Technische Hochschule Brandenburg
University of Applied Sciences

FACHBEREICH INFORMATIK UND MEDIEN
KÜNSTLICHE INTELLIGENZ
AMS-PROJEKT

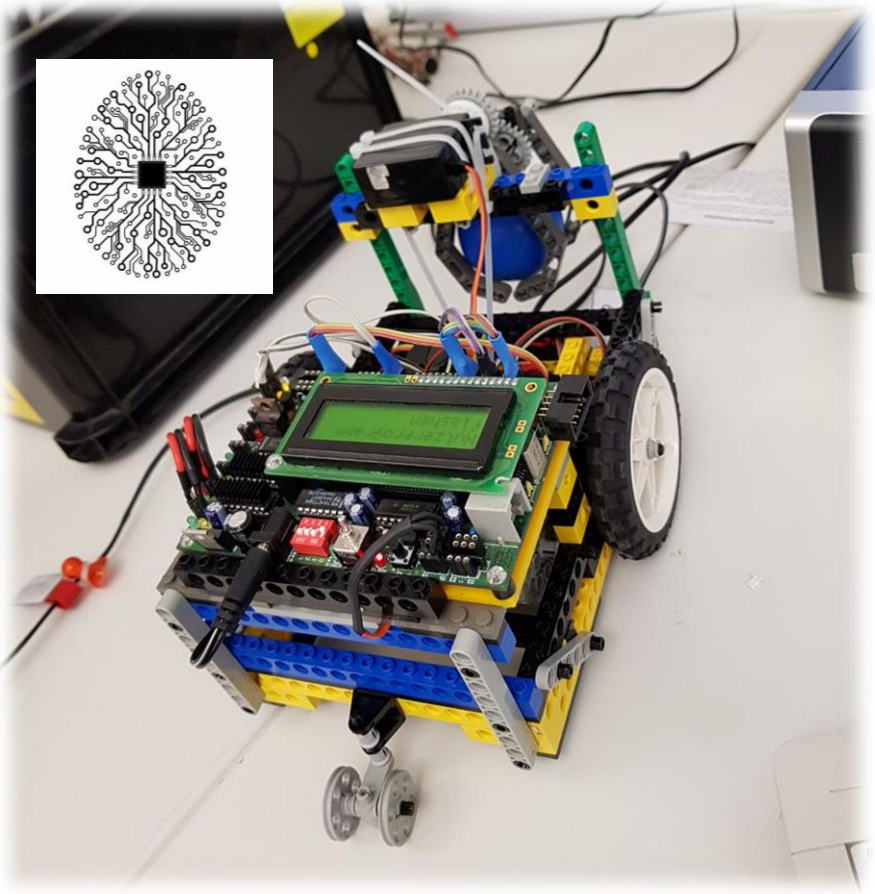

Abbildung 1: "SION" - Autonomes Auto

Studenten:

- A. Steve Ngalamo Egoue
- Daniele Fokam Njilo

Prüfer:

- Prof. Dr.-Ing. Jochen Heinsohn
- Dipl.-Inform. Ingo Boersch

Inhaltsverzeichnis

I. Einleitung:

Im Rahmen des AMS-PROJEKT wurden wir dieses Jahr gefordert einen Roboter zu bauen, der Pizza von einem Ort zu einem anderen liefern kann. Der Roboter sollte entweder in der Lage sein, den besten (kürzeste) Weg selbst zu finden, und nach der Lieferung zurück zum Startpunkt gehen (das heißt Ab- und Rückfahrt selbst finden und fahren), oder alles konnte auch manuell eingegeben werden (zweite Möglichkeit). Für die **autonome Fahrt** (erste Möglichkeit) sollte er bereits Fahrkarten kennen und den besten Weg selbstständig planen und auf vorher bekannte Hindernisse reagieren.

So war die Aufgabenstellung, und von uns wurde die zweite Lösungsmöglichkeit ausgewählt. Die Implementierung der Lösung beziehungsweise der Aufbau des Roboters wurde aber eher Schritt für Schritt gemacht.

II. Lösungsweg:

➢ Strategievorstellung:

Wie schon voher gesagt, haben wir die zweite Lösungsmöglichkeit ausgewählt. Also:

- Erstens wurde die Funktion „geradeaus“ in der „main-Methode“ implementiert
- Zweitens wurde die Funktion „Kreuzungsaktion“ danach implementiert, in der den Fahrplan manuell eingegeben werden muss.
- Wenn alles einwandfrei funktioniert hat, haben wir dann die „Servos-Motor“ der Angreifern angebaut, sowie die dazugehörige Funktion wieder in der „main-Methode“ implementiert.
- Zum Schluss wurde also dann den „Licht-ErkennungsKode“ („Start-Kode“) implementiert, damit das Auto beim Anmachen des Lichtes von allein losfahren könnte.

