Tic-Tac-Toe-Roboter

Klara Uhlig, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Abstract*—Dieses Paper befasst sich mit der Entwicklung eines selbstständig spielenden Tic-Tac-Toe-Roboters. Er ist in der Lage, sein Spielfeld zu zeichnen, die Spielzüge seines Gegenspielers mithilfe einer Webcam zu erfassen und daraufhin über einem Algorithmus über seinen nächsten Spielzug zu entscheiden und sein Kreuz auf dem Spielfeld zu setzen. Er soll, wenn möglich, selbst gewinnen oder das Gewinnen des Gegners verhindern. Der Roboter besteht aus Lego-Mindstorms-Bauteilen, die Programmierung erfolgt über MatLab.**

***Schlagwörter*—Roboter, Lego-Mindstorms, Projektseminar, Tic-Tac-Toe.**

I. Einleitung

Das Ziel dieses Projektes war es, eine eigene Idee für einen Roboter zu finden und in kleinen Gruppen von zwei bis drei Personen umzusetzen. Zur Verfügung standen hierfür Lego-Mindstorms-Bauteile mit dem Steuerungscomputer NXT, sowie verschiedene Sensoren und eine Webcam. Zur Programmierung wurde zusätzlich MatLab, eine Software zum Lösen von mathematischen Problemen, Analysieren von Daten und entwickeln von Algorithmen verwendet. Es besteht außerdem die Möglichkeit graphische Benutzeroberflächen zu erstellen und zu gestalten, die für dieses Projekt auch genutzt wurde. Die Entscheidung fiel auf einen Tic-Tac-Toe spielenden Roboter, da die Komplexität des Strategiespieles im Vergleich zu anderen Spielen wie Schach oder Dame nicht zu hoch ist und im Rahmen des einwöchigen Projektes in den grundlegenden Funktionen umsetzbar ist. Anders als bei vergleichbaren Spielen am Computer wird mit Stift und Papier gespielt und nicht nur auf einer graphischen Benutzeroberfläche, was für ein realistischeres Spielgefühl sorgt.

II. Vorbetrachtungen

A. Spielregeln Tic-Tac-Toe

Tic-Tac-Toe ist ein klassisches Strategiespiel, das zwei Spieler gegeneinander spielen. Zuerst wird ein quadratisches 3 mal 3 Spielfeld aufgemalt, es können also 9 Kästchen belegt werden. Es wird abwechselnd gesetzt, üblicherweise setzt ein Spieler ein Kreuz “X” und der andere Spieler ein Kreis “O” in ein freies Feld. Ziel ist es nun, 3 seiner Kreuze oder Kreise in einer Reihe, Spalte oder Diagonalen zu setzen. Wem dies zuerst gelingt, hat gewonnen. Es darf immer nur ein Feld pro Spielzug besetzt werden.

Wie in Tabelle 1 zu sehen ist, ist die Wahrscheinlichkeit, dass der Spieler, der beginnt, auch gewinnt, deutlich höher als bei den andern beiden Spielverläufen.

TABELLE I
SPIELSTATISTIK

	Anzahl	Anteil
Anzahl aller Spielverläufe	255 168	100 %
Enden mit Sieg des ersten Spielers	131 184	51,41 %
Enden mit Sieg des zweiten Spielers	77 904	30,53 %
Enden Unentschieden	46 080	18,06 %

Statistik zum Sieg ohne Berücksichtigung der gleichen Spielzüge durch Drehung oder Spiegelung

B. Ähnliche Roboter

In der Vorbetrachtung wurden zunächst auch noch andere Roboter mit ähnlichen Funktionsweisen betrachtet. Grundlegend muss sich der Roboter nämlich in allen drei Dimensionen bewegen können, also vor- und zurückfahren, den Stift nach links und rechts fahren, sowie den Stift heben und absenken. Lego-Roboter, die als Drucker mit Stift und Papier arbeiten, sowie 3D-Stanzer, die Bilder in Blumenschaum oder ähnliches stanzen, wiesen den gewünschten Aufbau und eine ähnliche Funktionsweise auf. Hierbei wurden häufig Konstruktionen mit Schienen, auf denen Zahnräder laufen, verwendet, was aufgrund der mangelnden Bauteile aber nicht in Frage kam. Dennoch war die Betrachtung dieser Roboter hilfreich, da diese meist eine Übersetzung der Motordrehung über verschiedene Zahnräder aufwiesen, um die Genauigkeit der Motoren zu verbessern.

