SP2007 Teamrobotik
(Lego Mindstorms)

Daniel Meyer, Nils Müller & Felix Penzlin

SS 2007

Inhaltsverzeichnis

1 Vorwort

1.1 Einführung

Die Arbeitsgruppe für Echtzeitsystem und Kommunikation am Institut für Verteilte Systeme der Otto-von-Guericke-Universität Magdeburg bietet seit mehreren Jahren Softwarepraktika zur Thematik autonomer Systeme an. Ein Softwarepraktikum ist im Zuge des Grundstudiums vorgesehen. Im Sommersemester 2007 wird hierfür erstmals Lego Mindstorms NXT als Plattform eingesetzt. Sechs Gruppen mit jeweils drei Studenten werden mit NXT-Bausätzen ausgestattet, um die Möglichkeiten des neuen Bausatzes zu ermitteln und vielfältige Aufgaben zu bewältigen.

1.2 Das Team

Unsere Projekt-Gruppe setzt sich aus Studenten unterschiedlichster Spezialisierungen zusammen. Was bei der Terminfindung für die Zusammenarbeit Probleme bereitet, hilft bei der Arbeit selbst durch die verschiedenen Blickwinkel. So besteht das Team aus einem Informatiker, Daniel Meyer, einem Wirtschaftsinformatiker, Nils Müller, und einem Ingenieur-Informatiker, Felix Penzlin.

1.3 Motivation

Die Motivationen für die Bewerbung ist so verschieden, wie die Mitglieder der Gruppe. Gemeinsam ist uns jedoch, dass wir in der Kindheit mit Lego gespielt haben. Autonome Systeme interessieren ebenfalls die gesamte Gruppe. Auch Neugierde ist durch die neue Technik im Spiel. Die Teamarbeit als solche ist ebenfalls ein Motivationsgrund.

1.4 Aufgabenstellung und Überblick

Die erste Teilaufgabe war es sich mit der vorhanden Hard- und Software auseinanderzusetzen. Dabei sollte darauf geachtet werden welche Entwicklungsumgebungen für den NXT verfügbar waren und was für Vor- und Nachteile sie jeweils hatten. Die Ergebnisse dieser kleinen Recherche finden sich in Kapitel 2.

Nachdem wir uns mit den Entwicklungsumgebungen beschäftigt hatten sollten wir uns den Sensoren zuwenden. Es sollte festgestellt werden in welcher Form Messdaten vorliegen, von welchem Typus sie sind und wie präzise die Sensoren arbeiten. Die Ergebnisse zu den Messungen sind in Kapitel 3 dargestellt.

Die Hauptaufgabe setzt sich aus folgenden Teilaufgaben zusammen:

- Der Roboter wird an einer zufälligen Stelle des Spielfelds aufgestellt.
- Nun wird der Infrarotball, nicht direkt sichtbar für den Roboter, versteckt.
- Jetzt startet der Roboter mit der Ballsuche.
- Es sollte bei der Ballsuche nicht zu Kollisionen mit Hindernissen kommen.
- Ist der Ball gefunden und gegriffen, wird ein akustisches Signal gegeben.

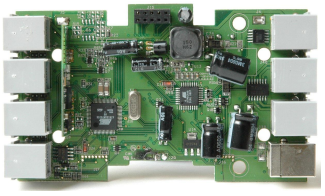
- Nach Geben des Signals, wird die Taschenlampe eingeschaltet.
- Zu der Taschenlampe soll der Roboter nun erneut kollisionsfrei fahren und den Ball freigeben.

Während der Roboter den Ball holt, wird er vom NXT einer Partner-Gruppe überwacht. Falls er ausfällt oder die Aufgabe beendet, ist es ihre Aufgabe den Ball zu holen. Auf die Anforderungen an den Roboter und die gewählten Lösungsansätze gehen wir in Kapitel 5 ein. Die Programmierung des Roboters für die gewählte Aufgabe wird in Kapitel 6 behandelt.

Als Zusatzaufgaben wurden gestellt den Roboter per Laptop oder Handy steuern zu können, Senormessdaten per Bluetooth zu übertragen und übertragene Sensormessdaten zu Visualisieren. Auf diese drei Zusatzaufgaben und ihre Umsetzung gehen wir im Kapitel 6 ein.

2 NXT und Technik

2.1 Der NXT



Der Bausatz besteht aus 519 Elementen. Das Kernstück jedoch ist der NXT-Baustein. Dieser verleiht mit einem Atmel 32-bit ARM7 Microcontroller, welcher auf 256Kbyte Flash-RAM und 64 Kbyte RAM [AT91SAM7S256] zugreift, sowie einen Atmel 8-bit AVR Microcontroller mit 4KByte Flash-RAM und 512 Byte RAM [ATmega48] den Modellen Intelligenz. Die Stromversorgung erfolgt über sechs Mignon-Batterien oder durch einen speziellen Akku. Verbindung zum Rechner kann über den USB-2.0 Anschluss oder über Bluetooth aufgenommen werden. Die Interaktion mit der Außenwelt erfolgt über drei Ausgänge, vier Eingänge, einen Lautsprecher und eine Punktmatrix LC-Anzeige von 100x64 Pixeln. Die Ein- und Ausgänge sind sechsadrig ausgeführt.

2.2 Sensoren und Aktoren

Im Lieferumfang enhalten sind zwei Berührungssensoren, welche zwischen den Zuständen gedrückt und nicht gedrückt unterscheiden. Sie sind auch als Schalter einsetzbar.

Der Akustiksensor ermöglicht es mit Hilfe eines Mikrofons die Intensität von Geräuschen zu ermitteln. Der Wertebereich liegt zwischen 0 und 1023.

Beim Lichtsensor stehen dem Nutzer zwei Betriebsmodi zur Verfügung. Entweder schaltet man ihn in den Umgebungslicht-Modus. Dann wird über eine Diode das einfallende Licht gemessen. Oder aber es wird die integrierte Leuchtdiode aktiviert, wobei dann an Hand des reflektierten Lichts die Farbe von Objekten geschätzt werden kann. Die Anbindung erfolgt analog. Die zurückgegebenen Werte liegen im Intervall von 0 bis 1023.

