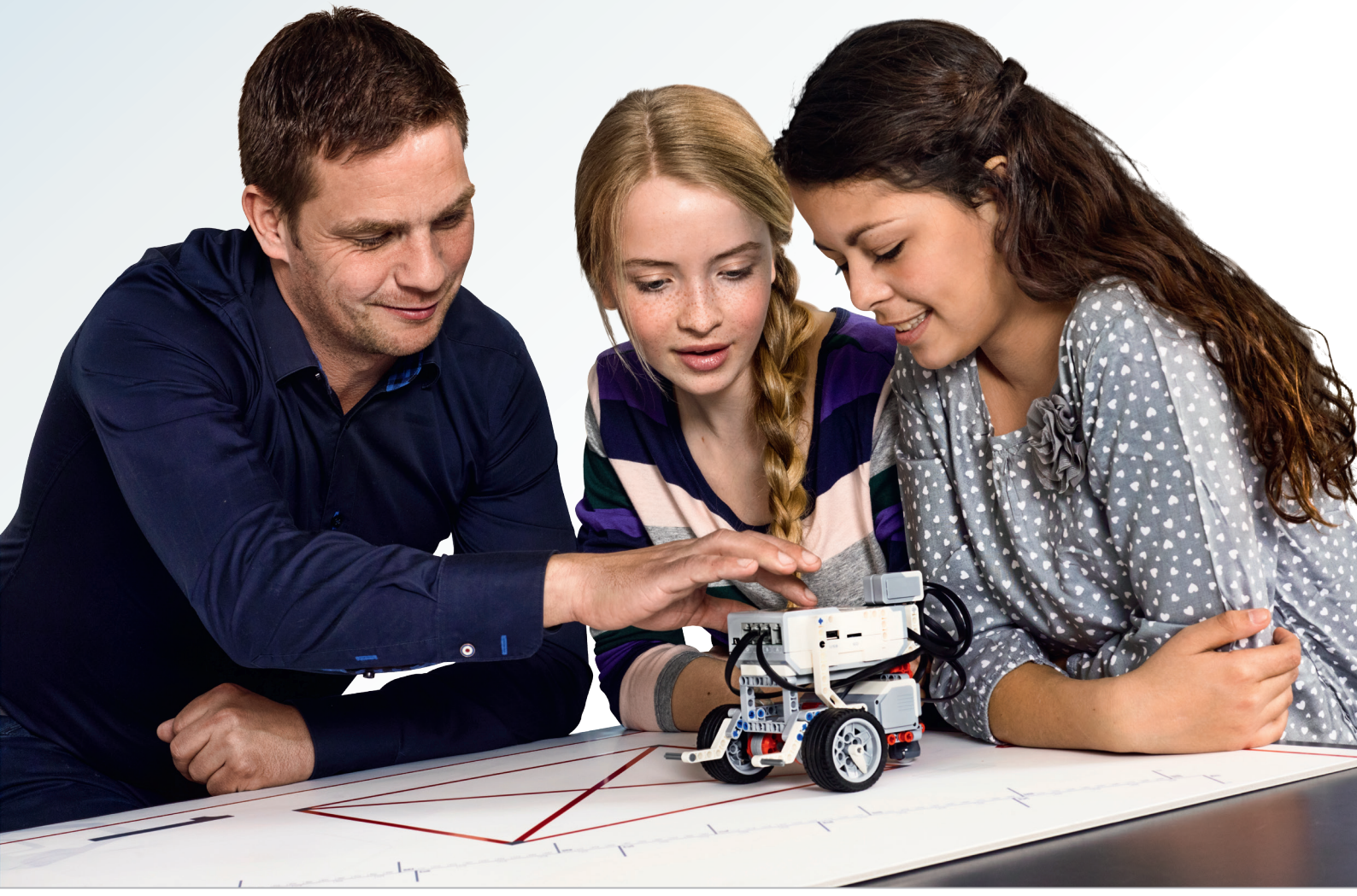
mindstorms education EV3

LEGO® MINDSTORMS® EDUCATION EV3

UNTERRICHTSKONZEPTE FÜR MINT-FÄCHER MIT SCHWERPUNKT PROGRAMMIEREN UND INFORMATIK



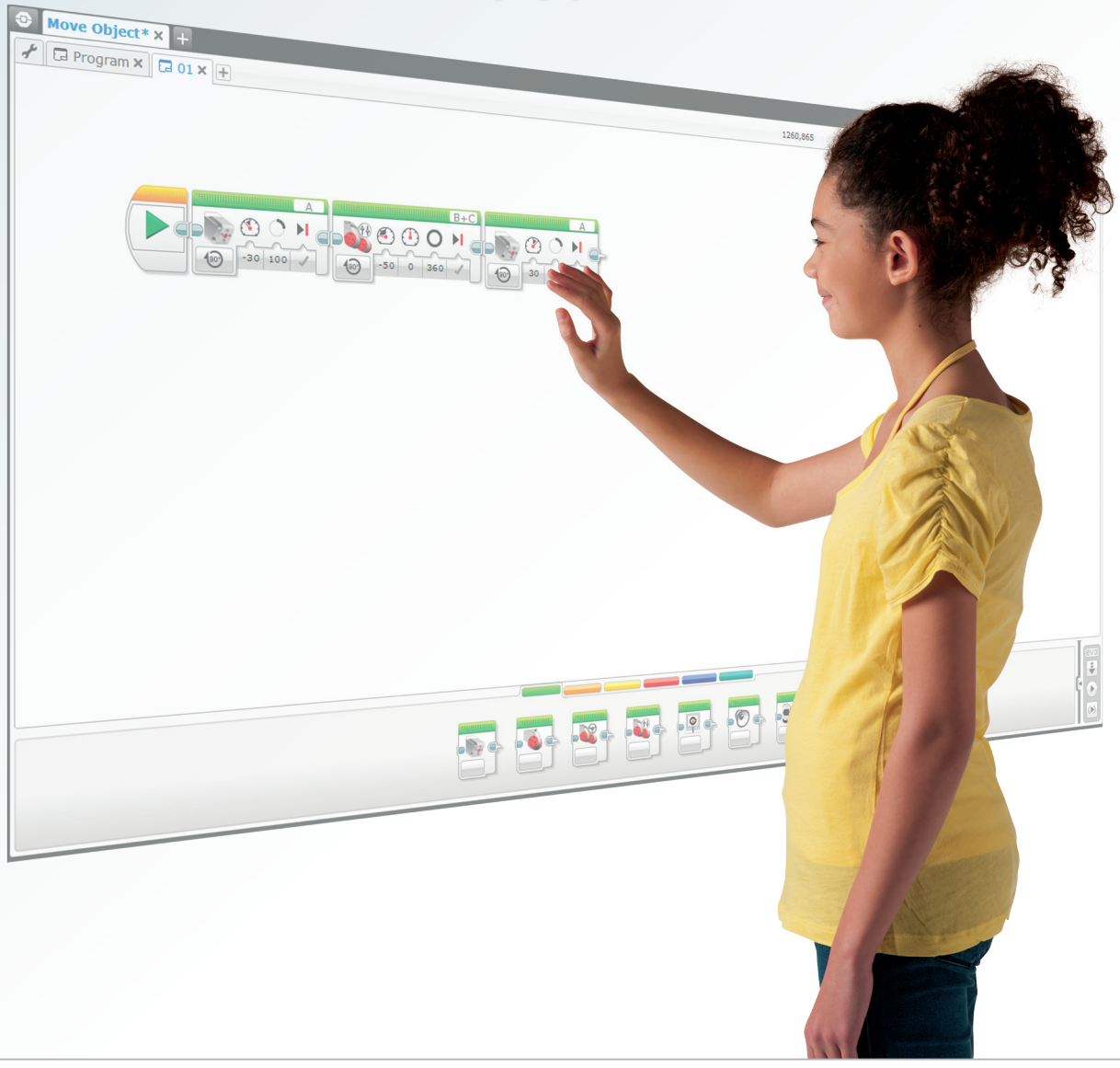
LEGOeducation.de

LEGO education

mindstorms education EV3

EINFÜHRUNG

LEGO® MINDSTORMS® Education EV3 Unterrichtskonzepte für MINT-Fächer mit Schwerpunkt Informatik und Programmieren



LEGOeducation.de

education

LEGO® MINDSTORMS® Education EV3 Unterrichtskonzepte für MINT-Fächer mit Schwerpunkt Informatik/ Programmieren

Wir von LEGO® Education freuen uns, Ihnen diese Unterrichtskonzepte für MINT-Fächer präsentieren zu können. Diese vorgefertigten Unterrichtseinheiten sind für den Einsatz an weiterführenden Schulen in der Altersklasse ab 11 Jahren vorgesehen.

Die hier vorgelegten Materialien unterstützen Lehrkräfte bei der Durchführung spannender Computerprojekte, die auf maßgeblichen Technologien der heutigen Zeit basieren und den Schülern die Möglichkeit geben, sich grundlegende Gedanken aus Informationstechnik und Ingenieurwissenschaften in einem realistischen Kontext anzueignen.

– Hintergrund zu diesen Unterrichtskonzepten
– Zielgruppe
– Die 12 Unterrichtseinheiten: Überblick und Lehrplanbezüge
– Gebrauch der Unterrichtskonzepte
  - **Organisation der Unterrichtseinheiten**
  - **Roboter Educator**
  - **Nutzung des Internets inkl. Video-Plattformen**
  - **Textbasiertes Programmieren**

HINTERGRUND ZU DIESEN UNTERRICHTSKONZEPTEN

Der fachmännische Einsatz von Computern und die Fähigkeit zum Programmieren sind heutzutage wichtige Grundlagen für den Erfolg in Ausbildung und Beruf. Der Bedarf an der Vermittlung dieser Fertigkeiten in der Schule wird damit kontinuierlich größer. Deshalb hat man bei LEGO Education diese Unterrichtskonzepte entwickelt, die den Schülern unter Einsatz des MINDSTORMS Education EV3 dabei helfen, dieses abstrakte Thema anzupacken. Sie umfassen 12 Unterrichtseinheiten, die - je nach Organisation - rund 36 Stunden gezielter Arbeit im Klassenzimmer entsprechen.

ZIELGRUPPE

Die Aufgaben dieser Unterrichtskonzepte helfen Lehrkräften dabei, ihre Schüler zum Nachdenken über die Bedeutung der Computer-Programmierung für das tägliche Leben zu bringen. Die Schüler gewinnen durch eine Kombination von direkter Lehre, Forschen und Experimentieren sowie Anleitungen der LEGO MINDSTORMS Education EV3 Software Programmiererfahrung. Beispielhafte Lösungen werden ebenfalls bereitgestellt. Die Materialien richten sich an Schüler weiterführender Schulen ab einem Alter von 11 Jahren, die Unterrichtskonzepte können aber auch problemlos an den Einsatz in höheren Jahrgangsstufen angepasst werden. Obwohl die Inhalte mit Fokus auf Computereinsatz und Informationstechnologie entwickelt wurden, gibt es lehrplanübergreifende Ansätze in andere Fachbereiche wie Naturwissenschaften, Mathematik und Technik.

Die hier vorgelegten Stundenplanungen reduzieren die Vorarbeit des Lehrers beträchtlich, mit zunehmender Erfahrung lassen sich problemlos weiterführende Unterrichtseinheiten entwerfen.

Am Ende jeder Einheit befinden sich Seiten mit beispielhaften Bildern und Programmen der jeweiligen Lektionen. Diese können an die Schüler ausgegeben werden.

Viel Spaß und Erfolg!

DIE 12 UNTERRICHTSEINHEITEN: ÜBERBLICK UND LEHRPLANBEZÜGE

Unterr.-Einheit	Aufgabe	Lehrplanbezüge / LEGO® MINDSTORMS® Education Kompetenzen	Behandelte Blöcke der EV3-Software
1	Wenden in drei Zügen Einführung in die Arbeit mit Computern sowie EV3-Hard- und -Software.	Gebrauch von zwei oder mehr Programmiersprachen (davon wenigstens eine textbasiert) zur rechnergestützten Problemlösung.	– Hebelsteuerung – Warten – Ultraschallsensor – Klang
2	Wenden in drei Zügen (textbasierte Programmierung)	Gebrauch von zwei oder mehr Programmiersprachen (davon wenigstens eine textbasiert) zur rechnergestützten Problemlösung.	– Hebelsteuerung – Warten – Ultraschallsensor – Klang
3	Roboter im Rückwärtsgang Verwendung der Blöcke Stein-Anzeige und Stein-Statusleuchte. Warnlichter bei Autos.	Verstehen, dass Algorithmen eine Serie von Befehlen in einer bestimmen Reihenfolge ausführen können. Einsatz des Standardsteuerung-Blocks, um einen Rad-Roboter geradeaus zu bewegen. Einsatz des Warten-Blocks in Verbindung mit Berührungssensoren. Nutzung von Stein-Statusleuchte und Stein-Anzeigefunktionen. Ausbau von Programmierfertigkeiten durch die Entwicklung komplexerer Algorithmen.	– Standardsteuerung – Warten – Berührungssensor

Unterr.-Einheit	Aufgabe	Lehrplanbezüge / LEGO® MINDSTORMS® Education Kompetenzen	Behandelte Blöcke der EV3-Software
4	Mit Licht den Weg weisen - automatische Scheinwerfer Farbsensor. Umgebungslicht-Einstellungen.	Verstehen diverser Schlüssel-Rechenverfahren, die algorithmische Denkweise widerspiegeln. Verstehen einfacher Boole'scher Logik (logischer Operatoren wie UND, ODER und NICHT) sowie einige ihrer Anwendungsgebiete in Schaltkreisen und beim Programmieren.	– Warten – Farbsensor – Anzeige – Zeit – Schleife – Berührungssensor – Schleifen-Interrupt
5	Ampeln und automatische Schienensysteme Folgen einer Linie. Automatisiertes Fahrzeug.	Verstehen, dass Algorithmen eine Serie von Befehlen in einer bestimmen Reihenfolge ausführen können. Vertiefung des Verständnisses Boole'scher Logik und ihrer Einsatzmöglichkeiten. Einsatz des Warten-Blocks in Verbindung mit dem Farbsensor. Verstehen, dass der Farbsensor mehrere Funktionen hat und eine Reihe von Parametern messen und auf sie reagieren kann. Verwendung des Farbsensors zur Erkennung von LEGO®-Systemfarben und Messung der Stärke reflektierten Lichts. Tieferes Verständnis des Schleifen-Blocks. Begreifen des Konzepts eines Schalters und wie dieser für 'Wahr'- und 'Falsch'-Operationen verwendet wird.	– Warten – Standardsteuerung – Farbsensor – Schleife – Schalter – Schleifen-Interrupt
6	Es piept beim Rückwärtsfahren Ultraschallsensor. Hilfsmittel zur Objekterfassung. Sensoren beim Rückwärtsfahren.	Verstehen, dass Algorithmen eine Serie von Befehlen in einer bestimmen Reihenfolge ausführen können. Vertiefung des Verständnisses Boole'scher Logik und ihrer Einsatzmöglichkeiten. Einsatz des Warten-Blocks in Verbindung mit dem Farbsensor. Verstehen, dass der Ultraschallsensor mit Schallwellen arbeitet, die von Objekten zurückgeworfen werden, und dass er zur Reaktion auf bestimmte Entfernungen programmiert werden kann. Einen Rad-Roboter so programmieren, dass er rückwärtsfährt, ein Geräusch abhängig von der Entfernung zu einem Objekt ausgibt und in einer bestimmten Entfernung zu diesem Objekt anhält. Tieferes Verständnis des Schleifen-Blocks. Begreifen des Konzepts eines Schalters und wie dieser für 'Wahr'- und 'Falsch'-Befehle verwendet wird. Verständnis des Mathe-Blocks und seiner Funktionen. Verstehen, dass Werte über Datenleitungen von einem Block zu einem anderen übertragen werden können.	– Standardsteuerung – Warten – Ultraschallsensor – Schleife – Mathe – Klang

Unterr.-Einheit	Aufgabe	Lehrplanbezüge / LEGO® MINDSTORMS® Education Kompetenzen	Behandelte Blöcke der EV3-Software
7	Schlüsselloses Startsystem.	Verstehen diverser Schlüssel-Rechenverfahren, die algorithmische Denkweise widerspiegeln. Verstehen einfacher Boole'scher Logik (logischer Operatoren wie UND, ODER und NICHT) sowie einige ihrer Anwendungsgebiete in Schaltkreisen und beim Programmieren. Einsatz des Logik-Blocks in Verbindung mit dem Schalter-Block. Verwendung mehrerer Sensoren in Kombination, um ein Programm auf dem EV3-Stein zu aktivieren.	– Warten – Berührungssensor – Ultraschallsensor – Anzeige – Zeit – Stein-Tasten – Logik – Schalter – Schleife – Standardsteuerung
8	Entwicklung einer Geschwindigkeitsregelanlage.	Verstehen diverser Schlüssel-Rechenverfahren, die algorithmische Denkweise widerspiegeln. Einsatz des Variable-Blocks zur Speicherung von Information. Entwicklung mehrstufiger Programme. Entwerfen eigener Blöcke.	– Warten – Berührungssensor – Schleife – Schalter – Variable – Mathe – Standardsteuerung – Eigene Blöcke
9	Streunende Roboter Wie Arrays funktionieren.	Sinnvoller Einsatz von Datenstrukturen wie Listen, Tabellen und Arrays. Verstehen einfacher Boole'scher Logik (logischer Operatoren wie UND, ODER und NICHT) sowie einige ihrer Anwendungsgebiete in Schaltkreisen und beim Programmieren. Einsatz der Stein-Tasten zur Bewegungssteuerung des Rad-Roboters. Einsatz des Variable-Blocks zur Speicherung von Information. Einsatz des Array-Operationen-Blocks.	– Variable – Warten – Stein-Tasten – Schleife – Klang – Array-Operationen – Zeit – Standardsteuerung – Eigene Blöcke
10	Entwicklung eines führerlosen, automatisierten Rad-Roboters.	Abbildung realer Problemstellungen und physikalischer Systeme durch Entwicklung, Gebrauch und Bewertung rechnergestützter Abstraktionen.	Alle Programmier-Blöcke der vorangegangenen Einheiten können verwendet werden.
11	Bau und Programmierung eines führerlosen Fahrzeugs.	Abbildung realer Problemstellungen und physikalischer Systeme durch Entwicklung, Gebrauch und Bewertung rechnergestützter Abstraktionen.	Alle Programmier-Blöcke der vorangegangenen Einheiten können verwendet werden.
12	Überprüfung und Veränderung eines führerlosen, automatisierten Roboters.	Abbildung realer Problemstellungen und physikalischer Systeme durch Entwicklung, Gebrauch und Bewertung rechnergestützter Abstraktionen.	Alle Programmier-Blöcke der vorangegangenen Einheiten können verwendet werden.

Gebrauch der Unterrichtskonzepte

ORGANISATION DER UNTERRICHTSEINHEITEN

Gewöhnlich müssen in jeder Unterrichtseinheit drei Hauptaufgaben erfüllt werden. Die Stundenplanungen enthalten Lösungsvorschläge zu jeder Aufgabe, zudem sind die entsprechenden EV3-Programme als Download verfügbar. Die Unterrichtseinheiten, in denen üblicherweise das Robot Educator-Modell eingesetzt wird, sollten alle notwendigen Bauarbeiten miteinschließen. Es ist sinnvoll, den Schülern während der Einheiten 1 bis 9 genug Zeit zu geben, Erfahrung im Modellbauen zu sammeln. So können sie letztlich gut vorbereitet die finalen drei Einheiten angehen und sind der abschließenden Konstruktionsaufgabe gewachsen.

Jede Aufgabe sollte eine Phase der Diskussion mit den Schülern über die Struktur und das Design des Programms beinhalten.

ROBOT EDUCATOR

Jede Unterrichtseinheit setzt von den Schülern eine gute Kenntnis der EV3-Software voraus. Diese kann durch eine erfolgreiche Beschäftigung mit den Robot Educator-Übungen am Ende jeder Stundenplanung (siehe ' Anmerkungen für die Lehrkraft') erworben werden. Flankiert von direkter Unterweisung und selbstständiger Beschäftigung sorgen diese Übungen dafür, dass sich die Schüler die Fähigkeiten und das Verständnis aneignen, die zum Bestehen der Herausforderungen der einzelnen Einheiten notwendig sind.

Bei späteren Unterrichtseinheiten sind die Robot Educator-Übungen eingeteilt in 'Neue Robot Educator-Übungen' sowie bereits zuvor behandelte Übungen, die jedoch für die aktuelle Einheit relevant sind.

NUTZUNG DES INTERNETS INKL. VIDEO-PLATTFORMEN

In diesen Unterrichtskonzepten befinden sich keine direkten Links zu Suchmaschinen oder Video-Plattformen, da diese im Lauf der Zeit obsolet werden. Zudem gibt es in manchen Ländern Beschränkungen von Video-Plattformen, oder der Zugriff auf sie ist sogar vollständig blockiert.

Um dieser Problematik gerecht zu werden, finden sich in diesen Materialien Schlüsselwörter, die Sie in Ihre bevorzugte Suchmaschine eingeben können, um auf die entsprechenden Inhalte zu stoßen.

Ein Beispiel:

Um in die Unterrichtskonzepte einzusteigen wird empfohlen, ein aufschlussreiches Video zu BMWs fahrerlosem Auto anzusehen, das ursprünglich in der BBC-Sendung "Top Gear" ausgestrahlt wurde. Um dieses Video zu finden, tippen Sie folgende Wörter in eine Suchmaschine bzw. die Suchfunktion einer Video-Plattform:

Schlüsselwörter: **BMW, fahrerlos, Auto, Top Gear**

Mit diesen vier Begriffen finden Sie das Video und können das fahrerlose Auto in Aktion erleben.

TEXTBASIERTES PROGRAMMIEREN

In dieser ersten Ausgabe der Unterrichtskonzepte hat man sich bei LEGO® Education für ROBOTC als textbasierte Programmiersprache entschieden.

Es ist allerdings nicht die Absicht dieser Konzepte, Lehrer und Schüler im Gebrauch einer textbasierten Programmiersprache zu unterrichten. Es geht vielmehr um das Begreifen von EV3-Programmen als visuelle Äquivalente zu textbasierten Lösungen. Alles, was Sie über ROBOTC wissen müssen, finden Sie online unter www.robotc.net.

Inhalt

mindstorms education EV3

UNTERRICHTSEINHEIT 1

Wenden in drei Zügen

LEGOeducation.de

LEGO education

Wenden in drei Zügen

EINFÜHRUNG IN DIE ARBEIT MIT COMPUTERN, EV3 UND MINDSTORMS

In dieser Einheit werden die Schüler mit der Arbeit an Computern und dem Konzept der Programmierung vertraut gemacht. Hierbei und in den folgenden Unterrichtseinheiten erkennen sie die Bedeutung der Computerprogrammierung für das tägliche Leben.

Jede Einheit baut auf zuvor erworbenen Kenntnissen und Erfahrungen auf und stattet die Schüler mit den nötigen Fähigkeiten aus, ein führerloses, automatisiertes Fahrzeug zu entwickeln - einen Rad-Roboter, der ohne Fahrer von Punkt A nach Punkt B kommt.

Um die Unterrichtseinheiten in einen Kontext zu setzen, ist es sinnvoll, Diskussionen unter den Schülern anzuregen. Mögliche Fragen wären:

- Welche besonderen Merkmale sind nach Meinung der Schüler für ein automatisiertes Fahrzeug notwendig? Diskutieren Sie den Einsatz von Sensoren zur Sicherheit und Navigation.
- Was zeichnet ein automatisiertes Fahrzeug sonst noch aus?
- Welche automatisierten Systeme gibt es im Auto der Eltern? Dies könnten beispielsweise eine Einparkhilfe oder Abstands- bzw. Geschwindigkeitsregelsysteme sein.

Anmerkung: Vielleicht planen Sie auch den Besuch einer Autofabrik oder eines Händlers ein, um mehr über automatisierte Funktionen von Fahrzeugen zu erfahren.

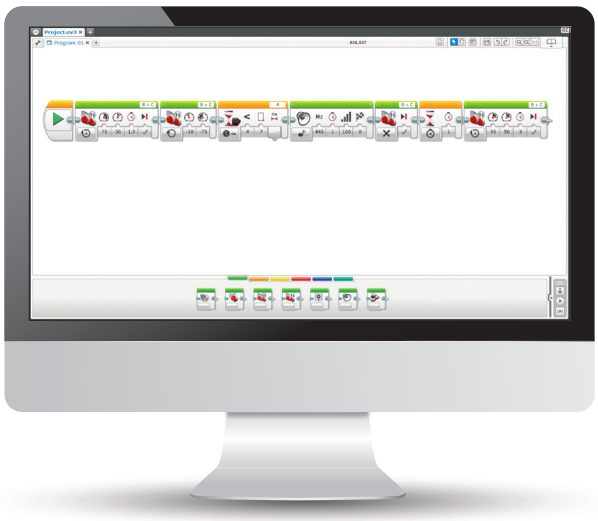
EINFÜHRUNG DES EV3 UND DER MINDSTORMS EDUCATION EV3-SOFTWARE

Für den Erfolg dieses Unterrichtskonzepts ist es wichtig, zunächst ausreichend Zeit in den Bau des Robot Educator-Grundmodells zu investieren.

Die Einheit 1 beschäftigt sich hauptsächlich mit dem Einsatz von Motoren und der Entwicklung eines Algorithmus, der den Rad-Roboter eine 'Wende in drei Zügen' durchführen lässt.

Erläutern Sie den Schülern, dass eine Wende in drei Zügen im Autoverkehr dazu dient, eine 180°-Wende durchzuführen.

Suchen Sie nach Online-Videos, in denen Wendemanöver in drei Zügen bzw. das Umkehren auf einer Straße zu sehen sind. Diese könnten als nützliche Referenz für Ihre Schüler dienen.



STUNDENPLANUNG

Diese Unterrichtseinheit umfasst drei Hauptaufgaben. Lösungsvorschläge sind bei den Erläuterungen der Aufgaben sowie im Anhang der Einheit zu finden.

ERGEBNISSE

In dieser Einheit lernen die Schüler,

- dass Algorithmen eine Serie von Befehlen in einer bestimmten Reihenfolge ausführen können.
- wie der Hebelsteuerung-Block funktioniert und sich zur Lenkung ihres Rad-Roboters einsetzen lässt.
- einen einfachen Rad-Roboter zu bauen und zu programmieren, wobei sie Hebelsteuerung-Blöcke und entsprechendes Timing nutzen, um eine 180°-Wende in drei Zügen auszuführen.
- mit dem Warten-Kommando und dem Ultraschallsensor-Block umzugehen..

BEGRIFFE

Eingabe, Ausgabe, Algorithmus, Warte, Ultraschallsensor, debuggen.

EINFÜHRUNG

- Erläutern Sie den Schülern, dass sie im Lauf dieser Unterrichtseinheit einen Rad-Roboter bauen und so programmieren werden, dass er eine Wende in drei Zügen ausführt. Zunächst werden sie allerdings etwas Zeit in die Beschäftigung mit der EV3-Software investieren, mit der die Bewegung der Modelle gesteuert wird.
- Präsentieren Sie Ihren Schülern ein vollständiges EV3-Grundmodell. Die Schüler sollen ein entsprechendes Modell zusammenbauen und sich dabei an der in der EV3-Software enthaltenen Anleitung orientieren. Diese ist im Robot Educator-Menü zu finden.
- Zeigen Sie Ihren Schülern, wie man auf die EV3-Software zugreift und wie der Hebelsteuerung-Block verwenden wird, um den Rad-Roboter in Bewegung zu versetzen.

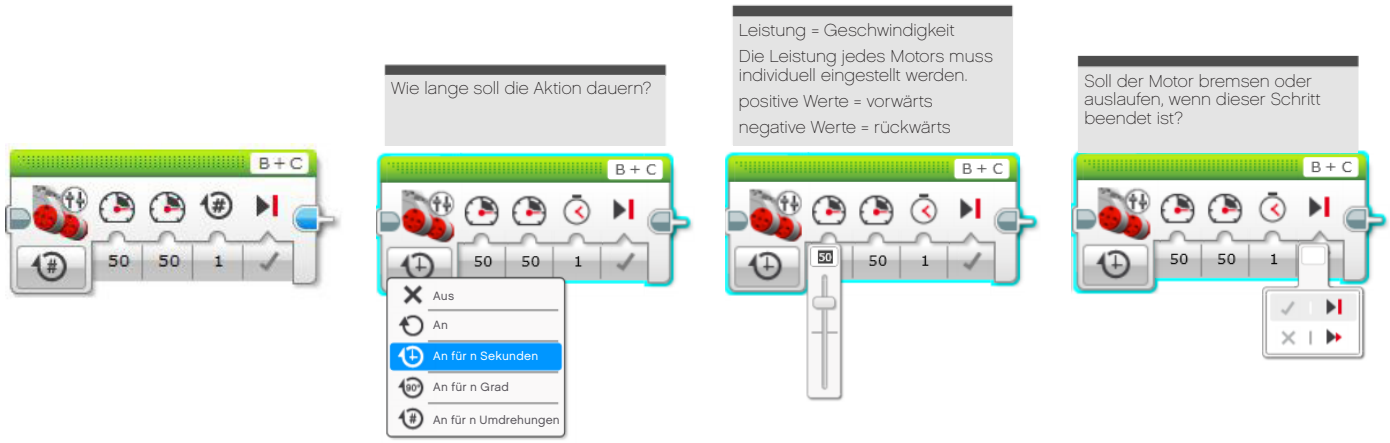
Bild 1

Bild 2


Bild 3

EINFÜHRUNGSAUFGABE

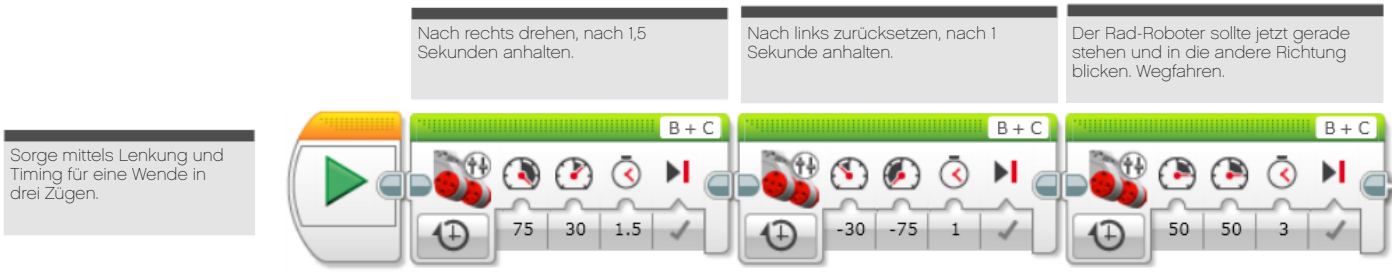
- Die Schüler arbeiten zu zweit oder zu dritt an der Konstruktion des Grundmodells.
- Anschließend starten sie in der Software ein neues Projekt und experimentieren mit dem Hebelsteuerung-Block.
- Ermuntern Sie die Schüler, unterschiedliche Möglichkeiten herauszufinden, den Rad-Roboter Kurven fahren zu lassen. Welchen Effekt hat die Veränderung der Leistung der einzelnen Motoren?



HAUPTAUFGABE 1

- Sammeln Sie Feedback der Schüler zu den unterschiedlichen Methoden, wie man den Rad-Roboter mittels Hebelsteuerung-Block Kurven fahren lässt. Dabei soll auch erläutert werden, wie sie auf die jeweilige Methode gekommen sind.
- Sagen Sie den Schülern, dass sie ihre Erkenntnisse nun nutzen sollen, um eine Wende in drei Zügen zu simulieren.
- Dies wäre ein guter Zeitpunkt, den Schülern anhand eines Online-Videos zu zeigen, wie eine Wende in drei Zügen ausgeführt wird.
- Die Schüler verwenden nun die EV3-Software, um ein Programm zum Wenden in drei Zügen zu entwickeln. Dabei nutzen sie unterschiedliche Leistungseinstellungen der beiden Motoren.
- Fordern Sie Ihre Schüler auf, das Programm kontinuierlich zu testen und zu debuggen (also Fehler auszumerzen), um zu gewährleisten, dass der Rad-Roboter eine 180°-Wende macht.
- Anmerkung: Sie können mit Klebeband eine Straße abstecken, um die Schüler mit einer eingeschränkten Fläche zu konfrontieren, innerhalb der der Rad-Roboter wenden muss.
- Anmerkung: Im Lösungsvorschlag steht, dass der Rad-Roboter am Ende 'gerade stehen sollte'. Sie sollten sinnvollerweise anmerken, dass das in der Realität nicht immer der Fall ist und die Schüler mitunter vor der abschließenden Geraden eine Kurve einbauen. Stellen Sie klar, dass es bei den Aufgaben keinen falschen Lösungsweg gibt.

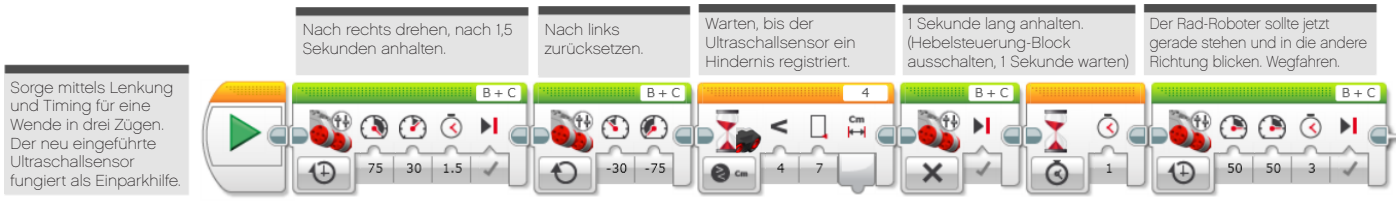
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 1
REITER MAIN 1



HAUPTAUFGABE 2

- Führen Sie den Ultraschallsensor in den Unterricht ein.
- Fragen Sie die Schüler, wie sich das Programm eigenständiger machen ließe und alle Hindernisse erfasst werden könnten, die womöglich während der Rückwärtsfahrt des Rad-Roboters auftauchen.
- Demonstrieren Sie den Warten-Block und seinen Einsatz in Verbindung mit dem Ultraschallsensor.
- Die Schüler sollen ihr Programm so erweitern, dass der Rad-Roboter zu jedem beliebigen Zeitpunkt auf den Ultraschallsensor reagiert und anhält.
- Machen Sie den Schülern klar, dass ihre Programme einen Autofahrer simulieren, der auf die Bremse tritt, wenn er beim Rückwärtsfahren einem Hindernis zu nahe kommt.

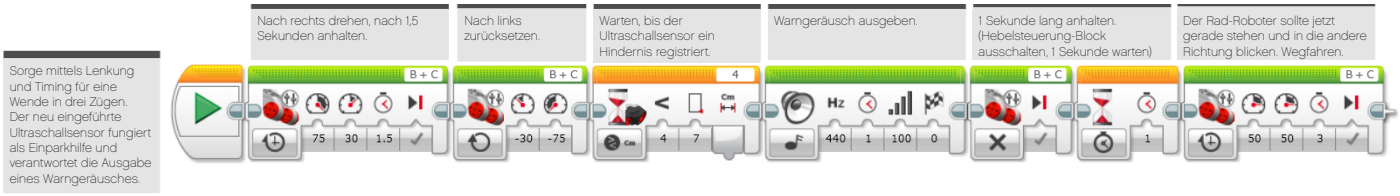
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 1
REITER MAIN 2



HAUPTAUFGABE 3

- Nun wird der Ultraschallsensor eingesetzt, um dem Rad-Roboter Sicherheitsfunktionen (wie Warngeräusche) hinzuzufügen.
- Fragen Sie die Schüler, was bei vielen modernen Autos passiert, wenn diese zurücksetzen und sich einem Hindernis nähern (ein Warngeräusch ist zu hören).
- Führen Sie den Klang-Block ein. Er dient dazu, vor dem Anhalten des Rad-Roboters ein Warngeräusch auszugeben. Zeigen Sie den Schülern, wie der Klang-Block funktioniert und wo im Programm er eingefügt werden sollte.
- Die Schüler müssen ihr bestehendes Programm durch Einfügen des Klang-Blocks abändern. Er soll ein Warngeräusch ausgeben, wenn sich der Rad-Roboter in einer bestimmten Entfernung zu einem Hindernis befindet.
- Fordern Sie die Schüler auf, ihre Programme durchgehend zu debuggen, um die Geräuschausgabe und die Bremsdistanz zu optimieren.

LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 1
REITER MAIN 3



ABSCHLIESSENDE DISKUSSION

- Rekapitulieren Sie, was die Schüler in dieser Unterrichtseinheit gelernt haben.
- Sorgen Sie dafür, dass die Schüler die Begriffe, die in dieser Unterrichtseinheit verwendet wurden, richtig einsetzen und verstehen. Wiederholen Sie die Bedeutung der Schlüsselbegriffe.
- Lassen Sie ein oder zwei Gruppen ihre Programme demonstrieren. Diskutieren Sie darüber, was an den Programmen gut funktioniert und wo man sie verbessern könnte.
- Eröffnen Sie den Schülern, dass sie in der Folgewoche dieselben Aufgaben durchführen werden. Dabei verwenden sie allerdings nicht die EV3-Software, sondern eine textbasierte Programmierlösung.

AUFGABEN DIESER UNTERRICHTSEINHEIT

Die heutigen Aufgaben sollen dich mit der LEGO MINDSTORMS Education EV3-Software vertraut machen.

Du hattest bereits ein wenig Zeit, mit dem Hebelsteuerung-Block zu experimentieren und durch seinen Einsatz deinen Rad-Roboter in Bewegung zu versetzen. Nun wirst du an deinen Fähigkeiten feilen müssen, um die nachfolgenden drei Aufgaben zu meistern.

Viel Erfolg!

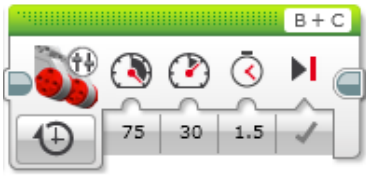
AUFGABE 1

Programmiere deinen Rad-Roboter so, dass dieser eine Wende in drei Zügen ausführt.

Du musst deinen Rad-Roboter während der Vorwärtsfahrt drehen. Dann setzt du mit ihm zurück, bevor er nach vorne weiterfährt.

Sieh dir noch einmal das Online-Video an, um dir die Wende in drei Zügen in Erinnerung zu rufen, und achte auf die Straßenmarkierungen.

Geeignete Blöcke



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 2

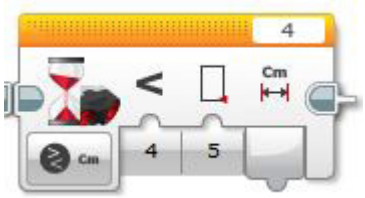
Jetzt wirst du mit einem der EV3-Sensoren experimentieren - dem Ultraschallsensor.

Programmiere deinen Rad-Roboter so, dass er eine Wende in drei Zügen vollführt, und setze den Ultraschallsensor als 'Einparkhilfe' ein. Dein Rad-Roboter soll beim Rückwärtsfahren in einer bestimmten Entfernung zu einem Hindernis anhalten.

Kann dein Rad-Roboter 'in die Bremsen steigen', bevor er wieder davonfährt?

Hier ist dein Wissen über den Warten-Block gefragt. Zudem solltest du den Ultraschallsensor an der Rückseite deines Rad-Roboters anbringen.

Geeignete Blöcke



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 3

Simulation von Warngeräuschen.

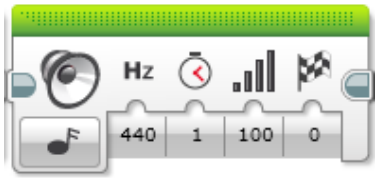
Was passiert bei vielen modernen Autos, wenn diese rückwärts fahren und sich einem Hindernis nähern?

Nachdem dein Fahrzeug dank Ultraschall-'Parksensor' anhält, sollst du dein Programm nun so erweitern, dass der Rad-Roboter ein Warngeräusch ausgibt, kurz bevor er im Rückwärtsgang auf die Bremse tritt.

Du wirst dein Programm ständig debuggen müssen, damit das Warngeräusch genau dann verklingt, wenn dein Rad-Roboter anhält. Welche Teile des Programms musst du verändern?

Geeignete Blöcke

Nutze dieselben Blöcke wie in den Aufgaben 1 und 2, aber erwäge auch den Einsatz von diesem hier:



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

Nach dem Programmieren ist es wichtig, seine Gedanken und Beobachtungen niederzuschreiben.

Denke über folgende Punkte nach und notiere dann im Kasten unten, wie diese Programmiersitzung verlaufen ist.

- Wie könntest du dein Programm verbessern?
- Könnte dein Programm geradliniger sein? Hast du zu viele Blöcke verwendet? Lässt sich das Programm effizienter gestalten?
- Wo könnte dein Programm in der 'realen Welt' Anwendung finden?

Gedanken und Beobachtungen

ROBOT EDUCATOR-ANLEITUNGEN

Die folgenden Robot Educator-Anleitungen helfen Lehrern und Schülern bei der Lösung der Aufgaben.

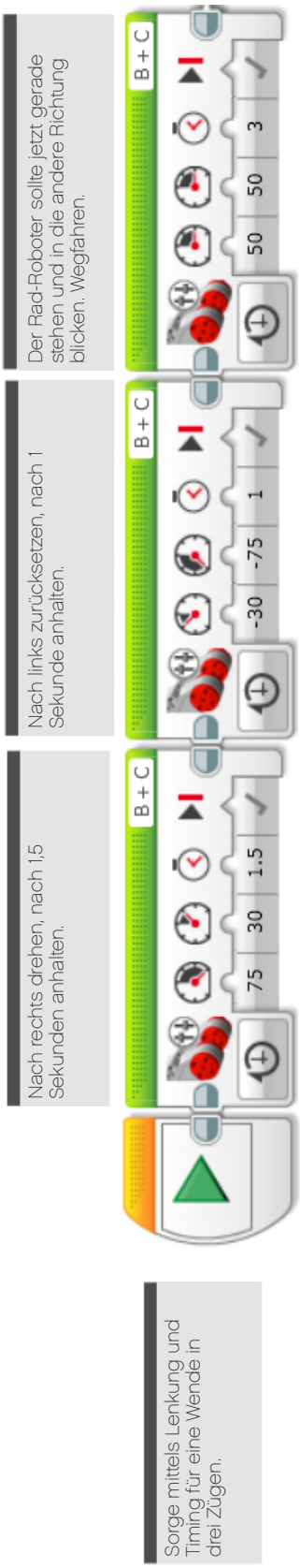
NEUE ROBOT EDUCATOR-ANLEITUNGEN

Grundlagen > Hebellenkung

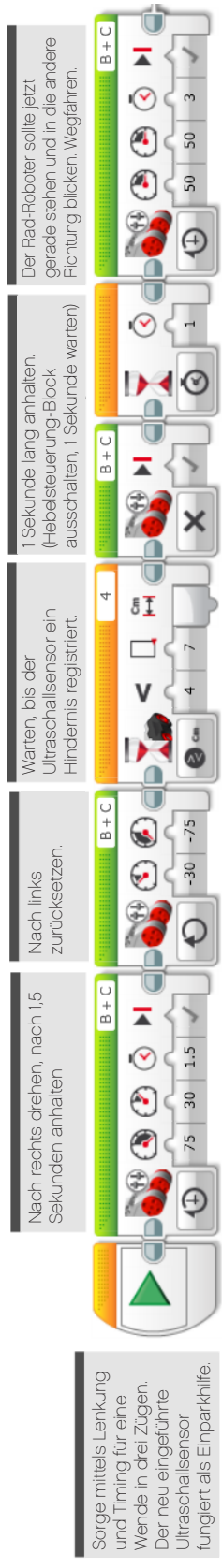
Grundlagen > vor Objekt stoppen

ANHANG FÜR UNTERRICHTSEINHEIT 1: BILDER UND PROGRAMME

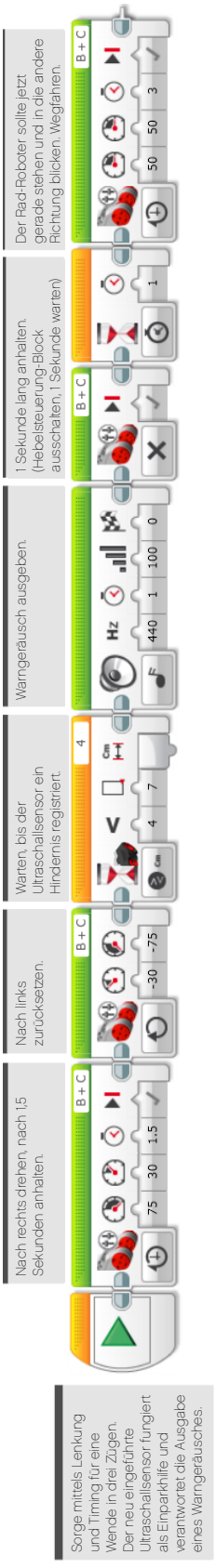
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 1
REITER MAIN 1



LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 1
REITER MAIN 2



LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 1
REITER MAIN 3

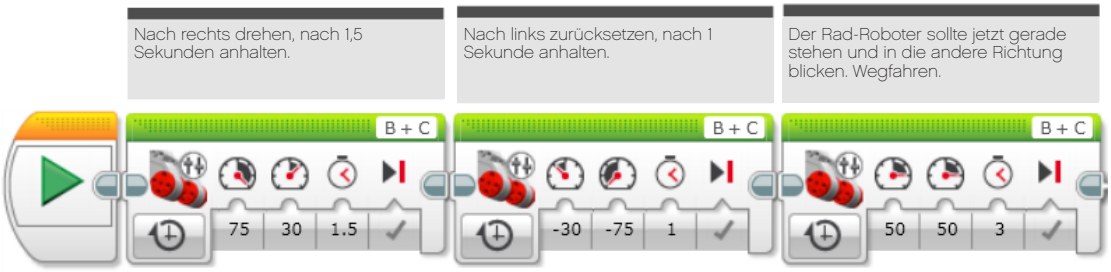


LÖSUNGSVORSCHLAG IN ROBOTC
DATEINAME Session1_1.c

LEGO Start Page | Lesson 1_1.c

```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//
3
4  /*
5  Create a three point turn using timing and steering.
6  */
7
8  task main()
9  {
10   //Turn to the right and stop after 1.5 seconds
11   setMotorSpeed(motorB, 75);
12   setMotorSpeed(motorC, 30);
13   sleep(1500);
14
15   //Reverse to the left and stop after 1 second
16   setMotorSpeed(motorB, -30);
17   setMotorSpeed(motorC, 75);
18   sleep(1000);
19
20   //You should now be straight and facing the other way. Drive off.
21   setMotorSpeed(motorB, 50);
22   setMotorSpeed(motorC, 50);
23   sleep(3000);
24 }
25
```

Sorge mittels Lenkung und Timing für eine Wende in drei Zügen.

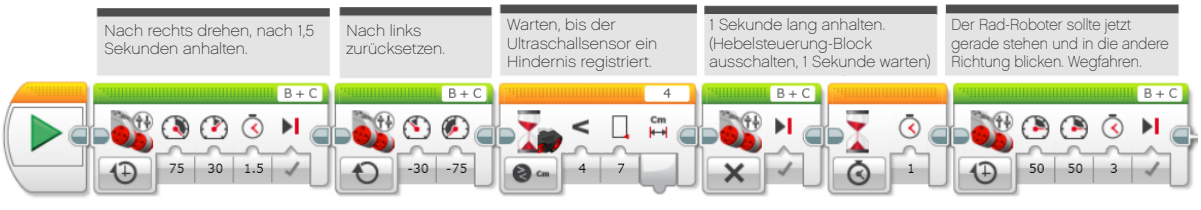


LÖSUNGSVORSCHLAG IN ROBOTC
DATEINAME Session1_2c

LEGO Start Page | Lesson 1_2.c

```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard!!*//
3
4  /*
5  Create a three point turn using timing and steering.
6  Introducing the Ultrasonic Sensor to act as parking sensors.
7  */
8
9  task main()
10 {
11   //Turn to the right and stop after 1.5 seconds
12   setMotorSpeed(motorB, 75);
13   setMotorSpeed(motorC, 30);
14   sleep(1500);
15
16   //While the ultrasonic sees a value greater than 4cm.
17   while(getUSDistance(sonarSensor) > 4)
18   {
19     setMotorSpeed(motorB, -30);
20     setMotorSpeed(motorC, -75);
21   }
22
23   //Once the ultrasonic sees a value less than 4cm...
24   //Stop the robot for 1 second.
25   setMotorSpeed(motorB, 0);
26   setMotorSpeed(motorC, 0);
27   sleep(1000);
28
29   //You should now be straight and facing the other way. Drive off.
30   setMotorSpeed(motorB, 50);
31   setMotorSpeed(motorC, 50);
32   sleep(3000);
33 }
```

Sorge mittels Lenkung und Timing für eine Wende in drei Zügen. Der neu eingeführte Ultraschallsensor fungiert als Einparkhilfe.



LÖSUNGSVORSCHLAG IN ROBOTC
DATEINAME Session1_3c

LEGO Start Page | Lesson 1_3.c*

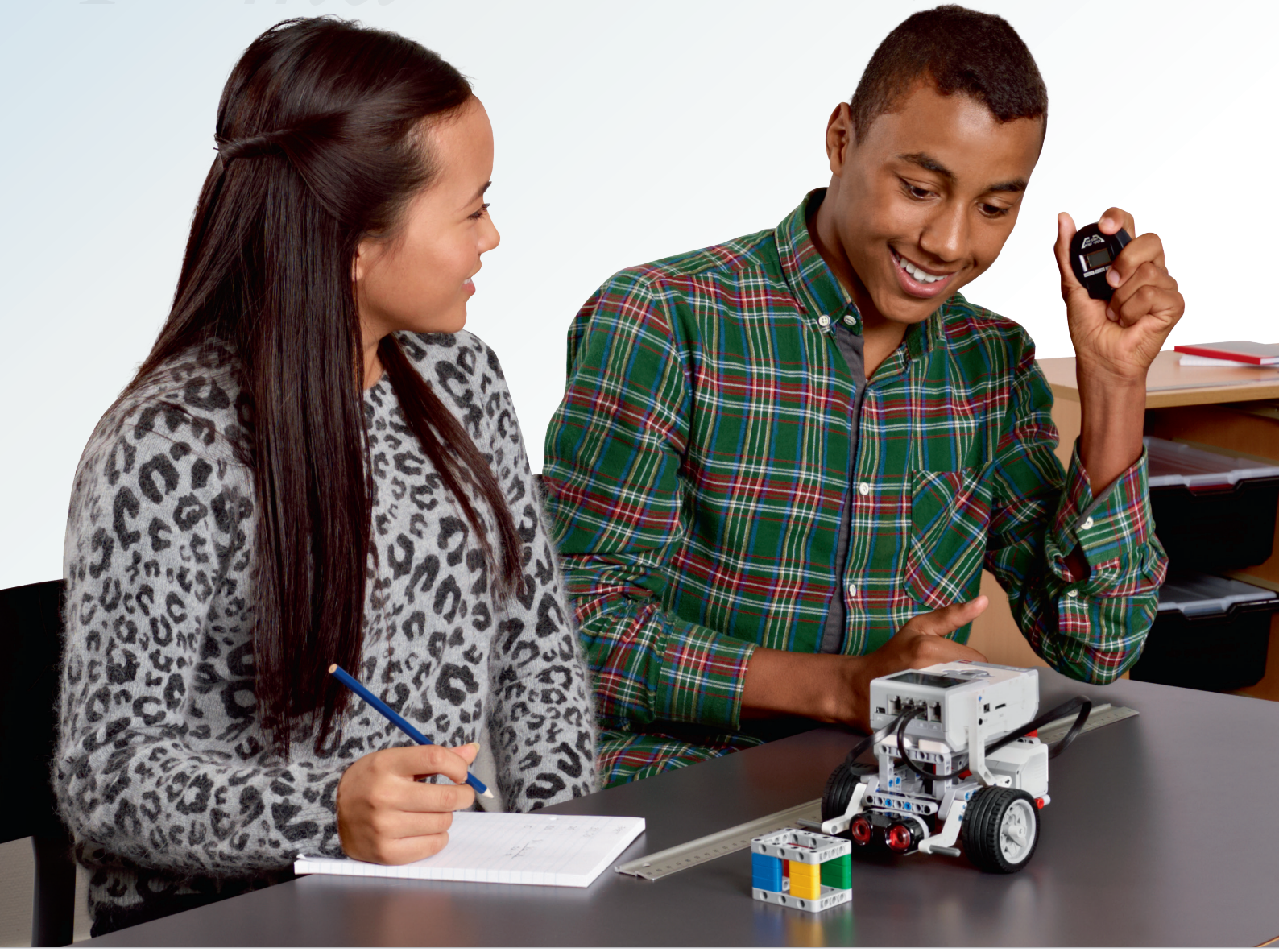
```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard!!*//

/*
Create a three point turn using timing and steering.
Introducing the Ultrasonic Sensor to act as parking sensors.
*/

task main()
{
  //Turn to the right and stop after 1.5 seconds
  setMotorSpeed(motorB, 75);
  setMotorSpeed(motorC, 30);
  sleep(1500);

  //While the ultrasonic sees a value greater than 4cm.
  while(getUSDistance(sonarSensor) > 4)
  {
    setMotorSpeed(motorB, -30);
    setMotorSpeed(motorC, -75);
  }

  //Play sound alert.
  playTone(440, 1000);

  //Once the ultrasonic sees a value less than 4cm...
  //Stop the robot for 1 second.
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  sleep(1000);

  //You should now be straight and facing the other way. Drive off.
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);
  sleep(3000);
}
```

Sorge mittels Lenkung und Timing für eine Wende in drei Zügen. Der neu eingeführte Ultraschallsensor fungiert als Einparkhilfe und verantwortet die Ausgabe eines Warngeräusches.

mindstorms education EV3

UNTERRICHTSEINHEIT 2

Wenden in drei Zügen (textbasierte Programmierung)



LEGOeducation.de

LEGO education

Wenden in drei Zügen (textbasierte Programmierung)

Diese Unterrichtseinheit ist eine Wiederholung der ersten, allerdings werden die Schüler von visueller auf textbasierte Programmierung umsteigen.

