Follow-Me Robot

Mingze Ma, 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

BEI selbstfolgenden Robotern handelt es sich um intelligente Geräte, die in der Logistik, im Lagerwesen, im Einzelhandel und im Haushalt weit verbreitet sind und durch autonome Aktionen und Objekterkennungstechnologien die Verfolgung von Personen oder Objekten in Echtzeit ermöglichen. Mit der rasanten Entwicklung der Automatisierungstechnik haben diese Roboter ein großes Potenzial zur Verbesserung der Effizienz und zur Verringerung des Personalaufwands. Auch durch die Integration mit anderen Technologien ergeben sich sehr viele Möglichkeiten.

Gleichzeitig ist diese Technologie aber nicht unerreichbar. LEGO ist ein berühmtes Spielzeug, aber im heutigen digitalen Zeitalter können einfache Roboter aus LEGO auch Funktionen ausführen, die schwierig klingen. In diesem Artikel wird zum Beispiel gezeigt, wie die Auto-Following-Technologie mit LEGO-Bauteilen und einfacher Programmierung realisiert werden kann. Dies ist eine relativ einfache Methode, die als Einführung genutzt werden kann, um Menschen zu inspirieren, wie diese Technologie realisiert werden kann und welche Möglichkeiten für die Zukunft bestehen.

II. Vorbetrachtungen

In diesem Abschnitt werden der Zweck des Projekts, die wichtigsten im Projekt verwendeten Komponenten und das für die Programmierung erforderliche MATLAB vorgestellt.

A. Zweck des Projekts

Das Ziel von Tracking-Robotern besteht in erster Linie darin, sich selbständig bewegen zu können. Zweitens müssen sie in der Lage sein, relevante Daten über das verfolgte Objekt zu erhalten, z. B. die Entfernung und die Orientierung. Schließlich werden durch Programm die erfassten Daten verarbeitet, um Befehle zur Verfolgung des Objekts zu generieren. Die dafür erforderliche Komponenten und Software wird im Folgenden beschrieben.

DOI: 10.24352/UB.OVGU-2025-010

B. Erforderliche Komponenten

1) Motor: Die in Abbildung 1 gezeigten Motoren werden im Projekt verwendet, um die Antriebsfunktion zu realisieren. Gleichzeitig ermöglicht der Geschwindigkeitsunterschied zwischen den beiden Motoren das Abbiegen des Roboters.

Abbildung 1: LEGO-Motor [1]

2) Ultraschallsensor: Um das zweite Ziel zu erreichen, nämlich den Abstand zwischen dem Roboter und dem zu messenden Objekt zu ermitteln, ist ein Ultraschallsensor erforderlich, wie in Abbildung 2 dargestellt.

Abbildung 2: LEGO-Ultraschallsensor [2]

C. MATLAB

Um die Komponenten zu steuern und die von den Sensoren erhaltenen Daten zu verarbeiten, ist es notwendig, MATLAB zur Programmierung zu verwenden. Zunächst ist es notwendig, die grundlegende Syntax der MATLAB-Programmiersprache, die Variablendeklarationen und die grundlegenden Funktionsoperationen zu beherrschen. Noch wichtiger ist, dass man Funktionen schreiben und aufrufen kann, damit ist es möglich, den Motor zu steuern und auch die vom Ultraschallsensor gelieferten Daten zu betrachten und zu verarbeiten. Auf dieser Grundlage wird das Programm geschrieben, um die genannten Ziele zu erreichen.

III. Hauptteil

Im Hauptteil werden das Konzept und die Realisierung erläutert.

A. Aufbau

Wie in Abbildung 3 gezeigt, das ganze Projekt besteht aus einer NXT-Box, zwei Motoren, zwei Ultraschallsensor und einige LEGO Beuteile.

Aus einer NXT-Box wurde ein Rahmen gefertigt, der als Hauptkörper des Roboters diente. Zwei Motoren wurden auf beiden Seiten des vorderen Endes des Rahmens montiert, um als Energiequelle für die Fahrenfunktion des Roboters zu dienen. Die Abbiegenfunktion wird durch den Unterschied in der Drehgeschwindigkeit zwischen den beiden Rädern erreicht. Am hinteren Ende ist nur ein nicht motorisiertes Rad angebracht, um den Hauptrahmen zu stützen. Der Vorteil von nur einem Hinterrad ist, dass es dem Verhaltensmuster der Differenzial-lenkung besser entspricht und den Reibungswiderstand beim Abbiegen vermeidet.

Eine weitere Überlegung für den Hauptrahmen besteht darin, den Schwerpunkt des Roboters zu senken, indem die NXT-Box flach in der Mitte des Hauptrahmens platziert wird. Der unmittelbare Vorteil einer gleichmäßigen Verteilung des Hauptgewichts (d. h. der NXT-Box) besteht darin, dass die Stabilität der Struktur optimiert wird. Dies verhindert eine Überlastung der Komponenten, die den Hauptrahmen mit den Motoren verbinden, was eine präzise Roboterbewegung gewährleistet.

