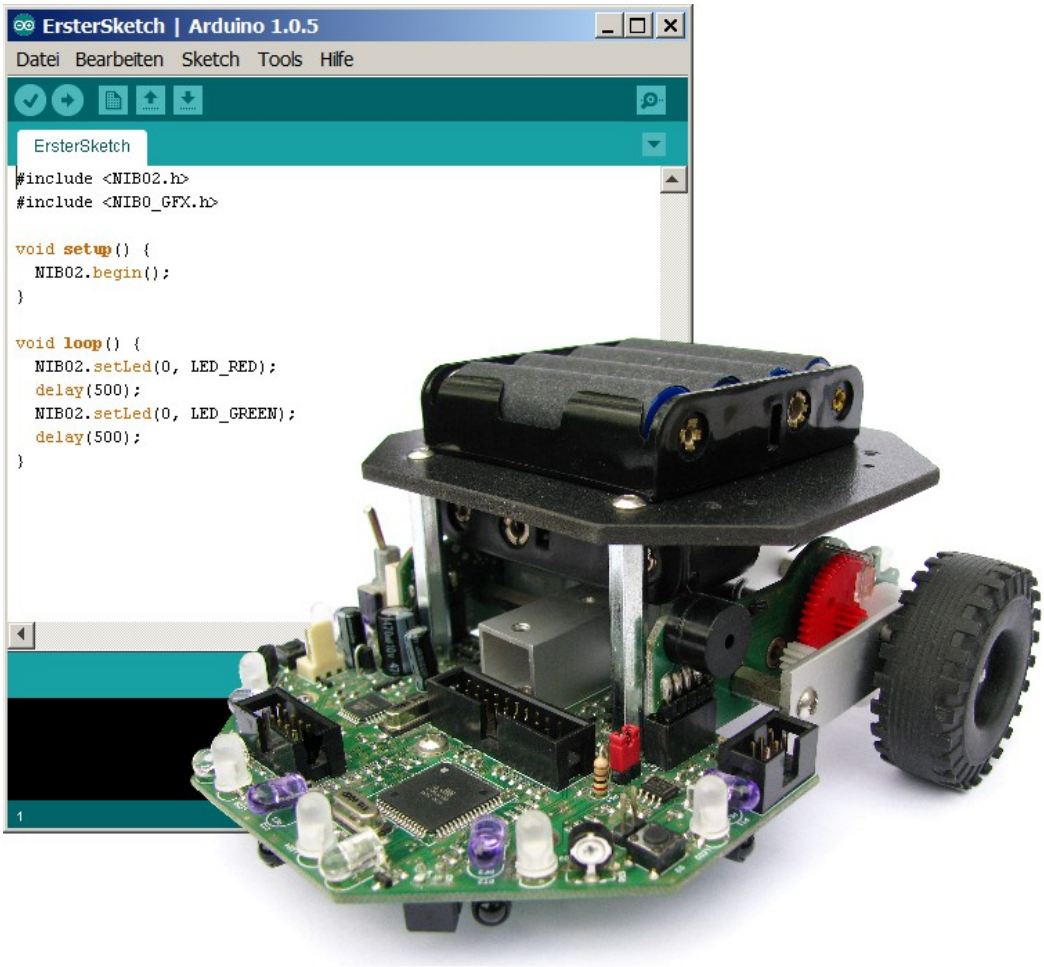
Roboterbausatz NIBO2

ARDUINO 1.05 Programmier-Tutorial



nicai systems

Version 2014/12/02

Inhaltsverzeichnis

1 Einleitung

Dieses Tutorial bietet eine Schritt für Schritt Anleitung für die erste Inbetriebnahme des NIBO2 mit **ARDUINO**. Mittels kleiner, ausführlich erklärter Beispiele soll der Leser mit der grundlegenden Funktionalität der Kernkomponenten des Roboters vertraut gemacht werden.

Sie lernen in den folgenden Abschnitten wie die Programmierumgebung installiert wird, wie die LEDs zum Blinken/Leuchten gebracht werden können, wie Sie Text und Sensorwerte auf dem Display sichtbar machen können und letztendlich auch, wie Sie den NIBO2 in Bewegung bringen!

Das Tutorial richtet sich an Robotik-/Programmieranfänger, um ihnen einen leichten Einstieg in die Gebiete der Programmierung, insbesondere der Mikrocontroller-Programmierung zu ermöglichen.

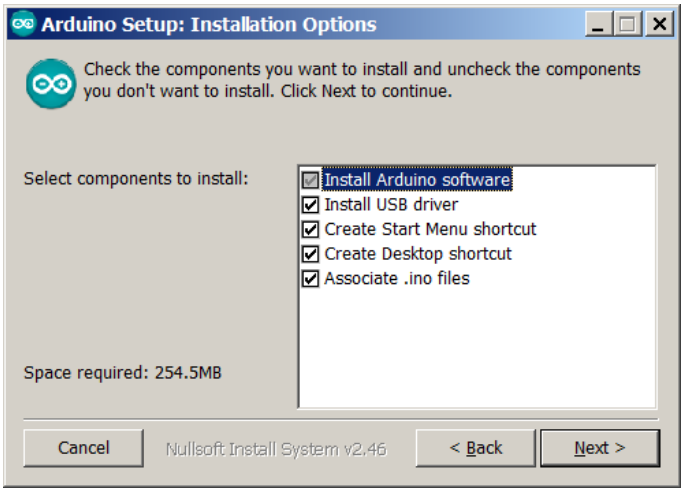
2 Installation der Programmierumgebung

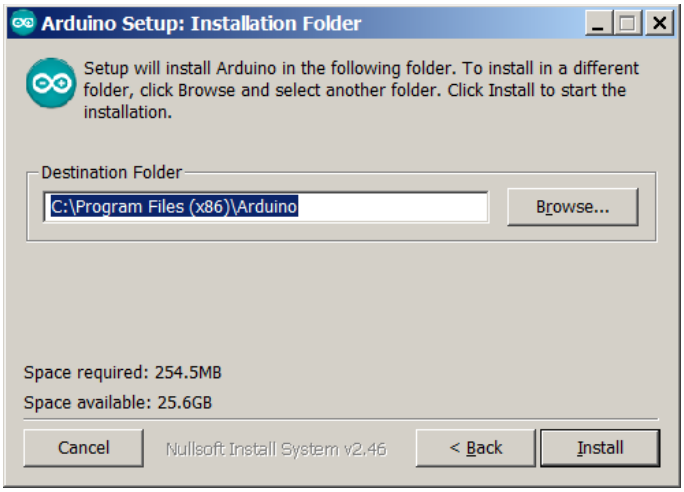
2.1 ARDUINO

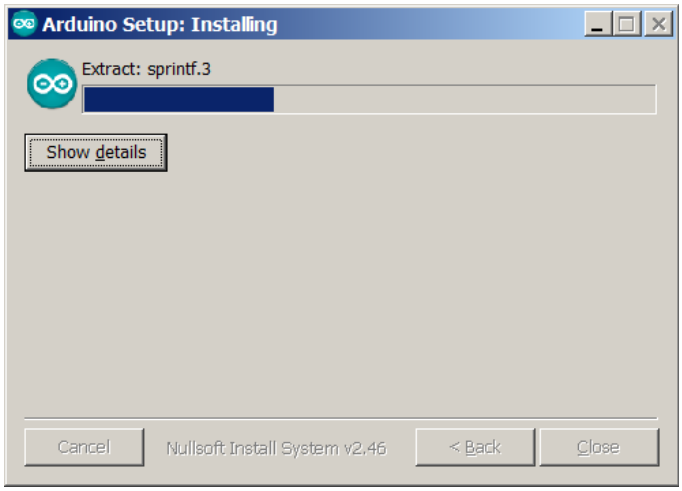
Zunächst muss das ARDUINO installiert werden. Hierzu laden Sie sich die Installationsdatei *Arduino 1.0.5* von **http://arduino.cc** für Ihr entsprechendes Betriebssystem herunter, für Windows wählen Sie die Installer-Version aus.

