DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

Realisierung eines Vier-Gewinnt Roboters

ggf. Untertitel mit ergänzenden Hinweisen

Studienarbeit T3_3200

Studiengang Elektrotechnik

Studienrichtung Automation

Duale Hochschule Baden-Württemberg Ravensburg, Campus Friedrichshafen

von

Simon Gschell / Patrik Peters

Abgabedatum:	6. Juli 2025
Bearbeitungszeitraum:	01.01.2025 - 31.06.2025
Matrikelnummer:	123 456
Kurs:	TEA22
Betreuerin / Betreuer:	Prof. Dr. ing Thorsten Kever

Erklärung

gemäß Ziffer 1.1.14 der Anlage 1 zu §§ 3, 4 und 5 der Studien- und Prüfungsordnung für die Bachelorstudiengänge im Studienbereich Technik der Dualen Hochschule Baden-Württemberg vom 29.09.2017 in der Fassung vom 24.07.2023.

Ich versichere hiermit, dass ich meine Studienarbeit T3_3200 mit dem Thema:

Realisierung eines Vier-Gewinnt Roboters

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Friedrichshafen, den 6. Juli 2025

Simon Gschell / Patrik Peters

Kurzfassung

Diese Studienarbeit wurde im Rahmen der sechsten Akademiephase angefertigt. Im ersten Teil der Arbeit wurden verschiedene Spieltheorien miteinander verglichen. Aufbauend auf den Erkenntnissen der ersten Studienarbeit, lag der Schwerpunkt diesmal in der praktischen Umsetzung eines Vier-Gewinnt-Roboters. Das Ziel dieser Arbeit bestand darin, einen Roboter zu entwickeln, der eigenständig Spielzüge beim Spiel „Vier gewinnt“ ausführen kann und somit in der Lage ist, gegen einen anderen Roboter anzutreten.

Für die Realisierung des Projekts wurde ein LEGO Spike Prime Set verwendet. Die Konstruktion des Roboters besteht aus verschiedenen zusammengesetzten LEGO-Bauelementen. Vereinzelt wurden auch Elemente selbst entworfen und mittels 3D-Drucker angefertigt. Die Steuerung und Überwachung der Aktoren, wie zum Beispiel der Motoren und Sensoren, erfolgt mithilfe eines LEGO Spike Hub. Dieser Mikrocontroller verarbeitet die eingehenden Sensordaten und steuert die Bewegungen des Roboters entsprechend der programmierten Logik.

Die Programmierung erfolgte in der Sprache MircoPython in der LEGO Spike App. Das Kernstück des Programms ist der Alpha-Beta-Algorithmus, mit ihm wird der nächste optimale Zug berechnet. Durch die Kombination aus mechanischer Konstruktion und programmierter Software entstand ein funktionsfähiger Prototyp, der die gestellten Anforderungen erfüllt und einen Spielzug eigenständig ausführen kann.

Abstract

This student research project was carried out during the sixth academy phase. In the first part of the project, various game theories were examined and compared. Building on the insights from that initial work, the focus this time was on the practical development of a Four-in-a-Row robot. The goal was to create a robot capable of making its own moves in the game "Connect Four,älowing it to compete against another robot.

The project was implemented using a LEGO Spike Prime set. The robot itself was built from a range of LEGO components, with some parts specially designed and produced using a 3D printer. The motors and sensors are managed by a LEGO Spike Hub, which acts as the robot's microcontroller. This hub processes sensor data and directs the robot's movements according to the programmed logic.

Programming was done in MicroPython using the LEGO Spike app. At the heart of the software is the alpha-beta algorithm, which determines the best possible move at each turn. By combining mechanical design with custom software, the project resulted in a working prototype that meets the requirements and can play the game autonomously.

Inhaltsverzeichnis

1 Einleitung

2 Grundlagen

In diesem Kapitel werden die zentralen Begriffe und Methoden ausführlich vorgestellt. So wird das notwendige Grundlagenwissen vermittelt, auf dem die weitere Ausarbeitung aufbaut.

2.1 Vier-Gewinnt

Das Spiel Vier-Gewinnt wird auf einem Spielfeldraster mit sechs Zeilen und sieben Spalten gespielt. Zum Spielbeginn erhält jeder Spieler 21 Spielchips, entweder Rote oder Gelbe. Ziel des Spiels ist es, möglichst schnell vier Chips der eigenen Farbe in eine Reihe zu bringen – waagrecht, senkrecht oder diagonal. Die Spieler werfen abwechselnd ihre Chips in das Spielfeld, bis entweder ein Spieler das Ziel erreicht oder alle 42 Felder belegt sind [Has20].

