Adventskranz mit LED-Steuerung über Raspberry Pi

(Autor: Michael Abendroth)

Projektbeschreibung

Dieses Projekt steuert eine Reihe von LEDs auf einem Raspberry Pi, um die Adventssonntage im Dezember eines Jahres anzuzeigen. Es wird jede Woche eine neue LED aufleuchten, die den jeweiligen Adventssonntag darstellt. Das Projekt berechnet automatisch die Adventssonntage basierend auf dem Datum und steuert die LEDs entsprechend an.

Hardwareanforderungen

- **Raspberry Pi** (Jedes Modell mit GPIO-Unterstützung, z.B. Raspberry Pi 3 oder 4)
- **4 LEDs** (verschiedene Farben können verwendet werden)
- **4 Widerstände** (jeweils 100 Ω)
- **Jumper-Kabel** (zur Verbindung der LEDs mit den GPIO-Pins)
- **Breadboard** (optional, für einfache Verdrahtung)

Softwareanforderungen

- Python 3
- RPi.GPIO Bibliothek
- Zeitmodul (`time`)

Funktionsweise des Projekts

- **LED-Steuerung**: Über GPIO-Pins werden 4 LEDs angesteuert. Jede LED steht für einen Adventssonntag, beginnend mit dem ersten Advent.
- **Berechnung der Adventssonntage**: Das Skript berechnet die Adventssonntage für das gegebene Jahr und leuchtet entsprechend eine neue LED auf. An jedem Adventssonntag wird die entsprechende LED zusammen mit den LEDs der vorherigen Sonntage aktiviert.
- **Testmodus**: Ein Testdatum kann vorab gesetzt werden, um die Funktionalität ohne das aktuelle Datum zu testen.
- **Endlosschleife**: Das Skript läuft in einer Endlosschleife und überprüft jede Minute, ob sich der Adventssonntag geändert hat, und passt die LEDs an. Die Schleife kann mit `Ctrl+C` unterbrochen werden.

---

1. Berechnung der Adventssonntage

Die Adventssonntage sind die vier Sonntage, die vor Weihnachten liegen. Weihnachten fällt immer auf den 25. Dezember, und der vierte Advent ist immer der letzte Sonntag vor Weihnachten.

Wie wird der vierte Advent berechnet?

- Der vierte Advent ist der Sonntag, der dem 25. Dezember am nächsten liegt, aber immer noch davor.
- Die Berechnung erfolgt durch Bestimmung des Wochentages des 25. Dezembers und dann das Zurückzählen der nötigen Tage, um den Sonntag der Vorwoche zu erhalten.

Berechnung der vorherigen Adventssonntage:

- **Dritter Advent**: Ein Sonntag eine Woche vor dem vierten Advent.
- **Zweiter Advent**: Ein Sonntag zwei Wochen vor dem vierten Advent.
- **Erster Advent**: Ein Sonntag drei Wochen vor dem vierten Advent.

Code zur Berechnung der Adventssonntage:

Im folgenden Code wird die `get_advent_sundays`-Funktion verwendet, um die vier Adventssonntage für ein gegebenes Jahr zu berechnen.

```
#!/usr/bin/env python3

#test_adventssonntage.py

from datetime import date, timedelta

def get_advent_sundays(year):
    """
    Berechnet die Adventssonntage für das gegebene Jahr.
    :return: Dictionary mit Adventssonntagen.
    """

    christmas = date(year, 12, 25)
    # Berechnung des 4. Advent (letzter Sonntag vor Weihnachten)
    fourth_advent = christmas - timedelta(days=(christmas.weekday() + 1))
    third_advent = fourth_advent - timedelta(weeks=1)
    second_advent = third_advent - timedelta(weeks=1)
    first_advent = second_advent - timedelta(weeks=1)

    return {

        1: first_advent,
        2: second_advent,
```

```
        3: third_advent,
        4: fourth_advent
    }


if __name__=='__main__':
    current_year = date.today().year
    advent_sundays = get_advent_sundays(current_year)
    for sonntag in advent_sundays:
        print(f"Advent {sonntag}: {advent_sundays[sonntag]}")
```