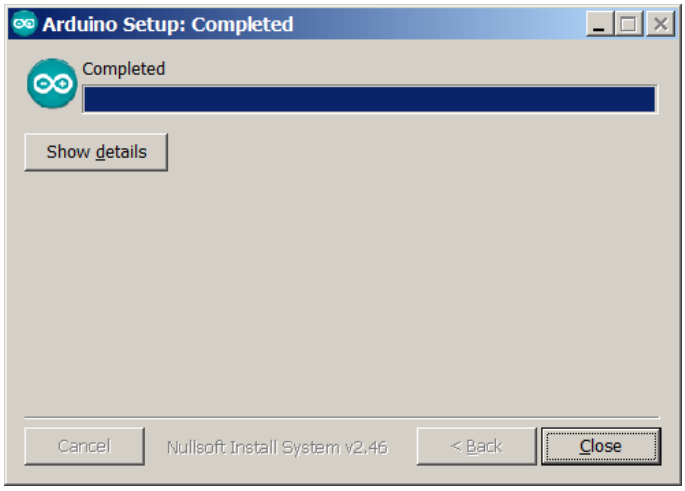
Im Folgenden wird die Installation für Windows beschrieben:

Doppelklicken Sie nun die heruntergeladene Datei ***arduino-1.0.5-windows.exe*** und installieren Sie das Arduino wie vom Installer vorgeschlagen auf Ihrem Computer.




Arduino Setup: Installation Folder
Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.
Destination Folder
C:\Program Files (x86)\Arduino
Browse...
Space required: 254.5MB
Space available: 25.6GB
Cancel
Nullsoft Install System v2.46
< Back
Install


Arduino Setup: Installing
Extract: sprintf.3
Show details
Cancel
Nullsoft Install System v2.46
< Back
Close


Arduino Setup: Completed
Completed
Show details
Cancel
Nullsoft Install System v2.46
< Back
Close

3 Installation der NiboRoboLib

Zunächst wird die NiboRoboLib installiert. Die **neueste** Version und eine **Installationsanleitung** als .pdf-Datei finden sich unter:

http://www.roboter.cc/niboRoboLib

Alternativ befindet sich die Dateien auch auf der beiliegenden CD.

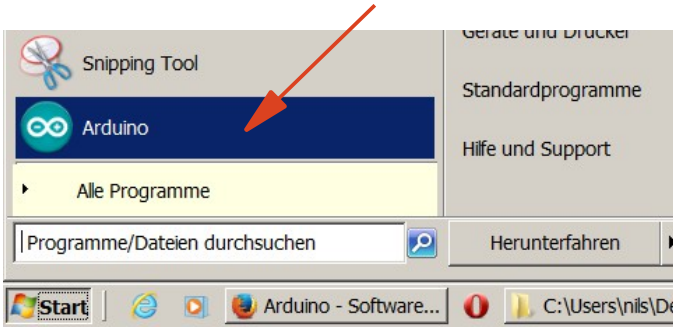
Die **NiboRoboLib** enthält:

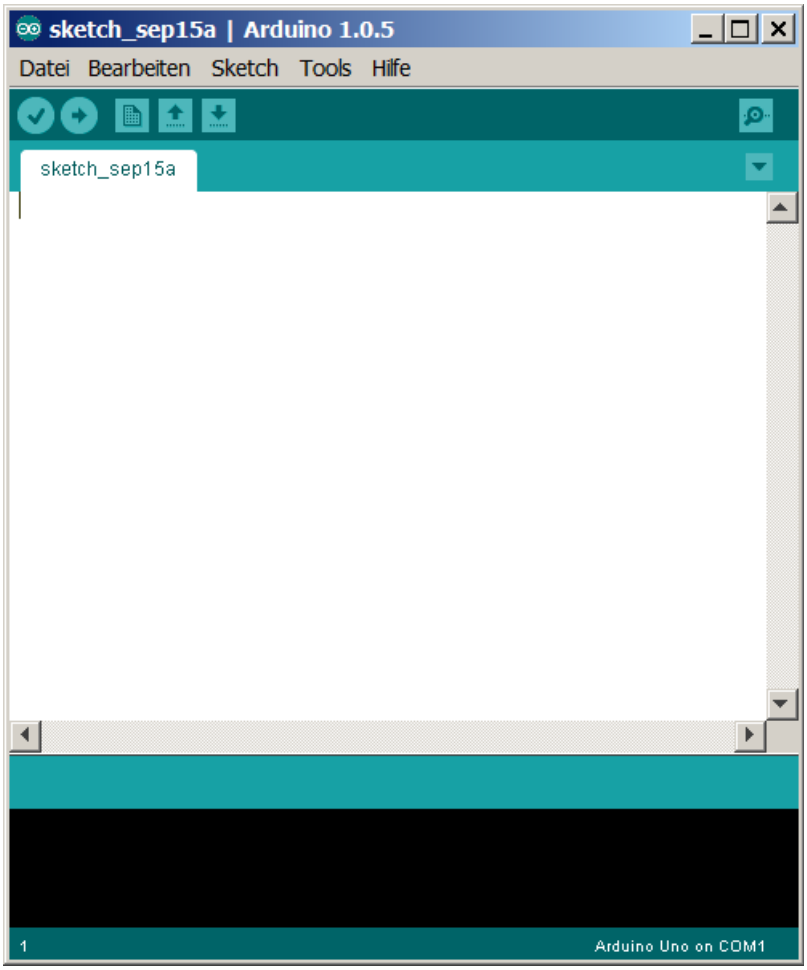
- Alle benötigten **Treiber** für den UCOM-IR2-X
- Alle benötigten **Treiber** für den NIBObee
- **RoboDude** (Übertragungsprogramm für `.hex`- und `.xhex`-Dateien)
- **C-Bibliothek** und **Testprogramme** für den NIBO2
- **C-Bibliothek** und **Testprogramme** für den NIBObee
- **Kalibrierprogramme** für die Sensoren
- **ARDUINO**-Bibliothek für den NIBO2
- **ARDUINO**-Bibliothek für den NIBObee

Während des Installationsvorgangs kann man wählen, ob man alle Pakete installieren möchte, oder nur eine Auswahl.

