circpy-rpipico

Wednesday 14 January 2026 3 versions

1. CircuitPython with Raspberry-Pi-Pico and NeoPixel Matrices

1.1. Links to Online Resources

https://**agmc26-mfhn.dortoka.ipv64.de/**

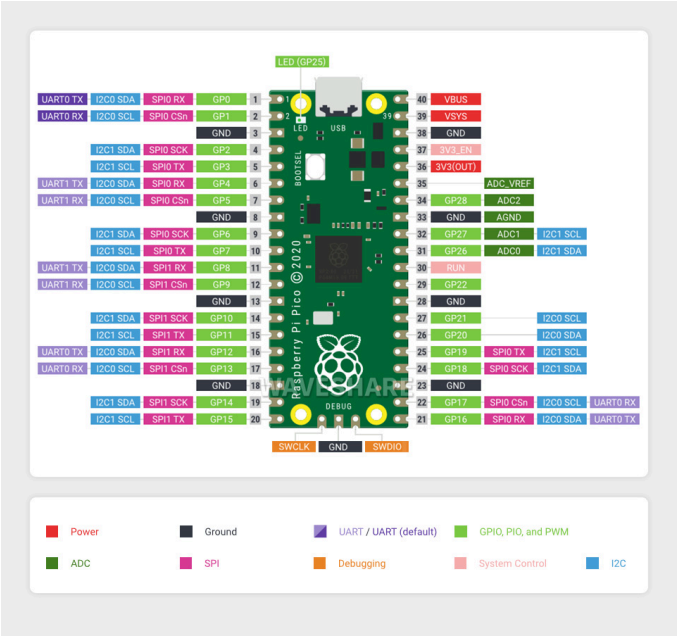
2. Why do we use these parts?

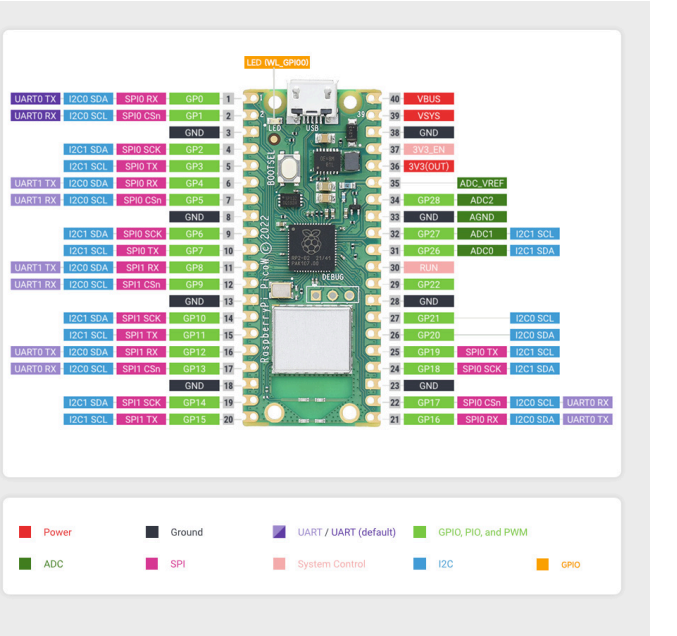
2.1. Micro-Controllers

Raspberry-Pi-**Pico**:

- boards are cheap and processors are fast
- supported by many languages (C, Arduino, µPython, Rust, ...)
- Picos have a large eco system:
  - good documentation
  - used by many hobbyist

Model	Processor	RAM	Flash	Cores	Clock speed	WLAN
Pico	RP2040	264KB	2MB	2x ARM Cortex-M0+	133MHz	no
Pico-W	RP2040	264KB	2MB	2x ARM Cortex-M0+	133MHz	yes
Pico2	RP2350	520KB	4MB	2x ARM Cortex-M33	150MHz	no
Pico2-W	RP2350	520KB	4MB	2x ARM Cortex-M33	150MHz	yes





