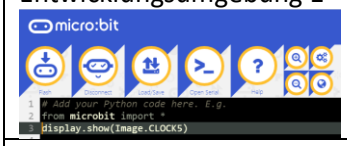
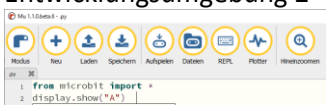
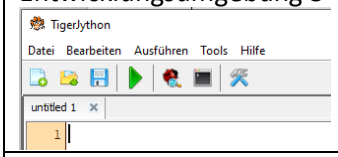
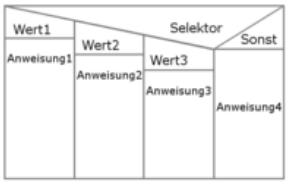
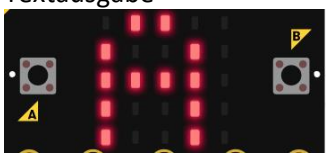
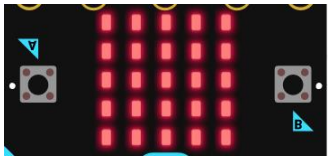
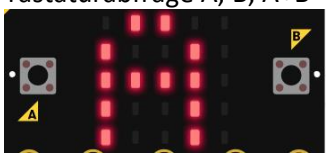
	Micro:Bit
Entwicklungsumgebung 1 	https://python.microbit.org/v/2 (Nicht mit Firefox ausführbar) Der Python-Editor ist ideal für alle, die ihre Programmierkenntnisse vertiefen möchten. Eine Auswahl an Snippets und eine Reihe von vorgefertigten Bildern, Sounds und Musik unterstützen Sie bei der Erstellung Ihres Codes.
Entwicklungsumgebung 2 	Mu ist ein Editor, der speziell und ausschließlich für die Programmiersprache Python 3 entwickelt wird. Der Editor hat eine sehr übersichtliche Oberfläche, die nur aus eine Icon-Leiste besteht, es gibt keine weiteren Menüs. https://codewith.mu/en/download (unter Modus - Micro-Bit wählen)
Entwicklungsumgebung 3 	TigerJython ist eine einfache Entwicklungsumgebung für das Programmieren mit Python. Sie enthält alle für die Programmierung notwendigen Komponenten. Variante 1 - Online www.python-online.ch/microbit/MicrobitOnline.php Variante 2 - als Programm www.tigerjython.ch/de/products/download Vorbereitende Arbeiten unter *1
Bibliothek	`from microbit import *` # Beginn eines jeden Programms # bei Bedarf weitere Bibliotheken einbinden
Kopfschleife (For-Schleife) Wiederhole so oft wie	`from microbit import *` `for i in range(5):` # Programmzeilen werden 4 mal wiederholt (0 – 4) `  display.show(i)` # gibt den Wert i auf dem Display aus (0 – 4) `  sleep(500)` # nach der Wertedarstellung 0,5 sek Pause
Kopfschleife (While-Schleife) Prüfe vor Ausführung	`while True:` # wiederholt die Blöcke, solange Bedingung wahr `  tue etwas` # die folgenden Zeilen werden eingerückt
Verzweigung (bedingte Anweisung) 	`from microbit import *` `age = 17` # Variable age wird mit 17 angelegt `while True:` # Endlosschleife für Abfrage der Tasten `  if age > 18:` # Abfrage, ob age größer 18 `  display.show(Image. SMILE)` # dann Displayanzeige Smilie `  elif age < 18 :` # Abfrage, ob age kleiner 18 `  display.show(Image.SAD)` # dann Displayanzeige trauriger Smilie `  else:` # Alternative wenn age == 18 `  display.show(Image. YES)` # dann Displayanzeige Haken
Textausgabe 	`from microbit import *` `display.show("A")` # Ausgabe eines Zeichens oder `while True:` # Endlosschleife `  display.scroll('Axel')` # Ausgabe Scrolltext oder `  display.scroll("Python ", delay=60)` # Geschwindigkeit der Darstellung
Bildausgabe auf Matrix 	`from microbit import *` `while True:` # Endlosschleife `  display.show(Image.HAPPY)` # Happy für Smilie weitere Bilder siehe Anhang
Bildausgabe auf Matrix Eigenbau mit differenzierter Helligkeit	`from microbit import *` `boat = Image("01234:"` # Jedes Pixel wird individuell angesprochen `  "56789:"` # die Intensität des Pixel kann mit Werten `  "00000:"` # zwischen 0 und 9 angesprochen werden `  "99999:"` # Alternative Darstellung `  "00000")` # („01234:56789:00000:99999:00000“) `display.show(boat)` # eigentliche Bilddarstellung
Bildausgabe auf Matrix Animation 	`from microbit import *` `display.show(Image.ALL_CLOCKS, loop=True, delay=100)` # Anzeige Sek. oder `bild1 = Image(" 00000:00000:00000:00000:00000")` # Bild 1 erstellen `bild2 = Image(" 99999:99999:99999:99999:99999")` # Bild 2 erstellen `bilder = [bild1, bild2]` # Erzeugung einer Liste `while True:` # Endlosschleife `  display.show(bilder, delay= 200)` # Animationsdarstellung mit Werten aus Liste

