OTTO-VON-GUERICKE-UNIVERSITÄT MAGDEBURG

Softwarepraktikum Teamrobotik SS 2007

Alexa Kernchen & Diana Schult

30.08.2007

Inhaltsverzeichnis

Einleitung

Diese Dokumentation beschäftigt sich mit der Projektarbeit im Rahmen des Softwarepraktikums Teamrobotik SS 07.

Das Praktikum befasste sich mit den Robotern von „Lego-Mindstorms“. Die Teams bestanden aus 2 bis 3 Personen.
Im Gegensatz zu den vorherigen Jahrgängen wurde uns der neue Lego-Mindstorms Roboter, der NXT, zur Verfügung gestellt. Zusätzlich bekamen wir auch den alten Roboter, RCX, ausgehändigt.

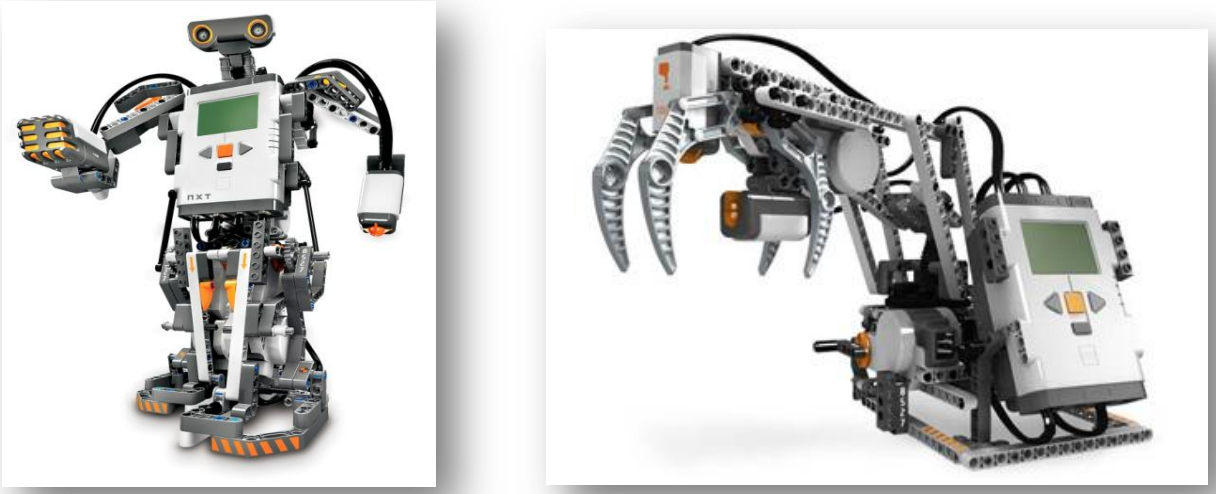


Abbildung 1: NXT- Bauvarianten

Aufgabenstellungen

Erste Schritte

Nachdem wir den NXT-Baukasten ausgehängt bekamen, bauten wir den in der Bauanleitung beschriebenen Roboter und erstellten mit der NXT-View Software im NXT das erste Programm.

Abbildung 2: Lego Mindstorms NXT Education Base Set

Anschließend beschäftigten wir uns mit der mitgelieferten Software von Mindstorms und erstellten die ersten größeren Programme für den Roboter.

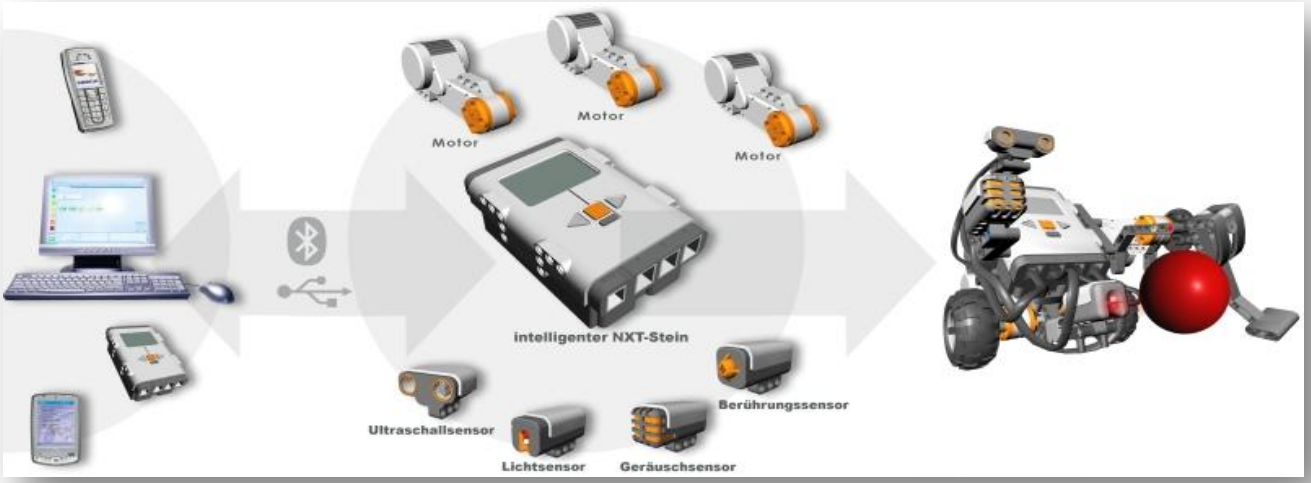


Abbildung 3: Erste Schritte

Diese ersten Schritte gaben uns einen groben Einblick in die Möglichkeiten des NXT und motivierten uns für die folgenden Aufgabenstellungen.

1. Aufgabe

Aufgabenstellung

Unsere erste Aufgabe bestand im Kennenlernen der Hard- und Software. Außerdem sollten wir die Entwicklungswerkzeuge testen.

Hardware

Zur Verfügung stand uns der NXT Baustein mit integriertem wiederaufladbarem Akku, drei Motoren mit integriertem Rotationssensor und fünf weitere Sensorentypen (ein Ultraschallsensor, ein Akustiksensor, zwei Drucksensoren und ein Lichtsensor), die im folgenden noch genauer beschrieben werden.

