

Microcontroller+Robotik-Kurs

Zweimal drei Stunden für Kinder ab 12 Jahren

Ein Roboter

Ein Roboter nimmt Information aus der Umgebung auf und führt dazu Bewegungen aus. Bei einem Industrieroboter kann zum Beispiel ein Greifarm mit einem Werkzeug arbeiten, wenn ein passendes Bauteil vorbeikommt. Du kannst nach diesem Kurs ein Fahrzeug bauen, das merkt, wenn es eine Wand berührt, und dann rückwärts eine Kurve fährt usw.

Eine Maschine in Menschengestalt, die einfach nur dumm im Kreis herum läuft, ist also kein richtiger Roboter.

Programmieren und Bauen

Software: Damit der Roboter so reagiert, wie du es willst, musst du es ihm beibringen. Das heißt, du lernst, seine Steuerung zu programmieren. Wir verwenden den Mikrocontroller „micro:bit“, er wurde für Jugendliche entwickelt.

Hardware: Wir bauen unser Fahrzeug mit Teilen aus dem Spielzeugbaukasten „fischertechnik“. Kinder konstruieren damit Autos oder Kräne. Wir bauen zusätzlich die Steuerungselektronik ein.

Elektrischer Strom

In unserem Mikrocontroller+Robotik-Kurs arbeiten wir mit Digitalelektronik, also mit Strom. Weil die Spannung kleiner als 9 V ist, ist sie ungefährlich. Du spürst den Strom gar nicht, wenn du an die Kontakte fasst. Ganz anders als die 230 V aus der Steckdose, sie wären lebensgefährlich.

Damit Strom fließen kann, muss ein „Stromkreis“ geschlossen sein: Die „Stromquelle“, zum Beispiel eine Batterie, treibt den Strom an. Er fließt vom +Pol über ein Kabel zu einem „Verbraucher“, zum Beispiel einem Motor. Auf der anderen Seite des Motors fließt der Strom über ein Kabel zurück zum -Pol der Batterie. Wenn das Kabel irgendwo unterbrochen ist, zum Beispiel mit einem Schalter, dann kann kein Strom im Stromkreis fließen. Dabei ist es egal, ob der Schalter vor oder hinter dem Motor eingebaut ist.

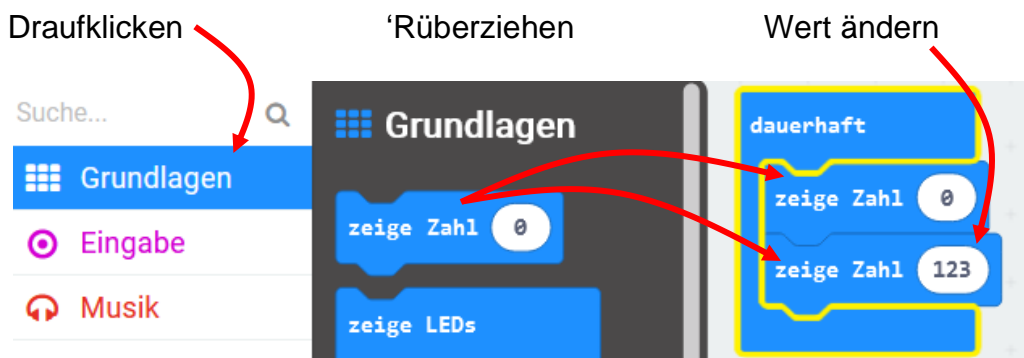
Probiere es aus: Der Motor läuft nur, wenn er mit beiden Kabeln mit der Batterie verbunden ist.

Es ist wichtig, ob ein Kontakt am +Pol oder am -Pol der Stromquelle angeschlossen ist. Damit wir das immer einfach sehen können, verwenden wir für den +Pol rote Kabel und für den -Pol schwarze Kabel (oder so ähnlich).

Probiere es aus: Wenn du die Kontakte am Motor vertauschst, dann dreht sich der Motor in die andere Richtung.

In diesem Kurs lernst du, per Maus zu programmieren: Du setzt die **Blöcke** mit den Anweisungen graphisch zu einem Programm zusammen. <https://makecode.microbit.org/#> startet auf deinem PC die Entwicklungsumgebung für den Microbit. Im linken Bereich wird er simuliert; du kannst deine Programme also auch dann ausprobieren, wenn du keinen Mikrocontroller hast.

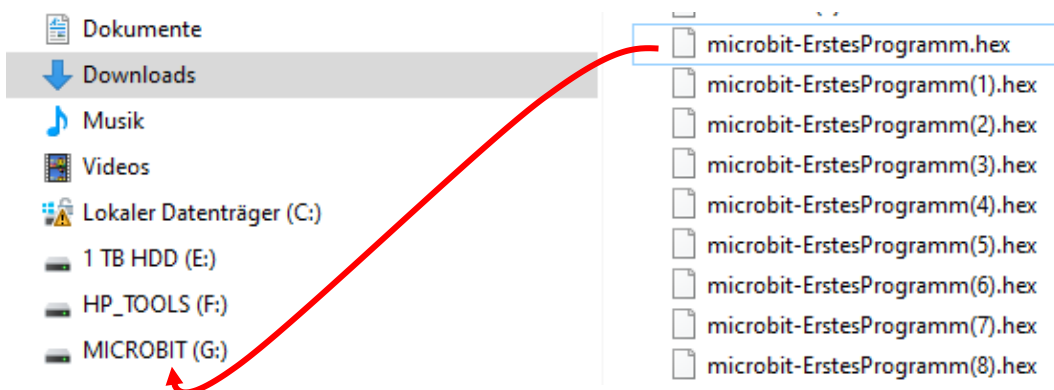
Setze die gewünschten Anweisungen aus den verschiedenen Kategorien in das Programmierfenster. Dazu klickst du mit der Maus auf die blaue Kategorie Grundlagen und ziehst ein paar Anweisungen nach rechts, zum Beispiel das klammerförmige dauerhaft oder zeige Zahl, das du in die Klammer von dauerhaft setzt. Etwa so wie im Bild.



Gib deinem Programm im Schriftfeld rechts von „Herunterladen“ einen Namen und speichere es ab



Danach liegt dein Werk im Downloadordner. Sein Code *.hex ist für uns zwar nicht verständlich, aber das macht nichts. Schließe den Microcontroller über ein Micro-USB-Kabel an deinen PC an. Dann erscheint im Windows-Explorer das Laufwerk MICROBIT. Dorthin verschiebst du deinen .hex-Code, und nach etwas Blinken auf der Rückseite läuft auch schon dein Programm auf ihm.



Probiere es aus: Spiele auch mit anderen Befehlen aus der Kategorie Grundlagen!
 Was bewirkt die Zeitangabe in pausiere? Ihr Wert wird in Millisekunden ($1/1000$ s) angegeben.
 Klicke in die 5x5 Kästchen bei zeige LEDs!

wenn-Verzweigung

In den nächsten beiden Abschnitten entwickeln wir Schritt für Schritt einen Reaktionstest. Aber vorher lernen wir ein paar einfache Anweisungen.

Die Farben der Blöcke zeigen, wo du sie findest: Das violette `Zeichne x ... y ...` steht in der **Kategorie LED** (Jede dieser LEDs ist ein Pixel der Leuchtanzeige. Mehr zu LEDs lernst du später). Setze Anweisung in die Klammer dauerhaft und teste bei `x` und `y` andere Zahlen!

Die andere blaue Klammer beim `Start` wird nur einmal ausgeführt. Probiere auch sie aus!

Gib das nächste Programm ein und starte es, zum Beispiel mit der Taste auf der Rückseite des Microbits: Es wartet erst eine Weile und schaltet danach eine LED ein.

Wenn du anschließend die Taste A drückst, **dann** wird eine 0 angezeigt und als Nächstes mit " " wieder gelöscht.

Anders ausgedrückt: „Knopf A ist geklickt“ ist die **Bedingung** dafür, dass die 0 gezeigt wird.

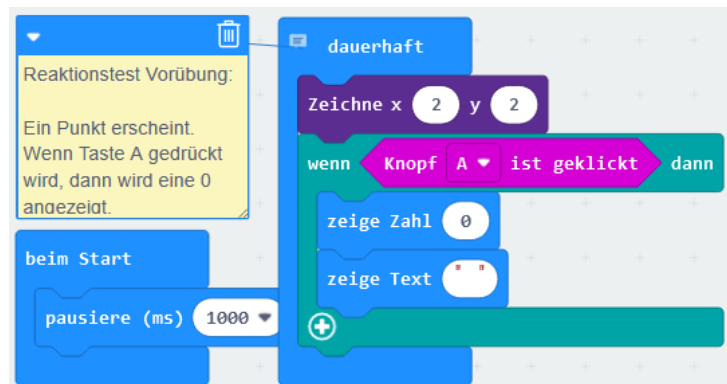
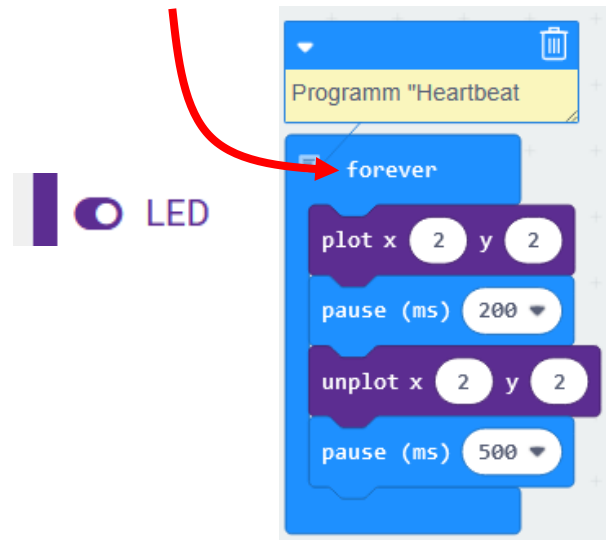
Das ist schon fast ein Reaktionstest. Wenn zusätzlich deine Reaktionszeit angezeigt werden soll, brauchst du Variablen. Der nächste Abschnitt zeigt dir, wie du mit den Microbit Berechnungen ausführen lassen kannst. Du kannst ihn aber auch überspringen, der Roboter fährt ohne Variablen.

Unterscheide beim `Start`, das die Anweisungen nur einmal ausführt, und `dauerhaft`, wo sie immer wiederholt werden!

