Thema:
Einführung in LEGO® Mindstorms ®

Seminararbeit
Im Rahmen des Seminars „Unterstützung von Landminendetektion durch Bildauswertungsverfahren und Robotereinsatz"

Institut für Informatik

Themensteller: Dr. Dietmar Lammers
Betreuer: Dr. D. Lammers, Dipl.-Inform. St. Wachenfeld
Vorgelegt von: Stefanie Filius
Sandra Kühn
Stefan Schellhammer

Abgabetermin: 05.01.2004

Inhaltsverzeichnis

1 Einleitung 3
1.1 Motivation der Arbeit 3
2 Definition und Klassifikation von Robotern 4
3 LEGO® Mindstorms™ 5
3.1 Geschichte von LEGO® Mindstorms™ 5
3.2 Technikbauteile 5
3.2.1 Motoren 5
3.2.2 Getriebe 6
3.3 Mindstorms™-Bauteile 9
3.3.1 RCX 9
3.3.2 Sensoren 9
3.4 Software 15
3.4.1 Architektur 15
3.4.2 Betriebssysteme und Sprachen 17
4 Grenzen, Probleme von LEGO® Mindstorms™ und deren Lösungsmöglichkeiten am Beispiel eines Raum-Kartier Roboters 20
4.1 Anforderungen an das Kartieren eines Raumes 20
4.1.1 Aufgabenstellung 20
4.1.2 Szenario 20
4.1.3 Anforderungen an einen Kartierungsroboter 20
4.1.4 Problemlösungsstrategien 22
4.2 Realisierung 25
5 Schlussbetrachtung 27
A Anhang 28
A.a Literaturverzeichnis 28

1 Einleitung

1.1 Motivation der Arbeit

Ziel dieser Arbeit ist es, eine möglichst weitreichende Einführung in den Roboterbau mit LEGO® Mindstorms™ zu geben. Viele damit in Verbindung stehende Themen wie Konstruktionen, Konzepte und Möglichkeiten von LEGO® Mindstorms™ werden betrachtet, wobei aber auch auf damit zusammenhängende Probleme und Grenzen eingegangen wird. Da die Begriffe LEGO® Mindstorms™ und Roboter praktisch untrennbar sind, wird zunächst der Begriff des Roboters definiert. Darauf aufbauend werden, nach einem kurzem geschichtlichen Abriss, einige wichtige Technic™ und Mindstorms™ Bauteile vorgestellt und deren Funktionsweise erläutert. Einige softwaretechnische Besonderheiten in Bezug auf die Programmierung von LEGO® Mindstorms™ Roboter werden hiernach erläutert.

Den Zusammenhang zwischen dem Seminarthema "Unterstützung von Landminendetektion [...]" und dieser Arbeit ist durch das Beispiel Kartieren eines Raumes gegeben, welches ebenfalls die Erkundung eines unbekannten Gebietes zur Aufgabe hat. Bei der Betrachtung dieses Raum-Kartier-Roboters werden zunächst Anforderungen definiert und Problemfelder aufgezeigt und im weiteren Verlauf mögliche Problemlösungsmöglichkeiten präsentiert.

2 Definition und Klassifikation von Robotern

Dieser Abschnitt dient dazu, den im Laufe dieser Arbeit sehr häufig verwendeten Begriff Roboter zu definieren und Kriterien aufzuzeigen, mit deren Hilfe LEGO® Roboter eingeordnet werden können.

Es existiert eine große Vielfalt an divergierenden Definitionen für Roboter angefangen von „Apparaturen menschlicher Gestalt[I] " bis hin zu einem reinen „Arbeitsgerät und Automat". Eine weitreichende und unserer Meinung nach passende Definition liefert, im Bezug auf LEGO® Mindstorms™, die MS Encarta 2000, welche einen Roboter als ein „Selbständiges, programmierbares, elektromechanisches Gerät" definiert, das „in der Industrie und wissenschaftlichen Forschung für jeweils eine spezielle Aufgabe oder eine begrenzte Anzahl von Aufgaben eingesetzt wird." Des Weiteren sagt sie: „Roboter stellen eine Unterkategorie von Automaten dar. Obwohl keine generell anerkannten Kriterien existieren, die Roboter von anderen Automaten unterscheiden,[...]".

Des Weiteren kann ein Roboter durch fünf Merkmale[II] charakterisiert werden. Dabei handelt es sich um Gehirn, Körper, Energiequelle, Aktoren und Sensoren. Auf diese Komponenten wird im Rahmen dieser Arbeit je nach Bedeutungsschwere mehr oder weniger intensiv eingegangen.

Roboter lassen sich nach der Stelle der Entscheidungskompetenz in ferngesteuerte und autonome, sowie nach dem Grad der Bewegungsfreiheit in angeleinte und mobile Roboter unterteilen. Bei den von uns im Rahmen dieser Arbeit betrachten LEGO® Mindstorms™ Robotern handelt es sich ausschließlich um autonome und mobile Roboter.

[I] Duden Fremdwörterbuch 1999
[II] Knudsen J., Noga M.: LEGO®Mindstorms™ Roboter

3 LEGO® Mindstorms™

3.1 Geschichte von LEGO® Mindstorms™

Die Entwicklung des dieser Arbeit zugrunde liegenden LEGO® Mindstorms™ Robotics Invention System 2.0 (RIS 2.0) findet seine Ursprünge in der Erfindung des ersten „Automatic Binding Brick“[I] - dem ersten LEGO® Stein - im Jahre 1949. Nächster Meilenstein auf dem Weg zum RIS war die Markteinführung der LEGO® Technic™ Serie, welche gewissermaßen den Grundstock an „Ressourcen“ für die erfolgreiche, stabile Konstruktion eines Mindstorms™ Roboters bietet. Jedoch vollzog sich die eigentliche Erfindung des LEGO® Mindstorms™ Sets im eher wissenschaftlichen Rahmen. Ausschlaggebend dafür war das Buch „Mindstorms“ von Prof. Dr. Seymour Papert am Massachusetts Institute of Technology, MIT, womit er auch zum Namensgeber des späteren Produktes wurde. In diesem Buch geht er auf die große Chance ein, Kindern Problemlösungsstrategien durch Programmierung zu vermitteln. Aufgegriffen und weiterverfolgt wurde diese Idee dann vom LEGO® Konzern, der in der Folgezeit das Forschungsprojekt am MIT sponserte. Aus dieser Forschung entstand der „programmable brick“, das „Gehirn“ des Roboters. Er stand Pate für das spätere Endprodukt des LEGO® Konzerns. Dieser brachte 1998 die neue Marke „LEGO® Mindstorms™“ mit dem RIS 1.0 auf den Markt. Kernstück war der RCX, eine LEGO® Version des „programmable brick“.

3.2 Technikbauteile[II]

3.2.1 Motoren

Im LEGO® Mindstorms™ Set sind Getriebemotoren enthalten. Diese werden mit 9V bei 500 mA betrieben. Sie weisen verglichen mit den LEGO® Standard Motoren ein höheres Drehmoment bei dementsprechend niedrigerer Drehzahl auf. Maximal kann eine Drehzahl von 300U/min erreicht werden, diese jedoch fällt unter Last ab oder mit nachlassendem Ladezustand der Batterien.

Abbildung 1: LEGO® Getriebemotor

Es gibt drei Modi in denen ein Motor laufen kann: On (vorwärts und rückwärts), Off und Floating. Im Floating Modus ist die Achse frei beweglich, wird aber nicht angetrieben.

[I] Vgl. http://www.lego.com/eng/info/default.asp?page=timeline6 , LEGO® Timeline

[II] ausführlich in Hystad, Dean „Building LEGO® Robots For FIRST™ LEGO® League“, 2002

Eine direkte Einstellung der gewünschten Motordrehzahl ist nicht möglich. Es kann lediglich eine der acht vorgegebenen Motorleistungsstufen ausgewählt werden.

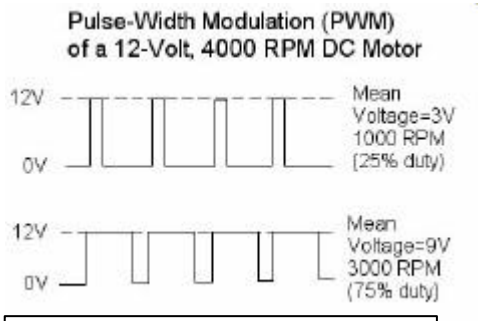

Abbildung 2

Aufgrund der primär digitalen Arbeitsweise des RCX besteht keine Möglichkeit ein Signal auszusenden, welches den Motor veranlassen könnte auf halber Kraft zu arbeiten. Um dies dennoch erreichen zu können, wird eine Pulse Width Modulation (PWM) vorgenommen. Dabei wird der Motor im ständigen Wechsel zwischen dem On und Floating Modus betrieben.

