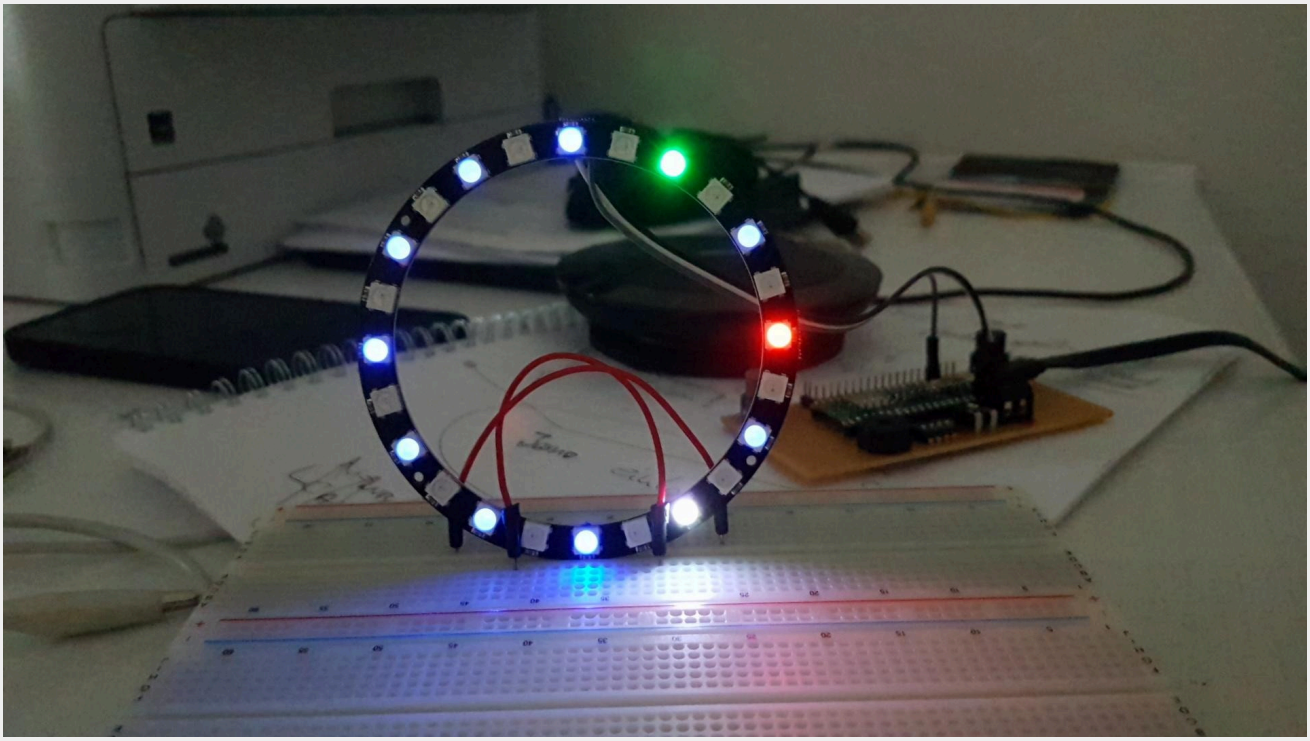
LED-Ring-Uhr

Die hier vorgestellte Uhr basiert auf einem Paspberry Pi Pico-Mikrocontroller und einem LED-Ring bestehend aus 24 Neopixel-LEDs, auf dem die Uhrzeit analog anzeigt wird. Das Programm ist vollständig in der Programmiersprache Python implementiert und baut auf Micropython auf.

Die Uhr ermittelt beim Starten die aktuelle Uhrzeit über eine API (application programming interface) eines Zeitservers (https://timeapi.io). Zusätzlich findet einmal täglich eine Synchronisierung statt.

Das Ziffernblatt zeigt Stundenmarkierungen in blau an, die Stunden 3, 6, 9 und 12 werden heller dargestellt. Ein roter Punkt zeigt die Stunde, ein grüner die Minute und ein weißer die Sekunde an. Der Sekundenzeiger ändert dabei im Sekundentakt die Helligkeit. Werden Stunde und Minute von dem gleichen Punkt dargestellt, so wechselt die Farbe der LED jede Sekunde zwischen rot und grün.

