Kurs 4: Wie Roboter ticken

Kurs-Atmosphäre

Alle Teilnehmer wie Kursleiter waren sehr motiviert. Es herrschte rege Arbeitsstimmung, was manchmal auch ein verspätetes Essen oder Abendschichten zur Folge hatte. Bis zuletzt arbeiteten wir mit Begeisterung an den Robotern.

Die Hilfsbereitschaft und die Kompetenz der Kursleiter waren maßgebend für den Erfolg unserer Arbeit, deshalb seien unseren Kursleitern an dieser Stelle nochmals ein dickes Lob und auch ein Dankeschön ausgesprochen. Aber auch unter den Teilnehmern gab es auch immer jemanden, der bereit war mit Rat und Tat zur Seite zu stehen. Teamwork wurde bei uns sehr groß geschrieben: Vor der eigentlichen (Bau- und Programmier-) Arbeit stand immer die gemeinsame Planung. Nichts wurde dem Zufall überlassen. Obwohl wir in unserem Robotikkurs nochmals in einzelne Untergruppen unterteilt waren, half man jedem Kursteilnehmer, der Hilfe benötigte. Aber auch während der Arbeit herrschte immer eine rege Kommunikation zwischen den Teammitgliedern. Ohne diesen Teamgeist wäre eine Realisierung unseres Projekts nie möglich gewesen.

Man arbeitete konzentriert, zielorientiert, begeistert und mit vollem Elan, doch fehlte gleichzeitig nicht der nötige Spaß in unserem Kurs. Wir fühlten uns vom ersten Moment an im Kurs wohl und freuten uns immer auf die Stunden zusammen. Wir waren ein eingespieltes und überaus erfolgreiches Team, es war einfach genial.

Bild 1: Test des Transportroboters

Bild 2: Die Fischertechnik Gruppe

Bild 3: Die Transportrobotergruppe und ihre Kompetenz

Die Aufgabe

Für die zwei Wochen in Adelsheim stellten uns unsere Kursleiter zwei Platten eines Lego-Mindstorms-Wettbewerbs zur Verfügung. Auf der Platte sind einige schwarze Linien eingezeichnet, auf denen kleine Lego Tonnen positioniert wurden.

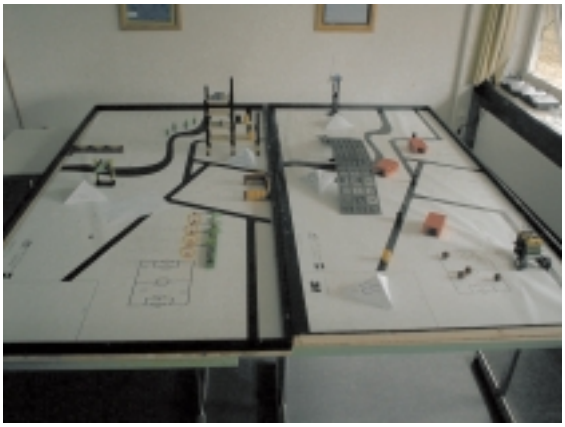


Bild 1: Unsere Arbeitsplatte

Der Lego-Sucher-Roboter hatte die Aufgabe, die schwarzen Linien „abzufahren“ und dabei die Tonnen einzusammeln. Die Tonnen sollte er dann an einen festen Sammelpunkt auf der Platte transportieren.

Der dort stationierte Fischertechnik-Roboter hatte die Aufgabe, die Tonnen mit seiner Zange zu greifen und auf die andere Platte zu transportieren, wo der Lego-Transport-Roboter schon bereitzustehen hatte. Danach galt es, die Tonnen auf den Transporter zu stellen.

Der Transportroboter sollte dann auf der zweiten Platte entlang der Linien zu einem Regal fahren. Am Regal angekommen, musste er die Tonne mithilfe eines Förderbandes abladen, das an der Ladefläche angebracht war.

Einige der Kursteilnehmer hatten ihren Aufgabenteil schon vorzeitig beendet, so blieb noch etwas Zeit, die Aufgabe um zwei weitere kleine Roboter zu erweitern: Es entstanden der mobile Lego-Roboter „Blackbox“, der eine Brücke herunterklappte, über diese hinüberfuhr, eine Windmühle in Kraft setzte und wieder zum Ausgangspunkt zurückfuhr, und der „Musikroboter“, der eine Melodie spielte.

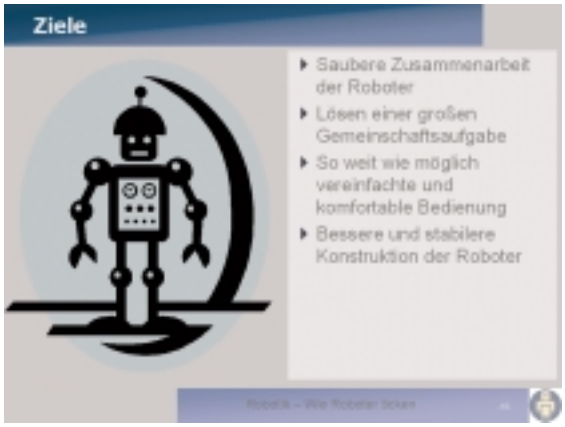


Bild 2: Unsere Gemeinschaftsziele

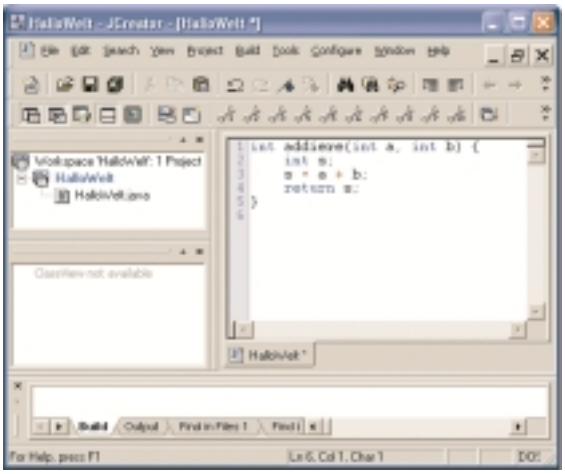
Die Programmiersprache Java

Einführung

Bei der Science Academy lernten wir, wie man Roboter entwickelt. Dazu gehört auch die Programmierung. Wir haben uns im Vorbereitungswochenende in Donaueschingen auf die Programmiersprache Java festgelegt.

Grundlegendes

Bevor unsere Exkursion in die Fachbegriffe geht, wollen wir uns mit dem Grundstein der Java-Programmierung befassen, der Entwicklungs-umgebung.



Die Entwicklungsumgebung

Die Fläche, in der bereits ein paar "seltsame Sätze" stehen, ist die Fläche, in welcher der sogenannte Quellcode steht. Als Quellcode bezeichnet man das Programm, wenn es noch als für Menschen lesbarer Text geschrieben ist. Später wird der Quellcode kompiliert, d.h. er wird in eine für den Computer verständliche Form gebracht und in einer eigenen Datei, einer sog. Class-Datei, gespeichert. Ein weiteres Programm, der Interpreter, liest diese Datei in den Programmspeicher des Computers ein, danach kann das Programm ausgeführt werden.

Java ist Klasse

Ein Java-Programm besteht aus Klassen, sie enthalten Klassen- und Objektvariablen, Konstruktoren und Methoden. Eine Klasse ist gewissermaßen ein „Bauplan“ eines Objekts. Durch Aufruf des Konstruktors wird eine Klasse instanziiert, d.h. aus dem Bauplan wird ein reales Objekt. Beachtet werden muss jedoch, dass der Konstruktor den selben Namen trägt, wie die Klasse in der er sitzt. Anweisungen in Java enden immer mit einem Semikolon. Variablen in Java dienen zum Speichern von Werten. Es gibt unterschiedliche Variablentypen, wie z.B. Ganzzahl-Typen, Gleitkoma-Typen oder Zeichen-Typen.
Methoden verändern die Werte von Variablen oder rufen andere Methoden auf. Sie bestehen aus:

- optionalen Eingabeparametern
- einem optionalen Rückgabewert
- Anweisungen in einem geschweiften Klammernpaar.

Hier eine Beispielmethode:

```
int addiere(int a, int b) {
    int s;
    s = a + b;
    return s;
}
```

Diese Beispiel-Methode hat zwei Ganzzahl-Parameter a und b. Sie werden in der Methode addiert und in der Variablen s gespeichert. Der Wert von s wird dann als Rückgabewert an die aufrufende Klasse zurückgegeben. Dies geschieht mit dem Schlüsselwort return.

Allgemeines zu LEGO-Robotern

Der Arbeitsraum der Roboters

Die Roboter agierten auf einer Folie der Lego-League. Auf ihr waren schwarze Linien aufgedruckt, sowie einige Hindernisse (Brücke, Häuser) angebracht. Die Größe betrug ca. 3m_.

