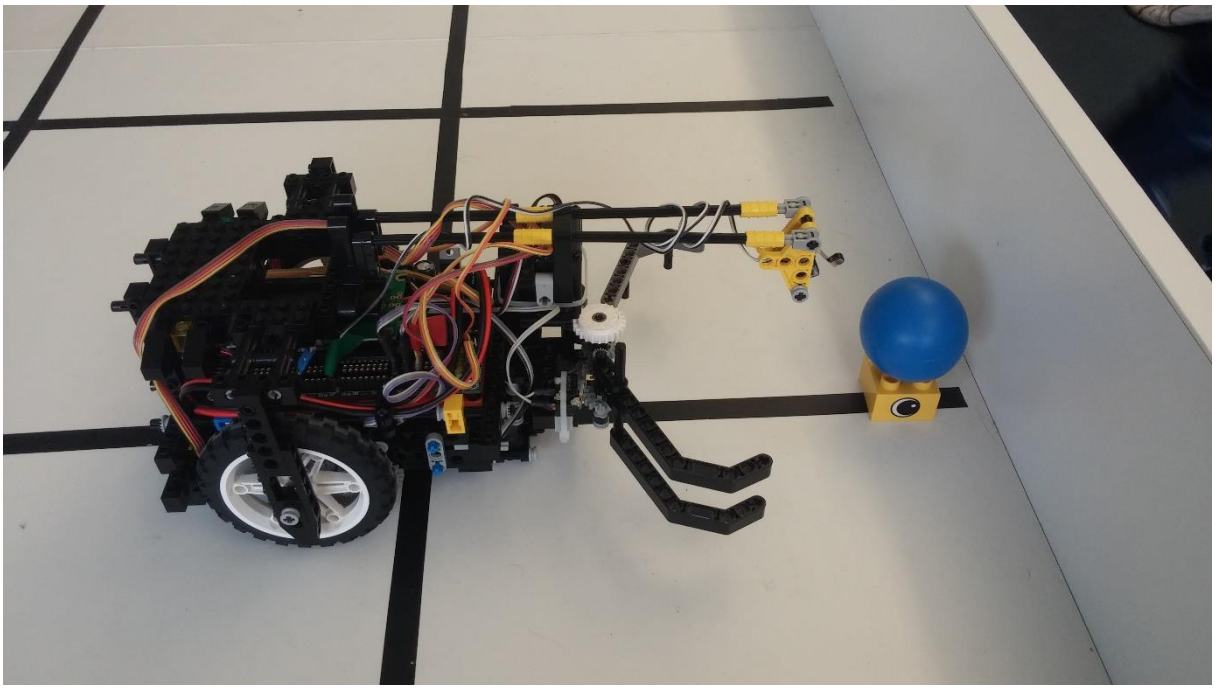
Projekt: Künstliche Intelligenz

vom 22.01.2015



„Jack the Gripper“ am 22.01.2015

Gruppe 3

Daniel Schönbohm

Jan Dikow

Mario Kaulmann

Inhalt

1. Aufgabe

Die Aufgabe des KI-Projekts ist es, einen Lego-Roboter zu entwickeln, der mittels Sensoren, Steuereinheit (AKSEN-Board) und selbstentwickelter Software auf einem Spielfeld eine Route planen und abfahren kann, um Passagiere (blaue Bälle) abzuholen und hinter die Linie AB zu befördern, ohne gesperrte Kreuzungen zu passieren.
Dabei bekommt der Roboter, kurz vor dem Start, das globale Wissen über die aktuelle Karte zur Verfügung gestellt, welche die Positionen der Fahrgäste sowie die Position der offenen und gesperrten Kreuzungen enthält.

2. Ideen und Strategie

Ideen

- Der Roboter sollte Wege abkürzen können.
- Vielleicht kann man durch eine Greifvorrichtung, die den Fahrgast (blauer Ball) nach hinten durchleitet einen Wendevorgang nach Aufnahme des Fahrgastes vermeiden.
- Die Endlosschleife der AksenMain-Funktion sollte nur eine Funktion mache() enthalten, die anhand einer Datenstruktur, die beim Planungsprozess gefüllt wird, die richtigen Steuerungsanweisungen für den Roboter auswählt.
- Alle Sensoren und Antriebe werden als Konstanten im Quellcode kodiert.
- Die interne Darstellung der Kreuzungen auf der Karte wird als zweistellige Zahl so codiert, dass eine Bewegung in Nord- / Südrichtung eine Veränderung um 1 und eine Bewegung in Ost- / Westrichtung eine Veränderung um 10 bewirkt.