Ein Schwungrad verwandelt die pulsierende Bewegung in einen gleichmäßigen Fluss. Das bedeutet, dass beim Starten des Motors zusätzliche Kraft benötigt wird, um das Schwungrad mit anzutreiben. Doch wenn sich der Roboter in Bewegung gesetzt hat und der Untergrund gleichmäßig beschaffen ist, erreicht er eine konstante Drehzahl. Sobald sich jedoch dem Rad nur ein geringer Widerstand entgegenstellt, blockiert das Rad, da im Floating Modus keine Kraft gegen den Widerstand hält.

3.2.2 Getriebe

Die Motoren ermöglichen eine Fortbewegung des Roboters. Jedoch ist es beim Roboterbau oft nicht möglich den Motor an der Position anzubringen, an welcher die mechanische Kraft, beispielsweise zur Fortbewegung, benötigt wird. Des Weiteren können an einem Motor ganz unterschiedliche Anforderungen in Bezug auf Drehgeschwindigkeit, Drehmoment und Drehrichtung gestellt werden. Daher sollte bereits in einem frühen Stadium des Roboterbaus eine geeignete Übersetzung für den Roboter ausgewählt werden. Wurde dies nicht beachtet, kann es passieren, dass sich der Roboter nicht mit den angestrebten Leistungsparametern[I] fortbewegt und unter Umständen eine komplette Neukonstruktion erforderlich ist.

Welche Möglichkeiten gibt es nun sinnvolle Getriebe mit LEGO® Bausteinen zu bauen? Die Technic™-Serie stellt dazu diverse Bausteine zur Verfügung. Im folgenden sollen nun die wichtigsten Vertreter und ihr jeweiliges Einsatzgebiet vorgestellt werden.

[I] Leistung[KW]:
Die Leistung[P_e] einer Antriebswelle ergibt sich aus dem Produkt von Drehzahl[n] und dem Drehmoment[M]. Während die Drehzahl die Anzahl Achsenumdrehungen pro Zeiteinheit angibt, ergibt sich das Drehmoment aus dem Produkt der Länge des Hebelarms[r] und der auf ihn wirkenden Kraft[F]. Somit ergeben sich folgende Formeln:
$P_e = M*n$
$M = r*F$

Zahn-/Kron- und Kegelräder

Zahnräder zählen wohl zu den am häufigsten verwendeten Bauteilen in einem Getriebe. Durch Aneinanderreihung von mehreren dieser Zahnradkombinationen in Verbindung mit LEGO®-Achsen können diese zu komplexen Getriebezügen zusammengebaut werden. Dadurch kann eine beliebige Drehmoment-Drehzahl Kombination hergestellt werden kann.

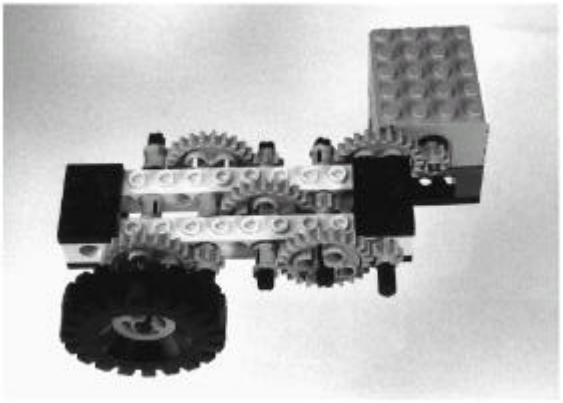

Abbildung 3

Es gibt Zahnräder mit 8, 16, 24 und 40 Zähnen. Dabei ist der Radius, der für die Übersetzung[I] wichtig ist, proportional zur Anzahl der Zähne. Jedoch ist zu beachten, dass jedes angeschlossene Zahnrad die Bewegungsrichtung umkehrt. So ist erst bei einer ungeraden Anzahl ineinander greifender Zahnräder die Bewegungsrichtung des Antriebsrades wiederhergestellt. Anhand der Abbildung lässt sich die daraus resultierende Übersetzung exemplarisch berechnen. Dabei soll die Antriebseinheit an der auf der rechten Seite der Abbildung befindlichen langen Welle angeschlossen werden. Berechnet wird nun der Output am Ende des Zuges an der letzten Welle. Zur Konstruktion wurden nur Kombinationen aus 24er und 8er Zahnräder verwendet, die in folgender Weise zusammengeschaltet wurden: 3:1*3:1*3:1*3:1*3:1 ⇔ 243:1[II] .

Riemenscheiben

Ähnlich wie Zahnräder werden auch Riemenscheiben und die dazugehörigen Gummiriemen zur Kraftübertragung und Drehmoment/Drehzahl Beeinflussung genutzt. Dabei erfolgt die Berechnung der Übersetzung ausschließlich über den Radius. Beide Riemenscheiben, die über einen Gummiriemen verbunden sind drehen sich in dieselbe Richtung. Das bedeutet, dass die Drehrichtung durch dieses Getriebe unverändert bleibt. Ist jedoch eine Richtungsänderung erwünscht, kann dies über eine Überkreuzung des Riemens

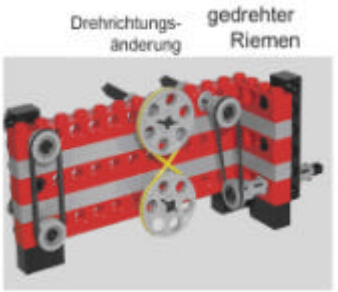

Abbildung 4: Riemenscheiben

[I] Übersetzung:
Eine Übersetzung im Sinne(in der Notation/in der Sprechweise??) der Kraftfahrzeugtechnik gibt das Verhältnis der Radien oder Zahnzahlen eines Zahnradpaares in einem Getriebe an. In diesem Zusammenhang spricht man auch von Untersetzung, wenn z.B. die Drehzahl heruntergesetzt wird. Aufgrund der Gesetze der Mechanik wird damit gleichzeitig eine Erhöhung des Drehmoments bewirkt. In den oben beschriebenen Formeln eingefügt ergibt sich:
$M_{out} = M_{in} * i$
mit: M_{out}: Drehmoment [Ausgang] in Nm
M_{in}: Drehmoment [Eingang] in Nm
i: Übersetzungsverhältnis [ohne Einheit]

[II] vgl. Martin, Fred „Robotic Explorations “, 2001 pp. 161-162

erreicht werden. Auch Drehungen in andere Bewegungsrichtungen sind, wie in Abbildung 2, ersichtlich möglich.

Schneckenrad[I]

Abbildung 5

Das Schneckenrad ist, aufgrund seiner Charakteristika, ein einzigartiges LEGO® -Bauteil. Es wird auf eine Achse aufgesteckt und kann ein Zahnrad antreiben. Dabei resultiert jede volle Umdrehung der Schnecke einer Vorwärtsbewegung eines Zahns. Somit ergibt sich eine n:1-Untersetzung. Eine weitere außergewöhnliche Eigenschaft ist die Unmöglichkeit der Bewegung der Schnecke über das verbundene Zahnrad. Das heißt, nur die Schnecke kann das Zahnrad antreiben, aber nicht vice versa. Obwohl bei jedem Einsatz einer Schnecke immer mit hohen Reibungsverlusten zu rechnen ist, zeichnen diese Eigenschaften das Bauteil für bestimmte Aufgaben[II] aus.

Differenzial[III]

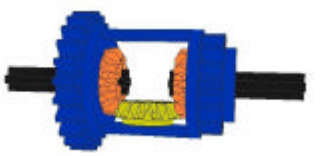

Abbildung 6

Als Differenzial wird ein Planetengetriebe bezeichnet, welches den Antrieb von zwei Wellen durch eine Antriebswelle mit gleichem Drehmoment gestattet. Allerdings kann die jeweilige Drehgeschwindigkeit variieren. Üblicherweise wird ein Differential zur Kraftübertragung genutzt, um, beispielsweise im Falle eines Automobils, Unterschiede von kurveninnerem und kurvenäußerem Rad auszugleichen. Zu diesem Zweck, der auch im Roboterbau äußerst nützlich sein kann, wird der Rahmen (blau) angetrieben. Sobald der Rahmen sich dreht, wird das am Rahmen gelagerte Kegelrad (gelb) die beiden mit ihm verbundenen Kegelräder (orange) mitnehmen. Diese drehen sich also im Idealfall mit derselben Geschwindigkeit. Da bei einer Kurvenfahrt das äußere Rad sich relativ zum Inneren schneller dreht, rotiert nun auch das mittlere Kegelrad und schafft somit den Ausgleich. Hierbei gibt der Rahmen die mittlere Drehgeschwindigkeit vor, von der kurvenäußeres und kurveninneres Rad um den gleichen Betrag abweichend schneller bzw. langsamer drehen.

