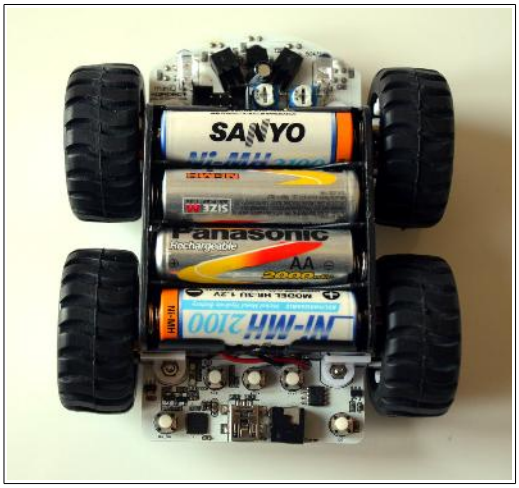
Programmierung von MiniQ-Arduino-Robotern mit der nxc2arduino-Bibliothek

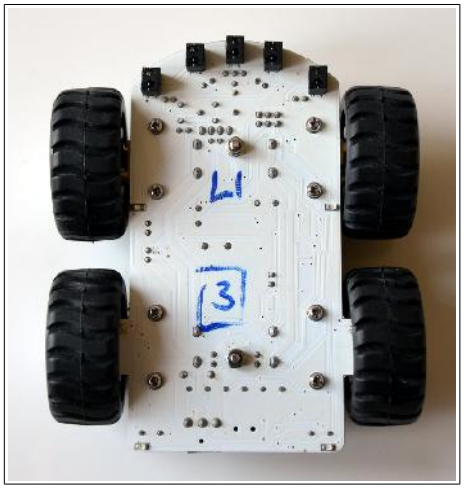
Zur Hardware

Die MiniQ-Roboter sind fertig montierte Roboter, für die man zur Stromversorgung vier AA-Akkus braucht. Sie haben

- vier (bzw. zwei) Elektromotoren zur Fortbewegung,
- zwei eingebaute Rad-Encoder für die Messung der gemachten Radumdrehung,
- fünf Infrarot-Sensoren für die Linienverfolgung,
- zwei Infrarot-LEDs und einen Infrarotempfänger für die Hindernisermittlung bzw. zur Programmierung einer Infrarotfernsteuerung
- drei programmierbare Taster und
- einen Summer.



Oberseite

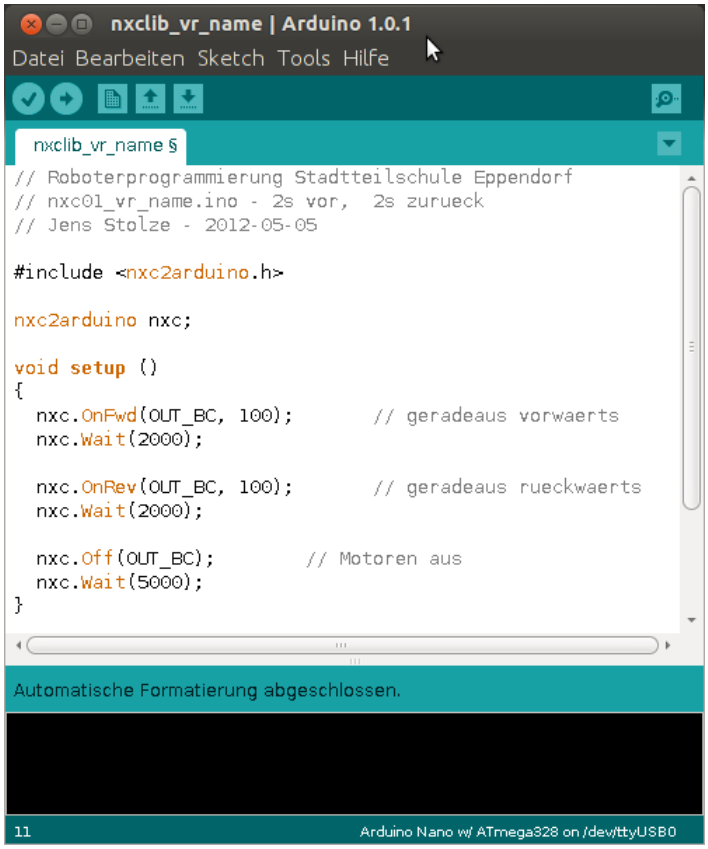


Unterseite

Sie werden mit dem weit verbreiteten Atmega 328P Mikrocontroller gesteuert und sind voll kompatibel mit den Arduinobausteinen.

