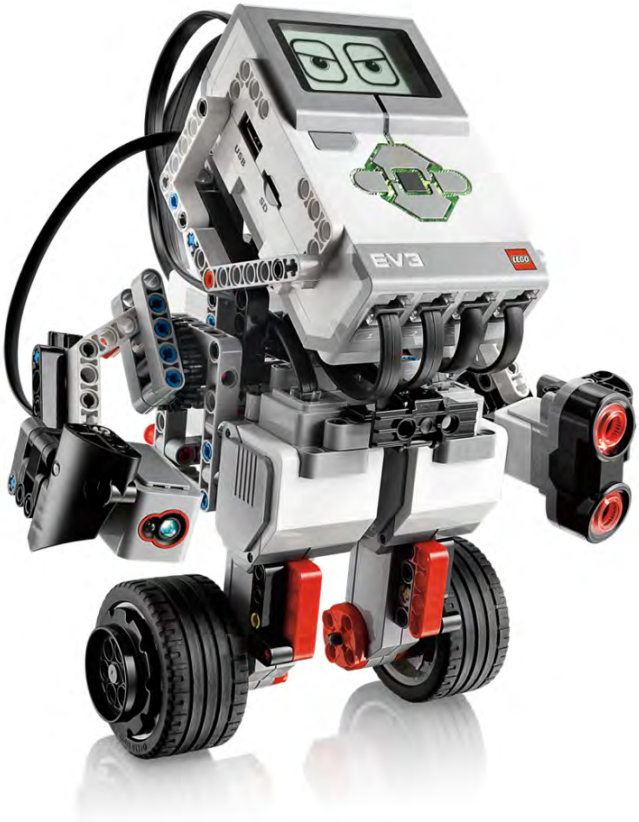
Technik für Kinder e.V.
Begeistern durch Machen

UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik

Modul:

Roboter-Programmierung



mindstorms NXT
mindstorms EV3

Version: 1.0

Dieses Modul wurde an der Lehr- und Forschungseinheit Mathematik und Informatik (LMI) der Universität Passau in Kooperation mit dem Verein „TfK - Technik für Kinder e.V.“ im Rahmen des Projekts Technikgruppen an Schulen - Kinder für Technik begeistern erarbeitet.
Mehr zu diesem Projekt erfahren Sie auf der Internetseite

http://www.fim.uni-passau.de/fim/fakultaet/lehrstuehle-professuren-und-fachgebiete-der-fim/didaktik-der-informatik/projekte/technikgruppen.html .

Ute Heuer

Didaktik der Informatik
ute.heuer@uni-passau.de

Wolfgang Pfeffer

Didaktik der Informatik / Mathematik
wolfgang.pfeffer@uni-passau.de

Modulübersicht

Im Rahmen des Kooperationsprojekts zwischen der Universität Passau und dem Verein Technik für Kinder e.V. sind folgende Module entstanden:

Modul Roboter-Programmierung

Dieses Modul bietet einen sehr kurzen Einstieg in die Programmierung eines Lego EV3 Roboters mit der Lego Mindstorms Education Software. Neben Hardware- und Software Anforderungen wird die Entwicklungsumgebung vorgestellt und anschließend Einstiegsaufgaben zu Motoren, Sensoren sowie vermischte Aufgaben bereitgestellt.

Modul App-Entwicklung auf Mobile Devices

Dieses Modul thematisiert umfassend die App-Entwicklung auf Mobile Devices mit dem AppInventor, einer graphischen Programmierumgebung. Ein einfacher und handlungsorientierter Zugang zu dieser Thematik steht dabei im Vordergrund. Neben Hardware- und Softwareanforderungen wird eine ausführliche Installationsanleitung angeboten. Die Einführungsaufgabe 'Schritt für Schritt zur ersten eigenen App' ermöglicht eine Einführung in den Umgang mit dem AppInventor. Gleichzeitig werden fast alle wichtigen Elemente behandelt. Anschließend umfasst das Modul 10 Aufgaben mit unterschiedlichem Schwierigkeitsgrad sowieso ausführlichen Lösungen. Exkurse zu Themen wie GPS und Dijkstra-Algorithmus runden die Handreichung ab.

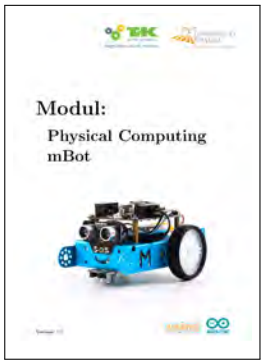
Modul App-Entwicklung mit Java Android

Dieses Modul baut thematisch auf dem Modul 'App-Entwicklung auf Mobile Devices' auf, anstelle der graphischen Programmierung werden die Applikationen nun mit Java Android entwickelt. Da die Einstiegshürde in Java Android vergleichsweise hoch ist, sind grundlegende Java-Kenntnisse unumgänglich. Es wird ein sehr ausführlicher sowie kleinschrittiger Einstieg in die Programmierung mit Java Android geboten. Anschließend stehen verschiedene Projekte mit umfangreichen Lösungen zur Auswahl (z.B. Vokabel-App, Quiz-App, 2048-App, Datenspeicher oder LegoPilot-App). Dabei werden wichtige Konzepte der Sekundarstufe II behandelt.

Modul Physical Computing

Dieses Modul kombiniert Elemente aus der Physik und der Informatik und basiert auf dem Einplatinencomputer Raspberry Pi. Zunächst geht es darum, (einfache) Schaltungen zu erstellen und mit konstanter Spannungsquelle zu arbeiten. Hierbei wird mit verschiedenen Schaltungselementen experimentiert (z.B. LED, Widerstand, LDR, Poti, Servomotor). Anschließend soll an manchen Pins gezielt Spannung angelegt oder nicht angelegt werden. Dazu werden die GPIO-Pins mit der Programmiersprache Python konfiguriert und angesteuert. Die Handreichung beinhaltet ein eigenes Kapitel zu Python, in welchem man sich anhand kleiner Aufgaben mit der Syntax der Programmiersprache vertraut machen kann.

Modul Physical Computing mBot



Dieses Modul kann als Schnittstellenmodul zwischen Roboter-Programmierung und Physical Computing gesehen werden. Bei mBot handelt es sich um ein handliches Roboterfahrzeug, das mit verschiedenen Sensoren und Motoren ausgestattet ist, die mit dem Mikrocontroller Arduino verbunden werden können. Im Bezug auf die Programmierung bietet das Modul zwei Zugangswege: Ähnlich zur App-Entwicklung mit dem AppInventor kann der mBot mithilfe der graphischen Programmierumgebung Scratch angesteuert werden. Ein zweiter Teil des Moduls bietet die Programmierung des mBot mit Hilfe von Arduino-C.

Quelle des Bildes: https://www.colourbox.de/vektor/buch-vektor-5814145

Inhaltsverzeichnis

Kapitel 1

Über das Modul

Roboter-Programmierung – Dieses Modul bietet einen kleinen Einstieg in die Programmierung von Lego Mindstorms EV3 Roboter mit der Lego Mindstorms Education Software. Aufgrund des vielfältigen Materials, das es in Bezug auf diese Thematik bereits gibt, ist das Modul im Umfang sehr knapp gehalten.

Neben einem kurzen Kapitel zu Hardware- und Softwareanforderungen enthält die Handreichung einen Abschnitt zum Einstieg in die Lego-Entwicklungsumgebung. Anschließend finden sich Aufgaben zu den verschiedenen Motoren und Sensoren des EV3-Roboters. Abgerundet wird das Modul durch vermischte Aufgaben.

Quelle des Bildes: https://www.colourbox.de/search/find?allow_empty_search=0&media_type=&order=relevance&orientation=all&resolution=all&editorial=no&q=einf

Kapitel 2

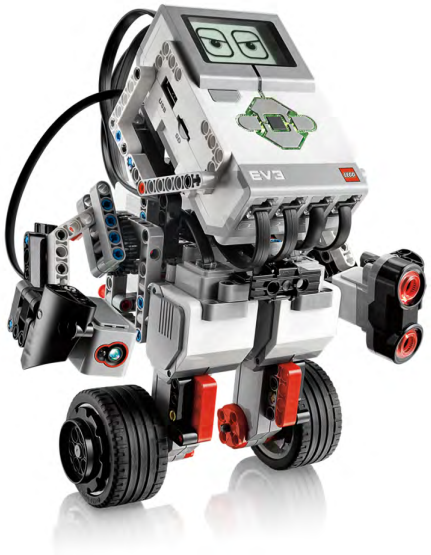
Hardware und Software Anforderungen

Überblick:

Dieses Kapitel gibt einen Überblick über die Hard- und Software, die für dieses Modul benötigt werden.

2.1 Hardware

Jeder Arbeitsplatz benötigt einen Lego Mindstorms EV3 Roboter sowie einen Laptop bzw. Desktop-PC.



2.2 Software

Als Entwicklungsumgebung verwenden wir die Lego Mindstorms Education Software. Sie müssen hierfür eine Lizenz (entweder Einzellizenz oder Schullizenz) erwerben und können diese anschließend unter https://education.lego.com/de-de/lesi/middle-school/mindstorms-education-ev3/all-about-ev3/software herunterladen. Die Lizenz ist beim Kauf eines EV3-Bausatzes in Normalfall darin enthalten.

