Roboter zum Eierfärben

Kein Ersatz für den Osterhasen, aber für präzises und kreatives Eierfärben!

Korchunova Oleksandra, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Im Rahmen eines spannenden LEGO-Mindstorms-Projekts im Fachbereich Elektrotechnik und Informationstechnik im Jahr 2025 wurde ein Roboter entwickelt, der Ostereier automatisiert bemalen kann. Neben der konzeptionellen Planung erfolgte auch die Umsetzung eines präzisen Programmcodes, der eine flexible und effiziente Steuerung ermöglicht.**

***Schlagwörter*—LEGO-Mindstorms, MATLAB, Präsentation, Präzision, Roboter.**

I. Einleitung

Beim herkömmlichen Färben von Ostereiern bedarf es großer Geduld und einer ruhigen Hand, um gleichmäßige und ansprechende Muster zu erzielen. Häufig führt die manuelle Technik zu ungleichmäßigen Farbaufträgen, insbesondere bei der Verwendung mehrerer Farben.

Der vorgestellte Eierbemalungsroboter automatisiert diesen Prozess und ermöglicht so eine gleichmäßige sowie kreative Gestaltung. Durch die Minimierung von Farbfehlern und die Vereinfachung von Mustervariationen eröffnet das System neue Möglichkeiten für künstlerische Designs.

Die robuste Konstruktion des Roboters garantiert, dass das Ei sicher fixiert und gleichmäßig rotiert wird, während der bewegliche Pinselhalter auf einer vorgegebenen Bahn präzise Linien und Muster aufträgt. Optional können integrierte Sensoren den Farbauftrag überwachen und optimieren, was zu konsistenten und ästhetisch ansprechenden Ergebnissen führt.

II. Vorbetrachtungen

Im Vorfeld wurden grundlegende Aspekte für die Umsetzung des Projekts analysiert:

- Einsatz von LEGO Mindstorms als Plattform für die Roboterentwicklung.
- Verwendung von Motoren zur präzisen Steuerung der Farbapplikation.
- Integration unterschiedlicher Muster durch gezielte Bewegungsabläufe.
- Herausforderungen bei der fixen Positionierung der Eier.

A. Aufgaben des Roboters

Eine präzise Definition der vom Roboter zu erfüllenden Aufgaben ist essenziell, um zu verhindern, dass vielversprechende Konzepte in endlosen Optimierungsroutinen stecken bleiben. Deshalb wurde ein solides konzeptionelles Fundament geschaffen, auf dem später die Programmierung und der Aufbau mit LEGO-Komponenten basierten.

DOI: 10.24352/UB.OVGU-2025-018

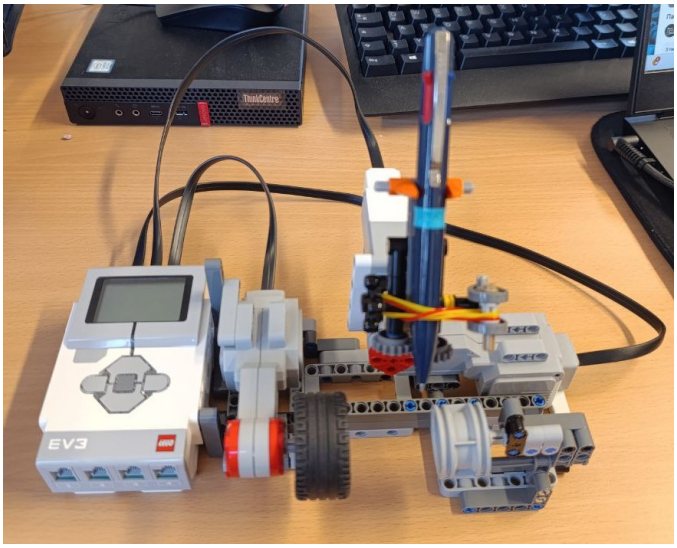

Abbildung 1. Roboter zum Eierfärben

B. Funktionsprinzip der LEGO-Komponenten

Vor dem Start der Konstruktion lag der Schwerpunkt auf der Untersuchung der Arbeitsweise der EV3-Motoren (siehe Abbildung 1). Besonderes Augenmerk wurde dabei auf den Umgang mit dem Pinsel und die Möglichkeit der Integration von Farbsensoren gelegt, da diese maßgeblich zur Qualität der Bemalung beitragen.

