

# Willkommen im



# Calliope Workshop



## Vorstellung

Wir, der Oberlab e.V. sind ein eingetragener, gemeinnütziger Verein und eine frei zugängliche Forschungswerkstatt, die Hightech-Geräte für Bastler, Technik-Interessierte und Unternehmen bereitstellt. Wir sind ein Maker Space, der Jung und Alt für Wissenschaft, Technologie und Digital Fabrication begeistern will. Hier ist Platz für Design, Prototyping und Experimente.

Kurz: Das OberLab ist ein offener Technik-Spielplatz für kleine und große Tüftler!

# Calliope Workshop

## Unsere Ausstattung

- Lasercutter und Schneidplotter
- 3D-Drucker
- Fachbereiche Software, Textil,
- Holz- und Elektrotechnik
- Software- und Coding-Tools
- Experimentier-Labor
- Mehr Infos unter [www.oberlab.de](http://www.oberlab.de)

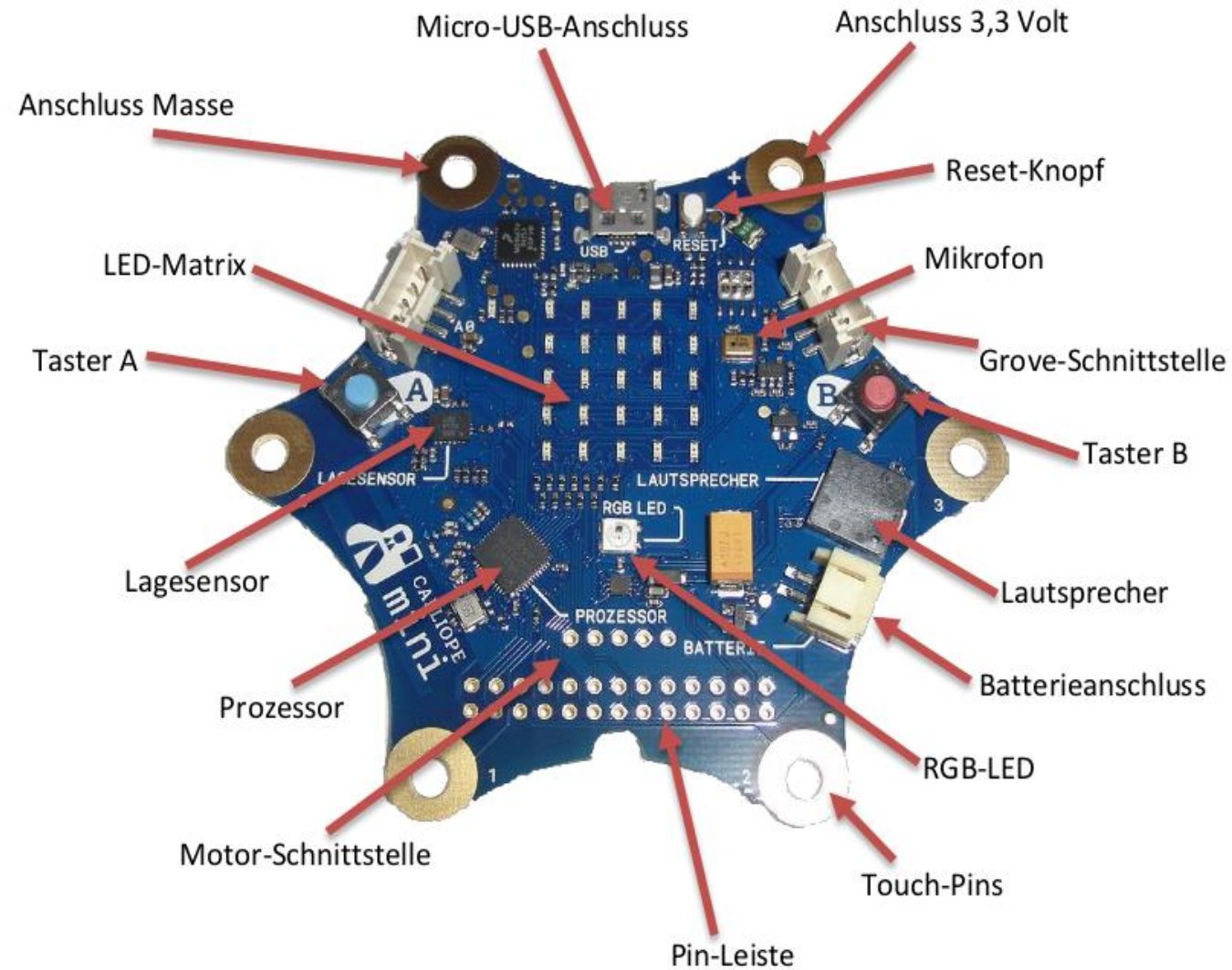


## Hygiene-Regeln

- Abstand halten, direkten Kontakt vermeiden.
- Niesen oder Husten in die Armbeuge.
- Vor und nach dem Toilettengang die Hände waschen und desinfizieren. Mund und Nasenschutz verwenden.
- Die Hygiene-Regeln auch in den Pausen befolgen.
- Hygiene Mittel stehen kostenlos zur Nutzung bereit.

# Calliope Workshop

## Die Hardware




## Mein erstes Programm - Online-Editor:

<https://makecode.calliope.cc/#editor>

# Calliope Workshop

## Mein erstes Programm – Taste A zeigt uns ein Herz

Blöcke	JAVA-Script	Python
	<pre>input.onButtonPressed(Button.A, function () {   basic.showLeds(`     . # . # .     # . # . #     . # . # .     .. # ..     .....   `) }) input.onButtonPressed(Button.B, function () {   basic.showLeds(`     .....     .....     .....     .....     .....   `) })</pre>	<pre>def on_button_pressed_a():   basic.show_leds("""     . # . # .     # . # . #     . # . # .     .. # ..     .....     """) input.on_button_pressed(Button.A, on_button_pressed_a)  def on_button_pressed_b():   basic.show_leds("""     .....     .....     .....     .....     .....     """) input.on_button_pressed(Button.B, on_button_pressed_b)</pre>

# Calliope Workshop



## Java



Java ist eine Programmiersprache, die derzeit für viele Geräte gebraucht wird. Was genau dahinter steckt, verraten wir Ihnen nachfolgend:

- Java ist eine Programmiersprache und zugleich eine sogenannte Laufzeitumgebung.
- Eine Laufzeitumgebung ist ein kleiner Teil einer Software, der dafür sorgt, Programme kompakt zu halten.
- Dadurch wird weniger Speicher bei gleicher Leistung benötigt.
- Insbesondere bei komplexen Vorgehen auf Webseiten, wie beispielsweise einer Buchung, wird Java im Hintergrund aktiv und arbeitet diverse Prozesse ab.
- Java gibt es allerdings auch in Browsern. Bevor andere Programme wie Flash oder JavaScript zum Einsatz kamen, war Java die am weitesten verbreitete Software, um komplexe Inhalte wie Animationen oder Grafiken auf einer Webseite darzustellen.
- Java wird neben dem Einsatz auf PCs auch für Anwendungen auf Smartphone und Tablets, Smart-TVs, Servern oder Spielekonsolen verwendet.



# Calliope Workshop



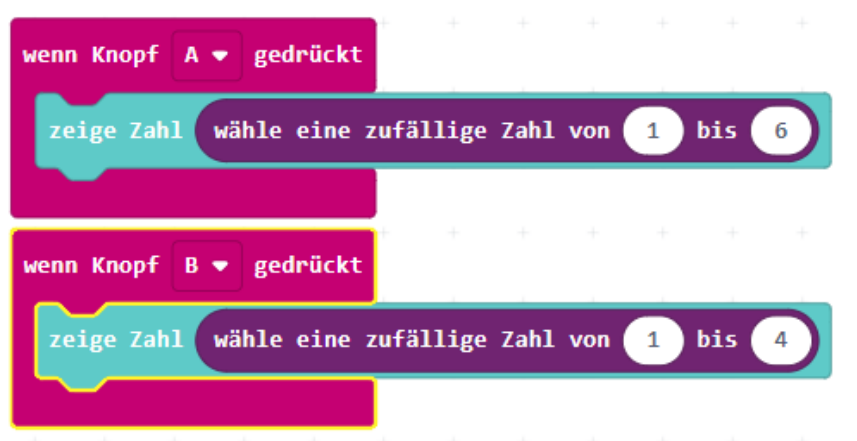
**Python** ist eine universelle, üblicherweise interpretierte, höhere Programmiersprache

- Sie hat den Anspruch, einen gut lesbaren, knappen Programmierstil zu fördern.
- So werden beispielsweise Blöcke nicht durch geschweifte Klammern, sondern durch Einrückungen strukturiert.
- Python unterstützt mehrere Programmierparadigmen, z. B. die objektorientierte, die aspektorientierte und die funktionale Programmierung.
- Ferner bietet es eine dynamische Typisierung. Wie viele dynamische Sprachen wird Python oft als Skriptsprache genutzt.
- Die Sprache weist ein offenes, gemeinschaftsbasiertes Entwicklungsmodell auf, das durch die gemeinnützige Python Software Foundation gestützt wird, die de facto die Definition der Sprache in der Referenzumsetzung CPython pflegt.

