Neuenhagen, den 12.09.14

Liebe Team-Mitglieder der Einstein-Constructors, liebe Interessierte,

lange habe ich darüber nachgedacht, was ich in diesem Teil online stelle. Ich habe mich dann dafür entschieden, erst einmal die grundlegenden Fertigkeiten für das Ansprechen der Motoren und das arbeiten mit ihnen zu trainieren. Das lief allerdings nicht immer nach Plan, weshalb dies schon Version Nummer 3 ist. Und damit ihr wisst, was euch genau erwartet, kommt als erstes die Gliederung:

1. Ein neues Programm beginnen
2. Vorbereiten des zweiten Programms
3. Wir programmieren unser zweites Programm

## 1. Ein neues Programm beginnen

In diesem Teil werden wir, wie angesprochen, ein Programm schreiben, bei dem wir den Roboter ein Stück vorwärtsfahren lassen. Dafür benötigen wir einen neuen Ordner. Legt diesen in eurem Mindstorms®-Java-Übungsordner an. Ihr könnt ihn „02-Motoren_1“ nennen. So habe ich ihn zumindest benannt. Dann könnt ihr nämlich die anderen Ordner nach diesem Schema benennen (z.B. „09-Schleifen_2“ o.ä.).
Jetzt startet ihr bitte Eclipse. Nun könnt ihr euren Workspace im erscheinenden Fenster auf diesen Ordner umstellen.

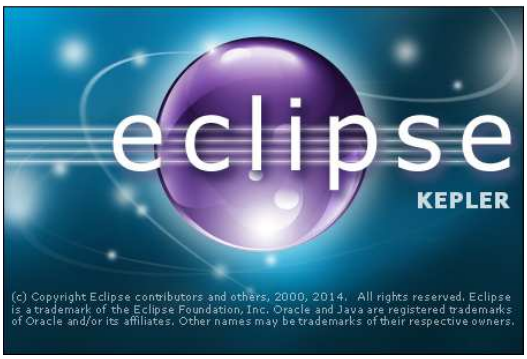



Abbildung 1: Eclipse startet

Auf dem Startbildschirm, der sich jetzt hoffentlich geöffnet hat, geht ihr jetzt oben „File“ und erstellt ein neues leJOS-Projekt. Falls ihr nicht mehr wisst, wie das geht könnt ihr entweder jetzt schon aufgeben, oder ihr schaut in Teil 3 meines Tutorials nach (http://www.appis.net/sites/lucas.html). Die weiteren Schritte:

1. Als Namen tragen wir den Ordner Namen ohne die Nummer ein (also „Motoren_1):

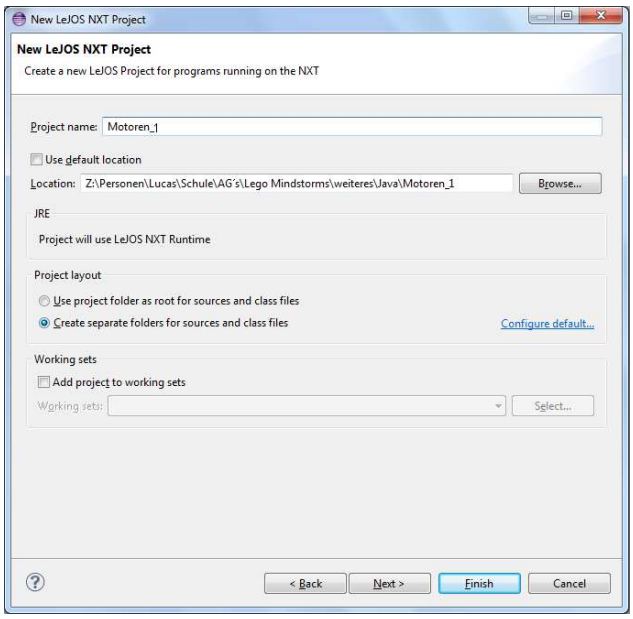



Abbildung 2: Das leJOS-Projekt „Motoren_1“ erstellen

2. Den Rest belasst ihr so. Ihr könnt jetzt auf „Finish“ klicken.
3. Nun wechselt ihr zur „Workbench“. Ihr müsstet jetzt die vom letzten Teil bekannte Oberfläche sehen:

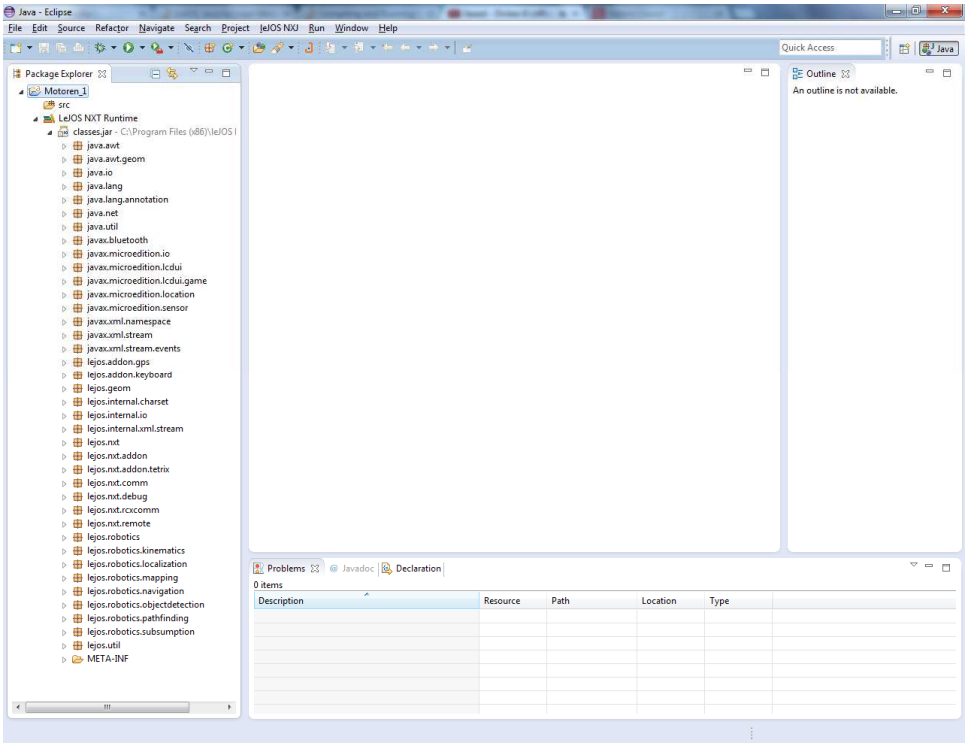



Abbildung 3: Die Eclipse-Workbench von „Motoren_1"

4. Ich denke mal, dass ihr inzwischen mitbekommen habt, dass wir eigentlich nur alle Schritte aus dem 3. und 5. Teil wiederholen. Logischerweise ist der nächste Schritt eine Klasse „motoren_1_*x*“ zu erstellen. *x* erersetzen wir durch ein Schlagwort, das beschreibt, was das Programm tun soll. Z.B. könnte man statt x „forward“ (vorwärts) einersetzen. Die Klasse erstellen wir in einem „package“ (Packet). Diese dienen dazu, verschiedene Klassen zusammenzufassen. Das Ziel des Programms wäre es bei folgendem Namen: „Motoren_1_forward“, vorwärts zu fahren. → Wir kreieren ein Packet „motoren“ im Projekt „Motoren_1“. Nun erstellen wir die Klasse „Motoren_1_forward“:

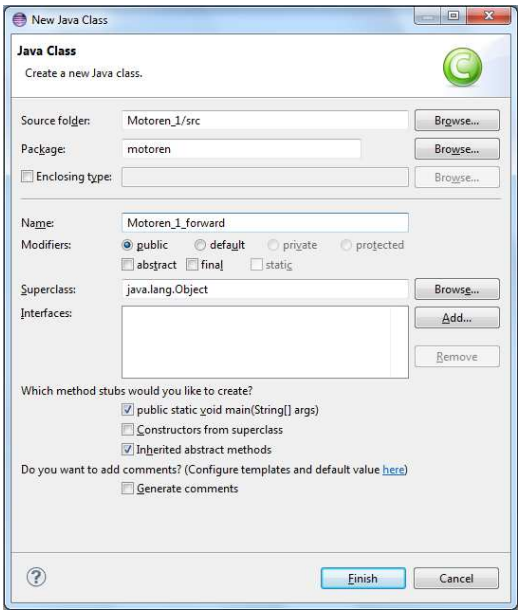



Abbildung 4: Erstellen der Klasse „Motoren_1_forward“

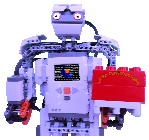
Denkt bitte beim Erstellen der Klasse daran, dass ihr ein Häkchen hinter „public static void main“ setzt. Nur dann wird daraus unsere Hauptklasse. Des Weiteren muss der erste Buchstabe des Klassennamens großgeschrieben werden. Nun könnt ihr ein weiteres Mal auf „Finish“ klicken.

## 2. Vorbereiten des zweiten Programms

Auch beim planerischen Vorbereiten des 2. Programms gehen wir ähnlich dem in unserem „Hallo Welt“-Programm vor.

Das heißt, dass wir unseren PAP-Designer starten (sofern vorhanden).

1. Strg + n für ein neues Projekt.
2. Als Namen tragen wir „Motoren_1“ ein.
3. Den PAP betiteln wir mit dem Namen der erstellten Klasse, also „Motoren_1_forward.java“.
4. Strg + s zum Speichern. Wählt wieder euren Ordner „02-Motoren_1“ aus. Entweder ihr erstellt einen Unterordner „Vorbereitung“ oder ihr speichert den PAP woanders.
5. Jetzt setzen wir den wichtigen Kommentar: den Autorenkommentar. Alle wichtigen Schritte, und wie man mit dem PAP-Designer arbeitet, steht in Teil 5 meines Tutorials: http://www.appis.net/media/files/Java_fuer_den_NXT_Teil_3.pdf
6. Für den Kommentar (der aus Autorname, Datum und Zweck des Programms besteht) benötigen wir einen groben Plan, was das Programm bewirken soll. Beim letzten Mal ging es darum, die Grundlagen der Programmierung des NXT’s mit Java kennen zu lernen. Heute wollen wir das ganze ausbauen, indem wir das erste Mal mit einem Motor arbeiten. Damit wir etwas sehen, lassen wir den Roboter einfach ein Stückchen vorfahren, also ist das Ziel / der Zweck des Programms: Bewegungen des Roboters mit Java.
7. Um herauszufinden, welche Möglichkeiten man hat, einen NXT vorwärtsfahren zu lassen, starten wir jetzt NXT-G.



