Der Einsatz von Lego-Mindstorms im Informatikunterricht der 11. Klasse an der Leonard-Bernstein-Oberschule (Gymnasium)

Darstellungsschwerpunkt:

Sicherung und Transfer grundlegender algorithmischer Strukturen mit NQC

Schriftliche Prüfungsarbeit im Rahmen der Zweiten Staatsprüfung
für das Amt des Studienrats

vorgelegt von Rafael Schreiber
2. Schulpraktisches Seminar im Bezirk Hellersdorf (S)

Berlin, den 13.4.2004

Inhaltsverzeichnis

1 Einleitung

„LEGO-Roboter? Ist das nicht Spielzeug?“, fragten mich Freunde und Kollegen mit ironischem Unterton, wenn ich über den Inhalt meines Unterrichts sprach. Einmal abgesehen davon, dass diese Frage verallgemeinernd Spiel und Lernprozess als gegensätzlich darstellt, macht sie doch auch deutlich, was der Einsatz aller neuen Unterrichtsmittel als Erstes verlangt: Seine eigene Rechtfertigung und die Kennzeichnung seiner Funktionalität für den Unterrichtseinsatz, sowohl prinzipiell als auch im Detail.

In dieser Arbeit soll die Funktionalität des LEGO-Mindstorms-Systems für den Informatikunterricht betrachtet werden, indem sein in einer 11. Klasse innerhalb einer Unterrichtsreihe erfolgter Einsatz didaktisch begründet, dokumentiert und reflektiert wird. Dabei bedeutet Funktionalität Eignung oder zumindest Nutzbarkeit im Hinblick auf das zu erreichende Ziel:

In der vorliegenden Arbeit soll untersucht werden, inwieweit LEGO-Mindstorms in Verbindung mit der Programmiersprache NQC dazu geeignet ist, einen Kontext herzustellen, in dem die Schülern die im informatischen Anfangsunterricht mit der Lern- und Programmierumgebung *Robot Karol* erlernten grundlegenden algorithmischen Strukturen wiederfinden und anwenden können. Diese Anwendung des Gelernten wird, wie noch zu begründen ist, als Transfer verstanden, welcher gegenüber dem Begriff der Sicherung schwerpunktmäßig betrachtet werden wird. Im Fokus der Beobachtungen dieser Arbeit stehen insofern hauptsächlich jene Leistungen der Schüler, welche den Transfer, d.h. die Rekonstruktion von Wissen und Können, betreffen, und zwar im Spannungsfeld des Vorwissens der Schüler, der neu zu erarbeitenden Inhalte und der Unterrichtsmaterialen LEGO-Mindstorms und NQC.

Es gibt von vornherein gewisse Erwartungen, die von meiner Seite mit dem Unterrichtseinsatz von LEGO-Mindstorms und der Programmiersprache NQC verbunden sind. Das betrifft insbesondere den Aspekt der Schülermotivation, mithilfe dessen der Lernprozess intensiviert und Lernerfolg in besonderem Maße erreicht werden könnte. Der Grund für diese Erwartung ist, dass die Arbeit mit LEGO-Mindstorms eine über den schulischen Bezug hinausgehende lebensweltliche Relevanz hat und auf konkret fassbare Probleme verweist. Schließlich fliegt sogar auf der internationalen Weltraumstation ISS ein LEGO-Mindstorms-Roboter mit.[1]

[1] Im November 2001 wurde der von den deutschen Hobbybastlern Konrad und Bastian Schwarzbach aus Krefeld entwickelte LEGO-Mindstorms-Roboter „Jitter“ nach dem Gewinn eines Wettbewerbs zur ISS gebracht und dort erfolgreich als Sammler herumfliegender Kleinteile eingesetzt. Quelle: http://ais.gmd.de/de/pm/011012.html (19.3.04)

2 Theoretische Grundlagen

In diesem Kapitel werden die für diese Reihe wichtigen Unterrichtsmaterialien LEGO-Mindstorms und NQC vorgestellt sowie ihre inhaltlichen und didaktischen Einsatzmöglichkeiten mit Blick auf das Unterrichtsvorhaben untersucht.
Darüber hinaus sollen die für die Konzeption der Unterrichtsreihe tragenden didaktischen Grundbegriffe *Sicherung* und *Transfer* erläutert werden. Orientiert an Hans AEBLIS moderner Formalstufentheorie und kritisch gewürdigt unter Beachtung anderer Didaktiker und Lernpsychologen wird die Bedeutung von Sicherung und Transfer für den Lernprozess gekennzeichnet.
Dem folgend wird der Einsatz von LEGO-Mindstorms zu Sicherung und Transfer in Bezug gesetzt und seine Funktionalität im Hinblick darauf analysiert. Zuletzt wird daraus die grundsätzliche didaktisch-methodische Vorgehensweise abgeleitet.

2.1 Die Unterrichtsmaterialien LEGO-Mindstorms und NQC

2.1.1 LEGO-Mindstorms und der Roboterbegriff

Die mit dem eigentlich als Spielzeug entwickelten LEGO-Mindstorms-System zusammengebauten Geräte werden allgemein mit der Bezeichnung „Roboter" versehen. Ich möchte hier kurz den so oft verwendeten Begriff des Roboters in Bezug auf LEGO-Mindstorms einordnen.

Es existieren vielfältige Definitionen für den Roboter, von „*elektronisch gesteuerter Automat*"[2] über „*Apparaturen menschlicher Gestalt*"[3] bis hin zu „*programmgesteuerte Maschinen*"[4]. Eine mir brauchbar scheinende Definition bietet die MS Encarta 2002, 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.*"[5] Wie im Folgenden zu sehen sein wird, treffen zumindest die ersten drei Attribute, nämlich selbstständig, programmierbar und elektromechanisch, auf LEGO-Mindstorms zu.

LEGO-Mindstorms ist das Ergebnis einer Zusammenarbeit der dänischen Firma LEGO, die seit 1949 Spielsteine zum Zusammenstecken produziert und dem Massachusetts Institute of Technology (MIT). Am MIT erforschte die Arbeitsgemein-

[2] SCHOLZE-STUBENRECHT u. a. (Hrsg.): *Duden. Die deutsche Rechtschreibung.* Dudenverlag Mannheim 2001.
[3] SCHOLZE-STUBENRECHT u. a. (Hrsg.):*Duden. Fremdwörterbuch*. Dudenverlag Mannheim 1999.
[4] ENGESSER, H. (Hrsg.): *Duden Informatik. Ein Sachlexikon für Studium und Praxis*. Dudenverlag Mannheim 1993.
[5] MICROSOFT (Hrsg.): *Encarta Enzyklopädie*. Microsoft 2002.

schaft „Epistemology and Learning-Group“ (Erkenntnistheorie und Lernen) seit den späten siebziger Jahren unter Führung von Seymour PAPERT den Einsatz von Computern für das Lernen.

Aufbauend auf PAPERTS Buch „*Mindstorms: Children, computers and powerful ideas*“[6] (1980) und der darin vorgestellten Programmierumgebung „Logo“ wurde am MIT ein „Roboterlabor“ eingerichtet mit dem Ziel, einen autonomen Roboter für Lernzwecke herzustellen. Man integrierte dazu einen Mikroprozessor inklusive Speicher, Ein- und Ausgängen sowie Display in einen Baustein, den so genannten „Robot Command Explorer“ (RCX), salopp „Brick“. Dieser RCX kann mittels Infrarotschnittstelle eine Verbindung zu einem Computer aufbauen und über diesen programmiert werden. LEGO unterstützte derweil das Projekt finanziell.

Abb.1 LEGO-Mindstorms-Roboter

Am 27.1.1998 wurde in London von LEGO das erste *LEGO Mindstorms Robotics Invention System* (RIS) vorgestellt, ein Baukasten, „*der zusätzlich zu dem RCX noch 700 Lego-Bauteile, zwei Motoren, zwei Berührungssensoren, einen Helligkeitssensor eine Infrarotschnittstelle* [und] *ein Handbuch*“[7] enthielt sowie die auf Seymour PAPERTS „Logo“ basierende grafische Programmierumgebung *RCX-Code*. Im *RCX-Code* werden Programme durch Aneinanderfügen grafischer Blöcke aufgebaut, die jeweils Anweisungen bzw. Kontrollstrukturen darstellen.

Mit dem RIS ist es also möglich, einen Roboter nach dem LEGO-Prinzip zusammenzubauen, der sich im Raum frei[8] bewegen, entsprechend der verbauten Sensoren die Umwelt wahrnehmen und je nach Programmierung darauf reagieren kann. Insofern ist die Einsatzfähigkeit eines LEGO-Roboters auch direkt an die verfügbaren Sensoren gekoppelt. Ebenfalls in der dem RIS 1.0 im Jahre 2001 folgenden Edition 1.5 und in der aktuellen Version 2.0 sind nur Berührungs- und Helligkeitssensoren enthalten. Über LEGO können jedoch separat weitere Sensoren bezogen werden, z.B. Rotations- oder Temperatursensoren. Zudem gibt es sowohl kommerzielle Hersteller von Sensoren (z.B. Mindsensors[9]) als auch viele private Bastler[10]. Hauptproblem dabei ist die Beschränkung der Sensoreingänge am RCX

---

[6] PAPERT, S.: *Mindstorms: Children, computers and powerful ideas*. Basic Books New York 1980.
[7] ABEND, M.: Robotik und Sensorik. Selbständige Entwicklung „unscharfer“ Algorithmen zur räumlichen Orientierung unter Verwendung des Lego-Mindstorms-Systems. Schriftliche Prüfungsarbeit zur zweiten Staatsprüfung für das Amt des Studienrates. Berlin 2001, S.8.
[8] Das heißt ohne Verbindung über Kabel oder Funk zu einer Steuereinheit während der Laufzeit.
[9] Http://www.mindsensors.com .
[10] Vgl. die Webseite von M. GASPERI: http://www.plazaearth.com/usr/gasperi/lego.htm (19.3.04).

auf drei, wobei mithilfe von so genannten Multiplexern auch diese konstruktionsbedingte Einschränkung bis zu einem gewissen Punkt umgangen werden kann.
Der Eigenbau von Sensoren durch Nutzer zeigt die große Akzeptanz, die LEGO-Mindstorms erfahren hat. Ursprünglich für Kinder ab elf Jahren konzipiert, erwarb sich Mindstorms schnell eine große Fangemeinde auch unter Erwachsenen.

2.1.2 Die Programmiersprache NQC

Einer der Hauptgründe für den Erfolg von LEGO-Mindstorms war die frühzeitige Offenlegung der Programmierschnittstellen durch LEGO. Dies zog Tüftler und Programmierer an und führte zur Umsetzung unterschiedlicher Programmiersprachen für Mindstorms. Auf technischer Ebene war außerdem durch die Speicherung der Firmware im RAM auch deren Ersetzung möglich, was für die Umsetzung von Interpretersprachen wie Java[11] oder Forth[12] auf Mindstorms von entscheidender Bedeutung war.

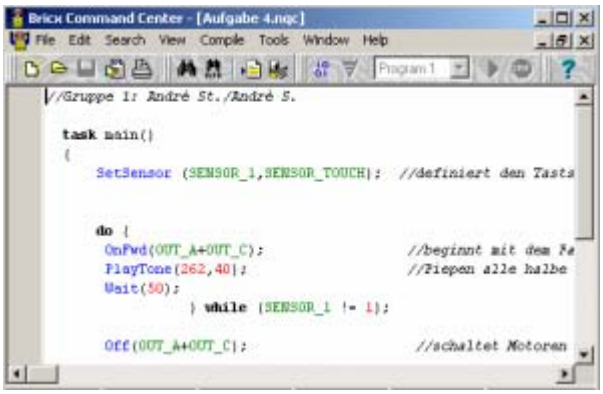

Abb.2 Bricx mit NQC-Quellcode

Eine der für Mindstorms entwickelten Programmiersprachen war das von Dave BAUM entwickelte NQC (Not Quite C). Es ähnelt der verbreiteten Programmiersprache C und ist plattformunabhängig[13]. Das Aufsetzen auf der standardisierten LEGO-Firmware sorgt dabei für weitgehende Stabilität und Zuverlässigkeit.

Außerdem bietet NQC durch die bessere Ressourcennutzung des RCX einen größeren Funktionsumfang als der mitgelieferte RCX-Code, so z.B. Möglichkeiten zum prioritätsgesteuerten Hardwarezugriff oder zur ereignisgesteuerten Programmierung[14]. Nicht zuletzt gibt es von Dave BAUM eine sehr gute Dokumentation für NQC[15] und von Marc OVERMARS eine hervorragende Entwicklungsumgebung

---

[11] Zum LeJOS -Projekt (Java für Lego) führt momentan folgender Link: http://www. lejos.sourceforge.net (19.3.04).
[12] Zum PbForth-Projekt (Forth für Lego) führt momentan folgender Link: http://www.hempeldesigngroup.com/lego/pbForth/homePage.html (19.3.04).
[13] Interviewer: „*Why did you decide to make NQC?*" Dave BAUM: *„I wanted to do my programming on the Macintosh*" (http://www.lmsm.info/back-issues/1102/Dave_Interview.html; 19.3.04).
[14] Vgl. STOLT, M.: *Roboter im Informatikunterricht* (Studienarbeit). 2001, S.22. Quelle: http://www.webniks.de/websites/www.buechergilde-hamburg.de/stolt/lego/ (19.3.04).
[15] Der von BAUM verfasste „NQC Programmer's Guide" ist momentan noch auf seiner Webseite http://www.baumfamily.org/nqc_old/index.html (19.3.04) zu beziehen. BAUM hat sich jedoch aus der NQC-Entwicklung zurückgezogen und die Weiterentwicklung der Open-Source-Gemeinde überlassen. Er kündigt an, seine Webseite in Kürze schließen zu wollen (ebd.).

namens „Bricx Command Center“. Diese bietet z.B. Syntax-Highlighting[16], eine Prozedur- bzw. Funktionsübersicht und zeilengenaue Fehleranzeige. Nicht zuletzt sind über NQC schon mehrere Publikationen erschienen.[17]

Allerdings hat NQC auch Nachteile: „*NQC unterliegt noch sehr vielen Einschränkungen, wie zum Beispiel einer sehr begrenzten Variablen- und Datentypenanzahl[18], 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.*“[19,20]

Die zitierten Nachteile sind prinzipieller technischer oder rechtlicher Art und stellen somit meines Erachtens nur für Entwickler eventueller professioneller und aufwändiger LEGO-Anwendungen Einschränkungen dar, nicht aber für den Schuleinsatz und insbesondere nicht für die beabsichtige Thematisierung der grundlegenden algorithmischen Strukturen.

2.1.3 Didaktische Hintergründe zu LEGO-Mindstorms

LEGO-Mindstorms ist ein relativ junges Produkt, welches aber schnell das Interesse von Lehrern in Bezug auf seine Einsatzfähigkeit im Informatikunterricht weckte. Der Grund dafür ist einerseits die Vielseitigkeit der möglichen Anwendungen im Unterricht, andererseits der zu erwartende hohe motivationale Effekt für die Schüler.

Als mögliche Inhalte für den Informatikunterricht mit LEGO-Mindstorms nennen HIRSCH, MAGENHEIM und REINSCH[21]:

1) Die Modellierung eines Informatiksystems: Gemäß den Autoren könne hier das sich gegenseitige Bedingen von Hard- und Software gezeigt werden, z.B. durch den Bau eines Lego-Roboters und dessen Programmierung hinsichtlich einer speziellen Aufgabe.

---

[16] Syntax-Highlighting ist die farbliche Herausstellung von Schlüsselwörtern oder ganzen Strukturen in einem Programm-Code. Es dient der Richtigschreibung und dem Überblick.

[17] Z.B. KNUDSEN/NOGA: *Das inoffizielle Handbuch für LEGO-Mindstorms-Roboter*. O’Reilly-Verlag Köln. 2000.

[18] Der RCX des RIS 2.0 erlaubt nur die Datentypen Integer und Boolean bei maximal 48 Variablen.

[19] FILIUS, S.; KÜHN, S., SCHELLHAMMER, S.: *Einführung in Lego Mindstorms*. 2004, S.17. Quelle: http://wwwmath.uni-muenster.de/info/u/lammers/EDU/ws03/Landminen/Abgaben/Gruppe1/Ausarbeitung.pdf (31.3.04).

[20] NQC selbst ist jedoch Open-Source und unterliegt der Mozilla Public License (vgl. http://bricxcc.sourceforge.net/nqc/; 19.3.04).

[21] HIRSCH, M.; MAGENHEIM, J.; REINSCH, T.: „*Zugänge zur Informatik mit Mindstorms.*“ In: LogIn 20 (2000) S.34-46.

2) Die Vermittlung informatischer Prinzipien: Damit meinen die Autoren beispielsweise die Einführung einfacher Kontrollstrukturen und die Erarbeitung elementarer Algorithmen anhand der Steuerung des LEGO-Roboters. „*Mit fortgeschrittenen Lerngruppen bieten sich auch vertiefende Betrachtungen an, etwa über die Effizienz der Steuerungsalgorithmen ...*“ (Ebd., 39)

3) Die Einübung der objektorientierten Sichtweise: Die Autoren schlagen vor, dass der LEGO-Roboter als erzeugtes Objekt auf der Basis eines auf dem Papier gefertigten Entwurfs als affektiver Einstieg in die objektorientierte Denkweise dienen könne (vgl. ebd., 37 ff.).

HIRSCH, MAGENHEIM und REINSCH unterteilen die Arbeit mit LEGO-Mindstorms in zwei grundsätzliche Phasen, den physikalischen Zusammenbau der Roboter und die Programmerstellung (vgl. ebd., 35). Dabei problematisieren sie die Montage von LEGO-Mindstorms-Robotern durch die Schüler im Rahmen des Unterrichts aufgrund des erforderlichen Aufwandes: „ *... umso klarer haben uns auch unsere unterrichtspraktischen Erfahrungen gezeigt, dass dies* [die Montage] *nur unter einem erheblichen Betreuungsaufwand* [...] *zu realisieren ist. Zudem ist ein Zeitbedarf von mehreren Stunden zu berücksichtigen.*“ (Ebd., 38)
Auf der anderen Seite stellen HIRSCH, MAGENHEIM und REINSCH die hohe motivationale und mediale Qualität von LEGO-Mindstorms-Robotern heraus. „*Sie genügen ... einerseits Kriterien wie Altersgemäßheit, Bezug zum Erfahrungshorizont der Schülerinnen und Schüler, Wissenschaftsorientierung und Zukunftsbedeutung. Andererseits repräsentieren sie in den Erfahrungen vieler Schülerinnen und Schüler auch einen am Spiel orientierten ... Freizeitkontext.*“ (Ebd., 37)
Was das für den Unterricht bedeuten kann, wird bei Michael ABEND deutlich, der für seine Examensreihe ebenfalls LEGO-Mindstorms einsetzte: „*Sie* [die Motivation] *und die Arbeitsatmosphäre im Unterricht übertrafen meine Erwartungen erheblich. Von der ersten bis zur letzten Stunde ... konnte ich keine Ablenkung mit anderen Dingen ... registrieren.*“ (ABEND, 41)
Aufgrund dieser Erfahrungen lässt sich ein vielfach erfolgreicher und unter verschiedenen Schwerpunkten möglicher Einsatz des Unterrichtsmaterials LEGO-Mindstorms konstatieren.

