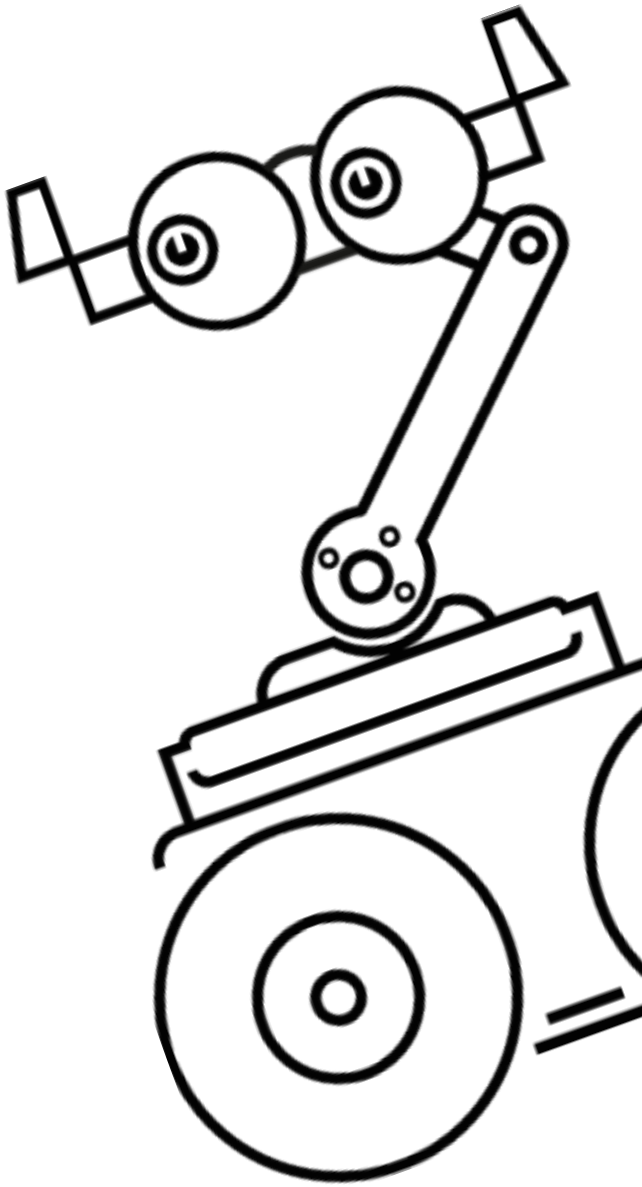
Orientierung

Roberta-Experiment mit dem LEGO Mindstorms NXT



ROBERTA
INITIATIVE

Roberta ist ein eingetragenes Warenzeichen der Fraunhofer-Gesellschaft e.V.
Roberta ist seit 2010 Mitglied der Fraunhofer Academy

Inhalt

1. Einführung

Grundlagen

Im Folgenden wird das Experiment »Braitenberg-Vehikel« vorgestellt, beschrieben und anhand von Programmbeispielen veranschaulicht.

Für das Experiment wird das handelsübliche Roboter-Baukastensystem LEGO Mindstorms NXT verwendet. Die Programmierung erfolgt mit der kostenfreien Programmiersprache NXC (Not eXactly C) und der ebenfalls kostenfreien Programmierumgebung BricxCC.

Als Robotermodell wird der sogenannte Jeffrey-Bot (siehe Abbildung 1) verwendet, welcher als CAD-Modell auf den internen Seiten des Roberta-Portals (www.roberta-home.de) allen zertifizierten und registrieren Roberta-Teachern zur Verfügung steht. An dieses Modell werden im Verlauf des Experiments verschiedene Sensorarten angebracht.

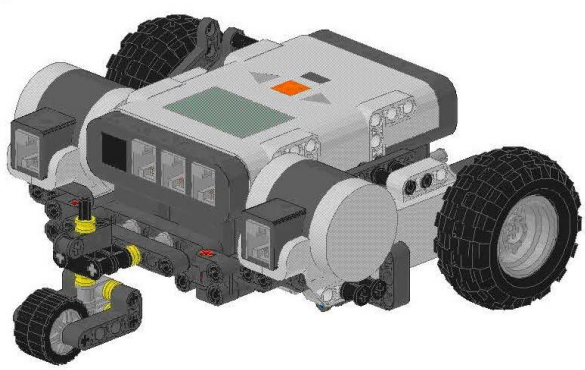


Abbildung 1: NXT-TriBot ohne Sensoren

Die Lösungsvorschläge und Programme sind möglichst einfach gehalten. Man kann jeweils sehr viel komplexere Programme entwerfen.

2. Projekt »Orientierung«

Allgemeine Informationen

Lernbereich

Das Projekt »Orientierung« beinhaltet:

- wie sich Menschen, Tiere und Roboter in ihrer Umwelt orientieren,
- welche Navigationsarten in der Robotik zur Orientierung genutzt werden,
- eine Einführung in die Odometrie,
- Navigation eines omnidirektional angetriebenen Roboters,
- Funktionsweise eines omnidirektionalen Antriebs,
- verschiedene Sensortypen und deren Einsatzmöglichkeiten,
- Konstruktion, Steuerung und Programmierung eines omnidirektional angetrieben Roboters.

Hilfreiche Fragestellungen zu diesem Thema

- Woher wissen wir, wo wir sind?
- Wie finden wir uns in unserer alltäglichen Umwelt zurecht?
- In wie weit nutzen wir dafür unsere Augen, Ohren, Hände und Füße?
- Welche unserer Sinne (Hören, Sehen, Riechen, Tasten, Schmecken) nutzen wir, um uns zu orientieren?
- Hilft uns unser Gedächtnis?
- Haben wir eine Karte im Kopf?

Tipps

Hinführung auf das Thema durch

- Nutzung eines Kompasses zusammen mit einer Wanderkarte;
- Sechstanten und/oder Sternenkarten;
- Verbinden der Augen und Orientierung durch Zurufe (Hörsinn);
- Verbinden der Augen und nutzen eines Stocks zur Orientierung (Tastsinn).

Einarbeitung in das Thema

Einführung

Wahrscheinlich kennt das jeder, man betritt ein, unbekanntes Gebäude (Schule, Verwaltung, Kaufhaus etc.), läuft darin auf der Suche nach einem Raum, Zimmer, Geschäft umher und stellt nach einer gewissen Zeit fest, man hat sich verlaufen. Man blickt sich orientierungslos um auf der Suche nach markanten Punkten, die einem weiterhelfen könnten, doch man bleibt unfündig. Auch ein angestrengter Versuch, sich zu erinnern, eine im Gehirn abgelegt Karte des Ortes zu finden, bleibt erfolglos.

Man hat die Orientierung verloren! Einhergehend mit dem Verlust der Orientierung fehlt ein wesentlicher Bestandteil der Navigation, die Zielbildung (siehe Abbildung 1). Auf die Instrumente der Navigation wird im Punkt »Omnidirektionaler Antrieb« näher eingegangen.

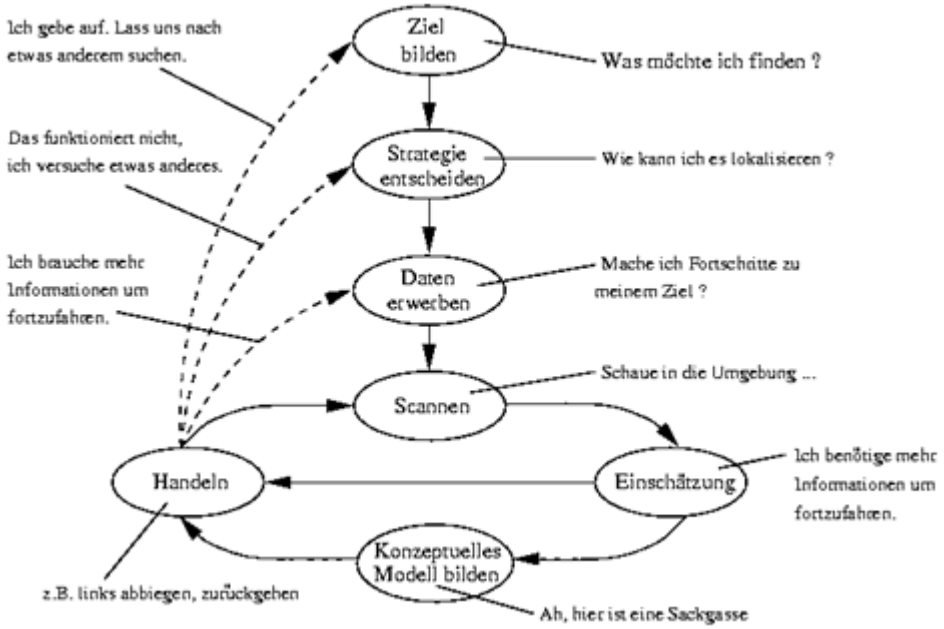


Abbildung 1: Navigationsprozess nach Jul and Furnas, 1997

Die räumliche Orientierung wird vor allem durch Bewegung im Raum erlernt. Während die kleinräumige Orientierung in den ersten Lebensjahren eingeübt wird (das Körperschema ist das erste räumliche Vorstellungsbild), folgt in den späteren Jahren das Erlernen der geographischen Orientierung.

Dabei entwickelt sich zuerst unten–oben, dann vorne–hinten und zuletzt links–rechts. Das Gehirn lernt wichtige Markierungen und Informationen, die für das Zurechtfinden in einer Umgebung nötig sind, herauszufiltern und in so genannten mentalen Karten abzuspeichern.

Unter Orientierung im Raum werden grundsätzlich zwei verschiedene Dinge verstanden.

- Wissen um die eigene Position
  Das Wissen über die Position in einem realen oder virtuellen Raum (mit Hilfe von Sinnen bzw. Sensoren), um mit diesem Wissen Wege zu finden oder seinen Standort beschreiben zu können. Beispiel: Beim Orientierungslauf findet man seinen Weg mit Karte und Kompass.
- Wissen um die Lage eines Objektes
  Die Beschreibung der Lage eines Objektes in einem Raum.

Das Wort »orientieren« bedeutet wörtlich »nach Osten ausrichten«, und beruht auf die Ausrichtung mittelalterlicher Kirchen, deren Altar nach Osten ausgerichtet wurde.

Nachfolgend werden die für die Orientierung von Mensch, Tier und Robotern wichtigen Sinne bzw. Sensoren erläutert. Zwar werden die Sinne und Sensoren jeweils einzeln erläutert, allerdings ist meistens ein Zusammenspiel mehrerer Sinne für eine sinnvolle Orientierung notwendig. Wie mehrere Sinne zusammenwirken wird erst auffällig, wenn dieses Zusammenwirken fehlerhaft ist. Ein Beispiel ist das Lesen während der Autofahrt. Hier registriert das Gleichgewichtsorgan (Vestibular) Bewegungen, während das Auge einen Stillstand wahrnimmt. Diese Diskrepanz verursacht bei vielen Menschen Übelkeit während der Autofahrt.

Sinne

Sehsinn: Mensch/Tier

Die visuelle Wahrnehmung (von lat.: visus – Sicht, Sehen) ist die Wahrnehmung von Objekten. Lichtstrahlen, die von den Objekten ausgesandt, gebeugt oder reflektiert werden, verursachen eine Reizung der Netzhaut (Retina).

Die visuelle Wahrnehmung ist der wohl wichtigste Sinn für Menschen, um sich in seiner Umwelt zu orientieren. Allerdings gilt: »Wir sehen nicht mit den Augen, sondern mit dem Gehirn, aber ohne Augen sehen wir auch nicht«[1]. Mit Hilfe des Sehsinns werden Objekte abgeschätzt nach Größe, Farbe, Form, Entfernung und Gewicht. Aber auch relative Bewegungen eines Objektes zu einem selbst werden über den Sehsinn ermittelt. Wie eingangs erwähnt erstellt das Gehirn im Laufe der Zeit (beim wiederholten Spazierengehen im (selben) Wald etc.) eine immer detailreichere mentale Karte.

In der Tierwelt finden sich natürlich ebenso zahlreiche Tierarten, die hauptsächlich ihre Augen zur Orientierung nutzen. Allerdings gibt es hier verschiedene Arten der visuellen Wahrnehmung:

- Wärmestrahlenauge für infrarotes Licht (Klapperschlangen)
- Punktaugen zur Messung der Tageshelligkeit (Nesseltiere)
- Facettenaugen für ultraviolettes Licht (Bienen → siehe Band 1 der Roberta Reihe).

[1] http://www.optomotorik.de/blicken/index.htm [11/2009]

Hörsinn: Mensch/Tier

Die auditive Wahrnehmung (von lat. ich höre) ist die Wahrnehmung von Schall. Die Ohren nehmen Töne wahr, unterscheiden Frequenzen, bestimmen Richtungen und Entfernungen von Geräuschquellen. Für die Bestimmung, aus welcher Richtung ein Geräusch kommt (Lokalisation), werden zwei Ohren benötigt. Aufgrund der Zeitdifferenz, mit der ein Ton zuerst auf das linke (rechte) und dann auf das rechte (linke) Ohr trifft, lässt sich bestimmen, aus welcher Richtung der Ton stammt. Dabei kann das Gehirn eine Differenz von bis zu einer hunderttausendstel Sekunde unterscheiden. Vor allem sehbehinderte Menschen nutzen ihren meist etwas stärker ausgeprägten Hörsinn, um sich in ihrer Umwelt besser zurechtzufinden. Bei Tieren wie Hunden oder Katzen reicht die Wahrnehmung von Frequenzen bis in den Ultraschallbereich (>20 kHz).