Kapitel 3

Lego Mindstorms Education EV3 Software

Überblick:

In diesem Kapitel verwenden wir zur Programmierung die intuitive Entwicklungsumgebung von LEGO verwenden - die LEGO MINDSTROMS Education EV3 Software. Die Software eignet sich gut für kleine bis mittelgroße Projekte und kann für Schülerinnen und Schüler vor allem beim Einstieg in die Thematik hilfreich sein.
Die EV3-Software arbeitet mit einer grafischen Programmieroberfläche - ein mögliches Programm ist in folgender Abbildung dargestellt:

Der Roboter fährt zunächst eine Umdrehung vorwärts, anschließend eine Umdrehung rückwärts und spielt zum Abschluss einen Ton ab.
Im nächsten Kapitel stellen wir noch eine weitere Möglichkeit da, den Roboter zu programmieren. Hier werden wir mit der C-ähnlichen Programmiersprache NXC (Not eXactley C) arbeiten.

3.1 Die Entwicklungsumgebung

3.1.1 Aufbau

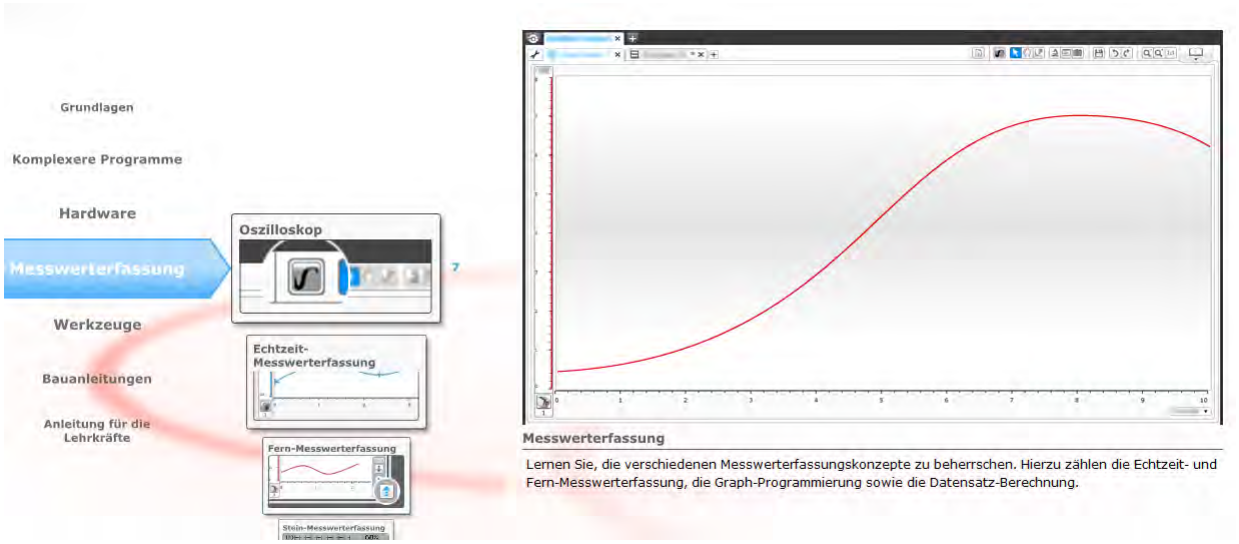
Im Vergleich zu der NXT-Software bietet die EV3-Software einige Neuerungen:

Die ersten zwei Schaltflächen auf der linken Seite bieten Bauanleitungen für verschiedenen Roboter-Modelle, wie z.B. `Znap`, `Elefant` (beide nur mit Erweiterungsset möglich) oder `Gyro Boy` und `Farbsortierer`:

Unter der Rubrik `Schnellstart` findet man die Bedienungsanleitung und einige Videos, in denen die Funktionsweise der EV3-Software erläutert wird. Aus diesem Grund werden wir an dieser Stelle auch auf eine ausführliche Beschreibung verzichten.
Unter `Datei` kann man ein neues Projekt anlegen[1] oder ein vorhandenes öffnen.

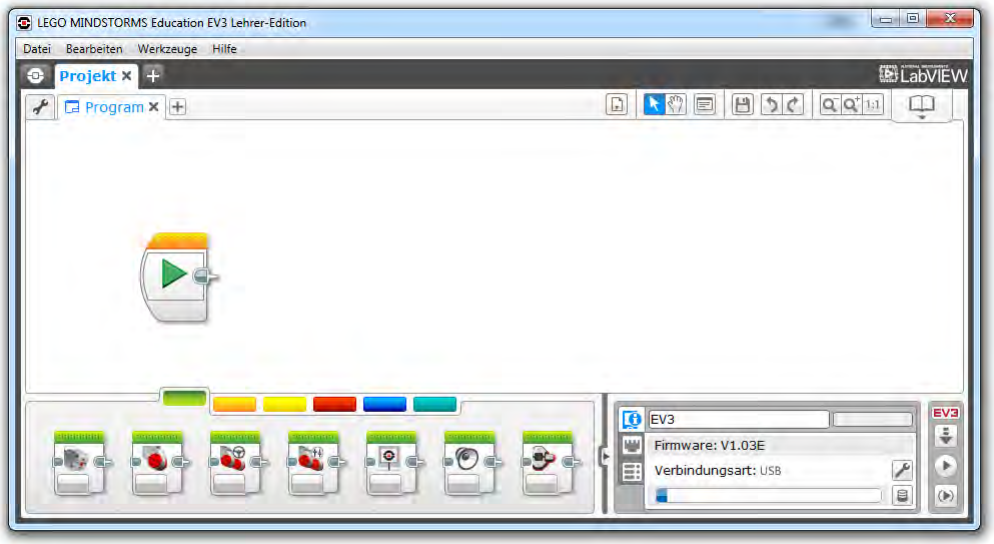
Unter der Rubrik `Robot Educator` befinden sich weitere Tutorials zu den einzelnen Sensoren, Messwerterfassung und weiteren Werkzeugen.

[1] dies geht auch schneller über das +-Zeichen unter der Menüleiste



3.1.2 Neues Projekt anlegen

Wie oben beschrieben können wir neues Projekt durch Klicken auf das +-Zeichen unter der Menüleiste anlegen. Ist der EV3-Roboter mit dem Computer verbunden, leuchtet das EV3 Zeichen rechts unten rot und wir können durch Klicken auf den Pfeil ⇓ unser Projekt auf den Roboter laden bzw. mit Play ▷ das Programm auf den Roboter laden und sofort abspielen.



Die verfügbaren Bausteine sind unten in verschiedenfarbigen Kategorien geordnet und können via Drag & Drop zu unserem Programm hinzugefügt werden. Es werden nur die Baustein ausgeführt, die mit dem Startbaustein verbunden sind - alle anderen Bausteine werden bei der Programmausführung nicht berücksichtigt und heben sich auch farblich vom Programm ab:

3.2 Die Motoren

In diesem Abschnitt lernst du, wie du die Motoren der EV3 Roboter ansteuerst. Wir arbeiten dabei mit der klassischen Variante des Roboters[2]. Die zwei Motoren seien dabei an den Ports `B` und `C` angeschlossen und der Roboter per USB-Kabel mit dem Computer verbunden.

3.2.1 Vorwärts und rückwärts fahren

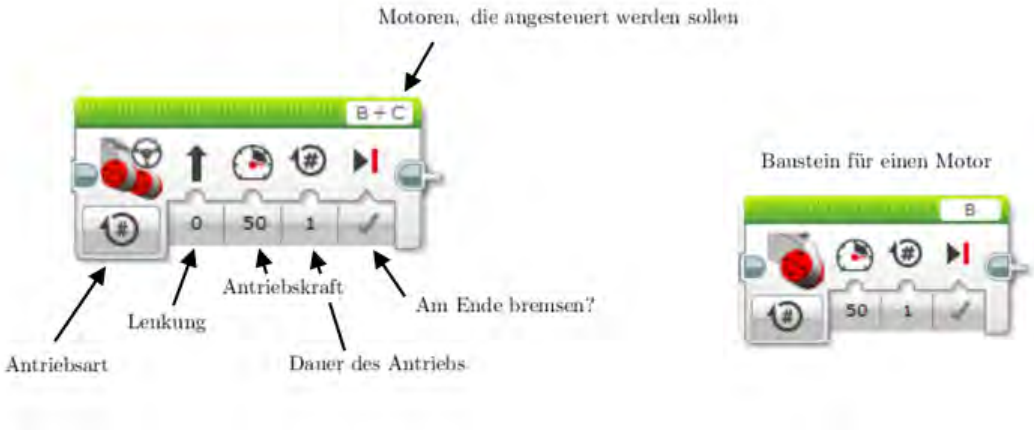
Möchte man den Roboter nur vorwärts fahren lassen, könnte man intuitiv folgendes Programm erstellen:

Man schaltet einfach beide Motoren an. Teste obiges Programm! Was stellst du fest? Hast du eine Erklärung hierfür?

Vorwärts fahren mit fester Umdrehung

Wir legen ein neues Projekt an und wollen unseren Roboter zunächst um eine Rad-Umdrehung vorwärts fahren lassen. Beachte dabei, dass in deinem Roboter zwei Motoren für den Antrieb verbaut sind!

