem Roboter neue Koordinaten übergeben oder ihn manuell zur Startposition zurückfahren. Diese Programmstruktur kann auch durch den Programmablaufplan in Abb. 2 im Anhang, nachvollzogen werden.

IV. Endresultate

Nach einigen Tests zeigt sich, dass der Roboter unter Laborbedingungen sein Ziel findet und das Programm danach wieder auf eine Eingabe aus der grafischen Benutzeroberfläche wartet. Dabei kam es jedoch häufig zur Abweichung zwischen der realen und der im Programm ausgeführten Bewegung. Bei Fehlfunktionen der Getriebe, die manchmal blockierten, wenn man sie nicht vor jeder Durchführung kontrollierte oder weil der Untergrund dem Roboter unterschiedliche Haftungen auf dem abgefahrenen Weg verlieh, wodurch die Drehungen und Fahrweiten ungenau wurden, kam es zu Zielverfehlungen bis zu einer ganzen Einheitslänge von 20 cm. Zudem kann die Zielabbruchsbedingung umgangen werden, wenn sich genau auf dem Ziel ein Hindernis befindet, weil so ein Ausweichen eingeleitet wird, welches den Roboter hinter dem Ziel herauskommen lässt. Da er nun nach seinem Algorithmus in der ursprünglichen Zielrichtung weiterfährt, kann er das Ziel sowohl im Programm als auch im Realen nie erreichen, wodurch auch der Algorithmus zum automatischen Fahren nie sein Ende findet.

V. Zusammenfassung und Fazit

In diesem Projekt sollte ein Roboter konstruiert werden, dem ein Ziel übergeben wird, das er dann eigenständig anfährt, ohne dabei mit Hindernissen zu kollidieren oder dass er über eine einfache Steuerung manuell bedient werden kann. Das Grundziel wurde dabei zwar erreicht, jedoch nur bei sehr einfachen und vorbestimmten Hindernisstrukturen. Um den Roboter zu optimieren, kann man einen Bug2-Algorithmus anstelle des Ausweichpfades setzen, um auch individuelle Hindernisse um-

fahren zu können und die Positionen der Hindernisse speichern, um sie späterer Verarbeitung zugänglich zu machen.

ANHANG

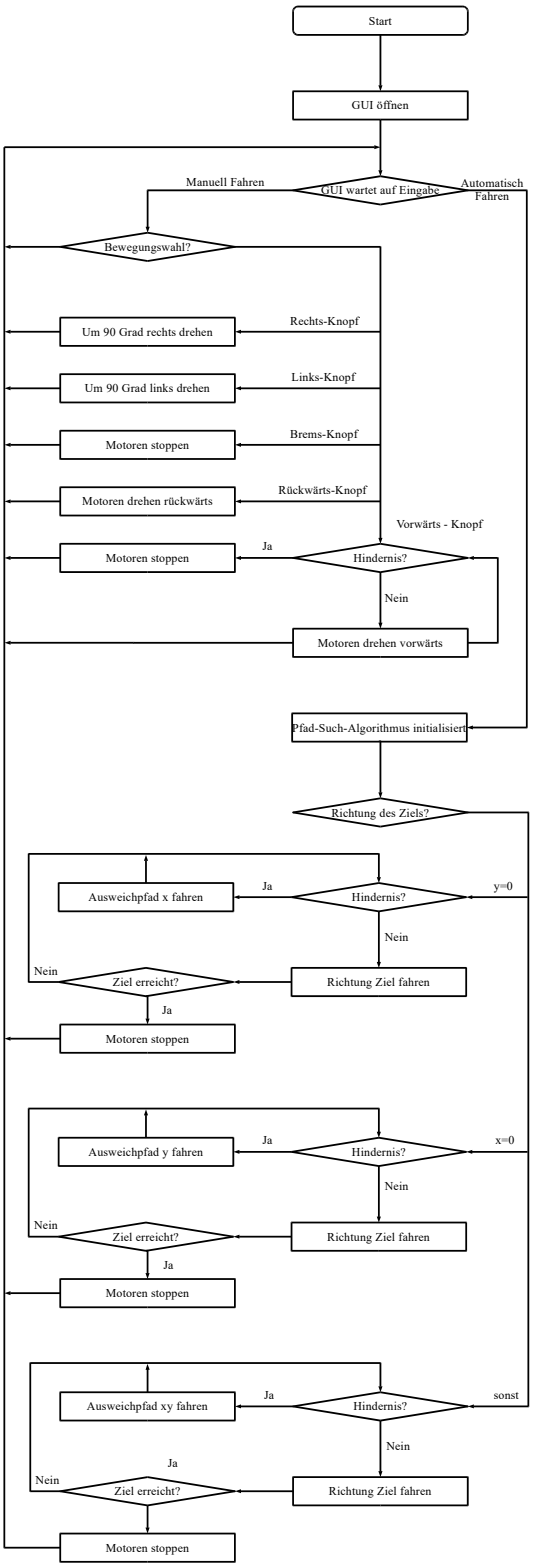


Abbildung 2: Programmablaufplan des Erkundungsroboters

LITERATURVERZEICHNIS

[1] WIKIPEDIA, DIE FREIE ENZYKLOPEDIE: *Lego Mindstorms NTX*. https://de.wikipedia.org/wiki/LEGO_Mindstorms_NXT, Version: 03.03.2020

