2019

Autonome Fahrzeuge

Robert Kamga Kengmogne und Landry Ngueyep Kabès

Prof. Heinsohn und Herr Boersch

19.1.2019

Inhalt

1 Einleitung

Künstliche Intelligenz (KI) ist ein Thema, das heutzutage sehr besprochen wird. Von Da Vinci-Robotern aus bis zu autonomen Fahrzeugen hat sich KI peu-à-peu in fast allen Branchen durchgesetzt. Dieses Durchsetzten lässt sich von den Einen positiv betrachten und eher negativ von den anderen, die sich jedoch Sorgen um ihre Arbeitsplätze machen. Was wäre, wenn in den nächsten hundert Jahren keine Ärzte mehr zu einer Operation gebraucht werden, weil Roboter vielleicht diese Arbeit übernähmen oder sogar keine Pizza-Lieferanten mehr, weil autonome Fahrzeuge die Pizzen so schnell wie möglich zu Kunden bringen würden.
Im Rahmen des Projekts **(Name des Projekts)** haben studierende der Studiengänge Informatik und Medizininformatik die Aufgabe bekommen, ein völlig autonomes Fahrzeug in Betrieb zu nehmen. Dabei mussten sie selber das Fahrzeug anhand LEGO-Bausteine zusammenbasteln und den entsprechenden Programmcode implementieren. Die Studierenden haben sich stets donnerstags zw. Vierzehn und achtzehn Uhr im KI-Labor des Fachbereichs Informatik der technischen Hochschule Brandenburg in ihren zweier-Teams getroffen, um am Projekt arbeiten zu können. Am Ende des Projekts soll das Fahrzeug in der Lage sein, eine bzw. mehrere Pizzen zu liefern, nachdem ihm einen *Fahrplan (siehe Tabellenverzeichnis)* gegeben wird. Im Fahrplan können einige Straßen gesperrt (Durch ein „x“ repräsentiert) oder zugängig (Durch einen „.“ Oder „F“ Repräsentiert) sein. Durch „F“ werden auch Lieferorte abgebildet. Am Ende des Projekts gab es einen kleinen Wettbewerb. Dabei müssen zwei unterschiedliche Autos Pizzen liefern. Abgesehen davon, dass es höchstens drei Lieferorte geben kann, bekommt das Auto, das schneller und an richtigen Orten alle drei Pizzen geliefert hat, mehrere Punkte.

2 Material und Methode

2.1 Material (Bilder hinzufügen)

Zur Realisierung des Projekts wurden viele Materialien verwendet u.a.

- Lego-Bausteine, die dazu gedient haben das Auto zu bauen (*Abbildung 1*).
- Zwei Motoren zur Inbetriebnahme des Autos (*Abbildung 2*).
- Ein Aksen-Board (Abbildung 3)
- Vier Opto-sensoren, die zur Straßen- und Kreuzungserkennung vorne ans Auto gebracht wurden (*Abbildung 4*).
- Ein Servomotor zum Antrieb des Greifers, der zum Tragen der Pizza dient (*Abbildung 5*).
- Ein Taster zum Stoppen des Autos am Lieferort (*Abbildung 6*).
- Ein Lichtsensor, der unter dem Auto liegt, um den Start wahrzunehmen (*Abbildung 7*).