Hinweis: Du findest den benötigten Block in der unteren Leiste in der grünen Rubrik!



Bestimmte Zeit vorwärts fahren

Ändere dein Programm so ab, dass dein Roboter nun 5 Sekunden lange vorwärts fährt. Anschließend soll er eine Note abspielen.

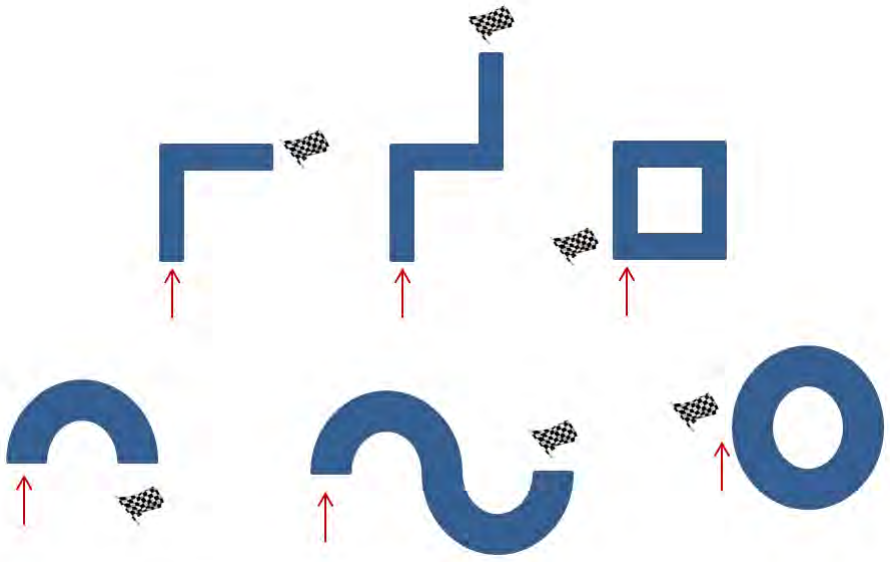
Rückwärts fahren

Überlege dir, wie man den Roboter rückwärts fahren lassen könnte. Experimentiere dazu mit der Antriebskraft. Lass anschließend deinen Roboter zunächst 3 Sekunden vorwärts und anschließend drei Sekunden rückwärts fahren.

[2] Bauanleitung: `Robot Educator` → Bauanleitungen → Fahrgestell

3.2.2 Bestimmte Strecken fahren

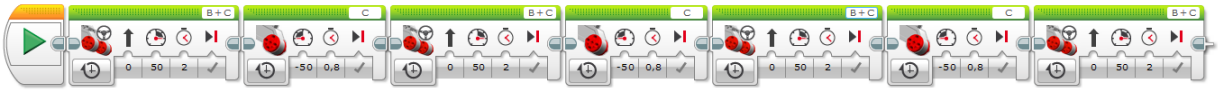
Wir möchten nun nicht mehr bloß vorwärts und rückwärts fahren, sondern auch Kurven fahren und rechtwinklig abbiegen können. Überlege dir zunächst verschiedene Möglichkeiten, wie du es schaffst, dass dein Roboter eine Kurve fährt bzw. rechtwinklig abbiegt. Versuche dann folgende Strecken mit deinem Roboter zu fahren:



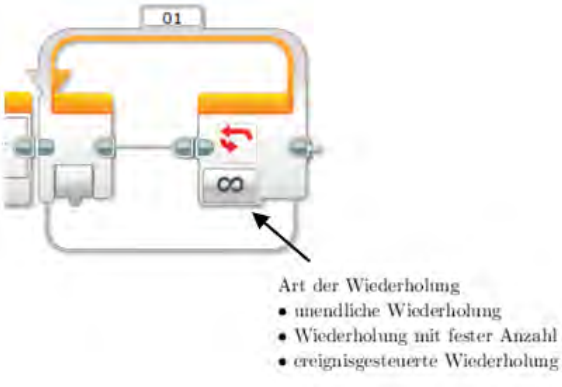
Überlege dir noch eigene Strecken und versuch diese zu fahren!!

Wiederholungen

Betrachten wir exemplarisch die quadratische Strecke genauer. Deine Lösung könnte etwa wie folgt aussehen:



Vielleicht ist dir bei der Lösung aufgefallen, dass die Sequenz «Vorwärts fahren → Abbiegen» drei mal vorkommt. Um sich hier Zeit zu sparen, kann man auch eine Wiederholung verwenden. Diesen Baustein findest du in der orangen Rubrik:



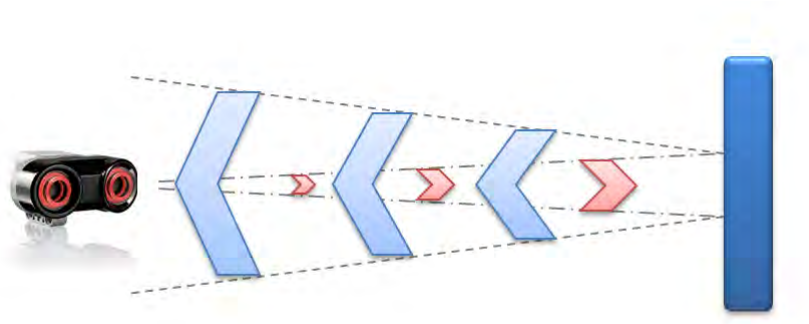
Alle Blöcke in dieser Wiederholung werden so oft wiederholt, wie angegeben. Schaffst du es, obiges Programm abzukürzen?

3.3 Die Sensoren

In diesem Abschnitt lernen wir die verschiedenen Sensoren des EV3-Roboters kennen.

3.3.1 Der Ultraschallsensor

Funktionsweise eines Ultraschall-Sensors



Ein Ultraschallsensor lässt sowohl das Senden als auch das Empfangen von Schallimpulsen zu und kann damit Entfernungen messen. Nachdem der Ultraschallsensor einen Schallimpuls ausgesendet hat, wartet er auf den Empfang dieses Schallimpulses. Der ausgesendete Schallimpuls wird von einem Objekt reflektiert und kann somit vom Ultraschallsensor empfangen werden. Anhand der Zeit, die zwischen Aussenden und Empfangen des Schallimpulses vergeht, berechnet der Ultraschallsensor die Entfernung zu dem detektierten Gegenstand. Fledermäuse orientieren sich mit dem gleichen Prinzip in der Dunkelheit bei der Beutejagd.

Wir ergänzen unseren Roboter nach Anleitung um den Ultraschallsensor und verbinden diesen mit Port 4.

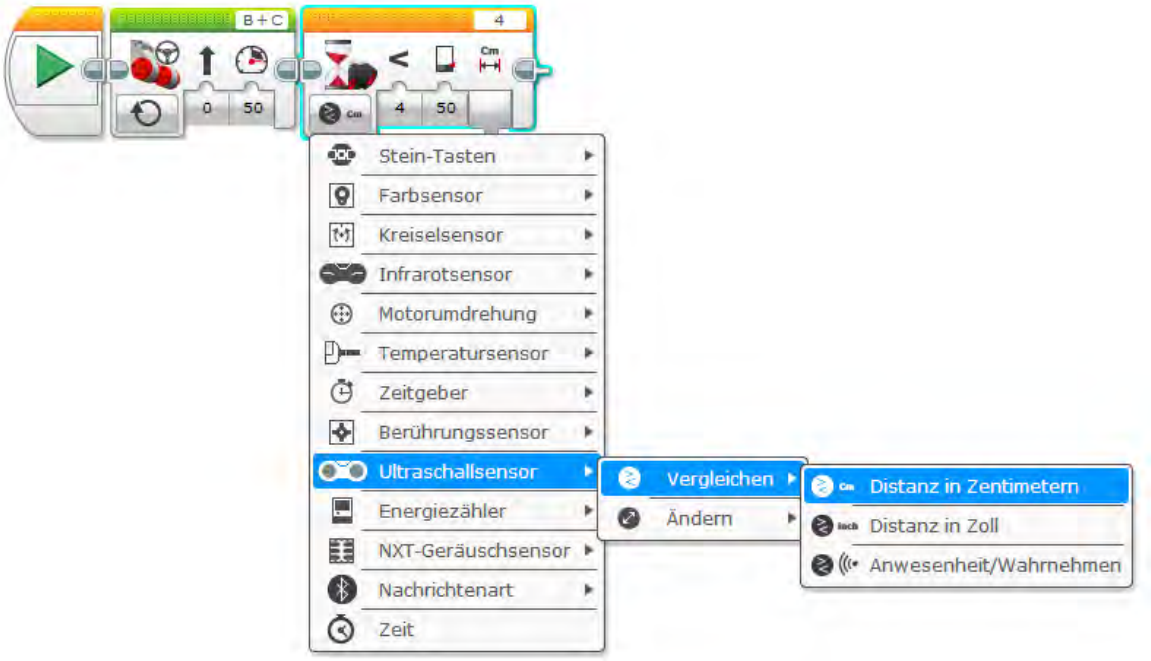
Port-View

Wenn wir beim Roboter in die dritte Kategorie wechseln, finden wir die Rubrik ≪Port-View≫. Hier können wir uns die Werte der angeschlossenen Sensoren anschauen. Schau dir die Messwerte an, die der Ultraschallsensor liefert, wenn du die Entfernung zu deiner Hand oder einem anderen Gegenstand misst.