Für das Jahr 2024 werden die Adventssonntage wie folgt berechnet:

- **1. Advent**: 2024-12-01
- **2. Advent**: 2024-12-08
- **3. Advent**: 2024-12-15
- **4. Advent**: 2024-12-22

Erläuterung der Codezeile

```
fourth_advent = christmas - timedelta(days=(christmas.weekday() + 1))
```

Berechnet das Datum des **vierten Adventssonntags**. Hier ist eine detaillierte Erklärung:

Hintergrund:

- `christmas`: Das Datum des 25. Dezember für das gegebene Jahr.
- `christmas.weekday()`: Diese Methode liefert den Wochentag für das Datum von Weihnachten. Die Wochentage werden als Integer-Werte zurückgegeben:
  - Montag = 0,
  - Dienstag = 1,
  - ...
  - Sonntag = 6.
- **Adventssonntage**: Der vierte Advent ist immer der letzte Sonntag vor dem 25. Dezember.

Funktionsweise:

1. `christmas.weekday()`: Gibt den Wochentag von Weihnachten zurück. Zum Beispiel:
   - Wenn Weihnachten an einem Montag (0) ist, gibt die Methode `0` zurück.
   - Wenn Weihnachten an einem Freitag (4) ist, gibt die Methode `4` zurück.

2. `christmas.weekday() + 1`: Um den Abstand vom nächsten Sonntag zu berechnen, wird `1` addiert. Der Grund:
    - Ein Sonntag hat den Wert `6` in `weekday()`. Um den letzten Sonntag zu erreichen, muss die Anzahl der Tage bis zum vorherigen Sonntag berechnet werden, was durch `weekday() + 1` erfolgt.
3. `timedelta(days=(christmas.weekday() + 1))`: Erstellt ein Zeitintervall in Tagen, das genau die Anzahl der Tage bis zum vorherigen Sonntag umfasst.
4. `christmas - timedelta(...)`: Subtrahiert das Zeitintervall von `christmas`, um den Sonntag vor Weihnachten zu berechnen.

Beispiel:

Angenommen, Weihnachten ist am **25. Dezember 2023**, was ein Montag ist:

- `christmas.weekday()` ergibt `0` (Montag).
- `christmas.weekday() + 1` ergibt `1` (der Abstand zum Sonntag davor).
- `timedelta(days=1)` ergibt ein Intervall von einem Tag.
- `christmas - timedelta(days=1)` gibt den **24. Dezember 2023**, den vierten Adventssonntag.

Wenn Weihnachten an einem Samstag (25. Dezember 2021) liegt:

- `christmas.weekday()` ergibt `5`.
- `christmas.weekday() + 1` ergibt `6` (der Abstand zum vorherigen Sonntag).
- `timedelta(days=6)` wird vom Datum subtrahiert, um den Sonntag, den 19. Dezember 2021, zu berechnen.

2. Berechnung des Vorwiderstandes für LEDs

Wenn man LEDs an den GPIO-Pins des Raspberry Pi anschließt, ist es notwendig, einen Widerstand in Reihe mit der LED zu schalten, um zu verhindern, dass zu viel Strom durch die LED fließt und sie beschädigt wird. Der Wert des Widerstandes hängt von der LED und der Betriebsspannung ab.