➢ Verteilung der Arbeit:

Es gab keine richtige Verteilung der Arbeit unter uns. Wir haben also zusammen gearbeitet.

III. Die Hardware:

Den Aufbau des Roboters haben wir auch nicht alles auf einmal geschafft. Wir sollten wie gesagt vorher Schritt für Schritt gehen und bezüglich der vom Roboter erwarteten Bewegung etwas bauen.

Für den Aufbau brauchten wir einige Komponente und zwar Lego-Bausteinen, Sensoren, Motoren, Aksenboard, Greifer, Servomotor, Taster, Getriebe und Räder.

Am Anfang haben nur wir einen Roboter aufgebaut, der einfach autonom fahren könnte, ohne erstmals eine bestimmte Route zu folgen. Da wir noch nicht viel aufgebaut hatten, war der Roboter noch ziemlich schnell. Wir hatten eigentlich nur eine Basis aus Legosteine gemacht, Getriebe und Räder montiert, und Motoren befestigt.

Als nächstes Ziel sollte jetzt der Roboter eine Linie folgen, das heißt geradeaus fahren. Dafür sollten wir dem Aufbau Sensoren hinzufügen und zwar ganz vorne, damit die Sensoren die schwarze Linie in der Mitte erkennen und sie vermeiden. Nur so könnte der Roboter richtig geradeaus fahren.

Am Anfang war alles ja noch sehr ruhig und machte auch viel Spaß, aber irgendwann begann der Druck zu steigen, weil auch die Anforderungen stiegen. Nachdem der Roboter schon geradeaus fahren konnte sollten wir jetzt Greifer bauen, und damit auch die entsprechenden Getriebe montieren und einen Taster noch dazu, damit der Roboter weiß, dass er zum Lieferpunkt der Pizza angekommen ist. Gehen wir einfach mal tiefer in den Aufbau…

2.1. Aufbau von Motoren und Getrieben:

Unser Roboter hat zwei Motoren, die für das Antreiben der jeweiligen Räder des Roboters verantwortlich sind. Sie bilden somit den Grundrahmen seiner Fortbewegung. Das Getriebe eines Rades besteht aus sechs Gängen. Drei von ihnen sind die kleineren Modelle mit nur acht Zähnen. Die anderen drei gehören zu den größeren Modellen, die 40 Zähne haben. Dies schafft eine gute Balance zwischen großen und kleinen Zahnrädern.

2.2. Anbringen der Sensoren:

Wir haben für den Roboter drei unterschiedlichen Arten von Sensoren genutzt, und zwar **Photodetektor** für den Start, **Drucksensoren** (Taster) für die Greifer und **Optoreflexkoppler**:

➢ Der Photodetektor:

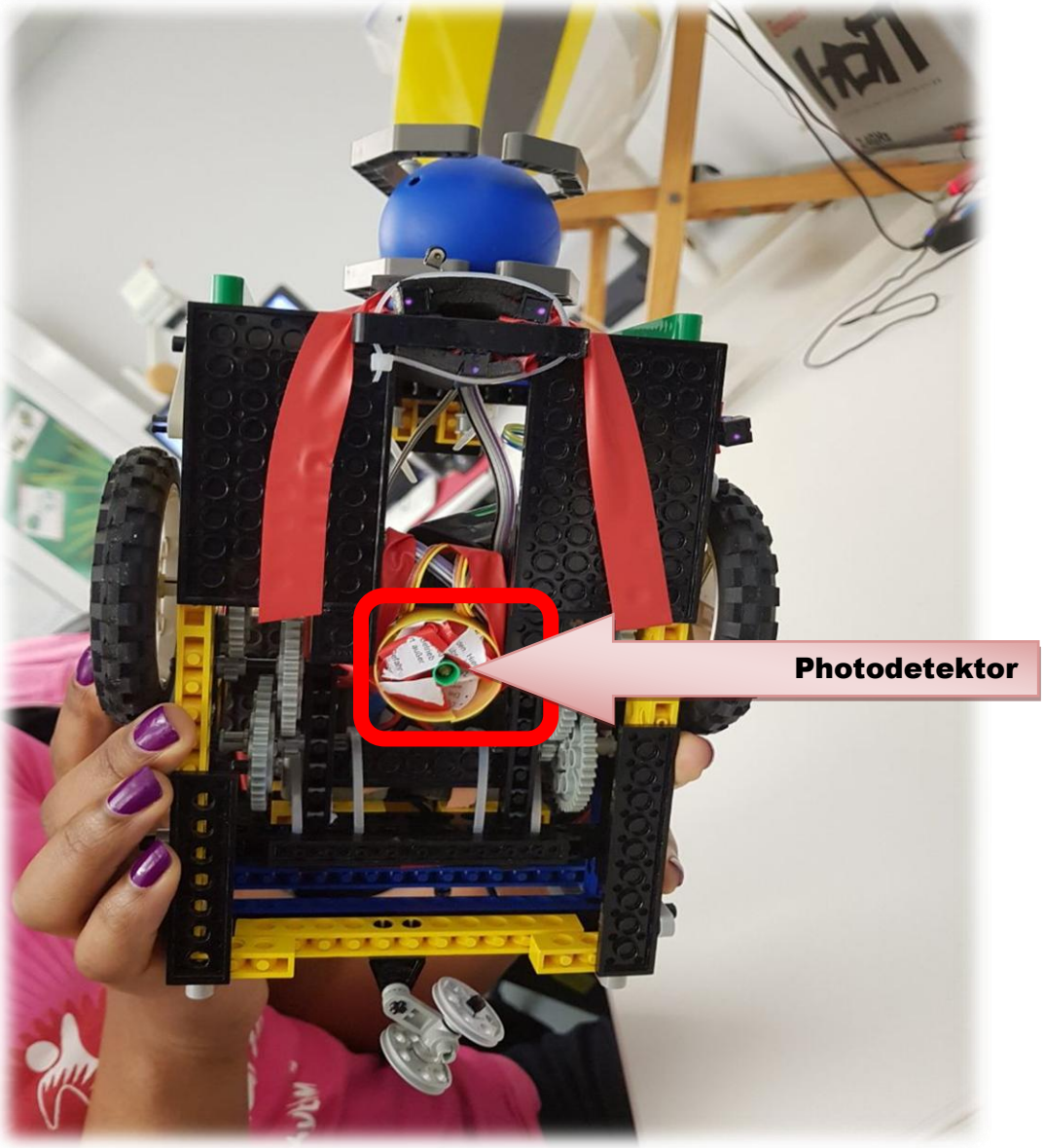

Abbildung 2: Photodetektor

hilft dabei, die Fahrt zu starten. Das ist möglich, indem er ein Lichtsignal empfängt. Das bedeutet, dass er beginnt zu fahren, nur wenn er das Licht sieht. Sonst bleibt er stehen. Wir haben den Sensor ein bisschen in der Mitte des Roboters angebracht.

