

Selbstgebaute Wetterstation

Wie hoch sind Temperatur und Luftfeuchtigkeit?

- [Methodenkarte: Selbstgebaute Wetterstation](#)
- [Installation Thonny](#)
- [Das Raspberry Pi Pico mit Thonny programmieren](#)
- [Zusammenbau](#)
- [Der Code](#)
- [Online-Ressourcen](#)

Methodenkarte:

Selbstgebaute Wetterstation

Selbstgebaute Wetterstation

Zielgruppe	Dauer	Level	Gruppengröße
ab 10 Jahren	5 bis 10 Stunden	3	3 TN's

Kurzbeschreibung

In diesem Projekt wird ein Temperatursensor mit einem OLED-Display an einen Raspberry Pi Pico angeschlossen, um Temperatur- und Luftfeuchtwerte anzuzeigen. Ziel ist es, die Grundlagen der Sensorik und Programmierung zu vermitteln, die Teamarbeit zu fördern und ein Bewusstsein für die Verbindung von Natur und digitaler Technologie zu schaffen.

Ziele

- Teamarbeit fördern
- Technologien und Nachhaltigkeit kritisch reflektieren
- Grundlagen der Sensorik und von MicroPython vermitteln

Material	Werkzeug
<ul style="list-style-type: none">• Raspberry Pi Pico Set• Breadboard• Jumper, Dupont Kabel-Set• Temperatur-/Feuchtigkeitssensor DHT22• OLED-Display SSD1306	<ul style="list-style-type: none">• Lötset/Station• Zangenset• Sortier-/Projektbox

Ablauf

1. Vorbereitung:

Material und Werkzeuge werden bereitgestellt und vorab getestet, z.B. dass alle Bauteile richtig auf das Breadboard gesteckt sind (siehe QR-Code). Ein Demomodell dient als Anschauungsobjekt. Die Workshopleitung stellt sicher, dass die Programmierumgebung Thonny installiert ist und passende Zusatzmaterialien vorliegen. Die Teams überlegen, wie das Projekt an ihre Lebenswelt anknüpfen kann, und spielen mögliche Workshop-

Szenarien durch. Je nach Alter, Vorwissen und Beziehungsarbeit kann die Workshopzeit reduziert werden (z.B. bei der Phase Projektvorbereitung und Projektstart).

2. **Projektstart:**

Die Teilnehmenden starten mit einem Check-in, klären die Tagesziele ab und erarbeiten gemeinsame Regeln. Eine Ideensammlung zeigt mögliche Einsatzbereiche des Temperatursensors, unterstützt durch Videos und Recherchen. In Kleingruppen verteilen die Teilnehmenden Rollen wie Programmierer/in, Ingenieur/in oder Tester/in, um die Arbeit zu strukturieren.

3. **Praktische Arbeit, Aufbau und Test:**

Die Teams bauen das Projekt-Set mit den bereitgestellten Materialien zusammen. Nach dem Aufbau testen sie den Temperatursensor, indem sie Werte wie Temperatur und Luftfeuchtigkeit überprüfen. Experimente, wie etwa das Anhauchen des Sensors, verdeutlichen die Funktionsweise.

4. **Programmierung:**

Die Programmierumgebung Thonny wird genutzt, um den Code auf dem Raspberry Pi Pico anzupassen. Änderungen an der Reihenfolge oder Anzeige der Daten auf dem OLED-Display erlauben erste Erfahrungen mit Coding. Erweiterungen im Code bieten Raum für Kreativität.

5. **Reflexion:**

Die Teilnehmenden reflektieren in Gruppenarbeit, was gut funktioniert hat und was verbessert werden könnte. Sie teilen ihre Erfahrungen und erhalten Feedback von den Workshopleitung. Der Workshop endet mit einem Ausblick auf Anschlussprojekte.

Autor*in: Shelly Pröhl (*Büro Berlin des JFF*)

Installation Thonny

Wir programmieren das Raspberry Pi Pico mit der Skriptsprache MicroPython in der kostenlosen Entwicklungsumgebung (IDE) Thonny. Dafür verwenden wir einen Computer oder Laptop.

Thonny	https://thonny.org/
Micropython	https://micropython.org/

Was ist eine IDE?

