HAW Hamburg
Studiengang Technische Informatik

**Projekt Roboterfußball**

# Dokumentation von Robot 2

J. Heitsch, T.Steinbach

6. März 2008

Betreut durch Prof. Dr. rer. nat. Kai von Luck
und Prof. Dr. rer. nat. Gunter Klemke

# Inhaltsverzeichnis

# 1 Allgemeines

## 1.1 Team

- J. Heitsch (1821866)
- T.Steinbach (1823293)

## 1.2 Ziel des Projektes

Der Pokal muss zurück nach Hamburg.

## 1.3 Zeitplanung

Für das Projekt stehen 16 Tage zur Verfügung. In den ersten Terminen wurden die Eigenschaften der Sensoren studiert und eine Hardware- Platform aufgabaut, welche aber im Laufe des Projektes einige Modifikationen über sich ergehen lassen musste. Später nachdem die Hardware soweit stand wurden Grundfunktionen in Software geschrieben, wie z.B.:

- Kollisionsvermeidung mit Wänden und anderen Hindernissen.
- Ortung des Balls
- Erkennung der Tore

## 1.4 Tools

Die Software wurde auf der Eclipse IDE (CDT) geschrieben. Als Compiler diente der auf das Aksenboard angepasste SDCC.

# 2 Aufbau

## 2.1 Hardware

### 2.1.1 Platform

Für die Realisierung des Ziels stand uns eine Omidirektinale Platform von qfix zur Verfügung.

Diese verfügt über drei angetriebene Omnidirektionale Räder welche jeweils um 120 Grad versetzt sind. Durch diese Anordnung ist es möglich in jede Richtung zu fahren und sich dabei, oder auch auf der Stelle, zu drehen. Weitere Aufbauten auf der Platform wurden mit LEGO- Technik Bausteinen realisiert.

Bei der Verkabelung der Motoren sollte darauf geachtet werden, dass die Motoren schnell gewechselt werden können. Bei Getriebedefekten sollte nur das Getriebe getauscht werden und nicht der ganze Motor. Das erspart Lötarbeiten.

### 2.1.2 Kicker

Es wurde mit Hilfe eines Elektromagneten ein einfacher Kicker gebaut. Der Kicker wird an einen Motorport angeschlossen. Der Kicker darf nur sehr kurz ausgelöst werden, da die H-Brücken der Motortreiber die Leistung des Kickers (ca. 2A) nicht dauerhaft aushalten. Leider ist der Hub des Kickers nicht besonders groß, aber für das kleine Spielfeld ausreichend. Für einen Leistungsfähigeren Kicker müsste man sich z.B. die Kicker der Small Size League anschauen. Dort werden die Kicker über aufgeladene Kondensatoren ausgelöst.

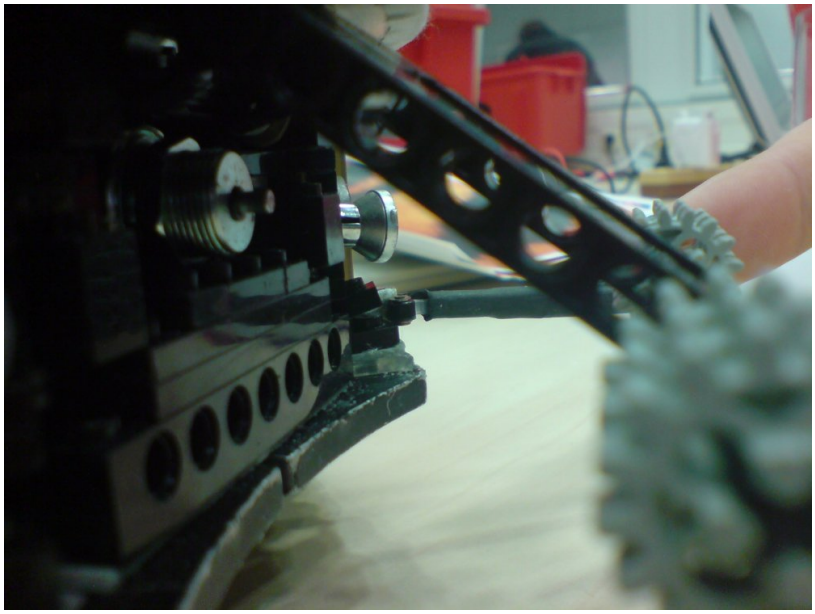

Abbildung 2.1: Der Kicker

### 2.1.3 Sensoren

Als Sensoren standen uns zunächst folgene Typen zur Verfügung:

- SHARP Distanz Sensoren (Messbereich ca. 10-80cm)
- IR Sensoren
- Tor Sensoren
- kurz Distanz Sensor
- Taster

#### SHARP Distanz Sensoren

Messbereich ca. 10-80cm Die Distanz Sensoren werden für die Erkennung von Wänden und anderen Hindernissen benötigt.
Es sind ingesammt vier Distanzsensoren verbaut.
Einer ist direkt nach vorne gerichtet, zwei Überwachen den Bereich schräg vorne und der letzte ist auf einem Servo montiert um einen seitlichen Blick zu realisieren.
Die Sensoren sind soweit wie möglich von den Rändern der Plattform weg montiert, da somit der Messbereich besser ausgenutzt wird und weniger Messfehler im Nahbereich auftreten.

