Selbstbalancierender Roboter - Der intelligente Weg zum Ausgleich

Ahmed Abdelgaber Abdou M. Badawy, Mechatronik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Zur Zeit besteht ein wachsendes Interesse für Anwendungsgebiete wie Service-Robotik, Logistik- und Produktionsindustrie. Mit hohen Präzision und Schnelligkeit können intellegente Roboter Aufgabe erledigen, die für Menschen zu gefährlich oder zu anstrengend sind. Da selbstbalancierender Roboter leicht und leistungsfähig, können solche Aufgaben ausgeführt werden.**

Die Structure des selbstbalancierenden Roboter entspricht die Kombination des linearen umgekehrten Pendels und des mobilen Roboters mit Rädern. Da der Roboter von Natur aus unstable ist, wird aus der Regelungstechnik ein Regler verwendet, damit der Roboter auf nur zwei Rädern aufrecht halten kann. Bestehend aus Sensoren und Aktoren kann der Roboter durch ständiges Ausgleichen der vertikalen Neigung ein stabiles Zustand erreichen. Im Rahmen des Projektseminars wurde ein selbstbalancierenden Roboter mit Hilfe eines Gyroscope gebaut und mit MATLAB programmiert.

***Schlagwörter*—zweirädriger Roboter, PID-Regler, Regelungstechnik, Self-balancing Robot, Segwayroboter**

I. Einleitung

INNERHALB der Robotik-Forschung werden immer neue Entwicklungen erstanden, welche zu einem wachsenden Bedarf an Roboter führt. Sie können unterschiedliche Formen und Größen annehemn, von kleinen Armrobotern bis zu großen industriellen Maschinen. Insbesondere in der Lagerverwaltung übernehmen immer Roboter komplexe Aufgaben und stellen maximale Effizienz und Produktivität sicher [1]. Allerdings ist das Problem, Lagerroboter erfordern große Auflagefläche, haben hohe Anschaffungskosten und aufgrund der technischen Komplexität benötigen regelmäßige Wartung und Reparatur. Zur Losung dieses Problems kommt der selbstbalanciernde Roboter zum Einsatz. Selbstbalanciernde Roboten sind sehr flexibel, platzsparend und sehr wendig. Ein gutes Beispiel dafür ist der zweirädrige Roboter Handle, der von Boston Dynamics entwickelt wurde [2]. Die Kombination der Räder und der Gelenke in den Beinen ermöglicht ihm, bis zu 45 Kilogramm zu heben und eine Geschwindigkeit von bis zu 14,5 km/h zu erreichen. Der Roboter wird hauptsächlich zum Entladen von LKW und das Transportieren von Waren eingesetzt.

Innerhalb des LEGO-Mindstorms-Projekts ist ein selbstbalancierender Roboter mit einem Gyroscope zu bauen, der stabil aufrecht bleiben und äußere Störungen ertragen kann. Dieser wird mit MATLAB programmiert und die Konstruktion erfolgte mit LEGO Mindstorms.

DOI:10.24352/UB.OVGU-2023-016

II. Vorbetrachtungen

Aus der Regelungstechnik wird ein PID-Regler zur Stabilisierung des Roboters verwendet. Das mechanische Design ist von entscheidender Bedeutung. Es hat einen Einfluss auf die Stabilität des gesamten Systems.

A. Segwayfunktionsweise

Durch den Einsatz von Gyroskop- und Beschleunigungssensoren kann der Segway das Gleichgewicht halten, indem die Geschwindigkeit der beiden Räder reguliert wird. Die Position und Bewegungen des Fahrers sind kontinuierlich durch die Steuerungselektronik zu überwachen. In Abbildung 1 sieht man, dass der Roboter sich aufgrund des Drehmoments nach vorne umkippen lässt. Dieses Drehmoment wird durch die Gewichtskraft $mg \cdot \sin(\theta)$ erzeugt. Die Motoren kommen hier zum Einsatz, um das Drehmoment entgegenzuwirken und der Roboter aufrecht halten zu können.