2.2 Alpha-Beta-Pruning-Algorithmus

Alpha-Beta-Pruning ist ein Verfahren, das bei Spielen wie Schach, Dame oder Vier Gewinnt eingesetzt wird, um den optimalen nächsten Zug zu bestimmen. Ziel des Algorithmus ist es, die Suche im Spielbaum effizienter zu machen. Im Unterschied zum Minimax-Algorithmus werden beim Alpha-Beta-Pruning gezielt Teilbäume weggelassen, die für das Endergebnis keine Rolle spielen. Während der Tiefensuche durch

den Spielbaum arbeitet der Algorithmus mit zwei Schrankenwerten, dem Alpha (α) und dem Beta (β). Zu Beginn wird Alpha auf $-\infty$ und Beta auf $+\infty$ gesetzt. An jedem MAX-Knoten wird das Maximum aus dem bisherigen Alpha und den Werten der Nachfolgeknoten ausgewählt. Das bedeutet, Alpha wird immer dann erhöht, wenn ein nachfolgender Knoten einen höheren Wert liefert als das aktuelle Alpha. Am MIN-Knoten hingegen wird Beta jeweils auf das Minimum aus dem bisherigen Beta und den Werten der Nachfolgeknoten gesetzt. Sobald an einem Knoten die Bedingung $\alpha \geq \beta$ erfüllt ist, wird der restliche Teilbaum nicht weiter betrachtet. In diesem Fall hat der MIN bereits eine bessere Alternative gefunden, sodass MAX diesen Zweig des Baums nicht mehr wählen würde [Ado09].

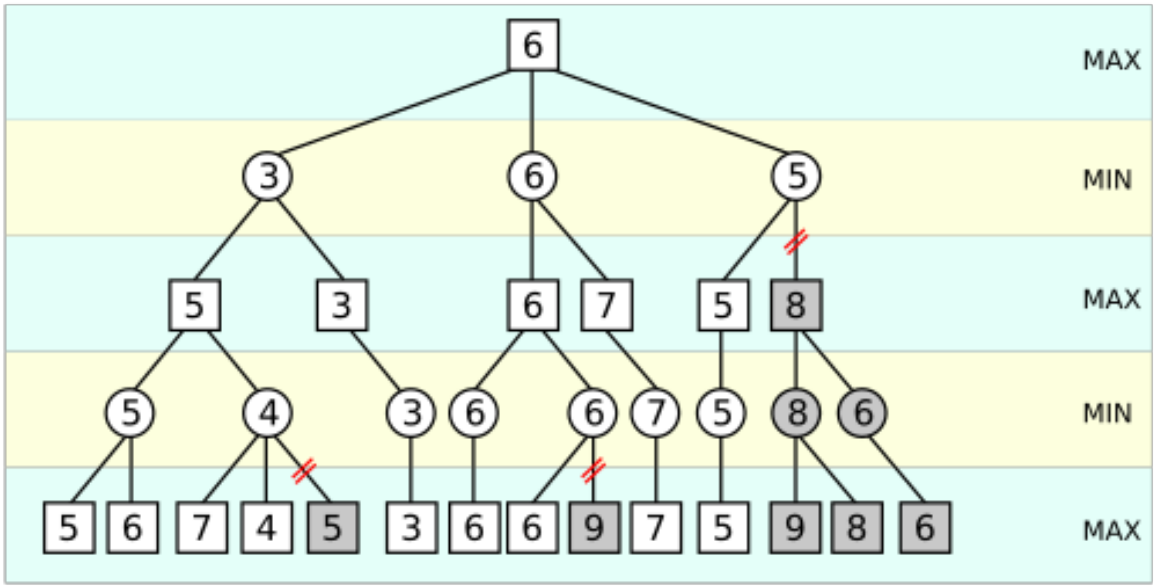

Abbildung 2.1: Alpha-Beta Spielbaum

2.3 Mikropython

MicroPython ist eine speziell für Mikrocontroller angepasste Version der Programmiersprache Python. Im Gegensatz zur Desktop-Variante lässt sich MicroPython-Code direkt auf Hardware mit begrenzten Ressourcen ausführen[SS23][PLJ23]. Im Unterschied zu Standard-Python 3 bringt MicroPython allerdings nur einen Teil der

gewohnten Python-Standardbibliotheken mit.
Wodurch es weniger Speicherplatz benötigt. Zudem besitzt MicroPython einen eigenen Interpreter, der direkt auf einem Mikrokontroller ausgeführt werden kann. Dadurch eignet sich MicroPython besonders gut für die Programmierung des LEGO Spike Hubs[Bel24].