2.2. CircuitPython vs. MicroPython

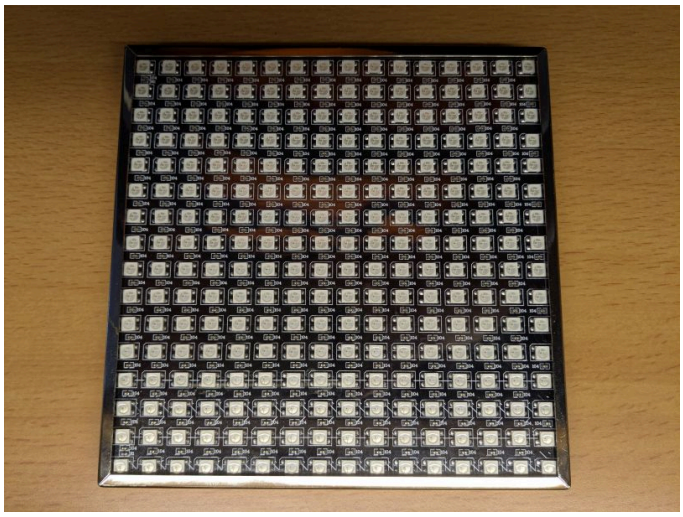
MicroPython (**MPy**) and CircuitPython (**CPy**) are closely related.
The **Adafruit** company is sponsoring **CPy**, which is a derivative of **MPy**.

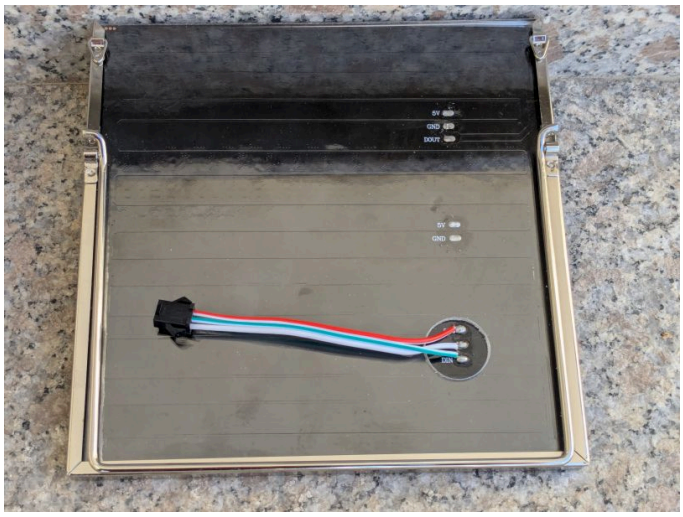
- The **CPy** interpreter uses only a small part of the on-board Flash memory
  - the other part is used as a FAT filesystem, which is
  - visible as an USB flash drive, named **CIRCUITPY**
- **CPy** has a *very* large eco system of libraries (with documentation and examples)
  - CircuitPython documentation
  - Adafruit libraries
  - Community libraries
- Interpreter:
  - “fast enough”: usually no need for compiled programs
  - fast turn-around-times during development
  - **CPy** is “*FUN*” to use

2.3. 16×16 NeoPixel-Matrix

“NeoPixel” is a term created by Adafruit for “smart” RGB-LEDs.
They are usually based on **WS2812B** chips.
See the “NeoPixel ÜberGuide” for a detailed explanation.

- all LEDs are pre-soldered to a flexible PCB: size is only 16cm by 16cm
- all 256 “NeoPixels” are chained together (in a zigzag style)
- NeoPixels can be driven with only 3 pins: +5V, GND and *Data*
- NeoMatrices have an “affordable” price tag





All resources can be be downloaded from the `assets` directory, as well.

2.4. Optional Enhancements

- **IKEA** picture frames
  - 16cm x 16cm LERBODA
  - 25cm x 25cm SANNAHED
  - 32cm x 32cm LOMVIKEN
- **Hornbach**
  - Diffusors made from “Grey Acrylic Glas”

3. How do you start?

3.1. Select a micro-controller Board

- without WiFi and Bluetooth
  - RPi-Pico, RPi-Pico2
  - VCC-GND YD-RP2040
  - several boards by **WaveShare**
- with WiFi and BLE
  - RPi-Pico-W, RPi-Pico2-W
  - several boards by **Pimoroni**

3.2. Download the matching Firmware

... from Adafruit's CircuitPython "store" at https://circuitpython.org/down...
... or from the `assets` directory.

