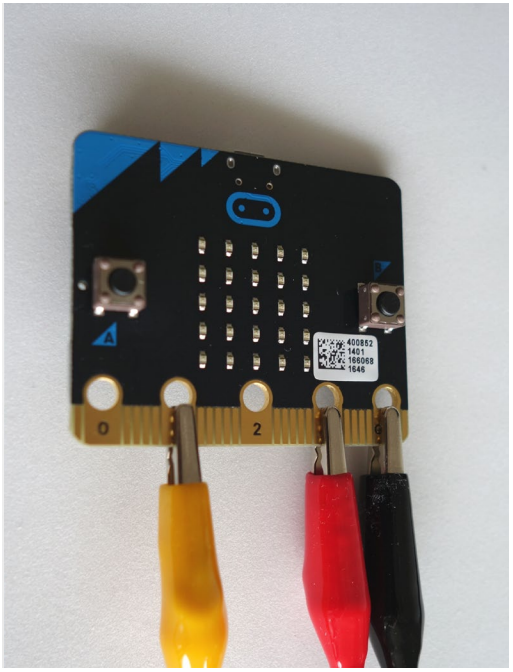
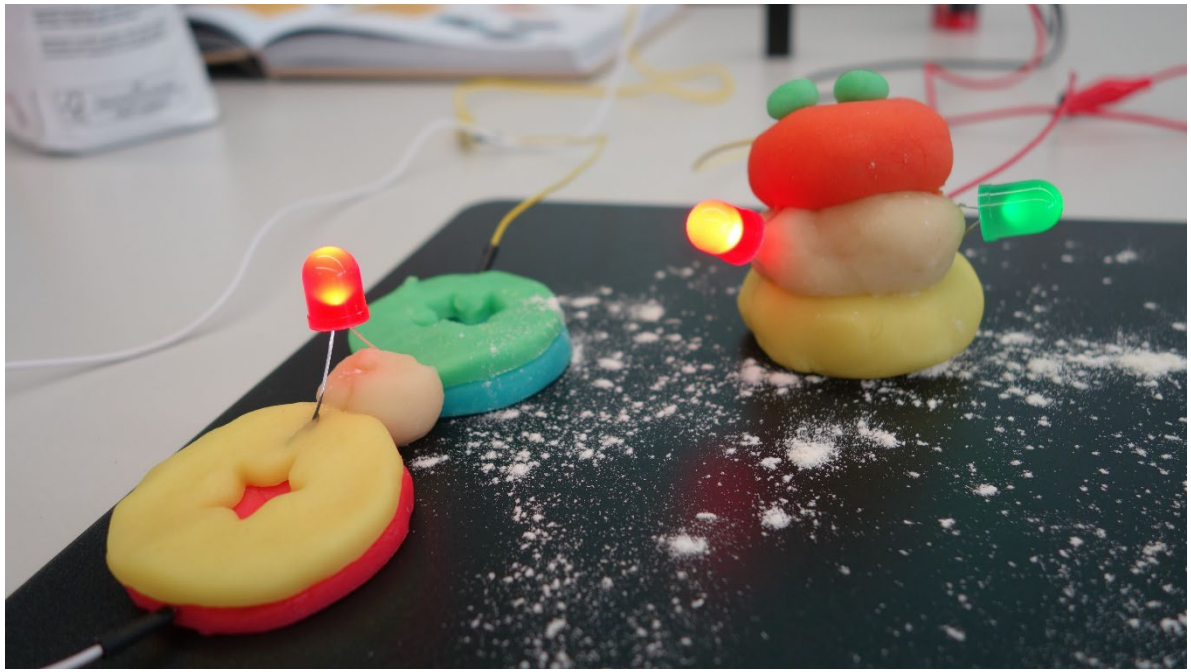
Physical Computing – Verbindung der physischen mit der virtuellen Welt





Kursüberblick

Kurstag 1 (Mittwoch, 14.11.18, 14.00 - 17.00 Uhr)

- Einführung in Physical Computing
- Micro:bit Kit
- Micro:bit Challenge Cards

Kurstag 2 (Mittwoch, 5.12.18, 14.00 – 17.00 Uhr)

- Überblick über andere Physical Computing Plattformen
- Vertiefte Themen (Fehlersuche, Umsetzung eines Projekts etc.)
- Didaktische Hinweise

Optional: Für die Wochen zwischen den Kurstagen kann das Micro:bit Kit für die Weiterarbeit mit nach Hause genommen werden.

Micro:bit Kit

- **Basic Kit (CHF 50,-)**
- **Komplettes Kit (CHF 120,-)**

Minicomputer inkl. Zubehör
passend zu den Challenge Cards.

Was ist Physical Computing?

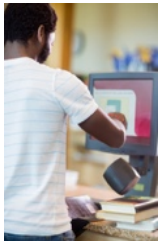
















Projekte im Physical Computing

Warum Physical Computing in der Schule?

Projekte im Physical Computing

- sind «hands-on», konstruktiv, vielfältig, projektorientiert.
- fördern eine Vielfalt von Kompetenzen.
- machen abstrakte Inhalte der Informatik greifbar.
- haben Lebensweltbezug.

Warum Physical Computing in der Schule?

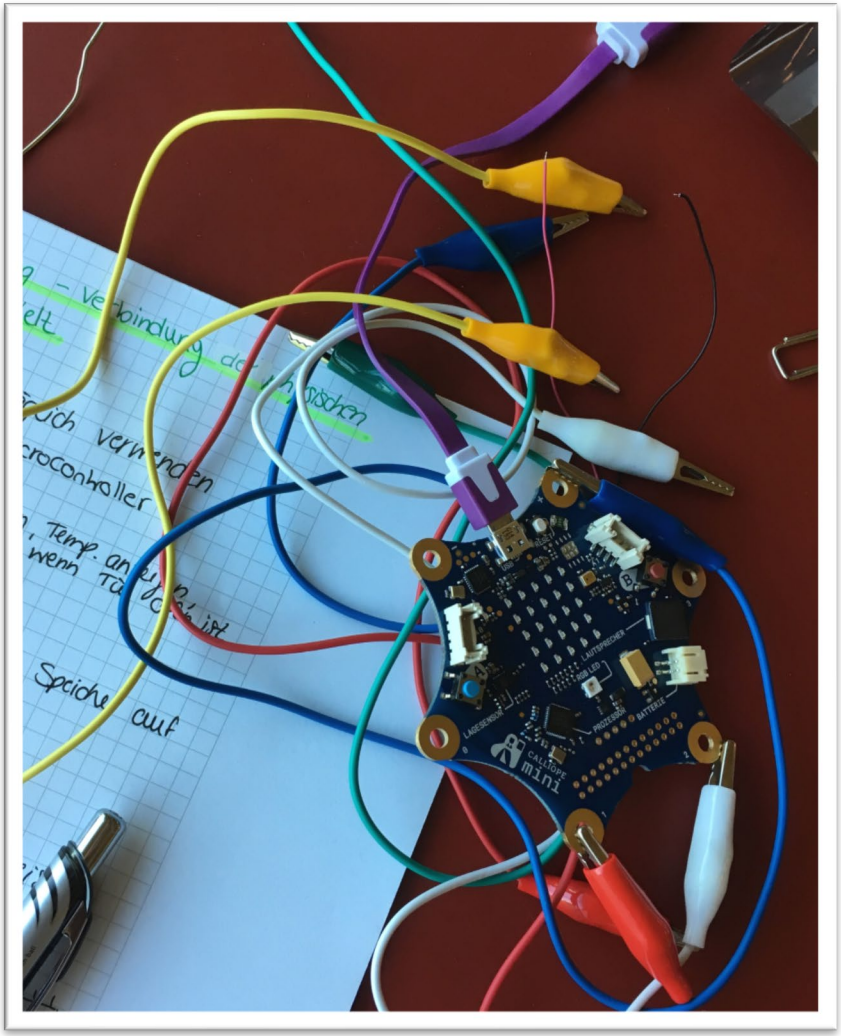
- Es geht nicht «nur» um die Programmierung in einem virtuellen Kontext, sondern bezieht die physische Welt sowie die Interaktion mit dem Nutzer mit ein.

- Man muss sich mit Aspekten der Elektronik, Gestaltung, Physik, Benutzerfreundlichkeit usw. auseinandersetzen.

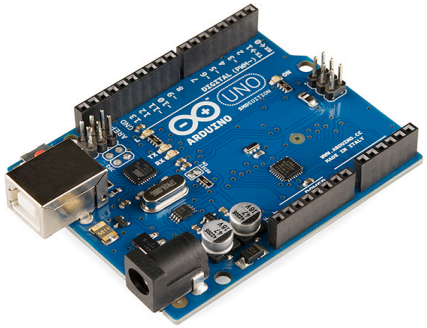
- Da man sich mit dem Informationsfluss von der physischen Welt über die digitale zurück zur physischen Welt auseinandersetzen muss, wird das Verständnis von Digitalisierung, Codierung, Datenstrukturen, Algorithmen und Computern allgemein gefördert.

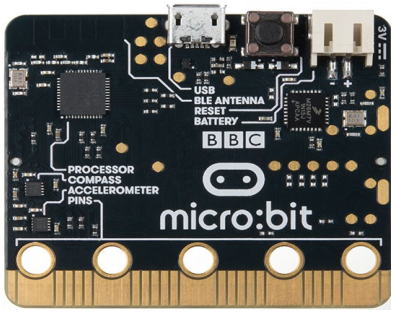
Warum Physical Computing in der Schule?

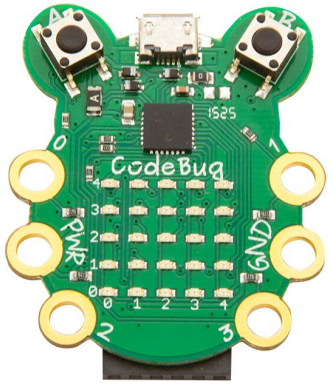
- Projekte in Physical Computing befinden sich an den Schnittstellen zu anderen Disziplinen (z.B. textiles und technisches Gestalten, Kunst, Naturwissenschaften, Technik)
- Physical Computing kann dazu beitragen, Informatik fächerintegriert zu unterrichten.
- Physical Computing macht auch einfach nur Spass!

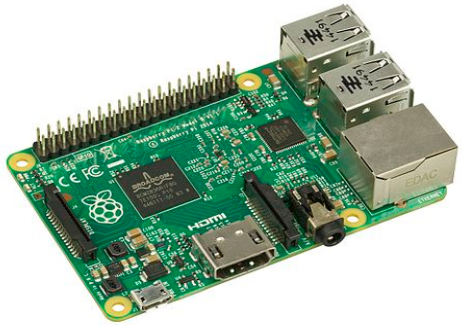


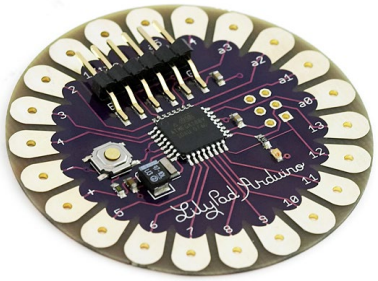
Minicomputer – Ein niederschwelliger Zugang zur digital vernetzten Welt

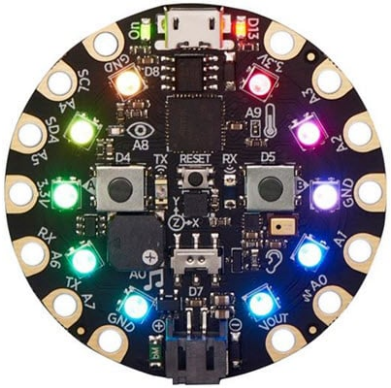


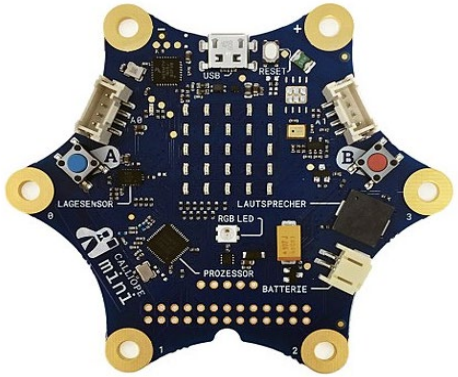












Physical Computing Themenheft und Challenge Cards


pädagogische hochschule schwyz

Physical Computing – Verbindung der physischen mit der virtuellen Welt


pädagogische hochschule schwyz

micro:bit Challenge-Cards

Physical Computing –
Meistere die Challenges und erlebe, wie man die ph
virtuelle Welt verbindet.