2.2 Sicherung und Transfer als didaktische Artikulationsstufen

Lernen wird in der Lernpsychologie als Prozess verstanden, der zu „*einer relativ dauerhaften Veränderung einer Verhaltens- und Erkenntnisstruktur eines Menschen* [führt]*, die durch die Auseinandersetzung dieses Menschen mit seiner natürlichen und soziokulturellen Umwelt erfolgt, d.h. weder organisch noch ausschließlich reifungsbedingt ist.*“[22] Es ist die Aufgabe von Unterricht, für diese Art von Auseinandersetzungen einen Rahmen zu schaffen, sie zu ermöglichen und zu steuern. Dabei ist die Unterrichtssituation an sich schon eine künstliche, denn zum einen ist sie durch ihre Kennzeichnung als solche per se unnatürlich, zum anderen wird hier vom Lernenden eine Auseinandersetzung mit Gegenständen gefordert, die nicht unbedingt zu seinen persönlichen Alltagserfahrungen gehören.

Auch aufgrund dieser Diskrepanzen orientiert sich die Lern- und Lehrtheorie an den psychologischen Erkenntnissen zum Ablauf des Lernens. Diese wurden erstmals von Johann Friedrich HERBART Mitte des 19. Jahrhunderts in vier so genannten Formalstufen zusammengefasst, welche die prinzipielle Form jeden Erkenntniserwerbs darstellen. Die Formalstufentheorie wurde von Tuiskon ZILLER auf die Unterrichtspraxis übertragen und ergänzt.[23] HERBARTS und ZILLERS Formalstufen als Unterrichtsprinzip waren und sind sehr umstritten, insbesondere wegen ihrer zeitweise schematischen, ja dogmatischen Anwendung in der Praxis. In abgewandelter und ergänzter Form bilden sie jedoch weiterhin die Basis für die Struktur des Unterrichtsprozesses. Im Folgenden soll diese Struktur in Anlehnung an Hans AEBLI erläutert werden, da aus ihr die Bedeutung der Grundbegriffe Sicherung und Transfer direkt hervorgeht.

AEBLI nennt seine Darstellung eine „*moderne Formalstufentheorie*“[24]. Als zentrales Motiv und Ausgangspunkt des Lernens sieht er das **Problemlösen**, basierend auf der Annahme, „*dass die Strukturen unseres Denkens und Wahrnehmens eine Tendenz zur Geschlossenheit (closure) haben: wir möchten die Lücke schließen, Zusammenhang, Kohärenz herstellen, unseren Denkfiguren und Wahrnehmungsgestalten die innere Geschlossenheit sichern.*“ (Ebd., 280)
Das Lösen eines Problems ist jedoch nur ein erster Schritt zur oben angesprochenen erwünschten dauerhaften Veränderung einer Verhaltens- und Erkenntnisstruktur, denn noch ist die Problemlösung an das konkrete Problem gebunden. Erforderlich ist eine Systembildung und Abstraktion, die durch die Ausbildung von Handlungsplänen, Begriffen und Operationen (AEBLI) erreicht wird. Diese Hand-

[22] MESSNER, H.: Wissen und Anwenden. Zur Problematik des Transfers im Unterricht. Klett Stuttgart 1978, S.44.
[23] Zu HERBARTS und ZILLERS Formalstufentheorie vgl. MÜßENER, G. (Hrsg.): *Didaktische Texte zu Unterricht und Erziehung in Wissenschaft und Schule*. Deimling Wuppertal 1991.
[24] AEBLI, H.: *Zwölf Grundformen des Lehrens*. Klett Stuttgart 1985, S.24.

lungspläne, beschrieben mit Hilfe von Begriffen und ausgeführt als Operationen, sind im Gegensatz zur Problemlösung im ersten Lernschritt bereinigt von Seiteneffekten und Irrwegen, bieten aber gleichzeitig genug Raum für die Variation der Mittel zur Problemlösung oder der Zwischenziele. Das Ziel dieser Lernphase des so genannten „**Durcharbeitens**“ nennt AEBLI „*Beweglichkeit des Handelns*“ (ebd., 311ff.) und diese Beweglichkeit meint das Loslösen der Problemlösungsstrategie vom konkreten inhaltlichen Problem.

Der zweiten Lernphase muss eine dritte folgen, die die erworbenen Begriffe und Operationen festigt: **Üben und Wiederholen**. „*Das Üben dient der Automatisierung von gedanklichen und praktischen Abläufen*.“ (Ebd., 326) Der Übungsbegriff wird in der Literatur nicht überall eindeutig vom Wiederholungsbegriff abgegrenzt. EISENHUT unterscheidet die Begriffe insofern, als dass Automatisierung durch Wiederholen erreicht wird, dabei Verhalten und Wissen auf gleich bleibender Ebene fixiert und Gehirnkapazitäten freigesetzt werden.[25] Die Übung dagegen sei „*ein Verbessern, Festigen ... bis zu einer optimalen Leistungsgrenze*“ (ebd., 26). Bei diesem Verständnis des Übungsbegriffs liegt der Fokus stärker auf einer Progression und dem Prozess als auf dem Ergebnis. Lernpsychologisch „*... wirkt* [der Erfolg in Übungssituationen] *als Verstärker (re-inforcement). Er erhöht die Wahrscheinlichkeit für die Wiederholung der Reaktion in einer vergleichbaren Situation*“ (ebd., 27).
Diese Verstärkung kann allerdings nur bei Übungserfolg eintreten. Die Frage, wie dieser garantiert werden soll, führt zu grundsätzlichen Anforderungen an erfolgreiche Übungen. Manfred BÖNSCH spezifiziert einige dieser Anforderungen in Form so genannter „Gesetze“, welche die verschiedenen Anforderungen an Übungen und damit auch an einen „Übungsleiter“ bzw. Lehrer deutlich machen[26]:

1) Thorndikes Frequenzgesetz: Eine höhere Anzahl an Wiederholungen bewirkt einen höheren Übungserfolg.
2) Das Gesetz des Formwechsels: Bloße Wiederholung ist ermüdend. Übungsreichtum und neue Zusammenhänge sind für den Übungsgegenstand essentiell.
3) Das Gesetz der Reinhaltung: Es ist unbedingt darauf zu achten, dass sich beim Üben keine Fehler einschleichen, die mitgelernt werden.
4) Das Jostsche Verteilungsgesetz: Die Übungen müssen quantitativ richtig verteilt sein, am Besten regelmäßig über einen längeren Zeitraum.

[25] Vgl. EISENHUT, G.; HEIGL, J.; ZÖPFL, H.: Üben und Anwenden. Zur Funktion und Gestaltung der Übung im Unterricht. Klinkhardt Bad Heilbrunn 1981, S.27.
[26] Siehe BÖNSCH, M: *Üben und Wiederholen im Unterricht*. Ehrenwirth München 1988, S.34 ff.

5) Das Gesetz der Bereitschaft: Die Schüler müssen die Notwendigkeit der Übung einsehen. Sie müssen motiviert sein.
6) Das Gesetz des Erfolges: Der Lehrer muss die Übung so gestalten, dass ein Übungserfolg möglich ist.
7) Das Gesetz der Begabung: Begabte Schüler haben schnelleren Übungserfolg, der aber unter Umständen auch schneller wieder verloren gehen kann. Auch bei begabten Schülern muss auf Beständigkeit der Übung geachtet werden.
8) Das ontogenetische Übungsgesetz: Kinder lernen langsamer, behalten aber Gelerntes besser als Erwachsene.

(Nach BÖNSCH 1988, 34 ff.)

Diese „Gesetze“[27] findet man in ähnlicher Art und Weise bei EISENHUT als Übungsprinzipien (EISENHUT, 41-71) und auch bei AEBLI in den „Allgemeinen Regeln zur Gestaltung der Übungsarbeit“ (AEBLI, 339 ff.) wieder.

Insgesamt ist Üben also ein komplexer Prozess, der nur bei richtiger Durchführung die Dauerhaftigkeit ausgebildeter kognitiver Strukturen erreichen kann. Somit ist er auch für die von MESSNER oben angeführte Lerndefinition zentral. AEBLI bezeichnet das so: „*Durcharbeiten ... verfeinert das Gewebe der Begriffe und Operationen und macht sie durchsichtig. Üben und Wiederholen jedoch macht sie robust und solide. Es kon-solid-iert sie.*“ (AEBLI, 326)

Diese Aussagen AEBLIS zum Üben und Wiederholen führen zum Kern des Begriffes der **Sicherung.** Sicherung des Lernerfolges ist das Ergebnis von Übung und Wiederholung, also die dauerhafte Verfügbarkeit des Verstandenen. In diesem Sinne verwende ich den Begriff im Weiteren.

Folgt man AEBLI weiter bei der Darstellung seiner modernen Formalstufentheorie, schließt an Übung und Wiederholung die Phase[28] der **Anwendung** an. Die vom Schüler erworbenen und gefestigten „*Handlungspläne, Begriffe und Operationen*“ seien „*Instrumente zur Bewältigung von neuen Problemen ...*“ (AEBLI, 353). Diese Instrumente dienen der Fähigkeit zum Handeln in einer Welt, deren Erscheinungen und Gegenstände der Lernende laut PIAGET erst assimilieren, d.h. sich einverleiben muss[29]. Das bedeutet eine Anwendung der vom Schüler erworbenen Begriffe und Operationen auf einen neuen Gegenstand. Diese Anwendung ist als Versuch einer Erkenntnis zu verstehen, bei dem Begriffe und Operationen nicht sofort optimal passen werden, aber neue Aspekte enthüllen können und andere Begriffe

[27] Bönsch verwendet selbst Anführungszeichen.
[28] In dem Zusammenhang werden Phasen auch „Artikulationsstufen“ genannt.
[29] Vgl. PIAGET, J.: *Psychologie der Intelligenz*. Klett Stuttgart 1980 (orig.1947).

und Operationen in ihrer Anwendung plausibel machen: „*Und so geht es hin und her zwischen Subjekt und Objekt, zwischen Betrachter und Gegenstand*“ (AEBLI, 357). Der Schüler wählt aus einem Fundus von Begriffen und Operationen, bis der Gegenstand erschlossen oder der Fundus erschöpft ist.

Insofern erschließt der Schüler das Problem multiperspektivisch und sukzessive. Aus meiner Sicht liegt hier der qualitative Unterschied zwischen Anwenden und Üben, da bei Letzterem die Begriffe und Operationen, die gewählt werden müssen, durch den sachstrukturellen Zusammenhang der Übung mehr oder weniger vorgegeben sind. Beim Anwenden soll es zu einer *Rekonstruktion* der in der Lernsituation aufgebauten Struktur kommen.

MESSNER unterscheidet dabei folgende Situationen:

REKONSTRUKTION	**unter vertrauten Bedingungen**	**unter neuen Bedingungen**
in unveränderter Form	Reproduktion	Anwendung (Transfer)
in veränderter Form	Transformation	Anwendung (Transfer)

(Kategorienschema von MESSNER, 53)

An diesem Schema lässt sich gut erkennen, was Anwenden im Gegensatz zum Üben bedeutet: Die Rekonstruktion erlernter Strukturen unter neuen Bedingungen. Neue Bedingungen können inhaltlich, sozial oder medial sein. „*Lern- und Anwendungssituationen können sich in verschiedener Hinsicht unterscheiden - **inhaltlich, sozial oder medial** -, entsprechen sich jedoch in struktureller Hinsicht ... Die Verschiedenheit zwischen Lern- und Anwendungssituation ist demnach immer nur eine äußere, die ihre strukturelle Ähnlichkeit nicht berührt.*“ (MESSNER, 58)

Die Anwendung ist in MESSNERS Schema begrifflich nicht vom Transfer getrennt. Beides betrifft den Aufbau von Strukturen unter neuen Bedingungen.

AEBLI dagegen hält die beiden Begriffe auseinander, um den oben erklärten Wechselprozess zwischen Betrachter und Gegenstand als aktiven und eine gewisse Zeit dauernden Prozess zu markieren (vgl. AEBLI, 360 f.). Er spricht von Anwendung, nicht von Transfer, weil er den Transferbegriff behavioristisch geprägt sieht (vgl. ebd.), meint aber dasselbe wie MESSNER: **Transfer** ist das Anwenden erlernter Strukturen unter neuen Bedingungen.

MESSNERS Schema macht die inhaltliche Nähe von Sicherung und Transfer deutlich. Dabei stellt er klar, dass es „*zwischen den einzelnen Kategorien der beiden Dimensionen ...* [nur] *graduelle Übergänge* [gibt]“ (MESSNER, 58). Der grundsätzliche inhaltliche Zusammenhang von Sicherung und Transfer besteht in der bereits

erwähnten Rekonstruktion von Strukturen, bei Sicherung unter vertrauten, bei Transfer unter neuen Bedingungen.

Die Verbindung dieser beiden Artikulationsstufen wird unterstützt durch ROSENBACH, der Lernprozesse radikal auf zwei Lernebenen reduziert, nämlich „*Prozesse des Verstehens*“ und „*Verfügbarkeit des Verstandenen*“[30], und dabei Üben und Anwenden zusammen unter das Zweitgenannte subsumiert. Üben und Anwenden beschreiben also die Prozesse, die zu den Ergebnissen Sicherung und Transfer führen.

Diese inhaltliche Nähe von Sicherung und Transfer ermöglicht mir in der Anlage der Unterrichtsreihe eine über weite Strecken gemeinsame Betrachtung der beiden Phasen. Wie erwähnt, wird dem Transfer die zentrale Rolle bei der Betrachtung zukommen, auch deswegen, da erreichter Transfer die im Anspruch darunter stehende Sicherung mit einschließt.

In Bezug auf meinen Unterricht bedeutet das: Den Schülern wird die Möglichkeit gegeben, ihre Kenntnisse über die algorithmischen Grundstrukturen unter neuen Bedingungen anzuwenden. Konstituiert werden diese neuen Bedingungen durch neue inhaltliche Faktoren (neue Aufgaben), neue mediale Faktoren (LEGO-Mindstorms, NQC) sowie unter Umständen neue soziale Faktoren (Sozialformen im Unterricht).[31] Wenn die Schüler in der Lage sind, ihre Kenntnisse und Fähigkeiten unter diesen neuen Bedingungen zu rekonstruieren, hat der Transfer stattgefunden. Betrachtet man Sicherung als Rekonstruktion unter weitgehend vertrauten Bedingungen könnte diese auch als extrem eingeschränkter Transfer verstanden werden. Vertraute Bedingungen werden dadurch erreicht, dass eine Wiederholung der konkret zu sichernden algorithmischen Grundstruktur am Beispiel von NQC erfolgt.

---

[30] ROSENBACH, M.: *Die Sicherung des Lernerfolges*. 2003.
Quelle: http://bebis.cidsnet.de/weiterbildung/sps/allgemein/bausteine/struktur/lernerfolg.htm (28.2.04).

[31] „Neu“ bedeutet hier nur, dass es anders ist als in der Situation des Strukturerwerbs, nicht, dass es für die Schüler prinzipiell neu ist.

2.4 Methodisches Vorgehen

Es stellt sich grundsätzlich die Frage, ob für das Erreichen von Sicherung und Transfer, für den angestrebten Inhalt der Algorithmik oder für die Arbeit mit einem Unterrichtsmittel wie LEGO-Mindstorms bestimmte methodische Varianten besser geeignet sind als andere und deswegen bei der Planung besonders in Erwägung gezogen werden müssen.

Bezüglich der Anwendung gibt AEBLI eine deutliche Richtung vor: „*Auf der Stufe der Anwendung streben wir auch das Ziel an, den Schüler selbständig werden zu lassen ... Das neue Ziel ... besteht darin, den Schüler von Anleitungen unabhängig zu machen, ihn in die Lage zu versetzen, seine Begriffe und Denkoperationen vor neuen Gegenständen und Problemen selbständig anzuwenden.*“ (AEBLI, 367)

Hier kommt eine Grundanforderung an den Lehrer zum Ausdruck, nämlich die, dem Schüler Gelegenheit zum selbstständigen Handeln, zur Selbsttätigkeit zu geben, denn „*ohne Selbsttätigkeit ist keine Selbständigkeit der Schüler zu erreichen*.“[32]

Schon für Hugo GAUDIG sollte Selbsttätigkeit die den Charakter der Schule beherrschende Tätigkeitsform sein, bei der der Schüler in allen Phasen des Lernens „*ohne Fremdeinwirkung ... freitätig*“[33] ist und sich als „*handelndes Subjekt*“ und „*Täter seiner Taten*“ (ebd.) erfährt. MEYER rückt das dahingehend zurecht, dass er die Rolle des Lehrers im Rahmen selbsttätigen Lernens hervorhebt, und zwar zur Schaffung der Voraussetzungen und des Rahmens für Selbsttätigkeit (vgl. MEYER 1987, 418). Den geeigneten Rahmen für Selbsttätigkeit sieht er im **Handlungsorientierten Unterricht**[34].