Gegebene Werte:

- **Versorgungsspannung (Vcc):** 3.3 V (typisch für Raspberry Pi GPIO)
- **Durchlassspannung der roten LEDs (Vf):** ca. 2.0 V
- **LED-Strom (If):** 20 mA (typisch, maximaler Strom der LED)

Formel für den Vorwiderstand:

$$R = \frac{Vcc - Vf}{If}$$

Berechnung:

$$R = \frac{3.3\,\mathrm{V} - 2.0\,\mathrm{V}}{0.02\,\mathrm{A}} = \frac{1.3\,\mathrm{V}}{0.02\,\mathrm{A}} = 65\,\Omega$$

Da 65 Ω ein ungewöhnlicher Wert ist, wird der nächsthöhere Standardwert gewählt: **68 Ω**, weil
in meinem Vorrat **68 Ω** nicht vorhanden ist, nehme ich **100 Ω**, dann leuchtet die LED nicht ganz so hell.

---

3. Schaltplan und Anschluss der LEDs

Im folgenden Schaltplan ist der Anschluss der LEDs an die GPIO-Pins des Raspberry Pi beschrieben. Jede LED ist in Reihe mit einem 100-Ω-Widerstand geschaltet.

Anschlussplan:

- **GPIO 17 (Pin 11)**: LED 1 (erster Advent)
- **GPIO 27 (Pin 13)**: LED 2 (zweiter Advent)
- **GPIO 22 (Pin 15)**: LED 3 (dritter Advent)
- **GPIO 5 (Pin 29)**: LED 4 (vierter Advent)

Der Anschluss erfolgt wie folgt:

1. Verbinde den Anoden-Pin der LED (langer Pin) mit dem Widerstand.
2. Verbinde das andere Ende des Widerstands mit dem entsprechenden GPIO-Pin des Raspberry Pi.
3. Verbinde den Kathoden-Pin der LED (kurzer Pin) mit dem GND (Ground) des Raspberry Pi.

Schaltplan:

Korrekte Verdrahtung:
Die LED wird so angeschlossen, dass der GPIO den Stromfluss steuert. Die LED ist zwischen GPIO und Ground geschaltet, mit einem Vorwiderstand, der den Strom begrenzt.

Aufbau für jede LED:

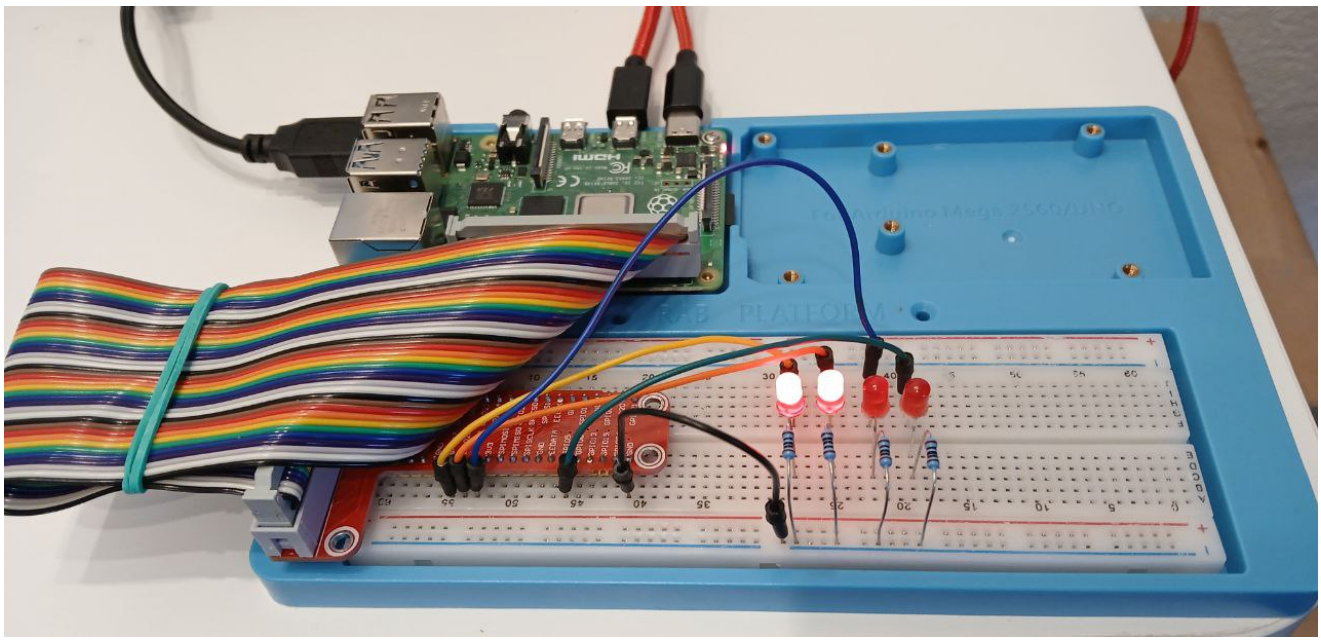
Der GPIO-Pin steuert den Stromfluss (HIGH: Strom fließt, LOW: kein Stromfluss).
Der Vorwiderstand wird zwischen GPIO und Anode (langes Bein) der LED geschaltet.
Die Kathode (kurzes Bein) der LED wird direkt mit Ground (GND) verbunden.
Schaltung (pro LED):

