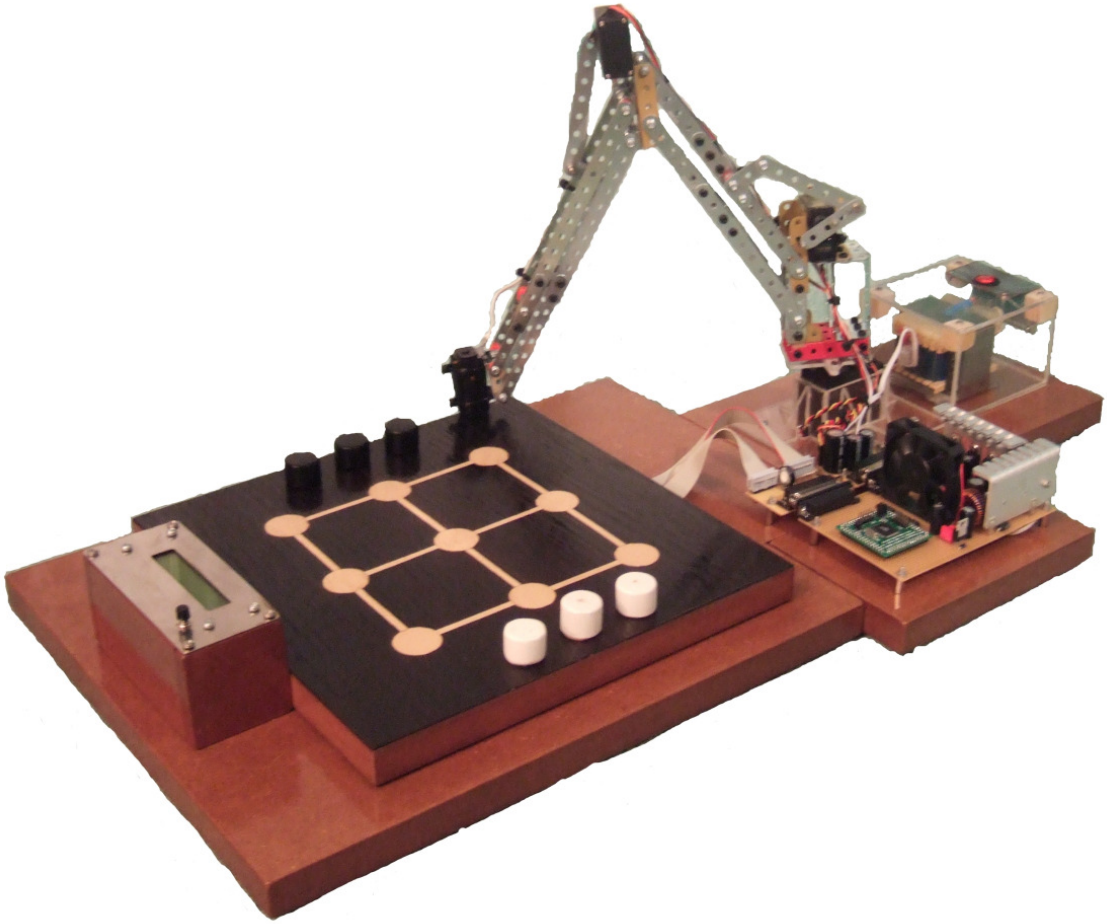
Konstruktion eines interaktiv spielenden Roboters

Programmierung und Bau eines Roboters, der in der Lage ist, „einfache Mühle“ gegen einen humanen Gegner zu spielen



Maturaarbeit

2006/2007

Mario Mauerer

Freies Gymnasium Zürich

Klasse O61

Betreuer: Th.Wurms

Inhaltsverzeichnis **Seite**

1. Vorwort

Ein Spiel – von Menschen für Menschen erfunden, um für Unterhaltung und Verbundenheit zu sorgen. Dieses Phänomen reicht vom einfachen „Memory“ bis zum weltbewegenden Fussball.
Doch kann man so ein Spiel auch rein mathematisch auffassen? Kann man somit einer Maschine beibringen, gegen einen Menschen zu spielen?
Dies eine Frage, die mich als Hobby-Elektroniker natürlich auch interessierte, und deshalb hab ich mich entschieden, die von meinem Betreuer vorgeschlagene Aufgabe anzunehmen – einem Roboter das Spielen beizubringen.

Dies ist mir geglückt, der Roboter ist in der Lage, das Spiel „einfache Mühle“ gegen einen Menschen zu spielen, und dies tut er erstaunlich gut.
Ich möchte mich hier auch bei denjenigen Bedanken, die mir beim Bau des Roboters behilflich waren, speziell erwähnen möchte ich hier Michael Leutenegger, der mir seine Werkstatt, Material und Erfahrung in der Holzbearbeitung bereitstellte.

2. Einleitung

Der Vorschlag von meinem Betreuer, einen spielenden Roboter zu bauen, war für mich verlockend, denn es war eine anspruchsvolle, neuartige Aufgabe.
Es galt, neu erworbenes Wissen mit Erfahrung zu verknüpfen. Denn als Hobby-Elektroniker wusste ich, wie man einen autonomen Roboter aufzubauen hat.
Doch ich wusste nicht, wie man einem Roboter das Spielen beibringen sollte, sprich, ich kannte den mathematischen Hintergrund hinter einem Spiel nicht. Diesen galt es zu erforschen und mit der Technik zu verbinden. Zudem hatte ich mich noch nie an ein so komplexes Projekt gewagt, bei dem so viele Faktoren stimmen müssen, damit es gut funktioniert.

Das Ziel war es, dass der Roboter autonom gegen einen Menschen spielen konnte. Das heisst, er muss mit korrekten Zügen auf diejenigen des menschlichen Spielers reagieren und versuchen, ihn zu besiegen.
Ein erstes Problem stellte sich in der Wahl des Spieles. Dabei musste beachtet werden, dass ein Spiel gewählt wurde, in dem beide Spieler die vollständige Information über den Spielzustand haben, das heisst, jeder Spieler kennt die Situation des Gegners. Bekannte Beispiele für solche Spiele sind Schach, Mühle, 4 gewinnt oder TicTacToe.
Andere Spiele, bei denen der Zufall eine Rolle spielt und bei denen man nicht weiss, in welcher Situation sich der Gegner befindet, konnten nicht verwendet werden, denn sie sind nicht gut berechenbar. Beispiele hierfür wären Kartenspiele wie Jass oder Poker.
Nach langen Diskussionen mit dem Betreuer entdeckten wir das Spiel „einfache Mühle“, welches sehr gut für das Projekt geeignet schien, denn es ist nicht zu komplex und auch ein Roboter, der es spielen sollte, war gut zu konstruieren.

So entschloss ich mich, einen Roboter zu bauen, der in der Lage ist, „einfache Mühle“ gegen einen humanen Gegner zu spielen.

Diese Dokumentation befasst sich mit diesem Roboter, sie erklärt das Spiel, den Aufbau des Roboters selbst und den mathematischen Hintergrund hinter der Steuerung.

3. Aufbau des Roboters

3.1 Das Spiel „einfache Mühle“

„Einfache Mühle“ ist eine reduzierte Version der „echten“, bekannteren Mühleversion. Im Gegensatz zur „Mühle“ gibt es bei der „einfachen Mühle“ nur 9 Felder. Abb.1 zeigt die schematische Ansicht des Spielfeldes.

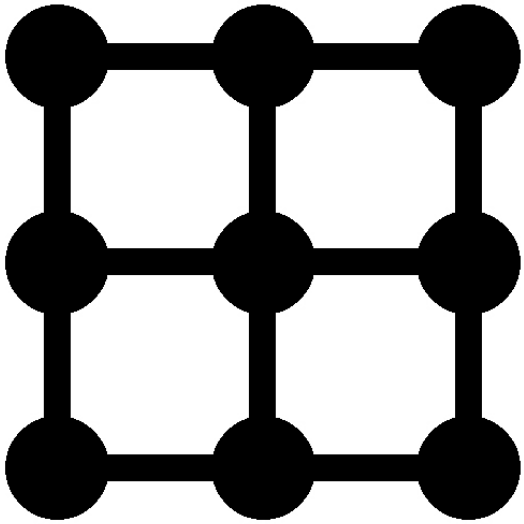


Abb.1: schematisches Spielfeld

Jeder Spieler hat 3 Spielfiguren, die sich zu Beginn des Spieles noch nicht auf den Feldern befinden.
Ziel des Spieles ist es, 3 eigene Spielfiguren in einer „Dreierreihe“ auf den 9 Feldern zu positionieren. Eine diagonale Dreierreihe zählt dabei aber nicht.
Ist dieses Ziel erreicht, gewinnt derjenige mit der Dreierreihe und das Spiel ist beendet.

Zu Beginn werden die Spielfiguren von ausserhalb auf das Spielfeld gesetzt. Sind alle 6 Figuren positioniert, darf gesprungen werden. Dabei darf mit den eigenen Figuren auf jedes freie Feld gesprungen werden. Solange sich nicht alle Figuren auf dem Feld befinden, darf aber noch nicht gesprungen werden.
Sobald ein Spieler eine Dreierreihe mit seinen Figuren gesetzt hat, ist das Spiel beendet und er gewinnt.

3.2 Das Steuerungsorgan

Damit so ein Roboter überhaupt funktionieren kann, braucht es ein Programm, welches die nötigen Berechnungen ausführt und die einzelnen Komponenten kontrolliert und steuert. (Zur Software: siehe Kapitel 4) Diese Software beinhaltet auch die Mathematik, die es dem Roboter erlaubt, ein Spiel korrekt zu spielen.

Die Software muss im Roboter gespeichert werden. Hierfür bietet sich ein so genannter Mikrocontroller bestens an. Er übernimmt nicht nur die Speicherung der Software, sondern leitet auch deren Ausführung, er stellt also die Schnittstelle zwischen der Software und der Hardware des Roboters dar.
Unter einem Mikrocontroller kann man sich einen kompletten PC in Miniaturformat vorstellen, er beinhaltet eine CPU, die die Software ausführt, einen programmierbaren Flash-Speicher für die Software, ein kleiner RAM-Speicher und besitzt diverse Schnittstellen zur Aussenwelt. So ein Mikrocontroller kann enorm viel, sie sind deshalb sehr vielfältig und man findet sie in vielen Alltagsgegenständen wie z.B. Radios, Waschmaschinen oder Kaffeemaschinen. Ein Mikrocontroller besteht aus nur einem einzigen Chip. Der Zugang zu den inneren Strukturen wird über Pins(=Anschlusskontakte) gewährleistet, welche mit der Schaltung verlötet werden. Mit ihnen können Daten oder Spannungspotentiale eingelesen oder ausgegeben werden. Ein Mikrocontroller ist also eine bequeme Lösung, um auch schwierigere Programme ausführen zu können, und gleichzeitig einen engen Bezug zum Gerät, welches gesteuert wird, herzustellen.

Für meine Arbeit verwende ich einen „ATmega128“ Mikrocontroller der Firma Atmel, im Vergleich zu anderen Controllern bietet er sehr viel Speicherplatz und Funktionen. Hier ein kurzer Steckbrief dieses Controllers:

- 128kB Flash-Speicher für das Programm (Vergleich: moderner PC: 100GB)
- Die CPU wird mit 16 MHz getaktet (Vergleich: moderner PC: 2GHz)
- Die CPU arbeitet mit 8 Bit Datenbreite (Vergleich: moderner PC: 32 Bit)
- Er besitzt 53 I/O-Pins(Input-Output), die eine Kommunikation mit dem Chip erlauben und mit denen sich z.B. Potentiale einlesen und ausgeben lassen.
- Er wird mit 5V betrieben und verbraucht etwa 50mA Strom
- Er kostet etwa 11.- CHF

Der Controller kann direkt im Roboter programmiert werden, er muss dazu nicht aus der Schaltung entfernt werden. Programmiert wird er über die parallele Schnittstelle eines PC, an die er direkt angeschlossen werden kann. Ein spezielles Programm auf dem PC überträgt dann die Software in den Flash-Speicher des Controllers.
Diese Software beinhaltet die vollständige Steuerung des Roboters, der Mikrocontroller befolgt lediglich die Anweisungen, die er von der Software erhält. Die Software kann den Chip Spannungspotentiale einlesen oder ausgeben lassen, sie

kann den Controller aber auch komplexe Berechnungen anstellen lassen oder andere interne Module des Chips bedienen. Somit lassen sich sehr komplexe Anwendungen realisieren.
Sobald der Chip mit Spannung versorgt wird, beginnt er mit der Abarbeitung des intern gespeicherten Programms. Schaltet man die Stromversorgung ab und danach wieder an, dann beginnt er wieder von vorne im Programm.
Mit seinen Pins kann der Controller nichts anderes machen, als sie auf „High“ oder „Low“ zu schalten, er kann also 5V anliegen lassen, oder den Pin auf 0V (=GND, Ground, Masse) ziehen. Diese beiden Zustände sind die Nullen und Einsen in der Digitaltechnik, mit ihnen lassen sich auch komplexe Daten gut übertragen.

Da der ATmega128 ein spezielles Gehäuse hat, welches ich nicht löten kann, die Pins liegen zu nahe beieinander, kaufte ich ihn fertig aufgelötet auf einem „Adapterboard“, welches die Pins des Controllers nach aussen an Kontakte führt, die ich verlöten kann. Zusätzlich befinden sich auf diesem Adapterboard noch ein paar wenige externe Bauteile, die der Chip zum korrekten funktionieren braucht, und die Anschlüsse für den Programmieradapter, damit man den Chip an dem PC anschliessen kann. Die Abb.2 zeigt den ATmega128 auf seinem Adapterboard fertig verlötet. Die linke Steckerleiste auf dem Bild ist für den Anschluss auf dem PC, und das ovale, silberne Bauteil vor dem Controller ist der Quarz, welcher die 16MHz Taktfrequenz liefert. Die Anschlüsse an dieses Adapterboard sind rundherum an der Kante des grünen Boards verteilt.

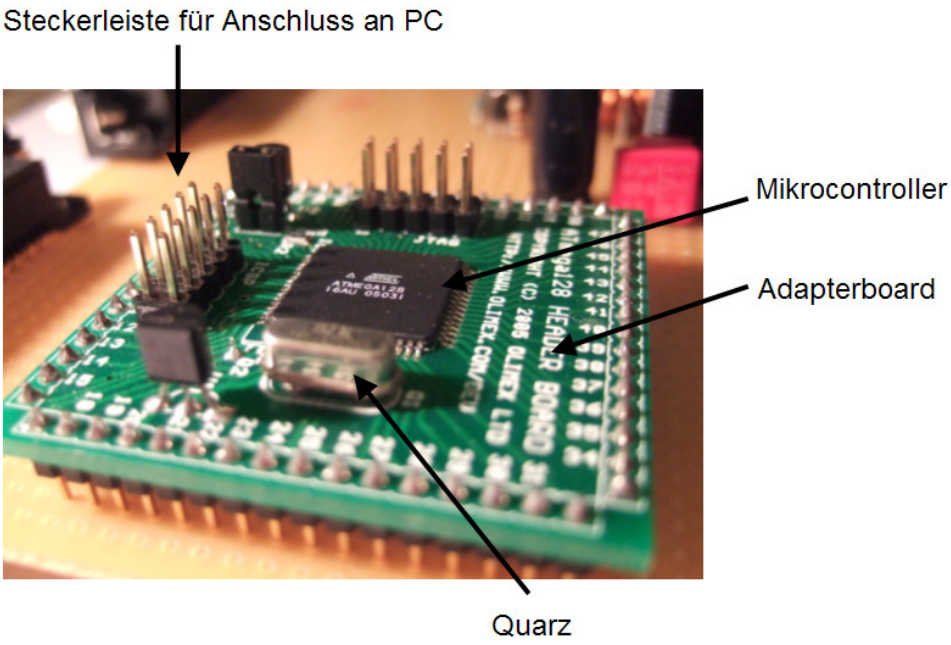


Abb.2: Der Mikrocontroller auf dem Adapterboard aufgelötet

3.3 Der Gesamtaufbau

Der Roboter besteht aus 3 zusammensteckbaren Modulen. Ich habe diese Modulbauweise gewählt, da man den Roboter ansonsten nur sehr schlecht hätte transportieren können, da er doch recht gross und schwer wäre. Die 3 Module sind wie folgt aufgebaut:

- **Greifarm-Modul:**
  Auf ihm befinden sich der Greifarm, welcher die Spielfiguren bewegt, und der Transformator, er setzt die Netzspannung auf eine niedere, ungefährliche Spannung herab. An ihm wird das Netzkabel angeschlossen. Am Greifarm befindet sich auch ein Elektromagnet, mit welchem die Spielfiguren bewegt werden können.
- **Mainboard-Modul:**
  Auf ihm befindet sich die Spannungsversorgung und der Mikrocontroller, es ist sozusagen das „Gehirn“ des Roboters, von hier werden alle Komponenten verwaltet und gesteuert. Die Spannungsversorgung stellt eine konstante, aber dennoch belastbare Spannungsquelle dar, von ihr beziehen alle Komponenten des Roboters die elektrische Leistung, die sie zum arbeiten benötigen.
- **Spielfeld-Modul:**
  Auf diesem Modul befinden sich das Spielfeld und ein Flüssigkristall-Display (LCD), welches es dem Roboter ermöglicht, mit dem Benutzer zu kommunizieren.