Vorgehensweise

- Entwicklung einer stabilen, rutschfesten Halterung für die Steuereinheit
- Anbau eines motorisierten Getriebes und Räder
- Entwicklung erster Softwareansätze zur Erprobung der Fahreigenschaften
- Planung der Sensorenverteilung am Boden des Roboters zur Erkennung der zur Orientierung nutzbaren schwarzen Linien und deren Platzierung
- Entwicklung der Softwarefunktionen, die zum Fahren auf einer Linie gebraucht werden und dessen Erprobung
- Planung und Umsetzung eines Planungsalgorithmus
- Optimierung der Fahrfunktionen
- Anbau eines Greifers und dessen Programmierung
- Test- und Optimierungsphase

Arbeitsfortschritte

1. Tag: Kennenlernen des AKSEN-Boards und der Sensoren / Eingewöhnung in die Programmierung / Fertigstellung der AKSEN-Board-Halterung

2. Tag: Entwicklung des Antriebs / Programmierung einer Fahrfunktion zum ausprobieren

3. Tag: Verbesserung des Antriebs / Drehung des AKSEN-Boards, da es nach Einstecken des Batteriesteckers die Räder blockierte / Entwicklung der aufLinieHalten() Funktion und Erprobung

4. Tag: Beginn der Dokumentation / Verbesserung der Radaufhängung / erste Planung der logischen Funktionen zur Routenfestlegung

5. Tag: Konstruktion eines Greifers / Besprechung des Planungsalgorithmus / Fehlersuche und Fehlerbehebung

6. Tag: Vorstellung des Planungsalgorithmus / Anbau des Greifers / erste Erprobung des Greifers / Fahrtests

7. Tag: Tests und Fehlerbehebung / Versuche mit Schrägfahren

8. Tag: Tests mit dem Planungsalgorithmus auf dem Roboter / Versuche mit Schrägfahren / Wechsel der Batterien / Anbau der Lichtsignale

In den folgenden Tagen erfolgten viele Tests und Anpassung, die vor allem durch Probleme mit der Energieversorgung bedingt waren.