2.4 LEGO Spike Hub:

Der LEGO Spike Hub ist das Herzstück des LEGO Spike Prime Sets. Er dient als programmierbare Steuereinheit mit sechs LPF2 input/output ports, an die alle LEGO-Sensoren und -Motoren angeschlossen werden können. Im Inneren arbeitet ein eigener Prozessor (100 MHz ARM Cortex-M4), unterstützt von 320 KB RAM und 1 MB Flash-Speicher. Die Programmierung des LEGO Spike Hubs erfolgt in der Sprache MicroPython. LEGO stellt dafür eine eigene Entwicklungsumgebung (IDE) bereit, mit dieser der Hub einfach programmiert werden kann. Hierfür kann der Hub über USB oder via Bluetooth mit dem Computer verbunden werden. Die Steuereinheit wird über einen wiederaufladbarer Lithium-Ionen-Akku mit Strom und Spannung versorgt [LEG20b].

Weitere technische Merkmale des LEGO Spike Hubs sind:

- Individuell anpassbaren Lichtmatrix (5x5)
- Lautsprecher
- Taster mit integrierter Statusleuchte
- Tasten für die Navigation und Steuerung durch Menüs am Hub
- Lautsprecher

- sechsachsiger Gyrosensor

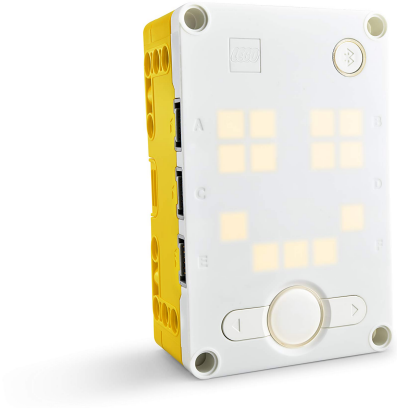

Abbildung 2.2: LEGO Spike Hub

2.5 Sensorik

2.5.1 LEGO Spike Kraft- oder Tuchsensor

Dieser Sensor erkennt, ob er gedrückt wurde, und misst dabei gleichzeitig die auf ihn ausgeübte Kraft. Mit einer Abtastrate von 100 Hz erfasst er Kräfte im Bereich von 2,5 bis 10 Newton und arbeitet dabei mit einer Genauigkeit von $\pm 0,65$ Newton. Der gemessene Wert wird als Prozentwert ausgegeben, wobei 100% einem Tastendruck von 10 Newton entsprechen. Typischerweise wird der Sensor zum Erkennen von Hindernissen oder als Start- bzw. Stopptaste eines Roboters eingesetzt. Der Kraft- oder Tuchsensor wird direkt am LEGO Spike Hub angeschlossen [LEG20c].

Abbildung 2.3: LEGO Spike Kraft- oder Tuchsensor

2.5.2 LEGO Spike Farbsensor

Dieser elektronische Farbsensor wurde speziell für LEGO Spike entwickelt. Seine Abtastrate beträgt 1 kHz und er kann direkt am Hub angeschlossen werden. Der Sensor kann bis zu acht verschiedene Farben erkennen, darunter Schwarz, Blau, Rot, Weiß, Braun, Gelb, Pink und Grün. Außerdem misst er sowohl die Intensität des reflektierten Lichts als auch die des Umgebungslichts [LEG20a][LEG20c].

Für die Farberkennung erfasst der Sensor die Farbwerte sowohl im RGB- (Rot, Grün, Blau) als auch im HSV-Farbraum (hue = Farbton, saturation = Sättigung, value = Helligkeit). Die Messergebnisse werden als Ganzzahlen ausgegeben [LEG20a].

Zur Reflexionsmessung sendet der Sensor weißes Licht auf eine Oberfläche und misst das zurückgeworfene Licht. Diese Funktion wird häufig für Linienführung eingesetzt [Bet25][LEG20a].

Abbildung 2.4: LEGO Technic Farbsensor

2.6 Aktorik

2.6.1 LEGO Spike Winkelmotor

Der LEGO Spike Winkelmotor nicht nur ein einfacher Elektromotor, aufgrund eines integrierten Drehsensors kann er nicht nur die Drehrichtung, sondern auch die relative und absolute Position (in Grad) sowie die Drehgeschwindigkeit erfassen. Eine vollständige Umdrehung wird dabei in 360 einzelne Zählimpulse unterteilt, wobei die Genauigkeit des Motors bei ±3 Grad liegt. Muss der Motor ein Drehmoment von mehr als 5 Ncm aufbringen, blockiert er. Mit einer Abtastrate von 100 Hz erfasst der Motor die Position sowohl beim automatischen als auch im manuellen Betrieb. Der LEGO Spike Winkelmotor eignet sich somit nicht nur für Bewegungsaufgaben, sondern auch zur Positionsbestimmung [LEG20c].

