



Raspberry PI Doku

▼ Szenario

Wir müssen was mit der Kühlung im Serverraum machen. Seitdem wir neue Server angeschafft haben, schalten sich die Geräte im Sommer wegen Überhitzung ab und im Winter wachsen dort Eiszapfen von der Decke. Die Kühlung wird über einen RaspberryPi gesteuert.

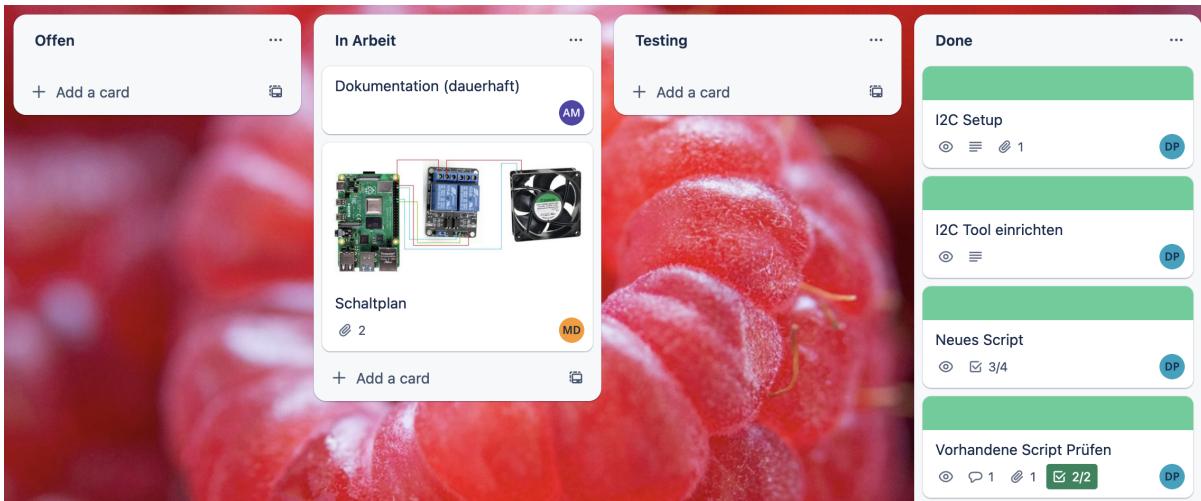
Das hat ein ehemaliger Azubi vor etwa drei Jahren entworfen und seitdem wurde nicht mal mehr das Betriebssystem angepackt. Ich glaube, das wurde damals zeitgesteuert programmiert. Das könnten Sie auf jeden Fall mal prüfen, das macht ja gar keinen Sinn. Moderne Systeme reagieren ja auf die Temperatur und nicht auf die Uhrzeit.

Sie finden auf dem Server alle Unterlagen, die wir zur aktuellen Steuerung haben. Damit könnten Sie sich erstmal einen Überblick verschaffen.

Entwickeln Sie bitte einen funktionierenden Prototypen und ganz wichtig dokumentieren Sie alles sorgfältig. Am besten in einem Dokument mit Schaltplänen, technischen Daten, alle Einstellungen, Programmcode und so weiter. Dann müssen wir beim nächsten Mal nicht wieder alles zusammensuchen.

▼ Projekt Management und Tools

Trello



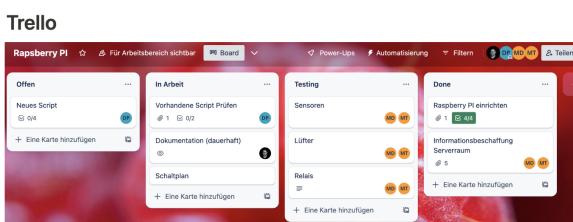
Für das Projektmanagement verwenden wir Trello um die Arbeitsaufträge in sinnvolle Arbeitspakete zu unterteilen. Somit ist es übersichtlicher.

Notion



Raspberry PI Doku

+ :: ▾ Projekt Management und Tools



Notion ist eine leistungsstarke Plattform, die es uns ermöglicht, unsere Dokumentation an einem zentralen Ort zu speichern und zu verwalten. Jeder im Team hat Zugriff auf die Dokumente und kann sie bearbeiten, kommentieren oder aktualisieren.

