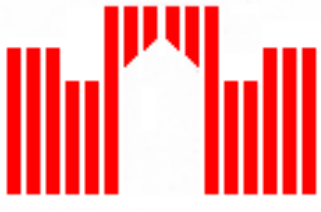

Fachhochschule Brandenburg
Fachbereich
Informatik und Medien

Dokumentation
Projekt

Künstliche Intelligenz
Autonome Mobile Systeme

System:

- Ghost Rider -

Sebastian Moritz
Kai Koplin

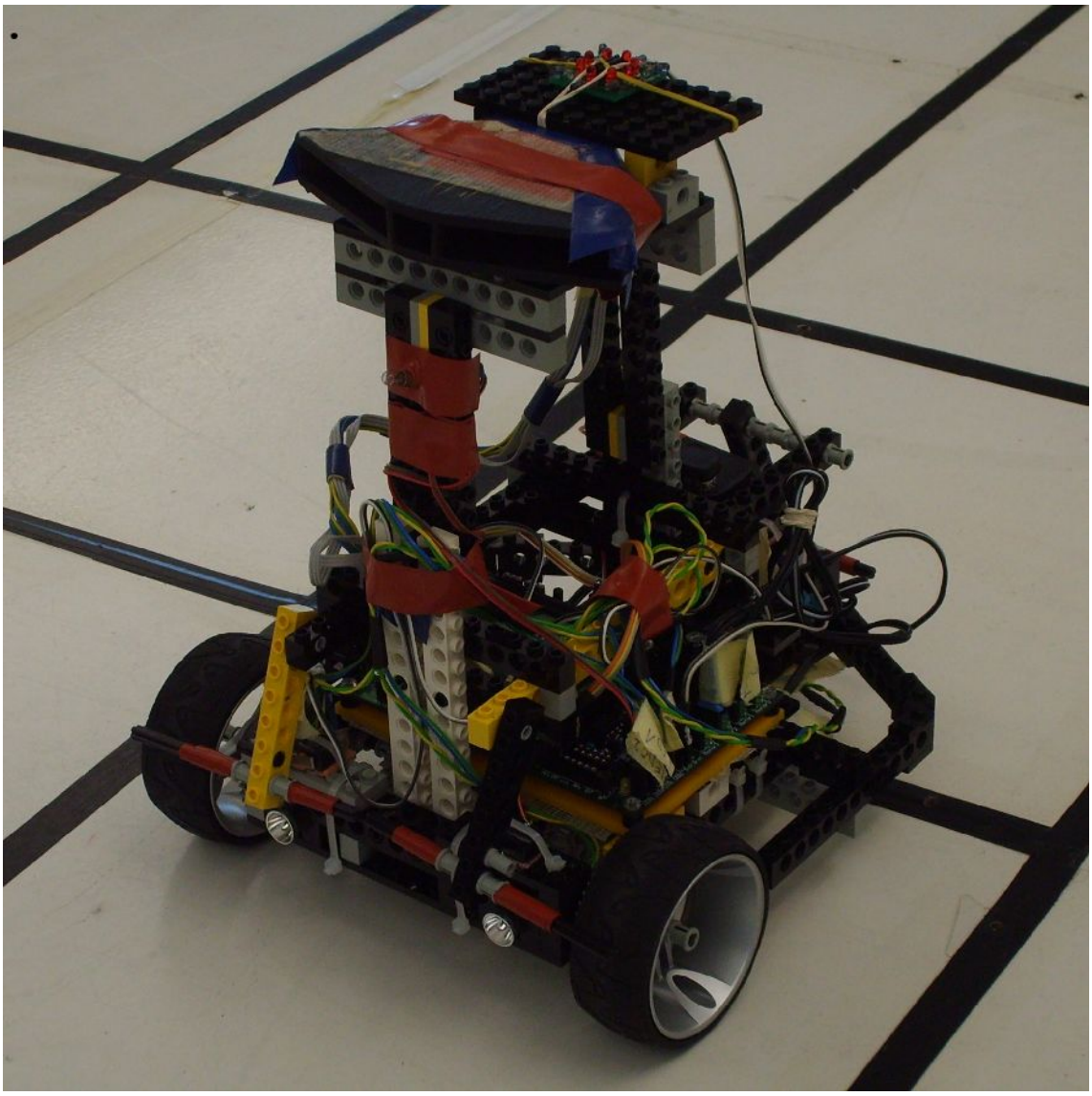

Abbildung 1: Roboter Ghost Rider

Inhaltsverzeichnis

1 Hardware

1.1 Aufbau und Antrieb des Roboters

Fortbewegt wird der Roboter durch zwei Elektromotoren, die autonom von einander angesteuert werden können. Dadurch lässt sich der Roboter bei entgegengesetzt laufenden Motoren nahezu auf der Stelle um 360 Grad drehen. Die Räder selbst werden über ein Getriebe bewegt, um den Roboter die nötige Kraft und Geschwindigkeit zu geben. Abbildung 2 stellt das verwendete Getriebe des Roboters dar.

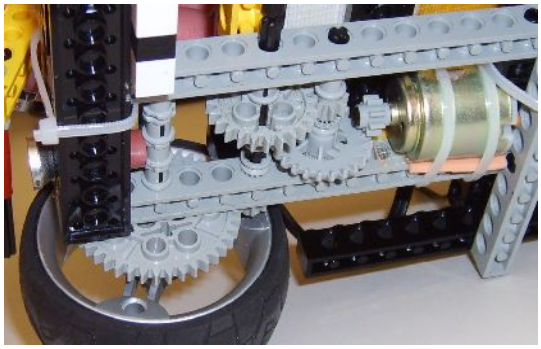

Abbildung 2: Getriebe

Um die Kippanfälligkeit des Roboters zu verringern, wurde das gesamte Gewicht des Roboters auf die beiden Räder und einen zusätzlichen Aufstützpunkt verteilt . Darüber hinaus wurde das Controllerboard sowie der Akku so tief wie möglich in den Roboter eingebaut, um einen möglichst niedrigen Schwerpunkt zu erhalten.

1.2 Sensorik zur Detektion des Gegners

Jeder Roboter sendet ein moduliertes Signal aus, das von einem gegnerische Roboter lokalisiert werden kann. Es gibt zwei unterschiedliche Signale eines für den Hasen und eines für den Fuchs. Eine Darstellung des IR-Senders kann der *Abbildung 3* entnommen werden.

Für den Empfang des modulierten Signals des gegnerischen Roboters wurden 3 IR-Dioden verwendet. Die Dioden wurden in ein trichterförmiges Gehäuse eingebaut, so dass jeder Fernsteuerungsempfänger ein IR-Signal aus einer bestimmten Richtung empfangen konnte. Durch diese Einschränkung konnte eine grobe Abschätzung darüber getroffen werden, wo sich der gegnerische Roboter gerade befindet. *Abbildung 4* zeigt das Gehäuse des Empfängers mit den innen liegenden Dioden.

