Automatische Wasserpumpe

Bau dein eigenes Wasserkraftwerk!

- Methodenkarte: Automatische Wasserpumpe
- Installation Thonny
- Das Raspberry Pi Pico mit Thonny programmieren
- Zusammenbau
- Der Code
- Online-Ressourcen

Methodenkarte: Automatische Wasserpumpe

Bau dein eigenes Wasserkraftwerk!

Zielgruppe	Dauer	Level	Gruppengröße
ab 8 Jahre	5 bis 10 Stunden	3	3 TN's

Kurzbeschreibung

Das Projekt umfasst den Bau eines automatisierten Bewässerungssystems. Mithilfe eines Raspberry Pi Pico, eines Relaismoduls und einer Wasserpumpe entwickeln die Teilnehmenden eine Steuerung, die die Pumpe ein- und ausschaltet. Weitere Bauteile wie Breadboard und Jumper-Kabel ermöglichen die lötfreie Verbindung.

> **Ziele**
> *- Verständnis für Stromkreisläufe fördern*
> *- Grundlagen von MicroPython vermitteln*
> *- technische Zusammenhänge reflektieren*
> *- Umwelt-Technologie-Wechselwirkungen vermitteln*

Material	Werkzeug
• Raspberry Pi Pico Set • Breadboard • Jumper, Dupont Kabel Set • Relais 5V KY-019 • 3V-5V DC Wasserpumpe • Schlauch	• Lötset/Station • Schraubenzieherset • Sortier-/Projektbox

Ablauf

1. **Vorbereitung:**
   In der Vorbereitungsphase wird das Material bereitgestellt und getestet, ob alle Bauteile, wie der Raspberry Pi Pico, Relais und die Wasserpumpe, funktionieren (siehe QR-Code). Bauteile wie Pins müssen vorbereitet und korrekt verkabelt werden. Außerdem sollte ein Demomodell gebaut und die Programmierumgebung Thonny installiert werden. Je nach Alter, Vorwissen und Beziehungsarbeit kann sich die Workshopzeit verkürzen (z.B. in den Phasen Projektvorbereitung und Projektstart).
2. **Projektstart:**
   Der Workshop beginnt mit einer Check-in-Runde, bei der die Teilnehmenden ihre Wünsche und Erwartungen sammeln und Regeln für die gemeinsame Zusammenarbeit festlegen. Die gemeinsame Einstimmung auf das Thema erfolgt, indem die Teilnehmenden z.B. Ideen zu Einsatzmöglichkeiten von Wasserpumpen und technischen Bauteilen zusammentragen. Dabei hilft ihnen auch eine Recherche im Internet.
3. **Praktische Arbeit, Aufbau und Test:**
   Nach der Einführung zu den einzelnen Arbeitsschritten arbeiten die Teilnehmenden selbstständig in Dreiergruppen. Sie bauen die Bauteile zusammen und testen die Funktionalität der Wasserpumpe. Experimente, etwa das Messen des Wasserflusses, fördern das technische Verständnis. Bei Bedarf unterstützt die Workshopleitung individuell.
4. **Programmierung:**
   In dieser Phase programmieren die Teilnehmenden den Raspberry Pi Pico mit der Software Thonny. Sie verändern Parameter im Code wie etwa die Laufzeit der Pumpe und experimentieren mit Anpassungen im Programm. Zusätzliche Funktionen, z.B. die Anpassung der Bewässerung an Pflanzenbedürfnisse, können integriert werden.
5. **Reflexion:**
   Zum Abschluss reflektieren die Gruppen ihre Erfahrungen. Dabei heben sie die Stärken der Zusammenarbeit hervor und sammeln Verbesserungsvorschläge für zukünftige Projekte. Ein Feedback der Workshopleitung und ein Ausblick auf mögliche Erweiterungen schließen den Workshop ab.

