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).

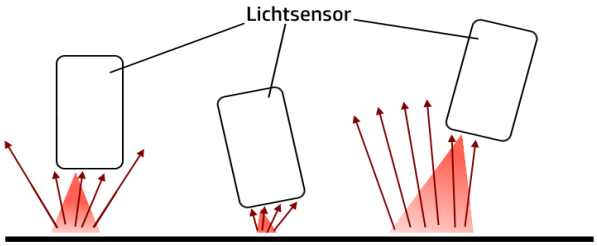

Abb. 1: vereinfachte Darstellung der Funktionsweise des Lichtsensors

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.

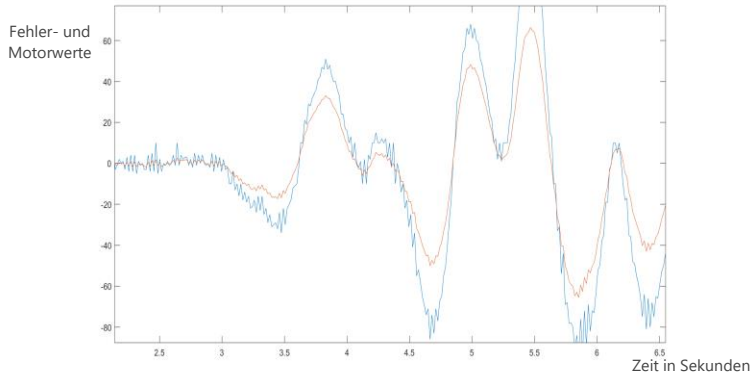

Abb. 2: Testdurchlauf Teil 1

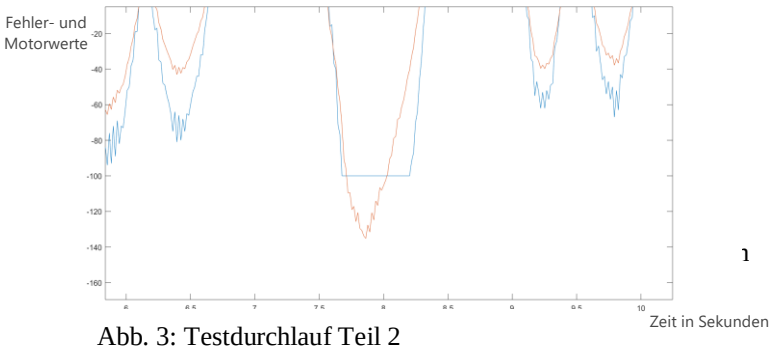

Abb. 3: Testdurchlauf Teil 2

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.

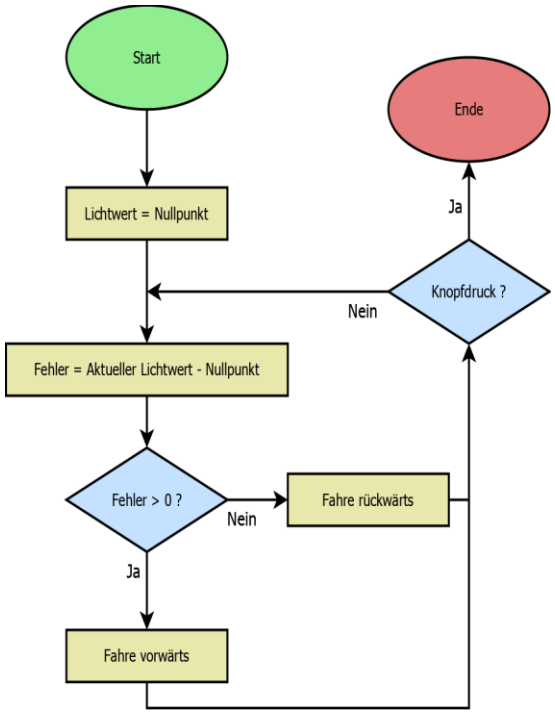

Abb. 4: Programmablaufplan des Balance Bot Programms

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