Sprechen Sie mit den Schülern über unterschiedliche Programmierumgebungen und die Gründe für deren Vorhandensein.

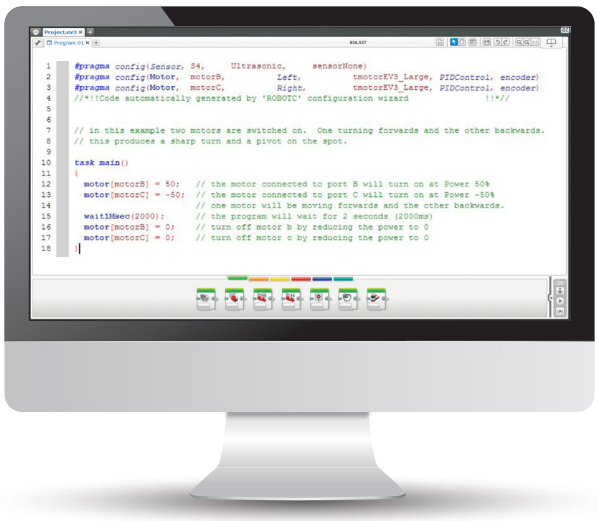
Beschäftigen Sie sich mit der Erkenntnis, dass Menschen unterschiedlich sind und wir alle auf verschiedene Arten lernen. Manche Menschen bevorzugen textbasierte Programmiersprachen, andere visuelle. Weisen Sie darauf hin, dass die Schüler während der kommenden Wochen textbasierte Lösungen entwickeln sollen, die ihren visuellen EV3-Programmen entsprechen.

Lassen Sie die Schüler mit ihrem Roboter einige einfache Fahraufgaben lösen, damit sie vollkommen verstehen, wie die verschiedenen Bewegungs- bzw. Motor-Blöcke arbeiten.

HINFÜHRUNG DER SCHÜLER AN DIE ROBOTC-SOFTWARE

Weisen Sie darauf hin, dass sich die ROBOTC-Software in vielerlei Hinsicht von der bisherigen Programmierung unterscheidet. Der Nutzer muss der Software beispielsweise mitteilen, welche Sensoren und Motoren verwendet werden.

Diese Unterrichtseinheit beschäftigt sich vornehmlich mit der Wiederholung der Aufgaben von Einheit 1. Sie dreht sich also um das Schreiben eines Algorithmus, der den Rad-Roboter eine Wende in drei Zügen ausführen lässt.



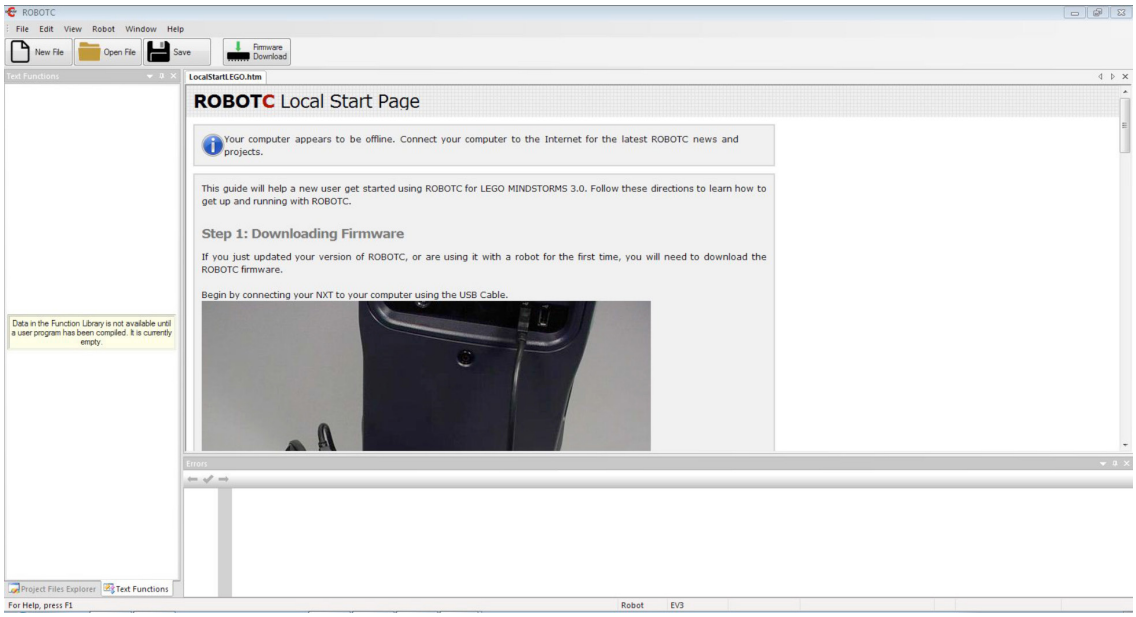
ERGEBNISSE

In dieser Einheit lernen die Schüler,

- die ROBOTC-Software einzusetzen, um Textversionen der in Einheit 1 entwickelten EV3-Programme zu verfassen.
- die Bedeutung von Syntax, also von Strichpunkten, Klammern etc. zu verstehen.
- eine ROBOTC-Sitzung anzulegen.
- dass Algorithmen eine Serie von Befehlen in einer bestimmten Reihenfolge ausführen können.
- einen einfachen Rad-Roboter zu bauen und zu programmieren um eine 180°-Wende in drei Zügen auszuführen.
- mit dem Geschwindigkeits- und Ruhe-Befehlen umzugehen.

EINFÜHRUNG

- Erinnern Sie die Schüler an die Inhalte der letzten Unterrichtseinheit: Sie mussten einen Rad-Roboter bauen. Verwenden Sie (falls nicht schon geschehen) die Anleitung im EV3 Basis-Set, um das Basis-Modell zu bauen.
- Zeigen Sie den Schülern, wie man die ROBOTC-Software einrichtet und in welchem Pull Down-Menü der unterstütze Roboter-Typ zu finden ist. Erklären Sie den Schülern, dass einige Unternehmen aus der Roboter- und Automatisierungssparte diese Software einsetzen.



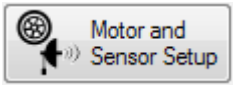
- Ist in den Plattform-Einstellungen kein EV3 zu finden, dann befolgen Sie die Online-Anleitung auf der ROBOTC-Website, um zur EV3-Plattform zu wechseln.

  Hierzu müssen Sie folgendes tun:

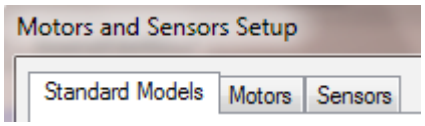
  1. Ändern Sie in der Software den Plattform-Typ zu LEGO® MINDSTORMS® EV3
  2. Machen Sie ein Update der EV3-Firmware (Betriebssystem), indem sie auf "Download EV3 Linux Firmware" und anschließend "Standard File" klicken.
  3. Installieren Sie die ROBOTC-Firmware durch Klicken auf "Download Firmware" und "Standard File".

ROBOTC für die Programmierung vorbereiten

- Erklären Sie, dass man ROBOTC (im Gegensatz zur EV3-Software) mitteilen muss, welche Sensoren und Motoren verwendet werden. Es gibt zwei Möglichkeiten, die Software auf die Hardware abzustimmen:
  4. Voreingestellte Modelle verwenden
  5. Die Software manuell konfigurieren
- Zunächst muss der "Motor and Sensor Setup"-Button geklickt werden.

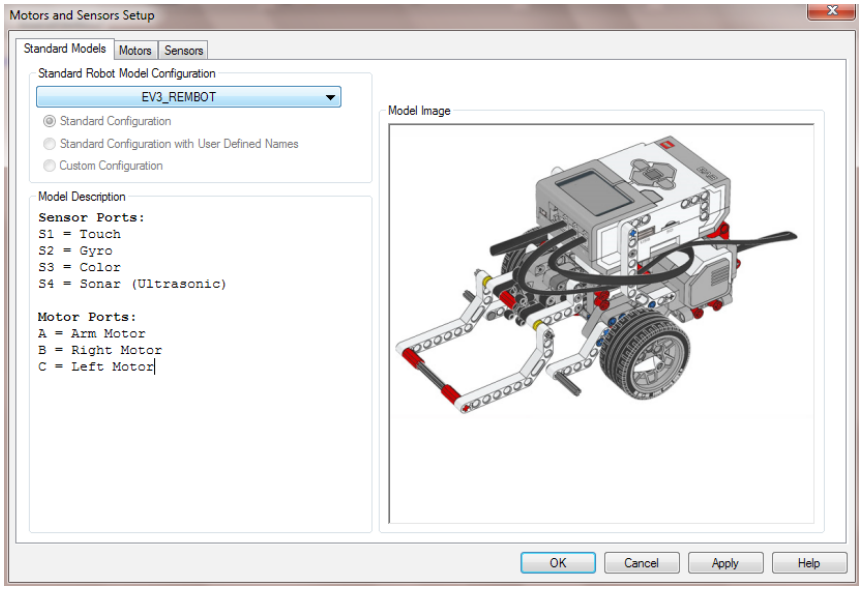


- Nun hat der Nutzer die Wahl zwischen einem Standard-Modell oder der Konfiguration eigener Roboter über die "Motors"- und "Sensors"-Reiter.

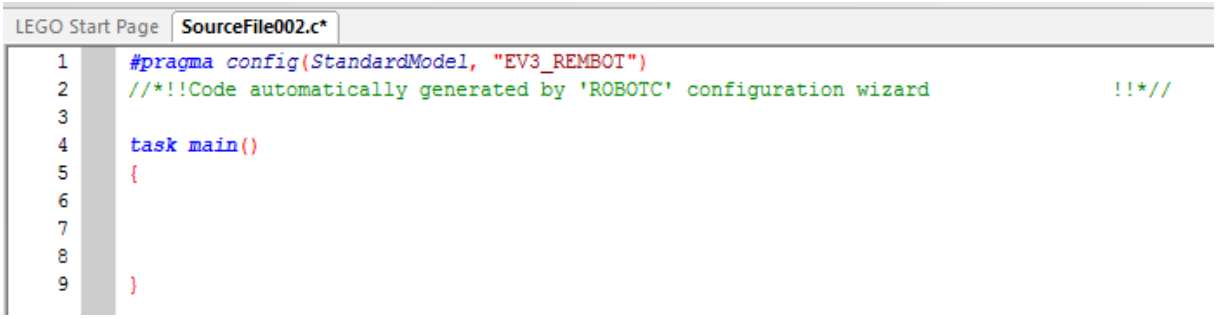


Standard-Modelle nutzen

- Klicken Sie auf den Reiter "Standard Models"
- Nun klicken Sie auf das Pull Down-Menü und wählen "EV3__REMBOT". Dies ist das Standard-Robot Educator-Modell. Nach der Auswahl erscheint ein Bild des Robot Educator-Modells sowie eine Auflistung aller Sensor- und Motor-Schnittstellen.

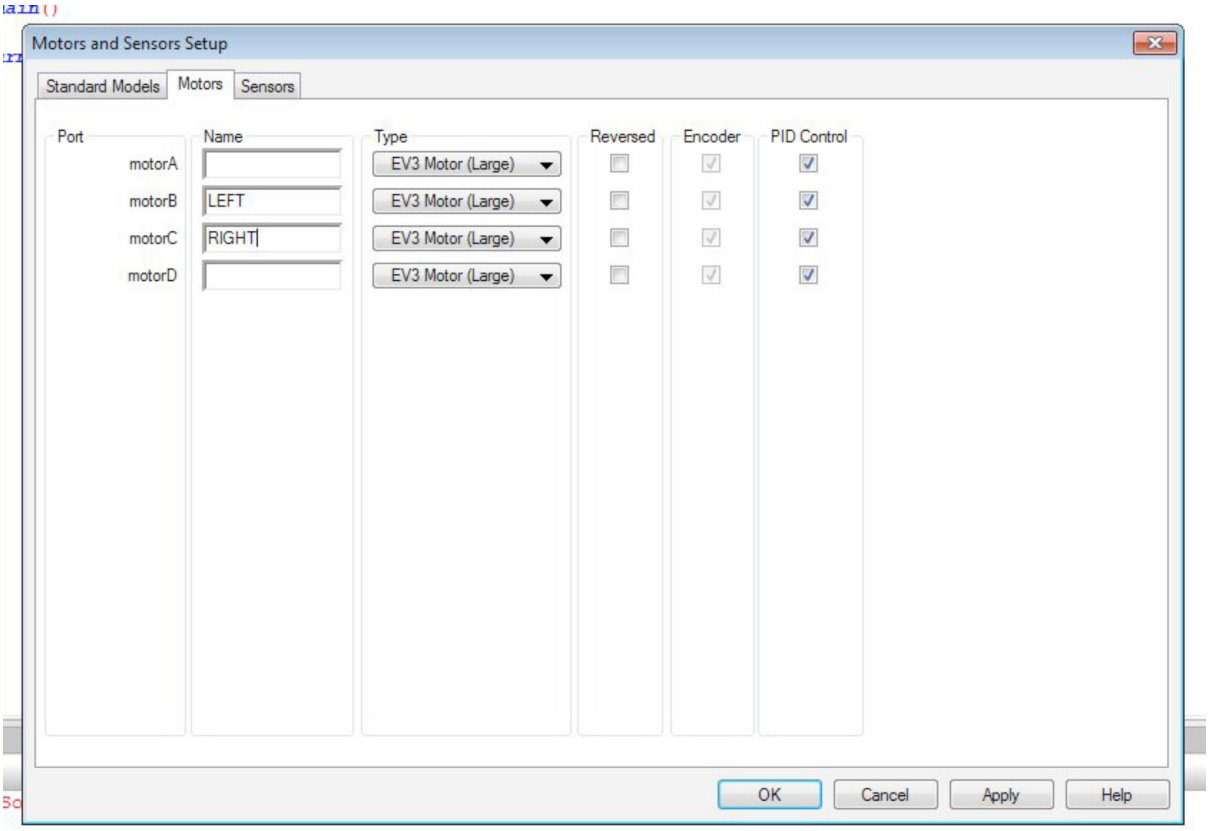


- Beachten Sie, dass oben im Programmier-Fenster die Befehlszeile "#pragma config" den Wert "EV3__REMBOT" besitzt.

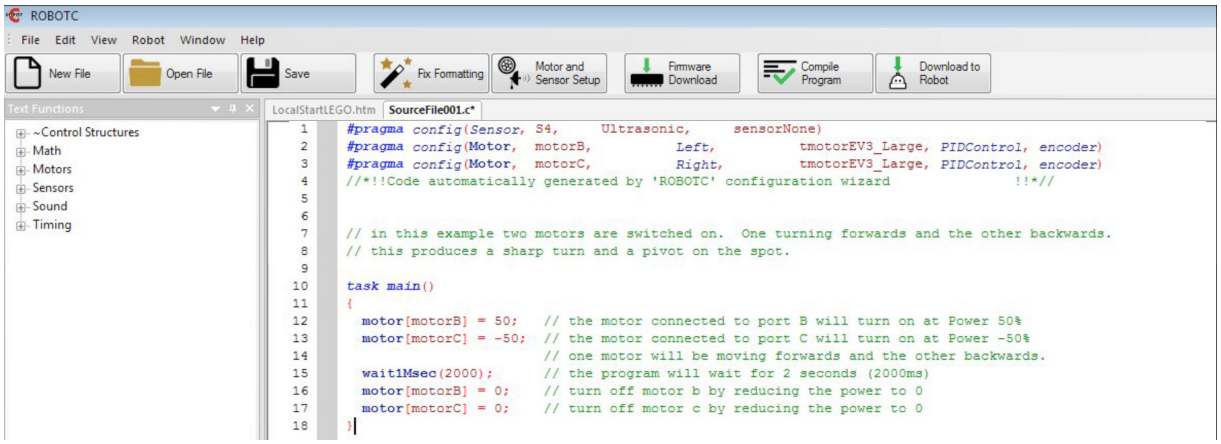


Motoren und Sensoren manuell konfigurieren

- Klicken Sie auf den "Motors"-Reiter.
- Wie unten dargestellt geben Sie "motorB" den Namen "LEFT", "motorC" den Namen "RIGHT". Anschließend klicken Sie auf OK.



- Nun zeigen Sie den Schülern, wie man in textbasierter Programmiersprache eine einfache Drehung des Rad-Roboters auslöst.
- Weisen Sie zudem darauf hin, dass man durch Voranstellen zweier Schrägstriche Anmerkungen in das Programm einfügen kann (siehe unten). Es ist wichtig, dass die Schüler jeden Abschnitt des Textprogramms genauso gut verstehen wie die Blöcke der EV3-Software.



EINFÜHRUNGSAUFGABE

- Die Schüler arbeiten zu zweit oder zu dritt am Bau des Basis-Modells. Die ist unter Umständen bereits in Unterrichtseinheit 1 geschehen.
- Anschließend starten die Schüler ein neues Programm und experimentieren mit verschiedenen Möglichkeiten, den Roboter zu drehen. Hier sind drei Beispiele:

LEGO Start Page | **Explore1.c** | explore2.c | explore3.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//

task main()
{
  //motor c reverses at power 50
  //motor b drives forward at power 50
  //robot will pivot centrally for 1 second (1000ms).
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, -50);
  sleep(1000);
}
```

Beispiel 1: Auf der Stelle drehen

LEGO Start Page | Explore1.c | **explore2.c** | explore3.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//

task main()
{
  //motor c drives at power 50
  //robot will pivot on stationary wheel for 1 second (1000ms).
  setMotorSpeed(motorC, 50);
  sleep(1000);
}
```

Beispiel 2: Um ein Rad drehen

LEGO Start Page | Explore1.c | explore2.c | **explore3.c**

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//

task main()
{
  //motor b drives forward at power 75
  //motor c drives forward at power 50
  //robot will turn gently for 1 second (1000ms).
  setMotorSpeed(motorB, 75);
  setMotorSpeed(motorC, 50);
  sleep(1000);
}
```

Beispiel 3: Langsame Kurve

HAUPTAUFGABE 1

- Sammeln Sie Feedback der Schüler zu den unterschiedlichen Methoden, wie man den Rad-Roboter mittels Hebelsteuerung-Block Kurven fahren lässt. Dabei soll auch erläutert werden, wie sie auf die jeweilige Methode gekommen sind.
- Sagen Sie den Schülern, dass sie ihre Erkenntnisse nun nutzen sollen, um eine Wende in drei Zügen zu simulieren.
- Dies wäre ein guter Zeitpunkt, den Schülern anhand eines Online-Videos zu zeigen, wie eine Wende in drei Zügen ausgeführt wird.
- Die Schüler verwenden nun ROBOTC, um ein Programm zum Wenden in drei Zügen zu entwickeln. Dabei nutzen sie unterschiedliche Leistungseinstellungen der beiden Motoren.
- Fordern Sie Ihre Schüler auf, das Programm kontinuierlich zu testen und zu debuggen (also Fehler auszumerzen), um zu gewährleisten, dass der Rad-Roboter eine 180°-Wende macht.
- Anmerkung: Sie können mit Klebeband eine Straße abstecken, um die Schüler mit einer eingeschränkten Fläche zu konfrontieren, innerhalb der der Rad-Roboter wenden muss.
- Anmerkung: Im Lösungsvorschlag steht, dass der Rad-Roboter am Ende 'gerade stehen sollte'. Sie sollten sinnvollerweise anmerken, dass das in der Realität nicht immer der Fall ist und die Schüler mitunter vor der abschließenden Geraden eine Kurve einbauen. Stellen Sie klar, dass es bei den Aufgaben keinen falschen Lösungsweg gibt.
- Lassen Sie die Schüler die zwei Herangehensweisen an diese Aufgabe vergleichen. Welche war einfacher, welche effektiver?

LÖSUNGSVORSCHLAG
DATEINAME Session2_1.c

LEGO Start Page | Lesson 2_1.c

```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//
3
4  /*
5  Create a three point turn using timing and steering.
6  */
7
8  task main()
9  {
10   //Turn to the right and stop after 1.5 seconds
11   setMotorSpeed(motorB, 75);
12   setMotorSpeed(motorC, 30);
13   sleep(1500);
14
15   //Reverse to the left and stop after 1 second
16   setMotorSpeed(motorB, -30);
17   setMotorSpeed(motorC, 75);
18   sleep(1000);
19
20   //You should now be straight and facing the other way. Drive off.
21   setMotorSpeed(motorB, 50);
22   setMotorSpeed(motorC, 50);
23   sleep(3000);
24 }
25
```

HAUPTAUFGABE 2

- Führen Sie den Ultraschallsensor in den Unterricht ein. Zeigen Sie den Schülern, wie man der Konfiguration über das "Motors and Sensors Setup" weitere Sensoren hinzufügt.
- Fragen Sie die Schüler, wie sich das Programm eigenständiger machen ließe und alle Hindernisse erfasst werden könnten, die womöglich während der Rückwärtsfahrt des Rad-Roboters auftauchen.
- Demonstrieren Sie den "While"-Befehl und seine Verwendung im Zusammenhang mit dem Ultraschallsensor. Weisen Sie auf die Motoreinstellungen innerhalb und außerhalb der While-Schleife hin.
- Die Schüler sollen ihr Programm so erweitern, dass der Rad-Roboter zu jedem beliebigen Zeitpunkt auf den Ultraschallsensor reagiert und anhält.
- Machen Sie den Schülern klar, dass ihre Programme einen Autofahrer simulieren, der auf die Bremse tritt, wenn er beim Rückwärtsfahren einem Hindernis zu nahe kommt.
- Lassen Sie die Schüler die zwei Herangehensweisen an diese Aufgabe vergleichen. Welche war einfacher, welche effektiver?

LÖSUNGSVORSCHLAG
DATEINAME Session2_2.c

LEGO Start Page | Lesson 2_2.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard!!*//

/*
Create a three point turn using timing and steering.
Introducing the Ultrasonic Sensor to act as parking sensors.
*/

task main()
{
  //Turn to the right and stop after 1.5 seconds
  setMotorSpeed(motorB, 75);
  setMotorSpeed(motorC, 30);
  sleep(1500);

  //While the ultrasonic sees a value greater than 4cm.
  while(getUSDistance(sonarSensor) > 4)
  {
    setMotorSpeed(motorB, -30);
    setMotorSpeed(motorC, -75);
  }

  //Once the ultrasonic sees a value less than 4cm...
  //Stop the robot for 1 second.
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  sleep(1000);

  //You should now be straight and facing the other way. Drive off.
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);
  sleep(3000);
}
```

HAUPTAUFGABE 3

- Nun wird der Ultraschallsensor eingesetzt, um dem Rad-Roboter Sicherheitsfunktionen (wie Warngeräusche) hinzuzufügen.
- Fragen Sie die Schüler, was bei vielen modernen Autos passiert, wenn diese zurücksetzen und sich einem Hindernis nähern (ein Warngeräusch ist zu hören).
- Führen Sie den Klang-Befehl ein. Er dient dazu, vor dem Anhalten des Rad-Roboters ein Warngeräusch auszugeben. Zeigen Sie den Schülern, wie der Klang-Befehl funktioniert und wo im Programm er eingefügt werden sollte.
- Die Schüler müssen ihr bestehendes Programm durch Einfügen des "Sound"-Befehls abändern. Er soll ein Warngeräusch ausgeben, wenn sich der Rad-Roboter in einer bestimmten Entfernung zu einem Hindernis befindet.
- Fordern Sie die Schüler auf, ihre Programme durchgehend zu debuggen, um die Geräuschausgabe und die Bremsdistanz zu optimieren.

LÖSUNGSVORSCHLAG
DATEINAME Session2_3.c

LEGO Start Page | Lesson 2_3.c*

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard!!*//

/*
Create a three point turn using timing and steering.
Introducing the Ultrasonic Sensor to act as parking sensors.
*/

task main()
{
  //Turn to the right and stop after 1.5 seconds
  setMotorSpeed(motorB, 75);
  setMotorSpeed(motorC, 30);
  sleep(1500);

  //While the ultrasonic sees a value greater than 4cm.
  while(getUSDistance(sonarSensor) > 4)
  {
    setMotorSpeed(motorB, -30);
    setMotorSpeed(motorC, -75);
  }

  //Play sound alert.
  playTone(440, 1000);

  //Once the ultrasonic sees a value less than 4cm...
  //Stop the robot for 1 second.
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  sleep(1000);

  //You should now be straight and facing the other way. Drive off.
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);
  sleep(3000);
}
```

ABSCHLIESSENDE DISKUSSION

- Rekapitulieren Sie, was die Schüler in dieser Unterrichtseinheit gelernt haben.
- Sorgen Sie dafür, dass die Schüler die Begriffe, die in dieser Unterrichtseinheit verwendet wurden, richtig einsetzen und verstehen. Wiederholen Sie die Bedeutung der Schlüsselbegriffe.
- Lassen Sie ein oder zwei Gruppen ihre Programme demonstrieren. Diskutieren Sie darüber, was an den Programmen gut funktioniert und wo man sie verbessern könnte.
- Fragen Sie die Schüler nach ihrer Meinung zu textbasierten Programmen. Welcher Anteil der Klasse würde lieber mit einer textbasierten Programmiersprache arbeiten wollen?

AUFGABEN DIESER UNTERRICHTSEINHEIT

Heute sollst du dich mit der ROBOTC-Software vertraut machen.

Du hattest bereits ein wenig Zeit, mit dem Befehl zur Motorgeschwindigkeit (setMotorSpeed) zu experimentieren und durch seinen Einsatz deinen Rad-Roboter in Bewegung zu versetzen. Nun wirst du an deinen Fähigkeiten feilen müssen, um die nachfolgenden drei Aufgaben zu meistern.

Viel Erfolg!

AUFGABE 1

Programmiere deinen Rad-Roboter so, dass dieser eine Wende in drei Zügen ausführt.

Du musst deinen Rad-Roboter während der Vorwärtsfahrt drehen. Dann setzt du mit ihm zurück, bevor er vorwärts weiterfährt.

Sieh dir noch einmal das Online-Video an, um dir die Wende in drei Zügen in Erinnerung zu rufen, und achte auf die Straßenmarkierungen!

Geeignete Programmierbefehle

setMotorSpeed **sleep**

Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 2

Jetzt wirst du mit einem der EV3-Sensoren experimentieren - dem Ultraschallsensor.

Programmiere deinen Rad-Roboter so, dass er eine Wende in drei Zügen vollführt, und setze den Ultraschallsensor als 'Einparkhilfe' ein. Dein Rad-Roboter soll beim Rückwärtsfahren in einer bestimmten Entfernung zu einem Hindernis anhalten.

Kann dein Rad-Roboter 'in die Bremsen steigen', bevor er wieder davonfährt?

Hier ist dein Wissen über den "While"-Befehl gefragt. Zudem solltest du den Ultraschallsensor an der Rückseite deines Rad-Roboters anbringen.

Geeignete Programmierbefehle

setMotorSpeed **sleep** **while** **getUSDistance**

Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 3

Simulation von Warngeräuschen.

Was passiert bei vielen modernen Autos, wenn diese rückwärts fahren und sich einem Hindernis nähern?

Nachdem dein Fahrzeug dank Ultraschall-'Parksensor' anhält, sollst du dein Programm nun so erweitern, dass der Rad-Roboter ein Warngeräusch ausgibt, kurz bevor er im Rückwärtsgang auf die Bremse tritt.

Du wirst dein Programm ständig debuggen müssen, damit das Warngeräusch genau dann verklingt, wenn dein Rad-Roboter anhält. Welche Teile des Programms musst du verändern?

Geeignete Programmierbefehle

setMotorSpeed **sleep** **while** **getUSDistance** **playTone**

Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

Nach dem Programmieren ist es wichtig, seine Gedanken und Beobachtungen niederzuschreiben.

Denke über folgende Punkte nach und notiere dann im Kasten unten, wie diese Programmiersitzung verlaufen ist.

- Wie könntest du dein Programm verbessern?
- Könnte dein Programm geradliniger sein? Hast du zu viele Blöcke verwendet? Lässt sich das Programm effizienter gestalten?
- Wo könnte dein Programm in der 'realen Welt' Anwendung finden?

Gedanken und Beobachtungen

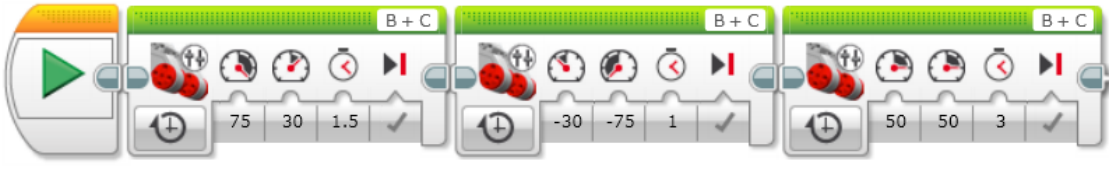
ANHANG FÜR UNTERRICHTSEINHEIT 2: BILDER UND PROGRAMME

LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session2_1.c

LEGO Start Page | Lesson 1_1.c

```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//
3
4  /*
5  Create a three point turn using timing and steering.
6  */
7
8  task main()
9  {
10   //Turn to the right and stop after 1.5 seconds
11   setMotorSpeed(motorB, 75);
12   setMotorSpeed(motorC, 30);
13   sleep(1500);
14
15   //Reverse to the left and stop after 1 second
16   setMotorSpeed(motorB, -30);
17   setMotorSpeed(motorC, 75);
18   sleep(1000);
19
20   //You should now be straight and facing the other way. Drive off.
21   setMotorSpeed(motorB, 50);
22   setMotorSpeed(motorC, 50);
23   sleep(3000);
24 }
25
```

Sorge mittels Lenkung und Timing für eine Wende in drei Zügen.

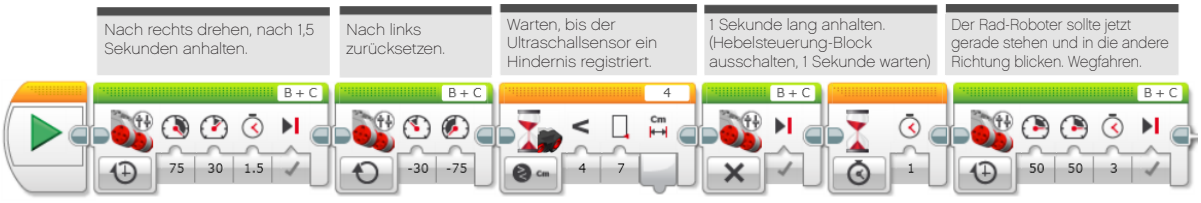


LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session2_2.c

LEGO Start Page | Lesson 1_2.c

```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard!!*//
3
4  /*
5  Create a three point turn using timing and steering.
6  Introducing the Ultrasonic Sensor to act as parking sensors.
7  */
8
9  task main()
10 {
11   //Turn to the right and stop after 1.5 seconds
12   setMotorSpeed(motorB, 75);
13   setMotorSpeed(motorC, 30);
14   sleep(1500);
15
16   //While the ultrasonic sees a value greater than 4cm.
17   while(getUSDistance(sonarSensor) > 4)
18   {
19     setMotorSpeed(motorB, -30);
20     setMotorSpeed(motorC, -75);
21   }
22
23   //Once the ultrasonic sees a value less than 4cm...
24   //Stop the robot for 1 second.
25   setMotorSpeed(motorB, 0);
26   setMotorSpeed(motorC, 0);
27   sleep(1000);
28
29   //You should now be straight and facing the other way. Drive off.
30   setMotorSpeed(motorB, 50);
31   setMotorSpeed(motorC, 50);
32   sleep(3000);
33 }
```

Sorge mittels Lenkung und Timing für eine Wende in drei Zügen. Der neu eingeführte Ultraschallsensor fungiert als Einparkhilfe.

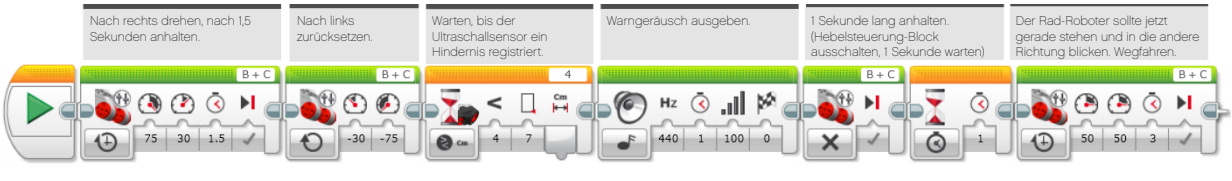


LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session2_3.c

LEGO Start Page | Lesson 1_3.c*

```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard!!*//
3
4  /*
5  Create a three point turn using timing and steering.
6  Introducing the Ultrasonic Sensor to act as parking sensors.
7  */
8
9  task main()
10 {
11   //Turn to the right and stop after 1.5 seconds
12   setMotorSpeed(motorB, 75);
13   setMotorSpeed(motorC, 30);
14   sleep(1500);
15
16   //While the ultrasonic sees a value greater than 4cm.
17   while(getUSDistance(sonarSensor) > 4)
18   {
19     setMotorSpeed(motorB, -30);
20     setMotorSpeed(motorC, -75);
21   }
22
23   //Play sound alert.
24   playTone(440, 1000);
25
26   //Once the ultrasonic sees a value less than 4cm...
27   //Stop the robot for 1 second.
28   setMotorSpeed(motorB, 0);
29   setMotorSpeed(motorC, 0);
30   sleep(1000);
31
32   //You should now be straight and facing the other way. Drive off.
33   setMotorSpeed(motorB, 50);
34   setMotorSpeed(motorC, 50);
35   sleep(3000);
36 }
37
```

Sorge mittels Lenkung und Timing für eine Wende in drei Zügen. Der neu eingeführte Ultraschallsensor fungiert als Einparkhilfe und verantwortet die Ausgabe eines Warngeräusches.



mindstorms education EV3

UNTERRICHTSEINHEIT 3

Roboter im Rückwärtsgang

LEGOeducation.de

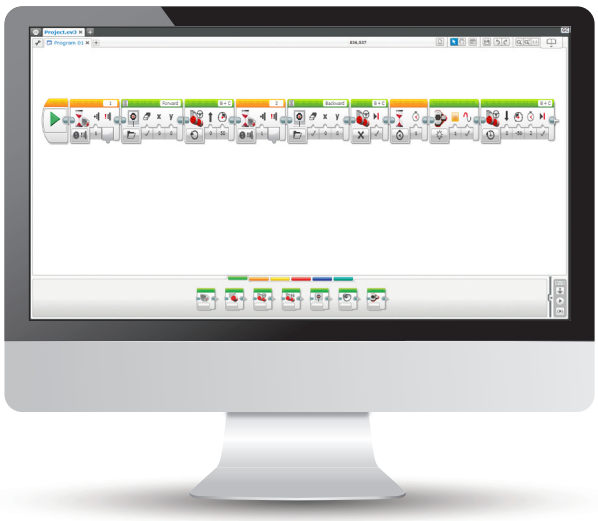
education

Roboter im Rückwärtsgang

In dieser Unterrichtseinheit programmieren die Schüler ihren Rad-Roboter so, dass er beim Rückwärtsfahren visuelle Signale gibt, wie sie in der Realität für Fußgänger, andere Verkehrsteilnehmer und den Autofahrer selbst wichtig sind.

Unter Verwendung der Stein-Statusleuchte- sowie Anzeige-Blöcke entwickeln die Schüler Programme, die das Verhalten ihres Rad-Roboters sichtbar kommunizieren.

Die Schüler setzen zudem den Berührungssensor ein, um Vorwärts- und Rückwärtsgang zu simulieren.



Unterrichtseinheit 3
Stundenplanung

ERGEBNISSE

In dieser Einheit lernen die Schüler,

- dass Algorithmen eine Serie von Befehlen in einer bestimmten Reihenfolge ausführen können.
- wie man den Standardsteuerung-Block einsetzt, um den Rad-Roboter geradeaus zu bewegen.
- den Warten-Block in Verbindung mit Berührungssensoren zu verwenden.
- die Statusleuchte- und Anzeige-Funktionen zu nutzen.
- durch die Entwicklung komplexerer Algorithmen ihre Programmierfertigkeiten weiterzuentwickeln.

BEGRIFFE

Eingabe, Ausgabe, Algorithmus, Warte, Berührungssensor, debuggen, Standardsteuerung-Block.

EINFÜHRUNG

- Erläutern Sie den Schülern, dass sie im Lauf der Unterrichtseinheit durch Programmieren ihres Rad-Roboters typische Elemente (wie Rücklichter oder Anzeigen am Armaturenbrett) des Rückwärtsfahrens von Autos simulieren werden. Zudem lernen sie den Berührungssensor kennen.
- Lassen Sie die Schüler kurz darüber diskutieren, was beim Rückwärtsfahren eines Autos passiert. Betonen Sie, dass Rücklichter andere Verkehrsteilnehmer warnen. Zudem verfügen manche Autos über Anzeigen am Armaturenbrett, die Fahrer und Insassen über den eingelegten Gang (und damit die Fahrtrichtung) informieren.
- Demonstrieren Sie anhand der EV3-Software die drei verschiedenen Zustände des Berührungssensors (gedrückt, ausgelassen, angestoßen) sowie den Stein-Statusleuchte-Block.

HAUPTAUFGABE 1

- Sagen Sie den Schülern, dass sie in dieser Aufgabe ein Programm schreiben werden, welches das Schalten eines Autos in den Rückwärtsgang und die gleichzeitige Anzeige entsprechender Warnlichter simuliert.
- Die Schüler sollen ein Programm entwickeln, das ihren Rad-Roboter vorwärts fahren lässt. Ist (über Betätigung des Berührungssensors) der 'Rückwärtsgang' eingelegt, bewegt sich der Roboter rückwärts und zeigt über die EV3-Stein-Statusleuchte ein Rückfahr-Warnlicht an.
- Den Schülern sollte vor Anpacken dieser Aufgabe etwas Zeit gegönnt werden, sich mit dem Berührungssensor und dem Stein-Statuslicht-Block vertraut zu machen.
- Anmerkung: Die Schüler können zwar den ihnen bekannten Hebelsteuerung-Block für den Antrieb des Rad-Roboters nutzen, der Einsatz des Standardsteuerung-Blocks ist aber womöglich vorteilhafter.

LÖSUNGSVORSCHLAG DATEINAME CS LESSON 3 REITER MAIN 1

HAUPTAUFGABE 2

- Bringen Sie einen zweiten Berührungssensor als 'Vorwärtsgang' ins Spiel.
- Zeigen Sie den Schülern, dass sich ein weiterer Berührungssensor einsetzen lässt, und dass die MINDSTORMS-Software unterscheiden kann, welcher Sensor ausgelöst wird.
- Die Schüler bauen auf ihren vorherigen Lernerfahrungen auf und entwickeln ein Programm, welches einen Vor- und Rückwärtsgang simuliert. Der zweite Berührungssensor (der 'Vorwärtsgang') löst den Start des Programms aus.

LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 3
REITER MAIN 2

HAUPTAUFGABE 3

- Setzen Sie den EV3-Stein als Armaturenbrett-Anzeige ein.
- Die Schüler entwickeln ihr Programm weiter, um über Verwendung des Anzeige-Blocks das Display des EV3-Steins in das Geschehen miteinzubeziehen. Bei Vor- und Rückwärtsbewegung des Rad-Roboters stellt das Display die Bewegungsrichtung in Form von Pfeilen dar.
- Demonstrieren Sie den Schülern, wie der Anzeige-Block verwendet wird.
- Die Schüler benötigen mitunter etwas Zeit, um die Möglichkeiten des Anzeige-Blocks zu erkunden und im Programm sinnvoll einzusetzen.
- Welche Aufgabe könnte das Display in einem automatisierten Auto übernehmen? Weisen Sie darauf hin, dass den Passagieren eines führerlosen Fahrzeugs über das Display wichtige Informationen mitgeteilt werden könnten.

LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 3
REITER MAIN 3

ABSCHLIESSENDE DISKUSSION

- Lassen Sie ein oder zwei Gruppen ihre Programme sowie den Rad-Roboter in Aktion vorführen.
- Auf wie viele Varianten des Programms sind die Schüler insgesamt gekommen? Dies gemeinsam herauszufinden, hilft bei der Verdeutlichung der Tatsache, dass es für ein bestimmtes Problem zahlreiche Lösungen gibt.
- Stellen Sie die Arbeit mit einem weiteren Sensor (Farbe) in Aussicht. Dieser kommt beim nächsten Schritt zur Entwicklung eines voll automatischen Rad-Roboters zum Einsatz.

AUFGABEN DIESER UNTERRICHTSEINHEIT

Um die heutigen Aufgaben zu lösen, musst du auf deinen bislang erworbenen Programmierfähigkeiten aufbauen. Du wirst einen weiteren Sensor (den Berührungssensor) verwenden sowie die Funktionen des EV3-Steins nutzen. Der Stein wird so programmiert, dass sich seine Lichter aktivieren lassen und das Display als Anzeige fungiert.

Nach der erfolgreichen Beschäftigung mit der dritten Aufgabe wird dein Rad-Roboter einen Vor- und Rückwärtsgang, Rückwärtsleuchten sowie eine Armaturenbrett-Anzeige simulieren.

AUFGABE 1

Kannst du ein Programm schreiben, das deinen Rad-Roboter vorwärts und nach einem Druck auf den Berührungssensor rückwärts fahren lässt?

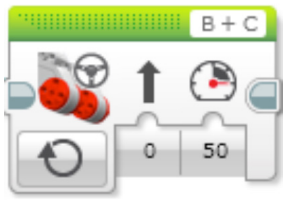
Erweitere anschließend dein Programm:

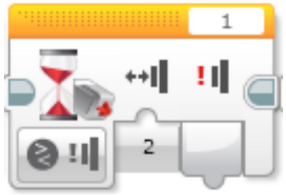
Was lässt sich an Autos beobachten, wenn sie rückwärts fahren? Wie können Fußgänger und andere Verkehrsteilnehmer erkennen, dass das Auto zurückstößt?

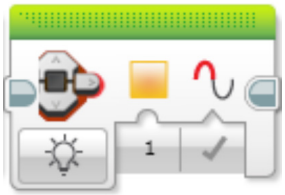
Dein Rad-Roboter sollte beim Rückwärtsfahren ein Warnlicht anzeigen.

Verwende den EV3-Stein und das Statuslicht, um Rückwärtsleuchten zu simulieren.

Geeignete Blöcke







Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 2

Lässt sich dein Programm so erweitern, dass der Rad-Roboter über zwei Gänge (vorwärts und rückwärts) verfügt?

Dein Rad-Roboter sollte 'starten' (vorwärts fahren), wenn der 'Vorwärtsgang' aktiviert ist.

Tipp: Du brauchst einen zweiten Berührungssensor.

Geeignete Blöcke

Verwende dieselben Blöcke wie in der ersten Programmieraufgabe, aber erwäge auch den Einsatz des folgenden:

Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 3

Was geschieht im Auto beim Schalten in unterschiedliche Gänge?

Oft (und vor allem bei Automatik-Autos) gibt es am Armaturenbrett eine Anzeige, die dem Fahrer mitteilt, welchen Gang er gerade eingelegt hat.

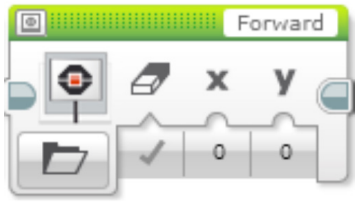
Kannst du so eine Anzeige in deinem Programm unter Verwendung des Anzeige-Blocks simulieren?

Du solltest dich eingehend mit dem Anzeige-Block beschäftigen, um Grafiken zu finden, mit denen sich die Vor- bzw. Rückwärtsbewegung verständlich darstellen lässt.

Dein Programm sollte das bereits bestehende erweitern und weiterhin die Rückleuchten beinhalten!

Geeignete Blöcke

Verwende dieselben Blöcke wie in der ersten und zweiten Programmieraufgabe, aber erwäge auch den Einsatz des folgenden:



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

Unterrichtseinheit 3
Arbeitsblätter

Nach dem Programmieren ist es wichtig, seine Gedanken und Beobachtungen niederzuschreiben.

Denke über folgende Punkte nach und notiere dann im Kasten unten, wie diese Programmiersitzung verlaufen ist.

- Wie könntest du dein Programm verbessern?
- Könnte dein Programm geradliniger sein? Hast du zu viele Blöcke verwendet? Lässt sich das Programm effizienter gestalten?
- Wo könnte dein Programm in der 'realen Welt' Anwendung finden?

Gedanken und Beobachtungen

ROBOT EDUCATOR-ANLEITUNGEN

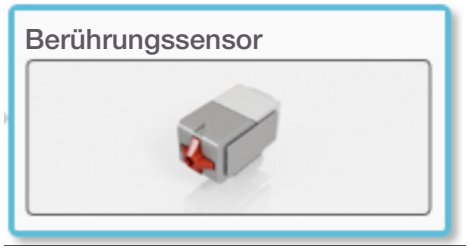
Die folgenden Robot Educator-Anleitungen helfen Lehrern und Schülern bei der Lösung der Aufgaben.

NEUE ROBOT EDUCATOR-ANLEITUNGEN

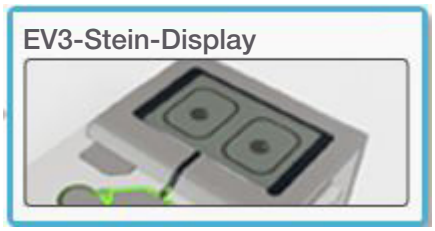
Grundlagen > Geradeausfahrt

Hardware > Stein-Statusleuchte

Hardware > Berührungssensor



Hardware > Stein-Anzeige

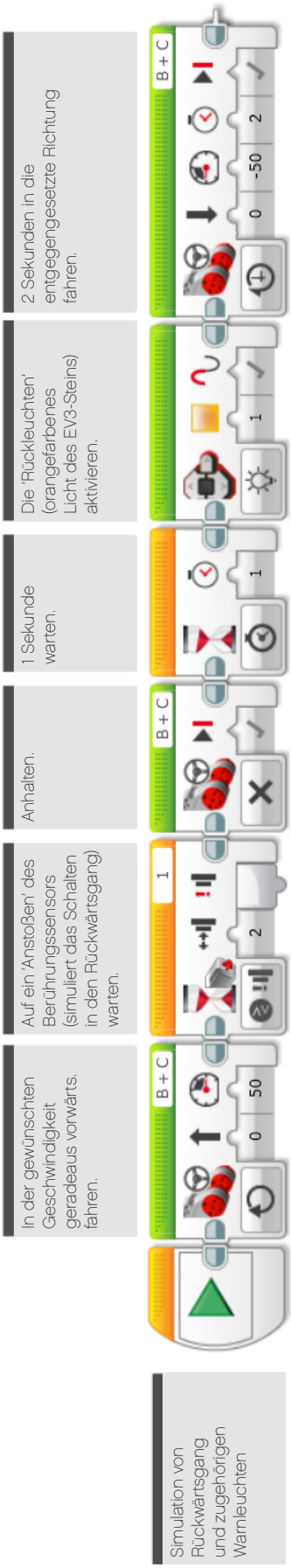


BEREITS BEHANDELTE ROBOT EDUCATOR-ANLEITUNGEN

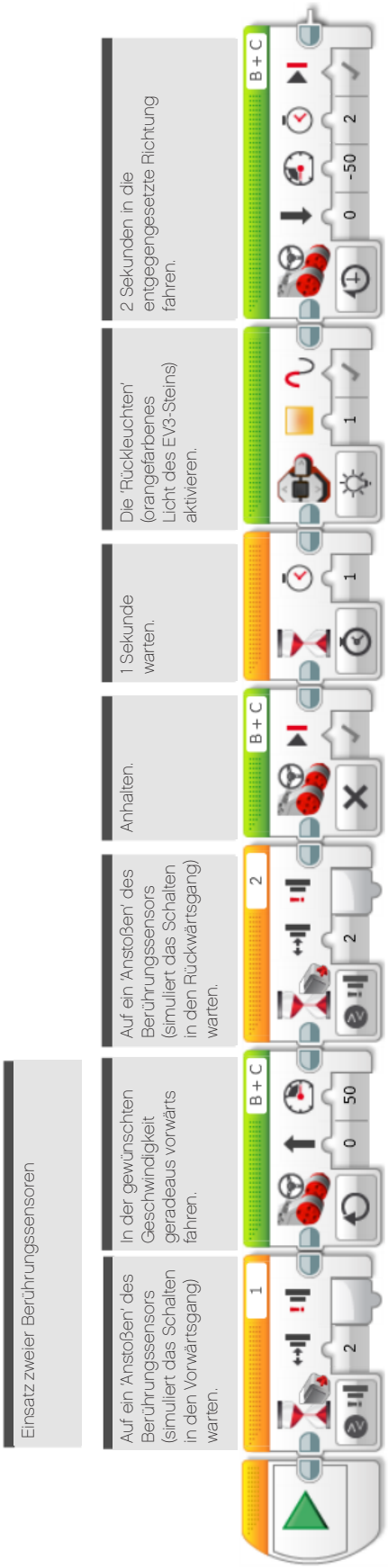
Grundlagen > vor Objekt stoppen

ANHANG FÜR UNTERRICHTSEINHEIT 3: BILDER UND PROGRAMME

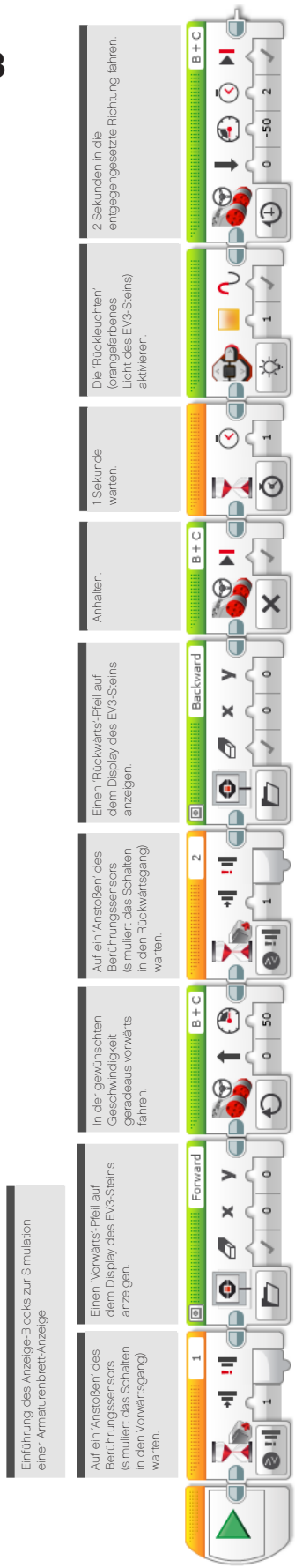
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 3
REITER MAIN 1



LÖSUNGSVORSCHLAG
DATEINAME CS LESSON
REITER MAIN 2



LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 3
REITER MAIN 3



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session3_1.c

LEGO Start Page | Lesson 3_1.c*

