Follow-Me-Robot

Xiyue Xu, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Ziel des Projektseminars Elektrotechnik/Informationstechnik (LEGO Mindstroms) 2025 war es, eine Objektverfolgungsfunktion mit Hilfe eines Roboters aus einem LEGO Baukasten zu realisieren. Die Steuerung des Roboters erfolgt über die Steuerungskomponente NXT, die Datenerfassung über angeschlossene Sensoren und die Programmierung über MATLAB.**

***Schlagwörter*—LEGO Mindstroms, MATLAB, Verfolgen, Abstandsensor, Projektseminar**

I. Einleitung

DIE vorliegende Studie beschäftigt sich mit der Entwicklung von Robotersystemen unter Verwendung von LEGO-Bauelementen und der NXT-Steuerungseinheit. Im Forschungsprozess wurden technische Herausforderungen in Bezug auf Roboterstrukturen, Bewegungsmechanismen und Funktionsimplementierung systematisch analysiert. Unser Ziel bestand in der Realisierung eines innovativen LEGO-Roboters mit praktischer Relevanz, der durch kreative Funktionalität und technische Originalität überzeugt.

II. Vorbetrachtungen

In den Vorbetrachtungen werden zunächst bereits existierende Lösungen kurz vorgestellt und mit relevanten Literaturquellen untermauert. Darüber hinaus werden die für die eigene Lösungsfindung angewandten Verfahren erläutert.

A. Forschungszusammenhang

Lego-Robotik nimmt als interdisziplinäres Forschungsfeld eine wichtige Rolle in den Bereichen Bildung und technologische Entwicklung ein.Im Bildungssektor ermöglichen LEGO-Roboter durch ihre niedrigschwellige Zugänglichkeit eine praxisorientierte Vermittlung von MINT-Disziplinen (Mathematik, Informatik, Naturwissenschaften, Technik). Sie fördern nicht nur logisches Denken und kreative Problemlösungskompetenzen, sondern demokratisieren gleichzeitig den Zugang zu robotischen Grundprinzipien – insbesondere durch visuelle Programmierschnittstellen, die auch jüngere Zielgruppen einbeziehen. Parallel dazu dienen LEGO-basierte Prototypen in der Forschungs- und Entwicklungsphase als agiles Testmedium: Forschende können algorithmische Ansätze oder Steuerungskonzepte kosteneffizient und iterativ validieren, ohne auf komplexe Hardware-Infrastrukturen angewiesen zu sein.

B. Forschungsmotivation

Vor dem Hintergrund der LEGO-Robotik-Forschung zielt dieses Projekt auf die Entwicklung eines praxisrelevanten Folgeroboters ab, der Automatisierung, Informatik, Konstruktion und Sensortechnik interdisziplinär vereint. Durch die Integration von Sensorik, Steuerungsalgorithmen und modularer Mechanik wird ein iterativer Lernzyklus von der Konzeption bis zur Optimierung angestrebt. Das Projekt verkörpert die Kernprinzipien der LEGO-Robotik: die Transformation komplexer technologischer Herausforderungen in modular erweiterbare Systeme zur Förderung akademischer und praktischer Kompetenzen.

C. Forschungsfragen

Diese Untersuchung konzentriert sich auf drei zentrale Fragestellungen: Erstens die algorithmische Umsetzung einer Steuerungslogik zur Einhaltung eines angemessenen Abstands zum Zielobjekt. Zweitens die physikalische Gestaltung und Implementierung von Lenkmechanismen sowie deren präzise zeitliche Steuerung während der Bewegung. Drittens die strukturelle Optimierung des Roboters, um eine stabile Balance zu gewährleisten und gleichzeitig ästhetische Aspekte zu berücksichtigen. Diese Fragestellungen vereinen Aspekte der Sensorik, Regelungstechnik und mechanischen Konstruktion innerhalb eines interdisziplinären Forschungsrahmens.