Abbildung 4: Der NXT mit seinen Sensoren und Motoren

Der NXT Baustein verfügt über acht Schnittstellen. Dazu gehört der USB-Anschluss, drei Anschlüsse für Motoren und vier weitere für Sensoren. Intern besitzt der NXT-Baustein einen 32-bit Mikroprozessor, 56 kbyte FLASH- und 64kbyte RAM-Speicher. Im Gegensatz zum alten Roboter, dem RCX, besitzt der NXT außerdem eine unbegrenzte Anzahl an Programmspeicherplätzen.
Als Kommunikationsschnittstellen besitzt der NXT einen USB-Anschluss und integriertes Bluetooth. Zudem stehen integrierte Lautsprecher und ein graphisches Display, das über interne und externe Software programmierbar ist, zur Verfügung.

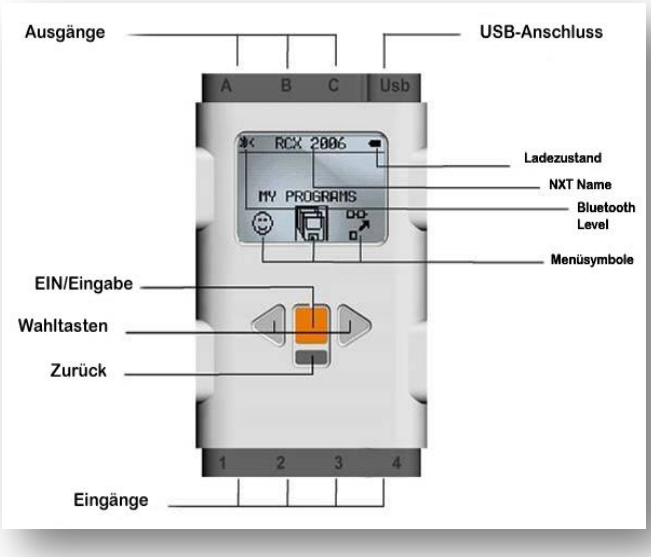


Abbildung 5: Der NXT

Der Ultraschallsensor nutzt Ultraschallmessungen, um seine Umgebung wahrzunehmen. Er sendet ein Signal aus und wartet auf dessen Rückkehr.

Abbildung 6: Der Ultraschallsensor

Anhand der gemessen Zeit, die das reflektierte Signal braucht, kann er die Entfernung nun genau bestimmen. Je länger es dauert bis der Sensor das wiederkehrende Signal empfängt, desto größer ist die Entfernung zum Objekt.

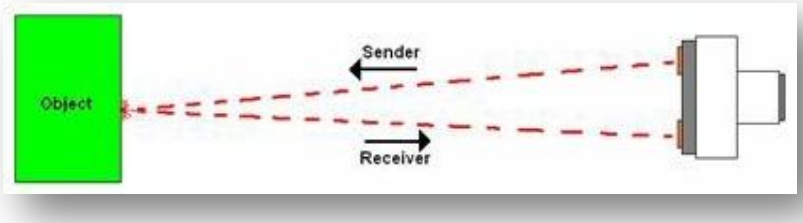


Abbildung 7: Grundlage der Entfernungsberechnung: die Zeit zwischen Aussendung und Empfang des Signals

Der Geräuschsensor misst den Schalldruck in db und dbA. Er kann auf leise und laute Kommandos reagieren, sowie auf Kommandos die von einer Person gegeben werden. Außerdem erkennt bestimmte Klangmuster und unterschiedliche Tonhöhen.

Abbildung 8: Der Geräuschsensor

Der Lichtsensor erlaubt Lichtmessungen auf einer Skala zwischen 0 (kein Licht) und 100 (sehr hell). Es ist ebenfalls möglich das Infrarotlicht (reflected light) unterhalb des Sensors auszuschalten, um nur das Umgebungslicht zu messen (ambient light).

Abbildung 9: Der Lichtsensor

Der Berührungssensor hat eine kreuzförmige Vertiefung im Druckknopf um dem Nutzer mehr Möglichkeiten zu bieten, das Gerät in sein Roboterdesign zu integrieren.

Abbildung 10: Der Berührungssensor

Er kennt nur zwei Zustände: an (Stromfluss) und aus (Stromfluss unterbrochen), aber mit Hilfe von z.B. Bricx kann der genaue Druck bestimmt werden, der auf den Sensor ausgeübt wird.

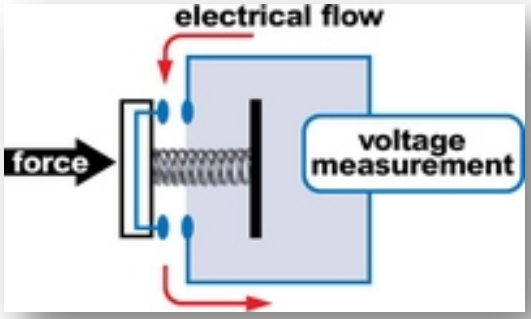


Abbildung 11: Aufbau des Sensors

Software

Zu den bekanntesten Softwareprodukten für den NXT gehören LeJos (Java, OpenSource), BricxCC (NXC, OpenSource), Robolap 2.9 (kommerziell) und die mitgelieferte Software von LEGO-Mindstorms (kommerziell).

Die Lego Mindstorms Edu-NXT-Software bot eine gute Einstiegsmöglichkeit, da auf leichte Art und Weise die einzelnen Sensoren und Motoren angesprochen werden konnten und somit die Programmierung vereinfacht wurde, ohne weiter Programmierkenntnisse besitzen zu müssen. Allerdings stößt man aufgrund der Programmstruktur schnell an die Grenzen der Übersichtlichkeit. Ein weiteres Manko betrifft den größeren Speicherumfang für die Programme dieser Software, im Gegensatz zu einem äqivalenten Bricx-Programm.