# Calliope Workshop



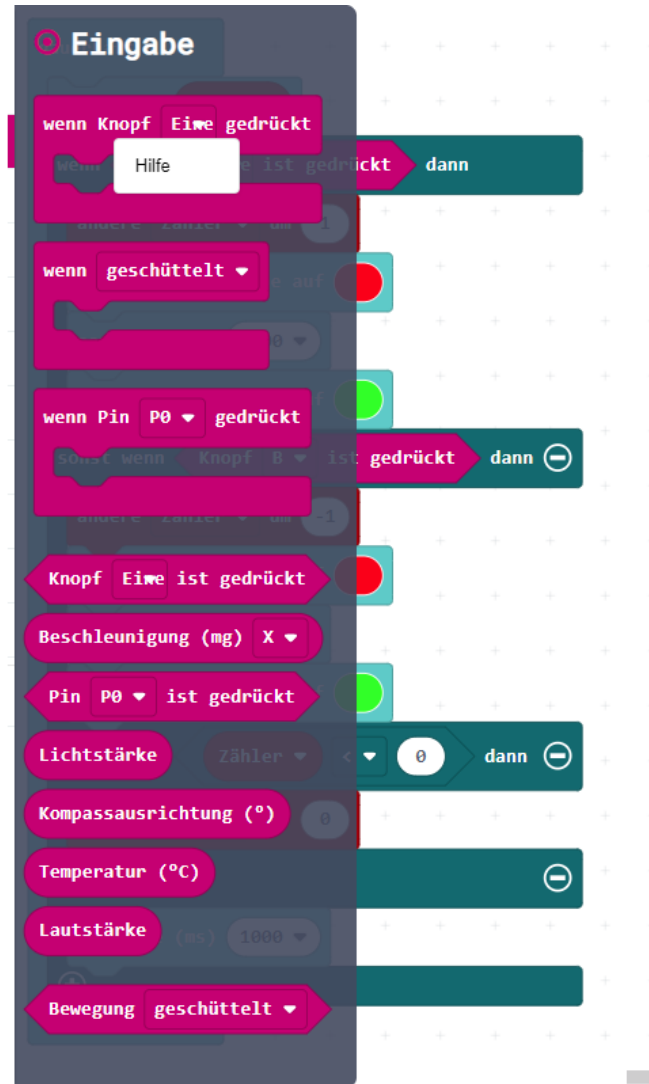
**Würfel** - Taste A Zahlen von 1-6, Taste B Zahlen 1-4

Blöcke	JAVA-Script	Python
	<pre>input.onButtonPressed(Button.A, function on_button_pressed_a() {     basic.showNumber(randint(1, 6)) }) input.onButtonPressed(Button.B, function on_button_pressed_b() {     basic.showNumber(randint(1, 4)) })</pre>	<pre>def on_button_pressed_a():     basic.show_number(randint(1, 6)) input.on_button_pressed(Button.A, on_button_pressed_a)  def on_button_pressed_b():     basic.show_number(randint(1, 4)) input.on_button_pressed(Button.B, on_button_pressed_b)</pre>

# Calliope Workshop

Die Online-Hilfe nutzen:

rechte Maustaste



## Auf Knopfgedrückt

Starten Sie einen [Ereignishandler](#) (ein Teil des Programms, das ausgeführt wird, wenn etwas passiert, z. B. wenn eine Schaltfläche gedrückt wird). Dieser Handler funktioniert, wenn die Taste gedrückt oder oder zusammen gedrückt wird. Wenn Sie diese Funktion in einem Webbrowser verwenden, klicken Sie auf die Schaltflächen auf dem Bildschirm anstelle der Schaltflächen auf dem Calliope mini.ABAB

- Für Button oder : Dieser Handler funktioniert, wenn die Taste gedrückt und innerhalb von 1 Sekunde freigegeben wird.AB
- Für und zusammen: Dieser Handler funktioniert, wenn und beide nach unten gedrückt werden, dann wird einer von ihnen innerhalb von 1,5 Sekunden nach dem Drücken der zweiten Taste freigegeben.ABAB

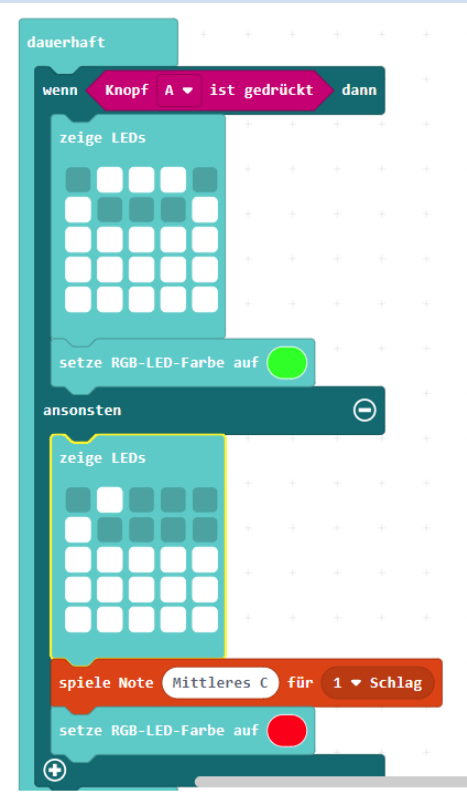
wenn Knopf E1we gedrückt



# Calliope Workshop

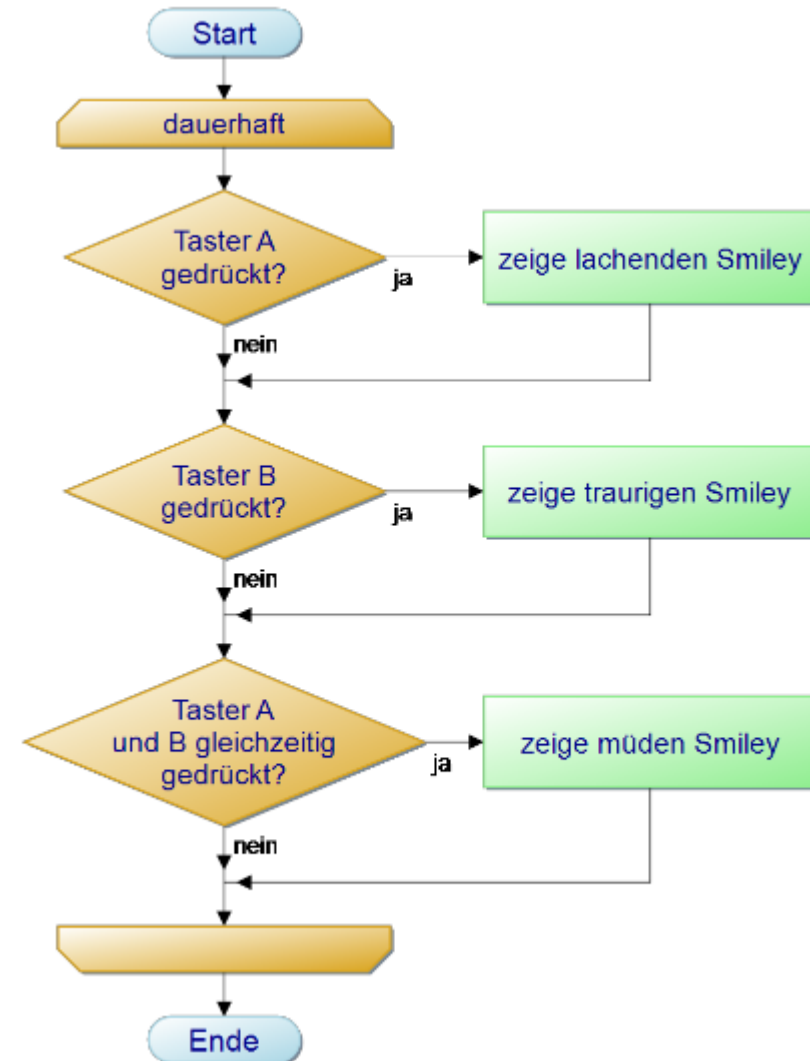
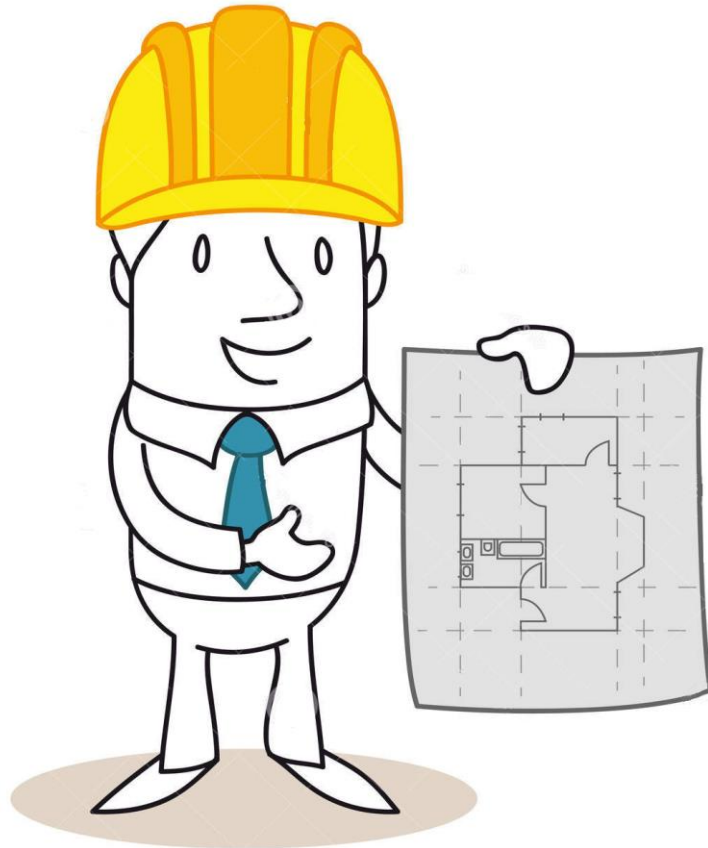


**Keksdose offen?** - Anzeige ob Kontakt (PIN P 0 / Taste A) geschlossen oder offen ist  
Taste A gedrückt: Anzeige Schloss geschlossen und LED=grün; ansonsten Symbols Schloss offen und LED=rot.