Prototyp

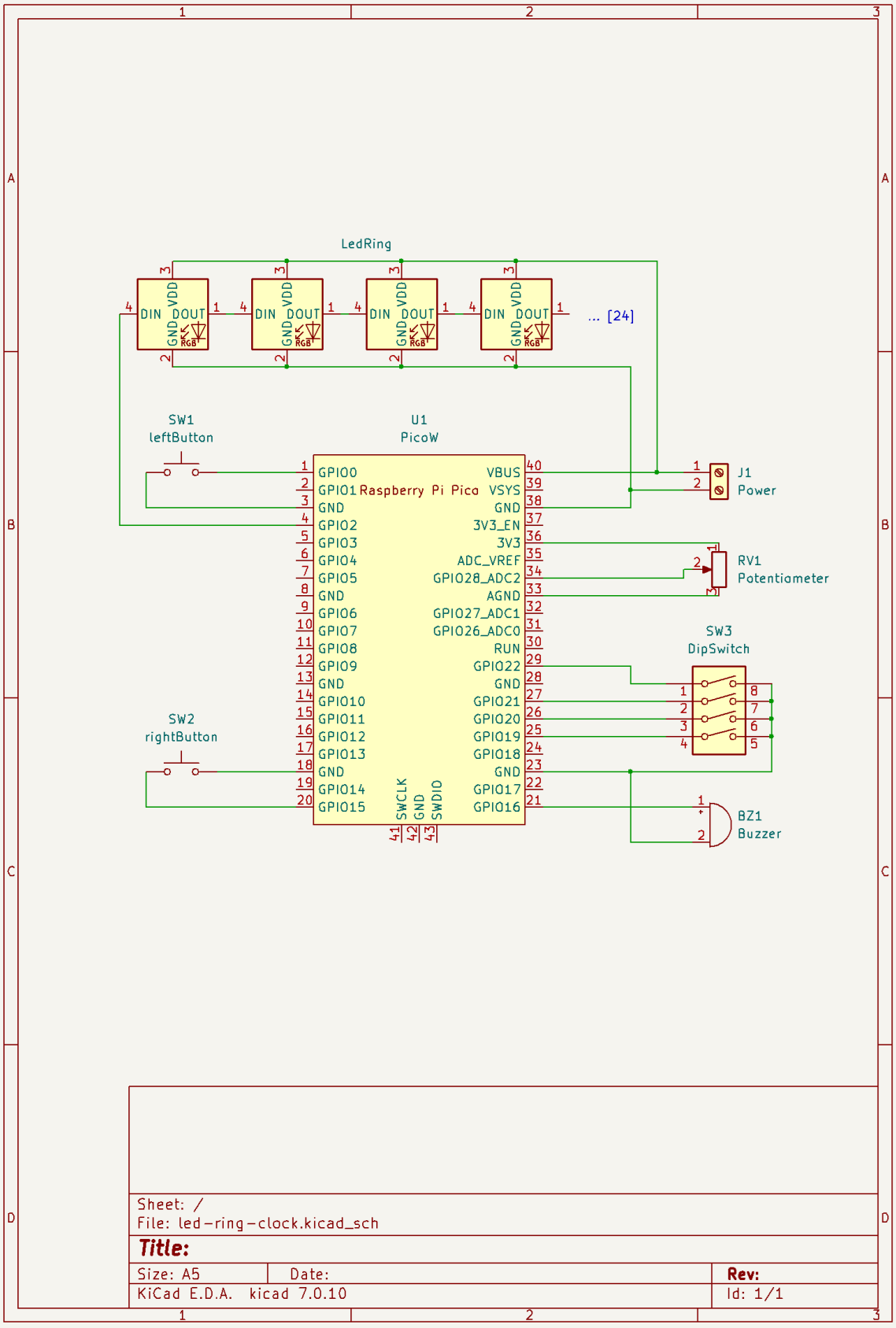


Hardware

Die Hardware ist eine selbstentwickelte Schaltung. Sie ermöglicht Eingaben über zwei Taster, einen vierpoligen DIP-Schalter und ein Trimm-Potentiometer. Ausgaben können optisch auf 24 ringförmig angeordneten LEDs angezeigt werden sowie akustisch über einen Piezo-Summer erfolgen. Die Steuerung erfolgt über einen Mikrocontroller vom Typ Raspberry Pi Pico-W, der ein WiFi-Modul besitzt und somit über WLAN kommmunizieren kann.

Für die LED-Ring-Uhr werden nicht alle Ein-/Ausgabekomponenten benötigt. Die Schaltung ist so ausgelegt, dass sie auch für andere Anwendungen genutzt werden kann (siehe Abschnitt "Weitere Projekt-Ideen").

Schaltplan


LedRing
DIN
DOUT
VDD
GND
... [24]
SW1
leftButton
U1
PicoW
Raspberry Pi Pico
GPIO0
GPIO1
GND
GPIO2
GPIO3
GPIO4
GPIO5
GPIO6
GPIO7
GPIO8
GPIO9
GPIO10
GPIO11
GPIO12
GPIO13
GPIO14
GPIO15
VBUS
VSYS
3V3_EN
3V3
ADC_VREF
GPIO28_ADC2
AGND
GPIO27_ADC1
GPIO26_ADC0
RUN
GPIO22
GPIO21
GPIO20
GPIO19
GPIO18
GPIO17
GPIO16
SWCLK
SWDIO
J1
Power
RV1
Potentiometer
SW3
DipSwitch
SW2
rightButton
BZ1
Buzzer
Sheet: /
File: led-ring-clock.kicad_sch
Title:
Size: A5
Date:
Rev:
KiCad E.D.A. kicad 7.0.10
Id: 1/1

Bauteilliste

Basisplatine:

- 1 Lochrasterplatine, Hartpapier, 50x100mm
- 1 Raspberry Pi Pico-W
- 1 10kOhm-Trimmpotentiometer
- 2 Eingabetaster
- 1 DIP-Schalter 4-polig
- 1 Piezo-Schallwandler (passiv)
- 1 Anschlussklemme 2-polig
- 2 Stiftleisten 20-polig
- 2 Stiftleisten 2-polig

Anzeige:

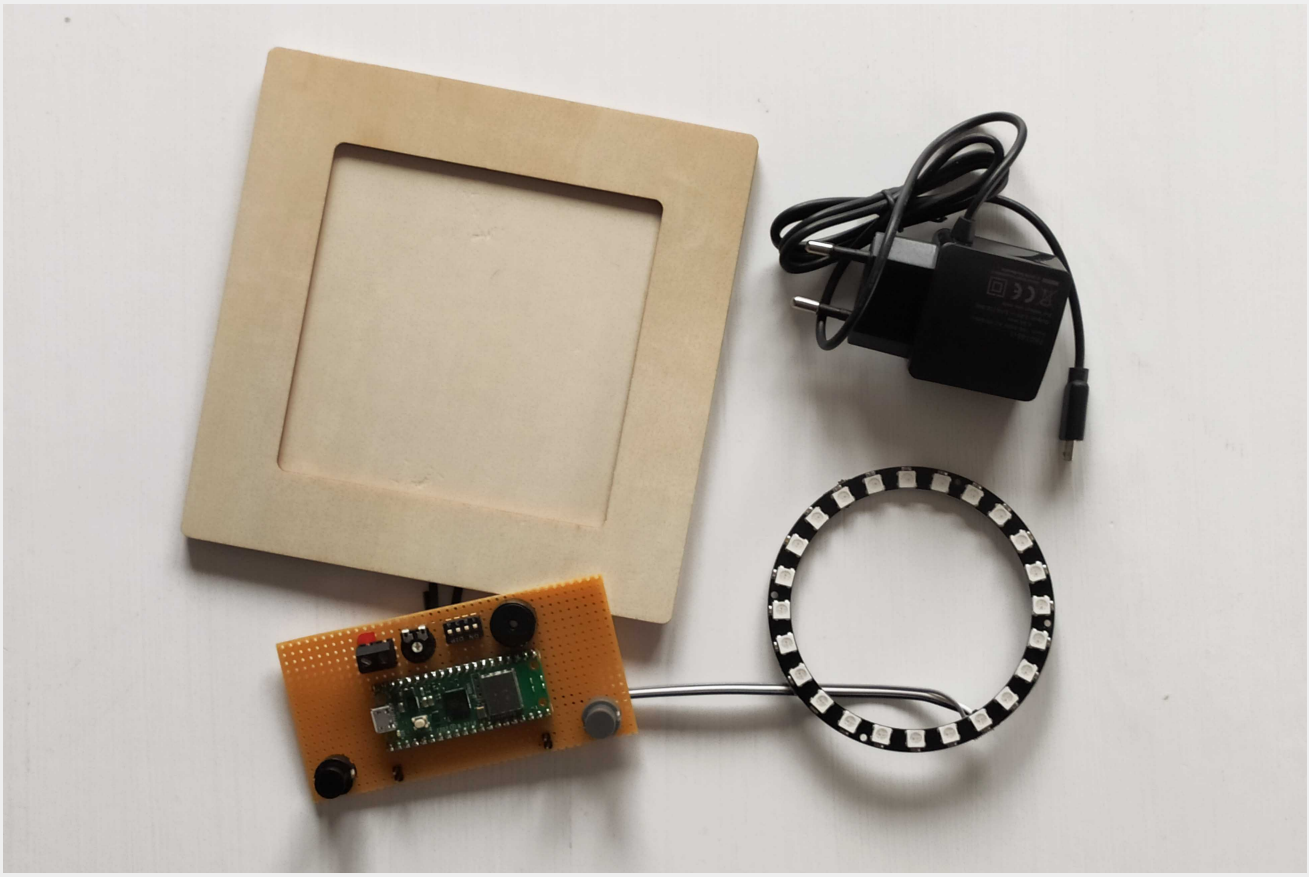
- 1 24-LED-Ring WS2812 5V

Stromversorgung:

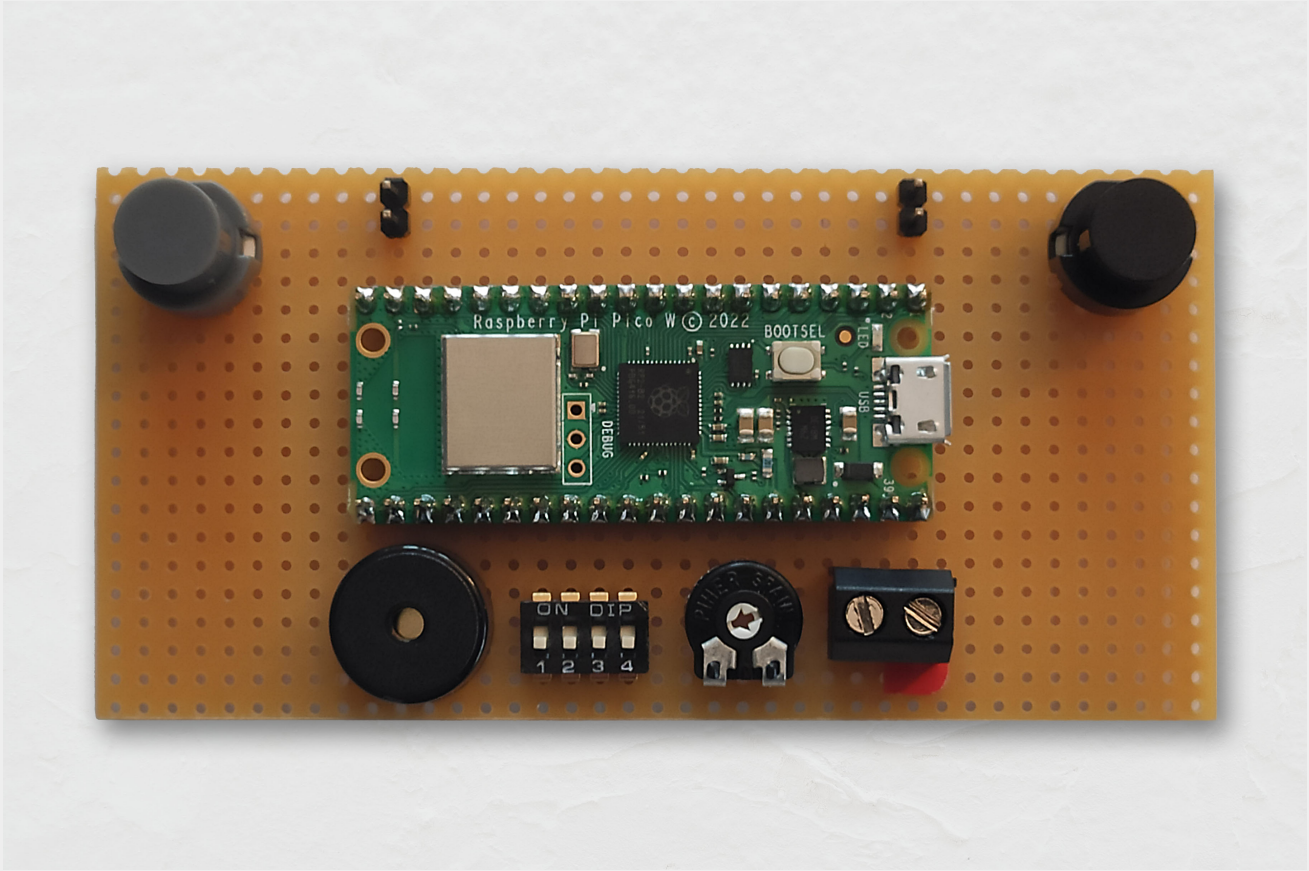
- 1 Netzteil mit Micro-USB-Anschluss 5V/2A

Gehäuse:

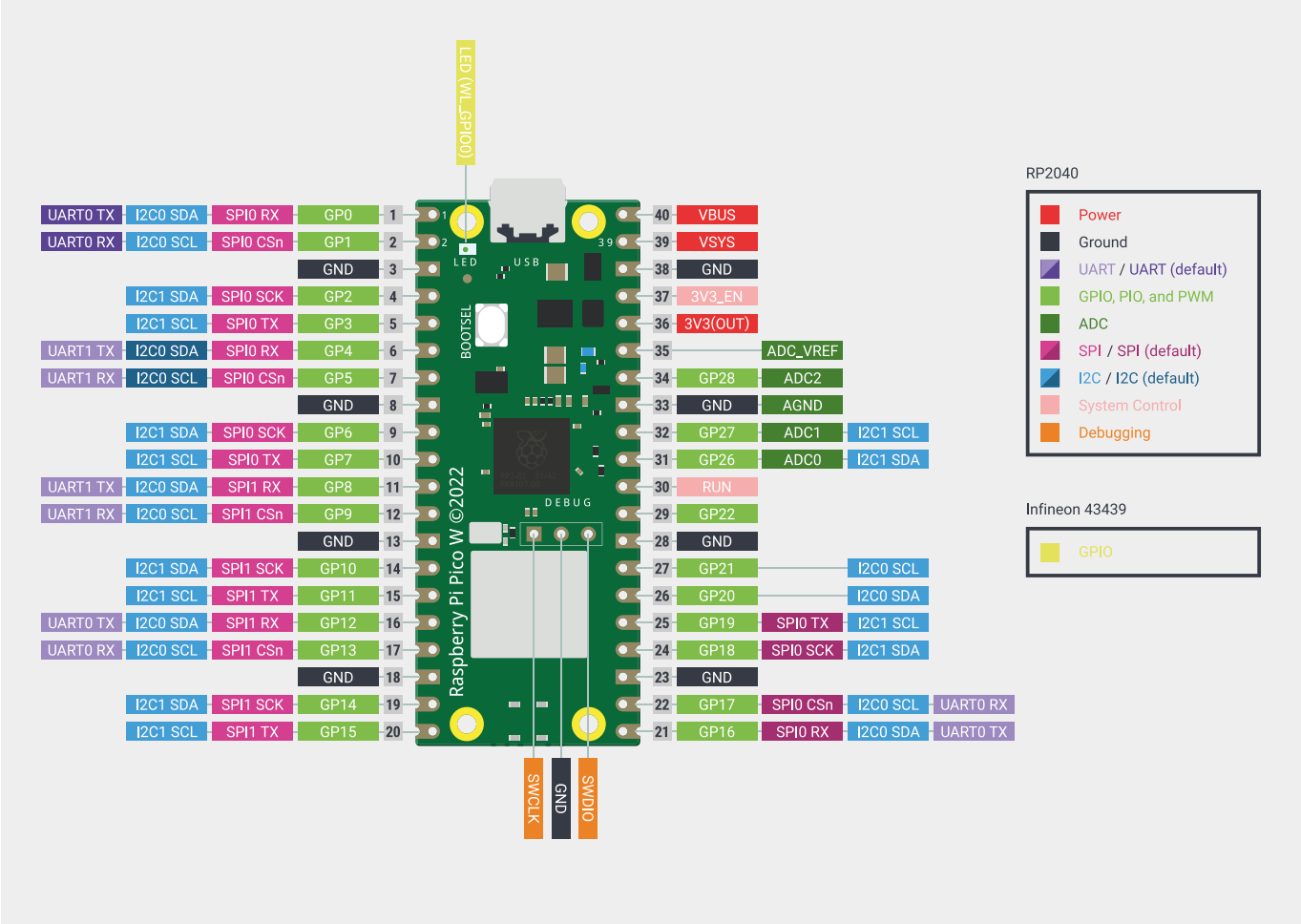
- 1 Bilderrahmen 15x15cm



Aufgebaute Basisplatine



Pinbelegung Pico-Modul



Bildquelle: https://www.raspberrypi.com/documentation/microcontrollers/images/picow-pinout.svg

Anschlussbelegung

Anschluss	Pin	Nutzung
Versorgungsspannung		
VBUS	40	Anschlussklemme VCC
GND	38	Anschlussklemme GND
Taster		
GPIO 0	1	Taster links
GND	3	Taster links GND
GPIO 15	20	Taster rechts
GND	18	Taster rechts GND
DIP-Schalter		
GPIO 19	25	DIP-Schalter 1
GPIO 20	26	DIP-Schalter 2
GPIO 21	27	DIP-Schalter 3
GPIO 22	29	DIP-Schalter 4
GND	23	DIP-Schalter GND
Potentiometer		
3,3V	36	Potentiometer Außenkontakt rechts
ADC 2 (GPIO 28)	34	Potentiometer Abgriff

Anschluss	Pin	Nutzung
AGND	33	Potentiometer Außenkontakt links
LED-Ring		
GPIO 2	4	LED-Ring IN
GND	38	LED-Ring GND
VBUS	40	LED-Ring VCC
Piezo-Summer		
GPIO 16	21	Piezo-Summer
GND	23	Piezo-Summer GND

