Projekt MoJo

Joe-Marco Sperling, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In dieser Arbeit wird die Zusammenarbeit zwischen Mensch und Maschine untersucht. Mit theoretischem Wissen aus den Anfängen des Seminars wird ein Roboterarm gebaut und programmiert, dessen Aufgabe es ist, einem Control-Stick zu folgen und dabei möglichst präzise zu sein. Die Umsetzung dieses Projekts gelingt mit Hilfe der LEGO-Mindstorms-Klemmbausteine und dem Programm MATLAB, in dem der Code für den Roboterarm geschrieben wird. Der Arm folgt hierbei den am Control-Stick angebrachten Gyro-Sensoren von LEGO. Diese Sensoren senden Daten an den Code, der daraufhin die Motoren am Arm steuert. Im weiteren Verlauf der Arbeit werden die auftretenden Probleme und deren Lösungen detailliert erläutert. Zudem werden exemplarische Code-Schnipsel vorgestellt, die die Funktionsweise des Systems verdeutlichen und zur Lösung der Herausforderungen beitragen.**

***Schlagwörter*—Gyro-Senoren, LEGO-Mindstorms, MATLAB, Mensch-Maschine Interaktion, Przisionssteuerung, Robotik.**

I. Einleitung

Im Verlauf des LEGO-Mindstorms-Projekt-Seminars war die Aufgabe, einen Roboter zu entwickeln. Unter vielen Ideen wird ein Roboterarm ausgewählt, der einem Gyroskop folgt. Der Arm, der in Abb. 1 zu sehen ist, kann beispielsweise verwendet werden, um einen Raum zu scannen, wenn eine Kamera daran befestigt ist, oder den Raum zu erhellen, wenn eine Taschenlampe angebracht wird. Damit der Arm funktioniert, müssen die Gyroskope ausgelesen und in Daten umgewandelt werden, die von den Motoren ausgeführt werden können. Dies geschieht in Echtzeit. Damit der Arm immer vom gleichen Ausgangspunkt startet, integriert man einen Kill-Switch, der das Programm abbricht und auf die Nullposition zurücksetzt. Zusätzlich muss man die Ungenauigkeiten, die durch die LEGO-Mindstorms-Bauteile entstehen, berücksichtigen und den Code entsprechend anpassen, obwohl auf Präzision großer Wert gelegt wird.

II. Vorbetrachtungen

In der Vorbetrachtung wird detailliert auf die auftretenden Probleme eingegangen. Zudem werden die Lösungen für diese Herausforderungen im Detail erläutert, um ein besseres Verständnis der Vorgehensweise zu ermöglichen.

A. Genauigkeit der Motoren

Die LEGO-Mindstorms Bausteine haben ein Problem mit der Genauigkeit, welches sowohl das Auslesen der Sensoren als auch deren Limitierungen betrifft, aber auch die Ausführung der Motoren betrifft. Die Motoren drehen sich dank des Brake-Modus „Brake“ bereits relativ genau, jedoch war die Präzision nicht ausreichend, da die Zahnräder einen Großteil

