**Dokumentation: (Modulare) Wetter‑ und Luftqualitätsstation**

Konrad - Hardwareprojekt – Sommersemester 2025

**1 Einleitung**

Die Luftqualität in Innenräumen hat direkten Einfluss auf Gesundheit, Wohl‑befinden und Konzentrationsfähigkeit. Gleichzeitig ist es für Bastlerinnen und Bastler reizvoll, mithilfe günstiger Mikrocontroller eigene Mess‑, Anzeige‑ und Alarm­systeme aufzubauen. Ziel dieses Projekts ist deshalb der Entwurf einer **Wetter‑ und Luftqualitäts‑station**, die

* **Innen‑ wie Außenmessungen kombiniert** (VOC‑basiertes IAQ, eCO**₂**, Temperatur, rel. Luft‑feuchte und UV‑Index),
* ihre Daten untereinander **per MQTT** auszutauschen und eine Erweiterung über OpenHAB / Webserver ermöglicht,
* **Warnungen** akustisch (Buzzer) und optisch (RGB‑Ampel) signalisiert,
* Mess­reihen langfristig loggt und visualisiert.

**2 Projektziel**

1. **Innenstation** auf Raspberry Pi 5 mit BME680 Sensor, Schalter, RGB‑Ampel, Buzzer und 16×2‑LCD.
2. **Außenstation** auf ESP32 (DHT22 + UV‑Sensor) Deep‑Sleep.
3. **Daten­transport** über MQTT / WLAN; Verarbeitung, Logging und Charts intern oder über OpenHAB.
4. **Dokumentation** Hardware, Software und Erweiterungs­potenzial.

**3 Komponentenübersicht**

|  |  |  |
| --- | --- | --- |
| **Kategorie** | **Bezeichnung** | **Zweck** |
| Mikrocontroller | **Raspberry Pi 5** | Innenstation, Python‑Stack |
|  | **ESP32 DevKit‑C** | Außenstation, MQTT‑Sender |
| Sensoren | **BME680 (I²C)** | Temp., Feuchte, gVOC/IAQ, **eCO₂ ppm** |
|  | **DHT22** | Temp. + rF (Outdoor) |
|  | **GUVA‑S12SD** | UV/Licht‑Verhältnisse |
|  | **Button** | LCD Menüsteuerung |
| API | **OpenUv.io** | Aktuelle UV‑Werte für den gesamten Ort |
| Aktoren | **RGB‑LED (GPIO 5/26/6)** | Ampelanzeige IAQ |
|  | **Piezo‑Buzzer (GPIO 24, PWM)** | Akustischer Alarm |
| Anzeige | **LCD 16×2, I²C PCF8574** | Live‑Werte & Menü |
| Sonstiges | Flachbandkabel, Gehäuse‑Set, LCD und BME Gehäuse (3D‑Druck), Breadboard | Strom & Aufbau |

**4 Systemarchitektur**

OpenUV

│

│

┌──────────────┐ MQTT/Wi‑Fi ┌──────────────┐

│ ESP32 │───────────────>│ Raspberry Pi │

│ (Außen) │ │ Innenstation │

│ DHT22 + UV │<────────────── │ BME680 │

└──────────────┘ „Heartbeat“ └──────────────┘

▲ ▲

│ Webserver/ │

└─────────> OpenHAB <──────┘

**5 Hardware‑Aufbau**

**5.1 Innenstation**

* **BME680** an I²C‑Bus 1, Adresse 0x77.
* **LCD 16×2** an I²C‑Bus 1, Adresse 0x27.
* **RGB‑LED** auf GPIO 5 (R), 26 (G), 6 (B).
* **Button** auf GPIO 17.
* **Piezo‑Buzzer** als PWM‑Ausgang auf GPIO 24.

Alle Komponenten werden vom Pi gespeist.

Ein Bild, das Elektronik, Kabel, Im Haus, Elektrische Leitungen enthält.

KI-generierte Inhalte können fehlerhaft sein.Ein Bild, das Elektronik, Uhr, Text, Elektronisches Gerät enthält.

KI-generierte Inhalte können fehlerhaft sein.

**5.2 Außenstation**

* **ESP32 DevKit‑C**
* **DHT22** an GPIO 4.
* **GUVA‑S12SD** UV‑Sensor am Analog‑Pin ADC1\_CH6 = GPIO34.
* Versorgung: Flachbandkabel durch das geschlossene Fenster.
* Deep‑Sleep 5 min

Ein Bild, das Person, draußen, Hand, Auto enthält.

KI-generierte Inhalte können fehlerhaft sein.Ein Bild, das draußen, Fenster, Kompositmaterial, Gebäude enthält.

KI-generierte Inhalte können fehlerhaft sein.