Blöcke	JAVA-Script	Python
	<pre>basic.forever(function () {   if (input.buttonIsPressed(Button.A)) {     basic.showLeds(`       . # # # .       # . . . #       # # # # #       # # # # #       # # # # #       `)     basic.setLedColor(0x00ff00)   } else {     basic.showLeds(`       . # . . .       # . . . .       # # # # #       # # # # #       # # # # #       `)     music.playTone(262,     music.beat(BeatFraction.Whole))     basic.setLedColor(0xff0000)   } })</pre>	<pre>def on_forever():   if input.button_is_pressed(Button.A):     basic.show_leds("""       . # # # .       # . . . #       # # # # #       # # # # #       # # # # #       # # # # #       """)     basic.set_led_color(0x00ff00)   else:     basic.show_leds("""       . # . . .       # . . . .       # # # # #       # # # # #       # # # # #       # # # # #       """)     music.play_tone(262,     music.beat(BeatFraction.WHOLE))     basic.set_led_color(0xff0000)   basic.forever(on_forever)</pre>

# Calliope Workshop

## Einen Plan haben



# Calliope Workshop



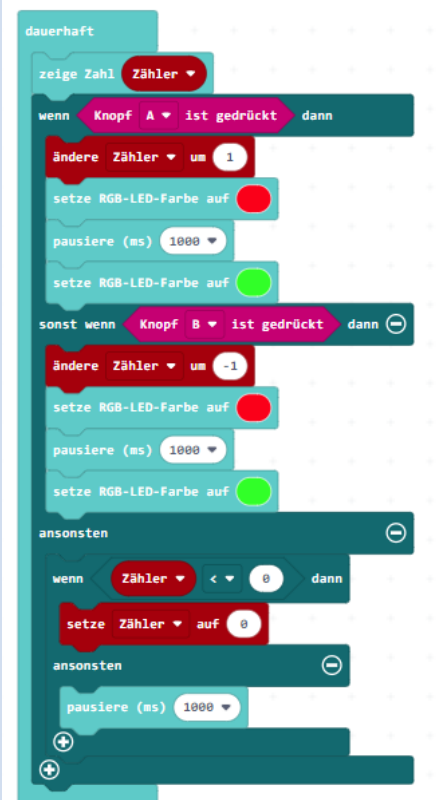
**Personenzähler** - Beim Eintreten in ein Freibad (Taste A - Drehkreuz) wird ein Zähler um +1 erhöht, beim Austritt (Taste B - Drehkreuz) wird der Zähler um 1 subtrahiert. Zwischen den Zählvorgängen soll 1 sec Pause liegen und in der Zeit eine Ampel von grün auf rot schalten.

Blöcke	JAVA-Script	Python
	<pre>let Zähler = 0 basic.forever(function () {   basic.showNumber(Zähler)   if (input.buttonIsPressed(Button.A)) {     Zähler += 1     basic.setLedColor(0xff0000)     basic.pause(1000)     basic.setLedColor(0x00ff00)   } else {     if (input.buttonIsPressed(Button.B)) {       Zähler += -1       basic.pause(1000)       basic.setLedColor(0xff0000)       basic.pause(1000)     } else {       basic.setLedColor(0x00ff00)     }   } })</pre>	<pre>Zähler = 0 def on_forever():   global Zähler   basic.show_number(Zähler)   if input.button_is_pressed(Button.A):     Zähler += 1     basic.set_led_color(0xff0000)     basic.pause(1000)     basic.set_led_color(0x00ff00)   else:     if       input.button_is_pressed(Button.B):         Zähler += -1         basic.pause(1000)         basic.set_led_color(0xff0000)         basic.pause(1000)     else:       basic.set_led_color(0x00ff00)   basic.forever(on_forever)</pre>

# Calliope Workshop



**Personenzähler – Erweiterung 1** Beim vorigen Personenzähler ist es möglich, dass eine Minus- Anzeige zustande kommt, wenn beim Zählerstand Null die Taste B noch einmal gedrückt wird. Ändere das Programm ab, dass eine Anzeige unter Null nicht möglich ist.

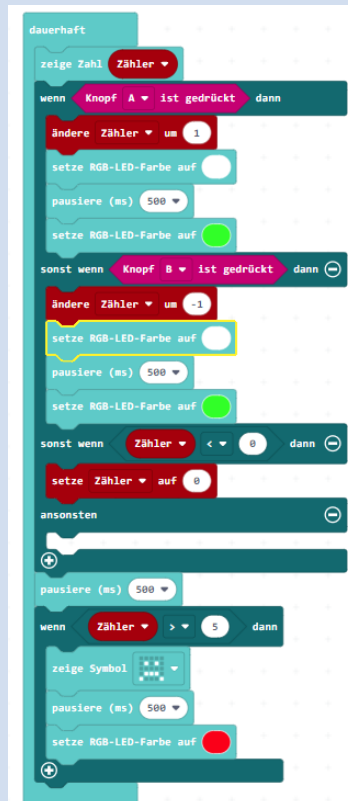
Blöcke	JAVA-Script	Python
	<pre>let Zähler = 0 basic.forever(function on_forever() {    basic.showNumber(Zähler)   if (input.buttonIsPressed(Button.A)) {     Zähler += 1     basic.setLedColor(0xff0000)     basic.pause(1000)     basic.setLedColor(0x00ff00)   } else if (input.buttonIsPressed(Button.B)) {     Zähler += -1     basic.setLedColor(0xff0000)     basic.pause(1000)     basic.setLedColor(0x00ff00)   } else if (Zähler &lt; 0) {     Zähler = 0   } else {     basic.pause(1000)   } })</pre>	<pre>let Zähler = 0 basic.forever(function on_forever() {    basic.showNumber(Zähler)   if (input.buttonIsPressed(Button.A)) {     Zähler += 1     basic.setLedColor(0xff0000)     basic.pause(1000)     basic.setLedColor(0x00ff00)   } else if (input.buttonIsPressed(Button.B)) {     Zähler += -1     basic.setLedColor(0xff0000)     basic.pause(1000)     basic.setLedColor(0x00ff00)   } else if (Zähler &lt; 0) {     Zähler = 0   } else {     basic.pause(1000)   } })</pre>

# Calliope Workshop



**Personenzähler – Erweiterung 2** Wird eine Höchstzahl von Besuchern (5) überschritten, soll ein ☹️ erscheinen und der Eingang gesperrt werden (rote LED).

## Blöcke



## JAVA-Script

```
let Zähler = 0
basic.forever(function () {
  basic.showNumber(Zähler)
  if (input.buttonIsPressed(Button.A)) {
    Zähler += 1
    basic.setLedColor(0xffffff)
    basic.pause(500)
    basic.setLedColor(0x00ff00)
  } else if (input.buttonIsPressed(Button.B)) {
    Zähler += -1
    basic.setLedColor(0xffffff)
    basic.pause(500)
    basic.setLedColor(0x00ff00)
  } else if (Zähler < 0) {
    Zähler = 0
  } else {
  }
  basic.pause(500)
  if (Zähler > 5) {
    basic.showIcon(IconNames.Sad)
    basic.pause(500)
    basic.setLedColor(0xff0000)
  }
})
```

## Python

```
Zähler = 0
def on_forever():
    global Zähler
    basic.show_number(Zähler)
    if input.button_is_pressed(Button.A):
        Zähler += 1
        basic.set_led_color(0xffffff)
        basic.pause(500)
        basic.set_led_color(0x00ff00)
    elif input.button_is_pressed(Button.B):
        Zähler += -1
        basic.set_led_color(0xffffff)
        basic.pause(500)
        basic.set_led_color(0x00ff00)
    elif Zähler < 0:
        Zähler = 0
    else:
        pass
    basic.pause(500)
    if Zähler > 5:
        basic.show_icon(IconNames.SAD)
        basic.pause(500)
        basic.set_led_color(0xff0000)
basic.forever(on_forever)
```