Diese Voruntersuchungen ermöglichten es, die Potenziale des LEGO-Systems realistisch einzuschätzen und lieferten gleichzeitig wichtige Erkenntnisse für die spätere Softwaresteuerung. So konnte ein stabiler und gut strukturierter Quellcode entwickelt werden.

C. Technische Umsetzung und Softwarekomponenten

Obwohl zahlreiche Designkonzepte entwickelt wurden, stießen erste Ansätze aufgrund der Einschränkungen in MATLAB und der mechanischen Herausforderungen beim Zusammenbau der LEGO-Elemente auf Schwierigkeiten. Diese Faktoren reduzierten zunächst die Funktionalität des Roboters. Zudem führte die Implementierung in MATLAB zu unerwarteten Problemen, die wiederholte Anpassungen des Codes erforderten.

III. Konstruktion

A. Design

Der Eierbemalungsroboter basiert auf dem EV3-Controller (siehe Abbildung 2), der als zentrale Steuereinheit fungiert und

Abbildung 2. EV3-Controller

Abbildung 3. Großer Motor

die unterschiedlichen Bewegungsabläufe koordiniert. Zur präzisen und gleichmäßigen Farbapplikation werden drei separate Motoren eingesetzt:

- Motor A: Zuständig für die Rotation des Eis [2]. Eine gleichmäßige Drehung sorgt für einen durchgängigen Farbauftrag ohne unregelmäßige Verläufe. Geschwindigkeit und Drehrichtung können an das gewünschte Muster angepasst werden (siehe Abbildung 3).
- Motor C: Steuert die horizontale Bewegung des Pinsels [2], wodurch unterschiedliche Muster erzeugt werden können (siehe Abbildung 3).
- Motor B: Regelt die vertikale Bewegung des Pinsels, was eine differenzierte Farbapplikation in variablen Höhen erlaubt. Je nach Design kann der Pinsel sanft oder mit mehr Druck aufgetragen werden, um unterschiedliche Intensitäten zu erzielen (siehe Abbildung 4).

B. Herausforderungen und Lösungsansätze

Zu Beginn des Projekts wurde der Roboter mit einer drehbaren Halterung für das Ei entworfen, um eine gleichmäßige Rotation und eine exakte Farbapplikation zu gewährleisten (siehe Abbildung 1). Allerdings traten in den ersten Tests diverse Probleme auf:

- Stabilität des Eis: Das Ei verrutschte während der Drehbewegung, was zu ungleichmäßigen Mustern führte. Dieses Problem konnte durch eine verbesserte Fixierung gelöst werden.
- Präzision der Motoren: Erste Tests zeigten ungenaue Bewegungen, wodurch die Muster unsauber wurden. Eine Feinjustierung der Steuerparameter in MATLAB erhöhte die Genauigkeit und ermöglichte eine gleichmäßige Farbverteilung.

Abbildung 4. Kleiner Motor

- Synchronisation der Motoren: Für die Optimierung der Muster war eine exakte Abstimmung der Motoren notwendig. Durch mehrere Testphasen und Anpassungen der Bewegungsabläufe konnte eine präzise Synchronisation erreicht werden (siehe Abbildung 5).

Diese Optimierungen führten zu einer verbesserten Stabilität der Konstruktion, reduzierten Vibrationen und einer insgesamt gesteigerten Leistung des Roboters.

C. Programmierung in MATLAB [1]

Für die Steuerung des Roboters wurden mehrere Malmodi implementiert, um eine vielseitige und präzise Gestaltung der Eier zu ermöglichen. Die Befehle werden über ein Kabel an den EV3-Controller übertragen, da Versuche mit Bluetooth aufgrund instabiler Verbindungen nicht erfolgreich waren.

Der Code ist modular aufgebaut und umfasst einzelne Blöcke, die die Bewegungen der drei Motoren steuern. Dabei werden Parameter wie Geschwindigkeit, Drehrichtung und Bewegungsdauer festgelegt. Zwei Hauptmodi wurden programmiert:

1) Gleichmäßige Farbapplikation: Das Ei rotiert kontinuierlich, während der Pinsel konstant Farbe aufträgt.
2) Gestreifte Farbgebung: Der Pinsel wird in definierten Intervallen abgesenkt und angehoben, um ein gestreiftes Muster zu erzeugen.

Die Steuerung erfolgt mithilfe einer `for`-Schleife, die die fortlaufende Bewegung der Motoren sicherstellt. Zudem wird eine `if`-Bedingung verwendet, um den Farbauftrag zu kontrollieren und im Fehlerfall den Betrieb zu unterbrechen, um eine mangelhafte Bemalung zu vermeiden.

