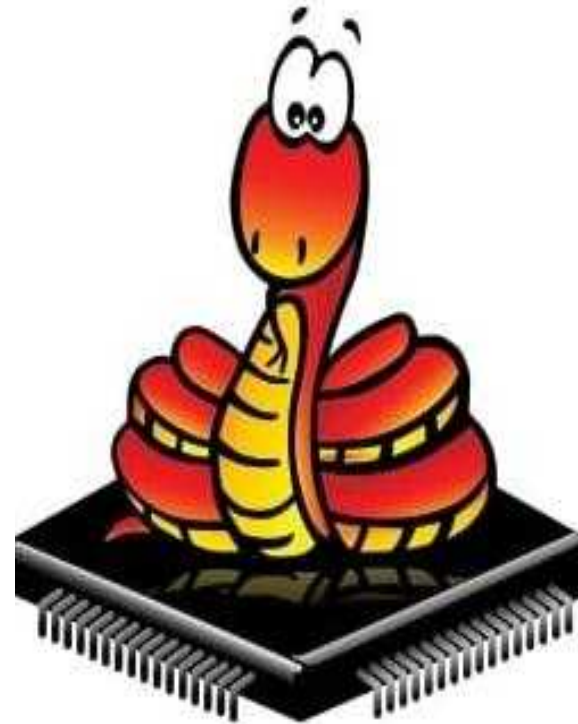
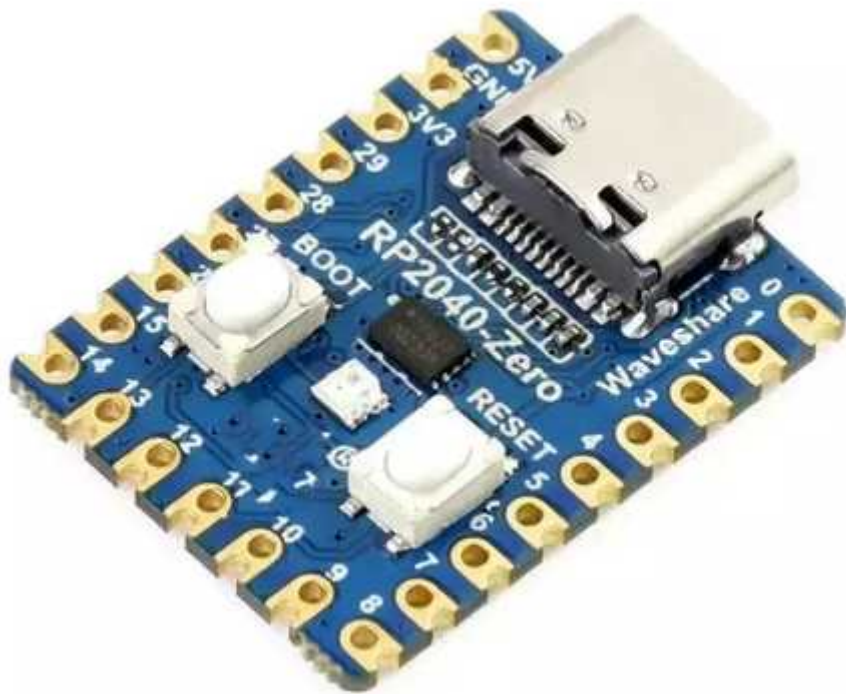


Anwendungen in Micropython



Kapitel-Übersicht

1

Vorstellung einiger Boards

- ESP8266 , ESP32
- RaspberryPico RP2040

2

Python bzw. Micropython

- Ranking Programmiersprachen
- Vorführung: Python am PC

3

Entwicklungsoberfläche

- Installation von Thonny
- Einstellungen

4

Praktische Beispiele

Zielobjekt: CW-Bake ,CW-Keyer

- Led / Blink- Led
- Autostart
- Taster + Led
- Poti am AD-Wandler
- Tonerzeugung / Taste + Ton

5

Verweise

Kapitel 1

1

Vorstellung einiger Boards

- ESP8266 , ESP32
- RaspberryPico RP2040

2

Python bzw. Micropython

- Ranking Programmiersprachen
- Vorführung: Python am PC

3

Entwicklungsoberfläche

- Installation von Thonny
- Einstellungen

4

Praktische Beispiele

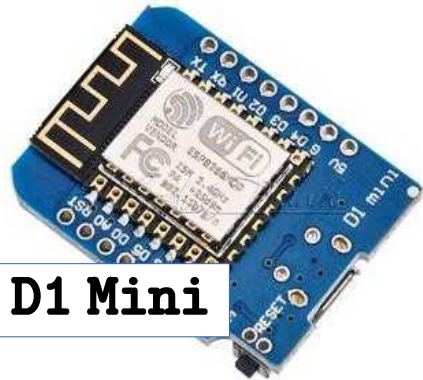
Zielobjekt: CW-Bake ,CW-Keyer

- Led / Blink- Led
- Autostart
- Taster + Led
- Poti am AD-Wandler
- Tonerzeugung / Taste + Ton

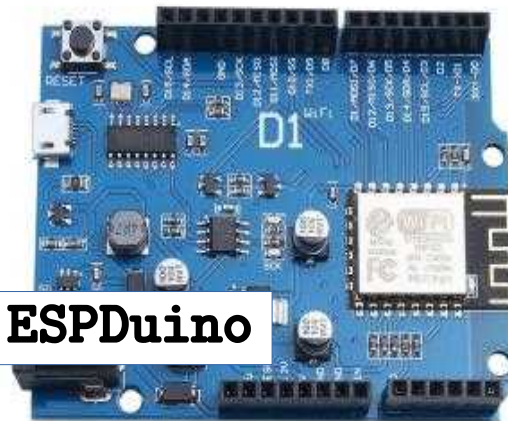
5

Verweise

ESPxx – Boards



D1 Mini



ESPduino

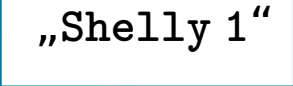


Esp32 s2 mini



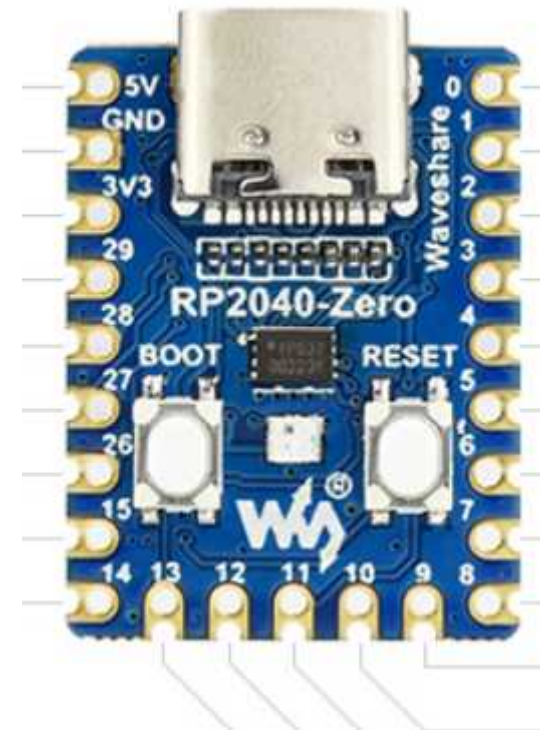
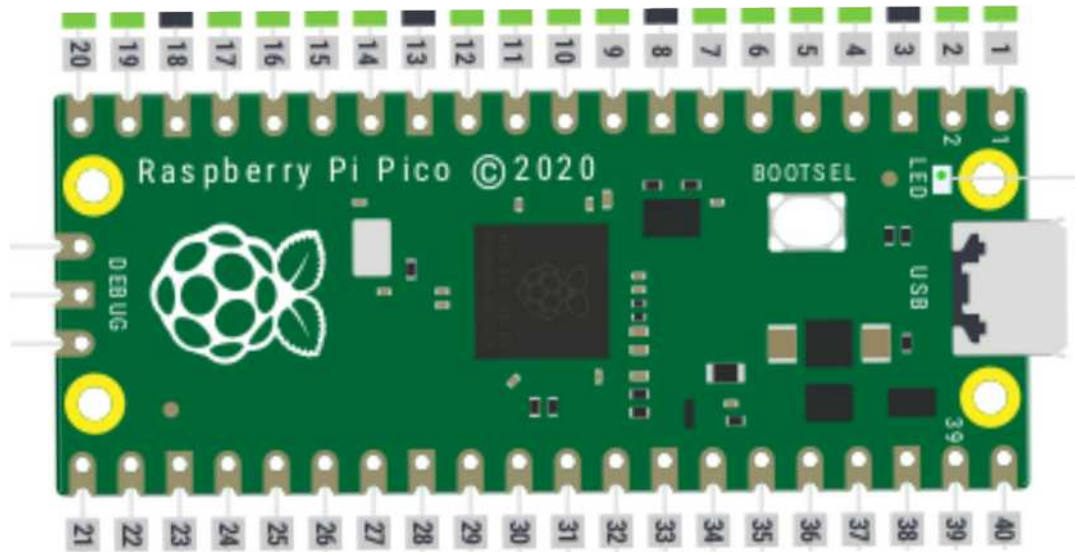
ESP32-C3 Mini

Anwendungsbeispiel ESP8266



Raspberry Pi Pico Boards

Ref.: [6]