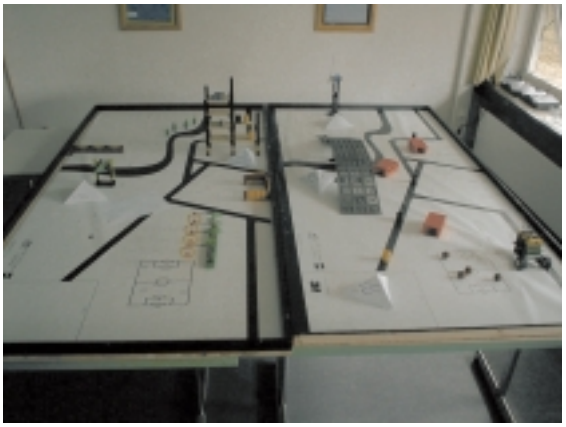


Bild 1: Die Folie der Lego-League

Der RCX

Dieser Baustein ist das Herzstück der Roboter. Man hat die Möglichkeit mit dem PC Programme zu schreiben, per IR-Schnittstelle in den RCX zu übertragen und die Programme dann autonom vom RCX ausführen zu lassen. Der RCX wird mit 6 Mignon Batterien betrieben und verfügt über 3 Sensoreingänge und 3 Motorausgänge.
Zur Programmierung des RCX wird vom Hersteller eine Software mitgeliefert, doch weil diese Software nicht so leistungsfähig war wie die professionelle Programmiersprache „Java“, verwendeten wir die mitgelieferte nicht. Mit der Applikation LeJOS des JCreator schrieben wir deshalb die Programme und übertrugen sie anschließend über eine Infrarotverbindung auf den RCX (Robotic Command Explorer).

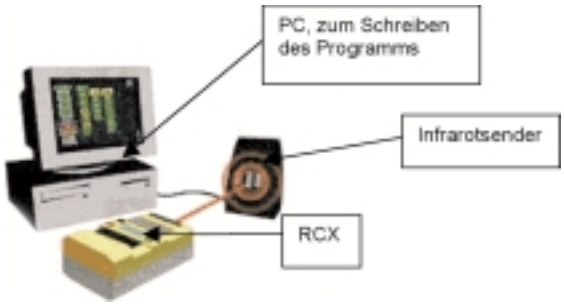


Bild 2: Übertragung des Programms auf den RC

Bild 3: Der RCX und seine Ein/Ausgänge

Motoren

Bild 3: Ein Motor

Electronic Technic Mini-Motor 9VMotoren dieses Typs treiben die Roboter an.

Der Berührungssensor

Bild 4: Ein Berührungssensor

Der Berührungssensor registriert Berührungen und gibt dann Signale an den RCX weiter.

Der Infrarotsensor

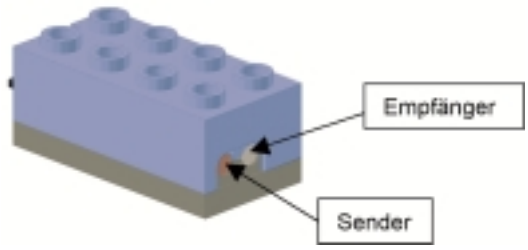


Bild 5: Ein Lichtsensor (IR)

Dieser Lichtsensor registriert verschiedene Lichtwerte. Er sendet und empfängt Licht.

Der Rotationssensor

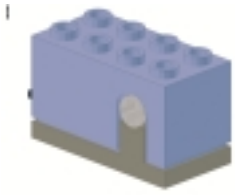


Bild 6: Ein Rotationssensor

Diese Sensoren zählen die Umdrehungen einer Achse und geben die gemessenen Werte an den RCX weiter.

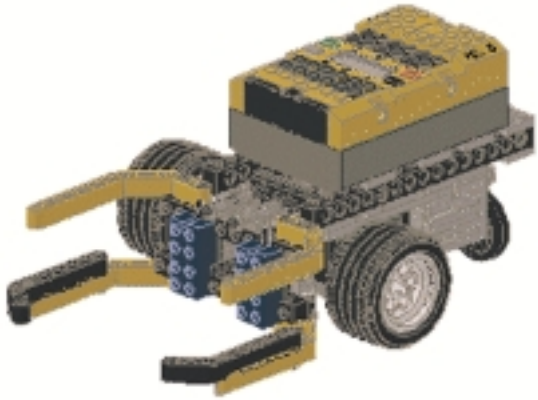
Außer diesen vorgestellten Bausteinen wurden natürlich noch zahlreiche andere Bauteile verwendet, die wir jedoch nicht alle vorstellen können.

Sucher-Robo

Sucher-Robo-Gruppe (von links nach rechts): Michael Rehermann, André Dau, Armin Richter und Matthias Groß

Aufgabe des Roboters:
Die Aufgabe des Roboters bestand darin, Tonnen, die zufällig auf einer schwarzen Linie verstreut standen, einzusammeln und zu einem Lieferband zu bringen.

Aufbau des Roboters:



Der Sucher-Robo besaß zwei blaue Lichtsensoren; zwei Motoren, zwei große Antriebsräder, ein kleines frei bewegliches Hinterrad ohne Antrieb, einen RCX-Baustein und eine Greifarmkonstruktion mit integriertem Berührungssensor.

Bau

Die Anforderungen:

Um die uns gestellte Aufgabe zu erfüllen, musste unser Roboter sehr kompakt und stabil gebaut sein und einen kleinen Wenderadius haben, damit er beim Wenden mit seinen Greifarmen nicht an Hindernissen stieß. Des weiteren durfte der Roboter die eingesammelten Tonnen nicht wieder verlieren. Auch musste die Gabelkonstruktion so verwirklicht werden, dass bei einer Berührung der Wand bzw. eines Hindernisses der integrierte Berührungssensor ausgelöst wurde. Das Getriebe musste so übersetzt werden, dass der Roboter sich nicht zu schnell bewegte. Da sich diese

Anforderungen nur bedingt unter einen Hut bringen ließen, waren wir gezwungen beim Bau einige Kompromisse einzugehen.

Allgemein zum Bau:
Unser Bau lehnte an eine bereits fertige Bauanleitung des Modells “Tippy“ (Lit.: Brian Bagnall, Lego Mindstorms Programming, Prentice Hall, 2002) an, doch mussten wir natürlich noch eine Menge Veränderungen vornehmen, sodass im Endeffekt ein komplett neues Modell entstand. Zum Beispiel besaß unser Modell im Gegensatz zum Tippy eine Gabel um die Tonnen einsammeln zu können. Der komplette Bau des Tippy war nicht kompakt genug, um unseren Ansprüchen gerecht zu werden. Zusätzlich mussten wir noch einen Berührungssensor sowie zwei Lichtsensoren unterbringen.

Unser Roboter besaß zwei Motoren für je ein anzutreibendes Rad, die bei einer Geradeausfahrt beide in die gleiche Richtung und mit gleicher Geschwindigkeit drehten. Bei Drehungen nach rechts oder links bewegten sich die Motoren in entgegengesetzter Drehrichtung. So konnte sich unser Roboter auch auf der Stelle drehen oder in die gleiche Richtung mit unterschiedlicher Geschwindigkeit um eine Kurve fahren.
Um sich orientieren zu können, besaß unser Roboter zwei Lichtsensoren. Diese waren im Abstand von ca. 2 cm zentral in der Gabel montiert. Sie hatten die Aufgabe den aktuellen Reflektionswert des vom Sensor ausgesendeten und vom Untergrund reflektierten Lichts zu messen. Auf Grund der Tatsache, dass ein schwarzer und ein weißer Untergrund verschiedene Reflektionswerte zurückwerfen, konnte der Roboter, je nach Programmierung z.B. an der schwarzen Linie entlang fahren.

Um Hindernissen bzw. Wänden auszuweichen, spielte der eingebaute Berührungssensor eine wesentliche Rolle. Dieser war direkt hinter der Gabel befestigt und hatte die Aufgabe Kollisionen, die an der Gabel stattfanden, an den RCX zu melden. Dieser reagierte dann mit dem entsprechenden Befehl an die Motoren (näheres später).

Probleme bei Bau/der Konstruktion:
Während des Baus erwiesen sich einige Detailverwirklichungen als Probleme, die nur schwer zu lösen waren:

<u>1) Die Gabelkonstruktion:</u>
Die Konstruktion an sich war relativ einfach aufzubauen, jedoch war es nicht einfach, die Größe, Form und den Aufbau der Gabel so zu bewerkstelligen, dass der Wenderadius, die Kompaktheit des Roboters, sowie die Aufnahme-kapazität der Gabel nicht darunter litt. Beim Aufbau war wichtig, dass die Gabel “schwingend“ aufgehängt war, da sonst der sich im Anhang an die Gabel befindliche Berührungssensor nicht auslösen würde.

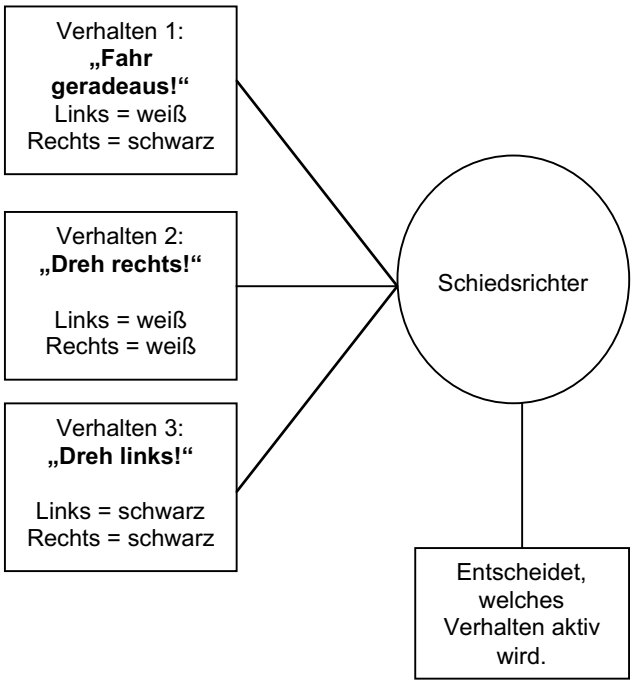
Wir hatten noch ein weiteres konstruktionsbedingtes Problem mit der Gabel. Die eingesammelten Tonnen wurden nach dem Umdrehen an der Wand nicht komplett mitgenommen. Daraufhin ergänzten wir die Gabel um die zwei um 45 Grad gekrümmten Teile, die ein Verlieren der eingesammelten Tonnen nicht mehr möglich machten.