Zur Software

Programmiert werden die MiniQ-Roboter mit der Arduinosoftware in einer Untermenge der Programmiersprache C mit einigen Erweiterungen für die Ansteuerung der Hardware. Die Arduino-IDE sieht so aus:



Mithilfe der nxc2arduino-Bibliothek wird die Programmierung weiter erleichtert, da einige Befehle von NXC, der C-ähnlichen Programmiersprache für NXT-Roboter, auf die Arduino-Roboter übertragen worden sind. Eine Beschreibung der Befehle findet sich auf den Seiten uv bis xy.

Zur Programmierung

Drei Stufen durchläuft ein Arduino-Roboter-Programm. Es muss

- zuerst **erstellt**,
- dann **übersetzt** (Fachbegriff: kompiliert) und
- zuletzt vom PC auf den Arduino-Roboter **übertragen**

werden. Zur Übertragung wird ein **USB-Kabel** benötigt.

Datum: ***Name:***

Online-Betreuungangebot

Online-Hilfen befinden sich unter

http://gesamtschule-eppendorf.de/material/informatik/arduino/arduino-roboter/

Hier findet man Informationen zu den folgenden Bereichen:

- Unterricht
- Software
- Hardware

Zum Programmierlehrgang

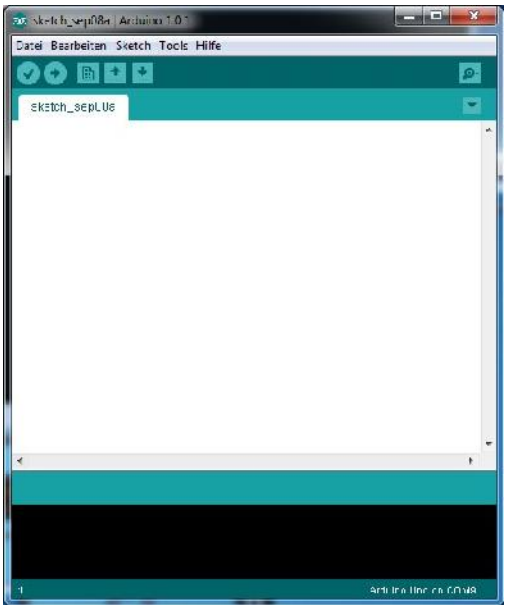
Um den Umgang mit der Arduino-IDE kennenzulernen und die Funktionskette Arduino-Roboter ↔ USB-Kabel ↔ PC zu überprüfen, ist als **erste Aufgabe** das Beispielprogramm nxclib_vr_name.ino aus den Beispielen zu laden, zu übersetzen und auf den Arduino-Roboter zu übertragen.

Was zu tun ist, ist auf den folgenden Seiten als Text- und als Bilderanleitung beschrieben. Außerdem findet sich diese Informationen auch auf den auf der vorigen Seite beschriebenen Online-Hilfe-Seiten.