Tastaturabfrage A, B, A+B



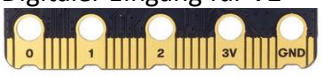

```
from microbit import *
while True:                          # Endlosschleife für Abfrage der Tasten
  if button_a.is_pressed():          # Abfrage, ob A gedrückt
    display.show("A")                # Displayanzeige “A”
  elif button_b.is_pressed():        # Abfrage, ob B gedrückt
    display.show("B")                # Displayanzeige “B”
  else:                              # Alternative wenn nichts gedrückt
    display.show("0")                # Displayanzeige „0“
```

Digitaler Eingang Pin 0 -2

Kontakt gegen 3V

```
from microbit import *
while True:                          # Dauerschleife zur Abfrage der Pins
  if pin2.read_digital():            #  Abfrage, ob Pin2 Ja (1), dann
    display.show(Image.YES)          # Symbol Ja
  else:                              # Abfrage, ob Pin 2 Nein (0), dann
    display.show(Image.NO)           # Symbol Nein
```

Digitaler Eingang für V2



Fingerkontakt reicht

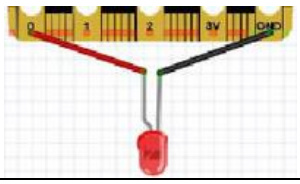
Zusätzlich: pin_logo

```
from microbit import *
pin0.set_touch_mode(pin0.CAPACITIVE) # Berührungsmodus kapazitiv für V2
while True:                          # Dauerschleife zur Abfrage d. Pins
  if pin0.is_touched():              # wurde Pin_logo berührt
    display.show(Image.HAPPY)        # Symbol Glücklich
  else:                              # wird Pin_logo nicht berührt
    display.show(Image.SAD)          # Symbol Unglücklich
```

Analoger Eingang Pin 0 -2
0 V bis 3,3 V -> 0 bis 1023
Gegen + messen

```
from microbit import *
while True:                          # Dauerschleife
  display.show(pin0.read_analog())   # Anzeige Wert zw. 0 und 1023
```

Digitaler Ausgang Pin 0 -2

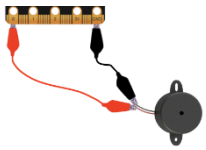


```
from microbit import *
while True:                          # Dauerschleife
 pin0.write_digital(1)               # Pin 0 Digital 1 (3,3 V) gegen Masse
 sleep(500)                          # Pause 0,5 sek.
 pin0.write_digital(0)               # Pin 0 Digital 0 (0 V) gegen Masse)
 sleep(500)                          # Pause 0,5 sek.
```

Analoger Ausgang Pin 0 – 2
Gegen Masse (GND)
(Pulsweitenmodulation)

```
from microbit import *
while True:                          # Dauerschleife
  pin0.write_analog(512)             # Ausgabe ca. 1,6 V (Wert 0 bis 1023)
```

Soundausgabe an Pin 0



Passiv
Piezo / Summer
5V
16 Ohm

```
from microbit import *
from music import *                  # Biblothek Musik einbinden
pin0.write_digital(1)                # Summer einschalten
tune = ["C4:4", "D4:4", "E4:4", "C4:4"]  # Liste erzeugen; Oktave, Note,:, Dauer
play(tune)                           # abspielen der Liste oder
```

```
from microbit import *
from music import *                  # Biblothek Musik einbinden
song = [262, 294, 330, 349, 392, 392,]  # Liste mit Frequenzen
for f in song:                       # mit jedem Durchlauf Übergabe a pitch
  pitch(f, 500)                      # Erzeugt Ton „f“ für 500 ms oder
```

```
from microbit import *
from music import *                  # Biblothek Musik einbinden
play(BADDY)                          # abspielen vordefinierter Musik
                                     # weitere Musik siehe Anhang
```

