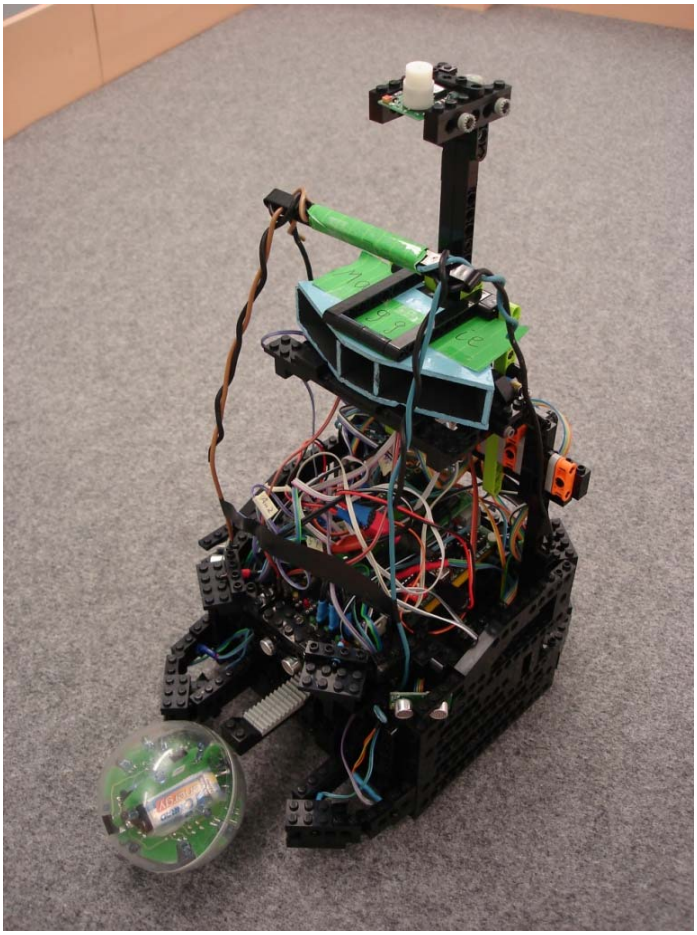
# Projekt

# Fussballroboter Maggie



**Jessica Walther**

**Marko Koplin**

**Christoph Paschen**

FACHHOCHSCHULE
BRANDENBURG

# Inhalt

# 1 Einleitung

Diese Dokumentation entstand im Rahmen des Projektes „Autonome Sehende Roboter“, welches im siebentem Fachsemester an der Fachhochschule Brandenburg ausgewählt werden konnte. Der ursprüngliche Projektinhalt befasste sich mit der Aufgabe, das RCUBE-System in Verbindung mit einem AKSEN-Board so zu programmieren, dass eine anspruchsvolle Auswertung und Verarbeitung durchgeführt werden konnte.

Im weiteren Rahmen wurde auch das diesjährige interne Messen der AKSEN-Board-einsetzenden Fachhochschulen in Deutschland vorbereitet und durchgeführt. Hierbei handelt es sich um einen kleinen Wettkampf, bei dem Fußballroboter auf Basis eines AKSEN-Boards konstruiert werden und am Tage des Wettkampfes gegeneinander antreten.

An unserer Fachhochschule fanden sich fünf Studenten des siebenten Fachsemesters, die unterschiedliche Ansätze verfolgten und so zwei mobile autonome Systeme konstruierten, die diese Aufgabe bestehen sollten. Die hier vorliegende Dokumentation stellt den Ansatz von Jessica Walther, Marko Koplin und Christoph Paschen vor, die den Fußballroboter Maggie entwarfen und bauten.

# 2 Lösungsansatz

Ziel war es einen äußerst robusten und relativ wartungsarmen Roboter zu entwickeln, bei dem die Akku-Packs ohne großen Aufwand austauschbar sein sollten. Auch sollte der Antrieb so umgesetzt werden, dass der Roboter wendig und dabei relativ schnell beweglich bleibt.

Die Strategie sollte so umgesetzt werden, dass zu aller erst der Ball gesucht und detektiert werden sollte. Im Anschluss daran, soll der Roboter zum Ball fahren und ihn in „Besitz“ bringen. Ist das geschehen, soll das Zieltor gesucht und auf dieses hingesteuert werden. Ist der Roboter in der Nähe des Tores könnte bereits ein Schuss abgegeben werden. – Zudem soll es, wenn möglich in beide Richtungen (nach vorne bzw. hinten) möglich sein, zu navigieren und den Ball zu führen/schießen.

Die durch das Regelwerk vorgegebenen Einschränkungen sollen dabei Beachtung finden.

# 3 Technik des Roboters

## 3.1 Überblick

Die Technik des Roboters ist im Laufe der Entwicklung immer komplexer geworden. Letztendlich waren so gut wie alle analogen und digitalen Anschlüsse auf dem AKSEN-Board belegt. Insgesamt wurden folgende Komponenten verbaut:

| Komponente | Anzahl |
|---|---|
| Motoren | 3 |
| IR-Empfänger | 19 |
| IR-Sender | 2 |
| IR-Fernsteuerungsempfänger | 3 |
| Ultraschallsensoren | 4 |
| Kompass | 1 |
| Tastensensor | 1 |

## 3.2 Sensoren

Da der Ball Infrarotstrahlen sendet, kommen IR-Empfänger zum Einsatz. Diese IR-Empfänger haben nur einen beschränkten Einfallswinkel. Um eine größtmögliche Erkennungsrate rund um den Roboter zu erreichen wurden besonders viele Sensoren verwendet. Die Verteilung ist in den folgenden beiden Abbildungen zu erkennen, wobei Abbildung 2 die höher liegende Etage darstellt. Jeder einzelne rote Pfeil symbolisiert einen Sensor.

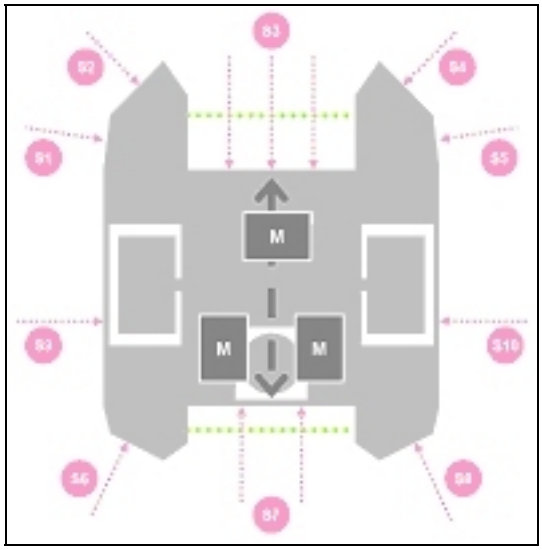



**Abbildung 1: Unterste Sensorebene**

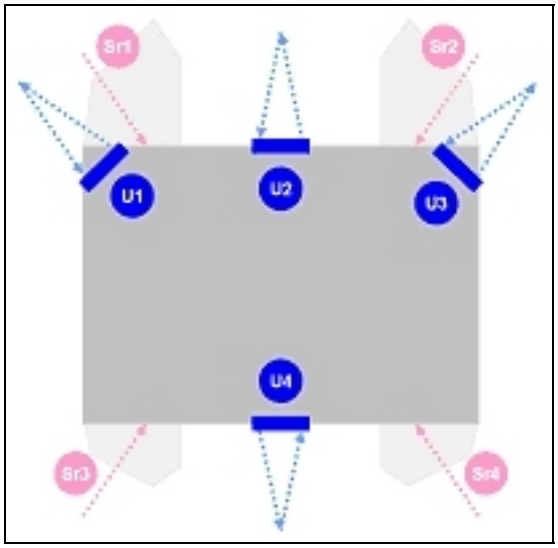



**Abbildung 2: Mittlere Sensorebene**

Im vorderen Bereich wurden drei und im hinteren zwei Sensoren zusammen geschaltet. Dies wurde über eine selbst angefertigte Platine erreicht, die die Sensoren mit einer „Oder“-Verknüpfung zu einem Signal zusammenführt (rechte Abbildung, blaue und rote Markierung). Dadurch können bessere Werte erzielt und Steckplätze auf dem AKSEN-Board gespart werden.

