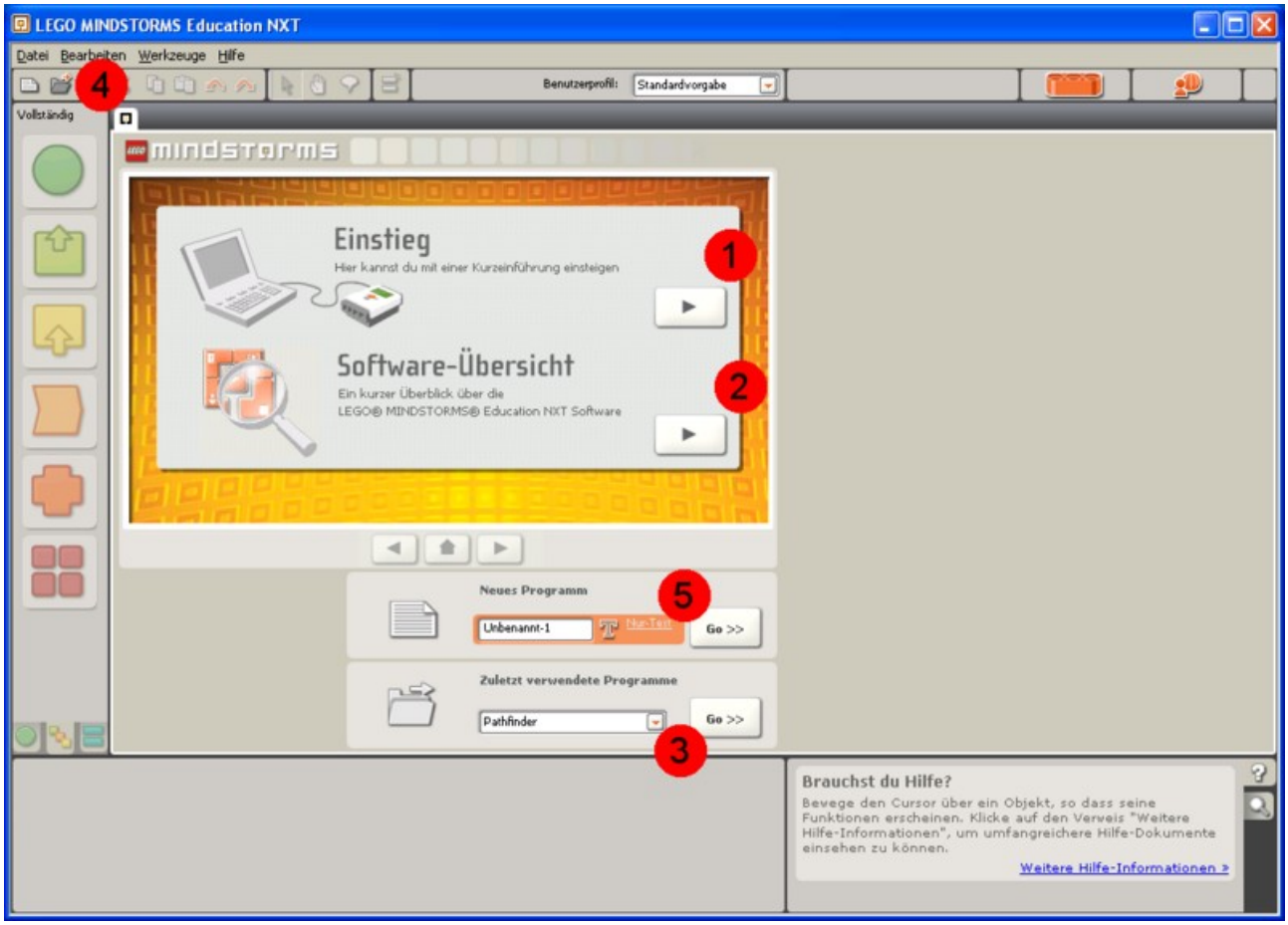
NXT-G ist eine Programmierumgebung für LEGO-Mindstorms, die trotz einiger kleiner Schwächen, einen leichten Einstieg in die Programmierung erlaubt. Mit dieser Software können auch Kinder programmieren, sowie sie Lesen und Schreiben können. Eigentlich alle Strukturen eines *richtigen Programmes* tauchen hier bereits auf und können recht intuitiv eingesetzt werden.

Die Oberfläche

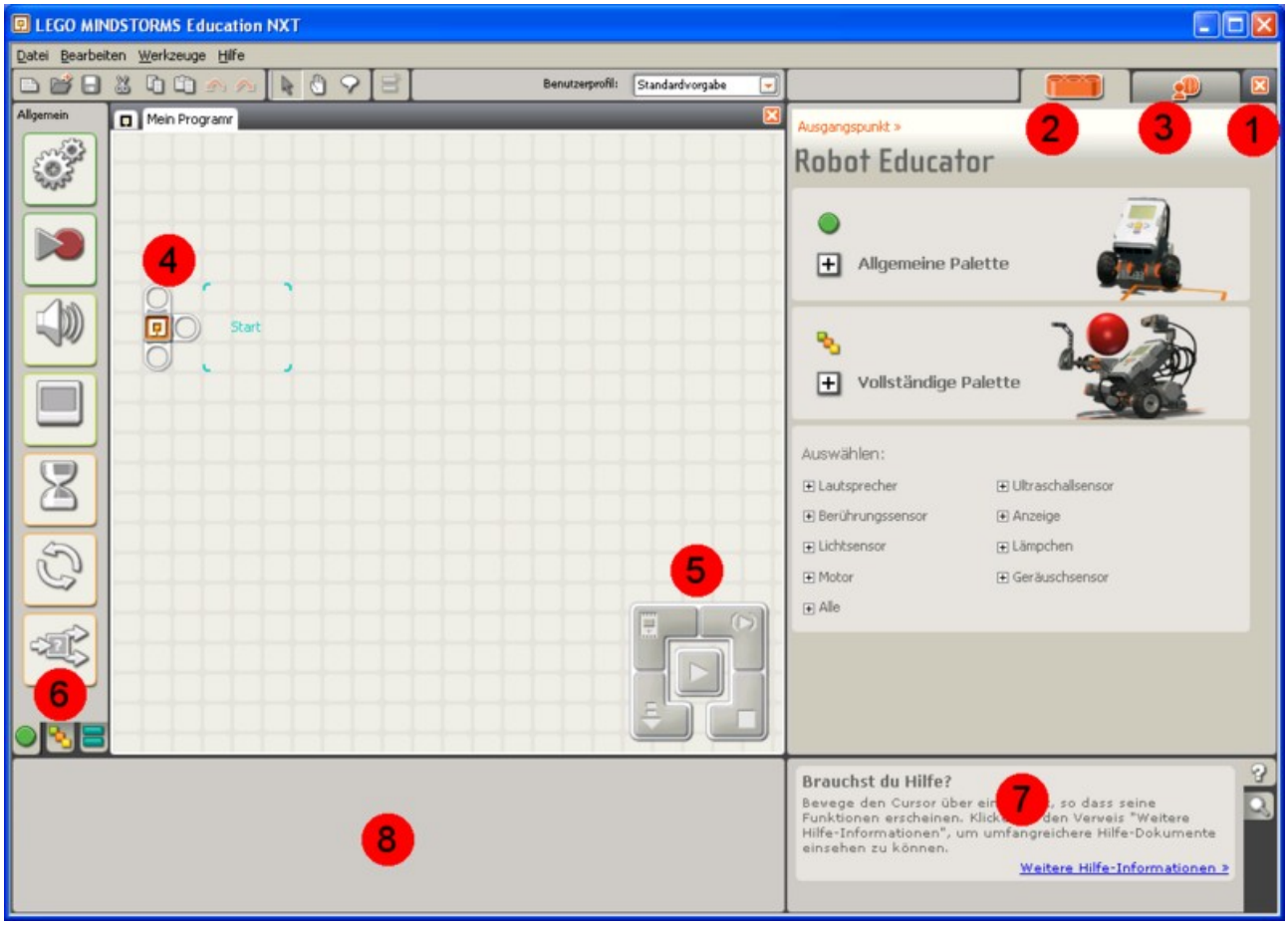
Bei Start der Education-Version des Programmes ergibt sich folgendes Bild:



Unter *Einstieg* (1) und *Software-Übersicht* (2) kann man sich eine kurze Einführung in den Umgang mit dem System geben lassen. Weiter, zu eigentlichen Programmieroberfläche, kommt man nur, wenn man entweder ein vorhandenes Programm öffnet oder ein *Neues Programm* beginnt. Ein vorhandenes Programm öffnen kann man über *Zuletzt verwendete Programme* (3), hier öffnet sich eine Liste mit den Programmen, die man zuletzt verwendet hat. Will man ein Programm öffnen, welches man auf diesem Rechner noch nicht bearbeitet hat, so muss man über *Datei -> öffnen* im Menü gehen oder das Icon *Datei öffnen* (4) in der Iconleiste.

In der Regel wird man ein neues Programm erstellen wollen, dazu gibt man den Namen des Programmes in das Feld *Neues Programm* (5) ein und klickt dann auf den Button *Go* direkt dahinter.

Es öffnet sich nun die eigentliche Arbeitsfläche.



Ein großer Teil der Bildschirmfläche wird von dem *Robot Educator*, dem integrierten Lernsystem belegt. Hier finden sich sehr viele Anleitungen, zum großen Teil mit Video-Unterstützung. Der Robot Educator erlaubt es das System auf eigene Faust kennen zu lernen. für die praktische Arbeit kann man den Platz vergrößern, indem man den Educator schließt, dazu dient das *Kreuz* (1) direkt oberhalb des Educator-Fensters. Will man den Educator wieder aktivieren, so langt ein Klick auf den *Dreier-Stein* (2) in der gleichen Zeile.

Zwischen dem Kreuz und dem Stein befindet sich noch ein kleiner Knopf (3) mit dem Namen *Mein Portal*. Klick man auf diesen Knopf, so wird der Educator ersetzt durch einen Bereich mit zwei Links auf Lego-Portale im Internet. Diese Links kann man natürlich nur nutzen, wenn der Rechner eine Internet-Verbindung besitzt.

Im eigentlichen Arbeitsbereich befinden sich zwei ganz wichtige Elemente, der Ablauf-Träger mit dem Startpunkt (4) und der Controller (5). Nur Programmblöcke, die mit dem Ablaufträger verbunden sind, können auf den Roboter übertragen werden. Ein Block, der keine Verbindung mit dem Ablaufträger hat, wird deshalb auch wesentlich schwächer dargestellt. Über den Controller kann das Programm auf den Roboter übertragen werden.

In der linken Spalte (6) des Arbeitsbereiches findet sich die Palette bzw. die drei Paletten. Hier sind alle Blöcke zu finden, die man für eigene Programme benutzen kann.

Rechts unten (7) befindet sich der *Hilfe-Bereich*. Der Text in diesem Bereich ist immer auf das aktuell markierte Element bezogen, wodurch sich die Hilfe an die aktuelle Arbeitssituation anpasst. Bei größeren Projekten kann es sinnvoll sein auf die Hilfe zu verzichten, und stattdessen eine Gesamtübersicht über das Projektfenster zu bekommen. Dazu klickt man rechts neben dem Hilfefenster auf die kleine Lupe.

Das Arbeitsfenster lässt sich nicht zoomen**,** also in der Darstellungsgröße verändern. Man kann den sichtbaren Bereich lediglich verschieben, man nennt das *Scrollen.* Zum Scrollen benutzt man im einfachsten Fall die Cursor-Tasten, die mit den Pfeilen drauf, auf der Tastatur.

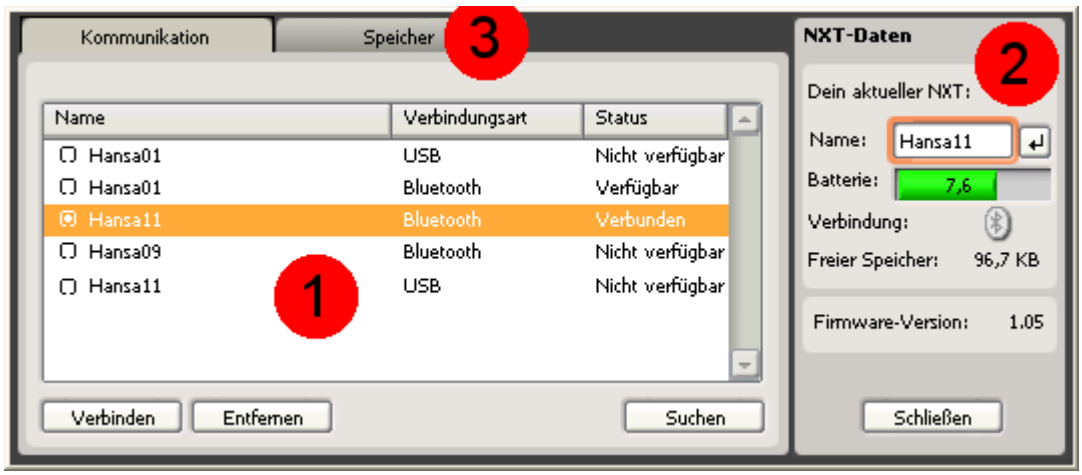
Links neben der Hilfe findet sich ein momentan vollkommen leerer Bereich (8). Immer dann, wenn ein Block aktiviert ist, finden sich hier die zugehörigen Einstell-Möglichkeiten.

