DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

# Vier-Gewinnt Roboter

## ggf. Untertitel mit ergänzenden Hinweisen

## Studienarbeit T3_3100

Studiengang Elektrotechnik

Studienrichtung Automation

Duale Hochschule Baden-Württemberg Ravensburg, Campus Friedrichshafen

von

Patrik Peters / Simon Gschell

| | |
|---|---|
| Abgabedatum: | 6. November 2024 |
| Bearbeitungszeitraum: | 10.10.2024 - 13.06.2025 |
| Matrikelnummer: | 187 /0815 |
| 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_3100 mit dem Thema:

*Vier-Gewinnt Roboter*

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.

Musterstadt, den 6. November 2024

---

Patrik Peters / Simon Gschell

# Kurzfassung

Problemstellung

Ziel der Arbeit

Vorgehen und angewandte Methoden

Konkrete Ergebnisse der Arbeit, am besten mit quantitativen Angaben

In der vorliegenden Studienarbeit wird die Entwicklung eines Roboters für das Spiel „Vier gewinnt“ unter Verwendung des LEGO Spike Prime Systems behandelt. Ziel des Projekts ist es, einen Roboter zu entwerfen, der in der Lage ist, autonom die Rolle eines menschlichen Gegners im Spiel „Vier gewinnt“ zu übernehmen. Der Roboter soll nicht nur die Spielzüge des menschlichen Mitspielers erkennen und darauf reagieren, sondern auch selbstständig Spielsteine in das Spielfeld einwerfen und sicherstellen, dass das Spielfeld für den nächsten Zug bereit ist. Dies erfordert eine präzise Steuerung des Roboters, insbesondere beim Platzieren der Spielsteine, sowie die Fähigkeit, das Spielfeld zu überwachen, um die Position der bereits gesetzten Steine zu erkennen. Das Projekt umfasst verschiedene technische und organisatorische Aspekte. Dazu gehört die mechanische Konstruktion des Roboters, einschließlich des Mechanismus zum Einwerfen der Spielsteine und die Überwachung des Spielfelds, sowie die Entwicklung der Software, die die Spielzüge und das Verhalten des Roboters steuert. Ein weiterer wichtiger Punkt ist die Sensorik: Der Roboter muss in der Lage sein, die aktuelle Spielsituation durch Farbsensoren zu erfassen, um zu wissen, welche Felder im Spielfeld bereits besetzt sind und wo er seinen nächsten Spielstein platzieren kann. Zum Abschluss des Projekts wird ein kleines Turnier organisiert, bei dem die von verschie-

denen Teams entwickelten Roboterlösungen gegeneinander antreten. Dies bietet die Möglichkeit, den Roboter unter realen Bedingungen zu testen und seine Fähigkeiten im direkten Vergleich mit den Lösungen der anderen Kommilitonen zu messen. Um zusätzliche Motivation zu schaffen, wird die beste Lösung am Ende prämiert, was den Wettbewerbsgedanken fördert und den Anreiz erhöht eine effektive Lösungen zu entwickeln.

# Abstract

English translation of the „Kurzfassung“.

# Inhaltsverzeichnis

# 1 Einleitung

# 2 Grundlagen

## 2.1 Spielregeln

### 2.1.1 Vorschriften:

- max. Zeitbegrenzung eines Zuges: -> scannen des Spielfeldes und Rechenzeit müssen begrenzt sein

## 2.2 Geschichte

## 2.3 Spieltheorie

## 2.4 Spike

## 2.5 MicroPython

# 3 Vorgehen

## 3.1 Vorgehensweise

Zur Realisierung lässt sich dieses Projekt in drei wesentliche Aspekte aufteilen.

- Entwicklung eines mechanischen Konzepts: Es soll eine funktionale Vorrichtung entworfen werden, der in der Lage ist, Spielsteine präzise in den Spielständer einzuführen und diese anschließend für den nächsten Spielzug freizugeben.