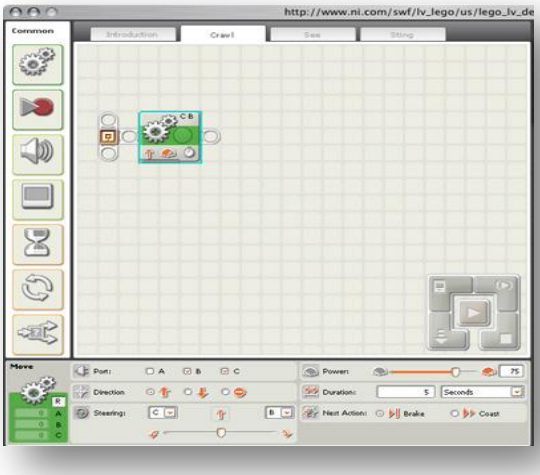


Abbildung 12: Lego Mindstorms Edu-NXT-Software

Das OpenSource-Programm Bricx war schnell und leicht zu installieren. Programmiert wurde in der Programmiersprache NXC (Not eXactly C). Bricx bietet im Gegensatz zu den anderen Programmen ebenfalls die Möglichkeit mit dem RCX zu kommunizieren.
Im Internet sind außerdem hilfreiche (englische) Manuals zu finden, mit denen über eine kurze Einarbeitungszeit schnell die nötigen Befehle erlernt werden können. Hierbei wird schnell deutlich, dass ein mächtigerer Befehlssatz zur Verfügung steht und nun die Sensoren und Motoren explizit angesprochen, kalibirert und verwendet werden können. Positiv ist uns besonders aufgefallen, dass mit Hilfe des integrierten Programmes „Watch the Bricx“ die Werte jedes einzelnen Sensors und Motors abgefragt, gespeichert und grafisch dargestellt werden können.

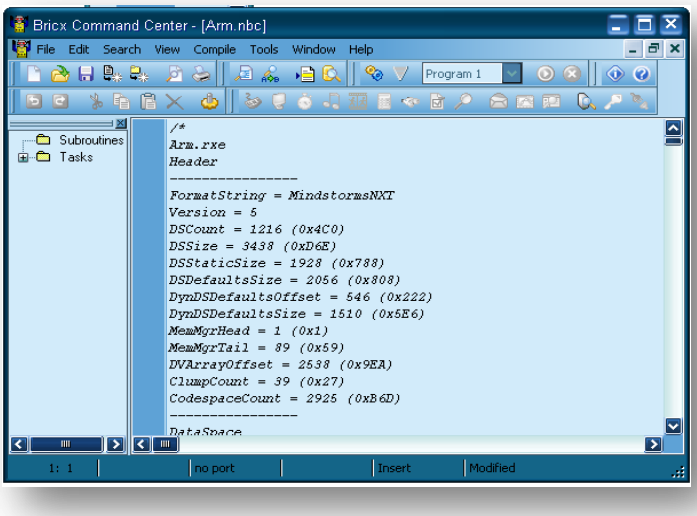


Abbildung 13: Bricx

Eine Bewertung für Robolap war uns nicht möglich gewesen, da dieses kommerziell ist und daher extra gekauft werden müsste. Ebenso konnten wir LeJos nicht testen, da es uns nicht möglich war es lauffähig zu installieren.

2. Aufgabe

Aufgabenstellung

Nach diesen allgemeinen Grundlagen sollten wir uns nun mit den Sensorschnittstellen, deren Messdaten und eigenen Test mit dem Motor, Ultraschallsensor, Lichtsensor und Akkustiksensor, beschäftigen.

Sensorschnittstellen

Wie anfangs erwähnt stehen uns drei Motoren zur Verfügung. Diese 9V-Schrittmotoren besitzen einen integrierten Rotationssensor, der eine Genauigkeit bis 1° hat.

Im folgenden erklären wir nun die vier uns zur Verfügung stehenden Sensoren:
Der Akustiksensor registriert Töne und Schallmuster und misst den Schalldruck in dB und dBA.
Über einfaches Ansprechen des Drucksensors ist herauszufinden, ob er gedrückt wurde oder nicht. Mit Hilfe von erweiterten Befehlen des NXC kann auch herausgefunden werden, in was für einem Maße dieser gedrückt wurde.
Der Lichtsensor kann auf Licht und Dunkelheit in einem Raum, sowie auf Lichtintensität, reagieren. Desweiteren bringt er zwei Modi mit (reflected und ambient light). Beim reflected light ist noch eine zusätzliche LED zugeschaltet.
Die Funktionsweise des Ultraschallsensor besteht darin, dass er ein Signal aussendet und auf die Rückkehr des Signals wartet um über die Zeitdifferenz die Entfernung zu bestimmen. Die gemessenen Werte werden für den Benutzer dann in inch oder cm zur Verfügung gestellt.

Bauform unseres NXT

Im Laufe des Arbeitens mit dem Roboter hat sich dieser in der Bauform weiterentwickelt. Wir haben ihm nun einen Arm hinzugefügt, der einen Ball greifen und transportieren kann. Desweiteren sind wir nun vollständig bei der Programmierung auf Bricx umgestiegen, da nun ein komplexeres programmieren notwendig wird.

Abbildung 14: Unser Max2

Die Tests

Akku-Test

Aufgrund des Zeitaufwandes den Akku zu testen hatten wir uns auf zwei Einzeltests beschränkt. Diese bestanden darin den Roboter unter Standardbelastung und mit 250g Butter beladen fahren zu lassen und die Zeit zu messen bis der Akku versagt. Zu beachten ist jedoch, dass sich der Roboter bei einem geringen Akkustand selbst ausschaltet. Somit konnte die tatsächliche Laufzeit nicht berechnet werden.
Beim Test ließen wir unseren NXT im Kreis fahren, wobei in den letzten Minuten die Laufform etwas von der Kreisform abwich. Ohne zusätzliche Belastung hielt unser NXT 5 Stunden und 3 Minuten, mit Belastung 4 Stunden und 35 Minuten, durch. Bei Belastung ist uns aufgefallen, dass er deutlich langsamer fuhr. Daraus schlossen wir, dass er auch eine kürzere Distanz zurückgelegt hatte.