```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//
3
4  task main()
5  {
6    //Clear any previous bumper values
7    resetBumpedValue(touchSensor);
8
9    //Wait for touch sensor to be bumped.
10   while(getBumpedValue(touchSensor) == 1)
11   {
12     setMotorSpeed(motorB, 50);
13     setMotorSpeed(motorC, 50);
14   }
15
16   setMotorSpeed(motorB, 0);
17   setMotorSpeed(motorC, 0);
18   sleep(1000);
19
20   setLEDColor(ledOrangeFlash);
21
22   setMotorSpeed(motorB, -50);
23   setMotorSpeed(motorC, -50);
24   sleep(2000);
25
26 }
27
```

Simulation von Rückwärtsgang und zugehörigen Warnleuchten

LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session3_2.c

LEGO Start Page | Lesson 3_2.c*

```
#pragma config(Sensor, S1, touchSensor1, sensorEV3_Touch)
#pragma config(Sensor, S2, touchSensor2, sensorEV3_Touch)
#pragma config(Sensor, S3,colorSensor, sensorEV3_Color)
#pragma config(Sensor, S4,sonarSensor, sensorEV3_Ultrasonic)
#pragma config(Motor,  motorB, rightMotor,tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor,  motorC, leftMotor,tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard!!*//

task main()
{
  resetBumpedValue(touchSensor1);
  resetBumpedValue(touchSensor2);

  //Wait for Touch Sensor #1 to be bumped
  while(getBumpedValue(touchSensor1) == 0)
  {
    //Do nothing until bumped.
  }

  //Wait for Touch Sensor #2 to be bumped
  while(getBumpedValue(touchSensor2) == 0)
  {
    setMotorSpeed(motorB, 50);
    setMotorSpeed(motorC, 50);
  }

  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  sleep(1000);

  setLEDColor(ledOrangeFlash);

  setMotorSpeed(motorB, -50);
  setMotorSpeed(motorC, -50);
  sleep(2000);

}
```

LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session3_3.c

LEGO Start Page | Lesson 3_3.c*

```
1  #pragma config(Sensor, S1, touchSensor1, sensorEV3_Touch)
2  #pragma config(Sensor, S2, touchSensor2, sensorEV3_Touch)
3  #pragma config(Sensor, S3, colorSensor, sensorEV3_Color)
4  #pragma config(Sensor, S4, sonarSensor, sensorEV3_Ultrasonic)
5  #pragma config(Motor,  motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
6  #pragma config(Motor,  motorC, leftMotor, tmotorEV3_Large, PIDControl, driveLeft, encoder)
7  //*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//
8
9  task main()
10 {
11   resetBumpedValue(touchSensor1);
12   resetBumpedValue(touchSensor2);
13
14   //Wait for Touch Sensor #1 to be bumped
15   while(getBumpedValue(touchSensor1) == 0)
16   {
17     //Do nothing until bumped.
18   }
19
20   //Display the text "Forward" on the LCD Screen
21   drawTextAt(0, 0, "Forward");
22
23   //Wait for Touch Sensor #2 to be bumped
24   while(getBumpedValue(touchSensor2) == 0)
25   {
26     setMotorSpeed(motorB, 50);
27     setMotorSpeed(motorC, 50);
28   }
29
30   //Display the text "Backward" on the LCD Screen
31   drawTextAt(0, 0, "Backward");
32
33   setMotorSpeed(motorB, 0);
34   setMotorSpeed(motorC, 0);
35   sleep(1000);
36
37   setLEDColor(ledOrangeFlash);
38
39   setMotorSpeed(motorB, -50);
40   setMotorSpeed(motorC, -50);
41   sleep(2000);
42 }
43
```

mindstorms education EV3

UNTERRICHTSEINHEIT 4

Mit Licht den Weg weisen

LEGOeducation.de

LEGO education

Mit Licht den Weg weisen

Die Schüler machen sich mit dem Farbsensor vertraut und nutzen dessen unterschiedliche Modi (Wahrnehmung von Farbe bzw. Lichtstärke), um diverse automatisierte Funktionen eines Autos zu simulieren.

Durch den Einsatz von Sensoren, die auf Umwelt-Veränderungen reagieren, werden die Programme der Schüler zunehmend anspruchsvoller.

Die Schüler entwickeln Programme, die automatisierte Scheinwerfer simulieren, und erweitern sie anschließend, um eine zusätzliche manuelle Steuerung der Scheinwerfer zu ermöglichen.

Unterrichtseinheit 4
Stundenplanung

ERGEBNISSE

In dieser Einheit lernen die Schüler,

- dass Algorithmen eine Serie von Befehlen in einer bestimmten Reihenfolge ausführen können.
- einfache Boole'sche Logik zu verstehen und einige ihrer Einsatzmöglichkeiten zu nennen.
- den Warten-Block in Verbindung mit dem Farbsensor einzusetzen.
- dass der Farbsensor mehrere Funktionen hat und eine ganze Reihe von Parametern messen und auf sie reagieren kann.
- ihr Verständnis des Anzeige-Blocks zu erweitern und ihn vielfältiger einzusetzen.
- die Funktion des Schleife-Blocks und das Konzept der parallelen Programmierung (Multitasking) zu verstehen.

BEGRIFFE

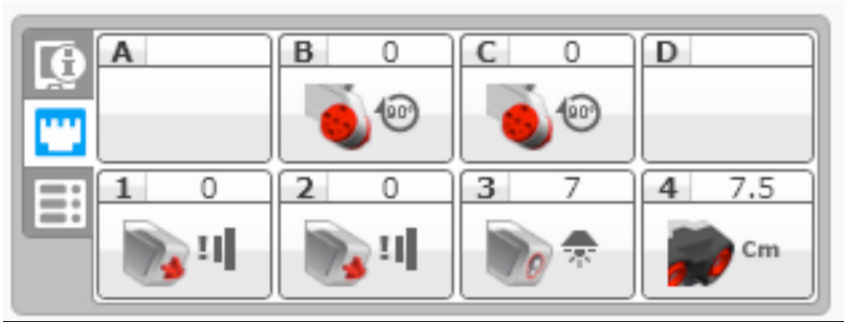
Eingabe, Ausgabe, Algorithmus, Warten, Farbsensor, debuggen, Umgebungslicht, Schleife, Boole'sche Logik, parallele Programmierung.

EINFÜHRUNG

- Eröffnen Sie Ihren Schülern, dass diese während der nächsten zwei Unterrichtseinheiten die unterschiedlichen Funktionen des Farbsensors kennenlernen werden. Über ihre Programme simulieren sie automatische Scheinwerfer, zudem machen sie ihren Roboter mittels Farbsensor selbstständiger.
- Die Schüler werden in dieser Unterrichtseinheit untersuchen, wie Änderungen des Umgebungslichts Antworten eines Programms auslösen können. Erklären Sie ihnen, dass dies ein Beispiel für Boole'sche Logik ist (die Kontrolle eines Programms durch eine Entscheidung).
- Diskutieren Sie mit den Schülern, in welchen Alltagssituationen Veränderungen des Umgebungslichts bestimmte Vorgänge auslösen (nennen Sie Beispiele wie automatische Scheinwerfer oder Straßenlaternen).
- Kündigen Sie an, dass die Schüler sich mit Schleifen beschäftigen werden und damit, wie man diese in die eigenen Programme einfügt.

HAUPTAUFGABE 1

- Führen Sie den Farbsensor in den Unterricht ein und demonstrieren Sie dessen unterschiedliche Modi:
  – Farbe: Wahrnehmung von LEGO®-Systemfarben
  – Stärke des Umgebungslichts: Wahrnehmung unterschiedlicher Helligkeiten
  – Stärke des reflektierten Lichts: Erlaubt dem Roboter, einem vorgegeben Weg zu folgen.
- Fragen Sie Ihre Schüler, welche Teile eines Autos auf die Änderung des Umgebungslichts reagieren könnten. Die Antwort: Automatische Scheinwerfer. Sie gehen an, wenn die Helligkeit der Umgebung unter einen bestimmten Wert fällt.
- Die Schüler schreiben ein Programm, das diese Funktion simuliert. Dabei verwenden sie den Farbsensor im Modus "Stärke des Umgebungslichts vergleichen" sowie das Display des EV3-Steins, um die Scheinwerfer zu simulieren (hierzu können sie beispielsweise das Bild einer Glühbirne verwenden).
- Zeigen Sie den Schülern, wie man in der MINDSTORMS EV3-Software die Anschlussansicht verwendet, um Umgebungslichtwerte ablesen zu können.
- Die Schüler müssen Messungen des Umgebungslichts durchführen und sich entscheiden, ab welchem Wert das Display ausgelöst wird.
- Sie können Ihre Schüler auch auf den Robot Educator (Bauanleitung > Farbsensor nach vorne gerichtet - Fahrgestell) verweisen, falls diese Probleme beim Anschluss des Farbsensors haben. Natürlich dürfen die Schüler auch eigene Konstruktionen verwenden.



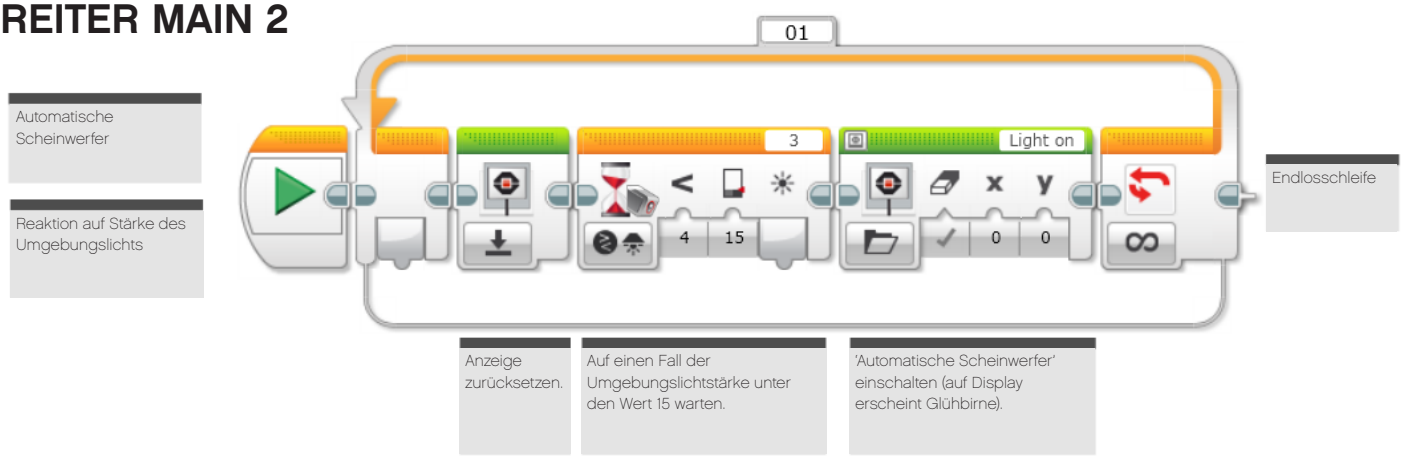
Anschlussansicht, welche die Intensität des Umgebungslichts anzeigt (Anschluss 3)

LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 4
REITER MAIN 1

HAUPTAUFGABE 2

- Die Schüler entwickeln ein Programm, das auf Änderungen der Stärke des Umgebungslichts reagiert und das die automatischen Scheinwerfer exakter simuliert. Das Programm sollte die 'Scheinwerfer' anschalten, wenn es 'dunkel' wird und sie ausschalten, wenn es wieder 'hell' wird.
- Führen Sie den Schleife-Block in den Unterricht ein und zeigen sie, wie dieser in der EV3-Software funktioniert.
- Erläutern Sie das Konzept einer Schleife und wie diese eingesetzt werden kann, um Programme endlos laufen zu lassen, bis sie von Hand angehalten werden.
- Betonen Sie, dass die Einstellungen für die Stärke des Umgebungslichts in den Programmen der Schüler kontinuierlich überwacht werden müssen und stetiges Debuggen nötig ist, um eine korrekte Funktion zu gewährleisten.

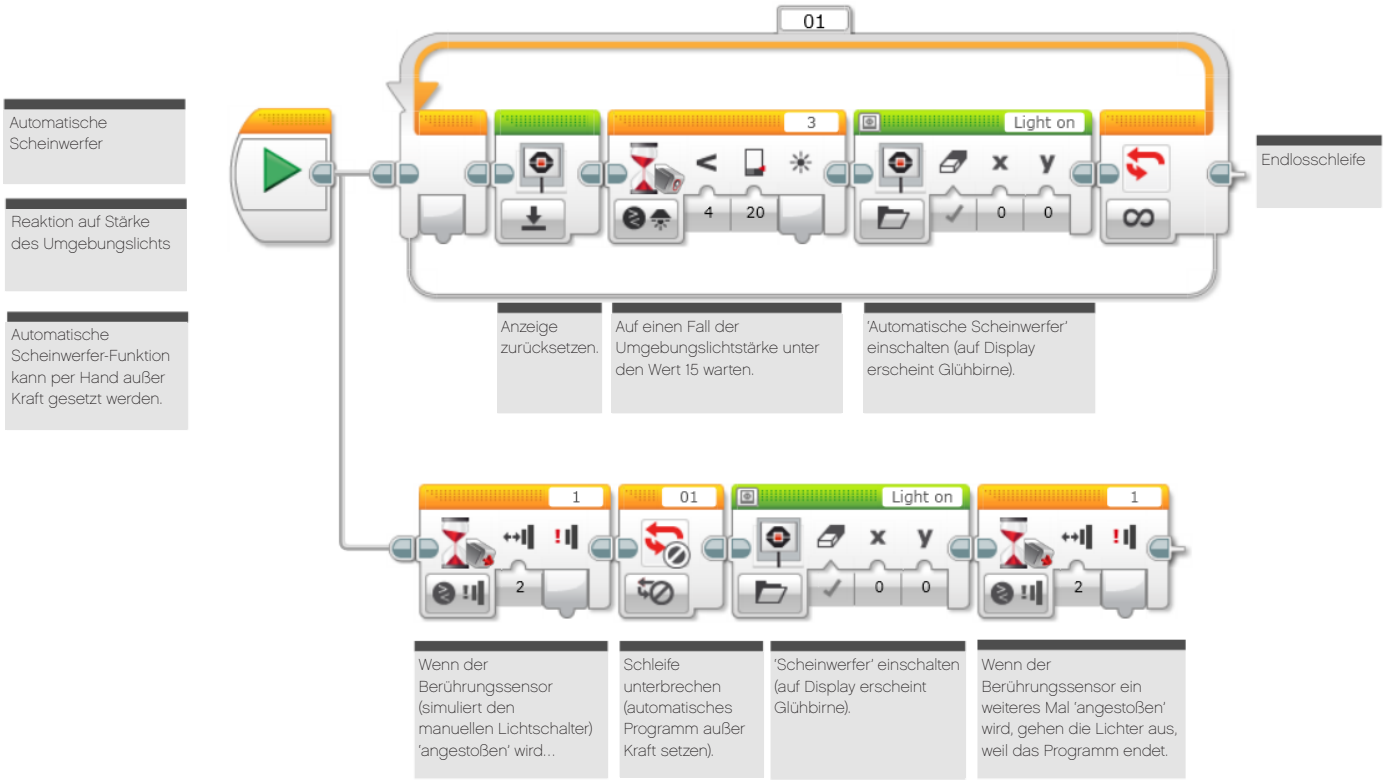
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 4
REITER MAIN 2



HAUPTAUFGABE 3

- Setzen Sie parallele Programmierung (Multitasking) ein, um automatische und manuelle Lichtschalter zu simulieren.
- Die Schüler integrieren Multitasking in ihre Software. Sie erweitern ihr 'Automatische Scheinwerfer'-Programm mit einer Option, die dem 'Fahrer' die Möglichkeit gibt, die Automatik auszuhebeln und die Lichter per Hand einzuschalten.
- Die Schüler verwenden einen Berührungssensor, um den manuellen Schalter zu simulieren.
- Durch diese Aufgabe wird der Umgang mit der Boole'schen Logik vertieft.
- Demonstrieren Sie den Schülern, wie man eine 'Leitung' vom Start-Block wegzieht, um ein paralleles Programm zu schaffen.
- Zeigen Sie den Schülern den Schleifen-Interrupt-Block und erläutern Sie, dass man diesen verwendet, um einen beliebigen Vorgang zu unterbrechen, der innerhalb der Schleife des Parallel-Programms abläuft.

LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 4
REITER MAIN 3



ABSCHLIESSENDE DISKUSSION

- Anmerkung: Für einen realistischeren Ansatz könnten die Schüler versuchen, den Anzeige-Block durch den Stein-Statusleuchte-Block zu ersetzen. Sie könnten auch beide kombinieren.
- Diese Unterrichtseinheit behandelt eine Vielfalt neuer Konzepte und führt diverse neue Blöcke der EV3-Software ein. Verwenden Sie nun Zeit darauf, diese Konzepte zu rekapitulieren und sicherzustellen, dass die Schüler deren Funktionsweise verstanden haben.
- Lassen Sie ein oder zwei Gruppen ihre Programmierlösungen vorstellen und diskutieren.
- Geben Sie eine Vorschau auf die nächste Unterrichtseinheit, in der die Schüler den Farbsensor auf andere Weise einsetzen werden, um das Fahrzeug noch selbstständiger zu machen.

AUFGABEN DIESER UNTERRICHTSEINHEIT

In dieser Unterrichtseinheit befassen wir uns mit einer Funktion des Farbsensors - seiner Fähigkeit, Veränderungen der Stärke des Umgebungslichts zu messen und darauf zu reagieren.

Automatische Scheinwerfer von Autos messen die Helligkeit der Umgebung und reagieren auf den gemessenen Wert (schalten sich also automatisch an und ab).

In dieser Einheit lernst du zudem, wie du deinem Rad-Roboter durch den Einsatz paralleler Programmierung (Multitasking) zwei Befehle gleichzeitig geben kannst.

AUFGABE 1

Bei manchen modernen Autos gehen die Scheinwerfer selbstständig an, wenn es dunkel wird.

Kannst du ein Programm schreiben, das solche automatischen Scheinwerfer simuliert?

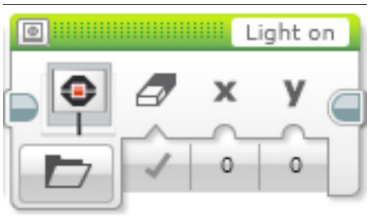
Findest du ein 'Glühbirnen'-Bild für das Display des EV3, das du in deinem Programm verwenden kannst?

Du brauchst den Farbsensor, um die Glühbirne anzuschalten.

Du benötigst die Umgebungslicht-Werte aus der Anschlussansicht, um die ordnungsgemäße Funktion deines Programms zu gewährleisten.

Anmerkung: Finde heraus, ob es eine Möglichkeit gibt, den Anzeige-Block durch den Stein-Statusleuchte-Block zu ersetzen - oder sogar beide zu verwenden!

Geeignete Blöcke



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 2

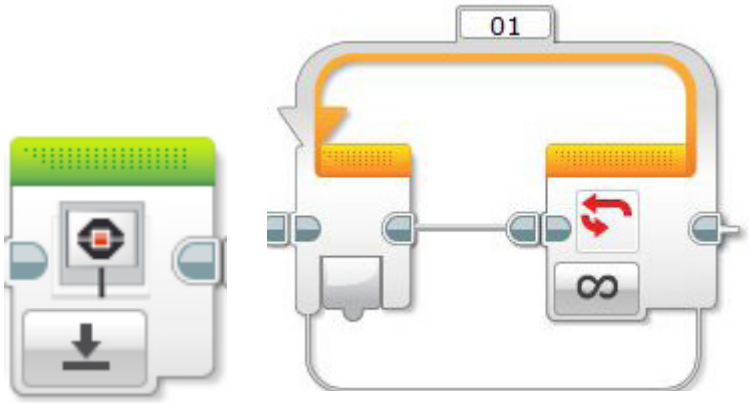
Nachdem deine automatischen Scheinwerfer nun erfolgreich in der 'Dunkelheit' angehen, sollen sie auch von selbst erlöschen, wenn es wieder 'hell' wird.

Um das zu erreichen, musst du dafür sorgen, dass sich dein Programm ständig wiederholt. So musst du es nicht immer wieder neu starten.

Anmerkung: Auch hier könntest du versuchen, den Anzeige-Block durch den Stein-Statusleuchte-Block zu ersetzen - oder sogar beide zu verwenden!

Geeignete Blöcke

Verwende dieselben Blöcke wie in der ersten Programmieraufgabe, aber erwäge auch den Einsatz der folgenden:



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 3

Was ist, wenn du als Fahrer Kontrolle über die automatischen Scheinwerfer haben willst und sie auch per Hand an- und ausschalten möchtest?

Moderne Autos mit Lichtautomatik geben dem Fahrer gewöhnlich die Möglichkeit, das automatische Programm außer Kraft zu setzen.

Kannst du dies in deinem Programm durch den Einsatz von paralleler Programmierung (bzw. Multitasking) simulieren?

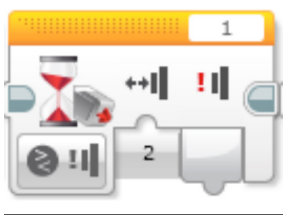
Du könntest einen Berührungssensor verwenden, um den manuellen Schalter zu simulieren.

Tipp: Du benötigst auch den Schleifen-Interrupt-Block, um die Automatik außer Kraft zu setzen.

Anmerkung: Und auch diesmal könntest du versuchen, den Anzeige-Block durch den Stein-Statusleuchte-Block zu ersetzen - oder sogar beide zu verwenden!

Geeignete Blöcke

Verwende dieselben Blöcke wie in der ersten und zweiten Programmieraufgabe, aber erwäge auch den Einsatz der folgenden:



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

Nach dem Programmieren ist es wichtig, seine Gedanken und Beobachtungen niederzuschreiben.

Denke über folgende Punkte nach und notiere dann im Kasten unten, wie diese Programmiersitzung verlaufen ist.

- Wie könntest du dein Programm verbessern?
- Könnte dein Programm geradliniger sein? Hast du zu viele Blöcke verwendet? Lässt sich das Programm effizienter gestalten?
- Wo könnte dein Programm in der 'realen Welt' Anwendung finden?

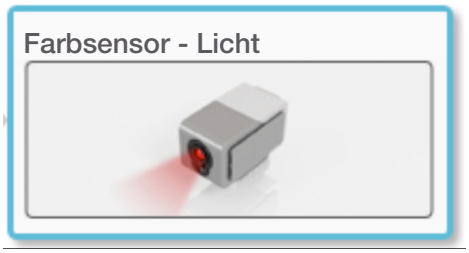
Gedanken und Beobachtungen

ROBOT EDUCATOR-ANLEITUNGEN

Die folgenden Robot Educator-Anleitungen helfen Lehrern und Schülern bei der Lösung der Aufgaben.

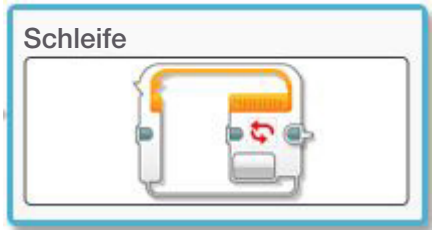
NEUE ROBOT EDUCATOR-ANLEITUNGEN

Hardware > Farbsensor - Licht



Komplexere Programme > Multitasking

Komplexere Programme > Schleife



Bauanleitungen > Farbsensor nach vorne gerichtet - Fahrgestell

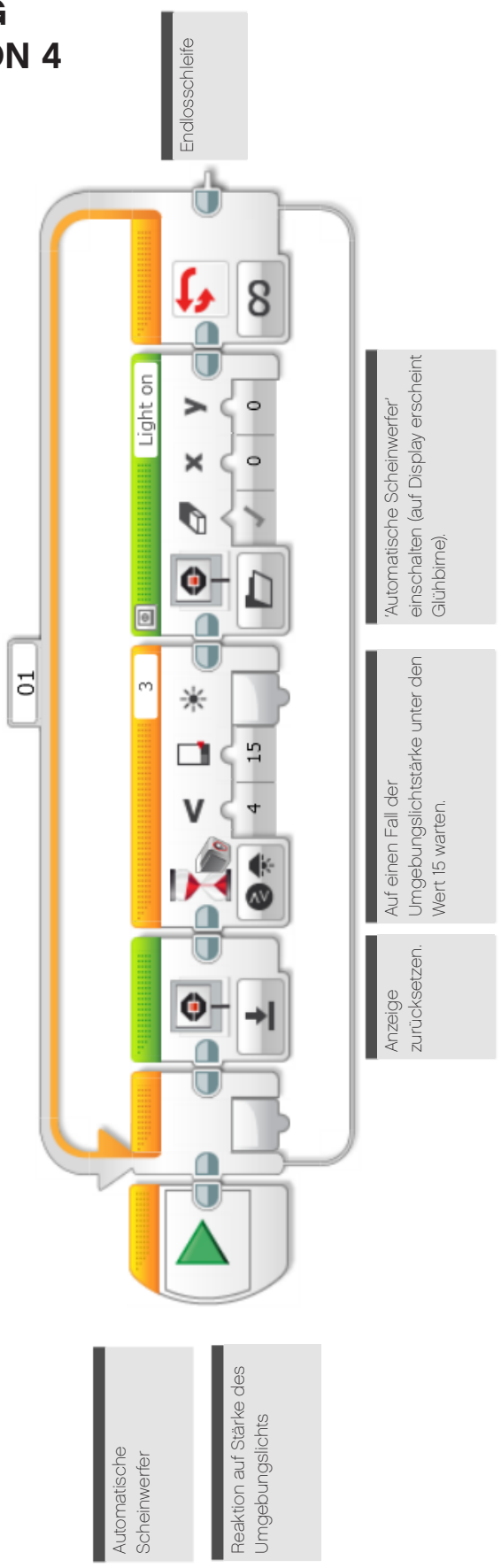
BEREITS BEHANDELTE ROBOT EDUCATOR-ANLEITUNGEN

Hardware > Stein-Anzeige

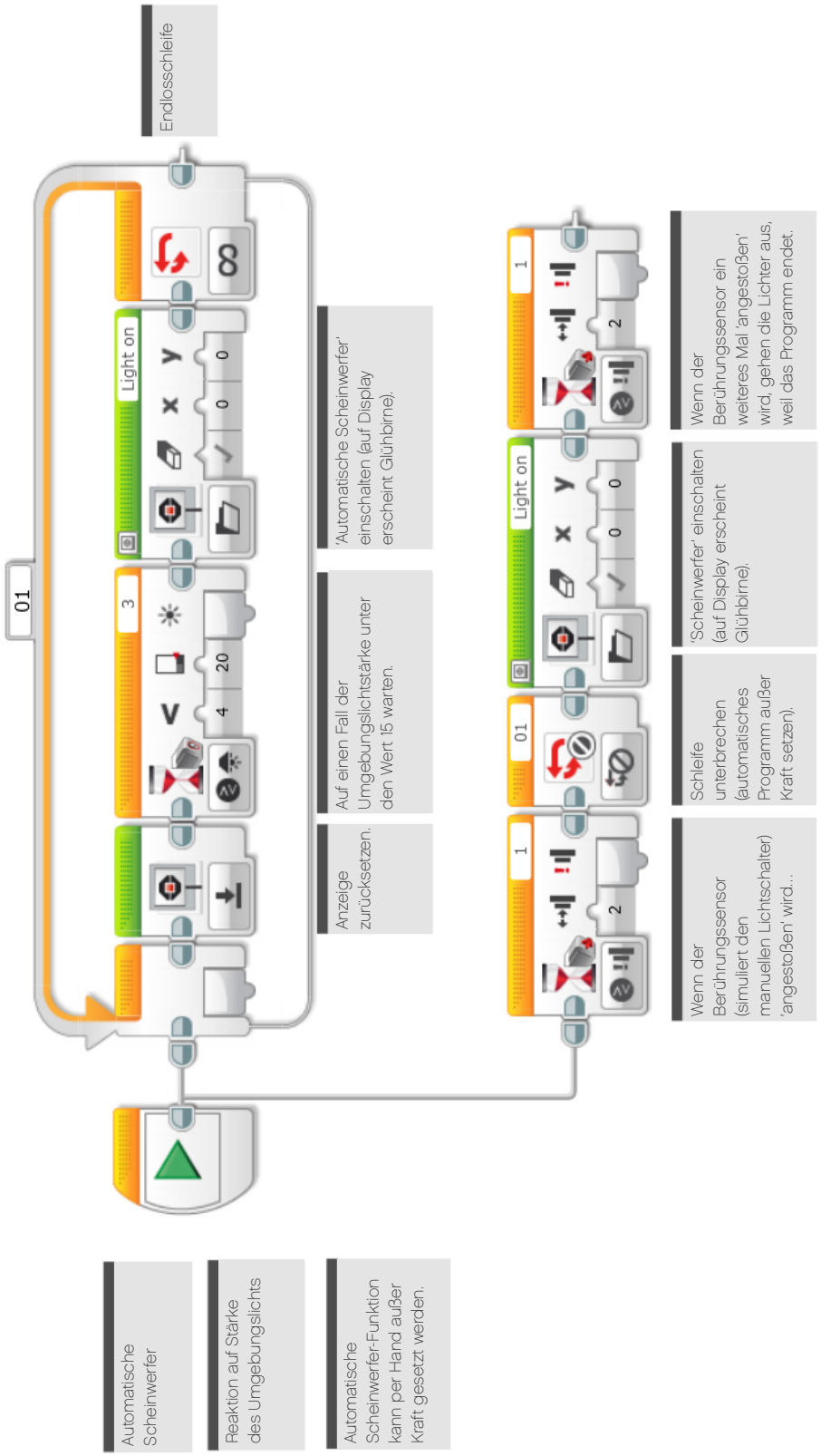
ANHANG FÜR UNTERRICHTSEINHEIT 4: BILDER UND PROGRAMME

LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 4
REITER MAIN 1

LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 4
REITER MAIN 2



LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 4
REITER MAIN 3



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session4_1.c

LEGO Start Page | Lesson 4_1.c*

```
#pragma config(Sensor, S1, touchSensor, sensorEV3_Touch)
#pragma config(Sensor, S2, gyroSensor, sensorEV3_Gyro)
#pragma config(Sensor, S3, colorSensor, sensorEV3_Color, modeEV3Color_Ambient)
#pragma config(Sensor, S4, sonarSensor, sensorEV3_Ultrasonic)
#pragma config(Motor,  motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor,  motorC, leftMotor, tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//

task main()
{
  //Wait for the level of ambient light intensity to drop below 15.
  while(getColorAmbient(colorSensor) > 15)
  {
    //Do nothing while we're waiting
  }

  //Display words "Light on" on the LCD Screen.
  drawTextAt(0, 0, "Light on");

  //Stay on for 5 seconds.
  sleep(5000);
}
```

Automatische Scheinwerfer

Reaktion auf Intensität des Umgebungslichts

Auf einen Fall der Umgebungslichtstärke unter den Wert 15 warten.

'Automatische Scheinwerfer' einschalten (auf Display erscheint Glühbirne).

Licht 5 Sekunden lang anlassen.

LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session4_2.c

LEGO Start Page | Lesson 4_2.c*

```
#pragma config(Sensor, S1, touchSensor, sensorEV3_Touch)
#pragma config(Sensor, S2, gyroSensor, sensorEV3_Gyro)
#pragma config(Sensor, S3, colorSensor, sensorEV3_Color, modeEV3Color_Ambient)
#pragma config(Sensor, S4, sonarSensor, sensorEV3_Ultrasonic)
#pragma config(Motor,  motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor,  motorC, leftMotor, tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//

task main()
{
  //Loop forever
  while(true)
  {
    //Erase the display to clear the LCD screen
    eraseDisplay();

    //Wait for the level of ambient light intensity to drop below 15.
    while(getColorAmbient(colorSensor) > 15)
    {
      //Do nothing while we're waiting
    }

    //Display words "Light on" on the LCD Screen.
    drawTextAt(0, 0, "Light on");
  }
}
```

LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session4_2_alternative.c

Eine alternative Lösung wie hier wird angeboten, wenn die Programmierer nicht nur eine Kopie des EV3-Programms angefertigt, sondern auch ein zweites textbasiertes Programm von maximaler Effizienz entwickelt haben.

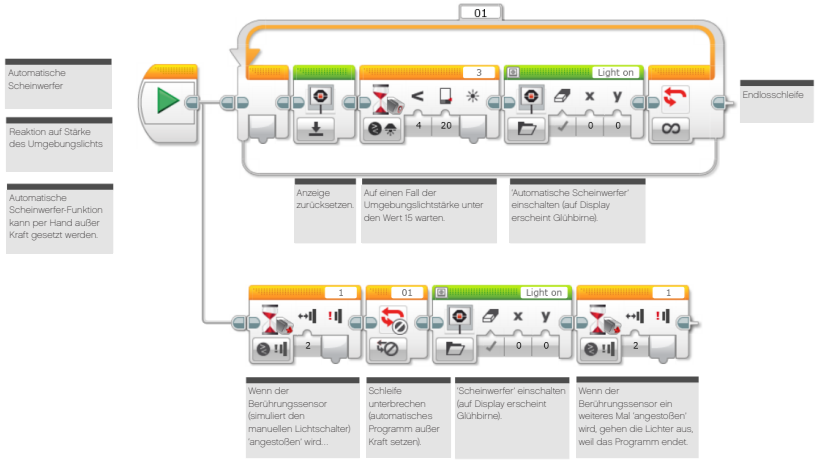
LEGO Start Page | Lesson 4_2_alternative.c*

```
1  #pragma config(Sensor, S1, touchSensor, sensorEV3_Touch)
2  #pragma config(Sensor, S2, gyroSensor, sensorEV3_Gyro)
3  #pragma config(Sensor, S3, colorSensor, sensorEV3_Color, modeEV3Color_Ambient)
4  #pragma config(Sensor, S4, sonarSensor, sensorEV3_Ultrasonic)
5  #pragma config(Motor,  motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
6  #pragma config(Motor,  motorC, leftMotor, tmotorEV3_Large, PIDControl, driveLeft, encoder)
7  //*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//
8
9  task main()
10 {
11   while(true)
12   {
13     if(getColorAmbient(colorSensor) > 15)
14     {
15       eraseDisplay();
16     }
17     else
18     {
19       drawTextAt(0, 0, "Light on");
20     }
21   }
22 }
23
```

LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session4_3.c

LEGO Start Page | Lesson 4_3.c*

```
#pragma config(Sensor, S1, touchSensor, sensorEV3_Touch)
#pragma config(Sensor, S2, gyroSensor, sensorEV3_Gyro)
#pragma config(Sensor, S3, colorSensor, sensorEV3_Color, modeEV3Color_Ambient)
#pragma config(Sensor, S4, sonarSensor, sensorEV3_Ultrasonic)
#pragma config(Motor,  motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor,  motorC, leftMotor, tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//

task monitorTouchSensor()
{
  resetBumpedValue(touchSensor);

  //Wait for Touch Sensor #1 to be bumped
  while(getBumpedValue(touchSensor) == 0)
  {
    //Do nothing until bumped.
  }

  //Stop task "main" (which is task #0)
  stopTask(0);

  //Display words "Light on" on the LCD Screen.
  drawTextAt(0, 0, "Light on");

  resetBumpedValue(touchSensor);

  //Wait for Touch Sensor #1 to be bumped
  while(getBumpedValue(touchSensor) == 0)
  {
    //Do nothing until bumped.
  }
}

task main()
{
  startTask(monitorTouchSensor);

  //Loop forever
  while(true)
  {
    //Erase the display to clear the LCD screen
    eraseDisplay();

    //Wait for the level of ambient light intensity to drop below 15.
    while(getColorAmbient(colorSensor) > 15)
    {
      //Do nothing while we're waiting
    }

    //Display words "Light on" on the LCD Screen.
    drawTextAt(0, 0, "Light on");
  }
}
```

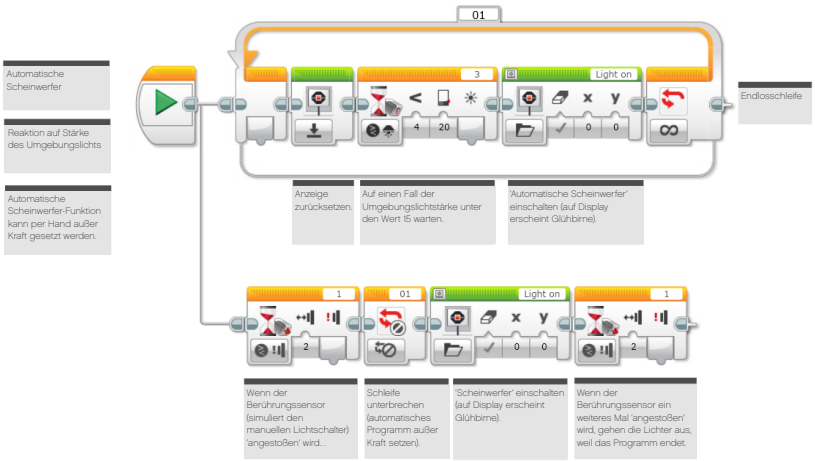


LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session4_3_alternative.c

Eine alternative Lösung wie hier wird angeboten, wenn die Programmierer nicht nur eine Kopie des EV3-Programms angefertigt, sondern auch ein zweites textbasiertes Programm von maximaler Effizienz entwickelt haben.

LEGO Start Page | Lesson 4_3_alternative.c*

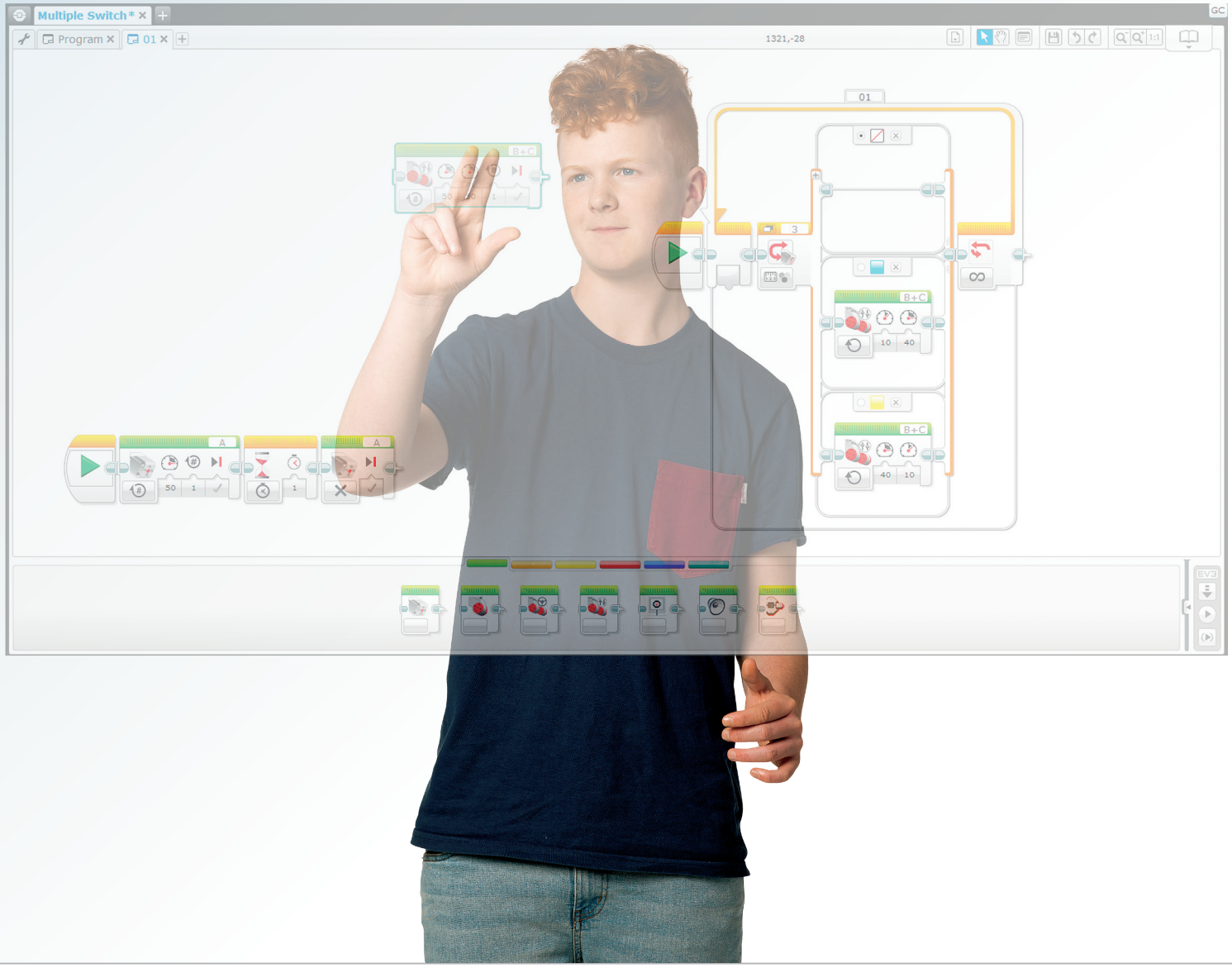
```
#pragma config(Sensor, S1, touchSensor, sensorEV3_Touch)
#pragma config(Sensor, S2, gyroSensor, sensorEV3_Gyro)
#pragma config(Sensor, S3, colorSensor, sensorEV3_Color, modeEV3Color_Ambient)
#pragma config(Sensor, S4, sonarSensor, sensorEV3_Ultrasonic)
#pragma config(Motor,  motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor,  motorC, leftMotor, tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//

task main()
{
  resetBumpedValue(touchSensor);

  //Loop forever
  while(getBumpedValue(touchSensor) == 0)
  {
    //Wait for the level of ambient light intensity to drop below 15.
    if(getColorAmbient(colorSensor) > 15)
    {
      //Erase the display to clear the LCD screen
      eraseDisplay();
    }
    else
    {
      //Display words "Light on" on the LCD Screen.
      drawTextAt(0, 0, "Light on");
    }
  }

  //Display words "Light on" on the LCD Screen.
  drawTextAt(0, 0, "Light on");

  resetBumpedValue(touchSensor);

