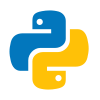
****

Ipek, Atilla & Roser, Kevin

27.05.2025| Gewerblich-Technische Schule Offenburg, E1FI1 2024/2025, Hug Christopher

Internet der Dinge

mit Wemos D1 Mini ESP8266

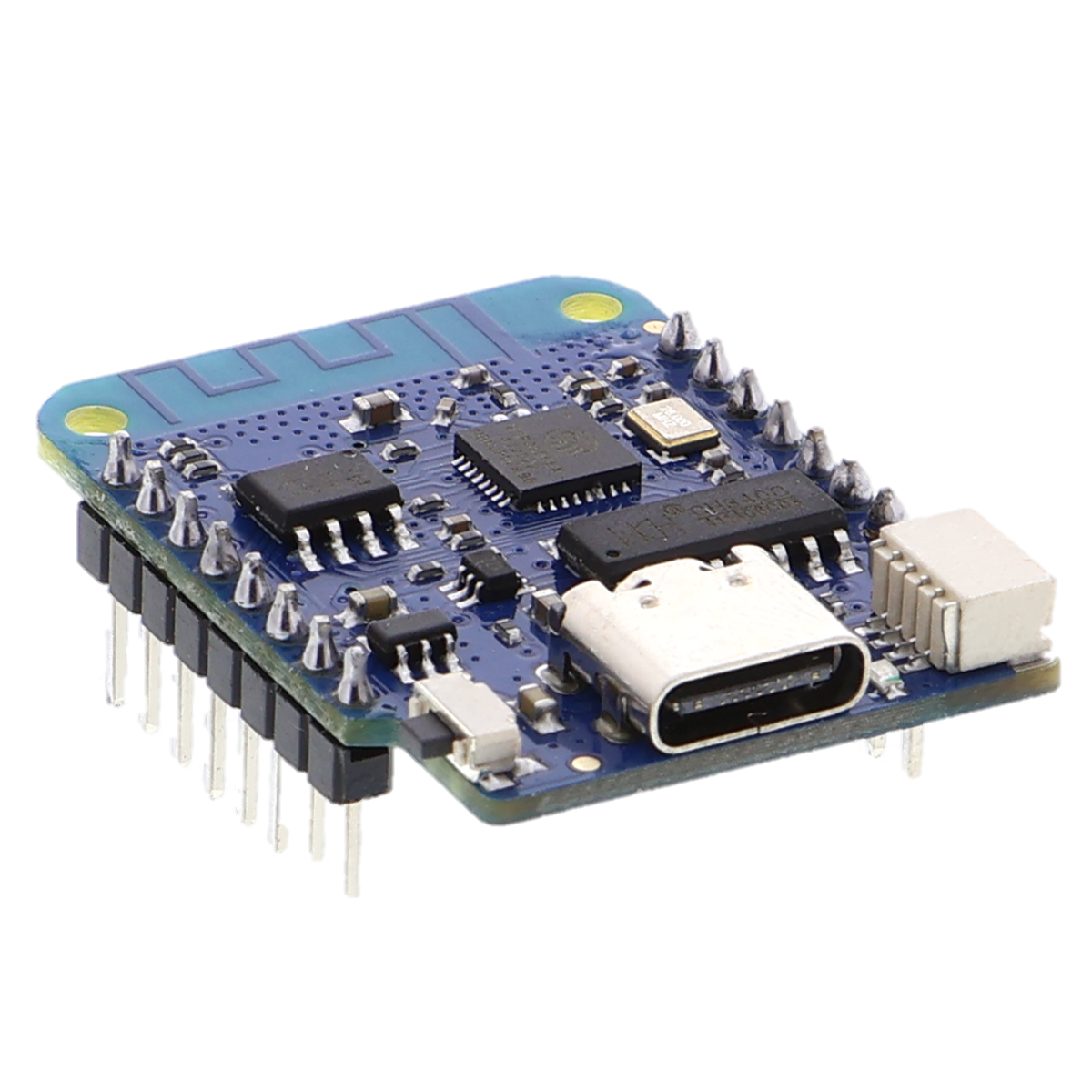
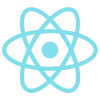
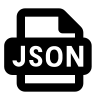
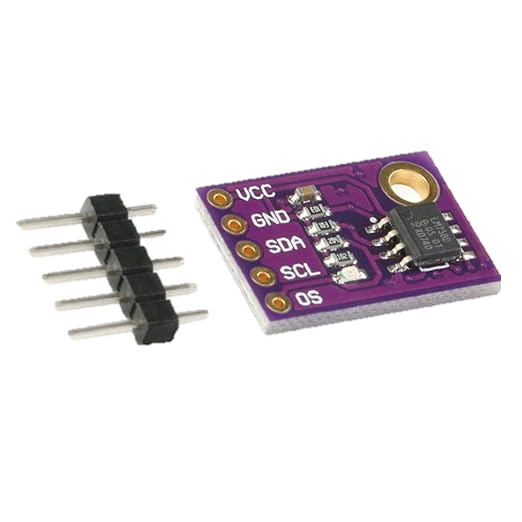
****

