Problemstellungskarte Bereich: Gehirn

Problemstellung

Für die Lagerung und Logistik der Gärtnerei *Spichtig* waren bis anhin zwei Personen angestellt, die den Lagerbestand an Düngemittel und Blumenerde überwacht haben. Sobald die Materialien knapp wurden, haben sie nachbestellt. Leider kam es ab und zu vor, dass diese Nachbestellungen vergessen gingen oder nicht erfasste Blumen oder Gartenwerkzeuge in der Lagerhalle auftauchten.

Isabelle und Martin Spichtig, die beiden Geschäftsführer und Inhaber der Gärtnerei *Spichtig,* möchten in Zukunft nicht mehr im Ungewissen sitzen und auf keinen Fall Materialbestellungen versäumen – denn Zeit ist schliesslich Geld. Ihre beiden Kinder, Daniel und Jennifer, haben sie deshalb zur Installation eines *Ziva-Lagerungsroboters* überredet.

Dieser Lagerungsroboter soll den **Lagerbestand im Überblick haben** – *Wie viele Säcke an Düngemittel und Blumenerde sind gerade im Lagerbestand?* – Unterschreitet der Bestand eine bestimmte Menge, soll er ausserdem neues Material automatisch nachbestellen – *Wir brauchen unbedingt neue Säcke mit Blumenerde, wenn nur noch drei Stück in der Lagerhalle liegen.*

Modellierung: Da wir keine Säcke voll Blumenerde haben, müssen wir uns anderweitig zu helfen wissen. Für unseren Zweck soll ein Drucksensor ausreichen. Ein Drücken auf den Sensor soll das Holen eines einzelnen Sacks voller Blumenerde simulieren.

Leitfragen

Durch die Beantwortung der folgenden Leitfragen wird die Installation des Ziva-Lagerungsroboters für euch einfacher:

- Wie kann erreicht werden, dass unser Roboter den Bestand an Düngemittel auf seinem Display anzeigt?
- Wie kann erreicht werden, dass die Anzahl an Düngemittel um einen beliebigen Wert abnimmt, sobald der Drucksensor betätigt wird?
- Wie kann erreicht werden, dass unser Roboter eine Nachbestellung tätigt (der Ausgabewert auf dem Display wieder erhöht wird), wenn der Bestand an Düngemittel einen bestimmten Mindestwert erreicht hat?

Hypothesen

1. Ideen festhalten

Es ist nun an der Zeit, euren Ideen freien Lauf zu lassen. Was denkt ihr? Wie könnte wohl eine mögliche Lösung für das vorangehende Problem aussehen? Überlegt euch mögliche Szenarien, wie ihr das Problem lösen könntet und haltet eure Ideen im Forscherbuch fest.

2. Modellzeichnungen

Zeichnet euch skizzenhaft eine Modellzeichnung eines möglichen Lösungsprogramms und erläutert schriftlich die Bedeutung der einzelnen Bausteine. Hierzu stehen euch als Hilfestellung die untenstehenden Sprachbausteine zur Verfügung.

Sprachbausteine

Start, Schalter, Schleife, Variable, Rechnen (Vergleichen), Anzeige (EV3), Warten-Auf

Bausteine

Bevor ihr das Problem der Gärtnerei *Spichtig* lösen könnt, lernt ihr in den drei Baustein-Posten die Funktionen und Einsatzmöglichkeiten von Schaltern, Schlaufen, Textausgaben und Variablen kennen. Dieses Wissen hilft euch bei der Problemlösung des Ziva-Lagerungsroboters.

- Baustein 1: Die Schleife
- Baustein 2: Der Schalter
- Baustein 3: Textausgabe & Variablen

Baustein 1: Die Schleife

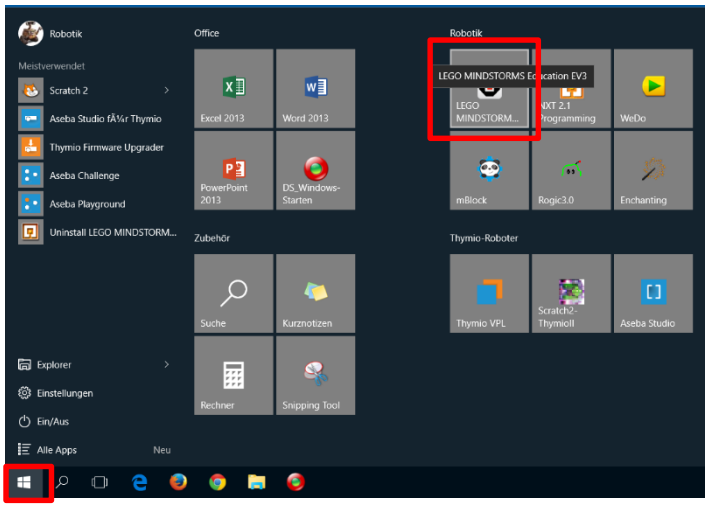
Ziele

Ich weiss, was eine Schleife ist und wie diese programmiert wird. Ich kann einen Roboter so programmieren, dass er ein Geräusch beliebig oft wiederholt.

Material

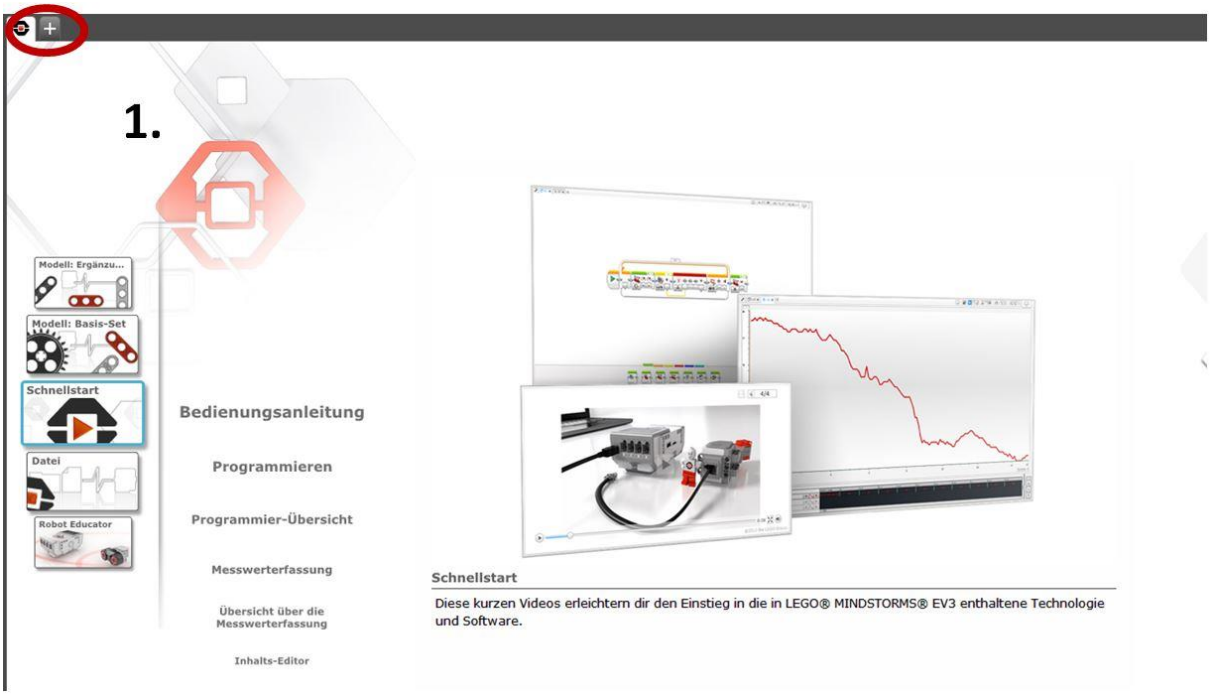
- 1 EV3 Lego Mindstorms Roboter
- 1 Notebook mit der Software LEGO Mindstorms EV3
- 1 USB Verbindungskabel