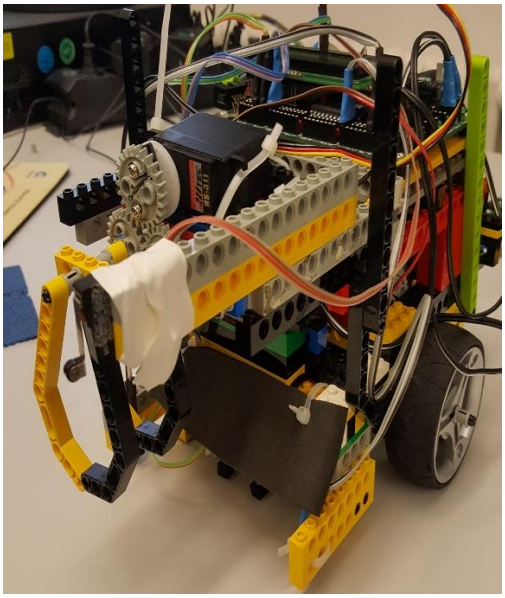

Abbildung 1: Auto

Abbildung 2: Motor

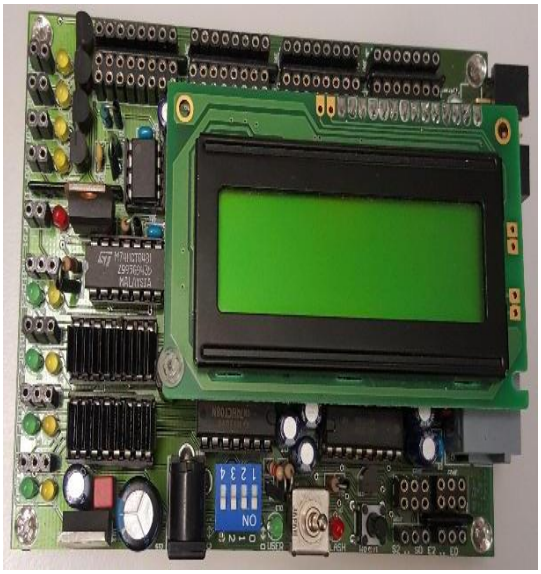

Abbildung 3: Aksen-Board

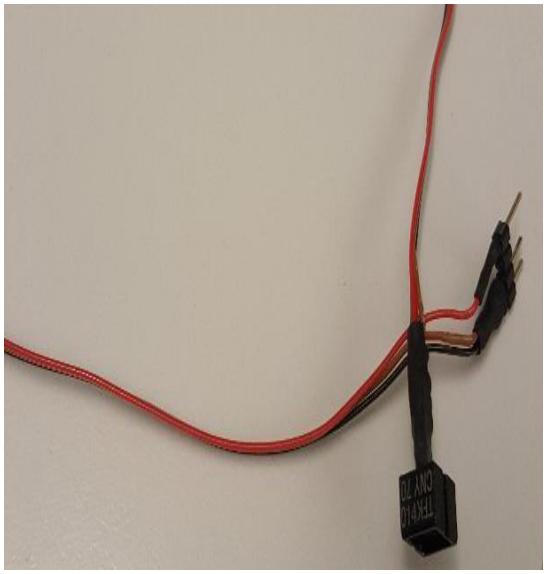

Abbildung 4: Optokoppler

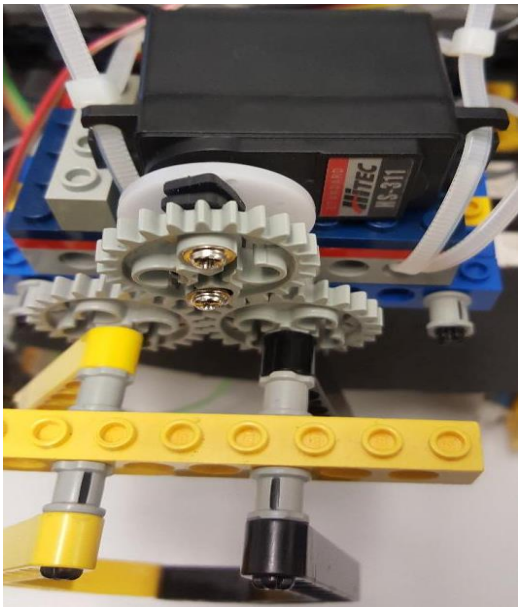

Abbildung 5: Servomotor

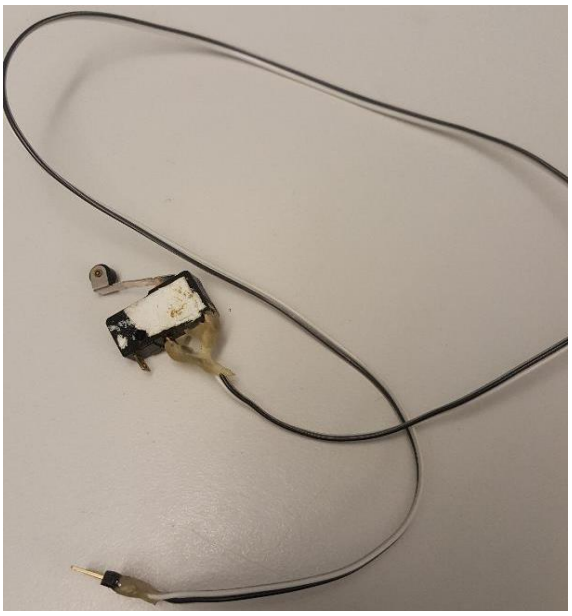

Abbildung 6: Taster

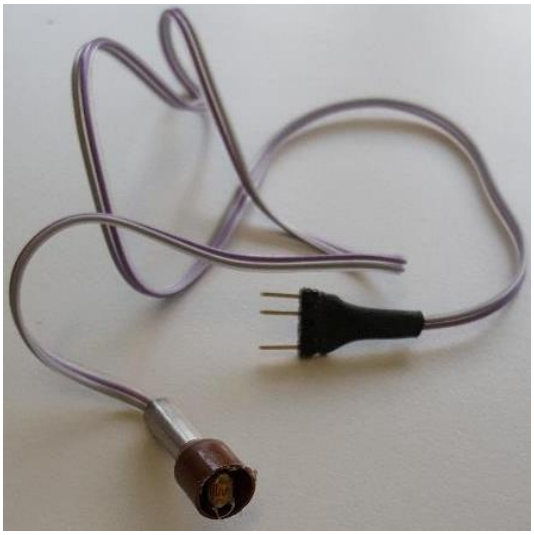

Abbildung 7: Lichtsensor

2.2 Methode

Studierenden sind Lego-Bausteine zur Verfügung gestellt worden. Anhand dieser Bausteine haben sie selber in zweier Gruppen ihre Autos gebaut. Nach mehreren Stunden bzw. Tagen ist unser Auto zustande gekommen. Am Anfang wurde nach der Seriennummer des Legos gegoogelt, um das Auto zu bauen. Nach einiger Zeit, wenn das Auto fast fertig war, wurde festgestellt, dass das Automodell abgesehen seiner Länge (zu lang und zwar länger als eine Straßenkante) gar nicht zu der Umsetzung der Aufgabe passen wird. Aus diesem Grund wurde es komplett abgebaut.

Ein neues Modell (*Abbildung 1*) ist auf Basis eines anderen Autos gebaut worden, das von Studierenden der vorherigen Studienjahre aufgebaut wurde. Nachdem das Auto fertig war, wurde nach der Robustheit des Autos geprüft, d.h. wie fest die Bausteine zusammengebracht wurden, damit das Auto beim Stoßen auf irgendetwas im Straßennetz nicht auseinandergeht. Es wurde leider noch einige Kleinigkeiten festgestellt, die dazu geführt haben, dass das Auto ein paar Mal abgebaut werden musste wie zum Beispiel die Tatsache, dass Getriebe nicht richtig montiert waren oder ein Motor kaputt war.

Bei den ersten Tests wurde bemerkt, dass der eine oder der andere Sensor nicht die richtig funktioniert hat. Es hat sehr lange gedauert bis das Auto gerade fahren konnte bzw. auf eine Straße fahren konnte. Es lag an Bauproblemen des Autos, denn es war einerseits nicht richtig fest gebaut, andererseits wackelte es von vorne nach hinten und umgekehrt. Deswegen wurden zwei Räder jeweils vorne und hinten als Stützpunkte hinzugefügt.

3 Implementierung bzw. Umsetzung

3.1 Flutung

