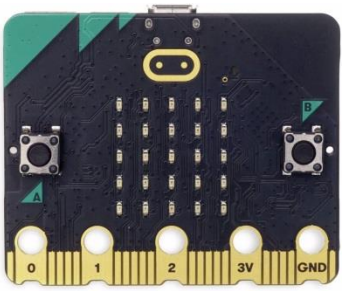
PROGRAMMIERANLEITUNG
Micro:bit TECHNIK - BASISSET



Das **Micro:bit Technik-Basisset** bietet eine einfache und kostengünstige Möglichkeit, verschiedene elektronische Schaltungen mit LEDs, Widerständen, Fotodiode, Potentiometer, Summer usw. ohne Löten aufzubauen und in Micro:bit-Programmen einzusetzen.
Der Aufbau erfolgt auf einem vorgefertigten Brett (120 x 80 mm), wobei die Bauteile und Drähte mit steckbaren Klemmfedern fixiert und dadurch problemlos ab- bzw. umgebaut werden können.
Die **Programmieranleitung** enthält zahlreiche grafische Programmierbeispiele (MakeCode©) für Anfänger und leicht Fortgeschrittene: LEDs ein- und ausschalten, LEDs dimmen, Verkehrsampel, Lauflicht, Geschicklichkeitsspiel mit Summer, Alarmanlage, lichtabhängiger Schalter usw.
Dieses Experimentierset eignet sich auch sehr gut für fächerübergreifenden Unterricht in den Fächern Informatik, Physik, Werken usw.

Grundlagen

1. Hinweise:

- Baue das **Grundbrett** laut beiliegender Anleitung zusammen, befestige einen Micro:bit mit fünf Senkkopfschrauben an den Gewindehülsen und setze zwei neue 1,5 V AAA-Batterien in den Micro:bit-Batteriehalter (3 V).
- Aufladbare Batterien (zB. NiMH, NiCD) haben eine Spannung von 1,2 Volt und sind daher nur bedingt einsetzbar. Ideal und nachhaltig wäre aber die Verwendung von **USB-Powerbanks**.
- Halte den Micro:bit von Feuchtigkeit fern und berühre ihn möglichst nicht an den Kontakten.

2. Anforderungen:

Um den Micro:bit in Betrieb zu nehmen, braucht man:

- ein Laptop oder einen PC mit Windows 10 (8, 7) oder Mac (OSX oder Linux)
- ein Micro-USB-Kabel zum Anschluss des Micro:bit an den Computer
- einen Internet-Zugang (Chrome, Edge, Firefox …) - **Aber**: Für den Betrieb ohne Internet gibt es eine **App**: https://makecode.microbit.org/offline-app

Der Micro:bit kann auch über eine App mit einem **Tablet / iPad oder Smartphone** via Bluetooth programmiert werden. Dazu muss aber der Micro:bit mit diesen Geräten **gekoppelt** werden.
Eine **Video-Anleitung** dafür findet man auf der **Homepage** von **microbit.org** unter:
https://microbit.org/get-started/user-guide/mobile/#pair-your-micro:bit-with-the-app

3. Den Micro:bit vorbereiten:

Schließe den Micro:bit über ein Micro-USB-Kabel an einen freien USB-Anschluss des PCs. Das Kabel dient sowohl zur Stromversorgung des Micro:bit als auch zur Datenübertragung. Der Micro:bit erscheint im Windows Explorer (PC) oder Filemanager (Mac) als Laufwerk mit dem Namen [MICROBIT] und einem Laufwerksbuchstaben, zB. [E:]. Der Micro:bit kann dann über dieses Laufwerk mit einer Programmdatei (*.hex) versorgt werden.

Bei neuen Micro:bits ist ein **Demo-Programm** vorinstalliert, das die Funktionen des Micro:bit zeigt und zu verschiedenen Aktivitäten aufruft, zB. Schütteln, Neigen, Taste drücken, usw. - Es wird später einfach durch eigene Programme überschrieben!

Bei Anschluss des Micro:bit an einen Computer muss die Batterieversorgung nicht getrennt werden, denn der Micro:bit schaltet automatisch auf USB-Versorgung um.

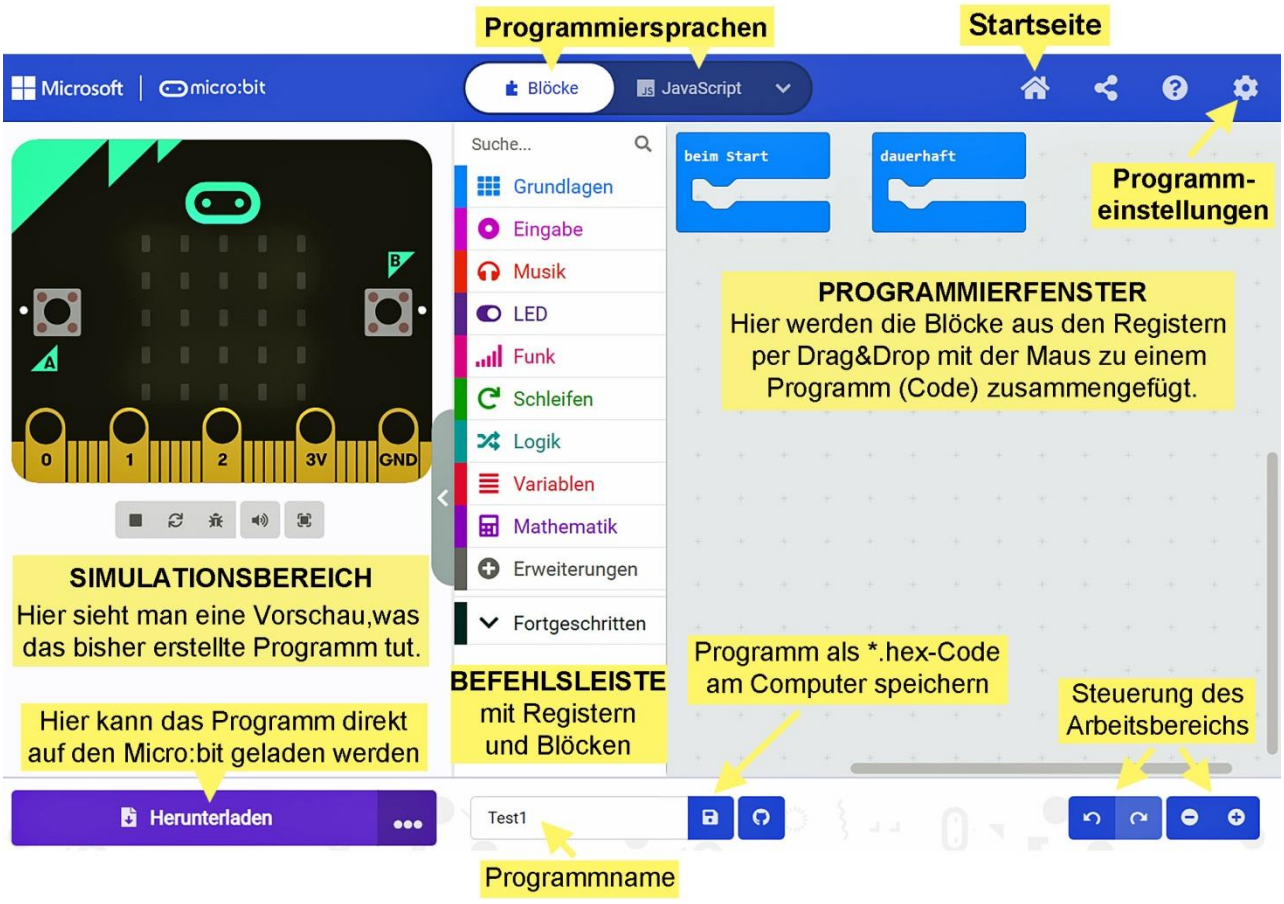
4. Der Makecode-Editor:

Zum Programmieren verwenden wir die grafische Programmierplattform **Makecode©** von Microsoft: https://makecode.microbit.org/. Eine grafische Programmierung ist ideal für Anfänger, die noch keine Programmiersprache kennen, denn sie ist intuitiv und leicht zu erlernen.
Makecode läuft im Browser, daher braucht kein eigenes Programm installiert zu werden.

Programmierumgebung

1. Programmstart:

- Schließe den Micro:bit über ein **Micro-USB-Kabel** am Computer an.
- Der Micro:bit wird im Explorer als **Laufwerk** (zB. MICROBIT [E:]) angezeigt.
- Öffne den Browser (Chrome, Edge, Firefox …) und gib folgende **Programm-URL** ein: https://makecode.microbit.org/
- Wähle die Schaltfläche [Neues Projekt] aus und gib dem Projekt einen Namen (z.B. Test1). Nun erscheint die **Programmieroberfläche**:



2. Programmbeschreibung:

Die Programmieroberfläche von **Makecode©** besteht aus drei Bereichen:
SIMULATIONSBEREICH, BEFEHLSLEISTE, PROGRAMMIERFENSTER

Im Simulationsbereich ist ein Micro:bit abgebildet, der das laufende Programm abspielt.