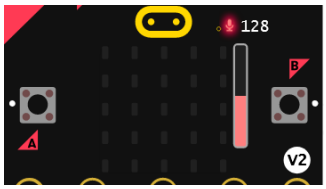
Soundausgabe V2 (Intern)


```
from microbit import *
from music import *                  # Biblothek Musik einbinden
speaker.on()                         # Lautsprecher an; off() – Lautsprecher aus
audio.play(Sound.GIGGLE)             # spielt Soundeffekt GIGGLE
                                     # weitere Effekte siehe Anhang
```

Sprachausgabe V2

```
from microbit import *
import speech                        # Bibliothek Sprache installieren
speech.say("Hallo, World")           # Sprachausgabe Lautsprecher
```

Mikrofon V2

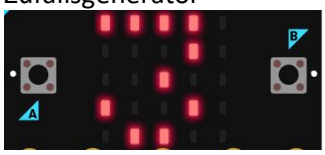



```
from microbit import *
while True:                                    # Dauerschleife
  display.show(microphone.sound_level())       # Wert zw. 0-255 oder
```

```
while True:                                    # Dauerschleife
  if microphone.current_event() == SoundEvent.LOUD:   # wenn Laut
    display.show(Image.HEART)                  # großes Herz
    sleep(200)                                 # Pause
  if microphone.current_event() == SoundEvent.QUIET:  # wenn leise
    display.show(Image.HEART_SMALL)            # kleines Herz oder
```

```
lights = Image("11111:11111:11111:11111:11111")   # Bild vorbereiten
while True:                                       # Dauerschleife
  display.show(lights * microphone.sound_level())  # Helligkeit in
                                                   # Abhängigkeit von Lautstärke
```

Zufallsgenerator

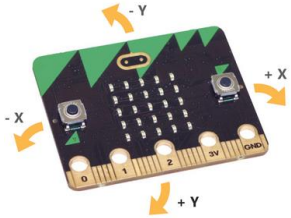



```
from microbit import *
import random                                  # Bibliothek Zufall installieren
display.show(str(random.randint(1, 6)))  # Ausgabe einer Zahl zw. 1 und 6 oder
```

```
wert = random.randrange(100) + random.random() # Ausgabe v. Gleitkommazahl
display.scroll(str(wert))                      # Ausgabe 4 Stellen nach d. Komma oder
```

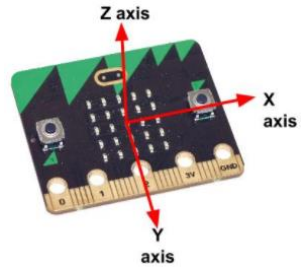
```
name = [" Axel  ", " Uwe  ", " Daniel  "]  # Erzeugen einer Liste
display.scroll(random.choice(name))      # Ausgabe eines Zufallswertes
```

Bewegungsmeldung (Lage)
Beispiel Wasserwaage



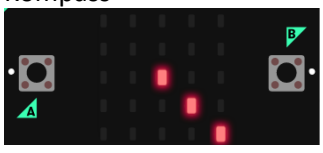

```
from microbit import *
while True:                                    # Dauerschleife zur Abfrage der Lage
  reading = accelerometer.get_x()              # Auslesen des x-Wertes
  if reading > 20:                             # Abfrage ob Wert größer 20
    display.show("R")                          # Ausgabe von R wie rechts
  elif reading < -20:                          # Abfrage ob Wert kleiner 20
    display.show("L")                          # Ausgabe von L wie links
  else:                                        # Ansonsten (hier der Wert 20)
    display.show("-")                          # Ausgabe von “-“ für Gerade
```