- Erfassung der Ist-Situation: Durch den Einsatz geeigneter Sensorik soll der Roboter die aktuelle Position der gelben und roten Chips im Spielstand erfassen und diese Informationen an den Mikrocontroller weiterzugeben.

- Erstellung eines effizienten Algorithmus: Ein in microPython programmierter Algorithmus muss entwickelt werden, der innerhalb der begrenzten Rechenkapazitäten des verwendeten Controllers effizient arbeitet und die nötigen Steuerbefehle für die Roboteraktionen bereitstellt. Für die beste Strategie soll dabei mathematische Spieltheorie analysiert werden und diese in das System eingebunden werden.

## 3.2 Anforderung

## 3.3 Konzept

## 3.4 Nutzwertanalyse

## 3.5 Zeitplan

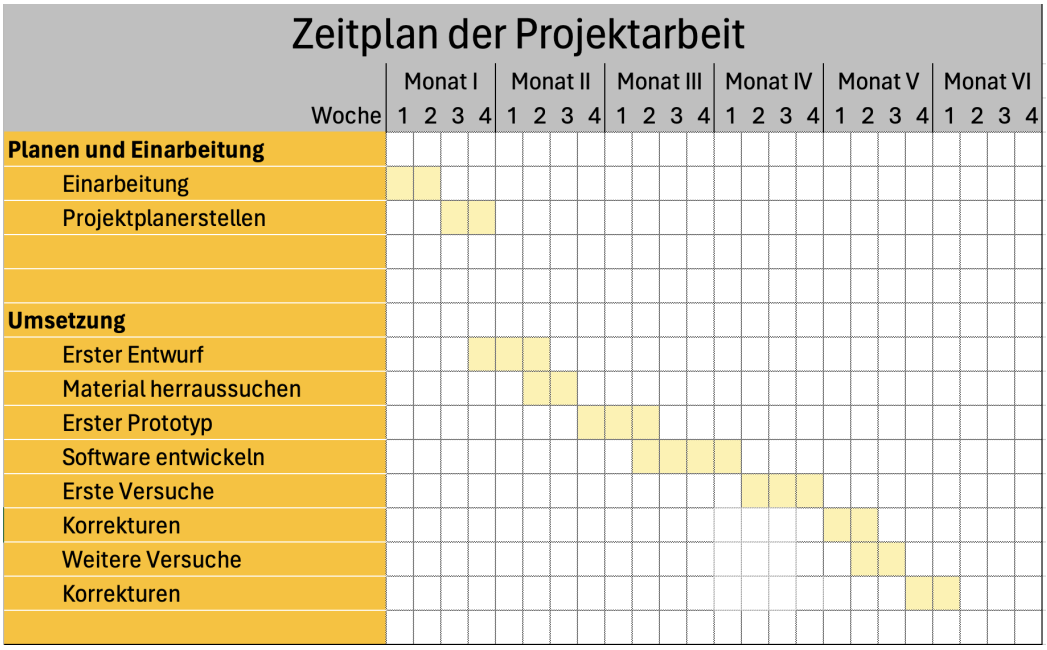



**Abbildung 3.1:** Zeitplan

# 4 Umsetzung und Ergebnisse

## 4.1 Voraussetzungen

Voraussetzung für dieses Projekt ist ein fundiertes Verständnis der Programmierung sowie Kreativität bei der Konstruktion. Diese Fähigkeiten bringen wir durch unsere abgeschlossene Ausbildung im Bereich Mechatronik und Elektronik mit. Darüber hinaus sind Kenntnisse in der Softwareentwicklung erforderlich, insbesondere im Hinblick auf die Programmierung des LEGO Spike Prime Systems, um den Roboter erfolgreich zu steuern und die Interaktion mit der Spielumgebung zu gewährleisten. Die Studienarbeit erstreckt sich über zwei Praxisphasen, was einem Zeitraum von insgesamt sechs Monaten entspricht. In dieser Zeit werden die theoretischen Grundlagen, die während des Studiums erlangt haben, in die Praxis umsetzen, um eine vollständige Robotiklösung zu entwickeln, die den Anforderungen des Projekts gerecht wird.