Inhalt

Grundlagen

– Der micro:bit (Ausstattung)
– Zubehör
– Ein Programm auf den micro:bit hochladen
– Analoger Input und Output
– Digitaler Input und Output

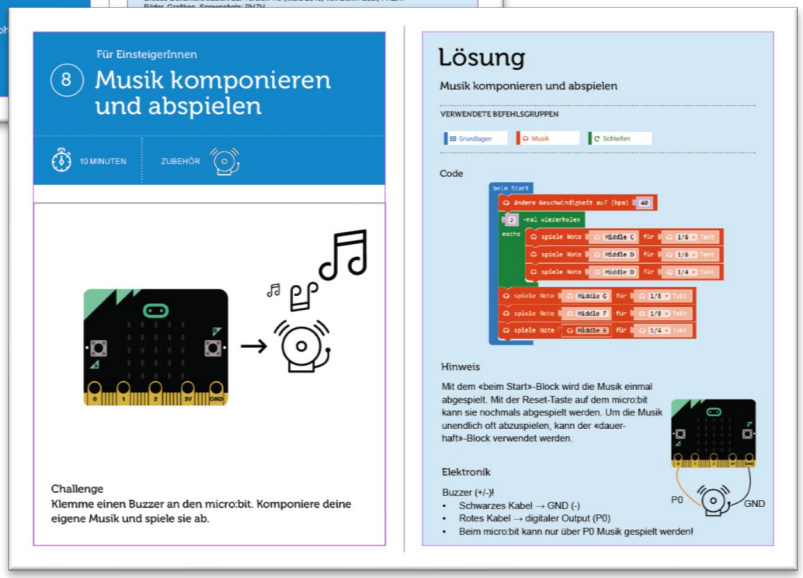
Challenges

1. Hello World!
2. Die Tasten A und B benutzen
3. Die Tasten A und B steuern das Licht
4. Eine Taste steuert das Licht
5. Einen verstellbaren Widerstand benutzen
6. Ein Licht dimmen
7. Einen Motor steuern
8. Musik komponieren und abspielen
9. Emojis mit der Fingerspitze verändern
10. Den Kompass benutzen
11. Die Helligkeit messen
12. Den Lagesensor benutzen
13. Die Temperatur messen
14. Die Farben des Regenbogens
15. Einen Servo-Motor steuern
16. Einen DC-Motor steuern
17. Einen linearen Motor steuern (Solenoid)

Impressum

Version 2.0 (Juli 2018)
Dr. Dorit Assaf
Pädagogische Hochschule Schwyz
dorit.assaf@phsz.ch, www.phsz.ch

Dieses Dokument basiert auf Version 1.0 (März 2018) von Dorit Assaf, PHZH.


Für EinsteigerInnen

8 Musik komponieren und abspielen

10 MINUTEN

ZUBEHÖR

Challenge
Klemme einen Buzzer an den micro:bit. Komponiere deine eigene Musik und spiele sie ab.

Lösung

Musik komponieren und abspielen

VERWENDETE BEFEHLSGRUPPEN

Code

Hinweis

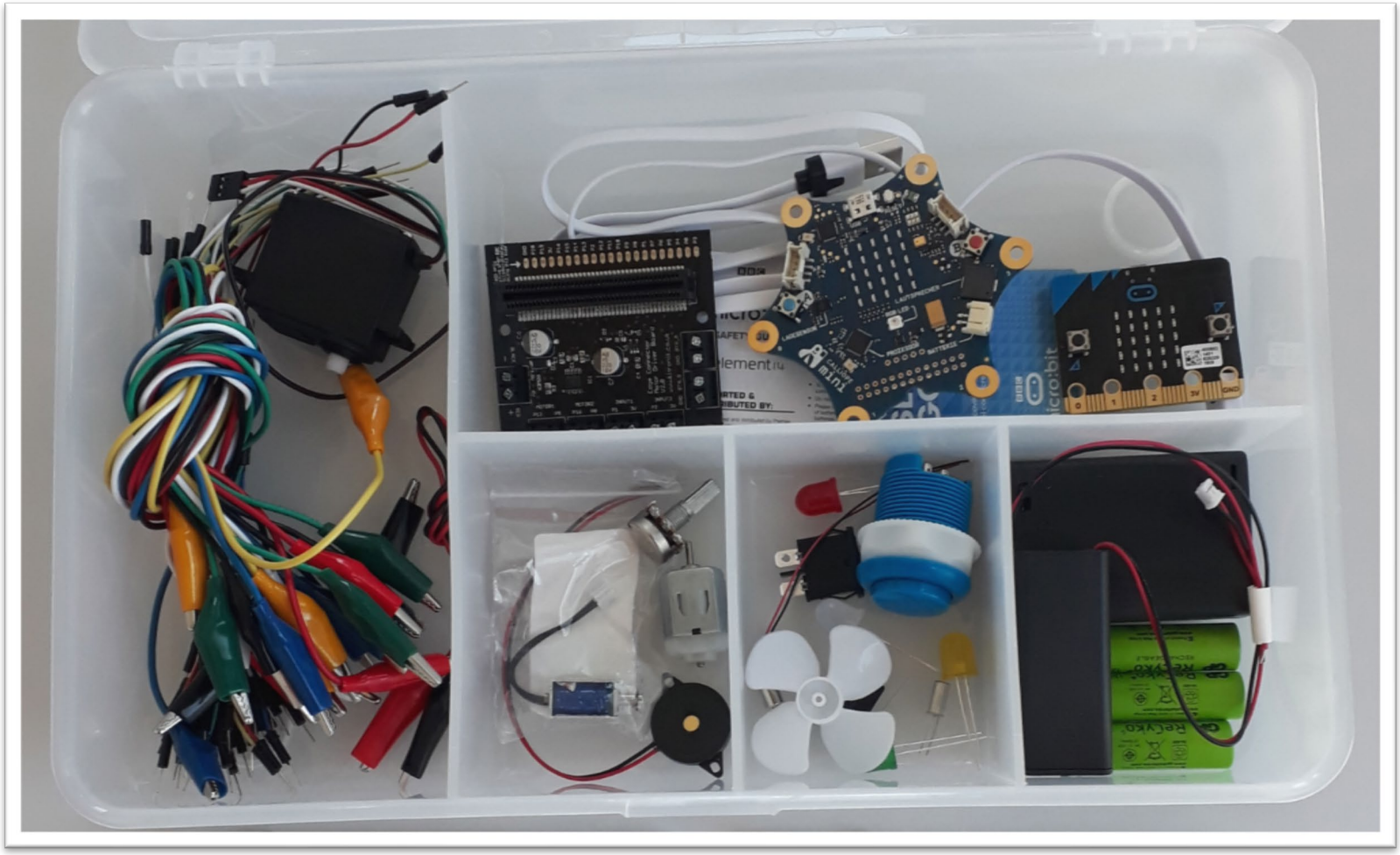
Mit dem «beim Start»-Block wird die Musik einmal abgespielt. Mit der Reset-Taste auf dem micro:bit kann sie nochmals abgespielt werden. Um die Musik unendlich oft abzuspielen, kann der «dauerhaft»-Block verwendet werden.

Elektronik

Buzzer (+/-)!
- Schwarzes Kabel → GND (-)
- Rotes Kabel → digitaler Output (P0)
- Beim micro:bit kann nur über P0 Musik gespielt werden!

P0 GND

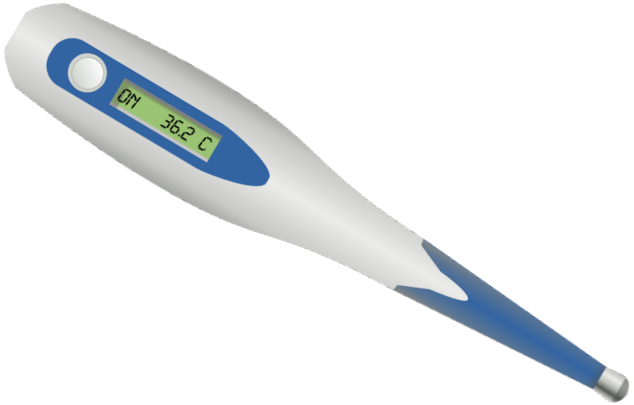
Kit zum Themenheft und Challenge Cards



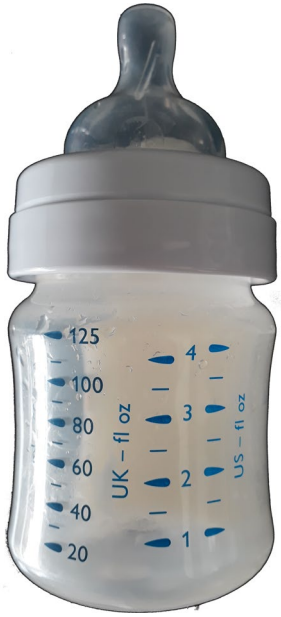

Aufgabe:
Bearbeite Challenge Cards 1-4
Calliope mini Challenge-Cards

Digitalisierung in der Informatik

Ein analoges Thermometer



Ein digitales Thermometer.



Skalen mit verschiedener Auflösung.

Digitalisierung in der Informatik

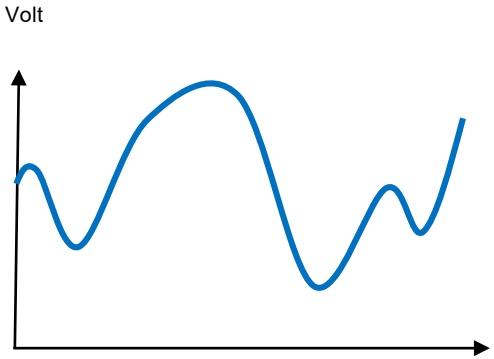
Ein Bild mit unterschiedlicher Auflösung.

2’457’600 Pixel

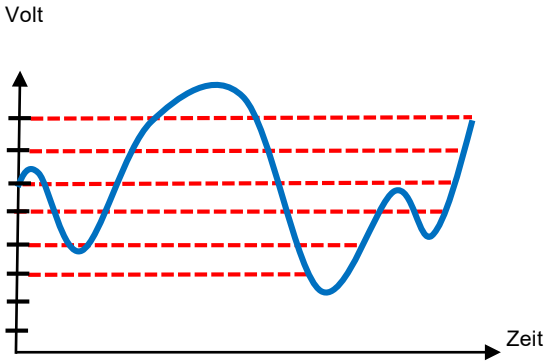
2’730 Pixel

983 Pixel.

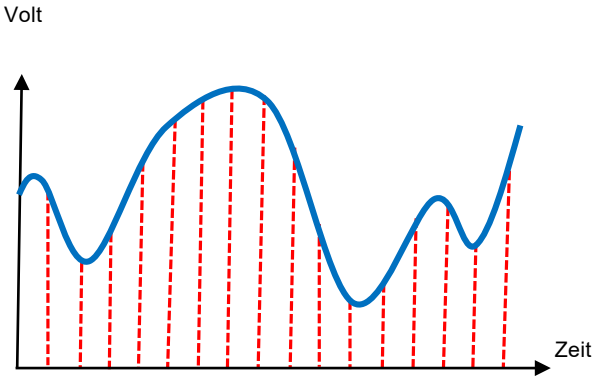
Digitalisierung in der Informatik



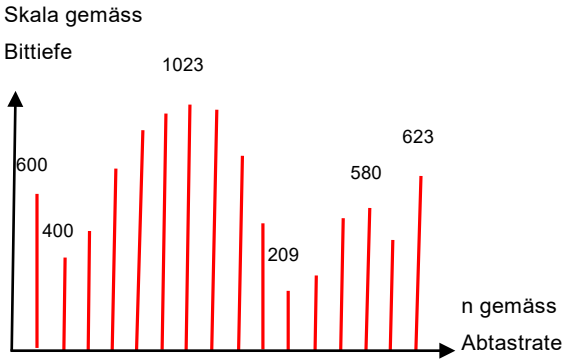
Ein kontinuierliches, analoges Signal.



Die Bittiefe entspricht der Auflösung der Skala bei der Digitalisierung des Abtastwerts.



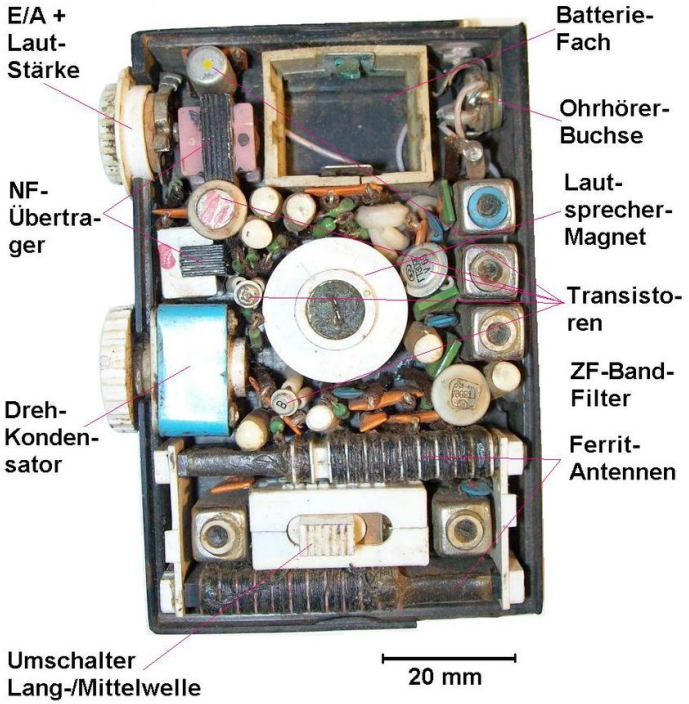
Die Abtastrate ist die Häufigkeit mit der das analoge Signal vom Analog-Digital-Wandler abgetastet wird.