Hinweis zum Starten der Software LEGO Mindstorms EV3 starten:

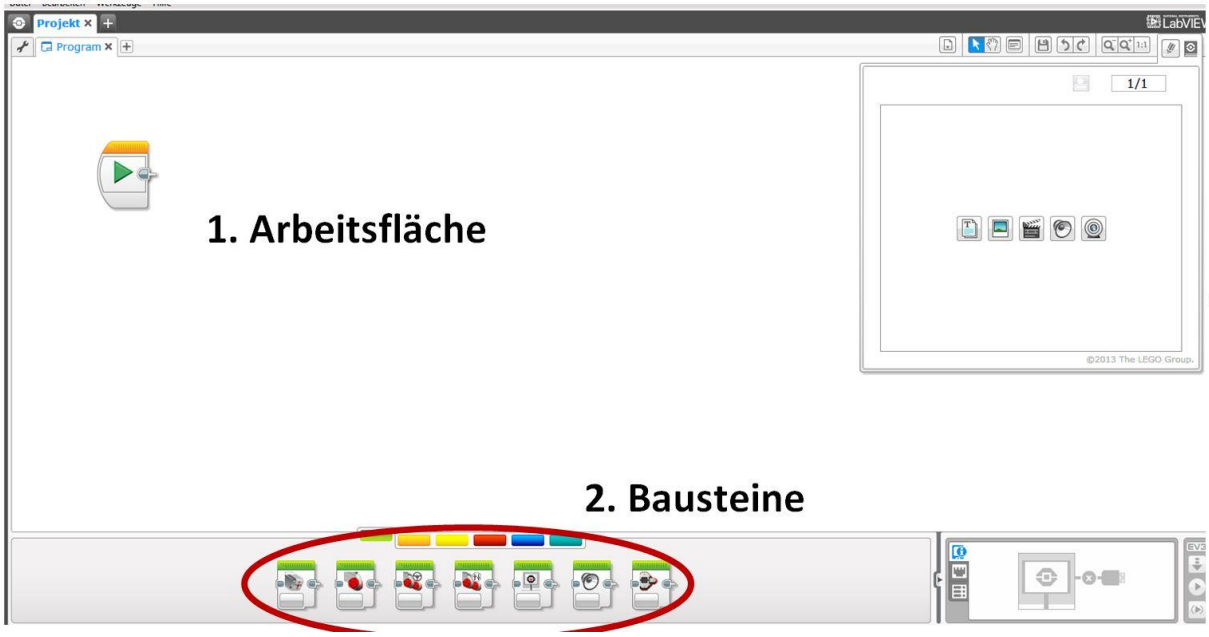


Vorgehen

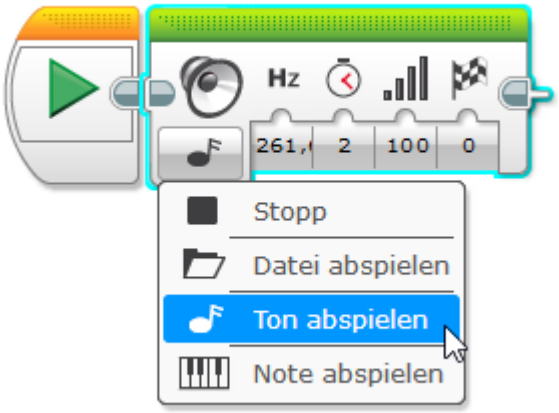
Um einen Roboter zu programmieren, braucht ihr ein entsprechendes Programm. Öffnet die Software „LEGO MINDSTORMS EV3“ und erstellt ein neues Projekt. Die Abbildung hilft euch dabei.



Per „Drag & Drop“ könnt ihr Bausteine in die Arbeitsfläche ziehen. Indem ihr die verschiedenen Bausteine aneinanderreiht, schreibt ihr ein Programm. Dieses Programm kann vom Roboter anschliessend ausgeführt werden.



Baut das folgende Programm nach:



Spielt das Programm nun ab. Wenn ihr alles richtig gemacht habt, sollte ein lauter Ton erklingen.

Auftrag 1: Töne abspielen
Versucht ein Programm zu schreiben, bei dem der Roboter drei verschiedene Töne von sich gibt.

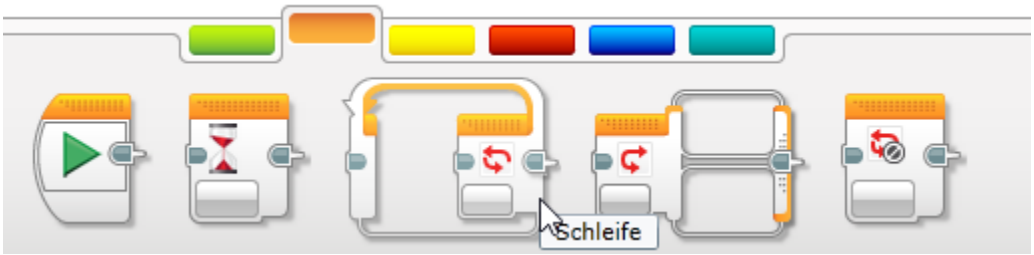
Hinweis: Eine mögliche Lösung dazu findet ihr unter den Musterlösungen.

Auftrag 2: Töne wiederholen
Schreibt nun ein Programm, welches die drei Töne zwei Mal wiederholt. Es sollen insgesamt also sechs Töne erklingen.

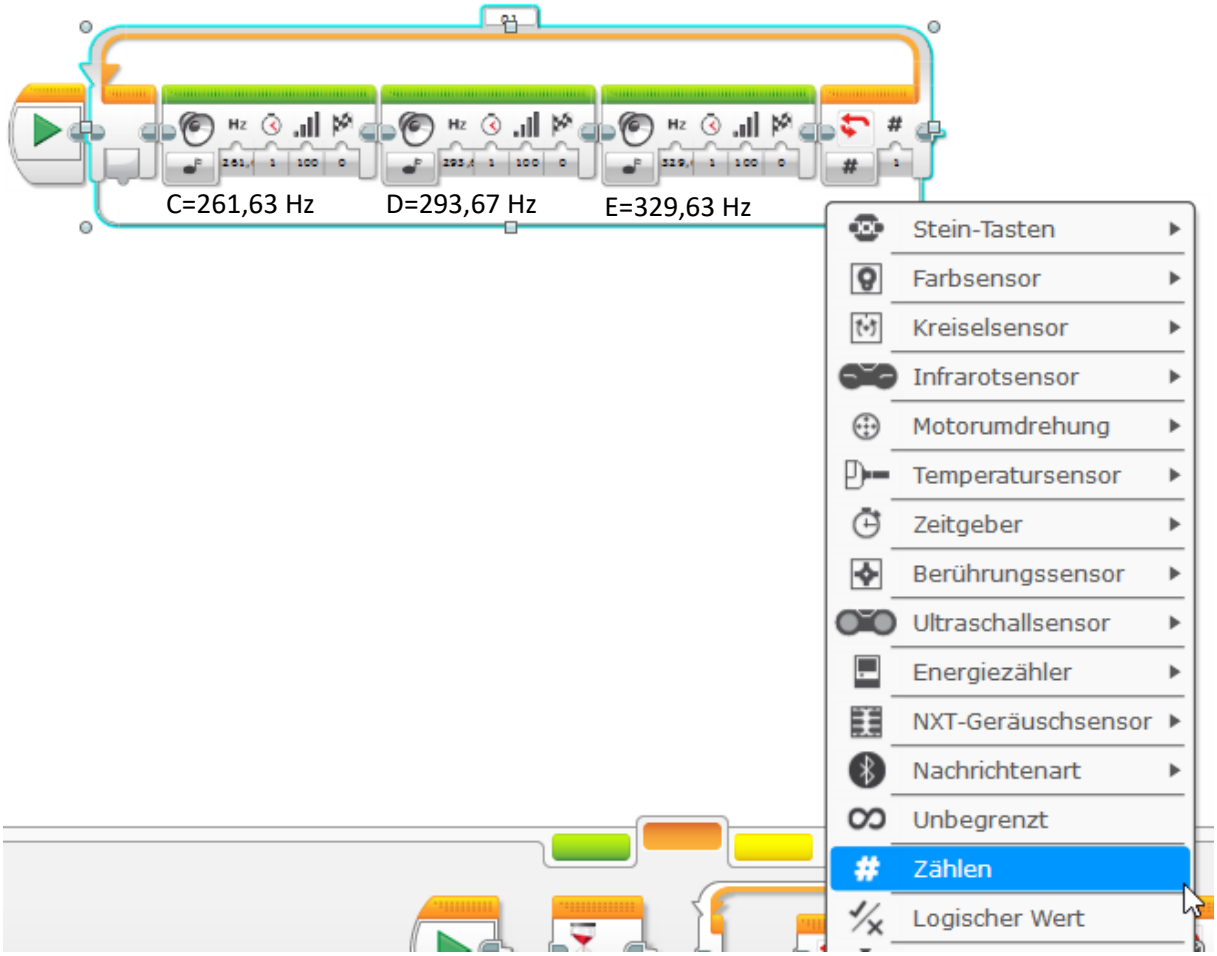
Hinweis: Eine mögliche Lösung dazu findet ihr unter den Musterlösungen.