2) Das Getriebe:
In Folge der sehr kompakten Bauweise des Roboters erwies es sich als schwierig, das recht große Getriebe auf dem begrenzten Platz unterzubringen. Das Getriebe musste die richtige Getriebeübersetzung aufweisen, sodass der Roboter sich nicht zu schnell bewegte. Sonst würde er an Kurven und Ecken, an denen er nur die Richtung korrigieren sollte, über den Kurs hinausfahren und dann umdrehen, aber das war ja nicht der Sinn der Sache. Trotzdem sollte das Ganze noch eine akzeptable Stabilität aufweisen.

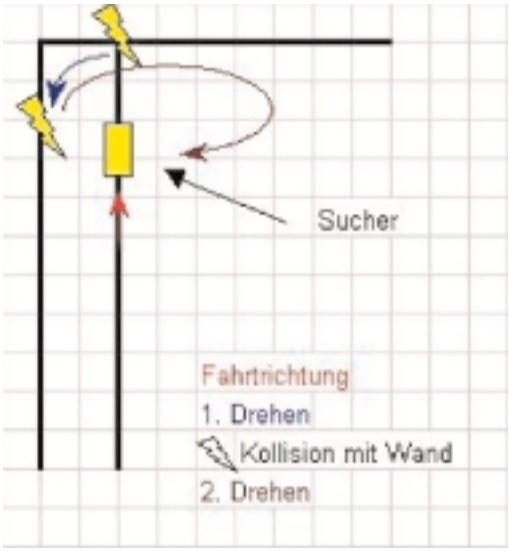
3) Der Wenderadius:
Beim Bau war unter anderem wichtig einen möglichst kleinen Wenderadius zu erreichen. Deswegen nahmen wir 2 antreibende Räder und ein frei bewegliches Spornrad, sodass sich der Roboter auf der Stelle drehen konnte. Auch setzten wir die 2 Antriebsräder direkt hinter die Gabel, um einen möglichst kleinen Wenderadius zu erreichen, damit die Gabel oder das Heck nicht an der Wand bzw. an einem Hindernis streifte.

Die Programmierung:
Bei der Programmierung arbeiteten wir hauptsächlich mit dem Interface Behavior (genaueres steht im Kapitel „Java“).

Zuerst einmal musste der Roboter der schwarzen Linie folgen können. Dieses bewerkstelligten wir, indem wir ihn am linken Rand entlangfahren ließen. Der linke Sensor sollte im Idealfall immer einen Weißwert melden, der rechte einen Schwarzwert. War dies der Fall, so wurden beide Motoren auf vorwärts gestellt. Kam der Roboter nach links ab, so meldete der rechte Sensor einen Weißwert. Daraufhin wurde der rechte Motor auf rückwärts gestellt und der linke auf vorwärts, so dass der Roboter nach rechts fuhr. Kam der Roboter nach rechts ab, meldete der linke Sensor einen Schwarzwert und die Motoren wurden entsprechend gestellt.



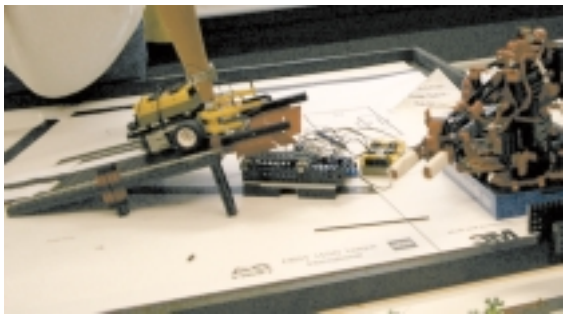
War der Roboter gegen eine Wand gefahren, so sollte er wenden und solange auf der Stelle drehen bis er die schwarze Linie wieder gefunden hatte. Bei Auslösung des Berührungssensors wurde der „Berührungszähler“ auf 1 gesetzt. Dies bewirkte, dass der Roboter erst eine halbe Sekunde zurück fuhr und sich dann mindestens 1,5 Sekunden drehte. Gleichzeitig wurden die „Steuerungs-befehle“, wie vorwärts fahren oder lenken, außer Kraft gesetzt, sodass die Drehung nicht unterbrochen werden konnte. Erst wenn der rechte Lichtsensor schwarz und der linke Sensor weiß meldete (sich also wieder auf dem richtigen Kurs befand), wurden die „Steuerungsbefehle“ wieder aktiviert, damit er wieder normal geradeaus fahren konnte. Prallte er aber bevor er die schwarze Linie wieder fand gegen eine andere Wand (in einer Ecke), so sollte der Berührungszähler auf 2 gestellt werden. Er sollte nun mindestens 1,5 Sekunden nach links drehen. Sobald er die schwarze Linie wieder gefunden hatte, sollte er wieder normal geradeaus fahren. (Anm.: Er musste sich deshalb min. 1,5 Sekunden drehen, da er sonst wieder gegen die Wand gefahren wäre. So konnte er erst wieder nach frühestens 1,5 Sekunden reagieren.)



Schematische Darstellung des Fahrverhaltens bei einer Kollision

Nun konnte er den Rest der schwarze Linie abfahren und somit alle Tonnen einsammeln. Am Ende seiner Route stand eine Rampe die er hinauffahren sollte. Oben angekommen sollte der Roboter stoppen. Dies erreichten wir, indem wir am

oberen Rand der Rampe auf die schwarze Linie ein weißes Stück Papier und links daneben auf den weißen Untergrund ein schwarzes Stück Papier klebten. Wenn der Roboter am oberen Rand angekommen war, so sahen seine Lichtsensoren statt links weiß und rechts schwarz das genaue Gegenteil. Daraufhin wurde der Befehl eingeleitet, der den Roboter ausschaltete.



links: Sucher-Robo auf der Rampe
mitte: Lieferband (unter der Rampe)
rechts: Fischertechnik-Roboter

Der Transport-Roboter

Die Erbauer

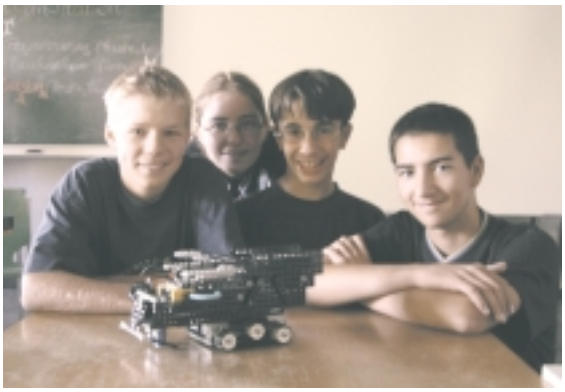


Bild 1: Die Transportrobotergruppe und ihr SaSoBeMo

Die Aufgabe des Roboters

Der Transport-Roboter aus Lego sollte kleine Tonnen von einem Fischertechnik-Roboter aufgeladen bekommen. Diese sollte er dann in einem Regal einlagern. Anschließend sollte er wieder zu seinem Ausgangspunkt zurückfahren, um weitere Male beladen zu werden.

Die Ausstattung des Roboters

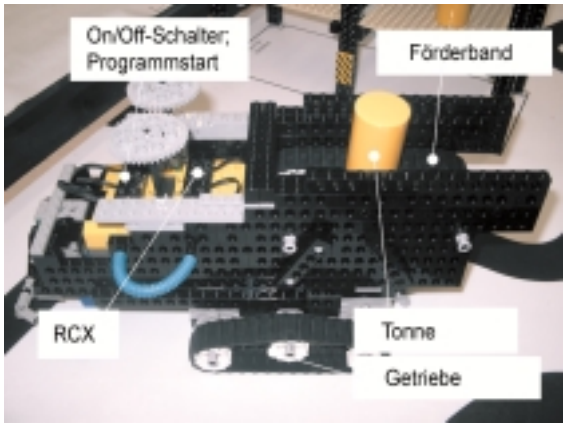


Bild 2: Der Transport-Roboter

Ausgestattet wurde der Roboter von uns mit zwei Lichtsensoren, einem Berührungssensor, einem Förderband, einem RCX, sowie fünf Motoren, zwei für die rechte Seite, zwei für die linke Seite und einem für das Förderband. Als Bereifung besaß der Roboter Gummiketten. Er wurde so gebaut, dass die Tonnen nicht herunterfallen und man schnell und einfach an den RCX herankommen konnte, um die Akkus auszuwechseln.

Der Bau des Transportroboters

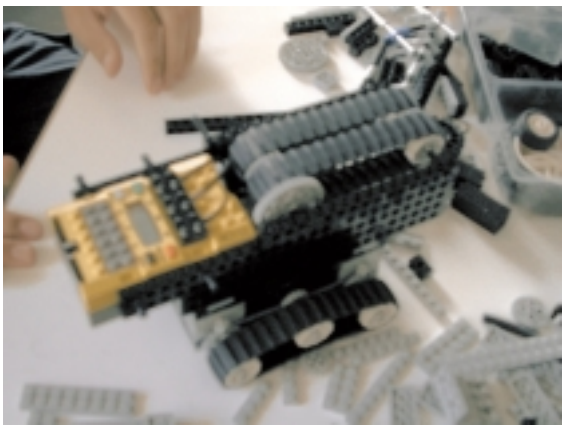


Bild 3: Bau des Roboters

Wichtig beim Bau des Roboters waren die Stabilität, das Ladevolumen und ein ruhiges Fahren. Nach vielen Anläufen schafften wir es, ein Modell zu konstruieren, das all die oben genannten Eigenschaften besitzt.