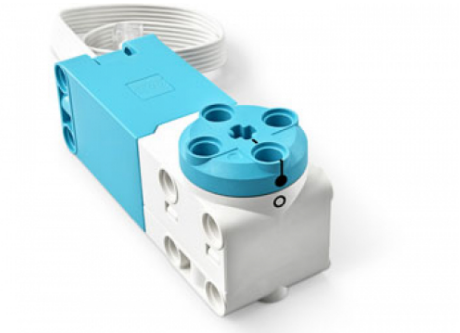

Abbildung 2.5: LEGO Spike Winkelmotor

3 Vorgehen

Im folgendem Kapitel wird auf die Planung des Vier-Gewinnt-Roboters eingegangen. Zunächst werden die Anforderungen definiert. Anschließend werden verschiedene Konzepte genauer betrachtet und miteinander verglichen.

3.1 Aufgabenpräzisierung

3.2 Anforderungen an den Roboter

In der Tabelle 3.1 sind die Anforderungen an den Vier-Gewinnt-Roboter in einer Anforderungsliste zusammengetragen. Dabei wird zwischen Forderungen und Wünschen unterschieden. Anforderungen, die mit ***F*** gekennzeichnet sind, müssen unbedingt umgesetzt werden. Während hingegen Anforderungen, die mit ***W*** markiert sind, als Wünsche zu verstehen sind und nicht zwingend im System realisiert werden müssen.

Tabelle 3.1: Anforderungstabelle für einen Vier-Gewinnt-Roboter

Nr	Anforderung an das System	F/W
-	Komplettes Spielfeld scannen	F
-	Das Ende des Spiels erkennen	F
-	Autonom fahren	F
-	Begrezungen des Spielfeld erkennen	F
-	Steine selber platzieren	F
-	Immer nur ein Stein pro Spielzug	F
-	Abwarten bis der Gegner sein Zug beendet hat	F
-	Nach jeden Zug rechts vom Spielfeld wegfahren	F
-	maximal 90 Sekunden pro Spielzug	F
-	optimalen Spielzug berechnen	F
-	Scannen bis ein neuer gegnerischer Stein erkannt wurde	W
-	Volle Spalten überspringen	W
-	Wenn eine leere Zeile erkannt wurde zur nächsten Spalte wechseln	W
-		F

Legende: F = Forderung, **W** = Wunsch,

3.3 der Algorithmus

Struktur und Funktionalität des Python-Codes für ein 4-Gewinnt-Spiel

1. Initialisierung und Konfiguration

Zu Beginn werden grundlegende Variablen und Datenstrukturen definiert, die den Zustand des Spiels abbilden. Das Spielfeld wird als zweidimensionale Liste mit 6 Zeilen und 7 Spalten realisiert, wobei freie Felder mit dem Symbol `"` gekennzeichnet sind. Zusätzlich wird ein Dictionary zur Zuordnung der numerischen Spielwerte (0 für den Computer und 2 für den Menschen) zu deren Symbolen („O“ bzw. „X“) erstellt. Die Bewertungsmatrix `werte` dient zur heuristischen Gewichtung der Spielfeldpositionen, wobei zentrale Felder höher bewertet werden als Randpositionen, da sie strategisch vorteilhafter sind. Diese Initialisierung schafft die Grundlage für die spätere Spiellogik

und die Bewertung von Spielsituationen.

2. Ausgabefunktion

Die Funktion `ausgabe()` visualisiert das aktuelle Spielfeld in der Konsole. Dabei werden numerische Spielzustände in die zugehörigen Zeichen („O“ oder „X“) umgewandelt und zeilenweise ausgegeben. Nicht belegte Felder erscheinen als " und am unteren Rand wird eine Spaltenbeschriftung (0 bis 6) zur Orientierung angezeigt. Die Funktion dient sowohl der Information als auch der Interaktion mit dem menschlichen Spieler. Sie ermöglicht die Nachvollziehbarkeit der Spielzüge und den aktuellen Spielverlauf.

3. Bewertungsfunktion

Die Funktion `bewertung()` liefert einen numerischen Wert, der die aktuelle Spielsituation heuristisch einschätzt. Für jedes belegte Feld wird basierend auf der Bewertungsmatrix ein Wert addiert oder subtrahiert: positive Werte für den Computer, negative für den Menschen. Dadurch entsteht ein Maß für die Positionierungsvorteile des Computers gegenüber dem Gegner. Die Funktion kommt im Minimax-Algorithmus zum Einsatz, wenn die maximale Suchtiefe erreicht ist oder keine Gewinnsituation erkannt wurde. Sie ermöglicht eine differenzierte Einschätzung von Zwischenständen jenseits reiner Gewinnbedingungen.