  while(getBumpedValue(touchSensor) == 0)
  {
    //Do nothing until pressed.
  }
}
```



mindstorms education EV3

UNTERRICHTSEINHEIT 5

Ampeln und automatische Schienensysteme



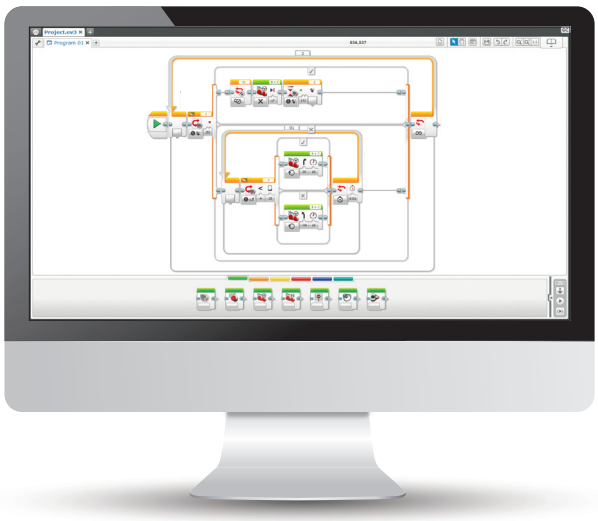
LEGOeducation.de

LEGO education

Ampeln und automatische Schienensysteme

In dieser Unterrichtseinheit vertiefen die Schüler ihr Verständnis des Farbsensors und knüpfen bei der Entwicklung eines Programms zur Simulation eines 'Stop&Go'-Ampelsystems an ihre vorherigen Erfahrungen mit dem Schleife-Block an.

Zudem programmieren sie ihren Rad-Roboter so, dass dieser einem vorgegebenen Weg (bzw. einer Linie) folgt, und sie lernen den Schalter-Block kennen.



Unterrichtseinheit 5
Stundenplanung

ERGEBNISSE

In dieser Einheit lernen die Schüler,

- dass Algorithmen eine Serie von Befehlen in einer bestimmten Reihenfolge ausführen können.
- ihr Verständnis der Boole'schen Logik und deren Einsatzmöglichkeiten zu vertiefen.
- den Warte-Block in Beziehung zum Farbsensor einzusetzen.
- dass der Farbsensor mehrere Funktionen hat und eine ganze Reihe von Parametern messen und auf sie reagieren kann.
- den Farbsensor so zu verwenden, dass dieser LEGO®-Systemfarben erkennen und die Stärke reflektierten Lichts messen kann.
- ihr Verständnis des Schleife-Blocks zu erweitern.
- das grundlegende Konzept eines Schalters zu verstehen und wie dieser für 'Wahr'- und 'Falsch'-Operationen verwendet wird.

BEGRIFFE

Eingabe, Ausgabe, Algorithmus, Warten, Farbsensor, debuggen, Umgebungslicht, reflektiertes Licht, Schleife, Boole'sche Logik, Schalter.

EINFÜHRUNG

- Sagen Sie den Schülern, dass sie ein weiteres Mal mit dem Farbsensor arbeiten werden. Diesmal befassen sie sich allerdings mit dessen Fähigkeit, LEGO®-Systemfarben zu erkennen und auf diese zu reagieren. Dabei vertiefen die Schüler auch ihr Verständnis dafür, wie der Sensor auf Licht reagiert. In dieser Einheit müssen sie sich reflektierten Lichts bedienen, um ihren Rad-Roboter über eine vorgegebene Strecke zu bewegen. Das Gesamtprojekt umfasst drei Aufgaben.
- Erläutern Sie den Schülern, dass der Einsatz des Farbsensors den Rad-Roboter selbstständiger macht. So soll simuliert werden, wie ein 'Roboter-Auto' auf Verkehrsampeln reagiert.
- Lassen sie die Schüler überlegen, wo selbstständige Fahrzeuge im Alltag eine Rolle spielen (z.B. bei den Terminal-Zubringerzügen der Flughäfen Frankfurt oder London Heathrow).
- Die Schüler werden in dieser Einheit zudem herausfinden, wie Schalter arbeiten und wie sich diese in ihre Programme integrieren lassen.
- Merken Sie an, dass die Schüler auch die Funktion von Schleifen und ihre Verwendung in Programmen untersuchen werden.

Unterrichtseinheit 5
Stundenplanung

HAUPTAUFGABE 1

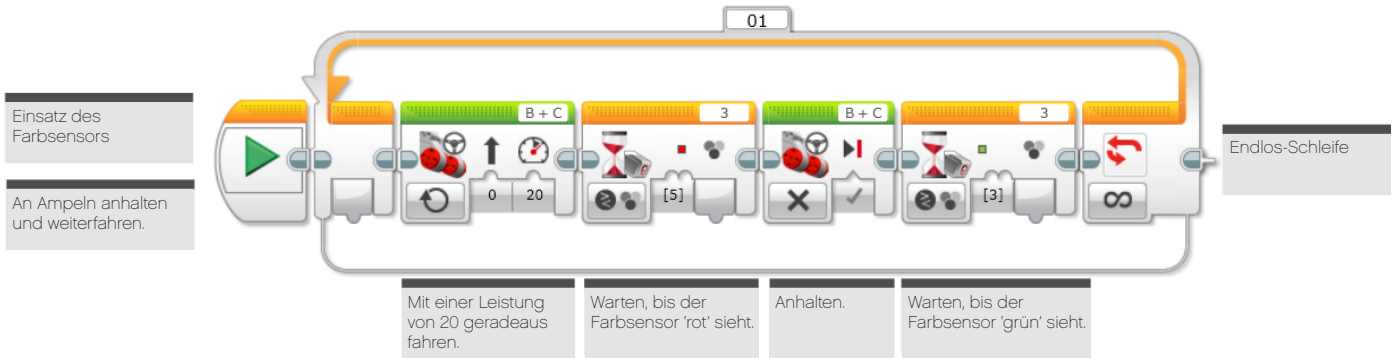
- Die Schüler programmieren ihren Rad-Roboter so, dass er den Tisch entlang fährt und an einer roten 'Ampel' anhält. Auf diese Weise beginnen sie damit, die Funktion des Farbsensors zur Wahrnehmung von LEGO®-Systemfarben zu erforschen.
- Um die Aufgabe zu lösen, müssen sie den Warten-Block verwenden. Merken Sie an, dass der Roboter wahrgenommene Farben ausschließen oder ignorieren kann.
- Die Schüler entwickeln ein Programm, das die Motoren des Rad-Roboters stoppt, wenn der Farbsensor ein rotes Licht wahrnimmt.
- Die Schüler müssen ihrem Farbsensor 'mitteilen', dass er die Farbe rot wahrnehmen soll.
- Unter Umständen ist es hilfreich, wenn die Schüler mit einer ganzen Palette von Farben experimentieren, um sich in das Konzept zu vertiefen und die gewünschte Funktion ihres Programms sicherzustellen.

LÖSUNGSVORSCHLAG DATEINAME CS LESSON 5 REITER MAIN 1

HAUPTAUFGABE 2

- In dieser Aufgabe müssen die Schüler auf exaktere Weise das Konzept der Verkehrsampel simulieren, indem sie ihren Rad-Roboter auf eine Serie von grünen und roten Signalen reagieren lassen.
- Befassen Sie sich erneut mit dem Schleife-Block, wiederholen Sie das Konzept einer Schleife und deren Vermögen, ein Programm endlos laufen zu lassen, bis es von Hand beendet wird.
- Durch die Anordnung der vorher verwendeten Befehls-Blöcke innerhalb einer Schleife kann eine Strecke mit mehreren 'Ampeln' versehen werden.
- Weisen Sie die Schüler darauf hin, dass alle anderen Farben abgewählt werden müssen, damit der Farbsensor möglichst effektiv auf die relevanten Farben (rot und grün) reagiert.

LÖSUNGSVORSCHLAG DATEINAME CS LESSON 5 REITER MAIN 2



HAUPTAUFGABE 3

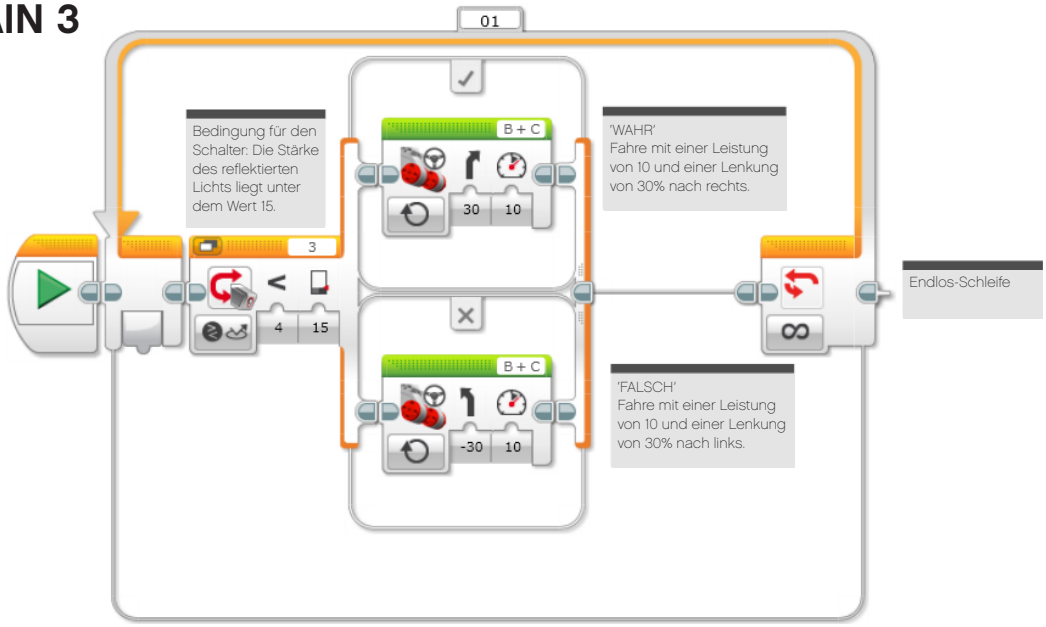
Einer Linie folgen - Bau eines automatisierten, führerlosen Fahrzeugs.

- Bei dieser Aufgabe müssen die Schüler darüber nachdenken, wie sie ein automatisiertes Fahrzeug eine Straße oder einen Kurs entlang leiten können. Wie lässt sich ein Rad-Roboter entwickeln, der unter Einsatz reflektierten Lichts einer Linie folgt?
- Die Schüler müssen mit dem Schalter-Block (der innerhalb einer Schleife platziert wird) vertraut gemacht werden. Erklären Sie den Schülern, dass der Schalter-Block zur Entwicklung eines Programms verwendet werden kann, das den Rad-Roboter selbstständig agieren lässt.
- Erläutern Sie, dass sich mit dem Schalter-Block der Ablauf eines Programms kontrollieren lässt, und dass ein Standard-Schalter, der den Berührungssensor verwendet, ein klassisches Beispiel Boole'scher Logik ist. Zeigen Sie den Schülern, wie der Schalter auf den Farbsensor eingestellt wird und erklären Sie den Begriff des Schwellenwerts. Der Schwellenwert wird verwendet, um eine Wahr/Falsch-Entscheidung zu treffen (reagiere oberhalb des Schwellenwerts auf eine, unterhalb auf andere Weise).
- Machen Sie den Schülern deutlich, dass sich der Rad-Roboter in dieser Aufgabe an einer Linie entlang schlängeln muss. Er bewegt sich nach links bzw. rechts, je nachdem, in welche Richtung der Schwellenwert überschritten wird. Weisen Sie darauf hin, dass die Standardsteuerung-Blöcke auf 'An' stehen müssen.
- Sobald der Rad-Roboter der Linie folgt: Können die Schüler ihn verbessern, so dass er sich mehr wie ein Auto verhält, also geradeaus fährt und keine Schlangenlinien?
- Sie werden einige Zeit darauf verwenden müssen, das Konzept eines Schalters zu erklären und ihn als Beispiel Boole'scher Logik zu präsentieren.
- Die Schüler setzen erneut den Farbsensor ein, diesmal muss er aber so programmiert werden, dass er auf die Stärke reflektierten Lichts reagiert.
- Tipp: Die Schüler müssen die Stärke des reflektierten Lichts in der Anschlussansicht ablesen, um abschätzen zu können, welcher Wert in den Warten-Block eingegeben werden muss.
- Tipp: Am besten lässt sich diese Aufgabe durch Verwendung von schwarzem Klebeband auf einer sehr hellen (oder weißen) Oberfläche angehen.
- Eine mögliche Erweiterung der Aufgabe wäre das Hinzufügen eines zweiten Farbsensors sowie die Kombination des Folge-der-Linie- mit dem Ampel-Programm zur Simulation automatisierten öffentlichen Verkehrs.

LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 5
REITER MAIN 3

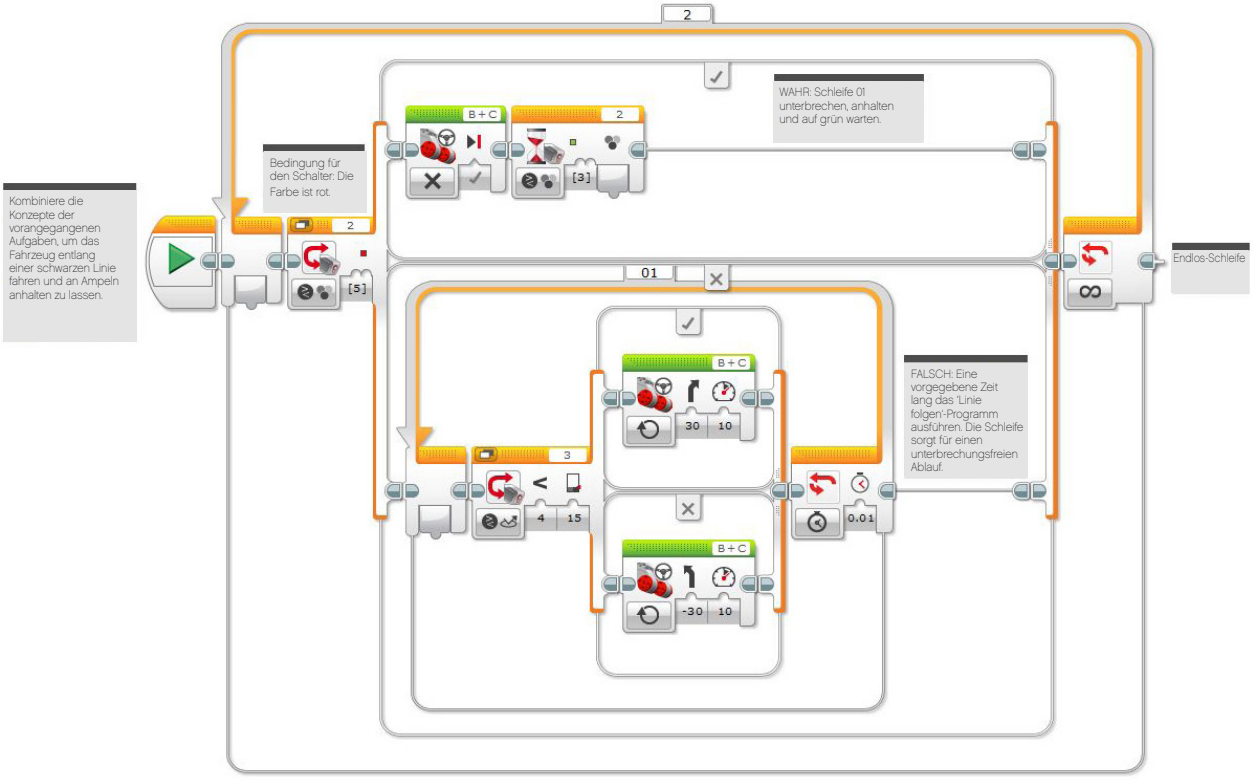
Einsatz des Farbsensors

Unterschiedliche Intensität reflektierten Lichts ausnutzen, um einer Linie zu folgen (aufbauend auf dem Verständnis von Umgebungslicht).



LÖSUNGSVORSCHLAG FÜR ERWEITERUNG DER AUFGABE
DATEINAME CS LESSON 5
REITER EXTENSION

Kombiniere die Konzepte der vorangegangenen Aufgaben, um das Fahrzeug entlang einer schwarzen Linie fahren und an Ampeln anhalten zu lassen.



AUFGABEN DIESER UNTERRICHTSEINHEIT

In dieser Einheit verwendest du den Farbsensor und den Schalter-Block, um (boolesche) Entscheidungen zu treffen. Der Rad-Roboter trifft seine Entscheidungen anhand wahrgenommener Farben.

AUFGABE 1

Beim Autofahren ist es wichtig, Verkehrssituationen zu erkennen und Verkehrsregeln zu beachten.

Was tut ein Verkehrsteilnehmer, wenn er sich einer Ampel nähert?

Autonome Fahrzeuge würden eine Art Sensor benötigen, um Ampeln wahrzunehmen und selbstständig auf sie reagieren zu können.

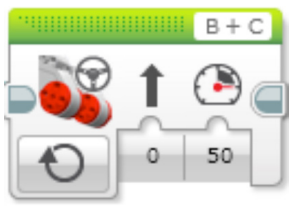
Bei dieser Aufgabe musst du deinen Rad-Roboter so programmieren, dass er auf ein 'Stopp'-Kommando reagiert. Welche Farbe solltest du in deinem Programm verwenden?

Bediene dich des Warten-Blocks, um den Farbsensor so zu programmieren, dass er Rot wahrnimmt und den Rad-Roboter anhält.

Verfeinere dein Programm, so dass du deinen Rad-Roboter in einer angemessenen Entfernung von der Ampel anhalten lässt.

Stelle sicher, dass der Rad-Roboter nur auf Rot reagiert, indem du alle anderen Farben ausschließt.

Geeignete Blöcke



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 2

Du hast deinen Rad-Roboter so programmiert, dass er an einer Ampel anhält. Nun soll er wieder weiterfahren!

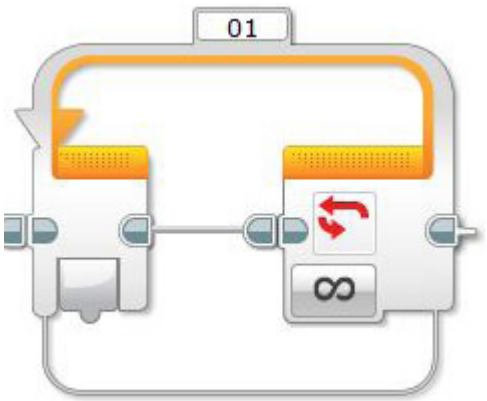
Entwickle ein Programm, das den Farbsensor nutzt, um die beiden Kommandos 'Stopp' und 'Weiterfahren' zu erkennen und darauf zu reagieren.

Welche Farben verwendest du?

Was wäre, wenn mehrere Ampeln entlang der Straße stünden? Kannst du dein Programm so verändern, dass sich der 'Stopp - Weiterfahren'-Algorithmus wiederholen lässt?

Geeignete Blöcke

Verwende dieselben Blöcke wie in der ersten Programmieraufgabe, aber erwäge auch den Einsatz des folgenden:



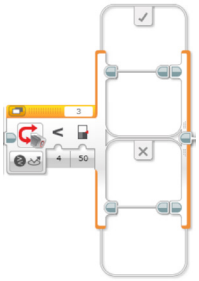
Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 3

Bei dieser Aufgabe machen wir den Rad-Roboter noch selbstständiger. Du musst dein Modell ein wenig umbauen, damit der Farbsensor nach unten zeigt. Stelle dir ein Auto vor, das auf 'Autopilot' eine vorgegebene Strecke entlang fährt - ein bisschen so wie die führerlosen Züge, die man an manchen Flughäfen findet.

Deine Aufgabe ist, den Rad-Roboter so zu programmieren, dass er genau das tut! Du musst ein Programm entwickeln, das eine schwarze Linie erkennt und ihr folgt. Dein Rad-Roboter soll sich entlang dieser Linie fortbewegen, ohne den Kontakt zu ihr zu verlieren. Du wirst dein Programm kontinuierlich austesten und debuggen müssen, um die Fahrt des Rad-Roboters möglichst unterbrechungsfrei und reibungslos zu machen.

Tipp: Du musst die Farbsensor-Einstellungen in der Anschlussansicht ändern, damit die Stärke des reflektierten Lichts gemessen wird.



Geeignete Blöcke

Verwende dieselben Blöcke wie in der ersten und zweiten Programmieraufgabe, aber erwäge auch den Einsatz des obigen.

Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

Unterrichtseinheit 5
Arbeitsblätter

Nach dem Programmieren ist es wichtig, seine Gedanken und Beobachtungen niederzuschreiben.

Denke über folgende Punkte nach und notiere dann im Kasten unten, wie diese Programmiersitzung verlaufen ist.

- Wie könntest du dein Programm verbessern?
- Könnte dein Programm geradliniger sein? Hast du zu viele Blöcke verwendet? Lässt sich das Programm effizienter gestalten?
- Wo könnte dein Programm in der 'realen Welt' Anwendung finden?

Gedanken und Beobachtungen

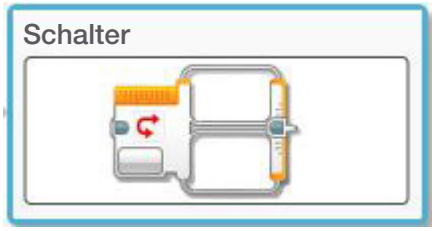
ROBOT EDUCATOR-ANLEITUNGEN

Die folgenden Robot Educator-Anleitungen helfen Lehrern und Schülern bei der Lösung der Aufgaben.

NEUE ROBOT EDUCATOR-ANLEITUNGEN

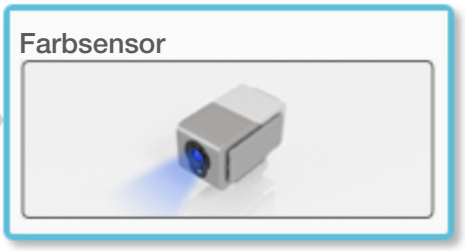
Grundlagen > An Linie stoppen

Komplexere Programme > Schalter

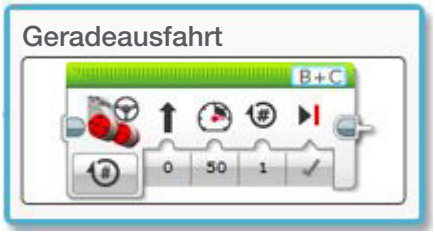


BEREITS BEHANDELTE ROBOT EDUCATOR-ANLEITUNGEN

Hardware > Farbsensor - Farbe

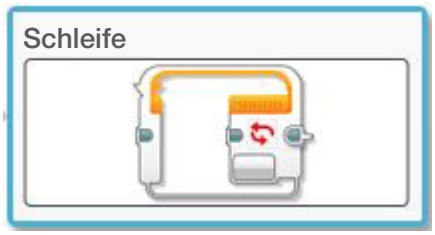


Grundlagen > Geradeausfahrt



Grundlagen > Kurvenfahrt

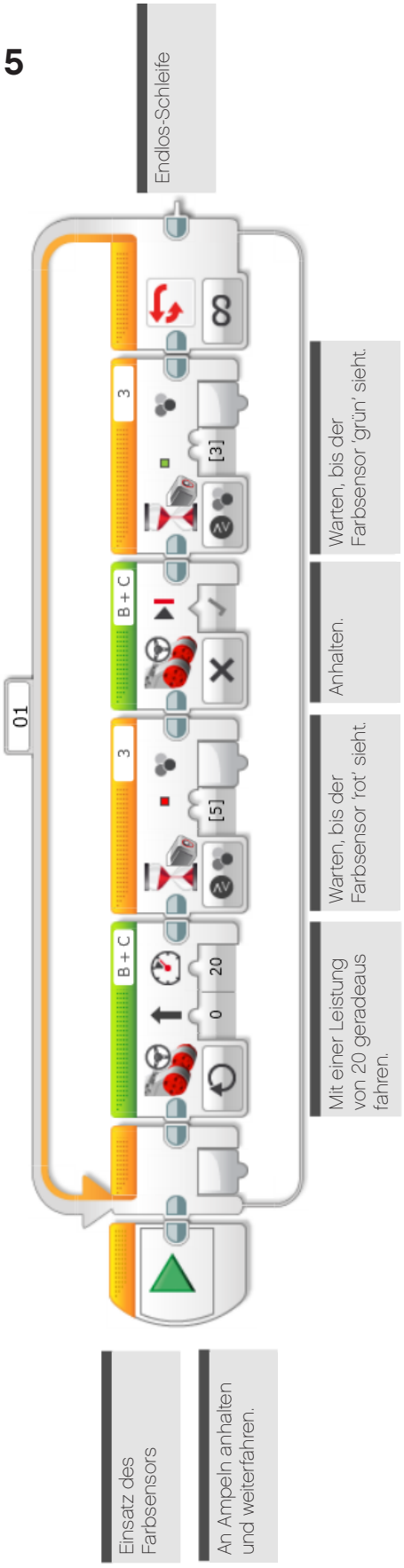
Komplexere Programme > Schleife



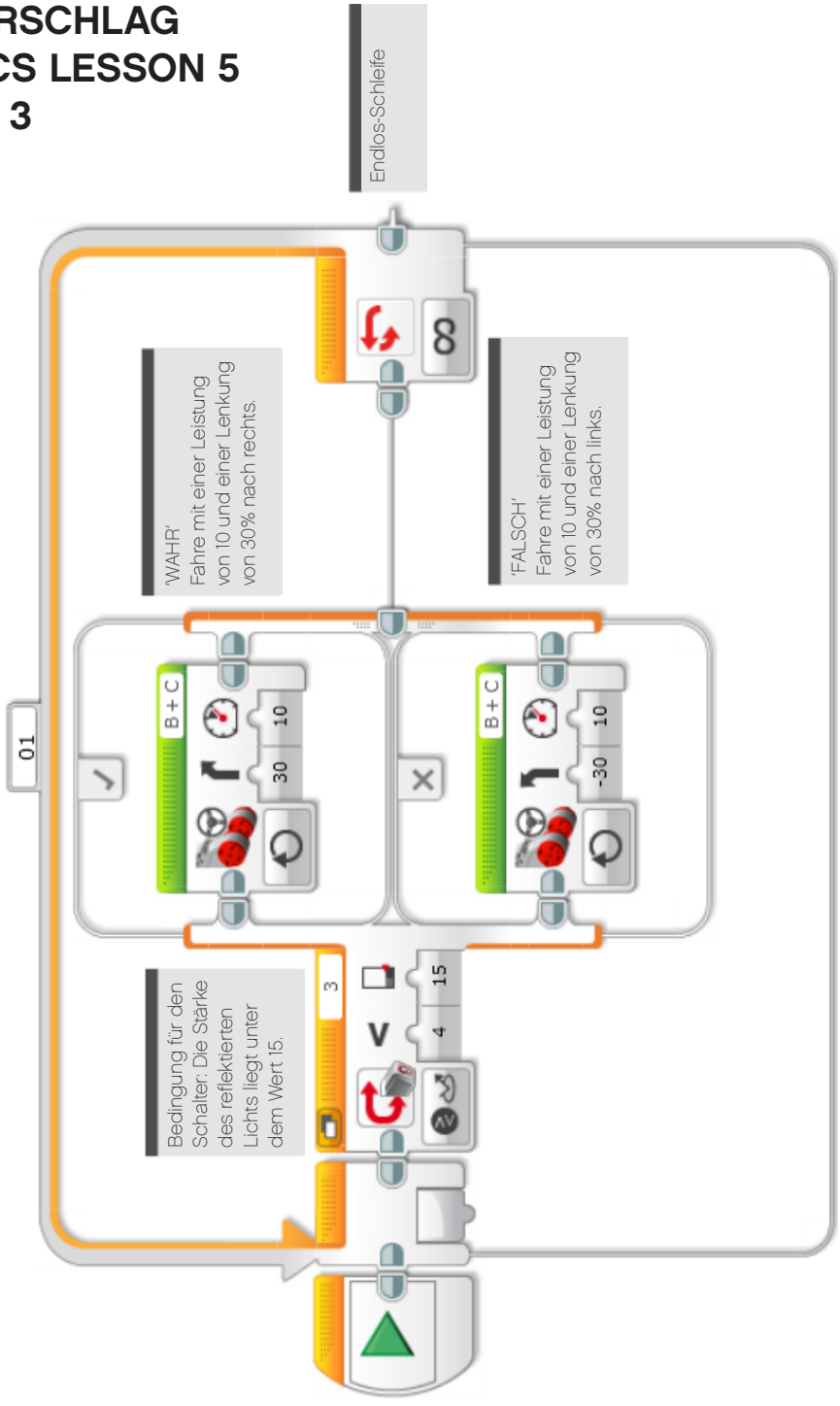
ANHANG FÜR UNTERRICHTSEINHEIT 5: BILDER UND PROGRAMME

LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 5
REITER MAIN 1

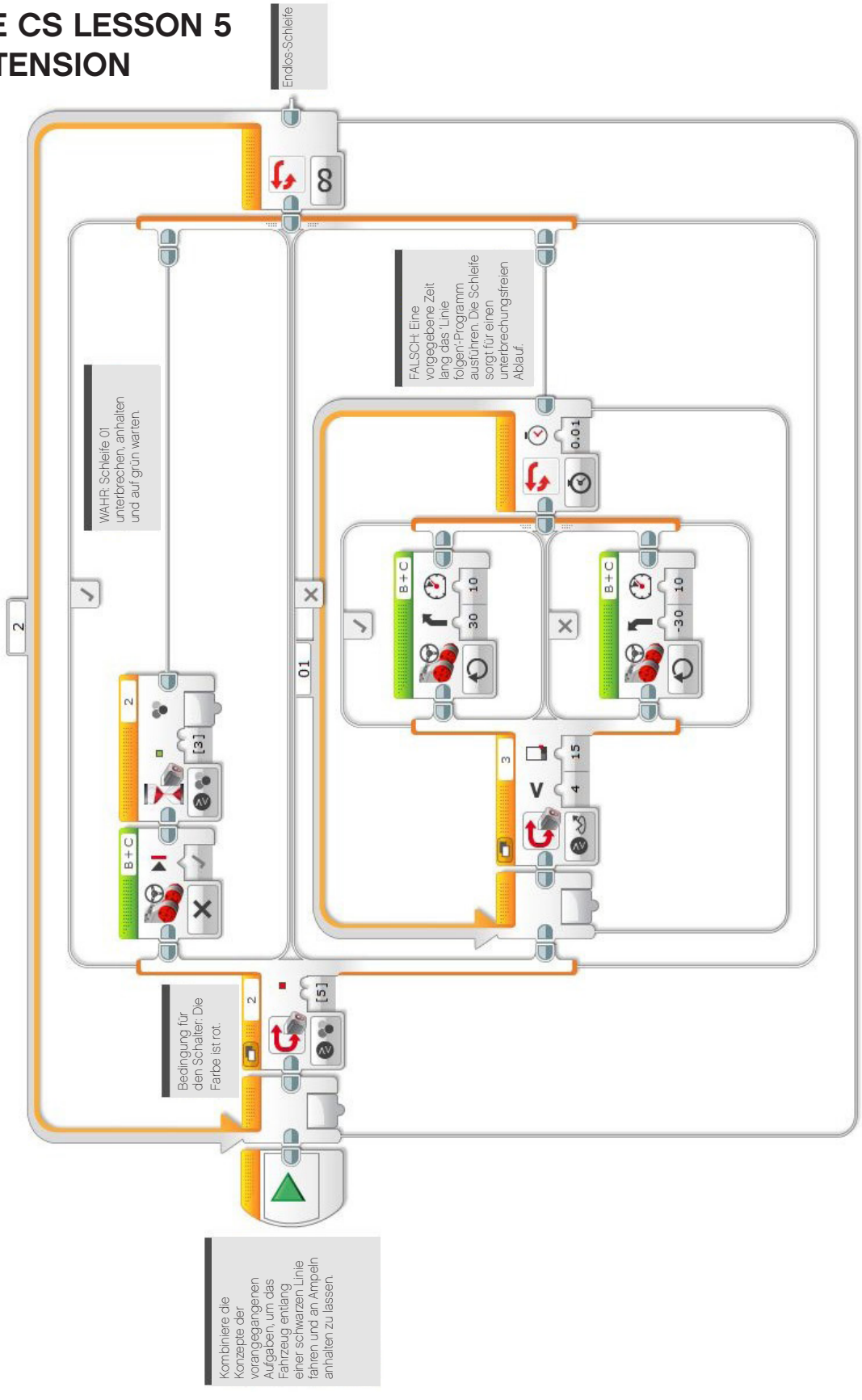
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 5
REITER MAIN 2



LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 5
REITER MAIN 3



LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 5
REITER EXTENSION



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session5_1.c

LEGO Start Page | Lesson 5_1.c*

```
1  #pragma config(Sensor, S1, touchSensor, sensorEV3_Touch)
2  #pragma config(Sensor, S2, gyroSensor, sensorEV3_Gyro)
3  #pragma config(Sensor, S3, colorSensor, sensorEV3_Color, modeEV3Color_Color)
4  #pragma config(Sensor, S4, sonarSensor, sensorEV3_Ultrasonic)
5  #pragma config(Motor,  motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
6  #pragma config(Motor,  motorC, leftMotor, tmotorEV3_Large, PIDControl, driveLeft, encoder)
7  //*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//
8
9  task main()
10 {
11   //Loop while the sensor does not see red.
12   while(getColorName(colorSensor) != colorRed)
13   {
14     //Drive Forward
15     setMotorSpeed(motorB, 20);
16     setMotorSpeed(motorC, 20);
17   }
18 } //End of program - turn off all motors automatically
19
```

Einsatz des Farbsensors

An einem vorgegeben Punkt anhalten.

Vorwärts fahren.

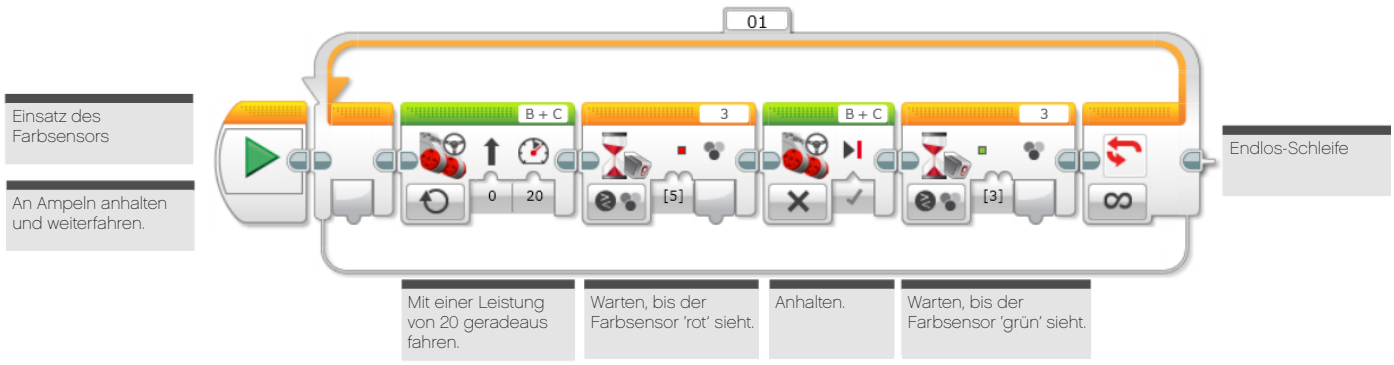
Warten, bis der Farbsensor 'rot' sieht.

Anhalten.

LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session5_2.c

LEGO Start Page | Lesson 5_2.c*

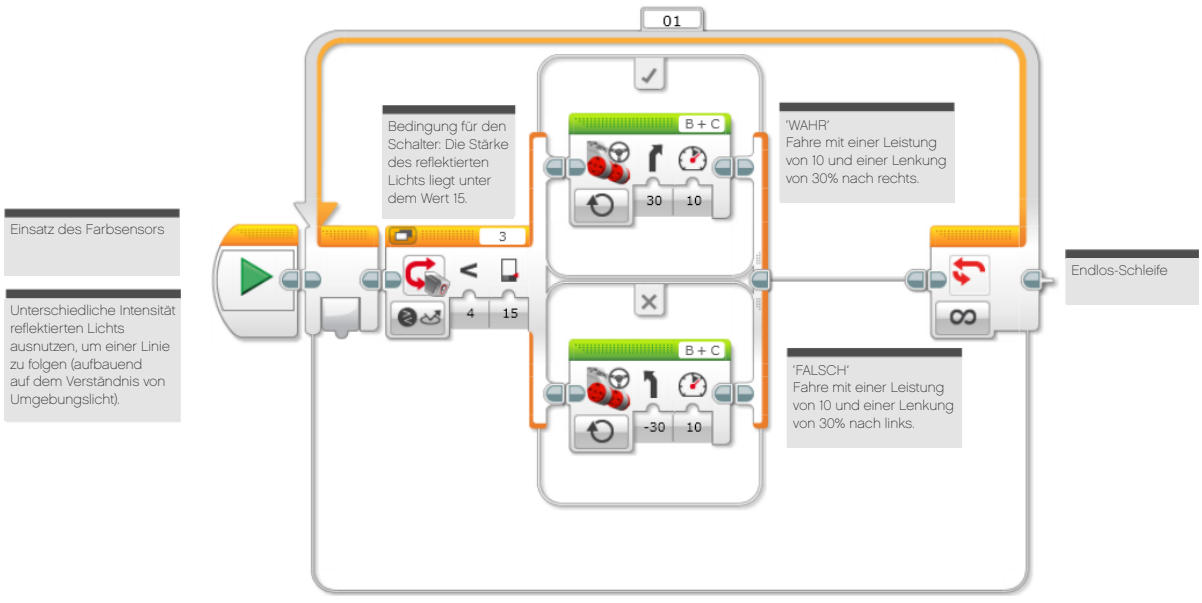
```
#pragma config(Sensor, S1, touchSensor, sensorEV3_Touch)
#pragma config(Sensor, S2, gyroSensor, sensorEV3_Gyro)
#pragma config(Sensor, S3, colorSensor, sensorEV3_Color, modeEV3Color_Color)
#pragma config(Sensor, S4, sonarSensor, sensorEV3_Ultrasonic)
#pragma config(Motor,  motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor,  motorC, leftMotor, tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//

task main()
{
  //Repeat Forever
  while(true)
  {
    //Loop while the sensor does not see red.
    while(getColorName(colorSensor) != colorRed)
    {
      //Drive Forward
      setMotorSpeed(motorB, 20);
      setMotorSpeed(motorC, 20);
    }

    //After we've seen red, we need to wait until we see green.
    while(getColorName(colorSensor) != colorGreen)
    {
      //Stop the Robot
      setMotorSpeed(motorB, 0);
      setMotorSpeed(motorC, 0);
    }
  }
} //End of program - turn off all motors automatically
```



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session5_3.c

LEGO Start Page | Lesson 5_3.c*

```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//
3
4  task main()
5  {
6    //Repeat Forever
7    while(true)
8    {
9      //If we see a dark value...
10     if(getColorReflected(colorSensor) < 15)
11     {
12       //Rotate away from the line.
13       setMotorSpeed(motorB, 10);
14       setMotorSpeed(motorC, 30);
15     }
16     //If we see a light value (or not dark)...
17     else
18     {
19       //Rotate towards the line.
20       setMotorSpeed(motorB, 30);
21       setMotorSpeed(motorC, 10);
22     }
23   }
24 }
25
```



mindstorms education EV3

UNTERRICHTSEINHEIT 6

Es piept beim Rückwärtsfahren

LEGOeducation.de

LEGO education

Es piept beim Rückwärtsfahren

Die Schüler beschäftigen sich erneut mit dem Ultraschallsensor und dem Konzept der Einparkhilfe (also Sensoren, die ein Warngeräusch auslösen, wenn sich das Fahrzeug einem Hindernis nähert).

In dieser Unterrichtseinheit entwickeln die Schüler Programme, die Parksensoren simulieren, wie man sie in modernen Autos finden kann. Bei der abschließenden Aufgabe müssen sie ein Programm schreiben, das eine Wiedergabe von warnenden Pieptönen bewirkt. Diese werden kürzer, je näher der Rad-Roboter einem Objekt kommt.

Die Schüler benötigen in dieser Unterrichtseinheit eine ganze Reihe von Programmiertechniken und -Blöcken, einschließlich paralleler Programmierung (Multitasking), Schleifen und Schalter. Sie lernen zudem Mathe-Blocks und Datenleitungen kennen.

Online-Quellen:

Es wird empfohlen, nach Online-Videos zu suchen, welche die Funktion von Parksensoren demonstrieren.

Unterrichtseinheit 6
Stundenplanung

ERGEBNISSE

In dieser Einheit lernen die Schüler,

- dass Algorithmen eine Serie von Befehlen in einer bestimmten Reihenfolge ausführen können.
- ihr Verständnis der Boole'schen Logik und deren Einsatzmöglichkeiten zu vertiefen.
- den Warte-Block in Abhängigkeit zum Farbsensor einzusetzen.
- dass der Ultraschallsensor mit Schallwellen arbeitet, die von Objekten zurückgeworfen werden, und dass er programmiert werden kann, auf bestimmte Entfernungen zu reagieren.
- ihren Rad-Roboter so zu programmieren, dass er rückwärts fährt, ein Geräusch abhängig von der Entfernung zu einem Objekt ausgibt und in einer bestimmten Entfernung zu dem Objekt anhält.
- ihr Verständnis des Schleife-Blocks zu erweitern.
- das grundlegende Konzept eines Schalters zu verstehen und wie dieser für 'Wahr'- und 'Falsch'-Operationen verwendet wird.
- den Mathe-Block und seine Funktionen zu verstehen.
- dass Werte über Datenleitungen von einem Block zu einem anderen übertragen werden können.

BEGRIFFE

Eingabe, Ausgabe, Algorithmus, Warten, Ultraschallsensor, debuggen, Schleife, Boole'sche Logik, Schalter, Mathe-Block, Datenleitung, Interrupt.

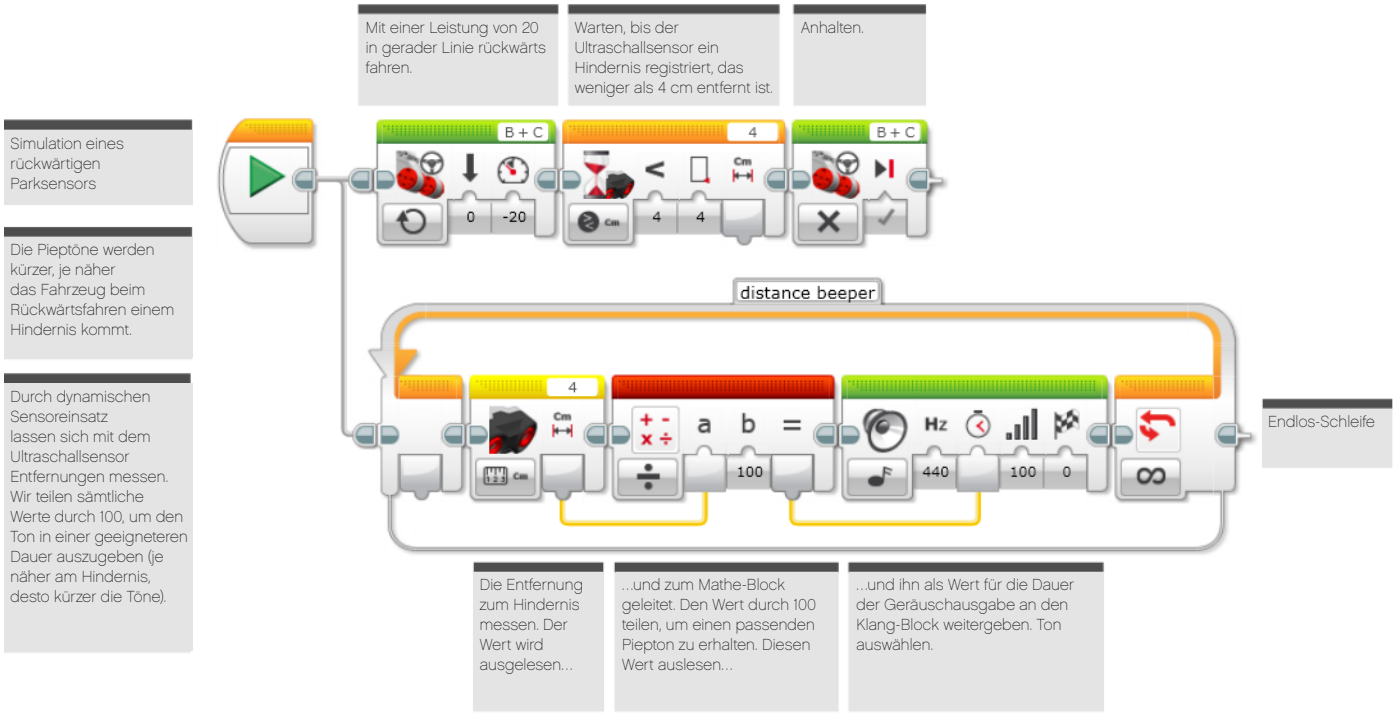
EINFÜHRUNG

- Sagen Sie den Schülern, dass sie im Verlauf dieser Einheit Programme entwickeln werden, die relativ genau einen Parksensor simulieren.
- Weisen Sie darauf hin, dass Einpark-Hilfen Pieptöne ausgeben - zunächst lange, dann immer kürzere, je näher das Auto einem Objekt (normalerweise einer Wand) kommt. Dies wird durch den Einsatz von Ultraschallsensoren bewerkstelligt.
- Sagen Sie den Schülern, dass sie denselben Ultraschallsensor verwenden werden wie in der ersten Unterrichtseinheit. Diesmal aber machen sie umfassender von ihm Gebrauch und erforschen seine Funktionsweise genauer.
- Wie funktioniert der Sensor? Bitten Sie die Schüler um Erklärungen. Sie sollten darauf kommen, dass Ultraschallwellen über das 'Auge' des Sensors ausgesendet und von Objekten zurückgeworfen werden. Die Zeit, die die Welle für den Rückweg braucht, lässt einen Schluss auf die Entfernung des Objekts zu.
- Sie sollten den Schülern ein bisschen Zeit geben, mit dem Sensor zu experimentieren und herauszufinden, wie er genau funktioniert. Beeinflusst die Geschwindigkeit die Genauigkeit des Sensors?

HAUPTAUFGABE 1

- Bei dieser Aufgabe sollen die Schüler ein Programm entwickeln, das ihren Rad-Roboter einen Piepton ausgeben lässt, wenn er sich beim Rückwärtsfahren einem Hindernis nähert.
- Je näher der Roboter dem Objekt kommt, desto kürzer sollen die Töne werden. Das Fahrzeug soll automatisch anhalten, sobald eine bestimmte Entfernung zum Hindernis erreicht ist.
- Der Piepton (und speziell die Dauer der Töne) wird durch den Mathe-Block und Datenleitungen geregelt.
- Eine Erklärung der Funktionsweise sehen Sie im Programm weiter unten.
- Nehmen Sie sich etwas Zeit und führen Sie den Schülern den Mathe-Block und dessen Arbeitsweise vor.
- Die Schüler müssen den Ultraschallsensor an der Rückseite des Basis-Modells anbringen.
- Diese Aufgabe verlangt ein weiteres Mal paralleles Programmieren (Multitasking). Mitunter müssen Sie Ihre Schüler daran erinnern, wie man die dafür nötige Datenleitung vom Start-Block aus verlegt.

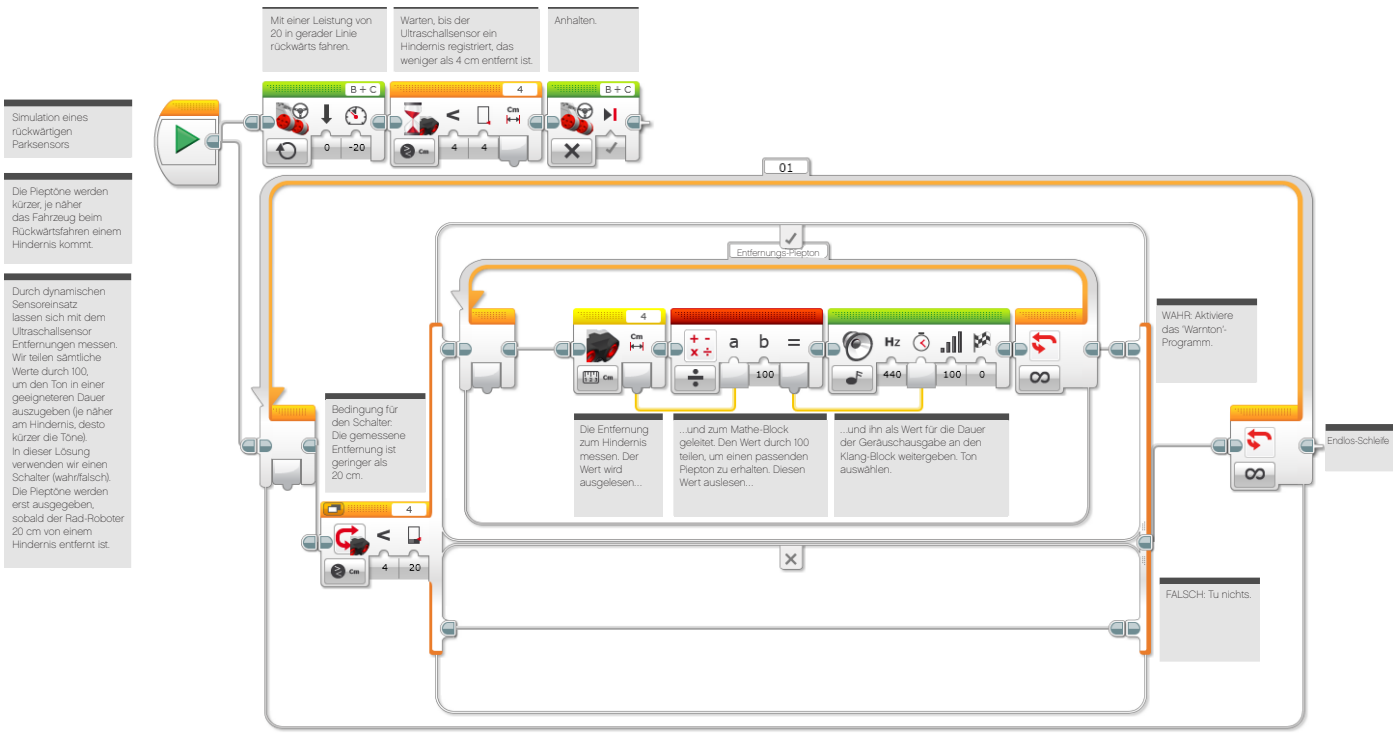
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 6
REITER MAIN 1



HAUPTAUFGABE 2

- Diese Aufgabe baut direkt auf der ersten auf.
- Fragen Sie die Schüler, was mit dem Programm bislang 'falsch läuft'.
- Die Schüler werden anmerken, dass der Rad-Roboter von Beginn an Töne ausgibt, unabhängig von der Entfernung des Hindernisses.
- Wie lässt sich dieser Zustand verbessern und was kann man tun, damit sich der Rad-Roboter mehr wie ein Auto verhält?
- Das Geräusch/der Piepton könnte erst ab einer bestimmten Entfernung von einem Hindernis ausgegeben werden.
- Um dies zu erreichen, müssen die Schüler einen Schalter-Block mit Wahr/Falsch-Anweisung verwenden.

LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 6
REITER MAIN 2



HAUPTAUFGABE 3

- Der Rad-Roboter der Schüler dürfte sich nun recht ähnlich einem rückwärtsfahrenden Auto verhalten.
- Der Piepton sollte erst bei einer bestimmten Entfernung zu einem Hindernis ertönen und der Rad-Roboter sollte bei einer bestimmten Entfernung anhalten.
- Fragen Sie die Schüler, wie man das Programm weiter verbessern könnte.
- Weisen Sie darauf hin, dass bei einem rückwärtsfahrenden Auto zwei Verhaltensweisen zu beobachten sind, die der Rad-Roboter der Schüler bislang nicht zeigt: Es verlangsamt das Tempo bei der Annäherung an das Hindernis, und wenn es bremst, dann wird die Ausgabe des Warngeräuschs beendet.
- Um diesen beiden zusätzlichen Verhaltensweisen zu simulieren, müssen die Schüler ihre Programme erweitern.
- Hierfür benötigen sie einen weiteren Mathe-Block und müssen mit dem Ultraschallsensor die Geschwindigkeit abbilden, damit der Rad-Roboter langsamer wird, wenn sich die Entfernung zum Hindernis verringert.
- Sie benötigen zudem einen Schleifen-Interrupt-Block, um die Tonausgabe zu beenden, sobald der Rad-Roboter anhält.

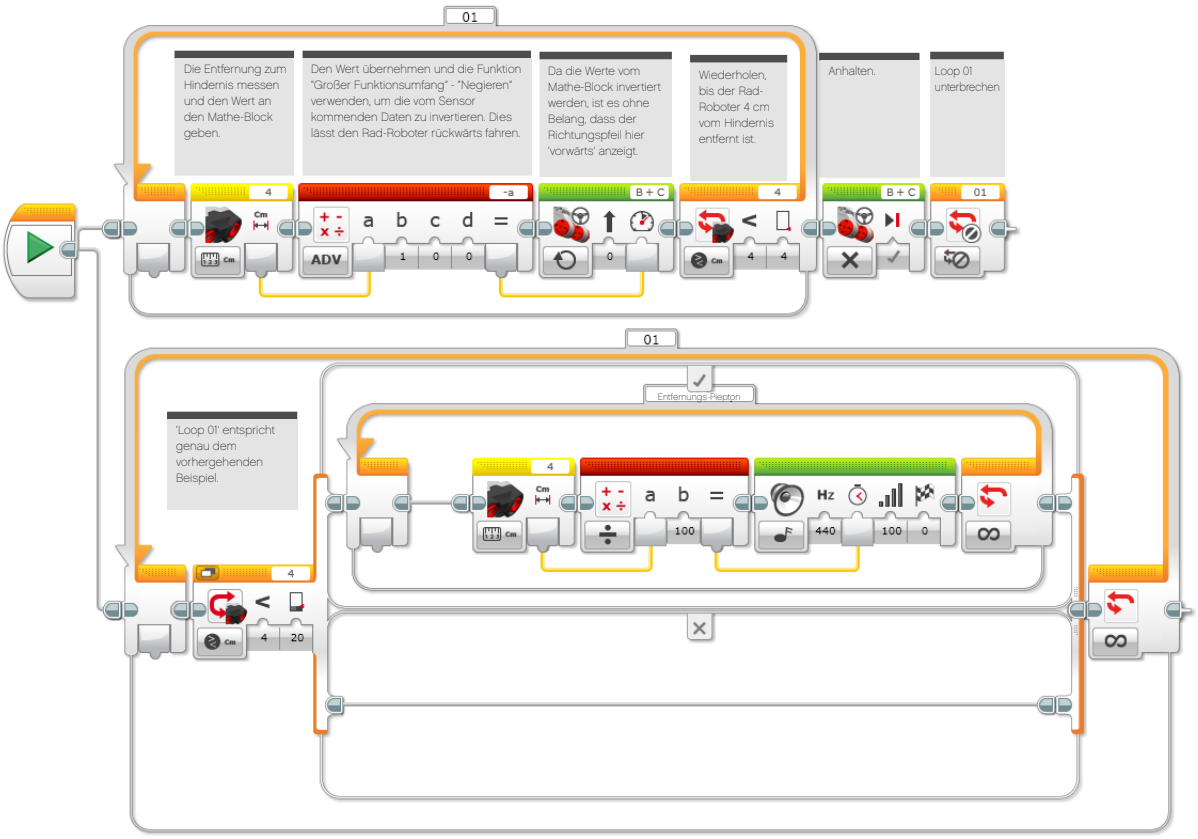
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 6
REITER MAIN 3

Simulation eines rückwärtigen Parksensors

Die Pieptöne werden kürzer, je näher das Fahrzeug beim Rückwärtsfahren einem Hindernis kommt.

Durch dynamischen Sensoreinsatz lassen sich mit dem Ultraschallsensor Entfernungen messen. Wir teilen sämtliche Werte durch 100, um den Ton in einer geeigneteren Dauer auszugeben (je näher am Hindernis, desto kürzer die Töne). In dieser Lösung verwenden wir einen Schalter (wahr/falsch).

Die Pieptöne werden erst ausgegeben, sobald der Rad-Roboter 20 cm von einem Hindernis entfernt ist. Diesmal beginnt der Rad-Roboter, langsamer zu werden, wenn er sich dem Hindernis nähert. Zudem wird die Tonausgabe beendet, wenn der Roboter anhält (bei 4 cm Entfernung zum Hindernis).



ABSCHLIESSENDE DISKUSSION

- Dies ist ein guter Zeitpunkt, die gesamte Klasse ihre Lösungen untereinander austauschen zu lassen.
- Sie könnten alle Rad-Roboter auf einmal rückwärts in Richtung eines Hindernisses fahren lassen und herausfinden, welcher am genauesten ein Auto simuliert.
- Lassen Sie die Schüler ein Video von ihren Rad-Robotern anfertigen, damit sie deren Verhalten zu einem späteren Zeitpunkt auswerten können.

AUFGABEN DIESER UNTERRICHTSEINHEIT

In dieser Unterrichtseinheit findest du heraus, wie sich der gelbe Sensor-Block in Verbindung mit dem Mathe-Block einsetzen lässt. Zudem machst du Gebrauch vom Schleife-Block.

Im Verlauf der drei Aufgaben programmierst du deinen Rad-Roboter so, dass der Parksensor eines Autos simuliert wird.

AUFGABE 1

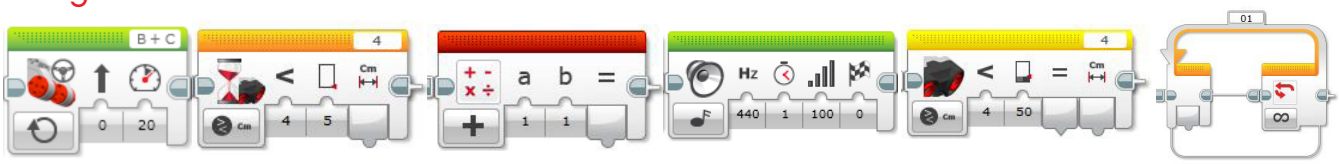
Was geschieht, wenn Autos mit Parksensor rückwärtsfahren? Ein Piepton ist zu hören, der kürzer wird, je näher das Auto dem Hindernis kommt.

Kannst du ein Programm entwickeln, das deinen Rad-Roboter rückwärtsfahren lässt, einen Piepton bei der Annäherung an das Hindernis auslöst und den Roboter automatisch bei einer bestimmten Entfernung stoppt?

Tipp 1: Du musst paralleles Programmieren (Multitasking) einsetzen.

Tipp 2: Du musst dein Wissen über Mathe-Blöcke und Datenleitungen anwenden, um die Frequenz der Pieptöne während der Annäherung des Rad-Roboters an das Hindernis zu erhöhen.

Geeignete Blöcke



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 2

Was ist dir in Bezug auf dein Programm und vor allem in puncto Pieptöne aufgefallen?

Sie sollten kürzer werden, wenn sich dein Rad-Roboter dem Hindernis nähert.

Allerdings ertönen Warngeräusche in der Realität erst, wenn sich das Fahrzeug in einer gewissen Entfernung zum Hindernis befindet.

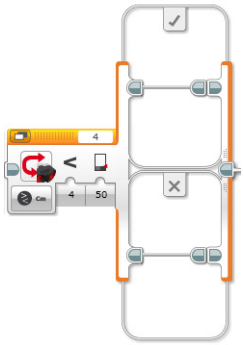
Kannst du diese Eigenschaften in deinem Programm simulieren?

Baue auf deinem bestehenden Programm auf. Du musst es jedoch leicht abändern, um das Piepen erst aber einer bestimmten Entfernung vom Hindernis ertönen zu lassen.

Tipp: Nutze eine Wahr/Falsch-Anweisung sowie Boole'sche Logik. Welchen Programmier-Block benötigst du dafür?

Geeignete Blöcke

Verwende dieselben Blöcke wie in der ersten Programmieraufgabe, aber erwäge auch den Einsatz des folgenden:



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 3

Dein Rad-Roboter sollte mittlerweile recht exakt einen Parksensor simulieren.

Jetzt ist es an der Zeit, deine Programmierfähigkeiten noch ein bisschen deutlicher unter Beweis zu stellen.

Du sollst zwei weitere Funktionen hinzufügen:

1. Kannst du dafür sorgen, dass das Piepen aufhört, wenn der Rad-Roboter in einer bestimmten Entfernung zum Hindernis zum Stehen kommt?
2. Kannst du deinen Rad-Roboter so programmieren, dass er langsamer wird, sobald er zu Piepen beginnt?

Tipp 1: Um das Piepen zu beenden, musst du die Schleife unterbrechen.

Tipp 2: Du musst durch einen zweiten Mathe-Block irgendwo in deinem Programm die Geschwindigkeit zur Entfernung in Relation setzen. Findest du heraus, an welcher Stelle?

Geeignete Blöcke

Verwende dieselben Blöcke wie in der ersten und zweiten Programmieraufgabe, aber erwäge auch den Einsatz des folgenden:

Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

Session 6
Arbeitsblätter

Nach dem Programmieren ist es wichtig, seine Gedanken und Beobachtungen niederzuschreiben. Denke über folgende Punkte nach und notiere dann im Kasten unten, wie diese Programmiersitzung verlaufen ist.

- Wie könntest du dein Programm verbessern?
- Könnte dein Programm geradliniger sein? Hast du zu viele Blöcke verwendet? Lässt sich das Programm effizienter gestalten?
- Wo könnte dein Programm in der 'realen Welt' Anwendung finden?

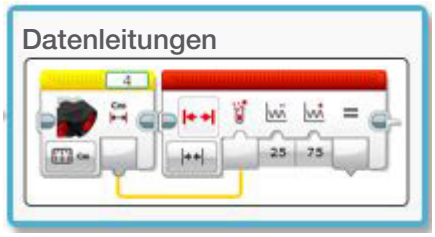
Gedanken und Beobachtungen

ROBOT EDUCATOR-ANLEITUNGEN

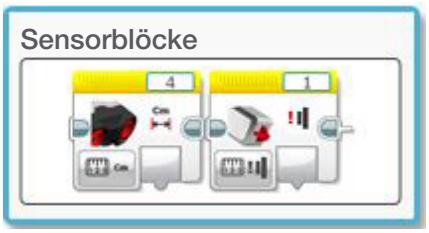
Die folgenden Robot Educator-Anleitungen helfen Lehrern und Schülern bei der Lösung der Aufgaben.

NEUE ROBOT EDUCATOR-ANLEITUNGEN

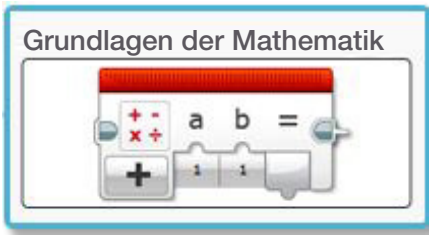
Komplexere Programme > Datenleitungen



Komplexere Programme > Sensor-Blöcke

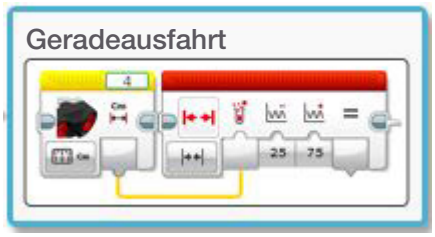


Komplexere Programme > Mathe - Grundlagen

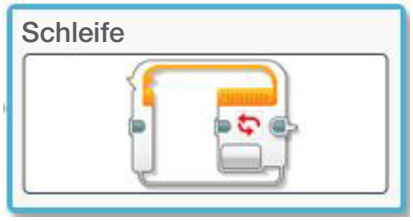


BEREITS BEHANDELTE ROBOT EDUCATOR-ANLEITUNGEN

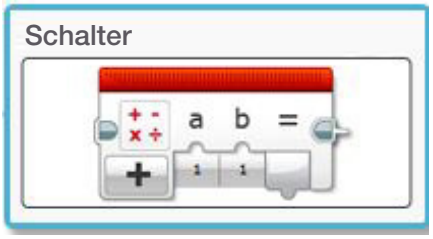
Grundlagen > Geradeausfahrt



Komplexere Programme > Schleife



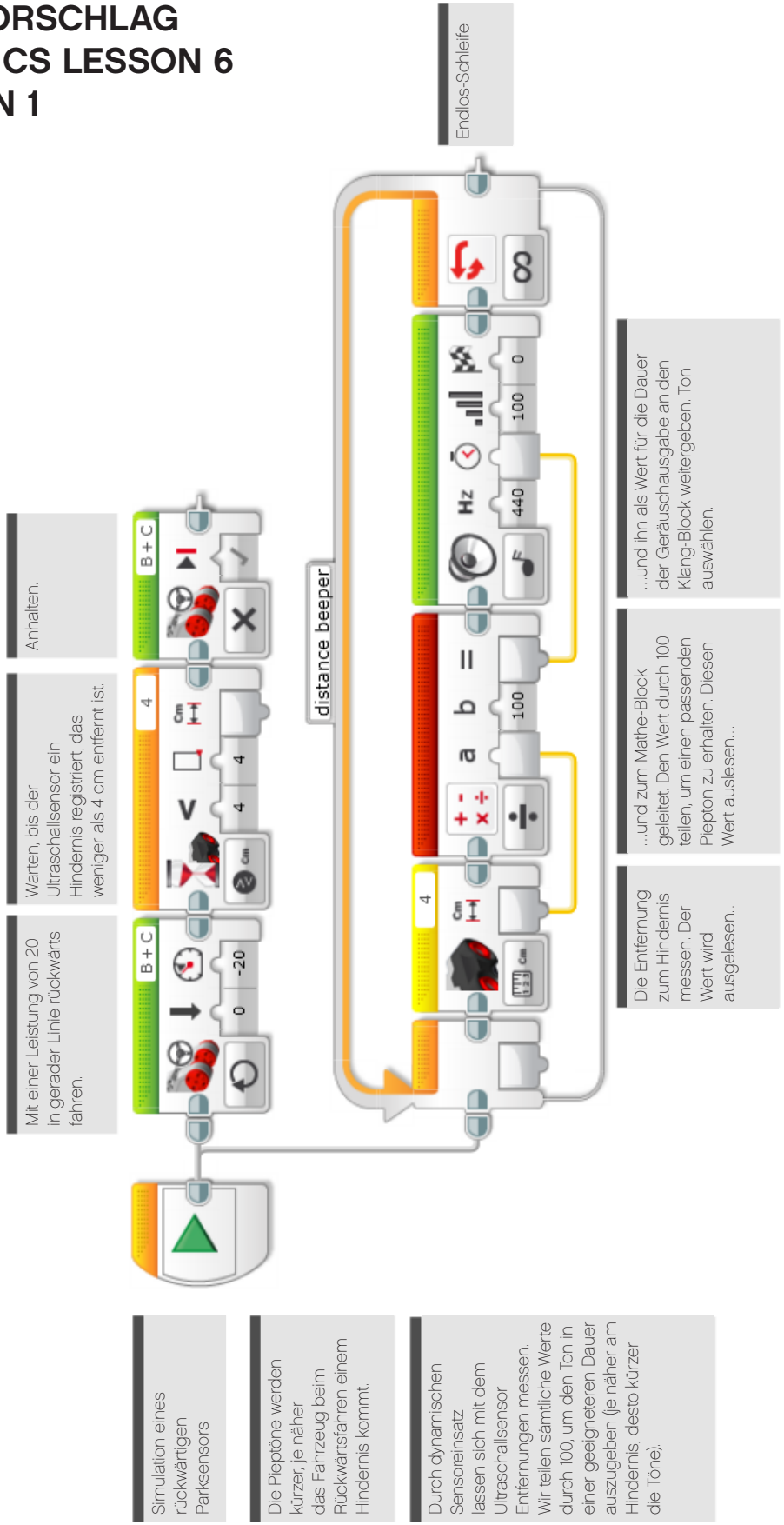
Komplexere Programme > Schalter



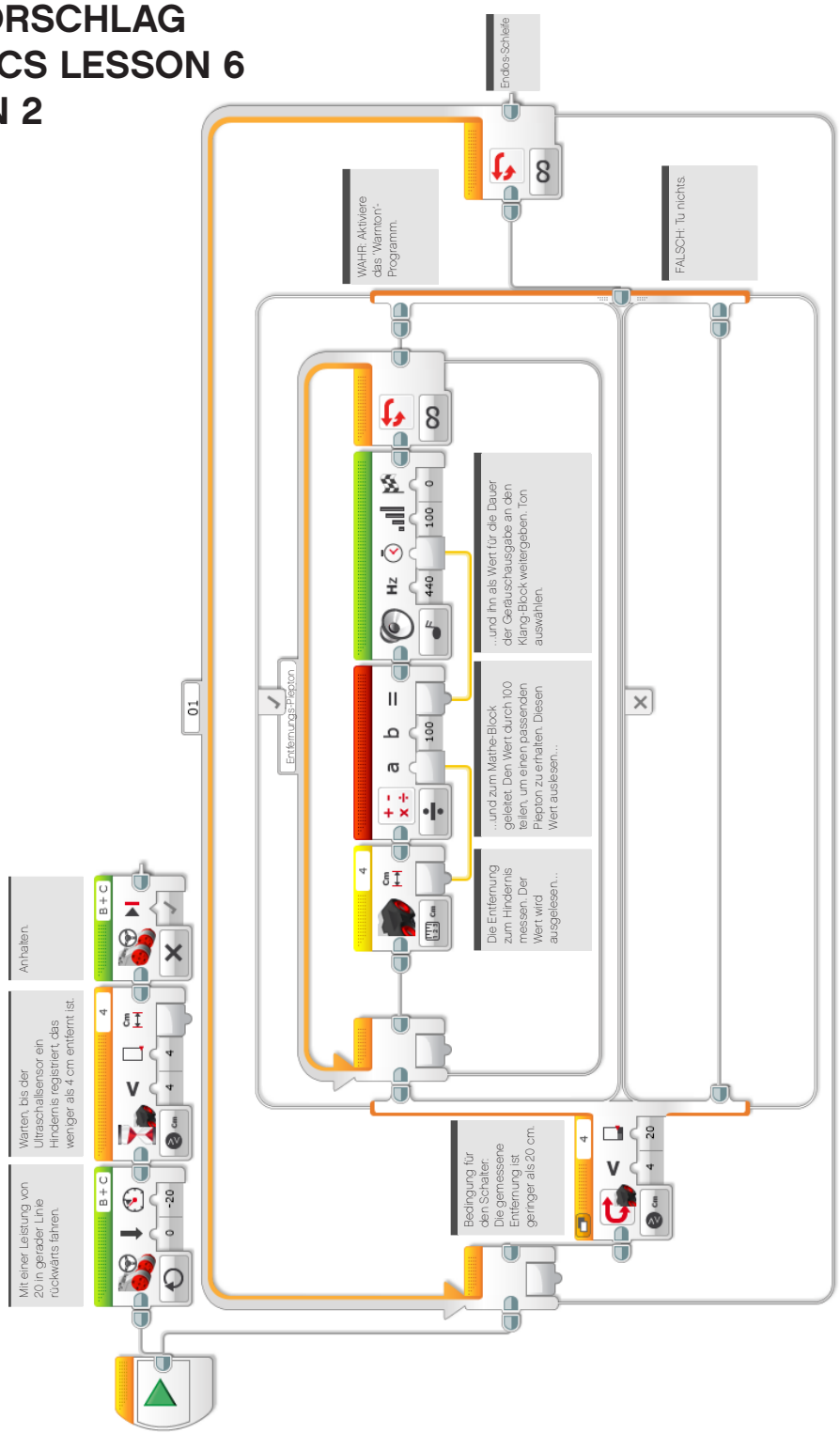
Komplexere Programme > Multitasking

ANHANG FÜR UNTERRICHTSEINHEIT 6: BILDER UND PROGRAMME

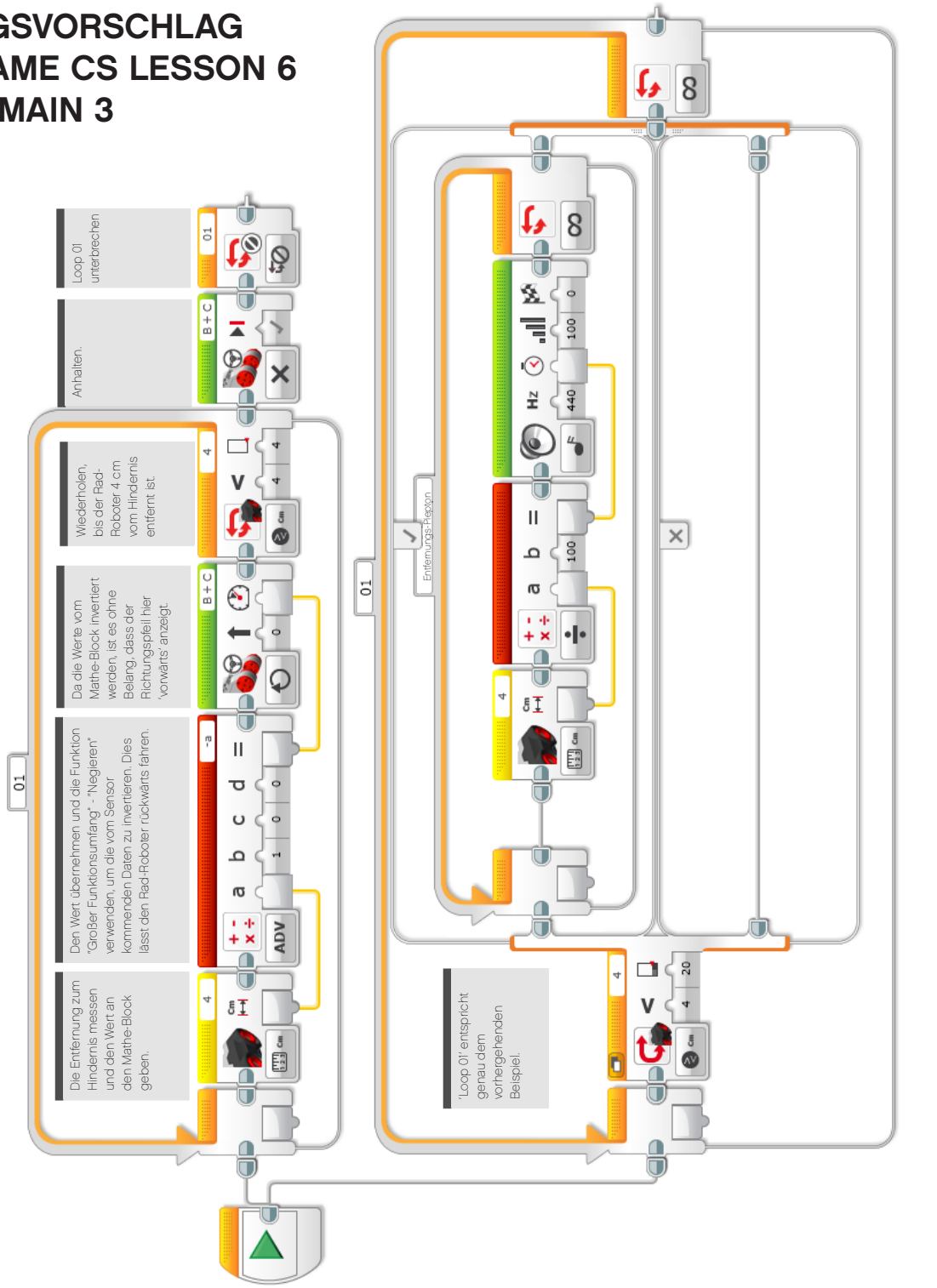
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 6
REITER MAIN 1



LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 6
REITER MAIN 2



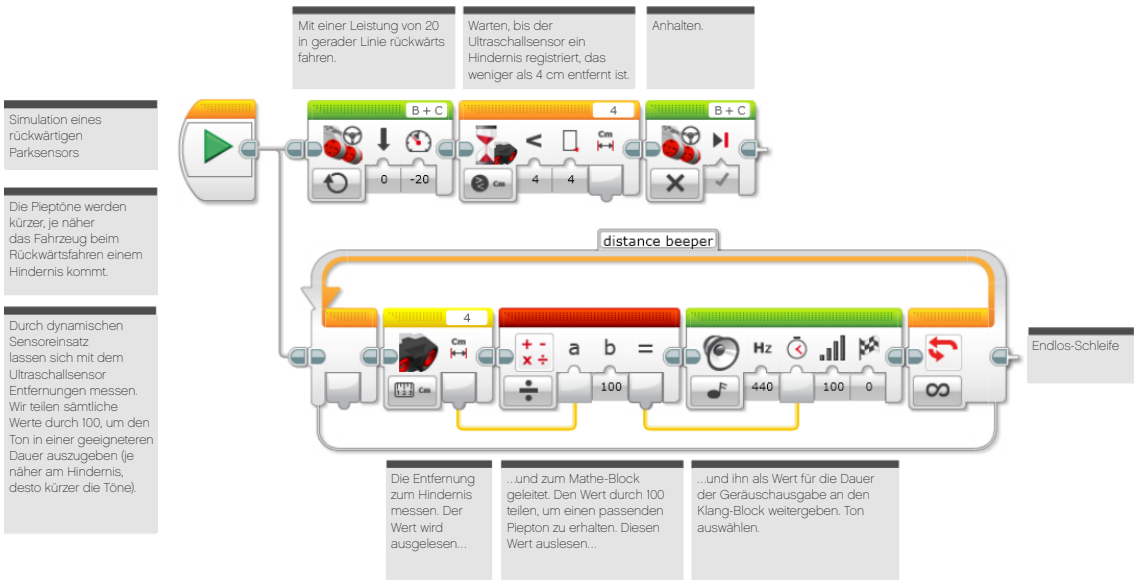
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 6
REITER MAIN 3



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session6_1.c

LEGO Start Page | Lesson 6_1.c*

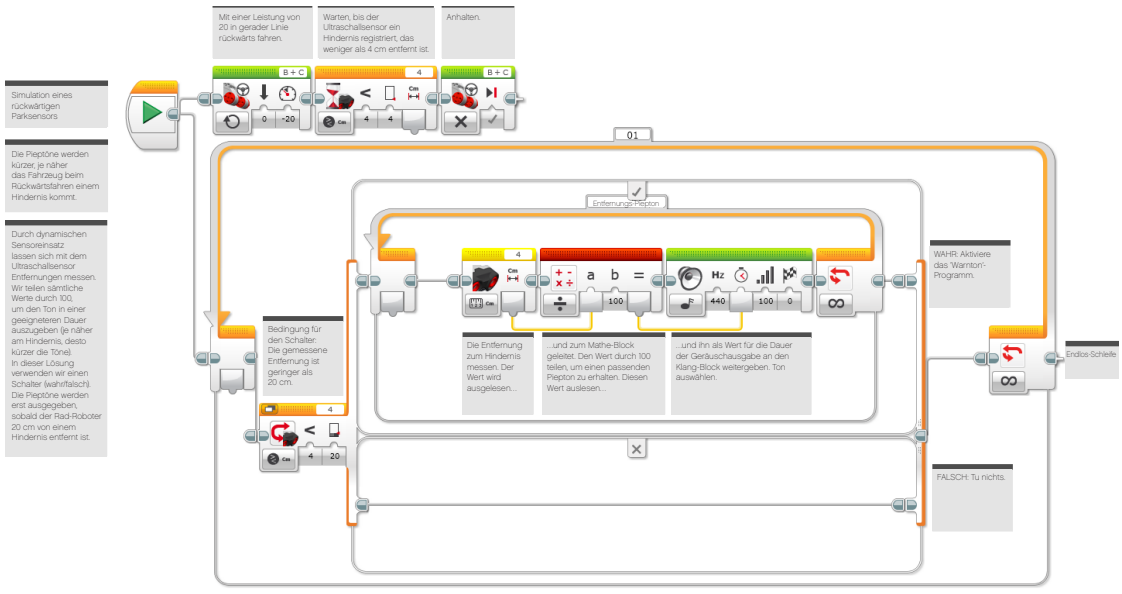
```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//
3
4  task distanceBeeper()
5  {
6    while(true)
7    {
8      //Start playing a tone (in milliseconds)
9      playTone(440, getUSDistance(sonarSensor) * 10);
10
11     //Delay for the length of the tone
12     sleep(SensorValue(sonarSensor) * 10);
13
14     //Add 25ms of "silence" to avoid a constant tone
15     sleep(25);
16   }
17 }
18
19 task main()
20 {
21   //Start the "distance beeper" task.
22   startTask(distanceBeeper);
23
24   while(getUSDistance(sonarSensor) > 4)
25   {
26     setMotorSpeed(motorB, -20);
27     setMotorSpeed(motorC, -20);
28   }
29
30 }
31
```



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session6_2.c

LEGO Start Page | Lesson 6_2.c*

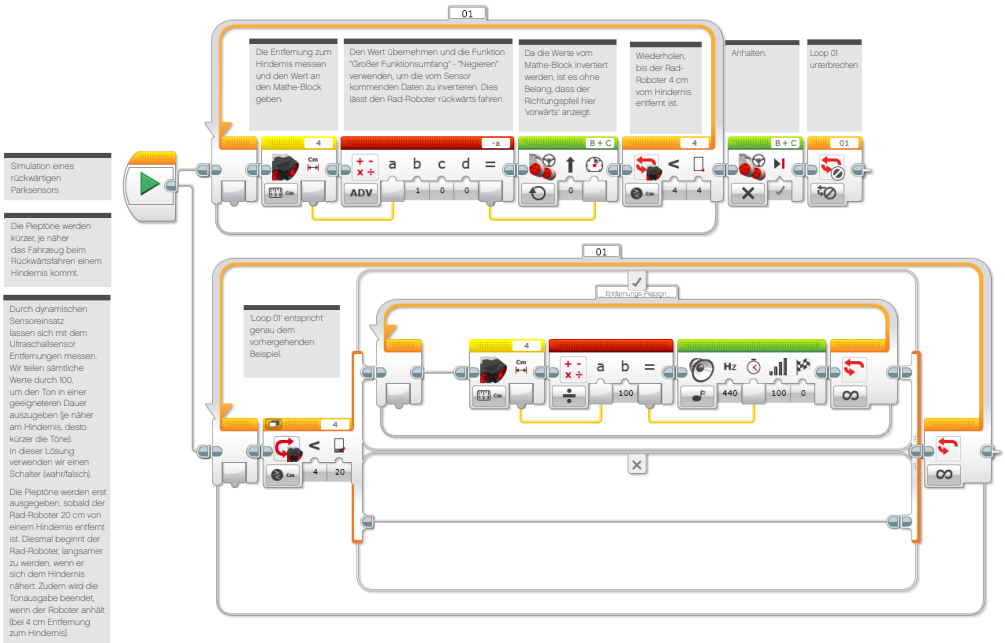
```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//
3
4  task distanceBeeper()
5  {
6    while(true)
7    {
8      //Start playing a tone (in milliseconds)
9      playTone(440, getUSDistance(sonarSensor) * 10);
10
11     //Delay for the length of the tone
12     sleep(getUSDistance(sonarSensor) * 10);
13
14     //Add 25ms of "silence" to avoid a constant tone
15     sleep(25);
16   }
17 }
18
19 task main()
20 {
21   //Start the "distance beeper" task.
22   startTask(distanceBeeper);
23
24   while(getUSDistance(sonarSensor) > 4)
25   {
26     setMotorSpeed(motorB, -20);
27     setMotorSpeed(motorC, -20);
28   }
29
30   stopTask(distanceBeeper);
31 }
32
```



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session6_3.c

LEGO Start Page | Lesson 6_3.c*

