Design und Realisierung von “kostengünstigen“ fussballspielenden Robotern

05. Mai 2003

Eine Studienarbeit von:

Michael Manger

Hochschule für angewandte Wissenschaften

Berliner Tor 7

20099 Hamburg, Germany

Betreuer: Prof. Dr. Kai von Luck

Inhaltsverzeichnis

Kapitel 1

Einleitung

Der Informatik wird oft vorgeworfen, sie sei zu abstrakt, nicht greifbar und sogar etwas realitätsfremd. Mit Hilfe des Robotfußballs[RoboCup] möchte man die wissenschaftliche Seite der Informatik der breiten Menschheit näher bringen.
Robotfußball soll demonstrieren, wie weit die Wissenschaft im Feld der Steuerung- und Regelungstechnik, digitaler Bildverarbeitung, künstlicher Intelligenz, autonomer Multiagentensysteme und der Sensorik schon fortgeschritten sind. Die Lösungen, die bei solchen gebietsübergreifenden Problemen wie Roboterfußball entstehen, sind in vielen, auch zukünftigen Einsatzgebieten, wie im Katastropheneinsatz, Haushalt oder auch industrielle Fertigungsanlagen, von autonomen Systemen effektiv und effizient einsetzbar. Somit ist Roboterfußball ein nicht zu unterschätzendes, wissenschaftliches Werkzeug, dem es gelingt, eine Problemlösung mit Hilfe von verschiedenen Gebieten der Wissenschaft zu erreichen. Und das aller wichtigste ist, es macht Spaß...
Die Idee der fußballspielenden Roboter stammt ursprünglich aus dem Jahr 1995. Professor Jon-Hwan Kim von KAIST, Korea, gründete ein Komitee zur Organisation von MiroSot (Micro-Robot World Cup Soccer Tournament) Wettkämpfen. Dieses Komitee wurde nach Zusammenschluss einiger Teilnehmer zur FIRA [FIRA] (Federation of International Robot-soccer Association) umgenannt. 1997 wurde eine weitere Organisation (RoboCup) gegründet. Beide Organisationen unterscheiden sich in den Größen der Roboter und in den Spielregeln.
Durch regelmäßigen Turnierveranstaltungen werden die Roboter immer effektiver und für die breite Masse teils nicht mehr bezahlbar.
Mit dieser Studienarbeit möchte ich versuchen die Ziele der großen Vorbilder durch geringere Kosten zu imitieren.
Im Kapitel 2 werde ich das Anwendungsszenarium vorstellen. Anschließend im Kapitel 3 beschäftige ich mich mit dem Design der Welt. Darunter ist nicht nur der Aufbau des Spielfeld, sondern auch die Konstruktion des Roboters zu finden. Im Kapitel 4 beschäftige ich mich der Programmierung der Roboter.

Kapitel 2

Allgemeine Aufgabenstellung

Die Aufgabe besteht darin, zwei Roboter zu bauen und zu programmieren, die in einem begrenzten Feld gegeneinander Fußballspielen. Um diese Aufgabe zu realisieren, sollte ein gewisses Grundmaterial zur Verfügung stehen, welches mir die HAW - Hamburg bereitstellt.

2.1 Leistungsumfang

Diese Roboter soll sich in einem begrenzten Feld autonom bewegen und einem Ball in ein ihr zugewiesenes Tor schießen. Abgesehen vom Zentralen Element des MIT 6.270 Board, sollen sie sich nur mit Hilfe von analogen Infrarot Entfernungsmessern (Sharp GP2D12) den Ball und Gegner erkennen und sich in der Umgebung orientieren.

2.2 Hardwareanteil

2.2.1 Das Gehirn

Die Zentrale Hardware des Roboters ist eine Konstruktion von Fred Martin, Pankaj Oberoi und Randy Sargent [Martin] und wurde schlicht als MIT 6.270 Board [MIT6.270] (siehe Abbildung 2.1) bezeichnet. Die CPU besteht aus einem Motorola 68HC11.
Dieses Board beinhaltet mehrere Anschlussmöglichkeiten für diversen Aktuatoren und Sensoren. Weiterhin können die Sensoren an digitalen oder analogen Schnittstellen angeschlossen werden. Für Kontrollen hat das Board ein zwei Zeilen Display.

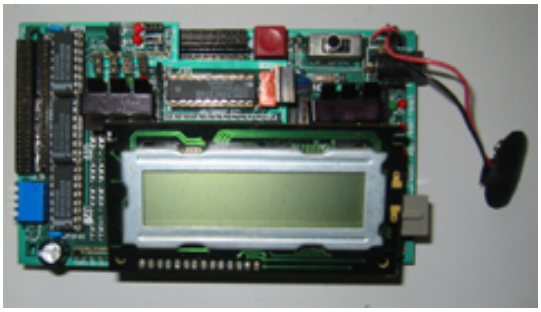


Abbildung 2.1: MIT 6.270 Board

2.2.2 Die Augen

An Sensoren stehen mir Abstandssensoren der Firma Sharp vom Typ GP2D12 [SHARP] (siehe Abbildung 2.2) zur Verfügung. Der große Vorteil dieser Sensoren ist, dass sie nicht Lichtempfindlich sind. Der zuverlässige Arbeitsbereich dieser Sensoren beträgt 10 - 80 cm. Der Stromverbrauch eines Sensors liegt bei ca. 50mA/h. Die Versorgung der Sensoren wird über einen Verstärker gewährleistet. Dieser Verstärker, der vier Anschlussmöglichkeiten besitzt, verfügt über einen eigenen 7,2 V Stromversorger. Die Signale werden über Operationsverstärker an die Ausgänge geleitet und an die analogen Eingänge des MIT 6.270 Boards gelegt.

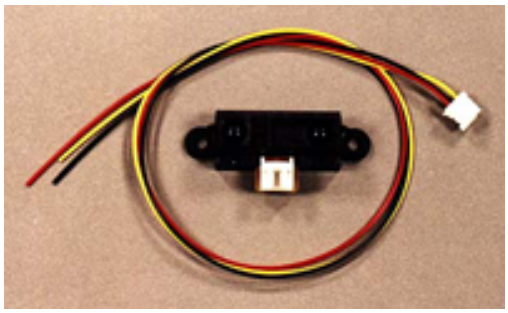


Abbildung 2.2: Sharp Sensor vom Typ GP2D12

2.2.3 Die Füße

Die Plattform basiert auf einer Differentialsteuerung. Dies bedeutet, dass es ein Pendelrad am Heck des Roboters gibt. Im Frontbereich gibt es auf der linken und rechten Seite jeweils ein angetriebenes Rad. Mit diesen Antriebsrädern kann der Roboter angetrieben und gelenkt werde. Bei dieser Bauweise kann sich der Roboter um den Mittelpunkt der Antriebsachse drehen und ist dadurch sehr wendig.
Diese Antriebselemente sind umgebaute Servo Motoren (siehe weiter unten), die aus einem DC Motor und Getriebe besteht.

2.2.4 Die Finger

Um zu erkennen, ob der Roboter unabsichtlich eine Wand oder Gegner berührt wird er mit Bumpern (siehe Abbildung 2.3) ausgestattet. Diese Bumper sind Mikroschalter, die entweder an dem digital oder analog Eingang des MIT 6.270 Board angeschlossen werden.

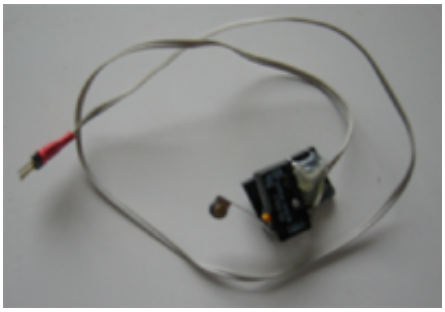


Abbildung 2.3: Mikroschalter

2.2.5 Die Ohren

Damit der Roboter erkennt, wo sein gegnerisches Tor ist, wird der Roboter mit Infrarotsensor für moduliertes Infrarot Licht ausgerüstet (siehe Abbildung 2.4). Dieser Sensor dient zum Finden von sogenannten Infrarot Beacons bzw. Leuchtfeuern (siehe Abbildung 2.5). Der Sensor erkennt moduliertes Infrarot Licht mit einer Trägerfrequenz von 40 kHz. Auf die Trägerfrequenz kann eine Frequenz von 100 Hz oder 125 Hz moduliert werden. Die Sensoren können an die Ports 4 bis 7 des MIT 6.270 Boards angeschlossen werden. Sie müssen jedoch abgeschirmt werden, damit nicht fremdes Licht das Ergebnis verfälscht.

Abbildung 2.4: Beacon: Ist auf dem Roboter zu befestigen

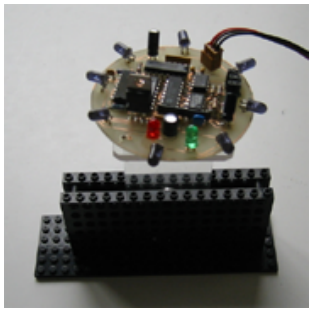


Abbildung 2.5: Beacon-Sender: Am Torbereich zu befestigen

2.2.6 Der Körper

Das Gerüst der Prototypen ist aus einzelnen Steinen der LEGO - Technik aufgebaut. Aus Zeitgründen wurde nur eine Art Prototyp (Starrauge, siehe Anhang C) mit allen Komponenten aufgebaut. Die anderen beiden Prototypen (Einauge und Zweiauge, siehe Anhang C) beinhalten nur Komponenten, die für die Versuche notwendig sind.
In dieser Studienarbeit wurde auch aus Zeitgründen auf den Aufbau der Roboter durch beständiges Material verzichtet. Dies werde ich im Bereich meiner Diplomarbeit nachholen, da die Roboter zusätzlich mit anderen Bauteilen ausgestattet werden und ein anderes Regelwerk (RoboCup Junior [RoboCupJr]) bekommen.

2.2.7 Servo-Motoren / Getriebe-Motor

Die Servo Motoren, die aus dem Bereich des Modellbaus stammen, wurden aus kostengründen zu einem Getriebe Motor umgebaut. Dabei wurde aus dem Servo Motor lediglich die Logik und die physikalischen Begrenzer entfernt. Die Servo Motoren lassen sich um einen festen Winkelbetrag drehen. Bestandteil sind ein DC Motor (siehe Abbildung 2.6), ein Getriebe und ein Positionssensor mit dessen Hilfe über eine Schaltung der Drehwinkel kontrolliert werden kann. Die Positionierung geschieht durch Pulsweitenmodulation (PWM) und die Kontrollinformation durch den Positionssensor. Der Servo Motor ist durch das eingebaute Getriebe relativ kräftig und lassen sich leicht montieren und ansteuern.

2.3 Softwareanteile

Als Entwicklungsumgebung wird das Programm Interactive C Version 3.2 [NewTon] von den Newton Research Labs verwendet. Interactive C ist ein Ableger von der Programmiersprache C. Es bietet eine Bibliothek mit diversen Funktionen, die speziell

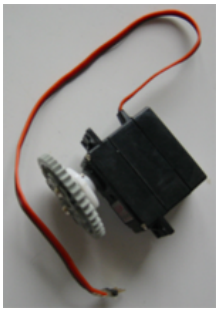


Abbildung 2.6: Motor: Von aussen identisch, wie ein Servo

für das MIT 6.270 Board bestimmt sind. Interactive C bietet leider nur einfache Kontrollstrukturen wie *for, while, if-else* und *break*. Ein einfaches Multitasking ist auch möglich.
In der Interactive C Bibliothek gibt es Funktionen für das Einlesen von Sensordaten und die Bereitstellung von Steuerdaten auf die Ausgänge. Es gibt Zeit-, Ton-, Menü- und Diagnose-Funktionen.

2.4 Vorraussetzungen

Um das Projekt im Bereich der “kostengünstigen“Klasse zu halten, sollten alle Hilfsmittel, die für das Fußballspiel nötig sind, nicht den Rahmen sprengen. Das heißt, der Roboter sollte nicht nur mit Sensoren vollgestopft werden, sondern eine gute Mischung zwischen Sensorentechnik und Software haben. Weiterhin kann man die Kosten reduzieren, indem man das Fußballspiel in einem abgegrenzten Bereich stattfinden lässt. Sprich, das Feld wird mit eine Wand umschlossen.

2.5 Persönliche Motivationen

Im Rahmen von Projekten der HAW [HAW] weckte in mir ein großes Interesse ein Robotfußballspiel mit Hilfe der Motorola Boards zu entwerfen. In dieser Studienarbeit möchte ich unterschiedliche Design- und Programmier-Möglichkeiten ausprobieren, um die Ballerkennung zu verbessern.

Kapitel 3

Design der Welt

Die Welt beinhaltet nicht nur die beiden Roboter, sondern auch das Spielfeld, der Ball und die Tore.

