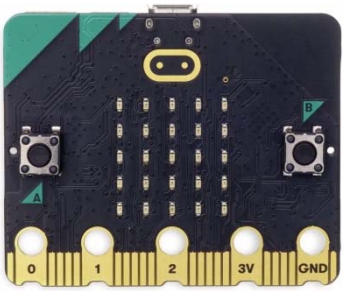
PROGRAMMIERANLEITUNG
Micro:bit ACTIVITY-BOARD



Der **BBC Micro:bit** ist ein preiswerter Minicomputer, der speziell für Schulen entwickelt wurde, um Jugendlichen einen einfachen und spielerischen Einstieg in das Programmieren zu ermöglichen.
Der neue **Micro:bit** (Version 2 = **V2**) hat folgende **Funktionen**:

- LED-Matrix (5 x 5 = 25 LEDs) zur Darstellung von Zeichen, Zahlen, Buchstaben und Texten
- integrierte Sensoren: Beschleunigung, Lage, Licht, Temperatur, Magnetfeld (Kompass)
- integriertes Mikrofon, Lautsprecher und berührungsempfindliches Logo
- 2 Taster (A, B) und 17 Ein- bzw. Ausgänge, die man beliebig programmieren kann
- über Bluetooth bzw. Funksignal kann der Micro:bit mit weiteren Micro:bits oder Tablets und Smartphones verbunden werden

Grundlagen

1. Allgemeine Hinweise:

- Baue das **Activity-Board** laut beiliegender Anleitung zusammen und befestige einen Micro:bit mit zwei Senkkopfschrauben an den Gewindehülsen. Setze zwei neue 1,5 V AAA-Batterien in die Batteriebox (3 V) und fixiere diese mit Hilfe der Zugfeder auf der Unterseite des Boards.
- Aufladbare Batterien (z.B. NiMH, NiCD) haben eine Spannung von 1,2 Volt und sind daher nur bedingt einsetzbar. Ideal und nachhaltig sind aber **Powerbanks** mit USB-Anschluss.
- 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
- eine Batteriebox mit zwei 1,5 V AAA-Batterien (3 V) für den Betrieb ohne Computer (oder eine Powerbank mit USB-Anschluss)

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 Anleitung dazu im pdf-Format findet man auf der **Winkler-Schulbedarf Homepage** www.winklerschulbedarf.com unter Service - Basteltipps (Download) - Micro:bit.

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, z.B. [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, z.B. Schütteln, Neigen, Taste drücken, usw. - Es wird später einfach durch eigene Programme überschrieben!

Wenn der Micro:bit nicht mit dem Computer verbunden ist, benötigt er eine Batteriebox mit zwei 1,5 V **AAA-Batterien** (3 V). Bei einem erneuten PC-Anschluss muss die Batterieversorgung aber nicht getrennt werden, denn der Micro:bit schaltet automatisch auf USB-Versorgung um.

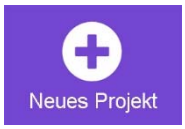
4. Der Makecode-Editor:

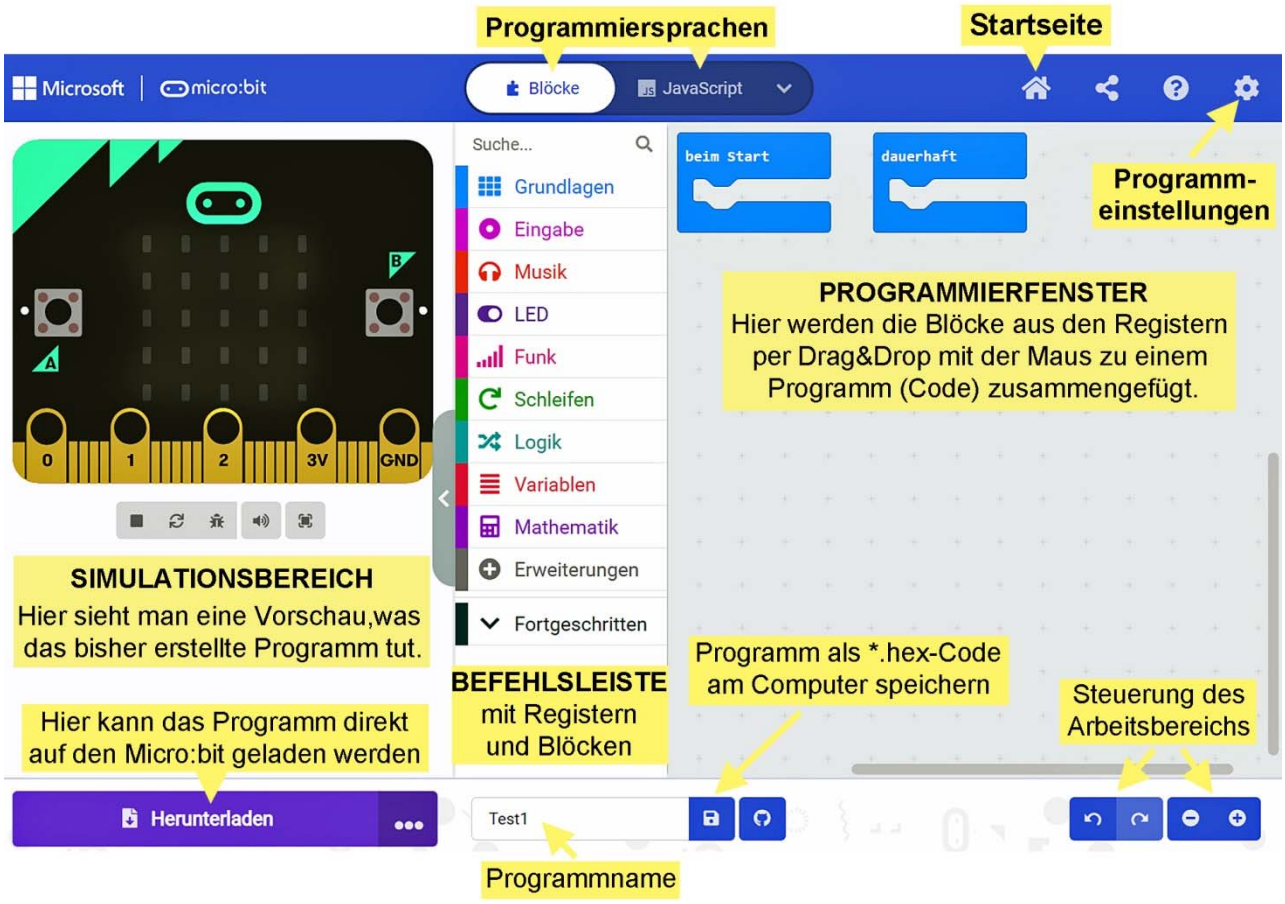
Zum Programmieren verwenden wir die grafische Programmierplattform **Makecode©** von Microsoft: https://makecode.microbit.org/. Eine grafische Programmierung ist ideal für Anfänger, die noch keine Programmiersprache kennen, denn sie ist intuitiv und leicht zu erlernen. **Makecode** läuft im Browser,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

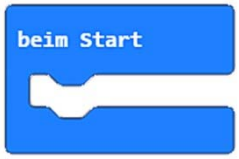
In der Befehlsleiste befinden sich verschiedenfarbige Register mit Blöcken zum Programmieren. Nach Anklicken der Register erscheinen diverse Blöcke, die man mit der Maus per Drag&Drop ins Programmierfenster ziehen kann. Die Blöcke erscheinen im Programmierfenster zuerst grau und nehmen ihre Originalfarbe 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: z.B. Sprache, Programme löschen, zusätzliche Blockregister einfügen usw.