Gesten (Bewegung)




```
from microbit import *
while True:                                    #Dauerschleife z. Abfrage der Bewegung
  be = accelerometer.current_gesture() # Auslesen der Bewegung
  if be == "shake":                            # ist die Bewegung „schütteln“ oder
# up, down, left, right, face up, face down (Logo nach oben/unten, freefall,
# 3g, 6g, 8g (Beschleunigungswert G)
    display.show(Image.HAPPY)          # zeige Symbol “Lustig”
  else:                                        # ansonsten
    display.show(Image.ANGRY)          # zeige Symbol “Traurig”
```

Kompass




```
from microbit import *
compass.calibrate()                            # Gerät kalibrieren, bewegen bis
while True:                                    # alle LED leuchten
  needle = ((15 - compass.heading()) // 30) % 12  # Kompassfunktion u Berechnung
  display.show(Image.ALL_CLOCKS[needle])       # Ausgabe des Zeigers Rtg. Nord
```

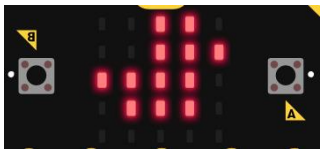
Temperatur auslesen

```
from microbit import *
while True:                                    # Dauerschleife für Abfrage
  display.show(temperature())                  # Anzeige der Temperatur
```

Helligkeit auslesen

```
from microbit import *
while True:                                    # Dauerschleife für Abfrage
  display.show(display.read_light_level())     # Anzeige Helligkeit zw. 0 und 255
```

Drahtlose Kommunikation zwischen 2 micro:bits



Wird ein Micro:bit geschüttelt wandert die Ente zum anderen Micro:bit

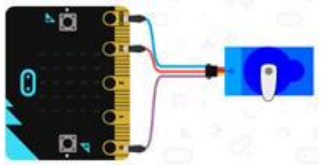
```
from microbit import *
import radio                                    # entsprechende Bibliothek laden
radio.config(group=23)                          # einheitliche Funkgruppe festlegen
radio.on()                                      # Kommunikation einschalten
while True:                                     # Dauerschleife
  message = radio.receive()                     # Variable übergeben
  if message:                                   # wenn Nachricht dann
    display.show(Image.DUCK)                    # Bild der Ente
  if accelerometer.was_gesture('shake'): # wenn Micro:bit geschüttelt dann
    display.clear()                             # lösche alle Pixel
    radio.send('duck')                          # sende Ente
```

Neopixel

```
from microbit import *
from neopixel import NeoPixel                   # entsprechende Bibliothek laden
num_pixels = 18                                 # Anzahl der LED´s übergeben
np = NeoPixel(pin5, num_pixels)                 # Initialisieren inkl. Angabe Port
np[12] = (255, 255, 0)                          # der LED 12 einen Farbwert übergeben
np.show()                                       # Neopixel einschalten  oder
```

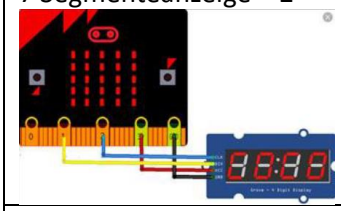
```
for x in range(0, 18):                          # mit einer Schleife alles LED´s
  np[x] = (0, 255, 0)                           # einen Farbwert zuweisen
  np.show()                                     # Neopixel einschalten
```

Servo



```
from microbit import *
while True:                                     # Dauerschleife
  pin0.write_analog(50)                         # 50 = 1ms für Rechtsanschlag
  sleep(1000)                                   # Pause 1 sek
  pin0.write_analog(75)                         # 75 = 1,5ms für Mittelstellung
  sleep(1000)                                   # Pause 1 sek
  pin0.write_analog(100)                        # 100 0 2ms für Linksanschlag
  sleep(1000)                                   # Pause 1 sek
```

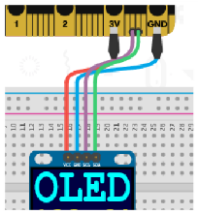
7 Segmenteanzeige *2



```
from microbit import *                          # Achtung – vorbereitende Arbeit *3
from mb7seg import FourDigit                    # Funktion aus Modul mb7seg laden
d = FourDigit(pin0, pin1)                       # Anschlüsse definieren 0=CLK, 1=DIO
for n in range(1000):                           # Schleife
 d.show(n)                                      # Zählt von 1 bis 999      oder
```

```
d.show(5,-1)                                    # Position der ersten Ziffer (leer,-1,-2,-3)
```