[I] vgl.. Baum, Dave „Definitive Guide to LEGO® Mindstorms – Second Edition“,2003 pp. 165-166
[II] Beispielsweise ein Greifarm, der sich nicht selbstständig, sondern nur über den Motor geöffnet bzw. geschlossen werden darf.
[III] vgl. http://www.theoinf.tu-ilmenau.de/~nuetzel/mindstorms/asr2002/

3.3 Mindstorms™-Bauteile

3.3.1 RCX

Das Herzstück des Mindstorms™ Sets ist der Robotic Command Explorer (RCX), ein Legobaustein, mit integriertem Computersystem auf Basis eines Hitachi H8 Series Microcontrollers. Eine 8-bit CPU steuert den Gesamtablauf des RCX. Der RCX enthält einen 16KB ROM Speicher, sowie 32KB statischen RAM. Im ROM ist ein Treiber installiert, der beim Anschalten des RCX automatisch geladen wird. Der größte Teil des RAM Speichers wird von der Firmware und verschiedenen Systemparametern benutzt. 6 KB werden für den Benutzer reserviert. Dort kann der Benutzer selbstgeschriebenen Programme speichern.

Abbildung 7: RCX 1.0

Wie in Abbildung 7 ersichtlich, gibt es insgesamt 6 Steckplätze, von denen die grau unterlegten, bzw. die mit 1-3 nummerierten, für die Sensoren und die schwarz unterlegten beziehungsweise die mit A-C markierten, für die Motoren vorgesehen sind.

Ein LCD und vier Knöpfe erlauben eine direkte Interaktion.

Der RCX wird von 6 AA Batterien mit Strom versorgt. Eine ausreichende Ladung der Batterien sollte gewährleistet sein um die erforderliche Motorleistung, insbesondere unter Last, zu erhalten.

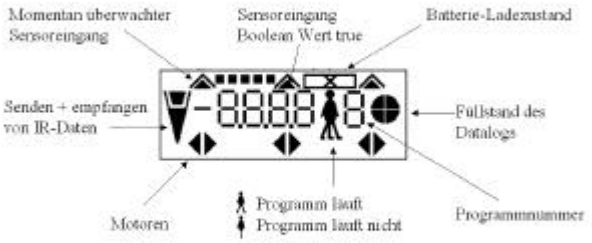

Abbildung 8: LC-Display

3.3.2 Sensoren

Insgesamt werden von LEGO® vier verschiedene Standard Sensoren zur Verfügung gestellt, die in aktive und passive beziehungsweise stromkonsumierende und Energie unabhängige Sensoren eingeteilt werden können. Sie sollen im folgenden näher betrachtet und ihre Funktionsweise erläutert werden.

Zu den passiven Sensoren gehören der Tastsensor und der Temperatursensor.

Tastsensor

Tastsensoren werden dazu benutzt, Hindernisse in der Umgebung zu erfassen. Die relativ kleine Berührungsfläche kann durch verschiedene Konstruktionen vergrößert werden.

Abbildung 9

Wie funktioniert ein Tastsensor?

Der Tastsensor ist der einfachste Sensor. Bei Nichtbetätigung ist der Widerstand, wie bei einem Schalter, unendlich groß. Bei Betätigung hingegen ist der Widerstand ca. 500Ω groß. Durch die veränderten Widerstandswerte kann der RCX feststellen, ob ein Gegenstand berührt wurde oder nicht.

Temperatursensor

Der Temperatursensor kann Temperaturen von –20 bis +50 Celsius erfassen.

Abbildung 10

Wie funktioniert ein Temperatursensor?

Die Funktionsweise des Temperatursensors ist ähnlich der des Tastsensors. In diesem Sensor ist ein temperaturveränderlicher Widerstand eingebaut durch den der RCX auf gemessene Temperaturwerte schließen kann.

Zu den aktiven Sensoren gehören der Lichtsensor und der Rotationssensor.

Lichtsensor

Der Lichtsensor erfasst Helligkeitswerte von 0,6 bis 760 Lux. Dadurch kann der RCX auf Lichtsignale oder sogar auf Licht im Infrarot-Bereich reagieren. Durch eine eingebaute Leuchtdiode ist es möglich nicht nur externe Helligkeiten und Lichtquellen zu erfassen, sondern auch das Vorhandensein von Gegenständen anhand des von ihnen reflektierten Lichtes zu erkennen.

Abbildung 11

Wie funktioniert ein Lichtsensor?

Da beim Lichtsensor das Auslesen der Werte bei gleichzeitiger dauerhafter Energieversorgung gewährleistet sein muss, wird alle 3 ms die Betriebsspannung für 0,1 ms abgenommen. In diesem Intervall misst der RCX den angelegten Spannungswert. Dadurch wird eine weitere Verkabelung vermieden.

Rotationssensor

Der Rotationssensor misst Umdrehungen einer Achse. Für diese Achse

Abbildung 12

ist eine Aussparung in seiner Mitte angebracht. Der Rotationsmesser zählt in 22,5 Grad Schritten. Sollte eine genauere Zählweise benötigt werden, kann dies durch Übersetzung mit einem entsprechendem Getriebe erreicht werden.. Die Zählweise von 22,5 Grad bedeutet, dass 16 Schritte einer vollen Umdrehung entsprechen. Der RCX ist in der Lage Schritte von –32.768 bis +32.767 zu zählen. Dabei werden Vorwärtsdrehungen addiert und Rückwärtsdrehungen abgezogen.

Wie funktioniert ein Rotationssensor?

Nach jeder vollen Umdrehung hat sich die Ausgangsspannung des Rotationssensors 16 mal in einer immer wiederkehrenden Folge von vier Werten geändert. Eine Umdrehung wird also durch eine wie in Abbildung 13 dargestellte Folge beschrieben. Anhand der Reihenfolge der unterschiedlichen Spannungswerte kann der RCX auf die Drehrichtung schließen.

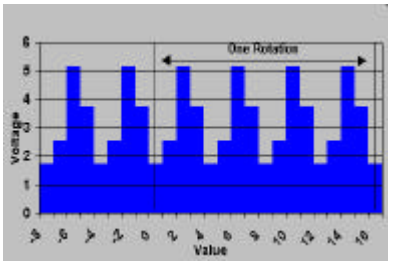

Abbildung 13

Sensoreneingänge

Der RCX verfügt über drei Sensoreneingänge, an denen jeweils ein oder mehrere Sensoren zusammen angeschlossen werden können[I]. Hinter jedem Sensoreingang steht ein A/D Wandler, welcher alle 3ms die Sensorwerte am Sensoreingang abfragt. Eine Ausnahme stellt der Rotationssensor dar, da bei diesem die Notwendigkeit einer permanenten Überwachung besteht. An den Sensoreingängen werden Spannungen zwischen 0-5V gemessen, die intern als raw value in einem Bereich von 0-1023 gespeichert werden. Die Software verarbeitet dann diese raw values in Abhängigkeit von der Sensor-Konfiguration zu boolean values und processed values.

Der boolean value bildet raw values auf binäre Werte ab, die für einfache Entscheidungen verwendet werden können. Zum Beispiel ist für die Benutzung eines Tastsensors die Verwendung des boolean values ausreichend, da auf den Sensor entweder Druck ausgeübt wird oder nicht. Wie der raw value in einen boolean value umgerechnet wird, bestimmt ein, durch den Softwareuser selbst konfigurierbarer, slope parameter.

Beim processed value hingegen können die Wertebereiche, je nach eingestelltem Sensormodus, unterschiedlich aussehen. Das processed value kann ein boolean, percentage oder die Anzahl Umdrehungen sein.

Um nun die erforderliche Konfiguration eines Sensors vornehmen zu können, muss ein Sensortyp und ein Sensormodus eingestellt werden:

[I] vgl. http://www.smallrobots.de/ws2000/dackel/sensoren2.htm

Sensortyp

Der Sensortyp legt fest auf welche Art und Weise der RCX mit dem Sensor umgehen soll. Nur auf diese Art kann der RCX feststellen, um welchen Sensor es sich handelt. Außerdem bestimmt der Sensortyp den Defaultmodus, in dem der Sensor arbeitet.