Das Ergebnis der Digitalisierung eines analogen Signals durch einen Analog-Digital-Wandler ist eine Reihe von diskreten Zahlen.

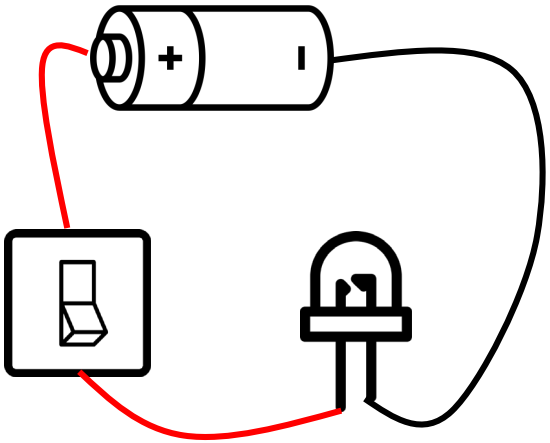
Analoge und digitale Schaltkreise

Tragbares Transistorradio "Transita" der Firma Nordmende aus den 1960er Jahren.

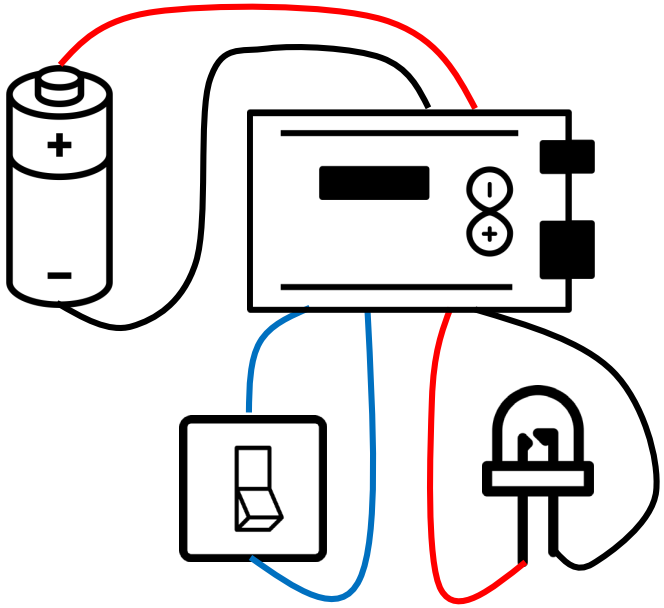


Geöffnetes Taschenradio der 1960er/1970er Jahre. Dieses Radio funktioniert ausschliesslich mit analogen elektronischen Bauteilen.

Analoge und digitale Schaltkreise

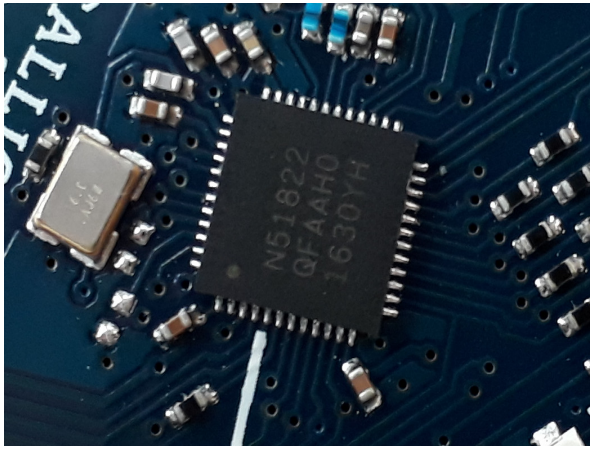


Ein klassischer analoger Schaltkreis mit Stromquelle, LED und Schalter.

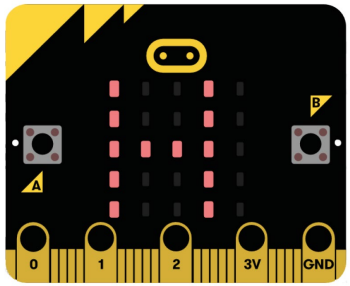


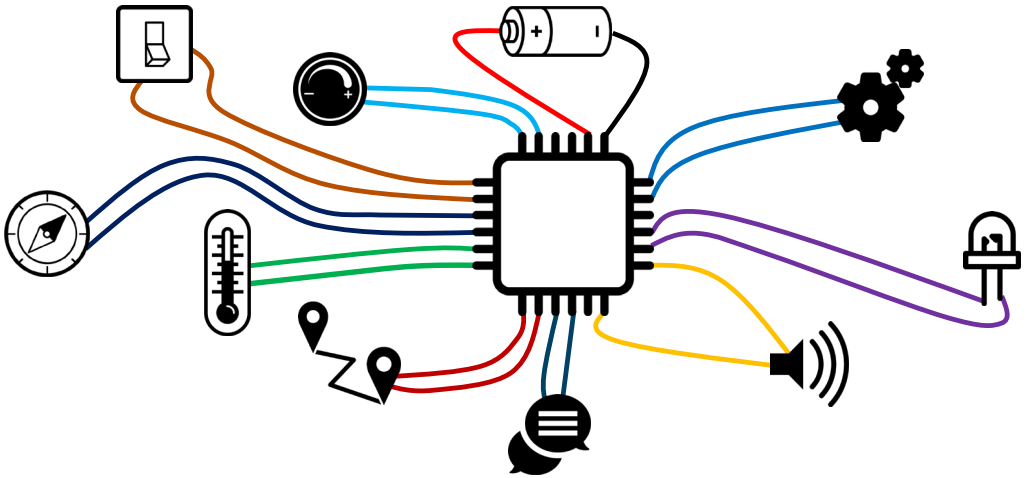
Ein digitaler Schaltkreis mit Stromquelle, Mikrocontroller, Schalter und LED.

Mit der physischen Welt in Kontakt treten



Die Pins eines Mikrocontrollers haben verschiedenste Funktionen, viele davon können vom Nutzer programmiert werden.

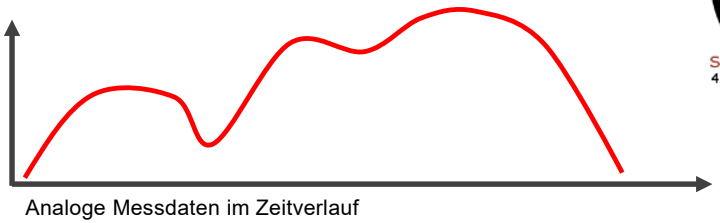


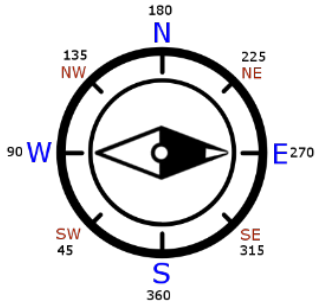


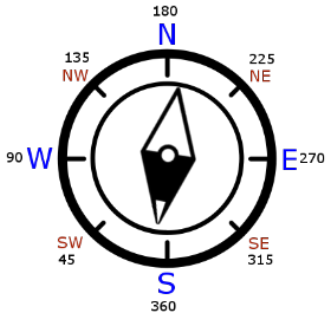
An die Pins eines Mikrocontrollers können die verschiedensten Komponenten angehängt werden, um mit der physischen Welt zu interagieren. Dabei benötigt jede Komponente mindestens zwei Pins, um einen geschlossenen Stromkreislauf aufzubauen. Über die Programmierung definiert man, welche Pins für welche Funktion zuständig sind.

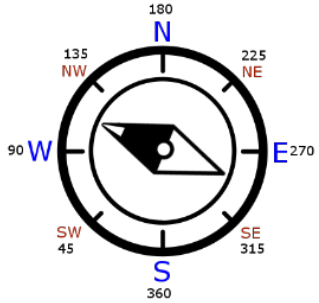
Sensoren – analoger Input

Ausrichtung
Kompass
Z.B. 0-1023







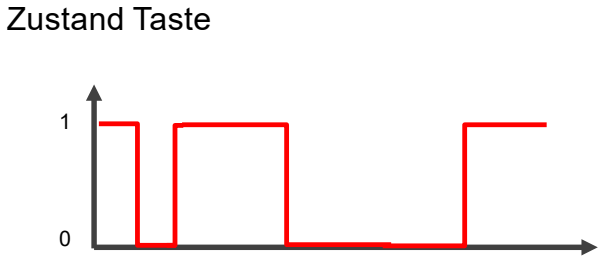


Analoger Input am Beispiel eines Kompasses. Die Messwerte haben einen Wertebereich von 0-360° (wenn bereits auf den Winkel geeicht wurde) oder sonst z.B. die 10 Bit Auflösung von 0-1023.

Sensoren – digitaler Input

Taste gedrückt
(z.B. Zustand «1»)

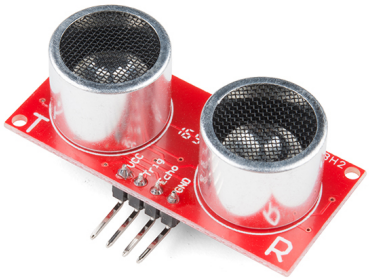
Taste nicht
gedrückt (z.B.
Zustand «0»)

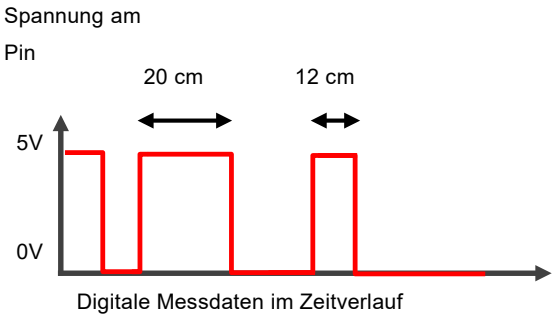


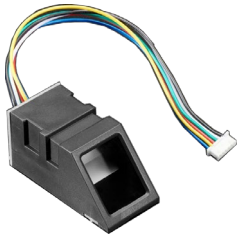
Digitale Messdaten im Zeitverlauf

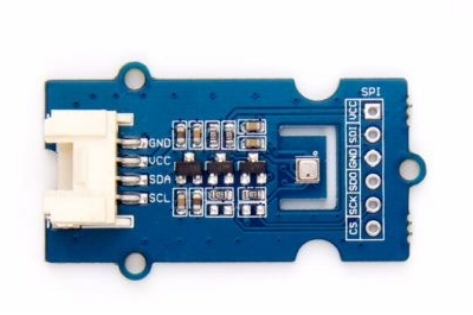
Digitaler Input am Beispiel einer Taste. Der Zustand ist entweder gedrückt oder nicht gedrückt bzw. 5V oder 0V und wird entweder als 1 oder 0 im Mikrocontroller repräsentiert.