III. Umsetzung des Roboters

Die Idee war es, einen Tic-Tac-Toe spielenden Roboter zu entwickeln, der einen menschlichen Spielpartner ersetzen kann. Hierzu musste nicht nur ein Spielalgorithmus entwickelt werden, sondern auch das Ansteuern der einzelnen Felder mit einem selbst gebauten Roboter programmiert werden.

DOI: 10.24352/UB.OVGU-2019-071 Lizenz: CC BY-SA 4.0

A. Der mechanische Aufbau

Der Roboter besteht aus 3 Motoren. Der erste Motor treibt die Reifen an, die zum Vor- und Zurückfahren da sind. Er befindet sich ganz hinten am Roboter. Der zweite Motor ist für die Bewegung nach links und rechts verantwortlich und sitzt unter dem NXT. Der letzte Motor zum Absetzten und Hochfahren des Stiftes ist über dem NXT verbaut.

Abbildung 1: Beispiel-GUI

Die Kamera hängt in einer festen Position über dem Spielfeld. Für jedes der 9 Felder sind feste Koordinaten festgelegt. Vor dem Spielzug des Menschen wird ein Bild gemacht und der gerundete Grauwert der einzelnen Felder ermittelt. Nach einer 10-sekündigen Pause wird ein zweites Bild gemacht und die neuen Grauwerte der Spielfelder ermittelt. Die beiden Werte werden miteinander verglichen und das mit der größten Abweichung erkannt. Danach wird der Spielzug des Menschen in die GUI (Graphic User Interface) als “X” eingetragen. Da eine Erkennung eines Kreuzes auf dem Papier aufgrund der Qualität der Kamera schwierig ist, wird vom Menschen kein Kreuz gesetzt, sondern das Spielfeld ausgemalt um die Änderung des Grauwertes zu erhöhen, wie auch in Abbildung 3 zu sehen ist.

B. Programmierung

Am Anfang des Programms wird programmintern eine 3x3-Matrix M mit Nullen erstellt. Für jeden Spielzug des Menschen wird eine 1 in das jeweilige Feld der Matrix eingetragen, für jeden Spielzug des Computers eine 4. Die Werte wurden so gewählt, damit die später berechneten Summen der Zeilen, Spalten und Diagonalen eindeutig sind. Für die Situation aus Abbildung 1 ergibt sich also die folgende Matrix:

$$M = \begin{bmatrix} 0 & 0 & 1 \\ 4 & 4 & 4 \\ 1 & 0 & 1 \end{bmatrix} \quad (1)$$

Um seinen eigenen Zug zu setzen, werden die Zeilen, Spalten und Diagonalen der Matrix summiert. Danach wird geprüft, ob das Gewinnen des Roboters möglich ist, also ob die Summe in einer Zeile, Spalte oder Diagonalen gleich 8 ist. Wenn dies der Fall ist, wird das verbleibende Feld besetzt (siehe Anhang). Hierfür wird geprüft, welches der Felder nicht besetzt ist,