Es gibt vier verschiedene Sensortypen, die mit den Standard LEGO Sensoren korrespondieren: „Touch“, „Temperature“, „Light“ und „Rotation“. Es gibt noch einen fünften Typ, „None“, der als Standard der Sensoreneingänge immer voreingestellt ist. Er kann zum Lesen von raw values passiver Sensoren benutzt werden, sollte aber im Normalfall immer auf den, dem Sensor entsprechenden geändert werden.

SENSOR TYPE	CODE	CLASS	DEFAULT MODE	NQC NAME
None	0	Passive	Raw	SENSOR_TYPE_NONE
Touch	1	Passive	Boolean	SENSOR_TYPE_TOUCH
Temperature	2	Passive	Celsius	SENSOR_TYPE_TEMPERATURE
Light	3	Active	Percentage	SENSOR_TYPE_LIGHT
Rotation	4	Active	Rotation	SENSOR_TYPE_ROTATION

Aus: extreme mindstorms S.20

Slope Parameter und Sensor-Modus

Der Sensor Modus legt die Interpretation des raw values fest.

Er bestimmt welche Art von Messung vorgenommen wird und legt den slope Parameter des Sensors, durch den die Umrechnung eines raw values in ein boolean value festgelegt wird, fest. Der slope Parameter ist eine ganze Zahl zwischen 0 und 31. Diese Zahl wird einfach an den Code des Modus angehängt.

SENSOR MODE	CODE	NQC NAME
Raw	0x00	SENSOR_MODE_RAW
Boolean	0x20	SENSOR_MODE_BOOLEAN
Edge Count	0x40	SENSOR_MODE_EDGE
Pulse Count	0x60	SENSOR_MODE_PULSE
Percentage	0x80	SENSOR_MODE_PERCENT
Celsius	0xa0	SENSOR_MODE_CELSIUS
Fahrenheit	0xc0	SENSOR_MODE_FAHRENHEIT
Rotation	0xe0	SENSOR_MODE_ROTATION

Aus: extreme mindstorms S.23

Manche Programmiersprachen, wie zum Beispiel der RCX Code, bestimmen automatisch den Modus nach dem Sensortyp. Andere Sprachen, wie zum Beispiel NQC, ermöglichen jede mögliche Kombination von Sensortyp und Modus, wobei zu beachten ist, dass nicht jede Kombination sinnvoll ist.

Boolean Values

Besitzt der zur Umrechnung erforderliche slope Parameter den Wert null, ergibt sich folgende Transformation:

CONDITION	BOOLEAN VALUE
raw > 562 (ca. 55%)	0
raw < 460 (ca. 45%)	1
460 ≤ raw ≤ 562	Unchanged

Aus: extreme mindstorms S.21

Besitzt der slope Parameter einen Wert ungleich null, werden die raw values wie folgt umgerechnet. Der aktuell gelesene Wert wird mit dem davor gelesenen Wert und die resultierende Differenz mit dem eingestellten slope Parameter verglichen. Ist die Differenz beispielsweise höher als der Wert des slope Parameters, wird ein boolean value von null zurückgegeben. Durch den slope Parameter ist es möglich, langsame Veränderungen zu tolerieren und den Cutoff Punkt, die Grenze an der sich der boolean value ändert, dynamisch anzupassen.

CONDITION	BOOLEAN VALUE
Change > slope	0
Change < -slope	1
Current > (1023 – slope)	0
Current < slope	1

Change = current value minus previous value
Slope = the value of the slope parameter
Current = current raw value

Aus: extreme mindstorms S.21

Processed Value

Es gibt acht verschiedene Modi, die bestimmen in welches processed value der raw value umgerechnet werden soll.

Raw Modus

Der raw Modus ist der einfachste Modus. Der processed value des Sensors ist identisch mit dem raw value (integer zwischen 0 und 1023)

Boolean Modus

In diesem Modus ist der processed value identisch mit dem boolean value. Dies ist der Defaultwert eines Tastsensors.

Edge Count Modus

In dem Edge Count Modus wird gezählt, wie oft sich der boolean value gerändert hat. Dabei werden Änderungen von 1 nach 0 und von 0 nach 1 berücksichtigt. Es wird mit 0 angefangen zu zählen. Zu beachten ist, dass Änderungen, die keine 300ms auseinander liegen, nicht erfasst werden.

Pulse Count Modus

Der Pulse Count Modus zählt ebenfalls die Veränderungen des boolean values. Allerdings inkrementiert er nur bei einer Veränderung von 1 nach 0. Dieser Modus ist sehr geeignet, um die Anzahl der Betätigungen des Berührungssensor zu erfassen.

Percentage Modus

Im Percentage Modus wird das raw value auf den Bereich von 0 bis 100 skaliert. Dies ist der Defaultwert eines Lichtsensors.

Rotation Modus

Dieser Modus dekodiert den Output eines Rotationssensors. Dies ist der Defaultwert eines Rotationssensors. Für andere Sensoren ist dieser Modus nicht sinnvoll.

Celsius Modus

Dieser Modus rechnet ein raw value in eine Temperatur um. Dies ist der Defaultwert eines Temperatursensors. Für andere Sensoren ist dieser Modus nicht sinnvoll.

Fahrenheit Modus

Intern rechnet der RCX mit Celsius. Wenn der Fahrenheit Modus gewählt ist, rechnet er den Celsiuswert in Fahrenheit um.

Mehrere Sensoren an einen Eingang

Meistens werden mehr Sensoren benötigt, als Steckplätze auf dem RCX zu Verfügung stehen. Wenn mehrere Sensoren an einem Eingang angeschlossen werden, muss sich der Benutzer darüber im klaren sein, wie die Sensoren miteinander arbeiten. Steckt man die Sensoren einfach aufeinander an einem Eingang, werden sie mit einem OR verknüpft. Schaltet man die Sensoren in Reihe,

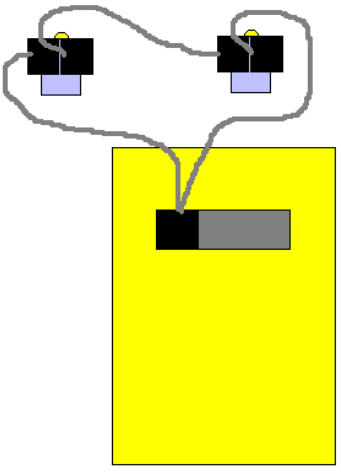

Abbildung 14

erhält man ein AND. Soll jedoch bestimmt werden, welcher der Sensoren ein Signal ausgelöst hat, muss ein künstlicher Widerstand eingebaut werden. Dafür können zwei kleine LEGO® Lighting Bricks verwendet werden. Wird der Sensor ohne Lämpchen betätigt, liest der RCX ein raw value, das kleiner ist als der raw value des Sensors mit Lämpchen[I].

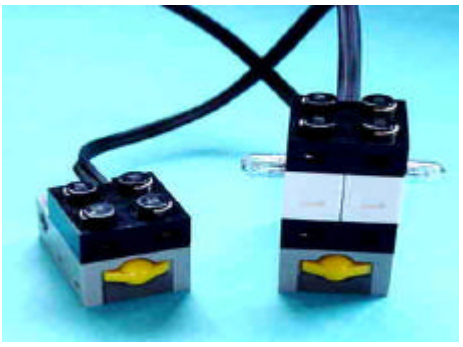

Abbildung 15

Abgesehen von den Standard LEGO Sensoren können, mit einem passenden Adapter, auch die verschiedensten Sensoren (Lautstärke, Spannung, Feuchtigkeit...) an einen RCX angeschlossen werden.[I]

3.4 Software

3.4.1 Architektur

Die Softwarearchitektur und somit die logische Struktur des RCX lässt sich durch ein Vier-Schichten Modell veranschaulichen.

Auf der untersten Ebene befindet sich, wie bei jeder Rechnerarchitektur, die Hardware. Dazu gehören neben vielen anderen Elektronikbauteilen vor allem der im Abschnitt Hardware bereits angesprochene, 8-bit Hitachi H8 Prozessor. Auf dieser Ebene führt der Prozessor den Maschinecode aus und einige zusätzliche Hardwarekomponenten sorgen für die Umwandlung von Sensorsignalen in digitale Daten.