Eine IDE (Integrated Development Environment) ist eine Software, die euch beim Schreiben, Testen und Ausführen von Code (Programmen) unterstützt. Sie vereint viele hilfreiche Werkzeuge an einem Ort, darunter:

- **Texteditor:** Zum Schreiben und Bearbeiten von Code.
- **Debugger:** Zum Finden und Beheben von Fehlern im Code.
- **Terminal:** Zum Ausführen des Codes und Anzeigen von Ergebnissen.

Was ist Thonny?

Thonny ist eine einfache und benutzerfreundliche IDE, die speziell für Python entwickelt wurde. Sie eignet sich besonders gut für Einsteiger*innen, die das Programmieren gerade erst lernen. Thonny bietet eine übersichtliche Benutzeroberfläche und viele hilfreiche Funktionen, die den Einstieg ins Programmieren erleichtern. Es ist ein großartiges Tool, um erste Schritte mit Python und MicroPython zu machen.

drawing drawing

Installation von Thonny auf verschiedenen Betriebssystemen

Installation auf Windows

- **Gehe zur offiziellen Thonny-Website.**
- Klicke auf den Download für Windows.
- Lade die Installationsdatei herunter und öffne sie, wenn der Download abgeschlossen ist.

- Folge den Anweisungen des Installationsassistenten, um Thonny zu installieren.
- Sobald die Installation abgeschlossen ist, kannst du Thonny über das Startmenü öffnen.

Installation auf Linux (Ubuntu)

- **Öffne das Terminal** auf deinem Ubuntu-System.
- Gib den folgenden Befehl ein, um das Paket zu aktualisieren:

```
sudo apt update
```

- Installiere Thonny, indem du den folgenden Befehl eingibst:

```
sudo apt install thonny
```

- Warte, bis die Installation abgeschlossen ist. Danach kannst du Thonny im Anwendungsmenü finden und starten.

Installation auf macOS

- **Gehe zur offiziellen Thonny-Website.**
- Klicke auf den Download für macOS.
- Lade die .dmg-Datei herunter und öffne sie, wenn der Download abgeschlossen ist.
- Ziehe das Thonny-Symbol in den Programme-Ordner, um die Installation abzuschließen.
- Öffne Thonny, indem du es im Programme-Ordner findest oder über Spotlight suchst.

„ Übung

Versuche nach der Installation, siehe unten, über die IDE Thonny eine Bibliothek zu installieren, zum Beispiel die Bibliothek 'NeoPixel'.

- Thonny öffnen ->
- Menü -> Tools -> Manage Plug-ins
- Suche nach 'neopixel' -> installieren

Das Raspberry Pi Pico mit Thonny programmieren

Um ein neues Raspberry Pi Pico zu programmieren, müssen wir es zunächst vorbereiten. Es mag anfangs nach vielen Schritten klingen, aber sobald ihr es einmal gemacht habt, geht der Rest richtig schnell! (🍷 ☺ 🍷) Wir teilen den Prozess in drei Schritte auf:

1. **Installation von MicroPython auf dem Raspberry Pi Pico**
2. **Raspberry Pi Pico mit Thonny öffnen**
3. **Ein Programm auf dem Raspberry Pi Pico speichern/laden**

Installation von Micropython auf dem Raspberry Pi Pico

- **Ladet euch Micropython herunter:**

Besucht die [MicroPython-Website](#) und ladet die passende UF2-Datei für das Raspberry Pi Pico herunter.

(Zum Beispiel [v1.24.0 \(2024-10-25\) .uf2](#))

- **Schließt das Raspberry Pi Pico an den Computer an:**

Verwendet ein Micro-USB-zu-USB-A-Kabel. Haltet dabei den BOOTSEL-Knopf beim Raspberry Pi Pico gedrückt, während ihr das USB-Kabel anschließt.