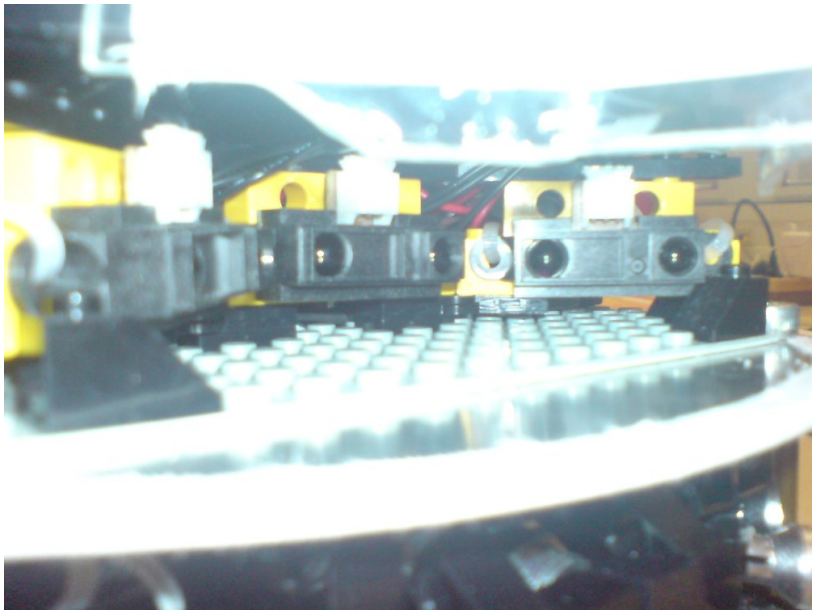

Abbildung 2.2: Sharp Distance Sensoren

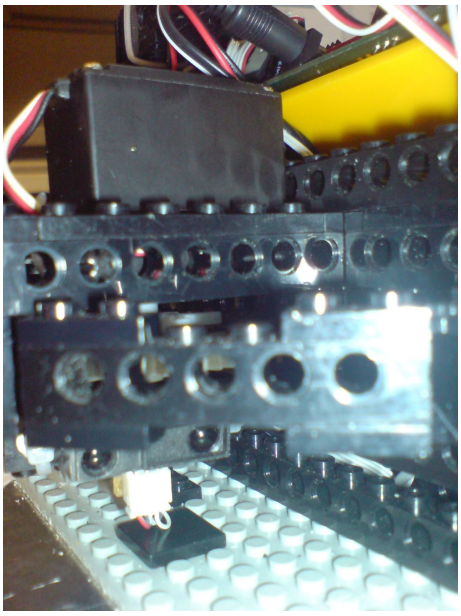

Abbildung 2.3: drehbarer Sharp Distance Sensor am Heck

### IR Sensoren

Die IR Sensoren werden für die Erkennung und Ortung des Balles verwendet. Es sind insgesammt zehn Sensoren um den Roboter verteilt. Die Reflektoren verbessern dabei die Reichweite. Hinzukommt eine mit Sensoren bestückte Leiste die den Ball vor dem Roboter erkennt und zwei Sensoren, welche bestimmen können, ob der Ball direkt vor

dem Roboter liegt bzw. ob der Ball kurz vor dem Roboter liegt.
Die Sensoren sind in Gruppen eingeteilt: Die zehn Sensoren rund um den Roboter sind jeweils in zweier Paaren parallel geschaltet, um die Reichweite und den Erkennungsbereich zu erweitern und um Ports einzusparen.
Hier gibt es zudem aus der Softwaretechnischen Sicht einen Sinn, der die Parallelschaltung rechtfertigt, denn die parallel geschalteten Sensoren beeiflussen das Verhalten zusammen, sollte jeder Sensor das Verhalten beeinflussen wäre der Software aufwand erheblich größer, bei gleichem Nutzen. Mehr dazu in 2.2.

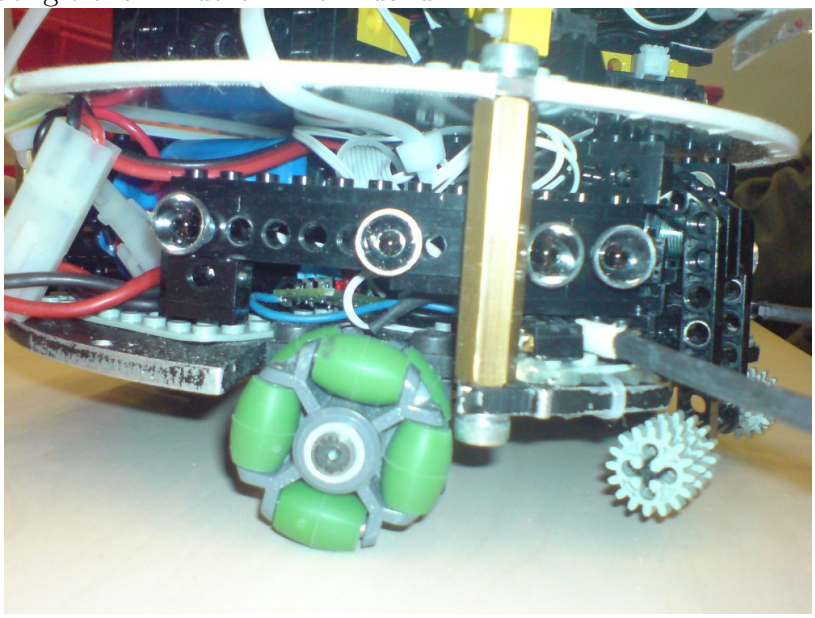

Abbildung 2.4: Paarweise parallele IR Sensoren

### Tor Sensoren

Die Tor Sensoren sind für das Auffinden der Toren da.
Es sind insgesammt vier verbaut worden. Die drei nach vorne gerichteten sind in einer Fokusiereinheit unterberacht. Der vierte ist seitlich angebracht. Seine Funktion wird in 2.2 weiter erläutert.