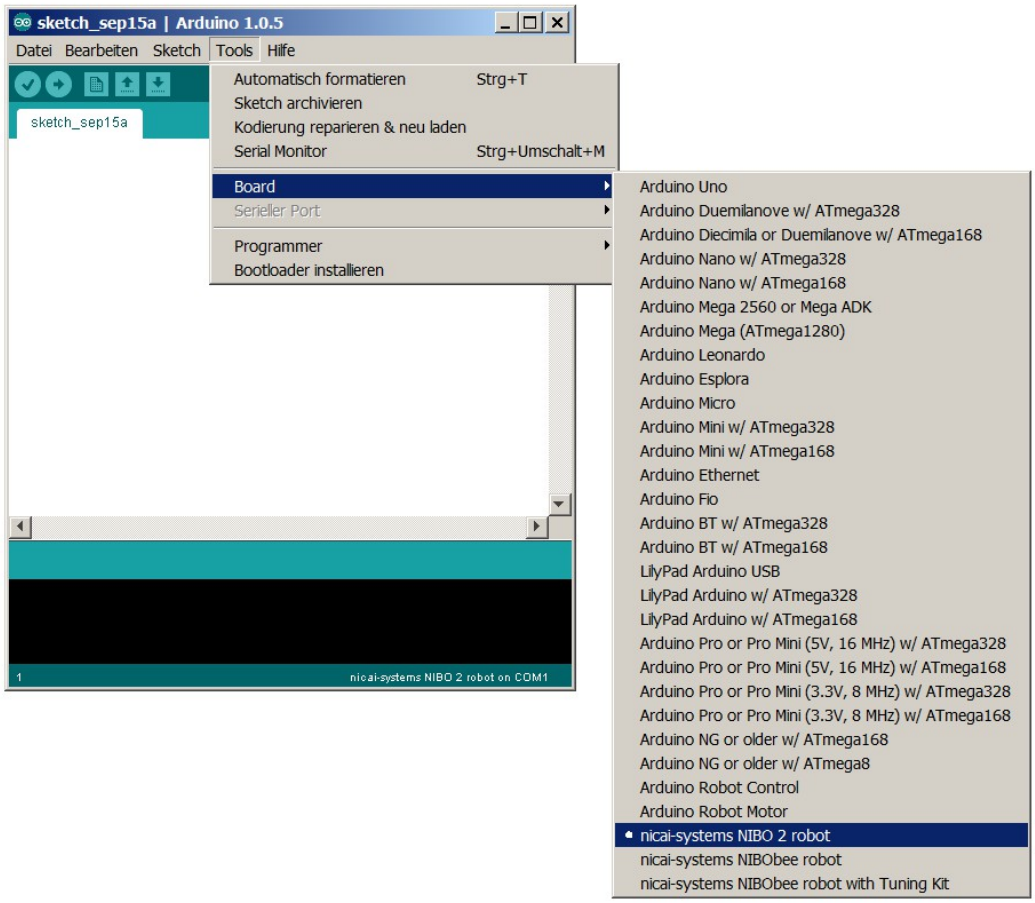
4 Vorbereitungen

Starten Sie die ARDUINO-Oberfläche:

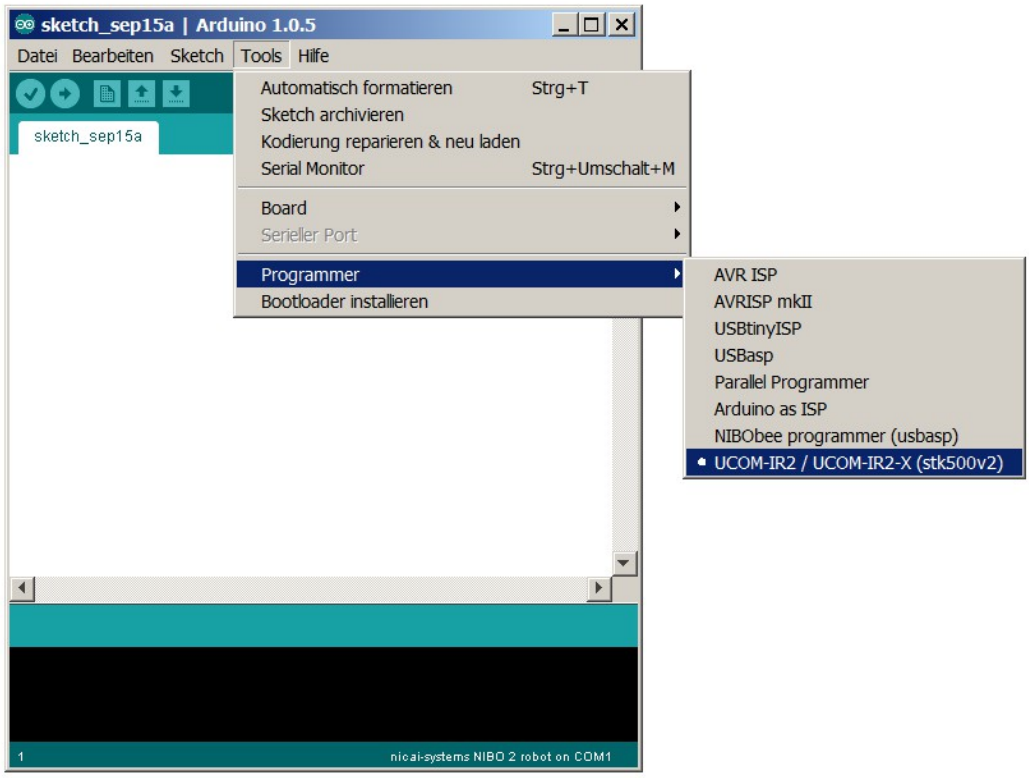




Über ***Tools – Board – nicai-systems NIBO 2 robot***
wird der NIBO2 als Board ausgewählt:

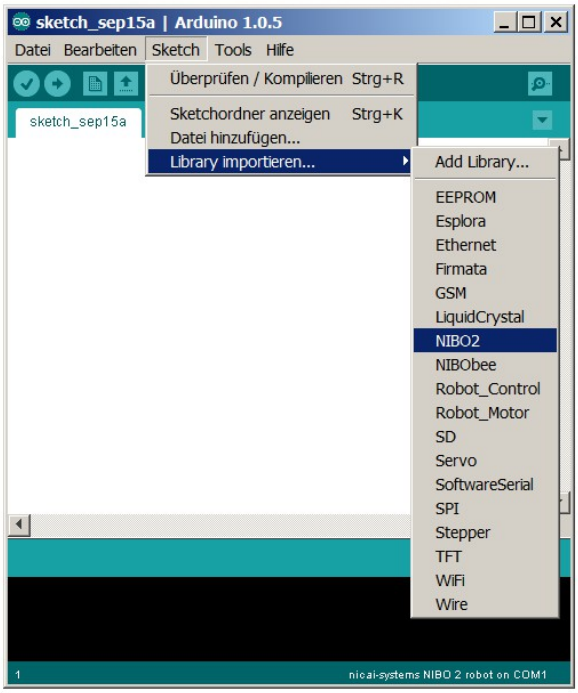


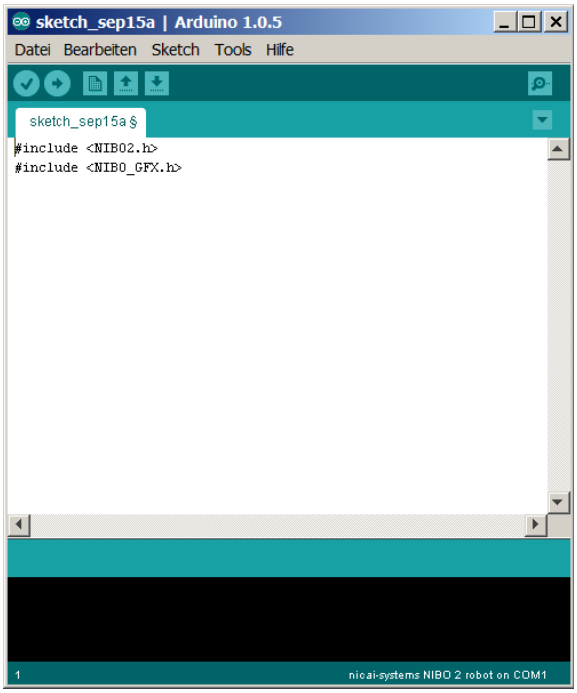
Über ***Tools – Programmer – UCOM-IR2 / UCOM-IR2-X (stk500v2)***
wird der UCOM-IR2-X als Programmer ausgewählt:



5 Der erste Sketch

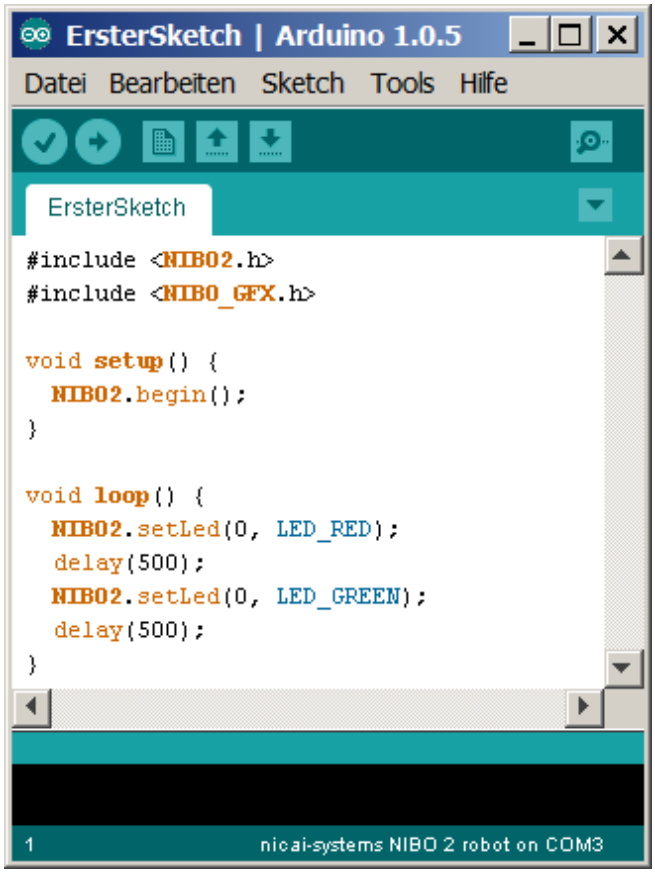
Zunächst wird die NIBO2 Library importiert:





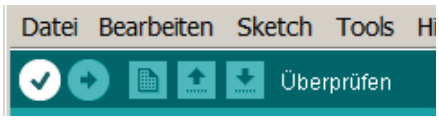
Speichern Sie Ihren Sketch unter dem Namen *ErsterSketch*.

Ergänzen Sie den entstandenen Quellcode wie folgt:



Speichern Sie Ihr geändertes Programm erneut ab!

Jetzt wird das Programm überprüft, indem Sie den ***Überprüfen-Knopf*** drücken:

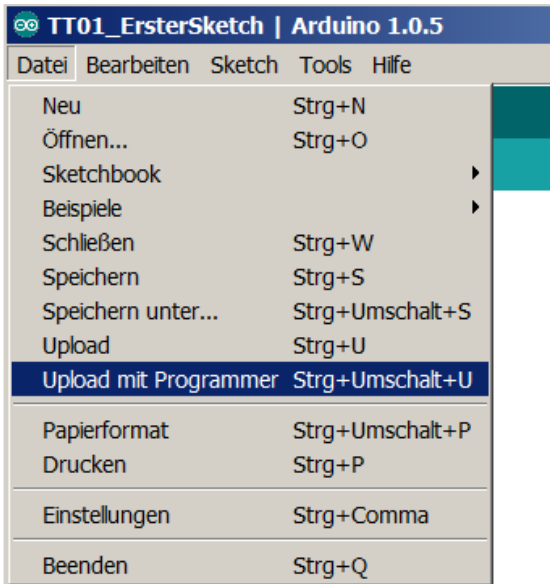


Der Sketch wird nun compiliert und dabei überprüft. Eventuell auftretende Fehlermeldungen erscheinen im schwarzen Bereich des Hauptfensters.

Wenn alles ohne Probleme geklappt hat, erscheint in dem türkisen Bereich die Meldung *Kompilierung abgeschlossen*.

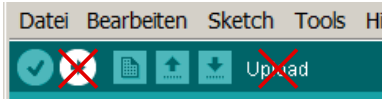
Schließen Sie jetzt den UCOM-IR2(-X) an Ihren Rechner an und wählen Sie im Menü unter „***Tools***“ -> „***Serieller Port***“ die COM-Schnittstelle des Programmieradapters aus:

Nun müssen Sie nur noch den NIBO2 mit dem Programmieradapter verbinden und dann einschalten. Jetzt können Sie den ersten Sketch auf den Roboter übertragen. Wählen Sie dazu im Menü „***Datei***“ -> „***Upload mit Programmer***“ aus:



Wenn das rechte Rücklicht des Roboters (*LED0*) abwechseln rot und grün aufleuchtet, hat alles bestens geklappt!

Hinweis: Es **muss** immer die Funktion „***Upload mit Programmer***“ verwendet werden! Nur „***Upload*“** funktioniert nicht!



Erläuterungen zum Quellcode:

Grundsätzlicher Aufbau eines ARDUINO-Sketches:

```
#include <Bibliothek.h>
  //  durch die #include-Anweisungen werden die gewählten Bibliotheken
  //  eingebunden. Dort sind Klassen, Variablen und Konstanten definiert.


void setup() {
  //  alle Anweisungen, die zwischen den geschweiften Klammern des
  //  setup-Blocks stehen, werden während des Programmablaufs
  //  genau einmal am Anfang ausgeführt.
}

void loop() {
  //  alle Anweisungen, die zwischen den geschweiften Klammern des
  //  loop-Blocks stehen, werden während des Programmablaufs
  //  immer wieder erneut ausgeführt.
}
```

Zurück zu unserem ersten Sketch:

ErsterSketch

```
1  #include <NIBO2.h>
2  #include <NIBO_GFX.h>
3
4  void setup() {
5    NIBO2.begin();
6  }
7
8  void loop() {
9    NIBO2.setLed(0, LED_RED);
10   delay(500);
11   NIBO2.setLed(0, LED_GREEN);
12   delay(500);
13 }
```

01-02 In den ersten beiden Zeilen werden durch die `#include` Anweisungen die Header-Dateien der NIBO2 Bibliothek eingebunden.

04-06 Zwischen der Zeile „`void setup() {`“ und der Zeile mit der schließenden geschweiften Klammer „`}`“ befindet sich der Funktionsblock für die Programminitialisierung. Die Funktion wird genau einmal nach dem Einschalten des Roboters ausgeführt.

Durch die Anweisung „**NIBO2**.begin();“ wird der Roboter initialisiert.
Wichtig: Diese Anweisung **muss** in jeder setup-Funktion als erstes ausgeführt werden!

08-13 Zwischen der Zeile „`void loop() {`“ und der Zeile mit der schließenden geschweiften Klammer „`}`“ befindet sich die Hauptroutine des Programms. Die Funktion wird immer wieder, bis zum Ausschalten des Roboters ausgeführt.

09 Innerhalb des Funktionsblocks der Hauptroutine wird mit der Anweisung „**NIBO2**.setLed(0, LED_RED);“ die LED 0 auf rot geschaltet.

10 Anschließend wird durch die Anweisung „`delay(500);`“ eine Pause von einer halben Sekunde eingelegt, das entspricht 500 Millisekunden.

11-12 Nach der Pause wird die LED 0 auf grün geschaltet und mit der nächsten Zeile wird wiederum eine Pause von 500 Millisekunden eingelegt.

13 Nach der Pause endet die loop-Funktion und wird anschließend automatisch wieder von vorn abgearbeitet.

Das war´s schon!