Messwerte mit Lego-Software auslesen

Wir wollen natürlich nicht nur die Messwerte auf dem Display unseres Roboters angezeigt bekommen, sondern diese auch in unseren Programmen verwenden können. Wir erzeugen ein neues Projekt und wechseln in der Leiste unten in die gelbe Rubrik. Hier gibt es einen Baustein, mit dem wir die Werte des Ultraschallsensors auslesen können:

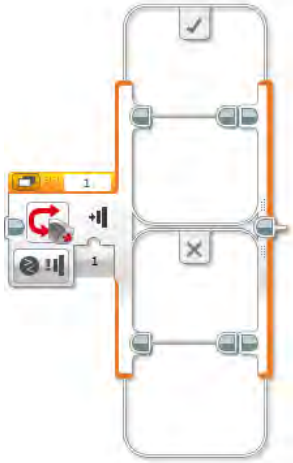
Diese Bausteine werden wir zunächst allerdings nicht verwenden. Wir können auch in vielen anderen Bausteinen auf die Daten von Sensoren zurückgreifen, z.B. kann man im Wartebaustein nicht nur n Sekunden warten, sondern auch solange warten, bis der gemessene Wert eines Sensors unter / gleich / über / einem bestimmten Schwellwert ist:



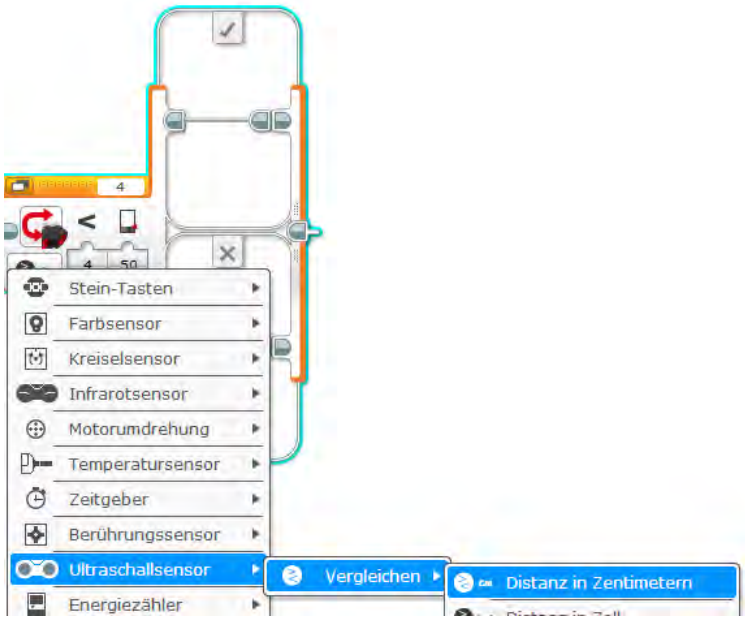
Fahren bis zum Hindernis

Unser nächstes Programm soll den Roboter solange gerade aus fahren lassen, bis er ein Hindernis bemerkt, dass nur noch 20 cm von ihm entfernt ist. In diesem Fall soll er anhalten.

Hinweise: Für diese Aufgabe gibt es viele unterschiedliche Lösungen. Versuch die Aufgabe einmal unter Verwendung eines Warte-Bausteins und einmal unter Verwendung eines Schalter-Bausteins zu lösen. Den Schalter findest du in der orangen Kategorie:



Mit dem Schalter können wir zwei Fälle unterscheiden. Im linken Teil testet der Schalter, ob eine gewisse Bedingung erfüllt ist. Falls ja, führt er die Bausteine aus, die wir in den oberen Teil ziehen können, falls nein, führt er die unteren Bauteile aus. Als Bedingung können wir hier auch auf den Ultraschallsensor zugreifen:



Überlege dir zudem noch, wie du sicherstellst, dass der Roboter ständig fährt und erst stehen bleibt, falls er ein Hindernis erkennt. Mit einem Schleifen-Interrupt Baustein (orange Kategorie) kann man eine endlose Wiederholung abbrechen.

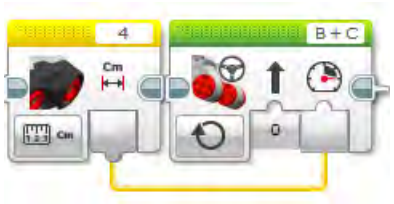
Unfallfreies Fahren

Bisher bleibt der Roboter stehen, wenn er ein Hindernis vor sich erkennt. Nun soll er in diesem Fall kurz rückwärts fahren, sich drehen und anschließend wieder vorwärts fahren, bis er wieder ein Hindernis erkennt, usw.

Hinweis: Du kannst dein Programm mit der Zurücktaste (links oben) am Roboter beenden.

Geschwindigkeit regulieren

Detektieren wir mit dem Ultraschallsensor kein Hindernis, wollen wir «Gas geben». Je näher wir uns einem Hindernis nähern, umso langsamer wollen wir werden. Mit Hilfe einer Datenleitung können wir beispielsweise die gemessenen Werte des Ultraschallsensors als Maß für die Motorleistung verwenden:



Mit der Datenleitung wird der vom Ultraschallsensor gemessene Wert direkt als Leistung des Motors festgelegt.

Erstelle ein Programm, dass die Motorleistung in Abhängigkeit zu der Entfernung des nächsten Hindernisses steuert. Befindet sich das Hindernis weniger als 20 cm entfernt, soll der Roboter wieder anhalten, kurz zurücksetzen, sich drehen und anschließend wieder mit regulierter Motorleistung weiterfahren.

Display-Anzeige

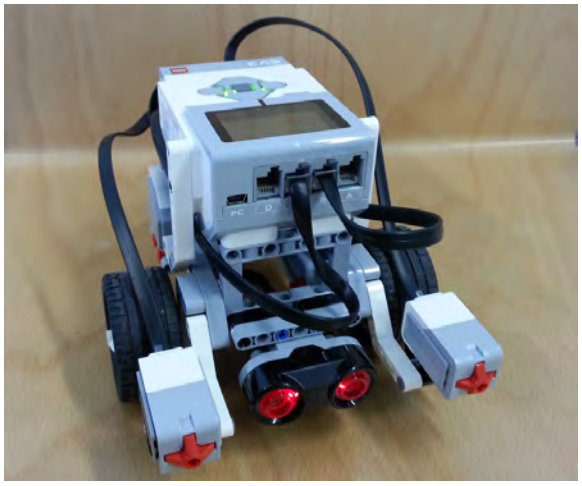
Mit dem Baustein

kann man beispielsweise vorhandene Bilddateien auf dem Roboterdisplay ausgeben.

Erweitere das Programm vom Unfallfreien Fahren so, dass der Roboter verärgert schaut, wenn er auf ein Hindernis getroffen ist und glücklich wenn er normal vorwärts fährt.

3.3.2 Der Berührungssensor

Baue an deinem Roboter die zwei Berührungssensoren entsprechend der Anleitung an. Schließe den in Fahrtrichtung rechten Berührungssensor an Port 3 und den linken Sensor an Port 2 an. Finde heraus, welche Werte der Berührungssensor in gedrücktem und nicht gedrücktem Zustand ausgibt (*Hinweis:* `Port View`). Dein Roboter sollte wie folgt aussehen:



Ton abspielen

Der Roboter soll zunächst noch nicht fahren, sondern einfach zwei verschiedene Töne abspielen, je nachdem welchen Berührungssensor man drückt.

Fahren bis zum Hindernis

Gleiche Aufgabe wie beim Ultraschallsensor. Verbinde hierzu die beiden Drucksensoren mit einer «Stoßstange». Wenn der Roboter bemerkt, dass er an einem Hindernis angefahren ist, soll er zunächst anhalten und einen Ton abspielen.
Anschließend soll er rückwärtsfahren, sich drehen und wieder normal weiterfahren.

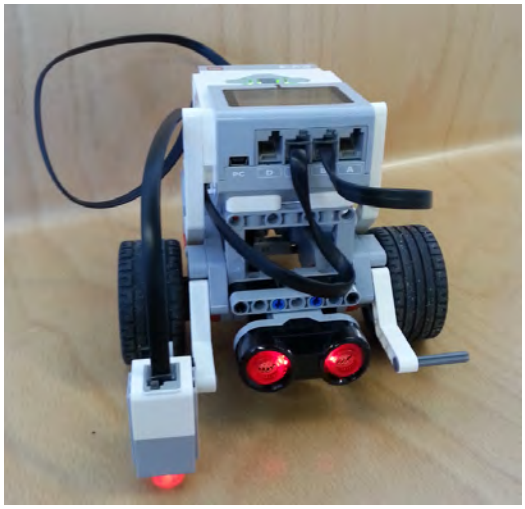
Geschicktes Rückwärtsfahren

Unterscheide folgende Fälle: Falls der in Fahrtrichtung rechte Berührungssensor gedrückt wird, soll der Roboter rückwärts fahren und sich anschließend gegen den Uhrzeigersinn drehen und weiterfahren. Falls der linke Berührungssensor gedrückt wird, soll der Roboter rückwärts fahren und sich anschließend im Uhrzeigersinn drehen und weiterfahren.