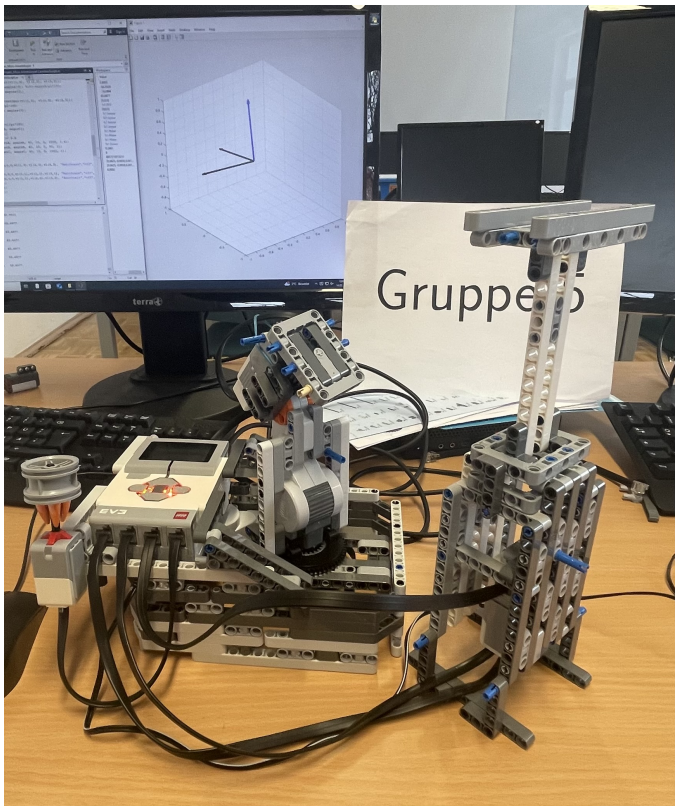

Abbildung 1. Fertiger Roboter mit fertigem Control-Stick

der Ungenauigkeit ausmachten. Aus diesem Grund wurde die Funktion der SpeedControl geschrieben, die den Motor verlangsamt, je näher man dem Zielwinkel kommt, das sieht man in der Abb. 2

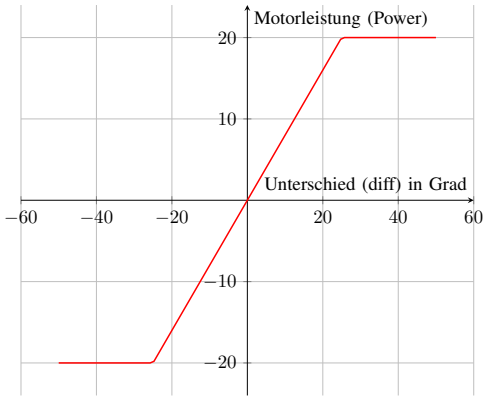

Abbildung 2. Diagramm der Motorleistung in Abhängigkeit von der Entfernung in grad.

DOI: 10.24352/UB.OVGU-2025-015

B. Genauigkeit der Sensoren

Wie in II-A erwähnt, gibt es auch Probleme mit den Sensoren. Die LEGO-Gyrosensoren haben das Problem, dass sie nur eine Rotationsachse wahrnehmen und anfangen können zu driften. Das größte Problem ist jedoch die Genauigkeit der Gradzahl, die sie ausgeben, da diese nur ganze Zahlen sein können. Für diese Probleme wurden jedoch schnell Lösungen gefunden. Wenn es nur eine Achse gibt, muss man einfach für jede Achse einen Sensor verwenden. Das Driften wurde durch die Bewegung des Sensors beim Einschalten verursacht, daher musste dieser nur stillgehalten werden. Für das letzte Problem wurden die Gyrosensoren auf Drehgeschwindigkeit umgeschaltet. Dies ermöglicht eine feinere und schnellere Anpassung der Motoren, um den Arm in Echtzeit zu steuern, ohne dass große Fehler durch Trägheit oder ungenaue Winkelmessungen auftreten.

C. Vektoren um Vektoren rotieren

Man rotiert Vektoren, um die Orientierung des Roboterarms basierend auf den Gyrosensorwerten zu berechnen. Da der Arm sich im 3D-Raum bewegt und die Sensoren die Bewegung entlang einer Achse messen, übersetzt man die Gyroskopdaten in Vektorbewegungen, um die Ausrichtung des Arms zu bestimmen. Diese Rotation ist notwendig, um die komplexen Bewegungen des Arms korrekt nachzuvollziehen und Fehler durch ungenaue Sensorwerte zu minimieren. Die Rotation der Vektoren ermöglicht es uns, die aktuelle Orientierung des Arms in Echtzeit zu berechnen und die Motoren präzise anzupassen. Die rotateVectors-Funktion verarbeitet die Gyroskopdaten, um die Armbewegung exakt zu steuern. Das ist in Abb. 3 zu sehen. Für die Transformation der Vektoren wird die „Rodrigues' Rotation Formula" verwendet, die im Abschnitt III-C gezeigt und erklärt wird. Ohne diese Formel wäre die Umsetzung nicht möglich gewesen.