Ihr habt sicher gemerkt, dass das Programm länger geworden ist. Bei mehreren Wiederholungen wird auch das Programm länger. Und spätestens bei unendlich vielen Wiederholungen haben wir ein echtes Problem: Denn wie sollen wir unendlich solcher Bausteine aneinanderreihen? Die Programmierer haben dafür eine einfache Antwort gefunden: **Die Schleife!**

Den Schleifen-Baustein findet ihr bei den orangen Bausteinen. Mit „Drag &Drop“ könnt ihr ihn auf die Arbeitsfläche ziehen. In die Schleife können andere Bausteine eingefügt werden. Alles was in der Schleife ist, wird anschliessend wiederholt.



Baut nun folgendes Programm nach:



Lasst den Roboter das Programm nun abspielen. Was passiert? Irgendwie nicht viel! Der Roboter spielt die bekannten drei Töne ab, danach ist fertig. Wir müssen dem Roboter nämlich noch sagen, wie viele Male er die drei Töne wiederholen soll.

Auftrag 3: Wiederholungen
Testet was passiert, wenn wir die Zahl bei der Schleife auf 3 ändern.

Hinweis: Eine mögliche Lösung dazu findet ihr unter den Musterlösungen.

Auftrag 4: Unterschiedliche Wiederholungen

Findet heraus, wie das Programm angepasst werden muss, dass die drei Töne unendlich oft wiederholt werden.

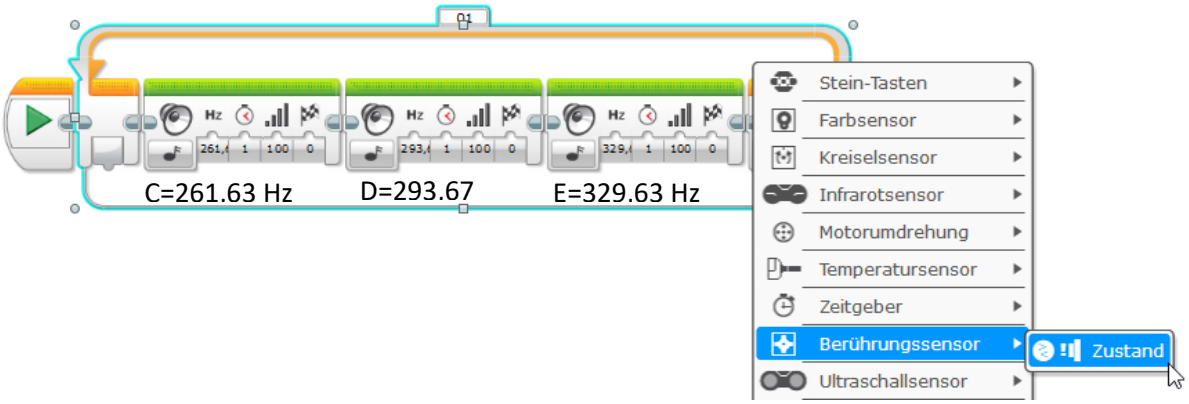
Hinweis: Eine mögliche Lösung dazu findet ihr unter den Musterlösungen.

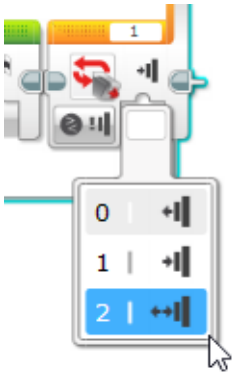
Ihr seht also: Mit einer Schleife kann ganz viel Platz gespart werden, und das Programm wird übersichtlicher. Drei Wiederholungen und sieben Wiederholungen sind genau gleich lang. Und auch unendlich viele Wiederholungen sind nun möglich.

Als letztes schaut ihr euch nun an, wie der Roboter die drei Töne so lange wiederholt, bis ein Abstellknopf gedrückt wird.

Anstatt den Roboter eine bestimmte Anzahl oder unendlich viele Wiederholungen durchführen zu lassen, soll sich das Programm nun nur so lange wiederholen, bis der Berührungssensor gedrückt wird.

Baut dazu folgendes Programm nach:





Achtet unbedingt darauf, dass ihr beim Programm den gleichen Anschluss (1) wählt, wie beim Roboter.

Auftrag 5: Wiederholung beenden

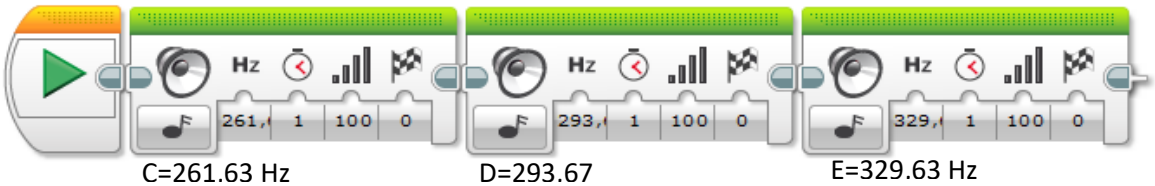
Baut ein Programm mit Tönen, das sich nur so lange wiederholt, bis der Berührungssensor gedrückt wird.

Hinweis: Eine mögliche Lösung dazu findet ihr unter den Musterlösungen.

Musterlösungen

Bausteinposten 1: Die Schleife

Auftrag 1: Töne abspielen



C=261.63 Hz D=293.67 E=329.63 Hz

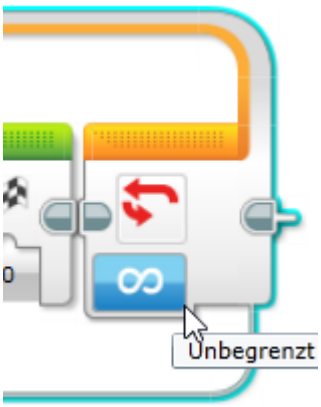
Auftrag 2: Töne wiederholen

C=261,63 Hz D=293,67 E=329,63 Hz C=261.63 Hz D=293.67 E=329.63 Hz

Auftrag 3: Wiederholungen

Die drei Töne werden drei Mal wiederholt. Ändert ihr den Wert auf sieben, werden die drei Töne sieben Mal wiederholt usw.