Die Implementierung ist sehr systematisch gewesen. Am Anfang musste das Auto geradeaus fahren oder eine Linie (spiegelt hier eine Straße wieder) verfolgen können. Weiterhin musste eine Acht gefahren werden. Das Ziel dabei ist gewesen, das Links- bzw. Rechts- oder auch noch Geradeausfahren zu üben, was dazu gebracht hat, die Funktionsfähigkeit der Sensoren zu prüfen. Dann musste jede Gruppe einen Vortrag darüber halten, wie das Projekt algorithmisch realisiert werden könnte. Unsere Gruppe ist auf Basis von drei unterschiedlichen Suchverfahren gegangen, die uns im Fach „Grundlage der Wissensverarbeitung“ bekannt gemacht worden sind. Es sind einerseits die Tiefensuche, andererseits die Breitensuche und darüber hinaus die A*-Suche. Nach Analyse und Interpretation dieser Verfahren bezüglich der im Projekt dargestellten Aufgabe, hat sich herausgestellt, dass die Breiten- und A*-Suche die zwei für die algorithmische Umsetzung der Aufgabe angemessenen Suchverfahren sind, abgesehen von ihrer Optimalität d.h. dass sie als erstes die beste Antwort oder besser gesagt die beste Route liefern. Da sich unsere Gruppe für die Breitensuche entschlossen hat, wurde diese Suche nach einem bestimmten Schema implementiert.

Das Vorgehen zur Implementierung der Breitensuche wurde von jeder Gruppe vorgestellt. In diesem Sinne hat eine Gruppe einen sehr gut durchdachten Algorithmus vorgestellt, dessen Implementierung von Prof. Heinsohn und Herrn

Boersch empfohlen wurde. Es geht prinzipiell zuerst darum, den vorgegebenen Fahrplan zu fluten d.h. den Fahrplan (vom Startpunkt ausgehend) mit Zahlen von 0 (Am Startpunkt) bis höchstens 70 auszufüllen (**Eine Excel-Tabelle oder eine Tabelle auf Papier als Beispiel dazu machen oder Scrnshot vom Bildschirm**). Dabei gibt es mehrere Möglichkeiten. Man kann direkt den Fahrplan fluten oder man benutzt eine Art Hilfskarte bzw. Flutkarte, die dem Fahrplan gleich groß ist. Da es im Fahrplan gesperrte Kreuzungen gibt, werden diese Kreuzungen durch eine bestimmte Zahl (71) abgebildet (Siehe ***initFlukarte ()***).