Jedes Modul besteht aus einer Grundplatte aus Holz (eine Art feiner Pressspan), auf der sich dann die funktionalen Komponenten wie Greifarm, Spielfeld oder Mainboard befinden.
Diese einzelnen Module sind über elektrische Steckkontakte (RS232-Buchsen) mit dem Mainboard-Modul verbunden. Somit werden die einzelnen Module auch elektrisch leitend verbunden, damit man verschiedene Spannungen oder Signale von einem Modul auf das andere übertragen kann.
Das Greifarm-Modul wird zusätzlich mit zwei Dübeln in das Spielfeld-Modul gesteckt, was eine gute Stabilität gewährleistet. Abb.3 zeigt die einzelnen Module. (nicht zusammengesteckt)

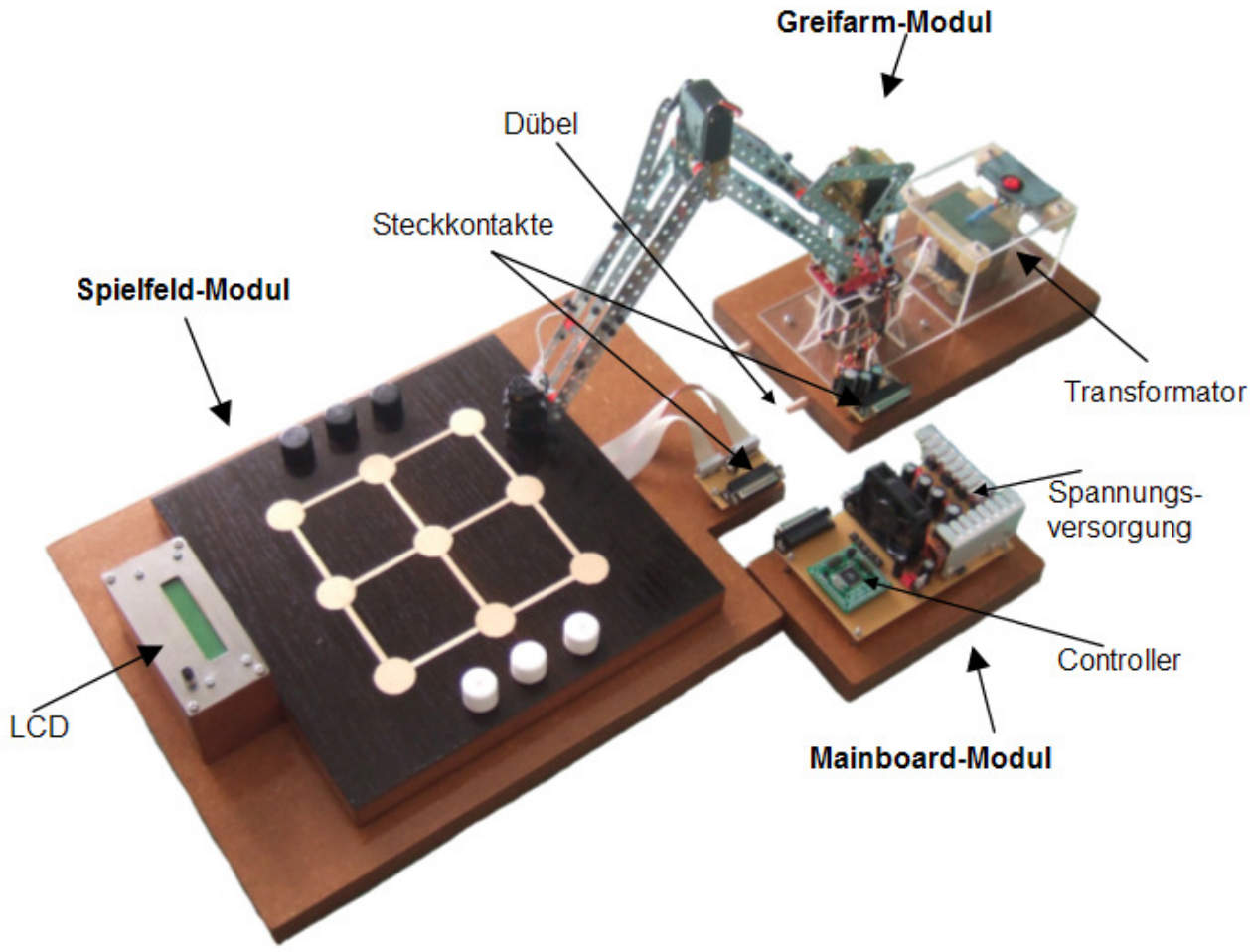


Abb.3: Die einzelnen Module des Roboters

3.4 Das Mainboard-Modul

Das Mainboard-Modul besteht neben der Grundplatte aus Holz aus einer einzigen Platine. Auf der Platine werden die einzelnen elektronischen Bauteile zu Schaltkreisen verlötet.
Auf dem Mainboard-Modul befinden sich zwei funktionale Komponenten des Roboters, der Mikrocontroller und die Spannungsversorgung.
Dieses Modul wird über zwei Steckkontakte, einer für das Spielfeld-Modul und einer für das Greifarm-Modul, an den Roboter angesteckt.
Die Spannungsversorgung muss gekühlt werden, da sich die Regler erhitzen, deshalb sind sie an Kühlkörpern befestigt und ein Lüfter sorgt dafür, dass sich die Kühlkörper nicht zu stark erhitzen. Die Abb.4 gibt einen Überblick über das Mainboard-Modul.

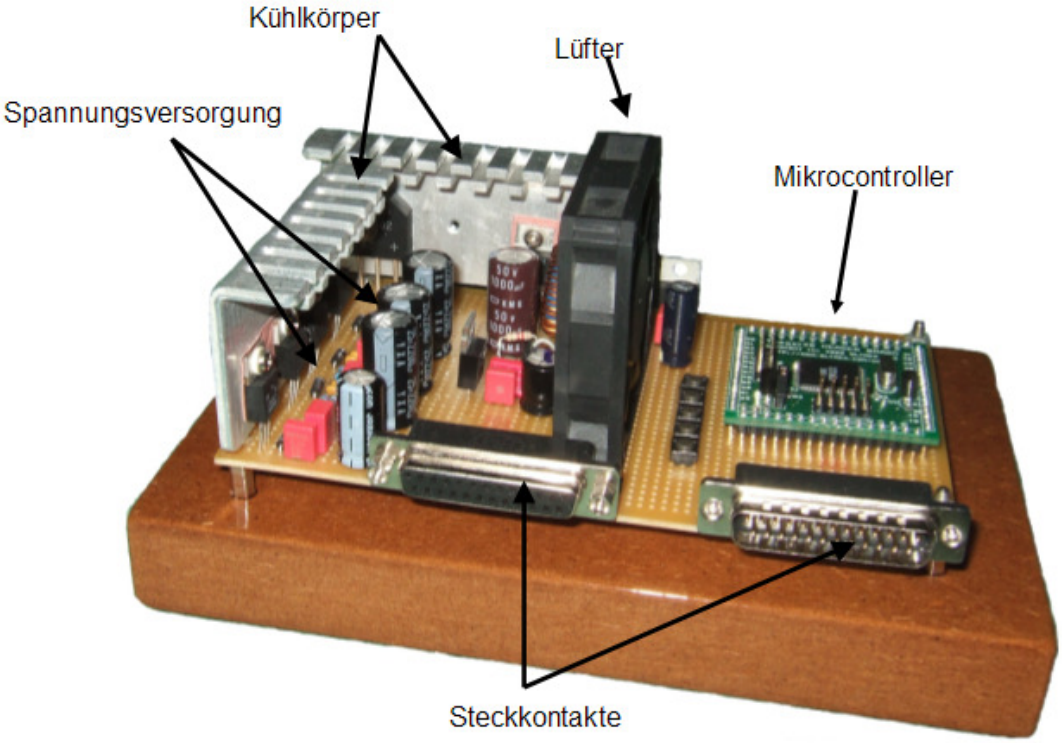


Abb.4: Überblick über das Mainboard-Modul

Im Folgenden soll nun die Spannungsversorgung genauer erklärt werden.

Die einzelnen elektrischen Komponenten des Roboters benötigen verschieden hohe Gleichspannungen, damit sie korrekt funktionieren. So brauchen z.B. der Mikrocontroller und das LCD 5V, der Antrieb des Greifarmes benötigt hingegen zwischen 4V und 6V.
Doch diese elektrische Leistung muss von irgendwoher stammen, z.B. aus der Steckdose oder aus einem Akku. Ich habe auf Akkus verzichtet, da sie aufgeladen werden müssen und nur eine begrenzte Lebensdauer haben. Deshalb wird die elektrische Leistung aus der Steckdose bezogen. Doch diese liefert 230V Wechselspannung, was eine lebensgefährlich hohe Spannung ist und von den Bauteilen nicht verarbeitet werden kann. Deshalb muss diese Spannung zu kleineren Spannungen umgewandelt werden. (Z.B. zu 5V Gleichspannung)
Der folgende Schaltplan (Abb.5) zeigt den schematischen Aufbau der kompletten Spannungsversorgung.

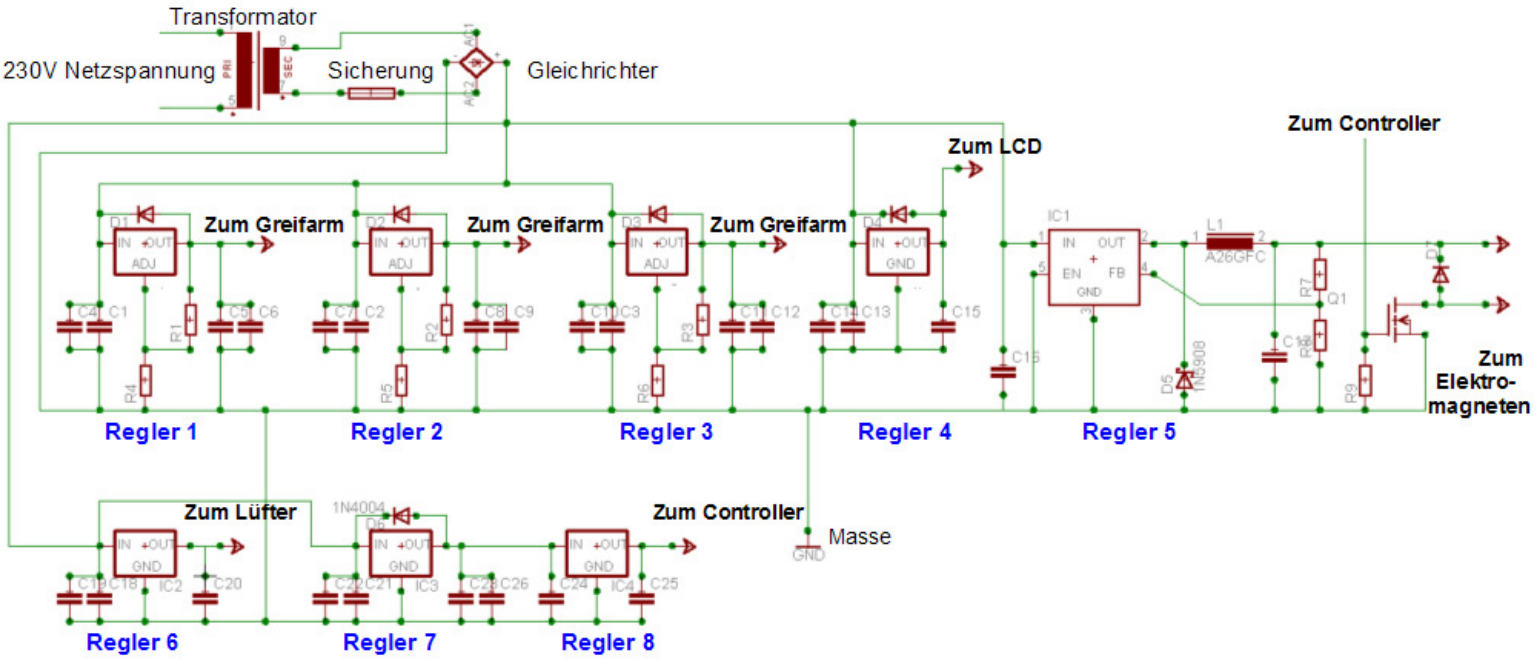


Abb.5: der komplette Schaltplan der Spannungsversorgung

Doch, was zeigt dieser Schaltplan, und wie funktioniert diese Spannungsversorgung? Dafür sollen zuerst die Symbole benannt werden, dafür dient Abb.6.

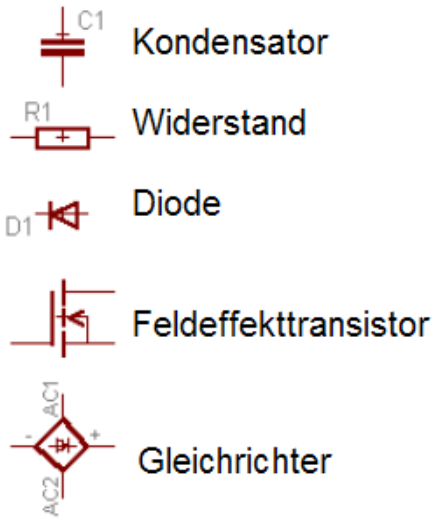


Abb.6: Die Bauteile

Kondensator: Er dient zum kurzzeitigen speichern von elektrischer Energie. Er glättet die Spannung, die an ihm anliegt, und er kann für kurze Momente sehr hohe Ströme liefern.

Widerstand: Er leistet dem elektrischen Strom widerstand, er limitiert so den fliessenden Strom.

Diode: Sie lässt den Strom nur in eine Richtung durch.

Feldeffekttransistor: Er lässt den Strom nur fliessen, wenn am linken Anschluss (=“Gate“) 5V anliegen, ansonsten sperrt er.

Gleichrichter: Er verwandelt eine Wechselspannung in eine (pulsierende) Gleichspannung.

Nun soll die Funktion der Spannungsversorgung besprochen werden:

Der Transformator und der Gleichrichter:

Wie schon erwähnt, wird das Stromnetz als Quelle elektrischer Energie für den Roboter benutzt. Doch die Netzspannung von 230V Wechselspannung (die Polung wechselt 50 mal in der Sekunde) ist für die Bauteile nicht geeignet. Zum einen ist sie viel zu gross, und zum anderen wechselt die Polung. (Wechselspannung). Also muss diese Spannung zu üblicheren Spannungen heruntertransformiert werden. Dafür verwendet man einen Transformator. An ihn werden die 230V Netzspannung angeschlossen. Mittels induktiver Vorgänge wird diese Spannung auf 17V heruntertransformiert. So eine Spannung lässt sich viel besser verarbeiten. Doch der Transformator gibt ebenfalls Wechselspannung heraus, sprich, auch die Polung dieser 17V wechselt 50 mal in der Sekunde. Mittels eines Gleichrichters muss dies geändert werden, denn die elektronischen Komponenten wie z.B. der Mikrocontroller brauchen eine Gleichspannung.

Der Gleichrichter, der direkt an den Transformator angeschlossen wird, wandelt die Wechselspannung mittels 4 Dioden in eine pulsierende Gleichspannung um. An den Ausgängen des Gleichrichters kann man nun die 17V abgreifen, nun aber gleichgerichtet. Nach dem Transformator befindet sich noch eine Sicherung, die verhindert, dass im Fehlerfall (Kurzschluss) der Transformator abbrennt, indem sie den Stromkreis unterbricht.

Diese Gleichspannung wechselt die Polung nicht mehr, aber sie ist nicht konstant, sie pulsiert nämlich noch. Dies ist deshalb, da die Wechselspannung auch pulsiert (Sinusschwingung), dabei aber zusätzlich noch die Polung wechselt. Nun wechselt sie nicht mehr die Polung, aber das „Pulsieren“ bleibt noch vorhanden. (Überreste der Sinusschwingung des Stromnetzes)

Der Trafo wird über einen Kippschalter mit dem Stromnetz verbunden. Dieser Kippschalter ist sogleich auch der Hauptschalter des gesamten Roboters.

Die elektronischen Bauteile, wie z.B. der Mikrocontroller oder das LCD brauchen aber eine konstante Gleichspannung. Deshalb werden so genannte Spannungsregler benötigt, welche aus der pulsierenden Gleichspannung eine stabile, konstante, genau definierte Spannung „herstellen“.

Die Spannungsregler:

Spannungsregler werden überall in elektronischen Geräten eingesetzt, es sind reine Standardbauteile.

Ihr Zweck ist es, eine pulsierende, ungleichmässige Gleichspannung, wie sie von einem Gleichrichter geliefert wird, in eine, auch unter Belastung konstante Gleichspannung zu verwandeln. Diese Regler arbeiten sehr genau und ihre geregelten Ausgangsspannungen weichen auch bei Belastung und Erhitzung nicht vom Sollwert ab. Deshalb habe ich sie auch im Roboter verwendet, denn der Mikrocontroller braucht eine konstante und „gute“ Gleichspannung.
Es gibt verschiedene Typen von Spannungsreglern. In meinem Roboter wurden drei davon eingesetzt:

<u>Einstellbare Linearregler:</u>
Abb. 7 zeigt das Schaltplan-Symbol eines solchen Reglers und die Standardbeschaltung, die es braucht, damit der Regler selbst korrekt arbeiten kann.

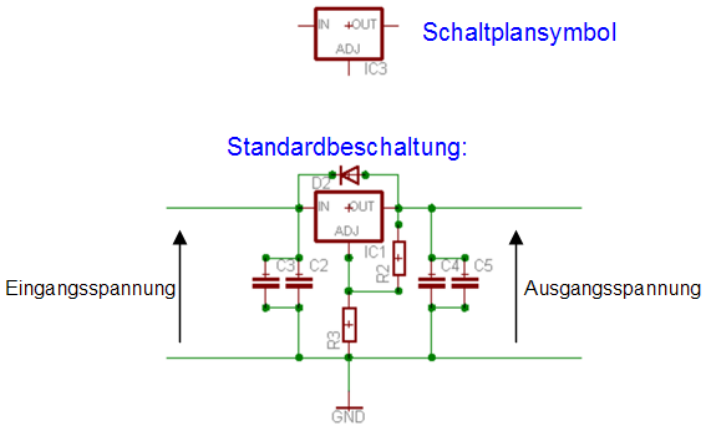


Abb.7: einstellbare Linearregler

Diese Regler verwandeln die „unsaubere“ Eingangsspannung in eine einstellbare, geregelte Ausgangsspannung. Dabei muss die Eingangsspannung höher als die Ausgangsspannung sein.
Wie man in Abb. 5 sieht, sind im Roboter 3 solcher einstellbarer Linearregler-Schaltungen verbaut. (Regler 1-3). Sie stellen alle eine Spannung von 5.5V für den Antrieb des Greifarmes bereit. Ich musste einstellbare Regler verwenden, da es keine Regler gibt, die, fest eingestellt, 5.5V ausgeben würden. Also habe ich einstellbare genommen, und die Ausgangsspannung auf 5.5V eingestellt. Diese Regler können einen Strom von 1.5A liefern, ohne zerstört zu werden. Dies reicht gut aus, der Antrieb des Greifarmes braucht maximal 1A.
Der Antrieb des Greifarmes benötigt zwischen 4V und 6V. Die Regler sind auf 5.5V eingestellt, so bleibt noch etwas Sicherheitsreserve.

Im unteren Teil der Abb.7 erkennt man die Beschaltung eines solchen Reglers. Mit den beiden Widerständen kann die Ausgangsspannung eingestellt werden.
Die Kondensatoren haben verschiedene Funktionen. Auf der Eingangsseite glätten sie die ungleichmässige Gleichspannung des Gleichrichters, dies ist nötigt, damit der Regler sauber arbeiten kann. Es wurden zwei Kondensatoren parallel geschaltet, einer mit grosser und einer mit kleiner Kapazität. Den kleinen Kondensator braucht es, um schnelle Spannungsspitzen abzufangen, die den Regler aus der Bahn werfen könnten, denn die grossen Kondensatoren sind dafür zu träge.
Die Kondensatoren auf der Ausgangsseite sorgen dafür, dass sich der interne Regelkreis des Reglers nicht aufschwingt, falls er dies tun würde, würde er keine saubere Spannung am Ausgang liefern.
Die Diode verhindert eine Beschädigung des Reglers, wenn die Spannungsversorgung abgeschaltet wird.
Im Roboter braucht es drei solcher Spannungsregler-Schaltkreise, da der Greifarm 3 Antriebe hat.
Die Abb.8 zeigt ein Bild eines solchen Reglers. Es zeigt aber nur den Regler alleine, ohne externe Beschaltung.

