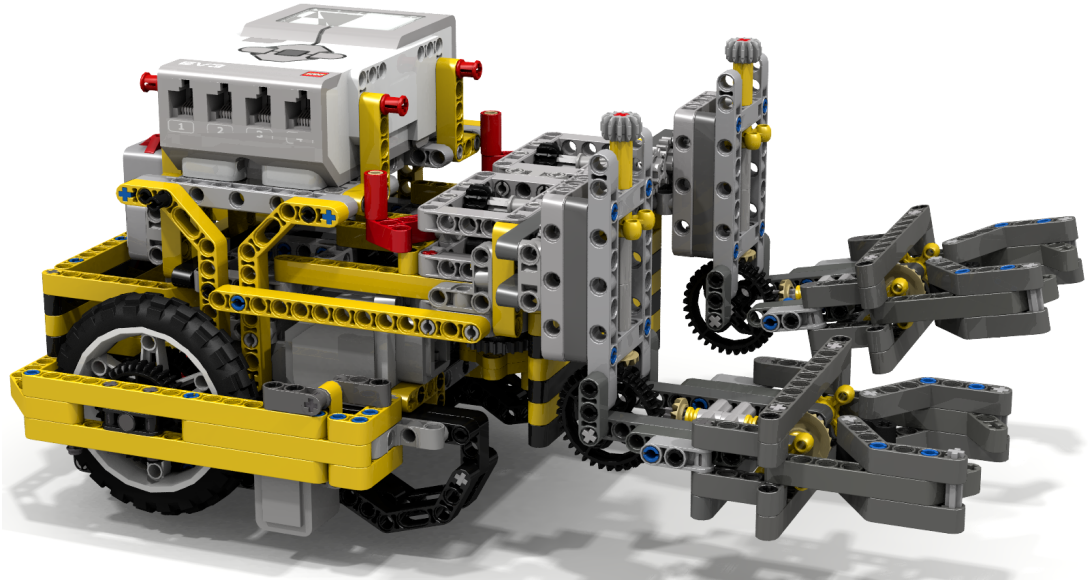
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

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 2 vom Wintersemester 2018/2019

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-2019-035
ISSN:	2629-6160
Redaktionsschluss:	Juli 2019
Seminarzeitraum:	11.–22. Februar 2019
Bezug:	Open Access, Digitale Hochschulbibliothek Sachsen-Anhalt `http://edoc2.bibliothek.uni-halle.de/`

1. Auflage, Magdeburg, Otto-von-Guericke-Universität, 2019

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

Der Farbsortierroboter

Lara Müller, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

Abstract *- Auch in diesem Jahr 2019 wurde das Lego Mindstorms Praktikum von der Otto von Guericke Universität durchgeführt. Seit 2013 konstruieren und programmieren Studenten verschiedene Roboter aus LEGO, die vor einer Jury aus Studenten vorgestellt werden. Wie auch der Farbsortierroboter, mit dem sich in diesem Paper auseinandergesetzt wird. Dabei wurden die Schwerpunkte auf die Planung der Idee und die Umsetzung des Konzeptes gelegt. Der Farbsortierroboter sollte in der Lage sein LEGO-Steine nach ihrer Farbe zu sortieren und in die entsprechenden Behälter zu sortieren. Bei der Umsetzung des Projektes traten einige Probleme beim Programmieren und beim Konstruieren auf. Dennoch ist es den Entwicklern gelungen, am Ende des zwei wöchigen Projektseminars, einen autonom farbsortierenden Roboter zu präsentieren.*

Farbsortierroboter, LEGO, NXT, Otto-von-Guericke Universität

I. Einleitung

Seit Beginn der Industrialisierung ersetzen Maschinen nach und nach die Arbeit der Menschen. In den folgenden Jahren sind Maschinen essenzieller geworden und nicht mehr aus der Wirtschaft und dem privaten Haushalt weg zu denken. Heutzutage ist die Automatisierung von Abläufen präsenter denn je, denn sie arbeiten größtenteils genauer und schneller und sparen Zeit, die der Mensch in andere Tätigkeiten stecken kann. Besonders bei älteren Menschen sind Maschinen hilfreich, da sie gesundheitlich nicht mehr in der Lage sind jede Arbeit so auszuführen wie ein gesunder und junger Mensch. Auch in der Unterhaltungsbranche sind Maschinen nicht mehr wegzudenken, zum Beispiel Smartphone oder Computer. Sogar Aktivitäten ohne Maschinen werden mit moderner Technik verbessert, wie im Fußball der Videoassistent oder die Kameraführung beim Golf. Sie vermitteln Nähe, wo eigentlich größere globale Distanzen auftreten und verbinden Kontinente und Menschen miteinander. Der wichtigste Aspekt der modernen Maschinen ist die Verringerung der Arbeit des Menschen, worauf sich der Farbsortierroboter bezieht. 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. Der Mensch muss nichts weiter tun, als die zu sortierenden Gegenstände in die Maschine zu packen.

II. Vorbetrachtungen

Damit der Farbsortierroboter etwas sortieren kann, benötigt er einen Farbsensor dessen Funktionsweise erst mal genauer betrachtet werden muss und in welcher Reihenfolge der Prozess ablaufen soll.

A. Farbsortierroboter Prinzip

Die Idee dahinter ist ein Roboter der Gegenstände, wie Lego-Steine, nach ihrer Farbe sortiert. Zunächst soll ein Farbsensor die Farben der Gegenstände erkennen. Nach erkennen der Farbe wird der Stein in einen Behälter transportiert, der seiner Farbe entspricht. Der Vorgang wird so oft wiederholt, bis alle Steine einsortiert sind oder der Benutzer das Programm beendet. Das Prinzip ist recht simple „Ohne großen Aufwand Ordnung zu schaffen“. Dies ist heutzutage auch schon in großen Firmen zu finden. Ein Beispiel ist bei der Tomatenernte zu finden. Bei dieser werden die geernteten Tomaten über ein Fließband laufen und nach Verfärbungen, verfaulten Stellen, Sonnenbrand, gebrochene, zu kleine oder zu große Tomaten und alle Arten von Fremdmaterial, wie Tiere, Baumwollstiele, Metall, Kunststoffe, Steine, Glas, Holz, usw. getrennt. Das Verfahren nachdem die Tomaten sortiert werden, achtet dabei nicht nur auf die Farbe, sondern auch auf andere Faktoren die beim Lego Farbsortierroboter nicht umsetzbar wären.

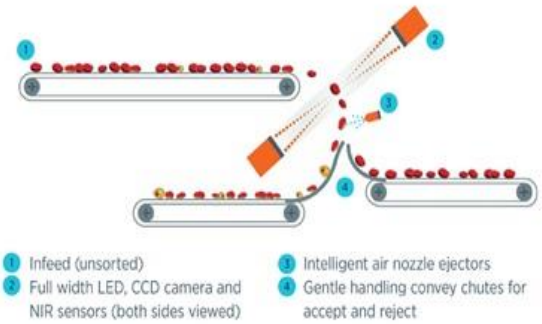


Abbildung 1: Tomatensortierungsverfahren

DOI: 10.24352/UB.OVGU-2019-036

B. Farbsensor

Zum Erkennen der Farben kommt der RGB-Sensor (Farbsensor) von Lego zum Einsatz. Der Sensor strahlt das zu scannende Objekt mit den Farben rot, blau und grün an. Um die Farbe genau zuzuordnen erfasst ein Sensor die reflektierten Farben. Wird mehr grünes Licht zurück geworfen, wird der Gegenstand als grün anerkannt. Das gleiche Prinzip wird ebenfalls bei rot und blau angewendet. Der Farbsensor kann auch noch gelb (rot und grün gleichstark reflektiert, blau keine Reflektion), weiß (alle 3 Farben gleichstark reflektiert) und schwarz (keine Farbe wird reflektiert) erkennen. Weitere Farben, wie Orange, werden bei den 6 genannten Farben (rot, grün, blau, gelb, schwarz, weiß) untergeordnet. Mit dem Wissen das es 6 zu sortierende Farben gibt braucht der Behälter 6 verschiedene Fächer für je eine Farbe.

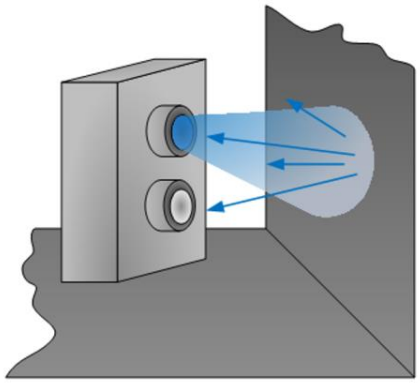


Abbildung 2: Funktionsweise Farbsensor

C. Starten und pausieren des Programms

Das System soll auf Knopfdruck starten und stoppen. Im Falle eines Falls soll ein Notaus-Schalter existieren, um schwerere Probleme zu verhindern. Zur Auswahl der Umsetzung stand ein Start/Stopp Knopf in der GUI (Graphical User Interface) oder ein Schalter aus einem Tastsensor. Diese Funktion ist hilfreich um zu verhindern, dass sich Gegenstände verkeilen und das System unnötig Strom verbraucht.

III. HAUPTTEIL

A. Farberkennung

Begonnen wurde mit der Einrichtung des Farbsensors in MATLAB. Dafür muss man den Sensor ansprechen und einen Modus wählen. Von den mehreren zur Auswahl stehenden Modi des Farbsensors wurde der Modus „FULL“ verwendet. Die Farben werden größtenteils richtig erkannt und in einem Textfeld ausgegeben. Um die Genauigkeit des Sensors zu verbessern, bekam der Sensor einen Kasten, der äußere Störfaktoren verhindern und genauere Ergebnisse liefern soll.

B. Einordnung der Steine

Die erste Idee zum Einordnen der Steine war eine Rampe, die mittels Winkel, die Steine in die richtigen Fächer einsortiert. Dabei stellte sich heraus, dass damit zu wenige Fächer (maximal 3) angesteuert werden würden. Eine andere Möglichkeit war ein Wagen am Ende des Fließbandes, der vor und zurück fährt. Damit konnten beliebig viele Fächer angesteuert und die benötigte Anzahl konnte gebaut werden. Die zweite Möglichkeit wurde umgesetzt, weil eine hohe Anzahl von Fächern für das Projekt benötigt wird und nur der Wagen diese bietet.

C. Mechanische Umsetzung

Um das Projekt umzusetzen bekam jede Gruppe einen Lego Mindstorm-Baukasten. Zusätzlich gab es einen NXT, zwei Motoren, einen Farbsensor und einen Tastsensor. Für die Grundlage wurde ein Gummiband über Ketten gespannt, um ein Fließband zu erzeugen, womit die Steine transportiert werden konnten. Dieses Fließband wurde dann statisch so fixiert, dass der Farbsensor nun auch angebaut werden kann. Der Farbsensor wurde am hinteren Ende des Fließbandes so angebracht, dass er von oben auf das Fließband blickt und so die Steine einzeln scannen kann. Dann bekam der Sensor seinen Kasten, welcher noch stabiler gebaut wurde, um ein umkippen zu verhindern. Nachdem Sensor und Fließband standen, wurde mit Zahnrädern ein Motor am Fließband befestigt und die Geschwindigkeit eingestellt. Anschließend wurde noch ein weiteres Fließband mit dem anderen Verbunden. Der Unterschied zum ersten war, dass die Übersetzung langsamer war. Durch das langsamere erste Laufband fallen die Lego-Steine vereinzelt auf das zweite und der Sensor bekommt immer einen Stein unter sein Blickfeld. Teil A des Farbroboters war fertig, nun folgt Teil B mit dem Wagen und den Fächern. Das Model des Wagens zum Auffangen der Steine ähnelt dem Güterwaggon eines Zuges. Es wurden lange Streben benutzt, um das Grundgerüst zu bauen. Daran werden anschließend jeweils 4 Räder befestigt. Zuletzt erhält der Wagen einen Motor, der die Hinterräder antreibt. Damit der Wagen immer im richtigen Abstand zum Fließband steht, wurde ein Platzhalter angebaut, wodurch die Ausrichtung immer stimmte. Kleine kosmetische Bauteile wurden noch hinzugefügt, wie eine Rampe gleich zum Anfang, um leichter die Steine auf das Fließband zu bekommen und noch eine weitere Rampe am Ende des schnelleren Fließbandes, beim Farbsensor, damit die Steine leichter und zielgenauer in die Fächer des Wagens fallen. Soweit ist der Roboter fertig und es fehlt nur noch der Programmierteil.

D. Funktionsweise des Programms

Die Einführung von der Programmiersprache MATLAB wurden in der ersten Wochen des zwei wöchigen Praktikums durchgeführt. In der zweiten Woche lag der Fokus auf den einzelnen Projekten der Gruppe. Der erste Schritt lag in der Erstellung eines Programmablaufplanes [1], um eine Übersicht zu bekommen, was die grundlegenden Schritte im Programm sind. Begonnen wird mit dem Tastsensor. Ist der Sensor nicht gedrückt (also 0), so fragt das Programm immer wieder den Tastsensor ab, bis er gedrückt wird (also 1 anzeigt). Sobald der

Tastsensor gedrückt wird, startet das Fließband und dreht solange, bis ein Stein unter dem Farbsensor liegt. Dann scannt der Farbsensor den Stein und stoppt den Motor der Fließbänder. Über eine if-Bedingung geht das Programm jede Farbe durch. Wenn er die richtige gefunden hat, folgt folgendes. Sobald das Fließband steht, bewegt sich der Wagen um einen gewissen Rotationswinkel und der Motor des Wagens stoppt. Wenn der Wagen steht, startet der Motor des Fließbandes wieder und der Stein rutscht über die Rampe in ein Fach der entsprechenden Farbe. Der Motor des Fließbandes läuft weiter, bis wieder ein Stein unter dem Farbsensor steht. Währenddessen bewegt sich der Wagen zu seiner Ausgangsposition zurück. Der Prozess wird solange durchlaufen, bis der Tastsensor nicht mehr gedrückt wird oder der NXT von MATLAB getrennt wird. Der Grund für den ständigen Durchlauf ist eine while-Schleife in der der Quellcode steht. Zu Beginn der Schleife wird der Tastsensor mit einer If-Bedingung abgefragt wie bei den Farben.

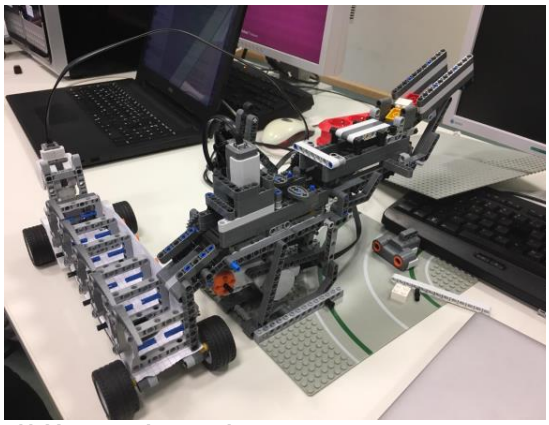


Abbildung 3: Farbsortierroboter

IV. Ergebnisdiskussion

Am Ende der zwei Wochen des Projektseminars wurde ein funktionierender Farbsortierroboter gebaut der Lego-Steine nach ihrer Farbe sortiert. Es wurden einige Tests durchgeführt wie Dauer des Transportes eines Steins von Start bis in sein Fach. Im Schnitt braucht ein Stein 13 Sekunden und daraus lässt sich schließen das 50 Steine um die 11 Minuten brauchen, sofern alles ohne Probleme abläuft. Bei einem weiteren Test wurde die Fehlerquote ermittelt und in einem selbsterstellten Diagramm dargestellt.

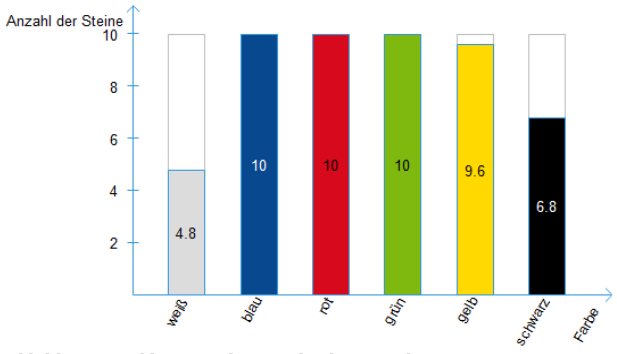


Abbildung 4: Fehlerquote der verschiedenen Farben

Bei dem Test wurden von jeder Farbe 10 Steine verwendet, welche vom Roboter einsortiert wurden. Dabei kam heraus, dass grün, rot, blau zu 100% korrekt zugeordnet wurden und gelb nur mit 96%. Jedoch gab es bei weiß und schwarz die größten Abweichungen. Dies hat folgenden Grund: um eine Farbe richtig zu erkennen, werden die Steine beleuchtet. Wenn der weiße Stein an der Reihe ist wirft dieser viel von dem blauen und grünen Licht zurück. Dabei bleibt das rote Licht auf der Strecke. So wird weiß, bei grün oder blau meist eingeordnet. Das gleiche gilt für schwarz. Wenn die Oberfläche der Steine nicht glatt, sondern rau sei, so würde der Sensor genauer arbeiten können, wodurch weiß und schwarz genauer einsortiert werden könnten.

A. *Probleme*

Bei der Umsetzung des Projektes kam es zu Problemen. Ein Problem entstand, als der Code geschrieben wurde, denn beim ersten Versuch gab der NXT die ganze Zeit einen Ton von sich. Dies war ein Zeichen dafür, dass zu viele Befehle auf einmal an den NXT gesendet wurden und er dadurch überlastet war. Das Problem wurde durch Befehle wie „WaitFor()“ unterbunden und das Programm lief wie geschnitten Brot. Ein zweites Problem tauchte beim Start und Stopp Knopf in der GUI auf. Durch die while-Schleife wird immer wieder das Programm durchlaufen. Wenn der Stopp-Knopf betätigt wird, so stoppt das System nur für den Moment, indem der Knopf gedrückt wurde. Um das zu lösen, wurde ein Tastsensor als Schalter eingebaut, der noch in die while-Schleife kommt, wodurch das System vor der Farbabfrage den Schalter abfragt. So kann das System mit dem Schalter gestartet und gestoppt werden. Das war es noch nicht mit den Problemen, ein letztes gibt es noch. Beim ungünstigen Hinlegen der Steine auf das Fließband kann es hin oder wieder passieren, dass sich die Steine verkeilen und den Roboter auseinander drücken, oder zu Steingeschossen werden. Dieses Problem entstand allein durch die Art des Reinlegens der Steine. Die spartanische Lösung war es die Steine einzeln reinzulegen. Es gibt auch die Möglichkeit, das Problem zu nutzen und jemanden mit den Steingeschossen abzuschießen. Beim Bau des Roboters wurde festgestellt, dass ein Lego-Baukasten allein nicht ausreicht und mindestens ein zweiter benötigt wurde.

V. Zusammenfassung und Fazit

Das Projekt hat in dem Umfang mir viel Freude bereitet und dabei noch eine neue Programmiersprache gelehrt. Das Seminar hat die Zusammenarbeit zwischen den Kommilitonen gefördert und wir halfen uns gegenseitig bei Problemen. Die Arbeitsteilung im Team funktionierte einwandfrei.

Wenn das Projektseminar länger als zwei Wochen ginge, 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.

ANHANG

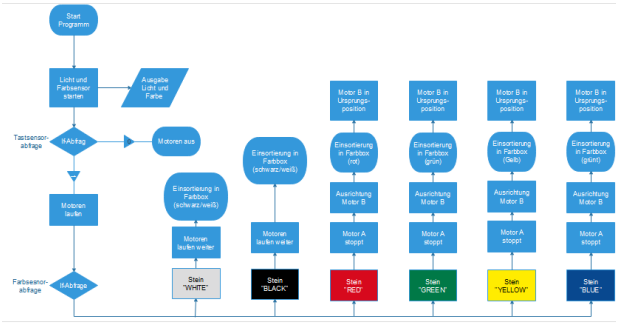


Abbildung 5: Programmablaufplan

Literaturverzeichnis

Informationen aus dem Internet:

- https://www.tomra.com/de-de/sorting/food/your-produce/fruit/tomatoes

Abbildungen :

1. https://www.tomra.com/-/media/images/sorting-solutions/food/working-principles/iris-wp.ashx?&h=210&hash=65C5F92B8C4D781AC8C79560477CB15CB527896D
2. Vortrag Sensor von OVGU mit Snipping Tool ausgeschnitten
3. Eigene Aufnahme
4. Eigene Aufnahme
5. Eigene Aufnahme

LEGO Farbsortierroboter

Noah Salomo, Elektro und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

Wie seit vielen Jahren gab es auch dieses Jahr ein Lego Mindstorms Projektseminar an der OVGU mit vielen praktischen und innovativen Projekten. Dieses Paper befasst sich mit einem Roboter der Legosteine nach Farbe Sortiert. Dabei wird auf die Technische Umsetzung des Roboters sowie auf die Programmierung eingegangen, aber auch die Probleme welche das Projekt mit sich brachte wurden nicht ausgelassen. Unter anderem wird die Idee Vorgestellt so wie der Erfolg oder vielleicht Misserfolg des Projektes beleuchtet.

I. Einleitung

In einer Welt, in der nahezu alles von Geschwindigkeit und geringen Kosten, also Effizienz beherrscht wird muss man, um mit seinen Konkurrenten mithalten zu können, immer auf dem technisch neuesten Stand sein. Das heißt im Moment, so kommt es einem fast schon vor, möglichst wenig Arbeiten von Hand auszuführen und so viel Arbeit wie nur irgendwie möglich von autonomen Maschinen verrichten zu lassen. Eine Tätigkeit, die sich nahezu in jeder Industrie finden lässt und welche sich auch noch hervorragend dazu eignet automatisiert zu werden, da sie wenig eigenständiges Denken und dennoch ziemlich viel Zeit benötigt, ist das Sortieren von Baumaterialien, Post, Abfall, Lebensmittel, oder, um die Liste hier mal abzukürzen, von sozusagen allem. Nun ist es leider so, dass es in der Industrie kaum noch Platz für Innovationen in der Sortierung gibt, da die Automatisierung dieser Tätigkeiten bereits über Jahrzehnte hinweg vorangetrieben wurde. Ein Bereich, in dem jedoch noch relativ wenig Automatisierte Arbeit zu finden ist und in dem weiterhin eine kleine Verbesserung noch keine großen Veränderungen verursacht, ist unser tägliches Privatleben und unser Haushalt. Dieser Bereich unseres Leben steht erst am Anfang seiner Automatisierung. Erst seit den Letzten paar Jahren gibt es Roboter, die uns den Alltag Erleichtern und zum Beispiel den Rasen mähen oder Staubsaugen. Auch im Bereich der automatischen Sortierung gibt es im Haushalt noch viel Platz für praktische, aber dennoch einfache Innovationen. Wer wünscht sich nicht, dass die Eigene Wäsche automatisch nach Farbe und Waschart sortiert, gewaschen und natürlich auch wieder perfekt zusammengelegt im Schrank einsortiert wird? Auch im Kinderzimmer können Kinder sowie Eltern von automatisiertem Sortieren profitieren. Welches Kind hat sich nicht schon mal gewünscht, es müsste nicht immer mühsam das benötigte Teil beim Lego spielen aus der letzten Ecke der Legokiste heraus kramen, oder die gesamte Kiste auskippen, welche im Nachhinein unter lautem Gemeckere der Eltern wieder eingeräumt werden muss. Mit (fast) diesem Hintergedanken kam die Idee des Lego Sortierroboters zu Stande, welcher wie der Name schon sagt genau für diese Situation gemacht ist. Man müsste nie mehr nach Legosteinen suchen und hätte sie immer perfekt Sortiert zur Hand. Genau mit dieser Vision vor Augen wurde der Lego Farbsortierroboter beim diesjährigen Projektseminar entwickelt und gebaut.

DOI: 10.24352/UB.OVGU-2019-037

II. Hauptteil

Die Idee

Die Idee des Farbsortierroboters ist es, einen Haufen bunter Legosteine in viele kleinere gleichfarbige Häufchen zu verwandeln oder diese Häufchen direkt in Kisten Einzusortieren. Dafür wird ein Roboter benötigt der die Legosteine im besten Fall selbständig aufnimmt oder diese über ein Trichter zugeführt bekommt. Dann soll er die Farbe der Steine erkennen, unerwünschte Steine entfernen und die übrig gebliebenen zuletzt nach Farben sortiert entweder auf verschiedene Lagerplätze legen oder die Steine direkt in dafür bereitstehende Gefäße einsortieren. Für die erste Idee würde man einen Arm benötigen, welcher automatisch die Position der Steine erkennt und diese dann selbständig aus dieser Position heraus nimmt und sie unter einem Farbsensor platziert - oder diesen alternativ direkt am Arm platziert hat. Zuletzt müsste der Arm die Steine dann auf der dafür vorgesehenen Position platzieren und sie wieder loslassen. Diese Idee wurde sehr schnell wieder verworfen, da sie eine technische Präzision erfordert, welche mit Legosteinen nur sehr schwer zu erreichen ist. Außerdem benötigt die Erkennung der Steine im Raum und das richtige Greifen dieser gute Informatikkenntnisse, welche sich kaum in einem halben Jahr C++ und einer Woche Vorbereitung aneignen lassen. Aus diesen zwei Gründen wurde die Idee, dass man den Roboter mit Steinen Füttern muss, weiterverfolgt. Um diesen Roboter zu verwirklichen wird als erstes eine Art motorisiertes Fließband benötigt, welches die Steine zu einem Farbsensor transportiert, der dann die Farbe der Steine erkennt. Der Farbsensor muss adäquat befestigt sein und die Farbe möglichst fehlerfrei erkennen. Als nächstes müssen die Steine zu ihrer finalen Position befördert werden und ungewollte Steine entweder vom Fließband gestoßen oder an eine geeignete Stelle gebracht werden. Dies kann entweder über eine Rampe realisiert werden, welche sich in die entsprechende Position begibt, oder über ein weiteres Fließband. Um die richtige Verteilung der Steine zu erreichen kann sich entweder der Roboter selbst, ein Teil des Roboters oder die für die Steine vorgesehenen Kisten bewegen. Der letzte wichtige Punkt ist, dass der Farbsensor immer nur genau einen Stein gleichzeitig erkennen darf, was entweder mit einem Stopper oder mit einer großen Geschwindigkeitsdifferenz zwischen zwei Fließbändern realisiert werden kann.

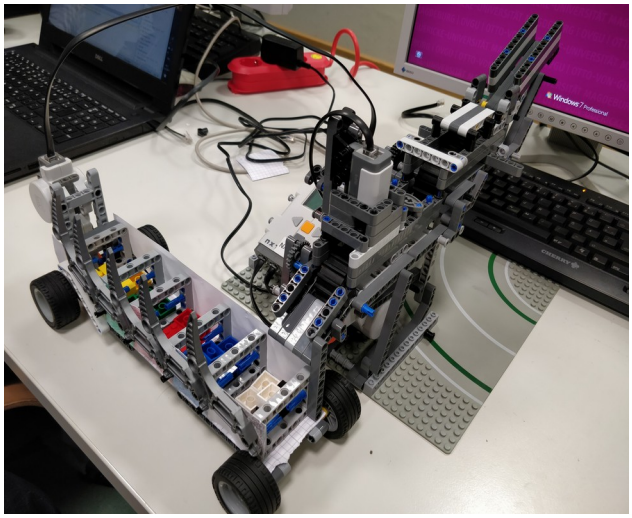


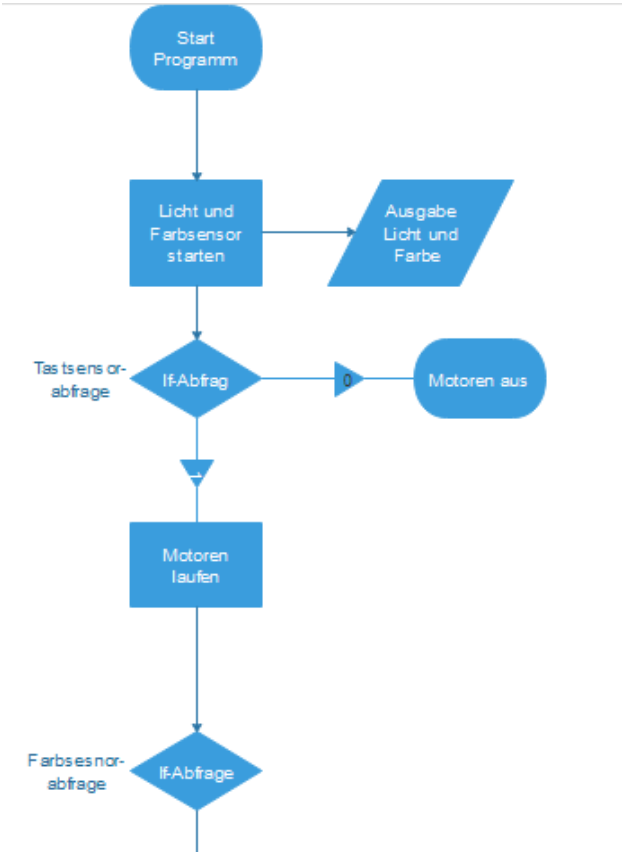
Abbildung 1: Fertiggestellter Roboter

Auch im Hinblick auf die Programmierung gibt es einige Möglichkeiten. Es gibt zwei verschiedene Wege die Farbe der Legosteine zu erkennen: zum einen kann man einen Farbsensor direkt von Lego verwende,. welcher sehr leicht in das Programm eingebunden werden kann. Dieser erkennt aber nur relativ wenig verschiedene Farben. Die zweite Möglichkeit ist eine Webcam, bei der man selbst entscheiden kann wie viele Farben erkannt werden sollen. Bei dieser ist die Einbindung in das Programm anspruchsvoller und die eigentliche Farberkennung durch die Kamera muss selbst programmiert werden. Auch hier wieder mit Blick auf die unzureichenden Programmierkenntnisse wurde die zwar bessere, dafür aber kompliziertere Idee mit der Webcam nicht weiterverfolgt. Von dem Eben genannten Punkt einmal abgesehen ist die restliche Programmierung recht gradlinig. Am Anfang muss der Motor der das Fließband antreibt eingeschalten werden. Dann muss das Fließband gestoppt werden, sobald der Farbsensor eine andere Farbe als die des Fließbandes erkennt

und als letztes muss ein zweiter Motor eingeschaltet werden, der dafür sorgt das der Stein an seine Richtige Position kommt. Die einzigen zwei Fragen die noch bleiben sind, ob man auswählen können soll welche Farben einsortiert werden und welche nicht und ob der Roboter nur über ein Graphical User Interface, also ein GUI gesteuert wird oder ob man ihn auch noch über einen Schalter ein- und ausschalten kann.

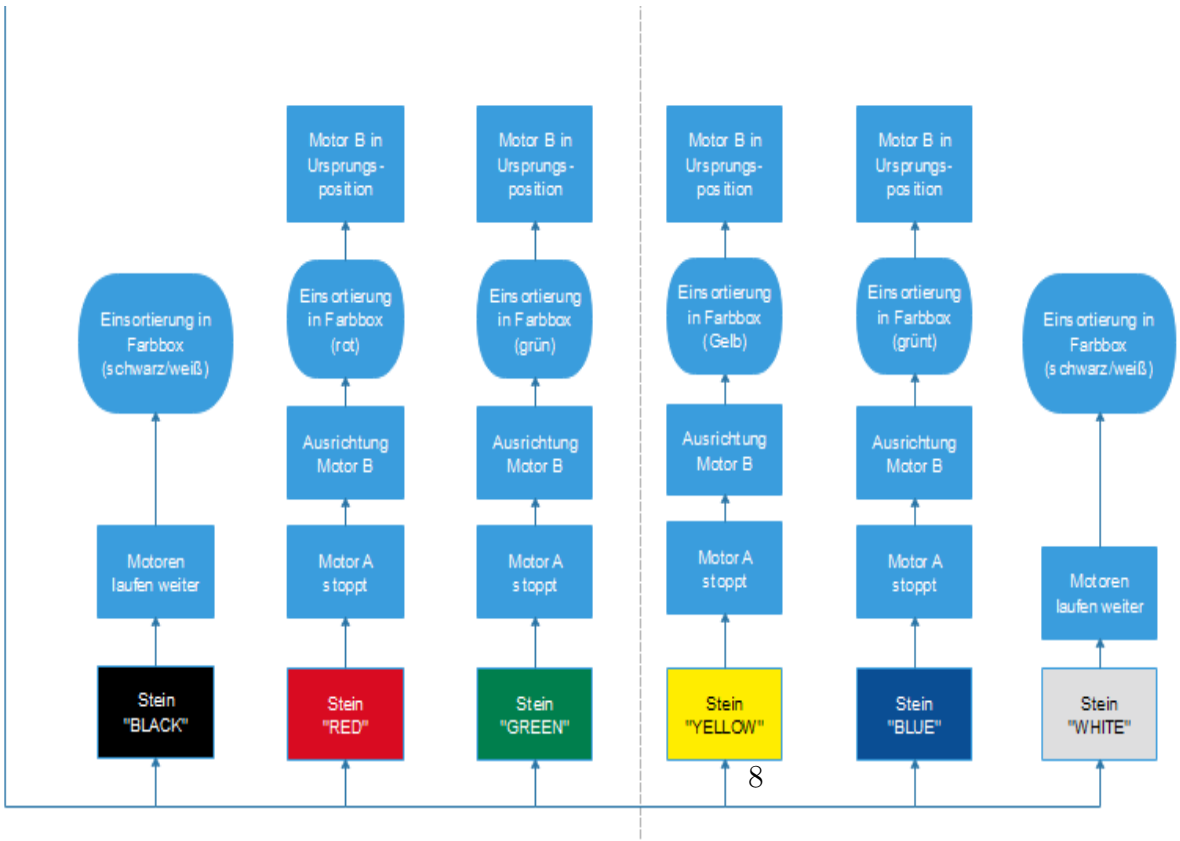
Die Umsetzung

Nach einer Woche der Einführung in MatLab (der bei diesem Projekt verwendeten Programmiersprache) und dem Vertrautmachen mit dem Lego Mindstorms Baukastens begannen alle Gruppen mit ihrem Projekt. Als Bausteine für ihr Projekt erhielt jede Gruppe einen Baukasten welcher neben den bekannten Legotechnic Bausteinen auch ein NXT (das Steuermodul des Roboters), zwei Motoren, einen Tast, Licht, Ultraschall,Farbsensor. Da unsere Gruppe aus drei Mitgliedern bestand wurde Zeitgleich mit der Konstruktion des Roboters sowie mit seiner Programmierung begonnen. Die Konstruktion wurde damit begonnen das ein Gerüst für das Fließband gebaut wurde welches die Lego Steine unter den Farbsensor befördert. Mittig in diesem Gerüst sitzt der Motor welcher das Fließband über vier Zahnräder antreibt und am oberen teil des Gerüstes ist das Fließband befestigt. Links und recht des Fließbandes ist jeweils eine kleine Wand um das herunterfalle der Legosteine zu verhindern und mittig am linken Ende des Fließbandes sitzt der Farbsensor ungefähr 3cm über dem Fließband in einem Kasten der ihn von einfallendem Licht abschirmt welches die Genauigkeit des Sensors beeinträchtigt. Bei Tests wurde sehr schnell festgestellt das eine Vorrichtung benötigt wird welche dafür sorgt das immer nur genau ein Stein unter dem Sensor ist da die Steine sonst nicht richtig sortiert werden können. Es wurde sich gegen eine Schranke entschieden da diese einen weiteren Motor benötigen und das Programm unnötig verkomplizieren würde. Aus diesem Grund wurde auf anraten von Herrn Magdowski eine rein technische Lösung eingebaut. Ein zweites Fließband rechts etwas oberhalb des Ersten wird von dem selben Motor viel langsamer Übersetzt angetrieben was dafür sorgt das selbst wenn die Steine auf dem Zweiten Fließband dicht an dicht liegen ist auf dem ersten immer noch genug Platz zwischen den Steinen das nicht zwei gleichzeitig einsortiert werden. Auf der linken Seite des Roboters ist die Vorrichtung die dafür sorgt das die Steine an ihren Vorgesehenen Platz befördert werden. Es wurde sich dort gegen eine Rampe entschieden stattdessen gibt nun einen Kleinen Wagen der mit fünf Fächern ausgestattet ist in welche die Legosteine einsortiert werden. Der Wagen wird von einem weiteren Motor bewegt und wird auch an den NXT angeschlossen kann aber zum Transport einfach von ihm getrennt werden. Am Schluss wurde sich gegen eine Vorrichtung entschieden welche Steine vom Fließband schiebe die nicht mit einsortiert werden sollen und einfach eines der fünf Fächer für diesen Zweck benutzt. Nach mehren Testläufen stellte sich das Problem heraus das sich die Steine unter dem Sensor verklemmten und von dem aus Gummi gemachten Fließband weg geschossen wurden deshalb wurde über den Fließbändern eine Art Decke eingebaut welche das verhinderte. Dazu gab es auch noch einen haptischen Schalter und eine Rampe welche die Steine auf das zweite Fließband leitet als Erweiterung.



Erster Teil der Programm Ablaufplanes

Zweiter Teil Des Programmablaufplanes



Das Programm welches dem Roboter sein „Leben“ gibt besteht nur aus einer Reihe an If-Abfragen und Schleife. Nach dem Start des Programmes wird als erstes geguckt ob der Schalter an ist und dann wird dementsprechend das Fließband angeschaltet oder nicht. Wenn das Fließband an ist werden diese solange weiterlaufen bis der Farbsensor eine andere Farbe erkennt als das Schwarz des Fließbandes und sobald das eintritt stoppt das Band. Dann wird der Wagen mit den verschieden Kisten in die der Farbe entsprechende Position gefahren und das Fließband läuft genau soweit weiter das der Stein in das Fach fallen kann. Danach fährt der Wagen wieder in seine Ausgangsposition zurück und dann fängt das Programm wieder an zu gucken ob der Schalter an ist. Die einzige Ausnahme tritt ein wenn der Sensor einen Stein erkennt denn man nicht haben möchte dann verschiebt sich der Wagen einfach nicht, das Fließband läuft einfach weiter und der Stein Fällt in das erste Fach des Wagens. Da es im Wagen genug Fächer für alle Farben gab die der Farbsensor erkennen kann und auch noch eines

für alle anderen wurde sich gegen eine GUI entschieden über die man einstellen konnte nach welchen Farben man sortieren möchte und nach welchen nicht. Somit konnte der Roboter ohne großartige Probleme Konstruiert, Programmiert und Vorgestellt werden.

III. Fazit

Als Fazit lässt sich festhalten das der Bau des Roboters fast Problemlos über die Bühne lief und auch Programmiertechnisch einwandfrei funktioniert hat. Natürlich lassen sich auf jeden Fall noch viel Verbesserungen Einbau wie zum Beispiel eine Webcam um nach mehr Farben Sortieren zu können oder auch der eingangs erwähnte Greifer welcher sich automatisch Steine nimmt. Als letzter Satz soll aber festgehalten werden das der Roboter das zuverlässig tut wofür er erdacht wurde und somit ein voller Erfolg in die Geschichte des Logo Mindstorm Projektseminars der OVGU eingeht.

Der Farbsortierroboter

Lukas Zimmermann, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

Wie seit 2013 jedes Jahr fand auch 2019 wieder das Lego Mindstorms Projektseminar der Otto von Guericke Universität Magdeburg statt, es gab wieder viele Projektideen, welche Ausgearbeitet, Realisiert und am Ende vor einer Jury und Studenten vorgestellt wurden. So auch der Farbsortierroboter, mit welchem sich in diesem Paper befasst wird. Seine Aufgabe ist es Lego Steine zu scannen und anschließend durch einen Mechanismus nach Farben einzusortieren. Der Farbsortierer brachte ein paar Probleme bei Bau und Programmierung mit sich, auf diese Probleme und ihre Lösungen wird im weiteren Verlauf des Papers eingegangen.

Schlagwörter: Farbsortierer, Lego, Mindstorms, NXT, Otto von Universität, OvGU, Projektseminar

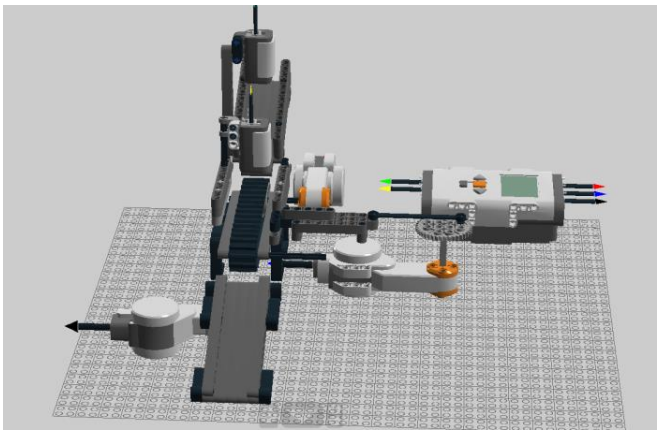
I. Einleitung

Wie in unserer heutigen Gesellschaft üblich wird immer mehr automatisiert und autonomisiert, ob das Autos, Produktionsabläufe oder Abläufe in unserem Alltag sind, alles wird smarter und eigenständiger. Es gibt mittlerweile in fast allen Bereichen Geräte bzw. Maschinen die selbstständig ihre Arbeit verrichten, ob das nun Staubsaugerroboter, welche der WG helfen den Boden wenigstens einigermaßen begehbar zuhalten oder Kühlschränke die einem sagen was noch drin ist und fragen ob sie neue Milch bestellen sollen, sind.
Da sich gerade die Automatisierungstechnik für uns als interessant darstellte, hatten wir überlegt was wir bauen könnten, schnell wurde die Idee geboren einen Roboter bzw. eine Maschine zu entwickeln, welche Dinge nach ihrer Farbe sortieren kann. Als Einsatzzweck könnte er in einer Abgewandelten Form in z.B. Kitas zum Einsatz kommen um die Lego, oder Duplo Steine zu sortieren. Auch wäre es möglich ihn abzuwandeln um damit andere Dinge zu sortieren, beispielsweise Münzen, um die Zählung von Kassen zu erleichtern. Es gibt viele verschiedene Gebiete in welchen man den Farbsortierroboter in verschiedenen abgewandelten Formen einsetzen könnte diese Vielseitigkeit in seinem Nutzen war für uns interessant. Die Anforderungen, welche er erfüllen sollte, waren erstmal eine zuverlässige Sortierung von Legosteinen in der Größe 2 x 2.

II. Hauptteil

Die Idee war es einen Roboter beziehungsweise eine automatisierte Vorrichtung zu entwerfen, welche Legosteine

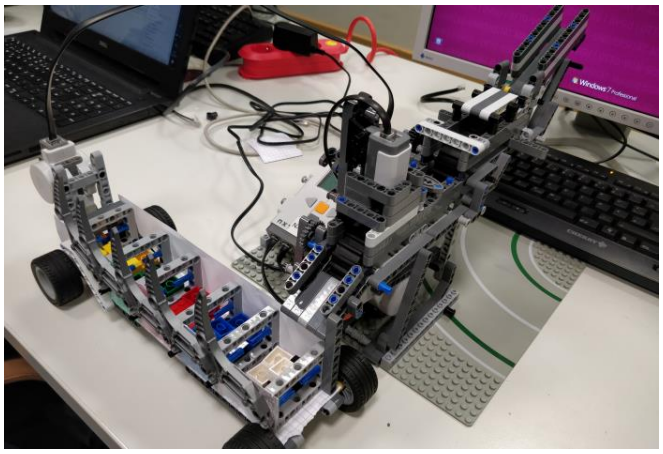
nach ihrer Farbe sortiert. Dieser Roboter sollte Steine, die auf ihn gelegt bzw. befördert wurden, vereinzeln können und selbstständig einsortieren können.
Relativ schnell wurde ein Entwurf erstellt wie in der Abbildung 1 zusehen.



(Abb. 1) Erster Entwurf des Farbsortierroboters

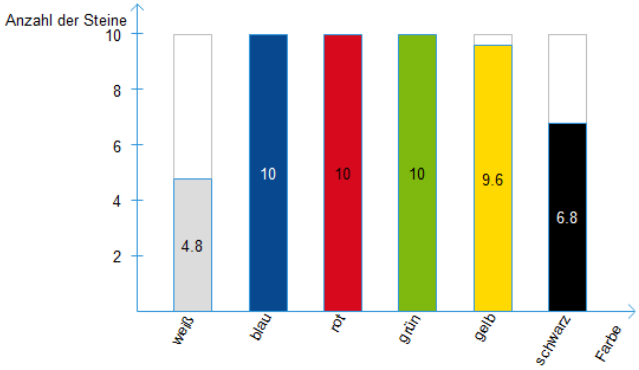
In diesem ersten Entwurf ist ein Fließband, der Farbsensor, der NXT und drei Motoren zusehen. Es war ursprünglich geplant, dass ein Motor das Fließband bewegt. Ein zweiter Motor sollte eine Rutsche am Ende des Fließbandes immer so verstellen, das wenn z.B. ganz links immer die gelben Steine rein sollen, die Rutsche bei erkennen eines gelben Steines ganz nach links fährt. Der dritte Motor sollte einen Stempel bewegen, der Steine vom Förderband schiebt, welche nicht zu den Farben gehören die einsortiert werden sollen, z.B. ein schwarzer Stein. Bei diesem Entwurf war aber nicht klar wie die Vereinzelung der Steine realisiert werden sollte. Aus diesem Grund wurde ein neuer Entwurf erstellt, welcher leider nicht bildlich festgehalten wurde in der Abbildung 2 ist jedoch der Fertige Roboter zusehen, an diesen kann man den Aufbau jedoch klar erkennen.

DOI: 10.24352/UB.OVGU-2019-038



(Abb. 2) Zweiter Entwurf des Farbsortierroboters

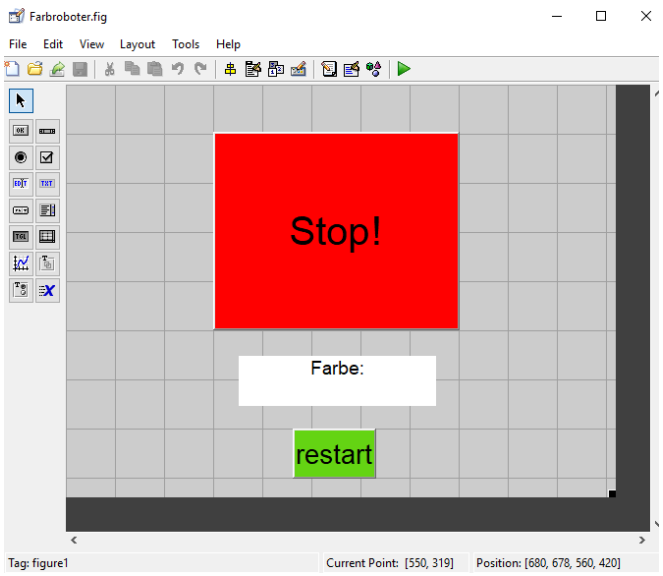
Wie zusehen ist, hat sich der Aufbau nochmal stark geändert, statt der anfangs drei Motoren sind es nur noch zwei, welche leider etwas schlecht zu sehen sind. Bei diesem Entwurf wurden dann zwei Förderbänder statt eins genommen, das obere Förderband ist dabei durch eine Übersetzung langsamer als das andere, dadurch sollte sichergestellt werden, dass die Steine in einem vergrößerten Abstand bei dem Farbsensor ankommen.
Sortiert werden die Steine in dieser Version nicht durch eine Rutsche die sich bewegt, sondern durch einen Wagen der je nach erkannter Farbe den richtigen Behälter vor dem Fließband platziert.
Als letztes wurde noch überlegt wie man den Roboter auch ohne GUI starten und stoppen konnte, das wurde durch einen Schalter der durch einen einrastenden Hebel entweder fest auf „AUS“ oder fest auf „EIN“ gestellt werden konnte, realisiert.
Bei der Software war schnell klar, dass das Fließband halten sollte, sobald der Farbsensor einen Stein erkennt, um zu gewährleisten, dass der Wagen mit den Behältern genug Zeit hat sich richtig vor dem Fließband zu positionieren.
Nun zum genauen Aufbau des Farbsortierers, wie gesagt wurden zwei Förderbänder eingesetzt, wovon eines langsamer läuft als das andere, so wurde sichergestellt, dass die Steine einen größeren Abstand zu einander bekommen und der Farbsensor so nicht durcheinanderkommt. Dieser Aufbau reichte jedoch noch nicht um die Steine Zuverlässig zu vereinzeln, also wurde die Rutsche, die die Steine auf das Förderband bringen sollte, schmaler gemacht. Im Anschluss konnte nebeneinander nur noch maximal ein Stein passen und so wiederum wurde sichergestellt, dass die Steine nacheinander auf das Förderband rutschen. Als nächstes Problem wurde jedoch schnell festgestellt, dass die Steine trotzdem noch übereinander auf dem Förderband landen konnten, dies wurde durch eine Art „Dach“ über dem Förderband weitestgehend eliminiert. Zwar rutschten trotzdem ab und zu noch Steine, welche zu nah aneinander waren, durch, jedoch war es mehr die Ausnahme als die Regel. Da solch ein Dach auch nochmal auf dem zweiten Förderband platziert wurde, war das Problem quasi ganz behoben. Als Nächstes kam der Farbsensor ins Gespräch, bei diesem war die Zuverlässigkeit für das Vorhaben sehr wichtig, schnell kristallisierte sich jedoch heraus, dass er so seine Macken hat, da geplant war Steine in den Farben: Gelb, Blau, Grün, Weiß, Schwarz und Rot sortieren zu können, wurde ein Versuch gestartet, wie zuverlässig er die einzelnen Farben erkennen könnte. Im folgenden Diagramm ist ein klares Muster zu erkennen.



(Abb. 3) Diagramm der Zuverlässigkeit, der Farberkennung

Die Farben selbst erkannte er sehr zuverlässig, darunter Blau, Rot und Grün zu hundert Prozent, bei Gelb gab es eine kleine Abweichung und bei Schwarz und Weiß gab es eine böse Überraschung, Schwarz erkannte er wenigstens in über fünfzig Prozent der Fälle, Weiß jedoch in unter fünfzig Prozent. In Folge musste eine Lösung gefunden werden, schnell kamen wir auf die Idee den Sensor abzuschotten um das Erkennen der Farben des Sensors nicht vom Umgebungslicht abhängig zu machen, dies brachte jedoch nur leichte Besserung, dann wurde sich der Sensor genauer angesehen, da viel auf, dass der Sensor drei LEDs verbaut hatte, eine rote, eine grüne und eine blaue. Darin sahen wir das Problem, da die Steine nicht matt sind, sondern leicht reflektieren, kann es passieren, dass wenn eine LED von einem weißen oder schwarzen Stein mehr reflektiert wird als die anderen, der Farbsensor so durch die Reflektion eher z.B. grün erkennt als weiß. Dieses Problem war bis zum Ende leider nicht lösbar.
Nun zum Wagen, welcher die Behälter für die Steine enthält, er enthält fünf Behälter, jeder Behälter ist fest für eine Farbe vorgesehen.
Im ersten Behälter (Abb. 2 ganz rechts) sollten die weißen und schwarzen Steine rein, in den nachfolgenden Behältern dann von rechts nach links, zuerst die blauen, dann die roten, dann die grünen und dann im letzten die gelben Steine landen. Programmiertechnisch wurde das alles durch die Motoren selbst geregelt. Damit ist gemeint, dass die Umdrehungen, bzw. der Weg der gefahren werden sollte durch den Tachowert des Motors festgelegt wurde, diesen Wert, haben wir rein durch probieren und immer wieder anpassen der Werte herausgefunden. Zum Ablauf, sobald ein Stein vom Farbsensor erkannt wurde, sollte das Förderband stoppen, in dieser Zeit sollte der Wagen, den richtigen Behälter vor dem Förderband platzieren und ab dem Moment, ab dem er richtig steht, fährt das Förderband wieder an, für dieses Anfahren haben wir eine bestimmte Umdrehung festgesetzt, innerhalb dieser Umdrehung wurde sichergestellt, dass der Stein in seinem Behälter landet, nach dieser bestimmten Umdrehung bleibt das Förderband wieder stehen und es wird abgewartet bis der Wagen wieder auf seiner Ausgangsposition steht, sobald er das

tut, läuft das Förderband wieder ganz normal weiter bis zum nächsten Stein.
Das Programm sollte nachdem alles funktionierte mit einer GUI (Benutzeroberfläche) gesteuert werden können, in dieser sollte es möglich sein, zum einen den Roboter zu stoppen und anschließend wieder zu starten, auch sollte man sehen was der Farbsensor gerade für eine Farbe erkannt hat.
Den Entwurf der GUI sieht man im folgenden Bild.



(Abb. 4) Die GUI zum Steuern des Farbsortierers

Da die GUI gerne mal Probleme bereitete, wurde entschieden, dass der Roboter, wie bereits erwähnt, noch einen Schalter zum Ein- und Ausschalten benötigte, dieser wurde durch einen Taster realisiert, der Taster wurde so verbaut, dass er durch einen Hebel auf „AN“ einrasten konnte, so lief der Roboter nur wenn der Hebel eingerastet war und ging aus sobald der Hebel auf „AUS“ gestellt wurde.
Nachdem alle angestrebten Funktionen grob liefen, wurden die ganzen Werte angepasst und so wurde die Genauigkeit des Roboters erhöht, es wurde experimentiert, ob die Förderbänder nicht etwas schneller laufen könnten, jedoch wurde so das Risiko, dass die Steine nicht gut genug vereinzelt werden oder sich gar verklemmen, stark erhöht.
Auch wurde versucht, dass der Wagen sich schneller bewegt, jedoch ist dieser dadurch nach ein paar Steinen immer ungenauer geworden, das kam nicht in Frage und so wurde er wieder langsamer.

III. ERGEBNISDISKUSSION


(Abb. 5) Hier gut zu erkennen, die Abschottung des Farbsensors und die Dächer zur Steinvereinzelung

Wie in der Abbildung 2 zusehen, hat es letzten Endes der zweite Entwurf in den Bau geschafft, dieser Entwurf war für uns am besten umzusetzen und hat am Ende auch überraschend gut funktioniert. Ein paar Probleme sind jedoch bekannt und darunter gibt es auch welche, die bis zuletzt nicht gelöst werden konnten, so zum einen die schon erwähnte Fehlerrate bei weißen und schwarzen Steinen, diese war vom Farbsensor abhängig und so mussten wir damit leben, bis auf die Abschottung (Abb. 5), welche eine kleine Besserung brachte, haben wir keine weiteren Möglichkeiten gefunden, um dieses Problem zu lösen. Die Zuverlässigkeit der Erkennung der anderen Farben war dennoch sehr überraschend, denn es ist immer noch Lego womit dort gearbeitet wurde. Als nächstes großes Problem, kam die Steinvereinzelung und die Minimierung der Möglichkeiten zum Verklemmen der Steine in den Vereinzelungsvorrichtungen. So wurde darauf geachtet, dass die Wände der Förderbänder möglichst glatt sind und auch die erwähnten Dächer, welche das übereinanderliegen der Steine verhindern sollten, möglichst abgerundet gestaltet werden. Bis es soweit war, dass die Steine wenigstens einigermaßen zuverlässig vereinzelt wurden, verging einige Zeit, da es bei verschiedensten Ausführungen immer wieder zum Verklemmen der Steine kam. In den ersten Ausbaustufen, ist es des Öfteren vorgekommen, dass die Steine so stark unter das „Dach“ geklemmt wurden, dass das Dach ab einem bestimmten druck nicht mehr standhalten konnte und so mit viel Kraft nach oben geschleudert wurde, diese Dinge konnten durch ein verstärktes erstes Dach und durch eine verschmälerte Rutsche minimiert werden. (Abb. 5)
Ein weiteres, zwar nicht so großes aber doch vorhandenes Problem, ist die Schnelligkeit, des Sortiervorgangs, um die Genauigkeit aller Komponenten gewährleisten zu können, musste die Geschwindigkeit der Motoren relativ stark runtergenommen werden, vor allem, die des Wagens, da er sonst auf Dauer einfach zu ungenau wurde. Auch die kleinen Pausen zwischen den Vorgängen, sind für die Schnelligkeit nicht vorteilhaft. Dieses Problem könnte man durch einen anderen Aufbau ohne Wagen regeln, da die Lösung mit wagen für uns aber am besten funktionierte und es in diesem Aufbau

nicht um Schnelligkeit ging, war das für uns kein Problem. Wollte man aber wirklich eine größere Zahl an Steinen sortieren, so müsste man den Aufbau abwandeln oder man bräuchte viel Geduld. Lösungsansätze wurden dafür nicht direkt besprochen, da schlicht, nicht erforderlich, müsste der Farbsortierroboter aber schneller sein, dann müsste man entweder den ersten Entwurf nochmal ausgraben, da sich bei diesem sicherlich durch die bewegliche Rutsche, die Förderbänder schneller bewegen könnten, mit Sicherheit kann ich das aber nicht beantworten.
Ansonsten kann man zum Endergebnis nur eines sagen, es funktioniert!
Der Farbsortierroboter macht das was er soll, er sortiert Legosteine in der Größe 2 x 2 ziemlich zuverlässig nach ihren Farben, wenn man weiß und schwarz außer Acht lässt. Was uns besonders Stolz macht, ist sein kompakter und stabiler Aufbau, wodurch er sich sehr gut transportieren lässt und einen äußerst robusten Eindruck macht.

IV. Zusammenfassung und Fazit

Wie bereits mehrmals erwähnt ist am Ende ein überraschend gut funktionierender Farbsortierroboter, rein aus Lego, zustande gekommen. Er macht das was er soll, und das bestehend aus zwei Motoren, einem Farbsensor, einem Schalter und einem NXT. Die Zuverlässigkeit ist beeindruckend, wenn man die gennannten Probleme außer Acht lässt. Und seine Vielseitigkeit ist mit etwas Anpassung und anderen Komponenten unendlich. So hätten wir gerne noch versucht Münzen zu sortieren, jedoch ist der Farbsensor, dafür einfach zu ungenau. Auch die Steinvereinzelung klappt sehr gut, wenn man es mit den zugeführten Steinen nicht übertreibt. Bei der Umsetzung gab es bis auf die genannten Probleme keine großen Schwierigkeiten, bei den Programmiervorgängen hakte es ab und zu mal, aber meistens lief es trotzdem wieder nach kurzer Zeit.

Lego Mindstorms Vier-Gewinnt-Roboter

Robin Henryk Drews, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In diesem Paper geht es um die Entwicklung und Konstruktion eines autonom spielenden Vier-Gewinnt-Roboters mit Lego Mindstorms. Dazu wird der MiniMax-Algorithmus und die Spielerkennung per Webcam verwendet. Der implementierte Algorithmus basiert auf der Spieltheorie. Dies soll dem Roboter ermöglichen intelligente Spielzüge zu machen. Auch die Realisierung des Roboters wird in diesem Paper verdeutlicht, in dem die Konstruktion und der Einwurfmechanismus erklärt werden.**

***Schlagwörter*—Lego Mindstorms, Vier-Gewinnt, Projektseminar, Spieltheorie, MiniMax-Algorithmus, Robotik.**

I. Einleitung

HEUTZUTAGE sind künstliche Intelligenzen ein wichtiges Thema, sie findet in vielen Bereichen unseres Lebens Anwendungsgebiete, z.B. in der Diagnostik, bei der Gesichts- oder Bilderkennung und in Computerspielen. Besonders in Computerspielen hat sie eine zentrale Rolle eingenommen. Somit ist es nicht mehr zwingend notwendig, einen anderen menschlichen Gegenspieler zu haben. Jedoch braucht die künstliche Intelligenz dabei ein an Menschen orientiertes Spielverhalten, um ein realistisches Spielgefühl zu erzeugen. Dieses Spielverhalten ist besonders bei Gesellschaftsspielen relevant. Um eine künstliche Intelligenz für diese Spiele zu entwickeln, kommt oftmals die mathematische Spieltheorie zum Einsatz. In Spielen wie Skat, Dame oder auch 4-Gewinnt findet das Spielgeschehen nur auf dem Bildschirm statt. Um dazu eine Alternative zu schaffen wird in diesem Paper ein Roboter behandelt, welcher an einem physischen Spielfeld 4-Gewinnt spielen kann. Dabei wird das Spielfeld mittels einer Webcam erkannt und dann ausgewertet. Der Roboter berechnet darauf beruhend seinen eigenen Spielzug und wirft den Spielstein autonom ein.

II. Vorbetrachtungen

Um eine künstliche Intelligenz für den Roboter zu entwickeln muss das Spiel zunächst genauer beschrieben werden. Danach können Ansätze aus der Spieltheorie angewandt werden.

A. 4-Gewinnt

4-Gewinnt ist ein zwei Personen Gesellschaftsspiel, bei dem es das Ziel ist, 4 der eigenen Steine diagonal, horizontal oder vertikal in einer Reihe zu platzieren. Ein Beispiel für eine Gewinnsituation ist in der Abbildung [1] dargestellt. Um eine solche oder ähnliche Situation zu erreichen hat jeder Spieler 21 Spielsteine. Wie auch Dame und Schach ist 4-Gewinnt ein endliches Nullsummenspiel mit perfekter Information. Die Bezeichnung „endlich“ gibt hierbei nur an, dass es eine maximale Anzahl von Zügen gibt. Mit dem Ausdruck „Nullsummenspiel“ beschreibt man ein Spiel, welches als Summe der Verluste und Siege beider Gegenspieler Null hat. Bei einem Spiel mit perfekter Information können beide Spieler alle vorherigen Spielzüge nachvollziehen und haben immer einen aktuellen Überblick über das Spielgeschehen. Ein Beispiel für Spiele ohne perfekter Information sind Kartenspiele.

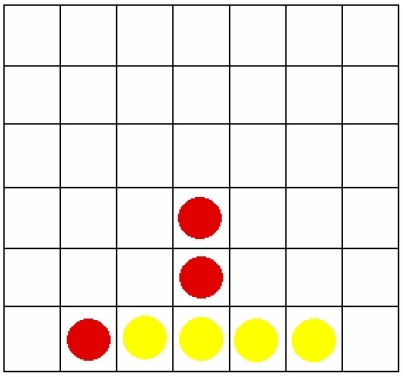


Abbildung 1. Spielfeld mit Gewinnsituation für Gelb, Quelle: o.S. aus http://www.mathematik.uni-muenchen.de/~spielth/artikel/VierGewinnt.pdf Diagramm I1.1

B. Erfassung und Auswertung der Spielsituation

Um die Spielsituation zu erkennen wird das Spielfeld in eine 6x7 Matrix vereinfacht. Eingeworfene Steine und leere Felder bekommen dann einen Wert zugeteilt. Für das Beispiel in Abbildung [1] hieße die Matrix M. Somit hätte diese Matrix die folgende Form:

$$\mathrm{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 2 & 2 & 0 \end{bmatrix}$$

Dabei bekommt Rot die Zahl 1 und Gelb die Zahl 2 zugeteilt, jedes leere Feld hat die Zahl 0. Somit wird bei einer Spielsituation M erkannt, wo welche Spielsteine liegen. Nun lässt sich eine Bewertungsfunktion, ähnlich der Funktion [1] aufstellen, welche die Gewinnchancen für den jeweiligen Spieler berechnet.

$$f(M) = \begin{cases} 0, & \text{Unentschieden} \\ +1, & \text{Spieler gewinnt} \\ -1, & \text{Gegenspieler gewinnt .} \end{cases} \quad (1)$$

10.24352/UB.OVGU-2019-039

Diese ist jedoch hauptsächlich für simplere Spiele, wie Tik-Tak-Toe, geeignet. Für komplexere Spiele wie 4-Gewinnt müsste diese Bewertungsfunktion noch angepasst werden: [2].

$$f(M) = \begin{cases} \infty & \text{Spieler gewinnt} \\ n \in \mathbb{N} > 0 & \text{Vorteil für Spieler} \\ 0 & \text{kein Vorteil für beide Spieler} \\ n \in \mathbb{N} < 0 & \text{Vorteil für Gegenspieler} \\ -\infty & \text{Gegenspieler gewinnt} \end{cases} \quad (2)$$

Dann kann mithilfe eines Algorithmus der nächste Spielzug berechnet werden. Dieser sucht nach Zügen mit dem höchsten Zahlenwert. Jedoch sollen die Züge, die möglichst schnell zum Sieg führen, bevorzugt werden.

C. MiniMax-Algorithmus

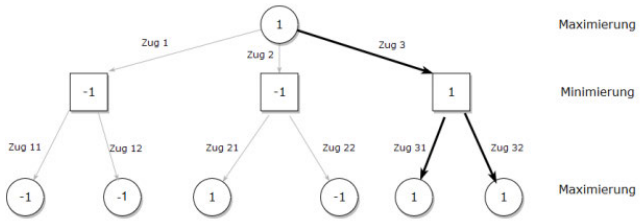


Abbildung 2. Beispielhafter Suchbaum beim MiniMax-Algorithmus, Quelle: http://www.gm.fh-koeln.de/ciopwebpub/Galitzki17.d/Galitz17.pdf S.16 Abbildung 5

Für das Herausfinden der besten Spielstrategie bei 4-Gewinnt eignet sich der MiniMax-Algorithmus. Dieser ist für endliche Zwei-Personen-Nullsummenspiele mit perfekter Information ausgelegt. Die Annahme auf der dieser Algorithmus basiert ist, dass jeder Spieler stets seine Gewinnchance maximieren will. Somit soll er die maximale Gewinnchance für den Spieler berechnen. Hierbei werden Suchbäume verwendet um die möglichen Züge zu simulieren. In Abbildung [2] ist ein Suchbaum dargestellt. Zur Veranschaulichung hat dieser jedoch nur eine Breite von Drei und eine Suchtiefe von 2. Ein Suchbaum für 4-Gewinnt bräuchte eine Breite von Sieben. Der Ausgangszustand des Suchbaumes ist ein Wurzelknoten. Von diesem aus werden alle Zugmöglichkeiten und die daraus folgenden Zugmöglichkeiten berechnet. Züge mit einer höheren Gewinnchance werden mit einem höheren Zahlenwert beschrieben. Der Zahlenwert ergibt sich daraus, dass der Spieler in seinem Zug das Maximum der Werte wählt. Bei den Knoten, wo der Gegenspieler am Zug ist, wird immer das Minimum der Werte genommen. Dies ist dadurch bedingt, dass der Spieler seinen Zug stets frei wählen kann. Der Zug des Gegenspielers kann jedoch nicht kontrolliert werden. Züge mit höheren Werten werden von der künstlichen Intelligenz bevorzugt. Nun wird immer abwechselnd der „Max- und Min-Zug“ simuliert, bis das Spiel zu Ende ist. Jedoch stellt dies bei komplexeren Spielen wie Schach und 4-Gewinnt ein Problem dar. Die Suchbäume sind zu groß und benötigen Unmengen an Speicher und Leistung. Im Beispiel von Abbildung [2] wäre der Zug 3 der vom Algorithmus bevorzugte Spielzug, da er den höchsten Zahlenwert darstellt. Es ist auch gut zu erkennen, dass immer abwechselnd ein „Max- und Min-Zug“ simuliert wird. Somit würde die Gewinnchance der künstlichen Intelligenz gesteigert werden und das Spielerlebnis würde authentischer werden.

III. Hauptteil

A. Spielfelderkennung

Zur Erkennung des Spielfeldes gab es mehrere Ansätze. Zunächst wurde versucht das Spielfeld mittels eines Farbsensors in Matlab umzusetzen. Jedoch erwies sich dies als äußert umständlich bei der mechanischen Umsetzung, weshalb dieser Ansatz verworfen wurde. Der zweite Ansatz war die Verwendung einer Webcam. Hierbei wurden die Ecken des Spielfeldes auf dem Bild festgelegt. Auf diesen beruhend wurden dann die Positionen der restlichen Zellen berechnet und festgelegt. Dies stellte sich als Zuverlässig heraus.

B. Berechnung des besten Spielzuges

Aufgrund der begrenzten Zeit wurde für die Berechnung der Spielzüge ein eher einfacher Algorithmus geschrieben. Dieser erkannte, ob er die eigene Reihe vervollständigen konnte. War dies der Fall, wurde dieser Zug priorisiert. Wenn jedoch der Gegenspieler seine Reihe vervollständigen konnte, versuchte der Algorithmus dies zu blockieren. Trat keiner der beiden Fälle ein, warf die künstliche Intelligenz an einer zufälligen Stelle einen Spielstein ein. Hierzu war es notwendig, dass die Spielsituation erfasst wurde, wie in [II-B] beschrieben.

$$f(M) = \begin{cases} 0, & \text{kein Ergebnis} \\ 1, & \text{Rot gewinnt} \\ 2, & \text{Gelb gewinnt} \\ 3, & \text{Unentschieden} \\ 4, & \text{Betrug} \end{cases} \quad (3)$$

Danach wertete eine Prüfroutine aus, ob ein Gewinner feststeht, es unentschieden ist oder betrogen wurde. Eine solche Routine ist mit Funktion [3] dargestellt. Dabei wurde als Betrug das Einwerfen von Zwei Spielsteinen oder das Einwerfen von keinem Spielstein angesehen. Bei dem Kriterium Null wurde der oben genannte Algorithmus ausgeführt.

C. Mechanische Umsetzung des Roboters

Um der künstlichen Intelligenz das 4-Gewinnt spielen zu ermöglichen, wurde aus dem Lego-Mindstorms-Set ein Roboter gebaut. Dieser verwendet einen Motor als Antrieb, um die einzelnen Zellen des Spielfeldes anzufahren. Dabei fährt er immer parallel zum Spielfeld. Die Distanz zwischen den Zellen wurde ausgemessen und als fester Wert im Programm eingetragen. Hierbei war es wichtig, zu testen ob der erste Stein in der Ausgangsposition die Zelle genau trifft. Beim Einwerfen der Spielsteine gab es anfangs das Problem, dass das Spielfeld zu hoch war. Dieser Höhenunterschied wurde durch den Bau eines Turmes überwunden. Auf diesem wurde das

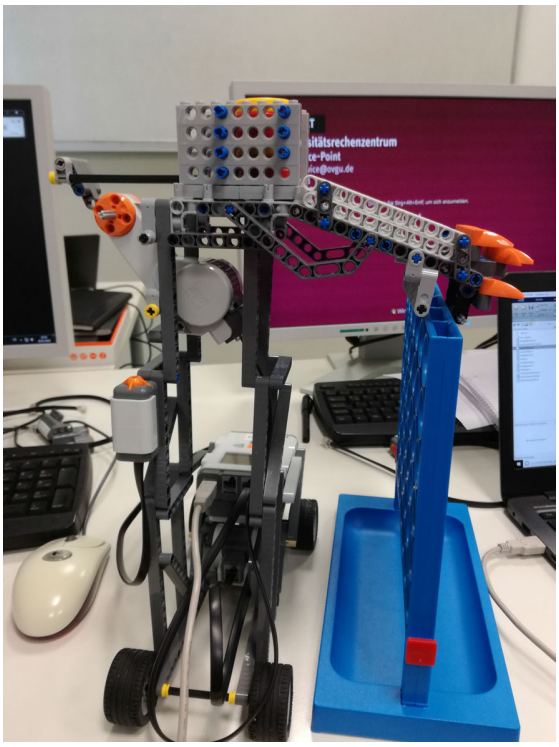


Abbildung 3. 4-Gewinnt-Roboter

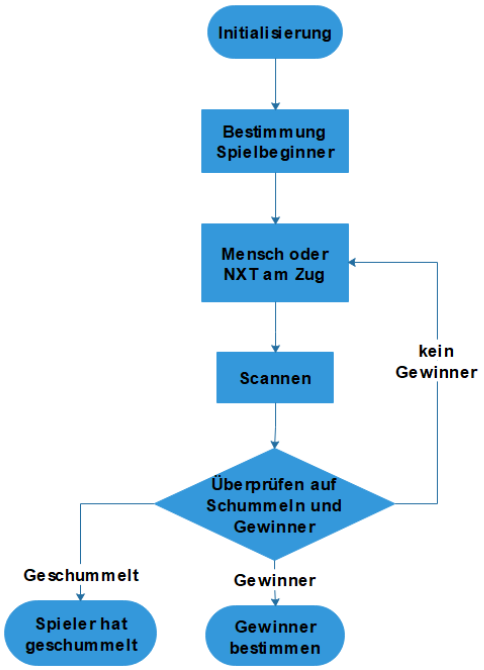


Abbildung 4. Programmablaufplan

Spielsteinlager befestigt. Dieses hatte nur Platz für 6 Spielsteine. Somit musste man während des Spieles immer neue Steine nachlegen. Über eine Rampe gelangten die Spielsteine dann in die einzelnen Zellen. Da die Rampe etwas höher lag als das Spielfeld, wurden am vorderen Teil der Rampe, 2 Leitschienen angebracht. So wurde sichergestellt, dass die Steine im Spielfeld landen. Eine weitere Notwendigkeit war es die Ränder des Spielfeldes etwas anzuschleifen um eine trichterartige Öffnung zu erzeugen. Dies hatte den Effekt, dass die Spielsteine nicht mehr am Rand stecken blieben. Zum Auswerfen der Steine wurde ein einfacher mechanischer Arm konstruiert. Dieser hatte eine festgelegte Schubdistanz. Nach dem Einwerfen fuhr der Roboter zweimal eine kurze Distanz vor und zurück. Dies sollte dabei helfen, Verkantungen von Spielsteinen zu verhindern. Außerdem wurde am Turm ein Tastsensor befestigt. Dieser wurde dazu verwendet dem Roboter zu signalisieren, dass er am Zug ist.

