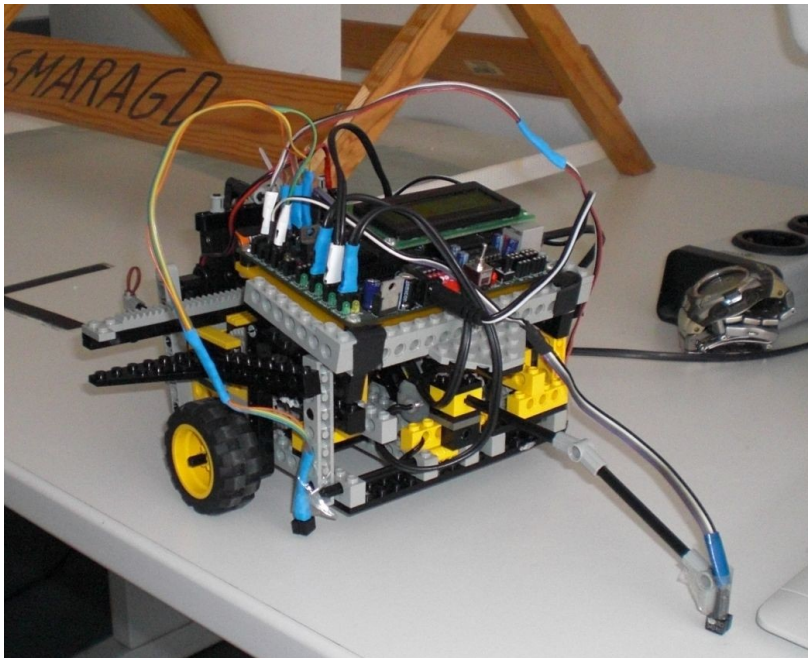

# Projekt-Bericht – Autonome Mobile Systeme

Wintersemester 2010/2011




Bertram Sändig 2008/2078
Paul Lehmann 2008/2023

FACHHOCHSCHULE BRANDENBURG

# Inhaltsverzeichnis

# Aufgabe / Spielfeld

Die Aufgabe bestand darin, seinen selbstgebauten Lego-Roboter durch ein Labyrinth zu führen. Auf diesem Spielfeld (s. Abb. 1) gab es schwarze Linien, die man als Orientierung benutzen kann oder man ignoriert sie einfach. Die Roboter sollten losfahren, sobald ein Licht an den beiden Startpunkten angeschalten wurde. War dies geschafft, so erhielt man den Lebenspunkt. Anschließend sollten sie sich durch eben diesen Parcours bewegen und dabei möglichst noch die Wegpunkte abfahren, die auf der Strecke verteilt waren. Jeder Wegpunkt brachte einem weitere drei Punkte. Am Ende der Strecke gab es dann die Methanmoleküle, bei denen man das Kohlenstoff mithilfe seines Roboters abspalten sollte, sodass nur noch die Wasserstoffatome da sind. Jedes erfolgreich getrennte Kohlenstoffatom brachte einem zehn Punkte. Um die Sache etwas zu erschweren, wurden die letzten zwei schwarzen Linien, die zum Ende hin führen weggenommen, wodurch man sich etwas anderes einfallen lassen muss, wenn man sich vorher an Linien orientiert hat.