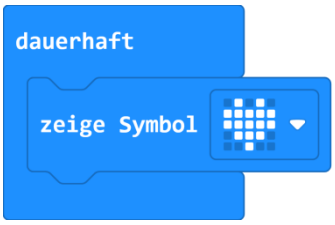
In der Befehlsleiste befinden sich verschiedenfarbige Register mit Blöcken zum Programmieren. Nach Anklicken der Register erscheinen diverse Blöcke, die man mit der Maus per Drag&Drop ins Programmierfenster ziehen kann. Die Blöcke erscheinen im Programmierfenster zuerst grau und nehmen ihre Originalfarbe erst wieder an, wenn sie richtig im Programm verankert sind. Die Blöcke lassen sich mit einem Klick der rechten Maustaste **duplizieren** und **löschen** oder man schiebt sie wieder zurück in die Befehlsleiste. Die Blöcke sind so geformt, dass sie nur dann ineinander passen, wenn sie logisch zu den Programmbefehlen passen. Dadurch werden Programmierfehler stark reduziert. **Fortgeschrittene** können aber statt der grafischen **Block-Programmierung** auch **JavaScript** oder **Python** verwenden.

Nach einem Klick auf das Zahnradsymbol (rechts oben) können Einstellungen durchgeführt werden: zB. Sprache, Programme löschen, zusätzliche Blockregister einfügen usw.

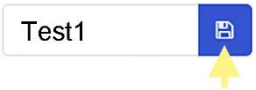
Ein Klick auf das Haus-Symbol (oben) öffnet die Startseite.

3. Ein erstes Testprogramm speichern:

Lösche den **Start-Block** [beim Start] durch Verschieben in den Registerbereich. Ziehe vom Register [Grundlagen] den **Symbol-Block »Herz«** in die **Block-Klammer »dauerhaft«**.



Klicke unten auf das **Diskettensymbol** neben dem **Programm-namen** (Test1). Das Programm wird nun lokal auf dem Computer als microbit-Test1.hex gespeichert.



4. Das Testprogramm auf den Micro:bit übertragen:

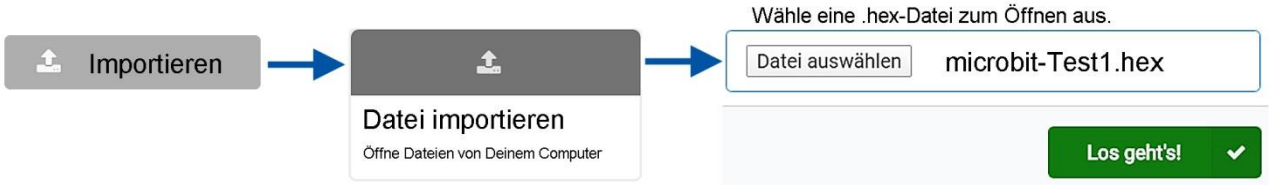
Die **Übertragung des Programms auf den Micro:bit** kann auf **zwei Arten** erfolgen:

- Öffne den Datei-Explorer und ziehe die Datei microbit-Test1.hex mit der Maus auf das Laufwerk [MICROBIT]. Bei der Übertragung blinkt zuerst ein gelbes Licht (Rückseite) und dann startet das Programm.
- Klicke erstmals im **Makecode-Editor** auf die Schaltfläche [Herunterladen], wähle das Laufwerk [MICROBIT] und klicke auf [Speichern]. Ab dem zweiten Mal kann jedes Programm durch einen einfachen Klick auf [Herunterladen] auf den Micro:bit übertragen werden.
- Mit der **Reset-Taste** auf der Rückseite des Micro:bit können Programme neu gestartet werden.

5. Ein Programm (hex-Datei) importieren:

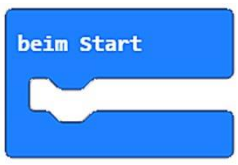
Um den **Programmcode einer hex-Datei** lesen und bearbeiten zu können, muss sie im Programmeditor **Makecode** geöffnet werden. Das kann auf **zwei Arten** erfolgen:

- Ziehe die entsprechende **hex-Datei per Drag&Drop** direkt vom Datei-Explorer auf das Programmierfenster von Makecode. Dort kann das Programm dann bearbeitet werden.
- Eine **hex-Datei** kann aber auch von der **Startseite von Makecode** importiert werden:
  Klicke auf die graue Schaltfläche [Importieren], dann auf [Datei importieren].
  Über [Datei auswählen] kann im Datei-Explorer die gewünschte hex-Datei gewählt werden.
  Nach einem Klick auf [Los geht's] wird das Programm auf dem Makecode-Editor geöffnet.



6. Die wichtigsten Blöcke für den Anfang:

- Aus dem **Register [Grundlagen]**:



Alle Blöcke (= Programme) innerhalb der **Start-Klammer** werden nur einmal beim **Start** ausgeführt.

Blöcke innerhalb dieser Klammer werden vom Micro:bit so lange als **Endlosschleife** ausgeführt, bis man den Strom abschaltet.

Dieser Block zeigt die eingefügte **Zahl** (hier „3“) auf der **LED-Matrix** an.

Dieser Block zeigt den eingefügten **Text** (hier „Hallo!“) am Micro:bit als **LED-Laufschrift** an.

Dieser Block zeigt das gewählte **Symbol** (hier „Herz“) als **LED-Symbol**.
Mit der **Pfeilauswahl** erscheint eine Auswahl von 40 Symbolen.

Diese Funktion stellt die **Micro:bit LED-Anzeige** mit **25 LEDs** dar.
Durch Anklicken der dunkelblauen Felder kann man die einzelnen LEDs an- und ausschalten und damit eigene Symbole erstellen.

Durch das Einfügen eines **Pause-Blocks** wird der Programmablauf für eine bestimmte Zeit (hier 100 ms) verzögert.
Die Angabe ist in **Millisekunden** (ms) → **1 Sekunde = 1000 ms**

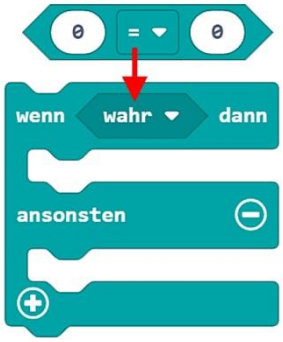
- Aus dem **Register [Eingabe]**:

Bei einem Klick auf **Knopf A** des Micro:bit wird der Programmblock innerhalb der Klammer ausgeführt.
Mit der Pfeilauswahl kann man weitere Knöpfe aktivieren: **B** und **A+B**

- Aus dem **Register [Schleifen]**:

Alle Befehle (Blöcke) innerhalb des **Wiederhol-Blocks** werden in der gewählten Anzahl (hier 4-mal) wiederholt.

- Aus dem **Register [Logik]**:



Dieser sechseckige »**Vergleichs-Block (0 = 0)** vergleicht zwei Werte, zB. ob etwas kleiner, größer, gleich groß usw. ist und gibt das Ergebnis als **„wahr“** oder **„falsch“** zB. weiter an einen **»Wenn-Block«.**

Ein **»Wenn-Block« (wenn/dann)** mit Verzweigung prüft, ob eine eingegebene **Bedingung wahr** ist (zB. 0 = 0). Ist sie **„wahr“**, wird der obere Programmteil ausgeführt, **ansonsten** der untere.
Am ⊕ kann der **»Wenn-Block«** erweitert, am ⊖ reduziert werden.

- Aus dem **Register [Fortgeschritten] + [Pins]**:

Dieser Block prüft ein **digitales Eingangssignal an P0** und gibt ein **„high“** = **1** oder **„low“** = **0** an das Programm weiter.

Dieser Block prüft ein **analoges Eingangssignal an P0** und gibt es als **Wert von „0 - 1023“** weiter an das Programm.

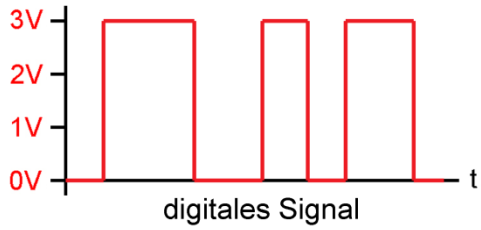
Ist der **Wert** des **digitalen Ausgangsblocks „1“**, liegt an **P0** eine Spannung von **3,3 V**. Ist der **Wert „0“**, fließt keine Spannung.

Der **analoge Ausgangsblock** gibt die Spannung in **Stufen von 0 - 1023** ab. → **1023 = 3,3 V**
Ist zB. der Wert **512**, dann fließen ca. **1,65 V**.

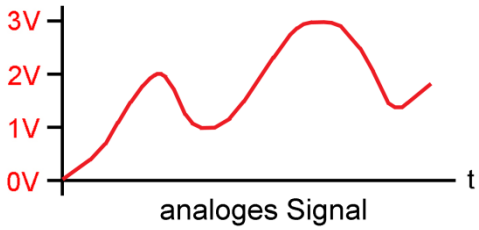
Mit der **Pfeilauswahl** können die Pins (P0, P1, P2 …) geändert werden.