## 4.2 Aufbau

## 4.3 Software

### 4.3.1 Herausforderung der Programmierung

Die größte Herausforderung bei der Realisierung des Vier-Gewinnt-Roboters besteht in der begrenzten Rechenleistung des verwendeten Mikrocontrollers. Anders als bei leistungsstarken Computern, die in der Lage sind, vollständige Spielalgorithmen zu berechnen und dadurch eine perfekte Strategie zu verfolgen, muss der Roboter mit deutlich eingeschränkten Ressourcen auskommen. Ein umfassender Algorithmus, der jede mögliche Zugkombination im Vier-Gewinnt-Spiel analysiert, erfordert erheblichen Speicherplatz und Rechenleistung, da mit jedem neuen Zug die Anzahl der möglichen Spielverläufe exponentiell ansteigt. Auf einem leistungsstarken Computer wäre es theoretisch möglich, eine "perfekteSStrategie zu entwickeln, die den gesamten Spielbaum durchläuft und immer den besten Zug auswählt. Der Mikrocontroller des Roboters hingegen hat nicht die Kapazität, all diese Berechnungen in angemessener Zeit durchzuführen. Aus diesem Grund kann der Roboter nur eine begrenzte Anzahl von Zügen im Voraus berechnen. Er muss sich auf Algorithmen stützen, die in der Lage sind, kurzfristige Entscheidungen zu treffen, anstatt langfristige Strategien zu verfolgen. Das bedeutet, dass der Roboter durch Heuristiken (d.h. Daumenregeln oder Annäherungen) gesteuert wird, die ihm helfen, gute, aber nicht immer optimale Züge zu machen. Diese Heuristiken können einfache Prinzipien wie das Verhindern eines unmittelbaren Sieges des Gegners oder das Setzen eigener Spielsteine in strategisch günstige Positionen umfassen.

# 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.

# Abbildungsverzeichnis

# Tabellenverzeichnis

# A Ergänzungen

## A.1 Details zu bestimmten theoretischen Grundlagen

## A.2 Weitere Details, welche im Hauptteil den Lesefluss behindern

# B Details zu Laboraufbauten und Messergebnissen

## B.1 Versuchsanordnung

## B.2 Liste der verwendeten Messgeräte

## B.3 Übersicht der Messergebnisse

## B.4 Schaltplan und Bild der Prototypenplatine

# C Zusatzinformationen zu verwendeter Software

## C.1 Struktogramm des Programmentwurfs

## C.2 Wichtige Teile des Quellcodes

# D 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

# E Tips und Beispiele zu LaTeX-Befehlen

Dieses Kapitel können Sie einfach löschen, indem Sie in der Präambel am Anfang der Zeile „*include{chapter/anhang_vorlagen}*“ das Symbol % zum Auskommentieren einfügen.

## E.1 Wichtige LaTeX-Befehle

| | |
|---|---|
| *label*{} | Definition eines Labels, auf welches referenziert werden kann, z. B.: *label*{fig:MyImage} |
| *ref*{} | Setzen einer Referenz zu einem Label z. B.: ...siehe Tabelle~*ref*{tab:messdaten}. |
| *pageref*{} | Gibt die Seitenzahl zu einer Referenz zurück |
| *autocite*{} | Literaturreferenz einfügen |
| *autocite*[7]{} | Literaturreferenz einfügen, hier mit zus. Referenz auf Seite 7 |
| *autocites*{Abc15, Def16} | Mehrere Literaturreferenzen, hier Abc15 und Def16, einfügen |
| *footnote*{} | Fußnote einfügen |
| ~ | Einfügen eines geschützten Leerzeichens |
| $*Formel* $ | Eingabe einer Formel im Text |
| $*l*=*SI*{10}{*meter*}$ | Korrekte Ausgabe Maßzahl und Einheit in Formeln, hier $l = 10$ m |
| *index*{Kraft} | Aufnahme des Begriffs „Kraft“ in das Sachwort- |