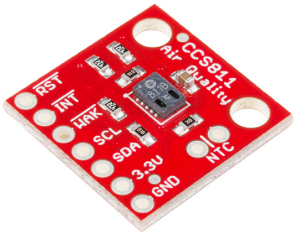
Sensoren – digitaler Input über Kommunikationsschnittstellen I²C, UART

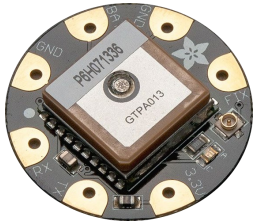








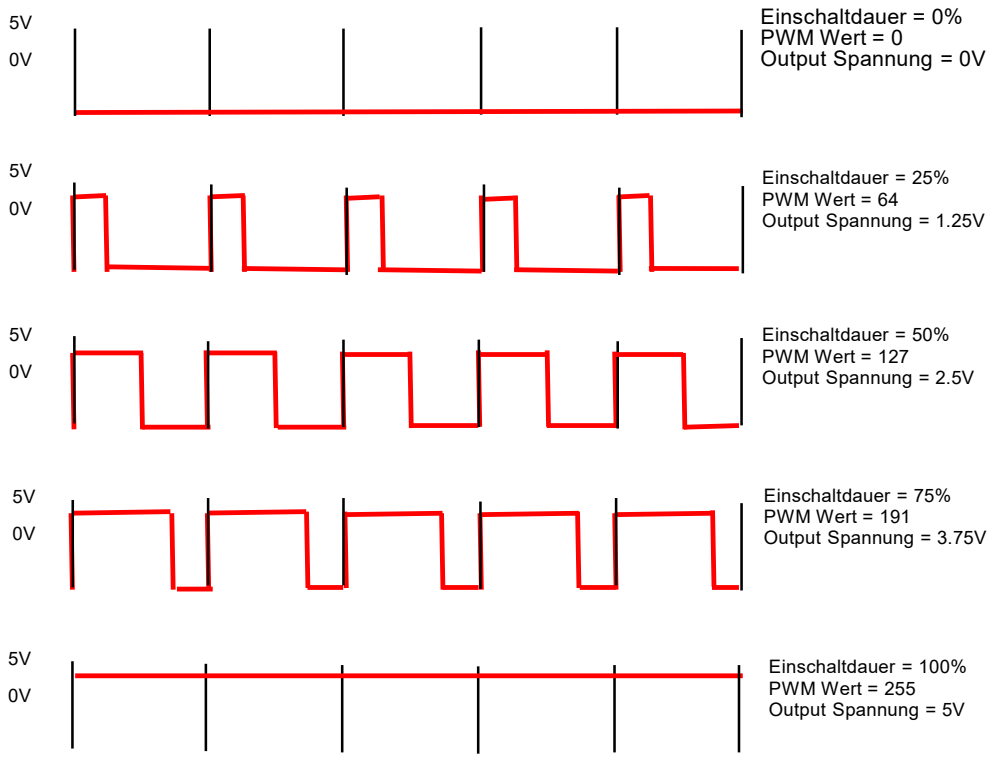




Verschiedene Sensoren mit unterschiedlichen Signalen. Oftmals kann die Art des digitalen Inputs der Pinbezeichnung an den Elektroden entnommen werden. Die Pins UART (RX,TX) und I²C (SCL, SDA) verraten, dass die Werte entweder über die serielle Kommunikation oder I²C übermittelt werden können.

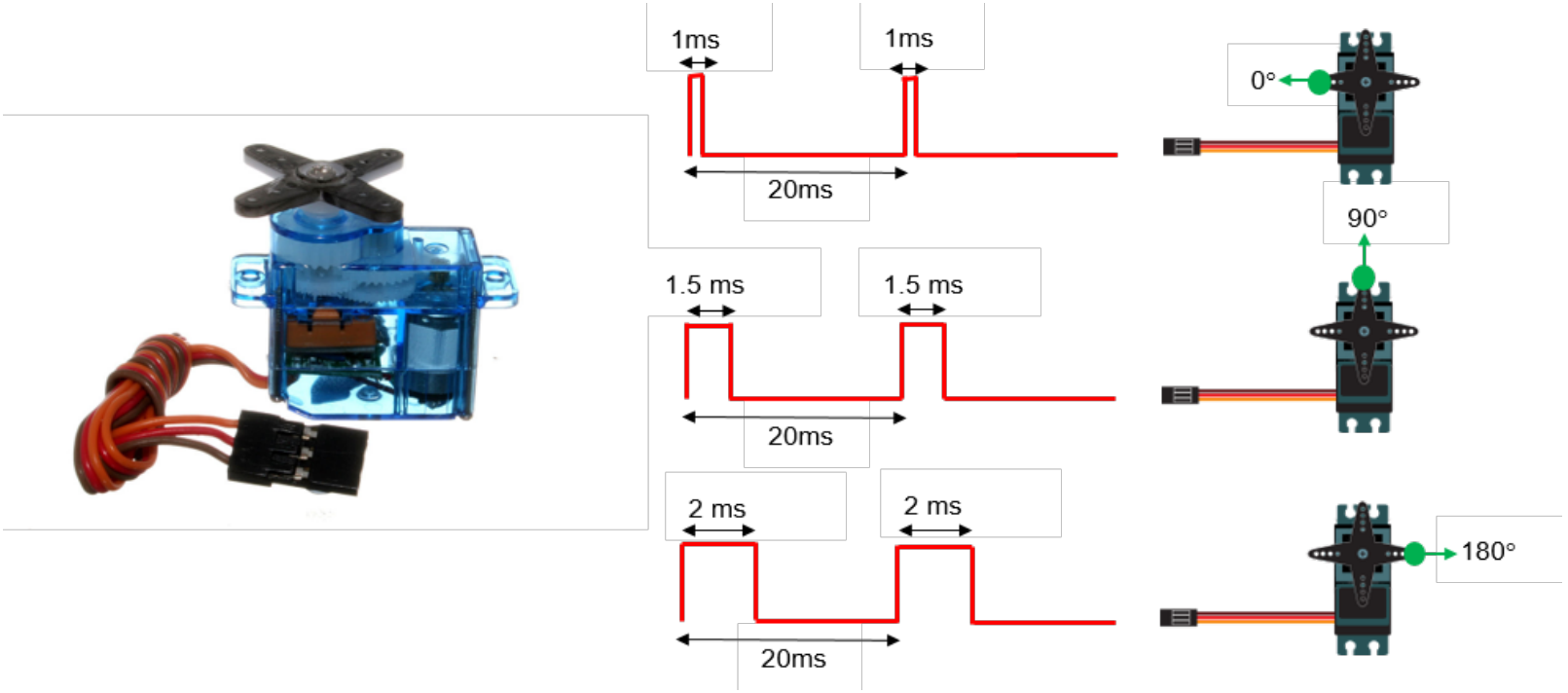

Aufgabe:
Bearbeite Challenge Cards 5-10
Calliope mini Challenge-Cards

Aktoren – «analoger» Output (PWM)



Durch Pulsweitenmodulation (PWM) erreicht der Mikrocontroller eine kontinuierliche Durchschnittsspannung zwischen 0V-5V am Pin. PWM wird als analoger Output bezeichnet.

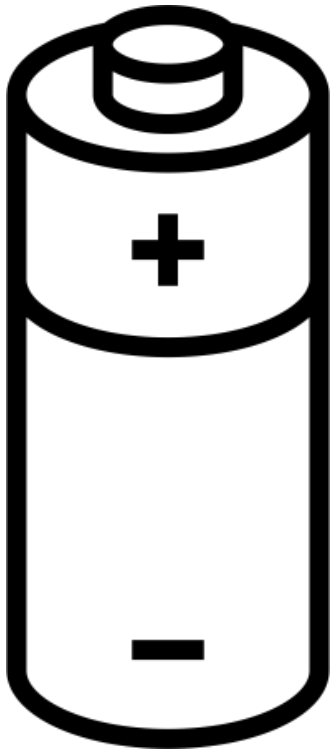
Aktoren – digitaler Output (PWM)



Ein digitales Signal für die Steuerung eines RC-Servo Motors. Die Pulsweite bestimmt den Soll-Winkel des Motors.

Stromversorgung

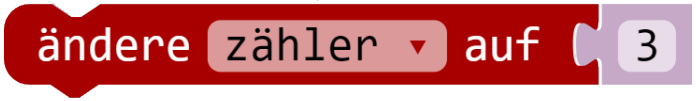
- Aktoren benötigen viel Strom (A).
- Aktoren benötigen oftmals eine höhere Spannung (V).
- Der micro:bit kann nicht so viel Strom und/oder Spannung liefern, wie der Aktor benötigt.
- Darum muss ein Aktor immer seine eigene Stromquelle (Batteriepack) haben.
- DC-Motoren müssen über ein Motor-Treiber (Motor-Board) am micro:bit angeschlossen werden.




Aufgabe:
Bearbeite Challenge
Cards 15-19
Calliope mini
Challenge-Cards

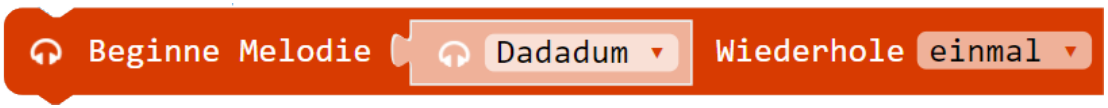
Programmierkonzepte – Anweisung

Ein Anweisungsblock.



Anweisung für die Deklaration einer Variablen und der Zuordnung eines Werts.

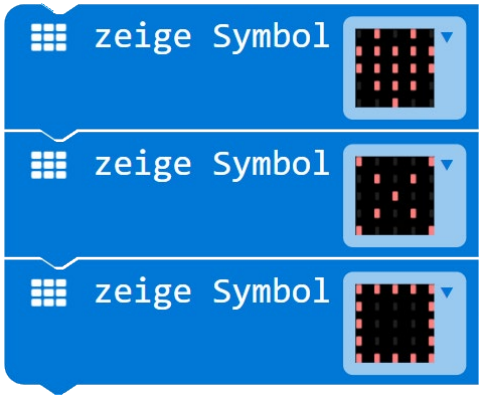
Anweisung für die Anzeige eines Lauftexts auf einem LED-Display.



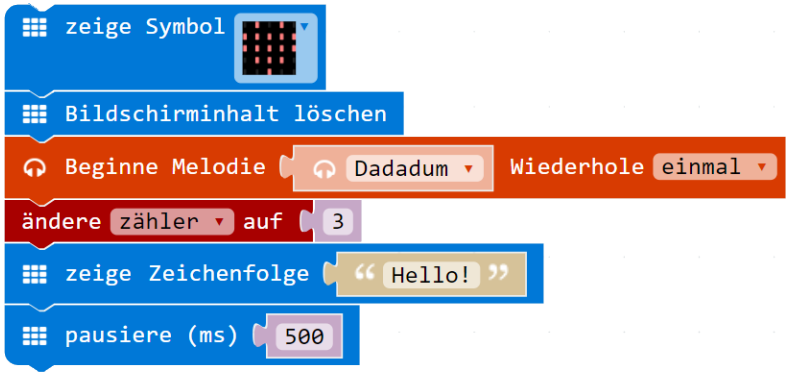
Anweisung für das Spielen einer Melodie auf einem Buzzer

Programmierkonzepte – Sequenz

Eine Sequenz von Anweisungen. Die Abarbeitung erfolgt von oben nach unten.

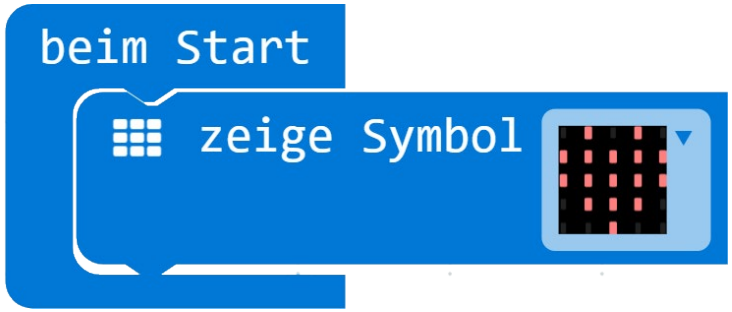


Eine Sequenz von Anweisungen.

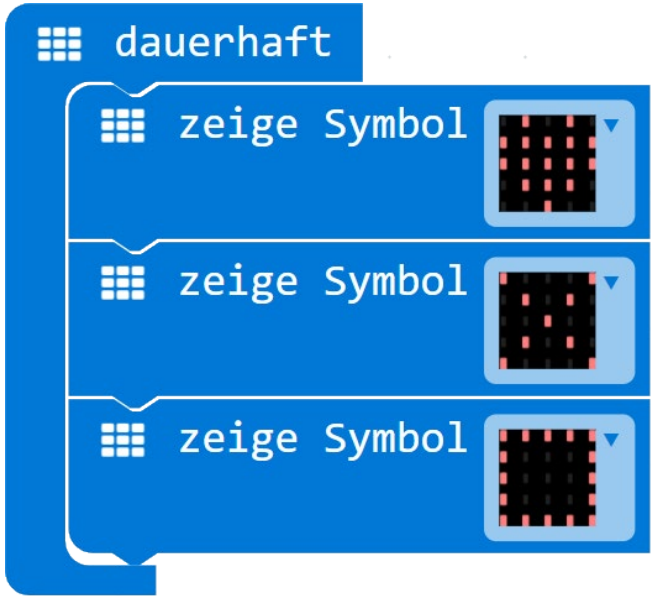


Beliebig viele verschiedene Anweisungen können zusammengesteckt werden.