3.3.3 Der Licht- und Farbsensor

Bei diesem Sensor handelt es sich um eine Kombination aus Licht- und Farbsensor. Der Sensor kann acht verschiedene Farben erkennen und dient darüber hinaus auch als Lichtsensor, der die Lichtstärke messen kann. Dieser Sensor eignet sich somit gut, um beispielsweise einen Roboter auf einem Tisch fahren zu lassen, ohne runter zu fallen, eine Linie zu verfolgen oder Bausteine nach ihren Farben sortieren.

Wir verwenden den Sensor zunächst als Lichtsensor. Baue den Roboter entsprechend folgender Abbildung um:



Tischkante detektieren

Der Roboter soll auf die Tischkante zu fahren und rechtzeitig stehen bleiben, bevor er abstürzt. Schau dir zunächst an, welche Werte der Lichtsensor auf dem Tisch misst und welche Werte er misst, wenn er über die Tischkante hinausragt. Mit diesen Werten kannst du nun deinen Roboter rechtzeitig stoppen lassen.

Hinweis: Lass den Roboter nicht zu schnell fahren, sonst bleibt er nicht rechtzeitig stehen - mit Leistung 25 sollte es funktionieren.

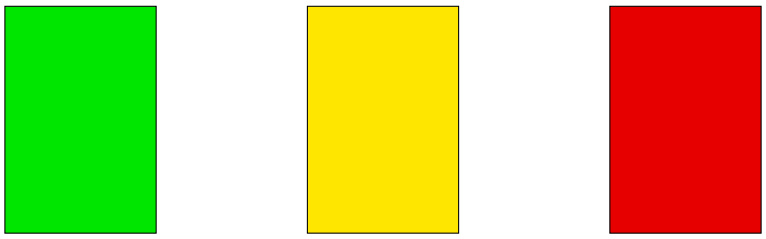
Zurückweichen

Detektiert der Roboter eine Tischkante, soll er nicht wie bisher stehen bleiben, sondern kurz zurückweichen, sich drehen und wieder weiterfahren. Erstelle ein Programm, dass deinen Roboter auf dem Tisch fahren lässt, ohne abzustürzen!

Hinweis: Es kann passieren, dass der Roboter beim Rückwärtsfahren abstürzt. Du kannst versuchen dies zu vermeiden, indem du zwei Lichtsensoren verbaust!

Ampel

Für diese Aufgabe brauchst du ein grünes, ein gelbes und ein rotes Kärtchen:



Baue den Roboter so um, dass der Farbsensor nach vorne zeigt.

Mit den farbigen Kärtchen wollen wir nun eine Art Ampel nachstellen: Zeigt man dem Roboter ein grünes Kärtchen, soll er mit starker Leistung vorwärts fahren. Bei gelber Karte soll er langsamer werden und bei roter Karte stehen bleiben. Zeigt man ihm anschließend wieder die gelbe bzw. grüne Karte, soll er weiterfahren.

Ampel und Hindernisse

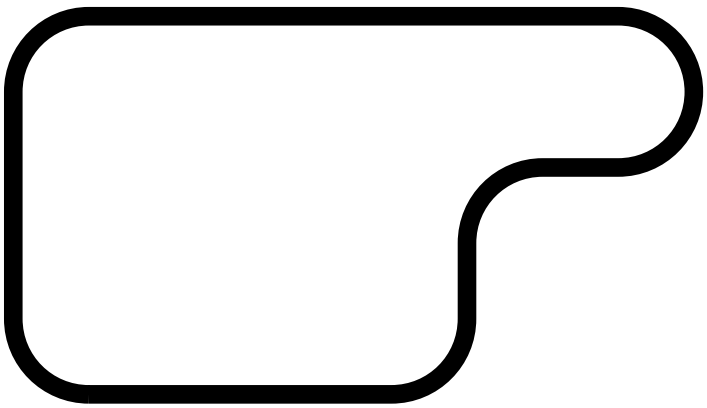
Erweitere deinen Roboter mit dem Ultraschallsensor. Ergänze dein Programm so, dass der Roboter weiterhin auf die farbigen Kärtchen entsprechend reagiert, gleichzeitig aber auch Hindernissen ausweicht, d.h. z.B. nicht gegen eine Wand fährt.

Display-Anzeige

Du kannst auch hier wieder dein Programm erweitern und den Roboter unterschiedlich schauen lassen, je nachdem welche Farbe ihm gerade gezeigt worden ist.

Linienverfolgung

Der Roboter soll mit Hilfe des Lichtsensors einer schwarzen Linie folgen, die auf einem weißen Hintergrund abgebildet ist. Für diese Aufgabe musst du dir zunächst einen Parcour überlegen und auf einem größeren Papier (DinA0) ausdrucken. Mache die Kurven für den Anfang nicht zu steil!



Beispiel für einen möglichen Parcour

Für die Linienverfolgung gibt es verschiedene Ansätze:

(1) Setze die Linienverfolgung zunächst mit einem Sensor um, der den Roboter an der Grenze (z.B. außen) zwischen schwarzer Linie und weißem Hintergrund entlangfahren lässt.

Hinweise: Ermittle zunächst über `PortView`, welche Werte der Lichtsensor auf der schwarzen Linie bzw. auf weißem Hintergrund misst.

Überlege dir dann, wie du den Roboter ansteuern musst, damit er entlang der Grenze zwischen weißer und schwarzer Linie fährt.

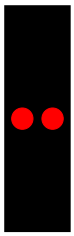
Wettbewerb[a]

Fügt eurer Strecke eine Startlinie hinzu und messt die Zeit, die eure Roboter jeweils für die Strecke brauchen. Wer schafft das Programm, mit dem der Roboter am schnellsten um die Strecke kommt?

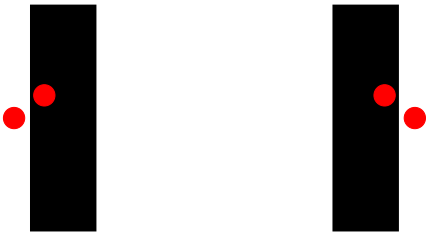
[a] **Quelle des Bildes:** https://www.colourbox.de/bild/stoppuhr-bild-5173766

(2) Setze die Linienverfolgung anschließend mit 2 Sensoren um, indem ...

(a) sich die beiden Sensoren auf der schwarzen Fahrbahn befinden sollen



Überlege dir, wie du reagieren musst, falls sich ein Sensor abseits der Linie befindet:



(b) sich die beiden Sensoren auf dem weißen Hintergrund befinden sollen

Überlege dir, wie du reagieren musst, falls sich ein Sensor auf der Linie befindet:

Wettbewerb[a]

Führe für obige zwei Szenarien wieder einen Wettbewerb durch und schaut, wer das Programm schreibt, mit dem der Roboter am schnellsten um den Parcour kommt.

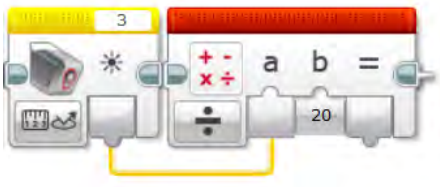
[a]**Quelle des Bildes:** https://www.colourbox.de/bild/stoppuhr-bild-5173766

Weiche Linienverfolgung

Obige Ansätze lassen sich gut umsetzen, haben aber den Nachteil, dass sie meist sehr ruckartige Bewegungen des Roboters nach sich führen. Ein anderer Ansatz hierfür ist die weiche Linienverfolgung, die man bis hin zu einer proportionalen Linienverfolgung noch weiter verfeinern kann.

Wir verwenden hierfür einen Roboter mit einem Lichtsensor, der an der äußeren Linie des Parcours gegen den Uhrzeigersinn entlangfahren soll, d.h. die Linie ist immer auf der linken Seite des Roboters.

Je nachdem wie stark die Lichtintensität ist, soll der Roboter mehr oder weniger nach links / rechts lenken. Ziel ist es, möglichst an der Grenze zwischen hell und dunkel (Sensorwert 50) entlangzufahren. Hierfür unterscheiden wir 5 verschiedene Stufen[3] Aus diesem Grund teilen wir die gemessene Lichtintensität (Messwerte zwischen 0 und 100) durch 20. Dies erreicht man mit dem Mathematik-Baustein Division und einer Datenleitung:

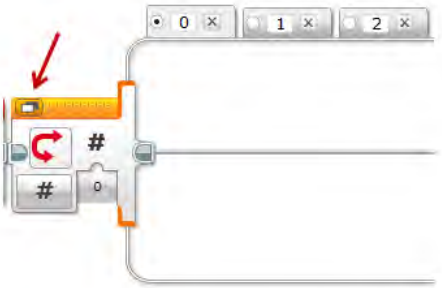


Um die fünf Fälle unterscheiden zu können, benötigen wir einen Schalter, der nach einem numerischen Wert unterscheidet. Wir weisen dem Schalter über die +-Schaltfläche die Fälle 0,1,2,3 und 4 zu:

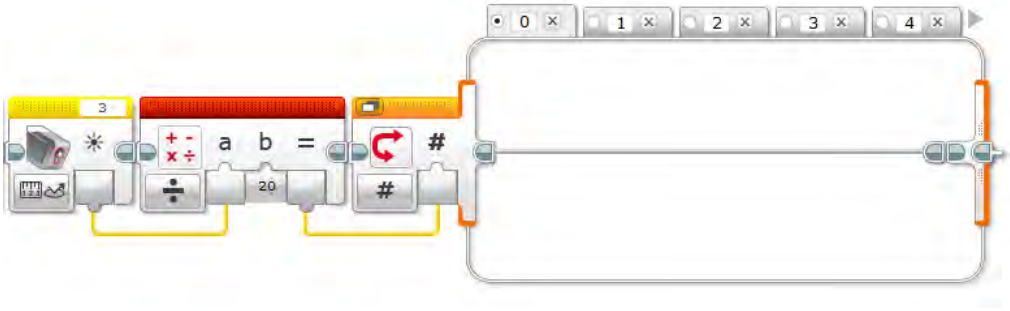
[3]Natürlich kann man die Stufen auch noch weiter verfeinern.