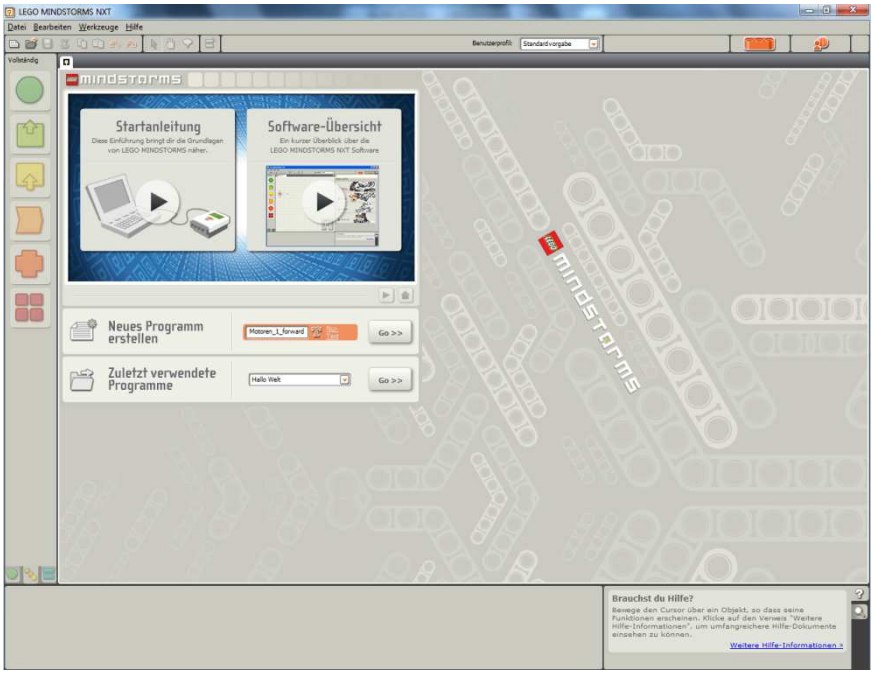



Abbildung 5: Die Start-Oberfläche von NXT-G

8. Wir erstellen ein neues Programm und nennen es „Motoren_1_forward“. Nach dem „Go > >“-Klick gleich mit Strg + s in dem entsprechenden Ordner speichern.
9. Auch hier im kleinen NXT-Logo am Anfang des Programms den Autoren-, Datums- und Zweckkommentar ersetzen.
10. Nun können wir uns aus der Allgemein-Palette den Bewegungs-Block herausziehen.


Abbildung 6: Inhalte der „Allgemein“-Palette, zu sehen: der Bewegungsblock

11. Uns interessiert an diesem Block vor allem welche Möglichkeiten wir haben, um die Dauer der Bewegung zu regulieren. Diese Information finden wir  am unteren Bildschirmrand. Dort gibt es eine Auswahlbox zu den Parametern der Dauer:

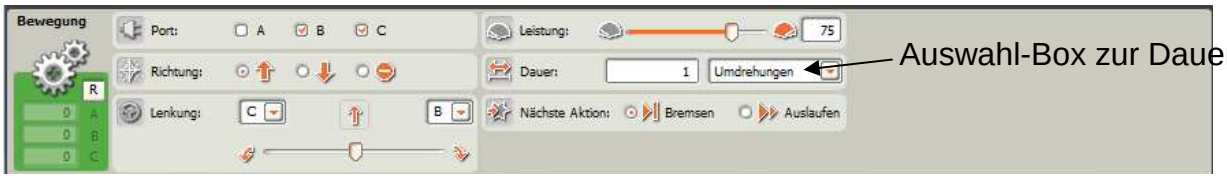



Abbildung 7: Informationsfeld zum Bewegungsblock

12. Das Feld zeigt uns an, dass wir die Dauer über Umdrehungen, Grad der Drehung oder als Zeit eingeben können. Eine weitere Möglichkeit ist das unbegrenzte Drehen der Motoren.
13. Am besten ist es, wenn wir 2 Einstellungen mal ausprobieren, und davor immer auf dem NXT-Bildschirm anzeigen lassen, nach welchem Parameter es geht. Ich schlage vor, dass wir uns nur mit der Begrenzung über Sekunden und Gradzahl beschäftigen. Eine Umdrehung ist logischerweise eine Drehung des Motors um 360°, deshalb können wir uns die Umdrehung als Parameter sparen, da wir einfach nur die Anzahl an Umdrehungen in Grad umrechnen müssten.
14. Zuerst erstellen wir jetzt den PAP, anschließend übertragen wir den Plan in die Lego®-Mindstorms®-Sprache und zum Schluss programmieren wir den NXT mit Java:

- Öffnet wieder den PAP-Designer
- Zeiht vom rechten Rand ein Ausgabe-Feld zwischen den Start- und den Ende-Block
- Als Inhalt tragen wir ein: „Ausgabe von : „Gradzahl“ “
- Danach setzen wir einen Vorgang-Block darunter.

- In diesen tragen wir ein, wie sich der Roboter nun bewegen soll, also geben wir als Inhalt „Bewegung vorwärts (720 Grad)“ an. 720° sind 2 Umdrehungen.
- Jetzt folgt erneut ein Ausgabe-Block, diesmal mit dem Inhalt: „Ausgabe von: „Sekunden“ “.
- Die Ausführung erreichen wir durch einen Vorgang-Block mit dem Inhalt: „Bewegung vorwärts (4 Sekunden)“.