➢ **Der Drucksensor:**

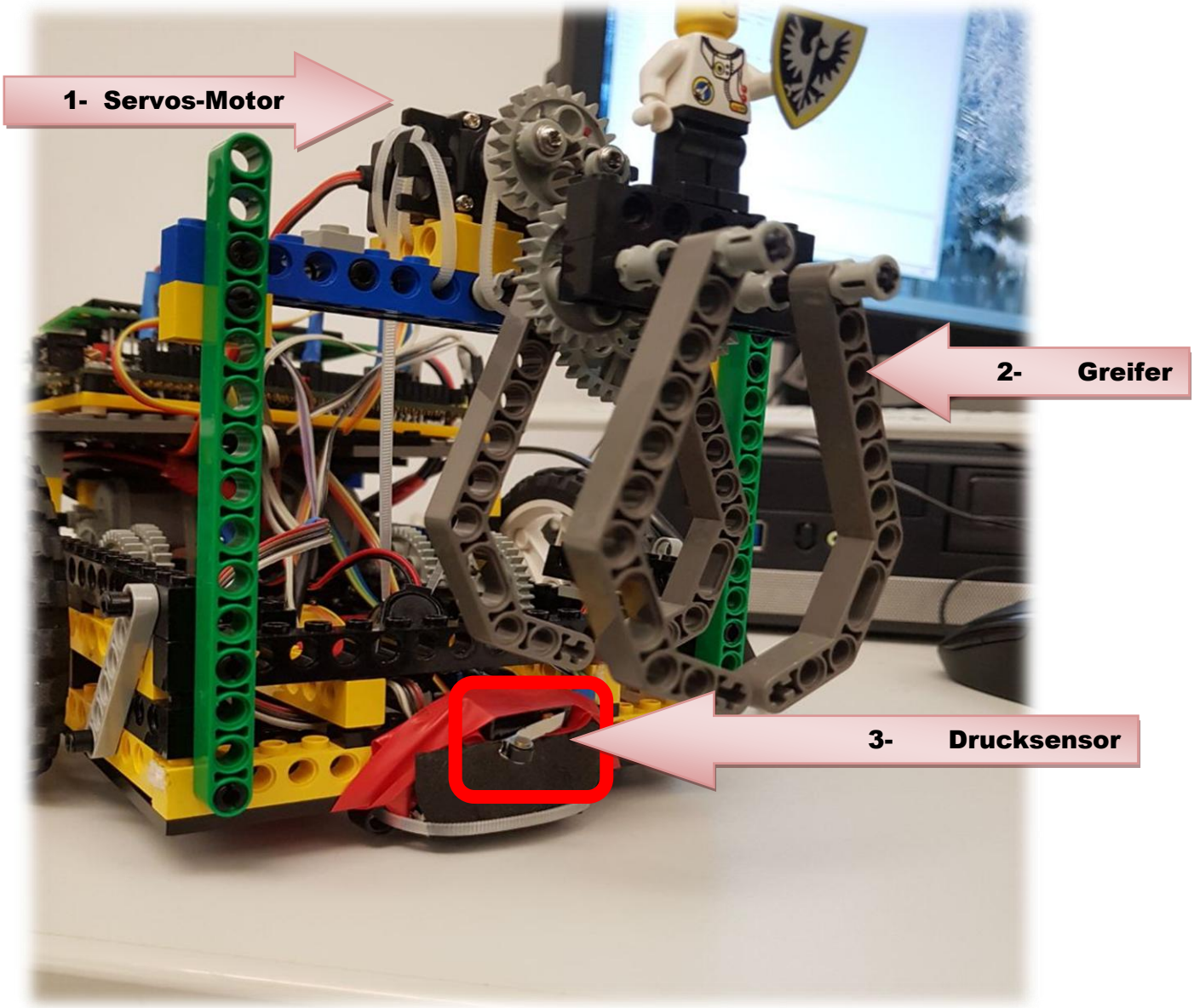

Abbildung 3: Servos-Motor (1), Greifer (2), Drucksensor (3)

Dieser Sensor hat für die Steuerung der Greifer geholfen, damit der Roboter mit Hilfe dieser Greifer die Pizza annimmt und später abgibt (liefert). Bei einem Druck des Sensors konnte die Greifer sich öffnen und die Pizza konnte also geliefert werden. Der Sensor war so programmiert, dass vor der Lieferung die Greifer zu wären, bei der Lieferung sollten die Greifer sich öffnen und einige Minuten danach noch sich wieder schließen. Das Schließen der Greifer nach der Lieferung war dafür, dass wir die nächste zu liefernde Pizza dem Roboter geben können. Den Sensor haben wir ganz vorne angebracht, damit das Ziel oder den Punkt der Lieferung leicht erkennbar wird.

➢ Die Optoreflexkoppler:

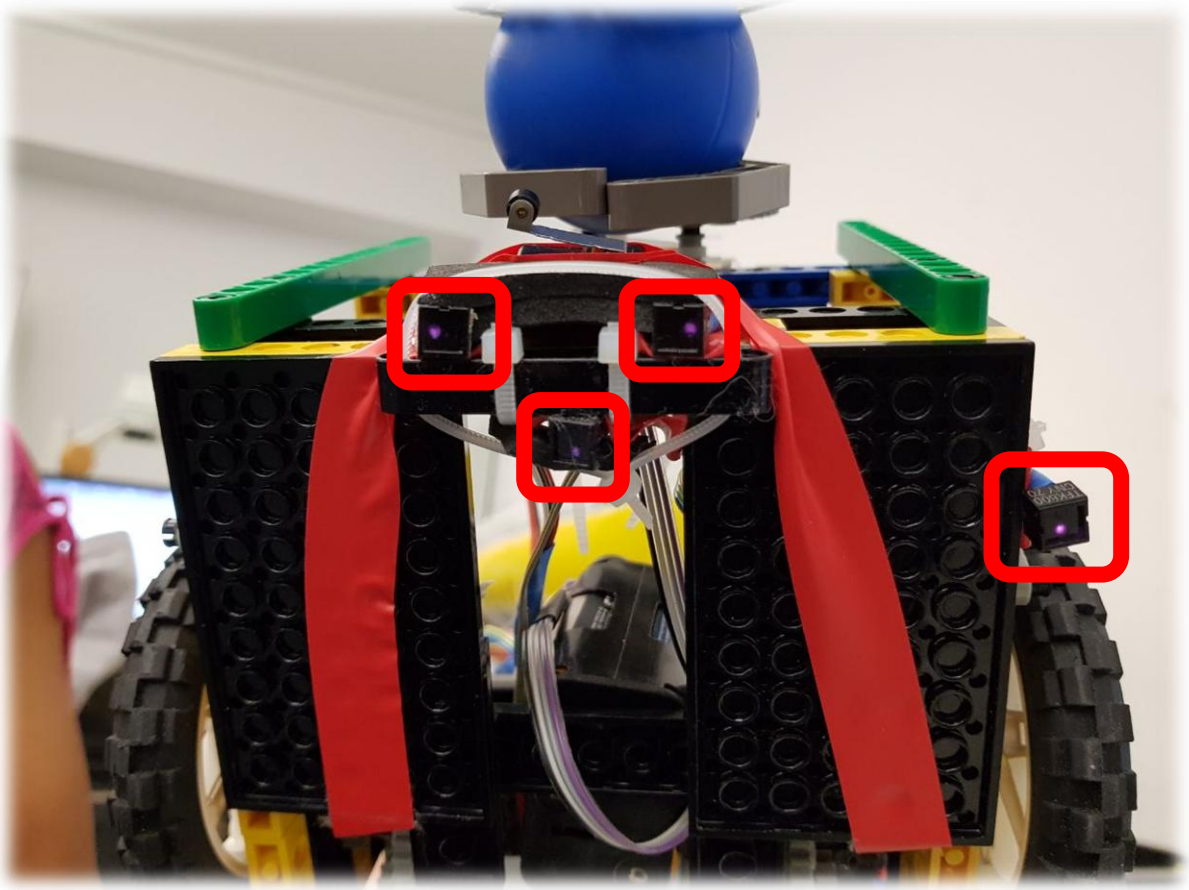

Abbildung 4: Optoreflexkoppler

Sie sind Sensoren, die wir am Meistens benutzt haben. Wir haben insgesamt vier davon benutzt, und zwar eins in der Mitte vorne, zwei anderen ganz daneben und eins auf eine Seite. Damit sie wirklich fest bleiben haben wir ein Klettband benutzt. Die vorne angebrachten Sensoren waren für die Linienfolge und der Sensor an der Seite für die Kreuzungserkennung benutzt.

Der **Optoreflexkoppler** enthält eine Infrarot-LED und einen entsprechenden Infrarotempfänger. Sie befinden sich nur wenige Millimeter über dem Boden, um sicherzustellen, dass die Empfänger das reflektierte Licht erfassen und nicht vom Umgebungslicht beeinflusst werden. Sie dienen als wichtiges Werkzeug, damit der Roboter seine Orientierung behält und erkennt, wo es hell ist und wo dunkle Farben sind. Dies ist besonders nützlich für die Linienverfolgung, da sie schwarz sind, während der Rest des Bodens weiß ist.

2.3. Die Räder:

Um den Roboter zu bewegen, wurden zwei Räder in der relativen Mitte der Konstruktion installiert. Um die nötige Stabilität und Mobilität bei der Fahrt von hinten zu bieten, wurde der Schwerpunkt hinter der Batterie und dem Aksenboard platziert, ein Stützrad wurde eingebaut.