Der einzige digitale Sensor im Bausatz ist der Ultraschall-Sensor. Er ermittelt die Entfernung zu Objekten, indem er Ultraschall-Impulse aussendet und die benötigte Zeit des Echos misst. Er ist über den I^2C-Bus angebunden und gibt einen Wert zwischen 0cm und 255cm zurück.

Drei Motoren werden mitgeliefert. Sie sind mit auf ein Grad genauen Winkelmessern ausgestattet und erlauben verschiedene Ansteuerungsmodi. So können sie bei konstanter Geschwindigkeit oder mit konstantem Strom betrieben werden. Auch aktives Bremsen und Halten ist möglich.

Neben den genannten Sensoren lassen sich durch die einfachen Protokolle neue entwickeln oder aber kaufen. Es lassen sich neben Beschleunigungs-/Rotations-Sensoren und Kompassen auch Sensoren zur Farbbestimmung oder speziell zur Infrarotsuche finden.[HiTechnic]

2.3 Entwicklungsumgebungen

Zur Kontrolle des NXT wird eine Sprache benötigt. Im Jahre 2006 wurde von Lego die Firmware des NXT als OpenSource freigegeben.[OpenSource] Es ist somit möglich für den NXT eine eigene Firmware zu schreiben oder eigene Compiler zu entwerfen. Entwicklern wurden schon vor dem offiziellen Verkaufsstart Baukästen zur Verfügung gestellt, so dass bereits mehrere Entwicklungsumgebungen existieren. So gibt es neben dem von Lego vertriebenem Robolab, NXT-G und Robot C die Umgebungen NXC, Lejos und pbLua.

2.3.1 Robolab

Von National Instruments für den RCX entwickelt, stellt Robolab für Umsteiger von RCX auf NXT eine interessante Möglichkeit dar. Es handelt sich um eine symbolische Programmiersprache. Eine Lizenz kostet etwa €50.

2.3.2 NXT-G

Ebenfalls symbolisch gearbeitet wird in NXT-G. Es wurde speziell für den NXT entwickelt und basiert auf LabView von National Instruments. Es liegt dem im Handel erhältlichem Baukasten bei. Auf Grund des recht intuitiven Aufbaus ist es gut für Anfänger geeignet. Größere Projekte werden jedoch leicht unübersichtlich, auch lassen sich sehr spezielle Funktionen nicht direkt nutzen.

2.3.3 Robot C

Bei Robot C handelt es sich um eine Programmiersprache, die sich stark an C orientiert. Da sie kommerziell vertrieben wird, ist die Entwicklung hier ein Stück weiter als bei den vergleichbaren Projekten. In unseren Augen ist eine Investition trotz der Unterschiede nicht sinnvoll.

2.3.4 Lejos (NXJ)

Auf Java baut LeJOS auf. Es besteht aus einer Virtual Machine für den NXT, sowie Kompilierer, Linker und Upload-Programm. Die Virtual Machine wurde aus der TinyVM für RCX entwickelt und steht daher unter Mozilla Public License. Die Firmware des NXT muss also entfernt werden um LeJOS nutzen zu können. Hierdurch steht auch mehr Platz für eigene Programme zur Verfügung. Java ist stark objektorientiert, was es vereinfacht ordentlichen Code zu schreiben. Wiederverwendbarkeit ist in besonderem Maße gegeben. Gleitkomma-Arithmetik, Exception-Handling, Threads und Synchronisation sind implementiert. Informatik-Studenten der Otto-von-Guericke Universität Magdeburg verwenden in den ersten zwei Semestern in erster Linie Java, was die Sprache für uns besonders interessant macht. Allerdings befindet sich das Projekt derzeit noch in einem frühen Alpha-Stadium. So kann maximal ein Programm gleichzeitig auf dem NXT ausgeführt werden, da ein Menü derzeit noch fehlt. Auch die Bluetooth-Kommunikationsfähigkeiten sind derzeit noch eingeschränkt. Eine eigene integrierte Entwicklungsumgebung fehlt, allerdings ist mit Eclipse eine sehr gute Umgebung zur Java-Programmierung erhältlich. Programme lassen sich auf Windows und Linux compilieren und übertragen.

2.3.5 NXC

Ebenfalls in der Entwicklung, wenn auch weiter als LeJOS, befindet sich das ebenfalls freie Not eXactly C. Diese C-ähnliche Programmiersprache, sie gleicht NQC für den RCX, kommt mit einer eigenen integierten Entwicklungsumgebung für Windows, BricxCC. NXC-Code wird in Next Byte Code übersetzt, welcher anschließend assembliert wird. NBC gleicht Assemblercode. Da sich NXC und NBC mischen lassen, erhählt der Programmierer hiermit ein mächtiges Werkzeug. Dokumentationen und Tutorials sind für NXC reichlich zu finden. Um NXC-Programme nutzen zu können, muss die Original-Firmware nicht entfernt werden. Übersetzen von Programmen ist sowohl unter Linux als auch Windows möglich.

2.3.6 pbLua

Anders als bei den bisher betrachteten Programmiersprachen wird Lua interpretiert. So ist pbLua ein Interpreter der auf dem NXT die Original-Firmware ersetzt und in Lua geschriebene Programme zur Laufzeit übersetzt. Die Notwendigkeit eines Kompilierers entfällt somit. Lua wurde ursprünglich entwickelt, um Scripte in Programme einzubinden, damit diese besser wartbar werden. Der Lua-Interpreter ist sehr klein und überzeugt durch seine hohe Geschwindigkeit. Die Sprache ähnelt Python und besticht durch ihre Einfachheit. Zwar ist pbLua noch in einem frühen Entwicklungsstadium, dennoch sind die meisten Funktionen bereits enthalten. Die Bluetooth-Unterstützung ist hier sehr fortgeschritten. So kann beispielsweise mit einem Bluetooth-GPS-Empfangsgerät kommuniziert werden.[pbLuaGPS] Ein weiterer Vorteil ist die Simulierbarkeit. Programme können vorher auf dem Rechner getestet werden.

2.3.7 Wahl der Entwicklungsumgebung

Die Auswahl bei uns beschränkt sich auf NXT-G, NXC und Lejos. So entdeckten wir pbLua zu spät und die anderen kommerziellen Umgebungen standen nicht zur Wahl. NXC überzeugt im ersten Moment durch die mitgelieferte Umgebung und die fehlende Notwendigkeit eines Resets des NXT. Allerdings wurde das Projekt mit wachsender Größe immer unübersichtlicher. Auch bereitete die Rotation auf der Stelle große Probleme. So nutzen wir NXC für kleinere Programme, für größere Aufgaben jedoch wählen wir LeJOS.