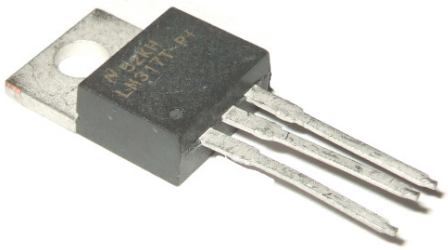


Abb.8: einstellbarer Linearregler

Abb.9 zeigt die Position der drei einstellbaren Linearregler auf dem Mainboard, wie auch die Lage des Gleichrichters und der anderen Regler. (Regler 5 und 6 werden vom Lüfter verdeckt)

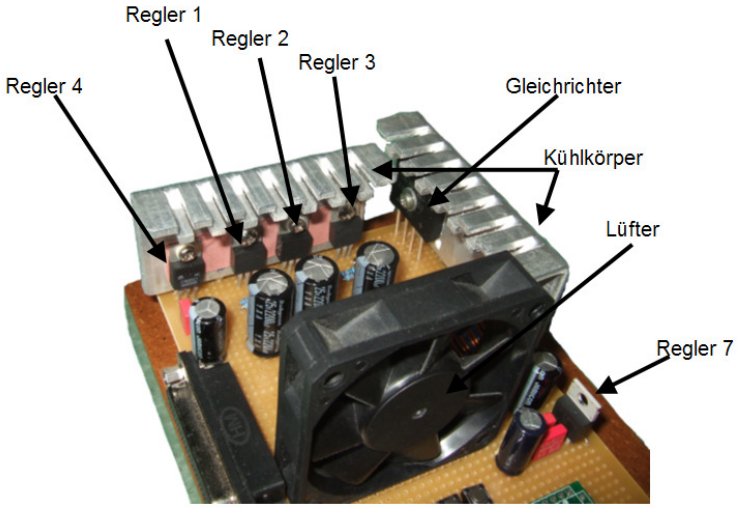


Abb.9: Lage der Regler

In Abb.9 sieht man vor den Reglern die externe Beschaltung (die schwarz/grauen Zylinder sind z.B. die grossen Kondensatoren).

Man sieht auch, dass die Regler durch Löcher in der Platine gesteckt werden. Auf der Unterseite werden sie mit den anderen Bauteilen verlötet.

Diese Linearregler haben aber einen Nachteil. Sie erzeugen die Ausgangsspannung, indem sie die überflüssige Spannung in Wärme verwandeln. (Die Eingangsspannung muss höher sein als die Ausgangsspannung, diese Spannungsdifferenz wird in Wärme verwandelt) Diese Leistung, die in Wärme verwandelt wird, nennt sich Verlustleistung. Sie wird wie folgt berechnet: (Eingangsspannung-Ausgangsspannung) • fliessender Strom.
Ohne Kühlkörper kann so ein Regler ca. 2W Verlustleistung abgeben, ohne thermisch zerstört zu werden. Im Roboter treten aber höhere Verlustleistungen auf: Die Spannungsdifferenz beträgt (17-5.5) 11.5V. Dabei fliessen bei Belastung des Greifarmes pro Regler ca. 800mA. Also: 11.5•0.8 = 9.2W Verlustleistung, die in Wärme verwandelt werden. Jetzt ist auch klar, wieso die Regler an einem Kühlkörper befestigt sind, und ein Lüfter für ausreichend Luftzirkulation sorgen muss. So erwärmt sich der Kühlkörper im Betrieb auf ca. 30°C. Diese 3 Regler produzieren die meiste Verlustleistung, da sie am meisten Strom liefern müssen.
Neben den einstellbaren Linearreglern gibt es aber noch fest eingestellte Linearregler:

<u>Nicht einstellbare Linearregler:</u>
Diese Regler funktionieren fast genau gleich wie die einstellbaren Linearregler. Nur braucht es hier keine externen Widerstände, um die Spannung einzustellen, denn diese Regler geben eine fest eingestellte Spannung aus.
So haben Regler 4 und Regler 8 eine fest eingestellte Ausgangsspannung von 5V, Regler 7 gibt 9V aus und Regler 6 erzeugt 12V, mit denen der Lüfter betrieben wird. Dabei übernimmt Regler 4 die Stromversorgung für das LCD.
Abb.10 zeigt die Standarbeschaltung dieser Regler – sie ähnelt stark der von den einstellbaren Linearreglern.

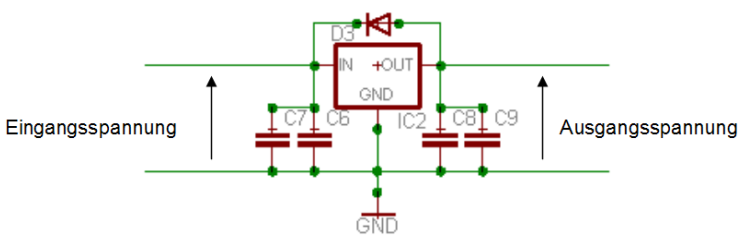


Abb.10: fest eingestellte Linearregler

Die Kondensatoren und die Diode haben auch hier dieselbe Bedeutung wie bei den einstellbaren Reglern. Hier ist aber nur Regler 4 auf einem Kühlkörper montiert, da er sich leicht

erwärmt.

Wie man im Schaltplan (Abb.5) erkennen kann, ist der Eingang von Regler 8 direkt mit dem Ausgang von Regler 7 verbunden.
Denn Regler 8 kann nur eine sehr kleine Verlustleistung vertragen (ca. 500mW). Dieser Regler versorgt den Controller mit einer Spannung von 5V und ist fest auf dem Adapterboard des Controllers verlötet, weshalb ich ihn nicht gegen einen Regler tauschen konnte, der eine höhere Verlustleistung vertragen würde.
Der Controller braucht etwa 50mA Strom, also kann man auch hier wieder die Verlustleistung an Regler 8 berechnen: (17-5)•0.05 = 600mW. Falls man Regler 8 also direkt an den Gleichrichter anschliessen würde, würde er thermisch zerstört werden, da die Verlustleistung zu gross wäre. Also nutzt man einen Trick, indem man die Verlustleistung auf einen zweiten Regler verteilt. Regler 7 regelt die 17V nämlich auf 9V herunter. An diesen 9V ist nun Regler 8 angeschlossen. Mit diesem Trick hält sich die Verlustleistung in Grenzen.

Im Roboter wurde aber noch ein dritter Spannungsreglertyp verbaut:

Schaltregler:

Zur Spannungsversorgung des Elektromagneten wurde ein Schaltregler verwendet. (Regler 5, s.Abb.5) Dieser Regler funktioniert anders als ein Linearregler, was man auch schon an seiner externen Beschaltung erkennen kann, die viel mehr Bauteile enthält. Er hat den Vorteil, dass er sich kaum erhitzt, denn er produziert die Ausgangsspannung nicht, indem er die überflüssige Spannung in Hitze verwandelt. Ansonsten hat er die gleiche Aufgabe; die unsaubere Spannung vom Gleichrichter kommt an seinen Eingang, und am Ausgang liefert der Regler eine saubere Gleichspannung. Abb.11 zeigt die Standardbeschaltung solcher Regler.

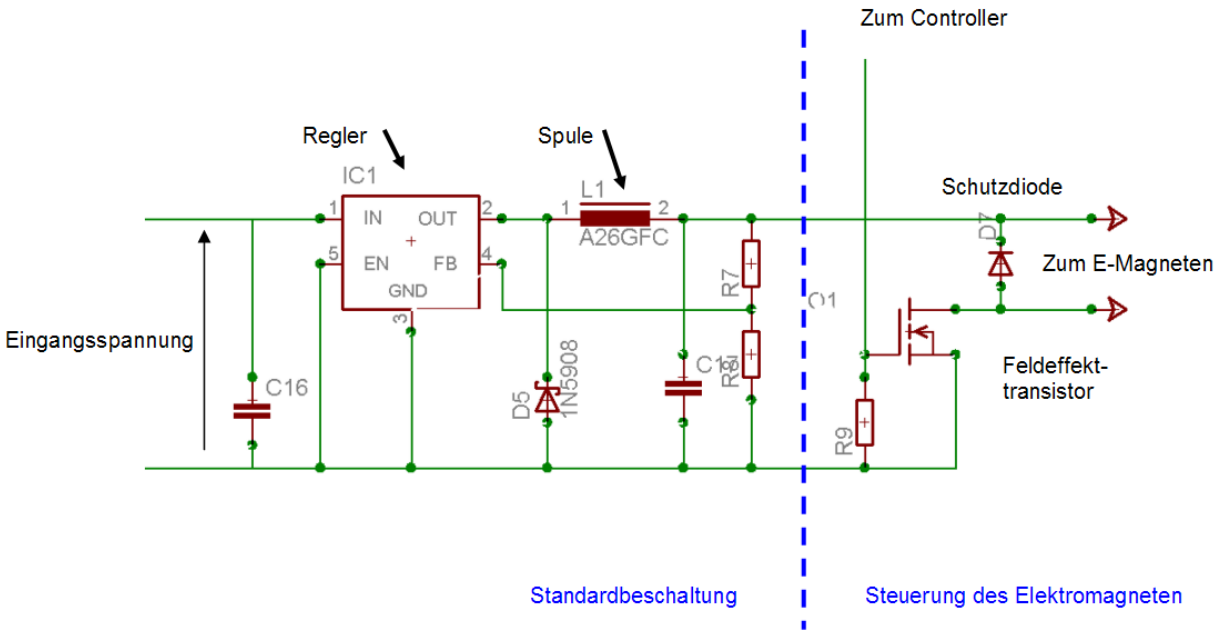


Abb.11: Schaltplan des Schaltreglers

Mit den beiden Widerständen in der Standardbeschaltung lässt sich die Ausgangsspannung einstellen, sie liegt bei etwa 5.7V.
Für den Elektromagneten hab ich diese Art von Versorgung gewählt, da sich ein Linearregler viel zu stark erhitzt hätte, denn es fliessen Ströme von über 1A, was eine zu hohe Verlustleistung bei einem Linearregler bedeutet hätte, er wäre trotz Kühlung zerstört worden.
Dieser Schaltregler arbeitet, wie der Name schon sagt, indem er eine Spule mit Energie auflädt. Ist sie aufgeladen, wird der Strom für die Spule abgeschaltet, und die Energie der Spule entlädt sich teilweise in einen Kondensator, welcher so eine Spannung (=Ausgangsspannung) aufbaut. Danach wird die Spule wieder mit Strom beschickt, sie lädt sich wieder auf, und der Prozess beginnt von neuem. Die Ausgangsspannung hängt davon ab, wie stark sich die Spule in den Kondensator entlädt. Dies hängt wiederum davon ab, wie lange die Pause ist, in der die Spule keinen Strom geliefert bekommt. Diese Pause wird vom Regler genau so eingestellt, dass die korrekte Ausgangsspannung entsteht. Dieser Regler kann so Ströme bis 3A liefern, das ist doppelt soviel wie ein Linearregler. Dabei erwärmt er sich aber kaum, was den Vorteil ausmacht.
Im Alltag findet man solche Schaltregler auch in fast allen elektronischen Geräten, ein PC-Netzteil arbeitet z.B. nach dem gleichen Prinzip.

Der Strom darf aber nicht immer in den Elektromagneten fliessen, denn man möchte die Figuren ja auch absetzen können. Also muss man den Strom steuern können. Dies erreicht man, indem man einen Schalter einbaut. Dieser Schalter besteht aus

einem Feldeffekttransistor. (FET) Die Steuerleitung des FET ist mit dem Controller verbunden. Lässt der Controller an dieser Leitung 5V anliegen, so leitet der FET – der Schalter ist geschlossen und der Elektromagnet zieht an. Legt der Controller diese Leitung aber auf GND/0V/Masse, so sperrt der FET – der Schalter ist offen und es fliesst kein Strom. Die Schutzdiode fängt die Induktionsspannung ab, die beim Abschalten des Stromes durch den Elektromagneten entsteht, welche den FET zerstören könnte. Abb.12 zeigt die Lage des Schaltreglers auf dem Mainboard.

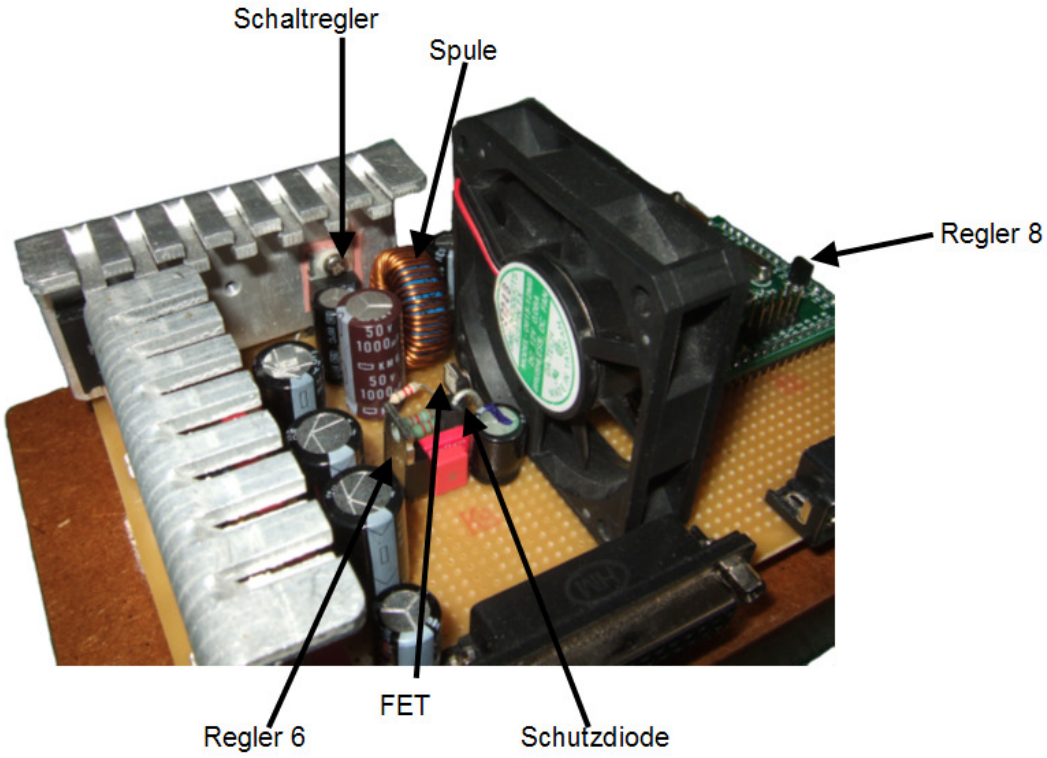


Abb.12: Sicht auf den Schaltregler

3.5 Das Greifarm-Modul

Auf dem Greifarm-Modul befinden sich der Greifarm und das Gehäuse, in dem sich der Transformator befindet. Auf der Grundplatte (Holz) des Moduls befindet sich ein Steckkontakt, der mit dem Mainboard-Modul verbunden wird. Der Greifarm hat den Zweck, die Figuren des Roboters auf dem Spielfeld zu bewegen.

Am Transformator-Gehäuse, welches aus geklebten Plexiglasplatten besteht, befindet sich der Hauptschalter für den Roboter und der Netzstecker, womit der Transformator mit dem Stromnetz verbunden wird. Der Hauptschalter verbindet dann

intern den Transformator mit dem Stromnetz, er schaltet direkt die 230V, wofür er auch ausgelegt ist.
Vor dem Transformatorgehäuse befindet sich der Greifarm. Er ist mittels eines Metallbaukastens („Mecano“) aufgebaut. Es bot sich diese Art von Aufbau an, da solche Metallteile flexibel einsetzbar sind. Der Greifarm wird von so genannten Servos angetrieben. Diese Servos werden normalerweise im Flugmodellbau verwendet, um die Ruder der Modellflugzeuge im Flug anzusteuern.
Doch sie lassen sich prima verwenden, um einen Greifarm anzusteuern, denn sie sind relativ leicht und doch sehr stark. So ein Servo besteht aus einem Elektromotor, einem Getriebe und einer Ansteuerungselektronik, alles in einem schwarzen Gehäuse eingebaut. Der Elektromotor bewegt über das Getriebe einen Dreharm. Dieser Dreharm führt so die Bewegung aus.
Abb. 13 zeigt ein Servo, am Greifarm montiert.
Die Servos am Roboter können in 1cm Abstand zur Drehachse eine Kraft von 110Nm ausüben, das bedeutet, sie können ca. 11kg in einem Abstand von 1cm zur Drehachse heben. Im Betrieb haben alle Servos eine Stromaufnahme von ca. 800mA. Sie benötigen dabei eine Spannung zwischen 4V und 6V, die Servos am Roboter werden mit 5.5V betrieben.