Programmierkonzepte – Programmeinstieg und Endlosschleife



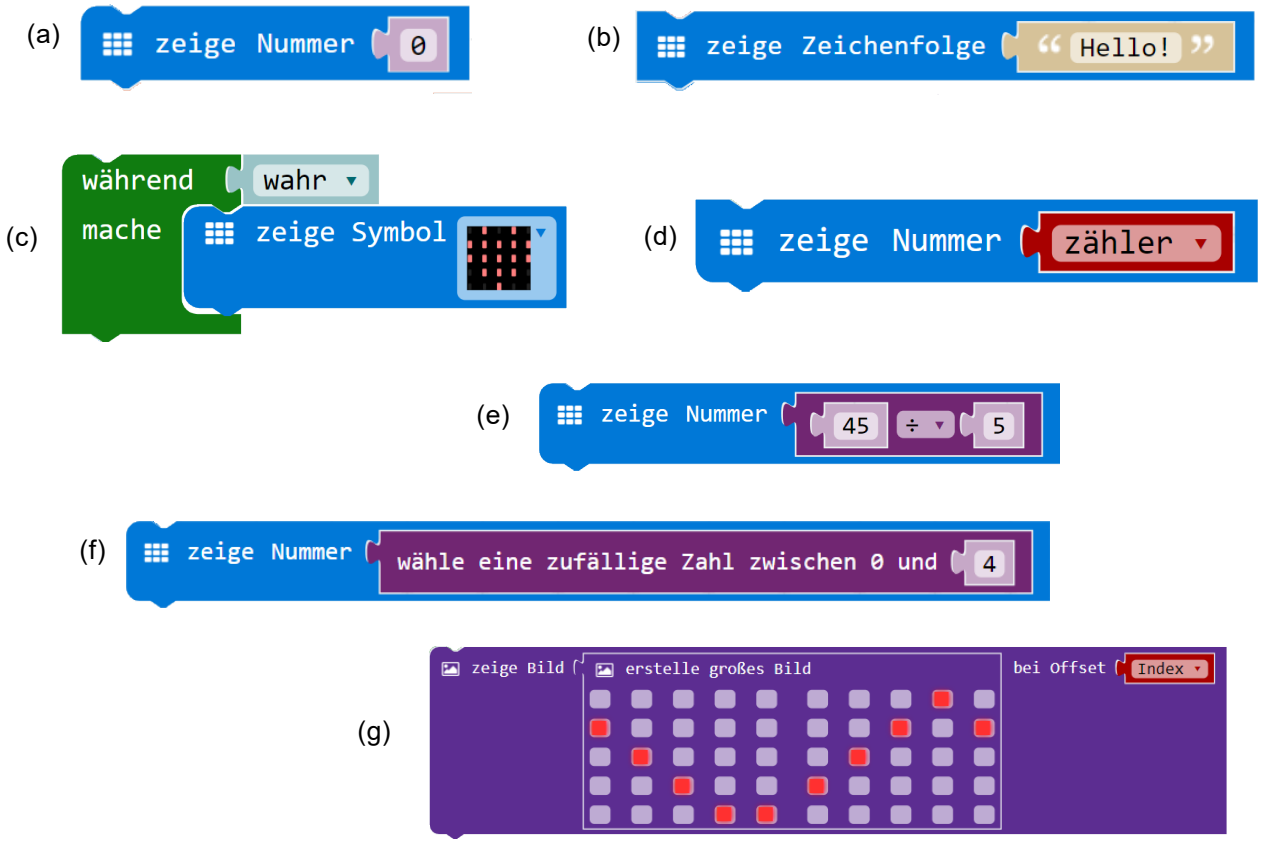
Programmeinstiegsblock



Eine Endlosschleife

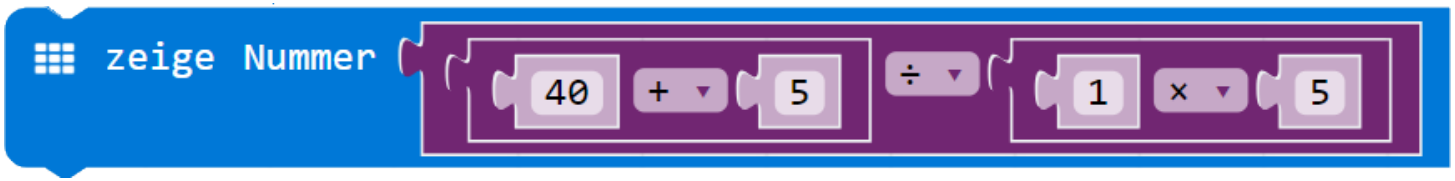
Programmierkonzepte – Parameter

Ein Parameterblock



Verschiedene Arten von Parametern. (a) Eine Zahl, (b) eine Zeichenkette, (c) ein Zustand «wahr», (d) der Wert einer Variablen, (e) das Ergebnis einer Rechnung, (f) eine Zufallszahl, (g) ein Pixelbild.

Programmierkonzepte – Parameter

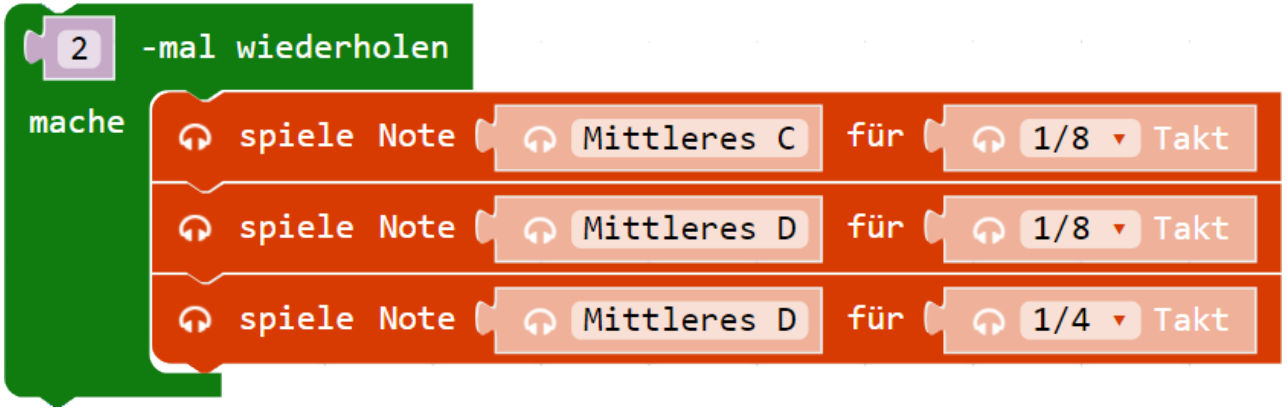


Parameter kann man beliebig verschachteln.

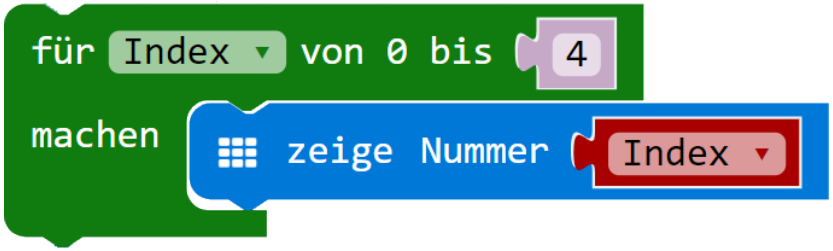
Über Parameter werden Sensoren ausgelesen.

Programmierkonzepte – Schleife Wiederhole x-Mal (FOR)

(a)



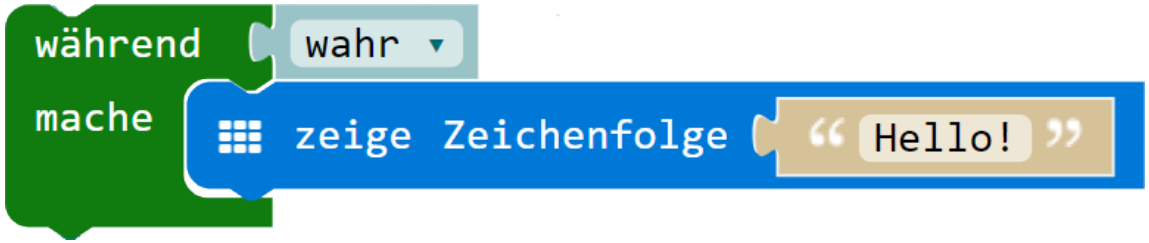
(b)



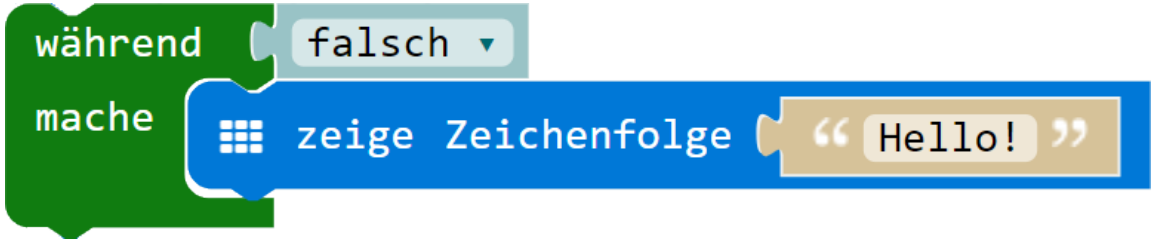
Eine Wiederhole x-mal Schleife (FOR). (a) Der Parameter gibt an, wie oft die Schleife wiederholt wird. (b) der Index enthält die aktuelle Zahl der Wiederholung.

Programmierkonzepte – Schleife
Wiederhole solange (WHILE)

(a)



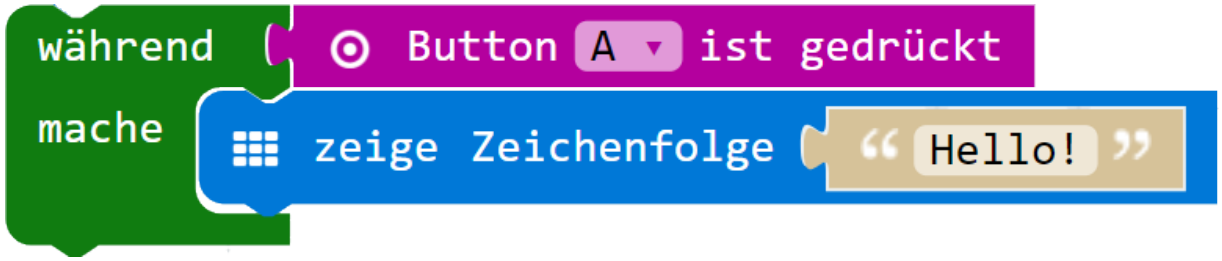
(b)



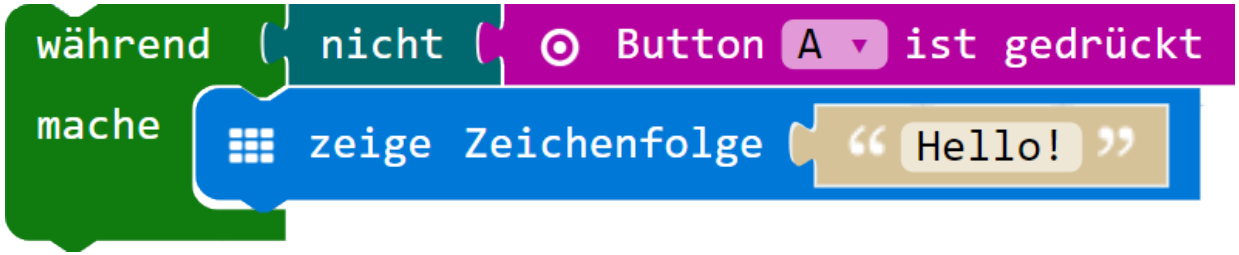
Eine «wiederhole solange» Schleife (WHILE). (a) wenn der Parameter «wahr» ist, wird die Schleife ausgeführt. (b) bei «falsch» wird die Schleife abgebrochen.

Programmierkonzepte – Schleife
Wiederhole solange (WHILE) mit Abbruchbedingung

(a)



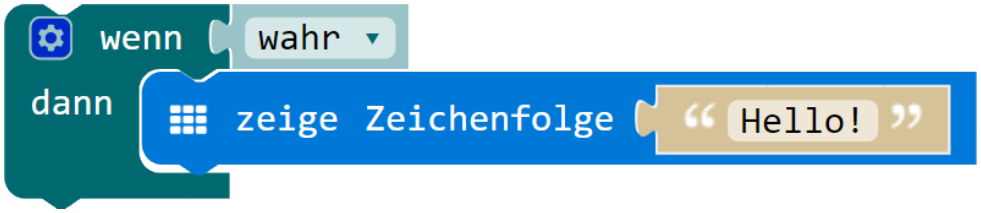
(b)



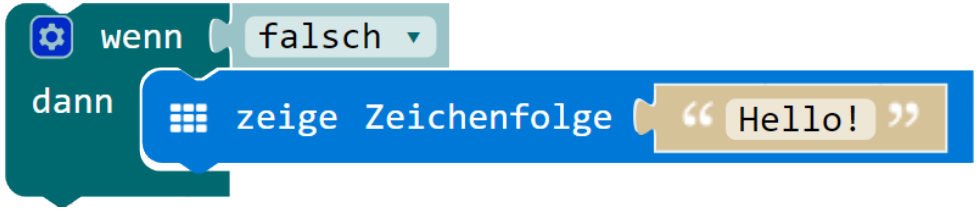
Die Abbruchbedingung ist abhängig von der Taste A. (a) Wiederhole, solange Taste A gedrückt ist. (b) Wiederhole, bis Taste A gedrückt ist.