D. Funktionsweise des Programms

Sobald die Webcam ausgerichtet war, kann das Programm über die graphische Benutzeroberfläche initialisiert werden. Der komplette Programmablaufplan ist in Abbildung [4] dargestellt. Zunächst wurde zufällig der Spielbeginner festgelegt. Über diese graphische Benutzeroberfläche konnte auch die von der künstlichen Intelligenz erkannte Matrix überprüft werden. Dies ermöglichte eine Echtzeitüberprüfung. Began der menschliche Spieler, musste dieser zunächst seinen Spielzug ausführen. Wenn er mit diesem fertig war, drückte er den Tastsensor. Dies signalisierte dem Roboter, dass er am Zug ist. Daraufhin scannte dieser das Spielfeld und lies die in [III-B] erwähnte Prüfroutine durchlaufen. Wenn eins der Kriterien: Betrug, Rot gewinnt, Gelb gewinnt oder Unentschieden festgestellt wurde, wurde das Programm beendet. Eine entsprechende Anzeige über den Ausgang des Spieles wurde in der graphischen Benutzeroberfläche ausgegeben. Wenn keins dieser Kriterien wurde das Spiel fortgesetzt. Daraufhin berechnete der Algorithmus dann seinen Zug und führte diesen aus. Dieser Prozess wiederholte sich solange bis eins der spielbeendenden Kriterien erfasst wurde.

IV. Ergebnisdiskussion

Nach Vollendung des Projektseminars Lego Mindstorms war ein Roboter entwickelt worden, der autonom gegen einen menschlichen Gegenspieler 4-Gewinnt spielen konnte. In mehreren Testdurchläufen schaffte der Roboter es gegen den menschlichen Spieler zu gewinnnen. Jedoch versuchte der Mensch auch nicht die Gewinnsituationen des Roboters zu verhindern. Dies zeigte, dass der Roboter sich selber erfolgreich Gewinnsituationen kreieren konnte und auch gegnerische Gewinnsituationen verhinderte. Ein größeres Problem beim Spielen stellte hierbei jedoch die Erkennung durch die Webcam dar. Es musste aufgepasst werden, weder das Spielfeld noch die Webcam zu verschieben, weil sonst das Spiel nicht mehr erkannt wurde. Trat dies ein war eine Rekalibrierung der Webcam erforderlich. Diese war relativ zeitaufwendig und verzögerte das Spiel. Auch musste man bei der Webcam auf die Beleuchtung achten. Die Farberkennung war lichtabhängig, wodurch bei anderer Beleuchtung auch andere Farben erkannt wurden. Dies wurde teilweise umgangen indem die neuen Farbspektren in der regulären Farberkennung ergänzt wurden. Auch stellte das Spielsteinlager und das Einwerfen der Spielsteine ein Problem dar. Beim Einwerfen gab es anfangs das Problem, dass die

Rampe etwas höher gelegen war als das Spielfeld. Dadurch entstand eine Lücke, durch die die Spielsteine heraus fallen konnten. Dies wurde durch die in Abschnitt [III-C] erwähnten Leitschienen behoben. Beim Spielsteinlager wurde zuerst ein geschlossenes Lager gebaut. Bei diesem erwies es sich jedoch als schwierig die Steine hineinzufüllen. Ein weiteres Problem stellte das Verkanten der Steine dar. Deshalb wurde dann ein halboffenes Magazin verwendet. Dies hatte den Vorteil, dass es zum einen besser befüllbar war und zum anderen konnte man sehen, wenn sich Steine verkanteten. Somit konnte man das Problem schnell beheben. Eine weitere Hürde war die Ausrichtung des Roboters. Bei dieser musste darauf geachtet werden, dass der Roboter genau parallel zum Spielfeld fuhr, weil sonst im Verlaufe des Spieles eine zu hohe Abweichung entstand. Als Folge trafen die Spielsteine dann nicht ihre Zellen. Im schlimmsten Fall wurde sogar das Spielfeld verschoben.

V. Zusammenfassung und Fazit

Während des Projektseminars Lego Mindstorms wurde erfolgreich ein Roboter entwickelt, der autonom gegen einen menschlichen Gegenspieler 4-Gewinnt spielen konnte. Jedoch gibt es noch einige Verbesserungsmöglichkeiten. Eine Idee war es ein Area of Interest für die Webcam festzulegen. Dies würde die Kalibrierung erleichtern und den Zeitaufwand dafür deutlich verringern. Eine weitere Verbesserungsidee war es, den Roboter in einer festen Schiene fahren zu lassen. Dies würde den Vorteil bieten, dass man diesen nicht vor jedem Spiel neu ausrichten müsste. Auch könnten so Fehler und Ungenauigkeiten besser vermieden werden, da so Fehler vom Menschen ausgeschlossen wären. Das in [IV] beschriebene Spielsteinlager könnte auch noch ausgebessert werden, denn hier traten immernoch Verkantungen der Spielsteine auf. Hierzu wäre ein Lager, welches genauso breit ist wie die Spielsteine selbst, angebracht. Dies mit Lego zu konstruieren erwies sich jedoch als recht anspruchsvoll. Auch könnte man die Kapazität des Lagers noch auf 21 Spielsteine erhöhen, sodass man während des Spielens keine Steine mehr nachlegen müsste. So könnte man dann ein ganzes Spiel am Stück durchspielen. Eine andere Verbesserung wäre ein besserer Spielalgorithmus. Dieser gestaltete sich bei diesem 4-Gewinnt-Roboter relativ simpel. Mit einer richtigen Implementierung des MiniMax-Algorithmus könnte der Roboter ein besseres Spielverhalten aufweisen. Somit wäre es auch möglich die Suchtiefe des Algorithmus variabel einzustellen. Dies wäre eine Variante der Implementierung von verschiedenen Schwierigkeitsgraden. über die graphische Benutzeroberfläche könnte man eine solche Auswahlmöglichkeit realisieren.

Literatur- und Bildverzeichnis

[1] Wikipedia, Nullsummenspiel, https://de.wikipedia.org/wiki/Nullsummenspiel (Abruf: 15.03.2019)
[2] Wikipedia, Minimax-Algorithmus, https://de.wikipedia.org/wiki/Minimax-Algorithmus (Abruf: 15.03.2019)
[3] Christian Kanzow und Alexandra Schwartz *Spieltheorie: Theorie und Verfahren zur Lösung von Nash- und verallgemeinerten Nash-Gleichgewichtsproblemen, S.5-7.* 02.Oktober 2018.
[4] Hrsg. v. Görz, Günther / Schneeberger, Josef / Schmid, Ute *Handbuch der Künstlichen Intelligenz, S.624-627.* Oktober 2013.

Lego Mindstorms Vier-Gewinnt-Roboter

Robert Göpfert, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In diesem Paper wird die Entwicklung eines Roboters mit LEGO Mindstorms beschrieben, welcher autonom Vier Gewinnt gegen einen menschlichen Gegner spielen kann. Dafür wurde aufbauend auf der mathematischen Spieltheorie ein einfacher Spielalgorithmus implementiert. Die Spielfelderkennung wurde weiterhin durch die Verwendung einer Webcam automatisiert.**

***Schlagwörter*—Künstliche Intelligenz, Minimax-Algorithmus, Projektseminar, Robotik, Spieltheorie, Vier Gewinnt**

I. Einleitung

EIN stark wachsendes Teilgebiet der Informatik ist die Entwicklung künstlicher Intelligenz, um Mustererkennung und -vorhersage und logische Schlüsse basierend auf einer Wissensdatenbank zu automatisieren. Anwendungsgebiete sind die Erkennung von Krankheitsbildern in der Medizin, Bild- und Gesichtserkennung in der Kriminalistik und die Nachbildung menschlichen Verhaltens von Computergegnern in Videospielen. Zu Computerspielen zählen auch Brettspiele, wie Solitär, Schach und das in diesem Paper behandelte Vier Gewinnt. Bezüglich des Projektseminars wurde dafür mit LEGO Mindstorms ein Roboter entwickelt, der ein physisches Spielfeld mit einer Webcam erkennt und basierend auf diesen Informationen den nächsten Zug berechnet und an entsprechender Stelle seinen Spielstein einwirft.

II. Vorbetrachtungen

Um eine künstliche Intelligenz für ein Spiel zu entwickeln, werden Ansätze der Spieltheorie benutzt. Dazu wird das Spiel zunächst näher beschrieben.

A. Vier Gewinnt

Vier Gewinnt wird zu zweit auf einem senkrecht stehenden Spielbrett gespielt. Ziel ist es, als erster 4 eigene Spielsteine in eine Reihe horizontal, vertikal oder diagonal in eine Reihe zu bringen. Dabei handelt es sich bei Vier Gewinnt um ein endliches Nullsummenspiel mit perfekter Information. Endlich heißt, dass nach einer endlichen Anzahl an Zügen das Spiel endet. Ein Nullsummenspiel ist ein Spiel, bei dem die Summe der Verluste und Gewinner beider Spieler zusammengenommen gleich null ergeben. [1] Bei einem Spiel mit perfekter Information lässt sich der Spielstand jederzeit von allen Spielteilnehmern einsehen. In Abbildung 1 ist eine Spielsituation dargestellt, in der der Spieler mit den gelben Steinen gewinnt.

DOI: 10.24352/UB.OVGU-2019-040

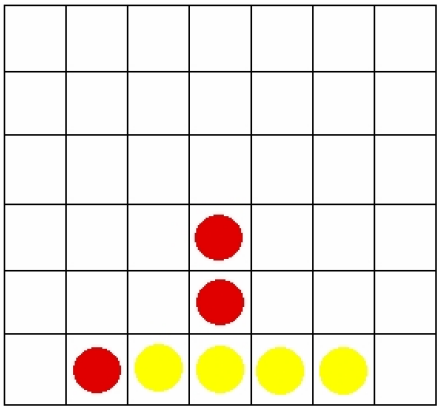


Abbildung 1. Beispielhafte Spielsituation mit Gewinn für Gelb, Quelle: urlhttp://www.mathematik.uni-muenchen.de/~spielth/artikel/VierGewinnt.pdf

B. Bewertungsfunktion

Um einen Spielalgorithmus für Vier Gewinnt zu implementieren, ist eine Bewertungsfunktion notwendig, mit der es möglich ist, eine Spielsituation nach den Gewinnchancen für die jeweiligen Spieler zu evaluieren. Eine ideale Bewertungsfunktion stellt die Funktion 1 dar.

$$f(M) = \begin{cases} 1 & \text{Spieler gewinnt} \\ 0 & \text{Unentschieden} \\ -1 & \text{Gegenspieler gewinnt} \end{cases} \quad (1)$$

Die Verwendung einer idealen Bewertungsfunktion ist allerdings nur bei einfachen Spielen wie Tic-Tac-Toe angebracht. Für Spiele wie Schach oder Vier Gewinnt existieren zu viele verschiedene Kombinationen, um all diese in einer annehmbaren Zeit zu bearbeiten. Dafür wird die Bewertungsfunktion wie in Funktion 2 angepasst.

$$f(M) = \begin{cases} \infty & \text{Spieler gewinnt} \\ n \in \mathbb{N} > 0 & \text{Vorteil für Spieler} \\ 0 & \text{kein Vorteil für beide Spieler} \\ n \in \mathbb{N} < 0 & \text{Vorteil für Gegenspieler} \\ -\infty & \text{Gegenspieler gewinnt} \end{cases} \quad (2)$$

Dabei stehen negative Werte für einen Vorteil für den Gegenspieler und positive für den Spieler. Je größer ihr Betrag ist, desto größer ist der Vorteil für den jeweiligen Spieler. ∞ steht für den Gewinn des jeweiligen Spielers.

Eine Bewertungsfunktion speziell für Vier Gewinnt könnte so gestaltet werden, dass höhere Werte zurückgegeben werden, je vollständiger eine Reihe ist. Außerdem könnten Spielsituationen

mit Zwickmühlen bevorzugt werden, indem sie höher als andere Vervollständigungen bewertet werden. Zwickmühlen sind bei Vier Gewinnt Situationen, die zwei unvollständige sich überkreuzende Reihen enthalten, die nicht gleichzeitig blockiert werden können.

C. Minimax-Algorithmus

Der Minimax-Algorithmus wird eingesetzt, um die optimale Strategie für ein endliches Nullsummenspiel mit perfekter Information, wie Vier Gewinnt, zu ermitteln. Allgemein gesagt werden ausgehend von der aktuellen Spielsituation alle möglichen zukünftigen Szenarien bestimmt und nach der Gewinnchance für den Spieler bewertet. Danach wird eine Strategie ausgearbeitet, um die Spielsituation mit der höchsten Gewinnchance zu sichern.

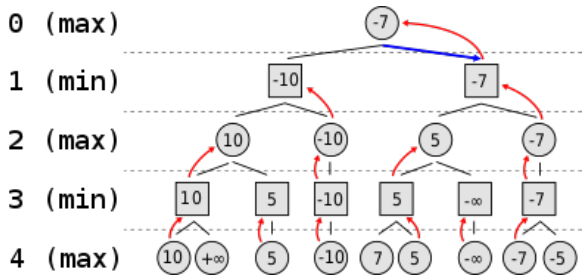


Abbildung 2. Suchbaum eines Minimax-Algorithmus, Quelle: https://upload.wikimedia.org/wikipedia/commons/6/6f/Minimax.svg

Um alle künftigen Spielsituationen abzudecken, wird rekursiv ein Suchbaum aufgebaut. In Abbildung 2 ist ein Suchbaum mit der Tiefe 4 dargestellt. Die aktuelle Spielsituation bildet den Wurzelknoten des Baumes. Alle davon abgehenden Kindknoten enthalten unterschiedliche Spielsituation nach einem Zug des Gegners. Ist die maximale vorgegebene Tiefe erreicht, wird auf die Spielsituationen der untersten Blätter eine Bewertungsfunktion wie in Abschnitt II-B beschrieben angewendet.

Anschließend wird in den Ebenen, in denen der Gegenspieler am Zug ist, die kleinsten Werte aus den Kindknoten ausgewählt und den Elternknoten zugewiesen (es wird minimiert). In den Ebenen, in denen der Spieler am Zug ist, werden die größten Werte aus den Kindknoten ausgewählt und den Elternknoten zugewiesen (es wird maximiert). Ist der Wurzelknoten erreicht, wird aus dessen Kindknoten der größte Werte ausgewählt. Dieser Knoten ist dann auch der Zug der gespielt wird. [2], [4]

III. Hauptteil

A. Erkennung des Spielfeldes

Der erste Ansatz zur Spielfelderkennung bestand darin, mit einem Farbsensor die einzelnen Zellen zu scannen und anhand der Farbwerte und der gescannten Position eine Matrix aufzubauen. Um Zeit einzusparen, sollten dabei die Zellen, die bereits gescannt wurden, übersprungen werden. Diese Variante der Spielfelderkennung war jedoch trotz der Optimierung zu zeitintensiv, da im ungünstigsten Fall sieben Zellen pro Scan untersucht werden müssen.

Als effizientere Variante stellte sich die Verwendung einer Webcam heraus, da mit dieser das gesamte Spielfeld auf einmal eingescannt werden konnte. Probleme bereitete allerdings die Erkennung der Position und Farbe der einzelnen Zellen. Anfangs wurde die in Matlab integrierte Funktion „imfindcircles“ verwendet, welche kreisförmige Objekte in einem Bild erkennen kann.

Da diese jedoch unzuverlässig arbeitete, wurden die Positionen der Zellen vorher festgelegt, sodass nur noch die Ausrichtung der Kamera eine Fehlerquelle darstellt. Anschließend wurden die Farbwerte an den entsprechenden Positionen ausgelesen, einer Zahl zugeordnet und in die Matrix eingetragen.

B. Ermittlung des besten Zuges

Nachdem das Spielfeld in eine Matrix übertragen wurde, bewertete eine Prüfroutine die aktuelle Spielsituation. Eine solche ist mit Funktion 3 dargestellt. Sie diente dazu, das Spiel zu beenden, wenn ein Gewinner feststand, es unentschieden stand oder geschummelt wurde. Der Algorithmus erkennt das mehrfache Einwerfen während eines Zuges und das Entfernen von Spielsteinen sowie das Auslassen eines Einwurfes als Schummeln und beendet bei entsprechenden Aktionen das Spiel.

$$f(M) = \begin{cases} 0 & \text{, noch kein Ergebnis} \\ 1 & \text{, rot hat gewonnen} \\ 2 & \text{, gelb hat gewonnen} \\ 3 & \text{, unentschieden} \\ 4 & \text{, es wurde geschummelt} \end{cases} \tag{3}$$

Da sich die Arbeit am Bau des Roboters nur auf zwei Wochen beschränkte, wurde ein sehr einfacher Algorithmus geschrieben. Dieser priorisiert die Vervollständigung einer eigenen Reihe vor der Blockierung einer gegnerischen Reihe. Kann weder vervollständigt noch blockiert werden, wird an zufälliger Stelle ein Stein eingeworfen.

C. Anfahren der Position und Einwurf des Steins

Die Grundidee des Aufbaus bestand darin, die Spielsteine aus einer erhöhten Position in das Spielbrett einzuwerfen. Daher wurde ein Turm konstruiert, der sich auf vier Rädern parallel zum Spielfeld bewegen kann, wie in Abbildung 3 zu sehen ist. An der Turmspitze schob ein mechanischer Arm die Steine aus einem Vertikalmagazin auf eine geneigte Ebene, auf der sie dann in die Öffnungen des Spielbrettes rutschten. Um einen konstanten Abstand zum Brett sicherzustellen, wurden an das Ende der Rutsche Leitschienen angebracht. Seitlich am Turm befand sich weiterhin ein Taster, mit dem der menschliche Gegenspieler signalisieren konnte, dass er seinen Zug beendet hat.

Der Roboter wurde anfangs so ausgerichtet, dass das Ende der Rutsche über dem Brett positioniert war und ein aus dem Magazin geschobener Stein in die erste Spalte fallen würde. Von dort aus wurden die restlichen Spalten angefahren.

Nachdem der Roboter die gewünschte Position angefahren hatte, erfolgte der Einwurf eines Steines und der Roboter fuhr zweimal eine kleine Strecke nach links und rechts, um eine mögliche Verkantung des Steines mit dem Spielbrett zu lösen.

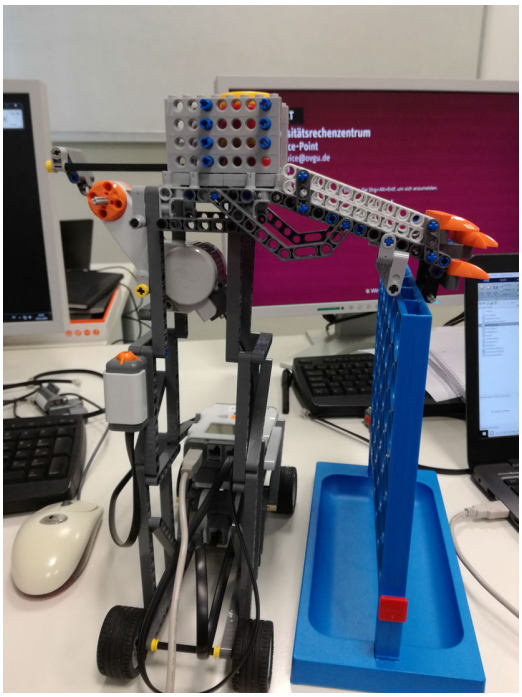


Abbildung 3. Am Spielbrett ausgerichteter und einsatzbereiter Vier-Gewinnt-Roboter

D. Gesamtes Programm

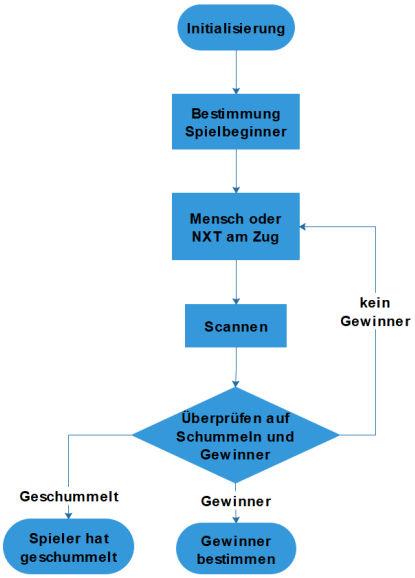


Abbildung 4. Gesamter Programmablaufplan des Vier-Gewinnt-Roboters

In Abbildung 4 ist der Programmablaufplan des gesamten Programms abgebildet. Nachdem die Kamera ausgerichtet wurde, konnte das Spiel über die graphische Benutzeroberfläche gestartet werden. Über diese wurde ausgegeben, welcher Spieler beginnt. Begann der menschliche Spieler, führte dieser seinen Zug durch und betätigte anschließend den Taster am Roboter. Daraufhin wurde das Spielfeld gescannt und überprüft, ob geschummelt wurde und ob ein Spieler das Spiel gewonnen hat oder es unentschieden ist. Traf eines der Kriterien zu, wurde das Spiel beendet und eine entsprechende Meldung über die Benutzeroberfläche ausgegeben. Wenn nicht, wurde das Spiel fortgesetzt, indem der Roboter seinen Zug ausführte, das Spielfeld scannt und auf die Betätigung des Tasters wartet.

IV. Ergebnisdiskussion

Während des Projektseminars wurde ein Roboter entwickelt, welcher autonom gegen einen menschlichen Spieler Vier Gewinnt spielen konnte. Aufgrund der zeitlichen Begrenzung wurde nur ein einfacher Spielalgorithmus verwendet, gegen den der menschliche Spieler relativ einfach gewinnen konnte. Dies bestätigten auch mehrere Testspiele, in denen der Roboter nur gewann, wenn die Testperson ihm die Gewinnchancen nicht verbaute. Weiterhin beendet der Algorithmus das Spiel automatisch, wenn ein Gewinner feststeht, es unentschieden steht oder geschummelt wurde.

Ein Problem stellte anfangs der Einwurf der Spielsteine dar. Trotz der relativ präzisen Rutschenkonstruktion kam es oft vor, dass sich die Spielsteine mit dem Rand des Spielbrettes verkanteten. Dies konnte gelöst werden, indem der innere Rand der Öffnungen trichterartig angeschliffen wurde, sodass die Steine direkt in die Öffnung hineinfielen. Weiterhin kam es häufig zum gleichzeitigen Einwurf mehrerer Spielsteine, wenn sich nur noch zwei oder drei Steine im Magazin befanden. Dieses Problem konnte nicht gelöst werden, man konnte nur dafür sorgen, dass immer genug Steine nachgelegt wurden. Auch die Erkennung der Spielsituation mittels Webcam verlief nicht ohne Probleme. Da anfangs die relativ ungenaue Matlab-Funktion „imfindcircles“ verwendet wurde, musste dabei auf eine optimale Belichtung des Spielfeldes geachtet werden. Auch mit der finalen Lösung durch die Festlegung der Zellenpositionen musste auf die exakte Ausrichtung der Webcam und halbwegs gute Lichtverhältnisse geachtet werden. Dies sorgte zum einen dafür, dass der anfängliche Kalibrierungsvorgang durch mehrere Versuche länger dauerte, zum anderen durften das Spielfeld und die Webcam während des Spielvorgangs nicht gegeneinander verschoben werden.

V. Zusammenfassung und Fazit

Während des Projektseminars gelang es, einen Roboter zu bauen, der in der Lage war, Vier Gewinnt gegen einen menschlichen Gegner zu spielen. Zukünftige Verbesserungen finden sich vor allem in der Programmierung. Zum einen kann die Spielfelderkennung mittels Webcam verbessert werden, indem farbige Markierungen an die Ecken des Spielfeldes angebracht werden. Aus den Positionen dieser Marker können dann die Positionen der Zellenmittelpunkte bestimmt werden. Somit wäre die Spielfelderkennung unabhängig von der Positionierung und vom Ausrichtungswinkel der Webcam und die zeitintensive Kalibrierung zu Beginn eines Spiels würde entfallen. Zum anderen kann der Spielalgorithmus zum Minimax-Algorithmus abgeändert werden, sodass der Roboter intelligentere Züge spielen kann. Weiterhin kann für die Farbbestimmung einer Zelle ein Mittelwert aus den Farbwerten mehrerer Pixel gebildet werden. Somit würde man aussagekräftigere Werte

erhalten und die Spielfelderkennung etwas unabhängiger von der Umgebungshelligkeit machen.

Außerdem kann das Magazin und der Einwurfmechanismus verbessert werden, sodass es bei wenigen Spielsteinen nicht mehr zu Verkantungen und mehrfachem Einwurf pro Zug kommen kann.

LITERATUR

[1] Wikipedia, Nullsummenspiel, https://de.wikipedia.org/wiki/Nullsummenspiel (Abruf: 19.03.2019)
[2] Wikipedia, Minimax-Algorithmus, https://de.wikipedia.org/wiki/Minimax-Algorithmus (Abruf: 19.03.2019)
[3] Christian Kanzow und Alexandra Schwartz *Spieltheorie: Theorie und Verfahren zur Lösung von Nash- und verallgemeinerten Nash-Gleichgewichtsproblemen, S.5-7*. 02.Oktober 2018.
[4] Hrsg. v. Görz, Günther / Schneeberger, Josef / Schmid, Ute *Handbuch der Künstlichen Intelligenz, S.624-627*. Oktober 2013.

Unbekannte Objekte Transporter

Laith Aladwan, MTK
Otto-von-Guericke-Universität Magdeburg

I. Abstract

Das 21. Jahrhundert ist ein Jahrhundert der Robotik. Roboter haben das Potenzial, die Lücke zwischen der kybernetischen und der physischen Welt zu schließen. Die Robotik wird in der Gesellschaft aufgrund ihres Einflusses in allen Bereichen des Lebens, einschließlich Medizin und Gesundheitswesen, Gebäudetechnik, Fertigung, Lebensmittelproduktion, Logistik und Transport, eine immer wichtigere Rolle spielen.

II. Einleitung

Heutzutage können Roboter bestimmte gefährliche Funktionen eines Menschen ausführen, die besonders wichtig sein konnte, wie zum Beispiel bei der Räumung einer Bombe. Bombenräumroboter werden seit über 40 Jahren eingesetzt, um Kampfmittel sicher zu deaktivieren, wo sie hunderte, wenn nicht sogar tausende Male eingesetzt wurden. Einer der ersten Bombenentschärfungsroboter war die Schubkarre Mark 1. 1972 kam Oberstleutnant Peter Miller von der Britischen Armee auf die Idee, das Chassis einer elektrisch angetriebenen Schubkarre zu benutzen, um verdächtige Geräte wie Autobomben zu schleppen, damit sie sicher gezündet werden können, ohne jemanden zu verletzen. Durch den Bombenräumroboter können anonyme Objekte von sicherer Entfernung aufgehoben werden und zu sicheren Platz gebracht werden, also der Roboter hilft dabei, das Leben der Menschen vor Gefahr zu schützen. Der Prototyp der Schubkarre erwies sich jedoch als schwierig zu manövrieren, so dass die Militärfahrzeuge und das Technikum in Chertsey mit einbezogen wurden, um die Steuerungs- und Verfolgungssysteme zu verbessern. [1]

III. Vorbetrachtung

Die ursprünglichen Bombenräumroboter wurden von einer Reihe von Seilen gesteuert. Im Zuge des technologischen Fortschritts wurde jedoch ein Telekommunikationskabel verwendet, um Befehle an die elektrischen Systeme des Roboters zu übertragen. Es bestand jedoch auch die Gefahr, dass sich das Kabel an Gegenständen verfängt - ähnlich wie bei einem Gartenschlauch oder einem Staubsauger.

Heute wird die Mehrheit des Bombenräumroboters durch drahtlose Kommunikation gesteuert. Dies erhöht zwar die Reichweite erheblich.

Wir haben einen Roboter gebaut, der nach dem gleichen Prinzip funktioniert, aber einfacher. Also haben wir ihn so gebaut, dass er die Farbe eines Balles scannt und dann, basierend auf der Farbe des Balles, ihn an einen bestimmten Ort bringt.

IV. Aufbau

Für den Bau dieses Projekts wurde ein Automobil gebaut, seine Fahrwerkekonstruktion wurde mit Legobausteinen gebaut, das mit zwei Motoren durch Ketten gesteuert werden. Dahinter befindet sich ein Greifer, der mit einem Motor und zwei Haken gemacht wurde. Über dem Greifer befindet sich ein Farbsensor, der die Farbe des Balles scannt und dann an die NXT (das Gehirn des Roboters) sendet. Aufgrund der hohen Helligkeit konnte der Farbsensor die Farbe manchmal nicht gut erkennen. Deshalb haben wir eine Verkleidung über dem Farbsensor angebracht, um zu verhindern, dass das übermäßige Licht durchgelassen wird. Vorne ist der Ultraschall-Sensor, der den Abstand zwischen dem Automobil und den Anonymen Objekten misst, in diesem Fall den Kugeln, wir haben den Sensor so gebaut, dass er den Fahrzeugmechanismus nicht stört.

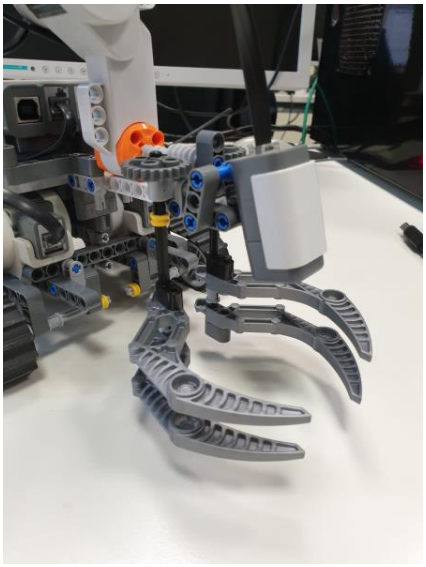


Abbildung 1: Greifer und Farbesensor

DOI: 10.24352/UB.OVGU-2019-041

V. AUFGETRETENE PROBLEME

• *Ungenauigkeit der Motoren*

Die Motoren zählen manchmal den Drehwinkel nicht genau, so dass, wenn das Automobil vorwärts fährt, ein Motor einen anderen Drehwinkel als der andere Motor hat, und dann wird dieser Drehwinkel im nächsten Schritt verwendet, und das führt zu einer geringeren Genauigkeit durch mehrfache Ausführung des Programms. Das führt auch dazu, dass der Roboter die Bälle nicht mehr findet und sie nicht greifen kann.

Abbildung 2: Motor

• *Ungenauigkeit des Farbesensors*

Wie bereits gesagt, scannt der Farbsensor die Farbe der Bälle falsch wegen der hohen Helligkeit, und deshalb ist die Decke gebaut, aber auch hier liest der Farbsensor immer noch die falschen Werte, und das führt dazu, dass die Bälle falsch platziert werden. Dieses Problem kann nur geändert werden, wenn eine andere Technologie oder einen neuen Sensor verwendet werden.

• *Unterschiedliche Oberflächen*

Das Material der Oberfläche spielt eine wichtige Rolle bei der Steuerung des Automobils, da es sich um ein Kettenfahrzeug handelt. Es besteht ein proportionaler Zusammenhang zwischen der Reibung zwischen der Oberfläche und den Ketten und der Geschwindigkeit, der Leistung des Fahrzeugs und dem Drehwinkel. Je rauer die Oberfläche ist, desto langsamer fährt der Roboter und desto mehr Leistung benötigt er, was zu einer ungenauen Lenkung führt. Der Roboter ist programmiert, um über eine homogene, glatte Oberfläche zu fahren.

Abbildung 3: Farbesensor und die Decke

VI. VERBESSERUNGSMÖGLICHKEITEN

Ketten

Der Bewegungsmechanismus kann gewechselt werden, anstelle von Ketten, Räder können verwendet werden, aber dann werden 4 Motoren benötigt, was mit einem NXT nicht möglich war, da er nur 3 Ports hat.

Kamera

Eine Kamera kann hinzugefügt werden, um die Spur zu sehen und die Objekte zu sehen, was dazu führen kann, dass der Ultraschallsensor und der Farbsensor aufgegeben werden, was jedoch die Genauigkeit der Arbeit beeinträchtigen kann.

Greifer

Der Greifer ist am Roboter befestigt. Er kann durch Hinzufügen von Motoren verbessert werden, um ihn sich bewegen zu lassen und die Objekte aufzunehmen und zu .transportieren.

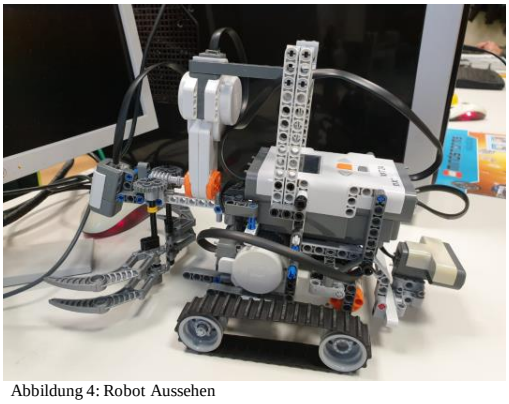


Abbildung 4: Robot Aussehen

VII. ANWENDUNGSMÖGLICHKEITEN

• *Zuhause*

Der Roboter kann verwendet werden, um den Raum von den Kinderspielzeugen oder anderem Zeug zu räumen.

• Militärisch

Der Roboter kann zum Entschärfen von Bomben oder zum Aufspüren von Minen eingesetzt werden, um Verletzungen zu vermeiden.

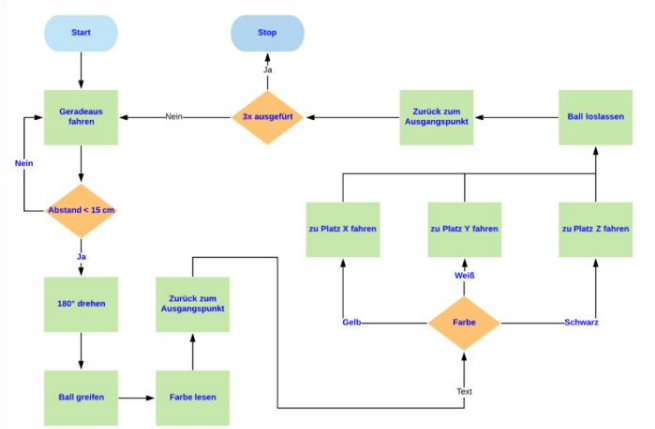


Abbildung 5: PAP

VIII. ZUSAMMENFASSUNG

Roboter, die Bomben entschärfen, sind kompliziert und haben mehr Eigenschaften als die bereits zusammengebauten, eine andere Verwendung für diese Roboter ist die Disassemblierung von unbekannten Objekten wie Minen. Und das wird in vielen Situationen sehr nützlich, diese Art von Robotern kann viele Leben retten. Der Lego-Roboter kann dies nicht tun, weil er aus schwachem Material gebaut wurde, und es wäre gefährlich, einen Roboter zu bauen, der ohne Erfahrung mit Bomben eingesetzt werden kann.
Aber das Endergebnis zeigt, dass diese Roboter, die Bomben entschärfen, durch den Einsatz verschiedener Materialien noch verbessert werden können und von der Mehrheit der Menschen für Notfälle eingesetzt werden können und das in schwierigen Situationen helfen würde.
Das Ziel dieses Projektes wurde trotz der ungelösten Probleme bereits erreicht, mit einigen zusätzlichen Informationen über die Schwierigkeiten und Herausforderungen beim Bau eines solchen Roboters.
Eine weitere Schlussfolgerung ist, dass man mit einfachen Werkzeugen wie Legosteinen etwas Kompliziertes tun kann, um einen nützlichen Roboter zu bauen, obwohl sie nicht immer genau sind, aber man kann mit ihnen Zeit und Geld sparen, indem man den Roboter mit Legos Mindstorm programmiert.

LITERATURVERZEICHNIS.

[1] Peter Ray Allison. (2016, July). Titel: What does a bomb disposal robot actually do? Available: http://www.bbc.com/future/story/20160714-what-does-a-bomb-disposal-robot-actually-do*Basic format for computer programs and electronic documents (when available online):* ISO recommends that capitalization follow the accepted practice for the language or script in which the information is given.

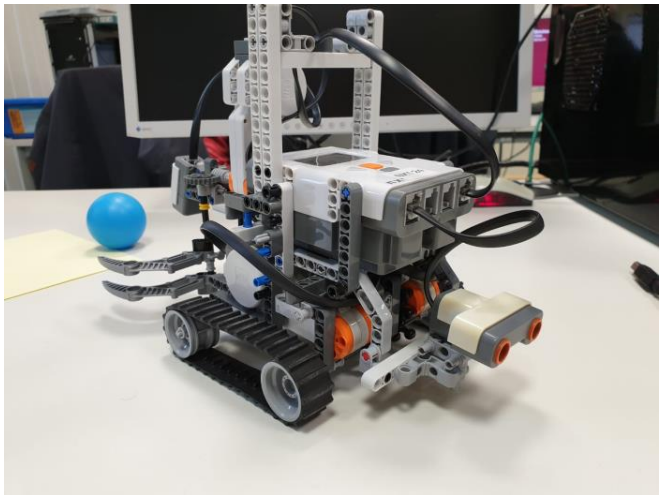


Abbildung 6: vordere Seite des Roboters

Unbekannte Objekte Transporter

MAHMOUD ABDEL GHANI M MQBOUL , ELEKTROTECHNIK/INFORMATIONSTECHNIK
Otto-von-Guericke-Universität Magdeburg

I Einleitung

Roboter sind Maschinen, die sich selbstständig bewegen und verschiedene Tätigkeiten erledigen können. Das unterscheidet Roboter von ferngesteuerten Maschinen ,die Befehle von Menschen brauchen. Diese Roboter werden vielfältig eingesetzt.[1] In der Industrie sind die Roboter häufig zu sehen, da sie Traglasten bewältigen können und bessere Prozessgeschwindigkeiten und Zykluszeiten als die Menschen haben.[2]

Außerdem können die Roboter auch Zuhause eingesetzt werden, wie zum Beispiel Sauger Roboter, die komplett eigenständig die Reinigung von Böden aller Arten übernehmen können.

Die Roboter können das Leben vereinfachen und die Produktivität beschleunigen.

Heutzutage können Roboter auch gefährliche Aufgaben eines Menschen ausführen, die besonders wichtig sein können, um Menschenleben zu beschützen, wie zum Beispiel beim Entschärfen einer Bombe.

Mit Hilfe von Robotern kann der Bombenentschärfer unbekannte Objekte aus sicherer Entfernung aufheben und zu einem sicheren Ort bringen, und so dabei helfen, das Leben vor Gefahr zu schützen.[3]

Das Problem eines solchen Roboters ist, dass der Roboter nur von einer erfahrenen Person ferngesteuert werden kann.

Aus diesem Grund könnte es bei Notfällen noch nicht hilfreich sein.

Das Ziel dieses Projekts war, einen Roboter zu erbauen , der autonom Objekte finden und sie nach ihre Farbe sortieren kann, ohne dass der Roboter ferngesteuert werden muss.

Schnellere Reaktionszeiten und effektivere Resultate zählen zu den Vorteilen eines solchen Roboter, da für die Menschen nicht viele Zeit bleibt und die Spezialisten lange brauchen, um kommen zu können.

Dieses Projekt wurde im Zeitraum vom 11.Februar bis 22.Februar 2019 in LEGO Mindstorms aufgebaut, in Zusammenarbeit von Laith Aladwan und Mahmoud Mqboul.

II Aufbau

Für den Aufbau dieses Projekts wurden drei Motoren, zwei Sensoren (Farb- und Ultraschall-), drei farbige Bälle und Legobaustein benötigt,

Zwei Motoren wurden für das Bewegen und Steuern des Roboters genutzt, der Roboter wurde wie ein Kettenfahrzeug mit Legosbausteine strukturiert.

Das Kettenfahrzeugs verhält sich besser mit Hindernissen als gewöhnliches Fahrzeug mit vier normalen Reifen.

Ein Greifer, der mit einem Motor und zwei Haken gebaut wurde, wurde an der Vorderseite aufgebaut.

Eine Abdeckung wurde über dem Greifer mit einem Platz in der Mitte für den Farbsensor angelegt. Solche Konstruktion hilft dabei, dass beim Lesen der Farben die

DOI: 10.24352/UB.OVGU-2019-042

Objekte (Bälle) dunkler als der Raum bleiben. Dies führt zu genaueren Ergebnissen.

Auf der Rückseite des Fahrzeugs wurde der Ultraschallsensor angelegt, damit er den Abstand ohne Behinderungen messen konnte.

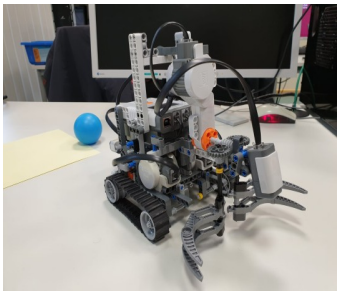


Abb.1 Vorderseite

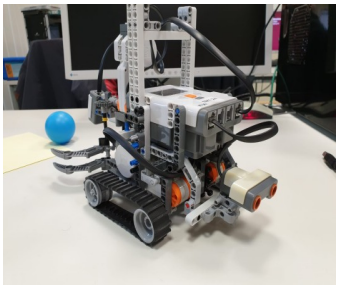


Abb..2 Hinterseite

III Programmablauf

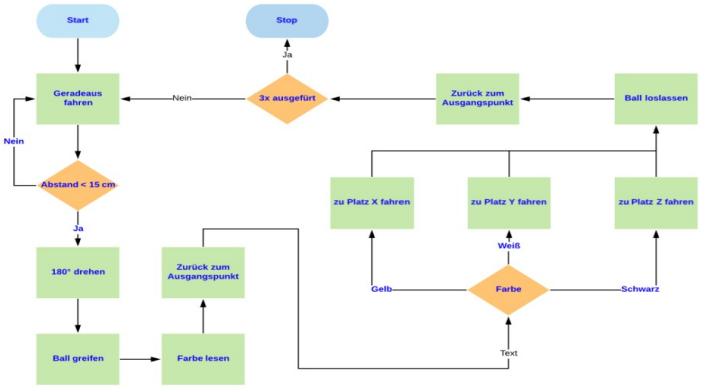


Abb. 3 PAP

Der Roboter soll beim Starten des Programms nach vorne fahren, bis der Abstand zwischen ihm und dem Ball weniger als 15 cm beträgt. Danach dreht er sich um 180°,greift den Ball und liest die Farbe des Balles ein.

Nach dem Greifen und Lesen der Farbe des Balles fährt der Roboter zurück zum Ausgangspunkt.

Das Ergebnis vom Lesen der Farbe wurde bereits im Programm gespeichert, in Abhängigkeit davon fährt der Roboter mit dem Ball zu einem für diese Farbe vordefinierten Platz.

Der Ball wurde dann daraufhin abgelegt,am Ende fährt der Roboter wieder zum Ausgangspunkt.

Sobald das Programm dreimal ausgeführt wurde, wird es automatisch beendet .

IV Aufgetretene Probleme + Lösungsvorschläge

- **Ungenauigkeit der Motoren**

Wenn das Fahrzeug nach vorne fahren oder sich um einen bestimmten Winkle drehen sollte,traten immer Probleme bei der mehreren Ausführung des Programms auf, weil die Motoren sich um 10° mehr oder weniger drehten. Das heißt, dass es schwierig wurde die Bälle wieder zu finden und sie zu greifen.

- **Ungenauigkeit des Farbsensors**

Der Farbsensor ist in der Lage, die Farbe der Bälle zu erkennen.Bei erhöhter Helligkeit wurden die Farben trotz der Abdeckung falsch gelesen, dies kann dazu führen, dass die Bälle zum falschen Platz gebracht werden.

- **Unterschiedliche Oberflächen**

Da der Roboter ein kleines Kettenfahrzeug ist und die Motoren nicht stark genug sind, spielt das Material der Oberflächen eine große Rolle bei der Geschwindigkeit und Drehzahl der Motoren des Fahrzeuges.

Deswegen konnte der Roboter nur auf einer homogenen Oberfläche benutzt werden.

Der Roboter kann mit besseren Motoren und Sensoren seine Aufgaben perfektionieren. Die Motoren sollten stärker und exakter beim Fahren, Drehen und Greifen sein.

Die Sensoren sollten unter allen Umständen die gleichen

Messungsergebnisse liefern, dann könnte dieser Roboter zum Einsatz kommen.

V Allgemeine Optimierungsvorschläge

- **Kamera**

Eine Kamera kann beim Erkennen der Objekte helfen, sie könnte sogar Objekten folgen und mit einer Kamera könnte man auf Ultraschall- und Farbsensor verzichten. Aber die Ergebnisse würden nicht so genau wie bei den Originalsensoren sein.

- **Greifer**

Der Greifer des Roboters ist festgelegt,aber er dadurch verbessert werden,dass ein zusätzlicher Motor verbaut wird. Der Greifer kann sich mit diesem Motor vertikal bewegen.Das führt zu einer besseren Behandlung von kleineren und größeren Objekten.

VI mögliche Anwendungsgebiete

- **Militärisch**

solcher Roboter kann für Lebensgefährliche Aufgaben angewendet werden, wie beim Entdecken von unbekannten Objekt oder auch Minenbomben.

- **Zuhause**

Der Roboter kann auch als Aufräumroboter genutzt werden. Da er autonome Funktionen hat, kann er langwierige Alltagsaufgaben der Menschen übernehmen.

- **Krankenhäuser**

Es wäre hilfreich für die Krankenhäuser, wenn sie die infizierten Sachen bewegen können, ohne sie anzufassen, und somit mehr Krankheiten zu vermeiden. Hier kommt der Roboter zum Einsatz, weil er alles, ohne direkten Kontakt zu Menschen, sortieren kann.

VII Zusammenfassung

Die Roboter der Bombenentschärfer sind viel komplizierter und haben mehr Eigenschaften als das schon aufgebaute Fahrzeug. Eine andere Nutzung für diese Roboter ist die Beschädigung der unbekannten Objekte oder Bomben. Das kann der Lego Roboter nicht leisten, da er mit schwachen Material aufgebaut ist und es gefährlich wäre, ohne Erfahrung solche Waffen an den Roboter zu bauen.

Aber das Endresultat zeigt, dass diese Bombenentschärfer Roboter noch verbessert werden können, damit die Mehrheit der Menschen ihn bei Notfällen nutzen können.

Das Ziel dieses Projekts war trotz der ungelösten Probleme schon erreicht und zwar mit ein paar zusätzlichen Informationen über die Schwierigkeiten und Herausforderungen beim Bau eines solchen Roboters.

Eine andere Schlussfolgerung ist, dass mit einfachen Werkzeugen wie Legobausteinen etwas kompliziertes erbaut werden kann, obwohl sie nicht immer genau sind. Aber es ist möglich damit Zeit und Geld zu sparen, wenn die ersten Ideen mit Legos Mindstorm realisiert und programmiert werden.

VIII Literatur

[1]Planet-wissen, Roboter:https://www.planet-wissen.de/technik/computer_und_roboter/roboter_mechanische_helfer/index.html (abgerufen am 21.03.2019 um 14.15)

[2]Wepreserve,Der-Einsatz-von-Robotern-in-der-Industrie:http://www.wepreserve.eu/der-einsatz-von-robotern-in-der-industrie/ (abgerufen am 20.03.2019 um 9.30)

[3]FrankfurterRundschau:Ethik-und-Technik, https://www.fr.de/fr-serien/ethik-technik-11073924.html Version:15.05.17 (abgerufen am 20.03.2019 um 9.40)

IX Anhang

Abdeckung des Greifers

Bombenentschärfen Roboter

https://www.gettyimages.de/detail/nachrichtenfoto/denver-police-departments-newest-bomb-squad-bomb-nachrichtenfoto/161205106 (abgerufen am 21.03.2019 um 14.20)

Ballista - Alexanders großer Traum

Bericht zum Projekt des LEGO Mindstorms Seminars

Benedict Kunzmann, Elektrotechnik

Kurzfassung

Schon in der Antike war das Katapult eines der stärksten Waffen. Diese großen Maschinen konnte Schlachten von alleine entscheiden. Allerdings waren diese Katapulte sehr groß und schwer nachzubauen. Durch moderne Technik ist es allerdings möglich einen Katapult-Roboter nachzubauen und die Funktionsweise sowie die Physik und Mathematik dahinter praxisnah zu erklären. Die Hauptfunktion besteht darin eine Kugel über den Roboter zu werfen und so die B Bewegung eines Katapults nachzuahmen. Außerdem besitzt der Roboter Sensoren, womit er vor jeglichen Gefahren in der Umgebung geschützt ist.

1. Einleitung

Roboter sind seit dem 21. Jahrhundert ein unverzichtbarer Teil unserer Industrie und unserer Gesellschaft. Nicht nur in großen Fabrikhallen werden Roboter eingesetzt, sondern mittlerweile auch zu Hause. Diese übernehmen schwierige Aufgaben wie das Zusammenbauen eines Autos oder einfachere Aufgaben wie das Staubsaugen im Haus. Allerdings können Roboter auch zur Visualisierung von physikalischen und mathematischen Prozessen genutzt werden. In diese Kategorie fällt auch der Katapult-Roboter. Dieser wurde in einem Zeitraum vom 12 .Februar bis zum 23 .Februar konzipiert und entwickelt. Das Ziel war es ein Roboter zu bauen, der die Hauptfunktionsweise eines Katapults [das werfen von Gegenständen] imitiert. Zuzüglich sollte der Roboter noch die Funktion besitzen Hinternissen auszuweichen und vor Kanten zu stoppen und umzudrehen. Das Katapult soll erst schießen, wenn es genug Platz hat und sicher auf dem Boden steht. Der Roboter wurde mit Lego unter Verwendung des NXT-Bausteins aufgebaut und in MATLAB programmiert.

2. Vorbetrachtungen

Das Leben in der Gesellschaft wird immer technischer. Jeder hat ein Smartphone in der Hand oder steuert sein zu Hause mit einem Sprachassistenten. In naher Zukunft werden

DOI: 10.24352/UB.OVGU-2019-043

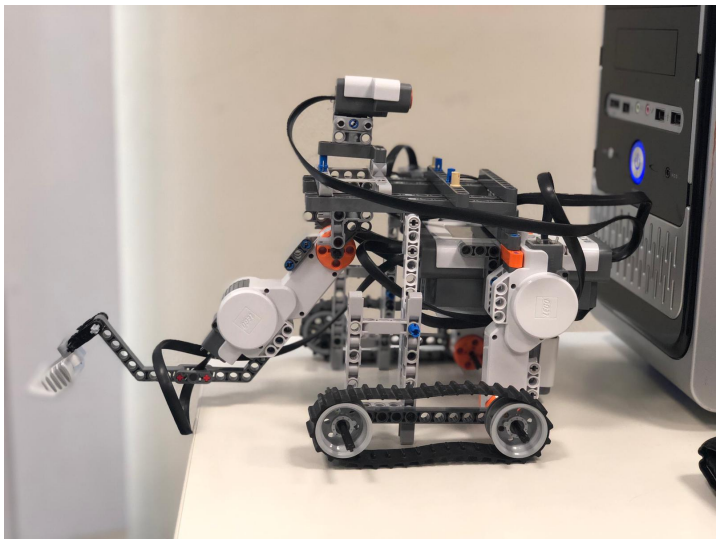
auch immer mehr KI's den Alltag der Menschen erleichtern. Aus diesen Gründen muss man die Menschen früh mit Technik in Verbindung bringen. Das Katapult trägt dazu bei, dass junge Menschen den Umgang mit Robotern lernen und ein Gefühl und Verständnis für den Umgang mit Robotern entwickeln.

3. Mechanische Umsetzung

Im Zentrum des Roboters befindet sich das NXT. Im 90° Winkel dazu befinden sich 2 Motoren, die für das Fahren des Roboters verantwortlich sind. Mittels einer kleinen Konstruktion auf der Oberseite der NXTs wurde ein dritter Motor verbunden, der Gegenstände wie ein Katapult wegwerfen kann. Auf dem NXT befindet sich ein Ultraschallsensor und unter dem NXT befindet sich ein Lichtsensor. Außerdem befindet sich noch ein Tastsensor auf der linken Seite des Roboters.

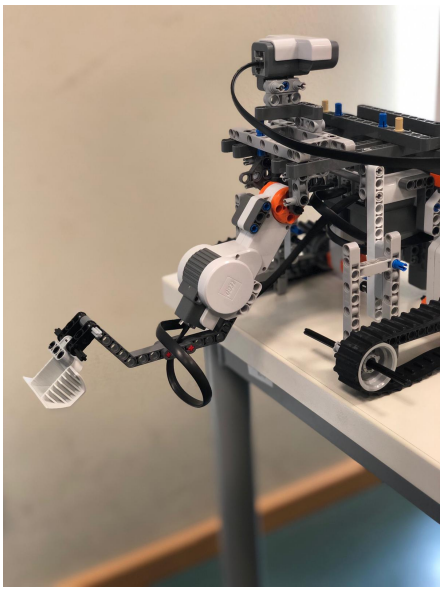
3.1 Die Ketten

Beide Ketten werden mit Hilfe von 2 Motoren angesteuert. Dadurch ist es möglich, das der Roboter wendiger wird und Kurven besser fahren und Hindernissen besser ausweichen kann. Die Räder wurden mit Gummiketten verbunden, um eine bessere Standfestigkeit zu erzeugen und die Bewegungen gleichmäßiger ablaufen zu lassen.



Um eine höhere Sicherheit zu gewährleisten wurden die Räder in der Mitte mit einem Stab befestigt, der mit dem NXT verbunden ist. Zum Schluss wurde die hinteren beiden Räder mit einem offenen Differential verbunden, damit im Fall einer Umdrehung nur die eine Seite der Räder arbeitet und die andere still steht.

3.2 Der Wurfarm



Der Wurfarm ist das Hauptteil unseres Roboters. Er besteht aus einem Motor und einer Verlängerung mit Schaufel, in die man kleine Kugeln hineinlegen kann. Der Motor selber ist mit einem Stab an der Oberkonstruktion befestigt, wodurch er sich nur radial bewegen kann, was zu einer optimalen Flugbahn des Gegenstandes führt.

4. Programm

Das Steuerprogramm ist so konzipiert, dass der Roboter viel mit dem Benutzer kooperiert. So hat der Benutzer immer die Kontrolle und es wird wahlloses rumfahren vermieden.

4.1 Sensoren

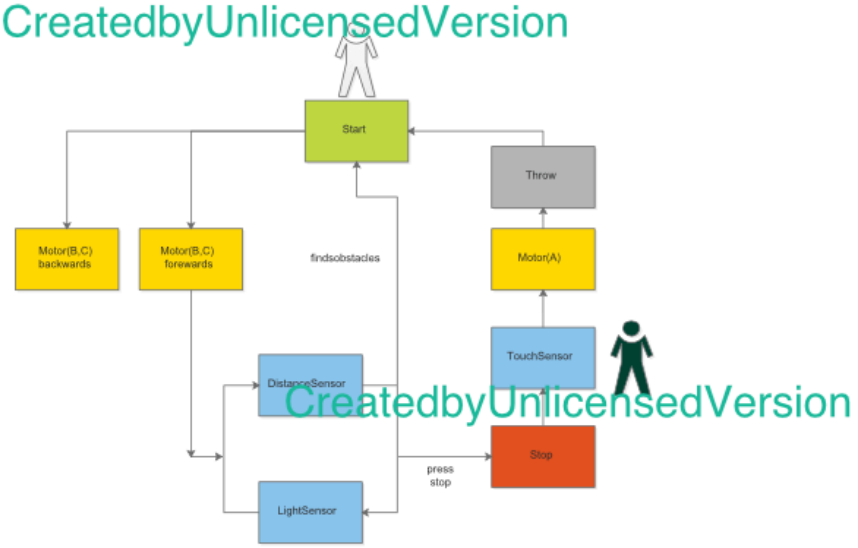
Um diesen Programmplan umzusetzen werden einige Sensoren verwendet. Der Ultraschallsensor misst die Distanz nach vorne und signalisiert Gegenstände in geringer Entfernung. Zudem misst ein Lichtsensor die Entfernung zum Boden und kann so vor Kanten warnen. Der Tastsensor wird benutzt um den Wurfarm zu bewegen.

4.2 GUI

Zur besseren Übersicht wurde eine GUI eingebaut. Dadurch hat man die Möglichkeit die wichtigsten Funktionen selber zu bestimmen. Es gibt eine Start-Funktion um den Roboter nach vorne fahren zu lassen. Zudem gibt es einen Rückwärtsgang, um die Wurfposition zu optimieren. Als letztes gibt es noch einen Notstop, der den Roboter stoppen soll, wenn dieser zu viel Input hat oder ein Programmschritt falsch ausführt.

4.3 Ablaufplan

Unser Programm startet mit einem Klick auf den Startbutton. Danach fährt der Roboter geradeaus. Im selben Moment werden auch die beiden Sensoren [Ultraschall-, Lichtsensor] aktiviert. Die beiden senden und empfangen ständig Signale von der Umgebung. Wenn der Roboter ein Hindernis gefunden haben sollte, dann stoppt der Roboter, fährt seinen Wurfarm nach oben (um den Radius zu verringern) und dreht sich



um 90°. Nach dieser Aktion stoppt der Roboter erstmal und man muss erneut auf den Startbutton drücken, um den Roboter erneut fahren zu lassen. Danach geht der Vorgang wieder von vorne los. Eine andere Möglichkeit ist den Roboter mit Hilfe des Stopknopf zu stoppen. Nachdem man dies getan hat, kann man eine Kugel in die Schaufel legen, den Tastsensor betätigen und die Kugel wegschleudern. Der Roboter fährt dem Wurfarm anschließend zurück und der gesamte Vorgang kann durch den Starknopf erneut starten.

5. Die Ergebnisse

Das Hauptziel des Projektes wurde erfüllt: Der Roboter kann die Kugel über den Roboter werfen. Dieser Vorgang funktioniert allerdings nicht in allen Fällen gleich. Ab und zu wirft der Roboter die Kugel nicht über sich selber drüber, sondern die Kugel fliegt dann auf den Roboter. Die Bewegung funktioniert dagegen schon zuverlässiger. Wenn ein Hindernis zu nah kommt wird dieser zuverlässig erkannt und der Roboter dreht um. Auch der Lichtsensor erledigt seine Arbeit zuverlässig. Er erkennt die Tischkante rechtzeitig und lässt den Roboter umkehren. Allerdings sind nach dem Wenden öfter zu viele Informationen zum NXT gelangt, wodurch der Roboter öfter überfordert war.

6. Zusammenfassung

Es war ein gelungenes Projekt. Durch Lego Mindstorms sind der Kreativität keine grenzen gesetzt. Selbst bei einem fertigen Projekt kann man immer noch Kleinigkeiten finden, die man optimieren kann. Dazu zählt die Optimierung des Wurfarmes [für einen weiteren Wurf], mehr Stabilität des gesamten Roboters und auch das Programm lässt sich noch besser schreiben. Für diese Feinheiten fehlt allerdings die Zeit in dieser doch vollgepackten Projektwoche. Da der Roboter aber auch für lehrende Zwecke eingesetzt werden soll, wurde das Ziel des Teams erfüllt.

Literaturverzeichnis

http://www.nxtprograms.com/catapult/steps.html

https://de.wikipedia.org/wiki/Roboter#Technische_Grundlagen6

Ballista – ein Traum von Alexander

Hannan Javed Mahadik, Mechatronik
Otto-von-Guericke-Universität Magdeburg

Abstract - Robots and machine are taking over a lot of manual work, earlier done by men, and making it better by being not only quicker in doing the same task, but also in a much more efficient way.

This paper is heavily based on the Ballista (Catapult) used by many great rulers including Alexander the Great in a bid to win wars. It is an engineering marvel built to launch projectiles across great distances inflicting quite some damage. Depending on the needs, a catapult can be modified and used to destroy or create an opening in mountain ranges or just simply as a war weapon. We have created a miniature, albeit working prototype of the same using the Lego Mindstorms NXT kit and programming of the same using MATLAB.

Thanks to the programming kit made by students of RWTH

Aachen, which is available for public use, programming of our Ballista / Catapult model was made much easier and efficient.

Keywords – Autonomous, Ballista, MATLAB, Lego Mindstorms NXT kit, Boolean logic

I. INTRODUCTION

One of the main reasons for building an autonomous model of the great Ballista is naturally efficiency. The Ballista should be able to move on its own, and additionally be astute enough to sense when there subsists an obstruction in its path and then turn accordingly in either the left or right direction, unless asked to halt by the user incharge. Naturally, to obviate any accidents due to the shortcomings of the technology we have available today, the final kineticism, i.e. of launching the projectile, was to be made manual by using a simple and reliable button.

Engendering an aperture in a mountain range may not sound as important as it actually is. Doing so can greatly cut down the transportation time of humans, which is especially important when lives are at stake as immediate medical attention may not be available in that segment of society anteriorly disunited from cities/towns due to the mountain range. Also, in reducing the transportation time, and the distance travelled, will help financially, as it will not only reduce the costs incurred by utilizing lesser amount of petrol, but it is also sustainable as the environment won't be as damaged.

This was the main aim of our project as we tried to engineer a sustainable and useful machine.

II. PREVIEW

A. *Use of Boolean Logic*

The prototype engineered heavily relies on simple Boolean logic to decide whether it is possible to keep moving forward or if it requires to turn in either direction due to an obstruction or hinderance in its path. Utilizing either 0 or 1, signals are sent to the NXT Motors and processed accordingly. As the model only needed to decide if an obstruction subsisted or not, Boolean was not only logical, but withal efficient.

The turning of the ballista was made possible by simply having one motor run in one direction, and the other in the opposite, ensuring that the result is quick and desirable.

B. *Basic Programming*

MATLAB was to be used as the primary software to program the 'brain' of the Ballista.

The functions were to be independently written, tested, and then concatenated to ascertain stability and efficiency.

III. BODY

A. *Brainstorming*

At the very beginning, the group reviewed the current literature available to come up with a fairly different, and exciting idea.
Out of the many impressive models, the Segway Robot Model was the one that seemed the most impressive as it had to balance itself on its 2 wheels, and also while it was something the general audience could easily relate to. Using this Segway model, the initial idea was

DOI: 10.24352/UB.OVGU-2019-044

to build a prototype that might help disabled people by implementing a similar mechanism into automatic wheelchairs, making them 'smart' and hence eliminating the risk of them tipping over and causing unnecessary accidents.

Figure 1: The original Segway Model [1]

However, this was not to be chosen due to a lot of technical as well as mechanical issues which proved to be a deciding factor in picking out the final project idea.

B. *Metamorphosis*

So as to not have the efforts put in go to waste, the Segway model was dismantled and converted into something different, which was then to be used as the main project. The 'body' of the Segway rider was used as the main motor responsible for the 'arm' of the Ballista, while the 'head' of the same, which was basically an Ultrasound

Sensor, was used to detect obstacles as shown in the adjoining figure(2).

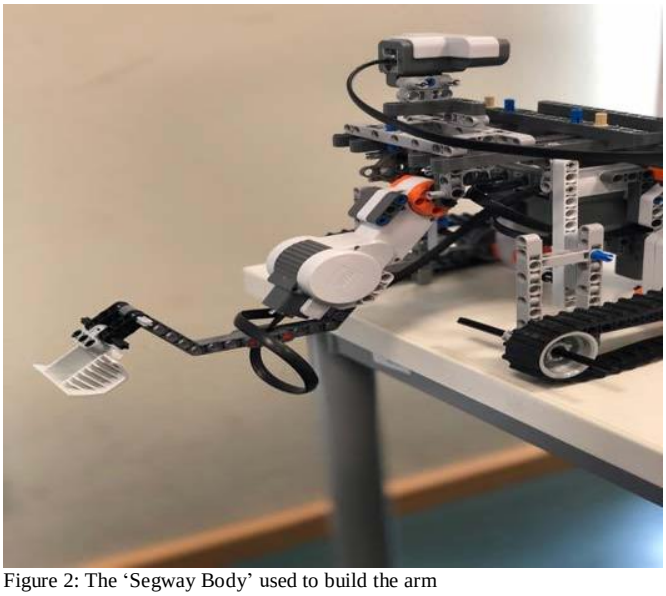


Figure 2: The 'Segway Body' used to build the arm

C. *Building the Ballista*

As mentioned earlier, the 'head' of the Segway rider, an Ultrasound sensor, was attached to the top of the Ballista model, so as to identify any obstacle that may lie in the path of the prototype. The sensor sends signals to the robot according to the distance at which the hinderance stood, ensuring that the Ballista did not simply try to knock the obstacle away, but instead stop and turn in a different direction in a uniformly manner.

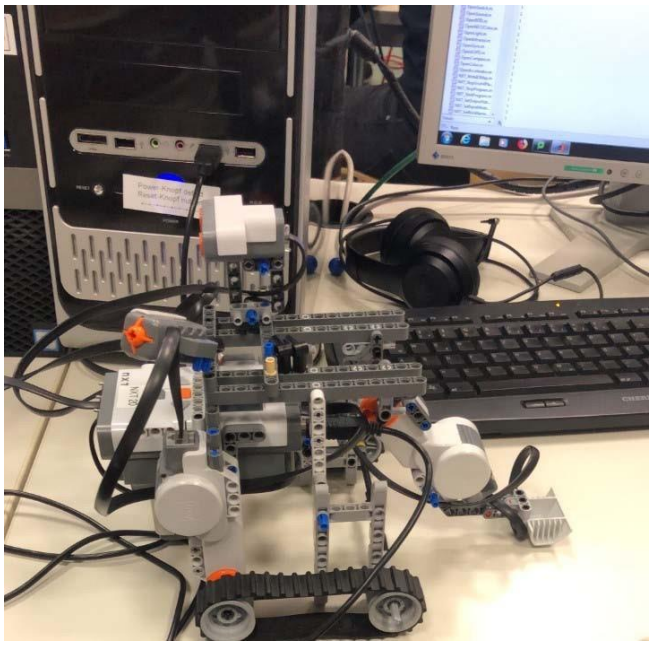


Figure 3: First build

An arm made with a simple Lego pieces were attached to the motor, with the motor programmed to run at full power to ensure proper projectile motion, having adequate speed ensured that the projectile travels the required distance quickly, and has desired impact/effect. A touch sensor was attached to the prototype to send a signal to the Lego Mindstorms brick to activate the catapult arm.

Figure 4: The Ultrasound sensor placed at the top of the model

Originally, four wheels were attached to either side of the Lego Mindstorms NXT brick. This however, affected the way the

Ballista's movement and stability. To overcome this issue, caterpillar tread tracks were used as replacement. This proved to be a vital decision as the prototype was now able to even tread through rougher terrains without running the risk of it tipping over.

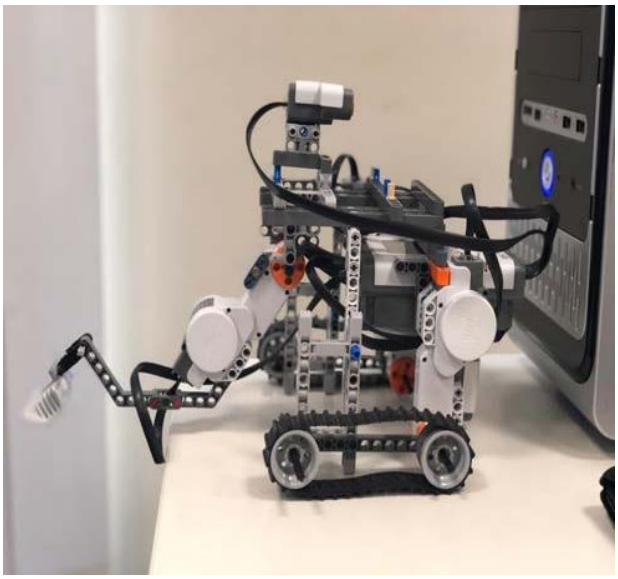


Figure 5: The Caterpillar tread tracks

The next hurdle was deciding how we could make the Ballista recognize edges. This was needed to ensure that the Ballista did not simply fall over, and damage or waste the resources used. This was done by simply relying on a light sensor, which was to be attached to the prototype in such a way that it pointed directly downwards. This was done to measure the reflectivity of the surface that the Ballista was to drive upon. In case of a cavity, the intensity of the light reflected back would not be as high as the one predefined as per the written code. This would then signal the Ballista not to move any further and instead to turn around in either direction and then proceed accordingly.

Therefore, to summarise, the main components of the model were:

- One Ultrasound Sensor
- One Touch Sensor
- One Light Sensor
- Three Motors
- Tread tracks

D. Programming

All of the required programming was done on MATLAB using the predefined NXT functions released by RWTH Aachen.

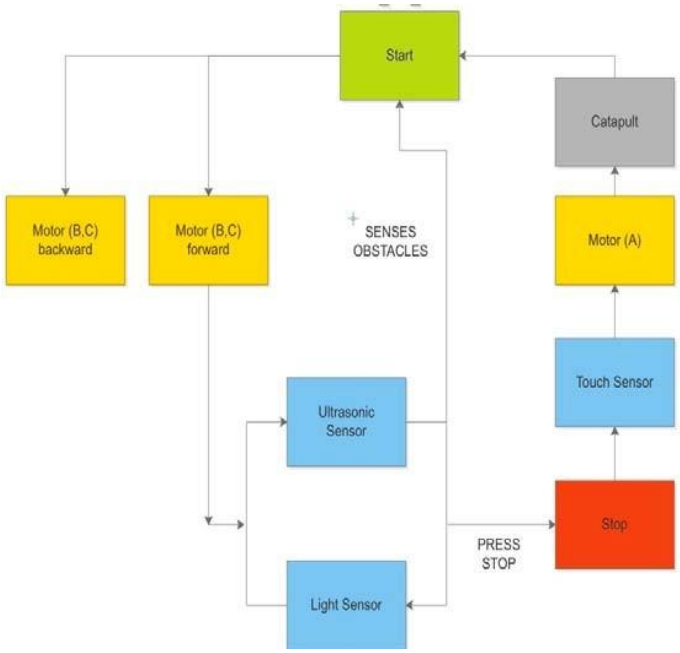


Figure 6: The Program Layout

The program was planned out to be written in parts as seen above to ensure efficiency, and also save time, as more than one person could simultaneously work on different functions. It is efficient because, in the case of an error, it is easier to identify and correct it when the code is separated in parts, as compared to debugging one big block of code.

```
OpenSwitch(SENSOR_2);
OpenUltrasonic(SENSOR_1);
OpenLight(SENSOR_3,'ACTIVE');

while 1
    % if switch is pressed he shoots with the catapult
    if GetSwitch(SENSOR_2) == 1
        handles.motorA = NXTMotor('A','Power',-100,'TachoLimit',100);
        handles.motorA.SendToNXT();
        pause(1);
        handles.motorA = NXTMotor('A','Power',10,'TachoLimit',100);
        handles.motorA.SendToNXT();
    end

    dist = GetUltrasonic(SENSOR_1);
    fprintf(['Distance:' num2str(dist) newline]);
    % drives forward until the distance is smaller than 15
    if dist < 20
        handles.motorA = NXTMotor('A','Power',-50,'TachoLimit',100);
        handles.motorA.SendToNXT();
        pause(0.3);
        handles.motorC = NXTMotor('C','Power',-50,'TachoLimit',475);
        handles.motorC.SendToNXT();
        handles.motorB = NXTMotor('B','Power',50,'TachoLimit',475);
        handles.motorB.SendToNXT();
        handles.motorB.WaitFor();
    end
    pause(0.2);

    x = GetLight(SENSOR_3);
    pause(0.1);
    if x < 600
        handles.motorC = NXTMotor('C','Power',-60,'TachoLimit',300);
        handles.motorC.SendToNXT();
        handles.motorB = NXTMotor('B','Power',-60,'TachoLimit',300);
        handles.motorB.SendToNXT();
        pause(2);
        handles.motorC = NXTMotor('C','Power',-50,'TachoLimit',475);
        handles.motorC.SendToNXT();
        handles.motorB = NXTMotor('B','Power',50,'TachoLimit',475);
        handles.motorB.SendToNXT();
    end
```

Figure 7: Snapshot of the Functions for the Sensors

The above attached image consists of the most important part or the 'heart' of the program. The 'while loop' makes sure that the program runs infinitely until the brick is manually shut down by the user.

The first if-else function reads the signal from the Touch sensor which relies on Boolean logic as its 'condition'. The Tacholimit ensures that the arm of the Ballista doesn't actually hit the body of the Robot and cause a malfunction. It is set to full power (i.e. 100) so that the Projectile takes flight with enough speed and power necessary to get the desired result upon impact. Thereafter, the power is set to 10, but in the opposite direction, which helps the arm to return to it's original position.

The following two functions are responsible with the movement of the robot. The Tacholimit here was set to 475 after carefully examining how many rotations the motor needed for the Ballista to rotate by 90 degrees. Albeit the Tacholimit in the next function is set to the value of 300, it does not provoke any change as the value is arbitrary and is only used to signal the prototype to move in the opposite direction and not over the brink of the surface in question.

IV. RESULT

After many runs and re-runs of the project, it is safe to conclude that the original goal set was achieved and the expectations met, if not exceeded. One thing which could be improved upon is the arm. A motor with higher power could ensure that the projectile travels with an even greater speed, and hence, have an even better impact.

Other than the aforementioned, the sensors could be further improved upon, technically, to reduce any errors. However, doing so would only make a minor difference in the functioning of the robot.

V. SUMMARY & CONCLUSION

The Ballista built by the ancient Romans was truly such an extraordinary piece of machinery. A weapon with great potential to be critically devastating in wars.

Using simple Boolean logic, the prototype built was able to traverse simple courses, that included few basic obstacles, large enough for the Ultrasound sensor to sense, and for the Ballista to then process and proceed accordingly. Also, the possibility of the Ballista traversing and falling over edges into pits or likewise was eliminated.

With a camera, instead of the ultrasound sensor, and with a few changes in the code, the Ballista could be built 'smarter'. It could be programmed to identify the type of obstacle and decide if it is an actual physical barrier of importance, or is it something that has no value, and could simply be driven across.

Of course, with the right implementation of technology, this prototype could be modified in infinite number of ways, and then recreated to a life-like scale. The possibilities are endless.