D. Forschungslimitationen

Die Studie nutzt LEGO-Bildungssets mit der NXT-Steuereinheit (32-bit ARM7-Mikrocontroller, 48 MHz Taktfrequenz) und einfachen Ultraschallsensoren (Messbereich 0 bis 255 cm, ±3 cm Genauigkeit). Aufgrund der hardwarebedingten Grenzen – begrenzte Rechenleistung (256 kB Flash-Speicher) und verzögerte Sensordatenverarbeitung – sind präzise Echtzeit-Objektverfolgung oder komplexe Regelungsalgorithmen nicht umsetzbar. Die Funktionalität bleibt auf Basisanwendungen mit reduzierter Reaktionsgeschwindigkeit beschränkt. Die in diesem Abschnitt verwendeten Daten beziehen sich auf Angaben aus [1].

III. Konkrete Umsetzung

In diesem Abschnitt werden die konzeptionellen Grundlagen und die praktische Umsetzung in drei Kernbereichen erläutert: Bewegungssteuerung, Code-Implementierung und Struktureller Aufbau.

A. Modulbeschreibung

Wie in Abbildung 1 zu sehen ist, handelt es sich bei dem gezeigten Bauteil um einen LEGO-Servomotor, der zur Steuerung von Drehbewegungen eingesetzt wird.

DOI: 10.24352/UB.OVGU-2025-011

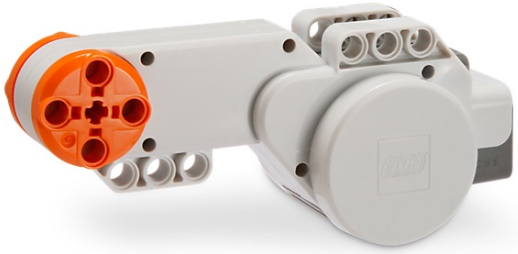

Abbildung 1: LEGO-Motor

Wie in Abbildung 2 dargestellt, erfasst der Ultraschallsensor Objektdistanzen im Bereich von $6\,\mathrm{cm}$ bis $255\,\mathrm{cm}$ und dient als zentrale Komponente der Abstandsregelung.

Abbildung 2: LEGO-Ultraschallsensor

Wie in Abbildung 3 zu sehen ist, Die Kernkomponenten des LEGO NXT umfassen die NXT-Steuereinheit (mit ARM7-Prozessor, Bluetooth-Unterstützung und Anschlüssen für Sensoren und Aktoren), verschiedene Sensoren (wie Tast-, Ultraschall-, Geräusch- und Lichtsensoren) sowie Servomotoren.

Abbildung 3: LEGO - NXT

B. Bewegungssteuerung

IV. Bewegungssteuerung

Das Regelungssystem folgt einem sequenziellen Ablauf: Nach dem Start aktiviert es die Sensoren zur Umgebungserfassung. Durch den Vergleich der Sensordaten (Links-/Rechtssensor) wird die Richtungssteuerung bestimmt: Bei einer Differenz von <-1 erfolgt eine Linkskurve, bei >1 eine Rechtskurve. Anschließend wird der gemessene Abstand mit dem Sollwert verglichen: Werte von <-3 lösen Rückwärtsfahrt aus, Werte von >3 Vorwärtsfahrt. Andernfalls stoppt das System. Dieser Algorithmus gewährleistet eine grundlegende Objektverfolgung unter den hardwarebedingten Echtzeitgrenzen des NXT-Systems.

In Abbildung 4 wird ein Programmablaufplan gezeigt.