Für die Orientierung mittels Ultraschall sind vor allem Wale, Delphine (bis 200 kHz) und Fledermäuse (bis 200 kHz) bekannt. Dabei senden diese Ultraschallsignale aus, die von Objekten reflektiert und anschließend wieder von den Ohren (Fledermaus) bzw. vom Unterkiefer[2] (Delphin) aufgefangen werden.

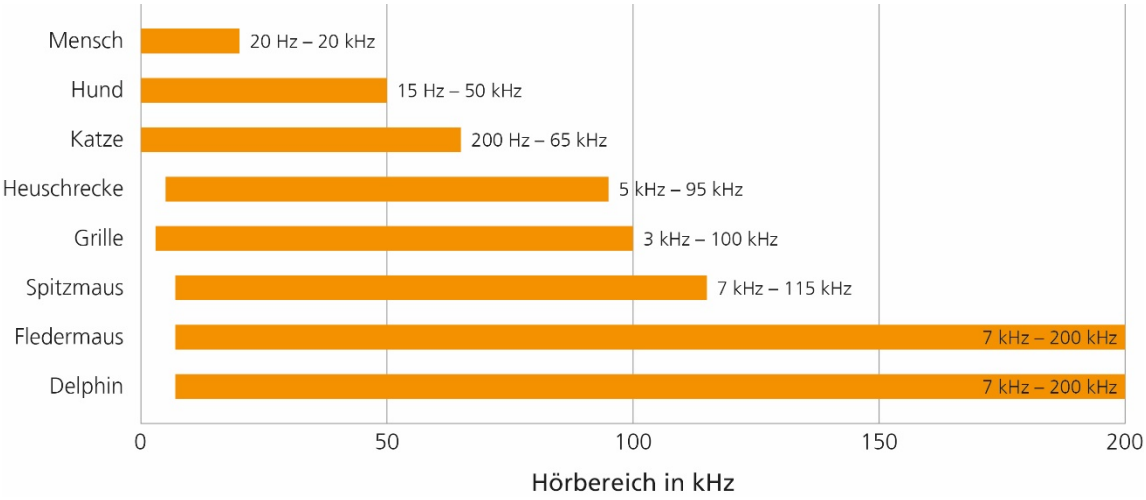


Abbildung 2: Hörbereich von Mensch und Tier (Quelle: http://www.leifiphysik.de)

[2] Das Signal wird an das Ohr weitergeleitet.

Gleichgewichtssinn: Mensch/Tier

Der Gleichgewichtssinn dient zur Bestimmung der Körperhaltung und Orientierung (oben, unten, Drehungen, Winkel und Beschleunigung) relativ zum Boden.

Das wichtigste Gleichgewichtsorgan ist der Vestibularapparat, der sich im Innenohr befindet. Drehbewegungen können durch die Bogengänge erfasst werden und Beschleunigungen werden im Sacculus und Utriculus erkannt.

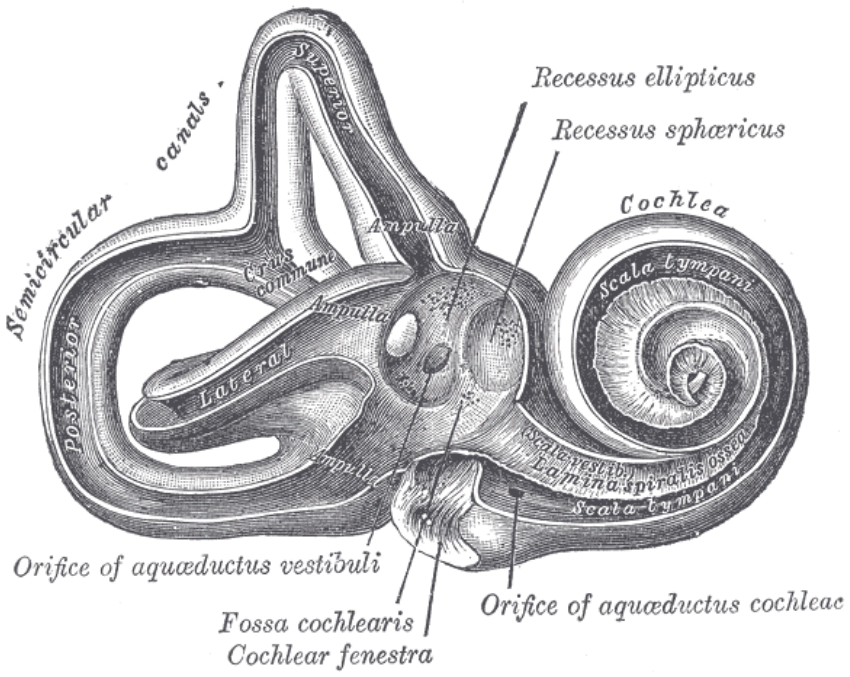


Abbildung 3: Der Aufbau des Vestibularapparates (Quelle: https://www.bartleby.com/107/illus921.html)

Bei Vögeln wurde unlängst ein zweites Gleichgewichtsorgan in seitlichen Auslappungen des Rückenmarks entdeckt. Es ist allein für die Kontrolle des Gehens und Stehens verantwortlich. Der Gleichgewichtssinn im Innenohr steuert hingegen die Bewegungen der Vögel im Flug (Quelle: http://de.wikipedia.org/wiki/Gleichgewichtsorgan, Juli 2010). Ein weiteres Beispiel zur Thematik »Gleichgewicht« findet sich in Band 4 – RCX »Themen und Experimente« der Roberta Reihe unter dem Kapitel Gangarten.

Tastsinn: Mensch/Tier

Als haptische Wahrnehmung (griech.: haptikos = greifbar) wird die Sinneswahrnehmung bestimmter mechanischer Reize der Haut und die Bewegungsempfindung durch Muskeln, Gelenken und Sehnen definiert. Eine Aufteilung erfolgt dabei in:

- taktiler Wahrnehmung (Druck, Berührung und Vibrationen, durch die Haut);
- kinästhetischer Wahrnehmung (Bewegung, durch Muskeln, Gelenke, Sehnen).

Der Tastsinn dient vor allem einer nahen räumlichen Orientierung.

Im Tierreich zeichnet sich der Maulwurf durch einen ausgeprägten Tastsinn aus. Dieser lässt sich in vier Regionen einteilen:

- Rhinarium (Rüsselscheibe)
- Vibrissen (»Schnurr«-Haare/Tasthaare im Gesichtsfeld),
- Tasthaarmanschette im Handwurzelbereich und einen
- Tastschwanz.

Mit Hilfe dieser Regionen kann er selbst kleinste Erschütterungen wahrnehmen und durch die Hinzunahme seines Gehörs die Richtung präzise zuweisen.

Geruchssinn: Mensch/Tier

An der olfaktorischen Wahrnehmung (griech.: ozein = riechen) sind zwei sensorische Systeme beteiligt:

- Das olfaktorische System → es enthält die auf die Wahrnehmung von Duftmolekülen spezialisierten Sinneszellen. Die Duftmoleküle werden in einen elektrischen Impuls umgewandelt und an das Gehirn weitergeleitet.
- Das nasal-trigeminale System → Nasale Trigeminus-Fasern reagieren erst ab höheren Konzentrationen als das olfaktorische Nervensystem.

Wie wichtig der Geruchssinn für den Menschen und beispielsweise für den Hund zur Orientierung sein kann, spiegelt allein die Größe des Geruchsorgans wieder: Mensch 2 x 5 cm², Hund 2 x 25 cm² (Quelle: http://de.wikipedia.org/wiki/Olfaktorische_Wahrnehmung, Juli 2010). Ein Mensch hat des Weiteren »nur« etwa 5 Millionen Riechzellen, ein Dackel etwa 125 Millionen und ein Schäferhund etwa 220 Millionen (Quelle: http://de.wikipedia.org/wiki/Haushund, Juli 2010). Dies erklärt auch, warum der Hund ein 1000fach feinere Nase als der Mensch besitzt.

Magnetisches Feld: Tier/Roboter

Vögel besitzen Moleküle auf ihrer Netzhaut, welche ihnen ermöglichen, sich am magnetischen Feld der Erde zu orientieren. Sie nutzen diese als eine Art Kompass.

In der Robotik wird der Kompass für die globale Navigation (Nord, Süd) eingesetzt. Im Abschnitt »NxtBot« wird ein Kompasssensor zur Richtungsbestimmung/Orientierung eingesetzt.

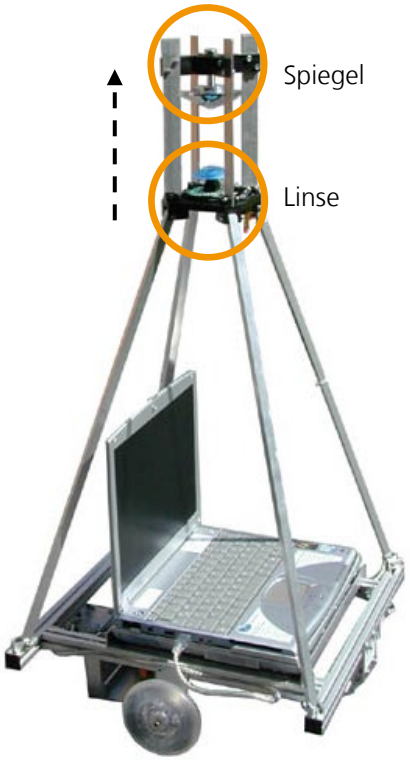
Sensoren

Kamerasysteme: Roboter

Kamerasysteme könnten auch als das Auge des Roboters bezeichnet werden. Weit verbreitet sind hier Mono, Stereo- und omnidirektionale (Rundum-) Kameras. Bei einer Omnikamera (Rundumkamera) ist die Linse auf einen konvex gewölbten Spiegel gerichtet, dadurch ergibt sich ein 360° Blickfeld.

Stereokameras besitzen in der Regel zwei nebeneinander angebrachte Objektive und ermöglichen die gleichzeitige Aufnahme der für 3D-Fotos erforderlichen stereoskopischen Halbbilder.

a)



b)

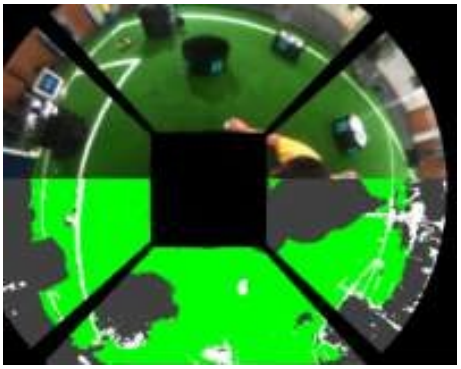


Abbildung 4: Omnikamera und Rundumbild (Quelle: VolksBot® Fraunhofer IAIS)

Laserscanner: Roboter

Das Laserscanning gehört genau genommen zu den Kamerasystemen. Allerdings handelt es sich dabei nicht um ein passives System (Lichtstrahlen werden aufgenommen), sondern um ein aktives System, bei dem Oberflächen oder Körper mittels Laserstrahlen abgetastet werden.

Beim 3D-Laserscanning werden die Konturen von Objekten und Räumen beispielsweise mit der Laufzeit des Laserstrahls digital erfasst. Dabei entsteht eine Menge von 3D-Abtastpunkten, die als Punktwolke bezeichnet wird. Die Koordinaten der gemessenen Punkte werden aus den Winkeln und der Entfernung in Bezug zum Ursprung (Gerätestandort) ermittelt (siehe Abbildung 5)[3].

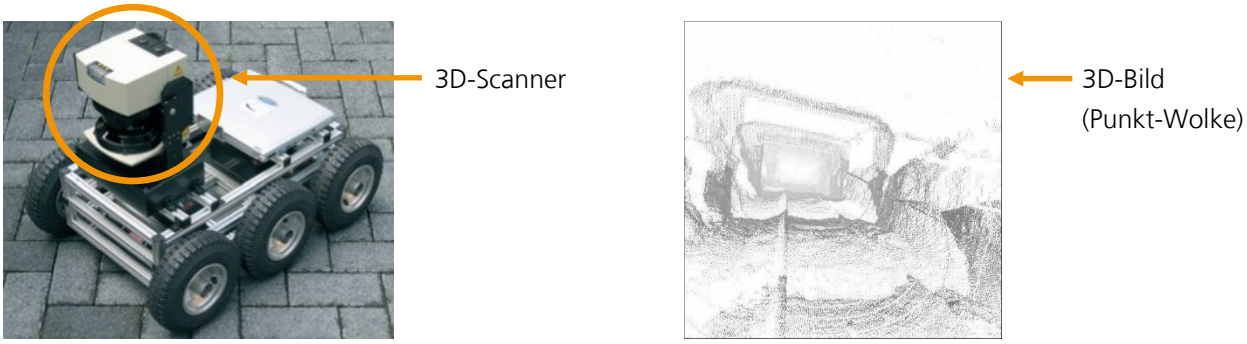


Abbildung 5: Kurt3D (links) 3D-Bild (rechts) (Quelle: Kurt3D Fraunhofer IAIS)

Ultraschall: Roboter

Ähnlich wie Tiere Ultraschall für die Orientierung einsetzen, nutzt auch die Robotik Ultraschall. Allerdings dienen Ultraschallsensoren Robotern eher zur Kollisionsvermeidung (als Einparkhilfen beim Auto) als zur Orientierung. Im Anschnitt »NxtBot« wird ein Ultraschallsensor eingesetzt, um Hindernissen kollisionsfrei auszuweichen.