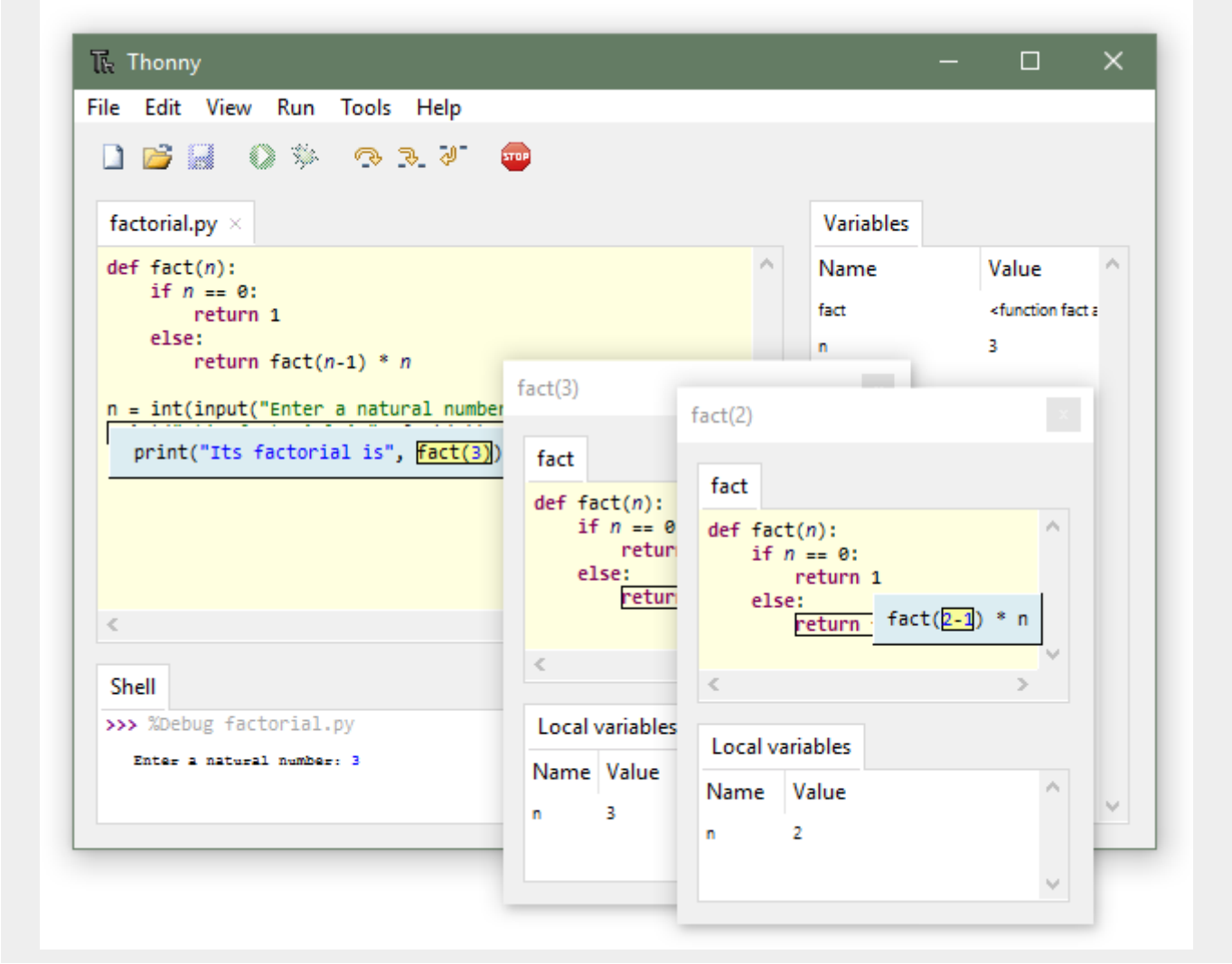
Weiterführende Links

- Raspberry Pi Pico – Raspberry Pi
- Raspberry Pi Pico W, RP2040 + WLAN Mikrocontroller-Board
- 24Bit RGB LED Ring WS2812 5V
- DIY Bilderrahmen für Tischdisplays und Bastelarbeiten

Entwicklungsumgebung

Thonny

Thonny ist ein kostenloses Entwicklungsprogramm für den PC, das von einem unabhängigen Entwickler namens Thonny erstellt wurde. Es handelt sich um eine integrierte Entwicklungsumgebung (IDE) mit Open-Source-Code, die verwendet werden kann, um verschiedene Anwendungen mit der Programmiersprache Python zu erstellen.



Bildquelle: https://thonny.org/img/screenshot.png

Download:

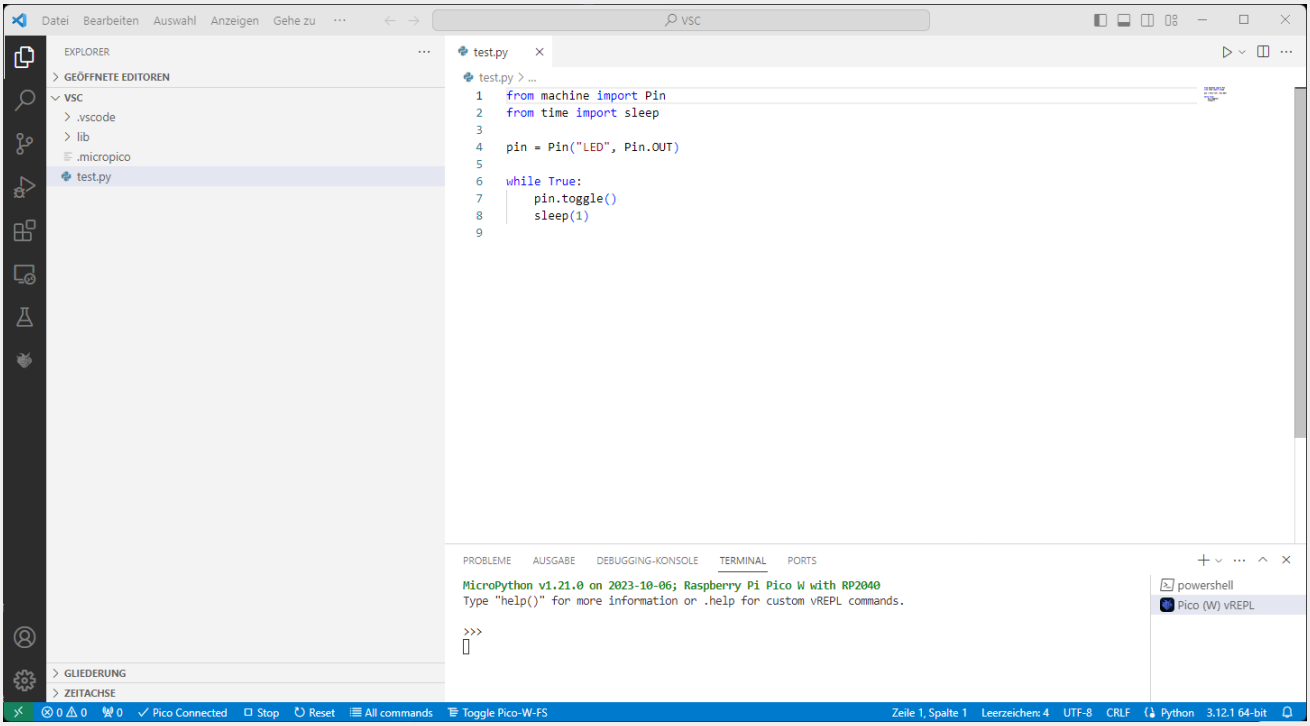
- Thonny, Phython IDE for beginners (Portable variant with 64-bit Python 3.10)

Installation:

1. Im Windows-Explorer ein Verzeichnis mit dem Namen `C:\PortableApps\Thonny` anlegen
2. Heruntergeladenen Zip-Datei öffnen und den gesamten Inhalt in das Vereichnis `C:\PortableApps\Thonny` kopieren
3. Thonny starten
4. Unter `Tools→Options…` im Reiter `General` bei `Language` aus der Liste `Deutsch` auswählen
5. Thonny schließen und wieder neu starten, damit die Änderung der Sprache greift
6. Unter `Extras→Optionen…` im Reiter `Interpreter` aus der Liste `MicroPython (Raspbeery Pi Pico)` auswählen
7. Unter `Extras→Optionen…` im Reiter `Themes & Schriftart` beide Schriftgrößen auf 8 setzen

Visual Studio Code (experimentell)

Die Nutzung von Microsoft Visual Studio Code (VSC) ist zwar sehr komfortabel, aber erfordert aber eine längere Einarbeitung. Außerdem ist die Anbindung an den Raspberry Pi Pico noch in einem experimentellen Stadium.