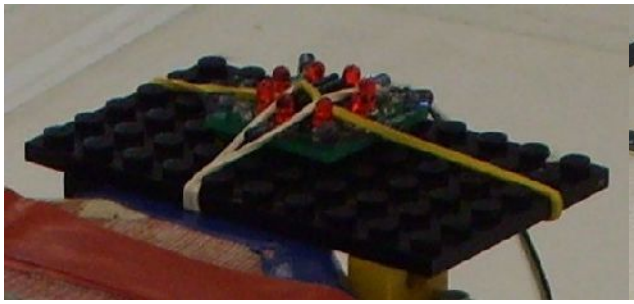

Abbildung 3: IR-Sender

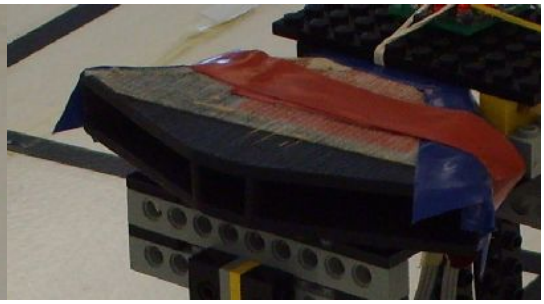

Abbildung 4: IR-Empfänger

1.3 Sensorik zur Detektion von Hindernissen

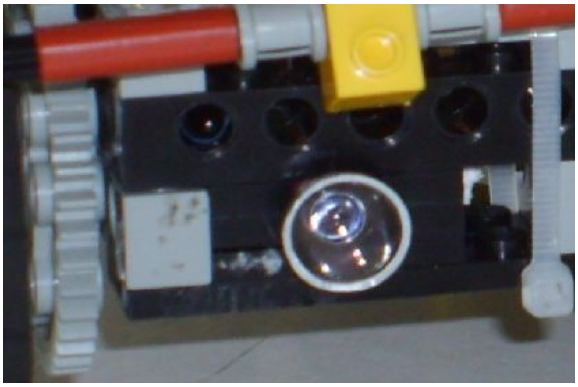

Abbildung 5: Infrarot-Sensoren

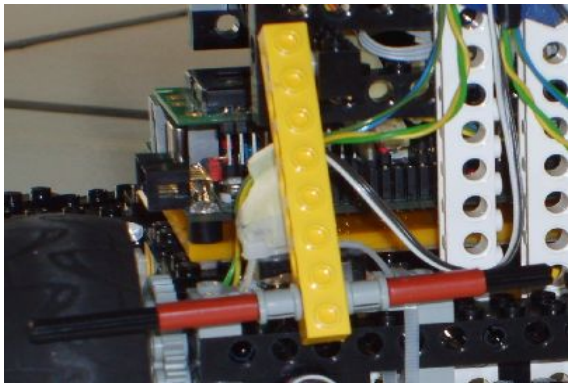

Abbildung 6: Taster

Für die Erkennung von Hindernissen wurden zwei unterschiedliche Arten von Sensoren verwendet, zum einen Sensoren, die Infrarot-Licht detektieren können und zum anderen Taster, die auf Berührung reagieren.

Bei den Infrarot-Sensoren wurden zwei Typen von Dioden kombiniert, zum einen welche die Infrarot-Licht emittieren können und zum anderen welche die Infrarotstrahlung detektieren können.
Diese Kombination wurde sowohl vorne rechts als auch vorne links für die Abstandsberechnung benutzt. Eine Darstellung dieser Sensoren kann der *Abbildung 5* entnommen werden.

Die Infrarot-Sensoren haben den Vorteil, dass sie Hindernisse aus weiter Distanz erkennen können. Ein Nachteil ist, dass die Messung für die Distanz stark von der Umgebungshelligkeit abhängt. Dies kann bei stark schwankenden Lichtverhältnissen zu Problemen führen. Darüber hinaus haben sie den Nachteil, dass sie schmale Hindernisse nicht erkennen können.

Aus diesen Gründen wurden zusätzlich noch die Taster montiert, die unabhängig von den Lichtverhältnissen auf Berührung reagieren.
In *Abbildung 6* ist der Schwenkarm zu erkennen, der den Taster auslöst.

2 Die Programmierung

2.1 Allgemeines

Bei der Programmierung wurde schon wie bei der Konstruktion darauf geachtet, die gewünschte Funktionalität, so einfach wie möglich zu erhalten. Das bedeutet, dass auf Kompliziertes, wie Multitasking verzichtet wurde.

Da die Programmierung im allgemeinen stark von der Verschaltung der einzelnen Anschlüsse, wie LED's auf dem Motor abhängt, wurden für solche Dinge Precompiler-Anweisungen benutzt.

Diese ermöglichen an zentraler Stelle die Umschaltung der Anschlüsse von einer LED auf eine andere. So ist es möglich, die LED's am Board zu ändern und nur an einer Stelle des Programms die Zuordnung des Anschlusses innerhalb des Programmes zu ändern.