Der modulare Aufbau des Quellcodes ermöglicht es, Muster flexibel anzupassen und zukünftige Funktionen problemlos zu integrieren.

```
motorEgg = motorB;
motorBrush = motorA;
motorMove = motorC;
```

```
disp('Wählen Sie einen Malmodus:');
disp('1 - Einheitliche Farbgebung');
disp('2 - Gestreifte Färbung');
mode = input('Geben Sie 1, 2 ein: ');
if mode == 1
disp('Gleichmäßiges
Färben beginnt...');
motorBrush.setProperties
('power',20,'limitValue', 145);
start(motorBrush);
pause(1);
for i = 1:50
powerValue = 57;
motorEgg.setProperties
('power',-25);
start(motorEgg);
motorMove.brakeMode='brake';
motorMove.setProperties
('power',30,'limitValue',
powerValue);
start(motorMove);
motorMove.waitFor;
motorMove.setProperties
('power',-30,'limitValue', powerValue);
start(motorMove);
motorMove.waitFor;
stop(motorEgg);
end
motorBrush.setProperties
('power',-20);
start(motorBrush);
stop(motorBrush);
stop(motorMove);
elseif mode == 2
for i = 1:13
motorEgg.setProperties
('power',-30);
start(motorEgg);
motorBrush.setProperties
('power',10,'limitValue', 140);
start(motorBrush);
pause(2.5);
stop(motorBrush);
motorBrush.setProperties
('power',-10);
start(motorBrush);
pause(0.5);
motorMove.setProperties
('power',10,'limitValue', 4);
start(motorMove);
pause(0.5);
stop(motorEgg);
end
stop(motorBrush);
stop(motorMove);
stop(motorEgg);
disp('Falsche Wahl!');
end
disp('Programm ist abgeschlossen.');
```

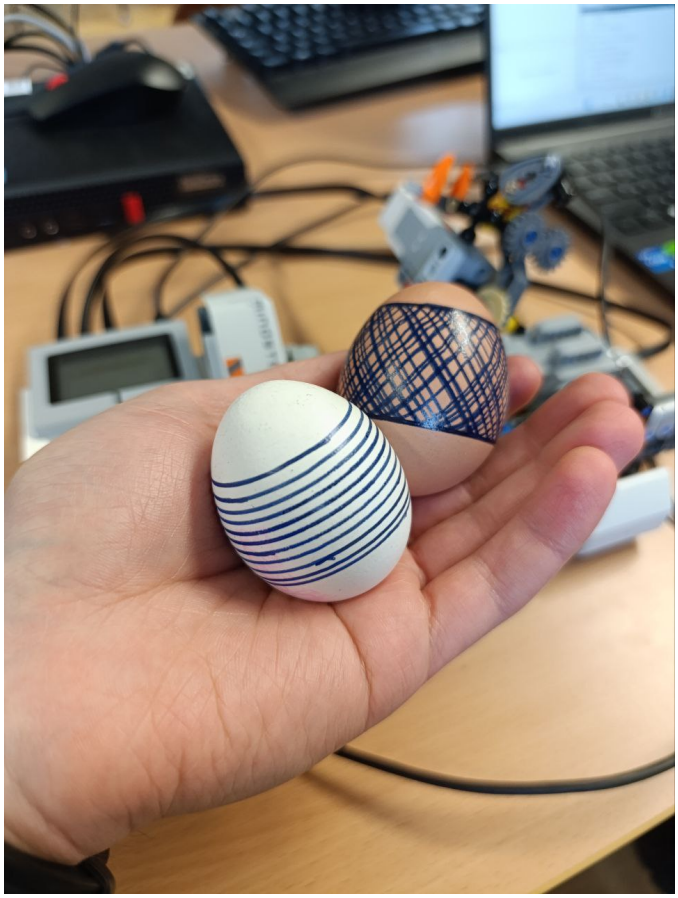

Abbildung 5. Bemalte Eier

IV. Diskussion der Ergebnisse

Nach zahlreichen Testläufen mit verschiedenen Mustern und Farbvarianten konnte bestätigt werden, dass der Roboter die vorgegebenen Anforderungen erfüllt. Die einzige Einschränkung lag in der kabelgebundenen Verbindung zum Laptop, die für die MATLAB-Steuerung notwendig ist. Dennoch arbeitete die Motorsteuerung zuverlässig und ermöglichte eine präzise Ausführung unterschiedlicher Maltechniken.