Kapitel 2

1

Vorstellung einiger Boards

- ESP8266 , ESP32
- RaspberryPico RP2040

2

Python bzw. Micropython

- Ranking Programmiersprachen
- Vorführung: Python am PC

3

Entwicklungsoberfläche

- Installation von Thonny
- Einstellungen

4

Praktische Beispiele

Zielobjekt: CW-Bake ,CW-Keyer

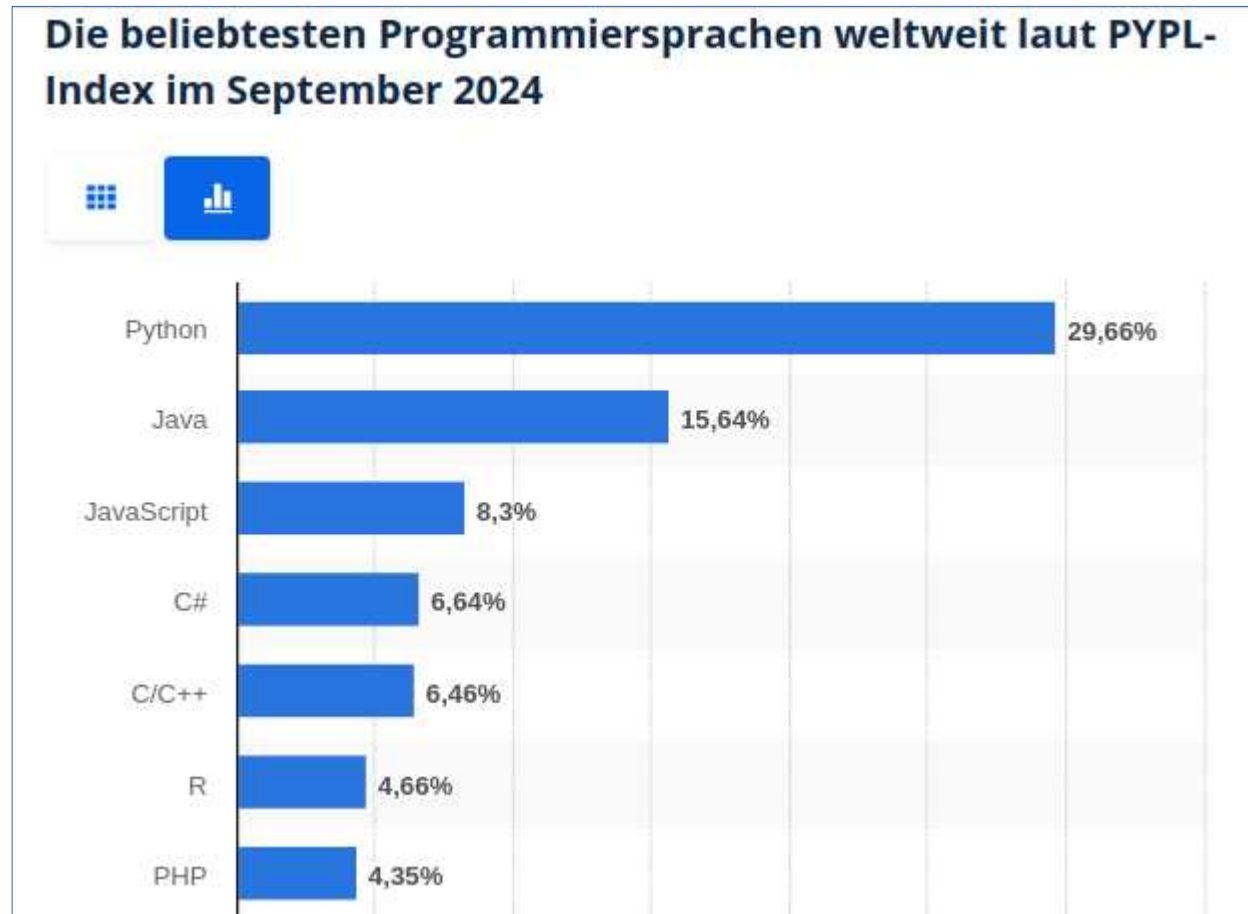
- Led / Blink- Led
- Autostart
- Taster + Led
- Poti am AD-Wandler
- Tonerzeugung / Taste + Ton

5

Verweise

Ranking der Programmiersprachen

<https://de.statista.com/statistik/daten/...>

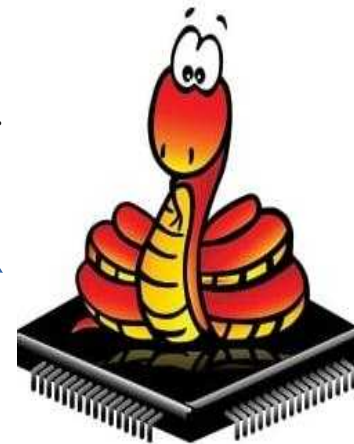


Micropython = Python für Mikroprozessor

Aus



wird für den Mikroprozessor



„Micropython“.

Stand Sept 2024:
Python 3.12.5
Micropython v1.23

Python am PC in der Kommandozeile

Ref.: [1]

Start Python über Kommandozeile:

```
dk2jk@linux:~$ python3.12
Python 3.12.5+ (main, Aug 9 2024, 08:50:51) [GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import time
>>> time.asctime()
'Thu Sep 19 11:05:34 2024'
>>> time.asctime().split()
['Thu', 'Sep', '19', '11:05:41', '2024']
>>> time.asctime().split()[3]
'11:05:48'
>>> █
```

Python am PC als Script

Datei : mytime.py

Das Gleiche als ‚Script‘ im Texteditor:

```
mytime.py
1 import time          # modul time einbinden
2 t=time.asctime()     # zeit lesbar holen 'Thu Sep 19 10:55:14 2024'
3 t1=t.split()         # string teilen ['Thu', 'Sep', '19', '10:55:24', '2024']
4 hms= t1[3]           # das 3. element der liste '10:55:39'
5 text= f'Es ist jetzt {hms} Uhr'
6 print (text)         #'es ist jetzt Thu Sep 19 10:53:23 2024 Uhr'
7
```

Start des Python-Scripts über Kommandozeile:

```
dk2jk@linux:~/Schreibtisch/swt2024$ python3 mytime.py
Es ist jetzt 11:55:41 Uhr
```

Kapitel 3

1

Vorstellung einiger Boards

- ESP8266 , ESP32
- RaspberryPico RP2040

2

Python bzw. Micropython

- Ranking Programmiersprachen
- Vorführung: Python am PC

3

Entwicklungsoberfläche

- Installation von Thonny
- Einstellungen

4

Praktische Beispiele

Zielobjekt: CW-Bake ,CW-Keyer

- Led / Blink- Led
- Autostart
- Taster + Led
- Poti am AD-Wandler
- Tonerzeugung / Taste + Ton

5

Verweise

IDE ‚Thonny‘ installieren

Ref.: [3]

Installieren von Thonny:
<https://thonny.org/>

Thonny
Python IDE for beginners



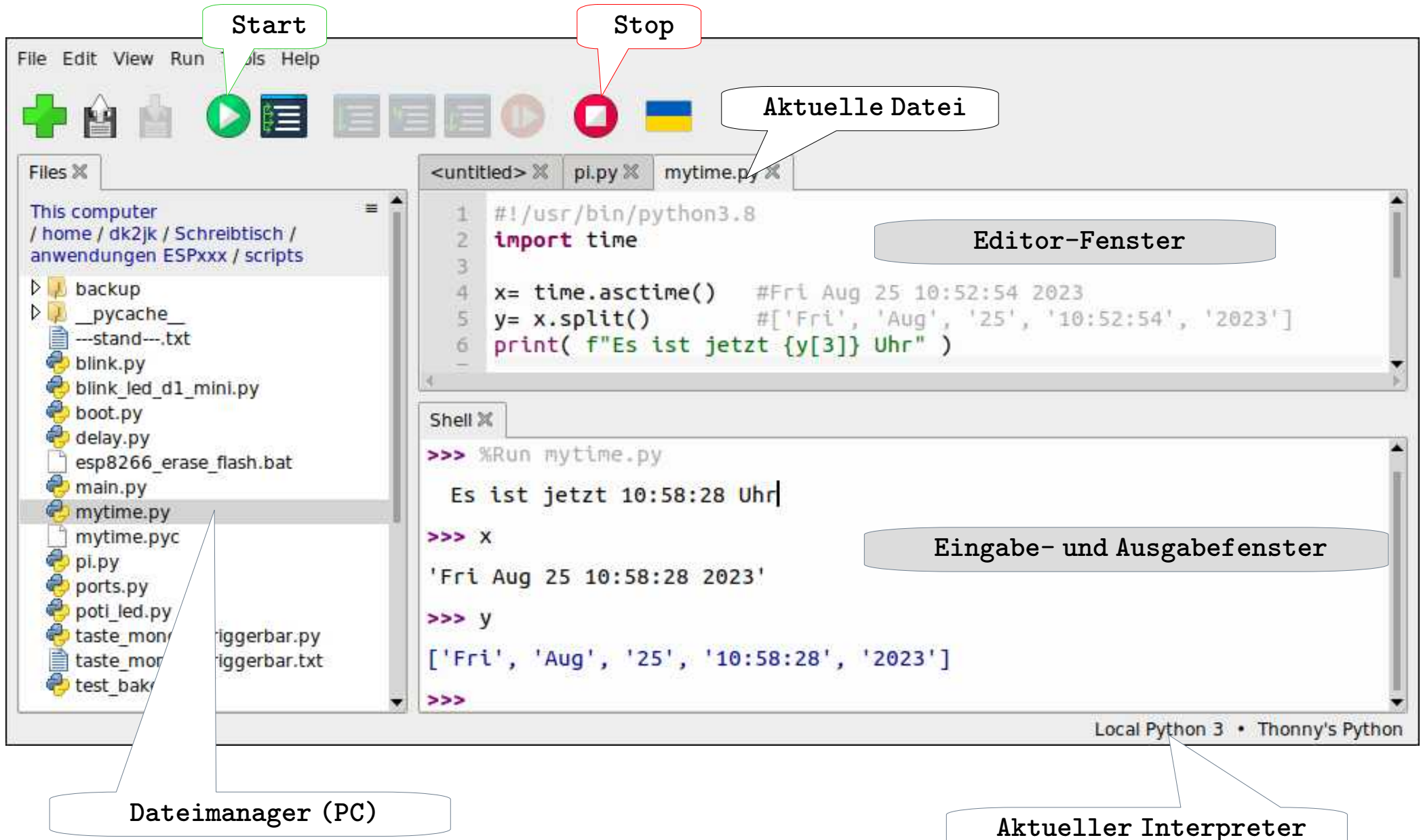
Download version [4.1.1](#) for
[Windows](#) • [Mac](#) • [Linux](#)