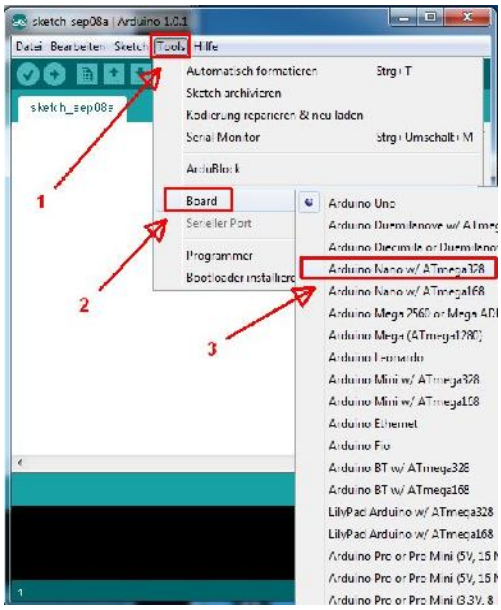
Kurzanleitung in Textform

- Arduinosoftware starten
- Arduino-Roboter anschließen (hier am Beispiel des MiniQ-4WD)
- unter "Tools - Board" "Arduino Nano/Atmega328" auswählen
- unter "Tools - Serieller Port" den korrekten Port auswählen (z.B. COM9 unter Windows und /dev/ttyUSB0 unter Linux)
- Wenn die nxc2arduino-Bibliothek im Programmverzeichnis installiert ist, dann ist das Testprogramm für die Bibliothek aus "Datei - Beispiele - nxc2arduino - nxclib_vr_name" zu laden und zu übersetzen (Strg-r).
- Nun muss der MiniQ-Roboter angehoben werden, damit er nicht gleich losfährt, wenn nach dem Hochladen des Programms (Strg-r) das Programm automatisch gestartet wird. Zum Anschalten muss der Knopf an der Hinterseite links mit der Bezeichnung "PW_ON" gedrückt werden. Außerdem müssen dann die blauen LEDs an der Unterseite des Roboters leuchten, sonstigenfalls sind die eingelegten Akkus durch neue auszutauschen.
- Danach ist das Programm als *`n2a_01_vr_name`* abzuspeichern.

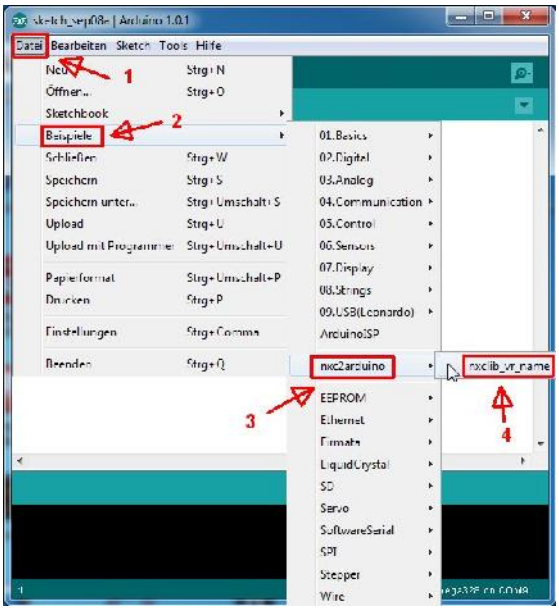
Langfassung als Bilderanleitung



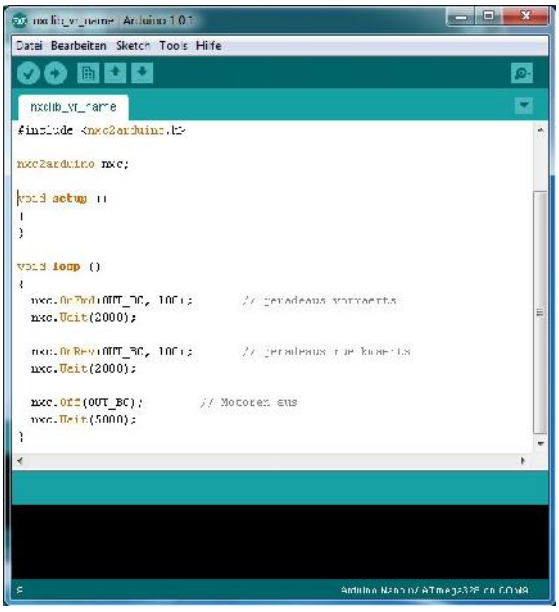
01 - So meldet sich das Arduino-Programm nach dem Start.



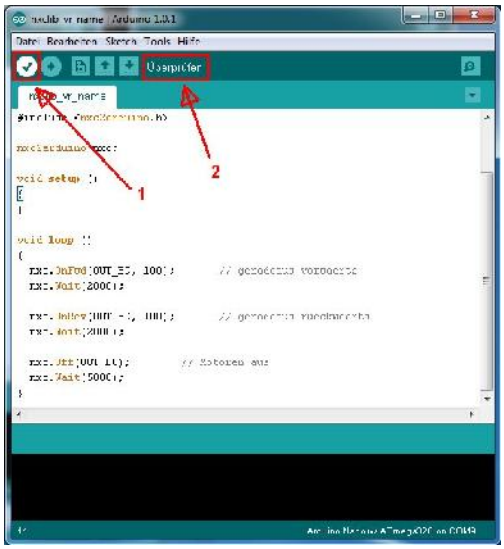
02 – Für die MiniQ-Arduino-Roboter muss „Arduino Nano w/ATmega328“ als das richtige Board ausgewählt werden (siehe 1 bis 3).



