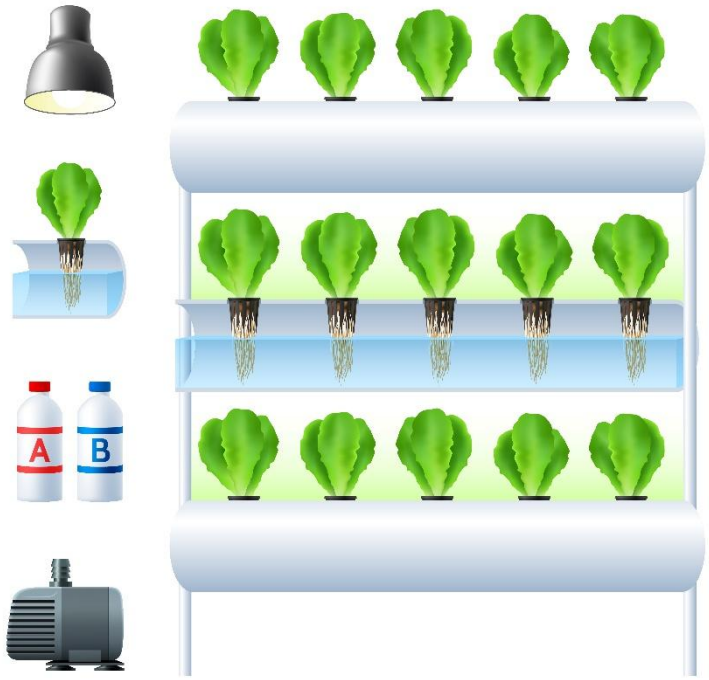
Im Rahmen des Moduls „Projektarbeit“

Indoor Farming Station



Koordination: Prof. Dr.-Ing. Pascal Stoffels
Matthias Wilbert
Thorsten Murach

Projektpartner:

Name	Matrikelnummer	E-mail
Steven Haupenthal	5013788	stha00005@htwsaar.de
Alexander Masseli	5014288	alma00016@htwsaar.de
Daniel Shapiro	5015020	dash00004@htwsaar.de

Indoor Farming Station

Indoor Farming Station

1. Vorhaben

1.1 Anforderungen

Das Ziel unseres Projektes besteht darin, ein bestehendes Indoor-Farming-System durch eine auf einem Pi basierende digitale Lösung möglichst effizient zu überwachen und die dabei gewonnenen Daten für die Zukunft nutzbar zu machen. Hierfür wurden verschiedene Sensoren am Pi angeschlossen (Temperatur, Luftfeuchtigkeit usw.).

Alle von den Sensoren gesammelten Daten werden zentral auf dem Pi erfasst, gespeichert und in einer von uns entworfenen Weboberfläche dargestellt. Diese Visualisierung erfolgt durch Diagramme sowie Tabellen und andere Möglichkeiten. Zusätzlich haben wir uns das Ziel gesetzt, unser gesamtes Projekt möglichst modular zu gestalten, um eine Erweiterbarkeit gewährleisten zu können, sodass bei Bedarf die von uns aufgebaute Kontrollinstanz ohne große Hürden erweitert werden kann.

1.2 Projektplan

Arbeitspakete im Detail:

AP1: Einrichtung des Raspberry Pi

- Zeitraum: 11. Mai
- Inhalt: Grundkonfiguration, Repository erstellt, Entwicklungsumgebung eingerichtet
- Ergebnis: Raspberry Pi betriebsbereit.

AP2: Anschließen und Testen der Sensoren

- 11. Mai – 6. Sep
- Inhalt: Schrittweise Integration der Sensoren. Anpassung einzelner Sensoren sowie laufende Tests der Funktionsfähigkeit.
- Ergebnis: Alle Sensoren liefern zuverlässig Werte.

AP3: Aufsetzen der Website (Dash)

- Zeitraum: 21. – 26. Mai
- Inhalt: Umsetzung einer funktionierenden „Baseline“ mit Login-/Register-Seite und Admin/User-Seiten. Navigation und Basis-Dashboard.
- Ergebnis: Erste Website mit Admin/User-Verwaltung.

