Die Calliope mini

Die **LED-Matrix** besteht aus einem 5x5-Raster roter Leuchtdioden, auf dem du dir Muster und Texte anzeigen lassen kannst. Jedes der roten Lämpchen kann einzeln angesteuert werden. Außerdem kannst du mit dem integrierten Lichtsensor die Helligkeit in der Umgebung messen.

Der **USB-Anschluss** dient der Verbindung zwischen Calliope und Computer. Durch ein eingestecktes Verbindungskabel wird dein Programm von deinem Computer auf die Calliope übertragen.

Die **Groove-Adapter** sind Verbindungs-Stecker, an die Erweiterungsmodule, wie beispielsweise weitere Sensoren, angeschlossen werden können.

Der **Reset-Knopf** dient dem Neustarten eines Programms auf der Calliope.

Die **Status-LED** leuchtet gelb, wenn die Calliope an Strom angeschlossen ist, und blinkt, wenn ein Programm auf die Calliope übertragen wird.

Die **Touch-Pins + und –** können einen Stromkreis bilden, um beispielsweise eine externe LED dauerhaft leuchten zu lassen.

Die **Tasten A und B** sind Eingaben. Wird eine Taste gedrückt, so wird ein Schaltkreis geschlossen und die Calliope kann auf die Eingabe reagieren. Über dein Programm kannst du selbst bestimmen, wie die Calliope auf das Drücken einer oder beider Tasten reagiert.

Der **Lautsprecher** kann Töne wiedergeben oder Musik abspielen.

Die **RGB-LED** kannst du in unzähligen verschiedenen bunten Farben leuchten lassen.

Funk ermöglicht die Kommunikation zwischen zwei Calliopes. Über Funk werden Signale gesendet und so Nachrichten weitergegeben.

Der **Batterie-Anschluss** dient der Stromversorgung der Calliope. Solange die Calliope per USB am Computer angeschlossen ist, ist eine weitere Stromversorgung nicht nötig.

Bluetooth ermöglicht, die Calliope mit einem Smartphone oder Tablet zu verbinden und so Daten zu übertragen.

Die **Touch-Pins 0, 1, 2 und 3** sind Eingaben. Da die Pins berührungsempfindlich sind, geht dies durch einfaches Anfassen. Für eine Eingabe über die Touch-Pins muss ein geschlossener Stromkreis erzeugt werden, daher einfach gleichzeitig mit der anderen Hand den Minus-Pin (-) berühren, denn die Pins 0–3 agieren wie Pluspole. Außerdem kannst du hier auch weitere Sensoren anschließen.

Der **Prozessor** ist das Herz der Calliope. Er verbindet alle Funktionen der Calliope miteinander und verarbeitet alle Informationen und Befehle.

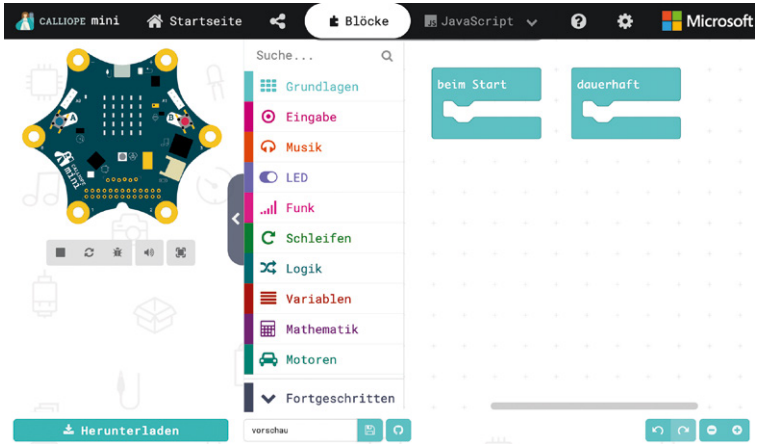
Die **verschiedenen Sensoren** können Werte ihrer entsprechenden Funktion messen. Der kombinierte **Lage-, Bewegungs- und Beschleunigungssensor mit Kompass** kann erkennen, ob die Calliope bewegt wird, wie schnell sie bewegt wird, in welche Richtung sie gehalten wird und wie sie gedreht ist. Durch den **Temperatur- und den Lautstärkesensor** können die Umgebungswärme und die Lautstärke in der Umgebung gemessen werden.

Die **Motor-Pins** ermöglichen das Anschließen von bis zu zwei Motoren, die dann durch dein Programm gesteuert werden können.

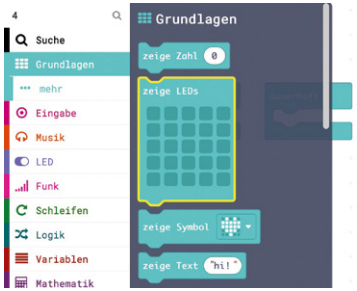
Wie funktioniert die Programmierumgebung MakeCode?

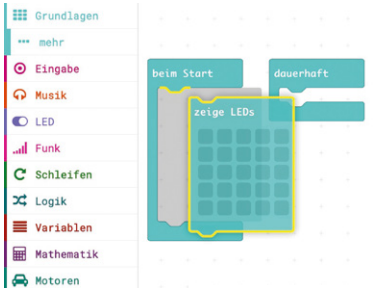
Den in diesen Workshops verwendeten Editor findest du unter https://makecode.calliope.cc. Auf der Startseite werden alle bereits programmierten Projekte unter „Meine Projekte“ angezeigt. Starte ein neues Projekt, um zu beginnen.

Der Aufbau dieses Editors ist wie folgt: Links ist eine Calliope zu sehen, die das geschriebene Programm live simuliert, in der Mitte befinden sich die Programmierblöcke, unterteilt in verschiedene Kategorien, und rechts steht der Programmcode. Beim Starten eines neuen Projekts besteht der Code aus einem leeren „beim Start“- und einem leeren „dauerhaft“-Block.



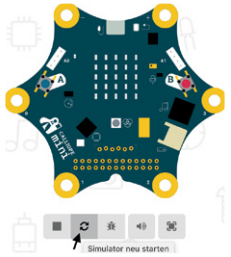
Klickst du auf eine der Block-Kategorien, so zeigt sich eine Auswahl an Programmierblöcken. Um einen der Codeblöcke zu verwenden, muss dieser mit gehaltener linker Maustaste an die gewünschte Stelle gezogen werden.
Die Form der einzelnen Blöcke gibt Aufschluss darüber, wo die Blöcke angehängt werden können.





Um einen Block wieder zu löschen, muss dieser einfach zurück auf die Blockauswahl gezogen werden. Alternativ kann auch mit der Tastensteuerung, wie STRG+C und ENTF, gearbeitet werden.

Um die Simulation des Programms zu steuern, befinden sich unter der angezeigten Calliope verschiedene Knöpfe. Verweilst du mit der Maus auf diesen, so werden die entsprechenden Erklärungen angezeigt.



Um das Programm auf die Calliope zu laden, klicke unten links auf „Herunterladen“. Es erscheint ein Pop-up-Fenster mit allen weiteren Anweisungen.

Die Dateien werden im .hex-Format heruntergeladen und können als solche auch vom Rechner in den MakeCode-Editor importiert werden. Um ein Programm zu importieren, kann entweder der entsprechende Button auf der Startseite verwendet oder die Datei per Drag’n’Drop in den Editor geladen werden.

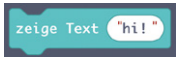
Ein **Programm** ist eine Arbeitsanweisung zur Lösung eines Problems oder einer Aufgabe, die in einer Programmiersprache so präzise geschrieben ist, dass sie von einer Maschine wie einem Computer, einem Handy oder eurer Calliope ausgeführt werden kann.

Nicht nur in der Informatik nennt man solche Arbeitsanweisungen einen **Algorithmus**. Auch im Alltag kommen Algorithmen vor. Das Backen eines Kuchens ist hierfür das beste Beispiel. Um diese Aufgabe zu lösen, wird immer gleich vorgegangen und damit ist auch das Ergebnis immer das Gleiche – ein leckerer Kuchen. Wie bei jedem Algorithmus ist auch hierbei die genaue Einhaltung der einzelnen Schritte wichtig: Beispielsweise kannst du einen Kuchen nicht zuerst backen und dann erst die Zutaten verrühren.

Selbst sehr schwierige Algorithmen werden meist in einfachen Teilschritten gelöst.

1 Das erste Programm: Hello World!

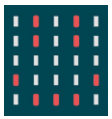
Beim Starten der Calliope soll der Schriftzug „Hello World!“ über die LED-Matrix laufen.



Strukturen, die bestimmte Befehlsfolgen wiederholt ausführen lassen, nennt man **Schleifen**. Dabei kann die Anzahl der Wiederholungen fest vorgegeben oder an eine Bedingung geknüpft sein. Auch endlose Wiederholungen sind möglich.