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

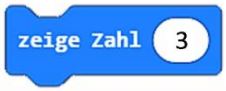
3. Die wichtigsten Blöcke für den Anfang:

- 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**

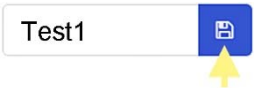
- Aus dem **Register [Schleifen]**:

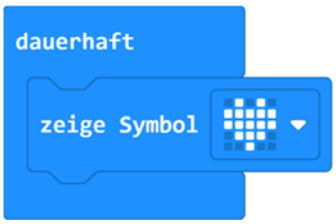
Alle Befehle (Blöcke) innerhalb des **Wiederhol-Blocks** werden in der gewählten Anzahl (hier 4-mal) wiederholt.

4. 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 **Programmnamen** (Test1). Das Programm wird nun lokal auf dem Computer als microbit-Test1.hex gespeichert.





5. Das Testprogramm auf den Micro:bit übertragen:

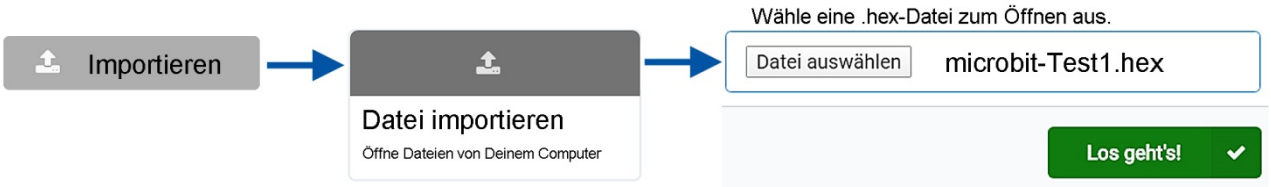
Die **Übertragung des Programms auf den Micro:bit** kann auf **zwei Arten** erfolgen:

- 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.

6. 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** direkt **per Drag&Drop** 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.



Micro:bit-Programme für das Activity-Board

- Programme mit dem Zusatz **„nur V2“** laufen nur am neuen **Micro:bit V2**.
  V2-Sound-Programme kann man aber am **Micro:bit V1** abspielen, wenn man an **Pin 0** und **GND (-)** z.B. mit Hilfe von Krokoklemmen einen hochohmigen **Minilautsprecher** (Buzzer) oder einen **kleinen Ohrhörer** klemmt.
- Mit der **Reset-Taste** auf der Rückseite des Micro:bit kann ein Programm neu gestartet werden.
- Die vorgeschlagenen **Namen** der **„hex-Dateien“** können natürlich geändert werden.

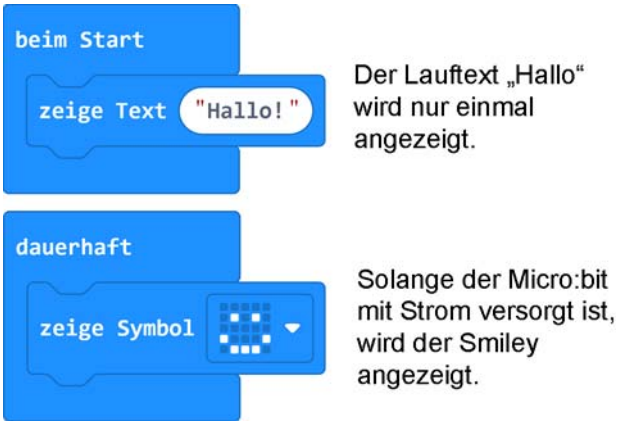
Programm 1: Begrüßung

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

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

Programm-Code: (microbit-Begrüßung1.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-Begrüßung1.hex) auf den Micro:bit.



Weitere Aufgaben:

Ändere den Text auf: **„Ich bin ein Microbit“** und das Symbol auf ein **»Herz-Symbol«.**

Programm 2: Herzklopfen

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

Programm-Code: (microbit-Herzklopfen1.hex)

Dieser **Pause-Block** bewirkt, dass der vorherige Block (Herz) 200 ms lang angezeigt wird.

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

Programm 3: Blinker

Vorgabe: Das kleine »**Quadrat-Symbol«** soll dauerhaft mit einer bestimmten **Frequenz** blinken.

Programm-Code: (microbit-Blinker1.hex)

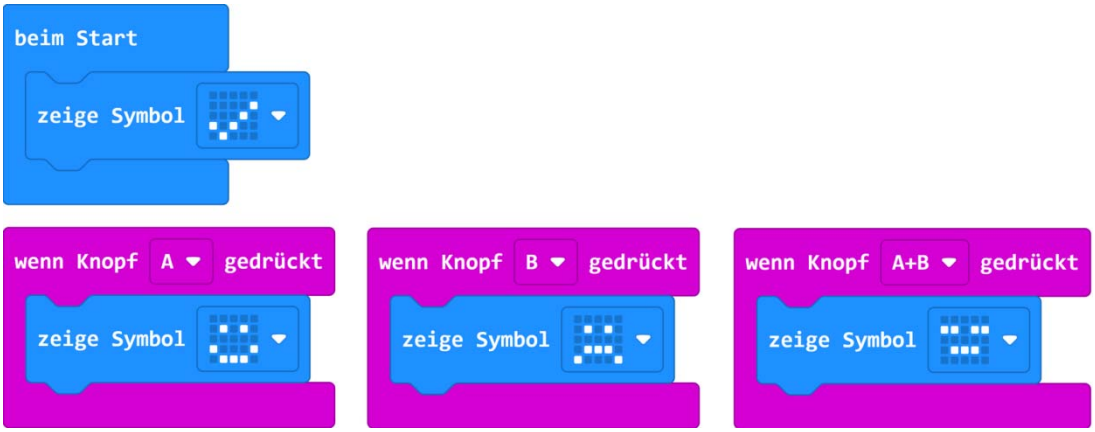
Dieser Block löscht das Quadrat-Symbol für 500 ms.

Weitere Aufgabe: Ändere das **Symbol** und die **Blinkfrequenz**

Programm 4: Knopf A und B

Vorgabe: Beim **Start** soll ein **»Häkchen-Symbol«** leuchten.
Bei Betätigung der **Knöpfe A, B und A+B** sollen verschiedene »**Smileys«** aufleuchten.

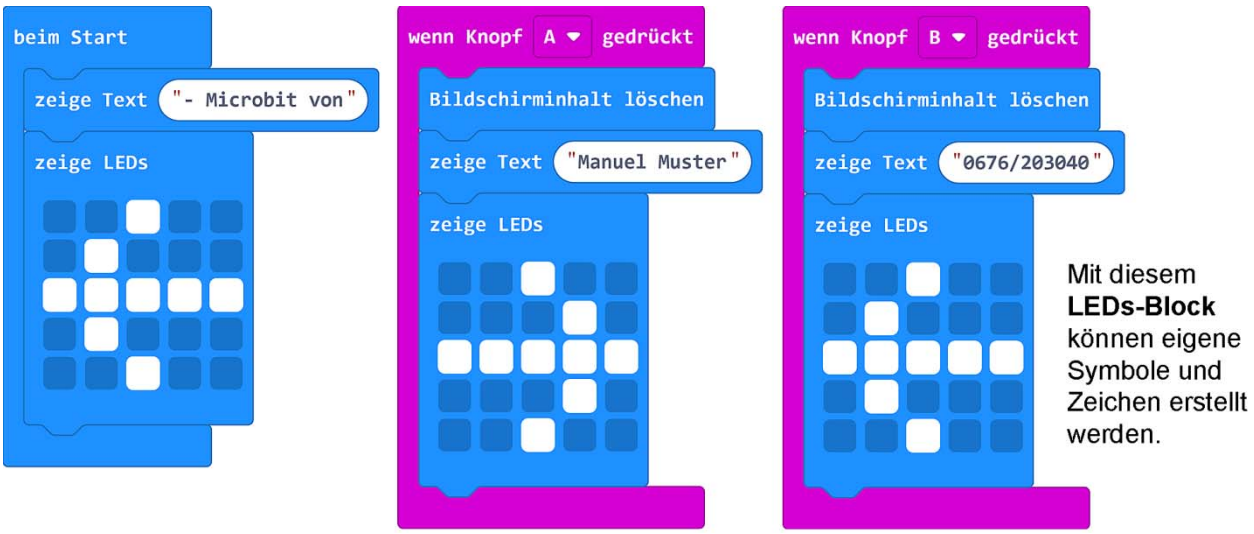
Programm-Code: (microbit-KnopfAB-1.hex)