Download-Seite

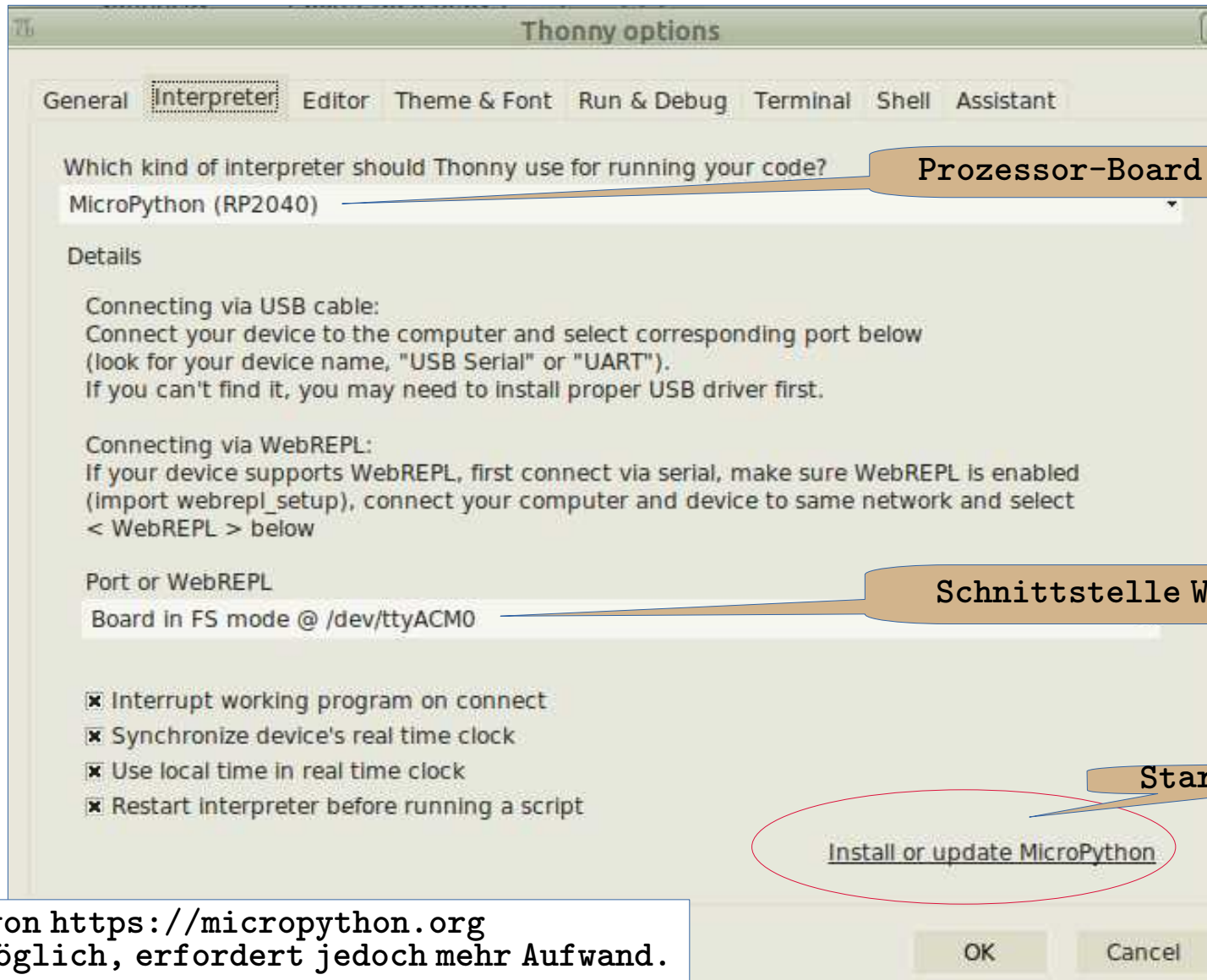


Desktop-Symbol

Entwicklungsumgebung ,Thonny' Ansicht



Flash Laden mit Thonny



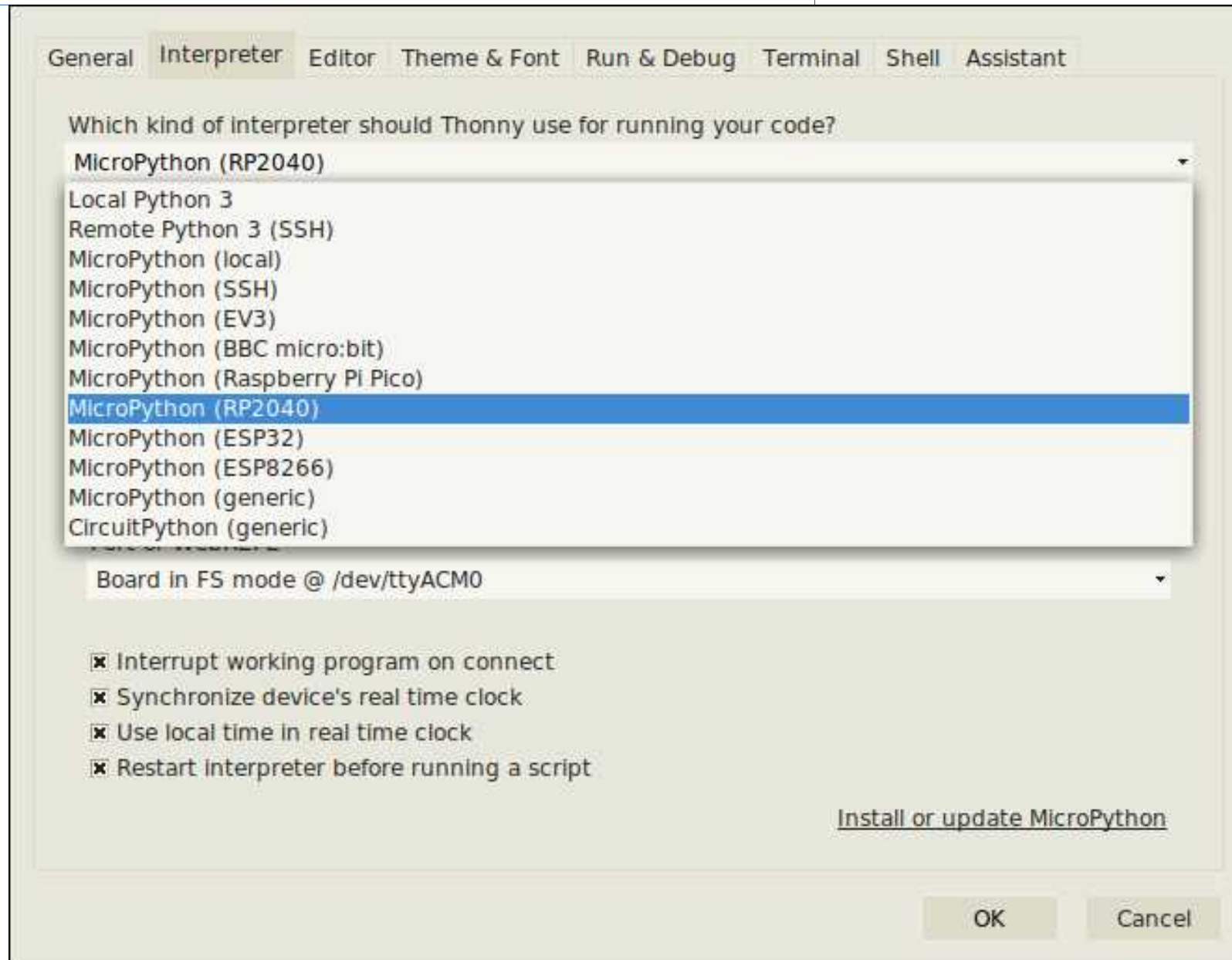
Prozessor-Board wählen

Schnittstelle Wählen

Start Installation

Download von <https://micropython.org>
ist auch möglich, erfordert jedoch mehr Aufwand.

Starten des Interpreters



„Micropython“ , Erster Start

(Hier mit dem Prozessor ESP8266)

The screenshot shows the Thonny IDE interface. The top menu bar includes File, View, Run, Tools, and Help. Below the menu is a toolbar with icons for file operations and execution. The left sidebar contains a file explorer with two sections: 'This computer' and 'MicroPython device'. The 'This computer' section shows a directory structure with files like pot1_led.py, test_bake.py, test_led.py, test_poti.py, test_taste-ton-led.py, and test_taste-ton.py. The 'MicroPython device' section shows a file named boot.py. The main area is divided into two panes. The top pane is a code editor titled '<untitled> x' with a single line of code '1'. The bottom pane is a shell window titled 'Shell x' showing the output of the MicroPython interpreter: 'MicroPython v1.15 on 2021-04-18; ESP module with ESP8266' and 'Type "help()" for more information.' followed by a prompt '>>>> |'. The status bar at the bottom right indicates 'MicroPython (ESP8266) • /dev/ttyUSB0'.

Dateien auf dem PC

Dateien auf dem MC

Interpreter Version auf dem MC

Eingestellter Interpreter

Serielle Schnittstelle

Vorführung: Micropython interaktiv'

```
Shell X
>>> print(hello)

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'hello' isn't defined

>>> print('hello')
hello
```

```
Shell X
>>> x=3.14
>>> y=2
>>> x*y
6.28
>>> 'A'*3
'AAA'
>>> |
```

MicroPython (ESP8266) • /dev/ttyUSB0

Das erste Script

```
[ pi.py ] X
1 x=3.14
2 y=2
3 print(x*y)

Shell X
>>> %Run -c $EDITOR_CONTENT
6.28
>>>
```