3. Roboter

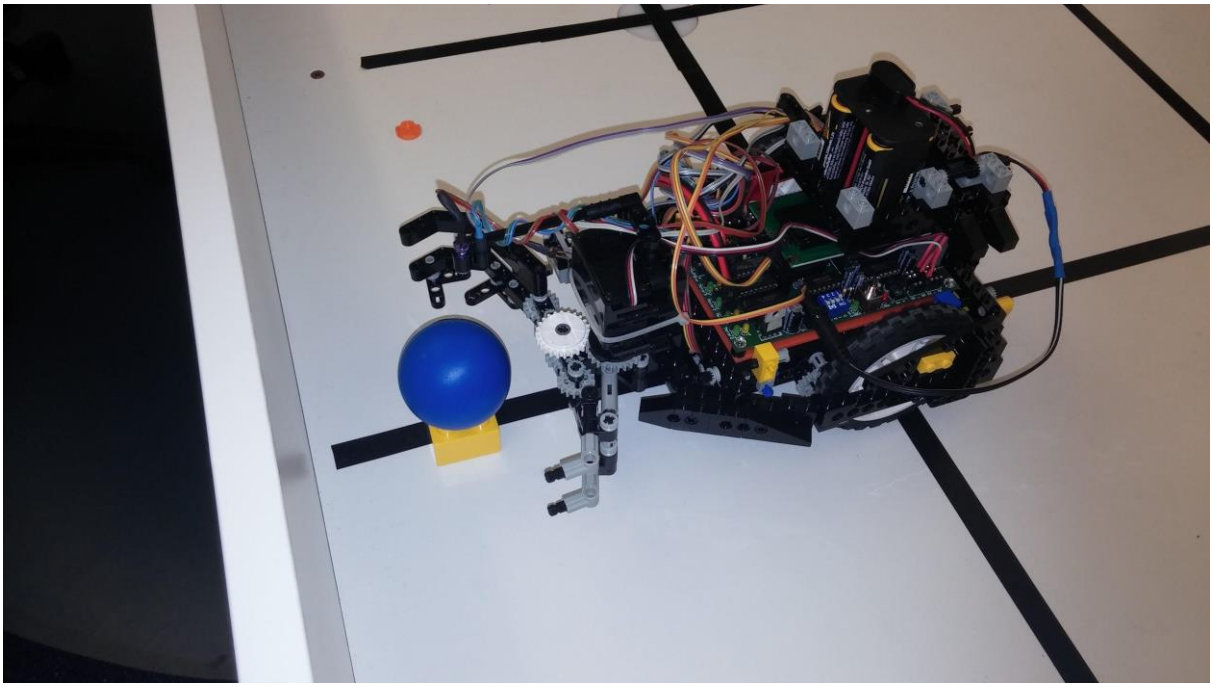


Abb. 1: Dieses Bild zeigt den Roboter bevor der Greifer optimiert wurde. Hier hat der Greifer noch den Infrarotsensor und den Infrarotsender über dem Greifer, die nach unten messen. Dadurch wurde der Fahrgast häufig nicht wahrgenommen, da der Sensor oft am Fahrgast vorbeigemessen hat, wodurch der Roboter oft gegen die Wand gefahren ist. Außerdem ist der Roboter in dieser Version noch länger, wodurch er bei einer Drehung den Fahrgast manchmal mit dem offenen Greifer in der Drehung vom Sockel gestoßen hat.

4. Hardware

Eine möglichst große Wendigkeit wurde durch das Umsetzen eines differentiellen Antriebs erzielt. Um die Gewichtsverteilung besser zu regulieren, wurden zusätzliche Stabilisatoren verbaut, welche die Achsen außerhalb der Räder unterstützen. Die Räder werden jeweils über ein Getriebe mit einer 1-zu-54-Übersetzung angetrieben. Um das Gleichgewicht zu wahren, dient ein dritter Auflagepunkt als Stütze in Form einer flachen, abgerundeten Scheibe.

Der Akku befindet sich genau über der Achse und bildet so den schwersten Punkt des Roboters, um die Reibung des Auflagepunktes zu vermindern. Zu erwähnen ist, dass das Alter einiger Akkuzellen zum erneuten Kollabieren der Stromversorgung und so zum Abstürzen des Roboters führte. An dem Akkublock ließ sich ein Kurzschlussstrom von 2,3 Ampere messen. Das Austauschen einiger Zellen führte zu einer Verbesserung und ergab bei erneuten Messen eine Steigerung auf 6,7 Ampere.

Ausgestattet ist der Roboter mit insgesamt vier Sensoren. Drei diese Sensoren sind Optoreflexkoppler und der vierte ist ein Photodetektor. Der Photodetektor ist dafür zuständig, das Startsignal zu erkennen und befindet sich an der Unterseite des Roboters zwischen Achse und Auflagepunkt. Ebenfalls an der Unterseite befinden sich zwei der Optoreflexkoppler im Abstand von ca. 3 cm zueinander zwischen Greifer und Auflagepunkt. Die Sensoren dienen der Methode "aufLinieHalten". Der dritte Sensor befindet sich hinten am Roboter auf der rechten Seite, kurz hinter den Rädern. "Hinten" ist daran zu erkennen, dass der Greifer an der Vorderseite montiert ist. Der einzelne Sensor dient der Erkennung der überfahrenen Kreuzung vor dem Fahrgast und ist somit Teil der Prozedur, welche für das Abbiegen und die Positionierung auf den Fahrgast zu zuständig ist.