2 Aktion – Reaktion

2.1 Zunächst soll ein Smiley *dauerhaft* auf der LED-Matrix zu sehen sein.

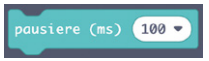


Verzweigungen sind Strukturen, die dazu dienen, alternative Abläufe zu beschreiben. Es handelt sich somit um Fallunterscheidungen, bei denen ein Codeteil ausgeführt wird, wenn eine Bedingung erfüllt ist, und ein anderer, wenn die Bedingung nicht erfüllt ist.

2.2 Erst wenn Taste A gedrückt wird, wird der Smiley angezeigt.

2.3 Wenn Taste B gedrückt wird, wird dein Name angezeigt.

2.4 Nachdem dein Name angezeigt wurde, soll dein Alter angezeigt werden. Warte kurz zwischen den beiden Anzeigen.



3 Sensoren

3.1 Wenn die Calliope geschüttelt wird, wird ein Muster deiner Wahl angezeigt.

3.2 Wird die Calliope nach rechts geneigt, so erscheint ein Pfeil nach rechts. Wird die Calliope nach links geneigt, so erscheint ein Pfeil nach links.

4 Musik

4.1 Programmiere das hier abgebildete Programm nach. Kannst du das Lied schon erkennen?

Bislang hast du Schleifen nur verwendet, um ein Programm dauerhaft ablaufen zu lassen. Nun wollen wir Schleifen auch innerhalb eines Programms verwenden.

4.2 Spiele die Musikfolge von eben zweimal hintereinander ab.

4.3 Die Musik soll nicht direkt beim Starten der Calliope anfangen zu spielen, sondern erst, wenn Taste A gedrückt wird. Außerdem soll nach der Musik der Text „Happy Birthday“ angezeigt werden.

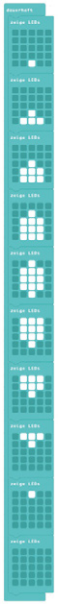
5 99 Luftballons

Variablen, oft auch Platzhalter genannt, sind Wertespeicher, die zum Merken und Verwalten von Daten dienen. Legt man eine neue Variable an, so nennt man das das Deklarieren einer Variablen. Eine Variable kann einen Wert zugewiesen bekommen, dann entspricht die Variable diesem Wert. Den Wert einer Variablen kann man jederzeit ändern. Bekommt eine Variable zu Beginn des Programms einen Wert zugewiesen, so nennt man das das Initialisieren einer Variablen.

Wir wollen Luftballons steigen lassen und die Calliope soll mitzählen, wie viele bereits abgehoben sind.

5.1 Erstelle Bilder, die hintereinander ausgeführt das Abheben eines Luftballons darstellen.

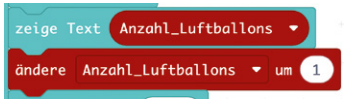
5.2 Um die aufgestiegenen Luftballons zu zählen, musst du eine Variable erstellen.
Diese Variable nennen wir *Anzahl_Luftballons*.
Mit welchem Wert musst du die Variable initialisieren und wann muss die Initialisierung passieren?
Zur Erinnerung: Variablen werden **einmalig** initialisiert.



5.3 Wann muss die Variable erhöht werden? Erhöhe *Anzahl_Luftballons* an der entsprechenden Stelle.

5.4 Lasse dir nach jedem Luftballon die Anzahl der fliegenden Ballone anzeigen.
Hinweis: Das Runde passt ins Runde.

5.5 Überlege dir kurz, wieso die hier dargestellte Reihenfolge nicht die richtige Anzahl an Luftballons anzeigen würde.



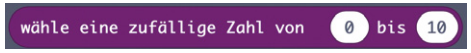
Du kennst nun die Grundfunktionen der Calliope.

Du hast noch Zeit?
Dann werde selbst kreativ. Hier ein paar Ideen:

- Komponiere ein eigenes Lied.
- Erstelle ein Daumenkino.
- Spiele mit der RGB-LED und den Sensoren: Wenn es beispielsweise zu laut in der Klasse ist, dann wird die RGB-LED rot, sonst leuchtet sie grün.

1 Das ist doch Zufall!

Designe deinen eigenen 12er-Würfel. Wenn die Calliope geschüttelt wird, soll eine Zufallszahl zwischen 1 und 12 angezeigt werden.

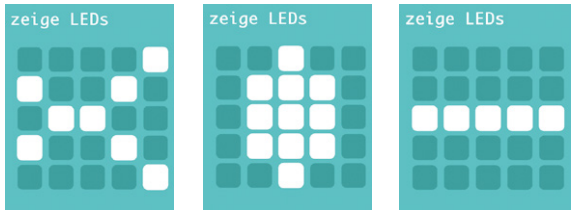


2 „Schere, Stein, Papier“

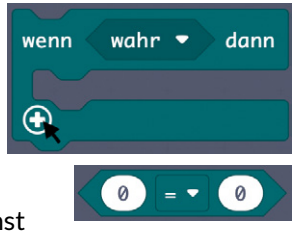
Die Calliope soll gegen dich „Schere, Stein, Papier“ spielen können.

2.1 Wenn die Calliope geschüttelt wird, soll eines der drei Symbole Schere, Stein oder Papier angezeigt werden. Erstelle dazu zunächst eine Zufallszahl, welche die jeweiligen Symbole Schere, Stein und Papier repräsentieren soll. Speichere diese Zufallszahl in einer Variablen, die du *Symbol* nennst.
Info: Die Calliope mini kann nur zufällige Zahlen auswählen, keine zufälligen Bilder. Deshalb muss man einen kleinen Umweg gehen und den Bildern Zahlen zuweisen.

2.2 Nun muss die Zufallszahl noch als Symbol und nicht als Zahl ausgegeben werden. Zeichne dazu die jeweiligen Symbole auf der LED-Matrix. Beispielsweise so:



2.3 Zuletzt muss jeder Zufallszahl eine der Zeichnungen zugeteilt werden.
Hinweis: Da du zwischen drei Fällen – Schere, Stein oder Papier – unterscheiden musst, benötigst du eine Verzweigung. Bislang hast du nur mit Verzweigungen gearbeitet, die der Beginn eines Programms waren, beispielsweise „Wenn geschüttelt“. Verzweigungen können aber auch innerhalb eines Programms verwendet werden. Nutze dazu den hier abgebildeten Block. Um mehrere Unterscheidungen hinzuzufügen, klicke auf das Plus-Symbol.



Das Feld, in dem jetzt noch „wahr“ steht, kannst du durch andere Inhalte ersetzen. Welche Inhalte eingesetzt werden können, erkennst du an der Form. Wenn beispielsweise die Variable *Symbol* gleich 1 ist, dann passiert was? Und so weiter.

2.4 Zähle mit, wer öfter gewinnt, du oder die Calliope.
Lege dazu zwei Variablen an, die deine Punkte und die der Calliope mitzählen sollen. Hast du einen Durchgang gewonnen, drücke Taste A und deine Punkte werden um eins erhöht. Hat die Calliope gewonnen, drücke Taste B und ihre Punkte werden um eins erhöht. Werden die Tasten A und B gleichzeitig gedrückt, so werden zuerst deine Punkte und dann die der Calliope angezeigt.

3 Whac a Mole (Schlag den Maulwurf)

Wenn der Maulwurf aus seinem Loch schaut, muss er so schnell wie möglich gefangen werden. Das Auftauchen des Maulwurfs wird auf der Calliope durch die LEDs dargestellt. Gefangen werden soll der Maulwurf mit den Tasten A und B.

3.1 Der Spielaufbau

Wir unterteilen das Spielfeld in zwei Hälften. Später soll der Maulwurf entweder auf der linken oder auf der rechten Seite auftauchen und dementsprechend mit Taste A oder Taste B gefangen werden.

3.2 Auftauchen des Maulwurfs

Erstelle eine Zufallszahl, die du *Maulwurf* nennst und die die Werte 0 oder 1 annehmen kann.
Wenn die Zufallszahl **Maulwurf = 0** ist, dann soll der Maulwurf **links** auftauchen. Wenn die Zufallszahl **Maulwurf = 1** ist, dann soll der Maulwurf **rechts** auftauchen. Pausiere nach dem Auftauchen des Maulwurfs eine halbe Sekunde (500 ms), bevor der nächste Maulwurf auftaucht.

3.3 Fangen des Maulwurfs

Wurde die richtige Taste zum Fangen des Maulwurfs gedrückt, so soll ein Haken auf der LED-Matrix erscheinen. Andernfalls soll ein Kreuz angezeigt werden. Lasse die jeweilige Anzeige für eine halbe Sekunde stehen.

Zur Erinnerung: Taucht der Maulwurf links auf, so wird er mit Taste A gefangen. Taucht er auf der rechten Seite auf, so wird er mit Taste B gefangen.

Hinweis: Du musst mit sogenannten verschachtelten Verzweigungen, d. h. zwei Verzweigungen ineinander, arbeiten:

Wenn die Taste A gedrückt wird,
soll *entweder* ein Haken
oder ein Kreuz erscheinen.

Was ist die Bedingung dafür, dass ein Haken erscheint?

