

InfluxDB & Grafana

AUFGABENHEFT

Sensorik



Inhaltsverzeichnis

Inhalt

Vorwort.....	3
Sensorik.....	4
Lernziele Sensorik.....	5
Benötigte Bauteile.....	6
Sensorik – Was ist ein Sensor?.....	7
Sensorik – Was ist ein Raspberry Pi?.....	8
Sensorik – Sensoren & Raspberry Pi.....	9
Sensorik – Was ist SSH?.....	11
Sensorik – SCD41 Sensor.....	12
Sensorik – Aktoren.....	15
Sensorik – Relai.....	16
Datenbanken.....	19
Lernziele Datenbank.....	20
Datenbanken – Was ist eine Datenbank?.....	21
Datenbanken - Das ACID-Prinzip.....	22
Datenbanken – Welche Datenbanken gibt es?.....	23
Datenbanken – Einführung InfluxDB.....	25
Datenbanken – Warum Buckets wichtig sind.....	25
Datenbanken – Der API-Token.....	26
Datenbanken – Warum API-Token wichtig sind.....	26
Datenbanken – Skript Allgemein.....	27
Datenbanken – Python-Skript und InfluxDB.....	27
Dashboards.....	30
Lernziele Dashboard.....	31
Dashboards – Was ist ein Dashboard?.....	32
Dashboards – Der Nutzen eines Dashboards.....	32
Dashboards – Grafana.....	33
Dashboards – Warum Panels wichtig sind.....	33
Dashboards – Was ist ein Panel in Grafana?.....	34
Dashboards – Abfragen in Datenbanken.....	35
Dashboards – Abfragen in Grafana.....	36
Dashboards – Einheiten.....	38
Dashboards – Einsatz von Panels.....	41
Dashboards – Farben und Gestaltung.....	41
Dashboards – Dashboard und Zeit.....	41
Dashboards – Darstellungsarten in Grafana.....	43
Dashboards – Wann welche Darstellungsart wählen?.....	43
Dashboards – Darstellungsarten Wahl.....	45
Dashboards – Thresholds.....	45
Dashboards – Multipanels.....	46

Vorwort

Vorwort

Liebe Leserinnen und Leser,

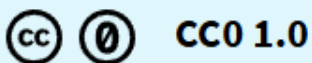
herzlich willkommen zu diesem Aufgabenheft, das im Rahmen eines Kursprojekts an der Universität Potsdam entstanden ist. Es führt Sie Schritt für Schritt durch einen praxisorientierten Workshop, der Sensorik im Kontext der fortschreitenden Digitalisierung erlebbar macht. Ob Sie als Lehrkraft spannende Experimente in Ihrem Unterricht integrieren oder als Schüler*in selbst aktiv werden möchten – dieses Heft bietet Ihnen alle nötigen Materialien und Anleitungen.

Zugleich ist das Heft so konzipiert, dass es auch unabhängig vom Workshop ein wertvoller Begleiter bleibt: Nutzen Sie die Aufgaben und Hintergrundinformationen jederzeit für eigene Projekte, Hausarbeiten oder als Einstieg in die Welt der digitalen Mess- und Regelungstechnik. Die Aufgaben sind klar strukturiert, motivierend formuliert und fördern kreatives sowie kritisches Denken.

Alle enthaltenen Arbeitsblätter, Beispielskripte und Präsentationen stehen Ihnen Open Source auf GitHub unter <https://github.com/DIYSensorikWorkshop/DIY-Smart-Sensorik-Workshop> zur Verfügung – vollständig unter einer OER-Lizenz. Sie können die Materialien beliebig anpassen, weitergeben und erweitern. Wir wünschen Ihnen viel Freude beim Entdecken, Ausprobieren und Forschen!

Ihr Projektteam „Urban Microclimate“ der Universität Potsdam

InfluxDB & Grafana AUFGABENHEFT Sensorik by Urban Microclimate is marked CC0 1.0 Universal. To view a copy of this mark, visit <https://creativecommons.org/publicdomain/zero/1.0/>

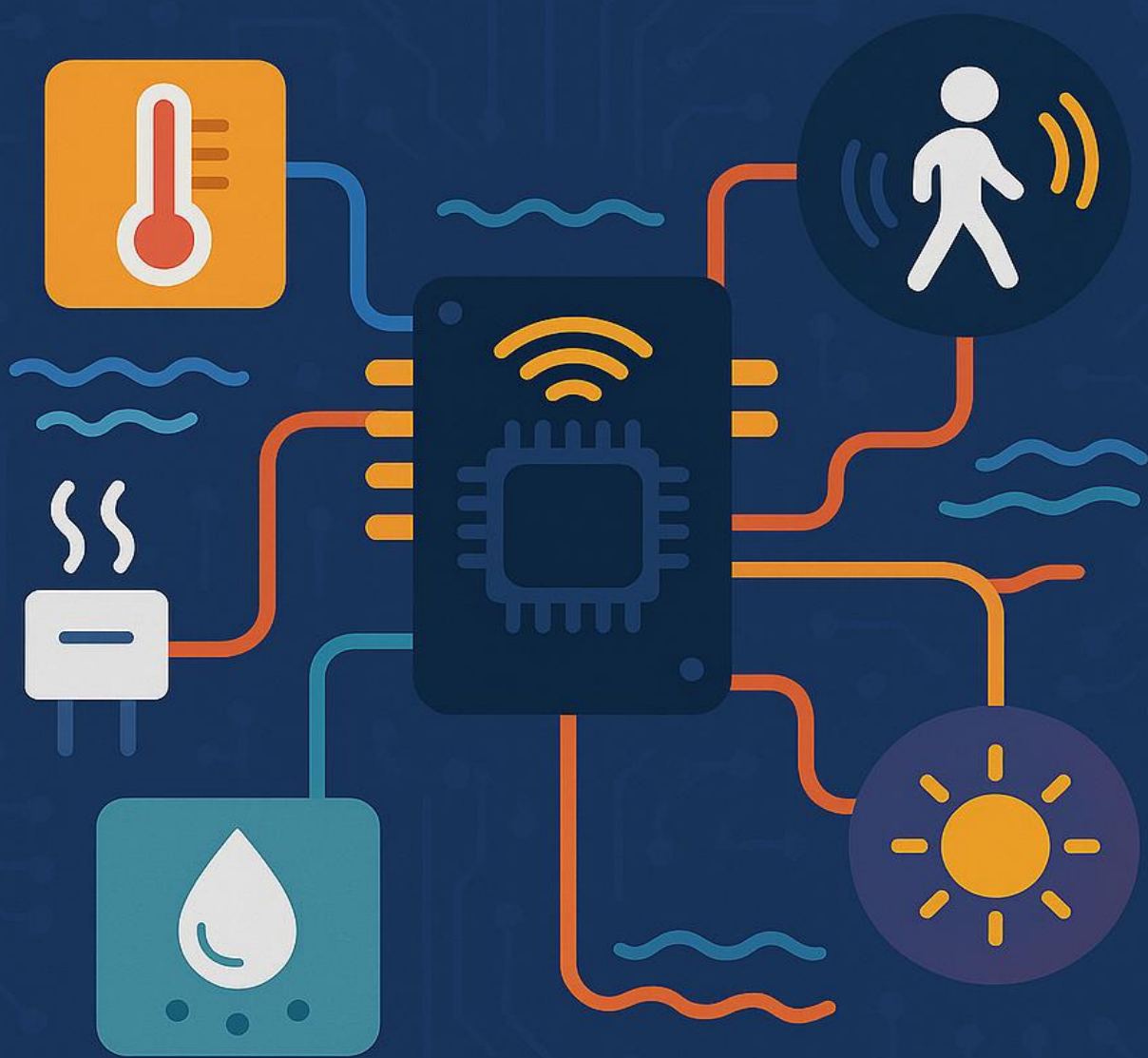


CC0 1.0 Universal

By marking the work with a CC0 public domain dedication, the creator is giving up their copyright and allowing reusers to distribute, remix, adapt, and build upon the material in any medium or format, even for commercial purposes.

© CC0: This work has been marked as dedicated to the public domain.

Sensorik



Lernziele Sensorik

Lernziele

- Verständnis grundlegender Sensorprinzipien (z. B. analog vs. digital) ☐
- Kenntnis der Anbindung von Sensoren an den Raspberry Pi (GPIO, I²C) ☐
- Installation und Nutzung relevanter Python-Bibliotheken (gpiozero, smbus2, Adafruit) ☐
- Durchführung von Messungen und Interpretation der Sensordaten ☐
- Fehlerdiagnose und -behebung bei Sensorproblemen ☐
- Praktische Anwendung eines CO₂-Sensors (SCD41) ☐

Die folgenden Aufgaben bauen alle aufeinander auf.

Bitte lasse bereits verbundene Bauteile wie den SCD41-Sensor dauerhaft angeschlossen, da sie in den nächsten Schritten weiterhin genutzt werden.

Alle Erweiterungen z. B. die LED-Anzeige oder das Relais werden zusätzlich ergänzt, nicht ersetzt.

So entsteht Schritt für Schritt ein vollständiges Mess und Steuersystem.

Bauteile

• Was brauchen wir?

Grundausrüstung:

Ein Raspberry Pi mit Netzteil, eine eingerichtete microSD-Karte mit Raspberry Pi OS sowie eine Maus, die dich per SSH mit dem Pi verbinden kannst.

Part 1 (Sensorik) verwenden wir den SCD41-Sensor, um CO₂-, Temperatur- und Luftfeuchtedaten zu sammeln. Dazu brauchst du außerdem Jumper-Kabel, um den Sensor über die I²C-Schnittstelle mit dem Pi zu verbinden.

In **Part 2 (LEDs)** kommen drei LEDs zum Einsatz, die mithilfe von Widerständen (zwischen 20 und 220 Ohm) an ein Steckbrett angeschlossen werden. Sie zeigen später den Temperaturbereich farblich an und dienen auch als CO₂-Warnsignal.

Auch hier benötigst du wieder **Jumper-Kabel**, um die Verbindung zum Raspberry Pi herzustellen.

In **Part 3 (Relais & Heizmatte)** erweitern wir das System um ein Relais-Modul, mit dem du eine Heizmatte steuern kannst.

Dafür brauchst du zusätzlich Schalta Draht oder Klingeldraht für die Stromversorgung sowie einige Jumper-Kabel zur Steuerung über den GPIO.

Alle Bauteile ergänzen sich Schritt für Schritt, bereits angeschlossene Komponenten werden verbunden und werden im nächsten Abschnitt weiterverwendet.