Der Controller

Der Controller dient zur Kommunikation mit dem Roboter bzw. dessen zentralem Stein, der sogenannten *Brick*. Hierüber kann man z.B. Programme übertragen und starten.

Der Controller besteht aus vier Knöpfen, die ein Quadrat bilden und einem mittleren Knopf. Der mittlere Knopf dient dazu, ein Programm auf den Roboter zu übertragen und gleich zu starten. Mit dem Knopf links unten kann man ein Programm übertragen ohne es gleich zu starten, das ist besonders dann sinnvoll, wenn der Roboter über das USB-Kabel verbunden ist, sich also nicht frei bewegen kann. Der Knopf rechts unten dient zum Stoppen eines laufenden Programmes, setzt aber natürlich eine Verbindung zum Roboter voraus. Den Knopf rechts oben wird man seltener benötigen, er dient dazu nur markierte Teile eines Programmes an den Roboter zu übertragen und nicht das gesamte Programm.

Starten wird man in der Regel mit dem Knopf links oben, der *NXT-Fenster* benannt ist. Es öffnet sich ein kleines Fenster:



Im mittleren Bereich des Fensters (1) sind alle Roboter zu sehen, mit denen man in der letzten Zeit einmal eine Verbindung aufgebaut hatte. Die Roboter sind eventuell sogar doppelt aufgeführt, einmal mit einer Verbindung über das USB-Kabel und einmal bei einer Verbindung über die Funkverbindung *Bluetooth.* Falls die Software den Roboter automatisch finden kann, dazu darf nur ein einziges Gerät verfügbar sein, wird die Verbindung eventuell automatisch aufgebaut, ansonsten muss man es aus der Liste auswählen und auf *Verbinden* klicken.

Sollte der gewünschte Roboter nicht in der Liste auftauchen, so klickt man auf *Suchen.* Falls über Bluetooth verbunden wird und mehrere Roboter im Raum sind kann es etwas dauern, bis der gewünschte Roboter in der Liste auftaucht. Manchmal muss man die Suche auch mehrfach durchführen. Sollte auch das nicht helfen, so muss man den Roboter ausschalten und nach kurzer Zeit wieder einschalten, wobei Bluetooth natürlich auch auf dem Roboter aktiviert sein muss. Verbindungen über das USB-Kabel sind wesentlich problemloser als solche per Bluetooth.

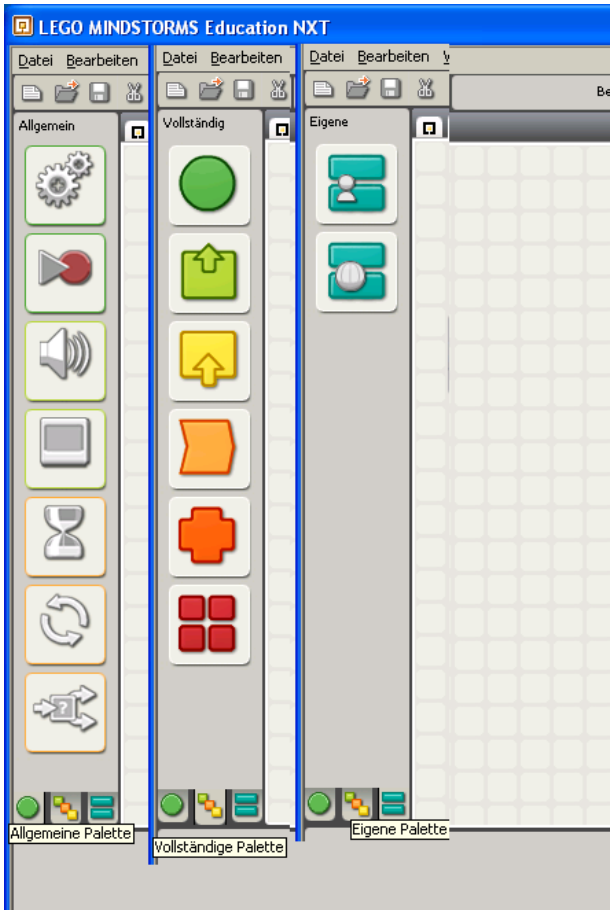
Sowie die Verbindung zu dem Roboter aufgebaut wurde, verändert sich der Daten-Bereich (2). Hier ist jetzt der Name des Roboters zu lesen und zu sehen, wie voll die Batterien noch sind. Sowie diese Änderungen erfolgt sind, kann man das Fenster über den Knopf *Schließen* wieder beseitigen.

Beim Übertragen von Programmen kann es gelegentlich zu Problemen kommen, wenn der Speicher des Roboters voll ist. Dann muss man Klangdateien oder Programme auf dem Roboter löschen, dazu dient der Reiter *Speicher* (3) im gerade beschriebenen Fenster. Hier wählt man Dateien aus, die man nicht mehr dringend benötigt, und klickt dann auf *Löschen*, um wieder freien Speicherplatz zu bekommen.

die Paletten

Bisher haben wir alle Blöcke aus der allgemeinen Palette bezogen. Die Software verfügt über insgesamt drei Paletten, von denen immer nur eine zu sehen ist:

- Allgemeine Palette
- Vollständige Palette
- Eigene Palette



Die *Allgemeine Palette* ist gut für den Einstieg, hier stehen die wichtigsten Blocks zur Verfügung. Wenn man etwas mehr Erfahrungen gewonnen hat, dann möchte man alle Block zur Verfügung haben und wird dann nur noch die *Vollständige Palette* nutzen wollen. Die Software kann man um eigene Blocks erweitern, diese eigenen Blocks lassen sich dann über die *Eigene Palette* verwalten.

Hinter jedem Icon in der *Vollständigen* Palette verbirgt sich eine ganze Leiste mit Blocks. Das sind der Reihe nach:

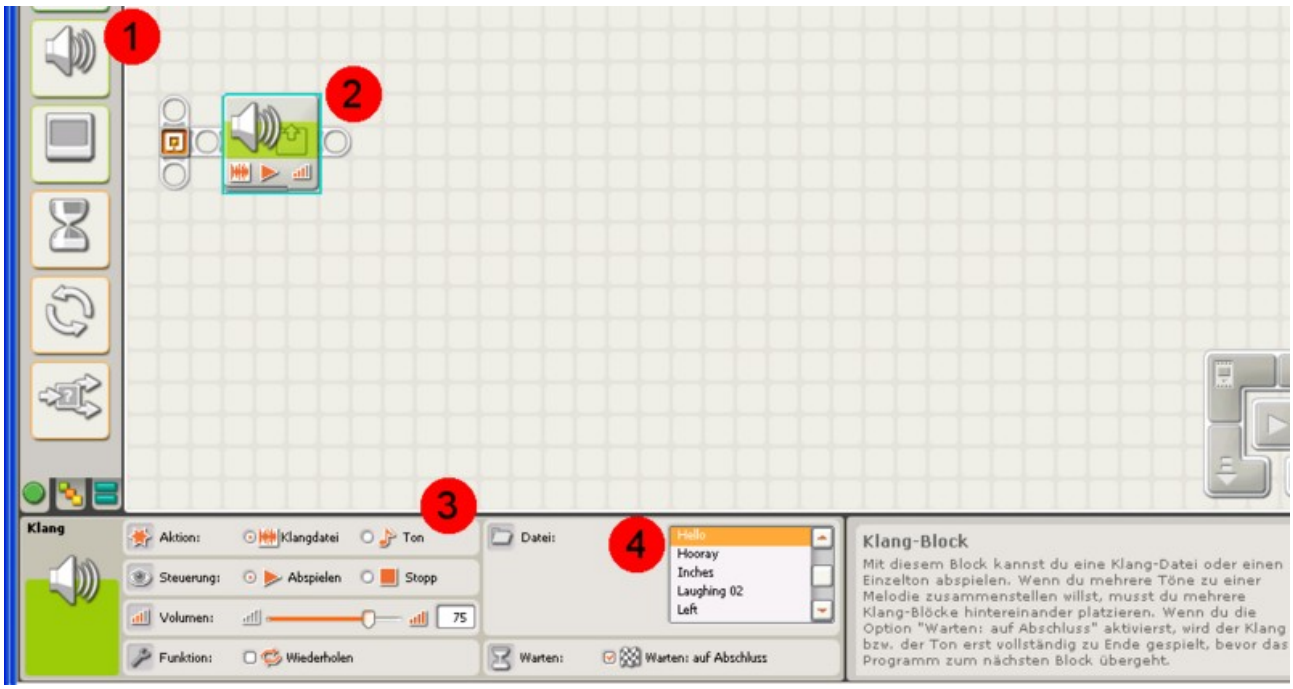
- Allgemein, entspricht der Allgemeinen Palette
- Aktion
- Sensor
- Ablauf
- Daten
- Großer Funktionsumfang

einfache Programme in NXT-G

Die Erstellung eines Programmes ist recht einfach, man klickt einfach einen der Blöcke aus der Palette mit der Maus an, führt die Maus mit dem Block über den Ablaufträger und klickt einmal auf die Oberfläche, der Block sollte dann mit dem Ablaufträger verbunden sein. Alle Blöcke, die mit dem Ablaufträger verbunden sind, bilden gemeinsam das Programm.

Das erste Programm

Für das erste Programm ziehen wir aus der allgemeinen Palette einen *Klang-Block*, (1) an den Ablaufträger (2)

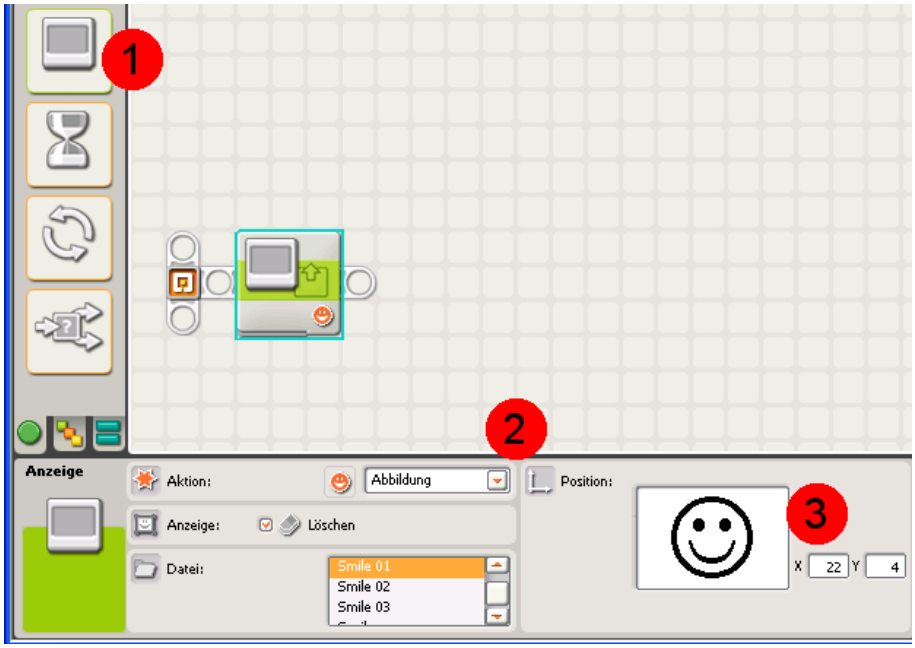


Im bisher leeren Bereich unten auf der Seite tauchen nun die Einstell-Möglichkeiten (3) für den Block auf. Bei einem Klang-Block hat man die Auswahl zwischen Klangdateien und Tönen. Im einfachsten Fall belässt man es erst einmal bei allen Vorgaben und wählt nur unter *Datei* den Eintrag *Hallo* aus (4). Sowie man das gemacht hat, hört man die Klangdatei, aber auf dem Computer.

Überträgt man das Programm jetzt auf den Computer und startet es, dazu dienen die Knöpfe im Controller, so sollte die Datei auf dem Roboter zu hören sein.

Das zweite Programm

Bevor wir uns mit den Motoren beschäftigen ein weiteres kleines Programm und zwar wollen wir uns mit dem Anzeige-Block beschäftigen. Also zuerst den Klang-Block löschen, oder mittels *Datei -> Neu* einen neuen Arbeitsberich erstellen. In diesen Bereich ziehen wir nun einen *Anzeige-Block* (1) aus der Palette. Im unteren Bereich (2) sind jetzt die möglichen Einstellungen für diese Art von Block zu sehen.