GPIO ---[100 Ω]---|>|--- GND

Verdrahtung für den Adventskranz:

LED-Nr. GPIO-Pin Widerstand Verbindung
LED 1 GPIO 17 100 Ω Anode: GPIO 17, Kathode: GND
LED 2 GPIO 27 100 Ω Anode: GPIO 27, Kathode: GND
LED 3 GPIO 22 100 Ω Anode: GPIO 22, Kathode: GND
LED 4 GPIO 5 100 Ω Anode: GPIO 5, Kathode: GND

100 Ω ist der Vorwiderstand, der den Strom auf ein sicheres Niveau begrenzt.
Warum diese Verdrahtung?
GPIO kann keine Spannung liefern: Die GPIOs des Raspberry Pi liefern maximal 3,3 V.
Daher fließt der Strom von GPIO zur LED und endet bei GND.
Vorwiderstand notwendig: Ohne Widerstand würde die LED zu viel Strom ziehen und den GPIO-Pin oder die LED beschädigen.
GND als Rückführung: Die LED benötigt eine Rückführung zum GND, damit ein Stromkreis geschlossen wird.



Testprogramm für die Verdrahtung

```
import RPi.GPIO as GPIO
import time

"""
```

```
    GPIO 17 ---[100 Ω]---|>|--- GND  (LED 1)
    GPIO 27 ---[100 Ω]---|>|--- GND  (LED 2)
    GPIO 22 ---[100 Ω]---|>|--- GND  (LED 3)
    GPIO 5  ---[100 Ω]---|>|--- GND  (LED 4)
    |>| symbolisiert die LED.
    100 Ω ist der Vorwiderstand, der den Strom auf ein sicheres Niveau
begrenzt.
"""


# GPIO-Pins für die LEDs

LED_PINS = {

    1: 17,  # LED 1 an GPIO 17
    2: 27,  # LED 2 an GPIO 27
    3: 22,  # LED 3 an GPIO 22
    4: 5    # LED 4 an GPIO 5
}


def setup_gpio():
    """Initialisiert die GPIO-Pins."""
    GPIO.setmode(GPIO.BCM)
    for pin in LED_PINS.values():
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, GPIO.LOW)  # LEDs ausschalten


def clear_gpio():
    """Räumt die GPIO-Konfiguration auf."""
    GPIO.cleanup()


def test_leds():
    """Testet alle LEDs nacheinander."""

    for led, pin in LED_PINS.items():
        GPIO.output(pin, GPIO.HIGH)  # LED an
        print(f"LED {led} an (GPIO {pin})")
        time.sleep(1)  # 1 Sekunde warten
        GPIO.output(pin, GPIO.LOW)   # LED aus
        print(f"LED {led} aus (GPIO {pin})")
        time.sleep(1)  # 1 Sekunde warten

def main():

    """Hauptprogramm für den LED-Test."""
    setup_gpio()
```

```
    try:
        test_leds()
    finally:
        clear_gpio()

if __name__ == "__main__":

    main()
```