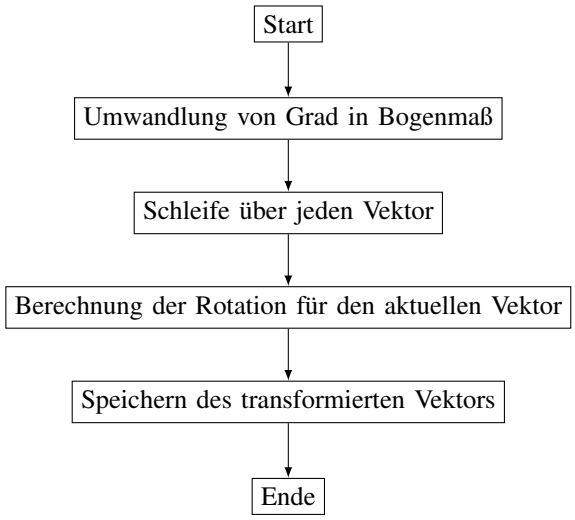

Abbildung 3. Programmablaufplan der Vektor Transformation

III. Umsetztung

A. Funktionsweise im Ganzen

Im Code werden die Werte der Gyrosensoren, die sich im Controllstick befinden, genutzt, um die Drehbewegung des Roboterarms in Echtzeit zu messen. Diese Messwerte werden in Rotationswinkel umgerechnet und dienen dazu, die aktuelle Ausrichtung des Arms zu bestimmen. Die Motoren werden dann entsprechend den berechneten Winkelabweichungen angepasst, um die gewünschte Position zu erreichen. Die Funktion SpeedControl sorgt dabei für eine präzise Steuerung der Motorleistung, indem sie den Unterschied zwischen dem aktuellen und dem Zielwinkel berücksichtigt.

B. Programmablaufplan

In Abb. 4 wird die Hauptschleife des Programms dargestellt und zeigt auch, was passiert, wenn die Abbruchbedingung erfüllt ist. Zusätzlich wird in diesem Diagramm visualisiert, welche spezifischen Schritte unternommen werden, um den Ablauf des Programms zu steuern und zu beenden.

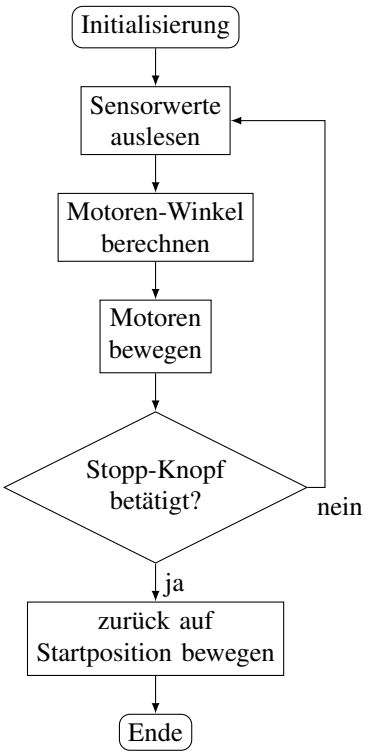

Abbildung 4. Programmablaufplan MATLAB-Skript

C. Gleichungen

Im Folgenden wird die Gleichung gezeigt und erklärt, die diesem Roboter ermöglicht haben seine Aufgaben auszuführen.

$$\mathbf{v}_{\text{neu}} = \mathbf{v}\cos\theta + (\mathbf{k}\times\mathbf{v})\sin\theta + \mathbf{k}(\mathbf{k}\cdot\mathbf{v})(1-\cos\theta)$$

hierbei gilt
$\mathbf{v}$ ist der ursprüngliche Vektor.
$\mathbf{k}$ ist der Einheitsvektor, der die Rotationsachse beschreibt.
θ ist der Winkel, um den der Vektor rotiert.
Die Formel ist unter der Quelle [1] zu finden

IV. Ergebnisdiskussion

Am Ende konnte die Idee erfolgreich umgesetzt werden und der Roboterarm funktionierte mit der gewünschten Funktionalität. Zwar brachten die LEGO Mindstorms-Bauteile einige Einschränkungen hinsichtlich Präzision und Rechenleistung mit sich, doch ließen sich diese durch kreative Lösungsansätze gut überwinden. Mit sorgfältiger Planung und der Anpassung der vorhandenen Ressourcen konnte die Leistung des Roboterarms optimiert werden.

In Abb. 5 ist ein früher Entwurf des Roboters zu sehen, der als Ausgangspunkt für die weitere Entwicklung dient. Dieser Entwurf stellt die Grundlage für den fertigen Roboter dar, dessen endgültige Version in Abb. 1 gezeigt wird. Während der Entwurf noch einige grobe Aspekte und Anpassungen aufwies, wurde er im Laufe der Zeit weiter verfeinert und optimiert, sodass er schließlich die gewünschte Funktionalität und Präzision erreichte.