Zwei Ultraschallsensoren sind oberhalb der Vorderseite des Hauptrahmens angebracht, um den Abstand zum zu verfolgenden Objekt zu messen. Der Abstand zwischen den beiden Ultraschallsensoren sollte so groß wie möglich sein, um einen möglichst großen Messbereich zu erhalten. Je größer der Messbereich ist, desto genauer ist die Erkennung, und es werden auch Situationen vermieden, in denen das richtige Objekt nicht verfolgt werden kann.

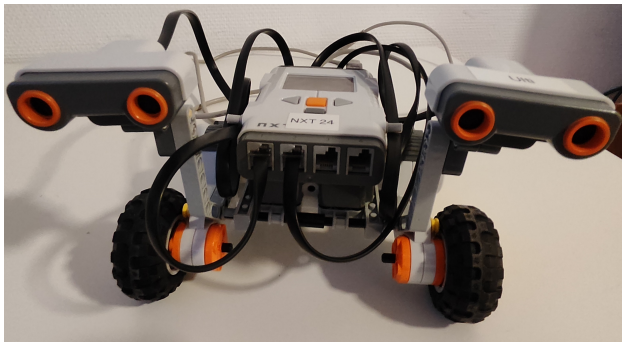

Abbildung 3: ursprünglicher Aufbau des Follow-Me Robot

B. Programmablaufplan

In Abbildung 4 wird ein Programmablaufplan gezeigt. Die Hauptlogik der Follow Me Roboters lässt sich zunächst nachvollziehen.

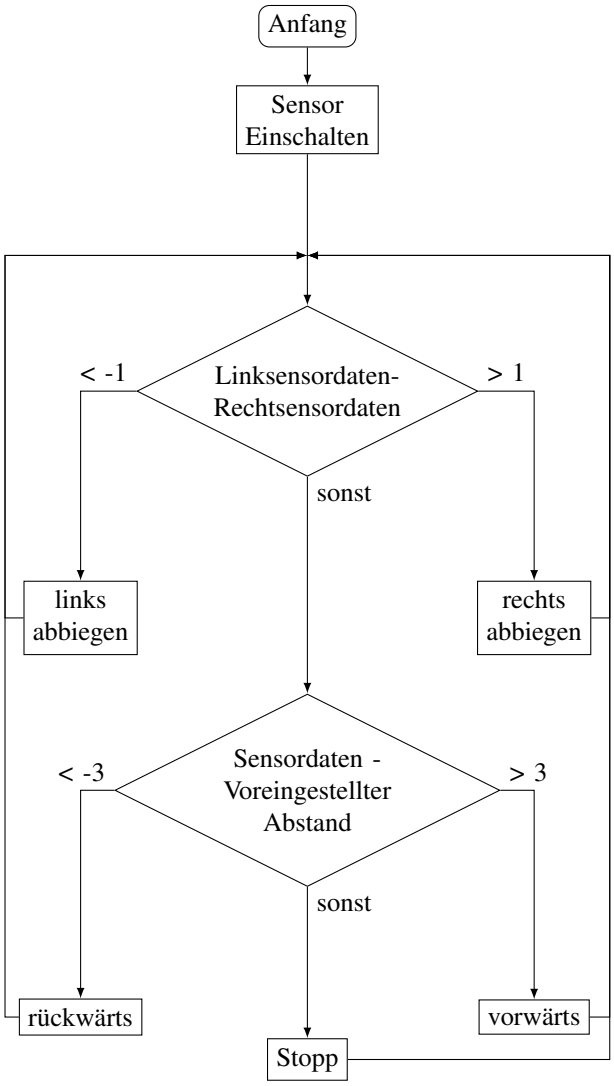

Abbildung 4: Programmablaufplan von Follow Me Robot

C. MATLAB-Programm

Mit Hilfe von Programmablaufplan lässt sich die Logik der Verfolgung von Robotern gut nachvollzogen werden. Als Nächstes muss der Roboter mit MATLAB programmiert werden, um ihn so zu steuern, dass er die gewünschte Funktion erfüllt.

1) Parameter: Zunächst müssen Variablen definiert werden, beispielsweise Zielentfernung (`targetDistance`), Entfernungstoleranz (`toleranz`), Abbiegenentfernung (`AbbiegenDistance`), Fahrgeschwindigkeit (`motorPower`), Abbiegengeschwindigkeit (`AbbiegenPower`) und vor allem die vom Ultraschallsensor in Echtzeit erfassten Entfernungsdaten((`currentDistance_1`) und (`currentDistance_2`)). Daraus kann der Abstandsunterschied (`Diff`) zwischen der linken und rechten Seite berechnet werden, um die Abbiegenanweisungen zu erhalten. Liegt die Differenz innerhalb der Lenkabstandstoleranz, werden die Echtzeit-Abstandsdaten mit dem Sollabstand verglichen und berechnet, um die Solldifferenz zu ermitteln und festzustellen, ob eine Abbiegenanweisung

erforderlich ist.