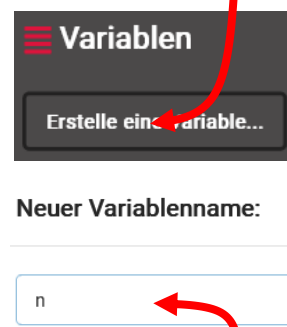
Variablen*

Nun ja, das Programm im vorigen Abschnitt war noch kein richtiger Reaktionstest, weil dort keine Zeit gemessen wird. Dazu muss die Zeit gezählt werden: 1, 2, 3... . Wir nehmen dafür eine Variable.

Beim Zahnradsymbol rechts oben kannst du die Sprache umstellen.



Klick!



Eingabe

1.



1. Erstelle eine Variable in der roten Kategorie Variablen. Gib als Namen n ein und klicke auf OK. Programme wie rechts gezeigt. Du setzt n mit der Maus, nicht mit der Tastatur ein.

2. Das funktioniert, aber es war noch nicht der Renner. Lassen wir den Microbit mal rechnen:

$\dots + \dots$ findest du in der Kategorie Mathematik,

Im Beispiel zählt es bei jedem Durchgang die Variable n um eins hoch: Es beginnt mit $n = 0$ und gibt dem neuen n den Wert des alten n plus 1.

Danach wird $2n$ angezeigt, das sind die geraden Zahlen bzw. das Zweiereinmaleins 2, 4, 6, 8,

Aufgabe 1: Die **ungeraden Zahlen** sind um eins kleiner als die nächste gerade Zahl. Lass sie der Reihe nach anzeigen, 1, 3, 5, 7, ...!

Aufgabe 2: Für 2^n schreibt er $2**n$. Lass die Zweierpotenzen 1, 2, 4, 8, 16, ... ausgeben!

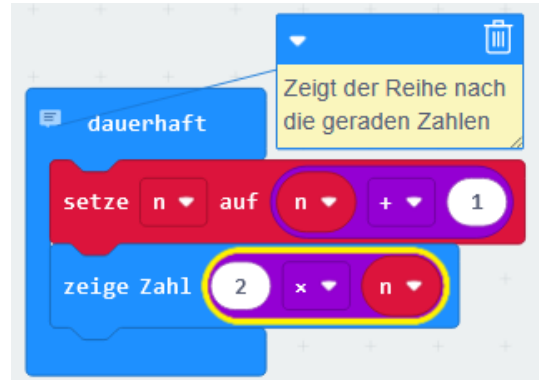
3. Jetzt programmieren wir einen **Handzähler**, wie er zum Beispiel bei Verkehrszählungen verwendet wird: Pro Knopfdruck wird die angezeigte Zahl um eins erhöht. Oder: Knopf A ist geklickt ist die Bedingung dafür, dass n um eins erhöht wird.

4. Zurück zum **Reaktionstest**:

Beim Start wird $t=0$ gesetzt. Die dauerhaft-Schleife schaltet die LED ein, wartet 0,1 s und gibt t dann einen um 1 höheren Wert, also zunächst 1. Im nächsten Durchgang, wieder 0,1 s später, den Wert 2 usw.

Sobald du die Taste A drückst, wird der aktuelle Wert von t angezeigt, z.B. 3; das steht dann für 0,3 s.

2.

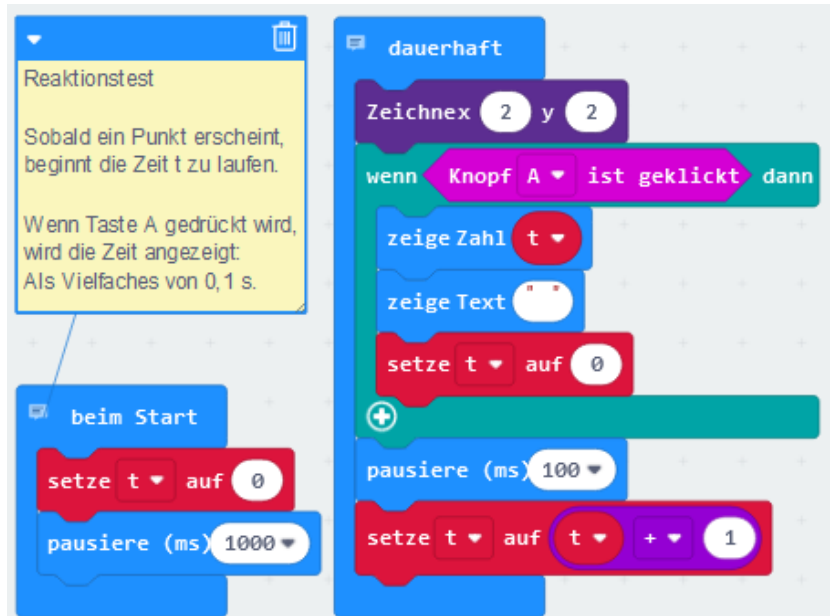


$$n_{\text{neu}} = n_{\text{alt}} + 1$$

3.



4.



Projekt optische Täuschung

Betrachte die vier LEDs in der Abbildung rechts:
Abwechselnd leuchten die Diagonalen.

Um das zu programmieren, beginnst du am besten mit dem Programm „Heartbeat“ zwei Seiten weiter vorher, das die LED $x=2, y=2$ an- und ausschaltet. Dort ergänzst du nun die LED $x=0, y=0$ im Gleichtakt mit der ursprünglichen. Und die LEDs $x=0, y=2$ sowie $x=2, y=0$ im Gegentakt.
(Die Auflösung des Programmcodes ist weiter hinten versteckt :-)

Die optische Täuschung besteht darin, dass das Gehirn zwei Pixel interpretiert, die entweder senkrecht oder waagrecht springen. Diese Interpretation kannst du forciert „umklappen“, wenn du eine Seite mit dem Finger verdeckst.

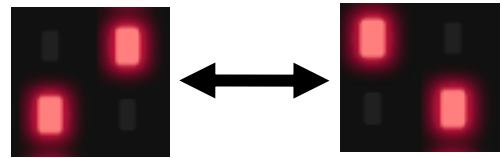
Übrigens würde `Zeige LEDs` die gesamte 5x5-Anzeige zu langsam schalten. Deswegen haben wir `zeichne x ... y ...` gewählt, das ist schneller.

Untersuche, welche Taktfrequenz am interessantesten ist! Wenn du Variablen kennst, beispielsweise `pausiere t`, dann kannst du diese Geschwindigkeit vielleicht sogar mit Taste A schneller und mit Taste B langsamer machen.

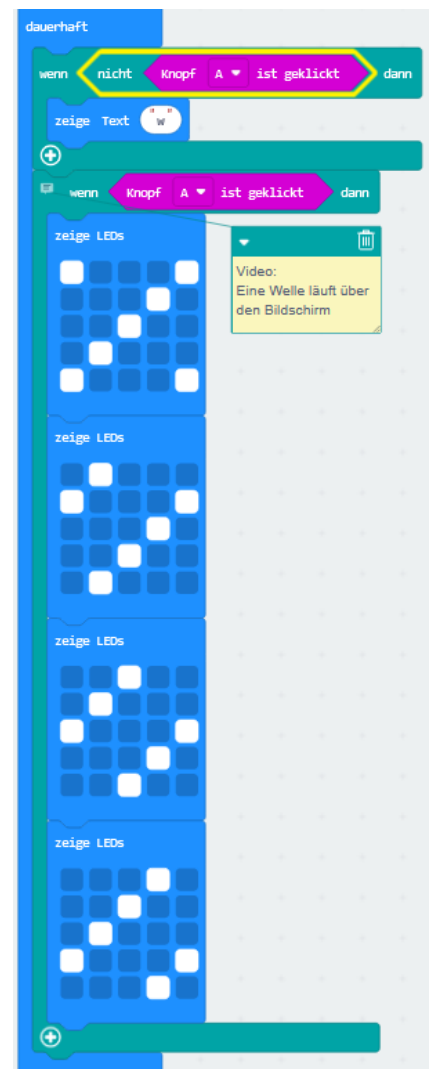
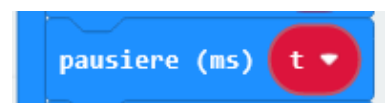
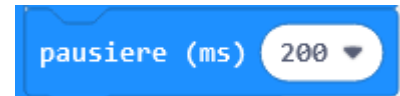
Noch eine einfache Übung:

Erstelle ein Programm, das beim Betätigen der Taste A ein von dir gewähltes Bild 1 anzeigt, bei Taste B ein Bild 2 und sonst ein Bild 3 !

Das abgebildete Programm stellt ein Art **Videofilm** einer Welle dar, die nach rechts unten läuft.



Tipp:

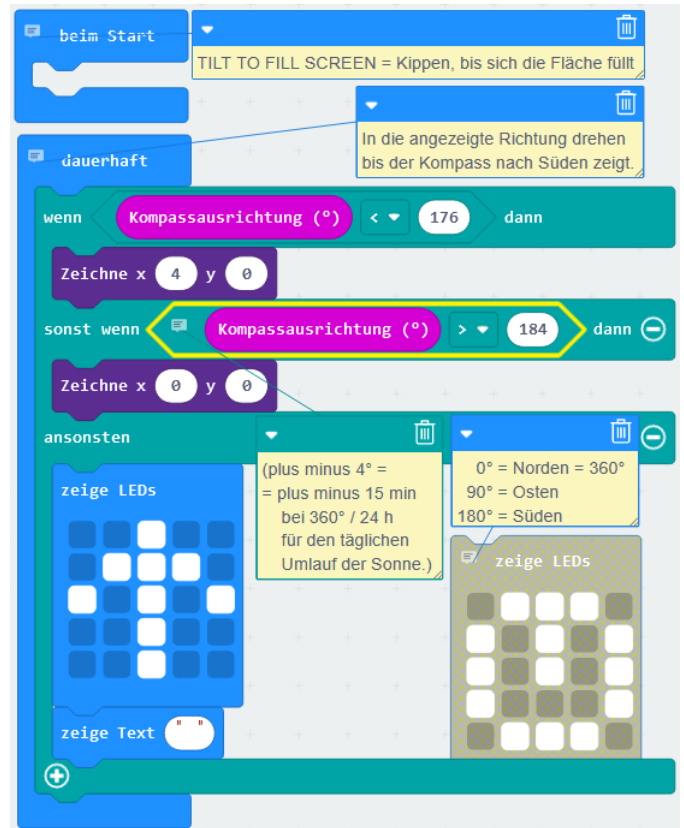


Magnetfeld

Um den Roboter zu orientieren, ist ein **Kompass** praktisch. Im Microbit ist einer eingebaut, leider spinnt er, wenn in der Nähe ein Elektromotor läuft. Die Kompassrichtung gibt dir eine Zahl zwischen 0° und 360° . Du kannst sie für die Bedingung in der während-Schleife verwenden

Im Beispielprogramm meldet dir eine Markierung in der Anzeige, in welche Richtung du den Microbit drehen sollst, bis der Pfeil schließlich nach Süden zeigt.