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**

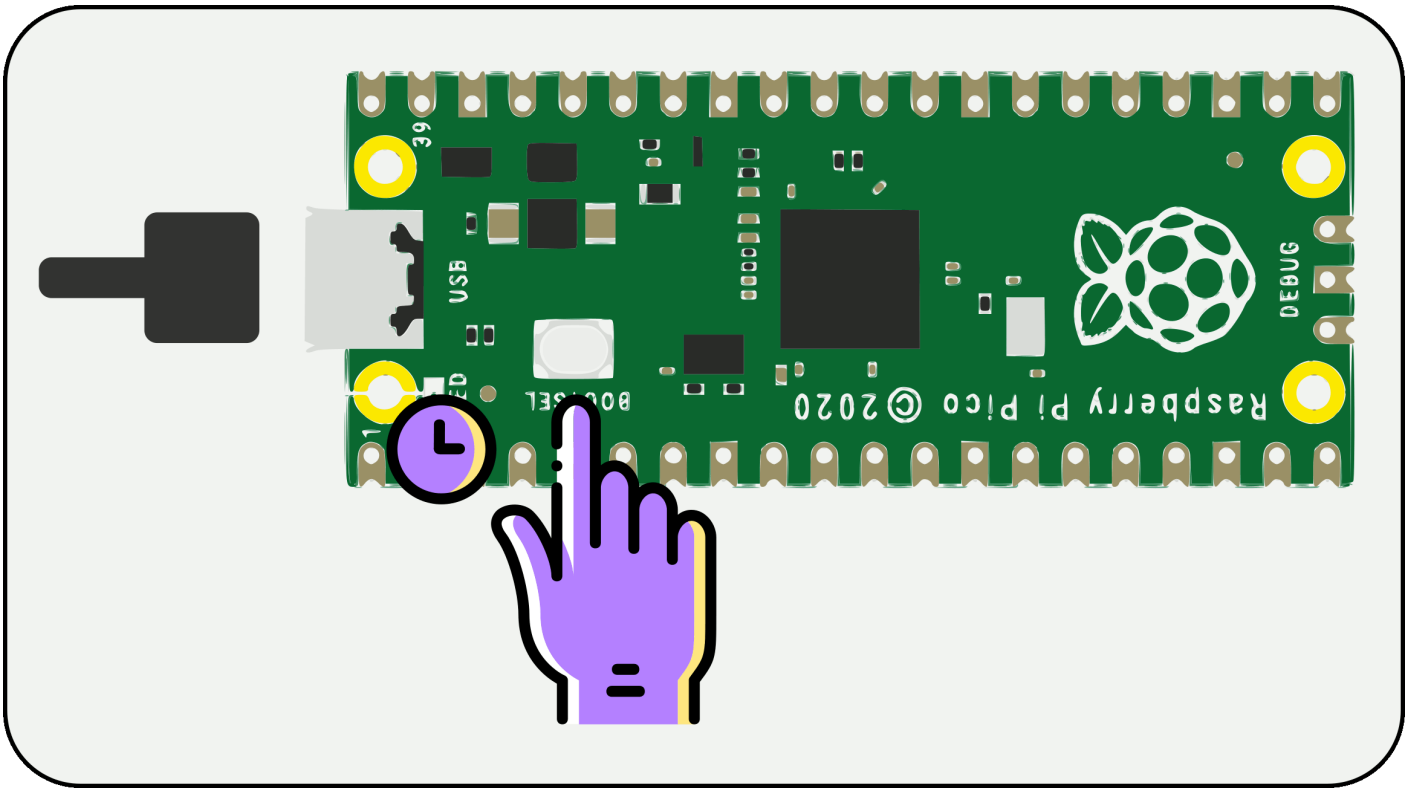
Installation von Micropython auf dem Raspberry Pi Pico

- **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! (□ ^ ᵕ ^□)



