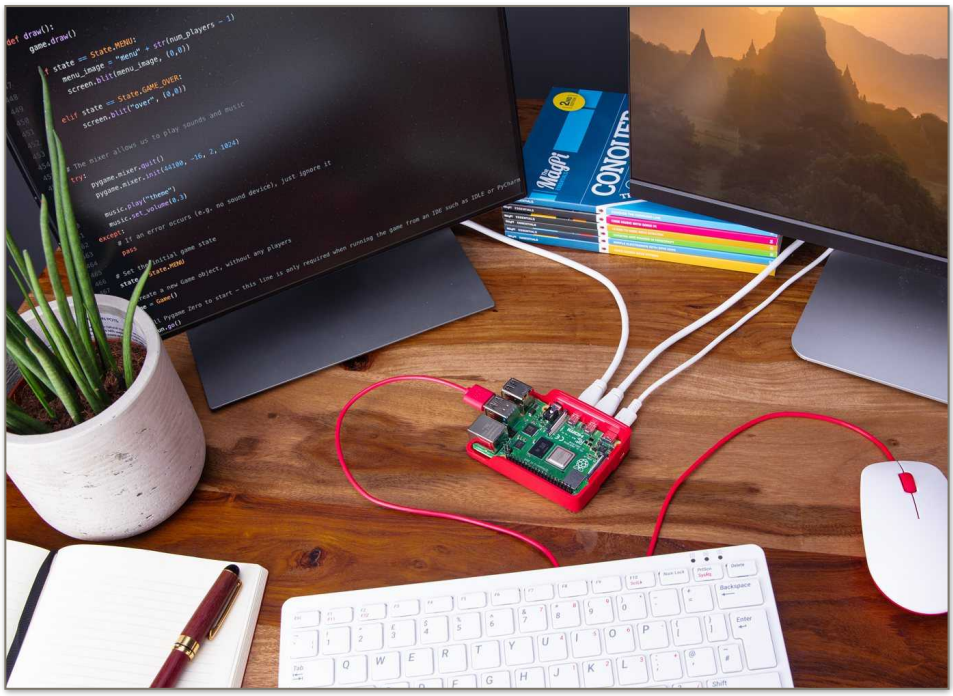
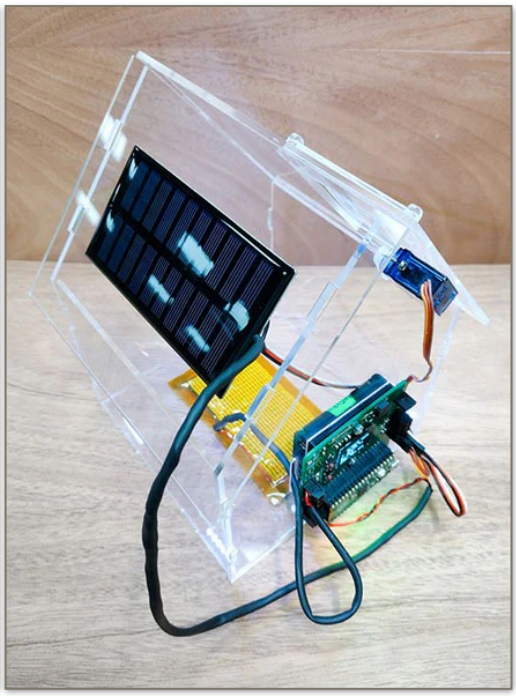
Aufgabenheft

Projekt: Smart Home Dashboard mit Raspberry Pi





INHALTE:

- Grundlagen des Raspberry Pi – Einrichtung, Betriebssystem, und erste Schritte
- Sensorik mit dem SCD41 – Messen von CO_2, Temperatur und Luftfeuchtigkeit
- Aktorik und Steuerung – LEDs und Relais für smarte Automatisierung
- Anwendungsentwicklung – Aufbau eines interaktiven Smart Home Dashboards

BESCHREIBUNG:

In diesem Aufgabenheft lernst du, wie du deinen Raspberry Pi einrichtest, einen SCD41-Sensor (zur Messung von CO_2, Temperatur und Luftfeuchtigkeit) anschließt und einfache Aktoren (LEDs und ein Relais) steuerst. Mit diesem Wissen kannst du ein eigenes Smart Home Dashboard bauen, das Umgebungsdaten misst und automatisch reagiert.