Handlungsorientierter Unterricht ist ein Unterrichtskonzept, bei dem den Schülern „*der handelnde Umgang mit Lerngegenständen ermöglicht werden soll. Die materiellen Tätigkeiten der Schüler bilden dabei den Ausgangspunkt des Lernprozesses.*“[35] Dies bedeutet das Arbeiten mit fassbaren Gegenständen im Gegensatz zum Lernen anhand abstrakter Modelle, der bloßen Vorstellungen von Dingen. „*Es geht darum, handelnd Denkstrukturen aufzubauen und den Zugang zur Welt nicht über ihre Abbilder und ... Surrogate zu vereinseitigen.*“[36] Nach MEYER ist beim Handlungsorientierten Unterricht die „Kopf- und Handarbeit“ ausgewogen, je-

[32] MEYER, H.: *Unterrichtsmethoden*. II. Praxisband. Cornelsen Frankfurt am Main 1987, S.418.
[33] GAUDIG, H.: *Die Schule im Dienste der werdenden Persönlichkeit*. Leipzig 1917 (zit. nach REBLE, A: *Die Arbeitsschule*. Klinkhardt Bad Heilbrunn 1963, S.72).
[34] Die Großschreibung orientiert sich an Hilbert MEYER (1987).
[35] LENZEN, D. (Hrsg.): *Enzyklopädie Erziehungswissenschaft.* Bd.3 Stuttgart 1986, S.600.
[36] GUDJONS, H.: *Didaktik zum Anfassen*. Klinkhardt Bad Heilbrunn 1997, S.112.

doch sind es die Handlungsprodukte, die den Verlauf des Unterrichtsprozesses strukturieren (vgl. MEYER 1987, 214). Auch bei GUDJONS ist die Produktorientierung eines der grundlegenden Prinzipien des Handlungsorientierten Unterrichts. Er nennt insgesamt fünf Prinzipien:

1) Aktivierung vieler Sinne: Hiermit meint GUDJONS die Verbindung von Denken und Handeln unter Einbeziehung körperlich-sinnlicher Erfahrungen (anfassen, hören, riechen usw.).
2) Selbstverantwortung und methodische Kompetenz: Das meint die Mitorganisation und Mitverantwortung der Schüler bei Planung und Durchführung des Unterrichts. Methodenkompetenz bedeutet, „*Arbeitstechniken, Verfahrensweisen und Lernstrategien sachgerecht, situationsbezogen und zielgerichtet gebrauchen zu können.*“[37]
3) Produktorientierung: Produkte sind der Endpunkt eines Weges, aber über die Produkte soll auch der Weg dokumentiert und das Ergebnis resümiert werden. Bei AEBLI heißt dieser Vorgang „Arbeitsrückschau“ (AEBLI, 368). So soll der Zusammenhang von Handeln und Denken erreicht werden, denn „n*ur etwas ‚machen' ist kein Handlungsorientierter Unterricht*“ (GUDJONS, 117).
4) Kooperatives Handeln: Dem von- und miteinander Lernen kommt größte Bedeutung zu, um Selbstverantwortung im Rahmen der Lerngruppe wahrnehmen zu können, um methodische Kompetenzen zu erwerben und um Produkte in Selbsttätigkeit erstellen zu können. So wird „*leicht einsehbar, dass der Handlungsorientierte Unterricht kooperative Arbeitsformen braucht*“ (ebd.). Damit sind Formen wie Partner- oder Kleingruppenarbeit, aber auch das gemeinsame Klassengespräch gemeint.
5) Lebensbezug: Bezugspunkt des Handelns soll die Lebenswelt der Schüler sein. Die Handlungsprodukte haben im Idealfall tatsächlichen Einfluss auf die Lebenswelt.

(Nach GUDJONS, 115 ff.)

Zusammenfassend kann festgestellt werden: Anwendung kann zu Sicherung und Transfer führen, besonders dann, wenn die Anwendungsphase stark auf der Selbsttätigkeit der Schüler beruht. Diese Selbsttätigkeit scheint durch das Konzept des Handlungsorientierten Unterrichts in besonderem Maße verwirklichbar, da

[37] GRIMUS, M. u. a.(Hrsg.): *Evaluierungsprojekt "Neue Medien in der Grundschule"*. Österreichische Computer Gesellschaft 2000, S.207.

hier der Schwerpunkt auf dem Tun, dem materiellen Handeln liegt (Tätigkeit) und zusätzlich auf der Selbstverantwortung und Selbstorganisation bezüglich dieses Handelns (**Selbst**tätigkeit). Im Hinblick auf den Transfer ist Handlungsorientierung also ein sinnvoller methodischer Ansatz für das Unterrichtsvorhaben, denn Selbsttätigkeit und Selbstverantwortung fördern den erwünschten Rückgriff der Schüler auf verinnerlichtes Wissen und bekannte Konzepte.

Aus dem Konzept der Handlungsorientierung geht ebenso hervor, auf welchen Sozialformen sinnvollerweise der Fokus liegen sollte, nämlich auf kooperativen Formen wie Partner- oder Gruppenarbeit. Dabei liegt der Schwerpunkt neben dem individuellen Lernen auf der sozialen Interaktion. Die Fähigkeit zum konstruktiven Umgang und Austausch mit anderen Menschen wiederum ist eine Schlüsselqualifikation für jedes Handeln und ein primäres Bildungsziel.[38]

3 Unterrichtsvoraussetzungen

In diesem Kapitel werden die Rahmenbedingungen für die Unterrichtsreihe dargelegt. Die allgemeinen Unterrichtsvoraussetzungen umfassen Aufbau und Struktur der Lerngruppe. Die speziellen Unterrichtsvoraussetzungen beschreiben die äußere Situation und die organisatorischen Umstände, durch die die Reihe bedingt ist, sowie detailliert die Kenntnisse und Fähigkeiten, welche die Schüler zu Beginn der Reihe besitzen.

3.1 Aufbau und Struktur der Lerngruppe

Seit Beginn des Schuljahres 2003/2004 unterrichte ich selbstständig den klassenübergreifenden Basiskurs 11ü. Der Basiskurs besteht aus neun männlichen Schülern. Drei Schüler dieses Kurses kommen aus Aufbauklassen, das heißt, sie wechselten zum 11. Schuljahr von der Realschule ans Gymnasium. Der Informatikunterricht umfasst in diesem Jahr drei Stunden pro Woche.
Bis auf einen haben alle Schüler zu Beginn des Schuljahres informatische Vorkenntnisse angegeben, die sich insbesondere auf den Bereich Internet und HTML beziehen. Es gibt großes Interesse und Aufnahmebereitschaft gegenüber neuen Inhalten. Dazu ist zu ergänzen, dass der Informatikkurs ein Interessenkurs ist, der von den Schülern zusätzlich und fakultativ zu den drei Wahlpflichtkursen belegt wird. Auch darin spiegelt sich das starke fachliche Interesse wider, welches die Lern- und Arbeitsatmosphäre in positiver Weise prägt.

---

[38] Vgl. GIESECKE, H.: Was ist ein „Schlüsselproblem"? Anmerkungen zu Wolfgang Klafkis neuem Allgemeinbildungskonzept. Neue Sammlung 37 1997, S.562.

Bezüglich der Leistungsfähigkeit heben sich insbesondere zwei Schüler vom Rest der Lerngruppe ab; sie beschäftigen sich in besonderem Maße auch in ihrer Freizeit mit Computern, haben algorithmische Vorkenntnisse und verarbeiten neue Inhalte schneller. Im Verlauf der ersten Monate haben sich diese beiden, André S. und André St., leistungsmäßig als „Spitzengruppe“ etabliert. André St. besitzt selbst einen LEGO-Mindstorms-Baukasten, wobei er den Roboter nach eigenem Bekunden bisher nur mit dem grafischen *RCX-Code* programmiert hat. Zwei weitere Schüler, Claudius und Daniel, sind nach meiner Beobachtung fachlich eher schwach, aber motiviert.
Die Zusammensetzung des Kurses aus Schülern mehrerer Klassen zeigt sich auch in der sozialen Struktur der Lerngruppe. Jene Schüler, die sich aus den Klassen kennen, arbeiten und sitzen in der Regel zusammen. Jedoch ist die Kursstärke mit neun so gering, dass es in der Praxis zwangsläufig zu sozialem Austausch und Lernkontakten kommt.

3.2 Organisatorische Bedingungen des Unterrichts

Für diesen Kurs sind pro Woche eine Einzel- und eine Doppelstunde veranschlagt. Der für die Einzelstunde am Montag zur Verfügung gestellte Raum stellte sich jedoch als technisch ungeeignet für die Arbeit mit LEGO-Mindstorms heraus (fehlende, aber notwendige USB-Schnittstellen an den Rechnern), so dass ich in Absprache mit den Schülern und der Schule ab der zweiten Woche der Reihe dienstags von der siebten bis zur neunten Stunde, also drei Stunden im Block, unterrichte. In Stunden, in denen keine Rechnerarbeit notwendig ist, werde ich in der Woche vorher kurzfristig umplanen und in Absprache mit den Schülern die Einzelstunde auf den Montag legen, damit dieser Dreistundenblock umgangen werden kann. Ich betrachte diesen Block ambivalent, einerseits können drei Stunden hintereinander am Nachmittag auf die Schüler sehr erschöpfend wirken und letztlich ineffektiv sein (was ich befürchtete), andererseits könnte es die Effizienz der Arbeit mit LEGO-Mindstorms auch steigern, da die Arbeitsphasen nicht so oft unterbrochen werden und die technischen Umstände (Holen der Roboter, Anschluss, Installation) nicht wiederhergestellt werden müssen.
In dem Unterrichtsraum steht für jeden Schüler ein mit dem lokalen Netz und dem Internet verbundener Rechner bereit.
Die Schule verfügt über acht LEGO-Mindstorms-Baukästen des Typs „Robotics Invention System 2.0“. Sechs Roboter sind bereits von anderen Schülerkursen zusammengebaut, so dass ich diese nach einer Funktionsüberprüfung für meinen Kurs nutzen kann. Es gibt nur die Sensorentypen Berührungs- und Helligkeitssensor.

Die Programmierumgebung „Bricx-Command-Center“ in der Version 3.3 mit enthaltenem NQC-Compiler ist frei im Internet zugänglich[39] und wird zusätzlich von mir im lokalen Netz als Installationsdatei freigegeben. Die Sicherheitskonzeption der Schulrechner erfordert eine neue Installation dieser Software nach jedem Rechnerneustart, was aber aufgrund der relativ geringen Größe der Datei kein organisatorisches Problem darstellt.

3.3 Kenntnisse und Fähigkeiten der Lerngruppe

Die hier dargestellte LEGO-Mindstorms-Unterrichtsreihe schließt an den algorithmisch orientierten Anfangsunterrichts mithilfe der Lern- und Programmierumgebung *Robot Karol* an. *Robot Karol* ist direkt zur Einführung in die Algorithmik und zum Erlernen algorithmischer Basiskonzepte für Schüler und Schülerinnen gedacht. Das System basiert auf der Idee eines in einer dreidimensionalen Bildschirmwelt lebenden Roboters, der mittels einer sehr einfachen Programmiersprache in dieser Welt bewegt werden kann.[40] Der didaktischen Ansatz für den Anfangsunterricht, der hinter *Karol* steht, wird in der Informatik „variablenfrei“ genannt, wobei es im Kern um die Abwesenheit von Datenstrukturen geht. Diese unterschiedliche Gewichtung algorithmischer Strukturen, nämlich der Ablaufstrukturen gegenüber den Datenstrukturen hat direkte Konsequenzen für den nachfolgenden Unterricht.
Die inhaltlichen Unterrichtsvoraussetzungen haben für die Betrachtung von Transfer und Sicherung eine so große Bedeutung, dass sie für das Verständnis der weiteren Darstellung hier im Detail ausgeführt werden müssen.

Im Folgenden werden die **grundlegenden algorithmischen Strukturen** dargestellt**,** deren Kenntnis aufgrund der vorhergehenden Arbeit mit *Robot Karol* für diese Reihe bei den Schülern vorausgesetzt wird und die den Ausgangspunkt für die erwünschten Transferleistungen bilden.

- **Anweisungen** sind elementare Programmbestandteile; sie weisen Aktionen an und werden als Sequenz nacheinander ausgeführt. Eine beendete, d.h. durchgeführte Anweisung beendet auch die Aktion. *Karol* beherrscht nur wenige Aktionen, die mit Anweisungen wie z.B. „`Schritt`“ oder „`Aufheben`“ programmiert werden.

[39] Http://sourceforge.net/projects/bricxcc/ (19.3.04).
[40] Dieses Konzept wurde 1981 von Richard PATTIS in „*Karel the Robot: A Gentle Introduction to the Art of Programming*“ veröffentlicht und hat seitdem viele Anpassungen und Änderungen erfahren. Die hier verwendete Variante stammt von Ondrej Krsko und wurde von Ulli Freiberger ins Deutsche übertragen. Der Vorteil der Programmierumgebung auf der affektiven Ebene ist die für den Schüler wichtige sofort sichtbare Reaktion auf seine Eingaben und auf der inhaltlicher Ebene die starke Reduktion auf algorithmische Ablaufstrukturen.

- **Verzweigungen** bestimmen den Abarbeitungsverlauf der Anweisungen eines Programms. Den Schülern bekannt ist sowohl die Verzweigung ohne Alternative („`wenn BED dann ANW`“) als auch die Verzweigung mit Alternative („`wenn BED dann ANW sonst ANW`“). Verzweigungen sind abhängig von einer Bedingung, d.h. einem Ausdruck, der immer auf einen der Booleschen Werte *true* oder *false* zurückzuführen ist. Ein Beispiel einer Bedingung aus *Karol* ist „`IstWand`“. Bedingungen sind bei *Karol* nicht durch Boolesche Operanden kombinierbar, solche sind den Schülern auch nicht bekannt.
- **Wiederholstrukturen (Schleifen)** steuern die Wiederholung von Anweisungen abhängig von Bedingungen. Bekannt ist den Schülern die Zählschleife in Form von „`wiederhole n mal ANW *wiederhole`“, die vorprüfende Schleife in den zwei semantisch gleichen Formen „`solange BED tue ANW *solange`“ und „`wiederhole solange BED ANW *wiederhole`“ sowie die nachprüfende Schleife in den zwei semantisch gegensätzlichen Varianten „`wiederhole ANW *wiederhole bis BED`“ und „`wiederhole ANW *wiederhole solange BED`“. Bei nachprüfenden Schleifen wird der Schleifenkörper in jedem Fall mindestens einmal ausgeführt.
- **Unterprogramme** (Prozeduren) sind aus dem normalen Programmablauf ausgegliederte Programmteile, die mithilfe ihres Namens aus dem Hauptprogramm ausgerufen werden. Die Nutzung von Unterprogrammen führt in der Regel zu übersichtlicheren, kürzeren und leichter zu verändernden Programmen. Prozeduren werden in *Karol* als selbstdefinierte Anweisungen bezeichnet und haben die Form „`Anweisung name ANW *Anweisung`“. Außerdem bekannt ist den Schülern der Parameter als Platzhalter und seine Nutzung in parametrisierten Prozeduren in der Form „`Anweisung(X) name ANW *Anweisung`“.

Darüber hinaus kennen die Schüler den Begriff der Effizienz eines Algorithmus' als gekennzeichnet durch die nötige Abarbeitungszeit des Algorithmus und die benötigten Betriebsmittel (z.B. belegter Speicher). Da die Verwendung von Schleifen und Prozeduren sich direkt auf die Effizienz eines Algorithmus auswirken kann, ist dieser Begriff relevant.

Eine nicht direkt algorithmische, aber dennoch inhaltliche Voraussetzung ist das Wissen um die Notwendigkeit formeller Strukturierung von Programmcode. Das umfasst:

- Einrückungen zur Verdeutlichung zusammengehöriger Teile
- Kommentare zur Funktionsbeschreibung zusammengehöriger Teile
- die Verwendung aussagekräftiger Bezeichner.

Die Notwendigkeit letztgenannter Inhalte besteht nicht für die Funktionsfähigkeit eines Programms, sondern für die Analyse und das Verständnis komplexerer Programme. Für Schüler, die gerade am Anfang exemplarische Einzelprobleme algorithmisieren, sind die genannten Anforderungen verständlicherweise oft unfunktionaler Ballast, den sie gerne beiseite lassen.

4 Die Unterrichtsreihe

Die durchgeführte Unterrichts*einheit* zu LEGO-Mindstorms umfasst inhaltlich nicht nur Aspekte zur Sicherung und zum Transfer. Ich betrachte in dieser Arbeit jedoch nur den Teil der Unterrichtseinheit, der sich mit der Sicherung und dem Transfer befasst, im Folgenden Unterrichts*reihe* genannt.
Die Unterrichtsreihe wird in den folgenden Ausführungen unter dem Gesichtspunkt ihrer Einbettung in den Rahmenplan, der sachstrukturellen Besonderheiten von LEGO-Mindstorms und NQC sowie der algorithmischen Basisstrukturen dargestellt. Außerdem wird die didaktisch-methodische Planung des Unterrichts unter den Aspekten Sicherung und Transfer dargelegt und erläutert.

4.1 Bezug zum Rahmenplan

Die Gesamtplanung des ersten Halbjahres geht schwerpunktmäßig auf den Bereich der Algorithmik ein. Der derzeit gültige Berliner Rahmenplan Informatik von 1993 vermerkt zu dem Thema Grundsätzliches: „*Im Vordergrund des Informatikunterrichts steht die Algorithmik. Dabei soll die Fähigkeit zum selbständigen Finden und Formulieren algorithmischer Lösungen komplexer Probleme systematisch entwickelt werden.*“[41]
Der Rahmenplan nennt für das erste Unterrichtsjahr folgenden Punkt als inhaltlich zentral: „*Konstruktion von Teilalgorithmen zu Anwendungsfällen*“ (ebd., 9). Dafür sind im Rahmenplan 35 Stunden - fast die Hälfte der zu unterrichtenden 80 Stunden - vorgesehen, wobei „*algorithmische Elemente kennen gelernt*“ (ebd.), „*Elemente der Programmiersprache erlernt*“ (ebd.) und wieder verwendbare „*Bausteine verwendet*“ (ebd.) werden sollen.

[41] SENATSVERWALTUNG FÜR SCHULE, BERUFSBILDUNG UND SPORT (Hrsg.): Vorläufiger Rahmenplan für Unterricht und Erziehung in der Berliner Schule - Gymnasiale Oberstufe - Fach Informatik. Berlin 1993, S.12.

Die im Rahmenplan vorgesehene Konstruktion von Algorithmen ist ein zentraler Unterrichtsgegenstand der LEGO-Mindstorms-Unterrichtsreihe und die Programmierung des LEGO-Mindstorms-System selbst ist ein einsichtiger Anwendungsfall.

4.2 Sachanalyse

Der Unterricht befasst sich mit der Programmierung von Steuerungsalgorithmen für LEGO-Mindstorms mit NQC. Die Programmierung in NQC ist dabei der zentrale Gegenstand des Unterrichts. Aufgrund der Spezialisierung der Sprache NQC auf die Programmierung von LEGO-Mindstorms-Robotern einerseits und der technischen Parameter von LEGO-Mindstorms auf Hardwareebene andererseits besteht ein enger Zusammenhang zwischen Mindstorms, NQC und den algorithmischen Möglichkeiten von NQC. Ich lege im Folgenden die sachstrukturellen Besonderheiten und Zusammenhänge von LEGO-Mindstorms, NQC und den algorithmischen Basisstrukturen dar.