3.1 Design des Spielfeldes

Bei der Entscheidung der Größe des Spielfeldes ist es wichtig die Größe der Roboter und deren technischen Möglichkeiten zu beachten, denn wenn das Spielfeld zu groß gewählt wird, dann können die Roboter eventuell den Ball nicht finden, da die Infrarot Sharp-Sensoren nur eine gewisse Rechweite haben. Wenn das Feld zu klein ist, dann würden sich die beiden Roboter gegenseitig stören und der Spielfluss würde darunter leiden. Eine sinnvolle Größe könnte ein Bereich von 150 cm x 100 cm sein. Dieses Feld wird mit einer Wand bestehen aus Brettern umschlossen. Da der Spielball ein Durchmesser von 70mm hat, sollte die Höhe der Wand größter sein, denn sonst können die Roboter die Wände mit dem Ball verwechseln. Eine gute und sinnvolle Höhe wäre 12 cm, denn dann würde zwischen dem Ball und Ende der Wandhöhe ein Spielraum von 50 mm bestehen.
Ein weiteres Problem besteht darin: Wenn der Ball in eine Ecke rollt, dann könnten die Roboter Probleme bekommen den Ball wieder aus der Ecke zu entfernen und es könnte zu Verklemmungen zwischen beiden Robotern führen. Um dieses zu verhindern ist es sinnvoll aus dem Spielfeld ein Achteck zu machen.
Weiterhin wäre es leichter den Ball zu erkennen, wenn der Ball nicht die Möglichkeit bekommt ganz an der Wand zu liegen. Also ist es eine weitere sinnvolle Konstruktion des Spielfeldes, wenn man an der Wand lang Dreieckleisten der Breite von 30 mm und der Höhe von 10 mm installiert. Das hält auf jeden Fall den Ball von der Wand ab.
Zum Schluss muss noch das Problem mit den Toren geklärt werden. Es gibt zwei Möglichkeiten einer Torerkennung. Die erste besteht aus einer Konstruktion von Bumpern.

Dabei handelt es sich um eine Konstruktion, wo man eine Fläche mit Druckschaltern erstellt, die nach innen ins Spielfeld aufgehängt wird. Wenn der Ball die Fläche berührt, dann wird ein Tor signalisiert. Das Problem besteht nur darin, dass auch die Roboter die Flächen berühren und dabei ein Torsignal auslösen können. Eine sichere Torkonstruktion besteht darin eine Fläche in der Wand auszusägen, sodass der Ball das Innere des Spielfeldes verlassen kann. Das Tor sollte eine Größe von 15 x 8 cm aufweisen.
Damit die Roboter auch wissen, wo die Tore sind, wird in der Mitte der Tore jeweils ein Beacon-Sensor angebracht.

Abbildung 3.1: Spielfeld

3.2 Design der Roboter

Um genauer auf das Design der Roboter einzugehen, muss man sich erst mal überlegen, was von den vorhandenen Materialien der HAW für solch ein Fußballspiel sinnvoll ist. Da wären zum einen die Handhabung der Sharp-Sensoren, die Motoren, die Bumper, der Beacon, das Gerüst des Roboters, die Unterbringung der Batterien, die Schussvorrichtung für den Ball und vor allem das MIT Board zu klären. Der Prototyp des spielenden Roboters ist eine Mischung aus Elektronik, Mechanik und LEGO - Technik. Bei den fertig konstruierten Robotern kann das Gerüst aus LEGO - Technik weg fallen und könnte durch beständiges Material ersetzt werden.

3.2.1 Das Erkennen des Balles

Die schon angesprochenen lichtunempfindlichen Infrarot Sharp-Sensoren sind die einzigen Hilfsmittel, die den Roboter in der Lage bringt, den Ball zu erkennen. Damit der Roboter auch in der Lage ist einen Ball von einem Hindernis zu unterscheiden platziert man zwei Infrarot Sharp-Sensoren so übereinander, dass der untere Sensor (ca. 4 cm Höhe) die Mitte des Balles erfasst und der obere Sensor (ca. 8,5 cm Höhe) über den Ball wegschaut. Wenn nun ein Ball vor den Sensoren auftaucht, dann sind die Werte der beiden Sensoren unterschiedlich und man kann dadurch erkennen, dass es sich um einen Ball handelt. Wenn der Durchmesser des Balles nun zu klein gewählt wird, dann sind die Werte der Sensoren nur minimal Unterschiedlich und es könnte zu Fehlinterpretationen führen. Deshalb wurde der Durchmesser des Balles so gewählt, dass er leicht erkennbar ist. Ein guter und geprüfter Wert ist ein Durchmesser von 70mm.
Ein Roboter sollte mit nicht mehr als vier Sharp-Sensoren beinhalten, da es sonst bei dem MIT Board zu einem Übermaß an Sensorenwerten kommen kann. Ein weiterer Grund, dass jeder Roboter nur vier Sharp-Sensoren bekommt, ist, dass die Schutzschaltung (Verstärker) für die Sharp-Sensoren nur vier Anschlussmöglichkeiten besitzt. Dieser Verstärker ist in der Hinsicht wichtig, da sonst die Sharp-Sensoren die Spannungsversorgung über das MIT Board erhalten und es dadurch zu Zerstörung des Boardes bei falscher Handhabung führen kann.
Es gibt nun einige Möglichkeiten, wie man die Sharp-Sensoren sinnvoll platziert.

Starrauge

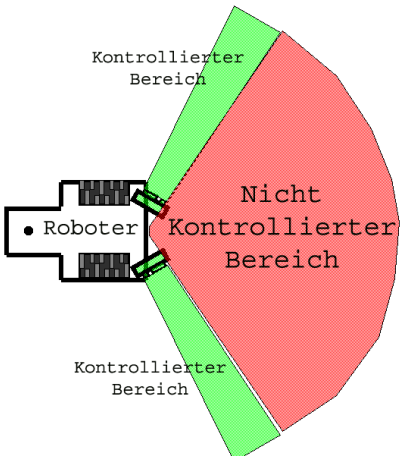


Abbildung 3.2: Designanordung des Starrauges

Die erste Möglichkeit besteht darin die beiden Sharp-Sensoren-Paare (der untere Sensor auf Ballhöhe und der obere über dem Ball) in einem Winkel von ca. 45Grad nach

außen hin ausgerichtet fest zu installieren. Siehe Abbildung 3.2: Starrauge und Prototyp im Anhang C. Der 45Grad Winkel ist in der Hinsicht interessant, da der Roboter nicht nur zur Seite, sondern auch etwas nach vorne schauen kann. So hat der Roboter die Möglichkeit sich von der Wand fernzuhalten und gleichzeitig Hindernisse, die vor ihm auftauchen können rechzeitig zu erkennen. Jedoch hat diese Art Anordnung einen gravierenden Fehler: Direkt vor dem Roboter ist ein Bereich, den die Sharp-Sensoren nicht überwachen können. Eventuell deckt man diesen Bereich mit einem weiteren Sharp-Sensoren-Paar ab, was aber zu Komplikationen führt, da man halt nur vier Sharps anschließen kann oder man nimmt die lichtempfindlichen Infrarot Sensoren, um diesen Bereich abzudecken. Da besteht dann aber das Problem, das noch mehr Sensordaten ausgewertet werden muss und diese Art von Sensoren sehr lichtempfindlich ist und es dadurch zu Fehlverhalten der Roboter führen kann.

Einauge

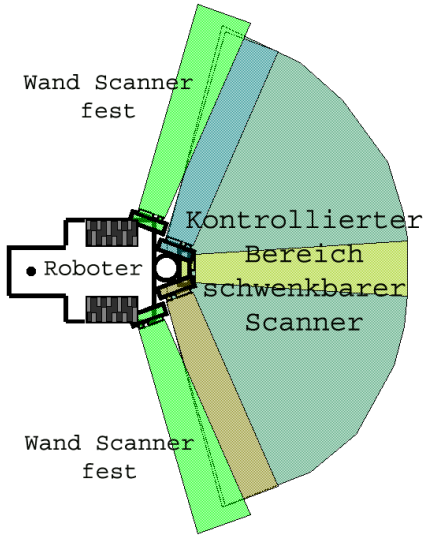


Abbildung 3.3: Designanordung des Einauges

Die zweite und etwas sinnvollere Möglichkeit besteht darin, dass man ein Sharp-Sensoren-Paar drehbar vorne in der Mitte des Roboters anbringt. Das heißt, durch einen Servo werden die Sensoren schwenkbar sein und so kann der Roboter einen viel größeren Bereich nur mit zwei Sharp-Sensoren überwachen. Sie Abbildung 3.3: Einauge und Prototyp im Anhang C. Die beiden übrig gebliebenen Sharp-Sensoren werden als Abstandsmesser jeweils, wie oben schon erwähnt, in einem 45Grad Winkel, nach außen hin zeigend, vorne am Roboter fest installiert. Diese Sensoren werden nur den

Abstand von der Wand und eventuelle Hindernisse kontrollieren.
Wenn nun das drehbare Sensoren-Paar ein Ball erkennt, dann richten sich die Sensoren auf den Ball aus und der Roboter dreht sich soweit, bis der Ball direkt vor dem Roboter liegt.
Auch diese Art Konstruktion hat ein entscheidendes Problem. Da die Drehung der Sensoren zu lange dauert und der Roboter in der Bewegung ist, kann es passieren, dass er einfach an dem Ball vorbei fährt, weil gerade das Sensoren-Paar in die andere Richtung schaut.
Der Roboter muss dann so programmiert werden, dass er immer wieder nur ein Stück vorfährt und dann nach dem Ball scannt. Dies würde aber zu lange dauern und den Spielfluss arg stören.

Zweiauge

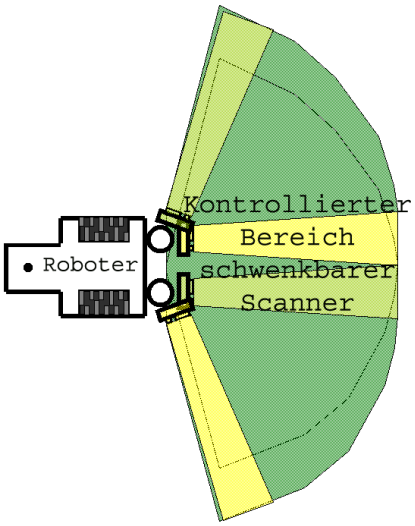


Abbildung 3.4: Designanordung des Einauges

Eine weitere Möglichkeit besteht darin zwei Sensoren-Paare drehbar vorne zu platzieren. Sie Abbildung 3.4: Zweiauge und Prototyp im Anhang C. Damit die Sensoren-Paare sich nicht gegenseitig stören, halten sie ein gewissen Abstand zu einander. Jeweils einer der Sensoren-Paare ist nach vorne ausgerichtet, wobei das zweite Sensoren-Paar in einem Winkel von ca. 45 Grad nach außen zeigt. Damit hat man die Kontrolle nach vorne und auch zur Seite. Durch die Drehung der Sensoren-Paare kann der Roboter nun ein viel größeren Bereich gleichzeitig kontrollieren, als mit der Konstruktion des Einauges (siehe Abbildung 3.3: Einauge). Wenn nun einer der Sensoren-Paare einen Ball erkennt, dann richtet sich der Roboter, wie auch schon bei der Konstruktion des Einauges beschrieben, auf den Ball aus und der Roboter dreht sich solange,

bis das Sensoren-Paar, der den Ball unter Kontrolle hat, nach vorne zeit. Dabei kann der Roboter mit dem anderen Sensor-Paar ein großen Teil seiner Umgebung weiterhin kontrollieren und gegebenenfalls Hindernisse, wie Wände oder einen anderen Roboter ausweichen.
Die Chance den Ball zu erkennen ist bei diesem Aufbau drastisch erhöht worden und der Prototyp hat im Feldversuchen gute arbeit geleistet.
Ein Nachteil an dieser Konstruktion ist, dass man eine neue Taktik programmieren muss (siehe Kapitel 4: Programmierung der Roboter), denn bei den anderen beiden Konstruktion lag die Taktik darin, dass der Roboter mit einem angemessenen Abstand sich an der Wand lang orientiert. Da nun die Sensoren-Paare nicht ständig auf die Wand ausgerichtet sind, muss diese art Taktik überarbeitet werden. Weiterhin kann es immer noch passieren, dass der Roboter an einem Ball einfach dran vorbei fährt, da die Sensoren nicht ständig den ganzen Bereich kontrollieren können.

3.2.2 Kollisionserkennung

Beide Roboter müssen eine Art Kollisionserkennung haben, damit sie bei einem unvermeidbaren Zusammenstoss mit der Wand oder mit dem Gegner regelkonform reagieren können. Hierbei kommen die Bumper ins Spiel, die sinnvoll am Roboter installiert werden sollten. Weiterhin sollte der Roboter so verkleidet sein, dass der andere Roboter ihn als Wand identifiziert.

3.2.3 Torerkennung

Die Torerkennung kann der Roboter mit dem schon oben erwähnten Beacon - Sensor durchführen. Dieser Sensor wird vorne am Roboter, direkt nach vorne ausgerichtet, oberhalb der Sharp-Sensoren fest angebracht.

3.2.4 Ballschussvorrichtung

Die Schussvorrichtung ist noch mal ein sehr schwieriger Teil der Konstruktion. Da der Ball durch einen Magneten greifbar ist, ist es zwar nicht so schwer, ihn am Roboter zu fixieren, jedoch den Ball in seine Vorrichtung zu bekommen. Bei der ersten Möglichkeit der Anordnung der Sharp-Sensoren (siehe Abbildung 3.2: Starrauge. Zwei Sensoren-Paare fixiert in einem 45Grad Winkel nach außen zeigend) ist es einfach den Ball mit einer Trichtervorrichtung in die Schussvorrichtung zu führen. Doch bei

der zweiten Möglichkeit (siehe Abbildung 3.3: Einauge. Ein Sensor-Paar schwenkbar vorn am Roboters) ist es schwieriger durch eine Konstruktion den Ball rechts oder links an den Sensoren-Paar vorbei zu führen, denn da die Sensoren-Paar immer den Ball fixieren, würde der Roboter es nie zulassen, dass der Ball in die Schussvorrichtung gelangt. Dieses Problem muss softwaremäßig gelöst werden.
Bei der dritten Konstruktionsmöglichkeit gilt das gleiche Prinzip wie bei der ersten Konstruktion. Da die beiden Sensoren-Paare weit genug auseinander sind, ist genügend Platz den Ball in die Mitte durch die Trichtervorrichtung zu führen und an den Roboter zu fixieren.
Die eigentliche Schussvorrichtung ist vergleichbar mit einem Katapult, welches mit einem Motor angetrieben wird. Zur Fixierung des Balles wird auf dessen Höhe ein Magnet an der Vorrichtung angebracht.