BIBLIOGRAPHY

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

2) *[1] - http://www.nxtprogram.com/NXT2/segway/*

BALLISTA - EIN TRAUM VON ALEXANDER

Preetdeepan Prashantkumar Pradhan, Electrical Engineering & Information Technology
Otto-von-Guericke-University of Magdeburg

Abstract — In this modern era of work segregation, autonomous robots have challenged the innate perspective of mankind. They not only save human labour, but also provide assistance where no mankind has ever stepped its feet. The modification of this medieval weapon into a functioning prototype, with several uses in the field of constructions or recreational sport. The project uses the Boolean Logic partially, as the robot moves around, detects hinderances and turns 90° left.
The apparatus building requirements were from the 'Lego Mindstorms Education Set', that developed the corresponding programme in MATLAB. In order to provide a stable communication between MATLAB and the LEGO-NXT Module, the use of MATLAB Package from RWTH-Aachen was very essential.
As a result, we received a working model and a few more comments based on our development process that should be met in order to be able to use the plowing machine in practice. On the one hand there are additional sensors to detect pedestrians or other things that are not obstacles and on the other hand it would be a satellite transmitter to detect which surface has to be processed and where, for example, the earth can be taken.
Keywords — Autonomous Driving, Boolean Logic, Lego Mindstorms, Matlab.

I. INTRODUCTION

The quick uprising of robots, that outperform humans have sustainably replaced and made our lives relatively easier to work with. The use of robots, that can be easily programmed and worked with have significantly increased. The best example for the same is the Automobile sector, Packaging system and most importantly in the Food & Drugs division.

Robots have made our daily work habits much convenient, as they take over simple or maybe complicated tasks, freeing mankind to explore the unexplored, and be busy with the same.

There have been technological discoveries, inventions and most importantly discussions about improvement of existing technologies throughout the existence of humanity.

One such prime example, is the Catapult, a medieval weapon, that can be used in various other fields, especially with constructions of temporary bridges using boulders for the military in places, where human involvement is very shrewd, crude and difficult. Here is the good use of the robot, as it will be able to provide assistance.

Exactly what the project "Ballista" sets, that has been brought to life during the period of 11.02.2019 till 22.02.2019 in the LEGO Mindstorms seminar. The aim was to create a prototype as a model based on the designs of the ancient Greek Philosopher Diodorus Siculus. This is a simple illustration of what followed thereafter, the rise and fall of kingdoms, dynasties, eras and most certainly empires.

This model should, while following simple codes that can be easily brought about by any technician can cause fatal damage to the opponent with a reaction time so slow, that the model can detect obstacles, thereafter change its direction of movement and recalibrate to function again. The model follows a line (as a simplified road) recognise obstacles and test whether its projectile is affected by the

II. PREVIEW

Boolean Logic

The use of Boolean Logic is extremely essential for the robot's movement, as the detection of an obstacle greater than the robot's height would make the robot change its path of movement. The explicit and exact data is delivered using the Boolean logic, as it provides only and only two values, either 0 or 1, where specific commands were allocated to both the values.

Basic Programming

For basic programming, command overview, where codes could be written, was used. In addition to the communication between MATLAB and the NXT module, it also describes the basic program sequences, as well as the integration and use of functions. A measure had to be found to output the currently running processes.

DOI: 10.24352/UB.OVGU-2019-045

III. BODY

Figure 1. Segway

A. Brainstorming

The basic concept was a vehicle that has simulated fixed steering to make manoeuvring easier and making the tropics almost negligible. The initial point of framework was a Segway. Due to faulty measurements and imbalanced structure, the idea was scrapped. The key points of failure were carefully analysed, discussed and thereafter improved, leading the main point of focus to make sure that the vehicle was compact, robust and outstandingly constructed so that a low centre of gravity was achieved. Thus making it easier to move with objects or the armament, and reduce the risk of the vehicle tipping over and being damaged. This shall definitely not be the case with this particular model, but in practice it shall be a very important point that cannot be neglected, if the robot has to drive over rough terrain.

Figure 1. is the model representation of a Segway, and the origin of the idea of a catapult.

The initial construction of the Segway was very useable, but after fixing the wheels to the Segway, a major error was noticed and thereafter all other challenges were brought about, for example the stability due to very high and displaced centre of gravity, weight of the NXT Motor, the slow processing of the NXT Program, and the very high latency. This was the basic idea on which the structure and the project development was done. Although a beta, this seemed to be a working model, that could be improvised on and could have been made more interesting and useful.

Also, necessity is the mother of invention, which has been the guiding factor in the complete changeover of a basic Segway model into one of the greatest and the deadliest weapon of the medieval and ancient Greek era.

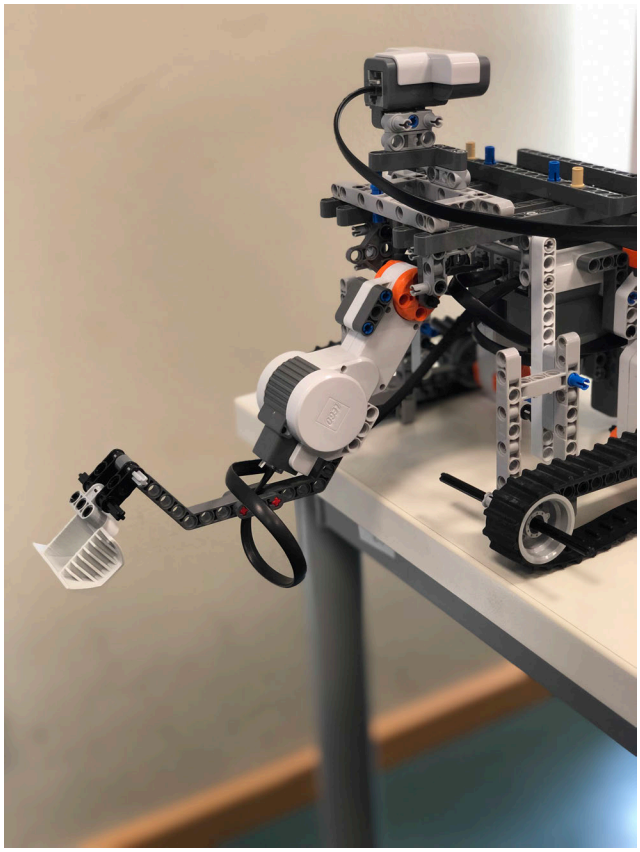


Figure 2. Head of Segway converted into Catapult head.

Figure 2. is the step one of the changeover of the Segway. The initial framework of the Segway had a head, which was later converted into the arm of the catapult, there was a programme written for the same, to assign engine force to the arm, the power supplied to the same was of 100 percent.

The robot is able to swing its arm in a semi circle with 100 percent power, which enables the object or the armament loaded to move in a projectile. This is indeed one basic movement, compared to the conventional variants of a catapult, where realignment and reloading is vital and consumes significantly large amount of time. The reconfiguration of the programme, after adoption should definitely remove all the errors and help stabilise, and achieve the required projectile with the adjustment of the angle of the arm.

After this the robot's linear movement with the arm placement was worked upon.

The wheels of the Segway were replaced with Caterpillar tread track, that made the linear movement stable and straight. The caterpillar tread tracks were fixed to two adjoining motors that were supplied with an engine power maximum of 30 percent, which could make the robot's trajectory linear, stable and coherent. This also provided time for the other sensory movements of the robots to detect obstacles, make the necessary change of path and avoid edges to prevent falling off course during travel time.

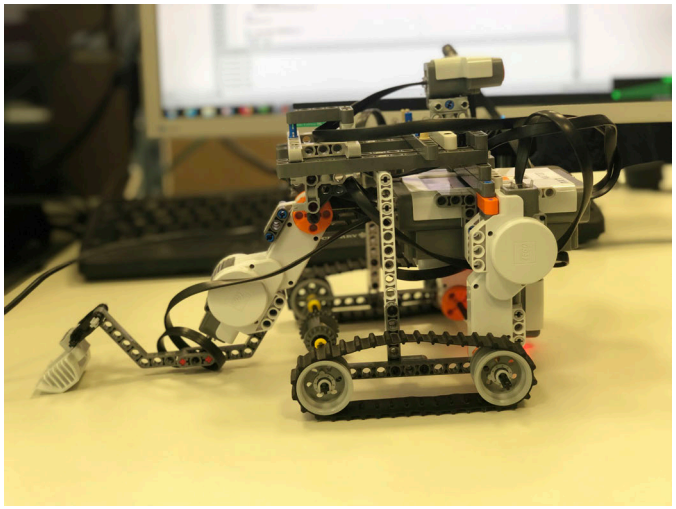


Figure 3. Final Build of Ballista

Figure 3. After the initial testing of the linear movement of the robot, the decision to install the sensors (after calibration) in order to increase the sophistication of the robot was challenging. After that it was about the robot to detect obstacles and later to test them for displacement.

Keeping in mind about the terrain change, and stability the following sensors were installed. The following NXT sensors are important for this:

• One Ultrasonic Sensor

• One Light Sensor

• One Touch Sensor

The Ultrasonic sensor is located right on the top of the robot, disguised as the head of the robot as in seen in Figure 3, and this sensor faces to the direction of travel. The former is used to detect whether an obstacle is on the road, so that the following can give a command to the adjoining engines connecting the wheels to turn, if the obstacle is greater than the height of the robot, so that any and every kind of hinderance during the projectile can be avoided. It is also used to detect when the robot can avoid the obstacle with a left turn or if it is still on the line.

As previously described, the Light sensor located in the front centre of the robot so that it can track the edges. It can measure the reflectivity of the surface, therefore if there is a hole or a gap in the road, light from the sensor will travel a longer distance hence the reflection shall be invariably low causing the sensor to realise an edge. Henceforth causing the robot to change its direction and movement. After realising an edge, the robot tends to go back a bit, and then take a left turn, since the edge could be lengthwise longer than expected by the robot.

```
OpenSwitch(SENSOR_2);
OpenUltrasonic(SENSOR_1);
OpenLight(SENSOR_3,'ACTIVE');

while 1
    % if switch is pressed he shoots with the catapult
    if GetSwitch(SENSOR_2) == 1
        handles.motorA = NXTMotor('A','Power',-100,'TachoLimit',100);
        handles.motorA.SendToNXT();
        pause(1);
        handles.motorA = NXTMotor('A','Power',10,'TachoLimit',100);
        handles.motorA.SendToNXT();
    end

    dist = GetUltrasonic(SENSOR_1);
    fprintf(['Distance:' num2str(dist) newline]);
    % drives forward until the distance is smaller than 15
    if dist < 20
        handles.motorA = NXTMotor('A','Power',-50,'TachoLimit',100);
        handles.motorA.SendToNXT();
        pause(0.3);
        handles.motorC = NXTMotor('C','Power',-50,'TachoLimit',475);
        handles.motorC.SendToNXT();
        handles.motorB = NXTMotor('B','Power',50,'TachoLimit',475);
        handles.motorB.SendToNXT();
        handles.motorB.WaitFor();
    end
    pause(0.2);

    x = GetLight(SENSOR_3);
    pause(0.1);
    if x < 600
        handles.motorC = NXTMotor('C','Power',-60,'TachoLimit',300);
        handles.motorC.SendToNXT();
        handles.motorB = NXTMotor('B','Power',-60,'TachoLimit',300);
        handles.motorB.SendToNXT();
        pause(2);
        handles.motorC = NXTMotor('C','Power',-50,'TachoLimit',475);
        handles.motorC.SendToNXT();
        handles.motorB = NXTMotor('B','Power',50,'TachoLimit',475);
        handles.motorB.SendToNXT();
    end
```

Figure 4. Essential Functions from the source code

Figure 4. portrays all the indispensable functions.

The 'while loop' function is an infinite loop and ensures the successful running of the program until user intervention manually by pressing the switch. Boolean logic plays a significant role while enabling the 'if-else' functions, as it detects the signal from the Touch Sensor as a 'condition'. Any stalling of the robot's arm can be prevented and controlled using the 'Tacholimit'. The power supplied to the same is 100 or full power which judges the projectile perfectly ensuring the desired result.

The returning power to the arm is set at 10 in the opposite direction, which ensures that the robot's arm returns to its original position easily and slowly.

The movement of the robot are endowed on two main functions. The Tacholimit was fixed on 475 after careful analysis in consideration to the rotations of the motor. The next Tacholimit was fixed on 300, it does alter any change, as the value is random and is only used to guide the Ballista to proceed in the "negative" direction and not over the edge of the surface in consideration.

The Touch sensor fixed to the robot, is actually a button that enables the user to fire the Ballista after carefully fixating the robot with the desired coordinates.

The Touch sensor is absolutely user based and has no vital connection to the movement of the robot in whole.

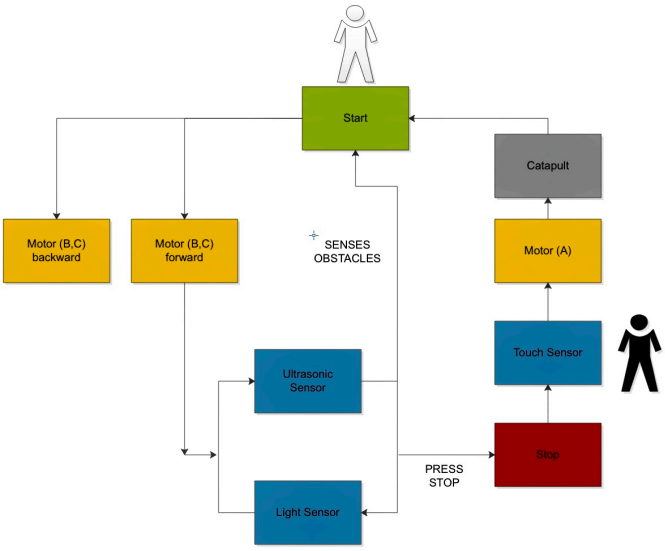


Figure 4. Program Layout
Yellow: Motors
Blue: Sensons
Start and Touch Sensor require human intervention

The basic program outlet, the basic construction and the execution has been described in the above mentioned Figure 5. The user presses the Start button, the commanding motors are driven with power and the wanted movement takes place, the light sensor and ultrasonic sensor help the robot drive and stay stable, and with the user's command with the touch sensor enables the robot to fire i.e. move the arm and project the armament for the desired outcome.

IV. RESULT

After repetitive running, collecting, analysing and summarising the robots', data, movement and analysis, one could conclude that the manuscript-code is usable and the original aim was achieved. The point of major improvement is the arm, a motor with a higher power efficiency can replace the already existing motor, so that the delivery of power to the arm can be sufficient for a stronger projectile.

The sensors need major improvement, technically, to avoid unfruitful errs. High precision can be derived.

V. SUMMARY

By using Boolean Logic as the basis of sensory movements of the robot, the establishment of a linear movement is possible, in spite of obstacles, gaps, holes or edges present. When the robot faces a hinderance, it finds a way independently and then continues on a new path on the marked section.

This can be considered as a basic prototype for high-grade military artillery.

For the decision-making, there should also be found a way to clearly distinguish between a disruptive factor (branches or boulders) and a normal obstacle (car or motorcycle).

VI. BIBLIOGRAPHY

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

II. http://www.nxtprograms.com/NXT2/segway/

Der Spotfinder

Daniel Behling, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

Abstract- In dieser Arbeit handelt es sich um die Entwicklung und Realisierung eines mit der Software Matlab programmierten Roboter. Es war die Aufgabe, sich innerhalb von zwei Wochen mit dem Programm vertraut zu machen und in Zweiergruppen selbst eine Idee zu finden was der, mit Lego Mindstorms gebaute Roboter, einmal können soll. Die Arbeit unterteilt sich in eine Einleitung, bei der auf die Idee eingegangen und etwas zu den möglichen Anwendungsgebieten gesagt wird. In der darauf folgenden Vorbetrachtung werden dabei die bereits vorhandenen Lösungen vorgestellt. In dem Hauptteil wird auf das Konzept des Roboters und auf die Umsetzung eingegangen. Abschließend folgt dann noch eine kurze Ergebnisdiskussion und ein Fazit. Der „Spotfinder“ sollte in der Lage sein durch das scannen des Untergrundes und dem Finden von farbigen Punkten selbstständig fortbewegen können. Je nachdem was dann für eine Farbe gefunden wurde, kann er zusätzlich festgelegte Befehle ausführen.

Der Spotfinder ist ein selbstständig fahrender Roboter der sich das das erkennen von Farben auf einem Untergrund fortbewegt.
Bei ihm handelt es sich nicht um einen Roboter, der im voraus ein speziell festgelegtes Anwendungsgebiet hat. Vielmehr geht es dabei um die Funktions- und Herangehensweise des Roboters, was ihm ermöglicht ein breites Feld an Anwendungszwecken abzudecken. Das Anwendungsgebiet kann ihm also im nach hinein zugewiesen werden.
Mögliche Gebiete wären die Untersuchung von Dellen, Kratzern oder Unebenheiten an Gebäuden, Brücken oder den Tragflächen eines Flugzeuges.

Wie der Name schon sagt soll der Spotfinder ein bestimmten Punkt finden, analysieren und danach auswerten.
Der Roboter sollte in der Lage sein, ein bestimmtes Gebiet bzw. einen bestimmten Untergrund selbstständig zu scannen und je nach Ergebnis verschiedene Schlüsse ziehen. Um dies zu ermöglichen muss er mit Sensoren ausgerüstet werden, die Veränderungen im Untergrund wahrnehmen können.
Wurde ein Unterschied wahrgenommen, muss der Roboter diese Information verarbeiten und auswerten. Und je nach Auswertung führt er dann einen ihm im voraus einprogrammierten Befehl aus. Damit die Sensoren einen Bereich gleichmäßig abdecken können muss der Roboter zusätzlich kompakt und wendig sein.

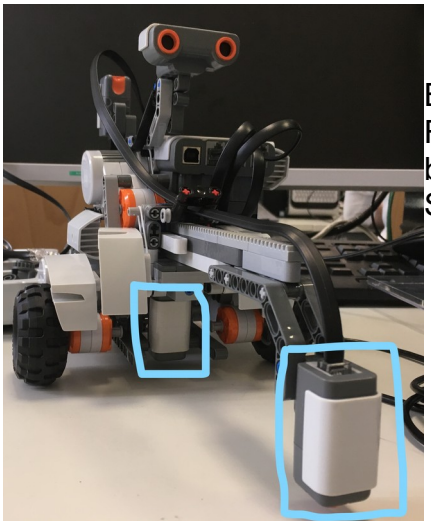


Bild 1: Roboter mit beiden Sensoren

Um das Konzept der Spotfinders zu realisieren mussten ein paar Dinge beachtet werden. Als erstes musste geklärt werden wie der Roboter den Untergrund scannt. Dafür wurden zwei verschiedene Sensoren genutzt.

DOI: 10.24352/UB.OVGU-2019-046

Der erste (vordere) sollte einen Unterschied im Untergrund erkennen und der zweite (hintere) dann genau feststellen um was für einen Untergrund es sich handelt.
Um einen Bereich gleichmäßig scannen zu können sollte sich der Roboter dafür um seine eigene Achse drehen. Ein drittes, loses Rad wurde für diesen Zweck angebracht um eine einfache Rotation zu ermöglichen.
Der Roboter sollte sich dann durch eine Endlosschleife so lange drehen bis er einen Wechsel auf der Oberfläche registriert hat. Nach dem registrieren stoppen die Motoren und er fährt so lange geradeaus bis der zweite Sensor den (im jetzigen Fall) Punkt gefunden hat.
Findet der erste Sensor nach 360 Grad drehen keinen Punkt fährt er seinen, mit einem Motor gesteuerten, Arm aus um so den Radius zu erhöhen.
Der Arm wurde durch ein Zahnrad ausgefahren und besaß drei verschiedene Stufen. Wenn er nach dem zweiten mal um seine Achse drehen erneut keinen gefunden hat würde er seinen Arm wieder ausfahren und so wiederholt sich der Prozess.
Ein Problem was dabei auftauchte war, dass sich die Vorrichtung durch ihr
eigenes Gewicht nach unten gezogen hat und so der Abstand des Sensors zum Untergrund mit höherer Stufe immer geringer wurde.

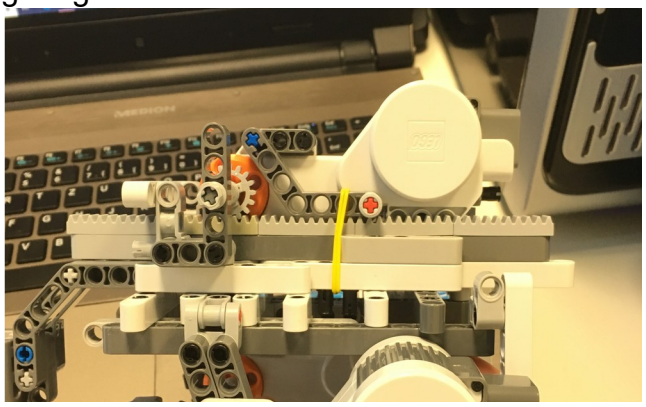


Bild 2: Vorrichtung des ausfahrbaren Armes

Dies führte wiederum zu Messfehlern des Sensors. Um das weites gehend zu verhindern wurde ein Gummiband eingespannt welches den Arm waagerecht hielt aber auch nicht zu viel Spannung besaß, sodass er jenen problemlos wieder einfahren konnte.
Das Scannen der Punkte erfolgte wie oben schon angerissen durch eine 360 Grad Drehung der Motoren, die dafür entgegengesetzt angesteuert wurden. Wenn jetzt bei den zweiten 360 Grad, also der ersten Stufe des Armes, ein Punkt registriert wurde, fährt er seinen Arm ein und so lange geradeaus bis der zweite Sensor den Punkt erkannt hat. Je nach dem, um welche Farbe es sich bei den Versuchen handelte, wurde entweder ein Ton abgespielt, die Entfernung nach vorne gemessen, die Raumhelligkeit ausgegeben oder im Programm Matlab ein Text angezeigt. Zusätzlich wurden dem Roboter zwei Abbruchkriterien einprogrammiert. Das erste war eine bestimmte Farbe z.B. schwarz, die vorab vom Nutzer festgelegt wurde. Das zweite Kriterium war, dass wenn der Sensor alle drei Radien gescannt hat und trotzdem keinen Punkt finden konnte.
Für den Fall, dass eines der beiden Kriterien eintraf, sollte der Roboter wieder zum Startpunkt fahren.
Um dies zu realisieren, wurde jeder Winkel und jede Entfernung der einzelnen Punkte abgespeichert und die Motoren nach jedem scann des zweiten Sensors auf Null zurückgesetzt.
Die Werte wurden in einer Matrix abgespeichert und Rückwärts abgerufen, sodass der Roboter in der Lage war seinen abgefahrenen Weg wieder zurückfahren und sich in seine Ausgangsposition begibt.

Abschließend kann man sagen das der Roboter am Ende in der Lage war, alle ihm gestellten Anforderungen im Grunde bewältigen zu können. Er hat mehrere Tests problemlos durchlaufen können und zum Ende nur noch selten Fehlermeldungen ausgegeben. Jedoch sind uns auch während den ersten Tests immer wieder kleine Fehler und Verbesserungsmöglichkeiten offensichtlich geworden.
Ein Problem, was während der Testphase bemerkbar wurde, ist das der Spotfinder mit seinem vorderen Sensor zwangsläufig den vorherigen Punkt, der ja 180 Grad hinter ihm lag, überfahren und
wahrnehmen würde. Um dieses Problem zu lösen wurde ihm einprogrammiert alle Punkte in einem bestimmten Bereich hinter ihm (ca. 175- 195 Grad) zu ignorieren.
Dies wurde dann auch wie erwartet ausgeführt und bereitete dann keine Schwierigkeiten mehr.
Ein Fehler, welcher jedoch nicht so leicht durch das programmieren lösbar war, ist die Ungenauigkeit und Empfindlichkeit der Lego Mindstorms Sensoren selber. Tests die am Vormittag noch reibungslos funktioniert haben, weil die Sonne noch nicht geschienen hat, wiesen ab Mittag fast immer Fehler auf. Zum Beispiel wurden gelbe Punkte als weiß erkannt und blau als schwarz.

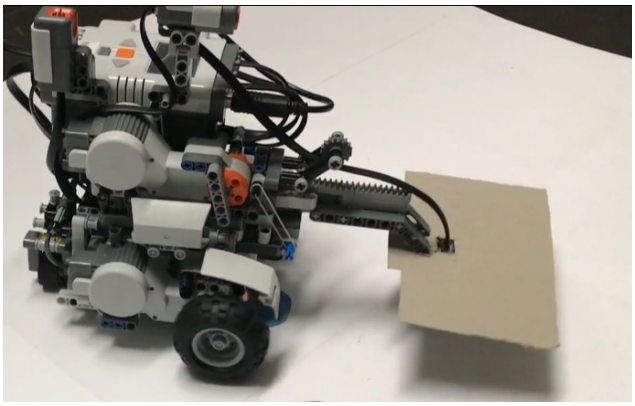


Bild 3: Spotfinder mit Verdeck am vorderen Sensor

Eine relativ simple und einfache Lösung dafür war ein Verdeck aus Pappe, die für einen gleichmäßigen Lichteinfall sorgte. Eine weitere Maßnahme die bei den Tests sehr geholfen die Messergebnisse zu verbessern war die Verwendung von möglichst hellen und nicht reflektierenden Farben.
Ein weitere Komplikation die während Versuchen deutlich wurde, ist die Prozessorleistung des NXT Computers selber. Als probiert wurde einige Prozesse bei den Tests zu beschleunigen um alles möglichst flüssig aussehen zu lassen, wie
z.B. den Roboter nach dem scannen des schwarzen Punktes schneller zum Startpunk zurück zu navigieren war nicht möglich.
Der NXT konnte die Masse an Informationen nicht so schnell verarbeiten und gab Fehlermeldungen aus. Deshalb wurden nach bestimmten Bewegungen immer kurze Pausen einprogrammiert, um den Spotfinder die Zeit zu geben seine Aufgaben ohne Probleme auszuführen.
Insgesamt gab es aber keine Probleme die nicht behoben werden konnten. Und als Ergebnis ist zu sagen das der Roboter allen vorab gestellten Erwartungen gerecht wurde.

Das Fazit für den Roboter als auch die zwei Projektwochen ist durchweg positiv. Nach nur zwei Wochen war man in der Lage einen eigenen Roboter zu entwerfen und mit Hilfe der Software Matlab zu programmieren. In dem Fall ein Roboter der sich auf einer Fläche mittels farbiger Punkte, selbstständig fortbewegen konnte und in der Lage war seinen abgefahrenen Weg wieder zurückzulegen. Für zukünftige Verbesserungen/ Optimierungen könnte man mehr Suchradien einführen oder den Mechanismus komplett erneuern, sodass sich der Spotfinder dauerhaft dreht und dabei seinen Arm ein- und ausfährt.

Zusätzlich könnte man noch ein GUI (guided user interface) erstellen, mit welchem der Benutzer selber und immer kurz vor dem durchlaufen des Programms einer Farbe einen jeweiligen Befehl zuordnen kann, um dies um zusetzten waren zwei Wochen jedoch zu kurz. Ein persönlicher Verbesserungsvorschlag wäre vielleicht eine dritte Woche hinzuzufügen, da man allein eine Woche braucht um sich mit Matlab vertraut zu machen und sich so noch viel mehr Möglichkeiten bieten. Abgesehen davon war es eine sehr schöne, interessante, aber auch anspruchsvolle Erfahrung, die es jedoch absolut Wert war.

Spotfinder - Entwicklung eines Lego Roboters zur autonomen Routensuche

Lasse Kramer, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Führerlose Transportfahrzeuge nutzen zur Navigation fast ausnahmslos das Prinzip der Spurführung entlang einer Leitlinie. Niedrige Kosten sowie Robustheit und Zuverlässigkeit sind prägnante Vorteile. Da die Leitlinien fest auf oder in den Boden aufgetragen bzw. eingelassen sind, ist eine Änderung dieser nicht spontan möglich. Im Rahmen des Projektseminar Elektrotechnik/Informationstechnik (LEGO Mindstorms) der Otto-von-Guericke Universität Magdeburg wurde ein Roboter entwickelt, der eine neue Art der Navigation nutzen soll. In diesem Paper werden die Idee, konstruktive Umsetzung und der Programmablauf genauer erläutert. Nachfolgend wird auch auf die in den Erprobungen aufgetretenen Probleme eingegangen und beschrieben wie diese gelöst worden. So entstand ein Roboter, der in bis zu drei Suchradien seine direkte Umgebung nach farbigen Punkten scannt. Gefundene Markierungen werden angefahren und dann farbabhängige Aktionen ausgeführt. Zwei Abbruchkriterien führen dazu, dass der Roboter zurück zum Ausgangspunkt fährt. Es wird außerdem eine Einschätzung darüber gegeben, in wie weit der entwickelte Roboter, bzw. die neue Art der Navigation Einsatz in der Industrie finden kann.**

***Schlagwörter*—Autonom, Farbmakierungen, FTF, Lego-Mindstorms, Matlab, Navigation**

I. Einleitung

AUTONOM arbeitende Roboter haben längst den Weg in Haushalte der ganzen Welt gefunden. Sie nehmen Menschen zeitintensive oder meist langweilige Arbeiten wie Saugen, Wischen oder Rasenmähen ab. Aber noch viel länger finden Roboter Einsatz in der Industrie. Hier übernehmen sie Arbeitsschritte die hohe und immer gleichbleibende Qualität voraussetzen, wie zum Beispiel das Schweißen oder Lackieren von Fahrzeugkarossen. Es treten aber auch monotone Tätigkeiten auf, die Roboter zuverlässig abarbeiten können. Hierzu zählt beispielsweise das Transportieren von Gütern innerhalb einer Fabrik und das anschließende Ausführen einer Aktion wie dem Einscannen von Barcodes zur Lagerquittierung oder dem Senden einer Bestandsmenge. Um Roboter Wege zurück legen zu lassen gibt es bereits Lösungen, auf die im Folgenden eingegangen wird. Diese bieten es aber nicht an, die Route zu ändern. Der Roboter legt demnach immer denselben Weg zurück. Es kann aber durchaus von Bedeutung sein die Route zu variieren. So ist es möglich Lagerorte flexibler zu gestalten oder auf auftretende Aufgaben mit hoher Priorität, z.B. die Überprüfung eines Bestands, spontan zu reagieren. Im Rahmen des Projektseminar Elektrotechnik/Informationstechnik (LEGO Mindstorms) vom 11.02.2019 bis 22.02.2019 an der Otto-von-Guericke Universität Magdeburg wurde sich diesem Problem angenommen und der Spotfinder entwickelt.

DOI: 10.24352/UB.OVGU-2019-047

Ziel war es, einen Roboter zu entwickeln der durch Rotation um die eigene Achse den Boden in unmittelbarer Umgebung nach Farbmarkierungen scannt. Durch einen ausfahrbaren Farbsensor kann der Suchradius schrittweise bis zu zweimal vergrößert werden. Wird eine Farbe erkannt, fährt der Roboter diesen Punkt an und führt eine farbabhängige Aktion durch. Die Anordnung der Markierungen kann dabei nach jedem oder sogar während eines laufenden Durchgangs verändert werden. Es soll zwei Abbruchkriterien geben, zum einen der Fall, dass kein weiterer Punkt gefunden wird, zum anderen, dass der gefundene Punkt die Farbe Schwarz hat. Beim Eintreten einer dieser Fälle soll der Roboter ohne erneut scannen zu müssen, nur anhand abgespeicherter Drehwinkel und Strecken, autonom zurück zum Ausgangspunkt fahren.

II. Vorbetrachtungen

Die Art des Spotfinders eine Route abzufahren würde wohl am ehesten bei sogenannten fahrerlosen Transportfahrzeugen (FTF) eingesetzt werden. Diese automatisierten und flurgebundenen Fördermittel werden meist in der Industrie zum Materialtransport eingesetzt. Eine einfache Möglichkeit des Fahrwerkaufbaus wird in Abbildung 1 gezeigt. Zwei starre Räder dienen zum Abstützen, das dritte Rad ist schwenkbar und wird in den meisten Fällen per Elektromotor angetrieben. So kann das FTF vor- sowie rückwärts fahren und ist linienbeweglich. Die Lastaufnahme findet je nach Anwendungszweck auf viele unterschiedliche Möglichkeiten statt und wird hier nicht weiter erläutert, da bereits Lösungen für die komplexesten Anforderungen existieren. Zu erwähnen ist nur, dass die Tragfähigkeit von einigen Kilogramm bis zu 50 Tonnen betragen kann, übliche Geschwindigkeiten liegen im Bereich von einem Meter pro Sekunde und die Einsatzdauer kann theoretisch 24 Stunden betragen. Zur Navigation wurden bereits diverse Verfahren entwickelt. Als robust und preiswert stellte sich dabei die Spurführung mittels kontinuierlicher Linie heraus. Zwei Arten werden im Folgenden genauer betrachtet. [1]

A. Induktive Leitlinie

Hierbei wird in den Boden eine von Wechselstrom durchflossene Drahtschleife verlegt, diese gibt die Fahrspur vor. Das Fahrzeug wird mit induktiven Sensoren ausgerüstet, welche dazu dienen, das wechselnde Magnetfeld zu erkennen. Ein Rechner an Bord des FTF übernimmt die Steuerung des Antriebsrades entlang dieser Leitlinie. Durch Referenzpunkte, z.B. Magnete oder Metallstücke entlang der Fahrspur, können Fehler in der Positionsbestimmung genullt werden und eine Genauigkeit von bis zu zwei Millimetern erreicht werden. [1]

B. Optische Leitlinie

Anstatt der induktiven Sensoren kommen bei dieser Art der Navigation wahlweise Kameras, Farb- oder Lichtsensoren zum Einsatz. Diese sollen eine auf dem Fußboden aufgetragene, farbige Leitlinie erkennen. Die Regelung funktioniert danach analog. Referenzpunkte können zum Besipiel durch quer zur Fahrbahn aufgetragene Farbmarkierungen gekennzeichnet werden. Hierbei können ebenso Genauigkeiten von zwei Millimetern erreicht werden. [1]

Auch im Projektseminar Lego-Mindstorms wurde sich dieser Technik schon bedient, dazu ein Verweis auf ein Projekt aus dem letzten Jahr [2].

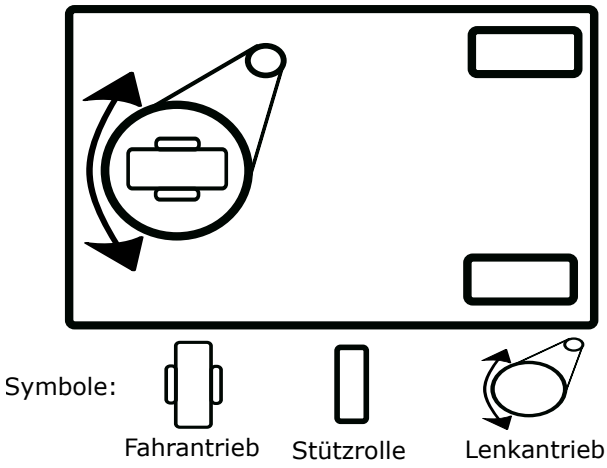


Abbildung 1. Eine Möglichkeit des Fahrwerkaufbaus eines fahrerlosen Transportfahrzeugs [1]

III. Hauptteil

Wie eingangs erwähnt sind die aktuellen Möglichkeiten der Navigation beschränkt auf feste Fahrspuren. Im Folgenden wird erklärt, wie der Spotfinder aufgebaut und programmiert wurde um eine flexible Routensuche zu ermöglichen. Der Aufbau wurde mit Hilfe von Lego-Mindstorms umgesetzt, der dazugehörige NXT-Brick wurde per Matlab und einer von der RWTH Aachen entwickelten Toolbox programmiert.

A. Konstruktion

Im Fokus standen eine möglichst kompakte Bauweise, die aber genug Stabilität beim Rotieren bietet. Auch auf eine ordentliche Führung der Kabel zu den Sensoren und Motoren wurde geachtet. Der Roboter steht auf drei Rädern, wobei zwei von ihnen separat mit Motoren angetrieben werden können um eine Rotation um die eigene, vertikale Achse zu ermöglichen. Das dritte, kleine Rad am hinteren Ende dient lediglich zum Abstützen und ist frei rotierbar gelagert, so dass es alle Bewegungen des Roboters zulässt. Ein weiterer Motor ermöglicht es einen Arm auszufahren, an dem sich ein Farbsensor befindet der den Boden scannt. Der Sensorarm wird durch Schienen horizontal und durch einen Spannmechanismus vertikal geführt. Ein zweiter Farbsensor befindet sich mittig zwischen den angetriebenen Rädern im Drehpunkt des Roboters.

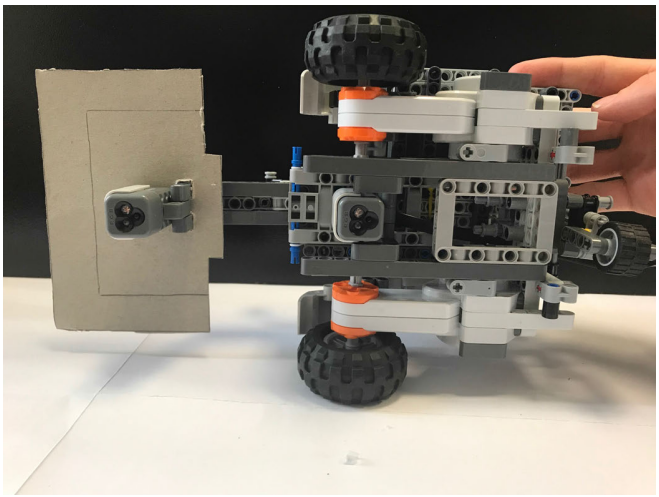


Abbildung 2. Unterseite des Spotfinders mit beiden Farbsensoren, den Antriebsrädern und dem Stützrad

Beide Farbsensoren sind in Abbildung 2 zu sehen. Außerdem wurden ein Ultraschall- und ein Lichtsensor verbaut. Diese dienen der farbabhängigen Ausführung von Aktionen (Tabelle I). Abbildung 4 zeigt den fertigen Aufbau des Spotfinders.

B. Programmablauf

Zu Beginn jedes Programmstarts befindet sich der Sensorarm im eingefahrenen Zustand. Der Roboter dreht sich um seine eigene Achse und stoppt sobald eine farbige Markierung auf dem Boden gefunden wurde oder eine volle Umdrehung stattgefunden hat. Da der zuletzt angefahrene Punkt zwangsläufig beim nächsten Scan gefunden wird muss immer überprüft werden ob die gefundene Markierung bereits angefahren wurde. Hierzu werden die Motordrehwinkel ausgewertet um somit die relative Drehung des Roboters festzustellen. Wurde eine Markierung bei 180° Roboterdrehung gefunden, handelt es sich um den zuletzt angefahrenen Punkt, dieser wird ignoriert und der Scan läuft weiter.

Wurde keine Markierung gefunden wird überprüft in welcher Position sich der Sensorarm befindet. Bis zu zweimal kann der Arm ausgefahren werden um den Suchradius zu vergrößern. Ist eine gültige Markierung gefunden, werden die Motordrehwinkel in eine Matrix an Stelle $i \times 1$ (i ist der Zähler der aktuell gefundenen Spots) geschrieben. Die Drehwinkel werden auf null zurückgesetzt und der Arm in Ausgangsposition gefahren. Jetzt fährt der Roboter geradeaus bis der Sensor auf der Drehachse über dem Farbpunkt steht. Der zurück gelegte Weg wird in die Matrix an Stelle $i \times 2$ geschrieben. So steht fortlaufend in den Zeilen der Matrix die Drehwinkel und Strecken aller gefunden Markierungen. Folgende Matrix ist beim Dreh des Demovideos (siehe Anhang) für die Abbschluspräsentation im Rahmen des Projektseminars entstanden.

$$\begin{vmatrix} 40 & 258 \\ 151 & 255 \\ 63 & 333 \\ 146 & 239 \\ 158 & 255 \end{vmatrix}$$

Anhand dieser Matrix fährt der Roboter später zum Ausgangspunkt zurück. Auch jetzt werden die Zähler der Motoren zurückgesetzt.

Der Farbsensor zwischen den Antriebsrädern kann jetzt die Farbe scannen und die zugehörige Aktion wird ausgeführt. Alle Aktionen in Abhängigkeit der gescannten Farbe können Tabelle I entnommen werden. Die entsprechenden Texte werden dem Nutzer im Command Window von Matlab ausgegeben. Abbildung 5 im Anhang zeigt ein Besipiel für die Ausgabe. Dieser Ablauf kann beliebig oft wiederholt werden, bis einer der zwei folgenden Abbruchbedingungen eintritt:

1. Die letzte, schwarze Markierung wurde erreicht.
2. Der Suchradius wurde bereits zweimal erweitert und auch beim dritten Scan konnte kein Farbpunkt gefunden werden.

Jetzt wird ein neues Programm gestartet, welches auf die Matrix zurückgreift und den Roboter autonom, ohne erneut die Markierungen scannen zu müssen zum Ausgangspunkt zurückfahren lässt. Beide Farbsensoren werden abgestellt. Zuerst dreht sich der Roboter um 180°. Beginnend mit der letzten Zeile der Matrix wird nun zuerst der in Spalte zwei gespeicherte Weg gefahren und danach dreht sich der Roboter um den in der ersten Spalte gespeicherten Winkel. Diese Schleife wird fortlaufend bis zur ersten Zeile wiederholt. Zum Schluss dreht sich der Roboter erneut um 180° und steht dann in seiner Ausgangslage.

C. *Erprobung*

Die zum Ziel gesetzte Stabilität des Spotfinders konnte nach wenigen Programmdurchläufen erzielt werden. Es zeigte sich, dass der ausfahrbare Arm in den Rotationen zu viel Spiel aufwies und so der optimale Abstand zum Untergrund nicht eingehalten werden konnte. Seitliche Schienen zur horizontalen Führung und ein Spannmechanismus, der den Arm permanent vertikal fixiert, lösten dieses Problem.

In den Erprobungen zeigte sich schnell, dass die Farbsensoren hohe Fehleranfälligkeit aufweisen. Schatten, Verschmutzungen oder Knicke im Untergrund wurden als Farbmarkierungen erkannt. Außerdem wurden dunkle Farben wie Grün oder Blau oft als Schwarz erkannt. So kam es immer wieder zu vorzeitigen Programmabbrüchen. Eine Ursache dafür wurde in dem einfallendem und vom Untergrund reflektierendem Licht gesehen. Die Farbsensoren arbeiten in dem sie weißes Licht aussenden und das verbleibende reflektierte Licht auswerten. Tritt zu viel Licht aus der Umgebung in den Prozess, kommt es zu fehlerhaften Scans. Um möglichst viele Fehlerquellen auszuschließen wurde der Versuchsaufbau auf ein weißes Plakat aufgetragen und eine Umgebung gewählt, in der Licht kaum direkt einfällt um die starke Schattenbildung zu minimieren. Des Weiteren wurden die Farbmarkierungen anstatt aus reflektierendem Papier nun aus mattem Karton ausgeschnitten. Der im Drehpunkt des Spotfinders liegende Sensor scannt aufgrund der Vielzahl an umliegenden Teilen permanent im Schatten. Um gleiche Bedingungen unter den Sensoren zu schaffen wurde eine Pappe am Arm, über den dort befestigten Sensor angebracht. Diese Pappe ist in Abbildung 2 und 4 zu sehen.

All diese Maßnahmen führten dazu, dass Programmabläufe vollständig durchgeführt werden konnten. Dabei wurden

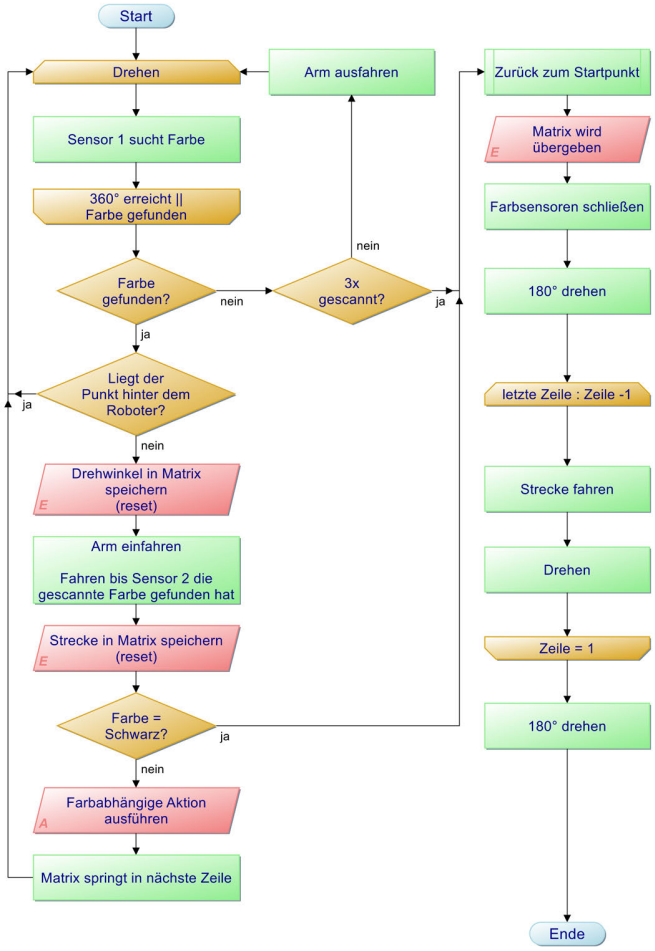


Abbildung 3. Programmablaufplan

die farbabhängigen Aktionen zuverlässig ausgeführt und die Rückkehr zum Ausgangspunkt erfolgte präzise. Durch eine Abfrage der Motorwinkel beim Drehen des Roboters ist es gelungen, den zuletzt angefahrenen Farbpunkt zu ignorieren.

Tabelle I
FARBABHÄNGIGE AKTIONEN

Farbe	Sensor	Aktion
Rot	Ultraschall	Entfernungsmessung in Fahrtrichtung
Grün		Der NXT-Brick spielt einen Ton ab
Blau		Textausgabe im Command Window von Matlab
Gelb	Licht	Helligkeitsmessung
Schwarz		Der Spotfinder fährt zum Ausgangspunkt

IV. ERGEBNISDISKUSSION

Die zu Beginn formulierten Ziele wurden erfolgreich umgesetzt. Der Spotfinder kann selbstständig Farbmarkierungen erkennen und diese anfahren. Die farbabhängigen Aktionen werden zuverlässig ausgeführt und beim Eintreten eines Abbruchkriteriums fährt der Roboter autonom zurück. In den Erprobungen zeigte sich jedoch, dass dafür ein speziell modulierter Versuchsaufbau nötig ist. Ein erfolgreicher Einsatz des Spotfinders ist also noch stark von äußeren Bedingungen

abhängig. Deutlich wurde aber, dass dafür nicht das Programm, sondern die eingesetzten Farbsensoren die Fehlerquellen verantwortlich sind. Durch präzisere Modelle könnten Erfolge auch in weniger anspruchsvollen Umgebungen erzielt werden.

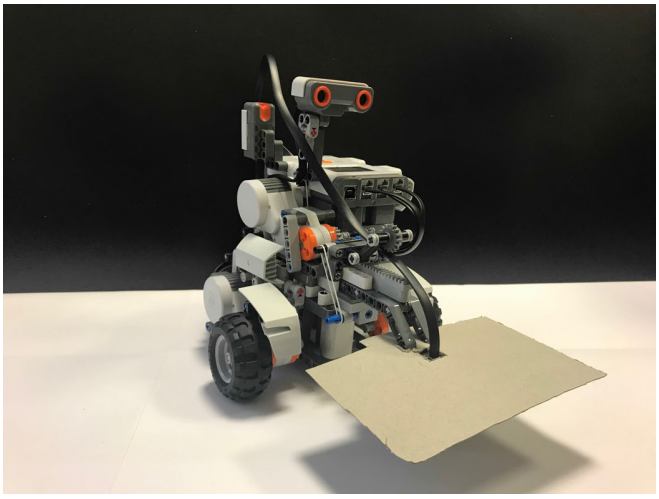


Abbildung 4. Ergebnis der zweiwöchigen Entwicklung: Der Spotfinder

V. Zusammenfassung und Fazit

Im Rahmen des Projektseminars ist es gelungen eine neue Art der Navigation und autonomen Routensuche mittels Lego-Mindstorms umzusetzen. Es ist ein Roboter entwickelt worden, der autonom eine Route aus farbigen Punkten abfährt, in Abhängigkeit der gescannten Farbe die spezifische Aktion ausführt und beim Eintreten einer Abbruchbedingung zuverlässig zum Ausgangspunkt zurückkehrt.

Navigationsarten wie die Spurführungen nach induktiver oder optischer Leitlinie wird der Spotfinder aber nicht ablösen können. Die hohe Fehlerrate der Farbsensoren trotz konstruktiver Maßnahmen wäre in der Praxis wohl ein Ausschlusskriterium im Vergleich zur sehr genauen Spurführung. Die laborähnliche Umgebung, die für die reibungslosen Probedurchläufe geschaffen werden musste, ist in der Praxis nicht umsetzbar. Außerdem zeigte sich, dass durch die Vielzahl an Scans der Programmablauf sehr viel Zeit kosten kann, Zeit, die in der Industrie nicht vorhanden sein wird. Dennoch hat der Spotfinder durchaus prägnante Vorteile. Es ist ohne Probleme möglich, die Route nach oder sogar während eines Programmablaufs zu verändern. Dazu müssen nur die farbigen Markierungen verschoben werden. Des Weitern ist die kompakte Bauweise zu nennen. Trotz drei Motoren und vier Sensoren ist der Spotfinder sehr klein und kann problemlos portabel eingesetzt werden. Sinnvoll wäre es sicherlich den Ultraschall- und Lichtsensor zu ersetzen. Barcodescanner, GPS-Sender oder Kameras wären Beispiele für anwendungsorientierte Lösungen. Lego allein bietet hier nicht genug Auswahl, Matlab hingegen lässt es zu zum Beispiel Webcams einzubinden.

Es muss das Ziel sein den Spotfinder weiterzuentwickeln. Fraglich ist ob drei Suchradien ausreichen, mit mehr Scans könnte ausnahmslos die gesamte umliegende Fläche nach Markierungen überprüft werden. Auch der maximale Suchradius kann je nach Anwendung durch einen Arm anderer länge angepasst werden. Eine Idee, die aus Zeitmangel während des Projektseminars nicht umgesetzt wurde, ist das Erstellen einer grafischen Benutzeroberfläche (Abk. GUI von englisch graphical user interface). Mittels einer GUI könnte der Nutzer des Spotfinders vor jedem Programmablauf per drop-down Menü den Farben Aktionen zuordnen. Eine im Programmcode festgelegte Zuordnung würde so umgangen werden. Des Weiteren könnte in der GUI ein Textfeld eingebunden werden, in dem die Ausgabe stattfindet, dies geschieht bis jetzt noch im Command Window von Matlab.

Anhang

Im Rahmen des Projektseminars wurde dieses Video gedreht, es zeigt einen kompletten Ablauf des Spotfinders: https://www.youtube.com/watch?v=JccdCNdJkYo. [3]

```
Helligkeit:
        468

Entfernung nach vorne in cm:
        89

Blue like the ocean
Black Spot found, FINISH!!!
Ich fahre nach Hause, das ist meine Route:
        40      258
        151     255
        63      333
        146     239
        158     255

Ich bin zurueck!
```

Abbildung 5. Text des Command Window aus Matlab nach dem Programmdurchlauf aus dem Demovideo

Literaturverzeichnis

[1] Wikipedia, the free encyclopedia: *Fahrerlose Transportfahrzeuge*. https://de.wikipedia.org/wiki/Fahrerloses_Transportfahrzeug. Version: März 2019

[2] Sattler, Bennet: *Bau einer autonomen Räumraupe*. http://journals.ub.uni-magdeburg.de/ubjournals/index.php/LEGO/article/view/552/527. Version: Februar 2018

[3] Magdowski,Mathias: *Spotfinder aus dem Lego-Praktikum 2019 an der Otto-von-Guericke-Universität Magdeburg*. https://www.youtube.com/watch?v=JccdCNdJkYo. Version: Februar 2019

Autonom fahrendes Vehikel - Bericht zum Projekt des Lego Mindstorms Seminars

Jan Adamczyk, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Die nächste Generation des Fahrens nach den Elektrofahrzeugen gehört dem autonomen Fahren. So sagte einst: Elon Musk, der Pionier des autonomen Fahrens: „In 20 Jahren wird sich der Besitz eines Autos, das nicht autonom fährt, anfühlen wie heute der Besitz eines Pferdes.“ [1] Immer mehr große Automobilunternehmen forschen und investieren in diesem Bereich. Da dieses Thema in der Gesellschaft derzeit heiß diskutiert wird und es immer noch Bedarf in Sachen Forschung und Sicherheit gibt, beschäftigt sich dieses Projekt mit dem selbständigen Fahren am Vorbild des „EZ10“ der Deutschen Bahn. Heutige Entwicklungen, im Bereich autonomes Fahren, zeigen einen neuen Trend auf. Unternehmen setzen darauf die modernen Autos mit Assistenz-Systemen auszustatten, wie zum Beispiel dem Lenk-Assistenten, um den Straßenverkehr sicherer zu gestalten, gleichzeitig aber auch dem Menschen die Fahrt so angenehm wie möglich zu machen. Des Weiteren kommt die priorisierte Entwicklung der Elektrofahrzeuge für ein umweltbewusstes Fahren dazu. Mit „Lego Mindstorms“ wird im folgenden Beitrag ein Einstieg in das breite Spektrum der Elektromobilität beschrieben. Ziel dieses Projektes ist: Grundlagen zu erarbeiten im Rahmen der Möglichkeiten von „Lego NXT“ mit geringer und kostengünstiger Hardware-Ausstattung. Dabei werden wichtige Funktionen des Roboters beschrieben und erklärt. Außerdem werden wesentliche Ergebnisse dargestellt und diskutiert. Anschließend wird ein Ausblick über mögliche Entwicklungen in der Zukunft und Verbesserungen gegeben. Bei dem Lego Mindstorms Projekt wurden zuerst Grundkenntnisse in Matlab vermittelt. Dies geschah in den ersten drei Tagen. Anschließend wurde eine Idee eines Roboters entwickelt und vor der Seminargruppe vorgestellt in Form eines Kurzvortrages mit Powerpointpräsentation. In den darauf folgenden Tagen ist durch Arbeitsaufteilung ein Roboter gebaut und programmiert wurden. Dabei ist der Roboter durch viele Testläufe optimiert und Überarbeitungen im Quellcode angepasst wurden. Schließlich kam ein funktionsfähiger Roboter heraus, der den Projektleitern und der Seminargruppe präsentiert werden konnte.**

***Schlagwörter*—Assistenz-Systeme, Autonomes Fahren, Umweltbewusstes Fahren**

I. Motivation des Projektes

DIE Sicherheit im Straßenverkehr ist ein wichtiges Thema. Jährlich kommt es zu zahlreichen Verkehrsunfällen durch menschliches Versagen in Deutschland. Laut einer Studie von Statista[2] sterben „circa 3300 Menschen“ bei solchen Unfällen. „Im Jahr 2018 betrug die Zahl der Toten 3249. Das sind 270,75 Menschenleben pro Monat.“ Deshalb besteht Bedarf weiter im Bereich autonomes Fahren zu forschen und zu entwickeln, damit die Zahl der tödlichen Unfälle verringert werden kann. Der Einsatz eines autonomen Roboters kann durch viele Sensoren, Kameras, Algorithmen und Daten über die Verkehrslage zuverlässiger sein, als ein Mensch, weil ein Roboter Gefahren früher erkennen und darauf entsprechen reagieren kann. Somit schützt er Menschenleben aller Verkehrsbeteiligten[4]. In dem Projekt „Lego Mindstorms“ wurde ein Roboter gebaut, der sich autonom mithilfe seiner Sensoren frei über eine Oberfläche bewegen kann. Er findet Kanten und berücksichtigt diese bei seinem Fahrverhalten. Außerdem ist die Konstruktion in der Lage über eine Brücke zu fahren und sich aus einer Ecke zu befreien. Dabei passieren hin und wieder kleinere Fehler. Die Programmierung des Roboters erfolgte über Matlab unter Verwendung des „Lego-NXT-Bausteines“. Um auf die Sensoreingänge und Motorausgänge des NXT zugreifen zu können, hat die RWTH[RWTH01] Aachen Universität eine Verknüpfung zwischen Matlab und NXT zur Verfügung gestellt. Die Fortbewegung wird mit zwei Lichtsensoren und zwei Motoren realisiert. Durch das Auslesen der Sensoren, die vorn an dem Roboter angebracht wurden, ermittelt ein Algorithmus die Geschwindigkeit und durch gezieltes Ansteuern der beiden Motoren findet eine Lenkbewegung in die gewünschte Richtung statt.

DOI: 10.24352/UB.OVGU-2019-048

II. Allgemeine Informationen

Es existieren bereits autonome Fahrzeuge mit Assistenz-Systemen in Deutschland. Hier sind ein paar Beispiele zur Verdeutlichung:

A. Stand der Technik in Deutschland

Das Bundesforschungsministerium fördert seit 2015 intelligente Mobilität Projekte mit über 100 Millionen Euro[6]. Zudem gibt es seit 2018 14 Teststrecken für autonomes Fahren in Deutschland[5]. Da wären zum Beispiel das Digitale Testfeld Autobahn (DTA) auf der A9 zwischen München und Nürnberg, sowie das Testfeld Hamburg für automatisiertes Fahren in städtischen Raum.
Darüber hinaus heißt es laut einer Studie des Instituts der deutschen Wirtschaft (IW), das mindestens 52 Prozent aller Patente für autonomes Fahren auf der Welt aus Deutschland stammen. Weltweit führend sind die drei deutschen Unternehmen Bosch, Audi und Continental, aber auch Mercedes und VW investieren immer mehr in die autonome Mobilität und sammeln Daten. Es existieren rund 1000 Testfahrzeuge auf deutschen Straßen, die laut Gesetz bis 10 Kilometer pro Stunde selbstständig fahren dürfen und zum Beispiel selbständig Einparken. Für schnellere Fahrten muss ein Mensch am Steuer sitzen, der den Computer überwacht und bei eventuell auftretenden Problemen oder Gefahren eingreifen muss.

B. Konkretes Beispiel EZ10

Im Testfeld Bad Birnbach (Bayern) ist seit Oktober 2017 der erste autonom fahrende Mini-Bus EZ10 der Deutschen Bahn im Einsatz. Der Mini-Bus bietet Platz für sechs Personen und befördert sie von Bad Birnbach zu dem zwei Kilometer außerhalb liegenden Bahnhof. Er wird elektrisch betrieben, dabei kommt EZ10 vollkommen ohne Lenkrad und Pedale aus. Lediglich ein Fahrbegleiter ist mit an Bord, der bei Gefahr oder technischen Versagen eingreifen muss. Allerdings ist die Personenbeförderung bisher auf zehn Kilometer pro Stunde begrenzt. In Zukunft soll nach und nach die Geschwindigkeit für autonomes Fahren angehoben werden. So liegt ein Gesetzesentwurf für 25 Kilometer pro Stunde bereits vor, aber mit hohen Anforderungen. Dieses Beispiel ähnelt unserem Lego-Roboter[7].

III. Konstruktion und Funktionsweise des Roboters

Der autonom fahrende Lego-Roboter sieht wie folgt aus (siehe Abbildung 1).

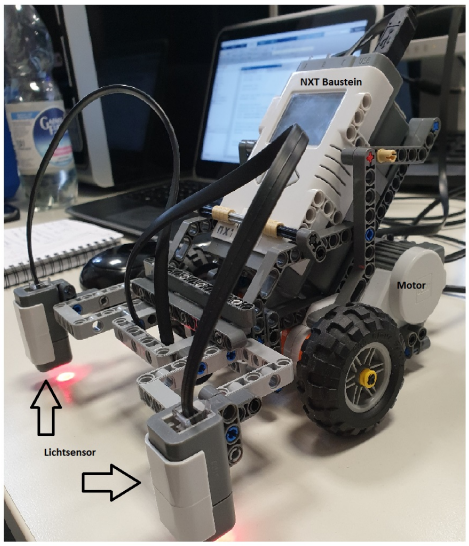


Abbildung 1. Lego Roboter

Der erste Roboter wurde nach einem Beispiel aus dem Lego-Mindstroms Baukasten nach gebaut. Man stellte schnell fest, dass der Grundaufbau des dreirädrigen Vehikels ausreichend für unser Vorhaben war, aber der Aufbau modifiziert werden musste. So fiel der Entschluss, die Aufhängung für die Räder in Kombination mit den zwei Motoren, die jeweils links und rechts an der Achse zum Rad befestigt wurden, beizubehalten. Ein drittes, kleineres Rad befindet sich hinten, welches frei beweglich ist und so das Fahren in Kurven vereinfacht. Anschließend wurde an das Chassis, eine vorn überstehende Konstruktion angefügt, um die zwei Lichtsensoren in Position zu bringen. Nach einigen Versuchen musste die Position korrigiert werden, da der Lenkradius zu groß war und dadurch gelegentlich Abstürze von der Tischplatte erfolgten. Zusätzlich hat eine Kippfunktion der Lichtsensoren nach vorn für mehr Spielraum und früheres Erkennen der Kanten gesorgt. Der NXT-Baustein ist zentral über der Hauptachse angebracht, um eine optimale Gewichtsverteilung zu garantieren. Für den Fall, dass der Roboter rückwärts fahren muss, hat er einen Ultraschallsensor hinter dem NXT erhalten. Diese Modifikation war nötig, damit der Roboter nicht wie bei früheren Versuchen beliebig weit nach hinten fuhr und dadurch teilweise von der Tischkante fiel oder an Gegenstände stieß. Zum Starten und Beenden des Roboters ist ein Tastsensor verwendet worden, der durch einmaliges Drücken den Roboter in einen Start-Modus versetzt. Drückt man den Taster erneut, kalibriert der Roboter seine Sensoren auf die Unterlage und fährt anschließend los.

A. Algorithmus zur Fahrroutine:

Die Fahrroutine wird mit einer while-Schleife realisiert. Hierbei werden die Quellcodes in eckigen Klammern angegeben. (Es handelt sich bei dem Zeichen: „-“ um einen Bindestrich.)
[while ((d-delta-light$_1 < 500)||(d - delta - light_2 < 500))$]
(Für Werte unter 500 bleibt der Roboter in der Schleife und fährt.)
Falls ein Sprung zwischen zwei Werten über 500 auftritt, führt das zum Abbruch der Schleife und der Roboter bleibt stehen. Dieses Ereignis tritt bei einer Kante oder plötzlich auftretenden Lichtwechsel ein. Bleibt die Änderungsrate zwischen den Werten unterhalb der Grenze von 500, so läuft der Algorithmus weiter.
[Kalibriere Linken (L$_1$)$SensormitRechten(L_2)Sensor$]
durch Mittlung von 500 Einzelmessungen
[lightValCount=500]
[L$_1 cal = mean(getlight(Sensor_1, lightValCount)$]
[L$_2 cal = mean(getlight(Sensor_2, lightValCount)$]
Bei der Kalibrierung speichert der NXT ein Delta aus 500 Messwerten, das man als Vergleichswert nimmt. Dabei wird eine 1x2 Matrix (mit Startwert [0 0]) angelegt, in der die aktuell aufgenommenen Sensor-Lichtwerte in Bezug auf den Kalibrierungswert gespeichert werden. Später wird die absolute Helligkeitsänderung (Betrag) gespeichert. Ein Delta-Max wird ebenfalls berechnet - es dient als Hilfsgröße zur Bestimmung der „Motorpower“. Die „Motorpower“ wird normiert und mithilfe einer Konstante berechnet. Die Konstante wurde empirisch ermittelt, auf den Wert „30„ festgelegt und somit das gewünschte Ziel einer flüssigen und konstanten Fahrt erzielt.
[const=30]
[normA=(Delta-Max-delta-light$_2(2))/L_1 cal;$]
[normB=(Delta-Max-delta-light$_1(2))/L_2 cal;$]
Der oben genannte Ausdruck normiert den jeweils gegenüberliegenden Sensorwert auf den Kalibrierungswert. Dies ist wichtig, da die maximale Motorleistung nicht über einen Wert von +100 bzw. -100 (in diesem Fall handelt es sich um ein Minus) liegen kann. Mit dem folgenden Ausdruck wird die endgültige Motorpower berechnet:
[PowerA = int8((normA*100) + const);]
[PowerB = int8((normB*100) + const);]

Die berechnete Leistung wird auf den jeweiligen Motor übertragen, wodurch sich nachfolgende Szenarien ergeben:

Tabelle I
VERSCHIEDENE SZENARIEN

$\Delta Sensorlinks$	$\Delta Sensorrechts$	Status	Vorgang
< 100	< 100	Oberfläche	vorwärts Fahren (beide Motoren)
> 200	< 100	Kante links	Rechter Motor
< 100	> 200	Kante rechts	Linker Motor
> 200	> 200	Kante vorn	zurück Fahren (beide Motoren)

Hierbei ist zu beachten, dass der linke Sensor den rechten Motor bzw. der rechte Sensor den linken Motor steuert. Der Lichtsensor erkennt Grauwerte, das heißt die Farben weiß bis schwarz. Wenn die Sensoren weiß erkennen, sind die Delta-Werte der Sensoren unter dem Wert 100 und dementsprechend steuern sie die Leistung zu den Motoren für das Vorwärtsfahren. Wird der Delta-Wert größer, erkennt der Sensor grau oder schwarz. Daraus folgt, das der Sensor eine Kante gefunden hat und die Leistung am Motor reduziert werden muss, damit eine Kurve gefahren wird. Ein Beispiel: Erkennen die Sensoren weiß, befindet sich der Roboter auf der Oberfläche und fährt gerade aus. Kommt es zu einer Änderung des Delta-Wertes beim linken Sensor, reduziert der Algorithmus die Motorleistung beim rechten Motor, wodurch der linke Motor konstante Power hat und eine Kurve nach rechts gefahren wird. Dabei kann der Motor auch rückwärts laufen, umso stärker wird die Lenkbewegung. Auf diese Weise reagiert der Roboter auf Kanten. Für den Fall das beide Lichtsensoren schwarz erkennt stoppt der Roboter sofort alle Motoren und hält an. Dies tritt an Ecken oder Kanten auf, bei denen der Schwellwert von 500 überschritten wird. Anschließend setzt er zurück und nimmt wieder Fahrt auf, wenn mindestens ein Sensor weiß erkennt. Der Roboter fährt solange bis man den Taster erneut drückt. Dann befindet sich der Roboter wieder im Start-Modus und kann erneut gestartet werden. Hierbei ist es möglich die Messwerte am Computer aus den Matrizen abzurufen und einzusehen, um weitere Analysen durch zuführen. Durch Drücken von etwa zwei Sekunden, wird ein Ton vom NXT abgespielt und der Roboter schaltet sich aus.

IV. ERGEBNISDISKUSSION

Das Ergebnis ist ein autonom fahrendes Vehikel, das sich frei auf einer Oberfläche bewegen und aus Ecken befreien kann. Wird der Roboter auf einen Tisch gesetzt, so kann der Roboter durch seine Programmierung nicht herunterfallen und damit auch eine Brücke überwinden. Mit diesem Projekt ist es gelungen einen Einstieg in die Elektromobilität im Rahmen der Möglichkeiten von Lego-Mindstroms zu finden. Die ursprüngliche Idee einen Roboter zu bauen, der autonom eine Unterlage abfährt, vermisst und anschließend eine Kurve auf die Unterlage zeichnet, musste früh verworfen werden. Grund dafür war der daraus resultierende komplexe Vorgang, der den zeitlichen Rahmen gesprengt hätte. Die Fokussierung auf das selbständige Fahren ohne Zwischenfälle reduzierte die Komplexität des Projektes. Dennoch ist der Roboter nicht perfekt. In zehn Prozent aller Fälle kam es zu einem Zwischenfall bei dem z.B. ein Rad über einer Kante stand. Durch Optimierungen im Algorithmus könnte dieses verhindert werden. Eine andere Möglichkeit wäre die Konstruktion zu ändern, sodass es keinen „toten Winkel“ mehr gibt. Obwohl der Roboter sich aus einer Ecke befreien kann, geschieht dies teilweise sehr langsam. Der Roboter benötigt viele Versuche, da er nur durch kleine Korrekturen macht, indem er sich dreht, danach kann er die Weiterfahrt aufnehmen. Bisher wurde die Situation immer gelöst, jedoch hat es den Anschein, dass sich das Fahrzeug in einer Endlosschleife befindet. Trotz dieses nicht optimalen Verhaltens funktioniert das Erkennen der Unterlage und Kanten einwandfrei. Die Tests ergaben, das Störungen der Sensoren durch ungünstige Lichtverhältnis, schwarze Linien und farblich ungleiche Unterlagen hervorgerufen werden können, die die Funktion des Roboters beeinflussen. Hindernisse sind weiterhin ein Problem, da die maximale Anzahl von vier Sensoren nicht ausreicht, um z.B. einen weiteren Ultraschallsensor zu verwenden. Mit mehr Zeit und unter Verwendung zusätzlicher Sensoren könnte man den Roboter weiter verbessern und die genannten Probleme beheben, indem man eine gewisse Toleranz gegenüber Schwankungen des Lichts und Unterlage programmiert.

V. ZUSAMMENFASSUNG UND FAZIT

Im Rahmen des Lego-Projektes ist ein autonom fahrendes Vehikel gelungen, dessen Basisfunktionen sehr realitätsnah sind. Dennoch ist dieser Roboter ungeeignet für den realen Betrieb. Grund dafür sind die begrenzten sensorischen Möglichkeiten und die Ungenauigkeiten im programmierten Algorithmus. Dennoch erfüllt der Roboter die Zielsetzungen dieses Projektes und ist ein gelungener Einstieg in den Bereich des autonomen Fahrens. Der Roboter ist in der Lage frei und autonom auf der Oberfläche zu fahren. Er erkennt Kanten und kann auf sie reagieren, damit er zum Beispiel nicht von einem Tisch herunter fällt. Situationen wie eine Brücke überwinden oder sich aus einer Ecke befreien gehören zu seinen Hauptfunktionen. Hierbei kommt es in einem von zehn Versuchen zu einem Fehler. Die Ungenauigkeiten der Lego-Bauteile führen teilweise zum Abbruch der Fahrt oder scheinbare Endlosschleifen in Ecken. Daher ist die Konstruktion noch verbesserungswürdig, indem man durch zusätzliche und leistungsstärkere Sensoren die Trägheit der Motoren reduziert und die toten Winkel vermeidet. Der Algorithmus könnte in seinen Details weiter angepasst werden. Dies war leider durch die geringe Zeit während des Projektes nicht mehr möglich. Eine realisierbare Erweiterung des autonomen Roboters ist ihn mit einem Staubsauger und Auffangbehälter auszurüsten. Damit wird der Roboter zum Staubsaugerroboter erweitert. Ein neuer Algorithmus müsste entwickelt werden, damit ein effizientes Säubern des Raumes erreicht wird. Die Ausstattung mit weiteren Sensoren für einen 360 Grad Blick könnte das Niveau des Roboters auf bereits entwickelten autonomen Staubsauger anheben. Heutzutage müsste der Roboter zusätzlich noch Internet Zugang erhalten, damit er mit Sprachsteuersystemen, wie zum Beispiel mit Alexa, kommunizieren und ferngesteuert werden kann. Eine dazugehörige App könnte dem Nutzer einen Überblick über

die Reinigungsdauer und den gesäuberten Bereich geben. Es ist wichtig sich mit dem autonomen Fahren auseinanderzusetzen, denn der technologische Fortschritt ist rasant. Bereits in ein paar Jahren sollen autonome Fahrzeuge zum Alltag gehören und den „Lifestyle“ der Städte und sogar auf dem Land verbessern. Sei es der Transport von Gütern mit selbstfahrenden LKWs oder die Beförderung von Menschen mithilfe intelligenter Busse zum gewünschten Ziel. Diese neue Innovation wird selbst im privaten Besitz eine Rolle spielen. Ein Zeichen von Luxus der künstlichen Intelligenz das Fahren zu überlassen und sich selber anderen Dingen zu Widmen.