1. select your board
2. download the "latest" Firmware: "adafruit-circuitpython-*manufacturer-boardname-version*.UF2"
3. and save it to your PC

3.3. Install this Firmware to your Board

Read the installation notes in the Adafruit learn guide .

- Flash procedure:

1. insert USB cable into Board (but leave PC side unconnected!)
2. press BOOT button on Board
3. next, insert USB cable into your PC (and still keep the BOOT button pressed)
4. now, release the BOOT button
5. a new USB drive, called "**RPI-RP2**", appears ("**RP2350**" for Pico2)
6. copy the appropriate ".UF2" file to the USB drive "**RPI-RP2**"
7. the Board reboots and USB drive "**RPI-RP2**" disappears
8. a new USB drive, called "**CIRCUITPY**", appears
9. **Finished!**

3.4. Install additional Tools on your PC

1. install an Editor or IDE:
   - Thonny IDE
   - MU Editor
2. install Adafruit's **CIRCUP** tool to manage the Adafruit and Community libraries on your Board
3. use your PC's Filemanager to explore and manage the files on the **CIRCUITPY** drive

3.5. Addon: Installation of MicroPython

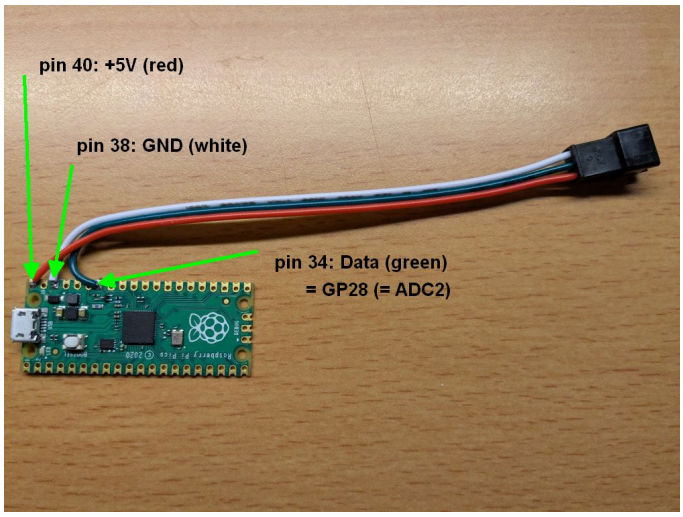
Either get the firmware online or from the `assets` directory.
Then follow the installation instructions in the quick-ref guide .

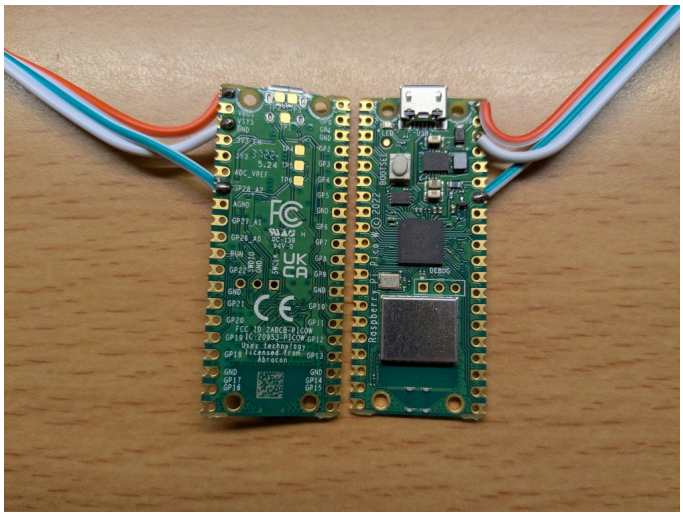
---

4. What do you need?

Connect a NeoPixel cable to the Rapsberry-Pi-Pico:

- Solder a 3-wire cable to pins 34, 38, and 40 of your **Pico** board.
  The other end of this cable has a plug, which fits to the NeoPixel matrix.
  (Detailed view of “Pico-W” boards with NeoPixel cables.)





