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

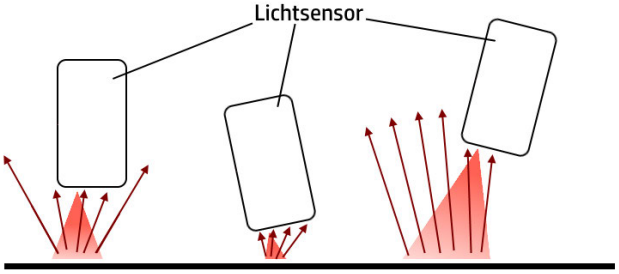

Abbildung 2. Vereinfachte Darstellung der Funktionsweise des Lichtsensors

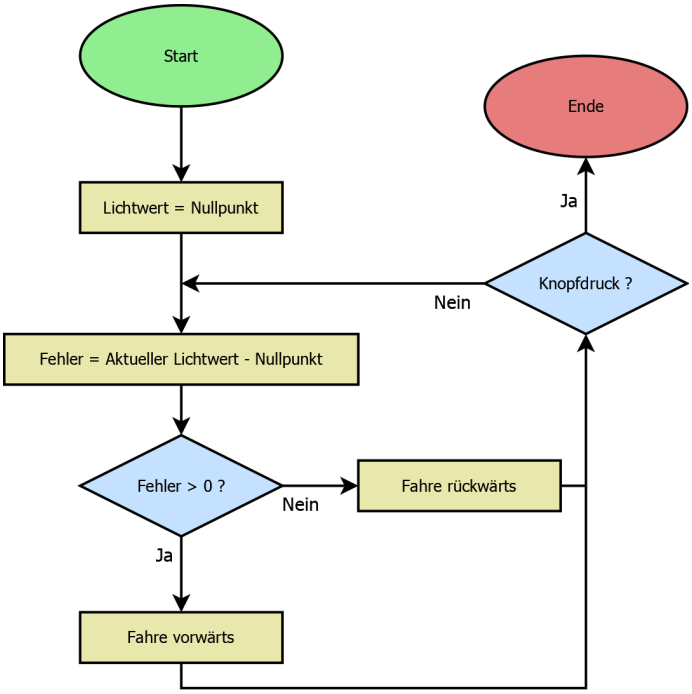

Abbildung 3. Vereinfachter Programmablaufplan

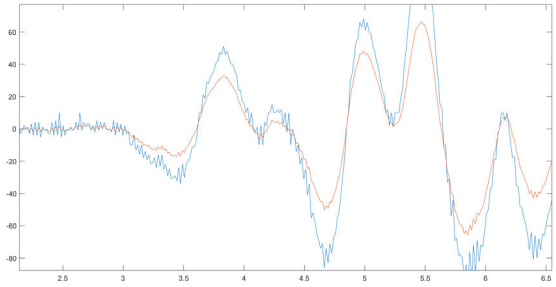

Abbildung 4. Graph der Fehler- und Motorwerte

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

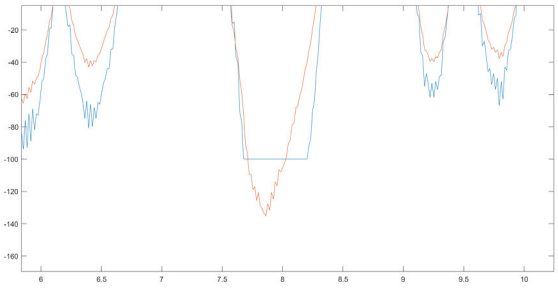

Abbildung 5. Graph der Fehler- und Motorwerte mit abgeschnittenden Werten

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