3.4 Spielstand anzeigen

Um mitzuzählen, wie oft der Maulwurf richtig getroffen wurde, musst du eine neue Variable erstellen, die du *Spielstand* nennst. Vergiss nicht, diese Variable zu initialisieren. Wenn der Maulwurf richtig getroffen wurde, soll sich der Spielstand um einen Punkt erhöhen. Außerdem soll anstelle des Hakens dann der neue Spielstand angezeigt werden und ein Ton erklingen. Wurde der Maulwurf nicht richtig getroffen, bleibt es bei einem angezeigten Kreuz.

3.5 Zwei Maulwürfe

Ergänze dein Programm so, dass auch auf beiden Seiten gleichzeitig jeweils ein Maulwurf erscheinen kann. Um beide Maulwürfe zu fangen, müssen die Tasten A und B gleichzeitig gedrückt werden.

3.6 Werde kreativ

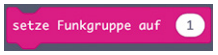
Erweitere dein Spiel so, wie es dir gefällt. Beispielsweise könnte

- ein Spiel nach 10 Runden vorbei sein,
- der Maulwurf an verschiedenen Stellen auftauchen,
- …

1 Funk

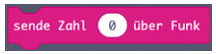
Zwei Calliopes sollen sich gegenseitig Nachrichten schicken.

1.1 Findet euch in Gruppen zusammen, sodass jede Gruppe zwei Calliopes besitzt. Einigt euch auf eine Zahl, diese Zahl stellt ihr *beim Start* des Programms als Funkgruppe ein. Achtet darauf, dass **jede Gruppe ihre eigene Funkgruppe** besitzt.



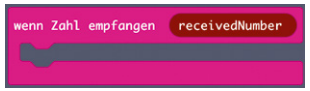
Info: Damit zwei Calliopes miteinander kommunizieren können, müssen sie in der gleichen Funkgruppe sein. Die Funkkanäle gehen von 0 bis 255.

1.2 **Senden:** Wenn Taste A gedrückt wird, soll eine beliebige Nummer versendet werden.



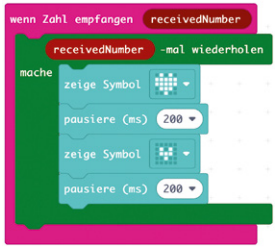
Tipp: Sobald du in der Simulation etwas sendest, erscheint eine zweite Calliope zur Vorschau. Du kannst also auch deine Programme mit Funk hier testen, bevor du sie auf deine Calliope lädst.

1.3 **Empfangen:** Wenn deine Calliope eine Nummer empfangen hat, soll dir dies durch ein Herz auf der LED-Matrix angezeigt werden.

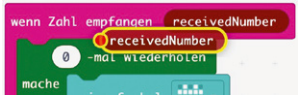


Info: *receivedNumber* ist Englisch und bedeutet *empfangene Nummer*. Wie du auch an der Form erkennen kannst, ist *receivedNumber* eine Variable. Das bedeutet, dass in *receivedNumber* die Zahl gespeichert wird, die über Funk gesendet wurde.

1.4 Ergänze dein Programm so, wie hier abgebildet. Sende verschiedene Zahlen und beobachte, was passiert. Wie oft pocht das Herz?



Hinweis: Klicke auf die Variable *received-Number* und halte die linke Maustaste gedrückt. Nun kannst du die Variable an die gewünschte Stelle ziehen.



1.5 Genauso wie Zahlen können auch Texte über Funk verschickt werden. Erweitere dein Programm so, dass, wenn Taste B gedrückt wird, ein Text über Funk versendet wird. Wenn ein Text empfangen wird, soll genau dieser auf der LED-Matrix angezeigt werden.

Info: *receivedString* ist Englisch und bedeutet *empfangene Zeichenfolge*. Genauso wie *receivedNumber* ist auch *receivedString* eine Variable, mit dem Unterschied, dass in *receivedString* keine Zahl, sondern die Zeichenfolge gespeichert wird, die über Funk gesendet wurde.

1.6 Wenn du bislang etwas sendest, weißt du nicht, ob dies geklappt hat. Dies soll nun geändert werden. Nach dem Senden soll auf deiner LED-Matrix ein Haken erscheinen. Dieser soll dort für 1 s stehen bleiben und dann soll der Bildschirminhalt wieder gelöscht werden.

1.7 Die Calliope soll sich automatisch bedanken, wenn sie etwas empfangen hat. Verschicke den Text „Danke“ nach dem Empfangen.

1.8 Vielleicht ist dir aufgefallen, dass, wenn ein Text versendet wurde, beim Versenden von „Danke“ eine Endlosschleife auftritt, d. h., die Calliope bedankt sich wieder und wieder. Das passiert, weil sich die Calliope bedankt, wenn sie einen Text erhält. „Danke“ ist allerdings auch ein Text, also bedankt sie sich dafür wieder. Das geht immer so weiter. Die Calliope soll sich nur noch einmal bedanken, wenn sie etwas empfangen hat. Sende „Danke“ *nur, wenn* die empfangene Nachricht NICHT „Danke“ ist. Um dies zu realisieren, muss die empfangene Nachricht mit dem Text „Danke“ verglichen werden und **ungleich** sein.

Worin ist noch gleich die empfangene Nachricht gespeichert? Verwende die entsprechende Variable für den Vergleich.

2 „Schere, Stein, Papier“ mit Funk

Zwei Calliopes sollen gegeneinander „Schere, Stein, Papier“ spielen und ihre eigenen Punkte mitzählen, indem sie über Funk miteinander kommunizieren.

2.1 Programmiere zunächst wieder ein einfaches „Schere, Stein, Papier“, wie du es bereits kennst: Wenn die Calliope geschüttelt wird, soll eines der drei Symbole Schere, Stein oder Papier angezeigt werden. Nenne die Variable, die Schere, Stein oder Papier zufällig wählt, *Symbol*.

2.2 Nach dem Schütteln soll das Symbol, das als Zahl in der Variablen *Symbol* gespeichert ist, über Funk versendet werden. Vergiss nicht, beim Start die Funkgruppe einzustellen.

2.3 Erstelle eine weitere Variable, in der du deine Punkte speichern wirst. Nenne diese *MeinePunkte*. Vergiss auch hier nicht, die Variable zu initialisieren.

2.4 Jetzt wird es ein bisschen kniffelig. Die Punkte des Gewinners eines Durchgangs sollen erhöht werden. Dabei zählt jede Calliope nur die eigenen Punkte. Was muss also passieren, wenn du eine Zahl empfangen hast? Überlege dir zunächst, in welchen Fällen deine Punkte erhöht werden.
Hinweis 1: Du musst die von dir und deinem Gegner geschüttelten Symbole miteinander vergleichen.
Hinweis 2: Dein Symbol ist in der Variablen *Symbol* und das deines Gegners in der Variablen *receivedNumber* gespeichert.

2.5 Wenn Taste A gedrückt wird, sollen dir deine Punkte für eine Sekunde angezeigt werden und dann wieder verschwinden.

2.6 Vielleicht ist dir aufgefallen, dass es beim Auswerten der Punkte zu Ungenauigkeiten kommen kann. Das liegt an Folgendem: Wenn die Calliope die Nummer des Symbols deines Gegners bekommt, bevor dein Symbol entschieden wurde, wird das gegnerische Symbol mit deinem vorherigen Symbol verglichen.

Um dies zu vermeiden, hast du zwei Möglichkeiten: ENTWEDER wartest du nach dem Empfangen wenige Sekunden in der Hoffnung, dass in der Zeit auch dein Symbol zufällig ausgewählt worden ist, ODER du baust einen Schalter in Form einer Variablen ein, der auf *wahr* springt, wenn dein neues Symbol feststeht, und auf *falsch* nach dem Auswerten. *Während* der Schalter noch auf *falsch* steht, wartet die Calliope mit der Auswertung. Die Auswertung findet damit immer nur statt, wenn deine Variable auf *wahr* steht und somit deine Calliope bereit ist. Letzteres ist die elegantere und sicherere Lösung.

Eine **Funktion** ist ein Programmkonstrukt, das Funktionalitäten eines Programms wiederverwendbar macht. Funktionen sind somit dann wertvoll, wenn du denselben Programmcode an verschiedenen Stellen verwenden möchtest. Sie ermöglichen, dass du eine Befehlsfolge nur einmal schreiben musst und dann immer wieder unter dem Namen der Funktion aufrufen und damit verwenden kannst.
Das macht dein Programm kürzer, übersichtlicher und weniger aufwendig in der Programmierung.

1 Ampelschaltung

1.1 Die Ampel soll zunächst auf Grün sein. Realisiere dies, indem du die RGB-LED bei Starten des Programms grün leuchten lässt.

1.2 Wenn ein Fußgänger über die Straße gehen möchte, soll er auf einen Ampelknopf (Taste A) drücken können, sodass die Ampel auf Rot springt und 3 Sekunden lang rot bleibt. Anschließend wird die Ampel wieder grün. Vergiss nicht die Gelbphasen zwischen Grün und Rot bzw. zwischen Rot und Grün, die 1 Sekunde lang sein sollen.
Zur Erinnerung: 1 Sekunde (s) = 1000 Millisekunden (ms).