AP4: Erstellung und Erweiterung der Datenbank

- Zeitraum: 11. Mai – 7. Sep
- Inhalt: Entwicklung einer SQLite-Datenbank. Mehrfach angepasst (z. B. relativer Pfad, Exportfunktionen als CSV). Tests durchgeführt. Speicherung der Sensordaten (z. B. Temperatur, Feuchtigkeit, Wasserstand) erfolgreich.
- Ergebnis: Stabile Datenbank mit Export und Logging.

AP5: Anzeige und Visualisierung der Sensordaten auf der Website

- Zeitraum: 22. Mai – heute
- Inhalt: Visualisierung der Sensorwerte in Tabellen und Graphen. Optimierungen am Dashboard (z. B. Buttons für Graphen, Anzeige der letzten 24h). Logs für Sensorwerte implementiert.
- Ergebnis: Website zeigt Sensorwerte in Echtzeit inkl. Logs und Exportmöglichkeiten.

AP6: Erweiterung der Steuerung (Pumpen, Lüfter, Licht, User-Dashboard)

- Zeitraum: 1. Sept – heute
- Inhalt: Integration von Steuerungsmöglichkeiten für Pumpen, Lüfter und Licht. Statusanzeige über Buttons (an/aus). Pumpenzeitschalter hinzugefügt. User-Dashboard mit begrenzten Möglichkeiten erstellt. Anzeige von Lüfter- und Pumpenstatus auf der Website.
- Ergebnis: Neben der Überwachung sind nun auch Steuerungsfunktionen verfügbar, die das System deutlich praxisnäher machen.

Meilensteine:

MS1 (11. Mai): Raspberry Pi eingerichtet

MS2 (21. Mai): Website-Grundgerüst steht

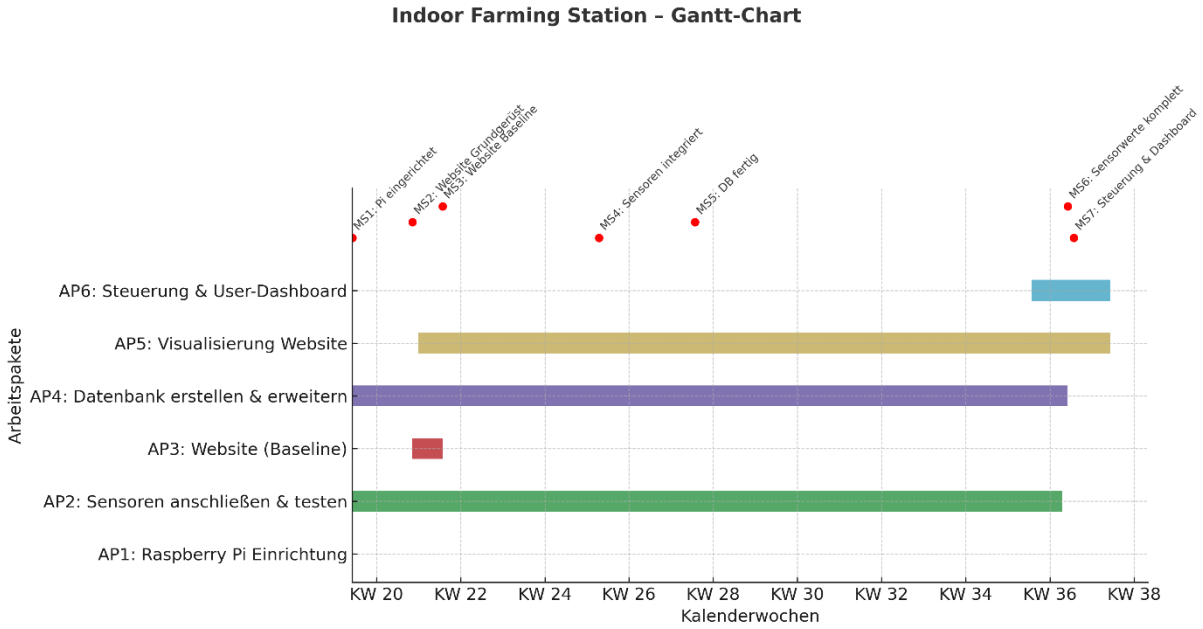
MS3 (26. Mai): Funktionierende Baseline der Weboberfläche