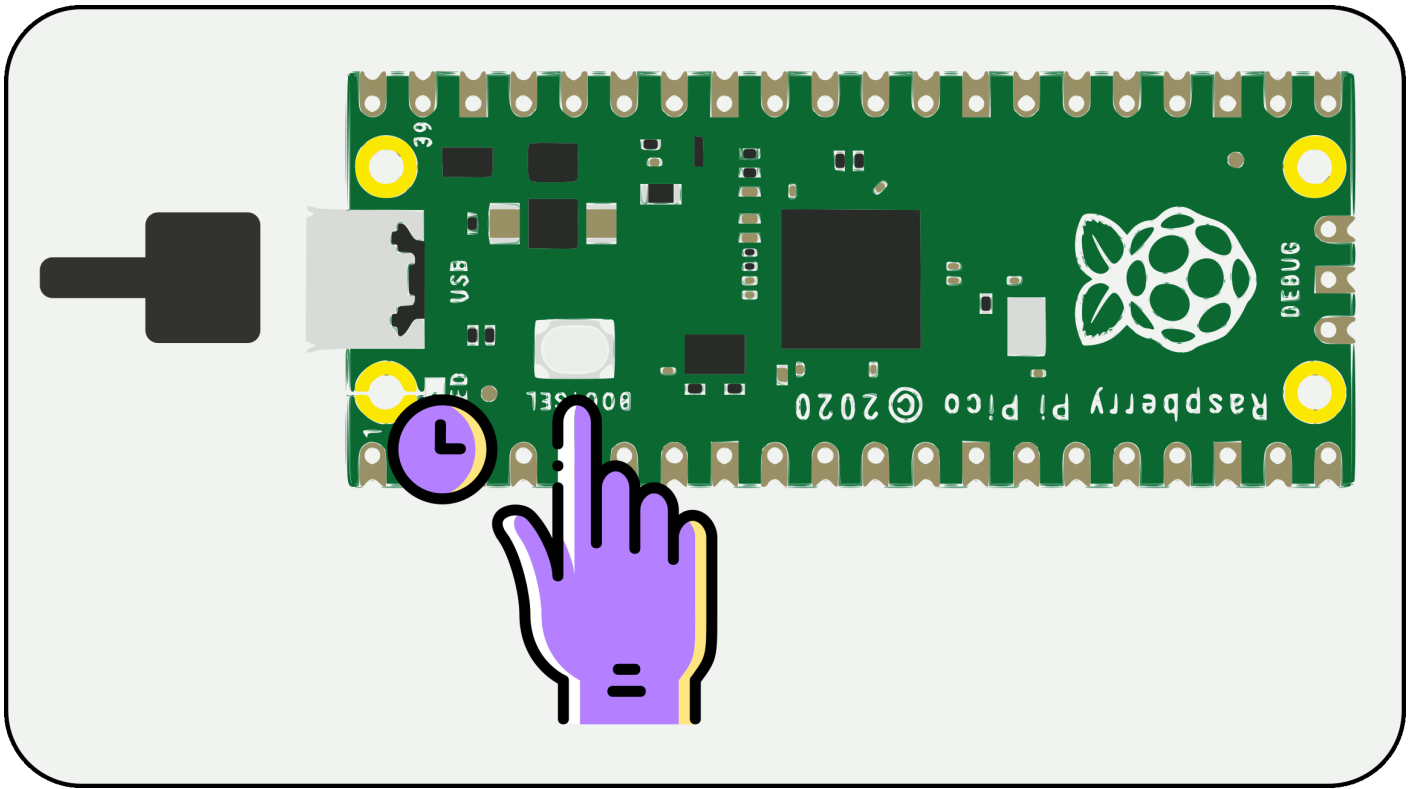
(siehe Animation)

- **Kopiert die UF2-Datei auf das neue Laufwerk:**

Sobald das Pico als neues Laufwerk auf eurem Computer erscheint, zieht die heruntergeladene UF2-Datei per Drag-and-Drop in dieses Laufwerk. Dadurch wird MicroPython auf dem Raspberry Pi Pico installiert.

Während der Installation oder dem Laden von MicroPython auf das Raspberry Pi Pico trennt sich das Laufwerk automatisch vom Computer. Dies zeigt an, dass die Installation abgeschlossen ist. Zieht auf keinen Fall das Kabel während dieses Prozesses vom Pico ab, da dies die Installation unterbrechen und zu Fehlern führen könnte.