7. Ein- und Ausgänge des Micro:bit:

Der Micro:bit besitzt 20 programmierbare **Ein- bzw. Ausgänge (= Pins oder Ports)**, wobei aber ohne zusätzlichem „Breakout-Board“ nur die **Pins P0, P1 und P2** zur Verfügung stehen. Diese drei Pins können, je nach Programm, als **digitale oder analoge Eingänge** bzw. **Ausgänge** genutzt werden.
An den Pins des **Micro:bit (V2)** können **Verbraucher** (zB. LEDs) mit einer **maximalen** Stromstärke von **200 mA** (V1 - 100 mA) angeschlossen werden.



digitales Signal

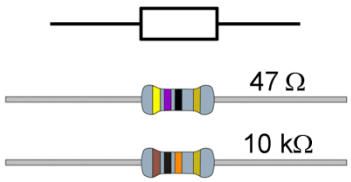


analoges Signal

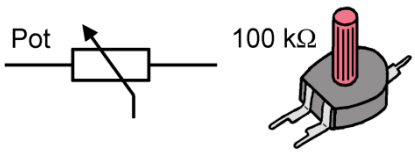
- **Analog oder digital**: Die Pins am Micro:bit arbeiten mit einer Spannung von **0 - 3,3 Volt**. Für **digital** genutzte **Pins** gibt es nur **zwei Zustände**: **Null** (**0**) oder **Eins** (**1**) → dabei entspricht der Zustand **„0“** einer Spannung von 0 Volt und der Zustand **„1“** einer Spannung von 3,3 Volt. **Analog** genutzte **Pins** arbeiten mit **Spannungssignalen** zwischen **0** und **3,3 Volt.**
- **Pin als digitaler Eingang**: Eintreffende Spannungen (zB. von einem Sensor) ab ca. 2,5 Volt erkennt der Micro:bit als **Zustand „1“ (high)**; Spannungen darunter als **Zustand „0“ (low)**.
- **Pin als analoger Eingang**: Die eintreffende Spannung (zB. von einem Sensor) wird vom Micro:bit in **Stufen** (= **Wert**) von **0 - 1023** umgewandelt. Eine anliegende Spannung von **1,1 V** hat demnach einen **Wert von 341** (= ca. ein Drittel von 1023).
- **Pin als digitaler Ausgang**: Es gibt nur **zwei Zustände**: **„0“ = 0 Volt** und **„1“ = 3,3 Volt**.
- **Pin als analoger Ausgang**: Die Spannung wird in **Stufen** (= **Wert**) von **0 - 1023** ausgegeben. Der **Wert 1023** entspricht **3,3 V**. Der **Wert 512** entspricht also einer Spannung von ca. **1,65 V**.

8. Elektronische Bauteile:

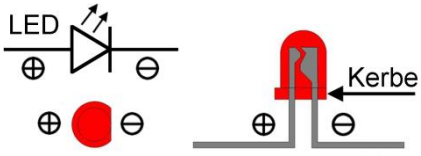
- **Widerstand (R)**:
  Mit Widerständen kann man Spannung und Stromstärke in einem Stromkreis reduzieren, zB. für LEDs.
  Die **Widerstandswerte** erkennt man an den **Farbringen**:
  gelb-lila-schwarz-gold = **47 Ohm** (Ω)
  braun-schwarz-orange-gold = **10 kOhm** (kΩ)



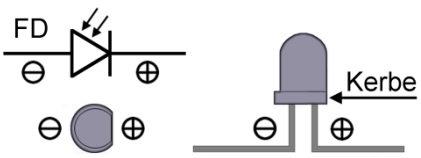
- **Potentiometer (Pot)**:
  Potentiometer sind **regelbare** Widerstände.
  Das beiliegende **Pot** hat einen Wert von **0 - 100 kOhm**.



- **Leuchtdiode (LED)**:
  Das Set enthält drei LEDs: **Rot, Gelb** und **Grün**
  LEDs müssen richtig gepolt werden. Der Anschluss unterhalb der **flachen Einkerbung** am Gehäuse muss an **Minus (-)**.
  Da LEDs nur eine Spannung von ca. 2 V und eine Stromstärke von 20 mA vertragen, benötigen sie bei 3 V einen **Vorwiderstand** von 47 Ohm.



- **Fotodiode (FD)**:
  Die Fotodiode sieht aus wie eine dunkle LED.
  Der Anschluss unterhalb der flachen **Einkerbung** am Gehäuse muss aber (im Gegensatz zur LED) an **Plus (+)**.



- **Summer (Su)**:
  Summer erzeugen einen Ton mit gleichbleibender Tonhöhe.
  Der **schwarze Anschlussdraht** muss an **Minus (- GND)**.

Einführung in die Programmierung des Micro:bit

- Die ersten Programme dienen dem **Kennenlernen** von **Grundfunktionen** beim Programmieren des Micro:bit und werden noch **ohne Schaltungsaufbau** durchgeführt.
- Setze zwei neue **Mignon-Batterien** (je 1,5 V) in die Batteriebox (3 V) und schließe sie an den Micro:bit oder verwende eine **USB-Powerbank** (5 V).
- Die vorgeschlagenen **Namen** der **„hex-Dateien“** können natürlich geändert werden.

Programm 1: Begrüßung

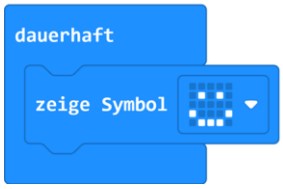
Öffne den **Makecode-Editor** (https://makecode.microbit.org/), klicke auf [Neues Projekt] und gib ihm den Namen „Hallo1“.

Vorgabe: Nach dem Einschalten soll der Micro:bit einmal den Lauftext **„Hallo!“** anzeigen und dann **dauerhaft** einen **»freundlichen Smiley«.**

Programm-Code: (microbit-Hallo1.hex)

Im linken **Simulationsbereich** des **Makecode-Editors** sieht man bereits eine Vorschau, was das Programm tut.
Speichere, wie auf Seite 3 beschrieben, das fertige Programm auf dem Computer.
Verbinde den Micro:bit über ein Micro-USB-Kabel mit dem Computer und übertrage das Programm (microbit-Hallo1.hex) auf den Micro:bit.

Der Lauftext **„Hallo“** wird nur einmal angezeigt.



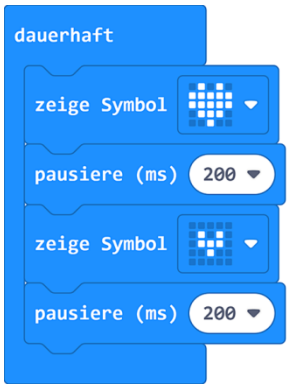
Solange der Micro:bit Strom hat, leuchtet der Smiley.

Weitere Aufgabe: Ändere den Text auf: **„Ich bin ein Microbit“** und das Symbol auf ein **„Herz“**.

Programm 2: Herzklopfen

Vorgabe: Ein großes und ein kleines **»Herz-Symbol«** sollen jeweils **200 ms** lang abwechselnd aufleuchten.

Programm-Code: (microbit-Herzklopfen1.hex)



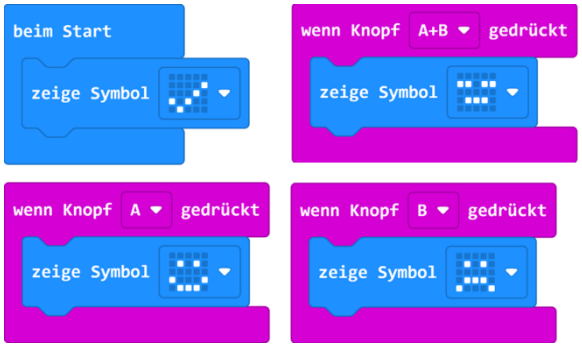
Der **Pause-Block** lässt das „Herz“ 200 ms lang aufleuchten.

Weitere Aufgabe: Ändere die **Herzfrequenz** durch längere Pause-Zeiten (zB. 500 ms).

Programm 3: Knopf A und B

Vorgabe: Nach Drücken der **Knöpfe A, B und A+B** sollen verschiedene **»Smileys«** aufleuchten.

Programm-Code: (microbit-KnopfAB-1.hex)



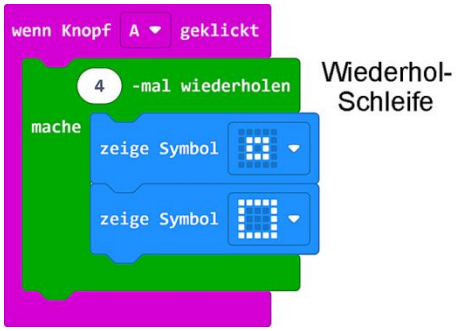
Weitere Aufgabe: Nach Drücken der **Knöpfe A, B** und **A+B** sollen die **Buchstaben „A“**, **„B“** und **„C“** angezeigt werden.

Programm 4: Wiederholung

Durch den Einsatz einer **»Wiederhol-Schleife«** aus dem **Register [Schleifen]** kann man die Anzahl der Wiederholungen von eingefügten Programmteilen genau festlegen.

Vorgabe: Ein kleines und ein großes **»Quadrat-Symbol«** sollen nach Betätigung von **Knopf A 4-mal blinken**.