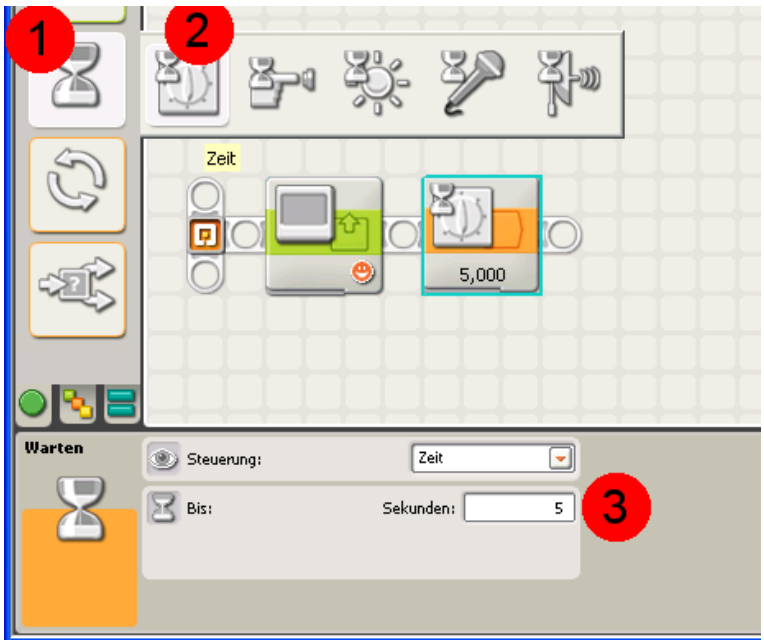


Man hat hier u.a. die Auswahl zwischen Abbildungen und eigenen Texten. Rechts in diesem Bereich (3) bekommt man eine Vorschau dessen, was man auf dem Display des Roboters sehen würde.

Überträgt man nun dieses Programm auf den Roboter, so muss man sich auf eine Enttäuschung gefasst machen. Es ist auf dem Display eigentlich nichts zu sehen. Das hängt damit zusammen, dass ein normaler Block keine Zeit verbraucht, und anschließend sofort der nächste Block aktiv wird, oder wie hier beim letzten Block das Programm beendet wird. Das Bild wird also auf das Display gebracht und sofort danach ist das Programm zuende, wodurch das Display wider gelöscht wird.

Man muss das Programm also etwas warten lassen, damit das Bild zu sehen ist.

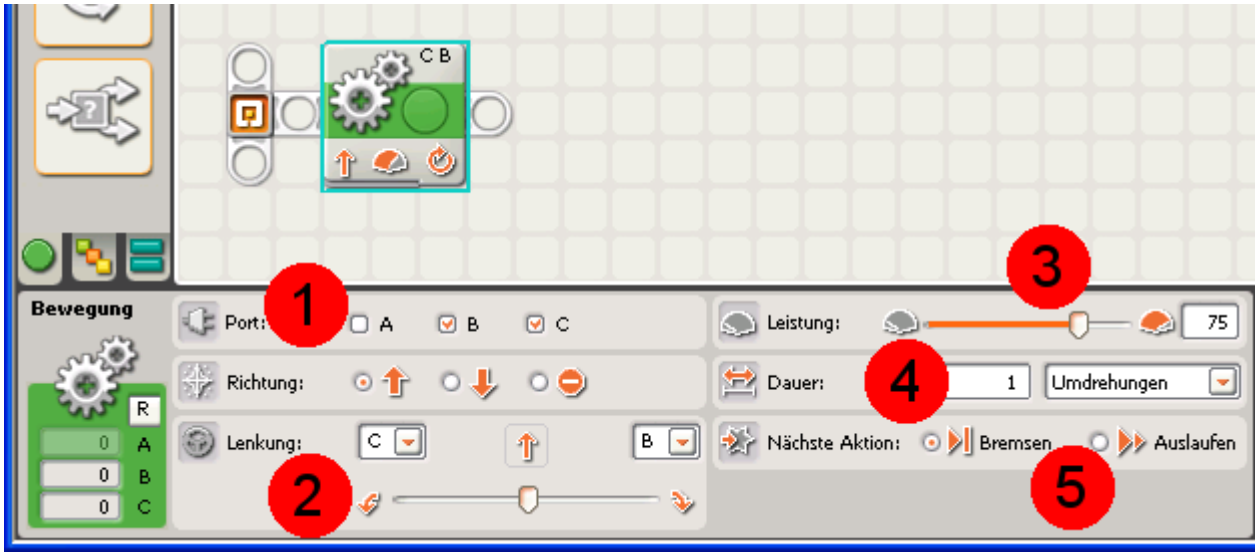
Für diese Aufgabe gibt es in der Palette mehrere Warteblöcke, die sich alle hinter dem Icon mit der Sanduhr (1) verbergen.



Geht man mit der Maus über dieses Icon, so erscheint eine schwebende Leiste mit 5 weiteren Icons. Das erste dieses Icons steht für den *Zeitsensor,* (2) den wir hier einsetzen wollen. Normalerweise ist dieser Sensor auf eine Wartezeit von 1 Sekunde eingestellt, im Eigenschaftsbereich kann man den Wert z.B. auf 5 Sekunden (3) stellen.

Motoren an

Mit dem nächsten Programm wollen wir erreichen, dass sich der Roboter in Bewegung setzt. Dazu dient der Bewegungs-Block.



Für diesen Block gibt es recht viele Einstellmöglichkeiten. Ganz wichtig ist die Auswahl der Motoren. Bei Fahrzeugen, wie dem Tribot benutzt man in der Regel zwei Motoren für den Antrieb, üblicherweise sind diese Motoren mit den Ausgängen B und C der Brick verbunden. Hier müssen unter *Port* (1) die richtigen Ausgänge gewählt sein. Unter *Richtung* kann man die Drehrichtung der Motoren auswählen bzw. die Motoren stoppen.

Unter *Lenkung* (2) kann man einstellen, ob beide Motoren sich gleichartig bewegen sollen, oder unterschiedlich. Bewegen sich die Motoren unterschiedlich, so fährt der Roboter einen Kreis. Zieht man den Regler ganz an einen seitlichen Anschlag, so bewegt sich der entsprechende Motor sogar in der Gegenrichtung, wodurch sich der Roboter nahezu auf der Stelle dreht.

Der Regler *Leistung* (3) erlaubt es festzulegen, mit welchem Anteil an der maximalen Leistung die Motoren drehen. Je höher dieser Wert ist, desto schneller fährt der Roboter. Bei höherer Leistung leidet aber die Genauigkeit erheblich, weil dann z.B. die Antriebsräder leicht durchdrehen.

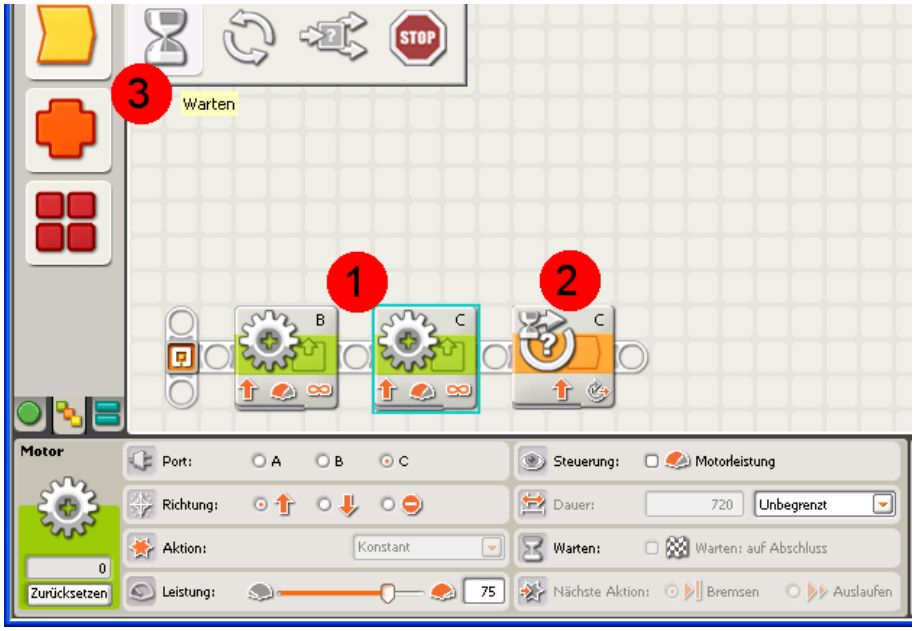
Unter Dauer (4) kann man festlegen, wie die Bewegung der Motoren begrenz wird. Hier hat man die Wahl zwischen:

- *Unbegrenzt,* das ist in der Regel aber kürzer als erwartet, weil wie im zweiten Programm der Block selber keine Zeit verbraucht und dann das Programm beendet ist.
- *Gradzahl,* das ist eine Anzahl von Grad für die Umdrehung der Motoren. 360 Grad würden einer Motorenumdrehung entsprechen.
- *Umdrehungen,* gibt an, wie viele Umdrehungen Motor bzw. Rad machen sollen.
- *Sekunden* gibt die Zeit an, die die Motoren laufen sollen.

Unter *Nächste Aktion* (5) legt man fest, ob die Motoren am Ende der Bewegung gebremst werden sollen oder auslaufen dürfen. Wenn eine exakte Bewegung erwartet wird, dann sollte man die Motoren bremsen.

Bewegungsblock oder Motorblock

Der im letzten Beispiel benutzte Bewegungs-Block ist ein recht komplexes Gebilde. Besonders wenn Lenkbewegungen mit ins Spiel kommen ist es nicht ganz einfach zu kontrollieren, wie sich die einzelnen Motoren wirklich bewegen sollen. Transparenter wird diese, wenn man stattdessen Motorblöcke benutzt, dafür wird das Programm dann etwas größer.



Die beiden Motorblöcke (1) stammen aus der *Vollständigen Palette* und dort aus der Leiste Aktion. Ein Motorblock kann immer nur einen Block zur Zeit ansteuern, daher sind zwei Motorblöcke notwendig. In den Eigenschaften stellt man unter *Dauer* für beide Motoren *Unbegrenzt* ein.

Damit die Bewegung kontrollierbar wird, muss nun noch ein Warten-Block (2) hinzukommen. Dieser Block findet sich in der *Vollständigen Palette* unter Warten. Dort gibt es kein Icon direkt für den Drehsensor, wohl aber ein Warten-Icon (3). Bei diesem Icon muss man unter Sensor noch den Drehsensor auswählen.

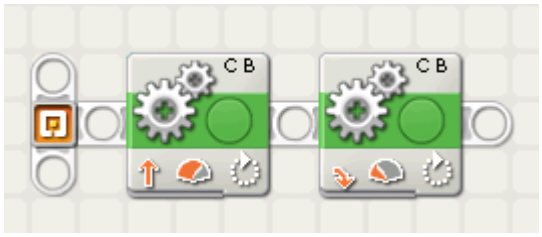
Der Drehsensor ist eine geniale Zusatzfunktion der Motoren. Jeder Motor verfügt über einen eingebauten Drehsensor, der über den normalen Motoranschluss abgefragt wird. Es wird also kein extra Anschluss benötigt. Auch wenn die Arbeit mit den Motorblöcken auf den ersten Blick aufwändiger scheint, für eine genauere Kontrolle, z.B. bei Drehungen, sind sie unverzichtbar.

Immer wieder - Entscheidungen

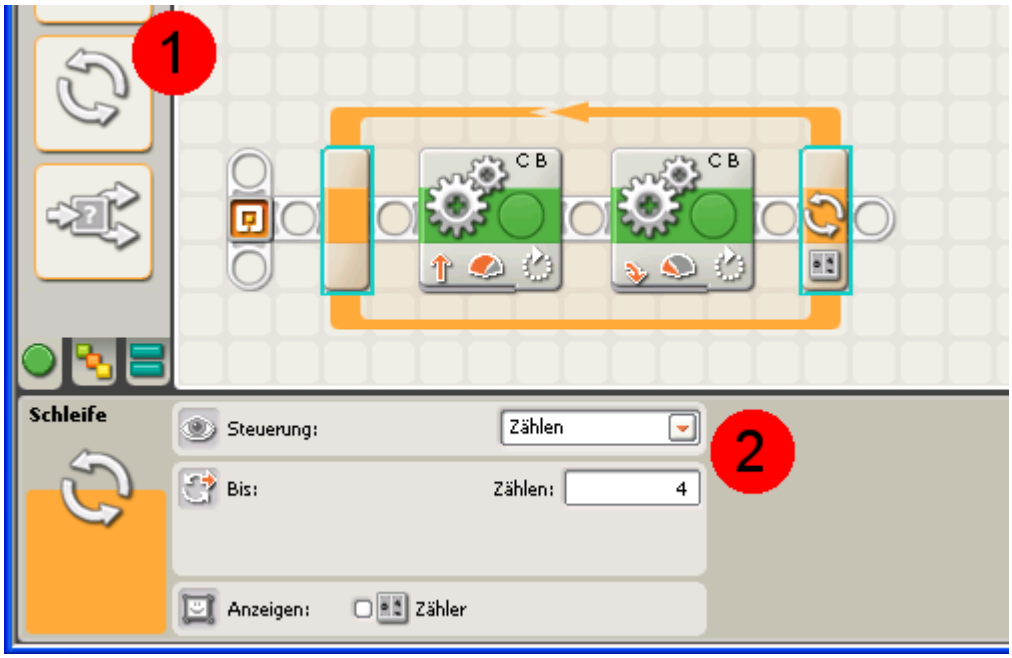
Bei den bisherigen Beispielen waren die Programm-Blöcke alle direkt hintereinander auf dem Ablaufträger zu finden. Für die vile Probleme benötigt man etwas komplexere Abläufe.

wir fahren ein Quadrat

Im nächsten Schritt soll der Roboter ein Quadrat abfahren und möglichst nah am Startpunkt wieder ankommen. Er muss also eine Strecke vorwärts fahren und dann die Lenkung einschlagen und ein Stück Kurve fahren. Das Programm könnte dann folgendermaßen aussehen:



Wenn man jetzt die beiden Blöcke dreimal kopieren würde, dann käme ein Quadrat zustande. Statt die Blöcke dreimal zu kopieren, ist es aber sinnvoller eine Wiederholung einzusetzen. Dazu dient der Block *Schleife* (1):



Es handelt sich um den vorletzten Block der allgemeinen Palette. Die beiden Bewegungs-Blöcke müssen dann markiert und mit der Maus in den Schleifen-Block hineingeschoben werden. Die Voreinstellung für den Schleifen-Block besteht in einer unendlichen Wiederholung. Für unser Bespiel stellt man unter *Steuerung* (2) auf *Zählen* und gibt dann im Feld *Zählen* die Zahl 4 ein.