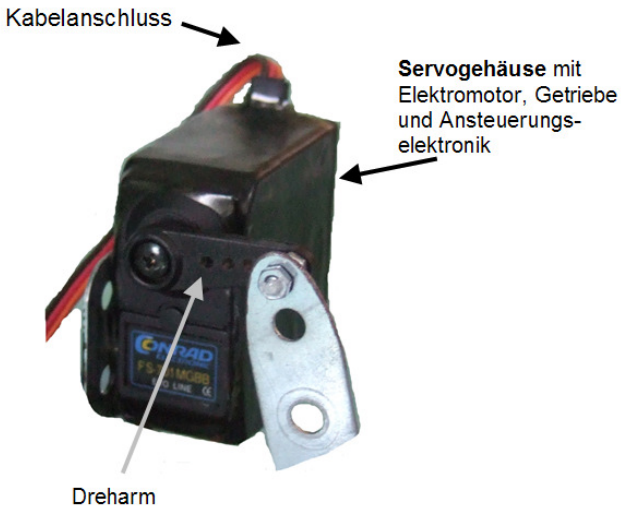
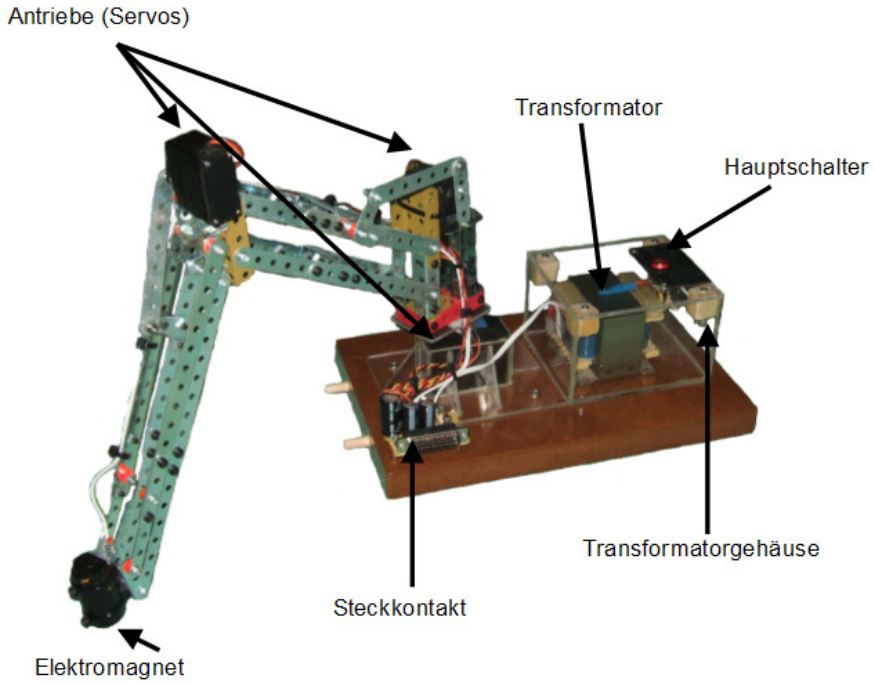


Abb.13: Das Servo

Angesteuert werden diese Servos über nur 3 Leitungen. Auf zwei Leitungen wird der Strom übertragen, auf der dritten das Signal für die Position des Dreharmes. Dieses Signal wird vom Controller erzeugt, die interne Servoelektronik wertet es dann aus und fährt den Dreharm in die entsprechende Stellung.
Die Servos wurden mit Epoxidkleber an den Greifarm geklebt. Das unterste Servo ist das grösste, denn es trägt den gesamten Greifarm. Es ist mittels einer Halterung aus Plexiglas befestigt. Auf seinem Dreharm steht der gesamte Aufbau des Greifarmes. An der Spitze des Armes befindet sich ein selbstgewickelter Elektromagnet. Er hebt die Figuren und platziert sie so auf dem Spielfeld. Er wird mit ca. 5.7V angesteuert. Dabei fliessen Ströme von ca. 1.2A. Abb.14 zeigt das beschriftete Greifarm-Modul.



Das unterste Servo dreht den gesamten Greifarm, welcher auf seiner Drehachse ruht. Die zwei anderen Servos heben und senken den Arm.

Abb.14: das Greifarm-Modul

Die Servos haben eine sehr „ruckartige“ Stromaufnahme, das heisst, sie nehmen den Strom nicht gleichmässig auf, sondern in kurzen Impulsen. Bei Testläufen hat sich gezeigt, dass durch diese Impulse alle Spannungen im Roboter kurzzeitig abgesenkt werden(sie brechen zusammen), sodass der Mikrocontroller abgestürzt ist.
Dies wird verhindert, indem sich auf dem Steckkontakt 3 grosse Kondensatoren befinden, welche genau diese Stromimpulse sehr gut dämpfen können. So bleibt die Spannung erhalten, da die Kondensatoren diese impulsförmigen Ströme kurzzeitig liefern können. Zudem befindet sich auf dem Steckkontakt noch die Sicherung, welche sich in der 17V-Leitung befindet. Auf diesem Steckkontakt werden hauptsächlich Ströme durchgeleitet, zum Einen fliesst der gesamte Versorgungsstrom des Roboters vom Trafo zur Spannungsversorgung im Mainboard, zum Anderen werden die Ströme für die 3 Servos und den Elektromagneten auch durch den Steckkontakt übertragen. Abb.15 zeigt den Steckkontakt auf dem Greifarm-Modul.

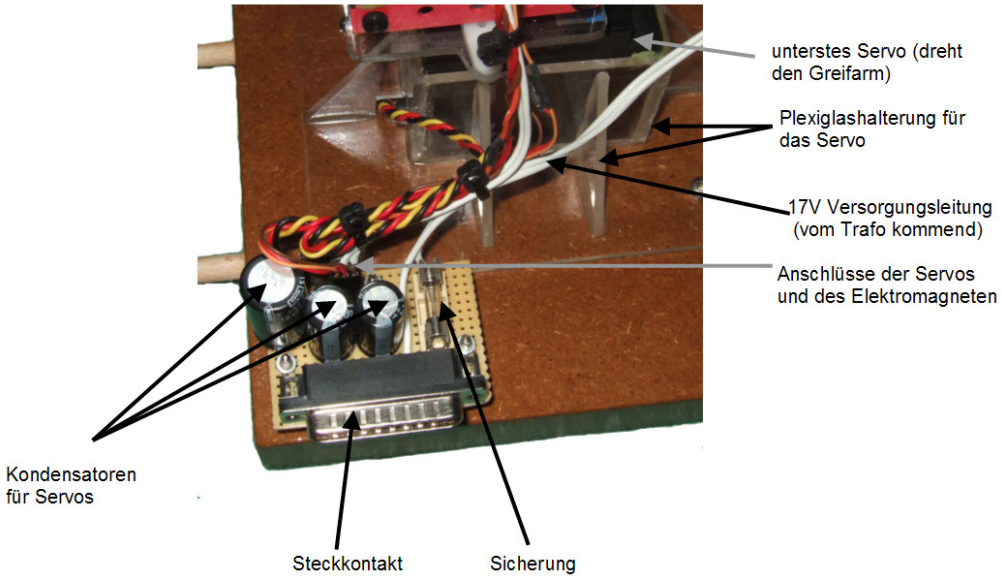


Abb.15: Steckkontakt

Da der ganze Greifarm auf einer dünnen Servoachse gelagert ist, schwankt er. Um Fehlpositionierungen zu vermeiden, wartet die Software deshalb eine Sekunde, bevor sie den Arm eine Figur aufnehmen oder absetzen lässt, damit der Arm nicht mehr zu stark schwankt.

3.6 Das Spielfeld-Modul

Dieses Modul ist das grösste und schwerste des Roboters, denn es trägt das massive Spielfeld.

An der Vorderseite des Spielfeldes befindet sich ein LCD, hinter einer Edelstahlplatte verbaut. Dieses LCD erlaubt dem Roboter, mit dem Benutzer zu kommunizieren. Zusätzlich befindet sich ein Taster auf dieser Edelstahlplatte, damit kann der Benutzer mit dem Roboter kommunizieren.

Das LCD wird direkt von Controller angesteuert. Dafür werden 4 Pins am Controller benötigt.

Das Spielfeld-Modul besitzt auch einen eigenen Steckkontakt, der an das Mainboard-Modul angesteckt wird.

Das Gesamte Modul besteht auch hier wieder aus einer hölzernen Grundplatte (Pressspan) und dem Spielfeld, welches aus demselben Material aufgebaut ist.

<u>Das Spielfeld:</u>

Damit man das Spielfeld nicht auf das Holz zeichnen musste, was nicht sonderlich anschaulich gewesen wäre, wurde es aus dünnem Furnierholz zurechtgeschnitten und aufgeklebt. Auf das schwarze und weisse Furnier wurden die jeweiligen Schablonen angelegt und dann mit einem Cutter ausgeschnitten. So ist die Spieloberfläche stabil und wie bei echten Spielfeldern aufgebaut.

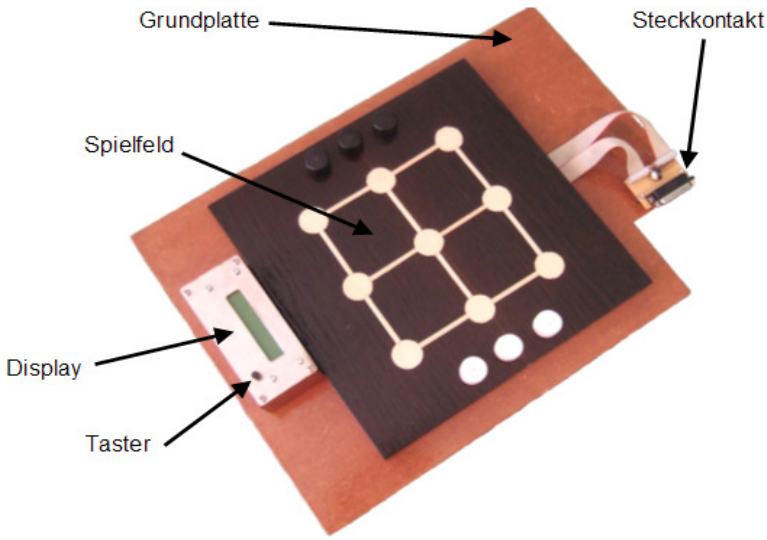


Abb.16: Das Spielfeld-Modul

Auf dem Spielfeld werden die Figuren positioniert, doch der Roboter muss die Position der Figuren erkennen können. Es reicht aus, wenn er weiss, auf welchem Feld eine Figur steht. Die Software ordnet dann die Figur dem jeweiligen Spieler zu. Damit der Roboter die Figur erkennen kann, wurden Sensoren unter dem Spielfeld angeordnet, insgesamt 9, denn es gibt ja 9 mögliche Felder, die angespielt werden können. Als Sensoren wurden so genannte Reed-Sensoren verwendet. Diese wirken als Schalter und schliessen einen Kontakt, sobald ein magnetisches Feld um den Sensor herum stark genug wird. Folglich befindet sich in allen 6 Spielfiguren ein kleiner Magnet, der ebendiese Sensoren ansprechen lässt.
Dieser Magnet hat zwei Funktionen: zum einen steuert er die Reed-Kontakte, zum anderen dient er auch als Angriffspunkt für den Elektromagneten, welcher seine Anziehungskraft am kleinen Magneten im Spielstein ausüben kann und so die Steine anhebt.
Der Vorteil der Reed-Kontakte ist, dass sie sehr klein sind, die im Spielfeld verbauten sind ca. 1cm lang und haben 2mm im Durchmesser. Zudem sind sie sehr einfach auszuwerten, dazu später mehr.
Die Verkabelung der Reed-Sensoren und des LCD verläuft vollständig unter dem Spielfeld. Dazu wurden Kanäle in die Unterseite des Spielfeldes eingefräst, in denen dann die Kabel verlaufen. Am Hinteren Teil des Spielfeldes kommen dann nur zwei Flachbandkabel zum Vorschein, die in den Steckkontakt münden, was das Ganze aufgeräumt wirken lässt.
Damit die Reed-Sensoren möglichst nah am Furnier, also an der Spieloberfläche sind, wurden von unten Löcher in das Spielfeld gebohrt, in die dann die Sensoren verklebt wurden. Leider stimmen bei gewissen Feldern diese Löcher nicht genau mit dem weissen Furnier überein, sprich: die Sensoren befinden sich nicht genau unter den weissen Furnier-Spielfeldern, die Spielfiguren müssen da leicht versetzt abgesetzt werden, damit der Sensor anspricht. Da der Spielstein aber magnetisch ist und vom Reed-Sensor leicht angezogen wird, spürt man die korrekte Position beim Absetzen, ansonsten wird man von der Software über das Display auf den Fehler hingewiesen.

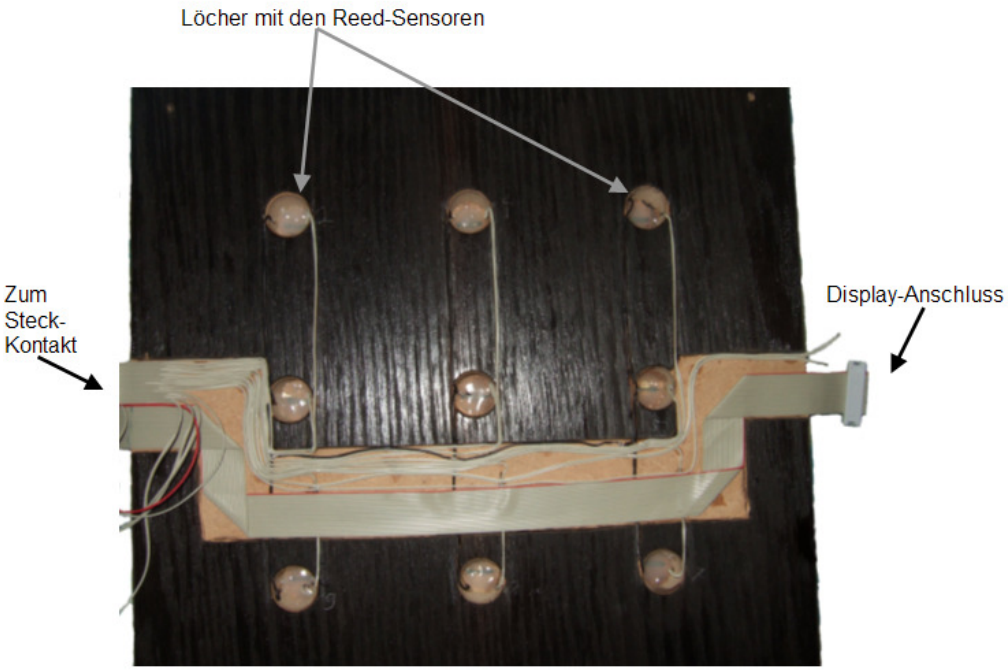
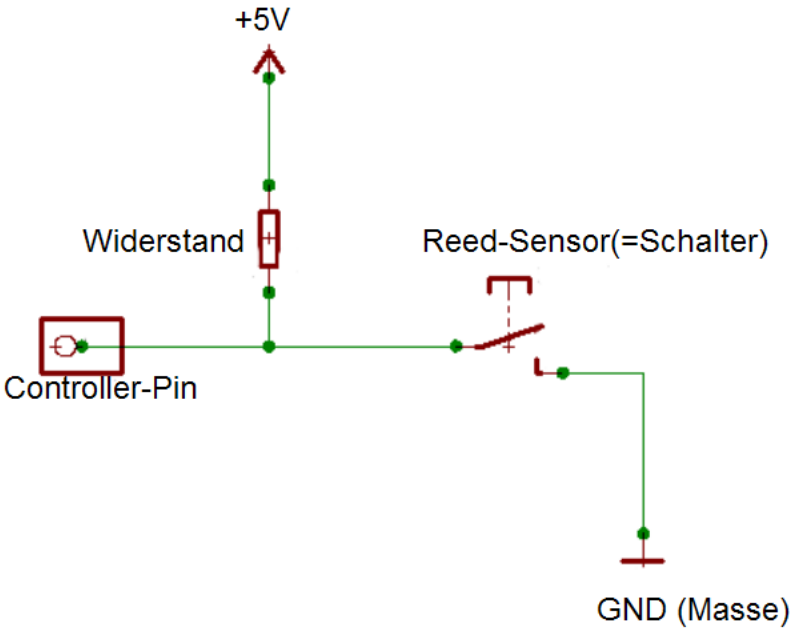


Abb.17: fast vollständig verkabeltes Spielfeld (Unterseite)

Die Reed-Sensoren sind leicht auszuwerten, da sie nur ein Schalter sind. Dieser Schalter schliesst lediglich eine elektrische Verbindung. Wird ein Spielstein abgesetzt, dann schliesst der Reed-Sensor seinen internen Schalter.
Die Pins am Controller können Spannungszustände einlesen. Sie können unterscheiden, ob an einem Pin 5V gegen Masse anliegen, oder ob der Pin auf Masse liegt. Die Pins werden in der Software für diese Funktion konfiguriert.
Und genau diese zwei Zustände können vom Reed-Kontakt erzeugt werden.
Abb.18 zeigt schematisch das Funktionsprinzip.



Ist der Reed-Kontakt geöffnet (kein Stein auf dem Spielfeld), dann ist der Pin über den Widerstand (10k Ohm) mit +5V verbunden, somit liegen am Pin 5V an. Dieser Widerstand heisst „Pullup-Widerstand“, da er den Pin auf die 5V „hochzieht“.
Dies kann der Controller erkennen. Für ihn heisst das, dass sich kein Stein auf dem Feld befindet. Liegt hingegen ein

Abb.18: Funktionsprinzip der Steine-Erkennung

Stein auf dem Feld, dann ist der Schalter des Reed-Sensors geschlossen, und der Pin wird direkt mit Masse verbunden. Da der Widerstand(Pullup) sehr gross ist, und der Pin direkt an Masse liegt, ist der Einfluss der Masse viel stärker, daher liegen nahezu 0V (=Masse) am Pin an. Dies kann der Controller registrieren und weiss daher, dass sich ein Stein auf dem Feld befindet.
Für das Einlesen der Spielfelder werden also 9 Controller-Pins benötigt, da es 9 Felder gibt.

Abb.19 zeigt den Steckkontakt des Spielfeldes. Man erkennt die 9 Pullup-Widerstände für die einzelnen Spielfelder. Die elektrischen Leitungen werden über zwei Flachbandkabel bewerkstelligt, eines für das LCD und das andere für die 9 Spielfelder.
Das Potentiometer ist ein einstellbarer Widerstand, mit ihm lässt sich der Kontrast des LCD einstellen.

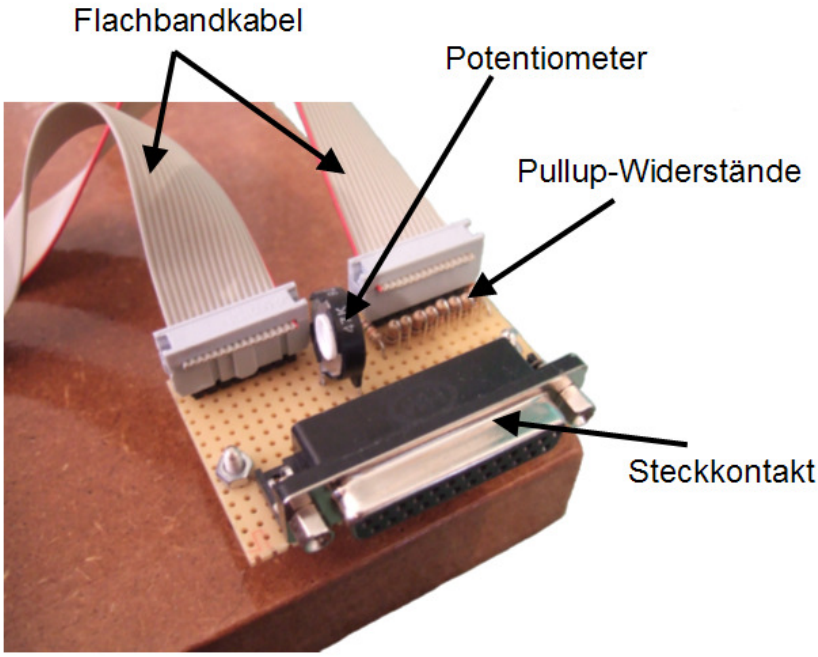


Abb.19: Steckkontakt

Es ist noch anzumerken, dass die Reed-Sensoren die verschiedenen Spielfiguren nicht unterscheiden können. Die Zuordnung der verschiedenen Figuren zu den Spielern erledigt die Software.