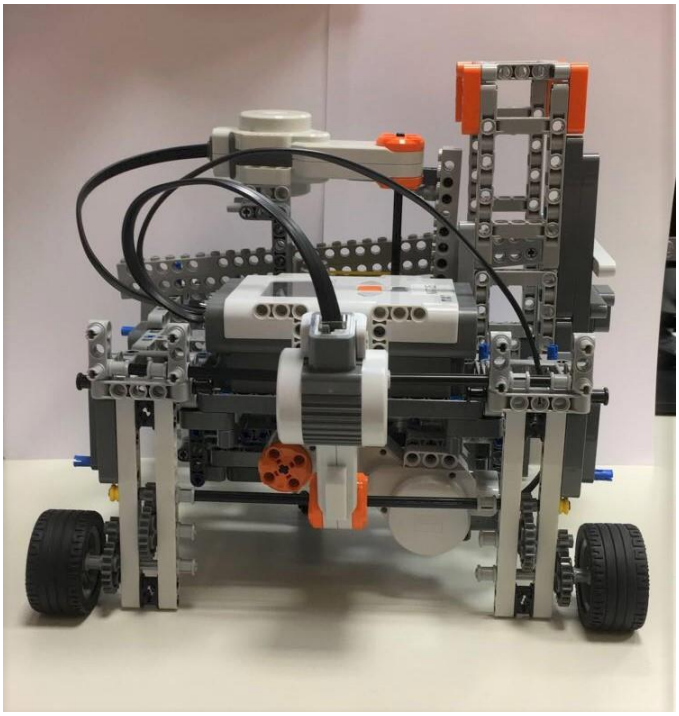

Abbildung 2: Roboter von hinten

also welches Feld noch keinen weiteren Text wie “X” oder “O” hat. Außerdem wird mit der Variable “Spielzug” dafür gesorgt, dass nur ein Spielzug gemacht wird, da als Bedingung für alle weiteren Züge gelten muss, dass diese gleich Null ist. Wird ein Spielzug gemacht, wird diese um eins erhöht. Auf der GUI wird dann ein “O” gesetzt. Außerdem wird geprüft, ob der Mensch gewinnen kann, also ob die Summe in einer Spalte, Zeile oder Diagonalen 2 ist. In diesem Fall wird auch das verbleibende Feld besetzt. Sollte das Gewinnen beider Parteien nicht möglich sein, wird ein zufälliges freies Feld besetzt. Nachdem der Spielzug auf der GUI angezeigt wird, fährt der Roboter zu dem jeweiligen Feld hin.

C. Anfahren der Spielfelder

Um ein Spielfeld anzufahren, müssen alle 3 Motoren richtig programmiert sein. Zunächst muss der Roboter aus seiner Ausgangsposition zum Spielfeld hinfahren. Hierzu wurden die Felder von 1 bis 9 durchnummeriert. Das Feld oben links ist dann zum Beispiel das Feld 1 und das Feld unten rechts das Feld Nummer 9. In der Ausgangsposition ist der Stift ganz rechts, also bei Feld 3. Um nun zum Beispiel zu Feld 7 unten links in der Ecke zu gelangen, muss der Roboter also noch zwei weitere

Kästchen nach vorne und der Stift zwei Kästchen nach links bewegt werden. Dort wird dann die Funktion zum zeichnen der Kreuze aufgerufen, in der die Motoren zum Vorwärts- und Seitwärtsfahren gleichzeitig angesteuert werden. Um die Motoren und den NXT überhaupt über MatLab nutzen zu können, wurde ein zusätzlicher Ordner mit entsprechenden Befehlen und Funktionen bereitgestellt. Mit diesen Funktionen kann die Geschwindigkeit der Motoren, sowie ein Tacho Limit eingestellt werden, welches den Motor nach einer gewissen Zeit stoppt. Jeder Befehl muss anschließend an den NXT zurückgesendet werden. Es besteht außerdem die Möglichkeit, auf einen anderen Motor zu warten, was zum Beispiel für das Zeichnen des Feldes genutzt wurde.