Programm 5: Name + Telefonnummer

Vorgabe: Der Micro:bit soll nach dem **Starttext „- Microbit von“** und einem **Pfeil** den **Namen** und die **Telefonnummer** des Besitzers nach **Knopfdruck (A, B)** anzeigen.

Programm-Code: (microbit-Name-Telefon.hex)



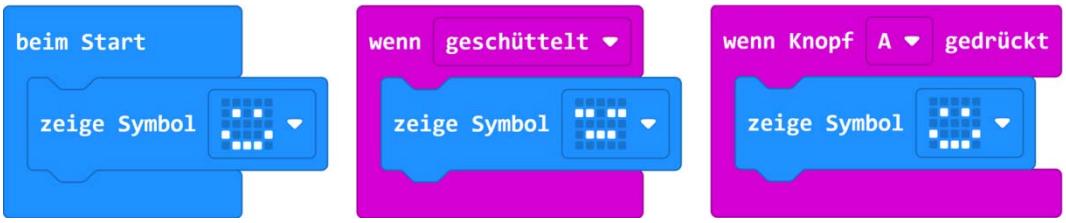
Mit diesem **LEDs-Block** können eigene Symbole und Zeichen erstellt werden.

Programm 6: Schütteln

Die **Block-Klammer »wenn geschüttelt«** vom **Register [Eingabe]** nutzt den **Lagesensor (= Beschleunigungssensor)**, der z.B. durch **Schütteln** aktiviert werden kann.

Vorgabe: Beim Start soll ein **»freundlicher Smiley«** aufleuchten. Wenn man den Micro:bit kräftig schüttelt, soll ein **»grimmiger Smiley«** erscheinen.
Nach einem Klick auf **Knopf A** soll der **»freundliche Smiley«** wieder leuchten.

Programm-Code: (microbit-Schütteln1.hex)



Weitere Aufgabe: Nach dem Schütteln soll der Text **„Hallo“** erscheinen.

Programm 7: Neigen

Verwende vom Register **[Eingabe]** vier Mal die **Block-Klammer »wenn geschüttelt«** (Lagesensor) und passe sie mit der **Pfeilauswahl** an.

Vorgabe: Bei der Neigung nach links soll der Micro:bit ein **„L“** anzeigen; bei der Neigung nach rechts ein **„R“**; bei der Neigung nach vorne (Logo nach unten) ein **„V“** und bei der Neigung nach hinten (Logo nach oben) ein **„Z“**.

Programm-Code: (microbit-Neigen1.hex)

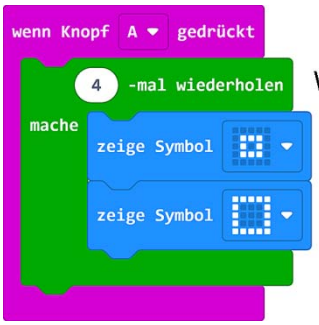
Weitere Aufgabe: Setze an Stelle der Buchstaben vier verschiedene **»Smiley-Symbole«** ein.

Programm 8: Wiederholung 1

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)



Wiederhol-Schleife

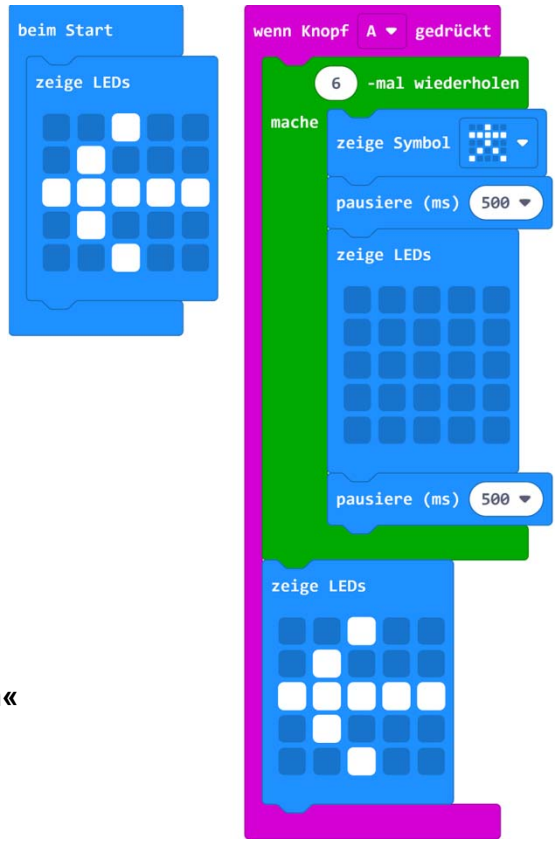
Weitere Aufgaben:

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

Programm 9: Wiederholung 2

Vorgabe: Beim Start soll ein **»Pfeil-Symbol«** in Richtung **Knopf A** zeigen.
Nach einem Klick auf **Knopf A** soll ein **»Figur-Symbol« 6-mal** blinken und dann wieder der **Pfeil** erscheinen.

Programm-Code: (microbit-Wiederholung2.hex)



Weitere Aufgaben:

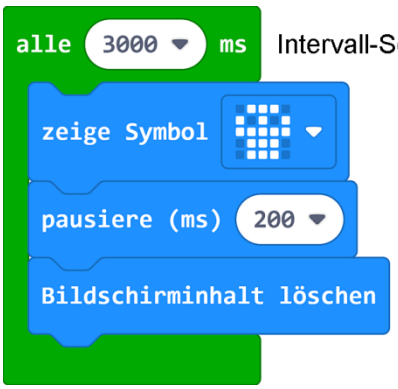
- Ändere die **Anzahl der Wiederholungen**
- Ändere das **Figur-Symbol**
- Ändere die **Blinkfrequenz** an den **»Pause-Blöcken«**

Programm 10: Zeitintervall

Die **»Intervall-Schleife«** aus dem Register **[Schleifen]** wiederholt einen eingefügten Programm-Code fortlaufend in einem bestimmten **Zeitintervall.**

Vorgabe: Das **»Kopf-Symbol«** soll alle 3 Sekunden (= 3000 ms) 200 ms lang aufleuchten.

Programm-Code: (microbit-Zeitintervall1.hex)



Intervall-Schleife

Weitere Aufgaben:

- Ändere das **Zeitintervall** der **»Intervall-Schleife«**
- Ändere das **Symbol** und die **Leuchtdauer**

Programm 11: Zahlenwürfel

Die **mathematische Funktion »Zufallszahl«** befindet sich im **Register [Mathematik]**. Ziehe sie per Drag&Drop in das weiße Feld des **Blocks »zeige Zahl«**.

Vorgabe: Nach Betätigung von **Knopf A** soll auf dem Micro:bit eine **zufällige Zahl von 1 - 6** angezeigt werden. Der Micro:bit wird dadurch zu einem praktischen **Zahlenwürfel**.