3.2 Lieferorte

Nach dem Fluten kommt die Suche der Lieferorte. Dafür musste der Fahrplan auf der Suche nach „Fs“ durchgelaufen werden (Siehe ***lieferOrteFinden ()***) . Wenn ein „F“ an einem bestimmten Index gefunden wird, entspricht der Index genau dem Index in der Flutkarte. Der Kosten, der in diesem Index steht, wird in Betracht genommen und es wird angefangen, ab diesem Kosten runter zu zählen, bis der Startpunkt (Kosten 0) erreicht wird. Daraus resultiert eine Kostenroute vom Zielknoten zum Startknoten. Da vom Startknoten zum Zielknoten gefahren werden muss, bleibt nichts übrig als diese Kostenroute umzudrehen (Die Kostenroute wird in einem Array gespeichert, daher ist das Umdrehen kein Problem). Die absoluten Richtungen (Norden, Süden, Westen und Osten), die nichts mit den möglichen Richtungen (Geradeaus, Links und Rechts) des Autos zu tun haben, mussten ebenfalls betrachtet werden. Für diese absoluten Richtungen wurden jeweils die Werte 0, 180, -90 und 90 ausgewählt und vergeben (Siehe ***setAbsolutWay(char route[], int kompass[])***). Die absolute Ausgangsrichtung des Autos ist Norden. Davon ausgegangen, muss die aktuelle Position des Autos stets gemerkt werden. Die Richtungen Geradeaus (G), Links (L) und Rechts (R) sind aus der Formel (**Soll - Ist**) berechnet worden, wobei der Soll die Richtung ist, in die gefahren werden soll und der Ist die Richtung, in die das Auto guckt. Wenn das Ergebnis aus der obigen Formel:

- 0 gleich ist, ist die Fahrtrichtung gerade (G)
- 90 gleich ist, ist die Fahrtrichtung rechts (R)
- -90 gleich ist, ist die Fahrtrichtung links (L)

Beispiel:

Nehmen wir an, eine Route wäre „NNW“ (Norden-Norden-Süden). Umgerechnet kommt die Kostenroute „0 - 0 -90“ raus. Abgesehen davon, dass die Ausgangsrichtung des Autos der Norden (also Kosten 0) ist, wird die ite Fahrrichtung nach der oben genannten Formel ausgerechnet, wobei der Sollwert der ite Wert in der Kostenroute ist.

Für die Berechnung der ersten Fahrrichtung kommt der Befehl „geradeaus (G)“ als Ergebnis raus, denn Soll – Ist (0 - 0) ist gleich 0 d.h. keine Richtungsänderung. Nach dem Durchlauf der oben als Beispiel genommenen Route bekommt fährt das Auto nach dem Befehl „GGL“.

3.3 Funktionen

Es wurde vier Methoden zu den möglichen Fahrtrichtungen implementiert. Die Methode „***geradeaus ()***“ hilft dabei, wie der Name schon sagt, geradeaus zu fahren. Sie wird verlassen, sobald eine Kreuzung erkannt wird. Sie ruft die Methode „***followLine ()***“ auf, die dazu dient, eine Linie bzw. Straße verfolgen zu können. Für das Links- bzw. Rechtsabbiegen sind jeweils die Methoden „***linksAbbiegen*** ()“ und „***rechtsAbbiegen ()***“ implementiert worden. Mit Hilfe eines Interpreters (Die Methode heißt ***planVorwaerts (char route[])***) werden die drei Routen nach und nach gelesen. Zuerst wird die Route zum ersten Lieferort interpretiert. Nachdem eine Pizza geliefert wird, muss das Auto bis zum Startpunkt zurückfahren und die nächste Pizza wird dort händisch beladen damit das Auto sie zum zweiten Lieferort bringt usw.

Das Zurückfahren impliziert nun, dass die Route vom Startpunkt zum Lieferort anders rumgefahren werden muss d.h. vom Lieferort zum Startpunkt. Aber das Auto muss sich davor am Lieferort umdrehen. Dieses Umdrehen kann schieflaufen, wenn die falsche Linie als erstes von einem Sensor beim Umdrehen getroffen wird. Deswegen ist die Idee gewesen, das Auto solange rückwärtsfahren zu lassen, bis ein Befehl links- bzw. rechtsabbiegen kommt. Das bedeutet beispielsweise, dass das Auto zum Startpunkt nur rückwärtsfahren wird, wenn es vom Startpunkt aus bis zum Lieferort nur geradeaus gefahren ist.

