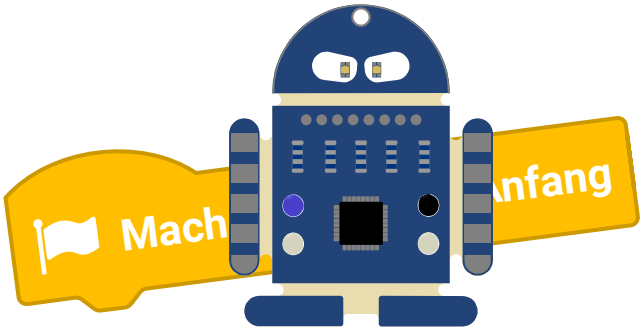


Code!

B1 Los geht‘s!

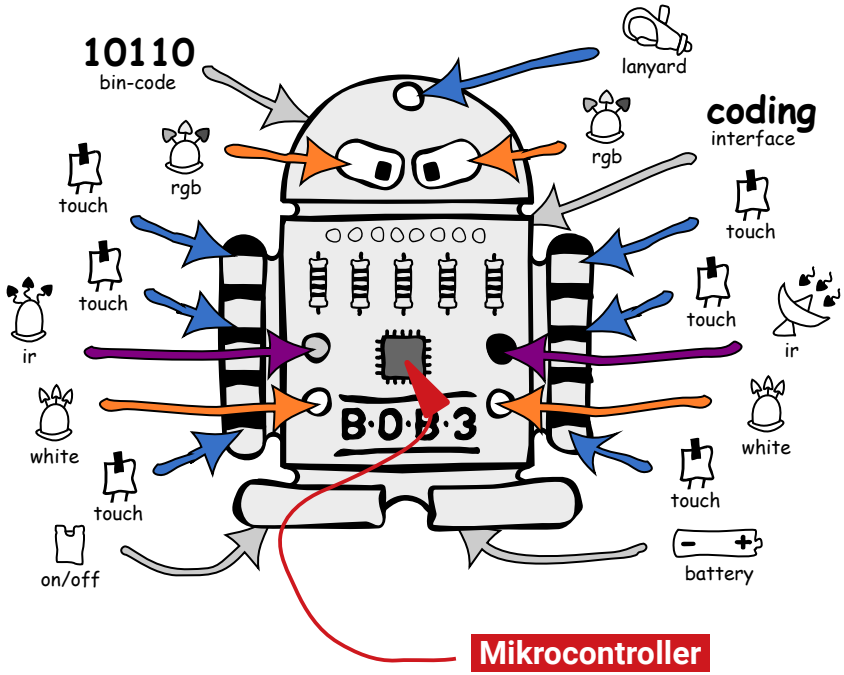
Thema:	Erste Schritte
Bereich:	Basics/Grundlagen
Voraussetzung:	Keine
Lernziele:	Überblick über die Programmierumgebung, Compilieren und Übertragen, erste Programmierschritte, Augen-Leds ansteuern, Farben ändern, Anweisungen verstehen und anwenden, Bauch-Leds ansteuern, Parameter ändern
Anspruch:	★☆☆☆
Aufgaben:	A1 – A7
Zeitbedarf:	20 min

B1 Los geht‘s!

Thema: Erste Schritte
Bereich: Basics
Anspruch: ★☆☆☆
Zeitbedarf: 20 min

BOB3 ist ein kleiner Roboter, der genau das macht, was du möchtest. Damit er dich verstehen kann, musst du alle Befehle in einer **Programmiersprache** schreiben. Wir verwenden die Programmiersprache **Scratch**. In Scratch sind die einzelnen Programmierbefehle bunte Blöcke, so wie z.B. diese hier:

Die Befehle werden dann compiliert, also in Maschinensprache übersetzt und auf den Bob übertragen. Der **Mikrocontroller** vom Bob (das kleine schwarze Kästchen auf dem Bauch) kann die Maschinensprache verstehen und weiss dann genau, was zu tun ist!



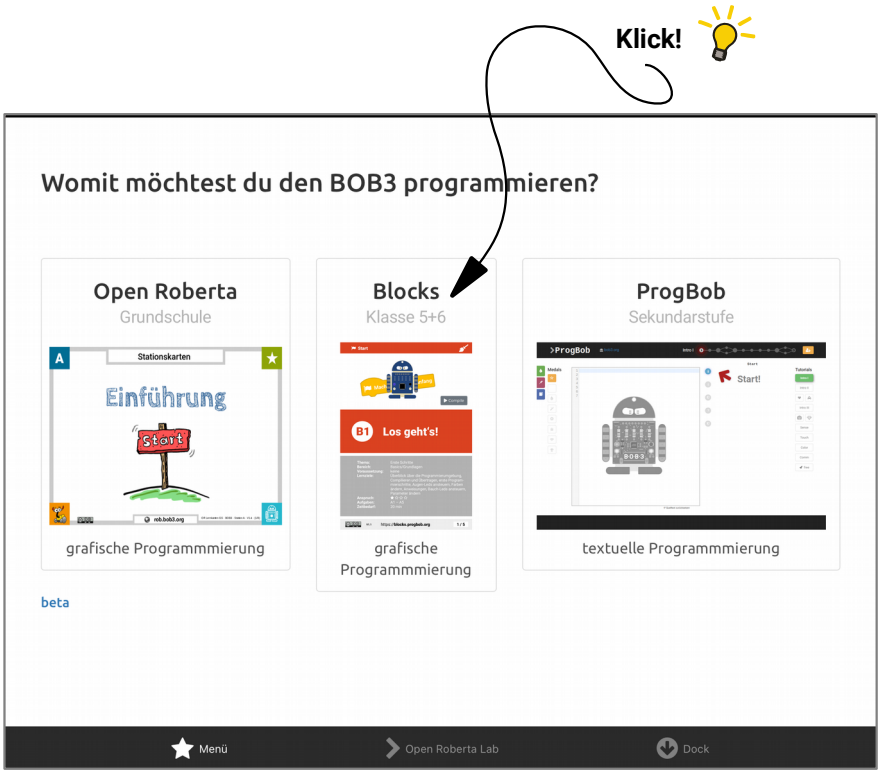
B1 Los geht‘s!

iPad

Thema: Erste Schritte
Bereich: Basics
Anspruch: ★☆☆☆
Zeitbedarf: 20 min

Jetzt bereiten wir alles für die **Programmierung** vor:

- **Aufgabe 1a:** Starte die BOB3 App
- **Aufgabe 1b:** Schalte das BobDock ein und verbinde es
- **Aufgabe 2:** Wähle die **mittlere** Lernkarte ‚**Blocks**‘



B1 Los geht‘s!

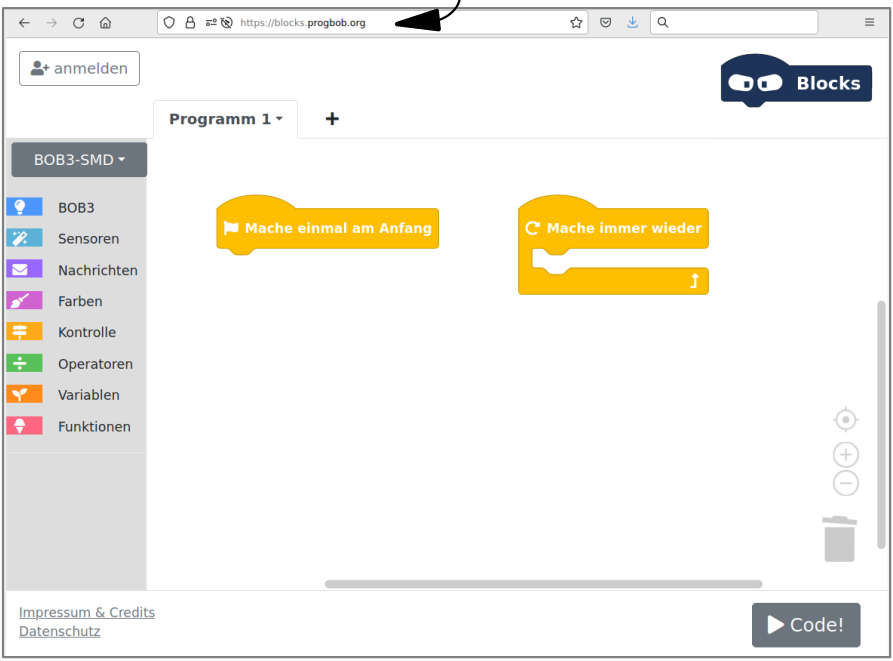
PC/ Laptop

Thema: Erste Schritte
Bereich: Basics
Anspruch: ★☆☆☆
Zeitbedarf: 20 min

Jetzt bereiten wir alles für die **Programmierung** vor:

- **Aufgabe 1:** Verbinde den BOB3 mit dem Laptop oder mit dem PC
- **Aufgabe 2:** Starte den Webbrowser und tippe **blocks.progbob.org** als Adresse ein:

blocks.progbob.org



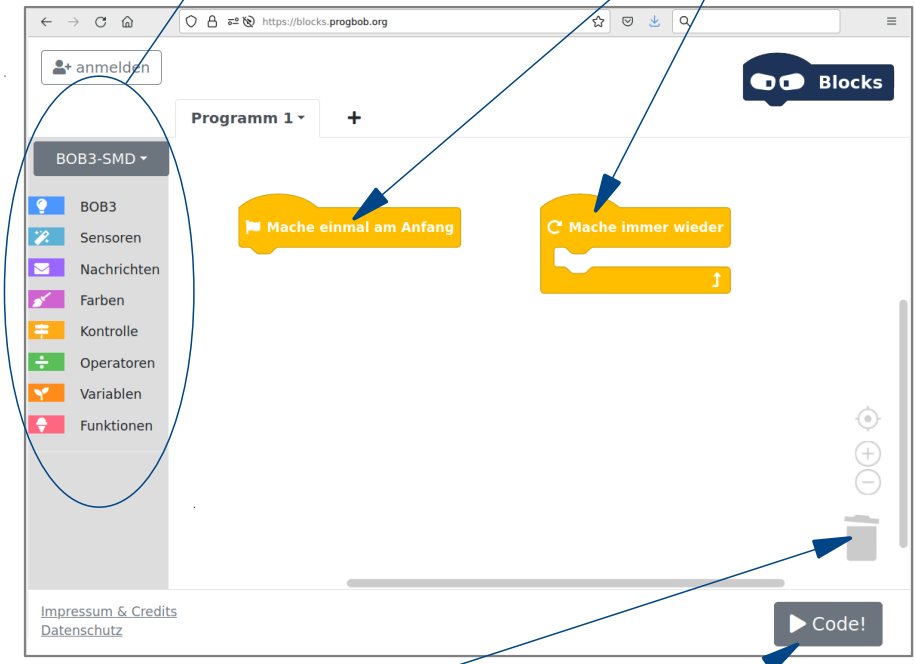
B1 Los geht‘s!

Thema: Erste Schritte
Bereich: Basics
Anspruch: ★☆☆☆
Zeitbedarf: 20 min

Die **Programmierumgebung** von BOB3 hat verschiedene Bereiche und einige Funktionen. Schau dir erstmal alles ganz genau an:

Befehls-Blöcke:
Alles was Bob machen kann, also alle verschiedenen Blöcke findest du hier

Programm-Blöcke:
Alles was Bob machen soll, fügst du hier ein



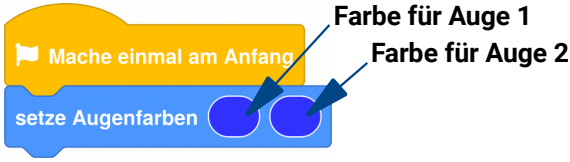
Mülleimer:
Löschen von Blöcken

Code-Button:
Programm auf den BOB3 übertragen

B1 Los geht‘s!

Thema: Erste Schritte
Bereich: Basics
Anspruch: ★☆☆☆
Zeitbedarf: 20 min

Aufgabe 3: **Jetzt programmieren wir!** Bob soll beide Augen in Blau einschalten. Verwende den Block «*Mache einmal am Anfang*» und den Block «*setze Augenfarben*» aus der blauen Rubrik ‚**BOB3**‘. Dein Programm sollte jetzt so aussehen:



Aufgabe 4: Klicke nun unten rechts auf den **Code-Button**. Was macht der Bob? Leuchten seine Augen blau?

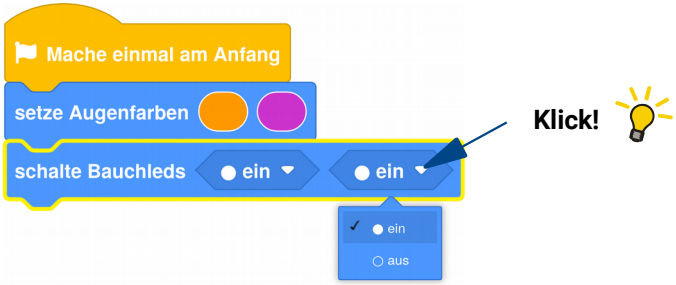
Aufgabe 5: **Viele Farben!** Per Klick auf ein Farbfeld kannst du die Augenfarben ändern, probiere mal deine **Lieblingsfarben** aus! Können die beiden Augen auch in verschiedenen Farben leuchten?

B1 Los geht‘s!

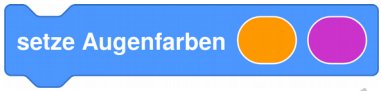
Thema: Erste Schritte
Bereich: Basics
Anspruch: ★☆☆☆
Zeitbedarf: 20 min

Aufgabe 6: **Mehr Licht bitte!** Jetzt soll Bob mal zeigen, was er kann: Wir schalten auch noch die beiden hellen weißen Leds an seinem Bauch ein. Suche den Befehlsblock «***schalte Bauchleds***» und programmiere folgendes Programm:

Aufgabe 7: Per Klick kannst du die Eigenschaft ‚**ein**‘ in ‚**aus**‘ ändern. Probiere das mal aus: Jetzt sollen nur die **beiden Augen** und die **linke Bauch-Led** leuchten!

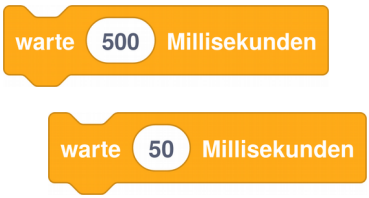


Wissensbox



Anweisung

Eine ‚*Anweisung*‘ ist eine Aufforderung an den Mikrocontroller (bzw. Computer), etwas auszuführen. Der Befehlsblock «***setze Augenfarben***» ist zum Beispiel eine Anweisung.



B2 Erste Programme

Thema:	Sequenzen
Bereich:	Basics/Grundlagen
Voraussetzung:	Station **B1**
Lernziele:	Erste eigene Programme, Sequenzen, Prinzip der Verzögerung, «Mache immer wieder»-Block, Blinklichter erzeugen, Varianten entwickeln
Anspruch:	★☆☆☆
Aufgaben:	A1 – A10
Differenzierung:	A11 + A12
Zeitbedarf:	30 min

B2 Erste Programme

Thema: Sequenzen
Bereich: Basics
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

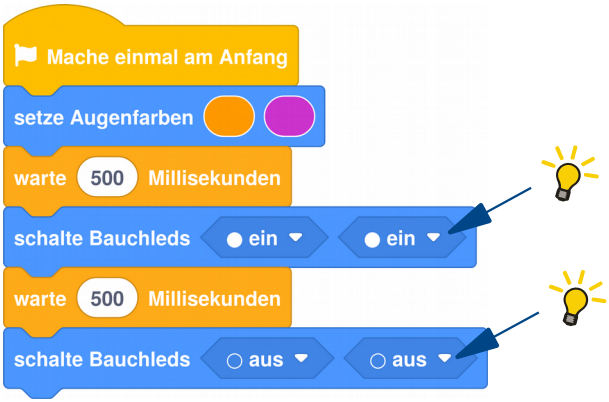
Jetzt wollen wir mit dem ‚**Prinzip der Verzögerung**' arbeiten und lernen dafür eine neue Anweisung kennen:

Wissensbox

Prinzip der Verzögerung

Die Anweisung «*warte 500 Millisekunden*» bewirkt, dass der Mikrocontroller von BOB3 500 Millisekunden (das ist eine halbe Sekunde) wartet.

Aufgabe 1: Wir probieren das mal aus! Erweitere dein Programm um zwei «***warte 500 Millisekunden***»-Blöcke aus der Rubrik ‚Kontrolle' und noch einen «***schalte Bauchleds***»-Block. Achte auf die **Reihenfolge** der einzelnen Blöcke und die Einstellungen der Bauchleds!