Grundlegend:

PC, Raspberry Pi mit Netzteil, Micro-SD

Part 1 Sensorik

SCD41

Jumper Kabel

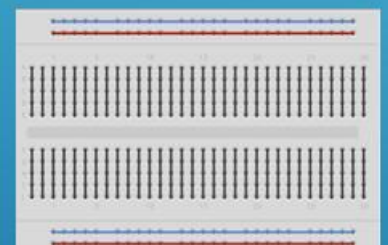
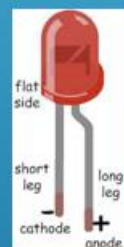


Part 2 LEDs

3 LEDs & Widerstände (20 -220ohm)

Jumperkabel

Steckbrett



Part 3 Relais & Heizmatte

Relais-Modul

Heizmatte

Jumper Kabel

Schalta Draht



Sensorik

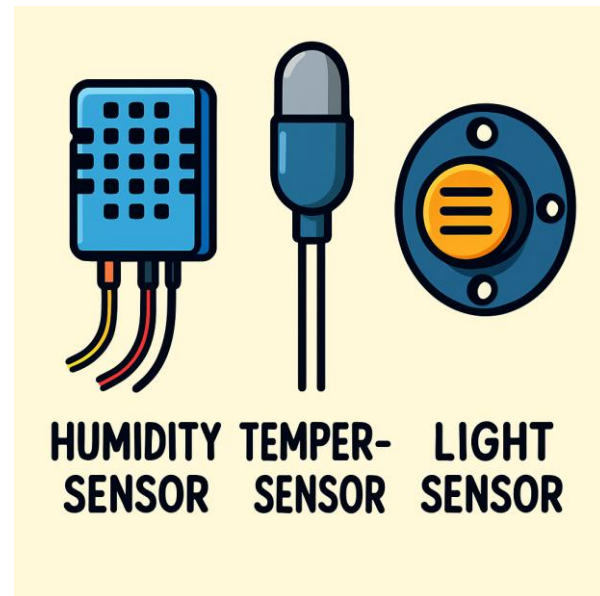
Was ist ein Sensor?

Ein Sensor ist ein kleines Gerät oder Bauteil, das Informationen aus der Umgebung wahrnimmt und in elektrische Signale umwandelt. Dabei „fühlt“ der Sensor eine physikalische Größe – zum Beispiel Temperatur, Licht, Druck oder Bewegung – und liefert dafür messbare Werte.

Beispiel: Ein Temperatursensor erfasst, wie warm oder kalt es ist, und gibt diesen Wert als Spannung oder digitalen Zahlencode aus. Ein Lichtsensor registriert, wie hell es um ihn herum ist, und kann so etwa das Einschalten einer Straßenlaterne steuern.

Moderne Sensoren arbeiten häufig digital: Sie messen zuerst analog und wandeln das Ergebnis dann mit einem kleinen Computerchip in einen binären Code um, den Mikrocontroller oder Computer weiterverarbeiten können. So ermöglichen Sensoren in Smartphones, Autos oder in der Gebäudetechnik, dass Geräte automatisch auf ihre Umwelt reagieren – sei es beim Regeln der Heizung, Erkennen von Hindernissen oder Auslösen eines Alarms.

Kurz gesagt: Sensoren sind unsere „Sinne“ für Maschinen und schaffen die Grundlage für viele praktische und spannende Anwendungen in der digitalen Welt.



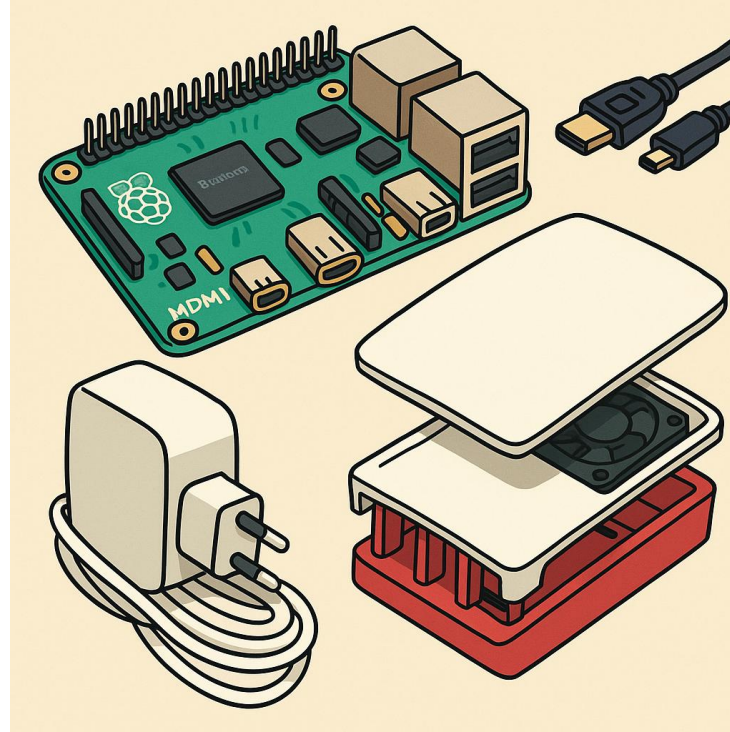
Sensorik

Was ist ein Raspberry Pi?

Ein Raspberry Pi ist ein kleiner Computer, der kaum größer als eine Postkarte ist. Trotzdem steckt auf seiner Platine alles drin, was man für einen PC braucht:

- **Rechner und Arbeitsspeicher:** Er verarbeitet Programme und speichert Daten.
- **Anschlüsse:** Du schließt Bildschirm (HDMI), Maus und Tastatur (USB) sowie Internet (LAN oder WLAN) an.
- **GPIO-Pins:** Damit kannst du direkt LEDs, Sensoren oder Motoren steuern.

Meist läuft auf dem Raspberry Pi ein kostenloses Betriebssystem namens Raspberry Pi OS (eine Linux-Version). Damit programmierst du in Sprachen wie Python oder Scratch und baust zum Beispiel deine eigene Wetterstation, einen Musik-Player oder ein kleines Robotermodell. Dank der vielen Tutorials und der großen Community ist der Raspberry Pi ideal für alle, die spielerisch in die Welt der Computer- und Elektronik-Projekte einsteigen wollen.



Sensoren & Raspberry Pi

Der Raspberry Pi wird erst richtig spannend, wenn er nicht nur Programme abspielt, sondern echte Daten aus der Umwelt sammelt und daraus Entscheidungen trifft. Dazu verbindest du deinen Sensor über die Stromversorgung (3,3 V und GND) und die Datenleitungen direkt mit den GPIO-Pins des Pi – bei einfachen digitalen Sensoren eine einzige Signalleitung, bei I²C-Sensoren die beiden Leitungen SDA und SCL. Nachdem du in den Einstellungen des Raspberry Pi die I²C-Schnittstelle aktiviert hast, installierst du in Python Bibliotheken wie `gpiozero` für einfache Sensoren oder `smbus2` für I²C-Kommunikation. In deinem Skript öffnest du dann diese Schnittstelle, liest regelmäßig Messwerte aus und verarbeitest sie weiter. Liegt zum Beispiel die Temperatur über

25 °C, kann dein Pi automatisch einen angeschlossenen Lüfter einschalten, oder registriert ein Lichtsensor Dunkelheit, so lässt er eine LED leuchten. Auf diese Weise entsteht aus der Kombination von Hardware und Software ein interaktives Mess- und Steuersystem, mit dem du Digitalisierung im Alltag selbst erleben kannst!

Vorbereitung und Installation

1. 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).

Sensorik

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`

Übung 1 - 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.)

Was ist SSH?

SSH steht für „Secure Shell“ und ist ein sicheres Protokoll, mit dem du dich über ein Netzwerk auf einem anderen Computer – zum Beispiel deinem Raspberry Pi – einloggen kannst, ohne direkt Maus oder Tastatur anzuschließen. Statt eines Bildschirms siehst du dann nur ein Terminalfenster auf deinem Rechner. Alles, was du eintippst, wird verschlüsselt übertragen, so dass niemand deine Befehle oder Passwörter mitlesen kann. Mit SSH kannst du Dateien kopieren, Programme starten oder Einstellungen ändern, als würdest du direkt am Pi sitzen. Für den ersten Zugriff musst du nur einmalig in den Einstellungen SSH aktivieren und dich mit dem Standard-Nutzer (häufig „pi“) und seinem Passwort („raspberrypi“) anmelden – am besten änderst du das Passwort sofort, um dein Pi sicher zu halten



Wissensfrage

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

Diskussionsaufgabe

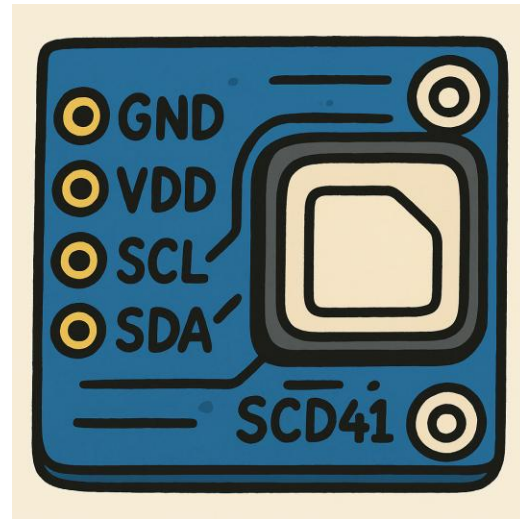
Diskutiere mit deinen Mitschüler*innen:

- 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?

Sensorik

SCD41 - Sensor

Der SCD41 ist ein winziger Sensor der Firma Sensirion, der gleichzeitig Kohlendioxid (CO₂), Temperatur und Luftfeuchtigkeit misst. Er nutzt dafür ein spezielles Messverfahren nach dem NDIR-Prinzip: Infrarot-Licht wird in eine kleine Kammer geschickt und von CO₂-Molekülen absorbiert. Die absorbierte Energie führt zu winzigen Druckänderungen, die ein Mikrofon im Sensor erkennt – daher heißt es Photoakustik. Auf dem gleichen Chip sitzen integrierte Temperatur- und Feuchtesensoren, mit denen die CO₂-Werte automatisch korrigiert werden, damit sie auch bei wechselnden Umgebungsbedingungen genau bleiben [Sensirion](#). Per I²C-Bus (Datenleitungen SDA und SCL) lässt sich der SCD41 einfach an Mikrocontroller wie den Raspberry Pi anschließen; zusätzlich benötigt er eine Versorgungsspannung von 3,3 V und Masse (GND). Mit hoher Messgenauigkeit im Bereich von 400 bis 5000 ppm ist er ideal, um in Klassenzimmern, Wohnungen oder Büros die Luftqualität im Auge zu behalten und bei zu hohem CO₂-Wert automatisch für frische Luft zu sorgen.