Aber beachte: Jedes Mal, nachdem du ein neues Programm überspielt hast, musst du den Kompass neu kalibrieren. Die Anzeige fordert dich auf mit „TILT TO FILL SCREEN“: Du kippst ihn so lange, bis der blinkende Punkt das ganze Quadrat ausgefüllt hat. Probiere es einfach!



solange-Schleife

Eine recht praktische Anweisung ist während *Bedingung* mache.

In unserem dreistündigen Kurs reicht die Zeit nicht, um sie zu üben. Aber bekommst du selber heraus, was es mit ihr auf sich hat?

während *wahr* ergibt eine **Endlosschleife**.

Mit wenn anstelle von während wäre die LED bei Tastendruck hier immer nur so kurz an, dass du sie nicht wahrnehmen könntest. Da müsstest du noch eine Pause einbauen. Probiere es gerne aus!



Das geht nicht:

Es gibt auch **rosafarbene Klammern** für die Eingabe. Aber wie das Beispiel zeigt, funktionieren sie oft nicht, denn der Buchstabe A erscheint hier nie.

+^{des/2} verzichten wir auf „nebenläufige Programme“ in der Programmierumgebung von Microsoft makecode.



Analoge Ausgaben

Jetzt schließen wir Elektronik an deinen Mikrocontroller an, dafür ist er schließlich da. Er hat an der Unterkante eine Reihe von Kontakten. Fünf davon sind etwas breiter und tragen Namen: 0, 1 usw. Verbinde eine Leuchtdiode (LED) mit zwei Krokodilkabeln mit 3 V und GND („Ground“)

Leuchtet die LED oder nicht? Klappst es bei deiner Nachbargruppe? Offenbar gibt es noch etwas zu lernen: Vertausche die beiden Anschlüsse. Merke dir, wo das etwas kürzere Beinchen sein muss bzw. wo die Klemme unterbrochen ist! Dort ist die **Kathode**, die mit **GND** bzw. dem negativen Pol verbunden sein muss, damit Strom fließt.

(Die Erklärung dafür ist, dass eine Diode Strom nur in der einen Richtung durchlässt. Im Schaltplan zeigt die Spitze des Dreiecks dann von + nach -

Lass die LED nicht zu lange an 3 V leuchten, die Spannung ist zu groß und macht sie dann kaputt!

Jetzt wollen wir die LED mit dem Mikrocontroller ein- und ausschalten. Dazu brauchen wir ein Programm, das Pin P1 wahlweise auf 1,5 V oder auf 0 V legt, siehe rechts. Die Kategorie Pins klappt du unten bei Fortgeschritten auf.

- Pin P1 ist der große messingfarbene Kontakt 1,
- für 0 V steht im Programmcode der Wert 0.
- für 1,5 V steht im Programmcode der Wert 511, das ist etwa die Hälfte des Maximalwerts 1023.

Wenn du die Leuchtdiode richtig gepolt zwischen Pin P1 und GND anschließt, dann leuchtet sie, sobald Pin P1 eingeschaltet (hier 1,5 V) ist.

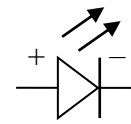
Aufgabe für Fortgeschrittene: Die Taste A soll eine LED abwechselnd an- und ausschalten.

Tip: Du könntest eine Variable $a = 1$ setzen, mit der du weißt, ob die LED leuchtet. Jeder Tastendruck muss sie mit -1 multiplizieren, dann wechselt sie wie die LED immer hin und her: 1, -1 , 1, -1 , ...

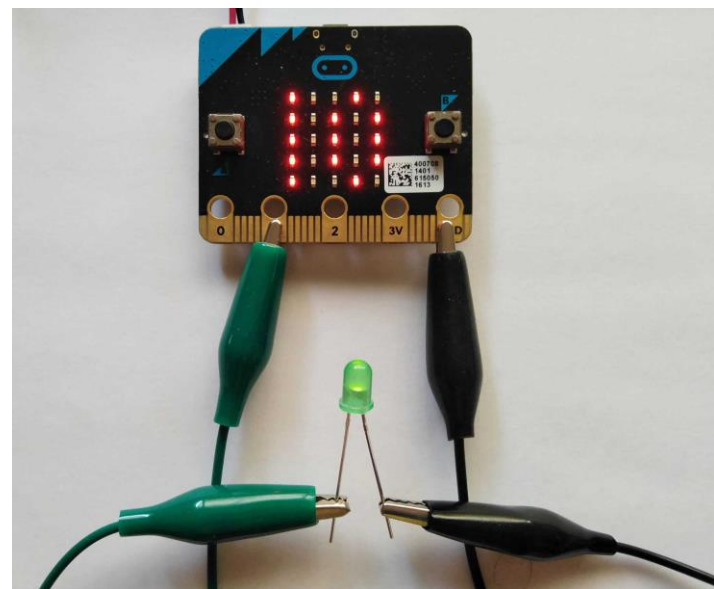
„Analog“ heißt, zwischen dem höchsten und dem niedrigsten Wert (hier 1023 und 0) gibt es auch alle anderen. Probiere gerne mal 111 aus! (→ Anhang, PWM)



LED



Kathode:
Dicker Strich bzw. kurzes Beinchen, -



Mit „beim Start“ würde es stattdessen nur einmal ablaufen. $1023 \cdot 1,5 \text{ V} / 3 \text{ V} \approx 511$



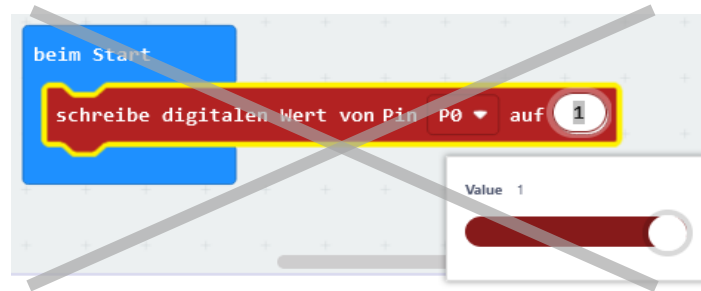
Zwei Pins für eine rote und eine grüne LED:
 Programmiere damit eine **Fußgängerampel!**
 Meistens zeigt sie rot, aber wenn du die Taste drückst, springt sie nach kurzer Zeit auf grün.

Oder: Hast du schon einmal beachtet, wie eine Autoampel von rot über gelb auf grün schaltet?

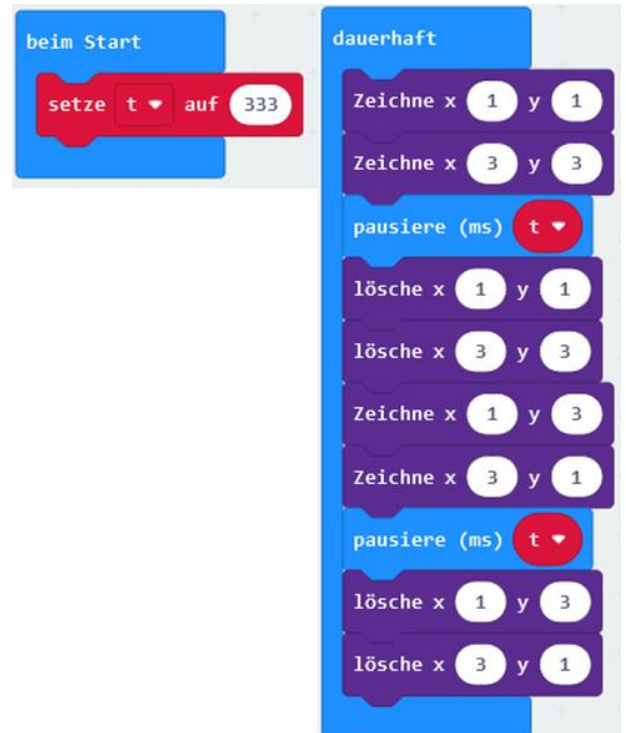


Fehlersuche

- Verwendest du wirklich den dauerhaft-Block?
- Wir legen das kurze Beinchen der LED immer auf GND und schalten das lange Beinchen.
- Schreibe digitalen Wert würde im Gegensatz zum analogen nur *Ein* = 3 V und *Aus* = GND zulassen, keine Abstufungen dazwischen.



Und da ist noch ein Programm von vorhin; hast du es selbst herausbekommen?



Ein „Micro Bit 2 Go Set“ gibt es beispielsweise bei <https://www.pollin.de/search?query=Microbit> .

Dieses Script findest du auch bei www.physik.de.rs/schule/robotik

Kontakt:
kurs@nueland.de

„Digital“ heißt, es gibt keine stufenlosen, kontinuierlichen Werte, sondern zum Beispiel nur 0 und 1 bzw. *aus* und *an*. Wir haben bereits eine digitale Eingabe kennengelernt, nämlich Knopf A ist gedrückt oder nicht gedrückt.

Jetzt verwenden wir die Pins für digitale Eingaben. Stell das rechts abgebildete Programm zusammen!

„Wenn Pin P0 ist gedrückt“?
 Programmierer haben ein schlechtes Sprachgefühl, am Pin gibt's nichts zu drücken. Die Bedingung sollte treffender lauten „wenn **Pin P0 mit GND verbunden** wird“.

Ein Teil unseres Stromkreises von Pin P0 zu GND besteht aus Alufolie, etwa Schokoladenpapier.