GPS: Roboter

Das Global Positioning System (GPS) basiert auf Satelliten, die ständig Signale ausstrahlen, aus deren Signallaufzeit GPSEmpfänger ihre eigene Position bestimmen können.

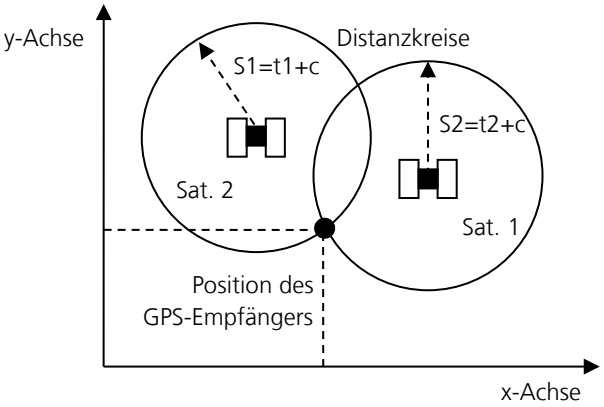


Abbildung 6: Berechnung der GPS-Koordinaten in der Ebene

[3] Siehe auch: http://www.volksbot.de/3dscan-de.php [01/2009]

Mit der Annahme, dass sich der GPS-Empfänger auf der Erdoberfläche befindet und eine sehr genaue Uhr besitzt, genügen, wie in Abbildung 6 gezeigt, theoretisch die Signale zweier Satelliten, um die (genaue) Ebenenposition zu bestimmen. Für die Positionsbestimmung im Raum (Position + Höhe) sind mindestens 3 Satelliten nötig. In der Praxis haben aber die meisten GPS-Empfänger keine Uhr, die genau genug ist, um daraus die Laufzeiten korrekt berechnen zu können. Deshalb wird meist das Signal eines vierten evtl. eines fünften Satelliten benötigt. Die Bestimmung der Richtung und der Geschwindigkeit kann erst nach einer gewissen Zeit, in der sich der GPS-Empfänger fortbewegt, erfolgen.

Klassifizierung von Sensoren

Sensoren können in drei Klassen unterteilt werden:

- Proprioception: Messungen der Eigenbewegung relativ zum internen Roboterkoordinatensystem.
- Exteroception: Messungen der Lage von Umgebungen und Objekten relativ zum Roboterkoordinatensystem.
- Exproprioception: Messung der Lage des Roboters oder seiner Teile relativ zur Umgebung.

3. Mögliche Experimente

Hinweis

Ein ebenfalls sehr gelungenes Experiment zum Thema Lokalisierung findet sich im Buch: Programmierung mit LEGO Mindstorms NXT von Karsten Berns und Daniel Schmidt.

Einführung

Eine für den Menschen sehr wichtige Fähigkeit ist, sich ohne die Orientierung zu verlieren nach vorn, zurück, links und rechts bewegen zu können. Gleichzeitig sind Menschen in der Lage, sich während dieser Bewegungen, um die eigene Achse zu drehen.

In ähnlicher Weise kann sich auch ein omnidirektional angetriebener Roboter bewegen, weshalb ein an das LEGO Mindstorms angepasster omnidirektionaler Roboter für eines der beiden Experimente ausgewählt wurde. Im zweiten Experiment wird der Standard NXT-Roboter (Tribot) benutzt und mit Sensorik ausgestattet.

In diesem Experiment wird gezeigt, wie ein LEGO Mindstorms® Roboter beim Navigieren einem Hindernis ausweichen kann. Folgende Modelle werden in den Experimenten beschrieben:

- ohne Sensoren (OmniBot) - RCX-Modell;
- mit Sensoren (NxtBot) - NXT-Modell.

Omnidirektionaler Antrieb

Merkmale

- Der Name omnidrive (omnidirektionaler Antrieb) ist abgeleitet von omni wheels (Allseitenräder).
- Es werden mindestens 3 Allseitenräder benötigt.
- Die Räder besitzen auf der Lauffläche kleine Rollen, deren Achsen quer zur Radachse liegen. Dies ermöglicht ihnen, eine passive Bewegung in Achsrichtung.

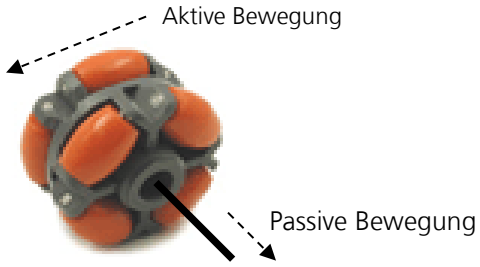


Abbildung 7: Bewegung von Allseitenrädern

- Die Allseitenräder sind sternförmig (120°) angeordnet.

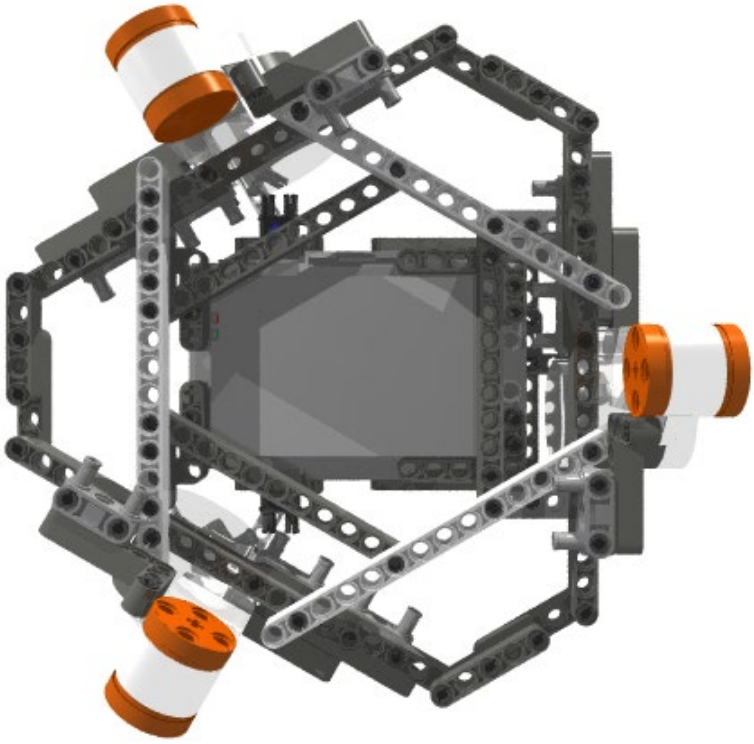


Abbildung 8: Anordnung der Allseitenräder

- Fahren in alle Richtungen (insbesondere zur Seite) ohne vorherige Rotation → somit kein Wendekreis.
- Redundanz → die gleiche selbe Bewegung ist mit verschiedenen Radkombinationen möglich
- Höherer Energieaufwand als bei einem herkömmlichen Zweiradantrieb.

NxtBot

Merkmale

- Der Roboter kann mit verschiedenen Sensoren ausgestattet werden.
- Es wird mindestens ein Sensor benötigt.
- Die Sensoren können auch miteinander kombiniert werden.

1 2 3 4 5

Abbildung 9: Verschiedene Sensoren zur Orientierung

Sensor 1: Lichtsensor

Sensor 2: Tastsensor

Sensor 3: Ultraschallsensor

Sensor 4: Soundsensor

Sensor 5: Kompasssensor

Beispiele zur einfachen Einführung der Sensoren:

Zu 1) Ziehen einer schwarzen Linie, über die der Roboter fährt.

Zu 2) Der Roboter tastet sich an einer Wand entlang.

Zu 3) Der Roboter orientiert sich anhand von Gegenständen.

Zu 4) Der Roboter folgt einem Geräusch[4].

Zu 5) Der Roboter orientiert sich anhand des Erdmagnetfelds.

[4] Z. B. mit 2 Geräuschsensoren mit der Annahme, dass das Geräusch an dem Sensor, der näher zur Geräuschquelle ist, lauter wahrgenommen wird. Experimente haben allerdings gezeigt, dass die NXT-Geräuschsensoren dafür nicht genau genug sind.

4. Experiment »Omnidirektionaler Antrieb«

Information
Omni directional steht für »in jede Richtung«.

Abbildung in die Roboterwelt
Für den Bau eines omnidirektional angetrieben Roboters werden zunächst drei Allseitenräder (siehe oben) benötigt. Die Allseitenräder werden nicht von LEGO® angeboten, sie können aber leicht, mittels einer geeigneten Nabe, mit den Standard Lego-Technik-Achsen verbunden werden.

Nachfolgend wird zusätzlich zur Konstruktion auch die Steuerung eines omnidirektionalen angetriebenen Roboters beschrieben. Diese beinhaltet die Klärung der wichtigsten Begriffe sowie die möglichen Steuerungsarten. Zusätzlich wird beschrieben, was es bei der Konstruktion eines omnidirektionalen Roboters (OmniBots) zu beachten gilt. Abschließend folgt die Programmierung des OmniBots mit und ohne Rotationssensoren.

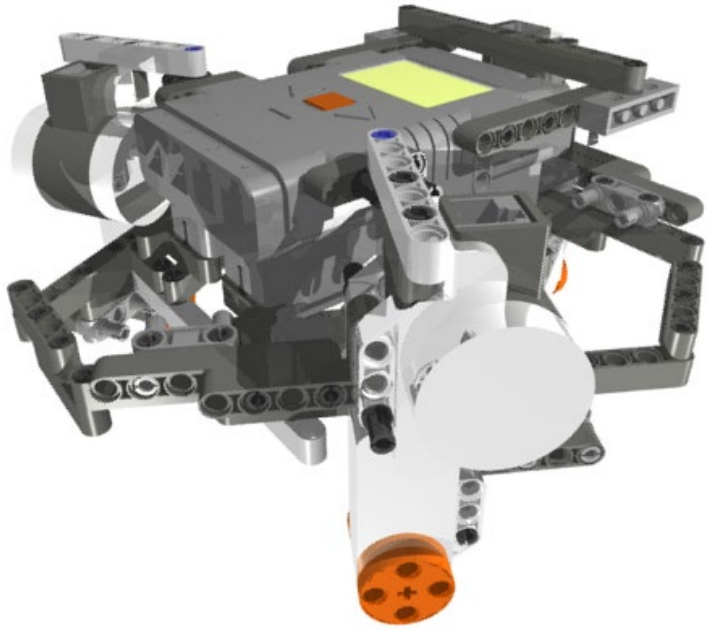


Abbildung 10: OmniBot (ohne Allseitenräder)

Die wesentlichen Konstruktionsschritte werden unter dem Punkt Konstruktion OmniBot gezeigt.

Der hier beschriebene OmniBot lässt sich leicht mittels NXT-RCX Adapterkapbel für die Motoren auch mit der NXT-Steuereinheit realisieren.

Dead Reckoning

Die Koppelnavigation (engl. Dead Reckoning) bildet die Basis vieler aktueller Navigationssysteme von landbasierten mobilen Systemen.

Nach der Initialisierung der Startposition wird anhand der gemessenen Daten (z. B. Motorgeschwindigkeit) über die Zeit die Nachfolgeposition errechnet. Dies kann entweder innerhalb (intern engl. on-board) eines Mobilen Systems (OmniBot → RCX) geschehen oder außerhalb des Mobilen Systems (PC).

Odometrie

Odometrie (von griech.: hodós, »Weg« und métron, »Maß«). Demnach entspricht Odometrie der Wegmessung, mit Hilfe derer die Positionsbestimmung eines Roboters beispielsweise durch die Beobachtung seiner Räder möglich ist. Die Richtungsänderung wird dabei automatisch, ständig aktualisiert. Als Bezugspunkt dient das Koordinatensystems des Roboters. Abbildung 11 zeigt dies exemplarisch für den OmniBot. Die x-Achse des Koordinatensystems zeigt in Fahrrichtung des Rades 1, die y-Achse liegt dazu senkrecht. Beide setzten im Mittelpunkt des Roboters an. Sie bilden die Orientierung des Roboters.