4 Ergebnisse

Die Implementierung im Allgemeinen ist sehr komplex gewesen. Jede Etappe ist eine große Herausforderung gewesen. Es wurde am Anfang an der Flutung und dem Durchsuchen aller Routen zu den einzelnen Lieferorten sehr viel Zeit verbracht. Trotzdem lief alles gut bis die zweite und die dritte Route im Aksen-Board nicht betrachtet wurden d.h. es ließ sich nur die erste Route lesen. Nach Feststellung der knappen Zeit wurde die Implementierung des Plans zur Seite geschoben und es wurde an dem Interpreter weitergearbeitet, damit das Auto beim Anmachen des Lichts im Straßennetz startet, bei der Ablieferung der Pizza den Greifer auf und danach zumacht. Das Umdrehen und das Rückwärtsfahren sind die Stellen gewesen, woran das Projekt gehapert hat. Daran wurde sehr viel Zeit verbracht, obwohl das Rückwärtsfahren am Ende nicht hundert Prozent gelungen ist. Das was richtig funktioniert hat, ist das Finden der Lieferorte anhand der Breitensuche, wobei nur die erste Route im Aksen-Board gelesen werden konnte. Der Grund warum die zwei anderen Routen nicht gelesen werden konnten, ist unbekannt geblieben. Es gab einen Verdacht auf die Speicherkapazität des Aksen-Boards, aber es hat gar nicht daran gelegen, dass die zwei anderen Routen nicht gelesen werden konnten, denn das Aksen-Board eine Speicherkapazität von 6900 Bytes. Deswegen wurde Schluss gefolgert, dass es an dem Aufbau des Autos oder vielleicht an der Implementierung der Methoden „***rueckfahrt ()***“ und „***umdrehen ()***“ (was sehr bezweifelt wurde) liegen könnte. Am Tag der Projektvorstellung, ist nichts gut gelaufen, denn das Auto hat nichts anderes getan als geradeaus zu fahren. Das würde heißen, dass das Aksen-Board eine leere Route bekommen hätte (Siehe ***planVorwaerts()***).

5 Zusammenfassung

Es ging im Projekt darum, ein völlig autonomes Fahrzeug zu bauen und dieses in Betrieb zu nehmen, was fähig wäre, Pizzen zu liefern. Am Ende des Projekts entstand ein kleiner Wettbewerb, wobei Punkte je richtig gelieferte Pizza vergeben wurden. Ein Auto ist ausgeschieden, wenn es Bausteine verliert. Nachdem die Breitensuche als Suchverfahren ausgewählt wurde, musste diese von jeder Gruppe implementiert werden. Das von unserer Gruppe gebaute Auto hatte am Anfang viele technische Probleme. Es war instabil und die Getriebe musste gewechselt werden. Darüber hinaus ist einer der Motoren kaputt gegangen und die Sensoren funktionierten nicht richtig. Das Ganze führte dazu, dass viel Zeit verbraucht wurde, um diese Probleme zu lösen. Was die Implementierung der Breitensuche angeht, hat sie ebenfalls lange gedauert. Vor der Implementierung der Breitensuche wurde theoretisch zur Kenntnis genommen, dass die Breitensuche einen sehr großen Speicherbedarf hat. Diese Theorie hat sich jedoch durch die Umsetzung dieses Projekts anhand der Anzahl verwendeter Arrays bewiesen lassen. Es wurde erst während der Implementierung festgestellt wie aufwändig das Programm gewesen ist. Das Auto war eine Woche vor dem Wettbewerb nur in der Lage eine Pizza zu liefern und wir waren über den Rückweg unsicher, abgesehen von den Problemen mit dem Umdrehen. Das Projekt war sehr interessant, aber auch dadurch sehr stressig, dass die Sachen nicht wie gewünscht funktioniert haben. Außerdem wurde viel Zeit in diesem Projekt investiert.