| | |
|---|---|
| | verzeichnis |
| *index*{Induktion!Vollständige} | Aufnahme des Begriffs „Vollständige“ in das Sachwortverzeichnis unter „Induktion“. |
| *nomenclature*[etc]{etc.}{et cetera} | Aufnahme der Abkürzung „etc.“ für „et cetera“ in das Abkürzungsverzeichnis. Die Angabe [etc] dient als Sortierschlüssel |
| *clearpage* | Ausgabe aller Gleitobjekte und Umbruch auf eine neue Seite |

## E.2 Vorlagen für LaTeXUmgebungen

### E.2.1 Listen und Aufzählungen

Es gibt folgende Listentypen. Die wichtigsten:

- Einfache Liste mit *itemize*-Umgebung
- ...

1. Nummerierte Liste mit *enumerate*-Umgebung
2. ...

a. wobei man bei der *enumerate*-Umgebung leicht die Art der Nummerierung ändern kann,

b. ...

und durch verschachtelte Umgebungen verschiedene Aufzählungsebenen darstellen kann:

a) Erster Aufzählungspunkt der ersten Ebene

b) ...

- Erster Punkt der zweiten Ebene
- Zweiter Punkt der zweiten Ebene

c) Das sollte an Beispielen zunächst einmal genügen.

### E.2.2 Bilder und Grafiken

Bilder können als PDF-, JPG-, und PNG-Bilder in LaTeXeingebunden werden. Damit eine Grafik in hoher Qualität dargestellt wird, sollte das Dateiformat der Grafik vektorbasiert sein, d.h. als PDF-Datei vorliegen. Viele Zeichenprogramme unterstützen einen PDF-Export (z. B. GIMP, Adobe Illustrator, etc.). Für Grafiken aus PowerPoint sei folgende Vorgehensweise beim Export empfohlen:

1. Die gewünschte Grafik in PowerPoint zeichnen.
2. Gewünschten Bildbereich markieren, rechte Maustaste klicken und „Als Grafik speichern ...“ wählen.
3. Grafik im Format EMF abspeichern. Das EMF-Format ist vektorbasiert.[1]
4. Mit dem Programm XnView die Grafik im EMF-Format in PDF wandeln und abspeichern.
5. Die so erzeugte PDF-Datei enthält eine vektorbasierte Grafik und kann in LaTeX eingebunden werden.

Abbildung E.1 zeigt ein Beispielbild einer Grafik, welche aus PowerPoint exportiert wurde.


**Abbildung E.1:** Beispiel für die Einbindung eines Bildes (PDF-, JPG-, und PNG-Bilder können eingebunden werden).

Der Quellcode des Beispielbildes aus Abbildung E.1 ist in Listing E.1 zu sehen.

[1] Mit dem Mac kann in PowerPoint die Grafik direkt im PDF-Format exportiert werden. Die weiteren Schritte entfallen daher.

**Listing E.1:** Quellcode der Abbildung E.1.

```
\begin{figure}[hbt]        % here, bottom, top
\centering                 % Zentrierung
\includegraphics[width=0.6\linewidth]{images/MyImage}
\caption[Beispiel für die Einbindung eines Bildes.]{Beispiel für die
    Einbindung eines Bildes (PDF–, JPG–, und PNG–Bilder können
    eingebunden werden).}
\label{fig:MyImage}
\end{figure}
```