```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//
3
4  task distanceBeeper()
5  {
6    while(true)
7    {
8      //Start playing a tone (in milliseconds)
9      playTone(440, getUSDistance(sonarSensor) * 10);
10
11     //Delay for the length of the tone
12     sleep(getUSDistance(sonarSensor) * 10);
13
14     //Add 25ms of "silence" to avoid a constant tone
15     sleep(25);
16   }
17 }
18
19 task main()
20 {
21   //Start the "distance beeper" task.
22   startTask(distanceBeeper);
23
24   while(SensorValue(sonarSensor) > 4)
25   {
26     setMotorSpeed(motorB, -getUSDistance(sonarSensor));
27     setMotorSpeed(motorC, -getUSDistance(sonarSensor));
28   }
29
30   stopTask(distanceBeeper);
31 }
32
```



LEGO mindstorms education EV3

UNTERRICHTSEINHEIT 7

Schlüsselloses Startsystem

LEGOeducation.de

LEGO education

Schlüsselloses Startsystem

Manche modernen Autos lassen sich ohne Einstecken und Drehen eines Zündschlüssel anlassen. Der Schlüssel bleibt in Hosen- oder Handtasche und erlaubt dank Ausstattung mit einem Funkchip das ferngesteuerte Anlassen des Motors. Der Fahrer muss nur einen Knopf sowie die Kupplung drücken, und schon springt das Auto an. Die Schüler lernen in dieser Unterrichtseinheit, wie sie durch Verwendung einer Kombination von Sensoren sowie algebraische Logik dasselbe mit ihrem Rad-Roboter machen können. Befindet sich zum Beispiel ein Objekt in Reichweite des Ultraschallsensors und der Berührungssensor wurde gedrückt? Lautet die Antwort auf beide Fragen 'Ja', dann löst das Programm eine Reaktion aus. Die Schüler werden eben diese Logik verwenden, um ihren Rad-Roboter zu kontrollieren.

Unterrichtseinheit 7
Stundenplanung

ERGEBNISSE

In dieser Einheit lernen die Schüler,

- diverse Schlüssel-Rechenverfahren zu verstehen, die algorithmische Denkweise widerspiegeln.
- einfache Boole'sche Logik (logische Operatoren wie UND, ODER und NICHT) zu verstehen sowie einige ihrer Anwendungsgebiete in Schaltkreisen und beim Programmieren zu nennen.
- den Logik- in Verbindung mit dem Schalter-Block zu verwenden.
- Gebrauch von mehreren Sensoren in Kombination zu machen, um ein Programm auf dem EV3-Stein zu aktivieren.

BEGRIFFE

Eingabe, Ausgabe, Boole'sche Logik, algebraisch, Schalter, Wahr, Falsch.

EINFÜHRUNG

- Sagen Sie den Schülern, dass sie während dieser Unterrichtseinheit ihren Rad-Roboter so modifizieren werden, dass er wie ein modernes, schlüsselloses Auto auf einen Knopfdruck hin bereit zum Losfahren ist.
- Bei den bisherigen Aufgaben wurde immer nur ein Sensor zu einer bestimmten Zeit verwendet. Aber was wäre, wenn man zwei Sensoren gleichzeitig nutzen würde? Gibt es hierfür in der Realität überhaupt Bedarf? Führen Sie aus, dass bei einem Antidiebstahlsystem verschiedene Sensoren einen Alarm auslösen können. Aber was passiert, wenn in einem Programm zwei Sensoren gleichzeitig aktiviert werden?
- Moderne Autos, die sich schlüssellos starten lassen, bedürfen dreier Dinge zum Anlassen des Motors. Die Schüler sollen erklären, um was es sich dabei handelt: Das Vorhandensein eines Schlüssels, das Treten der Kupplung und der Druck auf einen Startknopf. Wie lässt sich so eine Funktionsweise auf den Rad-Roboter adaptieren? Dazu müssen wir uns mit Logik befassen.
- Die Herausforderung besteht darin, ein Programm zu entwickeln, das die Kontrolle über den Rad-Roboter erst dann erlaubt, wenn die richtige Kombination an Startfaktoren gegeben ist..

EINFÜHRUNG FORSCHUNGSAUFGABE

- Die Schüler arbeiten in Paaren oder kleinen Gruppen, um ein Berührungssensor-Modul zu bauen ("Robot Educator > Bauanleitungen > Berührungssensor - Fahrgestell").
- Die Schüler arbeiten ebenfalls in Paaren oder kleinen Gruppen, um ein Ultraschall-Modul zu bauen ("Robot Educator > Bauanleitungen > Ultraschallsensor - Fahrgestell").

HAUPTAUFGABE 1

- Die Schüler müssen etwas Erfahrung im Umgang mit mehr als einem Sensor machen. Sorgen Sie dafür, dass sie in dieser Einheit die dafür nötige Zeit bekommen.
- Stellen Sie zudem sicher, dass am Basis-Modell sowohl Ultraschallsensor, als auch Berührungssensor angeschlossen sind. Die Anleitungen hierfür finden Sie im Handbuch oder den Robot Educator-Bauanleitungen. Erläutern Sie den Schülern, dass sie ein Programm entwickeln sollen, das Nachrichten auf dem Display des EV3-Steins ausgibt, wenn einer der Sensoren ausgelöst wird.
- Sobald die Schüler den Rad-Roboter zum Anzeigen von Nachrichten gebracht haben, müssen sie ihr Programm in eine Schleife einfügen, damit diese Nachrichten wiederholt werden.

LÖSUNGSVORSCHLAG
DATEINNAME CS LESSON 7
REITER MAIN 1

HAUPTAUFGABE 2

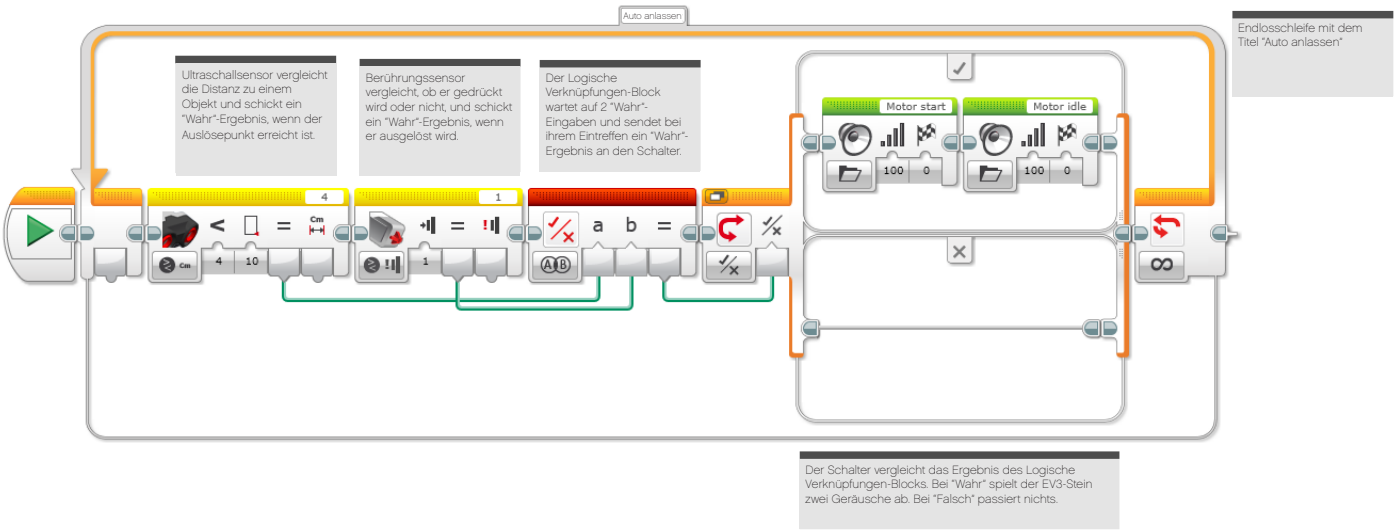
- Holen Sie Feedback der Schüler zum ersten Programm ein. Fragen Sie, wie sich die beiden Sensoren in eine Programmzeile integrieren ließen. Erklären Sie, dass hierfür ein Programmier-Block namens Logische Verknüpfungen benötigt wird.
- Erklären Sie den Schülern, dass logische Verknüpfungen bezüglich ihres Verhaltens einem Venn-Diagramm ähneln. Zeichnen Sie eines mit zwei überlappenden Kreisen an die Tafel oder zeigen Sie es im Internet. Erläutern Sie die Funktionsweise. Weisen Sie auf die zahlreichen Bedingungen hin, die dieses Diagramm hat, und erläutern Sie, wie das Diagramm mit dem Logische Verknüpfungen-Block der EV3-Software in Zusammenhang steht. Erklären Sie die folgende Tabelle und wie die Ergebnisse erzielt werden.

Modi	Verwendete Eingaben	Verwendete Eingaben
UND	A, B	"Wahr", wenn sowohl A als auch B "Wahr" sind, andernfalls "Falsch"
ODER	A, B	"Wahr", wenn entweder A oder B (oder beide) "Wahr" sind, "Falsch", wenn sowohl A als auch B "Falsch" sind
EXKLUSIVES ODER	A, B	"Wahr", wenn ausschließlich A oder B "Wahr" sind, "Falsch", wenn sowohl A als auch B "Wahr" sind, "Falsch", wenn sowohl A als auch B "Falsch" sind
NICHT	A	"Wahr", wenn A "Falsch" ist, "Falsch", wenn A "Wahr" ist

HAUPTAUFGABE 2

- Der Logische Operationen-Block der EV3-Software erzeugt abhängig von den Kriterien, die im Block festgelegt sind, eine Wahr/Falsch-Ausgabe. Die meisten Leute verwenden ihn für A und B, und die Ausgabe von "Wahr". So auch in dieser Aufgabe.
- Im untenstehenden Beispiel wird der Ultraschallsensor verwendet, um den Schlüssel in der Tasche des Fahrers zu simulieren. Der Berührungssensor wird eingesetzt, um das Auto anzulassen. Lassen Sie die Schüler das entsprechende Programm entwickeln.

LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 7
REITER MAIN 2

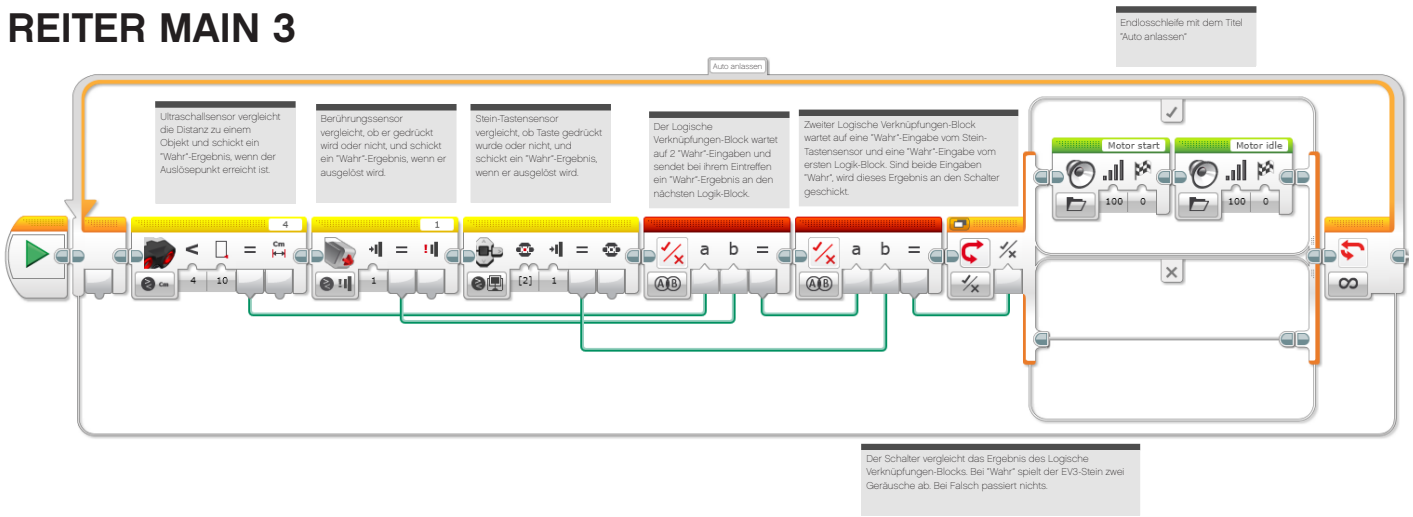


- Fragen Sie die Schüler, ob dieses Geschehen der Realität entspricht.

HAUPTAUFGABE 3

- Stellen Sie sicher, dass die Schüler verstehen, dass drei Kriterien “Wahr” sein müssen, um das Auto zu starten.
- Erstens muss sich der Schlüssel im Auto befinden.
- Zweitens muss die Kupplung getreten werden.
- Drittens muss ein Startknopf gedrückt werden, um den Motor anzulassen.
- Fordern Sie die Schüler auf, ein Programm zu entwickeln, das diesen Vorgang simuliert. Welche Blöcke werden benötigt?
- Anmerkung: Einige Autos benötigen zum Start nur zwei der genannten Kriterien. Sie könnten einige Schüler oder Gruppen anweisen, ein Programm zu entwickeln, das nur zwei Bedingungen hat.

LÖSUNGSVORSCHLAG DATEINAME CS LESSON 7 REITER MAIN 3



ABSCHLIESSENDE DISKUSSION

- Wie kamen die Schüler mit den Datenleitungen zurecht? Lassen Sie sie ihre Gedanken hierzu wiedergeben und stellen Sie klar, dass Datenleitungen beliebig bewegt und geordnet werden können. Dies ist wichtig, um Programme möglichst klar und verständlich zu gestalten.
- Mögliche Aufgaben-Erweiterung: Können die Schüler ein Programm entwickeln, das eine Fehlermeldung anzeigt, wenn die Kriterien für das Anlassen des Autos nicht erfüllt sind?
- Sprechen Sie den bereits bekannten Warten-Block an und machen Sie klar, dass dieser üblicherweise für einen Sensor auf einmal verwendet. Der Logische Verknüpfungen-Block erlaubt dem Benutzer, mehrere Sensoren gleichzeitig zu verwenden.
- Lassen Sie die Schüler die Textversionen der Herausforderungen mit den EV3-Software-Varianten vergleichen. Sie sollen den Ablauf jeder Lösung unter die Lupe nehmen, um zu verstehen, wie beide Versionen funktionieren. Auf ihrem Arbeitsblatt oder in ihrem Heft sollen die Schüler ihre Gedanken zum Einsatz der beiden verschiedenen Sprachen darlegen.

AUFGABEN DIESER UNTERRICHTSEINHEIT

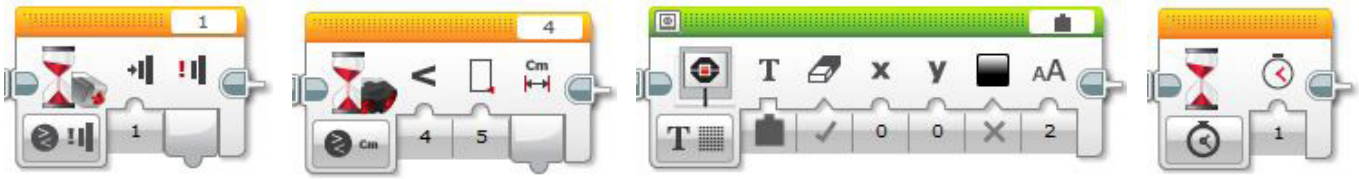
Du sollst ein schlüsselloses Startsystem für deinen Rad-Roboter entwickeln. Sobald eine Kombination verschiedener Sensoren aktiviert ist, wird das Fahrprogramm ausgeführt. Um die folgenden Aufgaben erfolgreich abzuschließen, benötigst du mehrere unterschiedliche Sensoren, die Aufgaben 2 und 3 setzen zudem den Einsatz eines oder mehrerer Logische Verknüpfungen-Blöcke voraus. Auf diesem Arbeitsblatt findest du keine Lösungen, dafür werden Blöcke vorgestellt, mit denen du das Problem angehen kannst..

AUFGABE 1

Programmiere deinen Rad-Roboter so, dass er ein Motorgeräusch wiedergibt, wenn entweder der Berührungssensor gedrückt wird oder der Ultraschallsensor innerhalb einer Entfernung von 5 cm ein Objekt wahrnimmt.

Tipp: Stelle den Ultraschallsensor auf "Vergleichen - **Distanz in Zentimetern" (<).**

Geeignete Blöcke



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 2

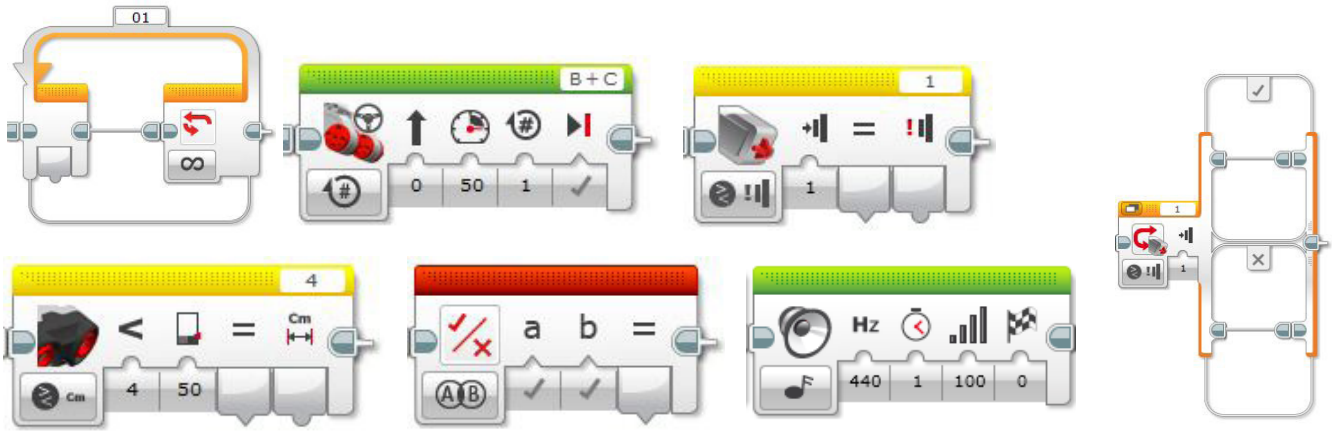
Die zweite Aufgabe dreht sich um den Einsatz des Logische Verknüpfungen-Blocks und zweier Sensoren, die zusammen Informationen an einen weiteren Block geben.

Wie funktioniert ein schlüsselloses Auto? In dieser Aufgabe ist der Berührungssensor die 'Zündung' und der Ultraschallsensor wird verwendet, um den Schlüssel im Auto wahrzunehmen. Beide Sensoren müssen korrekt aktiviert werden, um deinen Rad-Roboter zu starten.

Nachdem du gelernt hast, wie man mehr als einen Sensor in einem Programm verwendet, musst du nun die gelben Sensor-Blöcke nutzen, um eine Logik für den Logische Verknüpfungen-Block zu liefern. Jeder Sensor-Block wird dazu verwendet, ein "Wahr"-Ergebnis zu erzielen. Dieses Ergebnis wird zum Logische Verknüpfungen-Block geleitet. Sind die richtigen Voraussetzungen gegeben, dann wird ein Ergebnis des Logische Verknüpfungen-Blocks dem Schalter-Block übergeben. In diesem Programm liefern zwei Sensor-Blocks Eingaben für einen Logische Verknüpfungen-Block. .

Geeignete Blöcke

Verwende dieselben Blöcke wie in der ersten Programmieraufgabe, aber erwäge auch den Einsatz der folgenden:



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 3

Nun musst du deinen Rad-Roboter so programmieren, dass er reagiert, wenn über drei verschiedene Sensoren mitgeteilt wird, dass die entsprechenden Voraussetzungen erfüllt sind.

Die Sensoren sind

- Berührungssensor = Zündung
- Ultraschallsensor = Schlüssel im Auto
- Stein-Tasten = Kupplung

Der Logische Verknüpfungen-Block kann zwei Eingaben erhalten. Was aber sollen wir tun, wenn wir drei Eingaben verwenden wollen? Überlege den Einsatz von zwei Logik-Blocks.

Zwei Sensoren müssen mit dem ersten Logische Verknüpfungen-Block verbunden sein. Das Ergebnis wird anschließend zum nächsten Logische Verknüpfungen-Block geleitet, mit dem der dritte Sensor verknüpft ist.

Das Ergebnis wird anschließend zum Schalter-Block geleitet.

Geeignete Blöcke

Verwende dieselben Blöcke wie in der ersten und zweiten Programmieraufgabe, aber erwäge auch den Einsatz des folgenden:

Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

Nach dem Programmieren ist es wichtig, seine Gedanken und Beobachtungen niederzuschreiben.

Denke über folgende Punkte nach und notiere dann im Kasten unten, wie diese Programmiersitzung verlaufen ist.

- Wie könntest du dein Programm verbessern?
- Könnte dein Programm geradliniger sein? Hast du zu viele Blöcke verwendet? Lässt sich das Programm effizienter gestalten?
- Wo könnte dein Programm in der 'realen Welt' Anwendung finden?

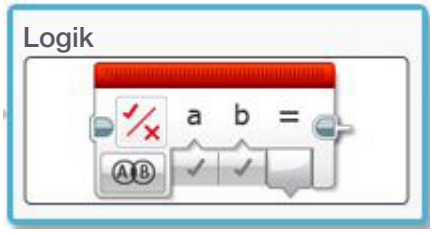
Gedanken und Beobachtungen

ROBOT EDUCATOR-ANLEITUNGEN

Die folgenden Robot Educator-Anleitungen helfen Lehrern und Schülern bei der Lösung der Aufgaben.

NEUE ROBOT EDUCATOR-ANLEITUNGEN

Komplexere Programme > Logische Verknüpfung

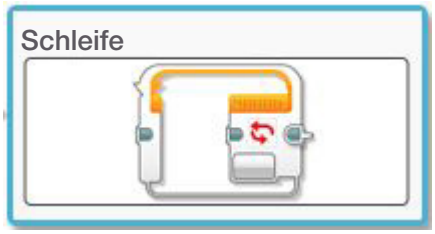


BEREITS BEHANDELTE ROBOT EDUCATOR-ANLEITUNGEN

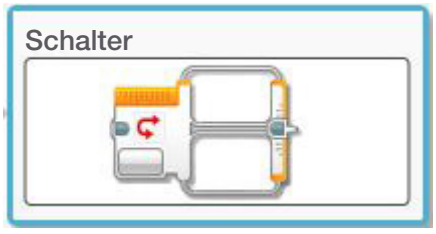
Grundlagen > Geradeausfahrt

Grundlagen > Vor Objekt stoppen

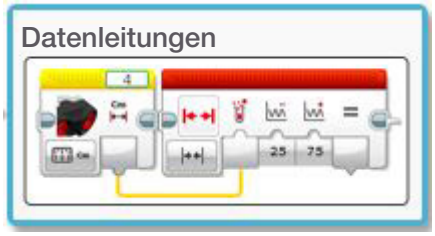
Komplexere Programme > Schleife



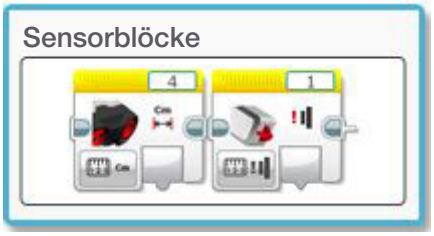
Komplexere Programme > Schalter



Komplexere Programme > Datenleitungen

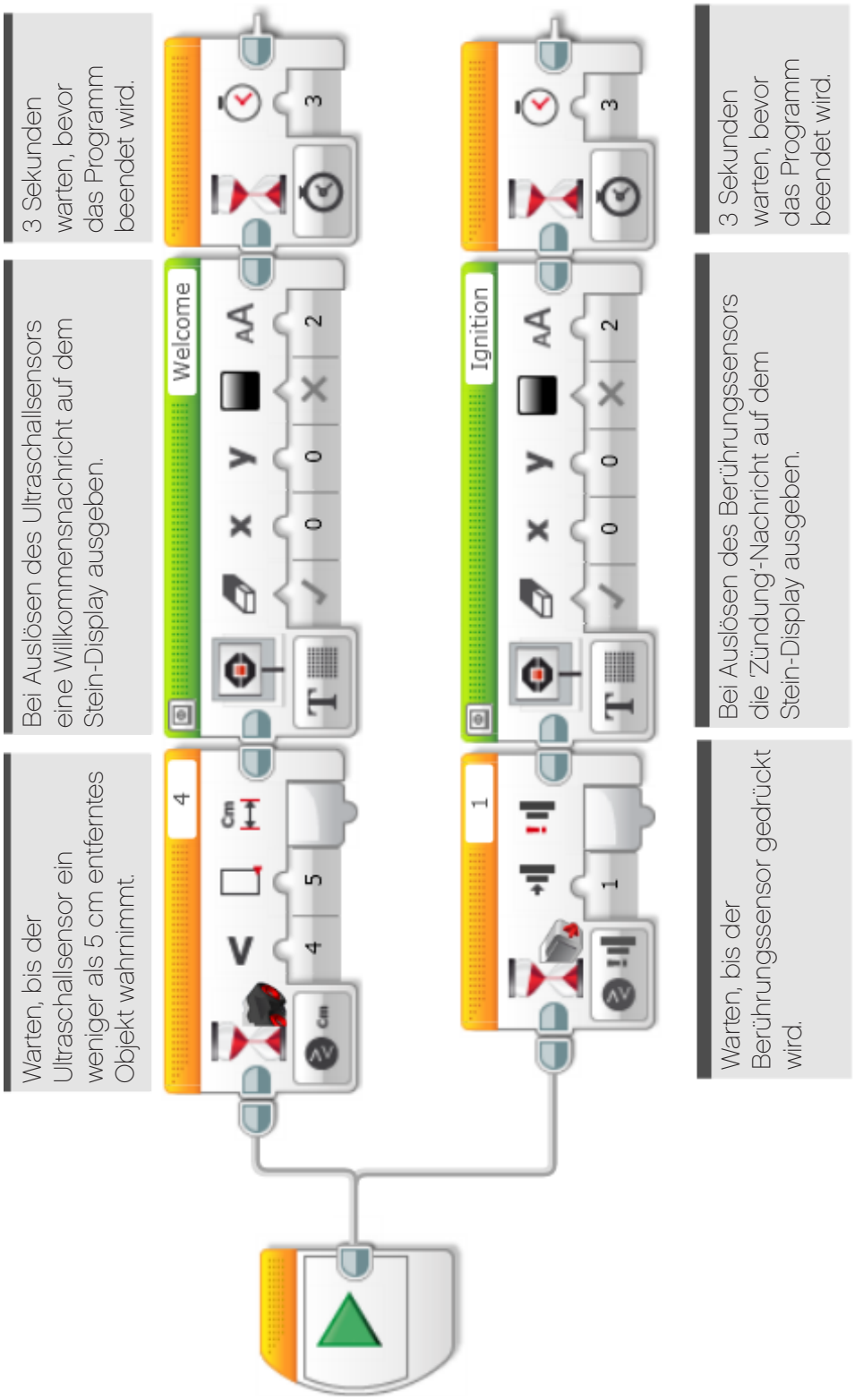


Komplexere Programme > Sensor-Blöcke

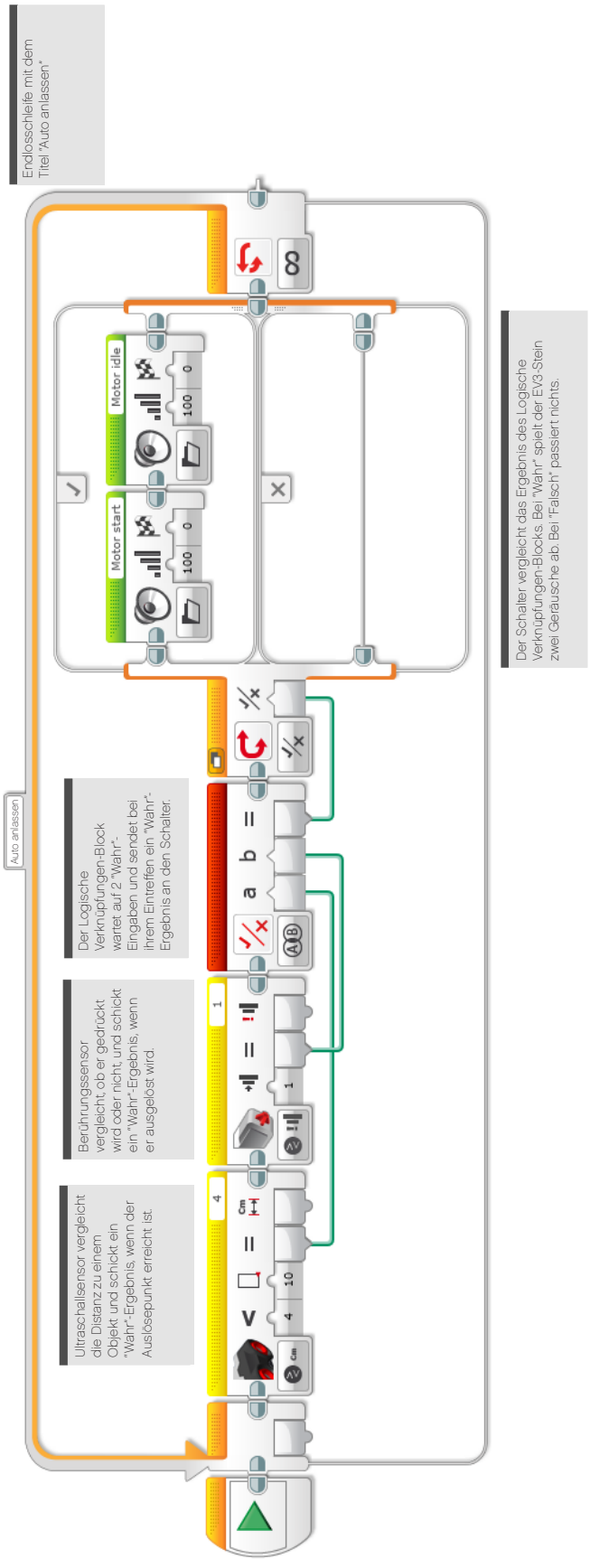


ANHANG FÜR UNTERRICHTSEINHEIT 7: BILDER UND PROGRAMME

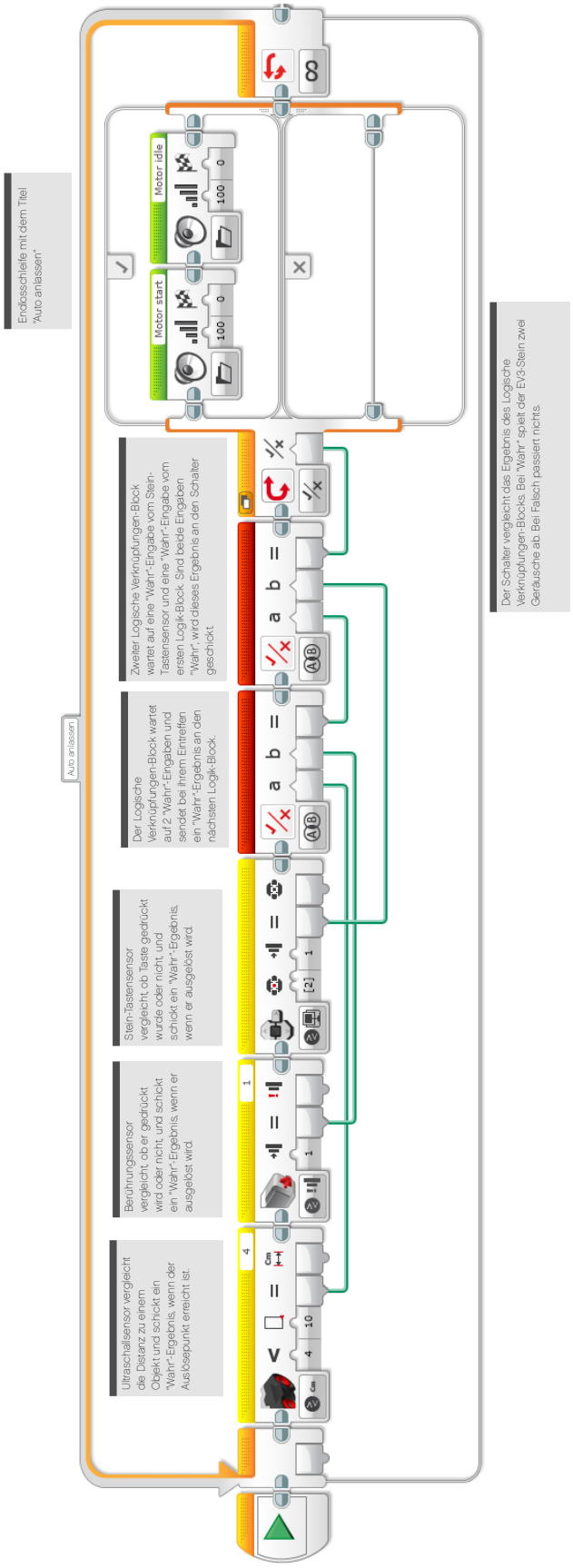
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 7
REITER MAIN 1



LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 7
REITER MAIN 2



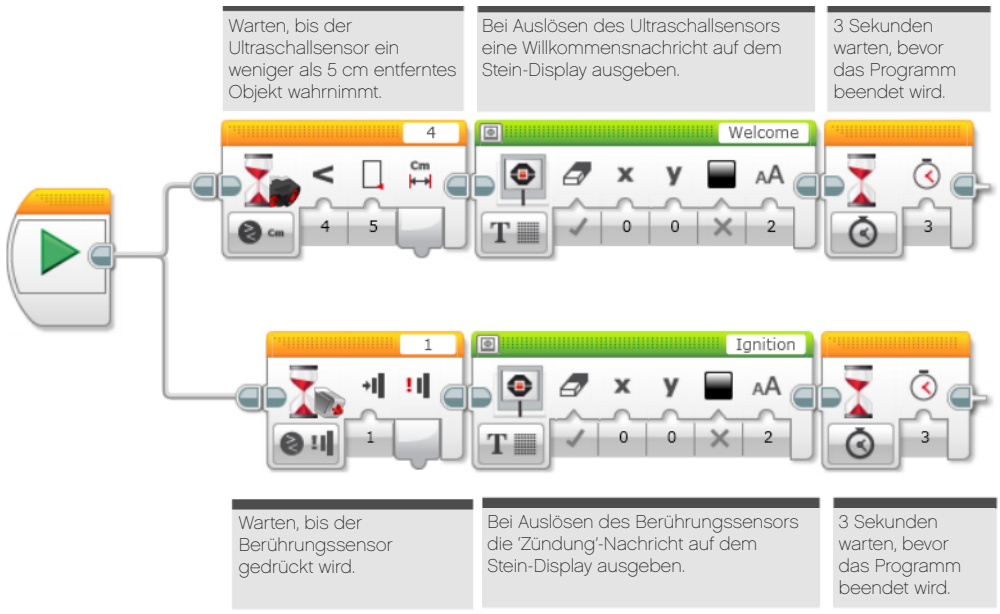
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 7
REITER MAIN 3



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session7_1.c

LEGO Start Page | Lesson 7_1.c*

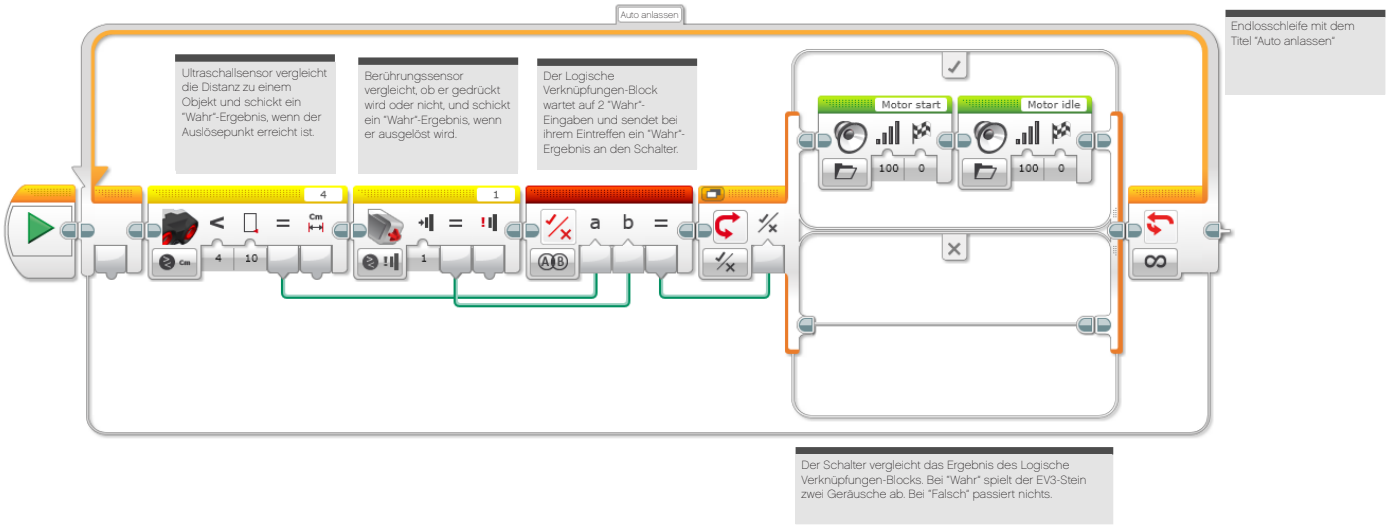
```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//

task touchTask()
{
  //Wait for the touch sensor to be pressed.
  while(getTouchValue(touchSensor) == 0)
  {
    //Do Nothing.
  }