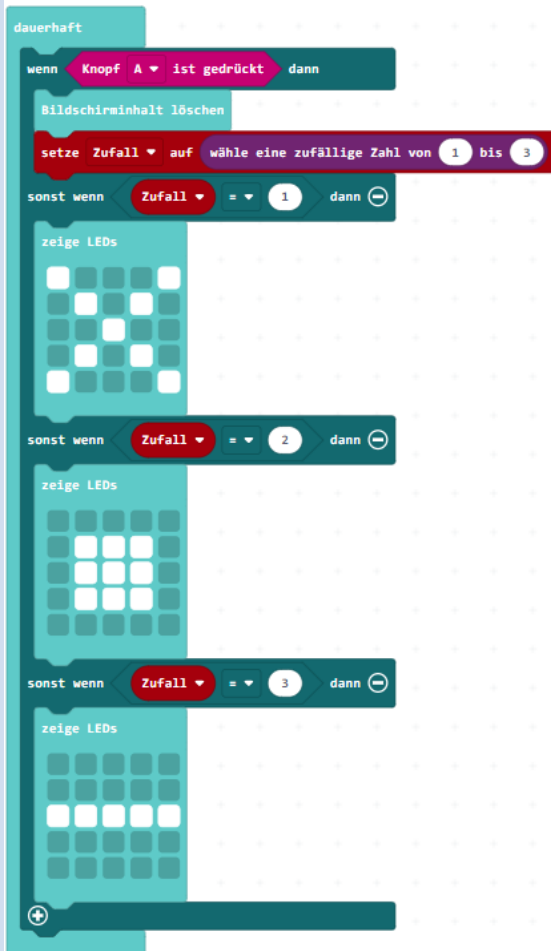
# Calliope Workshop



## Zufallsspiel: Schere, Stein, Papier

Programmiere das Schere, Stein, Papier Spiel

### Blöcke



### JAVA-Script

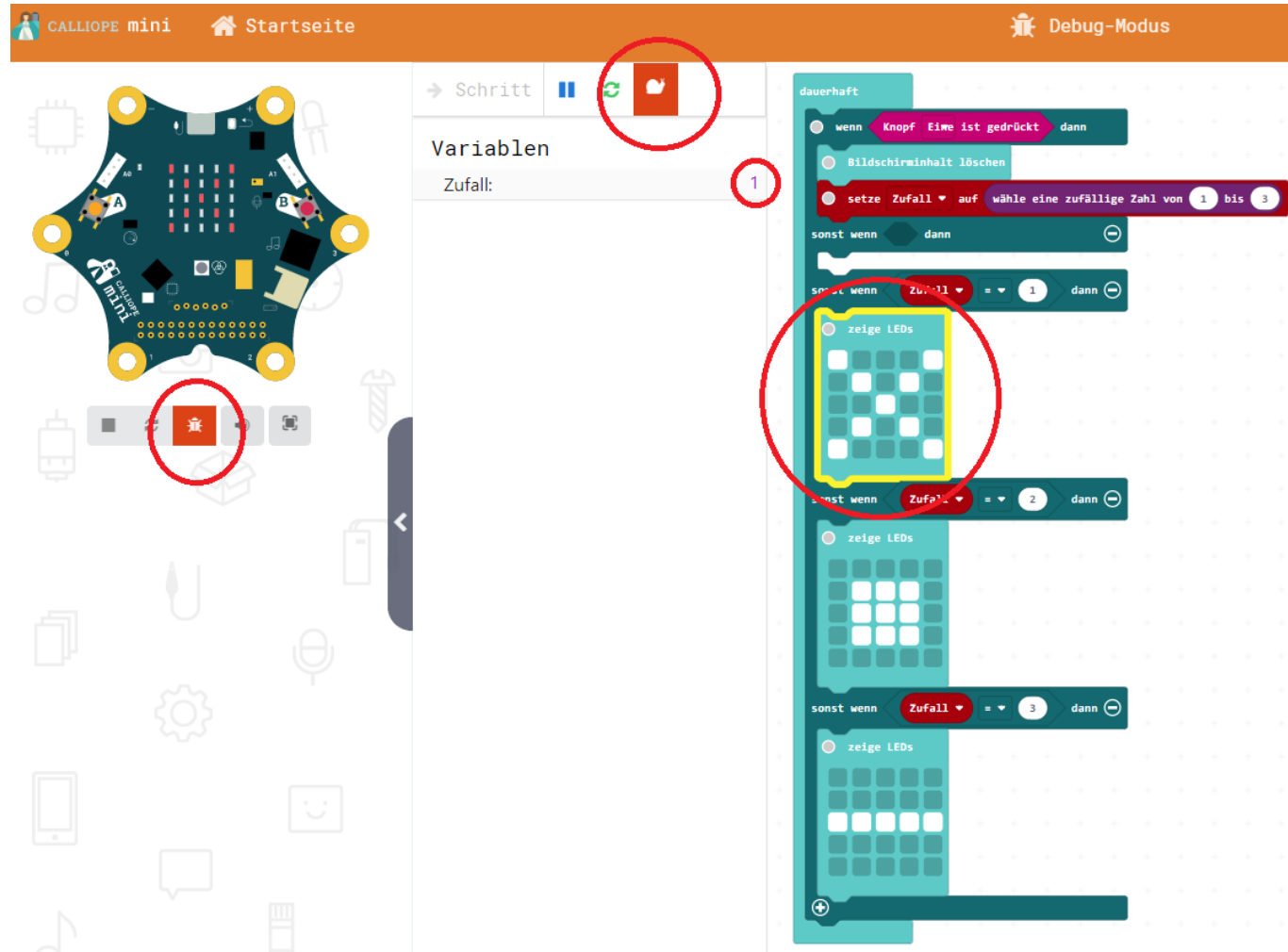
```
let Zufall = 0
basic.forever(function () {
  if (input.buttonIsPressed(Button.A)) {
    basic.clearScreen()
    Zufall = randint(1, 3)
  } else {
    if (Zufall == 1) {
      basic.showLeds(`
        # . . #
        . # . # .
        # . # . .
        . # . # .
        # . . . #
        `)
    } else if (Zufall == 2) {
      basic.showLeds(`
        . . . . .
        . # # # .
        . # # # .
        . # # # .
        . . . . .
        `)
    } else if (Zufall == 3) {
      basic.showLeds(`
        . . . . .
        . . . . .
        # # # # #
        . . . . .
        . . . . .
        `)
    }
  }
})
```

### Python

```
Zufall = 0
def on_forever():
  global Zufall
  if input.button_is_pressed(Button.A):
    basic.clear_screen()
    Zufall = randint(1, 3)
  else:
    if Zufall == 1:
      basic.show_leds("""
        # . . #
        . # . # .
        # . # . .
        . # . # .
        # . . . #
        """)
    elif Zufall == 2:
      basic.show_leds("""
        . . . . .
        . # # # .
        . # # # .
        . # # # .
        . . . . .
        """)
    elif Zufall == 3:
      basic.show_leds("""
        . . . . .
        . . . . .
        # # # # #
        . . . . .
        . . . . .
        """)
basic.forever(on_forever)
```

# Calliope Workshop

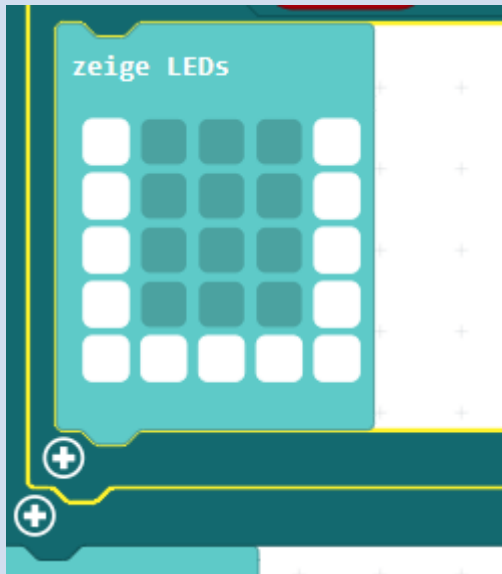
## Fehlersuche: Der Debug-Modus



# Calliope Workshop

## Zufallsspiel: Schere, Stein, Papier - füge noch einen Brunnen hinzu

### Blöcke



### JAVA-Script

```
let Zufall = 0
basic.forever(function () {
  if (input.buttonIsPressed(Button.A)) {
    basic.clearScreen()
    Zufall = randint(1, 4)
  } else if (false) {

  } else {
    if (Zufall == 1) {
      basic.showLeds(`
        # . . #
        . # . # .
        . . # . .
        . # . # .
        # . . #
      `)
    } else if (Zufall == 2) {
      basic.showLeds(`
        . . . . .
        . # # # .
        . # # # .
        . # # # .
        . . . . .
      `)
    } else if (Zufall == 3) {
      basic.showLeds(`
        . . . . .
        . . . . .
        # # # # #
        . . . . .
        . . . . .
      `)
    } else if (Zufall == 4) {
      basic.showLeds(`
        # . . #
        # . . #
        # . . #
        # # # # #
      `)
    }
  }
})
```

### Python

```
Zufall = 0

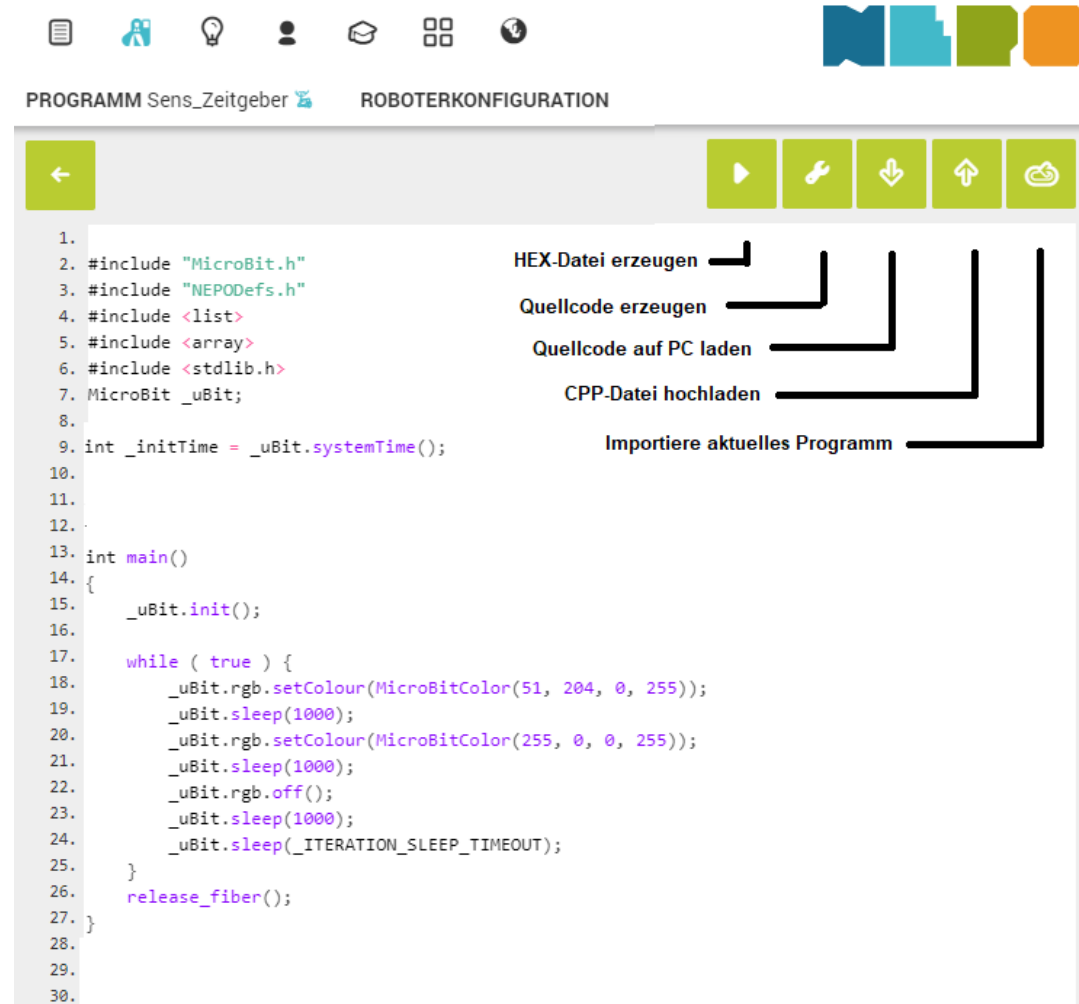
def on_forever():
    global Zufall
    if input.button_is_pressed(Button.A):
        basic.clear_screen()
        Zufall = randint(1, 4)
    elif False:
        pass
    else:
        if Zufall == 1:
            basic.show_leds("""
                # . . #
                . # . # .
                . . # . .
                . # . # .
                # . . #
            """)
        elif Zufall == 2:
            basic.show_leds("""
                . . . . .
                . # # # .
                . # # # .
                . # # # .
                . . . . .
            """)
        elif Zufall == 3:
            basic.show_leds("""
                . . . . .
                . . . . .
                # # # # #
                . . . . .
                . . . . .
            """)
        elif Zufall == 4:
            basic.show_leds("""
                # . . #
                # . . #
                # . . #
                # # # # #
            """)
basic.forever(on_forever)
```