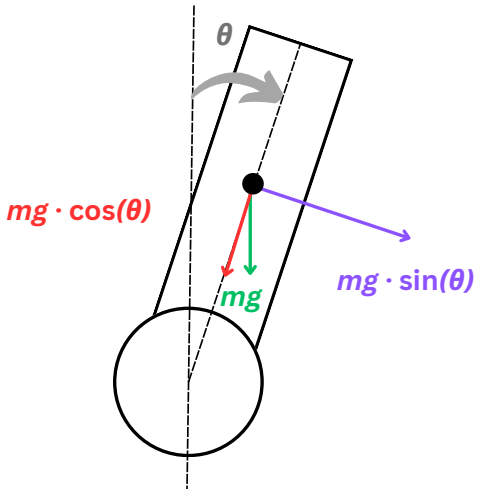

Abbildung 1. Modell eines selbstbalancierenden Roboters

B. PID-Regler

In vielen Bereichen der Technik und Industrie lassen sich Regler eingesetzt. Ein Regler hat die Aufgabe, die Ausgangsgröße eines Systems oder Prozesses zu halten oder zu steuern. Der häufigste ist der PID-Regler, der aus 3 verschiedenen Regler besteht [3]. Der P-Regler steht für den Proportionsanteil und verhält sich proportional zur Regelabweichung. Der I-Anteil minimiert den Fehler im Laufe der Zeit durch Integration der

Regelabweichung. Um schnellere Reaktionen zu erzielen, ist der D-Regler dafür geeignet, der die Ableitung der Regelabweichung nach der Zeit nimmt.

III. UMSETZUNG

A. Aufbau

Der selbstbalancierende Roboter besteht hauptsächlich aus ein LEGO-Stein, Gyroscopesensor und 2 Elektromotoren. Beim Bauen müssen viele Faktoren betrachtet werden. Eine gleichmäßige Verteilung des Gewichts ist entscheidend, um der Roboter stabil zu halten und die Störungen aufgrund des Gewicht zu reduzieren. Dies kann dazu führen, dass das gesamte System zu Schwingung kommt. Bei dem ersten und zweiten Prototyp lag der LEGO-Stein vertikal mit der Radachse. Als der Roboter aufrecht gehalten und losgelassen wird, fällt er viel schnell zu Boden. Das liegt daran, dass eine Reihe von Batterien sich auf der Rückseite des Steins befindet, was aufgrund des Gewichts ein hohes Drehmoment erzeugt. Deshalb wurde die beiden Konstruktionen verworfen. Allerdings beim dritten Prototyp wurde der Stein senkrecht zur Radachse und parallel zum Boden platziert, sodass der Schwerpunkt so nahe wie möglich zum Boden, um eine bessere Stabilität zu gewährleisten. Durch Halten und Loslassen lässt sich deutlich erkennen, dass der Roboter langasmer nach vorne kippt. Zur Bestimmung der Position des Roboters kann ein Gyroscop- oder Lichtsensor verwendet. Anfänglich wurde die Sensoren verbaut und deren Werte aufgezeichent, um zu sehen, welche Sensor bessere Ergebnisse ausgibt. Nach einigen Test konnte man erkennen, dass der Gyroscope genauere Werte aufweist. aber es wird im Laufe der Zeit ungenau wegen des Drifts. Der Gyroscope ist auf der Radachse angebracht, wodurch die Rotationsbewegungen des Roboters in Echtzeit richtig erfasst werden können. Durch die Montage von kleinen Reifen war es deutlich, dass das von den Motoren erzeugte Moment nicht ausreichte, um die Position des Roboters gut genug zu korrigieren. Um das Problem zu lösen, sollten große Reifen verbaut werden, um ein hohes Moment zu gewährleisten [4]. Der endgültige Aufbau ist in Abbildung 5 zu sehen.