Ein mit dem LEGO-Mindstorms-Baukasten (RIS) zusammengebauter Roboter bietet ein relativ übersichtliches Potenzial an Handlungsmöglichkeiten. Er kann fahren (vorwärts und rückwärts sowie unterschiedlich schnell), er kann sich drehen (indem sich die beiden Motoren entgegengesetzt bewegen oder einer abgeschaltet ist), er kann einstimmige Töne und Melodien ausgeben sowie Text auf dem eingebauten LCD-Display anzeigen. Außerdem ist noch das Senden von Anweisungen über die Infrarot-Schnittstelle und zeitgesteuertes Warten möglich.
Die genannte Funktionalität bedingt weitgehend das Anweisungsset der Sprache NQC. Für die Ablaufsteuerung von Programmen bietet NQC die grundlegenden algorithmischen Strukturen. Folgende sachliche Besonderheiten gibt es dabei:

Anweisungen werden im Gegensatz zu *Karol* immer mit einem Semikolon abgeschlossen. Anweisungssequenzen innerhalb von Schleifen, Verzeigungen oder Prozeduren werden in geschweifte Klammern eingeschlossen.

Bedingte Verzweigungen sind in klassischer Weise als „`if{}else{}`“ eingebaut; das weist Ähnlichkeit zu *Karols* „`wenn Bedingung dann Anweisung sonst Anweisung`“- Darstellung auf. NQC bietet das Konzept der Mehrfachauswahl mittels „`switch`“ und „`case`“, das den Schülern nicht bekannt ist.

Bedingungen sind Boolesche Ausdrücke, die komplex sein können, d.h., sie können durch Operatoren (z.B. des Vergleichs) kombinierte Operanden enthalten. Hierin besteht ein Unterschied zu Bedingungen in *Robot Karol*. Bei LEGO-Mindstorms kann dies bezüglich der Abfrage der Berührungssensoren von Bedeutung sein, wenn an einem Roboter zwei dieser Sensoren angebracht sind.

Zählschleifen werden mit der „`repeat (Anzahl){}`“ - Konstruktion implementiert, die hohe Ähnlichkeit zu der den Schülern aus Karol bekannten „`Wiederhole n-mal`“ - Darstellung aufweist.

Es gibt zwei Typen **vorprüfender Schleifen** in NQC, die „`while (Bedingung){}`“- und die „`until (Bedingung){}`“ -Schleife, die unter entgegengesetzten Bedingungen den Schleifenkörper ausführen. Die genannte „`while`“ - Schleife entspricht semantisch *beiden* Varianten vorprüfender Schleifen bei *Karol* (vgl. 3.3), welche inhaltlich identisch sind. Neu für die Schüler in Bezug auf ihre Voraussetzungen ist insofern die Konzeption von „`until`“ bei vorprüfenden Schleifen, bei welcher die Bedingung nicht erfüllt sein darf, damit der Schleifenkörper ausgeführt wird.

Für **nachprüfende Schleifen** wiederum gibt es in *Karol* diese beiden beschriebenen Varianten, jedoch nicht in NQC. Für die nachprüfende „`solange`“ -Struktur in Karol gibt es in NQC die Entsprechung „`do{} while (Bedingung)`“, für das nachprüfende „`bis`“ aus *Karol* ist in NQC die sehr NQC-spezifische Variante „`until(Bedingung)`“ vorhanden. Letztere hat keinen definierten Schleifenkörper[42]. Sie hält das Programm an, bis die Bedingung zutrifft. Damit entspricht sie inhaltlich einem „`do{leerer Schleifenrumpf} while (Nicht(Bedingung))`“.

Dieses „`until`“ ist wegen des fehlenden Rumpfes eine algorithmisch ungenaue Konstruktion (in NQC über interne Makros realisiert) und für die Schüler unter Umständen verwirrend. Deswegen sollte diese Variante im Unterricht vermieden werden. Bezogen auf den Transfer ist zu sagen, dass dieser für die eben kritisierten speziellen Struktur kaum zu leisten ist.

Unterprogramme bietet NQC in vier verschiedenen Varianten. Es gibt Tasks, Programmteile, die bei Aufruf parallel ablaufen können und somit Multitasking erlauben. Eine Task, die „`task main()`“, ist zum Erstellen eines NQC-Programms immer erforderlich[43].

Weiterhin gibt es Prozeduren, die mit „`sub`“ deklariert werden und herkömmlichen Prozeduren entsprechen. Sie werden außerhalb von Tasks deklariert und erlauben keine Parameter. Aufgrund von Hardwarebeschränkungen sind maximal acht Prozeduren in einem Programm erlaubt. Außerdem können sich Prozeduren nicht gegenseitig aufrufen, was z.B. Rekursion verhindert.

---

[42] Insofern ist die Einordnung unter „nachprüfenden Schleifen“ auch nicht völlig korrekt. Es soll aber auch von dem `until` der vorprüfenden Schleife mit Schleifenkörper unterschieden werden.
[43] Tasks werden hier in vereinfachender Weise mit zu den Unterprogrammen gerechnet, weil sie auch ausgegliederten und mit einem Namen aufrufbaren Programmcode darstellen.

Diese Nachteile haben die mit „`void`“ deklarierten `Inline`-Funktionen nicht. Sie erlauben je einen Parameter und sind in der möglichen Anzahl theoretisch unbegrenzt. Dabei gibt es aber keine Rückgabewerte, womit eigentlich der informatische Funktionsbegriff, den BAUM verwendet, hier nicht zutreffend ist. Die `Inline`-Funktionen werden bei der Kompilierung an die Aufrufstellen kopiert und benötigen daher mehr Speicherplatz als gleiche Unterprogramme des Typs „`sub`“.

In ähnlicher Weise funktionieren auch Makros, eine vierte Möglichkeit, Programmteile im Sinne der Unterprogrammtechnik auszugliedern. Sie werden vom Präprozessor vor der Kompilierung an die entsprechenden Stellen im Quellcode kopiert. Im Unterschied zu `Inline`-Funktionen lassen sie mehr als einen Parameter im Aufruf zu.

Alle genannten Unterprogrammtypen ermöglichen auf Quelltextebene die Erstellung kürzerer und besser strukturierter Programme mit wiederverwendbaren Programmteilen und erfüllen damit aus Schülersicht die prinzipiellen Anforderungen an Unterprogramme.

Weitere relevante Unterschiede von NQC gegenüber Karol sind:

- NQC-Programme müssen vor der Übertragung auf den RCX kompiliert werden.
- NQC hat eine strenge Syntax, die z.B. auch die Groß- und Kleinschreibung betrifft. Diese Syntax erzwingt aber keine die Lesbarkeit verbessernde Programmerstellung (wie z.B. in Python).
- NQC-Schlüsselwörter sind englischsprachig.

4.3 Didaktisch-methodische Konzeption der Unterrichtsreihe

4.3.1 Didaktische Reduktion

NQC bietet grundsätzlich alle algorithmischen Konzepte, die transferiert bzw. gesichert werden sollen, also jene Inhalte, die in den Unterrichtsvoraussetzungen in 3.3 genannt wurden. Ausgehend von der Sachanalyse werden folgende Inhalte nicht für den Unterricht ausgewählt.

Bei Wiederholstrukturen:

- Die Schwierigkeiten der **NQC-spezifischen nachprüfenden `until`-Struktur** wurden bereits in der Sachanalyse angesprochen. Damit fällt die aus *Karol* bekannte „`wiederhole ANW *wiederhole bis BED`“ - Struktur für den Transfer aus, nicht jedoch nachprüfende Schleifen generell.

Unbenommen bliebe aber die Möglichkeit eines Transfers des Teilkonzeptes „`Ausführen-bis-Bedingung-eintritt`“, welches bei *Karol* in den nachprüfenden, bei NQC in den vorprüfenden Schleifen vorkommt.

Bei Unterprogrammen:

Obwohl es für die Nutzung von Unterprogrammen die vier dargestellten Möglichkeiten gibt, entscheide ich mich dafür, den Schwerpunkt auf die Prozedur „`sub`“ und die `Inline`-Funktion „`void`“ zu legen. „`Sub`“ und „`void`“ sind den Unterprogrammen bei *Karol* in Programmierung und Verwendung besonders ähnlich, wobei „`Sub`“ die unparametrisierten und „`void`“ die parametrisierten Unterprogramme deklariert. Daraus resultierend werden folgende Inhalte nicht thematisiert:

- **Tasks** dienen primär der Erstellung und Interaktion nebenläufiger Prozesse und sind insofern ein komplexes und eigenständiges Thema, das in dieser Reihe nicht vertieft werden soll. Der „`task main()`“ werden die Schüler zwar auf jeden Fall begegnen, diese kann aber als syntaktische Notwendigkeit kategorisiert werden.
- **Makros** sind keine echten Unterprogramme im Sinne der nur einmaligen Speicherbelegung und bieten auch keine didaktischen Vorteile der Benutzung. Die Möglichkeit mehrere Parameter zu benutzen, ist für den Transfer unerheblich, da es in *Robot Karol* nur maximal einen Parameter geben kann. Das soll über die Verwendung von `Inline`-Funktionen realisiert werden.

4.3.2 Didaktisch-methodischer Aufbau der Reihe

In dieser Reihe soll den Schülern die Möglichkeit gegeben werden, das ihnen bekannte Wissen über Anweisungen, Verzweigungen, Schleifen und Prozeduren in dem neuen Kontext LEGO-Mindstorms und NQC anzuwenden. Dazu wird der Unterricht im Kern handlungsorientierten Charakter tragen, da dies, wie in Kapitel 2.4 dargestellt, dem Prinzip der Anwendung und damit letztendlich Sicherung und Transfer zugute kommt. Die in 2.4 genannten Prinzipien der Handlungsorientierung werden folgendermaßen beachtet:

Die Produktorientierung wird durch Aufgabenstellungen erreicht, welche die Anforderungen spezifizieren, die ein LEGO-Mindstorms-Roboter erfüllen soll. Die Produkte sind dabei die Aktionen der LEGO-Mindstorms-Roboter und die erstellten Programme, auf denen diese Aktionen basieren. Die konkreten Aufgaben sind kooperativ und selbstverantwortlich zu lösen. Durch die Arbeit am Computer und am LEGO-Mindstorms-Roboter sowie die notwendige Bewegung der Schüler im Raum werden verschiedene Sinne aktiviert. Darüber hinaus werden in regelmäßigen Reflexionsphasen die Ergebnisse präsentiert und vergleichend analysiert.

Der Unterricht kann jedoch inhaltlich nicht so frei mit den Schülern geplant werden, wie es für Handlungsorientierten Unterricht optimal wäre, denn zum Erreichen von Sicherung und Transfer muss ein entsprechender Rahmen geschaffen werden. Dieser besteht in speziellen Aufgaben (siehe 4.3.3), welche die Schüler erhalten. Diese Aufgaben strukturieren den Transfer vor.

Zur Lösung der Aufgaben arbeiten sich die Schüler selbstständig in die Programmiersprache NQC ein. Damit wird zusätzlich der Forderung der Handlungsorientierung nach Selbsttätigkeit nachgekommen. Die an die Einarbeitung in NQC anschließende Anwendung dieser Sprache zur Lösung der Aufgaben ist bereits eine Transferebene (I) im Sinne von AEBLI. SCHUBERT und SCHWILL nennen dies *spezifischen Transfer*[44]. Eine weitere, anspruchsvollere Transferebene (II), der so genannte nichtspezifischer Transfer (ebd., 79), betrifft die Anwendung der bereits bekannten algorithmischer Strukturen im Kontext neuer Probleme, welcher durch die Aufgaben gebildet wird. Um diesen Transfer geht es mir primär.

Wenn Schüler Aufgaben bearbeiten und Lösungen mit LEGO-Mindstorms-Robotern vorführen, zeigt das die Fähigkeit der Schüler NQC anzuwenden. Dies ist spezifischer Transfer. Der nichtspezifische Transfer (Anwendung geeigneter algorithmischen Strukturen) lässt sich erst anhand der Art der Lösungen, also der

[44] SCHUBERT, S.; SCHWILL, A.: *Didaktik der Informatik*. Spektrum Akademischer Verlag Heidelberg 2004, S.79.

Quellcodes, bewerten: Haben die Schüler die am besten geeigneten algorithmischen Strukturen benutzt? Welche das jeweils sind, ist indirekt in den jeweiligen Aufgabenstellungen angelegt. Die Schüler müssen also die Problemstellung abstrahieren, die richtige algorithmische Struktur wählen und diese in NQC umsetzen, einer ihnen unbekannten Sprache.

Das Ergebnis jeder Aufgabe besteht insofern in separat zu betrachtenden Teilergebnissen, der funktionalen Lösung (Macht der Roboter, was er soll?) und der Implementation (Wurde die sinnvollste algorithmische Struktur gewählt?). Diese Teilergebnisse verweisen zurück auf die unterschiedlichen Transferebenen.

Eine funktionale Lösung der Aufgaben sollte auch auf Basis der unterrichtlichen Erarbeitung durch die Schüler möglich sein und nicht allein von der Transferleistung abhängen, die eine sehr anspruchvolle Anforderung darstellt. Für Schüler, die Probleme mit dem nichtspezifischen Transfer haben, muss die Erarbeitung einer Aufgabenlösung stattdessen den Zweck der Wiederholung und Übung (im Sinne von Sicherung) erfüllen. Transfer und Sicherung sollen also möglichst durch den Unterricht verbunden werden.

Die Erarbeitung von NQC soll im Hinblick auf dieses Ziel auf der Basis von NQC-Dokumentationen erfolgen, die den Schülern zur Verfügung gestellt werden.

Dokumentationen zur Verfügung zu stellen, bringt auch Probleme mit sich. Erstens enthalten sie häufig komplett erklärte Programmbeispiele. Fällt ein Beispiel mit einer Aufgabe zusammen und stoßen die Schüler darauf, ist unter Umständen nicht einmal mehr die Leistung des spezifischen Transfers (Erstellung eines funktionierenden NQC-Programms) zu garantieren, da die Lösungen einfach kopiert werden können.

Da sowohl die Voraussetzungen für den nichtspezifischen Transfer geschaffen werden als auch funktionale Lösungen (spezifischer Transfer) durch Erarbeitung prinzipiell ermöglicht werden sollen, sind die Anforderungen an die Auswahl der Dokumentationen sehr speziell. Sie müssen sowohl das Auffinden bekannter Ideen (Strukturen) ermöglichen und ihre NQC-Umsetzung beschreiben, als auch soweit erklärend sein, dass eine inhaltliche Hilfestellung gegeben ist. Gleichzeitig dürfen die Lösungen der Aufgaben nicht in den Beispielen wieder zu finden sein (s. o.).

Aufgrund der vorgenannten Überlegungen werden den Schülern folgende Dokumente zur Verfügung gestellt:

1. **Dave BAUM: NQC-Programmieranleitung**. Version 2.3 rev 1. Dieses Dokument wurde ausgewählt, weil es die vollständige NQC-Sprachreferenz vom Autor der Sprache selbst ist. Das Dokument enthält die Beschreibung aller relevanten Kontrollstrukturen und ist gleichzeitig gut strukturiert und verständlich. Die Beispiele sind in der Regel keine kompletten Programme, sondern Ausschnitte. Die den Schülern bekannten Bezeichnungen werden verwendet, allerdings nicht die Termini „vorprüfend", „nachprüfend" und „*Zähl*schleife".

2. **Matthew BATES: NQC Programmer's Reference Sheet.**[45] Dieses Dokument wurde ausgewählt, weil es eine sehr kompakte Syntax- und Befehlsübersicht über die Sprache NQC gibt, die auch den Schülern schnellen Zugriff auf benötigte Informationen ermöglicht. Die kurz gehaltenen Erläuterungen sind englischsprachig.[46]

Noch zu erwähnen ist Marc OVERMARS' „*Programmieren von LEGO-Mindstorms-Robotern mit NQC*". Dieses Dokument ist eine hervorragende Einführung in die LEGO-Mindstorms-Programmierung, die aufgrund vieler Beispiele schnell zum Erfolg führt. Leider decken die Beispiele viele Möglichkeiten und Varianten sinnvoller Aufgabenstellungen ab. Die Gefahr eines reinen Kopierens von Lösungen ist groß. Unter dem Aspekt des Transfers kann dieses Dokument aus meiner Sicht daher nicht sinnvoll eingesetzt werden. Für den Einstiegsunterricht in die Algorithmik mit LEGO-Mindstorms wäre es jedoch sehr gut geeignet.

Die gewählten Dokumente werden den Schülern als PDF-Dateien[47] auf dem lokalen Dateiserver zur Verfügung.

Ich drucke diese nicht für die Schüler aus, um sie im erarbeitenden Umgang mit elektronisch publizierten Dokumenten zu schulen. Dabei sind mir die Schwierigkeiten bewusst, die mit der Dokumentrezeption am Computer verbunden sind (langsameres Lesen, Notizen oder Markierungen schwer möglich). Ich nehme diese jedoch in Kauf, da das elektronische Dokument unsere Rezeption in zunehmendem Maße dominieren wird, es meist die erste Informationsquelle sein wird, die man zu einem Thema zur Hand haben wird, und es auch Vorteile im Handling bietet, z.B. durch schnelles Suchen, welches ich für wichtig erachte.

---

[45] BATES, M.: *NQC Programmer's Reference Sheet*. 2001.
Quelle: http://cosmos.ucdavis.edu/2002/robots/NQC_Ref.pdf (19.3.04).

[46] Das Verwenden auch englischsprachiger Dokumente ist damit zu rechtfertigen, dass alle Schüler dieses Kurses Englisch in der Schule lernen, und außerdem, dass Englisch die Sprache der Informatik ist und zumindest rudimentäre Kenntnisse für Informatik-Interessierte unabdingbar sind.

[47] Der notwendige *Adobe Acrobat Reader* ist auf den Schulrechnern bereits installiert. Die verfügbare Version erlaubt das Durchsuchen des Textes nach Suchbegriffen.

Für die Programmierung sollen die Schüler das „**Bricx-Command-Center**“ nutzen, das ihnen zur Verfügung gestellt wird. Es nimmt Routineaufgaben und -schwierigkeiten ab (Kompilierung, Übertragung), hilft mittels Syntax-Highlighting beim Verfassen des Quelltextes und erlaubt die weitgehende Konzentration auf Algorithmenentwurf und Programmerstellung.