Jedes Bild aus fremder Quelle ist mit einem Zitat in der Abbildungsunterschrift zu kennzeichnen. Nur eigene Bilder benötigen keine entsprechende Kennzeichnung. Bilder aus fremder Quelle mit eigenen Ergänzungen oder Änderungen sind mit Zitat und einer entsprechenden Bemerkung (z. B. „auf Basis [Quelle] mit eigenen Ergänzungen“ oder „eigene Darstellung auf Basis [Quelle]“) zu versehen. Der besseren Lesbarkeit halber sind im Abbildungsverzeichnis keine Zitate anzugeben. Hierfür kann im Befehl *\caption[]{}* innerhalb der eckigen Klammer eine modifizierte Abbildungsunterschrift eingegeben werden, welche in das Abbildungsverzeichnis übernommen wird. Der Text innerhalb der geschweiften Klammer wird direkt unter die Abbildung gedruckt und kann dagegen ausführlich mit Angabe eines Zitats sein. Sollte die Arbeit veröffentlicht werden, ist unbedingt darauf zu achten, dass nur dann Bilder von fremder Quelle übernommen werden dürfen, wenn hierfür das explizite Einverständnis des Urhebers vorliegt. Dieses Einverständnis ist persönlich einzuholen und separat zu dokumentieren.

Grafiken können auch mithilfe des Packages Tikz gezeichnet, bzw. programmiert werden. Grafiken mit Tikz werden mit dem *input*-Befehl in die *figure*-Umgebung geladen, wie nachfolgendes Beispiel in Abbildung E.2 zeigt:

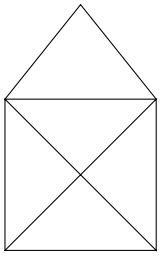
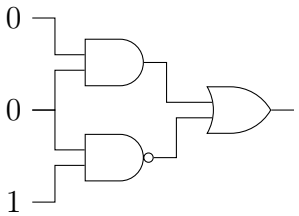


**Abbildung E.2:** Mit Tikz programmierte Grafik.

Ein etwas umfangreicheres Beispiel zur Digitaltechnik ist in Abbildung E.3 dargestellt:




Hier könnte Ihre Formel $y = (0 \wedge 0) \vee \overline{(0 \wedge 1)}$ stehen

**Abbildung E.3:** Mit Tikz programmierte Grafik, welche bereits vorgefertigte Bibliotheken für Symbole aus der Digitaltechnik nutzt.

In der Tikz-Umgebung können auch Diagramme mit dem *pgfplot*-Befehlssatz erzeugt werden. In Abbildung E.4 sehen Sie ein Beispiel.