Programm-Code: (microbit-Zahlenwürfel1.hex)

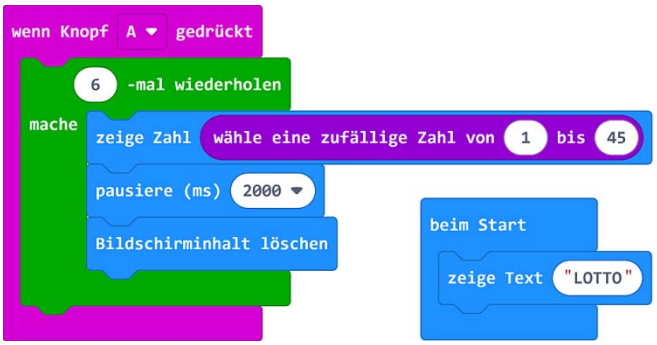
Weitere Aufgaben:

- Die Zufallszahl soll **5 Sekunden** lang angezeigt und dann **gelöscht** werden.
- Die Zufallszahl soll nicht durch Knopf A sondern durch **Schütteln** erzeugt werden.

Programm 12: Lotto 6 aus 45

Vorgabe: Nach dem **Starttext „Lotto“** sollen auf **Knopfdruck (A)** mit Hilfe einer **»Wiederhol-Schleife« 6 zufällige Lottozahlen** von **0 - 45** jeweils **2 Sekunden lang** gezeigt werden.

Programm-Code: (microbit-Lotto1.hex)

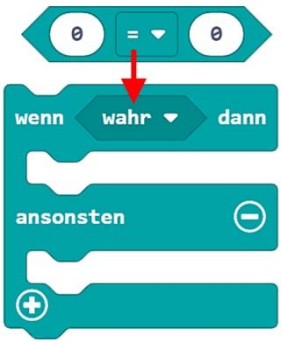


Weitere Aufgaben:

Beim Start soll der Text „**Joker**“ durchlaufen. Ändere das Programm so, dass **6 Jokerzahlen** von **0 - 9** jeweils **3 Sekunden** lang angezeigt werden.

Programm 13: Kopf oder Zahl

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 (z.B. 0 = 0), **dann** wird der obere Programmteil ausgeführt, **ansonsten** der untere.
Am ⊕ kann der »**Wenn-Block«** erweitert, am ⊖ reduziert werden.



Vorgabe: Bei einem Klick auf **Knopf A** soll eine **Zufallszahl** (0 oder 1) gewählt werden. Beträgt sie „0“, soll ein Kopf aufleuchten, ansonsten die Zahl „1“.

Programm-Code:
(microbit-KopfZahl1.hex)

Weitere Aufgaben:

Ergänze das Programm mit einem »**Startblock«** mit Fragezeichen. Kopf und Zahl sollen jeweils nur **2 Sekunden** lang aufleuchten.

Programm 14: 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 **Variablen** den Namen **„Klicks“** und bestätige mit **»OK«**. Es erscheinen dann drei neue, rote Blöcke.

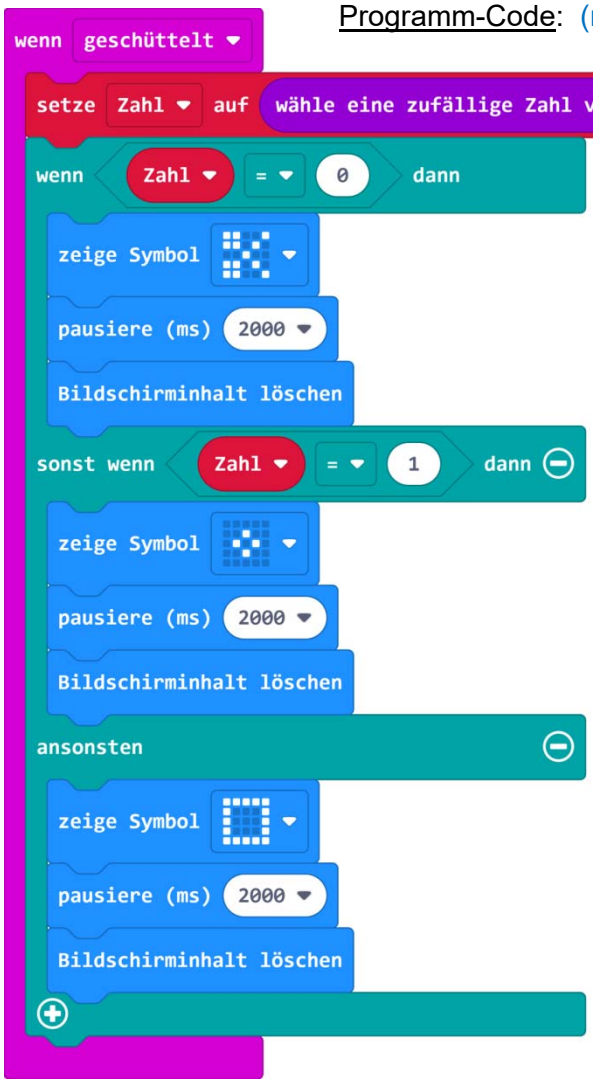
Vorgabe: Setze die **Variable »Klicks«** beim Start auf **„0“**. Bei einem Druck auf **Knopf A** soll die **Klicks-Anzahl** um **„1“ erhöht**, und angezeigt werden. Bei Betätigung von **Knopf B** soll die **Klicks-Anzahl** um **„1“ vermindert** und angezeigt werden.

Programm-Code: (microbit-Klicks-Zähler1)

Programm 15: Schere-Stein-Papier

Vorgabe: Erstelle eine **Variable** mit dem **Namen »Zahl«** und setze sie auf eine **Zufallszahl** von **0 - 2**. Nach dem **Schütteln** des Micro:bit soll durch einen erweiterten »**Wenn-Block«** eines der **drei Symbole** (Schere, Stein oder Papier) **2 Sekunden** lang aufleuchten.

Programm-Code: (microbit-Schere-Stein-Papier1.hex)