OLED *2



I²C

```
from microbit import *                          # Achtung – vorbereitende Arbeit *3
import oled                                     # Modul OLED laden
oled.init()                                     # Funktion initialisieren
oled.text(0, 0, "Hello Python")                 # Textausgabe Zeile 0 ab Position 0
oled.text(5, 1, "4567")                         # Ziffernausgabe ab Position 6
n = 123456789123                                # Variable erzeugen
oled.text(0, 2, str(n))                         # Variableninhalt als String ausgeben oder
```

```
n = 1000                                        # Variable einen Wert zuweisen
while True:                                     # Schleifenbeginn
 oled.text(0,0, "v: %-9d" %n)                   # formatierte Ausgabe
 n -=1                                          # Variable um 1 verringern
# v: = Ausgabetext; % = Platzhalter f. Variable; -9 = Anzahl d. darzustellenden Werte;
# % = String-Modulo-Operator; n = Variable
```

*1 - Hardware auswählen -> Tools/ Devices/ microbit:Calliope
- Tool flashen -> Tools/ Flash Target/ warten bis “All down” erscheint
- Ausführung am PC mit F5, Ausführung am Micro:bit mit Shift + F7

*2 - nur mit tigerjyton getestet

*3 - mbmodules.zip unter
http://www.tigerjython4kids.ch/download/mbmodules.zip herunterladen
- In einem eigenem Ordner entpacken. Es sind die einzelnen Module als py-Datei sichtbar.
- Entsprechendes Modul in TigerJyton laden und auf Microbit kopieren. Tools/Module hinunterladen/Editor. Nun Programm schreiben und laden.

Anhang

Symbole

HEART		HEART_ SMALL		HAPPY		SMILE		BUTTERFLY	
SAD		CONFUSED		ANGRY		ASLEEP		GIRAFFE	
SURPRISED		SILLY		FABULOUS		MEH		STICK FIGURE	
YES		NO		ARROW_N E,S,W,NW		TRIANGLE		SKULL	
TRIANGLE_ LEFT		CHESS BOARD		DIAMOND		DIAMOND SMALL		GHOST	
SQUARE		SQUARE_ SMALL		RABBIT		COW		UMBRELLA	
XMAS		PITCHFORK		TARGET		TSHIRT		SWORD	
ROLLER SKATE		DUCK		HOUSE		TORTOISE		SNAKE	

Pin-Belegung

0,1,2,8,12, 16	Allgemeine Digital-in/Digital-out	5, 11	vordefiniert Button A, B
0, 1, 2	Analog-in/analog-out (PWM)	13, 14, 15	vordefiniert SPI
3, 4, 6, 7, 9, 10	vordefiniert LED display	19, 20	vordefiniert I2C

Musik Micro.Bit

ADADADUM - Eröffnung von Beethoven's
ENTERTAINER - Eröffnungsfragment
PRELUDE -Eröffnung des ersten Prelude in C Dur
ODE - "Ode an Joy" Thema aus Beethoven's 9.
NYAN - das Nyan Cat Thema
RINGTONE - ein Klingelton
FUNK - ein Geräusch für Geheimagenten
BLUES - ein Boogie-Woogie Blues
BIRTHDAY - "Happy Birthday to You..."
WEDDING - der Chorus des Bräutigams aus"Lohengrin"
FUNERAL - der "Trauerzug"
PUNCHLINE - a lustiger Tonclip
PYTHON - John Philip Sousa's Marsch "Liberty Bell"
BADDY - Filmclip aus "The Baddy"
CHASE - Filmclick aus einer Jagdszene
BA_DING - ein Signalton, dass etwas geschehen ist
WAWAWAWAA - ein trauriger Posaunenklang
JUMP_UP - für Spiele, Aufwärtsbewegung
JUMP_DOWN - für Spiele, Abwärtsbewegung
POWER_UP - ein Fanfarenklangdass etwas erreicht wurde
POWER_DOWN - ein trauriger Fanfarenklang, der darauf hinweist, dass etwas verloren gegangen ist

Soundeffekte Micro:bit V2

kichern	GIGGLE	glücklich	HAPPY	aufsteigend	SOARING
Hello	HELLO	geheimnisvoll	MYSTERIOUS	Frühling	SPRING
traurig	SAD	gleiten	SLIDE	funkeln	TWINKLE