Programm-Code: (microbit-Wiederholung1.hex)

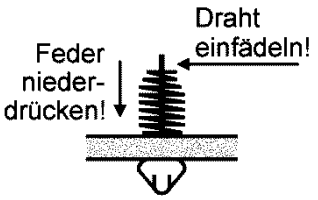


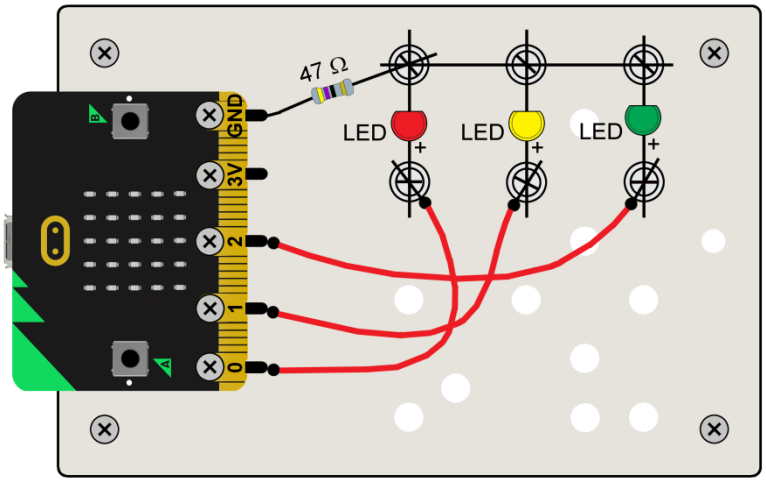
Weitere Aufgaben:

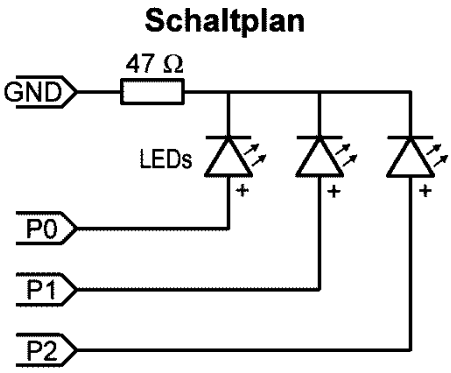
- Ändere die **Anzahl der Wiederholungen**
- Setze an Stelle der Symbole **2 Zahlen** (zB. 0 / 1)
- Ändere die Blinkfrequenz durch **»Pause-Blöcke«**

Programme für Schaltungsaufbau 1 - 3 LEDs

- Drücke 6 **Klemmfedern** laut Abbildung in die Bohrungen der Grundplatte. Nach Niederdrücken der Federn können Drähte und elektronische Bauteile eingefädelt werden.
  Verbinde die oberen 3 Klemmfedern mit einem ca. 50 mm langen, blanken Drahtstück.
  Kürze die Anschlussdrähte des Widerstands (47 Ω) nach Bedarf und stelle die restlichen Verbindungen mit drei roten **Verbindungsleitungen** her.
  An den Lötösen werden die Drähte mit **Schlauchhülsen** fixiert.
  Achte auf die richtige Verdrahtung und auf den **polungsrichtigen Einbau** der **LEDs**!







Programm 5: LED mit Knopf A schalten

Wenn der **digitale Ausgang an P0 auf „1“** gesetzt wird, liegt eine Spannung von ca. **3 V** an. Den Block **»schreibe digitalen Wert von „P0“ an „0“«** findet man in den **Registern [Fortgeschritten] + [Pins]**. Da unsere LEDs nur ca. 2 V und 20 mA vertragen, muss ein **Vorwiderstand** (47 Ω) eingebaut werden.

Vorgabe: Durch Drücken von **Knopf A** soll der Ausgang **„P0“** auf **„1“** gestellt werden und die **rote LED** an **„P0“** leuchten. Nach **2 Sekunden** (2000 ms) soll die LED automatisch erlöschen.

Programm-Code: (microbit-LED-A1.hex)

Weitere Aufgaben: Ändere die **Leuchtdauer** auf **6 Sekunden** und Ausgang **„P0“** auf **„P2“**.

Programm 6: LED mit Knopf A und B schalten

Vorgabe: Die **gelbe LED** an **„P1“** soll mit **Knopf A** ein- und **Knopf B** ausgeschaltet werden. Bei Betätigung von **Knopf A** soll zusätzlich die **Zahl „1“** und bei **B** die **Zahl „0“** aufleuchten.

Programm-Code: (microbit-LED-AB1.hex)

Weitere Aufgaben: alle **3 LEDs** sollen mit **Knopf A** ein- und **Knopf B** ausgeschaltet werden.

Programm 7: Fußgängerampel

Vorgabe: Die **rote LED** an **„P0“** und die **grüne LED** an **„P2“** sollen mit **Knopf A** und **B** abwechseln ein- und ausgeschaltet werden. Gleichzeitig soll mit **Knopf A** ein **»X-Symbol«** und mit **Knopf B** ein **»Figur-Symbol«** am Micro:bit aufleuchten.

Programm-Code: (microbit-Fußgaenger1.hex)

Weitere Aufgaben: Erweitere das Programm so, dass nach Drücken der **Knöpfe A + B** die **gelbe LED** leuchtet und die **Pins „P0“** und **„P2“** auf **„0“** gestellt werden.

Programm 8: Blinkende LED

Vorgabe: In einem **Block »dauerhaft«** soll die **rote LED** an **„P0“** alle **0,5 Sekunden** ein- und ausgeschaltet werden.

Programm-Code: (microbit-Blinken1.hex)

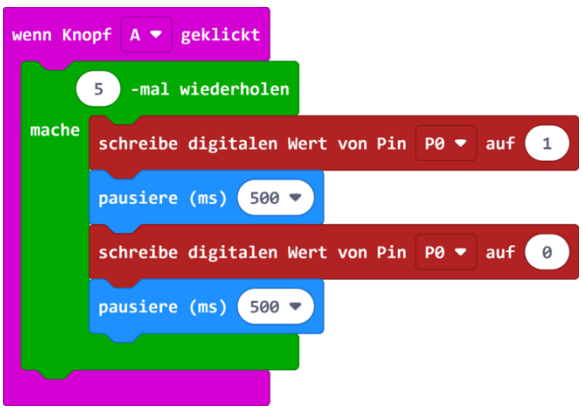
Weitere Aufgabe: Ändere die **Blinkfrequenz** durch andere **Pause-Zeiten**.

Programm 9: Blinken

Vorgabe: Mit Hilfe einer **»Wiederhol-Schleife«** soll die **rote LED** an **„P0“** nach einem Klick auf **Knopf A** im Abstand von **500 ms** **5-mal** blinken.

Programm-Code: (microbit-Blinken2.hex)

Weitere Aufgabe: Die **gelbe** und die **grüne LED** sollen im Abstand von **200 ms 10-mal** blinken.

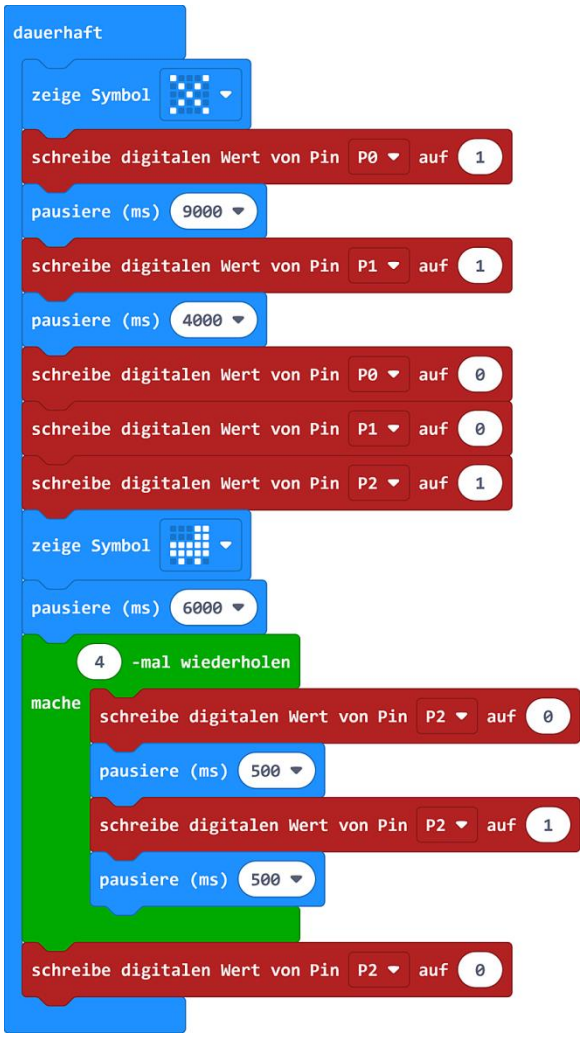


Programm 10: Verkehrsampel

Eine Verkehrsampel beginnt mit einer längeren **Rot-Phase**, gefolgt von einer **Rot-/Gelb-Phase** und endet in einer **Grün-Phase**, die zum Schluss **4-mal blinkt**.

Vorgabe: In einem **Block »dauerhaft«** soll eine **Verkehrsampel** programmiert werden. Die **Rot-Phase** soll **9 Sekunden**, die **Rot-/Gelb-Phase 4 Sekunden**, die **Grün-Phase 6 Sekunden** und die **Grünblink-Phase 4 Sekunden** dauern.
Bei **„Rot“** soll ein **»X-Symbol«** und bei **„Grün“** ein **»Fahrzeug-Symbol«** aufleuchten.

Programm-Code: (microbit-Ampel1.hex)



Weitere Aufgaben:

- Erweitere das Programm durch eine **Gelb-Phase** (3 Sekunden) nach der **Grünblink-Phase**.
- Ändere die **Laufzeiten** der verschiedenen Phasen an den **»Pause-Blöcken«**.

Programm 11: Lauflicht

Vorgabe: In einer **»dauerhaft-Schleife«** sollen die **drei LEDs** immer der Reihe nach jeweils **200 ms** lang aufleuchten.