Quellcode

```
/*@Autoren:
* Robert Kengmogne Kamga
* Landry Ngueyep Kabes
*/

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

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

#define WERT1 100
#define FEUER 7
#define WERT3 0
unsigned int kreuzungZaehler = 0, indexAgenda = 0;
//unsigned char _plan[] = "GGRGGLGGLGGLGGLGGRGGRGG";
#define SIZE_X 7
#define SIZE_Y 10
#define MAX (SIZE_X * SIZE_Y)
#define INIT (MAX + 1)

int start = 68, neuStart = 0; //or 64 for the other start point

//Lieferorte, wo Fs im Fahrplan stehen
int lieferOrte[3];

int aktAbsRichtungRobot = 0;


//erste Route vom Zielknoten Nr. 1 zum Startknoten
char route1[20];
//erste Route vom Startknoten zum Zielknoten Nr. 1
char route11[20];
//echte erste Lieferroute
//char echteRoute111[20];
//zweite Route vom Zielknoten Nr. 2 zum Startknoten
char route2[20];
//zweite Route vom Startknoten zum Zielknoten Nr.2
char route22[20];
//echte zweite Lieferroute
//char echteRoute222[20];
//dritte Route vom Zielknoten Nr. 3 zum Startknoten
char route3[20];
//dritte Route vom Startknoten zum Zielknoten Nr.3
char route33[20];
//echte dritte Lieferroute
```

```
//char echteRoute333[20];

char zufahrendeRoute[20];

int startIndex = 0, h = 0, n = 0, m = 0, l = 0,
r = 0, c = 0, j = 20, b = MAX, g = 0, indexRueckwaerts = 20, indexMain = 0, indexZufahrendeRoute = 0;
int kosten = 1;

int osten = 0, norden = 0, westen = 0, sueden = 0;

unsigned int flutKarte[MAX];
unsigned char hilfskarte[MAX];
int _agenda[MAX] = { 0 };
int _agendaFlutkarte[MAX] = { 0 };

int _kompass1[20] = { 0 };
int _kompass2[20] = { 0 };
int _kompass3[20] = { 0 };
//unsigned char _fa[] = "xxFxFxxx..x..xF.x.x.Fx.x...xx.x..xxx..x..xx..x..xxx.x..xF..x..Fx..x..x";
//unsigned char _fa[] = "xFxxxFxx.x.x.xF..x..Fx..x..xx..x..xx.x.x.xF..x..Fx..x..xx..x..xx.x.x.x";
//unsigned char _fa[] = "xxFxFxxx..x..xF.x.x.Fx.x...xx.x..xxx..x..xx...x.xxx..x.xF..x..Fx..x..x";
//unsigned char _fa[] = "xxxFxxxx.....xx..x..xx..x..xF..x..Fx..x..xx..x..xxx.x.xxx..x..xx..x..x";
//unsigned char _fa[] = "xFxxxFxx.x...xx..x.xxxx.x..xF..x..Fxx.x.xxx..x..xx.x..xxF..x..Fx...x.x";
unsigned char _fa[] = "xxxxxxxx..x..xx..x..xF..x..Fxx.x.xxF..x..Fx.xxx.xF..x..Fxx.x.xxx..x..x";



void setAbsolutWay(char route[], int kompass[])
{
	for (; m < 20; m++)
	{
		if (('N' == route[m]))
		{
			kompass[m] = 0;
		}
		if ('S' == route[m])
		{
			kompass[m] = 180;
		}
		if ('O' == route[m])
		{
			kompass[m] = 90;
		}
		if ('W' == route[m])
		{
			kompass[m] = -90;
		}
	}
	m = 0;
}
```

```
void fahrrichtungen(char route[], int kompass[])
{
	for (; n < 20; n++)
	{
		if ((route[n] != 0))
		{
			if (kompass[n] - aktAbsRichtungRobot == 0)
			{
				route[n] = 'G';
				aktAbsRichtungRobot = kompass[n];
			}
			if (kompass[n] - aktAbsRichtungRobot == 90)
			{
				route[n] = 'R';
				aktAbsRichtungRobot = kompass[n];
			}
			if (kompass[n] - aktAbsRichtungRobot == -90)
			{
				route[n] = 'L';
				aktAbsRichtungRobot = kompass[n];
			}
			/*if (kompass[i] - aktAbsRichtungRobot == 180)
			{
			route[i] = 'L';
			aktAbsRichtungRobot = kompass[i];
			}*/
		}
	}
}
//Flutkarte mit dem Wert 71 initialisieren
void initFlukarte()
{

	for (; l < MAX; l++)
	{
		flutKarte[l] = INIT;
	}
	flutKarte[64] = 0;
	l = 0;
}

//ertes Element der Agenda zurückgeben und dieses Element aus der Agenda löschen
int deleteFirstFromAgenda(int eineAgenda[])
{
	//char aktknoten = ' ';
	int aktknoten = eineAgenda[0];

	for (; r < MAX - 1; r++)
	{
```

```
                eineAgenda[r] = eineAgenda[r + 1];
        }
        indexAgenda--;
        r = 0;
        return aktknoten;
}

//Aktknoten expandieren und gleichzeitig in die Agenda und in die Flutkarte eintragen
void expandUndFluten()
{
        _agenda[indexAgenda++] = start;
        while (indexAgenda > 0)
        {
                int aktknoten = deleteFirstFromAgenda(_agenda);
                kosten = flutKarte[aktknoten] + 1;
                osten = aktknoten + 1;
                norden = aktknoten - 7;
                westen = aktknoten - 1;
                sueden = aktknoten + 7;
                if (osten < MAX && (osten % 7 != 6))
                {
                        if (('.' == _fa[osten]) || ('F' == _fa[osten])) && flutKarte[osten] == INIT)
                        {

                                flutKarte[osten] = kosten;
                                _agenda[indexAgenda++] = osten;
                        }
                }
                if ((westen >= 0))// && (z % 7 != 0))
                {
                        if (('.' == _fa[westen]) || ('F' == _fa[westen])) && (flutKarte[westen] == INIT))
                        {

                                flutKarte[westen] = kosten;
                                _agenda[indexAgenda++] = westen;