SCD41 - Verwenden

1. Verkabelung (I²C)

- **VIN** des Sensors → **3.3 V** (Pin 1)
- **GND** des Sensors → **GND** (Pin 6)
- **SCL** des Sensors → **GPIO 3** (Pin 5)
- **SDA** des Sensors → **GPIO 2** (Pin 3)

Achte unbedingt darauf, dass SDA und SCL nicht vertauscht sind und die Steckverbindungen festsitzen.

Sensorik

2. Sensor auslesen

1. Öffne den folgenden Link und kopiere den vollständigen Beispielcode:

https://github.com/DIYSensorikWorkshop/DIY-Smart-Sensorik-Workshop/blob/main/Python%20Code%20Sensorik/Part1_scd41.py

2. Verbinde dich per SSH mit deinem Raspberry Pi.

3. Navigiere mit `cd` zum Desktop des Pi

4. Lege mit Befehl `mkdir „Ordnername“` einen neuen Ordner für das Projekt an

5. Öffne mit dem Befehl `nano „Dateiname“` den Nano Editor um eine leere Python Datei zu erstellen

6. Füge den kopierten Code mit STRG-V in die Datei ein

7. Speichere und Schließe den Editor mit STRG S und STRG X

8. Anschließend starte mit dem Befehl `Python „Dateiname.py“` das Skript

3. Debugging-Tipps

- Verkabelung erneut prüfen (SDA/SCL, stabile Verbindungen)
- I²C im Raspberry-Pi-Konfigurationsmenü aktivieren
- Prüfe mit dem Befehl `i2cdetect -y 1` ob der Sensor erkannt wird
- Sicherstellen, dass python3 und die Adafruit-Bibliotheken installiert sind

Sensorik

Kreativ Aufgabe

Erkläre in **drei Sätzen**, warum ein CO₂-Sensor in einem Klassenzimmer oder Jugendzimmer sinnvoll sein kann. Überlege, welche Folgen es hat, wenn der CO₂-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?

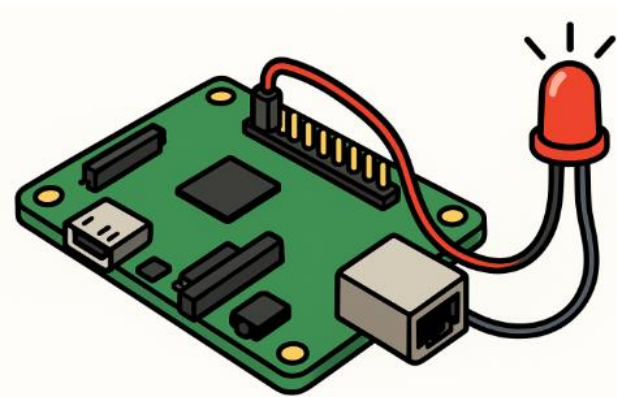
Diskussionsaufgabe

Diskutiere mit deinen Mitschüler*innen:

- Ist es sinnvoll, die Luftqualität **nur** über CO₂-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?

Aktoren

Aktoren sind sozusagen die „Muskeln“ eines elektronischen Systems: Sie wandeln elektrische Signale in eine physikalische Aktion um. Während Sensoren Informationen aus der Umwelt sammeln (zum Beispiel Temperatur oder Lichtstärke), führen Aktoren daraufhin eine Reaktion aus. Das kann ganz simpel sein, etwa eine LED, die aufleuchtet, oder ein Summer, der piept. Komplexere Aktoren sind Motoren, die drehende Bewegung liefern, oder Relais, mit denen du stärkere Verbraucher wie Lampen oder Pumpen schalten kannst. In deinem Raspberry-Pi-Projekt liest du also zuerst mit einem Sensor einen Wert aus und schickst dann per Python-Code ein Signal an einen Aktor, der daraufhin beispielsweise einen Lüfter einschaltet, ein Ventil öffnet oder eine Anzeige aktualisiert. So entsteht aus Messen und Steuern dein interaktives System!

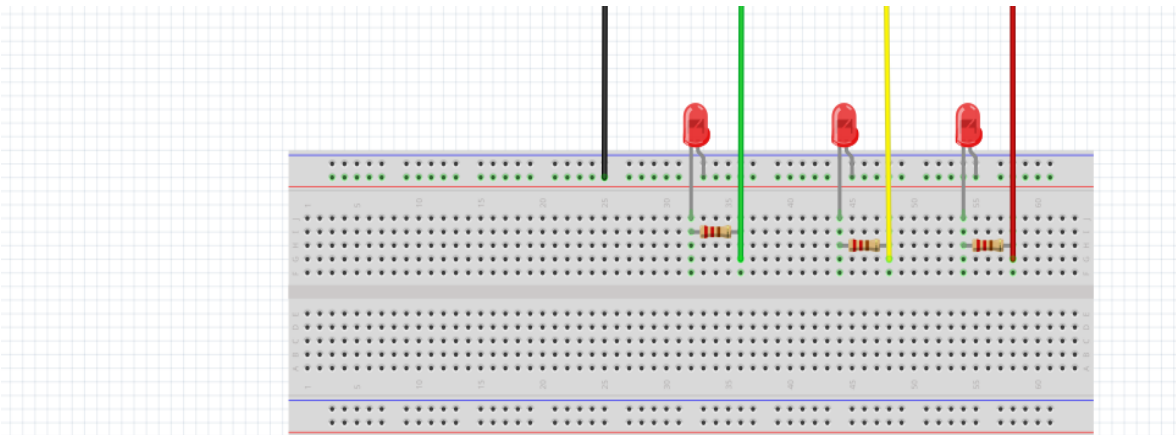


LED Steuerung

1. LED Anschließen

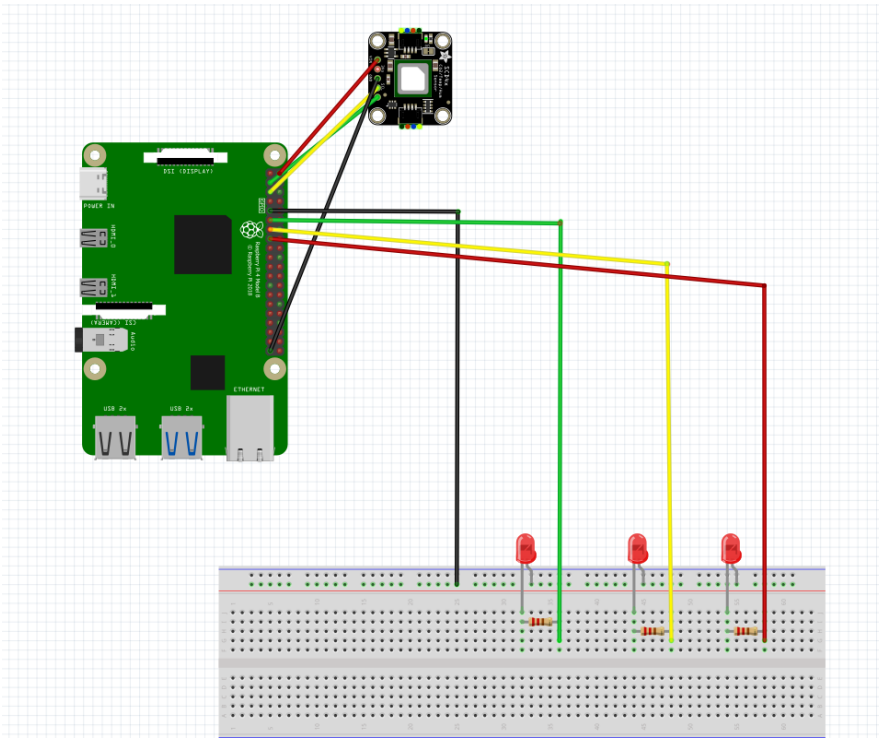
1. Die LEDs werden wie folgt über **Widerstände (20–220 Ohm)** mit GND verbunden. Sie zeigen später über Farben an, ob es zu kalt, angenehm oder zu warm ist.

LED Farbe	GPIO-Pin	Pin am Raspberry Pi	Schaltungshinweis
Rot	GPIO 22	Pin 15	Über 20Ω - 220Ω Widerstand mit GND verbinden
Gelb	GPIO 27	Pin 13	Über 20Ω - 220Ω Widerstand mit GND verbinden
Grün	GPIO 17	Pin 11	Über 20Ω - 220Ω Widerstand mit GND verbinden



Sensorik

So sollte der gesamte Aufbau aussehen:



1. Öffne den folgenden Link und kopiere den vollständigen Beispielcode:

[DIY-Smart-Sensorik-Workshop/Python Code Sensorik/Part2 led control.py at main · DIYSensorikWorkshop/DIY-Smart-Sensorik-Workshop](https://github.com/DIY-Sensorik-Workshop/Python_Code_Sensorik/Part2_led_control.py)

2. Verbinde dich per SSH mit deinem Raspberry Pi.

3. Navigiere zum vorher angelegten Ordner

4. Öffne mit dem Befehl nano „Dateiname“ den Nano Editor um eine leere Python Datei zu erstellen

6. Füge den kopierten Code mit STRG-V in die Datei ein

7. Speichere und Schließe den Editor mit STRG S und STRG X

8. Anschließend starte mit dem Befehl Python „Dateiname.py“ das Skript

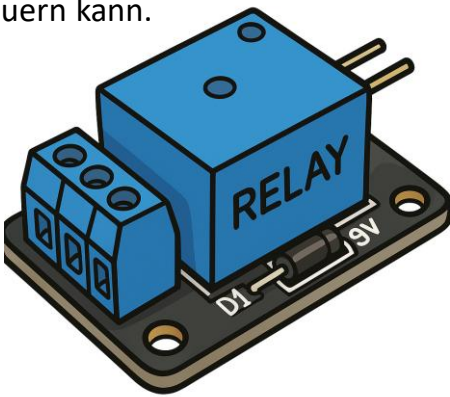
Relais

Ein Relais ist ein elektrischer „Fernschalter“, der mit einem kleinen Steuerstromkreis (Spule) einen zweiten, meist leistungsstärkeren Stromkreis schaltet. Im Inneren befindet sich eine kleine Spule, die bei Anlegen der Betriebsspannung ein Magnetfeld erzeugt. Dieses zieht einen beweglichen Arm („Anker“) an, der daraufhin Kontakte schließt oder öffnet. Auf diese Weise kann dein Raspberry Pi (über einen GPIO-Pin und einen Transistor oder Treiberbaustein) beispielsweise mit 3,3 V eine Lampe oder einen Motor an-/ausschalten, die mit 12 V oder 230 V betrieben werden.

Relais gibt es in verschiedenen Ausführungen:

- **Wechsler (SPDT):** Ein gemeinsamer Eingang (COM) kann auf einen von zwei Ausgängen (NO oder NC) geschaltet werden.
- **Schließer (SPST):** Schaltet bei Erregung einen Kontakt von offen auf geschlossen.
- **Öffner (SPST):** Hat im Ruhezustand geschlossene Kontakte, die bei Erregung öffnen.