Herangehensweise bei den weiteren Tests

Wir suchten uns die einzelnen Testobjekte und einen ruhigen, möglichst lichtgeschützten Raum. Der Test umfasste je fünf Messreihen pro Objekt. Mit Hilfe eines Messbandes und der Software Bricx und NXT-View wurden diese Tests dann durchgeführt.
Die Umgebungslautstärke und das Umgebungslicht mussten konstant sein, da sonst ungenaue Messwerte auftraten. Leider war dies nicht immer möglich, was in den Messreihen manchmal deutlich wurde.
Für die Messung standen uns folgende Software zur Verfügung:

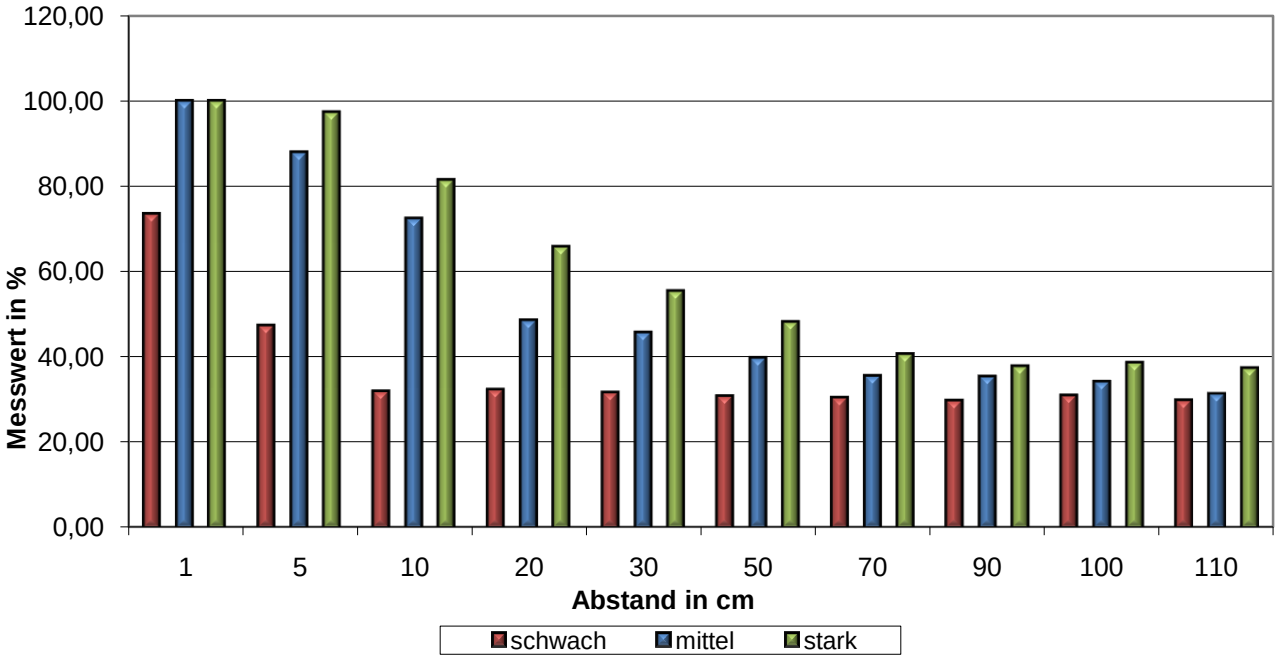
- Bricx
  - die Werte liegen zwischen 0 und 1023
  - mit diesem Programm haben wir hauptsächlich getestet, da dort gleich die Messreihen gespeichert und visualisiert werden konnten
- NXT View
  - Die Werte liegen zwischen 0 und 100 %
  - Beim Ultraschallsensor liegen die Werte in cm vor
  - Der Maximalwert 255/100% bedeutet, dass der Wert nicht korrekt gemessen werden kann
- NXT Software

Der Lichtsensor

Um die verschiedenen Lichtverhältnisse zu simulieren, benutzten wir drei verschiedene Lampen. Für diesen Test verwendeten wir die Software Bricx und rechneten anschließend die Werte in Prozent um.

Abbildung 15: starkes, mittleres und schwaches Licht

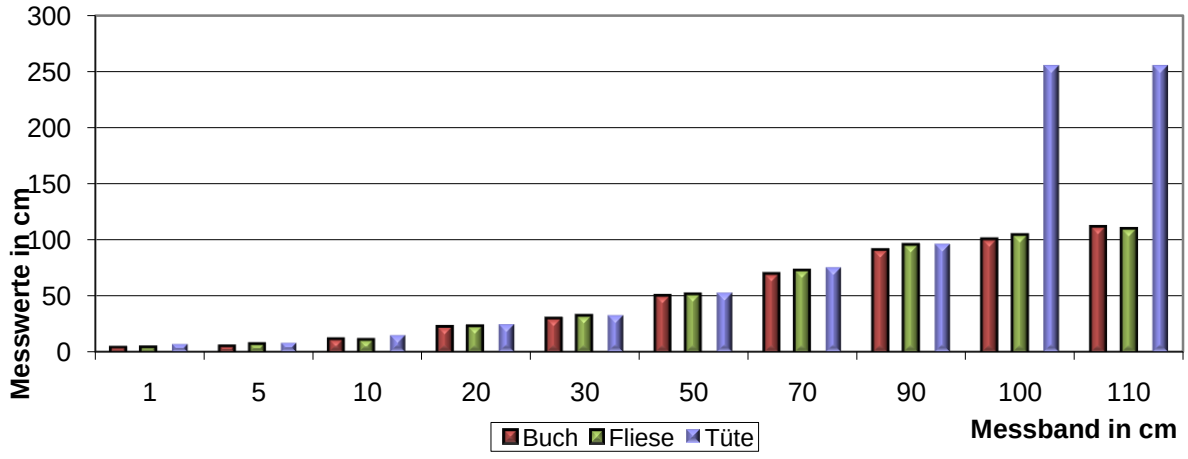
Bei einer sehr schwachen Lichtquellen wurde schon bei ca. 10 cm nur noch das Umgebungslicht wahrgenommen. Bei mittlerem und starken Licht geschah dieser Effekt erst nach ca. 50 cm. Wichtig: Das Umgebungslicht beeinflusste sehr stark das Ergebnis. Dadurch kam es zu Schwankungen bei den Messergebnissen.