Tip: Alle Beispiele aus diesem Tutorial sind auch unter „**Datei**“ -> „**Beispiele**“ -> „**NIBO2**“ -> „**Tutorial**“ zu finden.

6 LEDs in Aktion

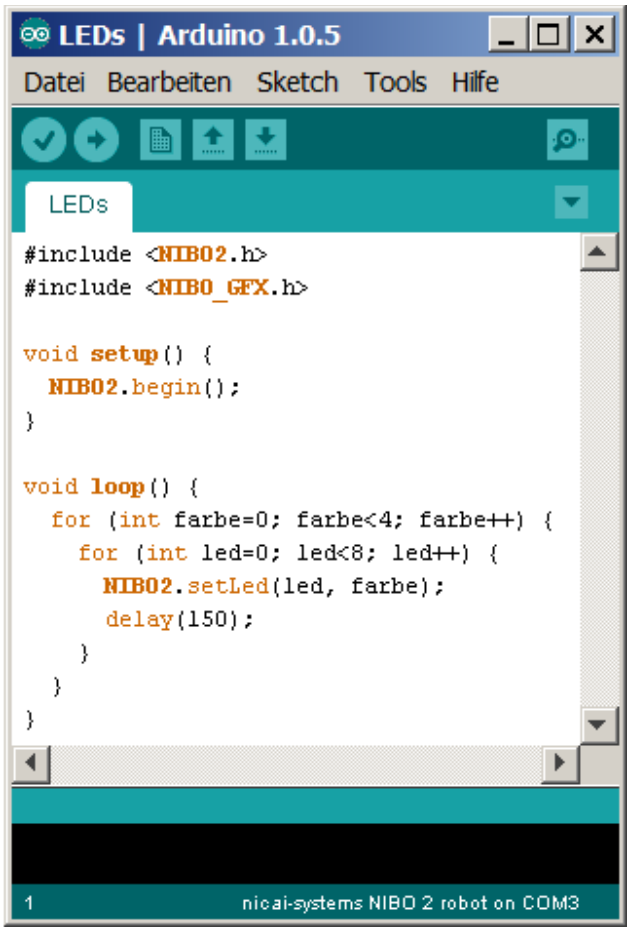
Hier wollen wir nun etwas kompliziertere Leuchtmuster gestalten.

Erzeugen Sie einen neuen Sketch, indem Sie im Menü „***Datei***“ -> „***Neu***“ auswählen. Speichern Sie den neuen Sketch unter dem Namen **LEDs**.

Als Bibliothek importieren Sie wieder „NIBO2“ wie in Kapitel 4 beschreiben.

Bei diesem Sketch sollen die Leuchtdioden nacheinander reihum in einer Farbe eingeschaltet werden. Die Farbe soll bei jedem Umlauf zwischen *Aus*, *Rot*, *Grün* und *Orange* wechseln.

Tippen Sie den folgenden Quellcode in das Editorfenster ein:



Erläuterungen zum Quellcode:

`01-06` Dieser Quellcode unterscheidet sich nur im Anweisungsblock der loop-Funktion von dem vorherigen Programm aus Kapitel 4.

`08-15` Die erste Anweisung innerhalb der loop-Funktion „`for (int farbe=0; farbe<4; farbe++)`“ dient zur mehrfachen Ausführung der Anweisungen im Block zwischen den geschweiften Klammern.

Bei dieser „for-Schleife“ wird der Block genau 4 mal ausgeführt, dabei hat die Variable „`farbe`“ jeweils einen anderen Wert: Beim ersten Durchlauf den Wert 0, und bei den folgenden Durchläufen die Werte 1, 2 und 3.

Die for-Schleife ist ein sehr mächtiges Universalwerkzeug - generell besteht sie aus den folgenden Komponenten:

```
for (Initialisierung; Bedingung; Fortsetzung) {
  // dieser Block wird solange ausgeführt, wie die Bedingung wahr ist.
}
```

`09-14` Im Initialisierungsbereich wird die Ganzzahl-Variable „`farbe`“ angelegt und mit dem Wert 0 initalisiert. Die Bedingung für einen Durchlauf ist, dass der Wert der Variablen `farbe` kleiner als 4 sein muss. Im Fortsetzungsbereich wird der Wert der Variablen `farbe` für den nächsten Durchlauf um eins erhöht.
Der Ausdruck „`farbe++`“ ist dabei eine Abkürzung für den Ausdruck „`farbe = farbe + 1`“.
Damit läuft die äußere for-Schleife über die Variable `farbe`, die nacheinander die Werte 0, 1, 2 und 3 annimmt.

`10-13` Die innere for-Schleife läuft über die Variable `led`. Die Variable nimmt dabei nacheinander die Werte 0 bis 7 an.

11 - 12

Im Anweisungsblock der inneren for-Schleife stehen die eigentlichen Anweisungen:

Mit der Anweisung „**NIBO2**.setLed(`led`, `farbe`);“ wird der LED mit der Nummer **`led`** die Farbe **`farbe`** zugewiesen. Die Werte der Farbe sind folgendermaßen definiert:

0: LED aus
1: LED rot
2: LED grün
3: LED orange

Durch die letzte Anweisung im Block wird eine Zeitspanne von 150 ms gewartet, bevor der nächste Schleifendurchlauf beginnt.

Ideen & Anregungen:

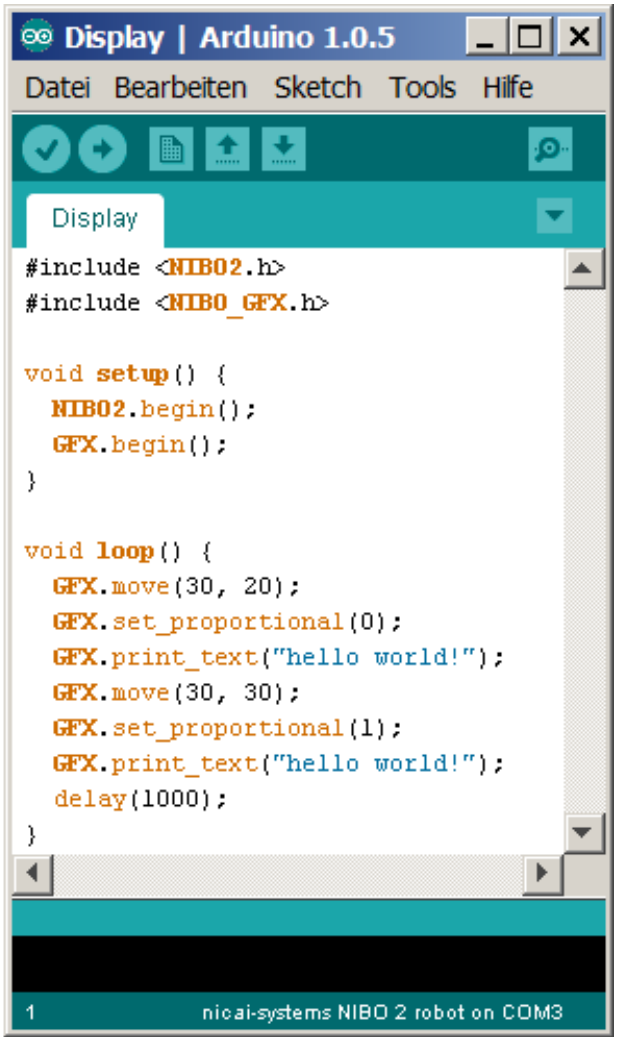
1. Verändern Sie Ihr Programm so, dass die Leuchtdioden langsamer hintereinander aufleuchten.
2. Verändern Sie Ihr Programm so, dass die Leuchtdioden schneller hintereinander aufleuchten.
3. Verändern Sie Ihr Programm so, dass sich die Laufrichtung umkehrt, sprich die Diode LED0 soll in jedem Durchlauf als letzte LED aufleuchten.
4. Verändern Sie Ihr Programm so, dass jede LED einzeln alle Farben durchläuft (Aus, Rot, Grün und Orange), bevor die nächste beginnt.