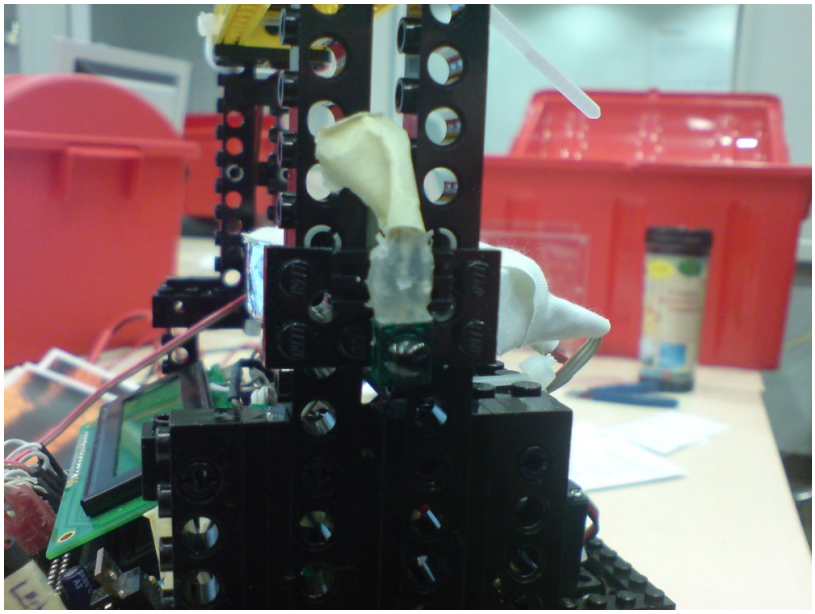

Abbildung 2.5: Der seitliche Tor Sensor

### kurz Distanz Sensor

Der Sensor ist unter dem Roboter angebracht und erkennt ob der Roboter auf dem Boden steht oder nicht.
Die Verwenung wird in 2.2 erläutert.

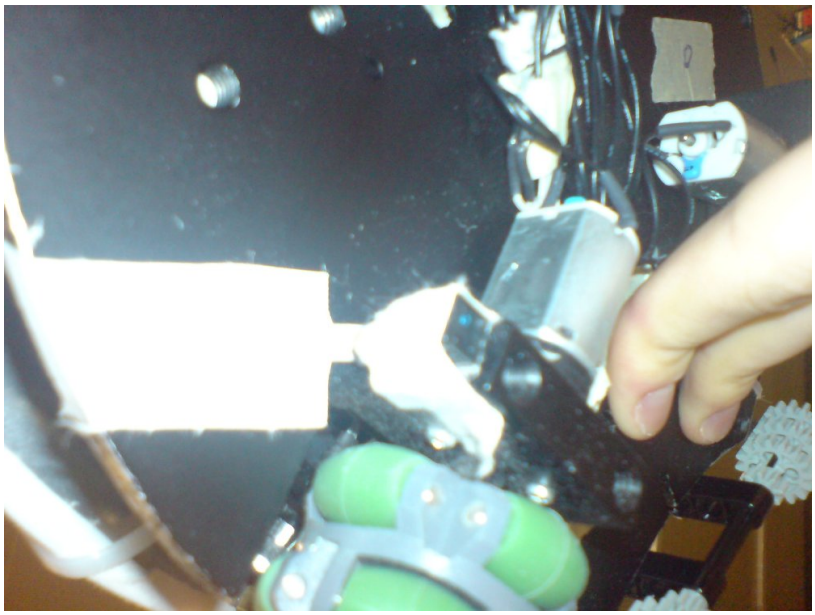

Abbildung 2.6: Flugerkennung

### Taster

Es sind nach vorne gerichtete Fühler verbaut. Diese reagieren auf die Berührung mit einem Gegenstand. Sollte z.B. der Ball zu einer Seite wegrollen, löst der Fühler aus und der Roboter kann korrigieren.

Abbildung 2.7: Fühler

Abbildung 2.8: Fühler betätigt

### 2.1.4 Verkleidung

Da bei den Matches teilweise erhebliche Schäden entstanden haben wir eine Schutzhülle gebaut. Sie besteht aus durchsichtigem Plexiglas. Für die Distanzsensoren wurden Ausschnitte in das Glas gemacht, da die Distanzsensoren nicht mit den Reflektionen am Glas klarkommen. Die Ballsensoren können hinter dem Glas geschützt werden. Sie haben keine Probleme mit Reflektionen. Wegen der Optik und um für andere Roboter besser als Hindernis erkannt zu werden, wurden große Teile der Hülle beklebt.