A. *Verweis auf Literaturquellen*

[1] Teslamag,Elon Musk über die Zukunft nicht-autonomer fahrzeuge: https://teslamag.de/news/elon-musk-zukunft-fahrzeuge-als-5365, Version: 04. November 2015;

[2] DStatis, Statistisches Bundesamt zur Unfallentwicklung auf deutschen Strassen 2017 https://www.destatis.de/DE/PresseService/Presse/Pressekonferenzen/2018/verkehrsunfaelle$_2$017/$Pressebroschuere\ unfallentwicklung.pdf?-blob = publicationFile; Stand$: 2018

[3]$MDR, StandderForschungzumAutonomenFahren\ inDeutschlandhttps : //www.mdr.de/nachrichten/\ vermischtes/forschung - automatisiertes - fahren100.html, Version$: 01.$April$2018;

[4] ADAC, autonome Roboter zuverlässiger als Menschen https://www.adac.de/rund-ums-fahrzeug/autonomes-fahren/technik-vernetzung/aktuelle-technik/ Stand: 20.12.2018

[5] gdv, Teststrecken in Deutschland https://www.gdv.de/de/themen/news/diese-staedte-und-regionen-werden-2018-zu-teststrecken-25874 Stand: 02.11.2017

[6] Bundesministerium für Forschung und Bildung, Förderung automatisiertes fahren https://www.bmbf.de/de/automatisiertes-fahren-4158.html Stand: 2019

[7] Deutsche Bahn, Bad Birnbach EZ10 https://www.deutschebahn.com/de/Digitalisierung/autonomes

$_f ahren_n eu/Testfeld_B ad_B irnbach - 1206846\ (nichtmehraktuell)alternativerlink : https : //www.autonomes - fahren.de/neuerungen - beim - shuttleservice - in - bad - birnbach/STand$: 27.$August$2018[$RWTH$01]$RWTH Aachen, RWTH MindstromsNXT : https : //www.rwth - aachen.de/go/id/a/?; abgerufen$: 18.02.2019

ANHANG

Überblick über Aufbau des Roboters

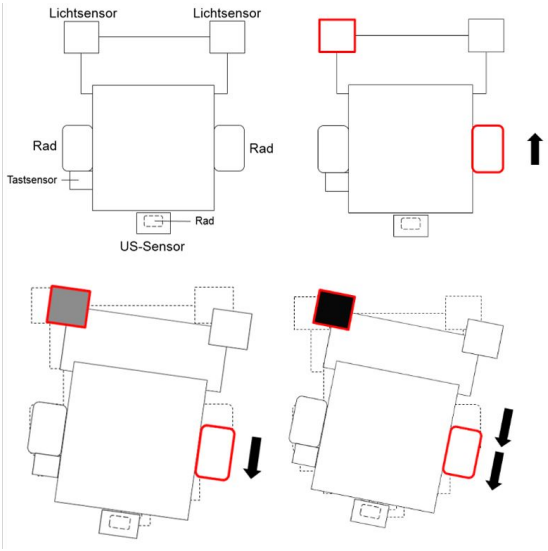


Abbildung 2. Veranschaulichung der Funktionsweise

Autonomes Fahren mit Lego Mindstorms auf Untergrund durch Erkennen von Kanten

Benjamin Mydla, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Autonome System treten in Industrie und Alltag in vielen verschiedenen Formen in Erscheinung. Oft greifen sie dabei mit einer sehr geringen Reaktionszeit in Abläufe ein, die ein Mensch nicht erreichen kann. Diese Arbeit präsentiert die Entwicklung einer linearen Steuerung eines selbstständig fahrenden Roboters. Dieser vermeidet es, eine befahrbare Oberfläche zu verlassen. Die Umsetzung erfolgt mit MATLAB auf einer Lego-Mindstorms-NXT-Hardware-Plattform unter zu Hilfenahme der RWTH - Mindstorms NXT Toolbox. Grundsätzliche Überlegungen was Geometrie des Aufbaus und Software-Umsetzung betrifft, werden erörtert und eine Lösung wird präsentiert.**

***Schlagwörter*—autonomes Fahren, Kantenerkennung ,Lego Mindstorms, lineare Steuerung**

I. Einleitung

IM Zuge steigender Nachfrage elektrischer Mobilität, dem Fortschreiten der Digitalisierung, sowie der zunehmenden Miniaturisierung - und der damit verbundenen Verbesserung der Rechenleistung - der verwendeten Hardware werden immer häufiger autonom arbeitende Systeme möglich und gewünscht. So bremsen Assistenzsysteme eigenständig, sollte ein Hindernis den Weg versperren, halten die Spur auf Fahrbahnen oder informieren über geeignete Parklücken [1]. Straßenbahnen und Züge navigieren vollautomatisch und fahrerlos [2] in fest geplanten Routen bzw. Schienensystemen in einer veränderlichen Umwelt. Selbst Verkehrsflugzeuge [3] sollen autonom ihren Weg durch den Luftraum finden. Für die Umsetzung dieser Systeme existiert eine große Bandbreite an Sensoren, sowie eine Fülle an programmiertechnischen Lösungen. So ist die Arbeit, die im Rahmen des Projektseminars Elektrotechnik/Informationstechnik (LEGO Mindstorms) im Wintersemester 2018/2019 in Zusammenarbeit mit Jan Adamczyk entstand, als ein erster Versuch zu verstehen, sich diesem umfangreichen, wie weitreichenden Thema zu widmen. Als Zielstellung gilt es ein autonom fahrendes Vehikel zu konstruieren, welches sich auf einer Unterlage bewegt und es vermeidet über die Unterlage hinauszufahren. Explizit soll der Automat auf einem Tisch fahren und nicht herunter fallen. Als Hardware dient ein LEGO-Mindstorms-Education-Paket [4], welches softwareseitig mit Mathworks MATLAB [5] in Kombination mit der Mindstorms NXT Toolbox for MATLAB der RWTH Aachen [6] programmiert wird.

II. Vorbetrachtungen

Ein Mindstorms-Projekt besteht aus zwei essentiellen Teilen, der Hardware, die als Plattform dient, und der Software, die auf dieser Plattform läuft und sie steuert.

DOI: 10.24352/UB.OVGU-2019-049

A. Die Hardware-Grundlagen

Als Grundlage dient das Lego-Mindstorms-Education-Paket, bestehend aus Lego-Technik-Bausteinen, drei Servomotoren mit eingebauten Rotations-, Ultraschall-, Licht- und Tastsensoren, Anschlusskabeln und dem NXT-Baustein. Der Baustein trägt den 32-Bit-Mikroprozessor, einen Lautsprecher, sowie die Anschlüsse für die Sensoren, Motoren und den PC. [7]

Zur Realisierung des Gefährts wird neben den Lego-Technik-Bauteilen, die das Fahrgestell und Fahrwerk bilden, dem NXT-Baustein der als Steuereinheit fungiert, auf Sensoren zurückgegriffen, die die äußeren Bedingungen detektieren.

B. Die Software-Grundlagen

Ursprünglich wird mit der von Lego bereitgestellten Programmierumgebung NXT-G grafikbasiert mit fertigen Modulen gearbeitet und die Programme werden auf die Steuerungseinheit des NXT-Bausteins geladen. Mittlerweile existieren alternative Möglichkeiten in anderen Sprachen wie, beispielsweise Java, C, LUA oder Assembler zu arbeiten [8].

In der Projektarbeit wird eine weitere Methode, den NXT anzusteuern, genutzt. Hierbei fungiert MATLAB als Fernbedienung des NXT über USB. So können Daten der an den NXT angeschlossenen Sensoren und Motoren von MATLAB empfangen und verarbeitet, sowie auszuführende Befehle an den NXT gesendet werden [6].

Für die Umsetzung des Projekts - Fahren auf einer Unterlage und Vermeidung von Kanten (Absturz des Roboters über die Kante) - sind also zwei Dinge nötig:

1) Eine Plattform, die den NXT-Baustein (, die Motoren und Sensoren trägt. Die Sensoren erfassen die Umgebung die die Bewegung in zwei Raumrichtungen gewährleistet und die Umgebung erfassen kann.
2) Ein MATLAB-Programm, welches die Sensor-Daten vom NXT entgegennimmt, und entsprechend Befehle an den NXT sendet, der die Motoren steuert.

Die Möglichkeiten der nutzbaren Sensoren sind dank MATLAB vielfältig, und lassen auch ein einfaches Einbinden von USB-Webcams zu [9]. So ist es ein Ansatz, die Umgebung mittels Visueller Erfassung aufzunehmen und auszuwerten.

C. Visuelle Erfassung der Umgebung

Dafür werden eine oder mehrere Kameras ortsfest auf dem Fahrzeug (Roboter) installiert, deren Sichtfeld jeweils so groß

sein muss, den Raum vor dem Fahrzeug in einer sinnvollen Größe zu erfassen. Damit die Fahrt inklusive Lenkmanöver, die Sicherheit beim Wenden und das Vermeiden des Absturzes von der Unterlage möglich ist. Die Dimension des Fahrzeugs, die Anzahl der Räder, die gewählte Lenkgeometrie und Antriebsart, ob direkter Antrieb je Rad oder ein Getriebe, spielen beim Spur- bzw. Wendekreis [10] eine Rolle. Diese Eigenschaften formulieren die Anforderung an die optische Erfassung, und wo diese platziert wird. So kann es von Nöten sein, zwei statt einer Kamera zu verwenden. Die Software muss aus den aufgenommenen Bilddaten Informationen gewinnen, um abschätzen zu können, wo sich eine Kante befindet und diese durch Steuerung der Motoren vermeiden. Die Tiefeninformation der Umwelt lassen sich dabei durch eine Kopplung zweier Kameras erreichen. ”Die Bestimmung einer absoluten Tiefeninformation erfordert die genaue Kenntnis der Parameter aller Komponenten des Aufbaus und der Kameras. Das die Bestimmung dieser Parameter nicht trivial ist, kann für einfache Anwendungen der Wert der Disparität selbst als einziger Indikator für die Tiefeninformation verwendet werden. Hieraus ist eine relative Entfernung/Tiefeninformation ableitbar, welche oft für die Steuerung einer Anwendung ausreicht, ... - [11]. Die Bereitstellung der Daten zur Steuerung ist demnach mit erheblichem Aufwand verbunden. Die tiefenwertbasierte Detektierung von Höhenunterschieden führt zu der Überlegung, statt die gesamten Bilddaten zu nutzen, die Information direkt aus der Umgebung unter Zuhilfenahme von Lichtsensoren abzulesen. Damit ist die aufgenommene Datenmenge und der Rechenaufwand wesentlich geringer und nebst Lego-Mindstorms keine gesonderte Hardware nötig. Ein Linienfolgen [12] nutzt dieses Konzept, das adaptierbar scheint.

D. Der Linienfolger

Ist ein Roboter, der sich in zwei Raumrichtungen bewegen kann und eine Sensorik besitzt, die zwischen Untergrund und einer aufgebrachten Linie unterscheidet und dieser folgt.

Um charakteristischere Daten erfassen zu können wird hier ein heller, meist weißer Untergrund benutzt, auf den eine schwarze Linie aufgebracht ist. Diese zwei Bereiche haben unterschiedliche Reflexionsverhalten, wobei Weiß viel und Schwarz wenig Licht reflektiert. Der Lichtsensor [13], der über eine rote LED Licht auf die Unterlage emittiert und über eine Photodiode den reflektierten Anteil (Grauwert) absorbiert, kann dafür genutzt werden diese unterschiedlichen Bereiche zu erkennen. Der Grauwert wird vom Sensor in einem Bereich von 0 bis 1023 dargestellt [14], wobei 0 keiner und 1023 einer sehr großen absorbierten Lichtintensität entspricht. Mit diesen Daten lassen sich zwei Antriebsmotoren betreiben, damit der Roboter beim Fahren auf oder in der Nähe der Linie bleibt und somit dieser folgt. Die Menge der absorbierten Photonen ist von der Position des Sensors in Bezug zur Linie abhängig. Mit diesen Informationen lässt sich eine Steuerung der Motoren zum Fahren realisieren.

Die Oberflächen einer Unterlage und deren Kante besitzen jeweils auch unterschiedliche Reflexionseigenschaften. Die Oberfläche selbst reflektiert viel, die Kante wenig Licht. Dadurch lässt sich das Prinzip des Linienfolgers an eine Kantenvermeidung anpassen.

III. HAUPTTEIL

Das grundsätzliche Konzept ist ein Roboter, der auf einem Untergrund, Beispielsweise auf einem Tisch, fährt und Kanten vermeidet, um die Oberfläche nicht zu verlassen. Er bleibt nicht stehen, sobald eine Kante erreicht wird, sondern korrigiert eigenständig die Fahrtrichtung um weiterzufahren.

A. Die Hardware

Als Antrieb dienen dabei zwei Lego-Mindstorms-NXT-Motoren, die jeweils direkt mit einem Rad verbunden sind. Die Stabilität gewährleistet ein drittes, drehbares Rad am Heck. Die schematische Darstellung des Roboters in Abbildung 1 zeigt die Räder als graue Kästen am blauen Chassis des Roboters. Um der Probleme des Traktionsverlustes eines der Räder in einer Kurvenfahrt Rechnung zu tragen, sind die zwei Lichtsensoren dabei wie in Abbildung 1 zu sehen, angebracht. Die Überlegung sieht wie folgt aus: Da sich der Drehmittelpunkt beim Differentialantrieb auf halber Strecke auf der Achse zwischen beiden Motoren befindet (Punkt M in Abbildung 1), gibt bei der gewählten Bauform das Heckrad den Radius des Spurkreises an. Der Wenderadius ist größer als der Spurradius [10] und wird als Mindestmaß verwandt. Die Sensoren müssen diesen Mindestabstand einhalten, so dass bei einer Wendung das Hinterrad nicht von der Unterlage abgleitet. Beide Sensoren sind mit einem Versatz nach außen, bezogen auf das jeweilige Antriebsrad, angebracht. Durch diesen Versatz wird eine Kante, auch wenn sich ihr unter einem spitzen Winkel genähert wird, erkannt. Was wiederum bei einem anschließenden Lenkmanöver den möglichen Verlust der Traktion des der Kante zugewandten Antriebsrad verhindert. Die Sensoren sind auf den Untergrund gerichtet und empfangen so das von der Unterlage reflektierte Licht. Der Abstand von Sensor zum Untergrund beträgt circa 10 Millimeter.

B. Die Software

Die Steuerung der Motoren beruht auf dem Helligkeitswert des reflektierten Lichts. Dabei ergeben sich die in Abbildung 2 dargestellten drei Abschnitte mit ihren zugeordneten Sensorwerten. Diese sind ein grobes Resultat verschiedener Testläufe eines Sensors. Das gewünschte Verhalten kann wie folgt beschrieben werden: Der Roboter soll ungestört vorwärts fahren solange beide Sensoren die Oberfläche detektieren. Er soll wenden, wenn ein Sensor eine Kante oder einen Abgrund misst und stehen bleiben, wenn beide gleichzeitig eine Kante oder einen Abgrund erfassen.

Der weite Bereich der Sensorwerte der einzelnen Abschnitte in Abbildung 2 resultiert aus unterschiedlichen Lichtverhältnissen während der Testphase und aus dem verschiedenen Reflexionsvermögen je nach Untergrund. Dies ist bei jedem Lauf zu berücksichtigen und erfordert notwendige Anpassungen, um für die Steuerung nutzbar zu sein.

Es ist deshalb, vor jedem Start die Fahrroutine auf die vorhandene Unterlage zu kalibrieren. Es werden dafür 50 bis 500 Einzelmessungen pro Sensor durchgeführt und die Werte gemittelt. So entsteht je ein Kalibrierungswert für den linken und rechten Sensor, der das Reflexionsverhalten der Unterlage

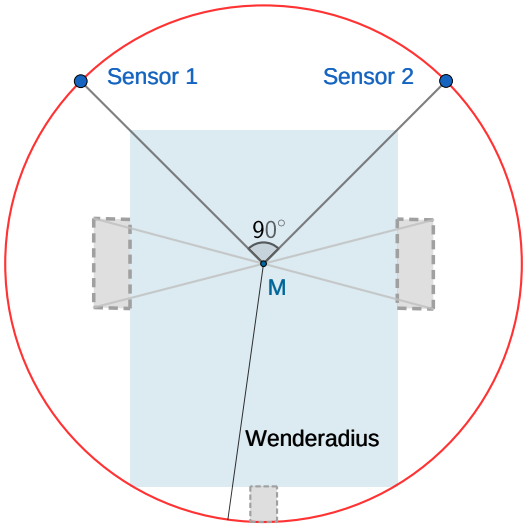


Abbildung 1. Positionierung der Sensoren aufgrund des Wenderadius

widerspiegelt. Die Unterlage muss homogen sein muss, damit das Reflexionsvermögen nicht zu stark variiert.

Für das Projekt ist dieser Umstand gegeben, so dass zur Ansteuerung der Motoren nicht die absolut gemessene Helligkeit, sondern eine berechnete Helligkeitsänderung, bezogen auf den Kalibrierungswert, genutzt wird. Siehe Gleichungen (1) und (2). Die Helligkeitsänderung ist ein Maß für die Entfernung des Sensors zum Untergrund (in Bezug auf die Ausgangslage). Mit diesen Daten ist es möglich, auftretende Höhendifferenzen durch die größeren Lichtunterschied, zu erkennen und somit Steuerungs-Signale für die Motoren zu generieren.

Die Geschwindigkeit mit der sich die Motoren drehen folgt dem Graphen aus Abbildung 3, der für den linken Motor aus der Gleichung (3) und für den rechten Motor aus Gleichung (4) folgt. Das Steuerungs-Prinzip ist dergestalt, dass der linke Sensor Einfluss auf das Drehverhalten des rechten Motors hat und der rechte Sensor auf das des Linken. Siehe dazu Abbildung 4, in der das Verhalten eines Motor-Sensor-Paares dargestellt ist.

Die Programm-Routine, wie in Abbildung 5 gezeigt, wiederholt sich nach dem Senden der Signale an die Motoren mit der Aufnahme der Sensorwerte, und läuft dann in einer Schleife, solange die berechnete Helligkeitsabweichung unter einem Maximum liegt. Das Abbruchkriterium wird erreicht, wenn einer der beiden Sensoren plötzlich einen zu starken Abfall der Helligkeit misst. Ursache dafür kann ein Hardware-Fehler, ein Störsignal (unvorhergesehene Lichtquellen) oder das plötzliche Geraten über eine Kante sein.

C. Gleichungen

$$\triangle Light_1 = |Cal_1 - Sensorwert_1| \quad (1)$$

$$\triangle Light_2 = |Cal_2 - Sensorwert_2| \quad (2)$$

$$MotorL = (\triangle Max - \triangle Light_2)/Cal_1) \times 100 + const \quad (3)$$

$$MotorR = (\triangle Max - \triangle Light_1)/Cal_2) \times 100 + const \quad (4)$$

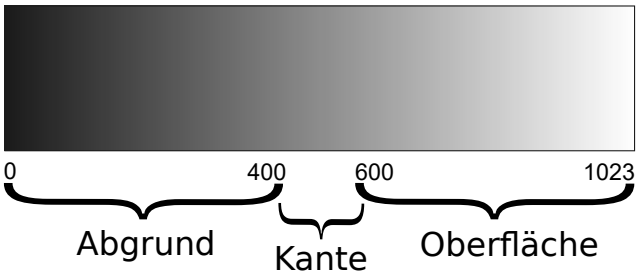


Abbildung 2. Darstellung der Sensorwerte zu den Messbereichen und der Grauwertskala

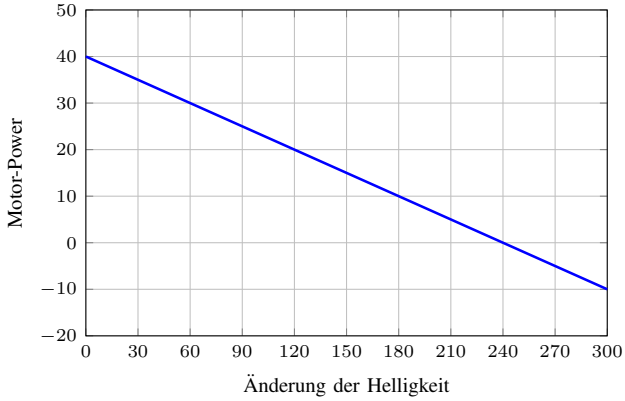


Abbildung 3. Helligkeitsänderung zu Motor-Power für jeweils Sensor-Motor-Paar: linker Sensor für rechten Motor und rechter Sensor für linken Motor. Vergleiche (3) und (4), wobei $\triangle Max = 60$, $Cal = 600$.

mit

$\triangle Max$ ... *Schwellwert*
$\triangle Light_1$ ... *Helligkeitsaenderung linker Sensor*
$\triangle Light_2$ ... *Helligkeitsaenderung rechter Sensor*
Cal_1 ... *Kalibrierungswert linker Sensor*
Cal_2 ... *Kalibrierungswert rechter Sensor*
const ... *Geschwindigkeitsregelung*

Der Schwellwert $\triangle Max$ ist ein Maß für die maximale Abweichung der Helligkeit vom Kalibrierungswert. Ist $\triangle Light$ größer als $\triangle Max$, so ist das Ergebnis negativ und der entsprechende Motor dreht Rückwärts. Ansonsten dreht der Motor vorwärts. Die Größe *const* ist für eine nachträgliche Geschwindigkeitsregelung vorhanden.

IV. Ergebnisdiskussion

Die Erkennung der Kanten einer farblich homogenen Oberfläche und die notwendige Steuerung der Motoren, um die Oberfläche nicht zu verlassen und weiterzufahren, wurde durch die implementierte Steuerung - bis auf einen Spezialfall - erreicht. Die genutzten Gleichungen (3) und (4), die die Motoren-Geschwindigkeit regeln, enthalten die nur empirisch bestimmten Größen $\triangle Max$ und *const*, die redundant sind. Bei der Erprobung der Fahrdynamik ließ sich mit Hilfe des konstanten Zusatzterms *const* die Fahrgeschwindigkeit

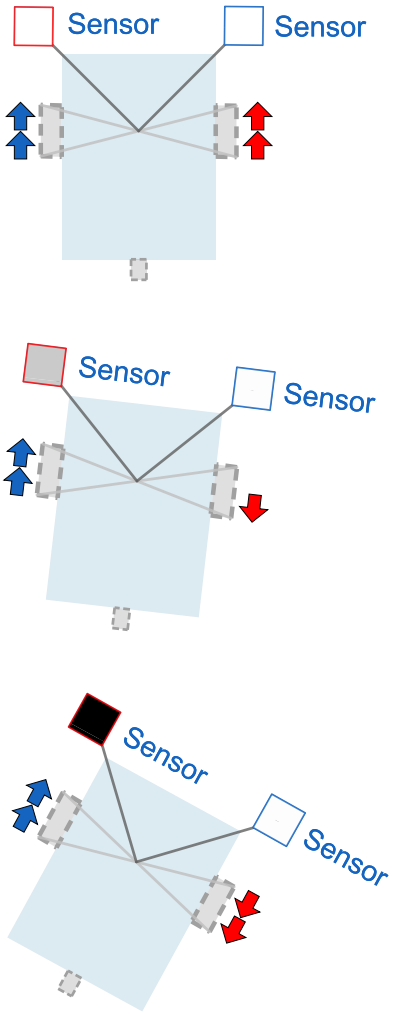


Abbildung 4. Beispiel für Motorenverhalten bezogen auf Sensorwerte. Weißer Sensor entspricht kleiner oder keiner Helligkeitsänderung, grauer Sensor ist Anfang einer Kante und schwarzer Sensor über einer Kante hinaus.

einfacher und direkter justieren, als mit $\triangle Max$. Im besonderen Fall war bei gleichzeitigem Auftreffen beider Sensoren auf eine Kante der Roboter in eine Schleife geraten. Beide Motoren bekamen dabei identische Anweisung und stoppten, fuhren rückwärts bis sie wieder auf der Unterlage waren und fuhren dann beide wieder vorwärts. Dieser Vorgang endete erst, wenn durch eine Störung beide Motoren stark unterschiedliche Werte bekamen. Im Normalfall sorgte die Steuerung dafür, dass der Roboter orthogonal zur Kante verblieb.

V. Zusammenfassung und Fazit

Die vorgestellte Steuerungs-Lösung bietet im Rahmen des Anforderungsprofils, auf einer farblich homogenen Oberfläche autonom zu fahren und diese nicht zu verlassen, eine mögliche Umsetzung. Sie ist übersichtlich in der Programmierung und enthält bei der Geschwindigkeitsregelung keine Bedingungen und/oder Verzweigungen, wodurch die Anfälligkeit für logische Fehler und die Laufzeit sinkt. Es sind jedoch die Implementierungsmöglichkeiten von Spezialfällen (Rückwärtsfahren, Fahren auf sich farblich ändernden Unterlagen) zu eruieren.

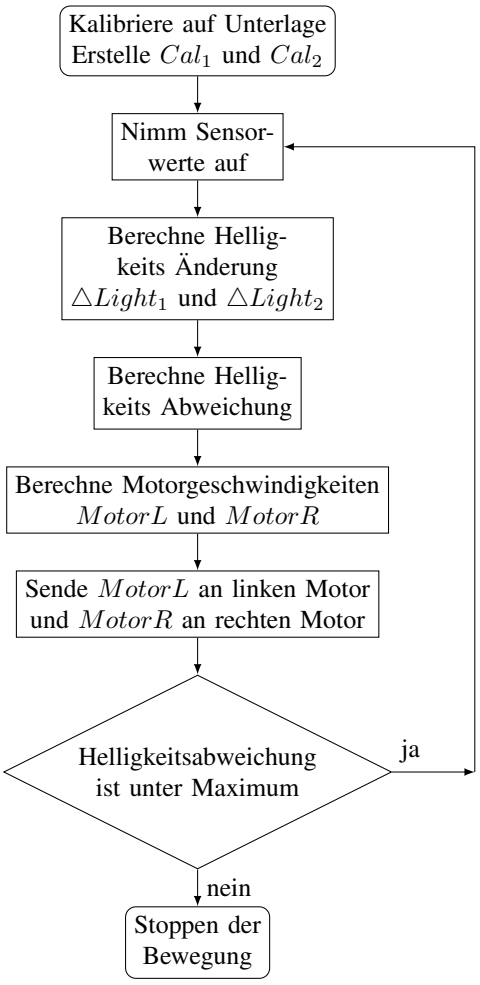


Abbildung 5. Programmablaufplan des Vorwärtsfahr-Algorithmus

Im Allgemeinen kann auf Oberflächen ein sich änderndes Reflexionsverhalten durch Lücken im Material oder durch aufgebrachte Farbe entstehen. Intensives, einfallendes Licht kann die Sensorik beeinflussen. Diese Fälle gilt es zu berücksichtigen.

Literaturverzeichnis

[1] ADAC: *ADAC Info - Fahrerassistenzsysteme*. https://www.adac.de/infotestrat/technik-und-zubehoer/fahrerassistenzsysteme/uebersicht/fahrerassistenzsysteme_uebersicht.aspx. Version: 2019. – [Online; Stand 15. März 2019]

[2] Wikipedia: *Automatic Train Operation — Wikipedia, Die freie Enzyklopädie*. https://de.wikipedia.org/w/index.php?title=Automatic_Train_Operation&oldid=183846734. Version: 2018. – [Online; Stand 15. März 2019]

[3] Finn Mayer-Kuckuk, Frankfurter R.: *Ohne Pilot über den Wolken*. https://www.fr.de/wirtschaft/ohne-pilot-ueber-wolken-10968097.html. Version: 2019. – [Online; Stand 15. März 2019]

[4] Lego: *LEGO® MINDSTORMS® Education*. https://education.lego.com/de-de/product/mindstorms-ev3. Version: 2019. – [Online; Stand 15. März 2019]

[5] Mathworks: *MATLAB*. https://de.mathworks.com/products/matlab.html?s_tid=hp_products_matlab. Version: 2019. – [Online; Stand 15. März 2019]

[6] Mathworks: *RWTH - Mindstorms NXT Toolbox*. https://de.mathworks.com/matlabcentral/fileexchange/18646-rwth-mindstorms-nxt-toolbox. Version: 2019. – [Online; Stand 15. März 2019]

[7] WIKIPEDIA: *Lego Mindstorms NXT — Wikipedia, Die freie Enzyklopädie*. https://de.wikipedia.org/w/index.php?title=Lego_Mindstorms_NXT&oldid=185430460. Version: 2019. – [Online; Stand 15. März 2019]

[8] WIKIPEDIA: *Lego Mindstorms NXT Programmierung— Wikipedia, Die freie Enzyklopädie*. https://de.wikipedia.org/w/index.php?title=Lego_Mindstorms_NXT&oldid=185430460#Programmierung. Version: 2019. – [Online; Stand 15. März 2019]

[9] MATHWORKS: *Webcam Support from MATLAB*. https://de.mathworks.com/hardware-support/matlab-webcam.html. Version: 2019. – [Online; Stand 15. März 2019]

[10] WIKIPEDIA: *Wendekreis (Fahrzeug) — Wikipedia, Die freie Enzyklopädie*. https://de.wikipedia.org/w/index.php?title=Wendekreis_(Fahrzeug)&oldid=185193522. Version: 2019. – [Online; Stand 17. März 2019]

[11] BRUENIG, Technischen Hochschule N.: *Bearbeitung und Gewinnung von Tiefeninformation durch die Kopplung zweier Kameras*. http://zuse1.efi.fh-nuernberg.de:8050/interaktion/index.php5?title=Bearbeitung_und_Gewinnung_von_Tiefeninformation_durch_die_Kopplung_zweier_Kameras&oldid=124. Version: 2019. – [Online; Stand 20. März 2019]

[12] WANG, Z.: *A MODEL OF LINE FOLLOWING ROBOT USING PID CONTROLLER: An Educational Platform Based on LEGO Mindstorms NXT Kit*. 2015

[13] WESTFALL, By S. ; MALOVICH, Dennis: *LEGO NXT: Features & Limitations*. Faculty of Science, University of Windsor, 2011. http://nxt.cs.uwindsor.ca/499football/features_limitations.pdf

[14] AZRAEL: *Unterlagen zum Praktikum Lego Mindstorms in der FEIT*. Fakultät für Elektrotechnik, Otto-vov-Guericke-Universität Magdeburg, 2012. https://elearning.ovgu.de/mod/resource/view.php?id=44972

Cocktailmixer aus LEGO

Adrian Hegmann, ETIT
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Der im Rahmen des LEGO Seminars entstandene Cocktailmixer stellt auf Grundlage seines flexiblen Aufbaus und anpassbaren Programmes eine kostengünstige, kompakte und mobile Alternative zu bisherigen Cocktailautomaten dar. Er wurde dazu programmiert, um aus zwei verschiedenen Ausgangsflüssigkeiten einen Cocktail zu mixen. In praktischen Tests wurden Fanta und Cola als Ausgangsflüssigkeiten gewählt. Die Auswahl, welches Getränk ausgegeben wurde, erfolgte per Tasterdruck durch den Benutzer. Insgesamt konnte die Maschine drei Getränke ausgeben: die erste Ausgangsflüssigkeit (Fanta),(Taster 1), die zweite Ausgangsflüssigkeit (Cola),(Taster 2) und die Mischung der Beiden (Spezi),(Taster 3).**

***Schlagwörter*—Cocktail, Cocktailautomat, Getränk, LEGO**

I. Einleitung

In Clubs und in Bars erfreuten sich Cocktails einer sehr großen Beliebtheit. Die fruchtigen Getränke gab es in unterschiedlichsten Variationen. Die Cocktails bestanden oft aus vielen verschiedenen Zutaten, die in geordneter Reihenfolge und im richtigen Mischverhältnis von dem Barkeeper zubereitet werden mussten. Heutige Cocktailautomaten wurden entwickelt, um die Cocktailzubereitung zu automatisieren, dabei kam es auf die Geschwindigkeit und Qualität an, mit der sie diese mixten. Während bereits einige schnelle und hochwertige Automaten produziert wurden, siehe die Beispiele [1] und [2], waren häufig deren Größe und Preis ein Grund für ihr Fehlen in privaten Haushalten. Der Cocktailmixer aus LEGO sollte eine Alternative zu den bisherigen Automaten darstellen, indem er handlich und flexibel, aber dennoch funktional war. Das Ziel war es, den Einsatz von Cocktailautomaten in der Gastronomie und in privaten Haushalten erstrebenswerter zu machen.

II. Stand der Technik/Grundlagen

Als Steuerungseinheit der Maschine wurde der mit Akku betriebene LEGO-NXT-Stein verwendet. Bei diesem handelte es sich um einen 32-Bit-Mikroprozessor mit vier Sensoreneingängen und drei Motorenausgängen. Weitere technische Daten des LEGO-NXT-Roboters wurden in [3] ausgeführt. Für die Programmierung wurde die MATLAB-Software des Entwicklers MathWorks zusammen mit der RWTH–Mindstorms NXT-Toolbox für MATLAB von der RTWH Aachen genutzt. Insgesamt wurden zwei Motoren, drei Tastsensoren und ein Distanzsensor in dem Projekt verbaut, siehe Abbildung 1. Bei den verwendeten Motoren handelt es sich um Servomotoren mit integriertem Rotationssensoren. Der Ultraschallsensor maß Abstände in einem Bereich von 6 cm bis 255 cm und gab diese in ganzen Zentimeterschritten zurück. Mit Ausnahme der zwei Drehventile, den Schläuchen und den Flüssigkeitstanks wurde die gesamte Maschine aus LEGO-Bauteilen konstruiert und gebaut.

DOI: 10.24352/UB.OVGU-2019-050

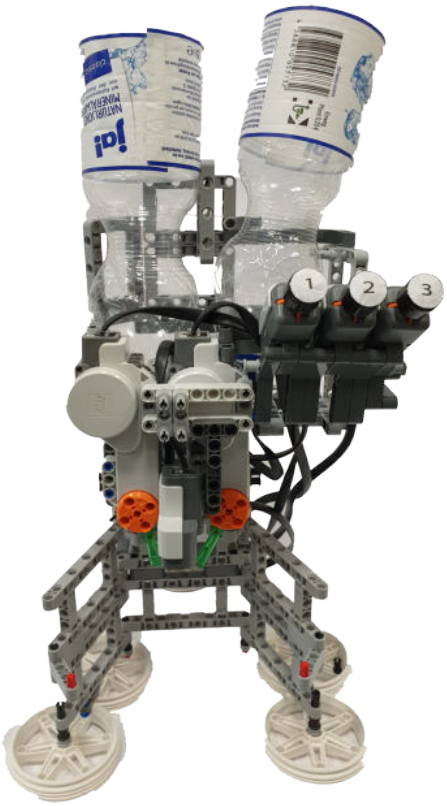


Abbildung 1. Die Vorderseite des Cocktailmixers, von oben nach unten, mit Flüssigkeitstanks, nummerierten Tastern, Servomotoren, Distanzsensor und Ausschank.

A. Bestimmung des Mischverhältnisses

Um ein Getränk zu mischen, wurden die Ventile einzeln und nacheinander geöffnet. Das erste Ventil wurde aufgedreht, und blieb eine Zeit t lang geöffnet, bevor es wieder geschlossen wurde. Während dieser Zeit t, wurde die gesamte benötigte Menge, der ersten Flüssigkeit, in das Gefäß gefüllt. Durch die Zeit t wurde sichergestellt, dass sich die erste Flüssigkeit im richtigen Verhältnis in dem Gefäß befand. Die folgende Formel diente zur Bestimmung der Zeit.

$$t = \frac{V \cdot n}{100} \cdot \frac{1}{v} \tag{1}$$

Dabei war V das Volumen des Gefäßes, das von dem Automaten gefüllt wurde. In den praktischen Tests wurden identische Becher verwendet und von der Maschine auf ein Volumen von 180 ml gefüllt. Die Variable n stand für den Anteil, der ersten Flüssigkeit, an der Gesamtmischung. Mit v wurde die durchschnittliche Durchflussgeschwindigkeit bezeichnet, mit der die Flüssigkeit das Gefäß füllte. Für

eine vollständige Füllung des Bechervolumens von rund 180 ml benötigte der Cocktailmixer eine mittlere Zeit von rund 36 s. Die Messungen der Füllzeit und des gefüllten Volumens wurden im Anhang dargestellt. Demnach, ergab sich eine durchschnittliche Durchflussgeschwindigkeit von $v = \frac{180}{36}\frac{\text{ml}}{\text{s}} = 5\frac{\text{ml}}{\text{s}}$. Nach dem die erste Flüssigkeit ausgegeben worden ist, wurde das zweite Ventil aufgedreht und die zweite Flüssigkeit ausgegeben. Das zweite Ventil blieb solange geöffnet, bis der Distanzsensor die Füllung des Gefäßes feststellte.

III. Hauptteil

A. Aufbau

Als Flüssigkeitsbehälter wurden zwei 0,5-Liter-Plastikflaschen genutzt, welche an ihrer Unterseite für eine einfache Nachfüllung aufgeschnitten wurden. Die Flaschen wurden kopfüber befestigt und bildeten den höchsten Punkt des Automaten, siehe Abbildung 1. Die Flüssigkeiten wurden durch die Schwerkraft angetrieben, bei geöffneten Ventilen konnten diese in das Gefäß des Benutzers fließen. Auf den Einsatz von Pumpen konnte damit verzichtet werden. Die Deckel der Flaschen wurden per Loch und Silikon mit dem Schlauchsystem verbunden. Die Körper der Flüssigkeitsbehälter waren austauschbar, so dass sie bei Bedarf durch Flaschen mit größeren Volumen ausgetauscht werden konnten. Die Flüssigkeitsbehälter wurden durch zwei Ventile mit dem Ausschank verbunden. Sie wurden so eingebaut, dass die Flüssigkeiten unabhängig voneinander ausgegeben werden konnten. Die Öffnung und Schließung der Ventile wurde durch zwei NXT-LEGO-Motoren gesteuert, wobei jeder Motor genau für ein Ventil zuständig war. Durch eine 90 Grad Drehung in die eine oder in die entgegengesetzte Richtung konnten die Ventile von den Motoren geöffnet und geschlossen werden, siehe Abbildung 2. Eine geregelte Ausgabe der verschieden Mischverhältnisse wurde durch die Motorensteuerung realisiert. Infolgedessen konnten die Ventile einzeln, nacheinander oder gleichzeitig geöffnet werden. Für die direkte Interaktion zwischen Mensch und Maschine wurden drei Tastsensoren verwendet, die für den Benutzer als nummerierte Taster erkenntlich waren. Jedes Mischverhältnis wurde durch einen anderen Taster dargestellt. Indem der Benutzer einen der Taster drückte, wurde ein Getränk bestellt. Die Befüllung des Gefäßes erfolgte automatisiert durch die Maschine. Um Unfälle zu verhindern, wurde die Software so programmiert, dass sich die Taster nicht drücken ließen, wenn kein Gefäß unter dem Ausschank stand. Der Distanzsensor wurde an der Vorderseite der Maschine platziert und befand sich damit über dem Getränkeausschank. Siehe Abbildung 1. Der Sensor maß den Abstand zur Unterlage. Wurde ein Gefäß unter den Sensor gestellt, ermittelte dieser die Veränderung des Abstandes und die Taster wurden aktiviert. Während das Gefäß befüllt wurde, maß der Sensor die Füllhöhe des Gefäßes, wenn die Füllhöhe den Grenzwert überschritt, also ein gewissen Abstand zwischen Sensor und Oberfläche unterschritt, wurde die Flüssigkeitsausgabe beendet. Ein bereits vollgefülltes Gefäß zu überfüllen wurde durch die automatisch Füllhöhenkontrolle verhindert. Der beispielhafte Programmablauf ist in Abbildung 5 dargestellt.

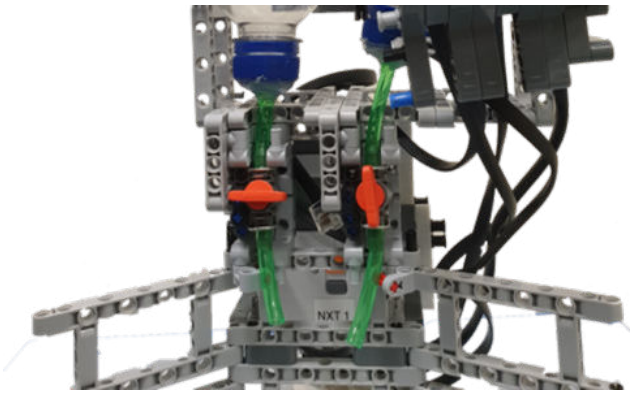


Abbildung 2. Die Ventile, links geschlossen und rechts geöffnet, in Verbindung mit dem Schlauchsystem und den Flüssigkeitstanks.

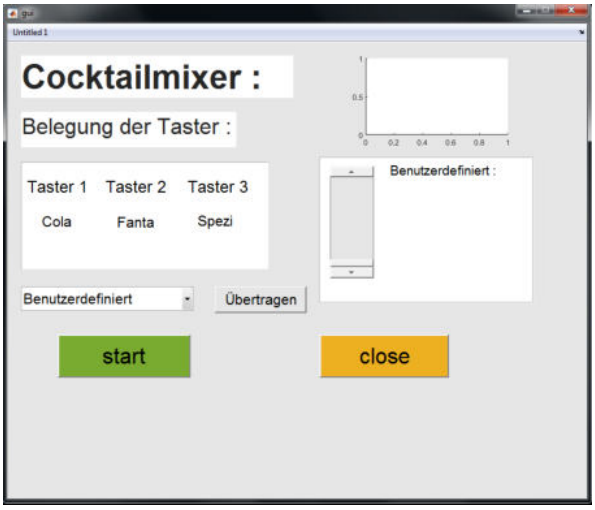


Abbildung 3. Die grafische Benutzeroberfläche, kurz GUI, des Programms.

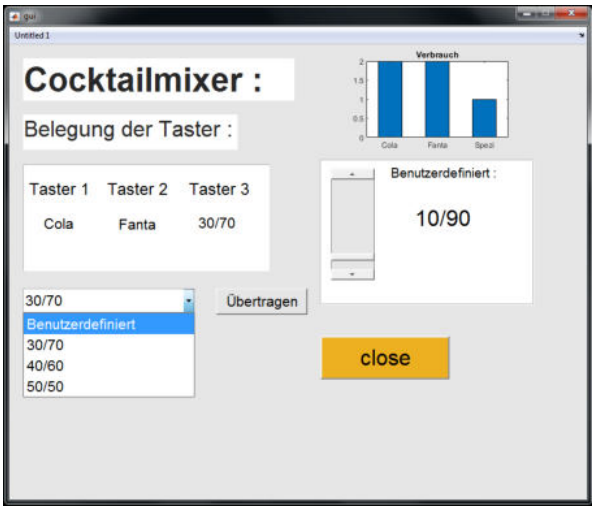


Abbildung 4. Die GUI nach einem Programmdurchlauf, mit geöffnetem Pop-up-Menü, eingestellten Schieberegler und Angabe der Benutzerauswahl im Balkendiagramm.

B. Programm

Der Cocktailmixer wurde gestartet und maß mit dem Distanzsensor den Abstand zwischen Sensor und Unterlage. Der gemessene Abstand wurde mit dem maximalen Abstand zwischen Sensor und Unterlage verglichen. Als maximaler Abstand wurde die ermittelte Entfernung zwischen Sensor und Unterlage bezeichnet, wenn kein Gefäß unter dem Ausschank stand. Dieser Wert galt als Referenz für den Vergleich und wurde als Grenzwert im Programm definiert. Wurde ein Gefäß unter den Distanzsensor gestellt, wurde der Abstand geringer und der Sensor erkannte, dass der gemessene Abstand den Grenzwert unterschritt. Das Gefäß wurde dann von dem Programm erfasst und die Taster wurden zur Eingabe aktiviert, siehe den Programmablauf in Abbildung 5. Nach dem Erkennen des Gefäßes, wurden die Daten der Tastersensoren abgefragt, um zu ermitteln, welcher der Taster gedrückt wurde. Je nachdem, welcher Taster von dem Benutzer ausgewählt worden ist, wurde eines von drei Unterprogrammen zur Befüllung und Mischung des Gefäßes gestartet. Im Unterprogramm wurde der aktuelle Füllstand des Gefäßes durch den Distanzsensor überprüft. In Abhängigkeit von dem ausgewählten Mischverhältnis, wurde die Zeit ermittelt, die das erste Ventil geöffnet sein musste, um die erste Flüssigkeit in dem ausgewählten Verhältnis auszuschenken, siehe Formel 1 zur Bestimmung der Zeit. Bei der Ermittlung der Zeit wurden die Fließgeschwindigkeit der Flüssigkeiten und das Volumen des Gefäßes berücksichtigt. Nach dem Ausschenken der ersten Flüssigkeit wurde das zweite Ventil solange geöffnet, bis der Grenzwert, für die maximale Füllhöhe, erreicht worden ist. Das Erreichen des Grenzwertes wurde als abgeschlossene Füllung des Gefäßes gedeutet. Der Füllvorgang wurde beendet und der Benutzer erhielt ein akustisches Signal als Hinweis, sein Getränk zu entnehmen. Das Programm zählte die Anzahl an Bestellungen und merkte sich nach jeder abgeschlossenen Füllung, wie häufig ein Taster gedrückt wurde. Diese statistischen Daten wurden an die GUI des Programms übergeben. Wurden die Taster betätigt ohne, dass sich ein Gefäß unter dem Ausschank befand, gab das Programm einen Signalton aus, um dem Benutzer auf seinen Eingabefehler aufmerksam zu machen.

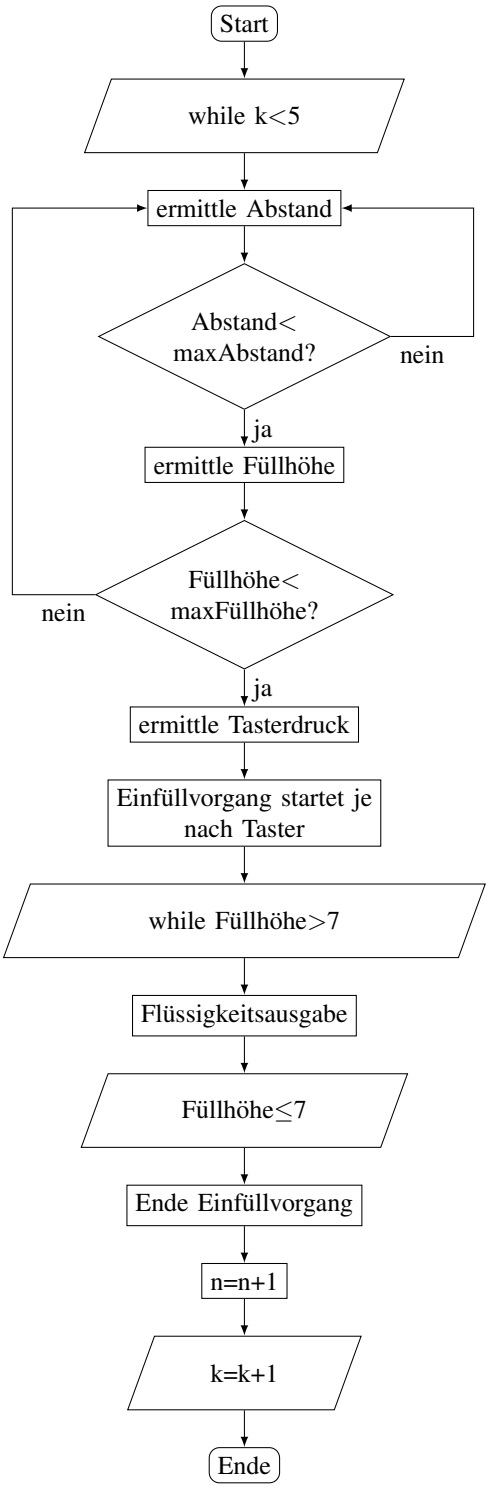


Abbildung 5. Programmablaufplan

C. GUI

Um die Interaktion von Benutzer und Maschine zu erweitern, wurde eine grafische Benutzeroberfläche dem hinzugefügt. Die GUI ist in Abbildung 3 dargestellt, wie sie zum Start des Programms erscheint. Dem Nutzer wurde ein genauer Überblick über die ausgewählten Mischverhältnisse ermöglicht. In einem Menü wurden die Taster zusammen mit den ausgewählten Mischverhältnissen angezeigt. Da der LEGO-NXT-Baustein auf vier Sensoreneingänge limitiert war und die Maschine beliebig viele Mischungen ausgeben solle, wurden ein Pop-Up-Menu und ein Schieberegler dem GUI hinzugefügt. Das Pop-Up-Menü bot dem Nutzer vordefinierte Mischverhältnisse an, welche in nachfolgenden Modellen für verschiedene Cocktail-Sorten stehen konnten. Mit einem Klick auf das ausgewählte Mischverhältnis und einem weiteren Klick auf den Bestätigen-Button, konnte die Auswahl des Nutzers an den dritten Taster übergeben werden. Um die Auswahlfreiheit des Nutzers zu

erhöhen, konnte über ein Schieberegler ein beliebiges Mischverhältnis, im Bereich von 0/100 bis 100/0 eingestellt und an den dritten Taster übergeben werden. In einem Balkendiagramm wurde dem Benutzer angezeigt, wie viele Cocktails insgesamt bestellt wurden und wie häufig die einzelnen Taster gedrückt worden sind. In der Abbildung 4 ist die GUI dargestellt, wie sie nach dem Programmdurchlauf erscheint.

IV. Schluss

A. Ergebnisse

Das Endergebnis war ein Cocktailautomat, mit der Funktion aus zwei Zutaten ein beliebiges Getränk in ein einem benutzerdefinierten Verhältnis zu mischen. Die Festlegung des Mischverhältnisses erfolgte über eine GUI, die es dem Benutzer erlaubte aus vordefinierten Verhältnissen zu wählen oder seine Eingabe über einen Schieberegler zu tätigen. Insgesamt standen dem Benutzer drei Taster, mit jeweils verschiedenen Mischverhältnissen (Getränken) zur Auswahl. Per Tasterdruck wurde die Eingabe bestätigt, die Ausgabe des Getränkes im richtigen Verhältnis erfolgte automatisiert durch die Maschine. Nach einer festgelegten Anzahl von Tasterdrücken gab das Programm nun in einem Balkendiagramm die Anzahl an Bestellungen pro Taster in dem GUI aus.

B. Fazit

Der im Seminar entstandene Cocktailmixer aus LEGO war funktionsfähig und wurde in der Praxis bereits mehrere Male erprobt. Diese Tests haben gezeigt, dass die Maschine die Grundfunktionen eines Cocktailautomaten erfüllte, aber das Systems noch nicht ausgereift genug war, um kommerziell eingesetzt werden zu können. Weder wurde die Geschwindigkeit, noch die Qualität erreicht, mit welcher heutige Automaten Getränke mixen. Des Weiteren war die Auswahlmöglichkeit mit nur drei Getränken, im Vergleich zu anderen Cocktailmaschinen stark eingeschränkt. Die Verwendung von nur zwei Ausgangsflüssigkeiten als Grundlage für die Getränkemischungen, war ebenfalls nicht ausreichend, um die gesamte Bandbreite an verschieden Cocktailsorten abzudecken. Ein wesentliches Unterscheidungsmerkmal zu bisherigen Automaten stellte das geringe Gewicht und die hohe Mobilität des Cocktailmixers aus LEGO dar. Der batteriebetriebene Cocktailautomat war leicht zu transportieren und auf Grund des LEGO-Aufbaus flexibel anpassbar.

C. Aussichten

Durch das Hinzufügen von weiteren Flüssigkeitstanks kann die Anzahl an auszugebenden Getränken erheblich gesteigert werden. Ein dritter Flüssigkeitstank könnte nach dem gleichen Prinzip, wie die zwei bisherigen Tanks, über einen dritten Motor und ein weiteres Ventil installiert werden. Wenn mehr als drei Ausgangsflüssigkeiten hinzugefügt werden sollen, ist es wegen der Limitierung des LEGO-NXT-Bricks auf 3 Motoreneingänge notwendig, mehrere Ventile über einen Motor zu steuern. Um finanzielle Ressourcen zu sparen, wäre es möglich den LEGO-NXT-Brick durch eine kostengünstigere Steuereinheit zu ersetzen. Leistungsstarke und kostengünstige Alternativen wären Mikrocontrollersysteme, wie die weitverbreiteten Arduino- oder Raspberry-Pi-Modelle. Diese würden das Ansteuern von Schlauchpumpen über Relais ermöglichen. Über ein Display kann die GUI an den Benutzer ausgegeben werden. Diesem wird es dann ermöglicht, seine Eingabe direkt über das Display zu tätigen. Durch die Verwendung eines Displays mit Touch-Funktion kann auf die Benutzung der Tastsensoren als Eingabegeräte verzichtet werden.

V. Anhang

Tabelle: Messung der Füllzeit und des gefüllten Volumens

Nr.	Füllzeit in Sekunden	Volumen in Milliliter
1	44,4	180
2	36,2	180
3	36,2	180
4	25,2	150
5	38,4	190
6	33,8	170
7	36,1	190
8	38,2	180
9	36,2	180
Mittel.	36,1	177,8

Literaturverzeichnis

[1] Vogel Communications Group GmbH und Co.KG: *Tüftler bauen Cocktailmaschine mit Simatic-Steuerung*. https://www.elektrotechnik.vogel.de/tueftler-bauen-cocktailmaschine-mit-simatic-steuerung-a-799225/. Version: März 2019

[2] Vogel Communications Group GmbH und CoKG: *Elektronischer Barkeeper mixt sieben Cocktails pro Minute*. https://www.elektronikpraxis.vogel.de/elektronischer-barkeeper-mixt-sieben-cocktails-pro-minute-a-518132/. Version: März 2019

[3] Wikipedia, the free encyclopedia: *Lego Mindstorm NXT*. https://de.wikipedia.org/wiki/Lego_Mindstorms_NXT. Version: März 2019

Bau eines Cocktailmixers

Jonas Tim Helmholz, ETIT/FEIT
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Während des LEGO-Projektseminars wurde ein Cocktailmixer gebaut, welcher aus zwei Zutaten Getränke mit unterschiedlichen Mischverhältnissen mixt. Dazu wurden ein NXT-Block, zwei Motoren, drei Taster, zwei Ventile und ein 40 cm Aquariumschlauch, sowie zwei 0,5 Liter PET-Flaschen verwendet. Zwei von drei Tastern ist jeweils ein fest definiertes Mischverhältnis zugewiesen. Das über den dritten Taster hinterlegte Mischverhältnis kann vom Nutzer, über das zugehörige GUI individuell definiert werden. Um Interessenforschung zu betreiben ist dem GUI ein Balkendiagramm hinzugefügt, worüber abgelesen werden kann, wie oft welches Getränk, in welchem Mischverhältnis durch den Nutzer ausgewählt ist. Zur Abstandsmessung zwischen Gefäß und Ausschank ist ein Ultraschallsensor eingesetzt, der bei fehlerhafter Gefäßposition einen Warnton auslöst sobald der Füllvorgang durch Drücken des Tasters gestartet werden soll. Bei korrekter Gefäßposition wird der Füllvorgang gestartet.**

***Schlagwörter*—Cocktailmixer, Getränkeautomat, LEGO Projektseminar, Mindstorms, NXT,**

I. Einleitung

COCKTAILS sind zu allen Veranstaltungen und Anlässen beliebt. Bei Großveranstaltungen, wie Konzerten oder Sportveranstaltungen, kann die zeitnahe Zubereitung der Cocktails problematisch werden, da das händische Zubereiten der Mixgetränke, z.B. durch den Barkeeper, zu viel Zeit in Anspruch nimmt. Deshalb werden Cocktails bei Großveranstaltungen eher selten angeboten. Ein Cocktailmixer ist leicht zu transportieren, flexibel, erweiterbar und kann das oben genannte Defizit in der Getränkeversorgung bei Großveranstaltungen ausfüllen.

II. Das Projekt

Nach 2 Wochen LEGO-Praktikum ist das Ergebnis ein Cocktailmixer, mit dem aus zwei Zutaten ein Mixgetränk hergestellt werden kann. Das Mischungsverhältnis ist variabel und lässt sich über das GUI durch den Benutzer anpassen. Über drei Taster werden die Einfüllvorgänge für das jeweilige Getränk gestartet.

III. Konzeption und Aufbau des Cocktailmixers

Das Ziel des Praktikums war es, einen Cocktailmixer zu bauen, mit dem aus zwei Zutaten ein Mixgetränk hergestellt werden kann. Bereits zu Beginn der Bauphase zeigte sich, dass die Herstellung des Cocktailmixers nicht ausschließlich mit LEGO-Teilen zu realisieren war. Deshalb wurden zusätzlich kleine Ventile einer Bewässerungsanlage und handelsübliche Aquariumschläuche genutzt. Damit der Cocktailmixer auch zuverlässige Ergebnisse liefert, wurde das Hauptaugenmerk auf den Ultraschallsensor bzw. den dazugehörigen Programmcode gelegt.

DOI: 10.24352/UB.OVGU-2019-051

A. Aufbau

Das Grundgerüst besteht aus handelsüblichen Lego-Technik-Teilen. Ebenso sind zwei Motoren, drei Taster, ein Ultraschallsensor und der NXT-Block verbaut. Das Ventilsystem ist in Abbildung 1 zu sehen.

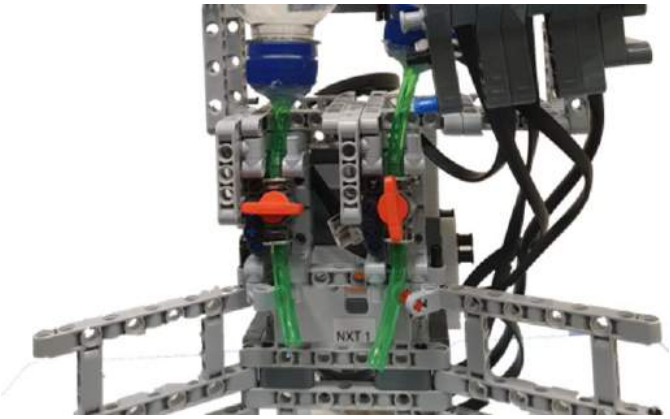


Abbildung 1. Ventilsystem

Dadurch, dass die Motoren des LEGO-NXT-Systems nicht sehr leistungsstark sind, können durch sie keine Pumpen angetrieben, sondern nur Ventile betätigt werden. Deshalb sind die Flüssigkeitsbehälter, welche aus 0,5-Liter-PET-Flaschen bestehen, an der höchsten Stelle des Cocktailmixers angebracht. Somit können die Flüssigkeiten allein durch Öffnen der Ventile ausgegeben werden.

Um das Befüllen der Flaschen zu vereinfachen, ist der nach oben gerichtete Boden der Getränkeflaschen entfernt. Der Schraubverschluss der Getränkeflaschen ist mithilfe von Silikonkleber mit den Schläuchen wasserdicht verbunden. Aufgrund genormter Größen der Flaschenverschlüsse können auch Flaschen mit größerem Volumen verwendet werden.

Beiden Reservoirflaschen ist jeweils ein Ventil zugeordnet, sodass das Abfüllen der Flüssigkeiten unabhängig von einander erfolgen kann.

Damit von dem Nutzer die Getränke ausgewählt werden können, sind drei Tastsensoren angebracht. Die Tastsensoren sind von eins bis drei nummeriert. Somit ist es dem Nutzer möglich, das gewünschte Mixgetränk gezielt auszuwählen.

Das Öffnen und Schließen der Ventile im 90-Grad-Winkel ist mittels einer LEGO-Achse durch den NXT-Motor realisiert. Durch die eingesetzte Motorsteuerung können die Ventile einzeln, nacheinander oder gleichzeitig geöffnet werden.

Mithilfe eines Ultraschallsensors wird überprüft, ob das Behältnis unter den Ventilen korrekt positioniert ist. Somit ist eine Fehlbedienung durch den Nutzer weitgehend ausgeschlossen. Ebenso wird dieser Sensor zur Füllhöhenkontrolle genutzt. Dies

geschieht alle 0,1 Sekunden. Damit kann gewährleistet werden, dass die Werte näherungsweise in Echtzeit dem Programm zur Verfügung stehen. Dies ist in sofern wichtig, als dass die Zutaten mit etwa 5 ml/s aus den Öffnungen strömen und die Ventile rechtzeitig geschlossen werden müssen, um ein Überlaufen bzw. Überfüllen des Bechers zu verhindern. Die Tabelle der Füllgeschwindigkeiten und des gefüllten Volumens, aus denen sich die Fließgeschwindigkeit errechnet, befindet sich im Anhang.

Wie aus der Abbildung 2 ersichtlich, ist die Standfläche des Cocktailmixers variabel. Damit kann gewährleistet werden, dass der Cocktailmixer eine kleine Transportgröße einnimmt, aber im Funktionszustand eine größtmögliche Standfestigkeit aufweist.

Abbildung 2. Frontansicht des Cocktailmixers

Abbildung 3. Seitenansicht des Cocktailmixers von rechts

B. GUI

Um die Interaktionen zwischen Benutzer und Maschine zu optimieren, ist eine grafische Benutzeroberfläche (GUI) hinzugefügt. Dadurch ist dem Benutzer ein genauer Überblick über die Mischverhältnisse gegeben. Die ausgewählten Mischverhältnisse werden zusammen mit den jeweiligen Tastern in einem Menü angezeigt.

Da der LEGO-NXT-Baustein auf vier Sensoren-Eingänge limitiert ist und die Maschine beliebig viele Mischungen ausgeben soll, kann der Nutzer auf das Drop-Down-Menü zurückgreifen. In diesem kann er aus einem der drei vorgegebenen Mischverhältnisse wählen oder per Slider sich selbst ein benutzerdefiniertes Mischverhältnis einstellen. Dies ist in einem Bereich von 0/100 bis 100/0 möglich. Die Ziffer vor dem Schrägstrich gibt die erste Flüssigkeit und die Ziffer dahinter die zweite Flüssigkeit im Verhältnis zur jeweils anderen Flüssigkeit an. Dies wird dann durch Drücken des Buttons “Übertragen” an den NXT übertragen. Durch Betätigen des Buttons “Start” wird das Programm auf dem NXT gestartet. Dies kann auch dann noch durchgeführt werden während das Programm schon läuft, um zum Beispiel das Mischverhältnis im laufenden Betrieb zu ändern ohne das GUI oder das Programm zu verlassen bzw. zu beenden.

Wie in Abbildung 4 dargestellt, ist das GUI trotz der Einstellungsmöglichkeiten einfach und übersichtlich gehalten, um eine möglichst einfache Bedienung zu gewährleisten.

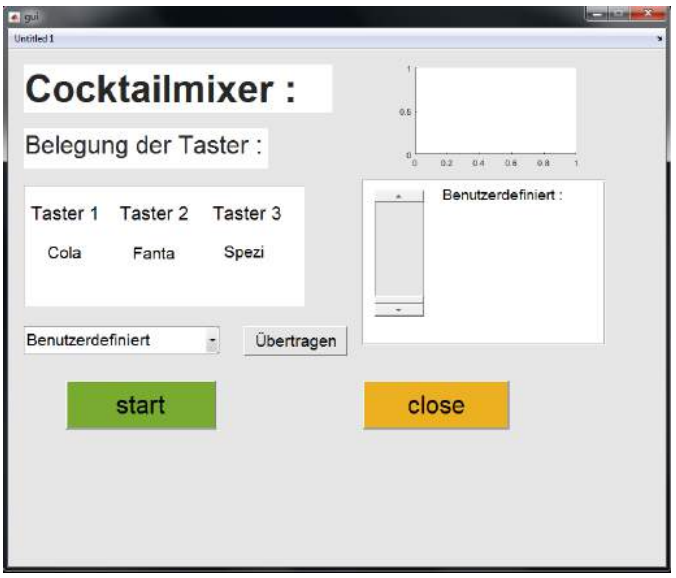


Abbildung 4. GUI

In der oberen rechten Ecke des GUI ist ein Diagramm eingebettet, um anzuzeigen, wie oft welcher Taster gedrückt und damit wie oft welches Mischverhältnis ausgewählt ist. Diese Anzeige erfolgt immer nach Ablauf der Hauptschleife. Die Anzahl der Wiederholungen wird durch den Nutzer im Programmcode eingestellt.

C. Software

Das Programm ist über MatLab realisiert, weshalb der Roboter nicht autark ohne PC funktioniert. Durch eine while-

Schleife läuft das Programm solange die Schleife aktiv ist. Nach Ablauf des gesamten Programms wird angezeigt, wie oft welches Mischverhältnis bestellt ist. Dadurch erhält der Nutzer einen Überblick über die nötige Menge der Zutaten für einen bestimmten Zeitraum. Durch eine der if-Funktionen wird überprüft, ob ein Becher oder ein anderes Gefäß unter der Ausgabe steht. Die andere if-Funktion überprüft, ob der Becher bereits gefüllt ist. Somit kann die maximale Einfüllhöhe nicht überschritten werden. Eine weitere while-Schleife überprüft während des Einfüllens die Füllhöhe und stoppt das Einfüllen, wenn die maximale Füllhöhe erreicht ist.

Sobald das Programm gestartet wird, misst der Distanzsensor den Abstand zwischen Sensor und Unterlage, welcher mit dem vordefinierten Abstand verglichen wird. Der vordefinierte Abstand ist der Abstand zwischen Sensor und Unterlage, wenn kein Becher unter dem Ausschank steht. Dieser Grenzwert ist im Programmcode definiert und dient als Referenz für den vorangegangenen Vergleich. Der Abstand wird geringer, sobald ein Gefäß unter dem Ausschank steht. Nun ist das Gefäß erkannt worden und die Taster werden freigegeben. Um zu erkennen welcher Taster gedrückt wird, werden die Daten der Taster ausgelesen. Jeder Taster startet ein eigenes Unterprogramm sobald dieser gedrückt ist und das Ausschenken beginnt.

Dieses Unterprogramm überprüft den aktuellen Füllstand des Gefäßes. Je nach ausgewähltem Mischverhältnis wird die Zeit berechnet, wie lange Ventil 1 geöffnet sein muss, um das richtige Verhältnis der ersten Flüssigkeit auszuschenken. Zur Berechnung der Zeit werden Fließgeschwindigkeit und das Volumen des Gefäßes berücksichtigt. Die zweite Flüssigkeit wird danach solange eingeschenkt, bis eine bestimmte Füllhöhe durch den Distanzsensor ermittelt wird. Dieser Grenzwert ist ebenfalls im Programmcode definiert. Durch das Balkendiagramm wird angezeigt, wie oft welcher Taster gedrückt ist. Sobald ein Taster betätigt wird, ohne das ein Gefäß unter dem Ausschank steht, ertönt ein Signalton, um den Nutzer auf den Fehler hinzuweisen.

Der Programmablaufplan zur Erklärung ist in Abbildung 5 dargestellt.

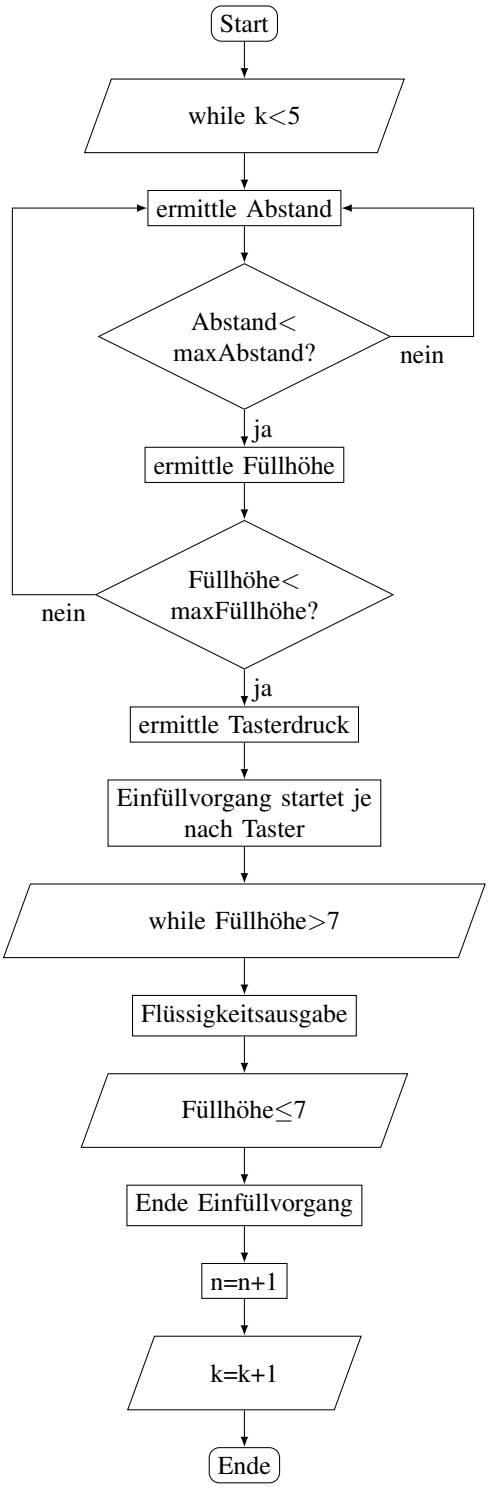


Abbildung 5. Programmablaufplan

IV. Ergebnisdiskussion

Das Endergebnis ist ein Cocktailmixer für zwei Zutaten, welche in unterschiedlichen Mischverhältnissen ausgeschenkt werden können. Die Dichtigkeit des Systems, insbesondere an der Verbindung zwischen Flaschenschraubverschluss und Schläuchen, ist eine im Verlauf der Betriebszeit des Cocktailmixers ein nicht zu vernachlässigendes Problem. Aus diesem Grund sind die elektronischen Bauelemente entweder oberhalb der Flüssigkeiten angebracht oder wirksam abgedeckt. Somit wird der Kontakt zwischen Flüssigkeiten und elektronischen Bauelementen verhindert. Ebenso wird durch den Einsatz von Pumpen und das Lagerungsverhältnis von elektronischen Bauelementen und der Flüssigkeiten die oben genannte Gefahr reduziert.

V. Zusammenfassung und Fazit

Durch den Einsatz eines weiteren NXTs könnten weitere Zutaten zur Herstellung von Mixgetränken genutzt werden,

wodurch sich dann eine größere Getränkevielfalt ergibt. Es wäre in diesem Fall allerdings erforderlich, die Fließgeschwindigkeit für die einzelnen Zutaten mithilfe von Durchflusssensoren zu ermitteln, um fehlerhafte Ausgaben einzuschränken. Ebenfalls wäre es von Vorteil, wenn anstatt der Ventile Peristaltikpumpen zum Einsatz kommen würden. Dadurch könnte die Geschwindigkeit des Ausschankes festgelegt werden und die Konstruktion des Coktailmixers würde eine stabiler Bauform erhalten.

ANHANG

Tabelle: Messung der Füllzeit und des gefüllten Volumens

Nr.	Füllzeit in Sekunden	Volumen in Milliliter
1	44,4	180
2	36,2	180
3	36,2	180
4	25,2	150
5	38,4	190
6	33,8	170
7	36,1	190
8	38,2	180
9	36,2	180
Mittel.	36,1	177,8

LITERATURVERZEICHNIS

[1] Vogel Communications Group GmbH und Co. KG: *Tüftler bauen Cocktailmaschine mit Simatic-Steuerung.* https://www.elektrotechnik.vogel.de/tueftler-bauen-cocktailmaschine-mit-simatic-steuerung-a-799225/. Version: März 2019.

[2] Vogel Communications Group GmbH und Co. KG: *Elektronischer Barkeeper mixt sieben Cocktails pro Minute.* https://www.elektronikpraxis.vogel.de/elektronischer-barkeeper-mixt-sieben-cocktails-pro-minute-a-518132/. Version: März 2019.

[3] Das Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme: *Roberta-Thema-Hardware des LEGO Mindstorms NXT Systems.* https://web.archive.org/web/20110627103625/http://www.roberta-home.de/de/was-bietet-roberta/roberta-reihe/roberta-thema-hardware-des-lego-mindstorms-nxt-systems. Version: März 2019.

Konstruktion eines 3D-Bildstanzers

Carolin Gold, Mechatronik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Zuge des Lego-Projekts an der Otto-von-Guericke-Universität Magdeburg im WS 18/19 wurde ein Bildstanzer mittels Lego® mindstorms® NXT 2.0 konstruiert und mit Hilfe von MATLAB programmiert. Der Bildstanzer erlaubt es, eine zweidimensionale, graue Rastergraphik in ein dreidimensionales Bild zu verwandeln, dessen Eindrucktiefe dem jeweiligen Grauton entspricht. Als Bearbeitungsmaterial wurde aufgrund seiner Eigenschaften Blumenschaum verwendet. Im Zuge der Entwicklung konnten die Pixeldichte von 10x15 px und die Geschwindigkeit des Vorgangs auf das Doppelte erhöht werden und erlauben nun eine Abbildung von 20x30 px in 12 min.**

***Schlagwörter*—LEGO, mindstorms, NXT, MATLAB, Stanzen, 3D-Druck**

I. Einleitung

3D-Drucker wurden in den letzten Jahren immer beliebter. Wer träumte nicht davon, zu Hause am Computer Dinge zu entwerfen und sich diese direkt ausdrucken zu können. Seien es praktische Dinge wie Blumenvasen oder Besteck, oder einfach nur Dekoration wie das eigene Modell vom Eiffelturm oder einer Comicfigur, 3D-Drucker bieten viele Anwendungsmöglichkeiten. Aber auch in der Industrie fand der 3D-Drucker Einzug zur Herstellung von Prototypen, von individuell passenden Bauteilen oder auch von Prothesen [1]. Neben dem 3D-Druck gibt es auch weitere Verfahren zur Herstellung dreidimensionaler Bauteile oder Gebilde. Hierzu gehört unter Anderem das Fräsen. Dienen beim 3D-Druck als Bearbeitungsmaterial meist Kunststoffe, werden beim Fräsen mitunter Metalle oder Holz verwendet. Im Gegensatz zum 3D-Druck, bei dem das dreidimensionale Objekt durch schichtenweises Aufbringen des Materials aufgebaut wird, wird dieses beim Fräsen aus einem Werkstoffblock durch Abtragen des Materials gewonnen. Unabhängig von der Herstellungsart werden immer und überall dreidimensionale Objekte definierter Form benötigt.

Ziel des Lego-Projekts war es, ein voll funktionsfähiges Gerät aus Teilen des Lego® mindstorms® NXT 2.0 zu konstruieren und mittels MATLAB (The MathWorks, Inc., Natick, Massachusetts) und der RWTH - Mindstorms NXT Toolbox (RWTH, Aachen, Deutschland) zu programmieren. Vorliegend wurde ein Bildstanzer entwickelt, welcher es erlaubt, eine eingelesene, zweidimensionale Rastergraphik in Graustufen Pixel für Pixel in ein dreidimensionales Bild zu verwandeln (siehe Abbildung 1).

II. Vorbetrachtungen

Die Idee hinter dem vorliegenden Projekt war es, ein Gerät zu konstruieren, welches es erlaubt ähnlich wie beim 3D-Druck oder beim Fräsen anhand einer zweidimensionale Vorlage ein dreidimensionales Objekt zu erzeugen.