MicroPython (ESP8266) • /dev/ttyUSB0

Kapitel 4

1

Vorstellung einiger Boards

- ESP8266 , ESP32
- RaspberryPico RP2040

2

Python bzw. Micropython

- Ranking Programmiersprachen
- Vorführung: Python am PC

3

Entwicklungsoberfläche

- Installation von Thonny
- Einstellungen

4

Praktische Beispiele

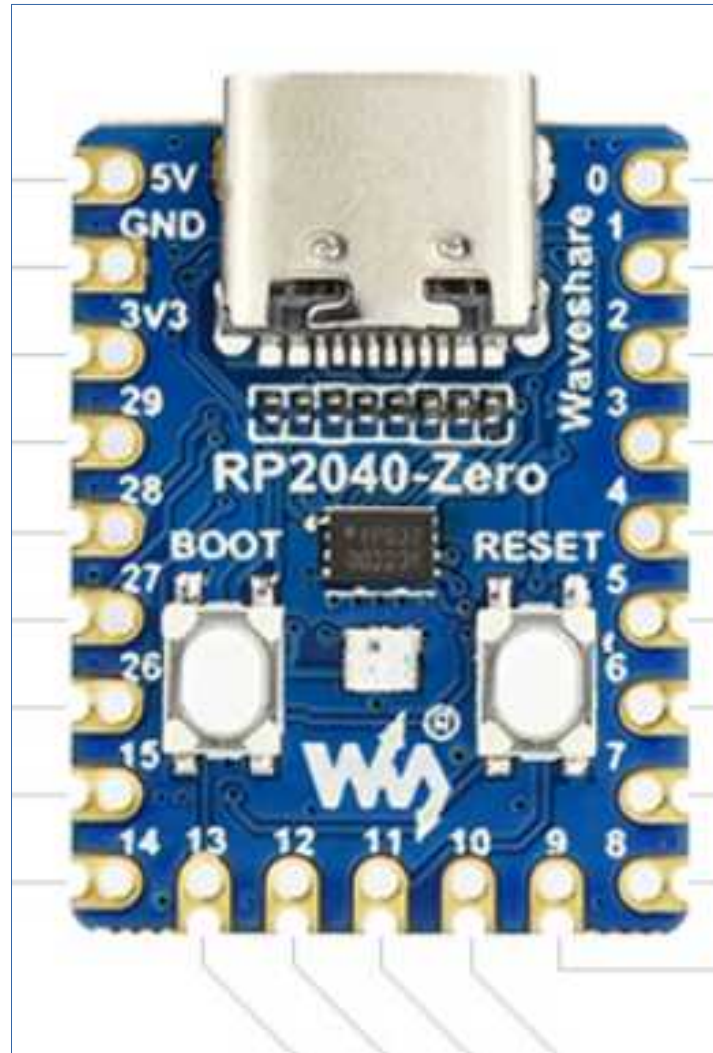
Zielobjekt: CW-Bake ,CW-Keyer

- Led / Blink- Led
- Autostart
- Taster + Led
- Poti am AD-Wandler
- Tonerzeugung / Taste + Ton

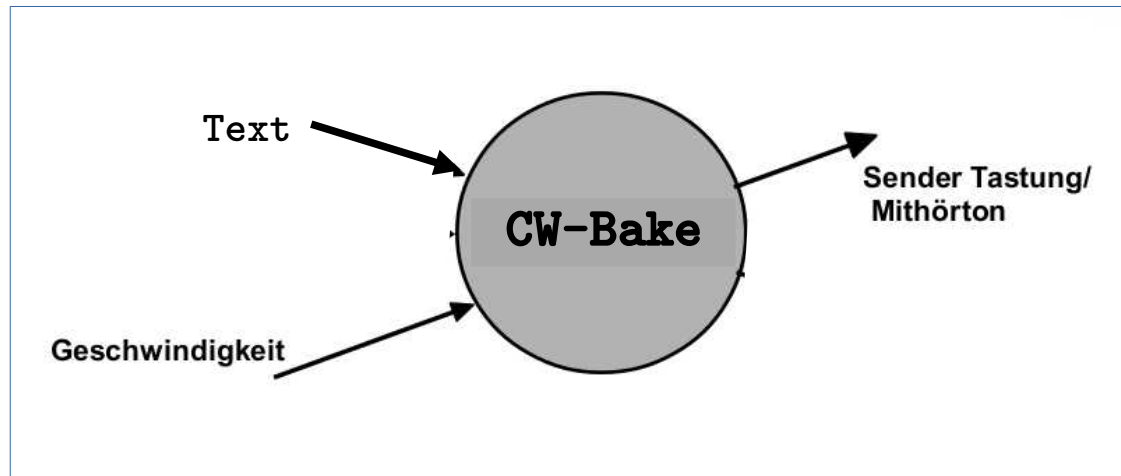
5

Verweise

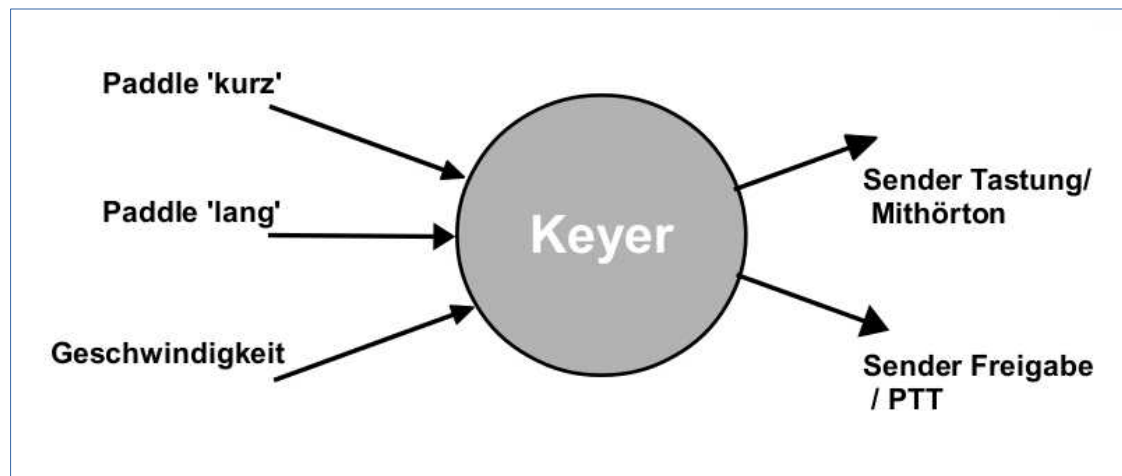
Das hier verwendete Board: RP2040-Zero



Anwendungsbeispiele : CW-Bake und CW-Keyer



CW-Bake
...Sendet vorgegebenen Text in CW



CW-Keyer
...Macht aus der Kombination der Paddle Eingänge Cw-zeichen

Benötigte Funktionen für ,keyer' bzw ,bake'

- Poti zum Einstellen der Morsegeschwindigkeit \rightarrow wpm ; $\text{tdit} = 1200 / \text{wpm}$
- Für Keyer: Paddle-Eingänge ,kurz' und ,lang'
- Für Bake: Text-Vorgabe , fest programmiert
- Für Bake: HF-Ausgang
- Schaltausgang, der den Sender tastet ,key'
- Option: Ausgang zum Freigeben des Tx / Sperren des Rx ,ptt'
- Option: Mithörton
- Zeitgeber für die Länge der Morse-Elemente (intern)

Schalten eines Ausgangs; Beispiel LED

Shell

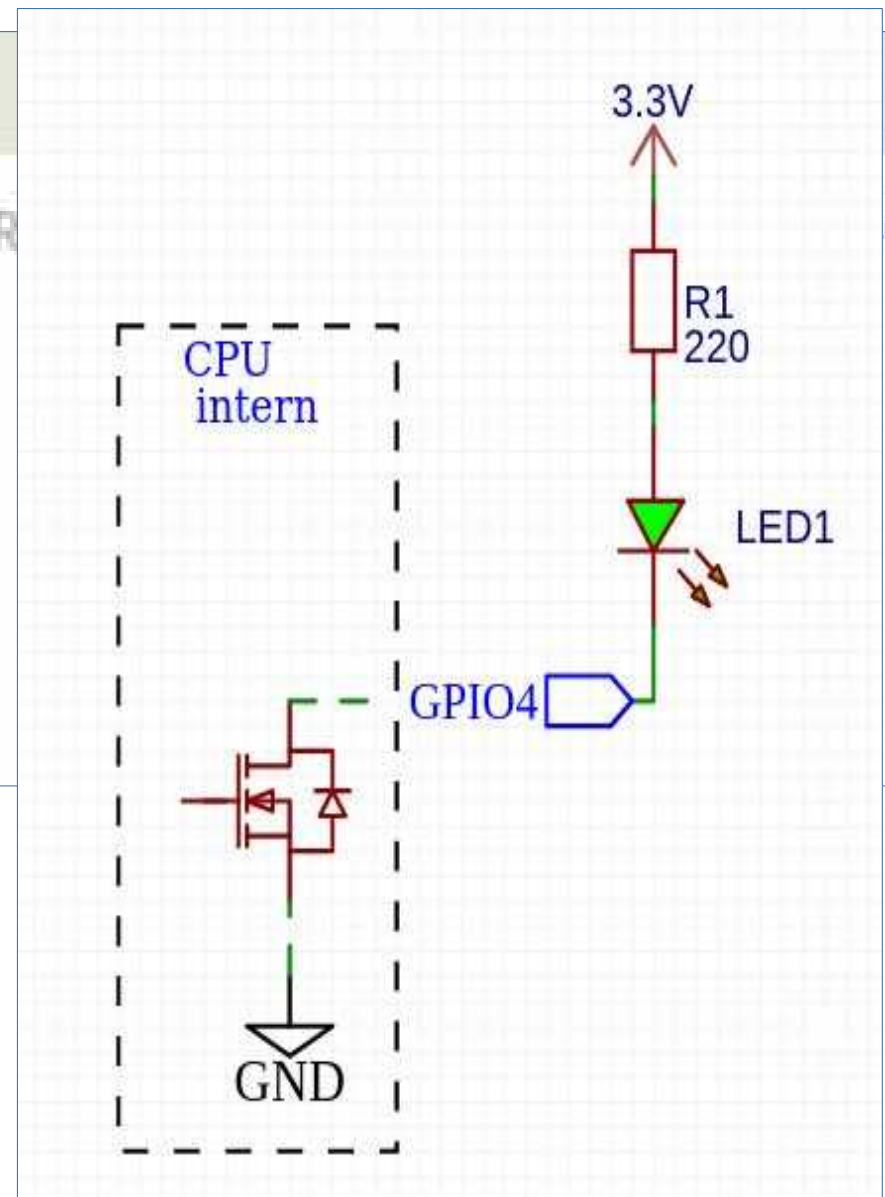
```
MicroPython v1.20.0 on 2023-04-26; R  
Type "help()" for more information.
```

```
>>> from machine import Pin  
>>> led = Pin( 7 ,Pin.OUT,value=1)  
>>> led.value(0)  
>>> led.value()
```

0

An : led.value(0)