Ausgehend von den theoretischen Überlegungen in Kapitel 2.4 bieten sich Partner- oder Gruppenarbeit als tragende **Sozialformen** für den Unterricht an. Unter Berücksichtigung der kleinen Lerngruppe von neun Schülern entscheide ich mich für Partnerarbeit, wovon ich mir gegenüber der Gruppenarbeit auch einen intensiveren Lernerfolg für den einzelnen Schüler verspreche, denn die Verantwortung für das Ergebnis liegt bei dieser Sozialform in erhöhtem Maße bei jedem Einzelnen. Die Fähigkeit der Schüler zur selbstständigen und disziplinierten Arbeit in Partnerarbeit ist von mir im vorangegangenen Unterricht erfolgreich erprobt worden.
Aufgrund der ungeraden Anzahl von Schülern wird es drei Zweiergruppen und eine Dreiergruppe geben. Aus meiner Sicht kommt der Kooperation bei der Selbsttätigkeit eine große Bedeutung zu, nicht nur zur Lösung des Problems, sondern auch bezüglich sozialer Kompetenzen. Daher entscheide ich mich nicht für eine weitere Zweiergruppe und einen einzeln arbeitenden Schüler.
Die Zusammensetzung der Partner gebe ich als binnendifferenzierende Maßnahme vor, um in etwa leistungshomogene Gruppen zu erhalten. Der Grund dafür ist, dass ich die Selbsttätigkeit jedes Schülers sichern will und verhindern möchte, dass leistungsstarke Schüler die gemeinsame Tätigkeit dominieren, was gerade aufgrund der wahrscheinlich hohen Motivation völlig ohne Böswilligkeit seitens der Schüler zu erwarten wäre. Um die besonders leistungsschwachen Schüler jedoch nicht von vornherein zu überfordern, werden diese nicht zu Partnern zusammengeschlossen, sondern es wird ihnen je ein Schüler der mittleren Leistungsstärke zur Seite gestellt.

Dies führt zu folgender Partneraufteilung in der Lerngruppe:

(Leistungseinschätzung: + überdurchschnittlich, o durchschnittlich, - unterdurchschnittlich)
Gruppe 1: André S.(+) / André St.(+)
Gruppe 2: Alexander(o) / Claudius(-)
Gruppe 3: Marco(-) / Robert(o) / Francesco(o)
Gruppe 4: Eric(o) / Daniel (-)

Die in dieser Konstellation zu erwartenden verschiedenen Leistungsfähigkeiten der Gruppen müssen über die Aufgabenstellungen abgefedert werden, z.B. über Zusatzaufgaben.

Die Partner- bzw. Gruppenzusammensetzungen sollen über den Zeitraum der LEGO-Mindstorms-Reihe stabil bleiben. Die Schüler sollen sich mit ihrer Gruppe identifizieren, was ich als weiteren Motivationsfaktor ansehe.

Um viele Sinne der Schüler anzusprechen wäre es zwar sinnvoll, die Roboter von den Schülern selbst zusammenbauen zu lassen, aufgrund des hohen Zeitbedarfs verzichte ich aber darauf. Zudem wäre es hinsichtlich des algorithmischen Schwerpunktes nicht zielführend. Die Tatsache, dass schon ausreichend zusammengebaute Roboter an der Schule existieren, ist insofern ein begrüßenswerter Umstand. Die vorhandenen Roboter sind im Detail recht unterschiedlich gebaute Modelle, was aber die grundsätzliche Funktionalität nicht beeinflusst. Relevanz hat aber, dass drei der vier Roboter mit zwei voneinander unabhängigen Berührungssensoren ausgestattet sind (vgl. 3.3). In der zweiten Stunde der Reihe werden die vorhandenen Roboter unter den gebildeten Gruppen verlost.

Die **geplante Stundenstruktur** in den Selbsttätigkeitsphasen der Reihe (Stunden 3-9) ist nahezu schematisch. Jede Gruppe holt sich ihren Roboter und installiert die Infrarotsender am Rechner sowie die Software „Bricx-Command-Center“. Dann kann sie sich eine Aufgabe vom Aufgabenstapel für ihre Gruppe nehmen. Die Aufgabenstellungen sind für alle Gruppen identisch (s. u.). Die Gruppe bearbeitet die Aufgabe, bis sie entscheidet, die Aufgabe als gelöst anzusehen. Die Funktionalität wird in der so genannten Testumgebung vorn im Klassenzimmer ausprobiert und die Lösung modifiziert, bis die Gruppe mit dem Ergebnis zufrieden ist. In dieser Phase erfolgt noch keine Präsentation. Der zweite Teil der Lösung, das NQC-Programm, wird auf eine Diskette gespeichert, um das Ergebnis in für die Schüler unveränderbarer Weise zu fixieren. Dann kann mit der Bearbeitung der nächsten Aufgabe begonnen werden. Die Auswertung und Präsentation der Lösungen findet am Ende des Unterrichts in der Reflexionsphase statt.

Aus der Zusammensetzung der Gruppen ergibt sich voraussichtlich ein recht unterschiedliches Arbeitstempo der Gruppen. Das bedeutet für die Planung, dass einerseits allen Gruppen genügend Zeit für die Lösung einer Aufgabe einzuräumen ist, andererseits Zusatzaufgaben für die schnelleren Gruppen bereitzustellen sind. Eine andere Möglichkeit wäre es, für die leistungsstärkeren Gruppen andere Aufgaben konzipieren, aber dies ginge zu Lasten der Vergleichbarkeit der Lösungen und zudem wäre es demoralisierend für die Schüler, wenn für sie der Eindruck entstünde, schon vorher vom Lehrer „in Schubladen gesteckt“ worden zu sein.
Für die geplanten Reflexionsphasen am Ende jedes mehrstündigen Blocks ist es erforderlich, dass alle (oder fast alle) Gruppen ihre Lösungen fertig gestellt haben. Insofern müsste eine Zeitbegrenzung für die Aufgaben gegeben werden. Ich vermute allerdings, dass es zu einer indirekten Wettkampfsituation zwischen den

mute allerdings, dass es zu einer indirekten Wettkampfsituation zwischen den Gruppen kommt, die ich nicht fördern will, die aber vermutlich dazu führen wird, dass die Gruppen ihre Aufgaben zügig bearbeiten und in den Reflexionsphasen ein Ergebnis voll oder zumindest zum Teil erstellt haben werden.

Pausen dürfen die Schüler innerhalb der Arbeitsphasen nach eigener Entscheidung machen; abgesprochen ist jedoch, dass diese nicht in der Reflexionsphase stattfinden sollen. Der zeitliche Beginn der Reflexionsphase bzw. das Ende der selbstständigen Arbeit wird an der Tafel notiert.

Für die **Reflexionsphase** werden von mir je 20 - 30 Minuten eingeplant. Hier werden die Lösungen gewürdigt. Zuerst soll eine Gruppe ihre Lösung mit dem Roboter vorführen, wobei die anderen Schüler beurteilen, ob die Aufgabe als erfüllt anzusehen ist. Dann sollen im Unterrichtsgespräch Lösungsideen sowie Probleme bei der Umsetzung ausgetauscht werden. Anschließend werden ein oder zwei Ergebnisquellcodes (von der Diskette) mit dem Beamer projiziert und können gemeinsam bezüglich Effizienz[48] des Algorithmus' sowie auf Lesbarkeit und Strukturiertheit des Quellcodes untersucht bzw. verglichen werden. Im Unterrichtsgespräch wird diskutiert, welche anderen Varianten möglich oder besser gewesen wären. Hier kann festgestellt werden, ob die Schüler die algorithmischen Strukturen verwendet haben und auf Basis welcher Überlegung sie diese wählten.

Die Verwendung der erwünschten algorithmischen Struktur in einer funktionierenden Lösung zeigt erreichten Transfer auf beiden Transferebenen an.

Die Schüler müssen selbst entscheiden, wie sie die jeweilige Aufgabe lösen und welche algorithmischen Strukturen sie dabei benötigen bzw. am sinnvollsten einsetzen. Hier entsteht die Lösungsidee, durch welche die Schüler den nichtspezifischen Transfer leisten oder nicht leisten. Die Lösungen, bei denen die sinnvollste Kontrollstruktur nicht eingesetzt wurde, die aber trotzdem mit LEGO-Mindstorms funktionieren und damit (nur) das Erreichen des spezifischen Transfers anzeigen, bedeuten, dass für diese Schüler eine Wiederholung der Begriffe (nach AEBLI)

[48] Die Effizienz wird zur Vereinfachung nicht hinsichtlich der Ablaufgeschwindigkeit, sondern nur bezüglich des Betriebsmittelverbrauchs untersucht, d.h. bezüglich des Quellcode-Umfangs. Letzterer ist natürlich nicht von der Anzahl der verwendeten Codezeilen abhängig, sondern von der Anzahl der benötigten Anweisungen und Kontrollstrukturen. Zur Art der Zählung: Eine verwendete Kontrollstruktur z.B. „`do {} while (Bedingung)`" wird wie eine verwendete Anweisung gezählt, wobei die Anweisungen im Schleifenrumpf selbstverständlich extra gezählt werden. Insgesamt bedeutet diese Zählweise, dass es zwar effizienter ist, drei gleiche Anweisungen in einer Schleife umzusetzen als sie hintereinander zu schreiben, nicht aber zwei Anweisungen. Über den vergleichbaren Faktor der auf die Programmlänge reduzierten Effizienz kann in den meisten Fällen der Nutzen spezieller Ablaufstrukturen direkt erkannt und entsprechende Lösungen transparent gewürdigt werden. Das Problem der `Inline`-Funktionen (Quellcodelänge ≠ Binärcodelänge) kann trotzdem an entsprechender Stelle thematisiert werden.

durchgeführt und Gelegenheit zur Übung gegeben werden muss. An dieser Stelle wird *Sicherung* vorbereitet.

Eine andere Möglichkeit die Beobachtung der verschiedenen Transferleistungen zu gestalten wäre, die Schüler zur Verwendung von **Struktogrammen** vor der eigentlichen Implementation anzuhalten. Struktogramme stellen algorithmische Ablaufstrukturen in programmiersprachenunabhängiger Form grafisch dar und sind den Schülern aus *Karol* bekannt. Gegen ein Vorschreiben der Verwendung von Struktogrammen spricht hier aber zum einen die dadurch erfolgende stärkere Steuerung des Unterrichtsablaufs und das Vorstrukturieren des Erarbeitungsweges in einem auf Selbstverantwortlichkeit angelegten Unterricht.

Zum anderen ist zu bedenken, dass LEGO-Mindstorms und *Karol* inhaltliche Gemeinsamkeiten besitzen, die viele Aktionen oder Handlungen in gleicher Weise beschreibbar machen, z.B. vorwärts bewegen, drehen oder an ein Hindernis stoßen. Viele Aufgaben, die man einem LEGO-Mindstorms-Roboter stellen kann, könnten in analoger Weise einem *Robot Karol* gestellt werden. Durch das Vorschreiben der Erstellung von Struktogrammen würde vom Lehrer eine zusätzliche Analogie zu *Karol* hergestellt, wodurch die Gefahr entstehen könnte, dass die Transferleistung gemindert würde. Natürlich können die Schüler das ihnen bekannte Hilfsmittel Struktogramm aus eigenem Antrieb verwenden. Den Weg zur Lösung sollen sie jedoch selbst finden.

Bezüglich der Sicherung dienen die in den darauf folgenden Stunden angesetzten Arbeitsaufgaben als Übungsfeld für die in der Reflexion bewusst gemachten Strukturen[49]. Die jeweils folgenden Aufgaben haben zwar einen anderen algorithmischen Schwerpunkt, erlauben aufgrund der Progression der Aufgaben aber auch die Übung des Besprochenen[50] (vgl. dazu 4.3.3[51]).

[49] Dies betrifft auch die formellen Richtlinien zur Quellcodegestaltung (Einrückungen, Kommentare, sinnvolle Bezeichner, vgl. 3.3).
[50] In Bezugnahme auf BÖNSCHS „Gesetze“ zur Übung (siehe 2.2) beachtet diese Form der Übung die Gesetze 1, 2, 5, 6, 7.
[51] Beispiel: Arbeitsaufgabe 4 erfordert die Nutzung einer bedingten Schleife. Diese kann in Arbeitsaufgabe 5, deren Schwerpunkt auf der bedingten Verzweigung liegt, wieder sinnvoll eingesetzt werden.

4.3.3 Die Arbeitsaufgaben zur Vorstrukturierung des Transfers

Den Arbeitsaufgaben kommt eine zentrale Steuerungsleistung im geplanten Unterricht zu. Sie sind so angelegt, dass der Einsatz bestimmter Kontrollstrukturen bei der Lösung als besonders sinnvoll zu erachten ist, wobei es, wie dargelegt, für die rein funktionale Lösung der Aufgaben verschiedene Ergebnisse geben kann, was selbstverständlich auch gewürdigt wird.

Die relativ wenigen verschiedenen Handlungsmöglichkeiten, die ein LEGO-Mindstorms-Roboter hat, und die wenigen verschiedenen Sensoren, deren Status in der Vielzahl der Fälle die algorithmischen Bedingungen darstellt, erschweren die Erstellung von Arbeitsaufgaben, die sinnvoll, exemplarisch für die jeweilige Struktur und klein genug für den Unterrichtrahmen sind.
Weiterhin sind bei der Aufgabenerstellung einfache physikalische Fallstricke zu beachten, worauf STOLT (41) hinweist. Ein LEGO-Mindstorms-Roboter ohne Rotationssensor dreht sich anhand einer vorgegebenen Zeit. Die Zeit, die er aber für eine vorgesehene Drehung benötigt, ist jeweils abhängig vom Untergrund oder von der aktuellen Batteriespannung, jedenfalls nie gleich. Dies wurde von mir nachgeprüft und kann bestätigt werden. Auch bei scheinbar gleichem Untergrund reagiert der Roboter anders.
Wenn aber nicht einmal eine 90° Drehung sicher programmiert werden kann, fallen viele spannende Aufgabentypen weg, eben jene, die auf selbstständiger Bewegung im Raum beruhen. Das, was ein *Robot Karol* in seiner simulierten Welt kann, sie nämlich vollständig determiniert abzulaufen, ist mit LEGO-Mindstorms in der wirklichen Welt nicht so einfach umzusetzen. Andererseits ist diese Erkenntnis des Unterschieds von Theorie und Praxis durchaus für die Schüler von Interesse. Insofern sollen die Schüler auf dieses Problem stoßen, damit aber nicht frustriert werden.

Folgende Aufgaben sind zum Zwecke des Transfers in dieser Reihenfolge von den Schülern zu lösen.

Vor der ersten Aufgabe ziehen die Schüler ein Startblatt mit Hinweisen.

Startblatt
Für alle Aufgaben gilt: Suchen Sie effektive Lösungen, schreiben Sie kurze Programme! Verkomplizieren Sie dabei aber Ihre Programme nicht unnötig. Kommentieren Sie Ihre Quelltexte und achten Sie auch auf eine übersichtliche Gestaltung ihres Quellcodes. Für selbst gewählte Bezeichnungen aller Art wählen Sie bitte verständliche Bezeichner.
Speichern Sie fertige Programme in Ihrem Home-Verzeichnis und auf der bei mir erhältlichen Diskette ab.

Didaktischer Kommentar: Vor dem Beginn wird den Schülern ins Gedächtnis gerufen, welche Anforderungen neben der funktionalen Lösung noch an sie gestellt werden.

Arbeitsaufgabe 1
Erstellen Sie ein Programm, das als Erstes Folgendes leistet:
Ihr Roboter soll Sound ausgeben. Die Anzahl der Pieptöne soll dabei der Nummer Ihrer Gruppe entsprechen.
Hintergrund: So kann in Zukunft erkannt werden, ob wirklich das eigene Programm mittels Infrarot zum Roboter übertragen wurde und nicht zufällig ein fremdes.

Didaktischer Kommentar: Diese einfache Aufgabe dient der ersten Annäherung an den Umgang mit NQC, die Nutzung der Dokumentationen und das technische Handling. Die Schüler können die Aufgabe komplett am Arbeitsplatz ohne Nutzung der Testumgebung lösen. Algorithmisch liegt der Schwerpunkt auf Anweisungen und Anweisungssequenzen.

Arbeitsaufgabe 2
Erstellen Sie ein Programm, das Folgendes leistet:
Der Roboter soll erst piepen (siehe 1. Aufgabe), dann zwei Meter vor- und zurückfahren. Ein aufgeklebtes A4-Blatt in der Testumgebung dient als Start- und Endfeld.
Die Aufgabe gilt als gelöst, wenn der Roboter mit einem Teil seiner Länge die 2-Meter Marke erreicht hat und am Ende wieder im Startfeld hält.

Didaktischer Kommentar: Hier werden, wie auch in Aufgabe 1, Anweisungssequenzen verwendet. Die Schüler machen sich vertraut mit der Motorsteuerung sowie dem zeitgesteuerten Warten („`wait`“) bei LEGO-Mindstorms. Im Idealfall wird das Piepen zu Beginn bereits als Unterprogramm ausgelagert.

Zusatzaufgabe 2.1
Verändern Sie Ihr Programm so, dass der Roboter auf dem Hinweg möglichst langsam fährt und auf dem Rückweg möglichst schnell.

Didaktischer Kommentar: Zusatzaufgaben dienen, wie erläutert, der differenzierten Förderung der leistungsstärkeren Schüler. Hier wird die Kenntnis der Steuerungsmöglichkeiten des Roboters vertieft.

Arbeitsaufgabe 3

Erstellen Sie ein Programm, das Ihren Roboter zweimal um den in der Testumgebung platzierten quadratischen Karton fahren lässt, ohne diesen zu berühren. Der Roboter muss am Ende wenigstens teilweise wieder im Startfeld stehen.

Hinweis: Speichern Sie auch eventuelle Zwischenlösungen unter eigenem Namen ab.

Didaktischer Kommentar: Diese Aufgabe ist dafür konzipiert, bei der Programmierung Zählschleifen einzusetzen. Theoretisch sind zwei verschachtelte Zählschleifen die optimale Lösung. Die Schüler werden jedoch voraussichtlich aufgrund der oben erläuterten Drehungsproblematik diese Lösung nicht einsetzen können, sondern am Ende eine weniger universelle Lösung mit einzelnen Fahrtkorrekturen anbieten. Dies muss in der Reflexionsphase besprochen werden. Dafür müssen entstandene Quellcode-Varianten aufgehoben werden. Da die Durchführung dieser Aufgabe in Kapitel 5 näher betrachtet und analysiert werden soll, wird hier bereits der Lösungsweg der Schüler antizipiert:

1. Die Schüler erkennen die Wiederholstruktur in der Aufgabe („*zweimal*“).
2. Die Schüler reaktivieren die aus *Karol* bekannten Begriffe „Schleife“ oder „Wiederholung“ aufgrund von 1.
   Hier erfolgt **nichtspezifischer Transfer**.
3. Die Schüler suchen nach "Schleife" oder "Wiederholung" in BAUMS Dokument.
4. Die Schüler finden folgenden Textteil:
   Die repeat-Anweisung führt ihren Rumpf mehrmals aus:
   repeat (Ausdruck) Rumpf
5. Die Schüler erkennen die Analogie zur ihnen bekannten Zählschleife (BAUM, 18) und befinden diese Kontrollstruktur für geeignet.
6. Die Schüler erschließen die genau Syntax für „`repeat`“ aus dem syntaktischen Beispiel für „`while`“, welches sich genau über dem zitierten Textausschnitt befindet, oder sie verwenden BATES (1):
   Repeat Statements
   repeat (expression) {
   // body repeated expression times (expression read once)
   }
7. Die Schüler erstellen auf dieser Basis ein funktionsfähiges NQC-Programm (**spezifischer Transfer**). Eventuell zerlegen sie die Teilaufgabe des einmaligen Umfahrens in die weiteren Teilaufgaben „Strecke abfahren“ und „Drehen“.
8. Die Schüler speichern ihre Lösungsvariante entsprechend der Aufgabenstellung.