Es ist leider nicht ganz leicht die Bewegungsblöcke so einzustellen, dass der Roboter exakt ein Quadrat fährt und dann wieder im Ausgangspunkt ankommt. Oft rutschen die Räder etwas durch oder sind nicht ganz exakt im gleichen Abstand auf der Achse angebracht.

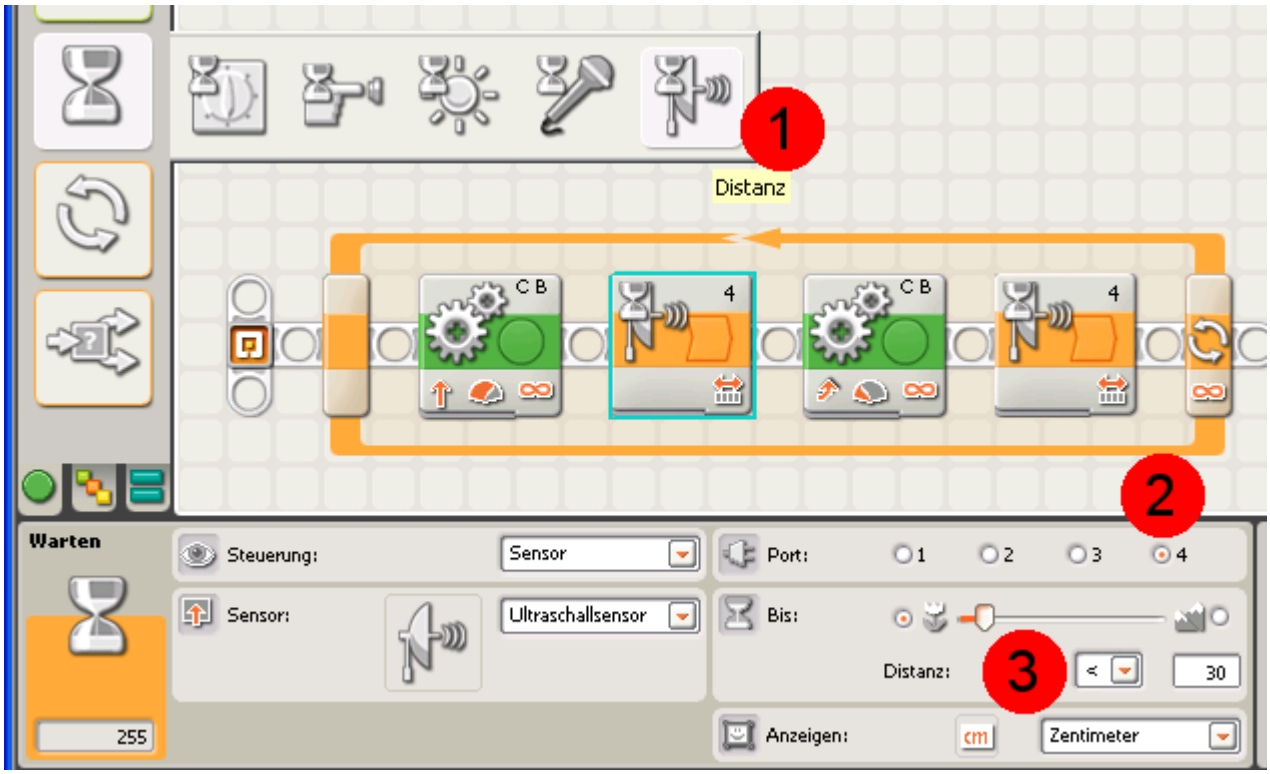
wir weichen Hindernissen aus

Das nächste Programm benutzt auch die mitgelieferten Sensoren. Zum Roboter gehören insgesamt 5 Sensoren:

- zwei Berührungssensoren
- ein Ultraschallsensor zur Abstandmessung
- ein Lichtsensor
- ein Geräuschsensor

Für den Anschluss der Sensoren verfügt der Roboter über vier Anschlüsse, die von 1 bis 4 durchnummeriert sind. Recht interessant ist der Ultraschallsensor, der es dem Roboter erlaubt Hindernisse zu erkennen. Mit dem folgenden Programm soll sich der Roboter vorwärts bewegen, bis er ein Hindernis erkennt. Wenn ein Hindernis erkannt ist, dann soll der Roboter in einer kleinen Kurve rückwärts fahren und dann erneut versuchen vorwärts zu fahren.

Wir brauchen also zuerst einen Schleifen-Block, der auf der Voreinstellung für unendlich viele Wiederholungen verbleibt. Das Programm kann dann folgendermaßen aussehen:

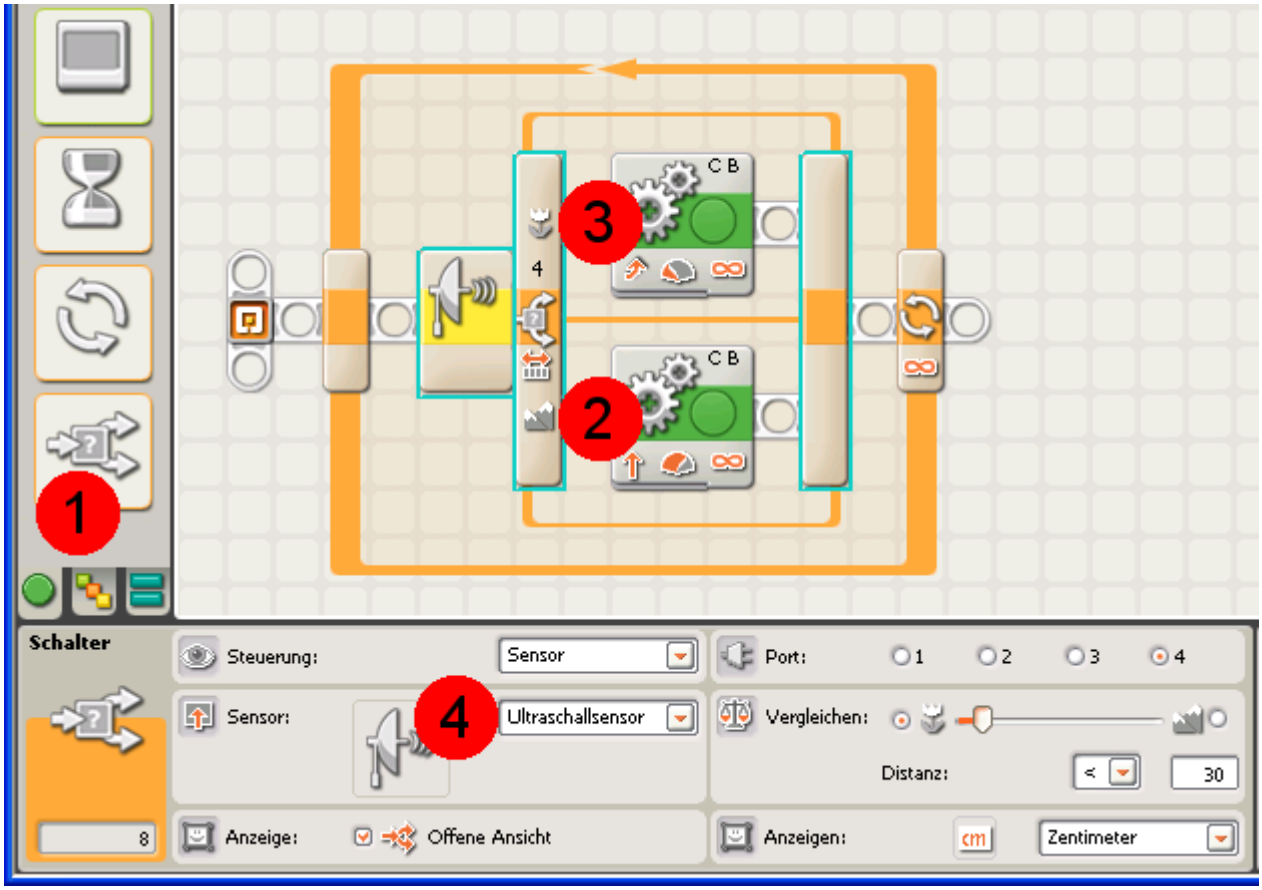


Innerhalb der Schleife taucht zuerst der bereits bekannte Bewegungs-Block auf. Nur dass jetzt hier unter *Dauer Unbegrenzt* aktiviert ist, die Bewegungsdauer soll ja aber den Sensor geregelt werden. Als nächster Block folgt dann der *Warten* Block *Distanz* (1). Der Ultraschallsensor ist mit dem Port Nummer 4 verbunden (2), wie man auch an an dem Block im Programm erkennen kann. Der Warten Block wartet solange, bis der Abstand zu einem Hindernis kleiner wird, als 30cm (3). Solange bis das eintritt, bewegt sich der Roboter vorwärts.

Kommt dem Roboter ein Hindernis in die Quere, so ist das Warten beendet und der Block nach dem Warteblock kommt zum Zug. Hier ist der Bewegungs-Block so eingestellt, dass unbegrenzt eine rückwärts gerichtete Drehbewegung stattfindet. Der Roboter also in einem Bogen rückwärts fährt, bis der nächste Warteblock feststellt, dass vorne wieder frei ist. Anschließend ist der Durchgang beendet und durch die Schleife beginnt der Ablauf wieder von Vorne. Die beiden Warteblöcke unterscheiden sich lediglich in der Art des Vergleiches bei *Distanz*. Im ersten Warteblock steht hier "< 30", beim zweiten Warteblock "> 30".

wir weichen Hindernissen aus 2

Für das eben beschriebene Problem gibt es eine Lösung, die statt mit zwei Warteblöcken mit einem Schalter arbeitet. Schalter sind die untersten Blöcke in der allgemeinen Palette.



Zuerst brauchen wir wieder einen Schleifenblock. In diesen Block hinein ziehen wir einen Schalter-Block. Im Schalter-Block befinden sich zwei Ablaufträger, die wir jeweils mit dem passenden Motorblock versehen. Für die Motorblöcke benutzen wir die gleichen Einstellungen wie beim vorigen Beispiel. In den Ablaufträger hinter dem Bergsymbol (2) für hohe Abstände kommt der Block für die Vorwärtsbewegung, auf den Ablaufträger hinter dem Icon mit der Blume (3) für geringe Entfernungen kommt der Block für die Rückwärtskurve.

Bei den Eigenschaften für den Schalter muss man den richtigen Sensor auswählen, hier den Ultraschall-Sensor (4). Bei der Distanz geben wir wieder 30 an, die Frage der Richtung ist hier nicht ganz so wichtig, da diese über die Icons sehr deutlich angezeigt wird.