Wichtig ist der Einbau einer **Freilaufdiode** parallel zur Spule, um Spannungsspitzen beim Abschalten abzufangen und die Steuerelektronik zu schützen. Relais ermöglichen so, dass dein Pi – ganz ohne direkte Berührung von Hochspannung – starke Ströme und Spannungen sicher schaltet und so vielfältige Aktoren ansteuern kann.



Relais-Steuerung

1. Relais und Heizmatte Verkabeln

- IN des Relai 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 (230V) unbedingt Sicherheitsregeln beachten und ggf. Fachleute hinzuziehen!

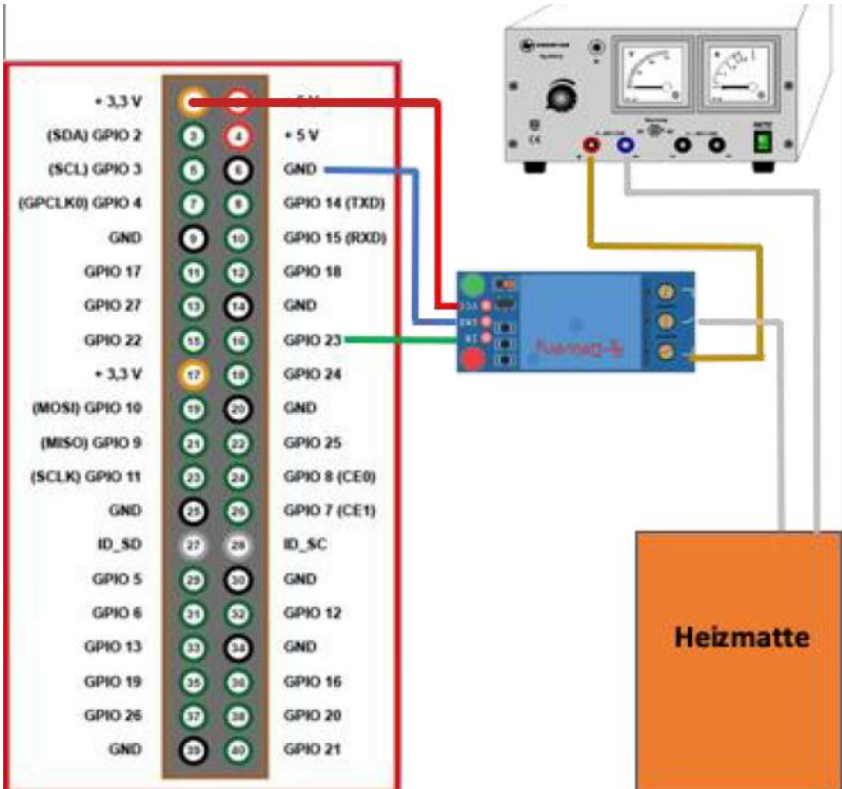
Farbe	Relais Pin	Raspberry Pi Pin
Rot	VCC	Pin 1 (3.3V)
Blau	GND	Pin 6 (GND)
Grün	IN	Pin 16 (GPIO 23)

Sensorik

Verdrahtung Heizmatte

Farbe	Von	Nach
 Braun/Gelb	Netzteil (+)	Relais COM
 Grau/Weiß	Relais NO	Heizmatte (+)
 Grau/Weiß	Heizmatte (-)	Netzteil (-)

Verbindungsdiagramm



2. Öffne den folgenden Link und kopiere den vollständigen Beispielcode:

https://github.com/DIYSensorikWorkshop/DIY-Smart-Sensorik-Workshop/blob/main/Python%20Code%20Sensorik/Part3_heat_control.py

2. Verbinde dich per SSH mit deinem Raspberry Pi.

3. Navigiere zum vorher angelegten Ordner

4. Öffne mit dem Befehl nano „Dateiname“ den Nano Editor um eine leere Python Datei zu erstellen

6. Füge den kopierten Code mit STRG-V in die Datei ein

7. Speichere und Schließe den Editor mit STRG S und STRG X

8. Anschließend starte mit dem Befehl Python „Dateiname.py“ das Skript

Sensorik

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 Anschlusseite einer LED, die positiv gepolt werden muss.“
 - C. „Begriff für eine Schaltung, bei der ein niedriger Pegel (0V) 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

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

Diskussionsaufgabe

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?

Datenbanken



Lernziele Datenbank

Lernziele

- Verständnis von Datenbankgrundlagen und DBMS-Funktionen
- Kenntnis der ACID-Eigenschaften und ihrer Bedeutung
- Unterschiede zwischen relationalen und zeitbasierten Datenbanken
- Einrichtung eines Buckets und Umgang mit Retention Policies in InfluxDB
- Erstellung und Verwaltung von API-Tokens für sicheren Datenzugriff
- Verbindung von Python-Skripten zu InfluxDB und Durchführung einfacher CRUD-Operationen
- Verfassen von Flux-Queries zur Abfrage von Zeitreihendaten

☐☐☐☐☐☐☐

Datenbanken

Was ist eine Datenbank?

Datenbanken sind spezialisierte Softwaresysteme, die dazu dienen, große Mengen an Informationen strukturiert, sicher und effizient zu verwalten. Im Kern speichern sie Daten in Tabellen, Dokumenten oder anderen Datenmodellen, sodass sie später gezielt abgefragt, verändert und analysiert werden können. Anders als einfache Dateien ermöglichen Datenbankmanagementsysteme (DBMS) den gleichzeitigen Zugriff vieler Nutzer, sorgen für Datenkonsistenz und schützen vor unberechtigtem Zugriff.

Definition Datenbank (DB):

Eine strukturierte Sammlung von Daten, die so organisiert ist, dass sie einfach gespeichert, abgefragt und verwaltet werden kann.

Definition Datenbankmanagementsystem (DBMS):

Eine Software, die die Erstellung, Pflege und Nutzung von Datenbanken übernimmt, mehrere Nutzerzugriffe koordiniert und Datenkonsistenz sowie -sicherheit gewährleistet.

Datenbanken

Das ACID-Prinzip

Traditionell basieren relationale Datenbanken auf dem relationalen Modell von Edgar F. Codd: Daten werden in Tabellen organisiert, die über gemeinsame Schlüssel miteinander verknüpft sind. Mit Hilfe der strukturierten Abfragesprache SQL (Structured Query Language) lassen sich komplexe Abfragen formulieren und Transaktionen durchführen, die nach dem ACID-Prinzip (Atomicity, Consistency, Isolation, Durability) für Zuverlässigkeit und Integrität sorgen. Typische Anwendungsfälle reichen von Geschäftsanwendungen über Finanzsysteme bis hin zu Webshops.

Definition

ACID:

Ein Set von vier Grundeigenschaften für Datenbanktransaktionen:

1. **Atomicity (Atomarität):** Eine Transaktion wird ganz ausgeführt oder gar nicht.
2. **Consistency (Konsistenz):** Die Datenbank wechselt nur von einem gültigen Zustand in einen anderen.
3. **Isolation (Isolation):** Gleichzeitige Transaktionen beeinflussen sich nicht gegenseitig.
4. **Durability (Dauerhaftigkeit):** Nach Abschluss einer Transaktion bleiben die Änderungen auch bei Stromausfall erhalten

Eine Transaktion ist in einem Datenbanksystem eine zusammengehörige Folge von Lese- und Schreiboperationen, die als eine einzige, unteilbare Einheit ausgeführt wird.

Beispiel mit allen vier Eigenschaften

Stell dir eine Bank-Überweisung vor:

1.Atomicity: Betrag wird komplett von A abgebucht und komplett auf B gebucht – oder gar nicht.

2.Consistency: Nach der Buchung gelten alle Regeln (z. B. Kontostand ≥ 0 , Summe aller Kontostände bleibt gleich).

3.Isolation: Wenn zwei Überweisungen gleichzeitig laufen, sieht jede nur einen abgeschlossenen Zustand – keine halben Abbuchungen oder Buchungsdoppelungen.

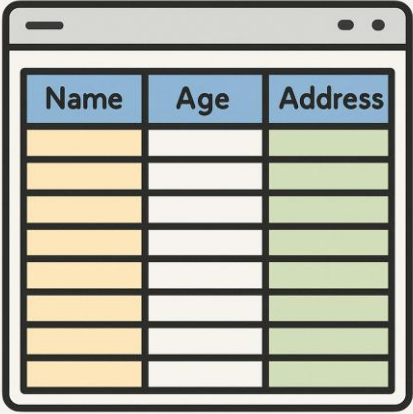
4.Durability: Sobald die Buchung bestätigt ist, ist sie dauerhaft auf Festplatte geschrieben und überlebt sogar Systemabstürze.

Datenbanken

Welche Datenbanken gibt es?

Relationale Datenbanken

Stell dir einen großen Tisch mit vielen Zeilen und Spalten vor – wie in Excel. Jede Zeile ist ein Datensatz, jede Spalte ein Feld (z. B. Name, Alter, Anschrift). Das Schema, also die genaue Reihenfolge und Art der Spalten, ist festgelegt. Mit der Sprache SQL kannst du Fragen an die Datenbank stellen („Zeige mir alle Schüler, die älter als 16 sind“) oder Tabellen verbinden (Joins). Relationale Datenbanken sind sehr zuverlässig, weil sie sicherstellen, dass alle Änderungen an den Daten vollständig und korrekt gespeichert werden.

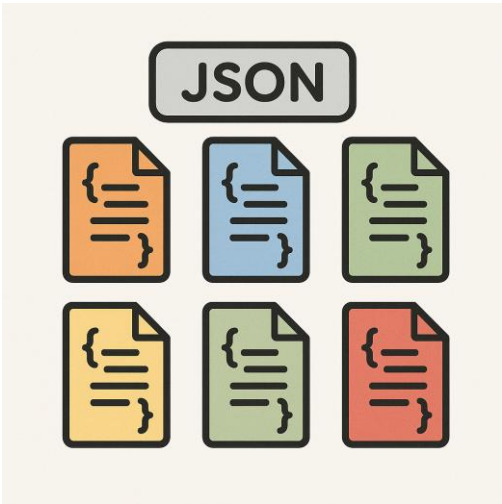


Das Diagramm zeigt eine Tabelle in einem Browserfenster. Die Tabelle hat drei Spalten: 'Name', 'Age' und 'Address'. Die Spaltenüberschriften sind in blauen Zellen, die Datenzeilen in gelben und grünen Zellen. Es gibt 8 Zeilen in der Tabelle.