MS4 (21. Juni): Sensoren vollständig integriert

MS5 (7. Juli): Datenbank implementiert und an Website angebunden

MS6 (7. Sep): Alle Sensorwerte werden gespeichert und angezeigt

MS7 (8. Sep): Steuerung von Pumpen, Lüftern und Licht sowie User-Dashboard fertiggestellt



2. Umsetzung

2.1 Reflektion

In unserem Projekt konnten wir bisher wesentliche Meilensteine erreichen und erfolgreich umsetzen. Von der Einrichtung des Pi über die Anbindung der Sensoren sowie das Aufsetzen der Website mittels Dash hatten wir viele verschiedene Aufgaben zu bewältigen. Durch die Umsetzung unserer Ideen und Besprechungen konnten wir bisher ein Minimum Viable Product (MVP) erstellen, das wir stetig bis zur endgültigen Deadline verbessern werden.

Die ursprünglichen Ziele konnten unseres Erachtens bisher mehr als ausreichend abgedeckt werden. Allerdings wollen wir diese, wenn es die Zeit erlaubt, sogar übertreffen, um unser Können und unsere Bewertung möglichst positiv zu beeinflussen.

Die Aufgabenverteilung im Team war sehr gut. Jeder übernahm zunächst Verantwortung für bestimmte Arbeitspakete, und bei komplexeren Arbeitspaketen wurde gemeinsam daran gearbeitet. Probleme traten zwar immer wieder auf, wie zum Beispiel, dass wir aufgrund technischer Schwierigkeiten mit Account-Daten nicht mehr richtig über unsere eigenen Konten committen konnten. Dadurch war eine genaue Einschätzung des Arbeitsaufwands für persönliches Interesse nicht möglich. Des Weiteren hatten wir leider auch Probleme mit einigen Sensoren, da einer defekt war. Die Fehlersuche in solchen Fällen nahm viel Zeit in Anspruch und schaffte Unsicherheit in den jeweiligen Situationen.

2.2 Architektur und Technologie Stack

Die entwickelte Lösung basiert auf einer **modularen Client-Server-Architektur**:

- **Hardware-Ebene:** Raspberry Pi als zentrale Steuereinheit, an den die Sensoren (Temperatur, Luftfeuchtigkeit, Wasserstand, pH, TDS) angeschlossen sind.
- **Datenerfassung:** Python-Skripte lesen die Sensordaten aus und speichern sie in einer SQLite-Datenbank.
- **Backend:** Flask/Dash bildet die Schnittstelle zwischen Datenbank und Web-Frontend. Hier werden APIs bereitgestellt und Dashboards mit Live-Daten aktualisiert.
- **Frontend:** Dash-Weboberfläche mit Login-/Registrierungssystem, Admin-/User-Dashboard, Visualisierung der Sensordaten (Graphen, Tabellen) sowie Buttons zur Steuerung der Aktoren.
- **Datenbank:** SQLite als leichtgewichtige Lösung, die lokal auf dem Raspberry Pi läuft.
- **Steuerung:** GPIO-Bibliotheken von Python ermöglichen die Ansteuerung der Aktoren (Pumpen, Lüfter, Licht).

Damit wurde ein schlanker, erweiterbarer **Technologie-Stack** realisiert, der alle Anforderungen des MVP erfüllt und eine gute Basis für spätere Erweiterungen bietet.

2.3 Features (MVP)

Benutzerverwaltung

- Registrierung und Login-System mit Admin-/User-Rollen.
- Eingeschränktes User-Dashboard und erweitertes Admin-Dashboard.

Sensorintegration

- Erfassung von Temperatur, Luftfeuchtigkeit, Wasserstand, pH-Wert und TDS.
- Speicherung aller Messwerte in einer Datenbank.
- Austausch und Erweiterung von Sensoren möglich (z. B. Ultraschall- durch Wasserstandsensor ersetzt).