Linienverfolger

Eines der klassischen Probleme für die Mindstorm-Roboter ist die Linienverfolgung. Man zeichnet dazu eine breite schwarze Linie auf ein helles Blatt Papier und versucht den Roboter dazu zu bringen dieser Linie zu folgen. Mit zwei Lichtsensoren wäre das überhaupt kein Problem:

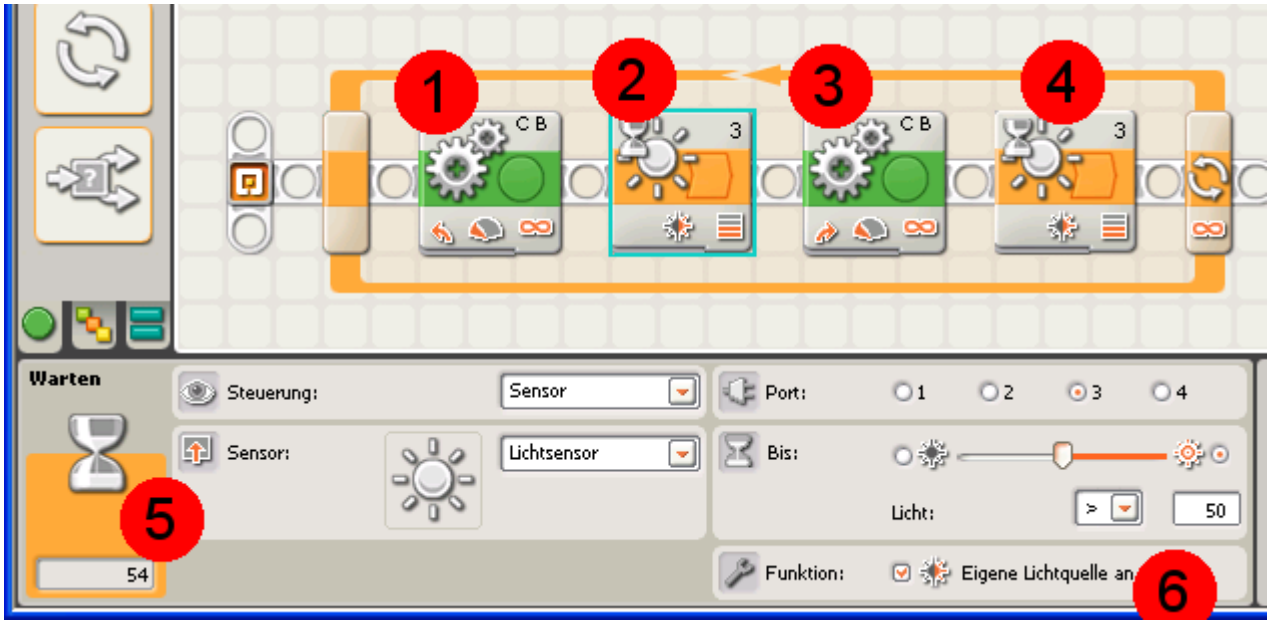
- solange beide Sensoren Schwarz sehen kann der Roboter vorwärts fahren
- wenn der linke Sensor Weiß sieht und der rechte Schwarz, dann muss der Roboter nach Rechts schwenken
- wenn der linke Sensor Schwarz sieht und der rechte Weiß, dann muss der Roboter nach Links schwenken
- wenn beide Sensoren Weiß sehen, dann muss der Roboter anhalten.

Leider gehört zu den Mindstorm-Robotern nur ein einziger Lichtsensor. Damit ist die Steuerung etwas schwieriger, wenn der Sensor Weiß sieht, dann weiß der Roboter leider nicht, auf welcher Seite die Linie verlassen wurde. Ein Ausweg besteht darin von vornherein einen Bogen zu fahren, z.B. nach Links. Wenn der Sensor dann Weiß sieht, muss der Roboter einen Rechtsbogen fahren, bis er wieder Schwarz sieht, worauf er dann erneut den Linksbogen fährt.

Bei diesem Verfahren folgt der Roboter nicht der Linie, sondern der Kante und das auf einem Zickzack-Kurs.

Linienverfolger mit zwei Warteblöcken

Im Prinzip kann ein einfacher Linienverfolger so arbeiten, wie das erste Programm mit dem Ultraschallsensor. Der Lichtsensor ist in der Regel am Port 3 angeschlossen.



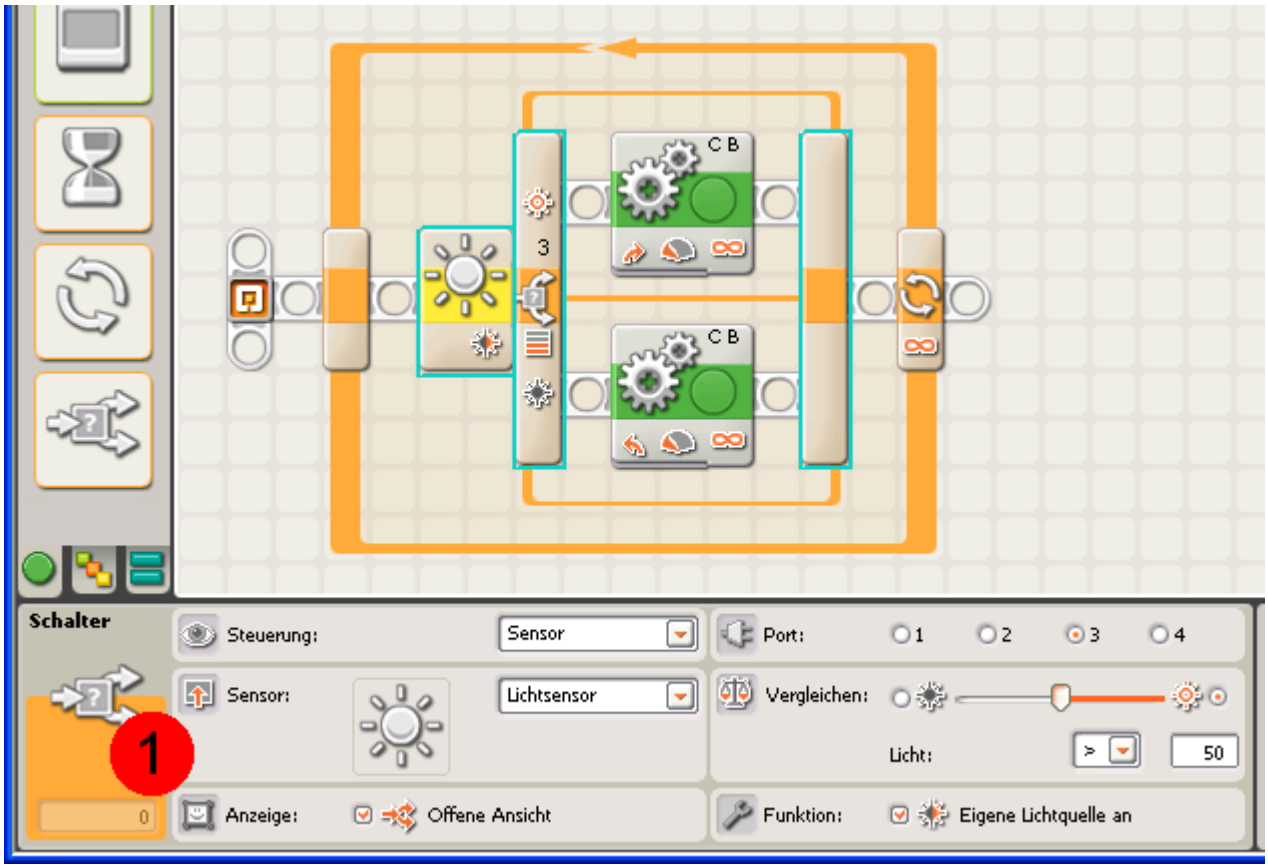
Zuerst kommt der Bewegungs-Block (1) mit der Linkskurve, dann der Warteblock (2) für einen hellen Wert. Darauf der Bewegungs-Block (3) mit der Rechtskurve und anschließendem Warteblock (4) für einen dunklen Wert.

Die Zahl, die als Grenze zwischen Hell und Dunkel zu nehmen ist, hängt von den Randbedingungen ab, wie z.B. der Helligkeit im Raum. Man muss sie ausprobieren, dazu klickt man einen der Warteblöcke an und setzt den eingeschalteten Roboter auf den Untergrund. Links unten im Fenster (5) kann man den aktuell gültigen Wert ablesen.

Es ist sinnvoll für dieses Programm die eigene Lichtquelle des Sensors zu aktivieren (6), dann ist der Roboter etwas unabhängiger von der Raumhelligkeit.

Linienverfolger mit Schalter

Natürlich kann man auch dieses Problem wieder mit einem Schalter-Block angehen.



Auch hier muss man im Vorfeld einmal den Grenzwert für die Helligkeit messen oder ausprobieren. Wenn die Anzeige des Sensorwertes relativ grau ist und einen Wert von 0 anzeigt, dann besteht vermutlich keine Verbindung zum Roboter. Im einfachsten Fall überträgt man dann das Programm neu auf den Roboter und klickt den Schalter an, worauf wieder der aktuelle Messwert angezeigt werden sollte.

Variablen und Datenverbindungen in NXT-G

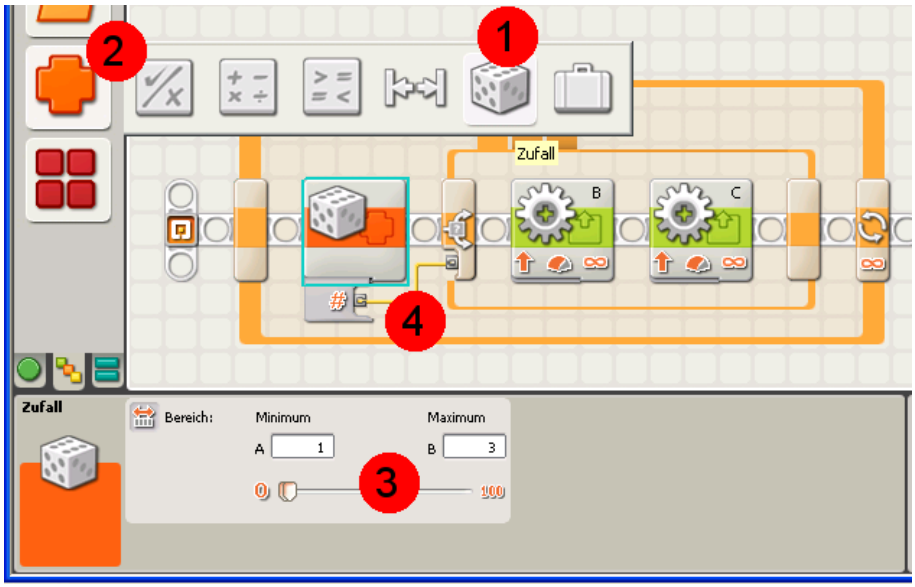
In der Vollständigen Palette gibt es eine Reihe von Blöcken, die bisher keine Rolle gespielt haben. Für die Nutzung dieser Blöcke sind in der Regel auch Datenverbindungen notwendig.

der Zufall steuert

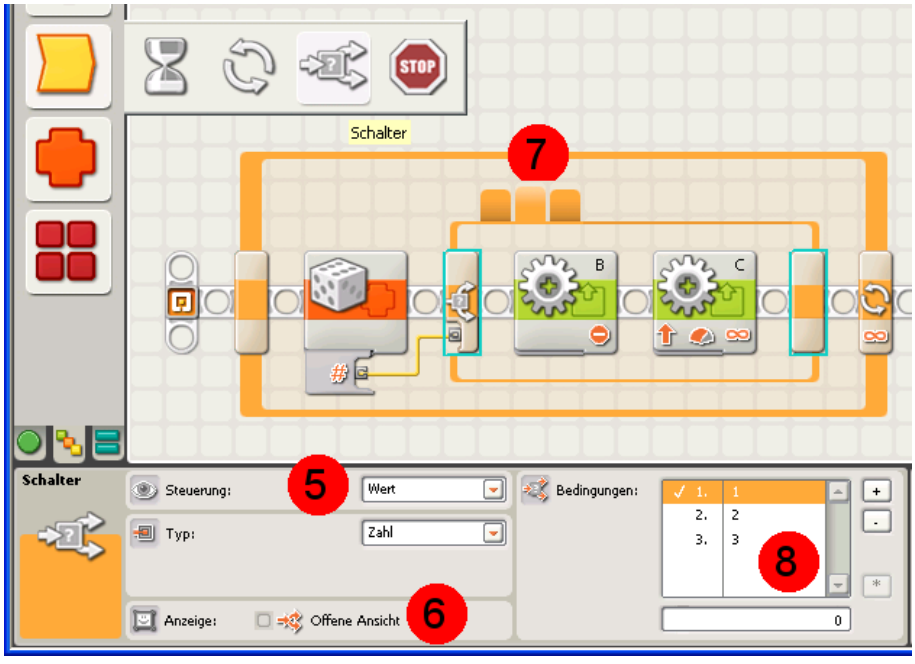
Beim folgenden Programm soll die Bewegung des Roboters vom Zufall abhängig sein. Abhängig vom Wert dieser Zahl soll sich der Roboter bewegen:

- 1, der Roboter fährt geradeaus
- 2, der Roboter fährt eine Linkskurve
- 3, der Roboter fährt eine Rechtskurve

Da die Entscheidungen natürlich ständig fallen sollen, ist das äußerste Element im Programm eine Schleife. Der nächste Block ist dann ein Block Zufall (1) aus der Leiste Daten (2).



Beim Block Zufall kann man den Bereich einstellen (3), aus dem er die Zahlen liefert. Im Beispiel ist der kleinste Wert eine 1 und der größte Wert die 3. Der Block liefert die Zufalls-Zahlen an seinem Datenausgang (4).