Ticket Refinement

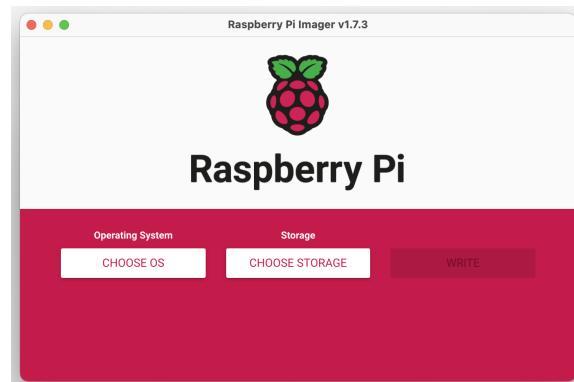
Während der Projekt wir haben Ticket-Refinement Prozess verwendet. Im agilen Projektmanagement, bei dem die Anforderungen und Details eines bestimmten Tickets (eine Aufgabe, ein Problem oder eine Funktion) weiter verfeinert werden, um sicherzustellen, dass es für das Team klar und umsetzbar ist

▼ Raspberry PI Setup

Wir haben eine headless Raspberry Pi OS-Version verwendet, die über den Raspberry Pi Manager installiert wurde.

Hier ist zum beachten

- SSH aktivieren
- WLAN-Name und Passwort einrichten
- Hostname einrichten

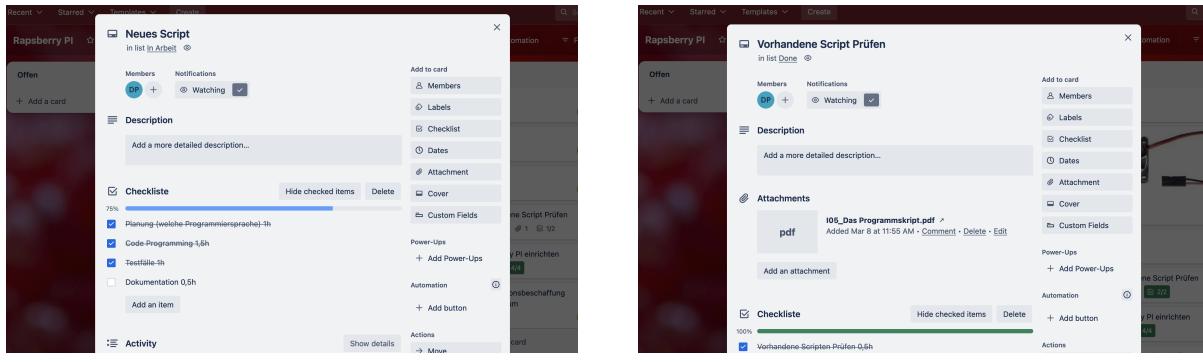


Statische IP Adresse

Eine statische IP-Adresse ist eine feste IP-Adresse, die einem Gerät zugewiesen wird und nicht verändert wird.

```
sudo nano /etc/dhcpcd.conf
_____
interface wlan0
static ip_address=192.168.0.10/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1
```

▼ Code



Neues Script

Vorhandene Script Prüfen

▼ Planung / Vorhandene Script Prüfen

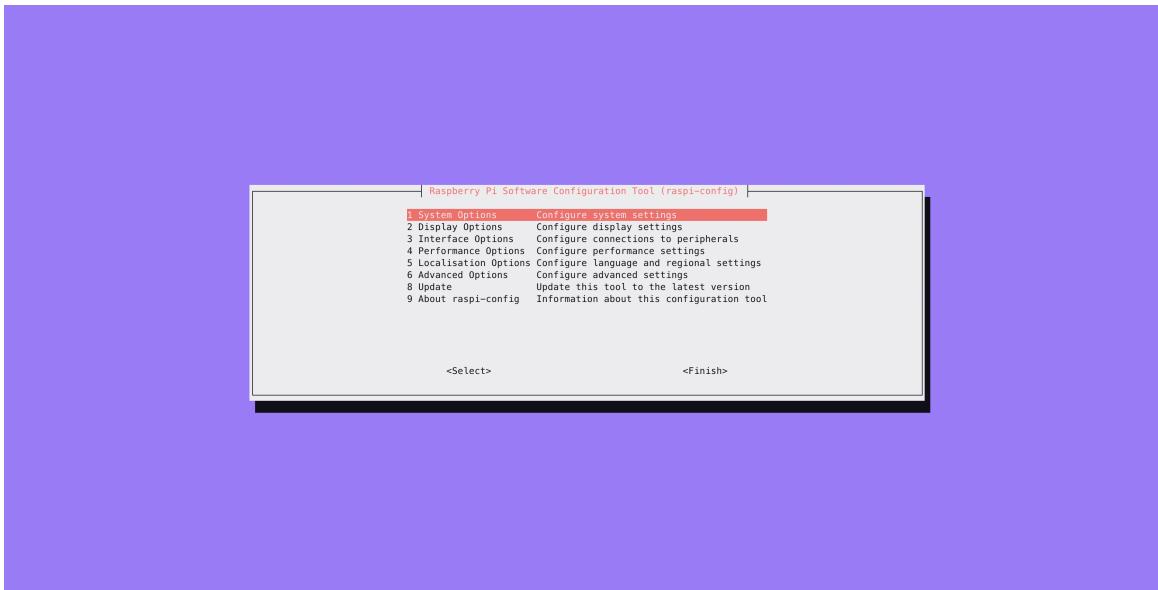
Das Problem des Skripts besteht darin, dass die Variable "sleep_timer" in der Schleife nicht definiert ist. Es sollte "interval_in_s" verwendet werden, wie es in der Zeile zuvor definiert wurde.