1.3 Das Ampelsystem soll auch erkennen, wenn ein anderes Auto an der Kreuzung steht. Erkannt wird dies durch eine Kontaktschleife, über die das Auto fährt (berühren von Pin P0). Genauso wie bei den Fußgängern wird die Ampel auch in diesem Fall rot.
Hinweis: Der Inhalt von „wenn Pin P0 gedrückt“ wird ausgeführt, wenn Pin P0 berührt und wieder losgelassen wurde.

1.4 Wie du unschwer erkennen kannst, ist der Inhalt von „wenn Taste A gedrückt“ und „wenn Pin P0 gedrückt“ der gleiche. Hier kommen jetzt Funktionen ins Spiel. Klicke auf „Erstelle eine Funktion…“.

Du findest dies in der Kategorie Fortgeschritten unter Funktionen. Es wird sich ein Fenster, wie hier abgebildet, zum Bearbeiten der Funktion öffnen. Ersetze „macheEtwas“ durch den Namen deiner Funktion, die du *Ampelschaltung* nennst. Klicke anschließend auf „Fertig“. Deine erstellte Funktion erscheint als Block in deinem Programmcode.

Mit Parametern wird zu einem späteren Zeitpunkt gearbeitet. Diese sind noch uninteressant für dich.

1.5 Aktuell ist die Funktion *Ampelschaltung* noch leer. Ziehe die Blöcke aus „wenn Taste A gedrückt“ in die Funktion.

1.6 Rufe anstelle der Befehlsfolgen aus 1.2 und 1.3 die Funktion *Ampelschaltung* auf.

Wie du feststellen kannst, tut dein Programm nach wie vor das Gleiche, der Code besteht aber aus weniger Blöcken. Noch nützlicher wird das Verwenden der Funktion, wenn noch mehr Szenarien hinzukommen, bei denen die Ampel auf Rot schalten soll.

Info: Soll eine Funktion individuelle Abweichungen haben, kann man ihr sogenannte Parameter übergeben, die dann innerhalb der Funktion verwendet werden können. Parameter fungieren im Grunde genauso wie Variablen.

Wir wollen nun, dass die Ampel bei einem Fußgänger länger rot bleibt als bei einem Auto.

1.7 Bearbeite deine Funktion (durch Rechtsklick auf diese) so, dass du ihr eine Zahl als Parameter hinzufügst. Dieser Parameter heißt standardmäßig *num*. Benenne diesen in *Wartezeit* um. Wenn du auf „Fertig“ geklickt hast, hat deine Funktion nun den Parameter *Wartezeit* neben ihrem Namen. Außerdem ist beim Aufruf der Funktion ein Feld mit der Zahl 1 hinzugekommen. Diese Zahl wird an die Funktion weitergegeben. Aktuell ist also *Wartezeit* = *1*.

1.8 Anstelle von 3 Sekunden soll die Ampel für Fußgänger 5 Sekunden und für Autofahrer 4 Sekunden rot bleiben.
Hinweis: Übergebe den entsprechenden Wert an die Funktion.

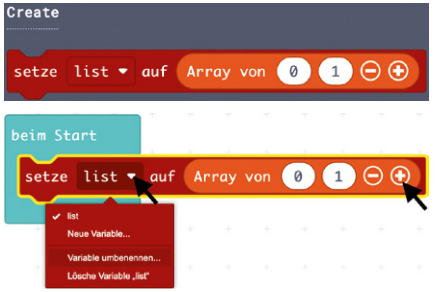
1.9 Die Pause nach der roten Ampel soll je nach Auto oder Fußgänger entschieden werden. Die jeweilige Zeit ist in *Wartezeit* gespeichert. Ziehe *Wartezeit* einfach an die entsprechende Stelle.

Ein **Array** ist eine Struktur, mit der viele gleichartige Daten verarbeitet werden können. Im Grunde kannst du dir ein Array wie eine Variable vorstellen, die Platzhalter für mehr als nur einen Wert ist. Die einzige Bedingung ist, dass die Werte, die in einem Array gespeichert sind, den gleichen Typ haben. Das bedeutet, in einem Array stehen entweder nur Zahlen oder nur Texte, nie beides gemischt. Sieh ein Array als eine Tabelle, in der du in jeder Zeile einen Wert speichern kannst. Die Zeilen sind dabei durchnummeriert, sodass du durch die Nummer der Zeile auf den entsprechenden Wert zugreifen kannst. Wie in der Informatik üblich, beginnt die Nummerierung der Zeilen mit 0.

2 Abstimmungstool

Die Klassensprecherwahl steht an. Es stehen vier mögliche Kandidaten und Kandidatinnen zur Wahl. Die Stimmen sollen geheim abgegeben und von der Calliope gezählt werden.

2.1 Erstelle beim Start des Programms ein neues Array. Da die Anzahl der Stimmen gezählt werden soll, benötigst du ein Array, das Zahlen speichert. Den entsprechenden Block findest du in der Kategorie Fortgeschritten unter Arrays. Standardmäßig heißt dieses Array *list* und besteht aus zwei Feldern. Eine sinnvolle Benennung von Variablen ist wichtig für ein nachvollziehbares Programm. Nenne dein Array *Stimmen* und füge zwei weitere Felder hinzu.
Außerdem haben am Anfang alle Kandidierenden 0 Punkte.



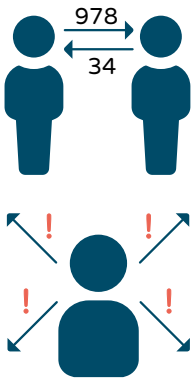
2.2 Bei Berühren von Pin 0 soll Kandidat 1 eine Stimme erhalten, bei Berühren von Pin 1 Kandidat 2 usw.
Am Beispiel von Kandidat 1: Wenn Pin 0 gedrückt wird, bekommt Kandidat 1 eine Stimme. Damit muss die erste Zeile des Arrays *Stimmen* um 1 erhöht werden. Welche Nummerierung hat noch gleich die erste Zeile eines Arrays?
Die folgenden Blöcke benötigst du für die Realisierung.

2.3 Ist die Abstimmung vorbei, soll das Ergebnis mit Drücken von Taste A angezeigt werden.
Hinweis 1: Rufe die Werte der Felder des Arrays *Stimmen* jeweils ab und gebe sie als Text aus.
Hinweis 2: Du kannst mehrere Texte aneinanderhängen, indem du den Block „verbinde“ aus der Kategorie Text verwendest. Diese findest du unter der Kategorie Fortgeschritten.

Corona-Warn-App

Corona ist seit Anfang 2020 das wohl präsenteste Thema weltweit. Um Infektionsketten nachverfolgen und im besten Falle unterbrechen zu können, wurde die Corona-Warn-App entwickelt, die dabei helfen soll. Die App soll Nutzer informieren, wenn sie in Kontakt mit einer infizierten Person geraten sind.

Das Prinzip



Teil 1: Wenn zwei Personen (mit Calliopes) nah beieinander sind, tauschen diese Calliopes ihre (festen, aber zu Beginn zufällig generierten) Nummern aus und speichern die Nummer des anderen.
Teil 2: Ist eine Person nun infiziert, kann sie dies anderen kundtun, indem sie die Nummer ihrer Calliope an alle anderen Calliopes schickt. Diese überprüfen, ob diese Nummer in ihrem Speicher vorkommt. Wenn ja, wird eine Warnung ausgegeben.
In der Realität geschieht Teil 1 über Bluetooth und Teil 2 über das Internet. Da der Zugriff mit Calliopes auf das Internet nicht möglich ist, wird für beides der gegebene Funk verwendet.
Für Teil 1 wird dazu die Sendeleistung auf 1 (Minimum = geringe Reichweite wie Bluetooth, bis ca. 40 cm) gestellt, für Teil 2 auf 7 (Maximum = hohe Reichweite ähnlich dem Internet, bis ca. 70 m).

1 Die Funkgruppe

Damit alle Calliopes miteinander Daten austauschen können, setze die Funkgruppe auf 255.

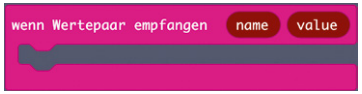
2 Nummern-Austausch bei Begegnung

Zunächst soll Teil 1 des Prinzips der Corona-Warn-App programmiert werden. Dazu muss bei Start des Programms die Sendeleistung eingestellt und der Calliope eine zufällige Nummer als ID zugeteilt werden. Bei Begegnung und Austausch zweier Calliopes wird lediglich diese Zahl verschickt, um die Identität des Nutzers zu schützen.
Außerdem muss eine leere Liste angelegt werden, in der später alle Kontaktpersonen als Nummern gespeichert werden.
Damit die Calliopes ihre Kontaktpersonen speichern können, müssen zwei Dinge passieren: Zum einen muss die Calliope selbst ihre eigene Nummer dauerhaft versenden. Zum anderen muss beim Empfangen einer Zahl geprüft werden, ob diese Zahl bereits in der Liste eingetragen ist; falls nicht, muss sie eingetragen werden.