4. Auswertungsfunktion

Die Funktion `auswertung()` überprüft, ob eine Spielsituation vorliegt, in der ein Spieler vier Spielsteine in einer Reihe platzieren konnte. Dabei werden horizontale, vertikale und diagonale Kombinationen systematisch untersucht. Ergibt sich eine solche Konstellation, so wird der entsprechende Spieler (0 für Computer, 2 für Mensch) als Gewinner zurückgegeben. Liegt keine Gewinnsituation vor, wird zudem überprüft, ob

das Spielfeld voll ist, was auf ein Unentschieden hinweist (Rückgabewert 1). In allen anderen Fällen wird -1 zurückgegeben, was den Fortbestand des Spiels signalisiert.

5. Zugfunktionen

Die Funktion `zug(y, s)` simuliert das Einwerfen eines Spielsteins in die Spalte `y` für den Spieler `s`. Dabei wird die unterste freie Position in der Spalte gesucht und entsprechend belegt. Die komplementäre Funktion `zugRueckgaengig(y)` entfernt den obersten Spielstein aus der Spalte und stellt somit den vorherigen Spielzustand wieder her. Diese Rücknahmefunktion ist essenziell für die Implementierung des Minimax-Algorithmus, da im Rahmen der rekursiven Bewertung zahlreiche hypothetische Züge getestet und wieder verworfen werden müssen. Beide Funktionen gewährleisten eine realitätsnahe Simulation von Spielzügen.

6. Minimax-Algorithmus

Die beiden Funktionen `maxFunktion(tiefe)` und `minFunktion(tiefe)` implementieren gemeinsam den klassischen Minimax-Algorithmus mit begrenzter Suchtiefe. `maxFunktion` wird vom Computer aufgerufen und versucht, den bestmöglichen (höchsten) Bewertungswert zu erzielen, während `minFunktion` die besten Gegenreaktionen des menschlichen Spielers simuliert. Beide Funktionen überprüfen vor jedem Zug, ob bereits ein Gewinn oder Unentschieden vorliegt, um die Rekursion ggf. zu beenden. Wenn die maximale Suchtiefe erreicht ist, erfolgt die Bewertung über die Heuristik. Diese Logik bildet das Kernstück der künstlichen Intelligenz im Spiel.

7. Zugauswahl des Computers

Die Funktion `besterZug()` bestimmt den optimalen Zug des Computers unter Anwendung des Minimax-Verfahrens mit einer festen Suchtiefe von vier. Für jede gültige

Spalte wird simuliert, wie sich der Zug auf die Spielsituation auswirken würde, wobei anschließend der resultierende Spielzustand durch den Minimax-Algorithmus bewertet wird. Der Zug mit dem geringsten Wert aus Sicht des Gegners (also dem besten aus Computersicht) wird als nächster Spielzug ausgewählt. Diese Funktion stellt sicher, dass der Computer stets eine taktisch sinnvolle Entscheidung trifft. Sie verbindet die Simulation potenzieller Spielverläufe mit einer strategischen Bewertung.

8. Spielsteuerung

Die Funktion `spiele()` koordiniert den gesamten Spielablauf zwischen Mensch und Computer. Nach der Initialisierung des Spielfelds folgt eine Spielschleife, in der der Mensch und der Computer abwechselnd ihre Züge tätigen. Der Spieler gibt seine Eingabe über die Konsole ein, während der Computer seinen Zug automatisch über die `besterZug()`-Funktion berechnet. Nach jedem Zug wird der Spielzustand neu ausgegeben und auf ein mögliches Spielende überprüft. Die Spielsteuerung bildet somit die zentrale Benutzerschnittstelle und kontrolliert die Spiellogik.

9. Programmausführung

Der abschließende Block `if __name__ == "__main__"` sorgt dafür, dass das Spiel nur dann automatisch gestartet wird, wenn das Skript direkt ausgeführt wird. Dies ist in Python eine gängige Praxis, um eine Datei sowohl als importierbares Modul als auch als eigenständig ausführbares Programm nutzen zu können. Dadurch wird verhindert, dass das Spiel beim Import des Skripts in andere Module unbeabsichtigt gestartet wird. Dieser Mechanismus fördert die Wiederverwendbarkeit und Modularität des Codes. In wissenschaftlichen Anwendungen ist dies ein Merkmal guter Programmierpraxis.

4 Umsetzung und Ergebnisse

Mechanischer Aufbau des LEGO-Spike-4-Gewinnt-Roboters

Für die Umsetzung des 4-Gewinnt-Roboters wurde eine mechanische Konstruktion gewählt, die es erlaubt, das Spielfeld zu scannen sowie Chips gezielt in eine Spalte einzuwerfen. Der Aufbau umfasst drei LEGO Spike-Motoren, einen Farbsensor und einen Drucktaster. Im Folgenden werden die einzelnen Komponenten detailliert beschrieben.