Bei der **Zufallszahl „0“** erscheint zwei Sekunden lang ein **Scheren-Symbol.**

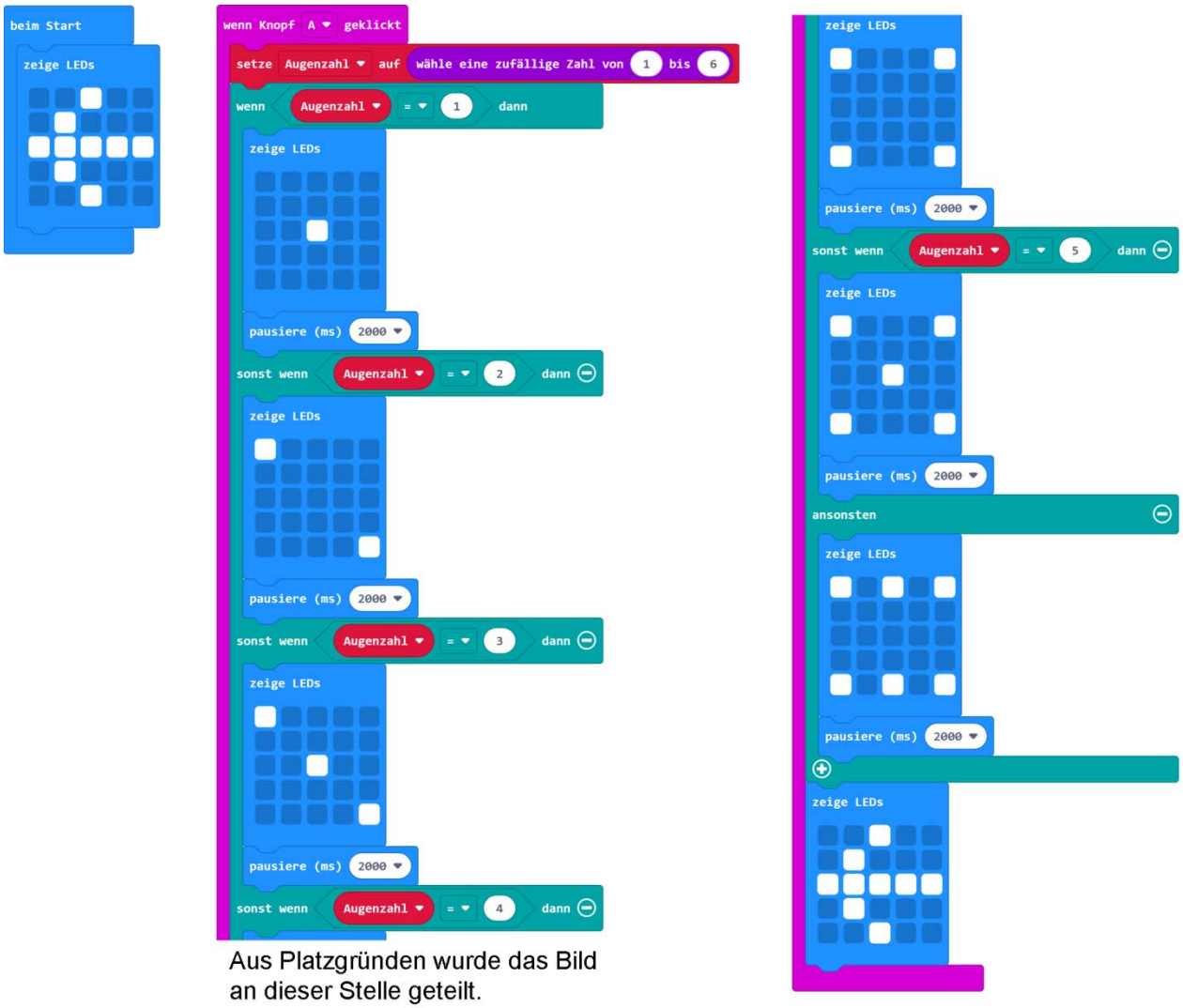
Bei der **Zufallszahl „1“** erscheint zwei Sekunden lang ein **Stein-Symbol.**

Bei der **Zufallszahl „2“** erscheint zwei Sekunden lang ein **Papier-Symbol.**

Programm 16: Augenwürfel

Vorgabe: Beim Start soll ein **Pfeil** in Richtung **Knopf A** zeigen. Erstelle eine **Variable** mit dem **Namen »Augenzahl«** und setze sie auf eine **Zufallszahl** von **1 - 6**.
Nach Betätigung von **Knopf A** sollen durch einen erweiterten »**Wenn-Block«** die Augenzahlen wie bei einem Würfel **grafisch** jeweils **2 Sekunden lang** angezeigt werden.
Nach der Anzeige soll wieder der **Pfeil** in Richtung **Knopf A** erscheinen.

Programm-Code: (microbit-Augenwürfel1.hex)



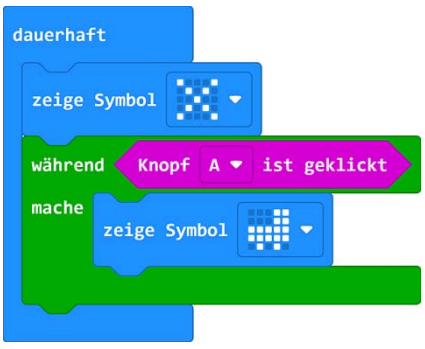
Aus Platzgründen wurde das Bild an dieser Stelle geteilt.

Programm 17: Während-Schleife

Für weitere Programme benötigen wir eine »**Während-Schleife«** aus dem Register **[Schleifen]**. Diese Schleife wiederholt ein eingefügtes Programm, solange die **Bedingung** im sechseckigen Kästchen **wahr** ist.

Vorgabe: Während **Knopf A** geklickt ist, soll ein »**Auto-Symbol«** angezeigt werden. Nach Loslassen von **Knopf A** soll wieder das »**X-Symbol«** aufleuchten.

Programm-Code: (microbit-Während-Schleife1.hex)



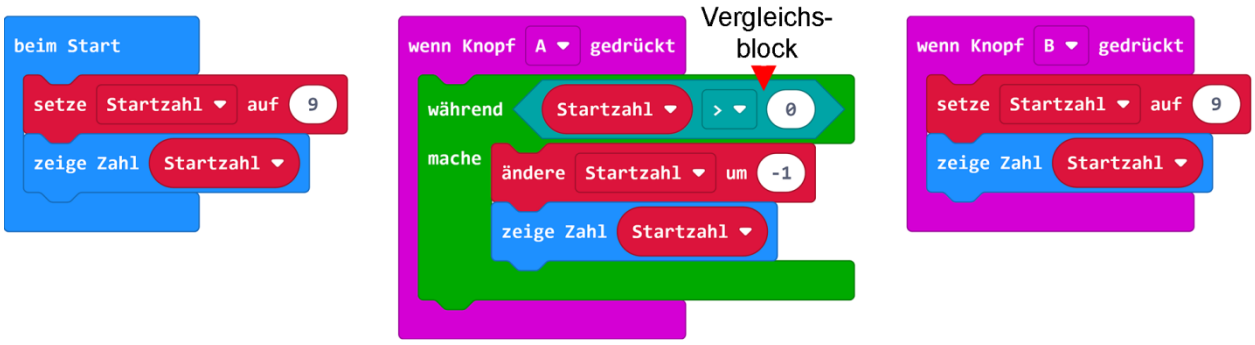
Weitere Aufgabe: Ergänze das Programm mit einer **zweiten »Während-Schleife«**. Bei Betätigung von **Knopf B** soll dann die **Zahl „1“** erscheinen.

Programm 16: Countdown

Für die **Bedingung** in der **»Während-Schleife«** benötigt man für das folgende Programm einen sechseckigen **»Vergleichs-Block« (0 > 0)** aus dem **Register [Logik]**.

Vorgabe: Erstelle eine **Variable** mit dem **Namen »Startzahl«** und setze sie beim **Start** auf **„9“**. Während die **Startzahl** größer (>) als „0“ ist, soll nach Klicken von **Knopf A** die Startzahl jeweils um **„1“ vermindert** und **angezeigt** werden. Mit **Knopf B** wird die **Startzahl** wieder auf **„9“** gesetzt.