Der Greifer ist als separates Werkzeug zu dem Roboter zu betrachten. Die Komponente ist über nur vier Verbindungselemente aus Legotechnik sowie ihre Strom- und Steuerleitungen verbunden. Das Werkzeug kann abmontiert werden, um es durch andere zu ersetzen. Der Greifer arbeitet mit einem Servomotor, dessen standardisierte Ausrichtung 0 Grad beträgt - Greifer offen. Wird diesem das Signal zum Zugreifen in Form eines Winkels gegeben, bewegen sich die Greifer aufeinander zu und umschließen den Fahrgast. Das Überdrehen des Servomotors durch eine sich variierende Größe des zu fassenden Objektes wird durch eine Rutschkupplung verhindert. Die Rutschkupplung manifestiert sich als weißes Zahnrad, welches in dem Greifergetriebe verbaut ist.

Das Signal zum Auslösen des Greifers gibt ein Schalter, welcher über dem Greifer montiert ist. Dieser stößt gegen die Bande der Umgebung und garantiert so ein Stoppen des Roboters, selbst wenn dieser keinen Fahrgast erkennt. In diesem Fall würde der Roboter also zugreifen und den jedoch nicht vorhandenen Fahrgast zum Ziel transportieren.

5. Software

Die Software ist in die zwei Bereiche Planung und Ausführung unterteilt, wobei das globale Array aktion[] die Schnittstelle zwischen den beiden Bereichen ist.

Planung

Die Planung beginnt damit, dass der String, der Informationen über die Karte bereitstellt, in ein zweidimensionales Array eingetragen wird. Dadurch entspricht die nun vorliegende Karte der folgenden Vorstellung.

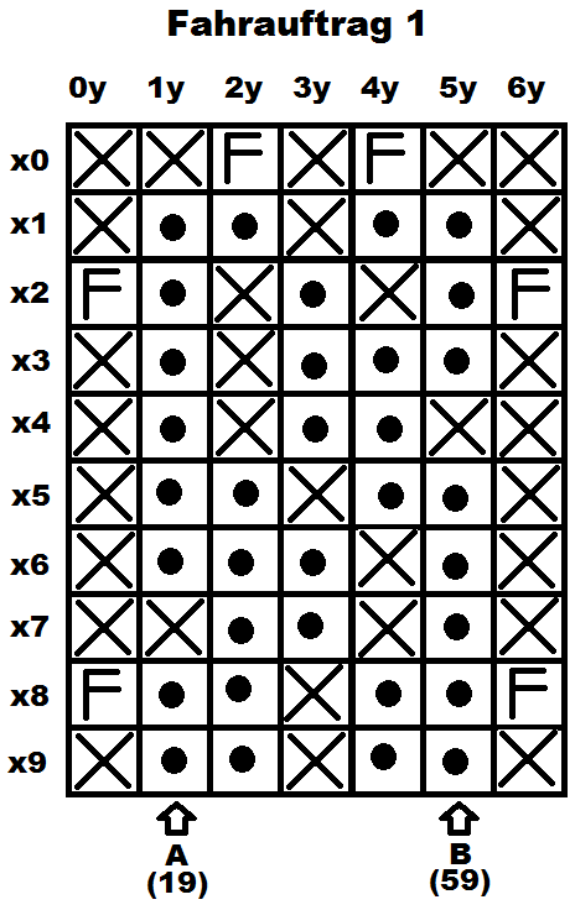


Abb. 2: Programminterne Darstellung einer Karte

Anschließend werden für die Breitensuche eine Schlange und ein Weg-Array initialisiert, um sicherzustellen, dass kein Datenmüll die korrekte Ausführung des Algorithmus behindert. Das Weg-Array hält Informationen über alle befahrbaren Kanten bis zum Fahrgast. Diese Informationen sind: Vorgänger, Nachfolger, Gewicht und Richtung der Kante.
Die Funktion planung() trägt zunächst den Startpunkt und ein ‘e’ als Endezeichen in die Schlange ein. Anschließend wird der Startpunkt in das Weg-Array eingetragen, mit einer -1 als Vorgänger, damit dieser Punkt als Startpunkt identifizierbar bleibt. Die Richtung des Startpunkts ist NORD, da die Startbedingung immer eine Nordausrichtung vorsieht und das Gewicht ist 0.