Bevor man nun den Ausgang des Zufalls-Blocks mit dem Schalter verbinden kann, sind ein paar Vorbereitungen notwendig. Beim Schalter muss unter *Steuerung* (5) von der Vorgabe *Sensor* auf *Wert* umgestellt werden und direkt darunter unter *Typ* von *Logiksignal* auf *Zahl*. Jetzt kann man den Datenausgang des Zufall-Blocks mit dem Dateneingang des Schalters verbinden (4).

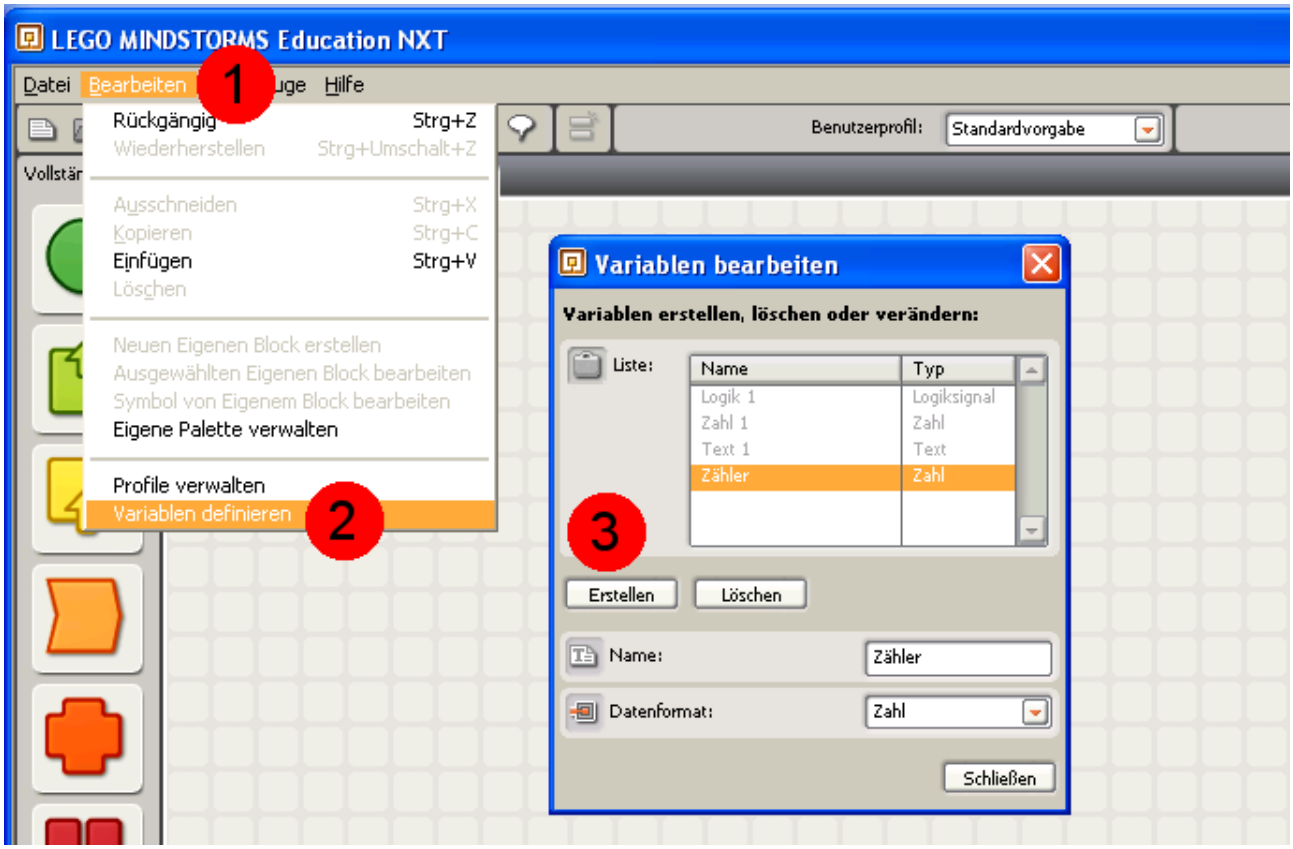
Nun kommt noch eine wichtige Änderung. Bisher hatten wir bei Schaltern intern immer zwei Ablaufträger, in unserem Beispiel brächten wir aber drei Ablaufträger. Da drei parallele Ablaufträger aber sehr viel Platz verbrauchen sieht die Software eine kompaktere Darstellung vor. Unter Anzeige nimmt man nun das Häkchen bei *Offene Ansicht* (6) weg. Die Darstellung verändert sich so, dass immer nur ein Ablaufträger sichtbar ist, welcher, das ergibt sich aus dem aktiven Reiter (7) oberhalb des Schalters. Leider sind diese Reiter nicht beschriftet, lediglich wenn man mit der Maus drüber geht, wird der zugehörige Wert angezeigt.

Unter Bedingungen (8) kann man einstellen, welche Fälle unterschieden werden müssen, bzw. wieviele Reiter am Schalter vorhanden sein müssen. Über den Button "+" kann man Fälle bzw. Reiter hinzufügen, mit dem "-" Button kann man entsprechend Reiter löschen.

Einem der Fälle ist ein Häkchen vorangestellt, das ist der Standard-Fall (Default), der immer dann eintritt, wenn keine Regel vorgesehen wäre. Würde unser Würfel also fälschlicherweise eine 4 oder eine 5 liefern, so käme dieser Fall in Aktion. Einen Standard-Fall kann man setzen, indem man die entsprechende Zeile hervorhebt und auf den Button mit dem "*" drückt.

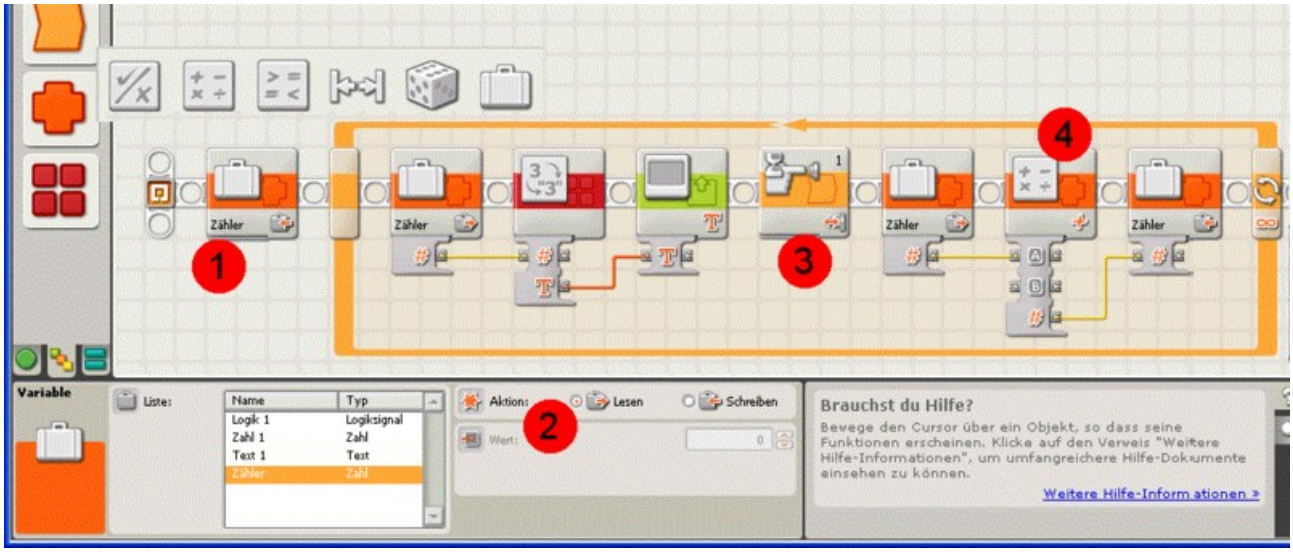
der Roboter zählt

Im ersten Schritt soll es darum gehen, dass der Roboter die Zahl der Tastendrücke am Berührungssensor zählt und den aktuellen Zählerstand jeweils auf dem Bildschirm ausgibt.



Bevor der Roboter zählen kann, müssen wir eine passende Variable einrichten. Dazu geht man über das Menü *Bearbeiten* (1) auf den Menüpunkt *Variablen definieren* (2), worauf sich ein Fenster (3) öffnet. In diesem Fenster klickt man auf *Erstellen* und gibt dann im Feld *Name* die Bezeichnung *Zähler* ein und wählt im Feld *Datenformat* den Typ *Zahl* aus. Die drei Variablen *Logik 1*, *Zahl 1* und *Text 1* sind vordefiniert und immer vorhanden. An Ende klickt man auf Schließen, worauf die Definition beendet ist.

Nun kann man die Variable in eigenen Programmen benutzen.



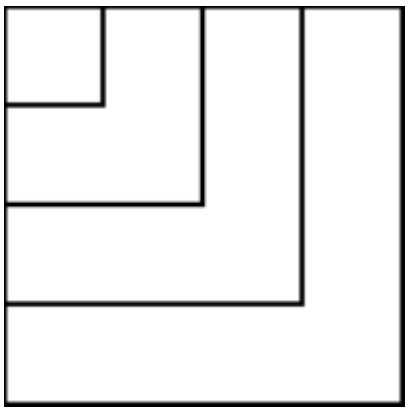
Im ersten Schritt sollte man eine Variable immer auf einen Anfangswert setzen. Dazu dient der Variablen-Block (1) vor der Schleife. Bei diesem Block wird unter *Aktion* (2) *Schreiben* ausgewählt und als *Wert 0*. Natürlich muss in der *Liste* auch die richtige Variable ausgewählt sein.

Innerhalb der Schleife folgt dann ein Variablen-Block zum Auslesen des Zählerstandes, damit dieser im Display angezeigt werden kann. Das Display kann aber Zahlen nicht direkt darstellen, daher muss die Zahl vorher in einen Text umgewandelt werden, dazu dient der Block *Konvertierung...*. Der Display-Block muss so eingestellt sein, dass er Texte darstellt und keine Abbildungen.

Nach der Darstellung im Display folgt der Warteblock für den Berührungssensor (3). Hier ist wichtig, dass man bei der *Aktion Stoß* auswählt, damit die Tastendrücke wirklich einzeln anfallen.

Nach dem Stoß muss der Zähler um 1 erhöht werden. Dazu braucht man zuerst einen Variablen-Block zum Lesen der Variablen, dann einen Mathe-Block (4) um 1 zu addieren und dann wieder einen Variablen-Block um den neuen Wert zu schreiben. In dem Mathe-Block sind zwei Zahlen-Felder und ein Feld für die Operation vorhanden. In dem Feld für die Operation wählt man Addition aus. In das Feld für den Wert B schreibt man die 1, wenn der Eingang A mit der Datenlinie vom lesenden Variablen-Block verbunden ist.

die Quadrate werden immer kleiner



Relativ einfach ist es in NXT-G ein Programm zu erstellen, mit dem der Roboter ein Quadrat abfährt. Etwas aufwändiger wird es, wenn er z.B. eine Folge von immer kleiner werdenden Quadraten abfahren soll.

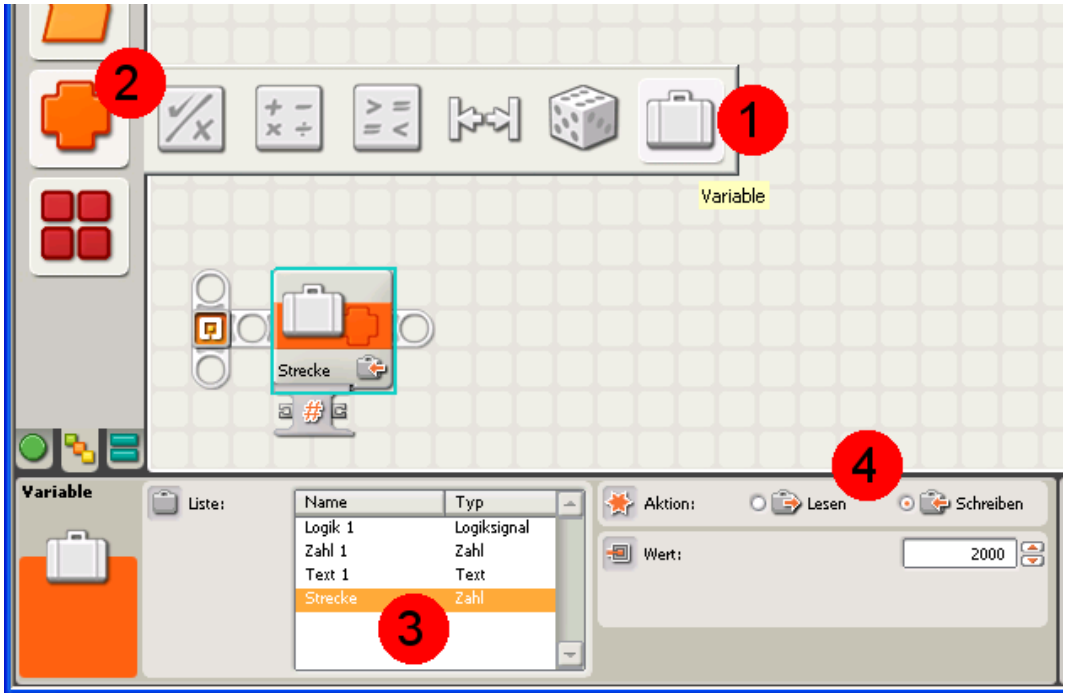
Hier muss die Länge der Fahrtstrecke flexibel abgelegt werden. Dafür verfügt das NXT-G die Möglichkeit mit Variablen zu arbeiten. Eine Variable muss über den Menüpunkt *Bearbeiten -> Variablen definieren* eingerichtet (deklariert) werden, bevor sie zur Verfügung steht.