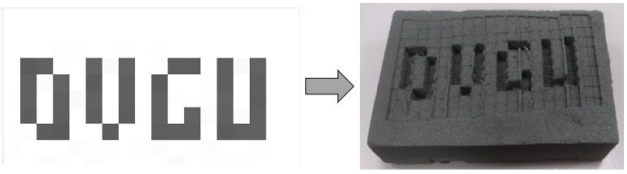


Abbildung 1. Umwandlung einer zweidimensionalen Graphik in ein dreidimensionales Bild

A. 3D-Drucker

Ein 3D-Drucker erstellt dreidimensionale Objekte, indem das Druckmaterial wie Kunststoff, Metall oder auch Sand geschmolzen und Schicht für Schicht zum Zielobjekt zusammengeführt wird [2]. Das Problem bei der Umsetzung mittels Lego® liegt darin, dass ein Schmelzen von Material nicht möglich ist. Einerseits gibt es hierfür keine passenden Bauteile, andererseits besteht Lego® selbst aus Kunststoff und würde somit beim Prozess schmelzen.

B. Fräse

Beim Fräsen wird durch Bewegung eines rotierenden, mit Zähnen besetzten Fräskopfs Material abgetragen. Als Material werden meist Holz, Kunststoffe oder Metalle verwendet [3]. Allerdings ist eine Bearbeitung von harten Materialien mit Lego® aufgrund des geringen Drehmoments der Motoren nicht möglich.

C. J.A.M.M.M.

Da Fräsen und 3D-Druck mit Lego aufgrund der oben genannten Tatsachen wegfällt, wurde als Vorbild zum Projekt der J.A.M.M.M., der im Wintersemester 2013 von Studenten der RWTH-Aachen entwickelt wurde, verwendet [4]. Diese Maschine drückt ein Höhenprofil von 22x20 Pixel innerhalb von 40 Minuten in Blumenschaum [5].

III. Konstruktion und Programmierung des Bildstanzers

Der konstruierte Bildstanzer ist weder ein 3D-Drucker, da er kein Material zuführt, noch eine Fräse, da er nicht durch Drehbewegungen Material entfernt. Er drückt mit einem zweidimensionalen Bild als Vorlage ein dreidimensionales Höhenprofil Pixel für Pixel in das Ausgangsmaterial (siehe Abbildung 1) und erzeugt somit ein ähnliches Endprodukt, wie die beiden vorher genannten Bearbeitungsvarianten. Als Bearbeitungsmaterial wurde aufgrund seiner Werkstoffeigenschaften

DOI: 10.24352/UB.OVGU-2019-052

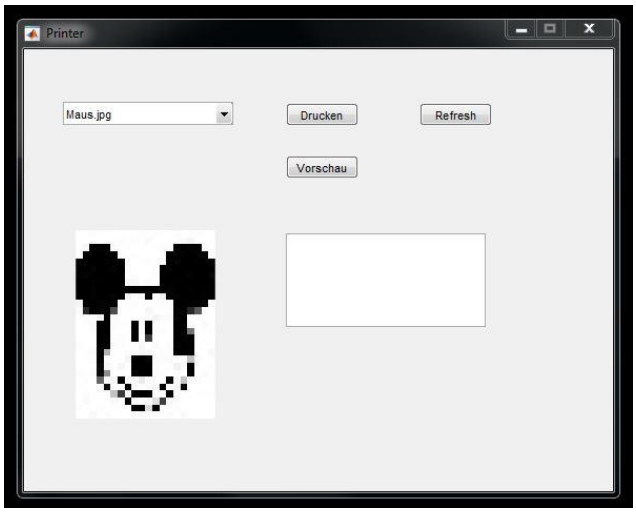


Abbildung 2. Graphische Benutzeroberfläche zur Steuerung des Bildstanzers

Blumenschaum, auch bekannt als Steckmasse, gewählt. Dieser lässt sich mit relativ geringem Kraftaufwand eindrücken und verformen und verbleibt dauerhaft im so geschaffenen Zustand.

A. Programm

Zur Steuerung des Geräts wurde eine graphische Benutzeroberfläche (siehe Abbildung 2) programmiert, welche es erlaubte, mittels Popup-Menü eine Graphik im .jpg-Format aus einem definierten Ordner auszuwählen. Die Aktualisierung des Menüs erfolgte über den Refresh-Button und erlaubte auch nach Start des Programms eine Erweiterung der Bildbibliothek. Über den Vorschau-Button konnte die ausgewählte Graphik dargestellt und gleichzeitig auf ihre Größe überprüft werden. War das Format größer als 20x30 px, wurde eine Fehlermeldung ausgegeben und der Druckvorgang konnte nicht gestartet werden. Der Programmablaufplan zur Steuerung über die graphische Benutzeroberfläche ist in Abbildung 3 dargestellt. Der Drucken-Button startete bei korrekter Bildgröße den eigentlichen Druckvorgang. Die eingeladene Graphik wurde als zweidimensionale Matrix ausgegeben. Jedes Feld der Matrix beschrieb ein Pixel der Graphik. Der Wert der Matrix war der jeweilige Grauton, wobei 0 schwarz und 255 weiß darstellte. Beim Druckvorgang wurde die eingelesene Matrix Pixel für Pixel von rechts oben nach links unten abgearbeitet. Da schwarz den tiefsten Eindruck ergeben sollte und weiß den geringsten, wurde die Drucktiefe über den Winkel des Motors mit Hilfe einer linearen Funktion in Abhängigkeit der Matrixwerte berechnet. Wodurch Eindrucktiefen von 0,1cm bis 1,5cm realisiert werden konnten. Der eigentliche Druckvorgang erfolgte nach Schema von Abbildung 4.

B. Antriebssysteme

Abgearbeitet wurde die Graphik zeilenweise von rechts nach links in x-Richtung. Nach Weiterfahrt um jeweils ein Pixel wurde über den Stanzkopf in z-Richtung gedruckt. Nach Beendigung einer Zeile wurde der Druckkopf wieder in die Anfangsposition in x-Richtung bis zum Tastsensor bewegt

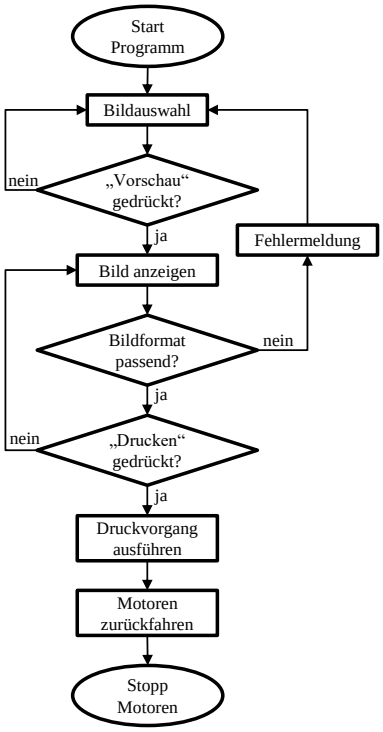


Abbildung 3. Programmablaufplan zur Steuerung der Bildauswahl und zum Starten des Druckvorgangs

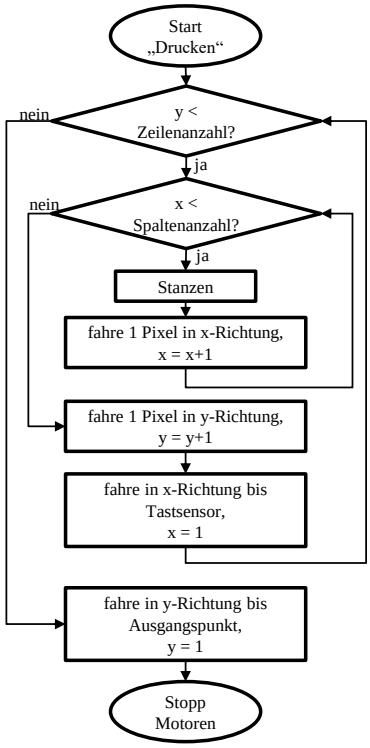


Abbildung 4. Programmablaufplan des Druckvorgangs

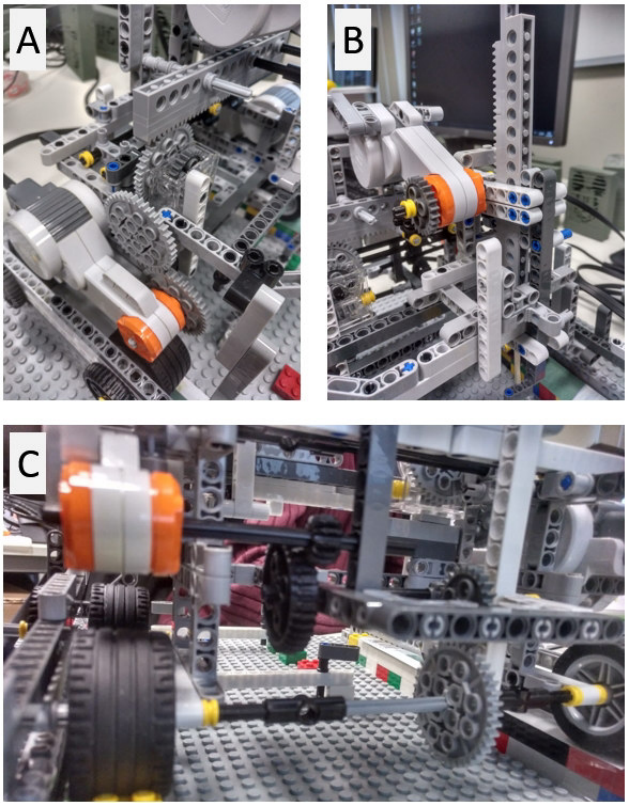


Abbildung 5. Konstruktion der Motoren und Übersetzungen für die Bewegung in x- (A), y- (C) und z-Richtung (B)

und durch Drücken von diesem zurückgesetzt. Das Gerät fuhr daraufhin ein Pixel in y-Richtung weiter und arbeitete nun die nächste Zeile ab. Der Vorgang wiederholte sich so lange, bis die komplette Graphik in ein dreidimensionales Bild übersetzt wurde. Realisiert wurde dieser Vorgang über drei Motoren, wobei jeweils einer für x-, y- und z-Richtung zuständig war. Die Bewegungen in x- und y-Richtung benötigten eine hohe Genauigkeit und Präzision, um die einzelnen Pixel genau nebeneinander setzen zu können. Dies wurde über Übersetzungen ins Langsame durch Zahnradsysteme bewerkstelligt. Das Abfahren der x-Richtung erfolgte über ein Zahnradsystem aus 4 Zahnrädern und einem Schneckengewinde. Es wurde hierbei eine Zahnradschiene in x-Richtung über ein Zahnrad geführt (siehe Abbildung 5A). Vor allem die Verwendung des Schneckengewindes sorgte für eine hohe Übersetzung. In y-Richtung wurde das komplette Gerät mittels Gummirädern, welche an einer Führungsschiene befestigt waren, über den Blumenschaum fortbewegt. Verwendet wurde hierbei ein System aus 5 Zahnrädern (siehe Abbildung 5C). Die Stanzbewegung erfolgte wiederum über eine Zahnradschiene, die an einem Zahnrad vorbei geführt wurde (siehe Abbildung 5B). Sie besaß eine schnellere Übersetzung, um den eigentlichen Stanzvorgang zu beschleunigen.

C. *Stanzkopfsystem*

Zum Einstanzen der Pixel in den Blumenschaum wurde zuerst ein Metallwürfel mit einer Kantenlänge von 5 mm

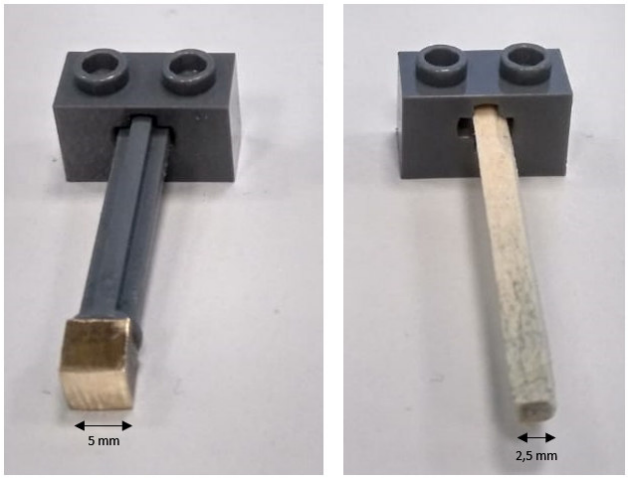


Abbildung 6. Stanzköpfe unterschiedlicher Kantenlänge

verwendet, welcher an ein Legoteil angeklebt wurde (siehe Abbildung 6 links). Dieser erlaubte mit der Limitierung der Fläche durch die Größe des Blumenschaums die Darstellung von Graphiken mit bis zu 10x15 px. Das Ersetzen des Stanzkopfs durch ein einfaches Streichholz, verdoppelte die mögliche Pixeldichte und erlaubte die Darstellung von Graphiken bis 20x30 px (siehe Abbildung 6 rechts).

IV. Ergebnisdiskussion

Das Ergebnis des Projekts war ein Bildstanzer, der Bilder mit einer Auflösung von 20x30 px in Blumenschaum druckt. Durch den 3D-Effekt der Stanztiefe besitzt das Bild einen 3D-Effekt mit guter Tiefenwirkung. Die erste Version des Bildstanzers erlaubte das Drucken von Graphiken mit einer Größe von maximal 10x15 px. Limitiert wurde dies durch die Größe des Stanzkopfs von 5 mm Kantenlänge und der Größe des Blumenschaumstücks von 8x11 cm und der Konstruktion des Geräts. Auch war das erste Programm relativ langsam und benötigte für die 10x15 px mehr als 15 min. Ein weiteres Problem lag in der Genauigkeit der Lego-Motoren, welche zum Spielen für Kinder ausgelegt und demnach nicht sehr präzise sind. Dies zeigte sich beispielsweise darin, dass sich der eigentliche Ausgangspunkt einer Motordrehung (0 Grad) um bis zu 30 Grad verschob und weiterhin als 0 Grad ausgegeben wurde. Dies führte dazu, dass jede Zeile in x-Richtung im Gegensatz zur vorherigen Verschiebungen aufwies. Dieses Problem lies sich durch das Anbringen eines Tastsensors zum Zurücksetzen des Motorwinkels auf 0 Grad am Beginn der jeweiligen Zeile beheben. In y-Richtung entstand das Problem, dass die Abstände, die das Gerät fährt ebenfalls von Zeile zu Zeile unterschiedlich waren. Entweder überschnitten sich die einzelnen Pixel oder es blieb zwischen den Zeilen eine Art Wand stehen. Es gab keine Systematik, nach der diese Unterschiede auftraten. Um dieses Problem zu verringern wurden die Gummireifen, statt direkt auf Legosteinen zu fahren, auf eine glatte Schicht aus Pappkarton überführt. Dies verhinderte ein Verhaken der Räder an den Steinen und verminderte die Abweichungen, konnte sie jedoch nicht vollständig beheben. In z-Richtung sollte der Motor sich nach

Abbildung 7. Logo der Otto-von-Guericke-Universität Magdeburg erstellt durch den Bildstanzer

dem Stanzen immer wieder zum Anfangswinkel von 0 Grad zurückfahren. Es zeigte sich, dass hier ein Überschwingen von 20 Grad stattfand. Nach Setzen der Endposition auf 20 Grad wies der Winkel keine Probleme mehr auf. Nachdem die Abweichungen verringert wurden, konnte der Stanzkopf verkleinert werden. Die Verkleinerung der Kantenlänge des Stanzkopfes führte zu der erwünschten Erhöhung der Pixeldichte auf das Doppelte der ursprünglichen Werte. Durch Behebung der oben beschriebenen Probleme und gleichzeitiges Erhöhen der Motorgeschwindigkeiten ergab sich zeitgleich eine Verdopplung der Geschwindigkeit auf 12 min für die maximalen Bildgröße. Vergleicht man diese Ergebnisse mit denen des Projekts J.A.M.M.M. an der RWTH-Aachen [4], war eine Erhöhung der Pixeldichte, sowie eine Beschleunigung des Vorgangs um das Dreifache zu verzeichnen. Beim J.A.M.M.M. dauerte eine Abbildung mit 20x22 px etwa 40 min [5].

V. Zusammenfassung und Fazit

Das Projekt erlaubte den Bau eines Bildstanzers mittels Lego® mindstorms® NXT 2.0 und MATLAB, welcher zweidimensionale Graphiken in dreidimensionale Bilder umwandeln konnte. Die Konstruktion erreichte eine maximale Pixeldichte von 20x30 px und eine Stanzgeschwindigkeit von 12 min für diese Bildgröße. Diese Kombination war beispielsweise ausreichend, um das Logo der OVGU Magdeburg darzustellen (siehe Abbildung 7). Im Verlauf des Projekts konnten das System und die verschiedenen Abläufe optimiert werden. Falsche Bildformate wurden herausgefiltert und das Gerät lief bei korrekter Bildeingabe nach dem Starten konstant bis zum Ende durch und erzielte gleichbleibend gute Ergebnisse. Unterschiedliche Anbringungen wie beispielsweise Tastsensoren am Ausgangspunkt sorgten dafür, dass sich der Einfluss der Abweichungen der einzelnen Motoren auf ein Minimum verringerten. Differenzen in der Darstellungsart entstanden schlussendlich nur noch durch die Verwendung unterschiedlicher Materialdicke des Blumenschaums. Im Vergleich zu ähnlichen Projekten konnten die Bildgröße und die Stanzgeschwindigkeit erhöht werden. Eine Vergrößerung des Systems und eine Verwendung größerer Blumenschaumblöcke würde des Weiteren eine Darstellung komplexerer Graphiken erlauben. Diese umfassen die Abbildung größerer Graphiken oder Verfeinerungen der Auflösung.

Anhang

Ein Video des Druckvorgangs des OVGU-Logos (siehe Abbildung 7) ist auf dem Youtube-Kanal von Mathias Magdowski unter dem Titel „Bildstanzer für Blumensteckmasse aus dem Lego-Praktikum 2019 an der OVGU Magdeburg“ zu finden [6].

Literaturverzeichnis

[1] Fischer, Andreas: *Der industrielle 3D-Druck wird erwachsen*. https://www.com-magazin.de/praxis/hardware/industrielle-3d-druck-erwachsen-1530491.html. Version: April 2018. – abgerufen am 14.03.2019

[2] TonerPartner: *3D Drucker - der neueste Hype bald auch für zu Hause?* https://www.tonerpartner.de/3d-drucker/. – abgerufen am 14.03.2019

[3] Koether, Wolfgang Reinhard; R. Reinhard; Rau: *Fertigungstechnik für Wirtschaftsingenieure*. 08. Hanser, 2012. – 460 S. – ISBN 978–3–446–44831–5

[4] Kpaczka, Marcin: *J.A.M.M.M.* https://mindstorms.lfb.rwth-aachen.de/index.php/56-robotswinter2013/roboter-des-jahres-2013/454-jammm128. Version: 2013. – abgerufen am 13.03.2019

[5] Shurrikane: *Lego Mindstorms - 3D Milling Machine J.A.M.M.M. (RWTH Aachen 2013)*. https://www.youtube.com/watch?v=gYN3RIQwWF0. Version: März 2014. – abgerufen am 13.03.2019

[6] Magdowski, Mathias: *Bildstanzer für Blumensteckmasse aus dem Lego-Praktikum 2019 an der OVGU Magdeburg*. https://www.youtube.com/watch?v=o3FvmyHkIcc. Version: Februar 2019. – abgerufen am 13.03.2019

Bau eines 3D-Stanzers

Bericht zum Projektseminar Elektrotechnik/Informationstechnik

Jonas Ratajczak, Mechatronik
Otto-von-Guericke-Universität Magdeburg

***Abstract*— Die folgende Arbeit dokumentiert die wesentlichen Bestandteile eines subtraktiven 3D-Druckers. Hierfür wird die prinzipielle Konstruktion der mechanischen Komponenten als LEGO Gerüst mit dem Mindstorms NXT Baukasten vorgestellt. Darauf aufbauend werden die physischen und informatischen Grundlagen des Druckprozesses erläutert. Dabei wird ein besonderes Augenmerk auf den MATLAB basierten Druckalgorithmus sowie die Vorteile der genutzten Werkstoffe gelegt. Abschließend werden die erreichbaren Bildqualitäten dieser Maschine besprochen und mögliche Verbesserungen am System aufgezeigt.**

***Schlagwörter*— 3D-Druck, Bildverarbeitung, LEGO, MATLAB, MINDSTORMS, Stanzen.**

I. Einleitung

In den letzten Jahren hat der Einsatz von 3D – Druckern immer mehr an Bedeutung gewonnen. Ob in der Medizin- und Zahntechnik, dem Automobilbau oder in der Verpackungsindustrie, überall finden sich Anwendungen [1]. Das Problem vieler dieser Anwendungen ist aber ihr hoher Preis. Das hier vorgestellte System zeigt durch Gebrauch eines günstigen Werkstoffes eine mögliche Alternative zu dieser Technik. An Stelle eines Druckprozesses tritt dabei ein Stanzprozess, der dreidimensionale Reliefs in den leicht verformbaren Werkstoff stanzt.

II. Vorbetrachtungen

Das System soll die Ergebnisse eines 3D – Druckers erzielen, dabei aber ähnlich wie eine Stanze arbeiten. Im Folgenden werden beide Verfahren vorgestellt und es wird auf ähnliche, bereits existierende Ansätze verwiesen.

A. 3D – Druck (additive Fertigung)

Für das dreidimensionale Drucken werden drei Verfahren verwendet. Bei der Stereolithographie (SLA) wird flüssiges Epoxidharz durch einen Laser schichtweise gehärtet. Beim Laser-Sintern (SLS) schmilzt ein Laser immer neue Schichten von Kunststoff-, Keramik- oder Metallpulver und schafft somit stabile und harte Strukturen. Bei dem weitverbreiteten Schmelzschichtungsverfahren (FDM) werden thermoplastische Kunststoffe erhitzt und schichtweise übereinandergeschichtet, härten aber nicht vollständig aus.[2]

B. Stanzen (subtraktive Fertigung)

Ziel des Stanzens ist es, Flachteile mit bestimmten Formen herzustellen. Dabei kommt eine Presse zum Einsatz, die einen Stempel in den Werkstoff drückt, der mittels Scherschneiden das Flachteil mit einer einzigen Bewegung fertigt. Als Werkstoff werden vor allem Metalle, aber auch Karton und Wellpappe bearbeitet. [3]

C. Bisherige Modelle

Das hier vorgestellte Projekt nahm sich ein im Jahr 2014 an der RWTH Aachen entwickeltes System [4] zum Vorbild, das zwar nach dem gleichen Arbeitsprinzip arbeitete, aber für seine Ergebnisse von 22x20 Pixel 40 Minuten brauchte [5]. Ähnliche Ergebnisse können auch von Fräsen erzielt werden. Eine hierfür beispielhafte LEGO CNC Fräse wurde von Arthur Sacek entwickelt [6]. Der Einsatz eines Bohrkopfes als Fräse ermöglichte dort genauere Ergebnisse. Bei den soeben genannten Modellen kam der auch in diesem Projekt verwendete Werkstoff zum Einsatz, welcher im nächsten Abschnitt beschrieben wird.

III. Umsetzung

Die Funktionsweise der Maschine lässt sich in eine mechanische und eine informatische Komponente aufteilen. In den nachfolgenden, aufeinander aufbauenden Unterabschnitten wird auf beide Komponenten eingegangen.

A. Grundkonzept

Das grundsätzliche Arbeitsprinzip des Stanzers ist das Einlesen von Bilddateien und die Verarbeitung dieser zu einem dreidimensionalen Relief. Die hierfür ausschließlich aus Grautönen bestehenden JPEG Dateien geben über die Helligkeit ihrer Pixel dem Stanzer Auskunft über die Tiefe des Reliefs an dieser Position.

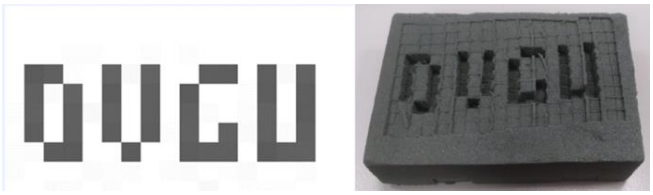


Abbildung 1: links: beispielhafte JPEG Datei, rechts: Gestanztes Relief.

Eine gute Veranschaulichung dieses Konzeptes liefert Abbildung 1. Für das zu stanzende Material fiel die Wahl auf Blumenschaum–Steckmasse, da diese einfach formbar und

DOI: 10.24352/UB.OVGU-2019-053

vergleichsweise günstig ist. Die Bearbeitung dieser geschieht über eine LEGO Mindstorms NXT – Vorrichtung, die durch ein MATLAB basiertes Programm gesteuert wird. Als Schnittstelle zwischen MATLAB und LEGO wurde die Mindstorms NXT Toolbox für MATLAB der RWTH Aachen verwendet [7].

B. Programm

Für das Programm des Stanzers wurde ein MATLAB Skript verwendet, welches eine wie in Abb. 2 dargestellte GUI aufruft.

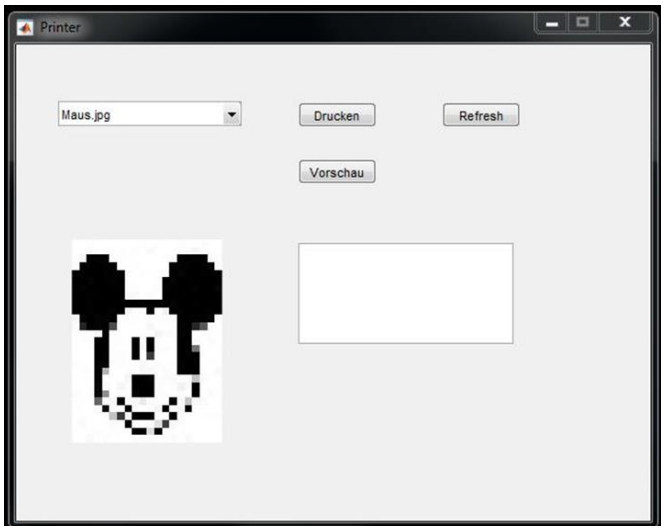


Abbildung 2: GUI des Programms

Die Auswahl der Bilddateien erfolgt über ein Popup Menü, welches bei jedem Betätigen des „Refresh“ Buttons sowie bei Erstaufruf der GUI alle JPEG Dateien des Ordners auflistet, in dem das Programm liegt. Die ausgewählte Bilddatei wird bei Betätigen des „Vorschau“ oder „Drucken“ Buttons in eine Matrix konvertiert, die im in Abb. 2 unten links zu sehenden Vorschaufenster ausgegeben wird. Bei zu kleinen Bildmaßen wird der zu stanzende Bereich verkleinert. Sind die gewählten Pixelmaße zu groß, wird die Datei zwar im Vorschaufenster gezeigt, aber im rechts danebenstehenden Feld wird eine Fehlermeldung ausgegeben und es ist nicht möglich, den Druckvorgang zu starten. Dieser besteht im spalten- und zeilenweisen Abarbeiten der gegebenen Grautonmatrix, wobei der Zeilenindex der Matrix die y – Koordinate und der Spaltenindex die x – Koordinate des Bildes darstellt. Der Wert der Matrix an der betrachteten Stelle gibt dem System Auskunft über die Tiefe des Reliefs. Die Eindringtiefe des Stanzkopfes ergibt sich dabei aus der Summe eines festen Abstands zum Werkstoff und dem invertierten und um den Faktor 1/6 skalierten Helligkeitswert des betreffenden Pixels. Das Abarbeiten vieler Pixel geschieht dabei in Form von zwei ineinander geschachtelten Schleifen und der Stanzkopfroutine in ihnen, was in Abb. 3 dargestellt wird. Anzumerken ist dabei, dass der Wechsel von einem zeitbasierten, sequenziellen Start – Stopp – System der Motoren zu diesem drehwinkelbasiertem Schleifensystem eine Beschleunigung des Prozesses von mehr als 100 Prozent erbrachte. Auf eine weitere Prozessbeschleunigung durch Nutzung des Rückfahrprozesses in x – Richtung zum Stanzen wurde jedoch verzichtet, da das (durch unvermeidbares teilweises Überlagern der Bewegungen) zu unterschiedlich ausgerichteten Schrägen in den Pixeln führen würde, was das Relief unsauberer erscheinen ließe. Um mechanisch, zeitlich und qualitativ optimale Ergebnisse zu erreichen werden außerdem die Motoren der x- und y- Richtung mit 100 Prozent ihrer Leistung, der Motor für den Stanzprozess allerdings nur mit 70 Prozent seiner möglichen Leistung betrieben.

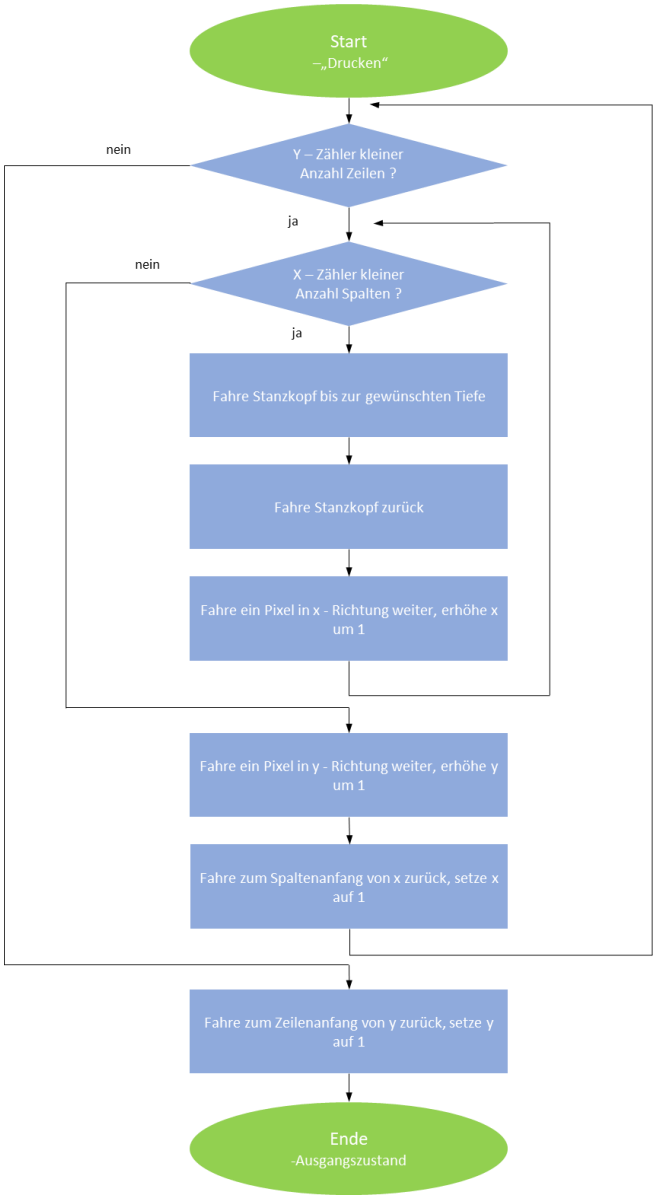


Abbildung 3: Programmablaufplan des Druck-Prozesses

C. Antriebssysteme

Die genaue Ansteuerung einzelner Pixel des Reliefs erfordert die Beweglichkeit des LEGO Systems in x – und y – Richtung des Bildes. Bei beiden Bewegungsrichtungen ist eine möglichst langsame Übersetzung der Motordrehzahlen durch Getriebe förderlich, da dies präzise Bewegungen und somit hohe Bildqualität ermöglicht. Die gesamte Vorrichtung fährt, wie in Abb. 4 zu sehen, auf vier mit Gummireifen bezogenen Rädern, die die Bewegung in y – Richtung ermöglichen. Die Räder werden von Schienen geführt, damit die Fahrspur gerade ist. Um eine zuverlässigere Bewegung zu gewährleisten, fahren die

Räder auf einer ebenen Papierunterlage (rechts im Bild zu sehen), und die durch die Schienen ausgeübten mechanischen Spannungen werden durch Erhöhung der Führung reduziert, wodurch sich der Druck auf die Räder vermindert (links im Bild zu sehen).

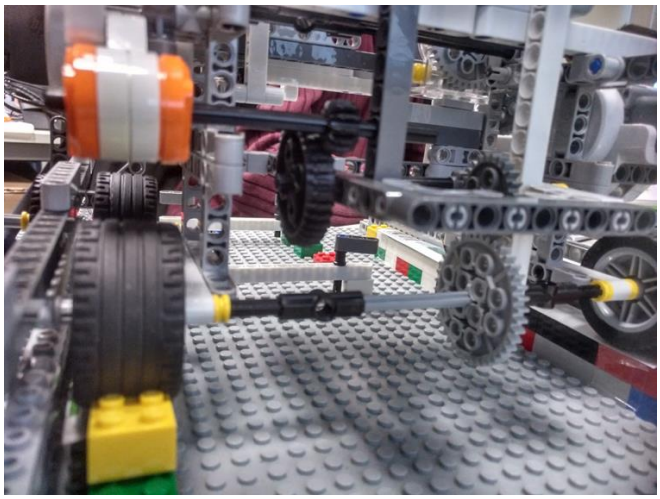


Abbildung 4: Antrieb für die y - Achse

Auf diesem Grundgerüst liegt die Vorrichtung für die Steuerung des Stanzkopfes in x – Richtung, wobei der Motor hierfür statisch an dem Gerüst fixiert ist. Über das in Abb.5 dargestellte Getriebe bewegt er eine Zahnradschiene, an welche der Stanzkopf fest installiert ist. Dessen Gewicht wird dabei durch einen Schlitten getragen, was in Abschnitt „D. Schreibkopfsysteme“ genauer erläutert wird.

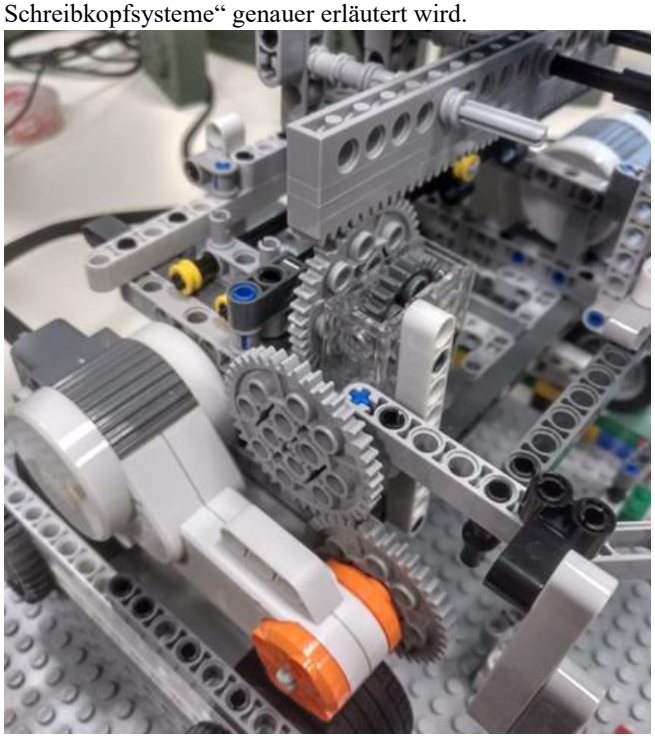


Abbildung 5: Antrieb für die x - Achse

Bei mehrfacher Bewegung des Schlittens summierte sich allerdings ein kleiner Messfehler in den Motoren auf, der zu einer scheinbaren Scherung des Reliefs, wie sie in Abb. 1 zu sehen ist, führte. Dieser Fehler wurde jedoch durch das Anbringen eines Tastsensors behoben, der den Winkelzähler des x - Motors regelmäßig bei Erreichen des rechten Bildrandes auf null zurücksetzt. Ein weiteres Problem war die teilweise Überlagerung von Bewegungen, wie sie bereits in Abschnitt „B. Programm“ angesprochen wurde. Diese führte sowohl dazu, dass die erste Spalte jeder Zeile schief war, als auch dazu, dass die Pixel jeder Zeile entgegengesetzt zu denen der ihr benachbarten Zeilen orientiert waren. Beide Probleme wurden dadurch behoben, dass der Zurückfahrprozess nach rechts keine Stanzungen ausführt, sondern nur der Justierung des Systems dient.

D. Schreibkopfsysteme

Der Schreib– oder genauer Stanzkopf basiert auf einem bereits im vorherigen Unterabschnitt erwähnten Schlitten. Dieser gleitet, wie in Abb. 6 zu sehen ist, durch die horizontale Zahnradschiene angetrieben auf der dafür vorhergesehenen Vorrichtung und trägt die Hauptlast des Stanzkopfes.

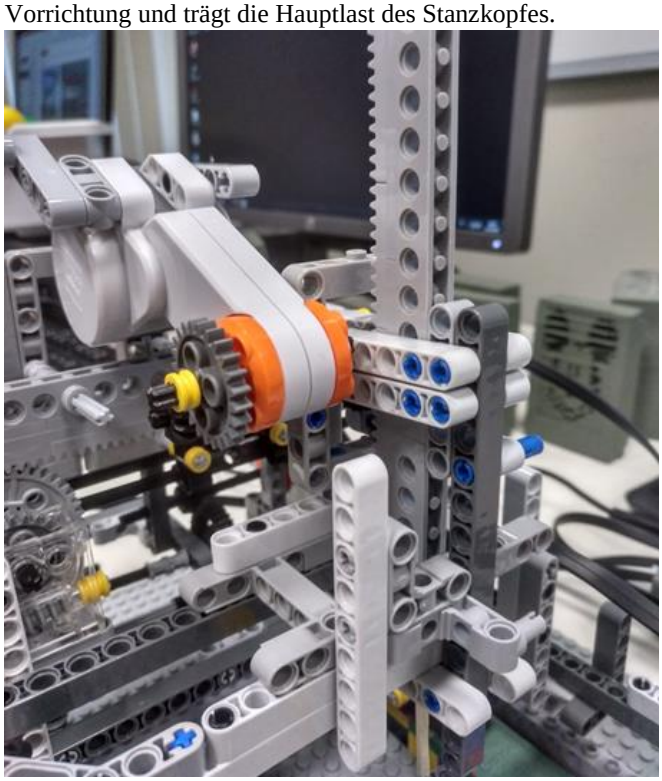


Abbildung 6: Stanzkopf mit Schlitten unter dem Motor

Auf diesem Schlitten ist eine vertikale Zahnradschiene so montiert, dass die Rotation des dritten Motors über ein schnell übersetzendes Getriebe zu einer Auf- oder Abwärtsbewegung der Schiene führt, womit der Stanzprozess ausgeführt wird (siehe Abb. 6). Nachteil der hohen Geschwindigkeit des Stanzprozesses war eine Aufsummierung der Messfehler im Motordrehwinkel, was zu einer allmählichen Fehlplatzierung der Schiene führte. Behoben wurde dieser Fehler durch statistische Messungen, die zeigten, dass eine Verkürzung des Hochfahrprozesses um 20 Grad die Schiene wieder zurück zur Ausgangsposition bringt. Am unteren Ende der Zahnradschiene ist der Stanzkopf montiert. Hierbei kamen zwei Systeme zum Einsatz, die in Abb. 7 zu sehen sind. Die erste Variante bestand aus einem quadratisch geschliffenen Metallstück, das an eine

LEGO – Stange geklebt wurde. Das erlaubte saubere und stets konstante Pixelerscheinungen. Die finale Variante, bestehend aus einem Holzstab, ermöglichte jedoch eine Verdopplung der Bildgröße und Bildqualität, was den höheren Verschleiß des Holzstabes kompensierte. Um gute Bildqualität zu gewährleisten, war auch das Fixieren der Steckmasse am Boden sowie die zusätzliche Verankerung des Gestells mit den in Abschnitt „C. Antriebssysteme“ beschriebenen Führungsschienen erforderlich. Diese sieht man in Abb. 6 im unteren rechten Bildteil.

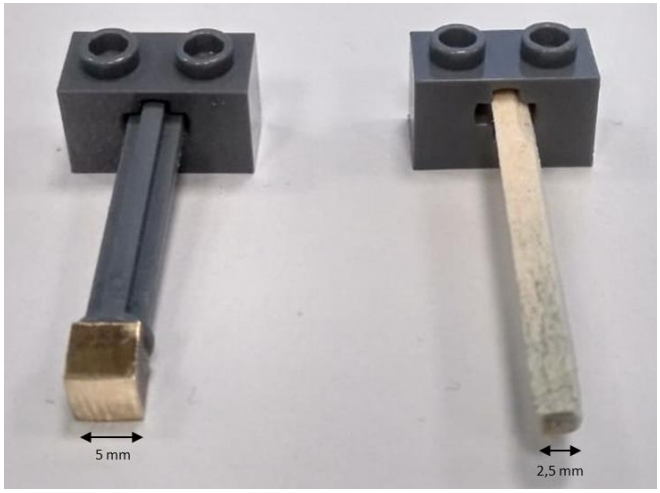


Abbildung 7: links: Metallstanzkopf rechts: Holzstanzkopf

IV. Ergebnissdiskussion

Mit der finalen Version des 3D - Stanzers ist es möglich, schwarz – weiß Bilder im 20 x 30 Pixel Format zu drucken. Die unterschiedlichen Pixeltiefen erzeugen dabei eine gute Schattierung und ermöglichen auch einen dreidimensionalen Bildtiefeneffekt. Wie in Abb. 8 zu sehen ist, haben die Pixel in x – Richtung des Bildes einen optimalen Abstand zueinander. Bei den Pixeln in y – Richtung kommt es bei 50 Prozent der Reihen zu vergrößerten Abständen, was sich in für das Gesamtbild geringfügig störenden Wänden äußert. Dieser Fehler wurde zwar bereits durch langsamere Übersetzungen, glattere Oberflächen und geringere mechanische Spannungen reduziert, konnte jedoch nicht vollständig beseitigt werden.

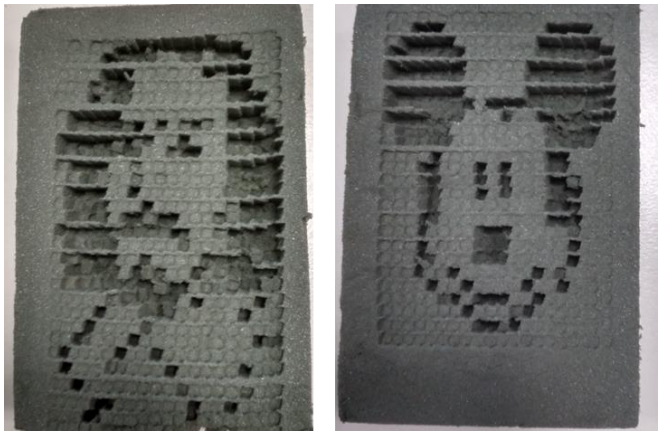


Abbildung 8: Stanzergebnisse, links Otto von Guericke, rechts Cartoon Figur

Für das Stanzen eines Bildes der maximalen Bildgröße benötigt der 3D-Stanzer etwa 12 Minuten, wobei der Prozess bei kleineren und helleren Bildern weniger Zeit beansprucht. Der Verschleiß des Stanzkopfes ist so gering, dass die Auswirkung auf die Bildqualität nach einer Anzahl von mehr als zehn Bildern als Minimal einzustufen ist. Damit produziert die Maschine auch nach längerer Zeit in Betrieb korrekte Bilder. Dies wurde von uns für den veranschlagten Untersuchungszweck hinsichtlich Kosten/Nutzen Verhältnis als ausreichend bewertet. Eine Verbesserung der Standzeit des Werkzeugs wäre durch einen Stanzkopf aus Metall mit den Abmessungen des verwendeten Holzstanzkopfes möglich.

V. Zusammenfassung

Durch den Einsatz von drei Motoren, einem NXT- Modul von LEGO sowie einem Streichholz als Stanzwerkzeug ist es möglich, einen 3D – Stanzer zu konstruieren. Die von der Maschine erzeugten Bilder sind, wenn auch durch dünne horizontale Streifen gestört, stets originalgetreu. Deshalb eignet sich der Stanzer eher zum Darstellen von Schwarzweißbildern als zum Darstellen dreidimensionaler Strukturen, wobei der Bildtiefeneffekt dennoch genutzt werden kann. Das Projekt dient zwar vorrangig dem Simulieren von dreidimensionalen Fertigungsprozessen, weißt aber besonders durch das Einlesen der Stanzdaten über Computerdateien großen Praxisbezug auf. Mögliche Verbesserungen des Systems wären die Verwendung genauerer Motoren sowie der Einsatz von Zahnradschienen an Stelle der Reifen, was aber viel mehr Zahnradschienen und breitere Zahnräder erforderte. Eine weitere deutliche Steigerung der Produktqualität würde der Einsatz einer Fräse als Schreibkopf bringen. Ein solches System könnte vermutlich sogar Anwendungsgebiete außerhalb der Forschung finden.

Literaturverzeichnis

[1] *Universität Trier: Neue Geschäftsmodelle mit 3D-Druck* https://www.uni-trier.de/fileadmin/fb4/prof/BWL/MIT/Download/08-SS_16/2016_09_30_Block_3D_Druck_CityCampus.pdf. Version: Juni 2019

[2] *3DRUCK.COM: Übersicht der aktuellen 3D-Druckverfahren* https://3druck.com/3d-druck-grundkurs/uebersicht-3d-druckverfahren/. Version: Juni 2019

[3] *Schober Technologies: Was ist Stanzen?* https://www.schobertechnologies.de/unternehmen/wissen-kompakt/stanzen/. Version: Juni 2019

[4] *Die J.A.M.M.M.E.R.: J.A.M.M.M.* https://mindstorms.lfb.rwth-aachen.de/index.php/roboter/56-robotswinter2013/roboter-des-jahres-2013/454-jammm128*. Version: März 2019*

[5] Shurrikane: 3D Milling Machine. https://www.youtube.com/watch?v=gYN3RIQwWF0&feature=youtu.be. Version: März 2019

[6] NETZ Konstrukteur: LEGO CNC Fräse erstellt dreidimensionale Objekte. https://netzkonstrukteur.de/lego-cnc-fraese-erstellt-dreidimensionale-objekte/. Version: März 2019

[7] RWTH Aachen: Mindstorms NXT Toolbox. https://www.mindstorms.rwth-aachen.de/. Version: März 2019

Bau einer Multifunktionalen Analoguhr-
Bericht zum Projekt des LEGO Mindstorms Seminar

Leonard Hasler, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

Abstract— Viele wichtige Prozesse der Welt sind zeitabhängig. Daher gibt es schon eine lange Geschichte von Zeitmessern, die in allen Gebieten genutzt werden. In dem folgenden Bericht wird der Aufbau einer Multifunktionalen Analoguhr, die im Zuge des LEGO Mindstorms Projektseminsars an der Otto von Guericke Universität Magdeburg angefertigt wurde, beschrieben. Die Analoguhr stellt mit drei Zeigern die Zeit auf wenige Sekunden genau dar. Mithilfe von Tastsensoren kann eine Stoppuhr oder ein Timer gestartet werden. Bei heutzutage immer genauer werdenden Uhren wird in dem Projekt eine Uhr in der LEGO Technologie entwickelt, die die obengenannten Funktionen aufweist und somit in vielen Gebieten einsetzbar ist.

I. Einleitung

WELTWEIT spielen in allen Gebieten der Industrie, der Wissenschaft und auch im Haushalt die verschiedensten Uhren eine wichtige Rolle. Diese Uhren können nicht nur die aktuelle Uhrzeit anzeigen, sondern auch häufig zu einer bestimmten Zeit stoppen. Insbesondere ist die Genauigkeit der Uhren sehr wichtig, damit Vorgänge richtig getimt werden können. In dem Projekt, welches zusammen mit Patrick Kallow während des LEGO Mindstorms Seminars bearbeitet wurde, wurde eine Analoguhr, die diesen Anforderungen entsprechen sollte, angefertigt. Die Multifunktionale Analoguhr kann die aktuelle Uhrzeit wiedergeben und auch mithilfe von Funktionen die Zeit stoppen oder rückwärts laufen lassen. Die Analoguhr wird mit LEGO unter Verwendung des NXT aufgebaut und in Matlab programmiert. Das von der RWTH Aachen bereitgestellte Toolkit, mit dem auf die Motorausgänge und Sensoreingänge des NXT unter Matlab zugegriffen werden kann, wird in dem Projekt genutzt. Die Funktionen der Uhr können mithilfe von zwei Sensoren ausgewählt werden.

II. Vielfalt der uhren

A. Geschichte der Uhren

Schon seit mehreren Tausend Jahren benutzen Menschen Zeitmesser, um sich an einem Tag zeitlich orientieren zu können. Genauer gab es vor mehr als 5000 Tausend Jahren in Ägypten [1] die erste bekannte Uhr, die heutzutage als Sonnenuhr bezeichnet wird. Aufgrund der Veränderung des Sonnenstandes traf das Sonnenlicht unterschiedlich auf einen Stab und mit dem Schatten des Stabes war eine Zeit ablesbar. Über die Jahre wurden weitere Uhren erfunden, unter anderem die Wasseruhr, Sand uhr und die Räderuhr. Die Räderuhr ist die erste mechanisch gefertigte Uhr. Sie beinhaltet eine Unrast, die einen annähernd gleichmäßigen Uhrenlauf ermöglicht. Jedoch war die Räderuhr noch sehr ungenau und recht grob. Die Unruh löste die Unrast ab und mit der Erfindung der Spiralfeder begann im 15. Jahrhundert die Entwicklung eines präziseren Uhrwerks. Damit entstanden die heute bekannten Taschenuhren und Armbanduhren. Im 19. Jahrhundert entstanden die ersten Digitaluhren, welche die Zeit durch den Ziffernwechsel direkt anzeigen, an Stelle von Zeigern.

Abbildung: 1-Atomuhr CS2 der PTB in Braunschweig [2]

B. Die Atomuhren

Mitte des 20. Jahrhundert entstand die erste Atomuhr, das besondere einer Atomuhr ist, dass einzelne Atome den Takt der Uhr angeben. Die erste Cäsium- Atomuhr wurde Ende der 1950er Jahre auf den Markt gebracht. Die Cäsium- Atomuhr funktioniert dadurch, dass zunächst Cäsium in einem Ofen verdampft wird und anschließend in einem Vakuumtank zu einem Atomstrahl gebündelt wird. Die Atome sind dann in einem der beiden tiefstmöglichen Energiezustände, welche das Cäsium einnehmen kann. Danach werden die Atome magnetisch sortiert, und durchqueren einen Hohlraumresonator in dem ein magnetisches Mikrowellenfeld herrscht. Die Atome wechseln mit einer gewissen Wahrscheinlichkeit ihren Zustand, abhängig von der Frequenz des Mikrowellenfeldes. Die Atome, die ihren Zustand geändert haben, werden registriert. Mit dieser Uhr kann eine Sekunde genauer beschrieben werden, und zwar ist eine Sekunde genau nach 9.192.631.770 Perioden des Mikrowellenfeldes verstrichen [3]. In dem Projekt wird die Analoguhr von der Genauigkeit nicht ansatzweise mit der Atomuhr vergleichbar sein. Jedoch ist das Ziel des Projekts, die Genauigkeit der Uhr, im Rahmen der LEGO Mindstorms Technologie, so gut wie möglich anzupassen.

DOI:10.24352/UB.OVGU-2019-054

III. Anfertigung der Uhr

A. *Technische Umsetzung*

Die LEGO Mindstorms Analoguhr besitzt die Form eines Würfels. Die Größe ist so ausgewählt, dass sie nicht allzu groß ist, jedoch trotzdem ausreichend Platz für den NXT Motor und die Zahnräder ist. Die zwischen dem Motor und den Zeigern die Kraft übersetzen. Auf der Rückseite der Analoguhr sind zwei separate Motoren angebracht, der untere Motor ist für den Sekundenzeiger verantwortlich.

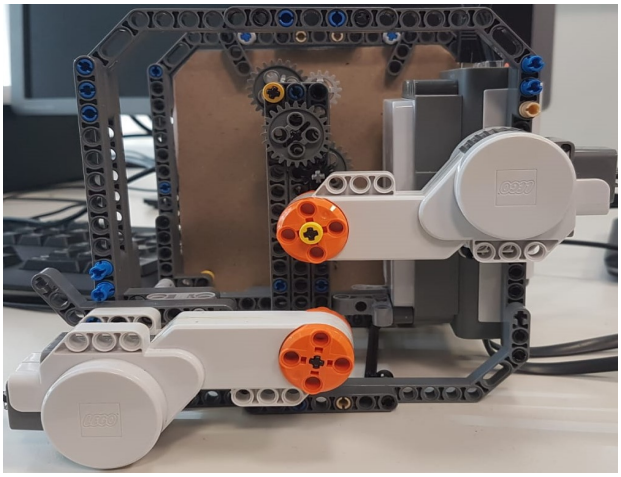


Abbildung 2- Rückseite der Multifunktionalen Analoguhr

Für das Bewegen des Sekundenzeigers wird keine Übersetzung genutzt, sondern der Motor bewegt eine LEGO Stange im Sekundentakt. Am anderen Ende der Stange befindet sich der Sekundenzeiger. Der obere Motor bewegt den Minuten- und den Stundenzeiger. Der Minutenzeiger wird, wie der Sekundenzeiger, direkt vom Motor angetrieben. Andererseits wird die Stange des Minutenzeigers über ein Zahnrad mit der Übersetzungseinheit verbunden. Dabei wird eine Übersetzungsrate von 1:12 benutzt, die mit der Gleichung (1) zu berechnen ist. Die Übersetzungsrate ergibt sich aus zwei 1:2 und einer 1:3 Übersetzung. Bei der Übersetzung werden 8,12 und 24 zähnige LEGO Zahnräder benutzt.

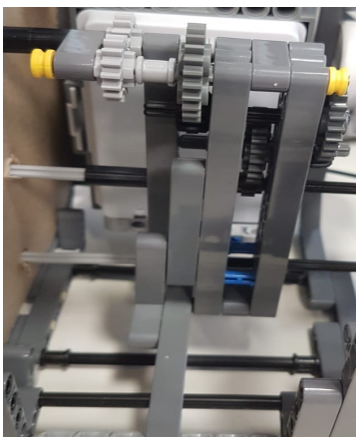


Abbildung 3- Das Räderwerk von der NXT abgewandten Seite

An der Vorderseite ist eine beklebte Pappe fixiert, auf der drei Ziffernblätter dargestellt sind. Der NXT ist an einer Seite angebracht und sorgt zusätzlich dafür, dass der Schwerpunkt relativ mittig liegt. Außerdem sind die beiden Tastsensoren an der Kante zwischen NXT und dem Ziffernblatt angebracht, die Bedienung und die Funktionen der Tastsensoren wird im nächsten Abschnitt weiter erläutert.

B. *Programmumsetzung*

Das in Matlab geschriebene Programm wird über ein USB- Kabel in den NXT übergeben und treibt die Uhr an. Das Programm besteht aus einigen Funktionen und verbindet die Uhrzeitanzeige mit den beiden weiteren Funktionen. Bei Programmstart stellt sich die Uhr auf die Systemzeit des Rechners, auf dem das Programm läuft, ein. Die Systemzeit kann sich leicht unterscheiden, bei Windows Rechnern kann man den Zeitserver auswählen und dann kann die Systemzeit auf die Millisekunde genau sein, denn manche Zeitserver gehen nach der Atomuhr. Die Funktion bewegt die Zeiger durch zwei verschachtelte For- Schleifen nach genau 0.96 Sekunden und Rechenzeit um sechs Grad. Die Zeit ergibt mit Abweichung möglichst genau eine Sekunde, so dass der Sekundenzeiger sehr genau läuft.
Bei Bedienung eines der beiden Tastsensoren wird zunächst die Zeit zurückgesetzt, dass bedeutet die Zeiger stellen sich auf die Ausgangslage und zeigen nach oben. Wenn als nächstes der obere Tastsensor betätigt wird, bewegen sich die Zeiger weiter, bis zum nächsten Betätigen des oberen Tastsensors, dann stoppt die Uhr in wenigen Millisekunden. Die Stoppuhr kann mit Drücken des oberen Sensors weiterlaufen oder mit dem unteren Tastsensor wird wieder auf die Systemzeit angepasst und die Uhrzeit angezeigt. Die Timer- Funktion wird gestartet, wenn die aktuelle Uhrzeit angezeigt wird und dann der untere Tastsensor betätigt wird. Nach Zurücksetzen der Zeiger ist es möglich den Sekundenzeiger auf eine gewünschte Zeit zustellen. Bei nochmaliger Betätigung des unteren Tastsensors läuft der Zeiger entgegen dem Uhrzeigersinn bis er wieder auf Null steht. Bei Erreichen des Startpunktes wird ein Ton abgespielt, der das Beenden des Timers akustisch hervorhebt. Wenige Sekunden später stellt sich die Uhr wieder auf Systemzeit ein und läuft wie eine gewöhnliche Analoguhr. Zur Veranschaulichung der erklärten Funktionsweise befinden sich die Programmablaufpläne im Anhang (2).

C. *Gleichungen*

$$i = \frac{z_2}{z_1} \tag{1}$$

i = Übersetzungsverhältnis
z_1 = Zähnezahl des treibenden Rades
z_2 = Zähnezahl des getriebenen Rades

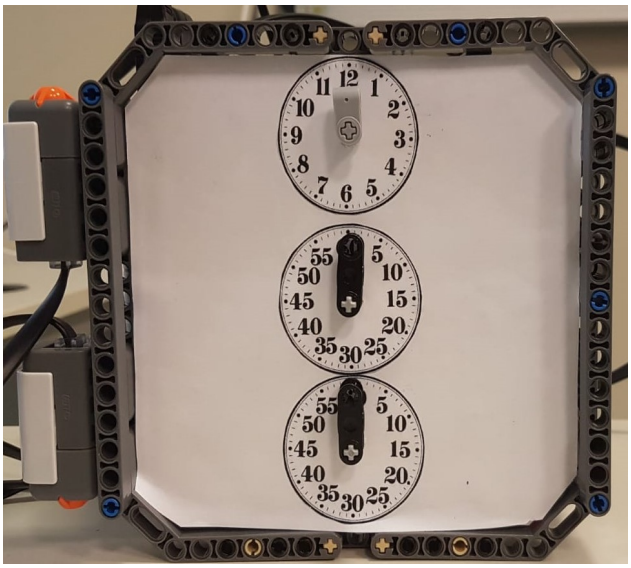


Abbildung 4- Die fertiggestellte Uhr

IV. Ergebnisdiskussion

Am Ende des Projekts erfüllt die Multifunktionale Analoguhr die gewünschten Anforderungen, dies beinhaltet die möglichst genaue Uhrzeitanzeige, sowie die Stoppuhr- und Timerfunktion. Auf dem Weg dahin sind einige Probleme aufgetreten, die alle weitestgehend gelöst wurden sind. Beim Aufbau der Uhr ergaben sich Probleme beim Auffinden bestimmter LEGO Teile, deshalb sind einige Teile nicht ganz farblich symmetrisch eingebaut, dies stört aber keineswegs die Funktionalität der Analoguhr. Beim Programmieren ist es vorgekommen, dass sich die Motoren und Zeiger endlos bewegen. Dieses Problem entstand dadurch, dass das sogenannte TachoLimit auf null oder einen negativen Wert verändert wurde, durch cleveres Kürzen oder Erweitern der Variablen ist es möglich, das Problem zu bewältigen. Ein etwas schwerwiegendes Problem ist die Überlastung des Motors oder des NXT. Dies trifft zu, wenn in zu kleinen Zeitabständen Bewegungen ausgeführt werden sollen. Die Problemlösung sind kleine Pausen die zur minimalen Verzögerung dienen und dem Fluss des Programms helfen. Das größte Problem, was den Unterschied zu den am Beginn genannten Uhren ausmacht, ist jedoch die Ungenauigkeit der Lego Motoren. Die LEGO Motoren haben nur einen Wirkungsgrad von ca. 95%, dadurch gibt es nach mehreren Umdrehungen der Zeiger schon deutliche Abweichungen. Deshalb ist die Analoguhr nicht wirklich in der Genauigkeit vergleichbar mit den modernen Uhren. Nach Ändern der Toleranzgrade konnte das Problem leicht verbessert werden, aber die Ungenauigkeit kann nicht vollständig behoben werden. Innerhalb der Lego Technologie ist es gelungen, die Uhrzeit möglichst genau darzustellen und die Funktionalität der Stoppuhr und des Timers ist gewährleistet.

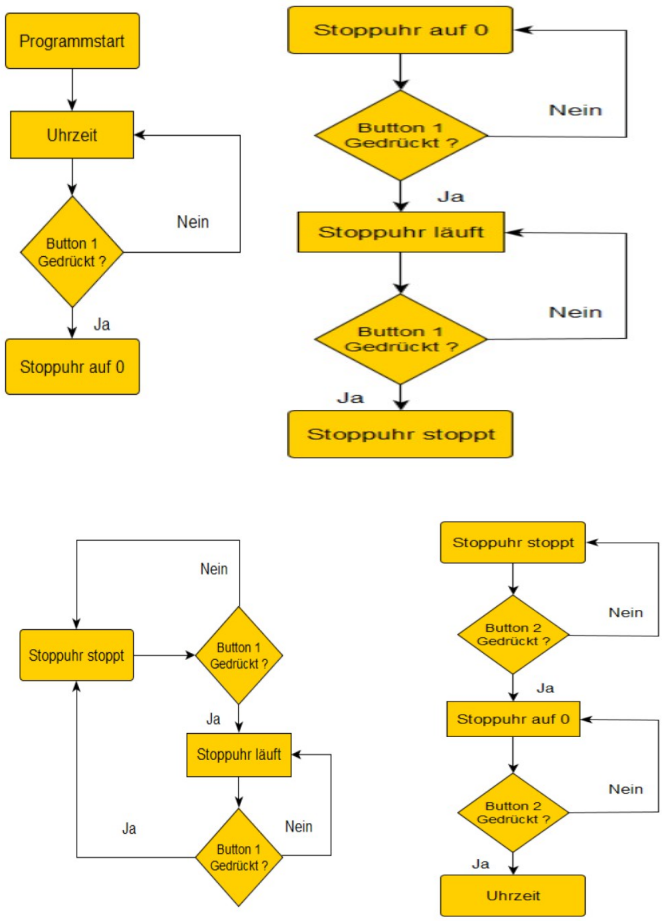
V. Zusammenfassung und Fazit

Alles in allem ist im Rahmen des Projekts eine funktionierende Analoguhr entstanden. In der Praxis existieren bereits Modelle mit gleicher Funktionsweise, die eine genauere Zeit anzeigen können. Das Ziel der LEGO Uhr war mit der vorhandenen LEGO Technologie eine solche Uhr zu bauen. Dies ist gelungen trotz den in Punkt IV genannten Problemen und Einschränkungen. Die Größe ist so bestimmt, dass man die Uhr leicht handhaben kann und sie so in verschiedenen Bereichen einsetzbar ist. Es bestehen einige Verbesserungsmöglichkeiten bei der Entstehung einer solchen Analoguhr. Einerseits ist es möglich, das Programm auf den NXT zuspielen. Dann könnte man die Übertragungszeit sparen, aber der NXT bräuchte eine genaue Systemzeit, die er in diesem Fall nicht hatte. Außerdem wären zusätzliche Funktionen möglich gewesen, die über weitere Tastsensoren bedient werden könnten. Z.B.: Ein Sekundenmodus, der bedeutet die genauere Anzeige einer Zeitspanne, in dem der Sekundenzeiger sich schneller bewegen würde. Diese Erweiterungen konnten in der zu Verfügung stehenden Zeit des Projekts nicht realisiert werden.

Anhang

1. Quellcode des Uhrzeitmodus

2. Programmablaufplan



LITERATURVERZEICHNIS

[1]Was War Wann?: *Geschichte der Uhr.* https://www.was-war-wann.de/geschichte/geschichte-der-uhr.html. Version: März 2019

[2]Carolin Pirich: *Die Eigenzeit der DDR.* https://www.deutschlandfunkkultur.de/die-eigenzeit-der-ddr.984.de.html?dram:article_id=153461. Version: März 2019

[3]Andreas Bauch: *Wie funktioniert eine Atomuhr?.* https://www.spektrum.de/frage/wie-funktioniert-eine-atomuhr/590812. Version: März 2019

Multifunktionale Analoguhr

Patrick Kallow, Mechatronik
Otto-von-Guericke-Universität Magdeburg

***Abstract*—Im Beginn des Papers wird der Grund erläutert, weswegen sich für genau dieses Projekt entschieden wurde. Zusätzlich werden die verschiedenen Anwendungsbereiche erklärt und es wird die allgemeine Idee und deren Aufbauvorstellung beschrieben. Indes werden in den Vorbetrachtungen einzelne Beispiele, welche dem Projekt der Multifunktionalen Analoguhr ähneln beschrieben und es wird ihr grundsätzlicher Aufbau und teilweise dessen Inspirationsquellen erörtert. Der Hauptteil befasst sich grundsätzlich mit der Umsetzung des Projekts und dessen Funktionsweise. In der Ergebnisdiskussion wird das endgültige Produkt diskutiert und am Ende wird ein Fazit gezogen.**

***Schlagwörter*: Lego, NXT, Stoppuhr, Timer, Uhr,**

I. Einleitung

Es wurde versucht mithilfe von NXT Bausteinen und Matlab eine analoge Uhr umsetzten, welche sowohl einen Timer als auch eine Stoppuhr enthält.
Damit man die Möglichkeit besitzt, die Zeit mit verschiedenen Methoden zu messen und die multifunktionale Analoguhr in möglichst vielen verschiedenen Bereichen nutzen zu können.
Diese würde in allen Anwendungsbereichen genutzt werden können, wo eine Uhrzeit, Timer oder eine Stoppuhr benötigt werden würde. Wobei es in diesen mithilfe der analogen Uhren nun möglich sein würde, genaue Zeitmessungen durchzuführen. Diese könnte man z.B. beim Kochen, in der Logistik und für die ungefähre Zeitmessung in Praktikumsversuchen nutzen. Des Weiteren wurde versucht die Uhr so stabil wie möglich zu konzipieren, um längeren Gebrauch standhalten zu können und zusätzlich die allgemeine Handhabung zu erleichtern.

II. Vorbetrachtungen

Beginnend werden nun einige ähnliche Projekte besprochen, welche bereits realisiert wurden. Dabei wird gezeigt welche vorteilhaften Eigenschaften von der multifunktionalen Analoguhr teilweise übernommen wurden und was die jeweiligen Probleme und Nachteile der einzelnen Projekte sind. Wodurch erkennbar wird wie sich die multifunktionale Analoguhr von des anderen Unterscheidet und heraussticht.

A. Lego Mindstorms Roboclock

In dieser Lösung [1] hatte der Ersteller eine Analoge NXT Uhr gebaut welche die Zeit wie eine gewöhnliche Tisch Uhr anzeigt. Jedoch konnte diese nur die Sekunden korrekt anzeigen und die Minuten stimmen nicht mit der bereits vergangenen Zeit überein und waren daher nur als kosmetischer Zusatz zu betrachten. Die multifunktionale Analoguhr hat sich hauptsächlich dessen statische Struktur zu eigen gemacht, um die multifunktionale analog Uhr möglichst kompakt und robust zu halten. Außerdem bot das Gerüst genügend Platz, damit es möglich war, das Zahnradnetzwerk für die einzelnen Zeiger zu konzipieren. Dabei können diese zusätzlich als Halterung genutzt werden.

B. Time Twister 3

Diese Lösung hatte eine Digitaluhr umgesetzt welche mithilfe von Gummibändern und sich drehenden Schrifteinzelteilen eine digitale Zeitanzeige ergab. Dies war jedoch nur auf die Minute genau, da sie keine Sekunden anzeigen konnte. Deshalb hatte die Uhr außerdem eine Verzögerung von mehreren Sekunden, wenn sich die Zeit umstellte, da die Einzelteile zu lange gebraucht hatten, um sich in die gewünschte Position zu begeben [2]. Dies war der Grund für den Anreiz eine Analoguhr zu konstruieren.

C. Lego Cuckoo clock

Diese von Lego entwickelte Kuckucksuhr konnte universell die Zeit anzeigen und alle 15 Minuten einen Kuckuck erscheinen lassen [3]. Sie hatte jedoch weder eine eingebaute Stoppuhr noch einen Timer. Sie zeigte jedoch die Möglichkeit ein Signal oder einen Ton nach dem Ablauf einer bestimmten Zeit ertönen zu lassen, sodass mithilfe dieser Grundlage der Timer entwickelt wurde. Welcher im Fall der multifunktionalen Analoguhr mit dem Prgramm Matlab umgesetzt wurden und nicht mit dem von Lego Mindstorms zur Verfügung gestellten Programm.

DOI: 10.24352/UB.OVGU-2019-055

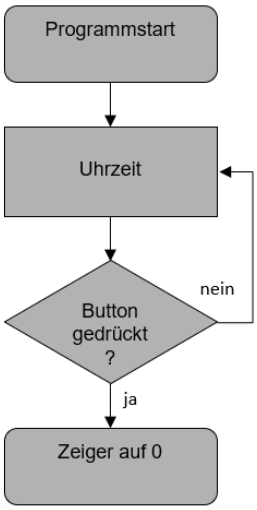


Abbildung 1: Programmablaufplan für den Anfangsteil des Programms

III. Hauptteil

Die multifunktionale Analoguhr sollte in der Lage sein, bei Einschaltung automatisch die Zeit anzuzeigen und zusätzlich die Funktionen einer Stoppuhr und die eines zu Timers besitzen, wodurch der Benutzer der multifunktionalen Analoguhr mehrere Möglichkeiten zur Messung des Zeitverlaufs hatte.

Der Zeitverlauf sollte mit einem Ziffernblatt, welches an der Vorderseite der Uhr befestigt wurde, ablesbar sein. Die multifunktionale Analoguhr musste zusätzlich zwei Tastsensoren besitzen, wobei der eine zum Aktivieren des Timers diente und der andere zum Aktivieren der Stoppuhr.

Somit war die multifunktionale Analoguhr so konzipiert, dass sie aus einem quadratischen Gerüst bestand, welches an den Ecken abgerundet war. Auf der linken Seite befand sich der Nxt Block, welcher mit den zwei Motoren im hinteren Teil verbunden wurde. Zusätzlich befanden sich links vor dem Nxt Block zwei Tastsensoren welche entgegengesetzt voneinander befestigt waren. Wenn die Uhr nun gestartet werden würde, schalten sie automatisch in den Modus „Aktuelle Uhrzeit“ ein. Dabei würden sich die Zeiger nach der Uhrzeit der internen System Clock des jeweiligen Rechners richten.

Die Uhr würde nach der Aktivierung so lange die Uhrzeit anzeigen, bis sie entweder einen neuen Befehl durch die Tastsensoren bekommt oder die Uhr ausgeschaltet wird.

Falls nun der obere Tastsensor betätigt werden würde, werden alle Zeiger auf die höchste Position gesetzt (Sekunde 60, Minute 60, Stunde 12) und der Stoppuhr Modus würde sich aktivieren. Falls nun wieder der obere Tastsensor betätigt würde, beginnt die Stoppuhr zu laufen und die nach dem Betätigen des Tastsensors vergangene Zeit anzuzeigen. Wenn der obere Tastsensor bei laufender Stoppuhr nun wieder gedrückt wird, dann würde die Stoppuhr stehen bleiben bis entweder der obere Tastsensor betätigt wird und sie weiterlaufen würde oder der untere Tastsensor, wodurch die Uhr wieder in den Modus „Aktuelle Uhrzeit“ wechseln würde und wieder die Uhrzeit der System Clock anzeigt. Würde während des „Aktuelle Uhrzeit“ Modus nun der untere Tastsensor betätigt, würde die Uhr sich wieder in den Stoppuhr Modus wechseln und die Zeiger würden wieder in die höchste Position gesetzt.

Nun kann der Benutzer den Sekundenzeiger in die gewünschte Position bringen und wenn dieser den Zeiger ein weiteres Mal betätigt, zählt der Timer bis auf 0 gibt dann einen Alarm aus und wechselt wieder in den „Aktuelle Uhrzeit“ Modus.

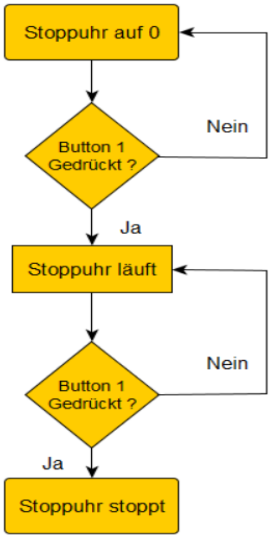


Abbildung 2: Programmablaufplan nach Betätigen das Stoppuhr Tastsensors

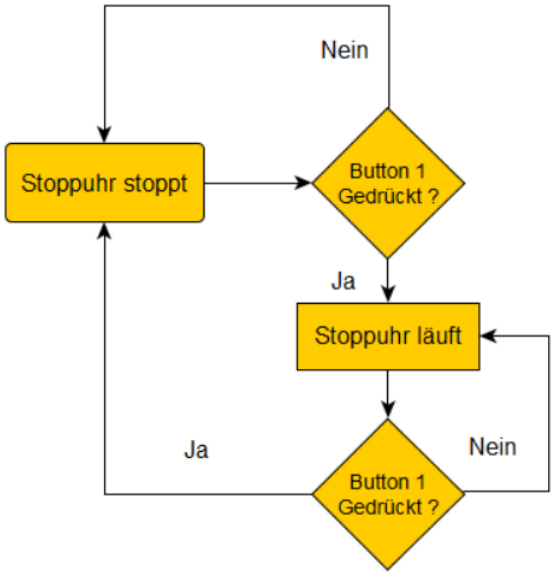


Abbildung 3: Programmablaufplan für die Stoppuhrfunktion

Die multifunktionale Analoguhr ist quadratisch aufgebaut und besitzt an den Ecken Abrundungen ähnlich zu der „Lego Mindstorms Robo Clock“ um sie Stabil und handlich zu hallten. Sie besitzt 2 Motoren der untere steuert die Sekunden Zahl indem er sich dem Matlabprogramm entsprechend, jede Sekunde um 6° dreht. Der zweite Motor, welcher sich in

gespiegelter Form über dem ersten befindet, ist an der Hauptachse mit dem Minutenzeiger verbunden und bewegt sich wie auch der Sekundenmotor dem Matlabprogramm entsprechend jede Minute um 6˚.
Diese Hauptachse ist zusätzlich mit einem Zahnradsystem von 8 Zahnrädern verbunden welche zusammen eine übersetzungsrate von 1:12 ergeben. Dies wurde zuerst mit dem Übersetzungsverhältnis von 12:24 bei der ersten Übertragung erreicht. Bei der zweiten wurden dann nochmal die gleichen Zahnräder verwendet welche wieder eine Übersetzungsrate von 12:24 besitzen und bei der letzten Zahnradkombination wurde eine Übersetzungsrate von 6:18 gewählt wodurch nun die gewünschte übersetzungsrate von 1:12 erreicht wurde was in Gleichung 1 zu erkennen ist.