Aufgabe: Baue dir eine **Alarmanlage**, die anfangs ein „-“ anzeigt und dann zu „!“ wechselt, wenn die Alufolie zerrissen wurde! Mit dieser könntest du zum Beispiel deine Türe versiegeln. Dann zeigt dir der Alarm, dass sie geöffnet wurde. Und so geht es:

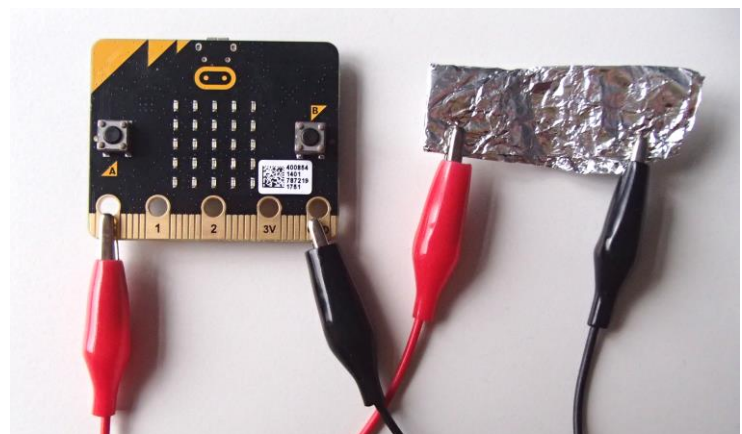
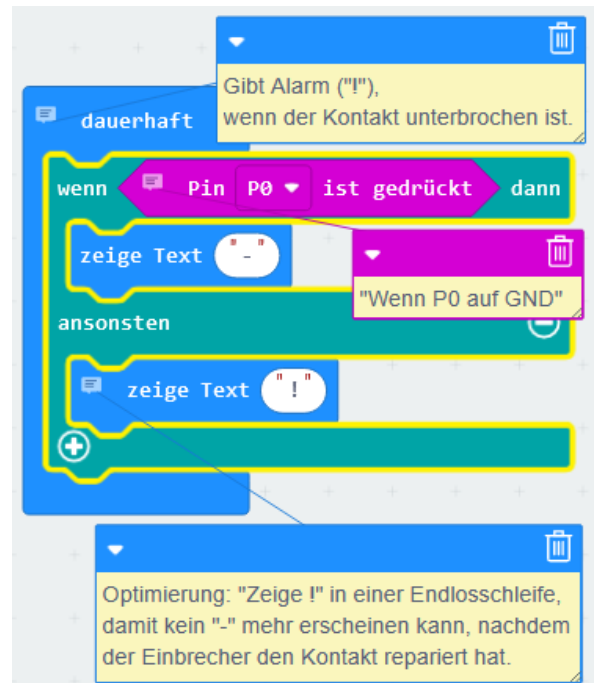
Die Alufolie und die Krokodilkabel verbinden Pin P0 und GND. In der wenn-Verzweigung ist also die Bedingung „Pin P0 ist gedrückt“ erfüllt. Wenn die Alufolie zerrissen wird, sind Pin P0 und GND nicht mehr verbunden.

Dann ist die gegenteilige Bedingung erfüllt, der Bereich von ansonsten trifft zu.

Damit du beweglicher wirst, ziehst du das USB-Kabel ab und verwendest nun den Batteriekasten.

Digital

Mathematisch:	0	1
Physikalisch:	GND	3 V
Schalter:	aus	ein
Licht:	aus	an
Logisch:	falsch	wahr
...



Pins

Lesen und schreiben (GPIO): P0, P1, P2

Tasten A bzw. B: gleichwertig zu P5, P11

Analog schreiben: P5, P8, P11, P13 bis P16

Wiederholung und weiter

Letztes Mal haben wir gelernt

- der Stromkreis braucht zwei Leitungen: Eine von der Quelle zum Verbraucher und eine zurück,
- wenn du den Motor umpolst, dreht er sich in die andere Richtung,
- du kannst Programme auf den Microbit laden,
- die Anweisungen der wenn-Verzweigung werden nur ausgeführt, wenn die *Bedingung* erfüllt ist,
- du kannst eine LED an GND und einen Pin anschließen, den du programmierst,
- LEDs leuchten bzw. leiten Strom nur in einer Richtung: Kurzes Beinchen auf GND.
- Pins lassen sich auch für Eingaben nutzen.

Wir üben das mit einer **Fußgängerampel**. Sie steht normalerweise auf Rot. Kurze Zeit nachdem jemand den externen **Knopf** drückt, schaltet sie eine Weile auf Grün und dann zurück.

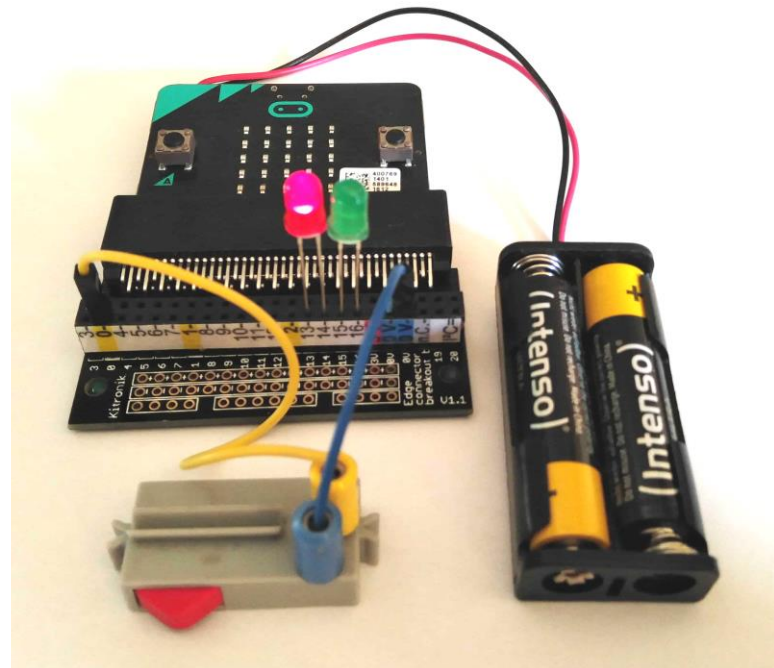
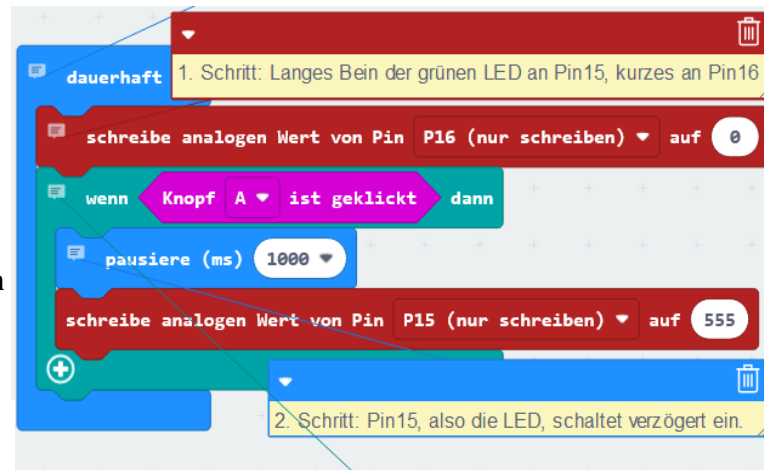
Damit wir die Ampelschaltung später leichter zur Robotersteuerung umbauen können, schließen wir die grüne LED zwischen Pin P15 und Pin P16 an (nicht zwischen einem Pin und GND).

Heute verwenden wir keine Krokodilkabel, sondern den **Platinenstecker** (Breakout Board). Damit erreichst du auch die anderen Pins des Microbit. In die Buchsen passen etwa die Stecker der Verbindungsleitungen oder direkt die LEDs.

Programmiere Schritt für Schritt und teste immer wieder aus, ob es funktioniert:

1. Schalte die grüne LED mit der Taste A an (das hatten wir ähnlich auf Seite 7),
2. die grüne LED soll erst verzögert einschalten,
3. programmiere außerdem eine Verzögerung, bevor die grüne LED wieder ausschaltet.
4. Die rote LED leuchtet, wenn die grüne aus ist. Stecke sie dazu mit dem kurzen Bein in Pin 15, mit dem langen in Pin P16 und setze an der richtigen Stelle Pin P16 auf 1023!
5. Schritt: Statt Taste A setzt du einen externen Schalter zwischen **Pin P5** und GND. (→ Anhang, Taster)

Die Pins P5 und P11 wirken wie die Tasten A und B. Wenn du also Pin P5 mit GND verbindest, wirkt das so, wie wenn du die **Taste A** drückst.



Nur sehr kleine Motoren kann der Microbit direkt betreiben. Durch größere müsste so viel Strom fließen, dass es ihn zerstören würde. Dafür gibt eine extra Platine, den „**Motortreiber**“. Er braucht

- ein Signal vom Microbit, wie sich der Motor drehen soll (zwei Eingänge),
- den Anschluss für den Motor (zwei Ausgänge),
- zusätzliche Energieversorgung für den Motor.

Los geht's:

- Stecke für den Anfang eine Kabelbrücke vom Microbit 3 V an den Motortreiber IN1 und von GND an IN2.
 - Verbinde den Motor mit H0-Steckern („H null“) und Kabel mit den beiden Buchsen „MOTOR-A“.
 - Lass die Schaltung vom Betreuer kontrollieren.
 - Schließe den 9 V-Block an + (rot) und - (blau).
- Der Motor dreht sich.
Vertausche IN1 und IN2:
→ Der Motor kehrt seine Drehrichtung um.

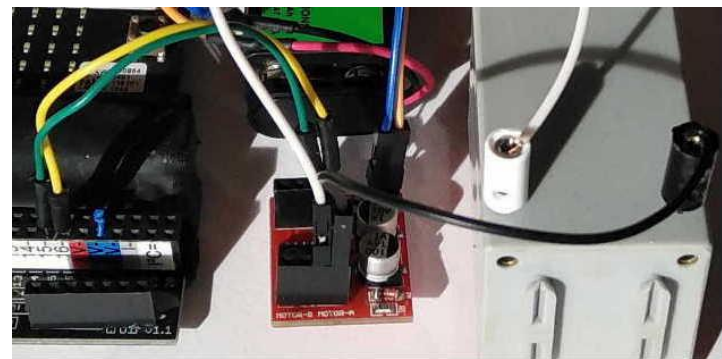
Immer nur 3 V und GND wären langweilig. Der **Microbit** soll steuern, wie sich der Motor dreht! Du brauchst zum Beispiel ein Programm, das Pin 15 auf 2 V und Pin 16 auf 0 V (= GND) legt. Danach soll ein Taster die Polung umschalten können, so dass sich der Motor kurz in die andere Richtung dreht. Schau auf die vorige Seite: Dieses Programm hast du bereits fertig!