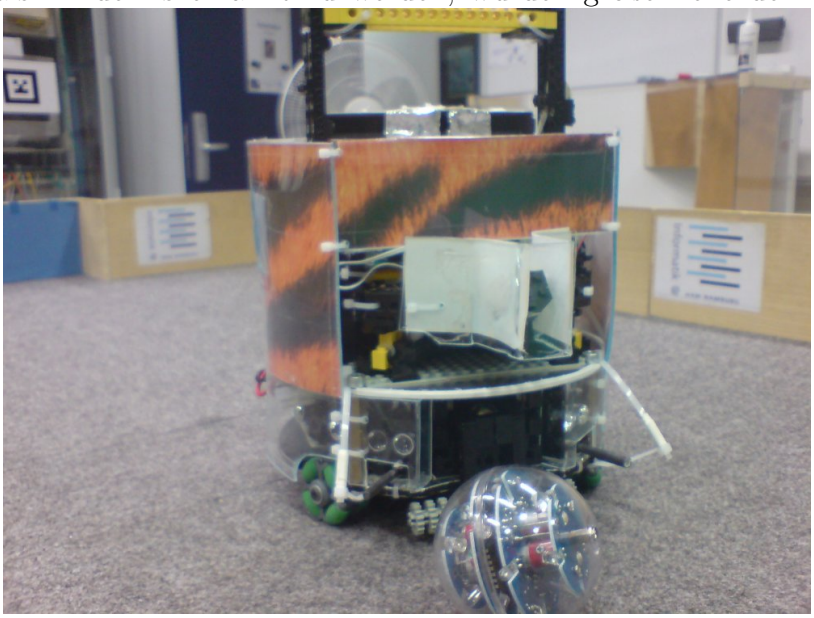

Abbildung 2.9: Verkleidung

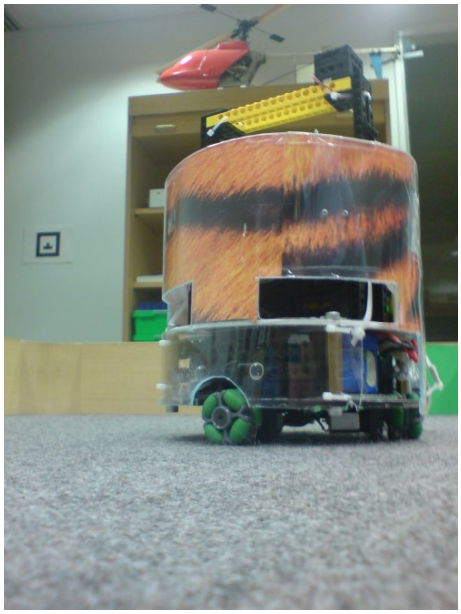

Abbildung 2.10: Verkleidung von Hinten

## 2.2 Software

### 2.2.1 Architektur

Die zuerst verwendete Subsumption Architektur stellte sich schnell als z.T. hinderlich da, da sich Verhalten in diesem Modell komplett gegeneinander ausschließen. Ein verbinden mehrerer Verhalten war nur mit sehr viel Programmieraufwand möglich, der nicht gerechtfertig gewesen wäre. Daher wurde die Architetur abgwandelt und die Idee der Dual Dynamics mit aufgegriffen. Es war jetzt möglich Verhalten miteinander zu kombinieren, was aber auch neue Probleme mit sich brachte, da Verhalten isoliert funktionierten, aber in Verbindung mit anderen diverse Konflikte mit sich brachten.

### 2.2.2 Realisierung DD

Die Idee der Dual Dynamics lässt sich am besten mit einer Abstimmung vergleichen. Jedes Verhalten kann eine Richtung, Drehrichtung und Geschwindigkeit vorschlagen. Dazu gibt es Prioritäten für jedes Verhalten. Das gewichtete Mittel aller Vorschläge wird berechnet und damit dann die Motoren gesetzt. Wir haben für die Ansteuerung der Motoren die omnidrivedegree Bibiliothek von Oliver Kökritz genutzt. Diese ermöglicht den Roboter in jede Richtung fahren zu lassen und zusätlich dabei eine Drehung zu vollziehen.

Da die omnidrivedegree Bibiliothek versucht immer mit die maximal mögliche Geschwindigkeit zu fahren, musste dieses angepasst werden, so dass auch ein langsameres

Fahren möglich ist, was gewünscht ist, um den Ball nicht zu häufig unkontolliert weg zu stoßen.