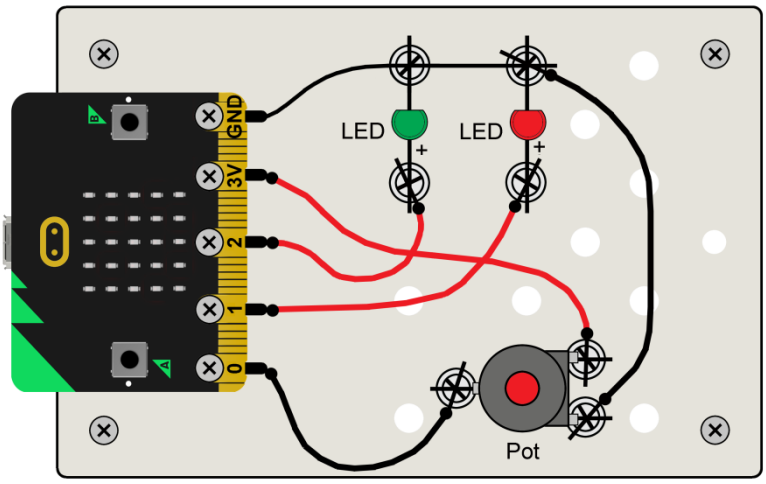
Programm-Code: (microbit-Lauflicht1.hex)

Weitere Aufgaben:

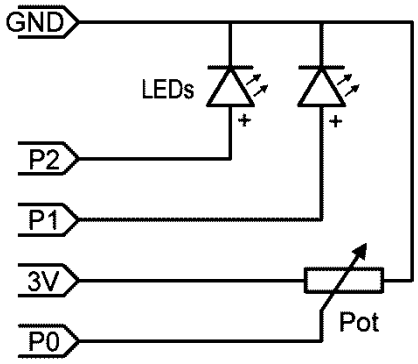
- Ändere die **Blinkdauer** der **LEDs**.
- Erweitere das Programm so, dass nach der **grünen LED** (P2) wieder die **gelbe LED** (P1) folgt.

Programme für Schaltungsaufbau 2 - 2 LEDs + Pot

- Drücke 7 **Klemmfedern** laut Abbildung in die Bohrungen der Grundplatte. Nach Niederdrücken der Federn können Drähte und elektronische Bauteile eingefädelt werden.
  Verbinde den **GND**-Anschluss und die oberen 2 Klemmfedern mit einem ca. 60 mm langen, blanken Drahtstück. Klemme die **LEDs** und das **Pot** in die entsprechenden Federn und stelle die restlichen Verbindungen mit zwei schwarzen und drei roten **Verbindungsleitungen** her.
  Achte auf die richtige Verdrahtung und auf den **polungsrichtigen Einbau** der **LEDs**!
  Für die folgenden Blink- und Dimm-Experimente kann der LED-Vorwiderstand wegfallen.



Schaltplan



Programm 12: Analoger Wert an P0

Mit dem **Block »analoge Werte von Pin „P0“«** aus den **Registern [Fortgeschritten] + [Pins]** wird ein **analoges Eingangssignal** in einen **Wert von 0 - 1023** umgewandelt und mit **»zeige Zahl«** vom **Register [Grundlagen]** am Micro:bit angezeigt.

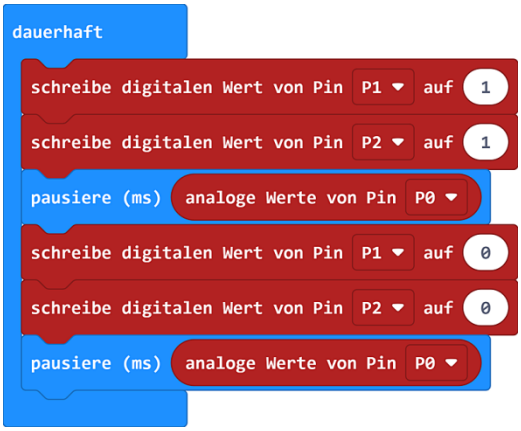
Vorgabe: Der **analoge Wert** an **„P0“** soll **dauerhaft** angezeigt werden. Drehe dazu die Achse des **Potentiometers** (Pot) und beobachte das Steigen und Sinken des **Wertes** von **0 - 1023**.

Programm-Code: (microbit-Wert-analog1.hex)

Programm 13: Blinkgeschwindigkeit mit Pot regulieren

Zieht man den **analogen Wert** (0 - 1023) des **Pots** an **„P0“** in das **Zeitfenster** des **»Pause-Blocks«**, kann man die **Blinkgeschwindigkeit** der LEDs mit dem Pot stufenlos einstellen.

Vorgabe: Die **zwei LEDs** sollen **dauerhaft** blinken. Dabei soll der **analoge Wert des Pots** an **„P0“** als **„Zeitwert in ms“** für die zwei **»Pause-Blöcke«** dienen.



Programm-Code: (microbit-Blinker+Pot1.hex)

Weitere Aufgabe: Ändere das Programm so, dass die zwei LEDs **abwechselnd** blinken.

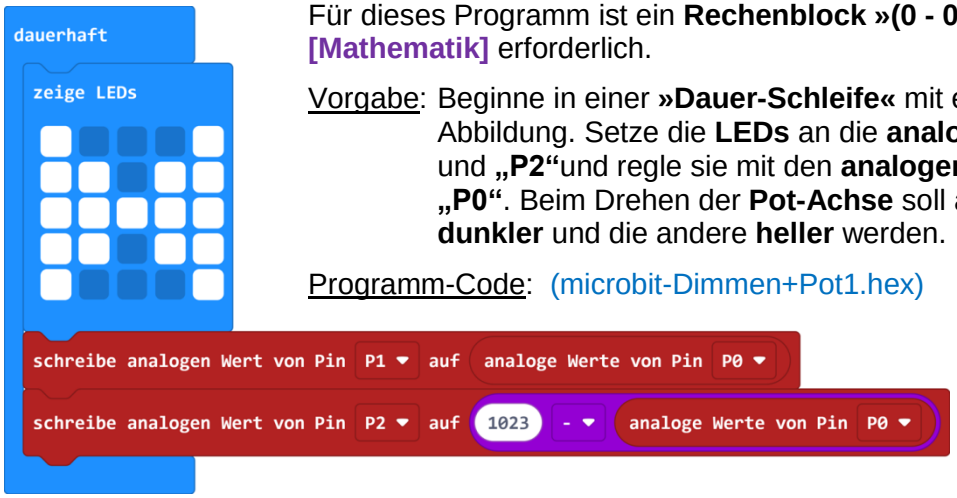
Programm 14: Säulendiagramm mit Pot

Der erforderliche **Block »zeichne Säulendiagramm von …«** befindet sich im **Register [LED]**.

Vorgabe: Zeichne mit den **analogen Werten** von **„P0“** (0 - 1023) ein **Säulendiagramm** auf der **LED-Matrix** des Micro:bit und verändere es mit dem **Pot** an **„P0“**.

Programm-Code: (microbit-Diagramm1.hex)

Programm 15: LEDs mit Pot dimmen

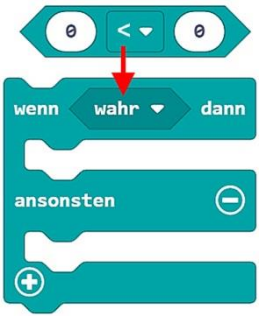


Für dieses Programm ist ein **Rechenblock »(0 - 0)«** aus dem **Register [Mathematik]** erforderlich.

Vorgabe: Beginne in einer **»Dauer-Schleife«** mit einem **»Symbol«** laut Abbildung. Setze die **LEDs** an die **analogen Ausgänge „P1“** und **„P2“**und regle sie mit den **analogen Werten** des **Pots** an **„P0“**. Beim Drehen der **Pot-Achse** soll aber eine **LED dunkler** und die andere **heller** werden.

Programm-Code: (microbit-Dimmen+Pot1.hex)

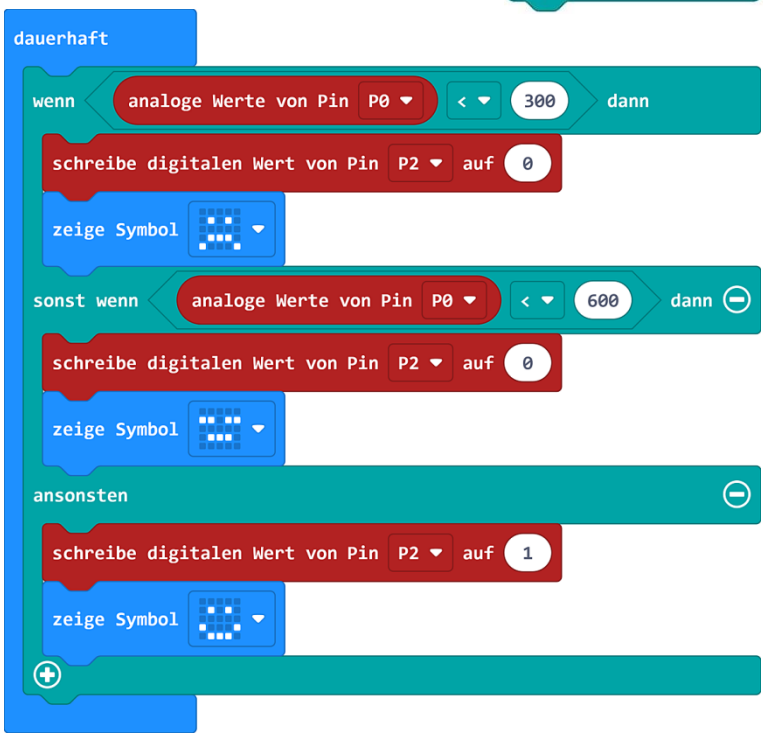
Programm 16: Wenn + 3 Smileys + Pot



Für das Programm benötigen wir einen **»Wenn-Block« (wenn/dann)** mit Verzweigung und einen sechseckigen **»Vergleichs-Block (0 < 0)** aus dem **Register [Logik]**.
Wenn die **Bedingung wahr** ist (**P0 < 300 oder P0 < 600**), dann werden die oben eingefügten Programmteile ausgeführt, **ansonsten** der untere.
Am ⊕ kann der **»Wenn-Block«** erweitert, am ⊖ reduziert werden.