Insgesamt zeigte sich, dass der Roboter stabil und effizient arbeitet. Kleinere Software- und Mechanikanpassungen könnten zukünftig die Präzision der Farbapplikation weiter optimieren (siehe Abbildung 5).

V. Zusammenfassung und Fazit

Das Projekt erreichte nicht nur das Ziel, einen funktionierenden Roboter zu entwickeln, sondern lieferte auch wertvolle Einblicke in die Verbindung von Mechanik, Programmierung und Design. Die präzise Steuerung ermöglichte einen gleichmäßigen Farbauftrag, und künftige Erweiterungen könnten die Effizienz und Funktionalität weiter steigern.

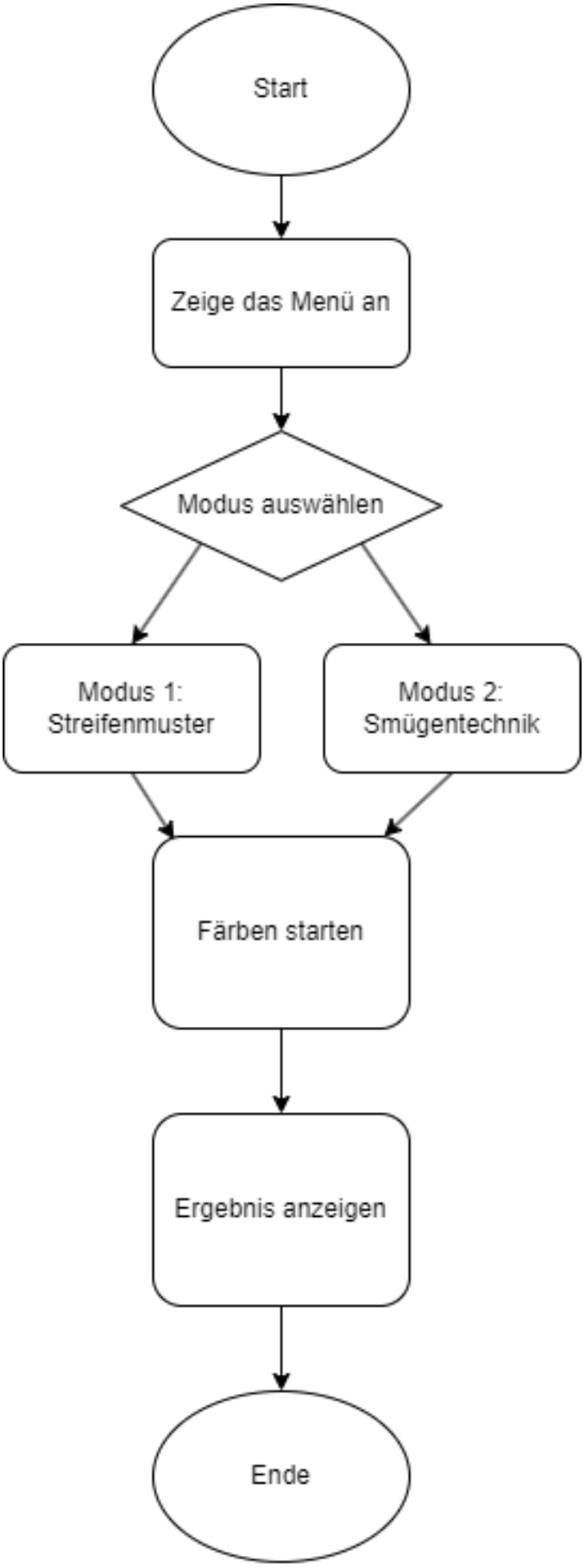

Abbildung 6. Programmablaufplan

LITERATURVERZEICHNIS

[1] MATHWORKS: *MATLAB*. Verfügbar unter: https://de.mathworks.com/products/matlab.html Version: 2024
[2] MINDSTORMS NXT: *NXT Motor*. Verfügbar unter: https://mindstormsnxt.blogspot.com/2006/08/closer-look-at-nxt-motors.html Version: 2006
[3] MATHWORKS: *MATLAB Syntax*. Verfügbar unter: https://de.mathworks.com/help/matlab/ref/function.html Version: 2024
[4] LEGO® MINDSTORMS® EV3: . Verfügbar unter: https://education.lego.com/en-us/product-resources/mindstorms-ev3/downloads/building-instructions/ Version: 2025
[5] *Lego Mindstorms EV3*. Verfügbar unter: https://www.academia.edu/42810872/ Jahr: 2020