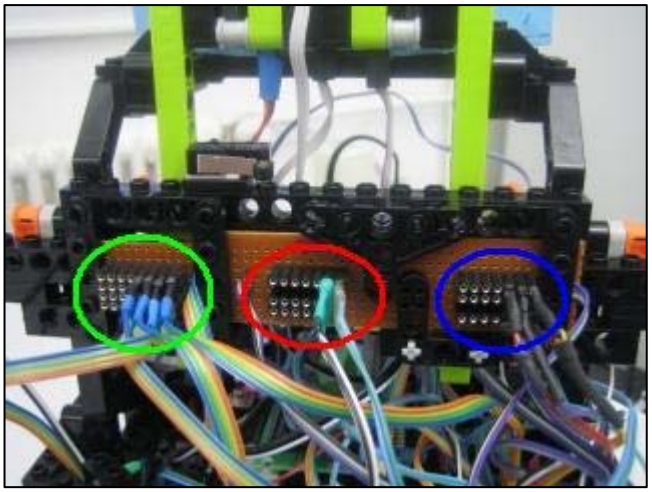


**Abbildung 3: Verknüpfung einzelner Sensorsignale**

Die Infrarotempfänger Sr1 bis Sr4 dienen der besseren Auswertung der unteren Sensoren, indem das Umgebungslicht mit eingerechnet wird. Dabei ist zu beachten, dass die oberen Empfänger nicht den Ball sehen dürfen.

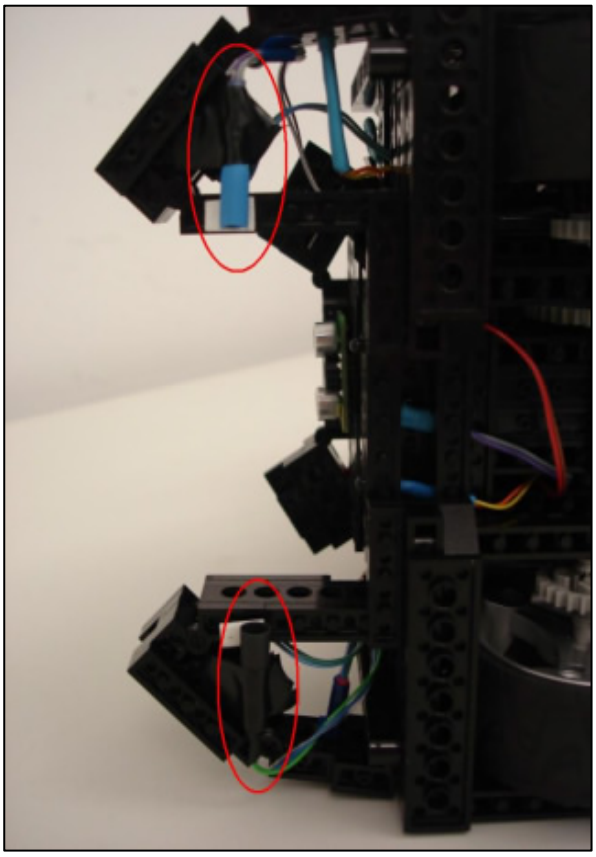


**Abbildung 4: Lichtschranke**

Um festzustellen, ob sich der Ball direkt vorne bzw. hinten am Roboter befindet, wurden Infrarot-Lichtschranken eingebaut. Diese bestehen aus jeweils einer Infrarot-Leuchte und einem Infrarot-Empfänger (rote Markierung auf linker Abbildung – Sicht von unten). Sobald der Ball zwischen die Lichtschranke rollt, bekommt der Empfänger kein Signal mehr. In der schematischen Darstellung stellt die grüne Punktelinie die Infrarot-Lichtschranke dar.

Um das Tor zu finden wurden Fernsteuerungsempfänger verbaut, die auf Infrarot-Licht mit einer bestimmten Frequenz reagieren (entweder 100 Hz oder 125 Hz).

Insgesamt wurden drei Stück dieser Sensoren in einem Holzrahmen verbaut und so ausgerichtet, dass der Front-Bereich des Roboters abgetastet werden kann. Der rechte und linke Sensor sind so angebracht, dass sie auf jeweils ihrer Seite begrenzt durch den

Holzrahmen ca. 85-90 Prozent des Sichtfeldes detektieren können. Der mittlere Sensor ist ebenfalls durch den Holzrahmen in seinem Sichtfeld eingeschränkt und somit auf einen recht schmalen Winkel genau in der Front des Roboters spezialisiert. Er stellt sicher, dass das Tor auch genau in Schussrichtung liegt. – Erhält keines der drei Empfänger ein Signal, kommt der Kompass zur groben Richtungsbestimmung zum Einsatz.

## *3.3 Motoren*

Für den Antrieb wurden zwei Motoren verwendet (für jedes Rad einen – grüne Markierung). Somit kann sich der Roboter auch auf der Stelle drehen. Die Übersetzung des Getriebes liegt bei 1:120. Durch die Verwendung eines Schneckenzahnrades konnte eine hohe Übersetzung auf wenig Raum erreicht werden.

Für die Schussvorrichtung befindet sich in der Mitte des Roboters eine Stoßvorrichtung auf einer Schiene (blaue Markierung), die mit Hilfe des dritten Motors (rote Markierung) vor und zurück bewegt werden kann.

Weil die Schussvorrichtung ebenfalls einen Motor benötigt und sich auf dem AKSEN-Board sich nur vier Motoranschlüsse befinden, wurden für den Antrieb nur zwei verwendet, obwohl im Chassis noch Platz für jeweils einen weiteren Motor pro Seite gewesen wäre.

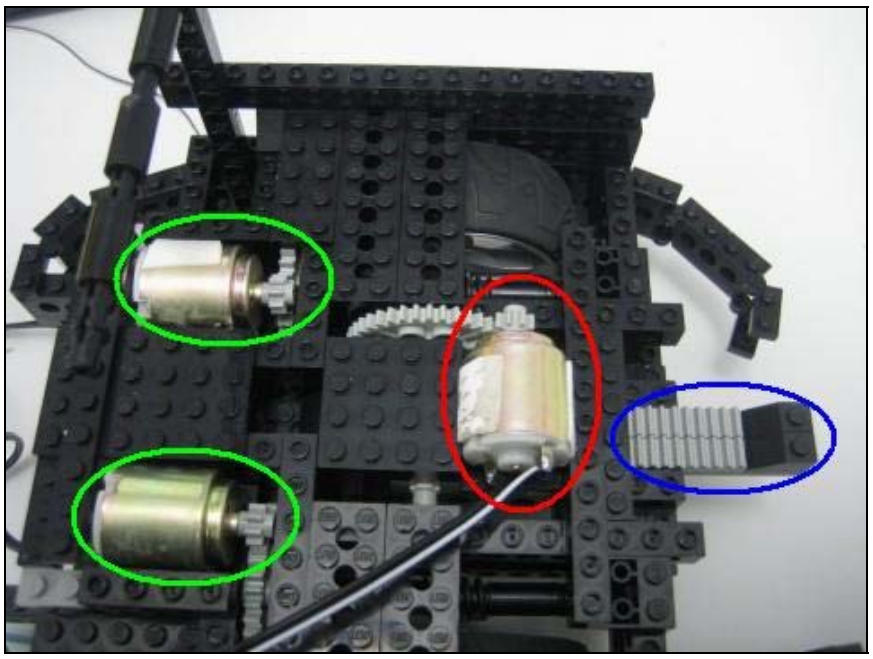


**Abbildung 5: Antriebsmotoren und Schussvorrichtung**

## *3.4 Weitere Komponenten*

Als weitere wichtige Komponenten wurden der bereits angedeutete Kompass und ein Tastensensor verbaut:

- Der Kompass dient zur besseren Orientierung. Dieser kann aber nur acht Richtungen unterscheiden und ist somit nicht genau. Er kann nur die grobe Ausrichtung bestimmen und ist somit alleine nicht zur Torsuche geeignet.
- Der Tastensensor dient nur einem Zweck: Sofortiges Anhalten aller Aktionen. Das hat sich beim Testen als sehr hilfreich herausgestellt, denn es ist nicht immer leicht, an den Resetknopf des AKSEN-Boards heranzukommen.

# 4 Ultraschallsensoren (I²C)

Um auch nahe Hindernisse, die sich nicht unmittelbar direkt am Roboter befinden, detektieren zu können, wurden Ultraschallsensoren eingesetzt. – Speziell wurden hier die Sensormodule SRF10 verwendet. – Die Module sind für Entfernungsmessungen von 4 bis maximal 600 Zentimeter ausgelegt und wurden bei Maggie durch eine erhöhte Messfrequenz so modifiziert, dass die Umgebung bis ca. 40 Zentimetern zuverlässig wahrgenommen werden kann.