4.1. Hardware

What	Price
Pico board	5-10 €
USB cable	5 €
16×16 NeoPixel matrix	25-30 €
3 pin Data cable	included with matrix
Sub total	approx. 40 €
IKEA frame	10 €
acrylic diffusor	10 €
Grand total	< 60€

4.2. Time & Tools

- soldering iron + some solder
- ruler + cutter knife
- cordless drill + some drill inserts (3-12 mm)
- approximately 1-2 hours

5. Short Introduction to Programming with Python

If you are new to programming with Python then you might have a look at the "Welcome To CircuitPython" guide by **Adafruit**. The chapter "How do I learn Python?" has links to guides for every level of experience.

Another guide "Getting Started with Raspberry Pi Pico and CircuitPython" is dedicated to programming **Pico** boards with **CPy**. The chapter "NeoPixel LEDs" is very helpful for this workshop.

These guides can also be downloaded from the `assets` directory.

5.1. Structure of a CircuitPython program

1. import needed libraries (and methods)
2. define your global variables (and constants)
3. define (aka. develop) your own functions
4. initialize all needed objects
5. start the main loop

5.2. Control structures

Python **does not** use any explicit syntactic brackets (like `begin` and `end`, or `{` and `}`) to delimit compound statements.
That's why *indentation* (usually by 4 spaces) is used to group statements.

Python does have most of the usual control structures.
But you need to terminate the 'test' with a colon `:`.

```
# IF statement
if (a == b):
    x = y
elif (b == c):
    x = z
else:
    z = x +y

# WHILE statement
i = 0
while (i < 10):
    i += 1

# FOR statement
for j in range(10):
    print(j)
```

5.3. Function definitions

The same is true for function definitions.
Put the name and all parameters onto the first line and terminate the line with a colon `:`.
Then you need to *indent* all statements of the function body.
The function definition ends, when you *outdent* to the level of the `def` keyword.

```
# Function defintion
def fname(x, y, z):
    return x * y * z

print( fname(2, 3, 5) )
```

5.4. Data types

Python does have all the usual data types, like integers, floats and strings.
Some conversions are done implicitly. Others need to be carried out explicitly by calling a function.

```
a, b = 5, 7
x, y = 9.7, 6.5
z = float( a + b )
c = int( x / y )
s = "my "
t = " know"
r = "ledge"
print(z, c, s+t+r)
```

5.5. Data structures

The basic data structures of Python are **lists**, **dictionaries**, and **tuples**:

```
# LISTS have indices that are integers and can be used as ARRAYS
# delimiters are [ and ]
a = [ 1, 2, 3, ]
rgb = [ 0xff, 0xcc, 0xd8 ]
b = [ 'a', 12, 3.14, [], rgb ]
print( a[1], b[4] )
```

```
# DICTIONARIES are similar to LISTS,
# except the indices are strings (or any other type)
# delimiters are { and }
word = {}
word['en'] = 'book'
word['de'] = 'buch'
```

```
word['fr'] = 'livre'
words = { 'one': 'eins', 'two': 'due', 'three': 'trois', }
```

```
# TUPLES are an immutable sequence of values
# delimiters are ( and )
t = ( 1, 2, 3, 4, 5 )
print( t[2] )
```

6. A few Examples

All Examples can also be downloaded from the `assets` directory.

6.1. Hello world!

6.1.1. Blink

```
# SPDX-FileCopyrightText: 2025 Pagong
# SPDX-License-Identifier: MIT

import time
import board
import neopixel

########################

# for Rpi-Pico with 16x16 NeoPixel-Matrix
NUM_COLS = 16
NUM_CELLS = 16
NUM_PIXELS = (NUM_COLS * NUM_CELLS)  # Update this to match the number of LEDs.

SPEED = 0.1        # Increase to slow down the effect. Decrease to speed it up.
BRIGHTNESS = 0.1   # A number between 0.0 and 1.0, where 0.0 is off, and 1.0 is max.

PIN = board.GP28   # This is the default pin on my RPi-Pico with 16x16 NeoPixel matrix
pixels = neopixel.NeoPixel(PIN, NUM_PIXELS, brightness=BRIGHTNESS, auto_write=False)

######################

black = 0
color = ( 0xff, 0xcc, 0xd8 )

while True:
    pixels.fill(black)
    pixels.show()
    time.sleep(5*SPEED)
```

```
    pixels.fill(color)
    pixels.show()
    time.sleep(SPEED)
```