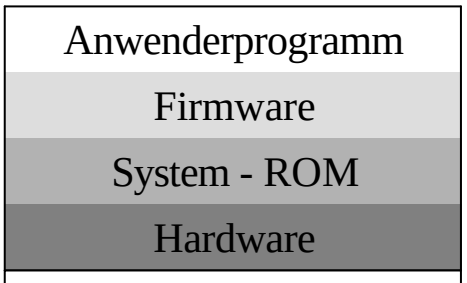

Abbildung 16

Über der Hardwareebene setzt die System ROM Schicht auf, in welcher Routinen zu finden sind, die alle grundlegenden Funktionalitäten zum Umgang mit dem RCX bereitstellen. Dazu gehören zum Beispiel die Displayanzeige, die Infrarot Kommunikation sowie die

[I] vgl. http://www.smallrobots.de/ws2000/dackel/sensoren2.htm

Initialisierung der Peripherie. Der nicht änderbare und von Energiezufuhr unabhängige ROM Code ist unter anderem auch dafür verantwortlich das es überhaupt möglich ist ein Betriebssystem, im folgenden auch Firmware genannt, in den RAM laden zu können.

Diese Firmware, welche zusammen mit den festverdrahteten ROM Routinen, eine Art Betriebssystem für den RCX darstellt, bildet die nächste Schicht des Modells. Es gibt mehrere Alternativen bei der Wahl der Firmware, auf die im nächsten Abschnitt noch näher eingegangen wird. Die Firmware interpretiert den Bytecode, konvertiert diesen zu Maschinencode und führt Standard System Operationen unter Zuhilfenahme der ROM Routinen aus.

Auf oberster Ebene befinden sich die Anwendungsprogramme. Diese werden zunächst auf einem Hostrechner geschrieben und kompiliert. Erst dann wird der Bytecode mittels Infrarotübertragung an den RCX gesendet und dort gemäß erläuterter Abfolge und nachfolgender Abbildung interpretiert und ausgeführt. Die einzige Ausnahme stellen Forth Anwendungsprogramme dar, auf die aber an entsprechender Stelle aber noch näher eingegangen wird.

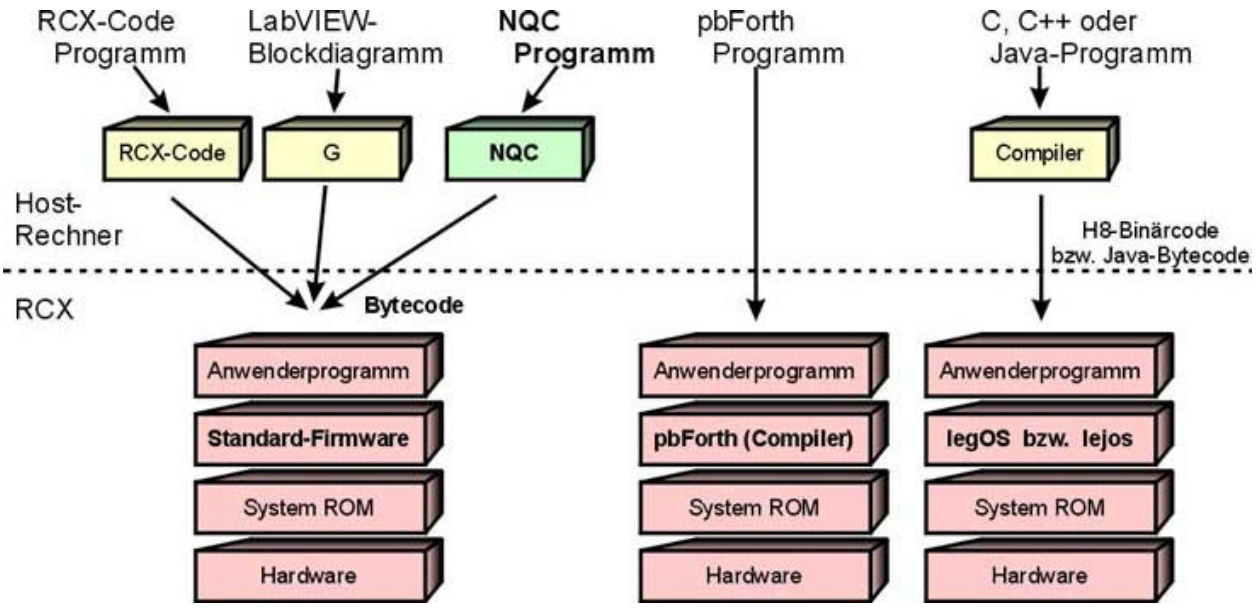

Abbildung 17

[I] Die Firma DCP Microdevelopments vertreibt einen solchen Adapter

3.4.2 Betriebssysteme und Sprachen

Durch die Speicherung der Firmware im RAM kann sie ohne weiteres ersetzt werden. Bei der Wahl der Firmware ist darauf zu achten, das die Anzahl und Art der verwendbaren Programmiersprachen sowie der Grad der Ausschöpfung des Leistungspotentials des RCX stark von der eingesetzten Firmware abhängen. Die folgende Abbildung zeigt die Alternativen in der Wahl der Firmware sowie die jeweils zugehörigen Programmiersprachen.

Firmware	LEGO® Standard	LeJOS	LegOS	Pb Forth
Sprache	RCX Code/ NQC	Java	C/C++	Forth

LEGO® Standard Firmware

Verbleibt man zunächst bei de im RIS enthaltenen LEGO® Standard Firmware kann mit der ebenfalls im Paket enthaltenen RCX Code generierenden Programmierumgebung gearbeitet werden. Hierbei handelt es sich um eine grafische, kindgerechte Programmierumgebung welche schon vorgefertigte Programmteile bereitstellt und sich sehr einfach und anschaulich bedienen lässt. Bei größeren Programmen macht sich allerdings schnell der eingeschränkte Funktionsumfang sowie ein Mangel an Übersichtlichkeit bemerkbar. Eine weiterer Nachteil ist die ausschließliche Nutzbarkeit unter Windows.

Eine hier zusätzliche erwähnenswerte Sprache ist das von Dave Baum entwickelte Not Quite C[I]. Diese C-ähnliche Sprache ist sehr weit verbreitet, unter anderem aufgrund ihrer Plattformunabhängigkeit sowie durch den vorhandenen Bekanntheitsgrad der Basissprache C. Sie verbindet den Vorteil der Stabilität und Zuverlässigkeit der LEGO® Standard Firmware mit einer, im Vergleich zum RCX Code, optimaleren Nutzung der Ressourcen des RCX. Dies kommt durch den in NQC vorhandenen zusätzlichen Funktionsumfang, bezüglich Sensormodi, Datalog[II] und Nutzung von Arrays zum Ausdruck.

Aber auch NQC unterliegt noch sehr vielen Einschränkungen, wie zum Beispiel einer sehr begrenzten Variablen- und Datentypenanzahl[III], die durch die Nutzung der LEGO® Firmware als Betriebssystem zustande kommen. Ein weiterer Punkt, der gegen die Nutzung der LEGO® Firmware als Betriebssystem spricht ist die Tatsache, dass es sich nicht um open source code handelt und somit die Weiterentwicklung ausschließlich vom Lizenzinhaber abhängt.

[I] http://www.baumfamily.org/nqc
[II] Funktion zum Speichern und Übertragen einer Datenfolge vom RCX zum Host
[III] Datentypen nur int und boolean, max. Variablenanzahl 48

LeJOS

Um Java als Programmiersprache für den RCX nutzbar zu machen, muss das von Jose Solorzano entwickelte LeJOS[I] aufgespielt werden. Es beinhaltet eine ca.16 KB große Mini VM und eine Standard Java API in reduziertem Umfang. Elf der 124 in J2SE enthaltenen Packages sind verfügbar, unter anderem die java.lang, java.util, java.io und java.net Packages teilweise mit eingeschränkter Anzahl an zur Verfügung stehenden Klassen und Exceptions. Nicht enthalten sind zum Beispiel Funktionen wie Garbage Collection, das switch statement oder eine Arithmetik für long Variablen[II]. Zusätzlich stehen sowohl eine spezielle API für den RCX und zur Roboterprogrammierung als auch Tools zum Kompilieren und Übertragen von Programmen zur Verfügung. Des Weiteren bietet das Java-fähige Betriebssystem als Einziges die Möglichkeit Fließkomma-Arithmetik und trigonometrische Funktionen auf dem RCX nutzbar zu machen. Zusammenfassend kann gesagt werden, dass mit LeJOS Java eine komplette, schnelle, effiziente und äußerst portable Sprache auf dem RCX nutzbar gemacht werden kann. Die Betreibung als open source Projekt sowie die weite Verbreitung der zugrunde liegenden Sprache sind als weitere Vorzüge ,nicht nur von LeJOS, sondern auch von LegOS anzusehen.