9. Die Schüler testen und korrigieren eventuell ihre Lösungen bis die Aufgabe erfüllt ist.

Arbeitsaufgabe 4

Erstellen Sie ein Programm, das folgende Aufgabe erfüllt:
Ihr Roboter soll sich vom Startfeld aus langsam auf ein Hindernis zu bewegen. Dabei soll er jede halbe Sekunde einen Piepton von sich geben (als Zeichen, dass alles in Ordnung ist).
Beim Auftreffen auf das Hindernis soll er anhalten, drei Pieptöne von sich geben, zwei Sekunden überlegen und dann „schweigend" einen Meter zurücksetzen.

Didaktischer Kommentar: Diese Aufgabe ist eine Kombination von Schleifen und Anweisungssequenzen, bei der zum ersten Mal ein Sensor abgefragt werden muss. Die Schleife beim Hinweg des Roboters kann in der Praxis sowohl vor- als auch nachprüfend sein, die von mir bereits kritisierte „`until`" -Schleife ohne Rumpf wird mit der Aufgabenstellung (jede halbe Sekunde piepen) aber ausgeschlossen. Allerdings wird nur jede halbe Sekunde abgeprüft, ob der Roboter Berührungskontakt hat. Für die eventuell auftauchende Problematik der notwendigen logischen Verknüpfung zweier Bedingungen (bei zwei Tastsensoren) kann der Lehrer den Gruppen individuelle Hilfestellung geben, da diese Thema außerhalb der Transferbetrachtung liegt und wegen des fehlenden Vorwissens von den Schülern nur schwer selbst zu erarbeiten ist.
Die beste erwartete Lösung ist die Verwendung einer vorprüfenden Schleife, da sie verhindert, dass der Roboter eine Ton ausgibt und wartet, wenn er schon zu Programmbeginn Berührungskontakt hat. Dies kann in der Reflexionsphase thematisiert werden. Da die Durchführung auch dieser Aufgabe in Kapitel 5 näher betrachtet und analysiert werden soll, wird hier der Lösungsweg der Schüler antizipiert:

1. Die Schüler erkennen die Wiederholstruktur in der Aufgabe („*jede halbe Sekunde piepen*").
2. Die Schüler erkennen die bedingte Struktur dieser Wiederholung, die an die Vorwärtsbewegung des Roboters gebunden ist.
3. Die Schüler reaktivieren den aus *Karol* bekannten Begriff „bedingte Schleife" aufgrund von 1. und 2.
   Hier erfolgt **nichtspezifischer Transfer**.
4. Die Schüler suchen nach „Schleife", „bedingte Schleife" , „Bedingung" oder „Wiederholung" in BAUMS Dokument.
5. [...] Die Schüler erarbeiten sich die Syntax, wenn notwendig hilft der Lehrer bei logischen Verknüpfungen.

6. Die Schüler erstellen auf dieser Basis ein funktionsfähiges NQC-Programm (**spezifischer Transfer**) Möglich sind Lösungen mit vorprüfender (beste Lösung) und mit nachprüfender Schleife.
7. Die Schüler testen und korrigieren ihre Lösungen, bis die Aufgabe erfüllt ist.

Zusatzaufgabe 4.1
Erweitern Sie Ihr Programm 4 so, dass der Roboter im Anschluss an 4. versucht, an dem Hindernis vorbeizufahren.
Information: Im Bereich hinter dem Objekt verläuft eine mit Klebestreifen angebrachte weiße Linie.

Didaktischer Kommentar: Diese Aufgabe erfordert die Veränderung der Lösung von 4. dahingehend, dass sich der Roboter ein Stück weit zur Seite bewegt und dann dasselbe versucht wie in 4. Da er nicht „weiß“, wie breit das Hindernis ist, muss dies in eine bedingte Schleife gefasst werden, um mehrmals versucht werden zu können. Vorbeigefahren ist der Roboter dann, wenn er auf die Linie im Hinterland des Hindernisses trifft. Dies ist die Abbruchbedingung, die die Schüler erkennen müssen.

Arbeitsaufgabe 5: Hakenschlagen
Erstellen Sie ein Programm, das folgende Aufgabe erfüllt:
Ihr Roboter soll in der Arena eine kurze Zeit geradeaus fahren und sich dann zufällig beliebig weit nach links oder rechts drehen, dann wieder geradeaus fahren, sich in eine zufällige Richtung drehen usw. Dabei darf er die Arena nicht verlassen. Trifft er auf die Arenenbegrenzung, soll er anhalten.

Hinweis: Die Arena ist das in der Testumgebung mit weißem Paketklebeband markierte vier Quadratmeter große Feld.

Didaktischer Kommentar: Abgesehen vom Einsatz des Lichtsensors erfordert diese Aufgabe die *bedingte Auswahl,* wobei die Bedingung von der Nutzung der Zufallsfunktion `Random()` abhängt. `Random()` ist neu für die Schüler, wird aber mit Blick auf die Dokumentationen voraussichtlich kein Problem darstellen. Die Verwendung von *Random()* selbst verlässt zwar den Bereich von Sicherung und Transfer, ermöglicht aber gleichzeitig eine sinnvolle Verwendung der bedingten Verzweigung an einer Stelle, wo sie nicht durch eine bedingte Schleife ersetzt werden kann. Eine Schleife wird aber zusätzlich für den Abbruch bei Randberührung benötigt. Die Aufgabenidee orientiert sich an einem Beispiel von OVERMARS (13).

Zusatzaufgabe 5.1
Verändern Sie Ihr Programm 5 so, dass der Roboter beim Auftreffen auf die Begrenzung nicht anhält, sondern sich wieder der Arena zuwendet und sein „Hakenschlagen“ fortsetzt. Erst beim dritten Auftreffen auf die Begrenzung soll er endgültig anhalten.

Didaktischer Kommentar: Die Abbruchbedingung der Außenschleife muss geändert werden. Sinnvoll ist hier außerdem die Nutzung einer Prozedur, die innerhalb einer Zählschleife (dreimaliges Wiederholen) aufgerufen wird.

Arbeitsaufgabe 6
Erweitern Sie Ihr Programm 5 so, dass der Roboter, wenn er sich entscheidet nach links zu fahren, dreimal piept, wenn er sich entscheidet nach rechts zu fahren, fünfmal piept.

Hintergrund: Man soll in der Lage sein, die Position des Roboters mit dem Gehör nachzuvollziehen.

Didaktischer Kommentar : Der Gedanke bei dieser recht konstruiert wirkenden Aufgabe ist, dass hier Unterprogramme für die Aufgabe des Piepens eingesetzt werden können, im besten Fall sogar mit Parametern, welche die Anzahl des Piepens übergeben (durch `Inline`-Funktionen realisiert).

Die selbstständige Bearbeitung der Aufgaben durch die Schüler kann in Einzelphasen unterteilt werden[52]:

1. Problemanalyse und Entwurf. Die Schüler überlegen sich in Diskussion mit ihrem Partner bzw. ihren Partnern eine grundsätzliche Lösungsidee für die Aufgabe und einigen sich darauf.
2. Implementierung. Die Schüler erstellen ein Programm zur Umsetzung ihrer Lösungsidee mithilfe der Dokumentationen. Bei der Programmierung wird auf Strukturierung, Kommentierung, übersichtliche Gestaltung und aussagekräftige Bezeichner geachtet. Die Schüler beheben syntaktische Fehler, bis das Programm lauffähig ist, wonach sie es auf den LEGO-Mindstorms-Roboter übertragen.
3. Funktionsüberprüfung (Test). Das Programm wird bezüglich der Spezifikation der Aufgabenstellung von den Schülern getestet. Das Testen führt entweder zu einer befriedigenden Lösung, zur Korrektur der Implementierung oder zur Korrektur der Lösungsidee.

[52] Bei den Phasenbezeichnungen orientiere ich mich an den Standardbegriffen aus dem Software-Engineering (vgl. dazu ENGESSER, 655 ff.).

4.4 Lernziele der Unterrichtsreihe

In diesem Kapitel werden die Lernziele dargestellt, die in dem Unterrichtsteil erreicht werden sollen, der unter dem Beobachtungsschwerpunkt Sicherung und Transfer steht. Insbesondere aus 4.3 leite ich folgende Lernziele ab.

Inhaltliche Lernziele

(I1) Die Schüler können die Entwicklungsumgebung „Bricx-Command-Center“ kompetent bedienen.

(I2) Die Schüler kennen die Syntax von NQC.

(I3) Die Schüler können Steuerungsalgorithmen für LEGO-Mindstorms-Roboter in NQC entwickeln.

(I4) Die Schüler verwenden aufgrund von Transferleitungen die Kontrollstrukturen Anweisung, Anweisungssequenz (a), bedingte Verzweigung (b), Schleife (c) und Unterprogramm(d) zum Algorithmenentwurf.

(I5) Die Schüler formulieren ihre Programme in strukturierter und lesbarer Form.

(I6) Die Schüler vergleichen die Effizienz von erstellten Algorithmen.

(I7) Die Schüler bewerten die Strukturiertheit und Lesbarkeit von Programmen.

Methodische und soziale Lernziele

(M1) Die Schüler werten elektronisch publizierte, auch fremdsprachliche, Dokumentationen und Hilfesysteme zu NQC unter dem Aspekt der Problemlösung bzw. Aufgabenstellung aus.

(M2) Die Schüler erarbeiten selbstständig Lösungsstrategien und setzen diese um, wobei sie im Sinne des Tätigkeitszyklus (vgl. SCHUBERT/SCHWILL, 41) Fehler erzeugen, bemerken, interpretieren und beheben.

(M3) Die Schüler präsentieren ihre erstellten Lösungen mithilfe des Beamers (Programmcode) bzw. ihres LEGO-Mindstorms-Roboters (Funktionalität).

(M4) Die Schüler arbeiten eigenverantwortlich bezogen auf die benötigte Zeit (im zeitlichen Rahmen).

(M5) Die Schüler erarbeiten ihre Lösungen in Partner- bzw. Kleingruppenarbeit gemeinsam, verbessern und ergänzen sich.

4.5 Synopse der Unterrichtsreihe

	Stunde	Thema	Material	Groblernziele	Kooperationsform	Didaktischer Kommentar
	1 10.11.03 15.00 - 15.45	Roboter und Lego-Mindstorms	LEGO-Mindstorms, OHP, Tafel, PC	Die Schüler kennen Merkmale eines Roboters und erkennen diese bei LEGO-Mindstorms wieder.	Schülervortrag von André St., UG, Partnerarbeit	Die Schüler sollen ihre Vorstellungen von Robotern zusammentragen, Informationen, die für eine Definition des Roboterbegriffs hilfreich sind, im Internet finden sowie diese Informationen in Beziehung zu den Informationen aus Andrés Vortrag (10 min) über LEGO-Mindstorms setzen. Leitfrage: Ist das ein Roboter?
	2 11.11.03 14.10-14.55	Bricx und Lego-Mindstorms	PC, Tafel, LEGO-Mindstorms	I1	UG, Partner- bzw. Kleingruppenarbeit	Hier erfolgt die Vorbereitung der Selbsttätigkeit: Die Roboter werden verlost, die Partner verbunden. Die Rahmenbedingungen werden abgesprochen, Installation und prinzipielles Handling ausprobiert. Die Funktionsweise des „Bricx-Command-Centers“ wird geklärt.
	3 11.11.03 15.00-15.45	Anweisungen und Anweisungssequenzen in NQC	**Arbeitsaufgabe 1, Arbeitsaufgabe 2,** LEGO-Mindstorms, PC, Beamer	I1, I2, I3, I4 (a), I5, M1, M2, M3, M4, M5	Partner- bzw. Kleingruppenarbeit UG	s. didaktische Kommentare zu Aufgabe 1 und 2 (S.32)
	4/5/6 18.11.03 14.10- 16.35	Wiederholstrukturen in NQC	**Arbeitsaufgabe 3, Arbeitsaufgabe 4,** LEGO-Mindstorms, PC, Beamer	I1, I2, I3, I4 (c), I5, I7, M1, M2, M3, M4, M5	Partner- bzw. Kleingruppenarbeit UG	s. didaktische Kommentare zu Aufgabe 1 und 2 (S.33 f.)
	7/8/9 25.11.03 14.10- 16.35	Verzweigungen und Prozeduren in NQC	**Arbeitsaufgabe 5, Arbeitsaufgabe 6,** LEGO-Mindstorms, PC, Beamer	I3, I4 (b, d), I5, I6, I7, M1, M2, M3, M4, M5	Partner- bzw. Kleingruppenarbeit UG	s. didaktische Kommentare zu Aufgabe 1 und 2 (S.35 f.)

→ Innerhalb der LEGO-Mindstorms-Einheit folgt dieser Reihe:
1.Variablen und Konstanten (eine Stunde)
2. Erstellung eines Linienfolger-Programms (drei Stunden)

5 Durchführung und Analyse ausgewählter Stunden

5.1 Auswahl der Stunden

Wie in 4.3.2 erläutert wurde, verlaufen die Stundenblöcke bezüglich ihrer Phasen in ähnlicher Form. Daher wähle ich zur genaueren Darstellung exemplarisch einen dieser Blöcke aus. Ich entscheide mich für die zusammengehörigen Stunden 4, 5 und 6 der Reihe, deren thematischer Schwerpunkt auf den Wiederholstrukturen liegt. Aus didaktischer Sicht interessant ist das Verhalten der Schüler bei Arbeitsaufgabe 3 (vgl. didaktischer Kommentar, 33) sowie die Analyse der verschiedenen möglichen Lösungen für Aufgabe 4.

5.2 Feinlernziele

(FZ 1) Die Schüler wählen aufgrund einer Transferleistung die Zählschleife als für Aufgabe 3 geeignete Kontrollstruktur. Indikator: Die Schüler setzen die Zählschleife im Programm ein.

(FZ 2) Die Schüler verwenden aufgrund einer Transferleistung die bedingte Schleife als für Aufgabe 4 geeignete Kontrollstruktur. Indikator: Die Schüler setzen die bedingte Schleife (vor- oder nachprüfend) im Programm ein.

(FZ 3) Die Schüler können funktionsfähige NQC-Programme erstellen. Indikator: Die Schüler führen die Lösungen der Aufgaben 3 und 4 in der Testumgebung vor.

(FZ 4) Die Schüler können die syntaktische Umsetzung der erwünschten Strukturen durch die zur Verfügung gestellten Dokumentationen herausfinden. Indikator: Die Schüler nutzen die gewünschten Informationen.

(FZ 5) Die Schüler erkennen die Problematik der undeterminierten Drehung des Roboters und die Auswirkung auf ihren Algorithmenentwurf bei Aufgabe 3. Indikator: Die Schüler verändern ihre Algorithmen so, dass Aufgabe 3 funktional gelöst wird.

(FZ 6) Die Schüler bewerten die vorprüfende Schleife als für Aufgabe 4 am besten geeignete Kontrollstruktur. Indikator: Beide Schleifentypen werden bezüglich der Aufgabenstellung und des Extremfalls (Berührung des Gegenstandes von Beginn an) verglichen.

5.3 Durchführung und Analyse

Die Stunde wurde unter Hinweis auf die bekannte Arbeitsform und mit dem demonstrativen Aufstellen eines quadratischen Kartons in der Testumgebung eröffnet, die Schüler installierten ihre Hard- und Software und holten sich vom Lehrerarbeitstisch ihre aktuelle Aufgabe. Dies war bei allen Gruppen **Aufgabe 3**, da die Aufgaben 1 und 2 in der Vorwoche von allen Gruppen beendet werden konnten.
In der Phase der Problemanalyse konnte ich gleich einen regen Austausch der Partner beobachten, der schnell in eine abgesprochene Handlung überging. Offensichtlich war rasch klar, wie ein Lösungskonzept aussehen konnte. Alle Schüler waren sehr konzentriert und arbeiteten zügig. Es war unverkennbar, dass hier unter den Schülern eine Wettbewerbssituation um das schnellste Ergebnis vorlag. Gemeinsam wurde von den Partnern in Dokumentationen am Rechner gesucht und gelesen. Manche Schüler lasen parallel an benachbarten Rechnern in Dokumentationen und kommunizierten dabei.

Als die Gruppen sukzessive zum Testen nach vorn kamen, konnte ich mir ihre aktuellen Programme am Rechner ansehen. Das Problem des Umfahrens des quadratischen Kartons war auf drei verschiedene Weisen angegangen worden:

1) Bei Gruppe 4 stand der Befehl zum Fahren und Drehen achtmal im Quellcode. Anscheinend war er mit *Kopieren* und *Einfügen* vervielfältigt worden; das Programm war eine veränderte und erweiterte Version der Lösung von Aufgabe 2, wie ich am Dateinamen erkennen konnte.
2) Gruppe 2 und Gruppe 3 hatten das zweimalige Umfahren in einer Schleife programmiert, nicht jedoch das viermalige Zurücklegen der identischen Seitenlänge des Quadrats (FZ 1).
3) Gruppe 1 hatte gleich die verschachtelten Schleifen eingesetzt (FZ 1).

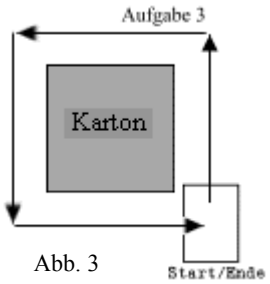

Abb. 3

Kein Programm funktionierte auf Anhieb, weil überall Weg- und Drehungslängen justiert und verändert werden mussten. Lösung 1 funktionierte bald am besten, weil für jede Drehung ein separater Wert ausgetüftelt werden konnte. Lösungen 2 und 3 waren zwar die algorithmisch besseren, stießen aber immer wieder gegen den Karton und kamen nie wie geplant auf dem Start/Endfeld an. Nach einigen Versuchen äußerten sich die Schüler auch entsprechend frustriert: „Der dreht sich immer anders.“ (André S.); „Das kann ja nicht funktionieren!“ (Robert). Ich erweiterte das Zielfeld auf den kompletten Bereich vor dem Karton (siehe Abb. 4). Damit

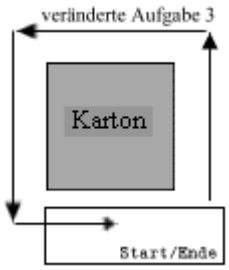

Abb.4

konnten die Gruppen nach einigen weiteren Versuchen auch Erfolge erzielen[53] (FZ 3). Die Spannung ließ an dieser Stelle etwas nach, da die Drehungsproblematik bereits erkannt worden war und die funktionale Lösung in gewisser Weise nur noch pro forma notwendig war. Hier offenbarte sich die Schwierigkeit im Umgang mit einer Aufgabe, deren korrekte Lösung eigentlich nicht möglich ist, bei der man den Schülern aber trotzdem einen Erfolg verschaffen will.
Da die Schüler letztendlich ein Ergebnis erzielen konnten, gab es keine starken Frustrationen bei den Schülern; sie waren bereit ihre Ergebnisse den praktischen Erfordernissen anzupassen. Insofern kann man diese Aufgabe bei der Arbeit mit LEGO-Mindstorms einsetzen, wenn man den Schwerpunkt auf Praxisorientierung legt. Dem Aspekt des Transfers der Schleifenstruktur ist diese Aufgabe aber weniger dienlich, da das Erbringen von Transfer nicht direkt mit einer funktionierenden Lösung belohnt wird.