[2] RWTH AACHEN: *RWTH - Mindstorms Toolbox*. https://www.mindstorms.rwth-aachen.de/trac/wiki/Download4.07, Version: 04.03.2020

[3] HOWIE CHOSET, CARNEGIE MELLON SCHOOL OF COMPUTER SCIENCE*: Robotic Motion Planning Bug Algorithms*. http://www.cs.cmu.edu/~motionplanning/lecture/Chap2-Bug-Alg_howie.pdf, Version: 03.03.2020

[4] WIKIPEDIA, DIE FREIE ENZYKLOPEDIE: *A*-Algorithmus*. https://de.wikipedia.org/wiki/A*-Algorithmus, Version: 05.03.2020

[5] SEBASTIAN LAGUE, YOUTUBE: *A* Pathfinding (E01: algorithm explanation)*. https://www.youtube.com/watch?v=-L-WgKMFuhE, Version: 05.03.2020

Bauversuch eines Erkundungsroboter

Mahmoud Jahjah, ETIT
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Wie seit 2013 fand es auch dieses Jahr ein Lego Mindstorms Projektseminar an der Otto von Guericke Universität mit vielen praktischen und erfindereich Projekten. In dieses Paper geht es um einen Versuch, der sich mit der Entwicklung eines Roboters befasst. Der Roboter soll am Ende des Versuchs Weg von Anfagnspunkt zum Endpunkt durch Hindernisse erkunden.**

***Schlagwörter*—Wegsuch Algorithmus, A-Stern Algorithmus, Bug O Algorithmus, Roboter.**

I. Einleitung

MIT der Entwicklung der Zivilisation begann der Mensch zu suchen, was sein Leben in jeder Hinsicht einfacher machte. Um die Entfernung zu erleichtern, hat er in letzten 2 Jahrhunderten von der Dampfmaschinen bis des ersten Verbrennungsmotors und dessen Entwicklungen bis zu den Elektromotoren erfunden. Aber er war damit nicht zufrieden, sondern arbeitete er nun auch an Autos, um sich selbst zu führen. Heutige Fahrzeugtechnik interessiert sich dafür, dass die Maschine sich mit guten Verständis der Umgebung im Raum bewegt. Obwohl diese Technologie Ihr endgüldige Form noch nicht erreicht hat, bietet sie vielfältige Produkte: von Haushaltsprodukten wie elektronischen Staubsaugern bis zu selbst parkende Autos. In dem Projekt , das LEGO Mindstorms Seminars im Zeitraum von 2 Wochen mit Herr Altarabeen und Herr Duwanoff bearbeitet wurde, wird ein Versuch durchgeführt, dass ein Roboter sich selbst von Startpunkt zu Endpunkt führt und die Hindernisse umgeht.

II. Stand der Technik

Der Roboter wird mit LEGO MindStorm NXT angeschaltet. Das NXT-Set ist doch besonderes, weil es NXT -Stein , Servmotoren und verschiedene Sensoren besitzt. Der NXT-Stein hat einen 32Bits-Mikroprozessor, Realsound-Lautsprecher und Anschlüsse für die Motoren und Sensoren, und lässt sich in MATLAB programmieren; Die Arten von Sensoren sind Ultraschallsensor, Lichtsensor, Schallsensor, und Farbsensor. Für den Roboter wird nur Ultraschallsensor benutzt, der den Abstand misst. Außerdem war die Aufbau des Robotes simpel, da die Programmierung der größte Teil des Projektes ist. und für die Bewegunghaben wir uns A-Stern-Wegsuchung-Algorithmus entschieden.

III. Haupttile

A. Aufbau und Program Struktur

Bei die Robotstruktur waren die beide Motoren mit den Vorderräder verbunden, und die hintere Räder durch Rädchen

Abbildung 1. Der Erkundungroboter

mit die Vorderräder gekopplt wurden. Für die Rechterotation des Robots werden die linke Rädchen mit Uhrzeige gedreht und sich die rechte gegen Uhrzeige gedreht, und vice versa, Außerdem wird Graphik-User-Interface(GUI) als die Control Panel des Robots gezeichnet. Mit zwei Optionen, entweder Manuel oder Autopilot. Im Autopilot wird die Koordinaten der Zielpunkt eingegeben. Außerdem die Leistung der Motoren wird auch vom User ausgewählt. Darüber hinaus wird ein Manuelkontroll addiert, dass der Roboter sich hin- und herbewegt und sich Links und Rechts orientiert.