B. Funktionsweise des PID-Reglers

Eine zentrale Komponente des selbstbalancierenden Roboters ist der PID-Regler. Der Regler ermöglicht es, dass der Roboter kontinuierlich die notwendigen Anpassungen der Motorleistungen vornimmt, damit das Gleichgewicht des Roboters aufrecht bleiben kann. Ein Regler besteht typischerweise aus drei Anteilen, die jeweils einen individuellen Fehler betrachten. Diese drei Fehler bezeichnet man als Proportionalfehler, Integralfehler und Derivatfehler. Die Definitionen der einzelnen Fehlerarten lautet wie folgt [5]:

$$\text{ProportionalFehler} = \text{Sollwert} - \text{Istwert} \tag{1}$$

$$\text{IntegralFehler} = \text{IntegralFehler} + \text{ProportionalFehler} \tag{2}$$

$$\text{Derivatfehler} = \text{ProportionalFehler} - \text{vorherigeProportionalFehler} \tag{3}$$

Die nächste Formel ergibt sich durch die Multiplikation der einzelnen Fehler mit ihrer jeweiligen Konstante, gefolgt von einer Addition aller Produkte.

$$\begin{aligned} \text{X} &\rightarrow \text{ProportionalFehler} \\ \text{Y} &\rightarrow \text{IntegralFehler} \\ \text{Z} &\rightarrow \text{Derivatfehler} \end{aligned}$$

$$\text{Stellgröße} = (K_\text{P} \cdot \text{X}) + (K_\text{I} \cdot \text{Y}) + (K_\text{D} \cdot \text{Z}) \tag{4}$$

Der Proportionalitätsfaktor K_P kontrolliert die Reaktionsgeschwindigkeit des Reglers auf Abweichungen zwischen der gewünschten Ausgangsgröße und der tatsächlichen Ausgangsgröße. Der Integralitätsfaktor K_I hingegen korrigiert Fehler, die sich im Verlauf der Zeit aufsummieren. Durch Einstellung des Differentialitätsfaktors K_D wird die Geschwindigkeit gesteuert, mit der der Regler auf Änderungen im Fehler reagiert. Um die optimalen Werte für K_P , K_I und K_D zu bestimmen, kann man erst ein Wert für K_P feststellen und dann beobachtet man das Verhalten des Systems. Schließlich stellt man K_I und K_D ein, falls das System eine Regelabweichung hat oder die Reaktiongeschwindigkeit auf Fehler beeinflusst werden soll.

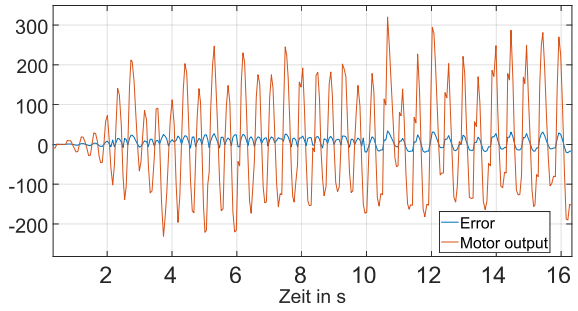

Abbildung 2. Versuch 1 mit $K_\text{P} = 6$ und $K_\text{D} = 0.9$

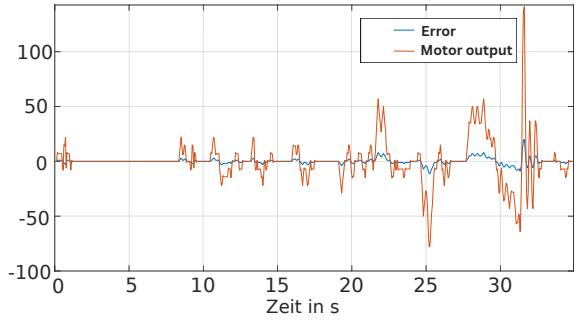

Abbildung 3. Versuch 2 mit K_P = 9 und K_D = 0.4

In Abbildung 2 und 3 werden die Differenz zwischen Soll- und Istwert sowie die Stellgröße dargestellt. Die blaue Kurve repräsentiert den Fehler oder den Eingangswert des Reglers,

während die orangefarbene Kurve die Motorwerte oder den Ausgangswert des Reglers anzeigt. Abbildung 2 zeigt ein kontinuierliches Auftreten von Fehlern, was zu Schwingungen führt. Dadurch wird der Regler stark beeinflusst und instabil. Im Gegensatz dazu sind in Abbildung 3 geringe Fehler und schnelle Reaktionen zu beobachten. Ein erfolgreicher Durchlauf zeichnet sich durch eine orangefarbene Kurve aus, die größer als die blaue Kurve ist und durch ein seltenes Auftreten von Fehlern gekennzeichnet ist.