7 Inbetriebnahme des Displays

Als nächstes wollen wir uns die Ansteuerung des Grafikdisplays vornehmen: Dazu legen Sie wieder einen neuen Sketch an, den Sie diesmal unter dem Namen **Display** speichern (wie in Kapitel 4 beschrieben).

Unser Programm soll den Text „**hello world!**“ auf dem Display ausgeben.

Tippen Sie den folgenden Quellcode in das Editorfenster ein:



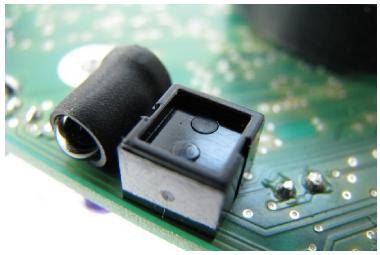
Erläuterungen zum Quellcode:

01-02	Zunächst werden wie immer die beiden Header-Dateien *NIBO2.h* und *NIBO_GFX.h* eingebunden.
04-07	In der setup-Funktion wird dieses mal nicht nur der Roboter selbst initialisiert, sondern durch den Aufruf der Funktion **GFX**.begin() wird auch das LC-Display initialisiert.
09-17	In der loop-Funktion wird mit dem Befehl **GFX**.move(30, 20) der Grafikcursor an die Stelle bewegt, die 30 Pixel vom linken Rand und 20 Pixel von der Oberkante des Displays entfernt liegt.
11	Die Anweisung **GFX**.set_proportional(0) sorgt dafür, dass die Texte nicht in Proportionalschrift ausgedruckt werden, das heißt alle ausgegebenen Zeichen haben die gleiche Breite.
12	Nun kann auch schon mit **GFX**.print_text(„hello world!“) der Text „hello world!“ ausgegeben werden.
13	Jetzt wandert der Cursor durch den Aufruf der Funktion **GFX**.move(30, 30) eine Zeile (zehn Pixel) tiefer.
14-15	Der Text soll nun in Proportional-Schrift ausgeben werden, dies geschieht durch den Aufruf der Funktion **GFX**. set_proportional(1). Die folgende Anweisung gibt wiederum den Text „hello world!“ auf dem Display aus, nur diesmal in der anderen Schriftart!
16	Jetzt warten wir noch eine Sekunde bevor wir dann im Sekundentakt das Bild neu aufbauen (das sollte man jedoch kaum sehen...)

Ideen & Anregungen:

1. Geben Sie die Texte „links oben“, „rechts oben“, „links unten“ und „rechts unten“ an den jeweiligen Stellen auf dem Display aus.
2. Geben Sie mittig auf dem Display einen Countdown aus, der Countdown soll im Sekundentakt von 20 auf 0 Zählen. Benutzen Sie dazu die `delay()` Funktion, achten Sie auf eine saubere Ausgabe!

8 Linien- / Bodensensoren



In diesem Kapitel wollen wir uns mit den beiden Boden- und mit den beiden Liniensensoren beschäftigen. Die Sensoren arbeiten alle nach dem gleichen Prinzip, da sich die beiden zentralen jedoch gut für die Detektierung einer Linie eignen, werden sie zur besseren Unterscheidung Liniensensoren genannt.

Die Sensoren arbeiten nach dem IR-Reflexionsverfahren. Die Helligkeit des Bodens wird automatisch zweimal gemessen, einmal bei eingeschalteter und einmal mit ausgeschalteter IR-LED. Dadurch lassen sich die Einflüsse des Umgebungslichts minimieren.

Die gemessenen Werte werden von der Bibliothek normalisiert und stehen als Ganzzahlen im Bereich von **0 (schwarz)** bis **1024 (weiß)** zur Verfügung.

Damit die Normalisierung durchgeführt werden kann, müssen die Sensoren zuvor kalibriert werden. Die Parameter der Kalibrierung werden im EEPROM dauerhaft gespeichert. Bei einer Neuprogrammierung des ATmega128 bleiben die Kalibrierwerte erhalten!

8.1 Kalibrierung der Linien- / Bodensensoren

Um die Bodensensoren des NIBO2 zu kalibrieren öffnen Sie den Sketch ***Kalibrierung.ino***, den sie im Menü unter „**Datei**“->„**Beispiele**“->„**NIBO2**“ ->„**Kalibrierung**“ finden, und übertragen das Programm auf den Roboter.

Nach abgeschlossener Übertragung können Sie die Sensoren kalibrieren:

1. *Rechtes Rücklicht leuchtet*: Stellen Sie den Roboter auf einen **schwarzen Untergrund** und drücken Sie den Taster S3.
2. *Linkes Rücklicht leuchtet*: Stellen Sie den Roboter auf einen **weißen Untergrund** und drücken Sie den Taster S3.
3. *Beide Rücklichter leuchten*: Drücken Sie den Taster S3 um die **Parameter** im EEPROM zu **speichern**.

8.2 Anzeige der Werte der Linien- / Bodensensoren

Wir werden in diesem Beispiel nicht nur die Bodensensoren kennen lernen, sondern auch das Display im Terminal-Modus verwenden.

Der Terminal-Modus funktioniert ähnlich wie die Kommandozeile unter Windows, andere Bezeichnungen hierfür sind Shell, Konsole oder DOS-Fenster.

Im Terminalmodus können nur normale Zeichen (ASCII-Zeichen) verwendet werden. Diese werden in einem festen Raster auf dem Display ausgegeben. Wenn das Ende einer Zeile erreicht ist, wird der weitere Text automatisch in der nächsten Zeile ausgegeben. Wird der untere Bildschirmrand erreicht, dann scrollt der Text automatisch nach oben weg.

Legen Sie wie in den vorangegangenen Beispielen einen neuen Sketch an, und speichern diesen unter dem Namen „**Bodensensoren**“. Importieren Sie die NIBO2-Bibliothek und ergänzen Sie den Quellcode im Editorfenster:

```
 1 #include <NIBO2.h>
 2 #include <NIBO_GFX.h>
 3
 4 void setup() {
 5   NIBO2.begin();
 6   GFX.begin();
 7   GFX.term.home();
 8   GFX.term.println("TT04: Bodensensoren");
 9   GFX.term.println("R1   R0   L0   L1");
10 }
11
12 void loop() {
13   delay(100);
14
15   GFX.term.cursor(0, 2);
16   GFX.term.clearLine();
17   GFX.term.print(FloorSensor.getR1());
18   GFX.term.cursor(5, 2);
19   GFX.term.print(FloorSensor.getR0());
20   GFX.term.cursor(10, 2);
21   GFX.term.print(FloorSensor.getL0());
22   GFX.term.cursor(15, 2);
23   GFX.term.print(FloorSensor.getL1());
24
25   GFX.term.cursor(0, 3);
26   GFX.term.clearLine();
27   GFX.term.print(FloorSensor.getR1(1));
28   GFX.term.cursor(5, 3);
29   GFX.term.print(FloorSensor.getR0(1));
30   GFX.term.cursor(10, 3);
```

```
31    GFX.term.print(FloorSensor.getL0(1));
32    GFX.term.cursor(15, 3);
33    GFX.term.print(FloorSensor.getL1(1));
34
35    GFX.term.cursor(0, 5);
36    GFX.term.clearLine();
37    GFX.term.print("supply: ");
38    GFX.term.print(NIBO2.getVoltage()/1000.0);
39    GFX.term.println(" V");
40  }
```

Erläuterungen zum Quellcode:

01-02 Mit den ersten beiden Zeilen wird - wie immer - die NIBO2-Bibliothek importiert.