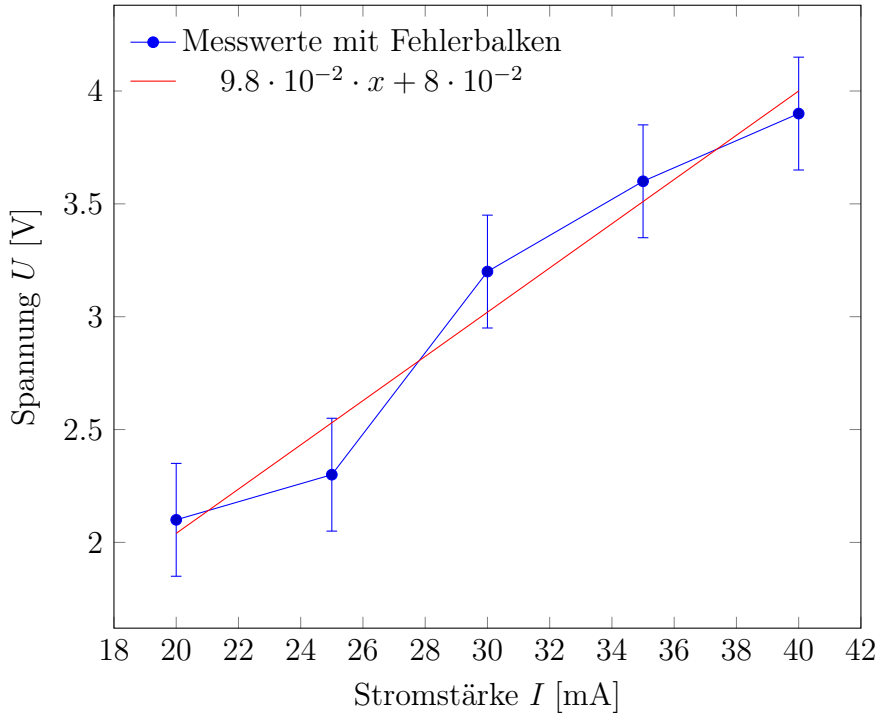



**Abbildung E.4:** Ein Diagramm, erstellt in der *tikzpicture*-Umgebung mit dem *pgfplot*-Befehlssatz. Das Diagramm stellt Messdaten, deren Fehlerbalken und eine Regressionskurve dar. Die Messdaten werden von einer separaten Datei eingelesen und die Regressionskurve wurde mit *pgfplot* berechnet und erstellt.

Auch hierzu der Quellcode in Listing E.2.

**Listing E.2:** Quellcode der Abbildung E.4.

```
\begin{figure}[hbt]
\centering
\input{pgfplot/mess_fehlerbalken.tex}
\caption[Diagramm, erstellt mit dem \textit{pgfplot}-Befehlssatz.]{Ein
    Diagramm, erstellt in der \textit{tikzpicture}-Umgebung mit dem \
    textit{pgfplot}-Befehlssatz. Das Diagramm stellt Messdaten, deren
    Fehlerbalken und eine Regressionskurve dar. Die Messdaten werden von
    einer separaten Datei eingelesen und die Regressionskurve wurde mit \
    textit{pgfplot} berechnet und erstellt.}
\label{fig:pgfplot}
\end{figure}
```

In Listing E.3 ist der Quellcode der Datei *mess_fehlerbalken.tex* dargestellt.

**Listing E.3:** Quellcode der Datei *mess_fehlerbalken.tex*.

```
\begin{tikzpicture}
\begin{axis}[scale=1.3,legend entries={Messwerte mit Fehlerbalken,
$\pgfmathprintnumber{\pgfplotstableregressiona} \cdot x
\pgfmathprintnumber[print sign]{\pgfplotstableregressionb}$}, legend
    style={draw=none},legend style={at={(0.01,0.98)},anchor=north west},
    xlabel=Stromstärke $I \; \mathrm{ \lbrack mA \rbrack}$,ylabel=
    Spannung $U \; \mathrm{ \lbrack V \rbrack}$]
\addlegendimage{mark=*,blue}
\addlegendimage{no markers,red}
\addplot+[error bars/.cd, y dir=both,y explicit]
table[x=x,y=y,y error=errory]
{pgfplot/messdaten_mitfehler.dat};
\addplot table[mark=none,y={create col/linear regression={y=y}}]
{pgfplot/messdaten_mitfehler.dat};
\end{axis}
\end{tikzpicture}
```

In Abbildung E.5 wird ein weiters Beispiel für ein Diagramm gezeigt. Oftmals wird eine zweite y-Achse verwendet, um verschiedene Skalen darstellen zu können.