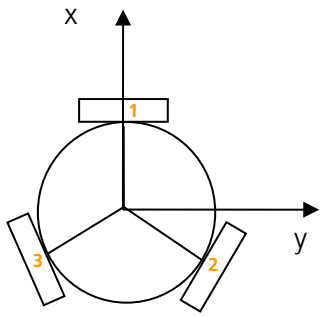


Abbildung 11: Koordinatensystem des Roboters

Die Odometrie ist ein grundlegendes Navigationsverfahren und zugleich die einfachste Form des Dead Reckoning. Aufgrund seiner Fehleranfälligkeiten (Schlupf, ungleiche Verteilung des Gewichtes auf einzelne Räder, etc.) wird es selten als einziges Verfahren zur Positionsbestimmung eingesetzt. So werden zusätzliche Sensoren (Kamerasysteme, Ultraschall etc.) zur Optimierung der Positionsbestimmung eingesetzt.

Odometer

Die für die Wegmessung eingesetzten Odometer (Wegmesser) lassen sich einteilen in:

- optischen Sensoren
  - z. B. Laserscanner zur Abstandsmessung von Objekten,
- Rotationssensoren
  - Messung der Rotationsgeschwindigkeit (z.B. wird gemessen, ob ein Lichtstrahl durch eine Scheibe tritt, die an eine Achse gekoppelt ist (Enkoder); Scheibe ist abwechselnd in schwarze – weiße Abschnitte codiert),
  - Berechnung der relativen Position,
- akustische Sensoren
  - z. B. Ultraschallsensoren (Schallwellen werden von Objekten in der Umgebung reflektiert)

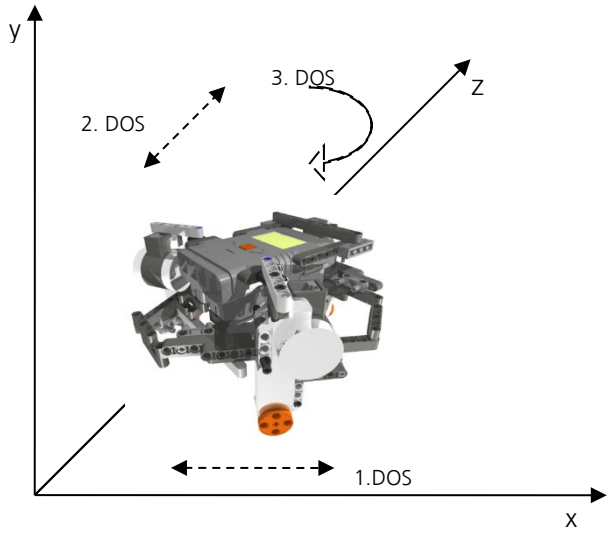
Im nachfolgen Experiment werden Rotationssensoren für die Wegmessung eingesetzt.

Translation

Der Begriff Translation (v. lat.: transferre (hin-)übertragen, bezeichnet im Allgemeinen eine Übertragung oder Übersetzung. In der Physik und in der Technik steht Translation für die geradlinige Bewegung eines Körpers, bei der alle seine Punkte eine parallele Bahn durchlaufen.

Freiheitsgrad

Unter Freiheitsgrad (engl.: Degrees of Freedom - DOS) versteht man die Anzahl verschiedener Richtungen, in die ein Körperteil oder ein mechanisches Bauteil bewegt werden kann.



= Andeutung der möglichen Richtungen

1. DOS → links, rechts

2. DOS → vor, zurück

3. DOS → rotation

Abbildung 12: Freiheitsgrade von OmniBot

Holonomie

Wie oben erwähnt, wird die Position eines Körpers in einer Ebene durch zwei Koordinaten eindeutig bestimmt. Die dafür verwendeten Koordinaten sind meist absolut (kartesisch oder polar). Zusätzlich beschreibt eine dritte Koordinate die absolute Orientierung (die Ausrichtung) des Körpers im Raum. Werden alle drei Koordinaten zusammen eingesetzt, spricht man von einem so genannten »Pose-System«.

Die Bewegung eines Körpers in einer Ebene wird durch drei entsprechende Freiheitsgrade festgelegt (siehe oben). Ist die Bewegung eines Körpers holonomisch, dann sind Position und Orientierung unabhängig voneinander veränderbar. Ein Auto kann sich zum Beispiel nicht holonomisch bewegen. Zum Abbiegen an einer Kreuzung muss es seine Orientierung ändern - man muss um die Kurve fahren. Dieses Fahrwerk ist in Geradeausrichtung sehr dynamisch. Der Nachteil eines omnidirektional angetriebenen Roboters liegt in der Abhängigkeit der drei Freiheitsgrade voneinander. Um in eine bestimmte Richtung fahren zu können, muss zunächst die Orientierung geändert werden, erst dann ist eine Translation in die gewünschte Richtung möglich. Dies kostet Zeit und schränkt Bewegungsabläufe stark ein.

Koordinaten des „Pose"-Systems:

- 2 für Position in der Ebene x-Achse, z-Achse (Translation)
- 1 für absolute Orientierung (Rotation)

→ Position und Orientierung sind unabhängig voneinander veränderbar.
→ Rotation ist unabhängig von Translation

Steuerung OmniBot

Für eine einfache Translation sollen die Räder bei einem omnidirektionalen Antrieb so gesteuert werden, dass diese sich zueinander drehen (siehe Abbildung 13a). Dadurch wird vermieden, dass das nicht angesteuerte Rad (hier mit der Nr. 1) die Richtung negativ beeinflusst. Allerdings geht dadurch ein Teil der Flexibilität verloren. Alternativ wäre auch noch die Variante wie in Abbildung 13b als Antriebsart möglich. Hier wird zusätzlich das Rad Nr. 1 mitbenutzt. Dieses wird abwechselnd nach links und nach rechts gesteuert. Der OmniBot ist somit, auf Kosten der Genauigkeit, beim vorwärts Fahren etwas schneller unterwegs.

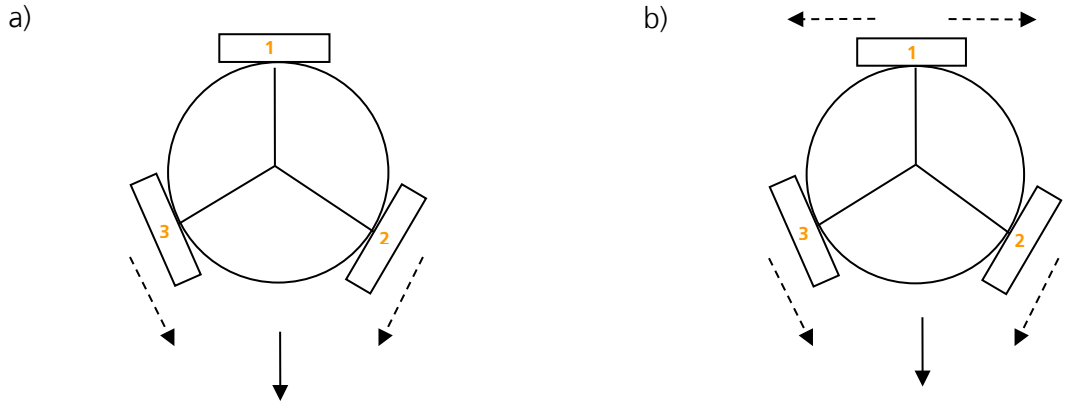


Abbildung 13: a) Antrieb mit zwei Rädern I b) mit drei Rädern

Werden alle drei Räder mit der gleichen Geschwindigkeit und in die gleiche Richtung angetrieben (siehe Abbildung 14), dreht sich der OmniBot auf der Stelle (hier im Uhrzeigersinn). Dies veranschaulicht gleichzeitig, dass ein omnidirektional angetriebener Roboter keinen Wendekreis besitzt.

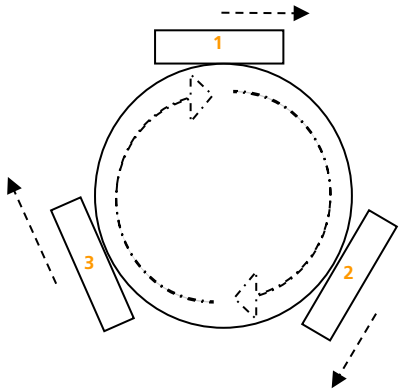


Abbildung 14: Drehen auf der Stelle

Um den Vorteil eines omnidirektionalen Antriebs an einem praktischen Alltagsbeispiel zu zeigen, wird der Einparkvorgang von OmniBot in Abbildung 15 b), im Vergleich zu einem »herkömmlichen« Automobil a) demonstriert.

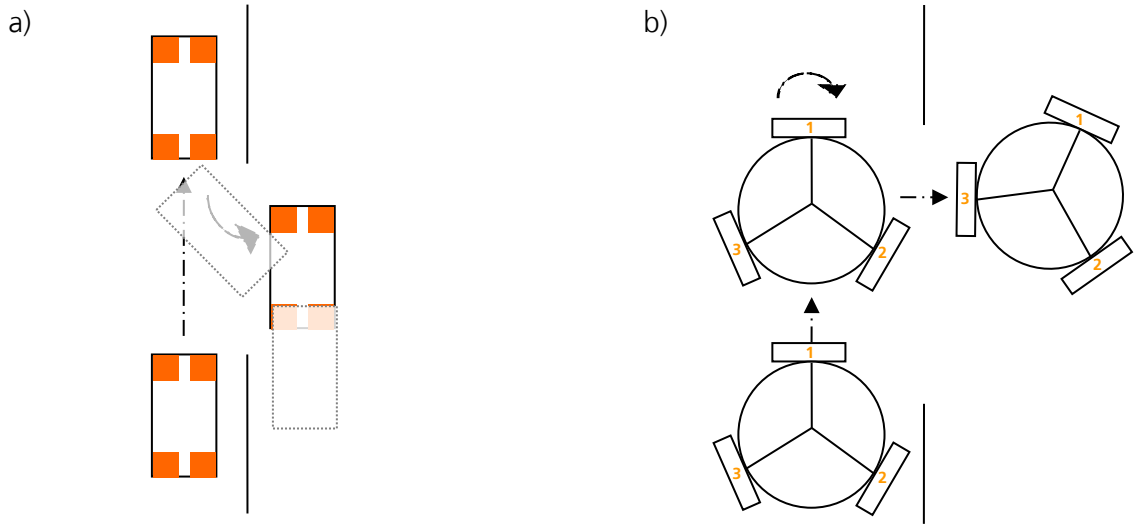


Abbildung 15: Einparkvorgang Auto a) und OmniBot b)

Steuerbefehle OmniBot

Grundsätzlich gibt es drei Steuerbefehle für jeden omniBot-Motor, »Aus« – »Vorwärts« – »Rückwärts«. Daraus ergeben sich 27 (=3³) Möglichkeiten, den OmniBot anzusteuern.

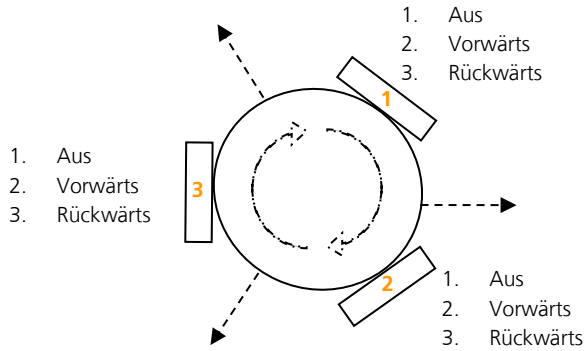


Abbildung 16: Drei Steuerbefehle für je einen Motor

Die wichtigsten Steuerbefehle werden aus der nachfolgenden Tabelle ersichtlich.

Motor 1	Motor 2	Motor 3	Fahrrichtung
Vorwärts	Aus	Rückwärts	300°
Vorwärts	Rückwärts	Aus	0°
Aus	Vorwärts	Rückwärts	240°
Rückwärts	Vorwärts	Aus	180°
Rückwärts	Aus	Vorwärts	120°
Aus	Rückwärts	Vorwärts	60°

Tabelle 1: Wichtige Steuerbefehle des OmniBot

Abbildung 17 verdeutlicht, welche Richtung wie viel Grad bezogen auf x und y-Richtung entspricht.

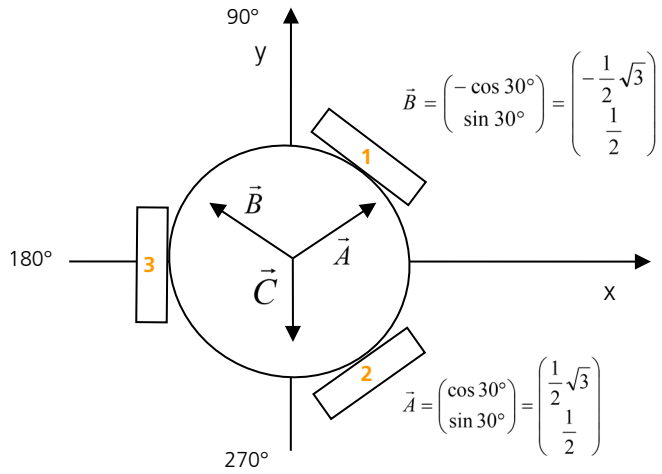


Abbildung 17: Einteilung der Richtung in Vektoren

Wie eine gradgesteuerte Lenkung des OmniBot eingesetzt wird und welche zusätzlichen Voraussetzungen dies mit sich bringt, behandelt der nächste Punkt »Steuerung mit Rotationssensoren«.