6.1.2. Rainbow

```
# SPDX-FileCopyrightText: 2025 Pagong
# SPDX-License-Identifier: MIT

import time
import board
import neopixel
import rainbowio

########################

# for Rpi-Pico with 16x16 NeoPixel-Matrix
NUM_COLS = 16
NUM_CELLS = 16
NUM_PIXELS = (NUM_COLS * NUM_CELLS)  # Update this to match the number of LEDs.

SPEED = 0.01       # Increase to slow down the effect. Decrease to speed it up.
BRIGHTNESS = 0.1   # A number between 0.0 and 1.0, where 0.0 is off, and 1.0 is max.

PIN = board.GP28   # This is the default pin on my RPi-Pico with 16x16 NeoPixel matrix
pixels = neopixel.NeoPixel(PIN, NUM_PIXELS, brightness=BRIGHTNESS, auto_write=False)

######################

black = 0

while True:
    pixels.fill(black)
    pixels.show()
    time.sleep(50*SPEED)

    for i in range(NUM_PIXELS):
        for j in range(NUM_PIXELS):
            color = rainbowio.colorwheel(i+j)
            pixels[j] = color
        pixels.show()
        time.sleep(SPEED)
```

6.2. 2D Matrices

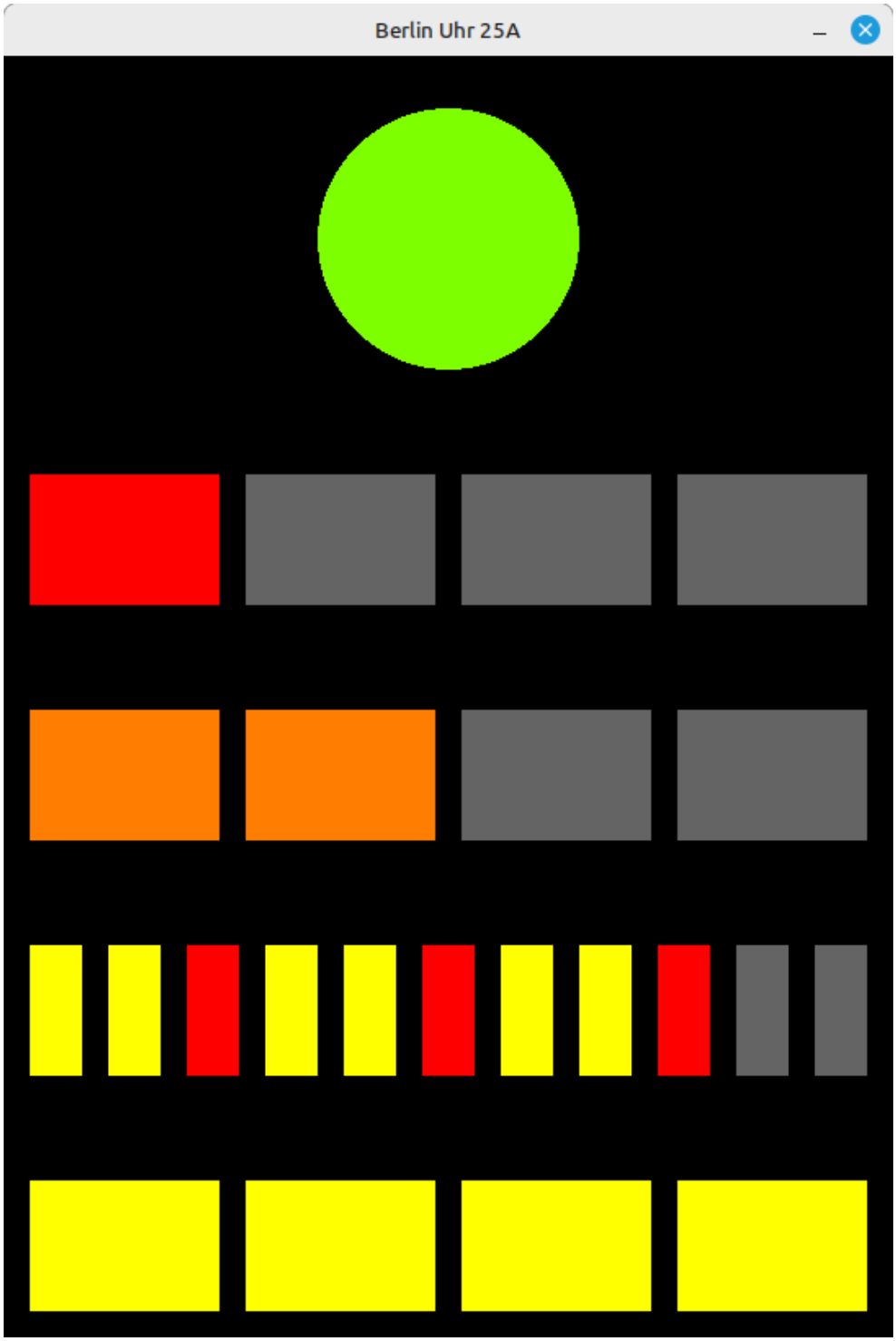
6.2.1. Sinus

```
# move through Sinus and Cosinus terrain
#
# 21 Mar 2025 - @pagong
# Uses Raspberry-Pi Pico with a 16x16 NeoPixel LED matrix

import time
import board
import random
import neopixel
import rainbowio

import neomatrix

######################

# for RPi-Pico with 16x16 NeoPixel-Matrix
NUM_COLS  = 16
NUM_CELLS = 16

NUM_PIXELS = (NUM_COLS * NUM_CELLS)  # Update this to match the number of LEDs.
SPEED = 0.01       # Increase to slow down the animation. Decrease to speed it up.
BRIGHTNESS = 0.1   # A number between 0.0 and 1.0, where 0.0 is off, and 1.0 is max.
PIN = board.GP28   # This is the default pin on RPi-Pico with 16x16 NeoPixel matrix

leds = neopixel.NeoPixel(PIN, NUM_PIXELS, brightness=BRIGHTNESS,
                         pixel_order=neopixel.GRB, auto_write=False)