IV. Programmierung

Zu Beginn wurde die Programmiersprache MATLAB eingesetzt. Allerdings kam es bei Ubermittlung der Daten zu Problemen, da ein Kabel mit dem Stein verbunden war. Dieses Kabel zog den Roboter nach vorne und beeinflusste dadurch das gesamte System als Störgröße. Diese Störung führte zu Regelabweichungen, welche möglicherweise Schwingungen im System hervorrufen konnten. Nach Recherchen wurde festgestellt, dass das MATLAB-Skript nicht auf den LEGO-Brick hochgeladen werden kann. Daher war die LEFO-Software die einzige Lösung, dass das Programm reibungslos ausgeführt werden konnte. Der Algorithmus kann auf beide Softwares angewendet werden.

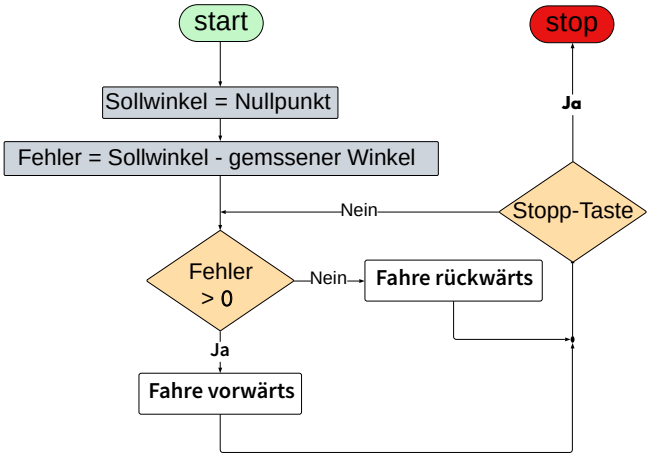

Abbildung 4. Programmablaufplan

In Abbildung 4 wird der Programmablauf für sowohl MATLAB als auch die LEGO Mindstorms Software dargestellt. Am Anfang wird der Winkel oder der Sollwinkel auf 0 gesetzt, um sicherzustellen, dass der Roboter in der aufrechten Position bei Position 0 bleibt. Es gibt zwei Möglichkeiten, um den aktuellen Winkel zu bestimmen. Die erste Möglichkeit besteht darin, die interne Integration der Winkelgeschwindigkeit zu nutzen. Diese Methode kann jedoch aufgrund von Drift zu Fehlern in der Berechnung des Winkels führen, was die Genauigkeit beeinträchtigen kann. Eine genauere Methode zur Berechnung des Winkels besteht darin, die Winkelgeschwindigkeit zu integrieren. Hierbei kann soll die Zeitfunktion definiert werden, um die Zeitdifferenz $(dt = t - to)$ zu berechnen. In einer Endlosschleife lässt sich nun der aktuelle Winkel kontinuierlich abfragen und mit dem Sollwinkel vergleichen. Wenn eine Abweichung zwischen dem aktuellen und dem Sollwinkel besteht, wird dies mithilfe des PID-Reglers korrigiert. Anschließend wird das korrigierte Signal an das System zurückgegeben, um die Position des Roboters entsprechend anzupassen und zu regeln.