LegOS

Mit LegOS[III] wurde 1999 der erste Versuch unternommen die LEGO® Firmware zu ersetzten. Das Hauptziel bestand darin, alle Einschränkungen des RCX Interpreters durch ein direktes Ausführen des Codes auf dem Prozessor zu umgehen. Umgesetzt wird dies durch ein in C geschriebenes Programm, welches mit einer Reihe von Systemroutinen verlinkt und anschließend von LegOS kompiliert und anstelle der Firmware geladen wird[IV]. Diese Methode setzt aufgrund der Umgehung der Firmware sehr tief auf der Hardwareebene an und dadurch sehr viel Erfahrung des Programmierers voraus. Der dadurch erreichte Zuwachs an Speicherkapazität und Schnelligkeit bietet sehr viele Möglichkeiten das Potential der Hardware möglichst optimal auszuschöpfen.

Pb Forth

Das Betriebssystem programmable brick Forth[V] ist ebenfalls eine Möglichkeit zur vollständigen Ersetzung der LEGO® Standard Firmware. Als Sprache kommt hier die seit langem auf dem Gebiet der Robotik etablierte und für kleine Systeme sehr geeignete Sprache Forth zum Einsatz. Die Besonderheit dieser Sprache liegt in ihrer Interaktivität. Auf den RCX wird ein Kernel heruntergeladen mit dem man sich, mit Hilfe eines simplen, auf dem Host

[I] vgl. http://lejos.sourceforge.net
[II] vgl. http://www2.hs-harz.de/~fstolzenburg/lego/folien/praktikum.pdf
[III] vgl. http://www.noga.de/legOS
[IV] vgl. Dario Laverde, Jürgen Stuber, Guilio Ferrari, Programming LEGO® Mindstorms™ with Java
[V] vgl. http://www.hempeldesigngroup.com/lego/pbForth/homePage.html

befindlichen Terminalfensters, in Verbindung setzten kann. Programme müssen also nicht mehr kompiliert und heruntergeladen werden, sondern können direkt in den Interpreter eingegeben und ausgeführt werden[I]. Diese Methode bedarf nicht einmal eines kompletten Programms sondern ist ebenfalls auf einzelne Anweisungen anwendbar. Die Sprache an sich ist leicht zu erlernen, verwendet allerdings die etwas gewöhnungsbedürftige umgekehrte polnische Notation. Pb Forth stellt minimale Anforderungen an Art und Kapazitäten des Host Betriebssystems und ist wie die beiden zuvor genannten alternativen Firmwares für den nicht kommerziellen Gebrauch kostenfrei.

[I] Knudsen J., Noga M.: LEGO®Mindstorms™ Roboter

4 Grenzen, Probleme von LEGO® Mindstorms™ und deren Lösungsmöglichkeiten am Beispiel eines Raum-Kartier Roboters

4.1 Anforderungen an das Kartieren eines Raumes

4.1.1 Aufgabenstellung

Es soll ein Roboter konstruiert werden, der autonom einen Raum befahren und diesen währenddessen kartieren kann. Der erste Schritt bei der Erstellung einer solchen Karte ist es die umgebenden Wände zu vermessen und zu erfassen. Darüber hinaus müssen für eine vollständige Kartierung alle im Raum befindlichen Gegenstände erfasst und aufgezeichnet werden. Diese Aufgabe soll unabhängig von Startpunkt und Startausrichtung des Roboters erledigt werden. Der Roboter soll mit Hilfe des LEGO® Mindstorms™ Bausatzes konstruiert und gesteuert werden. Da der RCX jedoch keine umfangreichen Daten sammeln kann, steht ein Notebook zur Verfügung, welches die vom RCX per Infrarot-Schnittstelle gelieferten Daten sammeln, weiterverarbeiten und auf deren Grundlage eine zweidimensionale Karte erstellen soll.

4.1.2 Szenario

Die Aufgabe, auf alle Spezifika eines Raumes einzugehen, erscheint zu komplex, daher ist es sinnvoll ein von der Realität abstrahierendes Szenario zu definieren. In diesem modellhaftem Raum existieren nur gerade Wände, die im rechten Winkel zueinander stehen und Gegenstände, die in einer für den Roboter wahrnehmbaren Höhe vorliegen. Des Weiteren sollte der Boden eben beschaffen sein.

4.1.3 Anforderungen an einen Kartierungsroboter

Bei der Kartierung eines Raumes ergeben sich einige Problemfelder, die sich durch Anforderungen an einen korrekt funktionierenden Roboter konkretisieren lassen. Sie sind unterteilbar in Probleme der Navigation, der Konstruktion, der Bewegung und der Algorithmen beziehungsweise Software. All diese Problemfelder weisen eine hohe Interdependenz auf, dem zufolge erzeugen mögliche Lösungen für Teilprobleme häufig neue Probleme in anderen Bereichen.

Abhängig davon, ob nur die Außenwände oder der gesamte Raum kartiert werden soll, existieren teilweise unterschiedliche Anforderungen. Im Folgenden wird für jedes

Problemfeld zunächst auf die gemeinsamen und dann auf jeweils spezielle Anforderungen eingegangen.

Navigation[I]

Mit Hilfe der Navigation soll sich der Roboter autonom in einem ihm unbekannten Terrain bewegen und zur Kartierung die Koordinaten der Hindernisse erfassen können. Dabei muss von der Prämisse ausgegangen werden, dass der Roboter an einer ihm unbekannten Position, mit einer ihm unbekannten Ausrichtung und, aufgrund der Ausstattung mit Sensoren, eingeschränkten Orientierungspunkten startet.

Konstruktion

Im Bereich der Konstruktion kommen alle Teilbereiche zusammen und müssen zu einem funktionierenden Ganzen verschmolzen werden. Da es hier jedoch oft Restriktionen in Bezug auf die zur Verfügung stehenden Teile gibt, beziehungsweise auf die Beschränktheit der Anzahl und Art der Sensoren und möglicher konstruktionstechnischer Probleme bei der Anbringung derselben, muss bei der Entwicklung der Softwareumsetzung immer auch bedacht werden, ob die Lösung noch praktikabel beziehungsweise umsetzbar ist.

Des Weiteren sollten hier Anforderungen wie Nutzlast, Gesamtgewicht und Stabilität berücksichtigt werden und auf ihre Implikationen hinsichtlich Größe, Geschwindigkeit und Beweglichkeit des Roboters untersucht werden.

Die Art des Antriebs, Position und Art der Sensoren sowie Bau des Chassis müssen sicherstellen, dass die Anforderungen der verwendeten Problemlösungsstrategie erfüllt werden.

Bewegung

Die Software sollte sich darauf verlassen können, dass das angeforderte Bewegungsmuster exakt ausgeführt wird. Da Ungenauigkeiten aufgrund der verwendeten LEGO® Bauteile vorprogrammiert sind, müssen diese durch eine geschickte Konstruktion minimiert, mit Hilfe geeigneter Sensoren kontrolliert und durch die Software korrigiert werden. Ungenauigkeiten treten in Bezug auf eine Geradeausfahrt oder eine winkelgenaue Kurvenfahrt auf.

Algorithmen und Software

Der Bereich der Algorithmen und Software befasst sich mit der Strategie zum Kartieren eines Raumes, Hindernisumfahrung sowie Steuerung des Roboters. Da der Roboter autonom

[I] ausführlich Bagnall, Brain „Navigation in LEGO® Mindstorms™ Programming“, 2002

handeln soll, muss die Software aufgrund der Informationen der Sensorik des Roboters Entscheidungen fällen.

4.1.4 Problemlösungsstrategien

Navigation

Damit ein Roboter seinen Startpunkt ermitteln kann braucht er Bezugspunkte an denen er sich orientieren kann. Ähnlich wie die Schifffahrt per Sextant oder heutzutage per GPS eine Positionsbestimmung mit Hilfe von Fixpunkten vornimmt, kann dies der Roboter. Eine Kopplung eines mit dem Roboter verbundenen Notebooks oder Palms mit einem GPS ist denkbar. Allerdings ist damit aufgrund der Einschränkungen des GPS für den zivilen Bereich nur eine Genauigkeit von 10-100m möglich, was für das Raumkartieren nicht akzeptabel ist. Weiterhin kann durch zwei festinstallierte terrestrische Positionssender die Position des Roboters trianguliert werden. Da jedoch eine absolute Positionsbestimmung weniger wichtig für die Kartierung eines Raumes ist und diese Lösungsstrategie einen inakzeptablen Arbeits- und Materialaufwand bedeuten würde, wird diese Möglichkeit hier vernachlässigt.