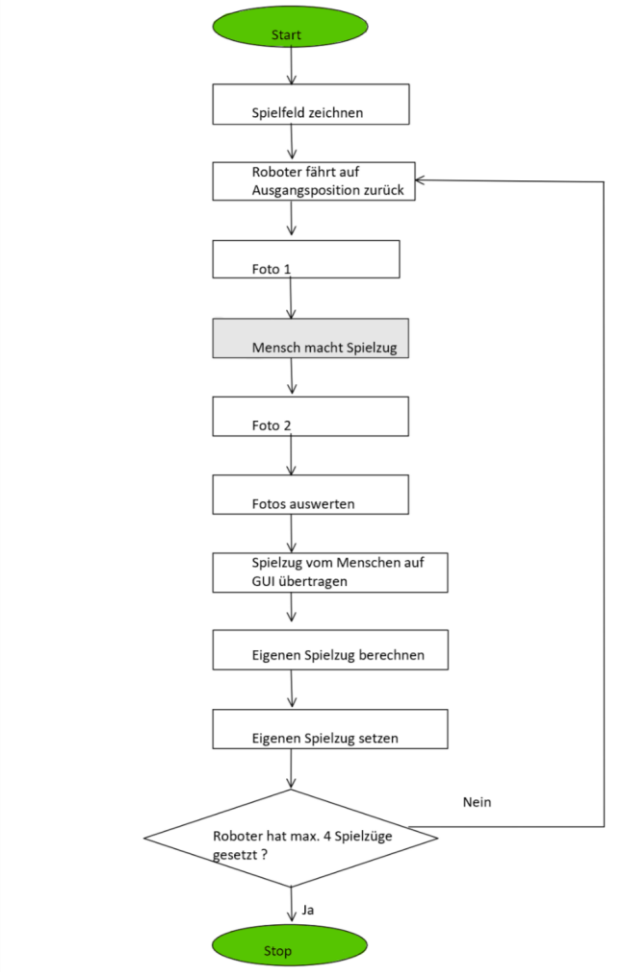

Abbildung 3: Programmablaufplan

IV. ERGEBNISDISKUSSION

Das größte Problem des Roboters sind die leistungsschwachen Motoren von Lego. Durch einen deutlichen Unterschied bei voll geladener oder halb geladener NXT Station ist es schwierig, den Roboter genau zu kalibrieren. Des weiteren bricht der NXT das Programm an einigen Stellen ab, wenn die Anzahl der Motorenansteuerungen hintereinander zu hoch ist. Auch durch kurze Pausen konnte das Problem nicht gelöst werden. Das führt zum Beispiel dazu, dass der Roboter nicht auf seine Ausgangsposition zurückfährt und in den folgenden Zügen nicht mehr die richtigen Felder anfahren kann. Ein weiteres Problem ist die derzeit feste Kalibrierung der Kamera. Wird die Kamera um ein paar Zentimeter falsch ausgerichtet, so misst diese falsche Werte bzw. die Spielfelder stimmen nicht mehr überein. Dies könnte man durch eine automatische Spielfelderkennung lösen, die dann allerdings wieder das Problem hätte, dass bei Versagen der Motoren ein "falsches" Feld eingescannt wird, da der Roboter zu fest vorgegeben Feldern hinfährt. Auch die Größe der Kreuze könnte sich nicht der eventuell ändernden Größe des Spielfeldes anpassen. Bei kleinerem Spielfeld wären die Kreuze dann zum Beispiel zu groß für die Felder.

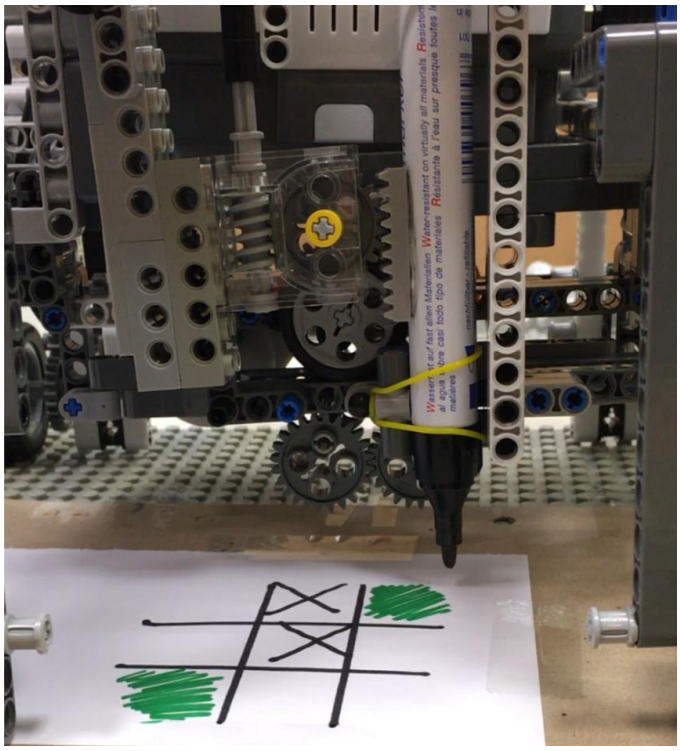

Abbildung 4: Front des Roboters mit Halterung des Stiftes und selbstgezeichnetem Spielfeld mit Zügen

Auch bei der Mechanik gibt es noch Probleme. So ist die eine Seite des Roboters schwerer als die andere, was dazu führt, dass der Stift an der einen Seite deutlich mehr aufdrückt als auf der anderen Seite. Dies kann zum Verhaken des Motors führen und der Stift fährt nicht an seine Ausgangsposition zurück. Grund dafür ist der Motor zum Absenken und Hochfahren des Stiftes. Dieser muss nämlich mit dem Stift nach links und rechts fahren. Hier wäre zum Beispiel ein Schienensystem oder ein weiteres Rad auf dem Roboter eine Lösung, damit das Gewicht nicht nur auf der Achse für die Querbewegung lastet.
Ein weiteres Problem ist das Erkennen des Spielendes, wenn ein Spieler gewonnen hat. Derzeit ist der Algorithmus so konzipiert, dass er nach 4 Spielzügen, also der maximalen Anzahl an Zügen, die der Roboter machen kann, das Programm