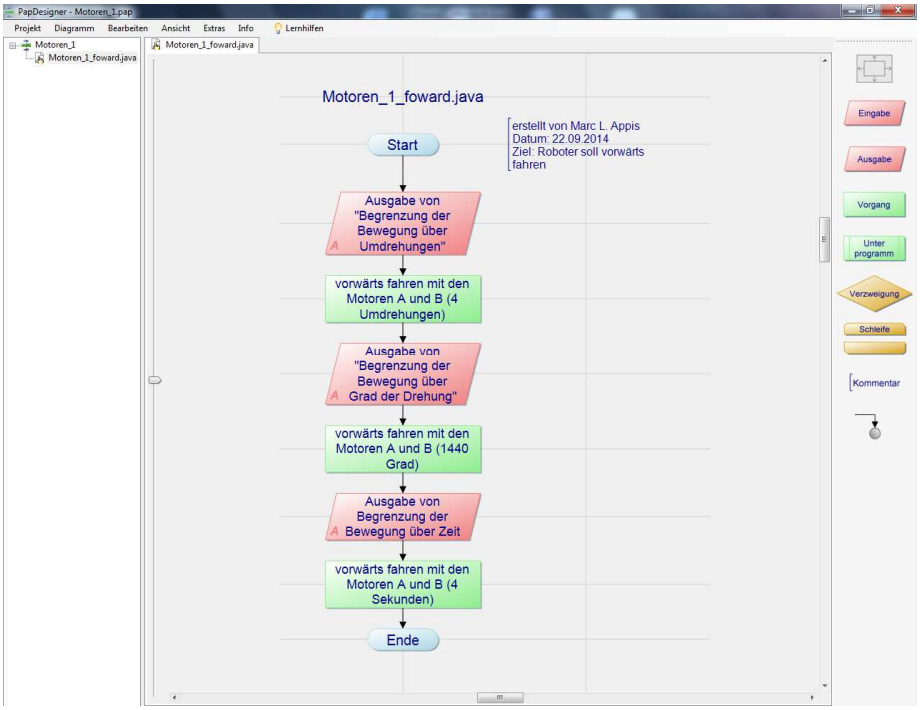



Abbildung 7: Der fertige PAP von „Motoren_1_forward.java“

Damit hätten wir den Plan für das Programm erstellt, und können nun eine programmtechnische Realisierung mit NXT-G beginnen.

- Wir löschen den Bewegungsblock mit der Entfernen-Taste.
- Wir ziehen aus der Aktionen-Palette den Anzeige-Block.
- Wir ändern die Einstellungen wie folgt:

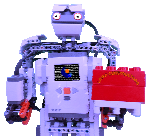

Abbildung 8: Einstellungen des Anzeige-Blocks

- Jetzt kommt der Bewegungs-Block ins Spiel. Wir ziehen ihn hinter den Anzeige-Block und stellen die Motorenleistung auf 100. Die Combo-Box hinter Dauer stellen wir auf „Gradzahl“ und in das Feld davor tragen wir 720 ein, wie es im PAP steht.


Abbildung 9: Einstellungen des Bewegungs-Blocks

- Die letzten beiden Schritte wiederholen wir einfach. D.h., dass wir jetzt als Anzeigetext „Sekunden“ eintragen und im Bewegungs-Block als Dauer Sekunden einstellen, sowie in das Edit-Feld 4 eintragen.

Das Programm sollte nun also so aussehen:



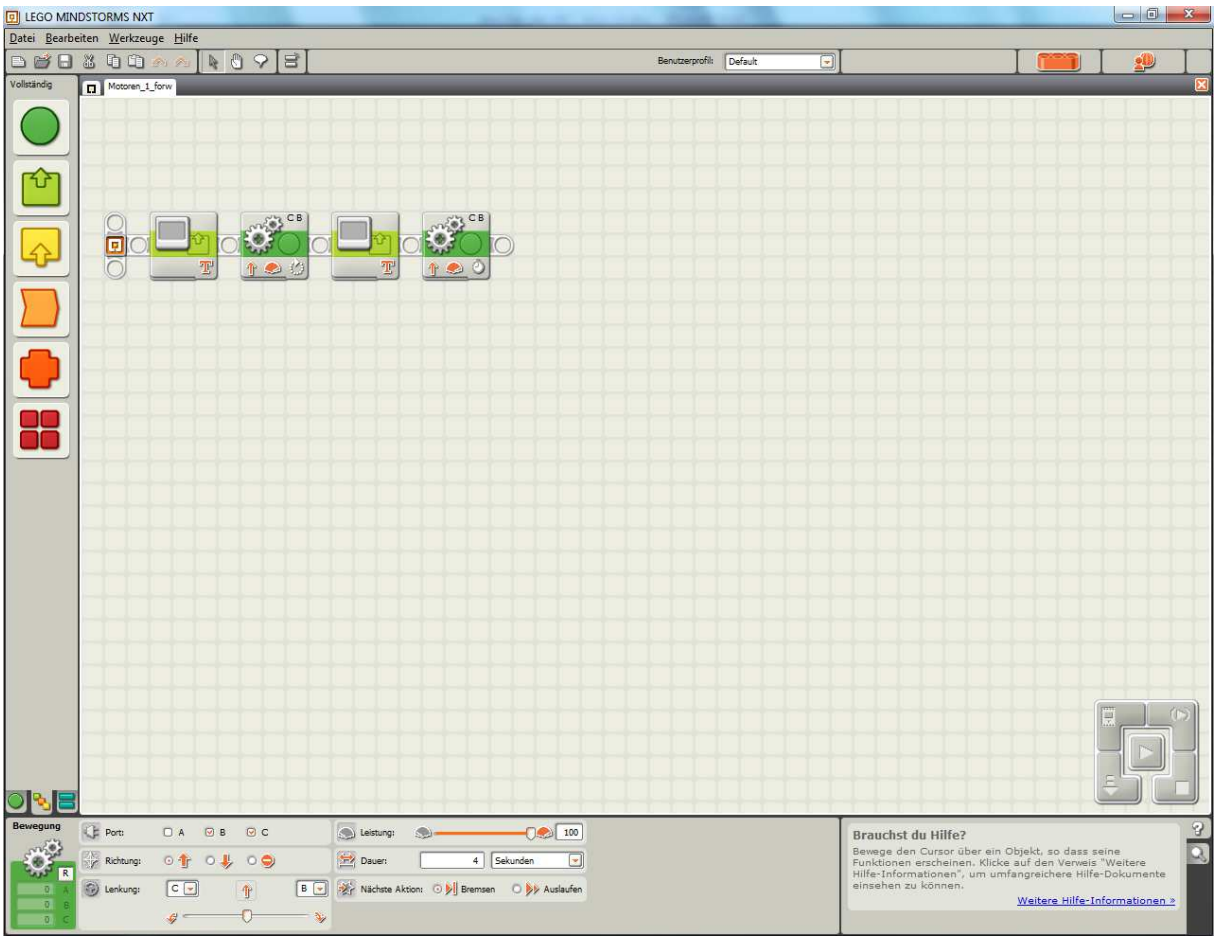



Abbildung 10: Das fertige Program

Jetzt können alle, die einen NXT zuhause haben, das Programm hochladen und ausprobieren.

Ich hoffe mal, dass es geklappt hat, denn das hat es bei mir.

## 3. Wir programmieren unser zweites Programm

Jetzt haben wir folgendes erreicht: Wir wissen, was der Roboter machen soll, wir haben einen genauen Plan, wie das Programm ablaufen soll und wir wissen, dass der Roboter in der Lage ist, das Programm auszuführen. Damit haben wir alle Vorrausetzungen erfüllt, damit wir beginnen können, das Programm in Java zu schreiben.

➔ Wir öffnen Eclipse

Da alle Pakete, Projekte und Klassen, die wir benötigen schon erstellt worden sind, können wir gleich damit beginnen, am Anfang der Datei „Motoren_1_forward.java“ den Autorenkommentar zu setzten. Wer (wie ich) faul ist kopiert diesen einfach aus dem PAP heraus.