Datenbankfunktionen

- Speicherung der Sensordaten in SQLite.
- Exportfunktion der Daten als CSV-Datei.
- Optimierung durch relative Pfade und zusätzliche Tabellenstrukturen.

Visualisierung

- Darstellung der Sensordaten in Echtzeit.
- Graphen für Zeitverläufe (inkl. 24-Stunden-Darstellung).
- Log-Funktion für Sensorwerte.

Steuerung

- Bedienung von Pumpen, Lüfter und Licht über Buttons in der Weboberfläche.
- Statusanzeige, ob ein Aktor an/aus ist.
- Zeitschaltung für die Pumpe.

Damit erfüllt das MVP alle Kernanforderungen: **Überwachung + Steuerung**.

2.4 Warnungen und Limits

Um einen sicheren Betrieb des Systems gewährleisten zu können, sind Sicherheitsmechanismen integriert, die bei Abweichungen von definierten Grenzwerten Warnungen ausgeben. Die Meldungen erscheinen im Dashboard im Sektor Log und sollen dafür sorgen das mögliche Fehler oder Anomalien aufgezeigt werden können, damit rechtzeitig Maßnahmen ergriffen werden können, um Schäden oder Ausfälle zu vermeiden.

Die Warngrenzen können bei Bedarf im Code angepasst werden. Sie sind in der Datei **admin_dashboardpage.py** hinterlegt. Um den entsprechenden Abschnitt zu finden, muss im Code nach „Warngrenzen“ gesucht werden.

Dort ist die Konstante **SENSOR_LIMITS** definiert. In dieser Struktur lassen sich die Mindest- und Maximalwerte einzelner Sensoren anpassen. Beispiel:

Name Sensor	Mindestwert	Maximalwert
"EC_Sensor":	0.6	1.8

2.5 Sensoranbindung

Übersicht

Dieses Python-Skript dient der Steuerung und Überwachung eines automatisierten Hydroponik- oder Bewässerungssystems. Es integriert mehrere Sensoren und Aktoren, speichert Daten in einer SQLite-Datenbank und ermöglicht die zyklische Steuerung von Pumpe, Lüfter und Beleuchtung.

Das System nutzt folgende Komponenten:

- **Sensoren**: DHT11, DS18B20, pH-Sensor, TDS-Sensor, Wasserstandssensor (ADS1115 & Ultraschall)
- **Aktoren**: Pumpe, Lüfter, LED-Licht
- **Datenbank**: SQLite (sensors.db)
- **Flowrate-Messung**: Impulszähler über GPIO

1. Hardware-Initialisierung

1.1 LEDs / Beleuchtung

- Pump LED: GPIO 12 (Pin 32) → zeigt Pumpenzustand an
- Light LED: GPIO 6 (Pin 33) → simuliert Lichtsteuerung

1.2 Lüfter

- Fan: GPIO 26 → gesteuert über PWM (PWMOutputDevice)
- Intensität wird über Datenbank konfiguriert: Stark/Mittel/Schwach

1.3 DHT11 Sensor

- Misst Luftfeuchtigkeit und Temperatur
- GPIO 17 (Pin 1)

1.4 Wasserstandssensor

- Über ADS1115 ADC, Kanal 3
- Misst analog den Wasserstand im Tank und konvertiert ihn in Prozent

1.5 Ultraschall-Sensor

- Trigger: GPIO 22 (Pin 15)
- Echo: GPIO 27 (Pin 13)

- Misst Tankfüllstand in cm → wird in Prozent umgerechnet
- Parameter: d_max=50 cm (leer), d_min=5 cm (voll)

1.6 pH-Sensor

- Über ADS1115 ADC, Kanal 2
- Kalibriert mit Messpunkten für niedrigen (mess_n) und hohen pH-Wert (mess_h)
- Berechnet pH-Wert aus Spannung

1.7 TDS/EC-Sensor

- Über ADS1115 ADC, Kanal 1
- Misst Leitfähigkeit (TDS) des Wassers
- Temperaturkompensation wird berücksichtigt