Programm-Code: (microbit-Countdown1.hex)

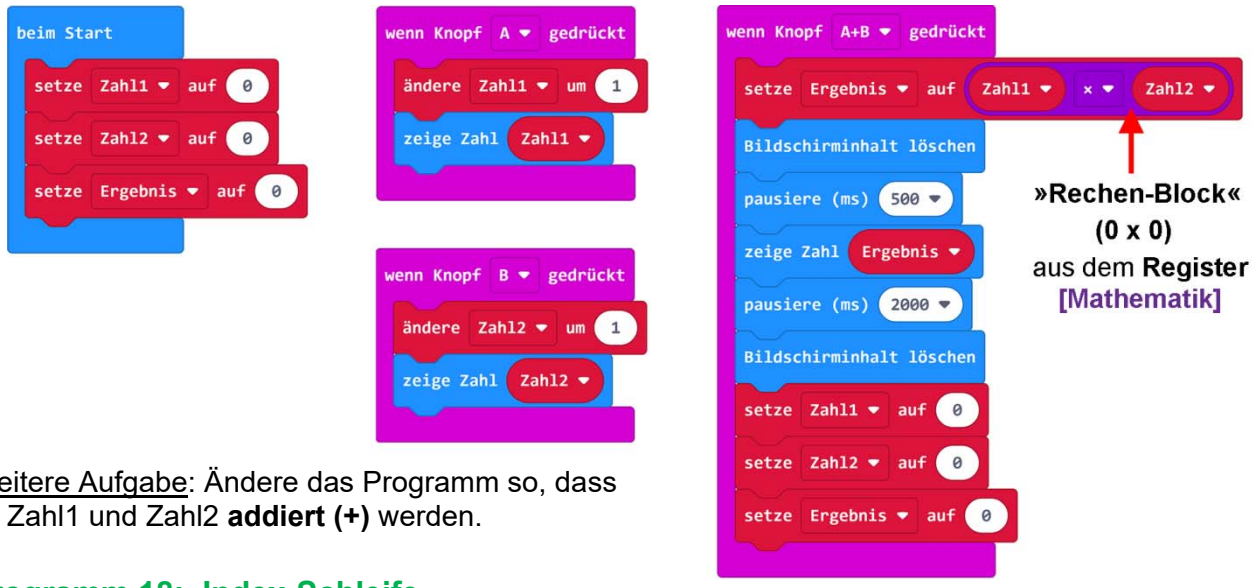


Weitere Aufgaben: Setze die Variable **»Startzahl«** beim **Start** auf **„0“**. Während die **Startzahl** kleiner (<) als **„9“** ist, soll nach Klicken von **Knopf A** die Startzahl jeweils um **„1“ erhöht** und **angezeigt** werden. Mit **Knopf B** soll die **Startzahl** wieder auf **„0“** gesetzt werden.

Programm 17: Rechner

Vorgabe: Erstelle **drei Variablen (»Zahl1«, »Zahl2«, »Ergebnis«)** und setze sie beim **Start** auf **„0“**. Durch mehrfaches Klicken der **Knöpfe A und B** sollen **zwei Zahlen** (Zahl1, Zahl2) erzeugt werden, die **nach Betätigung der Knöpfe A+B multipliziert** (x) werden. Nach **Anzeige des Ergebnisses** sollen die drei **Variablen wieder auf „0“** gesetzt werden.

Programm-Code: (microbit-Rechner1.hex)



Weitere Aufgabe: Ändere das Programm so, dass Zahl1 und Zahl2 **addiert (+)** werden.

Programm 18: Index-Schleife

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

Vorgabe: Nach Betätigung von **Knopf A** sollen mittels **»Index-Schleife«** die **Zahlen von 0 - 12** angezeigt werden.

Programm-Code: (microbit-Index-Schleife1.hex)

Programm 19: LED-Helligkeit

Mit dem **Block »Setze Helligkeit auf«** aus dem **Register [LED / … mehr]** kann man die **Helligkeit der LED-Matrix** ändern. Sie reicht von **„0“** (dunkel) bis **„255“** (hell).

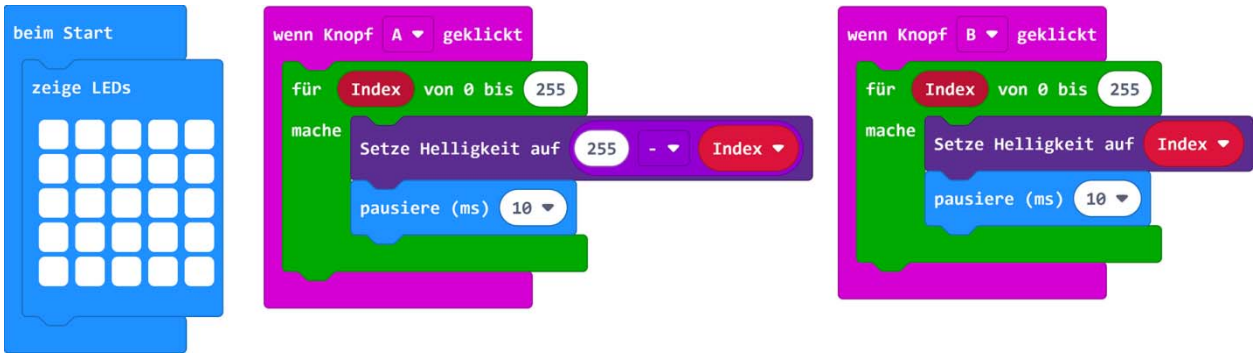
Vorgabe: Setze die **Helligkeit** des **»Herz-Symbols«** mit den **Knöpfen A und B** auf verschiedene Werte (50 / 150 / 255).

Programm-Code: (microbit-LED-Helligkeit1.hex)

Programm 20: LEDs dimmen

Vorgabe: Beim **Start** sollen alle **25 LEDs** hell leuchten. Bei Betätigung von **Knopf A** soll die **Helligkeit der LEDs** langsam **abnehmen** und mittels **Knopf B** wieder **zunehmen**. Verwende dazu **»Index-Schleifen«**, einen **»Rechen-Block«** und die **Variable »Index«**.

Programm-Code: (microbit-LEDs-Dimmen1.hex)



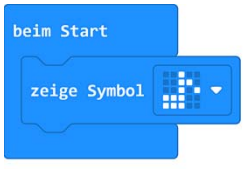
Weitere Aufgabe: Ändere die **Dimm-Geschwindigkeit** durch Ändern der **Pause-Zeiten**.

Programm 21: Sirene (nur V2)

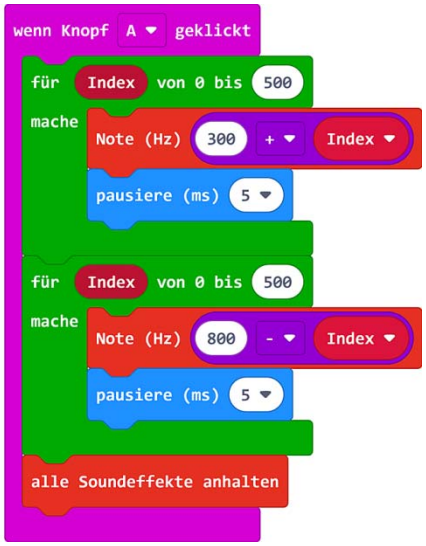
Mit dem **Block »Note (Hz)«** aus dem **Register [Musik]** kann man einen **Ton** mit einer bestimmten **Tonhöhe** (= **Frequenz** in Hertz [**Hz**]) erzeugen.

Vorgabe: Setze beim **Start** ein **»Note-Symbol«**. Mit einem Klick auf **Knopf A** soll ein **Sirenenton** von **300 - 800 Hz**, auf- und abschwellen. Verwende dazu die **»Index-Zahl«** (500) und **zwei »Index-Schleifen«**, die alle **5 ms** den Ton um **1 Hz** erhöhen, bzw. erniedrigen.

Programm-Code: (microbit-Sirene1.hex)



Der **»Rechen-Block (0+0)«** befindet sich im **Register [Mathematik]**.



Weitere Aufgabe:

Setze die zwei **»Index-Schleifen«** in eine **»Wiederhol-Schleife«** und lass den Ton **3-mal** auf- und abschwellen.

Programm 22: Töne erzeugen (nur V2)

Mit dem **Block »spiele Note …«** aus dem **Register [Musik]** kann man eine **bestimmte Note** eine **bestimmte Zeit** (Schlag) erklingen lassen.