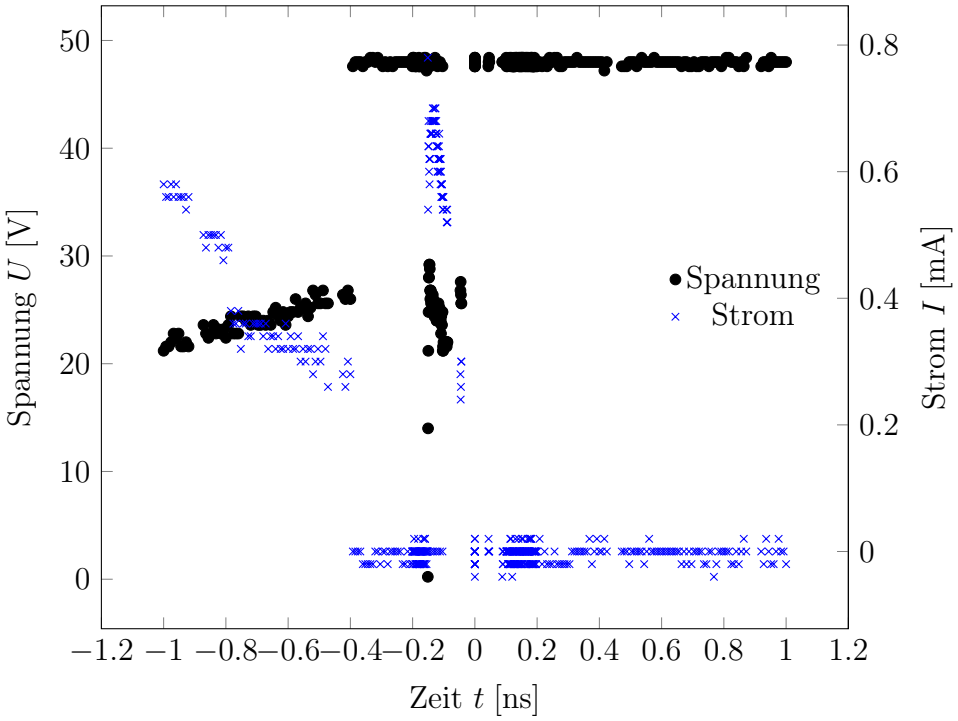



**Abbildung E.5:** Diagramm mit zwei unterschiedlichen y-Achsen.

### E.2.3 Tabellen

**Tabelle E.1:** Liste der verwendeten Messgeräte. Die Genauigkeitsangaben beziehen sich auf die Standardabweichung $1 \cdot \sigma$.

| Messgerät | Hersteller | Typ | Verwendung | Genauigkeit |
|---|---|---|---|---|
| Spannungs-versorgung | Voltmaker | HV2000 | Spannungs-versorgung der Platine | $\Delta U = \pm 5$ mV |
| Strommessgerät | Currentcount | Hotamp 16 | Strommessung am Versorgungspin des µC | $\Delta I = \pm 0.1$ A |

Der Quellcode der Beispieltabelle E.1 ist in Listing E.4 zu sehen.

**Listing E.4:** Quellcode der Tabelle E.1.

```
\begin{table}[hbt]
\centering
\renewcommand{\arraystretch}{1.5} % Skaliert die Zeilenhöhe der Tabelle
\captionabove[Liste der verwendeten Messgeräte]{Liste der verwendeten
    Messgeräte. Die Genauigkeitsangaben beziehen sich auf die
    Standardabweichung $1\cdot \sigma$.}
\label{tab:bsp}
\begin{tabular}{ccccc}
\textbf{Messgerät} & \textbf{Hersteller} & \textbf{Typ} & \textbf{
    Verwendung} & \textbf{Genauigkeit}\\
\hline
\hline
\parbox[t]{0.2\linewidth}{\centering Spannungs-\\versorgung} & Voltmaker
    & HV2000 & \parbox[t]{0.2\linewidth}{\centering Spannungs-\\
    versorgung der\\Platine} & $\Delta U = \pm 5 $~mV \\ % Der parbox-
    Befehl ist erforderlich, damit ein Zeilenumbruch erzeugt werden kann.
     c-Spalten (zentriert) erlauben nicht automatisch einen Zeilenumpruch
    . Linksbündig gesetzte p-Spalten erlauben automatisch den
    Zeilenumbruch.
Strommessgerät & Currentcount & Hotamp 16 & \parbox[t]{0.2\linewidth}{ \
    centering Strommessung\\ am Versorgungspin\\ des \textmu C} & $\Delta
     I = \pm 0.1$~A \\
\hline
```

```
13 \end{tabular}
14 \end{table}
```