                        }

                }
                if ((norden >= 0) && (norden % 7 != 0))
                {
                        if (('.' == _fa[norden]) || ('F' == _fa[norden])) && (flutKarte[norden] == INIT))
                        {
                                flutKarte[norden] = kosten;
                                _agenda[indexAgenda++] = norden;
                        }

                }

                if ((sueden <= MAX))
                {
                        if (('.' == _fa[sueden]) || ('F' == _fa[sueden])) && (flutKarte[sueden] == INIT))
                        {
```

```
                        flutKarte[sueden] = kosten;
                        _agenda[indexAgenda++] = sueden;
                    }

                }
            }
}

//Route vom Ziel zum Startpunkt herausfinden
void routeAusfuellen(char routenNummer[], int ortNummer)
{

    int i = 0;
    _agendaFlutkarte[indexAgenda++] = lieferOrte[ortNummer];


    while (indexAgenda > 0)
    {
        int ort = deleteFirstFromAgenda(_agendaFlutkarte);
        int osten = ort + 1, norden = ort - 7, westen = ort - 1, sueden = ort + 7;
        if (osten < MAX && (osten % 7 != 6))
        {
            if ((flutKarte[osten] < flutKarte[ort]))
            {
                routenNummer[i] = 'O';
                _agendaFlutkarte[indexAgenda++] = osten;
            }
        }
        if ((westen >= 0))
        {
            if ((flutKarte[westen] < flutKarte[ort]))
            {

                routenNummer[i] = 'W';
                _agendaFlutkarte[indexAgenda++] = westen;
            }

        }
        if ((norden >= 0) && (norden % 7 != 0))
        {
            if ((flutKarte[norden] < flutKarte[ort]))
            {
                routenNummer[i] = 'N';
                _agendaFlutkarte[indexAgenda++] = norden;
            }

        }

        if ((sueden <= MAX))
        {
            if ((flutKarte[sueden] < flutKarte[ort]))
```

```
                        {
                                routenNummer[i] = 'S';
                                _agendaFlutkarte[indexAgenda++] = sueden;
                        }

                }
                i++;
        }
}



void swapRoute(char route[], char routeUmzuDrehen[])
{
        for (; (c < 20 && j >= 0); j--)
        {
                if ((routeUmzuDrehen[j] != 0) && (routeUmzuDrehen[j] == 'S'))
                {
                        route[c] = 'N';
                        c++;
                }
                else if ((routeUmzuDrehen[j] != 0) && (routeUmzuDrehen[j] == 'N'))
                {
                        route[c] = 'S';
                        c++;
                }
                else if ((routeUmzuDrehen[j] != 0) && (routeUmzuDrehen[j] == 'O'))
                {
                        route[c] = 'W';
                        c++;
                }
                else if ((routeUmzuDrehen[j] != 0) && (routeUmzuDrehen[j] == 'W'))
                {
                        route[c] = 'O';
                        c++;
                }
        }
}

/*
*Entsprechende Route abhängig vom Ort ausfuellen
*Es gibt höchstens drei Routen, denn es gibt höchstens drei Lieferorte
*/
void routeFinden(int ortNummer)
{
        if (ortNummer == 0)
        {
                //routenFuellen(route1);
                routeAusfuellen(route1, ortNummer);

                //route1 umdrehen
                swapRoute(route11, route1);
```

```
                setAbsolutWay(route11, _kompass1);

                //aktAbsRichtungRobot = 0;
                fahrrichtungen(route11, _kompass1);
        }
        if (ortNummer == 1)
        {
                //routenFuellen(route2);
                routeAusfuellen(route2, ortNummer);

                //route2 umdrehen
                swapRoute(route22, route2);

                setAbsolutWay(route22, _kompass2);

                aktAbsRichtungRobot = 0;
                //fahrrichtungen(route22, echteRoute222, _kompass2);

                fahrrichtungen(route22, _kompass2);
        }
        if (ortNummer == 2)
        {
                //routenFuellen(route3);
                routeAusfuellen(route3, ortNummer);

                //route1 umdrehen
                swapRoute(route33, route3);

                setAbsolutWay(route33, _kompass3);

                aktAbsRichtungRobot = 0;
                //fahrrichtungen(route33, echteRoute333, _kompass3);
                fahrrichtungen(route33, _kompass3);
        }
}



//LieferOrt herausfinden --> Alle Fs im Fahrplan
void lieferOrteFinden()
{

        /*Laut dem Startpunkt sollte der Fahrplan von unten nach Oben durchgelaufen sein, damit
        *die optimale Lösung
        *geliefert wird*/