Auftrag 4: Unterschiedliche Wiederholungen



Auftrag 5: Wiederholung beenden

Der Roboter wiederholt solange die drei Töne, bis der Drucksensor (Schalter) einmal aktiviert wird. Da immer alle drei Töne in der Schlaufe wiederholt werden, stoppt der Roboter immer beim gleichen Ton!

Baustein 2: Der Schalter

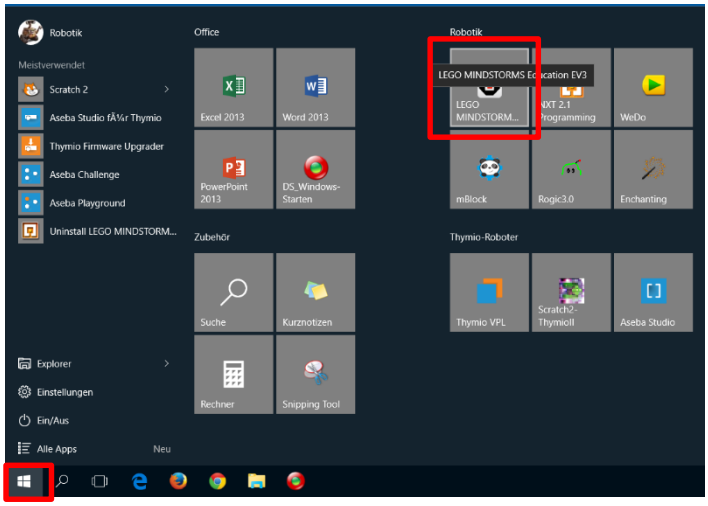
Ziele

Ich weiss, was ein (verschachtelter) Schalter ist und wie dieser programmiert wird. Ich kann einen Roboter so programmieren, dass er beim Betätigen von unterschiedlichen Sensoren verschiedene Geräusche abspielt.

Material

- 1 EV3 Lego Mindstorms Roboter
- 1 Notebook mit der Software LEGO Mindstorms EV3
- 1 USB Verbindungskabel

Hinweis zum Starten der Software LEGO Mindstorms EV3 starten:



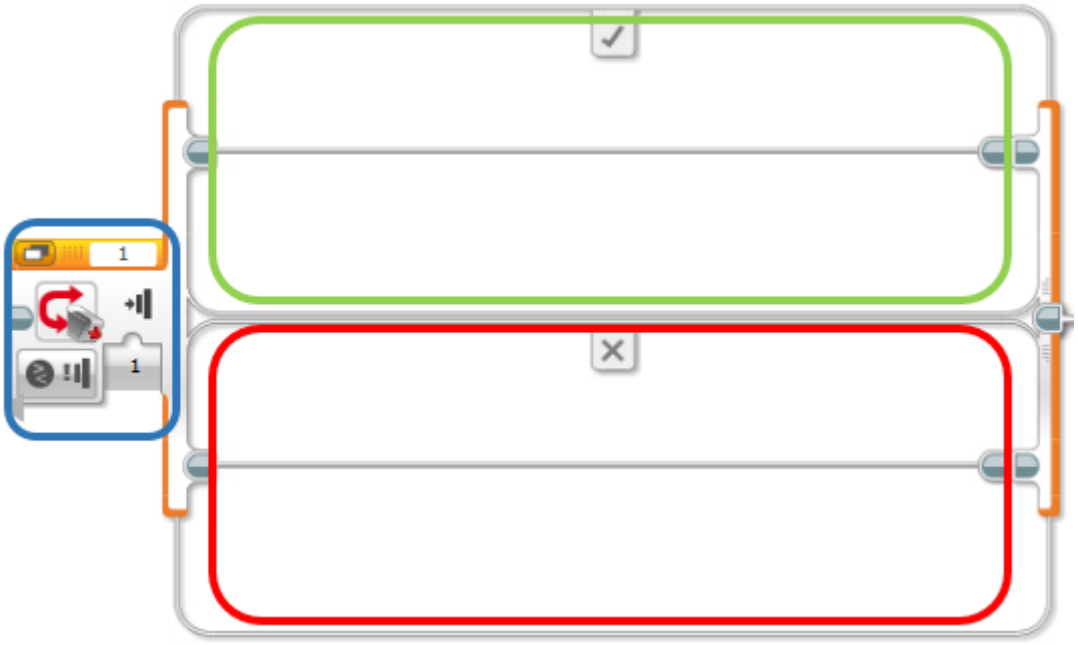
Vorgehen

Ihr könnt euch einen Schalter als eine Art Weiche vorstellen.

Ist die Weiche nach rechts gestellt, fährt der Zug nach rechts weiter. Ist sie nach links gestellt, fährt der Zug nach links weiter.

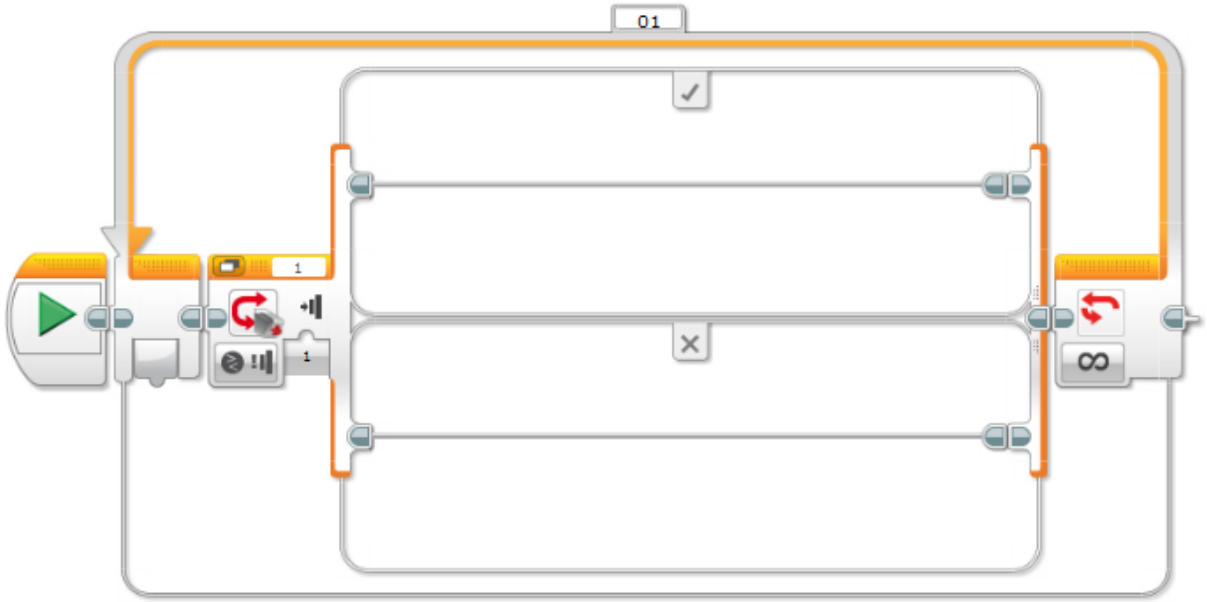
Ganz ähnlich funktioniert dies auch beim Programmieren: Kommt das Programm nämlich an eine Weiche (bzw. Schalter), fährt es auch so weiter wie die Weiche (bzw. der Schalter) gestellt ist.