Die drei Variablen *Logik 1*, *Zahl 1* und *Text 1* sind vordefiniert und immer vorhanden. Über einen Klick auf *Erstellen* richtet man eine neue Variable ein. Im Feld *Name* gibt man dann die Bezeichnung für die Variable an, im vorliegenden Beispiel *Strecke*. Jede der Variablen muss einem der drei Typen zugeordnet werden:

- Logiksignal (kennt nur Wahr oder Falsch)
- Zahl (für ganze Zahlen)
- Text (beliebige Zeichenketten)

Im Beispiel ist nur der Type *Zahl* sinnvoll. Klickt man abschließend auf *Schließen*, so steht die Variable für Anwendungen zur Verfügung. Man kann natürlich auch eine der vordefinierten Variablen benutzen, aber ein Programm wird besser verständlich, wenn die Variablen sprechende Namen besitzen, also solche, die ihre Aufgabe beschreiben.

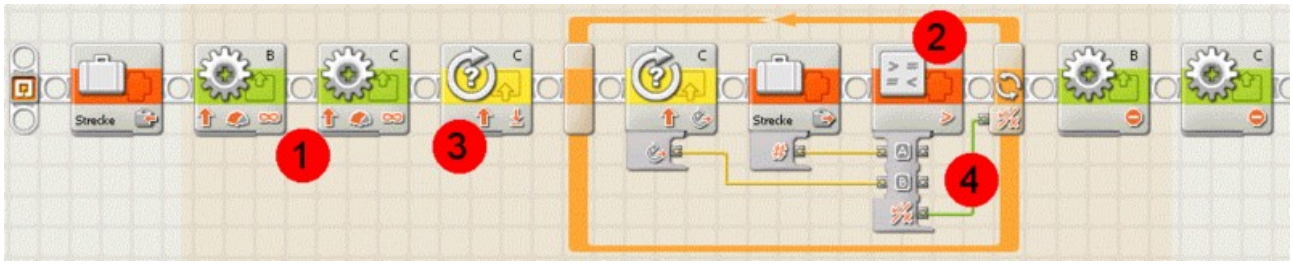
Um die Variable nun in einem Programm nutzen zu können, braucht man den Block mit dem *Koffer* (1) (*Variable*) aus der Leiste *Daten:* (2)



Bei den Eigenschaften des Blockes wählt man zuerst die richtige Variable aus der *Liste* (3) aus. Nun muss man noch festlegen, ob man die Variable *Lesen* oder *Schreiben* (4)möchte. Da man häufiger lesen als Schreiben will, ist Lesen die Voreinstellung. Am Anfang eines Programmes muss man aber die Variable zuerst einmal mit einem Wert versehen, also Schreiben. Klickt man das Feld Schreiben an, so wird darunter das Feld *Wert* aktiv. Hier trägt man nun die Strecke ein, die der Roboter anfänglich zurücklegen soll, im Beispiel 2000.

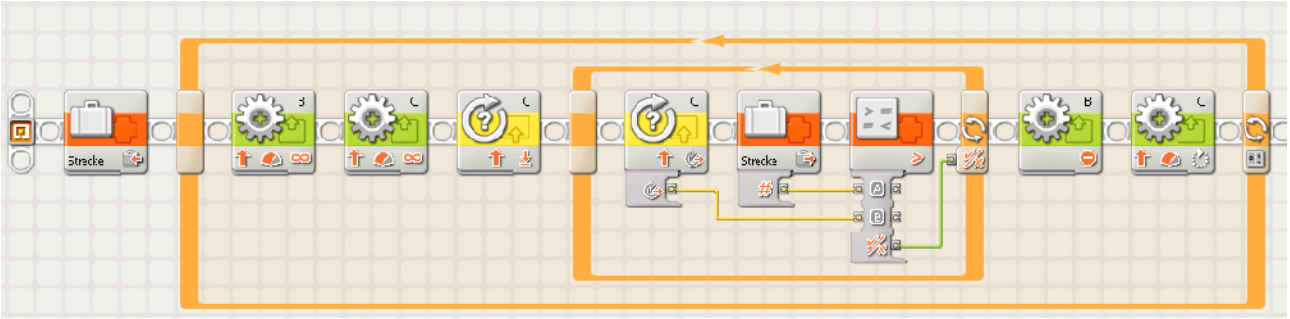
Nun müssen wir den Roboter dazu bringen die Motoren B und C (1) einzuschalten und dann solange vorwärts zu fahren, bis der Drehsensor eines der Motoren meldet, dass er mehr Schritte zurückgelegt hat, als über die Variable *Strecke* vorgegeben wurde. Dazu braucht man aus der Leiste *Daten* das Element *Vergleichen* (2). Nach dem Einschalten der Motoren wird hier der Drehsensor vorsichtshalber zurückgesetzt (3), damit er auf alle Fälle beim Zählerstand *0* beginnt.

Da der Vergleich immer wieder durchgeführt werden muss, kommt dieser Teil in eine Schleife. Es ergibt sich dann folgendes Programm:

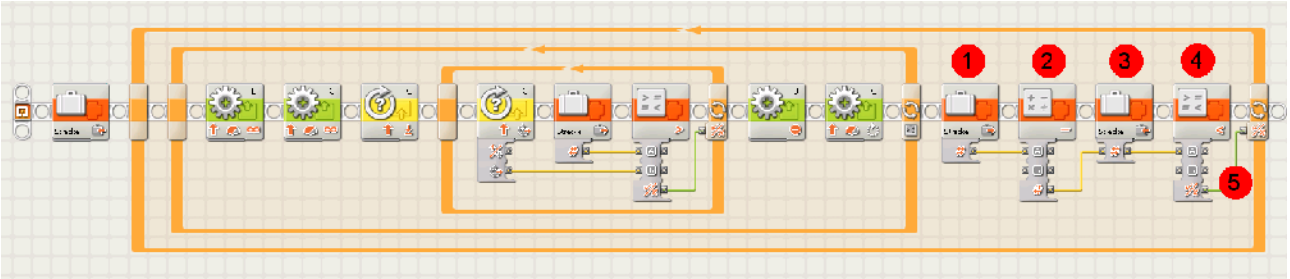


Das Ergebnis des Vergleichs soll darüber entscheiden, ob die letzten Schritte wiederholt werden sollen, oder nicht. Bisher haben wir Schleifen meist unendlich oft ablaufen lassen, oder mithilfe der Einstellung *Zählen* eine feste Zahl von Wiederholungen vorgegeben. Stellt man im Schleifen-Block unter Steuerung auf *Logiksignal*, so erscheint der Anschluss für Datenverbindungen, den wir dann mit dem Ausgang des Vergleichs-Blocks verbinden können (4).

Das Ergebnis des bisherigen Programmes ist relativ langweilig. Wenn alles klappt, dann fährt der Roboter soweit vorwärts, bis der Motor 2000 Grad für die Umdrehungen registriert hat, und hält dann an, wofür die beiden Stopp-Blöcke verantwortlich sind. Nun müsste sich der Roboter anschließend um 90° drehen und die nächste Kante abfahren. Nach vier derartigen Schritten hat er dann das Quadrat abgefahren.



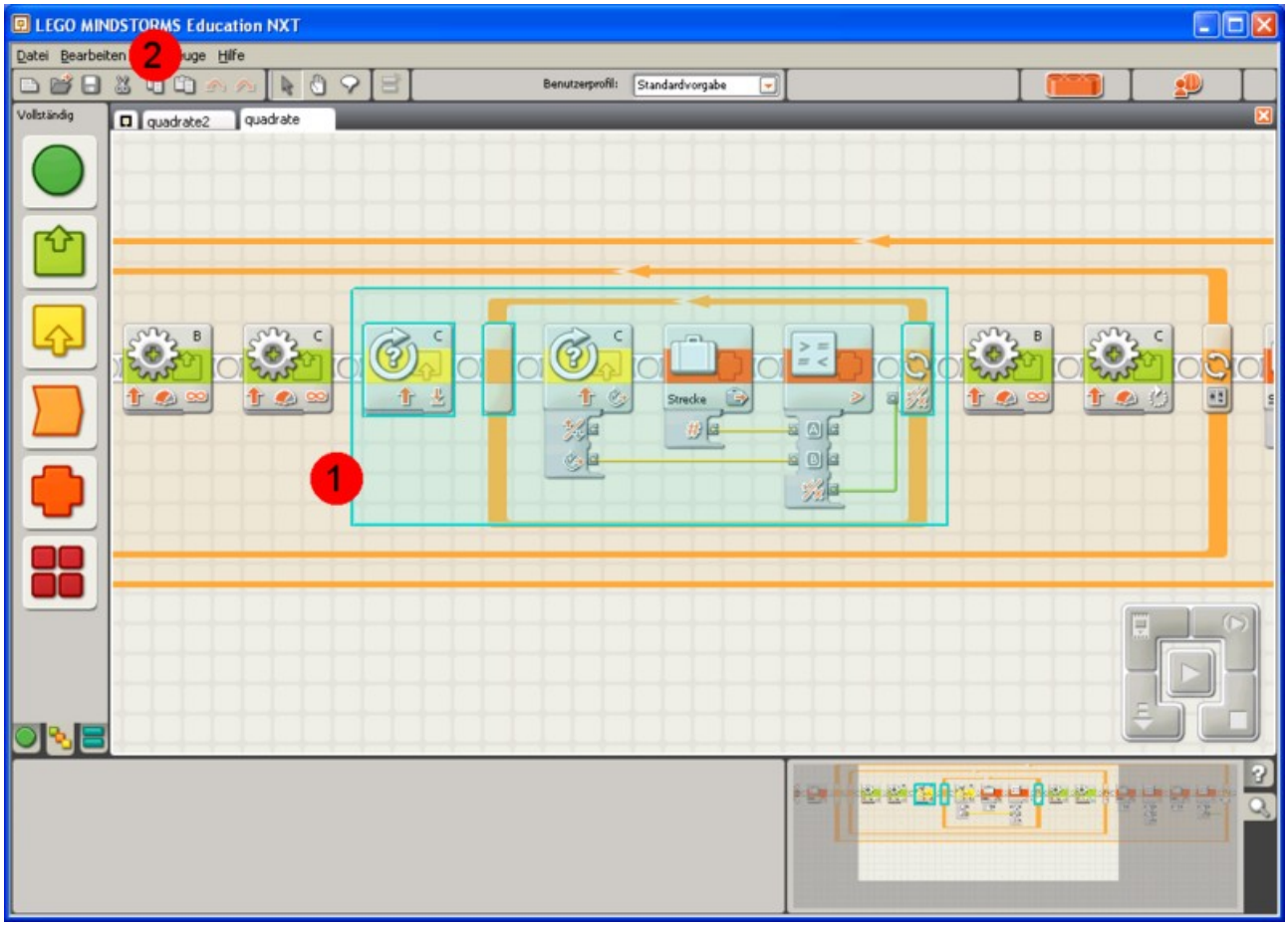
Nachdem der Roboter sein Quadrat abgefahren hat, muss die Streckenlänge reduziert werden, also z.B. auf 1500 statt vorher 2000, dann kann der Roboter das nächste Quadrat fahren. Diese beiden Schritte "Strecke verringern" und "Quadrat fahren" macht er dann entweder noch dreimal, oder solange bis die Streckenlänge unter einem Grenzwert (z.B. 100) liegt.