Grafik 1: Unsere Messwerte zum Lichtsensortest

Der Ultraschall

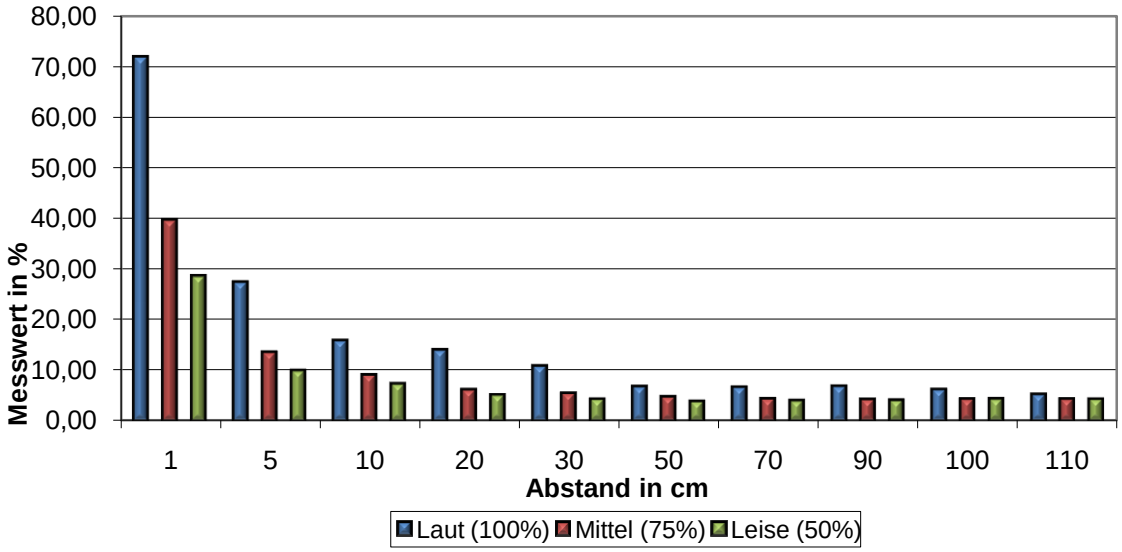
Der Test wurde mit einem Buch, einem Monitor (reflektierende Oberfläche) und einer Tüte (unebene Oberfläche) durchgeführt. Als Software verwendeten wir NXT-View. Der Wert 255 bedeutet, dass die Software keine genaue Entfernung mehr angeben kann. Besonders bei der Tüte kam es zu sehr starken Abweichungen in der Distanz bezogen auf die reale Entfernung, da die Tüte uneben war. Dies ist damit zu erklären, dass das ausgesendete Signal nicht immer zum Sensor zurückreflektiert wurde. Auch beim Monitor kam es nach ca. 90 cm zu starken Abweichungen. Die besten (genauesten) Werte traten beim Buch auf. Allerdings können Abweichungen verursacht werden, in dem das Buch etwas schräg gehalten wird.



Grafik 2: Ultraschalltest

Der Akkustiksensor

Auch hier wurde wieder die Software Bricx genutzt und die Werte in Prozent umgerechnet. Als Quelle benutzten wir die Notebook-Lautsprecher und einen konstanten 200 Hz Ton.
Auffällig war, dass der Akustiksensor sehr stark auf Umgebungsgeräusche reagierte und damit starke Schwankungen auftraten. Außerdem war ein sehr starker Abfall beim sehr lauten Ton, vom Abstand 1 cm zu 5 cm, und vom lauten Ton zu den beiden anderen zu bemerken. Ab ca. 10 cm waren beim mittleren und leisen Ton nur noch die Umgebungslautstärke zu bemerken.



Grafik 3: Messwerte unseres Akustiksensortests

3. und 4. Aufgabe

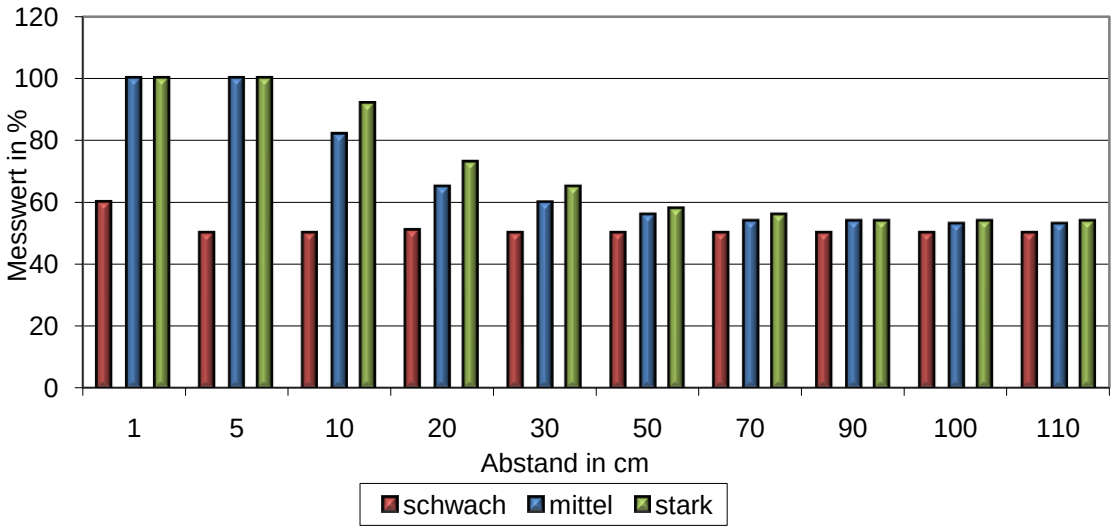
Da sich die Aufgabenstellungen drei und vier stark ähnelten, haben wir diese zusammengefasst.