Wir haben aufgrund der besseren Lesbarkeit in die Reiteransicht des Schalters gewechselt.

Zwischen Reiteransicht und der offenen Ansicht kannst du links oben am Schalterbaustein wechseln:



Als numerischen Wert für den Schalter wollen wir nun das Ergebnis unserer Division verwenden. Dies schaffst du wieder mit einer Datenleitung:



Nun können wir fünf verschiedene Stadien unterscheiden, in denen wir die Motoren unterschiedlich stark bewegen können. Überlege dir hierfür geeignete Werte! Und vervollständige das Programm.

3.4 Vermischte Aufgaben

Bei nachfolgenden Aufgaben sollst du nun Motoren und verschiedene Sensoren interagieren lassen.

Für die Bearbeitung der Aufgaben ist die Verwendung von Variablen notwendig. Diese stellen wir dir kurz vor:

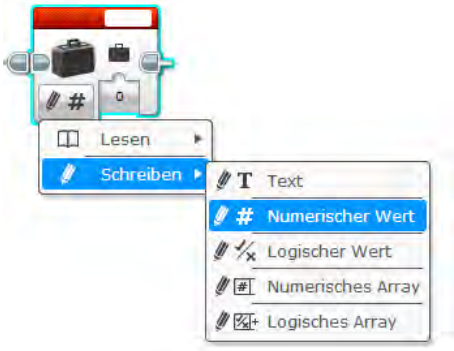
Variablen

Was ist überhaupt eine Variable?

Variablen spielen in der Programmierung[4] eine große Rolle. Einfach formuliert stehen sie für eine Art Behälter, in der man beispielsweise einen numerischen Wert, d.h. eine Zahl, einen logischen Wert, d.h. wahr oder falsch, oder einen Text, z.B. „*hallo*“ etc. ablegen kann. Den Wert der Variable kann man anschließend verändern und auslesen.

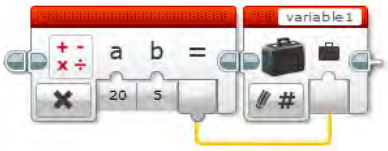
Mit Hilfe folgendes Bausteines kannst du eine neue Variable anlegen:

Durch Klicken auf das weiße Feld rechts oben, kannst du deiner Variable einen Namen zuweisen. Wir können nun entweder den Wert der Variable verändern (Schreiben) oder den Wert der Variable auslesen:



Schreiben

Wir können einer numerischen Variable entweder einen festen Wert zuweisen (links) oder einen Wert über eine Datenleitung festlegen. Dieser Wert kann beispielsweise das Ergebnis einer mathematischen Operation sein (s. rechte Abbildung):

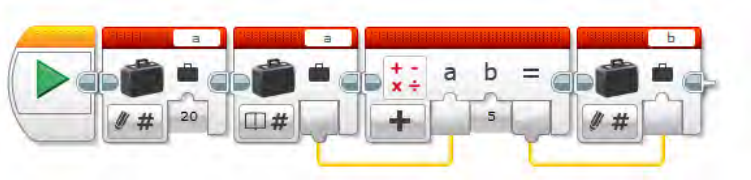


Dasselbe gilt auch für andere Variablentypen, z.B. für logische Variablen.

[4] Variablen nehmen auch in der Mathematik und Physik eine wichtige Rolle ein!

Lesen

Betrachte folgendes Programm und überlege dir, welcher numerische Wert nach Programmausführung in Variable b gespeichert ist:



Betrachten wir das Programm der Reihe nach:

(a) Zunächst wird eine Variable a erstellt und ihr der Wert 20 zugewiesen

(b) Anschließend wird der Wert von Variable a ausgelesen und als erster Wert in der Addition verwendet

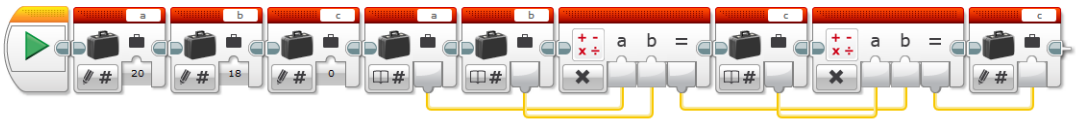
(c) Die Addition lautet somit $20 + 5 = 25$

(d) Das Ergebnis der Addition wird in Variable b geschrieben

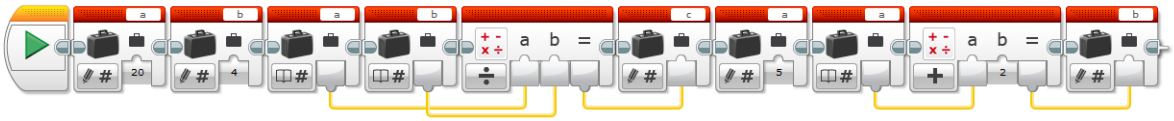
3.4.1 Aufgaben zu Variablen

Welche Werte sind in den Variablen a, b und c nach Ausführung der jeweiligen Programme gespeichert?

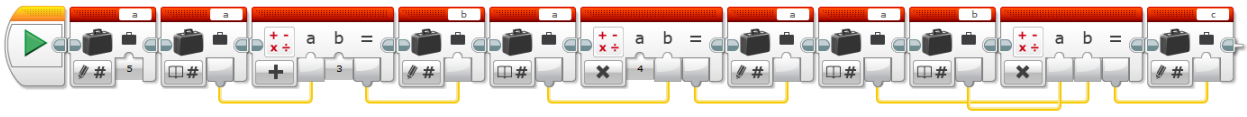
(a) Programm:



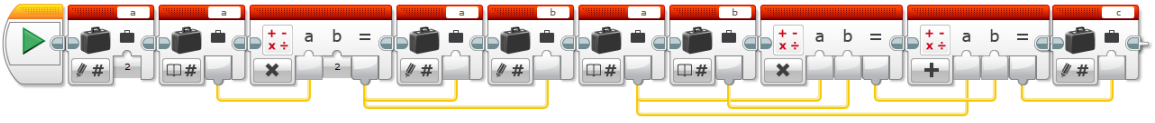
(b) Programm:



(c) Programm:



(d) Programm:



(e) Denke dir selber ein Programm aus und lasse deine Mitschüler die Variablenbelegungen herausfinden!

(f) Erstelle ein Programm, dass den Wert einer Variable a um 1 erhöht!

(g) Erstelle ein Programm, dass den Wert einer Variable a 20-mal hintereinander um 1 erhöht!

(h) Erstelle ein Programm, dass den Wert einer Variable b um den Wert einer Variable a mal hintereinander um 1 erhöht. D.h. hat die Variable a den Wert 15, soll die Variable b 15-mal hintereinander um 1 erhöht werden, hat die Variable a den Wert 101, dann wird b 101-mal um 1 erhöht.

3.4.2 Drucksensor und Ultraschallsensor

Schließe an deinen Roboter einen Berührungssensor und einen Ultraschallsensor an.

Aufgabe: *Das Programm soll zunächst das Drücken des Berührungssensors mitzählen. Misste der Ultraschallsensor einen Wert kleiner 10, soll der Roboter so oft einen bestimmten Ton abspielen, wie zuvor der Berührungssensor gedrückt wurde.*

Mögliches Szenario: Man drückt zunächst 5 mal den Berührungssensor und hält anschließend seine Hand unmittelbar vor den Ultraschallsensor. Dann soll der Roboter fünfmal einen bestimmten Ton abspielen.

Hinweise:

- Für dein Programm benötigst du eine Variable!
- Überlege dir, wie du es schaffst, dass der Roboter genau so oft einen Ton abspielt, wie du zuvor den Berührungssensor gedrückt hast. Stichwort: Wiederholung mit fester Anzahl!
- Warte zwischen den einzelnen Tönen ein bisschen.

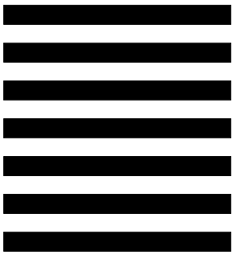
3.4.3 Drucksensor und Farbsensor

Schließe an deinem Roboter zusätzlich noch einen Farbsensor an.