  //Display Text on the LCD Screen
  drawTextAt(0, 0, "Ignition");

  //Display text for 3 seconds
  sleep(3000);
}

task main()
{
  //Start the second task to monitor the touch sensor
  startTask(touchTask);

  //Wait for the sonar sensor to see an object 5cm away
  while(getUSDistance(sonarSensor) > 5)
  {
    //Do Nothing.
  }

  //Display text on the LCD Screen
  drawTextAt(0, 0, "Welcome");

  //Display text for 3 seconds
  sleep(3000);
}
```



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session7_2.c

LEGO Start Page | Lesson 7_2.c*

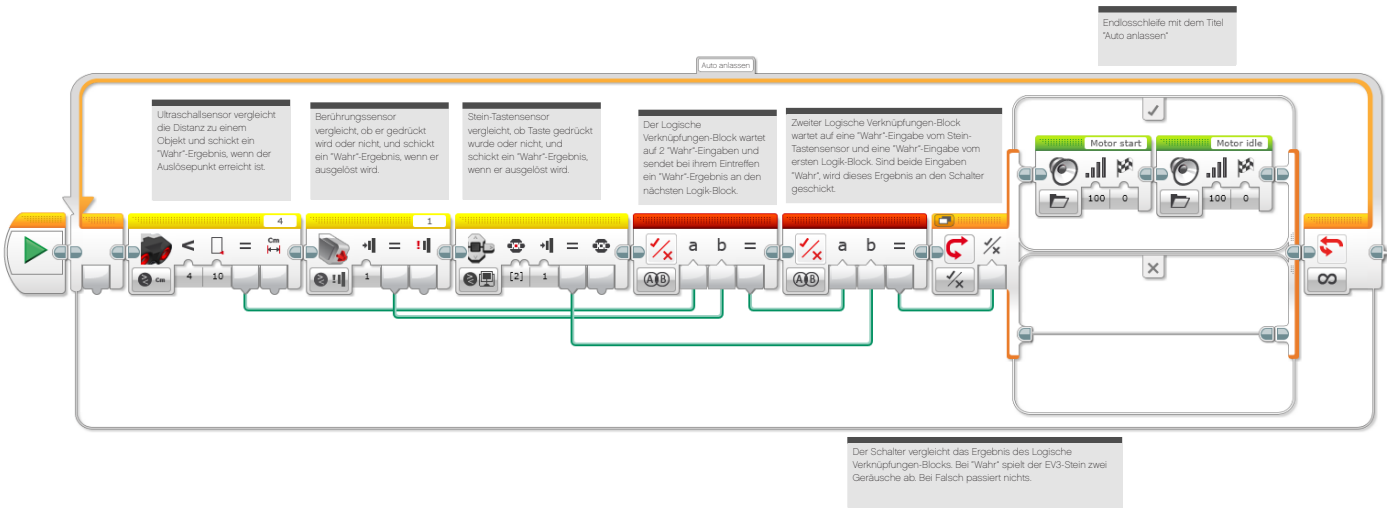
```
1  pragma config(StandardModel, "EV3_REMBOT")
2  /*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//
3
4  ask main()
5
6   //Repeat continuously
7   while(true)
8   {
9     //If Sonar Sensor is less than 10 AND Touch Sensor is pressed
10    if(getUSDistance(sonarSensor) < 10 && getBumpedValue(touchSensor) == 1)
11    {
12      //Play 2 sound files (not quite implemented yet...)
13      playSoundFile("Motor Start.rsf");
14      playSoundFile("Motor Idle.rsf");
15    }
16  }
17
18
```



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session7_3.c

LEGO Start Page | Lesson 7_3.c*

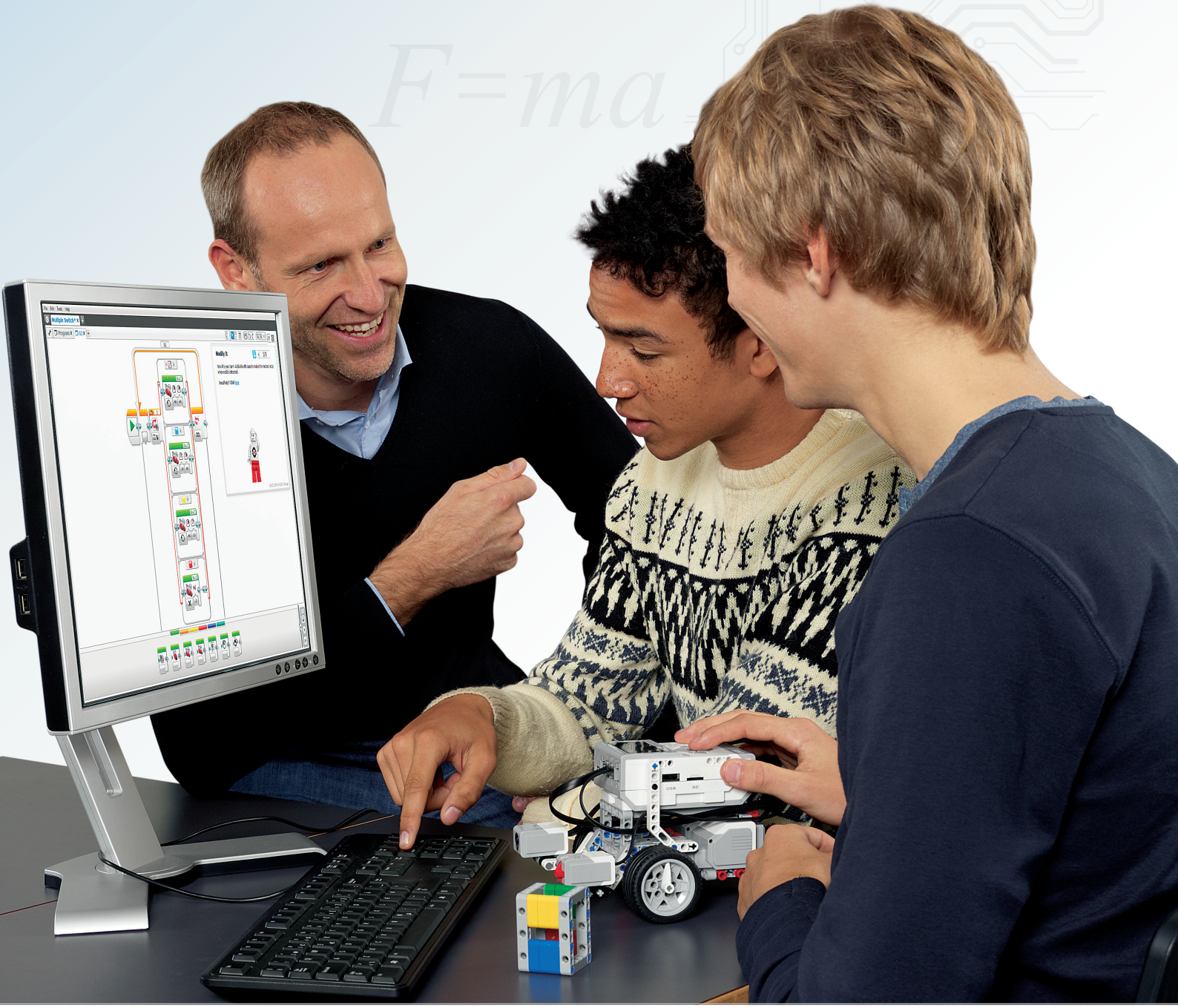
```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//
3
4  task main()
5  {
6    //Repeat continuously
7    while(true)
8    {
9      //If Sonar Sensor is less than 10 AND Touch Sensor is pressed
10     if(getUSDistance(sonarSensor) < 10 &&
11        getTouchValue(touchSensor) == 1  &&
12        getButtonPress(buttonEnter) == 1)
13     {
14       //Play 2 sound files
15       playSoundFile("Motor Start.rsf");
16       playSoundFile("Motor Idle.rsf");
17     }
18   }
19 }
20
```



mindstorms education EV3

UNTERRICHTSEINHEIT 8

Geschwindigkeitsregelanlage



LEGOeducation.de

education

Geschwindigkeitsregelanlage

In dieser Unterrichtseinheit machen Sie die Schüler mit Variablen vertraut. Am Ende der Einheit werden sie eine Geschwindigkeitsregelanlage für den Rad-Roboter entwickeln.

Ein Druck auf den Berührungssensor soll die Geschwindigkeit des Roboters erhöhen.

Geben Sie zu bedenken, dass in Autos mit automatischem Geschwindigkeitsregler gewöhnlich eine einfache Erhöhung oder Erniedrigung des Tempos über Steuerelemente am Lenkrad möglich ist. Dieses Konzept kann durch die Verwendung des Variable-Blocks simuliert werden. Die Geschwindigkeit wird durch den Druck des Berührungssensors eingestellt.

Unterrichtseinheit 8
Stundenplanung

ERGEBNISSE

In dieser Einheit lernen die Schüler,

- diverse Schlüssel-Rechenverfahren zu verstehen, die algorithmische Denkweise widerspiegeln.
- den Variable-Block zu verwenden, um Informationen zu speichern.
- mehrstufige Programme zu entwickeln.
- eigene Blöcke zu entwerfen.

BEGRIFFE

Eingabe, Ausgabe, Variable, Konstante, Schleife, Warten, Motor, Berührungssensor.

EINFÜHRUNG

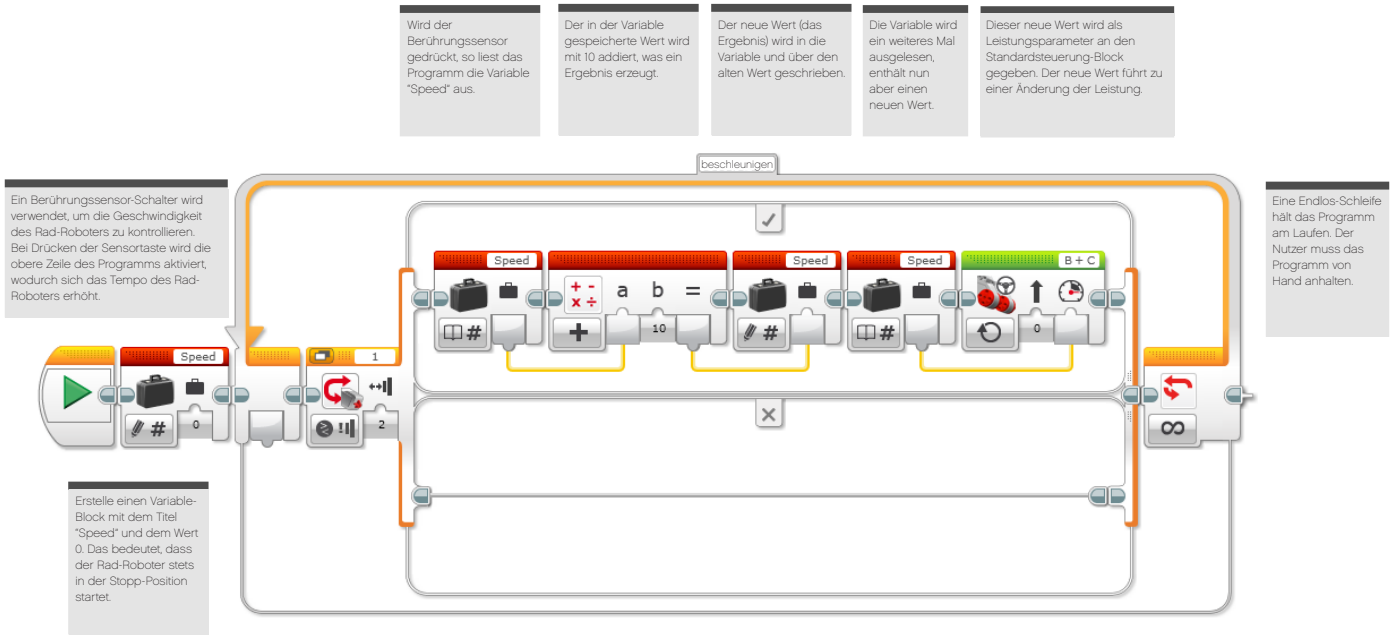
- Zeigen Sie ein Video eines Autos, das eine Geschwindigkeitsregelanlage (je nach Fahrzeugmarke auch als Tempomat oder Tempostat bekannt) verwendet. Lassen Sie die Schüler das Gesehene in Worte fassen. Erklären Sie, dass ein Bordcomputer das Tempo des Autos automatisch kontrolliert, ähnlich dem Autopilot eines Flugzeugs.
- Unterstützen Sie ihre Erläuterungen mit diesen Bildern

- Fragen Sie die Schüler, was die Plus- und Minustasten wohl zu bedeuten haben. Erklären Sie, dass mit diesen die Geschwindigkeit des Autos angepasst werden kann.
- Präsentieren Sie den Schülern die heutige Aufgabe: Durch den Einsatz zweier Berührungssensoren soll das Tempo des Rad-Roboters kontrolliert und aufrechterhalten werden.
- Zeigen Sie das zweite Bild. Was bedeuten die Tasten 'Accel' und 'Decel'? Lassen sie die Schüler herausfinden, dass diese dem Beschleunigen und Verlangsamen des Autos dienen.
- Erklären Sie, dass bei der heutigen Programmierung der Variable-Block verwendet wird.
- Unter Umständen sollten Sie den Schülern den Unterschied zwischen einer Konstante und einer Variable erläutern. Also dass eine Konstante verwendet wird, um ein Programm kontinuierlich mit einem fixen Wert zu versorgen, der vom Nutzer nur verändert werden kann, wenn das Programm nicht läuft. Führen Sie hierzu den Konstante-Block in Aktion mit einem Bewegungs-Block und dem Anzeige-Block vor.
- Eine Variable macht es dagegen möglich, in einem Programm Werte zu speichern, die von diesem Programm verwendet werden können. Der Unterschied zur Konstante ist, dass die Werte im laufenden Programm immer wieder überschrieben werden können.
- Zeigen Sie den Schülern, wie man den Variable-Block verwendet und lassen Sie ein Programm entwickeln, das eine Variable enthält.

HAUPTAUFGABE 1

- Mittlerweile müssten die Schüler mit dem Robot Educator-Fahrgestell-Modell vertraut sein. Haben sie doch noch keines gebaut, dann sollte dies jetzt geschehen. Bedenken Sie, dass die Schüler das Modell mit zwei nach vorne gerichteten Berührungssensoren ausstatten müssen.
- Vertiefen Sie das Verständnis der Schüler für den Variable-Block. Erklären Sie, dass es sich um einen Programmier-Block handelt, mit dem sich Daten (Text, Logik, eine Zahl oder Arrays) speichern lassen, die in einem laufenden Programm jederzeit überschrieben werden können.
- Erläutern Sie, dass Daten aus diesem Block gelesen und in ihn geschrieben werden müssen. Hierzu werden Mathe-, Text- oder Array-Operationen-Blöcken verwendet.
- Die Schüler sollen ihren Rad-Roboter so programmieren, dass er während der Fahrt durch Druck auf den Berührungssensor beschleunigt werden kann. Dazu müssen sie den wartenden Berührungssensor in einer Schleife unterbringen.

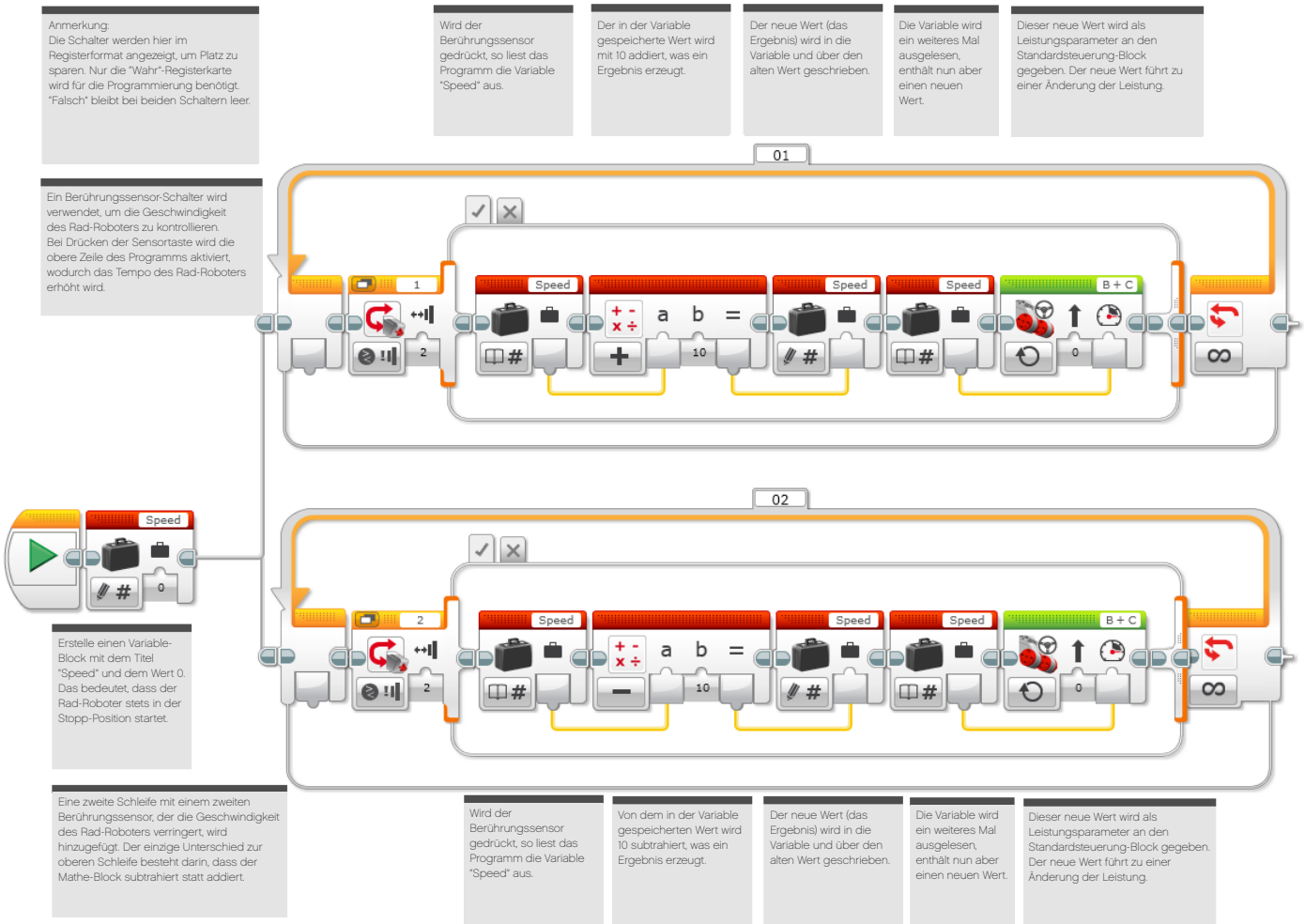
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 8
REITER MAIN 1



HAUPTAUFGABE 2

- Nach der Entwicklung des ersten Programms, welches den Rad-Roboter beschleunigt, diskutieren Sie nun mit den Schülern, wie diese das Programm erweitern würden, um den Roboter auch verlangsamen zu können.
- Stellen Sie folgende mögliche Lösung dar: Eine zweite Endlos-Schleife wird hinzugefügt, bei der einfach der Sensor-Port geändert wird (ein zweiter Sensor wird an das Modell gebaut) und der Mathe-Block nicht addiert, sondern subtrahiert.
- Erinnern Sie Ihre Schüler an das Konzept des Multitasking und sagen Sie ihnen, dass in dieser Aufgabe ihr Wissen über paralleles Programmieren benötigt wird. Weisen Sie darauf hin, dass ein Schleife-Block zur Arbeitsfläche hinzugefügt werden muss, bevor sie die zweite Datenleitung aus dem Start-Block ziehen.

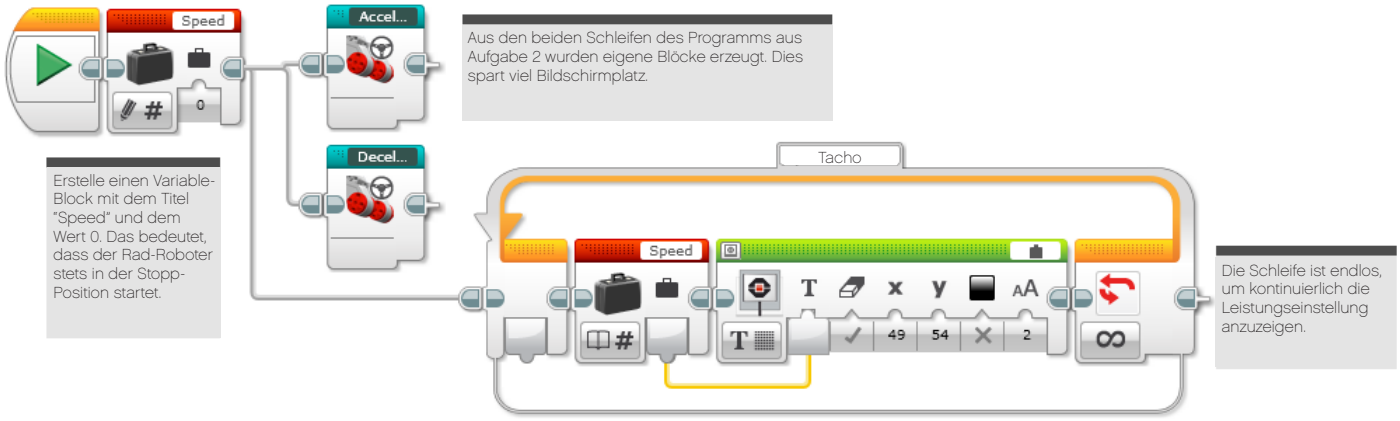
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 8
REITER MAIN 2



HAUPTAUFGABE 3

- Der Rad-Roboter wird nun also auf Knopfdruck langsamer bzw. schneller. Die Schüler sollen ihr Programm noch einmal erweitern, so dass das Display des EV3-Steins die Geschwindigkeit des Rad-Roboters anzeigt - also der Tacho eines Autos simuliert wird.
- Lassen Sie die Schüler über die Verwendung des Variable-Blocks nachdenken. Können sie ihn nutzen, um diese Information anzuzeigen?
- Eine mögliche Lösung finden Sie weiter unten auf dieser Seite.
- Bei dieser Aufgabe wird eine neue Fähigkeit erlernt - das Entwerfen eigener Blöcke. Zwei dieser Blöcke finden sich in der unten stehenden Lösung. Eigene Blöcke erlauben dem Nutzer, aus bereits geschriebenen Programmen Subroutinen zu erzeugen. Im Fall unten wurden die Beschleunigungs- und Verlangsamungsschleifen zur Erzeugung eigener Blöcke verwendet. Es gibt zwei Gründe für diese Aktion: Zum einen sparen die eigenen Blöcke Platz, zum anderen können diese Subroutinen in anderen Programmen wiederverwendet werden.

LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 8
REITER MAIN 3



ABSCHLIESSENDE DISKUSSION

- Fragen Sie die Schüler, wie der Konstante-Block für die Geschwindigkeit verwendet werden könnte. Führen Sie aufbauend auf dem Vergleich der Funktionsweise von Konstanten und Variablen aus, dass feste Werte nützlich sein könnten, um dem Rad-Roboter ein 'Tempo-Limit' zu verpassen.
- Das Aktivieren des ersten Beschleunigungssensors könnte eine Variable nutzen, die auf 30% gesetzt ist, was sich mit 30 km/h vergleichen ließe. Ein zweiter Berührungssensor könnte unter Verwendung eines weiteren Konstanten-Blocks die Geschwindigkeit auf 70% setzen.
- Lassen Sie die Schüler die Textversionen der Aufgaben mit den EV3-Software-Varianten vergleichen. Sie sollen den Ablauf jeder Lösung unter die Lupe nehmen, um zu verstehen, wie beide Versionen funktionieren. Auf ihrem Arbeitsblatt oder in ihrem Heft sollen die Schüler ihre Gedanken zum Einsatz der beiden verschiedenen Sprachen darlegen.

AUFGABEN DIESER UNTERRICHTSEINHEIT

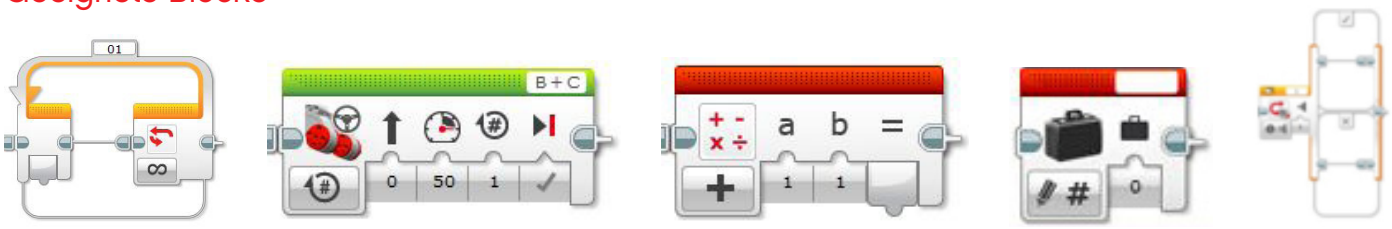
Für deinen Rad-Roboter entwickelst du heute eine Geschwindigkeitsregelanlage, wie man sie in modernen Autos finden kann. Du benötigst die beiden Berührungssensoren des EV3-Baukastens, um die Plus- und Minus-Tasten zu simulieren, die sich bei Autos mit Geschwindigkeitsregelanlage am Steuer befinden.

AUFGABE 1

Programmiere das Auto so, dass es in 10er-Einheiten beschleunigt. Verwende den Variable-Block für die Geschwindigkeit, der die 10er-Werte hinzu addiert werden.

Tipp: Sorge dafür, dass der Standardsteuerung-Block auf 'An' steht.

Geeignete Blöcke



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 2

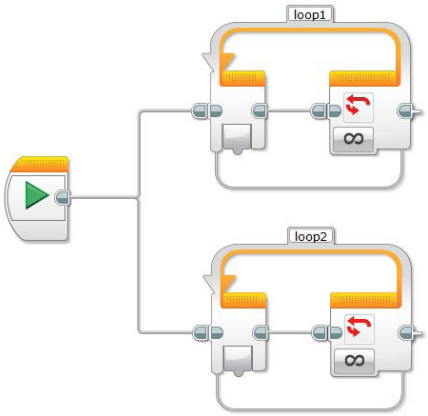
Nachdem du ein Programm geschrieben hast, das deinen Rad-Roboter beschleunigt, wird nun ein Unterabschnitt benötigt, der das Fahrzeug abbremst. Das geschieht ganz einfach durch das Hinzufügen einer zweiten Schleife und eines weiteren Schalter-Blocks.

In der Extra-Schleife soll sich ein zweiter Berührungssensor-Block befinden sowie ein Mathe-Block, der subtrahiert statt addiert.

Denke daran, dass du hier Multitasking mit zwei gleichzeitig laufenden Programmzeilen einsetzen musst.

Geeignete Blöcke

Verwende dieselben Blöcke wie in der ersten Programmieraufgabe, aber erwäge auch den Einsatz des folgenden:



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 3

Nachdem du bereits das Tempo deines Rad-Roboters durch die Verwendung zweier Berührungssensoren kontrollieren kannst, wäre es eine tolle Sache, wenn sich nun auch noch die Geschwindigkeit (die Motorleistung) auslesen und auf dem Display des EV3-Steins anzeigen ließe. Dein Lehrer sollte dir gezeigt haben, wie man aus bereits entwickelten Programmteilen eigene Blöcke macht. Diese sind in zweierlei Hinsicht nützlich: Zum einen wird auf dem Programmier-Bildschirm Platz eingespart. Zum anderen können diese Miniprogramme in anderen Programmen wiederverwendet werden, da sie sich gesondert abspeichern lassen.

Um die Leistung optisch wiederzugeben, liest du den Wert der Variable aus, welche die Motorleistung steuert, und gibst sie unter Verwendung des Anzeige-Blocks im Text-Pixel-Modus auf dem Display des EV3-Steins aus.

Geeignete Blöcke

Verwende dieselben Blöcke wie in der ersten und zweiten Programmieraufgabe, aber erwäge auch den Einsatz der folgenden:

Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

Nach dem Programmieren ist es wichtig, seine Gedanken und Beobachtungen niederzuschreiben. Denke über folgende Punkte nach und notiere dann im Kasten unten, wie diese Programmiersitzung verlaufen ist.

- Wie könntest du dein Programm verbessern?
- Könnte dein Programm geradliniger sein? Hast du zu viele Blöcke verwendet? Lässt sich das Programm effizienter gestalten?
- Wo könnte dein Programm in der 'realen Welt' Anwendung finden?

Gedanken und Beobachtungen

Unterrichtseinheit 8
Anmerkungen für Lehrkräfte

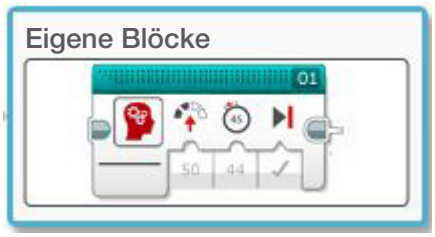
ROBOT EDUCATOR-ANLEITUNGEN

Die folgenden Robot Educator-Anleitungen helfen Lehrern und Schülern bei der Lösung der Aufgaben.

NEUE ROBOT EDUCATOR-ANLEITUNGEN

Komplexere Programme > Variablen

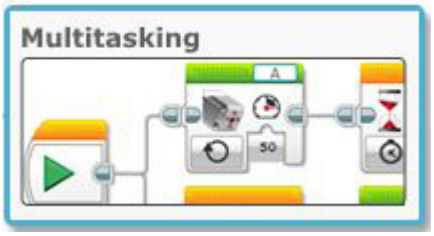
Werkzeuge > Eigene Blöcke



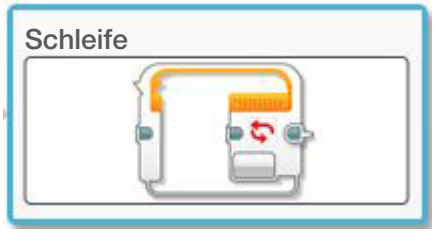
BEREITS BEHANDELTE ROBOT EDUCATOR-ANLEITUNGEN

Grundlagen > Geradeausfahrt

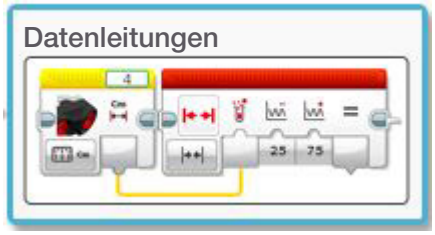
Komplexere Programme > Multitasking



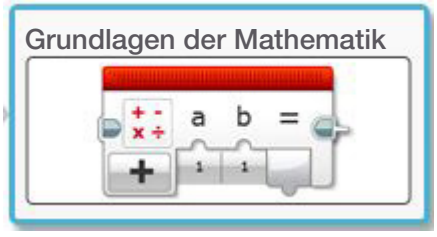
Komplexere Programme > Schleife



Komplexere Programme > Datenleitungen

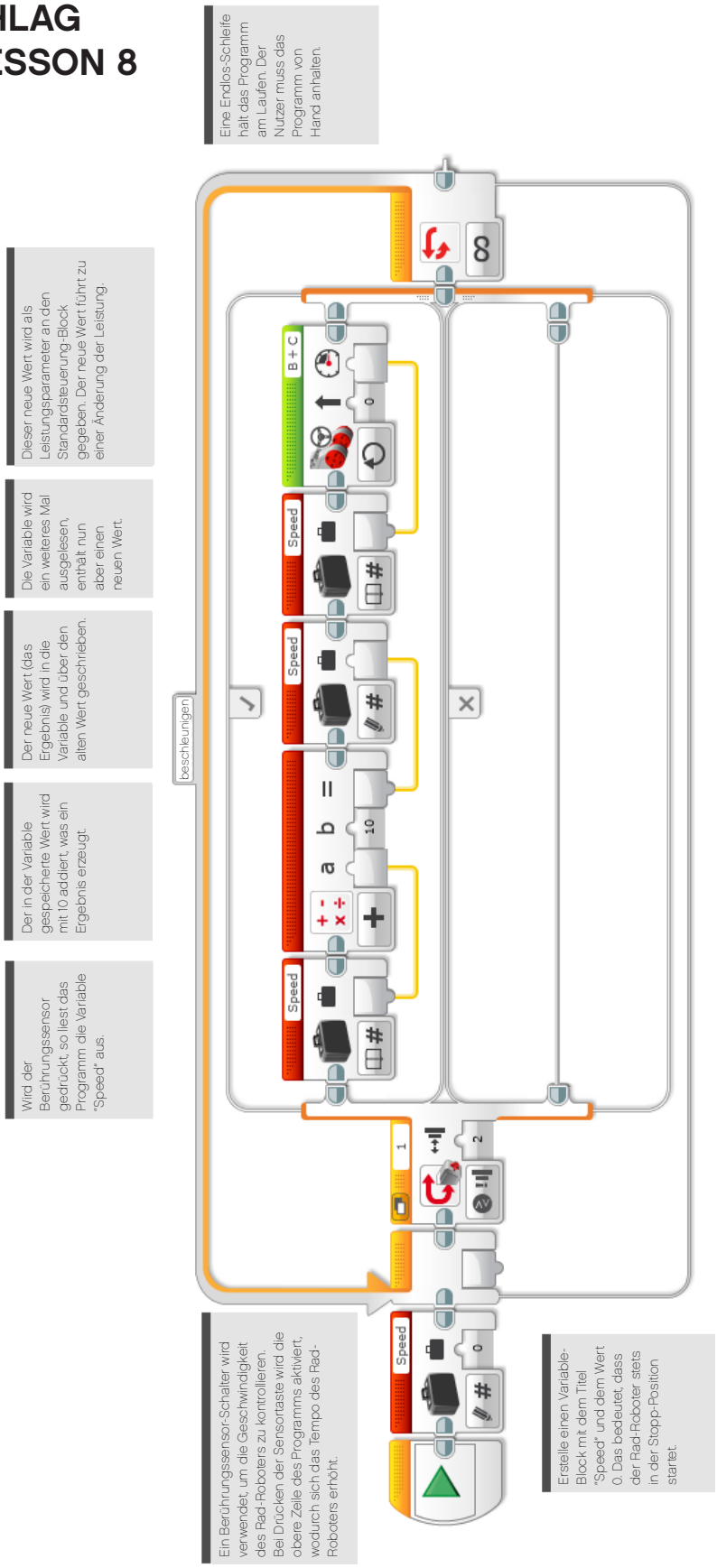


Komplexere Programme > Mathe - Grundlagen

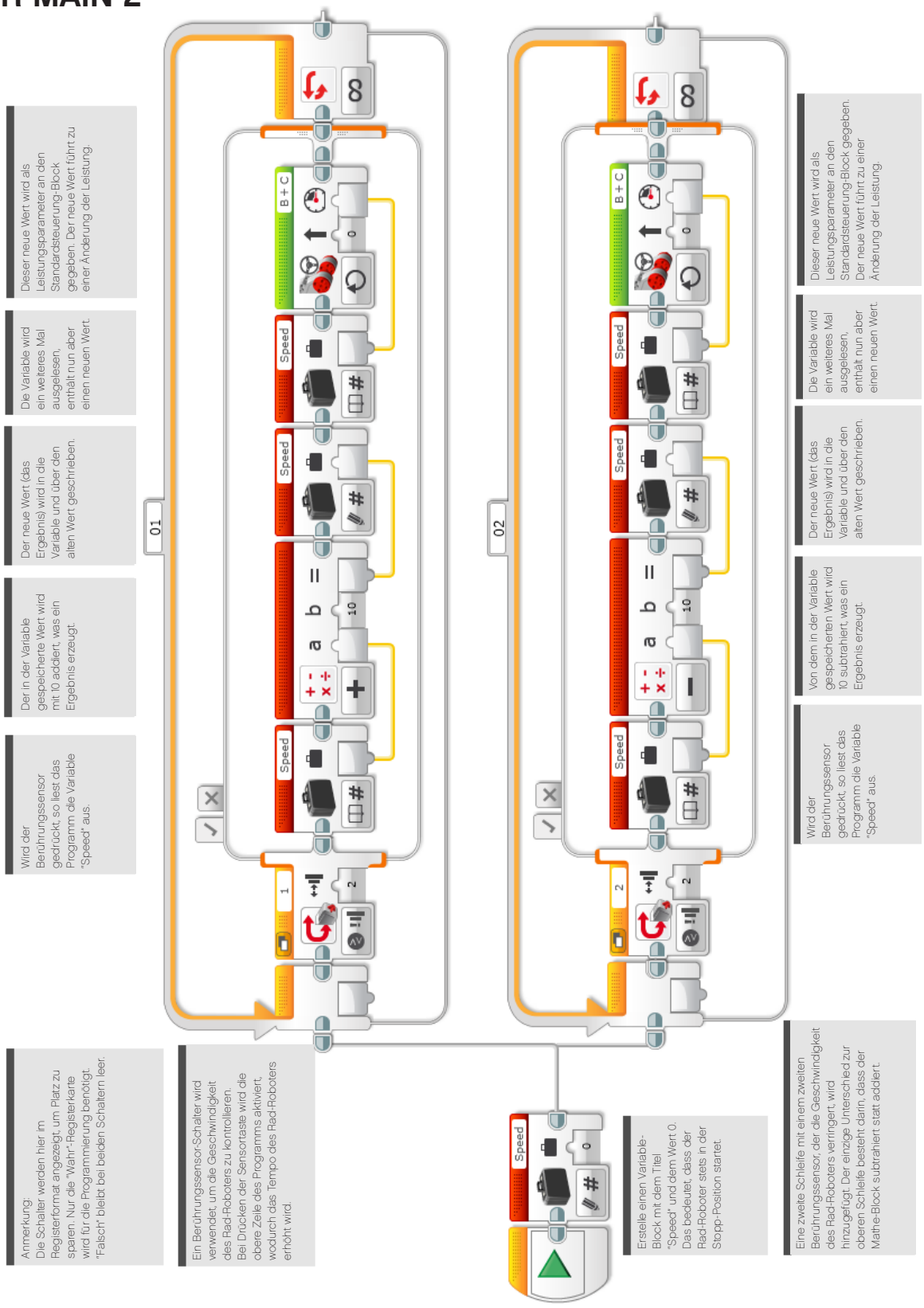


ANHANG FÜR UNTERRICHTSEINHEIT 8: BILDER UND PROGRAMME

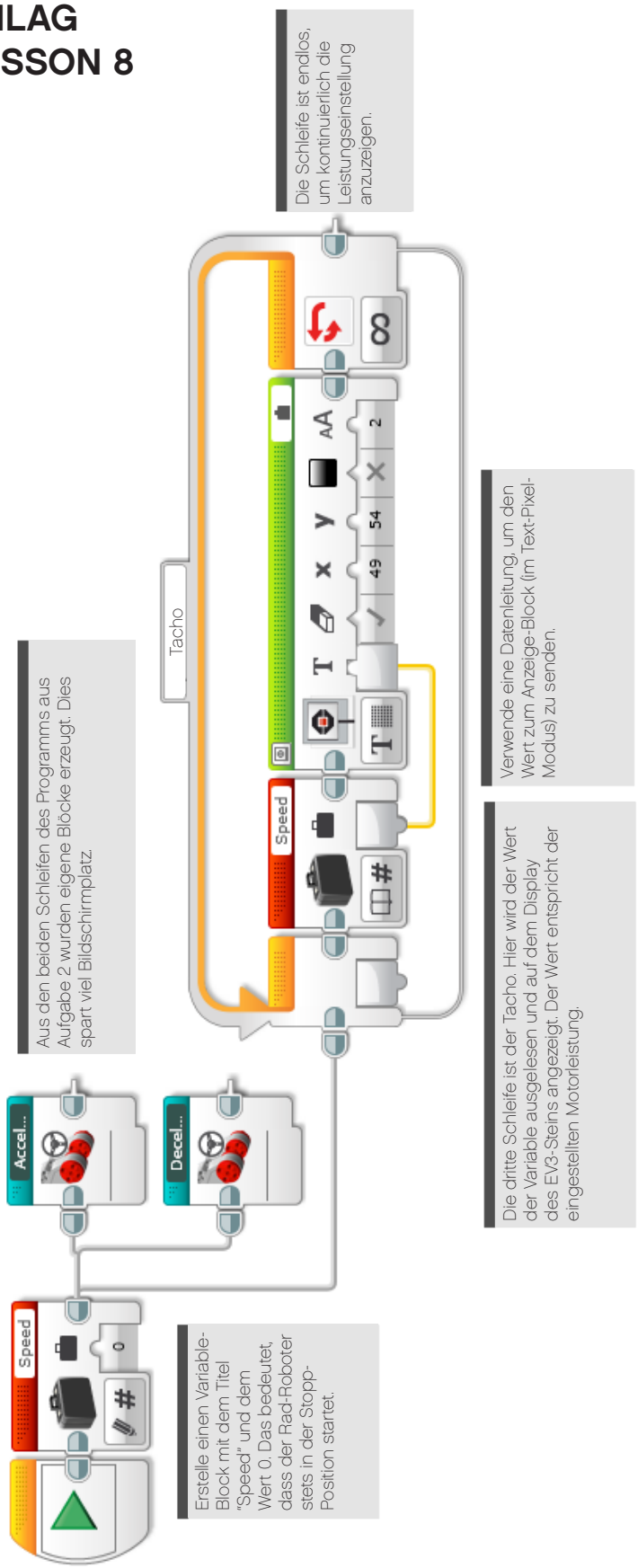
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 8
REITER MAIN 1



LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 8
REITER MAIN 2



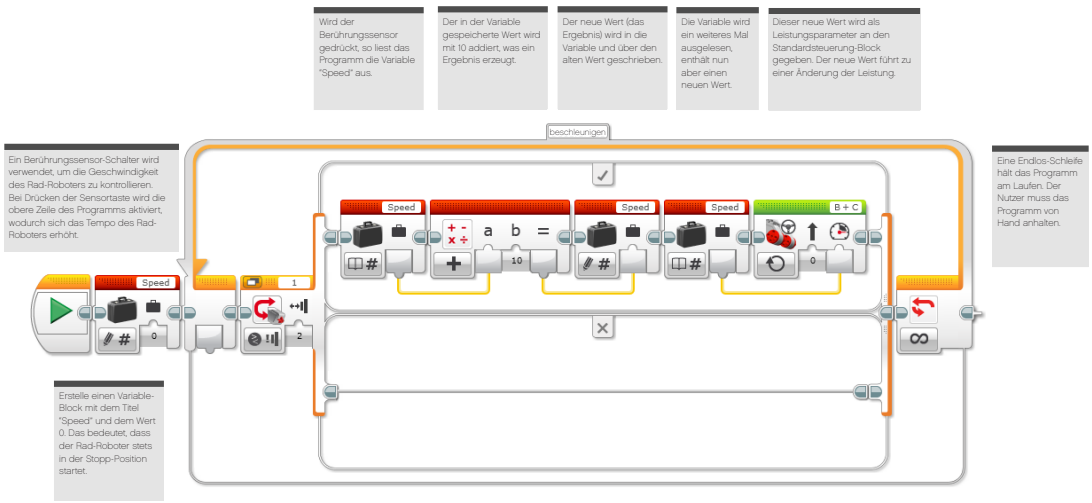
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 8
REITER MAIN 3



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session8_1.c

LEGO Start Page | Lesson 8_1.c*

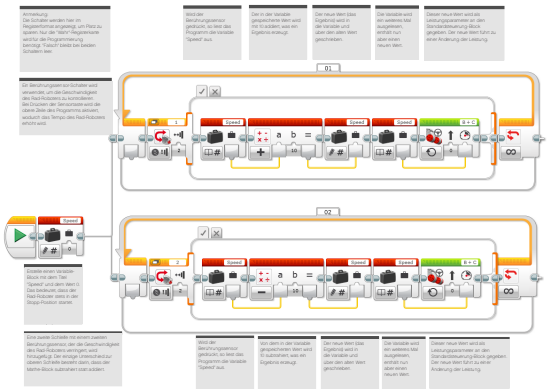
```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//

task main()
{
  //Create an integer (whole number) variable to store our speed value
  int speed = 0;

  //Repeat our control loop forever
  while(true)
  {
    //When I press the touch sensor button
    if(getTouchValue(touchSensor) == 1)
    {
      //Add 10 to our 'speed' variable
      speed = speed + 10;

      //Create a loop to wait for the touch sensor button to be released
      while(getTouchValue(touchSensor) == 1)
      {
        //Wait for button to be released
      }
    }

    //Set motorB and motorC speed to the value of the 'speed' variable
    setMotorSpeed(motorB, speed);
    setMotorSpeed(motorC, speed);
  }
}
```



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session8_2.c

LEGO Start Page | Lesson 8_2.c*

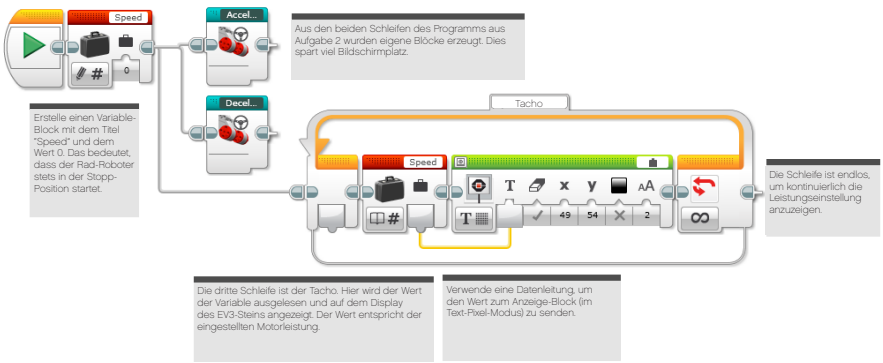
```
#pragma config(Sensor, S1, touchSensor, sensorEV3_Touch)
#pragma config(Sensor, S2, touchSensor2, sensorEV3_Touch)
#pragma config(Sensor, S3, colorSensor, sensorEV3_Color)
#pragma config(Sensor, S4, sonarSensor, sensorEV3_Ultrasonic)
#pragma config(Motor,  motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor,  motorC, leftMotor, tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//

task main()
{
  //Create an integer (whole number) variable to store our speed value
  int speed = 0;

  //Repeat our control loop forever
  while(true)
  {
    //When I press the touch sensor button
    if(getTouchValue(touchSensor) == 1)
    {
      //Add 10 to our 'speed' variable
      speed = speed + 10;

      //Create a loop to wait for the touch sensor button to be released
      while(getTouchValue(touchSensor) == 1)
      {
        //Wait for button to be released
      }
    }

    //When I press the touch sensor #2 button
    if(getTouchValue(touchSensor2) == 1)
    {
      //Subtract 10 to our 'speed' variable
      speed = speed - 10;

      //Create a loop to wait for the touch sensor #2 button to be released
      while(getTouchValue(touchSensor2) == 1)
      {
        //Wait for button #2 to be released
      }
    }