### 2.2.3 Initialisierung

Der Roboter wird beim Start einmal Initialisert:

```
void init()
{
    SENSOR_AVG_INIT(avoid_Mid)
  SENSOR_AVG_INIT(avoid_Lef)
  SENSOR_AVG_INIT(avoid_Rig)

  SENSOR_AVG_INIT(ball_Mid)
  SENSOR_AVG_INIT(ball_Lef)
  SENSOR_AVG_INIT(ball_Rig)
  SENSOR_AVG_INIT(ball_Beh)

  SENSOR_AVG_INIT(ball_outer_Rig)
  SENSOR_AVG_INIT(ball_outer_Lef)

  SENSOR_AVG_INIT(fly)

  lcd_cls();

  READ_AVOID_SENS
  READ_AVOID_SENS
  READ_AVOID_SENS
  READ_AVOID_SENS
  READ_BALL_SENS
  READ_BALL_SENS
  READ_BALL_SENS
  READ_BALL_SENS


  if(avoid_Rig.value >= DIST_SENSOR_NOT_OK_BAR){
    lcd_puts("Dist Rig not ok");
    sleep(5000);
  }
  if(avoid_Lef.value >= DIST_SENSOR_NOT_OK_BAR){
    lcd_puts("Dist Lef not ok");
    sleep(5000);
  }
  if(avoid_Mid.value >= DIST_SENSOR_NOT_OK_BAR){
    lcd_puts("Dist Mid not ok");
```

```
    sleep(5000);
  }
  if(avoid_Mid.value >= DIST_SENSOR_NOT_OK_BAR){
    lcd_puts("Dist Mid not ok");
    sleep(5000);
  }
  if(ball_outer_Rig.value >= BALL_SENSOR_NOT_OK_BAR){
    lcd_puts("Ball OR not ok");
    sleep(5000);
  }
  if(ball_outer_Lef.value >= BALL_SENSOR_NOT_OK_BAR){
    lcd_puts("Ball OL not ok");
    sleep(5000);
  }
  if(ball_Rig.value >= BALL_SENSOR_NOT_OK_BAR){
    lcd_puts("Ball IR not ok");
    sleep(5000);
  }
  if(ball_Lef.value >= BALL_SENSOR_NOT_OK_BAR){
    lcd_puts("Ball IL not ok");
    sleep(5000);
  }
  if(ball_Mid.value >= BALL_SENSOR_NOT_OK_BAR){
    lcd_puts("Ball M not ok");
    sleep(5000);
  }
  if(ball_Beh.value >= BALL_SENSOR_NOT_OK_BAR){
    lcd_puts("Ball B not ok");
    sleep(5000);
  }



        lcd_cls();
        INIT_GOAL(dip_pin(0));



        if(!read_bar_values && 0x0A != bar_values_are_read)
        {
            avoid_Mid_bar = 160;
            avoid_Rig_bar = 160;
            avoid_Lef_bar = 160;
            avoid_Loo_bar = 175;
```

```
        fly_bar = analog(FLY_SENSOR)+50;
        bar_values_are_read = 0;
    }
    else if(read_bar_values)
    {
        lcd_puts("AVOID");
            while(digital_in(0));
        sleep(500);


        while(digital_in(0))
        {
          READ_AVOID_SENS
          SHOW_AVOID_SENS
          lcd_setxy(1,15);
          lcd_puts("1");
          sleep(100);
        }
        avoid_Mid_bar = avoid_Mid.value;
        avoid_Rig_bar = avoid_Rig.value;
        avoid_Lef_bar = avoid_Lef.value;
        fly_bar = analog(FLY_SENSOR)+50;
        sleep(500);

        LOOK_LEFT
        while(digital_in(0))
        {
          READ_AVOID_SENS
          SHOW_AVOID_SENS
          lcd_setxy(1,15);
          lcd_puts("2");
          sleep(100);
        }
        avoid_Loo_bar = avoid_Loo.value;

        sleep(500);
        bar_values_are_read = 0x0A;
    }

    ball_Mid_bar = 150;
    ball_Lef_bar =
        ball_Rig_bar =
            ball_outer_Rig_bar =
                ball_outer_Lef_bar =
```

```
                    ball_Beh_bar = SEE_BALL_BAR;


    dd_init(&motor_ctr);

    lastBallSeen = 0;
    wasFlying = FLY_DELAY;
    lastTurn = NO_TURN;
    lcd_cls();

    WATCHDOG_AKTIVIEREN();

}
```

Es müssen in diesem Schritt Schwellwerte für die Distanz Sensoren eingelesen werden. Es gibt drei Varianten mit denen der Roboter initialisert werden kann

- Standardwerte
- Einlesen von Werten (Kalibrieren)
- Übername bereits eingelesener Werte nach Reset.

Diese Unterscheidung ist wichtig, da es manchmal zu spontanen Resets des Systems kommen kann (z.B. durch statische Aufladung). Geschiet dies während eines Spiels wäre der Roboter nicht mehr unter Kontrolle, sofern nicht alte Werte genommen werden könnten. Sind keine Werte eingelesen worden, so werden einprogrammierte Standardwerte genommen. Die Erkennung eines Programmabsturzes übernimmt der am Ende gestartete Watchdog.

### 2.2.4 Superloop

Die Superloop ist das eigentliche Programm:

```
void AksenMain(void)
{
    unsigned char prio;
  read_bar_values = 0;

  if(!digital_in(0))
    read_bar_values = 1;


    while(!digital_in(0));

    init();
```

```
  if(!read_bar_values && 0x0A != bar_values_are_read)
    lcd_puts("using default..");
  else if (0x0A != bar_values_are_read)
      lcd_puts("using stored..");

  if(dip_pin(2)){
      lcd_setxy(1,0);
      lcd_puts("ELFMETER MODUS");
  }


while(1)
{
  while(digital_in(0))
  {
      WATCHDOG_RESET();
      led(0,!dip_pin(0));
          dd_clear(&motor_ctr);
      if(lastBallSeen)lastBallSeen--;
      if(wasFlying)wasFlying--;
      if(was_reset && !wasFlying)
          was_reset = 0;

      READ_AVOID_SENS
      READ_BALL_SENS
      READ_FLY_SENSOR


          if(!dip_pin(2)){
              not_see_ball_return();
              see_ball_and_start();
          }

          has_ball_and_wall_right_left();

          ball_behind();
      avoid();
      avoid_slide();
      has_ball_see_goal_in_front();
      has_ball_goal_sideways();
      has_ball_no_goal();

          has_ball_dribble();
```

```
        ballOuterFound();
        ballInnerFound();
        drive();


      if(IS_FLYING)
      {
        STOP
        wasFlying = FLY_DELAY;
      }
      else
        dd_set(&motor_ctr);

        //lcd_cls();




    }//while(!digital_in(0))
    WATCHDOG_RESET();
    STOP
    sleep(500);
    lcd_cls();
    lcd_puts("START ME!!!!!");
    while(digital_in(0))WATCHDOG_RESET();
    lcd_cls();
    sleep(300);
  }//while(1)
}
```

Nach der allgemeinen Initialisierung gibt es zwei Schleifen in denen das Programm laufen kann. Jede der Schleifen beschreibt einen Zustand des Roboters, die Äussere ist der Zustand HALT und die Innere der Zustand RUN. Mit dem Taster auf dem Roboter kann zwischen den Zuständen gewechselt werden. Die Zustände werden benötigt um den Roboter stoppen zu können.