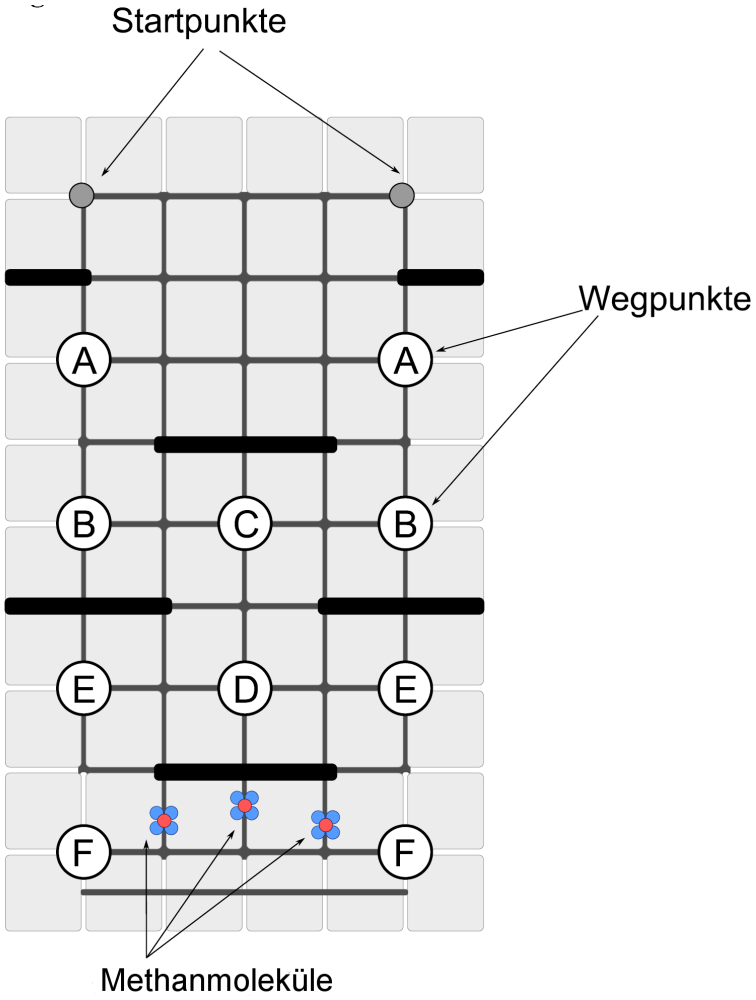



Abb. 1: Spielfeld © FHB

# Vorüberlegung

## *Aufgabenanalyse*

Der Roboter muss augenscheinlich zur Navigation mithilfe von Bodenlinien fähig sein. Des weiteren ist es auch nötig, dass er, zumindest für eine kurze Strecke, ohne Hilfsmittel in der Umgebung, sogenannte „Landmarken“ zurechtkommt. Er benötigt eine Mechanik, die die „Methanmoleküle“ von ihren Sockeln stoßen kann. Diese muss eine recht große Spannweite haben, darf aber während der Fahrt den Durchmesser des Roboters nicht so erhöhen, dass dieser zu klobig wird.

Geschwindigkeit ist ganz offensichtlich auch ein entscheidender Faktor. Wenn man von zwei fehlerfrei navigierenden Robotern ausgeht, gewinnt schließlich der schnellere.

## *Routenplanung*

A)

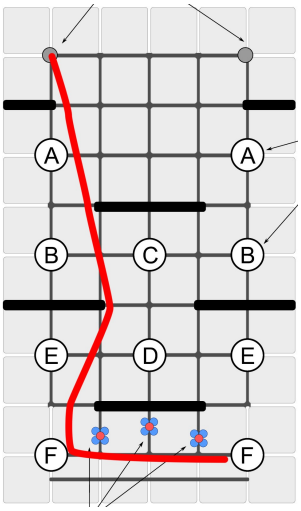



Abb. 2: Querfeldein

B)

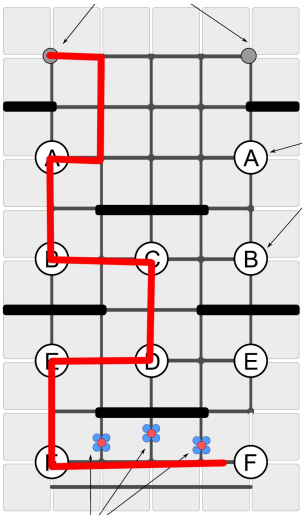



Abb. 3: Linie folgen

Während der Diskussion um die perfekte Route, kamen schnell zwei Varianten in die engere Auswahl. Route A (s. Abb. 2), die einige Wegpunkte aus lässt, um in kürzerer Zeit zu den drei Hauptzielen zu gelangen und Route B (s. Abb. 3), die Standartroute über jeden Wegpunkt zum Ziel.