Als erstes widmeten wir uns dem Bau des Getriebes und versuchten eine möglichst gute Übersetzung zu erreichen, sodass sich der Roboter schneller bewegen konnte. Dazu verwendeten wir direkt an den Motoren größere Zahnräder und an den Achsen relativ kleine.

Als nun das Getriebe fertig war, wurde die Höhe des Förderbandes bestimmt. Bis zu dieser Höhe bauten wir dann die Seitenwände des Roboters, wobei wir an der Vorderseite den RCX zwischen den beiden Seitenwänden integrierten. Danach kam der Vorbau an die Reihe. Dazu verlängerten wir die Seitenwände unten ein wenig und bauten einen Unterboden ein, auf dem wir die beiden Lichtsensoren anbrachten. Daraufhin wurde der Vorbau in der Mitte etwas erhöht um den Berührungssensor mit seinem Bügel einzubauen.

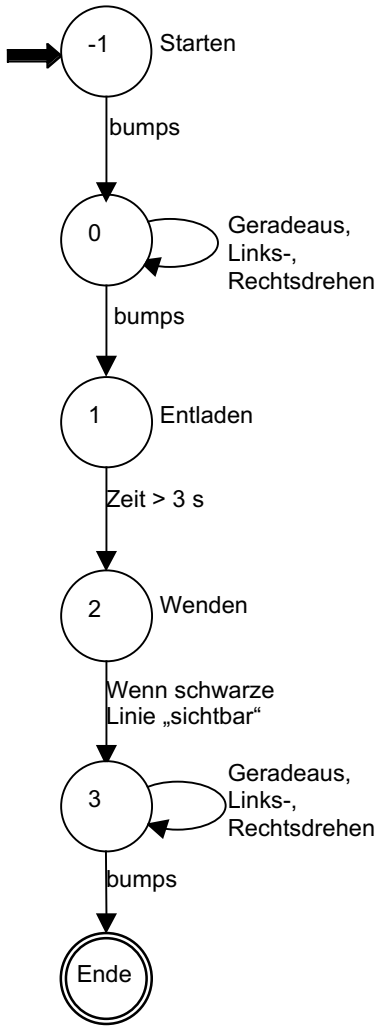
Die Programmierung des Roboters

Der Roboter wurde mit der Programmiersprache „Java“ programmiert. Als Editor diente uns dabei der so genannte JCreator. Mit ihm konnten wir das Programm schreiben und direkt per Infrarot auf den RCX übertragen.

Jede Bewegung, die der Roboter machen sollte, wurde in ein Verhalten geschrieben. Diese Verhaltens wurden dann alle einem Arbitrator (Schiedsrichter) übergeben, der dann entschied, welches der Verhalten nun die Kontrolle bekommen sollte. Wichtig war dabei die Anordnung. So hatte das letzte Verhalten die größte Wichtigkeit. Wann ein bestimmtes Verhalten die Kontrolle bekommen sollte, legten wir in der Methode „takeControl“ fest. Was der Roboter dann machen sollte, stand in der Methode „takeAction“. Außerdem verlangt jedes Verhalten noch die Methode „suppress“. In ihr wird festgelegt, wie der Roboter reagieren soll, wenn das Verhalten die Kontrolle verliert.

Natürlich war auch die Programmierung ein iterativer Prozess. Sehr viel Geduld musste investiert werden um die geschriebenen Programme zu testen und bestimmte Fehler herauszufinden.

Zustandsdiagramm des Roboters



Da wir für jedes Verhalten einen eigenen Zustand bzw. eine eigene Zustandsvariable (siehe Grafik oben) zuordneten, ließen wir den abhängig vom Zustand, zwei, drei, vier… mal piepsen. Somit konnten wir immer feststellen, ob der Roboter den nächsten Zustand erreicht hatte oder nicht und dann gegebenenfalls an der jeweiligen Stelle den Fehler suchen.

Im Zustand –1 drehte der Roboter um 180°
Im Zustand 0 sollte er an der schwarzen Linie entlang in Richtung Regal fahren, d.h. der Roboter musste geradeaus fahren, links und rechts drehen können sowie der linken Kante der schwarzen Linie folgen, bis der Berührungssensor einmal ausgelöst wurde und der Zustand um 1 höher gesetzt wurde.
Im Zustand 1 sollte er eine gewisse Zeit rückwärts fahren, entladen, ein wenig wieder vorwärts fahren und den Zustand auf 2 setzten.

Im Zustand 2 sollte er einfach 1 Sekunde drehen und zur Zustandsvariable 1 addieren, damit diese auf 3 stand.

Der Zustand 3 war in zwei „Spalten“ untergliedert: Solange bis der Berührungssensor nichts meldete, traten die Verhaltens „Geradeaus“, „Linksdrehen“ und „Rechtsdrehen“ mit vertauschtem Schwarz- und Weißwert in Kraft, wenn aber der Berührungssensor ausgelöst wurde leitete der Schiedsrichter, der in der Hauptklasse programmiert worden war, die Kontrolle an „Ende“ weiter. Nun sollte der Roboter anhalten und sich abschalten.

In der Klasse „Init“ legte man die Variablen und die Geschwindigkeiten der Motoren fest und aktivierte die Sensoren. Dies hätte man auch in die anderen Quellcodes schreiben können, aber diese Variante war erstens übersichtlicher, zweitens komfortabler zu bedienen, denn wenn man z. B. die Geschwindigkeit eines Motoren hätte ändern wollen, hätte man in allen Klassen die richtige Stelle suchen und jedes Mal die Werte ändern müssen.

In der eigentlichen Hauptklasse stand nur der Arbitrator, das Importieren der ganzen Verhalten und dem Aktivieren des ganzen.

Um die Programme auch ausführen zu können, mussten sie nach dem Schreiben als erstes kompiliert werden. Wenn dieser Vorgang abgeschlossen war, wurde das Programm auf den RCX gespielt und der Roboter führte das Programm aus.

Nachdem die Programmierung abgeschlossen war, haben wir unsere Programme im HTML-Format dokumentiert. Dazu mussten wir im Programm eintragen, wofür die einzelnen Klassen und Methoden standen oder was sie zu bedeuten hatten. Danach wurde auf Knopfdruck eine HTML-Dokumentation erzeugt, die man sich mit einem Internetbrowser ansehen kann.

Betriebsstörungen des Roboters

Auch wenn unser Roboter noch so stabil und gut programmiert war, traten dennoch manchmal Komplikationen auf. Diese häuften sich vor allem bei den Lichtsensoren und bei der Zeitsteuerung, da diese abhängig von der Spannung in den Batterien sind. Wenn nun die Spannung in den Akkus abnahm, meldeten die Lichtsensoren falsche Lichtwerte und der Roboter konnte seinen Weg nicht mehr finden. Genauso ist es auch bei der Zeitsteuerung. Wenn die Spannung abnahm, zählte die Steuerung nicht mehr gleichmäßig, sodass der Roboter nicht mehr die vorgegeben Wegstrecke fuhr. In unserem Fall führte das dazu, dass der Roboter zu weit vor dem Regal anhielt und somit die Tonnen nicht ins Regal einräumte, sondern davor ablud.

Chronologie

So, 24. August: Bau und Programmierung der Vorläufer namens „Wandpraller“ (mit einem Berührungssensor vorne). Er wendet, sobald er gegen eine Wand stößt und fährt dann weiter. Fernsehauftritt des „Wandprallers“ in Aktion und die Weiterentwicklung (schneller und mit Lichtsensoren ausgestattet). Erste Programmierarbeiten für diesen umgebauten Roboter.

Mo, 25. August: Quellcodefertigstellung des SaSoBeMo und Planung über einen neuen Roboter für die 2-Wochenaufgabe.

Di, 26. August: Bau des Roboters namens SaSoBeMo (Anfangsbuchstaben der Erbauer und Programmierer: Sascha, Sonja, Benjamin, Moritz). Konzepterstellung für die Tätigkeiten, die der Roboter nacheinander ausführen soll.

Mi, 27. August: Die Programmierung des Roboters ist bis auf das Förderband fertig. Durch viel Pech zerfällt der Roboter in X-Teile auseinander und so wurde er mit ganz neuem Aussehen konstruiert. Nur der Unterbau, die 4 Motoren und das Förderband übernommen wurden.

Do, 28. August: Endgültiges Programm für den Roboter und Stabilisierung des Grund und Aufbaugerüsts.

So, 31. August: Abschluss des Projekts SaSoBeMo mit der Dokumentation des Quellcodes. Der Bau der zwei Taster, die dazu da sind, dass der Fischer-technik-Roboter den Transporter einschalten kann, beendete den Bau und die Programmierarbeiten.

Blackbox

Die Erbauer

Bild 1: Die Erbauer bei ihrer Arbeit

Die Aufgabe des Roboters

Dieser Roboter agierte auf der Folie der Lego-League. Seine Aufgabe bestand darin, von einer festgelegten Ausgangsposition eine Zug-/Klapp-Brücke durch Anstoßen hinunterzuklappen, über diese hinüberzufahren und auf der anderen Seite eine Windmühle, ebenfalls durch Anstoßen, zu aktivieren. Danach sollte er umdrehen und über die Brücke wieder zum Ausgangspunkt zurückkehren.