Vorgabe: Bei einem **analogen Wert** am **Pot** unter **„300“** soll ein **»grimmiger Smiley«** und unter **„600**“ ein **»gelangweilter Smiley«** leuchten. **Ansonsten** soll ein **»Froher Smiley«** und die grüne **LED** an **„P2“** leuchten.

Programm-Code:
(microbit-Wenn+Pot1.hex)

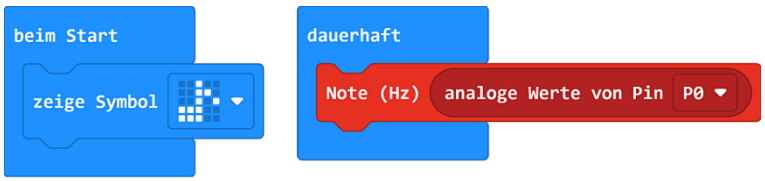


Weitere Aufgabe: Wenn der **Wert** am Pot unter **„300“** liegt, soll zusätzlich die rote **LED an P1** leuchten.

Programm 17: Tonhöhe mit Pot regulieren (Nur V2)

Für dieses Programm benötigt man den **Block »Note (Hz) …«** vom **Register [Musik]**.

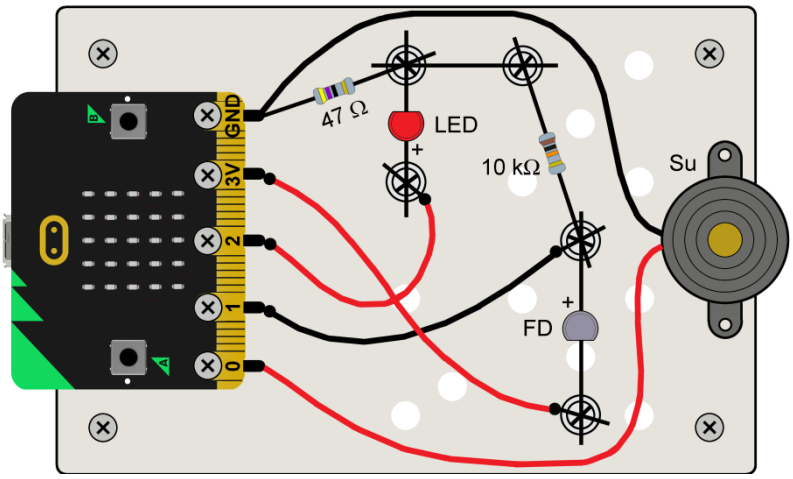
Vorgabe: Beim Start soll ein **»Noten-Symbol«** aufleuchten.
Die **Tonhöhe (Hz)** der Note im **Block »dauerhaft«** soll durch die **Werte** des **Pots** an **„P0“** bestimmt werden.

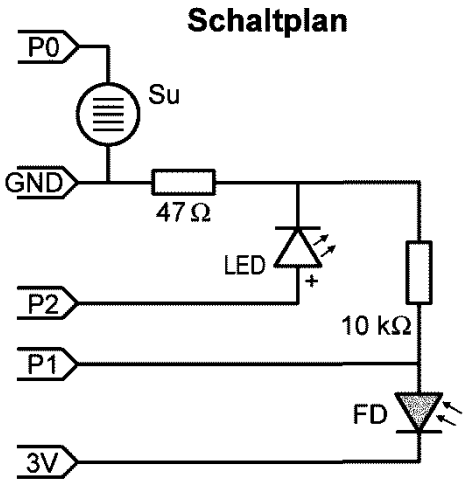


Programm-Code: (microbit-Ton+Pot1.hex)

Programme für Schaltungsaufbau 3 - LED + FD + Summer

- Drücke fünf **Klemmfedern** laut Abbildung in die Bohrungen der Grundplatte. Nach Niederdrücken der Federn können Drähte und elektronische Bauteile eingefädelt werden.
  Verbinde die oberen zwei Klemmfedern mit einem ca. 30 mm langen, blanken Drahtstück.
  Kürze die Anschlussdrähte der zwei **Widerstände** (47 Ω, 10 kΩ) nach Bedarf und klemme **Summer**, **Widerstände**, **LED** und **FD** in die entsprechenden Federn und Pins.
  Stelle die restlichen Verbindungen mit drei isolierten **Verbindungsleitungen** her.
  Achte auf richtige Verdrahtung und den **polungsrichtigen Einbau** von **LED**, **FD** und **Summer**!





Programm 18: Analoger Wert an P1

Die **analogen Werte** (0 - 1023) der **Fotodiode (FD)** an **„P1“** hängen sehr stark davon ab, ob die Messung im Freien, bei Sonnenlicht oder in einem Innenraum durchgeführt wird.

Vorgabe: Der **analoge Wert** der **Fotodiode (FD)** an **„P1“** soll **dauerhaft** angezeigt werden. Halte dazu die **Fotodiode** ins Licht, bzw. dunkle sie mit der Hand ab und beobachte das **Steigen** und **Sinken** des **Wertes**.

Programm-Code: (microbit-Wert-analog2.hex)

Programm 19: Säulendiagramm mit FD

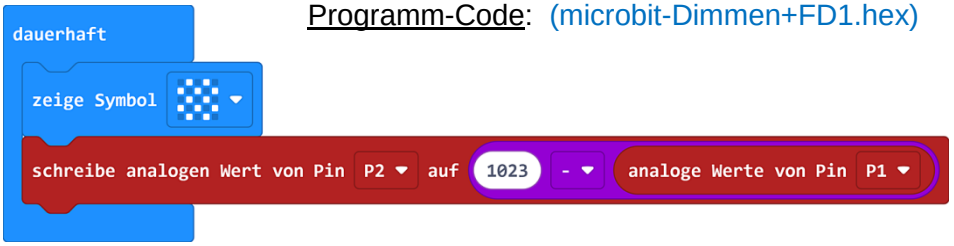
Vorgabe: Zeichne mit den **analogen Werten** der **FD** an **„P1“** (0 - 1023) ein **Säulendiagramm** auf der **LED-Matrix** und verändere es durch Abdunkeln der **Fotodiode (FD)**.

Programm-Code: (microbit-Diagramm2.hex)

Programm 20: LED mit FD Dimmen

Vorgabe: Beginne im **Block »dauerhaft«** mit einem **»Symbol«** Setze die **LED** an den **analogen Ausgang „P2“** und regle die Helligkeit der LED mit den **analogen Werten** der **FD**. Beim **Abdunkeln** der **FD** soll die **LED heller** werden.

Programm-Code: (microbit-Dimmen+FD1.hex)



Weitere Aufgabe:

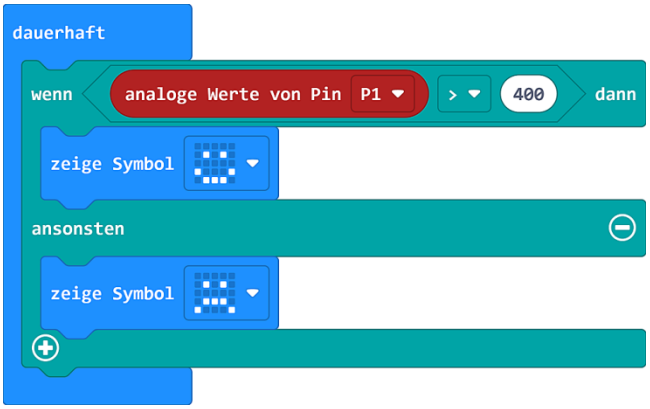
Beim Abdunkeln der **FD** soll die **LED dunkler** werden.

Programm 21: Wenn + Smileys + FD

Für das Programm benötigen wir einen **»Wenn/dann-Block«** mit Verzweigung und einen sechseckigen **»Vergleichs-Block (0 > 0)** aus dem **Register [Logik]**.

Vorgabe: Bei einem **analogen Wert** der **FD** an **„P1“** über **„400“** soll in einem **»Wenn-Block«** ein **»froher Smiley«** aufleuchten; ansonsten ein **»grimmiger Smiley«**.