Für den Control-Stick wurden zwei Varianten entwickelt, die in Abb. 6 zu sehen sind: eine mit einem großen Control-Stick und eine mit zwei kleinen Sticks. In Variante eins ist der Control Stick frei beweglich und enthält alle drei Gyrosensoren. In Variante zwei hingegen ist jeder Stick nur mit einem Gyrosensor ausgestattet, der jeweils nur eine Achse kontrolliert, ähnlich wie bei Controllern für Drohnen oder RC-Autos.

Variante zwei brachte jedoch Probleme mit sich, wie etwa eine mangelnde Kontrolle durch nicht wahrgenommene Bewegungen oder das Fehlen der dritten Achse, weshalb wir uns letztlich für Variante eins entschieden. Es zeigte sich, dass durch die richtige Wahl der Sensoren und Steuerungsstrategien selbst mit den begrenzten Möglichkeiten der LEGO Mindstorms-Technologie eine sehr leistungsfähige und präzise Steuerung des Arms möglich war. Dies beweist, dass auch innerhalb der Einschränkungen eines solchen Systems innovative Lösungen gefunden werden können, die diese überwinden.

V. Zusammenfassung und Fazit

Am Ende wurde ein voll funktionsfähiger Roboterarm entwickelt, der genau die beabsichtigte Aufgabe erfüllt. Es wurden alle Hürden überwunden und clever gelöst.
Um den Roboterarm zu erweitern, gibt es noch einige Ideen. Wenn man den Arm mit einer Webcam und Rädern ausstatten, könnte er einen Raum nach etwas absuchen, und der Control-Stick würde wegfallen. Eine weitere mögliche Weiterentwicklung wäre die Integration eines LiDAR-Sensors, um den Arm in die Lage zu versetzen, die Umgebung präzise zu scannen und Hindernisse zu erkennen. Dadurch könnte der Roboter autonom navigieren und Aufgaben in komplexeren Umgebungen ausführen. Allerdings wäre dies aufgrund der Limitationen der LEGO-Mindstorms-Bauteile, wie beispielsweise der begrenzten Rechenleistung und der fehlenden Sensoren für präzisere Bewegungen und Umwelterkennung, momentan nicht vollständig umsetzbar. Die beste Möglichkeit, den LiDAR-Sensor zu imitieren, wäre die Verwendung eines Ultraschallsensors, jedoch wäre auch dieser für komplexe Aufgaben aufgrund der eingeschränkten Präzision und Reichweite möglicherweise nicht ausreichend.

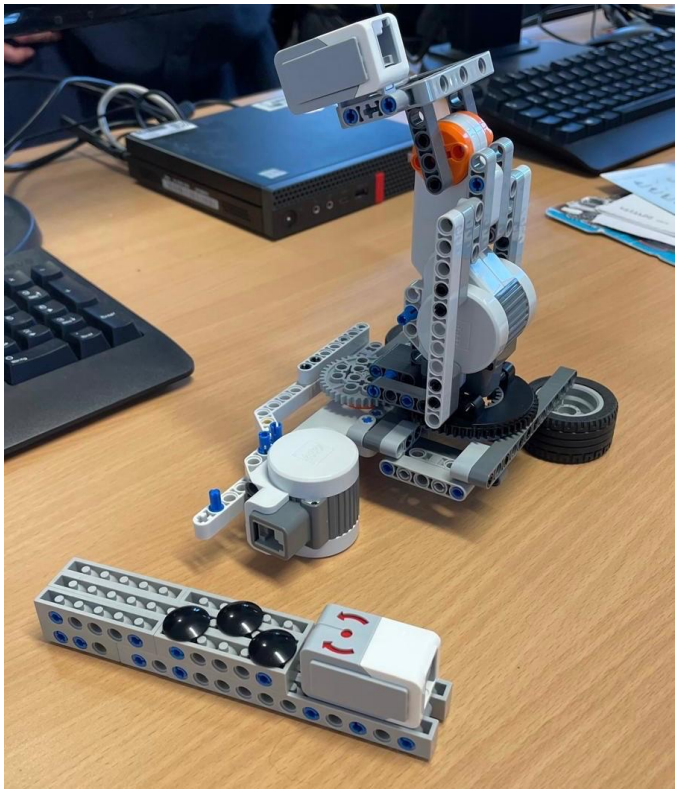

Abbildung 5. Früher Entwurf des Roboters und Control-Sticks

Abbildung 6. Zwei Varianten des Control-Sticks: Links Variante 2 und Rechts Variante 1

Literaturverzeichnis

[1] Wikipedia, the free encyclopedia: *Rodrigues' rotation formula*. https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula. Version: Februar 2025