Da die Sensormodule für den Anschluss an einen $I^2C$-Bus vorgesehen sind, müssen diese, von der Werkseinstellung abweichend, mit einer individuellen Adresse programmiert werden, um Konflikte auf dem Bus zu vermeiden und diese dann gezielt abfragen zu können.

Im Fall von Maggie wurden insgesamt vier Module eingesetzt (vgl. Abbildung 2 – U1-U4) und auf der zweiten Sensorebene so angeordnet, dass drei den vorderen und einer den hinteren Bereich des Roboters „aushorchen“ können. Der Anschluss erfolgte über eine Steckerleiste (vgl. Abbildung 3 – grüne Markierung), die speziell für den $I^2C$-Bus angefertigt wurde und mit einem Doppelstiftstecker am AKSEN-Board konnektiert ist.

Für die Abfrage der einzelnen Module von Seiten des AKSEN-Boards wurde eine $I^2C$-Bibliothek verwendet, die bei der Kompilierung des Gesamtprojektes zur Verfügung stehen muss.

# 5 Software

*Hinweis*: Der ausführliche Quellcode ist in Anlage 1 hinterlegt.

## *5.1 Hauptablauf*

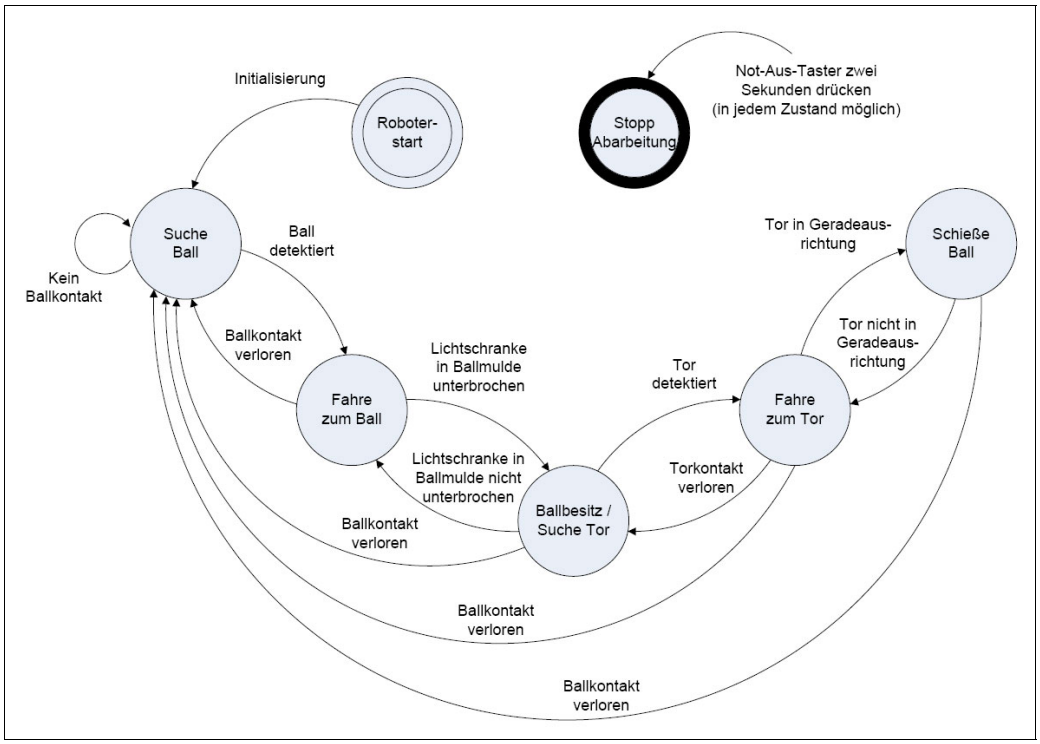



**Abbildung 6: Zustandsdiagramm der möglichen Verhalten**

Das Zustandsdiagramm in Abbildung 5 zeigt das Verhalten des Roboters während des Fußballspiels. Zuerst muss der Roboter gestartet und der Kompass initialisiert werden. Dann beginnt die Ballsuche (vgl. 5.3 Ballsuche) indem ständig die Infrarot-Empfänger nacheinander abgefragt werden. Wird der Ball detektiert, fährt der Roboter in diese Richtung, verliert er den Ball beginnt wieder die Ballsuche. Ist der Roboter im Besitz des Balls, d. h. die Lichtschranke vorne bzw. hinten wurde unterbrochen, beginnt die Torsuche (vgl. 5.4 Torsuche) mit Hilfe des Kompasses und der Infrarot-Fernsteuerungsempfänger. Empfangen diese das Signal des Tores wird sich entsprechend ausgerichtet und es erfolgt der Schuss bei Signalempfang des mittleren Sensors (vgl. 5.5 Ball schießen).

## *5.2 Allgemeine Festlegung*

Um die einzelnen Infrarot-Sensoren und deren Zustände im Zusammenhang besser zu handhaben werden diesen, bestimmte Werte zugeordnet: Zuerst wird der ermittelte Wert der

Sensoren mit dem Umgebungslicht (Sr1 bis Sr4) verglichen. – Vergleichen heißt, dass der Wert des Sensors von dem Wert des Umgebungslichtsensors subtrahiert wird. Ist der Ball nicht zu sehen, dann sind beide Werte nahezu identisch und das Ergebnis der Subtraktion ist näherungsweise Null. Wird der Ball detektiert, ist der Wert des Sensors, der ihn erkennt, kleiner als der Wert des Umgebungslichts. Die Subtraktion liefert somit ein Ergebnis größer als Null. Je größer der Wert ist, desto näher ist der Ball am Roboter. Übersteigt dieser Wert den SENSOR_REFERENZ_WERT von sechs, dann wird diesem ein bestimmter Wert zugeordnet. Der Wert entspricht der Zweier-Potenz seiner Nummer. Das heißt der Sensor S1 hat den Wert 1, Sensor S2 hat den Wert 2, Sensor S3 hat den Wert 4, usw. Nachdem alle Sensoren ausgewertet sind, werden die Werte der einzelnen Sensoren zusammenaddiert.

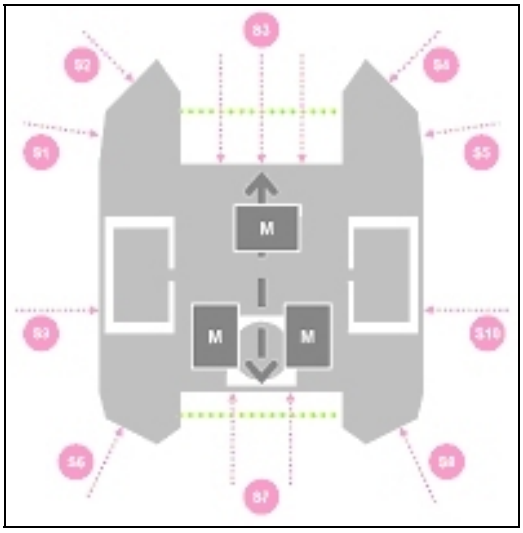



**Abbildung 7: Unterste Sensorebene**

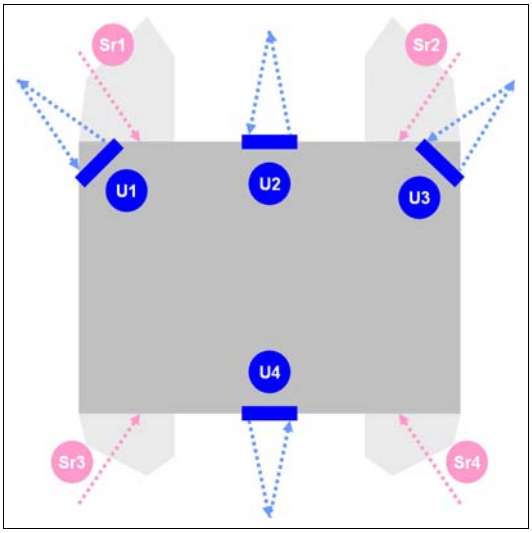



**Abbildung 8: Mittlere Sensorebene**

## *5.3 Ballsuche*

Die Ballsuche wird in drei Bereiche unterteilt: die Seiten, die Umgebung vorne und die Umgebung hinten. Die Seiten werden in der Hauptroutine überprüft. Wird dort nichts gefunden, wird hinten überprüft und dann vorne. Die Auswertung für vorne und hinten erfolgt wie unter 5.2 Allgemeine Festlegung erläutert. Nachfolgend wird die Ballsuche erläutert, wenn sich der Ball hinter dem Roboter befindet.