Dein erster Roboter

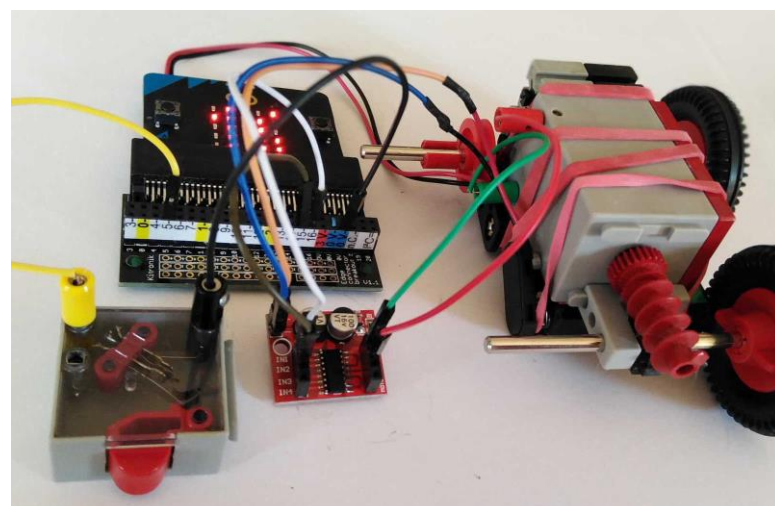
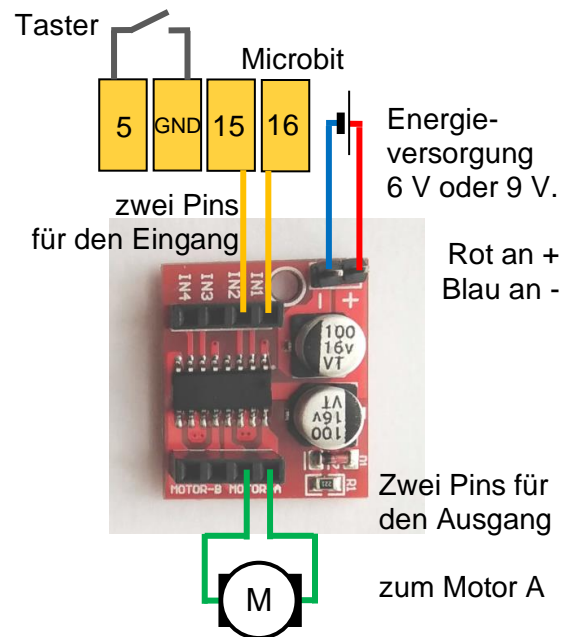
Baue mit dem Technikbaukasten einen Wagen, der von einem Elektromotor angetrieben wird! Ganz nach vorne setzt du den Taster, so dass er gedrückt wird, wenn der Wagen gegen ein Hindernis fährt. Und dann baust du die oben beschriebene elektronische Steuerung ein.

Funktioniert es? Nicht schlecht für den Anfang! Aber bald erkennst du: Es könnte besser sein, z.B. nicht immer wieder neu gegen die Wand fahren. Dann entwickle ein besseres Programm :-)

Microbit Akku H0-Stecker



Platinenstecker Motortreiber Motor



Ein Sensor, ein Motor

Der Roboter fährt nur vor oder zurück.

Für jede Gruppe ist bereits ein Fahrzeug zusammengebaut. Deine Aufgabe ist es, die Elektronik, die Verkabelung und die Programmierung einzubauen.

Wettbewerb: Du programmierst ihn so, dass er von der Startlinie bis zur Wand fährt und danach zurück zur Startlinie.

Fehler beheben

Der **Microbit** tut nicht, was er soll?

Nimm statt Motortreiber und Motor nochmal die rote und die grüne Leuchtdiode. Tun sie, was sie bei der Fußgängerampel tun sollten?

Ist der **Akku** geladen? Schließe den Motor zur Probe direkt an, ohne Microbit und Motortreiber.

Schreibe an eine sinnvolle Stelle im **Programm** zeige Text "x" oder ähnliches. Dann siehst du, wann das Programm dort ist, und kannst den Fehler eingrenzen.

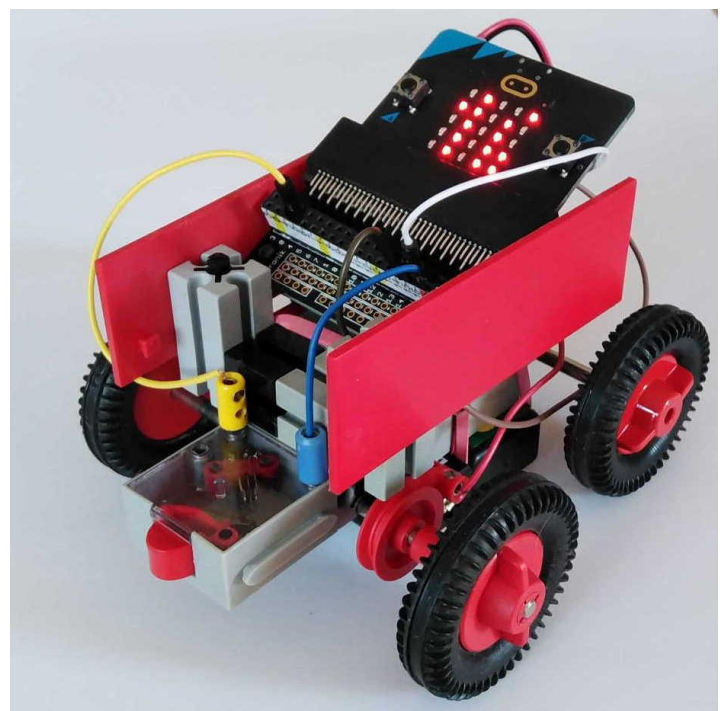
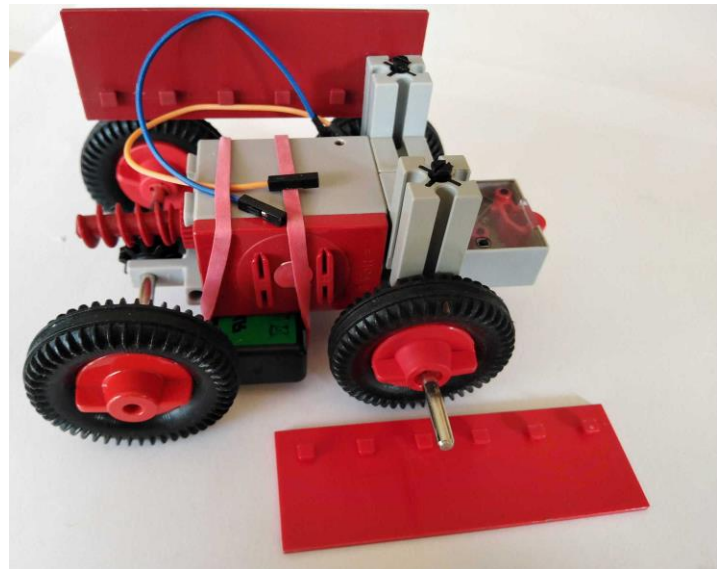
Der Microbit macht eigentlich gar nichts?
Hast du das abgespeicherte Programm aus dem **Downloadordner** auf ihn geladen?

Alles kontrolliert, funktioniert trotzdem nicht?
Vielleicht ist ein **Bauteil** defekt oder ein Kabel gebrochen. Nimm einen anderen Motortreiber! Tausche auch die anderen Teile Stück für Stück aus, bis alles aus neuen Komponenten besteht!

Der Wagen fällt in der **Kurve** leicht um?
Befestige die Akkus so tief wie möglich.

Du kommst mit den **Kabeln** durcheinander?
Verwende verschiedene Farben; 3 V ist rot, GND („Ground“ = Erde) blau oder schwarz.

Die **H0-Stecker** halten nicht in den Buchsen?
Biege sie mit einem Messer ganz wenig auf.



Tanzroboter

Zwei Motoren, zwei Sensoren:

Kurvenfahren geht mit der Panzersteuerung:

Wenn sich der linke Motor schneller dreht als der rechte, fährt er eine Rechtskurve. Und umgekehrt. Gut, dass der Motortreiber noch freie Buchsen hat und zwei Motoren ansteuern kann! Beim Microbit programmierst du dafür Pin 13 und 14.

Wettbewerb: 4 Meter von der Startlinie entfernt steht ein Baustein. Wer fährt am schnellsten um ihn herum und wieder zurück?

Mehr Ideen

Lass den Taster nach unten zeigen, so dass der **Boden** den Knopf drückt. Wenn er nicht mehr gedrückt wird, könnte der Grund das Ende der Tischplatte sein. Was soll der Roboter dann tun?

Auf Seite 4 hast du die deine Reaktionszeit ermittelt. Jetzt könnte ein **Schrittzähler** messen, wie weit dein Roboter gefahren ist.

Ein **Arm**, der sich die ganze Zeit dreht, könnte herumliegende Dinge einsammeln.

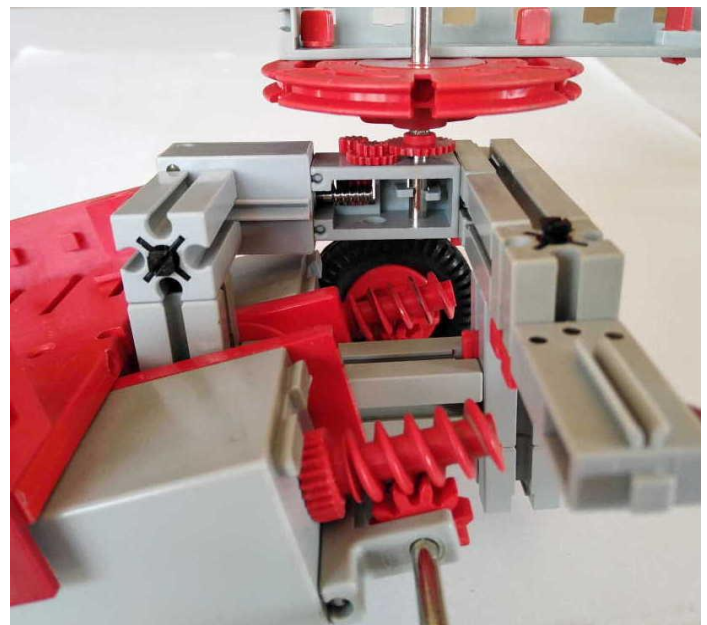
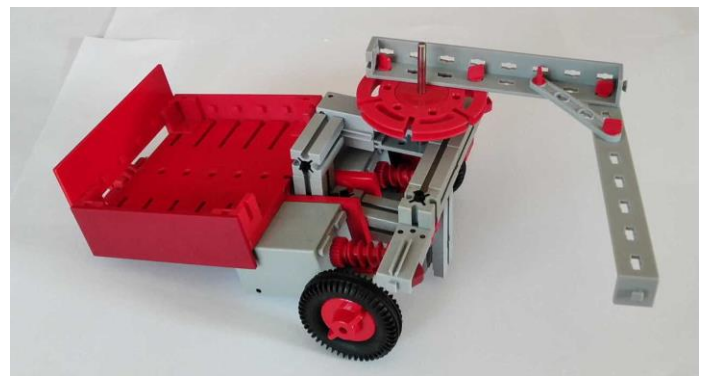
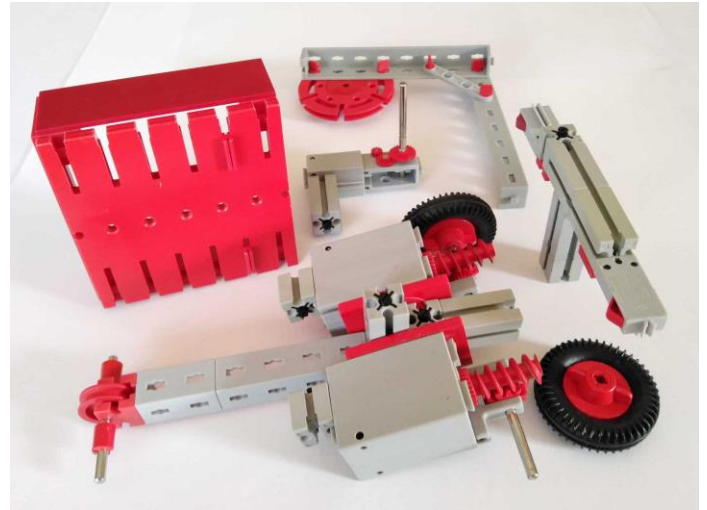
Beachte: Wenn du den Roboter mit dem **Kompass** steuern willst, muss der Motor in diesem Moment stillstehen, denn er beeinflusst das Magnetfeld.