`setup`-Funktion:

05-06 Wir initialisieren, wie in den vorangegangenen Beispielen, zuerst den Roboter und danach das Display.

07 Anschließend wird Text auf dem Display ausgegeben:
Durch Aufruf der Funktion **GFX**.term.home() wird der Cursor in die linke obere Ecke des Displays gesetzt.

08-09 Die Funktion **GFX**.term.println(„text“) gibt den Text an der aktuellen Stelle aus und setzt den Cursor an den Anfang der nächsten Zeile.

`loop`-Funktion:

13 In der ersten Zeile warten wir erst einmal 100 ms.

15 Danach bewegen wir den Cursor mit dem Befehl **GFX**.term.cursor(0, 2) an den Anfang der dritten Zeile. Da wir bei Null anfangen zu zählen, lautet die Koordinate für das erste Zeichen der dritten Zeile (0, 2).

16 Als nächstes sorgen wir erst einmal dafür, das die Zeile gelöscht wird. Dazu dient der Aufruf **GFX**.term.clearLine().

17 Der Ausdruck **FloorSensor**.getR1() liefert uns den aktuellen kalibrierten Messwert des rechten Bodensensors. Die normalisierten Werte liegen zwischen 0 (schwarz) und 1024 (weiß), sie können jedoch auch größer sein, wenn die Kalibrierung auf einem „dunkleren“ weiß durchgeführt wurde.

18-23	In den weiteren Zeilen werden die Werte des rechten Liniensensors (getR0()), des linken Liniensensors (getL0()) und des linken Bodensensors (getL1()) ausgegeben.
25-33	Im anschließenden Abschnitt werden die absoluten Helligkeiten auf dem Display ausgegeben. Die absoluten Helligkeiten bekommt man, indem eine Eins als Parameter der jeweiligen getXX-Funktion übergeben wird.
35-37	Zum Schluss wird die aktuelle Versorgungsspannung ausgegeben. Dazu bewegen wir den Cursor in die Zeile 6, löschen diese und geben den Text „`supply: `“ aus.
38-39	Die Funktion **NIBO2**.getVoltage() liefert uns die Spannung in Millivolt. Um die Spannung in Volt um zurechnen, muss der Wert durch 1000 geteilt werden. Da die Division als Kommazahl vorgenommen werden soll, müssen wir durch 1000.0 teilen. Ohne den Dezimalpunkt wird der Compiler eine Ganzzahl als Ergebnis erzeugen (z.B. 9V anstatt 9,13 V)! Damit es schöner aussieht geben wir zum Schluss noch ein „ V“ auf dem Display aus...

9 Distanzsensoren

In diesem Beispiel wollen wir uns mit den Distanzsensoren des Roboters beschäftigen. Die fünf Distanzsensoren werden von dem separaten Mikrocontroller ATmega88 (COPRO) angesteuert. Die Messung erfolgt ähnlich wie bei den Bodensensoren nach dem IR-Reflexionsverfahren. Dabei wird gemessen, welcher Anteil vom ausgesendeten Licht zurück reflektiert wird. Alle Messwerte sollen auf dem Grafikdisplay ausgegeben werden.

Zusätzlich soll wieder die aktuelle Versorgungsspannung des Roboters ausgegeben werden.

Legen Sie wie in den vorangegangenen Beispielen einen neuen Sketch an, und speichern diesen unter dem Namen „**Distanzsensoren**“. Importieren Sie die NIBO2-Bibliothek und tippen Sie anschließend folgenden Quellcode in das Editorfenster ein:

```
 1 #include <NIBO2.h>
 2 #include <NIBO_GFX.h>
 3
 4 void setup() {
 5   NIBO2.begin();
 6   GFX.begin();
 7   DistSensor.beginMeasure();
 8   GFX.term.home();
 9   GFX.term.println("TT05: Distanzsensoren");
10   GFX.term.println("R   FR  F   FL  L");
11 }
12
13 void loop() {
14   delay(100);
15
16   GFX.term.cursor(0, 2);
17   GFX.term.clearLine();
18   GFX.term.print(DistSensor.get(0), HEX);
19   GFX.term.cursor(16, 2);
20   GFX.term.print(DistSensor.get(4), HEX);
21
22   GFX.term.cursor(4, 3);
23   GFX.term.clearLine();
24   GFX.term.print(DistSensor.get(1), HEX);
25   GFX.term.cursor(16, 3);
26   GFX.term.print(DistSensor.get(3), HEX);
27
28   GFX.term.cursor(8, 4);
```

```
29    GFX.term.clearLine();
30    GFX.term.print(DistSensor.get(2), HEX);
31
32    GFX.term.cursor(0, 6);
33    GFX.term.clearLine();
34    GFX.term.print("supply: ");
35    GFX.term.print(NIBO2.getVoltage()/1000.0);
36    GFX.term.println(" V");
37  }
```

Erläuterungen zum Quellcode:

01-02	Mit den ersten beiden Zeilen wird - wie immer - die NIBO2-Bibliothek importiert.
	`setup`-Funktion:
05-06	Wir initialisieren, wie in den vorangegangenen Beispielen, zuerst den Roboter und danach das Display.
07	Jetzt starten wir die Distanzmessung durch Aufruf der Funktion **DistSensor**.beginMeasure().
08-10	Anschließend geben wir wieder ein paar Texte auf dem Display aus.
	`loop`-Funktion:
16-30	Die Werte der Distanzsensoren bekommen wir durch Aufruf der Funktion **DistSensor**.get(sensorNr). Die Zahlen liegen im Wertebereich von 0 bis 255. Dieses Mal wollen wir die Zahl jedoch nicht im Dezimalsystem ausgeben, sondern als hexadezimale Zahl. Dies geschieht mit dem zweiten Parameter „**HEX**“ beim Aufrufen der print-Funktion.
32-36	Im letzten Abschnitt geben wir wiederum die aktuelle Betriebsspannung aus.

10 Bewegung – Ab die Post!

Im letzten Beispiel wollen wir etwas Bewegung ins Spiel bringen: Der Roboter soll ein Stück vorwärts fahren, kurz warten, dieselbe Strecke rückwärts fahren, wieder kurz warten und von vorne beginnen. Die Motoransteuerung wird auch vom COPRO durchgeführt.

Dabei soll auch diesmal ständig die Batteriespannung auf dem Display ausgegeben werden.