Was macht das Programm? Teste es mit BOB3!

a Die Augen leuchten für 500 Millisekunden farbig auf und sind dann aus

b Das Programm startet nach 500 Millisekunden

c Die Bauchleds werden nach 500 Millisekunden für 500 Millisekunden eingeschaltet und sind dann aus

B2 Erste Programme

Thema: Sequenzen
Bereich: Basics
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

Das Programm aus Aufgabe 6 enthält eine **Sequenz**, es besteht aus einer Abfolge von fünf Anweisungen, die nacheinander ausgeführt werden.

Wissensbox

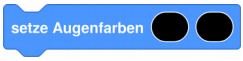
Sequenz

Eine Sequenz ist eine Abfolge von einzelnen Anweisungen, die nacheinander ausgeführt werden.

Aufgabe 2: Ändere dein Programm, der Bob soll **nacheinander** seine vier Leds einschalten, und zwar so:

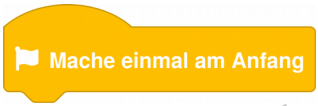
→ Auge 1 in *Orange* einschalten, Auge 2 *ausschalten*
→ 500 Millisekunden *abwarten*
→ zusätzlich Auge 2 in *Pink* einschalten
→ 500 Millisekunden *abwarten*
→ linke Bauchled *einschalten*, rechte Bauchled *aus*
→ 500 Millisekunden *abwarten*
→ zusätzlich rechte Bauchled *einschalten*

Die Farbe ‚schwarz' schaltet die Augen aus



Bisher haben wir mit dem «***Mache einmal am Anfang***»-Block gearbeitet. Alle Anweisungen die wir unter dem Block eingefügt haben, werden nach der Programmübertragung **genau einmal ausgeführt**:

Wissensbox



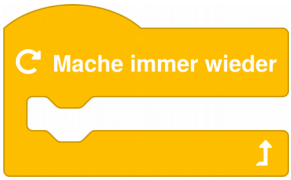
«*Mache einmal am Anfang*»-Block

Die Befehlsblöcke unter dem «*Mache einmal am Anfang*»-Block werden direkt nach dem Einschalten automatisch genau **einmal** ausgeführt.

B2 Erste Programme

Thema: Sequenzen
Bereich: Basics
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

Jetzt lernen wir einen neuen Block kennen, den «***Mache immer wieder***»-Block. Alle Anweisungen, die wir in diesen Block einbauen werden **immer wieder ausgeführt**:



Wissensbox

«*Mache immer wieder*»-Block

Die Befehlsblöcke die innerhalb des «*Mache immer wieder*»-Blocks stehen werden automatisch **immer wieder** ausgeführt.

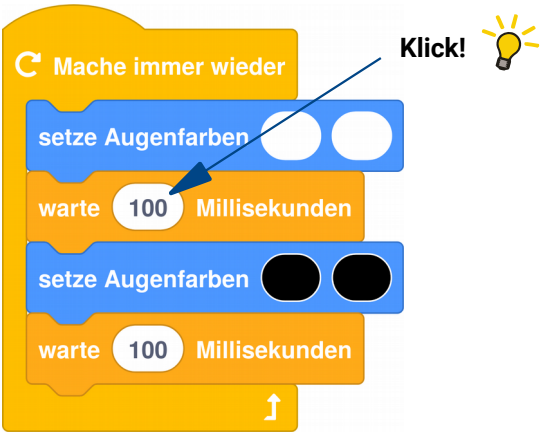
Aufgabe 3: Programmiere das folgende Programm und teste es auf dem BOB3. Was macht der Bob?

Aufgabe 4: **Ändere** dein Programm: Verwende nun anstelle des «*Mache einmal am Anfang*»-Blocks einen «***Mache immer wieder***»-Block. Was macht der Bob jetzt anders?

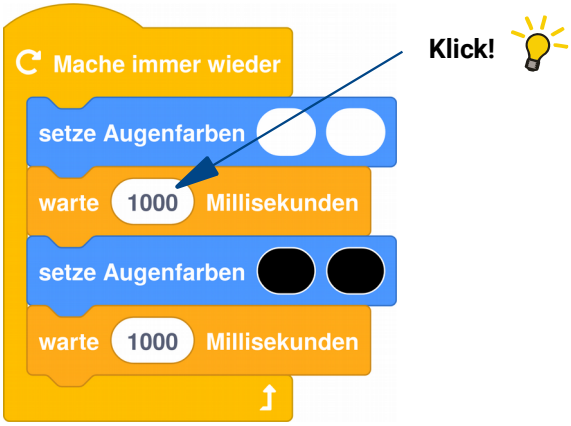
B2 Erste Programme

Thema: Sequenzen
Bereich: Basics
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

Aufgabe 5: Ändere die **500** in dem «*warte Millisekunden*»-Block in eine **100**. Was macht der Bob jetzt?



Aufgabe 6: Jetzt probiere mal **1000** Millisekunden aus! Überlege dir, was jetzt passieren wird und teste dein Programm mit BOB3!



Aufgabe 7: Probiere noch ein paar andere Zahlen aus und schreibe eine Regel auf: Bei welchen Zahlen verhält der Bob sich wie und warum macht er das?

Erste Programme

Thema: Sequenzen
Bereich: Basics
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

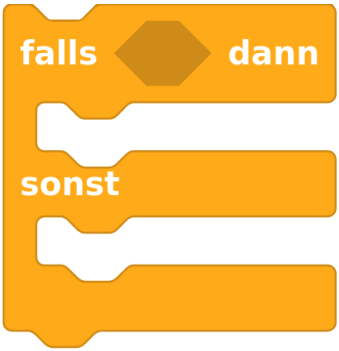
Aufgabe 8: **BOB3 als Baustellenlicht!** Programmiere das folgende Programm und teste es:

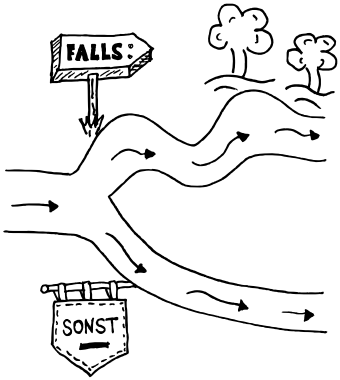
Aufgabe 9: Ändere das Programm aus Aufgabe 8, so dass Bob langsamer blinkt!

Aufgabe 10: Ändere das Programm aus Aufgabe 8, so dass Bob schneller blinkt!

Aufgabe 11: Ergänze das Programm aus Aufgabe 8, so dass zusätzlich zu den Augen beide Bauch-Leds abwechselnd weiß blinken!

Aufgabe 12: Jetzt wollen wir ein **Überkreuz-Blinken** programmieren! Ändere dein Programm aus Aufgabe 11 so:
Die Augen sollen gelb hin und her blinken, die Bauch-Leds sollen weiß hin und her blinken und immer wenn das **linke Auge** an ist, soll auch die **rechte Bauch-Led** an sein. Wenn das **rechte Auge** an ist, soll auch die **linke Bauch-Led** an sein!



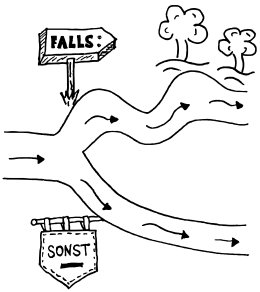


W1 Verzweigung

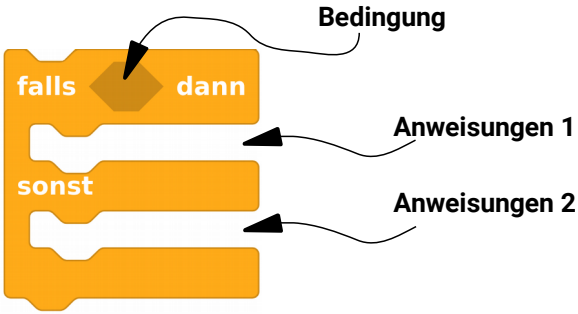
Thema:	Verzweigung
Bereich:	Wissen
Voraussetzung:	Station **B2**
Lernziele:	Bedeutung und Anwendung von Verzweigungen, Vergleichsoperatoren, Wahrheitswerte, «falls-dann»-Block
Anspruch:	★☆☆☆
Aufgaben:	A1 – A11
Differenzierung:	A12
Zeitbedarf:	30 min

W1 Verzweigung

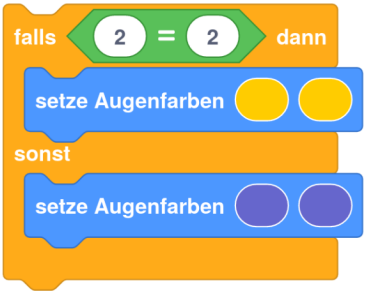
Thema: Verzweigung
Bereich: Wissen
Anspruch: ★☆☆☆
Zeitbedarf: 30 min



Eine **Verzweigung** ermöglicht, dass in Abhängigkeit von einer **Bedingung** bestimmte Anweisungen ausgeführt werden und andere dagegen nicht! **Falls** die Bedingung **wahr** ist, dann werden die ***Anweisungen 1*** ausgeführt, **sonst**, also wenn die Bedingung **falsch** ist, werden die ***Anweisungen 2*** ausgeführt:



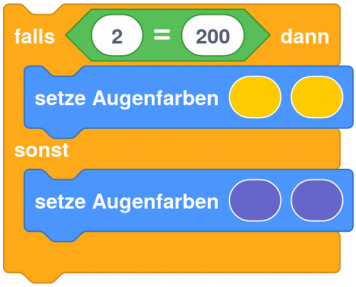
Aufgabe 1: Probiere mal folgendes Beispiel aus:



Wenn man dieses Programm auf dem BOB3 laufen lässt, dann bekommt man immer dasselbe Ergebnis: **die Augen leuchten gelb**!

Begründung: Da die **Bedingung „2 = 2“ wahr** ist, wird der falls-Zweig, also in diesem Beispiel die Anweisung «***setze Augenfarben gelb gelb***» ausgeführt.

Aufgabe 2: Nun **ändere** mal die Bedingung in ‚**2 = 200**'.
Was passiert jetzt?



Wenn man dieses Programm auf dem BOB3 laufen lässt, dann bekommt man immer dasselbe Ergebnis: **die Augen leuchten blau**!

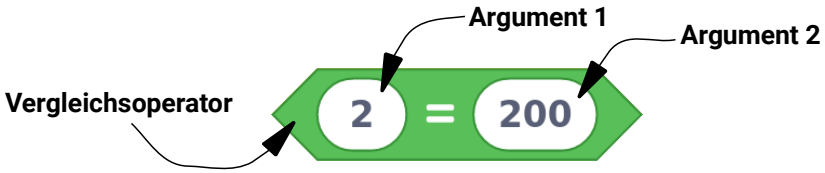
Begründung: Da die **Bedingung „2 = 200" falsch** ist, wird der sonst-Zweig, also in diesem Beispiel die Anweisung «***setze Augenfarben blau blau***» ausgeführt.

Wissensbox

Vergleichsoperator

Ein Vergleichsoperator ist ein logischer Operator, der auf zwei Argumente (z.B. Zahlen) angewendet wird und einen **Wahrheitswert** liefert.

Der Vergleichsoperator aus unserem Beispiel sieht so aus:



W1 Verzweigung

Thema: Verzweigung
Bereich: Wissen
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

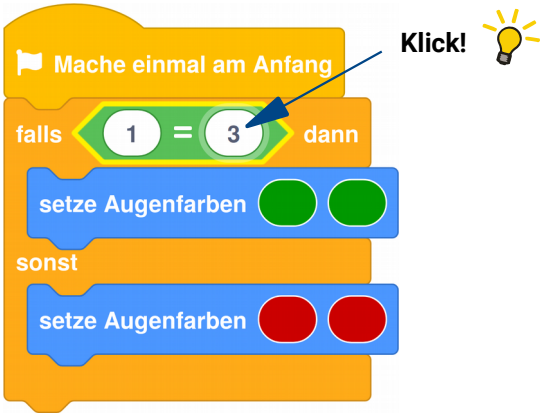
Wir verwenden drei verschiedene Vergleichsoperatoren:

Vergleichsoperator	Mathe	Beispiele	Erklärung
	=	**3 = 3** (wahr) **2 = 200** (falsch)	→ ergibt ‚**wahr**', falls das linke und das rechte Argument **gleich** sind
	<	**1 < 5** (wahr) **900 < 300** (f)	→ ergibt ‚**wahr**', falls das linke Argument **kleiner** als das rechte Argument ist
	>	**33 > 12** (wahr) **10 > 1000** (f)	→ ergibt ‚**wahr**', falls das linke Argument **größer** als das rechte Argument ist

Aufgabe 3: **BOB3 als Wahrheitsfinder!**

Jetzt programmieren wir den BOB3 so, dass er *wahr* und *falsch* erkennt: Wir geben eine Bedingung vor. **Falls** diese Bedingung **wahr** ist, sollen die Augen **grün** leuchten. **Sonst**, also wenn die Bedingung **falsch** ist, sollen die Augen **rot** leuchten! Probiere mal:

Aufgabe 4: Ändere die Bedingung ‚1 = 1' in ‚**1 = 3**'. Teste das neue Programm auf BOB3. Was passiert jetzt?



Aufgabe 5: Ändere die Bedingung in ‚**8 = 8**'.
Was erwartest du jetzt?

Aufgabe 6: Ändere die Bedingung in ‚**3 < 1**'.
Was erwartest du jetzt?

Aufgabe 7: Welche der folgenden Bedingungen sind **wahr** und welche sind **falsch**?

a) 9 = 6

b) 155 = 155

c) 8 = 5+3

d) 14 = 20-5

e) 5 < 4

f) 50 < 40

g) 400 > 500

W1 Verzweigung

Thema: Verzweigung
Bereich: Wissen
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

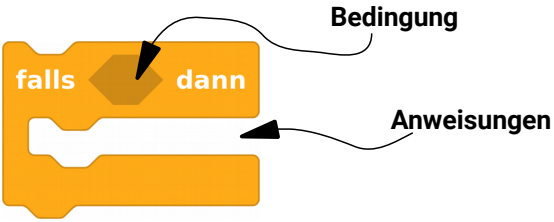
Aufgabe 8: Betrachte das folgende Programm. Wie viele Leds am Bob leuchten?

Aufgabe 9: Man kann einen «*falls-dann-sonst*»-Block um weitere «***sonst-falls***»**-Zweige** erweitern (per Rechtsklick auf den Block). Was macht das folgende Programm? In welchen Farben leuchten Bob's Augen?

Verzweigung

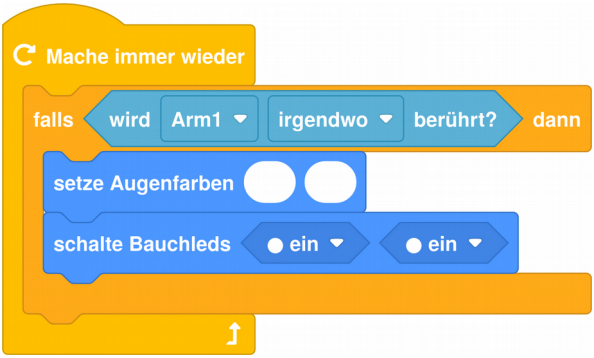
Thema: Verzweigung
Bereich: Wissen
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

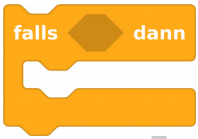
Da man manchmal den «*sonst*»-Zweig nicht benötigt, gibt es auch den «***falls dann***»-Block:



Falls die Bedingung **wahr** ist, dann werden die ***Anweisungen*** ausgeführt. Das sieht dann z.B. so aus: **Falls** die Bedingung «*wird Arm1 irgendwo berührt*» **wahr** ist, **dann** werden alle Lampen eingeschaltet.

Aufgabe 10: Probiere das folgende Programm aus:





Wissensbox

«*falls-dann*»-Block

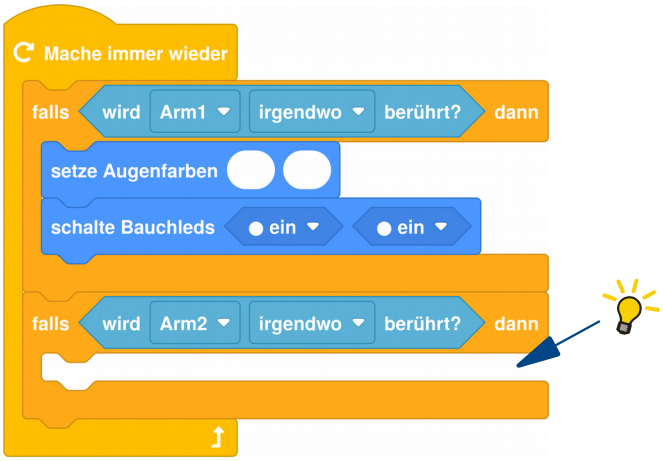
Ein «*falls-dann*»-Block ist eine bedingte Anweisung: **Falls** eine Bedingung **wahr** ist, **dann** werden die Anweisungen der Sequenz ausgeführt.