    //Set motorB and motorC speed to the value of the 'speed' variable
    setMotorSpeed(motorB, speed);
    setMotorSpeed(motorC, speed);
  }
}
```



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session8_3.c

LEGO Start Page | Lesson 8_3.c*

```
#pragma config(Sensor, S1, touchSensor, sensorEV3_Touch)
#pragma config(Sensor, S2, touchSensor2, sensorEV3_Touch)
#pragma config(Sensor, S3, colorSensor, sensorEV3_Color)
#pragma config(Sensor, S4, sonarSensor, sensorEV3_Ultrasonic)
#pragma config(Motor,  motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor,  motorC, leftMotor, tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//

//Create an integer (whole number) variable to store our speed value
int speed = 0;

void Accelerate()
{
  //When I press the touch sensor button
  if(getTouchValue(touchSensor) == 1)
  {
    //Add 10 to our 'speed' variable
    speed = speed + 10;

    //Create a loop to wait for the touch sensor button to be released
    while(getTouchValue(touchSensor) == 1)
    {
      //Wait for button to be released
    }
  }
}

void Decelerate()
{
  //When I press the touch sensor #2 button
  if(getTouchValue(touchSensor2) == 1)
  {
    //Subtract 10 to our 'speed' variable
    speed = speed - 10;

    //Create a loop to wait for the touch sensor #2 button to be released
    while(getTouchValue(touchSensor2) == 1)
    {
      //Wait for button #2 to be released
    }
  }
}

task main()
{
  //Create a string to store our text to be displayed to the LCD
  string displaySpeed;

  //Repeat our control loop forever
  while(true)
  {
    Accelerate();
    Decelerate();

    //Format our string to display "Speed: 50" or similar values
    stringFormat(displaySpeed, "Speed: %d", speed);

    //Draw the string on the LCD screen
    drawTextAt(49, 54, displaySpeed);

    //Set motorB and motorC speed to the value of the 'speed' variable
    setMotorSpeed(motorB, speed);
    setMotorSpeed(motorC, speed);
  }
}
```

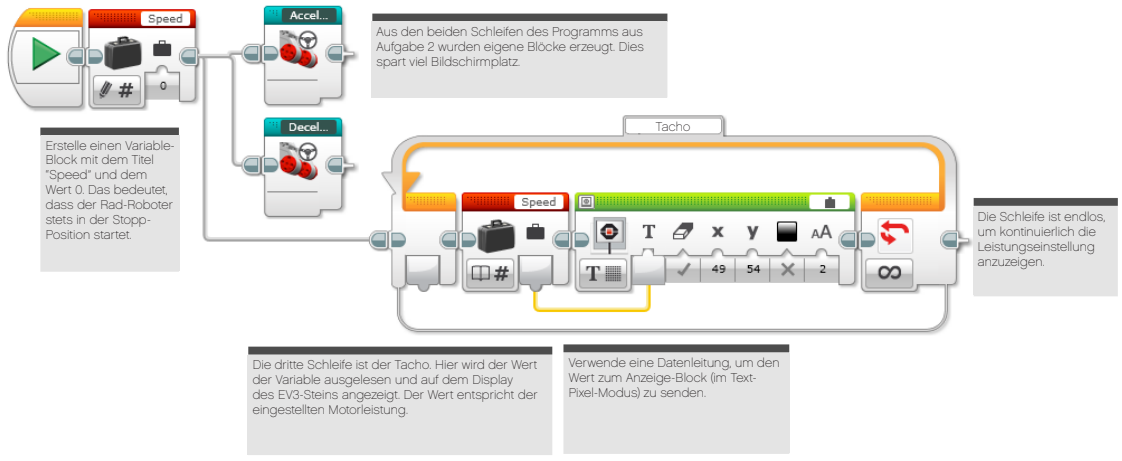


LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session8_3_alternative.c

Eine alternative Lösung wie hier wird angeboten, wenn die Programmierer nicht nur eine Kopie des EV3-Programms angefertigt, sondern auch ein zweites textbasiertes Programm von maximaler Effizienz entwickelt haben.

LEGO Start Page | Lesson 8_3_alternative.c*

```
#pragma config(Sensor, S1, touchSensor, sensorEV3_Touch)
#pragma config(Sensor, S2, touchSensor2, sensorEV3_Touch)
#pragma config(Sensor, S3, colorSensor, sensorEV3_Color)
#pragma config(Sensor, S4, sonarSensor, sensorEV3_Ultrasonic)
#pragma config(Motor,  motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor,  motorC, leftMotor, tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//

//Create an integer (whole number) variable to store our speed value
int speed = 0;

#include "myCodeLibrary.c"

task main()
{
  //Create a string to store our text to be displayed to the LCD
  string displaySpeed;

  //Repeat our control loop forever
  while(true)
  {
    Accelerate();
    Decelerate();

    //Format our string to display "Speed: 50" or similar values
    stringFormat(displaySpeed, "Speed: %d", speed);

    //Draw the string on the LCD screen
    drawTextAt(49, 54, displaySpeed);

    //Set motorB and motorC speed to the value of the 'speed' variable
    setMotorSpeed(motorB, speed);
    setMotorSpeed(motorC, speed);
  }
}
```



mindstorms education EV3

UNTERRICHTSEINHEIT 9

Streunende Roboter

LEGOeducation.de

LEGO education

Streunende Roboter

Diese Unterrichtseinheit konzentriert sich auf Arrays und wie diese verwendet werden, um die Aktionen des Rad-Roboters zu kontrollieren.

Im Lauf der Einheit finden die Schüler heraus, was Arrays sind, wie sie funktionieren und warum sie in der Computerprogrammierung wichtig sind. Sie lernen zudem, wie sich ein Array in ihr Programm integrieren lässt.

Die Schüler entwickeln ein Programm, das die Funktion bestimmter elektronischer Spielzeuge und Roboter (z.B. den Bee Bot) nachahmt.

Während der Einführung demonstrieren Sie anhand des Farbsortierers ein Array in Aktion. Die Schüler können sich das zugehörige Video ansehen, das in der Programmier-Software zu finden ist. Oder sie bauen das Modell selbst (was jedoch im Vorfeld zu dieser Einheit geschehen sollte).

Die Bauanleitung für das Farbsortierer-Modell befindet sich ebenfalls in der EV3-Software im Bereich "Modell: Basis-Set". Neben der Anleitung gibt es auch ein Beispielprogramm, das sich einfach auf das fertige Modell herunterladen lässt.

Diese Unterrichtseinheit unterscheidet sich von den anderen, da sie nur zwei Aufgaben umfasst. Der zu ihrer Bearbeitung nötige Zeitaufwand ist jedoch beträchtlich.

ERGEBNISSE

In dieser Einheit lernen die Schüler,

- Datenstrukturen wie Listen, Tabellen oder Arrays sinnvoll einzusetzen.
- einfache Boole'sche Logik zu verstehen und einige Einsatzmöglichkeiten zu nennen.
- die Stein-Tasten für die Bewegungssteuerung ihres Rad-Roboters zu nutzen.
- den Variable-Block zu verwenden, um Information zu speichern.
- den Array-Operationen-Block zu verwenden.

EINFÜHRUNG

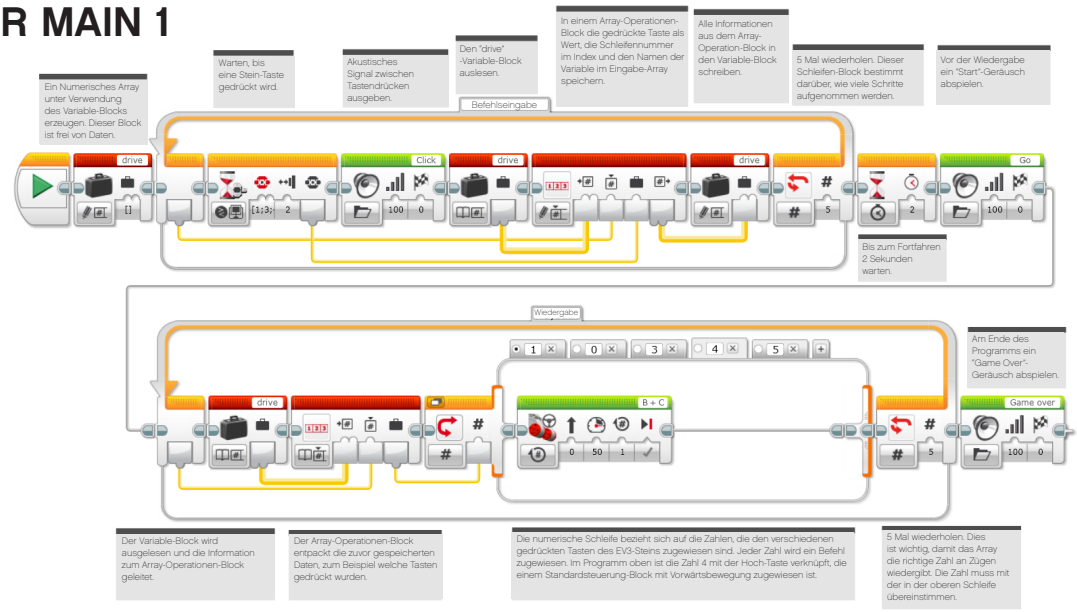
- Erklären Sie den Schülern, dass sie ihren Rad-Roboter so programmieren werden, dass dieser auf Befehle reagiert, die über die Tasten des EV3-Steins (vorwärts, rückwärts, links, rechts) eingegeben werden.
- Weisen Sie darauf hin, dass ein Array eine kurzzeitige Möglichkeit darstellt, eine gewisse Menge Zahlen in Reihenfolge zu speichern. Diese Zahlen können später im Programm verwendet werden.
- Führen Sie einige Male das Farbsortierer-Modell vor und lassen sie die Schüler in Pseudo-Code beschreiben, was hier passiert.
- Vergleichen Sie die Programmierung des Rad-Roboters mit der eines Satellitennavigationsgeräts. Lassen Sie die Schüler in Gruppen diskutieren, wie ein Programm aussehen könnte, das ihren Rad-Roboter einen Kurs abfahren lässt. Fragen Sie, wie Satellitennavigation funktioniert.

- Sagen Sie den Schülern, dass sie in dieser Einheit den Array-Operationen-Block benötigen. Erklären Sie, dass dieser Block verwendet wird, um Daten im ebenfalls benötigten Variablen-Block zu speichern. Diese Daten können abgefragt und in einem Programm verwendet werden.
- Zeigen Sie ein weiteres Mal das Farbsortierer-Modell. Lassen Sie die Schüler über die Funktionsweise diskutieren. Weisen Sie darauf hin, dass sich das Programm an die Reihenfolge der Farbklötzchen 'erinnert', um sie in die entsprechenden Becher einzusortieren.
- Ließe sich so etwas auch für die Steuerung des Rad-Roboters nutzen?

HAUPTAUFGABE 1

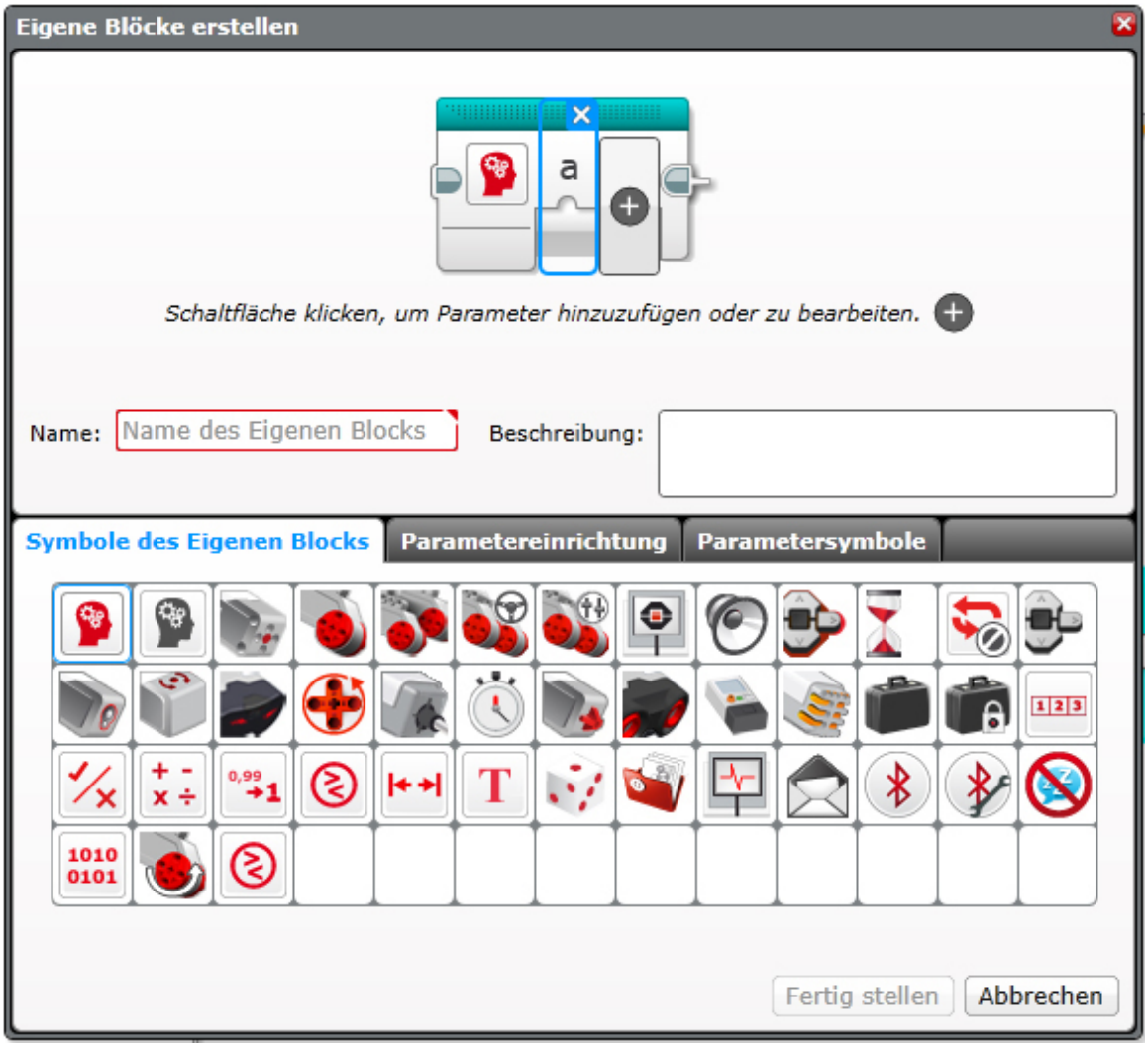
- Lassen Sie die Schüler das Farbsortierer-Programm erkunden und in Gruppen erklären, was in dem Programm geschieht. Verstehen sie es? Lassen Sie die Schüler das Programm ausarbeiten.
- Führen Sie die Schüler durch das Programm und vergewissern Sie sich, dass sie die Grundlagen der Erzeugung einer Variable durch Verwendung des Array-Operationen-Blocks verstehen, sowie die Notwendigkeit, eine Variable zu lesen und zu schreiben. Die Schüler sollten zudem wissen, dass der Prozess wie folgt abläuft:
  Variable lesen, schreiben, ein weiteres Mal lesen und die Information verwenden.
- Fordern Sie die Schüler auf, ein Programm zu schreiben, das ihren Rad-Roboter einen Kurs entlang fahren lässt. Beschränken Sie das Programm zunächst auf fünf Schritte.
- Erklären Sie, dass das Programm in zwei Abschnitte eingeteilt werden muss. Im ersten werden die Informationen gesammelt, im zweiten die gesammelten Informationen genutzt - ganz so, wie dies auch beim Farbsortierer geschieht.

LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 9
REITER MAIN 1

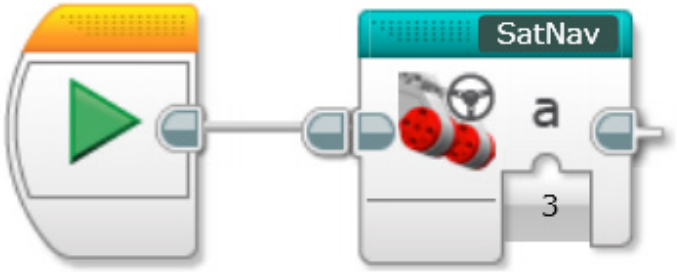


HAUPTAUFGABE 2

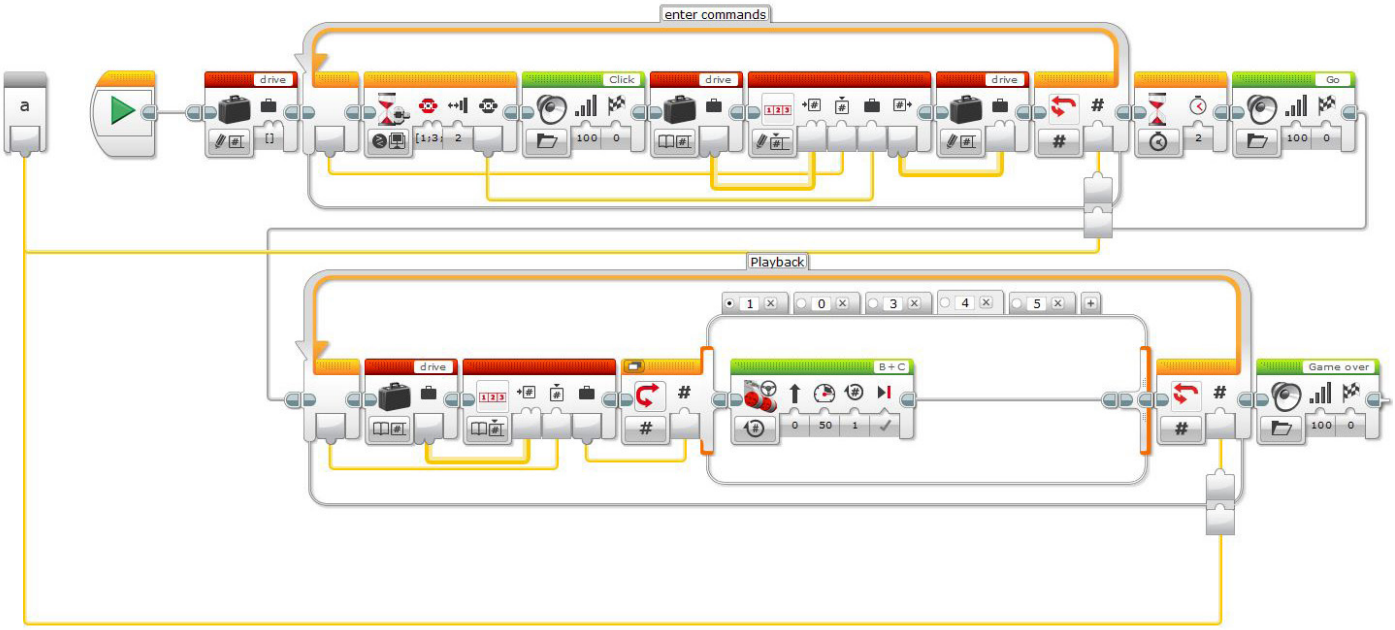
- In Aufgabe 1 ist die Schleifen-Zahl limitiert.
- Zeigen Sie den Schülern, wie man einen eigenen Block erstellt und dann einen Parameter hinzufügt.
- Wie Sie feststellen können, wird der Programmpalette ein neuer Reiter hinzugefügt, der zeigt, was im eigenen Block enthalten ist. Um den eigenen Block fertigzustellen, muss der Nutzer den 'a'-Parameter mit zwei Eingaben innerhalb des Programms verbinden. In unserem Programm sind das die zwei Schleifen-Blöcke. Auf diese Weise können Daten direkt in den eigenen Block eingegeben werden, statt in das enthaltene Programm.
- Weisen Sie darauf hin, dass die Schüler auf diese Weise einfach die gewünschte Zahl an Schritten eingeben können. Die Schüler sollen verstehen, dass ihnen das Hinzufügen eines Parameters beim Entwerfen eigener Blöcke erlaubt, Daten in den erstellten eigenen Block einzugeben.



LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 9
REITER MAIN 2



LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 9
REITER SATNAV



ABSCHLIESSENDE DISKUSSION

- Bringen Sie die Gruppen zusammen und lassen sie sich gegenseitig ihre Programmiererfolge präsentieren. Fragen Sie, wie man die Programme hätte verbessern können.
- Ist es durch die Erstellung eines eigenen Blocks mit Parameter einfacher für die Schüler, die richtige Zahl an Schritten zu programmieren?
- Lassen Sie die Schüler über alternative Lösungen nachdenken. Kommen sie auf eine andere Möglichkeit, den Rad-Roboter über einen Kurs zu bewegen?

AUFGABEN DIESER EINHEIT

Heute lernst du, wie man Arrays verwendet. Der Array-Operationen-Block ist wichtig, um viele Informationen zu speichern und sie bei Bedarf zu verwenden. Unter Zuhilfenahme dieses Blocks wirst du ein automatisiertes Fahrzeug entwickeln, das sich schrittweise bewegt. Die Eingabe der Richtungsbefehle erfolgt über die Tasten des EV3-Steins. Viel Erfolg!

AUFGABE 1

Nachdem du den Farbsortierer in Aktion erlebt hast, ist es nun an dir, ein Array zu erzeugen und deinen Rad-Roboter so zu programmieren, dass er mittels der EV3-Stein-Tasten im Raum umherbewegt werden kann. Die vier EV-Stein-Tasten lassen sich als Steuerung (links, rechts, vor, zurück) verwenden.

Beschränke das Programm zunächst auf fünf Befehle, indem du den Wert 5 in den Schleifen-Block eingibst.

Tipp 1: Dein Programm besteht aus zwei getrennten Phasen:

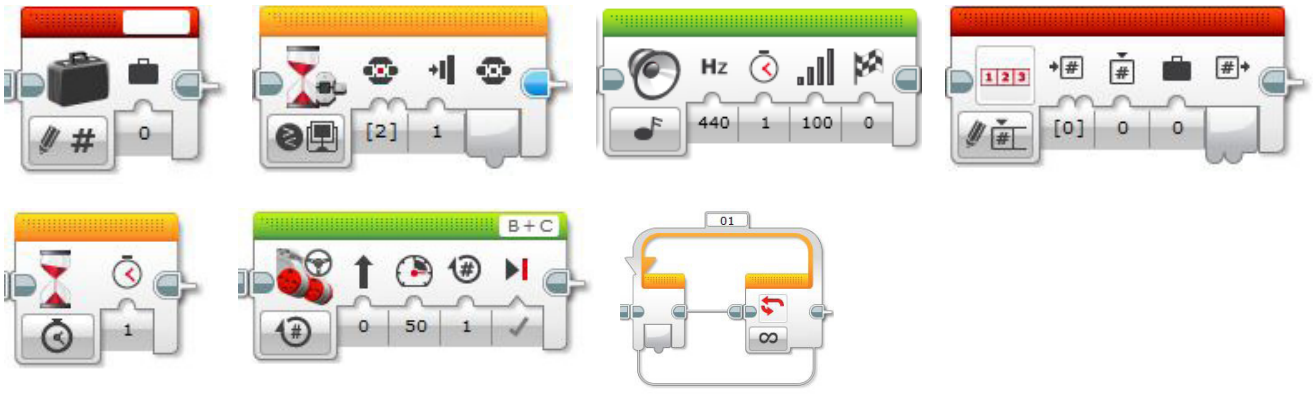
1. Daten sammeln
2. Daten verwenden

Tipp 2: In dieser Aufgabe werden zwei Schleifen-Blöcke benötigt, um die zwei Phasen zu realisieren.

Tipp 3: Der Einsatz des Variable-Blocks erfordert oft einen dreistufigen Prozess:

Variable-Block auslesen, Information hinzufügen und dann die Variable in den Block schreiben, um neue Daten zu speichern.

Geeignete Blöcke



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

AUFGABE 2

Erstelle einen eigenen Block, um auf einfache Weise die Zahl der Schritte in deinem Programm zu bestimmen.

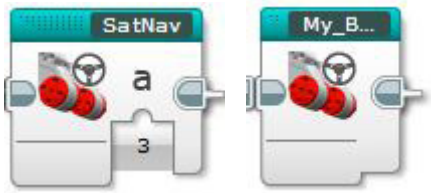
Um die Anzahl der Bewegungsschritte von 5 zu einem beliebigen Wert zu ändern, musst du beide Schleifen-Blöcke im Programm editieren. Dies lässt sich erheblich einfacher bewerkstelligen, wenn du einen eigenen Block mit einem Parameter erstellst. Der eigene Block erlaubt dir, die Schleifenzahl auf einfache und übersichtliche Weise zu verändern. Du musst den eigenen Block aus dem Programm erstellen, das du in Aufgabe 1 entwickelt hast.

Tipp 1: Bei der Erstellung eines eigenen Blocks musst du alle Blöcke markieren, die einbezogen werden sollen, allerdings NICHT den Start-Block.

Tipp 2: Wenn du zu einem späteren Zeitpunkt Parameter eingeben musst, dann stelle sicher, dass ein Parameter dem eigenen Block wie unten gezeigt hinzugefügt wurde. Verwende die '+'-Taste bei der Erstellung des Blocks.

Tipp 3: Der Parameter muss mit der Eingabe am Block innerhalb des Programms verbunden werden. In unseren Fall sind das die beiden Schleifen.

Geeignete Blöcke



Plane zunächst dein Programm und mache dir hier Notizen in Pseudocode.

Unterrichtseinheit 9
Arbeitsblätter

Nach dem Programmieren ist es wichtig, seine Gedanken und Beobachtungen niederzuschreiben. Denke über folgende Punkte nach und notiere dann im Kasten unten, wie diese Programmiersitzung verlaufen ist.

- Wie könntest du dein Programm verbessern?
- Könnte dein Programm geradliniger sein? Hast du zu viele Blöcke verwendet? Lässt sich das Programm effizienter gestalten?
- Wo könnte dein Programm in der 'realen Welt' Anwendung finden?

Gedanken und Beobachtungen

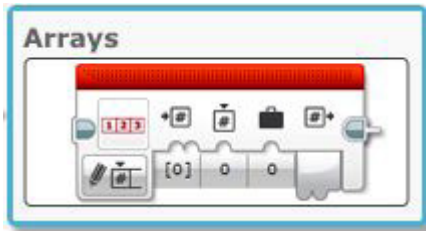
Unterrichtseinheit 9
Anmerkungen für Lehrkräfte

ROBOT EDUCATOR-ANLEITUNGEN

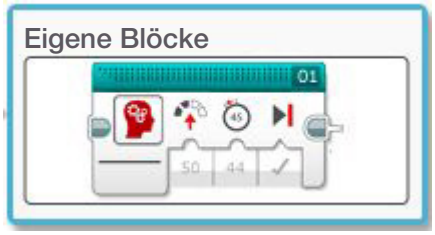
Die folgenden Robot Educator-Anleitungen helfen Lehrern und Schülern bei der Lösung der Aufgaben.

NEUE ROBOT EDUCATOR-ANLEITUNGEN

Komplexere Programme > Arrays

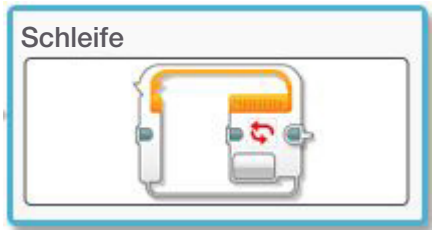


Werkzeuge > Eigene Blöcke

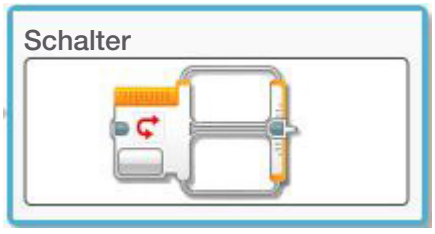


BEREITS BEHANDELTE ROBOT EDUCATOR-ANLEITUNGEN

Komplexere Programme > Schleife



Komplexere Programme > Schalter



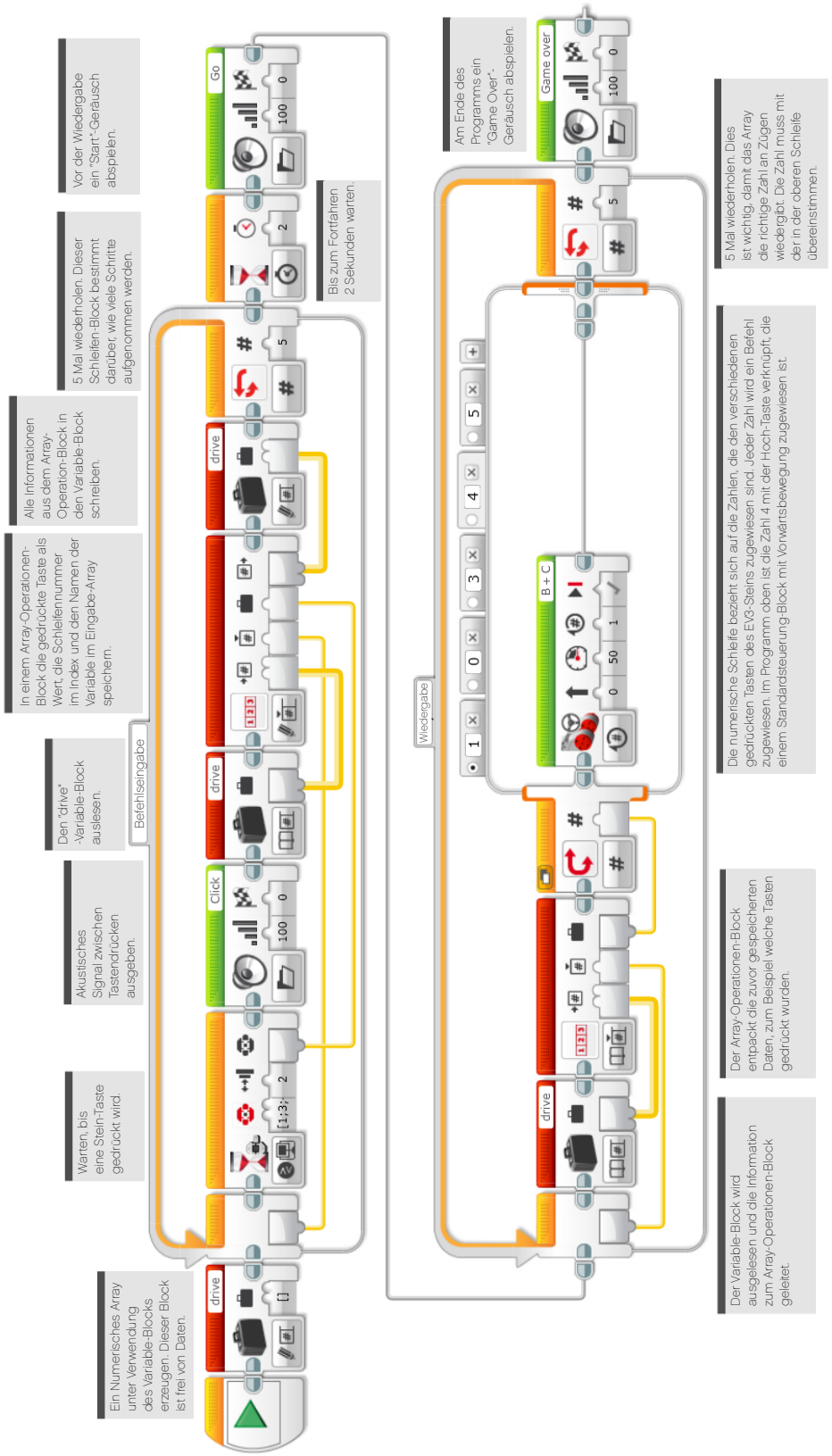
Komplexere Programme > Datenleitungen

Grundlagen > Geradeausfahrt

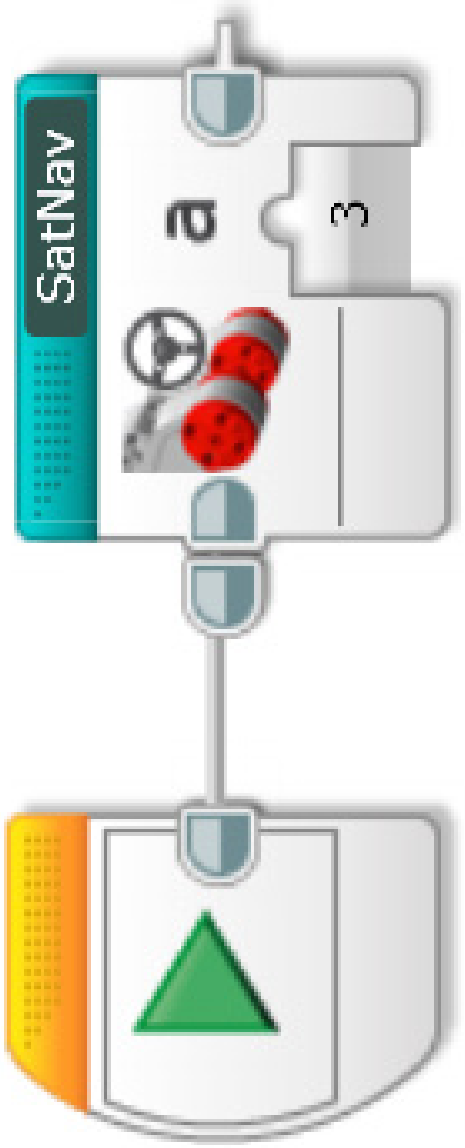
Grundlagen > Kurvenfahrt

ANHANG FÜR UNTERRICHTSEINHEIT 9: BILDER UND PROGRAMME

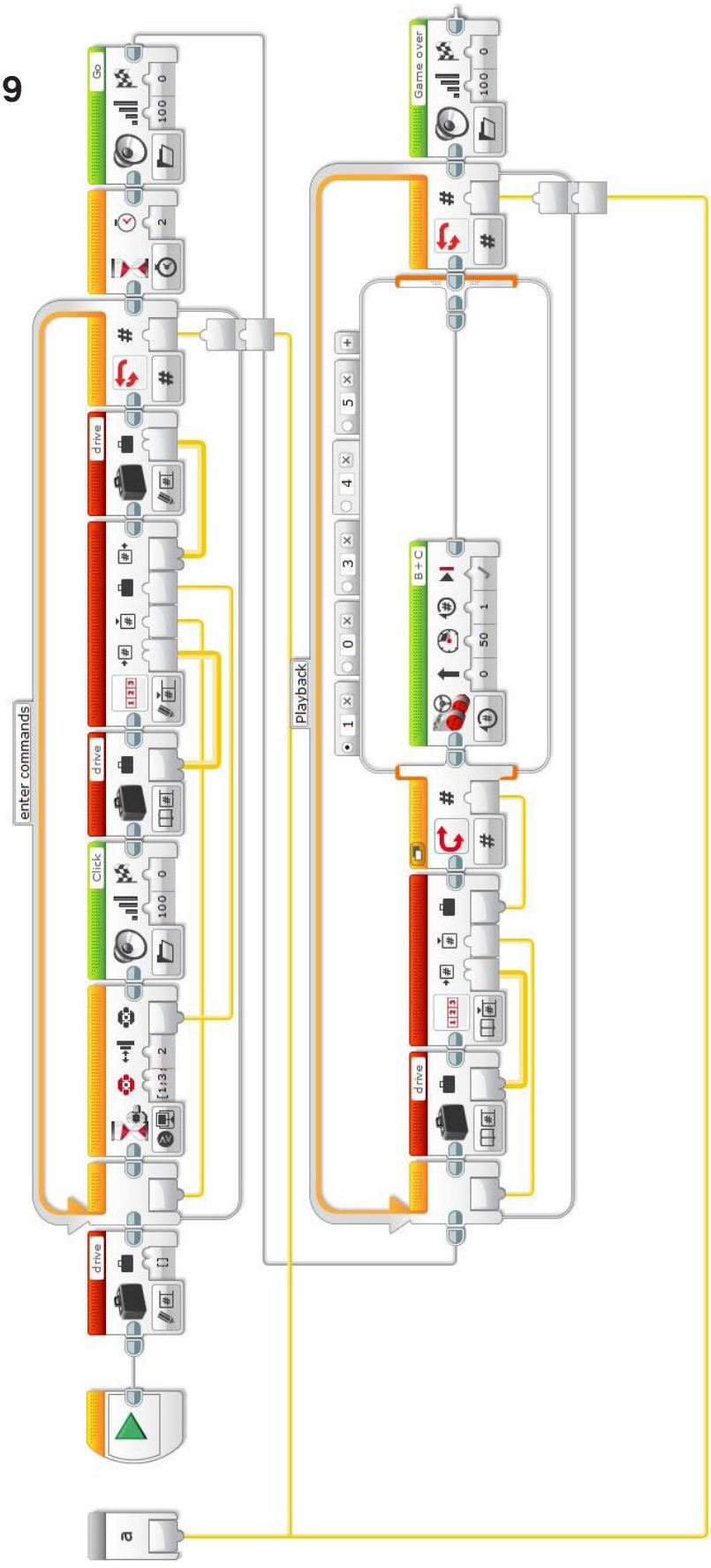
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 9
REITER MAIN 1



LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 9
REITER MAIN 2



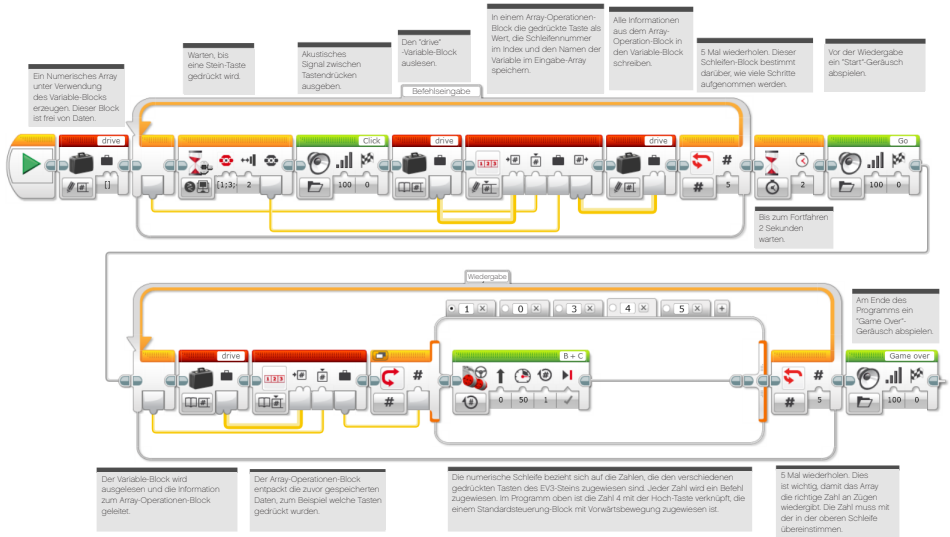
LÖSUNGSVORSCHLAG
DATEINAME CS LESSON 9
REITER SATNAV



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session9_1.c

LEGO Start Page | Lesson 9_1.c*

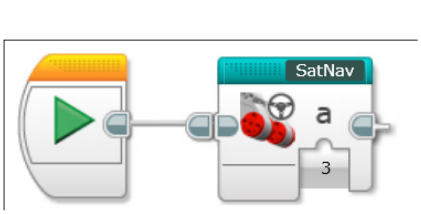
```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//
/*
Up Button    = 1: Down Button  = 3: Right Button = 4: Left Button  = 5
*/

int drive[5];

task main()
{
  for(int i = 0; i < 5; i++) //i = i + 1
  {
    while(getButtonPress(buttonAny) == 0)
    {
      //Wait for Any Button to be pressed
    }

    if(getButtonPress(buttonUp) == 1)           drive[i] = 1;
    else if(getButtonPress(buttonDown) == 1)    drive[i] = 3;
    else if(getButtonPress(buttonRight) == 1)   drive[i] = 4;
    else if(getButtonPress(buttonLeft) == 1)    drive[i] = 5;

    playSoundFile("Click.rsf");

    while(getButtonPress(buttonAny) == 1)
    {
      //Wait for All Buttons to be released
    }
  }

  sleep(2000);
  playSoundFile("Go.rsf");

  for(int i = 0; i < 5; i++)
  {
    if(drive[i] == 1)
    {
      //Forward Code
      moveMotorTarget(motorB, 360, 50);
      moveMotorTarget(motorC, 360, 50);
      waitUntilMotorStop(motorB);
      waitUntilMotorStop(motorC);
    }
    if(drive[i] == 3)
    {
      //Backward Code
      moveMotorTarget(motorB, -360, -50);
      moveMotorTarget(motorC, -360, -50);
      waitUntilMotorStop(motorB);
      waitUntilMotorStop(motorC);
    }
    if(drive[i] == 4)
    {
      //Turn Right Code
      moveMotorTarget(motorB, 360, 50);
      waitUntilMotorStop(motorB);
    }
    if(drive[i] == 5)
    {
      //Turn Left Code
      moveMotorTarget(motorC, 360, 50);
      waitUntilMotorStop(motorC);
    }
  }

  playSoundFile("Game Over.rsf");
  sleep(5000);
}
```

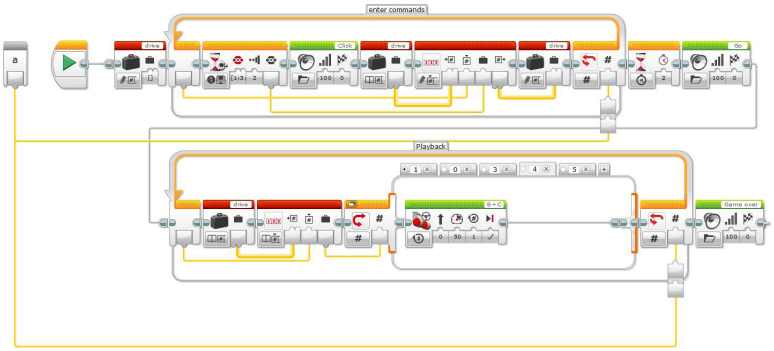


LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session9_2.c

LEGO Start Page | Lesson 9_2.c*

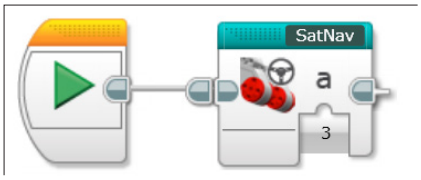
```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//

/*
Up Button    = 1
Down Button  = 3
Right Button = 4
Left Button  = 5
*/

//Maximum of 100 steps
int drive[100];

void recordSteps(int numberOfSteps)
{
  for(int i = 0; i < 5; i++) //i = i + 1
  {
    while(getButtonPress(buttonAny) == 0)
    {
      //Wait for Any Button to be pressed
    }

    if(getButtonPress(buttonUp) == 1)            drive[i] = 1;
    else if(getButtonPress(buttonDown) == 1)     drive[i] = 3;
    else if(getButtonPress(buttonRight) == 1)    drive[i] = 4;
    else if(getButtonPress(buttonLeft) == 1)     drive[i] = 5;

    playSoundFile("Click.rsf");

    while(getButtonPress(buttonAny) == 1)
    {
      //Wait for All Buttons to be released
    }
  }
}
```

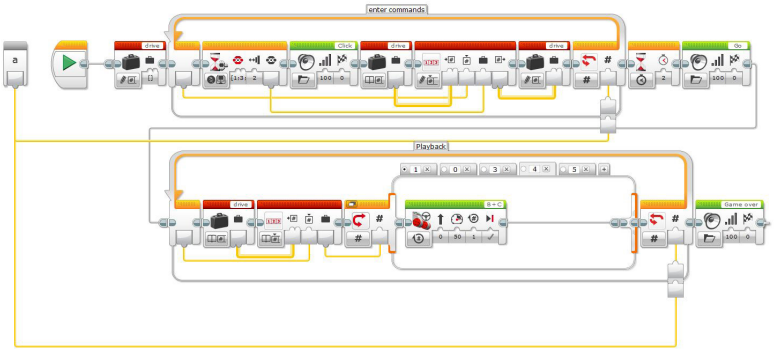




LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session9_2.c

```
void playSteps(int numberOfSteps)
{
  for(int i = 0; i < numberOfSteps; i++)
  {
    if(drive[i] == 0)
    {
      //No Step Found, End Code.
      return;
    }

    if(drive[i] == 1)
    {
      //Forward Code
      moveMotorTarget(motorB, 360, 50);
      moveMotorTarget(motorC, 360, 50);
      waitUntilMotorStop(motorB);
      waitUntilMotorStop(motorC);
    }
    if(drive[i] == 3)
    {
      //Backward Code
      moveMotorTarget(motorB, -360, -50);
      moveMotorTarget(motorC, -360, -50);
      waitUntilMotorStop(motorB);
      waitUntilMotorStop(motorC);
    }
    if(drive[i] == 4)
    {
      //Turn Right Code
      moveMotorTarget(motorB, 360, 50);
      waitUntilMotorStop(motorB);
    }
    if(drive[i] == 5)
    {
      //Turn Left Code
      moveMotorTarget(motorC, 360, 50);
      waitUntilMotorStop(motorC);
    }
  }
}

task main()
{
  recordSteps(8);

  sleep(2000);
  playSoundFile("Go.rsf");

  playSteps(8);

  playSoundFile("Game Over.rsf");
  sleep(5000);
}
```

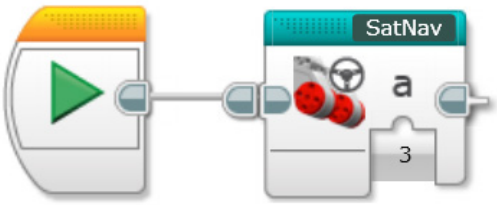




LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session9_2_alternative.c

LEGO Start Page | Lesson 9_2_alternative.c*

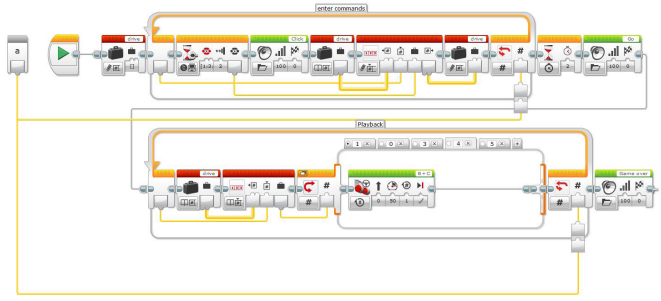
```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//
3
4  #include "9_2_Library.c"
5
6  task main()
7  {
8    recordSteps(3);
9
10   sleep(2000);
11   playSoundFile("Go.rsf");
12
13   playSteps(3);
14
15   playSoundFile("Game Over.rsf");
16   sleep(5000);
17 }
18
```



LÖSUNGSVORSCHLAG in ROBOTC
DATEINAME Session9_2_Library.c

LEGO Start Page | Lesson 9_2_alternative.c | lesson_9_2_Library.c*