Aufgabe: *Das Programm soll zunächst wieder das Drücken des Berührungssensors mitzählen. Angenommen wir haben den Sensor n mal gedrückt. Erkennt der Farbsensor die Farbe blau, soll der Roboter wieder n mal einen Ton abspielen. Erkennt der Farbsensor die Farbe grün, soll der Roboter um n Umdrehungen vorwärts fahren. Erkennt der Farbsensor die Farbe rot, soll der Roboter um n Umdrehungen nach hinten fahren. Erkennt der Roboter die Farbe gelb, soll er stehen bleiben und n Sekunden schlafen (ändere dazu die Augen auf dem Display).*

3.4.4 Linien zählen

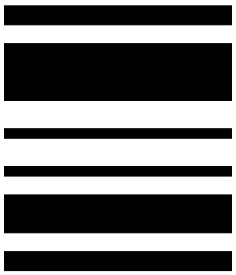
Schließe an deinem Roboter einen Farbsensor an, der nach unten zeigt.



Aufgabe: *Der Roboter soll die schwarzen Linien zählen, über die er fährt. Erstelle dir dazu z.B. mit dem PC ein DinA4-Papier mit schwarzen Strichen (s. rechts). Findet er 3 Sekunden lang keine schwarze Linie mehr, soll er stehen bleiben und so oft einen Ton abspielen, wie er zuvor schwarze Linien erkannt hat.*

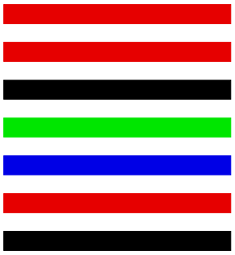
Zusatzaufgabe:

Funktioniert dein Programm auch mit unterschiedlich starken Linien?



3.4.5 Farbige Linien zählen

Wir benötigen wieder, wie bei obiger Aufgabe, einen Lichtsensor, der nach unten zeigt.

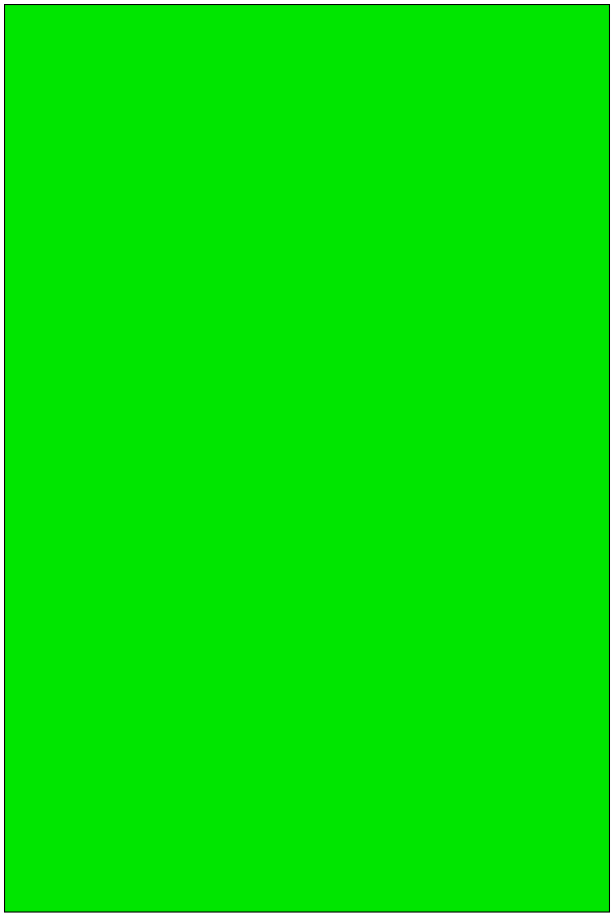


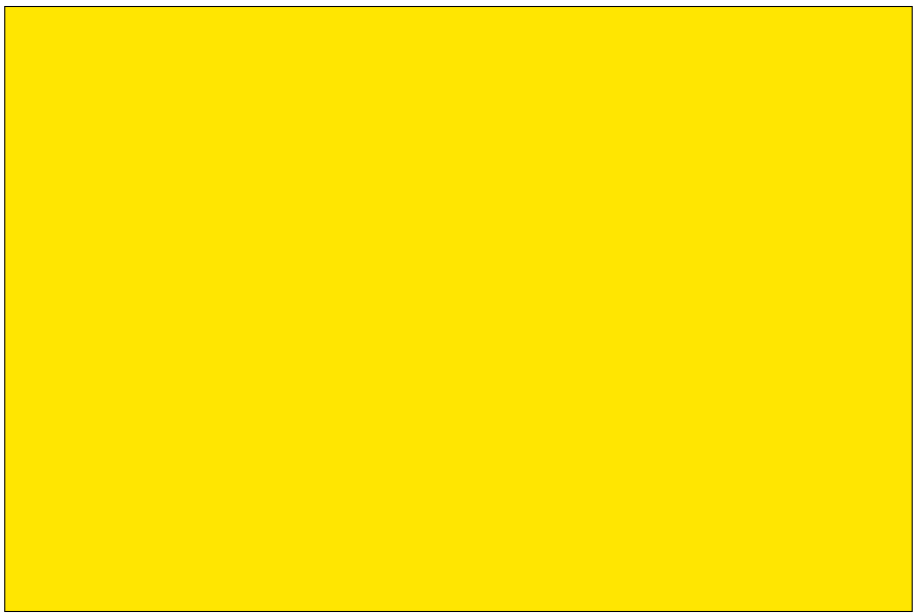
Aufgabe: Nun gibt es verschiedenfarbige Linien! Wir verwenden die Farben schwarz, rot, blau, grün und gelb. Der Roboter soll für jede Farbe mitzählen, wie oft diese vorkommt. Findet er 3 Sekunden lang keine farbige Linie mehr, soll er stehen bleiben und für jede Farbe sagen, wie oft sie vorkommt. Für den linken Fall sollte er folgendes „sagen“:

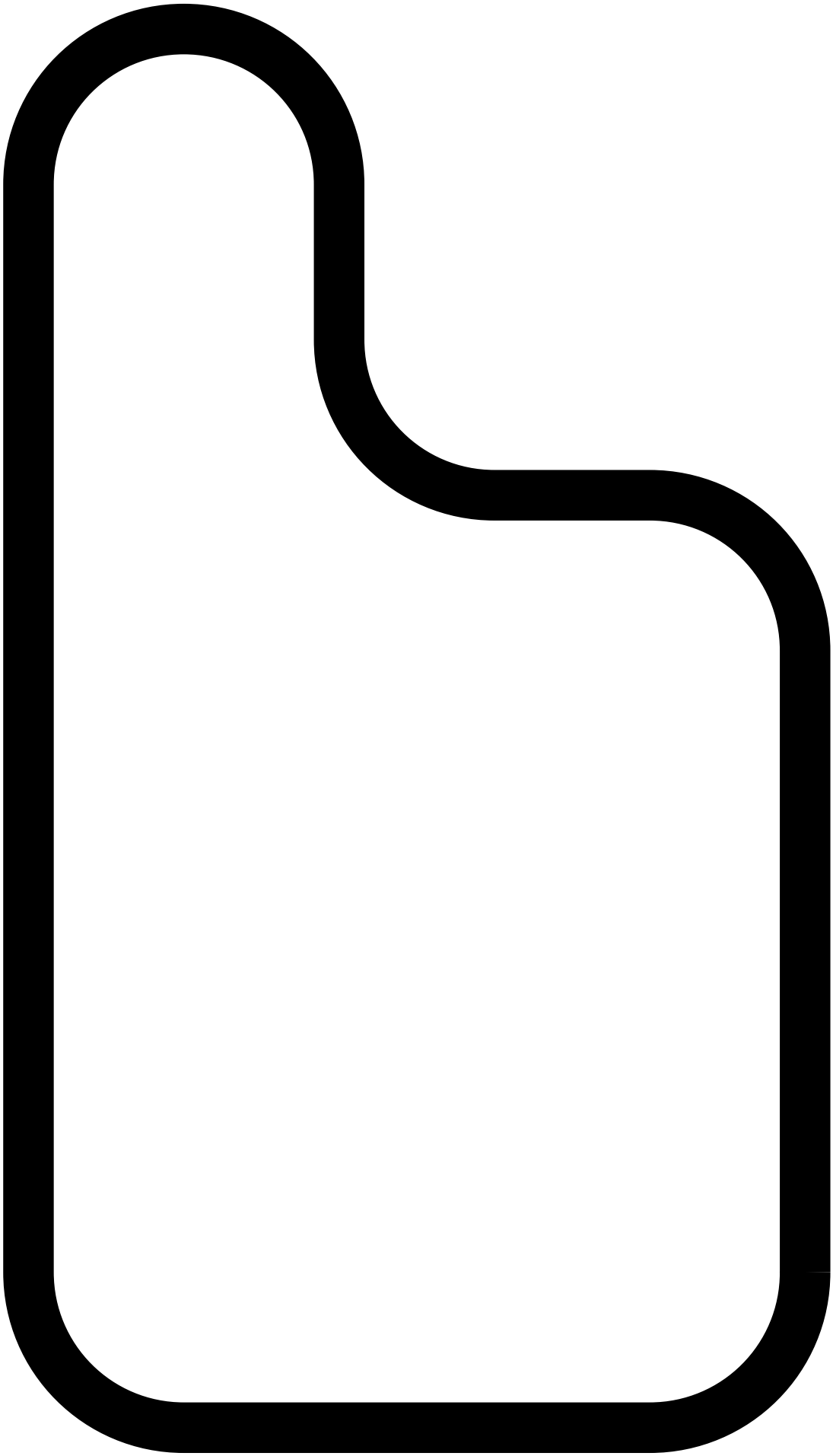
Black Two Red Three Blue One Green One Yellow Zero