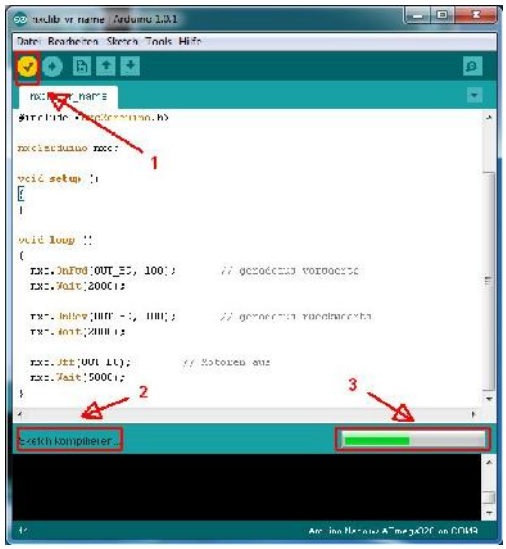
03 - Mit den Schritten 1 bis 4 lädt man das Beispielprogramm nxclin_vr_name" aus der nxc2arduino-Bibliothek.



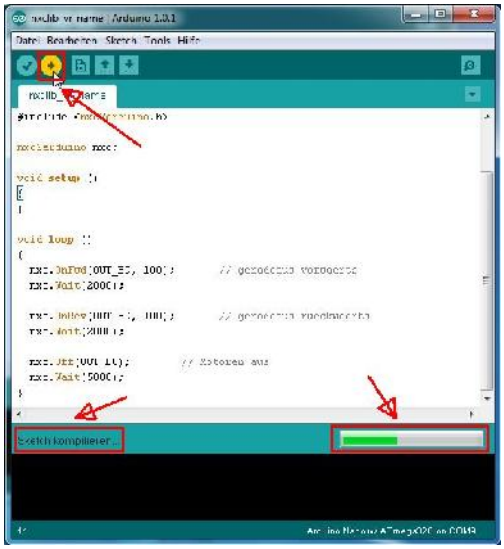
04 - Nun ist das Programm geladen.



05 - Übersetzen kann man es dann mit einem Mausklick auf den Haken in der Werkzeugleiste.



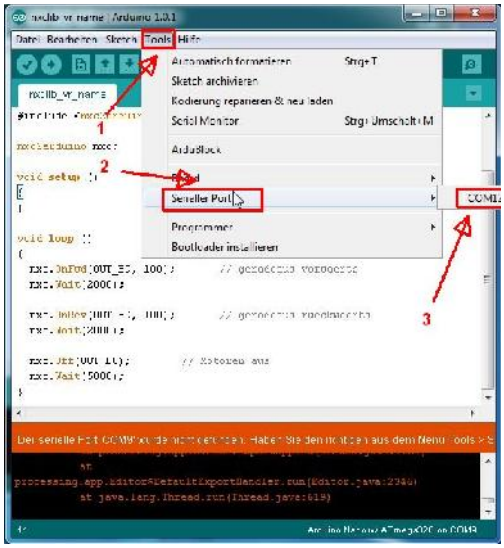
06 - Der Haken wird dann orange 1) und unten in der Statuszeile sieht man, wie weit das Programm übersetzt (Fachbegriff "kompiliert") ist.



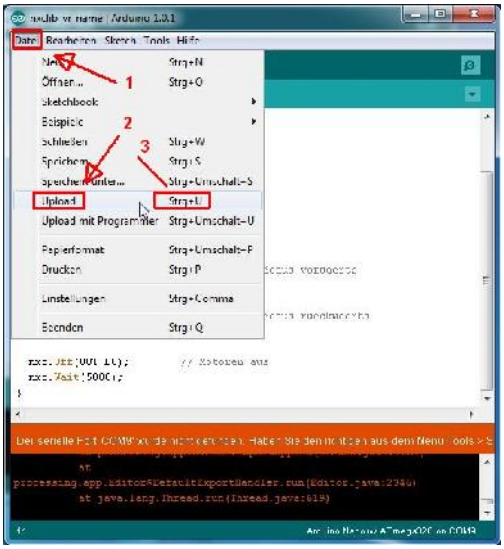
07 - Das Programm wird mit einem Mausklick auf den Rechtspfeil in der Werkzeugleiste hochgeladen 1). Der Fortschritt ist in der Statusleiste zu sehen (siehe 2 + 3).

08 - FEHLERMELDUNG! Wenn man den Arduinobaustein nicht eingeschlossen hat und/oder der falsche COM-Port voreingestellt ist, kommt es zu dieser Fehlermeldung in der Statuszeile.