Aufgabenstellung

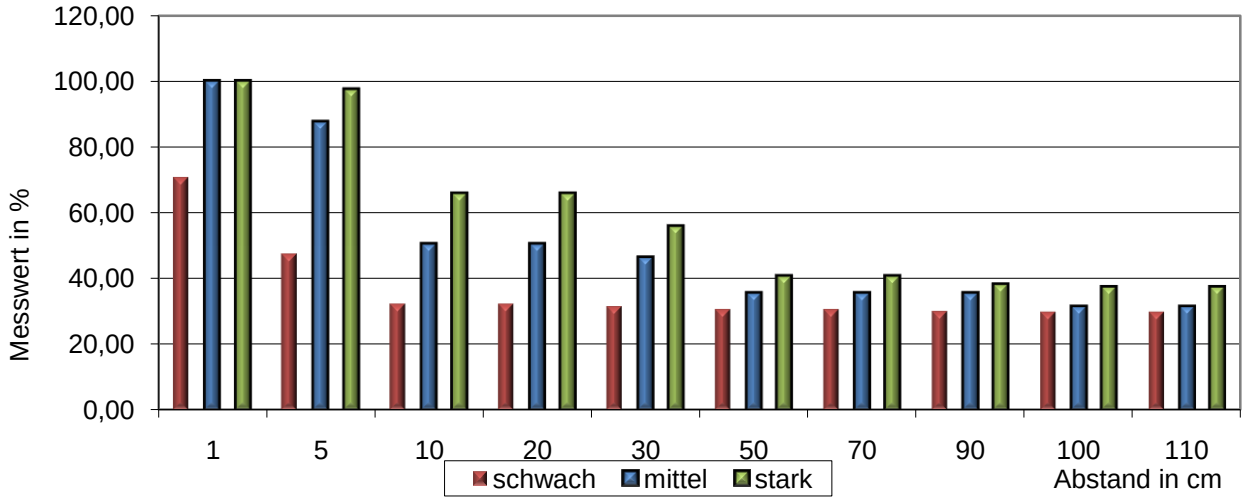
Nun war es unsere Aufgabe einen letzten Versuch zu wiederholen und dabei eine andere Software zu benutzen. Unsere Wahl viel auf den Lichttest und als Software NXT-View. Außerdem sollten wir erforschen, ab welcher Distanz Infrarot detektierbar war.
Für die praktische Aufgabe sollte ein aktiver oder passiver Greifer montiert werden, mit dem ein Ball zu einer Lichtquelle transportiert werden musste.

Test unter den neuen Bedingungen

Zur Durchführung dieses Tests hatten wir die interne Software des NXT, NXT-View, benutzt und zum Vergleich die Werte aus der vorhergehenden Aufgabenstellung, die mit Bricx gemessen wurden.
Dabei entstanden folgende Messwerte:



Grafik 4: Test mit NXT-View



Grafik 5: Test mit Bricx

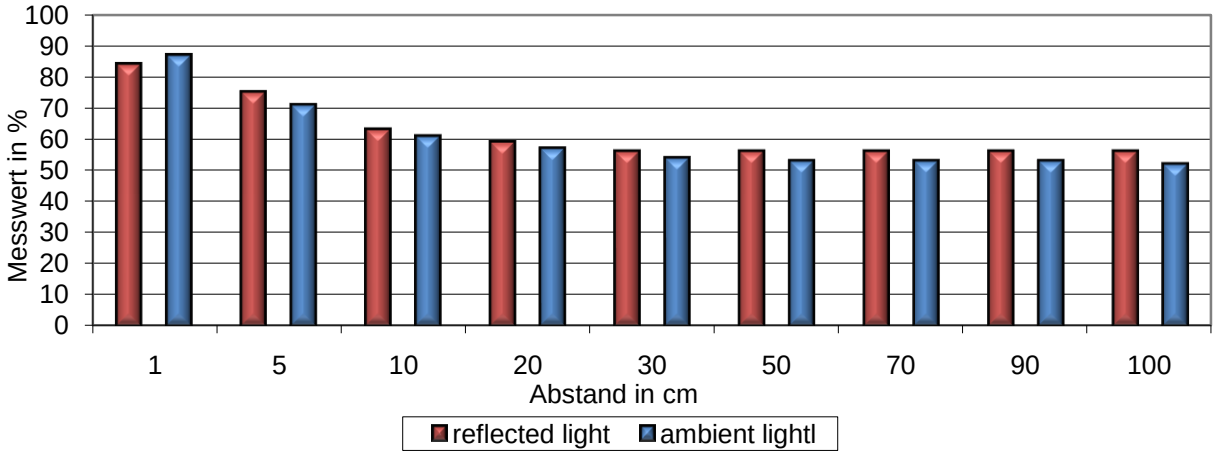
Beim ersten betrachten der Grafik erscheinen die Werte beider Tests ähnlich bis gleich zu sein. Allerdings fällt bei genauerem Betrachten auf, dass ab ca. 10 cm die Abweichungen bis zu 20 Einheiten (%) betragen. Diese Abweichungen können natürlich im Programmablauf erhebliche Fehler verusachen, falls diese Unterschiede nicht beachtet werden. Daraus folgernd sollte sich der Programmierer für eine der beiden Programme entscheiden um Werte für einen Sensor zu bestimmen.