Programmierkonzepte – Bedingung

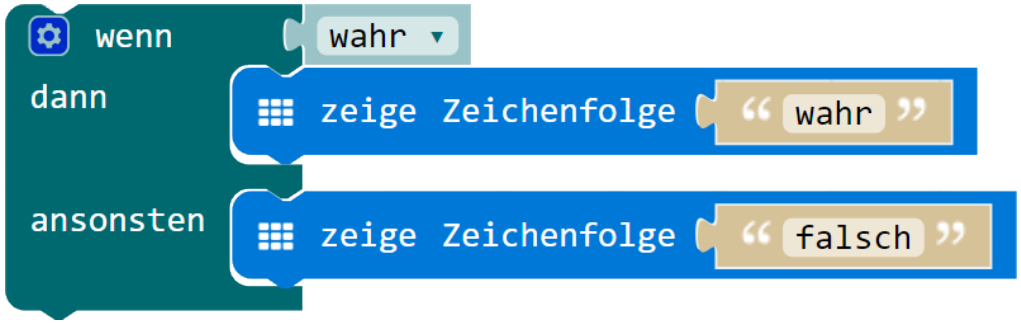
(a)



(b)

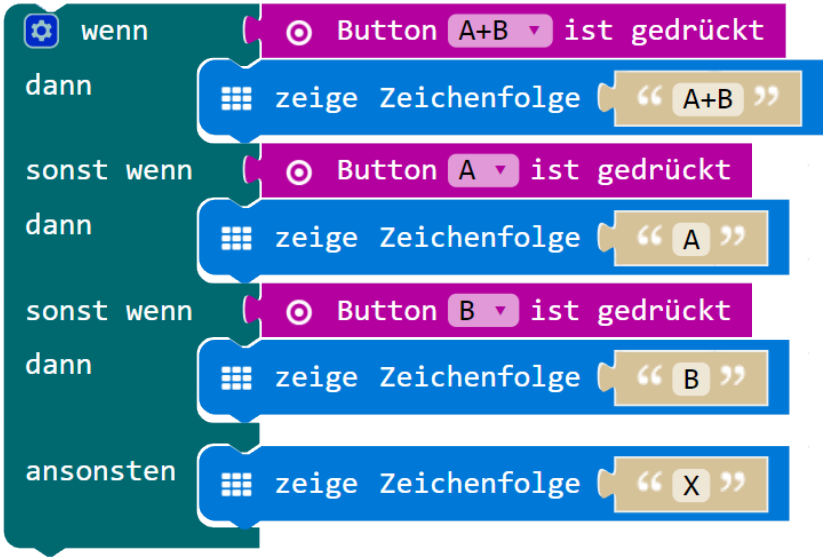


(c)

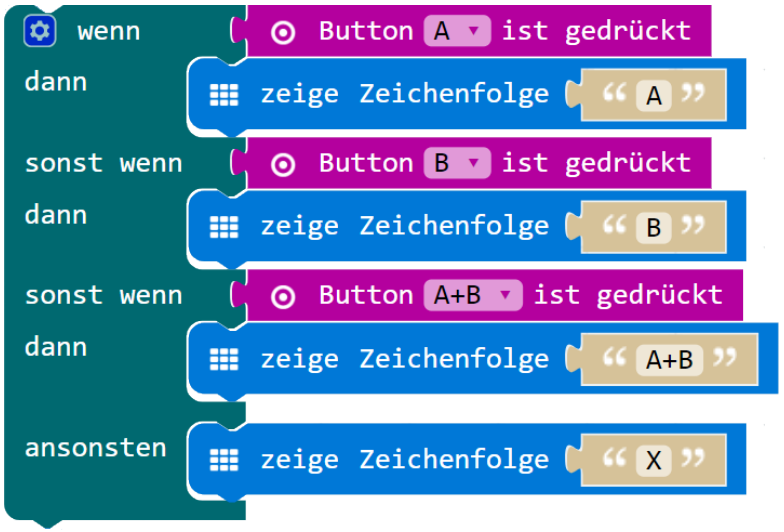


Bedingungsblock IF/ELSE und IF/ELSEIF/ELSE. (a) Wenn die Bedingung «wahr» ist, wird die Anweisung ausgeführt. (b) Wenn die Bedingung «falsch» ist, wird die Anweisung nicht ausgeführt. (c) Wenn «wahr», dann wird die erste Anweisung ausgeführt und wenn «falsch» dann die zweite.

Programmierkonzepte – Bedingung



Bedingungsblöcke können beliebig erweitert werden.



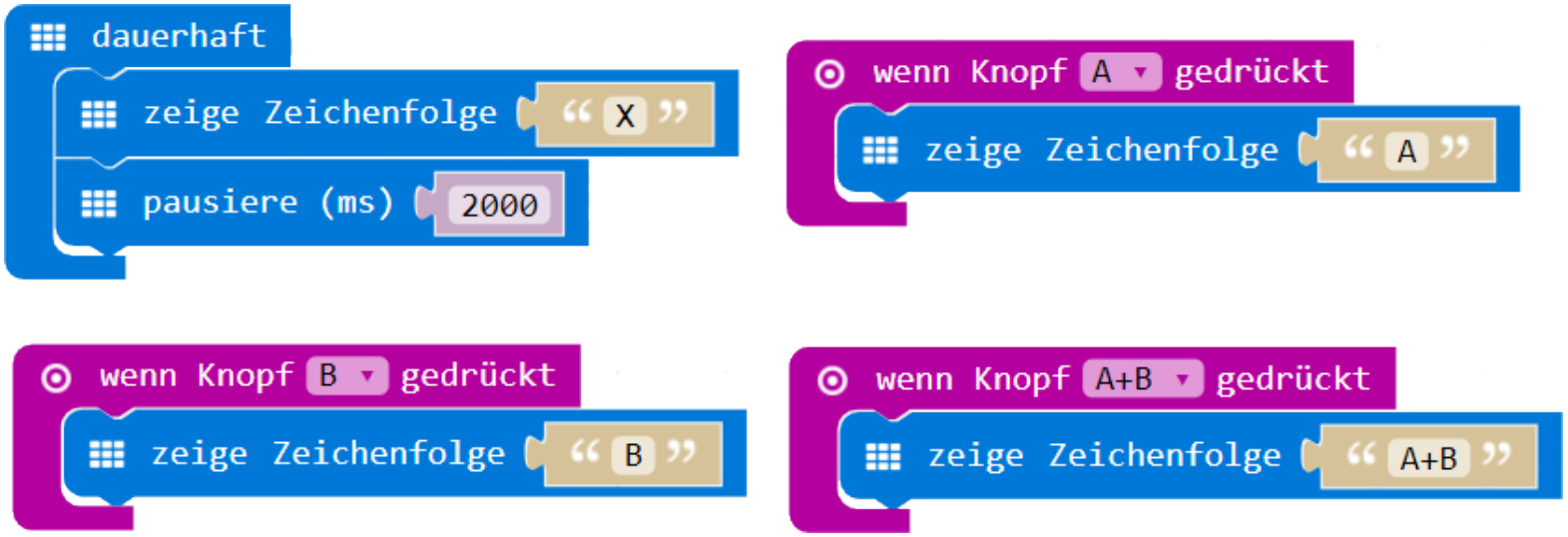
In diesem Bedingungsblock kommt es nie zur Frage, ob die Tasten A+B gedrückt sind, da die erste Bedingung in diesem Fall immer erfüllt ist.

Programmierkonzepte – Bedingung

Einzelne Bedingungen untereinander werden immer geprüft. «Ansonsten» funktioniert hier jedoch nicht.

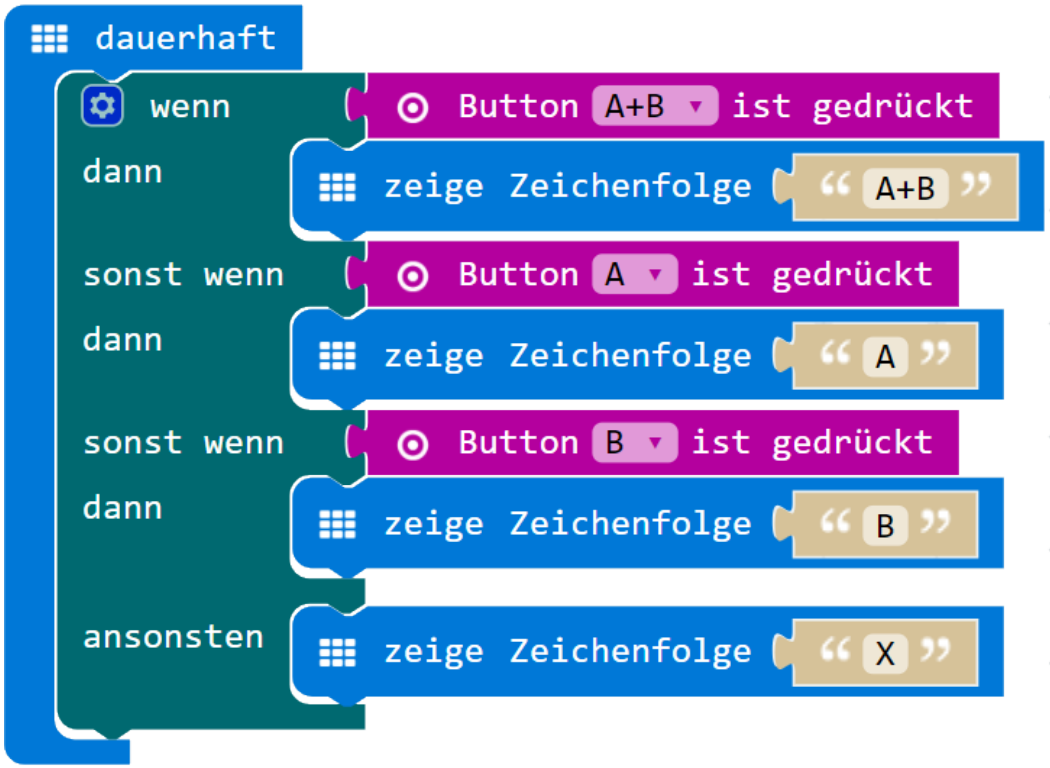
Programmierkonzepte – Ereignis (Interrupt)

Ein Ereignisblock.



Ein Programmcode bestehend aus drei Ereignisblöcken und einer Endlosschleife.

Ereignis vs. «Polling»



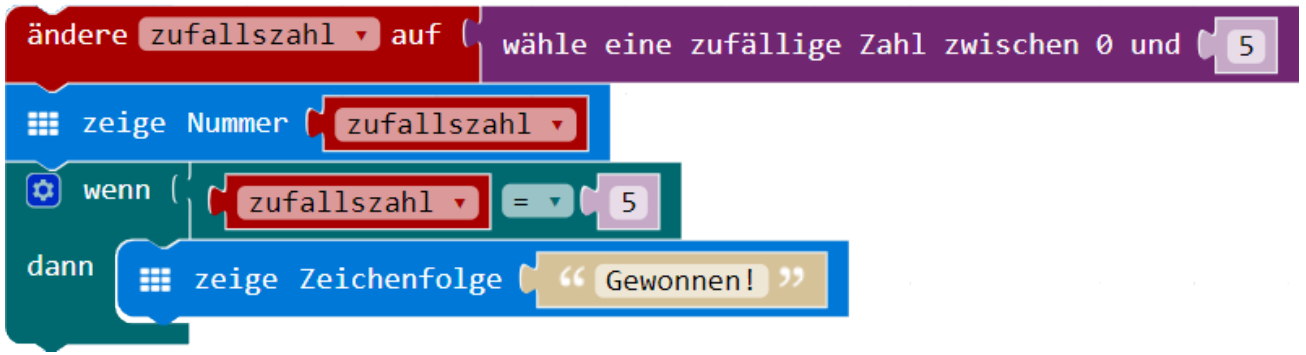
Über «Polling» in der Endlosschleife kann man die Sensoren ständig abfragen.

Programmierkonzepte – Variable

Der Definitionsblock und dazugehörige Parameter einer Variablen.