In der Ecke des Zimmers liegt ein **Luftballon**. Befestige an der Spitze des Roboters eine Nadel ...

Das passende Bauteil gibt es nicht im Technikkasten? Scheue dich nicht vor **Holz und Schrauben**, Heißkleber usw.!

Wettbewerb: Du bekommst einen Magneten, auf dem Boden liegen Schrauben. Du gewinnst Obst oder so viele Stückchen Schokolade, wie du innerhalb einer Minute an Schrauben einsammeln lässt.

*Schön, dass du bis hierher gekommen bist.
Schau mal, was du im Internet zum Microbit
findest und mach's gut!*



Der Kurs entstand in Zusammenarbeit von
C JT-Gymnasium Lauf und dem
FabLab Nürnberger Land e.V.



Anhang

Stückliste

Kosten
/Gruppe

Für den 1. Tag in Zweiergruppen:

- | | |
|---|----------|
| • Microbit bzw. lieber das Set mit USB-Kabel und Batteriekasten | 31,51 € |
| • rote und grüne LED 5 mm | 0,10 € |
| • drei oder vier Krokodilkabel | 3 € /3 |
| • Alufolie , Gewebeband | recycelt |
| • Laptop mit Internetanschluss, genügend Mehrfachsteckdosen für die Netzteile | Bestand |
| • Programmierumgebung auf microbit.org | frei |

Für den 2. Tag zusätzlich; erst in Zweier-, dann in Vierergruppen:

- | | |
|---|------------------|
| • Platinenstecker = Breakout Board (schwierig wg. Brexit , sonst aus D, D),
zum Beispiel mit selbst aufgelöteter Buchsenleiste oder Steckbrett | 5,52 €
1,65 € |
| • Verbindungsleitungen / Jumperkabel (mm, mw, ww je nach Komponenten) | 3 € /4 |
| • Motortreiber (Buchsen und Pins angelötet) oder verlässlicher bei ebay | 2,00 € |
| • Akku 9 V-Block , Ladegerät | 6,70 € |
| • Clip für 9 V-Block, selbst angelötete Kupplung ; Schrumpfschlauch | ≈ 0,7 € |
| • Elektromotor | ≈ 7 € |
| • 4 Kabel H0-Stecker auf Pin (auf Verbindungsleitung, siehe oben, schrauben) | 2,80 € |
| • Taster | schwankend |
| • bereits fertiggebauter Roboter 1 (ein Sensor, ein Motor) für jede Vierergruppe | offen |

Weiterbauen mit offenem Ende:

- [Fischertechnik-Sammlung](#), insbesondere zweiter Motor, günstig gebraucht
- Klebeband und weiteres Werkmaterial, Holz, Schrauben etc.
- Preise für den Wettbewerb

Erweiterte Fertigkeiten:

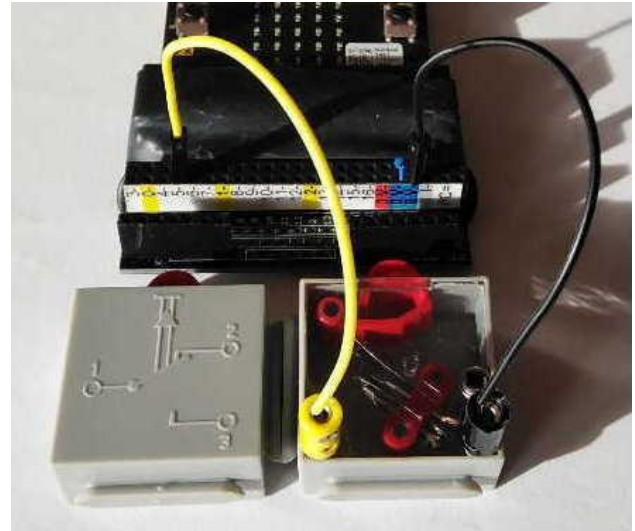
- Abisolieren
- H0-Stecker anschrauben, aufspreizen: Kleiner Schraubenzieher, Cuttermesser
- Löten, Schrumpfschlauch anbringen

Günstiger werden Sie sich schwerlich ausstatten können als beispielsweise mit zwei Leuchtdioden für je 5 Cent anstelle eines gekauften Ampelsets oder mit einer Riesensammlung an gebrauchtem Technikspielzeug anstelle von Apothekenpreisen für eine geringe Anzahl fabrikneuer Bausteine.

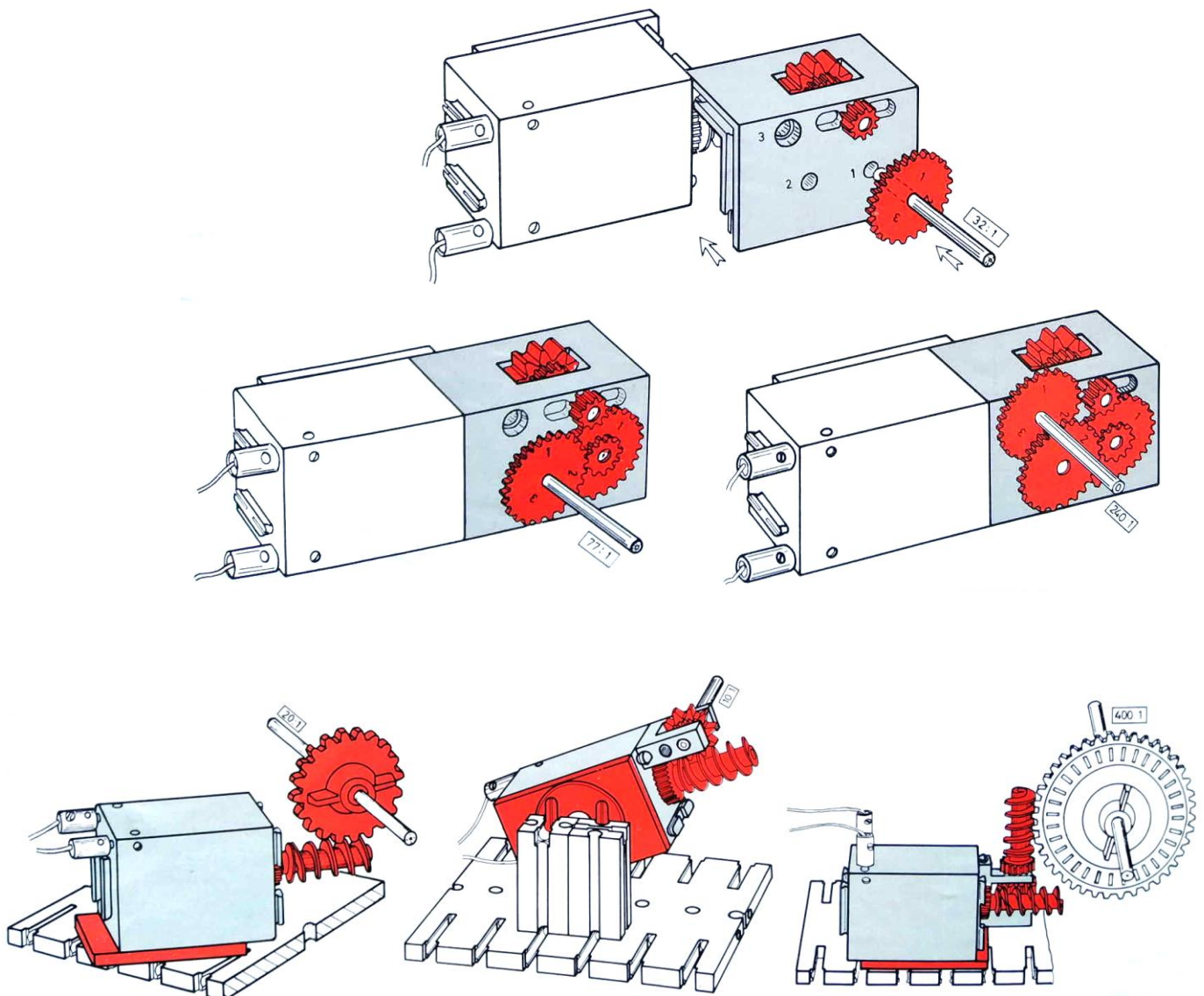
Anschließen des Ein-Tasters (1-3):

Auf der Rückseite des Gehäuses ist das Schaltbild gezeichnet. Nur wenn der Taster gedrückt ist, sind die Anschlüsse 1 und 3 verbunden.

Die Verbindung 1 und 2 wird bei Tastendruck geöffnet. Mit diesen Anschlüssen wäre es ein Aus-Taster.