Unter `package motoren;` setzten wir jetzt die (in jedem NXT-Programm mit Java wichtige) Import-Anweisung für das LeJOS-NXT-Paket. Darin sind alle wichtigen Funktionen und Klassen zu Steuern des Programms enthalten. Für alle weiteren Funktionen, die ihr verwendet und nicht von `import lejos.nxt.*;` abgedeckt werden, werden automatisch Import-Anweisungen geschrieben. Allerdings fehlt noch eine Anweisung (welche ich später noch erklären werde), die ihr bitte hinter `public static void main(String[] args)` setzt:

`throws Exception .`

Jetzt sind wir an der Stelle angelangt, an welcher wir mit dem Hauptteil des Programms beginnen. Der PAP fordert eine Text-Ausgabe. Wer im letzten Teil aufgepasst hat, weiß, dass man dieses Ziel mit `System.out.println();` erreichen kann. In die Klammern am Ende tragen wir den auszugebenden Text ein. In Java wird Ausgabe-Text immer in Anführungszeichen angegeben. → In die Klammern tragen wir (in Anführungszeichen) ein: „Gradzahl“.

Jetzt steuern wir die Motoren an. Der Projekt-Explorer zeigt uns an, dass es im lejos.nxt - Verzeichnis eine Klasse regulatedMotor gibt. Diese hat eine Methode „`setSpeed()`“. Das hört sich in meinen Ohren sehr gut an. Motoren sprechen wir über folgendes Schema an: `Motor.Portnummer.Methode`.

Auf unser Problem wenden wir die Methode wie folgt an: `Motor.C.setSpeed()`, um Motor C eine Geschwindigkeit zuzuweisen. Wenn wir jetzt noch herausfinden wollen, wie die Maximale Geschwindigkeit des Motors ist, setzten wir in die Klammern eine sogenannte get-Methode. Diese liefern immer einen Wert zurück, in unserem Fall die Maximale Geschwindigkeit. In die Klammern setzen wir also die Methode `getMaxSpeed()` nach dem Schema `Motor.Portnummer.Methode` ein. Die Zeile sollte also lauten:

```
Motor.C.setSpeed(Motor.C.getMaxSpeed());
```

In der nächsten Zeile setzen wir einfach eine Kopie der obigen ein, ersetzten aber C durch B.

```
Motor.B.setSpeed(Motor.B.getMaxSpeed());
```

Nun haben wir die Geschwindigkeit eingestellt und können jetzt die Dauer einstellen.

Die Dauer wird als Parameter der Funktion `rotate()` in Klammern gesetzt. Rotate bedeutet in etwa drehe(bis). Wenn wir also wollen, dass sich die Motoren um 720° drehen, dann muss die Programmzeile `Motor.C.rotate(720);` lauten.

[Hinweis: Wer die Motoren so eingebaut hat, dass der Roboter rückwärtsfährt, der setzt einfach ein – vor die 720]

Als nächstes müssen wir die Motoren synchronisieren, da das Programm sonst, wie normalerweise, eine Anweisung nach der anderen ausführen würde, also zuerst die Drehung des Motors B

und dann die Drehung des Motors C. Dem kann man vorbeugen, indem man hinter die Gradzahl in `Motor.C.rotate(720);` schreibt: ‚TRUE. True ist Englisch (wen wundert's ;-) ) und bedutet WAHR.

Die veränderten Zeilen lauten also wie folgt:

```
Motor.C.rotate(720, true);
Motor.B.rotate(720, true);
```

Damit hätten wir den ersten Teil. Nämlich die Steureung der Bewegung über die Gradzahl abgeschlossen und wir können uns der Umsetzung von Teil 2 widmen. Der Programmablaufplan fordert an dieser Stelle eine Ausgabe auf dem Bildschirm des Textes „Sekunden". Wir schreiben also erneut eine Zeile `System.out.println("Sekunden");`.

Das Ziel, das wir erreichen wollen ist, dass die Motoren nach 2 Sekunden aufhören, sich zu drehen. Um dies zu schaffen müssen wir einen sog. Thread (engl. für Faden) einsetzten. Dieser sorgt dafür, dass das Programm mehrere Aufgaben parallel bewältigen kann, in unserem Fall das Drehen der Motoren und das Mitzählen der Zeit, damit nach 2 Sekunden die Bewegung gestoppt werden kann.