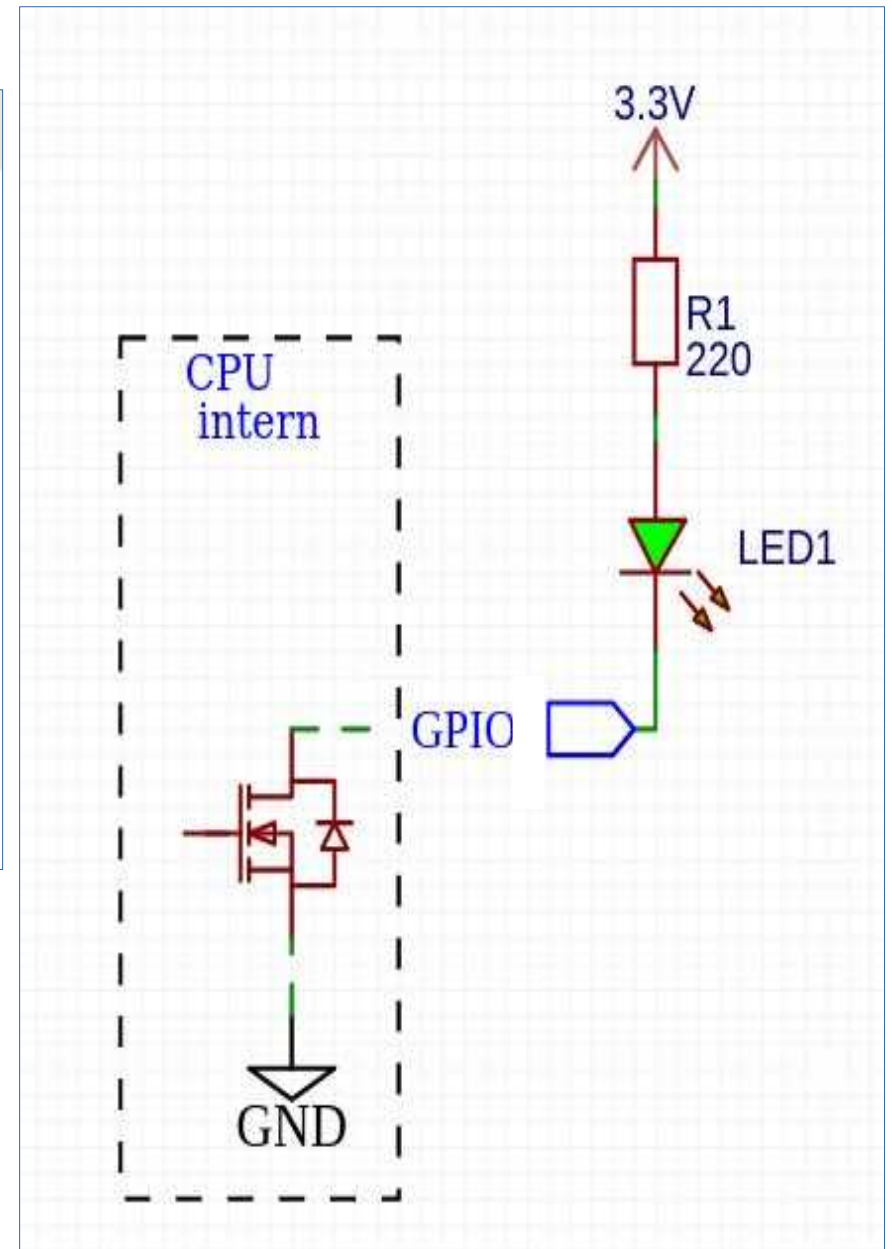
Aus: led.value(1)



Script Beispiel :Blink-Led

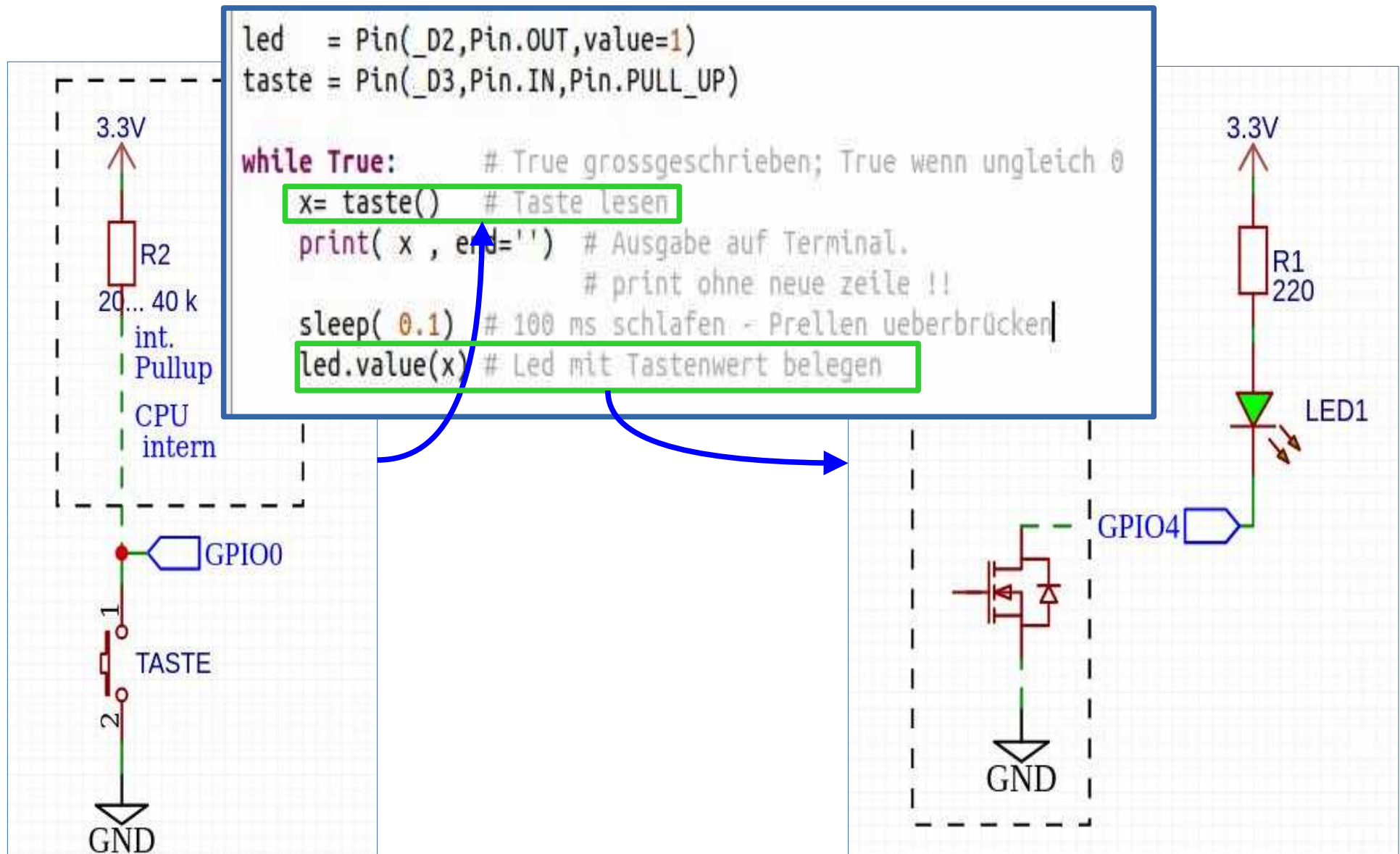
blink.py

```
1 from time import sleep
2 from machine import Pin
3 LED_BLAU_PIN = const(7)
4 led = Pin(LED_BLAU_PIN,Pin.OUT,value=1)
5
6 def blink():
7     while True:
8         x= not led()
9         led(x)
10        sleep(.33)
11
12 print("blink Led")
13 blink()
```



Abfragen eines Eingangs: Taster + Led

Script:
test_taste_led.py



Autostart am Beispiel ,Blink- Programm‘

'boot.py'|
'main.py'

```
[ boot.py ] *X  
# boot.py  
import gc  
gc.collect()
```

2 besondere Programme:

,boot.py' startet beim
Einschalten automatisch.

,main.py' wird nach
,boot.py' gestartet

In ,main.py' kann meine
Anwendung laufen oder ein
anderes Modul importiert
werden


```
[ test_led.py ] X  
from machine import Pin  
from time import sleep  
  
_D2= const(4)  
led = Pin(_D2,Pin.OUT,value=1)  
  
def blink():  
    while True:  
        led( not led() )  
        sleep(.5)
```

```
[ main.py ] X  
from test_led import blink  
blink()
```


Ausführen!

Poti am AD-Wandler

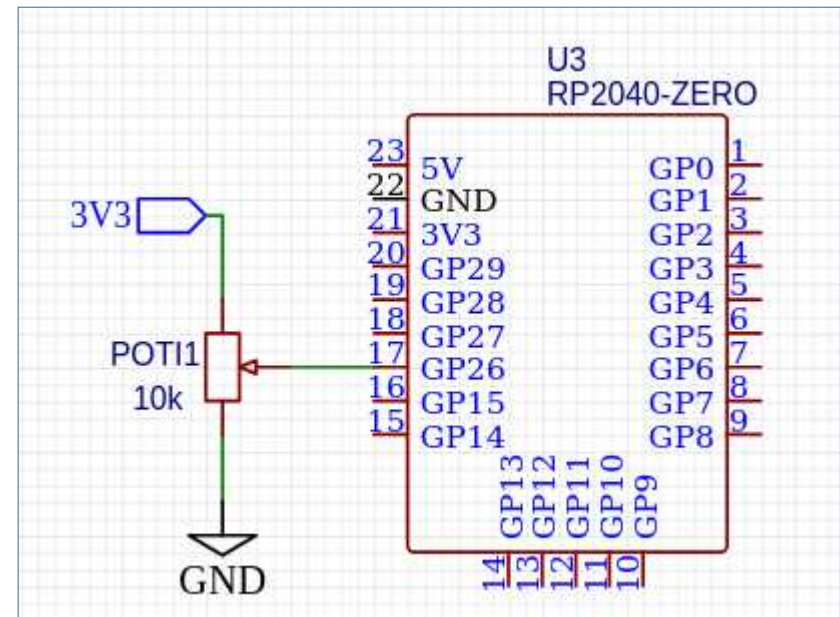
Script:
test_poti.py

<untitled> *

```
1 from machine import Pin,ADC
2
3 adc  = ADC(Pin(26)) #rp2040
4
5 import time
6 while True:
7     x= adc.read_u16()
8     print(x)
9     time.sleep(3)
```