Dieses Heft ist sowohl als Lernmaterial als auch als Dokumentation der eigenen Fortschritte gedacht. Notizen, Ergänzungen und eigene Lösungen sind ausdrücklich erwünscht! Viel Spaß

AUFGABE 1: RASPBERRY PI GRUNDLAGEN

Damit dein Raspberry Pi später Sensoren auslesen und Aktoren ansteuern kann, musst du ihn zunächst als Entwicklungsplattform einrichten.

1. **Vorbereitung und Installation**
   Lade dir das aktuelle **Raspberry Pi OS** (z.B. über den Raspberry Pi Imager) herunter und schreibe das Image auf eine microSD-Karte. Aktiviere dabei direkt **SSH** in den erweiterten Optionen oder lege nach dem Schreiben des Images eine leere Datei namens `ssh` in das Boot-Verzeichnis der SD-Karte.

2. **Inbetriebnahme und SSH-Zugriff**
   - Setze die microSD-Karte in den Raspberry Pi ein und starte ihn.
   - Verbinde den Pi per **LAN oder WLAN** mit deinem Netzwerk.
   - Finde die **IP-Adresse** des Raspberry Pi heraus (z.B. im Router-Menü oder mit `hostname -I`).
   - Melde dich von deinem PC aus per **SSH** an, z.B. mit `ssh pi@192.168.xxx.xxx`. Standard-Benutzername ist häufig `pi`, das Passwort oft `raspberry` (falls nicht geändert).

3. **I²C-Schnittstelle aktivieren**
   - Gib im Terminal `sudo raspi-config` ein und wähle im Menü „Interfacing Options“ → „I2C“.
   - Aktiviere I²C und starte den Pi neu.
   - Überprüfe, ob die Tools installiert sind: `sudo apt install i2c-tools`.

4. **Wichtige Bibliotheken installieren**
   - Python-Bibliotheken für Hardware-Zugriff:
     - `RPi.GPIO` (`sudo apt install python3-rpi.gpio`)
     - `gpiozero` (`sudo apt install python3-gpiozero`)
     - `Adafruit Blinka` und `Adafruit CircuitPython SCD4x` (`pip3 install adafruit-blinka adafruit-circuitpython-scd4x`)

   - Erstelle anschließend einen Projektordner, z.B. `mkdir ~/smart-home-dashboard`.

Lückentext – trage die richtigen Begriffe ein (SDA, SCL, 3.3V, GND)

1. **Pin 3** am Raspberry Pi ist _____.
2. **Pin 5** am Raspberry Pi ist _____.
3. **Pin 1** am Raspberry Pi liefert _____.
4. **Pin 6** am Raspberry Pi ist _____.

*(Tipp: Schau dir ein **Pinout** des Raspberry Pi an oder nutze dein neues Wissen über I²C und Stromversorgung.)*

Wissensfrage (Modul 1)

Welche **Sicherheitsaspekte** musst du beachten, wenn du den Raspberry Pi über SSH zugänglich machst (z.B. Standard-Passwort ändern)?

Diskussionsaufgabe (Modul 1)

Diskutiere mit deinen Mitschüler*innen:

- Welche **Vorteile** hat der „Headless“-Betrieb (also ohne Monitor und Tastatur) über SSH?
- Welche **Nachteile** oder **Risiken** können dadurch entstehen?
- Welche Lösung würdest du vorschlagen, um den Raspberry Pi möglichst sicher und gleichzeitig bequem nutzen zu können?

AUFGABE 2: SENSORIK – SCD41-SENSOR

Modul 2: Sensorik – SCD41-Sensor

Jetzt rüsten wir deinen Pi mit „Sinnesorganen“ aus: Der **SCD41-Sensor** misst CO_2, Temperatur und Luftfeuchtigkeit. Damit kannst du beispielsweise feststellen, wann gelüftet werden sollte.

1. **Verkabelung (I²C)**
   - Schließe **VIN** des Sensors an **3.3V** (Pin 1) an.
   - Verbinde **GND** des Sensors mit **GND** (Pin 6).
   - **SCL** des Sensors kommt an **GPIO 3** (Pin 5).
   - **SDA** an **GPIO 2** (Pin 3).