Der Schalter wird in diesem Beispiel mit einem Drucksensor dargestellt.
Und so sieht ein Schalter-Block aus:



	Im blauen Bereich könnt ihr festlegen, wie der Schalter „gestellt“ werden soll. Im obigen Beispiel handelt es sich um einen Drucksensor. Solange der Sensor gedrückt wird, soll also der Schalter umgelegt bleiben. Tipp: Achtet darauf, dass der Anschluss des Sensors im Programm und am Roboter übereinstimmt.
	Im grünen Bereich könnt ihr andere Blöcke einfügen. Diese Blöcke werden vom Programm durchgeführt, solange der Schalter (in diesem Fall der Drucksensor) **gedrückt** ist.
	Auch im roten Bereich könnt ihr andere Blöcke einfügen. Diese Blöcke werden vom Programm durchgeführt, wenn der Schalter (in diesem Fall der Drucksensor) **nicht gedrückt** ist.

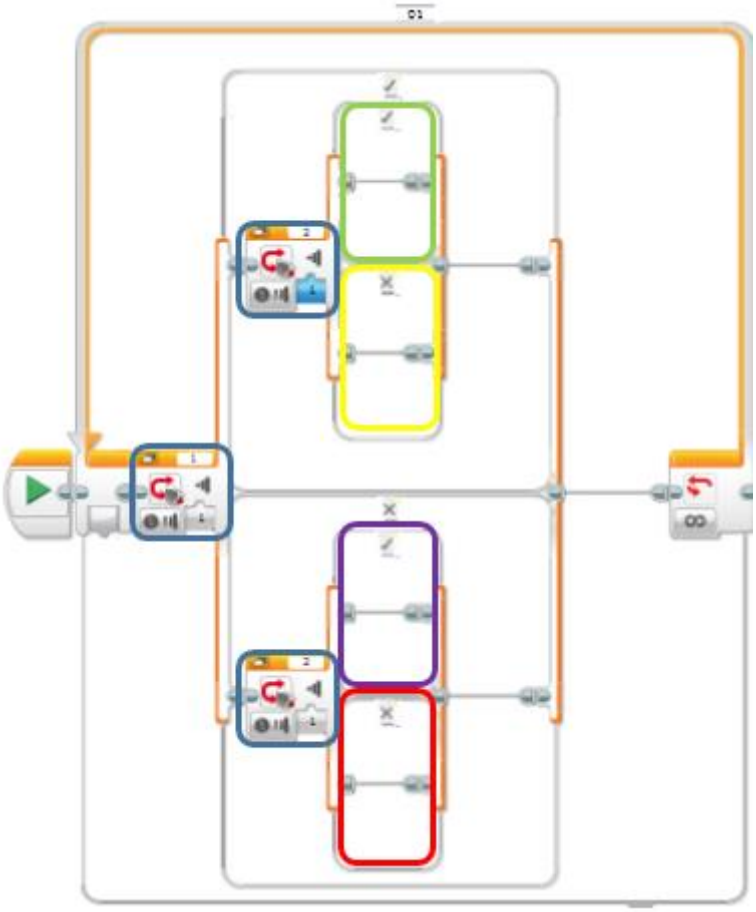
Aber Achtung: Das vorangehende Programm ist extrem schnell. Wenn es gestartet wird, hat es schon kontrolliert, ob der Schalter gedrückt ist oder nicht. Noch bevor wir überhaupt dazukommen, den Drucksensor zu betätigen. Wir müssen den Schalter deshalb meist in eine Schleife einfügen. So kontrolliert das Programm immer wieder, ob der Schalter gedrückt ist oder eben nicht.
Dies sieht dann ungefähr so aus:



Auftrag 1: Schalter:
Programmiert euren Roboter so, dass er, so lange ihr auf den Drucksensor drückt, einen beliebigen Ton von sich gibt. Sobald ihr den Sensor nicht mehr drückt, soll er still sein.

Hinweis: Eine mögliche Lösung dazu findet ihr unter den Musterlösungen.

Arbeiten mit zwei Drucksensoren: Für die nächste Aufgabe müssen wir das Prinzip des **verschachtelten Schalters** verstehen lernen. Ein verschachtelter Schalter sieht folgendermassen aus:



In den **blauen Bereichen** werden wiederum die Berührungssensoren definiert. Beachtet, dass der eine Sensor den Anschluss 1 hat, die anderen zwei den Anschluss 2. Dies muss so sein, immerhin haben wir ja zwei verschiedene Sensoren.
Um das Beispiel mit dem Zug fortzuführen: Der Zug kommt zu einer Weiche. Ist sie nach links geschaltet, fährt er links, ansonsten nach rechts. Dieser Vorgang wiederholt sich bei jeder Weiche.

Auftrag 2: Aufbau eines verschachtelten Schalters
Überlegt euch, welche Sensoren im **grünen**, im **gelben**, im **violetten** und im **roten** Bereich gedrückt sein müssen, damit genau diese Programmblöcke ausgeführt werden. Muss der erste oder der zweite Sensor betätigt werden? Oder doch beide? Oder gar keiner?

Hinweis: Eine mögliche Lösung dazu findet ihr unter den Musterlösungen.

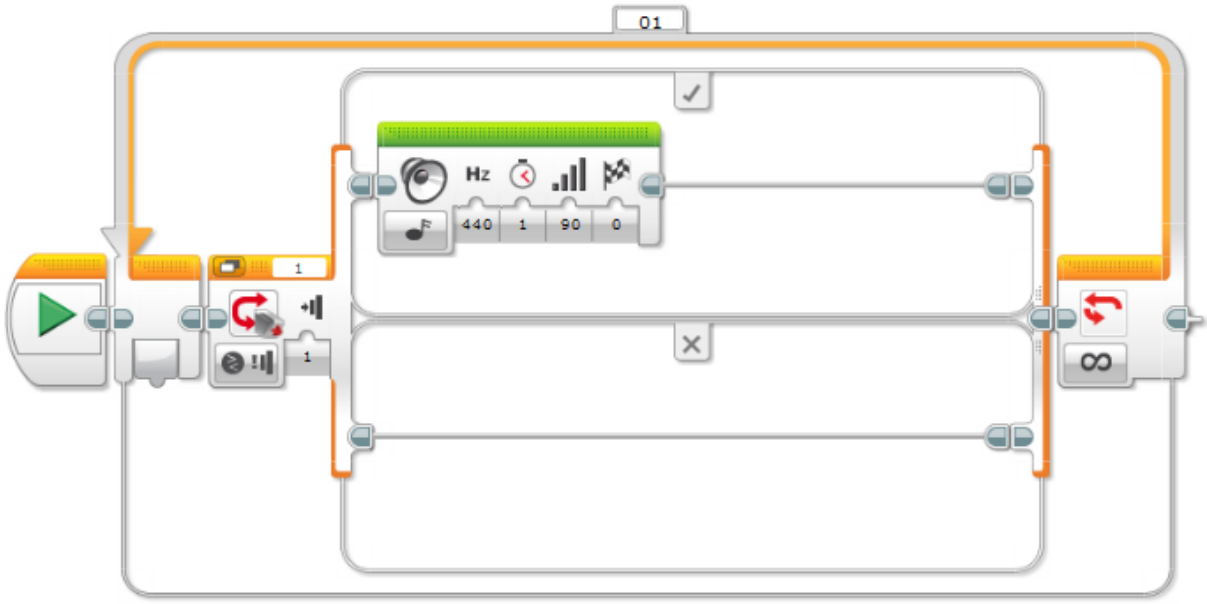
Auftrag 3: Der verschachtelte Schalter
Baut ein Programm mit einem verschachtelten Schalter, dass Folgendes passiert:
a. Wird Sensor 1 gedrückt, wird Ton C gespielt.
b. Wird Sensor 2 gedrückt, wird Ton D gespielt.
c. Werden beide Sensoren gedrückt, wird Ton E gespielt.
d. Wir kein Sensor gedrückt, wird kein Ton gespielt.