Kantensuche

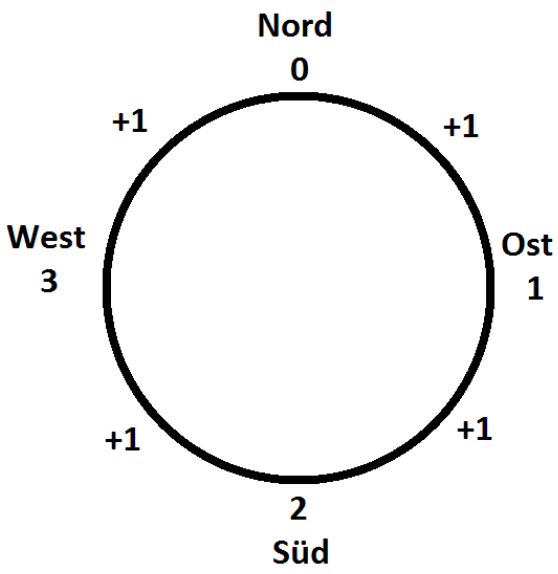
In einer Schleife, die erst abbricht, wenn das aktuelle Element das Endezeichen (‘e’) ist, wird das aktuelle Element aus der Schlange gelesen.

Zu dem aktuellen Element wird jetzt der Nachbar im Norden (danach in gleicher Weise auch für die anderen Richtungen Osten, Süden und Westen, in der Reihenfolge) bestimmt. Wenn diese Position keine gesperrte Kreuzung ist (‘x’), innerhalb der Karte liegt und der Nachbar noch nicht als Vorgänger im Weg-Array vorhanden ist, um Eintragungen von Rückwegen zu vermeiden, wird eine neue Kante in das Wegarray aufgenommen mit dem Nachbar als Nachfolger, dem aktuellen Element der Schlange als Vorgänger und der Richtung.
Für den Fall, dass sich ein Fahrgast (‘F’) auf der Nachbarposition befindet und ein Hinweg zu einem Fahrgast gesucht wird, wird die Route zu dem Fahrgast hinten in das aktion-Array eingetragen, aus dem danach die Fahranweisungen erstellt werden, die vorne in das aktion-Array eingetragen werden. Diese Schritte werden später noch erläutert.
Nachdem die Fahranweisungen zu dem Fahrgast erstellt worden sind, wird an der Nachbarposition, die Kennzeichnung für einen Fahrgast (‘F’) durch die Kennzeichnung für eine gesperrte Kreuzung (‘x’) auf der Karte ersetzt.
Die Schlange und das Weg-Array werden neu initialisiert, wobei nun die neue Startposition das letzte aktuelle Element der vorherigen Schlange ist, welches die Position vor dem Fahrgast ist. Außerdem wurde nun das Ziel geändert indem das ziel_flag auf 0 gesetzt wurde, was zur Folge hat, dass jetzt ein Rückweg zur AB-Linie gesucht wird. Die Schleife wird nun für das derzeitige Element, das noch aus der alten Schlange stammt, abgebrochen und von vorne gestartet.
Eine Besonderheit liegt bei der Richtung Süden vor, denn diese Richtung ist die einzige, die die Suche nach einem Rückweg beenden kann, da die AB-Linie die südlichste Linie im gesamten Streckennetz ist, muss eine Bewegung in Richtung Süden für die kürzeste Strecke der Abschluss sein.
Sollte ein Nachbar keine Zielposition kennzeichnen, bei Hinweg ein Fahrgast (‘F’) oder bei einem Rückweg eine Position auf der AB-Linie, wird der Nachbar unter der Bedingung, dass er sich nicht in der Schlange befindet in die Schlange aufgenommen.
Die Schleife wird nach der Planung des Rückwegs vom letzten Fahrgast dadurch beendet, dass bei der Suche nach weiteren Fahrgästen keine mehr zu finden sind, da ihre Markierungen ersetzt wurden, und die Schlange bis zum Endezeichen (‘e’) durchlaufen wird.