Steuerung OmniBot mit Rotationssensoren

Die oben eingeführten Richtungen und deren Angabe in Grad/Vektoren, können auch durch die Einheitsvektoren A, B und C beschrieben werden (siehe Abbildung 18 a). Zur Vereinfachung können die Vektoren im Zentrum des OmniBots angelegt werden b).

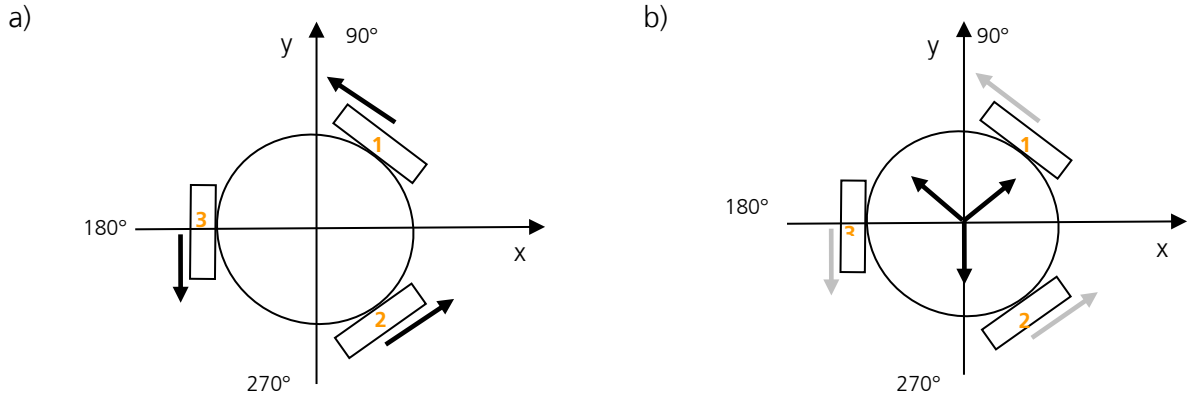


Abbildung 18: Einheitsvektoren

Um allerdings mit den Vektoren arbeiten zu können, werden die technischen Daten des OmniBot benötigt.

Hierzu zählen

- der Raddurchmesser,
- die Geschwindigkeit,
- das Übersetzungsverhältnis,
- die Umrechnung in Rotationsticks.

Technische Daten OmniBot

Der Raddurchmesser der Allseitenräder beträgt 4 cm. Wird dieser mit Pi (Π=3,1415…) multipliziert, ergibt sich ein Radumfang von **12,566 cm**. Aus diesem Radumfang errechnet sich die zurückgelegte Wegstrecke von 12,56 cm pro kompletter Radumdrehung.

Die Getriebemotoren besitzen eine Drehzahl von 350 u/min. Umgerechnet auf Sekunden erhält man eine Drehzahl von 5,8333 u/sek (350/60). Bei einer Übersetzung von 3:1 ergibt sich die endgültige Drehzahl von → 5,8333/3 = **1,9444/sek.**

Wird die Drehzahl mit dem Radumfang von 12,566cm multipliziert ergibt das eine theoretische Geschwindigkeit von **24,63cm/sek.** Aufgrund der der Anordnung der Allseitenräder (Geradeausbewegung des OmniBot im 60° Winkel zu den Radachsen) besitzt der OmniBot einen höheren Schlupf als andere Antriebstypen, der zu einer weit geringern Geschwindigkeit führt.

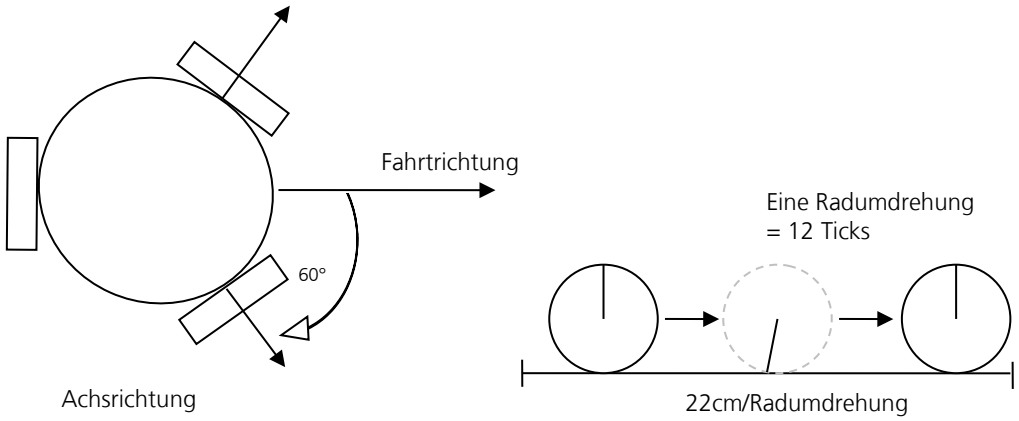


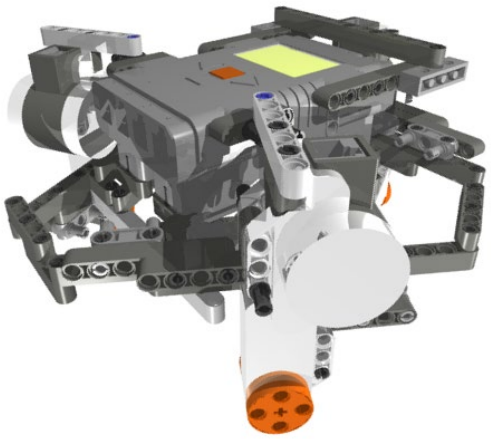
Abbildung 19: Fahrtrichtung und Radumdrehung

In mehreren Versuchsreihen legte der OmniBot innerhalb einer Sekunde auf glattem Untergrund zwischen 13cm und 15cm zurück. Somit beträgt die mittlere Geschwindigkeit etwa **14cm/sek.**

Die Übersetzung des Rotationssensors beträgt 1:5 (eine Umdrehung des Rotationssensors entspricht fünf Umdrehungen des Motors). Somit ergibt sich, dass der Rotationssensor bei einer kompletten Radumdrehung 10 »Ticks« ausgibt. Auf die Wegstrecke während einer Sekunde werden 12 »Ticks« gemessen. Dies ergibt ein Verhältnis von **12 »Ticks«** pro 14cm.

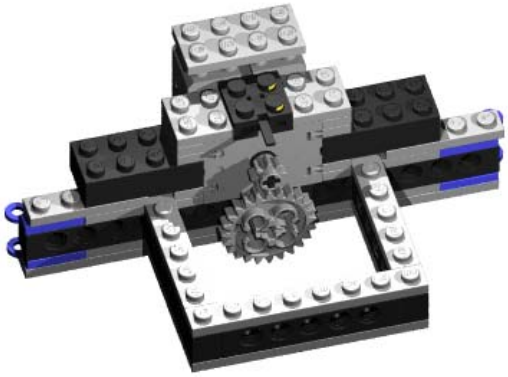
Zusammenfassung der technischen Daten

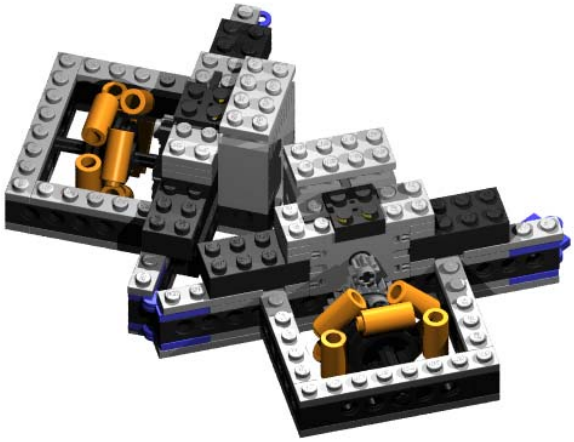
- Raddurchmesser 4cm
- Radumfang 12,56cm
- Drehzahl 350/min bzw. 5,833/sek
- Übersetzung 1:3
- Drehzahl an den Allseitenrädern 1,944
- Theoretische Geschwindigkeit 24,63/sek
- Übersetzung Rotationssensor 1:5
- Ticks pro Radumdrehung 10
- Praktische Geschwindigkeit 14cm/sek
- Abhängig vom Untergrund
- 12 Ticks pro Sekunde



Konstruktion

OmniBot





Überlegung zur Programmierung

Die nachfolgende Beschreibung ist angelehnt an die Publikation »Wendige Mini-Kicker« von Michael Rauls und Thomas Stahl 2002 (Quelle: http://www.rids.de/downloads/publications/wendigeminikicker-jufo2002.pdf)

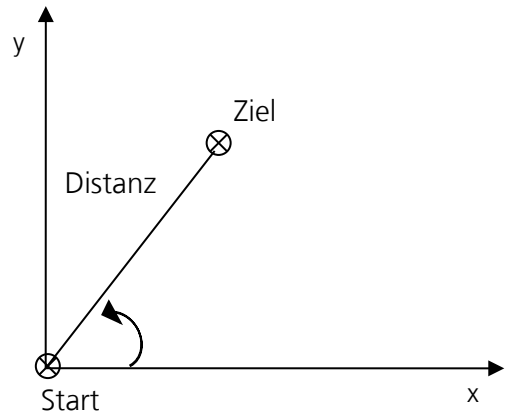


Abbildung 20: Bewegung des Körpers ohne Drehung

Wir betrachten die Bewegung des Körpers ohne Drehung um sein Zentrum. Die Richtung der Bewegung gibt der Einheitsvektor $\vec{V}$ an. $\vec{V}$ muss aus den drei Basisvektoren linear kombiniert werden:

$$\vec{V} = \alpha \cdot \vec{A} + \beta \cdot \vec{B} + \gamma \cdot \vec{C} \qquad (1)$$

Die Skalare α, β und γ sind proportional zur Winkelgeschwindigkeit des entsprechenden Rades. Damit eine Rotation um das Zentrum verhindert wird, müssen die Skalare folgende Nebenbedingung erfüllen:

$$\alpha + \beta + \gamma = 0 \qquad (2)$$

$$\leftrightarrow -(\alpha + \beta) = \gamma \qquad (3)$$

Hiermit haben wir eine recht einfache Vorschrift zur Bestimmung der einzelnen Radgeschwindigkeiten für eine Translation ohne Rotation gefunden.

Aus diesen Formel folgen nach Herleitung (siehe Quelle) die Formeln zur Errechnung der Skalare α, β und γ, um in die Richtung des Vektors $\vec{V}$ zu fahren.

$$\alpha = \frac{2}{3}\vec{A} \cdot \vec{V} \qquad (4)$$

$$\beta = \frac{2}{3}\vec{B} \cdot \vec{V} \qquad (5)$$

$$\gamma = \frac{2}{3}\vec{C} \cdot \vec{V} \qquad (6)$$

Soll der Roboter in Richtung der Y-Achse fahren (in der Grafik entspricht dies 90° bzw. dem Vektor in Richtung x-Achse).

$$\alpha = \frac{2}{3}\vec{A} \cdot \vec{V} \qquad (7)$$

Setzen wir nun die ermittelten Werte aus Abbildung 17 ein, erhalten wir folgende Gleichungen:

$$\alpha = \frac{2}{3} \cdot \begin{pmatrix} \frac{1}{2} \cdot \sqrt{3} \\ \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{2}{3} \cdot \frac{1}{2} = \frac{1}{3} \qquad (8)$$

$$\beta = \frac{2}{3} \cdot \begin{pmatrix} -\frac{1}{2}\sqrt{3} \\ \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{2}{3} \cdot \frac{1}{2} = \frac{1}{3} \qquad (9)$$

$$\gamma = \frac{2}{3} \cdot \begin{pmatrix} 0 \\ -1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{2}{3} \cdot -1 = -\frac{2}{3} \qquad (10)$$

Die Skalare α = 1/3, β =1/3 und γ =-2/3 stellen nun das Verhältnis dar, mit welchen Geschwindigkeiten die Motoren des Roboters angesteuert werden müssen. Sehr leicht lässt sich die Geschwindigkeit der Motoren über Pulsweitenmodulation regulieren. Hier werden die Motoren A und C jeweils nur halb so lange angesteuert, wie Motor B (der die ganze Zeit angesteuert wird).