3.2.5 Gerüst des Roboters

Beim Prototyp, so wie dem fertigen Roboter muss darauf geachtet werden, dass der Roboter genügend Platz aufweist, um eine gewisse Anzahl an Akkus transportieren zu können. Weiterhin sollte das MIT Board auf dem Roboter so platziert werden, so dass das Board und seine Tasten leicht erreichbar sind. Es ist also auf jeden Fall sinnvoll das Board so weit wie möglich nach oben auf dem Roboter zu installieren. Man sollte auch darauf achten den Roboter so Kompakt wie möglich zu konstruieren. Also ohne Ecken und Kanten, damit es zu keinen Verklemmungen der Roboter führt.

Kapitel 4

Programmierung der Roboter

Wie schon im „Kapitel 2.3 Softwareanteile“beschrieben, besitzt Interactive C nur wenige Kontrollstrukturen (*for, while, if-else* und *break*). Deshalb kann man auch nicht bei solchen primitiven Robotern verlangen, dass die Roboter solch Intelligenz aufweisen, wie die Roboter, die ihre eigenen Server im Hintergrund haben und ihre Handlungen mit jedem Mitspieler absprechen und koordinieren. Zur Zeit ist das MIT 6.270 Board auch nicht in der Lage, einen Programmcode mit mehreren tausend Zeilen und Prozessen zu verwalten. Vielleicht wird es später möglich sein, dass das MIT 6.270 Board nur einfache Befehle abarbeitet und die eigentliche Intelligenz auf einem Rechner bzw. PDA abläuft.
Doch in dieser Studienarbeit muss ich mich mit dem begrenzten Speicher des MIT 6.270 Board zufrieden geben. Das heißt also, dass man das Programm so weit es geht simpel halten soll, denn je komplizierter man denkt, desto umfangreicher wird das Programm und desto langsamer der Roboter bei der Reaktion.
In den nächsten Abschnitten werde ich die wesendlichen Programmteile vorstellen.

4.1 Programmteil: Scanner

Das wichtigste und auch komplizierteste an der Programmierung des Roboters ist die Erkennung des Balles mit Hilfe der Sensoren. Bei den drei Designkonstruktionen des Roboters (siehe: Starrauge, Einauge und Zweiauge) hab ich unterschiedliche Programmierungen verwendet, um den Ball zu erkennen. Das Einzige, was in jeder Konstruktionen beibehalten wurde, ist, wie der Roboter ein Ball, die Wände oder Hindernisse interpretiert. Dabei werden alle Sensorwerte mit den folgenden Befehlen ermittelt:

```
unten_Sensor = analog(sensorunten);
obenSensor   = analog(sensoroben);
```

Bei der Variablen *„sensorUnten“* rechts neben dem Befehl *„analog“* handelt es sich um einen Integer-Wert, den man im *„Defines“* festgelegt kann. Hierbei handelt es sich um die Anschlussnummer der Sensoren auf dem Board. Der Sensor liest Werte zwischen 25 und 176 aus und speichert den Wert in der Varibalen (*„untenSensor“*) ab. Um nun die Werte der Sensoren zu interpretieren wird folgender Befehl benutzt:

```
if((obenSensor < (untenSensor - kalibrierung)) && (untenSensor > ballerkennung)) rBER = 1;
else                                                                             rBER = 0;
```

Bei der Variablen *„ballerkennung“* handelt es sich um ein Integer-Wert, der festlegt, wann man sicher sein möchte, dass ein Ball erkannt wurde. Die Werte der Sensoren werden nun als „1“oder „0“interpretiert. Bei „1“wurde ein Ball erkannt und bei „0“nicht. Es ist weiterhin sinnvoll ein *„FLAG“* zu bestimmen, ob der Ball zu nah am Roboter ist. Dies geschieht mit dem Befehl:

```
if(untenSensor > zunah)   rBZN = 1;
else                      rBZN = 0;
```

Um eine Wand eindeutig zu interpretieren kann man den folgenden Programmteil benutzt:

```
if(rechtsWand > minentfernt)
{
  wrechts  = 1;
  wmrechts = 1;
}
else if ((rechtsWand > maxentfernt) && (wmrechts == 1))
      {
          wrechts = 1;
      }
      else
      {
          wrechts  = 0;
          wmrechts = 0;
      }
```

Bei diesen *„if-else“* Abfragen, wird die Wand erst dann eindeutig als Wand erkannt, wenn der Roboter eine gewisse Entfernung erreicht hat. Danach kann der Roboter sich etwas weiter von der Wand entfernen und wird trotzdem noch als Wand interpretiert. Auch bei der Wanderkennung ist es Hilfreich ein *„FLAG“* zu setzten, damit der Roboter erkennt, wenn er zu nah an der Wand ist.

```
if(rechtsWand > zunah)  WRF = 1;
else                    WRF = 0;
```

Wenn man nun die Sensor-Werte interpretiert hat, kommt es jetzt auf die Konstruktion an, wie man die Ergebnisse weiter verwendet.

Starrauge

Da bei dieser Konstruktion die Sensoren fest am Roboter installiert sind, ist es hier nicht notwendig, dass die Scanner ihren eigenen Task erhalten, denn die Werte der Scanner werden durch jeden Schleifendurchgang neu bestimmt und sofort verarbeitet.

Einauge

Hier ist es auf jeden Fall sinnvoll diesen Programmteil sein eigenen Task zu spendieren, da der Scanner unabhängig vom Antriebselement agieren muss, denn da der Scanner ständig in Bewegung ist und bei Erkennung des Balles diesen fixieren muss. Die Task übergibt dem Hauptprogramm nur die Position des Scanners, damit der Roboter sich auf den Scanner ausrichten kann.

```
while(1)
{
  /* Werte der einzelnen Sensoren erfassen und an eine Varibale übergeben */
  ...

  /* Ball erkennung: siehe oben, Flags setzten, bestehen aus 0 und 1 */
  ...

  /* Stellung des Scanners: gößer Mitte, kleiner Mitte, gleich Mitte */
  ...

  /* Scanner sucht */
  if(Ball nicht zu sehen oder noch nicht gesehen)
  {
     /* Scanner schwenkt im Bereich hin und her */
  }

  else if(Ball nicht zu sehen, aber schon gesehen)
  {
     /* Scanner versucht den Ball an der alten Position weider zu finden */
  }

  else if(Scanner hat Ball gefunden)
  {
   /* Flags setzten, das der Scanner den Ball entdeckt hat */
  }

  else if(Scanner hat Ball gefunden und ihn auch schon vorher gesehen)
  {
   /* Festlegen, wie weit der Scanner schwenken kann, damit der Ball unter kontrolle bleibt */
  }

  else if(Scanner hat Ball gefunden und hat ihn unter kontrolle)
  {
     /* den Scanner nur so weit schwenken, dass der Ball noch unter kontrolle ist */
  }
}
```

Diese Schleife behandelt alle Möglichkeiten mit welchen der Scanner in Berührung kommen kann. Darunter fallen die Fragen: Was soll der Scanner machen, wenn er kein Ball sieht? Was, wenn er ein Ball sieht? Was, wenn er den Ball wieder verloren hat? Die wichtigen Informationen werden als Globalen-Variablen dem Hauptprogramm zu Verfügung gestellt. Den vollständigen Programm-Code ist im Anhang zu finden.

Zweiauge

Im Grunde ändert sich der Scanner-Task bei dieser Art Konstruktion vom Einauge nicht. Der Unterschied besteht nur darin, dass man beide Sensor-Paare berücksichtigen muss.

4.2 Programmteil: Bewegung

In dieser Datei sollten nur Befehle für die Motorsteuerung und eventuell bei der Konstruktion von Starrauge oder Einauge den Programmteil *„Wallfollow“* enthalten. Beim *„Wallfollow“* handelt es sich um eine Tatik den Roboter an der Wand lang fahren zu lassen. Der Programmteil sieht wie folgt aus:

```
void wallFollowRechts(int entfernung)
{
   int sollWert;

   if(entfernung == normal)   sollWert = soll;
   else if(entfernung == nah) sollWert = minsoll;

   radWand       = motRechts;
   radKeineWand  = motLinks;
   clip          = clipWert;

   ist = analog(wandRechts);

   diff = sollWert-ist;
   if(diff < 0)
   diff = diff *(-1);

   if(diff < 5)
   {
      motor(motLinks,stufe7);
      motor(motRechts,stufe7);
   }
   else
   {
      /* zu weit weg*/
      if(sollWert >= ist)
      {
         speed = stufe7 - ((sollWert-ist)*faktor);
         if(speed < 50)    {speed = clip;}
         motor(radKeineWand,stufe7);
         motor(radWand,speed);
```

```
      }

      /*zu nah dran*/
      if(sollWert < ist)
      {
         speed = stufe7 - ((ist-sollWert)*faktor);
         if(speed < 50) {speed = clip;}
         motor(radWand,stufe7);
         motor(radKeineWand,speed);
      }
   }
}
```

Der Roboter versucht einen vordefinierten Abstand von der Wand bei zu behalten. Wenn der Roboter zu nah oder zu weit weg von der Wand kommt, dann wird der Abstand von der Wand dementsprechend korrigiert.

4.3 Programmteil: Status

In diesem Programmteil steckt die eigentliche „Intelligenz“des Roboters. Es ist ein *„State-Automat“*, der verschiedene Situationen berücksichtigt und dem Hauptprogramm mitteilt, wie der Roboter reagieren soll.

```
/* State - Automat */
int status(int last_state)
{
   int state;

   /* Ball Aktionen */
   if('1.Situation')
   {
       state = ...;
   }

   else if('2.Situation')
   {
       state = ...;
   }

  .
  .
  .

   /* Ball Aktion und Wand */
   else if('3.Situation')
   {
      state = ...;
   }

  .
  .
  .

  return state;
}
```

Kapitel 5

Zusammenfassung und Ausblick

In dieser Studienarbeit wurde gezeigt, dass man auch mit kostengünstigen Materialien ein brauchbares Robotfußballspiel aufbauen kann. Dabei wurde insbesondere auf die verschiedene Design Möglichkeiten bei der Konstruktion der Roboter und dessen Vor- und Nachteile eingegangen.
Jedoch wurde am Anfang der Studienarbeit auf die zu verwendenden und bereitgestellten Materialien eingegangen: Was für Materialien zur Verfügung steht und wofür sie gebraucht werden.
Anschließend wurde diskutiert, wie die Spielumgebung aussehen sollte: Wie groß die Spielfläche sein sollte, ob eine Umrandung der Spielfläche sinnvoll ist und wie die Tore zu platzieren sind.
Im Hauptteil der Studienarbeit (Das Design der Roboter) wurden drei verschiedene Möglichkeiten vorgestellt und diskutiert, wie der Roboter den Ball am besten erkennen kann. Zum einen war da die Konstruktion des Starrauges zu erwähnen, die einige Nachteile bei der Erkennung des Balles hat, wenn der Ball direkt vor dem Roboter auftaucht. Bei der zweiten Konstruktion (Einauge) war da das Problem, dass der Scanner zu langsam war, um den ganzen Bereich bei der Bewegung des Roboters abzudecken. Es kam dazu, dass der Roboter einfach an dem Ball vorbei fuhr, weil der Scanner gerade auf der anderen Seite nach dem Ball suchte. Die wirklich bessere Lösung war die Konstruktion des Zweiauges. Hier überblicke der Roboter in einem schnelleren Zeitraum den ganzen Scanner-Bereich und konnte so den Ball oder ein Hindernis besser erkennen und kontrollieren.
Zum Schluss beschäftigte sich die Studienarbeit mit der Programmierung der Roboter. Hier wurden auch die drei unterschiedlichen Konstruktionen der Roboter berücksichtig. Im großen und ganzen wurden die Ziele erfolgreich erfüllt und die Erwartungen übertroffen.
Es gibt sicherlich noch viel zu verbessern, vor allem im Bereich der Programmierung. Wie schon im Kapitel 4 (Programmierung der Roboter) angedeutet, könnte man später in der Lage sein, das eigentliche Programm auf einem PDA oder Rechner laufen zu

lassen, um das MIT-Board zu entlasten. Mit Hilfe des Bluetooth ist man auch in der Lage die Recheneinheit und den Roboter drahtlos zu verbinden. Bei weiterer Investition kann man auch die Ballerkennung verbessern, indem man noch mehr Sensoren am Roboter anbringt. Doch das würde das MIT-Board ohne eine weitere Recheneinheit überfordern und die eigentliche Zielsetzung dieser Studienarbeit nicht mehr erfüllen.