matrixType = (  neomatrix.NEO_MATRIX_BOTTOM + neomatrix.NEO_MATRIX_LEFT +
                neomatrix.NEO_MATRIX_ROWS + neomatrix.NEO_MATRIX_ZIGZAG   )

matrix = neomatrix.NeoMatrix(
    leds,
    NUM_COLS, NUM_CELLS,
    1, 1,
    matrixType,
)

grid = matrix._grid

######################

# prepare rainbow palette
palette = []
for k in range(256):
    palette.append(rainbowio.colorwheel(k))

# change direction of movement
def change_direction():
    xs, ys = 0, 0
    while (abs(xs) + abs(ys) == 0):
        xs = random.randint(-1, 1)
        ys = random.randint(-1, 1)
    return float(xs), float(ys)
```

```
def do_frame():
    for i in range(NUM_COLS):           # for each pixel row
        sinx = math.sin(start_x + step*i)
        pxl = grid[i]
        for j in range(NUM_CELLS):      # for each pixel column
            cosy = math.cos(start_y + step*j)
            val = 1.0 + (sinx * cosy)
            col = int(val * 127.5)      # scale it from -1 - +1 -> 0 - 255
            pxl[j] = palette[col]       # convert hue to rainbow color

####################

import math
step = (1.1 * math.pi) / float(NUM_COLS)
start_x = 0.0
start_y = 0.0

incr = 0.1
xsign = 0.0
ysign = 1.0

Debug = True

while True:
    t1 = time.monotonic_ns()
    do_frame()
    t2 = time.monotonic_ns()

    grid.show()
    t3 = time.monotonic_ns()

    if Debug:
        d1 = (t2 - t1) / 1000000.0
        print(f"Compute {d1} ms", end=" +\t")
        d2 = (t3 - t2) / 1000000.0
        print(f"Display {d2} ms", end=" =\t")
        print(f"Total {d1+d2} ms", end=" -->\t")
        print(f"{1000.0/(d1+d2)} fps")

    # move around in 2D space
    start_x += incr * xsign
    start_y += incr * ysign
    if (random.randint(0, 99) == 8):
        xsign, ysign = change_direction()

    time.sleep(SPEED)
```