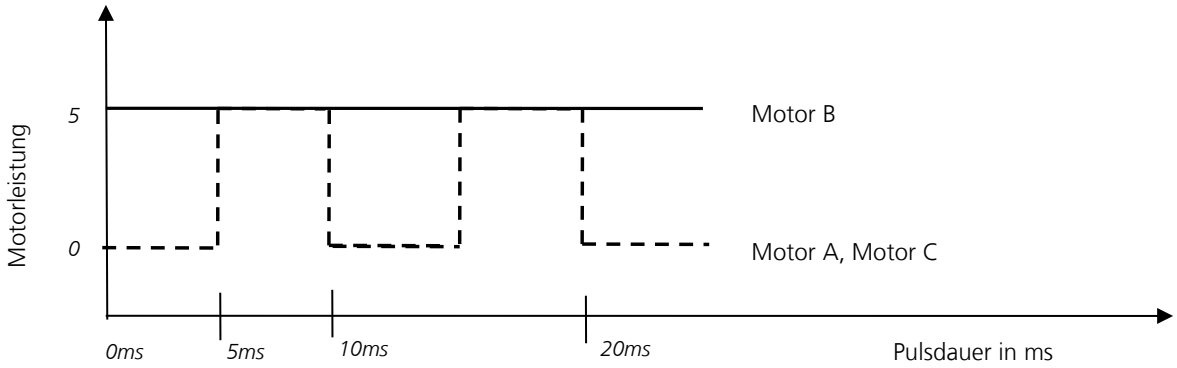


Abbildung 21: Pulsweitenmodulation

Im leJOS-Beispielprogramm wird dies wie folgt umgesetzt: Motor B läuft für die gesamte Dauer (`int milliseconds`), während Motor A und C jeweils alle 10ms (ein Zyklus) für 5ms angehalten werden.

Beispiel

```
void driveRight(int milliseconds) {
      int timer = (int)System.currentTimeMillis();
            //Motoanstreuung nach Pulsweitenmodulation
            //Motor B fährt doppelt so lang wie die beiden anderen
            Motoren

            while ((int)System.currentTimeMillis() < timer +
            milliseconds) {
                  Motor.A.backward();
                  Motor.B.forward();
                  Motor.C.backward();
                  wait(5);
                  Motor.A.stop();
                  Motor.C.stop();
                  wait(5);
            }
                  Motor.A.stop();
                  Motor.B.stop();
                  Motor.C.stop();
}
```

Soll der Roboter in eine andere Richtung fahren, lässt sich die Ansteuerung der Motoren leicht durch die oben genannten Formeln berechnen. Das Beispielprogramm zeigt das Fahren und Drehen des Roboters in verschiedene Richtungen. Das vollständige leJOSProgramm findet sich im Anhang und als downloadbare java-Datei im geschützten Bereich des Roberta-Portals.

5. Experiment »NxtBot«

Information

Einfach Beispiele und Aufgaben mit einem Sensor finden sich im Roberta Band 1 und Band 1-NXT. Die Verwendung der Lichtsensoren und Tastensensoren zur Orientierung findet sich in abgewandelter Form bei der Beschreibung des Experiments »Labyrinthe « in Band 4 der Roberta-Reihe.

Abbildung in die Roboterwelt

Der Bau des TriBot-Roboters (siehe Abbildung 22) findet sich sowohl in Band 1-NXT der Roberta-Reihe als auch in von Lego Education mitgelieferten Bauanleitung.

Dieser Roboter wird zusätzlich mit einem Kompasssensor und einem Ultraschallsensor (US) ausgestattet.

Nachfolgend wird die Funktionsweise der einzelnen Sensoren beschrieben sowie die Visualisierung der gelieferten Sensordaten und die Programmierung des Roboters. Abschließend wird erläutert, wie beide Sensoren sinnvoll miteinander kombiniert werden können und wie der Rotationssensor der Motoren genutzt wird.

Abbildung 22: NXT-TriBot

Navigation mit Kompassensor

Der digitale Kompasssensor[5] misst das Erdmagnetfeld. Im Lese-Modus kann der Sensor die aktuelle Position des Roboters im Bezug zum Erdmagnetfeld ermitteln. Im Kalibrier-Modus kann der Sensor angepasst werden, um zum Beispiel extern verursachte Schwankungen des Magnetfelds, wie sie z. B. in der Nähe von Batterien oder Elektromotoren auftreten, auszugleichen. Der Kompasssensor funktioniert nur bei horizontaler Ausrichtung und Einsatz einer NXT Fimware-Version höher 1.03. Es wird empfohlen den Sensor mindestens 15 cm von den Motoren entfernt und 10 cm vom NXT-Baustein entfernt zu montieren.

Der Sensor hat eine Auflösung von 1° und gibt einen Wert zwischen 0 und 359 aus. Dabei stellt 0 Norden da, 90 Osten 180 Süden und 270 Westen (siehe Abbildung 23).

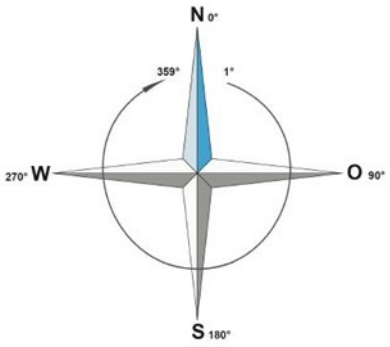


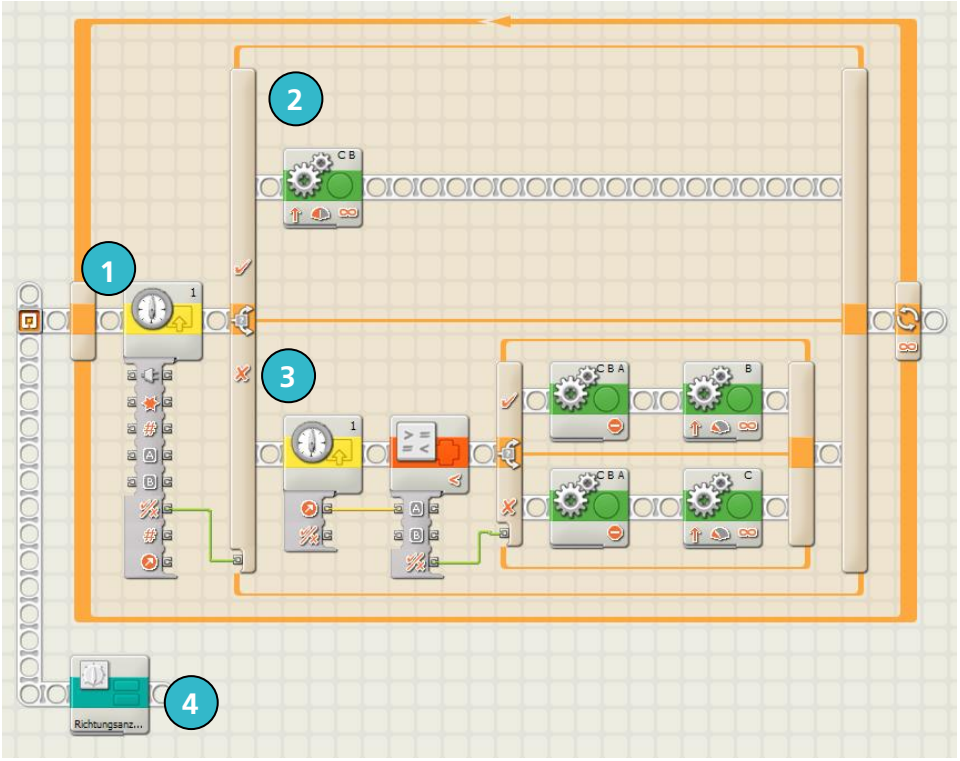
Abbildung 23: Messbereich des Kompassensors

Die genaue Spezifikation des Sensors findet sich im Kapitel Hardwareevaluation von Band 4 – NXT der Roberta-Reihe.

[5] Laut Angaben der Firma HiTechnic halten alle HiTechnic-Sensoren die von LEGO geforderten Spezifikationen bzgl. Volt (4.3 Volt) und Ampere (20mA) ein.

NXT-G Programm: Kompassensor

Dieses NXT-G Programm gibt zum einen die Werte des Kompasssensors aus 4) (Eigener Block) und zeigt dadurch an, in welche Richtung der Roboter gerade fährt; zum anderen können gewünschte Richtungen über den Kompasssensorblock[6] angegeben werden.



1) der Kompassblock gibt einen boolean Wert (Wahr bzw. Falsch) zurück, falls der aktuelle Wert (+/- Toleranzbereich) mit dem gewählten Wert übereinstimmt (Wahr) bzw. nicht übereinstimmt (Falsch). Bei Nicht-Übereinstimmung 3) wird verglichen, ob der Wert des Sensors größer oder kleiner des Vergleichswerts ist, daraufhin vollzieht der Roboter eine Links- bzw. Rechtsdrehung. Das Programm beginnt anschließend wieder bei 1). Bei Übereinstimmung 2) fährt der Roboter gerade aus.

[6] Der Kompassblock zum Programmieren des Sensors kann kostenlos bei der Herstellerfirma heruntergeladen werden.

LeJOSProgramm

Das folgende leJOS Programm[7] nutzt zur Navigation die bereits vorhandene Klasse *CompassPilot* (aus der leJOS API), um den Roboter nach Norden fahren zu lassen:

Zunächst wird der Kompasssensor kalibriert (`robot.calibrate()`). Hierfür dreht sich der Roboter einmal komplett um die eigene Achse. Anschließend wird der Roboter nach Norden (359 – siehe Abbildung 23) ausgerichtet (`robot.rotateTo(359)`). Mit `robot.travel(500)` wird nun die Distanz angegeben, die der Roboter nach Norden fahren soll.

```
import lejos.nxt.*;
import lejos.robotics.navigation.*;

public class CompassPilotTest
{
      static CompassPilot robot = new
      CompassPilot(SensorPort.S1,5.6f,16.0f,Motor.A, Motor.C);

      public static void main(String[] args ) throws Exception
      {
            //Wait for user to press ENTER
            robot.calibrate(); // Kompasssensor wird kalibriert
            robot.rotateTo(359); // Ausrichtung nach Norden
            robot.travel(500); // Fahrtrichtung Distanz von 500
      }

      public static void pause(int time)
      {
            try{ Thread.sleep(time);
            }
            catch(InterruptedException e){}
      }
}
```

[7] Verwendet wurde die leJOS-Version 0.7.0beta

Ultraschallsensor

Damit der Roboter auch Hindernissen ausweichen kann, empfiehlt es sich, einen Ultraschallsensor anzubringen.

Der Ultraschallsensor ist der einzige digitale Sensor, der standardmäßig von LEGO im NXT-Set mitgeliefert wird. Der Ultraschallsensor ermittelt den Abstand zu Gegenständen, indem er 12 Signalstöße bei 40 kHz aussendet und die benötigte Zeit misst, bis er eine Reflektion des ausgesandten Signals empfängt. Der messbare Distanzbereich liegt zwischen 0 cm und 255 cm und kann daher durch ein einziges Byte kodiert werden. Weitere Details zum Ultraschallsensor finden sich im Kapitel Hardwareevaluation von Band 4 – NXT.

Bevor der Sensor zum kontaktfreien Umfahren von Hindernissen genutzt wird, sollen die gemessenen Distanzen auf dem Display des Roboters grafisch dargestellt werden (siehe Abbildung 24).

Abbildung 24: Visualisierung der Sensordaten auf dem NXTDisplay

Hierfür werden die Werte des Sensors auf der y-Achse abgetragen, während die x-Achse des Displays pro Zyklus um eine Position erhöht (inkrementiert) wird.

NXT-G Programm: Ultraschall-Distanzwerte

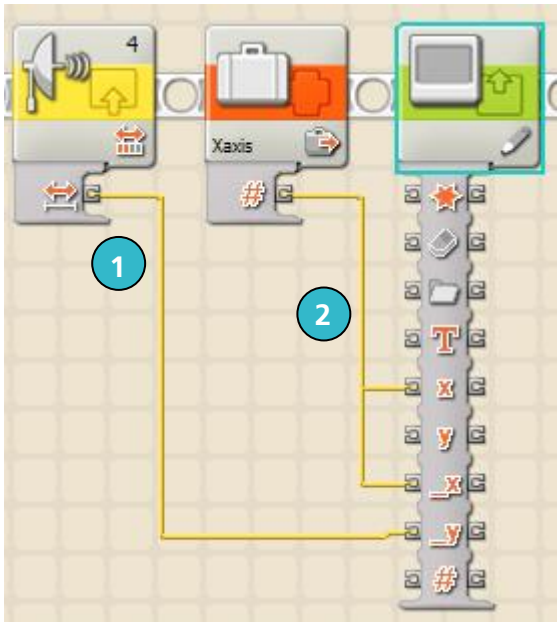


Abbildung 25: Teil eines NXT-G Programms

Dieses Programm ermöglicht, die gemessenen Distanzen durch den Ultraschallsensor mit den realen Abständen auf einen Blick zu vergleichen. Hierzu werden die Werte des Ultraschallsensors per Datenleitung 1) an Display des NXT (an die Schnittstelle für die x und y-Achse) weitergeben. Zusätzlich werden in 2) dem Display die Werte für die x-Achse eingespeist. Somit können die (y-) Werte des Sensors auf der x-Achse des Displays angezeigt werden.[8]

Für eine genaue Evaluierung des Sensors in Bezug auf die Messgenauigkeit und Reichweite empfiehlt es sich jedoch, Zahlenwerte auszugeben.

Hindernisvermeidung und Zielorientierung

Der Roboter kann nun so programmiert werden, dass Hindernissen ausgewichen werden kann und zugleich die ursprüngliche Himmelsrichtung (Zielrichtung) weiter beibehalten wird.

Das Programm »Radar_Display« befindet sich im internen Bereich des Roberta-Portals (www.roberta-home.de).