Vorgabe: Setze beim **Start** ein **»Noten-Symbol«**. Mit einem Klick auf **Knopf A** soll die **Note „Hohes C“ einen Schlag** lang mit **Pause** (500 ms) **4-mal** erklingen. Mit einem Klick auf **Knopf B** sollen die zwei **Noten „Mittleres G“ und „Hohes C“** abwechselnd jeweils **einen Schlag** lang **6-mal** erklingen.

Programm-Code:

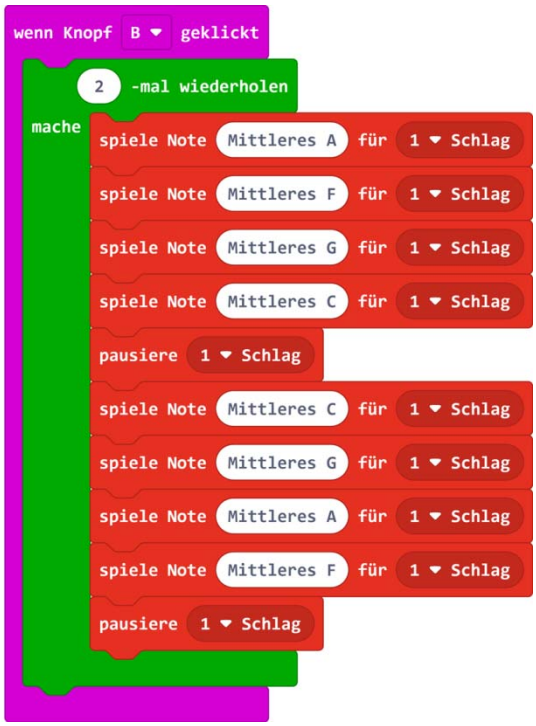
Weitere Aufgaben: Ändere die **Noten**, die **Schlagzahlen** und die **Anzahl** der **Wiederholungen**.

Programm 23: Türgong (nur V2)

Vorgabe: Beim **Start** soll ein **»erstaunter Smiley«** erscheinen. Nach einem Klick auf **Knopf A** soll ein **„3-Klang-Gong“** und mittels **Knopf B 2-mal** die **„Westminster-Melodie“** erklingen.

Programm-Code:

(microbit-Türgong1.hex)



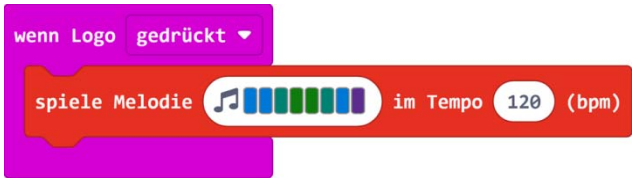
Weitere Aufgabe:

Ändere das Programm so, dass nach Betätigung von **Knopf A** die **„C-Dur Tonleiter“** erklingt. Mittels **Knopf B** sollen die ersten vier Töne von **„Bruder Jakob“** (Mittleres C, D, E, C) **2-mal** hörbar werden.

Programm 24: Kurzmelodie (nur V2)

Vorgabe: Nach Berühren des **Logos** oberhalb der LED-Matrix sollen die ersten **acht Töne** von **„Freude, schöner Götterfunken“** erklingen. (Beginne mit 2 blauen Kästchen!)

Programm-Code: (microbit.Kurzmelodie1.hex)



Weitere Aufgabe:

Erstelle selbst eine **kurze Melodie** (8 Töne).

Programm 25: Metronom (nur V2)

Ein **Metronom** erzeugt eine einstellbare **Anzahl eines Tones pro Minute (bpm)**. Es wird häufig von Musikern verwendet, um ein bestimmtes **Tempo** vorzugeben. Die erforderlichen **Blöcke »ändere das Tempo«** und **»Tempo (bpm)«** befinden sich im **Register [Musik]**.

Vorgabe: Setze das **Tempo** beim Start auf **„100 bpm“**. Als **Ton** soll dauerhaft ein **„Mittleres C“** für **„1/16 Schlag“** ertönen. **Knopf A** soll das Tempo um **5 bpm verringern** und **Knopf B** um **5 bpm erhöhen**. Bei Druck auf die **Knöpfe A+B** soll das Tempo angezeigt werden.

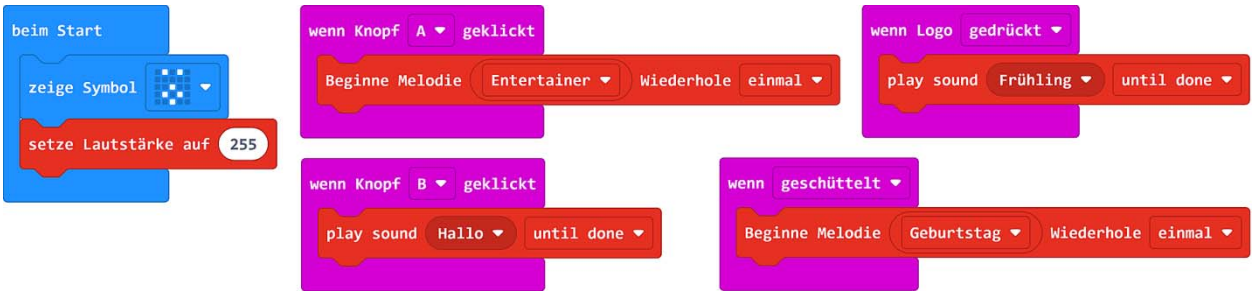
Programm-Code: (microbit-Metronom1.hex)

Programm 26: Soundeffekte (nur V2)

Mit dem **Micro:bit V2** können zahlreiche **Melodien** und **Soundeffekte** abgespielt werden. Man findet die **Blöcke »Beginne Melodie«** und **»play sound«** im **Register [Musik]**.

Vorgabe: Zeige beim **Start** ein **Symbol** und setze die **Lautstärke auf „255“**. Durch die **Knöpfe A** und **B**, **Logo drücken** und **Schütteln** sollen **2 Melodien** und **2 Soundeffekte** ablaufen.

Programm-Code: (microbit.Soundeffekte1.hex)



Weitere Aufgabe: Ändere die **Melodien** und **Soundeffekte**.

Programm 27: Klatsch-Schalter (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**.

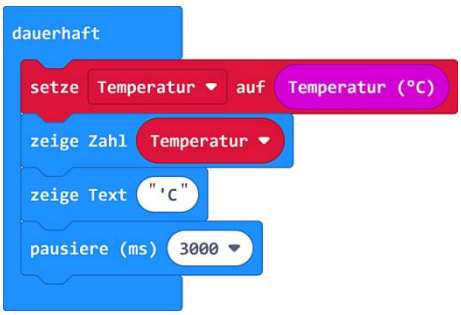
Vorgabe: Setze beim **Start** ein **Symbol** und stelle den **Schwellenwert** auf **128**. Der **Block »wenn laut«** soll auf ein **Klatsch-Geräusch** reagieren und eine **Melodie einmal** abspielen.

Programm-Code: (microbit-Klatsch-Schalter1.hex)

Programm 28: Thermometer

Der Micro:bit hat einen **Temperatursensor**, dessen Wert mit dem **Block »Temperatur (°C)«** aus dem **Register [Eingabe]** abgerufen werden kann.

Vorgabe: Setze eine **Variable »Temperatur (°C)«** und aktiviere damit den **Temperatursensor** des Micro:bit. Die Temperatur soll in **„°C“** alle **3 Sekunden** angezeigt werden.