Programm-Code: (microbit-Wenn+FD1.hex)



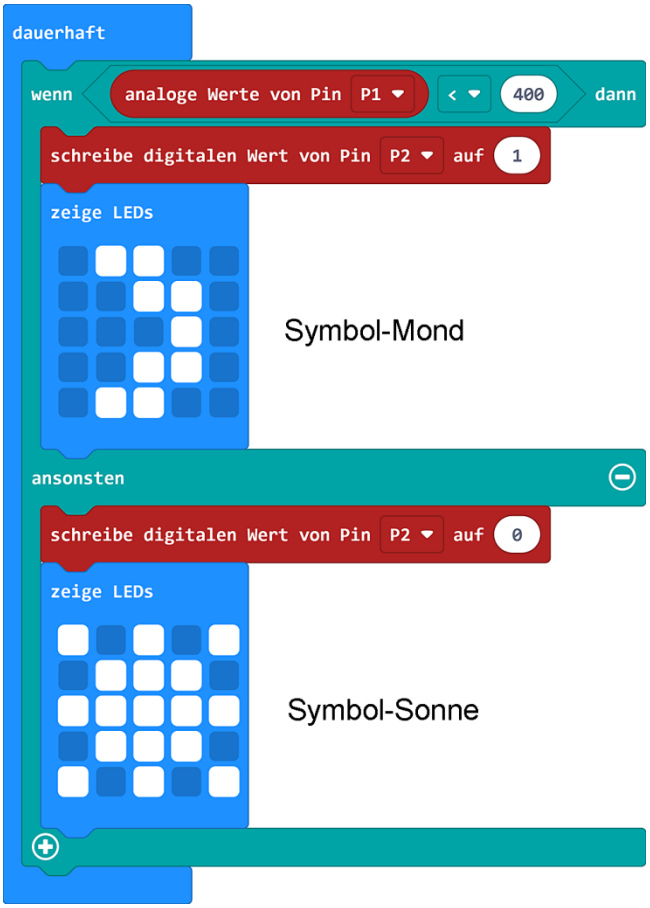
Weitere Aufgabe: Passe den **Grenzwert** der momentanen Lichtsituation an.

Programm 22: Straßenbeleuchtung

Wenn die Helligkeit einen bestimmten Wert unterschreitet, soll automatisch das Licht angehen. Da die LED nur ein- oder ausgeschaltet wird, benötigt man einen **digitalen Ausgangsblock »schreibe digitalen Wert von P2 auf …«** aus dem **Register [Pins]**.

Vorgabe: Bei einem **analogen Wert** der **FD** an **„P1“** unter **„400“** soll in einem **»Wenn-Block«** die **LED an „P2“** eingeschaltet und ein **»Mond«** gezeigt werden. Ansonsten soll die LED ausgeschaltet sein und eine **»Sonne«** aufleuchten.

Programm-Code: (microbit-Beleuchtung1.hex)



Weitere Aufgabe: Der **Summer an „P0“** soll unterhalb eines **Grenzwertes** von **„300“** ertönen.

Programm 23: Schubladen-Alarm

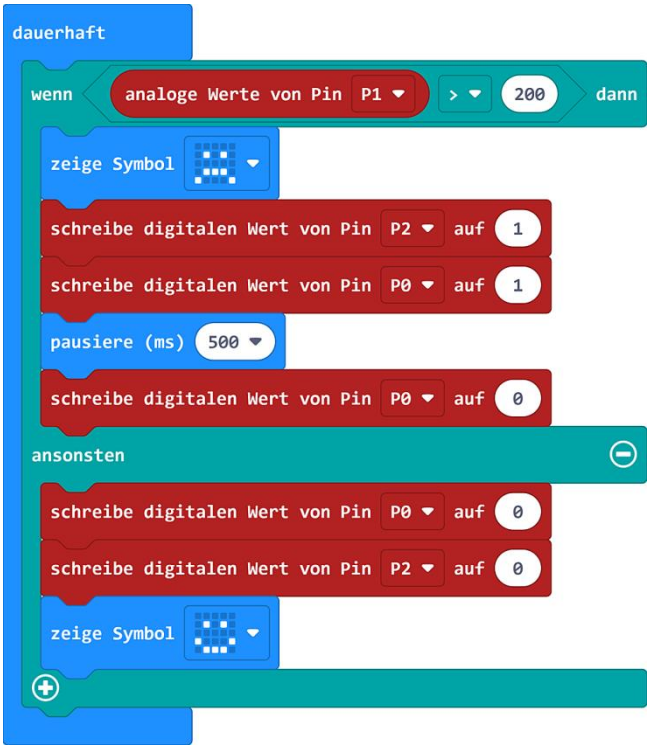
Wenn durch unbefugtes Öffnen zB. Licht in eine Schublade oder in ein Kästchen dringt, soll ein Alarmsignal ertönen.

Vorgabe: Wenn der **analoge Wert** der **FD** an **„P1“** größer ist als **„200“**, soll ein **»grimmiger Smiley«** erscheinen, die **LED** an **„P2“** leuchten und ein unterbrochener **Summerton** an **„P0“** ertönen.
Ansonsten soll nur ein **»fröhlicher Smiley«** leuchten.

Programm-Code: (microbit-Alarm1.hex)

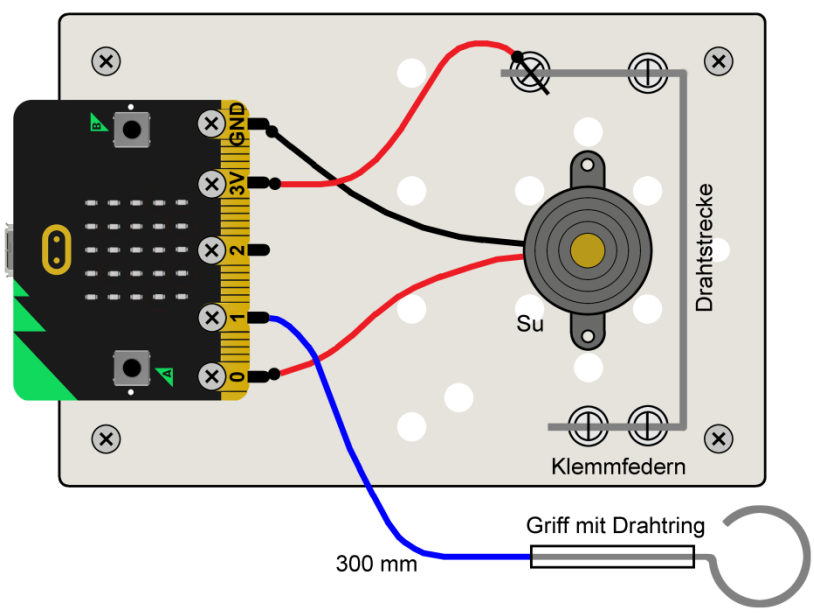
Weitere Aufgabe:

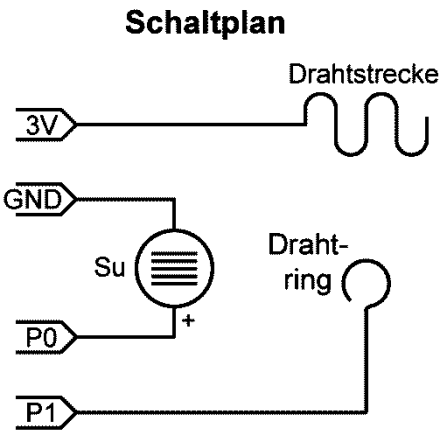
Ändere das Programm so, dass die **LED** im Rhythmus des Summertons blinkt.



Programme für Schaltungsaufbau 4 - „Heißer Draht“

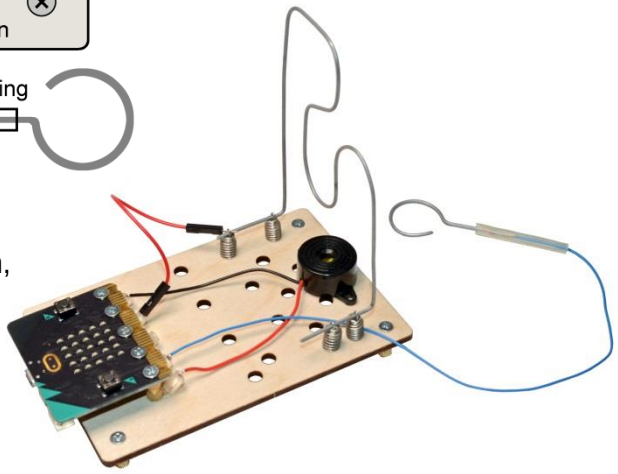
- Drücke vier Klemmfedern in das Grundbrett und klemme die vorbereitete **Drahtstrecke** darin fest. Baue die Schaltung laut Bild und Schaltplan auf. Achte auf die richtige Verdrahtung und auf den polungsrichtigen Einbau des Summers (rotes Kabel an P0). An den **Lötösen** werden die Drähte mit **Schlauchhülsen** fixiert.
  Folgende **Werkzeuge** sind beim Schaltungsaufbau und bei der Herstellung der Drahtstrecke erforderlich: Seitenschneider, Spitzzange, Abisolierzange





- **Spielanleitung „Heißer Draht“**:

  Bei diesem Geschicklichkeitsspiel geht es darum, einen Drahtring so über eine Drahtstrecke zu führen, dass der Ring den Draht nicht berührt. Sieger ist, wer bei einem Durchlauf (vor und zurück) die wenigsten „Berührungen“ hat.



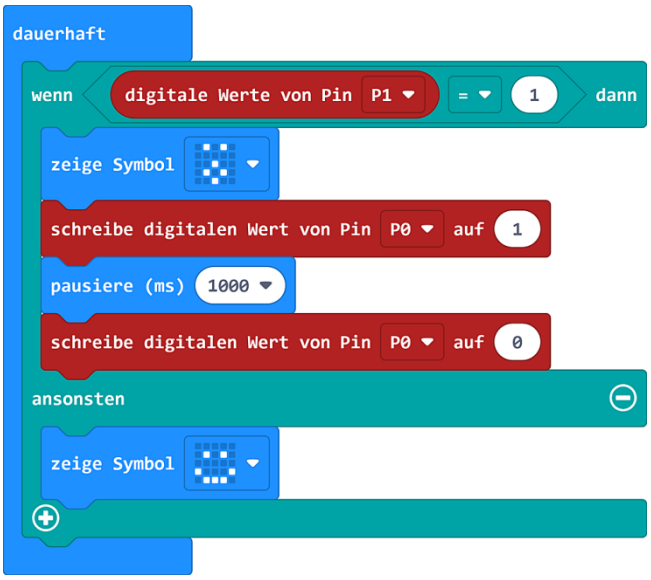
Programm 24: „Heißer Draht“ mit Summerton

Wenn der **Drahtring** die **Drahtstrecke** (= Rennstrecke) berührt, liegen +**3,3 V** an **„P1“**. Der Wert ist daher **„HIGH“ = „1“** und der Summer ertönt.