Literaturverzeichnis

[RoboCup] ***RoboCup***
URL: www.robocup.org

[RoboCupJr] ***RoboCup Junior***
URL: http://212.75.35.5/pjk/robolab/robocup/

[FIRA] ***Federation of International Robot-soccer Association***
URL: www.ihrt.tuwien.ac.at/FIRAEC/
www.FIRA.net

[Martin] ***Martin, Fred: 6.270: The MIT LEGO Robot Design Competition***
URL: www.media.mit.edu/people/fredm /projects/6270

[MIT6.270] ***6.270 - MIT's Autonomous Robot Design Competition***
URL: web.edu/6.270

[SHARP] ***Information zum Sharp-Sensor Typ GP2D12***
URL: www.acroname.com/robotics/parts/R48-IR12.html
www.elektronik-projekt.de/dominik/sensor.htm

[NEWTON] ***Newton Research Labs***
URL: www.newtonlabs.com/ic

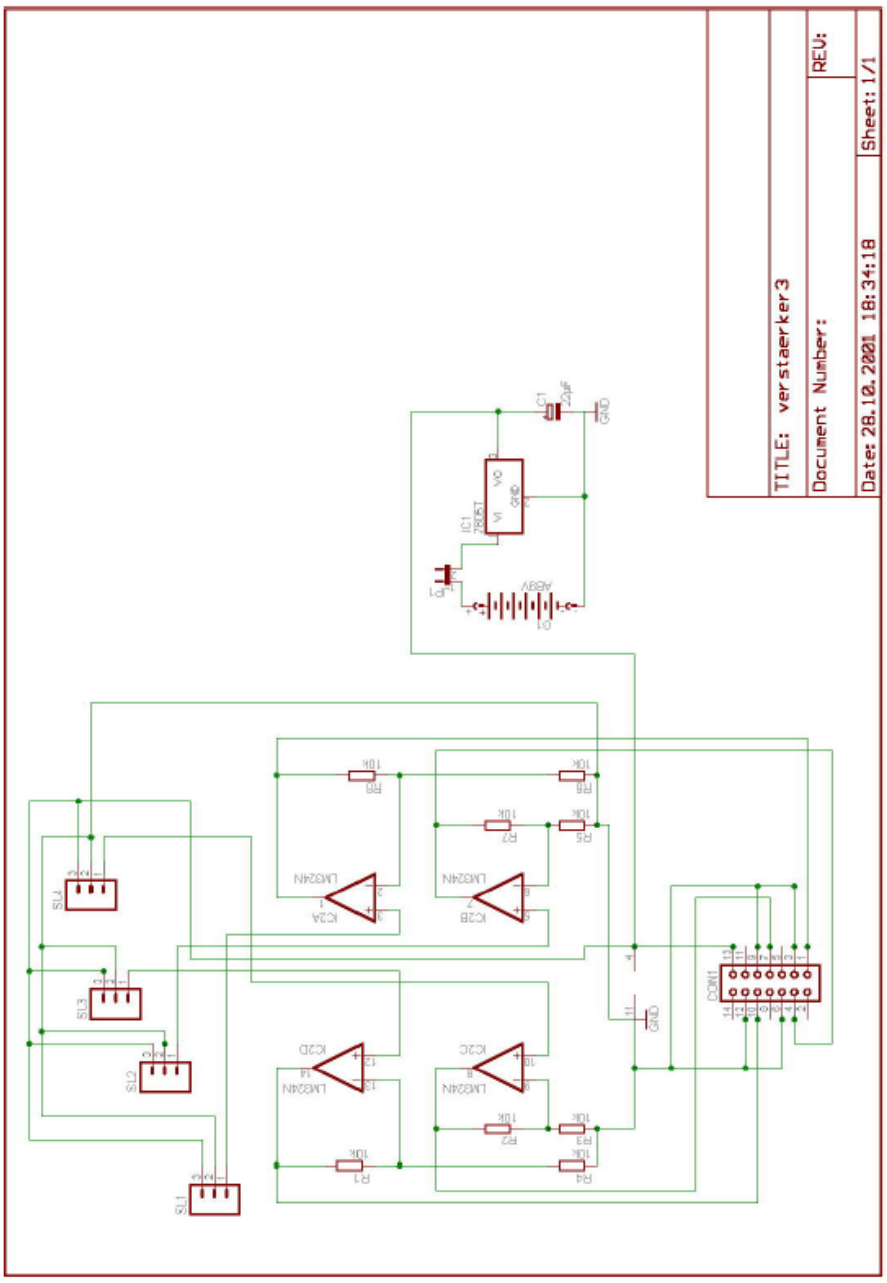
[HAWLego] ***Lego***
URL: www.informatik.haw-hamburg.de/ lego

Anhang A

Schaltplan Verstärker

Der Verstärker für die Sharp Sensoren beruht auf einer Schaltung die im Roboter Labor erstmals von Rainer Balzerowski gebaut wurde. Informationen zu der Schaltung gibt es auf den Seiten der HAW Hamburg [HAWLego]. Dort gibt es eine Anleitung, für den Anschluss von 3 Sharpsensoren an einen Eingang eines RCX der Lego Mindstorms.

Der hier verwendete Verstärker ist eine Erweiterung von Gunther Lemm auf vier Eingänge. Das Layout der Platine wurde von Timo Storjohann mit dem Programm Eagle erstellt. Die Platinen wurden zum Ätzen gegeben. Dadurch sind Platinen entstanden, die angenehm zu löten sind.


TITLE: verstaerker3
Document Number:
REV:
Date: 28.10.2001 18:34:18
Sheet: 1/1
LM324N
GND

Anhang B

Datenblatt Sharp Sensor GP2D12

GP2D12/GP2D15

General Purpose Type Distance Measuring Sensors

■ Features

1. Less influence on the color of reflective objects, reflectivity
2. Line-up of distance output/distance judgement type
   Distance output type (analog voltage) : **GP2D12**
   Detecting distance : 10 to 80cm
   Distance judgement type : **GP2D15**
   Judgement distance : 24cm
   (Adjustable within the range of 10 to 80cm)
3. External control circuit is unnecessary
4. Low cost

■ Applications

1. TVs
2. Personal computers
3. Cars
4. Copiers

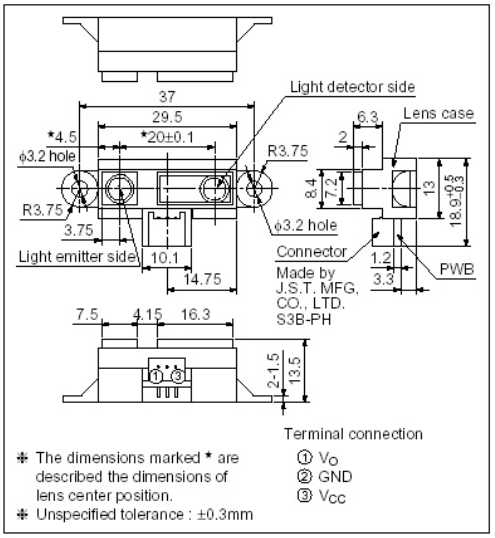
■ Absolute Maximum Ratings

(Ta=25°C, Vcc=5V)

Parameter	Symbol	Rating	Unit
Supply voltage	Vcc	−0.3 to +7	V
Output terminal voltage	Vo	−0.3 to Vcc +0.3	V
Operating temperature	T_{opr}	−10 to +60	°C
Storage temperature	T_{stg}	−40 to +70	°C

■ Outline Dimensions

(Unit : mm)



* The dimensions marked * are described the dimensions of lens center position.
* Unspecified tolerance : ±0.3mm

■ Recommended Operating Conditions

Parameter	Symbol	Rating	Unit
Operating supply voltage	V_{CC}	4.5 to +5.5	V

■ Electro-optical Characteristics

(Ta=25°C, V_{CC}=5V)

Parameter		Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Distance measuring range		ΔL	*1 *2	10	–	80	cm
Output terminal voltage	GP2D12	V_O	L=80cm *1	0.25	0.4	0.55	V
	GP2D15	V_{OH}	Output voltage at High *1	Vcc –0.3	–	–	V
		V_{OL}	Output voltage at Low *1	–	–	0.6	V
Difference of output voltage	GP2D12	ΔV_O	Output change at L=80cm to 10cm *1	1.75	2.0	2.25	V
Distance detectable limit of output	GP2D15	V_O	*1 *2 *3	21	24	27	cm
Average Dissipation current		I_{CC}	L=80cm *1	–	33	50	mA

Note) L : Distance to reflective object.

*1 Using reflective object : White paper (Made by Kodak Co. Ltd. gray cards R-27 - white face, reflective ratio : 90%).
*2 We ship the device after the following adjustment : Output switching distance L=24cm±3cm must be maintained by the sensor.
*3 Distance measuring range of the optical sensor system.
*4 Output switching has a hysteresis width. The distance specified by Vo should be the one with which the output L switches to the output H.

Fig.1 Internal Block Diagram

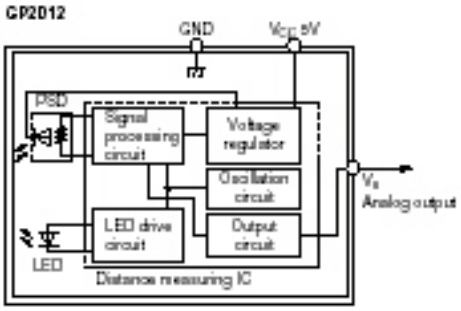


Fig.2 Internal Block Diagram

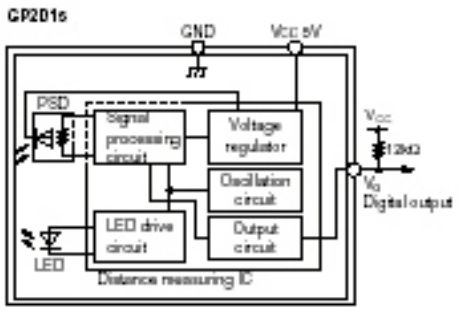


Fig.3 Timing Chart

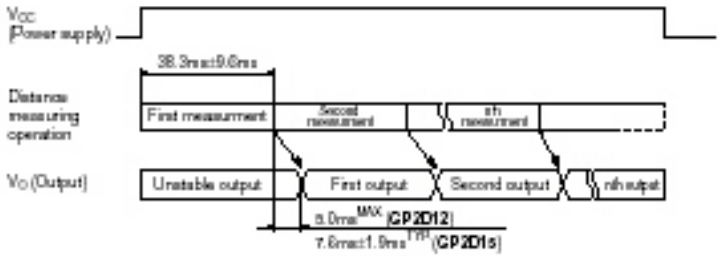


Fig.4 Distance Characteristics

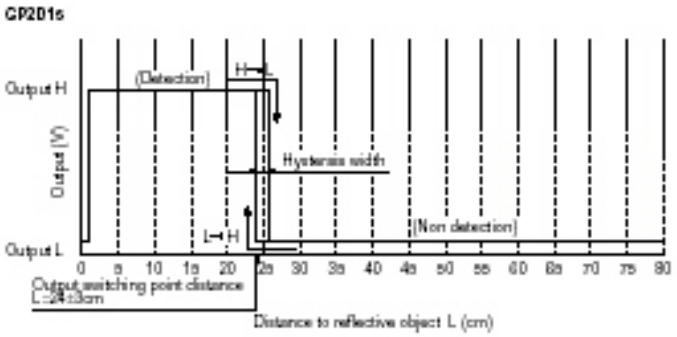


Fig.5 Analog Output Voltage vs. Surface Illuminance of Reflective Object

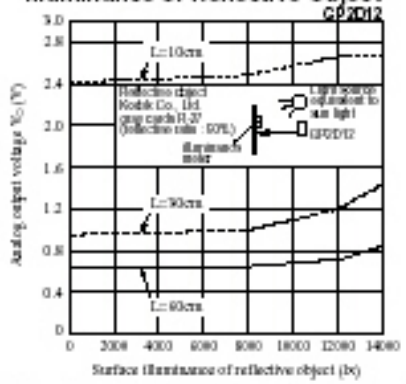


Fig.6 Analog Output Voltage vs.Distance to Reflective Object

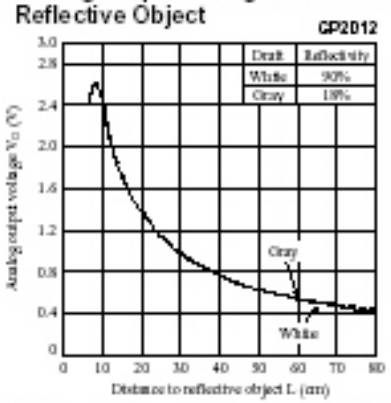


Fig.7 Analog Output Voltage vs.Ambient Temperature

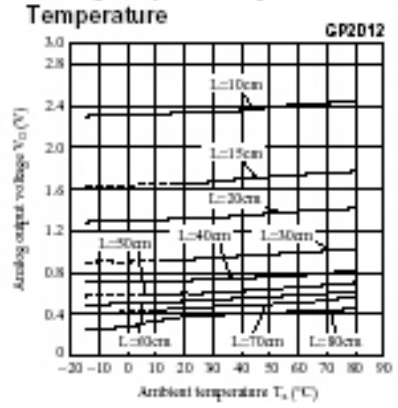
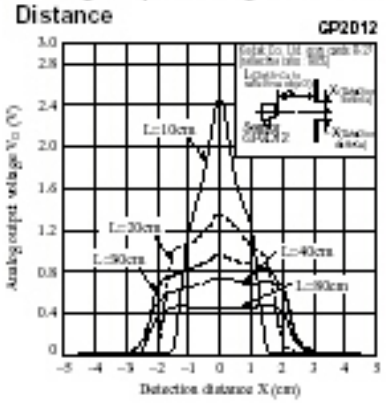


Fig.8 Analog Output Voltage vs.Detection Distance



NOTICE

- The circuit application examples in this publication are provided to explain representative applications of SHARP devices and are not intended to guarantee any circuit design or license any intellectual property rights. SHARP takes no responsibility for any problems related to any intellectual property right of a third party resulting from the use of SHARP's devices.
- Contact SHARP in order to obtain the latest device specification sheets before using any SHARP device. SHARP reserves the right to make changes in the specifications, characteristics, data, materials, structure, and other contents described herein at any time without notice in order to improve design or reliability. Manufacturing locations are also subject to change without notice.
- Observe the following points when using any devices in this publication. SHARP takes no responsibility for damage caused by improper use of the devices which does not meet the conditions and absolute maximum ratings to be used specified in the relevant specification sheet nor meet the following conditions:

(i) The devices in this publication are designed for use in general electronic equipment designs such as:
- Personal computers
- Office automation equipment
- Telecommunication equipment [terminal]
- Test and measurement equipment
- Industrial control
- Audio visual equipment
- Consumer electronics

(ii) Measures such as fail-safe function and redundant design should be taken to ensure reliability and safety when SHARP devices are used for or in connection with equipment that requires higher reliability such as:
- Transportation control and safety equipment (i.e., aircraft, trains, automobiles, etc.)
- Traffic signals
- Gas leakage sensor breakers
- Alarm equipment
- Various safety devices, etc.