Shell 

```
MPY: soft reboot
80
2256
9442
15619
22149
65535
65535
```



Tonerzeugung

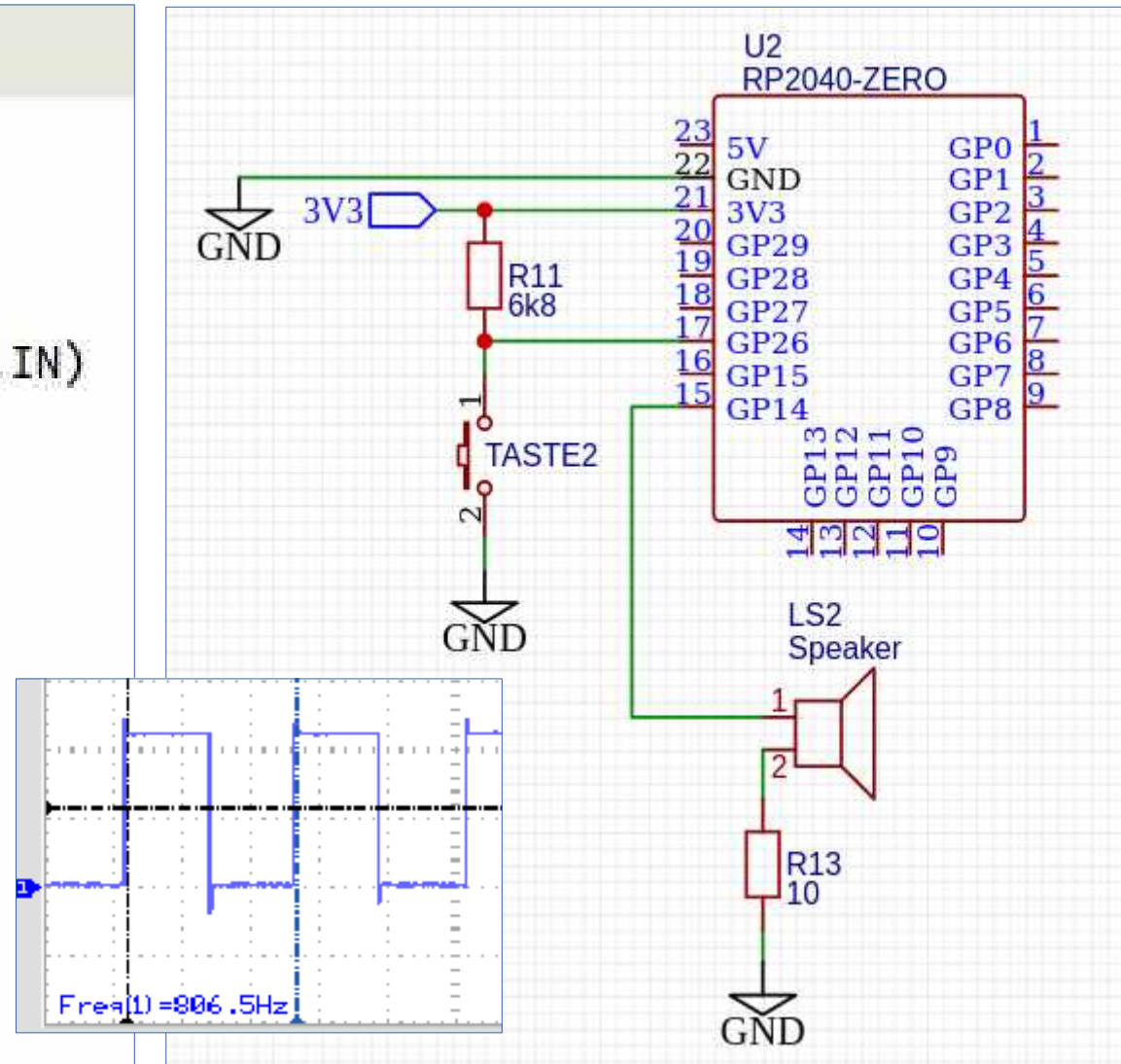
Script:
taste_ton.py

[taste_ton.py] *

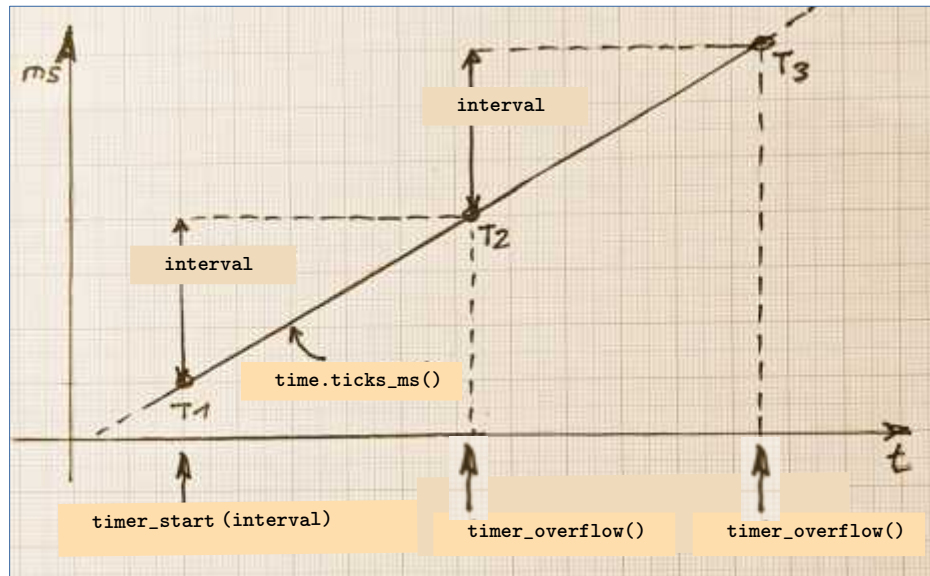
```

1  #aus unserem baukasten:
2  from ports_rp2040 import *
3  from machine import Pin,PWM
4
5  tunetaste = Pin(TUNE_PIN,Pin.IN)
6
7  # mithoerton
8  ton = PWM ( Pin(TON_PIN))
9  ton.freq(600)
10 ton.duty_u16(0)
11
12 while True:
13     if tunetaste()==0:
14         ton.duty_u16(32767)
15     else:
16         ton.duty_u16(0)
17     #sleep(.01)

```



Zeitgeber für die Länge der Morse-Elemente



`time.ticks_ms() = Systemtimer`

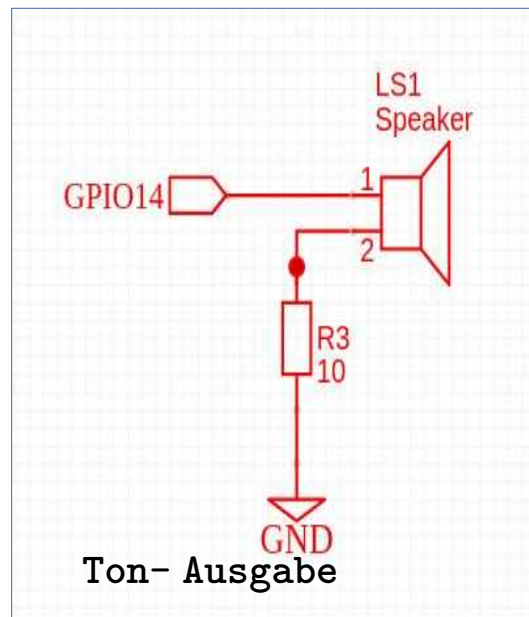
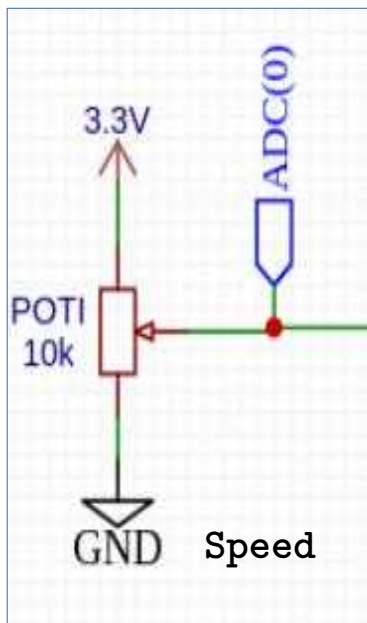
```
[ timer_demo.py ] *  
1  import time  
2  
3  interval=0  
4  tn=0  
5  
6  def timer_start(_interval):  
7      global tn, interval  
8      interval=_interval  
9      tn= time.ticks_ms()  
10  
11  def timer_overflow():  
12      global tn  
13      if time.ticks_ms() > tn:  
14          tn=tn+interval  
15          return True  
16      else:  
17          return False
```

Realisierung: CW- Bake

Aufgabe: Text als Morsezeichen im Lautsprecher ausgeben.