2.4 Bluetooth

Bluetooth, auch bekannt als IEEE 802.15.1, ist eine Spezifikation für kabellose Personal Area Networks (PANs). Der Bluetooth-Standard legt fest, wie Informationen zwischen den Kommunikationspartnern ausgetauscht werden. Eine Bluetooth-Verbindung kann zwischen NXT und beispielsweise Mobiltelefon, Rechner oder einem anderen NXT aufgebaut werden. Der NXT kann mit bis zu drei Bluetooth-Geräten gleichzeitig verbunden sein. Kommunikation ist jedoch nur mit einem Gerät auf einmal möglich. Erfüllt wird vom NXT der Bluetooth Class II V2.0 Standard. Er unterstützt das serial port profile (SPP). Bei unseren Versuchen hat sich jedoch gezeigt, dass viele Geräte Inkompatibilitäten aufweisen.[BTCompatibility] Bei einem Adapter ließen sich diese durch ein Treiber-Update beheben.

2.5 Infrarot-Ball

Der Lichtsensor des NXT kann Infrarot-Licht sehr gut sehen. Daher ist der zu suchende Ball mit 20 Infrarot-Leuchtdioden ausgestattet. Der Durchmesser des Balls beträgt 7,6cm. Zum Betrieb des Balls werden vier AAA-Batterien benötigt, welche er in Abhänigkeit von der Qualität dieser in 90 Minuten leert. Verwendung findet diese Ballart auch beim Roboterfussball.

3 Sensoren

3.1 Schnittstellen

Es exisitieren Hard- und Softwareschnittstellen. Im folgenden werden Hardwareschnittstellen behandelt. Schnittstellen dienem dem Anschluss externer Geräte. So enthält die Spezifikation Informationen über Übertragungsgeschwindigkeiten, Übertragunsverhalten sowie Eigenschaften der Schnittstellenleitungen, Stecker und Buchsen sowie deren Belegung. Sinn einer Spezifikation oder Normierung ist, dass unterschiedliche Geräte verschiedener Hersteller miteinander verbunden werden können.

3.2 Messdaten

Wie Messdaten vorliegen, lässt sich aus zwei sehr unterschiedlichen Standpunkten betrachten. Zum einen könnte man einfach die Art der Messdaten auf Softwareebene beschreiben. Der Drucksensor zum Beispiel kennt nur zwei Zustände. Eine Messung ließe sich daher mit einem Bit codieren. Der Datentyp hierfür ist boolean. In Lejos wird üblicherweise der Messwert als Prozentwert zurückgegeben. Er liegt dann im Intervall von 0 bis 100 wobei 100 dem maximal messbarem Wert entspricht. Technisch möglich ist beim NXT allerdings eine feinere Auflösung. So liegen die von der Hardware gelieferten Werte im Intervall von 0 bis 1023. Auf Seite der Hardware lassen sich die Sensoren in zwei Kategorien einteilen. Der Ultraschallsensor arbeitet digital und im Gegensatz dazu die anderen Sensoren analog. Die analogen Sensoren funktionieren in gleicher Weise wie RCX-Sensoren und sind zu diesen kompatibel. Digitale Sensoren kommunizieren über einen I^2C-Bus.

3.2.1 RCX-Sensoren

Für RCX-Sensoren ergibt sich der Messwert aus dem Spannungsabfall über der Messschaltung. Errechnet wird er wie folgt, wobei D dem Messwert entspricht, U der gemessenen Spannung und R dem Wiederstand der Messschaltung zum Messzeitpunkt. Der Innenwiderstand des NXT beträgt 10kΩ.

$$D_{RCX} = U * 204,6\frac{1}{V}$$

$$U = 5V - 10k\Omega * I = \frac{5V}{\frac{10k\Omega}{R} + 1}$$

Die Werte können hiermit nur in einem Intervall von 0 bis 1023 liegen. Manche dieser Sensoren lassen sich in zwei Betriebsmodi betreiben. Im passiven Modus wird nur der Widerstand in beschriebener Art und Weise gemessen. Hingegen im aktiven Betriebsmodus liegt für 3ms Betriebsspannung am Sensoreingang an. Anschließend wird für 0,1ms im passiven Modus gemessen, während sich die Schaltung mit Hilfe eines Kondensators selbst mit Strom versorgt.

3.2.2 I^2C

Inter-Integrated Circuit, auf deutsch etwa Interner Integrierter Schaltungs-Bus, ist eine Entwicklung von Philips aus den 80iger Jahren. Es handelt sich hierbei um einen einfachen seriellen, binären Bus mit Master-Slave-Prinzip. Zum Aufbau reichen zwei bidirektionale Leitungen neben Erde und Betriebsspannung aus. Eine dient hierbei als Taktleitung, die andere führt die Daten. Somit lassen sich auch die sechs Kontakte des NXT erklären. Zwei Adern dienen den RCX-Sensoren, vier werden für I^2C benötigt.

4 Sensor-Messungen

Bevor mit der Programmierung begonnen werden kann, ist es sinnvoll sich mit der gegebenen Hardware vertraut zu machen, um so Stärken und Schwächen zu ermitteln. Es wird Genauigkeit der Sensoren, sowie die Leistungsfähigkeit anderer Bauelemente ermittelt. Begonnen wird mit dem Akustiksensor, gefolgt von Licht- und Ultraschallsensor. Zuletzt werden die Laufzeit sowie die Leistungsfähigkeit der Aktoren ermittelt.