   Achte unbedingt darauf, dass du SDA und SCL nicht vertauschst und dass die Steckverbindungen fest sitzen.

2. **Sensor auslesen**
   - Teste zunächst mit `i2cdetect -y 1`, ob der Sensor unter der Adresse **0x62** erkannt wird.
   - Erstelle dann eine Python-Datei, z.B. `sensor_scd41.py`.
   - Verwende die Adafruit-Bibliotheken:

```
python
Kopieren
import time
import board
import adafruit_scd4x

i2c = board.I2C()  # Standard-I2C auf GPIO 2/3
scd = adafruit_scd4x.SCD4X(i2c)
scd.start_periodic_measurement()

while True:
    if scd.data_ready:
        print(f"CO2: {scd.CO2} ppm | Temp: {scd.temperature:.1f} °C |
Feuchte: {scd.relative_humidity:.1f} %")
    time.sleep(5)
```

   - Starte das Skript mit `python3 sensor_scd41.py` und beobachte die Messwerte.

3. **Debugging-Tipps**
   - Falls du keine Werte bekommst, kontrolliere nochmals die **Verkabelung** und prüfe die **I²C-Aktivierung**.
   - Teste die Adresse mit `i2cdetect -y 1`.
   - Achte darauf, dass du **Python3** und die richtigen Bibliotheken verwendest.

Kreativ-Aufgabe

Erkläre in **drei Sätzen**, warum ein CO_2-Sensor in einem Klassenzimmer oder Jugendzimmer sinnvoll sein kann. Überlege, welche Folgen es hat, wenn der CO_2-Wert zu hoch ist.

Wissensfrage

Nenne **mindestens zwei typische Fehlerquellen**, wenn der Sensor keine Daten liefert, obwohl das Skript korrekt ist. Was kannst du in diesen Fällen überprüfen oder verändern?

Diskussionsaufgabe

Diskutiere mit deinen Mitschüler*innen:

- Ist es sinnvoll, die Luftqualität **nur** über CO_2-Werte zu bestimmen, oder sollte man noch weitere Faktoren (z.B. Feinstaub, VOC) messen?
- Inwiefern kann Technik den Menschen helfen, gesündere Lebensräume zu schaffen, und wo könnten **Grenzen** oder **Nachteile** liegen?

AUFGABE 3: AKTOREN – LEDS UND RELAIS

Ein Sensor alleine reicht nicht, um aktiv auf deine Umgebung einzuwirken. Deshalb lernst du hier, wie du **LEDs** (zur Anzeige) und ein **Relais** (zum Schalten größerer Lasten) steuerst.

LED-Steuerung

1. **LED anschließen**
   - Wähle einen **GPIO-Pin** (z.B. GPIO 17, Pin 11).
   - Verbinde die **Anode** (längeres Bein) der LED über einen **Vorwiderstand** (220 Ω) mit GPIO 17.
   - Schließe die **Kathode** (kürzeres Bein) an GND (Pin 6).
2. **LED-Blinken programmieren**
   - Erstelle die Datei `led_control.py`.
   - Nutze `gpiozero`:

     ```
     python
     Kopieren
     from gpiozero import LED
     from time import sleep

     led = LED(17)
     while True:
         led.on()
         sleep(0.5)
         led.off()
         sleep(0.5)
     ```

   - Starte mit `python3 led_control.py`. Die LED sollte jetzt blinken.

Relais-Steuerung

1. **Relais verkabeln**
   - **IN** des Relais an einen GPIO (z.B. GPIO 23, Pin 16).
   - **VCC** an 5V, **GND** an Pi-GND.
   - Last (z.B. Heizmatte, Lüfter oder Lampe) in Reihe mit dem Relais schalten (COM/NO).
   - Bei Netzspannung (230 V) unbedingt Sicherheitsregeln beachten und ggf. Fachleute hinzuziehen!
2. **Relais-Code**

```
import RPi.GPIO as GPIO
import time

RELAY_PIN = 23
GPIO.setmode(GPIO.BCM)
GPIO.setup(RELAY_PIN, GPIO.OUT)

try:
    while True:
        GPIO.output(RELAY_PIN, GPIO.HIGH)  # Relais an
        print("Relais an (Heizmatte EIN)")
        time.sleep(5)
        GPIO.output(RELAY_PIN, GPIO.LOW)   # Relais aus
        print("Relais aus (Heizmatte AUS)")
        time.sleep(5)
except KeyboardInterrupt:
    GPIO.cleanup()
```