Im Zustand RUN werden alle Sensoren gelesen und anschließend alle Verhalten „aufgerufen“. Es sind durch die Verwendung von Markos jedoch keine Aufrufe mehr. Makros wurden verwendet um die Zeitverluste durch die Aufrufe zu vermeiden.

Am Ende der Schleife werden die gesammelten Vorschläge der Verhalten ausgewertet und die Motoren entprechend angesteuert.

### 2.2.5 Verhalten

Im folgenden werden die Verhalten kurz erläutert.

### see_ball_and_start

```
#define see_ball_and_start()\
{\
    if(SEE_BALL_MIDDLE && wasFlying &&! was_reset && dip_pin(3))\
    {\
        led(3,1);\
        SET_DD_LIMIT(0,0,0,0,0);\
    }\
    else\
    {\
        led(3,0);\
        wasFlying = 0;\
    }\
}
```

Dieses Verhalten wird aktiv, wenn der Roboter gerade gestatet wurde bzw. auf der Spielfläche abgesetzt wurde und das Verhalten durch den Dip- Schalter aktiviert ist. Es verhindert ein sofortiges Losfahren und verwirrt somit den gegnerischen Roboter, da dieser das Tor nicht erkennen kann.

### not_see_ball_return

```
#define not_see_ball_return()\
{\
    if(!SEE_BALL_MIDDLE &&\
        !SEE_BALL_IN_FRONT &&\
        !SEE_BALL_IN_FRONT_FAR &&\
        !SEE_BALL_OUTER_LEFT &&\
        !SEE_BALL_OUTER_RIGHT &&\
        !SEE_BALL_LEFT &&\
      !SEE_BALL_RIGHT &&\
      !SEE_BALL_BEHIND)\
    {\
    if(toggle_start>0)\
        toggle_start--;\
        else\
        {\
            led(2,1);\
            if(is_in_own_goal)\
                move_in_own_goal()\
            else\
            {\
                drive_to_own_goal();\
```

```
                    if(SEE_GOAL_MIDDLE || (SEE_GOAL_RIGHT &&
                        SEE_GOAL_LEFT))\
                        if(SEE_WALL_MIDDLE)\
                        {\
                            is_in_own_goal = TRUE;\
                            move_in_own_goal_toggle = 0;\
                        }\
                }\
            }\
    }\
    else\
    {\
        led(2,0);\
        toggle_start = TOGGLE_START_VALUE;\
        is_in_own_goal = FALSE;\
        INIT_GOAL(dip_pin(0)); /*tore richtig*/\
    }\
}
```

Dieses Verhalten wird aktiv, wenn der Roboter den Ball eine bestimmte Zeit lang nicht sieht. Dieses Verhalten startet das Verhalten drive_to_own_goal() welches den Roboter in das eigene Tor zurückfahren lässt um dieses zu versperren. Ist der Roboter im eigenen Tor angekommen wird das Verhalten move_in_own_goal() ausgeführt. Dieses lässt den Roboter im eigenen Tor rotieren in der Hoffnung Schüsse abzuwehren oder den Ball zu finden. Sollte der Ball also vom Gegner verdeckt sein, geht der Roboter in ein defensives Verhalten über um das eigene Tor zu verteidigen.

### has_ball_and_wall_right_left()

```
#define has_ball_and_wall_right_left()\
{\
    prio = 7;\
    if(SEE_BALL_IN_FRONT && SEE_WALL_LEFT && SEE_WALL_RIGHT)\
    {\
        if (!SEE_GOAL_BACK_LEFT)\
            TRY_TURN_LEFT(HIGH_SPEED, prio)\
        else\
            TRY_TURN_RIGHT(HIGH_SPEED, prio)\
    }\
}
```

Dieses Verhalten wird aktiv, wenn der Roboter mit dem Ball eine Ecke erkennt. Dies kann eine Ecke vom Spielfeld sein, oder eine ecke zwischen Spielfeld und anderem Roboter. In diesen Fällen versucht der Roboter sich mit dem Ball zum eigenen Tor zu drehen um

den gegnerische Roboter zu umfahren.

### avoid()

```
#define avoid()\
{\
    prio = PRIO_EXCLUSIV;\
    if(!lastBallSeen)\
      if(SEE_WALL_MIDDLE || SEE_WALL_LEFT || SEE_WALL_RIGHT)\
      {\
        if (avoid_Lef.value > avoid_Rig.value)\
                TRY_TURN_LEFT(SLOW_SPEED, prio)\
        else\
            TRY_TURN_RIGHT(SLOW_SPEED, prio)\
      }\
}
```

Dieses Verhalten implementiert das Ausweichen an Hindernissen.

### avoid_slide()()

```
#define avoid_slide()\
{\
    prio = 9;\
    if(0 == look_valid)\
    {\
        if(SEE_WALL_LOOK)\
            switch(looking_dir)\
            {\
                case RIGHT:\
                    SET_DD(90,TURN_SPEED,0,prio)\
                    break;\
                case LEFT:\
                    SET_DD(270,TURN_SPEED,0,prio)\
                    break;\
                default:break;\
            }\
    }\
    else\
        look_valid--;\
}
```

Das Verhalten wird aktiv, wenn sich der Roboter seitlich bewegt. Ob sich der Roboter seitlich bewegt, wird in der Funktion dd_set(\ldots) berechnet. Es verhindert, dass der

Roboter seitlich mit einem Hinderniss kollidiert. Hier ist der hintere drehbare Distanz Sensor aktiv.