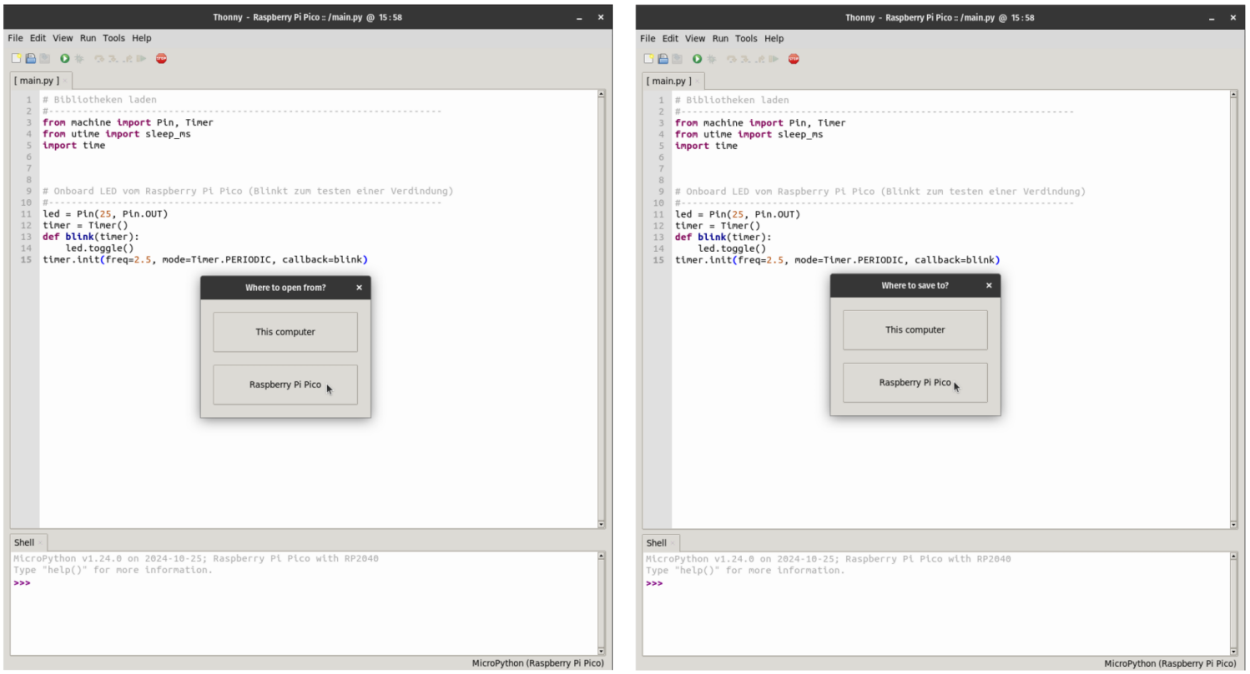
Mit Thonny ein Programm auf dem Raspberry Pi Pico speichern/laden

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. ᓚ(^. .^)