1. **Wichtige Hinweise**
    - Manche Relais sind „active-low“, d.h. du musst `GPIO.output(RELAY_PIN, GPIO.LOW)` verwenden, um das Relais einzuschalten.
    - Achte auf eine **gemeinsame Masse** (GND) von Pi und Relais-Modul.
    - Teste das Klicken des Relais oder die Relais-LED als Indikator.

Zuordnungsaufgabe:

Ordne folgende Begriffe den richtigen Erklärungen zu:

1. **Vorwiderstand**
2. **Anode**
3. **active-low**
4. **COM**
5. **NO**

A. „Der Kontakt am Relais, der normalerweise offen ist und sich erst beim Einschalten schließt.“
B. „Die Anschlussseite einer LED, die positiv gepolt werden muss.“
C. „Begriff für eine Schaltung, bei der ein niedriger Pegel (0 V) ein Signal aktiviert.“
D. „Begrenzt den Strom durch eine LED, damit sie nicht durchbrennt.“
E. „Der gemeinsame Anschluss eines Relais, an den die Last angeschlossen wird.“

Wissensfrage (Modul 3)

Erkläre den **Unterschied** zwischen „active-high“ und „active-low“ Relaismodulen. Warum kann es sein, dass dein Code umgekehrt reagieren muss?

Diskussionsaufgabe (Modul 3)

Diskutiere mit deinen Mitschüler*innen:

- Wo siehst du **Vorteile** darin, alltägliche Geräte (Heizung, Licht, Lüfter) automatisiert zu schalten?
- Gibt es auch **ethische** oder **praktische** Bedenken, wenn Maschinen für uns Entscheidungen treffen (z.B. „Heizung an/aus“)?
- Sollte man manche Dinge lieber **manuell** regeln?

AUFGABE 4 - INFLUXDB

Aufgabe 1 – Erstelle deinen ersten Bucket

Starte den InfluxDB-Client und öffne im Browser deiner Wahl die URL http://localhost:8086. Melde dich an und erstelle einen neuen Bucket. Notiere den Namen deines Buckets.

Name:

Aufgabe 2 – Generiere einen API Token

Für deinen neu erstellten Bucket benötigst du einen API Token. Erstelle einen Custom Token und vergib eine aussagekräftige Beschreibung. Wähle dabei den erstellten Bucket aus.

Wichtig: Speichere oder kopiere den API Token sofort, da er später nicht mehr angezeigt wird.

Beschreibung:

API Token:

Aufgabe 3 – Script verbindung

Öffne das Python Script „influxdb“ in einer IDE und fülle für die folgende Variablen die richtigen Werte aus:

url = “ _____“

token = “ _____“

org = “ _____“

bucket =“_____“

Nach erfolgreichem einfügen, starte das Script und schau in deinem Bucket auf InfluxDB ob sich was verändert hat.

Tipp: Rechts unter dem Graphen ist ein Icon welches eine Uhr symbolisieren soll, dahinter steht „Past …“, wähle z.B. Past 5m für die gemessenen Werte der letzten 5 Minuten. Nun sollte der Graph zu sehen sein.

AUFGABE 5 - GRAFANA

Aufgabe 1 – Dashboard erstellen

Grafana sollte nach der Installation auf dem Rechner laufen, es benötigt standardmäßig keinen aktiven Start. Gehe auf http://localhost:3000 und melde dich in Grafana an. Gehe Auf Dashboards und erstelle ein neues Dashboard. Speichere das Dashboard und notiere dir dein Title und die Description.

Title:

Description:

Aufgabe 2 – Panel erstellen

Gehe auf dein erstelltes Dashboard und drücke auf +Add visualization.

Wähle für die „Data source“ die InfluxDB aus.