Wir entschieden uns für Variante B. Nicht so sehr aus Respekt vor den Schwierigkeiten einer linienfreien Navigation, sondern weil sich bereits herauskristallisierte, dass unser Roboter sich schneller bewegen würde als die meisten anderen Modelle.

In der Retrospektive sind wir sehr glücklich mit dieser Entscheidung, da uns zu diesem frühen Zeitpunkt der Umfang einer „Querfeldein“-Navigation wahrscheinlich nicht komplett bewusst war.

## *Navigation*

In den ersten naiven Versuchen experimentierten wir mit einer puren Zeitsteuerung (2 s vor 1 s rechts 2 s vor etc.). Die Unschärfe, die sich bei dieser Methode anhäufte, war vollkommen unhaltbar. Machten die ersten Kurven noch einen fast perfekten Eindruck, und vermittelten die Illusion, diese Art der Navigation hätte Validität, so war bei längeren Strecken schnell klar, dass sich Fehler in unhaltbarem Maße aufschaukelten und die Endposition des Roboters unvorhersehbar wurde.

Wir wehrten uns aber weiter gegen eine über die ganze Zeit sensorüberwachte Steuerung, denn wir wussten, so etwas kostet Geschwindigkeit. Also entschieden wir uns für eine Mischung bei der wir teilweise ohne Sensorik fahren und uns dann an bestimmten Punkten neu ausrichten würden.

**Variante 1**

Bei unserem ersten Lösungsansatz sollte der Roboter mithilfe zweier Sensoren, die, sich gegenüberliegend, am Vorderteil des Roboters angebracht waren, auf eine Linie ausgerichtet werden.

(trifft der rechte Sensor zuerst auf eine Linie bleibt das rechte Rad stehen und das linke fährt allein bis der linke Sensor die Linie erreicht (s. Abb. 4))

Problem dieser Methode ist, dass wir zwar unseren Winkel immer wieder korrigieren können, aber eben auch nur unseren Winkel.

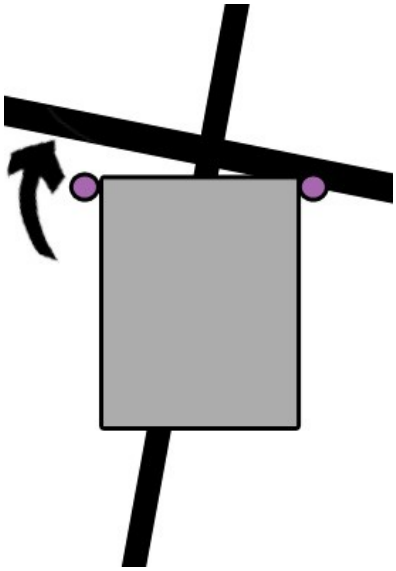


Abb. 4: Roboter trifft rechts auf schwarze Linie

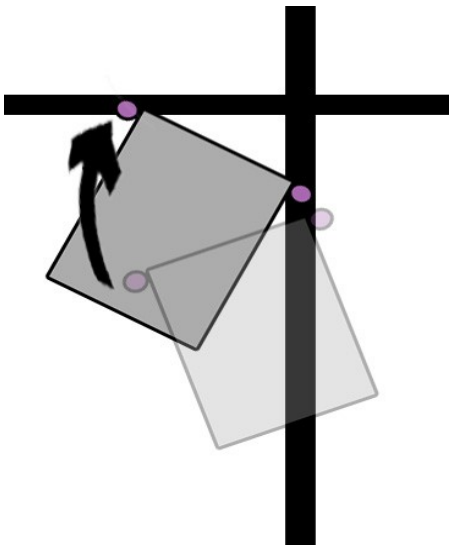


Abb. 5: Problem beim Abkommen von der Linie

Es konnte außerdem ein schwerer Fehler auftreten falls der Roboter erst auf eine senkrechte Linie stieß (s. Abb. 5).