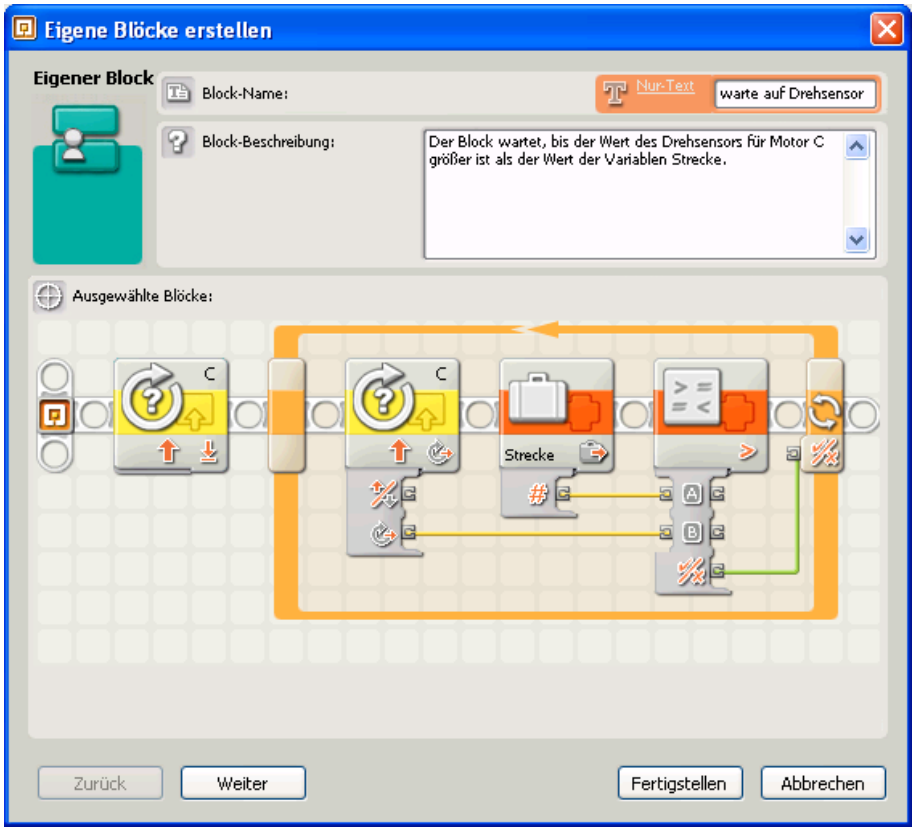
Das klingt zwar ganz einfach, erfordert aber insgesamt vier zusätzliche Blöcke, wenn man die neue Schleife nicht mitzählt. Der erste Block (1) liest den Wert der Variablen aus, der zweite Block subtrahiert 500 davon (2), der dritte Block schreibt den neuen Wert in die Variable (3) und der vierte Block vergleicht (4) den Wert der Variablen mit 100. Der Ausgang des Vergleichsblocks ist dann mit dem Dateneingang der Schleife verbunden (5).

Eigene Blöcke

Das letzte Beispiel ist recht groß geworden, das Programm passt nicht mehr ganz auf den Bildschirm. Durch das damit notwendige Scrollen wird die Übersicht erschwert. Zum Glück gibt es auch für dieses Problem eine Lösung. Man kann mehrere logisch zusammengehörige Blöcke zu einem eigenen Block zusammen fassen.



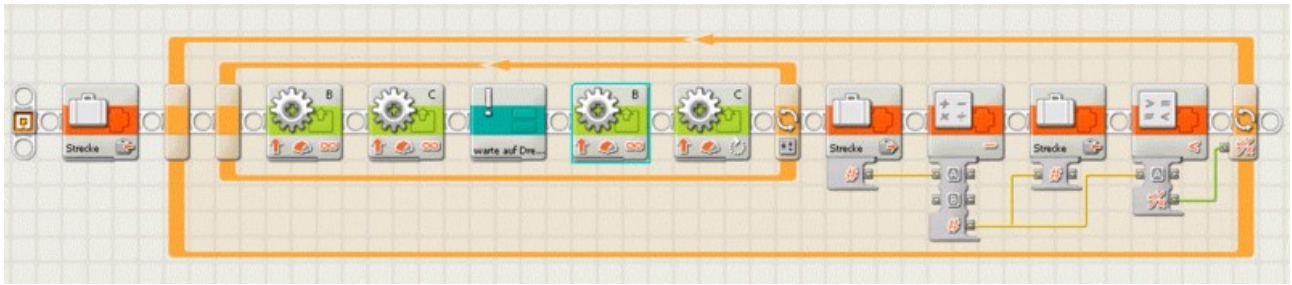
Dazu markiert man alle zusammengehörigen Blöcke mit der Maus und geht dann im Menü bearbeiten auf den Menüpunkt Neuen eigenen Block erstellen. Der Computer ist einen Augenblick beschäftigt, dann kommt ein Dialogfenster, indem man weitere Einstellungen vornehmen kann.



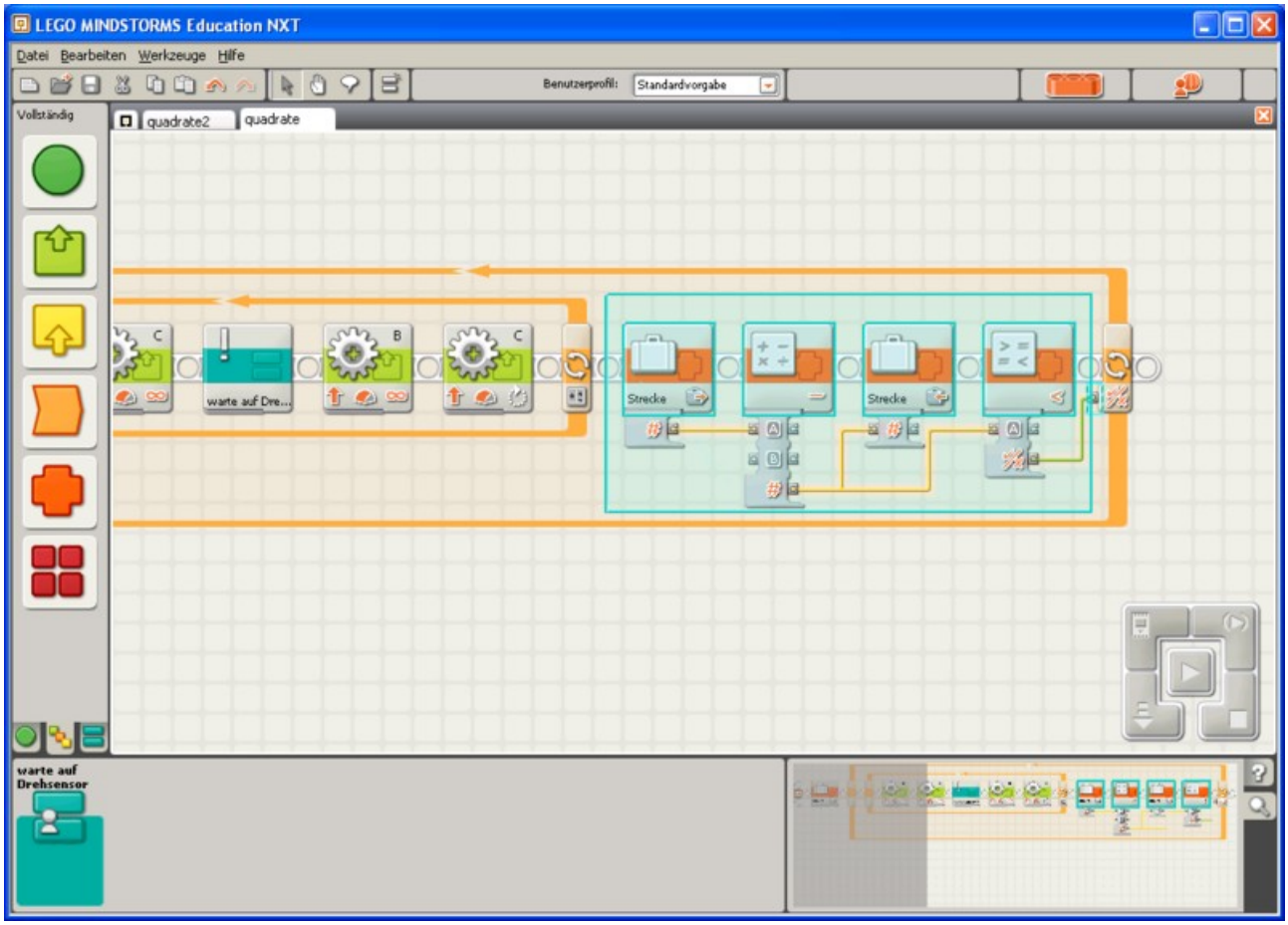
Vor allem muss man einen Namen für den Block festlegen, danach kann man dann eine längere Beschreibung für das eingeben, was der Block tut.

Wenn man auf den Button *Fertigstellen* klickt, dann wird der neue Button eingerichtet und mit einer Standard-Darstellung versehen. Wollte man diese Darstellung verändern hätte man auf *Weiter* klicken und dann den Button gestalten können.

Das Programm ist jetzt deutlich kürzer geworden und auch übersichtlicher, sofern der eigene Block einigermaßen sinnvoll benannt ist.

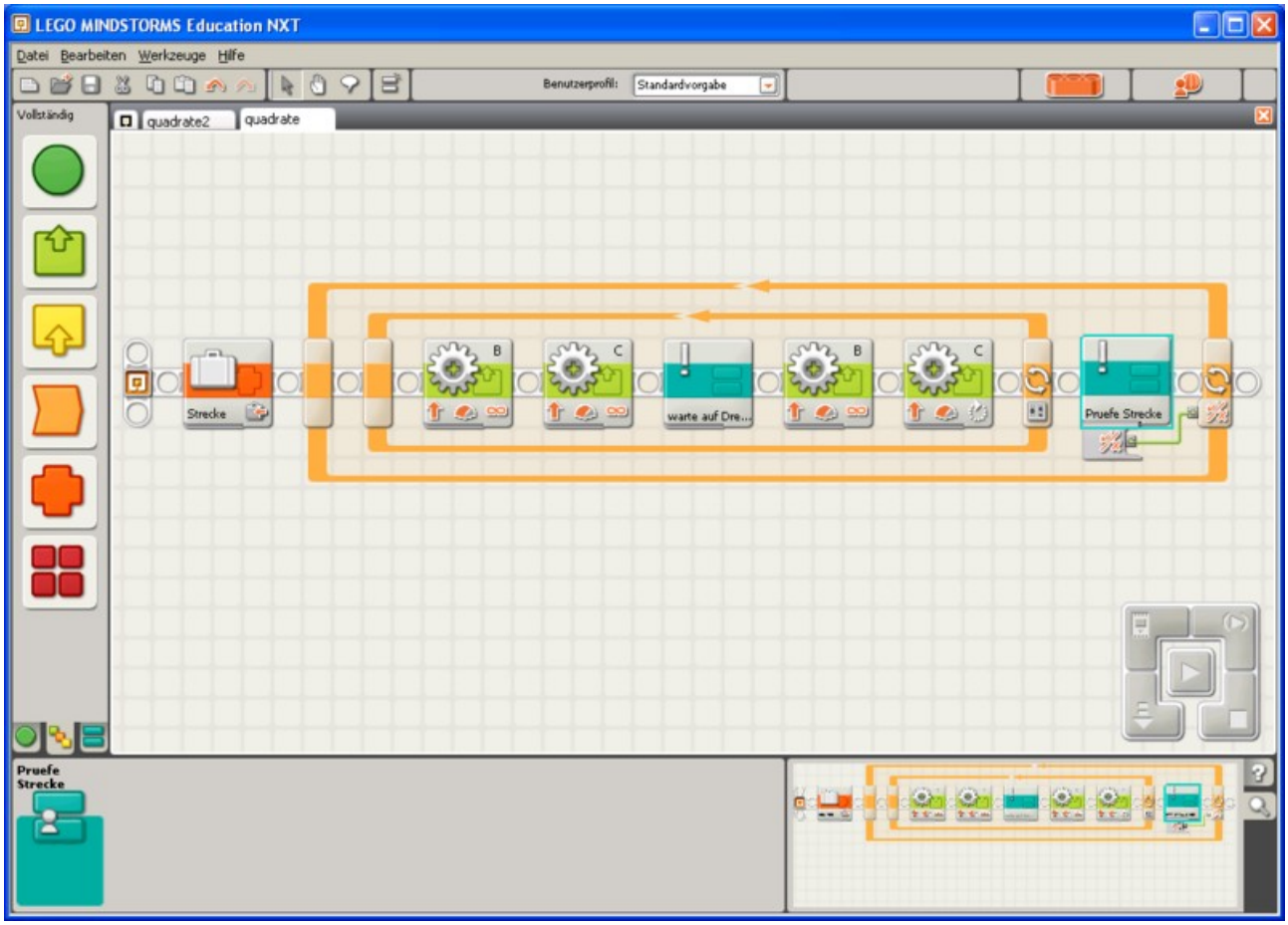


Leider passt das Programm immer noch nicht ganz auf den Bildschirm. Aber man kann ja weitere eigene Blöcke einrichten. Also wieder den entsprechenden Bereich markieren.



Der Bereich lässt sich zu einem Block zusammenfassen, obwohl eine Datenleitung herausführt. Diese Datenleitung führt später eben aus dem Block heraus. Eine sinnvolle Benennung der Blöcke ist wichtig, damit man auf einen Blick sehen kann, was der jeweilige Block tut. Im Zweifelsfall öffnet ein Doppelklick auf den Block diesen in einem neuen Arbeitsbereich.

Nun passt das gesamte Programm auf den Bildschirm.



Die beiden erstellten eigenen Blöcke findet man in der *Eigenen Palette* unter *Eigene Blöcke* und kann sie auch später jederzeit in eigenen Programmen einsetzen.

Quelle mit der jeweils aktuellen Version: „http://www.debacher.de/wiki/NXT-G“