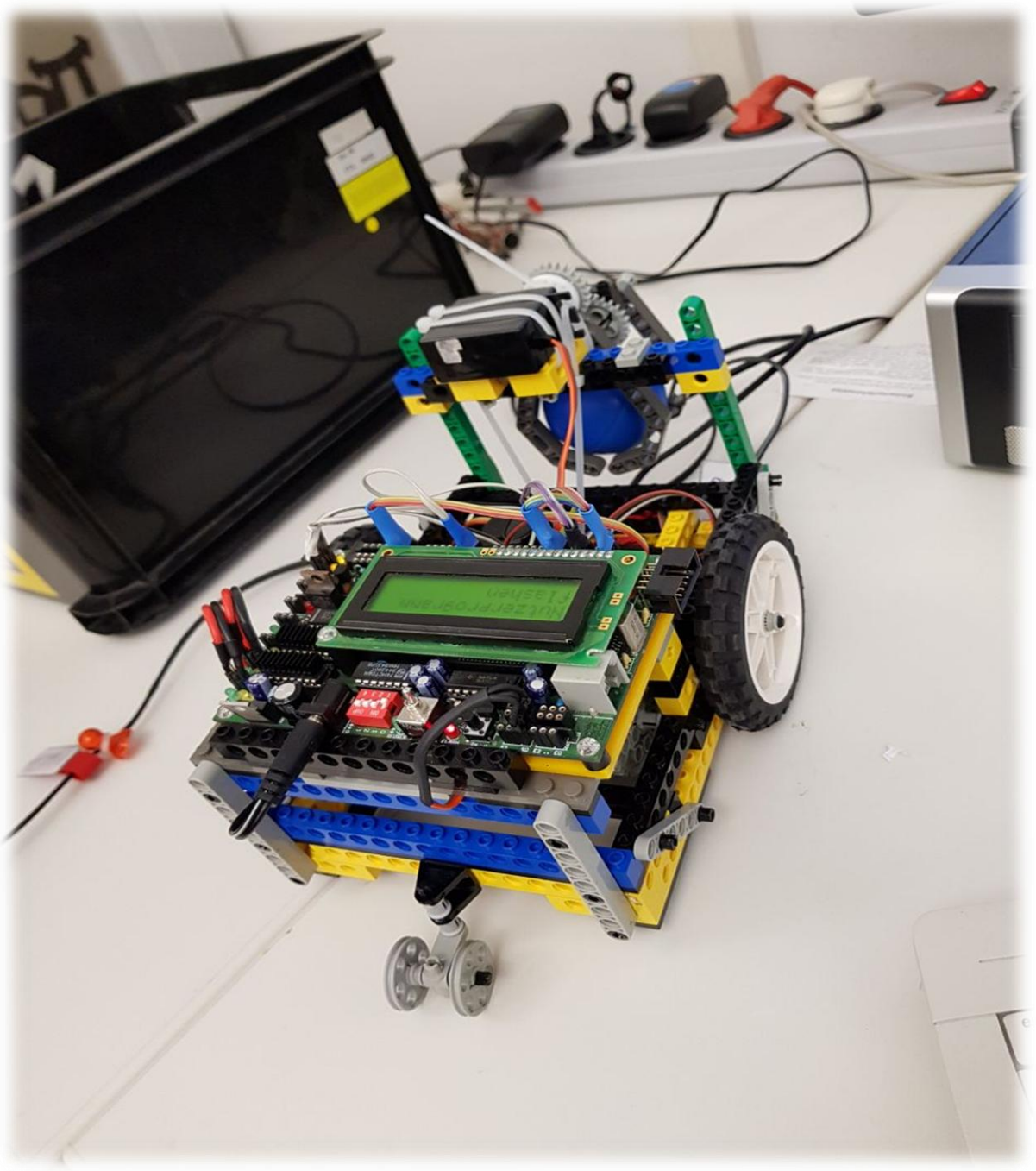

Abbildung 5: Roboter mit seinen Räder (Rechts auf dem Bild) und seinem Stützrad (Hinter)

2.4. Erfahrene Schwierigkeiten:

Ein Problem bei der Hardware war die Handhabung einiger defekter Kits. Beispielsweise waren mehr als drei Optoreflexkopplern defekt, so dass wir nicht mehr (wie erwünscht) mit vielen Optoreflexkopplern rechnen konnten. Wir wurden also gezwungen, nur 4 Sensoren dieser Art zu nutzen. Das Schlimmste war aber nicht, dass die Sensoren kaputt waren sondern, dass wir eher dachten, dass der Roboter so falsch reagierte, weil den Code nicht richtig implementiert wurde. Die Konsequenz davon war aber das Verlieren der Zeit, da wir wegen dieses Problems viel Zeit genommen hatten, um zu gucken, was in dem Code nicht stimmte. Nur durch Überprüfung der Sensoren konnte das Problem gelöst werden.

Eine andere Schwierigkeit war das Problem der Batterie. Batteriestand und Batterietyp haben einen wesentlichen Einfluss auf die Fahreigenschaften des Roboters. Er fuhr oft viel langsamer oder konnte nicht richtig fahren, also verpasste er die Linie. Deshalb hatten wir fast immer die Situation, wo der Roboter nicht mehr (so gut oder gut) wie das letzte Mal fuhrt.

IV. Die Software:

Damit der Roboter fahren konnte müssten wir ein Programm implementieren. Diese Implementierung haben wir mit der Programmiersprache C gemacht. Entsprechend der Evolution der Aufgabe müssten wir Schritt für Schritt auch implementieren, und zwar haben wir erstmals mit einem einfachen Code angefangen.

Der Code müsste erstmals funktionieren, so dass der Roboter genau geradeaus fahren (schwarze Linie folgen) sollte, ohne links oder rechts abzubiegen. Um das zu schaffen haben wir den folgenden Code geschrieben:

```
// Funktion geradeaus();

if (analog(14) >= 16)  // Korrektur nach Links
{

    /*motor_richtung(0, 0);  */     // Links
    motor_pwm(0, 0);
    /*motor_richtung(1, 0); */     // Recht
    motor_pwm(1, 6);

}
else

    if (analog(8) >= 16) // Korrektur nach Rechts
    {



    /*motor_richtung(0, 1); */     // Links
    motor_pwm(0, 6);
    /*motor_richtung(1, 1); */     // Recht
    motor_pwm(1, 0);

    }
    else
    {
        motor_pwm(0, 10);
        motor_pwm(1, 10);

    }
```