Vorgabe: Wenn der **digitale Eingang** an **„P1“** den **Wert „1“** hat, soll ein **»erstaunter Smiley«** erscheinen und der **Summer** an **„P0“** **1 Sekunde** lang ertönen. Ansonsten soll nur ein **»fröhlicher Smiley«** leuchten.

Programm-Code: (microbit-Draht+Ton1.hex)

Weitere Aufgabe: Ändere das Programm so, dass der **Summer 3-mal 500 ms** lang ertönt. (**Wiederhol-Schleife** verwenden!)



Programm 25: Klicks-Zähler

Für weitere Programme benötigen wir **Variablen. Variablen** sind **„Behälter“**, die Zahlen und Werte für ein laufendes Programm zwischenspeichern können.
Öffne das **Register [Variablen]**, klicke auf **»Erstelle eine Variable«**, gib der **Variable** den Namen **„Klicks“** und bestätige mit **»OK«**. Es erscheinen dann drei neue, rote Blöcke.

Vorgabe: Setze die **Variable »Klicks«** beim Start auf **„0“**. Bei einem Druck auf **Knopf A** soll die **Klicks-Anzahl** um **„1“ erhöht**, und angezeigt werden. Bei Betätigung von **Knopf B** soll die **Klicks-Anzahl** um **„1“ vermindert** und angezeigt werden.

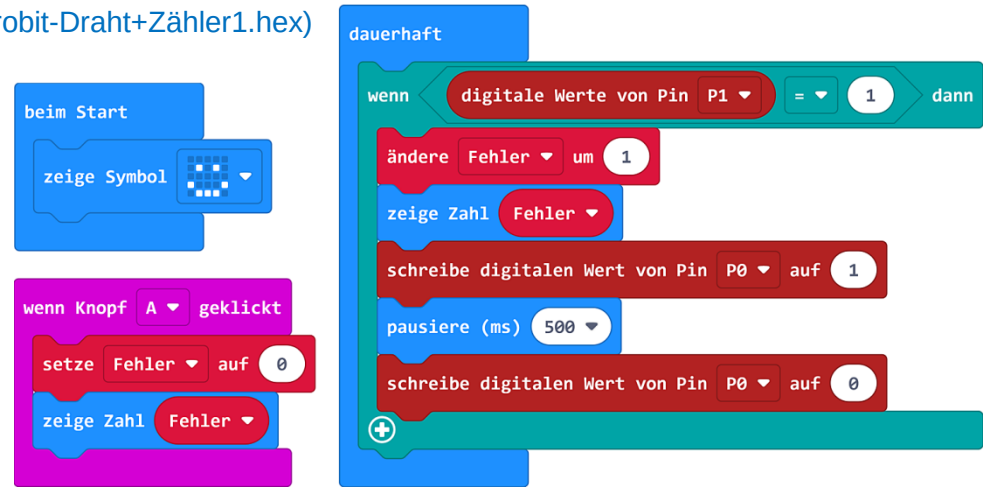
Programm-Code: (microbit-Klicks1.hex)

Programm 26: „Heißer Draht“ mit Fehlerzähler

Beim Berühren der „Rennstrecke“ soll der Summer ertönen und die Fehlerzahl gezeigt werden.

Vorgabe: Erstelle eine **Variable** mit dem Namen **»Fehler«** und zeige beim **Start** einen **»frohen Smiley«**. Wenn der **digitale Eingang** an **„P1“** den **Wert „1“** hat, soll die **Zahl** der **Variablen »Fehler«** gezeigt werden und der **Summer** an **„P0“ 500 ms** lang ertönen. Mit einem Klick auf **Knopf A** soll man die **Fehlerzahl** wieder auf **„0“** stellen können.

Programm-Code: (microbit-Draht+Zähler1.hex)



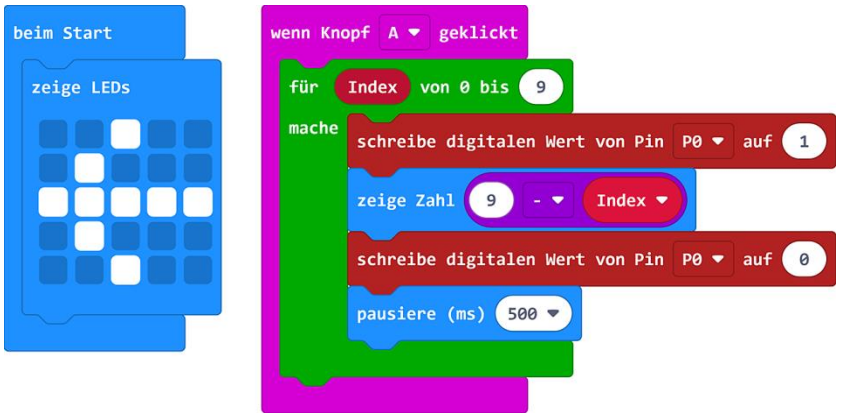
Programm 27: Countdown +Ton

»Index-Schleifen« funktionieren wie **»Wiederhol-Schleifen«** mit dem Vorteil, dass sie die **aktuelle Wiederholzahl** anzeigen und in der abrufbaren **Variablen »Index«** speichern können.

Vorgabe: Beim **Start** soll ein **»Pfeil-Symbol«** in Richtung **Knopf A** zeigen. Nach Betätigung von **Knopf A** sollen mittels **»Index-Schleife«** die **Index-Zahlen von 9 - 0** angezeigt werden. Nach jeder Zahl soll der **Summer** an **„P0“ 500 ms** lang ertönen.

Programm-Code:

(microbit-Index+Ton1.hex)



Weitere Aufgabe:

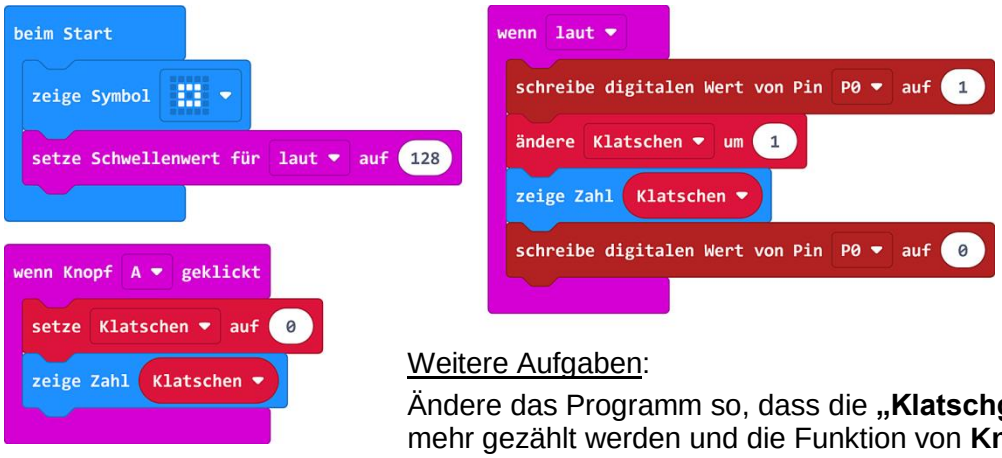
Der **Countdown** soll bei der **Zahl „5“** beginnen.

Programm 28: Klatsch-Zähler (nur V2)

Der **Micro:bit V2** ist mit einem **Mikrofon** ausgestattet, das zur Steuerung von Programmen genutzt werden kann. Die **Ansprechempfindlichkeit** (Schwellenwert) reicht von **0 - 255**.

Vorgabe: Erstelle die **Variable »Klatschen«**. Setze beim **Start** ein **»Quadrat-Symbol«** und stelle den **Schwellenwert** auf **128**. Der **Block »wenn laut«** soll auf ein **„Klatschgeräusch“** reagieren: Der **Summer** soll kurz ertönen und ein **Zähler** soll die **„Klatschgeräusche“** zählen. **Knopf A** soll den **Zähler** wieder auf **„0“** setzen.

Programm-Code: (microbit-Klatschen+Zahl1.hex)



Weitere Aufgaben:

Ändere das Programm so, dass die **„Klatschgeräusche“** nicht mehr gezählt werden und die Funktion von **Knopf A** wegfällt.
Beim Start soll ein **»gelangweilter Smiley«** leuchten.
Bei einem **„Klatschgeräusch“** soll der **Summer** kurz ertönen und ein **»erstaunter Smiley« 500 ms** aufleuchten.

Nachwort:

In dieser Programmieranleitung haben wir mit einfachen Programmideen versucht zu zeigen, wie elektronische Bauteile (LEDs, FD, Pot, Summer) in Micro:bit-Programmen aktiv und passiv eingesetzt werden können. Die gezeigten Programme sollen als Basis für weiterführende, eigene Programmideen dienen.