Zum gleichen Zweck wurden bei den Motoren Enumerations eingesetzt, um die Richtung und den Motor auf einfache Art, wie vor oder zurück ansprechen zu können. Im Original sind den einzelnen Motoren für die Drehrichtung dabei 0 und 1 zugewiesen.

Um die Einstellungen noch einmal zu vereinfachen, wurde die Geschwindigkeit, die von 0 -10 einstellbar ist, auf den Bereich -10 bis 10 erweitert. Wobei die negativen Zahlen die Rüclwärtsbewegung symbolisieren.

2.2 Logik

Wie im vorhergehenden Kapitel ausgeführt, wurde bei der Ausführung auf einfache Strukturen geachtet. Dies spiegelt sich auch in der eigentlichen Logik für die Ausführung der Aktionen wieder.

Als erstes wird abgefragt, in welchem Modus der Roboter laufen soll.
Dies geschieht über die Abfrage des PIN-Schalters 1. Dabei steht die Einstellung 0 für einen Hasen und die Einstellung 1 für einen Fuchs.

Nach diesem wird noch abgefragt, wie lange eine Runde dauern soll.
Dies kann man über den PIN-Schalter 2 einstellen.

Sobald dies alles geschehen ist, beginnt die eigentliche Schleife, die für die eingestellte Zeit läuft.

Als erstes werden dort die Abstandsensoren auf das aktuelle Bild ausgerichtet. Dies bedeutet, dass die entsprechenden Infrarot-LED's ausgeschaltet werden und das dann einkommende IR-Licht gemessen wird. Nun kann man diesen Wert von der eigentlichen Abstandsmessung ausrechnen.

Um die Möglichkeit zu haben, bei einer Annäherung an eine Wand, in die richtige Richtung auszuweichen, wurden an der Frontseite zwei IR-Abstandssensoren angebracht. Nun wird in Richtung des weniger reagierenden Sensors der Wand ausgewichen.

Um auf eine mögliche Fehlfunktion oder auch schlechte Bedingungen für die Sensoren zu reagieren, wurden auch einige Schalterelemente angebracht. Diese reagieren auf Druck an der Vorder- und Rückseite des Roboters.

2.3 Reaktion auf anderen Roboter

Wenn dies geschieht, fährt der Roboter eine kurze Zeit rückwärts in einer leichten Kurve, und dann wieder vorwärts.

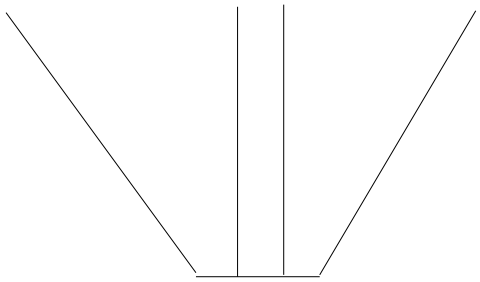

Abbildung 7:

Für die Detektion des anderen Roboters wurde ein Konstrukt, wie in der oberen Abbildung zu sehen ist, verwendet.

Damit war es auf einfache Weise möglich, die Position des anderen Roboters einzuschätzen. Der einfache Schluß aus dieser Anordnung ist, das man auf die einzelnen Kammern entsprechend ihrer Position reagiert. Das heist z.B., wenn der andere Roboter im linken Bereich zu sehen ist, dass der Roboter sich in diese Richtung dreht (als Fuchs).

Zur Rückseite ist nur ein Sensor angebracht. Auf eine Detektion an diesem Sensor wird, als Fuchs mit einer 180° Drehung reagiert. Diese wird ausgeführt, bis einer der anderen Detektoren den Roboter detektiert, oder sie völlig ausgeführt wurde.

Als Hase werden die gerade beschriebenen Reaktionen entsprechend andersherum ausgeführt. Dies heist, wenn der andere Roboter von einem der vorderen Detektoren erkannt wird, dreht sich der Roboter in die entsprechend andere Richtung oder macht eine 180° Drehung.