Abbildung 1

Abbildung 2

Inhaltsverzeichnis

[1. Einleitung 3](#_Toc199505577)

[1.1 Motivation/Problemstellung 3](#_Toc199505578)

[1.2 Zielsetzung 3](#_Toc199505579)

[1.3 Zielgruppe 3](#_Toc199505580)

[2. Projektbeschreibung 3](#_Toc199505581)

[2.1 Projektidee & Szenario 3](#_Toc199505582)

[2.2 Hardware 3](#_Toc199505583)

[2.3 Software 4](#_Toc199505584)

[2.3.1 IDE und verwendete Libraries 4](#_Toc199505585)

[2.4 Kommunikation & Datenformate 5](#_Toc199505586)

[2.5 Visualisierung 5](#_Toc199505587)

[2.6 Codeerläuterungen 5](#_Toc199505588)

[3. Schluss 6](#_Toc199505589)

[3.1 Zusammenfassung des Projekts 6](#_Toc199505590)

[3.2 Mögliche Erweiterungsideen: 6](#_Toc199505591)

[3.3 Reflexion (Was lief gut? Was war schwierig?) 6](#_Toc199505592)

[4. Kommunikationsplan 7](#_Toc199505593)

[5. Literatur- & Quellenverzeichnis 8](#_Toc199505594)

[6. Anhang 8](#_Toc199505595)

**Abbildungsverzeichnis**

[Abbildung 1 0](#_Toc199160986)

https://www.amazon.de/HiLetgo-Temperatursensor-Interface-Precision-Temperature/dp/B082M65D8P

[Abbildung 2 0](#_Toc199160987)

https://funduinoshop.com/elektronische-module/sonstige/mikrocontroller/wemos-d1-mini-v4.0-esp8266-mikrocontroller-lolin-d1-mini-v4.0-kompatibel

**Abkürzungsverzeichnis**

# Einleitung

## Motivation/Problemstellung

Warum ist IoT wichtig? Warum der Wemos D1 Mini?

## Zielgruppe

Schüler:innen, Lehrer, mit der Erweiterung jeder der im Office arbeitet

# Projektbeschreibung

## Projektidee & Ziele

Ziel unseres Projekts ist es, Temperaturdaten in zwei unterschiedlichen Dartstellungsformen Bereitzustellen und zu speichern:

* Kurzansicht: die letzten 50 Sekunden, in 5 Sekunden abständen
* Langansicht: die letzten 5 Tage, in 1 Stunde abständen

Die Daten werden einen HTTPS-Server zur Verfügung gestellt, der direkt auf dem Wemos D1 Mini gehostet wird.

Die Kurzansicht präsentiert die Temperaturwerte in tabellarischer, kompakter Form, während die Langansicht hingegen die Daten grafisch darstellt, wodurch sie so eine übersichtliche und verständliche Auswertung der Temperaturentwicklung über einen längeren Zeitraum ermöglicht.

Der Zugriff der Daten erfolgt plattformunabhängig über das lokale Netzwerk per Web-Interface.

Zusätzlich ist das Projekt robust mit der Watchdog-Überwachung und skalierbar durch die gewählte Softwarearichitektur mit dem modularen Aufbau.

*Blockschaltbild 1*

## Technische Spezifikationen

* **Mikrocontroller:** ESP8266 (Wemos D1 Mini)
* **Temperatursensor:** TMP75 (I2C-Interface)
* **Programmiersprache:** MicroPython
* **Webserver:** Integrierter HTTP-Server
* **Frontend:** React-basierte Single-Page-Application
* **Datenformat:** JSON und XML für API-Endpoints

## Hardwarearchitektur

Uns wurde als zentrale Steuereinheit der Wemos D1 mini zur Verfügung gestellt. Er ist ein kompakter Mikrocontroller auf Basis des ESP8266EX-Chips mit integriertem WLAN. Das Board verfügt über 4MB Flash-Speicher sowie einen 32 Bit-Prozessor und bietet eine Vielzahl an I/O Pins für externe Komponenten.

Im unserem Projekt nutzen wir die Serial-Data- (SDA) und Serial-Clock-Leitung (SCL) des I2C Busses, um den CJMCU-75 Temperatursensor auszulesen und die aktuelle Umgebungstemperatur zu erfassen.

Als Beispiels Aktor dient die OnBoard-LED des Wemos Dd1 Mini, welche über den GPIO-Pin 2 angesteuert werden kann.

## Software

### IDE und verwendete Libraries

Für die Entwicklung des Backend unseres Projekts wurde die Thonny IDE verwendet, da sie sich besonders gut für MicroPython eignet.  
Visual Studio Code zum Einsatz kam für das Frontend zum Einsatz, um die JSX-Dateien und die Gestaltung der Weboberfläche komfortabel zu gestalten.