Route eintragen und Anweisungen planen

Das Eintragen der Route, nachdem in der Karte das erstmögliche erreichbare Ziel gefunden wurde, beinhaltet zunächst die Bestimmung des Gewichts jeder einzelnen Kante, um eine Strecke mit möglichst wenigen Drehungen und zu befahrenden Kreuzungen zu planen. Das Gewicht der Startkante, mit Vorgänger -1 und dem Startpunkt als Nachfolger ist 0, alle anderen Kanten haben nach der Initialisierung die Gewichtung 100. Jetzt wird für die erste

Kante geschaut, ob es im Weg-Array Kanten gibt, die als Vorgänger den Nachfolger der ersten Kante haben. Sollte es solche Kanten geben, wird das Gewicht dieser Kanten auf den Wert der ersten Kante gesetzt und mit eins addiert, wenn die Richtungen der beiden Kanten übereinstimmen oder mit zwei addiert, wenn die Richtungen der Kanten nicht übereinstimmen, was bedeutet, dass man ohne Drehung nicht von der einen auf die andere Kante fahren kann. So wird mit allen weiteren Kanten nach der ersten Kante verfahren.
Jetzt wird die Richtung der letzten Kante hinten in das aktion-Array eingetragen, die Position für den nächsten Eintrag wird um eins gesenkt und ein Merkgewicht von 99 festgelegt.
In einer Schleife wird das Weg-Array von hinten durchgegangen um den Weg zum Ziel zu bekommen. Dabei wird immer der Vorgänger des aktuellen Elements mit dem Nachfolger aller anderen verglichen. Bei Gleichheit wird getestet, ob das Gewicht des Elements kleiner ist, als das Merkgewicht. Ist dies der Fall, wird das Merkgewicht auf den Wert des Gewichts des Elements gesetzt und die Richtung des Elements im aktion-Array gemerkt. Dann wird dieses Element das aktuelle Element, bis man beim Startpunkt angekommen ist.
Im Anschluss daran können aus den Fahrtrichtungen, die hinten im aktion-Array eingetragen sind die Fahranweisungen erzeugt werden.
Die Position der letzten eingetragenen Richtung wird hierbei als Startwert für die Zählschleife genutzt die endet, nachdem das letzte Element bearbeitet wurde.
In der Schleife wird die Differenz aus der Richtung des Roboters und der aktuellen Richtung im aktion-Array gebildet. Diese Differenz bestimmt die Fahranweisung/en, die vorne im aktion-Array eingetragen werden, nach folgendem Muster.



Die Differenz aus aktueller Richtung und gewünschter Richtung gibt die Bewegungsart an.

Differenz ist 0	=> keine Drehung
Differenz ist -1 oder 3	=> Rechtsdrehung
Differenz ist 2 oder - 2	=> Wenden
Differenz ist -3 oder 1	=> Linksdrehung

Bsp.:
aktuelle Richtung ist West gewünscht wird Nord.
3 - 0 = 3 => Rechtsdrehung

Abb. 3: Auswirkung von Richtungsdifferenzen bei der Planung

Auf die Drehanweisungen ‘l’ (links), ‘r’ (rechts), ‘w’ (wenden) muss immer ein ‘v’ (vorwärts) folgen um eine Bewegung entlang einer Kante zu realisieren, sonst würde sich später lediglich die Richtung des Roboters ändern.
Ist das Ziel erreicht worden, wird statt einem ‘v’ ein ‘g’ (greifen), wenn es ein Hinweg ist, oder ein ‘a’ (ablegen), wenn es ein Rückweg ist eingetragen.
Die Anweisungskette im vorderen Teil des aktion-Array wird mit einem ‘e’ als Endezeichen abgeschlossen.

Ausführung

Nach dem Starten des AKSEN-Boards wird die gelbe Lampe eingeschaltet, die Karte eingelesen und die Planung ab einer durch einen DIP-Schalter festgelegten Startposition durchgeführt. Danach wird der Greifer in seine korrekte Position gebracht und die Ampel auf Grün gestellt.
Sobald das Startsignal durch den Photosensor wahrgenommen wird, beginnt eine Endlosschleife, die alle im aktion-Array vorhandenen Aktionen der Reihe nach abarbeitet.

Aktionen

Folgende Aktionen sind definiert:

Vorwärts (‘v’)

Der Roboter fährt mit voller Geschwindigkeit vorwärts und hält sich dabei auf der Linie.