Hinweis: Eine mögliche Lösung dazu findet ihr unter den Musterlösungen.

Musterlösungen

Bausteinposten 2: Der Schalter

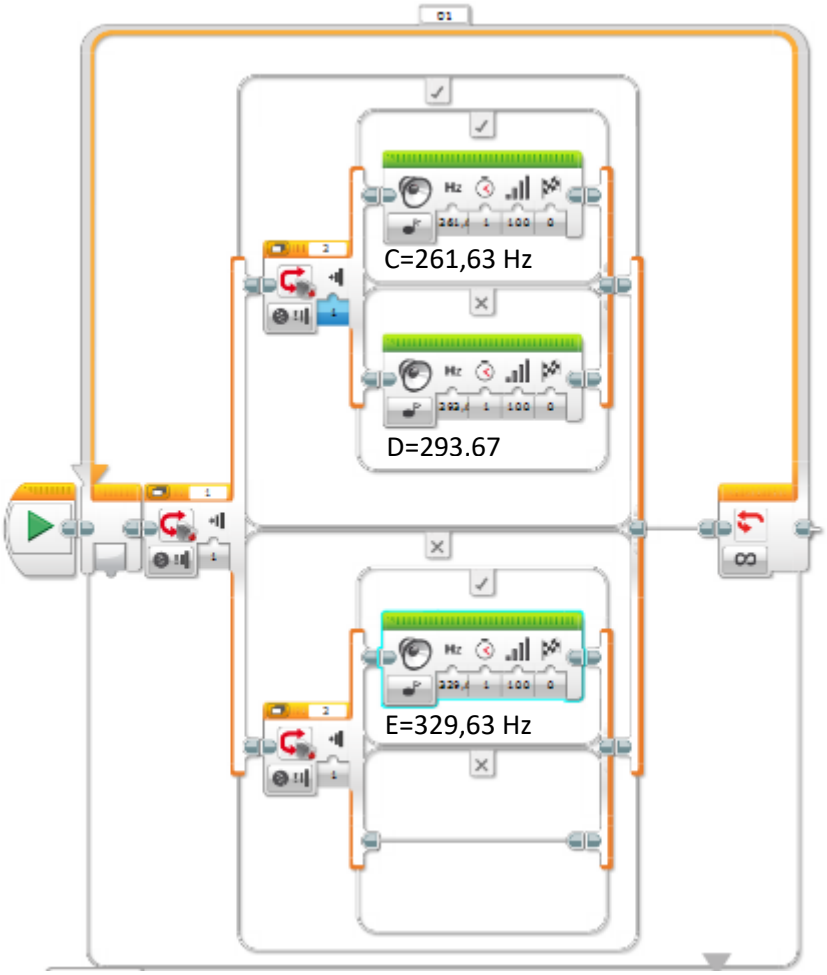
Auftrag 1: Der Schalter



Auftrag 2: Aufbau eines verschachtelten Schalters

	Um in den grünen Bereich zu kommen, muss der erste Sensor gedrückt sein (beachtet den grossen Hacken). Auch der zweite Sensor muss gerückt sein (beachtet wiederum den grossen Hacken). **Beide Sensoren müssen gedrückt sein.**
	Um in den gelben Bereich zu kommen, muss der erste Sensor gedrückt sein (beachtet den grossen Hacken). Der zweite Sensor darf aber nicht gedrückt sein (beachtet das grosse X). **Der erste Sensor muss gedrückt sein, der zweite dagegen nicht.**
	Um in den violetten Bereich zu kommen, darf der erste Sensor nicht gedrückt sein (beachtet das grosse X). Der zweite Sensor muss aber gedrückt sein (beachtet den grossen Hacken). **Der erste Sensor darf nicht gedrückt sein, der zweite dagegen schon.**
	Um in den roten Bereich zu kommen, darf der erste Sensor nicht gedrückt sein (beachtet das grosse X). Auch der zweite Sensor darf nicht gedrückt sein (beachtet auch hier das grosse X). **Beide Sensoren dürfen nicht gedrückt sein.**

Auftrag 3: Der verschachtelte Schalter



Wenn es euch zu lange geht bis der Ton gewechselt wird, verkleinert die Abspielzeit des Tons.

Baustein 3: Textausgabe & Variablen

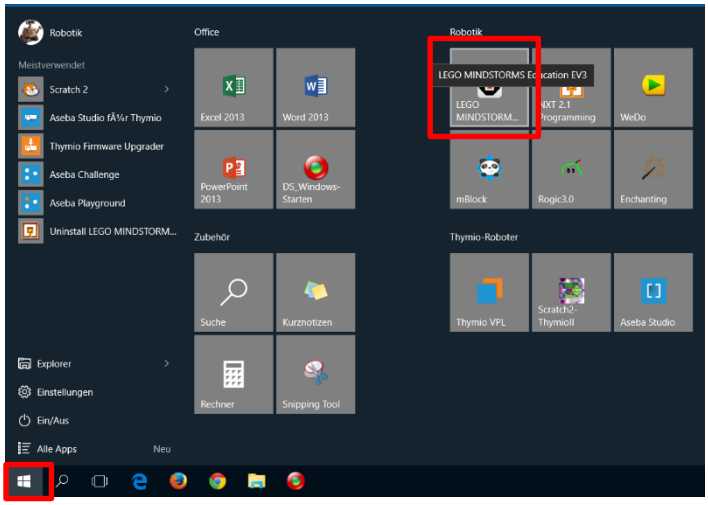
Ziele

Ich weiss, was eine Variable ist und wie man diese beim Programmieren verwendet. Ich kann einen Roboter so programmieren, dass plus und minus rechnet und das Resultat auf dem Display anzeigt.

Material

- 1 EV3 Lego Mindstorms Roboter
- 1 Notebook mit der Software LEGO Mindstorms EV3
- 1 USB Verbindungskabel

Hinweis zum Starten der Software LEGO Mindstorms EV3 starten:

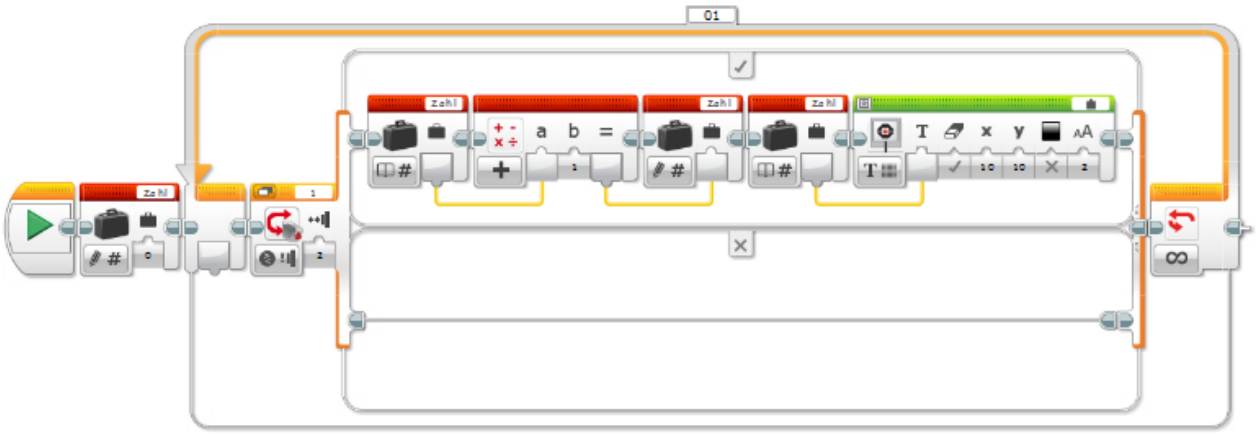


Vorgehen

Eine Variable ist für die Programmierer ein Wert, meist eine Zahl. Dieser Wert kann aber vom Roboter verändert werden (deshalb Variable!). Ein Programm (und somit ein Roboter) kann sehr viele verschiedene Variablen haben. Damit man jeweils weiss, von welcher Variable man spricht, gibt man ihr einen Namen, zum Beispiel „Zahl“.