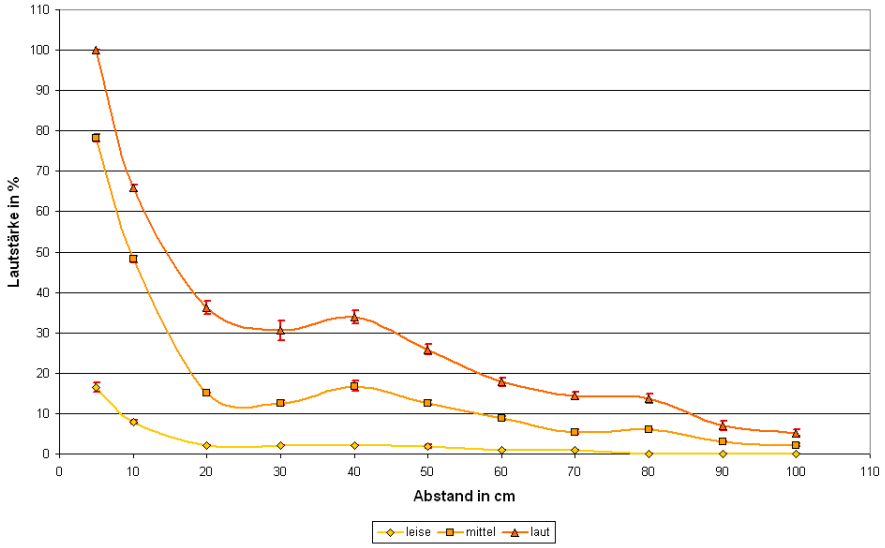
4.1 Akustiksensor

4.1.1 Versuchsaufbau

Zur Messung des Akustiksensor wird ein Mobilfunktelefon verwendet, welches in diskreten Abständen im Bereich von 5cm bis 100cm, mit dem Lautsprecher auf den Sensor ausgerichtet, durch den gleichen Klingelton in drei unterschiedlichen Lausträrken den Sensor anregt. Die Messung erfolgt in Prozent-Angaben. Auf eine geringe Umgebungslautstärke wurde genau geachtet. Diese lag in Messwerten des Sensors unter 3%.

4.1.2 Messergebnisse

Ermittelt wurden die Mittelwerte aus den drei Messserien mit jeweis fünf Messreihen. Die drei Lautstärken sind im Graph absteigend sortiert. Die Abweichungen zwischen den fünf Messungen ist als roter Balken dargestellt.

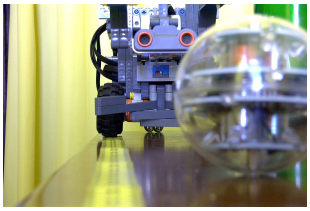


4.1.3 Auswertung

Bei geringer Lautstärke ist das Geräusch in größerer Entfernung praktisch nicht mehr wahrnehmbar. Die Abweichungen zwischen den fünf Messreihen sind sehr gering. Bei mittlere und hoher Lautstärke betragen die Abweichungen bis zu 2%. Überraschenderweise lagen die Messwerte bei einer Entfernung von 40cm höher als bei 30cm. Abgesehen von dieser Anomalie, gleichen die Kurven Hyperbeln, die sich asymptotisch Null nähern.

4.2 Lichtsensor

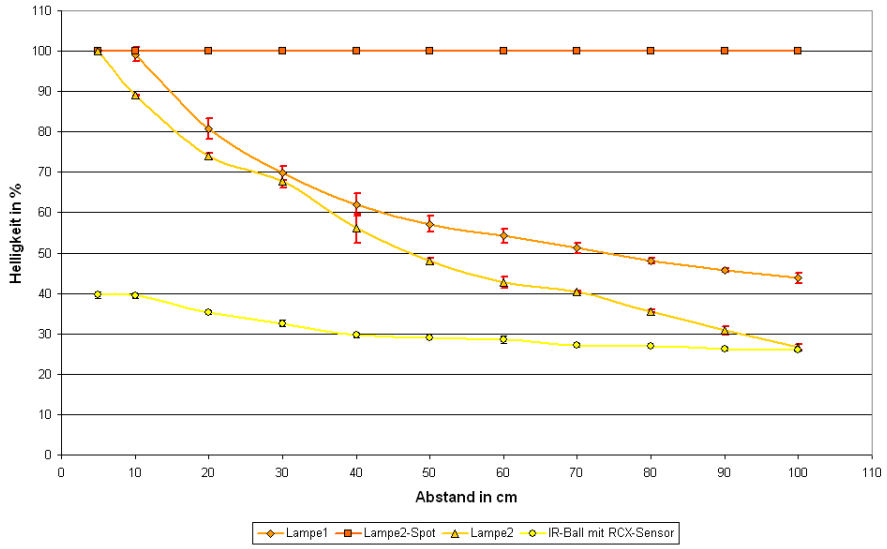
4.2.1 Versuchsaufbau



Auch der Lichtsensor wird in einem Bereich von 5cm bis 100cm gemessen, wobei er direkt auf die Lichtquelle ausgerichtet ist. Verwendung als Lichtqulle finden drei Taschenlampen unterschiedlicher Stärke, sowie der Infrarot-Ball. Wobei der Infrarot-Ball mit dem RCX-Sensor, die Taschenlampen mit dem NXT-Sensor gemessen werden. Als erste wird eine schwache Taschenlampe gemessen. Das Umgebungslicht liegt dabei zwischen 30% und 37%. Anschließend, bei einem Umgebungslicht von nur noch 1,5%, erfolgt die Messung einer stärkeren Taschenlampe die gebündeltes Licht erzeugt und einer starken Taschenlampe mit ungebündeltem Licht. Danach, bei einem Umgebungslicht von 23% wird der Infrarot-Ball, mit der Schraube auf den Lichtsensor ausgerichtet, verwendet. Die erste und vierte Messserie erfolgt in Prozent, bei der Messserie zwei und drei wurden Rohdaten-Werte verwendet. Wieder werden für jede Messung fünf Messreihen erhoben.

4.2.2 Messergebnisse

Aus den jeweils fünf Messreihen sind die Mittelwerte als Graph dargestellt. Die Standardabweichung zwischen den einzelnen Messungen ist als Fehler in rot zu finden.

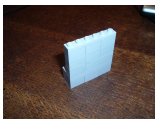


4.2.3 Auswertung

Verschiedene Probleme traten beim Messen auf. Die schwache Taschenlampe erschwerte durch einen Wackelkontakt das Erhalten konstanter Werte. Außerdem ließdie Leuchtkraft in der fünften Messreihe nach. Beim dritten Test waren die Messwerte im Abstand von 30cm tendenziell etwas zu hoch. Die gemessenen Werte beim Infrarot-Ball hängen sehr stark von seiner Ausrichtung ab, was in der Praxis berücksichtigt werden muss. So kann der Ball bis zu einer Entfernung von 50cm gut gesehen werden. Ein schätzen von Entfernungen über die Lichtwerte ist eher nicht möglich. Was sich hier nicht direkt erkennen lässt, ist, dass der NXT-Sensor etwas sensibler misst als der RCX-Sensor.