1.8 Temperatur-Sensor DS18B20

- Misst Wassertemperatur
- Wird direkt über W1ThermSensor ausgelesen

1.9 Flowrate-Sensor

- GPIO 24
- Zählt Impulse, um den Durchfluss in L/min zu berechnen
- Pulszählung über Button.when_pressed

2. Software-Hilfsfunktionen

- count_pulse(): Zählt Impulse des Flowrate-Sensors
- parse_time(value): Konvertiert Strings in datetime.time
- update_sensor_state(key, value): Aktualisiert Sensorwerte im Dictionary
- safe_round(value, ndigits=2): Rundet Werte auf 2 Nachkommastellen, falls nicht None

3. Sensorzustände

Alle Sensorwerte werden im Dictionary sensor_state gespeichert:

```
sensor_state = {
    "humidity": None,
```

```
    "temperature": None,
    "water_level": None,
    "ultrasonic": None,
    "ph": None,
    "tds": None,
}
```

4. Datenbank (SQLite)

- Pfad: ./SQLite/sensors.db
- Tabellen:
    - Humidity_Sensor
    - Temp_Sensor
    - Ultrasonic_Sensor
    - PH_Sensor
    - EC_Sensor
    - WaterLevel_Sensor
    - FlowRate_Sensor
    - Pump
    - Fan
    - Light

4.1 Datenbankfunktionen

- safe_db_execute(query, params): Führt Abfragen aus (SELECT, INSERT, UPDATE)
- safe_db_fetchone(query, params): Liefert erste Zeile einer SELECT-Abfrage

5. Konfigurationsfunktionen

- Pumpen-Zyklus: intervall (min), on_for (min) aus DB
- Lüfter-Zyklus: intervall (min), on_for (min) aus DB
- Lichtsteuerung: Zwei Lichtphasen aus DB (start_time, end_time)

6. Sensor-Auslesung

6.1 sensor_activate()

- Liest alle Sensoren aus (Wasserstand, Ultraschall, pH, TDS, Temperatur DS18B20)
- Berechnet Füllstand / pH / TDS
- Speichert Live-Werte in DB

6.2 read_dht()

- Liest DHT11 aus (Feuchtigkeit)
- Aktualisiert Sensorstate und DB

7. Pumpen- / Flowrate-Zyklus

- Funktion: pump_and_waterflow_cycle()
- Zyklisches Ein- und Ausschalten der Pumpe
- Während der Laufzeit wird die Flowrate berechnet:

$$Flow\left(\frac{L}{min}\right) = \frac{Impulse}{Zeit(s)} \times \frac{60}{K}$$

- Status wird in DB gespeichert (online/offline)

8. Lüfter-Zyklus

- Funktion: fan_cycle_and_intensity()
- Zyklisches Ein- und Ausschalten mit Intensität (Stark, Mittel, Schwach)
- PWM-Wert wird aus DB ausgelesen (1.0 / 0.75 / 0.5)

9. Lichtsteuerung

- Funktion: light_cycle()
- Zwei mögliche Lichtphasen, werden mit aktuellem Zeitpunkt verglichen
- LED wird entsprechend geschaltet

2.6 Dokumentation

Starten des Projekts

Um das MVP zu starten, sind folgende Schritte notwendig:

1. Raspberry Pi vorbereiten

- Raspberry Pi OS installieren.
- Python 3 sowie Pip aktualisieren:

  Bash:

  sudo apt-get update && sudo apt-get upgrade

sudo apt-get install python3 python3-pip

- Repository klonen

  Bash:

  git clone <https://github.com/SensenTV/Farming-Station>

  cd <projektordner>

- ***Wichtige Libraries:***

  Flask / Dash (für Backend & Frontend)

  SQLite3 (Datenbankanbindung)

  Plotly (Visualisierungen)

  RPi.GPIO (Ansteuerung der Hardware)

- Datenbank initialisieren

  Skript init_db.py ausführen, um Tabellen zu erstellen.

- Projekt starten

  Bash:

  python app.py

Die Website ist dann unter http://<raspberrypi-ip>:8050 erreichbar.