4. Funktionsweise der Software

4.1 Mathematischer Hintergrund (Spieltheorie)

Das Ziel dieser Arbeit ist es ja, dass der Roboter „einfache Mühle“ gegen einen humanen Gegner spielen kann, indem er mit regelkonformen Zügen auf die des Menschen reagiert. Doch woher weiss der Roboter, was ein regelkonformer Zug ist? Woher soll er wissen, was er machen soll?
Um diese Probleme zu lösen gibt es eine mathematische Theorie, die sich mit genau diesen Themen beschäftigt. Diese Theorie wurde auch beim Roboter angewandt, indem sie in das Steuerungsprogramm, welches im Mikrocontroller sitzt, eingeflochten wurde. So kann der Roboter, wie ein realer Mensch, das Spiel spielen.

Diese Spieltheorie funktioniert aber nur bei Spielen, in denen beide Gegner die vollständige Information über den Spielzustand haben, warum werden wir später sehen. Solche Spiele nennt man „Spiele mit perfekter Information“. Beispiele sind Schach, Mühle, TicTacToe, oder eben „einfache Mühle“.

Der mathematische Hintergrund soll nun am Beispiel von „einfache Mühle“ erklärt werden:
Bei diesem Spiel ziehen die Spieler abwechselnd. Nach jedem Zug herrscht ein anderer Zustand auf dem Spielfeld. Also kann man einen Ablauf aufzeigen, der alle möglichen Züge und Zustände beinhaltet. Ein solcher Ablauf wird „Spielbaum“ genannt. Abb.20 zeigt einen solchen Spielbaum für die ersten zwei Züge von „einfache Mühle.“

Zuoberst ist die so genannte „Wurzel“ des Spielbaumes. Sie zeigt den Ausgangszustand auf – in diesem Falle das leere Spielfeld, denn so wird das Spiel ja gestartet. In diesem Beispiel gibt es zwei Spieler, Spieler „Grau“ und Spieler „Blau“. Grau beginnt zu spielen – er muss eine Figur setzen. Da das Spielfeld 9 Felder hat, kann „Grau“ seine Figur beliebig auf diese setzen – er hat 9 Zugmöglichkeiten. Hat er seinen Stein gesetzt, ist ein neuer Spielzustand erreicht. Von diesen Zuständen gibt es bei diesem ersten Zug 9 Stück, denn „Grau“ hat ja 9 Felder zur Auswahl. Diese Zustände werden im Spielbaum unter „Ebene 1“ aufgelistet. In Abb.20 sieht man,

grafisch dargestellt, die 9 Spielzustände, die durch den ersten Zug von „Grau“ erreicht werden können.

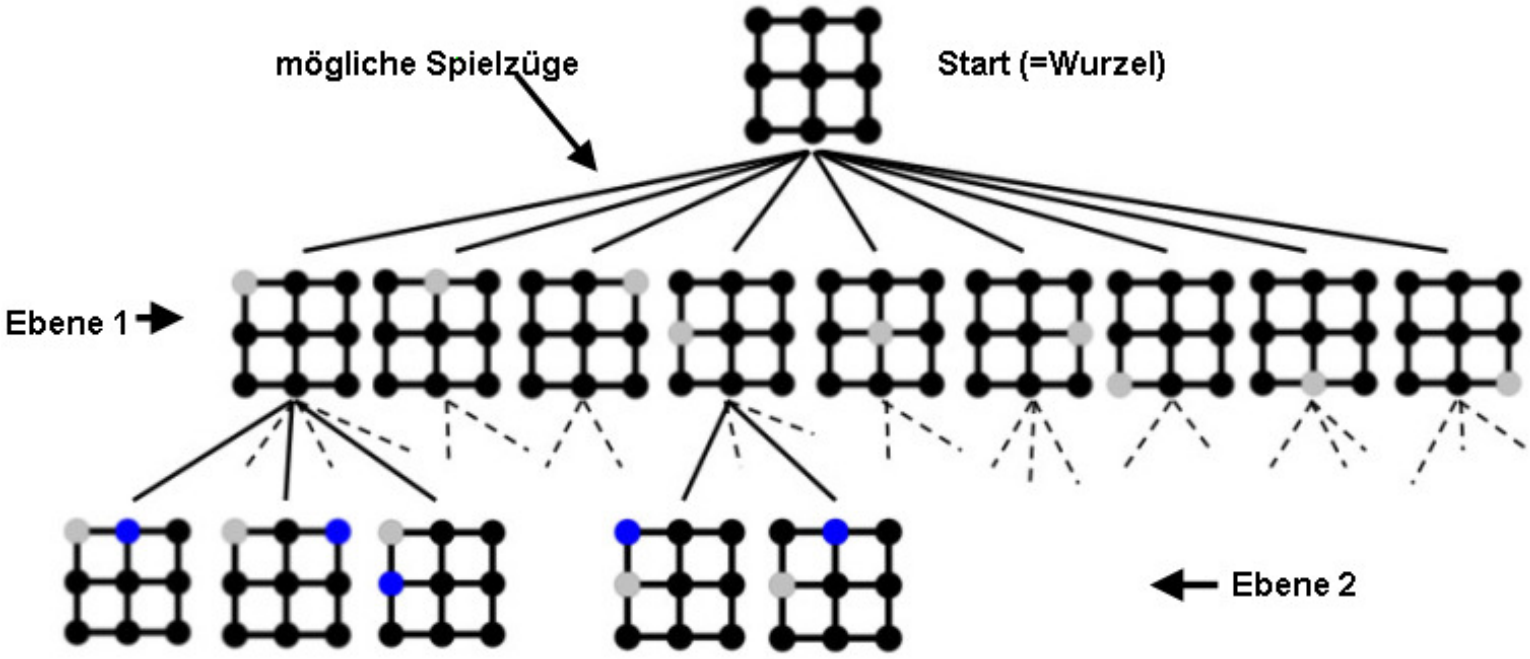
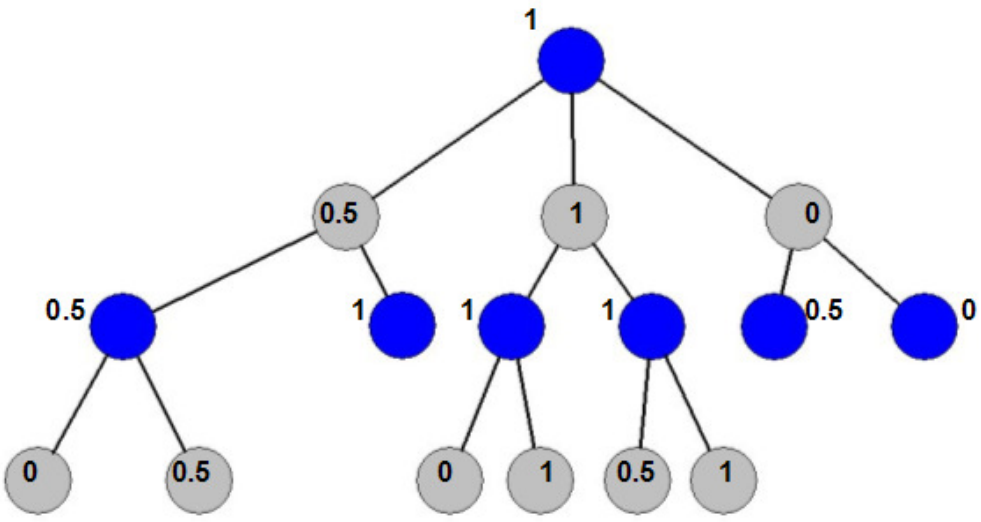


Abb.20: Spielbaum für die ersten zwei Züge bei „einfache Mühle“

Hat „Grau“ seinen Zug beendet, ist es an „Blau“, den nächsten Zug zu machen. Jetzt hat „Blau“ nur noch 8 Möglichkeiten, seinen Stein abzusetzen, denn ein Feld ist ja vom Stein von „Grau“ besetzt. In der „Ebene 2“ sind nun alle Spielzustände aufgelistet, die durch einen Zug von „Blau“ erreicht werden können. Für jeden Zustand aus der Ebene 1 kommen in der Ebene 2 8 weitere dazu. (die gestrichelten Linien sollen dies verdeutlichen, in der zweiten Ebene wurden nicht alle Zustände aufgezeichnet, da Platzmangel).
Ein Spielbaum dient also dazu, alle möglichen Spielzustände aufzulisten. Will man nur die Zustände aufzeigen, die durch die ersten beiden Züge erreicht werden können, dann muss man schon 72 Felder aufzeichnen, denn 9•8=72.
Ein Spielzustand im Spielbaum wird auch „Knoten“ genannt.
Es ist klar, dass je mehr Züge (=Ebenen) man betrachten möchte, umso breiter wird der Spielbaum. Schon bei einem einfachen Spiel wie „einfache Mühle“ wird der Baum sehr schnell sehr gross. Man stelle sich vor, wie komplex so ein Spielbaum

beim Schachspiel wird, dann wird einem auch klar, wieso Schachcomputer der ersten Generationen so enorme Rechenleistungen benötigten, nur schon, um den Spielbaum zu berechnen.

Doch dieser Spielbaum ist nur ein Hilfsmittel für den Rechner, um seinen Zug zu erzeugen. Der Spielbaum zeigt lediglich alle zukünftigen Spielzustände auf. Der Rechner kann diesen Spielbaum zwar berechnen, doch nur mit ihm kann er noch nichts anstellen. Es braucht noch eine Funktion, die ihm erlaubt, aus diesem Spielbaum den für ihn besten Zug herauszusuchen, denn der Rechner/Roboter möchte ja gegen den Menschen gewinnen.
Dazu gibt es den so genannten Minimax-Algorithums.
Dazu die Bedingungen: Es gibt zwei Spieler: Min und Max. Max ist ein guter Spieler, wenn er am Zug ist, spielt er den bestmöglichsten Zug.
Min ist sein Gegner. Auch er wird optimal spielen. Für Max bedeutet das aber, dass bei jedem Zug von Min, seine eigene Spielsituation verschlechtert, minimiert wird, sprich: Min spielt den für Max am ungünstigsten Zug. Dies ist auch im realen Spiel der Fall. Und genau mit diesem Sachverhalt kann der Roboter den für sich am vorteilhaftesten Zug errechnen.
Dies soll an folgendem Spielbaum erörtert werden:



Es gibt die Spieler „Blau“ und „Grau“. „Blau“ fungiert als „Max“, er wählt immer den besten Zug aus. „Grau“ fungiert als „Min“, er wählt den Zug aus, der für Blau am schlechtesten ist. Die Zahlen in den Feldern kennzeichnen

Abb.21: Minimax-Spielbaum

verschiedene Spielsituationen: Eine 1 bedeutet Sieg für Blau, eine 0 bedeutet Sieg für Grau, und eine 0.5 steht für unentschieden.

Dieser Spielbaum ist aus der Sicht von „Blau“ gezeichnet, denn er ist am Zug, was man an der blauen Wurzel erkennen kann. Der Baum ist 3 Ebenen tief.
Der Baum wird von unten her gegen die Wurzel hin entschlüsselt:
Unten links gibt es zwei mögliche Zustände: 0 und 0.5. Da Blau (=Max) diese Zustände erreichen kann (der Zustand darüber ist Blau), wählt er den besseren, also 0.5. Darum erhält der Spielzustand, der darüber steht, den Wert 0.5. Gehen wir nun eine Stufe höher. Da ist Grau am Zug. Er hat nun die Wahl zwischen 0.5 und 1. Da er der Min-Spieler ist, wird er sich für 0.5 entscheiden, denn ansonsten würde Blau gewinnen. Darum erhält dieser Knoten den Wert 0.5.
Genau so verfährt man mit den zwei anderen Ästen des Spielbaumes. Zum Schluss hat man den Wert für die Wurzel bestimmt ➔ er ist 1. Dies bedeutet, dass Blau garantiert gewinnen wird, wenn er den Zug wählt, der zum mittleren Ast des Baumes führt.
Spielbäume werden also von unten her gegen die Wurzel hin aufgeschlüsselt. Danach weiss man bei der Wurzel, welches der optimalste Zug ist, unter der Berücksichtigung, dass Min immer den Zug spielen wird, der am schlechtesten für Max ist. So kann der Roboter mithilfe des Spielbaumes denjenigen Zug finden, der in der jeweiligen Spielsituation am passendsten ist.

Die Suchtiefe bestimmt, wie viele Züge im Voraus der Baum gezeichnet wird, sprich, wie viele Züge im Baum dargestellt werden. Jeder Zug ist eine neue Ebene im Baum. (s.Abb.20). Vergrössert man die Suchtiefe, dann wächst der Baum sehr schnell und wird dementsprechend sehr gross. Optimal wäre es natürlich, wenn man zu Spielbeginn einen Baum aufstellen könnte, der bis ans Ende des Spieles reicht und man schon im Vorhinein wissen würde, welcher Zug zum Sieg führt. Dies ist aber aus praktischen Gründen (z.B. Speicherplatz / Rechenzeit) nicht möglich. Also begrenzt man die Suchtiefe auf ein paar wenige Ebenen.
Doch woher weiss man, welcher dann der beste Zug ist? Der Baum reicht ja nicht bis ans Ende des Spieles, sondern nur bis zu einer gewissen Spielsituation.

Dazu dient die Feldbewertungs-Funktion. Sie weist den untersten Spielzuständen im Baum (diejenigen an der Suchgrenze) numerische Werte zu, die davon abhängen, ob dieser Spielzustand für den Spieler gut oder schlecht ist. Mit diesen Werten kann

dann, wie im Beispiel der Abb.21, auf den Wert der Wurzel und somit auf den besten Zug geschlossen werden.
Je besser der Spielzustand für den Spieler, umso höhere Werte erhalten die Felder. Ist der Zustand jedoch schlecht, also gut für den Gegner, dann erhalten die Felder kleinere Werte.
Am Beispiel der einfachen Mühle untersucht diese Funktion, ob z.B. zwei Steine in einer Reihe liegen. Ist das der Fall, erhält das Feld eine bessere Bewertung, als wenn zwei feindliche Steine in einer Reihe liegen würden. Dazu später mehr.

Es gibt zwei Möglichkeiten, um einen Spielbaum aufzustellen: die Breiten- und die Tiefensuche.
Bei der Breitensuche werden alle möglichen Zustände erzeugt und abgespeichert. Es liegt also der vollständige Spielbaum (bis zu einer gewissen Suchtiefe) im Speicher des Rechners. Es ist klar, dass dieser Prozess viel Speicher benötigt.
Bei der Tiefensuche hingegen wird nie der gesamte Spielbaum abgespeichert, sondern nur Teile davon. Man sucht zuerst den erstbesten Spielzug von der Wurzel aus und berechnet von diesem Zug alle Nachfolgezustände bis zur Suchtiefe. Man speichert hier nur einen Ast des Spielbaumes, was natürlich wesentlich weniger Speicher benötigt.

Der Roboter erstellt also zuerst den Spielbaum, indem er entweder die Tiefen- oder Breitensuche verwendet. Dies tut er jedoch nur ein paar Züge tief. Dann bewertet er die Spielsituationen an der Suchgrenze und weist ihnen Werte zu, je nach Nützlichkeit dieser Spielsituationen. Mit diesen Werten kann er dann mit dem Minimax-Algorithmus den Wert der Wurzel und somit den für sich besten Zug herausfinden, den er dann auch spielt.
Hier erkennt man auch, warum dieses System nur bei Spielen mit perfekter Information funktioniert – die Positionen des Gegners müssen bekannt sein, um den Spielbaum erstellen zu können.

Es ist klar, dass Roboter, die mit solchen Algorithmen arbeiten, dem Menschen weit überlegen sind. Denn sie können eine enorme Menge von Spielzuständen betrachten, und das in einer sehr kurzen Zeit.

Das berühmteste Beispiel wird wohl der Sieg eines Schachcomputers über den damaligen Schachweltmeister Kasparow sein, der die Effizienz dieser Algorithmen, die auch bei einem enorm komplexen Spiel funktionieren, bewies.

4.2 Umsetzung der Spieltheorie im Programm

Der gesamte Roboter wird von einer Software gesteuert, welche im Mikrocontroller gespeichert ist und diesen agieren lassen kann. Die Software wurde in Basic geschrieben (eine Programmiersprache). Der gesamte Code des Roboters ist 1560 Zeilen lang und benötigt im Mikrocontroller 20.5kB Speicherplatz. (der Controller hätte für 128kB Code Platz, es herrscht also keine Speicherplatzknappheit.)
Dieses Programm steuert zum einen die Bewegungen des Greifarmes oder gibt Text auf dem LCD aus. Zum anderen liest sie aber auch den aktuellen Spielzustand vom Spielfeld ein und berechnet mithilfe der oben beschriebenen Spieltheorie den bestmöglichsten Zug, der dann auch ausgeführt wird.

Da „einfache Mühle“ zwei Spielphasen kennt (zuerst wird gesetzt, dann gezogen), muss auch die Software für beide Zustände separat geschrieben werden. Es gibt also für die beiden Spielphasen separate Algorithmen, die jedoch dieselbe Aufgabe haben.

Im Folgenden sollen verschiedene Algorithmen, die es für die Funktion des Roboters benötigt, genauer aufgezeigt werden. Dazu möchte ich deren Funktion an kommentierten Struktogrammen erläutern.
Der gesamte Code ist in „Modulbauform“ aufgeführt, sprich, verschiedene Module führen verschiedene Aufgaben aus. Diese Module sind in Unterprogrammen angeordnet, im Hauptprogramm gibt es also kleinere Programme, die „separat“ ausgeführt werden können.

<u>Der Einlese-Algorithmus:</u>

In diesem Unterprogramm liest der Controller die aktuelle Spielsituation ein, damit er weiss, wie die Figuren auf dem Spielfeld stehen. Dazu werden die Reed-Sensoren benutzt, denn mit ihnen weiss der Controller, ob eine Figur auf dem Feld steht oder nicht. Er muss dazu nicht wissen, von welchem Spieler die Figur ist, denn diese Information gibt der Reed-Sensor nicht, das ist Aufgabe der Software.

In diesem Unterprogramm müssen die beiden Spielphasen separat abgehandelt werden, es gibt zwei Algorithmen.