Legen Sie wie in den vorangegangenen Beispielen einen neuen Sketch an, und speichern diesen unter dem Namen „**Bewegung**“. Importieren Sie die NIBO2-Bibliothek und tippen Sie anschließend folgenden Quellcode in das Editorfenster ein:

```
 1 #include <NIBO2.h>
 2 #include <NIBO_GFX.h>
 3
 4 void setup() {
 5   NIBO2.begin();
 6   GFX.begin();
 7   GFX.term.home();
 8   GFX.term.println("TT06: Bewegung");
 9 }
10
11 int counter = 0;
12
13 void loop() {
14   delay(100);
15
16   GFX.term.cursor(0, 1);
17   GFX.term.clearLine();
18   GFX.term.print("supply: ");
19   GFX.term.print(NIBO2.getVoltage()/1000.0);
20   GFX.term.println(" V");
21
22   counter++;
23
24   switch (counter) {
25     case 20:
26       Engine.setSpeed(20, 20);
27       break;
28
29     case 40:
30       Engine.stop();
31       break;
32
```

```
33      case 60:
34        Engine.setSpeed(-20, -20);
35        break;
36
37      case 80:
38        Engine.stop();
39        counter = 0;
40        break;
41    }
42  }
```

Erläuterungen zum Quellcode:

01-02	Mit den ersten beiden Zeilen wird - wie immer - die NIBO2-Bibliothek importiert.
	`setup`-Funktion:
05-06	Wir initialisieren, wie in den vorangegangenen Beispielen, zuerst den Roboter und danach das Display.
07-08	Anschließend geben wir wieder Text auf dem Display aus.
11	In dieser Zeile definieren wir uns die globale Variable „`counter`“, mit der wir uns merken können was wir gerade tun...
	`loop`-Funktion:
14	In der ersten Zeile warten wir erst einmal 100 ms.
16-20	Danach geben wir zuerst die aktuelle Batteriespannung auf dem Display aus.
22	Mit der Anweisung „`counter++`“ erhöhen wir den Wert der Variablen „`counter`“ um Eins.

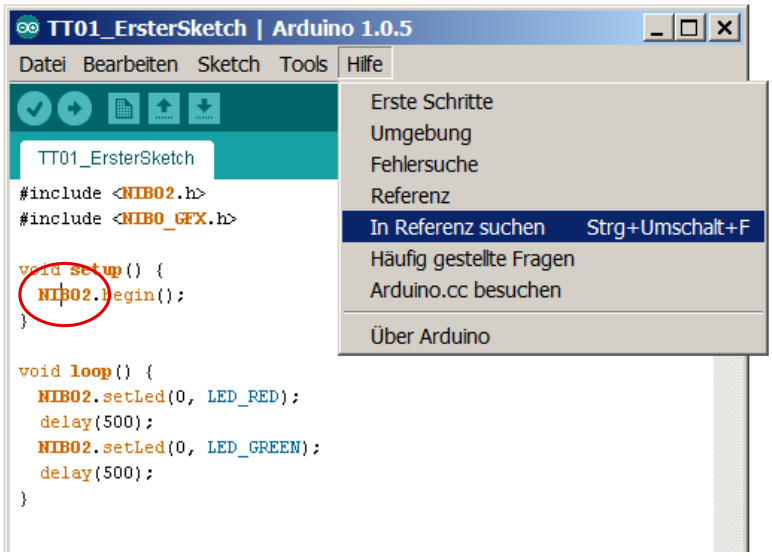
24-41	Die switch-Anweisung führt die verschiedenen Blöcke in Abhängigkeit vom Wert der Variablen „`counter`“ aus: Nur bei den Werten 20, 40, 60 und 80 werden die entsprechenden Blöcke abgearbeitet:
25-27	Wenn die Variable „`counter`“ den Wert 20 hat, wird durch Aufruf der Funktion **Engine**.setSpeed(20, 20) die Geschwindigkeit der beiden Motoren auf 20 Ticks pro Sekunde gesetzt.
29-31	Wenn die Variable „`counter`“ den Wert 40 erreicht, wird die Funktion **Engine**.stop() aufgerufen, dadurch stoppen die Motoren.
33-35	Beim Erreichen des Wertes 60 setzt sich der Roboter durch Aufruf der Funktion **Engine**.setSpeed(-20, -20) in umgekehrter Richtung in Bewegung.
37-40	Wenn die Variable „`counter`“ den Wert 80 erreicht, stoppt der Roboter. Durch manuelles Setzen der Variablen „`counter`“ auf den Wert 0 wird das Programm wieder von vorne abgearbeitet.

Ideen & Anregungen:

1. Geben Sie den Wert der Variablen `counter` auf dem Display aus.
2. Lassen Sie den Roboter eine kurze Zeit lang im Uhrzeigersinn drehen, und anschließend gegen den Uhrzeigersinn.

11 Dokumentation

Eine Übersicht über die NIBO2 ARDUINO Bibliothek kann folgendermaßen geöffnet werden: Klicken Sie im Editor in das Wort NIBO2 (siehe Bild) und wählen Sie aus dem Menü „**Hilfe**“ -> „**In Referenz suchen**“ aus:



Nun öffnet sich in Ihrem Webbrowser folgendes Fenster:

NIBO2 robot class reference for **ARDUINO**

NIBO2 · NIBO GFX · NDS3 · **ARDUINO**-Referenz

Einbindung durch: **#include <NIBO2.h>**

class instance **NIBO2**

Funktion	Beschreibung
void **begin** ()	Initialisierung des NIBO2 Roboters
unsigned int **getVoltage** ()	Versorgungsspannung in Millivolt (9.6V ≙ 9600) zurückgeben
void **setLed** (unsigned char led, unsigned char value)	Status-LED setzen, Farben: LED_OFF, LED_GREEN, LED_RED, LED_ORANGE
unsigned char **getLed** (unsigned char led)	Status-LED abfragen, Farben: LED_OFF, LED_GREEN, LED_RED, LED_ORANGE
unsigned int **getRandomSeed** ()	Zufallszahlen-Basis liefern
void **setLedsIntensity** (int light)	Helligkeit (0-1024) für die Status-LEDs (rot/grün) setzen
void **setHeadlightsIntensity** (int light)	Helligkeit (0-1024) für die Scheinwerfer setzen
void **setDisplayIntensity** (int light)	Helligkeit (0-1024) für die Displaybeleuchtung setzen
unsigned int **getAnalog** (unsigned char adc_channel, unsigned char active)	Rohwert eines analogen Kanals auslesen

class instance **Engine**

Funktion	Beschreibung
void **begin** ()	Initialisierung
void **setPWM** (int left, int right)	PWM Werte (-1023 ... 0 ... +1023) für die beiden Motoren setzen
void **setSpeed** (int left, int right)	Geschwindigkeit für die beiden Motoren setzen. Die Werte werden in Ticks/Sekunde angegeben. 40

12 Links zu weiterführenden Internetseiten

In diesem Unterkapitel ist eine ausgewählte Linksammlung zu themenähnlichen Internetseiten aufgeführt.

Entwicklungsumgebungen:

Arduino: http://www.arduino.cc
Webseite der Arduino-Entwicklungsumgebung. Dort gibt es die Open Source Arduino Oberfläche für verschiedene Betriebssysteme zum Download.

Atmel: http://www.atmel.com
Webseite vom Hersteller der Mikrocontroller. Dort gibt es Datenblätter, Applikationsbeispiele und die Entwicklungsumgebung AVRStudio.

WinAVR: http://winavr.sourceforge.net/
AVR-GCC Compiler für Windows mit vielen Extras und „Add-on“ für das AVRStudio.

AVRDude: http://savannah.nongnu.org/projects/avrdude/
Freie Programmiersoftware (Downloader, für den Nibo geeignet!).

Roboter.CC: http://www.roboter.cc
Online Code Compiler speziell für Robotik-Projekte mit vielen Beispielen und Forum.

Weitere Informationen:

- **Nibo Hauptseite**: http://nibo.nicai-systems.de
  Die Homepage des Nibo Herstellers. Liefert technische Informationen, die Bauanleitung und weitere Links.
- **Programmieradapter UCOM-IR**: http://ucom-ir.nicai-systems.de
- **Nibo Wiki**: http://www.nibo-roboter.de
  Wiki des Nibo. Liefert alle Informationen rund um den Nibo.
- **Mikrocontroller**: http://www.mikrocontroller.net
  Alles über Mikrocontroller und deren Programmierung.
- **AVRFreaks**: http://www.avrfreaks.net
  Informationen rund um den AVR.