Abbildung 26 zeigt ein Nassi-Shneiderman-Diagramm des Programmentwurfs, bei welchem der Roboter auf auftretende Hindernisse reagiert. Wurde kein Hindernis detektiert bzw. einem Hindernis erfolgreich ausgewichen, wird das Ziel angefahren. Die Abfrage befindet sich innerhalb einer Endlosschleife.

[8] Die zusätzliche Verwendung des _x und _y Eingangs am NXT-Display erzeugt eine Linie zwischen den gemessenen Punkten (x,y).

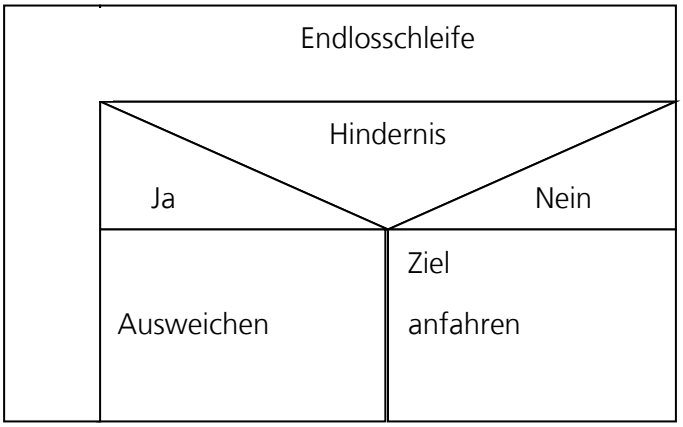


Abbildung 26: Nassi-Shneiderman-Diagramm

Die Umsetzung des Programmentwurfs in ein NXT-G Programm zeigt Abbildung 27.

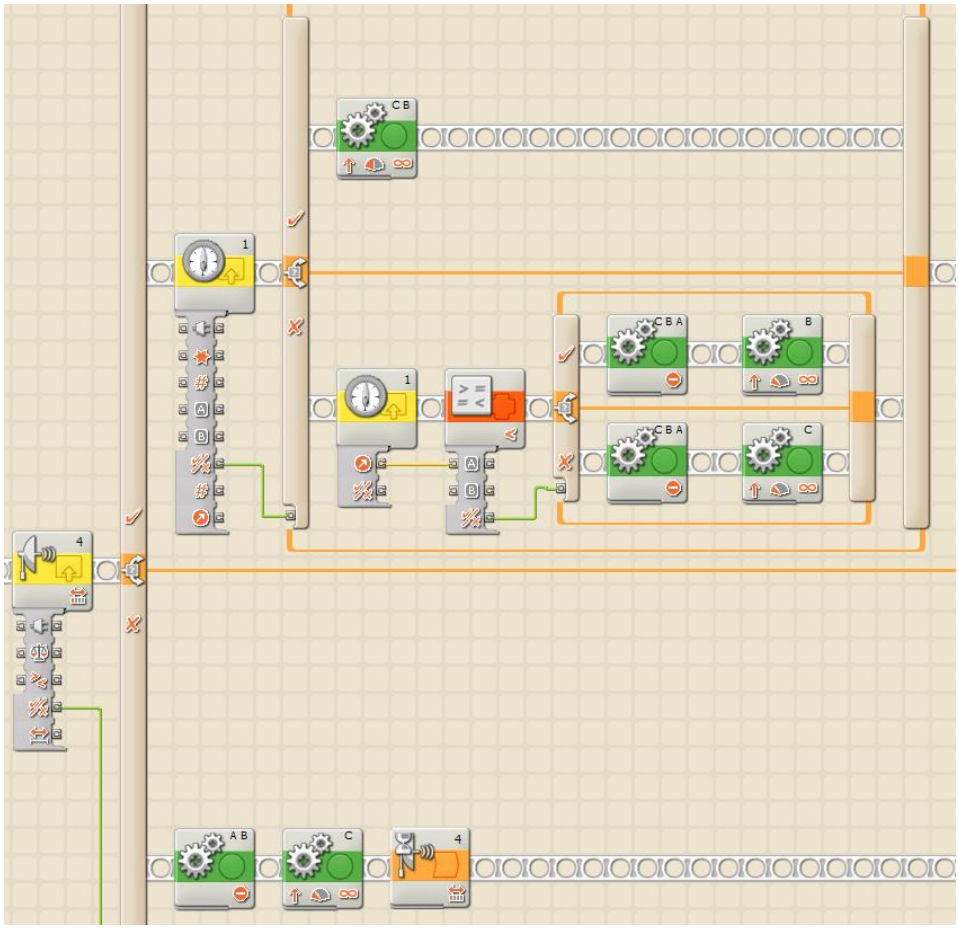


Abbildung 27: NXT-G Programm mit Kompass- und Ultraschallsensor

Hindernisvermeidung und Odometrie

Unter Verwendung der integrierten Motorencoder der NXT-Motoren (siehe Abbildung 28) ist es dem NxtBot zusätzlich zur Kollisionsvermeidung möglich, die zurückgelegte Wegstrecke aufzuzeichnen.

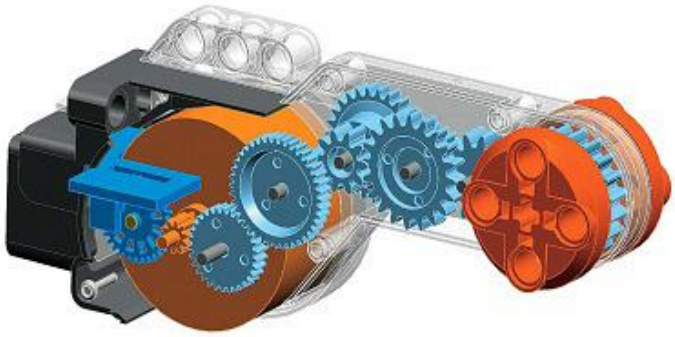


Abbildung 28: Aufbau des NXT-Motors

Zur Aufzeichnung der gefahrenen Strecke wird der NXT-G Block „Drehsensor“ und die Verwendung von Variablen benötigt.

Abbildung 29: Drehsensor und Variablen-Blöcke

Der Drehsensorblock registriert die Motorumdrehung (Motor-Ticks). Diese können mittels Datenleitungen an Variablen weitergegeben werden. Die Parameter der Variablen müssen hierfür auf »`Zahl`« und »`Schreiben`« festgelegt werden. Variablen können im Menü der Programmierumgebung (LME) über die Menüpunkte → »Bearbeiten« → »Variablen definieren« erzeugt werden.

Abbildung 29 zeigt zwei Variablen, die beide Zahlenwerte (#) aufnehmen können. Variable_1 kann gespeicherte Werte ausgeben (lesen), Variable_2 kann Werte aufnehmen (schreiben).

Abbildung 30 veranschaulicht, wie und wie viele Variablen zur Messung des zurückgelegten Weges benötigt werden.

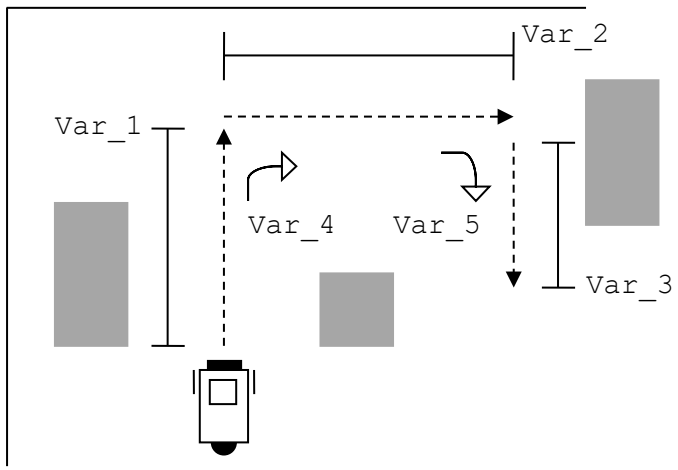


Abbildung 30: Messen der Wegstrecke und Richtung

Pro Strecke und Richtung wird eine Variable benötigt und mit den entsprechenden Werten beschrieben. Nach Beendigung des Parcours soll der NxtBot denselben Weg, zurück zum Ausgangspunkt fahren, ohne dabei Sensoren zur Hindernisvermeidung zu nutzen.

Gelangt der Roboter an ein frontales Hindernis, dreht er sich nach rechts (90°-Drehung). Ist dort wiederum ein Hindernis, muss er zwei Mal eine Rechtsdrehung vollziehen, um in die andere (entgegen gesetzte Richtung) zu gelangen. Bei einer Sackgasse muss der NxtBot somit insgesamt sechs Rechtsdrehungen vollziehen.

Werden nur Rechtsdrehungen zugelassen, kann die Anzahl der Drehungen in eine einzige Variable geschrieben werden.

Hinweis

Bei der Programmierung mit der grafischen Oberfläche NXT-G müssen die benötigten Variablen vorab angelegt werden. Sind die angelegten Variablen mit den Odometriewerten beschrieben, können während der Abarbeitung des Programms keine weiteren Variablen angelegt werden. Deshalb empfiehlt es sich, vor dem Start die während der Erkundung benötigten Variablen zu ermitteln und dementsprechend im Programm vorab anzulegen.

Abbildung 31 zeigt exemplarisch wie die verschiedenen Richtungen (rechts, links, zurück) festgehalten werden können.

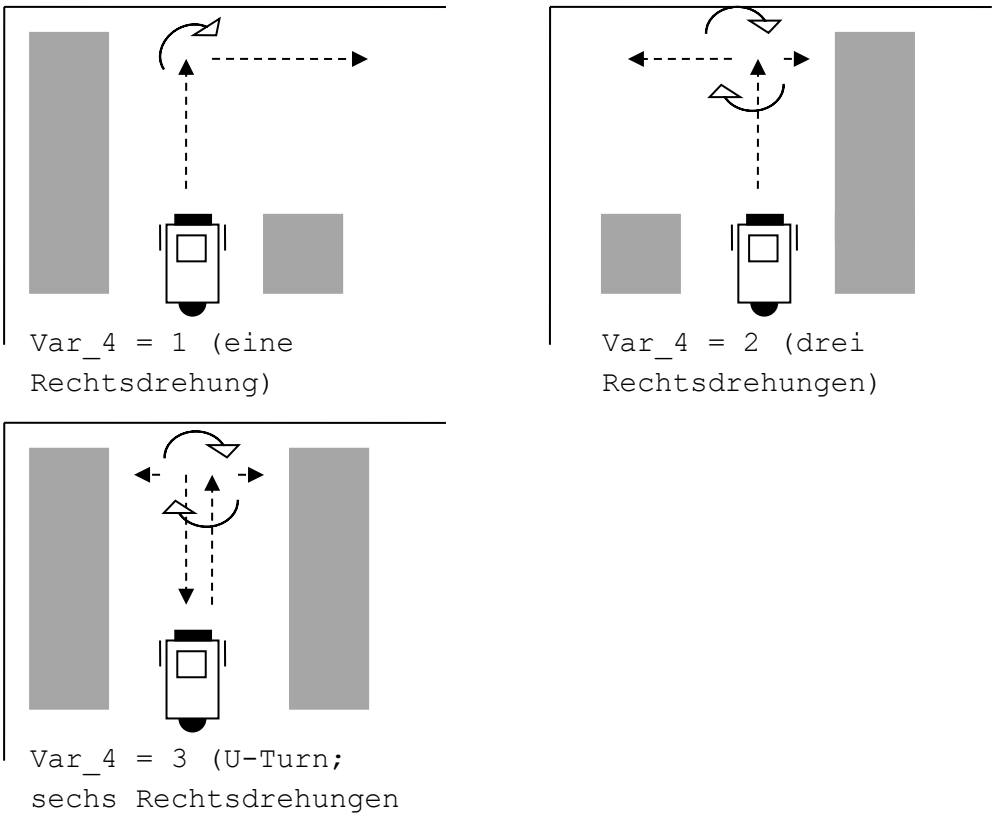


Abbildung 31: Speichern der Richtung

Dabei werden die jeweiligen Kurven (links, rechts, U-Turn) mit folgende Werten (Motorumdrehungen) gespeichert:

- Rechtskurve entspricht 360° (1x Motorumdrehung) Linker Motor
- Linkskurve entspricht 1080° (3x Rechtsdrehung) Linker Motor
- U-Turn entspricht 2160° (6x Rechtsdrehung) Linker Motor

Anhand dieser Werte kann beim Zurückfahren unterschieden werden, welche Kurve gefahren wurde, ohne dass die kompletten Drehungen des Hinweges nachgefahren werden müssen.

Hinweis
Beachten Sie bitte, dass es neben den hier aufgeführten Vorschlägen weitere Lösungs-möglichkeiten gibt.

Die Umsetzung dieser Vorgehensweise kann zum Beispiel durch einen wie in Abbildung 32 gezeigten Schalter erfolgen.