Wir haben :

- Ein Poti zum Einstellen der Morsegeschwindigkeit (ADC)
- Einen Tongenerator (PWM)
- Einen Lautsprecher-Ausgang



Wir benötigen noch:

- eine Liste der Morsezeichen in der Form:
Tabelle = {
 '!': '-.-.-',
 '1': '.----', etc.
- Eine Programm-Schleife, die den Text Buchstabe für Buchstabe nacheinander liest
- Einen Übersetzer „Buchstabe in Morsecode“:
 Aus ‚a‘ wird „.-“
- Einen Übersetzer der den Punkt-Strich-Morsecode in Töne wandelt:
 Aus „.-“ wird hörbar dit dah

Programm Schnipsel: CW- Bake

Script:
cw_bake.py

```
'test' --> liste ['-','.', '...', '-']
```

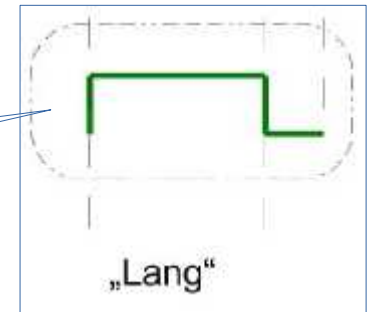
```
if element=='.':
```

```
    ton_an()  
    warte_millisekunden(dit_time_ms)  
    ton_aus()  
    warte_millisekunden(dit_time_ms)
```

```
if element=='-':
```

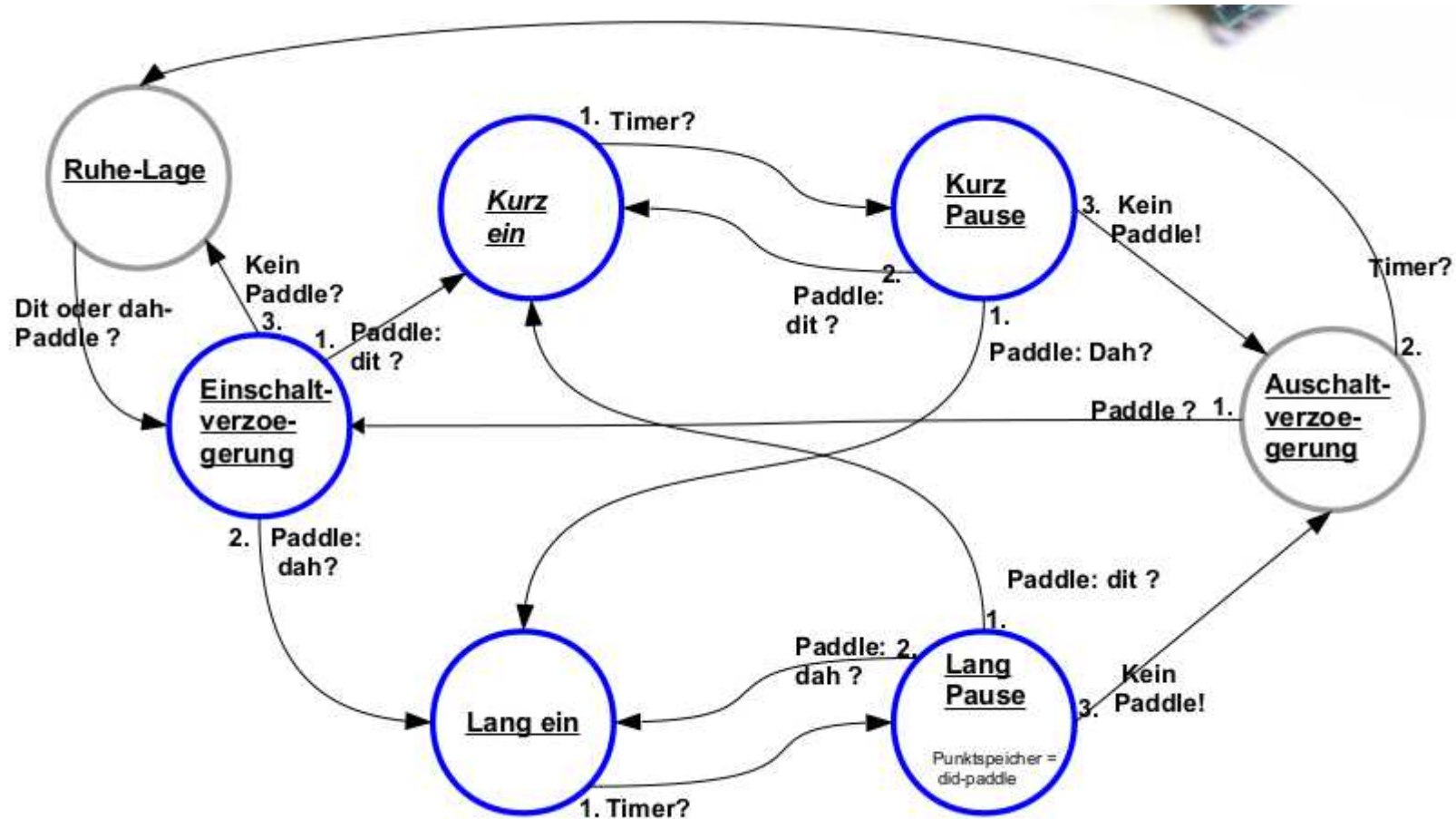
```
    ton_an()  
    warte_millisekunden(dit_time_ms*3)  
    ton_aus()  
    warte_millisekunden(dit_time_ms)
```

```
warte_millisekunden(dit_time_ms*2)
```

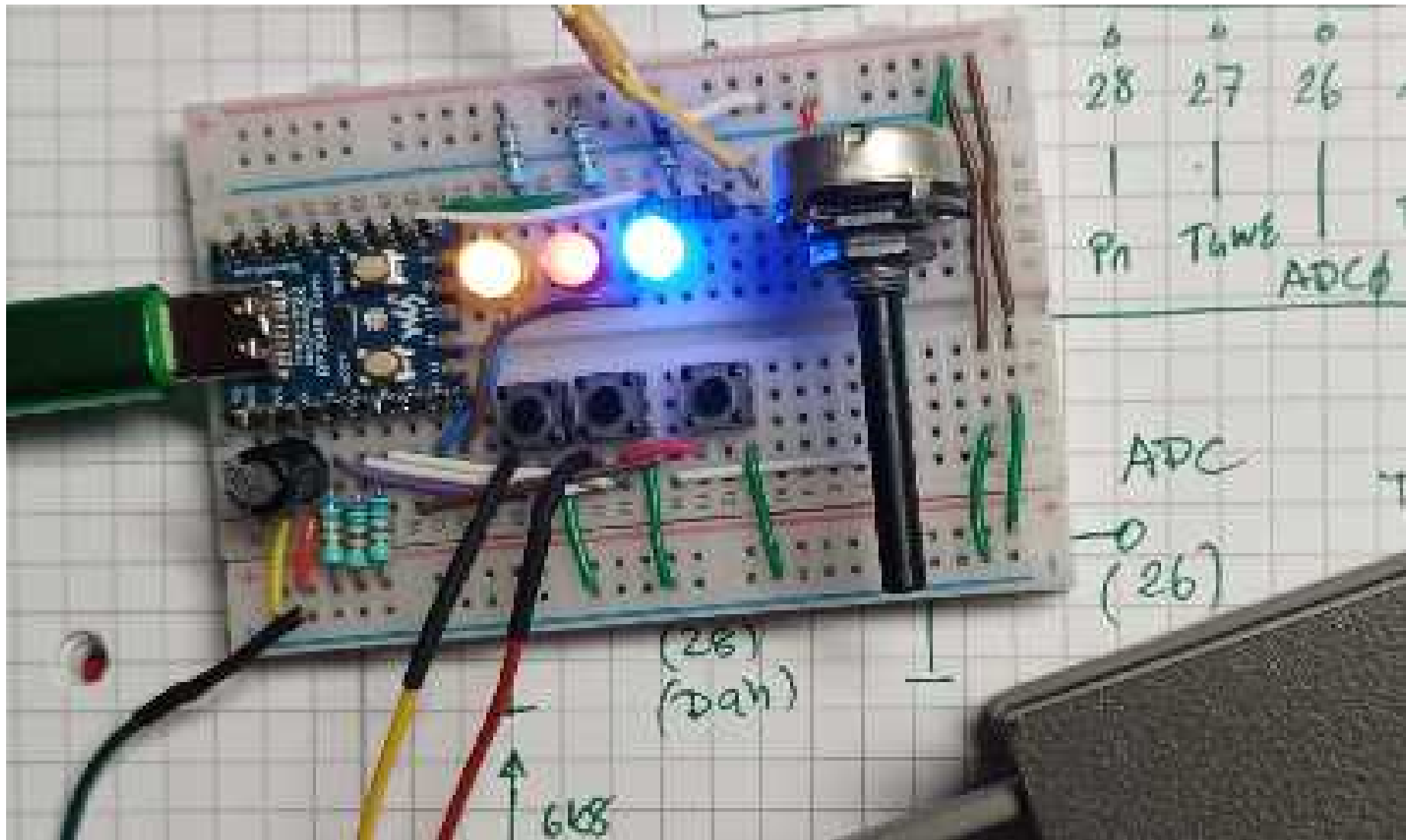


Buchstaben
Abstand

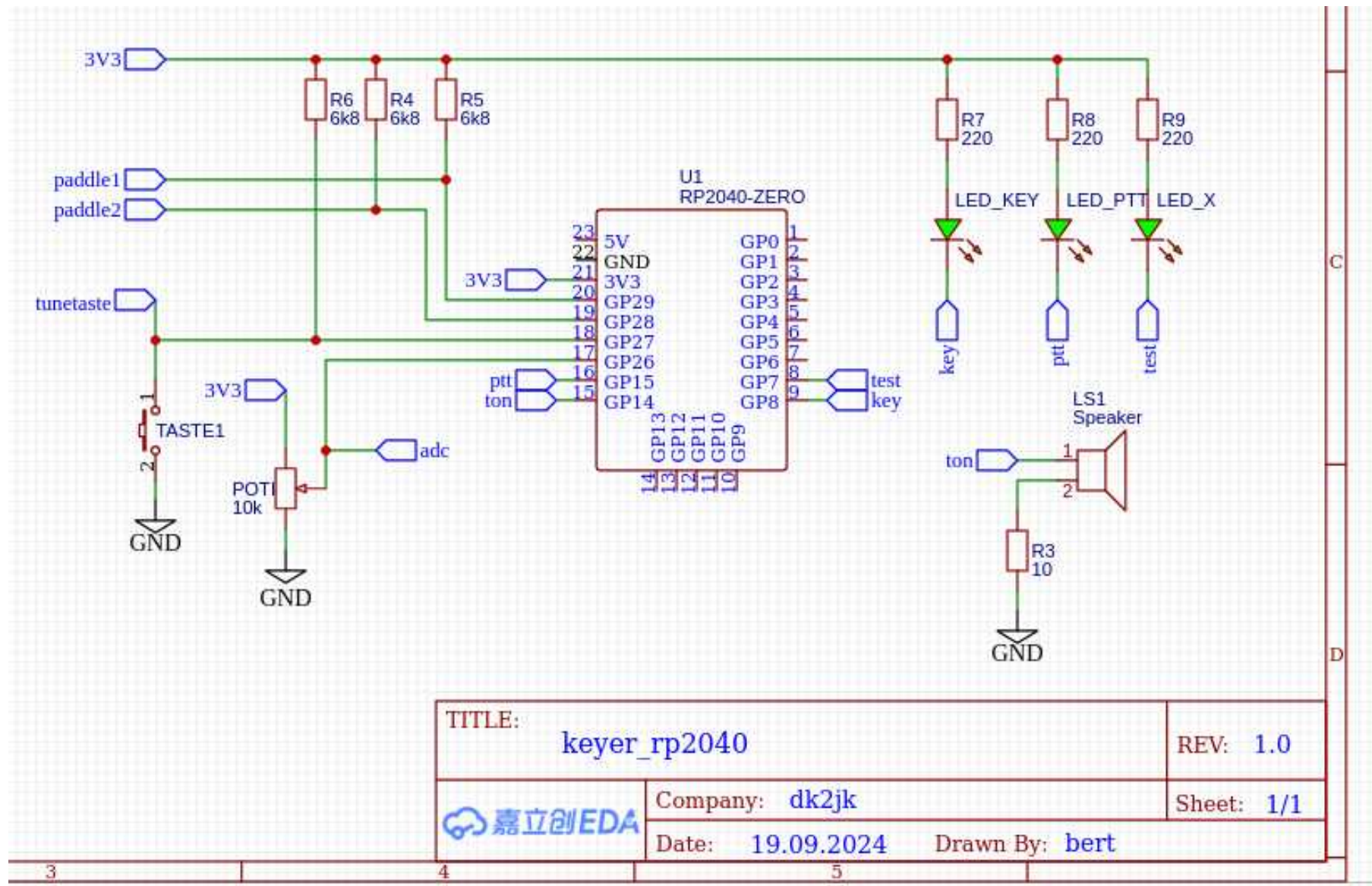
Keyer Zustandsdiagramm („state chart“)