Aus einem Anleitungsblatt für den Motor und das Getriebe:



1.) „Analog“ heißt hier, es gibt nicht nur die Werte 0 und 1 bzw. GND und 3 V, sondern einen (fast) stufenlosen Übergang. Die Anweisung `schreibe analogen Wert von Pin P1 auf ...` passt dazu. Mit Zahlen von 0 bis 1023 soll er am Pin P1 **Spannungen von 0 V bis 3 V** (gegenüber GND) ausgeben. Du überlegst dir leicht, dass das in kleinen Stufen von je $3V/1023 \approx 0,003 V$ passiert.

Übung a: Per Taste sollen an Pin P1 verschiedene Spannungen eingestellt werden können, z.B. 0,5 V, 1 V, ... (Wenn dir das zu schwer ist, nimm das abgebildete, einfachere Programm mit „170“).

Übung b: Kontrolliere mit einem Multimeter!

Übung c: Schließe eine rote LED an Pin P1 und GND an und steuere so ihre Helligkeit!

Ergebnis: Wenn du nur 0,5 V an die LED legst, scheint sie schwach.

Aber halt, da ist etwas faul: Unter der Schwellenspannung kann eine LED doch gar nicht leuchten!?

Ist das Ergebnis auch bei der eingebauten LED-Matrix so? Einzelne Pixel kannst du mit `zeichne x 0 y 0 Helligkeit ...` ansprechen. Setze eines auf den Maximalwert 255 und eines auf 1: Du siehst das gleiche Phänomen!

2.) Pulsweitenmodulation PWM

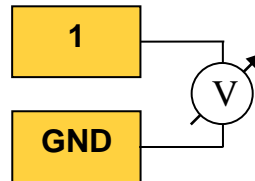
Intern kann der Microbit wirklich nur 0 V und 3 V. Deswegen simuliert er eine schwache LED, indem er sie in sehr kurzen Intervallen nur einen Teil der Zeit hell läuchten lässt. Für's Auge ist' wurschd.

Versuch: Nimm den Microbit mit dem obigen Programm an die Batterie und bewege ihn schnell! Dann siehst du keine durchgehende Leuchtspur, sondern erkennst, dass die schwache LED in Wirklichkeit blinkt!

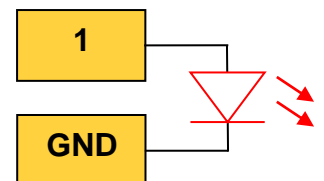
(Um PWM überzeugend selbst nachzuprogrammieren ist [makecode](#) aber leider zu langsam.)



b:

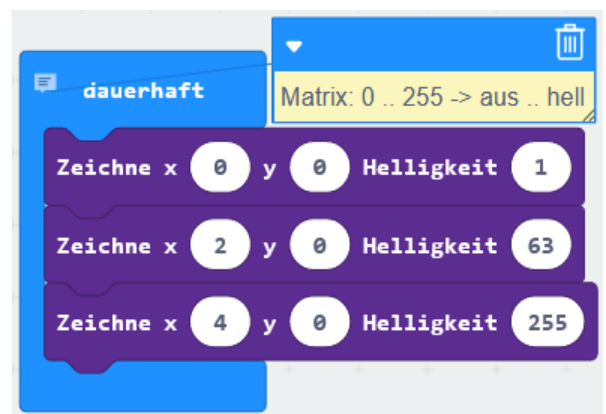


c:

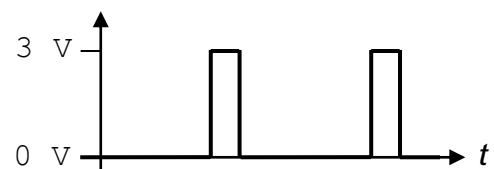


b: Ein Multimeter gibt es in der Schulsammlung.

c: Das kurze Beinchen (die Kathode) an GND.