### drive()

```
#define drive()\
{\
    if(!SEE_BALL_IN_FRONT && SEE_BALL_IN_FRONT_FAR)\
        SET_DD_LIMIT(180,100,0,7,3)\
    else\
        SET_DD(180,100,0,3)\
}
```

Das Verhalten lässt den Roboter permanent fahren.

### ballInnerFound()

```
#define ballInnerFound()\
{\
    prio = 5;\
  if (!SEE_BALL_MIDDLE && ! SEE_BALL_OUTER_LEFT && !
     SEE_BALL_OUTER_RIGHT)\
    if (SEE_BALL_LEFT)\
        if(!SEE_GOAL_LEFT && SEE_GOAL_RIGHT)\
        {\
            SET_DD(90,TURN_SPEED,0,prio)\
        }\
        else\
        {\
            SET_DD(180,TURN_SPEED,TURN_SPEED,prio)\
        } \
    else if (SEE_BALL_RIGHT)\
        if(!SEE_GOAL_RIGHT && SEE_GOAL_LEFT)\
        {\
            SET_DD(270,TURN_SPEED,0,prio)\
        }\
        else\
        {\
            SET_DD(180,TURN_SPEED,-TURN_SPEED,prio)\
        }\
}
```

Das Verhalten wird aktiv, wenn der Ball vorne seitlich gesehen wurde. Es wird unterschieden zwischen zwei Fällen. Sieht der Roboter den Ball und das Tor vor sich so fährt

er seitlich hinter den Ball um in der richtigen Schußposition zu sein ohne den Ball vorher an die Wand zu drängen. Sieht der Roboter hingegen nur den Ball, so fährt er schräg zu diesem.

### ballOuterFound()

```
#define ballOuterFound()\
{\
    prio = 7;\
  if (!SEE_BALL_MIDDLE)\
  {\
    if (SEE_BALL_OUTER_LEFT)\
        TURN_RIGHT(HIGH_SPEED, prio)\
    else if (SEE_BALL_OUTER_RIGHT)\
        TURN_LEFT(HIGH_SPEED, prio)\
  }\
}
```

Das Verhalten wird aktiv, wenn der Roboter den Ball neben sich sieht, es bewirkt, dass sich der Roboter sofort zum Ball dreht.

### ball_behind()

```
#define ball_behind()\
{\
    prio = PRIO_EXCLUSIV;\
    if(!SEE_BALL_MIDDLE &&\
        ! SEE_BALL_OUTER_LEFT &&\
        ! SEE_BALL_OUTER_RIGHT &&\
        ! SEE_BALL_LEFT &&\
        ! SEE_BALL_RIGHT){\
            if( SEE_BALL_BEHIND || lastBallSeen || (wasFlying &&
               SEE_WALL_MIDDLE))\
            {\
                if (!SEE_GOAL_BACK_LEFT)\
                    TRY_TURN_LEFT(HIGH_SPEED, prio)\
                else\
                    TRY_TURN_RIGHT(HIGH_SPEED, prio)\
                if (!lastBallSeen)\
                lastBallSeen = 50;\
            }\
        }\
        else\
            lastBallSeen = 0;\
```

```
}
```

Das Verhalten wird aktiv, wenn sich der Ball hinter dem Roboter befindet. Dieser Fall ist nach einem Ballverlusst mögglich, sowie nach einem Foul. Der Roboter dreht sich nun eine gewisse „Zeit“ in Richtung des eigenen Tores um den Ball sobald er ihn vorner erblickt hat von der richtigen Richtung anfahren zu können.

### has_ball_see_goal_in_front()

```
#define has_ball_see_goal_in_front()\
{\
    if(allowKick)allowKick--;\
    if(KICK_DISTANCE_OK)\
      if(SEE_BALL_IN_FRONT && !SEE_BALL_IN_FRONT_FAR)\
        if (SEE_GOAL_MIDDLE || (SEE_GOAL_LEFT && SEE_GOAL_RIGHT))\
        {\
            if(0 == allowKick)\
                {\
                    KICK\
                    allowKick = 50;\
                }\
        }\
}
```

Das Verhalten wird aktiv, wenn der Ball unter Kontrolle, das Tor sichtbar vor dem Roboter und der Abstand zum Tor gering genug ist. Es löst einen Schuß aus. Dieser kann erst nach einer kurzen Pause wiederholt werden. Dies soll verhindern, dass der Kicker dauerhaft auslöst, was die Motortreiber zu strak belasten würde.

### has_ball_goal_sideways()

```
#define has_ball_goal_sideways()\
{\
    prio = 5;\
  if(SEE_BALL_IN_FRONT/* && !SEE_GOAL_MIDDLE*/)\
  {\
    if (SEE_GOAL_RIGHT && !SEE_GOAL_LEFT)\
            SET_DD(90,SLOW_SPEED,-50,prio)/*BIG TURN*/\
    else if (SEE_GOAL_LEFT && !SEE_GOAL_RIGHT)\
            SET_DD(270,SLOW_SPEED,50,prio)/*BIG TURN*/\
  }\
}
```

Das Verhalten wird aktiv, wenn der Ball unter Kontrolle ist und das Tor zu erkennen ist, aber es noch nicht direkt vor dem Roboter ist. In diesem Fall richtet sich der Roboter zum Tor aus, indem er um den Ball slidet.