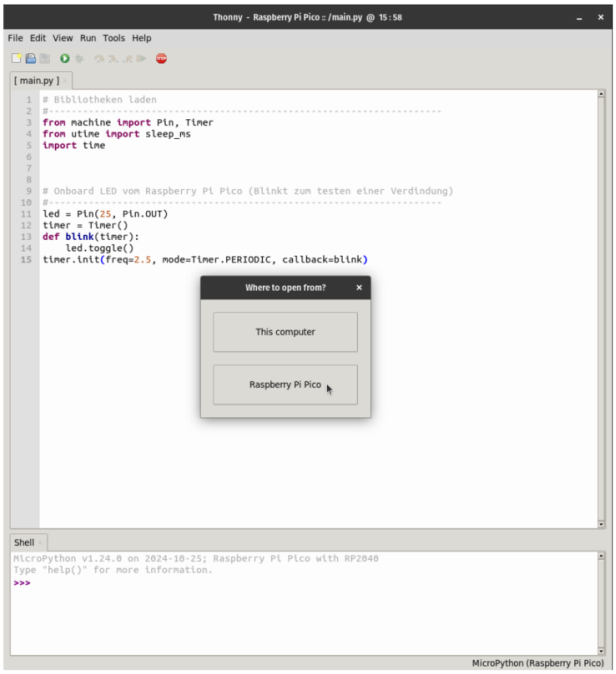
Raspberry Pi Pico mit Thonny öffnen

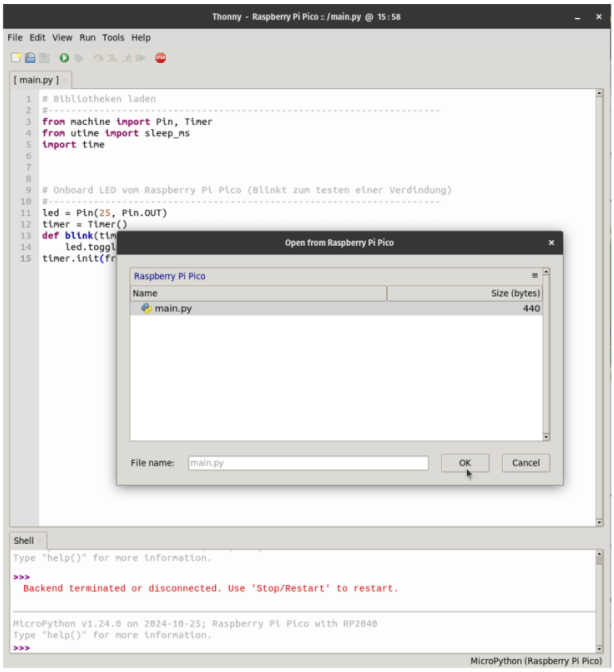
Normalerweise kommuniziert Thonny automatisch mit dem Raspberry Pi Pico, sobald es angeschlossen ist. Wenn ihr Thonny geöffnet habt, geht wie folgt vor:

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

auto_bewaesserungssystem_bb.png

> **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.

1) 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.

2) Relais 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.
- Schließe den **IN-Pin** des Relaismoduls mit einem Jumper-Kabel an den **GPIO-Pin GP15** des Raspberry Pi Pico an.

3) Wasserpumpe anschließen:

- Stecke das **rote Kabel** der Pumpe in den **NO-Anschluss (Normally Open)** des Relaismoduls.
- Verbinde das **schwarze Kabel** der Pumpe mit der **negativen Leiste** des Breadboards.
- Stecke ein Jumper-Kabel vom **COM-Anschluss (Common)** des Relaismoduls in die **positive Leiste** des Breadboards.

> **Optional:**
>
> Ihr könnt Jumper-Kabel an die Kabel der Pumpe anlöten, um die Kabellänge zu verlängern. Dadurch könnt ihr die Pumpe flexibler positionieren. Denkt daran, die Lötstellen mit Isolierklebeband oder Schrumpfschlauch abzudichten, damit sie besser geschützt und

stabiler sind. Das erhöht die Haltbarkeit und Sicherheit eurer Verbindung.

Viel Spaß beim Ausprobieren! Es ist spannend zu sehen, wie alles zum Leben erwacht, wenn es richtig verkabelt ist! (□ ˆ□ ˆ□)

Der Code

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

```
from machine import Pin
import time

# Initialisiere den GPIO-Pin für das Relais (hier GPIO 15)
relay = Pin(15, Pin.OUT)

# Funktion zum Einschalten der Pumpe
def turn_on_pump():
    relay.value(1)
    print("Pumpe eingeschaltet")

# Funktion zum Ausschalten der Pumpe
def turn_off_pump():
    relay.value(0)
    print("Pumpe ausgeschaltet")

# Hauptprogramm
try:
    while True:
        # Schalte die Pumpe für 5 Sekunden ein
        turn_on_pump()
        time.sleep(5)

        # Schalte die Pumpe für 10 Sekunden aus
        turn_off_pump()
        time.sleep(10)
        # Warte 7 Tage (7 * 24 * 60 * 60 Sekunden)
        #time.sleep(7 * 24 * 60 * 60)

except KeyboardInterrupt:
    # Schalte die Pumpe aus, wenn das Programm beendet wird
    turn_off_pump()
```

```
    print("Programm beendet")
```

Online-Ressourcen

/