- **Horizontalantrieb – Motor D**
  Der horizontale Antrieb des Farbsensors erfolgt über Motor D. Dieser ist dafür zuständig, die Spielfeldspalten nacheinander anzufahren. Der Motor ist mit einer Achse verbunden, welche zwei Räder antreiben. Die Bewegung erfolgt in gleichmäßigen Schritten: Eine Drehung um exakt 72 Grad bewegt den Schlitten um eine Spalte weiter. Diese Schrittweite wurde so gewählt, dass sie der Breite einer Spalte im Spielfeld entspricht. Dadurch ist eine exakte Positionierung des Sensors über jeder Spalte möglich, ohne dass zusätzliche Sensoren zur Positionsbestimmung notwendig sind.

- **Vertikalantrieb – Motor E, Kette und Farbsensor**
  Um das Spielfeld auch in vertikaler Richtung abfahren zu können, ist der Farbsensor an einer Kette montiert. Diese Kette wird durch Motor E angetrieben. Der Sensor ist an einem mittleren Segment der Kette befestigt und fährt beim Drehen der Kette entsprechend auf und ab. Ein Schritt des Motors um etwa 95 Grad bewegt den Sensor um genau eine Spielfeldhöhe weiter. Auf diese Weise können sämtliche sechs Reihen der aktuellen Spalte nacheinander abgescannt

werden. Der Sensor wurde dabei so montiert, dass er exakt über der Mitte jedes Feldes positioniert ist, um eine zuverlässige Farberkennung zu ermöglichen. Die Rückwärtsbewegung der Kette erlaubt es, den Sensor wieder nach unten zu fahren.

- **Chipauswerfer – Motor A**
  Das Einwerfen des eigenen Spielsteins erfolgt über Motor A. An diesem Motor ist eine Stange montiert, die bei einer vollständigen Umdrehung einen Spielchip aus dem Vorratsmagazin in die gewünschte Spalte stößt. Nach der Auslösung kann ein neuer Chip in die Abschussposition nachrutschen. In der Software ist eine Wartezeit nach dem Auslösen eingebaut, damit der Chip sicher im Spielfeld ankommt, bevor die nächste Aktion beginnt.

- **Startsignal – Drucksensor (Force Sensor)**
  Um dem Roboter mitzuteilen, dass der menschliche Spieler seinen Zug abgeschlossen hat, wurde ein Drucksensor verwendet. Dieser befindet sich an der Vorderseite des Roboters. Sobald der Spieler den Sensor leicht berührt, wird ein Signal ausgelöst, das Prozess startet.

- **Spielfeldscan – Farbsensor an Kette**
  Für die Farberkennung des Spielfeldes wurde ein LEGO-Farbsensor verwendet, der über die oben beschriebene Kettenkonstruktion vertikal verfahrbar ist. Die Farbmessung erfolgt jeweils in der Mitte eines Spielfeldes. Der Sensor erkennt RGB-Werte, die per Software verschiedenen Spielsteinfarben (in diesem Projekt benutzt: Rot, Gelb oder Leer) zugeordnet werden. Während des Spiels vergleicht der Algorithmus die gemessenen Werte mit diesen Referenzwerten, um die tatsächliche Belegung jedes Feldes möglichst robust zu bestimmen. Der Abstand zwischen Sensor und Spielfeld beträgt etwa 7 mm – dieser Wert hat sich als optimal für zuverlässige Farbmessung erwiesen.

Zusammenfassung
Die mechanische Konstruktion basiert auf einem kartesischen Koordinatensystem, bei dem der Sensor durch die Kombination aus horizontaler (Motor D) und vertikaler Be-

wegung (Motor E + Kette) jede Spielfeldposition präzise anfahren kann. Das System erlaubt eine vollautomatische Spielweise: Der Roboter erkennt die aktuelle Spielsituation, berechnet den optimalen Zug und setzt diesen physisch um.