4.3 Ultraschallsensor

4.3.1 Versuchsaufbau

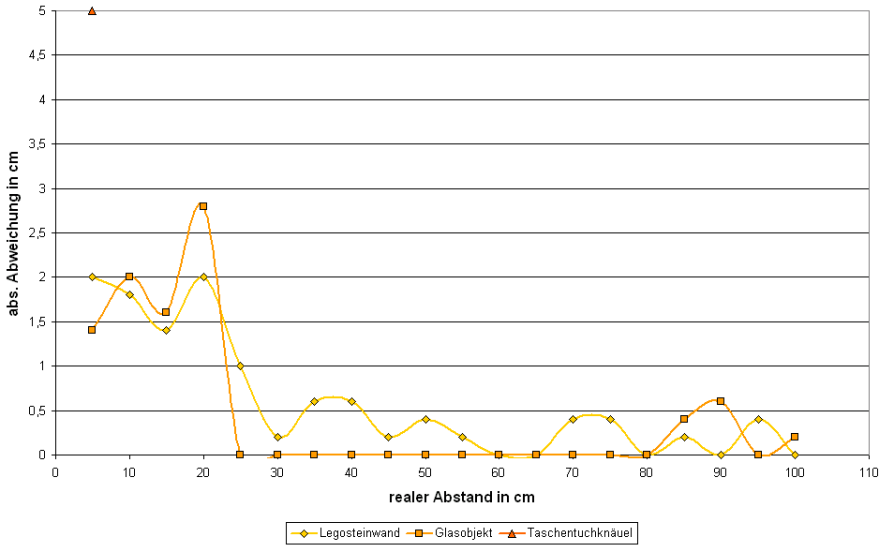


Gemessen wird der Ultraschallsensor mit drei Objekten: einer Legostein-Wand, einem Glasobjekt und einem Taschentuchknäuel. Zur Bestimmung des Abstandes wird immer zwischen der Spitze des Ultraschallsensors und des Objekts gemessen. Die Legowand steht orthogonal zum Sensor.

Die Versuchsgegenstände werden wiederum in diskreten Abständen im Bereich von 5cm bis 100cm zum Sensor platziert. Als Messwerte liefert der Ultraschallsensor Zentimeterangaben. Der Abstand zum Untergrund ist großgenug, sodass die Messungen also nicht durch Echos, die durch die Umgebung verursacht werden, gestört werden. Durchgeführt werden zu jedem Objekt erneut fünf Messreihen.

4.3.2 Messergebnisse

Aus dem Mittelwert der jeweils fünf Messreihen wird die Abweichung zum wahren Wert berechnet. Die Abweichung des Taschentuchknäuel ist nicht sichbar, da hier keine sinnvolle Messung möglich ist.



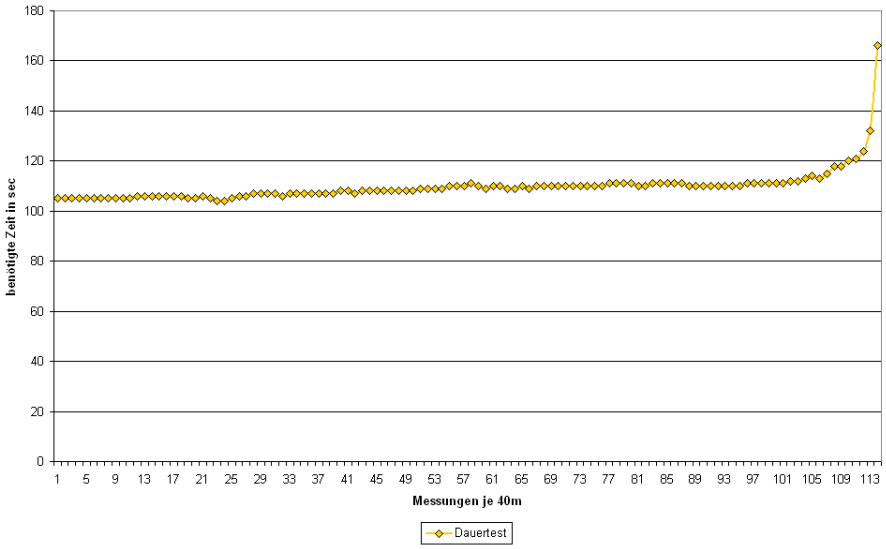
4.3.3 Auswertung

Die Entfernung zu schallschluckenden Objekten wie dem Taschentuchknäuel lässt sich mit dem Ultraschallsensor nicht ermitteln. Bei sehr geringen Entfernungen, unter 20cm, werden die Messwerte ungenau. Darüber liegt die Abweichung unter 1cm. Unter idealen Bedingungen entspricht der Ultraschallsensor also unseren Anforderungen.

4.4 Motortest

4.4.1 Ausdauertest

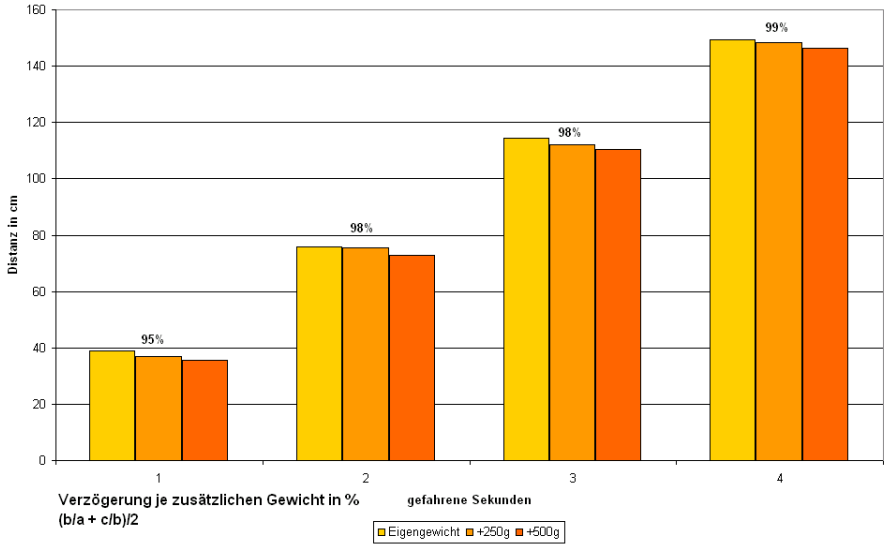
Für den Ausdauertest ist es nötig, dass der Roboter genau auf der vorgegebenen Strecke bleibt, da sich im Laufe der Zeit auch kleinste Abweichungen aufaddieren und so zu einer ungültigen Messung führen würden. Daher wurde die Achse des Roboters versteift. Beide Motoren können sich so nur noch in gleicher Ausrichtung und Geschwindigkeit drehen. Als Hindernisse werden zwei Bücherstapel im Abstand von einem Meter plus Länge des Roboters aufgebaut. Mit Hilfe von Berührungssensoren an Front und Heck nimmt der Roboter das Erreichen der Begrenzungen wahr. Gemessen wird die benötigte Zeit, um zehnmal die Strecke von zwei Metern abzufahren, also zehnmal vor und zurück zu fahren.