| Wert | S6 | S7 | S8 | Aktion | V1 | V2 | D1 | D2 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Geradeaus rückwärts fahren | FAST | FAST | 1 | 0 |
| 2 | 0 | 1 | 0 | Geradeaus rückwärts fahren | FAST | FAST | 1 | 0 |
| 5 | 1 | 0 | 1 | Geradeaus rückwärts fahren | FAST | FAST | 1 | 0 |
| 7 | 1 | 1 | 1 | Geradeaus rückwärts fahren | FAST | FAST | 1 | 0 |
| 4 | 1 | 0 | 0 | Links rückwärts fahren | FAST | STOP | 1 | 0 |
| 6 | 1 | 1 | 0 | Links rückwärts fahren | FAST | STOP | 1 | 0 |
| 1 | 0 | 0 | 1 | Rechts rückwärts fahren | STOP | FAST | 1 | 0 |
| 3 | 0 | 1 | 1 | Rechts rückwärts fahren | STOP | FAST | 1 | 0 |

**Abbildung 9: Zustände der Sensoren hinten**

Am hinteren Teil des Roboters sind fünf Infrarot-Empfänger angebracht; links, rechts, zwei mit „Oder“ verschaltete in der Mitte und zwei Sensoren für das Umgebungslicht etwa zehn Zentimeter darüber, die jeweils rechts und links in die Richtung der unteren Sensoren ausgerichtet sind. Entsprechend der Abbildung 9 wird die Motorsteuerung mit der Geschwindigkeit (V1 und V2 – Velocity) und der Drehrichtung (D1 und D2 – Direction) der Motoren ausgeführt.

*Beispiel*: Die Sensoren S6 und S7 detektieren den Ball, so wird für den Sensor S6 der Wert 4 und für S7 der Wert 2 zugewiesen. Daraus ergibt sich nach der Addition der Wert 6. Nun wird in der Methode „navigierenBACK(int value)“ im switch-Block (Fallunterscheidung) der Fall 6 ausgewählt und die Motoreinstellung laut Tabelle eingesetzt.

```
void navigierenBACK(int value){
    zaehler1 = 0;
    if(zaehler2 == 2){
        setzeSchussMotoreinstellung(MOTORRICHTUNG1,FAST);
    }
    zaehler2++;
    setzeMotorPower(STOP);

    switch(value){
    case 0:
    case 2:
    case 5:
    case 7:
        setzeMotoreinstellungen(1,0,FAST,FAST);
        ausgabe("geradeBACK");
        break;
    case 4:
    case 6:
        setzeMotoreinstellungen(1,0,FAST, STOP);
        ausgabe("linksBACK");
        break;
    case 1:
    case 3:
        setzeMotoreinstellungen(1,0,STOP,FAST);
        ausgabe("rechtsBACK");
        break;
    }
}
```

**Abbildung 10: Quelltext "navigierenBACK()"**

D. h., die Motorgeschwindigkeit von Motor1 wird auf FAST und für Motor2 auf STOP gesetzt und die Drehrichtung auf Eins beziehungsweise Null. Somit fährt der Roboter links rückwärts, da sich das eine Rad dreht, welches vom Motor1 angesteuert wird und das andere stehen bleibt.

Vorher wird in dieser Methode die Schussvorrichtung in die entsprechende Position gebracht, wenn sich die Fahrtrichtung geändert hat.

## 5.4 Torsuche

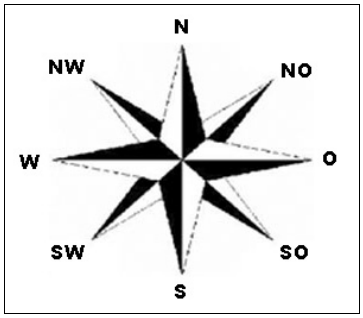



**Abbildung 11: Windrose**

| Himmelsrichtung | von | bis |
|---|---|---|
| Nordwest | 31 | 55 |
| Norden | 56 | 75 |
| Nordost | 76 | 105 |
| Osten | 106 | 126 |
| Südost | 127 | 145 |
| Süden | 146 | 165 |
| Südwest | 166 | 185 |
| Westen | 186 | 200 |

**Abbildung 12: Kompasswerte**

Die Auswahl der Torfrequenz erfolgt über den ersten Dipschalter, 1 bedeutet, dass die Infrarot-Fernsteuerungsempfänger eine Frequenz von 100 Hz und 0 heißt eine Frequenz von 125 Hz empfangen. Jeder der Empfänger kann nur bei einer bestimmten Ausrichtung des Roboters das Signal des Tor-Beacons empfangen.

Ein Schuss erfolgt nur dann, wenn der mittlere Empfänger das Signal empfängt: Sehen der linke beziehungsweise der rechte Empfänger den Beacon, dann dreht sich der Roboter langsam in Richtung des Tores. Wird kein Signal wahrgenommen, kommt es zur Auswertung des Kompasses. Dieser kann in die acht in Abbildung 11 aufgeführten Himmelsrichtungen eingeteilt werden. Der Kompass gibt entsprechend der Ausrichtung einen Wert zurück, der dann ausgewertet werden muss.

Die Abbildung 12 zeigt die entsprechenden Wertebereiche für die Himmelsrichtungen: Beim Rückwärtsfahren und im Besitz des Balles, wird sich nur entsprechend dem Kompass nach Süden ausgerichtet und dann auf Verdacht geschossen, da nach hinten kein IR-Fernsteuerungsempfänger angebracht ist.

| Wert | Aktion | V1 | V2 | D1 | D2 |
|---|---|---|---|---|---|
| South | Auf der Stelle nach links drehen | STOP | FAST | 0 | 1 |
| North | Geradeaus fahren | FAST | FAST | 0 | 1 |
| East | Langsam nach links drehen | FAST | SLOW | 0 | 1 |
| West | Langsam nach rechts drehen | SLOW | FAST | 0 | 1 |
| NorthEast | Nach links drehen | FAST | MED | 0 | 1 |
| NorthWest | Nach rechts drehen | MED | FAST | 0 | 1 |
| SouthEast | Stark nach links drehen | FAST | FAST | 0 | 0 |
| SouthWest | Stark nach rechts drehen | FAST | FAST | 1 | 1 |

**Abbildung 13: Auswertung des Kompass**

## 5.5 Ball schießen

Die Ausführung des Schussmechanismus erfolgt bei Erkennung des entsprechenden Tores wie unter 5.4 Torsuche erläutert.

Während der Fortbewegung des Roboters wird die Schussvorrichtung mit der Methode „setzeSchussMotoreinstellung()“ entweder nach vorne beim Rückwärtsfahren oder nach hinten beim Vorwärtsfahren ausgerichtet. Wurde das Tor gefunden und sich entsprechend ausgerichtet, wird die Methode „setzeSchussMotoreinstellungTyp()“ aufgerufen und mit der Übergabe von Null festgelegt, das nach vorne geschossen wird und bei jeder anderen Zahl nach hinten. Dazu wird der Motor für die Steuerung der Schussvorrichtung jeweils 150 Millisekunden aktiviert, damit der Ball weggestoßen wird.

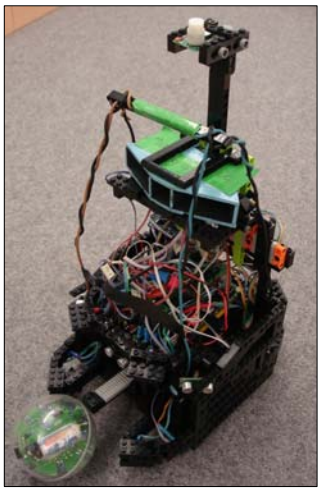


**Abbildung 14: Maggie mit Ball in Front**

## 5.6 Wandkollision vermeiden

Um eine Kollision mit der Wand zu vermeiden, wurden Ultraschallsensoren (U1 bis U4) integriert. – Die Lage und Ausrichtung dieser ist in Abbildung 8 und die Funktionsweise in Abschnitt 4 verdeutlicht. – Diese werden nach dem gleichem Prinzip wie unter Abschnitt 5.2 mit Hilfe der verschiedenen Zustände ausgewertet. Damit der Roboter sich nicht von den Wänden irritieren lässt, wenn dort ein Ball liegt, wird diese Funktion nur aktiviert, wenn er den Ball sucht und weder an den Seiten, hinten noch vorne den Ball detektiert.