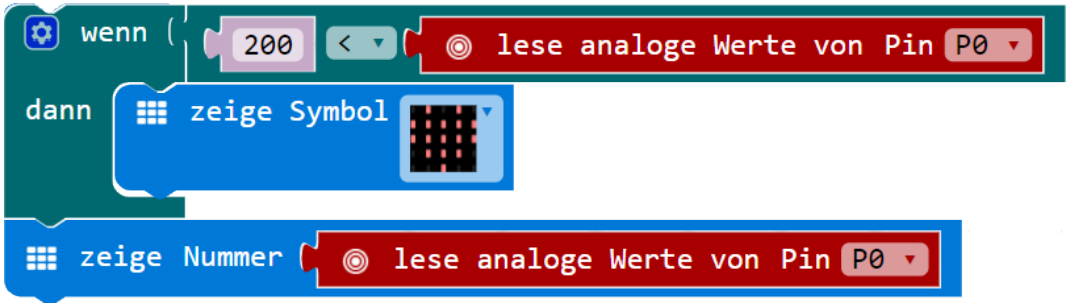
(a)

(b)

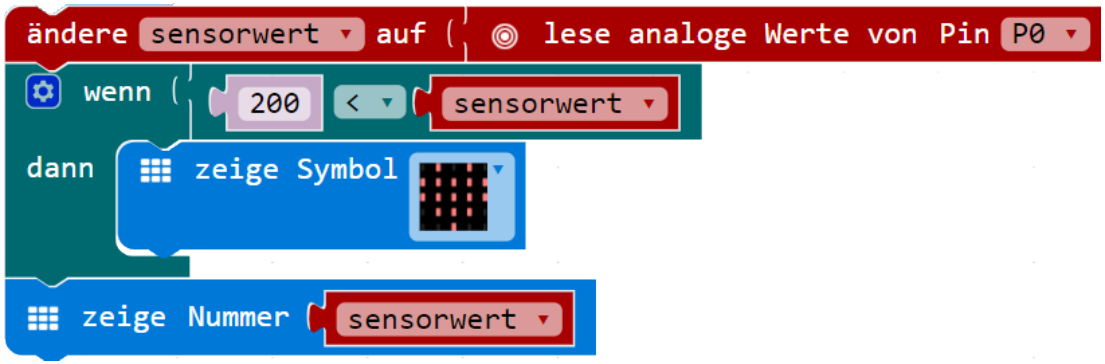


Variablen speichern Werte für den späteren Gebrauch im Programm. (a) Nach der Anzeige auf dem LED-Display hat man keinen Zugriff mehr auf die Zufallszahl. (b) Eine Variable speichert den Wert der Zufallszahl, damit sie später wiederverwendet werden kann.

Programmierkonzepte – Variable



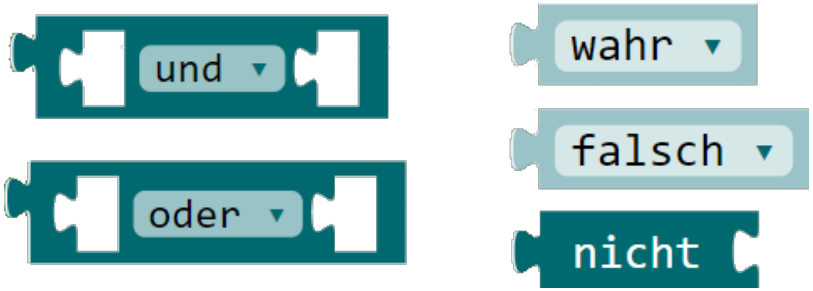
(a)



(b)

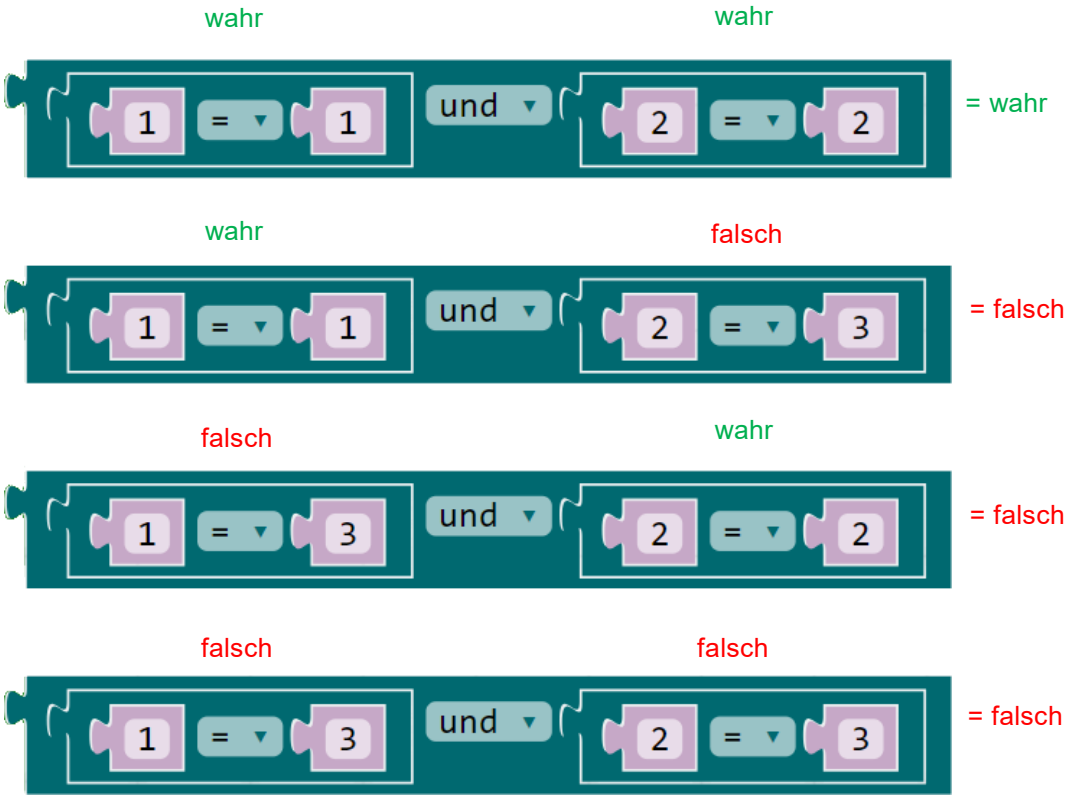
Speichern eines Messwerts über eine Variable. (a) Hier wird der Messwert zweimal ausgelesen und kann unterschiedlich sein. (b) Über die Variable kann man davon ausgehen, dass derselbe Messwert weiterverarbeitet wird.

Programmierkonzepte – Boolesche Algebra (Aussagenlogik)



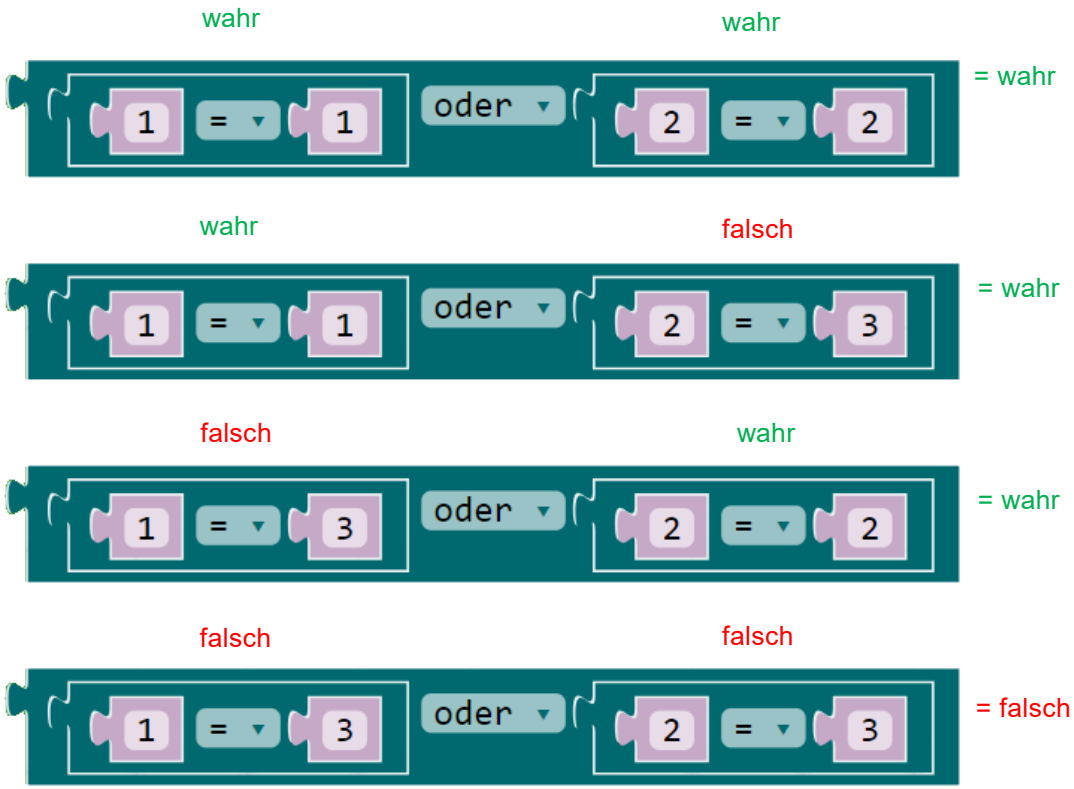
Ausdrücke der Booleschen Algebra.

Programmierkonzepte – UND (AND)



Bei einer UND-Verknüpfung müssen alle Teilaussagen «wahr» sein, damit die gesamte Aussage «wahr» ist.

Programmierkonzepte – ODER (OR)



Bei einer ODER-Verknüpfung muss mindestens eine Teilaussage «wahr» sein, damit die gesamte Aussage «wahr» ist.

Programmierkonzepte – NICHT (NOT)

= falsch

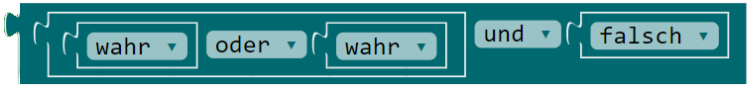
= wahr

= falsch

= wahr

Durch NICHT wird die Aussage ins Gegenteil gesetzt. Zwei NICHT-Blöcke heben sich wieder auf.

= falsch



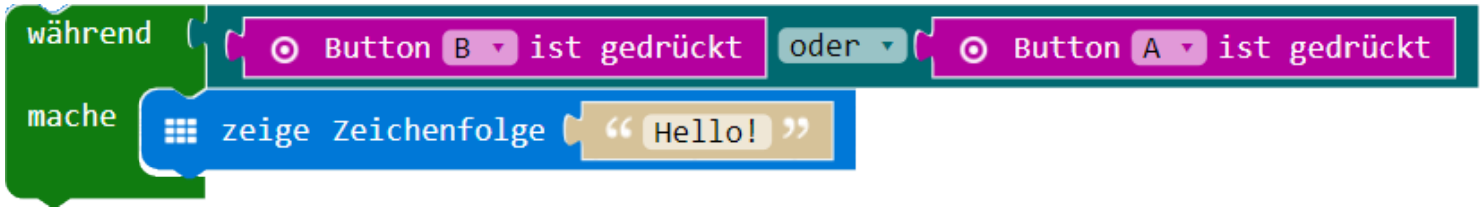
= falsch

= wahr

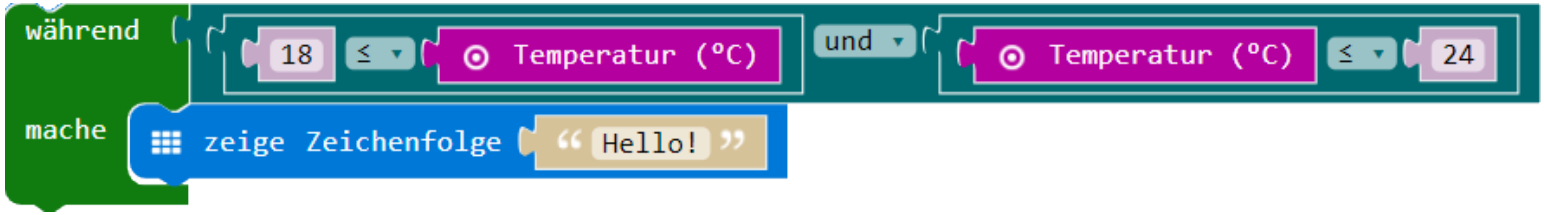
Aussagen der Booleschen Algebra können beliebig miteinander verschachtelt werden.