Abbildung 6: Geradeaus und Lienienfolgen Funktion

Dann sollten wir den Code implementieren, so dass der Roboter links und rechts abbiegen konnte. Dafür haben wir eine Funktion Kreuzungsaktion wie folgt implemeniert:

```
// RECHTS ===> R:

else if (fahrplan[zaehler] == 'R')
{

     lcd_setxy(1, 0);
     lcd_puts(" RECHTS ");

     motor_pwm(0, 10);
     motor_pwm(1, 10);

     sleep(Wartezeit);

     motor_richtung(0, 1);        // Links
     motor_richtung(1, 1);        // Recht

     sleep(drehungAbbruch2);


     do
     {} while (analog(11) < mitte);   // Mitte

}
```

Abbildung 7: Rechtsabbigen

```
// LINKS ===> L:

else if (fahrplan[zaehler] == 'L')
{

     lcd_setxy(1, 0);
     lcd_puts(" LINKS ");

     motor_pwm(0, 10);
     motor_pwm(1, 10);

     sleep(Wartezeit);

     motor_richtung(0, 0);        // Links
     motor_richtung(1, 0);        // Recht

     sleep(drehungAbbruch1);

     do {} while (analog(11) < mitte); // Mitte


}
```

Abbildung 8: Linksabbiegen

V. Abbildungsverzeichnis:

```
/*****************************************************************************
*****************************/
/* PROJEKT: AMS-Projekt                                                      */
/* AUTOREN: A. Steve Ngalamo Egoue      &       Daniele Fokam NJilo          */
/* SEMESTER: Wintersemester 2017-2018                                        */
/*****************************************************************************
*****************************/



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



/*****************************************************************************
*************************/
/*               KREUZUNG ERKENNEN UND ABBIGEN                               */
/*****************************************************************************
*************************/



// kreuzungsaktion - Fahrplan folgen:

unsigned int zaehler = 0;
unsigned int length = 23;



void kreuzungsaktion(){

   const int mitte = 15;
   const int gerade = 300;
   const int Wartezeit = 420;
   const int drehungAbbruch1 = 300;   // Bei "nach Links abbigen"
   const int drehungAbbruch2 = 200;   // Bei "nach Rechts abbigen"
   const int drehungAbbruch3 = 2200;  // 180 Grad - Drehung

   //char fahrplan[] =                                                        
     { "RLXGGGLRGRLRLRLGLRGGGXGGGLRGRLGLRLRGRLGLRGGGY" };              // F1
   //char fahrplan[] =                                                        
     { "LRGGGGGGGLRLRGGGGGGGLRXLRGGGGGRGGLGGGGGLRY" };                 // F7
   //char fahrplan[] =                                                        
     { "LRGGGGGGGLRLRGGGGGGGLRXLRGGGGGRGGLGGGGGLRY" };                 // F18
   //char fahrplan[] =                                                        
     { "LRGRGGLGLRXLRGRLGRLGRLGLRXLRGRLGLRGRGGLGLRGRLGLRY" };          // F18
   char fahrplan[] =                                                        
     { "LRGGGGGGLRXLRGGGGGGLRY" };                                       // F11
```

```
44
45
46      lcd_cls();
47      lcd_puts("Zaehler =  ");
48      lcd_ubyte(zaehler);             // Mitte kontrolieren
49
50
51      // GERADEAUS FAHREN ===> G:
52
53      if (fahrplan[zaehler] == 'G')
54      {
55
56          lcd_setxy(1, 0);
57          lcd_puts(" GERADE ");
58
59
60          motor_pwm(0, 10);
61          motor_pwm(1, 10);
62
63          sleep(gerade);
64
65      }
66
67
68      // RECHTS ABBIGEN ===> R:
69
70      else if (fahrplan[zaehler] == 'R')
71      {
72
73          lcd_setxy(1, 0);
74          lcd_puts(" RECHTS ");
75
76          motor_pwm(0, 10);
77          motor_pwm(1, 10);
78
79          sleep(Wartezeit);
80
81          motor_richtung(0, 1);           // Linkerer Motor
82          motor_richtung(1, 1);           // Rechter Motor
83
84          sleep(drehungAbbruch2);
85
86
87          do
88          {} while (analog(11) < mitte);  // Mittlerer Sensor
89
90      }
91
92
93      // LINKS ABBIGEN ===> L:
94
95      else if (fahrplan[zaehler] == 'L')
96      {
97
98          lcd_setxy(1, 0);
99          lcd_puts(" LINKS ");
100
101         motor_pwm(0, 10);
102         motor_pwm(1, 10);
103
104         sleep(Wartezeit);
105
106         motor_richtung(0, 0);               // Linkerer Motor
107         motor_richtung(1, 0);               // Rechter Motor
```

```
108
109         sleep(drehungAbbruch1);
110
111         do {} while (analog(11) < mitte);   //  Mittlerer Sensor
112
113
114     }
115
116
117
118     // Drehen und neue Lieferung machen:
119
120     else if (fahrplan[zaehler] == 'X')
121     {
122
123         lcd_setxy(1, 0);
124         lcd_puts(" Neuer Farhtplan !!! ");
125
126         motor_pwm(0, 10);
127         motor_pwm(1, 10);
128
129         sleep(Wartezeit);
130
131
132         // Drehen:
133         motor_richtung(0, 0);      // Linkerer Motor
134         motor_richtung(1, 0);      // Rechter Motor
135         sleep(drehungAbbruch3);
136
137     }
138
139
140     // Auto stehend halten: STOPEN
141
142     else if (fahrplan[zaehler] == 'Y')
143     {
144
145         motor_pwm(0, 10);
146         motor_pwm(1, 10);
147
148         sleep(Wartezeit);
149
150
151         // Drehen:
152         motor_richtung(0, 0);      // Linkerer Motor
153         motor_richtung(1, 0);      // Rechter Motor
154         sleep(2100);
155
156         // Auto halten
157         while (1)
158         {
159             lcd_setxy(1, 0);
160             lcd_puts(" STOP!!! ");
161
162
163             motor_pwm(0, 0);
164             motor_pwm(1, 0);
165         }
166
167     }
168
169
170     // Exeption:
171
```

```
172     else {
173
174         lcd_puts(" *** ERROR ***");
175
176         sleep(200);
177         zaehler = 0;
178
179     }
180
181
182
183     // Fahrplan durchführen und kontrolieren:
184
185     if (zaehler < (sizeof(fahrplan) - 1))
186         zaehler++;
187     else
188     {
189         zaehler = 100;
190     }
191
192 }
193
194
195
196
197 /*****************************************************************************
    *********************************/
198 /*                          HAUPTROUTINE                                     
                                */
199 /*****************************************************************************
    *********************************/
200
201
202 void AksenMain(void)
203 {
204
205
206     unsigned int control = 1;
207
208 // Lichtsensor für den automatischen Start:
209
210     while (analog(2) > 100)
211     {
212         lcd_cls();
213         lcd_puts("** WILKOMMEN **");
214         lcd_setxy(1, 0);
215         lcd_puts("Licht bitte an!!! ");
216         sleep(200);
217
218     }
219
220
221 /*****************: Generale                                                  
    Routine :*****************************************************************
    *****/
222
223     while (1)
224     {
225
226         // Motoren anmachen und richtung abgeben:
227
228         motor richtung(0, 1);      // Linkerer Motor
```

```
229         motor_richtung(1, 0);      // Rechter Motor
230
231
232
233 /************        I-   Lieferung abgeben und kurz zurückfahren:
     *********************************/
234
235         if (digital_in(8) == 0)
236         {
237
238             // Attribut für Kontrolle des Servosmotors und die Steuerung der
                  Angriffer:
239             control = 2;
240
241             // Servos  öffnen
242             servo_arc(0, 20);
243             sleep(300);
244
245
246             // Für kruz zurückfahren:
247
248             motor_richtung(0, 0);      // Linkerer Motor
249             motor_richtung(1, 1);      // Rechter Motor
250             sleep(400);
251
252         }
253
254
255 /**********          II-   Servos schliessen und drehen (180 Grad):
     *********************************/
256
257         else if (digital_in(8) == 1 && control == 2)
258         {
259
260             servo_arc(0, 100);
261
262
263             // Drehen:
264             motor_richtung(0, 0);      // Linkerer Motor
265             motor_richtung(1, 0);      // Rechter Motor
266             sleep(500);
267
268             do {} while (analog(11) < 15); // Mittlerer Sensor
269
270             control = 1;
271
272         }
273
274
275 /********      III-  Fahrplan weiter folgen:
     ************************************************/
276
277         else
278         {
279             //Servos Angriffe abschliessen:
280             servo_arc(0, 100);
281
282
283
284        /*********** Kreuzung erkennen und Fahrplan folgen: **********/
285
286             if (analog(0) > 10)
287             {
288                 kreuzungsaktion();
```

```
289             }
290
291             else
292             {
293
294     /************ Funktion geradeaus: Schwarze Linien folgen  ******/
295
296                 if (analog(14) >= 16)       // Korrektur nach Links
297                 {
298
299                     motor_pwm(0, 0);       // Linkerer Motor
300                     motor_pwm(1, 6);       // Rechter Motor
301
302                 }
303                 else
304
305                     if (analog(8) >= 16)  // Korrektur nach Rechts
306                     {
307
308                     motor_pwm(0, 6);      // Linkerer Motor
309                     motor_pwm(1, 0);      // Rechter Motor
310
311                     }
312                     else
313                     {
314                         motor_pwm(0, 10);
315                         motor_pwm(1, 10);
316
317                     }
318             }
319
320
321         }
322     }
323
324
325 }
```