## 5.7 Motoreinstellung und Stressfaktor

Um den Roboter in die entsprechenden Richtungen zu bewegen, muss die Methode „setzeMotoreinstellungen()“ genutzt werden. Die beiden Motorrichtungen und die beiden Geschwindigkeiten müssen mit übergeben werden. Damit werden die Motorrichtung und die Motorgeschwindigkeit für beide Motoren in die entsprechenden Methoden eingefügt.

Um ein zu langes Verharren zu verhindern, wurde ein Stressfaktor eingeführt. Dieser wird bei gleich bleibenden Motoreinstellungen erhöht und bei Überschreiten eines Grenzwertes eine Zufallszahl aus der aktuellen Zeit seit Drücken des RESET-Buttons und der Modulofunktion von 32 bei Vorwärtsbewegung und 8 bei Rückwärtsbewegung ermittelt und an „navigierenFRONT()“ beziehungsweise „navigierenBACK()“ weitergegeben. Anschließend wird die Zeit wieder auf Null gesetzt.

# 6 Probleme, Verbesserungsmöglichkeiten, Fazit

## *6.1 Probleme*

Ein Hauptproblem des Roboters ist die Geschwindigkeit. Zwar ist diese Art des Getriebes sehr gut, aber für den Einsatz zu langsam und daher nicht akzeptabel. Zudem kam die schlechte Sichtbarkeit des Balls hinzu, was auch an der zu langsamen Verarbeitung des Programms liegen kann. Dort würde sich vielleicht das Multithreading anbieten, was aber auf Grund des Zeitmangels nicht wieder aufgegriffen wurde.

## *6.2 Verbesserungsmöglichkeiten*

Die Verbesserungsmöglichkeiten sind hauptsächlich in der schnelleren und parallelen Verarbeitung der Abläufe zu sehen. Des Weiteren würden sich mehrere Motoren anbieten, damit der Roboter in erster Linie schnell zum Ball fährt. Zusätzlich würde sich die Verstärkung der Infrarot-Empfänger als Ausbaustufe positiv zeigen, wie wir es beim Wettkampf bei einem gegnerischen Roboter sehen konnten.

Ebenfalls verschaffen die „omni wheels“ eine Aufbesserung des Roboters, da diese leicht in alle Richtungen fahren können. Ferner müsste die Torausrichtung und der Einsatz des Kompasses optimiert werden, damit der Schuss definitiv das Tor trifft. Hinzu kommt die Möglichkeit eines Ballkäfigs, der den Ball zumindest für einige Sekunden vor dem Gegner schützt und diesen besser kontrollieren kann.

## *6.3 Fazit*

Dieses Projekt war Alles in Allem ein tolles Erlebnis und mal eine etwas andere Herausforderung. Trotz dem die Enttäuschung über das teilweise doch sehr frühe Ausscheiden im Wettkampf am Ende doch schon vorhanden war, zeigte es uns doch wieder einmal wie wichtig Teamarbeit und Variation im Entwicklungsprozess sein können.

Zu Beginn waren wir durch das Wissen aus dem vorangegangenen Projekt im sechsten Semester mit den wichtigsten Grundlagen und auch schon ein paar Verbesserungsideen gewappnet und hätten im Grunde so die Aufgabenstellung recht zügig bearbeiten können. Trotzdem kamen wir nur recht zäh voran und zwischendurch verzögerten sich auch einige wichtige Schritte. Am Ende entstand eine doch recht passable Lösung, die sich aber nicht mit der Schnelligkeit der anderen Teams messen konnte. Hierzu hätten wir schon viel früher mit unserem anderen Team „Zwischenwettkämpfe“ abhalten sollen.