### has_ball_dribble()

```
#define has_ball_dribble()\
{\
    prio = 4;\
/*  if(SEE_GOAL_LEFT || SEE_GOAL_RIGHT || SEE_GOAL_MIDDLE)*/\
        if(SEE_BALL_IN_FRONT || SEE_BALL_LEFT || SEE_BALL_RIGHT)\
        {\
            if(DRIBBLE_RIGHT)\
                SET_DD(270,TURN_SPEED,30,prio) /*SLIDE RIGHT*/\
            else if(DRIBBLE_LEFT)\
                SET_DD(90,TURN_SPEED,-30,prio) /*SLIDE LEFT*/\
        }\
}
```

Das Verhalten wird aktiv, wenn der Ball unter Kontolle ist und der Ball einen der seitliche Fühler des Roboters berührt. Der Roboter richtet sich jetzt durch seitliche Bewegungen optimal zum Ball aus.

### has_ball_no_goal()

```
#define has_ball_no_goal()/*habe den ball und sehe das tor nicht*/\
{\
  if(SEE_BALL_IN_FRONT/* && !SEE_BALL_IN_FRONT_FAR*/) /*habe den
      ball*/\
  {\
    if (!SEE_GOAL_RIGHT)\
      if (!SEE_GOAL_LEFT)\
        if (!SEE_GOAL_MIDDLE) /*sehe eigenes tor nicht*/\
        {\
          if (!SEE_GOAL_BACK_LEFT) /*das gegnerische tor ist rechts
              -> bin an der linken wand*/\
            if(LEFT != lastSlide && !AM_I_IN_AN_EDGE)\
                        SET_DD(270,100,50,7)/*MOVE_ARROUND*/\
            else\
                        SET_DD(90,100,-50,7)/*MOVE_ARROUND*/\
            else  /*geg. tor ist links -> bin an der rechten wand*/\
                if(RIGHT != lastSlide && !AM_I_IN_AN_EDGE)\
                        SET_DD(90,100,-50,7)/*MOVE_ARROUND*/\
                else\
```

```
                                        SET_DD(270,100,50,7)  /*MOVE_ARROUND*/\
                        }\
    }\
}
```

Das Verhalten wird aktiv, wenn der Ball unter Konrtolle ist, aber kein Tor gesehen wird. Der Roboter fährt nun um den Ball, ohne dessen Position zu verändern. Die Richtung für die Drehung wird mit dem seitlichen Tor Sensor ermittelt, sodass hier der kürzere Weg bevorzugt wird.

# 3 Fazit

## 3.1 Zeitplanung

### 3.1.1 Matches

Es ist sehr wichtig schon frühzeitig Matches gegen andere Teams zu spielen, da erst dort echte Probleme auftauchen. In der Endphase sollten Matches zumindest stündlich ausgeführt werden.

### 3.1.2 Hard und Software Zeitplanung

Ein allgemeines Problem ist das sich während des Projektes die Hardware noch verändert. Dies kostet enorm Zeit und macht teilweise schon geschriebenen Code überflüssig. Gut geplante Hardware kann dies verhindern. Andererseits sollte man am Anfang den Code möglichst einfach schreiben, da der Code schwer zu debuggen ist. Beim Roboterbau sollte man sich ruhig die Zeit nehmen sehrt stabil zu bauen, die Reperaturen kosten später wesentlich mehr Zeit.

### 3.1.3 Hardware

Beim Basteln sollte höchst sorgfältig gearbeitet werden. Nichts ist ärgerlicher als eine gebrochene Lötstelle an einer unzugänglichen Stelle. Zudem ist Roboterfußball recht ruppig. Die Kräfte die auf alle Teile des Roboters während einem Spiel wirken, sollten nicht unterschätzt werden. Wir haben das Problem mit einer Schutzhülle aus Plexiglas gelöst. Damit waren wir sehr stabil und haben die Gefahr des Verhakens mit einem anderen Roboter minimiert. Der Nachteil an dieser Lösung ist das hohe Gewicht, was vorallem Geschwindigkeitseinbußen bedeutet.

Im laufe der Zeit sind uns viele Ideen gekommen, wie der Roboter zu verbessern ist. Am wichtigsten ist es die Platform weiter auszubauen und zu vereinfachen. Hierzu werden neue Motoren benötigt, die den auftretenen Kräften gewachsen sind. Die aktuell eingesetzten Motoren sind zwar stabil, jedoch gehen die Getriebe sehr schnell kaputt. Die eingesetzen Getriebe haben als „Sollbruchstelle“ ein Plastikzahnrad, was sich durch ein Metallzahnrad aus einem zweiten defekten Getriebe ersetzen lässt. Das macht die Getriebe wesentlich lauter, jedoch nur etwas stabiler.

Um den Roboter unanfälliger zu machen würden wir vorschlagen eine weitere Alu-Platte auf die erste zu bauen und die Distanz Sensoren mit Alu Winkeln zu befestigen.

Zudem sollte eine solid Rumdumsicht für die Ballsuche ermöglicht werden. Hier wäre es möglich mehrer IR-Sensoren auf einem Streifen Metall anzubringen und dieses auf der

unteren Ebene fest zu installieren.

Ein gutes Platzieren und befestigen des Controllerboard wäre mit einer zweiten Ebene auch einfacher. LEGO könnte fast komplett ferngehalten werden.

Das Controllerboard könnte einfach gegen ein Leistungsfähigeres ausgetauscht werden, auf dem dann auch einfaches Debuggen möglich wäre.

### 3.1.4 Software

Der von uns eingesetze Dual-Dynamics Ansatz ist zu empfehlen, da er an vielen Stellen den Code vereinfachen kann. Ob der massive Einsatz von Makros einen erheblichen Geschwindigkeitsvorteil gegenüber einer Methoden orientierten Version bringt, könnte über eine Messung mit einem Oszilloskop geprüft werden.

# Abbildungsverzeichnis