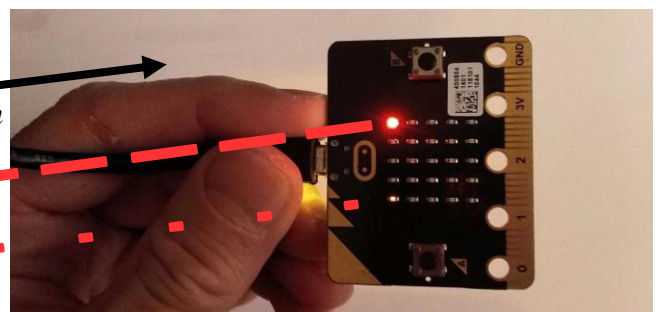
Pin P1



Schnell hin und her schwenken

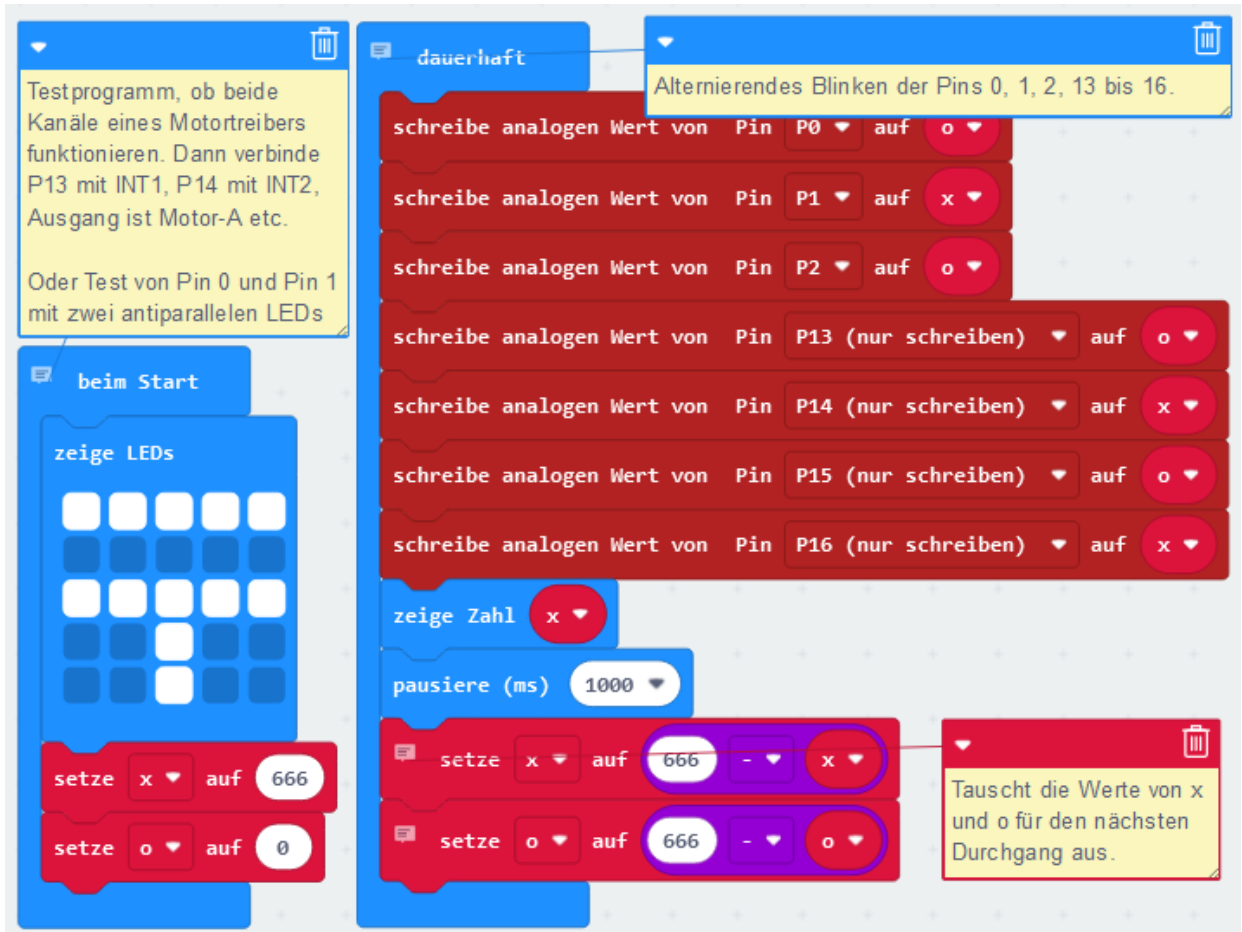
Leuchtspur helle LED

Leuchtspur schwache LED

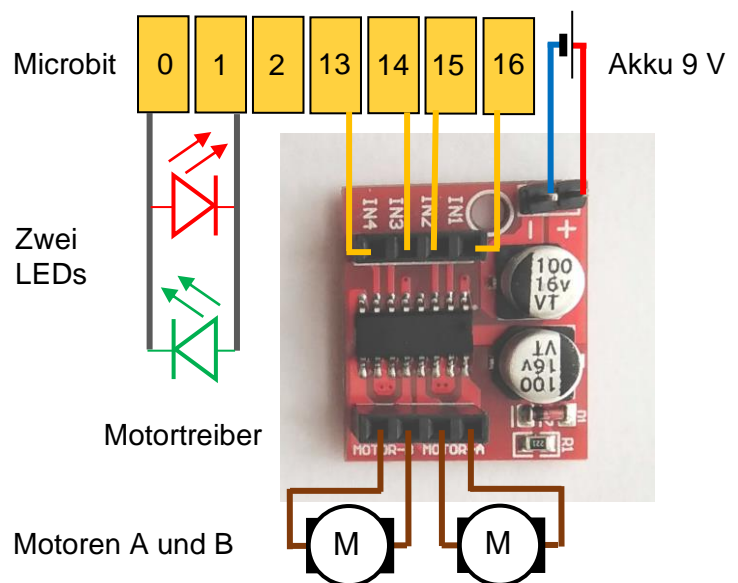


Zum Überprüfen von Pins am Microbit oder der Funktion des Motortreibers

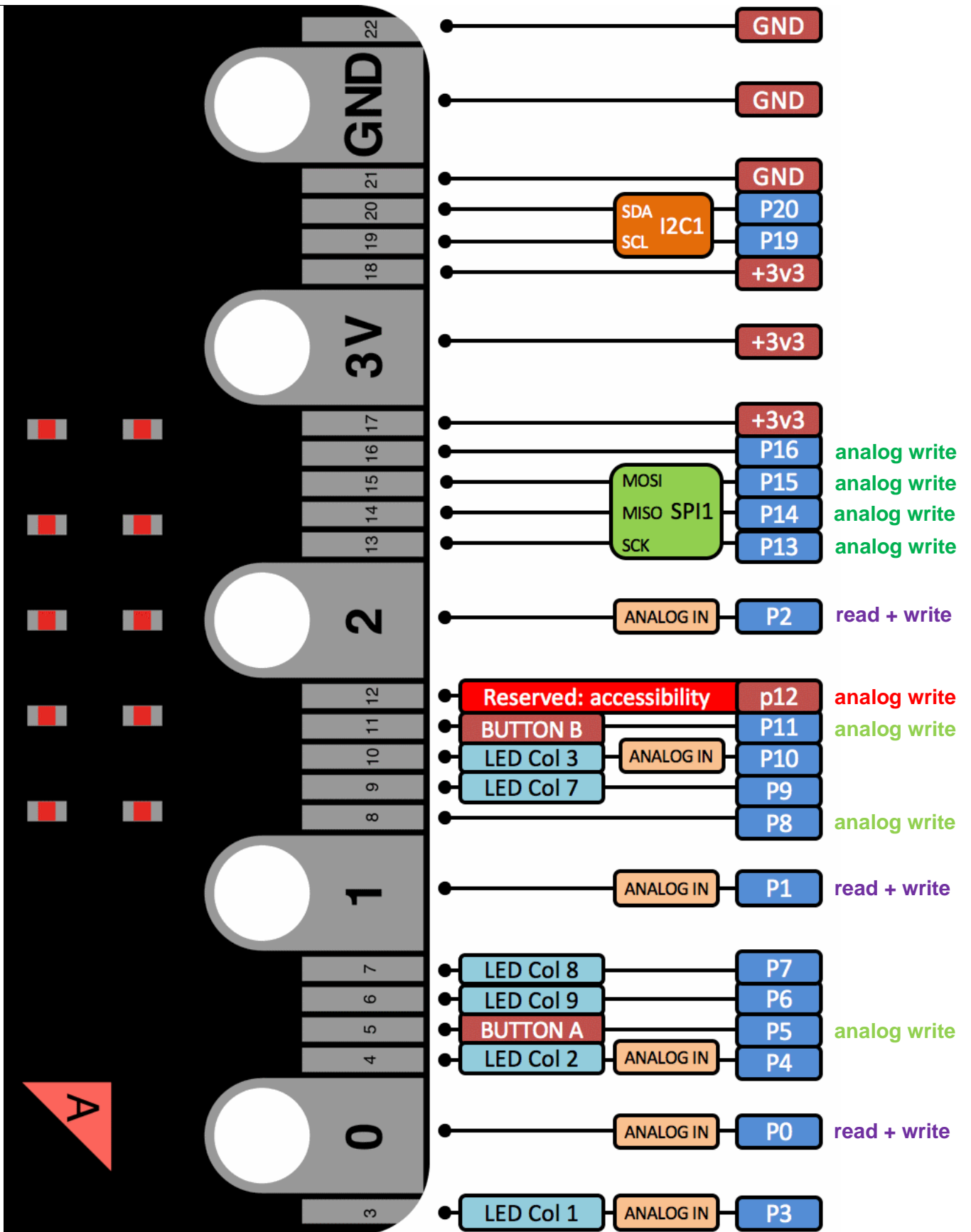
Programm:



Aufbau:



Tatsächlich beträgt die Nennspannung des Microbit 3,3 V, unter Last weniger. In diesem Script spreche ich der Einfachheit halber immer von 3 V.



a) Der kostengünstige und sehr erfolgreiche englische **Microbit** (<https://archive.microbit.org/de/>) ist in der deutschen Didaktik nahezu unbekannt. Stattdessen wurde er nachgemacht und steht als doppelt so teures Teil zum Verkauf (Vergleich siehe nächste Seite. Calliope <https://calliope.cc/>). Ihm fehlt die sehr wichtige Pinleiste an der Unterkante für einen einfachen Steckanschluss zu den weiteren Pins. Damit gehen viele interessante und weitreichende Möglichkeiten verloren.

b) Wenn du etwas professioneller weitermachen willst, dann schau mal nach „**Arduino**“!

c) Der Microbit lässt sich auch in anderen **Entwicklungsumgebungen** blockbasiert programmieren. Hier ein Vergleich von [makecode](https://makecode.com) mit <http://microblocks.fun/> und <https://www.open-roberta.org/> .

Kategorie/Editor	MakeCode primärer Editor von Microsoft	Open Roberta Lab	microBlocks (Alpha)
Programmier-Paradigma	ereignisbasiert	sequentiell	ereignisbasiert
Programm-Ausführung	direkt im Editor über Simulator	nur per copy-and-paste auf dem verbundenen Controller	in Echtzeit auf dem angeschlossenen Mikrocontroller
Nebenläufigkeit	umständlicher Work-around mittels warte- und im-Hintergrund-Anweisung	kaum möglich, da sequentiell!	klar kontrollierbar 😊
RF-Funk	gut zum Versenden von Zahlwerten mit Label zur Identifikation	nur Strings versendbar	inzwischen ebenfalls implementiert
Bedienbarkeit und Komfortfunktionen	sehr gut, viele Spielereien (Display-Symbole) vorhanden	ausgereift, aber weniger umfassend als MakeCode	noch relativ spartanisch, da im Alpha-Stadium
Umsetzung	relativ langsam	relativ langsam	schnell genug, um PWM umzusetzen

nach: Manuel Riel, Physical Computing, Stand 2019
 Rote Felder: Schlecht. Grüne Felder: Gut gelöst

d) Statt mit Blöcken lässt sich der Microbit auch in Sprachen wie etwa [Python](#) programmieren.

e) Es gibt viele interessante **Projekte** und Kurse im Internet, zum Beispiel links auf [diesen Seiten](#). Dort lässt sich auch ein tolles [Buch](#) herunterladen. Oder hervorragendes Material aus England.

f) **Lego** oder **fischertechnik**? Deren Original-Roboterbausätze sind sehr gut und sehr teuer. Du musst dich nicht mehr um Kleinzeug wie Pinbelegungen kümmern. Aber ist das ein Vorteil? Lego setzt auf sein eigenes, mit günstiger Elektronik weniger kompatibles Steckersystem.

Physical Computing mit BBC micro:bit und Calliope mini

- erprobt in **9. Jahrgangsstufen** in einer fächerverbindenden Unterrichtsreihe von Informatik und Physik
- konzipiert als Einführung in die **Mikro- bzw. Digitalelektronik** mit Einblick in das
- **Physical Computing** als Interaktion von virtuellen Informatiksystemen mit der physischen Realität

Preise der Mikrocontroller (Einzelpreise ohne Versand) sind vom Stand Mitte März 2019. Bei den Fotos der Mikrocontroller handelt es sich um Pressematerial der BBC bzw. der Calliope gGmbH.

BBC micro:bit



Calliope mini



Vorteile beider Mikrocontroller

zwei für den Unterricht entwickelte, im Wesentlichen **funktionsgleiche Mikrocontroller** mit

- ... **integrierten Sensoren**, z. B. für Temperatur, Beschleunigung, Helligkeit, Kompass etc.
- ... **integrierten Aktoren**, z. B. eine 5x5-LED-Matrix, RF- / Bluetooth-Funkmodul usw.
- ... der **Anschlussmöglichkeit** weiterer elektronischer Bauteile über analoge/digitale Pins, I²C



- ✓ **erhebliche Vorteile im Anfangsunterricht**, da zunächst keine Notwendigkeit besteht, weitere Hardwarekomponenten anzuschließen
- ✓ **moderne Plattform für Elektronikversuche**, auch im Kontext von Embedded Systems, Heimautomatisierung und vernetzten Devices

Version V2 auch mit Lautsprecher, Mikrofon u.a.

spezifische Vorteile des jeweiligen Mikrocontrollers

- integrierte Hardware bietet **vielfältige, motivierende Funktionalitäten** für sehr viele spannende Unterrichtsprojekte
- **handliches Design** ermöglicht eine einfache Erweiterung mit einem Steckbrett, etwa für günstige Bauteile aus dem Bastlerbereich (z.B. LEDs für wenige Cent, Lautsprecher u.a.)
- Kopfhörer der Schüler leicht anklemmbar
- **ca. 17 €** mit Batteriekasten und USB-Kabel 2022: Version V2 ca. 19 € bis 31 €

- besitzt **noch zusätzliche Hardware**, nämlich eine RGB-LED, einen kleinen Lautsprecher sowie ein rudimentäres Mikrofon
- **außergewöhnliche Form** eignet sich ebenfalls gut für den Anschluss von Komponenten über Krokodilklemmen
- Löttaugen, um eventuell selbst eine Buchsenleiste mit 2x13 Pins anzulöten
- **ca. 35 €** im vergleichbaren Starterset 2022: ca. 38 € bis 45 €

Programmierumgebungen

- **MakeCode** (von Microsoft)
 - **microBlocks** (u. a. von Jens Mönig)
 - **Open Roberta** (u. a. von Fraunhofer)
- ... unterstützen als **blockbasierte Plattformen** **beide** Mikrocontroller gleichermaßen!



microBlocks (Alphaversion) setzt auf die aus Snap! und Scratch bekannten Block-Konzepte.

Unterrichtsmaterial, weitere Informationen zum Physical Computing sowie ein Vergleich der obigen Programmierumgebungen für den Unterricht sind verfügbar unter:

<http://www.physik.de.rs/schule/pc>



Autorenteam: Manuel RIEL, Rudolf PAUSENBERGER
eine Kooperation der FAU Erlangen-Nürnberg und des FabLab NüLand e. V.

Kontakt via E-Mail: manuel.riel@fau.de, pausenberger@cjt-gym-lauf.de



Inhalt

1. Tag

Roboter	1
Programmieren und Bauen	1
Elektrischer Strom	1
Programme schreiben und starten	2
wenn-Verzweigung	3
Variablen*	3
Projekt optische Täuschung	5
Magnetfeld*	6
solange-Schleife*	6
Das geht nicht*	6
Analoge Ausgaben	7
Üben	8
Fehlersuche	8
Digitale Eingaben	9

2. Tag

Wiederholung und weiter	10
Einen Motor betreiben	11
Dein erster Roboter	11
Ein Sensor, ein Motor; Wettbewerb	12
Fehler beheben	12

Offenes Ende

Tanzroboter; Wettbewerb	13
Mehr Ideen; Wettbewerb	13

Anhang

Stückliste	14
fischertechnik; Taster, Getriebe	15
Analoge Ausgabe, PWM*	16
Testschaltung	17
Pinbelegung	18
Alternativen: Microbit, Arduino, Entwicklungsumgebungen, Python, Projekte	19

* ist optionale Ergänzung