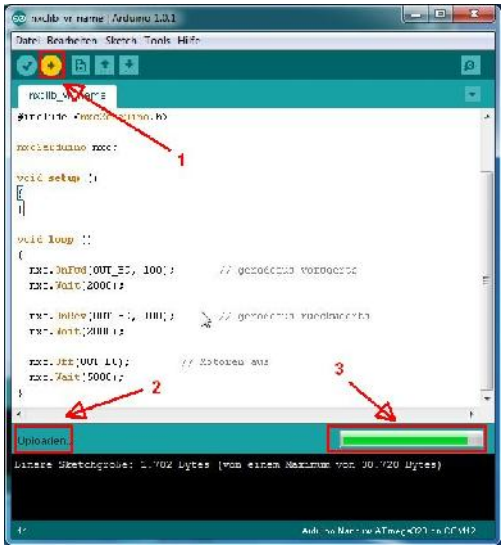
Datum: ***Name:***



09 - Über "Tools - Serieller Port" kann man dann den passenden Port wählen (hier steht jetzt COM12 zur Wahl). Das geht nur, wenn der Arduino-Roboter einmal angeschlossen und angeschaltet ist. VORSICHT - FESTHALTEN! Wenn ein Programm auf dem Roboter ist, dann kann er beim Anschalten wegfahren!

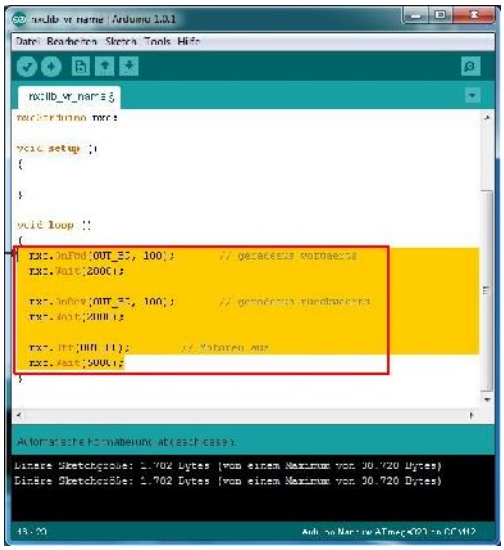


10 - Nun kann man wieder einen Upload starten. Dies geht auch über das Menü über "Datei - Upload" oder mit der Tastenkombination "Strg+u" (siehe 1 - 3).

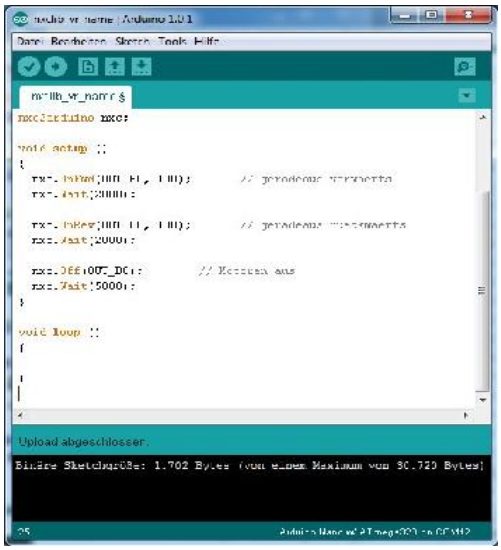


11 - Nun klappt der Upload problemlos... ;-)

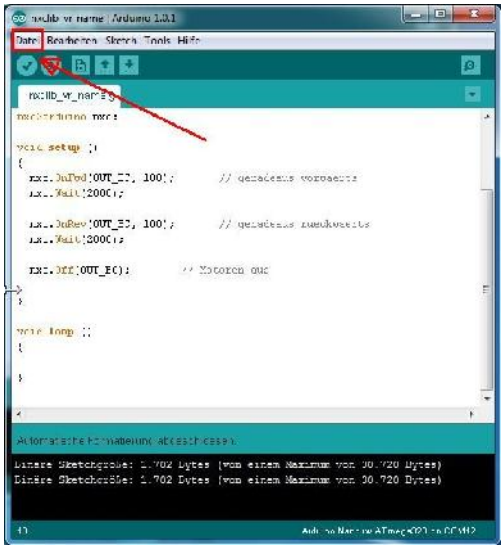
12 - Der Upload ist abgeschlossen (siehe Statusleiste unten).