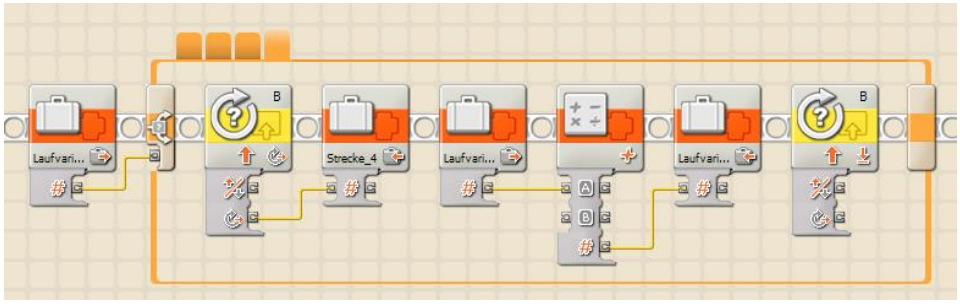


Abbildung 32: Verbinden eines Schalter-Blocks mit einer Variable

Dabei wird der Schalter über eine Datenleitung mit einer Variablen verbunden. Dies ermöglicht es, mehr als nur zwei Unterscheidungen (wie sonst beim Schalter üblich) zu treffen. Der Schalter wie in Abbildung 32 zeigt, vier Zuständen:

Laufvariable == 1; Laufvariable == 2; Laufvariable == 3; Laufvariable == 4

Je nachdem, welchen Wert die Laufvariable hat, wird der zwischen den Klammern liegende Block ausgeführt. Je nachdem, welchen Zustand der Roboter hat, wird die Laufvariable um 1 erhöht (inkrementiert); siehe auch Abbildung 33.

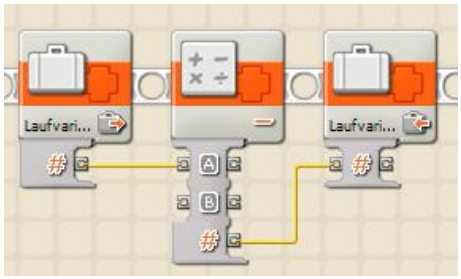


Abbildung 33: Erhöhung der Laufvariablen

Das gesamt Programm »Path-Finding V2« findet sich im internen Bereich des Roberta-Portals (www.roberta-home.de)

Literaturverzeichnis

Buch

Berns, Karsten; Schmidt, Daniel.
Programmierung mit LEGO Mindstorms NXT: Robotersysteme, Entwurfsmethodik, Algorithmen, Berlin 2010.

Web

www.mpg.de/bilderBerichteDokumente/dokumentation/jahrbuch/2005/psycholinguistik/forschungs-Schwerpunkt2/pdf.pdf
Ein Bericht aus der Kognitionsforschung der Max-Planck-Gesellschaft. Hierin wird anhand der funktionalen Magnet-Resonanz-Tomographie (fMRT) gezeigt, wie das menschliche Gehirn Orientierung ermöglicht.

www.kontexis.de/dateien/special/Raeumeerleben.pdf
Erklärt das Entstehen der räumliche Orientierung und dessen Zusammenhangmit mathematischem Verständnis.

Linkliste

http://kisd.de/~krystian/nxt/ [12/2009]
Internetseite mit Beispielen zur Nutzung des Ultraschallsensors als Radar.

http://www.planetwissen.de/natur_technik/voegel/zugvoegel/index.jsp [11/2008]
Eine leichtverständliche und mit Animationen angereicherte Seite, rund um Vogelschwärme.

http://www.planetwissen.de/natur_technik/tierisches/tierische_orientierung/index.jsp
Erläutert leicht verständlich, verschiedene Arten der Orientierung im Tierreich.

www.innovationsreport.de/html/berichte/biowissenschaften_chemie/bericht-33847.html [11/2008]
Erläutert, wie Vögel das Magnetfeld der Erde wahrnehmen.

http://www.wdr.de/tv/quarks/global/pdf/Q_Voegel.pdf [11/2008]
Beschreibt die Orientierung von Vögeln

Anhang

Programmierung »OmniBot«

Programm OmniBot ohne Sensoren

Einstiegsprogramm in NQC

```
task main()
{
      SetSensorType (SENSOR_1, SENSOR_TYPE_ROTATION);
      SetSensorType (SENSOR_2, SENSOR_TYPE_ROTATION);
      SetSensorType (SENSOR_3, SENSOR_TYPE_ROTATION);

      start odometrie;
}

task odometrie()
{
      while(true){
            OnRev(OUT_A+OUT_C);
      }
}
```

Programm das alle drei Motoren der Allseitenräder für die Vorwärtsfahrt nutzt.

```
task main()
{
      SetSensorType (SENSOR_1, SENSOR_TYPE_ROTATION);
      SetSensorType (SENSOR_2, SENSOR_TYPE_ROTATION);
      SetSensorType (SENSOR_3, SENSOR_TYPE_ROTATION);

      start odometrie;
}

task odometrie()
{
      OnRev(OUT_A+OUT_C);
      while(true){
            OnRev(OUT_B);
            Wait(50);
            OnFwd(OUT_C);
            Wait(50);
      }
}
```

Programm OmniBot mit Sensoren

```
task main()
{
      SetSensorType (SENSOR_1, SENSOR_TYPE_ROTATION);
      SetSensorType (SENSOR_2, SENSOR_TYPE_ROTATION);
      SetSensorType (SENSOR_3, SENSOR_TYPE_ROTATION);
      SetPower (OUT_A+OUT_C,8);

      CreateDatalog (600);

      start odometrie;
}

task odometrie()
{
      OnRev(OUT_A+OUT_C);

      while(true){

            AddToDatalog(SENSOR_1); //SENSOR_1 mist MOTOR_A
            AddToDatalog(SENSOR_3); //SENSOR_2 mist MOTOR_C

            while(SENSOR_1<SENSOR_3){
                  SetPower(OUT_A,8);
                  SetPower(OUT_C,4);
                  AddToDatalog(SENSOR_1);
                  AddToDatalog(SENSOR_3);
            }
            while(SENSOR_3<SENSOR_1){
                  SetPower(OUT_C,8);
                  SetPower(OUT_A,4);
                  AddToDatalog(SENSOR_1);
                  AddToDatalog(SENSOR_3);
            }
      }
}
```

Programm – Omnidrive.java

Programm mit PWM Steuerung

```
import josx.platform.rcx.*;

class Omnidrive {
      int power;
      nt radiusWithAllMotosOn = 45;
      public Omnidrive() {
            setPower(5); // 0 kleinste und 7 größte
// Sensor.S1.setTypeAndMode(4, 0xE0);
// Sensor.S1.setTypeAndMode(0, 0x00);
      Sensor.S1.setTypeAndMode(SensorConstants.SENSOR_TYPE_ROT,
      SensorConstants.SENSOR_MODE_ANGLE);
            Sensor.S1.activate();
            Sensor.S1.setPreviousValue(0);
            Sensor.S2.setTypeAndMode(SensorConstants.SENSOR_TYPE_ROT,
            SensorConstants.SENSOR_MODE_ANGLE);
            Sensor.S2.activate();
            Sensor.S2.setPreviousValue(0);
            Sensor.S3.setTypeAndMode(SensorConstants.SENSOR_TYPE_ROT,
            SensorConstants.SENSOR_MODE_ANGLE);
            Sensor.S3.activate();
            Sensor.S3.setPreviousValue(0);
      }
      static void main(String[] args) {
            Omnidrive omnidrive = new Omnidrive();
            omnidrive.driveForward(2000);
            wait(1000);
            omnidrive.driveBackward(2000);
            wait(1000);
            omnidrive.turnLeft(2000);
            wait(1000);
            omnidrive.turnRight(2000);
            wait(1000);
            omnidrive.driveLeft(3000);
            wait(1000);
            omnidrive.driveRight(3000);
            wait(1000);
            omnidrive.driveLeftCurve(45, 7000);
            wait(1000);
            omnidrive.driveLeftCurve(20, 7000);
            wait(1000);
            omnidrive.driveRightCurve(45, 7000);
            wait(1000);
            omnidrive.driveRightCurve(20, 7000);
            wait(1000);
```

```
/*                while (true) {
                  LCD.refresh();
// LCD.showNumber(Sensor.S1.readValue());
                  LCD.setNumber(0x3001, Sensor.S3.readValue(), 0x3002);
                        wait(100);
                        if(Sensor.S1.readValue() >= 100) {
                              Sensor.S1.setPreviousValue(0);
                        }
                  }*/
      }
      void setPower(int p) {
            power = p;
            Motor.A.setPower(power);
            Motor.B.setPower(power);
            Motor.C.setPower(power);
      }

      void driveForward(int milliseconds) {
            Motor.A.backward();
            Motor.C.forward();
            wait(milliseconds);
            Motor.A.stop();
            Motor.C.stop();
      }

      void driveBackward(int milliseconds) {
            Motor.A.forward();
            Motor.C.backward();
            wait(milliseconds);
            Motor.A.stop();
            Motor.C.stop();
      }

      void driveLeft(int milliseconds) {
            int timer = (int)System.currentTimeMillis();
            while ((int)System.currentTimeMillis() < timer +  milliseconds) {
                  Motor.A.forward();
                  Motor.B.backward();
                  Motor.C.forward();
                  wait(5);
                  Motor.A.stop();
                  Motor.C.stop();
                  wait(5);
            }
            Motor.A.stop();
            Motor.B.stop();
            Motor.C.stop();
      }
```

```
	void driveRight(int milliseconds) {
		int timer = (int)System.currentTimeMillis();
//Motoanstreuung nach Pulsweitenmodulation
//Motor B fähr nur halbsolang wie die beiden anderen Motoren
		while ((int)System.currentTimeMillis() < timer +  milliseconds) {
			Motor.A.backward();
			Motor.B.forward();
			Motor.C.backward();
			wait(5);
			Motor.A.stop();
			Motor.C.stop();
			wait(5);
		}
		Motor.A.stop();
		Motor.B.stop();
		Motor.C.stop();
	}
	void driveLeftCurve(int radius, int milliseconds) {
		int timer = (int)System.currentTimeMillis();
		while ((int)System.currentTimeMillis() < timer +  milliseconds) {
// fährt wenn alle Motoren an sind etwa 45 cm Radius (radiusWithAllMotosOn
// bei alle anderen Radien muss die Geschw. prozentual angepasst werden
			Motor.A.backward();
			Motor.B.forward();
			Motor.C.forward();
			if (radius > radiusWithAllMotosOn) {
				wait(radiusWithAllMotosOn);
				Motor.B.stop();
				wait(radius - radiusWithAllMotosOn);
			}
			else {
				wait(radius);
				Motor.A.stop();
				Motor.C.stop();
				wait(radiusWithAllMotosOn-radius);
			}
			Motor.A.stop();
			Motor.B.stop();
			Motor.C.stop();
		}
	}
	void driveRightCurve(int radius, int milliseconds) {
		int timer = (int)System.currentTimeMillis();
		while ((int)System.currentTimeMillis() < timer +  milliseconds) {
// fährt wenn alle Motoren an sind etwa 45 cm Radius (radiusWithAllMotosOn)
// bei alle anderen Radien muss die Geschw. prozentual angepasst werden
			Motor.A.backward();
			Motor.B.backward();
			Motor.C.forward();
```

```
                if (radius > radiusWithAllMotosOn) {
                      wait(radiusWithAllMotosOn);
                      Motor.B.stop();
                      wait(radius - radiusWithAllMotosOn);
                }
                else {
                      wait(radius);
                      Motor.A.stop();
                      Motor.C.stop();
                      wait(radiusWithAllMotosOn-radius);
                }
                Motor.A.stop();
                Motor.B.stop();
                Motor.C.stop();
          }
    }
    void turnRight(int milliseconds) {
          Motor.A.backward();
          Motor.B.backward();
          Motor.C.backward();
          wait(milliseconds);
          Motor.A.stop();
          Motor.B.stop();
          Motor.C.stop();
    }
    void turnLeft(int milliseconds) {
          Motor.A.forward();
          Motor.B.forward();
          Motor.C.forward();
          wait(milliseconds);
          Motor.A.stop();
          Motor.B.stop();
          Motor.C.stop();
    }
    public static void wait(int time) {
          try {
                if (time > 0) {
                      Thread.sleep(time);
                }
          }
          catch (InterruptedException e) {
                // nothing
          }
    }
}
```

Kontakt

Die Roberta-Initiative im Web

roberta-home.de

lab.open-roberta.org

FAQ rund um die Roberta-Initiative

roberta-home.de/faq

Informationen zum Datenschutz

roberta-home.de/datenschutz

Info

Verfasser: Thorsten Leimbach, Thomas Breuer I Roberta-Initiative

© 2004-2018 Fraunhofer-Institut für Intelligente Analyse und Informationssysteme IAIS

Version: 1.0

Stand: Oktober 2010

Warenzeichen

LEGO® und Mindstorms™ sind eingetragene Warenzeichen der Firma The LEGO Group.

Microsoft®, Windows® und Internet Explorer® sind eingetragene Warenzeichen der Firma Microsoft Corporation.

Java™ ist eingetragenes Warenzeichen der Firma SUN Microsystems Inc.

Roberta, Open Roberta und NEPO sind eingetragen Warenzeichen der Fraunhofer-Gesellschaft e.V.

ROBERTA
INITIATIVE