Links drehen (‘l’) und rechts drehen (‘r’)

Nach dem Greifen dreht sich der Roboter zur Seite bis seine Sensoren die Linie erreichen. Wurde zuvor nicht gegriffen, kann man davon ausgehen, dass sich der Roboter auf einer Kreuzung befindet. In diesem Fall führt er eine kurze Vorwärtsbewegung aus, um in eine bessere Position zu gelangen und, falls er vorher einen Ball abgelegt hat, wartet er kurz, damit der Ball wegrollen kann. Erst dann dreht er sich zur Seite bis er die Linie erreicht. Sollte die nächste Aktion das Greifen sein, ist diese Drehung modifiziert: Er fährt vorwärts, bis sein Heck die Kreuzung erreicht und führt die Drehung rückwärts aus, während ein Rad steht, um zu vermeiden, dass der Greifer versehentlich den Ball wegstößt.

Greifen (‘g’)

Der Roboter fährt vorwärts und hält sich auf der Linie bis der Taster an seiner Vorderseite an die Wand stößt. Dann greift er den Ball und fährt kurz rückwärts, um sich in eine bessere Position für die Folgeaktion zu bringen.

Ablegen (‘a’)

Der Greifer öffnet sich und gibt den Ball frei.

Wenden (‘w’)

Hat der Roboter zuvor einen Ball abgelegt, fährt er kurz vor, um ihn wegzustoßen. Dann wendet er sich, indem er sich zweimal so lange nach rechts dreht, bis er eine Linie erreicht.

Ende (‘e’)

Der Roboter stoppt und das AKSEN-Board begibt sich in eine leere Endlosschleife.

Für die Ausführung dieser Aktionen werden einige Methoden benutzt, von denen im Folgenden die beiden komplexeren Methoden erläutert werden.

aufLinieDrehen

Dieser Methode wird die Drehrichtung des Fahrzeugs übergeben. Außerdem kann festgelegt werden, dass ein bestimmtes Rad stehen bleiben soll. Sollen sich beide Räder drehen, wird hier 0 übergeben. Anhand dieser Daten werden der zu verwendende Optokoppler und die Drehgeschwindigkeiten der einzelnen Räder festgelegt.
Für den Fall, dass sich der Sensor zu Beginn noch über einer schwarzen Linie befindet, dreht sich der Roboter in die entsprechende Richtung, bis der Sensor weiß sieht. Anschließend dreht er sich so lange, bis der Sensor wieder eine Linie erkennt. Nun dreht er sich ein weiteres Mal, mit geringerer Geschwindigkeit, bis der Sensor die schwarze Linie überschritten hat und hält an. Jetzt befindet sich die Linie, auf die sich der Roboter drehen sollte, direkt zwischen den beiden Optokopplern.
Die niedrigere Geschwindigkeit bei der letzten Aktion ist nötig, um den Bremsweg zu verkürzen. Andernfalls würde sich der Roboter, wenn der Akku frisch geladen ist, manchmal so weit drehen, dass die zu verfolgende Linie nicht mehr zwischen den Sensoren ist.

aufLinieHalten

Der Roboter fährt zuerst kurz vorwärts, um sicherzustellen, dass die Sensoren nicht mehr direkt über der Kreuzung sind. Dann startet eine Schleife, die erst abbricht, wenn sich der Roboter wieder auf einer Kreuzung befindet, die Optokoppler also beide Schwarz sehen. Innerhalb dieser Schleife korrigiert der Roboter seine Fahrbewegung nach links, wenn der links Sensor die zu verfolgende Linie sieht und nach rechts, wenn der rechte Sensor die Linie sieht. Solange sich die Linie dazwischen befindet, fährt der Roboter mit voller Geschwindigkeit vorwärts. Nach Abbruch der Schleife wird die Fahrbewegung gestoppt, sofern keine weitere Vorwärtsbewegung folgt.
Der Methode kann außerdem eine bestimmte Stopp-Aktion übergeben werden: Wird “GREIFEN” übergeben, stoppt der Roboter und greift, sobald der Taster an seiner Vorderseite beim Vorwärtsfahren gegen eine Wand stößt. Anschließend wird die Schleife abgebrochen. Wird “HECK_AUF_KREUZUNG” übergeben, fährt der Roboter vor, bis der Optokoppler, der hinten seitlich angebracht ist, die Kreuzung erkennt, stoppt dann und bricht die Schleife ebenso ab. Diese Aktion wird für das Rückwärtswenden beim Abbiegen vor dem Greifen benötigt.