Die Spielfiguren der einzelnen Spieler werden durchnummeriert. Der Roboter hat die Figuren von 1-3, während der Mensch die Figuren von 4-6 hat. Ebenso sind die Spielfelder von 1-9 durchnummeriert. So kann eindeutig unterschieden werden. Der Spielfeld-Zustand wird in ein Array gespeichert. Ein Array kann man als Tabelle betrachten. Unter dem Wert des jeweiligen Spielfeldes (z.B. Feld 3) wird dann der Wert der draufstehenden Figur gespeichert (z.B. Figur 2). Steht keine Figur auf dem Feld, wird eine 0 zugewiesen. Im Code trägt dieses Array den Namen „Jetztfeld“, da es das momentane Feld beinhaltet. Von diesem Array wird dann der Spielbaum erzeugt.
Die beiden Spieler (Roboter / Mensch) werden unter „Spieler“ gespeichert. Ist diese Variable =1, dann ist der Roboter gemeint, ist sie =2, dann ist der Mensch gemeint.

Spielphase 1: Setzen:

<table>
<tr><th colspan="3">Struktogramm</th><th>Kommentar</th></tr>
<tr><td colspan="3">Ist Pina.3=0 und Jetztfeld(1)=0?
Ja Nein</td><td rowspan="3">Pina.3 ist der Anschluss des Reed-Sensors. Ist dieser Wert gleich Null, so steht eine Figur drauf. Ist gleichzeitig kein Platz im Array für dieses Feld (Hier: Feld 1) belegt, dann ist eine neue Figur auf das Feld hinzugekommen. Falls nicht: gehe zum nächsten Feld.</td></tr>
<tr><td>Spieler =1</td><td>Spieler = 2</td><td rowspan="2">∅</td></tr>
<tr><td>Jetztfeld(1)= Figur 1

Incr Figur 1</td><td>Jetztfeld(2)= Figur 2

Incr Figur 2</td><td>Nun muss noch geklärt werden, welcher Spieler den neu dazugekommenen Stein abgesetzt hat. Unter „Figur 1“ ist die momentan gespielte Figur vom Roboter gespeichert (Werte von 1-3). Unter „Figur 2) dementsprechend die gespielte Figur vom Gegner (4-6)
Danach müssen die Werte noch um eins erhöht werden, da beim nächsten Zug eine neue Figur am Zuge ist. (Incr Figur 1/2)</td></tr>
</table>

Fig. 1: Struktogramm des Einlese-Algorithmus der Spielphase 1

Fig.1 zeigt das Struktogramm für diesen Einlese-Algorithmus. Es zeigt nur einen Abschnitt, insgesamt besteht das gesamte Einlese-Unterprogramm aus 9 solcher Algorithmen, die sich jedoch nur in der Benennung der Mikrocontroller-Anschlüsse und Nummerierung des Arrays unterscheiden, funktional sind sie identisch.

Spielphase 2: Springen:

Diese Phase erfordert einen anderen Einlese-Algorithmus als bei der ersten Phase. Nun kommen keine Steine dazu, sie ändern nur noch ihre Felder. Es muss bestimmt werden, von welchem Feld wohin welche Figur gezogen ist. Fig.2 zeigt das Struktogramm für diesen Algorithmus.

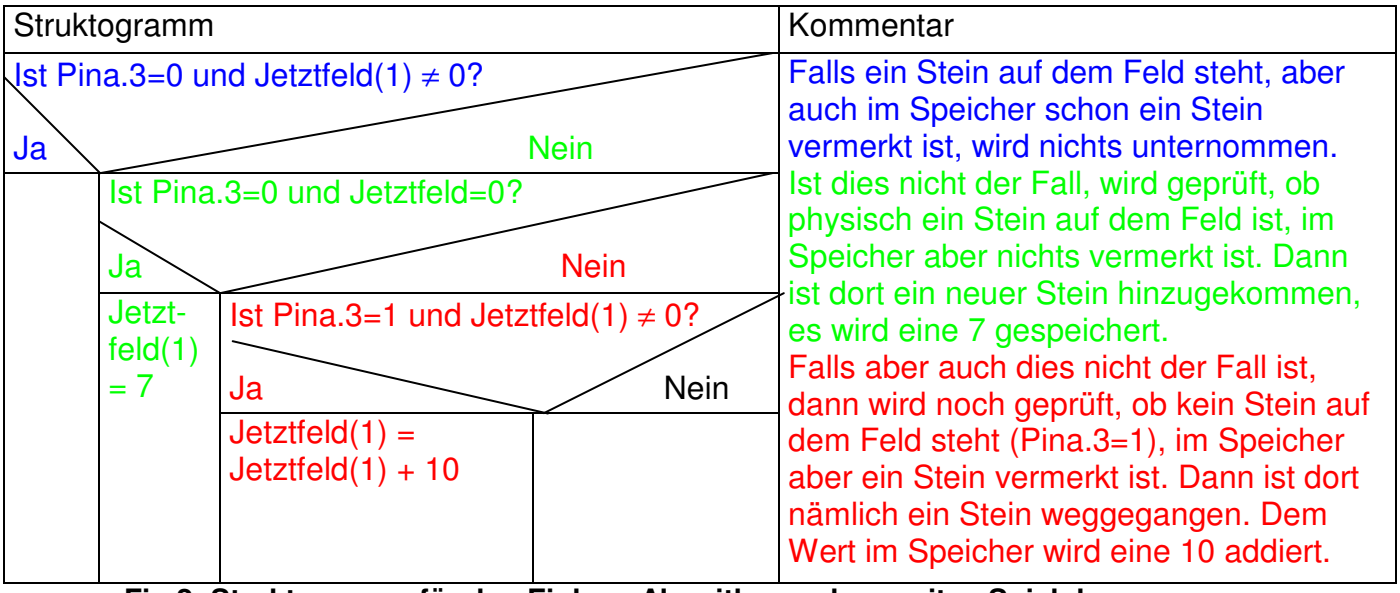


Fig.2: Struktogramm für den Einlese-Algorithmus der zweiten Spielphase.

Hier erkennt man, dass beide Einlese-Algorithmen auf Abfragen beruhen. Es werden lediglich alle möglichen Zustände, die auf einem Spielfeld passieren können, abgefragt (nichts passiert, Figur kommt auf Feld, Figur geht weg).

Auch hier besteht das gesamte Unterprogramm aus 9 solcher Abfrage-Blöcken, für jedes Feld einen.

Da man nicht weiss, welche Figur auf ein Feld gekommen ist, wird unter dem Feld, auf dem die Figur dazukam, eine 7 gespeichert, da diese unter den Spielsteinnummern nicht vorkommt.

Auf dem Feld, auf welchem eine Figur weggegangen ist, wird eine 10 zum Spielsteinwert hinzuaddiert.

Nachdem alle Felder abgefragt wurden, hat man in dieser zweiten Spielphase eine 7 und eine 10+x im Jetztfeld-Array. Vom 10+x Feld zieht man die 10 wieder ab und man weiss, welche Figur auf das Feld mit der 7 gekommen ist, sie wird durch die korrekte Nummer ersetzt. Beim 10+x Feld wird eine 0 geschrieben, denn dort ist ja nun keine Figur mehr.

Nachdem nun der Mikrocontroller alle Felder eingelesen hat, kann er daraus den Spielbaum generieren.

Die Spielbaum-Algorithmen:

Die Software ist dafür ausgelegt, zwei Ebenen tief zu suchen, sprich, sie rechnet zwei Züge weiter, was bei diesem einfachen Spiel völlig ausreicht.
Zudem wurde Breiten- und Tiefensuche kombiniert. Die erste Ebene wird vollständig abgespeichert, es wird in die Breite gesucht. Die zweite Ebene hingegen ist nur teilweise abgespeichert. (Tiefensuche).
Auch hier mussten für die beiden Spielphasen verschiedene Algorithmen geschrieben werden.

Ebene 1, Spielphase 1:

In der ersten Spielphase wird nur mit einer Figur gesprungen, nämlich mit derjenigen, die noch am Rande, ausserhalb des Spielfeldes steht. Zusätzlich nimmt die Anzahl der möglichen Spielfelder ab, denn es kommen immer mehr Steine auf das Spielfeld.
Die Ebene 1 sucht alle möglichen Züge, die der Roboter ausführen kann. Er kann seine Figur also überall dort hinsetzen, wo eine „0“ im Jetztfeld steht.
Alle möglichen Felder werden in einem Array abgespeichert. (Ebene1) Das Array hat 81 Speicherplätze, denn es können 81 verschiedene Spielsituationen auftreten, die alle gespeichert werden müssen. Gespeichert wird „hintereinander“, sprich, die ersten 9 Plätze des Arrays stehen für die Spielsituation 1, die nächsten 9 für die nächste Spielsituation usw.
Fig.3 zeigt das Struktogramm für diesen Algorithmus.
Die Software prüft zuerst, wie viele Zugmöglichkeiten es gibt (wie viele freie Felder). Dann wird die zukünftige Figur, mit der der Roboter spielen wird, auf ein freies Feld gesetzt. Dieser Spielzustand wird im Ebene1-Array gespeichert. Danach wird das nächste freie Feld mit der Figur besetzt und dieser Zustand im Array gespeichert usw. bis alle möglichen Zustände gespeichert sind. Dies ist die Breitensuche, denn es werden alle möglichen Zustände gespeichert, was hier überhaupt kein Speicherproblem darstellt, denn es werden maximal 81Bytes belegt, was relativ wenig ist.

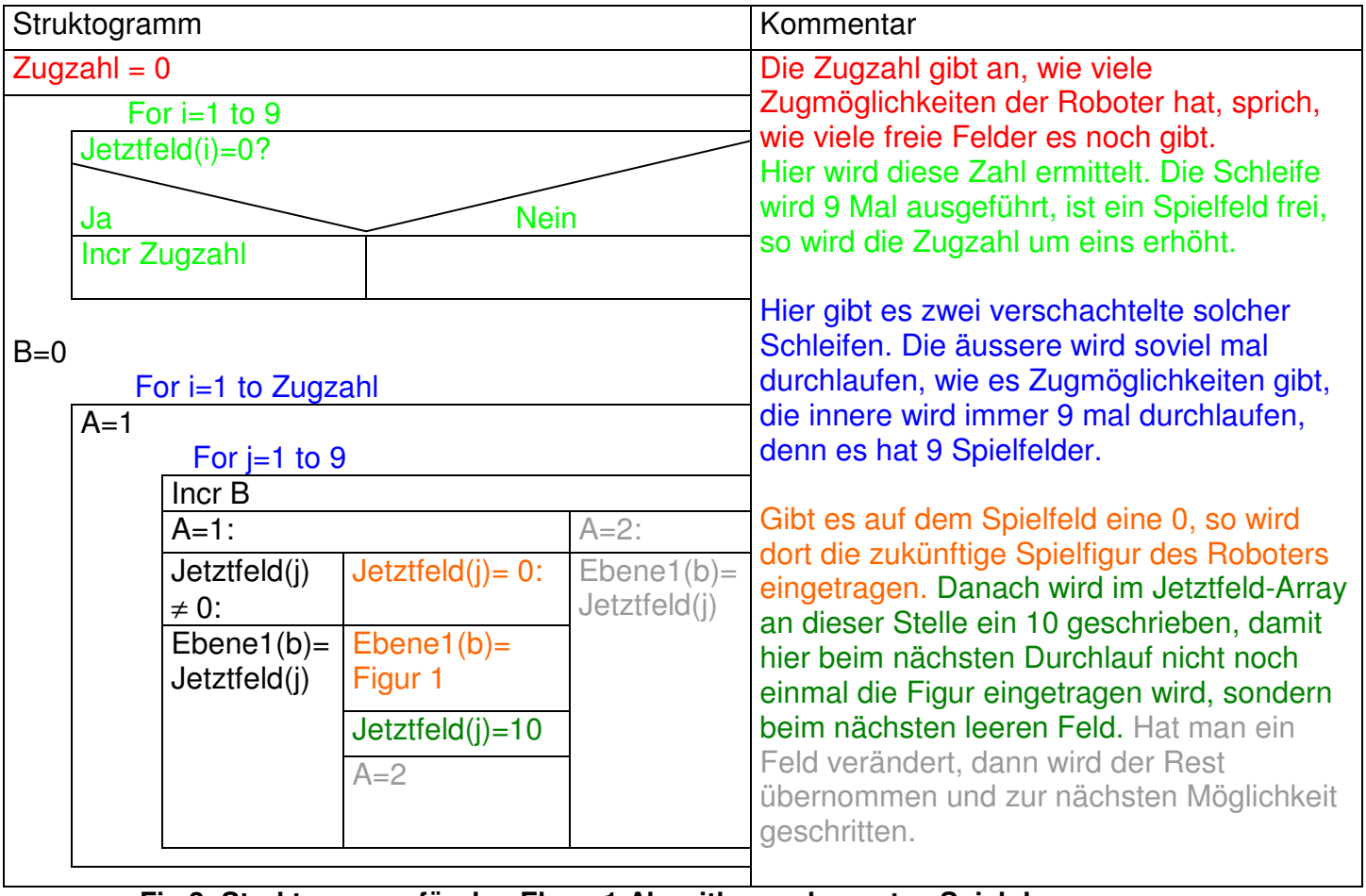


Fig.3: Struktogramm für den Ebene1-Algorithmus der ersten Spielphase

Ebene 1, Spielphase 2:

Im Unterschied zur ersten Spielphase sind hier die Zugmöglichkeiten konstant. Es gibt nämlich immer 9. (3 Figuren und 3 leere Felder). Dafür muss der Algorithmus die verschiedenen Figuren berücksichtigen, er spielt nicht mehr, wie in der ersten Spielphase mit nur einer, sondern mit 3 Figuren, die er bewegen kann.

Auch hier werden alle Möglichkeiten wieder im Ebene1-Array gespeichert. Es werden alle 81 Plätze belegt, da es mögliche 9 Zustände mit je 9 Feldern gibt.

Fig.4 zeigt das Struktogramm für diesen Algorithmus.

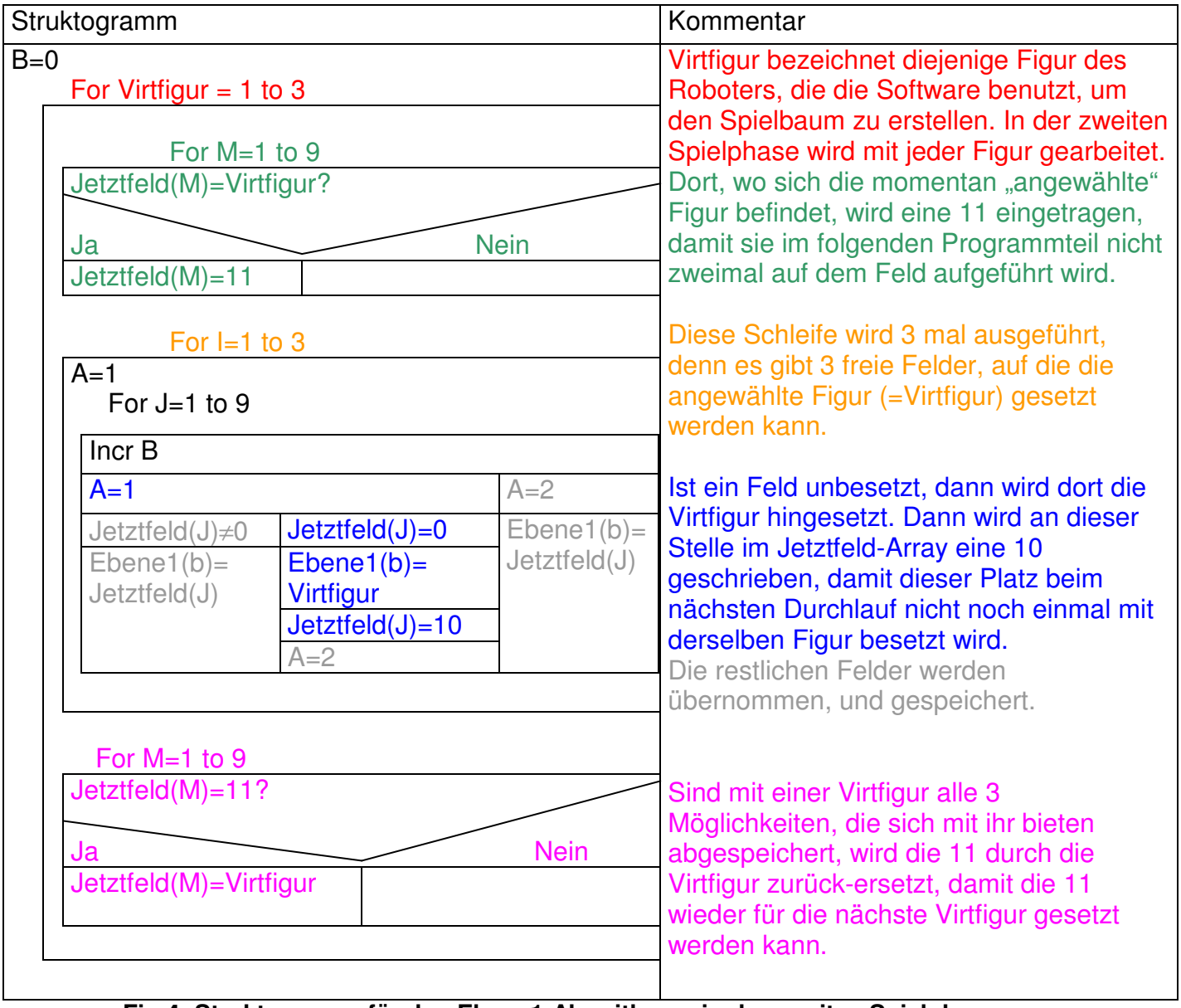


Fig.4: Struktogramm für den Ebene1-Algorithmus in der zweiten Spielphase

Ebene 2, Spielphase 1:

Nachdem die erste Ebene erzeugt und vollständig abgespeichert wurde, werden nun noch die Züge berücksichtigt, die der menschliche Gegner führen kann, es wird die zweite Ebene des Spielbaumes erzeugt. Im Gegensatz zur ersten Ebene wird diese nicht mehr vollständig abgespeichert, da sie zuviel Speicherplatz verbrauchen würde. Stattdessen wird nur von einem Zustand der Ebene 1 alle möglichen Folgezüge des Gegners gespeichert. Dieser Spielzustand, der sich dadurch ergibt, wird dann in die Bewertungs-Funktion geschickt, doch dazu später mehr.

Auch hier nimmt die Anzahl der möglichen Züge stetig ab, denn es werden mit jedem Spielzug weniger Felder. Auch dies muss hier der Algorithmus berücksichtigen.