Sicherheit geht vor – lasst das Pico in Ruhe arbeiten! (🍷 ☺ 🍷)



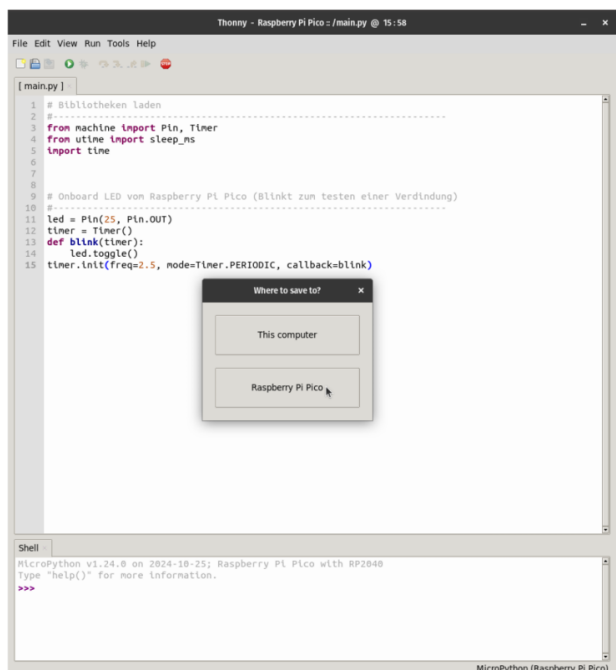
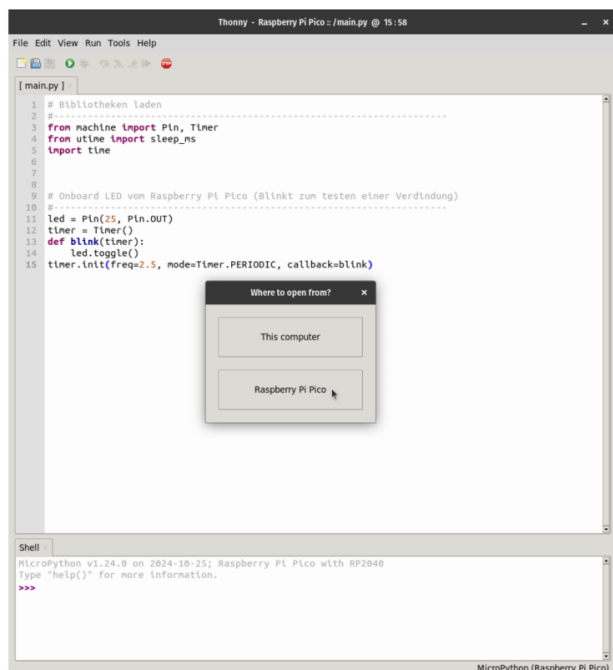
Mit Thonny ein Programm auf dem Raspberry Pi Pico speichern/laden

Nachdem ihr Thonny geöffnet habt, könnt ihr über das Menü die Option **Dateien/Files** auswählen, um Dateien zu verwalten.

- Mit der Option **Öffnen/Open** öffnet sich ein Dialog, in dem ihr auswählen könnt, ob ihr eine Datei von eurem Computer oder direkt vom Raspberry Pi Pico laden möchtet.
- Mit der Option **Speichern/Save** öffnet sich ein Dialog, in dem ihr auswählen könnt, wo ihr euer Programm speichern möchtet – entweder auf dem Computer oder auf dem Raspberry Pi Pico.

So könnt ihr eure Programme einfach verwalten und sicherstellen, dass sie immer an der richtigen Stelle gespeichert sind!

Achtet beim Speichern auf dem Raspberry Pi Pico darauf, dass die Datei den Namen **main.py** hat. Nur mit diesem Dateinamen erkennt das Pico euer Programm automatisch und führt es nach dem Starten aus. ⌘ (^) . (^)