(iii) SHARP devices shall not be used for or in connection with equipment that requires an extremely high level of reliability and safety such as:
- Space applications
- Telecommunication equipment [trunk lines]
- Nuclear power control equipment
- Medical and other life support equipment (e.g., scuba).

- Contact a SHARP representative in advance when intending to use SHARP devices for any "specific" applications other than those recommended by SHARP or when it is unclear which category mentioned above controls the intended use.
- If the SHARP devices listed in this publication fall within the scope of strategic products described in the Foreign Exchange and Foreign Trade Control Law of Japan, it is necessary to obtain approval to export such SHARP devices.
- This publication is the proprietary product of SHARP and is copyrighted, with all rights reserved. Under the copyright laws, no part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, in whole or in part, without the express written permission of SHARP. Express written permission is also required before any use of this publication may be made by a third party.
- Contact and consult with a SHARP representative if there are any questions about the contents of this publication.

Anhang C

Prototypen der Roboter

In diesem Anhang sehen sie unterschiedliche Prototypen die ich im Laufe der Studienarbeit konstruiert und getestet hab.

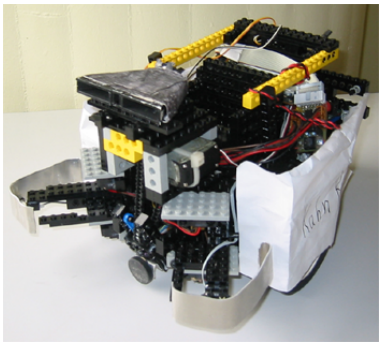


Abbildung C.1: Starrauge

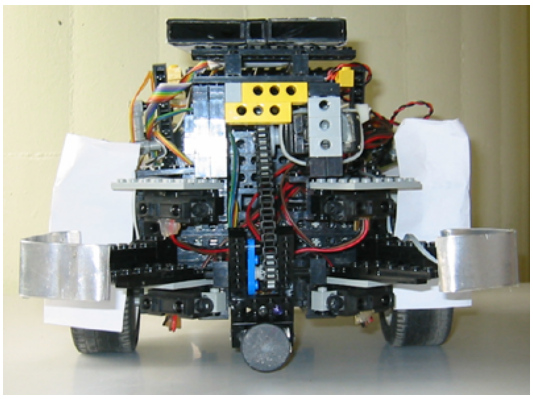


Abbildung C.2: Starrauge

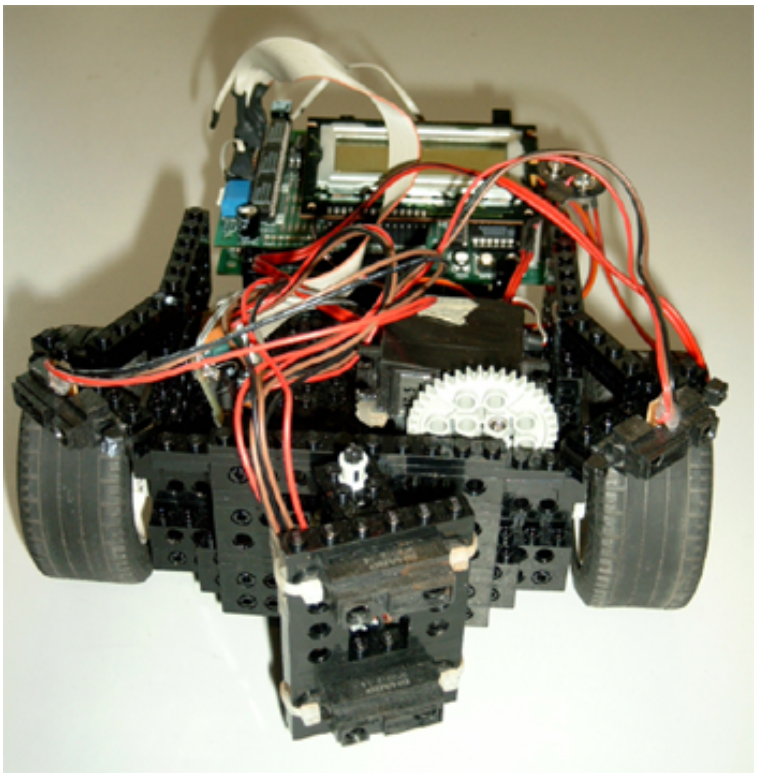


Abbildung C.3: Einauge

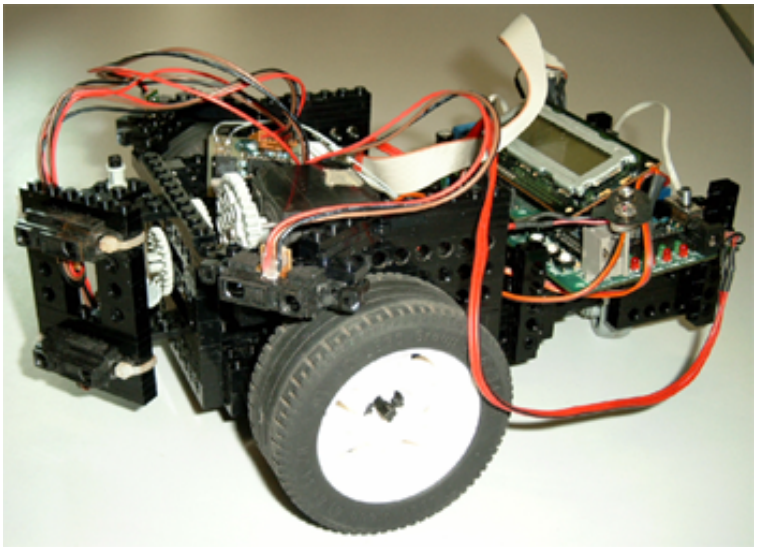


Abbildung C.4: Einauge

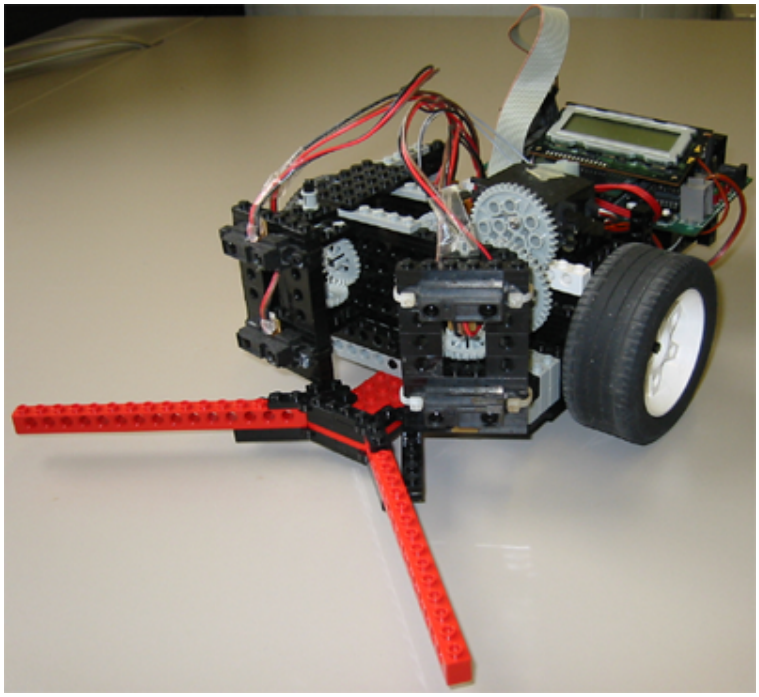


Abbildung C.5: Zweiauge

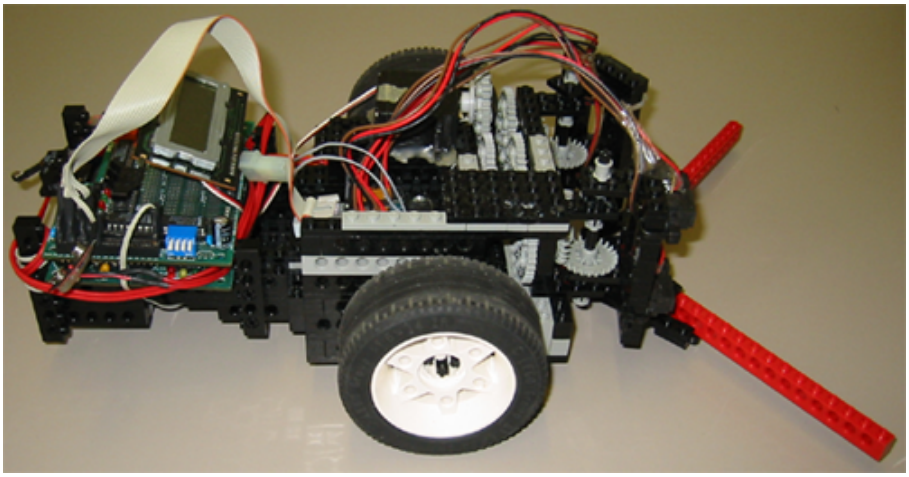


Abbildung C.6: Zweiauge

Anhang D

Vollständige Programmteile

In diesem Anhang sehen sie im ersten Bereich die Programme für den Roboter Starrauge. Anschließend werden die wesentlichen Programmteile von Einauge und Zweiauge abgebildet.

Defines.c

```
/* Globale Variablen */
int w_links, w_rechts, b_links, b_rechts, vorne, goalL, goalM, goalR, goalB,
    BRF, BLF, WRF, WLF, VOK, hv_l_bump, hv_r_bump, hh_bump, hgoalL, hgoalR, hgoalM, hgoalB;

/* fuer Sensor - Werte */
#define vorne_max_entfernt   150
#define schuss_pos           15
#define zu_nah               130

/* Servo */
#define schuss               60.0
#define null_pos             15.0
#define schuss_vorbereiten   0.0
#define kontakt_zeit         2500

/* Wall-Follow */
#define soll       50        /* so weit von der Wand weg bleiben - Wall Follow */
#define min_soll   110       /* naeher an die Wand fahren - Wall Follow */
#define faktor     1
#define clip_Wert  25

/* Seitenwand entfernung */
#define maxentfernt          40
#define minentfernt          50
#define ballerkennung        50

/* Allgemeine defines */
#define normal      0    /* Weitergabe Variablen von main.c nach bewegen.c */
#define nah         1    /* Weitergabe Variablen von main.c nach bewegen.c */
#define nach_rechts 2    /* Weitergabe Variablen von status.c(status) nach status.c(beacon_entscheidung()) */
#define nach_links  3    /* Weitergabe Variablen von status.c(status) nach status.c(beacon_entscheidung()) */
```

```
/* Goal and Home */
#define goal        125

/* Motorsteuerung */
#define mot_links   1
#define mot_rechts  0
#define stufe7      100
#define stufe6      90
#define stufe5      70
#define stufe4      60
#define stufe3      50
#define stufe2      30
#define stufe1      20
#define stufe0      0

/* Schalter */
#define vorne_links_bump    digital(3)
#define vorne_rechts_bump   digital(2)
#define hinten_bump         digital(1)
#define aus_Schalter        analog(27)

/* Beacon */
#define beacon_hinten     4 /* nach hinten scannen */
#define beacon_links      5
#define beacon_mitte      6
#define beacon_rechts 7
#define impmin            1 /* Wert, wie genau die Beacon ist */

/* Sensoren */
#define wand_links        20
#define ball_links        21
#define wand_rechts       19
#define ball_rechts       18
#define sensor_vorne      14
```