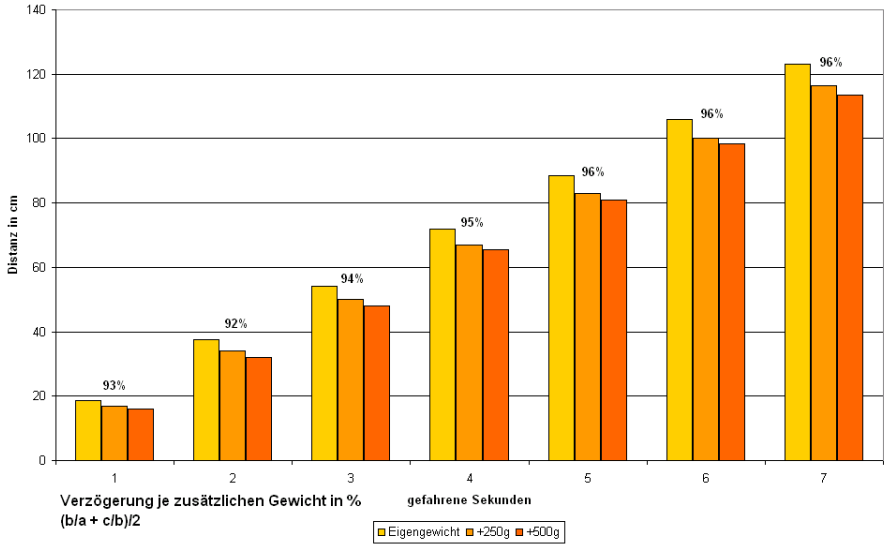


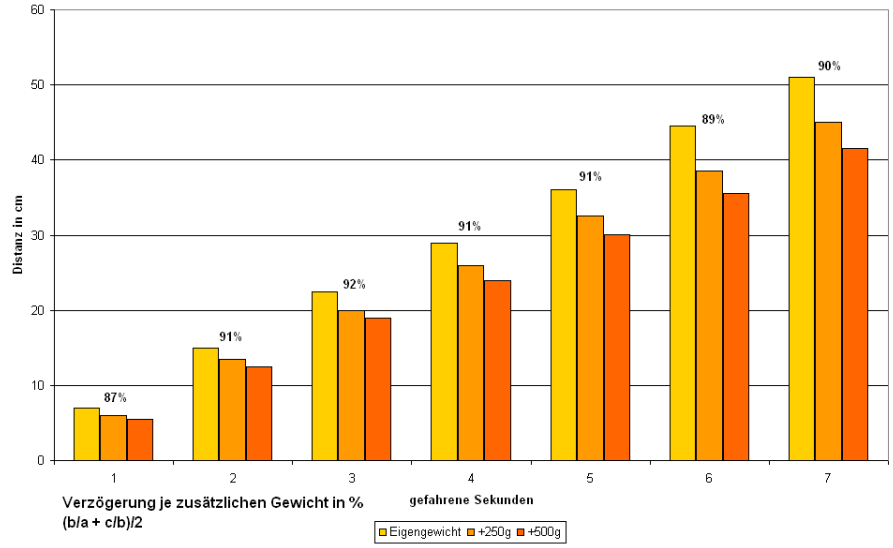
Am Anfang wird zum Abfahren der Strecke eine Zeit von etwas über 100 Sekunden benötigt. Diese steigt dann im Laufe der Zeit etwa linear, allerdings nur in geringem Maße, bricht dann jedoch sehr plötzlich nach oben aus. Die mit einer Akkuladung bewältigte Strecke beträgt ca. 4532 Meter. Die Gesamtlaufzeit liegt bei 3 Stunden, 26 Minuten und 29 Sekunden, was als sehr gut angesehen werden kann.

4.4.2 Leistungstest

Beim Leistungstest wird die gefahrene Strecke unter Belastung in einer bestimmten Zeit gemessen. Die zusätzlichen Gewichte betragen 250g und 500g. Das Eigengewicht des Modells liegt bei 500g. Zwischen einer und sieben Sekunden liegt die Fahrzeit. Erhöht wird sie in Sekundenschritten. Durchgeführt wurden 3 Messserien, wobei für die erste eine Geschwindigkeit von 25%, die zweite 50% und die dritte 100% gewählt werden.


160
140
120
100
80
60
40
20
0
Distanz in cm
95%
98%
98%
99%
1
2
3
4
Verzögerung je zusätzlichen Gewicht in %
(b/a + c/b)/2
gefahrene Sekunden
Eigengewicht
+250g
+500g


140
120
100
80
60
40
20
0
Distanz in cm
93%
92%
94%
95%
96%
96%
96%
1
2
3
4
5
6
7
Verzögerung je zusätzlichen Gewicht in %
(b/a + c/b)/2
gefahrene Sekunden
Eigengewicht
+250g
+500g



Die Verzögerung durch die Belastung kann als gering betrachtet werden. Ein linearer Zusammenhang zwischen gewählter Geschwindigkeit und wirklich gefahrener Strecke ist nicht ersichtlich.