# Calliope Workshop



## Calliope C++ Editor und Compiler mit Open Roberta:

<https://lab.open-roberta.org/>



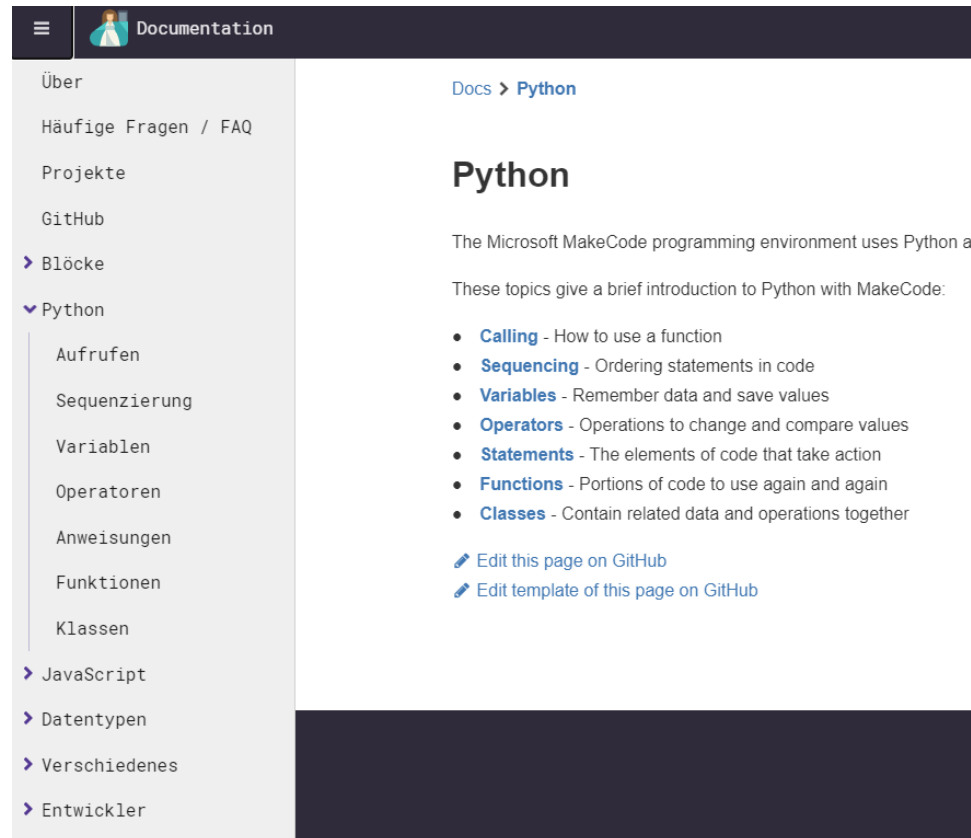
C++ Beispiel  
LED schalten im  
Open Roberta Editor

# Calliope Workshop



Wie geht's weiter?

Online Tipps und Hilfen: <https://makecode.calliope.cc/about>

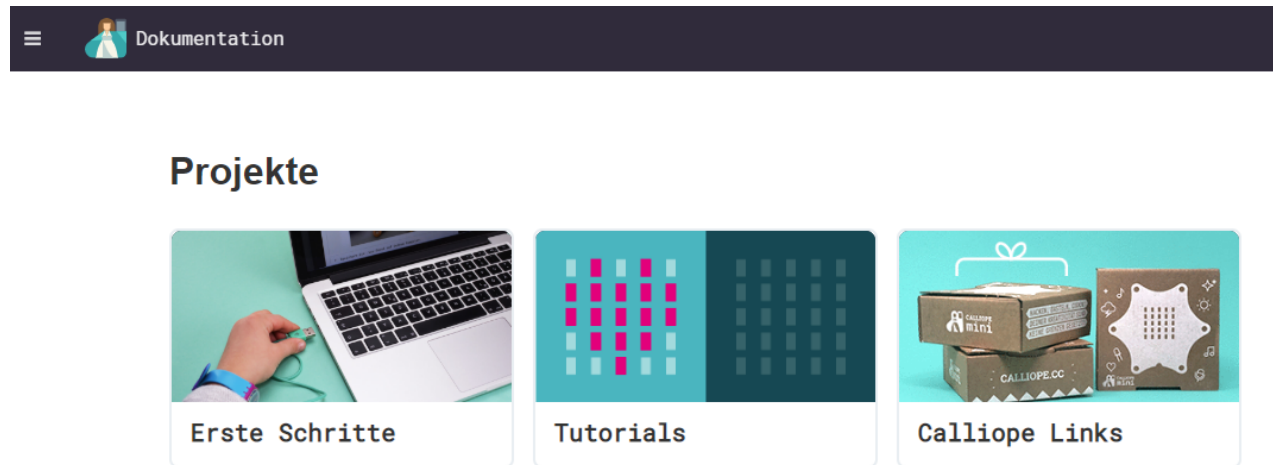


# Calliope Workshop



Wie geht's weiter?

Projekte und Anregungen: <https://makecode.calliope.cc/projects>



## Siehe auch

[Erste Schritte](#), [Tutorials](#), [Calliope Links](#)

[✎ Bearbeiten dieser Seite auf GitHub](#)

[✎ Vorlage dieser Seite auf GitHub bearbeiten](#)

# Calliope Workshop



**Übungsaufgaben:**

# Calliope Workshop



## Aufgabe

Mit der Taste A soll eine Treppenhausfunktion realisiert werden. Nach Betätigung der Taste A schaltet die LED für 2s ein und danach wieder aus.



# Calliope Workshop



## Lösung

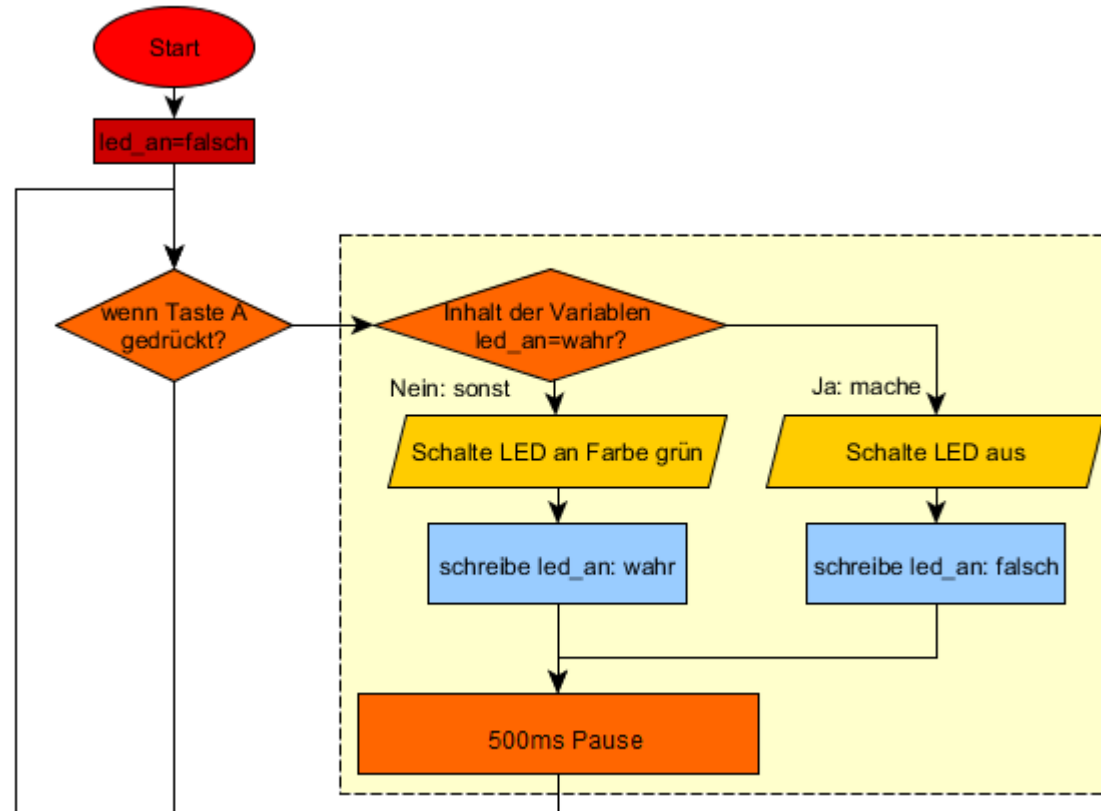


### Python

```
def on_forever():
    if input.button_is_pressed(Button.A):
        basic.set_led_color(0xff0000)
        basic.pause(2000)
        basic.turn_rgb_led_off()
basic.forever(on_forever)
```