W1 Verzweigung

Thema: Verzweigung
Bereich: Wissen
Anspruch: ★ ☆ ☆ ☆
Zeitbedarf: 30 min

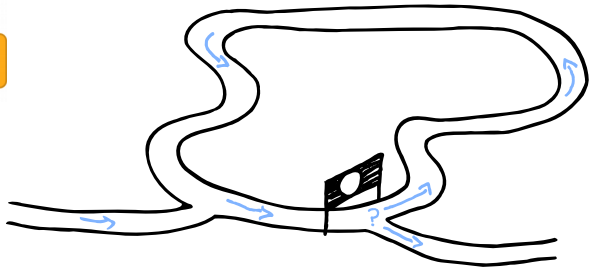
Aufgabe 11: **Wir programmieren Bob als Leselicht:**

Ergänze das Programm: Falls Arm1 irgendwo berührt wird, dann werden alle Leds eingeschaltet, mit Berührung von Arm2 sollen alle Leds ausgeschaltet werden!



Aufgabe 12: Programmiere Bob als **blinkendes Leselicht!**

Verändere dein Programm aus Aufgabe 11 so:
Falls **Arm1** irgendwo berührt wird, dann sollen die beiden **Augen** in deinen Lieblingsfarben wild blinken.
Falls **Arm2** irgendwo berührt wird, dann sollen die beiden **Bauch-Leds** in Weiß blinken.

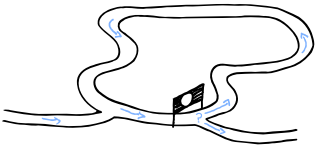


W2 Schleifen

Thema:	Schleifen
Bereich:	Wissen
Voraussetzung:	Station **W1**
Lernziele:	Bedeutung und Anwendung von Schleifen kennenlernen, «Wiederhole x-mal»-Schleife, «Wiederhole bis»-Schleife, Endlosschleife, Abbruch von Schleifen
Anspruch:	★★☆☆
Aufgaben:	A1 – A9
Zeitbedarf:	30 min

W2 Schleifen

Thema: Schleifen
Bereich: Wissen
Anspruch: ★★☆☆
Zeitbedarf: 30 min



Beim Programmieren ist es manchmal nützlich, wenn man einen bestimmten Programmteil **mehrfach wiederholen** kann. Für diesen Zweck gibt es die Kontrollstruktur **Schleife**. Wir verwenden drei verschiedene Schleifen:

«***wiederhole x mal***»-Schleife

die Zahl suchst du selber aus

«***wiederhole bis***»-Schleife

«***wiederhole fortlaufend***»-Schleife

Wir schauen uns zunächst die «***wiederhole x mal***»-**Schleife** an:

Wissensbox

«*wiederhole x mal*»-Schleife

Die «wiederhole *x mal*»-Schleife ermöglicht, dass alle Anweisungen, die innerhalb der Schleife stehen genau x-mal ausgeführt werden.

W2 Schleifen

Thema: Schleifen
Bereich: Wissen
Anspruch: ★★☆☆
Zeitbedarf: 30 min

Wir schauen uns dazu ein Beispiel an, Bob soll uns genau **3-mal** zuzwinkern:

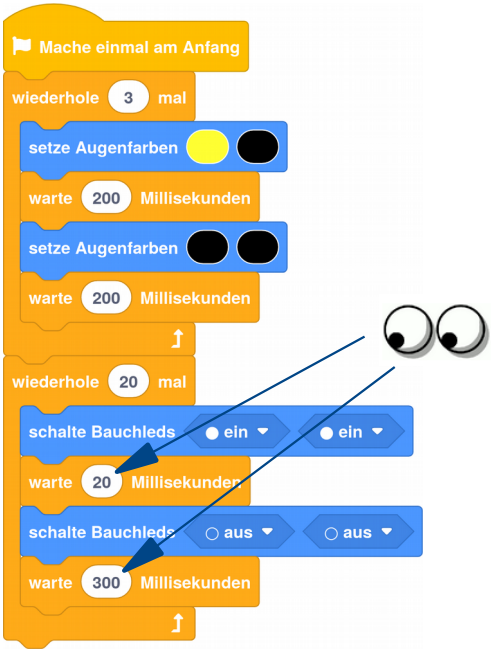
Aufgabe 1: Probiere das Beispiel aus, verwende den «***wiederhole 3 mal***»-Block aus dem Bereich **Kontrolle** und zähle mit! Wie oft zwinkert Bob dir zu?

Aufgabe 2: Ändere dein Programm:
Bob soll dir jetzt **acht mal** zuzwinkern. Hast du eine Idee, wie das geht? Probiere mal!

Aufgabe 3: Erweitere dein Programm:
Bob soll dir zuerst **acht mal** mit **Auge 1** zuzwinkern. Danach soll er mit beiden **Bauch-Leds** schnell **10 mal** aufblitzen!

Aufgabe 4: Jetzt mal mehr Tempo!
Ändere die **Geschwindigkeit**:
Bob soll dir wieder zuerst **acht mal** mit **Auge 1** freundlich zuzwinkern. Dann soll er mit den beiden **Bauch-Leds** sehr sehr schnell **20 mal** aufblitzen. Und zwar so schnell, dass man kaum noch mitzählen kann!

Tip: Ein schönes **Aufblitzen** mit den Leds entsteht, wenn man zuerst ganz **kurz** abwartet (z.B. 20 Millisekunden) und dann **länger** wartet (z.B. 300 Millisekunden):



Jetzt schauen wir uns die «***wiederhole bis***»-**Schleife** an:

Wissensbox

«*wiederhole bis*»-Schleife

Mit der «*wiederhole bis*»-Schleife werden alle Anweisungen innerhalb der Schleife solange wiederholt, bis die Bedingung wahr ist.

W2 Schleifen

Thema: Schleifen
Bereich: Wissen
Anspruch: ★★☆☆
Zeitbedarf: 30 min

Wir probieren mal ein Beispiel aus:

Bob soll wie eine **Auto-Diebstahlsicherung** funktionieren. Solange kein Dieb in der Nähe ist, der IR-Sensor also keinen Dieb bemerkt, wartet Bob ab. Während er wartet, soll er mit den Augen in dunkelrot kurz aufblitzen. Sobald der IR-Sensor einen Wert größer als 15 bemerkt, (wenn also z.B. deine Hand vor dem Sensor ist) dann soll er mit den Bauch-Leds ein weißes Warnblinklicht machen!

Aufgabe 5: Probiere das Beispiel aus, verwende den «***wiederhole bis***»-Block und einen «***wiederhole 50 mal***»-Block aus dem Bereich **Kontrolle**. Dann halte mal deine Hand oder ein Blatt Papier vor den IR-Sensor. Was macht der Bob?

Aufgabe 6: Ändere dein Programm:
Die Bauch-Leds sollen jetzt **abwechselnd blinken**.
Wenn die linke Bauch-Led an ist, dann soll die rechte aus sein und umgekehrt!

Im Moment funktioniert unsere **Auto-Diebstahlsicherung** so, dass genau 50 mal das Alarmblinken stattfindet. Eigentlich müsste Bob aber solange Alarm anzeigen, bis der Besitzer des Autos den Alarm abschaltet!

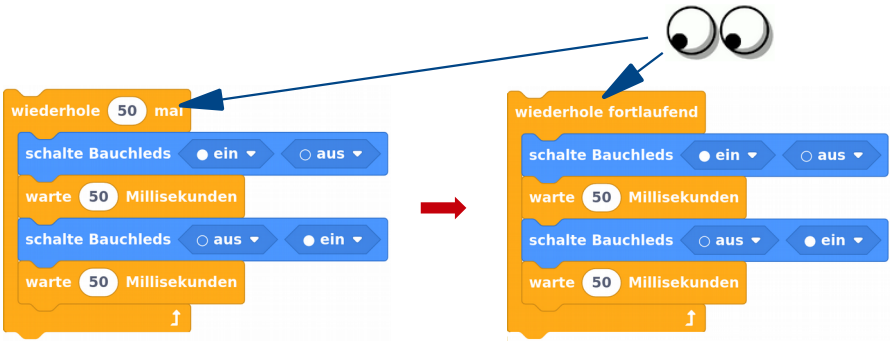
Für diesen Zweck lernen wir jetzt die «***wiederhole fortlaufend***»-**Schleife** kennen:

Wissensbox

«*wiederhole fortlaufend*»-Schleife

Die «*wiederhole fortlaufend*»-Schleife ist eine sogenannte **Endlosschleife**. Alle Anweisungen innerhalb der Schleife werden immer wieder wiederholt.

Aufgabe 7: **Ersetze** den «***wiederhole 50 mal***»-Block durch einen «***wiederhole fortlaufend***»-Block und teste dein Programm! Welchen Unterschied stellst du fest?



Unsere Auto-Diebstahlsicherung wäre jetzt also **für immer** im Alarm Modus. Oder zumindest solange, bis die Batterie leer ist ;-)

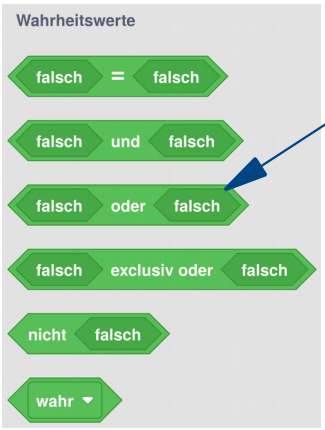
Jetzt programmieren wir noch eine **Alarm-Reset** Funktion für den Besitzer des Autos, damit dieser das Alarmblinken auch wieder abstellen und den Sensor-Detektion Modus wieder starten kann. Hierfür benötigen wir den «***die Schleife abbrechen***»-Block:

Wissensbox

Abbruch einer Schleife

Mit dem «*die Schleife abbrechen*»-Block kann man jede Schleife direkt **beenden**.

Aufgabe 8: **Erweitere** dein Programm:
Mit Berührung der Armsensoren soll der **Alarm neu gestartet** werden. **Falls** einer der beiden Arme (Arm 1 oder Arm 2) berührt wird, **dann** soll das Alarmblinken ausgeschaltet und die Schleife abgebrochen werden!

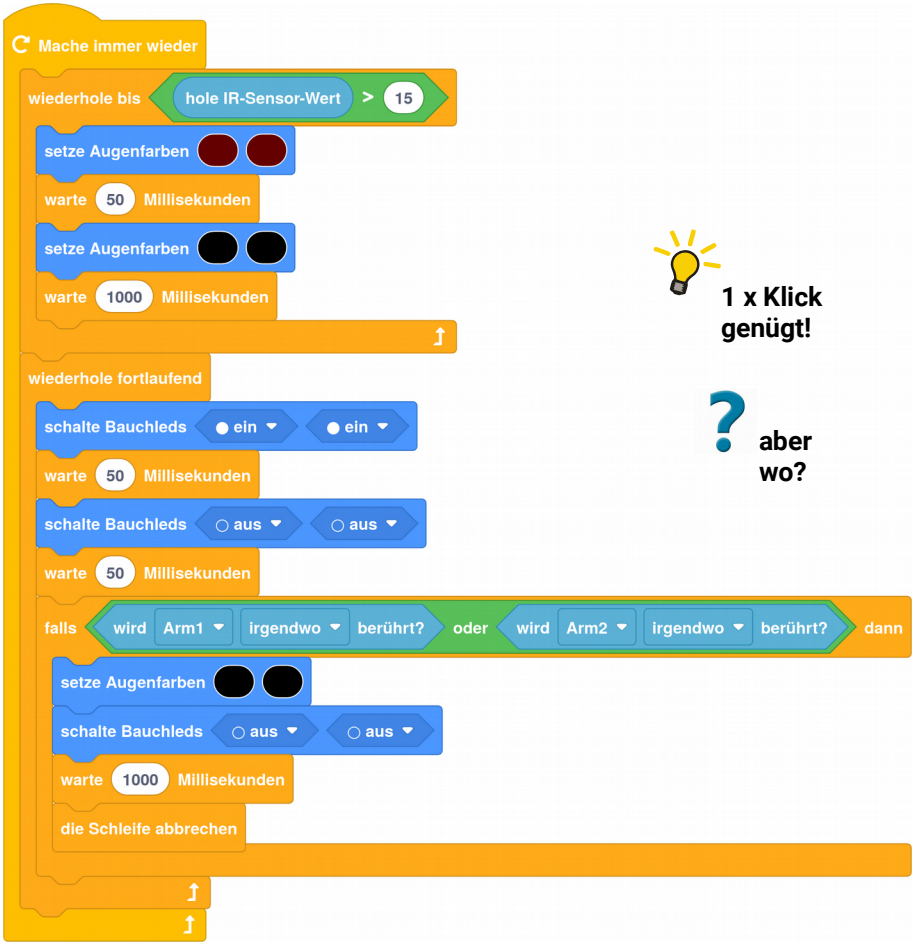


Tip:
Verwende den ‚oder' Block aus der Rubrik Operatoren

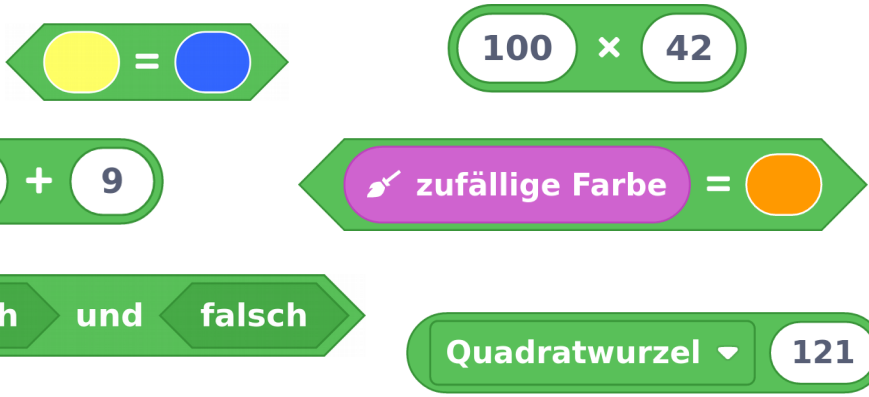
Schleifen

Thema: Schleifen
Bereich: Wissen
Anspruch: ★★☆☆
Zeitbedarf: 30 min

Dein Programm sollte jetzt in etwa so aussehen:



Aufgabe 9: Jetzt programmieren wir eine **Profi-Reset-Funktion**! Der Alarm soll nur neu gestartet werden können, wenn man **beide Arme gleichzeitig** berührt. Ändere dein Programm an einer einzigen Stelle! Weißt du wo?



W3 Operatoren

Thema:	Operatoren
Bereich:	Wissen
Voraussetzung:	Station **W2**
Lernziele:	Bedeutung und Anwendung von Operatoren kennenlernen, mit Ganzzahl-Operatoren und dem Farb-Vergleichsoperator arbeiten, Wahrheitswerte kennenlernen und anwenden
Anspruch:	★★☆☆
Aufgaben:	A1 – A15
Zeitbedarf:	30 min

W3 Operatoren

Thema: Operatoren
Bereich: Wissen
Anspruch: ★★☆☆
Zeitbedarf: 30 min

In dieser Lernstation beschäftigen wir uns mit sogenannten **Operatoren**.

Operatoren kennst du schon aus dem Mathe-Unterricht: **Plus**, **Minus**, **Mal** und **Geteilt** sind Operatoren. Vielleicht hast du auch schon mal mit der **Wurzel** gerechnet. Bestimmt hast du auch schon mit den Vergleichsoperatoren **Größer**, **Kleiner** und **Gleich** gearbeitet.

Wir schauen uns mal ein paar Beispiele an:

Hier verbindet der **‚+' Operator** die beiden Zahlen ‚**1**' und ‚**9**' zum Ausdruck ‚**1+9**'.
Der Ausdruck hat den Wert ‚**10**'.

Hier verbindet der **‚Quadratwurzel' Operator** die Zahl ‚**121**' zum Ausdruck ‚**√121**'.
Der Ausdruck hat den Wert ‚**11**'.

Hier verbindet der **‚<' Operator** die beiden Zahlen ‚**3**' und ‚**200**' zum Ausdruck ‚**3<200**'.
Der Ausdruck hat den Wert ‚**wahr**'.

Hier verbindet der **‚>' Operator** die beiden Zahlen ‚**3**' und ‚**200**' zum Ausdruck ‚**3>200**'.
Der Ausdruck hat den Wert ‚**falsch**'.