Da die Regel eine Wand zwischen dem Roboter und dem anderen Roboter untersagte, wurde in der Logik zuerst auf die Detektion des anderen Roboters geachtet, und wenn dieser erkannt wurde nur dementsprechend gehandelt.

3 Fazit

Der Roboter hat zwar nicht unseren Erwartungen entsprochen. Aber gerade durch die Schwierigkeiten während des Projektes haben wir sehr viel lernen können.

4 Quellcode

```
//Autoren: Kai Koplin / Sebastian Moritz

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

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

#define IR_PROC_FUCHS 4
#define IR_PROC_HASE 5
#define MAX_FEHLER 3
#define DRIVE

enum Richtung {
        links,
        rechts
};

enum IRRichtung {
        frontR,
        frontL,
        backR,
        backL
};

int MotorSpeedR=10,MotorSpeedL=10;
int abstIR[] = {255,255,255,255};
unsigned short hase;
short abstFrontR,abstFrontL,abstBackR,abstBackL;
//Funktionen bekannt machen
void initAbstandInfrarot();
unsigned short getIRAbst(int);

void setSpeed(int speed);
void setSpeedL(int speed);
void setSpeedR(int speed);
void kurve(int m,int strength);
void std_kurve(int m);


//Hauptprogrammroutine
void AksenMain(void)
{

        int inKurve = 0,tstFire = 0;
        int tstFR,tstFL;
        int kurveRichtung = links;
        hase = dip_pin(0);

        setSpeed(10);
        mod_ir_an();

        if(hase){
                setze_ir_senden(4);
```

```
		mod_ir0_takt(5);
		mod_ir1_takt(5);
		mod_ir2_takt(5);
		mod_ir3_takt(5);
} else {
		setze_ir_senden(5);
		mod_ir0_takt(4);
		mod_ir1_takt(4);
		mod_ir2_takt(4);
		mod_ir3_takt(4);
}
mod_ir0_maxfehler(MAX_FEHLER);
mod_ir1_maxfehler(MAX_FEHLER);
mod_ir2_maxfehler(MAX_FEHLER);
mod_ir3_maxfehler(MAX_FEHLER);

// in der folgenden Schleife werden die Aktionen durchgeführt
while(1) {
		int frontR_IRStatus,frontC_IRStatus,frontL_IRStatus,back_IRStatus;


		frontR_IRStatus  =  mod_ir0_status();
		frontC_IRStatus =  mod_ir1_status();
		frontL_IRStatus =  mod_ir3_status();
		back_IRStatus    =          mod_ir2_status();


		/*Abfragen der Infrarotwerte für ausgeschaltete Infrarot-LED*/
		initAbstandInfrarot();

		/*die Front -Taster werden abgefragt*/
		tstFR  = !digital_in(frontR);
		tstFL  = !digital_in(frontL);


		/*Abfrage ob einer der beiden Taster gedrückt wurde*/
		if(tstFR || tstFL) {

			if(tstFL) {
				kurveRichtung = links;
			} else {
				kurveRichtung = rechts;
			}
			/*Rückwährts fahren*/
			setSpeed(-10);
			/*Rückwärts in Kurve fahren*/
			kurve(kurveRichtung,12);
			sleep(500);
			/*in normaler Geschwindigkeit vorwärts fahren*/

		} else {
			/*
				Wenn keiner der beiden Taster zum tragen kommt werden die Abstände mit den
				Infrarot-Sensoren gemessen
			*/
			abstFrontR = getIRAbst(frontR);
			abstFrontL = getIRAbst(frontL);
			abstBackR = getIRAbst(backR);
			abstBackL = getIRAbst(backL);

			/*Abfrage ob eine wand vor dem Roboter ist*/
			 if(frontR_IRStatus && !frontC_IRStatus) {
				 kurve(links,MotorSpeedL-1);
```

```
                        //lcd_puts("Rechts");
                } else if(frontL_IRStatus && !frontC_IRStatus) {
                        kurve(rechts,MotorSpeedR-1);
                        //lcd_puts("Links");
                } else if(frontC_IRStatus){
                        lcd_puts("Front");
                } else if(abstFrontR|| abstFrontL) {
                        /*Abfrage ob der roboter nicht schon mit einer Kurve auf dieses Ereignis reagiert*/
                        //if(!inKurve) {