Eine vielversprechende und ebenfalls für die Aufgabenstellung adäquate Lösungsmöglichkeit bietet die Berechnung der relativen Positionsänderungen in bezug zum Startpunkt. Einfachstes Beispiel, welches sich auch vollständig mit LEGO® Mindstorms™ Bauteilen realisieren ließe, ist die Anbringung von Rotationssensoren, die die aus der Achsendrehung resultierenden Bewegungen erfassen können. Eine weitere Möglichkeit zur Erfassung einer Bewegung ist die Verwendung einer Computermaus. Voraussetzung dafür ist allerdings ein mit dem RCX kommunizierendes Notebook oder anderes technisches Gerät.

Beim Wandvermessen dienen die Wände als Orientierung. Aufgrund der Daten des Rotationssensors kann die zurückgelegte Strecke und damit die aktuelle Position des Roboters errechnet werden.

Konstruktion

Da bei der Kartierung eines Raumes immer auch mit gewollten oder ungewollten Kollisionen des Roboters mit Hindernissen und Wänden umgegangen werden muss, sollte das Chassis und die Sensorbefestigungen dafür ausgelegt sein. Oftmals ist es auch erforderlich das „Sichtfeld“ der Sensoren durch eine geeignete Konstruktion zu vergrößern. Einige Anregungen, insbesondere zu Tastsensoren, sind in der, dem LEGO® Mindstorms™ Set beigefügten „Constructopedia“, enthalten. Abhängig von der Anordnung der Wände und der gewünschten

Genauigkeit der zu erstellenden Karte muss der Roboter eine dementsprechende Größe und Manövrierfähigkeit aufweisen.

Das Problem des „Steckenbleibens in Ecken“ bei der Wandvermessung lässt sich beispielsweise durch einen runden Roboter umgehen. Dieser müsste die Möglichkeit besitzen seine Fahrtrichtung beliebig ändern zu können. Nimmt man zusätzlich an, dass in dem Raum nur rechte Winkel vorhanden sind, bietet sich ein Roboter mit, wie im Kapitel „Realisierung“ beschriebenen Charakteristika an. Um ein Steckenbleiben während der Verfolgung der Wand zu verhindern ist es sinnvoll Laufschienen beziehungsweise Stoßstangen anzubringen.

Bewegung

Die Art des Antriebs legt die Bewegungsmuster eines Roboters fest, und ist daher für entstehende Fehler verantwortlich. In Hinblick auf die Vielzahl existierender Antriebsmöglichkeiten für einen LEGO®-Roboter seien hier nur eine kleine Auswahl vorgestellt und in Hinblick ihrer Eigenschaften untersucht.

Geradeausfahrt

Solange es Aufgabe des Roboters ist einer Wand zu folgen, kann er diese als ständigen Bezugspunkt benutzen und damit die Geradeausfahrt sicherstellen. Des Weiteren werden verschiedene Möglichkeiten aufgezeigt, die Wand als stetigen Kontrollmechanismus zu nutzen. Zum einen kann ein Tastsensor den ständigen Kontakt mit der Wand kontrollieren und bei etwaigen Abweichungen das Programm dazu veranlassen einen Wandwiederfindungsalgorithmus zu starten. Da diese Art des Kontakthaltens mit der Wand jedoch die Konstruktion stark beansprucht, kann es sinnvoll sein auf eine berührungsfreie Lösung zu setzen. Ein Lichtsensor kann, aufgrund des Betrages einer von der Wand reflektierten Lichtquelle, seine ungefähre Entfernung zur Wand bestimmen.

Obwohl die Wand beim vollständigen Kartieren des Raumes als ständiger Bezugspunkt fehlt, scheint dieses Problem prinzipiell einfach zu lösen zu sein. Ein einfacher Raupenantrieb mit je einem Motor an jeder Seite sollte in der Lage sein, den Roboter bei identischen eingestellter Motorleistung geradeaus fahren zu lassen. Jedoch variiert selbige von Motor zu Motor so stark, dass Spurabweichungen unausweichlich sind und die unterschiedliche Drehzahl der Achsen durch weitere Sensoren[I] und softwaretechnische Korrekturmechanismen ausgeglichen werden müsste. Abhilfe kann eine Art Autoantrieb schaffen bei welchem nur ein Motor für die Vorwärts- bzw. Rückwärtsbewegung sorgt und ein weiterer die Steuerung übernimmt. Schwachpunkt dieser Fortbewegungsart ist die Ungenauigkeit der Lenkung. Baut man jedoch einen Roboter ohne bzw. mit eingeschränkter Lenkmöglichkeit, der beispielsweise durch

[I] Bspw.: Anbringung von zwei Rotationssensoren zur softwaretechnischen Korrektur der Drehzahlunterschiede

Ausfahren eines zweiten Fahrgestells 90° Kurven fahren kann, ist die Geradeausfahrt nur noch von der Verformbarkeit der verwendeten LEGO®-Teile abhängig.

Winkelgenaue Kurvenfahrt

Die einfachste Lösung, für das Fahren einer winkelgenaue Kurve, ist für eine bestimmte Zeit, welche auf Erfahrungswerten basiert, die entsprechenden Motoren zu aktivieren bzw. deaktivieren bis die gewünschte Drehung erfolgt ist. Da diese Zeit aufgrund des Untergrundes, des Ladezustandes der Batterien und möglichen Hindernissen variiert, ist eine verlässliche Drehung des Roboters so nicht möglich. Abhilfe könnten hier wiederum zwei an den Achsen angebrachte Rotationssensoren schaffen, mit denen die Software die Drehung des Roboters kontrollieren kann. Berücksichtigend, dass es nur 3 Sensorsteckplätze gibt, ist diese Option sehr genau in Hinblick auf die Notwendigkeit von anderen Sensoren zu prüfen. Steht wie in unserem Fall ein Notebook oder Palm[I] zur Verfügung, ist es auch denkbar dort anschließbare Geräte, wie zum Beispiel einen Kompass, für die Kontrolle der Kurvenfahrt einzusetzen. Unter der Vorraussetzung einer entsprechenden softwaretechnischen Anbindung ist damit eine winkelgenaue Kurvenfahrt möglich.

Algorithmen und Software

Algorithmen sind sehr stark aufgabenspezifisch. Im Weiteren soll, jeweils für die Wandvermessung als auch für die Raumkartierung, ein Algorithmus kurz skizziert werden. Einzelne Implementierungen werden hier nicht vorgestellt.

Es wird von einem rechteckigen Roboter ausgegangen, der seine Fahrtrichtung um 90 Grad ändern kann ohne sich dabei drehen zu müssen. Das Chassis des Roboters ist so beschaffen, dass sich der Roboter, bei schrägem Auftreffen auf eine Wand, an der Wand ausrichtet. An jeder Seite ist je ein Tastsensor angebracht, der Berührungen mit der Wand über die vollständige Breite, beziehungsweise Länge des Roboters erfasst. Ein Rotationssensor ermöglicht die erforderliche Streckenmessung.

Ist zu Beginn keiner der Tastsensoren betätigt, befindet sich der Roboter in der Mitte des Raumes. Er muss sich eine geeignete Startposition in einer Ecke des Raumes suchen, um mit dem Wandfolgungsalgorithmus starten zu können. Er fährt geradeaus bis ein Tastsensor betätigt wird. Dann folgt er der Wand bis ein zweiter Tastsensor betätigt wird. Nun kann von einer Eckposition des Roboters ausgegangen werden.

Der Wandfolgungsalgorithmus startet von dieser Position aus. Der Roboter folgt solange einer Wand, bis entweder der in Fahrtrichtung befestigte Tastsensor gedrückt wird, oder der durch die Wand gedrückte Tastsensor freigegeben wird. In beiden Fällen errechnet der Roboter die

[I] Bspw.: Palm Navigator von Precision Navigation

Strecke, die er zurückgelegt hat und trägt die Daten ins Datalog ein. Danach entscheidet er aufgrund der Kombination und der Reihenfolge der zuletzt eingetretenen Ereignisse in welche Richtung er weiterfahren muss. Werden beide Tastsensoren gedrückt, fährt er von der Wand weg, der er zuvor gefolgt war, entlang der gerade erfassten Wand. der Algorithmus beginnt wieder von vorne. Wird der Tastsensor, der zuvor durch die Wand betätigt wurde freigegeben, fährt der Roboter in Richtung der zuvor gefolgten Wand. Der Tastsensor erfasst eine neue Wand. Der Algorithmus wird in einer immer wiederkehrenden Schleife durchlaufen, bis der Roboter den Ausgangspunkt wieder erreicht hat. Diesen kann er aufgrund seiner gesammelten Daten als solchen identifizieren.