Der Ball (Infrarottest)

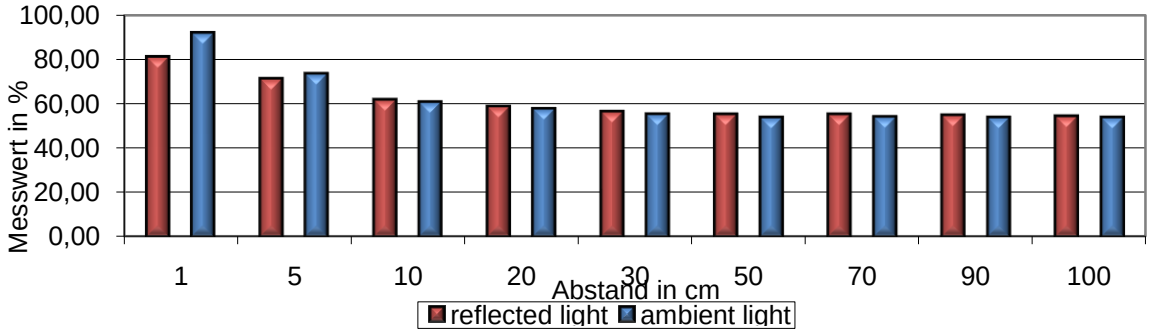
Der von uns durchgeführte Test fand in einem relativ hellen Raum statt. Daher kam es bei ambient light bereits ab einer Distanz von ca. 50 cm und bei refelcted light ab einer Distanz von ca. 30 cm keine effektiven Werte (nur noch Umgebungslicht) heraus. Es ist also bei der Raumwahl zu beachten, dass ein möglichst lichtgeschützter Raum benutzt werden sollte, da das Umgebungslicht sehr stark das Ergebnis beeinflusst.

Die Messwerte bei Bricx waren im Vergleich zum NXT-View annähernd gleich. Die Abweichungen bei ambient light zu reflected light traten wahrscheinlich aufgrund des helleren Umgebungslichts auf.

Folgende Werte ergaben sich bei unserem Test:



Grafik 6: Ballerkennung mit Lichtsensor: NXT-View



Grafik 7: Ballerkennung mit Lichtsensor: Bricx

Der Greifer

Die erste Überlegung für einen Arm, war eine Art Kippvorrichtung. Dies erwies sich jedoch als nicht vorteilhaft, da folgender Nachteil auftrat: Der Arm war zu schwer. Somit traten Schwierigkeiten bei niedrigem Batteriestand auf, vorallem beim hochheben. Daraus resultierend kamen Probleme mit dem Motor dazu. Bei Bewegungen des Roboters hatte der Ball zu viel Spielraum innerhalb des Arms. Daher kam es zu Ungenauigkeiten beim manövrieren.

Als zweite Version entschieden wir uns für einen Greifer, der vorn am Roboter befestigt wurde. Dies erforderte jedoch einen vollständigen Umbau des Roboters. Es mussten also die Motoren versetzt werden. Der NXT-Baustein und die Sensoren bekamen ebenfalls eine neue Position. Nun war es dem Roboter möglich beim direkten ansteuern des Balls ihn effektiv zu greifen und problemlos zu bewegen. Als Nachteil dieser Konstruktion folgte, dass der Roboter nun einen größeren Wendekreis erforderte, da die Arme den Roboter verlängerten. Dies musste wiederum bei der Programmierung beachtet werden.

Der Roboter

Nachdem die richtige Version für den Greifer gefunden war, mussten, wie schon erwähnt, einige Änderungen vorgenommen werden.

Da unser Roboter immer über den Rand schaute, bauten wir den Ultraschallsensor unter den Lichtsensor, anstatt darüber. Außerdem wurde der alte Lichtsensor aus dem RCX-Bausatz hinzugefügt, damit eventuelle Messfehler abgefangen wurden. Ein Sensor aus dem neuen Baukasten konnte leider nicht eingebaut werden, da er durch seine Größe in die Baumform nicht eingefügt werden konnte.

Nachdem der Ball und die Taschenlampe korrekt gefunden wurden, programmierten wir den NXT so, dass er sich an verschiedene Lichtwerte gewöhnen konnte. Diese Werte wurden vor dem eigentlichen Programmablauf gespeichert.

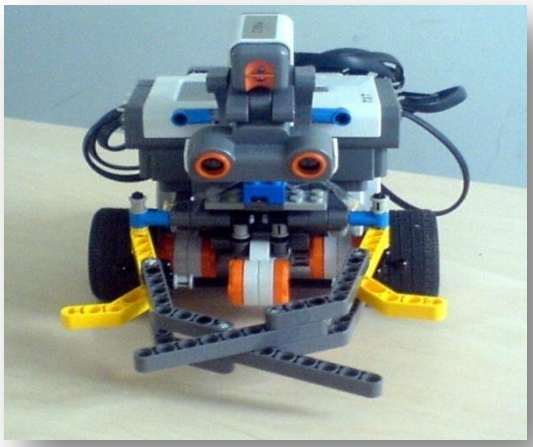


Abbildung 16: Unser Max2

Programmieransätze

Unsere Grundidee bei der Balldetektion war, dass der Roboter um sich selbst drehen sollte und den hellsten Punkte lokalisierte, diesen ansteuerte und den Ball dann griff.
Der erste Ansatz für die Rotation bestand aus einer rekursiven Verkleinerung des Rotationswinkels (360°, 180° und 45°) und bei jeder Rekursion fünf Messwerte zu bestimmen. Das Probleme war jedoch, dass die Rotation ungenau war, also manchmal zu weit vor oder nach dem Objekt gemessen wurde. Somit war auch die Ansteuerung ungenau. Hat der NXT also einen Punkt angesteuert, ist er trotzdem weit am Objekt vorbeigefahren.
Beim zweiten Ansatz verwendeten wir den ersten als Grundlage, allerdings wurden die Messabstände verkleinert. Er war jedoch auch dort zu ungenau bei der Rotation.
Unser finaler Ansatz bestand aus einer kontinuierlichen 360° Drehung, bei der er zu jeder Taktung ein Messwert speicherte. Mithilfe des Rotationssensors wurde die Position des maximalen Wertes genau bestimmt.
Die Ansteuerung war besonders bei niedriger Akkuleistung schwierig, da der NXT dann nicht korrekt geradeaus fuhr. Obwohl wir die Motoren möglichst synchron eingestellt hatten (OnFwdSync), liefen sie trotzdem nicht immer ganz synchron.