## Aufgabe

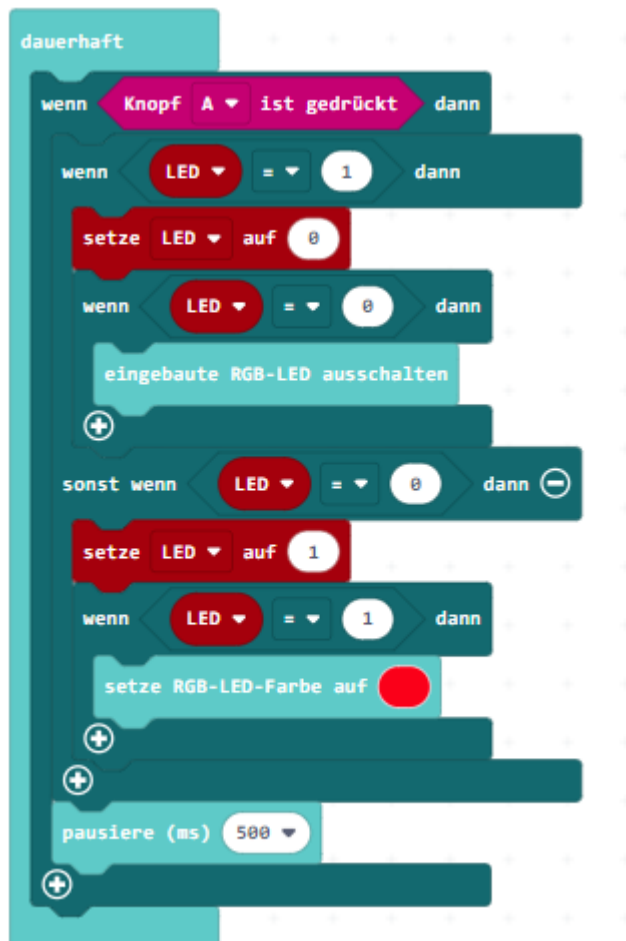
Mit der Taste A soll die RGB-LED ein- oder ausgeschaltet werden und der Zustand nach dem Loslassen der Taste erhalten bleiben (Wechsel-Schalter).



# Calliope Workshop



## Lösung



## Python

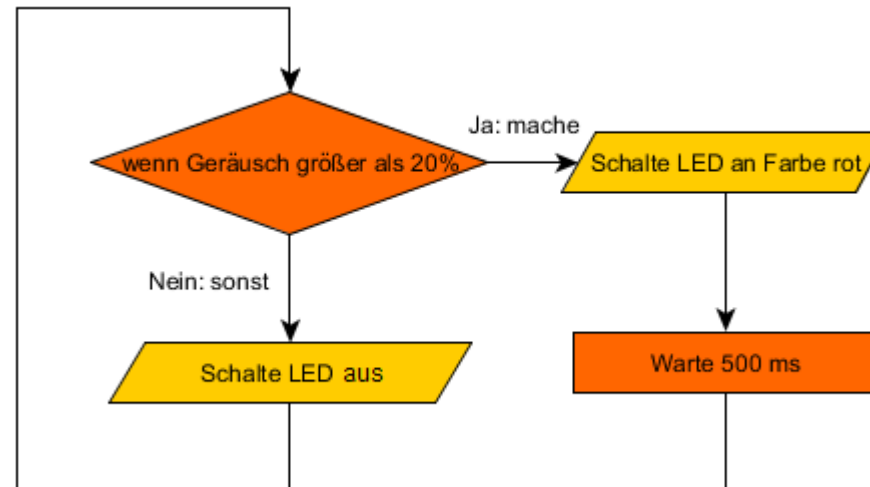
```
LED = 0
```

```
def on_forever():  
    global LED  
    if input.button_is_pressed(Button.A):  
        if LED == 1:  
            LED = 0  
        if LED == 0:  
            basic.turn_rgb_led_off()  
    elif LED == 0:  
        LED = 1  
        if LED == 1:  
            basic.set_led_color(0xff0000)  
        basic.pause(500)  
basic.forever(on_forever)
```

# Calliope Workshop

## Aufgabe

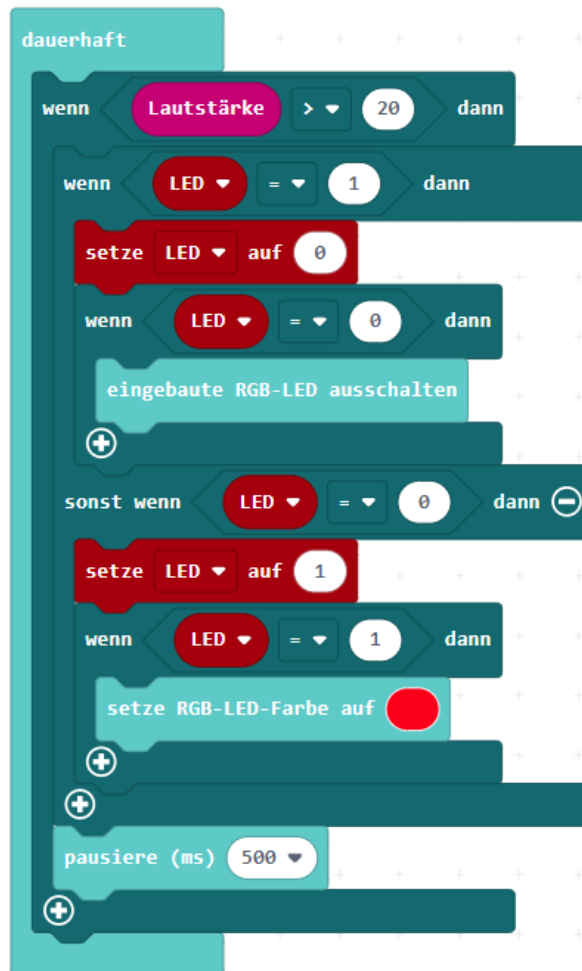
Auf der Basis der vorherigen Aufgabe soll ein Klatschschalter realisiert werden. Mit einem Geräusch (Lautstärke > 20%) wird die LED ein/ausgeschalten.



# Calliope Workshop



## Lösung



## Python

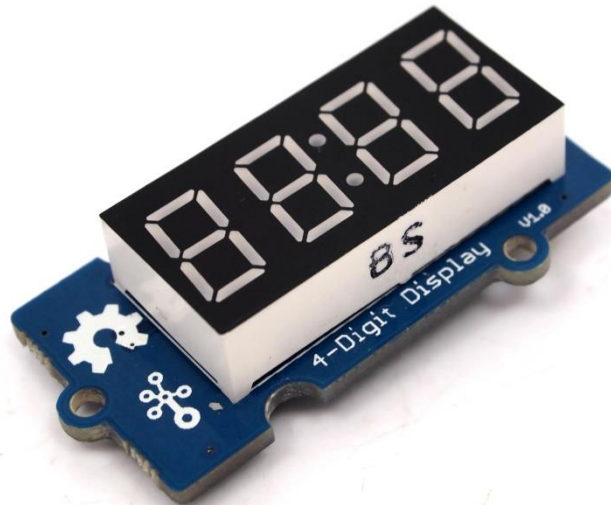
```
LED = 0
```

```
def on_forever():
    global LED
    if input.sound_level() > 20:
        if LED == 1:
            LED = 0
        if LED == 0:
            basic.turn_rgb_led_off()
    elif LED == 0:
        LED = 1
        if LED == 1:
            basic.set_led_color(0xff0000)
        basic.pause(500)
basic.forever(on_forever)
```

# Calliope Workshop

## Aufgabe

Am Calliope-Anschluss A1 (rechts) kann ein „Grove 4-Digit Display“ angeschlossen werden. Das Display hat 4 Stellen. Erstelle mit MakeCode ein Programm, das die beiden rechten Stellen von 1 – 3 hochzählt, wenn die Taste A betätigt ist.



# Calliope Workshop



## Lösung



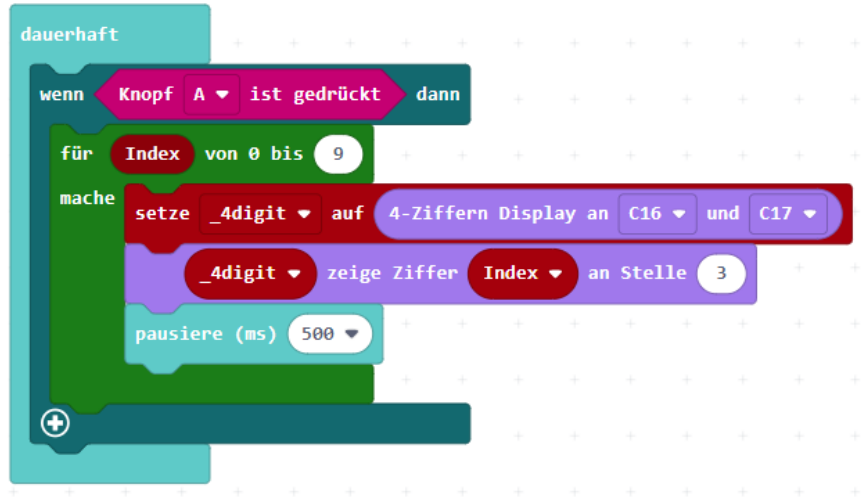
## Python

```
_4digit: grove.TM1637 = None

def on_forever():
    global _4digit
    if input.button_is_pressed(Button.A):
        _4digit = grove.create_display(DigitalPin.C16, DigitalPin.C17)
        _4digit.bit(1, 3)
        basic.pause(1000)
        _4digit.bit(2, 3)
        basic.pause(1000)
        _4digit.bit(3, 3)
        basic.pause(1000)
        _4digit.bit(1, 2)
        basic.pause(1000)
        _4digit.bit(2, 2)
        basic.pause(1000)
        _4digit.bit(3, 2)
        basic.pause(1000)
        _4digit.clear()
    basic.forever(on_forever)
```

## Lösung mit einer Schleife

Wenn Taste A gedrückt wurde, zählt die rechte Anzeige bis 9



## Python

```
_4digit: grove.TM1637 = None

def on_forever():
    global _4digit
    if input.button_is_pressed(Button.A):
        for Index in range(10):
            _4digit = grove.create_display(DigitalPin.C16, DigitalPin.C17)
            _4digit.bit(Index, 3)
            basic.pause(500)
        basic.forever(on_forever)
```

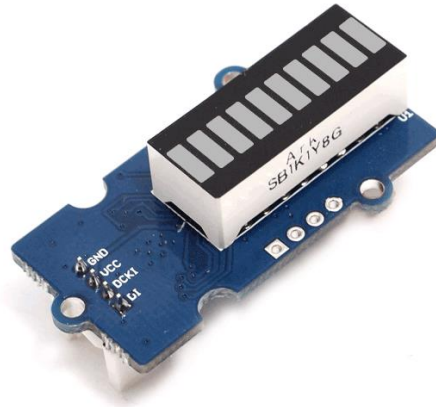


# Calliope Workshop



## Aufgabe

Am Calliope-Anschluss A1 (rechts) kann eine „Grove LED Bar“ angeschlossen werden. Die LED-Bar hat 10 Segmente (1xrot, 1xgelb, 8xgrün). Mit Open Roberta <https://lab.open-roberta.org/#> programmieren wir ein Lauflicht. Die LED-Segmente sollen von 1-10 nacheinander einschalten und nach 0,5s wieder ausschalten. Öffne den Quellcode-Editor und löse die Aufgabe in C++.