# 7 Anlage 1 – Quellcode

```
//Standard-Include-Files
#include <stdio.h>
#include <regc515c.h>
#include "maggie.h"

//Diese Include-Datei macht alle Funktionen der
//AkSen-Bibliothek bekannt.
//Unbedingt einbinden!
#include <stub.h>
#include "../I2C/SRF10/SRF10.h"

//Deklaration der Variablen
#define MOTOR1 1      //Motor1
#define MOTOR2 0      //Motor2

#define MOTORSCHUSS 2 //Schussmotor

//definierte Geschwindigkeiten
#define STOP 0
#define SLOW 5
#define MEDIUM 7
#define MFAST 9
#define FAST 10

//Motorrichtung vorwaerts und rueckwaerts fuer Schuss
#define MOTORRICHTUNG0 0
#define MOTORRICHTUNG1 1

//Torfrequenz, maximaler Fehler und Schwellwert
#define GOAL_FREQUENCY_100 5
#define GOAL_FREQUENCY_125 4
#define IR_MAX_ERROR 4
#define THRESHOLD     12

//Referenzwerte
#define SENSOR_REFERENZ_WERT 6
#define BALLNAEHE 247
#define WANDNAEHE 12

//Deklaration der Methoden
void AksenMain(void);
int UmgebungPruefenVorne();
int UmgebungPruefenHinten();
int UmgebungAufWandPruefen();
int getNotausWert();
int getSonarwertVorne();
int getSonarwertHinten();
```

```
int getSonarwertLinks();
int getSonarwertRechts();
void kompassAuswerten();
void kompassAuswertenHinten();
void navigierenFRONT(int value);
void navigierenBACK(int value);
void setzeMotoreinstellungen(int mot1Richt, int mot2Richt, int mot1Geschw, int mot2Geschw);
void setzeMotorPower(int motGeschw);
void setzeSchussMotoreinstellung(int motRicht, int motGeschw);
void setzeSchussMotoreinstellungTyp(int typ);
unsigned char getTorwert(int digitalausgang, unsigned char frequence);
void findeTor();
void findeTorHinten();
void ShiftHistory(int *sensorHistory);
void LedsON();
void LedsOFF();
void ausgabe(char *value);
void ausgabeInt(int value);
void vermeideWand(int value);

//Kompassrichtungen
enum compassDirection
{
       NORTH = 0,                   // Tor liegt vor dem Roboter
       NORTHEAST = 1,               // Tor liegt vorne rechts vom Roboter
       EAST = 2,                    // Tor liegt rechts vom Roboter
       SOUTHEAST = 3,               // Tor liegt hinten rechts vom Roboter
       SOUTH = 4,                   // Tor liegt hinterm Roboter
       SOUTHWEST = 5,               // Tor liegt hinten rechts vom Roboter
       WEST = 6,                    // Tor links vom Roboter
       NORTHWEST = 7                // Tor liegt vorne links vom Roboter
};

enum compassDirection getCompassDirection();
enum compassDirection getCompassDirectionSouth();

//>>>>> Beginn globale Variablen (eigentlich lokal AksenMain) <<<<<//
//Sensorik Vorne
int irVRechtsAussen;                // VRA
int irVRechts;                      // VR
int irVRechtsOben;                  // VRO
int irVMitte;                       // VM
int irVLinksOben;                   // VLO
int irVLinks;                       // VL
int irVLinksAussen;                 // VLA

//Sensoren Links/Rechts
int irLinks;                        // L
int irRechts;                       // R
```

```
//Umgebungslicht Vorne
unsigned int umgebungslichtRechtsAussen;
unsigned int umgebungslichtRechts;
unsigned int umgebungslichtMitte;
unsigned int umgebungslichtLinks;
unsigned int umgebungslichtLinksAussen;

unsigned int umgebungslichtR;
unsigned int umgebungslichtL;

int aktiveSenHistoryFRONT[5] = {-1,-1,-1,-1,-1};


//Sensorik Hinten
int irHRechtsAussen;                // HRA
int irHRechtsOben;                  // HRO
int irHMitte;                       // HM
int irHLinksOben;                   // HLO
int irHLinksAussen;                 // HLA

//Umgebungslicht Hinten
unsigned int umgebungslichtRechtsHinten;
unsigned int umgebungslichtMitteHinten;
unsigned int umgebungslichtLinksHinten;

int aktiveSenHistoryBACK[3] = {-1,-1,-1};

//Sonarwerte fuer Wandvermeidung
unsigned int sonarwertVorne;        // SoV
unsigned int sonarwertHinten;       // SoH
unsigned int sonarwertLinks;        // SoL
unsigned int sonarwertRechts;       // SoR

int aktiveSonarHistory[3] = {-1,-1,-1};

int notaus = 0;
int zaehler1 = 0;
int zaehler2 = 0;

int letzterMotorbefehl1 = 0;
int letzterMotorbefehl2 = 0;
int letzterMotorbefehl3 = 0;
int letzterMotorbefehl4 = 0;

int stressfaktor = 0;

//>>>>> Ende globale Variablen <<<<<//
```

```
//Hauptprogrammroutine
void AksenMain(void) {
	// Endlosschleife bis der Notausschalter betaetigt wird
	while(getNotausWert()) {

		int ballschranke = 0;
		int ballschrankeHinten = 0;

		int umgebung_vorn = 0;
		int umgebung_hinten = 0;

		//Sensorik Links/Rechts
		irLinks = analog(3);
		irRechts = analog(2);

		//Umgebungslicht Seiten
		umgebungslichtL = ((irVLinksOben+irHLinksOben)/2 > irLinks) ?
((irVLinksOben+irHLinksOben)/2)  - irLinks :0;
		umgebungslichtR = ((irVRechtsOben+irHRechtsOben)/2 > irRechts) ?
((irVRechtsOben+irHRechtsOben)/2) - irRechts :0;

		//ausgabeInt(UmgebungPruefenVorne());

		//Lichtschranken an
		led(3,1);
		led(2,1);
		sleep(10);
		ballschranke = digital_in(10);
		ballschrankeHinten = digital_in(12);

		if((ballschranke == 1) || (ballschrankeHinten == 1)) {
			if(ballschranke == 1){
				findeTor();
			}

			if(ballschrankeHinten == 1){
				findeTorHinten();
			}

		//Lichtschranken aus
		led(3,0);
		led(2,0);
		} else {
			umgebung_hinten = UmgebungPruefenHinten();
			umgebung_vorn = UmgebungPruefenVorne();

			if(UmgebungAufWandPruefen() != 0 && (umgebung_vorn == 0 ||
umgebung_hinten == 0)) {
				if(getSonarwertHinten()<WANDNAEHE){
					setzeMotoreinstellungen(0,1,FAST,FAST);
				}else{
```

```
                                        vermeideWand(UmgebungAufWandPruefen());
                                }
                        } else {
                                if((umgebung_hinten == 0) &&
                                        (umgebungslichtL>SENSOR_REFERENZ_WERT)){
                                setzeMotoreinstellungen(0,0,FAST,FAST);
                                ausgabe("LEFT");
                                }
                                if((umgebung_hinten == 0) &&
                                        (umgebungslichtR>SENSOR_REFERENZ_WERT)){
                                setzeMotoreinstellungen(1,1,FAST,FAST);
                                ausgabe("RIGHT");
                                }

                                if((umgebung_vorn==0) && (umgebung_hinten!= 0)) {
                                        navigierenBACK(umgebung_hinten);
                                } else {
                                        navigierenFRONT(umgebung_vorn);
                                }
                        }
                }
        }

        setzeMotoreinstellungen(1,1,STOP,STOP);
        LedsOFF();
}


//Dipschalter fuer Notaus
int getNotausWert(){
        notaus = digital_in(15);
        return notaus;
}

//Methoden fuer die Sonarwerte
int getSonarwertVorne(){
        sonarwertVorne = getDistance(0xE6);
        sleep(10);
        return sonarwertVorne;
}

int getSonarwertHinten(){
        sonarwertHinten = getDistance(0xE4);
        sleep(10);
        return sonarwertHinten;
}

int getSonarwertLinks(){
        sonarwertLinks = getDistance(0xE2);
        sleep(10);
        return sonarwertLinks;
}
```

```
int getSonarwertRechts(){
	sonarwertRechts = getDistance(0xE0);
	sleep(10);
	return sonarwertRechts;
}

//Methode, um zu pruefen wo sich ein Hindernis (Wand) befindet
int UmgebungAufWandPruefen(){
	aktiveSonarHistory[0] = 0;

	if(getSonarwertLinks()<WANDNAEHE) aktiveSonarHistory[0] = aktiveSonarHistory[0] + 4;
	if(getSonarwertVorne()<WANDNAEHE) aktiveSonarHistory[0] = aktiveSonarHistory[0] + 2;
	if(getSonarwertRechts()<WANDNAEHE) aktiveSonarHistory[0] = aktiveSonarHistory[0] +
1;

	return aktiveSonarHistory[0];
}

//Detektion des Balls mit den vorderen Sensoren
int UmgebungPruefenVorne(){

	//Sensorik Vorne
	irVRechtsAussen = analog(12);
	irVRechts = analog(13);
	irVRechtsOben = analog(11); //Umgebungslicht
	irVMitte = analog(0);
	irVLinksOben = analog(8);  //Umgebungslicht
	irVLinks = analog(10);
	irVLinksAussen = analog(9);