B. Algorithmus

Warum haben wir A-Stern-Algorithmus festgestellt und wie funktioniert er? Was ist in dem besonderes? Er ermittelt den kostengünstigsten Weg zwischen einem Anfang- und einem Zielknoten innerhalb eines Suchbaums. Es könnte auch sein, dass der Suchbaum ein matrix-förmiger Suchbaum ist, weil bei der quadratischen Darstellung des Bewegungsfelds die Genauigkeit der Bewegung des Robotes in diesem Fall verbessert wird. Darüber hinaus ist er ein informiertes und erschöpfendes Verfahren, aber nur im Suchprobleme praktisch, im unseren Situation ist er sehr anwendbar. Bei Vergrößern der Knoten innerhalb des Suchbaums werden die Kosten aller neuen Knoten durch die Funktion :

$$K = K_b + H$$

DOI:10.24352/UB.OVGU-2020-049

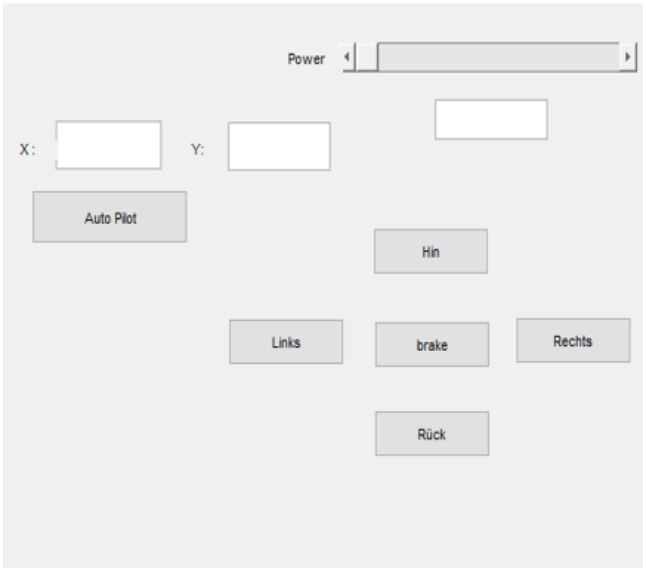


Abbildung 2. Graphic USer Interface (Control Panel)

Ermittelt, wo K Kosten des Knoten sind; K_b der Entfernung von dem Anfangspunkt ist; und H die Entfernung von dem Zielpunkt ist. Außerdem gibt es 2 Liste, eine heißt $OpenList$ und die andere $CloseList$. Im $OpenList$ werden die Kosten aller Knoten miteinander vergleicht. Der Punkt mit kleinsten Kosten wird im $CloseList$ beseitigt. für bessere Erklärung wird eine Demonstration [1]

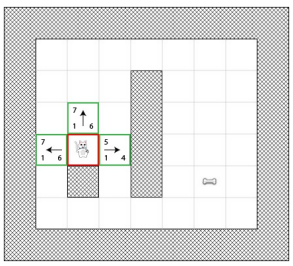


Abbildung 3. Anfangsbild

In diesem Beispiel wird das Rote Quadrat als Anfang und gegenwärtige Position der Katze bezeichnet. Außerdem wird der Knochen als Endpunkt gezeichnet. (sehe Figure 1) Jede grüne Quadrat stellt eine mögliche Bewegung der Katze dar. Zwei davon haben 7 als knoten Kosten K. Außerdem ist jede H werte rechts auf der unteren Ecke. Der Algorithmus geht dann zu die Kleinste K. in diesem Fall Rechts, und wird diese Quadrat als S genannt

Nachdem die Katze sich bewegt hat, wird S im Close Liste entfernt, und nochmal werden aller Kosten für die Punkte berechnet, aber dieses Mal haben wir ein Hindernis, d.h das die Katze kann nicht dadurch gehen. Hindernisse werden im Close Liste entfernt.(sehe Abbildung 3)

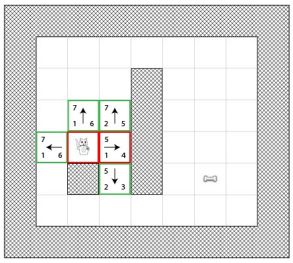


Abbildung 4. Die Katze hat sich bewegt und ein hindernisse sieht

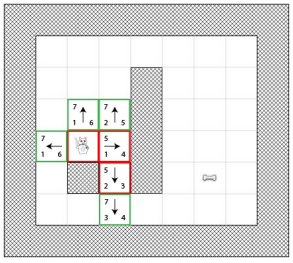


Abbildung 5. Sie hat die Hindernisse bemerkt und sich rechts bewegt

Nach der Entfernung des zweiten Punkts gibt es mehr als zwei Knoten, die gleiche K Werte haben. In diesem Fall wird die H Werte vergleicht. d.h das die Punkt mit Kleine H Werte wird bevorzugt.