**Variante 2**

Wir verwendeten weiterhin die zwei bereits erwähnten vorderen Sensoren und erweiterten das ganze um einen ausgelagerten mittigen „Teleskop“- oder auch „Rüsselsensor“. Wir fuhren ab jetzt mithilfe der alten Sensoren an die Querlinien und nutzen den neuen Sensor um uns auf die Senkrechten Linien (und darüber hinaus) auszurichten (s. Abb. 6). Zwar waren wir uns ab da an in keinem Moment über den tatsächlichen Winkel oder die Position des Robotermittelpunkts sicher, wir wussten aber das beides regelmäßig um die Linien, an denen sich ausgerichtet wurde, pendeln würde, wodurch wir immer in einem akzeptablen Bereich auf Kurs blieben.

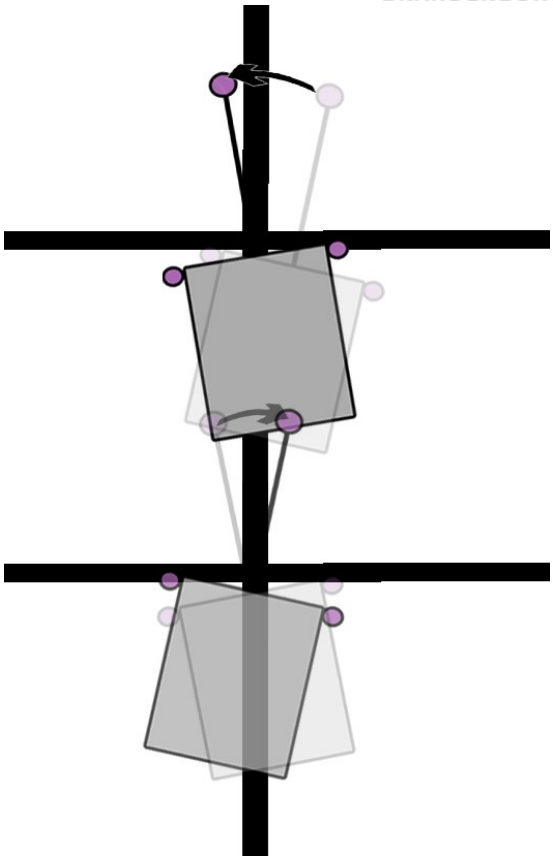


Abb. 6: Pendeln an der Linie

Wir waren sofort sehr überzeugt von der Effizienz aber noch viel mehr vom Chaos dieser Methode.

(Wenn man durch einen stockdunklen, engen Tunnel geht, muss man sich genaugenommen nicht an den Wänden entlangtasten. Man kann auch wild drauflos rennen und jede Kollision bringt einen wieder auf Kurs.)

## Fahrstil

Wenn man den Roboter von oben bei der "Arbeit" filmen würde, dann würde die Fahrt in etwa der roten Linie entsprechen (s. Abb. 7).