Hinweis: *finde Position* von liefert -1, falls das Element nicht in der Liste gefunden wurde.

3 Warnung bei Corona-Kontakt

Um bei Erkrankung auch alle Kontaktpersonen warnen zu können, soll nun Teil 2 des Prinzips der Corona-Warn-App realisiert werden.
Ist ein Nutzer positiv auf Corona getestet worden, kann er dies seinen Kontaktpersonen mitteilen, indem er Taste A drückt.
Dadurch wird die Sendeleistung entsprechend erhöht und ein Wertepaar „Achtung: Corona!“ und die eigene ID versendet. Anschließend kann die Sendeleistung wieder zurück auf den Normalzustand.
Bislang reagieren die Calliopes nur auf das Empfangen einer Zahl. Wird nun ein Wertepaar empfangen, muss in der Liste der Kontaktpersonen geprüft werden, ob die empfangene ID in der Liste steht. Ist dies der Fall, gab es einen Kontakt mit der erkrankten Person und die empfangene Nachricht soll ausgegeben werden.



Hinweis: In *Name* steht die Nachricht, in Value die *ID* des Infizierten.

Mögliche Erweiterungen:

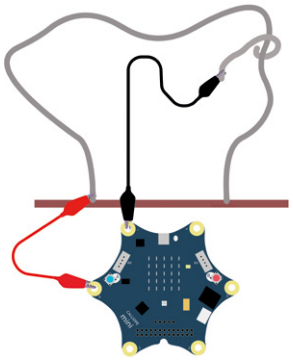
- Bei Eingabe / Start: Eigene ID anzeigen.
- Bei Eingabe: Bisherige Begegnungen anzeigen.
- Funktionsweise durch verschiedene LEDs veranschaulichen.
- Risiko durch RGB-LED anzeigen (grün = kein Kontakt, rot = Kontakt).
- Weitere Liste mit Zeitpunkt der Begegnung machen, sodass diese nach gewisser Zeit gelöscht werden kann.
- Nummer erst aufnehmen, wenn Kontakt länger als 10 Sekunden bestand.
- „Spion-Calliope“ bauen, die schaut, ob andere die Corona-Warn-App verwenden.

Heißer Draht

Der heiße Draht ist ein Geschicklichkeitsspiel mit dem Ziel, eine Drahtschlaufe über einen gebogenen Draht zu führen, ohne dass sich Schlaufe und Draht berühren. Dabei sind Draht und Drahtschlaufe jeweils an eine Stromquelle angeschlossen und bilden einen unterbrochenen Stromkreis. Berühren sich Draht und Schlaufe, so schließt sich der Stromkreis und der Fehler wird erkannt.

1 Der Aufbau

Baue dir deinen eigenen heißen Draht ähnlich wie auf der Skizze. Befestige dazu die beiden Enden deines Drahtes an einem Stück Pappe, sodass der Draht aufrecht steht. Biege den Draht so zurecht, wie du deinen heißen Draht gerne hättest.
Je mehr Kurven du einbaust, desto schwieriger wird dein heißer Draht für den Spieler. Verbinde ein Ende des Drahtes mit Pin 0 deiner Calliope. Biege dir zuletzt eine Drahtschlaufe zurecht und verbinde diese mit dem Minus-Pin auf deiner Calliope.



2 Fehler erkennen

2.1 Wenn der heiße Draht von der Schlaufe berührt wird, soll ein X auf dem LED-Bildschirm angezeigt werden. Denke daran, dass andernfalls der Bildschirm leer sein soll.
Hinweis 1: Dass der Draht berührt wird, erkennt man, indem Pin 0 gedrückt ist.
Hinweis 2: Der Inhalt des Blocks „wenn Pin P0 gedrückt“ wird ausgeführt, wenn Pin 0 berührt und wieder losgelassen wurde.
Der Block „Pin P0 ist gedrückt“ ruft dagegen den Pinzustand ab, d. h., es wird geprüft, ob der Pin zum Zeitpunkt der Abfrage berührt wird oder nicht. Deshalb ist es sinnvoll, diesen Block mit einer Verzweigung zu verwenden und den Zustand *dauerhaft* zu prüfen.

2.2 Während des Spielens guckt man konzentriert auf den Draht, deshalb soll das Berühren auch durch einen Ton signalisiert werden.
Hinweis: Achte darauf, dass du immer wieder kurze Töne spielst, *während* der Draht berührt wird. Spielst du lange Töne ab, so wird der Ton zu Ende gespielt, auch wenn in der Zwischenzeit der Draht schon nicht mehr berührt wird.

3 Fehlerpunkte

3.1 Das Berühren des heißen Drahtes soll zu Fehlerpunkten führen. Um diese zu zählen, musst du zunächst eine Variable *Fehlerpunkte* erstellen.

3.2 Sobald der Draht berührt wird, sollen die Fehlerpunkte um eins erhöht werden. Überlege dir selbst, ob die Punkte bei Berührung einmalig erhöht werden, unabhängig davon, wie lange der Draht berührt wird, oder ob sie erhöht werden, während der Signalton erklingt.

3.3 Wenn Taste B gedrückt wird, sollen die Fehlerpunkte angezeigt werden.

3.4 Wenn Taste A gedrückt wird, soll ein neues Spiel starten, d. h., die Fehlerpunkte werden zurückgesetzt.

4 Zeit stoppen

Zusätzlich zum Zählen der Fehlerpunkte soll gestoppt werden, wie lange von Start zu Ziel gebraucht wurde. Um diese Zeit zu erhalten, kannst du mit der Laufzeit arbeiten, die die Calliope standardmäßig misst.
Ziehe dazu die Laufzeit zu Beginn eines Durchgangs von der Laufzeit zu Ende des Durchgangs ab.

Info: Die Laufzeit ist, wie der Name bereits vermuten lässt, die Zeit, die ein Programm läuft, d. h. der Zeitraum vom Starten bis zum Beenden eines Programms. Bei den Programmen auf deiner Calliope beginnt die Laufzeit von 0, sobald du sie einschaltest, ein Programm draufspielst oder die Reset-Taste betätigst.

4.1 Erstelle zunächst drei Variablen: *Startzeit*, *Endzeit* und *Spielzeit*.

4.2 Wenn Taste A gedrückt wird, soll ein Durchgang starten. Speichere die Laufzeit zu diesem Zeitpunkt in der Variablen *Startzeit*.

4.3 Ist ein Durchgang zu Ende, soll Taste B gedrückt werden. Nach dem Drücken von Taste B sollen sowohl Fehlerpunkte als auch die benötigte Zeit angezeigt werden.
Hinweis 1: Zunächst musst du die *Endzeit* festlegen. Dies machst du, indem du die Laufzeit zum Zeitpunkt des Drückens von Taste B speicherst. Die *Spielzeit* errechnest du, indem du die *Startzeit* von der *Endzeit* abziehst.
Hinweis 2: Du kannst mehrere Texte aneinanderhängen, indem du den Block „verbinde“ aus der Kategorie Text verwendest. Diese findest du unter der Kategorie Fortgeschritten.

4.4 Die Laufzeit wird in Millisekunden angegeben. Die benötigte Spielzeit soll aber in Sekunden angezeigt werden. Wandle die *Zeit* in eine Sekundenangabe um.

Zur Erinnerung: 1 Sekunde (s) = 1000 Millisekunden (ms).

5 Spiel auf Spielzeit begrenzen

Bislang läuft die Erkennung, ob der Draht berührt, wird dauerhaft. Dies soll nun so geändert werden, dass die Überprüfung nur noch zwischen dem Drücken von Start (Taste A) und Ende (Taste B) erfolgt.

5.1 Erstelle zunächst eine Variable *Spiel*, die du mit *falsch* initialisierst, denn es ist kein Spiel gestartet.

5.2 Wird Taste A gedrückt, wird das Spiel gestartet (Spiel = wahr) und bei Drücken von Taste B wieder beendet (Spiel = falsch).

5.3 *Während* das Spiel läuft, muss geprüft werden, ob der Draht berührt wird.
Hinweis: Der Programmteil, der bislang dauerhaft gelaufen ist, wird nun benötigt, wenn ein Spiel gestartet wurde bzw. *während* ein Spiel läuft.

5.4 Deine Calliope reagiert nicht direkt auf das Drücken von Taste B? Das liegt daran, dass es zu einer kurzen Verzögerungszeit im Programm kommen kann. Um das Problem zu beheben, reicht schon eine Millisekunde Pause nach jeder Überprüfung, ob der Draht berührt ist.

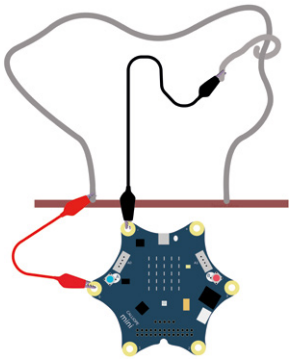
5.5 Wenn du möchtest, kannst du die aktive Spielzeit noch durch Leuchten der RGB-LED anzeigen lassen. Lasse diese beispielsweise grün leuchten, wenn ein Spiel läuft. Ist kein Spiel gestartet, leuchtet sie rot.