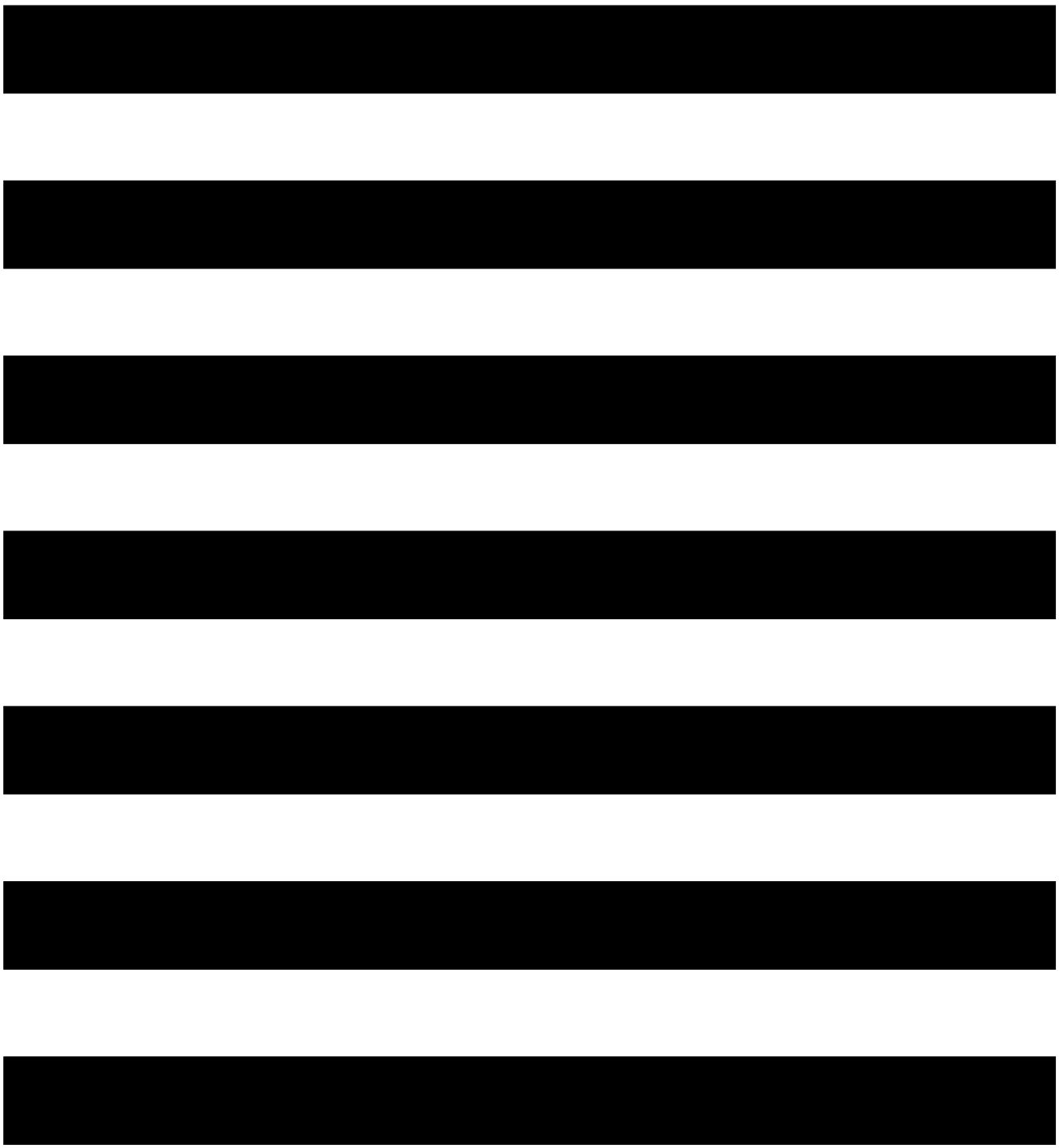
Kapitel 4

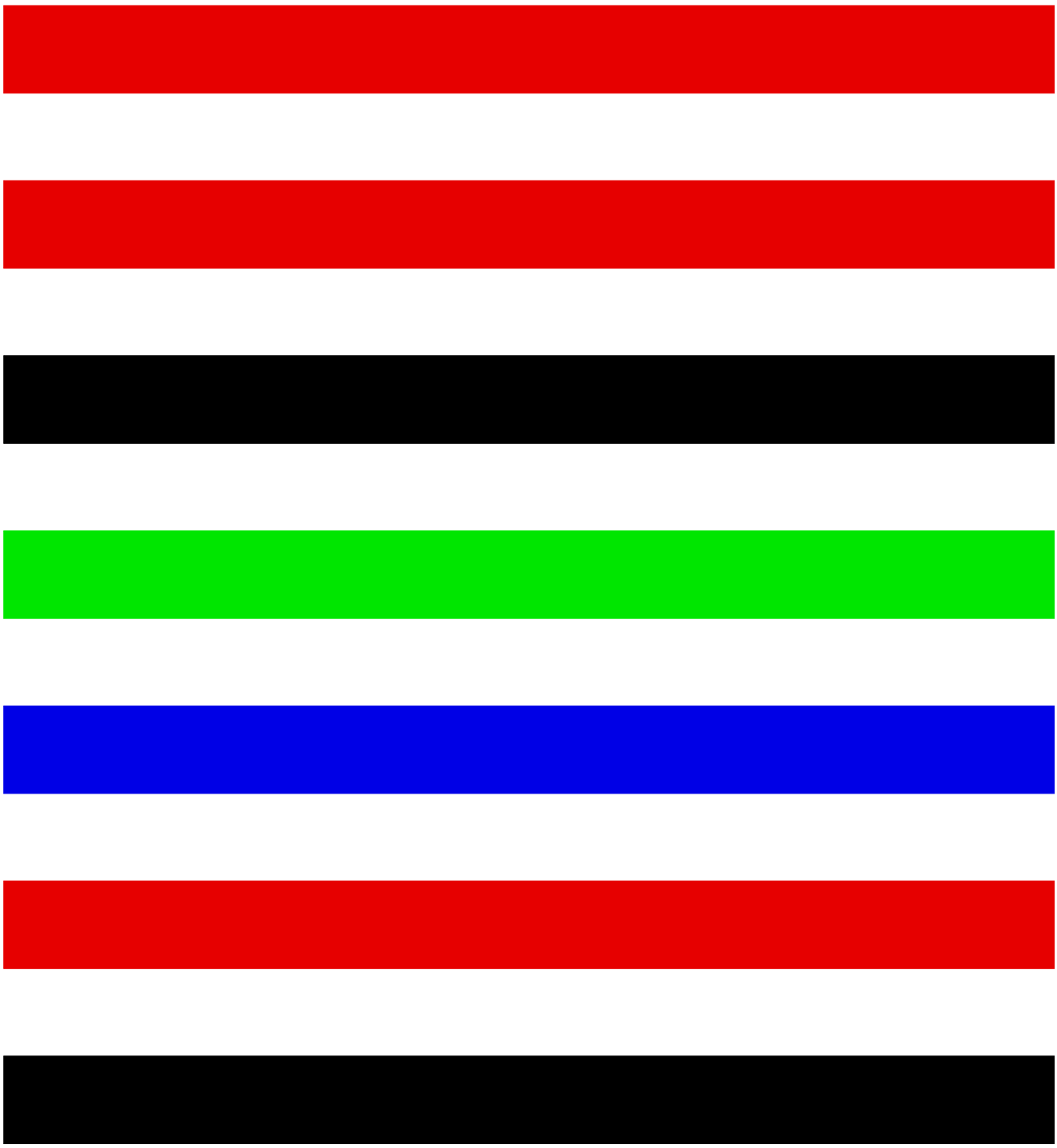
Kopiervorlagen











Kapitel 5

Haftung für Links zu Webseiten

In diesem Mdoul sind Querverweise (Links) zu Webinhalten fremder Betreiber angegeben. Auf die Inhalte dieser Webseiten haben wir keinen Einfluss.

Bei der erstmaligen Angabe haben wir den fremden Inhalt daraufhin überprüft, ob durch ihn eine mögliche zivilrechtliche oder strafrechtliche Verantwortlichkeit ausgelöst wird.
Rechtswidrige Inhalte waren zum Zeitpunkt der Verlinkung nicht erkennbar.

Trotz sorgfältiger inhaltlicher Kontrolle übernehmen wir keine Haftung für die Inhalte externer Links.
Für den Inhalt der verlinkten Seiten sind ausschließlich deren Betreiber verantwortlich.

Passau, im August 2016

Ute Heuer, Wolfgang Pfeffer