Hier verbindet der **‚und' Operator** die beiden Wahrheitswerte ‚**falsch**' und ‚**falsch**' zum Ausdruck ‚**falsch und falsch**'.
Der Ausdruck hat den Wert ‚**falsch**'.

Hier verbindet der **‚=' Operator** die beiden Farbwerte ‚**gelb**' und ‚**blau**' zum Ausdruck ‚**gelb=blau**'.
Der Ausdruck hat den Wert ‚**falsch**'.

W3 Operatoren

Thema: Operatoren
Bereich: Wissen
Anspruch: ★★☆☆
Zeitbedarf: 30 min

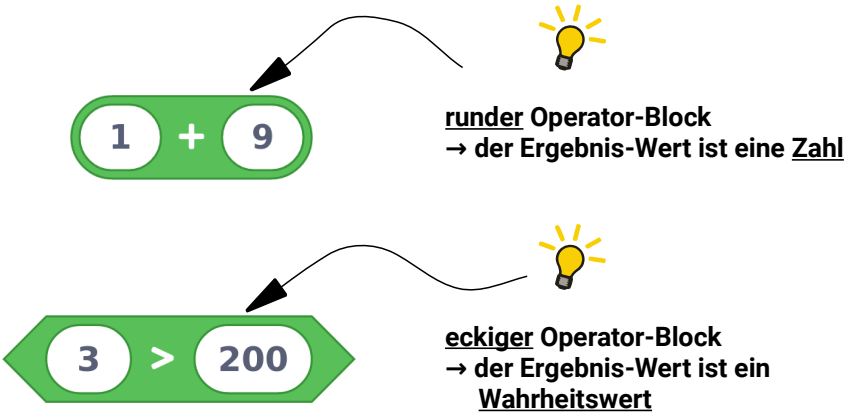
Wissensbox

Operator

Mit einem Operator lassen sich Konstanten und Variablen (also z.B. Zahlen) zu einem *Ausdruck* verbinden.

Rund oder sechseckig?

Dir ist bestimmt schon aufgefallen, dass einige der grünen Operator-Blöcke **rund** und die anderen **eckig** sind. Runde Operator-Blöcke liefern als Ergebnis eine Zahl, eckige Operator-Blöcke liefern als Ergebnis einen Wahrheitswert.



W3 Operatoren

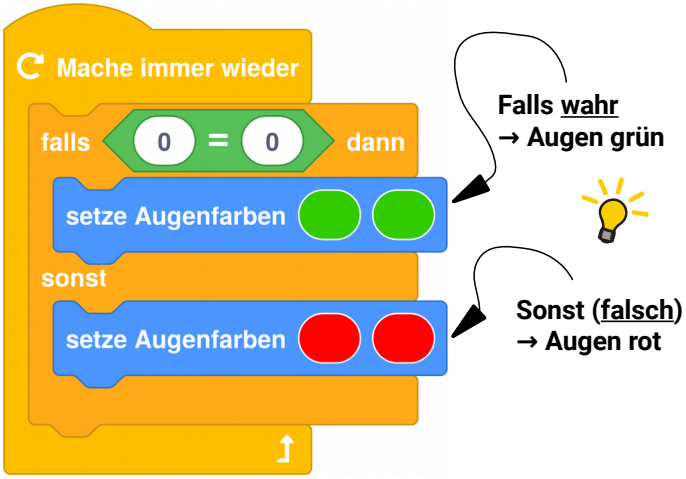
Thema: Operatoren
Bereich: Wissen
Anspruch: ★★☆☆
Zeitbedarf: 30 min

BOB3 als Logik-Meister!

Jetzt soll BOB3 seine **Rechenkünste** unter Beweis stellen und **wahr** und **falsch** erkennen:

Wir geben einen Ausdruck als Bedingung vor. Falls der Wert des Ausdrucks **wahr** ist, sollen die Augen **grün** leuchten. Im anderen Fall, also wenn der Ausdruck **falsch** ist, sollen die Augen **rot** leuchten!

Aufgabe 1: Programmiere das folgende Programm. Verwende einen «*falls dann*»-Block und **erweitere** ihn zu einem «*falls dann sonst*»-Block. Dann baue einen grünen ‚=' Operator Block und zwei «*setze Augenfarben*»-Blöcke ein. Teste dein Programm. Was macht der Bob?



Aufgabe 2: **Ändere** den Ausdruck in ‚**1 = 3**'. Was passiert jetzt?

Aufgabe 3: **Ändere** den Ausdruck nochmal: ‚**8 = 8**'. Was erwartest du jetzt?

Aufgabe 4: **Ändere** den Ausdruck in ‚**3 > 1**'. Was passiert jetzt?
Tip: Hierfür brauchst du einen anderen Operator Block!

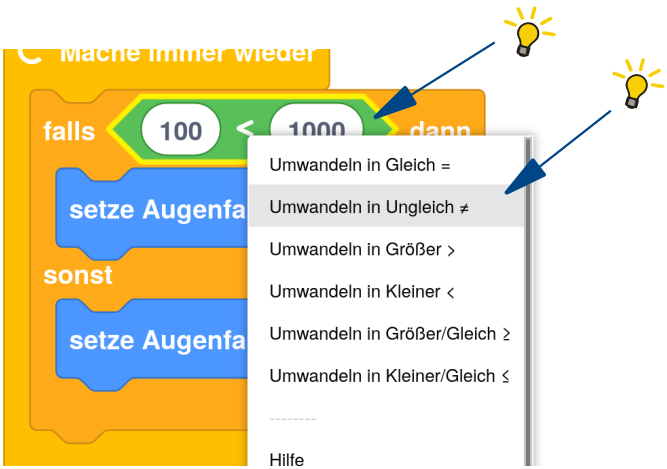
W3 Operatoren

Thema: Operatoren
Bereich: Wissen
Anspruch: ★★☆☆
Zeitbedarf: 30 min

Aufgabe 5: **Ändere** den Ausdruck in ‚**100 > 1000**'. Was erwartest Du jetzt?

Aufgabe 6: **Ändere** den Ausdruck nochmal: ‚**100 < 1000**'. Was sagt der Bob dazu?

Man kann die Blöcke auch **umwandeln**! Dafür klickt man mit der rechten Maustaste auf den Block und wählt aus dem Dropdown-Menü einen anderen Operator aus:



Aufgabe 7: **Ändere** den Ausdruck in ‚**100 ≠ 1000**'. Was passiert jetzt?

Aufgabe 8: Welche der folgenden Ausdrücke sind wahr und welche sind falsch?

a) 5 < 4

b) 5 ≠ 4

c) 5 ≠ 5

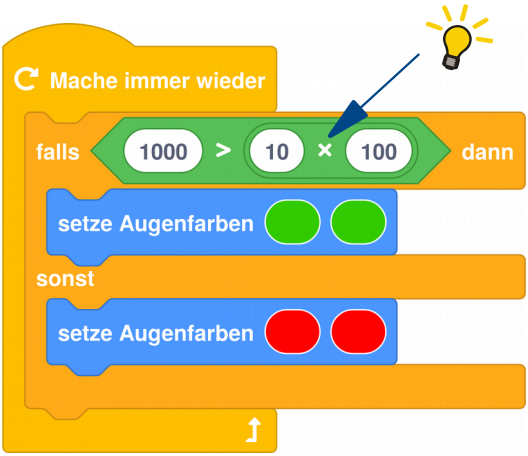
d) 4 > 5

W3 Operatoren

Thema: Operatoren
Bereich: Wissen
Anspruch: ★★☆☆
Zeitbedarf: 30 min

Jetzt wollen wir mal mehrere Operator-Blöcke miteinander **kombinieren**! Wir können BOB3 eine **Rechenaufgabe** stellen: Er soll ausrechnen, ob die Zahl **1000** größer ist als das Ergebnis von ‚**10 × 100**'. Was meinst du, kann er das?

Aufgabe 9: **Ändere** dein Programm: **Kombiniere** einen ‚**>**' Block mit einem ‚**×**' Block und ändere die Zahlen! Teste dein Programm. Was sagt Bob dazu? Kann er rechnen??



Aufgabe 10: **Ändere** den ‚**>**' Operator in einen ‚**=**' Operator. Was sagt Bob jetzt?

Aufgabe 11: Welche der folgenden Ausdrücke sind wahr und welche sind falsch?

a) 9 = 6

b) 538 = 539

c) 538 = 5 + 38

d) 600 − 62 = 538

W3 Operatoren

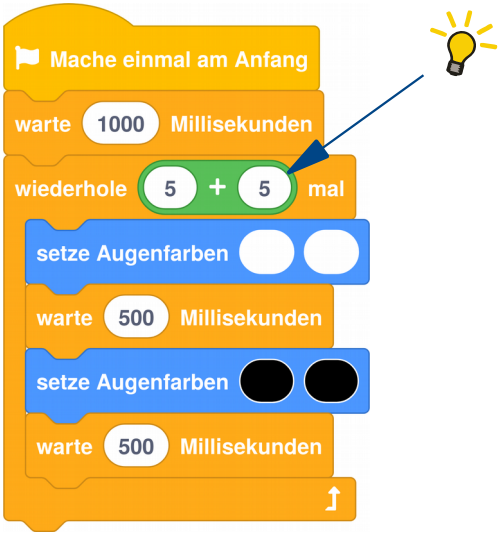
Thema: Operatoren
Bereich: Wissen
Anspruch: ★★☆☆
Zeitbedarf: 30 min

Jetzt programmieren wir ein ganz neues Programm!

Du kennst ja schon die «*wiederhole x-mal*»-Schleife. Mit dieser Schleife stellen wir Bob jetzt eine Rechenaufgabe und wir überprüfen, ob er richtig rechnet!

Aufgabe 12: Programmiere das folgende Programm: Wir stellen Bob eine Rechenaufgabe, er rechnet das Ergebnis aus und blinkt uns sein Ergebnis zu. Teste das Programm und zähle mit, wie oft Bob blinkt! Kann er gut rechnen?

!! Verwende einen «***Mache einmal am Anfang***»-Block:



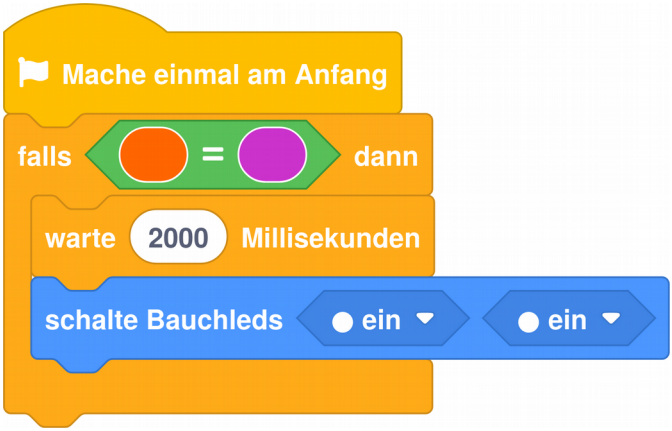
Aufgabe 13: **Ersetze** den grünen Ausdruck durch den folgenden. Was erwartest Du? Welches Ergebnis rechnet Bob aus?

W3 Operatoren

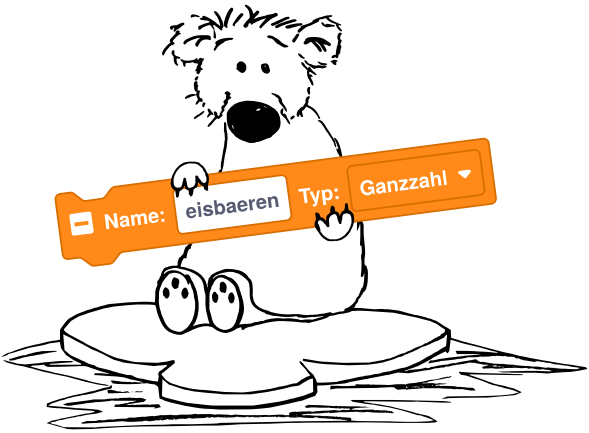
Thema: Operatoren
Bereich: Wissen
Anspruch: ★★☆☆
Zeitbedarf: 30 min

Wir können auch **Farben** vergleichen! Das probieren wir jetzt mal aus, dafür schreiben wir ein neues Programm:

Aufgabe 14: **Programmiere das folgende Programm:** Verwende dazu einen «*Mache einmal am Anfang*»-Block, einen «*falls dann*»-Block, einen «*warte*»-Block und einen «*schalte Bauchleds*»-Block. Als **Bedingung** bauen wir einen grünen Operator-Block ein, der **zwei Farben** miteinander **vergleichen** kann!
Wir geben Bob die Aufgabe, zwei Farben auf Gleichheit zu prüfen, er soll kurz überlegen und dann, **falls** die Farben gleich sind, beide **Bauchleds einschalten**.
Teste das Programm:



Aufgabe 15: Wie musst du die Bedingung verändern, damit die beiden Bauchleds eingeschaltet werden?



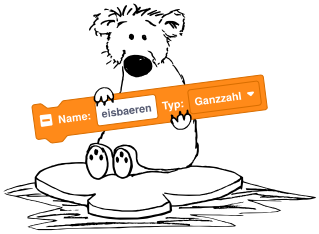
W4 Variablen

Thema:	Variablen
Bereich:	Wissen
Voraussetzung:	Station **W3**
Lernziele:	Bedeutung und Anwendung von Variablen kennenlernen, eine Ganzzahl-Variable deklarieren, initialisieren und im Programm verwenden, Zuweisung von neuen Werten verstehen und anwenden
Anspruch:	★★★☆
Aufgaben:	A1 – A8
Differenzierung:	A9
Zeitbedarf:	30 min

W4 Variablen

Thema: Variablen
Bereich: Wissen
Anspruch: ★★★☆
Zeitbedarf: 30 min

In dieser Lernstation beschäftigen wir uns mit **Variablen** und mit **Eisbären!**

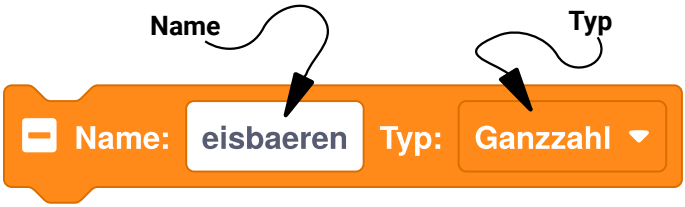


BOB3 zählt Eisbären im Zoo:

BOB3 ist im Zoo und soll Eisbären zählen. Im Gehege gibt es 4 Eisbären, aber manchmal sind weniger oder gar keine zu sehen, weil sie sich verstecken!

Falls Bob **einen** Bären sieht, dann soll **ein Auge** weiß leuchten, bei **zwei** Bären sollen **beide Augen** weiß leuchten. Falls er **drei** Bären entdeckt, dann soll zusätzlich noch **eine Bauch-Led** weiß leuchten, bei **vier** Bären sollen **alle vier Leds** weiß leuchten! Falls weit und breit **kein Bär** zu sehen ist, soll **nix** leuchten!

Für unser Programm verwenden wir eine **Variable**, die wir ‚***eisbaeren***‘ *nennen*:



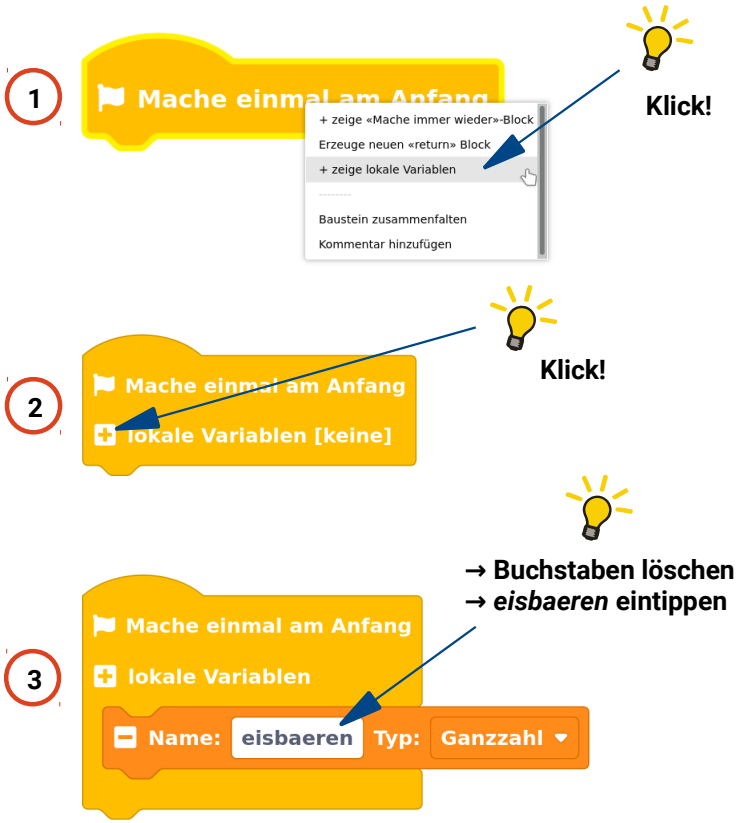
Eine **Variable** hat immer einen *Namen* und einen *Typ*. Unsere Variable soll die Anzahl der aktuell im Gehege zu sehenden Eisbären speichern, daher nennen wir sie ‚**eisbaeren**‘. Da es keine halben Eisbären gibt, nehmen wir als Typ für unsere Variable eine **Ganzzahl**. Variablen vom Typ ‚Ganzzahl‘ bekommen automatisch als Startwert eine ‚**0**‘.