Gib dem Panel ein Titel und eine Beschreibung und speichere das Dashboard

Titel:

Beschreibung:

Aufgabe 3 – Abfragen Ausfüllen

3.1 Es soll die Temperatur der letzten Stunde abgerufen werden, in der Datenbank ist sie unter „temperature“ gespeichert.

```
from(bucket: "SensorData")
  |> range(start: -1h)
  |> filter(fn: (r) => r._measurement == "sensor_data")
  |> filter(fn: (r) => r._field == "____________")
  |> filter(fn: (r) => r.sensor_id == "Sensor_1")
```

3.2 Der Direktor möchte alle Werte auf einem Panel sehen. Da sein Mitarbeiter zu langsam war, hat er ihn gekündigt. Nun sollst du die Arbeit erledigen. Zum Glück findest du einen Datenauszug:

temperature 20C , humidity 38% , co2 500ppm

temperature 21C , humidity 41% , co2 600ppm

temperature 22C , humidity 35% , co2 550ppm

```
from(bucket: "SensorData")
  |> range(start: -1h) // Daten der letzten Stunde
  |> filter(fn: (r) => r._measurement == "sensor_data")
  |> filter(fn: (r) => r._field == "____________"or r._field == "_________"or r._field == "_____")
  |> filter(fn: (r) => r.sensor_id == "Sensor_1")
```

Aufgabe 4 – Einheiten Verbinden

4.1

1: Temperatur	a) Prozent
2: Co2	b) Meter Pro Sekunde
3: Luftfeuchtigkeit	c) Celsius
4: Beschleunigung	d) Minuten
5: Zeit	e) Teile pro Million

4.2 Welche der Einheiten gehören zu Temperatur?
Kelvin, Petabytes, Stunden, Celsius, Watt, Hertz, Bar, Fahrenheit

4.3 Welche der Einheiten gehören zu Währung?
Gigahertz, Meilen pro Stunde, Euro, Voltage, US-Dollar

Aufgabe 5 – Gestaltung

Öffne die „simulation.py“ in deiner IDE und konfiguriere sie passend mit deiner InfluxDB. Erstelle ein neues Dashboard in Grafana. Erstelle ein Panel, welches den Wert anzeigt. 5.1 Suche unter Search options nach „Color scheme“, wähle Single Color und wähle eine Farbe aus.

5.2 Suche unter Search options nach „Graph styles“, setze den Gradient mode auf Opacity und passe den Fill opacity nach belieben an.

5.3 Setze Line interpolation auf smooth

5.4 Probiere dich weiter in den Einstellungen aus und schaue was sich verändert. Notiere deine Bemerkungen.

Aufgabe 6 – Darstellung

6.1 Nenne 5 Darstellungsarten die Grafana anbietet:

1: _____

2: _____

3: _____

4: _____

5: _____

6.2 Welche Darstellungsart eignet sich für…

Vermögenswachstum über Zeit: _____

Geschwindigkeit des Autos: _____

Mädchen-Jungs Verhältnis in der Klasse: _____

Klassenbuch: _____

Mitgliederanzahl: _____

6.3 Starte die „simulation.py“ aus Aufgabe 5. Probiere verschiedene Darstellungsformen aus. Erstelle zwei Panels, die den gleichen Wert anzeigen, sie dürfen nicht in der gleichen Darstellungsform sein.

Aufgabe 7 - Thresholds

Öffne die „temperature.py“ in deiner IDE und konfiguriere sie passend mit deiner InfluxDB. Erstelle ein neues Dashboard in Grafana. Erstelle ein Panel, welches den Temperaturwert anzeigt.

7.1 Öffne das Informationsblatt und trage die passenden Werte ein

Kalt, alles unter: ____ °C

Ideal zwischen: ____ °C - ____ °C

Warm alles über: ____ °C

Zu heiß, alles über: ____ °C

7.2 Bearbeite das Panel, suche unter Search options nach Thresholds. Dort wählst du unter „Show tresholds“ die Option „As filled regions“. Erstelle mithilfe der Lösung aus 7.1 alle benötigten Tresholds.

Bonusaufgabe

Öffne die „multipanel.py“ in deiner IDE und konfiguriere sie passend mit deiner InfluxDB. Erstelle ein neues Dashboard in Grafana. Jeder Wert soll im eigenen Panel als Liniendiagramm dargestellt werden. Schaue in der InfluxDB welche Werte gespeichert werden.