Zur Raumkartierung wird der Raum gedanklich in „Streifen“ eingeteilt, deren Breite der Breite des Roboters entspricht. Im Raumkartierungsalgorithmus fährt der Roboter jeden dieser Streifen ab und erfasst dabei im Raum befindliche Gegenstände.

Der Roboter sucht sich, mit Hilfe des oben beschriebenen Algorithmus, eine geeignete Startposition. Von dort aus fährt er entlang der Wand, bis der Sensor, der in Fahrtrichtung angebracht ist, betätigt wird. Der Roboter bringt sich in Position, um den nächsten „Streifen“ abzufahren. Dazu fährt der Roboter parallel zu der zuvor erfassten Wand, in Richtung eines neuen Streifens. Von dort aus fährt er gerade durch den Raum, bis der Tastsensor erneut betätigt wird. Der Algorithmus wird solange durchlaufen, bis der Roboter am „Ende des Raumes“ ankommt. Das Ende des Raumes wurde vorher, aufgrund der Daten des Wendevermessens bestimmt. Jeweils nach Erfassen eines Objektes wird die Position des Zusammenstoßes mit dem Objekt errechnet und die Daten in den Datalog gespeichert.

Zu beachten ist, dass kleinere Hindernisse hinter Hindernissen nicht erfasst werden. Um das Problem zu minimieren kann der Roboter senkrecht zu den bereits abgefahrenen Streifen den Raum erneut durchfahren. Es kann allerdings immer ein Raum konstruiert werden, in dem mit diesem Algorithmus nicht alle Hindernisse erfasst werden.

4.2 Realisierung

Aufgrund der Vielzahl der Problemfelder und dem eigentlichen Fokus dieser Seminararbeit wurde darauf verzichtet einen Raum-Kartierungsroboter zu konstruieren, der sich sicher in vollständig unbekanntem Terrain bewegt. Im folgenden soll aber eine mögliche Realisierung eines Roboters vorgestellt werden, der in der Lage ist einen Raum zu vermessen. Als Prämisse wurde angenommen, dass der Raum nur aus rechten Winkeln besteht und einen ebenen Untergrund besitzt. Die Startposition des Roboters kann beliebig gewählt werden.

Der Antrieb des Roboters besteht aus 8 nicht drehbaren Rädern, von denen jeweils vier zu einer unabhängigen Antriebseinheit gehören. Beide Antriebseinheiten sind orthogonal

zueinander angeordnet. Um Interferenzen der Antriebseinheiten auszuschalten und beide zu trennen, wurde ein "Lift" eingebaut, der es dem Roboter ermöglicht einen Antrieb per Lift nach Richtung Boden zu fahren und dadurch den anderen aufzubocken. Somit haben immer nur Räder eines Antriebes Bodenkontakt. Diese Art der Konstruktion stellt eine Geradeausfahrt mit tolerierbaren Abweichungen zur Verfügung. Um Entfernungen zu messen ist ein von diesem Lift unabhängiger Rotationssensor eingebaut worden. Tastsensoren an allen vier Seiten des Roboters, von denen sich je zwei gegenüberliegende einen Steckplatz auf dem RCX teilen, liefern Informationen über die Umgebung. Solange nun einer dieser Sensoren gedrückt ist, bedeutet dies, dass sich der Roboter an einer Wand befindet und weiter geradeaus fahren muss. Wird ein weiterer Tastsensor gedrückt ist der Roboter in einer Ecke angelangt und wird dann auf den anderen Antrieb umschalten. Somit ist es dem Roboter möglich einen viereckigen Raum mit Hilfe eines Wandverfolgungsalgorithmus zu vermessen.

5 Schlussbetrachtung

Durch die Betrachtung der hardware- und softwaretechnischen Komponenten wurde eine Vielzahl von Möglichkeiten aufgezeigt Roboter zu realisieren und implizit, zum Beispiel anhand der vorhandenen Bauteile, verdeutlicht mit welchen Arten von Aufgaben diese aufgrund der vorhandenen Möglichkeiten betraut werden können.

Durch das Beispiel des Raum-Kartier-Roboters sollte klar geworden sein das teilweise eine sehr starke Divergenz zwischen Ideen und theoretischen Überlegungen auf der einen und dem Versuch der praktischen Umsetzung auf der anderen Seite besteht.

Einige der im Rahmen dieses Seminarthemas gestellten „Unteraufgaben“ wie z.B. der Bau eines Roboters zum Auffinden einer Lichtquelle, Verfolgen einer Line oder Hindernisumfahrung bei Zufahrt auf ein Ziel, waren ohne Weiteres mit den LEGO® Robotern umsetzbar. Obwohl diese Themengebiete keinen Eingang in diese Arbeit fanden, können sie veranschaulichen, wie LEGO®-Roboter eng abgesteckte Aufgabenbereiche autonom zu erledigen in der Lage sind. Um jedoch komplexere Aufgaben wie zum Beispiel das Kartieren eines Raumes mit Hilfe von LEGO® Mindstorms™ lösen zu können, bedurfte es sehr vieler vereinfachender Annahmen und Anforderungen an den zu kartierenden Raum. Da sich diese Einschränkungen für eine praktische Aufgabenstellung, wie zum Beispiel Minensuche, jedoch nicht durchsetzen lassen oder deren Durchsetzung nicht sinnvoll beziehungsweise nicht erwünscht ist, kann kaum von einem praktischen Nutzen bei der Bewältigung komplexerer Aufgaben mit LEGO® Mindstorms™ die Rede sein. Dem LEGO® System fehlen zu diesem Zweck erhebliche Grundvoraussetzungen bezüglich Stabilität und Genauigkeit der Technic™- sowie der Mindstorms™-Bauteile.

Außerdem bleibt zu sagen das, obwohl LEGO® Mindstorms™ ursprünglich als Spielzeug mit einer Zielgruppe von Kindern über 12 Jahren auf den Markt gebracht wurde, erfreut es sich heute auch bei Erwachsenen großer Beliebtheit. Es hat sich ein hoher pädagogischer Wert herausgestellt, da es geeignet ist Grundkonzepte der Informatik im Allgemeinen und der Programmierung im Besonderen spielerisch zu vermitteln. Weiterhin werden beim Bauen von Robotern mit LEGO® Mindstorms™ automatisch Thematiken mobiler Systeme und Cross Compilation näher gebracht.

Abschließend kann aus den Erfahrungen, die bei der Erstellung dieser Arbeit gesammelt worden sind, abgeleitet werden, dass mit dem „Spielzeug“ LEGO® Mindstorms™ bemerkenswert weit entwickelte Roboter gebaut werden können, diese jedoch trotz allem, aufgrund der genannten Restriktionen nur eingeschränkt in realitätsnahen Szenarien komplexere Aufgaben bewältigen können.

A Anhang

A.a Literaturverzeichnis

Knudsen, Jonathan B., Noga Markus L. "LEGO®Mindstorms™ Roboter", 2000

Hystad, Dean „Building LEGO® Robots For FIRST™ LEGO® League", 2002

Martin, Fred „Robotic Explorations", 2001

Baum, Dave „Definitive Guide to LEGO® Mindstorms – Second Edition", 2003

Baum, Dave, Gaspari, Hempel, Villa „Extreme MINDSTORMS: An Advanced Guide to LEGO® MINDSTORMS™", 2003

Dario Laverde, Jürgen Stuber, Guilio Ferrari, "Programming LEGO® Mindstorms™ with Java"

Bagnall, Brain „Navigation in LEGO® Mindstorms™ Programming", 2002

Overmars, Mark „Programmieren von LEGO® Mindstorms™ -Robotern mit NQC"

http://www.theoinf.tuilmenau.de/~nuetzel/mindstorms/asr2002

http://www.smallrobots.de/ws2000/dackel/sensoren2.htm

http://www.baumfamily.org/nqc

http://lejos.sourceforge.net

http://www2.hs-harz.de/~fstolzenburg/lego/folien/praktikum.pdf

http://www.noga.de/legOS

http://www.lego.com/eng/info/default.asp?page=timeline6

http://www.hempeldesigngroup.com/lego/pbForth/homePage.html

http://www.informatik.fh-muenchen.de/~schieder/blockunterricht-02