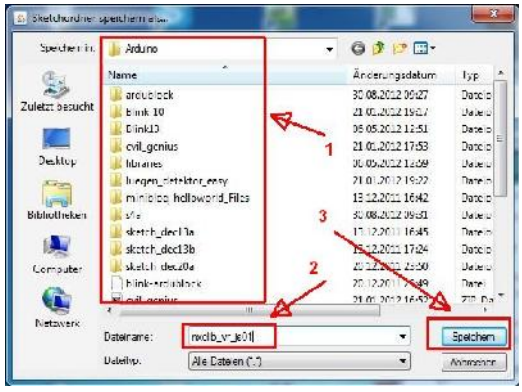
13 - Wenn ein Programm nur EINMAL ausgeführt werden soll, dann muss es zwischen den Klammern unter "void setup()" stehen. Dazu markiert man den Text mit gedrückter linker Maustaste und kopiert ihn mit "Strg-c" in die Zwischenablage. - In neueren Bibliotheksversionen ist dies schon durchgeführt und du kannst zu dem übernächsten Schritt gehen!



14 - Nachdem man die Schreibmarke an die gewünschte Stelle gesetzt hat, kopiert man den Text aus der Zwischenablage mit "Strg-v" rein. Voila!



15 - Nun speichert man den Text über "Datei - Speichern unter" ab.



16 - Man wählt den gewünschten Pfad 1) und den Dateinamen 2) und klickt auf den Speichern-Knopf. Anstatt "js" wählst du deine Initialien oder alle deiner Gruppe.

Datum: ***Name:*** 

Weitere Aufgaben:

2.) Unser Arduino-Roboter soll nacheinander 2 Sekunden vorwärts fahren, **2 Sekunden warten** und wieder 2 Sekunden zurückfahren. Korrigiere das vorhandene Programm und speichere dein Arbeitsergebnis unter *n2a_02_v2r_name* ab.

3.) Dein Arduino-Roboter soll 1s vorwärts fahren, sich um **90° drehen** und wieder 1s vorwärts fahren. Speichere dieses Arbeitsergebnis unter n2a_03_90_*name*.

4.) Nun soll er ein **Quadrat fahren**. Benutze dazu eine Schleife. Infos dazu findest du in der Programmhilfe. Speichere dieses Arbeitsergebnis unter n2a-04-quad-*name*.

5.) Dein Arduino-Roboter soll 3 **mal ein Quadrat fahren**. Benutze dazu die While-Schleife (siehe Vorlagenfenster). Speichere dieses Arbeitsergebnis unter n2a_05_3x4_*name*.

6.) Nun soll er eine **Spirale fahren**. Entwickle zuerst eine Vorgehensweise auf Papier und diskutiere sie mit deinen Nachbarn, bevor du dein Programm schreibst. Speichere das Arbeitsergebnis unter n2a_06_spiral_*name*.

7.) Dein Arduino-Roboter soll den **Lichtsensor abfragen** und

- solange geradeaus fahren, bis er auf eine schwarze Linie trifft,
- dann stoppen und
- einen Ton von sich geben.

Speichere das Arbeitsergebnis unter n2a_07_stop_*name*.

8.) Nun erweiteren wir das letzte Programm. Der Roboter soll

- zunächst nach vorne fährt,
- bei einer schwarzen Linie aber immer
  - stoppen,
  - 1 Sekunde rückwärts fährt,
  - sich zufallsgesteuert nach links oder rechts drehen und
  - dann wieder nach vorne fahren und
- wieder diese Prozedur von vorne beginnen (Dauerschleife)

Speichere das Arbeitsergebnis unter n2a_08_laut_*name*.

9.) Lasse deinen Arduino-Roboter mindestens **10 Sekunden** lang **tanzen.** Hier darfst du deiner Kreativität freien Lauf lassen... ;-). Speichere das Arbeitsergebnis unter n2a_09_tanz_*name*.

MERKE: Leider können die **Dateinamen** bei den Arduinos keine „-“-Zeichen vertragen, dafür gibt es keine Längenbegrenzung wie beim NXT!

nxc2arduino-Anweisungen – Teil 1

`// Roboterprogrammierung - GSE`
`// n2a_01_vr_name.ino - 2s vor, 2s zurueck`
`// Vorname Nachname - 2012-05-05`
 	
`#include <nxc2arduino.h>`
`nxc2arduino nxc;`
 	
`/* - 2s vorwaerts, 2s zurueck - */`
 	
`void setup()`
`{`
`  ...Anweisungen...`
`}`
 	
`void loop()`
`{`
`  ...Anweisungen...`
`}`
	Grundgerüst für nxc2arduino-Programme:
	– Hier stehen in Kommentarzeilen zuerst der Programmname und was das Programm macht
	– … und hier dein Vor- und Nachname sowie das aktuelle Datum.
	