Alle Parameter müssen mehrmals getestet werden, um einen geeigneten Wert zu ermitteln. Außerdem werden für verschiedene verfolgende Objekte unterschiedliche Parameter benötigt. Zum Beispiel erfordert der Parameter für die Verfolgung einer Person eine größere Entfernungstoleranz als bei der Verfolgung eines flachen Objekts, da die Kleidung relativ stärker gekrümmt ist.

2) Abbiegenanweisung: Die Abbiegenanweisung wird auf die folgende Weise erzeugt und ausgeführt. Wenn der absolute Wert der Differenz zwischen den beiden Entfernungsdaten `(abs(Diff))` größer ist als Summe von Abbiegenentfernung `(AbbiegenDistance)`, wird das Abbiegenprogramm eingegeben. Wenn die Entfernungsdaten der linken Seite größer als die der rechten Seite ist, wird die Geschwindigkeit des linken Motors auf die Abbiegengeschwindigkeit `(AbbiegenPower)` und die der rechten Seite auf den negativen Wert der Abbiegengeschwindigkeit eingestellt. Der umgekehrte Fall trifft ebenfalls zu.

Ist der Abstandsunterschied `(Diff)` kleiner als der Abbiegenentfernung `(AbbiegenDistance)`, so ist folgendes Proframm anzuwenden.

3) Verfolgenanweisung: Die Verfolgenanweisung wird auf die folgende Weise erzeugt und ausgeführt. Wenn die Entfernungsdaten größer als die Summe von Zielentfernung `(targetDistance)` und Entfernungstoleranz `(toleranz)` ist, wird die Geschwindigkeit des beiden Motors auf die Fahrgeschwindigkeit `(motorPower)` eingestellt. Das Gegenteil setzt die Geschwindigkeit des beiden Motors auf einen negativen Wert der Fahrgeschwindigkeit **(motorPower)**.

Liegt die Zielentfernung innerhalb des Entfernungstoleranz `(toleranz)`, wird die Geschwindigkeit beider Motoren auf Null gesetzt, d. h. der Roboter wird zum Stillstand gebracht.

4) Befehlszykluszeit: Alle Befehle werden erteilt, um den Zyklus von Erfassung, Berechnung, Beurteilung und Ausführung von Befehlen wieder aufzunehmen, um eine kontinuierliche Folgefunktion zu erreichen. Nach dem Testen ist 0,5 Sekunden eine angepasste Befehlszykluszeit, die schneller reagieren kann und nicht zu viele Befehle gibt, für deren Ausführung der Roboter keine Zeit hat.

IV. 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. Problemm - Beschränkung von Ultraschallsensor

1) Einschränkungen bei der Zielerkennung: Das Hauptproblem besteht darin, dass der Follow Me Roboter aufgrund der Beschränkungen der Ultraschallsensoren nicht in der Lage ist, genau zu erkennen, welchem Objekt er folgen soll. Er kann nur einem Objekt folgen, das sich direkt vor ihm befindet.

2) Detektionsbereich und dynamische Verfolgungsgrenzen: Das zweite Problem ist eine Frage der Perspektive. Der Abstand zwischen zwei Ultraschallsensoren ist der Bereich, den der Roboter korrekt erkennen kann. Das bedeutet, dass sich das verfolgte Objekt nicht zu schnell bewegen darf, sonst verliert der Roboter das Ziel. Auch wenn das Objekt unregelmäßig ist, z. B. wenn der Erkennungsbereich der Ultraschallsensoren genau durch die Fußseiten einer Person geht, kann das verfolgte Objekt nicht korrekt erkannt werden.

V. 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.

VI. Anhang

Der Quellcode ist im Anhang zu finden [3].

```
% 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] *LEGO NXT Motor*. https://forum.arduino.cc/t/lego-nxt-motor-arduino-motor-shield/266470, 2023. – Accessed: 2023-10-10

[2] *LEGO - Ultraschallsensor*. https://www.amazon.de/LEGO-NXT-Ultraschallsensor-Ultrasonic-sensor/dp/B000PM8I8O, 2023. – Accessed: 2023-10-10

[3] UNIVERSITY, RWTH A.: *RWTH Mindstorms NXT Toolbox*. https://www.mindstorms.rwth-aachen.de/de/trac/wiki/Download4.07, 2007. – Accessed: 2023-10-10