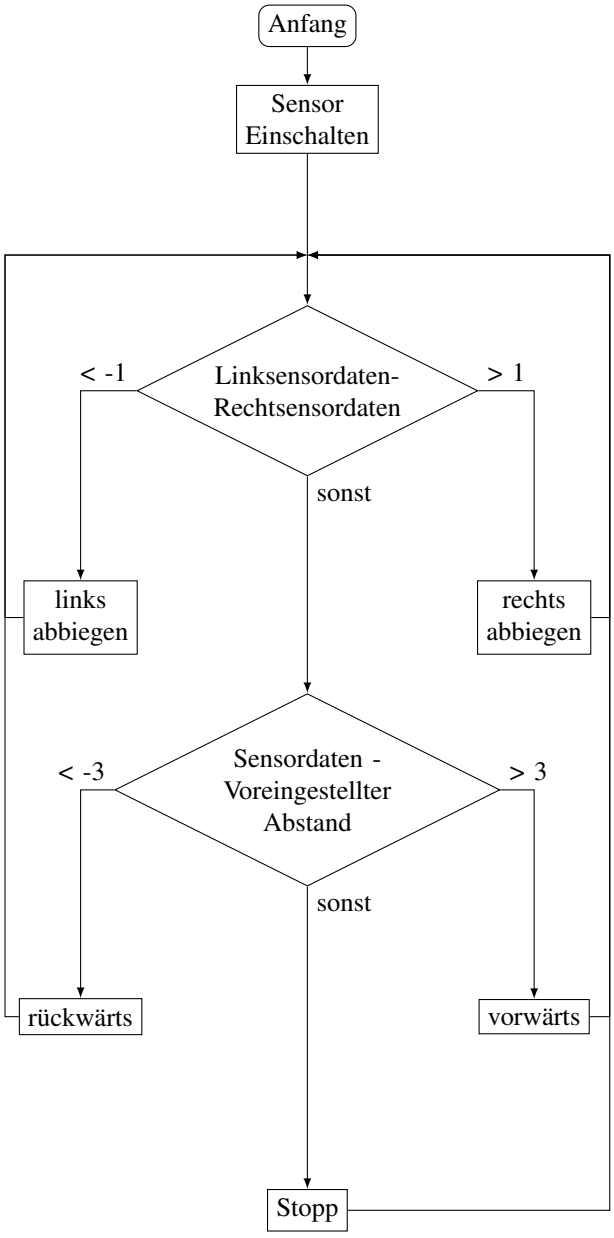

Abbildung 4: Programmablaufplan von Follow Me Robot

A. Code-Implementierung

Die Steuerungslogik wird mittels eines Programmablaufplans strukturiert und in MATLAB implementiert. Zunächst werden zentrale Parameter definiert: die Zielentfernung (`targetDistance`), die Entfernungstoleranz (`toleranz`), der Mindestabstand für Lenkmanöver (`AbbiegenDistance`), die Basisgeschwindigkeit (`motorPower`) sowie die Lenkgeschwindigkeit (`AbbiegenPower`). Die Ultraschallsensoren liefern Echtzeitdaten (`currentDistance_1`, `currentDistance_2`), aus deren Differenz (`Diff`) die Lenkentscheidungen abgeleitet werden.

Die Lenklogik aktiviert sich, wenn der absolute Wert der Sensordifferenz ($|$`Diff`$|$) den Schwellenwert `AbbiegenDistance` überschreitet. Überwiegt der linke Sensormesswert, wird der linke Motor mit `AbbiegenPower` vorwärts und der rechte rückwärts angesteuert – im umgekehrten Fall erfolgt eine spiegelverkehrte Steuerung. Liegt `Diff` innerhalb des Toleranzbereichs, vergleicht das System die gemessene Gesamtdistanz mit `targetDistance`. Überschreitet der Wert die Toleranzgrenze (`targetDistance` $\pm$ `toleranz`), fährt der Roboter vorwärts oder rückwärts; andernfalls stoppt er.

Ein Befehlszyklus von 0,5 Sekunden balanciert Reaktionsgeschwindigkeit und Systemstabilität, angepasst an die begrenzte Rechenleistung des NXT-Systems. Parameter wie `toleranz` werden objektspezifisch kalibriert – beispielsweise erfordert die Verfolgung von Personen höhere Toleranzen aufgrund ungleichmäßiger Oberflächenkonturen.

Der vollständige MATLAB-Code befindet sich im Anhang.

B. Struktureller Aufbau

Der Roboteraufbau orientiert sich an Funktionalität und Implementierungseffizienz. Für die Differentiallenkung sind zwei separat angetriebene Räder mit variabler Geschwindigkeit erforderlich, weshalb die Motoren seitlich angebracht wurden. Die Ultraschallsensoren wurden – basierend auf Vorversuchen – möglichst weit außen positioniert, um präzise Seitendistanzmessungen zu ermöglichen. Zur Gewichtsoptimierung wurde die NXT-Steuereinheit flach im Chassis platziert, wodurch eine stabile Gewichtsverteilung erreicht und Lenkmanöver begünstigt werden. Das folgende Abbildung 5 zeigt das finale Design des Follow-Me-Roboters

Abbildung 5: Aufbau des Follow-Me-Robot

V. Ergebnisdiskussion

A. Ergebnis

Damit hat der Follow-Me-Roboter sein Ziel erreicht, nämlich das Objekt zu verfolgen. Dies gilt insbesondere für flache Objekte, bei denen der Roboter in der Lage ist, Entfernungen schnell zu erkennen und Befehle zum Drehen und Folgen auszuführen. Auch das Verfolgen von Personen ist möglich, sofern die Fahrsgeschwindigkeit und die Entfernungstoleranz relativ groß gewählt werden.