Dafür müssen wir aber erst die Klasse Thread importieren, welche sich im Verzeichnis Java.lang befindet, damit der NXT überhaupt weiß, dass was er bei einem Thread tun soll. Wir ergänzen also unter `import lejos.nxt.*;` die Zeile `import java.lang.Thread;` . Nun können wir wieder in die Zeile unter der Ausgabe-Zeile wechseln und die Motoren vorwärts fahren lassen, in dem wir die Motoren einfach auf `forward()` einstellen: `Motor.C.forward();`. Wenn ihr wie oben das

Problem habt, dass euer Roboter nun rückwärtsfährt, so könnt ihr stattdessen auch backward() einsetzen. Als letztes müssen wir noch den Thread einstellen, sodass er die Aktion nach 2 Sekunden unterbricht. Wir schreiben also Thread. Automatisch seht ihr jetzt eine Liste von möglichen Befehlen. Da unser Thread nur die Aufgabe hat 2 Sekunden zu „schlafen“ und anschließend die aktuellen Vorgänge unterbrechen soll, ist es naheliegend die Methode `Thread.sleep(long milliseconds)` auszuwählen. 2 Sekunden sind umgerechnet 2000 Millisekunden, und das setzten wir in die Klammern der Methode.

Damit hätten wir das Programm soweit eigentlich abgeschlossen. Jetzt können wir uns noch der Behebung zweier kleiner Schönheitsfehler widmen. Wenn ihr das Programm jetzt ausführt werden die Aktionen so direkt hintereinander ausgeführt, dass ihr sie gar nicht unterscheiden könnt. Das können wir beheben, indem wir das Programm nach der ersten Vorwärtsbewegung einen Schlaf-Thread ausführen lassen. Am geeignetsten wären 3 Sekunden, also 3000 Millisekunden.

Des Weiteren wäre es günstig, wenn wir das Display löschen, nachdem wir „Gradzahl“ ausgegeben haben. Dazu sprechen wir das LCD-Display an und geben den Befehl, es zu säubern (engl.: clear).

Nun sind wir am Ende angelangt, das Programm ist fertig und läuft (zumindest bei mir). Ergänzen möchte ich noch Folgendes: Wenn ihr ein Programm mit NXT-G schreibt, und es dann testen wollt so müsst ihr natürlich die Lego-Software wieder auf den NXT überspielen. Dafür geht ihr auf der Menüleiste auf „Werkzeuge“ und anschließned auf „Firmware aktualisieren“:


Ich hoffe, dass euch auch der 6. Teil der Serie gefallen hat, und dass ihr etwas mitgenommen habt.

Genau so hoffe ich, dass ihr euch auf meinen nächsten Teil freut und mir bei Fragen und Anmerkungen euch bitte weiterhin an marc.appis@gmx.de wendet.

Bis dann!

```
/*
 *erstellt von: xxx
 *Datum: 01.10.2014
 *Ziel: Bewegungen des Roboters mit Java programmieren
 */

package motoren;

import lejos.nxt.*;
import java.lang.Thread;

public class Motoren_forward {

	public static void main(String[] args) throws Exception{
		// TODO Auto-generated method stub

		System.out.println("Gradzahl"); //Ausgabe von "Gradzahl"
		Motor.C.setSpeed(Motor.C.getMaxSpeed());

		//setzten der Geschwindigkeit auf die Höchstgeschwindigkeit

		Motor.B.setSpeed(Motor.B.getMaxSpeed());
		Motor.C.rotate(720, true);

		//mit einem Minus: Drehung in die entgegengesetzte Richtung

		Motor.B.rotate(720, true);

		Thread.sleep(3000);
		LCD.clear();

		System.out.println("Sekunden"); //Ausgabe von "Sekunden"
		Motor.C.forward(); //alternativ: backward() für eine Rückwärtsbewegung
		Motor.B.forward();
		Thread.sleep(2000);
	}

}
```

Der Quelltext von „Motoren_forward.java“