**6 Software‑Implementierung**

**6.1 Python‑Stack (Innen)**

* **main.py** orchestriert Threads für BME680‑Update, Ampel‑Logik, LCD‑Menü und MQTT‑Publishing.
* **BME680.py** kapselt Bosch BSEC‑Bibliothek, lädt/ speichert Kalibrier‑State und liefert IAQ, eCO₂ etc.
* **Ampel.py** wertet Qualitäts­klassen aus und steuert RGB + Buzzer (Debounce‑Timer 3 Zyklen).
* **Datenbank.py** puffert Messungen, bildet 30 s‑Mittel und schreibt CSV‑Zeilen.
* **Mqtt.py** stellt EspAußen‑Subscriber und MQTTPublisher‑Helper bereit.
* **lcd.py** bespielt das LCD, zeichnet eigene Symbole (ppm, IAQ, Sonne) und verwaltet Taster‑Menüs.
* **Buzzer.py** steuert den Buzzer an.
* **checkQuality.py** bewertet Raum‑Werte und stellt diese der Ampel‑Klasse aufbereitet bereit.
* **UvApi.py** fragt UV‑Werte von der API halbstündlich ab.
* **Auswertung.py** plottet die Werte in der CVS Datei zu Graphiken

**6.2 Arduino‑Stack (Außen)**

* **DHT\_Unified\_Sensor.ino** verbindet sich bei Wake‑Up mit WLAN, misst DHT22 & UV und publisht {"temp":…, "hum":…, "uv":…} auf wetterstation/esp32. Danach Deep‑Sleep 5 min.

**6.3 MQTT‑Topics**

wetterstation/

├─ esp32/

│  ├─ temperature

│  ├─ humidity

│  ├─ sun\_raw

│  └─ sun\_kategorie

└─ indoor/

├─ temp

├─ humidity

├─ iaq

└─ eco2

Alle Topics sind **retained**, QoS 1.

**6.4 OpenHAB**

* MQTT‐Thing „Innenstation“ und „Außenstation“ + Broker (Pi).
* Für jede einzelne Kategorie Channel + Link.
* Main‑Page zeigt:
  + Innen: Temperatur °C, rF %, IAQ‑Index, eCO₂ ppm.
  + Außen: Temperatur °C, rF %, UV (API), Sonne Kategorie (von Raw abgeleitet), UV Raw.

Ein Bild, das Screenshot, Diagramm, Design enthält.

KI-generierte Inhalte können fehlerhaft sein.

**7 Funktionsablauf**

1. **Raspi-/Boot** – Programm wird automatisch gestartet. (systemd)
2. **Kalibrierung (≈ 5 – 10 min)** – BME680 State wird inkrementell verbessert.
3. **Mess­schleife** (alle 3 s): BME680‑Daten lesen   
   ➜ Ampel‑Thread wer­tet Qualitäts­klassen 0/1/2 aus.   
   ➜ Button-Thread steuert LCD-Menü
4. **MQTT‑Publish** (alle 30 s): Innenwerte / ESP published outdoor werte selbst 5 minütlich.
5. **Datenbank** schreibt alle 5 min in CSV, mittelt die Innenwerte.
   * D.h : timestamp,iaq,eco2\_ppm,hum\_rel,temp\_c,temp\_out,hum\_out,uv\_kategorie,**uv\_raw,**uv\_api  
      2025-06-03T05:52:39,88.58,1109.01,58.21,21.56,21.4,55.0,0.0,**0,**0.04  
      2025-06-03T05:58:07,90.39,1130.98,58.3,21.57,21.4,55.0,0.0,**0,**0.04  
      2025-06-03T06:03:08,85.53,1063.22,58.41,21.56,21.4,55.0,0.0,**0,**0.04  
      2025-06-03T06:08:37,85.65,1062.61,58.4,21.57,21.4,55.0,0.0,**0,**0.04  
      2025-06-03T06:13:37,86.37,1069.95,58.44,21.57,21.4,55.0,0.0,**0,**0.04  
      2025-06-03T06:18:38,86.47,1069.09,58.61,21.57,21.4,55.0,0.0,**0,**0.11

**8 Limitationen & Ausblick**

* **Außenstation** mit Solar/Powerbank betreiben, keinen Stromanschluss mehr nötig.
  + Bsp; „Stromaufnahme ≈ 150 µA ⇒ 2 000 mAh‑Powerbank hält ≈ 11 Tage.“   
    -ChatGPT
* **LCD nur aktualisieren, wenn Werte sich ändern.**
* **LCD echter Power‑Off** per externem „Stromschalter“.
* **UV‑Sensor** verbessern, mehr Grad am Himmel abdecken.
* **Push‑Benachrichtigungen** bei schlechter Qualität.
* **Gehäuse‑Revision**.