        for (; (b > 0 && g < 3); b--)
        {
                if (('F' == _fa[b]) && (flutKarte[b] != 71))
                {
                        lieferOrte[g] = b;
                        routeFinden(g);
```

```
                        g++;
                }
        }
        b = 0;


}
void rechtsAbbiegen()
{
        /*
        *Analog6 == rechts
        *Analog0 == links
        */
        motor_richtung(1, 0);
        motor_richtung(0, 0);
        lcd_cls();
        lcd_puts("Rechts abbiegen");
        //Rechter Motor
        motor_richtung(0, 1);
        motor_pwm(0, FEUER);
        sleep(20);
        motor_pwm(0, 0);
        motor_pwm(1, FEUER);
        sleep(700);
        while (analog(14) < WERT1);
        }
void linksAbbiegen()
{
        /*
        *Analog6 == rechts
        *Analog0 == links
        */
        motor_richtung(1, 0);
        motor_richtung(0, 0);
        lcd_cls();
        lcd_puts("Links abbiegen");

        motor_richtung(1, 1);
        motor_pwm(1, FEUER);
        sleep(20);
        motor_pwm(1, 0);
        motor_pwm(0, FEUER);
        sleep(700);
        while (analog(12) < WERT1);
        //sleep(100);
}

void followLine()
{
        //int i = 0;
        //Rechter Motor
        motor_richtung(0, 0)
```

```
	//Linker Motor
	motor_richtung(1, 0);

	if ((analog(14) > WERT1) && (analog(12) > WERT1))
	{

		motor_pwm(0, FEUER);
		motor_pwm(1, FEUER);
	}
	//Analog(14) = rechts
	else if (analog(14) > WERT1)
	{

		motor_pwm(0, 0);
		motor_pwm(1, FEUER);
	}
	//Analog(12) = rechts
	else if (analog(12) > WERT1)
	{

		motor_pwm(1, 0);
		motor_pwm(0, FEUER);
	}

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


}
void followLineRueckwaerts()
{

	if ((analog(14) > WERT1) && (analog(12) > WERT1))
	{

		motor_pwm(0, FEUER);
		motor_pwm(1, FEUER);
	}
	//Analog(14) = rechts
	else if (analog(14) > WERT1)
	{

		motor_pwm(0, 0);
		motor_pwm(1, FEUER);
	}
	//Analog(12) = rechts
	else if (analog(12) > WERT1)
	{
```

```
                motor_pwm(1, 0);
                motor_pwm(0, FEUER);
        }

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

}
void geradeAusFahren()
{
        lcd_cls();
        lcd_puts("plan");
        while ((analog(6) < WERT1 && analog(0) < WERT1))
        {
                followLine();
                if (digital_in(0) == 0)
                {
                        otoKoppler();
                }
        }
}
void rueckFahrt()
{

        while ((analog(6) < WERT1 && analog(0) < WERT1))
        {
                followLineRueckwaerts();
        }
}
/*hat leider nicht richtig funktioniert*/
void planRueckwaerts(char route[])
{
        /*lcd_cls();
        lcd_setxy(0, 0);
        lcd_ubyte(digital_in(0));*/

        lcd_cls();
        lcd_ubyte(route22[0]);
        for (; indexRueckwaerts >= 0; indexRueckwaerts--)
        {

                if ('G' == route[indexRueckwaerts])
                {
                        geradeAusFahren();
                        sleep(200);

                }
                else if ('L' == route[indexRueckwaerts])
                {
```

```
                linksAbbiegen();
            }
            else if ('R' == route[indexRueckwaerts])
            {
                rechtsAbbiegen();
            }
            else
            {

            }

        }
            indexRueckwaerts = 20;
}
void planVorwaerts(char route[])
{
        /*lcd_cls();
        lcd_setxy(0, 0);
        lcd_ubyte(digital_in(0));*/
        lcd_cls();
        lcd_ubyte(route22[0]);
        for (; h < 30; h++)
        {
            geradeAusFahren();

            /*lcd_cls();
            lcd_puts("plan");*/
            if ('G' == route[h])
            {
                geradeAusFahren();
                sleep(200);

            }
            else if ('L' == route[h])
            {
                linksAbbiegen();
            }
            else if ('R' == route[h])
            {
                rechtsAbbiegen();
            }
            else
            {
                h = 30;
            }

        }

        h = 0;
}
void lichtAn(){
        if (analog(8) > 100)
```

```
	{
		motor_pwm(0, 0);
		motor_pwm(1, 0);
		lcd_puts("Bitte Licht an!");
	}
}
void otoKoppler()
{
	lcd_cls();
	lcd_setxy(0, 0);
	lcd_ubyte(digital_in(0));
	servo_arc(2, 1);

		lcd_puts("greiferAuf");
		motor_pwm(0, 0);
		motor_pwm(1, 0);
		servo_arc(2, 80);
		sleep(2000);
		motor_richtung(1, 1);
		motor_richtung(0, 1);
		rueckFahrt();
		lcd_puts("greiferZu");
		servo_arc(2, 1);
		//sleep(1000);
		//rueckFahrt();



}

void AksenMain(void)
{
	while (1)
	{

		servo_arc(2, 1);
		initFlukarte();
		expandUndFluten();
		lieferOrteFinden();
		lichtAn();
		motor_pwm(0, FEUER);
		motor_pwm(1, FEUER);
		planVorwaerts(route11);
		sleep(10000);

	}

}
```