Code für den Advenzkranz

```
from datetime import date, timedelta
import RPi.GPIO as GPIO
import time  # Importiere das time-Modul für Verzögerungen


"""
    GPIO 17 ---[100 Ω]---|>|--- GND  (LED 1)
    GPIO 27 ---[100 Ω]---|>|--- GND  (LED 2)
    GPIO 22 ---[100 Ω]---|>|--- GND  (LED 3)
    GPIO 5  ---[100 Ω]---|>|--- GND  (LED 4)
    |>| symbolisiert die LED.
    100 Ω ist der Vorwiderstand, der den Strom auf ein sicheres Niveau
begrenzt.
"""


# GPIO-Pins für die LEDs

LED_PINS = {

    1: 17,  # LED 1 an GPIO 17
    2: 27,  # LED 2 an GPIO 27
    3: 22,  # LED 3 an GPIO 22
    4: 5    # LED 4 an GPIO 5
}


def setup_gpio():
    """Initialisiert die GPIO-Pins."""

    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)  # Unterdrücke Warnungen
    for pin in LED_PINS.values():
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, GPIO.LOW)  # LEDs ausschalten

def clear_gpio():
```

```
    """Räumt die GPIO-Konfiguration auf."""
    GPIO.cleanup()

def control_leds(active_leds):
    """
    Steuert die LEDs.
    :param active_leds: Liste der aktiven LEDs (1 bis 4)
    """

    for led, pin in LED_PINS.items():
        if led in active_leds:
            GPIO.output(pin, GPIO.HIGH)  # LED an
            print(f"LED {led} an (GPIO {pin})")  # Debug-Ausgabe
        else:
            GPIO.output(pin, GPIO.LOW)   # LED aus
            print(f"LED {led} aus (GPIO {pin})")  # Debug-Ausgabe


def get_active_leds(test_date=None):

    """
    Gibt die aktiven LEDs basierend auf dem aktuellen oder angegebenen Datum
zurück.
    :param test_date: Optionales Testdatum im Format YYYY-MM-DD.
    :return: Liste der aktiven LEDs.
    """
    # Aktuelles Datum verwenden, falls kein Testdatum angegeben ist
    today = test_date or date.today()
    # Adventssonntage berechnen
    year = today.year
    advent_sundays = get_advent_sundays(year)
    # Debug-Ausgabe der berechneten Adventssonntage
    print(f"Berechnete Adventssonntage für {year}:")

    for i in range(1, 5):
        print(f"Advent {i}: {advent_sundays[i]}")
    # Bestimmen der aktiven LEDs basierend auf dem Datum

    active_leds = []
    for i in range(1, 5):
        if today >= advent_sundays[i]:
            active_leds.append(i)
    return active_leds

def get_advent_sundays(year):
    """
    Berechnet die Adventssonntage für das gegebene Jahr.
    :return: Dictionary mit Adventssonntagen.
    """
```

```
    christmas = date(year, 12, 25)
    # Berechnung des 4. Advent (letzter Sonntag vor Weihnachten)
    fourth_advent = christmas - timedelta(days=(christmas.weekday() + 1))
    third_advent = fourth_advent - timedelta(weeks=1)
    second_advent = third_advent - timedelta(weeks=1)
    first_advent = second_advent - timedelta(weeks=1)

    return {
        1: first_advent,
        2: second_advent,
        3: third_advent,
        4: fourth_advent
    }


def main(test_date=None):

    """Hauptprogramm."""

    setup_gpio()
    try:

        while True:  # Endlosschleife für kontinuierliche Ausführung
            active_leds = get_active_leds(test_date)
            print(f"Aktive LEDs: {active_leds}")
            control_leds(active_leds)  # Funktion zur Steuerung der LEDs
aufrufen
            time.sleep(60)  # Warte 60 Sekunden bevor die nächste
Überprüfung

    except KeyboardInterrupt:
        print("Programm wurde durch Benutzer beendet.")
    finally:

        clear_gpio()

if __name__ == "__main__":
    # Testmodus aktivieren, z.B. für den 2. Advent (Testdatum)
    test_date = date(2024, 12,22 )  # Beispiel: 2. Advent 2024


    main(test_date)
```