- *Name*: eisbaeren
- *Typ*: Ganzzahl
- *Startwert*: 0

W4 Variablen

Thema: Variablen
Bereich: Wissen
Anspruch: ★★★☆
Zeitbedarf: 30 min

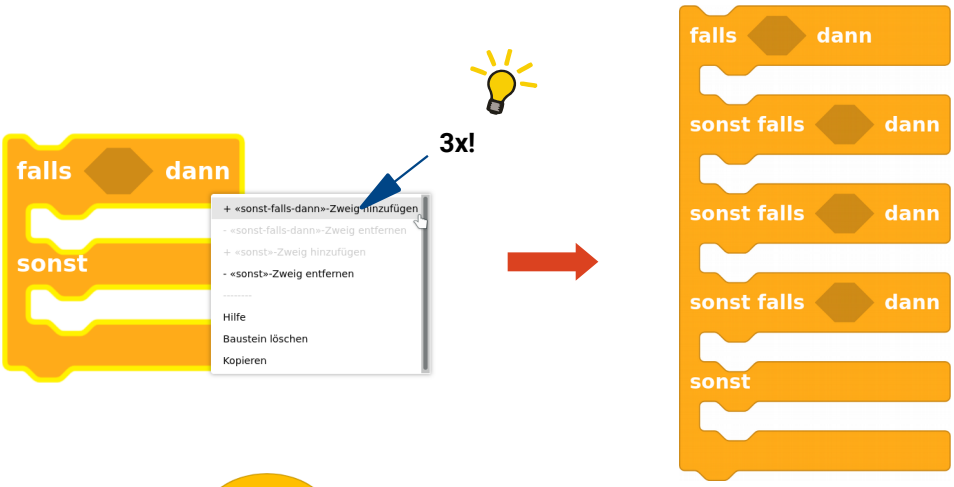
Aufgabe 1: Programmiere das folgende Programm. Verwende den «*Mache einmal am Anfang*»-Block.
Jetzt erzeugen wir uns eine **neue lokale Variable** mit dem Namen ‚eisbaeren', vom Typ ‚*Ganzzahl*'. Dafür klicke mit der rechten Maustaste auf den «*Mache einmal am Anfang*»-Block und klicke dann auf ‚**zeige lokale Variablen**'. Dann klicke auf das ‚**+**'. Anschließend ändern wir noch den Namen, klicke dazu in das Feld, lösche alle Buchstaben und tippe ‚eisbaeren' ein:



Variablen

Thema: Variablen
Bereich: Wissen
Anspruch: ★★★☆
Zeitbedarf: 30 min

Aufgabe 2: Zur Programmierung verwenden wir einen «***falls dann sonst***»-Block. **Erweitere** diesen Block um **drei** «***sonst falls dann***»-**Zweige** und baue ihn in dein Programm ein:

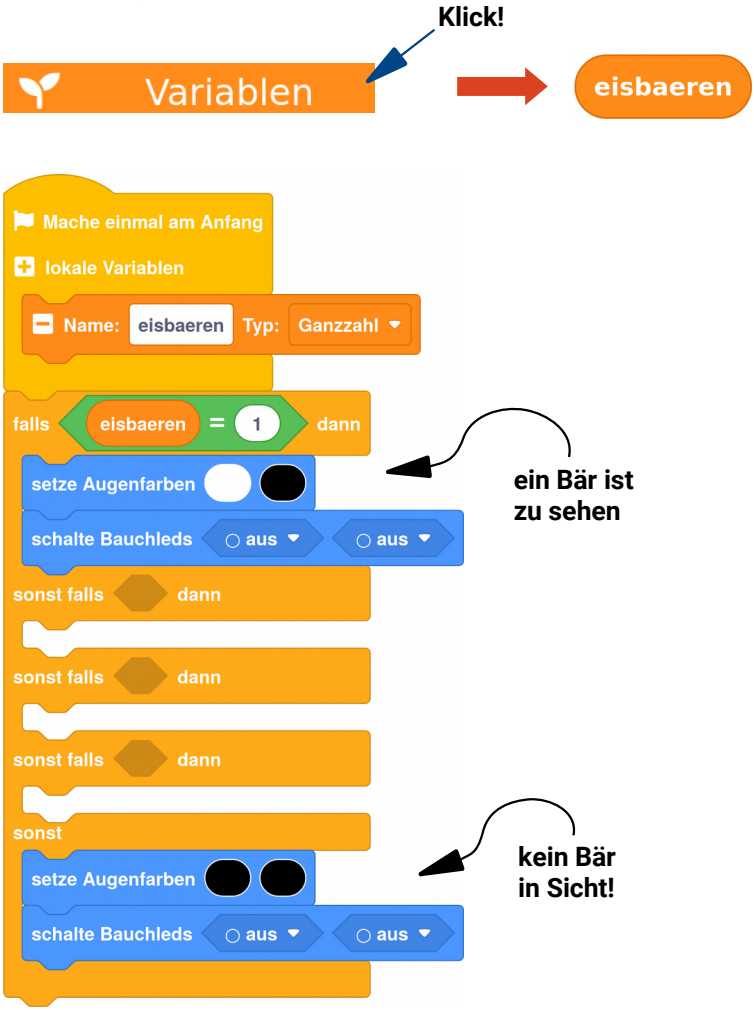


Variablen

Thema: Variablen
Bereich: Wissen
Anspruch: ★★★☆
Zeitbedarf: 30 min

Aufgabe 3:

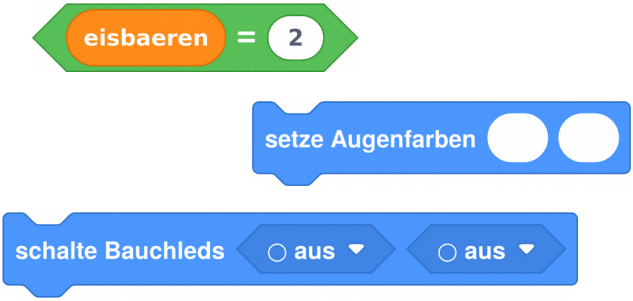
Wir starten mit den beiden Fällen, dass **ein Bär** oder **kein Bär** zu sehen ist. Programmiere den ersten «***falls dann***»-Zweig und den «***sonst***»-**Zweig**. Falls Bob **einen** Bären sieht, dann soll **ein Auge** weiß leuchten. Falls weit und breit **kein Bär** zu sehen ist, sollen **alle Leds aus** sein! Den **orangenen** Variablen-Block findest du in der Rubrik ‚Variablen':



W4 Variablen

Thema: Variablen
Bereich: Wissen
Anspruch: ★★★☆
Zeitbedarf: 30 min

Aufgabe 4: Jetzt programmieren wir den Fall, dass **zwei Bären** zu sehen sind. Baue dazu die folgenden Blöcke an den **richtigen Stellen** in dein Programm ein:



Aufgabe 5: Vervollständige dein Programm!
Programmiere die **beiden** letzten **Fälle**:

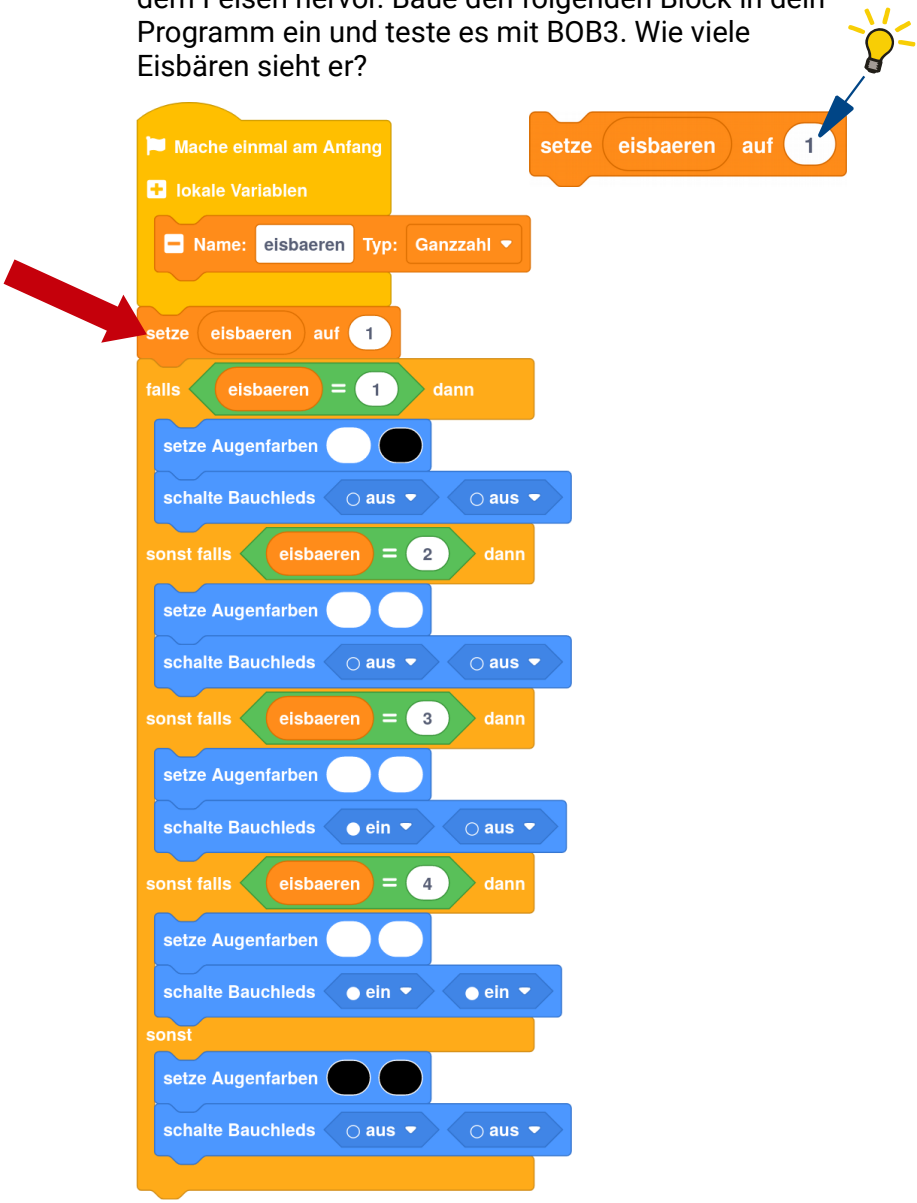
- **Drei Bären** in Sicht → beide Augen weiß einschalten und die linke Bauch-Led einschalten
- **Vier Bären** in Sicht → beide Augen weiß einschalten und beide Bauch-Leds einschalten

Wissensbox

Variable

Eine Variable ist ein **Speicherort** für *Zahlen*, *Farben* oder *sonstige Daten*.

Aufgabe 6: **Ui!!!! Ein kleiner Eisbär ist neugierig!** Er lugt hinter dem Felsen hervor. Baue den folgenden Block in dein Programm ein und teste es mit BOB3. Wie viele Eisbären sieht er?

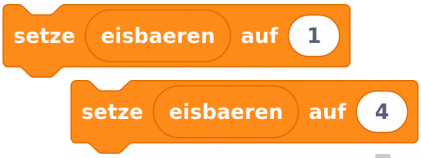


W4 Variablen

Thema: Variablen
Bereich: Wissen
Anspruch: ★★★☆
Zeitbedarf: 30 min

Aufgabe 7: **Fütterung!!!! Alle vier Eisbären kommen hervor!**
Ändere den Wert deiner Variablen ‚eisbaeren' auf ‚4' und probiere, ob BOB3 richtig zählen kann!

Aufgabe 8: **Papa Eisbär ist satt und verschwindet in seiner Höhle!**
Ändere den Wert der Variablen ‚eisbaeren' in deinem Programm und teste es mit BOB3.
Wie viele Bären zählt er jetzt?



Wissensbox

Variablen sind variabel

Da Variablen variabel sind, kann man den jeweils gespeicherten Wert beliebig ändern. Der Datentyp muss dabei immer gleich bleiben.

Aufgabe 9: **Programmiere ein neues Programm:**
Verwende den «*Mache einmal am Anfang*»-Block und erzeuge eine neue lokale Variable. Gib deiner Variablen den Namen ‚**pinsel**' und ändere den Typ auf ‚**Farbe**'.
Jetzt setze ‚pinsel' auf die Farbe Türkis.
Dann verwende deine Variable ‚pinsel', um beide Augen türkis einzuschalten!

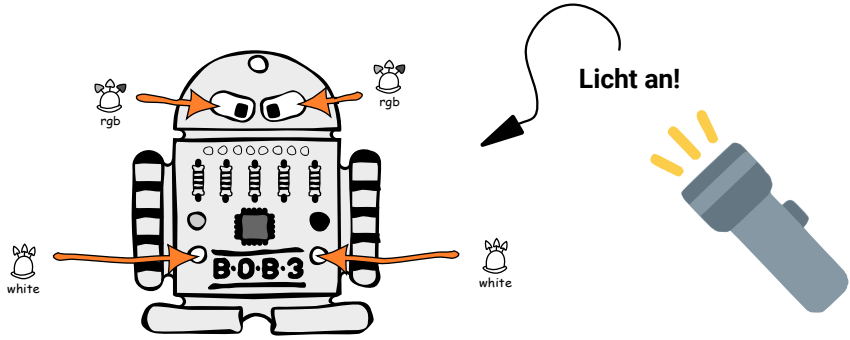
E1 Taschenlampe

Thema:	Mehrfachverzweigung
Bereich:	Experimente
Voraussetzung:	Station **W1**
Lernziele:	Implementation einer funktionsfähigen Taschenlampe mit Ein- und Ausschalter, Erweiterung als intelligente Taschenlampe mit unterschiedlichen Helligkeitsstufen
Anspruch:	★☆☆☆
Aufgaben:	A1 – A12
Zeitbedarf:	30 min

E1 Taschenlampe

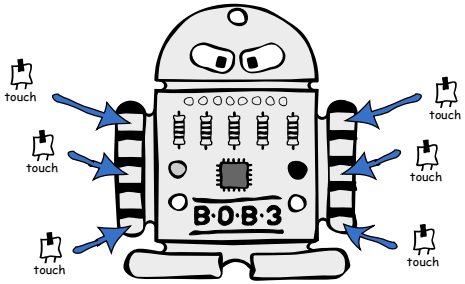
Thema: Verzweigung
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

In dieser Lernstation programmieren wir BOB3 als **Taschenlampe**. Dabei verwenden wir Arm 1 als *Einschalter*, Arm 2 als *Ausschalter* und alle vier LEDs von Bob als *Beleuchtung*!

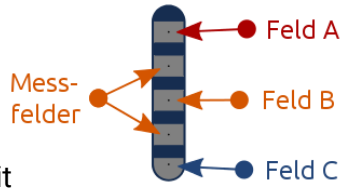


Wie funktionieren die Armsensoren von BOB3?

Wir wollen mit einer Berührung von Arm 1 alle LEDs einschalten und mit einer Berührung von Arm 2 alle LEDs wieder ausschalten. Woher weiss der Bob denn eigentlich, ob sein Arm berührt wird und wenn ja, welcher?



Beide Arme von BOB3 sind **Touch-Sensoren**. Die Arme „merken“ also, ob sie berührt werden, oder nicht! Weil Bob sogar bemerkt, **wo** du den jeweiligen Arm berührst (oben, mitte, unten) sind es **Multifeld-Touch-Sensoren**.



Jeder Arm hat 5 Metall-Felder:
→ 3 Aktivierungsfelder (A, B, C)
→ 2 Messfelder
Sobald du ein Aktivierungsfeld **gleichzeitig** mit einem Messfeld berührst, bekommt Bob ein Signal, ob Feld A, Feld B oder Feld C berührt wurde.