stoppt. Das führt auch dazu, dass nachdem der Mensch gewonnen hat, der Computer noch einen Zug errechnet und dann eventuell ausgibt, dass er selbst gewonnen hat, obwohl dies nicht der Fall ist.
Einige Probleme konnten im Verlauf des Projektes aber auch erfolgreich gelöst werden. Dazu zählt zum Beispiel, dass der Computer nur einen Zug setzen kann und nicht mehrere, wenn er gewinnen und der Gegenspieler im nächsten Zug auch gewinnen könnte. Auch das zunächst geplante Schienensystem, um vor- und zurückzufahren, konnte aufgrund von einer Bauteilknappheit gut durch Räder ersetzt werden. Auch die Halterung des Stiftes sorgte für Probleme. Die zuerst verwendete Halterung mit Gummibändern erwies sich als zu flexibel, denn bei jedem Hoch- und Runterfahren des Stiftes hatten die Gummibänder einen nicht vorhersehbaren Spielraum. Das Problem wurde gelöst, indem eine Zahnradschiene an den Stift festgeklebt wurde, wie in Abbildung 4 zu sehen ist.
Weitere Verbesserungsmöglichkeiten wären zum Beispiel eine Auswahl auf der GUI, ob der Mensch oder Computer anfangen soll oder die Auswahl zwischen verschiedenen Schwierigkeitsstufen. So könnten zum Beispiel auch die Spieltaktiken des Menschen ausgewertet und gezielt verhindert werden. Dazu müssten dann keine zufälligen Felder sondern gezielt die Felder belegt werden, die man zur Erstellung von Zwickmühlen braucht.

V. Zusammenfassung und Fazit

Zusammenfassend ist das Projekt ein Erfolg, da ein Roboter entwickelt wurde, der mit einem Menschen Tic-Tac-Toe spielen kann. Die Mechanik weist noch einige Probleme auf, die teilweise durch eine bessere Konstruktion gelöst werden könnten. Das größere Problem sind aber die relativ leistungsschwachen Motoren, die Lego-Mindstorms bereitstellt. Das größere Problem ist der NXT-Steuerungsroboter, da er nur eine relativ geringe Anzahl an Motoransteuerungen nacheinander ausführen kann. Werden zu viele Motoren nacheinander angesteuert, kommt es zum kurzzeitigen Abbruch des Programms und einige Motoren werden nicht angesteuert. Das führt dazu, dass der Roboter nicht mehr in seine Ausgangsposition zurückfährt und nicht mehr die richtigen Felder anfahren kann. Das Programm weist noch einige Probleme auf, zum Beispiel, dass es nicht automatisch abbricht, wenn der Mensch oder Roboter gewonnen hat. Weitere kleinere Probleme, wie zum Beispiel die Halterung des Stiftes oder, dass der Roboter maximal ein Zeichen pro Spielzug macht, konnten im Verlauf des Projektes behoben werden. Derzeitig kann der Roboter also mit einem menschlichen Spielpartner Tic-Tac-Toe auf einem Blatt Papier spielen. Der Roboter kann nach dem Start des Programmes das Spielfeld selbstständig zeichnen und die Züge des Menschen, der derzeit immer beginnt, mithilfe einer Kamera erkennen und darauf reagieren. Die Spielzüge vom Roboter und vom Menschen werden hierbei auch auf der GUI angezeigt, ebenso wie die Anzeige, ob der Mensch gewonnen oder verloren hat.

Anhang

A. *Quellcode, um letztes Feld zu belegen*

```
% Beispielhaft für die erste Zeile:
   Spielzug = 0;
  if ( SummeErsteZeile == 8 && spielzug ==0)
      if(get(handles.pushbutton1, 'String') ==' ')
          set(handles.pushbutton1, 'String', 'O');
          handles.matrix(1,1) = 4;
           Spielzug = 1;
           handles.feld = 1;
      end
      if(get(handles.pushbutton2, 'String') ==' ')
          set(handles.pushbutton2, 'String', 'O');
          handles.matrix(1,2) = 4;
         Spielzug = 1;
         handles.feld = 2;
      end
      if(get(handles.pushbutton3, 'String') ==' ')
          set(handles.pushbutton3, 'String', 'O');
          handles.matrix(1,3) = 4;
          Spielzug = 1;
          handles.feld = 3;
      end
  end
```

Literaturverzeichnis

[1-4] Eigene Aufnahmen

[5] Norman Do, How to Win at TicTacToe, The Australian Mathematical Society, Gazette, Volume 32 Number 3, July 2005, S. 151