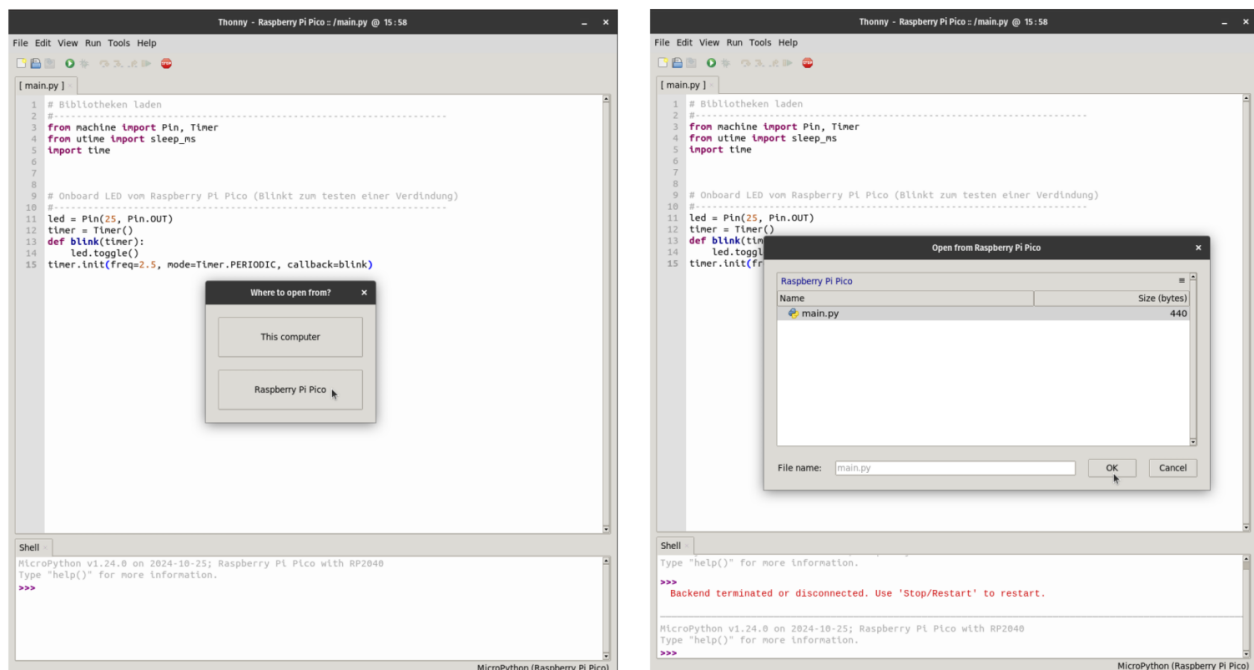
Raspberry Pi Pico mit Thonny öffnen

Normalerweise kommuniziert Thonny automatisch mit dem Raspberry Pi Pico, sobald es angeschlossen ist. Wenn ihr Thonny geöffnet habt, geht wie folgt vor:

1. Wählt im Menü die Option **Dateien/Files** aus.
2. Im neuen Dialog könnt ihr das **Raspberry Pi Pico** als Speicherort auswählen.
3. Anschließend könnt ihr eure **main.py**-Datei auf dem Pico finden, auswählen und öffnen.

So könnt ihr sicherstellen, dass euer Programm korrekt geladen und ausgeführt wird!

Nachdem ihr eure **main.py** zum Öffnen ausgewählt habt, könnt ihr das Programm in Thonny bearbeiten. Wenn ihr Änderungen vornehmt und auf **Speichern** klickt, wird das Programm automatisch auf dem Raspberry Pi Pico aktualisiert und gespeichert. So bleiben eure Änderungen direkt auf dem Pico erhalten! `~_(_)/`



Programm starten/stoppen

Nachdem ihr ein Programm erfolgreich auf das Raspberry Pi Pico gespeichert oder übertragen habt, könnt ihr es ausführen und bei Bedarf stoppen:

- **Starten:** Klickt in der Symbolleiste auf den runden **grünen Button**, um das Programm auf dem Raspberry Pi Pico zu starten. Alternativ könnt ihr die **Taste F5** drücken.
- **Stoppen:** Klickt in der Symbolleiste auf den **roten Stop-Button**, um das Programm zu stoppen. Alternativ könnt ihr die Tastenkombination **Strg + F2** verwenden.

Buttons in Thonny zur Steuerung des Programmes auf dem Raspberry Pi Pico



**Programm
starten**



**Programm
stoppen**

Zusammenbau

drawing

Beispiel-Verkabelung (siehe Abbildung oben):

In der Abbildung werden **rote**, **schwarze** und **grüne** Kabel (Jumper) verwendet, die jeweils eine spezifische Funktion haben:

- **Rot (Power/PWR):** Liefert Energie an das Bauteil, damit es funktioniert.
- **Schwarz (Ground/GRD):** Schließt den Stromkreis und leitet überschüssige Energie ab.
- **Grün (Data):** Überträgt die Daten zwischen dem Raspberry Pi Pico und dem Bauteil.