Wir verwendeten den selben Suchalgorithmus beim Ball und der Lampe, nur mit anderen Grenzwerten.
Nachdem der Ball und die Taschenlampe detektierbar waren, bearbeiteten wir den Hindernissmodus. Dabei verfolgten wir zwei Ansätze: Während man den Ball sucht, könnte parallel der Hindernissmodus ausgeführt werden. Das Timing wäre dabei jedoch schwierig, da die einzelnen Prozesse parallel ablaufen müssen. Der andere Ansatz war ein Wechsel zwischen Such- und Hindernissmodus, also ein sequentielles Abarbeiten. Aufgrund der Nachteile des parallelen Ablaufes entschieden wir uns für den sequentiellen Ablauf. Die Grundidee beim Hindernissmodus war, den Suchalgorithmus vom Licht zu adaptieren. Es wurden also kontinuierliche Messungen und Drehungen durchgeführt. Dabei waren unter bestimmten Umständen keine realistischen Werte ausgegeben worden. War z.B. ein Wert über dem Grenzwert, so traten im folgenden unerklärliche Messspitzen auf (Werte über den eigentlichen Wertebereich). Dies begründeten wir mit der Vermutung, dass sich der Roboter weitergredreht hatte,

bevor das ausgesendete Signal wieder beim Sensor angelangt war. Folgernd mussten Pausen zwischen den Messungen und Drehungen eingefügt werden.
Alle 20 Millisekunden wurde eine Drehung ausgeführt, dann nahm er den Messwert und speicherte diesen. Somit kamen nun vernünftige Werte heraus. Das Problem hier war jedoch, wenn der erste maximal gespeicherte Wert an einer Kante lag, fuhr der NXT an die Kante. Unsere Lösung war dabei, drei benachbarte Werte zu betrachten, um seinen „Sichtwinkel“ zu vergrößern. Es wurde nur der Messwert benutzt, wenn sich diese Werte in einem gewissen Rahmen ähnelten. Seltsam hierbei war, dass während der Messung spontan Spitzenwerte auftraten. Sobald eine Messung falsch wurde, waren alle Messungen danach über den eigenen Messbereich hinaus. Daher entschieden wir uns dafür, dass die Messwerte nur akzeptiert wurden, wenn der Maximalwert unter 255 lag, sonst wurde ein weiteres Mal gemessen.
Beide Modie wurden danach zusammengefügt. Dem Suchmodus wurde eine Bedingung hinzugefügt, die garantiert, dass er den Ball oder das Licht sah und nicht einfach einen hellen Punkt.
Beim Ralleymodus: Falls ein akzeptabler Wert gefunden wurde, fuhr er so lange geradeaus bis er zu nah an ein Objekt stand. Geschah dies, schaltet er in den Suchmodus.

Schlussendlich kann zusammengefasst werden, dass dies jedoch keine intelligente Programmierung darstellt. Der Roboter merkt sich nicht wo die Gegenstände stehen, sondern sucht solange bis er die Lichtquelle bzw. den Ball gefunden hat. Es unterlag also dem Zufall ob der den Ball fand oder nicht.

Zusatzaufgabe: NXT Mobile Application

Eine Zusatzaufgabe, bestand darin, den NXT mit einer externen Anwendung zu steuern. Wir entschieden uns für die Handysoftware.

Auf der Internetseite[1] kann für bestimmte Handytypen, die dort in einer Liste aufgeführt sind, das Programm heruntergeladen werden. Bei den aufgeführten Modellen sollte jedoch beachtet werden, dass das Programm bei anderen Handytypen trotzdem anwendbar sein kann, da es eine Java-Anwendung ist und somit theoretisch bei jedem Java-fähigen Handy funktionieren sollte.

Abbildung 17: NXT Mobile Application

Das Programm besteht aus drei Unterprogrammen. Als erstes „Remote Control“, mit dem der NXT via Handy bewegt werden kann, „Program Control“, dass das senden von Nachrichten und starten von Programmen ermöglicht und als letztes „Collected Data“, das Fotos und Daten auswählt.
Negativ ist uns jedoch dabei aufgefallen, dass es bei mehreren NXT oder WLAN in der Nähe zu Kommunikationsproblemen (Empfangsschwierigkeiten) kommen kann. Außerdem sollte, falls mehrere NXT benutzt werden, der Name des eigenen NXT geändert werden, da es sonst nicht klar ist, welcher der eigene ist.

[1] http://mindstorms.lego.com/Overview/Mobile%20Application.aspx

Fazit

Großen Zeitaufwand nahm die Programmierung und die Fehlerbehebung innerhalb des Programmes in Anspruch. Eine Herausforderung war ebenfalls die Aufgabenteilung zwischen den Teammitgliedern. Interne Probleme mit einem Teammitglied, das im Laufe der Arbeit ausstieg, haben gezeigt, dass vorallem die Verlässlichkeit zwischen den Teammitgliedern in einer Gruppenarbeit sehr wichtig ist.
Zurückblickend können wir sagen, dass das Softwarepraktikum den im Internet und in vorhergehenden Gesprächen vorgestellten Anforderungen entsprachen, die Betreuung sehr gut war und es uns vorallem sehr viel Spaß machte. Die praktische Arbeit am Roboter war eine gelungene Abwechslung zum eher theoretischen Teil der universitären Ausbildung.

Quellen

http://www.nxt-in-der-schule.de/
http://bricxcc.sourceforge.net/nbc/
http://www.informatik.uni-kiel.de/inf/von-Hanxleden/mindstorms/
http://www.ortop.org/NXT_Tutorial/html/essentials.html
http://ivs.cs.uni-magdeburg.de/EuK/lehre/sopras/legoss07/Lego_nxt_info.pdf
http://mindstorms.lego.com/Overview/Mobile%20Application.aspx

Abbildungsverzeichnis

Grafikverzeichnis