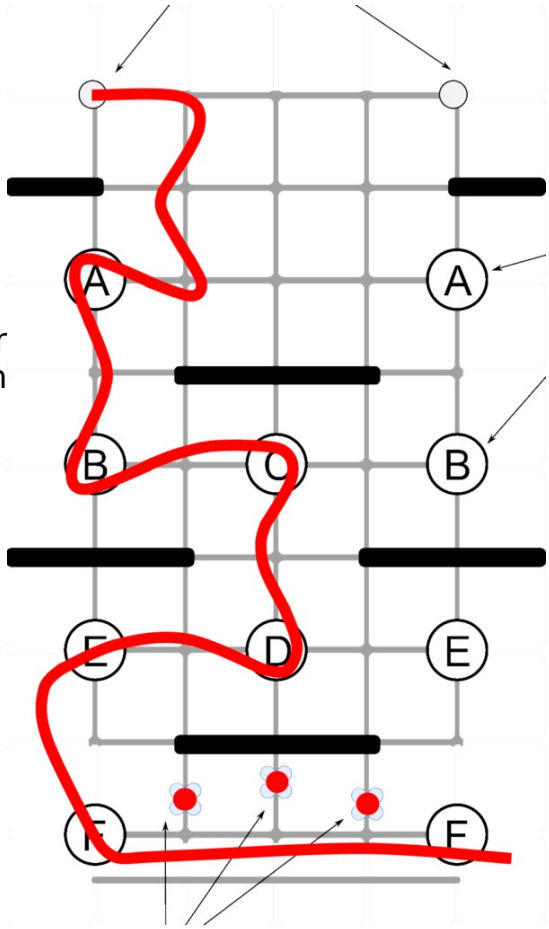



Abb. 7: Die „Ideallinie“

# Probleme

## "Beim letzten Mal gings noch"

Als uns beim ersten Termin erzählt wurde, dass dies der Satz sei, der in diesem Raum am häufigsten gesagt wird, mussten wir alle ein wenig schmunzeln, aber aus Freude wurde bittere Realität. Ein ums andere Mal hat der Roboter das, was er gerade eben noch einwandfrei gelöst hat, plötzlich vollkommen anders gemacht. Ob er gegen Wände fuhr, oder manche Linien nicht sah/sehen wollte, weil ihm eben nicht danach war, war oft nicht ganz klar.

Unsere Probleme waren oft die, dass wir aufgrund unserer Fahrweise verpflichtet waren schief auf eine Gerade zu kommen. Wenn wir nämlich zu gerade aus der Kurve kamen, dann hat sich der Roboter unter Umständen an der nächsten Kreuzung in die falsche Richtung korrigiert und ist vollkommen vom Kurs abgekommen. Die Lösung des Problems ist, wenn man einmal verstanden hat, worin das Problem lag, relativ simpel. Der Roboter fährt bevor er abbiegt immer noch ein kurzes Stück geradeaus, dass wir sicher sein können, dass er über die Kreuzung gefahren ist und somit schief auf die Gerade kommt.

Ein weiteres Problem bestand darin, dass wir stets versucht haben schnell zu bleiben und das auch in den Kurven, was dazu führte, dass unser mittlerer Sensor zum Teil die Linien nicht gesehen hat und einfach über sie rutschte. Dadurch ist der Roboter hin und wieder einfach in die Richtung zurückgefahren aus der er kam oder hat sich unaufhörlich auf der Stelle gedreht. Die Lösung dieses Problems hat uns nicht gefallen, da wir die Geschwindigkeit in den Kurven reduzieren mussten um die Linien sicher zu erkennen. Dadurch wurden wir allerdings langsamer.

# Software

***rkurve() und lkurve()***

Diese Funktionen dienen dazu, den Robter beim Abbiegen um einen festgelegten Wert vorzudrehen. Es werden die Motorrichtung, sowie seine Geschwindigkeit gesetzt.

***rkurve2(int ms) und lkurve2(int ms)***

Für den Schluss haben wir keine Querlinie mehr an der wir uns nach dem Abbiegen ausrichten könnten, deshalb gibt es diese beiden Funktionen, die im Prinzip genau dasselbe machen wie die beiden oben, allerdings ist hier der Wert durch Parameter variabel.

***gerade(int cm) und gerade2()***

Ähnlich wie für die Kurven auch hier zwei Funktionen für das Geradefahren, eine mit Parameter und eine ohne.

***drehenR() und drehenL()***

Wenn der Roboter mit rkurve() und lkurve() schon ein Stück vorgedreht hat, sucht diese Funktion nach der nächsten Linie und lässt den Roboter vorwärts fahren, sobald eine bemerkt wurde.

***vordrehenL()***

Diese Spezialfunktion ist nötig um den Roboter durch den rechten Kurs mit der parallel verlaufenden Linie zu bringen. Wir lassen den Roboter hier ein bisschen mehr als gewöhnlich drehen, um aus dem gefährdeten Bereich herauszukommen.

***start()***

Hier ist der Startvorgang geregelt, also das Bemerken des angeschalteten Startsignals durch den Lichtsensor.

***fahreBisR() und fahreBisL()***

Der Roboter fährt bis er mit seinem rechten bzw. linken Vordersensor über eine Linie gefahren ist, und verlässt dann die Funktion.