Aufbau des Roboters:

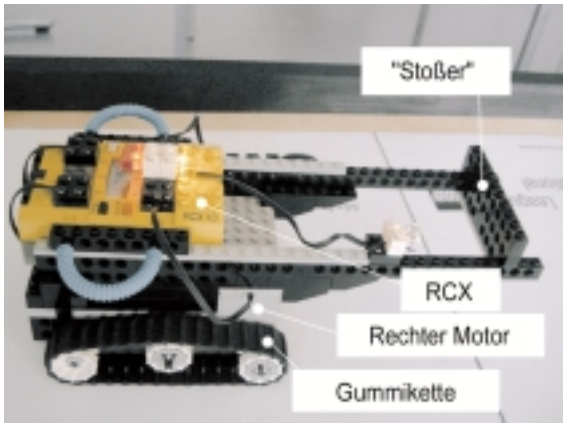


Bild 2: Dies ist die „Blackbox“ von der rechten Seite

Damit dieser Roboter die Brücke mit der hohen Steigung hinauffahren konnte, brachten wir an seinen Rädern Gummiketten an. Diese gewährten vor allem auf der Brücke einen besseren Halt.
Die Ketten wurden jeweils mit drei Laufrädern (auf dem Bild in weiß zu sehen) gespannt, wobei jeweils nur das Vorderrad über eine Untersetzung von einem Motor angetrieben wurde. Um eine getrennte Ansteuerung dieser Motoren zu ermöglichen, wurden sie an getrennte Ausgänge des RCX angeschlossen.

Um die Anzahl der Umdrehungen der Antriebs-achsen zu messen, wurde pro Antriebsseite ein Rotationssensor in den Antrieb der Untersetzung eingebaut.

Mit den vom Sensor gemessenen Werten wurde die gefahrene Wegstrecke berechnet.

Als „Werkzeug“ zum Herunterklappen der Brücke und zur „Aktivierung“ der Windmühle diente dem Roboter ein spezieller verlängerter Vorbau, der „Stoßer“.

Der Bau des Roboters:
Wichtig beim Bau des Roboters waren die Größe, die Stabilität, die Standfestigkeit, ein ruhiges Fahren und ein gutes Aussehen natürlich.

Als erstes haben wir uns dem Bau des Getriebes gewidmet und versucht, eine möglichst gute Untersetzung zu erreichen, sodass sich der Roboter ohne großen Kraftaufwand bewegen konnte. Dazu verwendeten wir direkt an den Motoren angebrachte kleine Zahnräder, die die größeren Zahnräder an den Achsen antrieben. Diese Zahnräder trieben wiederum noch größere Zahnräder an. Allerdings konnten wir diese nicht beliebig groß wählen, da sie sonst den Boden gestreift hätten.

Als nächstes bauten wir eine Haltevorrichtung für den RCX, der möglichst in der Mitte angebracht sein sollte, damit das Gewicht gleichmäßig auf den Roboter verteilt wurde. Später jedoch, als die Stoßvorrichtung hinzugefügt wurde, platzierten wir den RCX ein Stück weiter hinten, sodass er das Gewicht des „Stoßers“ ausglich.

Als Krönung der Ästhetik brachten wir an unserem Gefährt schließlich noch drei Lichter an, die aber steuerungstechnisch ohne Bedeutung waren.

Listing des Programms „BlackBox“:

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

/**
 * LEGO-Projekt mit leJOS: BlackBox.java<br>
 * In dieser Klasse startet der Roboter in der Basis. Er klappt über eine
 * Vorrichtung die Brücke herunter und fährt über diese zur
 * Windmühle, um diese zu aktivieren. Daraufhin kehrt er wieder über die
 * Brücke zur Basis in die Ausgangsposition zurück.
 * Die Klasse erbt alle Eigenschaften der Klasse RotationNavigator.<br>
 * Anmerkung: Die Winkelangaben entsprechen nicht denen in Wirklichkeit!
 * @version 1.0 vom 01.09.03
 * @author Benjamin, Moritz, Sascha
 */
public class BlackBox extends RotationNavigator {

    private static final float wheeldiameter = 3.2f;
    private static final float driveLength = 15.0f;
    private static final float ratio = 1/1;

     /**
      * Konstruktor: Alle drei Motoren werden auf Geschwindigkeit 7 gestellt.
      */
    public BlackBox() {
        super(wheeldiameter, driveLength, ratio);
        Motor.A.setPower(7);
        Motor.C.setPower(7);
        Motor.B.setPower(7);
        Motor.B.forward();
    }

    /**
    * Diese Methode tut die Arbeit: Der Roboter fährt 116cm vor und lässt
    * die Brücke herunter,dann fährt er 75cm zurück, dreht nach
    * rechts um 100°, fährt 22cm vor, dreht nach links um 96°, fährt 169cm
    * vorwärts über die Brücke und schaltet die Windmühle ein,
    * fährt 15cm zurück, dreht um 270° nach links, fährt 210cm
    * vorwärts über die Brücke, dreht um 100° nach rechts, fährt
    * 24cm vor und dreht zuletzt noch um 175° nach rechts.
    */
    public void doTheWork() {
            travel(116);
            travel(-75);
            rotate(100);
```

```
            travel(22);
            rotate(-96);
            travel(169);
            travel(-15);
            rotate(270);
            travel(210);
            rotate(100);
            travel(24);
            rotate(175);

            try {
                Button.RUN.waitForPressAndRelease();
            }
            catch (InterruptedException e) {}
    }

    /**
     * Hauptprogramm der Klasse
     */
    public static void main (String[] args) {
        BlackBox blacky = new BlackBox();
        blacky.doTheWork();
    }
}
```

Programmierung mit dem „Rotation-Navigator“
Der Roboter sollte sich unabhängig von seiner Umwelt bewegen können. Um dies zu ermöglichen nutzten wir die Möglichkeiten des Rotationssensors und programmierten die „Blackbox“ in Java mit dem „Rotation-Navigator“. Mit Hilfe dieser Klasse konnten wir den Roboter einen bestimmten, vorprogrammierten Weg abfahren lassen. Damit der Roboter wusste, wie weit er gefahren ist, zählten die auf beiden Seiten angebrachten Rotationssensoren, wie bereits erwähnt, die Anzahl der Motordrehungen. Aus diesen gemessenen Werten errechnete der RCX die zurückgelegte Wegstrecke. Dazu mussten wir den Umfang des Rades in Relation zum Umfang des Zahnrades am Motor setzten und diesen Wert zusammen mit dem Kettenabstand angeben. Anschließend wurde die Wegstrecke und die Winkel, um die sich der Roboter an bestimmten Stellen drehen sollte, in den Quellcode eingegeben.

Probleme mit dem „Rotation-Navigator“
Doch der Roboter tat natürlich nicht gleich von Anfang an das, was wir wollten. Dies lag, wie wir nach einiger Zeit herausfanden, daran, dass die Ketten im Gegensatz zu Rädern in Kurven rutschten, sodass der Roboter nicht mehr auf dem korrekten Kurs war. Da der „Rotation-Navigator“ leider nicht dafür ausgelegt war, mussten wir nun den Kurs des Roboters korrigieren. Dazu veränderten wir die Werte so lange, bis der Roboter eine einigermaßen vertretbare Bahn fuhr. Wir verbrachten sicherlich mehr Zeit mit dem Verändern der Werte als mit dem eigentlichen Programmieren.

Im Endeffekt kamen wir zu dem Schluss, dass sich mobile Roboter besser mit Sensoren steuern lassen, mit denen der Roboter auf seine Umwelt reagieren kann, da er so viel präzisier arbeitet.

Chronologie
So, 31. August:
Vorstellung der Aufgabe durch die Kursleiter.

Mo, 1. September:
Zusammenbau, Auseinanderbau und wieder Zusammenbau des Roboters, sowie gleichzeitiges Programmieren. Beides bereitete zu Beginn einige Schwierigkeiten.

Mi, 3. September:
Umbau der Blackbox, weil die Rotationssensoren schneller drehen sollten als die Räder.
Programmierung des Roboters ging bestens voran, ein Austesten der Ports und Kabel war lästig, aber notwendig, damit der Roboter vorwärts fährt und die Rotationssensoren nicht in die falsche Richtung zählten.

Do, 4. September:
Leichtere Modifizierungen hauptsächlich zur Verschönerung.

Fr, 5. September:
Völlige Fertigstellung der Blackbox, d. h. fertig mit dem Bau, der Programmierung, des Austestens und der Dokumentation des Quellcodes.

Der Musik-Roboter

Wie wir auf diese Idee kamen

Genau wie bei der „Blackbox“ war es auch bei diesem Musik-Roboter der Fall, dass ein paar Kursteilnehmer bei der großen Gemeinschafts-aufgabe mit gewissen Teilen schon etwas schneller fertig waren. Aus diesem Grund blieb für sie noch etwas Zeit, sich mit einer weiteren interessanten Aufgabe zu beschäftigen. Da wir auch musikalische Kursteilnehmer hatten, war ein Musik-Roboter eine naheliegende und lustige Idee.

Was kann ein Musik-Roboter?

Ein Musik-Roboter, der mit Lego gebaut wird, kann einstimmige Melodien, die ähnlich klingen wie ein Handyton, „singen“. Er besteht aus einem RCX, der selbst Töne erzeugen kann. Wir hatten am Ende einen Roboter so programmiert, dass er ein Lied auf Knopfdruck abspielen konnte. Wenn wir noch etwas mehr Zeit gehabt hätten, wäre es möglich gewesen, die Melodien auf andere Roboter zu übertragen, bzw. das Programm des Musik-Roboters in die Programme anderer Roboter zu integrieren. Auf diese Weise wären wir in der Lage gewesen, beispielsweise den Transportroboter beim Abladen der Tonnen „Alle meine Entchen“ spielen zu lassen

Die Programmierung eines Musik-Roboters

Um einen Musik-Roboter zu programmieren, gibt man in seinem Programm an, wie lange er welchen Ton mit welcher Frequenz spielen soll. So kann man ihm Ton für Ton alle möglichen Melodien einprogrammieren.

Vereinfachung der Programmierung

Damit wir nicht bei jedem Ton immer wieder Zeit und Frequenz ausrechnen mussten, erstellten wir eine Java-Klasse „Note“, in der für jeden Ton eine entsprechende Frequenz und für jeden Notenwert die entsprechende Zeit festgelegt ist. In der Haupt-klasse mussten wir jetzt nur noch angeben, welche Töne der Roboter spielen soll, und welchen Notenwert diese haben. Hier ein kleines Beispiel aus dem Java-Quelltext:

```
public class Musiker {
    public static Note[] lied =
       {new Note(Note.f2, 4),
        new Note(Note.d2, 4),
       };
    ...
}
```

Mit Hilfe der Angaben aus der Tabelle der anderen Klasse konnte der Computer nun die Frequenz und die Zeit erfahren und somit den richtigen Ton spielen.

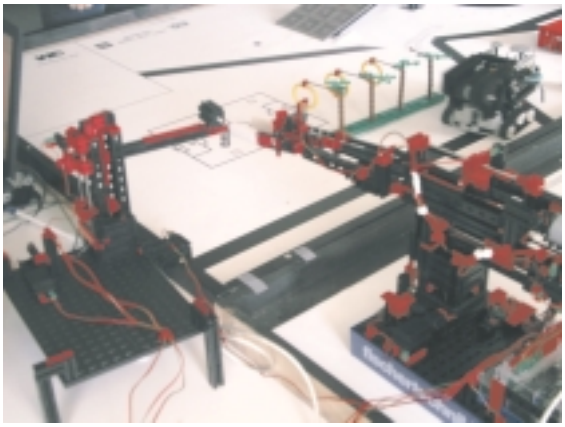
Ist der Musik-Roboter wirklich ein Roboter?

Auch wenn es auf den ersten Blick vielleicht nicht so aussieht, ist sogar der Musik-Roboter, obwohl er sich nicht bewegt und nur aus einem einzigen Klotz, dem RCX, besteht, wirklich ein Roboter. Er hat ein Programm eingespeichert bekommen, in dem genau das, was er ausführen soll, in diesem Fall die Töne, die er abspielt, festgelegt ist. Der Unterschied zu einem Industrieroboter ist im Grunde genommen sehr gering, wenn man nur das Programm betrachtet, da die Töne des Musik-Roboters dem

Anfahren verschiedener Positionen oder einem bestimmten Arbeitsvorgang entsprechen. Somit ist auch unser Musik-Roboter ein echter Roboter.

Der Fischertechnik-Roboter

Der Bau



Der Bau des Fischertechnik-Roboters unterscheidet sich wesentlich von dem der Lego-Roboter, da die zwei Bausysteme unterschiedliche Bauteile besitzen, die auf völlig verschiedene Weise verbunden werden müssen. Zum Beispiel werden die Fischertechnikteile zusammengeschoben oder geschraubt während die Legoteile mit kleinen Noppen zusammengesteckt werden.
Weitere Unterschiede ergaben sich schon bei der Aufgabenstellung, denn so musste unser Roboter nicht mobil sein (d.h. der Roboter ist stationär), was uns die Möglichkeit gab den Roboter um einiges größer zu bauen. Ein weiterer Vorteil der stationären Bauweise war die kontinuierliche Verbindung mit dem Computer, was uns die Wahl der Sensoren- (Bauteile, die elektrische Signale vom Roboter zum Computer liefern) und Motorenanzahl (Bauteile, die vom Computer gesteuert werden können, wie z. B. Motoren und Lampen) sehr vereinfachte. Im Gegensatz zu Lego musste unser Roboter nicht auf drei Sensoren und drei Motoren beschränkt werden, sondern konnte mit bis zu acht Motoren und sechzehn Sensoren an unser Interface (Bauteil, das die elektrischen Signale vom und zum Computer „übersetzt“ und weiterleitet) anschlossen werden.

Durch die Aufgabe, Tonnen von einem bestimmten Punkt zu einem anderen zu transportieren, ergab sich ein Problem mit dem genaueren Anfahren eines Punktes. Hierzu mussten wir die Anzahl der Umdrehungen eines bestimmten Motors zählen. Dies erreichten wir dadurch, dass wir an jeder Achse einen Taster befestigten, der bei jeder Umdrehung des Motors durch ein zahnradähnliches Bauteil 4-mal gedrückt wurde. Ein Taster ist ein Bauteil, dass bei Berührung einen Stromkreis unterbricht oder schließt.

Nun war es uns möglich einen Punkt exakt anzufahren. Obwohl das Fischertechnik-System durch das Zusammenschieben der Bauteile eigentlich stabiler als das Lego-System sein sollte, hatten wir zu Beginn mit der Stabilität zu kämpfen. Dieses Problem hatten wir allerdings schon im Nachfolgermodell behoben. Letztendlich haben wir einen Roboter erschaffen, der die Fähigkeiten besaß, Punkte in einem bestimmten Raum anzufahren und einen Gegendstand zu greifen, bzw. loszulassen.

Die Fischertechnik-Programmierung

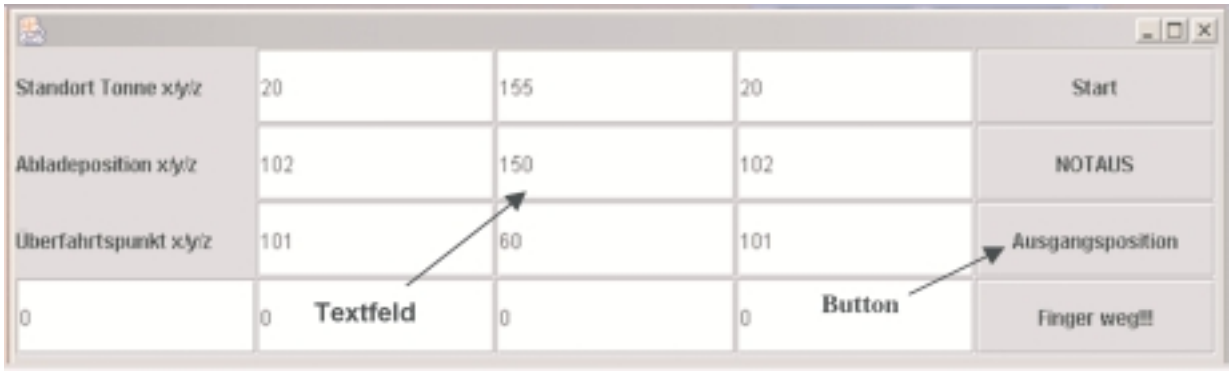
In welcher Sprache wir programmieren

Wir haben uns dazu entschieden in Java zu programmieren. Dazu wurden uns einige Zusätze zum „normalen“ Java zur Verfügung gestellt, die uns eine Verbindung vom Computer zum Roboter ermöglichten und die zur Steuerung eines Fischertechnik-Roboters nötig sind.

Was wollen wir Programmieren?

Als erstes bauten wir einen Roboter nach Anleitung. Unser allererstes Programm war relativ einfach. Nach dieser kleinen Vorbereitung begannen wir Schritt für Schritt auf unser großes Ziel, die Lösung der Gesamtaufgabe, hinzuarbeiten.

Unsere Aufgabe bestand grob gesagt darin, einen Gegenstand von einer Lego Mindstorms-Platte auf die andere Platte zu transportieren. Dazu programmierten wir zunächst eine Benutzeroberfläche mit Textfeldern und Buttons. Es gab Textfelder für den Ausgangspunkt des Gegenstands, für den sogenannten Überfahrtspunkt und für den Abladepunkt. Jeder Punkt im Raum besitzt drei Koordinaten (x, y und z), die jeweils für eine der Achsen stehen. In die Textfelder konnten wir also jeweils die Koordinaten eintragen, zu denen sich die Achsen bewegen sollten. Um zu überprüfen, ob die Achse sich während einer Bewegung schon am angegebenen (Anfahrts-, Ablade-, Überfahrts-) Punkt befindet, bauten wir Zähler ein (→ Bau Fischertechnik-Roboter). Merkt das Programm, dass eines der oberen Textfelder mit dem gezählten Stand übereinstimmt, stoppt es die Bewegung der Achse – sie befindet sich nun an der richtigen Position.



Auf die Benutzeroberfläche setzten wir ebenfalls vier Buttons, die mit bestimmten Funktionen ausgestattet waren:

Hier die Buttons (Schaltflächen):

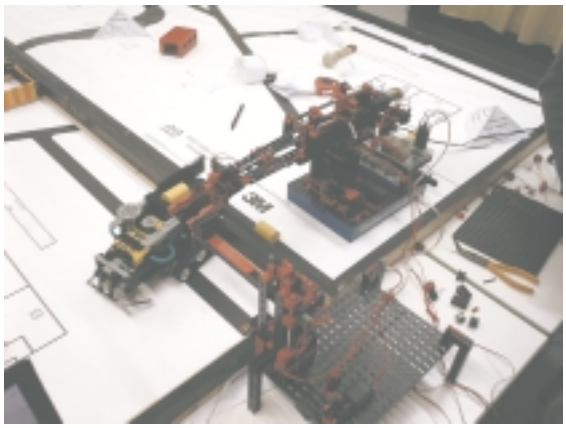
1. **Start:** Der Roboter startet seine Bewegung
2. **Notaus:** Der Roboter kann an der Position anhalten, an der er sich im Moment befindet
3. **Ausgangsposition:** Der Roboter kann sich selbstständig zu seiner Ausgangsposition bewegen
4. **Finger Weg:** Signallichter blinken, Musik wird abgespielt

Damit der Roboter sich bewegen konnte, mussten wir erreichen, dass sich die Achsen zu den angegebenen Koordinaten bewegen, wenn man die Buttons des User Panels betätigt. Dieses Problem lösten wir, indem wir die Buttons mit Events belegten, d.h. in ihre Klassen („Programm“ der Buttons) die Events mit den Befehlen zur Achsen-bewegung hineinschrieben.

Das haben wir realisiert, indem wir die Klasse Userpanel und die Klasse Steuerung programmiert haben. Das Userpanel erzeugt bei Knopfdruck ein Ereignis, Event genannt. Der Eventlistener, in der Steuerung überprüft, ob ein Button des Userpanels ein Event sendet. Wenn der Eventlistener ein Event empfängt, aktiviert er den Eventhandler, der dann gemäß seines Programms reagiert. Das heißt, wenn eine Button gedrückt wird, wird der Eventhandler aktiviert.

Beim Probeausführen des **Start**-Events bemerkten wir aber schnell, dass sich ein weiterer Button (z.B. **Notaus**) nicht betätigen ließ, solange das Event des Startknopfes nicht vollständig ausgeführt war. Um das gleichzeitige Drücken zweier Buttons zu ermöglichen, programmierten wir sogenannte Threads. Threads sind einzelne Ausführungspfade in einem Programm. Sie können im Hintergrund selbstständig parallel laufen. Es konnte also der Thread des Startbuttons laufen und der Thread des Notausbuttons konnte gleichzeitig gestartet werden. Mit den Threads war es nun auch möglich, dass sich die Achsen gleichzeitig bewegten.

Nachdem wir das Programm geschrieben hatten bauten wir einen neuen Roboter, der perfekt auf die Aufgabe zugeschnitten war. Das Programm mussten wir für diesen neuen Roboter fast nicht ändern, lediglich die Koordinaten des Ablade-, Auflade- und Überfahrtspunktes mussten korrigiert werden.