Download:

- Download Visual Studio Code - Mac, Linux, Windows (Windows - .zip - x86)

Installation:

1. Im Windows-Explorer ein Verzeichnis mit dem Namen `C:\PortableApps\VisualStudioCode` anlegen
2. Heruntergeladenen Zip-Datei öffnen und den gesamten Inhalt in das Vereichnis `C:\PortableApps\VisualStudioCode` kopieren
3. VSC starten
4. Unter „Erweiterungen (Extensions)“ in VSC die Erweiterung `German Language Pack for Visual Studio Code` suchen, diese installieren und VSC neu starten (https://marketplace.visualstudio.com/items?itemName=MS-CEINTL.vscode-language-pack-de)
5. Unter `Datei → Einstellungen → Design → Farbdesing` den Eintrag `Hell (Visual Studio)` wählen

Erweiterung für den Pico installieren:

1. Unter „Erweiterungen“ in VSC die Extension `MicroPico` suchen, diese installieren und den Anweisungen folgen (https://marketplace.visualstudio.com/items?itemName=paulober.pico-w-go)

Die Erweiterung `MicroPico` wurde umbenannt. Frühere Versionen hießen `Pico-W-Go`. Man findet daher in vielen Beschreibungen im Internet immer mal auch diese Bezeichnung.

Öffnen/Anlegen eines Projektordners:

1. Einen existierenden Ordner in VSC unter `Datei → Ordner öffnen` öffnen oder einen neuen Ordner mit `Datei → Ordner öffnen → Neuer Ordner` anlegen

Zum Pico verbinden (der Pico muss über das USB-Kabel angeschlossen sein):

1. Auf „MicroPico Device Controller“ klicken und dann auf „PACKAGES (PICO-W ONLY)“ klicken
2. In der unteren (blauen) Statuszeile werden jetzt Pico-spezifische Daten und Schaltflächen angezeigt:
    - `Pico Disconnected` / `Pico Connected`
    - Run / `Stop`
    - `Reset`
    - `All Commands`
    - `Toggle Pico-W-FS`

Anlegen eines VSC-Projekts (einmalig):

1. Über `Datei → Ordner öffnen …` den Projektordner öffnen
2. In der Statuszeile `All Commands` anklicken
3. Im Kommandofenster `MicroPico: Configure project` auswählen, um ein Projekt anzulegen. Hierdurch werden der Ordner `.vscode` und die Datei `.micropico` angelegt

Hochladen aller Dateien auf den Pico:

1. In der unteren (blauen) Statuszeile `All Commands` anklicken
2. Optional im oberen Kommandofenster `MicroPico: Delete all files from board` auswählen, um alle vorhandenen Dateien auf dem Pico zu laden
3. Im oberen Kommandofenster `MicroPico: Upload project to Pico` auswählen, um alle Dateien in dem Ordner auf den Pico zu laden
4. In der unteren Statuszeile `Reset` anklicken, um sicherzustellen, dass die neu geladenen Dateien genutzt werden

Hochladen einer einzelnen Dateien auf den Pico:

1. Im „Explorer“-Bereich auf die Datei mit der rechten Maustaste klicken und `Upload current file to Pico` auswählen
2. In der unteren Statuszeile `Reset` anklicken, um sicherzustellen, dass die neu geladene Datei genutzt werden

Programm auf dem Pico starten:

1. Im „Explorer“ die zu startende Programmdatei in den Editor laden
2. In der Statuszeile Run anklicken
3. Das Programm wird gestartet und die Ausgabe auf dem Terminalfenster ausgegeben

Programm anhalten:

1. In der Statuszeile `Stop` anklicken

How To's:

- https://www.laub-home.de/wiki/Raspberry_Pi_Pico_-_MicroPython_und_VSCode_als_IDE
- https://www.hackster.io/shilleh/how-to-use-vscode-with-raspberry-pi-pico-w-and-micropython-de88d6
- https://www.youtube.com/watch?v=O6lkYTfcMEg
- https://www.youtube.com/watch?v=lElb4lydGpU

Micropython

MicroPython ist eine schlanke und effiziente Implementierung der Programmiersprache Python 3, die eine Teilmenge der Python-Standardbibliothek enthält und für die Ausführung auf Mikrocontrollern und in eingeschränkten Umgebungen optimiert ist.

- MicroPython - Python for microcontrollers
- Raspberry Pi Documentation - MicroPython

Dokumentation zu MicroPython (allgemein und Pico-spezifisch)

- http://docs.micropython.org/en/latest/
- http://docs.micropython.org/en/latest/rp2/quickref.html
- Get Started with MicroPython on Raspberry Pi Pico

Download der Firmware

- MicroPython - Python for microcontrollers (Pico Firmware)
- MicroPython - Python for microcontrollers (Pico-W Firmware) (wird in diesem Projekt benötigt)

Aktuell genutze Version ist MicroPython Pico W v1.24.1(2024-11-29).uf2

Installationsanweisung (Firmware)

1. Um das Board in den Bootloader-Modus für das Firmware-Update zu bringen, die BOOTSEL-Taste gedrückt halten, während das Board an den USB-Anschluss angeschlossen wird.
2. Die uf2-Datei mit dem Firmware-Abbild wird dann auf den erscheinenden USB-Massenspeicher kopiert.
3. Sobald die Programmierung der neuen Firmware abgeschlossen ist, wird das Gerät automatisch zurückgesetzt und ist einsatzbereit.