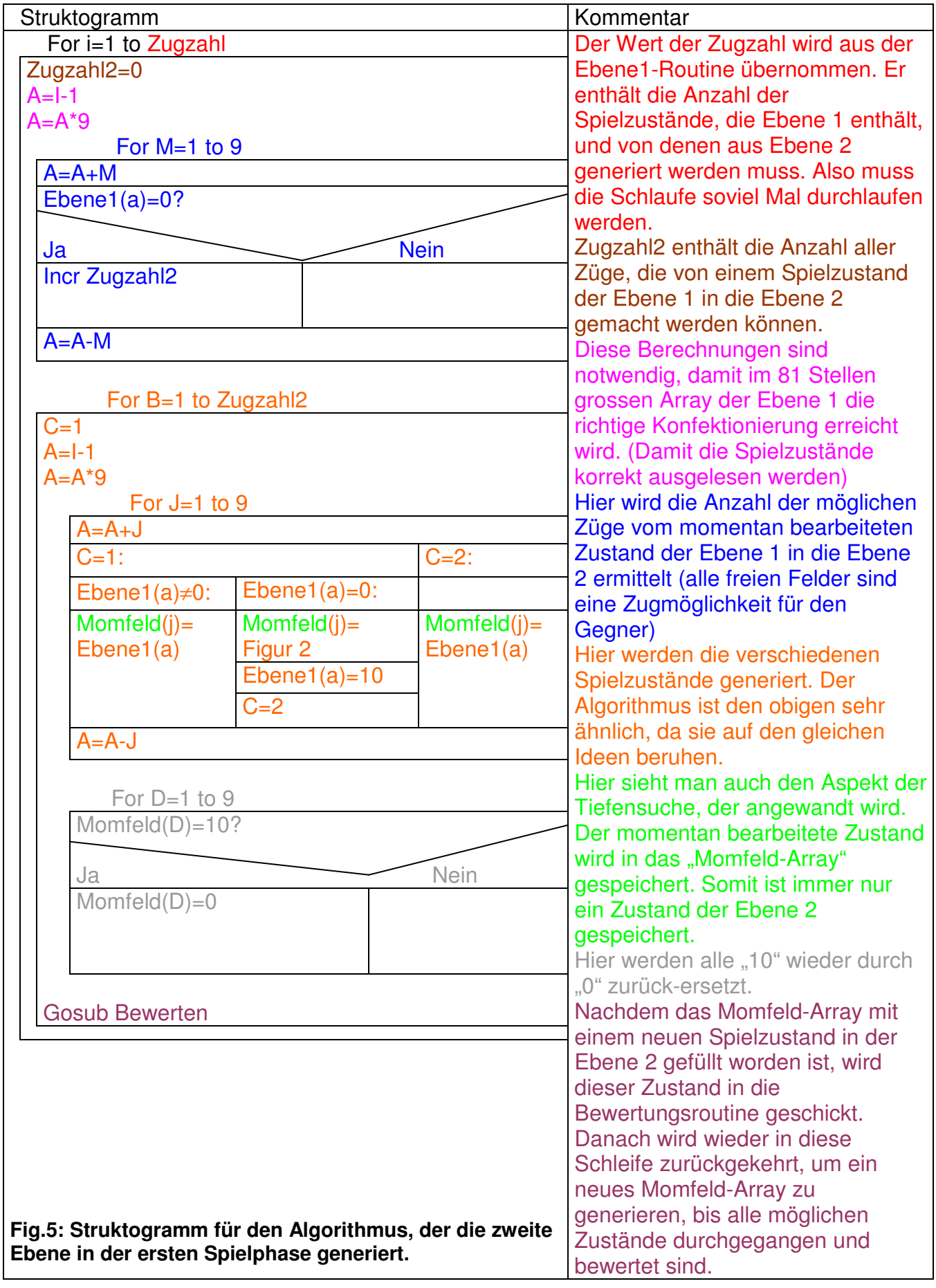


Fig.5: Struktogramm für den Algorithmus, der die zweite Ebene in der ersten Spielphase generiert.

Fig.5 zeigt den Aufbau des Algorithmus, der für die Erzeugung der zweiten Ebene aus der ersten in der ersten Spielphase zuständig ist. Der Aspekt der Tiefensuche, die hier angewandt wird, wird sichtbar. (siehe Kommentar)

Ebene 2, Spielphase 2:

In der zweiten Spielphase muss die Erzeugung der zweiten Ebene auch separat abgewickelt werden. Im Gegensatz zur ersten Spielphase gibt es hier wiederum eine konstante Zugzahl, es gibt immer 9 Möglichkeiten, zu ziehen. (3 Felder, 3 Figuren)
Fig.6 zeigt das Struktogramm für diesen Algorithmus.
Auch hier wird wieder mit „virtuellen“ Figuren gearbeitet, denn der menschliche Gegner hat ja die Möglichkeiten, eine von drei Figuren zu bewegen. Nur wird diesmal von 4 bis 6 durchgezählt, denn das sind die Nummern der Figuren des Gegners.
Der Algorithmus ist sehr ähnlich zum Ebene1, Spielphase 2-Algorithmus, denn auch dort wird mit „virtuellen“ Spielfiguren gearbeitet, um alle Kombinationen zu erzeugen. Wie in Spielphase 1 wird auch hier mit Tiefensuche gearbeitet – es wird jeweils nur ein Zustand ins „Momfeld-Array“ gespeichert, der dann bewertet wird.

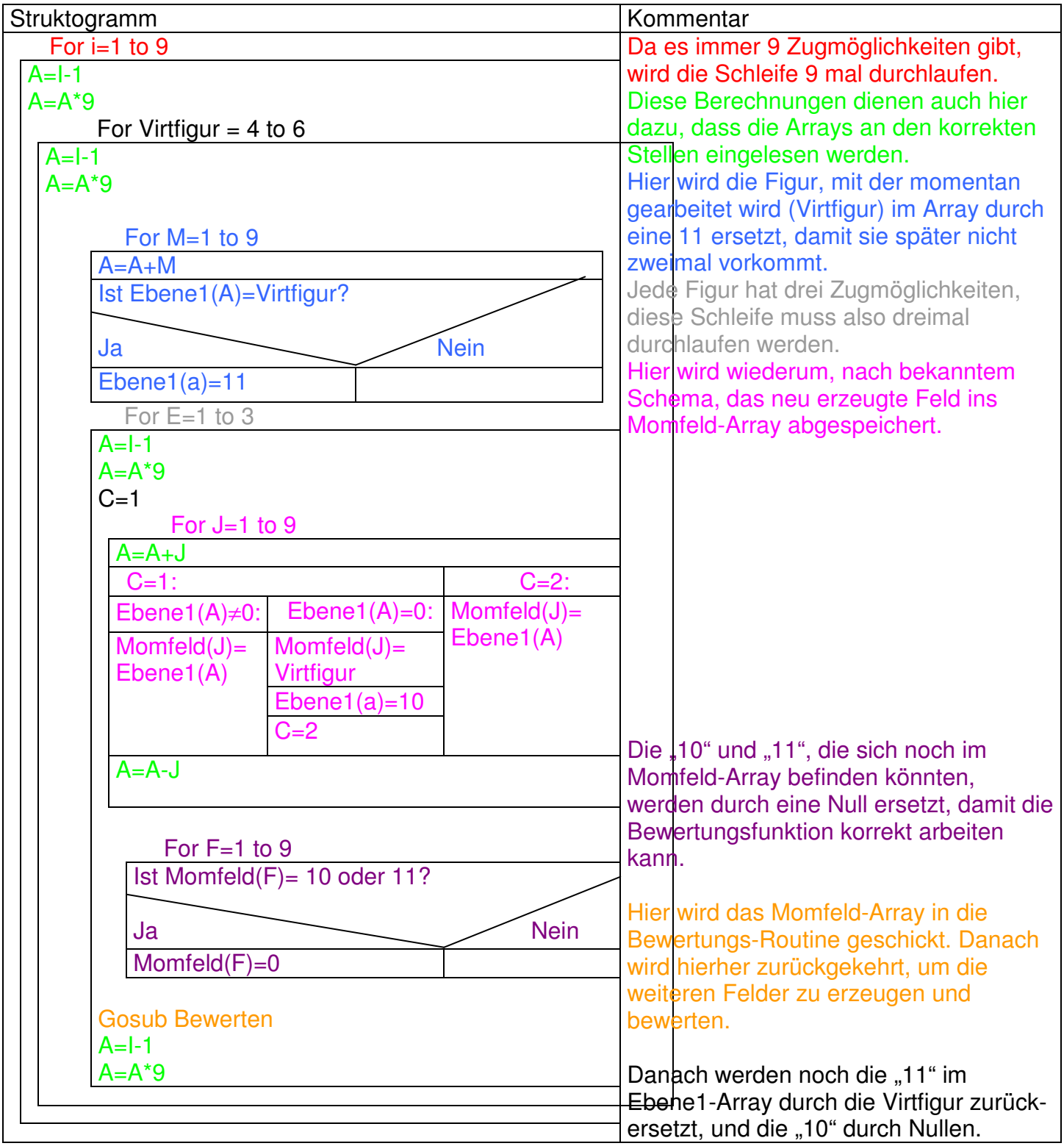


Fig.6: Struktogramm für den Ebene2-Algorithmus der zweiten Spielphase

Die Feldbewertungs-Funktion:

In dieser Funktion werden die Spielzustände aus der zweiten Ebene nacheinander bewertet. Aus dem Ebene2-Algorithmus gelangt ein Spielzustand in diese Funktion, er wird bewertet, danach wird im Ebene2-Algorithmus der nächste Spielzustand generiert, der dann wieder bewertet wird etc.

Bei der Bewertung werden den Spielzuständen numerische Werte zugeordnet, ist ein Spielzustand für den Roboter vorteilhaft, dann kriegt er höhere Werte, als wenn er für den Gegner vorteilhaft ist.

Zuerst hat jeder Spielzustand Null Punkte. Tabelle 1 zeigt, wie die einzelnen Zustände bewertet werden.

Zustand	Punktebewertung
Eine meiner Figuren im Zentrum ➔ 4 angrenzende Felder ➔ mehr Möglichkeiten, eine Dreierreihe zu bilden	+30
Es gibt eine Dreierreihe mit meinen Steinen➔ Sieg	+500
Es hat ein Paar mit meinen Figuren, davon ist kein Stein im Zentrum	+10
Es hat ein Paar mit meinen Figuren, davon ist ein Stein im Zentrum	+15
Diagonal zum gegnerischen Stein steht einer von mir ➔ er kann keine Zwickmühle aufbauen	+25

Tabelle 1: Bewertungskriterien

Nach genau den gleichen Kriterien wird auch die Situation des Gegners untersucht. Nur wird dann der Punktebetrag abgezogen und nicht addiert.

Die Werte, die vergeben werden, wurden experimentell und „per Gefühl“ ermittelt.

Die Software sucht mit Abfragen nach den obigen Zuständen, sprich, es sind alle oben genannten Zustände (Tabelle 1) abgespeichert. Die Software sucht dann, ob ein solcher Zustand im Momentanfeld-Array erreicht ist oder nicht. Die Bewertungs-Routine besteht also grösstenteils aus Abfragen, die prüfen, ob ein bestimmter Zustand tatsächlich vorhanden ist.

Die Bewertungen werden in einem separaten Array gespeichert. Hier tritt auch das Minimax-Prinzip zum Zuge: Es werden nur Werte gespeichert, die grösser sind, als derjenige, der bereits gespeichert ist. Denn der Roboter sucht sich den besten Wert aus, er ist der Max-Spieler.

Das Bewertungs-Array hat 9 Stellen. Denn in der ersten Ebene gibt es maximal 9 Spielzustände. Alle Bewertungen der Zustände der zweiten Ebene, die z.B. vom

ersten Zustand der ersten Ebene abstammen, werden an die erste Stelle des Bewertungs-Arrays geschrieben usw.
Am Schluss ist das Bewertungs-Array gefüllt. Die Software sucht dann diejenige Stelle, die den höchsten Wert enthält. Dies ist der für den Roboter beste Zug, unter Berücksichtigung der obigen Bewertungskriterien.
Nun liest die Software den Spielzustand im Ebene1-Array aus, der für am Besten befunden wurde. Dieser Zustand wird mit dem momentanen Zustand auf dem Spielfeld verglichen. So kann evaluiert werden, welche Figur wohin bewegt werden muss, um den Zustand aus dem Ebene1-Array zu erreichen. Dazu werden nur zwei Kriterien benötigt: das momentane Feld, auf der sich der Stein befindet und das Zielfeld, auf welches der Stein gesetzt werden soll. Diese beiden Felder werden durch Vergleich des Soll- und Ist-Feldes gefunden und an die Routine geleitet, welche den Greifarm steuert.

Die Bewegungs-Routine

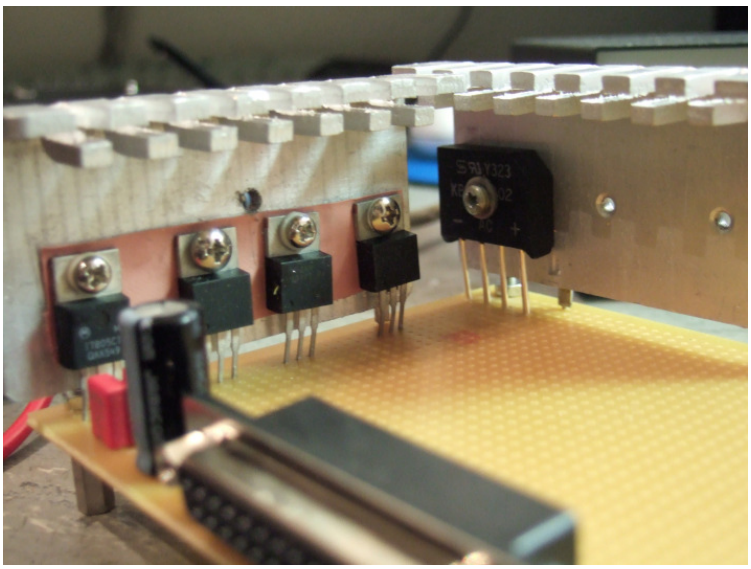
In dieser Routine werden die Figuren auf dem Feld bewegt. Dazu werden die beiden oben genannten Parameter benötigt (Momentanes Feld und Zielfeld der Figur) Diese Routine fährt den Greifarm zuerst über den Stein, der verschoben werden soll, dann aktiviert sie den Elektromagneten, der Stein wird angehoben. Danach wird der Greifarm über das Zielfeld gefahren, und der Stein wird durch Abschaltung des Elektromagneten fallengelassen.
Die Servos werden auf eine ziemlich einfache Art und Weise gesteuert, ihr Bewegungsspektrum wird in der Software durch Nummern „zerlegt“. Z.B. reicht das Bewegungsspektrum von 59 bis 254. 59 ist dabei der linke Anschlag, 254 der rechte. So muss von der Software dem jeweiligen Servo nur eine Nummer zugewiesen werden, und es fährt automatisch an die vorgegebene Position.

Es muss beachtet werden, dass die hier vorgestellten Algorithmen nur ein kleiner Teil (der wichtigste) der gesamten Software ist. Das gesamte Programm umfasst noch einige weitere Funktionen, die hauptsächlich mit der Initialisierung der Hardware zu tun haben, sie sind sehr „Mikrocontroller-spezifisch“.
Zudem wird, unter Benutzung der Spieltheorie, erreicht, dass der Roboter automatisch den für ihn besten Zug spielt und nie verliert (!). Die Spieltheorie verhindert dies effektiv.

5. Werdegang und Entwicklungsprozess der Arbeit

Wenn ich diese Arbeit als Ganzes betrachte, dann Frage ich mich, wie ich es geschafft habe, etwas in so vielen Hinsichten kompliziertes zu bauen und zu programmieren.
Doch man muss beachten, dass dieser Roboter ein langer Entwicklungsprozess hinter sich hat, in dem immer wieder kleine Ergänzungen hinzugefügt wurden, sodass die Arbeit langsam zum komplexen Ganzen heranwuchs.

Nachdem ich mich entschieden hatte, dieses Projekt (Bau eines spielenden Roboters) in Angriff zu nehmen, musste zuerst die Frage geklärt werden, welches Spiel man denn wählen wolle. Ein Kandidat war zunächst die normale Mühle, doch es stellte sich bald heraus, dass dieses Spiel ungeheuer komplex ist und für eine Maturaarbeit schlicht zu Umfangreich wäre. Andere Professoren, die dies verwirklicht haben, taten dies als Diplomarbeit. Mein Betreuer schlug dann die „einfache Mühle“ vor, zusammen mit „4 gewinnt“, welches jedoch aufgrund der komplexeren Hardware ausschied. Also fiel die Entscheidung auf „einfache Mühle“, es ist nicht zu komplex und für eine Maturaarbeit gut zu verwirklichen.
Bevor diese Entscheidung getroffen wurde, beschäftigte ich mich mit dem mathematischen Hintergrund, der Spieltheorie. Dazu diente mir das im Literaturverzeichnis unter Punkt 1 aufgeführte PDF.



Im März 2006 begann ich dann, konkrete Pläne von der Hardware zu entwerfen, ich bestimmte z.B. dass der Roboter in 3 Modulen aufgebaut werden musste, zwecks Portabilität. Gleichzeitig begann ich mit dem Aufbau der Spannungsversorgung, denn diese wurde unbedingt benötigt. Dazu konnte ich gut auf meinen Fundus

Abb.22: Frühstadium des Mainboard-Moduls an Bauteilen, den man als Hobby-Elektroniker hat, zurückgreifen, so stammt z.B. der Transformator aus einer ausgeschlachteten Stereoanlage.
Abb. 22 zeigt ein frühes Stadium des Mainboard-Moduls.
Ich hatte zu der Zeit auch schon genaue Vorstellungen, wie der gesamte Roboter auszusehen hat, also begann auch die Beschaffung der „speziellen Bauteile“ wie Mikrocontroller und Servos sehr früh.

In den Frühlingsferien 2006 wurde die Software so aufgebaut, dass sie damals schon funktionsfähig war. Das Programm, mit welchem ich den Controller programmiere, hat einen Simulator eingebaut, mit diesem hab ich bereits damals gegen den Simulator gespielt, also noch ganz ohne Hardware.
Um die Software zu entwickeln habe ich meine Ideen auf einen A3-Notizblock niedergeschrieben, was sich als sehr nützlich erwies, so konnte ich alle Speicherstrukturen und Abläufe koordiniert entwickeln. Dazu muss noch gesagt werden, dass sich so eine Software hauptsächlich „aus Fehlern“ entwickelt. Man programmiert immer nur kleine Abschnitte und Routinen, danach werden diese getestet, funktionieren sie nicht, dann gehe ich die jeweiligen Routinen von Hand durch, um die Fehler zu finden und sie auszumerzen.
Es ist noch anzumerken, dass es bei der Spieltheorie noch „Vereinfachungsstrategien“ gibt, die es ermöglichen, nicht den gesamten Suchbaum durchsuchen zu müssen. Diese wurden im Programm jedoch nicht angewandt und wurden deshalb bei der Erklärung der Spieltheorie nicht aufgegriffen. Der Controller benötigt nämlich für die Errechnung seines Zuges etwa 3 Milisekunden, was eine Vereinfachungsstrategie nicht nötig macht.