# Calliope Workshop



## Lösung



PROGRAMM NEPOprog

ROBOTERKONFIGURATION



```
1. #define _GNU_SOURCE
2.
3. #include "MicroBit.h"
4. #include "NEPODefs.h"
5. #include "Grove_LED_Bar.h"
6. #include <list>
7. #include <array>
8. #include <stdlib.h>
9. MicroBit _uBit;
10. Grove_LED_Bar _ledBar(MICROBIT_PIN_P8, MICROBIT_PIN_P2);
11.
12.
13.
14. int main()
15. {
16.     _uBit.init();
17.
18.     for (int __k0 = 0; __k0 < 10; __k0 += 1) {
19.         _ledBar.setLed(0, 6);
20.         _uBit.sleep(500);
21.         _ledBar.setLed(0, 0);
22.         _ledBar.setLed(1, 5);
23.         _uBit.sleep(500);
24.         _ledBar.setLed(1, 0);
25.         _ledBar.setLed(2, 5);
26.         _uBit.sleep(500);
```

# Calliope Workshop



## Aufgabe

Wir dimmen den roten Balken an der „Grove LED Bar“ mit Hilfe einer Schleife in 6 Helligkeitsstufen von aus bis hell.



```
1. #define _GNU_SOURCE
2.
3. #include "MicroBit.h"
4. #include "NEP0Defs.h"
5. #include "Grove_LED_Bar.h"
6. #include <list>
7. #include <array>
8. #include <stdlib.h>
9. MicroBit _uBit;
10. Grove_LED_Bar _ledBar(MICROBIT_PIN_P8, MICROBIT_PIN_P2);
11.
12.
13.
14. int main()
15. {
16.     _uBit.init();
17.
18.     while ( true ) {
19.         for (int __i = 0; __i < 6; __i += 1) {
20.             _ledBar.setLed(0, __i);
21.             _uBit.sleep(500);
22.             _uBit.sleep(_ITERATION_SLEEP_TIMEOUT);
23.         }
24.         _uBit.sleep(_ITERATION_SLEEP_TIMEOUT);
25.     }
26.     release_fiber();
27. }
```

Erweitere das Programm in Open Roberta für alle 10 LED-Balken als Lauflicht mit der Schleifenfunktion.

# Calliope Workshop

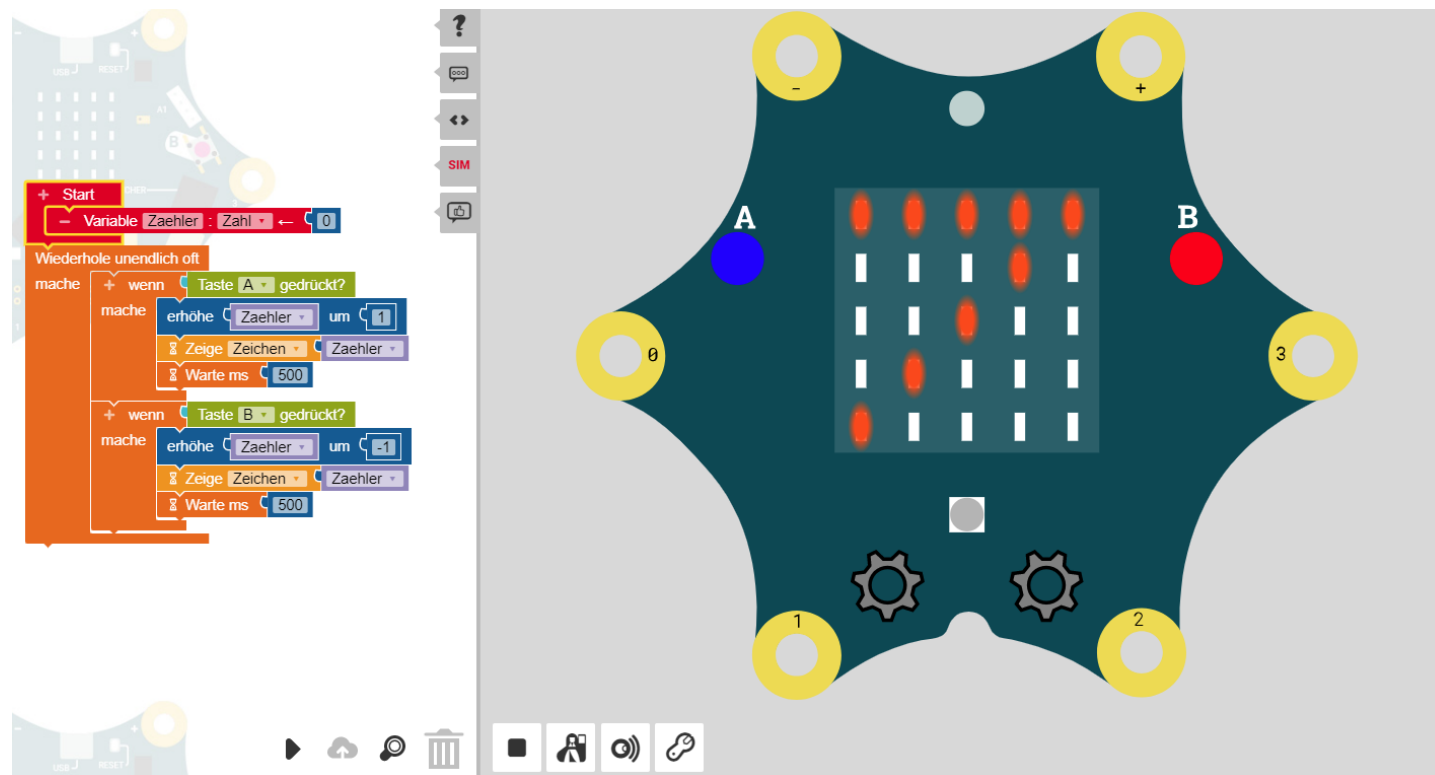
## Lösung



# Calliope Workshop

## Aufgabe

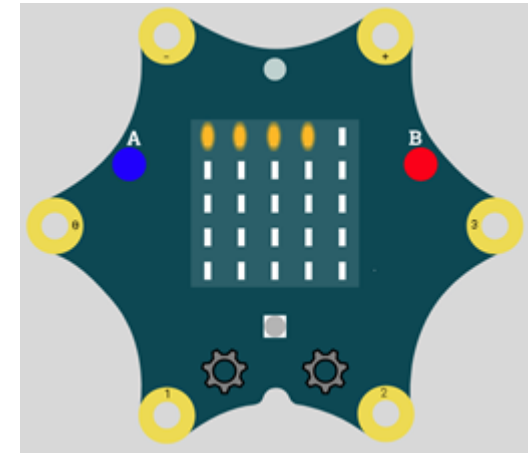
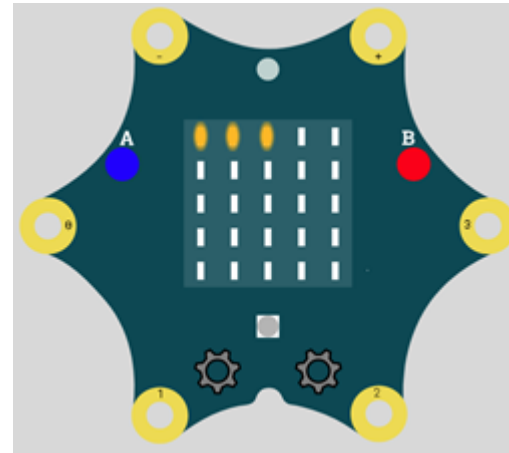
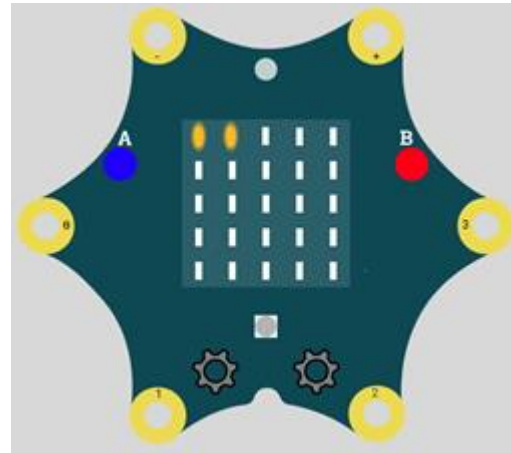
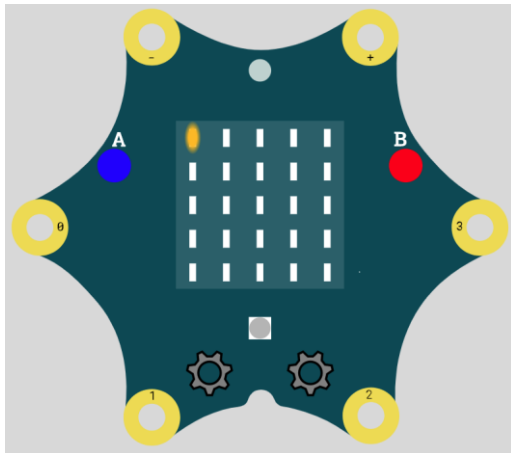
Mit einem Open Roberta Programm wollen wir zählen, wie oft die Taste A gedrückt wurde. Der Zählerstand soll als Zahl im LED-Display des Calliope angezeigt werden. Dazu können wir die Simulation nützen. Mit der Taste B soll der Zählerstand um 1 verringert werden.