Name	Age	Address

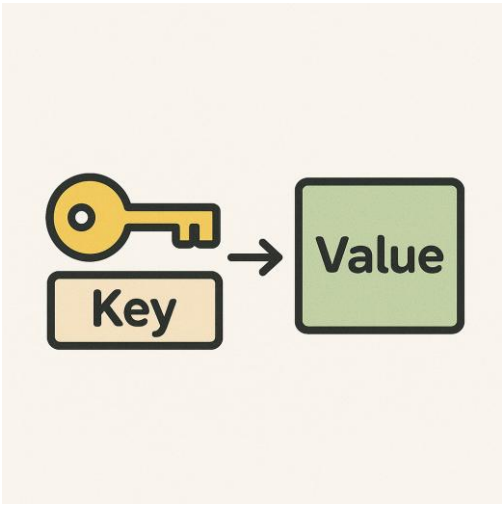
Dokumentenorientierte Datenbanken

Hier speicherst du Daten als einzelne „Dokumente“, meistens im JSON-Format. Ein Dokument kann ganz unterschiedlich aussehen – mal hat es fünf, mal zehn oder zwanzig Felder. Das ist praktisch, wenn du Daten hast, die nicht immer gleich aufgebaut sind, zum Beispiel Nutzerprofile, Blog-Beiträge oder Logdateien. Du suchst und liest ganze Dokumente, anstatt einzelne Tabellenfelder.



Schlüssel-Wert-Datenbanken

Das ist das einfachste Modell: Jeder Dateneintrag hat einen eindeutigen Schlüssel (Key) und einen dazugehörigen Wert (Value). Wenn du den Schlüssel kennst, bekommst du den Wert blitzschnell zurück. Solche Datenbanken werden oft als Zwischenspeicher (Cache) genutzt, wenn etwas besonders schnell gehen muss, etwa Webseiten-Daten im Arbeitsspeicher.

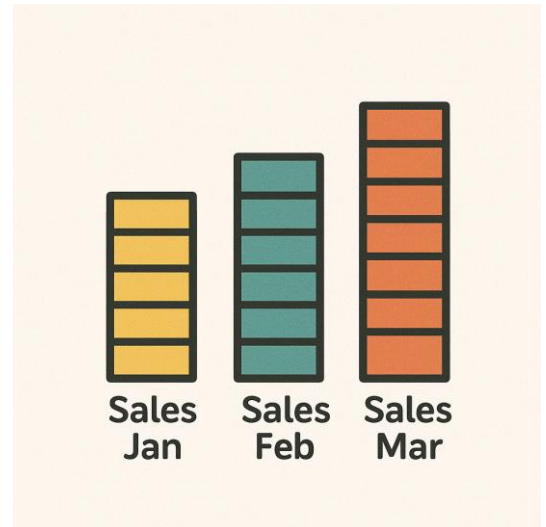


Datenbanken

Welche Datenbanken gibt es?

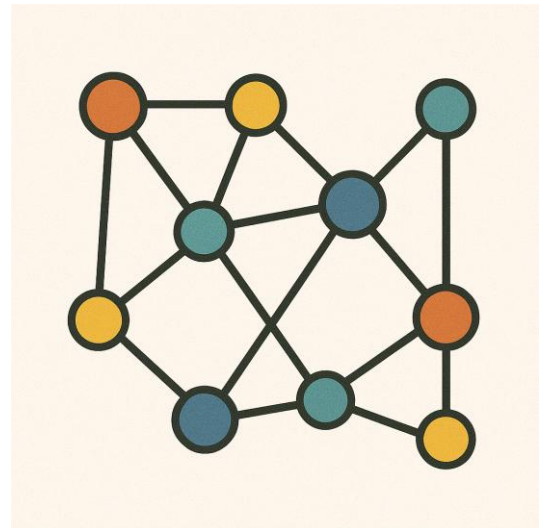
Spaltenorientierte Datenbanken

Diese sind anders als Excel-Tabellen aufgebaut: Daten werden nicht nach Zeilen, sondern nach Spalten gespeichert. Das ist besonders gut, wenn man oft große Datenmengen nach einzelnen Spalten sortieren oder zusammenfassen möchte, zum Beispiel alle Umsätze eines Jahres. Sie skalieren gut, wenn mehrere Rechner zusammenarbeiten müssen.



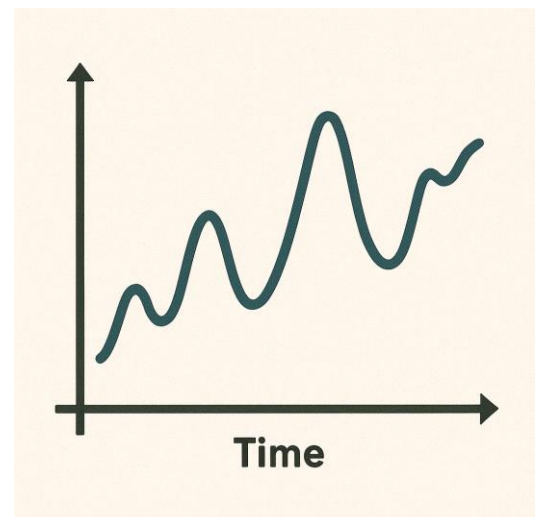
Graphdatenbanken

Stell dir ein Netz aus Punkten und Linien vor. Die Punkte (Knoten) sind Personen, Orte oder Dinge, die Linien (Kanten) beschreiben die Beziehungen zwischen ihnen (zum Beispiel „ist befreundet mit“ oder „arbeitet in“). Graphdatenbanken eignen sich hervorragend, um solche Verknüpfungen schnell abzufragen, etwa bei Freundschaftsnetzwerken oder Empfehlungssystemen („Kunden, die A gekauft haben, kauften oft auch B“).eiten müssen.



Zeitreihendatenbanken

Hier werden Messwerte mit Zeitstempeln aufgezeichnet, zum Beispiel Temperaturdaten oder Server-Metriken. Man speichert die Werte fortlaufend, kann sie nachträglich zusammenfassen (z. B. Durchschnitt pro Stunde) und alte Daten automatisch löschen, damit die Datenbank nicht unendlich groß wird. InfluxDB ist so eine Datenbank.



Datenbanken

Einführung InfluxDB

In InfluxDB werden Zeitreihendaten, also Messwerte, die über die Zeit hinweg erfasst werden, in sogenannten *Buckets* gespeichert. Man kann sich einen Bucket wie ein Einmachglas vorstellen: Hier legst du alle Messdaten ab, die thematisch zusammengehören. Jeder Bucket erhält einen Namen und eine Aufbewahrungsfrist (Retention Policy). Nach Ablauf dieser Frist entfernt InfluxDB automatisch die ältesten Einträge, sodass deine Datenbank nicht unkontrolliert wächst.

Warum Buckets wichtig sind

Ein Bucket ist mehr als nur ein Ablageort, er definiert auch, wie lange Daten aufbewahrt werden. Legst du zum Beispiel einen Bucket mit einer Retention Policy von 30 Tagen an, werden alle Daten, die älter als 30 Tage sind, automatisch gelöscht. Das ist besonders nützlich, wenn du Messreihen hast, die nach einer gewissen Zeit nicht mehr relevant sind.

Übung 1 – InfluxDB Bucket anlegen

Starte zunächst den InfluxDB-Client und öffne in deinem Webbrowser die Adresse <http://localhost:8086>. Melde dich im Webinterface mit deinen Zugangsdaten an, erstelle einen neuen Bucket und notiere dir den dabei vergebenen Namen deines Buckets.

Name des Buckets: _____



Datenbanken

Der API-Token

Ein **API-Token** ist ein geheimer Schlüssel, mit dem Anwendungen und Skripte sich gegenüber InfluxDB authentifizieren können, ohne Benutzername und Passwort zu verwenden. Jeder Token kann auf bestimmte Ressourcen und Rechte beschränkt werden (z. B. nur Lese- oder Schreibzugriff auf einen einzelnen Bucket). So stellst du sicher, dass jede Anwendung nur genau die Berechtigungen erhält, die sie benötigt.

Definition API-Token:

API steht für **Application Programming Interface**, auf Deutsch **Programmierschnittstelle**. Ein API-Token ist also ein geheimer Schlüssel, mit dem Programme bzw. Skripte Zugriff auf genau diese Schnittstelle deiner Anwendung (hier InfluxDB) erhalten.

Warum API-Token wichtig sind

Warum API-Tokens wichtig sind

1. **Sicherheit:** Ein kompromittierter Token kann jederzeit widerrufen und ersetzt werden, ohne Zugangsdaten von Nutzern ändern zu müssen.
2. **Least-Privilege-Principle:** Du vergibst genau die Rechte, die eine Anwendung braucht – nicht mehr und nicht weniger.
3. **Nachvollziehbarkeit:** InfluxDB protokolliert, welcher Token wann auf welche Daten zugegriffen hat.

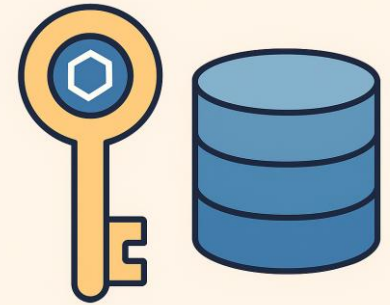
Datenbanken

Übung 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.

Beschreibung: _____

API-Token: _____



Wichtig

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

Skript Allgemein

Ein Skript ist eine Textdatei, die eine Abfolge von Anweisungen enthält, welche von einem Interpreter (z. B. einer Shell, Python oder JavaScript-Laufzeit) zeilenweise ausgeführt werden. Mit Skripten lassen sich wiederkehrende Aufgaben und Abläufe automatisieren, ohne dass man jede einzelne Anweisung manuell eingeben muss. Sie sind besonders nützlich für Systemadministration, Datenverarbeitung und einfache Programmieraufgaben.

Python-Skript und InfluxDB

Mit deinem Python-Skript stellst du eine direkte Verbindung zwischen deinen Programmen und dem InfluxDB-Server her. In der Regel findest du in der Datei „influxdb.py“ vier Variablen, die du anpassen musst: **url**, **token**, **org** und **bucket**. Die **url** ist die Adresse deines InfluxDB-Servers, üblicherweise `http://localhost:8086`, wenn du lokal arbeitest. Über diese Adresse sendet dein Skript alle Abfragen und Schreibbefehle an den Server.

Datenbanken

Der **token** ist dein geheimer Zugangsschlüssel, den hast du bereits in Aufgabe 2 generiert. Er erlaubt deinem Skript, sich ohne Benutzername und Passwort zu authentifizieren und genau die Rechte zu nutzen, die du zuvor festgelegt hast. Die Variable **org** gibt an, zu welcher Organisation in InfluxDB die Anfragen gehören. InfluxDB trennt Daten und Berechtigungen nach Organisationen, damit mehrere Teams getrennt arbeiten können. Schließlich steht **bucket** für den Speicherort deiner Messdaten, den „Eimer“, in das dein Skript die Werte füllt oder aus dem es Abfragen holt.