Main.c

```
void main(void)
{
    int PIDC, PIDB, state;

    PIDC = start_process(schuss_kontrolle());
    PIDB = start_process(bump());

    led_out0(1);
    led_out1(1);

    servo_on();
    servo_deg(15.0);

    motor(mot_links,stufe5);
    motor(mot_rechts,stufe5);
    motor(5,100);

    state = 0;
    WRF = 0;
    WLF = 0;
    BRF = 0;
    BLF = 0;
    VOK = 0;
    set_ir_receive_frequency(goal);
    ir_receive_on();
```

```
    while(aus_Schalter > 10)
    {
        /* Fussballspiel start */
        state = status(state);

/* printf("\nG:%d%d%d %d", goalL, goalM, goalR, goalB);
*/
   printf("\nW:%d%d B:%d%d %d S:%d  %d  %d%d%d %d", w_rechts, w_links, b_rechts,
                         b_links, vorne, state, VOK, goalL, goalM, goalR, goalB);

        /* state 0 : Stop - undefinierte Situation */
        if(state == 0)        stop();

        /* fahre vorwaerts */
        else if(state == 1)   vorwaerts();

        /* fahre langsam vorwaerts */
        else if(state == 2)   langsam();

        /* zurueck fahren */
        else if(state == 3)   zurueck();

        /* state 4 : Rechts drehen */
        else if(state == 4)   rechts_drehen();

        /* state 5 : Links drehen */
        else if(state == 5)   links_drehen();

        /* state 6 : Im Stand nach Rechts drehen - Ball zu nah */
        else if(state == 6)   rechts_stand_drehen();

        /* state 7 : Im Stand nach Links drehen - Ball zu nah */
        else if(state == 7)   links_stand_drehen();

        /* state 8 : Wall Follow Rechts */
        else if(state == 8)   wall_follow_rechts(normal);

        /* state 9 : Wall Follow Links */
        else if(state == 9)   wall_follow_links(normal);

        /* state = 10 : Wall Follow Rechts, aber naeher an der Rechten Wand */
        else if(state == 10)  wall_follow_rechts(nah);

        /* state 11 : Wall Follow Links, aber naeher an der Linken Wand */
        else if(state == 11)  wall_follow_links(nah);

        /* State 12 : Langsam im Stand nach rechts drehen */
        else if(state == 12)  langsam_rechts_stand_drehen();

        /* State 13 : Langsam im Stand nach links drehen */
        else if(state == 13)  langsam_links_stand_drehen();

        /* state 14 : sofort schiessen */
        else if(state == 14)  hab_ball_schiessen();
    }

    printf("\nende");
    motor(mot_links,stufe0);
    motor(mot_rechts,stufe0);

    kill_process(PIDB);
    kill_process(PIDC);
}
```

Status.c

```
/* Globale Variablen */
int v_l_bump, v_r_bump, h_bump;

void bump_Wert()
{
    v_l_bump = hv_l_bump;
    v_r_bump = hv_r_bump;
    h_bump   = hh_bump;
}


/* Sensoren Werte */
void sensor()
{
    int wm_links, wm_rechts, links_Wand, rechts_Wand, bm_links, bm_rechts, links_Ball, rechts_Ball;

    links_Wand   = analog(wand_links);    /* wand_links: siehe Define */
    rechts_Wand  = analog(wand_rechts);   /* wand_rechts: siehe Define */

    links_Ball   = analog(ball_links);    /* ball_links: siehe Define */
    rechts_Ball  = analog(ball_rechts);   /* ball_rechts: siehe Define */

    if(rechts_Ball > zu_nah)    BRF = 1;  /* FLAG: rechts_Ball ist zu na */
    else                        BRF = 0;  /* FLAG: rechts_Ball weit genug weg */

    if(links_Ball > zu_nah)     BLF = 1;  /* FLAG: links_Ball ist zu na */
    else                        BLF = 0;  /* FLAG: links_Ball weit genug weg */

    if(rechts_Wand > zu_nah)    WRF = 1;  /* FLAG: links_Wand ist zu na */
    else                        WRF = 0;  /* FLAG: links_Wand weit genug weg */

    if(links_Wand > zu_nah)     WLF = 1;  /* FLAG: links_Wand ist zu na */
    else                        WLF = 0;  /* FLAG: links_Wand weit genug weg */

    if(analog(sensor_vorne) < schuss_pos) VOK = 1;  /* FLAG: vorne ist der Ball am Robo */
    else                                  VOK = 0;  /* FLAG: vorne ist kein Ball am Robo */


    /* Boolean für vorderen Sensor */
    if(analog(sensor_vorne) < vorne_max_entfernt)  vorne = 1;
    else                                           vorne = 0;

    /* Wand erkennung */
    if(links_Wand > minentfernt)
    {
        w_links  = 1;
        wm_links = 1;
    }
    else
        if ((links_Wand > maxentfernt) && (wm_links == 1))
        {
            w_links = 1;
        }
        else
        {
            w_links  = 0;
            wm_links = 0;
        }



  if(rechts_Wand > minentfernt)
```

```
    {
        w_rechts  = 1;
        wm_rechts = 1;
    }
    else
        if ((rechts_Wand > maxentfernt) && (wm_rechts == 1))
        {
            w_rechts = 1;
    }
    else
        {
            w_rechts  = 0;
            wm_rechts = 0;
        }

    /* Ball erkennung */
    if((links_Wand < (links_Ball-20))&& (links_Ball > ballerkennung)) b_links = 1;
    else                                                             b_links = 0;

    if((rechts_Wand < (rechts_Ball-20))&& (rechts_Ball > ballerkennung)) b_rechts = 1;
    else                                                               b_rechts = 0;

    /* Beacon suchen */
    if(VOK == 1)
    {
        ir_receive_off();
        set_ir_receive_frequency(goal);
        ir_receive_on();
        sleep(0.3);
        if(ir_counts(beacon_links) > impmin)  goalL = 1;
        else                                  goalL = 0;
        if(ir_counts(beacon_mitte) > impmin)  goalM = 1;
        else                                  goalM = 0;
        if(ir_counts(beacon_rechts) > impmin) goalR = 1;
        else                                  goalR = 0;
        if(ir_counts(beacon_hinten) > impmin) goalB = 1;
        else                                  goalB = 0;
    }
}


/* Beacon Entscheidungen */
int beacon_entscheidung(int wohin)
{
    int stateB;

    /* kein Beacon zu sehen / das Tor ist hinter dem Robo */
    if(((goalL==0)&&(goalM==0)&&(goalR==0)&&(goalB==0))||
       ((goalL==0)&&(goalM==0)&&(goalR==0)&&(goalB==1)))
    {
        if(wohin == nach_rechts)
        {
            if((WRF==0)&&(WLF)&&(w_rechts==0)&&(w_links==0))         stateB = 4;
            else if((WRF==0)&&(WLF==0)&&(w_rechts==0)&&(w_links==1)) stateB = 4;
            else if((WRF==0)&&(WLF==0)&&(w_rechts==1)&&(w_links==0)) stateB = 6;
            else if((WRF==0)&&(WLF==0)&&(w_rechts==1)&&(w_links==1)) stateB = 6;
            else if((WRF==0)&&(WLF==1)&&(w_rechts==0)&&(w_links==1)) stateB = 4;
            else if((WRF==0)&&(WLF==1)&&(w_rechts==1)&&(w_links==1)) stateB = 6;
            else if((WRF==1)&&(WLF==0)&&(w_rechts==1)&&(w_links==0)) stateB = 5;
            else if((WRF==1)&&(WLF==0)&&(w_rechts==1)&&(w_links==1)) stateB = 7;
            else if((WRF==1)&&(WLF==1)&&(w_rechts==1)&&(w_links==1)) stateB = 3;
            else                                                     stateB = 6;
        }

        else if(wohin == nach_links)
```

```
        {
            if((WRF==0)&&(WLF)&&(w_rechts==0)&&(w_links==0))         stateB = 5;
            else if((WRF==0)&&(WLF==0)&&(w_rechts==0)&&(w_links==1)) stateB = 7;
            else if((WRF==0)&&(WLF==0)&&(w_rechts==1)&&(w_links==0)) stateB = 5;
            else if((WRF==0)&&(WLF==0)&&(w_rechts==1)&&(w_links==1)) stateB = 7;
            else if((WRF==0)&&(WLF==1)&&(w_rechts==0)&&(w_links==1)) stateB = 4;
            else if((WRF==0)&&(WLF==1)&&(w_rechts==1)&&(w_links==1)) stateB = 6;
            else if((WRF==1)&&(WLF==0)&&(w_rechts==1)&&(w_links==0)) stateB = 5;
            else if((WRF==1)&&(WLF==0)&&(w_rechts==1)&&(w_links==1)) stateB = 7;
            else if((WRF==1)&&(WLF==1)&&(w_rechts==1)&&(w_links==1)) stateB = 3;
            else                                                     stateB = 7;
        }
    }


    /* fahre nach rechts */
    else if((goalL==0)&&(goalM==0)&&(goalR==1)&&(goalB==0))
    {
        if(wohin == (nach_rechts||nach_links))
        {
            if((WRF==0)&&(WLF)&&(w_rechts==0)&&(w_links==0))         stateB = 12;
            else if((WRF==0)&&(WLF==0)&&(w_rechts==0)&&(w_links==1)) stateB = 12;
            else if((WRF==0)&&(WLF==0)&&(w_rechts==1)&&(w_links==0)) stateB = 1;
            else if((WRF==0)&&(WLF==0)&&(w_rechts==1)&&(w_links==1)) stateB = 2;
            else if((WRF==0)&&(WLF==1)&&(w_rechts==0)&&(w_links==1)) stateB = 12;
            else if((WRF==0)&&(WLF==1)&&(w_rechts==1)&&(w_links==1)) stateB = 12;
            else if((WRF==1)&&(WLF==0)&&(w_rechts==1)&&(w_links==0)) stateB = 5;
            else if((WRF==1)&&(WLF==0)&&(w_rechts==1)&&(w_links==1)) stateB = 12;
            else if((WRF==1)&&(WLF==1)&&(w_rechts==1)&&(w_links==1)) stateB = 3;
            else                                                     stateB = 12;
        }
    }

    /* fahre nach links */
    else if((goalL==1)&&(goalM==0)&&(goalR==0)&&(goalB==0))
    {
        if(wohin == (nach_rechts||nach_links))
        {
            if((WRF==0)&&(WLF)&&(w_rechts==0)&&(w_links==0))         stateB = 13;
            else if((WRF==0)&&(WLF==0)&&(w_rechts==0)&&(w_links==1)) stateB = 1;
            else if((WRF==0)&&(WLF==0)&&(w_rechts==1)&&(w_links==0)) stateB = 13;
            else if((WRF==0)&&(WLF==0)&&(w_rechts==1)&&(w_links==1)) stateB = 2;
            else if((WRF==0)&&(WLF==1)&&(w_rechts==0)&&(w_links==1)) stateB = 4;
            else if((WRF==0)&&(WLF==1)&&(w_rechts==1)&&(w_links==1)) stateB = 13;
            else if((WRF==1)&&(WLF==0)&&(w_rechts==1)&&(w_links==0)) stateB = 13;
            else if((WRF==1)&&(WLF==0)&&(w_rechts==1)&&(w_links==1)) stateB = 13;
            else if((WRF==1)&&(WLF==1)&&(w_rechts==1)&&(w_links==1)) stateB = 3;
            else                                                     stateB = 13;
        }
    }

    /* fahre vorwaerts */
    else if(((goalL==0)&&(goalM==1)&&(goalR==0))||
            ((goalL==0)&&(goalM==1)&&(goalR==1))||
            ((goalL==1)&&(goalM==0)&&(goalR==1))||
            ((goalL==1)&&(goalM==1)&&(goalR==0))||
            ((goalL==1)&&(goalM==1)&&(goalR==1)))                    stateB = 14;

    return stateB;
}

/* State - Automat */
int status(int last_state)
{
    int state;
```

```
sensor();
bump_Wert();

if((v_l_bump==0)&&(v_r_bump==0)&&(h_bump==0))    /* Bumper aktivier?? */
{
    if((w_links==0)&&(w_rechts==0)&&(vorne==0)&&(b_links==0)&&(b_rechts==0))
    {
        state = 1;    /* state 1 : vorwaerts fahren */
    }

    /* Ball Aktionen */
    else if((w_links==0)&&(w_rechts==0)&&(vorne==0)&&(b_links==0)&&(b_rechts==1))
    {
        state = 4;    /* state 4 : Rechts drehen */
    }

    else if((w_links==0)&&(w_rechts==0)&&(vorne==0)&&(b_links==1)&&(b_rechts==0))
    {
        state = 5;   /* state 5 : Links drehen */
    }

    /* Ball Aktion und Wand */
    else if(((w_links==0)&&(w_rechts==1)&&(vorne==0)&&(b_links==0)&&(b_rechts==1))||
            ((w_links==1)&&(w_rechts==0)&&(vorne==0)&&(b_links==0)&&(b_rechts==1)))
    {
        state = 6;   /* state 6 : Im Stand nach Rechts drehen - Ball zu nah */
    }

    else if(((w_links==0)&&(w_rechts==1)&&(vorne==0)&&(b_links==1)&&(b_rechts==0))||
            ((w_links==1)&&(w_rechts==0)&&(vorne==0)&&(b_links==1)&&(b_rechts==0)))
    {
        state = 7;   /* state 7 : Im Stand nach Links drehen - Ball zu nah */
    }

    else if((w_links==1)&&(w_rechts==1)&&(vorne==0)&&(b_links==0)&&(b_rechts==1))
    {
        if(((WRF==0)&&(WLF==0)) || ((WRF==0)&&(WLF==1)))   state = 4;  /* state 4 : Rechts drehen */
        else                                               state = 6;  /* state 6 : Im Stand nach Rechts drehen -
                                                                          Ball zu nah */
    }

    else if((w_links==1)&&(w_rechts==1)&&(vorne==0)&&(b_links==1)&&(b_rechts==0))
    {
        if(((WRF==0)&&(WLF==0)) ||((WRF==1)&&(WLF==0)))    state = 5;  /* state 5 : Links drehen */
        else                                               state = 7;  /* state 7 : Im Stand nach Links drehen -
                                                                          Ball zu nah */
    }


    /* Hindernis oder Ball */
    else if((w_links==0)&&(w_rechts==0)&&(vorne==1)&&(b_links==0)&&(b_rechts==0))
    {
        if(VOK==0)    state = 2;    /* langsam vorfahren */
        else          state = beacon_entscheidung(nach_rechts);
    }

    else if((w_links==0)&&(w_rechts==0)&&(vorne==1)&&(b_links==0)&&(b_rechts==1))
    {
        if(VOK==0)    state = 6;    /* state 6 : Im Stand nach Rechts drehen - Ball zu nah */
        else          state = beacon_entscheidung(nach_links);
    }

    else if((w_links==0)&&(w_rechts==0)&&(vorne==1)&&(b_links==1)&&(b_rechts==0))
    {
```

```
    if(VOK==0)    state = 7;    /* state 7 : Im Stand nach Links drehen - Ball zu nah */
    else          state = beacon_entscheidung(nach_rechts);
}


/* Hindernis oder Ball und Wand */
else if((w_links==0)&&(w_rechts==1)&&(vorne==1)&&(b_links==0)&&(b_rechts==0))
{
    if(VOK==0)                  state = 2;    /* langsam vorfahren */
    else if((VOK==0)&&(WRF==1)) state = 7;    /* state 7 : drehe im Stand nach Links, weil Hindernis zu nah */
    else if(VOK==1)             state = beacon_entscheidung(nach_links);
}

else if((w_links==1)&&(w_rechts==0)&&(vorne==1)&&(b_links==0)&&(b_rechts==0))
{
    if(VOK==0)                  state = 2;    /* langsam vorfahren */
    else if((VOK==0)&&(WLF==1)) state = 6;    /* state 6 : drehe im Stand nach Links, weil Hindernis zu nah */
    else if(VOK==1)             state = beacon_entscheidung(nach_rechts);
}

else if((w_links==1)&&(w_rechts==1)&&(vorne==1)&&(b_links==0)&&(b_rechts==0))
{
    if((WRF==0)&&(WLF==0)&&(VOK==0))         state = 2; /* langsam vorfahren */
    else if((WRF==0)&&(WLF==1)&&(VOK==0))    state = 4; /* state 4 : Rechts drehen */
    else if((WRF==1)&&(WLF==0)&&(VOK==0))    state = 5; /* state 5 : Links drehen */
    else if((WRF==1)&&(WLF==1)&&(VOK==0))    state = 3; /* state 0 : Stop - undefinierte Situation */
    else if(((WRF==0)&&(WLF==0)&&(VOK==1))||
            ((WRF==0)&&(WLF==1)&&(VOK==1)))  state = beacon_entscheidung(nach_rechts);
    else if((WRF==1)&&(WLF==0)&&(VOK==1))    state = beacon_entscheidung(nach_links);
    else if((WRF==1)&&(WLF==1)&&(VOK==1))    state = 14;  /* state 14 : sofort schiessen */
}

/* Hindernis und Ball und Wand */
else if(((w_links==0)&&(w_rechts==1)&&(vorne==1)&&(b_links==0)&&(b_rechts==1))||
        ((w_links==1)&&(w_rechts==0)&&(vorne==1)&&(b_links==0)&&(b_rechts==1))||
        ((w_links==1)&&(w_rechts==1)&&(vorne==1)&&(b_links==0)&&(b_rechts==1)))
{
    if(VOK==0)  state = 6;  /* state 6 : Im Stand nach Rechts drehen - Ball zu nah */
    else if(VOK==1)
    {
        if(w_rechts==1)     state = beacon_entscheidung(nach_links);
        else if(w_links==1) state = beacon_entscheidung(nach_rechts);
        else                state = beacon_entscheidung(nach_links);
    }
}

else if(((w_links==0)&&(w_rechts==1)&&(vorne==1)&&(b_links==1)&&(b_rechts==0))||
        ((w_links==1)&&(w_rechts==0)&&(vorne==1)&&(b_links==1)&&(b_rechts==0))||
        ((w_links==1)&&(w_rechts==1)&&(vorne==1)&&(b_links==1)&&(b_rechts==0)))
{
    if(VOK==0)  state = 7;  /* state 7 : drehe im Stand nach Links, weil Hindernis zu nah */
    else if(VOK==1)
    {
        if(w_rechts==1)     state = beacon_entscheidung(nach_links);
        else if(w_links==1) state = beacon_entscheidung(nach_rechts);
        else                state = beacon_entscheidung(nach_rechts);
    }
}

/* Wand Aktionen */
else if((w_links==0)&&(w_rechts==1)&&(vorne==0)&&(b_links==0)&&(b_rechts==0))
{
    state = 8;  /* state 8 : Wall Follow Rechts */
}
```

```
        else if((w_links==1)&&(w_rechts==0)&&(vorne==0)&&(b_links==0)&&(b_rechts==0))
        {
            state = 9;  /* state 9 : Wall Follow Links */
        }

        else if((w_links==1)&&(w_rechts==1)&&(vorne==0)&&(b_links==0)&&(b_rechts==0))
        {
            if(last_state == 8)   /* laste_state 8 : Wall Follow Rechts */
            {
                if(analog(wand_rechts) > analog(wand_links)) state = 8;  /* state 8 : Wall Follow Rechts */
                else
                {
                    if(WLF==1)  state = 7;  /* state = 7 : drehe im Stand nach Links, weil Hindernis zu nah */
                    else        state = 10; /* state = 10 : Wall Follow Rechts, aber naeher an der Rechten Wand */
                }
            }

            else if(last_state = 9)   /* laste_state 9 : Wall Follow Links */
            {
                if(analog(wand_links) > analog(wand_rechts)) state = 9;  /* state 9 : Wall Follow Links */
                else
                {
                    if(WRF==1)  state = 6;  /* state 6 : drehe im Stand nach Rechts, weil Hindernis zu nah */
                    else        state = 11; /* state 11 : Wall Follow Links, aber naeher an der Linken Wand */
                }
            }
        }

        /* eigentlich keine Moegliche Aktion */
        else if((b_links&&b_rechts)==1)  state = last_state;
    }

    else if((v_l_bump==1)||(v_r_bump==1)||(h_bump==1)) /* Bumper(s) ist/sind angesprochen worden */
    {
        WRF = 0;
        WLF = 0;
        BRF = 0;
        BLF = 0;
        VOK = 0;

        /* HINTEN : Vorfahren, Achtung!!!*/
        if((v_l_bump==0)&&(v_r_bump==0)&&(h_bump==1))
        {
            /* von hinten angestossen - schauen, ob vorne was ist */
            if((w_links==0)&&(w_rechts==0))state = 1;  /* fahre vorwaerts */
            else if((w_links==0)&&(w_rechts==1))  state = 5;  /* drehe links */
            else if((w_links==1)&&(w_rechts==0))  state = 4;  /* drehe rechts */
            else if((w_links==1)&&(w_rechts==1))  state = 2;  /* fahre langsam vorwaerts */
        }

        /* VORNE RECHTS / VORNE LINKS / VORNE LINKS und VORNE RECHTS : zurueck fahren */
        else if(((v_l_bump||v_r_bump)==1)&&(h_bump==0))  state = 3;  /* zurueck fahren */

        /* VORNE RECHTS und HINTEN / VORNE LINKS und HINTEN / VORNE LINKS und VORNE RECHTS
        und HINTEN RECHTS : Stop!!! keine Moeglichkeit sich zu befreien*/
        else if(((v_l_bump||v_r_bump)==1)&&(h_bump==1))
        {
            beep();
            state = 0;  /* zurueck fahren */
        }
    }
    return state;
}
```