Übersetzungsrate der einzelnen Zahnräder

$$x = 12 = \frac{24}{12} \cdot \frac{24}{12} \cdot \frac{18}{6} = 2 \cdot 2 \cdot 3 \quad (1)$$

Die Zahnradverküpfungen wurden seitlich übereinander versetzt aufgebaut und sind mit Achsen, welche die Momente und Kräfte übertragen verbunden. Dies wurde einerseits getan um Platz zu sparen, da dieser innerhalb der Uhr begrenzt ist und außerdem um die innere Struktur der Uhr zu verstärken, denn durch die senkrecht angebrachten Halterungen für die Achsen der Zahnräder, ist die Stabilität erheblich gestiegen.


Abbildung 4: Zahnradsystem für die Zeiger

Anschließend wurden zur weiteren Verstärkung des Gehäuses Stützstreben am unteren und oberen Ende der Uhr angebracht.

Nun wurde ein Stück Pappe zur Front der Uhr zurechtgeschnitten und mit Zifferblättern versehen, um die Lesbarkeit zu erhöhen. Diese wurden dann mittig zu ihren jeweiligen Anzeigen angebracht. Schlussendlich wurde ein längliches Lego Teil auf die Achse geschoben welches den Zeiger darstellt.

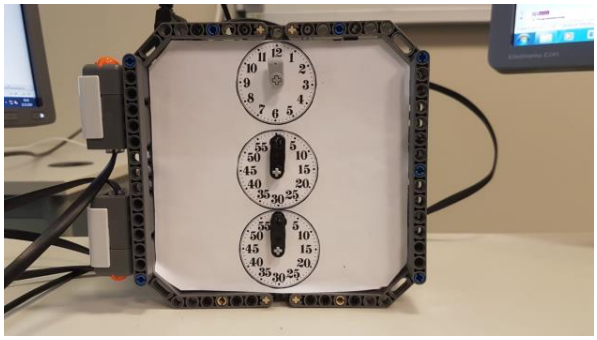


Abbildung 5: Zeigersystem

IV. Ergebnisdiskussion

In der Ergebnisdiskussion soll erläutert werden, was bei dem Projekt als Endergebnis herausgekommen ist und welche Probleme dabei aufgetreten sind.
Schlussendlich wurde die multifunktionale Analoguhr fertiggestellt. Die Uhr ist in der Lage die aktuelle Uhrzeit anzuzeigen, eine Stoppuhr und einen Timer einzustellen. Die größten Problematiken waren dabei das Zusammensuchen der verschiedenen Zahnräder um die Übersetzungsrate 1:12 von Minute zu Stunde umzusetzen. Außerdem war es problematisch, ausschließlich aus Lego ein leicht lesbares Ziffernblatt umzusetzen. Dies wurde damit gelöst, ein Ziffernblatt aus Papier und Pappe für die Uhr auszuschneiden und hinter den Zeigern zu befestigen.
Ein weiteres Problem war es, die Stärke der Motoren richtig einzustellen, damit diese sich im richtigen Zeitintervall bewegen, was durch mehrmaliges Testen gelöst wurde.
Das letzte Problem, welches gelöst werden musste, war die Instabilität der inneren Struktur da sich das Lego bei dem rücksetzten der Zeit oder dem Einschalten der Timer oder Stoppuhrfunktion stark verbog denn welches vergleichsweise schnell geschehen musste. Dies wurde gelöst indem weitere Stützstrukturen an der Uhr angebracht wurden und 2 Zahnradkombinationen geändert

V. Zusammenfassung und Fazit

Größtenteils wurden die meisten Ziele erreicht und die Hauptfeatures alle umgesetzt.
Lediglich das Feature des Timers eine hundertste Sekunde anzeigen zu können, war aufgrund der fehlenden Zeit und der Schwachen Struktur der Lego Teile nicht zu realisieren, da diese nicht der hohen Drehzahl standhalten konnten und sich schnell plastisch verformt haben.

Verweis auf die Aussagen der Literaturquellen

Laut [1] ist die Roboclock eine gewöhnlich funktionierende Tischuhr
[2] Nachteile des Time Twisters 3
[3] Funktion der Lego Cuckoo clock

LITERATURVERZEICHNIS

[1] Artem 16. (Juni 29, 2017, 16:12) letzter Besuch: 18.3.2019 Rebrickable forum [Online]. Account data: https://rebrickable.com/users/Artem%2016/mocs/

[2] Hans Andersson (Februar 2, 2014) letzter Besuch: 18.3.2019 tiltedtwister.com/tiltedtwister3 [Online]

[3] Lego Group. letzter Besuch: 18.3.2019 https://www.lego.com/en-us/mindstorms/build-a-robot/cuckoo-clock

Balance Bot – ein unkonventioneller Segway

Tobias Alexander Nickel, Elektrotechnik/Informationstechnik
Otto-von-Guericke-Universität Magdeburg

Zusammenfassung— **Im Zuge der Globalisierung ist es immer wichtiger geworden Güter, die um die Welt geschickt werden sollen, in Lagerhallen aufzubewahren. Hierbei ergibt sich jedoch das Problem, dass mehr Platz für die ganzen Waren benötigt wird. Häufig werden zum Einlagern der Waren große Gabelstapler benötigt, welche jedoch viel Platz in Anspruch nehmen und somit weniger Stellfläche für Lagerregale vorhanden ist.**
Eine Lösung dieses Problems ist ein auf nur zwei Rädern aufrecht fahrender Roboter, welcher nur mit einem Lichtsensor auskommt. Im Folgenden wird dessen Entwicklung und Funktionsweise genauer erläutert, unter der Zielstellung, Lagerungsabläufe mittels des Balance Bots zu optimieren.

***Schlagwörter*— Lego Mindstorms, Lichtsensor, P-I-D-Regler, Regelungstechnik, Segway, Segwayroboter**

I. Einleitung

Zur heutigen Zeit ist es nichts neues mehr Roboter als Hilfsmittel einzusetzen. In vielen Bereichen, wie zum Beispiel in der Automobilindustrie oder in der Hardwareproduktion, kommen Roboter zum Einsatz, weil die zu erledigenden Aufgaben für Menschen nicht mehr machbar, zu gefährlich oder auch zu zeitaufwendig sind. Bei genauerer Betrachtung der Einsatzgebiete von helfenden Robotern fällt auf, dass in Lagerhallen immer noch Gabelstapler benutzt werden um Waren in und aus den Regalen zu laden. Dafür werden Arbeitskräfte benötigt und die Gabelstapler nehmen sehr viel Platz ein, sodass die Lagerregale weit auseinander stehen müssen, damit der Gabelstapler dort durchpasst. Auch Inventuren werden oftmals noch von Hand durchgeführt und sind deswegen sehr Zeitaufwendig. In seltensten Fällen kommen Drohnen zum Einsatz, welche auf der einen Seite zwar die Arbeiter entlasten, aber auf der anderen Seite recht teuer sind.
Im Rahmen des Lege Mindstorms Projektseminars vom 11.02.2019 bis 22.02.2019, ist in Zusammenarbeit mit Carlo Schafflik der Blanace Bot entstanden. Dieser ist in der Lage auf zwei Rädern zu fahren und das nur mit Hilfe eines einzelnen Lichtsensors. Deswegen ist der Balance Bot verglichen mit einem Gabelstapler platzsparender und günstiger als eine Drohne. Trotzdem ist er in der Lage waren zu Transportieren und ausgestattet mit einem Barcodescanner kann er auch Inventuren durchzuführen.
Zur Umsetzung wurde ein NXT 2.0 mit einer passenden Software der RWTH-Aachen und als Programmiersprache Matlab benutzt. Im Folgenden werden Entwicklung, Aufbau und Funktionsweise des Balance Bot behandelt.

II. Vorbetrachtungen

Für die Realisierung der Idee wurde aus der Regelungstechnik ein P-I-D-Regler verwendet und Grundlagenwissen sowohl in Matlab als auch in Lego Mindstorms NXT 2.0.

A. Segwayfunktionsweise

Ein Vorbild für den Balance Bot war ein Segway. Dieser bewegt sich auch auf zwei Rädern und kann aufgrund von Gewichtsverlagerung vor- und rückwärts fahren und korrigiert Störfaktoren von außen sodass er immer im Gleichgewicht ist und nicht umkippen kann. Um dies zu realisieren ist ein Gyroskop verbaut, welches die genaue Lage des Segways bestimmen kann und die Korrektur von Störfaktoren an die Elektromotoren in den Rädern weiterleitet [1].

B. P-I-D-Regler

Ein Regler dient der Korrektur von Fehlern durch Störeinflüsse und setzt Werte auf einen festgelegten Ausgangspunkt zurück. Der P-I-D-Regler ist dabei einer von 3 verschiedenen Regelungsarten. Hierbei steht das P für den Proportionsanteil. Das bedeutet das Ausgangsignal ist proportional zum Eingangssignal. Das I steht für den Integralanteil, welcher das Eingangssignal aufintegriert und an das Ausgangssignal weiterleite. Das D steht für den Differenzialanteil. Dieser nimmt die Ausgangsgröße proportional zu der Änderungsgeschwindigkeit der Eingangsgröße. Alles zusammen bildet den P-I-D-Regler und dieser kann vereinfacht beschrieben, den aktuellen Fehler korrigieren, vergangene Fehler vorbeugen und zukünftige Fehler vermeiden. Somit ist im Idealfall der Ausgangspunkt immer gleich egal wie oft, schnell und stark das System, in dem der Regler angewendet wird, beeinträchtigt ist[2].

C. Grundlagen Programmierung

Wie schon oben in der Vorbetrachtung erwähnt, wurden als Programmiersprachen Matlab und Lego Mindstorms NXT 2.0 genommen. Grundsätzlich war gedacht, dass der Matlab Code den NXT Stein die Befehle via Bluetooth übergeben soll. Dabei ergaben sich einige Probleme was die Ausführungsgeschwindigkeit der Befehle betrifft. Aus diesem Grund wurde Matlab mit dem von Lego gestelltem Programm Lego Mindstorms NXT 2.0 ersetzt, um Unterschiede ziehen zu können. Genaueres folgt im Hauptteil D. Programmierung.

10.24352/UB.OVGU-2019-056

III. HAUPTTEIL

A. *Funktionsweise des Lichtsensors*

Bei dem Lichtsensor unterscheidet man zwischen zwei Modi. Den passiven und den aktiven Modus. Für den Balance Bot haben wir den aktiven Modus benutzt. Das heißt die rote LED leuchtet und das reflektierte Licht wird gemessen. Dabei geht der Helligkeitsbereich von 0 bis 1023.
Hierbei wurde sich für den aktiven Modus entschieden, weil der Sensor somit weniger anfällig gegenüber äußeren Lichtquellen ist. Nur das direkte Anleuchten mit einer Smartphonetaschenlampe verändert den Helligkeitswert.
Zu Beginn wird der Balance Bot gerade aufgerichtet. Damit er immer aus derselben Lage gestartet wird, wurde eine Halterung gebaut die den Roboter jedes Mal aus der exakt gleichen Position anfangen lässt. Das ist wichtig damit man immer denselben Bezugspunkt hat und die Reglerwerte anpassen kann. Ist er gestartet, wird der in der aktuellen Position gemessene Lichtwert als Ausgangspunkt genommen. Kippt der Roboter nach vorne, so wird mehr Licht empfangen, weil der Lichtsensor näher an den Boden kommt. In dem Moment muss der Roboter nach vorne Fahren, damit der Sensor wieder den Ausgangspunkt erreicht. Kippt der Roboter nach hinten, so wird weniger Licht empfangen, weil der Lichtsensor weiter vom Boden entfernt ist. In dem Fall muss der Balance Bot nach hinten fahren, um den Lichtsensor zum Ausgangspunkt zu bringen. In Abbildung 1 ist der Lichteinfall beim Lichtsensor schematisch dargestellt.
Die ganze Logik hinter den Aktionen steckt im Quellcode welcher einen Regler beinhaltet (siehe C. P-I-D-Regler Funktionsweise und D. Programmierung).

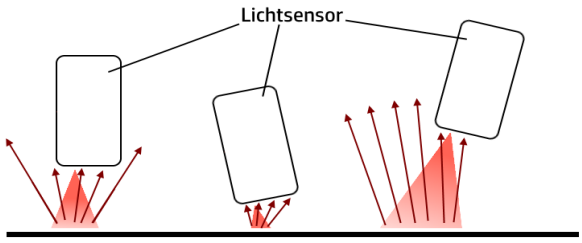


Abb. 1: vereinfachte Darstellung der Funktionsweise des Lichtsensors

B. *Aufbau*

Der Balance Bot besteht aus einem NXT Stein, auf dem das Programm läuft, einen Lichtsensor zur Erkennung des Abstands zum Boden, zwei Reifen, welche über eine durchgehende Achse miteinander verbunden sind und zwei Elektromotoren.
Anfänglich wurden kleine Reifen verbaut, die auch nicht mit einer durchgehenden Achse verbunden waren. Sehr schnell konnte man sehen, dass das Moment der kleinen Reifen nicht ausreichte, um die Lage des Roboters gut genug zu korrigieren. Als Lösung wurden größere Reifen verbaut, wodurch das Moment größer wurde. Auch die zwei einzelnen Achsen für jedes Rad waren ein Problem, weil die Elektromotoren von Lego ein gewisses Spiel aufwiesen und somit die Räder unterschiedlich gedreht wurden. Eine Achse für beide Räder behob diesen Fehler und der Balance Bot fuhr geradlinig.
Der NXT Stein wurde bei dem ersten Prototyp senkrecht eingebaut, doch diese Konstruktion wurde schnell wieder verworfen, da der Schwerpunkt nicht weit genug am Boden lag. Deswegen sitzt er nun waagerecht, knapp über der Achse der Reifen, wodurch ein tieferer Schwerpunkt erlangt werden konnte und der Balance Bot schon ohne Regler gut ausbalanciert ist.
Lego bietet zwei verschiedene Sensoren zur Helligkeitserkennung an. Es gibt einmal den Farbsensor und einmal den Lichtsensor. Ursprünglich sollte ein Farbsensor verbaut werden, doch nach einigen Test der beiden Sensoren konnte man erkennen, dass die geplotteten Graphen in Matlab, der jeweiligen Sensoren, sehr voneinander abweichen. Beim Farbsensor ist der Graph nicht so präzise gewesen wie beim Lichtsensor. Dass bedeutet der Lichtsensor gibt genauere Helligkeitsveränderungen an. Diese Veränderungen sind wichtig für den P-I-D-Regler, weil das die zu verarbeitenden Fehler sind. Ungenaue Helligkeitswerte würden zu ungenauen Fehlerkorrekturen führen und das würde den Roboter zum Umkippen bringen.
Obwohl der verbaute Lichtsensor sehr genau ist, ist die Differenz zwischen positiven und negativen Fehlerwerten anfangs zu gering gewesen. Damit es zu stärkeren Helligkeitsveränderungen kommen kann, muss der Lichtsensor mehr bzw. weniger Licht registrieren können. Das bedeutet es muss ein größerer Winkel gegeben sein. Da der Lichtsensor als erstes fast direkt an der Achse befestigt war, ist der Winkel bei Kippbewegungen sehr klein gewesen. Um dieses Problem zu lösen, wurde eine Verlängerung nach vorne gebaut, an der der Lichtsensor jetzt befestigt ist. Der Winkel bei Kippbewegungen ist nun größer und die Helligkeitsveränderung auch (fertiger Balance Bot im Anhang Abbildung 5 zu sehen).

C. *P-I-D-Regler Funktionsweise*

Der P-I-D-Regler ist das Herzstück des Balance Bots, denn nur dadurch kann er ausbalanciert werden. Zu Beginn ist es wichtig gewesen die Formel für den zu korrigierenden Fehler anzugeben. Dieser wurde folgenderweise definiert:

Fehler=aktueller Lichtwert-Nullpunkt

Mit dieser Formel konnten dann die einzelnen Elemente, P, I und D erstellt werden. Dafür ergaben sich folgende Formeln:

P=X × Fehler
I=Y × Integral → Integral=Integral + Fehler
D=Z × Differenz → Differenz=Fehler – vorh. Fehler

Für die Variablen X, Y und Z mussten Werte eingesetzt werden, welche durch ausprobieren ermittelt wurden. Als Hilfe dafür wurde ein Graph nach jedem Testdurchlauf in Matlab geplottet an dem man dann erkennen konnte ob die Variablen zu klein, groß oder richtig waren. In der folgenden Abbildung ist ein Testdurchlauf zu sehen. Dieser wurde in zwei Teilen aufgebteilt, um besser auf spezielle Fälle eingehen zu können.

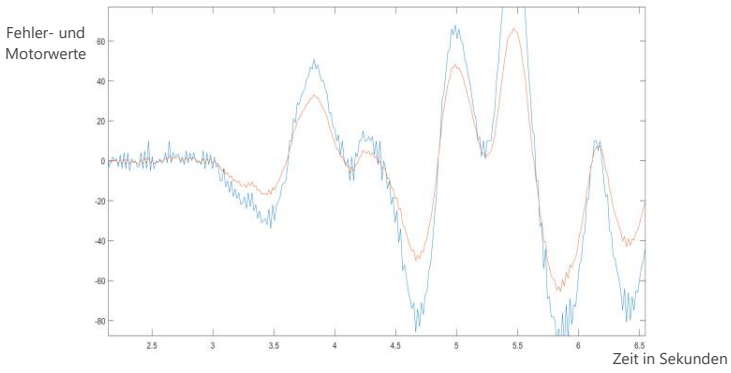


Abb. 2: Testdurchlauf Teil 1

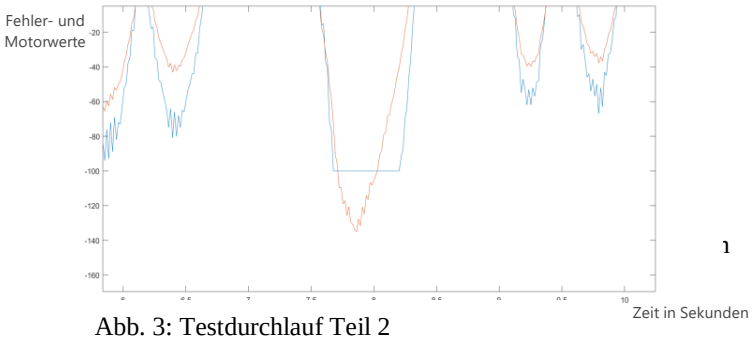


Abb. 3: Testdurchlauf Teil 2

In den beiden Abbildungen 2 und 3 gibt es zwei Graphen. Der orangene Graph zeigt den Fehler (Roboter kippt nach vorne (positiv) bzw. nach hinten (negativ)) und der blaue die Motorwerte, wie stark die Motoren arbeiten müssen, um den Fehler zu korrigieren. Abbildung 2 ist ein gutes Beispiel dafür, wie ein fast perfekter Durchlauf aussehen sollte. Die blaue Kurve ist immer größer als die Orangene. Doch nach einer Zeit während des Durchlaufs, ist die orangene Kurve größer als die Blaue. Dies ist in Abbildung 3 zu sehen. In solch einem Fall kippt der Balance Bot um, weil der Motor nicht mehr in der Lage ist den Fehler zu korrigieren.
Die Motoren von Lego haben eine maximale Leistung von 100 rpm doch überschreitet der Fehler die 100 rpm, indem der Roboter sehr weit in eine Richtung kippt, so können die Motoren nicht mehr gegensteuern.
Dieses Problem trat sehr häufig auf. Die Ursache dafür war, dass die Zeitspanne zwischen Matlabbefehle und Motoren zu groß war. Der Roboter befand sich schon im freien Fall nach vorne und erst dann versuchten die Motoren gegen zu steuern. Dabei hat der Fehler schon die 100 überschritten und die Motoren konnten nichts mehr machen, sodass der Balance Bot zu Boden fiel. Als Fazit konnte man ziehen, dass die Latenz zwischen Matlab und NXT 2.0 zu groß ist. Selbst mit direkter Verbindung mittels USB Kabel war keine Verbesserung zu sehen. Somit war ein Plan B gefordert welcher im nachfoldendem Teil D. Programmierung gezeigt wird.

D. Programmierung

Als Programmiersprache für das Programm des Balance Bots wurde anfangs Matlab benutzt. Nachdem es zu Problemen aufgrund einer zu hohen Latenz kam, gab es nur als einzige Lösung, das Programm direkt auf dem NXT 2.0 abspielen zu lassen. Nach reichlicher Recherche im Internet, gab es keine Möglichkeit, den Matlabcode auf dem NXT 2.0 zu übertragen. Deswegen wurde der Matlabcode in die Lego Mindstorms NXT 2.0 Software umgewandelt und es war nun möglich das Programm direkt auf dem NXT 2.0 laufen zu lassen. Das Problem mit der Latenz ist nun nicht mehr Relevant.

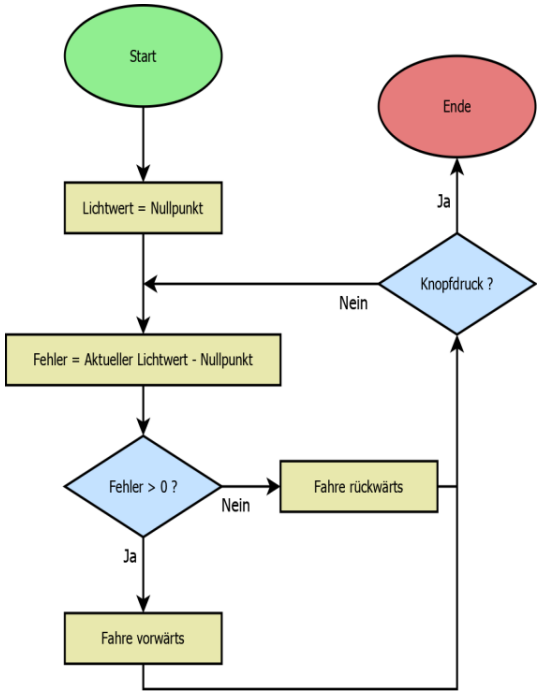


Abb. 4: Programmablaufplan des Balance Bot Programms

Der oben gezeigte Programmablaufplan gilt sowohl für den Matlabcode als auch für den in der Lego Mindstorms NXT 2.0 Software. Es beginnt damit, dass wenn der Blance Bot gestartet wir, der Lichtsensor den in der Position gemessenen Lichtwert als Nullpunkt nimmt. Damit wird dann der Fehler berechnet und es kommt zu der Abfrage, ob dieser größer als Null ist. Ist dies der Fall fährt der Roboter nach vorne. Im anderen Fall fährt er nach hinten. Am Ende kann der Balance Bot durch einen Knopf ausgeschaltet werden. Jedoch ist hierbei zu beachten, dass der Roboter zu sichern ist, weil er im ausgeschalteten Zustand nicht in der Lage ist auf Rädern zu stehen.

IV. Ergebnisdiskussion

Am Ende des Projektes ist der Balance Bot trotz anfänglich großen Problemen fertig und funktionsfähig geworden. Die Idee vom Anfang wurde umgesetzt. Ein Roboter, welcher auf zwei Rädern fahren kann und das ohne Gyrosensor sondern mit Lichtsensor, was das Ganze auch so besonders macht.

Die Kippbewegung nach vorne bzw, nach hinten darf nicht weiter als 30 Grad sein, da ansonsten die Motoren nicht genug Leistung zur Korrektur aufbringen können. Dennoch geschieht dies in der Regel nicht, weil die Motoren die Neigungen schnell genug korrigieren, weshalb nicht mehr als 30 Grad Neigung erreicht werden.
Mit Hilfe des geplotteten Graphen in Matlab ist es möglich gewesen, die Reglerwerte zu optimieren und somit flüssige Korrekturbewegungen zu erreichen.
Durch die Stützvorrichtung war es möglich den Blance Bot jedes Mal aus derselben Lage heraus zu starten, auf der die Reglerwerte angepasst sind.
Das umwandeln des Codes von Matlab in Lego Mindstorms NXT 2.0 lief reibungslos und brachte die erhoffte Verbesserung.
Letztendlich konnte das Hauptproblem „zu hohe Latenz“ behoben werden.

V. Zusammenfassung und Fazit

Der Balance Bot ist in der Theorie eine wichtige Ergänzung in Lagerhallen, weil er die zu erledigende Arbeit (Waren einlagern, Inventuren) erleichtert und schnell ausführt. Der Einsatz von Robotern ist nämlich in diesem Bereich nicht so stark vorhanden, wie in anderen Branchen zum Beispiel die Automobilindustrie.
In der Praxis würde sich der Balance Bot eher schlecht anstellen, denn er müsste weitaus größer und Leistungsfähiger sein. Außerdem wäre die Verwendung eines Gyroskops besser, denn der Lichtsensor ist neben seine Lichtanfälligkeit auch anfällig gegenüber unebenen und verschieden farbigen Böden. Das Licht wird anders reflektiert und es kommt zu ganz anderen Lichtwerten. Somit ist das ganze System wieder unstabil.
Trotz allem war das Ziel, den Roboter ohne Gyroskop zu bauen und das wurde erreicht und funktioniert mit wenigen Ausnahmen fehlerfrei. Das Zeigt, dass schon bestehende technische Innovationen auch auf eine andere Art und Weise umgesetzt werden können.
Als Erweiterung des Roboters könnte er am Kopf eine Kamera eingesetzt bekommen, womit der Barcodes einlesen kann oder patrouilliert und dabei Videoaufnahmen macht. Auch der dritte Motor, welcher den Oberkörper des Balance Bots bildet, könnte zum Bewegen gebracht werden. Dadurch könnte er dann ohne angeschubst zu werden selbstständig vor- und rückwärts fahren, wie ein Mensch auf einem Segway.

Anhang

Abb. 5: fertiger Balance Bot

Literaturverzeichnis

[1] Anja: Segwayfunktionsweise. http://autogeco.de/wie-funktioniert-segway-technologie/. Version: 17.04.2015

[2] Wikipedia, The free encyclopedia: P-I-D-Regler. https://de.wikipedia.org/wiki/Regler. Version: 29.01.2019

Balance Bot - Ein Roboter auf zwei Rädern

Carlo Schafflik, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Das folgende Dokument befasst sich mit dem Prinzip eines Roboters, der sich auf zwei nebeneinander angebrachten Rädern mit Hilfe eines Lichtsensors aufrecht halten kann. Das lässt sich durch ständiges Ausgleichen der vertikalen Neigung, gesteuert durch einen PID-Controller [1], des Roboters realisieren. Heutige mobile Roboter, besonders jene, die einen hohen Masseschwerpunkt haben, brauchen meist eine große Auflagefläche und sind darum langsam und relativ platzintensiv. Zur Lösung dieses Problems wurde ein Roboter im Rahmen des Lego-Mindstorms-Projektseminars gebaut, der sich mit einem Lichtsensor und zwei Rädern aufrecht halten kann.**

***Schlagwörter*—Balancieren, Lego, MatLab, PID-Controller**

I. Einleitung

In der heutigen Zeit werden Roboter immer wichtiger, da diese schwere Arbeiten von Menschen abnehmen, die Produktion von Massenartikeln vereinfachen oder sogar auf anderen Planeten die Oberfläche erforschen. Aber es gibt nur wenige Roboter die in einer Lagerhalle aktiv arbeiten. Wie zum Beispiel Dronen, die eine Inventur durchführen können, diese sind aber laut, teuer und können keine schweren Gegenstände transportieren. Der Einsatz von zweirädrigen Robotern könnte das lösen, denn diese sind sehr flexibel, platzsparend, stabil und sehr wendig.

Im Projektseminar Lego Mindstorms, dass im Zeitraum vom 11. Februar bis 22. Februar 2019 bearbeitet wurde, war es das Ziel, solch ein Roboter zu bauen. Das klare Hauptziel war es, einen Roboter auf zwei Rädern zu bauen, der sich mit einem Lichtsensor ohne äußerliche Hilfe, wie zum Beispiel Stützräder, aufrecht halten kann. Zusätzlich sollte er noch selbstständig vorwärts und rückwärts fahren und einen Gegenstand transportieren, was zeitlich nicht möglich war.

II. Vorbetrachtungen

Heute gibt es schon Roboter, die auf zwei Räder fahren können, wie zum Beispiel ein Segway [2], der dazu dient, einen Menschen zu transportieren. Es wurde auch schon ein Roboter [3] entwickelt, der Lasten heben, springen und sich schnell fortbewegen kann. Damit diese Roboter aufrecht stehen, muss der Neigungswinkel ständig kontrolliert werden. Der Neigungswinkel wird mit einem Gyrosensor abgelesen. Da es im Projekt nicht möglich war, einen Gyrosensor zu verwenden, musste eine andere Methode zur Überprüfung des Neigungswinkels entwickelt werden.

III. Umsetzung

A. Aufbau

Der erste Ansatz war es, einen Ultraschallsensor zu verwenden, der die Neigung durch den Abstand zum Boden misst. Da dieser aber sehr ungenau arbeitet und nur glatte Zentimeterwerte ausgibt war klar, dass eine andere Lösung entwickelt werden musste. Der bessere Ansatz war dann ein Farbsensor, der ein Stück vor der Achse angebracht wurde. Dieser Sensor besitzt aber ein sehr starkes Rauschen der Lichtwerte. Also war der finale Ansatz, den Lichtsensor zu verwenden, der auch sehr gut funktioniert. So standen dann die Teile fest, die gebraucht wurden: Zwei Motoren, an denen je ein Rad befestigt wurde, einen Lichtsensor der den Platz des Farbsensors eingenommen hatte und natürlich der NXT Brick. Der endgültige Aufbau ist in Abbildung 1 zu sehen. Es ist auch noch ein dritter Motor oben am Körper angebracht, der dafür zuständig gewesen wäre, dass der Roboter durch Gewichtsverlagerung nach vorne oder hinten fahren kann, was aber zeitlich nicht umgesetzt werden konnte. Außerdem wurden zuerst kleine Räder verwendet, die dann durch größere Räder ausgetauscht wurden, da dadurch eine schnellere Konterbewegung ausgeführt werden konnte.

B. Programmierung

Der wichtigste und auch der schwierigste Teil an diesem Roboter ist die Programmierung, denn hier wird die ganze Steuerung übernommen. Aber um das Programm zu verstehen, muss erst geklärt werden, wie der Lichtsensor die Neigung des Roboters ausgibt. In Abbildung 2 kann man sehen, wie sich die Position des Lichtsensors ändert. In der ersten Position (links) ist der Lichtsensor in der Idealstellung, das heißt, wenn der Roboter ausbalanciert ist und gerade steht. In der zweiten Position (mitte) kippt der Roboter nach vorne, wodurch der Lichtsensor näher zum Untergrund kommt und dadurch mehr Licht zum Lichtsensor zurück reflektiert wird. In der letzten Position (rechts) kippt der Roboter nach hinten und dadurch wird weniger Licht zurück reflektiert. Durch diese Änderung der Lichtintensität konnte man die Werte benutzen, um die Motoren anzusteuern. Um einen groben Einstieg in das Programm zu bekommen, ist in Abbildung 3 der Programmablaufplan skizziert. Wenn das Programm startet, wird zuerst der Nullpunkt festgelegt, also der Punkt, an dem der Roboter gerade steht und ausbalanciert ist. Der entsprechende Lichtwert wird in einer Variable gespeichert. Als Nächstes wird der Fehlerwert bestimmt, indem der aktuelle Lichtwert vom Wert des Nullpunktes subtrahiert wird. Durch diese Differenz kann man dann die Neigungsintensität sowie die Neigungsrichtung ablesen und für die Motoren weiterverarbeiten. Grob kann man sagen, dass wenn der Fehler positiv ist, der Roboter nach vorne fahren soll und wenn der Fehler negativ ist, rückwärts fahren soll. Als einfache Abbruchbedingung ist der Knopf am NXT Brick gut, um das Programm einfach zu beenden. Nachdem diese Grundlage vorhanden war, ging es dann an die Programmierung in MatLab. Um die Motoren korrekt

DOI: 10.24352/UB.OVGU-2019-057

ansteuern zu können, musste das mit einem PID-Controller geschehen, der aktuelle, vergangene und zukünftige Fehler ausgleicht. Der PID-Controller besteht aus 3 Komponenten: Einmal der proportionale Anteil, der die aktuellen Fehler berichtigt. Dieser lässt sich mit der Gleichung (1) berechnen. Dann gibt es noch den Integralanteil, der die vergangenden Fehler ausgleicht. Hier kann die Gleichung (2) verwendet werden. Zuletzt braucht man noch den Differentialanteil, der für zukünftige Fehler verantwortlich ist. Man verwendet hier Gleichung (3). „F“ ist hier der aktuelle Fehler. Die „I“ Variable ist dabei die Summe aller Fehler und die „D“ Variable die Differenz zwischen dem letzten und dem aktuellen Fehler. Dann gibt es noch die 3 Variablen x, y und z, die so angepasst werden mussten, dass eine gute Performance des Roboters entstand. Die endgültige Motorleistung errechnet sich dann aus diesen drei Komponenten, indem diese zusammen addiert werden. Um diese Stellschrauben besser anpassen zu können, wurde der Fehlerwert (orange) und die berechnete Motorleistung (blau) in MatLab geplottet, was man in Abbildung 4 sehen kann. Nach einigen Testreihen wurde festgestellt, dass die Kommunikation zwischen MatLab und dem NXT Brick, selbst mit USB-Kabel, viel zu langsam ist, was dazu führte, dass der Roboter sich nicht länger als ein paar Sekunden aufrecht halten kann. Deshalb erforderte es einen Weg den Code direkt auf den NXT Brick zu übertragen. Aus diesem Grund musste es mit der offiziellen Software ”LEGO MINDSTORMS NXT 2.0- [4] von Lego weiter gehen. Dann wurde der Quelltext aus MatLab umgeschrieben, sodass der Code direkt auf dem NXT Brick kompiliert und geladen werden konnte. Nach weiteren Anpassungen der 3 Reglerparameter in der neuen Software konnte ein sehr gutes Ergebnis erzielt werden, sodass der Roboter sehr stabil steht und ausbalanciert bleibt.

Abbildung 1. Balance Bot aus dem Lego-Projektseminar 2019

C. Gleichungen

$$p = x \cdot F \tag{1}$$

$$i = y \cdot I \tag{2}$$

$$d = z \cdot D \tag{3}$$

IV. Ergebnisdiskussion

Das Projekt war ein großer Erfolg. Es wurde in 2 Wochen einen voll funktionsfähigen Balancierroboter gebaut und programmiert, welcher unter verschiedenen Untergründen ohne Probleme stehen konnte. Es sind dabei auf viele Probleme enstanden, die aber nicht das Projekt beendet haben. Für jedes Problem wurde eine Lösung entwickelt und umgesetzt.

Die größte Herausforderung war es, den Code und den PID-Controller zu schreiben und anzupassen. Das war anfänglich in MatLab noch schwieriger, da hier mit einer großen Verzögerung gearbeitet werden musste, was auch bis zur Lego-Software eines der größten Probleme war. Ein weiteres Problem war die anfängliche Schwierigkeit mit dem Farbsensor, der ein zu starkes Rauschen hatte. Ein bestehendes Problem ist, dass der Lichtsensor auch auf das Umgebungslicht reagiert, was

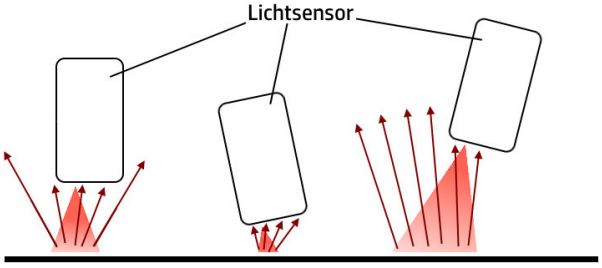


Abbildung 2. Vereinfachte Darstellung der Funktionsweise des Lichtsensors

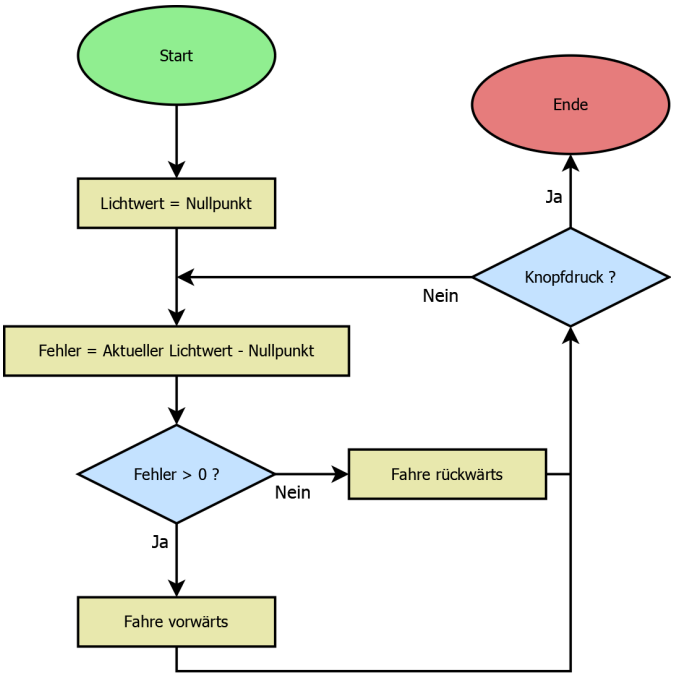


Abbildung 3. Vereinfachter Programmablaufplan

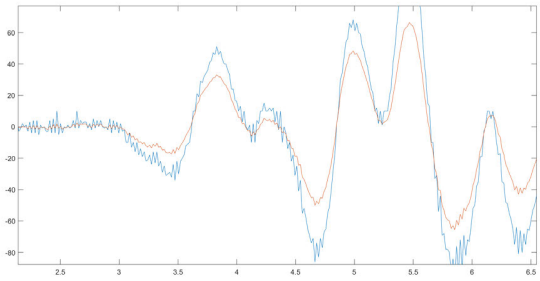


Abbildung 4. Graph der Fehler- und Motorwerte

also dazu führt, dass der Roboter nach vorne fährt, wenn auf einmal sehr viel Licht einstrahlt. Ein zusätzliches Problem bei der Programmierung in MatLab, welches durch die hohe Verzögerung aufgetreten ist, dass die errechnete Motorleistung über einen Wert von 100 oder unter -100 erreicht wurde. Abbildung 5 stellt dieses Problem grafisch dar. Die Motoren können nur den Maximalwert 100 oder -100 haben, was dazu führt, dass der PID-Controller nicht mehr gegen die Fehlerwerte steuern kann.

V. Zusammenfassung und Fazit

In dem LEGO-Mindstorms-Projektseminar wurde erfolgreich ein Roboter entwickelt, der selbstständig auf zwei Räder steht und sich auch bei äußeren Krafteinwirkungen ausbalancieren kann. Der Roboter konnte durch verschiedene Lösungsansätze verbessert werden. Es fehlt allerdings noch die geplante Steuerung für das Vor- und Zurückfahren, sowie ein automatisierten Bewegungsablauf. Außerdem kann man den Lichtsensor benutzen, damit der Roboter beispielsweise eine schwarze Linie

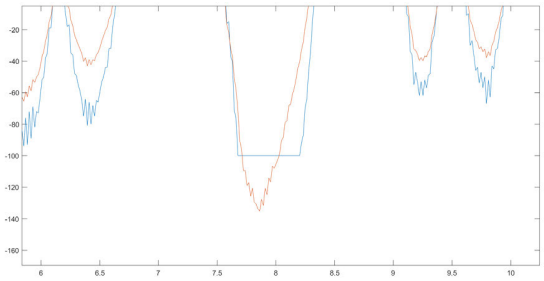


Abbildung 5. Graph der Fehler- und Motorwerte mit abgeschnittenden Werten

folgt und anstelle diesen dafür einen Gyrosensor verwenden, wodurch ein Fahren auf allen Untergründen möglich wäre.

Anhang

Hier noch ein kurzer Ausschnitt aus dem MatLab-Code mit dem PID-Controller:

```
StdLight = GetLight(SENSOR3);

while true
Error = GetLight(SENSOR3) - StdLight;

Integral = Integral + Error;
Derivative = Error - PrevError;

PrevPrevError = PrevError;

PrevError = Error;
(Leistung des Motors wird addiert)
PM = round(Kp*Error + Ki*Integral + Kd*Derivative);
end
```

Literaturverzeichnis

[1] Wikipedia, the free encyclopedia: *PID controller*, 2019. https://en.wikipedia.org/wiki/PID_controller
[2] Wikipedia, the free encyclopedia: *Segway*, 2019. https://en.wikipedia.org/wiki/Segway
[3] BostonDynamics: *Handle, Legs & Wheels: The Best of Both Worlds*. Waltham, Massachusetts: BostonDynamics, 2017. https://www.bostondynamics.com/handle
[4] LEGO: *Lego Mindstorm Software*, https://www.lego.com/de-de/mindstorms/downloads

Raumscanner

Valeriia Kantur, ETIT
Otto-von-Guericke-Universität Magdeburg

Heutzutage wird 3D-Modelierung in verschiedenen Tätigkeitsbereichen genutzt. Besonders interessant ist die Übertragung von Objekten und Fläche aus der realen Umgebung in den Computer für deren weiteren Analyse und Weiterverarbeitung. Um das zu realisieren, wird ein Gerät benötigt, welches in der Lage ist die Umgebung auszumessen und mit Hilfe entsprechender Software zu digitalisieren. In dieser Arbeit wird es also um die Hardware und Software Umsetzung eines 3D-Raumscanners gehen.

I. Einleitung

Fortschritt steht nie still und auch bewährte Methoden und Technologien werden zahlreichen Verbesserungen unterzogen. Eine von solchen Verbesserungen ist die 3D-Modellierung. Die Vielfalt der Einsatzbereiche ist so groß, dass es kaum möglich ist, alle aufzuzählen. Als Beispiel können folgende Bereiche genannt werden:

- Medizin: Operationen, Untersuchungen, Konstruktion individueller Prothesen;
- Restaurierung: Wiederherstellung kompletter historischer Objekte, basierend auf Teilfunden;
- Maschinenbau: Analyse und Weiterentwicklung von Bauteilen.

Der spezielle Anwendungsbereich dieser Arbeit ist Wohnungsrenovierung und Innenraumdesign. Wenn es um das Verändern bereits existierender Objekt geht, müssen die entsprechenden Abmessungen bekannt sein. Das Ausmessen der Räume soll mit einem 3D-Scanner geschehen. Mit Hilfe von Schritt-für-Schritt-Messungen und einer speziell entwickelten Software wird ein Computermodell des Raumes erstellt. Dieses Modell vereinfacht den gesamten weiteren Planungsprozess. Projekte, bei denen 3D-Modelierung benutzt wird, können viel schneller ausgeführt werden, weil Veränderungen im Modell sofort auch von Laien erkannt und verstanden werden können. Ein Kunde kann so schneller entscheiden, wie die Endausführung aussehen soll.

II. Vorbetrachtungen

Die Lösung dieser Aufgabe erfordert das Bestimmen eines passenden Scannertyps. Laut [1] teilen sich 3D-Scanner nach ihrer Arbeitsweise in zwei Typen: Kontaktscanner und berührungslose Scanner. Die Ersteren brauchen direkten Kontakt mit dem zu untersuchenden Objekt. Die Alternative dazu sind die berührungslosen Scanner. Sie können wiederum aktiv und passiv ausgeführt sein. Aktive Scanner strahlen Wellen aus und messen deren Wiederspiegelung für die weitere Analyse. Meist wird Licht einer LED (engl. **L**ight **E**mitting **D**iode) oder eines Laserstrahls benutzt, seltener Röntgenstrahlung, Infrarotstrahlung oder Ultraschall. Passive Scanner arbeiten mit Hilfe reflektierter Umgebungsstrahlung, wie zum Beispiel sichtbarem Licht.

In diesem Fall ist ein aktiver Scanner die beste Wahl.

Beispiel: Scannen mit Leica 3D Disto

Um den Raum entsprechend [2] und [3] auszumessen, wird ein Scanner mit optischem Abstandssensor verwendet. Laut [4] geschieht die Messung mittels Erfassung der Laufzeit eines Laserstrahls. Dieser wird von Leica 3D Disto ausgestrahlt und vom Messobjekt reflektiert. Durch die Laufzeit des Lichts kann der Abstand berechnet und danach kann die Raumabbildung dargestellt werden.

Ein eingebautes Goniometer ermöglicht eine Winkelbestimmung. Alle Abmessungen werden von einem Punkt ausgeführt. Eine Hand-Held-Unit erlaubt die berührungs- und kabellose Steuerung des Scanners. Sie dient gleichzeitig als Echtzeitdisplay von Messungen in der Form von Zeichnungen. Die Datenformate für Listen, Fotos, Tabellen und Zeichnungen sind standardisiert. Deshalb können die Ergebnisse problemloss importiert, betrachtet und weiterverarbeitet werden.

Beispiel: Scannen mit Canvas

Canvas [5] ist eine Applikation, die die Zusammenarbeit zwischen einem Structure Sensor [6] und dem iPad ermöglicht. Der Structure Sensor ist ein Tiefensensor. Der besteht gemäß [7] aus Infrarot (IR) -Projektor und IR-Kamera. Der IR-Projektor projiziert ein Muster aus IR-Licht, das wie Menge aus Punkten auf alle vom Projektor aus sichtbaren Objekte fällt. Durch die Nutzung von IR Licht, sind sie für die menschlichen Augen unsichtbar. Sie werden von der IR-Kamera erfasst und das entstehende Video wird an den Prozessor des Tiefensensors gesendet. Die Tiefe wird dann anhand der Verschiebung der Punkte ermittelt.

Mit dem Structure Sensor ist es möglich präzise 3D-Scans in Sekunden zu machen. Er ist leicht, kompakt, und für Tablets geeignet. Die weitere Datenverarbeitung ist mit der Canvas-App schnell zu realisieren.

III. Aufbau und Funktionsprinzip

Die Aufgabe des Projekts ist es, einen Raumscanner zu entwickeln und dessen Ergebnis in einem 3D-Modell

DOI: 10.24352/UB.OVGU-2019-059

darzustellen. Dabei sollen folgende Bedingungen erfüllt werden:

- Es soll ein passender Abstandssensor gewählt werden;
- Um den ganzen Raum abmessen zu können, soll sich der Scanner um seine Achse drehen (360° XY-Ebene);
- Die Höhe des Raums soll gemessen werden (Z-Richtung);
- Um für den Benutzer die Arbeit zu erleichtern wird ein separates Befehlsfenster in der App benötigt;
- Anzeige der Messergebnisse unmittelbar nach dem Scannen.

Der Algorithmus des Abmessens unter oben stehenden Bedingungen ist in Abbildung 1 zu sehen.

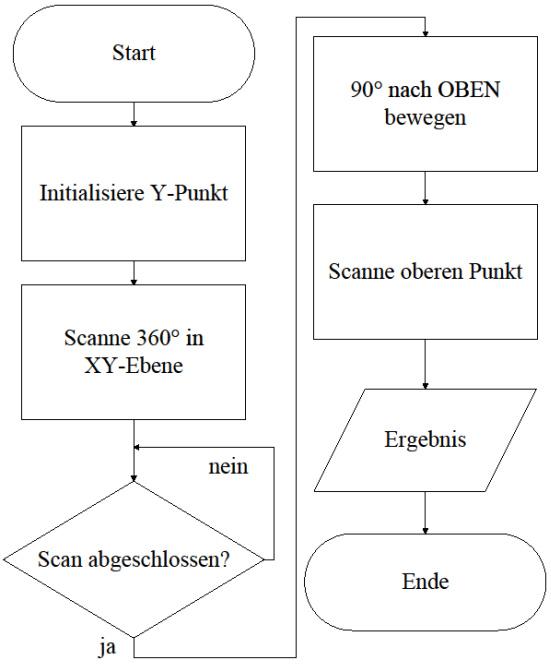


Abbildung 1:Programmablaufplan des Raumscanners

In der Abbildung 1 ist zu sehen, dass die Datenerfassung durch den Ultraschallsensor während einer 360° Umdrehung passiert. Nur wenn diese abgeschlossen ist, wird die Höhe gemessen.

Hierbei werden viel mehr Daten erfasst, als für die eigentliche Modellierung notwendig sind. In der Weiterverarbeitung werden die wichtigen Punkte aus dieser Datenmenge herausgefiltert.

Dies geschieht basierend darauf, dass der kleinste Abstand zu einer Wand dann erreicht ist, wenn der Scanner bzw. der Sensor senkrecht zur Wand zeigt. Von diesem Punkt ausgehend werden die Kontrollpunkte um 90° zu einander verschoben. Das Konzept der Suche nach diesen Kontrollpunkten ist folgendes: unabhängig davon, in welchem Teil des Raumes der Scanner steht, werden nach der Initialisierung der Y-Position vier 90°- Teildrehungen durchgeführt. Nach jeder Umdrehung werden die kleinsten Abstände ermittelt. Auf diese Weise können die vier Kontrollpunkte gefunden werden (Anhang 1).

Das Ermitteln von der Höhe wird durch die Drehung des Sensors um 90° nach oben realisiert.

Die Abbildung 2 stellt dar, aus welchen Komponenten der Raumscanner gebaut werden soll.

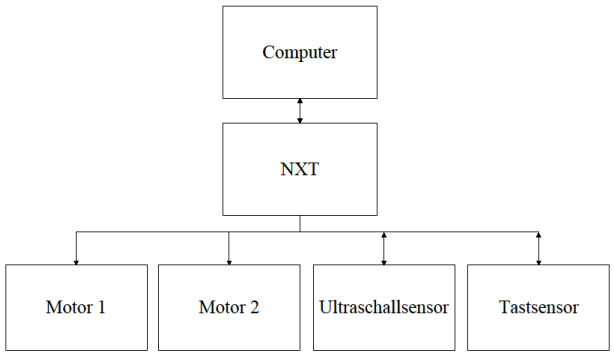


Abbildung 2: Blockschaltbild des Raumscanners

Folgende Bestandteile:

- NXT: mit seiner Hilfe werden die Informationen von Computer, Motoren und Sensoren zentralisiert. Diese Aufgabe übernimmt ein Mikroprozessor;
- Computer: Steuerung des Raumscanners;
- Motor 1, Motor 2: zur Drehung des Ultraschallsensors;
- Tastsensor: zur Ermittlung der Sensorposition;
- Ultraschallsensor: Aktiver Abstandssensor. Durch das Absenden von Schallwellen und das Empfängen der Echos, werden Objekte erkannt und deren Entfernung in cm errechnet [8].

Mit Rücksicht auf die technischen Möglichkeiten von LEGO MINDSTORMS wurde der in Abbildung 3 dargestellte Raumscanner entwickelt.

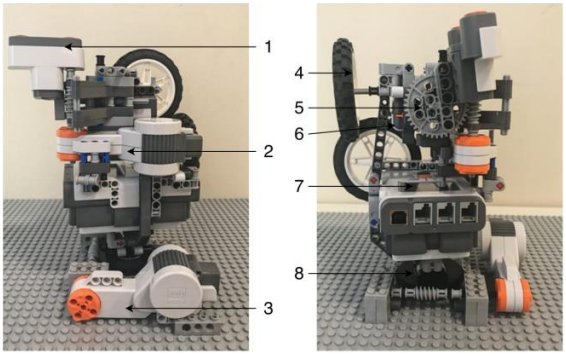


Abbildung 3: Reelles Aussehen des Raumscanners

1 – Ultraschallsensor
2 – Motor A (Bewegung in Z-Richtung)
3 – Motor B (Bewegung in XY-Ebene)
4 – Gegengewicht
5, 8 – Schneckengetriebe für präzisere Drehungen

6 – Tastsensor
7 – NXT

Die Steuerung des Geräts wird vom Computer durch Matlab realisiert. Erleichterung der Benutzung wurde ein GUI (**G**raphical-**U**ser-**I**nterface) Fenster eingefügt. Das ist in Abbildung 4 zu sehen.

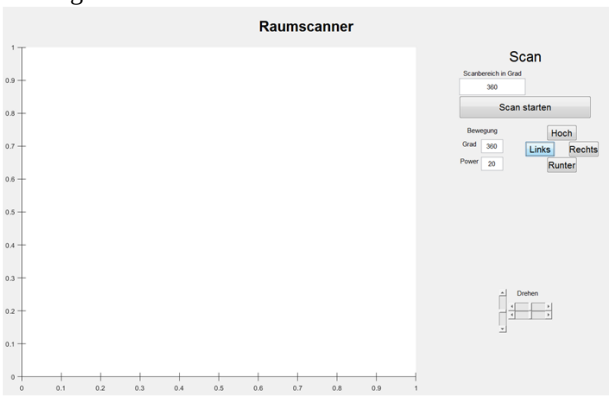


Abbildung 4: GUI des Raumscanners

Damit ist es möglich die Motoren direkt anzusteuern, bzw. die Bewegungsrichtung, die Leistung und den Drehwinkel einzugeben. Der Scanbereich ist ebenfalls einstellbar. Die GUI macht die Kommunikation zwischen Raumscanner und Benutzer eindeutig und schnell..

IV. Ergebnisdiskussion

Zum Testen der Funktion des Scanners wird ein leerer Raum benötigt. Es wurde eine leere Box benutzt. Das Ergebnis wird nach abgeschlossenem Scanprozess in einem neuen GUI-Fenster dargestellt, wie in der Abbildung 5 gezeigt.

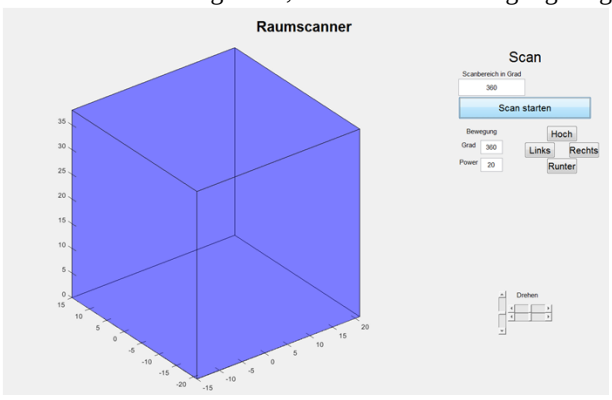


Abbildung 5: Raumabbildung im GUI-Fenster.

Durch die Drehung der Achsen kann man den gescannten Raum von alle Seiten ansehen.

Obwohl das Resultat der Messung korrekt ist, sollen folgende, bei der Entwicklung aufgetretene, Probleme auch beachtet werden:

- Die erhaltenen Abmessungen können wegen nicht so guter Standfestigkeit des Scanners während der Umdrehung ungenau sein;
- Die tatsächlichen Winkel bei der Drehung des Scanners sind nicht genau 90°.

V. Zusammenfassung und Fazit

Das Ziel des Projekts war einen Raumscanner zu entwickeln, der in der Lage ist, den ganzen Raum abzuscannen und eine digitalisierte Raumkopie zu erstellen. Das Ergebnis zeigt, dass das Ziel erfolgreich erreicht wurde.

Eine Weiterentwicklung kann in folgende Richtungen erfolgen:

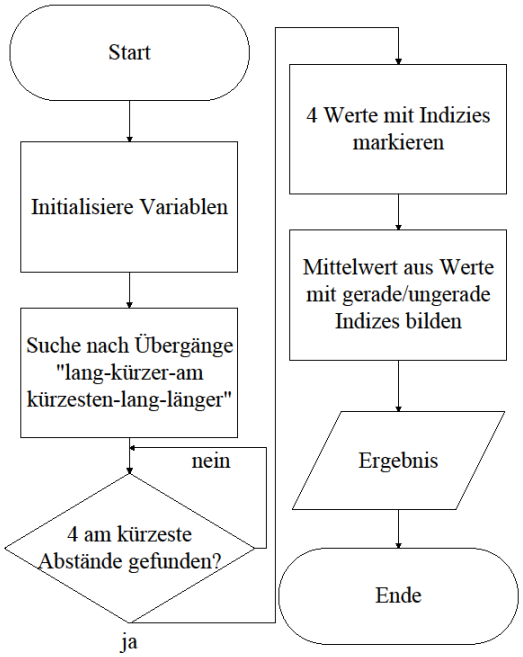
1. Vervollkommnung des Aufbaus des Raumscanners, um ihn stabiler zu machen;
2. Ergonomischere Gestaltung der Benutzeroberfläche;
3. Verbesserung des Programms um das Problem mit der Datenerfassung zu beseitigen.

Die Weiterentwicklung des Programms kann durch die Modifikation des Algorithmus zur Datenerfassung geschehen. Zum Beispiel können während des Scannens die Übergangszusammenhänge zwischen nach einander ermittelten Werten untersucht werden. Da die Stellung des Raumscanners beim kleinsten Abstand zur Wand bekannt ist, würden die Kontrollpunkte besser gefunden werden. Anstatt in 90° Abschnitten, könnten bei der kompletten 360° Umdrehung die Übergänge zwischen den Abständen «lang – kürzer – am kürzesten – länger – lang» verglichen werden. Die kürzesten Abstände entsprächen genau den gesuchten Werten. Nach der Markierung der Indizes der Werte und Bildung des Mittelwertes der im Raum gegenüberliegenden Abstände würde der Nullpunkt genau in der Mitte der Abbildung platziert werden. Dieses Resultat ist dann unabhängig von der realen Position des Scanners im Raum und vom Anfangswinkel des Scanvorgangs (Anhang 2).

ANHANG

```
s=1;
vektor=[];
for i=1:degrees
p = motorB.ReadFromNXT.Position;
while(p < 60)
p = motorB.ReadFromNXT.Position;
end
if i > 90
s=2;
end
if i > 180
s=3;
end
if i > 270
s=4;
end
vektor(i,s)= GetUltrasonic(SENSOR_1);
pause(0.001);
disp('SCHRITT: ' + i + '/'+
degrees+'ABSTAND: ' + vektor(i,s));
motorB.ResetPosition();
end
motorB.Stop();
vektor(vektor==0)=NaN;
[dim, ind] = min(vektor);
```

Anhang 1: Programmcode zur Ermittlung der vier Kontrollpunkte.



Anhang 2: Programmablaufplan der Weiterentwicklung des Programmcodes.

LITERATURVERZEICHNIS

[1] WIKIPEDIA, THE FREE ENCYCLOPEDIA: 3D-scanning (3D-Scannierung). https://en.wikipedia.org/wiki/3D_scanning, 2018

[2] VP Civil Surveying Instruments Pvt. Ltd.: Leica 3D Disto Laser Scanner. http://www.vpcivil.co.in/leica-3d-disto-laser-scanner/, 2011

[3] YouTube: Leica 3D Disto Room Scan Demo Video. https://www.youtube.com/watch?v=Kl5ykdAZjF8, 2011

[4] Sensorika: Opticheskie datchiki rasstoyaniya. (Sensorik: Optische Distanzsesoren) http:// www.sensorica.ru/s2-1.shtml

[5] YouTube: Canvas: Create A 3D Model Of Your Home In Minutes. https://www.youtube.com/watch?v=XA7FMoNAK 9M, 2016

[6] Structure: Scanning, Augmented reality, and more for mobile. https://structure.io/structure-sensor, 2016

[7] Andrew McWilliams. How a depth sensor works – in 5 minutes. https://jahya.net/blog/how-depth-sensor-works-in-5-minutes/, 2013

[8] WayCon Positionsmesstechnik: Ultraschallsensoren – Messprinzip. https://www.waycon.de/produkte/ultraschallsensoren/messprinzip-ultraschallsensoren/

Modell eines Patrouillenroboter

Justin Görke, Elektrotechnik/Informationstechnik

Otto-von-Guericke-Universität Magdeburg

Kurzfassung:

Der Schutz von wichtigen Dingen ist den meisten Menschen sehr wichtig und hochwertige Sicherheitssysteme sind sehr teuer, ebenso wie der Schutz durch Sicherheitspersonal. Diese Aufgabe könnte in Zukunft autonome Roboter übernehmen. Dies könnte Personal und auch Kosten sparen, ohne dabei die Sicherheit zu vernachlässigen. Im Beitrag wird das Lego Modell zum Patrouillenroboter erläutert und deren Funktionalität zum Bewachen von Objekten oder patrouillieren um dieses. Es wird dabei auf die Umsetzbarkeit eines solchen Sicherheitsroboters aus Lego beschrieben und seine Vor- und Nachteile dargestellt.

Einleitung:

In der heutigen Zeit kann man Roboter aus dem Leben gar nicht mehr wegdenken. Sie sind überall, in der Industrie beschleunigen sie viele Arbeitsschritte erheblich und sind dabei auch deutlich präziser, als der Mensch. Einfache Aufgaben wie Staubsaugen werden auch im Haushalt von Robotern übernommen, um den Menschen zu entlasten. So gibt es viele Einsatzmöglichkeiten, unter anderem auch das Überwachen von Gebieten oder auch Bewachen von Gegenständen. So könnte das Haus durch einen kleinen Roboter vor Einbrecher beschützt werden und so bei der Familie für Sicherheit sorgen. Dasselbe System könnte man auch auf größere Gebiete übertragen, um eine ganze Landschaft zu bewachen, wie es auch heutzutage bereits getan wird durch Patrouillenroboter. Der Einsatz von bessern und auch weitern Robotern könnte dem Menschen entlasten, sodass man wichtigeren Tätigkeiten nachgehen kann.

Stand der Technik:

In dem Projekt, das zusammen mit Fabian Hollburg bestritten wurde, wollten wir einen Roboter entwickeln, der es schafft, ein Objekt zu beschützen. Das Ziel war es eine Strecke vorzugeben, die der Roboter allein abfährt und immer wieder wie ein Radar seine Umgebung abtastet, um Bewegungen festzustellen. Das Ganze sollte mithilfe von Lego Mindstorms und den NXT Brick realisiert werden. Es sollten verschiedene Streckenoptionen zur Auswahl stehen, zum Beispiel ein Quadrat oder ein Rechteck. Die Strecke sollte ohne Bezugspunkte abgefahren werden so dass man den Roboter beliebig an anderen Orten einsetzen könnte, ohne vorher großartig eine Strecke festlegen zu müssen.

Es existieren bereits verschiedene Arten von Patrouillenroboter. Im Militär werden immer mehr unbemannte Fahrzeuge eigesetzt. Diese dienen meistens zur Aufklärung aber auch zur Grenzkontrolle und Überwachung. Meistens sind es Drohnen also Flugzeuge die ferngesteuert werden. Am Boden sind unbemannte Fahrzeuge noch eher die Seltenheit. Israel ist einer der wenigen Staaten, die auf diese Technik setzen. Sie nutzen seit einigen Jahren den sogenannten „Guardium“ (siehe Abbildung 2). Dieser wird als eine Mischung aus Kampfflugzeug und Geländewagen beschrieben.

Abbildung 1: „Guardium“ in Aktion

DOI: 10.24352/UB.OVGU-2019-061

Er soll autonom die Grenze von Israel zum Gaza Streifen patrouillieren. Es nutz für die Überwachung eine hochauflösende und eine Infrarot-Kamera, diese können um 360 Grad gedreht werden. Des Weiteren besitzt er ein Radar und sehr sensible Mikrofone. Beim Lego Modell wird eine Ultraschallsensor verwendet der sich um 360 Grad drehen kann, um so seine Umgebung zu beobachten. Desweitern wird auch ein Mikrofon verwendet, um Umgebungsgeräusche wahrzunehmen, wie beim „Guardium“ auch.
Es werden aber nicht nur im militärischen Bereich Sicherheitsroboter verwendet, auch zur Überwachung von Bürokomplexen oder Einkaufszentren werden im Silicon Valley bereits Roboter eingesetzt. Microsoft nutzt zum Beispielen K5 (siehe Abbildung 2).

Abbildung 2: Knightscope‘s Roboter K5

Dieser Roboter patrouilliert selbstständig an für ihn vorgesehenen Plätzen. Er beobachtet dabei das Geschehen um sich herum mit verschiedenen Kameras. Außerdem benutzt er einige Mikrofone, um auch Geräusche wahrnehmen zu können. Dabei achtet der K5 auf ungewöhnliches. Also zum Beispiel laute Schreie. Falls er etwas Feststellen sollte gibt er einen Alarm aus in Form von lauten Tönen und übermittelt ans nächste Sicherheitspersonal seine genaue Position, sodass schnell Hilfe zu ihm gelangen kann. Eine ähnliche Funktionsweise sollte auch unser Modell bekommen. Es soll wie der K5 auch Alarm schlagen, wenn es Veränderungen in der Nähe feststellt indem es laute Töne von sich gibt.

Hauptteil:

Die Idee bei dem Patrouillenroboter ist gewesen einen kleinen, wendigen und auch unauffälligen Roboter zu erschaffen. Er sollte die Möglichkeit haben seine Umgebung zu beobachten, diese speichern und dabei Veränderungen bemerken. Dazu verwendet der Roboter den Ultraschallsensor. Dieser war auf einem Stab angebracht und konnte sich um 360 Grad drehen. Durch die erhöhte Position des Sensors war es möglich seine komplette Umgebung zusehen und diese zu beobachten. Des Weiteren hat er ein Mikrofon, um Umgebungsgeräusche aufzunehmen. Er fuhr auf 5 Reifen, dabei waren an den Seiten jemals Zwillingsreifen angebracht. Die Zwillings Bereifung sorgte dabei für einen festeren Stand des Roboters. Jedes Reifenpaar wurde von einem separaten Motor angetrieben, um genauere Kontrolle beim Drehen und wenden zu haben.

Abbildung 3: Der Patrouillenroboter von vorne

Abbildung 4: Der Patrouillenroboter von hinten

Der letzte Reifen ist ein kleiner frei beweglicher Reifen, das heißt er konnte sich um 360 Grad drehen, ohne Antrieb. So ist es möglich den Roboter auf der Stelle drehen zu lassen. Dadurch konnte er gute 90 Grad und auch 180 Grad Drehungen vollziehen. Diese sind wichtig, wenn der Roboter als Streckenvorgabe zum Beispiel ein Quadrat abfahren soll. Dieses Fahrgestell wurde aber nicht von Anfang an verwendet. Es waren mehrere Umbauten nötig, um ihn so wendig zu machen wie er jetzt ist. In der ersten Version wurde ein Kettenantrieb verbaut. Das hatte als Vorteil das es sehr stabil war und leicht über kleine Hindernisse wie Kabel hinwegkam. Jedoch besaß diese Fahrzeug Nachteile. Es war durch die Ketten und die Bauweise sehr breit und nahm so viel Platz weg. Außerdem konnte es zu der Zeit keine genauen 90 Gard Wendungen vollführen. Es gab dabei immer kleine Abweichungen bis zu 5 Grad. Das lag auch zum Teil an der Ungenauigkeit der Lego Teile. Teilweise drehten die Räder durch auf denen die Ketten lagen oder wurden nicht richtig gedreht. So wurde der Entschluss gefasst auf Räder zu wechseln was sich als gute Entscheidung herausgestellt hat. Es sind dabei die größten Räder verwendet wurden, um nicht die Fähigkeit zu verlieren über kleine Hindernisse zu gelangen wie Kabel oder andere kleine Legosteine. Das meiste Gewicht lastete dabei auf den großen Rädern, wodurch das kleine Rad einfach über solche Hindernisse mitgezogen werden kann.

Die Möglichkeit 90 Grad Drehungen zu vollführen, ist für den Patrouillenroboter wichtig. Das Programm ist so geschrieben, dass man über die GUI in Matlab eine Route auswählen kann, auf der patrouilliert werden soll. Zur Auswahl stehen, ein Quadrat, eine Gerade und der Stand, wobei hier das Gefährt stehen bleibt und sich nur der Turm dreht. Also alle Messungen an einem Standort geschehen. Weitere Streckenoptionen sind geplant gewesen und auch in der GUI auswählbar, jedoch haben diese noch keine Funktion gezeigt. Diese sind ein Rechteck und ein Kreis. Man wählt aber nicht nur die Form der Route aus, sondern auch die Größe dieser Route also die Seitenlänge des Quadrats wird in Dezimeter abgefragt. Dazu kommt auch noch wie lange der Roboter diese Fahrten machen soll, also wie viele Wiederholungen gemacht werden bis es der er wieder zum Stillstand kommt. Wenn das Programm dann gestartet wird fährt der Patrouillenroboter die angegebene Strecke einmal ab und nimmt dabei Vergleichswerte auf, speichert diese ab und zeigt sie auch in der GUI in einem Diagramm an. Dafür hält er in jedem Eckpunkt an und dreht sein Turm um 360 Grad und nimmt dabei alle 36 Grad den Abstand zu seiner Umgebung auf. Dafür macht er immer wieder eine kurze Pause, um auch die Lautstärke zu messen. Er misst diese nur in der Pause da ansonsten die Motoren zu laut wären, um wirkliche Geräusche feststellen zu können. Wurden alle vier Punkte erreicht und alle Werte eingespeichert fährt der Roboter nun die Strecke ab und vergleicht an jedem Eckpunkt die Entfernungswerte mit den Vergleichswerten. Dies wird auch wieder im Diagramm veranschaulicht. Wenn er dabei Abweichungen über 10 Zentimeter feststellt oder falls das Mikrofon einen erhöhten Wert aufnimmt, schlägt er Alarm. Der Alarm ist dabei ähnlich einer Sirene, die nur über die GUI gestoppt werden kann.

Der Patrouillenroboter lässt sich leicht über die GUI steuern, da diese das ganze sehr veranschaulicht. Man kann leicht die verschiedenen Optionen auswählen und durch die Variation der Länge und Wiederholungen der Strecke kann man ihn leicht und schnell an verschiedenen Orten einsetzen. Die Sensoren sind dabei jedoch sehr ungenau. Der Ultraschallsensor hatte als maximal Reichweite 255 Zentimeter jedoch hat er bei einer Entfernung von über einen Meter keine genauen Werte zurückgeben können. So wie er eine schräge Fläche beschallt hat, bekam man immer als Entfernung den Maximalwert. Deshalb ist ein Ultraschallsensor zur Überwachung eher ungeeignet. Dass Mikrofon hat auch nur bedingt seinen Zweck erfüllt, da es immer nur zu einer kurzen Zeit aufgenommen hat. Wenn sich gerade kein Motor bewegt hat, wurde selten ein Geräusch

wahrgenommen bei dem er hätte Alarm schlagen sollen. Das Fahrzeug hatte auch Probleme bei langem Test das Quadrat zu halten da es teilweise bei 90 Grad Drehungen kleine Abweichungen von einem Grad gab. Das sorgte jedoch für immer größere Abweichungen von der eigentlichen Spur umso länger der Roboter lief. Das führte auch wieder dazu, dass teilweise Alarm geschlagen wurde obwohl es keinerlei Veränderungen in der Umgebung gab aber das Fahrzeug die Objekte um sich herum in einem anderen Winkel sah und so die Vergleichswerte nicht mehr übereinstimmten.

Zusammenfassung:

Der Patrouillenroboter ist ein gelungenes Lego Modell zu mehreren in der Realität bestehenden Vorbildern. Er zeigt viele Gemeinsamkeiten in der Funktionsweise mit seinen Verwandten. Dabei erreicht er eine gute Genauigkeit bei den Messungen und kann seine Hauptaufgabe, das beschützen von Objekten gut vollführen. Er ist für die Praxis dennoch ungeeignet denn nach längerem benutzen die Abweichungen immer stärker werden. Das Modell hat aber auch gute Seiten da kein ähnliches Produkt so klein und handlich ist wie der Patrouillenroboter. Wenn man noch etwas mehr Zeit in ihn investiert und einige kleine Verbesserungen vornimmt, wäre er für den Praxiseinsatz tauglich. Dafür müsse man nur die Ungenauigkeiten beim Drehen verbessern, sodass auch der längere Einsatz möglich wird. Man könnte auch andere Sensoren verwenden, denn der Ultraschallsensor von Lego hat bei größeren Entfernungen Probleme genau zu messen.