Fehlerhafte oder leere Einträge in einer dieser Variablen führen dazu, dass dein Skript keine Verbindung zum Server herstellen kann, Fehlermeldungen wirft oder Daten ins Leere schreibt. Deshalb ist es wichtig, die Werte exakt so einzutragen, wie sie in der Web-Oberfläche angezeigt werden.

Übung 3 – Skript Verbindung

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

url = “ _____ ”

token = “ _____ ”

org = “ _____ ”

bucket = “ _____ ”



Datenbanken

Nach erfolgreichem einfügen, starte das Script.

Direkt danach wechselst du zurück in die InfluxDB-Weboberfläche und öffnest unter **Buckets** deinen Bucket. Klicke auf das Uhr-Icon mit „Past ...“, wähle zum Beispiel **Past 5m** und schaue, ob die erwarteten Messwerte als Graph dargestellt werden. Erscheint der Graph, war deine Verbindung erfolgreich und dein Skript schreibt oder liest Daten korrekt aus deinem Bucket. Sollte nichts passieren, überprüfe noch einmal Tippfehler in URL, Token, org und bucket oder probiere ein anderes Zeitfenster aus.

DASHBOARDS



Lernziele Dashboard

Lernziele

- Verständnis der Funktion und des Nutzens von Dashboards
- Grundlagen der Arbeit mit Grafana (Installation, Dashboards, Panels)
- Konfiguration von Datenquellen und Erstellung von Abfragen in Grafana
- Auswahl geeigneter Visualisierungen (Line Chart, Bar Chart, Pie Chart, Stat Panel)
- Gestaltung von Dashboards mittels Layouts, Farben und Thresholds
- Erstellen von Multipanels für Übersicht und Vergleich mehrerer Sensorwerte

☐☐☐☐☐☐

Dashboards

Was ist ein Dashboard?

Ein Dashboard ist eine übersichtliche Seite oder Ansicht, die euch alle wichtigen Informationen auf einen Blick zeigt. Stellt euch vor, ihr hättet ein Armaturenbrett an eurem Fahrradlenker, auf dem eure aktuelle Geschwindigkeit, die zurückgelegte Strecke und der Akkustand eures E-Bikes angezeigt werden. Genau so funktioniert ein Dashboard am Computer oder im Internet: Es bündelt nur das Wesentliche, damit ihr nicht lange suchen müsst.

Ein Dashboard hilft euch vor allem dabei, schnell zu verstehen, worauf ihr achten solltet. Wenn ihr zum Beispiel auf einer Lernplattform angemeldet seid, seht ihr dort auf eurem Dashboard, wie viele Aufgaben ihr bereits gelöst habt, welche Tests noch anstehen und wie eure Durchschnittsnote aussieht. Ihr erkennt sofort, ob ihr gut dabei seid oder ob ihr noch etwas nacharbeiten solltet.

Oft bestehen Dashboards aus Grafiken und Farben, weil Bilder leichter zu verstehen sind als lange Zahlenreihen. Ein Balkendiagramm kann zum Beispiel anzeigen, wie viele Punkte ihr in den letzten Wochen gesammelt habt, und ein Kreisdiagramm zeigt euch, wie sich eure Lernzeit auf verschiedene Fächer verteilt. Farbliche Markierungen – Grün für „Alles gut“, Gelb für „Achtung“ und Rot für „Handlungsbedarf“ – machen auf einen Blick deutlich, wo ihr euch verbessern könnt.

Der Nutzen eines Dashboards

Ein Dashboard hilft dabei, Informationen schnell und übersichtlich zu erfassen, damit ihr:

- **Sofort seht**, welche Aufgaben noch offen sind oder wo Handlungsbedarf besteht,
- **Zeit spart**, weil ihr nicht lange suchen müsst,
- **bessere Entscheidungen trefft**, da wichtige Zahlen und Trends direkt als Grafiken oder Farbmarkierungen sichtbar sind.

So behaltet ihr eure Ziele im Blick und könnt euren Alltag (oder Projekte) effizienter gestalten.

Dashboards

Grafana

Grafana ist eine webbasierte Visualisierungsplattform, mit der du Zeitreihendaten aus verschiedenen Datenquellen übersichtlich aufbereiten kannst. Nach der Installation läuft Grafana meist als Dienst im Hintergrund und ist über den Browser erreichbar. In Grafana baust du **Dashboards** auf, die aus mehreren **Panels** bestehen. Jedes Panel zeigt eine bestimmte Darstellung, zum Beispiel Linien-, Balken- oder Kreisdiagramme, die direkt auf die zugrunde liegenden Datenquellen (z. B. InfluxDB) zugreifen. So lassen sich Messwerte in Echtzeit beobachten, Trends erkennen und Alarmer konfigurieren, wenn bestimmte Schwellwerte überschritten werden.

Übung 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: _____



Warum Panels wichtig sind

Ein Dashboard ohne Panels wäre nur eine leere Seite. Durch Panels lassen sich unterschiedliche Aspekte eurer Daten nebeneinander darstellen. So habt ihr alle relevanten Informationen, Trends, Vergleiche und Kennzahlen zentral an einem Ort und könnt schnell Entscheidungen treffen oder Probleme erkennen.

Dashboards

Was ist ein Panel in Grafana?

Ein Panel in Grafana ist eine einzelne Anzeige-Einheit innerhalb eines Dashboards, in der eine bestimmte Datenvisualisierung dargestellt wird. Man kann sich ein Dashboard wie ein leeres Plakat vorstellen, auf dem ihr verschiedene Rahmen (Panels) aufklebt, in jedem Rahmen steckt dann eine Grafik oder Tabelle, die genau eine Fragestellung beantwortet.

Im Detail läuft das so ab:

1. Datenquelle auswählen

Jedes Panel bezieht seine Daten aus einer oder mehreren konfigurierten Quellen (z. B. InfluxDB). Ihr wählt unter „Data source“ aus, woher die Werte kommen sollen.

2. Abfrage (Query) festlegen

Im nächsten Schritt definiert ihr, welche Messreihen oder Kennzahlen abgefragt werden. Grafana zeigt dabei eine Vorschau an, damit ihr sofort seht, ob eure Query die gewünschten Daten liefert.

3. Visualisierungstyp bestimmen

Grafana bietet verschiedene Darstellungsformen an:

- Liniendiagramm für Zeitreihen-Trends
- Balkendiagramm für Vergleiche
- Kreisdiagramm für Anteile
- Tabelle für Rohwerte
- Text-, Gauge- oder Heatmap-Panels für spezielle Anforderungen

4. Panel-Titel und Beschreibung vergeben

Damit ihr und andere Nutzer später auf einen Blick erkennt, was das Panel zeigt, gebt ihr einen aussagekräftigen Titel und eine kurze Beschreibung ein.

5. Layout und Optionen anpassen

In den Panel-Einstellungen könnt ihr Farben, Achsenbeschriftungen, Legenden und Schwellenwerte konfigurieren. So sorgt ihr dafür, dass eure Visualisierung klar und verständlich bleibt.

6. Speichern

Nach dem Anpassen speichert ihr das Panel und seht es sofort im Dashboard-Raster. Ihr könnt Panels jederzeit verschieben, vergrößern oder kopieren.

Dashboards

Übung 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: _____



Abfragen in Datenbanken

Abfragen (engl. *Queries*) sind Anweisungen, mit denen ihr gezielt Daten aus einer Datenbank herausholt. Stellt euch vor, eine Datenbank ist wie ein umfangreiches Buch voller Tabellen und Informationen. Eine Abfrage ist dann wie eine Suchanfrage im Inhaltsverzeichnis: Ihr sagt der Datenbank genau, welche Daten ihr braucht, zum Beispiel alle Einträge aus der vergangenen Stunde oder nur die Werte eines bestimmten Sensors.

- **Zweck:** Ihr wollt nicht das ganze „Buch“ durchlesen, sondern nur die für euch relevanten Seiten finden.
- **Aufbau:** Eine Abfrage besteht meist aus mehreren Teilen, die zusammen einschränken, was zurückgegeben wird:

1. Datenquelle auswählen (z. B. welche Tabelle oder welcher Bucket)

2. Zeitraum festlegen (z. B. letzte Stunde, letzter Tag)

3. Filter setzen (z. B. nur ein bestimmtes Feld wie „temperature“ oder nur ein bestimmter Sensor)

- **Ergebnis:** Die Datenbank liefert genau die Datensätze, die diesen Kriterien entsprechen. So spart ihr Zeit und arbeitet effizient.

Dashboards

Abfragen in Grafana

In Grafana steckt jede Visualisierung (jedes *Panel*) hinter einer solchen Abfrage. Sobald ihr in einem Panel eure Datenquelle (z. B. InfluxDB) ausgewählt habt, schreibt ihr eine Query, die Grafana ausführt und deren Ergebnis im Chart darstellt:

- **Query-Editor öffnen**

Nach Klick auf „Add visualization“ wählt ihr die Datenquelle und landet im Abfrage-Editor.

- **Zeitraum übernehmen**

Grafana fügt automatisch euer Dashboard-Zeitfenster ein (z. B. „Letzte 1 Stunde“).

- **Messung und Feld wählen**

Ihr gebt an, welche Messreihe (*measurement*) und welches Feld (*field*) ihr anzeigen wollt.

- **Zusätzliche Filter**

Sensor-ID oder andere Tags eingeben, um die Daten weiter einzuschränken.

- **Visualisierung prüfen**

Im Panel seht ihr sofort, ob eure Abfrage die gewünschten Kurven oder Werte liefert.

Durch diese enge Verzahnung von Abfrage und Anzeige könnt ihr in Grafana schnell verschiedene Aspekte eurer Daten ausprobieren, zum Beispiel Temperatur, Luftfeuchte und CO₂-Werte nebeneinander vergleichen oder nur einzelne Sensoren beobachten. So entstehen interaktive Dashboards, die genau das zeigen, was ihr gerade braucht



```
from(bucket: "SensorData")
  > range(start: -1h)
  > filter(fn: (r) =>
    r_measurement ==
    "sensor_data")
  > filter(fn: (r) => r_field
    = "temperature")
```

Dashboards

Übung 3 – Abfragen Ausfüllen

In diesem Abschnitt lernt ihr, wie man mit der Flux-Sprache in InfluxDB gezielt Zeitreihendaten abrufen. Ihr übt erst, einzelne Felder aus der Messung „sensor_data“ auszuwählen, und erweitert dann eure Query so, dass sie mehrere Felder gleichzeitig abrufen.