Die Fischertechnik-Roboter bei ihrer Aufgabe

Nun stellte sich aber noch die Frage, wie wir dem Roboter „mitteilen“ können, ob an der Auflade-position eine Tonne zum Aufladen vorhanden ist oder ob der Transportroboter zum Abladen der Tonnen schon bereit steht , also wann genau er seine Bewegung starten sollte. Dazu brachten wir an der Arbeitsplatte zwei Schalter an. Wenn die Lego-Roboter nun an ihren Positionen waren, betätigten sie die Schalter automatisch. Der Fischertechnikroboter erkannte, ob der Schalter gedrückt war und konnte dann seinen Auf- und Abladevorgang beginnen.

Exkursion

Kopfklinik

Bild 1: Im OP-Bereich der Kopfklinik

Nach der Begrüßung im DKFZ ging der Robotikkurs zuerst in die Kopfklinik der Universität Heidelberg. Dort hielt Werner Korb, einer der wissenschaftlichen Mitarbeiter der Kopfklinik, zunächst einen Vortrag über Medizinroboter im Allgemeinen und erzählte auch einiges über den Aufbau und die Funktionsweise des RobaCka, den wir kurze Zeit später selber sehen würden. Doch bevor wir zum Roboter gehen durften, mussten wir uns erst umziehen. Denn mit Straßenkleidung darf man den sterilen OP-Bereich nicht betreten. Es war für uns alle ziemlich ungewöhnlich, ganz in grün und mit Haube auf dem Kopf herumzulaufen. Doch mit der Zeit hatten wir uns an die seltsame „Uniform“ gewöhnt. Nach einer kleinen Unterweisung welche Bereiche wir nicht betreten durften, wurden wir endlich zum RobaCka geführt, der live natürlich noch viel eindrucksvoller war als auf den zuvor gesehenen Bildern.

Bild 2: Der RobaCka

Bau des RobaCka

Der Medizinroboter RobaCka wird vor allem für Fräsarbeiten am menschlichen Schädel eingesetzt, die er schnell und präzise erledigen kann. Er

besteht aus sechs in verschiedene Richtungen drehbaren Achsen, mit denen er sich optimal zu allen möglichen Positionen bewegen kann.

Damit eine Operation mit RobaCka ohne Zwischenfälle abläuft, ist er mit vielen Sicherheitsvorkehrungen ausgestattet: Es gibt z.B. eine Überwachungskamera, die den Roboter während der OP immer im „Blickfeld“ hat. Wenn sich ein Assistent versehentlich zwischen die Kamera und den Roboter stellt, schlägt das System Alarm, da der Roboter für die Kamera nicht mehr sichtbar ist und nicht mehr überwacht werden kann. Die Operation wird so lange unterbrochen, bis der Roboter wieder im Blickfeld der Kamera liegt.

Bei einer „Vor-Operation“ werden kleine Titanschräubchen in den menschlichen Schädel geschraubt. Diese markieren vier Eckpunkte des Fräsbereiches. Danach werden CT (Computer-Tomographie)-Aufnahmen des Schädels gemacht, auf denen auch die Schrauben zu sehen sind. Im Computer wird dann ein 3D-Modell des Schädels erstellt. Da sich die Schrauben auf dem Modell an exakt der gleichen Position befinden wie in Wirklichkeit, weiß der Roboter, wie er relativ zu den vier Schrauben operieren soll. Es entstehen also zwei Koordinatensysteme: eines im Computer-Modell und das andere am menschlichen Schädel. Ein drittes Koordinatensystem entsteht durch die Lage des Patienten während der realen Operation. Jetzt muss man dem Roboter nur noch zeigen, wo die Schrauben im realen OP liegen. Der Roboter fährt vor der eigentlichen Operation die Schrauben am Schädel ungefähr an. Der Arzt zieht den Manipulator (Roboterarm) danach auf die exakten Positionen der Schrauben, die der Roboter dann speichert. Nachdem er alle vier Schrauben registriert und gespeichert hat, weiß der Roboter genau, an welchen Stellen er fräsen muss. Zum Schluss werden die drei Koordinatensysteme „übereinandergelegt“.

Der Roboter hat ein weiteres System, das ihm sagt, wann ein einseitiger Druck gegen den Manipulator, der das Fräsen behindert und den Roboter in seiner Position verschieben könnte, zu hoch wird. Der untere Teil des Manipulators löst sich dann etwas und dreht sich automatisch zur Seite, um keine Schäden am Kopf anzurichten. (Dieser Vorgang ist ziemlich laut!) Falls der Roboter versehentlich zu tief in den Schädel eindringt, kann der Arzt den Manipulator manuell ein Stück nach oben ziehen.

Ein Computer- bzw. Roboterspezialist wie Hr. Korb ist außerdem immer mit im OP, da er im Falle anderer Komplikationen (computer-)technischer Art schnell und sicher helfen und eingreifen kann.

Steuerung des RobaCka

Der RobaCka wird mittels einer Benutzeroberfläche am PC gesteuert. Der Arzt trifft per Buttonklick immer die Auswahl, welcher Vorgang als nächstes ausgeführt werden soll. Doch bei der Bedienung der Benutzeroberfläche stellte sich zunächst ein Problem: Die Maus, mit dem der PC bedient wird, ist nicht steril. Wie löst man dieses Problem? Der Chirurg hat ein kleines steriles Gerät in seiner Hand,

auf dem farbige Knöpfe angebracht sind, deren Druck als Mausklick an den PC weitergeleitet wird. Dieses einfache sterile Gerät ersetzt also die nicht sterile Maus. Damit der Roboter überhaupt anfängt zu operieren, hält der Chirurg zwei bestimmte Knöpfe auf dem Gerät immer gedrückt (Tot-Mann-Schaltung).

Bild 3: Hr. Korb neben “seinem” Roboter

Lässt er einen oder sogar beide Knöpfe los, stoppt der Roboter seine Bewegung sofort und setzt erst wieder ein, wenn beide Knöpfe wieder ordnungsgemäß gedrückt sind. So kann der Chirurg auch bei kleineren auftretenden Fehlern die Operation sehr schnell abbrechen.

Chirurgische Klinik

Nach unserem Besuch in der Kopfklinik liefen wir mit unseren Kursleitern zur Chirurgischen Klinik der Universität Heidelberg, um dort den Roboter „da Vinci“ zu besichtigen. Heike Koos, eine Mitarbeiterin von „Intuitive Surgical“, der Herstellerfirma von „da Vinci“, erzählte uns viel Interessantes über den Roboter und konnte uns so einen weitreichenden Einblick in die Arbeit mit ihm geben.

Bild 4: Der manuell gesteuerte Manipulator des „da Vinci“

Einsatzgebiet des „da Vinci“

In der Chirurgie wird der „da Vinci“ oft für Operationen an inneren Organen im Bauchbereich eingesetzt. Einer seiner Vorteile ist z.B., dass man mit ihm minimal-invasive Operationen durchführen kann. Bei einer Operation in der Bauchgegend muss also nicht die ganze Bauchdecke aufgeschnitten werden, lediglich die kleinen Arme des Manipulators mit Zangen und einer Kamera an ihren Enden werden durch die Bauchdecke gebohrt. Nur diese „Einstiegslöcher“ müssen später chirurgisch versorgt (d.h. genäht) werden.

Der „da Vinci“

Der „da Vinci“ unterschied sich wesentlich vom zuvor gesehenen RobaCka, denn er gilt streng genommen nicht als Roboter, sondern ist ein Tele-Manipulator.

Er besteht aus dem operierenden Manipulator und einer separaten Bedienungseinheit, mit der er manuell gesteuert werden kann. Der Manipulator selbst besitzt drei Arme. An den Enden der äußeren Arme befinden sich kleine Zangen, am mittleren Arm eine Kamera.