Verworfene Ideen

Eine Idee zum effektiveren Fahren beinhaltet das Fahren von Abkürzungen.
Nach der Planung wurde das aktion-Array nach den Mustern ‘v’, ‘l’, ‘v’ und ‘v’, ‘r’, ‘v’ durchsucht, die dann durch ‘L’, ‘ ‘, ‘ ‘ und ‘R’, ‘ ‘, ’ ‘ ersetzt wurden. Die Muster wurden jedoch nicht ersetzt, wenn vorher die Anweisungen ‘w’ (wenden) oder ‘g’ (greifen) kam, weil sich die Optoreflexkoppler nach diesen Anweisungen nicht unbedingt über einer Kreuzung befinden, was die sichere Ausführung der Abkürzungsanweisungen negativ beeinflussen kann. Die nachfolgende Skizze verdeutlicht, wie die Abkürzungen ausgeführt werden sollten.

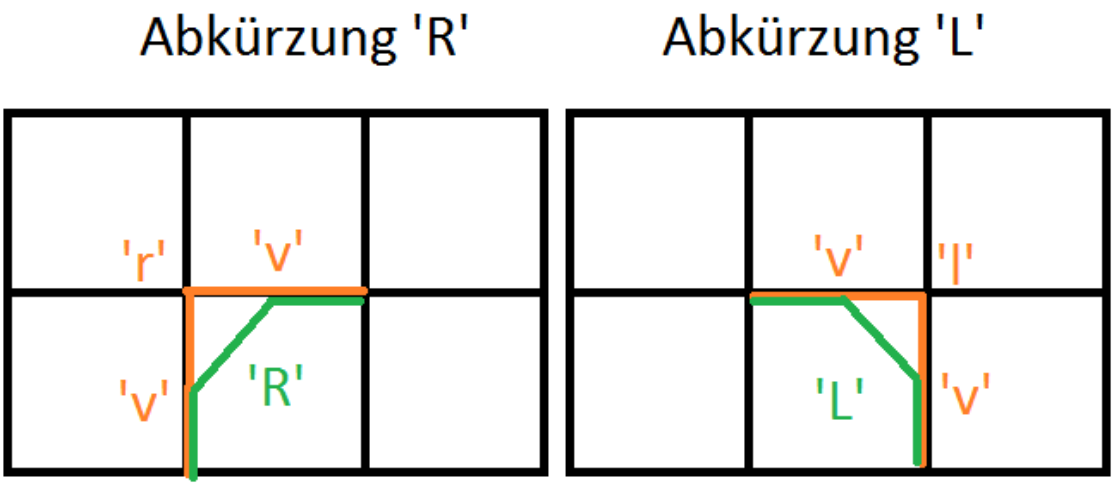


Abb. 4: Prinzipielle Darstellung (nicht implementierter) Abkürzungsmanöver (grün) im Vergleich zu den planmäßigen Fahrmanövern (orange)

Bei diesem Fahrmanöver sollte der Roboter für eine bestimmte Zeit geradeaus fahren und sich auf der Linie halten, dann eine bestimmte Zeit lang drehen und auf die zur vorherigen Linie senkrechte Linie zufahren, bis der näherliegende Sensor diese Linie überschreitet. Ab diesem Moment sollte er sich wieder auf der Linie halten.
Dieses Fahrmanöver funktionierte allerdings nur bei einem konkreten Akkustand. Stark unterschiedliche Akkustände führten zu abweichenden Fahr- und Drehzeiten, sodass auf dieses Fahrmanöver verzichtet wurde, um eine bessere Fahrsicherheit zu gewährleisten. Im Zuge dieser Erkenntnis wurden auch andere Fahrbewegungen dahingehend optimiert, dass keine langen zeitabhängigen Bewegungen stattfinden.

6. Projektverbesserungsvorschläge

Bessere Stromversorgung

Die meisten Probleme sind im Zusammenhang mit der Stromversorgung aufgetreten. Hätten bessere Akkus zur Verfügung gestanden, wäre mehr Zeit vorhanden gewesen, um andere Funktionen des Roboters zu verbessern.