Der Raspberry muss dafür wie folgt dokumentiert werden:

```
#!/usr/bin/python3
import RPi.GPIO as GPIO
import time
# die GPIOs koennen über die PINNummer oder die GPIONummer
# angesprochen werden
GPIO.setmode(GPIO.BCM) # hier GPIO-Nummer
GPIO.setup(23, GPIO.OUT)
interval_in_s = 120.0
while(True):
    print("on")
    GPIO.output(23, GPIO.HIGH)
    time.sleep(interval_in_s) # sleep interval in Sekunden
    print("off")
    GPIO.output(23, GPIO.LOW)
    time.sleep(sleep_timer)
```

▼ Benötigte Libraries & Setup

I2C



I2C (Inter-Integrated Circuit) ist ein serieller Kommunikationsbus, der es ermöglicht, mehrere Geräte über nur zwei Leitungen miteinander zu verbinden

```
sudo raspi-config
```

I2C Tool

Mit I2C-Tools können Entwickler den Status des Busses überprüfen, I2C-Adressen scannen, I2C-Geräte lesen und schreiben sowie andere Diagnose- und Debugging-Aufgaben ausführen.

```
sudo apt-get install i2c-tools python-pip
```

I2C Tool hat uns bei dem folgenden Fehler geholfen

```
dpeci@dpeci:~ $ python3 test.py
Traceback (most recent call last):
  File "test.py", line 8, in <module>
    calibration_params = bme280.load_calibration_params(bus, address)
  File "/home/dpeci/.local/lib/python3.7/site-packages/bme280/__init__.py", line 154, in load_calibration_params
    compensation_params.dig_T1 = read.unsigned_short(0x88)
  File "/home/dpeci/.local/lib/python3.7/site-packages/bme280/reader.py", line 40, in unsigned_short
    return self._bus.read_word_data(self._address, register) & 0xffff
  File "/home/dpeci/.local/lib/python3.7/site-packages/smbus2/smbus2.py", line 474, in read_word_data
    ioctl(self._fd, I2C_SMBUS, msg)
OSError: [Errno 121] Remote I/O error
```

```
dpeci@peci:~ $ i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:          -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:          -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:          -- -- -- -- -- -- -- -- -- -- -- -- -- --
40:          -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:          -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:          -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:          -- -- -- -- -- -- 77
```

Pip

Pip ist ein Paketmanager für die Programmiersprache Python. Pip vereinfacht die Installation, Aktualisierung und Deinstallation von Python-Paketen

```
sudo apt update
sudo apt install python3-pip
```

Pakete

Die folgenden Pakete sind erforderlich, um Skript auszuführen.

```
pip install RPi.GPIO
pip install smbus2
pip install bme280
```

▼ Code

Temperature

Mit dem folgenden Code können wir die aktuelle Temperatur ausdrucken.

```
import bme280
temperature,pressure,humidity = bme280.readBME280All()

print "Temperature : ", temperature, "C"
print "Pressure : ", pressure, "hPa"
print "Humidity : ", humidity, "%"'
```

```
dpeci@peci:~ $ python3 test.py
360d8f77-fe34-4813-b817-64f2f50c0da3
2023-04-19 10:36:21.666861+00:00
23.22507948082566
1018.81740114712
31.09650845293443
compensated_reading(id=360d8f77-fe34-4813-b817-64f2f50c0da3, timestamp=2023-04-19 10:36:21.666861UTC, temp=23.225 °C, pressure=1018.82 hPa, humidity=31.10 % rH)
dpeci@peci:~ $
```

Neue Script

Dieses Python-Skript überwacht kontinuierlich die Temperatur mit einem BME280-Sensor und schaltet einen Lüfter ein oder aus, abhängig von der gemessenen Temperatur. Das Skript verwendet auch die RPi.GPIO-, smbus2- und bme280-Bibliotheken.