6.2.2. Berlin-Uhr

Program code of Berlin-Uhr

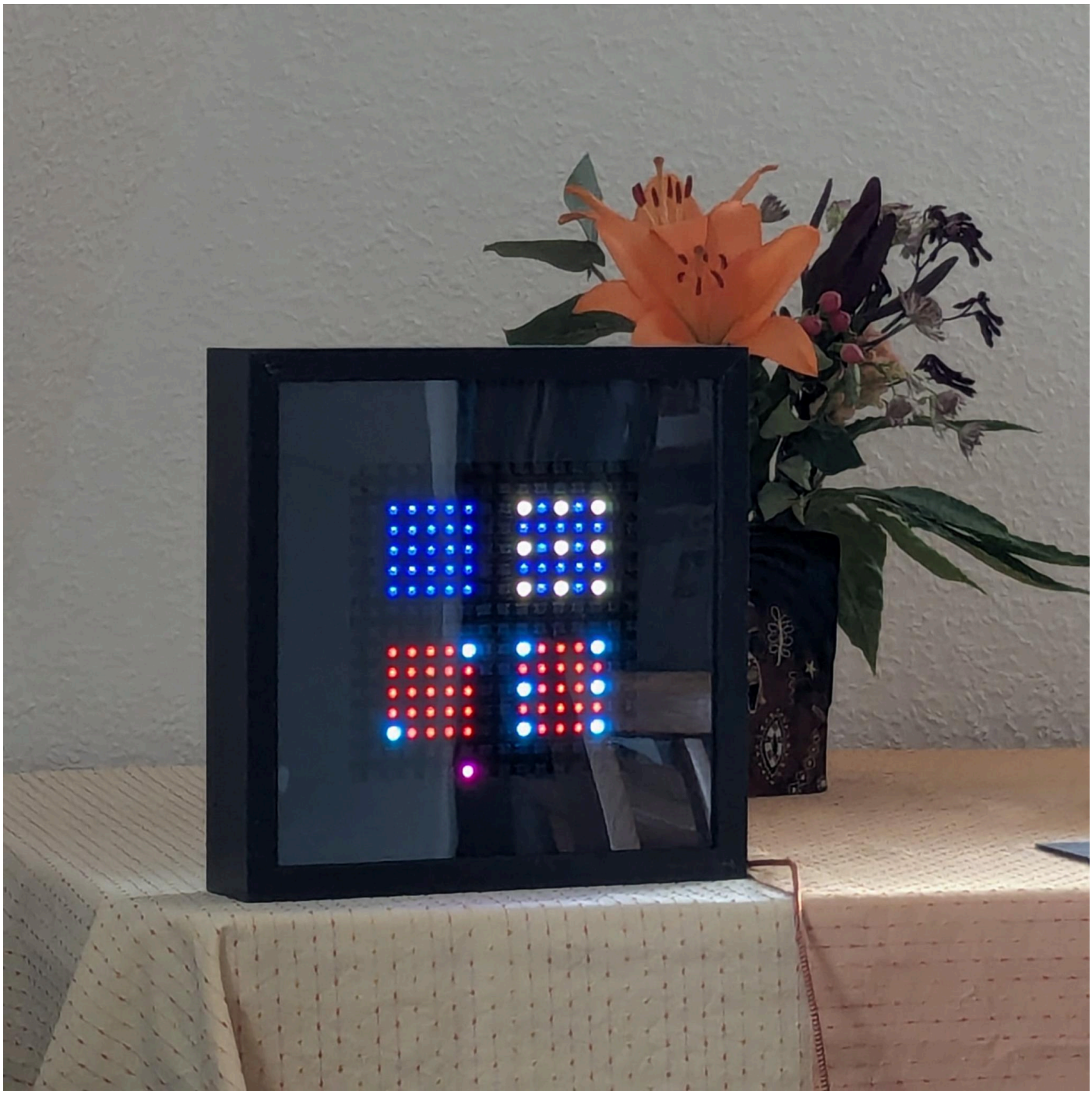


6.3. Further Examples

see my **Github** account for more: https://github.com/pagong

- NeoMatrix
- Demos
- Berlin-Uhr

6.3.1. Domino Clock



7. Backlinks

- index Friday 30 January 2026