Heißer Draht

Der heiße Draht ist ein Geschicklichkeitsspiel mit dem Ziel, eine Drahtschlaufe über einen gebogenen Draht zu führen, ohne dass sich Schlaufe und Draht berühren. Dabei sind Draht und Drahtschlaufe jeweils an eine Stromquelle angeschlossen und bilden einen unterbrochenen Stromkreis. Berühren sich Draht und Schlaufe, so schließt sich der Stromkreis und der Fehler wird erkannt.

1 Der Aufbau

Baue dir deinen eigenen heißen Draht ähnlich wie auf der Skizze. Befestige dazu die beiden Enden deines Drahtes an einem Stück Pappe, sodass der Draht aufrecht steht. Biege den Draht so zurecht, wie du deinen heißen Draht gerne hättest.
Je mehr Kurven du einbaust, desto schwieriger wird dein heißer Draht für den Spieler. Verbinde ein Ende des Drahtes mit Pin 0 deiner Calliope. Biege dir zuletzt eine Drahtschlaufe zurecht und verbinde diese mit dem Minus-Pin auf deiner Calliope.



2 Fehler erkennen

2.1 Wenn der heiße Draht von der Schlaufe berührt wird, soll ein X auf dem LED-Bildschirm angezeigt werden. Denke daran, dass andernfalls der Bildschirm leer sein soll.
Hinweis 1: Dass der Draht berührt wird, erkennt man, indem Pin 0 gedrückt ist.
Hinweis 2: Der Inhalt des Blocks „wenn Pin P0 gedrückt“ wird ausgeführt, wenn Pin 0 berührt und wieder losgelassen wurde.
Der Block „Pin P0 ist gedrückt“ ruft dagegen den Pinzustand ab, d. h., es wird geprüft, ob der Pin zum Zeitpunkt der Abfrage berührt wird oder nicht. Deshalb ist es sinnvoll, diesen Block mit einer Verzweigung zu verwenden und den Zustand *dauerhaft* zu prüfen.

2.2 Während des Spielens guckt man konzentriert auf den Draht, deshalb soll das Berühren auch durch einen Ton signalisiert werden.
Hinweis: Achte darauf, dass du immer wieder kurze Töne spielst, *während* der Draht berührt wird. Spielst du lange Töne ab, so wird der Ton zu Ende gespielt, auch wenn in der Zwischenzeit der Draht schon nicht mehr berührt wird.

2.3. Lagere das Erkennen der Berührung des Drahtes in eine Funktion aus, die du *heißerDraht* nennst.

3 Fehlerpunkte

3.1 Das Berühren des heißen Drahtes soll zu Fehlerpunkten führen. Um diese zu zählen, musst du zunächst eine Variable *Fehlerpunkte* erstellen.

3.2 Sobald der Draht berührt wird, sollen die Fehlerpunkte um eins erhöht werden. Überlege dir selbst, ob die Punkte bei Berührung einmalig erhöht werden, unabhängig davon, wie lange der Draht berührt wird, oder ob sie erhöht werden, während der Signalton erklingt.

3.3 Wenn Taste B gedrückt wird, sollen die Fehlerpunkte angezeigt werden.

3.4 Wenn Taste A gedrückt wird, soll ein neues Spiel starten, d. h., die Fehlerpunkte werden zurückgesetzt.

4 Zeit stoppen

Zusätzlich zum Zählen der Fehlerpunkte soll gestoppt werden, wie lange von Start zu Ziel gebraucht wurde. Um diese Zeit zu erhalten, kannst du mit der Laufzeit arbeiten, die die Calliope standardmäßig misst.
Ziehe dazu die Laufzeit zu Beginn eines Durchgangs von der Laufzeit am Ende des Durchgangs ab.

Info: Die Laufzeit ist, wie der Name bereits vermuten lässt, die Zeit, die ein Programm läuft, d. h. der Zeitraum vom Starten bis zum Beenden eines Programms. Bei den Programmen auf deiner Calliope beginnt die Laufzeit von 0, sobald du sie einschaltest, ein Programm draufspielst oder die Reset-Taste betätigst.

4.1 Erstelle zunächst die drei Variablen *Startzeit*, *Endzeit* und *Zeit* und eine Funktion *Spielzeit*, in der die *Startzeit* von der Endzeit abgezogen und in *Zeit* gespeichert wird.

4.2 Wenn Taste A gedrückt wird, soll ein Durchgang starten. Speichere die Laufzeit zu diesem Zeitpunkt in der Variablen *Startzeit*.

4.3 Ist ein Durchgang zu Ende, soll der Taste B gedrückt werden. Nach dem Drücken von Taste B sollen sowohl Fehlerpunkte als auch die benötigte Zeit angezeigt werden.
Hinweis: Du musst zunächst die *Endzeit* festlegen und dann die *Zeit* mithilfe deiner Funktion *Spielzeit* berechnen lassen. Die Endzeit legst du fest, indem du die Laufzeit zum Zeitpunkt des Drückens von Taste B speicherst.

4.4 Die Laufzeit wird in Millisekunden angegeben. Die benötigte Spielzeit soll aber in Sekunden angezeigt werden. Wandle die Zeit in eine Sekundenangabe um.

Zur Erinnerung: 1 Sekunde (s) = 1000 Millisekunden (ms).

5 Spiel auf Spielzeit begrenzen

Bislang läuft die Funktion *heißerDraht* dauerhaft. Dies soll nun so geändert werden, dass sie nur noch zwischen dem Drücken von Start (Taste A) und Ende (Taste B) aufgerufen wird.

5.1 Erstelle zunächst eine Variable *Spiel*, die du mit *falsch* initialisierst.

5.2 Wird Taste A gedrückt, wird das Spiel gestartet (Spiel = wahr) und bei Drücken von Taste B wieder beendet (Spiel = falsch).

5.3 *Während* das Spiel läuft, muss die Funktion *heißerDraht* aufgerufen werden.

5.4 Deine Calliope reagiert nicht direkt auf das Drücken von Taste B? Das liegt daran, dass es zu einer kurzen Verzögerungszeit im Programm kommen kann. Um das Problem zu beheben, reicht schon eine Millisekunde Pause nach dem Aufruf der Funktion *heißerDraht*.

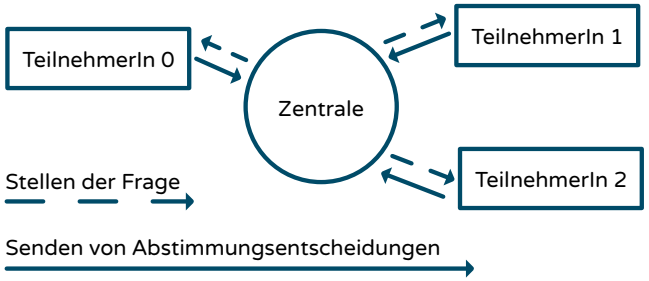
5.5 Wenn du möchtest, kannst du die aktive Spielzeit noch durch Leuchten der RGB-LED anzeigen lassen. Lasse diese beispielsweise grün leuchten, wenn ein Spiel läuft. Ist kein Spiel gestartet, leuchtet sie rot.

Abstimmungstool

Die Wahl zum/zur KlassensprecherIn, ein „Wer wird Millionär?“ mit eigenen Fragen oder eine schnelle Abfrage, ob der Nachmittagsunterricht verlegt werden soll – all das ist mit der Calliope möglich!

Das Prinzip

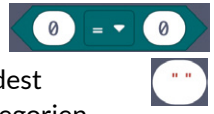
Die TeilnehmerInnen beantworten eine Frage, die Zentrale wertet die Antworten aus. **Erinnerung:** InformatikerInnen fangen immer bei 0 an zu zählen.



Findet euch in Gruppen zusammen, sodass **jede Gruppe zwei Calliopes** besitzt. Eine der Calliopes wird als Zentrale und die andere als TeilnehmerIn agieren, die Programmierung der verschiedenen TeilnehmerInnen unterscheidet sich nicht.

Zunächst sollen binäre Entscheidungen (Ja / Nein, an / aus …) getroffen werden.

1 Programmiere die Calliope für TeilnehmerInnen so, dass beim Drücken von Taste A ein „Ja“, beim Drücken von Taste B ein „Nein“ verschickt wird. Die Calliope in der Zentrale soll die empfangenen „Ja“- und „Nein“-Stimmen zählen und beim Drücken von Taste A die jeweiligen Anzahlen ausgeben. Anschließend sollen die Anzahlen zurückgesetzt werden.
Hinweis: Da Texte (strings) verschickt und empfangen werden, musst du die empfangene Antwort auch mit einem Text vergleichen. Den entsprechenden Block findest du in der Kategorie Text unter den Fortgeschritten-Kategorien.



Nun sollen auch mehrere Möglichkeiten (1 / 2 / 3 / 4, A / B / C / D) wählbar sein.