3.1

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

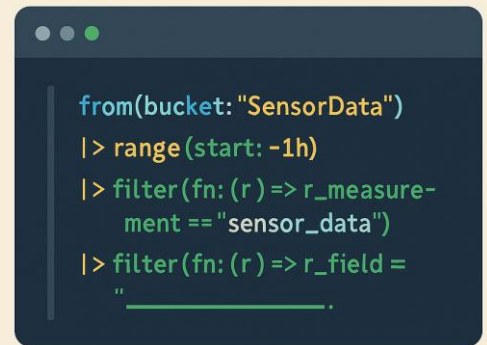
```
from(bucket: "SensorData")
  |> range(start: -1h)
  |> filter(fn: (r) => r._measurement == "sensor_data")
  |> filter(fn: (r) => r._field == "_____")
  |> filter(fn: (r) => r.sensor_id == "Sensor_1")
```

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 Query ergänzen. Zum Glück findest du einen Datenauszug:

```
temperature 20 °C , humidity 38 % , CO2 500 ppm
temperature 21 °C , humidity 41 % , CO2 600 ppm
temperature 22 °C , humidity 35 % , CO2 550 ppm
```

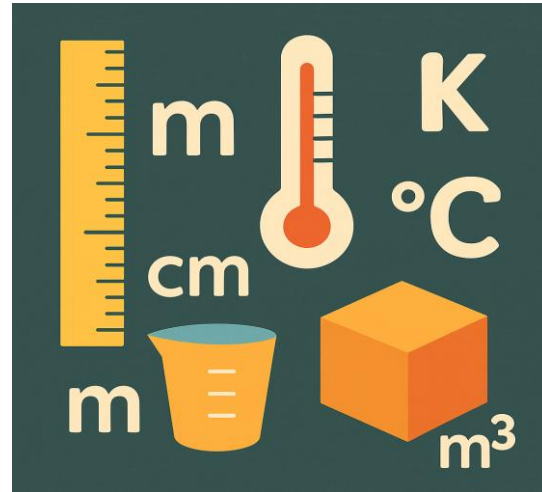
```
from(bucket: "SensorData")
  |> range(start: -1h)
  |> filter(fn: (r) => r._measurement == "sensor_data")
  |> filter(fn: (r) =>
    r._field == "_____" or
    r._field == "_____" or
    r._field == "_____"
  )
  |> filter(fn: (r) => r.sensor_id == "Sensor_1")
```



Dashboards

Einheiten

Einheiten sind unser gemeinsamer Maßstab, mit dem wir die Welt um uns herum verstehen und vergleichen können. Stellen wir uns vor, wir wollen die Länge eines Klassenzimmers messen, dafür nutzen wir das Meter (m). Oft brauchen wir kleinere Einheiten und sagen statt „1 Meter“ lieber „100 Zentimeter (cm)“, denn „Zenti-“ heißt so viel wie „ein Hundertstel“.



Auch für das Volumen, also den Raum, den etwas einnimmt, gibt es eine eigene Einheit: den Kubikmeter (m^3). Ein Würfel, dessen Seite genau 1 m lang ist, fasst genau 1 m^3 . Wenn wir hingegen wissen wollen, wie viel Platz zum Beispiel ein Liter Saft einnimmt, sprechen wir von 1 dm^3 , also einem Würfel mit 10 cm Kantenlänge.

Temperatur messen wir üblicherweise in Grad Celsius ($^{\circ}\text{C}$). Doch in der Wissenschaft kommt die Kelvin-Skala (K) zum Einsatz – sie beginnt beim absoluten Nullpunkt. 0 $^{\circ}\text{C}$ entsprechen 273,15 K. So hilft uns das internationale Einheitensystem, ob im Alltag, in der Technik oder im Labor, stets genau und verständlich zu messen.

Temperatur

Temperatur gibt an, wie warm oder kalt ein Raum oder ein Gegenstand ist. Gemessen wird sie in Grad Celsius ($^{\circ}\text{C}$), im Labor und in der Physik aber auch in Kelvin (K) – dabei entspricht 0 $^{\circ}\text{C}$ genau 273,15 K. Ein Thermometer zeigt uns auf einer Skala an, wo wir uns gerade befinden: Ist die Anzeige hoch, fühlen wir uns warm, ist sie niedrig, wird es kühl.

Für unser Wohlbefinden haben sich in Wohnräumen folgende Richtwerte etabliert:

- **Wohnzimmer:** 20–22 $^{\circ}\text{C}$ – angenehm für Alltag und Entspannung.
- **Schlafzimmer:** 16–18 $^{\circ}\text{C}$ – kühleres Klima fördert erholsamen Schlaf.
- **Küche:** 18–20 $^{\circ}\text{C}$ – etwas wärmer beim Kochen, ohne dass es zu heiß wird.
- **Badezimmer:** 22–24 $^{\circ}\text{C}$ – höhere Temperaturen sorgen für ein behagliches Gefühl beim Duschen.

Diese Idealwerte helfen uns, Energie zu sparen und gleichzeitig Komfort zu garantieren. Mit dem richtigen Thermostat und gut isolierten Fenstern lässt sich die Temperatur einfach regeln – für ein gesundes Raumklima in Schule, Zuhause oder Büro.

Dashboards

Kohlendioxid (CO₂)

Kohlendioxid (CO₂) ist ein farb- und geruchloses Gas, das beim Atmen und Verbrennen von fossilen Brennstoffen entsteht. Die Konzentration in der Luft messen wir in Parts per Million (ppm) – das heißt, wie viele CO₂-Moleküle auf eine Million Luftmoleküle kommen.

In geschlossenen Räumen beeinflusst CO₂ unser Wohlbefinden und unsere Konzentrationsfähigkeit. Typische Richtwerte sind:

- **Frischluft draußen:** ca. 400 ppm
- **Gut belüfteter Klassenraum oder Büro:** ≤ 1 000 ppm
- **Leicht erhöhte Konzentration:** 1 000–1 400 ppm (erste Anzeichen von Müdigkeit und schlechterem Lernen)
- **Hohe Konzentration:** > 1 400 ppm (Kopfschmerzen, Schläfrigkeit, schlechtere Luftqualität)

Mit einem CO₂-Messgerät oder smarten Sensoren lässt sich erkennen, wann gelüftet werden sollte. Regelmäßiges Stoßlüften (mehrmals täglich je 5–10 Minuten) hält den CO₂-Wert im grünen Bereich und sorgt für bessere Luft, mehr Konzentration und ein gesünderes Raumklima.

Luftfeuchtigkeit

Luftfeuchtigkeit beschreibt, wie viel Wasser in der Luft steckt, und wird in Prozent (%) angegeben. Ein Hygrometer misst den Anteil des tatsächlich in der Luft enthaltenen Wasserdampfs im Verhältnis zur maximal möglichen Menge bei der aktuellen Temperatur.

Für ein gesundes und angenehmes Raumklima gelten folgende Richtwerte:

- **Idealbereich:** 40–60 % – schützt Schleimhäute, beugt Erkältungen vor und verhindert Schimmelbildung.
- **Zu geringe Luftfeuchte (< 30 %):** trockene Augen, gereizte Atemwege, Rissbildung an Holzmöbeln.
- **Zu hohe Luftfeuchte (> 60 %):** erhöhte Schimmelgefahr, muffiger Geruch, beschlagene Fenster.

Tipps zur Regulierung:

- **Lüften:** Kurzes Stoßlüften (3–5 Minuten) hilft im Winter, im Sommer längeres Querlüften.
- **Luftbefeuchter oder Zimmerpflanzen:** Erhöhen die Luftfeuchte bei zu trockener Heizungsluft.
- **Raumtemperatur anpassen:** Warme Luft kann mehr Feuchtigkeit aufnehmen als kalte – ein moderates Heizen (18–20 °C) unterstützt ein ausgeglichenes Feuchtigkeitsniveau.

Dashboards

Übung 4 – Einheiten

In Aufgabe 4 übst du, verschiedene physikalische und alltägliche Größen ihren richtigen Einheiten zuzuordnen und aus Listen diejenigen Maße auszuwählen, die zu einer bestimmten Kategorie passen.

4.1

Ordne Temperatur, CO₂-Konzentration, Luftfeuchtigkeit, Beschleunigung und Zeit jeweils der passenden Einheit zu.

- | | |
|---------------------|------------------------|
| 1. Temperatur | a) Prozent |
| 2. CO ₂ | b) Meter Pro Sekunde |
| 3. Luftfeuchtigkeit | c) Kelvin |
| 4. Beschleunigung | d) Minuten |
| 5. Zeit | e) Teile pro Millionen |

4.2

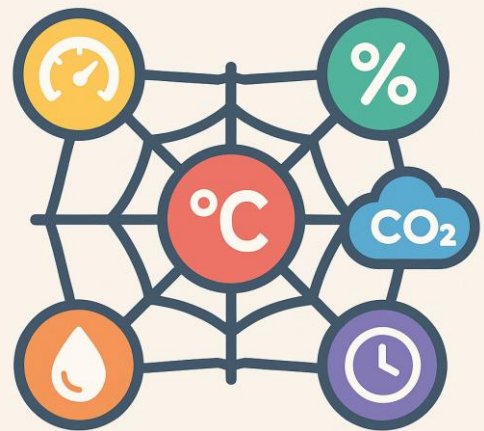
Welche der Einheiten gehören zu Temperatur?

Kelvin, Petabytes, Stunden, Grad Celsius, Watt, Hertz, Bar, Fahrenheit

4.3

Welche der Einheiten gehören zu Währung?

Gigahertz, Meilen pro Stunde, Euro, Voltage, US-Dollar



Dashboards

Einsatz von Panels

Ein Grafana-Dashboard setzt sich aus einzelnen **Panels** zusammen – das sind deine „Fenster“ in der Datenwelt, in denen du zum Beispiel ein Liniendiagramm, einen Balkenchart oder auch einfach nur eine große Zahl anzeigen kannst. Du kannst so viele Panels auf einer Seite platzieren, wie du brauchst, und sie in ihrer Größe frei anpassen. Möchtest du ein wichtiges Messsignal hervorheben, machst du das Panel groß; für weniger wichtige Informationen reichen kleinere Kästen.

Farben und Gestaltung

Damit dein Dashboard nicht nur informativ, sondern auch ansprechend aussieht, spielen **Farben und Stile** eine entscheidende Rolle. Im Bereich **Color scheme** wählst du bei Single-Value-Panels eine feste Farbe – etwa Grün für „alles gut“ oder Rot für „Alarm“. Unter **Graph styles** findest du Einstellungen wie den **Gradient mode** und die **Fill opacity**, mit denen du Linien- oder Flächendiagramme sanft verblassen lassen kannst. So lenkst du den Blick gezielt auf den oberen oder unteren Bereich deiner Kurve. Ein weiterer Feinschliff ist die **Line interpolation**: Wenn du sie auf „smooth“ stellst, zeichnet Grafana deine Datenlinie fließend und gebogen, statt eckig und kantig.