Was bedeuten die erbrachten Lösungen für den Transfer der einzelnen Gruppen? Die Gruppen 1-3 fanden in der Aufgabenstellung das Konzept der Wiederholung wieder („ ... *zweimal ... fahren lässt*“). Beobachten ließ sich bei diesen Gruppen die Nutzung der Dokumentation von BAUM mithilfe der Suchfunktion oder des Inhaltsverzeichnisses. Hier lässt sich Transfer vermuten, der noch durch die Präsentation in der Reflexionsphase untermauert werden muss. Gruppe 4 hatte zwar erkannt, dass es sich um eine vier- bzw. achtmalige Wiederholung handelte, jedoch keine Schleife als Kontrollstruktur eingesetzt. In der späteren Reflexionsphase klärte sich, dass dies an der fehlenden Idee und nicht der fehlenden Fähigkeit zur Umsetzung lag. Dieses Ergebnis kann nicht als erfolgreicher Transfer gewertet werden, da diese Gruppe die bestimmte Operation (im Sinne von AEBLI), nämlich die Verwendung von Schleifen bei Wiederholungen, in dieser neuen Situation nicht reproduziert hatte.

Die meisten **Kommentare** des Quellcodes wurden von den Gruppen erst direkt vor der Abgabe des Programms in Form seiner Speicherung auf Diskette vorgenommen und nicht bereits während der Programmierung. Die Schüler empfanden nach meiner Beobachtung das Verfassen von Kommentaren in der Regel als eher überflüssig und wenig sinnvoll. Das hängt meiner Ansicht nach mit der geringen Komplexität der von ihnen zu erstellenden Programme zusammen; die Funktionalität von Kommentierungen offenbart sich eben oft erst bei umfangreicheren Programmen. Trotzdem müssen die Schüler von Beginn an dazu angehalten werden. Im Gegensatz zu den Kommentaren wurde die formelle Strukturierung und Kenntlichmachung zusammengehöriger Teile mittels **Einrückungen und Zei-**

[53] Alle Gruppen beendeten diese Aufgabe innerhalb von zehn Minuten.

lenumbrüchen schon während der Programmierung beachtet (I5), vor der Abgabe dann nur noch verbessert. Hier scheint sich die gewonnene Übersicht schon während der Programmerstellung als hilfreich zu erweisen.
Eigene Bezeichner wurden noch nicht eingesetzt.

Bei **Arbeitsaufgabe 4** führte Gruppe 2 vor der Rechnerarbeit eine längere Diskussion zur Art des Lösungsansatzes. Gruppe 1 programmierte nach kurzer Problembesprechung parallel an zwei Rechnern, was ich zuließ, weil Kommunikation gegeben war und ich die Selbsttätigkeit nicht behindern wollte.

Für die Gruppen 3 und 4 brachte Arbeitsaufgabe 4 Schwierigkeiten mit sich. Gruppe 3 stieß in der Dokumentation von BAUM durch Nutzung des Suchbegriffes „Wiederholung“ auf das scheinbar passende nachprüfende „`until`“ (s. BAUM, 19), dessen Problematik in 4.2 schon ausführlich erläutert wurde. Die Schüler versuchten erfolglos, damit das Programm zu erstellen. Gruppe 4 erstellte anscheinend ausgehend von der Erläuterung des „`while`“ -Konstrukts bei BATES ein erstes Programm mit einer „`while(true){}`“-Endlosschleife, was nicht zum Ziel führte.
Irrtümer und deren Verbesserungen sind von mir als Unterrichtsziel vorgesehen (M2), doch hat die Herangehensweise der beiden Gruppen aus meiner Sicht etwas Problematisches. Ich stellte hier fest, dass sich bei den beiden genannten Gruppen die Problemanalyse- und Entwurfsphase mit der Implementationsphase vermischte. Durch die erst fehlerbehafteten Lösungsansätze gerieten beide Gruppen aber dennoch in eine Diskussion zur Problemanalyse (M2).

Für die Erarbeitung einer ersten Lösung bis zum ersten Test in der Testumgebung benötigten die Gruppen bei Aufgabe 4 im Durchschnitt zehn Minuten. Die Gruppen 1-3 arbeiteten mit bedingten Schleifen (FZ 2), wobei Gruppe 3 eine nachprüfende, Gruppe 1 sowie Gruppe 2 eine vorprüfende Schleife einsetzte. Gruppe 4 stellte fest, dass ihr oben beschriebener Lösungsansatz dazu führte, dass ihr Roboter zwar im Fahrbetrieb der Aufgabe entsprach, am Ziel jedoch eine nicht endende Tonfolge von sich gab.

In der aus drei Gruppenmitgliedern bestehenden Gruppe 3 schien die Gleichverteilung der Tätigkeiten während der Implementierungs- und Testphase nicht von selbst zu funktionieren, Francesco saß die meiste Zeit ohne konkrete Handlung neben Robert, der programmierte, und Marco, der den LEGO-Mindstorms-Roboter bediente. Ich bat die Gruppe darum, die einzelnen Tätigkeiten der Gruppenmitglieder auch zu tauschen. Darauf musste während der Unterrichtsreihe des Öfteren hingewiesen werden.

Die Gruppen 3 und 4 fragten unabhängig voneinander nach Hilfe bei der Konstruktion einer Bedingung, die beide Sensoren gleichzeitig abfragt. Hier half schon eine kurze Erklärung unter Verweis auf eine Tabelle bei BAUM (25) zum Thema Bedingungen, um den Schülern die weitere Arbeit zu ermöglichen.

Die Phase des Testens und Verbesserns dauerte in etwa bis zum Ende der zweiten Stunde des Dreistundenblocks, wobei auch kleinere technische Probleme auftraten[54], die Aufgaben jedoch gelöst wurden (FZ 3). Alle Gruppen holten sich sukzessive die Zusatzaufgabe 4.1, die ursprünglich nur für die leistungsstarken Schüler vorgesehen war[55]. Daraus ist weniger zu schließen, dass die Aufgaben zu leicht waren, eher schien der Leistungsunterschied der Gruppen nicht so groß zu sein bzw. sich nicht so stark auszuwirken. Nach meiner Beobachtung legte die als leistungsstark eingeschätzte Gruppe 1 mehr Wert auf Details beim Programmieren, testete häufiger und war dadurch letztlich nur unwesentlich schneller fertig. Für den Reihenverlauf sehe ich ein ungefähr gleiches Arbeitstempo als positiv an, gerade im Hinblick auf die Reflexionsphase.

In der **Reflexionsphase** verzichtete ich auf die Vorführung einer Lösung für Aufgabe 3 mit einem LEGO-Mindstorms-Roboter, weil die Gruppen in der Testphase ihre Lösungen und die der anderen Gruppen bereits ausführlich wahrgenommen hatten. Ich bat Gruppe 2 um eine Präsentation ihres Ergebnisses in Form der Erklärung ihrer Grundidee, der Darstellung ihrer Erarbeitungs- und Implementationsphase und der aufgetretenen Schwierigkeiten sowie der Vorstellung des Ergebnisquellcodes am Beamer. Gruppe 2 wurde ausgewählt, weil sie ein Ergebnis mittlerer Qualität erreicht hatte, also eine grundsätzlich richtige Lösung mit Verbesserungsmöglichkeiten. Diese schien mir geeignet um einerseits den richtigen Lösungsansatz zeigen zu lassen, andererseits Ergänzungen möglich zu machen und damit die didaktische Spannung in dieser Phase aufrechtzuerhalten.
Gruppe 2 hatte unter Nutzung der Dokumentationen (FZ 4, M1) und abgeleitet von der Aufgabenstellung direkt nach Zählschleifen gesucht. Die Schüler stellten in ihrer Präsentation einen direkten Bezug zu *Robot Karol* her: *„Naja, bei Karol gab es ja diese wiederhole-mehrmals-Schleifen und die gibt es ja hier in NQC auch.“ (Alexander).* Damit wurde deutlich, dass diese Gruppe den erwünschten nichtspezifischen Transfer im Sinne meiner Darstellung geleistet hatte (FZ 1)! Sie hatte für Problemlösung die allgemeine Operation „Zählschleife“ reproduziert und neu konkretisiert.

---

[54] So fielen z.B. Teile des Roboter aufgrund einer Kollision ab und wurden dann wieder falsch angebracht, so dass die Sensoren nicht mehr richtig funktionierten.
[55] Die Zusatzaufgabe 4.1 wurde bis zum Beginn der Reflexionsphase von keiner Gruppe beendet. Sie soll in der folgenden Veranstaltung beendet werden.

Für Gruppe 4, die den Transfer nicht geleistet hatte (s. o.), war diese Erklärung Teil der Wiederholung im Sinne der Sicherung. Ihnen wurde die Lösung durch Nennung bewusst gemacht und so die kognitive Struktur reaktiviert. Eine Reaktivierung war bei Gruppe 4 festzustellen, sie kommunizierte während der Präsentation mit Anstoßen, Tuscheln und Bemerkungen wie z.B.: „*Hättest du/hätten wir ja auch selber drauf kommen können.*"

Als Schwierigkeit bei der Arbeit nannte Gruppe 2 das undeterminierte Drehen, wobei sich weitere Schüler einbrachten, die das bestätigten und die Frage aufwarfen, ob LEGO-Mindstorms dann überhaupt sinnvoll eingesetzt werden könne[56] (FZ 5).

Gruppe 2 zeigte und erklärte dann ihr Programm am Beamer[57] (M3); die Lösung benutzte nur die äußere Zählschleife (siehe Anhang I). Die Präsentation und Erklärung des Programms durch die beiden Schüler war allerdings nicht so gehalten, dass man ihr ohne weiteres folgen konnte, es bestanden große Schwierigkeiten darin, den formalen Quellcode in verständliche und lebendige Sprache rückzuübersetzen. Sprachlich orientierten sich die Schüler an dem, was zu lesen war, die vorhandenen Kommentierungen erwiesen sich als eher ungeeignet, den Schülern bei der Präsentation zu helfen. Das Präsentieren und Darstellen, das schon bei Karol geübt wurde, fällt den Schülern grundsätzlich nicht leicht und muss weiter geübt werden.

Mehrere Schüler[58] meldeten sich zum vorgestellten Programmcode und wiesen auf die Möglichkeit hin, das Programm mit verschachtelten Schleifen zu erstellen, vorausgesetzt die äußeren Bedingungen (Boden) wären besser. Eine solche Lösung zeigte Gruppe 1 am Beamer kurz zum Vergleich. André St. sagte zur Lösungsidee: „*Das ist ja eigentlich völlig dasselbe, als wenn Karol im Viereck seine Welt abläuft und da haben wir ja auch Schleifen ineinander gesetzt. Jetzt mussten wir nur noch rauskriegen, ob es diese Zählschleifen auch in NQC gibt.*" In dieser Äußerung macht der Bezug auf *Karol* und die Verwendung des Begriffs „Zählschleife" den geleisteten Transfer deutlich.

Bezüglich des Transfers sind beide gezeigten Lösungsvarianten gültig. Die Lösung von Gruppe 2 zerlegt das Problem zwar weniger detailliert, die Transferleistung bezüglich der Kontrollstruktur Zählschleife wurde aber dennoch erbracht.

---

[56] Das beantwortete André St. bejahend mit Verweis auf Rotationssensoren.

[57] Der Beamer war an einem separaten Rechner angeschlossen und die Quellcodes befanden sich alle auf der Diskette, so dass zeitraubendes Einloggen und Anmelden der einzelnen Schüler bei der Präsentation entfiel.

[58] Das betraf André S. und André St. aus Gruppe 1 sowie Francesco aus Gruppe 3, obwohl letztere Gruppe das Ergebnis in Form von verschachtelten Schleifen nicht abgegeben hatte. Offensichtlich kam Francesco diese Idee erst nach der Fertigstellung der Lösung durch die Gruppe.

Gruppe 4 wurde um die Präsentation von **Aufgabe 4** gebeten, die mit der Vorführung der funktionalen Lösung am Roboter begann. Hier hielt ich es für besonders sinnvoll, dieser Gruppe die Präsentation mit LEGO-Mindstorms zu ermöglichen, da die anderen Gruppen die erst falsche Lösung beim Testen mitbekommen hatten. Auf diese Weise wurde das Endergebnis von der gesamten Lerngruppe (durch Beifall) gewürdigt. Gruppe 4 erläuterte danach ihren erst falschen Lösungsansatz, ihre Idee zur Korrektur („*Wir mussten als Bedingung das Auftreffen auf den Karton einbauen.*“; Eric) und stellte am Beamer ihren Quellcode vor. Ihre letztlich richtige Lösung (vgl. Anhang II) war nach der Idee der vorprüfenden Schleife entwickelt (M2).
Die Transferleistung der Gruppe 4 bezüglich bedingter Schleifen ist differenziert zu bewerten. Die Notwendigkeit einer Schleifenstruktur war der Gruppe gleich klar (Teiltransfer geleistet), Probleme hatte sie mit dem Konzept der Bedingung (die erst falsche Lösung mit der Endlosschleife verweist auf nicht geleisteten Transfer[59]). Die Frage, ob die Schleife besser vor- oder nachprüfend implementiert würde, stellte sich die Gruppe nicht, deren Wahl fiel zufällig.

Zum Vergleich wurde dieser Lösung das Programm von Gruppe 3 (mit nachprüfender Schleife, vgl. Anhang III) gegenübergestellt und um Bewertung gebeten. Robert schätzte dabei die Lösung seiner eigenen Gruppe als weniger günstig ein für den Fall, dass der Roboter gleich zu Beginn schon Berührung mit dem Karton hat (FZ 6), nicht ohne hinzuzusetzen, dass die Aufgabe trotzdem gelöst wurde[60]. André S. ergänzte im Unterrichtsgespräch die Fachtermini „vorprüfende Schleife“ und „nachprüfende Schleife“.
Die Leistung von Gruppe 3 ist mit Einschränkung als gelungener Transfer einzuschätzen. Die Abhängigkeit der Wiederholung von einer Bedingung als Konzept wiederzuentdecken und umzusetzen wurde geleistet, allein die bewusste und überlegte Wahl für einen Schleifentyp erfolgte nicht im gewünschten Maße.

Das schnelle Erkennen des eigenen Fehlers beim Vergleich der zwei Quellcodes durch Robert aus der Gruppe 3 zeigt beispielhaft die stattgefundene Reflexion in dieser Phase.
Die formelle **Struktur und Lesbarkeit des Quellcodes** wurde von der Lerngruppe für beide Ergebnisse als gut bewertet (I7). Beide Programme waren formell gut strukturiert. Die Kommentierungen unterschieden sich aber in der Ausführlich-

[59] Die letztlich doch richtige Verwendung der Bedingung wird nicht als Transferleistung verstanden; sie resultierte m. E. nicht aus dem Wiedererkennen des von *Karol* bekannten Konzeptes.
[60] Spätestens hier zeigte sich, dass Aufgabenstellung 6 im Hinblick auf Vor- und Nachprüfung noch genauer konstruiert werden müsste.

keit[61]. Die Schüler sollten einschätzen, welche Variante ihnen denn ein schnelleres Begreifen des Programms ermöglichen würde. Dieser etwas zu offene Impuls führte zu gegensätzlichen Meinungen unter den Schülern. Hier müssen die Kriterien unbedingt deutlicher gemacht werden. Möglich wäre es vielleicht außerdem, Quelltexte von Schülern anderer Gruppen auf Basis der Kommentierungen erklären zu lassen.
Insgesamt hat die Reflexionsphase ihren angedachten Zweck erfüllt, nämlich verschiedene Ergebnisse und Lösungsideen darzustellen und zu bewerten sowie Konzepte und Begriffe zu wiederholen. Anhand der vorgestellten Ergebnisse konnte der Transfer festgestellt werden, eine genauere Einschätzung der Transferleistung wurde durch die Erklärungen der Schüler möglich.

Die für diese drei Stunden vorgesehenen Feinlernziele sind erreicht worden, Schwierigkeiten gab es in diesen Stunden noch mit den für die Reihe vorgesehenen Groblernzielen M3 (Präsentation) und I7 (Bewertung der Lesbarkeit) aus bereits genannten Gründen.

6 Gesamtreflexion

Diese Arbeit untersuchte, inwieweit das Roboter-Baukastensystem LEGO-Mindstorms in Verbindung mit der Programmiersprache NQC dazu geeignet ist, zur Anwendung von algorithmischen Grundstrukturen zu dienen, und auf der Basis des mit *Robot Karol* erworbenen Vorwissens den Kontext für Rekonstruktion (Transfer) bzw. Reaktivierung (Sicherung) entsprechender kognitiver Strukturen zu schaffen. Sowohl LEGO-Mindstorms als auch NQC haben sich dabei als prinzipiell geeignet erwiesen. Beobachtet werden konnten direkte Verbindungen, welche die Schüler zwischen LEGO-Mindstorms und *Robot Karol* herstellten. Die konzeptionelle Übertragung von algorithmischen Strukturen wurde unterstützt durch inhaltliche Ähnlichkeiten, die den Bildschirmroboter Karol mit dem realen LEGO-Mindstorms-Roboter verbinden.

Dabei haben sich auch sachstrukturelle Besonderheiten von LEGO-Mindstorms (z.B. Drehungsproblem) und NQC (z.B. Boolesche Verknüpfungen) herausgestellt, LEGO-Mindstorms ist eben mehr als ein in die reale Welt versetzter *Karol*. Diese Schwierigkeiten erforderten genaue Planung sowie detaillierte Beachtung der Voraussetzungen, verhinderten aber die Transferleistungen nicht .

[61] Die Lösung von Gruppe 4 kommentierte so gut wie jede Zeile, was der Verständlichkeit des Programms eher abträglich war. Das ist aus meiner Sicht eine Form von Übergeneralisierung der Operation „Kommentieren“.