***korrigiereL() und korrigiereR()***

Diese Funktion ist nötig, um das gerade Stück zu schaffen das außen liegt und dem entsprechend je nach Seite nur die linke bzw. rechte Linie zur Verfügung stehen um sich “auszurichten”. Daher soll er sich korrigieren sobald er, entweder mit linkem oder rechtem Sensor, eine Linie wahrnimmt.

***fahreBisLR()***

Wenn der Roboter eine Kreuzung überfährt, dann korrigiert er sich in derart, dass er sich in die andere Richtung dreht.

***finishR()***

Das ist der Zieleinlauf nach der letzten Kurve, also erst das Linie finden, Arm ausfahren, zwischen den Linien hin und her pendeln und zu guter letzt stoppen.

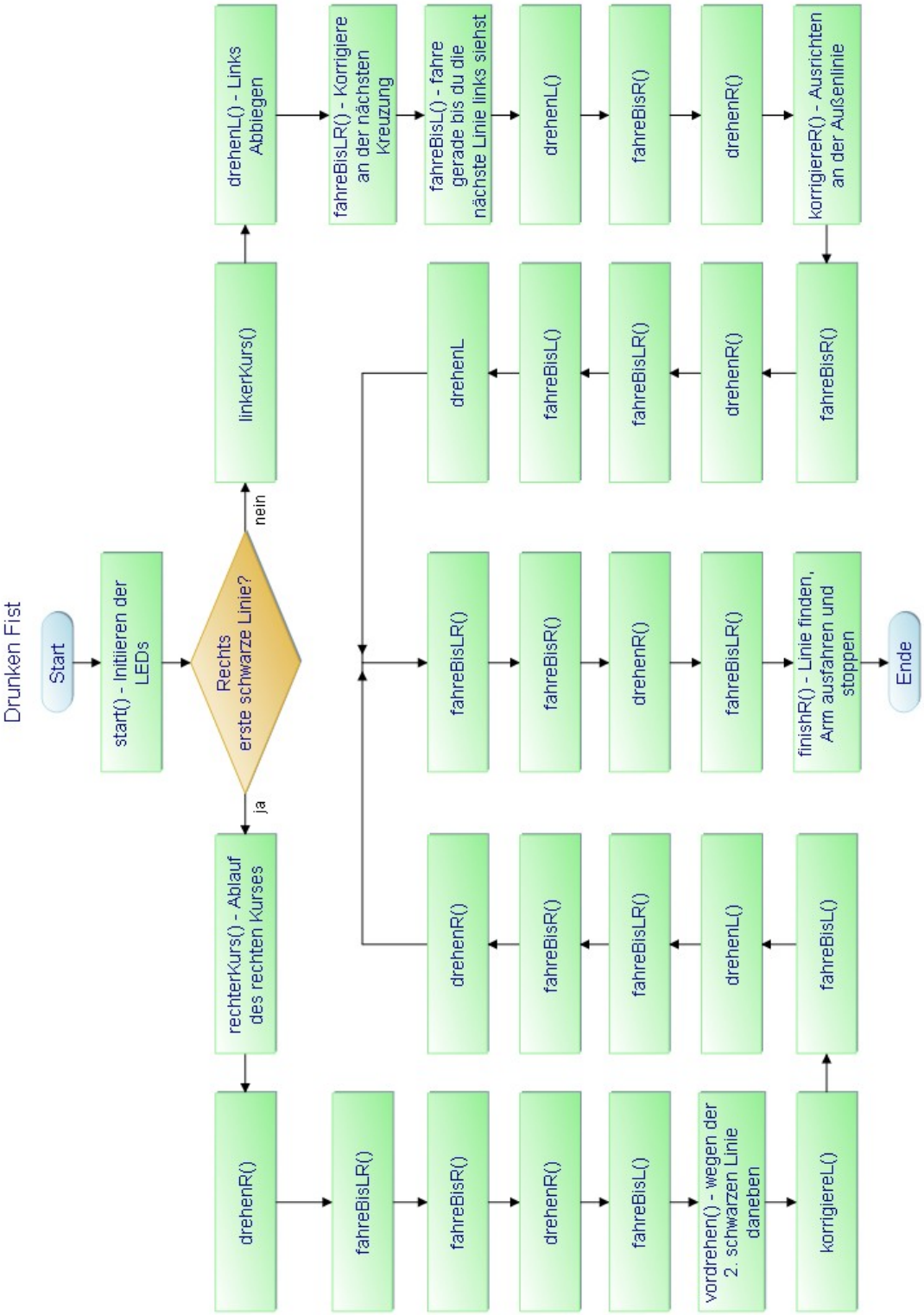
***sensorLR()***

Dient dem Anzeigen der drei Sensorwerte, nur für Testzwecke vorgesehen.

***rechterKurs() und linkerKurs()***

Gesamter Fahrablauf des rechten bzw linken Kurses.

# Programmablaufplan



# Die Hardware

***Sensoren***

- drei Optokoppler
- ein Lichtsensor
- ein AKSEN-Board
- drei Elektromotoren

***Die Optokoppler***

Jene dienten der Orientierung auf der Strecke. Einer war vorn in der Mitte angebracht und diente dazu uns jedes Mal davor zu bewahren, dass der Roboter sich an die nächstbeste Wand stellt und Pause macht. Zum einen diente er in Kurven dem Auffinden der richtigen Linie zum Abbiegen und zum anderen diente er auf gerader Strecke dem “Zurückpendeln” auf die Linie. Die anderen beiden waren vorne links bzw. vorne rechts angebracht und informierten über Kreuzungen.

***Der Lichtsensor***

Für den Lebenspunkt musste dieser Sensor seinen Dienst leisten, er hängt hinter dem Roboter um uns schon am Start einen kleinen Vorteil gegenüber den Gegnern zu verschaffen. Er registriert, wenn das Startsignal leuchtete und brachte so alles in Gang.