Programm-Code: (microbit-Thermometer1.hex)

Programm 29: Kompass

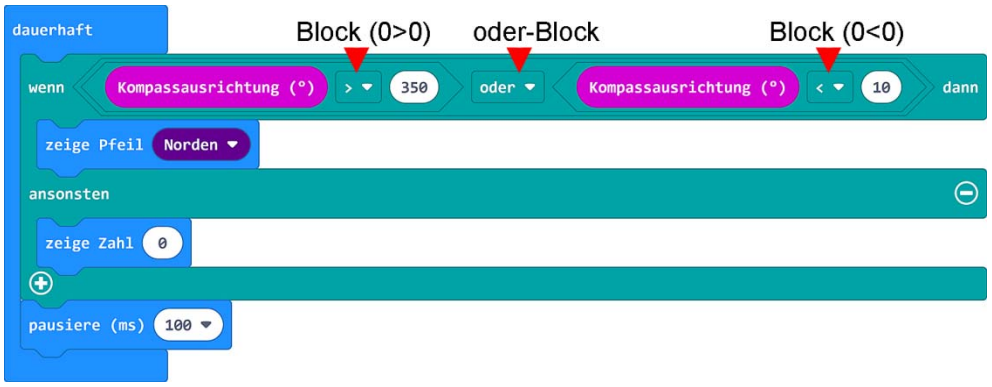
Der Micro:bit hat einen **Magnetfeld-Sensor**, der mit dem **Block »Kompassausrichtung (°)«** aus dem **Register [Eingabe]** aktiviert wird.
Beim ersten Programmstart muss der Kompass zuerst **kalibriert** werden. Es erscheint die Meldung: „Tilt to fill screen.“ Neige und drehe dazu den Micro:bit, bis alle Matrix-LEDs leuchten.

Vorgabe: Mit einem **»Wenn-Block«** soll beim **Drehen** des Micro:bit der „**Norden**“ innerhalb eines **Grenzbereiches** von **+/- 10°** mit einem **Pfeil** angezeigt werden. Außerhalb des Grenzbereiches soll die **Zahl „0“** aufleuchten.

Programm-Code: (microbit-Kompass1.hex)

Weitere Aufgabe:

Mache den Kompass **genauer**: Der Pfeil „Norden“ soll nur zwischen **357° und 3°** aufleuchten.

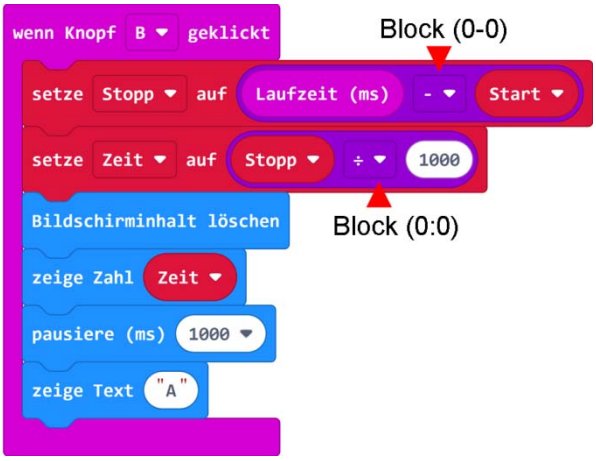


Programm 30: Stoppuhr

Beim Einschalten des Micro:bit beginnt eine **interne Uhr** (in ms) zu laufen. Aktuelle Zeiten können mit dem **Block »Laufzeit (ms)«** aus dem **Register [Eingabe]** abgerufen, in **Variablen** zwischengespeichert und bearbeitet werden. Die erforderlichen **Rechen-Blöcke »0 - 0«** und **»0 : 0«** befinden sich im **Register [Mathematik].**

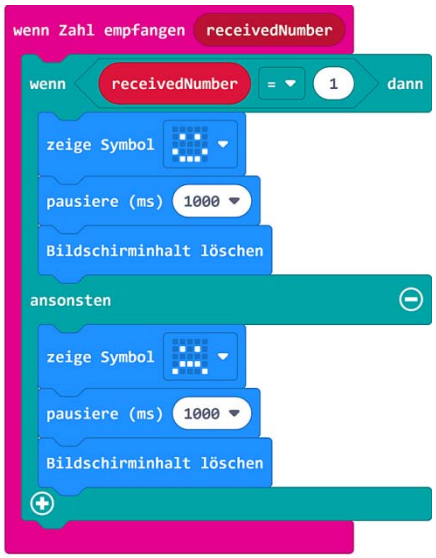
Vorgabe: Erstelle **drei Variablen (»Start«, »Stopp«, »Zeit«)**. Speichere die **Startzeit** mit **Knopf A** und die **Stoppzeit** mit **Knopf B.** Berechne die zurückgelegte **Zeit** und dividiere sie durch **1000**, damit das **Ergebnis in Sekunden** angezeigt wird.

Programm-Code: (microbit-Stoppuhr1.hex)



Programm 31: Funksignale

Wenn zwei Micro:bits auf einen gleichen **Funkkanal** (1 - 250) gesetzt werden, können sie miteinander in Verbindung treten. Senden kann man **Zahlen** und kurze **Texte**.
Die erforderlichen **Blöcke** befinden sich im **Register [Funk]**.



Vorgabe: Nach dem Einstellen eines **Funkkanals** (hier 12) soll mittels **Knopf A** die **Zahl „1“** und mit **Knopf B** die **Zahl „2“** gesendet werden. Beim **Empfänger** soll dann entweder ein **lachender** oder ein **böser Smiley eine Sekunde** lang aufleuchten.

Programm-Code:

(microbit-Funk1.hex)

Dieses Programm muss auf beide Micro:bits übertragen werden!

Weitere Aufgabe:

Füge nach den Smileys die **Sound-effekte »play sound«** „**Hallo**“ bzw. „**traurig**“ in das Programm ein.

Programm 32: Morsen (nur V2)

Beim Morsen werden die Buchstaben des Alphabets laut nebenstehender Liste als **kurze** (K) und **lange** (L) **Signale** gesendet.

Aufgabe für Fortgeschrittene:

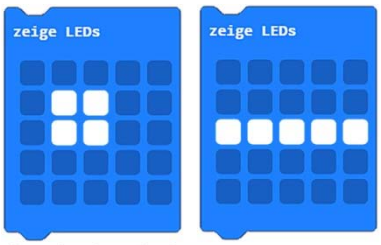
Mache aus dem **Programm 31** (Funksignale) einen **Morsesender** und **-empfänger**:

Setze die Funkgruppe z.B. auf „5“. Bei Klicken von **Knopf A** soll der **Text „K“** (**»sende Text „K“ über Funk«**) und von **Knopf B** der **Text „L“** gesendet werden.

Wird ein **„K**“ (= kurzes Signal) gesendet, soll beim **Empfänger** (**»wenn Text empfangen [receivedString = K]«**) ein **»Punkt-Symbol«** aufleuchten und die **Note „Hohes C“** für einen **halben Schlag** erklingen.
Ansonsten soll bei „L“ ein **»Strich-Symbol«** aufleuchten und die **Note „Hohes C“ zwei Schläge** lang erklingen.

Viel Erfolg beim Programmieren!

Morsealphabet			
A	•—	N	—•
B	—•••	O	———
C	—•—•	P	•——•
D	—••	Q	——•—
E	•	R	•—•
F	••—•	S	•••
G	——•	T	—
H	••••	U	••—
I	••	V	•••—
J	•———	W	•——
K	—•—	X	—••—
L	•—••	Y	—•——
M	——	Z	——••



Punkt-Symbol Strich-Symbol

Nachwort:

Mit dieser Programmieranleitung für Micro:bit-Einsteiger haben wir versucht, die wichtigsten Grundfunktionen dieses faszinierenden Minicomputers mit einfachen bis leicht fortgeschrittenen Programmideen zu veranschaulichen und zu festigen. Die gezeigten Programme sollen als Basis für weiterführende eigene Programmideen dienen.