E1 Taschenlampe

Thema: Verzweigung
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

Aufgabe 1: **Jetzt programmieren wir Bob als Taschenlampe!** Starte auf einer neuen Arbeitsfläche mit einem leeren «*Mache immer wieder*»-Block und baue einen «*falls dann*»-Block ein. *Falls* wir **Arm 1** irgendwo berühren, *dann* sollen **beide Augen weiß** leuchten. Verwende hierfür einen Armsensor Block und einen «*setze Augenfarben*»-Block. Teste dein Programm mit BOB3!

Unsere Taschenlampe wird jetzt also eingeschaltet, sobald Arm 1 **irgendwo** berührt wird. Das wollen wir ändern: Die Taschenlampe soll nur eingeschaltet werden, wenn Arm 1 **unten** berührt wird. Dazu müssen wir einen **Parameter** in unserem Armsensor Block ändern:

Aufgabe 2: Ändere den Parameter ‚**irgendwo**' in ‚**unten**' und probiere dein neues Programm aus. Welchen Unterschied stellst du fest?

E1 Taschenlampe

Thema: Verzweigung
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

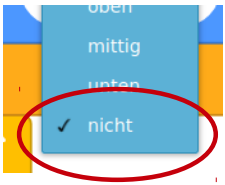
Wissensbox

Parameter

Ein Parameter ist eine Information/Vorgabewert, die/der beim Aufruf einer Funktion oder Anweisung dem Programm mit übergeben wird.

Aufgabe 3: Ändere den Parameter ‚**unten**' in ‚**oben**' und probiere dein neues Programm aus. Was passiert jetzt?

Aufgabe 4: Ändere den Parameter ‚**oben**' jetzt in ‚**nicht**' und **überlege** mit einer Mitschülerin oder einem Mitschüler, was das bedeutet! Welche Änderung erwartet ihr? Was macht Bob jetzt anders? Dann testet euer Programm!



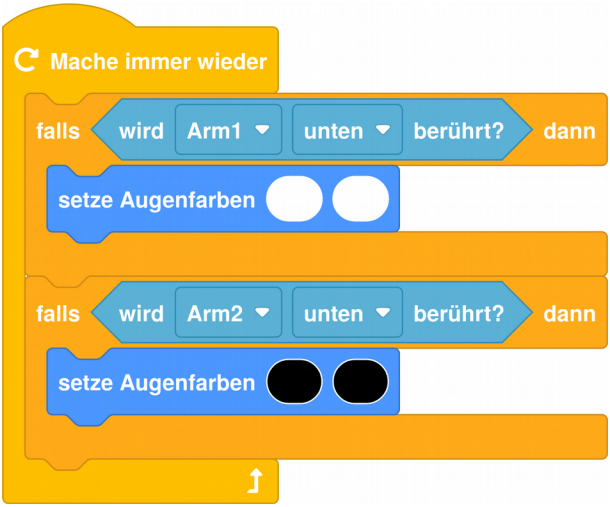
Aufgabe 5: Jetzt programmieren wir noch den **Ausschalter**! Ändere den Parameter ‚nicht' wieder in ‚unten' und erweitere dein Programm um einen **zweiten «*falls dann*»-Block**, so dass jetzt folgendes passiert:

Falls **Arm 1** unten berührt wird, werden beide Augen in Weiß eingeschaltet. Falls **Arm 2** unten berührt wird, werden beide Augen wieder ausgeschaltet. Probiere mal!

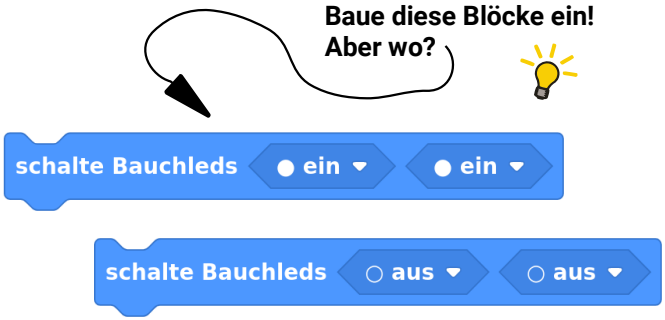
E1 Taschenlampe

Thema: Verzweigung
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

Dein Programm sollte jetzt in etwa so aussehen:



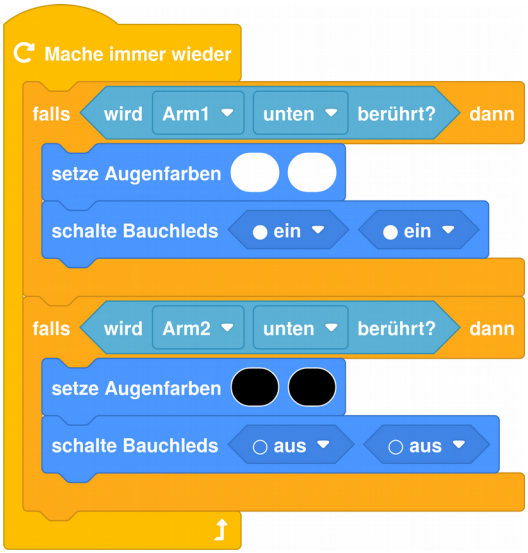
Aufgabe 6: **Wir brauchen mehr Licht!**
Erweitere dein Programm so, dass jetzt zusätzlich zu den beiden Augen auch noch die beiden Bauch-Leds eingeschaltet bzw. ausgeschaltet werden!



E1 Taschenlampe

Thema: Verzweigung
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

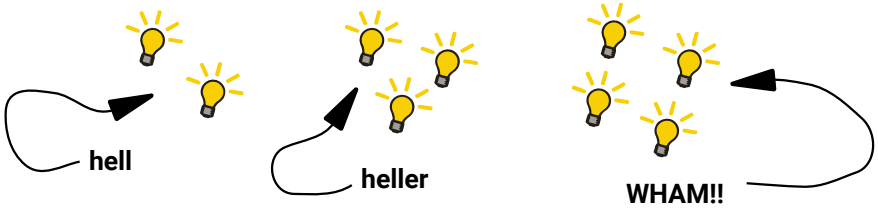
YUCHUU!! Unsere Taschenlampe ist fertig:



Weiter geht‘s mit Taschenlampe Nr. 2:

Jetzt programmieren wir eine neue, intelligente Taschenlampe. Wir verwenden den Arm 1 wie einen **Schieberegler**:

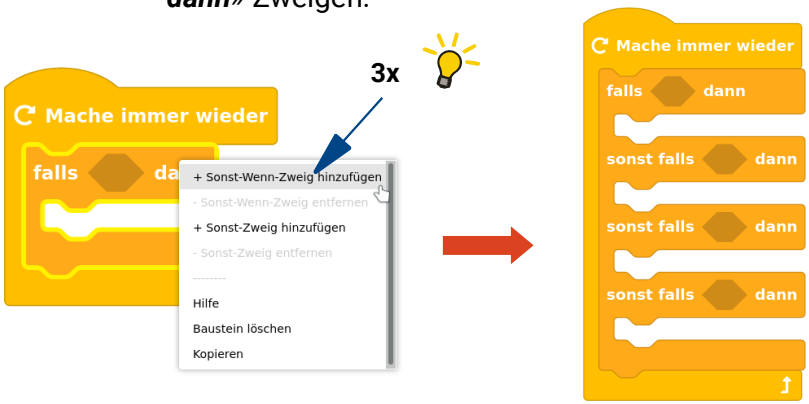
Wenn man Arm 1 *oben* berührt, gehen erstmal nur beide Augen in Weiß an. Damit die Taschenlampe **heller** leuchtet, berührt man den Arm 1 *mittig*, dann geht zusätzlich noch die Bauch-LED 3 an. Die **volle Helligkeitsstufe** wird erreicht, indem man Arm 1 *unten* berührt, dann hat Bob alle vier LEDs weiß an!



E1 Taschenlampe

Thema: Verzweigung
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

Aufgabe 7: **Programmiere die schlaue Taschenlampe!**
Starte auf einer **neuen** Arbeitsfläche mit einem leeren «*Mache immer wieder*»-Block. Füge einen «*falls dann*»-Block hinzu und **erweitere** diesen mit **drei** «***sonst falls dann***»-Zweigen:

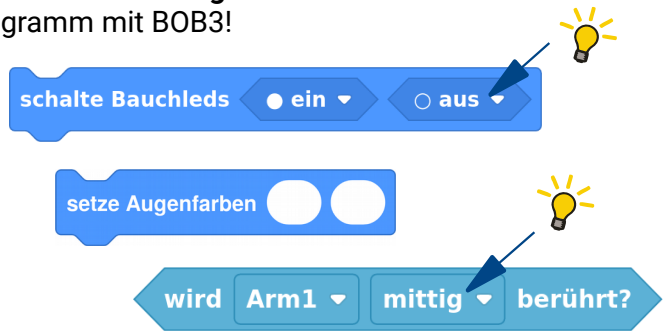


Aufgabe 8: Wir starten mit der **ersten Helligkeitsstufe**: Falls Arm 1 oben berührt wird, dann sollen beide Augen weiß eingeschaltet werden. Die beiden Bauch-Leds bleiben erstmal aus:

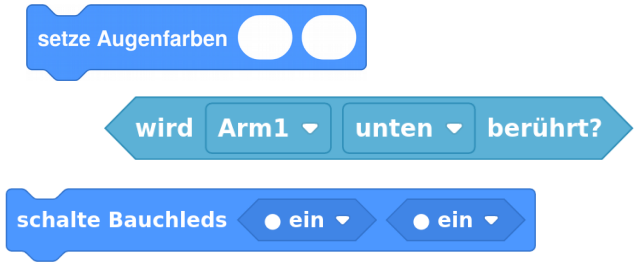
E1 Taschenlampe

Thema: Verzweigung
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

Aufgabe 9: Jetzt programmieren wir die **zweite Helligkeitsstufe**: Falls wir Arm 1 *mittig* berühren, dann soll **zusätzlich** die linke Bauch-Led eingeschaltet werden. Die zweite Bauch-Led bleibt erstmal aus. Baue die folgenden Blöcke **an der richtigen Stelle** ein und teste dein Programm mit BOB3!



Aufgabe 10: **Alle Lampen an!**
Erweitere dein Programm: Falls wir Arm 1 *unten* berühren, dann soll **zusätzlich** die rechte Bauch-Led eingeschaltet werden. Baue die folgenden Blöcke mit den richtigen Parametern **an der richtigen Stelle** ein und teste dein Programm mit BOB3!

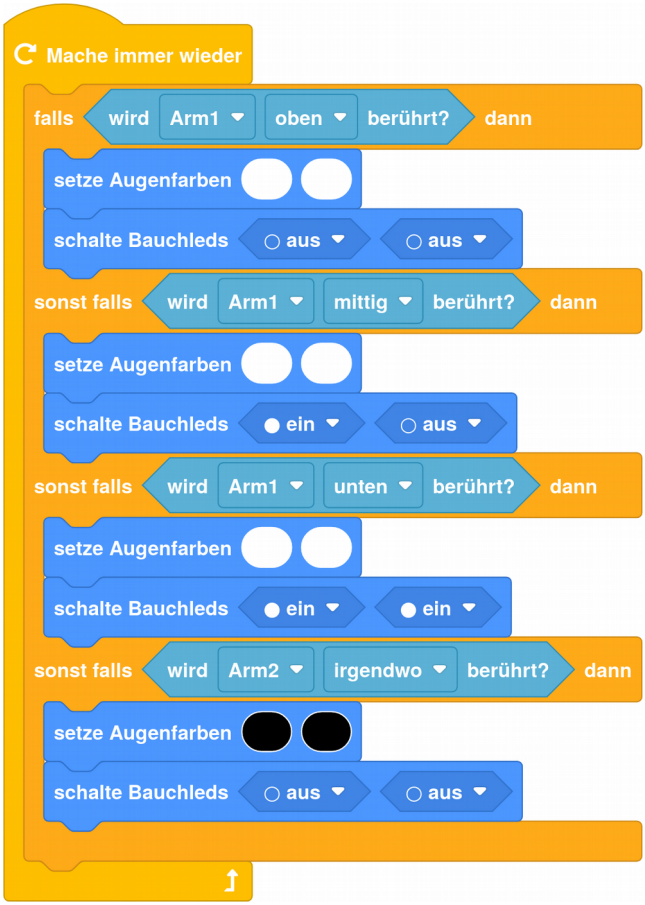


Aufgabe 11: Zum Schluss programmieren wir noch eine **Ausschaltfunktion**: Falls wir **Arm 2** *irgendwo* berühren, dann sollen **alle Leds ausgeschaltet** werden! Hast du eine Idee, wie das geht? Probiere mal!!

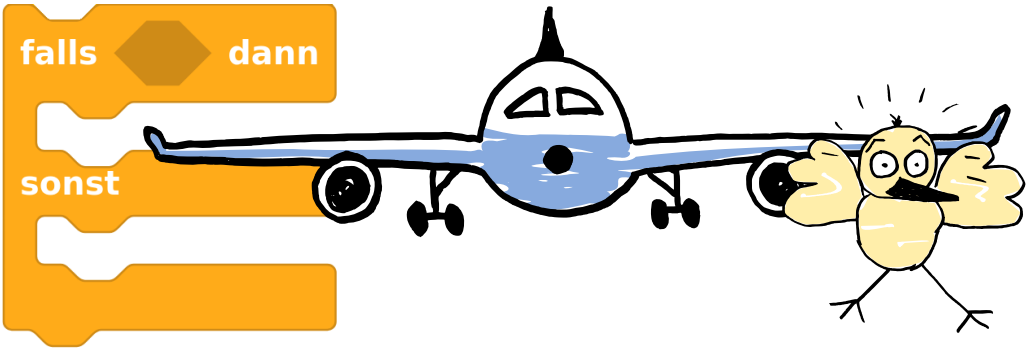
E1 Taschenlampe

Thema: Verzweigung
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 30 min

YUCHUU!! Unsere **Taschenlampe Nr. 2** ist fertig!! Dein Programm sollte jetzt in etwa so aussehen:



Aufgabe 12: Vergleiche deine Lösung und teste deine neue Taschenlampe mit BOB3! Am besten im Dunkeln :)

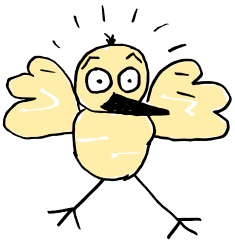


E2 Vogelwarnsystem

Thema:	Sensoren
Bereich:	Experimente
Voraussetzung:	Station **B2**
Lernziele:	Implementation einer Hinderniserkennung: Bob analysiert mit seinem IR-Sensor den Reflexionswert. Bei Überschreitung eines Grenzwertes aktiviert er ein Warnblitzlicht. Fallunterscheidung, Variable und Operator.
Anspruch:	★☆☆☆
Aufgaben:	A1 – A8
Differenzierung:	A9
Zeitbedarf:	40 min

E2 Vogelwarnsystem

Thema: Sensoren
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 40 min



Achtung, ein Vogel!

In dieser Lernstation programmieren wir BOB3 als Vogel-Warnsystem für Flugzeuge. Mit seinem **IR-Sensor** kann Bob berührungslos Objekte detektieren, also bemerken. Sobald ein Vogel (oder sogar ein ganzer Vogelschwarm) in die Nähe der Turbinen kommt, soll er die Augen in gelb einschalten und ein weißes Warnblitzlicht machen!

Aufgabe 1:
DER INFRAROTLICHT SENSOR

Um einen Vogel detektieren zu können, arbeiten wir mit Bob's **IR-Sensor**. Der Sensor besteht aus zwei Teilen: Einer *hellblauen* **IR-Sende-LED** und einem *schwarzen* **IR-Empfänger**.

Überlege mit einem Mitschüler/einer Mitschülerin, wo diese beiden Bauteile am Roboter zu finden sind.

Aufgabe 2:
DAS REFLEXIONS-VERFAHREN

Die **Detektion** funktioniert nach dem **Reflexionsverfahren**: Die IR-Sende-LED sendet IR-Licht aus, dieses trifft dann auf ein Hindernis, wird von dem Hindernis zurückreflektiert und dann von dem IR-Empfänger empfangen.

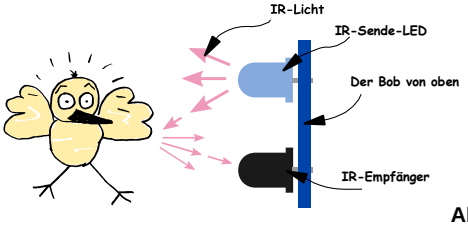


Abb. 1

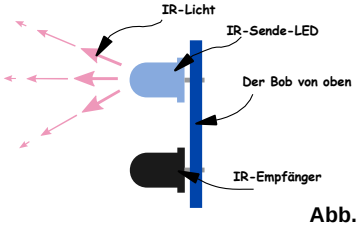


Abb. 2

Beschreibe, wie das Reflexionsverfahren funktioniert. Verwende dazu die Abbildungen 1 und 2.