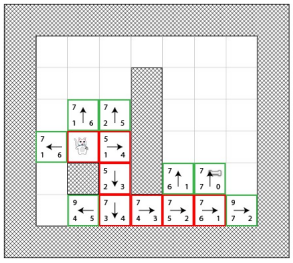


Abbildung 6. Die Katze hat das Ziel erreicht

Nach der Wiederholung des Vergleichens zwischen Knoten erreichtdie Katze ihr Zielpunkt (sehe Abbildung 6). Der Algorithmus ist, wie vorher gesagt wurde, ist nur im Such-probleme, aber in dem Projekt wird die Nutzung von A* mit Robotsbewegung geprüft. Der Algorithmus wird im Auto Pilot durchgeführt und mit manuelle Kontrolle im GUI angewendet.

IV. Ergebnisdiskussion

A. *Herausforderungen*

Die algorithmus endgültige Form ist mit der Frist nicht erreicht geworden, deshalb wird der Algorithmus verändert.

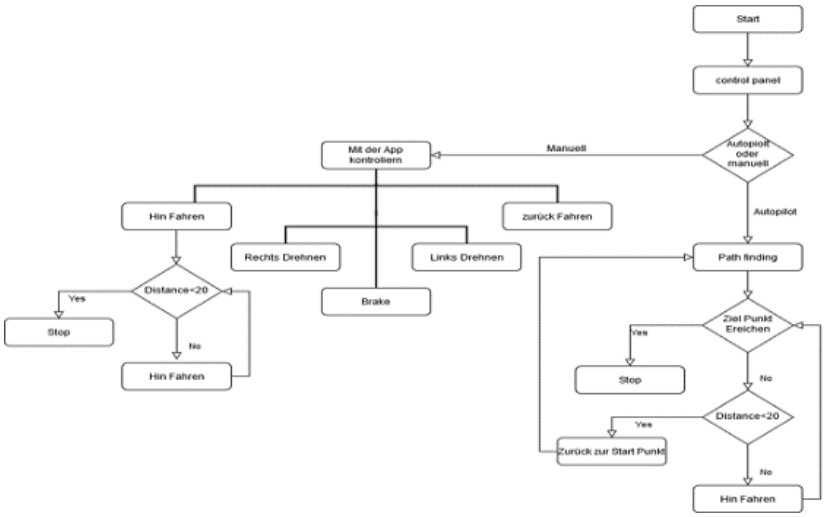


Abbildung 7. Abluafplan

und wird ein anderer Algorithmus ausgewählt, der Bug O heißt. [2] Sein Hauptkonzept ist den direkten Pfaden zum Ziel, und wenn ein Hindernis siehst, folgt er es.Auf der anderen Seite treten Aufgrund dieser Veränderung des Plans viele Behinderungen auf. z.B der Roboter die Hindernisse bis unendlich folgt. Außerdem wird die Genauigkeit des Roboterpositions mangelhaft, da die Abstand zwischen Anfang- und Endpunkt nicht punktlich ist. Beim Sturktur war die Ablenkung nicht immer getroffen, da die Rädchen nicht gut fest gehalten war.

B. Überwendungen

Die Verbindung von A* und Bug O Konzepten war die Lösung. Die unendliche Folgerung der Hindernisse und Positionsgenauigkeit kann durch die Anwendung der Matrix von A* übergenommen werden, da es dem Roboter bessere Einsicht gibt. Die Anwendung des Bug O Konzepts ermöglichst, dass der Robot sich gerade und 45Grad bewegt.

V. Zusammenfassung

Wegen der Zeit war der Roboter nicht fertig, deshalb haben wir die Hauptkonzept von A* und anderen Algorithmus, Bug O, zusammen benutzt, damit Das Robot sich zum Ziel bewegt. Warum wird Bug O entschieden? Da er in wenigen Zeit programmiert werden kann.

Literaturverzeichnis

[1] Wenderlich, Ray: *Introduction to A* Pathfinding*. Version 03.00, September 2011. https://www.raywenderlich.com/3016-introduction-to-a-pathfindinghttps://www.raywenderlich.com/3016-introduction-to-a-pathfinding. – PD 5214.5061.61

[2] Black, Paul E.: *Big O notation*. 6. 2006