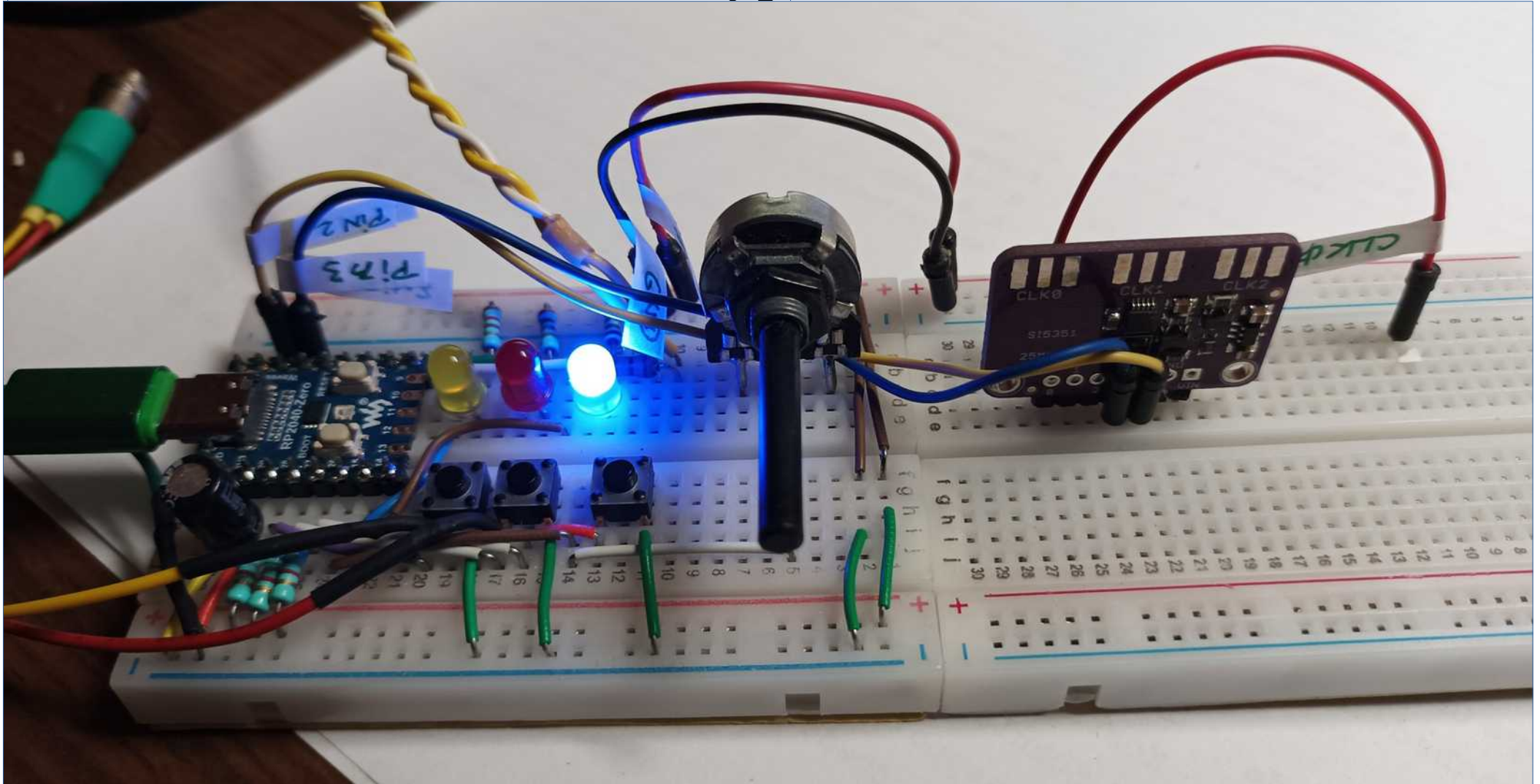
Keyer Prototyp



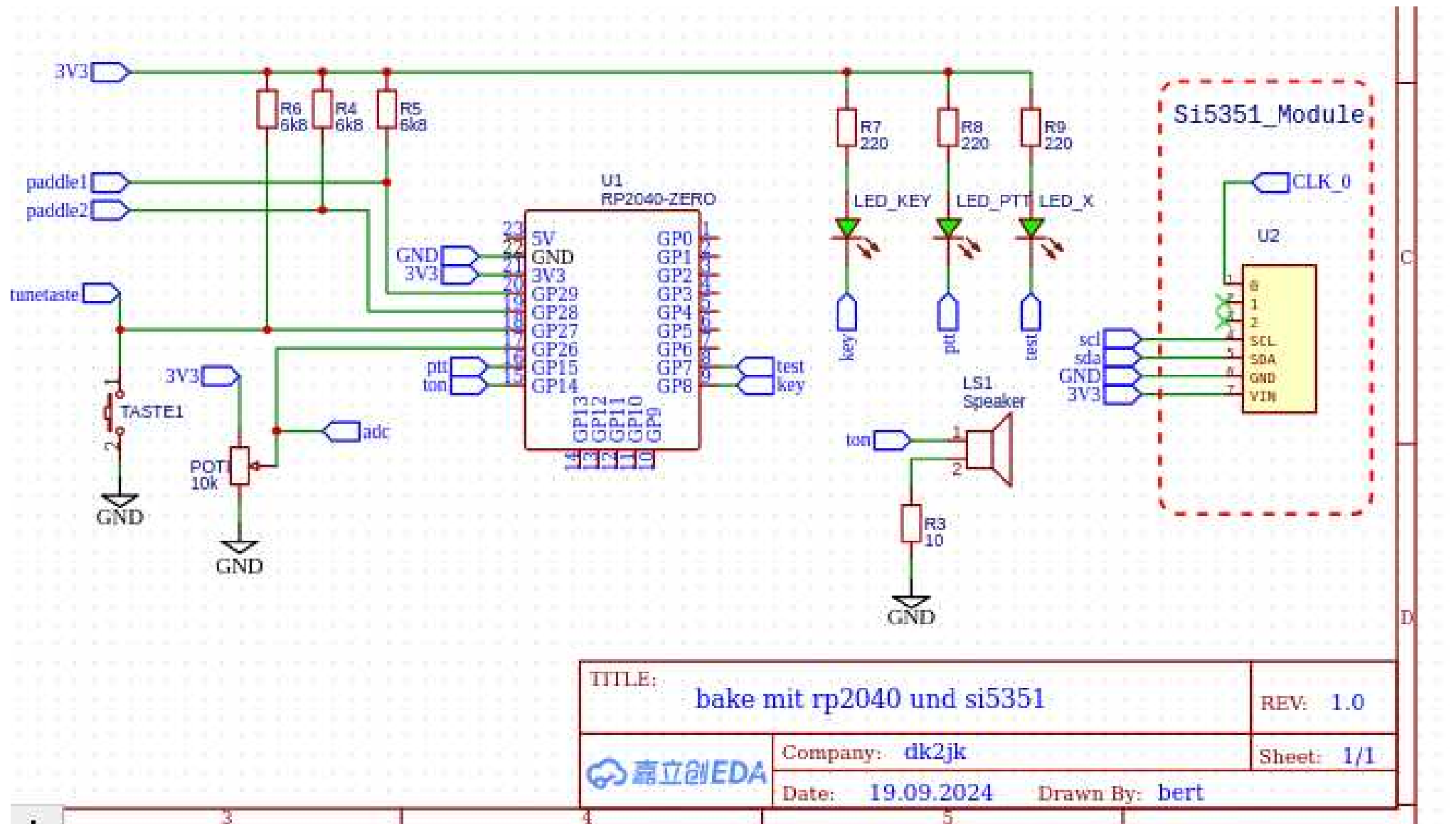
Keyer Schaltbild



Bake mit Si5351, Prototyp



Bake mit Si5351, Schaltbild



5 Verweise

- [1] Python für Windows / Linux:
<https://www.python.org/downloads/>
- [2] Micropython für verschiedene CPUs:
<https://micropython.org/>
- [3] Entwicklungsumgebung für Python ,Thonny‘ :
<https://thonny.org/>
- [4] „Das umfassende Handbuch“ :
<https://openbook.rheinwerk-verlag.de/python/>
- [5] Quick reference for the ESP8266 (englisch)
<https://docs.micropython.org/en/latest/esp8266/quickref.html>
- [6] Quick reference for RP2040 (englisch)
<https://docs.micropython.org/en/latest/rp2/quickref.html>
- [7] Dieses Projekt
https://github.com/dk2jk/anwendungen_micropython