E2 Vogelwarnsystem

Thema: Sensoren
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 40 min

Da unser Flugzeug fliegt, also immer in Bewegung ist, kann manchmal ein Vogel im Weg sein, manchmal aber auch nicht! Wir müssen also den IR-Sensor **immer wieder neu abfragen** und den jeweils aktuellen Wert auswerten, etwa so:

→ Frage an den Sensor: ‚Ist jetzt ein Vogel da?'

→ Frage an den Sensor: ‚Ist jetzt ein Vogel da?'

→ Frage an den Sensor: ‚Ist jetzt ein Vogel da?'

…

Die **Antwort** von unserem Sensor (ein Zahlenwert) speichern wir in einer **Variablen** ab, die wir ‚**vogeldetektor**' *nennen*:

vogeldetektor

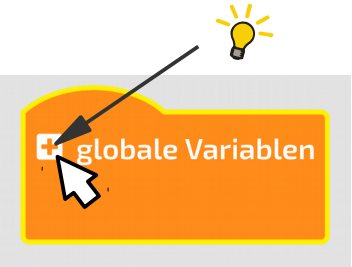
EINE VARIABLE

Eine **Variable** ist ein **Speicherort** für Zahlen, Zeichen oder sonstige Daten. Jede Variable hat einen *Namen* (den suchst du selber aus) und einen *Datentyp*.

Aufgabe 3:

DIE VARIABLE VOGELDETEKTOR

Klicke auf die Rubrik ‚**Variablen**' und erzeuge per Klick auf das Plus eine neue globale Variable. Dann ändere den Namen in ‚**vogeldetektor**'.

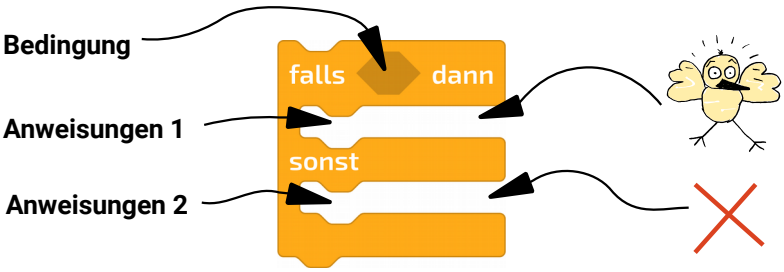


Aufgabe 4:
DER SENSORWERT

Jetzt speichern wir unseren **Sensorwert** ab. Verwende dazu aus der Rubrik ‚Variablen' einen ‚*setze … auf*'-Block, die neue Variable ‚*vogeldetektor*' und einen ‚*hole IR-Sensor-Wert*'-Block. Dein Programm sollte jetzt so aussehen:

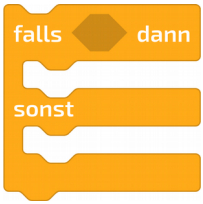
Manchmal ist **ein Vogel da** und manchmal ist **kein Vogel da**:

Für diese Fallunterscheidung braucht unser Programm eine spezielle Kontrollstruktur: einen ‚*falls-dann-sonst*'-*Block* als **Verzweigung**:



E2 Vogelwarnsystem

Thema: Sensoren
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 40 min

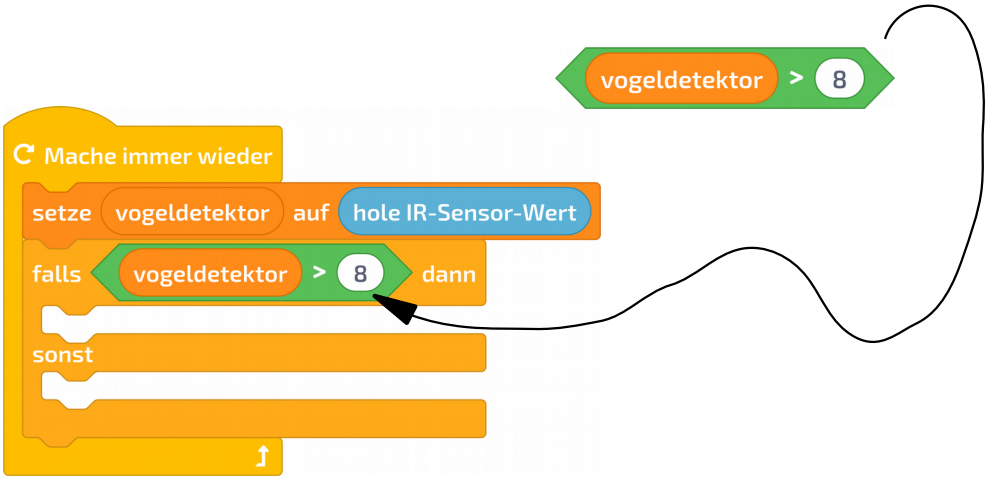


VERZWEIGUNG

Eine **Verzweigung** ermöglicht, dass in Abhängigkeit von einer **Bedingung** bestimmte Anweisungen ausgeführt werden und andere dagegen nicht! **Falls** die Bedingung **wahr** ist, dann werden die ***Anweisungen 1*** ausgeführt, **sonst**, also wenn die Bedingung **falsch** ist, werden die ***Anweisungen 2*** ausgeführt.

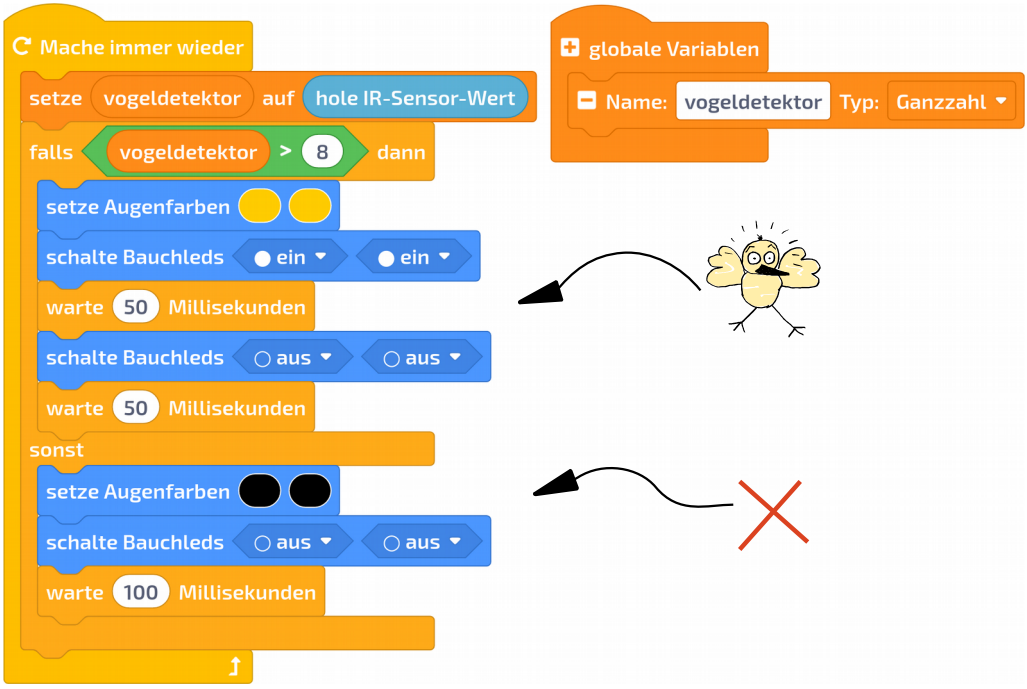
Aufgabe 5:
DIE BEDINGUNG

Bob soll warnen, **bevor** sich ein Vogel auf den Weg in die Turbine macht! Sobald der IR-Sensor einen Vogel detektiert, reagieren wir. Baue die folgende Bedingung in dein Programm ein:



Aufgabe 6:
DAS PROGRAMM

Ergänze dein Programm so, dass Bob im Falle einer Detektion die Augen-LEDs in gelb einschaltet und mit den Bauch-LEDs ein weißes Warnblitzlicht macht:



Aufgabe 7:
DER TEST

Teste dein Programm mit BOB3. Was passiert?

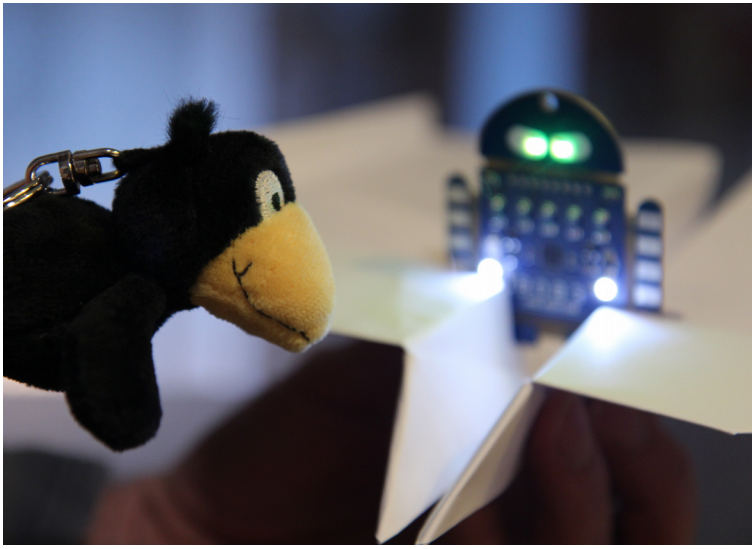
Tip: Der IR-Sensor verhält sich je nach Umgebungslicht unterschiedlich. Du kannst den Wert 8 aus der Bedingung auch mal variieren!

Aufgabe 8:

JETZT WIRD'S ERNST

Bastel dir einen Düsenjet aus Papier und befestige den Bob so, dass der IR-Sensor frei bleibt. Dann brauchen wir noch einen Vogel!!

Teste dein Programm mit BOB3. Was macht der Bob, wenn ein Vogel quert? Geht unser Warnblitzlicht an?



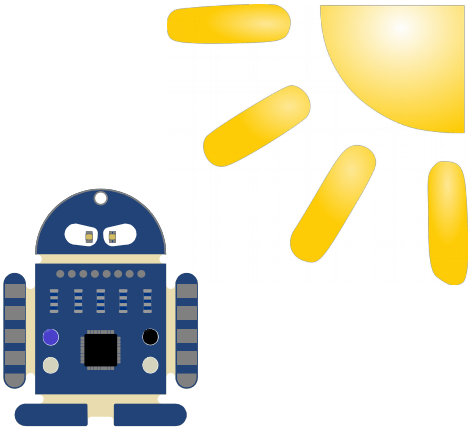
Aufgabe 9:

EXPERT MODE

Ändere dein Programm so, dass **beide Augen-LEDs** ganz schnell in **hellem gelb aufblitzen**, sobald der Sensor einen Vogel detektiert!

Teste dein Programm mit BOB3!

hole IR-Helligkeits-Wert



E3 Bob‘s smart-home

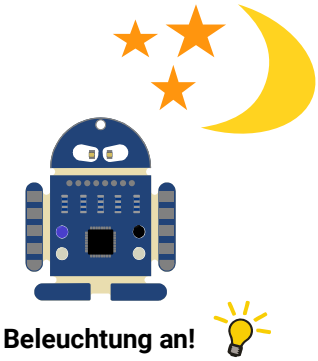
Thema:	Sensoren
Bereich:	Experimente
Voraussetzung:	Station **B2**
Lernziele:	Implementation einer automatischen Indoor Beleuchtung: Bob analysiert mit seinem IR-Sensor den Helligkeitswert der aktuellen Tageslichtsituation und schaltet ab einem Schwellwert automatisch alle Leds ein
Anspruch:	★☆☆☆
Aufgaben:	A1 – A3
Differenzierung:	A4
Zeitbedarf:	20 min

E3 Bob‘s smart-home

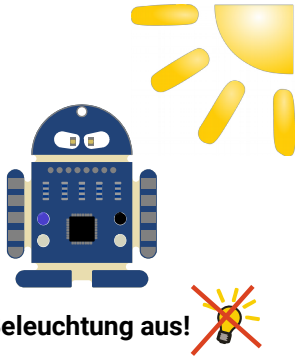
Thema: Sensoren
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 20 min

In dieser Lernstation programmieren wir BOB3 als Tageslicht-Sensor!

Bob möchte in seinem Zuhause Strom sparen. Das Licht soll automatisch eingeschaltet werden, wenn es dunkel ist und automatisch wieder ausgeschaltet werden, wenn es hell ist! Dazu muss Bob mit seiner IR-Sensorik detektieren, ob er sich im *Dunkeln* oder im *Tageslicht* befindet. Sobald der Sensor feststellt, dass es dunkel ist, soll er automatisch alle LEDs zur Beleuchtung weiß einschalten. Falls es wieder heller wird, sollen die LEDs automatisch wieder ausgeschaltet werden.



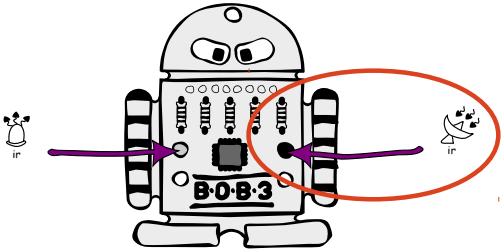
Beleuchtung an!



Beleuchtung aus!

Wie funktioniert der IR-Sensor?

BOB3 hat einen **IR-Sensor**, der aus zwei Teilen besteht: Einer durchsichtigen **IR-Sende-LED** und einem schwarzen **IR-Empfänger**. Die Abkürzung „IR“ steht für „Infrarot“. Infrarotlicht ist eine spezielle Lichtart.



Für unseren Einsatzzweck benötigen wir den schwarzen IR-Empfänger: Der Sensor bestimmt den jeweils **aktuellen IR-Lichtwert** und so weiß der Bob, ob es **hell** (viel IR-Licht) oder **dunkel** (wenig IR-Licht) ist und kann dann die Beleuchtung aus- bzw. einschalten!

E3 Bob‘s smart-home

Thema: Sensoren
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 20 min

Wissensbox

Licht

Es gibt verschiedene Lichtarten:

- Infrarotes-Licht (IR)
- Sichtbares Licht
- Ultraviolettes-Licht (UV)

Tageslicht und das Licht von **Glühlampen** besteht aus *sichtbarem Licht*, aus *IR-Licht* und aus *UV-Licht*. Das Licht von **LED-Lampen** und **Neonröhren** besteht nur aus *sichtbarem Licht*!

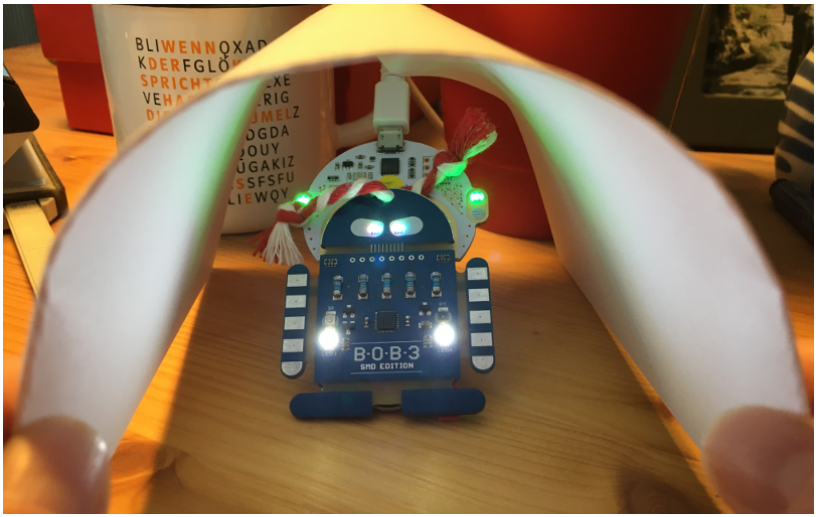
Bob ist ein Licht-Experte! Mit seinem IR-Empfänger weiß er genau, ob er im Dunkeln oder im Hellen ist. Das nutzen wir jetzt aus und programmieren eine Beleuchtung, die automatisch an- und auch wieder ausgeht:

Aufgabe 1: Verwende einen «*Mache immer wieder*»-Block und einen «*falls dann*»-Block. Falls der aktuell gemessene IR-Helligkeitswert kleiner ist als 12 (Schwellwert), dann sollen alle Leds eingeschaltet werden! Probiere das Programm mal aus! Was macht der Bob?