Software

Das Software für die LED-Ring-Uhr ist in der Programmiersprache Python geschrieben. Python ist eine einfach zu lernende, aber mächtige Programmiersprache mit effizienten abstrakten Datenstrukturen und einem einfachen, aber effektiven Ansatz zur objektorientierten Programmierung. Durch die elegante Syntax und die dynamische Typisierung ist Python als interpretierte Sprache sowohl für Skripte als auch für schnelle Anwendungsentwicklung hervorragend geeignet.

- Welcome to Python.org
- https://de.wikipedia.org/wiki/Python_(Programmiersprache)

Download von Python:

- https://www.python.org/downloads/

Installation von Python:

1. Go to the Python Downloads page and download the installation file.
2. After a few seconds, you should have an executable file downloaded to your computer. Double-click the file to open it.
3. Make sure you have the option `Add Python to PATH` ticked to add Python to path.
4. Then, you can proceed with the installation.

Quellcode

Der Quellcode ist in zwei Bereiche aufgeteilt. Die Applikationsdateien bilden die notwendigen Funktionen und Klassen für die Steuerung der LED-Ring-Uhr ab. Die Bibliothesdateien stellen Hilfsfunktionen und -klassen zur Verfügung, die allgemeiner Natur sind und somit auch in anderen Projekten genutzt werden können (siehe Abschnitt "Weitere Projekt-Ideen").

Applikationsdateien

- `clock/mainClock.py`: Steuerung des Programmablaufs
- `clock/ClockTime.py`: Berechnung der Uhrzeit
- `clock/LedRingClockDisplay.py`: Anzeige der aktuellen Uhrzeit auf dem LED-Ring
- `clock/secrets.py`: Zugangsdaten für das Wireless-LAN

Bibliotheksdateien

Hardwarekomponenten

- `lib/Button.py`: Auslesen der Taster
- `lib/Buzzer.py`: Ansteuerung des Piezosummers
- `lib/DipSwitch.py`: Auslesen der DIP-Schaltereinstellung
- `lib/LedRing.py`: Ansteuerung des LED-Rings
- `lib/PicoWlan.py`: enthält die Zugangsdaten für das Wireless-LAN
- `lib/Potentiometer.py`: Auslesen der Potentiometerstellung

Instanziierung der Hardwarekomponenten

- `lib/mymachine.py`: Instanziierung der konkreten Hardwarekomponenten

Weitere Funktionen

- `lib/TimeApiIo.py`: Holen der aktuellen Uhrzeit von einem Zeitserver
- `lib/Timer.py`: Hilfsfunktionen für Steuerung des zeitlichen Ablaufs

Repository

Der vollständige Quellcode ist unter Git-Versionskontrolle und in folgendem GitLab-Repository abgelegt:

- fritzi4you / LED-Ring Clock · GitLab

Für der Zugriff auf das Repository ist der Besitz eines passenden Accesstokens erforderlich! Mithilfe des Programms `curl` kann dann die entsprechende Version heruntergeladen werden. `glpat-XXXXXXXXXXXXXXXXXXXX` muss hierfür durch de entsprechenden Token ersetzt werden, `sha=vX.X.X` wird auf die gewünschte Version gesetzt, z.B. `sha=v0.0.1`.

Der Befehl in der Kommandozeile lautet:

```
curl --header "Private-Token: glpat-XXXXXXXXXXXXXXXXXXXX"
https://gitlab.com/api/v4/projects/52515253/repository/archive.zip
?sha=vX.X.X  --output led-ring-uhr.zip
```

Die Quellcodes der Softwareversion werden dann in ein Archiv mit dem Namen `led-ring-uhr.zip` heruntergeladen.

Zur Zeit kann über folgenden Link eine erste Prototypversion des Quellcodes heruntergeladen werden: Quellcode Version 0.0.1

Dokumentation der Pico-Bibliothek

Die selbstentwickelte Softwarebibliothek abstrahiert die bereitgestellten Komponenten der Hardwareschaltung und ermöglicht so die einfache Nutzung dieser. Desweiteren werden mehrere Hilfsklassen und -funktionen zur Verfügung gestellt.

- Pico-Bibliothek

Weiterführende Links

- Getting started with Raspberry Pi Pico
- MicroPython - Python for microcontrollers
- NeoPixel WS2812B RGB LED with Raspberry Pi Pico (MicroPython)
- TimeAPI

Weitere Projekt-Ideen

- Ring-Pong
- Elektronischer Würfel
- Wetterstation
- Reaktionstest
- Pixel Chaser
- Simon Game Variante (Rechts/Links-Sequenz) https://dronebotworkshop.com/pi-10-pico-simon/
- Ampel
- Lichtorgel mit Mikrofon https://www.elektronik-kompendium.de/sites/raspberry-pi/2701251.htm
- Lampe mit zeitlichem Farbverlauf