```
/* Up Button   = 1: Down Button  = 3: Right Button = 4: Left Button  = 5 */

//Maximum of 100 steps
int drive[100];

void recordSteps(int numberOfSteps)
{
  for(int i = 0; i < 5; i++) //i = i + 1
  {
    while(getButtonPress(buttonAny) == 0)
    {
      //Wait for Any Button to be pressed
    }

    if(getButtonPress(buttonUp) == 1)           drive[i] = 1;
    else if(getButtonPress(buttonDown) == 1)    drive[i] = 3;
    else if(getButtonPress(buttonRight) == 1)   drive[i] = 4;
    else if(getButtonPress(buttonLeft) == 1)    drive[i] = 5;

    playSoundFile("Click.rsf");

    while(getButtonPress(buttonAny) == 1)
    {
      //Wait for All Buttons to be released
    }
  }
}

void playSteps(int numberOfSteps)
{
  for(int i = 0; i < numberOfSteps; i++)
  {
    if(drive[i] == 0)
    {
      //No Step Found, End Code.
      return;
    }

    if(drive[i] == 1)
    {
      //Forward Code
      moveMotorTarget(motorB, 360, 50);
      moveMotorTarget(motorC, 360, 50);
      waitUntilMotorStop(motorB);
      waitUntilMotorStop(motorC);
    }
    if(drive[i] == 3)
    {
      //Backward Code
      moveMotorTarget(motorB, -360, -50);
      moveMotorTarget(motorC, -360, -50);
      waitUntilMotorStop(motorB);
      waitUntilMotorStop(motorC);
    }
    if(drive[i] == 4)
    {
      //Turn Right Code
      moveMotorTarget(motorB, 360, 50);
      waitUntilMotorStop(motorB);
    }
    if(drive[i] == 5)
    {
      //Turn Left Code
      moveMotorTarget(motorC, 360, 50);
      waitUntilMotorStop(motorC);
    }
  }
}
```



mindstorms education EV3

UNTERRICHTSEINHEIT 10

Entwicklung eines führerlosen, automatisierten Rad-Roboters

LEGOeducation.de

education

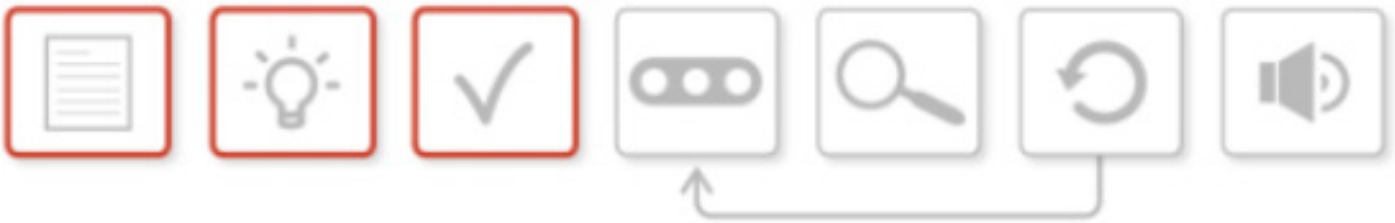
Entwicklung eines führerlosen, automatisierten Rad-Roboters

Im Lauf der letzten neun Unterrichtseinheiten haben die Schüler ihre Zeit damit verbracht, die verschiedenen Aspekte der Automatisierung eines Rad-Roboters kennenzulernen - von der Bereitstellung visueller Informationen für Passagiere und andere Verkehrsteilnehmer bis zur Befähigung des Fahrzeugs, einer festgelegten Route zu folgen. Jetzt ist es für die Schüler an der Zeit, ihr eigenes 'autonomes Auto' zu entwickeln und dabei ein Maximum der Funktionen zu integrieren, mit denen sie im Verlauf des Projekts Erfahrungen gesammelt haben.

Wie ein echtes Auto wird auch der Rad-Roboter der Schüler an seiner Ausstattung gemessen. Fahrzeughersteller bieten gewöhnlich eine breite Produktpalette an, von sehr einfachen Grundmodellen zu Autos der Oberklasse mit einer Vielfalt an Ausstattungsmerkmalen.

In diesem Projekt folgen die Schüler dem Konstruktionsprozess, der in LEGO® MINDSTORMS® Education EV3 Konstruktionsprojekte zugrunde gelegt wird - einem Unterrichtspaket, das für das Lernen in den MINT-Fächern entwickelt wurde (aber auch im Informatik-Kontext eingesetzt werden kann, wenn dort anwendungsorientierte Aktivitäten erforderlich sind).

Diese Unterrichtseinheit bündelt sämtliche Aspekte des Programmierens, welche die Schüler bislang kennengelernt haben. Die Schüler sollen in Gruppen die Voraussetzungen für die Entwicklung eines führerlosen, automatisierten Rad-Roboters erarbeiten. Dabei ist es sinnvoll, sich am Konstruktionsprozess zu orientieren, wie er im EV3 Konstruktionsprojekte-Unterrichtspaket beschrieben ist. Der heutige Ablauf gestaltet sich wie folgt:



- **Stellen der Konstruktionsaufgabe**
- **Ideensammlung**
- **Wahl der besten Lösung**
- Bauen und Programmieren einer Lösung
- Test und Analyse
- Überprüfung und Verbesserung
- Präsentation

Diese Einheit befasst sich mit den ersten drei Schritten. Die übrigen Punkte werden in den folgenden Unterrichtseinheiten abgearbeitet.

Unterrichtseinheit 10
Stundenplanung

ERGEBNISSE

In dieser Einheit lernen die Schüler,

- durch Entwicklung, Gebrauch und Bewertung rechnergestützter Abstraktionen reale Problemstellungen und physikalische Systeme abzubilden.

BEGRIFFE

Design, Gedankenskizze, Ideensammlung, Lösung, Konstruktionsaufgabe.

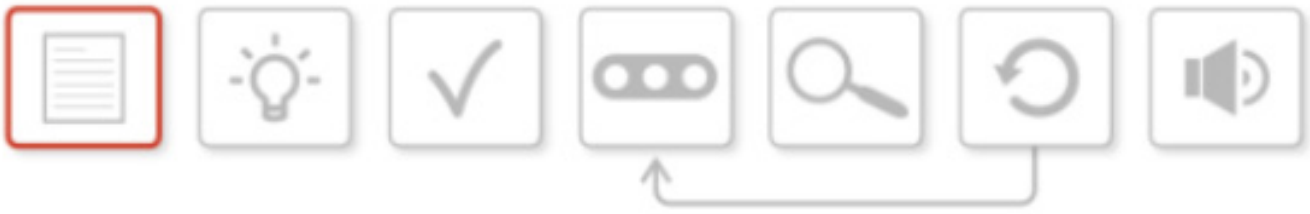
EINFÜHRUNG - DIE KONSTRUKTIONSAUFGABE

- Stellen Sie den weiter oben aufgeführten, siebenstufigen Ablauf des Konstruktionsprozesses vor. Erläutern Sie die einzelnen Phasen und sagen Sie den Schülern, dass sie die nächsten drei Unterrichtseinheiten mit diesen beschäftigt sein werden.
- Unterrichtseinheit 10 (diese Einheit): Stellen der Konstruktionsaufgabe, Ideensammlung (Gedankenskizze) sowie Wahl der besten Lösung
- Unterrichtseinheit 11: Bauen und Programmieren der Lösung, Test und Analyse
- Unterrichtseinheit 12: Überprüfung und Verbesserung des Ergebnisses, Präsentation der Konstruktion
- Die Konstruktionsaufgabe für dieses Projekt lautet:

  Entwickle und baue einen führerlosen, automatisierten Rad-Roboter, der von Punkt A nach Punkt B fährt und dabei Hindernissen ausweicht.

  Wenn wir die Klassifizierung von Autos heranziehen, dann wäre dies ein einfaches Standard-Modell. Die Schüler können ihrem Design jedoch weitere Funktionen hinzufügen, welche die 'Klasse' ihres Rad-Roboters erhöhen. Wie Sie diese Klassen letztlich bezeichnen wollen, bleibt ihnen und ihren Schülern überlassen.

- **Standard:** Der Rad-Roboter weicht Hindernissen aus.
- **Mittelklasse:** Zusätzlich reagiert das Fahrzeug auf Ampeln und Fußgänger.
- **Oberklasse:** Der Rad-Roboter lässt sich außerdem schlüssellos starten.
- **Luxusklasse**: Eine Geschwindigkeitsregelanlage ist ebenfalls an Bord.



HAUPTAUFGABE

IDEENSAMMLUNG (GEDANKENSKIZZE)

- Lassen Sie die Schüler eine Klasse für das Fahrzeug wählen, das konstruiert werden soll.
- Besprechen Sie mit den Schülern, dass es wichtig ist, sich die für die Entwicklung nötigen Programmier-Fähigkeiten zu erarbeiten.
- Lassen Sie die Schüler über folgende Fragen nachdenken: Welche Sensoren werden benötigt? Sind genug davon vorhanden? Welche Kompromisse muss man eingehen?
- Lassen Sie die Schüler Ideen für die Umsetzung der Konstruktionsaufgabe finden, ausarbeiten und mit den anderen teilen.
- Fordern Sie die Schüler auf, andere führerlose Fahrzeuge zu recherchieren, zum Beispiel:
- Docklands Light Railway (London)
- SkyLine (Flughafen Frankfurt)
- Dubai Metro

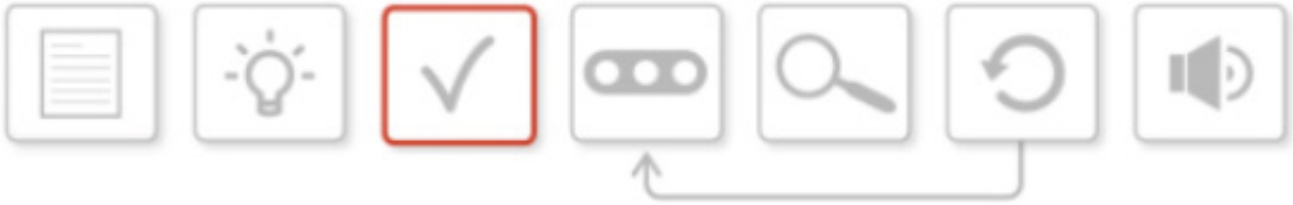
  Die Schüler könnten online nach Videos und Artikeln zu diesen Beispielen suchen, ihre Funktionen vergleichen und anschließend darüber diskutieren, welche Features in die eigene Konstruktion einfließen könnten.
- Die Schüler sollen mit den anderen LEGO®-Steinen im EV3-Baukasten experimentieren, um sich Ideen und Inspirationen zu holen. Die meisten werden das Robot Educator-Modell verwenden; manche Gruppen könnten aber auch abenteuerlustiger sein.

ROLLEN VERTEILEN

- Wer wird benötigt, um den skizzierten Roboter Realität werden zu lassen? Lassen Sie die Schüler herausfinden, welche Rollen in der Gruppe gebraucht werden. In jeder Gruppe wird es unterschiedliche Fertigkeiten geben. Ermuntern Sie die Schüler, über ihre Stärken und Schwächen zu reden. Wer ist ein guter Baumeister? Wer ist stark im Programmieren? Lassen Sie die Schüler untereinander Rollen zuweisen.
- **Mögliche Rollen im Team**
- Designer
- Modellbauer
- Rechercheur
- Vermarkter
- Programmierer

WAHL DER BESTEN LÖSUNG

- Die Schüler müssen dazu angespornt werden, die positiven und negativen Seiten ihrer Ideen abzuwägen und zu gewichten. Zuletzt sollen sie sich für ein finales Design entscheiden.
- Anschließend präsentieren die Schüler der Klasse ihre Wahl und erläutern die Gründe für ihre Entscheidung.

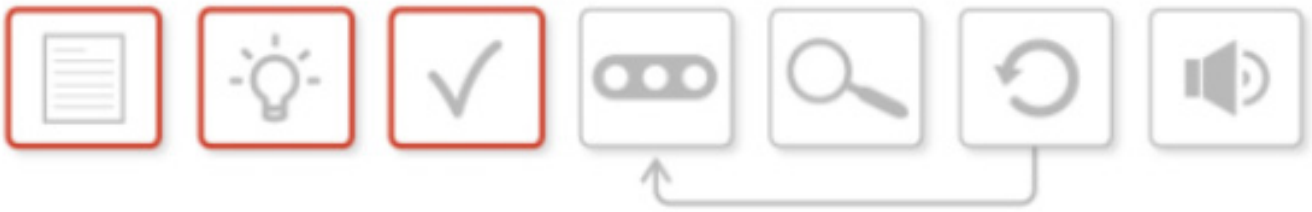


Die Schüler sollten jeden Schritt des Prozesses auf ihren Arbeitsblättern oder im Inhalts-Editor der EV3-Software schriftlich festhalten.

ABSCHLIESSENDE DISKUSSION

- Was sind die nächsten Schritte? Da nun eine Lösung gewählt wurde, können die Teams mit dem Bau und der Programmierung ihres Rad-Roboters beginnen.
- Teilen Sie den Schülern mit, dass sie in der nächsten Unterrichtseinheit ihren Rad-Roboter bauen, programmieren und testen werden.

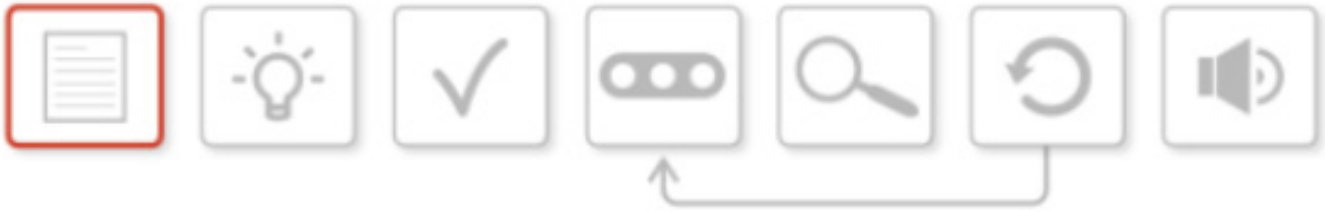
AUFGABEN DIESER UNTERRICHTSEINHEIT



Denkt über das Design eures Roboters nach.

Heute beschäftigt ihr euch mit den ersten drei Aspekten: Ihr erhaltet eine Kurzdarstellung der Konstruktionsaufgabe, sammelt in der Gruppe Ideen und wählt schließlich eine Lösung.

Euer Lehrer hat euch eine Kurzanweisung gegeben. Hier ist so nochmal zur Erinnerung:



ENTWICKELT UND BAUT EINEN FÜHRERLOSEN, AUTOMATISIERTEN RAD-ROBOTER, DER VON PUNKT A NACH PUNKT B FÄHRT UND DABEI HINDERNISSEN AUSWEICHT.

Der Designprozess beginnt damit, dass ihr im Team auf eine möglichst gute Idee kommt. Habt ihr alle positiven und negativen Aspekte eurer Einfälle abgewogen, dann müsst ihr einen davon auswählen und eure Wahl begründen. An dieser Stelle ist es elementar, im Team zu arbeiten (dies ist häufig eine der größten Herausforderungen eines Projekts!). Nicht immer wird die eigene Idee ausgewählt! Ihr müsst euch mit den anderen auf die beste Lösung einigen und dann die Gründe für die Entscheidung präsentieren.

IDEENSAMMLUNG ZUR KONSTRUKTIONSAUFGABE

Diskutiert die Aufgabe. Welche Version des Rad-Roboters will eure Gruppe konstruieren?

- **Standard**: Der Rad-Roboter weicht Hindernissen aus.
- **Mittelklasse**: Zusätzlich reagiert das Fahrzeug auf Ampeln und Fußgänger.
- **Oberklasse**: Der Rad-Roboter lässt sich außerdem schlüssellos starten.
- **Luxusklasse**: Eine Geschwindigkeitsregelanlage ist ebenfalls an Bord.

Sammelt nun Ideen für das Design. Welche Funktionen wollt ihr in die Konstruktion und die Programmierung des Modells aufnehmen? Muss die Bauweise des Roboters entsprechend angepasst werden? Hier ist Platz für eure Gedanken und Konstruktionsskizzen.

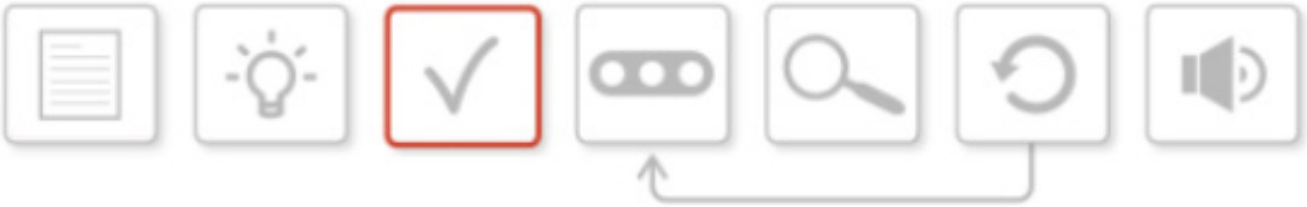
ROLLEN VERTEILEN

Jedes Projekt braucht ein Team, und ihre alle seid Teil dieses Teams.

Welche Aufgaben müssen erledigt werden?

Listet die verschiedenen Rollen auf, die ihr in eurem Team für notwendig haltet, und verteilt diese Rollen auf die Teammitglieder.

WAHL UND PRÄSENTATION DER BESTEN LÖSUNG



Nun ist es Zeit, das beste Ergebnis eurer Ideensammlung auszuwählen.

Bereitet eine kurze Präsentation vor, in der ihren eurem Lehrer und den Mitschülern erklärt, für welches Design ihr euch entschieden habt und was die Gründe dafür waren.

Zudem solltet ihr die Rollen darstellen, die den einzelnen Gruppenmitgliedern zugewiesen wurden.

Erläutert, wie ihr auf diese Rollenverteilung gekommen seid, und stellt die Stärken und Schwächen innerhalb der Gruppe heraus.

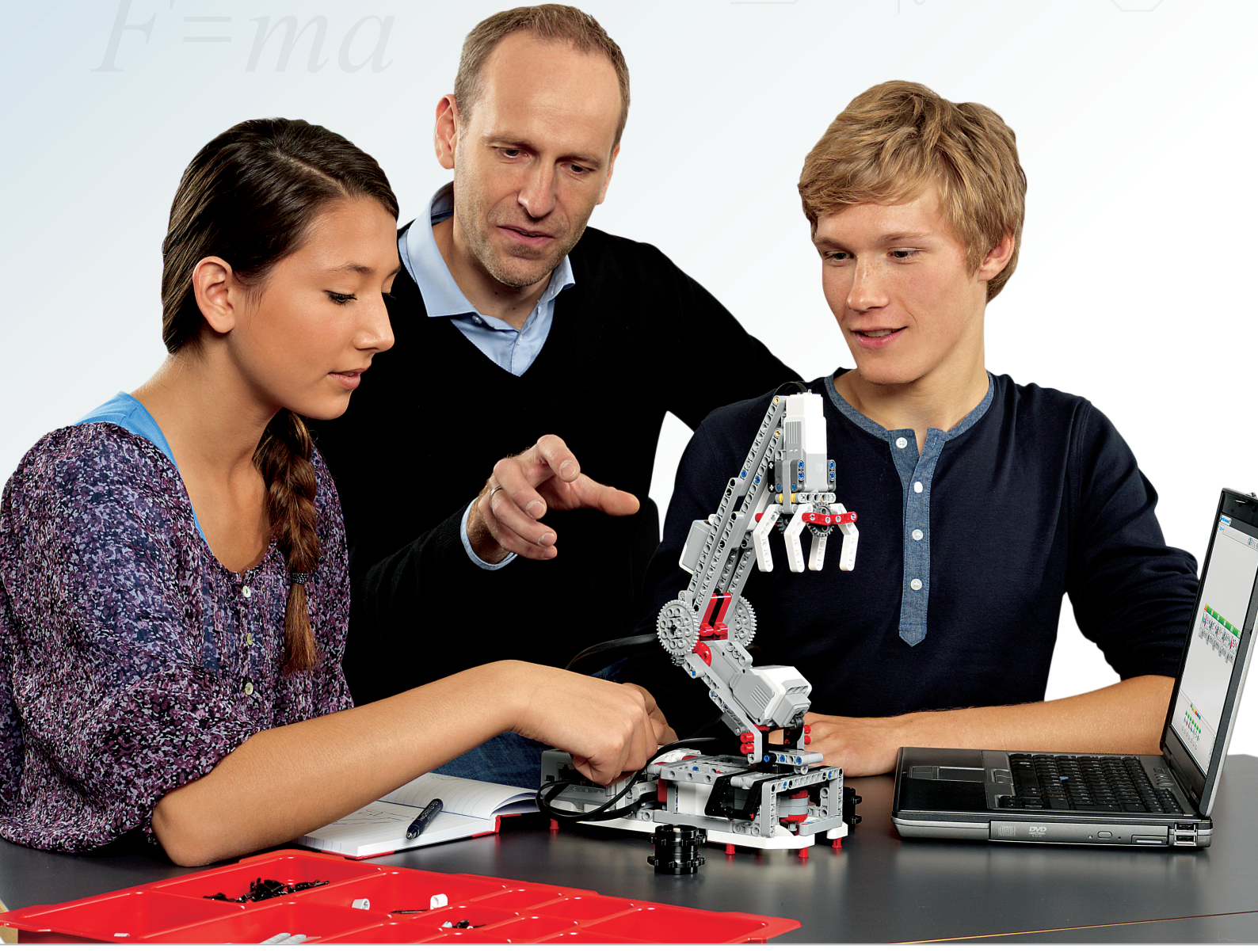
Die Präsentation sollte nicht länger als fünf Minuten dauern. Wer aus eurer Gruppe präsentiert und wie er das tut, ist allein eure Entscheidung.

Hier ist Platz für Notizen.

LEGO mindstorms education EV3

UNTERRICHTSEINHEIT 11

Bau und Programmierung eines führerlosen, automatisierten Rad-Roboters



LEGOeducation.de

LEGO education

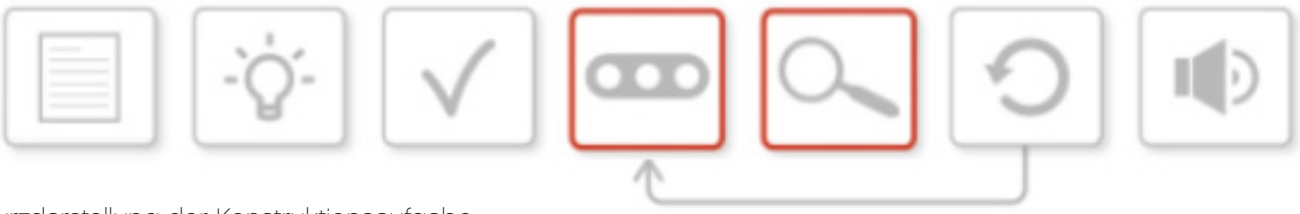
Bau und Programmierung eines führerlosen, automatisierten Rad-Roboters

Im Lauf der letzten zehn Unterrichtseinheiten haben die Schüler ihre Zeit damit verbracht, die verschiedenen Aspekte der Automatisierung eines Rad-Roboters kennenzulernen - von der Bereitstellung visueller Informationen für Passagiere und andere Verkehrsteilnehmer bis zur Befähigung des Fahrzeugs, einer festgelegten Route zu folgen. Jetzt ist es für die Schüler an der Zeit, ihr eigenes 'autonomes Auto' zu entwickeln und dabei ein Maximum der Funktionen zu integrieren, mit denen sie im Verlauf des Projekts Erfahrungen gesammelt haben.

Wie ein echtes Auto wird auch der Rad-Roboter der Schüler an seiner Ausstattung gemessen. Fahrzeughersteller bieten gewöhnlich eine breite Produktpalette an, von sehr einfachen Grundmodellen zu Autos der Oberklasse mit einer Vielfalt an Ausstattungsmerkmalen.

In diesem Projekt folgen die Schüler dem Konstruktionsprozess, der in LEGO® MINDSTORMS® Education EV3 Konstruktionsprojekte zugrunde gelegt wird - einem Unterrichtspaket, das für das Lernen in den MINT-Fächern entwickelt wurde (aber auch im Informatik-Kontext eingesetzt werden kann, wenn dort anwendungsorientierte Aktivitäten erforderlich sind).

Diese Unterrichtseinheit dreht sich um den Bau, die Programmierung und die Analyse des Rad-Roboters. Dabei ist es für die Schüler sinnvoll, sich am Konstruktionsprozess zu orientieren, wie er im EV3 Konstruktionsprojekte-Unterrichtspaket beschrieben ist. Der heutige Ablauf gestaltet sich wie folgt:



- Kurzdarstellung der Konstruktionsaufgabe
- Sammeln von Ideen
- Wählen der besten Lösung
- **Bauen und Programmieren einer Lösung**
- **Test und Analyse**
- Überprüfung und Verbesserung
- Präsentation

Diese Einheit befasst sich mit den Schritten 4 und 5. Die übrigen Punkte werden in der folgenden Unterrichtseinheit abgearbeitet.

Unterrichtseinheit 11
Stundenplanung

ERGEBNISSE

In dieser Einheit lernen die Schüler,

- durch Entwicklung, Gebrauch und Bewertung rechnergestützter Abstraktionen reale Problemstellungen und physikalische Systeme abzubilden.
- ihre Arbeit objektiv zu analysieren.
- als Team zusammenzuarbeiten.

BEGRIFFE

Entwicklung, Lösung, Kurzdarstellung, Test, Analyse.

EINFÜHRUNG

- Rekapitulieren Sie die Inhalte von Unterrichtseinheit 10 und stellen Sie sicher, dass die Schüler die nötigen Voraussetzungen besitzen, den Konstruktionsprozess dieser Unterrichtseinheit selbstbewusst anzugehen.
- Erklären Sie den Schülern, dass sie während dieser Einheit die nächsten zwei Stufen des Projekts bewältigen werden: Den Bau und die Programmierung des Rad-Roboters, sowie den Test und die Analyse, ob dieser seinen Zweck erfüllt.
- Rufen Sie den Schülern die Konstruktionsaufgabe für das Projekt in Erinnerung:

  Entwickle und baue einen führerlosen, automatisierten Rad-Roboter, der von Punkt A nach Punkt B fährt und dabei Hindernissen ausweicht.

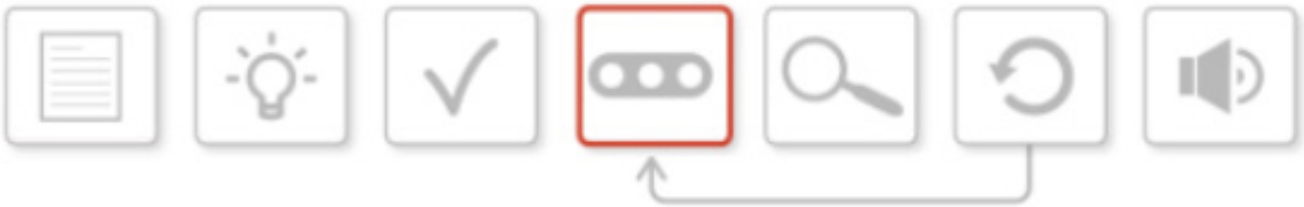
  Wenn wir die Klassifizierung von Autos heranziehen, dann wäre dies ein einfaches Standard-Modell. Die Schüler können ihrer Konstruktion jedoch weitere Funktionen hinzufügen, die die 'Klasse' ihres Rad-Roboters erhöhen. Wie Sie diese Klassen letztlich bezeichnen wollen, bleibt ihnen und ihren Schülern überlassen.

- **Standard**: Der Rad-Roboter weicht Hindernissen aus.
- **Mittelklasse**: Zusätzlich reagiert das Fahrzeug auf Ampeln und Fußgänger.
- **Oberklasse**: Der Rad-Roboter lässt sich außerdem schlüssellos starten.
- **Luxusklasse**: Zudem ist eine Geschwindigkeitsregelanlage an Bord.

Unterrichtseinheit 11
Stundenplanung

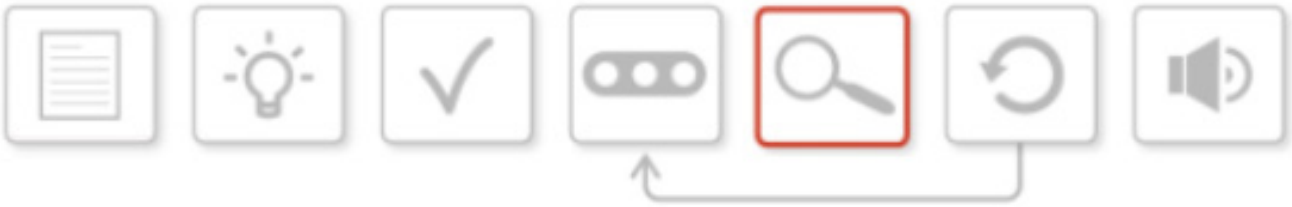
HAUPTAUFGABE

BAUEN UND PROGRAMMIEREN EINER LÖSUNG



- Die Schüler müssen auf die Arbeit aus Unterrichtseinheit 10 zurückblicken, vor allem auf das Design ihres Rad-Roboters.
- Ermuntern Sie die Schüler dazu, das Design und die von ihnen gewählte 'Klasse' des Fahrzeugs zu rekapitulieren und zu diskutieren.
- Mitunter beginnen die Schüler nun, die Aufgaben entsprechend der Rollen, die in der letzten Unterrichtseinheit festgelegt wurden, zu verteilen.
- So könnten den Schülern vorschlagen, sich in zwei Gruppen aufzuteilen - eine kleine, die den Bau des Modells übernimmt, während sich der Rest auf das Programmieren konzentriert.
- Erinnern Sie die Schüler daran, dass sie nicht erwarten können, mit ihrer ersten Konstruktion bzw. ihrem ersten Programm die Aufgabe zu voller Zufriedenheit und in Übereinstimmung mit der Konstruktionsanweisung zu lösen. Sie werden ihr Werk austesten und analysieren müssen.

TEST UND ANALYSE



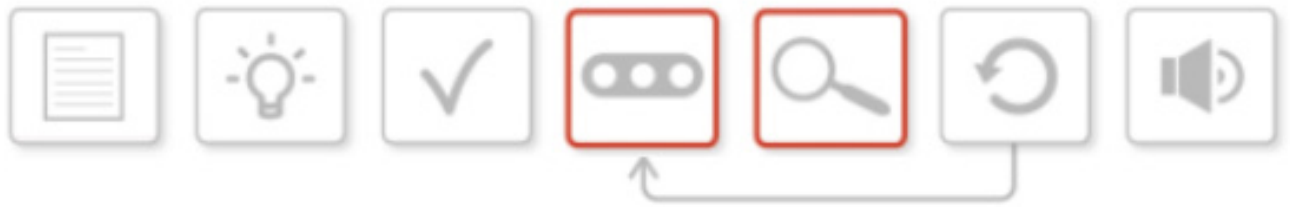
- Den Schülern muss bewusst gemacht werden, dass es sich hier um einen fortlaufenden Prozess handelt.
- Es ist wichtig, die Schüler darauf hinzuweisen, dass sie ihre Konstruktionen und Programme kontinuierlich analysieren sollten.
- Einige Schüler wollen womöglich die Konstruktion und das Programm erst fertigstellen, bevor sie mit dem Testen beginnen. Andere wählen vielleicht eine iterative Herangehensweise. Den Schülern sollte die Freiheit gelassen werden, die Methode anzuwenden, die ihnen am meisten zusagt. Dies verschafft ihnen die Möglichkeit, selbstbestimmt zu lernen.
- Erinnern Sie die Schüler während dieser Phase daran, dass sie immer wieder die Konstruktionsaufgabe und ihre eigenen Designs und Ideen heranziehen sollten - und dass diese Ideen sich im Verlauf der Einheit auch verändern können, vor allem in Unterrichtseinheit 12.

Die Schüler sollten jeden Schritt des Prozesses auf ihren Arbeitsblättern oder im Inhalts-Editor der EV3-Software schriftlich festhalten.

ABSCHLIESSENDE DISKUSSION

- Was sind die nächsten Schritte? Nachdem die Gruppen nun ihre Rad-Roboter gebaut und programmiert haben, wird sich die nächste Unterrichtseinheit auf folgende Punkte konzentrieren: Überprüfung der getanen Arbeit, falls nötig Überarbeitung von Konstruktion und Programm sowie Präsentation des Ergebnisses vor der Klasse.

AUFGABEN DIESER UNTERRICHTSEINHEIT



In dieser Einheit müsst ihr kontinuierlich die Konstruktionsaufgabe sowie die Lösungen in puncto Bau und Programmierung heranziehen, die ihr in der vorangegangenen Unterrichtseinheit ausgearbeitet habt.

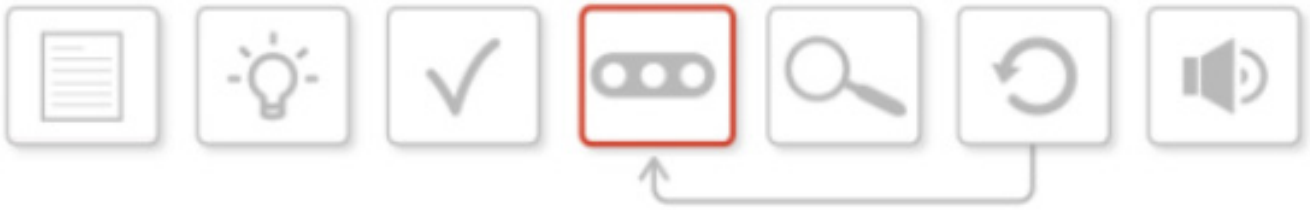
Zur Erinnerung, die Konstruktionsaufgabe lautete:

Entwickelt und baut einen führerlosen, automatisierten Rad-Roboter, der von Punkt A nach Punkt B fährt und dabei Hindernissen ausweicht.

Seht euch eure Aufzeichnungen zum Konstruktionsprozess an. Heute werdet ihr euch um Bau und Programmierung der Lösung kümmern sowie diese testen und analysieren.

Denkt daran, eure Lösungen beim Bauen und Programmieren im Inhalts-Editor der EV3-Software oder auf dem Arbeitsblatt schriftlich festzuhalten.

BAUEN UND PROGRAMMIEREN EINER LÖSUNG



Seht euch noch einmal eure Arbeit der vergangenen Unterrichtseinheit an, vor allem das Design, das ihr für euren Rad-Roboter entworfen habt.

Für welche 'Klasse' habt ihr euch entschieden (Standard, Mittelklasse, Oberklasse, Luxusklasse)?

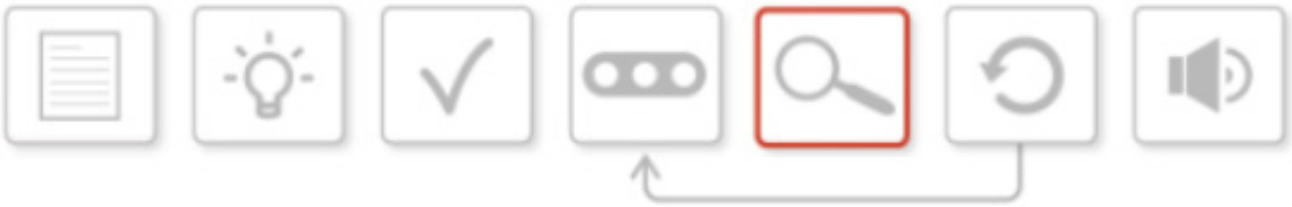
Rekapituliert eure Gedanken aus Einheit 10 und diskutiert das Design und die Folgen für die Programmierung.

Womöglich macht es Sinn, die Rollen neu zu verteilen. Verteilt Aufgaben, um die Zeit möglichst effektiv zu nutzen.

Wollt ihr die Gruppe teilen, damit einige von euch das Modell bauen, während sich andere mit der Programmierung befassen?

Wahrscheinlich werdet ihr die Konstruktionsaufgabe oder auch das eigene Design nicht beim ersten Versuch zu eurer vollen Zufriedenheit in die Tat umsetzen. Es wird nötig sein, die Konstruktion ständig auszutesten und das Programm von Fehlern zu befreien, um zu erreichen, was ihr euch vorgenommen habt.

TEST UND ANALYSE



Hier handelt es sich um einen fortlaufenden Prozess, der nicht bis zum Ende der Unterrichtseinheit aufgespart werden sollte! Testen und Analysieren sind äußerst wichtige Komponenten des Konstruktionsprozesses.

Ihr könnt die komplette Konstruktion und das Programm vor dem Testen fertigstellen. Oder ihr geht Test und Analyse schrittweise während der Bau- und Programmierarbeit an.

Am besten wäre, wenn ihr in der Gruppe diskutiert, welche Technik für euch sinnvoll ist.

Während dieser Phase solltet ihr stets die Konstruktionsaufgabe sowie eure eigenen Designs und Ideen zu Rate ziehen.

Beim Testen müsst ihr folgende Fragen beantworten:

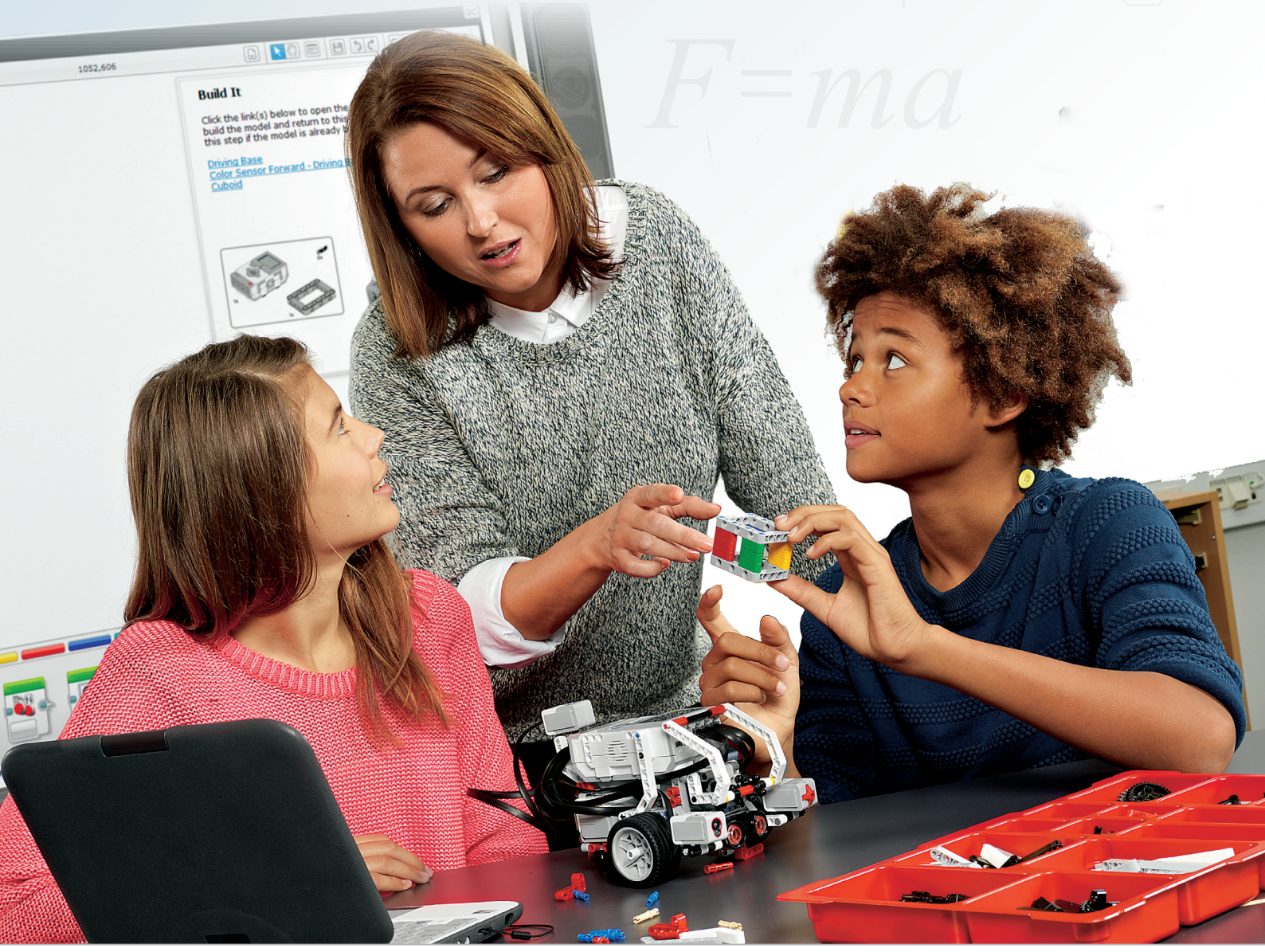
- Erfüllt der Rad-Roboter die Konstruktionsaufgabe?
- Sieht er so aus, wie ihr ihn entworfen habt?
- Tut er das, was er tun soll? Funktioniert also euer Programm?

Womöglich verändern sich eure Ideen während der Unterrichtseinheit, oder sie entwickeln sich weiter. Dies ist ein Teil des Prozesses und sollte auf dem Arbeitsblatt oder im Inhalts-Editor der EV3-Software schriftlich festgehalten werden.

mindstorms education EV3

UNTERRICHTSEINHEIT 12

Überprüfung, Verbesserung und Präsentation eines führerlosen, automatisierten Rad-Roboters



LEGOeducation.de

education

Überprüfung, Verbesserung und Präsentation eines führerlosen, automatisierten Rad-Roboters

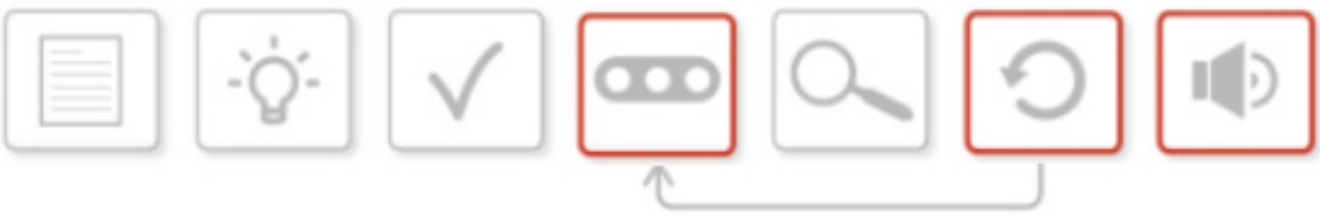
Im Lauf der letzten elf Unterrichtseinheiten haben die Schüler ihre Zeit damit verbracht, die verschiedenen Aspekte der Automatisierung eines Rad-Roboters kennenzulernen - von der Bereitstellung visueller Informationen für Passagiere und andere Verkehrsteilnehmer bis zur Befähigung des Fahrzeugs, einer festgelegten Route zu folgen. Jetzt ist es für die Schüler an der Zeit, ihr eigenes 'autonomes Auto' zu entwickeln und dabei ein Maximum der Funktionen zu integrieren, mit denen sie im Verlauf des Projekts Erfahrungen gesammelt haben.

Wie ein echtes Auto wird auch der Rad-Roboter der Schüler an seiner Ausstattung gemessen. Fahrzeughersteller bieten gewöhnlich eine breite Produktpalette an, von sehr einfachen Grundmodellen zu Autos der Oberklasse mit einer Vielfalt an Ausstattungsmerkmalen.

In diesem Projekt folgen die Schüler dem Konstruktionsprozess, der in LEGO® MINDSTORMS® Education EV3 Konstruktionsprojekte zugrunde gelegt wird - einem Unterrichtspaket, das für das Lernen in den MINT-Fächern entwickelt wurde (aber auch im Informatik-Kontext eingesetzt werden kann, wenn dort anwendungsorientierte Aktivitäten erforderlich sind).

Diese Unterrichtseinheit ist die Fortsetzung und Erweiterung von Einheit 11. In ihrem Verlauf wird von den Schülern verlangt, dass sie ihre Design- und Programmlösungen noch einmal überdenken und kritisch prüfen, was sie bislang erreicht haben. Sie müssen zusammenarbeiten, um festzustellen, was gut funktioniert hat und was verbessert werden könnte. Sie bekommen Zeit, ihre Konstruktionen und Programme zu überarbeiten und zu verfeinern. Nach dem Abschluss von Programmierung und Testen müssen sie ihre Arbeit schließlich den Mitschülern präsentieren, wobei sie sich auf den Konstruktionsprozess konzentrieren und selbst einschätzen sollten, wie sie die Konstruktionsaufgabe erfüllt haben.

Es wäre sinnvoll für die Schüler, sich am Konstruktionsprozess zu orientieren, wie er im EV3 Konstruktionsprojekte-Unterrichtspaket beschrieben ist. Der heutige Ablauf gestaltet sich wie folgt:



- Kurzdarstellung der Konstruktionsaufgabe
- Sammeln von Ideen
- Wählen der besten Lösung
- **Bauen und Programmieren einer Lösung**
- Test und Analyse
- **Überprüfung und Verbesserung**
- **Präsentation**

Diese Einheit befasst sich mit den Schritten 4, 6 und 7.

ERGEBNISSE

In dieser Einheit lernen die Schüler,

- durch Entwicklung, Gebrauch und Bewertung rechnergestützter Abstraktionen reale Problemstellungen und physikalische Systeme abzubilden.
- ihre Arbeit objektiv zu analysieren.
- ihre Arbeit kritisch zu hinterfragen und gemeinsam zu verbessern.
- ihren Weg durch den Konstruktionsprozess den Mitschülern zu präsentieren.
- als Team zusammenzuarbeiten.

BEGRIFFE

Design, Lösung, Konstruktionsaufgabe, Test, Analyse, Überprüfung, Überarbeitung.

EINFÜHRUNG

- Rekapitulieren Sie die Inhalte von Unterrichtseinheit 11 und stellen Sie sicher, dass die Schüler die nötigen Voraussetzungen besitzen, den Konstruktionsprozess dieser Unterrichtseinheit selbstbewusst anzugehen.
- Erklären Sie den Schülern, dass sie diese Einheit damit beginnen werden, ihren Rad-Roboter bzw. das Programm laufen zu lassen. Anschließend besteht ihre Aufgabe darin, bislang Erreichtes zu überprüfen und entsprechende Folgerungen in die Veränderung und Verbesserung ihrer Konstruktion bzw. ihres Programms einfließen zu lassen.
- Betonen Sie, dass diese Veränderungen den Rad-Roboter der Schüler in eine höhere 'Fahrzeug-Klasse' hieven könnten.
- Sagen Sie den Schülern, dass sie sämtliche Änderungen an Konstruktion und Programm auf ihren Arbeitsblättern oder im Inhalts-Editor der EV3-Software festhalten sollen. Dies wird bei der finalen Präsentation der Ergebnisse hilfreich sein.

- Rufen Sie den Schülern die Konstruktionsaufgabe für das Projekt in Erinnerung:

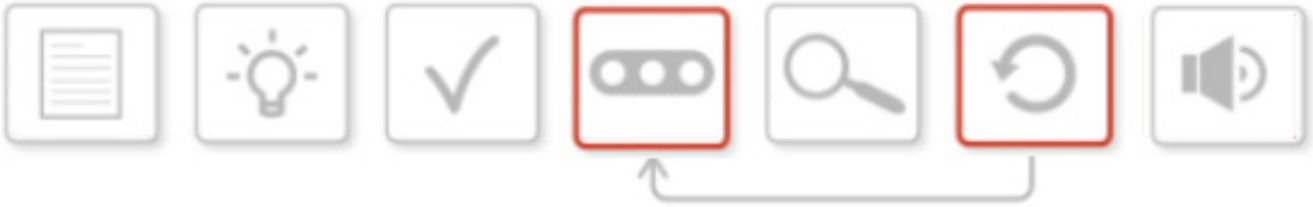
  Entwickle und baue einen führerlosen, automatisierten Rad-Roboter, der von Punkt A nach Punkt B fährt und dabei Hindernissen ausweicht.

  Wenn wir die Klassifizierung von Autos heranziehen, dann wäre dies ein einfaches Standard-Modell. Die Schüler können ihrer Konstruktion jedoch weitere Funktionen hinzufügen, die die 'Klasse' ihres Rad-Roboters erhöhen. Wie Sie diese Klassen letztlich bezeichnen wollen, bleibt ihnen und ihren Schülern überlassen.

- **Standard**: Der Rad-Roboter weicht Hindernissen aus.
- **Mittelklasse**: Zusätzlich reagiert das Fahrzeug auf Ampeln und Fußgänger.
- **Oberklasse**: Der Rad-Roboter lässt sich außerdem schlüssellos starten.
- **Luxusklasse**: Zudem ist eine Geschwindigkeitsregelanlage an Bord.

HAUPTAUFGABE 1

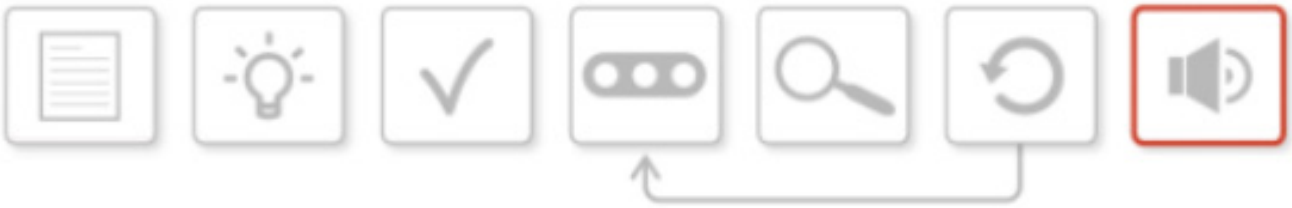
ÜBERPRÜFEN UND VERBESSERN/BAUEN UND PROGRAMMIEREN EINER LÖSUNG



- Sagen Sie den Schülern, dass sie während dieser Einheit ihren Rad-Roboter ständig überprüfen, testen und (womöglich) neu konstruieren werden.
- Sie könnten (auch gemeinsam mit der Klasse) Erfolgskriterien ausarbeiten, auf die sich die Schüler beziehen können.
- Es ist wichtig, die Schüler an diesem Punkt auf die ursprüngliche Konstruktionsaufgabe aufmerksam zu machen.
- Die Schüler lassen in ihren Teams die Programme ablaufen und diskutieren über Möglichkeiten, das Verhalten ihres Rad-Roboters zu verbessern, und wie diese Verbesserungen mit der Konstruktionsaufgabe zusammenpassen.
- Die Schüler werden zusätzliche Programmier-Optionen erkunden und mitunter ihre Konstruktion verändern müssen.
- Betonen Sie, dass die Schüler mit der kontinuierlichen Überprüfung und Überarbeitung des Designs ihren Rad-Roboter verbessern und in eine höhere 'Klasse' bringen könnten.
- Gewähren Sie den Gruppen zum Ende der Einheit genug Zeit, um ihre Programme ein letztes Mal zu testen. Bringen Sie hierzu alle Schüler zusammen. Während die Roboter fahren, sollten Sie und die Schüler die festgelegten Erfolgskriterien überprüfen.

HAUPTAUFGABE 2

PRÄSENTATION

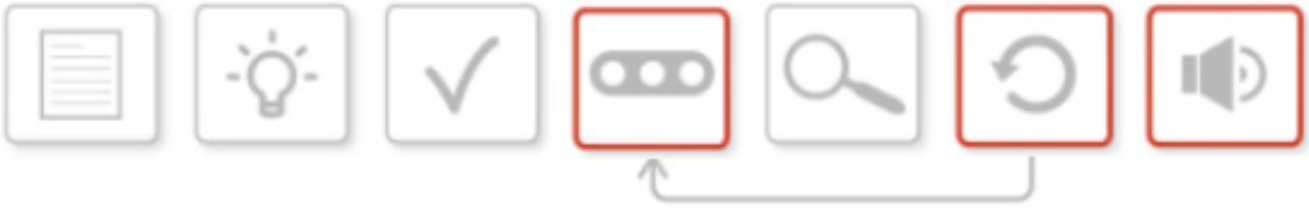


- Nachdem die Schüler nun den gesamten Konstruktionsprozess durchschritten und ihre endgültige Version des Rad-Roboters gebaut und programmiert haben, setzt die Präsentation ihrer Arbeit vor der Klasse den Schlusspunkt.
- Die Schüler sollten seit der Unterrichtseinheit 10 ihre Vorgehensweise fixiert haben, entweder auf den Arbeitsblättern oder im Inhalts-Editor der EV3-Software. Neben schriftlichen Notizen kann dies in Form von Videos, Fotos oder Bildschirmfotos ihrer Programme geschehen sein.
- Sagen Sie den Schülern, dass sie eine kurze Präsentation vorbereiten sollen, die ihren Weg durch den Konstruktionsprozess sowie ihre Entscheidungen auf diesem Weg erläutern soll. Erfolge und Misserfolge sollten herausgestellt werden, zudem sollte die Klasse angesprochen werden, die der jeweilige Roboter nach Meinung der Schüler erreicht. Auch wie genau die Konstruktionsaufgabe erfüllt wurde, sollte diskutiert werden.
- Die Präsentationen können in einem beliebigen Format zusammengestellt werden. Der Gebrauch von Präsentations-Software wie PowerPoint, Keynote oder Prezi ist ebenso möglich wie der Einsatz des EV3-Software-Inhalts-Editors.

ABSCHLIESSENDE DISKUSSION

- Die Präsentationen der Schüler setzen den Schlusspunkt der letzten drei Unterrichtseinheiten. Durch die Vorstellung ihrer Arbeit und die eigene Bewertung, wie genau die Konstruktionsaufgabe erfüllt wurde, beurteilen sich die Schüler selbst.
- Nach jeder Präsentation ist es wichtig für die jeweilige Gruppe, eine Rückmeldung von Mitschülern und Lehrer zu bekommen.

ARBEITEN DIESER UNTERRICHTSEINHEIT



Dies ist die letzte von drei Unterrichtseinheiten, in denen ihr an eurem automatisierten Rad-Roboter arbeitet. Erneut müsst ihr kontinuierlich die Konstruktionsaufgabe sowie eure Lösungen in puncto Konstruktion und Programmierung heranziehen, die ihr in den vorangegangenen Einheiten erarbeitet habt.

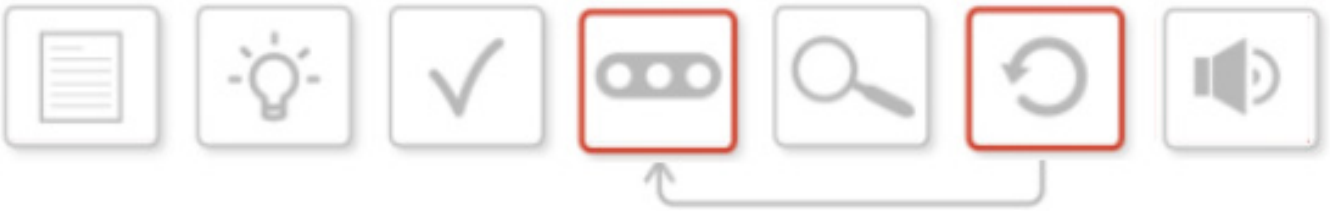
Zur Erinnerung, die Konstruktionsaufgabe lautete:

Entwickelt und baut einen führerlosen, automatisierten Rad-Roboter, der von Punkt A nach Punkt B fährt und dabei Hindernissen ausweicht.

Seht euch eure Aufzeichnungen zum Konstruktionsprozess an. Heute werdet ihr euch auf **die Überprüfung, Überarbeitung (also zusätzliches Bauen und Programmieren) und die Präsentation** eurer Arbeit konzentrieren.

Denkt daran, eure Arbeit im Inhalts-Editor der EV3-Software oder auf dem Arbeitsblatt schriftlich festzuhalten.

ÜBERPRÜFEN UND VERBESSERN/BAUEN UND PROGRAMMIEREN EINER LÖSUNG



Während dieser Aufgabe müsst ihr euren Rad-Roboter kontinuierlich überprüfen und (höchstwahrscheinlich) überarbeiten.

Mitunter habt ihr, eure Mitschüler oder euer Lehrer Erfolgskriterien für diese Aufgabe aufgestellt. Ist das der Fall, müsst ihr diese stetig in eure Arbeit miteinbeziehen.

Welche 'Klasse' sollte euer Rad-Roboter von Beginn an haben? Es ist gut möglich, dass ihr in der Verbesserungsphase euren Rad-Roboter so umdesignt und -programmiert, dass er eine höhere Klasse bekommt.

Lasst das Programm eures Roboters laufen. Erfüllt es die Vorgaben der Konstruktionsaufgabe? Kann es verbessert werden, indem ihr den Rad-Roboter neu programmiert oder neu konstruiert (oder beides)? Erkundet weiterführende Programmier-Optionen und ändert eure Konstruktion wenn nötig.

Gegen Ende dieser Einheit werdet ihr den Rad-Roboter dem Rest der Klasse demonstrieren und ihn an der Konstruktionsaufgabe und den Erfolgskriterien messen lassen müssen.

Hier oder im Inhalts-Editor ist Platz für Notizen:

PRÄSENTATION

Nachdem ihr soweit gekommen seid, den gesamten Konstruktionsprozess durchlaufen und eure endgültige Version des Rad-Roboters gebaut und programmiert habt, gibt es nun eine letzte Aufgabe: Euer Werk soll den anderen Teams präsentiert werden.

Ihr solltet seit der Unterrichtseinheit 10 Notizen über eure Tätigkeiten gemacht haben, entweder hier auf diesen Arbeitsblättern oder im Inhalts-Editor der EV3-Software.

Diese Notizen können schriftliche Anmerkungen sein, Videos, Fotos und Bildschirmfotos eurer Programmlösungen.

Ihr sollt nun eine kurze Präsentation anfertigen, die euren Weg durch den Konstruktionsprozess darstellt und die Entscheidungen erläutert, die ihr als Gruppe beim Design eures endgültigen Rad-Roboters und des zugehörigen Programms getroffen habt.

Die Präsentation sollte auch zeigen, wie ihr die Vorgaben der Konstruktionsaufgabe erreicht habt und aus welcher 'Fahrzeug-Klasse' euer Rad-Roboter kommt.

Ihr solltet Erfolge und Rückschläge herausstellen - und auch, wie ihr die Rückschläge überwunden habt.

Das Format der Präsentation dürft ihr selbst wählen. Ihr könnt zum Beispiel eine Präsentations-Software wie PowerPoint, Keynote oder Prezi oder auch den Inhalts-Editor der EV3-Software verwenden.