	//wenn Umgebungslicht > Wert, dann subtrahiere und bergebe Wert,
	//andernfalls auf 0 setzen (kein Hindernis)
	//Umgebungslichtmessung Vorne
	umgebungslichtLinksAussen = (irVLinksOben > irVLinksAussen) ? irVLinksOben -
irVLinksAussen : 0;
	umgebungslichtLinks = ( irVLinksOben > irVLinks ) ? irVLinksOben - irVLinks : 0;
	umgebungslichtMitte = (((irVRechtsOben + irVLinksOben)/2) > irVMitte) ?
(((irVRechtsOben + irVLinksOben)/2)- (irVMitte)) : 0;
	umgebungslichtRechts = ( irVRechtsOben > irVRechts ) ? irVRechtsOben - irVRechts :
0;
	umgebungslichtRechtsAussen = (irVRechtsOben > irVRechtsAussen) ? irVRechtsOben -
irVRechtsAussen : 0;



	//Ball_FRONT
	aktiveSenHistoryFRONT[0] = 0;
```

```
	if (umgebungslichtLinksAussen>SENSOR_REFERENZ_WERT) aktiveSenHistoryFRONT[0] =
aktiveSenHistoryFRONT[0] + 16;
	if (umgebungslichtLinks>SENSOR_REFERENZ_WERT) aktiveSenHistoryFRONT[0] =
aktiveSenHistoryFRONT[0] + 8;
	if (umgebungslichtMitte>SENSOR_REFERENZ_WERT*2) aktiveSenHistoryFRONT[0] =
aktiveSenHistoryFRONT[0] + 4;
	if (umgebungslichtRechts>SENSOR_REFERENZ_WERT) aktiveSenHistoryFRONT[0] =
aktiveSenHistoryFRONT[0] + 2;
	if (umgebungslichtRechtsAussen>SENSOR_REFERENZ_WERT) aktiveSenHistoryFRONT[0] =
aktiveSenHistoryFRONT[0] + 1;

	return aktiveSenHistoryFRONT[0];
}

//Detektion des Balls mit den hinteren Sensoren
int UmgebungPruefenHinten(){
	//Sensorik Hinten
	irHRechtsAussen = analog(6);
	irHRechtsOben = analog(7);  //Umgebungslicht
	irHMitte = analog(1);
	irHLinksOben = analog(5);   //Umgebungslicht
	irHLinksAussen = analog(4);

	//wenn Umgebungslicht > Wert, dann subtrahiere und bergebe Wert, andernfalls auf 0
setzen (kein Hindernis)
	//Umgebungslichtmessung Hinten
	umgebungslichtLinksHinten = ( irHLinksOben > irHLinksAussen ) ? irHLinksOben -
irHLinksAussen : 0;
	umgebungslichtMitteHinten = ((irHRechtsOben + irHLinksOben)/2> irHMitte) ?
((irHRechtsOben + irHLinksOben)/2)- irHMitte : 0;
	umgebungslichtRechtsHinten = ( irHRechtsOben > irHRechtsAussen ) ? irHRechtsOben -
irHRechtsAussen : 0;

	//Ball BACK
	aktiveSenHistoryBACK[0] = 0;

	//Sensor-berprfung Hinten:
	if (umgebungslichtLinksHinten>SENSOR_REFERENZ_WERT) aktiveSenHistoryBACK[0] =
aktiveSenHistoryBACK[0] + 4;
	if (umgebungslichtMitteHinten>SENSOR_REFERENZ_WERT*1.5) aktiveSenHistoryBACK[0] =
aktiveSenHistoryBACK[0] + 2;
	if (umgebungslichtRechtsHinten>SENSOR_REFERENZ_WERT) aktiveSenHistoryBACK[0] =
aktiveSenHistoryBACK[0] + 1;

	return aktiveSenHistoryBACK[0];
}
```

```
//die acht moeglichen Faelle, wo sich eine Wand befinden kann und
//die neuen Motoreinstellungen
void vermeideWand(int value){

	switch(value){
	case 0:
		ausgabe("nichts");  //nichts
		break;
	case 1:
		setzeMotoreinstellungen(0,0,SLOW,FAST);
		ausgabe("rechts Wand");  //leicht nach links fahren
		break;
	case 2:
		setzeMotoreinstellungen(1,1,FAST,SLOW);
		ausgabe("vorne Wand");  // leicht nach rechts fahren
		break;
	case 3:
		setzeMotoreinstellungen(0,0,MEDIUM,MEDIUM);
		ausgabe("rechts,mitte Wand");  // nach links fahren
		break;
	case 4:
		setzeMotoreinstellungen(1,1,FAST,SLOW);
		ausgabe("links Wand");  // leicht nach rechts fahren
		break;
	case 5:
		setzeMotoreinstellungen(0,1,MEDIUM,MEDIUM);
		ausgabe("rechts, links Wand");  // nach links fahren
		break;
	case 6:
		setzeMotoreinstellungen(1,1,MEDIUM,MEDIUM);
		ausgabe("links, mitte Wand");  // nach rechts fahren
		break;
	case 7:
		setzeMotoreinstellungen(1,0,FAST,SLOW);
		ausgabe("links rueckwaerts");  // nach links rckwrts fahren
		break;
	}
}

//Methode, um auf die Werte des Kompass zu reagieren beim Vorwaertsfahren
void kompassAuswerten() {

	//Kompasswert auslesen
	enum compassDirection kompass;
	kompass = getCompassDirection();

	switch(kompass) {
		case NORTH:
			setzeMotoreinstellungen(0,1,FAST,FAST);
			ausgabe("Norden");
			break;
```

```
		case NORTHEAST:
			setzeMotoreinstellungen(0,1,FAST,MEDIUM);
			ausgabe("Nordosten");
			break;
		case EAST:
			setzeMotoreinstellungen(0,1,FAST,SLOW);
			ausgabe("Osten");
			break;
		case SOUTHEAST:
			setzeMotoreinstellungen(0,0,FAST,FAST);
			ausgabe("Suedosten");
			break;
		case SOUTH:
			setzeMotoreinstellungen(0,1,STOP,FAST);
			ausgabe("Sueden");
			break;
		case SOUTHWEST:
			setzeMotoreinstellungen(1,1,FAST,FAST);
			ausgabe("Suedwesten");
			break;
		case WEST:
			setzeMotoreinstellungen(0,1,SLOW,FAST);
			ausgabe("Westen");
			break;
		case NORTHWEST:
			setzeMotoreinstellungen(0,1,MEDIUM,FAST);
			ausgabe("Nordwesten");
			break;
		default:
			ausgabe("???");
			break;
	}
}

//Methode, um auf die Werte des Kompass zu reagieren beim Rueckwaertsfahren
void kompassAuswertenHinten(){
	//Kompasswert auslesen
	enum compassDirection kompass;
	kompass = getCompassDirection();

	switch(kompass) {
		case NORTH:
			setzeMotoreinstellungen(1,0,FAST,STOP);
			ausgabe("Norden");
			break;
		case NORTHEAST:
			setzeMotoreinstellungen(0,0,FAST,FAST);
			ausgabe("Nordosten");
			break;
```

```
		case EAST:
			setzeMotoreinstellungen(1,0,FAST,SLOW);
			ausgabe("Osten");
			break;
		case SOUTHEAST:
			setzeMotoreinstellungen(1,0,FAST,MEDIUM);
			ausgabe("Suedosten");
			break;
		case SOUTH:
			ausgabe("Sueden");
			break;
		case SOUTHWEST:
			setzeMotoreinstellungen(1,0,MEDIUM,FAST);
			ausgabe("Suedwesten");
			break;
		case WEST:
			setzeMotoreinstellungen(1,0,SLOW,FAST);
			ausgabe("Westen");
			break;
		case NORTHWEST:
			setzeMotoreinstellungen(1,1,FAST,FAST);
			ausgabe("Nordwesten");
			break;
		default:
			ausgabe("???");
			break;
	}
}

//Navigation vorwaerts mit den entsprechenden Motoreinstellungen
void navigierenFRONT(int value){
	//Ausrichten der Schussvorrichtung
	zaehler2 = 0;
	if(zaehler1 == 2){
		setzeSchussMotoreinstellung(MOTORRICHTUNG0,FAST);
	}
	zaehler1++;
	setzeMotorPower(STOP);
	switch(value){
	case 0:
	case 17:
	case 21:
	case 27:
	case 31:
		setzeMotoreinstellungen(0,1,FAST,FAST); //geradeaus
		ausgabe("Stelle");
		break;
	case 4:
	case 5:
	case 10:
	case 14:
	case 15:
```

```
        case 20:
        case 30:
                setzeMotoreinstellungen(0,1,FAST,FAST);   // geradeausfahren
                ausgabe("Gerade");
                break;
        case 8:
        case 9:
        case 13:
                setzeMotoreinstellungen(0,1,FAST,SLOW);     //leicht nach links fahren
                ausgabe("leichtlinks");
                break;
        case 2:
        case 6:
        case 18:
                setzeMotoreinstellungen(0,1,SLOW,FAST);  //leicht nach rechts fahren
                ausgabe("leichtrechts");
                break;
        case 12:
        case 24:
        case 25:
        case 26:
        case 28:
        case 29:
                setzeMotoreinstellungen(0,0,MEDIUM,MEDIUM);  // nach links fahren
                ausgabe("links");
                break;
        case 3:
        case 7:
        case 11:
        case 19:
        case 22:
        case 23:
                setzeMotoreinstellungen(1,1,MEDIUM,MEDIUM);  // nach rechts fahren
                ausgabe("rechts");
                break;
        case 16:
                setzeMotoreinstellungen(0,0,FAST,FAST);     // stark nach links fahren
                ausgabe("starklinks");
                break;
        case 1:
                setzeMotoreinstellungen(1,1,FAST,FAST);     // stark nach rechts fahren
                ausgabe("starkrechts");
                break;
        }
}

//Navigation rueckwaerts mit den entsprechenden Motoreinstellungen
void navigierenBACK(int value){
        //Ausrichtung der Schussvorrichtung
        zaehler1 = 0;
        if(zaehler2 == 2){
```

```
                setzeSchussMotoreinstellung(MOTORRICHTUNG1,FAST);
        }
        zaehler2++;
        setzeMotorPower(STOP);

        switch(value){
        case 0:
        case 5:
                setzeMotoreinstellungen(1,0,FAST,FAST);
                ausgabe("geradeBACK");
                break;
        case 2:
        case 7:
                setzeMotoreinstellungen(1,0,FAST,FAST);
                ausgabe("geradeBACK");
                break;
        case 6:
                setzeMotoreinstellungen(1,0,FAST, STOP);
                ausgabe("leichtLinksBACK");
                break;
        case 3:
                setzeMotoreinstellungen(1,0,STOP,FAST);
                ausgabe("leichtRechtsBACK");
                break;
        case 4:
                setzeMotoreinstellungen(1,0,FAST,STOP);
                ausgabe("linksBACK");
                break;
        case 1:
                setzeMotoreinstellungen(1,0,STOP,FAST);
                ausgabe("rechtsBACK");
                break;
        }
}

void setzeMotorPower(int motGeschw){
        motor_pwm(MOTORSCHUSS, motGeschw);
        sleep(50);
}

void setzeSchussMotoreinstellung(int motRicht, int motGeschw){
        motor_richtung(MOTORSCHUSS, motRicht);
        motor_pwm(MOTORSCHUSS, motGeschw);
        sleep(100);
}

void setzeSchussMotoreinstellungTyp(int typ){
        if(typ==0){  //Typ 0 ist Schuss vorne
                motor_richtung(MOTORSCHUSS,MOTORRICHTUNG0);
                motor_pwm(MOTORSCHUSS,FAST);
                sleep(150);
```

```
                motor_richtung(MOTORSCHUSS,MOTORRICHTUNG0);
                motor_pwm(MOTORSCHUSS,STOP);
                sleep(150);
                motor_richtung(MOTORSCHUSS,MOTORRICHTUNG1);
                motor_pwm(MOTORSCHUSS,FAST);
                sleep(150);
                motor_richtung(MOTORSCHUSS,MOTORRICHTUNG1);
                motor_pwm(MOTORSCHUSS,STOP);
                sleep(150);
                motor_richtung(MOTORSCHUSS,MOTORRICHTUNG0);
                motor_pwm(MOTORSCHUSS,FAST);
                sleep(150);
                motor_richtung(MOTORSCHUSS,MOTORRICHTUNG0);
                motor_pwm(MOTORSCHUSS,STOP);
                sleep(150);
        } else {
                motor_richtung(MOTORSCHUSS,MOTORRICHTUNG1);
                motor_pwm(MOTORSCHUSS,FAST);
                sleep(150);
                motor_richtung(MOTORSCHUSS,MOTORRICHTUNG1);
                motor_pwm(MOTORSCHUSS,STOP);
                sleep(150);
                motor_richtung(MOTORSCHUSS,MOTORRICHTUNG0);
                motor_pwm(MOTORSCHUSS,FAST);
                sleep(150);
                motor_richtung(MOTORSCHUSS,MOTORRICHTUNG0);
                motor_pwm(MOTORSCHUSS,STOP);
                sleep(150);
                motor_richtung(MOTORSCHUSS,MOTORRICHTUNG1);
                motor_pwm(MOTORSCHUSS,FAST);
                sleep(150);
                motor_richtung(MOTORSCHUSS,MOTORRICHTUNG1);
                motor_pwm(MOTORSCHUSS,STOP);
                sleep(150);
        }
}

void setzeMotoreinstellungen(int mot1Richt, int mot2Richt, int mot1Geschw, int mot2Geschw) {
        //berprfung, ob sich die Motorwerte ber lngere Zeit nicht ndern, wenn dies der Fall
ist,
        //dann Zufallsbewegung auslsen
        unsigned int zufallszahl;
        unsigned int zeit;


        if(letzterMotorbefehl1 == mot1Richt && letzterMotorbefehl2 == mot2Richt &&
letzterMotorbefehl3 == mot1Geschw && letzterMotorbefehl4 == mot2Geschw){
                if (stressfaktor > 30000){stressfaktor = 1;}
                stressfaktor++;
                ausgabeInt(stressfaktor);
        }
```

```
	if(stressfaktor > 20){
		stressfaktor = 0;
		zeit = akt_time();
		//Rueckwaerts
		if(mot1Richt == 1 && mot2Richt == 0){
			zufallszahl = zeit % 32;
			navigierenFRONT(zufallszahl);
			sleep(700);
			clear_time();
		}else {
			zufallszahl = zeit % 8;
			navigierenBACK(zufallszahl);
			sleep(700);
			clear_time();
		}
		zufallszahl = 0;
	}

	letzterMotorbefehl1 = mot1Richt;
	letzterMotorbefehl2 = mot2Richt;
	letzterMotorbefehl3 = mot1Geschw;
	letzterMotorbefehl4 = mot2Geschw;

	////mot2Geschw = mot2Geschw + 2;
	//Geradeaus Mot1=0, Mot2=1
	mot2Richt = mot2Richt ^ 1; //XOR-Verknpfung mit 1 -> Ergebnis: Invertierter Wert
	motor_richtung(MOTOR1,mot1Richt);
	motor_richtung(MOTOR2,mot2Richt);
	motor_pwm(MOTOR1,mot1Geschw);
	motor_pwm(MOTOR2,mot2Geschw);
}

unsigned char getTorwert(int digitalausgang, unsigned char frequence){
	unsigned char wert;
	wert = 0;
	if(digitalausgang == 4){
		mod_ir0_takt(frequence);
		mod_ir0_maxfehler(IR_MAX_ERROR);
		wert = (mod_ir0_status()>THRESHOLD) ? mod_ir0_status() : 0;
	}
	if(digitalausgang == 5){
		mod_ir1_takt(frequence);
		mod_ir1_maxfehler(IR_MAX_ERROR);
		wert = (mod_ir1_status()>THRESHOLD) ? mod_ir1_status() : 0;
	}
	if(digitalausgang == 6){
		mod_ir2_takt(frequence);
		mod_ir2_maxfehler(IR_MAX_ERROR);
		wert = (mod_ir2_status()>THRESHOLD) ? mod_ir2_status() : 0;
	}
```

```
	if(digitalausgang == 7){
		mod_ir3_takt(frequence);
		mod_ir3_maxfehler(IR_MAX_ERROR);
		wert = (mod_ir3_status()>THRESHOLD) ? mod_ir3_status() : 0;
	}
	return wert;
}

void findeTor(){
	// Initalisierung Torerkennung
	// Anschluss an die DigitalAusgaenge 04 bis 07
	unsigned char irFrequence = (dip_pin(0) == 0) ? GOAL_FREQUENCY_100 :
GOAL_FREQUENCY_125;
	unsigned char frontLeft, frontRight, frontMid;
	mod_ir0_takt(irFrequence);
	mod_ir1_takt(irFrequence);
	mod_ir3_takt(irFrequence);

	mod_ir0_maxfehler(IR_MAX_ERROR);
	mod_ir1_maxfehler(IR_MAX_ERROR);
	mod_ir3_maxfehler(IR_MAX_ERROR);

	frontLeft  = mod_ir0_status();
	frontMid   = mod_ir1_status();
	frontRight = mod_ir3_status();

	if(frontMid    > THRESHOLD) {
		setzeMotoreinstellungen(0,1,FAST,FAST);
		ausgabe("Gerade Tor");
		if (digital_in(10) == 1) {
			//ausgabe("Ball und Tor");
			setzeSchussMotoreinstellungTyp(0);
			goal = 1;
		}
	} else if(frontLeft   > THRESHOLD) {
		setzeMotoreinstellungen(0,1,FAST,SLOW);
		//ausgabe("rechts Tor");
	} else if(frontRight  > THRESHOLD) {
		setzeMotoreinstellungen(0,1,SLOW,FAST);
		//ausgabe("links Tor");
	} else {
		//ausgabe("kein Tor");
		kompassAuswerten();
	}

	LedsOFF();
	lcd_cls();
}
```

```
void findeTorHinten(){
	if (digital_in(12) == 1) {
		enum compassDirection kompass;
		kompass = getCompassDirectionSouth();
		//bei vier Ausrichtung Sueden
		if(kompass == 4){
			setzeMotoreinstellungen(1,0,MFAST,MFAST);
			sleep(100);
			setzeSchussMotoreinstellungTyp(1);
			goal = 1;
		}else
			kompassAuswertenHinten();
	}
}

void ShiftHistory(int *sensorHistory) {
	sensorHistory[4] = sensorHistory[3];
	sensorHistory[3] = sensorHistory[2];
	sensorHistory[2] = sensorHistory[1];
	sensorHistory[1] = sensorHistory[0];
	sensorHistory[0] = 0;
}

void LedsON(){
	//IR_LEDs einschalten
	led(0,1);
	led(1,1);
	led(2,1);
	led(3,1);
	sleep(10);
}

void LedsOFF(){
	//IR_LEDs ausschalten
	led(0,0);
	led(1,0);
	led(2,0);
	led(3,0);
}

void ausgabe(char *value) {
	lcd_cls();
	lcd_puts(value);
	sleep(100);
}

void ausgabeInt(int value) {
	lcd_cls();
	lcd_uint(value);
	sleep(150);
}
```

```
enum compassDirection getCompassDirection() {
        unsigned char compassDir = analog(14);
        if ((compassDir > 55) && (compassDir <= 75)) return NORTH;
        if ((compassDir > 75) && (compassDir <= 105)) return NORTHEAST;
        if ((compassDir > 105) && (compassDir <= 126)) return EAST;
        if ((compassDir > 126) && (compassDir <= 145)) return SOUTHEAST;
        if ((compassDir > 145) && (compassDir <= 165)) return SOUTH;
        if ((compassDir > 165) && (compassDir <= 185)) return SOUTHWEST;
        if ((compassDir > 185) && (compassDir <= 200)) return WEST;
        if ((compassDir > 30) && (compassDir <= 55)) return NORTHWEST;
        return NORTH;
}

enum compassDirection getCompassDirectionSouth() {
        unsigned char compassDir = analog(14);
        if ((compassDir > 55) && (compassDir <= 75)) return NORTH;
        if ((compassDir > 75) && (compassDir <= 105)) return NORTHEAST;
        if ((compassDir > 105) && (compassDir <= 126)) return EAST;
        if ((compassDir > 126) && (compassDir <= 145)) return SOUTHEAST;
        if ((compassDir > 145) && (compassDir <= 165)) return SOUTH;
        if ((compassDir > 165) && (compassDir <= 185)) return SOUTHWEST;
        if ((compassDir > 185) && (compassDir <= 200)) return WEST;
        if ((compassDir > 30) && (compassDir <= 55)) return NORTHWEST;
        return SOUTH;
}
```

# 8 Anlage 2 - Abbildungsverzeichnis