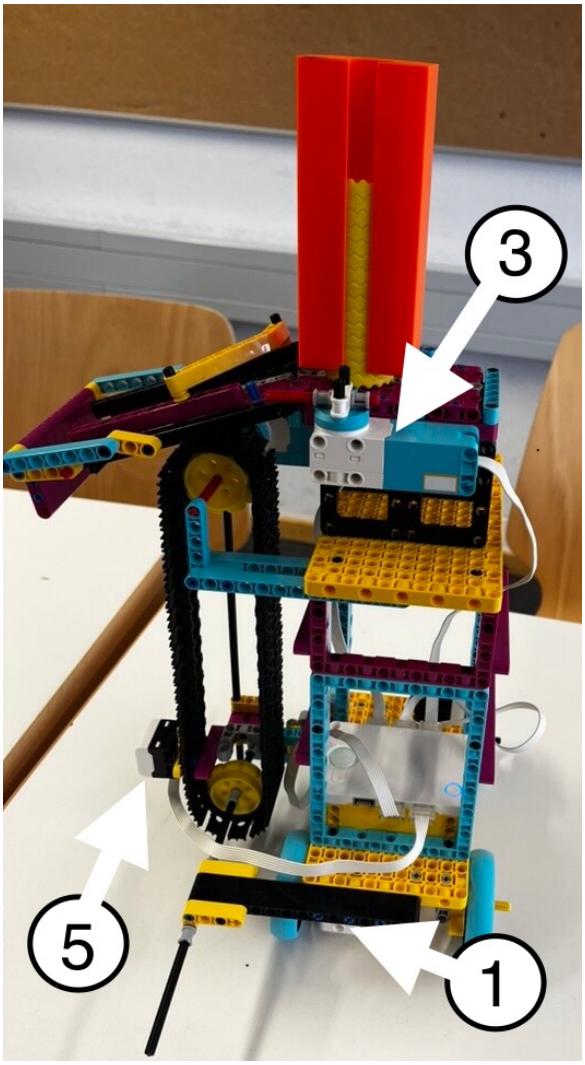

Abbildung 4.1: Seitenansicht links

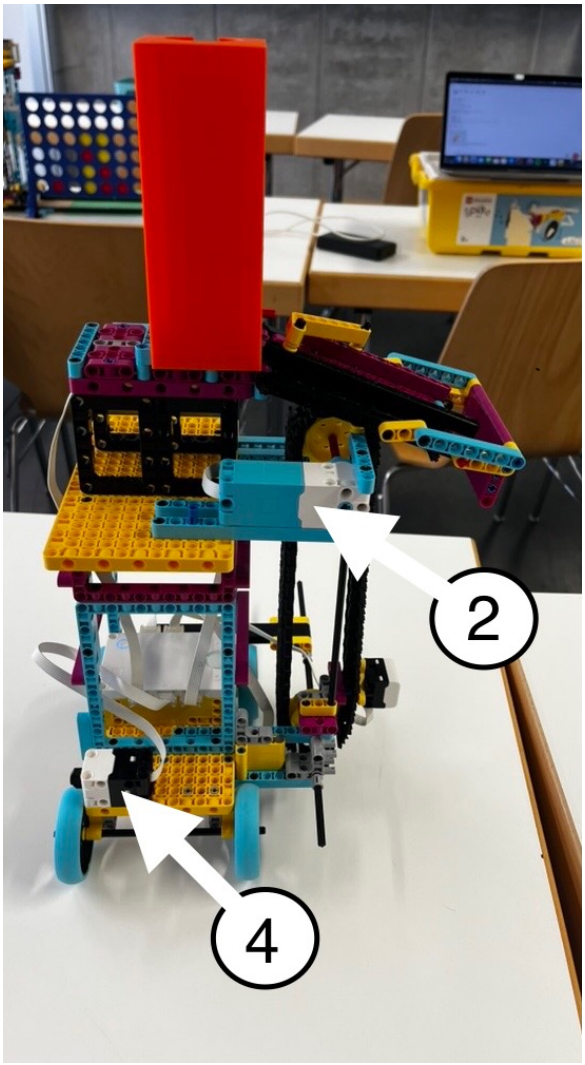

Abbildung 4.2: Seitenansicht rechts

5 Zusammenfassung

Auf zwei bis drei Seiten soll auf folgende Punkte eingegangen werden:

- Welches Ziel sollte erreicht werden
- Welches Vorgehen wurde gewählt
- Was wurde erreicht, zentrale Ergebnisse nennen, am besten quantitative Angaben machen
- Konnten die Ergebnisse nach kritischer Bewertung zum Erreichen des Ziels oder zur Problemlösung beitragen
- Ausblick

In der Zusammenfassung sind unbedingt klare Aussagen zum Ergebnis der Arbeit zu nennen. Üblicherweise können Ergebnisse nicht nur qualitativ, sondern auch quantitativ benannt werden, z. B. „. . . konnte eine Effizienzsteigerung von 12 % erreicht werden.“ oder „. . . konnte die Prüfdauer um 2 h verkürzt werden“.

Die Ergebnisse in der Zusammenfassung sollten selbstverständlich einen Bezug zu den in der Einleitung aufgeführten Fragestellungen und Zielen haben.

Literaturverzeichnis

[Ado09] Julius Adorf. *Adversariale Suche für optimales Spiel: Der Minimax-Algorithmus und die Alpha-Beta-Suche*. Proseminararbeit. Betreuer: Lars Kunze, Dominik Jain. Abgabetermin: 2. Dezember 2009. München: Technische Universität München, Fakultät für Informatik, Forschungs- und Lehreinheit Informatik IX, 2009. URL: https://www.juliusadorf.com/pub/alphabeta-seminar-paper.pdf.