SchussKontrolle.c

```
void schuss_kontrolle()
{
    int dauer;
    int i, w;
    dauer = 0;

    while(1)
    {
        if(VOK)
        {
            dauer++;
            if(dauer == kontakt_zeit)
            {
                servo_deg(schuss);
                sleep(0.3);
                servo_deg(null_pos);
                dauer=0;
            }
        }
        else
        {
            servo_deg(null_pos);
        }
    }
}
```

Bump.c

```
void bump()
{
    while(1)
    {
        /* VORNE LINKS wird der Bumper aktiviert */
        if(!vorne_links_bump)  hv_l_bump = 0;
        else                   hv_l_bump = 1;

        /* VORNE RECHTS wird der Bumper aktiviert */
        if(!vorne_rechts_bump) hv_r_bump = 0;
        else                   hv_r_bump = 1;

        /* HINTEN wird der Bumper aktiviert */
        if(!hinten_bump)  hh_bump = 0;
        else              hh_bump = 1;
    }
}
```

Bewegen.c

```
int  diff, ist, speed, rad_keine_wand, rad_wand, clip;

void stop()
{
    motor(mot_links,stufe0);
    motor(mot_rechts,stufe0);
}

void vorwaerts()
{
```

```
    motor(mot_links,stufe7);
    motor(mot_rechts,stufe7);
}

void langsam()
{
motor(mot_links,stufe5);
motor(mot_rechts,stufe5);
}

void zurueck()
{
    motor(mot_links,-stufe6);
    motor(mot_rechts,-stufe6);
    sleep(0.5);
}

void rechts_drehen()
{
    motor(mot_links,stufe6);
    motor(mot_rechts,stufe1);
}

void links_drehen()
{
    motor(mot_links,stufe1);
    motor(mot_rechts,stufe6);
}

void rechts_stand_drehen()
{
    motor(mot_links,stufe6);
    motor(mot_rechts,-stufe6);
}

void links_stand_drehen()
{
    motor(mot_links,-stufe6);
    motor(mot_rechts,stufe6);
}

void langsam_rechts_stand_drehen()
{
    motor(mot_links,stufe5);
    motor(mot_rechts,-stufe5);
}

void langsam_links_stand_drehen()
{
    motor(mot_links,-stufe5);
    motor(mot_rechts,stufe5);
}

void hab_ball_schiessen()
{
    servo_deg(schuss);
    sleep(0.3);
    servo_deg(null_pos);
}

void wall_follow_rechts(int entfernung)
{
    int sollWert;

    if(entfernung == normal)   sollWert = soll;
```

```
    else if(entfernung == nah) sollWert = min_soll;

    rad_wand = mot_rechts;
    rad_keine_wand = mot_links;
    clip = clip_Wert;

    ist = analog(wand_rechts);

    diff = sollWert-ist;
    if(diff < 0)
    diff = diff *(-1);

    if(diff < 5)
    {
        motor(mot_links,stufe7);
        motor(mot_rechts,stufe7);
    }
    else
    {
        /* zu weit weg*/
        if(sollWert >= ist)
        {
            speed = stufe7 - ((sollWert-ist)*faktor);
            if(speed < 50)  {speed = clip;}
            motor(rad_keine_wand,stufe7);
            motor(rad_wand,speed);
        }

        /*zu nah dran*/
        if(sollWert < ist)
        {
            speed = stufe7 - ((ist-sollWert)*faktor);
            if(speed < 50)  {speed = clip;}
            motor(rad_wand,stufe7);
            motor(rad_keine_wand,speed);
        }
    }
}


void wall_follow_links(int entfernung)
{
    int sollWert;

    if(entfernung == normal)   sollWert = soll;
    else if(entfernung == nah) sollWert = min_soll;

    rad_wand = mot_links;
    rad_keine_wand = mot_rechts;
    clip = clip_Wert;

    ist = analog(wand_links);

    diff = sollWert-ist;
    if(diff < 0)
    diff = diff *(-1);

    if(diff < 5)
    {
      motor(mot_links,stufe7);
      motor(mot_rechts,stufe7);
    }
    else
    {
        /* zu weit weg*/
```

```
        if(sollWert >= ist)
        {
            speed = stufe7 - ((sollWert-ist)*faktor);
            if(speed < 50)  {speed = clip;}
            motor(rad_keine_wand,stufe7);
            motor(rad_wand,speed);
        }

        /*zu nah dran*/
        if(sollWert < ist)
        {
            speed = stufe7 - ((ist-sollWert)*faktor);
            if(speed < 50)  {speed = clip;}
            motor(rad_wand,stufe7);
            motor(rad_keine_wand,speed);
        }
    }
}
```

In diesem Bereich sehen sie Programmteile von Ein- und Zweiauge. Da Zweiauge eine Erweiterung von Einauge ist, wird hier nur zwei Programmteile von Zweiauge abgebildet.