4. Code-Erklärung

Der Code steuert das Einschalten der LEDs basierend auf dem Adventssonntag. Er verwendet eine `while True`-Schleife, die alle 60 Sekunden die aktiven LEDs überprüft und

ansteuert.

- **setup_gpio**: Initialisiert die GPIO-Pins und setzt alle LEDs auf "aus".
- **clear_gpio**: Räumt die GPIO-Konfiguration auf, wenn das Programm beendet wird.
- **control_leds**: Steuert die LEDs, indem sie entweder eingeschaltet (HIGH) oder ausgeschaltet (LOW) werden, abhängig davon, ob die LED im `active_leds`-Array enthalten ist.
- **get_active_leds**: Berechnet die aktiven LEDs basierend auf dem Datum, das entweder das aktuelle Datum oder ein angegebenes Testdatum sein kann.
- **get_advent_sundays**: Berechnet die vier Adventssonntage für das angegebene Jahr.

Beispielausgabe:

- **Berechnete Adventssonntage für 2024**:
  - Advent 1: 2024-12-01
  - Advent 2: 2024-12-08
  - Advent 3: 2024-12-15
  - Advent 4: 2024-12-22

Aktive LEDs: [1, 2] (An den ersten beiden Adventssonntagen leuchten LED 1 und LED 2)

```
from datetime import date, timedelta
import RPi.GPIO as GPIO
import time  # Importiere das time-Modul für Verzögerungen

# GPIO-Pins für die LEDs
LED_PINS = {
    1: 17,  # LED 1 an GPIO 17
    2: 27,  # LED 2 an GPIO 27
    3: 22,  # LED 3 an GPIO 22
    4: 5    # LED 4 an GPIO 5
}

def setup_gpio():
    """Initialisiert die GPIO-Pins."""
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)  # Unterdrücke Warnungen
    for pin in LED_PINS.values():
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, GPIO.LOW)  # LEDs ausschalten

def clear_gpio():
    """Räumt die GPIO-Konfiguration auf."""
    GPIO.cleanup()

def control_leds(active_leds):
```

```
    """
    Steuert die LEDs.
    :param active_leds: Liste der aktiven LEDs (1 bis 4)
    """
    for led, pin in LED_PINS.items():
        if led in active_leds:
            GPIO.output(pin, GPIO.HIGH)  # LED an
            print(f"LED {led} an (GPIO {pin})")  # Debug-Ausgabe
        else:
            GPIO.output(pin, GPIO.LOW)   # LED aus
            print(f"LED {led} aus (GPIO {pin})")  # Debug-Ausgabe

def get_active_leds(test_date=None):
    """
    Gibt die aktiven LEDs basierend auf dem aktuellen oder angegebenen Datum
zurück.
    :param test_date: Optionales Testdatum im Format YYYY-MM-DD.
    :return: Liste der aktiven LEDs.
    """
    # Aktuelles Datum verwenden, falls kein Testdatum angegeben ist
    today = test_date or date.today()

    # Adventssonntage berechnen
    year = today.year
    advent_sundays = get_advent_sundays(year)

    # Debug-Ausgabe der berechneten Adventssonntage
    print(f"Berechnete Adventssonntage für {year}:")
    for i in range(1, 5):
        print(f"Advent {i}: {advent_sundays[i]}")

    # Bestimmen der aktiven LEDs basierend auf dem Datum
    active_leds = []
    for i in range(1, 5):
        if today >= advent_sundays[i]:
            active_leds.append(i)
    return active_leds

def get_advent_sundays(year):
    """
    Berechnet die Adventssonntage für das gegebene Jahr.
    :return: Dictionary mit Adventssonntagen.
    """
    christmas = date(year, 12, 25)
    # Berechnung des 4. Advent (letzter Sonntag vor Weihnachten)
    fourth_advent = christmas - timedelta(days=(christmas.weekday() + 1))
    third_advent = fourth_advent - timedelta(weeks=1)
    second_advent = third_advent - timedelta(weeks=1)
    first_advent = second_advent - timedelta(weeks=1)
```

```
    return {
        1: first_advent,
        2: second_advent,
        3: third_advent,
        4: fourth_advent
    }

def main(test_date=None):
    """Hauptprogramm."""
    setup_gpio()
    try:
        while True:  # Endlosschleife für kontinuierliche Ausführung
            active_leds = get_active_leds(test_date)
            print(f"Aktive LEDs: {active_leds}")
            control_leds(active_leds)  # Funktion zur Steuerung der LEDs
aufrufen
            time.sleep(60)  # Warte 60 Sekunden bevor die nächste
Überprüfung
    except KeyboardInterrupt:
        print("Programm wurde durch Benutzer beendet.")
    finally:
        clear_gpio()

if __name__ == "__main__":
    # Testmodus aktivieren, z.B. für den 2. Advent (Testdatum)
    test_date = date(2024, 12,22 )  # Beispiel: 2. Advent 2024
    main(test_date)
```