[Bel24] Charles A. Bell. *MicroPython for the Internet of Things: A Beginner's Guide to Programming with Python on Microcontrollers*. English. 2nd. Berkeley, CA: Apress, 2024. ISBN: 9781484298619.

[Bet25] Betzold GmbH. *LEGO Education SPIKE Technic Farbsensor*. Zugriff am 04.07.2025. 2025. URL: https://www.betzold.de/prod/E_761144/.

[Has20] SA Hasbro. *Das Originale 4Gewinnt Anleitung*. Hasbro Gaming, 2020.

[LEG20a] LEGO Education. *Technical Specifications: Technic Color Sensor*. https://assets.education.lego.com/v3/assets/blt293eea581807678a/blt62a78c227edef070/5f8801b9a302dc0d859a732b/techspecs_technic colorsensor.pdf?locale=en-us. Accessed: 2025-07-04. 2020.

[LEG20b] LEGO Education. *Technical Specifications: Technic Large Hub*. https://assets.education.lego.com/v3/assets/blt293eea581807678a/bltf512a371e82f6420/5f8801baf4f4cf0fa39d2feb/techspecs_technic largehub.pdf?locale=en-us. Zugriff am 03.07.2025. 2020.

[LEG20c] LEGO Education Community. *Exploring SPIKE™ Prime Sensors*. Accessed: 2025-07-04. 2020. URL: https://community.legoeducation.com/blogs/31/220.

[PLJ23] Ignas Plauska, Agnius Liutkevičius und Audronė Janavičiūtė. „Performance Evaluation of C/C++, MicroPython, Rust and TinyGo Programming Languages on ESP32 Microcontroller“. In: *Electronics* 12.1 (2023). Open Access Article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license, S. 143. DOI: 10.3390/electronics12010143. URL: https://www.mdpi.com/2079-9292/12/1/143.

[SS23] Albin Samefors und Felix Sundman. *Investiating Energy Consumption and Responsiveness of low power modes in MicroPython for STM32WB55*. Bachelor's thesis. 15 hp (first-cycle education). Jönköping, Sweden, Juni 2023. URL: https://www.diva-portal.org/smash/get/diva2:1778426/FULLTEXT01.pdf.

Abbildungsverzeichnis

Tabellenverzeichnis

A Nutzung von Künstliche Intelligenz basierten Werkzeugen

Im Rahmen dieser Arbeit wurden Künstliche Intelligenz (KI) basierte Werkzeuge benutzt. Tabelle 1.1 gibt eine Übersicht über die verwendeten Werkzeuge und den jeweiligen Einsatzzweck.

Tabelle 1.1: Liste der verwendeten KI basierten Werkzeuge

Werkzeug	**Beschreibung der Nutzung**
ChatGPT	• Grundlagenrecherche zu bekannten Prinzipien optischer Sensorik zur Abstandsmessung (siehe Abschnitt ...) • Suche nach Herstellern von Lidar-Sensoren (siehe Abschnitt ...) • ...
ChatPDF	• Recherche und Zusammenfassung von wissenschaftlichen Studien im Themenfeld ... • ...
DeepL	• Übersetzung des Papers von [...]
Tabnine AI coding assistant	• Aktiviertes Plugin in MS Visual Studio zum Programmieren des ... • ...
...	• ...

B Ergänzungen

2.1 Details zu bestimmten theoretischen Grundlagen

2.2 Weitere Details, welche im Hauptteil den Lesefluss behindern

C Details zu Laboraufbauten und Messergebnissen

3.1 Versuchsanordnung

3.2 Liste der verwendeten Messgeräte

3.3 Übersicht der Messergebnisse

3.4 Schaltplan und Bild der Prototypenplatine

D Zusatzinformationen zu verwendeter Software

4.1 Struktogramm des Programmentwurfs

4.2 Wichtige Teile des Quellcodes

E Datenblätter

Auf den folgenden Seiten wird eine Möglichkeit gezeigt, wie aus einem anderen PDF-Dokument komplette Seiten übernommen werden können, z. B. zum Einbindungen von Datenblättern. Der Nachteil dieser Methode besteht darin, dass sämtliche Formateinstellungen (Kopfzeilen, Seitenzahlen, Ränder, etc.) auf diesen Seiten nicht angezeigt werden. Die Methode wird deshalb eher selten gewählt. Immerhin sorgt das Package *„pdfpages“* für eine korrekte Seitenzahleinstellung auf den im Anschluss folgenden „nativen“ LaTeX-Seiten.

Eine bessere Alternative ist, einzelne Seiten mit *„\includegraphics“* einzubinden.

2

3

4