Boolesche Algebra im Einsatz

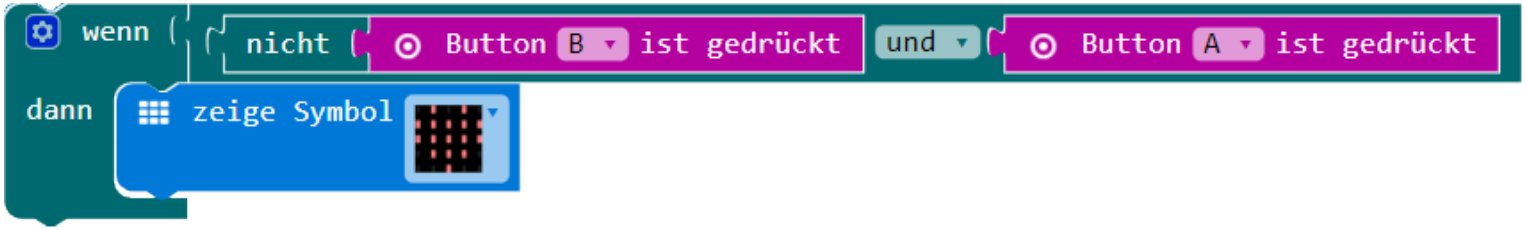
(a)



(b)



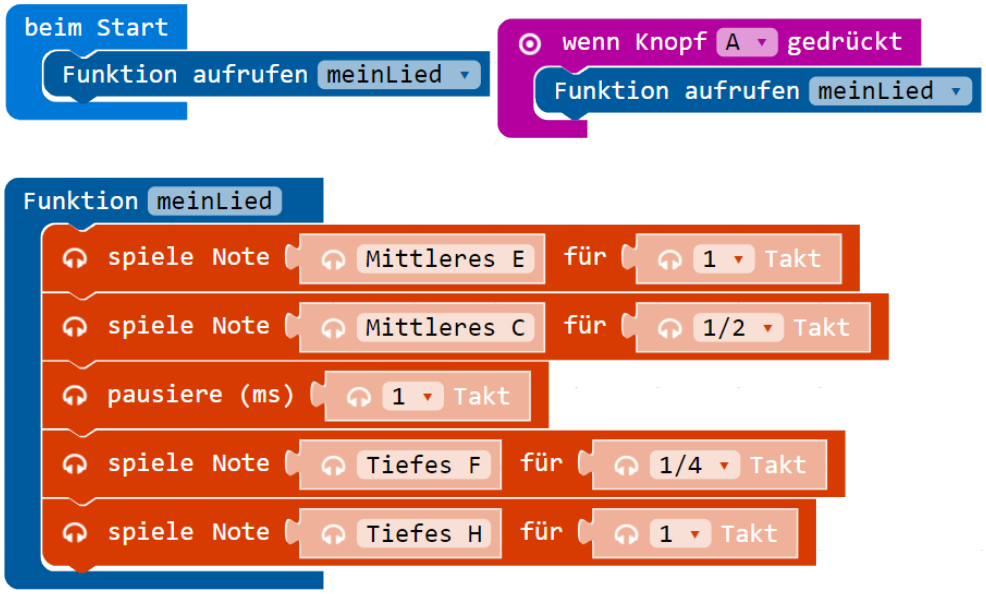
(c)



Ausdrücke der Booleschen Algebra werden vor allem bei Bedingungen und Schleifen verwendet.

Programmierkonzepte – Unterprogramm (Funktion)

Der Definitionsblock und dazugehörige Anweisungsblock eines Unterprogramms.



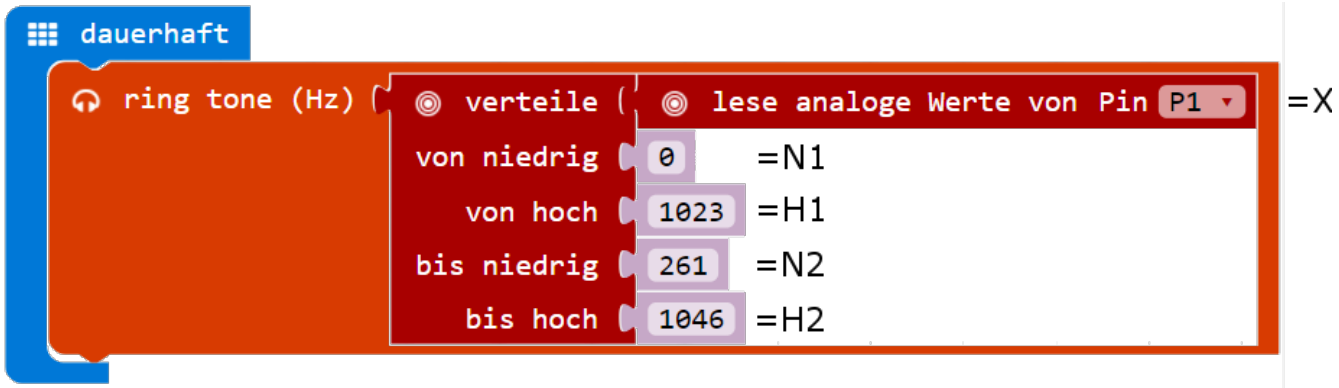
Ein eigenes Lied als Unterprogramm implementiert. Über die Anweisungsblöcke kann das Lied von verschiedenen Stellen im Programmcode aufgerufen werden.

Umsetzung einer Idee

- Welche Sensoren und Aktoren werden benötigt.
- Sensoren und Aktoren einzeln testen, Wertebereiche notieren, Test-Programmcode aufbewahren.
- Wieviel und welche Arten von Pins werden benötigt? Sind es analoge oder digitale Input oder Outputs?
- Dekomposition: Projektidee in kleine Einzelteile aufteilen und separat implementieren und testen.
- Zusammenbringen des Programmcodes und der Elektronik.
- Testen der Projektidee.

Wertebereiche von Inputs und Outputs aufeinander abstimmen – «Verteile»-Funktion

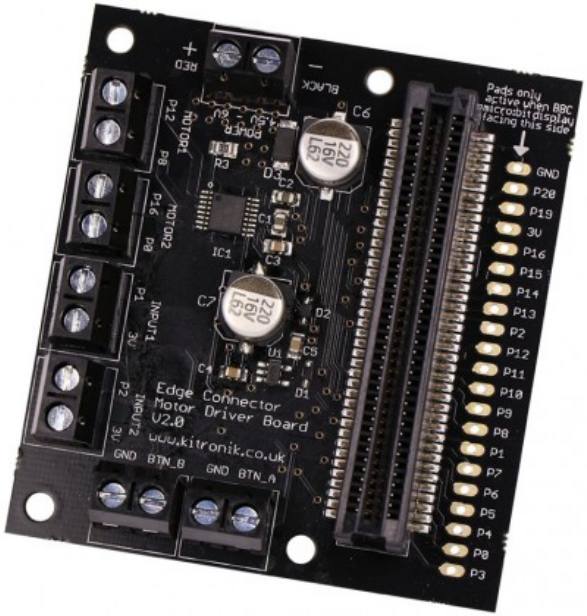
$$Frequenz\ Hz = (H2 - N2)\frac{(X - N1)}{(H1 - N1)} + N2$$



Beispiel: Das Mapping eines analogen Inputs auf Pin P1 mit dem Wertebereich von 0-1023 zum Frequenzbereich eines Buzzers 261Hz - 1046Hz.

$$525.7\ Hz = (1046 - 261)\frac{(345 - 0)}{(1023 - 0)} + 261$$

Wenn die Pins nicht ausreichen…



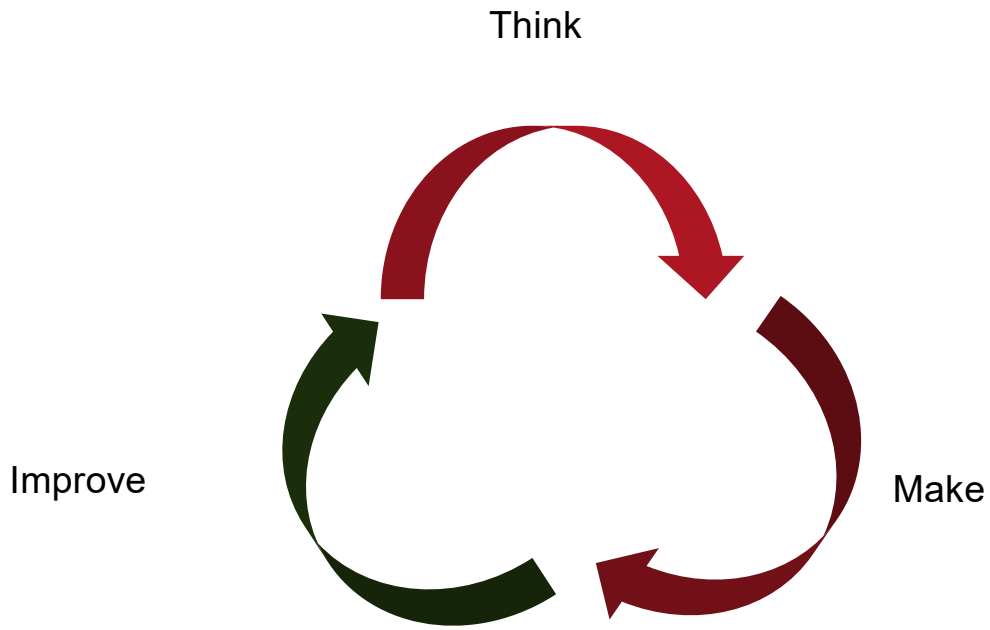
Micro:bit Motor Board

Motor 1 und Motor 2 Stecker nur für lineare und DC Motoren verwenden.

Edge Connector Breakout Board

Pins zur freien Verfügung: 1, 2, 8, ≥12

Problemlöseprozess



Think

Brainstorming, Diskutieren, Prognostizieren, Sammeln von Material, Expertise identifizieren, Gruppen bilden (oder alleine arbeiten), Zielsetzung, Skizzieren, Abgrenzen, Diagramme zeichnen, Recherchieren, Planen.

Make

Spielen/Ausprobieren, Bauen, Basteln, Kreieren, Programmieren, Experimentieren, Konstruieren, Auseinandernehmen, Vorgehensweisen und Materialien testen, andere beobachten, Code kopieren, Code teilen, Prozess dokumentieren, Probleme suchen und finden, Fragen stellen, Fehler beheben.

Improve

- **Wenn es nicht funktioniert oder das Team steckenbleibt:** Recherchieren, Ausdiskutieren im Team, Diskutieren mit Kollegen, das Problem von einem anderen Blickwinkel betrachten, andere Materialien verwenden, einzelne Komponenten des Projektes verändern, diskutieren, ob und wie ein ähnliches Problem zu einem früheren Zeitpunkt gelöst wurde, herumspielen, ein ähnliches Projekt finden zum Analysieren oder Auseinandernehmen, eine Expertin oder einen Experten fragen, cool bleiben, frische Luft schnappen, darüber schlafen.

- **Wenn das Team «fertig» ist:** Möglichkeiten finden, um das Produkt zu verbessern oder weiterentwickeln. Fragestellungen: Wie kann das Produkt so gestaltet werden, dass es schneller, langsamer, besser, genauer, schöner, ökologischer, cooler, stärker, smarter, flexibler, grösser, kleiner, effizienter, kostengünstiger, verlässlicher, leichter, eleganter, einfacher zu benutzen ist?

Was macht ein gutes Projekt aus?

- **Zweck und Relevanz:** Hat das Projekt einen persönlichen Bezug und Relevanz? Weckt das Projekt genügend Interesse, um Zeit, Energie und Kreativität darin zu investieren?
- **Komplexität:** Ein gutes Projekt beinhaltet verschiedene Themen und knüpft am Vorwissen an.
- **Wiederverwendbarkeit:** Schülerinnen und Schüler sollen etwas erschaffen, das sie untereinander teilen und wiederverwenden können (z.B. Programmcode, Konstruktionen usw.) Dies fördert auch die Motivation und Relevanz der Arbeit jedes einzelnen.
- **Gendergerecht:** Für einen gendergerechten Unterricht ist es ratsam, Themen zu wählen, die verschiedene Interessen der Schülerinnen und Schüler ansprechen. Projekte im Physical Computing bieten dafür gute Möglichkeiten, da sie an den Schnittstellen zu anderen Disziplinen angesiedelt sind.

Rahmenbedingungen

- **Zeit:** Genügend Zeit muss bereitgestellt werden, damit der Problemlöseprozess iterativ durchlaufen werden kann.
- **Fehlerkultur:** Fehler sind ein wichtiger Bestandteil des Lernprozesses. Die Schülerinnen und Schüler sollen dazu motiviert werden, Risiken einzugehen und einfach etwas auszuprobieren. Es ist auch ganz normal, dass etwas dabei kaputtgehen kann. Fehler sollen keinen negativen Faktor in der Beurteilung des Projektes ausmachen, im Gegenteil.
- **Kollaboration:** Programmieren lernt man am besten, wenn man bestehenden Programmcode analysiert, abändert und ergänzt. Code von anderen zu kopieren, um Ideen nachzubauen, ist eine wichtige Arbeitsweise in der Informatik und im Making.
- **Zugang zu Materialien:** Es ist sehr wichtig, dass die Schülerinnen und Schüler möglichst uneingeschränkten Zugang zu den Materialien (Software, Hardware, Bastelmaterial usw.) haben und dass die Infrastruktur funktioniert (WLAN, Computer usw.), am besten auch ausserhalb der offiziellen Lektionen.