### E.2.4 Formeln

Formeln lassen sich in LaTeX ganz einfach schreiben. Es gibt unterschiedliche Umgebungen zum Schreiben von Formeln. Z. B. direkt im Text $v = s/t$ oder abgesetzt

$$F = m \cdot a$$

oder auch, wie in wissenschaftlichen Dokumenten üblich, nummeriert

$$P = \frac{U^2}{R} \quad . \tag{E.1}$$

Mit einem Label in Formel E.1 lassen sich natürlich auch Formeln im Text referenzieren. LaTeX verwendet im Formelmodus einen eigenen Schriftsatz, welcher entsprechend der gängigen Konventionen kursive Zeichen verwendet. Sollen im Formelmodus Einheiten in normaler Schriftart eingefügt werden, dann kann dies über den Befehl $\backslash mathrm\{\}$ erwirkt werden, wie im Quellcode von Formel E.2 zu sehen ist.

$$P = \frac{U^2}{R} = \frac{(100\ \mathrm{V})^2}{100\ \Omega} = 100\ \mathrm{W} \quad . \tag{E.2}$$

Zum direkten Vergleich sind die Einheiten in Formel E.3 falsch dargestellt:

$$P = \frac{U^2}{R} = \frac{(100\ V)^2}{100\,\Omega} = 100\,W \tag{E.3}$$

Zur einfachen Eingabe von Einheiten kann auch das Package $\backslash siunitx$ verwendet werden:

$$P = 100\ \mathrm{W} = 100\ \mathrm{J\,s^{-1}} \tag{E.4}$$

Das sind nur ein paar wenige Beispiele und es gibt sehr viele Packages, um Besonderheiten in Formeln realisieren zu können, z. B. mehrzeilige Formeln mit vertikaler Ausrichtung. Nennen Sie Formeln nur, wenn diese zum besseren Verständnis auch wirklich nützlich sind.

Folgende Befehle sind innerhalb von Formel-Umgebungen nützlich:

| | |
|---|---|
| *text*{} oder *mathrm*{} | Damit kann in Formel-Umgebung Text geschrieben werden. |
| \, \: \; \quad \qquad | Zusätzlichen Abstand zwischen Symbolen einfügen. |
| *notag* | Nummerierung einer bestimmten Formel ausschalten. |

Hier noch ein kleines Beispiel aus der Mathematik:

$$\sum_{n=1}^{\infty} f\left(x_n\right) \cdot \Delta x = \lim_{\Delta x \to 0} \frac{f\left(x_0 + \Delta x\right) - f\left(x_0\right)}{\Delta x} = \frac{\mathrm{d}f}{\mathrm{d}x} = \dot{f}(x) \tag{E.5}$$

Und abschließend ein Beispiel aus der Physik zum Induktionsgesetz:

$$\oint\limits_{\partial \mathcal{A}(t)} \vec{E} \cdot \mathrm{d}\vec{s} = - \int\limits_{\mathcal{A}(t)} \frac{\partial \vec{B}}{\partial t} \cdot \mathrm{d}\vec{A} \tag{E.6}$$

# Besprechungsnotizen

## E.3 9.Oktober.2024

- Spieletheorie erfassen
- Speicherplatz im uController wird begrenzt sein
- Spielalgorithmus (wann wird geschaut wo Steine liegen - immer das ganze Feld abcannen?)
- Zeitplan erstellen
- https://education.lego.com/de-de/downloads/spike-app/software/

## • Zeitplan

- KW42: Literaturrecherche, Erstellung eines groben Konzeptes (Skizzen, Funktionsweise)
- KW43: Zusammenbau des Roboters, Tests der mechanischen Komponenten (schrittweise Ansteuerung)
- KW44:

# Anforderungsliste

**Tabelle E.2:** Anforderungsliste W-Wünsch F-Forderung

| **Nr** | **Anforderung an das System** | **F/W** |
|---|---|---|
| | **Allgemein** | |
| - | Lage der Steine erkennen | F |
| - | Das Ende des Speils erkennen | F |
| - | Beweglich - Steine in jede Spalte | F |
| - | Magazin für Steine | F |
| - | Immer nur ein Stein pro Spielzug | F |
| - | Abwarten bis der Gegner sein Zug beendet hat | F |
| - | Begrezungen des Spielfeld erkennen | F |
| - | | F |
| - | | F |
| - | | F |