***Das AKSEN-Board***

Der Kopf des Roboters, wenn man so möchte, sammelt alle Informationen, die es von den Sensoren erhält, verarbeitet diese und sendet dementsprechend Steuersignale an die Aktoren, in dem Fall waren das nur die Elektromotoren, also vorwärts fahren oder abbiegen.

***Die Elektromotoren***

Zwei der drei Motoren dienen bei uns dem Vortrieb bzw. dem Kurvenbefahren und der dritte wird dazu benutzt den Arm auszufahren, der die Kohlenstoffatome vom Wasserstoff trennen soll.

***Das Getriebe***

Die Übersetzung bei diesem Roboter ist 1:45. Wir haben bewusst auf viel Kraft verzichtet, um unseren Roboter schneller zu machen.

***Der "Arm"***

Eine Zahnstange, die ausgefahren wird, um die Methanmoleküle zu spalten.

# Zusammenfassung

Im Nachhinein betrachtet hat dieses Projekt sehr viel Aufschluss darüber gegeben, wie das bei der Arbeit mit Robotern ist. Ähnliches hört man auch wenn beim Film jemand mit Tieren und/oder kleinen Kindern arbeitet. Sie machen meist nie, was sie sollen, sondern eher was sie wollen, aber trotz aller Umstände bringen sie einen zum Lachen und bereiten Spaß. So erging es uns auch bei diesem Projekt, das uns zum Einen viel Spaß beschert hat, sei es beim "Basteln" mit Lego an sich, als auch beim Sticheln der Mitstreiter, weil sie mal wieder gegen die Wand gefahren sind. Zum Anderen hat uns der Roboter, häufiger als uns lieb ist, zur Ratlosigkeit getrieben, weil er eben das machte, beim nächsten Mal das, und beim nächsten Versuch, als man gedacht hat, man hätte durchschaut, was schief gelaufen ist, überrascht er einen aufs Neue und fährt eine ganz andere Linie oder tut auf einmal das, was man ihm gesagt hat. Dann waren die Fehlversuche davor schnell vergessen und man war der Meinung, man hätte alles richtig gemacht, nur um beim nächsten Mal erneut festzustellen, dass das nicht der Wahrheit entspricht.