4.4.3 Genauigkeitstests

Ziel des Genauigkeitstests war es die drei Motoren auf etwaige Unterschiede hin zu überprüfen und möglichst 2 Motoren zu finden, die sich ähnlich verhalten, um diese als Antriebsmotoren zu nutzen.

Manueller Gradtest Beim manuellen Gradtest haben wir die internen Rotationszähler zuerst auf 0 zurückgesetzt. Dann haben wir die Motoren manuell gedreht und danach die Rotationswerte abgelesen. Die Werte sind in Tabelle 1 zu sehen.

Tabelle 1: Manueller Gradtest der Motoren

Umdrehung	1	5	10	15	20
Rotationsmesser Motor 1	354°	1798°	3597°	5401°	7201°
Rotationsmesser Motor 2	360°	1793°	3608°	5405°	7200°
Rotationsmesser Motor 3	358°	1798°	3604°	5402°	7204°

Was auffällig ist, dass es zwar Abweichungen von den tatsächlichen Gradzahlen gibt, aber diese nur sehr geringfügig sind.

Gradtest Beim Gradtest haben wir die internen Rotationszähler zuerst auf 0 zurückgesetzt. Dann haben wir den Motor sich um eine vorgegebene Gradzahl drehen lassen. Die Drehung erfolgte daei mit einer Geschwindigkeit von 50% der Motorenkraft. Danach haben wir die Werte mittels des Rotationszählers ausgelesen. Die Werte sind in Tabelle 2 zu sehen. Bei diesem Test kam es schon zu größeren Abweichungen zum Optimalwert, besonders Motor drei weicht hier ab.

Tabelle 2: Gradtest der Motoren

Grad	180°	360°	720°	3600°	7200°
Rotationsmesser Motor 1	183°	362°	722°	3606°	7207°
Rotationsmesser Motor 2	183°	363°	726°	3603°	7203°
Rotationsmesser Motor 3	183°	370°	726°	3596°	7183°

Zeittest Beim Zeittest wollten wir herausfinden, wie das Laufverhalten der einzelnen Motoren ist. Zu diesem Zweck haben wir die Motoren für eine gewisse Zeit bei einer gewissen Geschwindigkeit angestellt und den zurückgelegten Weg in Grad ausgelesen. Die Werte für 100% Geschwindigkeit sind in Tabelle 3, für 50% Geschwindigkeit in Tabelle 4 und für 10% Geschwindigkeit in Tabelle 5 zu finden.

Tabelle 3: Zeittest der Motoren bei 100% Geschwindigkeit

Sekunden	1	5	10	15	20
Rotationsmesser Motor 1	860°	4305°	8605°	12928°	17282°
Rotationsmesser Motor 2	855°	4285°	8575°	12865°	17119°
Rotationsmesser Motor 3	853°	4282°	8576°	12866°	17145°

Tabelle 4: Zeittest der Motoren bei 50% Geschwindigkeit

Sekunden	1	5	10	15	20
Rotationsmesser Motor 1	423°	2122°	4242°	6362°	8486°
Rotationsmesser Motor 2	422°	2118°	4244°	6362°	8480°
Rotationsmesser Motor 3	422°	2116°	4227°	6338°	8448°

Tabelle 5: Zeittest der Motoren bei 10% Geschwindigkeit

Sekunden	1	5	10	15	20
Rotationsmesser Motor 1	72°	362°	725°	1085°	1445°
Rotationsmesser Motor 2	71°	350°	700°	1047°	1400°
Rotationsmesser Motor 3	72°	362°	722°	1083°	1438°

Bei den Zeittests traten zum teil größere Unterschiede zwischen den Motoren auf. So liegen zwischen Motor 1 und Motor 2 in Tabelle 3 bei 20 Sekunden Laufzeit 163 Grad. Das kann bei längereren Geradeausfahrten zu Problemen führen. Nicht nur dass die Motoren sich bei gleicher Laufzeit unterschiedlich verhalten, auch bei verschiedenen Geschwindigkeiten ist das Verhältnis zwischen den Motoren verschieden. So verhalten sich Motor 1 und Motor 3 bei einer

Geschwindigkeit von 10% nahezu gleich, aber bei einer Geschwindigkeit von 100% verschieden. Zusammenfassend kann man sagen, dass es uns nicht möglich war, 2 Motoren mit annährend gleichem Verhalten zu finden, um sie als sichere Antriebsmotoren für unseren Roboter zu verwenden.

5 Roboter

5.1 Anforderungen

Aus den Aufgaben ergeben sich zahlreiche Forderungen. So muss sparsam mit den Bausteinen umgegangen werden, denn die Steinanzahl ist stark begrenzt. Trotzdem muss er in der Lage sein, den Infrarot-Ball zu sehen und ihn zu greifen. Ausserdem ist nötig, dass er anschließend eine Taschenlampe ausfindig machen kann. Das größte Problem jedoch sind die Hindernisse. So ist der Rand nur minimal höher als der Ball, was die Freiheit beim Bau einschränkt.

5.2 Namensfindung

In der Gruppe wurde ein Name gewählt, welcher auch für die Erkennung über Bluetooth zur Identifizierung notwendig ist. So trug er zuerst den Namen Moritz, wurde dann jedoch auf Grund seinen Verhaltens in Marvin umbenannt und wird im folgenden auch so bezeichnet.

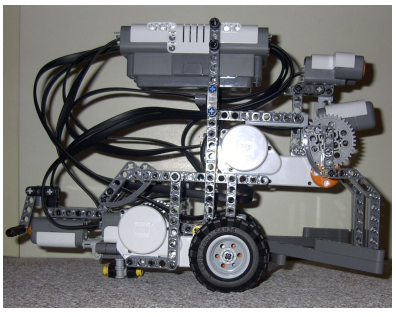
5.3 Probleme

Das größte Problem bestand am Anfang in der geringen Steinanzahl, wodurch Freiheit und Kreativität beim Bau erst einmal stark eingeschränkt wurden. Zuerst stand uns nur ein Lichtsensor zur Verfügung, wodurch das Finden der Taschenlampe, ohne dass der Sensor vom leuchtenden Ball gestört wird, erschwert wurde. Das Anbringen des Ultraschallsensors, so dass er den Rand erkennen kann, aber baulich nicht von Greifer oder Ball gestört wird, bereitete weitere Probleme. Die glatte Oberfläche des Tisches sorgt dafür, dass die Räder manchmal durchdrehen. Die größte Einschränkung jedoch war bei den ersten Überlegungen, dass nur drei Motoren angesteuert werden können.