2 Erweitere deine Programme so, dass mit den Pins 0, 1, 2 und 3 vier verschiedene Antwortmöglichkeiten gewählt werden können. Versende dazu entsprechend die Zahlen 0–3. Da das Drücken von einem Pin nicht immer einwandfrei funktioniert, soll der / die TeilnehmerIn bei einem erfolgreichen Drücken des Pins eine Rückmeldung (z. B. LED, Text …) erhalten. Mit Druck auf Taste B bei der Zentrale können dort die Ergebnisse einer Frage mit vier Antwortmöglichkeiten ausgegeben werden. Nachdem das Ergebnis angezeigt wurde, werden alle Anzahlen zurückgesetzt.

3 Eine erneute Abstimmung soll erst möglich sein, wenn in der Zentrale die Ergebnisse angezeigt wurden (und damit die nächste Frage ansteht). Dass die Ergebnisse in der Zentrale ausgewertet wurden, teilt die Zentrale den TeilnehmerInnen mit, indem sie ein „OK“ versendet. Die LED der TeilnehmerInnen soll entsprechend grün/rot leuchten. In der Zentrale soll außerdem die Anzahl der bereits empfangenen Antworten angezeigt werden.
Hinweis 1: Da die Zentrale und alle TeilnehmerInnen in derselben Funkgruppe sind, können die TeilnehmerInnen auch die Abstimmungen der anderen TeilnehmerInnen empfangen. Prüfe deshalb, ob die empfangene Nachricht das „OK“ der Zentrale ist, bevor die TeilnehmerInnen wieder zur Abstimmung freigegeben werden.
Hinweis 2: Arbeite mit einer Variablen, die du entsprechend auf wahr oder falsch setzt, wenn abgestimmt werden darf oder eben nicht.

Bislang wurde nur die Abstimmung selbst über die Calliopes realisiert. Nun sollen aber auch die Fragen von der Calliope der Zentrale gestellt werden.

4 Die Fragen sollen im Vorhinein in der Zentrale in einer Liste gespeichert werden und die entsprechende Frage vor jeder Abstimmung über die Anzeige-Matrix in der Zentrale angezeigt werden.
Eine neue Frage soll angezeigt werden, wenn Taste A und B gleichzeitig gedrückt werden.
Da mit dem Drücken von Taste A und B eine neue Frage gestartet wird, ist es sinnvoll, die Freigabe zur Abstimmung der Teilnehmer durch das „OK“ hierhin zu verschieben.

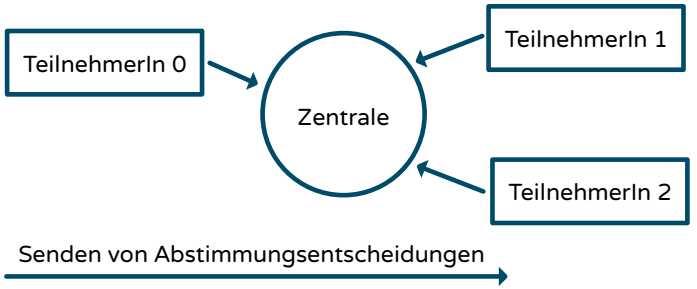
5 Die Frage soll nicht nur auf der LED-Matrix der Zentrale, sondern auch bei allen Teilnehmende angezeigt werden.
Hinweis 1: Du wirst nun zwei Texte hintereinander verschicken –„OK“ und die jeweilige Frage. Das bedeutet, dass der Inhalt von *received-String* bei den Teilnehmende direkt von dem als Zweites gesendeten Text überschrieben wird. Überlege dir deshalb, welchen Text du zuerst sendest. Welchen Text brauchst du nur kurz?
Hinweis 2: Achte auch hier wieder darauf, dass du nicht versehentlich auf die Antworten der anderen TeilnehmerInnen reagierst.
Wann sollen die anderen-Calliopes auf einen empfangenen Text reagieren?
Hinweis 3: Um Ausführungsfehler zu vermeiden, kannst du eine kurze Wartezeit zwischen den beiden Sendungen einfügen.

Abstimmungstool

Die Wahl zum/zur KlassensprecherIn, ein „Wer wird Millionär?“ mit eigenen Fragen oder eine schnelle Abfrage, ob der Nachmittagsunterricht verlegt werden soll – all das ist mit der Calliope möglich!

Das Prinzip

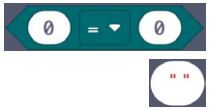
Die TeilnehmerInnen beantworten eine Frage, die Zentrale wertet die Antworten aus. **Erinnerung:** InformatikerInnen fangen immer bei 0 an zu zählen.



Findet euch in Gruppen zusammen, sodass **jede Gruppe zwei Calliopes** besitzt. Eine der Calliopse wird als Zentrale und die andere als TeilnehmerIn agieren, die Programmierung der verschiedenen TeilnehmerInnen unterscheidet sich nicht.

Zunächst sollen binäre Entscheidungen (Ja / Nein, an / aus …) getroffen werden.

1 Programmiere die Calliope für TeilnehmerInnen so, dass beim Drücken von Taste A ein „Ja“, beim Drücken von Taste B ein „Nein“ verschickt wird. Die Calliopes in der Zentrale soll die empfangenen „Ja“- und „Nein“-Stimmen zählen und beim Drücken von Taste A die jeweiligen Anzahlen ausgeben. Anschließend sollen die Anzahlen zurückgesetzt werden.
Hinweis 1: Da Texte (strings) verschickt und empfangen werden, musst du die empfangene Antwort auch mit einem Text vergleichen.



Den entsprechenden Block findest du in der Kategorie Text unter den Fortgeschritten-Kategorien.
Hinweis 2: Mithilfe des Blocks „verbinde“ kannst du mehrere Texte aneinanderhängen. Diesen findest du ebenfalls in der Kategorie Text.

Nun sollen auch mehrere Möglichkeiten (1/2/3/4, A/B/C/D) wählbar sein.

2 Erweitere deine Programme so, dass mit den Pins 0, 1, 2 und 3 vier verschiedene Antwortmöglichkeiten gewählt werden können. Versende dazu entsprechend die Zahlen 0–3. Da das Drücken von einem Pin nicht immer einwandfrei funktioniert, soll der / die TeilnehmerIn bei einem erfolgreichen Drücken des Pins eine Rückmeldung (z. B. LED, Text …) erhalten. Mit Druck auf Taste B bei der Zentrale können dort die Ergebnisse einer Frage mit vier Antwortmöglichkeiten ausgegeben werden. Nachdem das Ergebnis angezeigt wurde, werden alle Anzahlen zurückgesetzt.

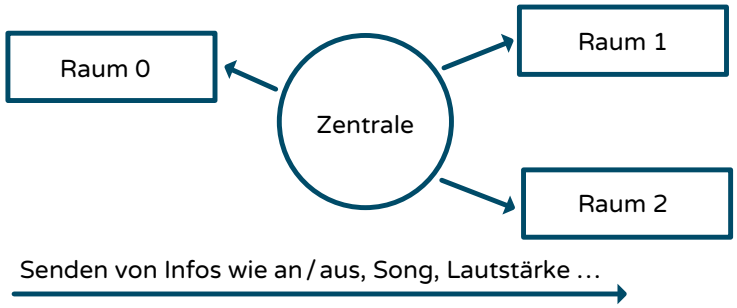
3 Eine erneute Abstimmung soll erst möglich sein, wenn in der Zentrale die Ergebnisse angezeigt wurden (und damit die nächste Frage ansteht). Dass die Ergebnisse in der Zentrale ausgewertet wurden, teilt die Zentrale den TeilnehmerInnen mit, indem sie ein „OK“ versendet. Die LED der TeilnehmerInnen soll entsprechend grün/rot leuchten. In der Zentrale soll außerdem die Anzahl der bereits empfangenen Antworten angezeigt werden.
Hinweis 1: Da die Zentrale und alle TeilnehmerInnen in derselben Funkgruppe sind, können die TeilnehmerInnen auch die Abstimmungen der anderen TeilnehmerInnen empfangen. Prüfe deshalb, ob die empfangene Nachricht das „OK“ der Zentrale ist, bevor die TeilnehmerInnen wieder zur Abstimmung freigegeben werden.
Hinweis 2: Arbeite mit einer Variablen, die du entsprechend auf wahr oder falsch setzt, wenn abgestimmt werden darf oder eben nicht.

Soundsystem

Von deinem Handy aus die Musik in deinem und jedem anderen Zimmer zuhause an- und ausschalten? In jedem Zimmer individuell die Musik, Lautstärke … zentral einstellen? Das können nur gute (und teure) Soundsysteme? Nein! Auch DU kannst das!

Das Prinzip

Die Zentrale steuert die Musik in den verschiedenen Räumen.
Erinnerung: InformatikerInnen fangen immer bei 0 an zu zählen.

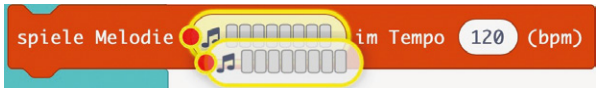


Findet euch in Gruppen zusammen, sodass **jede Gruppe zwei Calliopes** besitzt. Eine Calliope wird als Zentrale und die andere als Raum agieren, die Programmierung der verschiedenen Räume unterscheidet sich kaum.