B. Probleme – Beschränkungen des Ultraschallsensors

Hauptproblem: Aufgrund der Einschränkungen der Ultraschallsensoren kann der Follow-Me-Roboter nicht präzise erkennen, welchem Objekt er folgen soll. Er ist lediglich in der Lage, einem Objekt zu folgen, das sich direkt vor ihm befindet.

Erkennungsbereich: Die Perspektive der Sensoren begrenzt die Erkennungsgenauigkeit. Der Abstand zwischen den beiden Ultraschallsensoren definiert den Bereich, in dem der Roboter ein Objekt korrekt erfassen kann. Folglich darf sich das Zielobjekt nicht zu schnell bewegen, da der Roboter sonst das Ziel verliert. Zudem können unregelmäßig geformte Objekte, wie beispielsweise die Fußseiten einer Person, nicht zuverlässig erkannt werden, wenn sie sich im Erkennungsbereich der Sensoren befinden.

VI. Zusammenfassung und Fazit

Insgesamt hat das Projekt seine erklärten Ziele erreicht. Gleichzeitig kann es als Einstiegspunkt für das Verständnis der Autofollowing-Technologie dienen. Gleichzeitig gibt es noch weitere Probleme zu lösen und weitere Möglichkeiten zu erforschen.

VII. ANHANG

```
% Abbiegen Funkion
while true
% 1. Abstandswerte von beiden Sensoren lesen
    currentDistance_1 = GetUltrasonic(SENSOR_2);
    currentDistance_2 = GetUltrasonic(SENSOR_1);
    fprintf(['[Zustand] Linker Abstand: %.2f cm,
    Rechter Abstand:' ' %.2f cm\n'],
    currentDistance_1, currentDistance_2);

% 2. Entscheidung basierend auf den Abstandswerten
if abs(currentDistance_1 - currentDistance_2) >
       AbbiegenDistance

% 2.1 D_1 > D_2 nach rechts
if currentDistance_1 > currentDistance_2
% Drehe nach rechts
    motorB.Power = AbbiegenPower;
    motorA.Power = -AbbiegenPower;
    motorB.SendToNXT();
    motorA.SendToNXT();
    fprintf('[Bewegung] Drehe nach rechts\n');
else
    % Drehe nach links
    motorB.Power = -AbbiegenPower;
    motorA.Power = AbbiegenPower;
    motorB.SendToNXT();
    motorA.SendToNXT();
    fprintf('[Bewegung] Drehe nach links\n');
end

    % Verfolgen Funktion (je halbe Sekunde)

    else
    % 1. GetDistance
    currentDistance_1 = GetUltrasonic(SENSOR_2);
    currentDistance_2 = GetUltrasonic(SENSOR_1);

    % 2. Motor Steuerung
    if currentDistance_1 > (targetDistance +
                tolerance)
        % 1) forword
        motorB.Power = motorPower;
        motorB.SendToNXT();
        motorA.Power = motorPower;
        motorA.SendToNXT();
        fprintf('[Bewegung] Geht nach vorne\n');

    elseif currentDistance_1 < (targetDistance
        - tolerance)
        % 2) back
        motorB.Power = -motorPower;
        motorB.SendToNXT();
        motorA.Power = -motorPower;
        motorA.SendToNXT();
        fprintf('[Bewegung] Geht zurück\n');

    else
    % 3) Stopp
    motorB.Power = stoppPower;
    motorB.SendToNXT();
    motorA.Power = stoppPower;
    motorA.SendToNXT();
    fprintf('[Bewegung] Stopp\n');
end
```

LITERATURVERZEICHNIS

[1] THE LEGO GROUP: *Lego Mindstorms NXT Benutzerhandbuch*. Version 2.0. Billund, Dänemark: The Lego Group, September 2006. https://de.wikipedia.org/wiki/Lego_Mindstorms_NXT. – Offizielles Benutzerhandbuch