5.4 Lösungsansatz

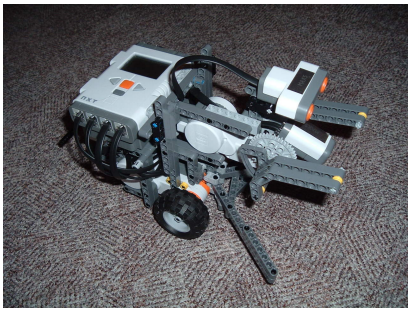
Dem NXC-Baukasten liegen ausschließlich Räder bei, somit fiel die Wahl des Antriebs leicht. Der Greifer sollte von oben arbeiten und nach Möglichkeit den Ball heraus befördern können.

5.4.1 Version 1



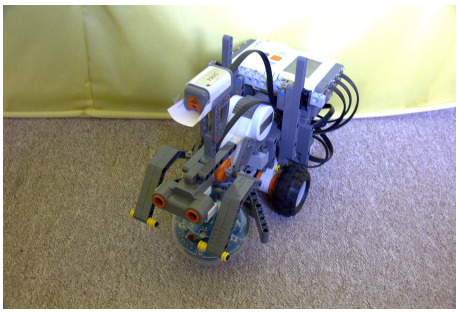
Die erste Version von Marvin wurde sehr hoch, was bei hartem Bremsen fast zum Umkippen führte. Das besondere an dieser Version ist, dass der Lichtsensor sich durch den Vorgang des Greifens nach oben zur Taschenlampe ausrichtet und so nicht durch den Ball gestört wird. Der Ultraschallsensor bei diesem Modell war allerdings viel zu hoch angebracht für das beschriebene Szenario. So musste hier nachbessert werden.

5.4.2 Version 2



Da der NXT-Baustein das schwerste Element darstellt, wird dieser bei der zweiten Version so tief wie möglich angebracht, ohne jedoch in eine andere Richtung zu wachsen. Es wird Fahrstabilität gewonnen, ohne neue Nachteile in Kauf nehmen zu müssen. Der Ultraschallsensor war jedoch immer noch zu hoch angebracht, was eine dritte Version nötig machte.

5.4.3 Version 3



Am Heck von Version 2 ist ein Touchsensor angebracht, welcher entfernt wird, um Marvin weiter zu verkleinern. Der Ultraschallsensor befindet sich nun an der Spitze, direkt über dem Ball und kann seiner Aufgabe nun besser nachkommen. Zusätzlich wurde ein Lichtsensor eingebaut, der ausschließlich der Ballsuche dient.

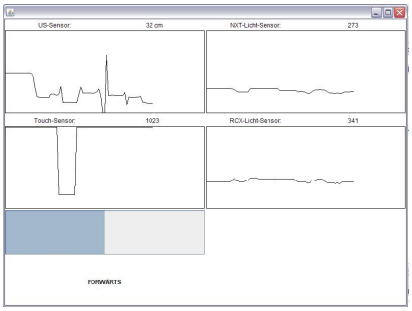
6 Programmierung

6.1 Sensor-Messungen

Die Programme zur Messung der Sensoren sind fast vollständig in NXC geschrieben. Ihre Aufgabe besteht darin, die Messwerte auf der LCD-Anzeige des NXT darzustellen. Ausschließlich der RCX-Licht-Sensor wurde mit LeJOS gemessen. Für die Messwerte bedeutet die verwendetet Software jedoch keine Veränderung.

6.2 Visualisierung

Gewünscht war es, dass die Messwerte zur Laufzeit über Funk übertragen werden und auf dem Rechner visualisiert werden, und dass eine Steuerung des NXT ebenfalls über Funk ermöglicht wird.



Um die Aufgabe zu lösen, verwenden wir die Biliothek iCommand für Java. Diese erlaubt es, dem NXT Befehle während der Laufzeit über Bluetooth zu senden und Messwerte zu erfragen. Die Steuerung des Roboters erfolgt mit Hilfe der Tastatur. Die Tasten W,A,S,D dienen zur Bewegung in beliebige Richtung, R und F legen die Geschwindigkeit fest, O und L wurden zum Öffnen und Schließen des Greifers gewählt. Dargestellt werden die vom NXT übertragenen Messwerte in vier getrennten Graphen sowie der Momentanwert als Zahl.

6.3 Balljagd

Die Grundidee hinter dem Programm war, dass der Roboter den Ball durch Rotation um die eigene Achse ausfindig macht, aufmerksam in die entsprechende Richtung startet und Kurskorrekturen vornimmt, falls er Hindernissen begegnet oder das Ziel seine Position verändert. Ist der Ball aufgenommen, was über den Berührungssensor festgestellt wird, so soll er in gleicher Weise zur Taschenlampe starten. Wird nun ein gewisser Lichtwert überschritten, so ist der Ball abzuschießen, da die gewünschte Nähe zur Taschenlampe erreicht ist. Als zusätzliche Aufgabe war eine Kommunikation mit dem Roboter eines anderen Teams gefordert. In unserem Fall sollte dieser, im Falle eines Ausfalls unseres Roboters, die Aufgabe zuende führen. Das andere Team verwendet NXC, wir dagegen NXJ, die Progammierung auf beiden Seiten ist daher verschieden. Zur Kommunikation wird eine Bluetoothverbindung hergestellt, unser Roboter befindet sich dabei in der Rolle des Slave, der Roboter des anderen Teams ist Master und führt so die Überwachung durch.

7 verwendete Software

- LeJOS
- iCommand
- BricxCC
- NBC

Literatur

[AT91SAM7S256] http://www.atmel.com/dyn/resources/prod_documents/6175s.pdf

[ATmega48] http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf

[HiTechnic] http://www.hitechnic.com

[OpenSource] http://mindstorms.lego.com/press/2057/Open%20Source%20Announcement.aspx

[pbLuaGPS] http://www.hempeldesigngroup.com/lego/pbLua/pbLuaGPS.html

[BTCompatibility] http://www.vialist.com/users/jgarbers/NXTBluetoothCompatibilityList

[sa06] Semesterarbeit 2006.18, C. Frischknecht & T. Other