# Calliope Workshop

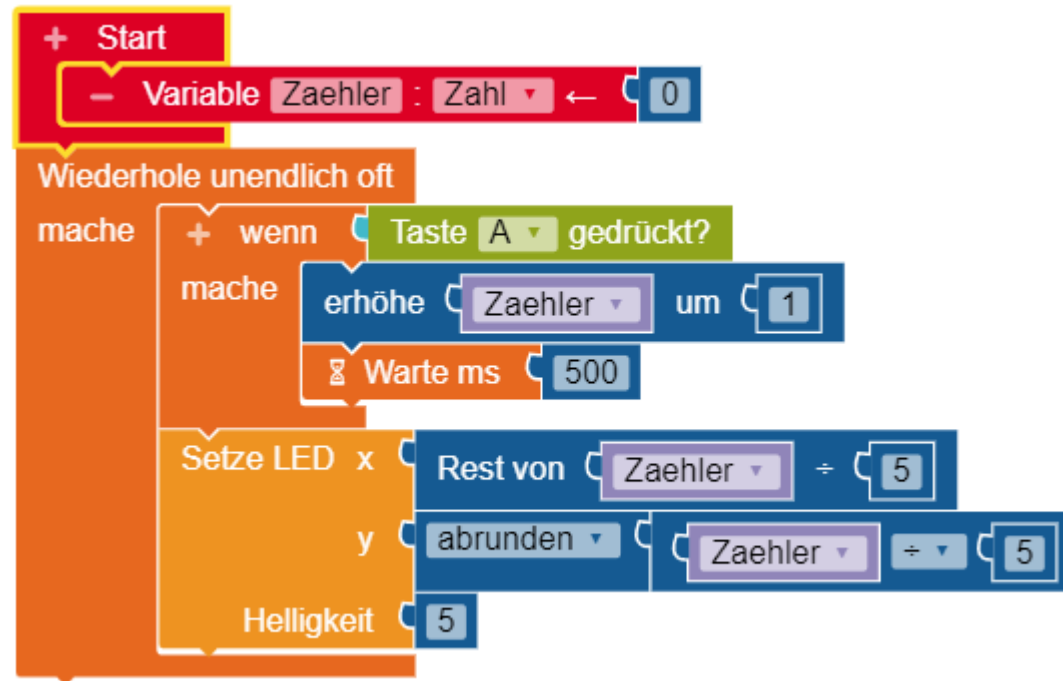
## Aufgabe

Mit einem Open Roberta Programm wollen wir zählen, wie oft die Taste A gedrückt wurde. Der Zählerstand soll als zusätzlicher **LED-Punkt** im LED-Display des Calliope angezeigt werden.



# Calliope Workshop

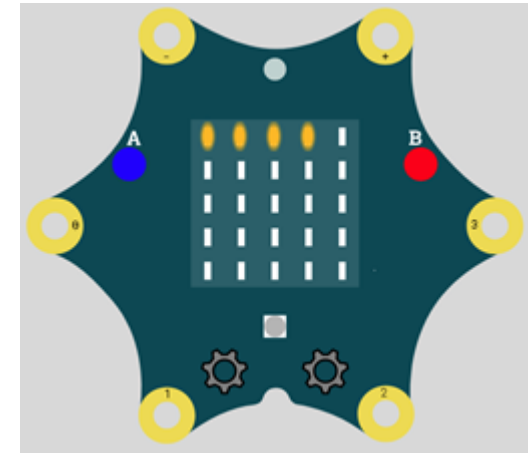
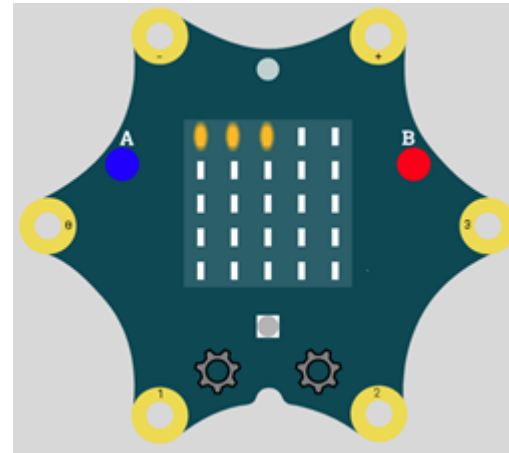
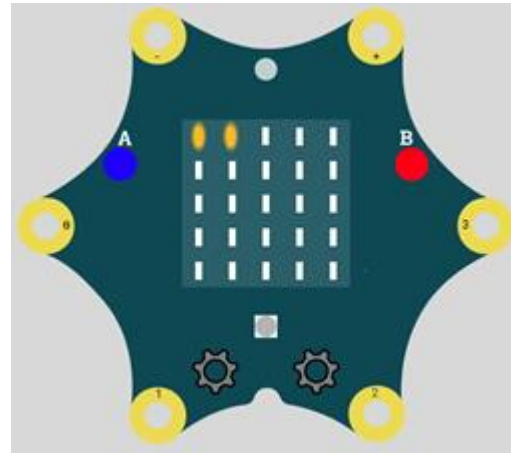
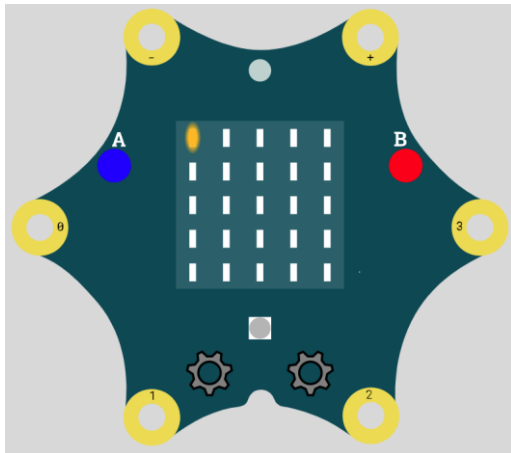
## Lösung



# Calliope Workshop

## Aufgabe

Mit einem Open Roberta Programm soll ein **Zähler** gestartet werden, der den Zählerstand als zusätzlichen **LED-Punkt** im LED-Display des Calliope angezeigt.

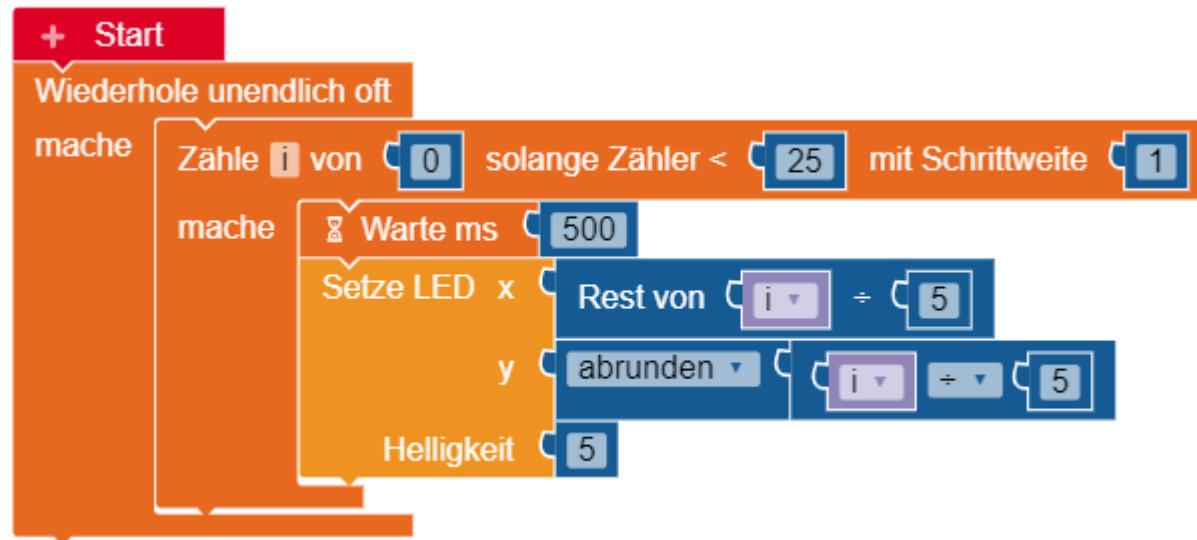




# Calliope Workshop



## Lösung



## C++

```
#define _GNU_SOURCE
```

```
#include "MicroBit.h"
#include "NEPODefs.h"
#include <list>
#include <array>
#include <stdlib.h>
MicroBit_uBit;
```

```
int main()
{
    _uBit.init();
    _uBit.display.setDisplayMode(DISPLAY_MODE_GREYSCALE);
    while ( true ) {
        for (int __i = 0; __i < 25; __i += 1) {
            _uBit.sleep(500);
            _uBit.display.image.setPixelValue((int) __i % ((int) 5), floor(__i / ((float) 5)), (5) *
_SET_BRIGHTNESS_MULTIPLIER);
            _uBit.sleep(_ITERATION_SLEEP_TIMEOUT);
        }
        _uBit.sleep(_ITERATION_SLEEP_TIMEOUT);
    }
    release_fiber();
}
```