Der Roboter hat also zum Beispiel die Variable „Zahl“. Diese hat zu Beginn den Wert 0. Der Roboter kann diesen Wert nun aber verändern. Im Programm, welches ihr euch gleich anschauen werdet, wird der Roboter die Variable immer um +1 verändern, wenn der Drucksensor gedrückt wird.

Baut das folgende Programm nach:



Ihr müsst genau hinschauen: Euer Programm soll genau gleich aussehen. Auch die kleinen Zahlen und die gelben Verbindungen müssen übereinstimmen!

Startet nun das Programm. Sobald ihr auf den Drucksensor drückt, wird eine Zahl auf dem Bildschirm angezeigt. Bei jedem Auslösen des Sensors wird die Zahl um eins erhöht. Der Roboter zählt für euch!

Auftrag 1: Funktionsweise Variable
Schaut euch das Programm und die verschiedenen Bausteine genau an. Was machen sie? Es ist wichtig, dass ihr **versteht**, wie das Programm funktioniert!

Hinweis: Eine mögliche Lösung dazu findet ihr unter den Musterlösungen.

Auftrag 2: Operationen ausführen
Verändert das Programm so, dass es den Wert immer verdoppelt. Beginnt mit dem Wert 2. Der Text am Display soll dieses Mal weiss sein. Ausserdem soll das Programm nach 10 Sekunden stoppen.

Hinweis: Eine mögliche Lösung dazu findet ihr unter den Musterlösungen.

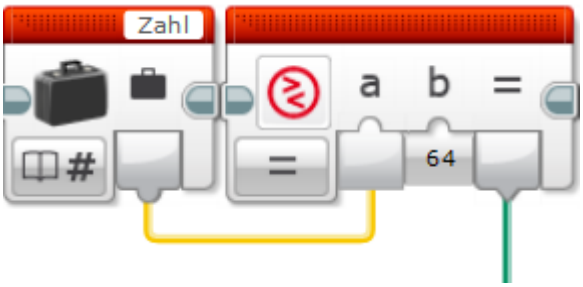
Auftrag 3: Werte bestimmen

Überlegt euch, wie der Anfangswert lautet, wenn das erste Resultat auf dem Display 1 sein soll.

Hinweis: Eine mögliche Lösung dazu findet ihr unter den Musterlösungen.

Arbeiten mit dem Vergleichen-Block: Um ein Programm zu stoppen, wenn die Variable einen bestimmten Wert, z.B. 64, erreicht hat, benötigt man den „Vergleichen-Block".
Doch wie benutzt ihr diesen?

Zuerst braucht ihr einen bekannten „Lesen" Block, um festzulegen, welchen Wert eure Variable im Moment besitzt. Diesen Wert gebt ihr an den Vergleichen-Block weiter (gelbe Linie). Bei der Zahl unter dem b (hier 64) könnt ihr eintragen, mit welcher Zahl der Wert verglichen wird.

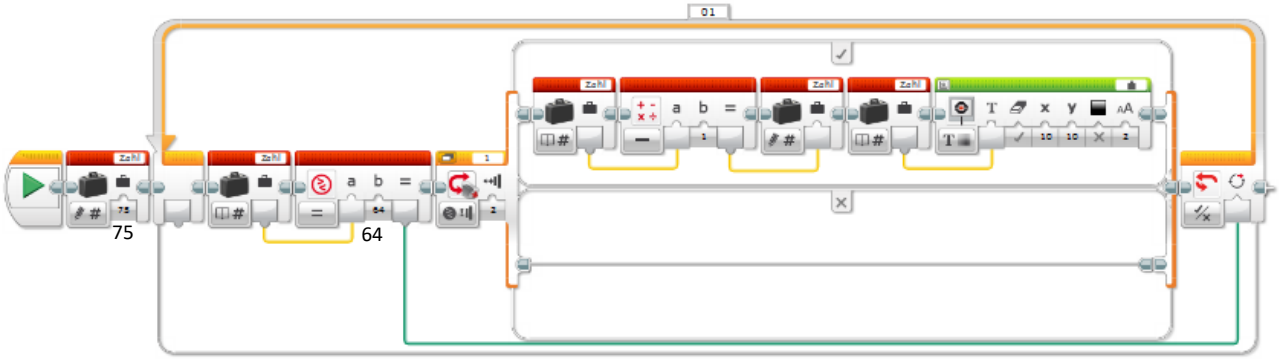


Mit einem Klick auf das grosse Gleichheitszeichen könnt ihr bestimmen, ob der Wert der Variable genau gleich, grösser oder kleiner als die Zahl b sein muss. In unserem Fall muss sie genau gleich sein (deshalb auch das Gleichheitszeichen).

Wie ihr euch vielleicht denken könnt, gibt auch der „Vergleichen-Block" einen Wert weiter (grüne Linie). Er gibt einen sogenannten **logischen Wert** weiter. Dieser logische Wert kann **„wahr"** oder **„falsch"** sein.
In unserem Falle ist dieser logische Wert genau dann „wahr", wenn der Wert unserer Variablen genau gleich ist wie die Zahl b, also 64!

Auftrag 4: Logischer Wert

Baut das untenstehende Programm nach und überlegt euch: Wann stoppt das Programm? Für was wird der „logische Wert" (grüne Linie) gebraucht?



Hinweis: Eine mögliche Lösung dazu findet ihr unter den Musterlösungen.

Auftrag 5: Vergleichen-Block

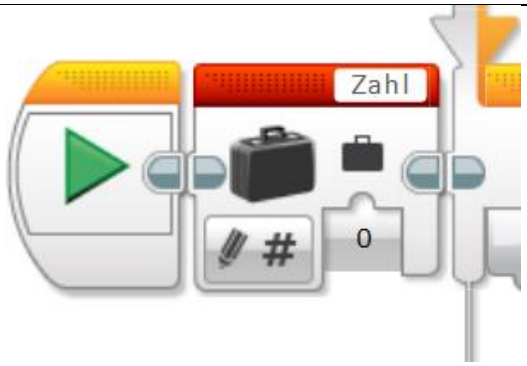
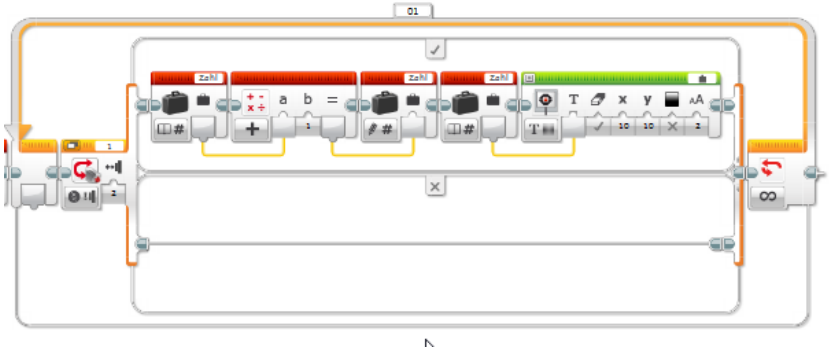
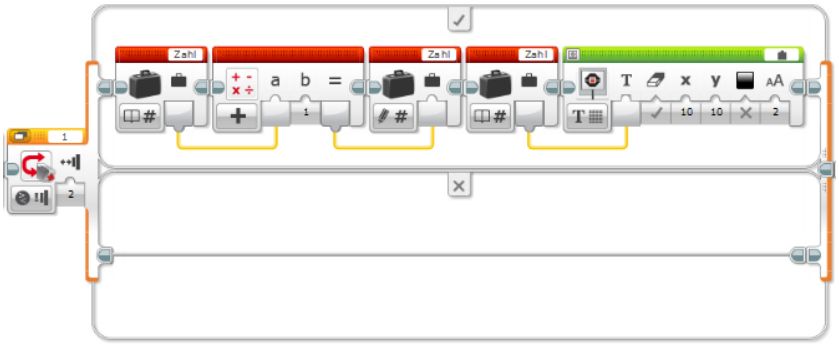
Verändert das Programm so, dass es bei 50 startet und in Zweierschritten aufwärts zählt. Das Programm soll stoppen, wenn der Wert der Variable „Zahl 65" übersteigt.