Scanner.c

```
void scanner_langsam_nach_rechts()
{
    grad   = grad + pl_grad;
    f_grad = (float)grad;
    servo_deg(f_grad);
}

void scanner_langsam_nach_links()
{
    grad   = grad - pl_grad;
    f_grad = (float)grad;
    servo_deg(f_grad);
}

void scanner_nach_rechts()
{
    grad   = grad + p_grad;
    f_grad = (float)grad;
    servo_deg(f_grad);
}

void scanner_nach_links()
{
    grad   = grad - p_grad;
    f_grad = (float)grad;
    servo_deg(f_grad);
}

void scanner()
{
    int ball_gesehen_grad, last_SRI, i, rs_unten, rs_oben, ls_unten, ls_oben;

    grad = rs_mitte;
    SRI = 1;                /* Richtung des Scanners: 0 = dreht nach links ; 1 = dreht nach rechts */
    BGE = 0;                /* Flag: Ball gehabt */
```

```
rBER = 0;               /* Flag: Ball gefunden */
lBER = 0;               /* Flag: Ball gefunden */
ball_gesehen_grad = 0;  /* um zu regestrieren, wo der Ball das letzte mal gesehen wurde */
BOK = 0;  rSKM = 0;  lSKM = 0;  rSGM = 0; lSGM = 0;  rSIM = 0;  lSIM = 0;

while(1)
{
    /* Werte des Scanners erfassen */
    rs_unten = analog(sensor_rechts_unten);
    rs_oben  = analog(sensor_rechts_oben);
    ls_unten = analog(sensor_links_unten);
    ls_oben  = analog(sensor_links_oben);

    /* Ball erkennung */
    if((rs_oben < (rs_unten-20)) && (rs_unten > ballerkennung))  rBER = 1;
    else                                                        rBER = 0;
    if((ls_oben < (ls_unten-20)) && (ls_unten > ballerkennung))  lBER = 1;
    else                                                        lBER = 0;

    /* Stellung des Scanners */
    if(grad < rs_mitte)  rSKM = 1;
    else                 rSKM = 0;
    if(grad > rs_mitte)  rSGM = 1;
    else                 rSGM = 0;
    if(grad < ls_mitte)  lSKM = 1;
    else                 lSKM = 0;
    if(grad > ls_mitte)  lSGM = 1;
    else                 lSGM = 0;
    if((grad > (rs_mitte-3))&&(grad < (rs_mitte+3)))  rSIM = 1;
    else                                              rSIM = 0;
    if((grad > (ls_mitte-3))&&(grad < (ls_mitte+3)))  lSIM = 1;
    else                                              lSIM = 0;

    if(((rBER==1)||(lBER==1))&&(BOK==1))
    {
        if((last_SRI==0)&&(grad > (ball_gesehen_grad-5)))
        {
            scanner_langsam_nach_links();
            if(grad < s_links)            SRI = 1;
        }
        else if((last_SRI==1)&&(grad < (ball_gesehen_grad+5)))
        {
            scanner_langsam_nach_rechts();
            if(grad > s_rechts)           SRI = 0;
        }
        if(grad < (ball_gesehen_grad-5))  SRI = 1;
        if(grad > (ball_gesehen_grad+5))  SRI = 0;
        last_SRI = SRI;
    }
    else if(((rBER==1)||(lBER==1))&&(BOK==0)&&(BGE==1))
    {
        if((grad - ball_gesehen_grad) < 0)
        {
            SRI = 1;  BOK = 1;
        }
        else if((grad - ball_gesehen_grad) > 0)
        {
            SRI = 0;  BOK = 1;
        }
        else if((grad - ball_gesehen_grad) == 0)
        {
            SRI = last_SRI;  BOK = 1;
        }
        last_SRI = SRI;
    }
```

```
        else if(((rBER == 1)||(lBER==1))&&(BOK==0)&&(BGE==0))  /* Scanner hat Ball gefunden */
        {
            BGE = 1;  BOK = 1;  ball_gesehen_grad = grad;  last_SRI = SRI;
        }
        else if(((rBER==0)||(lBER==0))&&(BGE==1))
        {
            BOK = 0;  BGE = 0;
            if(last_SRI = 0)
            {
                scanner_nach_links();
                if(grad < s_links)
                {
                    SRI = 1;  rBER = 0;  lBER = 0;  BOK = 0;  BGE = 0;
                }
            }
            else if(last_SRI = 1)
            {
                scanner_nach_rechts();
                if(grad > s_rechts)
                {
                    SRI = 0;  rBER = 0;  lBER = 0;  BOK = 0;  BGE = 0;
                }
            }
            last_SRI = SRI;
        }

        /* Scanner sucht */
        else if(((rBER==0)||(lBER==0))&&(BGE==0))
        {
            if(SRI==1)
            {
                scanner_nach_rechts();
                if(grad > s_rechts)  SRI = 0;
            }
            else if(SRI==0)
            {
                scanner_nach_links();
                if(grad < s_links)  SRI = 1;
            }
            last_SRI = SRI;
        }
    }
}
```

Status.c

```
/* Globale Variablen */
int status(int last_state)
{
    int state, m_rBER, m_lBER, m_rSKM, m_rSGM, m_rSIM, m_lSKM, m_lSGM, m_lSIM, links_Wand, rechts_Wand,
        WRF, WLF, w_links, w_rechts, wm_links, wm_rechts, rechts_Ball, links_Ball, BRF, BLF, text;

    m_rBER = rBER;  m_lBER = lBER;  m_rSKM = rSKM;  m_lSKM = lSKM;
    m_rSGM = rSGM;  m_lSGM = lSGM;  m_rSIM = rSIM;  m_lSIM = lSIM;

    links_Wand  = analog(sensor_links_oben);  /* wand_links: siehe Define */
    rechts_Wand = analog(sensor_rechts_oben); /* wand_rechts: siehe Define */
    rechts_Ball = analog(sensor_rechts_unten);
    links_Ball  = analog(sensor_links_unten);

    if(rechts_Wand > zu_nah)      WRF = 1;  /* FLAG: rechts_Wand ist zu na */
    else                          WRF = 0;  /* FLAG: rechts_Wand weit genug weg */
    if(links_Wand > zu_nah)       WLF = 1;  /* FLAG: links_Wand ist zu na */
```

```
else                          WLF = 0;  /* FLAG: links_Wand weit genug weg */
if(rechts_Ball > zu_nah_Ball) BRF = 1;  /* FLAG: rechts_Ball ist zu na */
else                          BRF = 0;  /* FLAG: rechts_Ball weit genug weg */
if(links_Ball > zu_nah_Ball)  BLF = 1;  /* FLAG: links_Ball ist zu na */
else                          BLF = 0;  /* FLAG: links_Ball weit genug weg */

/* Wand erkennung - Wand links*/
if(links_Wand > minentfernt)
{
    w_links   = 1;
    wm_links  = 1;
}
else
    if ((links_Wand > maxentfernt) && (wm_links == 1))
    {
        w_links = 1;
    }
    else
    {
        w_links  = 0;
        wm_links = 0;
    }

/* Wand erkennung - Wand rechts*/
if(rechts_Wand > minentfernt)
{
    w_rechts  = 1;
    wm_rechts = 1;
}
else
    if ((rechts_Wand > maxentfernt) && (wm_rechts == 1))
    {
        w_rechts = 1;
    }
    else
    {
        w_rechts  = 0;
        wm_rechts = 0;
    }

/* State - Machine */
if((w_links==0)&&(w_rechts==0)&&(m_rBER==0)&&(m_lBER==0))   state = 2;/* state 1 : vorwaerts fahren */
else if(((w_links==0)&&(w_rechts==0)&&(m_rBER==0)&&(m_lBER==1))||
        ((w_links==0)&&(w_rechts==1)&&(m_rBER==0)&&(m_lBER==1))||
        ((w_links==1)&&(w_rechts==0)&&(m_rBER==0)&&(m_lBER==1))||
        ((w_links==1)&&(w_rechts==1)&&(m_rBER==0)&&(m_lBER==1)))
{
    if(BLF==0)
    {
        if(((m_rSIM==1)&&(m_lSKM==1)&&(m_lSIM==0)&&(rSKM==0)&&(rSGM==0)&&(lSGM==0)) ||
           ((m_rSGM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSGM==0)))
        {
            state = 9; /* Ball links, links drehen */
        }
        if(((m_rSGM==1)&&(m_lSGM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSKM==0)) ||
           ((m_rSGM==1)&&(m_lSIM==1)&&(rSIM==0)&&(rSKM==0)&&(lSKM==0)&&(lSGM==0)))
        {
            state = 2; /* Ball rechts, langsam rechts drehen */
        }
        if((m_rSKM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSGM==0)&&(lSGM==0))
        {
            state = 7; /* Ball links, links drehen */
        }
    }
    else if(BLF==1)
```

```
    {
        if(((m_rSIM==1)&&(m_lSKM==1)&&(m_lSIM==0)&&(rSKM==0)&&(rSGM==0)&&(lSGM==0)) ||
           ((m_rSKM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSGM==0)&&(lSGM==0)))
        {
            state = 3; /* Ball hinter dem Trichter, zurueck */
        }
        if((m_rSGM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSGM==0))
        {
            state = 9; /* Ball links, langsam links drehen */
        }
        if((m_rSGM==1)&&(m_lSGM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSKM==0))
        {
            state = 10; /* Ball vorne, Beacon suchen */
        }
        if((m_rSGM==1)&&(m_lSIM==1)&&(rSIM==0)&&(rSKM==0)&&(lSKM==0)&&(lSGM==0))
        {
            state = 2; /* Ball vorne, vorwaerts */
        }
    }
}

else if(((w_links==0)&&(w_rechts==0)&&(m_rBER==1)&&(m_lBER==0))||
        ((w_links==0)&&(w_rechts==1)&&(m_rBER==1)&&(m_lBER==0))||
        ((w_links==1)&&(w_rechts==0)&&(m_rBER==1)&&(m_lBER==0))||
        ((w_links==1)&&(w_rechts==1)&&(m_rBER==1)&&(m_lBER==0)))
{
    if(BRF==0)
    {
        if(((m_rSIM==1)&&(m_lSKM==1)&&(m_lSIM==0)&&(rSKM==0)&&(rSGM==0)&&(lSGM==0)) ||
           ((m_rSKM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSGM==0)&&(lSGM==0)))
        {
            state = 2; /* Ball vorne, vorwaerts */
        }
        if(((m_rSGM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSGM==0)) ||
           ((m_rSGM==1)&&(m_lSGM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSKM==0)) ||
           ((m_rSGM==1)&&(m_lSIM==1)&&(rSIM==0)&&(rSKM==0)&&(lSKM==0)&&(lSGM==0)))
        {
            state = 8; /* Ball rechts, langsam rechts drehen */
        }
    }
    else if(BRF==1)
    {
        if((m_rSIM==1)&&(m_lSKM==1)&&(m_lSIM==0)&&(rSKM==0)&&(rSGM==0)&&(lSGM==0))
        {
            state = 2; /* Ball vorne, vorwaerts */
        }
        if((m_rSGM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSGM==0))
        {
            state = 8; /* Ball rechts, langsam rechts drehen */
        }
        if(((m_rSGM==1)&&(m_lSGM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSKM==0)) ||
           ((m_rSGM==1)&&(m_lSIM==1)&&(rSIM==0)&&(rSKM==0)&&(lSKM==0)&&(lSGM==0)))
        {
            state = 3; /* Ball hinter dem Trichter, zurueck */
        }
        if((m_rSKM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSGM==0)&&(lSGM==0))
        {
            state = 10; /* Ball vorne, Baecon suchen */
        }
    }
}
else if((w_rechts==0)&&(w_links==1)&&(m_rBER==0)&&(m_lBER==0))
{
    if(WLF==1)
    {
```

```
            if(((m_rSIM==1)&&(m_lSKM==1)&&(m_lSIM==0)&&(rSKM==0)&&(rSGM==0)&&(lSGM==0)) ||
               ((m_rSGM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSGM==0)) ||
               ((m_rSKM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSGM==0)&&(lSGM==0)) ||
               ((m_rSGM==1)&&(m_lSIM==1)&&(rSIM==0)&&(rSKM==0)&&(lSKM==0)&&(lSGM==0)))
            {
                state = 6; /* Wand links, rechts fahren */
            }
            if((m_rSGM==1)&&(m_lSGM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSKM==0))
            {
                state = 3; /* Wand vorne rechts, zurueck */
            }
        }
        else if(WLF==0)    state = 2;
    }
    else if((w_rechts==1)&&(w_links==0)&&(m_rBER==0)&&(m_lBER==0))
    {
        if(WLF==1)
        {
            if(((m_rSIM==1)&&(m_lSKM==1)&&(m_lSIM==0)&&(rSKM==0)&&(rSGM==0)&&(lSGM==0)) ||
               ((m_rSKM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSGM==0)&&(lSGM==0)))
            {
                state = 3; /* Wand vorne, links drehen */
            }
            if(((m_rSGM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSGM==0)) ||
               ((m_rSGM==1)&&(m_lSGM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSKM==0)) ||
               ((m_rSGM==1)&&(m_lSIM==1)&&(rSIM==0)&&(rSKM==0)&&(lSKM==0)&&(lSGM==0)))
            {
                state = 7; /* Wand links, links drehen */
            }
        }
        else if(WLF==0)    state = 2;
    }
    else if((w_rechts==1)&&(w_links==1)&&(m_rBER==0)&&(m_lBER==0))
    {
        if((WRF==0)&&(WLF==0))
        {
            if(((m_rSIM==1)&&(m_lSKM==1)&&(m_lSIM==0)&&(rSKM==0)&&(rSGM==0)&&(lSGM==0)) ||
               ((m_rSGM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSGM==0)) ||
               ((m_rSGM==1)&&(m_lSGM==1)&&(rSIM==0)&&(lSIM==0)&&(rSKM==0)&&(lSKM==0)) ||
               ((m_rSKM==1)&&(m_lSKM==1)&&(rSIM==0)&&(lSIM==0)&&(rSGM==0)&&(lSGM==0)) ||
               ((m_rSGM==1)&&(m_lSIM==1)&&(rSIM==0)&&(rSKM==0)&&(lSKM==0)&&(lSGM==0)))
            {
                state = 2; /* Wand vorne und links */
            }
        }
        else if((WRF==0)&&(WLF==1))  state = 6;
        else if((WRF==1)&&(WLF==0))  state = 7;
        else if((WRF==1)&&(WLF==1))  state = 3;
    }
    return state;
}
```