Dashboard und Zeit

Schließlich macht die **Interaktivität** dein Dashboard lebendig. Über die Zeitfilter ganz oben rechts kannst du blitzschnell zwischen verschiedenen Zeiträumen wechseln – von der letzten Stunde über den Tag bis hin zum Monat. Und sobald du mit der Maus über eine Kurve schwebst, zeigt dir der **Hover-Effekt** exakt den Messwert und den dazugehörigen Zeitstempel an. So verlierst du nie den Überblick – selbst, wenn du in die Details eintauchst.

Dashboards

Übung 5 – Gestaltung

In dieser Übung lernst du, wie du aus rohen Messdaten ein ansprechendes und gut lesbares Dashboard in Grafana gestaltest. Zuerst öffnest du dazu die Datei **simulation.py** in deiner Entwicklungsumgebung und trägst dort deine InfluxDB-Zugangsdaten ein. Damit fließen deine selbst erstellten Sensordaten automatisch in Grafana.

Anschließend erstellst du in Grafana ein neues Dashboard und fügst ein **Panel** hinzu, das den aktuellen Messwert übersichtlich als Einzelwert oder kleine Grafik darstellt



5.1

Suche unter Search options nach „Color scheme“, wähle Single Color und wähle eine Farbe aus.

Gewählte

Farbe: _____

5.2

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

Fill opacity: _____

5.3

Setze „Line Interpolation“ auf smooth, Notiere was sich ändert.

5.4

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

Dashboards

Darstellungsarten in Grafana

Grafana stellt dir eine Vielzahl von Paneltypen zur Verfügung, um unterschiedlichste Daten optimal zu visualisieren. Ob du Zeitreihen, Einzelwerte, Anteile oder Tabellen zeigen möchtest – für jeden Anwendungsfall gibt es das passende Format. Im nächsten Schritt lernst du nicht nur fünf dieser Darstellungsarten kennen, sondern entscheidest auch selbst, welche Form für welche Fragestellung am besten geeignet ist.

Wann welche Darstellungsart wählen?

Wann wählst du welche Darstellungsart?

- **Zeitreihen (Time series / Line Chart):** Ideal, wenn du Messwerte kontinuierlich über einen Zeitraum darstellen willst – zum Beispiel Temperatur- oder CO₂-Verläufe.
- **Einzelwert-Panel (Stat / Gauge):** Perfekt, um aktuelle Kennzahlen wie Luftfeuchtigkeit oder CPU-Auslastung als große Zahl bzw. Anzeige darzustellen.
- **Balken- und Säulendiagramme (Bar Chart):** Gut geeignet für Vergleiche zwischen Kategorien, etwa Umsätze pro Monat oder Schüler nach Notenstufen.
- **Kuchendiagramm (Pie Chart):** Zeigt Anteile an einer Gesamtheit, wie Marktanteile, Nutzungsarten oder das Mädchen-Jungen-Verhältnis in der Klasse.
- **Tabelle (Table):** Dient zur Auflistung von Einzelwerten, Logdaten oder einer Historie von Ereignissen – zum Beispiel deinem Klassenbuch oder einer Liste von Messzeitpunkten.

Neben den bekannten Panels gibt es in Grafana noch viele weitere Möglichkeiten, deine Daten anschaulich darzustellen. Möchtest du etwa herausfinden, wie sich Messwerte verteilen, nutzt du ein **Histogramm**, das Häufigkeiten in Balken zusammenfasst. Für komplexe Muster über Zeit und Kategorien eignet sich eine **Heatmap**, in den Farbabstufungen die Intensität anzeigen. Wenn du Log-Dateien oder Ereignisse direkt im Dashboard durchstöbern willst, kommt das **Logs Panel** zum Einsatz, während eine **Worldmap** oder **Geomap** perfekt ist, um geografische Daten wie Nutzerstandorte oder Temperaturen in verschiedenen Regionen zu visualisieren. Mit einem **Text-Panel** kannst du schließlich deinen Grafiken Erklärungen, Links oder Hinweise hinzufügen, und das **Status Panel** zeigt dir Zustandswechsel über die Zeit als farbige Blöcke an.

Dashboards

Übung 6 – Darstellung

In Aufgabe 6 übst du, die verschiedenen Visualisierungsarten in Grafana zu benennen, zu entscheiden, welche Darstellung für welche Daten am besten geeignet ist, und deine Wahl durch praktisches Erstellen und Vergleichen von Panels in deinem Dashboard zu überprüfen.

6.1

Nenne 5 Darstellungsarten die Grafana anbietet:

- 1: _____
- 2: _____
- 3: _____
- 4: _____

6.2

Welche Darstellungsart eignet sich für...

Vermögenswachstum über Zeit: _____

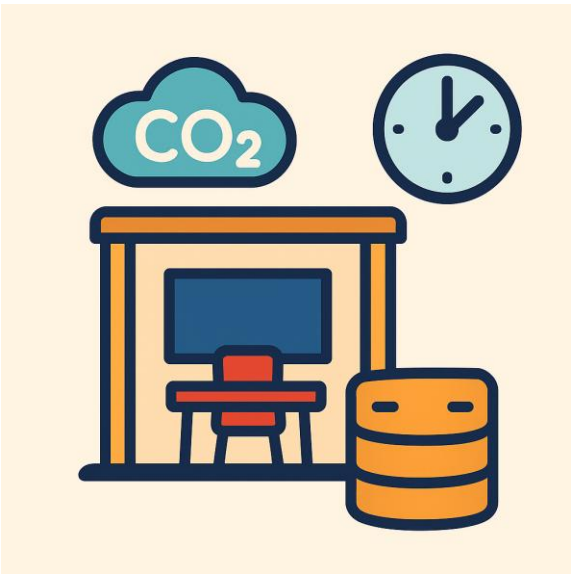
Geschwindigkeit von Autos: _____

Mädchen-Jungs Verhältnis in der Klasse: _____

6.3

Klassenbuch:

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.



Dashboards

Darstellungsarten Wahl

Welche Darstellungsart du wählst, hängt also immer vom Datentyp und deiner Fragestellung ab: Zeitreihen zeichnest du als Liniendiagramm oder Heatmap, Verteilungen als Histogramm, Einzelwerte mit Stat oder Gauge, Kategorienvergleiche mit Balken- oder Kreisdiagramm und räumliche Informationen auf einer Karte. So findest du für jeden Anwendungsfall die passende Visualisierung – und machst dein Dashboard klar, übersichtlich und aussagekräftig.

Thresholds

In Grafana kannst du mit sogenannten **Thresholds** (Schwellenwerten) genau festlegen, ab wann sich die Farbe oder das Aussehen eines Panels ändert, um kritische Zustände sofort erkennbar zu machen. Du gibst dafür eine oder mehrere Grenzwerte ein, zum Beispiel 800 ppm für CO₂ im Klassenzimmer, und wählst für jeden Bereich eine Farbe: Grün bis 800 ppm, Gelb bis 1 200 ppm und Rot ab 1 200 ppm. Sobald dein Messwert in einen dieser Bereiche fällt, passt Grafana die Farbe automatisch an und signalisiert dir so auf einen Blick, ob alles in Ordnung ist oder ob du dringend lüften musst. Thresholds helfen also dabei, wichtige Abweichungen sofort sichtbar zu machen, ohne dass du die genauen Zahlen im Detail studieren musst.

Übung 7 – Thresholds

In Aufgabe 7 übst du, wie du aus deinen Sensordaten in Grafana aussagekräftige Warn- und Komfortbereiche definierst: Du richtest dein Skript **temperature.py** so ein, dass es die Werte in InfluxDB schreibt, legst dann ein neues Dashboard an und fügst ein Panel hinzu, das den aktuellen Temperaturwert anzeigt.

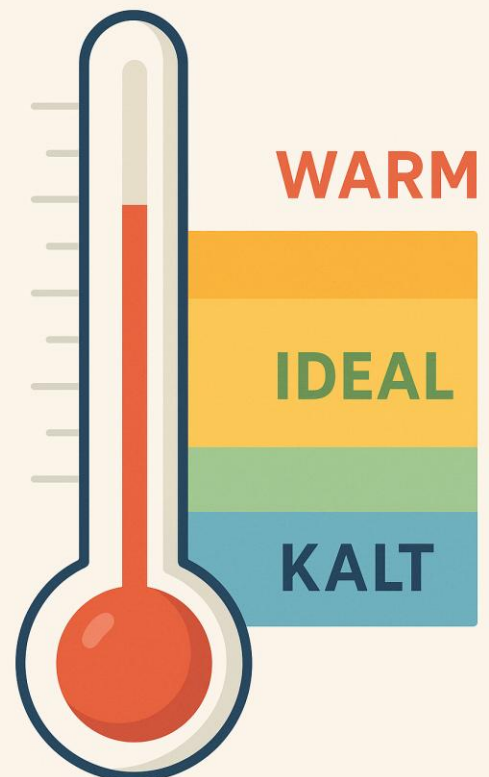
7.1

Kalt, alles unter: ____ °C

Ideal zwischen: ____ °C - ____ °C

Warm alles über: ____ °C

Zu heiß, alles über: ____ °C



Dashboards

7.2

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

Herzlichen Glückwunsch – du hast alle bisherigen Aufgaben erfolgreich gemeistert und bist nun bestens gerüstet für die **Bonusaufgabe**!

Multipanels

In dieser Bonusrunde geht es um **Multipanels**. Ein Multipanel-Dashboard besteht aus mehreren nebeneinander angeordneten Panels, in denen jeweils unterschiedliche Messwerte als eigene Liniendiagramme dargestellt werden. So bekommst du einen kompakten Überblick über alle relevanten Daten auf einen Blick, kannst leicht Zusammenhänge erkennen und verschiedene Werte direkt miteinander vergleichen. Gerade wenn mehrere Sensoren parallel laufen oder unterschiedliche Kennzahlen gleichzeitig interessant sind, sind Multipanels unverzichtbar für ein klares, strukturiertes Monitoring.

Bonusaufgabe

Öffne die Datei **multipanel.py** in deiner IDE und trage wie gewohnt deine InfluxDB-Zugangsdaten ein. Lege dann ein neues Dashboard in Grafana an und füge für jeden verfügbaren Messwert ein eigenes Panel hinzu – jeweils als Liniendiagramm. Schau abschließend in deiner InfluxDB nach, welche Werte gespeichert werden, und überprüfe, ob alle Daten korrekt und übersichtlich in deinem Multipanel-Dashboard angezeigt werden. Viel Spaß beim Tüfteln und Ausprobieren!