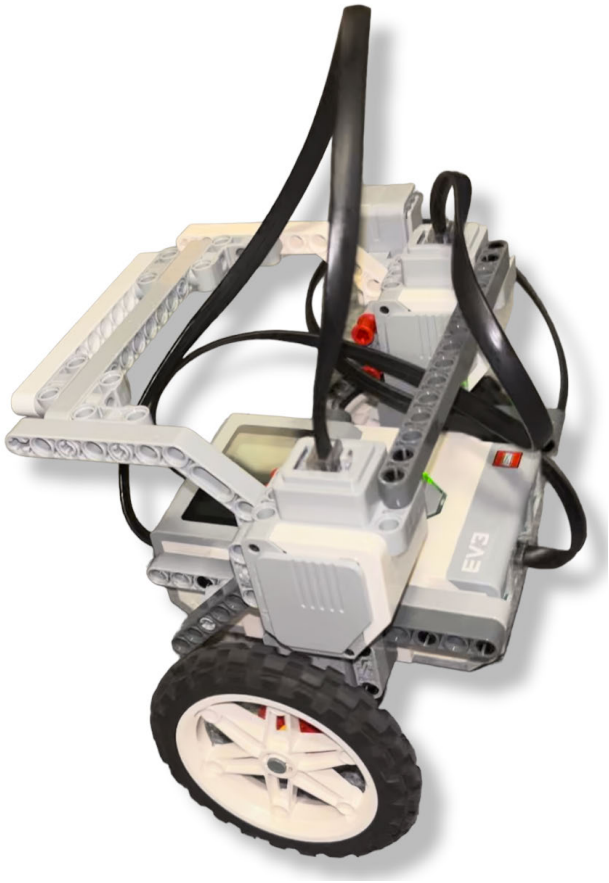

Abbildung 5. dritter Prototyp

V. Ergebnisdiskussion

Abschließend konnte der selbstbalancierende Roboter trotz großer Herausforderungen die gestellten Anforderungen erfüllen. Die Idee war, einen Roboter mit einem Gyrosensor zu bauen, der auf zwei Rädern fahren und Störungen tolerieren kann. Im Laufe der Zeit entstehen jedoch Fehler aufgrund des Drifts des Gyrosensors, wodurch der Roboter nur für eine begrenzte Zeit aufrecht bleiben kann, bevor der Winkel driftet. Um dieses Problem zu lösen, kann die Messung des Gyrosensors mit anderen Sensorinformationen wie beispielsweise Beschleunigungsdaten eines Accelerometers kombiniert und gefiltert werden, um eine genauere Messung des Winkels zu erzielen und den Drift zu minimieren. Mit Hilfe des geplotteten Graphen in MATLAB ermöglicht es, eine stetige Verbesserung der Regelparameter des Roboters. Die Übertragung des MATLAB-Codes auf den LEGO Mindstorms verlief reibungslos und dadurch konnte letztendlich das Hauptproblem, das durch Störungen aufgrund der Verbindung des Kabels verursacht wurde, behoben werden.

VI. Zusammenfassung und Fazit

Dieses Dokument beschreibt den Bau und die Steuerung eines selbstbalancierenden Roboters. Das Ziel war ein Roboter

zu entwickeln, der in aufrechter Position halten und Störungen von außen tolerieren kann. Anfangs wurde das Regelungssystem mit MATLAB programmiert, jedoch wurde das Programm auf die LEGO-Software übertragen, um auftretende Probleme zu lösen. Aufgrund der Drift des Gyrosensors wurde der Winkel durch Integration der Winkelgeschwindigkeit berechnet, um genauere Messungen zu erzielen.

Eine Erweiterung könnte das Hinzufügen von Greifarmen sein, um Gegenstände aufzuheben und zu manipulieren. Dafür könnten zusätzliche Sensoren ausstattet werden, dass der Roboter eine bessere Navigation und Hindernisvermeidung haben kann. Eine weitere Möglichkeit wäre die Integration eines dritten Motors mit einem Bein, sodass der Roboter sich aus eigner Kraft aufrichten kann.

ANHANG

[1] wlw inside business: https://www.wlw.de/de/inside-business/praxiswissen/logistikmanagement/roboterloesungen-fuer-die-logistik-produktiver-und-widerstandsfaehiger

[2] engineering.com: https://www.engineering.com/story/boston-dynamics-shows-off-logistics-version-of-handle-robot

[3]temperatur-profis:https://temperatur-profis.de/temperaturregler/reglertypen-pid/

[4] Autodesk Instructables:https://www.instructables.com/Self-Balancing-Robot-1/

[5] wikipedia:https://de.wikipedia.org/wiki/Regler
https://de.wikipedia.org/wiki/Regler