	– Eine spezielle Bibliothek muss immer eingebunden, damit wir einige NXC-Befehle nutzen können.
	
	– Dieser Kommentar beschreibt, was das Programm macht.
	
	Im setup-Bereich kommen Programmzeilen, die nur **einmal** ausgeführt werden sollen.
	
	Im loop-Bereich kommen Programmzeilen, die **wiederholt** ausgeführt werden sollen.

`nxc.OnFwd(OUT_BC, 100);`	Die Motoren B und C laufen mit 100/255 Kraft vorwärts
`nxc.OnRev(OUT_C, 100);`	Der Motor C läuft mit 100/255 Kraft rückwärts
`nxc.Wait(200);`	Wartet im Programm 200/1000tel Sekunde
`nxc.Off(OUT_BC);`	Schaltet die Motoren B und C aus
`long startzeit =` `nxc.CurrentTicks();`	Weist der langen Variablen „startzeit“ die Zeit in Millisekunden zu, die seit Programmstart vergangen sind
`#define FAHREN 50`	Definiert die Konstante FAHREN auf den Wert 50
`int licht;`	Definiert eine Variable `licht`
`int licht = 0;`	Legt eine Variable mit dem Wert 0 fest
`licht = nxc.Sensor(IN_3);`	Weist der Variablen den Sensorwert des Lichtsensors zu

`while (true)`
`{`
`  ...Anweisungen...`
`}`
	Dauerschleife – die Anweisungen werden immer wiederholt

`if (licht < GRENZE)`
`{`
`  ...JA-Anweisungen...`
`}`
`else`
`{`
`  ...NEIN-Anweisungen...`
`}`
	Führt die JA-Anweisungen durch, solange die Bedingung Wert der Variablen „licht“ < Wert von der Konstanten „GRENZE“ erfüllt ist
	
	Sonstigenfalls werden die NEIN-Anweisungen ausgeführt - der Programmteil ab „else“ kann auch weggelassen werden

nxc2arduino-Anweisungen – Teil 2

`int i = 0;`
`while (i < 5)`
`{`
`  i += 1;`
`  ...Anweisungen...`
`}`
	Wiederholt die Anweisungen in der Klammer 5 mal.

`for (int i=0; i < 20; i++)`
`{`
`  nxc.Wait(1000);`
`  Serial.print("Fahrzeit: ");`
`  Serial.println(dauer);`
`}`
	Wiederholt die folgenden Anweisungen 20 mal.
	
	- 1 Sekunde warten und
	- „Fahrzeit: “ auf der seriellen Konsole ausgeben und
	- den Wert für die Variable „dauer“ dorthin übertragen (mit einem Zeilenvorschub)

`long zeit;`	Die Variablen `zeit` wird als doppeltgenaue Variable bestimmt.
`fahrzeit  = fahrzeit + 10;`	Der Wert für `fahrzeit` wird um 10 erhöht.
`j -= 1;`	Der Wert für `j` wird um 1 erniedrigt.
`wartezeit = (i * j) / 100;`	Auch komplexere Berechnungen mit anderen Grundrechenarten sind möglich.
`buzzer();`	Gibt einen Ton aus.
`turn_time = random(400);`	Weist der Variablen turn_time einen zufällig ermittelten Wert zwischen 0 und 399 zu.

Weitere Informationen

- Unsere Online-Hilfen zur Arduino-Roboter-Programmierung:
  http://gesamtschule-eppendorf.de/material/informatik/arduino/arduino-roboter/

- zu NXC:
  - (engl.): http://bricxcc.sourceforge.net/nbc/nxcdoc/index.html
  - (dts.): http://lukas.internet-freaks.net/nxt.php

- zu der Programmierung der MiniQ-Roboter:
  - http://www.dfrobot.com/wiki/index.php?title=4WD_MiniQ_Complete_Kit_%28SKU:ROB0050%29

- zur Arduino-Programmierung:
  - http://www.netzmafia.de/skripten/hardware/Arduino/programmierung.html