Abbildung 1:
„Guardium“ in Aktion
http://www.spiegel.de/fotostrecke/militaerroboter-guardium-bewacht-israels-grenze-fotostrecke-90097.html,
22.03.2019

Abbildung 2:
Knightscope‘s Roboter K5
https://www.hi-heute.de/sicherheitservices/sicherheit-und-services/news/roboterauf-patrouille-im-shopping-center/,
22.03.2019

Quellen:

Entwicklung eines Patrouillenroboters

Fabian Hollburg, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Abstract*— In diesem Paper wird die Entstehung eines Patrouillenroboters behandelt, welcher im Rahmen des Projektseminars Elektro- und Informationstechnik der Otto-von-Guericke-Universität Magdeburg entwickelt wurde. Der Roboter ist in der Lage eine festgelegte, veränderbare Route abzufahren und durch die Benutzung eines Ultraschall- sowie eines Geräuschsensors Bewegungen und Geräusche zu erkennen und auszuwerten. Weiterhin wird der Bezug zu modernen Meldeanlagen der Sicherheitstechnik hergestellt. Als Hardware wurde hierfür Lego Mindstorms und als Software MATLAB verwendet.**

***Schlagwörter*— Lego Mindstorms, MATLAB, Patrouille, Sicherheitstechnik, Ultraschall**

I. Einleitung

SICHERHEIT hat für den Menschen schon immer eine große Rolle gespielt, so wurden Mauern und Zäune gebaut, Hunde abgerichtet oder Wachleute ausgebildet, um sein Hab und Gut zu beschützen. Obwohl man sich heutzutage die moderne Technik zu Nutze macht, werden immer noch Menschen als Wachmänner eingesetzt. So kam die Frage auf, ob diese Aufgabe nicht durch eine Maschine übernommen werden kann. Mithilfe von Lego Mindstorms Bauteilen und der Software MATLAB sollte ein Roboter gebaut werden, der möglicherweise eine Antwort auf diese Frage liefert. Die Idee war, dass der Roboter auf Befehl einen Rundgang startet, bei dem ein vorgeschriebener, anpassbarer Weg abgefahren werden soll. Währendessen sollte mithilfe der Sensoren die Umwelt bewacht und bei einer erkannten Gefahr ein Alarm ausgelöst werden. Die Ähnlichkeit mit einer Patrouille lieferte den Namen für das Projekt.

II. Vorbetrachtungen

Wie schon erwähnt, gibt es verschiedene technische Geräte, die der Mensch für seine Sicherheit nutzt.

A. Überwachungskameras

Der Einsatz von Sicherheitskameras ist weit verbreitet. Die Kamera wird an einer leicht erhöhten Stelle angebracht und zeichnet durchgängig ein Video des erfassten Bildes auf. So kann nachträglich ein Täter mithilfe des Videomaterials erkannt werden. Außerdem wirkt die Kamera abschreckend auf potenzielle Einbrecher. Sie hat jedoch keinen Zugang zu Maßnahmen, die einen bereits ablaufenden Einbruch verhindern könnten. Zudem wird durch das kontinuierliche Aufzeichnen sehr viel Speicherplatz in Anspruch genommen, was zu Abzüge bei der Videoqualität führt.

B. Einbruchmeldeanlagen

Ebenso beliebt sind Alarmanlagen, welche laut [1] aus mehreren Elementen bestehen: Die Melder sind Sensoren, die eine Gefahr erkennen sollen und somit das Signal für einen Alarm geben können. Dabei gibt es eine Vielzahl an Sensoren, zum Beispiele einen Glasbruchsensor, der an Türen oder Fenstern angebracht wird und erkennt, wenn diese eingeschlagen werden oder der Bewegungsmelder, welcher mit Ultraschall Entfernungen misst und dadurch in der Lage ist eine Bewegung zu registrieren.

Der Signalgeber wird ausgelöst, wenn ein Grund für einen Alarm erkannt wurde. Durch einen akustischen oder auch optischen Alarm wird Aufmersamkeit erzeugt und ein Schockeffekt beim Eindringling ereicht.

Eine Verarbeitung der Informationen geschieht in der Zentrale der Alarmanlage. Wenn ein Sensor ein Problem meldet, aktiviert die Zentrale den Signalgeber. Progamme laufen ebenfalls in der Zentrale ab.

Letzlich bleibt noch das Bedienelement, das die Alarmanlage aktiviert oder deaktiviert und anzeigen kann, wodurch der Alarm ausgelöst wurde.

An diesem Konzept orientiert sich auch der Patrouillenroboter, wobei vorallem die Idee des Bewegungsmelders einen großen Einfluss hatte.

Abbildung 1: Der Knightscope K5

C. Knightscope K5

Eine Inspiration und ein aktuelles Beispiel für einen Sicherheitsroboter liefert der K5 [Abb. 1] vom Unternehmen Knightscope, welcher sogar schon

DOI: 10.24352/UB.OVGU-2019-062

Anwendung in einzelnen Kaufhäusern findet. Die Maschine bewegt sich autonom über ein bestimmtes Gebiet und verfügt über verschiedene Funtkionen, die das Gelände sicher machen sollen. So besitzt er eine Kamera mit Rundumsicht, die durch Infrarot auch bei Dunkeltheit ein Video streamt und speichert. K5 ist dabei in der Lage selbstständing Gefahren wahrzunehmen. Genauer gesagt, kann er mithilfe von Gesichtserkennung gesuchte Verbrecher oder dergleichen identifizieren. Auf Parkplätzen untersucht er nebenbei die Kennzeichen und erfasst so als gestohlen gemeldete Autos. Weiterhin ist es ihm möglich, verletzte beziehungsweise hilfsbedürftige Personen zu erkennen, indem er ihre Bewegungen und Gesten analysiert. Fortführend kann er mit einem Alarmton auf sich aufmerksam machen und ist so eine große Hilfe für Sicherheitskräfte.

Abbildung 2: Frontansicht des Patrouillenroboters

III. Verwirklichung

In den folgenden Abschnitten wird die Realisierung des Projektes genauer beschrieben.

Vorab zwei benötigte Gleichungen:

$$U = 2 \bullet \pi \bullet r \quad (1)$$

U=Umfang, r= Radius

$$M = (360 \bullet s) \div W \quad (2)$$

M=Motordrehzahl, s=Weg, W= Radumfang

A. Konstruktion

Für den Bau des Roboters wurden nur wenige Teile benötigt. An den Seiten des NXT sind zwei Motoren für den Antrieb des Roboters angebracht. Der erste Prototyp war ein Kettenfahrzeug. Dieses war jedoch instabil und hatte Schwierigkeiten beim Drehen, darum wurden die Ketten durch Räder ersetzt. Außerdem besitzt der Roboter hinten ein kleines, bewegliches Stützrad, welches das Drehen erleichtert. Kernstück der Konstruktion ist der Ultraschallsensor, welcher auf einer senkrecht zum Erdboden zeigenden „Antenne“ befestigt ist. Durch einen weiteren Motor und zwei Zahnräder ist die Drehung der Antenne um 360 Grad möglich und somit die Rundumsicht des Ultraschallsensors gewährleistet. Weiterhin ist vorn ein Geräuschsensor verbaut.

Um an das Konzept einer Alarmanlage anzuschließen, kann man den Ultraschallsensor und den Geräuschsensor als die Melder sehen, den NXT als Zentrale und Signalgeber und die später erläuterte GUI als Bedienelement.

B. Einhaltung des Weges

Wie bei einer Patrouille sollte der Robter eine festgelegte Route abfahren. Dabei war es wichtig, die Route verändern zu können und eine Einhaltung dieser Route zu garantieren.

Das Programm bietet zwei Fahrtwegoptionen, eine Strecke und ein Quadrat. Mithilfe der Benutzeroberfläche kann die gewünschte Form gewählt und die Gesamtlänge der Strecke, beziehungsweise die Seitenlänge des Quadrats, in Dezimeter eingestellt werden. Dafür wurde ausgerechnet welches Tacholimit nötig ist, um den Roboter um einen Dezimeter fortzubewegen. Dazu wurde die Gleichung (1) benutzt, um den Umfang des Rades herauszubekommen. Nehmen wir an, dass der Umfang vier Zentimeter beträgt, dann müsste sich das Rad zweieinhalb mal drehen, um sich einen Dezimeter fortzubewegen. Mit Gleichung (2) kann die Anzahl der Umdrehungen als eine Motordrehzahl angegeben werden. Das benötigte Tacholimit wird mit einer Variablen, die durch den Nutzer bestimmt wird, multipliziert. Die Anzahl der Wiederholungen, also wie oft er die Route abfahren soll, kann ebenfalls bestimmt werden. Der Mindestwert hierfür beträgt eins.

Wurde die Strecke gewählt, fährt der Roboter zunächst solange vorwärts bis er die gewünschte Länge erreicht hat. Danach wird der Wert der Power der Motoren mit minus eins multipliziert, um ihn rückwärts fahren zu lassen, bis er wieder an seiner Startposition angekommen ist. Dieser Vorgang kann dann beliebig oft wiederholt werden.

Ist die Auswahl auf das Quadrat gefallen, fährt der Roboter ebenfalls zuerst vorwärts. Wurde die eingestellte Länge erreicht, bleibt er stehen und dreht sich um 90 Grad. Dazu wird nur der links angebrachte Motor

betrieben, der rechte Motor wird nicht angesteuert. Somit wird eine Drehbewegung nach rechts erreicht. Das benötigte Tacholimit für die Drehung kann bestimmt werden, indem man die Rechtsdrehung als Kreis betrachtet, wobei das rechte Rad den Mittelpunkt einnimmt. Der Abstand zwischen dem linken und rechten Rad bildet den Radius. Hat man diesen gemessen, lässt sich wieder mit Hilfe der Gleichung (1) der Umfang berechnen. Dieser wird als Weg in Gleichung (2) eingesetzt und ergibt die erforderliche Motordrehzahl für eine 360 Grad Drehung. Da der Wert für eine 90 Grad Drehung gebraucht wurde, wird das Ergebnis mit einem Viertel multipliziert. Das Produkt ist das nötige Tacholimit für eine 90 Grad Drehung.

Nach dieser Drehung fährt der Roboter weiter bis die geforderte Seitenlänge erreicht wurde. Wenn er sich drei mal gedreht hat, fährt er wieder zu seinem Startpunkt, das heißt, dass er das Quadrat einmal abgefahren hat. Eine weitere, vierte Drehung bringt ihn zurück in seine Ausgangsposition, von der aus er diesen Vorgang wiederholt.

C. Überwachungs- und Sicherheitssystem

Bei einer Patrouille ist es nicht nur wichtig eine feste Route einzuhalten, währenddessen soll auch die Umgebung ausgekundschaftet und überwacht werden.

Um dies zu ermöglichen nutzt der Roboter hauptsächlich den verbauten Ultraschallsensor und teilweise den Geräuschsensor. Dabei folgt er einem bestimmten Prinzip. An jeder Ecke des Quadrats beziehungsweise an jedem Ende der Strecke bleibt der Patrouillenroboter stehen und führt eine Messung mit Hilfe des Ultraschallsensors aus. Dafür wird er durch den an der Antenne angebrachten Motor um 360 Grad gedreht, wobei intervallweise die Distanz zu Objekten in seiner Nähe gemessen wird. Mit diesen Werten wird eine Matrix gefüllt, eine für jede Ecke des Quadrats beziehungsweise für jeden Endpunkt der Strecke. Dabei ist zu beachten, dass der Roboter zunächst immer eine Erkundungsfahrt macht, das heißt, dass er den gewünschten Weg einmal abfährt und ihn observiert. Die dadurch erhaltenen Werte sind die unveränderlichen Ausgangswerte, welche sich der Roboter stets merkt.

Erst bei den Wiederholungsdurchläufen beginnt die eigentliche Überwachung. Die gemessenen Distanzwerte werden ebenfalls in eigenen Matrizen gespeichert. Hat er alle für die Wiederholung nötigen Werte genommen, werden diese mit den Anfangswerten verglichen. Dafür werden die Matrizen der gleichen Ecke bzw. des gleichen Endes subtrahiert und liefern einen Differenzwert. Diese Differenz darf weder zu groß noch zu klein sein, wobei der Grenzwert hierfür bei plus/minus zehn Zentimetern liegt. Das bedeutet, dass eine Distanzänderung von über zehn Zentimetern, sei es eine Näherung oder eine Entfernung eines Objektes, vom Roboter erkannt wird. Wenn dies nicht der Fall ist, beginnt der nächste Durchlauf. Die dadurch erfassten Werte überschreiben die vorherigen Vergleichsmatrizen. Die Ausgangswerte bleiben hierbei gleich. Wenn jedoch eine Veränderung festgestellt, wurde aktiviert sich das Alarmsystem. Der Roboter bleibt auf der Stelle stehen und spielt einen Alarmton, der störend und aufsehenerregend ist. Er kann nur durch das Programm selbst abgestellt werden.

Auch der Geräuschsensor übernimmt eine Überwachungsfunktion. Er funktioniert wie ein Mikrofon und misst fast durchgängig die Umgebungslautstärke. Nur wenn sich der Ultraschallsensor dreht, wird keine Lautstärke gemessen, da die Motorengeräusche sonst ein Störfaktor wären. Wenn die gemessene Lautstärke zu hoch ist und einen Schwellwert überschreitet, wird ebenfalls der Alarm aktiviert.

Um die Messwerte zu veranschaulichen, wird in der programmeigenen GUI eine Übersicht der Messungen geboten. Jede Ecke beziehungsweise jedes Ende bekommt ein eigenes Kurvendiagramm, in dem die Anfangswerte und Vergleichswerte als Graph dargestellt werden. Die Ausgangswerte sind blau und die Vergleichswerte rot gefärbt, so kann der Nutzer erkennen welcher Wert sich geändert und den Alarm ausgelöst hat. Des Weiteren werden die aktuell gemessene Distanz und Lautstärke einzeln ausgegeben. Hierbei wird auch angezeigt, welcher Wert für einen Alarm verantwortlich war.

Abbildung 3: Rückansicht des Patrouillenroboters

IV. Ergebnisdiskussion

Das Ergebnis des Projekts ist ein Sicherheitsroboter, der auf einem veränderbarem Weg patrouilliert und dabei durch kontinuierliche Entfernungs- und Lautstärkemessungen seine Umgebung bewacht. Bei der Entstehung des Roboters traten einige Probleme auf, die leider nicht alle gelöst werden konnten und somit die Funktionalität bis zum aktuellen Stand einschränken.

Das größte Problem war oder ist das Abkommen vom Weg. Dem Roboter gelingt es nicht immer auf der vorgegebenen Route zu bleiben, vor allem nicht, wenn das Quadrat gewählt wurde. Bei der Strecke hat er dagegen keine großen Probleme. Erklärung hierfür sind die Drehbewegungen, die beim Abfahren des Quadrats auftreten, denn schon eine minimale Abweichung von 90 Grad kann aufgrund der mehrmaligen Wiederholungen zu starken Abweichungen führen. Dabei spielen die Ungenauigkeiten der Legoteile und Unebenheiten im Boden eine Rolle.

Durch das eben genannte Phänomen entsteht ein weiteres schwerwiegendes Problem und zwar Fehler beim Messen der Entfernungen. Wenn der Roboter von seiner Bahn abkommt, können auch die Messungen nicht übereinstimmen, wodurch der Roboter fälschlicherweise eine Gefahr erkennt. Außerdem hat der Ultraschallsensor Schwierigkeiten beim Erkennen von Objekten, die schräg zu ihm stehen. So kommen häufig unterschiedliche Werte bei unveränderten Entfernungen zu Stande. Auch hierdurch kann ein Fehlalarm ausgelöst werden.

Eine weitere Problematik wurde sogar bei unserer Abschlusspräsentation während der Vorführung des Roboters deutlich. Einer der Betreuer klatschte in die Hände, um mit Hilfe des Geräuschsensors einen Alarm auszulösen. Da die Lautstärke aber, wie ihn Abschnitt C des Haupteils erklärt, nicht durchgängig gemessen wird, gibt es sehr kurze Momente in denen laute Geräusche nicht erkannt werden. Unglücklicherweise fiel das Klatschen in so einen Moment.

Weiterhin besteht das Problem, das Distanzwerte erst miteinander verglichen werden, wenn alle Matrizen gefüllt sind. Das bedeutet, dass zum Beispiel der erste Wert der ersten Vergleichsmessung sich verändert haben kann, doch der Roboter dies erst bemerkt, wenn er die restlichen Messungen, also den Wiederholungsdurchlauf abgeschlossen hat. Dies verzögert die Erkennung eines Fehlers.

Trotz all dieser Ärgernisse ist das Produkt des Projektes zufriedenstellend. Der Roboter ist in der Lage das zu tun, was er sollte. Er führt seine Befehle aus und die GUI funktioniert ebenso problemlos.

V. Zusammenfassung und Fazit

Zum Abschluss kann man sagen, dass es möglich ist einen Roboter zu bauen, der die Aufgaben eines Wachmannes übernehmen könnte. Dabei hatten andere moderne Konzepte der Sicherheitstechnik, wie die Alarmanlage oder der Bewegungsmelder, einen Einfluss. Der Nutzen des Roboters wird jedoch durch die Grenzen der Lego Mindstorms Teile eingeschränkt und wäre noch kein verlässlicher Ersatz für einen Menschen.

Außerdem gibt es noch viel Raum für Verbesserungen, wie zum Beispiel die Ermöglichung des autonomen Fahrens durch Lichtsensoren oder das Anbringen einer Webcam, um auch die Funktion einer Sicherheitskamera einzubeziehen. Trotzdem könnte der Roboter bei einer Bewachung hilfreich sein.

Literaturverzeichnis

[1] Christian, Wie eine Alarmanlage funktioniert, https://alarmanlage-haus.net/wie-eine-alarmanlage-funktioniert/ , Abruf 18.03.2019

Konstruktion eines autonomen Portalstaplers

Niklas Ritz, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

Abstract—**In deutschen Häfen müssen Container von Brückenfahrzeugführern umgeschlagen werden. Dieser Umstand ist analog zum autonomen Fahren zu sehen und kann mithilfe von autonom agierenden Fahrzeugen verbessert werden.**
Im nachfolgenden Beitrag wird auf Basis der baulichen Eigenschaften eines Portalkrans eine technische Lösungsmöglichkeit dargestellt, die autonom agierend diese Umschlagvorgänge durchführt. Der dabei entstandene Portalstapler wird über eine GUI angesteuert und ist in der Lage, Container selbstständig zu Transportieren, zu Scannen und zu Sortieren.

***Schlagwörter*—Automatisierung, Autonom, Container, Portalkran, Transport**

I. Ausgangssituation

Im Folgenden Abschnitt wird die Situation beschrieben, die zur Entwicklung des Portalstaplers geführt hat. Außerdem werden die konkrete Umsetzung und die verwendeten Komponenten aufgeführt.

Eine vom Hamburger Hafen im Jahr 2018 herausgegebene Statistik führt die pro Jahr umgeschlagene Anzahl an 20-Fuß-Standardcontainern seit 1990 auf [4]. Dabei werden sowohl Import, als auch Export berücksichtigt und auch gesondert gelistet. Interessant ist hierbei, dass sich die absolute Anzahl an Gesamtumschlägen, also die Summe aus Import-und Exportumschlägen im Vergleich zu 1990 mehr als verdreifacht hat. So wurden 1990 insgesamt 1.968.986 20-Fuß-Container umgeschlagen, 2018 hingegen waren es 8.726.442 Container.
Dazwischen stiegen die Umschlagszahlen bis 2007 auf ein Maximum von 9.889.792, bevor sie sich konstant um den nun auch 2018 erreichten Wert gruppierten. So blieb diese Umschlagszahl seit 2015 weitestgehend konstant.

A. Problemstellung

Diese Umschlagvorgänge von Containern in deutschen Häfen müssen aufgrund gesetzlicher Vorschriften von einem Brückenfahrzeugführer vorgenommen werden. Dies zwingt die Hafenverwaltung dazu, alle Vorgänge mithilfe von Personal abzuwickeln, was ein Risiko für den reibungsfreien Ablauf im Hafen darstellt. Da immer größere "Ultra Large Container Ships" mit Kapazitäten von bis zu 21.000 Containern gebaut und eingesetzt werden, müssen immer schnellere und effizientere Warenumschläge realisiert werden, was auf Dauer allein durch Personalaufstockung nicht zweifelsfrei gewährleistet werden kann.
Ein modernes Containerschiff kann je nach Bauart mehrere sogenannter 20-Fuß Containereinheiten (TEU) laden. Diese müssen dann möglichst schnell und effizient in den Häfen umgeschlagen werden, um einen reibungsfreien und profitablen Ablauf zu gewährleisten. In der Regel werden dafür Portalkräne (PK) verwendet, um Container zu Entladen und zu Sortieren. Anschließend können die Behälter von Portalhubwagen (PHW) zu Lagerplätzen, Lastkraftwagen oder Zügen abtransportiert werden. PK und PHW müssen jedoch laut gesetzlicher Verordnung in Deutschland von einem Brückenfahrzeugführer bedient werden und dürfen nicht autonom agieren. Dadurch sind jedoch nur circa 20 bis 30 Umschlagvorgänge pro Stunde bei einem PK möglich. Hinzu kommt auch der Risikofaktor Mensch: Durch Ermüdung, Stress und Ablenkung können schwere Unfälle oder Verladefehler entstehen, im Falle einer Krankheit oder Rente muss Ersatz gefunden werden und es sind bestimmte Arbeitsschutzmaßnahmen und Ruhezeiten einzuhalten.

B. Lösungsansatz

Dies lässt sich mithilfe eines autonom agierenden PHW´s optimieren, da er keinen Kranführer benötigt und ohne Pause Umschlagvorgänge durchführen kann. Das Ziel des Projekts ist es, ein Fahrzeug zu konstruieren, welches sich geradlinig auf Rädern bewegen kann und die Bauart eines Portalkrans, beziehungsweise Portalhubwagens übernimmt. Es soll dem Fahrzeug möglich sein, Container anhand ihrer Farbe zu Erkennen und diese dann aufzunehmen und zu Sortieren. Weiterhin soll das Fahrzeug nicht benötigte Container mit und ohne aufgenommene Last geradlinig "überfahren" können.

C. Umsetzung

Diese Zielstellung wird mithilfe eines Lego-Technik-Bausatzes der Lego-Reihe "Lego-Mindstorms" realisiert und über eine MATLAB-Erweiterung der RWTH Aachen programmiert. Dafür werden ein Lego-NXT-Baustein, zwei Farbsensoren und drei Lego-Motoren verwendet.
Der NXT-Baustein fungiert als Schnittstelle zwischen dem Computer und den eingebauten Sensoren und Motoren.
Zur Erkennung der Container werden zwei Lego NXT 2.0 Farbsensoren des Lego-Mindstorms-Sets verwendet, welche bei Erkennung eines Wertes die Ausgabe "BLUE", "YELLOW", "RED", "GREEN" oder "WHITE" liefern. Wird keine Farbe in Reichweite erkannt oder ist der Gegenstand sehr dunkel, wird "BLACK" ausgegeben.
Den drei Motoren sind die Aufgaben Antrieb, Heben/Senken und Gabel ein-und ausfahren zugedacht. Motor A sorgt für die Vorwärts- und Rückwärtsbewegung des gesamten Fahrzeugs, Motor B hebt und senkt den Lastschlitten, welcher die Container aufnimmt und Motor C fährt die Gabeln ein und

DOI: 10.24352/UB.OVGU-2019-063

aus, die die Behälter anheben. Damit sind alle Motorschnittstellen am NXT-Stein belegt.
Als Container werden aus den Lego- Standardfarben Rot, Blau, Weiß, Grün und Gelb Quader gefertigt, die auf einem Gestell aus Lego-Technik Elementen befestigt sind, sodass die Farbsensoren eine optimale Erkennung der Farbwerte liefern.

II. Stand der Technik

Es erfolgt nun eine kurze Darlegung der aktuell verwendeten und eingesetzten Systeme und die daraus entstandene Bauart des Projektes.

A. Reale Vorbilder

Portalkräne und Containerbrücken werden hauptsächlich im Freien eingesetzt, um große oder sperrige Lasten zu bewegen. Folgende Merkmale zeichnen diese Konstruktion aus:

Sie besteht aus zwei senkrechten, schienenläufigen Stützen, zwischen denen eine Brücke installiert ist, an der beweglich oder statisch das Hubwerkzeug angebracht ist. Je nach Bauart, Hersteller und Abmessungen können auch mehrere Hubwerkzeuge installiert sein und verschiedene Krantypen vorliegen.

Es wird hauptsächlich zwischen den folgenden Arten unterschieden: Es gibt sogenannte RTG-Kräne, die statt auf Schienen auf gummibereiften Rädern fahren, Halbportalkräne, die nur eine Stütze besitzen und meist in Hallen eingesetzt werden und statisch fixierte Bockkräne. Der Antrieb kann sowohl rein elektrisch als auch über Verbrennungsmotoren und Aggregate erfolgen. Die Steuerung erfolgt von einem in der Brücke eingesetzten Führerhaus oder von einem externen Kontrollstand aus, was jedoch fast ausschließlich in geschlossenen Hallen angewandt wird. Für das Verbringen über längere Strecken werden PHW eingesetzt, die kleiner als PK dimensioniert sind, sich jedoch in Aufbau und Betriebsvorschriften kaum unterscheiden.

B. Konstruktion

Das Hauptaugenmerk soll auf der Transportfähigkeit über größere Strecken liegen. Damit entfällt eine Bauweise gleich der eines örtlich gebundenen Portalkrans und es wird ein Antrieb notwendig, der das gesamte System bewegen kann. Daher soll der Portalhubwagen als Vorbild dienen, da es sich um einen klein ausgeführten Portalkran auf Rädern oder Schienen handelt, der meist ein einziges Containerelement über größere Strecken auf dem Hafengelände transportiert.

Ziel des Projektes ist es also, ein autonomes Fahrzeug zu konstruieren, welches die aufgenommene Last, einen einzigen Container, über größere Strecken transportieren kann. Dabei soll es dem Fahrzeug möglich sein, die Container zu ordnen, zu sortieren und zu unterscheiden, jedoch nicht zu stapeln.

III. Merkmale und Funktionsweise

A. Bauliche Merkmale

Der autonome Portalstapler ist als vierbeiniger RTG-Portalhubwagen konstruiert worden. Dies ist das wohl charakteristischste konstruktive Merkmal, da das Fahrzeug die Vorteile eines Portalkrans und eines Gabelstaplers miteinander vereint: Es ist möglich, während des Transports andere Behälter einfach ohne Ausweichen zu "überfahren" und das feinfühlige Ausrichten der Seilwinde und das Einklinken der Hubvorrichtung entfallen, die Behälteraufnahme erfolgt also wie bei einem Gabelstapler durch zwei bewegliche Gabeln, die unter den jeweiligen Container geschoben werden.

Abbildung 2: Gesamtansicht

Der Antriebsmotor sitzt oben auf dem Gerüst auf und die Kraftübertragung erfolgt mithilfe von Zahnrädern entlang der beiden hinteren Stützsäulen zu den Hinterrädern. Nur so ist es möglich, die Zielstellung, das Heben und Sortieren von Containern übereinander hinweg, zu realisieren. Deshalb wurde eine Bauweise gleich der eines klassischen Gabelstaplers verworfen. Ebenfalls wurden oben auf dem Portal der NXT-Stein zur Ansteuerung der Sensoren und Motoren und der Motor des Lastaufzuges installiert. In Fahrtrichtung links sitzt zwischen den beiden Rädern der Lastschlitten mit den beweglichen Containergabeln, die durch den letzten, dritten Motor, welcher auf dem beweglichen Lastaufzug montiert ist, ein-und ausgefahren werden können.
Die beiden Farbsensoren wurden an der Front des Portalstaplers nach unten zeigend an einer Querstrebe zwischen den beiden vorderen Beinen direkt hintereinander montiert. Außerdem wurde ein Pflug in beide Fahrtrichtungen und eine Führungsschiene innerhalb des Portals angebracht, um die Container für die Aufnahme optimal auszurichten.
Zur besseren Erkennung wurden die Container einfarbig in Rot, Weiß, Blau, Gelb und Grün aus klassischen Lego-Steckbausteinen und Lego-Technik-Elementen gebaut.

B. Erkennung der Container

Bewegt sich der Portalstapler vorwärts, scannen beide Farbsensoren den Untergrund kontinuierlich ab. Erkennt nun der vordere Sensor eine Farbe, wird überprüft, ob es sich um die Zielfarbe handelt. Ist dies nicht zutreffend, wird weitergefahren und -gescannt. Handelt es sich hingegen um die gewünschte Farbe, so wird nun auch der Wert des zweiten Sensors abgefragt. Solange der zweite Sensor die Zielfarbe erkennt, befindet sich der Stapler noch nicht genau über dem Container und fährt weiter. Erkennt der hintere Sensor jedoch die Farbe Schwarz, so bedeutet dies, dass der Container am zweiten Sensor vorbeigezogen ist und nun aufgrund der Position des Sensors 2 der Behälter optimal steht.

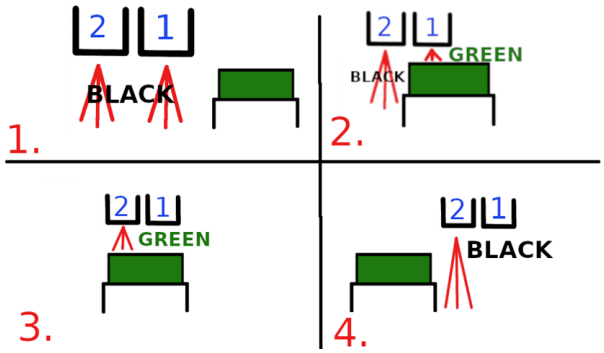


Abbildung 3:Schema der Erkennung

Nun stoppt das Fahrzeug, die Fracht wird nach dem Prinzip eines Gabelstaplers aufgenommen und es erfolgt der Rücktransport. Die zu fahrende Strecke wird hierbei aus der Anzahl der Motorumdrehungen ermittelt.

C. Ansteuerung und Betriebsmodi

Der Portalstapler wird mittels einer GUI angesteuert und verfügt über drei Betriebsmodi: Suchen, Scannen und Sortieren. Im ersten Modus sucht der Stapler den Container in der eingegebenen Farbe aus einer Anzahl unterschiedlich gefärbter Container heraus und bringt diesen zum Ausgangspunkt zurück und stellt ihn dort ab. Im Scanmodus fährt der Portalstapler von seinem Ausgangspunkt aus eine beliebig große, im aktuellen Fall bis zu fünf Einheiten lange, Containerkette ab und gibt dann in Echtzeit die Reihenfolge der Farben, also die Behälterreihenfolge an. Anschließend kehrt er zum Ausgangspunkt zurück. Die Anzahl der zu verarbeitenden Einheiten kann vorher in der GUI festgelegt werden, da dies den Einsatz nutzerfreundlicher und flexibler macht. Im Sortiermodus fährt das Fahrzeug jeden der Container entsprechend der eingegebenen Farbreihenfolge an und stellt die Behälter dann entsprechend der eingegebenen Reihenfolge vom Ausgangspunkt an aus rückwärts ab. Der erste Behälter steht also am Ausgangspunkt, der zweite in Fahrtrichtung rückwärts dahinter und so weiter. Zur Warnung wird beim Transportvorgang ein regelmäßiger Warnton abgegeben und nach jedem Vorgang ein Bestätigungston oder eine Tonfolge bei beendeten Aufgaben, um den aktuellen Betriebsstatus zu signalisieren.

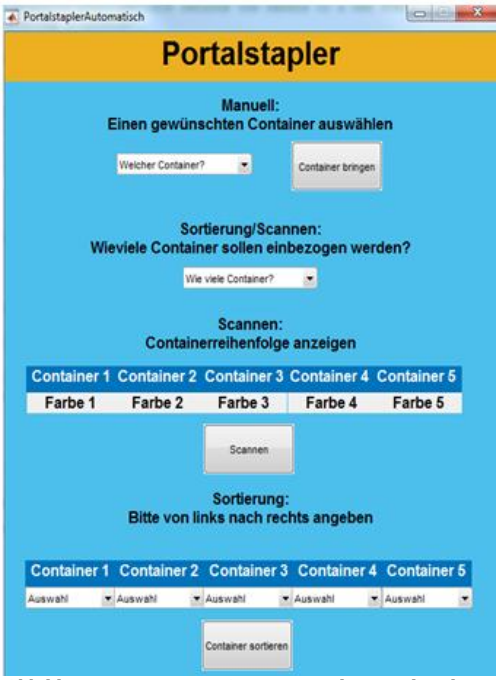


Abbildung 4:Die GUI zur Ansteuerung des Portalstaplers

IV. Ergebnisdiskussion

Das Ergebnis ist ein autonom agierender Portalstapler, der lediglich über die Auswahl der Container oder der Wahl der Betriebsmethode durch den Menschen bedient wird. Alle anderen Schritte, wie das Erkennen, Ordnen, Scannen, Aufnehmen und Verbringen der Behälter werden vom Fahrzeug selbst ausgeführt.

Die angestrebte Präzision bei Erkennung und Transport wurde durch direkt hintereinander angebrachte Sensoren erreicht. Die Containeraufnahme erfolgt deshalb zuverlässig und fehlerfrei.

Das aufnehmbare Gewicht ist aufgrund der Stabilität der Legobauteile begrenzt, jedoch wird dieser Grenzwert durch die Lego-Behälter nicht erreicht.

Mithilfe der GUI kann der Nutzer direkt festlegen, wie viele Behälter er berücksichtigen möchte und ob er diese Scannen, Sortieren oder nur einen speziellen Container gebracht haben möchte. Auf die Geschwindigkeit, mit der der Stapler agiert, hat der Bedienende hingegen keinen Einfluss.

Die Vorgänge funktionieren aufgrund der Abstimmung der Komponenten und dem Pflug zur Containerausrichtung nahezu reibungsfrei, lediglich die ungenaue Ausrichtung der Behälter im Vorfeld kann zu unregelmäßigen Abständen führen, wenn der Pflug beginnt, Container zu schieben, da diese sich teilweise ineinander verkeilen.

Abbildung 5: Einer der verwendeten Container

V. Zusammenfassung und Fazit

Abschließend wird nun ein Fazit zu den Ergebnissen gezogen und mögliche Probleme werden aufgezeigt.
Im Verlauf des Lego-Mindstorms Projektseminars ist es gelungen, ein autonom agierendes Fahrzeug zu konstruieren, welches verschiedenfarbige Container Erkennen, Sortieren, Scannen und Transportieren kann. Die Behälter werden zuverlässig erkannt, von dem montierten Pflug in Position gebracht, korrekt aufgenommen und abtransportiert. Genau wie bei dem realen Vorbild, dem PHW, ist ein Schienen-oder Radbetrieb möglich und aufgenommene Container können ohne Probleme über die anderen Behälter hinweg gehoben und transportiert werden.
Der Vorteil des Staplers liegt in seiner Automatisierung: Durch die maschinelle Erfassung und akkurate Arbeitsweise wird ein schneller, effizienter und reibungsfreier Ablauf gewährleistet.
Aufgrund des zeitlichen Rahmens war es nicht möglich, eine Stapelfunktion zu integrieren. Dies hätte ab einer Höhe von mehr als zwei Containern ohnehin zu konstruktionstechnischen Problemen geführt, da das Fahrzeug baulich nicht dafür ausgelegt wurde. Es kann stets nur eine Containerhöhe mit oder ohne aufgenommene Last "überfahren" werden.
Bei Implementierung dieser Funktion nach Umbau des Gerätes würde sich der Funktionsumfang allerdings den realen Anforderungen annähern, da die Container nun platzsparender sortiert und gelagert werden könnten.
Eine Funktion für eine variable Strecke oder festgelegte Ablagestellen, die das Fahrzeug ansteuern kann, konnte wegen der räumlichen Limitierung durch das USB-Kabel, welches NXT-Baustein und PC miteinander verbindet, nicht umgesetzt werden. Dies wäre auch durch die Schnittstellenanzahl des NXT-Steins unmöglich geworden, da ein NXT-Baustein nur über 3 Schnittstellen für Motoren verfügt und bereits alle Stellen belegt wurden, sodass die automatisierte Lenkung durch das Fahrzeug selbst entfällt.
Dieses Problem wiederum kann aber mit einem weiteren NXT-Stein und einer drahtlosen Verbindung zwischen PC-Steuerung und dem Fahrzeug behoben werden. Dann wäre es möglich, dass das Fahrzeug mittels eines weiteren Motors gesteuert wird und weitere Sensoren auch Fahrtwege und Hindernisse erkennen würden, sodass der Portalstapler vollends autonom und frei fahrend agiert.
Auch gibt es einen weiteren, denkbaren Ansatz für die Erkennung der Ladung: Nicht jeder Container kann eine andere Farbe erhalten, zumal dadurch auch leicht Verwechslungsgefahr bestünde, weshalb hier ein QR-Code Scanner Abhilfe schaffen könnte, um auch größere Behälteranzahlen bearbeiten zu können.
Eine weitere Alternative wäre der Betrieb des Systems auf Schienen. Damit würde man die klassischen Probleme eines Reifenfahrzeuges, wie Reifenverschleiß und Druckverlust, umgehen. Außerdem könnte man dann die Versorgung des Systems wie bei Zügen und Straßenbahnen mittels einer Oberleitung sicherstellen, was den Einbau von Batterien unnötig machen würde, falls das Fahrzeug elektrisch betrieben werden sollte.
Es stellen sich natürlich die gleichen Fragen wie beim Betrieb von autonomen Drohnen und Automobilen: Wie wird mit Unfällen verfahren? Wie können software-oder hardwarebedingte Ausfälle verhindert werden? Wie schützt man sich gegen Cyber-Angriffe? Wann werden die gesetzlichen Voraussetzungen für den Einsatz gegeben sein?
Im Falle einer Zulassung von autonom agierenden Fahrzeugen ist der Portalstapler eine Lösung für das eingangs geschilderte Problem. In großen Häfen wie Hamburg, Rotterdam in den Niederlanden oder Singapur wäre es möglich, mehrere Stapler miteinander zu vernetzen, um Kollisionen zu verhindern und so den Transfer noch effizienter zu gestalten. Auch autonome Lastwagen oder Lastendrohnen wären dann denkbare Hilfsmittel im Handelsverkehr.
Es bleibt natürlich abzuwarten, ob und wann ein solcher Betrieb in der Bundesrepublik Deutschland gesetzlich möglich gemacht werden wird.

Literaturverzeichnis

[1]"Portalkran" aus
https://www.konecranes.com/de-de/fachlexikon/portalkran,
Zugriffsdatum: 24.02.2019
[2]"Containerbrücke" aus
http://www.wikiwand.com/de/Containerbr%C3%BCcke
Zugriffsdatum: 24.02.2019
[3]"Ultra Large Container Ship" aus
https://de.wikipedia.org/wiki/Ultra_Large_Container_Ship
Zugriffsdatum: 24.02.2019
[4] "TEU-Statistik" aus
https://www.hafen-hamburg.de/de/statistiken/containerumschlag,
Zugriffsdatum:12.06.2019

Portalstapler zum Suchen, Sortieren und Scannen von Containern

Tim Christopher Tadge, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Abstract*— Das Projekt „Portalstapler“ entstand im Rahmen des „Lego-Mindstorms-Seminars“ und hatte das Ziel einen automatisierten Portalkran zu entwickeln. Dieser erlaubt das Erkennen, Aufnehmen und den Transport von Containern. Der „Portalstapler“ setzt sich aus Elementen eines Portalkrans und eines Gabelstaplers zusammen. Ein wichtiges bauliches Merkmal ist der seitlich angebrachte Lastschlitten, welcher dem Anheben der Container dient. Die Programmierung erfolgte in Matlab, unter Verwendung des Lego-Mindstorms NXT Toolkits der RWTH Aachen. Der Anwender hat die Möglichkeit Container zu suchen, sie zu sortieren oder deren Farben zu scannen. Die Erkennung der Container erfolgt über zwei hintereinander angebrachte Farbsensoren. Das Programm setzt sich aus aufeinander abgestimmte Teilprozesse zusammen, die sequentiell abgearbeitet werden.**

***Schlagwörter*—Automatisierung, Container, Gabelstapler, LEGO NXT, Matlab, Portalkran**

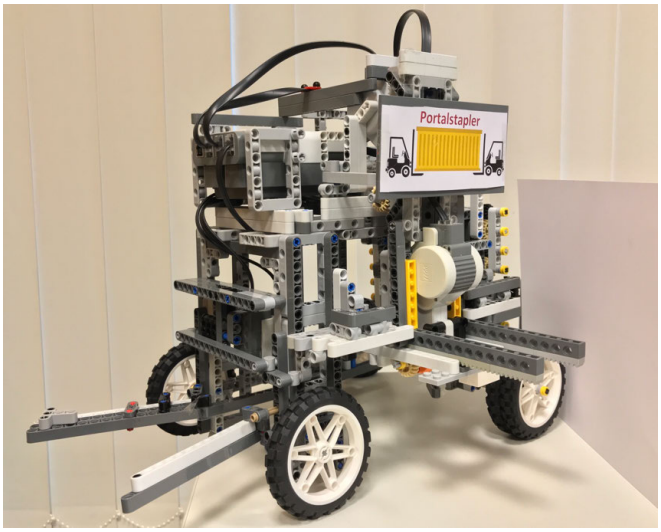


Abbildung 1: Gesamtansicht

I. Einleitung

In der heutigen Zeit werden Roboter bzw. automatisierte Prozesse in allen Lebensbereichen immer wichtiger. So werden Arbeitsabläufe einfacher und effizienter. Deshalb entstand im Rahmen des Lego-Mindstorms-Praktikums die Idee einen automatisierten Portalkran mit Hilfe von Lego NXT zu entwickeln. Ausschlaggebend dafür ist, dass jährlich rund Neunzig Millionen Tonnen Container im Hamburger Hafen umgeschlagen werden [1]. Jedoch geschieht dies noch immer manuell durch Arbeitskräfte, welche den Portalkran steuern. Dadurch sind nur zwanzig bis dreißig Umschlagvorgänge pro Stunde durch einen Kran möglich. Dem entgegen steht jedoch ein weltweit steigendes Güteraufkommen, welches schnelles und präzises Umschlagen der Container erfordert. Auch Aufgrund gesetzlicher Vorschriften ist ein automatisiertes Vorgehen in Deutschland bis jetzt nicht vorgesehen. Diese gesetzlichen Barrieren sind zu vergleichen mit selbstfahrenden Autos, welche auf deutschen Straßen noch nicht ohne Fahrer unterwegs sein dürfen.
Das Ziel besteht darin einen automatisierten Portalkran aus zu entwickeln, welcher das Suchen, Aufnehmen und Verbringen von Containern zum Ausgangsort ermöglicht. Es handelt sich also um drei voneinander getrennt zu betrachtende Prozesse, welche alle ein hohes Maß an Präzision erfordern.

II. Vorbetrachtungen

Der Portalstapler setzt sich aus mehreren verschiedenen Komponenten zusammen, welche ihre Vorbilder in der Industrie und dem Warentransport finden. Dazu gehört einerseits der Portalkran und andererseits Elemente eines Gabelstaplers. Daraus setzt sich auch der Name Portalstapler zusammen.

A. Portalkran

Ein Portalkran besteht aus einem Gestell, welches den Arbeitsbereich wie ein Portal überspannt [2]. Er wird auf Güterumschlag- und Lagerplätzen eingesetzt. Der Kran fährt meist auf zwei parallelen Schienen. Die Bauweise entspricht der eines Fachwerks bzw. einer Rahmenbauweise aus Stahl. Zum Anheben und Bewegen der Ladung fährt längs der Konstruktion die Laufkatze mit dem Hubwerk [2]. Über Seile wird dann, z.B. der Container angehoben. Ein Portalkran ermöglicht es dadurch die Last in alle drei Richtungen zu bewegen. Aufgrund der Bauart ist es möglich sehr große Lasten zu heben, wobei in der Regel kein Gegengewicht notwendig ist [2]. Eine Sonderform stellt der gummierte Portalkran dar. Der sogenannte RTG (rubber tyred gantry cran) kommt ohne Führungsschienen aus [2]. Dadurch wird ein Spurwechsel ermöglicht. Bedient wird der Portalkran durch einen Kranführer, welcher meist oben in der Kranbrücke sitzt.

DOI: 10.24352/UB.OVGU-2019-064

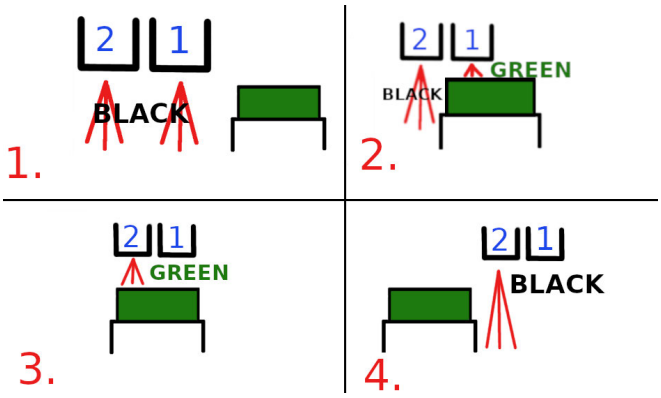


Abbildung 2: Schema der Containererkennung

B. *Gabelstapler*

Ein Gabelstapler dient dem innerbetrieblichen Warenumschlag und Transport. Ein wesentliches Element des Gabelstaplers ist die Hubeinheit, die aus Hubmast und Gabelträger zusammengesetzt ist [3]. Ausgelegt ist er insbesondere für den Transport von Paletten, das Prinzip der Warenaufnahme über Gabeln lässt sich jedoch auch anderweitig für den Warentransport implementieren. Der Gabelträger umfasst in der Regel zwei stählerne Gabeln, welche mittels Hydraulik vertikal bewegbar sind [3]. Auch Gabelstapler werden, wie ein Portalkran, durch mitfahrende Bediener gesteuert.

C. *Automatisierte Fahrzeuge in der Industrie*

Schon heute werden in der Industrie verschiedene Formen von automatisierten Fahrzeugen eingesetzt. Diese autonomen Logistikfahrzeuge dienen vorrangig dem Warentransport. Daraus ergeben sich eine Vielzahl wirtschaftlicher Vorteile, wenn es darum geht viele Güter umzuschlagen, oder andere sich wiederholende Logistikaufgaben auszuführen [4]. Hierfür gibt es Fahrzeuge im Einzelbetrieb oder im abgestimmten Flottenbetrieb [4]. Gerade hierbei spielt neben der Konstruktion auch die Software eine sehr wichtige Rolle. Durch die immer besseren technischen Möglichkeiten können so Gefahrensituationen schneller erkannt und Unfälle vermieden werden.

III. Realisierung

Bei der Umsetzung des Projekts war es wichtig, die als Ziel gesetzten Anforderungen zu erfüllen. Dabei galt es mit Hilfe von Lego Mindstorms NXT und Matlab, welches der Programmierung dient, alle grundlegenden Funktionen eines Portalkrans zu realisieren.

A. *Konstruktion und Mechanik*

Der grundsätzliche Aufbau des entwickelten Portalstaplers entspricht dem eines Portalkrans. Es ist durch die Bauweise eines Portals problemlos möglich über die Container hinweg zu fahren. Hierzu wird mit Hilfe eines Motors der Antrieb realisiert. Aufgrund der baulichen Voraussetzungen muss dieser Motor im oberen Teil des Fahrzeugs liegen. Um die Energie dennoch auf die unten liegenden Rädern zu bringen, erfolgt die Übertragung durch seitlich liegende Zahnräder. Bei den genutzten Rädern handelt es sich um eine Gummibereifung. Da nur ein Motor dem Antrieb des Portalstaplers dient, ist die Fahrweise trotzdem mit der auf Schienen zu vergleichen, welche bei einigen Portalkränen zum Einsatz kommt.

Der Unterschied des Portalstaplers zum echten Portalkran besteht darin, dass die Container nicht über Seile angehoben werden, sondern über einen Lastschlitten mit Gabeln, ähnlich denen eines Gabelstaplers. Als weitere bauliche Maßnahme befindet sich vorn und hinten am Fahrzeug ein Pflug, welcher dafür sorgt, dass die Container auf einer geraden Bahn beim Hubmechanismus ankommen. Den vorderen Pflug kann man in Abb. 1 sehen. Die Erkennung verschiedener Container wurde mit Hilfe von Lego derart angepasst, dass diese über verschiedene Farben realisiert wurde. Dazu sind vorn am Portalstapler zwei Farbsensoren angebracht. Diese befinden sich direkt hintereinander und nur rund einen Zentimeter über den Containern.

Abbildung 3: Weißer Container

Daraus ergeben sich letztlich konkrete Abmessungen für die Container, wodurch alle gleich genormt sind. Einen Container kann man in Abb. 3 sehen. Im Rahmen des Projekts wurden fünf Container in den Farben Blau, Gelb, Grün, Rot und Weiß gefertigt. Die Maße sind so gewählt, dass ein Container sich genau unter denn Farbsensoren hindurch bewegt und außerdem es möglich ist mit einen angehobenen Container über einen anderen zu fahren.

Der Lego Mindstorms NXT dient dazu alle mechanischen Komponenten zu verknüpfen. Über diesen erfolgt die Ansteuerung der jeweiligen Motoren und die Auswertung der Farbsensoren.

B. *Konstruktion und Erprobung des Lastschlittens*

Die Konstruktion des Lastschlittens orientiert sich größtenteils an der Hubeinheit eines Gabelstaplers. Hierbei galt es jedoch neben der vertikalen Bewegung des Schlittens, also der Gabeln, auch das Ein- und Ausfahren zu ermöglichen. Ohne diese zusätzliche horizontale Bewegung wäre es nicht möglich mit den Gabeln unter die Container zu gelangen. So war es auch nötig zwei Motoren für die Konstruktion des Last-

schlittens zu verwenden. Der fertige Lastschlitten ist in Abb. 4 dargestellt. Über Zahnschienen wird der gesamte Lastschlitten gehalten und mit Hilfe eines Motors vertikal bewegt. Am Motor befinden sich zwei Zahnräder, welche in die Zahnschienen greifen und so den Lastschlitten entweder halten oder hoch bzw. herunter bewegen.

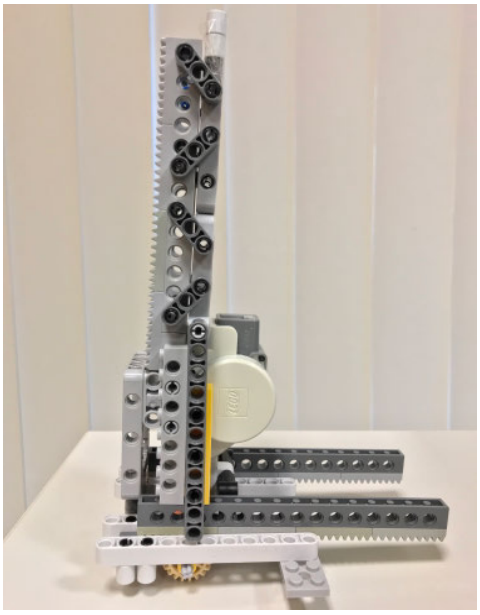


Abbildung 4: Lastschlitten

Am Lastschlitten selbst befindet sich ein weiterer Motor, welcher fest verbaut ist. Dieser dient dazu die zwei Gabeln vollständig heraus, also unter den Container zu fahren, oder wieder herein. Auch die Gabeln werden über Zahnräder, welche auf Zahnschienen laufen angetrieben. Bei der Erprobung des Lastschlittens erwies es sich, dass eine möglichst hohe Genauigkeit bei den einzelnen Bewegungsabläufen notwendig ist. So ist es möglich, sowohl die Hoch- bzw. Runterbewegung als auch das Hinein- bzw. Herausfahren präzise über die Umdrehungszahlen der einzelnen Motoren zu steuern. Die geforderten Umdrehungen haben sich durch die Anzahl der Zähne eines Zahnrades und der Aufnahme durch die Zahnschienen ergeben.

C. Programmierung

Die Programmierung des Roboters erfolgte vollständig in Matlab, unter Verwendung des Toolkits der RWTH Aachen. Um eine einfache und effiziente Ansteuerung des Portalstaplers für den Nutzer zu ermöglichen, erfolgt die Bedienung über eine graphische Benutzeroberfläche (GUI). Dazu wurden drei verschiedene Modi realisiert, welche vom Nutzer ausgewählt werden können. Dabei handelt es sich um das Suchen, Sortieren und Scannen von Containern. Im Folgenden wird hauptsächlich auf die Funktion des Suchens eingegangen und genauer erläutert.

Zuallererst wird es dem Nutzer ermöglicht aus einem Popup-Menü die gewünschte Farbe des Containers anzugeben und seine Auswahl dann zu bestätigen. Daraufhin fährt das Fahrzeug los und scannt mit dem weiter vorne liegenden Farbsensor kontinuierlich, ob die gewählte Farbe sichtbar ist. Solange keine Farbe erkannt wird, also kein Container sich unter dem Portalstapler befindet, oder eine andere Farbe erkannt wird, fährt das Fahrzeug weiter. Der Wert des Farbsensors wird dabei jede zehntel Sekunde erneut abgerufen. Wird der vom Nutzer gewählte Farbwert gemessen, so schaltet sich der zweite, weiter hinten liegende Farbsensor ein. Auch dieser misst nun jede zehntel Sekunde den Farbwert. Hierbei ist es so, dass solange, dieser Sensor die gewünschte Farbe misst der Portalstapler weiter fährt. Denn erst, wenn die eingegebene Farbe nicht mehr gemessen wird, ist der Container unter beiden Sensoren hindurch gefahren. In diesem Moment befindet er sich genau mittig und kann aufgenommen werden. Nun bleibt das Fahrzeug stehen und misst die insgesamt zurückgelegte Strecke, welche sich aus den Motorumdrehungen ergibt. Der Prozess der Containererkennung mit Hilfe der Farbsensoren ist schematisch in Abb. 2 dargestellt.

Im nächsten Programmabschnitt folgt die Containeraufnahme. Diese ist immer ein automatisierter Prozess. Zuerst fährt der obere Motor den Lastschlitten nach unten, woraufhin durch den anderen Motor die Gabeln ausgefahren werden. Folglich wird der Last-schlitten nun wieder nach oben fahren. Durch festgelegte Werte sind dabei die benötigten Umdrehungen definiert, welche den Motoren über ein Tacholimit übermittelt wird. Dadurch handelt es sich immer um die gleichen Bewegungsabläufe.

Sobald die Containeraufnahme beendet wurde, fährt der Roboter anhand der vorher ausgelesenen Motorumdrehungen zurück zum Ausgangsort. Dort wird der Container abgestellt, wobei es sich erneut um einen festgelegten Ablauf handelt. Dieser verläuft ähnlich wie der des Aufnehmens, mit dem Unterschied, dass sich die Gabeln diesmal einfahren. Nun ist der Programmablauf, welcher in Abb. 5 dargestellt ist, beendet und der Nutzer hat die Möglichkeit einen neuen Befehl ausführen zu lassen.

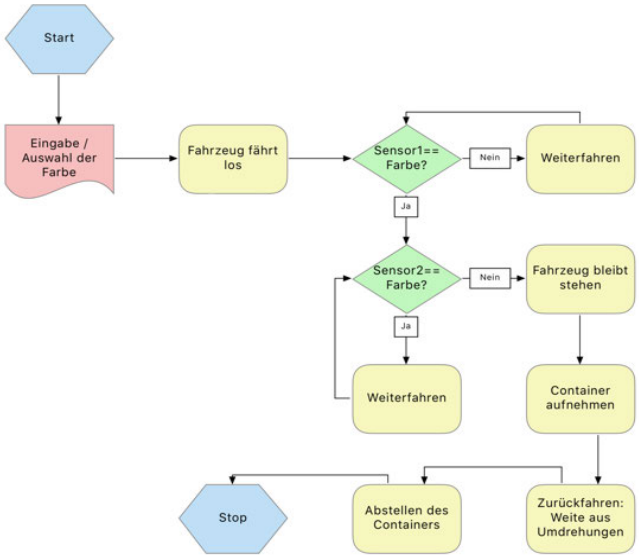


Abbildung 5: Programmablaufplan der Containersuche

Im Modus Sortieren kommt es zu ähnlichen Ablaufschritten mit dem Unterschied, dass der Anwender mehrere Containerfarben auswählen kann. Der Roboter holt dann einen Container nach dem anderen und stellt jeden ein wenig entfernt vom letzten auf. Beim Scannen wird die Bahn aus Containern einmalig abgefahren, woraufhin in der GUI die gemessenen Farben ausgegeben werden. Bei diesen beiden Modi wird vor dem Start jeweils mit Hilfe eines Popup-Menüs ausgewählt, wie viele Container in diesem Vorgang berücksichtigt werden sollen.

IV. Ergebnisdiskussion

Als Endergebnis ist ein voll funktionsfähiger, automatisierter Portalkran entstanden, welcher die Möglichkeit bietet unterschiedlich farbige Container zu suchen, aufzunehmen und zum Ausgangsort zu bringen. Diese grundlegenden Funktionen sind über drei Modi realisiert wurden. Letztlich funktionieren alle Funktionen zuverlässig.

Jedoch kam es während der Entwicklung des Roboters zu verschiedenen Problemen, auf welche im Folgenden eingegangen wird. So ergab sich zuallererst die Schwierigkeit, dass eine Aufnahme der Container mit Hilfe von Seilen nicht mit Lego und in den geplanten Dimensionen realisierbar ist. Dadurch entstand die Kombination des Portalkrans mit einem Gabelstapler.

Hieraus entwickelte sich jedoch das nächste Problem, die Konstruktion des Lastschlittens, welcher in der Lage sein sollte sich herauf und herunter zu bewegen, als auch die Gabeln herein und heraus zu fahren. Deshalb war es notwendig dafür zwei Motoren zu verwenden. Anhand des Lastschlittens ergaben sich dann auch die Dimensionen der Container.

Da an den Lego NXT nur drei Motoren angeschlossen werden können, konnte nur ein Motor zur Realisierung des Antriebs verwendet werden. Deshalb muss sich dieser Motor oben befinden und die Kraft mit Hilfe von Zahnrädern übertragen werden. Dabei kommt es durch die entstehende Reibung zu einem Verlust der mechanischen Energie. Dies hätte durch das mehrmalige Umlenken der Kraft minimiert werden können, würde die Konstruktion jedoch noch größer machen.

Des Weiteren stellte die Positionierung der Sensoren ein Problem dar, da sich daraus das entsprechende Timing zum Stoppen des Roboters ergibt. Hieraus folgte die Aneinanderreihung zweier Farbsensoren, wodurch die Messungen präziser wurden. Dies führt auch dazu, dass es nur noch einen sehr kleinen Spielraum zwischen dem Erkennen der Farbe und darauffolgendem Anhalten des Portalstaplers gibt.

Außerdem war es anfänglich eher zufällig bestimmt, ob die Container gerade im Fahrzeug ankommen. Dies konnte durch den angebrachten Pflug gelöst werden, wodurch die Container gerade im Roboter stehen. Der fertige Pflug ist in Abb. 1 links am Roboter zu sehen. Dabei musste dieser so konstruiert werden, dass er die Fracht nicht aufhält, die Container also hängen bleiben könnten.

Letztlich gab es auch ein Problem beim Vorgang des Sortierens. So lief das Programm immer nur bis zum ersten zu holenden Container ab. Ursache hierfür war eine zu kurze Wartezeit zwischen den einzelnen Vorgängen. Außerdem stellte es sich als effizienter heraus, den Nutzer vor einem Sortiervorgang wählen zu lassen, wie viele Container betrachtet werden sollen.

V. Zusammenfassung und Fazit

Das Paper legt die Entwicklung eines Portalkrans mit Lego Mindstorms NXT und Matlab dar. Der beschriebene Roboter ermöglicht die Auswahl einer Containerfarbe über eine GUI und das Aufnehmen und den Transport des Containers zum Ausgangspunkt. Der Roboter orientiert sich am Vorbild eines Portalkrans. Es wurden jedoch einige Anpassungen in der Konstruktion vorgenommen. Dazu gehört vor allem die Kombination mit einem Gabelstapler, woraus der Lastschlitten des Roboters resultiert.

Es besteht jedoch die Möglichkeit den Roboter weiter zu verbessern. So würde ein vierter Motor es ermöglichen, dass auch Kurven gefahren werden könnten. Dazu wäre es jedoch notwendig einen weiteren NXT zu verwenden. Des Weiteren könnte man den Portalstapler so konstruieren, dass die Fracht auch übereinander gestapelt werden kann. Um dies zu realisieren müsste man den Roboter deutlich größer bzw. höher bauen, sodass ein Container auf einen anderen abgestellt werden kann.

Das Ziel, einen automatisierten Portalkran zu konstruieren, ist mit diesem Projekt erfolgreich gelungen. So ist es möglich zuverlässig und präzise Container anliefern zu lassen, sie zu sortieren oder die Containerfarben zu scannen.

Literaturverzeichnis

[1] PORT OF HAMBURG: Statistiken Hamburger Hafen. https://www.hafen-hamburg.de/de/statistiken. Version: 15.02.2019

[2] WIKIPEDIA, THE FREE ENCYCLOPEDIA: Portalkran. https://de.wikipedia.org/wiki/Portalkran. Version: 18.03.2019

[3] WIKIPEDIA, THE FREE ENCYCLOPEDIA: Gabelstapler. https://de.wikipedia.org/wiki/Gabelstapler#Sicherheit. Version: 18.03.2019

[4] JETSCHKE: Automatisierte Fahrzeuge. https://www.jetschke.de/de/Produkte/Automatisierte-Fahrzeuge/ . Version: 18.03.2019

Tic-Tac-Toe Roboter

Svenja Langer, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Abstract* — Im Folgenden wird sich mit dem Thema eines interaktiven Roboters befasst, der in der Lage ist, gegen einen menschlichen Mitspieler Tic-Tac-Toe zu spielen. Dabei wird auf den Aufbau und die Konstruktion eingegangen, die den Roboter befähigen, mittels eines Stiftes das Feld und seine Spielzüge zu zeichnen. Des Weiteren wird auf die Programmierung und den erstellten Spielalgorithmus eingegangen. Diese analysieren das Spielfeld und berechnen die Züge des Roboters auf Grundlage der aufgenommenen Daten. Die Aufnahme des Spielfeldes erfolgt dabei mit Hilfe einer Kamera über dem Spielfeld. Der Roboter ist in der Lage, eine vollständige Runde zu spielen, unabhängig davon, ob es einen Gewinner gibt oder das Spiel mit einem Unentschieden abgeschlossen wird.**

***Schlagwörter* — Interaktiv, LEGO Mindstorms, Roboter**

I. Einleitung

Die Bedeutung von Robotern nimmt in der modernen Zeit immer mehr zu. Sie kommen in den verschiedensten Bereichen zum Einsatz und übernehmen teilweise die Aufgaben der Menschen und sollen sie somit entlasten. So gibt es mittlerweile auch die verschiedensten Spielroboter, die es ermöglichen, dass bestimmte Spiele auch ohne zweiten menschlichen Spieler gespielt werden können. Ein Beispiel für solche Roboter sind Schachroboter. Sie ermöglichen es, die Spielregeln und Strategien zu erlernen und zu üben. So kann unabhängig von einer weiteren Person zu jeder Zeit gespielt und gelernt werden. Das Prinzip des Spielens gegen einen nicht menschlichen Gegner könnte auch auf schwierigere Spiele oder Projekte zum Zweck des selbständigen Erlernens übertragen werden.
Ein weiteres einfacheres Beispiel für einen interaktiven Roboter wäre der Tic-Tac-Toe Roboter, welcher das Ziel in der Projektwoche war.
Der Roboter sollte zwei grundlegende Aufgaben bewältigen können.
Zum einen sollte er das typische Spielfeld selber zeichnen und innerhalb des Feldes die einzelnen Kästchen gezielt ansteuern. Dazu gehörte auch, dass der Roboter in das angesteuerte Feld sein Zeichen als Markierung setzt.
Die zweite Aufgabe bestand darin, tatsächlich spielen zu können. Dazu musste der Roboter erkennen, welche Züge der menschliche Mitspieler macht und musste dementsprechend mit eigenen Spielzügen reagieren. Wichtig dabei war, dass der Roboter nicht zufällig leere Felder füllt, sondern gezielt versuchte, selber zu gewinnen. Wenn das Beenden der Spielrunde mit einem Sieg des Roboters nicht möglich war, sollte der Sieg des Gegners gezielt verhindert werden.
Der Aufbau des Roboters wurde mit LEGO-Bauteilen sowie der NXT-Station realisiert. Die Programmierung erfolgte mittels MATLAB und gewährleistete, unter Verwendung des Toolkits der RWTH Aachens, die Ansteuerung des NXT und somit auch der Motoren. Zusätzlich wurde eine Webcam zur Spielfelderfassung verwendet.

II. Vorbetrachtungen

Der geplante Tic-Tac-Toe-Roboter basiert im Grunde auf einem Schreibroboter. Er ist geplant als eine Art Schreibmaschine, die anstatt verschiedener Buchstaben, das Spielfeld und die Züge zu Papier bringt. Eine gute Vorlage und Orientierungshilfe liefert der „LEGO Printer“.
Diese Schreibmaschine muss zusätzlich mit einem Spielalgorithmus sowie einer Spielfelderkennung ausgestattet werden. Für die Erkennung des Feldes gibt es zwei Möglichkeiten, die in bisherigen Robotern verwendet wurden.

A. LEGO Printer

Die „LEGO Telegraph Machine and Printer“ von J. K. Brickworks [1] ist eine aus LEGO-Bauteilen und NXT-Einheit bestehende Schreibmaschine, die Morsesignale als Buchstabentext ausgeben soll. Dabei werden die Morsesignale über einen Taster erfasst, vom Programm ausgewertet und dann per Stift auf ein Blatt Papier gebracht. Der Roboter ist so gebaut, dass der angebrachte Stift in zwei Richtungen bewegt werden kann. Einmal kann der Stift von rechts nach links und dann noch hoch und runter bewegt werden. Die Auf- und Abbewegung wir durch das Kippen der gesamten Bewegungsvorrichtung des Stifts realisiert. Für jede der beiden Richtungen wird je ein Motor verwendet. Über einen weiteren Motor kann die Position des Blattes verändert werden. Mit dem Bewegen des eingelegten Blattes wird ermöglicht, dass tatsächlich Buchstaben statt horizontalen Linien auf dem Blatt entstehen.
Ein weiterer Sensor ist unterhalb des Roboters angebracht. Der Farbsensor soll die Erkennung der Blattränder oben und unten übernehmen.

B. Möglichkeiten der Spielfelderkennung

Einerseits gibt es die Möglichkeit mittels der Licht- bzw. Farbsensoren von LEGO das gesamte Spielfeld abzufahren und somit Farbveränderungen innerhalb eines Feldes zu registrieren und an den Roboter zur Verarbeitung weiterzugeben [2].
Die andere Möglichkeit wäre die Verwendung einer Kamera,

DOI: 10.24352/UB.OVGU-2019-070

welche das Spielfeld aufnimmt und die Veränderungen berechnen kann [3].

III. Realisierung

A. Aufbau

Der Tic-Tac-Toe-Roboter ist aus LEGO-Bauteilen und einer NXT-Einheit, welche die drei verbauten Motoren ansteuert, aufgebaut.
Wie auch bei dem Roboter von J. K. Brickworks muss der Tic-Tac-Toe-Roboter einen Stift bewegen können, daher wurde die Umsetzung für die Bewegung von links nach rechts teilweise übernommen. Sie wird mit einem der Motoren angetrieben.
Um nicht nur durchgehende Linien zeichnen zu können, sorgt ein weiterer Motor dafür, dass der Stift angehoben und gesenkt werden kann. Um dies zu ermöglichen, ist der Stift mit Gummibändern befestigt. Zusätzlichen Halt bekommt der Stift durch eine angeklebte Zahnradschiene, die in ein Zahnrad greift und das Senken und Heben des Stiftes, im Gegensatz zur ersten Variante, kontrollierbar macht. Bei dieser ist die Schiene mittels weiterer Gummis am Stift befestigt. Dadurch können die Bewegungen nicht präzise durchgeführt werden und die Schiene wird eher entlang des Stiftes verschoben anstatt ihn tatsächlich zu bewegen. Der verwendete Motor, der direkt mit dem Zahnrad verbunden ist, ist so verbaut, dass er den Links- und Rechtsbewegungen des Stiftes folgen kann. Der dritte Motor treibt die Hinterachse an und ermöglicht es dem Roboter, sich vor und zurück zu bewegen. Die Vorderräder werden durch den Schub der Hinterachse bewegt. Für genaueres Anfahren des Spielfeldes treibt der Motor die Hinterachse nicht direkt, sondern über Zahnradgetriebe mit einer 9:1-Übersetzung an. Der gesamte Roboter fährt auf Rädern, statt ursprünglich geplant auf Schienen. Die Lage der verwendeten Motoren ist in der Abbildung 1 erkennbar.

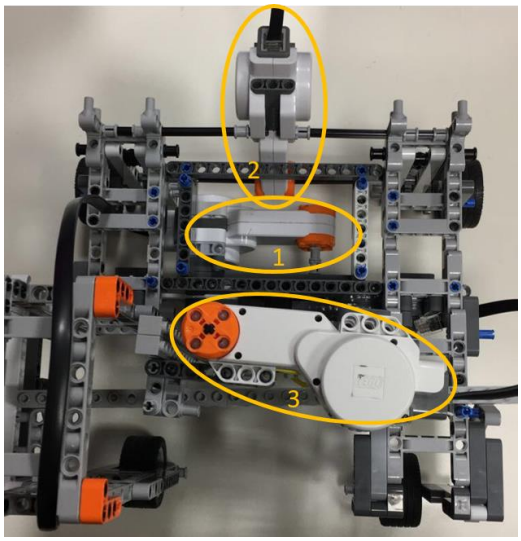


Abbildung 1: Roboter von oben
1 – Motor für links und rechts
2 – Motor für vor und zurück
3 – Motor für hoch und runter

Als Untergrund für den Roboter wird eine große LEGO-Platte verwendet. Auf der Platte ist ein Gestell befestigt, welches als Kamerahaltung fungiert. Die Kamera hat somit nach dem Einhängen immer einen konstanten Abstand zum Spielfeld und das Feld liegt immer im gleichen Bildbereich. Die Verwendung der Kamera zur Felderkennung und Auswertung wurde gewählt, weil es im Vergleich zur Variante mit dem Sensor weniger Zeit beansprucht, das Feld zu analysieren. Um dagegen mit einem Sensor das Feld zu erfassen, müsste der Roboter das Feld Kästchen für Kästchen abfahren, um die Farbunterschiede zu registrieren. Dies ist sehr zeitaufwendig und würde ein kleines Spielfeld erfordern. Damit die Ausgangsposition, von welcher der Roboter beginnt seine Befehle auszuführen, ebenfalls konstant ist, wird eine Art Stopper an die Platte gebaut. Steht der Roboter mit der Hinterachse am Stopper, so zeichnet er das Feld garantiert im Aufnahmebereich der Kamera.
Um das Zeichnen auf der unebenen Oberfläche der LEGO-Platte zu ermöglichen, empfiehlt es sich eine Pappe im Zeichenbereich unterzulegen und an der Platte zu befestigen. Die Anordnung des Roboters, sowie diverser anderer beschriebener Elemente, ist skizzenhaft der Abbildung 2 zu entnehmen.

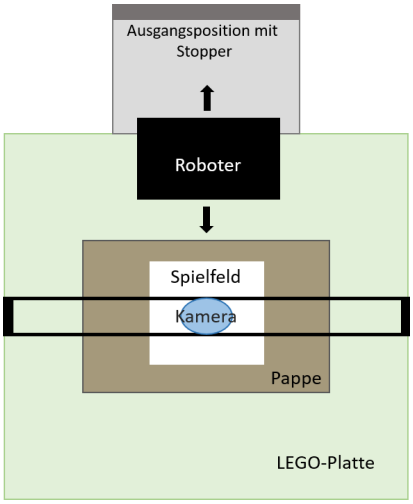


Abbildung 2: Skizze des Spielfeldes

B. Programmierung

Die Programmierung des Roboters erfolgt mit MATLAB. Der Spieler bedient den Roboter über ein Graphical User Interface kurz GUI. Die GUI enthält einen Startknopf, der das Programm startet und den Roboter aktiviert. Außerdem enthält die GUI neun Push-Buttons, die wie das Tic-Tac-Toe-Spielfeld angeordnet sind. So kann der Spielablauf überwacht werden. Es wird sichtbar, ob das vom menschlichen Spieler gesetzte Feld richtig erkannt wurde und ob der Computer das von ihm gewählte Feld auch tatsächlich richtig anfährt. Des Weiteren gibt es ein Ausgabefeld, in dem zum Ende des Spiels ausgegeben wird, ob der Spieler gewonnen oder verloren hat. Ein Beispiel, wie die GUI am Ende einer Spielrunde aussehen kann, ist die Abbildung 3.