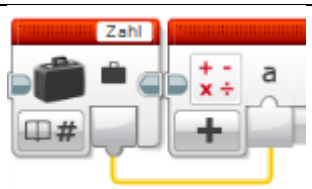
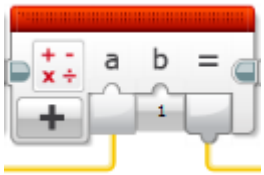
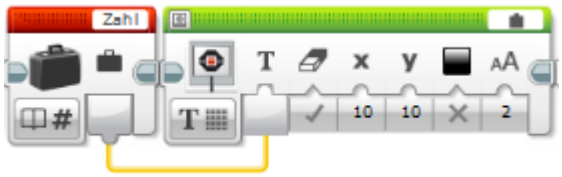
Hinweis: Eine mögliche Lösung dazu findet ihr unter den Musterlösungen.

Musterlösungen

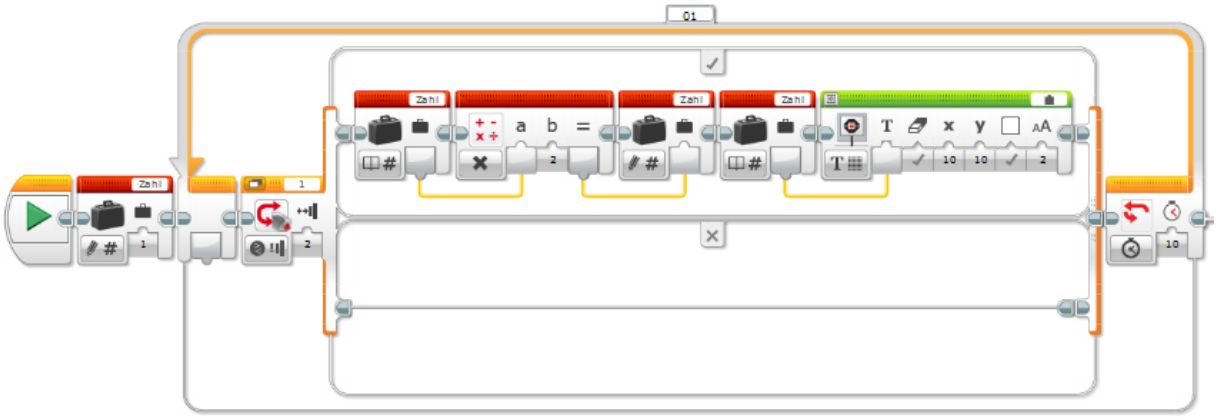
Bausteinposten 3: Textausgabe & Variablen

Auftrag 1: Funktionsweise Variable

Hier wird gleich nach dem Programm eine Variable erstellt. Die Variable heisst „Zahl“ und hat den Wert 0. Die Variable wird geschrieben (kleiner Bleistift). Es ist wichtig, dass dieser Baustein ausserhalb der Schleife ist. Wäre der Baustein innerhalb der Schleife, würde der Wert der „Zahl“ immer wieder auf 0 zurückgesetzt.	
Als nächstes kommt der Schlaufenblock. Er hat die Aufgabe, alles in der Schlaufe unendlich oft zu wiederholen. Würde das Programm nicht wiederholt werden, würde der Roboter nach einem Zählschritt aufhören zu zählen.	
Weiter geht es mit dem Schalter-Block. Er hat folgende Aufgabe: Wenn der Drucksensor ausgelöst wird, soll etwas geschehen. Wenn der Drucksensor nicht ausgelöst wird, passiert nichts.	
Nun kommt das Herzstück unseres Programms: Es besteht aus mehreren Blöcken.	

Zuerst wird der Wert der Variable „Zahl“ gelesen (Buchsymbol). Im ersten Durchgang ist dieser Wert 0. Er wird nun im nächsten Baustein übernommen (gelbe Linie).	
In diesem Block wird der Wert der Variable „Zahl“ vom vorherigen Block übernommen. Zu diesem Wert wird nun 1 dazu addiert (beachtet das grosse Pluszeichen und die 1). Es entsteht nun ein neuer Wert für die Variable „Zahl“, nämlich der Wert 1. Dieser Wert wird an den nächsten Block weitergegeben (gelbe Linie).	
Nun wird der Wert der Variable „Zahl“ erneut geschrieben. Dies ist wichtig, da nur so der Wert der Variable „Zahl“ geändert wird!	
Bei den letzten zwei Blöcken passiert Folgendes: Der Wert der Variable „Zahl“ wird nun gelesen. Mittlerweile ist dieser Wert auf 1 angewachsen. Dieser Wert wird an den nächsten Block weitergereicht (gelbe Linie) und von diesem an das Display deines Roboters gesendet.	

Auftrag 2: Operationen ausführen



Auftrag 3: Werte bestimmen

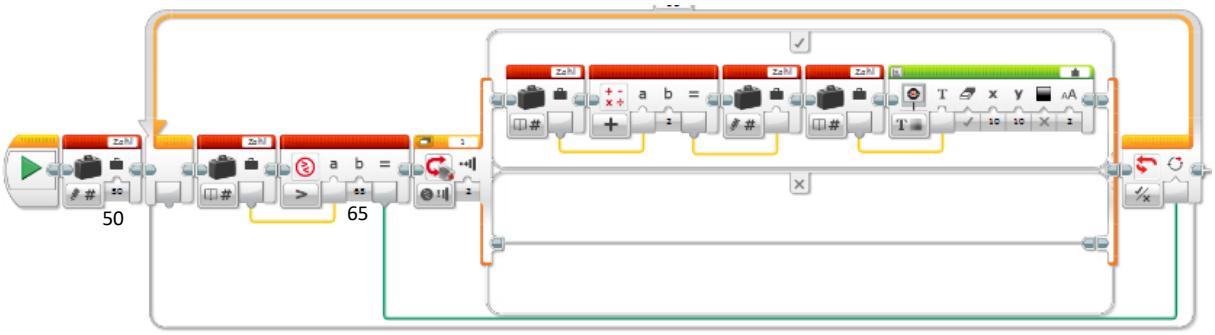
Es muss mit dem Wert 0.5 begonnen werden.

Auftrag 4: Logischer Wert

Das Programm stoppt genau dann, wenn der Zähler 64 erreicht. Der „logische Wert“ wird also dazu gebraucht, um die **Schleife zu beenden**.

Tipp: Auch Schalter können mit solchen „logischen Werten“ *umgelegt* werden.

Auftrag 5: Vergleichen-Block



Autorenschaft: Andrea Maria Schmid, Urs Meier (PH Luzern)
Unterstützt durch Studierende: Michael Wyrsch, Silvan Petermann
Version: März 2017