- Eine Konstante namens "TEMPERATURE" wird definiert, die eine Schwellenwert-Temperatur in Grad Celsius festlegt. Wenn die gemessene Temperatur diese Schwelle überschreitet, wird der Lüfter eingeschaltet.
- Die Variablen "port", "address" und "bus" werden initialisiert, um den I2C-Bus und die Adresse des BME280-Sensors zu konfigurieren.
- Eine Schleife wird definiert, die kontinuierlich die Temperatur misst und den Lüfter ein- oder ausschaltet, abhängig von der gemessenen Temperatur.

```
#!/usr/bin/python3

import RPi.GPIO as GPIO
import time
import smbus2
import bme280

# Requires: 'adafruit-circuitpython-bme280' from pip
TEMPERATURE = 23
port = 1
address = 0x77
bus = smbus2.SMBus(port)
calibration_params = bme280.load_calibration_params(bus, address)
fanPin = 23

GPIO.setmode(GPIO.BCM)

GPIO.setup(fanPin, GPIO.OUT)
check_interval = 1.0 # Seconds

def on():
    GPIO.output(fanPin, 1)

def off():
    GPIO.output(fanPin, 0)
```

```

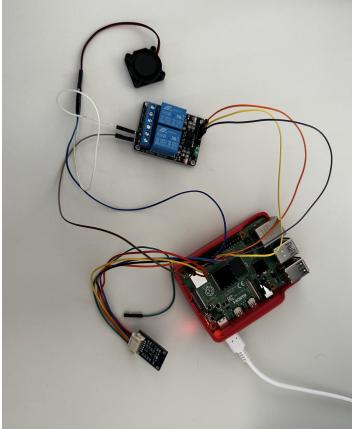
while True:
    sensorTemperature = data = bme280.sample(bus, address, calibration_params).temperature

    if sensorTemperature > TEMPERATURE:
        on()
    else:
        off()

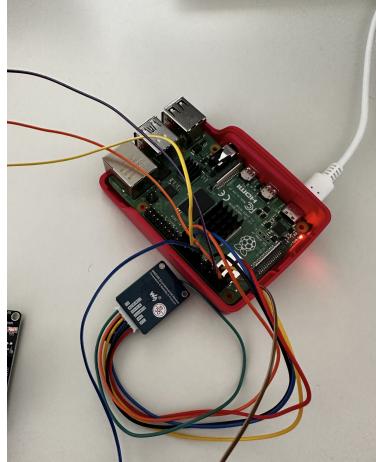
    time.sleep(check_interval)

```

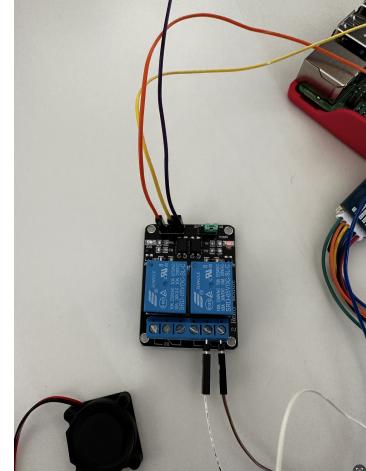
▼ Fotos



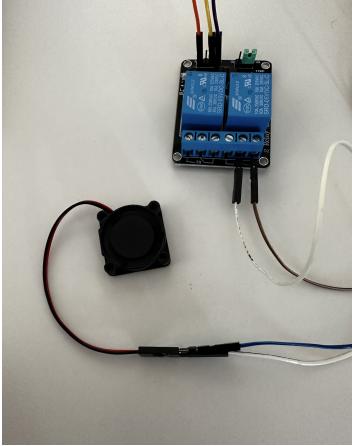
RasPi



RasPi mit Sensor



ResPi mit Lüfter und Relais



Lüfter und Relais