Abbildung 3: Graphical User Interface
X – Vom Spieler gesetztes Feld
O – Vom Roboter gesetztes Feld

Bevor das Programm gestartet wird, muss der Roboter in die Ausgangsposition gestellt werden, da die Anfahrtswege zum Spielfeld sowie zu den einzelnen Kästchen über den Befehl „Tacho Limit“ geregelt sind. Damit sind die Anfahrtswege immer identisch. Dabei muss beachtet werden, dass die Tacho-Limits der verschiedenen Motoren durch die Übersetzung des Motors für die Hinterachse nicht gleich sind. Horizontale und vertikale Linien sind einfach zu programmieren, da jeweils nur ein Motor angetrieben wird. Für die diagonalen Linien, die im Kreuz, welches der Roboter setzt, enthalten sind, müssen jedoch zwei Motoren gleichzeitig angesteuert werden. Auch hier muss beachtet werden, dass durch die Übersetzung verschiedene Werte genutzt werden müssen, um ein Kreuz zu erhalten. Nach dem Drücken des Startknopfes werden nacheinander die Motoren angesteuert, so dass der Roboter zuerst zum Spielfeld fährt und dieses dann zeichnet. Im Anschluss kehrt er in die Ausgangsposition zurück, welche sich außerhalb des Kamerabildes befindet. Sobald die Motoren gestoppt sind, wird das Spielfeld von der Kamera erfasst. Die einzelnen Kästchen des Spielfeldes sind fest im aufgenommenen Bild definiert. Daher ist die exakte Einhaltung der Kameraposition wichtig. Ansonsten würden die Felder verschoben oder nicht erkannt werden. Sobald das Bild aufgenommen ist, hat der Spieler zehn Sekunden Zeit seinen Zug zu machen. Danach wird ein zweites Bild aufgenommen. Die beiden Bilder werden in Grautöne umgewandelt und miteinander verglichen. Das Feld, in welchem die größte Veränderung vorliegt, wird als gesetzter Zug des menschlichen Spielers interpretiert. Damit dies zuverlässig funktioniert, dürfen sich die Lichtverhältnisse innerhalb der zehn Sekunden nicht verändern. Des Weiteren wird das gewählte Kästchen besser erkannt, je mehr des Kästchens ausgemalt ist. Es empfiehlt sich daher für den Spieler auf die typischen Zeichen (Kreuz bzw. Kreis) zu verzichten, um Fehlinterpretationen seitens des Programmes zu vermeiden. Im Gegensatz dazu kann der Roboter ohne Probleme das Kreuz beziehungsweise den Kreis verwenden, da seine Spielzüge nicht mit der Kamera erfasst werden, sondern vom Spielalgorithmus vorgegeben und abgespeichert werden.
Außerdem sollte darauf geachtet werden, dass der Spieler am Ende seines Zuges sowohl den Stift als auch seine Hand aus dem Bereich des Kamerabildes nimmt. Das als vom Spieler gesetzt erkannte Feld wird in der GUI gekennzeichnet. Der Push-Button, welcher dem gesetzten Kästchen entspricht, wird mit einem „x“ versehen. Im Anschluss ermittelt der Spielalgorithmus, an welche Stelle der Roboter seinen Zug macht. Der entsprechende Push-Button wird dann auf „o“ gesetzt. Die Feldposition wird an den Roboter weitergegeben und dieser fährt das jeweilige Kästchen an. Jedes Kästchen des Feldes hat eine zugewiesene Nummer und auf diese Nummer jeweils einen eigenen programmierten Anfahrtsweg. Hat der Roboter sein Kästchen erreicht, so setzt er ein Kreuz in dieses. Bei der Programmierung wurde darauf geachtet, dass das Kreuz an der gleichen Ecke des Kästchens beginnt und endet, damit bei der folgenden Rückkehr in die Ausgangsposition keine zusätzlichen Verschiebungen entstehen.
Beachtet werden muss, dass in der GUI der Roboter den Kreis und der Spieler das Kreuz hat. Da jedoch das Programmieren des Ablaufes für ein Kreuz wesentlich unkomplizierter ist, setzt der Roboter auf dem tatsächlichen Spielfeld das Kreuz und der Spieler malt am besten dass gesamte Kästchen aus. Nach der Rückkehr in die Ausgangsposition beginnt der Ablauf erneut ab dem Punkt, wo das erste Kamerabild aufgenommen wird. Abgebrochen wird dann, wenn die Schleife viermal wiederholt wurde. Das liegt daran, dass das Feld aus neun Kästchen besteht und somit maximal neunmal gesetzt werden kann. Bei zwei Spielern, die immer abwechselnd setzten, sind das fünf Züge für denjenigen der anfängt und vier Züge für den zweiten Spieler. Da immer der menschliche Spieler beginnt, kann der Roboter bei einem Unentschieden nicht mehr als viermal setzen und somit muss die Schleife viermal durchlaufen werden. Der gesamte Ablauf ist in der Abbildung 4 noch einmal schematisch dargestellt.

Für den Spielalgorithmus werden den gesetzten Kästchen Werte zugeordnet. Die Züge des Spielers erhalten den Wert „1“ und die Züge des Roboters den Wert „4“. Diese Werte werden in einer 3x3-Matrix abgespeichert. Die Matrix ist zum Beginn einer Spielrunde mit Nullen gefüllt. Nach jedem Spielzug, in dessen Folge sich die Werte ja ändern, werden jeweils die Spalten, Zeilen und Diagonalen auf ein Neues zusammengerechnet. Zuerst werden die erhaltenen Ergebnisse überprüft, ob sie den Wert acht annehmen. Ist dies der Fall, bedeutet das, dass der Roboter in einer Zeile, Spalte oder Diagonalen bereits zweimal gesetzt hat und mit seinem Zug nun gewinnen kann. Also wird die Nummer des leeren Kästchens an den Roboter weitergegeben. Ist dies nicht der Fall, wird als nächstes geprüft, ob einer der zusammengerechneten Werte zwei ist. Das würde bedeuten, dass der Spieler zwei gleiche Zeichen hat und gewinnen könnte. Dann würde ebenfalls die Position des dritten freien Kästchens weitergegeben werden, um den Sieg des menschlichen Spielers zu verhindern. Sollte beim Rechnen jedoch eine andere Zahl als zwei oder acht herauskommen, so setzt der Roboter in ein noch nicht belegtes Feld. Der Spielalgorithmus, das Zeichnen des Feldes, da Setzen des Kreuzes sowie das Aufnehmen und Verarbeiten des Kamerabildes bilden jeweils eigene Funktionen, die letztendlich zu einer Funktion zusammengefasst werden.

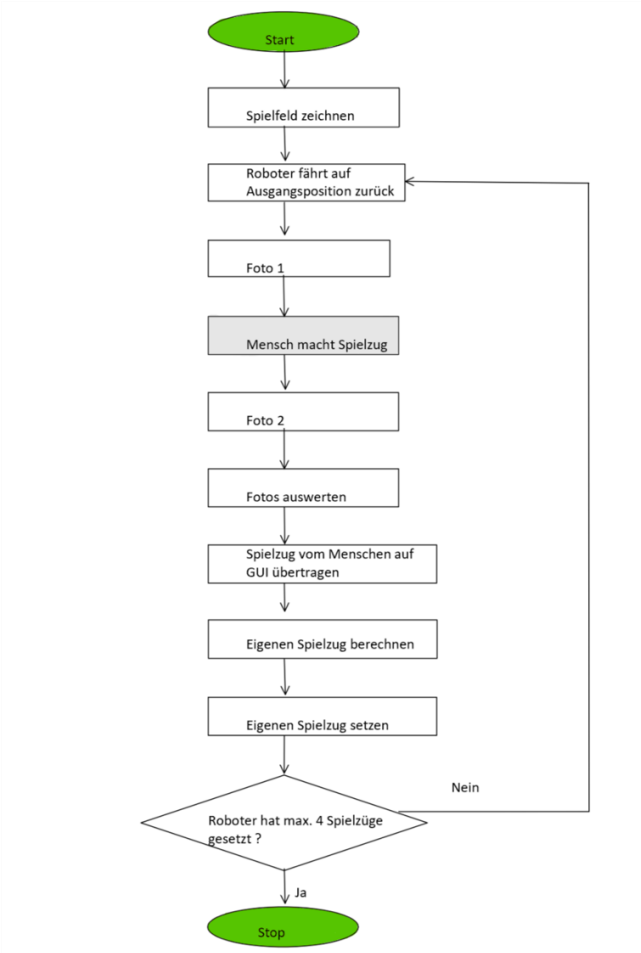


Abbildung 4: Programmablaufplan

C. Ergebnisse

Das Ergebnis des Projektes ist ein Roboter, der in der Lage ist, selbstständig mittels eines Stiftes ein Spielfeld zu zeichnen, die einzelnen Kästchen gezielt anzusteuern und sinnvolle Spielzüge zu machen. Diese können, wie auch der restliche Spielverlauf, mittels der erstellten GUI kontrolliert und nachvollzogen werden können. Die Kamera ermöglicht es dem Roboter mit dem Spieler zu interagieren, da sie dessen Züge erfasst.

Eine Spielrunde mit dem Ergebnis „unentschieden“ dauert etwa sechs Minuten. Gibt es bei einer Runde einen Gewinner, so werden insgesamt weniger Züge gemacht und somit verkürzt sich die Spielzeit.

IV. Fazit und Ausblicke

Das gewünschte Ziel des Projektes, einen funktionierenden Tic-Tac-Toe-Roboter zu bauen und zu programmieren, ist gelungen.

Die anfangs angestrebte Interaktion mit dem menschlichen Spieler funktioniert so, wie es geplant war. Sie kann aber noch verfeinert werden, indem sowohl Spieler als auch Roboter dem jeweils anderen signalisieren, wann ihr Zug beendet ist. Die festgelegte Reaktionszeit von zehn Sekunden des Spielers kann durch einen Taster ersetzt werden. Somit hat der Spieler die Möglichkeit genau zu überdenken, welchen Zug er machen möchte. Auch der Roboter kann mittels eines Signaltones nach dem ersten Kamerabild dem Spieler zeigen, dass er fertig ist und der Spieler am Zug ist. Diese Zusätze können das Spiel flüssiger machen und die Spielzeit somit verkürzen.

Die Erfassung und Auswertung des Spielfeldes mittels Kamera erfolgt zuverlässig, solange genug Fläche des Kästchens ausgefüllt wird und die Kamera ihre Position exakt beibehält. Um das Problem mit der Kameraposition zu vermeiden, kann die Programmierung der Kamera so geändert werden, dass sie sich jedes Mal neu auf das jeweilige Spielfeld kalibriert.

Um die Spieloptionen zu erweitern, kann der Spielalgorithmus erweitert werden, so dass nicht nur der Spieler, sondern auch der Roboter das Spiel beginnen kann.

Zusammenfassend kann das Projekt als gelungenes Grundkonzept angesehen werden, welches mit kleinen Änderungen noch optimiert werden kann.

V. Literaturverzeichnis

[1] J. K. Brickworks: LEGO Telegraph Machine and Printer. https://jkbrickworks.com/telegraph-machine-and-printer. Version: März 2019

[2] Michalis Koulouras: Beispiel für Sensorerkennung. https://www.youtube.com/watch?v=x1Q8h7qegjk. Version: März 2019

[3] Dave Corboy: Beispiel für Kameraerkennung. https://www.youtube.com/watch?v=IfKaSQDMjYM. Version: März 2019

Tic-Tac-Toe-Roboter

Klara Uhlig, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Abstract*—Dieses Paper befasst sich mit der Entwicklung eines selbstständig spielenden Tic-Tac-Toe-Roboters. Er ist in der Lage, sein Spielfeld zu zeichnen, die Spielzüge seines Gegenspielers mithilfe einer Webcam zu erfassen und daraufhin über einem Algorithmus über seinen nächsten Spielzug zu entscheiden und sein Kreuz auf dem Spielfeld zu setzen. Er soll, wenn möglich, selbst gewinnen oder das Gewinnen des Gegners verhindern. Der Roboter besteht aus Lego-Mindstorms-Bauteilen, die Programmierung erfolgt über MatLab.**

***Schlagwörter*—Roboter, Lego-Mindstorms, Projektseminar, Tic-Tac-Toe.**

I. Einleitung

Das Ziel dieses Projektes war es, eine eigene Idee für einen Roboter zu finden und in kleinen Gruppen von zwei bis drei Personen umzusetzen. Zur Verfügung standen hierfür Lego-Mindstorms-Bauteile mit dem Steuerungscomputer NXT, sowie verschiedene Sensoren und eine Webcam. Zur Programmierung wurde zusätzlich MatLab, eine Software zum Lösen von mathematischen Problemen, Analysieren von Daten und entwickeln von Algorithmen verwendet. Es besteht außerdem die Möglichkeit graphische Benutzeroberflächen zu erstellen und zu gestalten, die für dieses Projekt auch genutzt wurde. Die Entscheidung fiel auf einen Tic-Tac-Toe spielenden Roboter, da die Komplexität des Strategiespieles im Vergleich zu anderen Spielen wie Schach oder Dame nicht zu hoch ist und im Rahmen des einwöchigen Projektes in den grundlegenden Funktionen umsetzbar ist. Anders als bei vergleichbaren Spielen am Computer wird mit Stift und Papier gespielt und nicht nur auf einer graphischen Benutzeroberfläche, was für ein realistischeres Spielgefühl sorgt.

II. Vorbetrachtungen

A. Spielregeln Tic-Tac-Toe

Tic-Tac-Toe ist ein klassisches Strategiespiel, das zwei Spieler gegeneinander spielen. Zuerst wird ein quadratisches 3 mal 3 Spielfeld aufgemalt, es können also 9 Kästchen belegt werden. Es wird abwechselnd gesetzt, üblicherweise setzt ein Spieler ein Kreuz “X” und der andere Spieler ein Kreis “O” in ein freies Feld. Ziel ist es nun, 3 seiner Kreuze oder Kreise in einer Reihe, Spalte oder Diagonalen zu setzen. Wem dies zuerst gelingt, hat gewonnen. Es darf immer nur ein Feld pro Spielzug besetzt werden.

Wie in Tabelle 1 zu sehen ist, ist die Wahrscheinlichkeit, dass der Spieler, der beginnt, auch gewinnt, deutlich höher als bei den andern beiden Spielverläufen.

TABELLE I
SPIELSTATISTIK

	Anzahl	Anteil
Anzahl aller Spielverläufe	255 168	100 %
Enden mit Sieg des ersten Spielers	131 184	51,41 %
Enden mit Sieg des zweiten Spielers	77 904	30,53 %
Enden Unentschieden	46 080	18,06 %

Statistik zum Sieg ohne Berücksichtigung der gleichen Spielzüge durch Drehung oder Spiegelung

B. Ähnliche Roboter

In der Vorbetrachtung wurden zunächst auch noch andere Roboter mit ähnlichen Funktionsweisen betrachtet. Grundlegend muss sich der Roboter nämlich in allen drei Dimensionen bewegen können, also vor- und zurückfahren, den Stift nach links und rechts fahren, sowie den Stift heben und absenken. Lego-Roboter, die als Drucker mit Stift und Papier arbeiten, sowie 3D-Stanzer, die Bilder in Blumenschaum oder ähnliches stanzen, wiesen den gewünschten Aufbau und eine ähnliche Funktionsweise auf. Hierbei wurden häufig Konstruktionen mit Schienen, auf denen Zahnräder laufen, verwendet, was aufgrund der mangelnden Bauteile aber nicht in Frage kam. Dennoch war die Betrachtung dieser Roboter hilfreich, da diese meist eine Übersetzung der Motordrehung über verschiedene Zahnräder aufwiesen, um die Genauigkeit der Motoren zu verbessern.

III. Umsetzung des Roboters

Die Idee war es, einen Tic-Tac-Toe spielenden Roboter zu entwickeln, der einen menschlichen Spielpartner ersetzen kann. Hierzu musste nicht nur ein Spielalgorithmus entwickelt werden, sondern auch das Ansteuern der einzelnen Felder mit einem selbst gebauten Roboter programmiert werden.

DOI: 10.24352/UB.OVGU-2019-071

A. Der mechanische Aufbau

Der Roboter besteht aus 3 Motoren. Der erste Motor treibt die Reifen an, die zum Vor- und Zurückfahren da sind. Er befindet sich ganz hinten am Roboter. Der zweite Motor ist für die Bewegung nach links und rechts verantwortlich und sitzt unter dem NXT. Der letzte Motor zum Absetzten und Hochfahren des Stiftes ist über dem NXT verbaut.

Abbildung 1: Beispiel-GUI

Die Kamera hängt in einer festen Position über dem Spielfeld. Für jedes der 9 Felder sind feste Koordinaten festgelegt. Vor dem Spielzug des Menschen wird ein Bild gemacht und der gerundete Grauwert der einzelnen Felder ermittelt. Nach einer 10-sekündigen Pause wird ein zweites Bild gemacht und die neuen Grauwerte der Spielfelder ermittelt. Die beiden Werte werden miteinander verglichen und das mit der größten Abweichung erkannt. Danach wird der Spielzug des Menschen in die GUI (Graphic User Interface) als “X” eingetragen. Da eine Erkennung eines Kreuzes auf dem Papier aufgrund der Qualität der Kamera schwierig ist, wird vom Menschen kein Kreuz gesetzt, sondern das Spielfeld ausgemalt um die Änderung des Grauwertes zu erhöhen, wie auch in Abbildung 3 zu sehen ist.

B. Programmierung

Am Anfang des Programms wird programmintern eine 3x3-Matrix M mit Nullen erstellt. Für jeden Spielzug des Menschen wird eine 1 in das jeweilige Feld der Matrix eingetragen, für jeden Spielzug des Computers eine 4. Die Werte wurden so gewählt, damit die später berechneten Summen der Zeilen, Spalten und Diagonalen eindeutig sind. Für die Situation aus Abbildung 1 ergibt sich also die folgende Matrix:

$$M = \begin{bmatrix} 0 & 0 & 1 \\ 4 & 4 & 4 \\ 1 & 0 & 1 \end{bmatrix} \quad (1)$$

Um seinen eigenen Zug zu setzen, werden die Zeilen, Spalten und Diagonalen der Matrix summiert. Danach wird geprüft, ob das Gewinnen des Roboters möglich ist, also ob die Summe in einer Zeile, Spalte oder Diagonalen gleich 8 ist. Wenn dies der Fall ist, wird das verbleibende Feld besetzt (siehe Anhang). Hierfür wird geprüft, welches der Felder nicht besetzt ist,

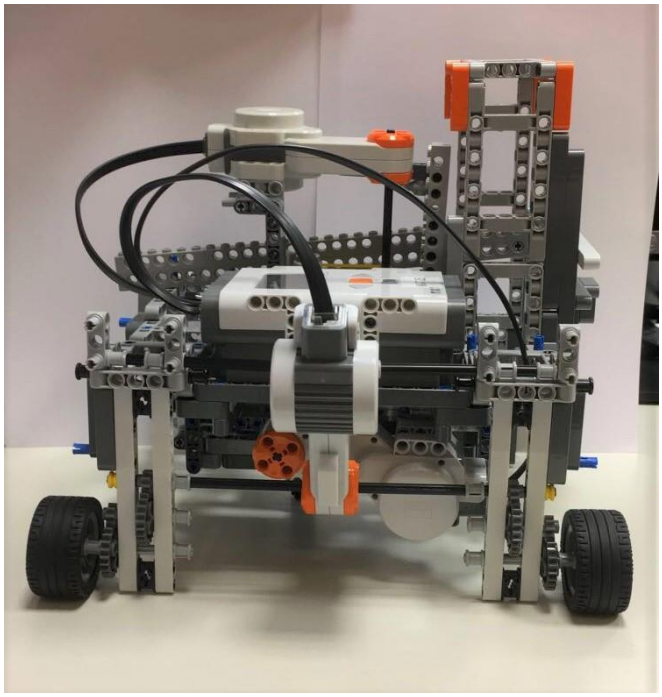


Abbildung 2: Roboter von hinten

also welches Feld noch keinen weiteren Text wie “X” oder “O” hat. Außerdem wird mit der Variable “Spielzug” dafür gesorgt, dass nur ein Spielzug gemacht wird, da als Bedingung für alle weiteren Züge gelten muss, dass diese gleich Null ist. Wird ein Spielzug gemacht, wird diese um eins erhöht. Auf der GUI wird dann ein “O” gesetzt. Außerdem wird geprüft, ob der Mensch gewinnen kann, also ob die Summe in einer Spalte, Zeile oder Diagonalen 2 ist. In diesem Fall wird auch das verbleibende Feld besetzt. Sollte das Gewinnen beider Parteien nicht möglich sein, wird ein zufälliges freies Feld besetzt. Nachdem der Spielzug auf der GUI angezeigt wird, fährt der Roboter zu dem jeweiligen Feld hin.

C. Anfahren der Spielfelder

Um ein Spielfeld anzufahren, müssen alle 3 Motoren richtig programmiert sein. Zunächst muss der Roboter aus seiner Ausgangsposition zum Spielfeld hinfahren. Hierzu wurden die Felder von 1 bis 9 durchnummeriert. Das Feld oben links ist dann zum Beispiel das Feld 1 und das Feld unten rechts das Feld Nummer 9. In der Ausgangsposition ist der Stift ganz rechts, also bei Feld 3. Um nun zum Beispiel zu Feld 7 unten links in der Ecke zu gelangen, muss der Roboter also noch zwei weitere

Kästchen nach vorne und der Stift zwei Kästchen nach links bewegt werden. Dort wird dann die Funktion zum zeichnen der Kreuze aufgerufen, in der die Motoren zum Vorwärts- und Seitwärtsfahren gleichzeitig angesteuert werden. Um die Motoren und den NXT überhaupt über MatLab nutzen zu können, wurde ein zusätzlicher Ordner mit entsprechenden Befehlen und Funktionen bereitgestellt. Mit diesen Funktionen kann die Geschwindigkeit der Motoren, sowie ein Tacho Limit eingestellt werden, welches den Motor nach einer gewissen Zeit stoppt. Jeder Befehl muss anschließend an den NXT zurückgesendet werden. Es besteht außerdem die Möglichkeit, auf einen anderen Motor zu warten, was zum Beispiel für das Zeichnen des Feldes genutzt wurde.

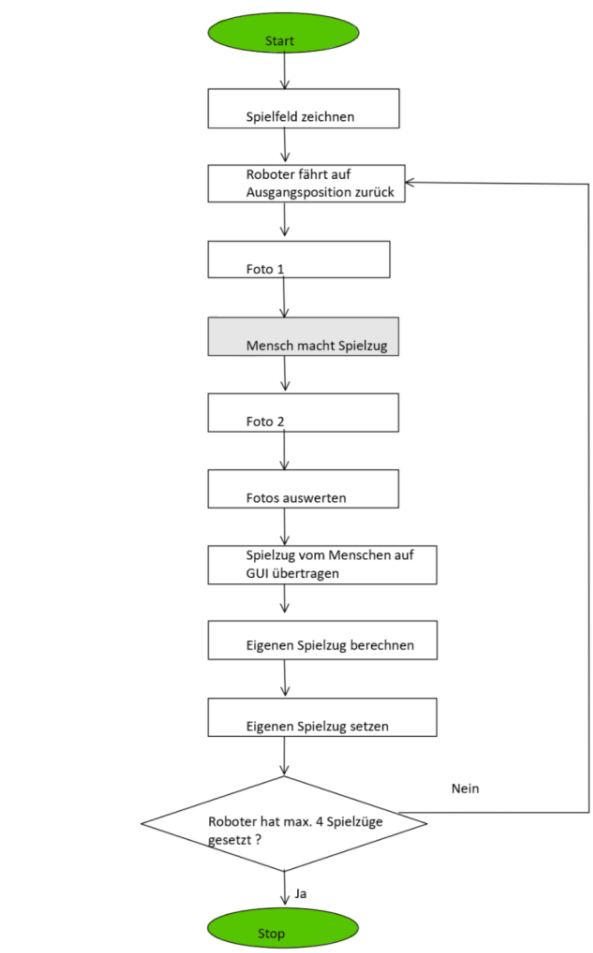


Abbildung 3: Programmablaufplan

IV. Ergebnisdiskussion

Das größte Problem des Roboters sind die leistungsschwachen Motoren von Lego. Durch einen deutlichen Unterschied bei voll geladener oder halb geladener NXT Station ist es schwierig, den Roboter genau zu kalibrieren. Des weiteren bricht der NXT das Programm an einigen Stellen ab, wenn die Anzahl der Motorenansteuerungen hintereinander zu hoch ist. Auch durch kurze Pausen konnte das Problem nicht gelöst werden. Das führt zum Beispiel dazu, dass der Roboter nicht auf seine Ausgangsposition zurückfährt und in den folgenden Zügen nicht mehr die richtigen Felder anfahren kann. Ein weiteres Problem ist die derzeit feste Kalibrierung der Kamera. Wird die Kamera um ein paar Zentimeter falsch ausgerichtet, so misst diese falsche Werte bzw. die Spielfelder stimmen nicht mehr überein. Dies könnte man durch eine automatische Spielfelderkennung lösen, die dann allerdings wieder das Problem hätte, dass bei Versagen der Motoren ein "falsches" Feld eingescannt wird, da der Roboter zu fest vorgegeben Feldern hinfährt. Auch die Größe der Kreuze könnte sich nicht der eventuell ändernden Größe des Spielfeldes anpassen. Bei kleinerem Spielfeld wären die Kreuze dann zum Beispiel zu groß für die Felder.

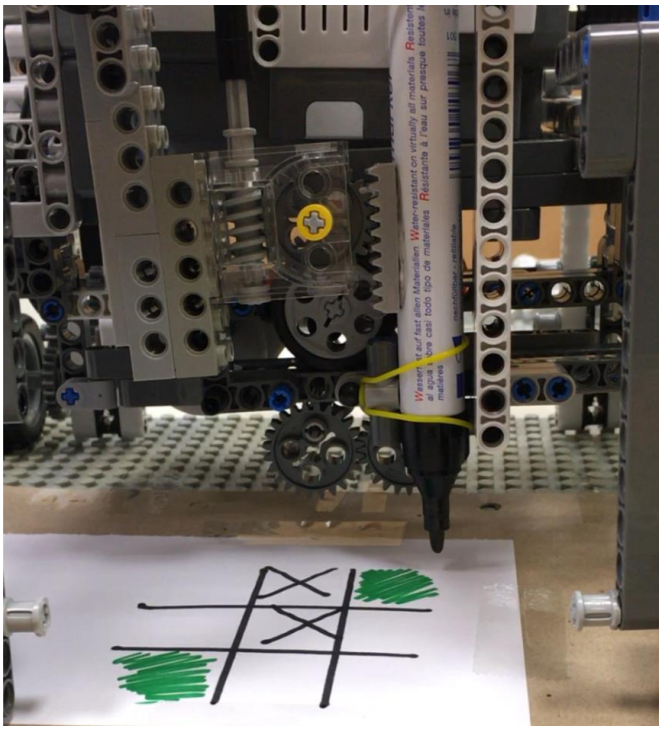


Abbildung 4: Front des Roboters mit Halterung des Stiftes und selbstgezeichnetem Spielfeld mit Zügen

Auch bei der Mechanik gibt es noch Probleme. So ist die eine Seite des Roboters schwerer als die andere, was dazu führt, dass der Stift an der einen Seite deutlich mehr aufdrückt als auf der anderen Seite. Dies kann zum Verhaken des Motors führen und der Stift fährt nicht an seine Ausgangsposition zurück. Grund dafür ist der Motor zum Absenken und Hochfahren des Stiftes. Dieser muss nämlich mit dem Stift nach links und rechts fahren. Hier wäre zum Beispiel ein Schienensystem oder ein weiteres Rad auf dem Roboter eine Lösung, damit das Gewicht nicht nur auf der Achse für die Querbewegung lastet.
Ein weiteres Problem ist das Erkennen des Spielendes, wenn ein Spieler gewonnen hat. Derzeit ist der Algorithmus so konzipiert, dass er nach 4 Spielzügen, also der maximalen Anzahl an Zügen, die der Roboter machen kann, das Programm

stoppt. Das führt auch dazu, dass nachdem der Mensch gewonnen hat, der Computer noch einen Zug errechnet und dann eventuell ausgibt, dass er selbst gewonnen hat, obwohl dies nicht der Fall ist.
Einige Probleme konnten im Verlauf des Projektes aber auch erfolgreich gelöst werden. Dazu zählt zum Beispiel, dass der Computer nur einen Zug setzen kann und nicht mehrere, wenn er gewinnen und der Gegenspieler im nächsten Zug auch gewinnen könnte. Auch das zunächst geplante Schienensystem, um vor- und zurückzufahren, konnte aufgrund von einer Bauteilknappheit gut durch Räder ersetzt werden. Auch die Halterung des Stiftes sorgte für Probleme. Die zuerst verwendete Halterung mit Gummibändern erwies sich als zu flexibel, denn bei jedem Hoch- und Runterfahren des Stiftes hatten die Gummibänder einen nicht vorhersehbaren Spielraum. Das Problem wurde gelöst, indem eine Zahnradschiene an den Stift festgeklebt wurde, wie in Abbildung 4 zu sehen ist.
Weitere Verbesserungsmöglichkeiten wären zum Beispiel eine Auswahl auf der GUI, ob der Mensch oder Computer anfangen soll oder die Auswahl zwischen verschiedenen Schwierigkeitsstufen. So könnten zum Beispiel auch die Spieltaktiken des Menschen ausgewertet und gezielt verhindert werden. Dazu müssten dann keine zufälligen Felder sondern gezielt die Felder belegt werden, die man zur Erstellung von Zwickmühlen braucht.

V. Zusammenfassung und Fazit

Zusammenfassend ist das Projekt ein Erfolg, da ein Roboter entwickelt wurde, der mit einem Menschen Tic-Tac-Toe spielen kann. Die Mechanik weist noch einige Probleme auf, die teilweise durch eine bessere Konstruktion gelöst werden könnten. Das größere Problem sind aber die relativ leistungsschwachen Motoren, die Lego-Mindstorms bereitstellt. Das größere Problem ist der NXT-Steuerungsroboter, da er nur eine relativ geringe Anzahl an Motoransteuerungen nacheinander ausführen kann. Werden zu viele Motoren nacheinander angesteuert, kommt es zum kurzzeitigen Abbruch des Programms und einige Motoren werden nicht angesteuert. Das führt dazu, dass der Roboter nicht mehr in seine Ausgangsposition zurückfährt und nicht mehr die richtigen Felder anfahren kann. Das Programm weist noch einige Probleme auf, zum Beispiel, dass es nicht automatisch abbricht, wenn der Mensch oder Roboter gewonnen hat. Weitere kleinere Probleme, wie zum Beispiel die Halterung des Stiftes oder, dass der Roboter maximal ein Zeichen pro Spielzug macht, konnten im Verlauf des Projektes behoben werden. Derzeitig kann der Roboter also mit einem menschlichen Spielpartner Tic-Tac-Toe auf einem Blatt Papier spielen. Der Roboter kann nach dem Start des Programmes das Spielfeld selbstständig zeichnen und die Züge des Menschen, der derzeit immer beginnt, mithilfe einer Kamera erkennen und darauf reagieren. Die Spielzüge vom Roboter und vom Menschen werden hierbei auch auf der GUI angezeigt, ebenso wie die Anzeige, ob der Mensch gewonnen oder verloren hat.

Anhang

A. *Quellcode, um letztes Feld zu belegen*

```
% Beispielhaft für die erste Zeile:
   Spielzug = 0;
  if ( SummeErsteZeile == 8 && spielzug ==0)
      if(get(handles.pushbutton1, 'String') ==' ')
          set(handles.pushbutton1, 'String', 'O');
          handles.matrix(1,1) = 4;
           Spielzug = 1;
           handles.feld = 1;
      end
      if(get(handles.pushbutton2, 'String') ==' ')
          set(handles.pushbutton2, 'String', 'O');
          handles.matrix(1,2) = 4;
         Spielzug = 1;
         handles.feld = 2;
      end
      if(get(handles.pushbutton3, 'String') ==' ')
          set(handles.pushbutton3, 'String', 'O');
          handles.matrix(1,3) = 4;
          Spielzug = 1;
          handles.feld = 3;
      end
  end
```

Literaturverzeichnis

[1-4] Eigene Aufnahmen

[5] Norman Do, How to Win at TicTacToe, The Australian Mathematical Society, Gazette, Volume 32 Number 3, July 2005, S. 151

Roboter zur Müllsortierung

Gabriel Giese, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Aufgrund des weltweit steigenden Müllaufkommens ist es nötig, neue Wege in der Wiederverwertung von Wertstoffen einzuschlagen. Jedoch ist die Aufteilung des Abfalls in ihre verwertbaren Bestandteile aufwändig und schwierig. In diesem Paper wird detailliert die Konstruktion eines Roboters mit drei Gelenken beschrieben, der durch einen Sensor Müll der Farbe nach sortiert. Der Roboter weist drei Motoren auf, welche den Arm ausrichten und den Müll greifen lassen. Er ist in drei Bauteile aufgeteilt, deren Bauweise separat behandelt wird. Weiterhin ist die Programmierung der Software, die den Roboter ansteuert, in ihrer Abfolge erklärt. Nach Konstruktion und Programmierung ist der Roboter fähig, die Gefäße, in denen der Müll gelagert werden soll, der Farbe nach zu erkennen, den Müll zu greifen, die Farbe zu erfassen und dann in den dafür vorgesehenen Behälter zu befördern.**

***Schlagwörter*—Farbe, Müll, Recycling, Roboter, Rotation, Sortierung**

I. Einleitung

RECYCLING von Abfällen ist ein immer wieder auftretendes Thema. Nicht nur durch das Vermeiden von Verschmutzungen der Umwelt, sondern auch als Aspekt der Wiedergewinnung von Rohstoffen. In der Industrie werden verwertbare Reststoffe als Gut betrachtet und in anderen Industriezweigen wiederverwertet. Jedoch sind gerade im Müll von Privathaushalten viele Ressourcen unzugänglich gemacht worden, weil es nicht zu einer Trennung kam. Die Bestimmung einzelner Materialien ist aufwendig und ohne Hilfsmittel praktisch unmöglich. Daher werden in der Recyclingindustrie ausschließlich Roboter und automatisierte Maschinen verwendet. Jedoch sind die Standorte dafür in konsumreichen Ländern, wie Deutschland, überlastet. Daher ist es notwendig, die Sortierung einfach zu gestalten. Eine Möglichkeit wäre ein Roboter, der das Material eines gegriffenes Gegenstandes erkennt und nach den spezifischen Eigenschaften diesen sortiert.

II. Vorbetrachtungen

Mechanische Roboter werden schon seit Jahrzehnten verwendet, um menschliche Arbeit zu erleichtern oder zu ersetzen. Dafür war es nötig, die elektrische Energie in eine Rotationsbewegung in Motoren umzuwandeln. Dadurch konnte eine Rotation erreicht werden. Um nun eine Beweglichkeit in Ähnlichkeit zum menschlichen Arm zu erreichen, setzte man mehrere Motoren hintereinander und immitierte so die Gelenke. Dadurch ist es einem solchen Roboter möglich, nahezu jeden beliebigen Punkt um ihn herum im Radius des voll ausgestreckten Arms zu erreichen. Solche Roboter findet man vor allem an Fließbandarbeit, wobei die Roboter immer die gleichen Schritte ausführen (Beispiel Automobilindustrie). Diese Roboter sind aber noch nicht fähig, sich selbstständig in anderen Situationen angepasst zu verhalten. Sie werden trotz aller Umstände stets die selben Schritte ausführen.

DOI: 10.24352/UB.OVGU-2019-072

III. Konstruktion des Roboters

Während des zweiwöchigen Praktikums ist es gelungen, einen Roboter aus Lego zu bauen, der selbstständig Farben von Bällen erkennen konnte und diese in Körbe einsortieren konnte. Im Vorfeld wurden dafür farbliche Markierungen an den Körben angebracht, die auch vom Roboter erfasst werden konnte, wodurch eine Umpositionierung der Körbe vor der Initialisierung möglich war. Baulich wurde dies durch drei Segmente des Roboters umgesetzt: Die Basis, der Arm und die Kralle.

A. Die Basis

Die Basis wurde flach gehalten und wurde an allen vier Kanten mit gummierten Ketten ausgestattet, um die Reibung zu erhöhen und damit das Verrutschen des Roboters zu verhindern. In der Basis befindet sich der erste Motor, der für die Rotation des darauf befindlichen Arms zuständig ist. Um den Kraftaufwand niedrig zu halten, wurden Zahnräder verwendet, die mit einer Übersetzung die nötige Kraft verringerten, aber dafür die Anzahl der Umdrehungen des Motors erhöhten. Durch die Installation des Motors in der Basis erhöhte sich auch die Stabilität, da eine Positionierung innerhalb des Armes nur zu einer Mehrbelastung der Drehachse führen würde. Der darauf befindliche Arm wurde mit Hilfe einer Kreuz-Achse befestigt. Dies bot aber während der ersten Tests der Drehbewegung erhebliche Schwierigkeiten, da die Masse des Arms die Achse verbog und somit den Arm auf der Basis aufsetzen ließ, was zum Verkanten geführt hat. Um dem Aufsetzen entgegenzuwirken, wurde auf der Unterseite des Arms eine Scheibe befestigt, die auf losen Kreuz-Achsen auf der Basis dauerhaft aufsetzte, um so die Wirkung eines Kugellagers zu immitieren. Dies hat sehr gut funktioniert und die Stabilität des Roboters für den Rest der Versuchsreihen erhöht. Dadurch, dass die Scheibe lose auf der Basis aufliegt, ist es möglich, den Arm durch einfaches Ziehen von der Basis zu trennen, wodurch der Aufbau sehr kompakt zu verstauen ist und dadurch sehr mobil war.

B. Der Arm

Der Arm ist das verbindende Stück zwischen Basis und Klaue. Er sorgt mit dem eingebauten Motor für das Absenken und Anheben der Klaue und sorgt gleichzeitig mit seiner Länge für einen gewissen Operationsradius zwischen der Basis, den zu greifenden Bällen und den zu befüllenden Körben.

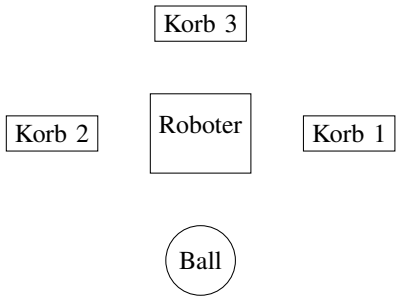


Abbildung 1. Testaufbau mit 3 Körben

Für das Heben und Senken der Klaue wurde ein weiterer Motor 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. Um ein unnötiges Ungleichgewicht und damit eine Überbelastung der Rotationsachse an der Basis zu vermeiden, wurde der Arm ausbalanciert, indem der NXT-Stein, der Controller, auf der entgegengesetzten Seite der Klaue montiert wurde. Sein Gewicht reicht, um der Hebelwirkung des Armes entgegenzuwirken und die Scheibe der Drehachse gleichmäßig aufliegen zu lassen.

C. Die Klaue

Die Klaue wurde so am Arm befestigt, dass sie immer waagerecht zum Boden steht. Dies wurde erreicht, indem sich ihr Befestigungspunkt direkt über ihrem Schwerpunkt befindet. Die Klaue besitzt ebenfalls einen Motor, sowie einen Farbsensor. Der Motor betreibt eine zu ihm in 90 Grad gerichtete Achse, welche eine Schraube führt. Diese Schraube greift in die zwei Zahnräder der Gabeln, die dann eine entgegengesetzte Drehung vollziehen und sich damit zueinander oder voneinander bewegen. Dadurch wird das öffnen und Schließen der Kralle realisiert. Der Farbsensor befindet sich mittig über den Gabeln, um sich über dem gegriffenem Ball zu befinden. Aufgrund der technischen Realisierung des Sensors, muss er sich etwa 2 Zentimeter über dem Ball befinden, um die Farbe zu erfassen.

D. Das Programm

Zur Programmierung wurde das Programm MatLab in der Version 2018a verwendet. Dazu wurde eine Toolbox der Universität Aachen zur Verfügung gestellt, welche grundlegende Befehle für die Ansteuerung des NXT-Steins beinhaltete. Wie man in Abbildung 2 verfolgen kann, beginnt der Prozess mit der Initialisierung und damit mit dem Ansprechen der Software des NXT-Steins. Bei der Korbfarbenerkennung fährt der Roboter mit einer Rotation des Arms und Absenken der Klaue mit dem Sensor über alle Körbe und nimmt dabei die Daten der farblichen Markierungen auf und speichert sie als Wort (Bsp. 'BLUE'). In den Testaufbauten wurden vorwiegend 3 Körbe benutzt, welche sich in einer Anordnung wie in Abbildung 1 befanden. Die Körbe und die Annahmestelle der Bälle befanden sich im selben Abstand mit jeweils 90-Grad-Versetzung um den Roboter. Daher konnte die Drehung des Motors für die 90-Grad-Drehung als feste Konstante im Quelltext übernommen werden, wobei Richung und Anzahl der 90-Grad-Drehungen verändert werden konnten. Die Nummerierung der Körbe gab die Reihenfolge der Erfassung an. Die Ausgangsposition befindet sich in Greifposition des Balles. Als erstes wird eine Rotation entgegen des Uhrzeigersinns vollführt und dann über die Ausgangsposition die anderen beiden Körbe erfasst. Das führt dazu, dass sich die Kabel des Motors der Basis nicht verwickelten. Nach Erfassen der Korbfarben fährt der Roboter in seine Ausgangsposition zurück und gibt im Sekundentakt einen Ton ab, der seine Bereitschaft kenntlich macht. Nun kann ein Ball manuell zwischen die Gabeln der Kralle gelegt werden und per Knopfdruck kann der Sortiervorgang gestartet werden. Jetzt greift die Kralle zu und der Arm hebt sich. Dann wird die Farbe des Balles erkannt und mit den Zeichenketten der Körbe verglichen. Es besteht die Nötwendigkeit erst den Ball zu greifen und dann die Farbe zu erkennen, da sich nur dann der Sensor im optimalem Abstand zum Ball befindet. Nun Dreht sich der Arm dem entsprechendem Korb zu, senkt die Kralle leicht über den Korb und öffnet sie. Der Ball wird dadurch in den richtigen Korb gelassen. Danach dreht sich der Arm entgegen seiner vorherigen Bewegung, um die Kralle abzusenken und so die Ausgangspostition zu erreichen. Nun wird wieder im Sekundentakt ein Ton abgegeben, um einen Ball einzulegen. Um Bälle, deren Farbe nicht vorhanden ist, nicht in die Körbe einzusortieren, wurde der Ball im Testaufbau über der Ausgangsposition fallen gelassen. Falls nach 10 Sekunden kein Ball eingelegt wurde, werden 3 differenzierte Töne ausgegeben, die erkennen lassen, dass der Roboter sich nun in den Ruhemodus begibt und sich von der Software trennt. Um den Vorgang neu zu starten, muss das Programm neu initialisiert werden.

Damit kann der Roboter die gewünschten Prozesse ausführen und die Bälle der Farbe nach sortieren, insofern die Farbe des Balls als Korb vorhanden ist.

E. Anwendung an dem Problem

Um nun die Problemstellung des starken, weltweiten Müllaufkommens zu berarbeiten, ist es nötig, den Roboter in diesen Felden arbeiten zu lassen. Dafür wäre es allerdings notwendig, das Sortierkriterium des Roboters anzupassen, da unterschiedliche Müllklassifizierungen nicht nach der Farbe zu unterscheiden sind. Gerade bei den Abfällen von Kunststoffen gibt es viele zu prüfende Kriterien, die eine sortenreine Sortierung erst möglich machen. Auch mehrschichtige Kunststoffe mit verschiedenen Eigenschaften erschweren den Prozess [1]. Mit einem geeigneten Sensor oder mehreren aufeinander folgenden Sortierungen nach verschiedenen Kriterien kann eine solche sortenreine Selektierung garantiert werden. Außerdem muss der Vorgang automatisiert werden. Das heißt, dass die manuelle Eingabe des Müll und die Bestätigung per Tastendruck ersetzt werden müssen. Wenn Beispielsweise der Müll auf einem Förderband transortiert wird gibt es immer Nachschub und der Roboter muss nicht warten, bis der Müll in geeigneter Position ist. Die Körbe können dann durch weitere Förderbänder ausgetauscht werden, um den Müll von weiteren Robotern oder Sortieranlagen spezifizieren zu lassen.

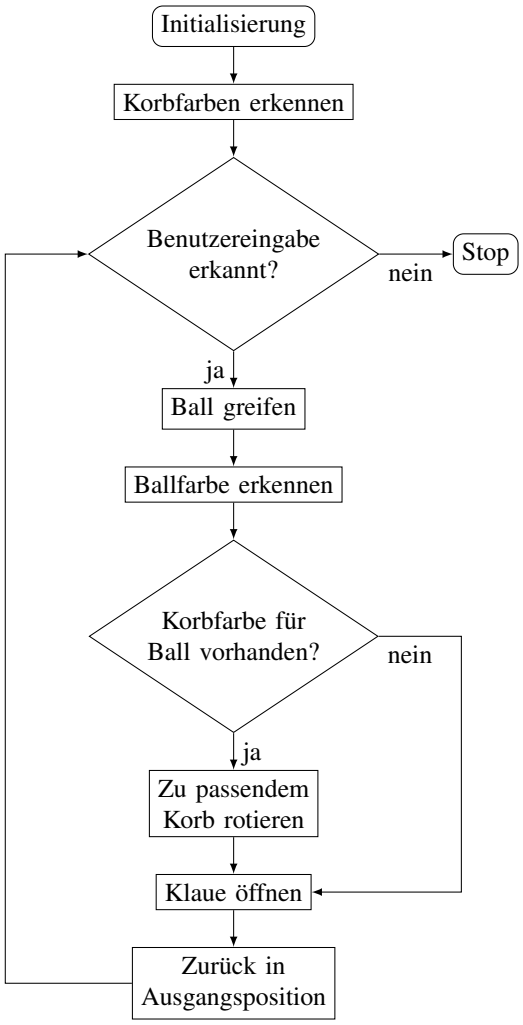


Abbildung 2. Programmablaufplan der Ballsortierung

Durch seine kompakte Bauweise ist es aber auch möglich, dass der Roboter in Privathaushalten benutzt werden kann, um den Müll bereits dort zu sortieren. Das macht es aber nötig, dass der bereits getrennte Abfall dann separat entsorgt und abgeholt werden muss. Eine Sortierung am Ort der Wiederverwertung ist daher vorteilhaft.

F. Verweis auf Literaturquellen

Das stetig steigende Aufkommen von Müll in Deutschland ist ersichtlich in [2].

IV. Ergebnisdiskussion

Der Roboter war in der Lage, den selbst gewählten Anforderungen zu entsprechen: Er konnte die Farbe der Körbe selbstständig erfassen und war damit flexibel in der Ausgangssituation, die Bälle wurden in der Farbe fast immer richtig erfasst, wobei Ausnahmen nur durch die Ungenauigkeit der verbauten Sensoren auftraten und die Bälle in den dafür vorgesehen Korb gelegt werden konnten. Auch beim Legen gab es selten Probleme, da sich die Motoren nicht immer um den den selben Winkel gedreht haben. Auch hier wäre es deutlich besseres Ergebnis erzielt worden, wenn genauere Motoren verbaut worden wären. Durch eine stabilere Bauweise könnte sich der Aktionsradius der Roboters jedoch auch noch erweitern. Durch eine andere Konstruktion wäre es auch denkbar, dass der Roboter die Bälle nicht vom Boden aufnehmen muss.

V. Zusammenfassung und Fazit

Es lässt sich sagen, das eine derartige Lösung möglich ist, es jedoch durch den nötigen Sensor nicht möglich war, den Roboter Müll sortieren zu lassen. Es ist aber durch eine größere Auswahl an Bauteilen definitiv möglich, auch das in einer sehr kompakten Bauweise umzusetzen.

Literaturverzeichnis

[1] Stumpfe, Miriam: *Probleme fürs Plastik-Recycling*. https://www.br.de/themen/wissen/recycling-kunststoffe-plastikmuell-100.html. Version: Mai 2019

[2] Umweltbundesamt: *Abfallaufkommen*. https://www.umweltbundesamt.de/daten/ressourcen-abfall/abfallaufkommen. Version: März 2019

Construction of a Robot for Sorting Balls

Report on the project of the LEGO Mindstorms seminar

Tsoi Samuel Tze-Kei, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

Abstract*—Nowadays, the technology development is unprecedentedly fast. The development of technology leads to a trend in many of the industries: automation. Automation has a lot of advantages for the industry, such as avoiding manual errors, lowering the running costs and extending the production time. The robotic arm plays an important role in automation, it has a wide range of uses in different industries. This LEGO robotic arm is inspired by the garbage sorting claw, so this LEGO robotic arm is designed for sorting. The preliminary idea is a robotic arm that can sort some object by its color, in this case is sorting the balls to the corresponding basket according to its color.

I. INTRODUCTION

A robotic arm is a very popular tool in many industries, such as logistics, recycling industry and manufacturing. Because of this trend, it was decided to build a robotic arm with LEGO NXT.

The goal of this project is to build a robot that can sort balls to different baskets according to its color. It is challenging to construct a robotic arm, because a functional robotic arm involve a lot of concepts from mechanics. All movement of the arm and the claw must work with the gear wheels.

At the end, the goal of sorting balls is achieved. The following will be status of technology, the detail of the construction process, the possible improvement of the robot, the result and conclusion. This is a photo of the final product:

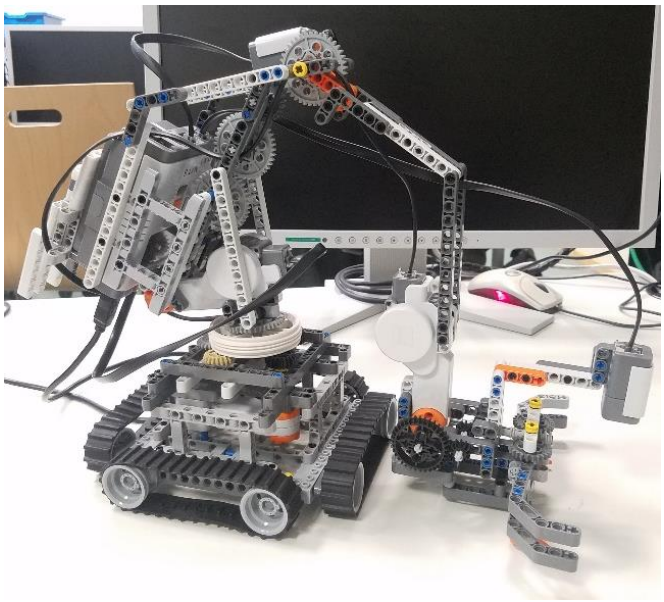


Figure 1: The finished LEGO robotic arm

II. STATUS OF TECHNOLOGY

A. The earliest robotic arm

The earliest robotic arm was built in 1937, the history of the “industrial robot” is long and it has been so well developed.

B. The robotic arm nowadays

For example in the container port, the robotic arm sorts the containers and loads them to different trucks or cargo ships. In the recycling industry, there is a robotic system called “ZenRobotics Recycler”[2], this system is using robotic arm to separate the construction material according to the type of material.

DOI: 10.24352/UB.OVGU-2019-073

Figure 2: The ZenRobotics Recycler
https://zenrobotics.com/wp-content/uploads/ZenRobotics-Recycler-overview2-1.jpg

III. THE CONSTRUCTION

A. The robotic arm is assembled from three parts: the base, the arm and the claw. The claw is a relatively easy part to build. There is a motor and a color sensor in the claw. The motor controls the movement of the claw so that it can open and close. The function of the light sensor is to detect the color of the balls and the color of the baskets. The reason to set the color sensor above the claw, but not at the side or under the claw, is to reduce the error of the color detection. Due to the limitation of the color sensor, that the brightness can affect the result a lot, setting the sensor above can ensure that there is enough light source for the object, then the result can be more accurate.

B. The second part of the robot is the arm, which connect the base and the claw. Only a motor is used in this part at the beginning, after several tries of experiment a touch sensor is added for the user input. The motor is connected to a set of gear wheels, when the motor spins, it can drive the gear wheels and it can lift up or put down the front part of the arm, which is connected to the claw. The touch sensor is used to detect the user input. If the touch sensor is pressed by the user, then the balls sorting program will start, otherwise it will just wait for the input by the user. The reason to add the touch sensor is to prevent the situation, where the balls sorting program is still running even there is no ball to sort or the ball has not yet been given.

C. The base is the hardest part of the whole robotic arm. The main component is the motor, which controls the rotation of the whole robotic arm and claw. The base is static, so four continuous tracks are used, which are made of rubber in order to increase the friction between the base and the ground. The biggest problem of the base is the balancing and the weight of the upper arm. There are 2 motors, 2 sensors, a lot of gear wheels and other components in the upper arm. When the upper robotic arm was lifted up, it will fall down immediately due to the weight of the claw. So the NXT block and some LEGO bricks are added to the back of the arm to balance the whole upper arm. But after this modification, another problem comes out. There is only one axis for the rotation and it is hard to rotate when the upper arm is too heavy. The solution of the rotation problem is to add a large wheel between the rotation axis and the arm and a set of gear wheels underneath. The wheel and the gear wheels can distribute the weight of the arm and reduce the friction when it rotates.

D. For the programming part, MATLAB is used as the programming language. When the program starts, the robotic arm will rotate to three default positions, where the baskets are, and scan the color of them. Then it will return to the start position and wait for user input by pressing the touch sensor. After that the program will go to a clause. If there is a signal from the touch sensor, then the claw will close and detect the color of the ball. It will compare the color of the ball and the colors of the baskets, which was scanned and recorded in three variables at the beginning. If the colors are matched, the arm will rotate to the corresponding position and drop the ball into the basket. If the color sensor cannot detect the color of the ball or the colors do not match, the claw will release the ball and wait for new signal from the touch sensor. The following is the flowchart of the whole balls sorting program:

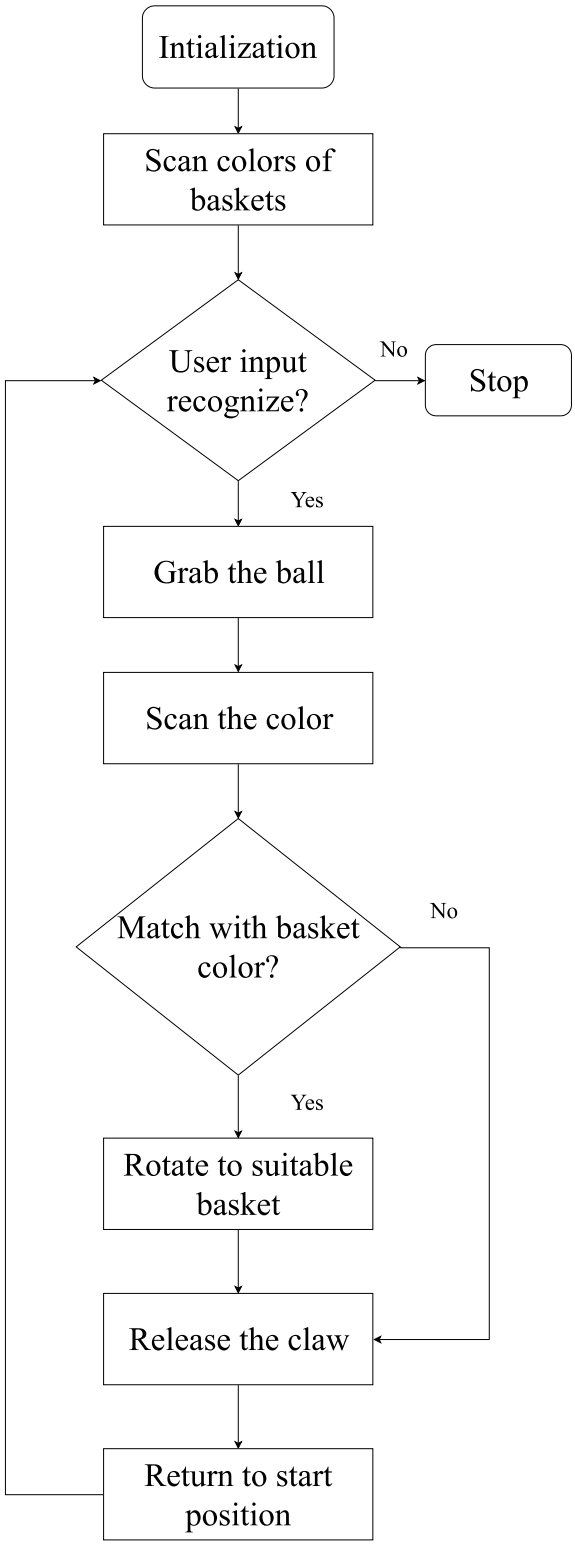


Figure 3: Flow chart of the program

E. Some problems of this robotic arm have not been solved, one of the biggest problem is the accuracy of the robot. As mentioned before, the result of the color sensor heavily depends on the brightness of the environment. So sometimes the robot cannot detect the correct color because of the limitation of the color sensor. Another problem is the rotation angle of the motor in the base part. This is also a problem of the hardware, the motor cannot turn to the same angle every time with the same program. So sometimes the robotic arm cannot deliver the ball to the basket because of the inaccuracy of the motor rotation. The program for the rotation part has adjusted for many times, to make the rotation as accurate as possible, but it can still have error.

F. The LEGO robot has a huge potential to be developed further. One of the possible developments is to be even more automated. It is possible to add a ramp, which deliver the balls to the claw automatically. The touch sensor can be used as a detector for the ball, when the ball come from the ramp and trigger the touch sensor, then the sorting part of the program starts. Due to the limitation of time, this automation has not been made but it is a possible enhancement for the robotic arm.

IV. RESULT

In this stage, the LEGO robotic arm can deliver three balls with different colors into three corresponding basket most of the time. When there is failure of the ball sorting part, the main reason is about the error of the color sensor or the inaccuracy of the motor rotation. Although there is some error because of the hardware, but the result is also satisfying. The GUI of the robot is simple, there is only a start button. Because the robotic arm is designed to use the touch sensor as the user input. This robotic arm model can apply for garbage classification, container sorting, package sorting and so on.

V. CONCLUSION

The LEGO robotic arm can achieve the goal of sorting the balls according to the color, so it is a successful project.

When the robotic arm is used in logistic industry, another possible development in the future is sorting by other criteria, not only by the color. For example sort the containers by its weight, destination to be delivered, types of product and so on. Compared with the robotic arm in the industry, the functions are similar. So it is a satisfying result to build a robotic arm with LEGO that can compare with the existed industrial robot. But this robot also has some limitations, such as the inaccuracy of the sensor and motor, it can only sort the balls by color and the upper arm is still not very stable. In summary, this robot has its limitation, but it can achieve the goal. So it is a successful project.

BIBLIOGRAPHY

[1] WIKIPEDIA, THE FREE ENCYCLOPEDIA: Industrial robot. https://en.wikipedia.org/wiki/Industrial_robot Version: March 2019

[2] Arion McNicoll and Stefanie Blendis, (2013, June 10) “Green machine: Intelligent robot system recycles waste,” https://edition.cnn.com/2013/06/07/tech/zenrobotics-recycling-robot/index.html

Bau eines Aufräumroboters

Stefan Dunkel, Elektro-und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Abstract*— In der heutigen Zeit werden immer mehr Arbeiten von Robotern übernommen. In der Industrie können so menschliche Arbeiter entlastet oder eingespart werden. Deshalb wurde im Rahmen des Projektseminars Lego Mindstorms versucht ein Roboter zu bauen der mithilfe einer Kamera einen festgelegten Bereich aufräumt. Das Fahrzeug wurde aus Lego konstruiert und mithilfe von Matlab programmiert. In diesem Paper wird auf die Realisierungsversuche für diesen Projekt eingegangen. Es wurde zuerst versucht einen festen Weg zu den Objekten zu berechnen der anschließend abgefahren wird. Dieser Ansatz wurde im Verlauf des Projekts durch eine aktive Regelung ersetzt. Zur Erkennung von Objekten wurde eine Webcam verwendet. Zur Bildverarbeitung wurden zwei verschiedene Methoden angewendet um die Position von Objekten im Bild zu erkennen. Um Objekte aufzunehmen wurde das Fahrzeug mit einem Greifer ausgestattet.**

***Schlagwörter*— Automatisierung, Aufräumen, Kettenfahrzeug, Webcam, Objekterkennung**

I. Einleitung

In einer Zeit wo der technische Fortschritt immer mehr voranschreitet gewinnen Roboter immer mehr an Bedeutung. Es können immer mehr Prozesse automatisiert und von Robotern ausgeführt werden um den Menschen zu entlasten. Es können ungewollte, schwere oder für den Menschen zu gefährliche Arbeiten von Maschinen übernommen werden. Dabei sind die meist sogar effizienter. Ein Beispiel für Roboter bei denen die Effizienz noch verbesser werden kann sich Aufräumroboter. Der klassische Staubsauger-Roboter bewegt sich im Chaosprinzip durch den Raum. Tut er das lange genug hat er irgendwann jede Stelle im Raum einmal befahren. Diese Methode ist sehr zeitaufwendig. Im Rahmen des Projektseminars Lego Minstorms war es das Ziel einen Roboter zu bauen der einen Bereich(z.B. eine Lagerhalle) selbstständig aufräumen kann und dabei möglichst effizient vorgeht. Eine Kamera die Den Bereich von oben filmt ermöglicht es dem Roboter direkt anzusteuern ohne vorher durch den Raum fahren und danach suchen zu müssen. Das spart Zeit die Sonst für das suchen von Objekten verschwendet wird.

II. Hauptteil

Das Ziel war es einen Roboter zu bauen der mithilfe einer externen Kamera einen Bereich erfasst Objekte erkennt selbstständig ansteuert, aufnimmt und an einen festgelegten Sammelpunkt ablegt.

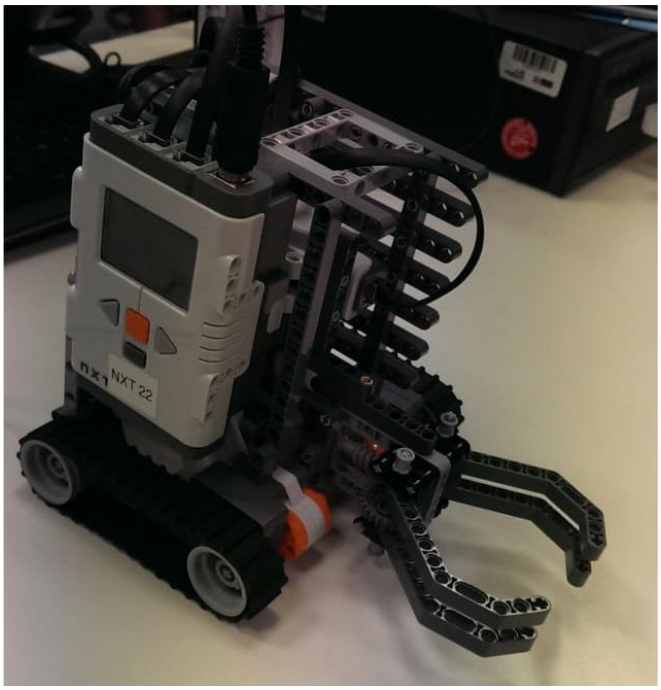


Abbildung 1: Aufräumroboter

A. Konstruktion

Zum Bau des Roboters(Abb.1) wurde Lego von LEGO-Mindstorms verwendet. Dabei wurden drei Motoren sowie ein NXT-Brick verwendet. Der Kern war der Greifmechanismus(Abb.2) um den herum der Roboter aufgebaut wurde. Die zwei Motoren für den Antrieb wurden links und rechts unten direkt neben den Ketten verbaut. Um den in der Front befestigten Greifer öffnen und schließen zu können wurde der dritte Motor im Heck verbaut. Der NXT-Brick wurde an der rechten Seite angebraucht, was ein Gegengewicht auf der linken Seite Notwendig machte. Die Kamera muss separat aufgebaut werden um die zu räumende Fläche von oben erfassen zu können.

DOI: 10.24352/UB.OVGU-2019-075

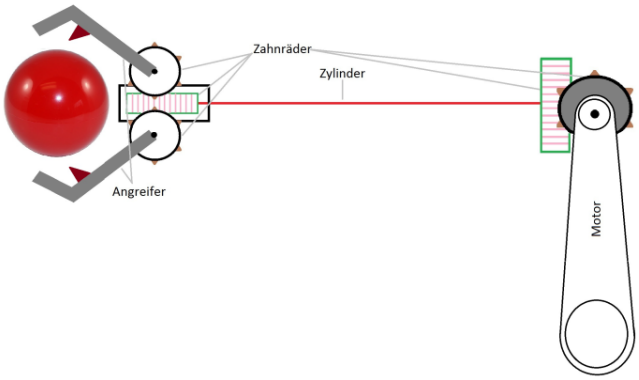


Abbildung 2: Greifmechanismus

B. Objekterkennung und Ansteuerung

Erster Ansatz:

Um in dem von der Kamera geschossenen Bild Objekte zu erkennen wird das Bild in ein Schwarz-Weiß-Bild umgewandelt. Dazu wird es zuerst in ein Graustufenbild konvertiert und anschießen dann mit einem Grenzwert für die Graustufen in das gewünschte Schwarz-Weiß-Bild. Dieses Bild ist nun in einer Matrix aus Nullen und Einsen gespeichert. Dabei ist eine Null eine Grenzwertunterschreitung(Weiß) und eine Eins eine Grenzwertüberschreitung(Schwarz). Diese Matrix wird nun erst spaltenweisen und dann zeilenweise untersucht. Das Ergebnis wird in einem Zeilenvektor geschrieben, jeweils für die X-Achse(Spalten) und die Y-Achse(Zeilen), für jede Spalte/Zeile ein Eintrag. Dabei wird wenn eine Eins in der Spalte/Zeile gefunden wurde eine Eins und falls nur Nullen gefunden werden eine Null für die untersuchte Spalte/Zeile in den Zeilenvektor geschrieben. So erhält man eine Projektion der Objekte auf die X- und Y-Achse. Damit kann man nun die Position der Objekte auf den Achsen bestimmen. Dazu wird zuerst die Position der Ersten Eins in dem jeweiligen Zeilenvektor bestimmt. Dann wird von dieser Position ausgehend die Position der nächsten Null ermittelt. Diese zwei Werte markieren den Anfang und das Ende des Objektes. Aus diesen Werten wird ein Mittelwert gebildet. So erhält man die Position des Objektmittelpunkts auf der jeweiligen Achse. Dieser Vorgang wird ebenfalls für die letzte gefundene Eins wiederholt nur in entgegengesetzter Richtung. Aus diesen Werten ergeben sich bis zu vier Schnittpunkte. An diesen Schnittpunkten werden Objekte vermutet. Auf diese Weise werden allerding mehr Objekte gefunden als tatsächlich existieren. Deshalb werden im nächsten Schritt die gefunden Objekte auf ihre Existenz überprüft. Dazu wird ermittelte Position in dem zuvor erstellten Schwarz-Weiß-Bild überprüft. Wenn an der Stelle eine Eins gefunden wird existiert das Objekt und kann angesteuert werden. Falls eine Null gefunden wird ist dort kein Objekt das verräumt werden kann. Die Position des ersten auf diese Weise bestätigten Objekts wird nun von dem Roboter Angesteuert. Dieser startet im Koordinatenursprung. Die Position des zu sammelnden Objekts wird erst in X- und dann in Y-Richtung angesteuert. Wurde der Zielort erreicht wird der Greifer geschlossen und der Roboter fährt den gleichen Weg zurück zu seiner Ausgangsposition. Das Objekt wird abgelegt und der gesamte Vorgang wiederholt bis keine Objekte mehr gefunden werden.

Zweiter Ansatz:

Beim zweiten Ansatz wurde für die Objekterkennung eine Toolbox von MATLAB verwendet. Diese sucht im Bild Objekten mit einer bestimmten Farbe. Es wird jetzt auch nicht mehr nur die Position des zu verräumenden Objekts sondern auch die des Roboters bestimmt. Hier würde für den Roboter die Farbe Rot und für des Objekt Blau verwendet. Das der Roboter seine Position im Bild erkennt ermöglicht das Objekt mit einer aktiven Regelung anzusteuern. Dazu wird der Mittelpunkt beider Objekte bestimmt. Das Ziel soll auf möglichst direktem Weg angesteuert werden. Dazu wird der Winkel zwischen den Objekten bestimmt. Mithilfe eines Grenzwertes wird der Kurs des Roboters immer wieder angepasst.(Abb.4) Dabei wird bei einer Unterschreitung nach links und bei Überschreitung nach rechts korrigiert. Nachdem der Kurs angepasst wurde bewegt er sich ein Stück vorwärts. Nun wird wieder die Position des Roboters überprüft und dieser Vorgang wiederholt bis ein Mindestabstand zwischen Objekt und Roboter unterschritten wurde. In diesem Fall wird der Greifer geschlossen und Objekt aufgenommen. Dies kann nun zum Koordinatenursprung oder jeder anderen beliebigen vorher festgelegten Punkt abgelegt werden. Dazu kann die selbe Art Regelung verwendet werden.

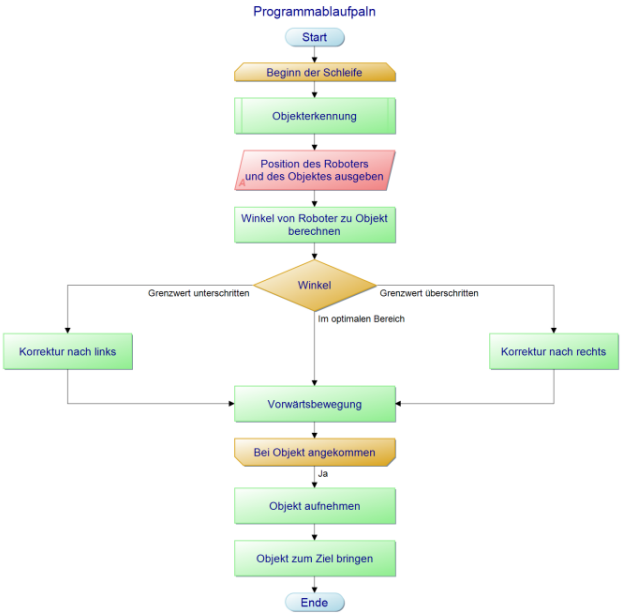


Abbildung 3: Programmablaufplan zweiter Ansatz

III. ERGEBNISDISKUSSION

Es wurde ein Kettenfahrzeug aus Lego gebaut das sich mit einer Differenziallenkung im Raum steuern lässt. Mithilfe des vorn am Fahrzeug verbauten Greifers ist es in der Lage Objekte vor sich aufzunehmen. Der erste Ansatz zur Objekterkennung funktioniert soweit allerdings gab es noch ein Problem bei der Erkennung mehrerer Objekte wenn diese sich überschneiden. (Abb.5) Dabei ist es dem Programm nicht möglich diese voneinander zu unterscheiden und sie werden als ein Objekt erkannt. Außerdem hat der Algorithmus einen

blinden Fleck in der Mitte, da nur die 4 äußersten Objekte erkannt werden können. Es hat sich gezeigt, dass die Ansteuerung des Objekts mit einem vorher berechneten Weg sehr ungenau und störanfällig ist. Da der Roboter aus Lego gebaut wurde nahm die Ungenauigkeit immer mehr zu je größer der zu räumende Bereich wurde. Nur eine kleine Abweichung in X-oder Y-Richtung führte dazu das Ziel verfehlt wurde.
Der zweite Ansatz löst diese Probleme allerdings traten dort wieder neue auf. Das zu verräumende Objekt wird erkannt. Die Regelung zur Ansteuerung des Objekts ist wegen einem Mangel an Zeit nicht mehr fertig geworden und ist nur in grob Implementiert und es treten immer noch Fehler auf. In der Testphase ist es einmal gelungen das Objekt mit einer aktiven Regelung anzusteuern. In allen weitern Versuchen ist der Roboter aus dem Bild gefahren oder hat sich im Kreis gedreht. Als Ursache wird ein Problem Bei der Winkelbestimmung vermutet das aber auf Grund mangelder Zeit nicht mehr behoben werden konnte.

IV. Zusammenfassung und Fazit

Das Ziel war es einen Roboter zu konstruieren der selbständig in einem von einer Kamera erfassten Bereich Objekte erkennt sie selbständig ansteuert, aufnimmt und anschießenden zu einem vorher festgelegenen Ziel bring. Dabei wurde versucht zwei verschiedene Methoden umzusetzen. Bei der ersten sollte ein fester Weg ermittelt und dann abgefahren werden. Die zweite Version benutzt zur Ansteuerung eine aktive Regelung. Das macht den Prozess weniger störanfällig und Abweichungen können ausgeglichen werden. Der Roboter wurde aus zeitliche Gründen und Fehlentscheidungen die zum Verlust von Zeit führten nicht fertig gestellt. Mit mehr Zeit könnt man erst die noch bestehen Probleme lösen undweitere Funktionen ergänzen. Zum Beispiel könnte man eine Funktion zur Erkennung von Hindernissen einbauen, da die momentane Version nur für eine Barriere freie Umgebung ausgelebt ist.

Anhang

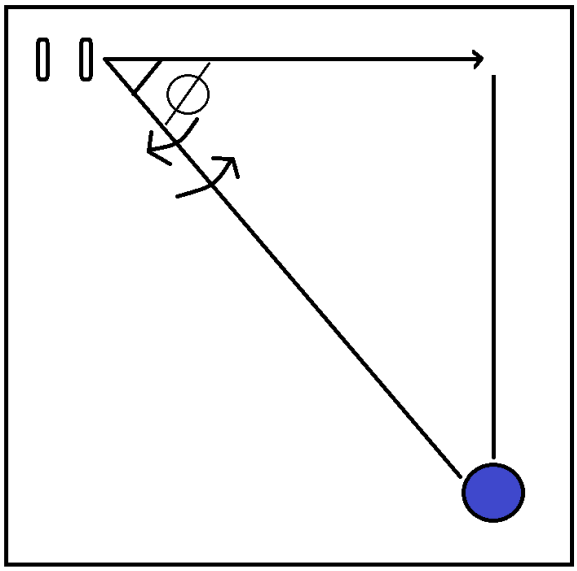


Abbildung 4: Model Regelung

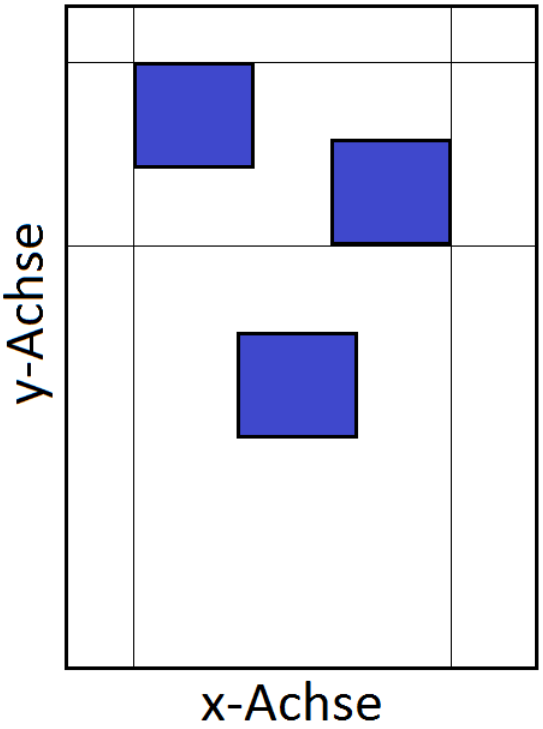


Abbildung 5: Überlagerungsproblem