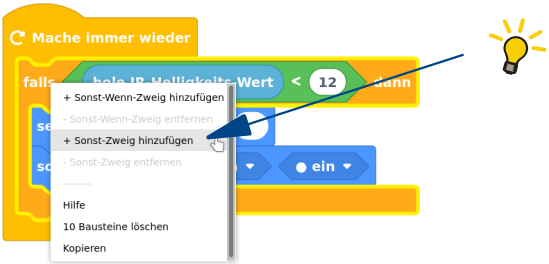
E3 Bob‘s smart-home

Thema: Sensoren
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 20 min

Teste dein Programm z.B. mit einem Zelt aus Papier: Was macht der Bob, wenn‘s dunkler wird? Gehen dann automatisch alle Lampen an?
Tip: Falls es nicht gut klappt, dann probiere mal andere Schwellwerte aus, z.B. 9, 10 oder 15.
Gehen die Lampen auch wieder aus, wenn es wieder heller wird?



Aufgabe 2: **Ergänze** dein Programm so, dass die Leds auch wieder automatisch **ausgeschaltet** werden, wenn es wieder heller ist. **Falls** der IR-Helligkeitswert kleiner als 12 ist, dann werden alle Leds **eingeschaltet** und **sonst** werden alle Leds **ausgeschaltet**. Füge dem «*falls dann*»-Block einen «***sonst***»-Zweig und die benötigten Blöcke zum Ausschalten aller Leds hinzu! Probiere mal!



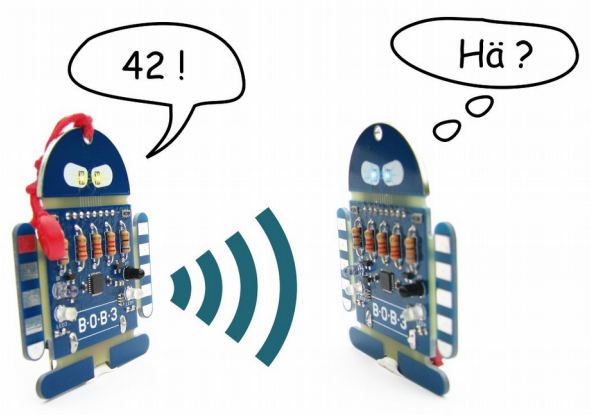
E3 Bob‘s smart-home

Thema: Sensoren
Bereich: Experimente
Anspruch: ★☆☆☆
Zeitbedarf: 20 min

YUCHUU!! Unsere **automatische Zimmer Beleuchtung** ist fertig!! Dein Programm sollte jetzt in etwa so aussehen:

Aufgabe 3: In der Lösung ist noch ein ‚**warte 100 Millisekunden**‘ Block eingebaut, damit der IR-Sensor nur alle 100 Millisekunden abgefragt wird. Damit wird ein Flackern der Leds verhindert. Baue den Block mal in dein Programm ein und teste es!

Aufgabe 4: Was erwartest du, wenn du den **Schwellwert** auf **3** setzt? Was bedeutet das? Welchen Unterschied beim Ausprobieren stellst du fest?
Hat der Bob jetzt Licht in seinem Zelt? ;-)



E4 Alice und Bob

Thema:	Kommunikation
Bereich:	Experimente
Voraussetzung:	Station **W1**
Lernziele:	Implementation einer Infrarot-Datenübertragung zwischen zwei Robotereinheiten
Anspruch:	★★☆☆
Aufgaben:	A1 – A10
Zeitbedarf:	30 min

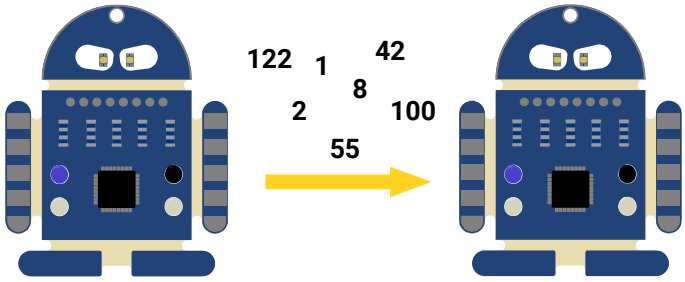
E4 Alice und Bob

Thema: Kommunikation
Bereich: Experimente
Anspruch: ★★☆☆
Zeitbedarf: 30 min

Hey Du ! Wer ich ?? Ja Du !!! In dieser Lernstation programmieren wir zwei BOB3 Roboter so, dass sie miteinander kommunizieren können!

Vorbereitung:

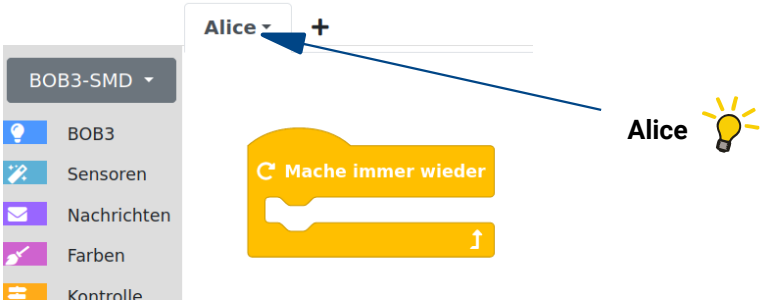
Suche dir eine Mitschülerin oder einen Mitschüler aus und bildet ein **2-er Team**. Jetzt müsst ihr euch überlegen, welcher Bob der **Sender** der Botschaft und welcher der **Empfänger** der Botschaft sein soll. Den Sende-Roboter nennen wir ‚**Alice**' und den Empfänger-Roboter nennen wir ‚**Bob**':



Sender (Alice) **Empfänger** (Bob)

Aufgabe 1: **Wir schreiben zuerst das Programm für Alice!**

Startet auf einer neuen Arbeitsfläche mit einem leeren «*Mache immer wieder*»-Block und gebt eurem Programm den Namen ‚**Alice**':



E4 Alice und Bob

Thema: Kommunikation
Bereich: Experimente
Anspruch: ★★☆☆
Zeitbedarf: 30 min

Alice soll eine Zahl senden. Dazu verwenden wir einen «***sende Wert***»-Block aus der Rubrik ‚**Nachrichten**‘:

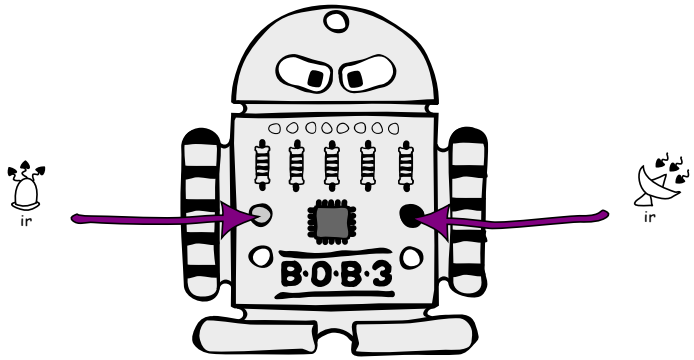
Wissensbox

«*sende Wert*»-Block

Der «*sende Wert*»-Block kann eine beliebige Zahl zwischen 0 und 255 senden.

Die Übertragung funktioniert mit den IR-Sensoren der Bobs:
Die durchsichtige/bläuliche **IR-LED** kann Botschaften **senden** und der schwarze **Phototransistor** kann Botschaften **empfangen**.

Aufgabe 2: Schaut mal, ob ihr bei euren Bobs die IR-Sende-Led und den schwarzen Phototransistor als IR-Empfänger entdecken könnt!

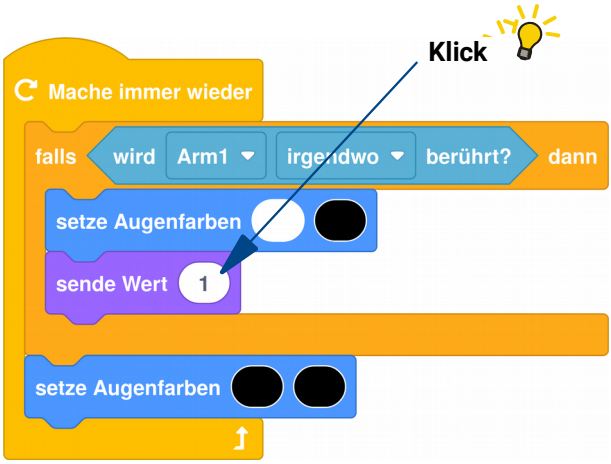


Zur Erinnerung:
Die IR-Sensoren reagieren empfindlich auf verschiedene Lichtarten. Zuviel Tageslicht, Glühlampen-Licht oder LED-Licht kann die Sensorik stören!

E4 Alice und Bob

Thema: Kommunikation
Bereich: Experimente
Anspruch: ★★☆☆
Zeitbedarf: 30 min

Aufgabe 3: **Programmiert das folgende Programm für Alice:** Falls Arm 1 irgendwo berührt wird, dann schalten wir Auge 1 in weiß an und senden die Zahl 1. Wenn der Arm nicht mehr berührt wird, sollen beide Augen aus sein. Verwendet einen «*falls dann*»-Block, einen Sensor-Block für den Armsensor, zwei «*setze Augenfarben*»-Blöcke und einen «*sende Wert*»-Block und ändert die ‚0' in eine ‚1':



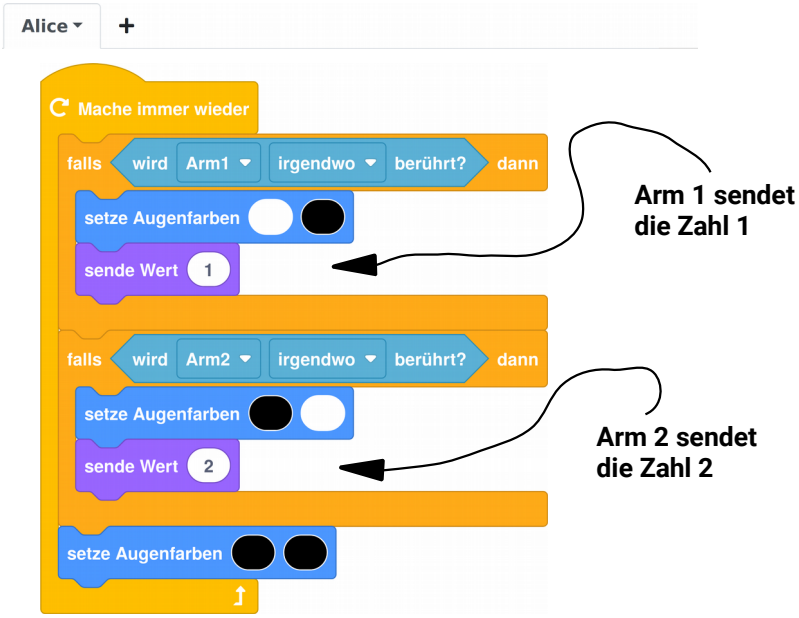
Aufgabe 4: **Erweitert** euer Programm um einen **zweiten** «*falls dann*»-Block**:** Falls Arm 2 irgendwo berührt wird, dann schalten wir Auge 2 in Weiß an und senden die Zahl 2.

Baut den neuen «*falls dann*»-Block an der **richtigen Stelle** ein und testet euer Programm auf Alice!

E4 Alice und Bob

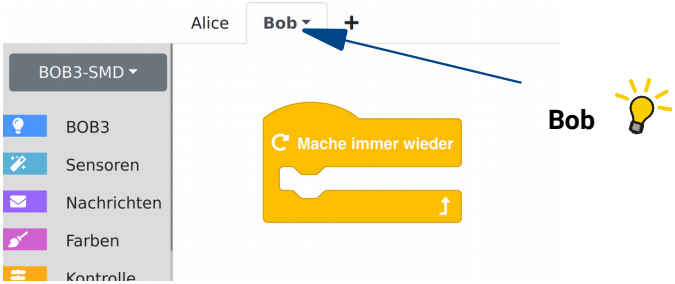
Thema: Kommunikation
Bereich: Experimente
Anspruch: ★★☆☆
Zeitbedarf: 30 min

Euer Programm für **Alice** sollte jetzt in etwa so aussehen:



Aufgabe 5: **Jetzt schreiben wir das Programm für Bob!**

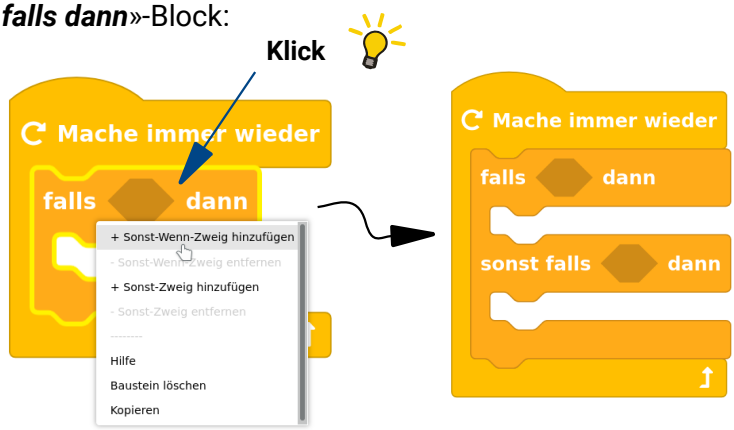
Startet auf einer **neuen Arbeitsfläche** mit einem leeren «*Mache immer wieder*»-Block und gebt eurem Programm den Namen ‚**Bob**‘:



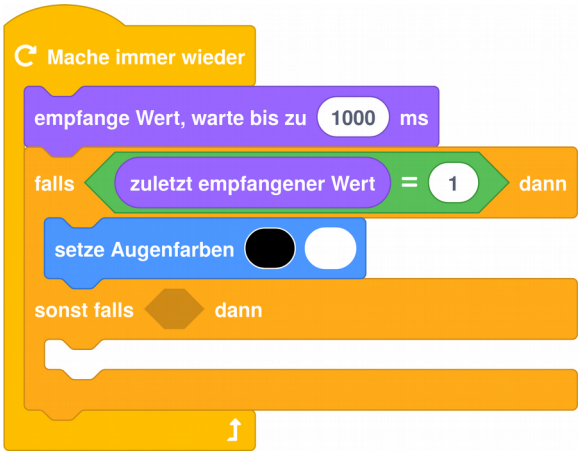
E4 Alice und Bob

Thema: Kommunikation
Bereich: Experimente
Anspruch: ★★☆☆
Zeitbedarf: 30 min

Aufgabe 6: Verwendet einen «*falls dann*»-Block und erweitert diesen per Rechts-Klick zu einem «***falls dann – sonst falls dann***»-Block:



Aufgabe 7: Programmiert das folgende Programm! Verwendet dazu Blöcke aus den Rubriken ‚**Nachrichten**', ‚**Operatoren**' und ‚**BOB3**':

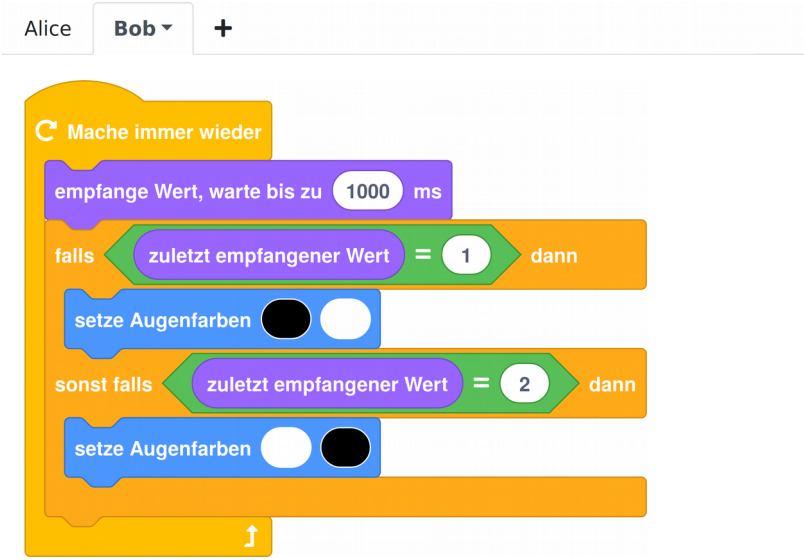


Aufgabe 8: **Vervollständigt euer Programm**: Wenn der zuletzt empfangene Wert eine 2 ist, dann soll das andere Auge weiß leuchten!

E4 Alice und Bob

Thema: Kommunikation
Bereich: Experimente
Anspruch: ★★☆☆
Zeitbedarf: 30 min

Euer Programm für **Bob** sollte jetzt in etwa so aussehen:



Aufgabe 9: Übertragt das Programm für Alice auf Alice und das Programm für Bob auf Bob!

Aufgabe 10: Testet euer Programm! Klappt es? Hat Bob die Nachricht bekommen?