Als herausragende Qualität des Materials LEGO-Mindstorms erwies sich die damit erzielbare **Motivation**, welche ein hohes Maß an Selbsttätigkeit auch über einen längeren Zeitraum möglich machte. Diese Selbsttätigkeit ist die Form des idealen Lernens im Sinne der Handlungsorientierung und hat sich auch in dem durchgeführten Unterricht bewährt. Beinahe alle Schüler haben in einer schriftlichen Kurzevaluation[62] zum erlebten Unterricht die Selbsterarbeitung von NQC als sehr positiv genannt. Die hohe Motivation der Schüler verringerte das Problem der nachmittäglichen Lage und dreistündigen Dauer der Veranstaltungen; die Schüler vergaßen teilweise sogar, ihre Pausen in Anspruch zu nehmen. Als optimal haben sich die Dreistundenblöcke trotzdem nicht erwiesen, gerade in den Reflexionsphasen ließ die Konzentrationsfähigkeit mancher Schüler nach.

Die Schüler waren in der Lage, die gestellten **Aufgaben** auf Basis der gegebenen Dokumentationen (M1) und unter Nutzung des „Bricx-Command-Centers“ (I1) zu lösen (I2, I3). Dabei wurden die Zusatzaufgaben in der Regel von allen Gruppen begonnen (M4), die Konzeption des „Zusatzes“ griff nicht in geplantem Maße, weil die verbindlichen Aufgaben schnell gelöst wurden. Diese hätten vielleicht schwieriger sein können. Andererseits wirkte sich der Erfolg, schon die *Zusatzaufgaben* zu bearbeiten, bei allen Gruppen positiv auf die Motivation aus. Eine Lösung wäre, weitere Zusatzaufgaben bereit zu halten. Das geplante „Abfedern“ der unterschiedlichen Leistungsstärken der Gruppen durch die Zusatzaufgaben hätte hier nicht erreicht werden können, wenn die angenommene leistungsstärkere Gruppe 1 tatsächlich viel schneller gewesen wäre als die anderen Gruppen.

Die starke Vorstrukturierung des Unterrichts durch die sehr genauen Aufgabenstellungen war erfolgreich, da so die einzelne Transferleistung genau beobachtet werden konnte. Möglich gewesen wären aber auch offenere Aufgaben, bei denen allerdings Auswertung und Vergleich der Lösungen aufwändiger, nichtsdestotrotz durchführbar gewesen wären. Der Vorteil komplexerer und weniger detailliert strukturierter Aufgaben hätte in sich stärker unterscheidenden Lösungen mit höherem Vergleichswert sowie einem deutlicher herausgestellten Sinn der Nutzung von Prozeduren und Kommentierungen gelegen. Ein Schüler merkte auch in der Kurzevaluation kritisierend an, „ *... dass wir immer vorgegebene Aufgaben hatten*.“ Die verwendeten Aufgabenstellungen selbst haben sich als didaktisch funktional erwiesen, die jeweils sinnvolle algorithmische Struktur wurde in der Regel erkannt und verwendet (I4). Schwierigkeiten im Praxistest brachten Aufgabe 4, die nicht deutlich genug zur bewussten Wahl vor- oder nachprüfender Schleifen

[62] Folgende Fragen wurden den Schülern gestellt: Was hat Ihnen am Unterricht gefallen? Was hat Ihnen am Unterricht nicht gefallen? Was hätten Sie für Verbesserungsvorschläge zum erlebten Unterricht?

zwang, Aufgabe 6, bei der die Lösung mancher Gruppen in einer schwer nachzuvollziehenden Kaskade von Pieptönen unterging, und Aufgabe 3, die bereits in ihrer Anlage problematisch war. Letztere ist unter dem Aspekt der Praxisorientierung mit Sicherheit sinnvoll, im speziellen Blickwinkel von Transfer und Sicherung einer ganz bestimmten Kontrollstruktur sollte sie jedoch nicht verwendet werden, solange kein Rotationssensor eingesetzt werden kann.

Die gegebenen **Dokumentationen** erforderten von den Schülern anfangs eine gewisse Orientierung und ein Einlesen. Die Möglichkeit der Stichwortsuche im elektronischen Dokument erwies sich als essentiell, insbesondere bei BAUMS Dokumentation. Diese war das meistgenutzte Dokument der Schüler für die Einarbeitung. BATES wurde weniger genutzt, ich stellte jedoch fest, dass André S. sich selbst ein Exemplar von BATES' Referenz ausgedruckt hatte und damit ab Stunde 7 arbeitete. Die meisten Schüler empfanden anscheinend sowohl die Englischsprachigkeit als auch die Komprimiertheit der Informationen bei BATES als weniger zugänglich. Trotzdem würde ich BATES' Referenz immer auswählen, bei in der Algorithmik fortgeschrittenen Schülern ist sogar denkbar, nur diese Referenz zur Verfügung zu stellen.

Die **Sozialform** der Partnerarbeit erwies sich als weit gehend produktiv, zum einen, weil inhaltlich miteinander beraten werden konnte und wurde (M5), zum anderen, weil ein LEGO-Mindstorms-Roboter auch für zwei gleichzeitig tätige Personen genug Handlungsmöglichkeiten bietet. Als schwieriger stellte sich die Arbeit der Dreiergruppe heraus, bei welcher anscheinend gruppendynamische Prozesse die Arbeitsteilung regelten und durch Lehrersteuerung für eine Gleichverteilung gesorgt werden musste. Trotz dieser Schwierigkeit ist die Dreiergruppe im Gegensatz zu einem einzeln arbeitenden Schüler als die bessere Variante anzusehen, denn die Kommunikation der jeweiligen Partner erwies sich in den Arbeitsprozessen sowohl bezüglich der inhaltlichen Lösung als auch für das soziale Verhalten als tragend.

Der beabsichtigte **Transfer** grundlegender algorithmischer Strukturen wurde im durchgeführten Unterricht erreicht; ohne vorherige Kenntnis der zu transferierende Inhalte wäre die adäquate Lösung der Aufgaben nicht möglich gewesen. Zusätzlich ermöglichte die Darstellung bzw. Präsentation der Ergebnisse in den Reflexionsphasen einen Einblick in die Lösungswege und die genauere Beurteilung der Transferleistung bei den präsentierenden Schülern. Die Lösungsidee, das Lösungskonzept verrät, inwiefern der nichtspezifische Transfer geleistet wurde.

Sicherung wurde in besonderem Maß als Prozess aufgefasst, der sich durch Wiederholung und Übung konstituiert. Die Wiederholung von Begriffen und Konzep-

ten in der Reflexionsphase musste, um Sicherung erreichen zu können, vom Lehrer gelegentlich durch Impulse gesteuert werden („Erklären Sie noch einmal den Unterschied...“). Problematisch bezüglich der Reflexionsphase ist die relative Passivität der nicht präsentierenden Gruppen, auch unter dem Aspekt der nachlassenden Konzentration am Ende der Veranstaltung. Zu überlegen wäre, ob nicht durch eine Form schriftlichen Arbeitens die Effektivität diese Phase noch besser abgesichert werden könnte. Ein Arbeitsauftrag für alle könnte z.B. lauten: „Vergleicht die beiden Programme und notiert euch wesentliche Unterschiede.“
Als erfolgreich für die Sicherung erwies sich die durch die in der Reihe folgenden Arbeitsaufgaben gegebene Möglichkeit der Übung, der Anwendung des Wiederholten. Unter dem Sicherungsaspekt wäre auch eine Reflexionsphase nach jeder einzelnen Aufgabe zu rechtfertigen gewesen, was aber den Unterricht zu stark fragmentiert, die Arbeitsphasen verkürzt und die Selbstverantwortung (bezüglich der Zeit) vollständig eingeschränkt hätte.

Das Erreichen der beabsichtigten Groblernziele wurde teilweise schon dargestellt; die zentralen algorithmischen Ziele (I3, I4, auch I1, I2) wurden erreicht. Schwierigkeiten gab es in der Bewertung der Lesbarkeit (I7). Der Effizienzvergleich von Lösungen (I6), nicht durchgeführt in den dargestellten und analysierten Stunden, funktionierte aufgrund der transparenten Vergleichsmöglichkeit Quellcodelänge sehr gut und reaktivierte die Schüler. Die methodischen und sozialen Lernziele wurden prinzipiell erreicht, besonders gut funktionierte das gemeinsame Erarbeiten in Partnerarbeit (M5). Wie schon erwähnt, gab es Schwierigkeiten bei den Präsentationen (M3), die weiter geübt werden.

Dem Einsatz von LEGO-Mindstorms im Bereich der Algorithmik sind zwar durch die Hardware Komplexitätsgrenzen gesetzt. Anspruchsvollere algorithmische Themen (komplexe Datenstrukturen, Rekursion) sind kaum oder gar nicht mit LEGO-Mindstorms umzusetzen. Und auch schon bei den hier thematisierten algorithmischen Grundstrukturen wirkten sich Hardwarebeschränkungen in direkter Weise auf die Programmierung aus. Trotzdem erwies sich der Unterricht mit LEGO-Mindstorms unter dem gewählten algorithmischen Schwerpunkt als sehr intensiv und erfolgreich, die Schüler setzten sich in konzentrierter und umfassender Weise sowohl mit den Unterrichtsmaterialien als auch mit der Algorithmik auseinander und erreichten die wesentlichen Lernziele. So scheint das Zitat von John Dewey, welches SCHUBERT/SCHWILL ihrer Didaktik der Informatik voranstellen, auch auf den durchgeführten Unterricht zuzutreffen: „*To be playful and serious at the same time is possible, in fact it defines the ideal mental condition*.“

7 Literatur

ABEND, M.: *Robotik und Sensorik. Selbständige Entwicklung „unscharfer“ Algorithmen zur räumlichen Orientierung unter Verwendung des Lego-Mindstorms-Systems. Schriftliche Prüfungsarbeit zur zweiten Staatsprüfung für das Amt des Studienrates.* Berlin 2001.

AEBLI, H.: *Zwölf Grundformen des Lehrens*. Klett Stuttgart 1985.

BATES, M.: *NQC Programmer's Reference Sheet*. 2001.
Quelle: http://cosmos.ucdavis.edu/2002/robots/NQC_Ref.pdf (19.3.04)

BAUM, D.: *NQC-Programmieranleitung. Version 2.3 rev 1*.
Quelle: http://lug.mfh-iserlohn.de/lego/NQC-Guide.2.3r1.German.pdf (19.3.04)

BÖNSCH, M: *Üben und Wiederholen im Unterricht*. Ehrenwirth München 1988.

BÖNSCH, M: *Wie sichere ich Ergebnis und Erfolg in meinem Unterricht?* NDSV Essen 1967.

EISENHUT, G.; HEIGL, J.; ZÖPFL, H.: *Üben und Anwenden. Zur Funktion und Gestaltung der Übung im Unterricht.* Klinkhardt Bad Heilbrunn 1981.

ENGESSER, H. (Hrsg.): *Duden Informatik. Ein Sachlexikon für Studium und Praxis*. Dudenverlag Mannheim 1993.

LENZEN, D. (Hrsg.): *Enzyklopädie Erziehungswissenschaft.* Bd.3 Stuttgart 1986.

FILIUS, S.; KÜHN, S., SCHELLHAMMER, S.: *Einführung in Lego Mindstorms*. 2004.
Quelle: http://wwwmath.uni-muenster.de/info/u/lammers/EDU/ws03/Landminen/Abgaben/Gruppe1/Ausarbeitung.pdf (31.3.04)

GAUDIG, H.: *Die Schule im Dienste der werdenden Persönlichkeit*. Leipzig 1917 (zit. nach REBLE, A: *Die Arbeitsschule*. Klinkhardt Bad Heilbrunn 1963).

GIESECKE, H.: *Was ist ein „Schlüsselproblem"? Anmerkungen zu Wolfgang Klafkis neuem Allgemeinbildungskonzept.* Neue Sammlung 37 1997.

GRIMUS, M. u. a. (Hrsg.): *Evaluierungsprojekt "Neue Medien in der Grundschule"*. Österreichische Computer Gesellschaft 2000.

GUDJONS, H.: *Didaktik zum Anfassen*. Klinkhardt Bad Heilbrunn 1997.

HIRSCH, M.; MAGENHEIM, J.; REINSCH, T.: „*Zugänge zur Informatik mit Mindstorms.*“ In: LogIn 20 (2000) S. 34-46.

HUBWIESER, P.: *Didaktik der Informatik*. Springer Berlin 2003.

JANK, W.; MEYER, H.: *Didaktische Modelle*. Cornelsen Frankfurt am Main 1994.

KNUDSEN, J.B.; NOGA, M.L.: *Das inoffizielle Handbuch für LEGO-Mindstorms-Roboter*. O'Reilly-Verlag Köln 2000.

MESSNER, H.: *Wissen und Anwenden. Zur Problematik des Transfers im Unterricht*. Klett Stuttgart 1978.

MEYER, H.: *Unterrichtsmethoden*. II. Praxisband. Cornelsen Frankfurt am Main 1987.

MICROSOFT (Hrsg.): *Encarta Enzyklopädie*. Microsoft 2002.

MÜßENER, G. (Hrsg.): *Didaktische Texte zu Unterricht und Erziehung in Wissenschaft und Schule*. Deimling Wuppertal 1991.

OVERMARS, M.: *Programmieren von LEGO-Mindstorms-Robotern mit NQC*. Version 3.03 Deutsche Übersetzung von Martin Breuer.
Quelle: http://mv-sirius.m.fh-offenburg.de/robotik/NQCTutorial_d.pdf (19.3.04)

PAPERT, S.: *Mindstorms: Children, computers and powerful ideas*. Basic Books New York 1980.

PATTIS, R.: *Karel the Robot: A Gentle Introduction to the Art of Programming.* (2nd Edition) Wiley Indianapolis 1994.

PETERßEN, W.H.: *Handbuch Unterrichtsplanung. Grundfragen, Modelle, Stufen, Dimensionen*. Ehrenwirth München 1991.

PIAGET, J.: *Psychologie der Intelligenz*. Klett Stuttgart 1980 (orig.1947).

ROSENBACH, M.: *Die Sicherung des Lernerfolges*. 2003.
Quelle: http://bebis.cidsnet.de/weiterbildung/sps/allgemein/bausteine/struktur/lernerfolg.htm (28.2.04)

SCHOLZE-STUBENRECHT u. a. (Hrsg.):*Duden. Fremdwörterbuch*. Dudenverlag Mannheim 1999.

SCHOLZE-STUBENRECHT u. a. (Hrsg.): *Duden. Die deutsche Rechtschreibung.* Dudenverlag Mannheim 2001.

SCHUBERT, S.; SCHWILL, A.: *Didaktik der Informatik*. Spektrum Akademischer Verlag Heidelberg 2004.

SENATSVERWALTUNG FÜR SCHULE, BERUFSBILDUNG UND SPORT (Hrsg.): *Vorläufiger Rahmenplan für Unterricht und Erziehung in der Berliner Schule - Gymnasiale Oberstufe - Fach Informatik*. Berlin 1993.

STOLT, M.: *Roboter im Informatikunterricht* (Studienarbeit). 2001.
Quelle: http://www.webniks.de/websites/www.buechergilde-hamburg.de/stolt/lego/ (19.3.04)

Weitere verwendete Internetquellen:

Michael GASPERIS Webseite
http://www.plazaearth.com/usr/gasperi/lego.htm (19.3.04)

Webseite des Lejos Projekts
http://www.lejos.sourceforge.net (19.3.04)

PBforth-Homepage
http://www.hempeldesigngroup.com/lego/pbForth/homePage.html (19.3.04)

Dave BAUM. im Interview mit dem LMSM (Lego Mindstorms Monthly)
http://www.lmsm.info/back-issues/1102/Dave_Interview.html (19.3.04)

Pressemitteilung zu Jitter
http://ais.gmd.de/de/pm/011012.html (19.3.04)

Bricx Homepage
http://sourceforge.net/projects/bricxcc/ (19.3.04)

8 Anhang

Anhang I (Aufgabe 3: Kiste umfahren, Lösung Gruppe 2)

```
//Aufgabe 3 Gruppe 2 (Alex+Claudius)

task main()  //Start
{

repeat(2)    //Gruppe2-Ton
{
PlayTone(880, 10);
Wait(50);
}

repeat(2) {         //fährt 2x
SetOutput(OUT_A+OUT_C,1); //Speed langsam
OnFwd(OUT_A+OUT_C);       //losfahren
Wait(200);          //Warte 2 Sekunden
OnRev(OUT_C);       //Motor 2 aus => linksdrehen
Wait(70);           //warte 0,7 Sek.

OnFwd(OUT_A+OUT_C);  //2. Strecke
Wait(200);
OnRev(OUT_C);
Wait(77);

OnFwd(OUT_A+OUT_C); //3.Strecke
Wait(220);
OnRev(OUT_C);
Wait(77);

OnFwd(OUT_A+OUT_C); //4. Strecke
Wait(200);
OnRev(OUT_C);
Wait(70);
}
```

Anhang II (Aufgabe 4: Hindernis, Lösung Gruppe 4)

```
task main()
{
Wait(100);
PlaySound (SOUND_DOUBLE_BEEP);       //Erkennungssound
PlaySound (SOUND_DOUBLE_BEEP);       //Erkennungssound 2.Teil

SetSensor(SENSOR_1,SENSOR_TOUCH);     //berührungssensor
OnFwd(OUT_A + OUT_C);                //Motoren A und C

while (SENSOR_1 == 0)                   //solange dass...
 {
 PlaySound(SOUND_CLICK);        //Ton
 Wait(50);                     //warten
      }

 Off(OUT_A+OUT_C);               //wenn sensor gedrückt

PlaySound(SOUND_UP);             //piepen
PlaySound (SOUND_DOUBLE_BEEP);
Wait(200);

 OnRev(OUT_A+OUT_C);               //zurücksetzen
 Wait(60);
 Off(OUT_A+OUT_C);        //Motoren aus
 Wait(50);
  }
```

Anhang III (Aufgabe 4: Hindernis, Lösung Gruppe 3)

```
//Gruppe 3: Francesco - Marco - Robert

  task main()
  {
      SetSensor (SENSOR_1,SENSOR_TOUCH);  //definiert den Tastsensor


      do {
       OnFwd(OUT_A+OUT_C);                 //beginnt mit dem Fahren
       PlayTone(262,40);                   //Piepen alle halbe Sekunde
       Wait(50);
                 } while (SENSOR_1 != 1);

       Off(OUT_A+OUT_C);                    //schaltet Motoren ab

       repeat (3){                         //Überraschung!
        PlayTone(262,40);
       }
       Wait (200);
       OnRev(OUT_A+OUT_C);

       Wait (200); //  1 Meter in 2 Sekunden

       Off(OUT_A+OUT_C);                    //schaltet Motoren ab
  }
```