1 Programmiere deine Calliope für die Zentrale so, dass beim Drücken von Pin 0 eine Nachricht mit „0“ verschickt wird. Die Calliope in Raum 0 soll beim Empfangen von einer „0“ eine Melodie spielen. Genauso würde Pin 1 für Raum 1 stehen und so weiter.

2 Nun soll zusätzlich die Information, welche Melodie gespielt werden soll, mitgeschickt werden. Nutze dafür das Senden / Empfangen von Wertepaaren mit „name“ und „value“. Mit den Tasten A und B soll zunächst die entsprechende Melodie ausgewählt und auf der Anzeige-Matrix angezeigt werden und durch das Drücken eines Pins die Information an den entsprechenden Raum verschickt werden.
Hinweis 1: Du benötigst eine Variable, die beim Drücken von Taste A hoch- und beim Drücken von Taste B runtergezählt wird. Abhängig vom Wert dieser Variablen werden entsprechend Melodien versendet.
Hinweis 2: Anstelle eines Texts / Strings kannst du auch eine Melodie versenden. Beim Empfänger steht diese dann in der Variablen „name“.

Da der Block für die Melodie nicht einzeln zur Verfügung steht, musst du ein bisschen tricksen. Baue zunächst den Block „Spiele Melodie … in Tempo … (bpm)“ in dein Programm ein, auch wenn du diesen in der Zentrale eigentlich nicht benötigst. Nun kannst du den Block zur Melodiebestimmung wie eine Variable an die gewünschte Stelle ziehen. Den Block „Spiele Melodie … in Tempo … (bpm)“ kannst du anschließend wieder löschen.



3 Bislang kann die Melodieauswahl in der Zentrale beliebig hoch- und runtergezählt werden. Wenn das Hochzählen die Anzahl deiner verschiedenen Melodien überschreiten würde, soll mit dem Zählen von vorne begonnen werden. Beschränke auch das Runterzählen.

Stoppuhr

Stoppuhren können Antworten auf viele Fragen liefern.
Wie lange kannst du die Luft anhalten? Wer kann am besten 10 Sekunden abschätzen? Wie schnell bist du auf 100 m?

Calliope-Stoppuhr

Um mit der Calliope die Zeit zu messen, gibt es zwei Möglichkeiten:

Variante 1: Die Calliope misst standardmäßig die Laufzeit eines Programms. Um die Zeit zwischen zwei Ereignissen zu messen, müssen die Laufzeiten vor dem ersten Ereignis und nach dem zweiten Ereignis verglichen werden.

Info: Die Laufzeit ist, wie der Name bereits vermuten lässt, die Zeit, die ein Programm läuft, d. h. der Zeitraum vom Starten bis zum Beenden eines Programms. Bei den Programmen auf deiner Calliope beginnt die Laufzeit bei 0, sobald du sie einschaltest, ein Programm draufspielst oder die Reset-Taste betätigst.

Variante 2: Die Calliope kann in regelmäßigen Zeitintervallen nach bestimmten Ereignissen schauen und, während sie auf ein entsprechendes Ereignis wartet, die vergangene Zeit mitzählen, indem sie die Zeit um den Wert der Wartezeit erhöht. Die Wartezeiten dürfen dabei nicht zu hoch sein, damit kein Ereignis verpasst wird.

Erinnerung: 1 Sekunde (s) = 1000 Millisekunden (ms).

1 Die Stoppuhr soll mit einem Druck auf Taste A beginnen und mit einem Druck auf Taste B enden. Anschließend soll die gemessene Zeit in Sekunden auf der Anzeigematrix ausgegeben werden.
Zusätzlich soll die RGB-LED während der Zeitmessung grün leuchten, ansonsten soll die LED Rot anzeigen.

Hinweis zu Variante 1: Wenn Taste A gedrückt wird, speichere die Laufzeit in einer Variablen. Wurde das Stoppen der Zeit beendet, bilde die Differenz aus der Laufzeit zu Beginn und der am Ende.
Hinweis zu Variante 2: Wenn Taste A gedrückt wurde, soll, während Taste B nicht gedrückt wird, eine kurze Zeit pausiert und gleichzeitig eine Variable um diesen Zeitwert hochgezählt werden.

Wird bei einem Rennen der Startschuss beim Start gegeben, kommt dieser zeitverzögert im Zielbereich, wo die Zeit gestoppt werden soll, an (Schallgeschwindigkeit: 340 m/s). Daher soll der Start zusätzlich per Funk dem Ziel gemeldet werden (Lichtgeschwindigkeit: 300.000 km/s).

Findet euch in Gruppen zusammen, sodass **jede Gruppe zwei Calliopes** besitzt.

2 Die erste Calliope soll mit Druck auf Taste A ein Startsignal geben und dann per Funk der zweiten Calliope mitteilen, dass der Start erfolgt ist. Bei Empfangen der Startmitteilung soll die Stoppuhr auf der zweiten Calliope gestartet werden. Beendet wird die Zeitmessung durch Drücken von Taste A auf der Ziel-Calliope. Damit auch erkannt wird, dass die Zeitmessung läuft, soll in dieser Zeit die RGB-LED der zweiten Calliope grün leuchten.

Stoppuhr

Stoppuhren können Antworten auf viele Fragen liefern.
Wie lange kannst du die Luft anhalten? Wer kann am besten 10 Sekunden abschätzen? Wie schnell bist du auf 100 m?

Calliope-Stoppuhr

Um mit der Calliope die Zeit zu messen, gibt es zwei Möglichkeiten:

Variante 1: Die Calliope misst standardmäßig die Laufzeit eines Programms. Um die Zeit zwischen zwei Ereignissen zu messen, müssen die Laufzeiten vor dem ersten Ereignis und nach dem zweiten Ereignis verglichen werden.

Info: Die Laufzeit ist, wie der Name bereits vermuten lässt, die Zeit, die ein Programm läuft, d. h. der Zeitraum vom Starten bis zum Beenden eines Programms. Bei den Programmen auf deiner Calliope beginnt die Laufzeit bei 0, sobald du sie einschaltest, ein Programm draufspielst oder die Reset-Taste betätigst.

Variante 2: Die Calliope kann in regelmäßigen Zeitintervallen nach bestimmten Ereignissen schauen und, während sie auf ein entsprechendes Ereignis wartet, die vergangene Zeit mitzählen, indem sie die Zeit um den Wert der Wartezeit erhöht. Die Wartezeiten dürfen dabei nicht zu hoch sein, damit kein Ereignis verpasst wird.

Erinnerung: 1 Sekunde (s) = 1000 Millisekunden (ms).

1 Die Stoppuhr soll mit einem Druck auf Taste A beginnen und mit einem Druck auf Taste B enden. Anschließend soll die gemessene Zeit in Sekunden auf der Anzeigematrix ausgegeben werden.
Zusätzlich soll die RGB-LED während der Zeitmessung grün leuchten, ansonsten soll die LED Rot anzeigen.

Hinweis zu Variante 1: Wenn Taste A gedrückt wird, speichere die Laufzeit in einer Variablen. Wurde das Stoppen der Zeit beendet, bilde die Differenz aus der Laufzeit zu Beginn und der am Ende.
Hinweis zu Variante 2: Wenn Taste A gedrückt wurde, soll, während Taste B nicht gedrückt wird, eine kurze Zeit pausiert und gleichzeitig eine Variable um diesen Zeitwert hochgezählt werden.

Wird bei einem Rennen der Startschuss beim Start gegeben, kommt dieser zeitverzögert im Zielbereich, wo die Zeit gestoppt werden soll, an (Schallgeschwindigkeit: 340 m/s). Daher soll der Start zusätzlich per Funk dem Ziel gemeldet werden (Lichtgeschwindigkeit: 300.000 km/s).

Findet euch in Gruppen zusammen, sodass **jede Gruppe zwei Calliopes** besitzt.

2 Die erste Calliope soll mit Druck auf Taste A ein Startsignal geben und dann per Funk der zweiten Calliope mitteilen, dass der Start erfolgt ist. Bei Empfangen der Startmitteilung soll die Stoppuhr auf der zweiten Calliope gestartet werden. Beendet wird die Zeitmessung durch Drücken von Taste A auf der Ziel-Calliope. Damit auch erkannt wird, dass die Zeitmessung läuft, soll in dieser Zeit die RGB-LED der zweiten Calliope grün leuchten.

3 Bei einem Rennen starten üblicherweise mehrere LäuferInnen. Erweitere dein Programm aus Aufgabe 2 so, dass jedes Mal, wenn jemand das Ziel erreicht, dessen Zeit durch Drücken von Taste A auf der Ziel-Calliope gespeichert wird. Sind alle LäuferInnen im Ziel angekommen, ist das Rennen vorbei und alle Zeiten können durch Drücken von Taste B angezeigt werden.
Hinweis: Verwende ein Array zum Speichern aller Zeiten.