Vorbereitung:

Stellen Sie sicher, dass Sie alle benötigten Komponenten haben: Raspberry Pi Pico, OLED-Display (SSD1306), Temperatur- und Luftfeuchtigkeitssensor (DHT22), Jumper-Kabel, einen Widerstand (4,7 kΩ) und ein Breadboard.

a) Stromversorgung einrichten:

- Verbinde den **5V-Pin** des Raspberry Pi Pico mit der **positiven Leiste** (rote Linie) des Breadboards.
- Verbinde einen **GND-Pin** des Raspberry Pi Pico mit der **negativen Leiste** (blaue Linie) des Breadboards.

Jetzt können alle Bauteile auf dem Breadboard mit Strom versorgt werden.

b) Temperatur-/Luftfeuchtigkeitssensor DHT22 anschließen:

- Stecke ein Jumper-Kabel vom **VCC-Pin** des Relaismoduls in die **positive Leiste** des Breadboards.
- Verbinde den **GND-Pin** des Relaismoduls mit der **negativen Leiste** des Breadboards.
- **Data-Pin** des Sensors an den **GPIO-Pin GP2** des Pico.

Hinweis:

Ein **Pull-up-Widerstand** (meist 4,7 kΩ) ist erforderlich, um die Datenleitung stabil auf "High" zu halten und Kommunikationsprobleme mit dem DHT22-Sensor zu vermeiden. Er stellt sicher, dass der Sensor seine Daten zuverlässig an den Mikrocontroller übertragen kann.

c) OLED Display SSD1306 anschließen:

- Stecke ein Jumper-Kabel vom **VCC-Pin** des Relaismoduls in die **positive Leiste** des Breadboards.
- Verbinde den **GND-Pin** des Relaismoduls mit der **negativen Leiste** des Breadboards.
- **SCL-Pin** des Displays an den **GPIO-Pin GP5** des Pico.
- **SDA-Pin** des Displays an den **GPIO-Pin GP4** des Pico.

Der Code

Hier ist ein Beispielcode für eure Programmierung in Micropython in der IDE Thonny auf dem Raspberry Pi Pico.

Ihr könnt diese Code Kopieren und in eine Datei namens 'main.py' speichern. Die Bezeichnung ist entscheidend, damit das Raspberry Pi Pico euren Code versteht!

Ganz besonders ist, dass ihr aber nicht nur diese Datei benötigt, sondern auch eine Datei, wo der Code gespeichert ist, welcher das Display steuert. Diesen könnt ihr hier Herunterladen ->

[ssd1306.py](#). Beide Dateien speichert ihr dann über die IDE Thonny auf euer Raspberry Pi Pico. :)

```
from machine import Pin, I2C
#
import time
import utime
import framebuf
#
import dht
from ssd1306 import SSD1306_I2C
#
# OLED pixel definition (WxH)
WIDTH = 128
HEIGHT = 32
# I2C0 pin assignments
SCL = 5
SDA = 4
# DHT22 sensor
sensor = dht.DHT22(Pin(2))
# Initialize I2C0, Scan and Debug print of SSD1306 I2C device address
i2c = I2C(0, scl=Pin(SCL), sda=Pin(SDA), freq=200000)
# Initialize OLED
oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)

# Initialize LED (Pin 25)
led = machine.Pin(25, machine.Pin.OUT)
# Toggle LED functionality
def BlinkLED(timer_one):
```

```
led.toggle()

# Initialize timer_one. Used for toggling the on board LED
timer_one = machine.Timer()

# timer_one initialization for on board blinking LED at 200ms interval
timer_one.init(freq=5,mode=machine.Timer.PERIODIC,callback=BlinkLED)

while True:
    time.sleep_ms(250)
    sensor.measure()
    oled.fill(0)
    temp = sensor.temperature()
    hum = sensor.humidity()
    oled.text("Temp. {} C".format(temp),5,5)
    oled.text("Feucht. {:.0f} % ".format(hum),5,15)
    #Show display
    oled.show()
    # Wait for Five seconds. Then proceed to collect next sensor reading.
    time.sleep_ms(5000)
```

Online-Ressourcen

/