                                /*Abfrage ob Rechts nicht vielleicht eine Wand ist*/

                        if(abstFrontR){
                                if(abstBackL){
                                        //kurve(links,MotorSpeedL-1); /*Kurve Links mit der Stärke 15*/

                                        std_kurve(rechts);
                                        lcd_puts("Rechts");
                                } else {
                                        //kurve(rechts,MotorSpeedR-1); /*Kurve Links mit der Stärke 15*/
                                        std_kurve(links);
                                        lcd_puts("Links");
                                }
                        }
                        else if(abstFrontL){
                                if(abstBackR) {
                                        //kurve(rechts,MotorSpeedR-1);/*Kurve Rechts mit der Stärke
15*/
                                        std_kurve(links);
                                        lcd_puts("Links");
                                } else {
                                        //kurve(links,MotorSpeedL-1);
                                        std_kurve(rechts);
                                        lcd_puts("Rechts");
                                }
                        }
                                        /*Abfrage welcher der beiden Front-Sensoren ein höheres Signal
bekommt*/
                                //inKurve = 1; /*setzen das auf das Ereignis schon reagiert wird*/
                        //}
                } else {
                        setSpeed(10);
                }
        }
        //sleep(200);
        //lcd_cls();
    }
}

unsigned short getIRAbst(int id) {
    short ret = analog(id);

    if(ret >= abstIR[id]) {
                ret = 0;
    } else {

        ret = (abstIR[id]-ret) > 5;//(abstIR[id]>200?3:6);
        //ret = (abstIR[id]-ret);

        if(id == frontL && abstFrontR){
                ret = ((abstIR[frontR]-ret) < (abstIR[id]-ret));
                abstFrontR = !ret;
        }
    }
```

```
	return ret;
}

void initAbstandInfrarot()
{
	led(frontR,0);
	sleep(5);
	abstIR[frontR] = analog(0);
	led(frontR,1);
	sleep(5);

	led(frontL,0);
	sleep(5);
	abstIR[frontL] = analog(1);
	led(frontL,1);
	sleep(5);

	led(backR,0);
	sleep(5);
	abstIR[backR] = analog(2);
	led(backR,1);
	sleep(5);

	led(backL,0);
	sleep(5);
	abstIR[backL] = analog(3);
	led(backL,1);
	sleep(5);
}
// minimum 2
void setSpeed(int speed) {
	setSpeedL(speed);
	setSpeedR(speed);
}

void setSpeedL(int speed) {
	if(speed > -1) motor_richtung(links,1);
	else {
		motor_richtung(links,0);
		speed*=-1;
	}
#ifdef DRIVE
	motor_pwm(links,speed);
#endif
	MotorSpeedL = speed;
}

void setSpeedR(int speed) {
	if(speed > -1) motor_richtung(rechts,1);
	else {
		motor_richtung(rechts,0);
		speed*=-1;
	}
#ifdef DRIVE
	motor_pwm(rechts,speed);
#endif
	MotorSpeedR = speed;
}

void std_kurve(int m) {

	if(m == links) {
		setSpeedR(10);
```

```
                setSpeedL(1);
        } else if(m == rechts) {
                setSpeedL(10);
                setSpeedR(1);
        }
}

void kurve(int m,int strength) {
        if(m == links) {
                int newSpeed = MotorSpeedL-strength;
                if(newSpeed < -10) newSpeed = -10;
                setSpeedL(newSpeed);
        } else if(m == rechts) {
                int newSpeed = MotorSpeedR-strength;
                if(newSpeed < -10) newSpeed = -10;
                setSpeedR(newSpeed);
        }
}
```