Übersicht

Dieser Python-Code steuert einen Adventskalender mit vier LEDs, die an einen Raspberry Pi angeschlossen sind. Die LEDs werden entsprechend der Adventssonntage aktiviert.

Modulimporte und Globale Variablen

```
from datetime import date, timedelta import RPi.GPIO as GPIO import time
```

- `datetime`: Für Datumsberechnungen
- `RPi.GPIO`: Zur Steuerung der GPIO-Pins des Raspberry Pi
- `time`: Für Verzögerungen im Programm

```
LED_PINS = { 1: 17,  # LED 1 an GPIO 17
                        2: 27,  # LED 2 an GPIO 27
```

```
                        3: 22,  # LED 3 an GPIO 22
                        4: 5    # LED 4 an GPIO 5 }
```

Dieses Dictionary ordnet jeder LED (1-4) einen GPIO-Pin zu.

Funktionen

setup_gpio()

```
def setup_gpio():
        GPIO.setmode(GPIO.BCM)
        GPIO.setwarnings(False)
        for pin in LED_PINS.values():
                GPIO.setup(pin, GPIO.OUT)
                GPIO.output(pin, GPIO.LOW)
```

Initialisiert die GPIO-Pins:- Setzt den GPIO-Modus auf BCM

- Deaktiviert Warnungen
- Konfiguriert die Pins als Ausgänge und schaltet sie aus

clear_gpio()

```
def clear_gpio():
        GPIO.cleanup()
```

Räumt die GPIO-Konfiguration auf, wenn das Programm beendet wird.

control_leds(active_leds)

```
def control_leds(active_leds):

    """
    Steuert die LEDs.
    :param active_leds: Liste der aktiven LEDs (1 bis 4)
    """

    for led, pin in LED_PINS.items():
        if led in active_leds:
            GPIO.output(pin, GPIO.HIGH)  # LED an
            print(f"LED {led} an (GPIO {pin})")  # Debug-Ausgabe
        else:
            GPIO.output(pin, GPIO.LOW)   # LED aus
            print(f"LED {led} aus (GPIO {pin})")  # Debug-Ausgabe
```

Steuert die LEDs basierend auf der Liste der aktiven LEDs:

- Schaltet die angegebenen LEDs ein
- Schaltet die anderen LEDs aus
- Gibt Debug-Informationen aus

get_active_leds(test_date=None)

```
def get_active_leds(test_date=None):

    """
    Gibt die aktiven LEDs basierend auf dem aktuellen oder angegebenen Datum
zurück.
    :param test_date: Optionales Testdatum im Format YYYY-MM-DD.
    :return: Liste der aktiven LEDs.
    """

    # Aktuelles Datum verwenden, falls kein Testdatum angegeben ist
    today = test_date or date.today()
    # Adventssonntage berechnen
    year = today.year
    advent_sundays = get_advent_sundays(year)
    # Debug-Ausgabe der berechneten Adventssonntage
    print(f"Berechnete Adventssonntage für {year}:")
    for i in range(1, 5):
        print(f"Advent {i}: {advent_sundays[i]}")
    # Bestimmen der aktiven LEDs basierend auf dem Datum

    active_leds = []

    for i in range(1, 5):
        if today >= advent_sundays[i]:
            active_leds.append(i)

    return active_leds
```

Bestimmt die aktiven LEDs basierend auf dem aktuellen oder einem Testdatum:

- Berechnet die Adventssonntage für das Jahr
- Gibt Debug-Informationen zu den Adventssonntagen aus
- Bestimmt, welche LEDs basierend auf dem Datum aktiv sein sollten

get_advent_sundays(year)

```
def get_advent_sundays(year):
    """
    Berechnet die Adventssonntage für das gegebene Jahr.
```

```
    :return: Dictionary mit Adventssonntagen.
    """

    christmas = date(year, 12, 25)

    # Berechnung des 4. Advent (letzter Sonntag vor Weihnachten)
    fourth_advent = christmas - timedelta(days=(christmas.weekday() + 1))
    third_advent = fourth_advent - timedelta(weeks=1)
    second_advent = third_advent - timedelta(weeks=1)
    first_advent = second_advent - timedelta(weeks=1)
    return {
        1: first_advent,
        2: second_advent,
        3: third_advent,
        4: fourth_advent
    }
```

Berechnet die Daten der vier Adventssonntage für ein gegebenes Jahr:

- Beginnt mit Weihnachten und rechnet rückwärts
- Gibt ein Dictionary mit den Daten der Adventssonntage zurück

main(test_date=None)

```
def main(test_date=None):

    """Hauptprogramm."""
    setup_gpio()
    try:
        while True:  # Endlosschleife für kontinuierliche Ausführung
            active_leds = get_active_leds(test_date)
            print(f"Aktive LEDs: {active_leds}")
            control_leds(active_leds)  # Funktion zur Steuerung der LEDs
aufrufen
            time.sleep(60)  # Warte 60 Sekunden bevor die nächste
Überprüfung

    except KeyboardInterrupt:
        print("Programm wurde durch Benutzer beendet.")
    finally:
        clear_gpio()
```

Hauptfunktion des Programms:

- Initialisiert die GPIO-Pins
- Läuft in einer Endlosschleife

- Aktualisiert die aktiven LEDs jede Minute
- Behandelt Benutzerunterbrechungen (Ctrl+C)
- Stellt sicher, dass die GPIO-Pins aufgeräumt werden

Programmausführung

```
if __name__ == "__main__":
    # Testmodus aktivieren, z.B. für den 2. Advent (Testdatum)
    test_date = date(2024, 12,22 )  # Beispiel: 2. Advent 2024

    main(test_date)
```

Startet das Hauptprogramm:

- Verwendet ein Testdatum (22. Dezember 2024) für Demonstrationszwecke
- Kann für den normalen Betrieb angepasst oder entfernt werden

Zusammenfassung

Dieser Code steuert einen Adventskalender mit vier LEDs, die an bestimmten GPIO-Pins eines Raspberry Pi angeschlossen sind. Er berechnet die Adventssonntage für das aktuelle Jahr und aktiviert die entsprechenden LEDs basierend auf dem aktuellen Datum. Das Programm läuft kontinuierlich und aktualisiert den Status der LEDs jede Minute. Es enthält auch Funktionen für Debugging und Testen mit einem spezifischen Datum.