Diese Arbeit (Erstellung der Software) musste so früh getätigt werden, denn einzig und allein von der Software hing es ab, ob die Arbeit gelingen würde oder nicht, denn ihre Entwicklung war für mich die grösste Hürde in dieser Arbeit; ich habe zum ersten Mal eine ernsthafte mathematische Theorie in einer komplexen Software umgesetzt. Die Hardware war zwar auch anspruchsvoll, doch ich hatte aus dem Hobby bereits Erfahrungen mit dem Aufbau solcher Schaltungen. Trotzdem bleibt diese Hardware bis jetzt meine grösste Errungenschaft.

Die Halterungen des Greifarmes und auch das Trafo-Gehäuse habe ich aus Plexiglas gefertigt, denn es sieht relativ gut aus und lässt sich ein Vielfaches leichter bearbeiten als Metall.
Die Blende des LCD besteht hingegen aus Edelstahl, es war sehr schwierig ihn zu bearbeiten, da er enorm zäh ist und die Werkzeuge stark verschleisst, doch es hat sich gelohnt.
Ich habe mich für eine Mikrocontroller-Steuerung entschieden, da diese im Gegensatz zu einer Steuerung über einen üblichen PC sehr nah an der Hardware abläuft und auch um ein Vielfaches leichter zu bewerkstelligen ist. Als Controller habe ich, wie schon erwähnt, einen AtMega128 gewählt, was ein ziemlicher „Rolls-Royce“ unter den Mikrocontrollern ist. Ich habe mich damals für einen so grossen Controller entschieden, da ich damals noch nicht wusste, wie umfangreich die Software werden würde, also wollte ich lieber auf der sicheren Seite bleiben. Müsste ich diesen Roboter nochmals bauen, würde ich mit Sicherheit einen kleineren und billigeren Mikrocontroller wählen, da auch diese genügend Ressourcen haben.

Dass ich den Greifarm mit Servos steuern wollte, war auch ziemlich schnell klar, denn sie lassen sich sehr einfach ansteuern und haben genügend Kraft, um den Arm zu bewegen. Als Baumaterial für den Arm selbst wurde „Mecano“ verwendet, da ich damals noch nicht die Mittel hatte, solche Metallteile selbst herzustellen, und dies auch zu aufwendig gewesen wäre. Ein Nachteil des Greifarmes ist seine Lagerung auf der dünnen Achse des untersten Drehservos. Dadurch gelangt der Arm leicht ins Schwanken und es muss vor dem Aufheben oder Absetzen der Figuren etwas gewartet werden, damit sich der Arm beruhigen kann. Ansonsten ist die Konstruktion recht ausgereift, es gab mehrere Prototypen, die ich ohne Pläne frei von Hand zusammenbastelte, bis sich dieser momentan verwendete Typ ergab.
Um die Figuren zu bewegen kam mir auch relativ schnell ein Elektromagnet in den Sinn, denn eine Greifzange wäre vom Hardwareaspekt schwerer umzusetzen gewesen. Zudem wollte ich als Sensoren die Reed-Sensoren verwenden, da sie sehr leicht auszuwerten sind und sich gut per Magnet auslösen lassen. Also verwendete ich den kleinen Magneten in den Spielstein zugleich dazu, um die Steine per Elektromagnet anheben zu können.
Die Spielsteine selbst wurden aus einem Besenstiel gefertigt. In ihre Unterseite wurde ein kleines Loch gebohrt, in das der Magnet eingeklebt wurde. Danach wurde

das Loch verspachtelt, die ganze Figur verschliffen und mehrmals lackiert. Leider war der Lack sehr lange klebrig, sodass sich an den Stellen, an denen die Figur auf dem Spielbrett aufliegt, ein schwarzer Fleck bildet.

Nachdem der Greifarm und das Mainboard-Modul mit der Spannungsversorgung fertiggestellt waren, ging es um den Bau des Spielbrettes und der Grundplatten. Zuerst wollte ich für die Grundplatten einen Kunststoff wie z.B. Polyethylen verwenden, doch dieses Material erwies sich als zu teuer.
Das Spielfeld, das ja doch das Zentrum des ganzen Roboters ist, wollte ich möglichst schön gestalten, also wendete ich mich an einen befreundeten Schreiner Lehrling (Michael Leutenegger), der mir mit seiner Erfahrung und Werkstatt beistand und mir auch das Material für die Grundplatten und das Spielfeld besorgen konnte, auch hier nochmals ein herzliches Danke!
Danach ging es ziemlich zügig vorwärts, in den Sommerferien wurden die Module fertiggestellt und getestet. Danach wurden sie auf den Grundplatten befestigt. Abb.23 zeigt einen Testlauf des Greifarmes

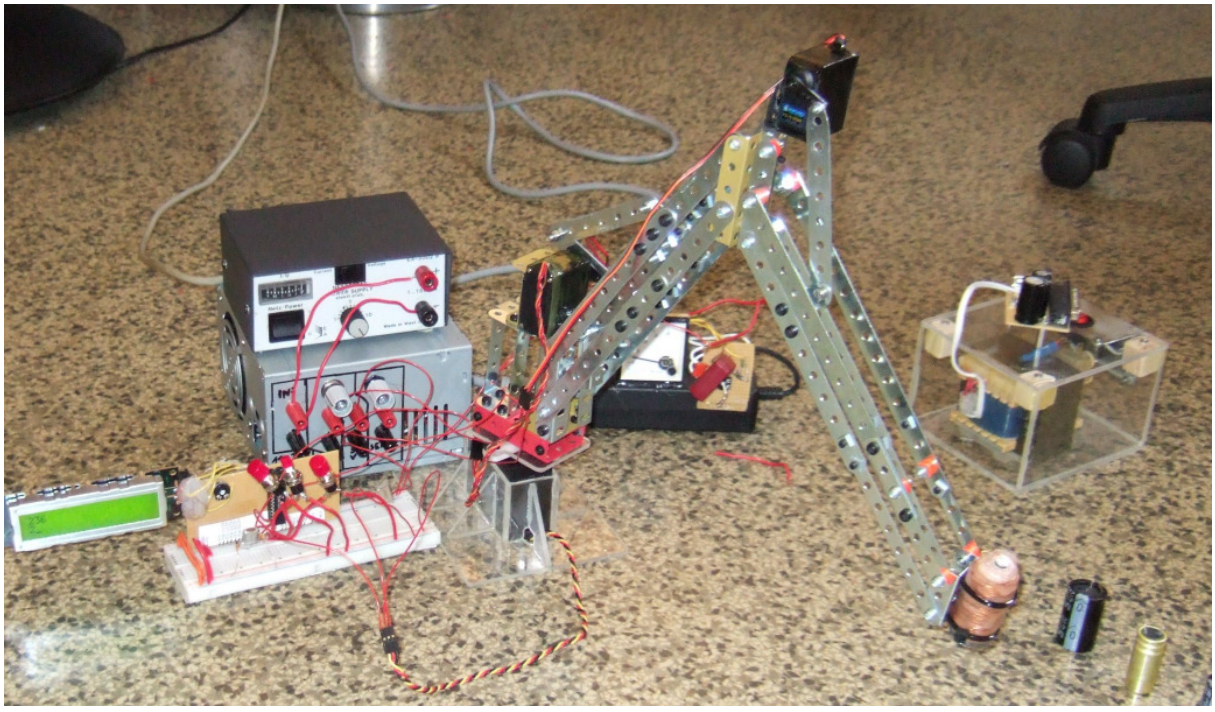


Abb.23: Testlauf des Greifarmes

Beim Spielfeld-Modul wurden die Kabelkanäle auf der Unterseite eingefräst und die Sensoren eingeklebt und verkabelt. Ebenso wurde das LCD angeschlossen.

Zudem wurde zu dieser Zeit auch nochmals die Software ein wenig erweitert und verbessert, besonders diejenigen Funktionen, die den Kontakt zum menschlichen Spieler herstellen (z.B. Fehlererkennung der Spielsteine, Ausgaben des LCD) wurden programmiert.
Nachdem alle Grundplatten und das Spielfeld lackiert waren, wurden die einzelnen Module zum ersten Mal zu einem lauffähigen Ganzen zusammengebaut.
Abb.24 zeigt einen Probe-Zusammenbau der noch unlackierten Grundplatten.

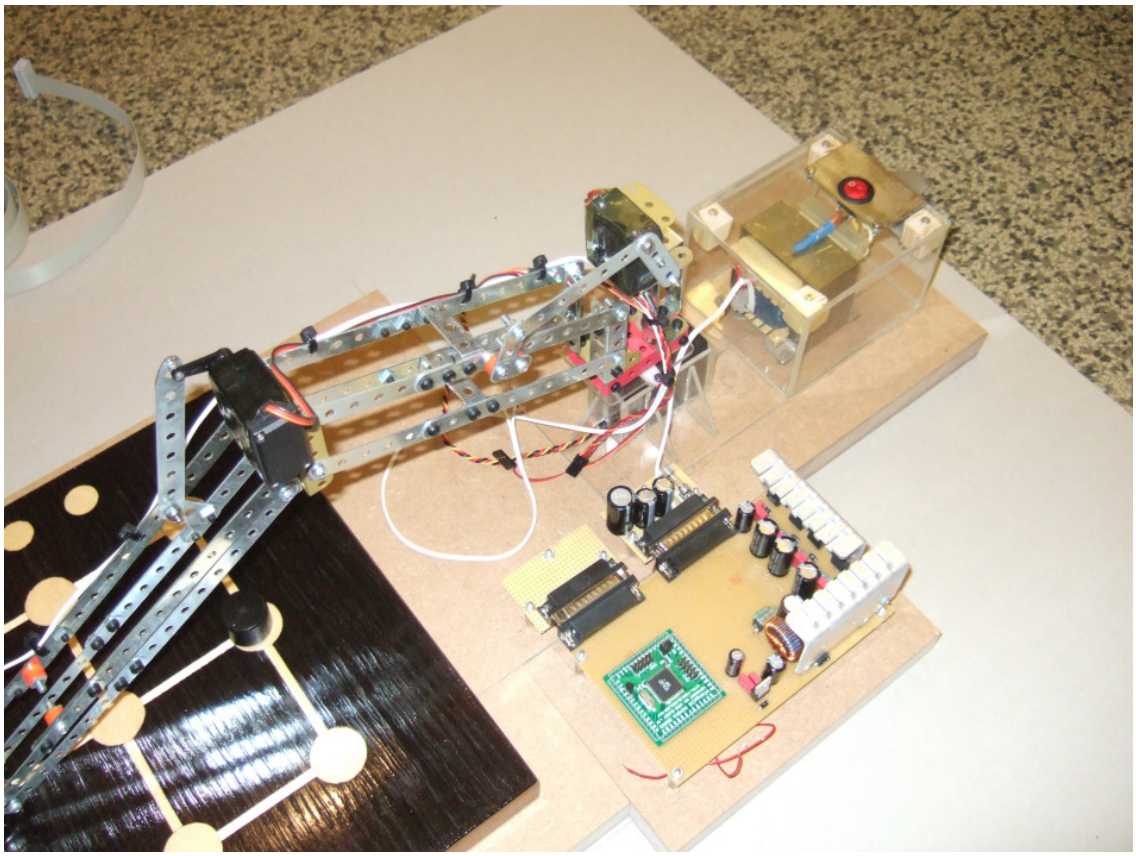


Abb. 24: Probe-Zusammenbau

Die Software, die ich schon sehr früh entwickelte, erwies sich tatsächlich als lauffähig, und nachdem noch die letzten kleinen Fehler beseitigt wurden, spielte der Roboter doch tatsächlich einfache Mühle.

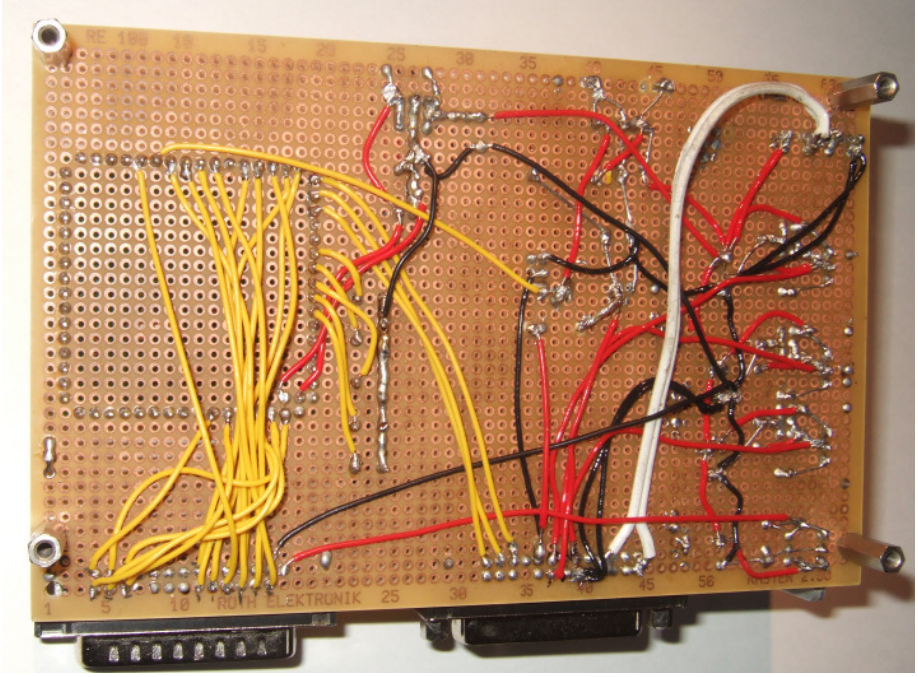


Abb.25: Verkabelung des Mainboards

Abb.25 zeigt noch die Verkabelung unter dem Mainboard. Diese Art von Aufbau ist sehr flexibel, was ich auch benötigte, denn ich musste spontan auch noch weitere Komponenten (z.B. den Lüfter) hinzufügen können.

Doch die Software wurde stetig verbessert, jetzt, da der Roboter spielte, zeigten sich noch die letzten Fehler und Ungereimtheiten. So wurde der Roboter zuerst programmiert, dass er beginnen durfte. Dies wurde noch geändert, nun kann der menschliche Spieler beginnen. Hier ist auch nochmals anzumerken, dass der Roboter nie verliert. Die Spieltheorie verhindert dies bei diesem einfachen Spiel effektiv.

Um den Roboter ausgiebig zu testen, wurde er von mehreren Personen und von mir oft bespielt. Dabei zeigten sich auch noch stark versteckte Fehler in der Software. Als meine Mutter gegen ihn spielte, gewann sie plötzlich. Was war passiert? Sie hatte (zufälligerweise) eine Zwickmühle beim Setzen der Steine aufgebaut, die die Bewertungsroutine zu dieser Zeit noch nicht kannte. So wurde die Routine noch um diesen Punkt erweitert, und es ist nun nicht mehr möglich, eine Zwickmühle aufzubauen. Es wurde auch nicht programmiert, dass der Roboter diese Zwickmühle aufbauen soll, denn das wäre unfair und das Spiel wäre nach 3 Zügen vorbei, also auch langweilig für den Spieler.

Zudem zeigten sich durch das stetige Spielen tiefere Grundzüge hinter dem Spiel „einfache Mühle“. So fand ich heraus, dass es nur eine einzige Zwickmühle gibt, diese könnte man beim setzen der Steine aufbauen – würde es der Roboter nicht verhindern. Auch während dem Spiel gibt es keine Taktik, um Zwickmühlen aufzubauen, das Spiel ist schlicht zu einfach und übersichtlich. Ich habe dem Roboter alle möglichen Zwickmühlen die es gibt (24 an der Zahl) einprogrammiert, er solle sie ausführen, falls es geht. Doch es geht nie, es ist immer eine andere Figur im Weg. So zeigt sich auch, dass eine Suchtiefe von zwei Ebenen völlig ausreichend ist,

es reicht für ein so einfaches Spiel aus, tiefer zu suchen würde nichts bringen, da es schlicht keine Taktik gibt, die durch ein tieferes Suchen erreicht werden könnte.

So entwickelte sich die gesamte Arbeit über einen recht langen Zeitraum zu einem komplexen Roboter, über den ich auch jetzt noch staune, dass er so gut funktioniert. Sehr viele Teile des Roboters entwickelten sich auch durch experimentelles Herausfinden, so war z.B. der Lüfter nicht geplant, musste aber eingebaut werden, damit die Kühler nicht zu heiss wurden.

6. Analyse und Fazit der Arbeit

Ich bin mit meinem Projekt sehr zufrieden. Es ist mir gelungen, einen Roboter zu bauen, der mit Hilfe einer mathematischen Theorie „einfache Mühle“ gegen einen humanen Gegner spielt. Das Einzige, was mich stört, ist, dass die Reed-Sensoren nicht genau unter den Spielflächen platziert wurden, ich habe falsch gebohrt. Also müssen die Spielfiguren immer leicht „abseits“ abgestellt werden…
Doch dies ist nur eine Kleinigkeit, und viele Personen, die den Roboter sahen, waren begeistert, denn es ist immer erstaunlich, wenn etwas „technisches“ in die Welt des Menschen eindringt. Denn ein Spiel spielen ist ja meistens den Menschen vorbehalten, doch auch Maschinen können es.
Dies war auch der grösste Anreiz für mich, diese Arbeit zu tätigen, denn mich interessierte es, wie so ein Roboter Taktik und Spielgefühl aufbauen kann.
Zudem war die Arbeit sehr stark mit meinem Hobby verbunden, ich konnte sehr viel davon einfliessen lassen, um die Hard- und Software zu entwickeln.
Auch das Schreiben dieses Berichtes bereitete mir viel Spass, da ich nie das Internet und nur selten ein Dokument zu Rate ziehen musste.

Fazit: So ein Roboter zu bauen ist anspruchsvoll, doch es lohnt sich, man sammelt neue Erfahrungen und es bereitet grosse Freude.

Literaturverzeichnis:

1.)
Martin Hirt, Daniel Matter, Rolf Bänziger, Werner Hartmann (1999).
Gruppenunterricht zum Thema Spieltheorie
http://www-is.informatik.uni-oldenburg.de/~dibo/teaching/java0102/spiele/spieltheorie.pdf
(März 2006)
2.)
Florian Lemmerich, Bastian Späth (2005)
Suchoptimierung und Evaluation bei Zwei-Personen-Nullsummen-Spielen am Beispiel von Mühle
http://www6.informatik.uni-wuerzburg.de/projects/other/muehle/Muehle_Lemmerich-Spaeth.pdf (März 2006)

Anhang:

Im Anhang befindet sich eine CD-Rom, auf der sich dieser Bericht in digitaler Form, das gesamte Programm, sowie weitere Fotos befinden.
Zum Lesen des Berichtes und des Programms wird der Adobe Acrobat Reader benötigt.