Das Besondere an „da Vinci“: Es gibt kein fertiges Programm, das gestartet wird und mit dem er dann selbstständig arbeitet, sondern der operierende Chirurg steuert die Maschine fern. Dazu fährt er mit den Fingern in kleine Schlaufen an der Bedienungseinheit, welche die Zangen des „da Vinci“ darstellen. Die Bewegungen, die der Chirurg mittels der Schlaufen ausführt, werden von den Zangen am Manipulator umgesetzt. Auf diese Weise wird eine Operation mit „da Vinci“ realistisch ausgeführt. Der Chirurg sieht den Bereich, in dem operiert wird, durch die oben erwähnte, am mittleren Arm des Manipulators angebrachte Kamera mit variabler Vergrößerung. Dies ermöglicht eine sehr präzise Arbeit.

Bild 5: Steuerung des “da Vinci”

Durch die separate Steuerung ist es sogar möglich, den „da Vinci“ von einem anderen Kontinent aus zu bedienen und so „Fernoperationen“ durchzuführen. Auf diese Weise werden viele neue Möglichkeiten im Bereich Operationstechniken eröffnet.

Eigene Versuche mit „da Vinci“

Natürlich durften wir zum Abschluss unseres Besuches in der Chirurgischen Klinik auch selbst einmal mit dem „da Vinci“ arbeiten. Bei dem Versuch, mit den Zangen des Roboters Fäden durch kleinste Nadelöhre zu fädeln, wurde uns klar, warum man zur Arbeit mit dem „da Vinci“ erst

speziell ausgebildet werden muss: Man muss erst ein Gefühl dafür entwickeln, wie stark der Roboter auf Bewegungen reagiert. Auch die vergrößerte Ansicht der „Arbeitsplatte“ durch die Kamera war ziemlich gewöhnungsbedürftig. Die anderen Kursteilnehmer konnten über einen Bildschirm alle ausgeführten Bewegungen genau verfolgen und so zusätzlich gute Tipps geben.

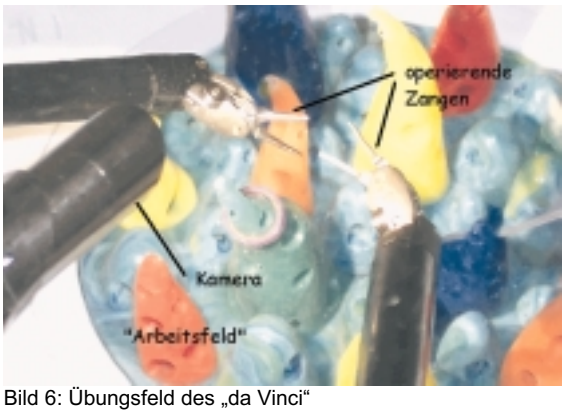


Bild 6: Übungsfeld des „da Vinci“

Der Besuch in den beiden Kliniken zur Besichtigung der Roboter war für uns alle etwas Besonderes, da man normalerweise nicht die Möglichkeit hat, diese OP-Bereiche und die Medizinroboter zu sehen.

Abschlusspräsentation

Die Abschlusspräsentation für die Eltern sollte einen Einblick in unsere Arbeit der zwei Wochen auf der Science Academy geben. Wir sollten unser Projekt in vier je 30-minütigen Schienen vorstellen, wobei wir uns darauf einigten, dass in jeder Zeitschiene drei Kursteilnehmer präsentierten.

Da wir mit unserer Arbeit an den Robotern jedoch noch nicht ganz fertig waren, kamen wir etwas in Zeitnot. Um die Präsentation noch rechtzeitig fertig stellen zu können, waren einige Überstunden von Nöten. In diesen erstellten wir mit Hilfe von PowerPoint mehrere Folien um unsere Kursarbeit darzustellen. Das Bild 1 zeigt eine Übersicht über unseren Präsentationsaufbau. Zuerst stellten wir unsere Roboter und unsere Arbeit vor, danach gab es eine Live-Vorstellung der Roboter. Zum Schluss hatte das Publikum die Möglichkeit, in einer Diskussionsrunde Fragen zu stellen.

Bild 1: Ablauf der Präsentation

Am Tag der Präsentation waren wir alle natürlich etwas aufgeregt und deshalb die Stimmung kurz vor dem Vortrag sehr angespannt. Der erste Teil der Präsentationen verlief in allen Schienen reibungslos. Doch oft traten kleinere Probleme bei den Live-Vorführungen der Roboter auf: Die Koordinierung der einzelnen Roboter funktionierte nicht ganz perfekt und wir mussten ein wenig nachhelfen. Doch trotzdem wurden unsere Präsentationen vom Publikum sehr gut aufgenommen und so mancher ließ sich richtig in den Bann der Roboter ziehen. Alles in allem waren die Präsentationen ein großer Erfolg.

Kurs 4:
Robotik – Wie Roboter ticken

Warum haben wir den Robotikkurs gewählt? Was hat uns besonders gefallen?
Wir, die Teilnehmer des Robotikkurses, haben es uns zum Ziel gemacht, die Robotik näher kennen zu lernen. Doch warum haben wir an dem Robotikkurs teilgenommen? Die meisten Teilnehmer wollten schon immer einmal programmieren oder hatten schon positive Erfahrungen damit. Außerdem fanden sie es viel interessanter, selbst Roboter zu bauen. Somit stellte der Robotikkurs eine perfekte Kombination zwischen Programmieren und Konstruieren dar.

Im Endeffekt waren wir mit dem Kurs alle sehr zufrieden. Wir haben viel gelernt, aber auch viel Spaß gehabt. Besonders gefallen hat uns die tolle Kursatmosphäre. Keiner von uns hat seine Entscheidung, den Robotikkurs gewählt zu haben, im Geringsten bereut. Wir würden jederzeit noch einmal an einer solchen Akademie teilnehmen. Im Folgenden wollen wir jeden Teilnehmer kurz vorstellen.


Der Robotikkurs mit den Kursleitern

Steckbriefe der Teilnehmer

Name:	**Sebastian Arlt**
Geburtsdatum:	30.12.1988
Wohnort:	Kürnbach nahe Karlsruhe
Schule:	Melanchthon-Gymnasium Bretten
Wir über ihn:	„spendabel (Kaugummis)“ - „der Fotoscheue“ – „guter Sinn für Humor“

Name:	**Moritz Binder**
Geburtsdatum:	18.01.1989
Wohnort:	Ettlingen bei Karlsruhe
Schule:	Eichendorff-Gymnasium Ettlingen
Wir über ihn:	„Nettigkeit in Person (noch eine Tasse Tee?)“ – „freundlich“ – „lustig“

Name:	**André Dau**
Geburtsdatum:	14.04.1989
Wohnort:	Stuttgart-Zuffenhausen

Schule: Ferdinand-Porsche-Gymnasium
Wir über ihn: „witzig“ - „stellt ironische Fragen“

Name: **Matthias Groß**
Geburtsdatum: 02.04.1988
Wohnort: Vöhringen bei Rottweil
Schule: Lina-Hähnle-Realschule Sulz
Wir über ihn: „Teslageneratorfan“ – „erklärt immer sehr ausführlich“ – „Programmierer“

Name: **Sonja Hammes**
Geburtsdatum: 17.11.1988
Wohnort: Hockenheim
Schule: Carl-Friedrich-Gauß-Gymnasium Hockenheim
Wir über sie: „Japanfan“ – „zielstrebig“

Name: **Frauke Jensen**
Geburtsdatum: 28.06.1989
Wohnort: Vaihingen (Enz) bei Stuttgart
Schule: Stromberg-Gymnasium Vaihingen
Wir über sie: „nett“ – „hilfsbereit“

Name: **Sascha Kocher**
Geburtsdatum: 17.10.1987
Wohnort: Lauchringen im Süden BWs
Schule: Klettgau-Gymnasium Tiengen
Wir über ihn: „sportlich“ – „aktiv“ – „Koordinator“ – „selbstbewusst“

Name: **Gregor Lux**
Geburtsdatum: 08.03.1988
Wohnort: Vellberg bei Schwäbisch Hall
Schule: Peutinger Gymnasium Ellwangen
Wir über ihn: „praktisch veranlagt“ – „spontan“

Name: **Kerstin Pöhl**
Geburtsdatum: 04.11.1987
Wohnort: Sachsenheim bei Stuttgart
Schule: Ellental-Gymnasium Bietigheim-Bissingen
Wir über sie: „engagiert“ – „sportlich interessiert“ – „unternehmungsfreudig“

Name: **Michael Rehermann**
Geburtsdatum: 13.01.1989
Wohnort: Deizisau
Schule: Gymnasium Plochingen
Wir über ihn: „nett“ – „guter Gesprächspartner“ – „Teamgeist“

Name: **Armin Richter**
Geburtsdatum: 25.02.1989
Wohnort: Radolfzell am Bodensee
Schule: Realschule Radolfzell
Wir über ihn: „witzig“ – „Konstrukteur“ – „großer Tüftler“

Name: **Benjamin Wallisch**
Geburtsdatum: 24.11.1988
Wohnort: Mössingen
Schule: Quenstedt-Gymnasium Mössingen
Wir über ihn: „Denker“ - „konzentriert“ – „zuverlässig“ - „Organisator“

Name:	**Helge Peters**
Geburtsdatum:	28.06.1975
Arbeitsstelle:	Institut für Prozessrechentechnik, Automation und Robotik, Universität Karlsruhe
Wir über ihn:	„Fischertechnikchef“ - „charmantes Lächeln“

Name:	**Matthias Taulien**
Geburtsdatum:	17.06.1952
Schule:	Hector-Seminar Mannheim
Wir über ihn:	„Javameister“

Name:	**Georg Wilke**
Geburtsdatum:	04.11.1968
Arbeitsstelle:	Hector-Seminar Heidelberg
Wir über ihn:	„Kompetenz in Person“ – „LEGO-Konstrukteur“