Das Backend des Projektes nutzt ausschließlich Micro-Python-kompatible Bibliotheken:

* ***„network, socket, time, gc“*** – System und Netzwerkfunktionen
* eigenentwickelte Module: ***„SoftwareWatchdog, LEDController, DataManager, WebHandler, WiFiManager, ESP8266WebServer“***

Zur Visualisierung der Daten haben wir uns entschieden, die Bibliothek ***„react-google-charts“*** zu nutzen. Mit ihr realisieren wir einfache und flexible Diagramme, mit denen der Client interagieren kann.

### Softwarearchitektur

Die Entwicklung des Programms erfolgte entlang eines klaren objektorientierten Konzepts unter Beachtung der SOLID-Prinzipien:

* Jede Klasse hat eine klar definierte Verantwortlichkeit **(SRP)**

TemperaturStation: Systemkoordination und Lifecycle-Management

DataManager: Datenerfassung und -verwaltung

WebHandler: HTTP-Request-Verarbeitung

WifiManger: Netzwerkverbindungsmanagement

* High-Layer-Module sind nicht abhängig von Low-Layer-Modulen **(DIP)**

TemperaturStation: agiert über die Interfaces

Die Konfigurationen werden zentral in der „Config“-Klasse verwaltet

Hardwarespezifische Funktionen sind gekapselt in separaten Modulen

* Das System ist leicht erweiterbar ohne Modifikationen **(OCP)**

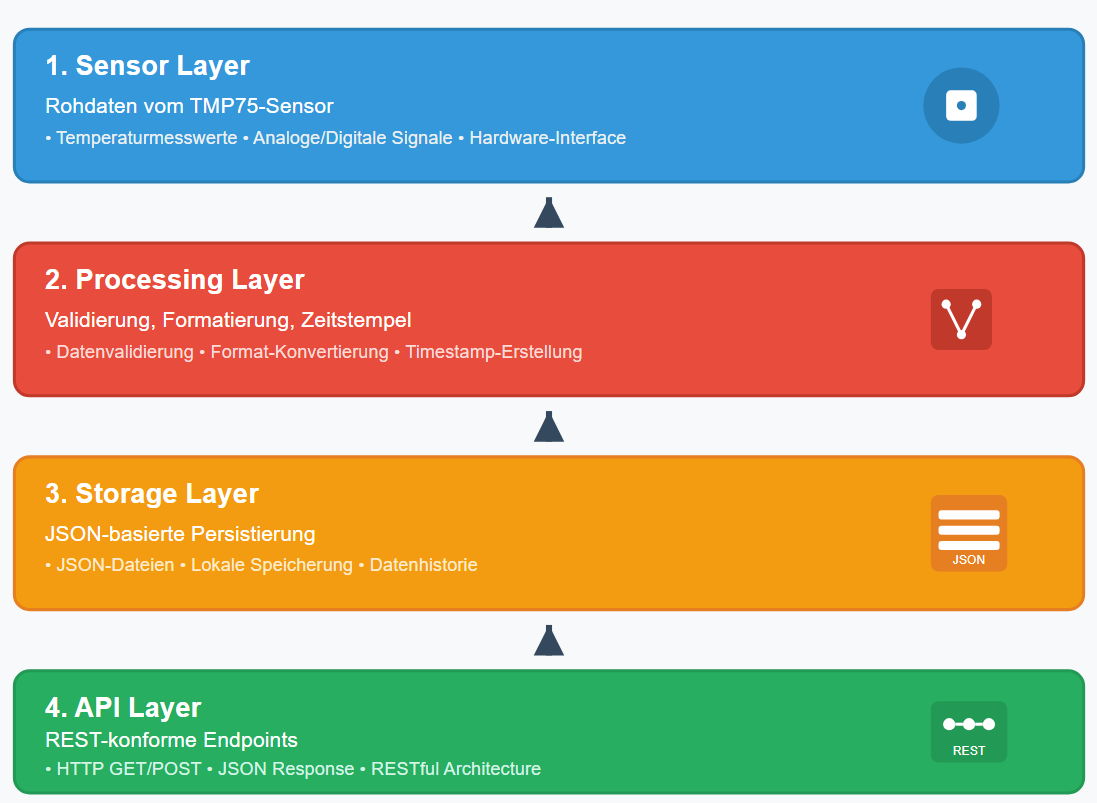
Web-API-Endpoints sind über WebHandler erweiterbar

Neue Sensoren können implementiert werden

### Klassenarchitektur

*Klassendiagramm 1*

### Datenarchitektur



Daten

### API-Design

RESTful Endpoints

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Method | Response | Zweck |
| / | GET | HTML | Web-Interface |
| /kurzzeitTemperatur\_json | GET | JSON | Aktuelle Messwerte |
| /langzeitTemperatur\_json | GET | JSON | Daten der letzen 5 Tage |
| /kurzzeitTemperatur\_xml | GET | XML | Aktuelle Messwerte nur als XML |

Response-Format

*// JSON Response*

[

["2024-01-15 14:30:25", 23.5],

["2024-01-15 14:30:30", 23.7]

]

*// XML Response*

<temperaturen>

<eintrag>

<zeitpunkt>2024-01-15 14:30:25</zeitpunkt>

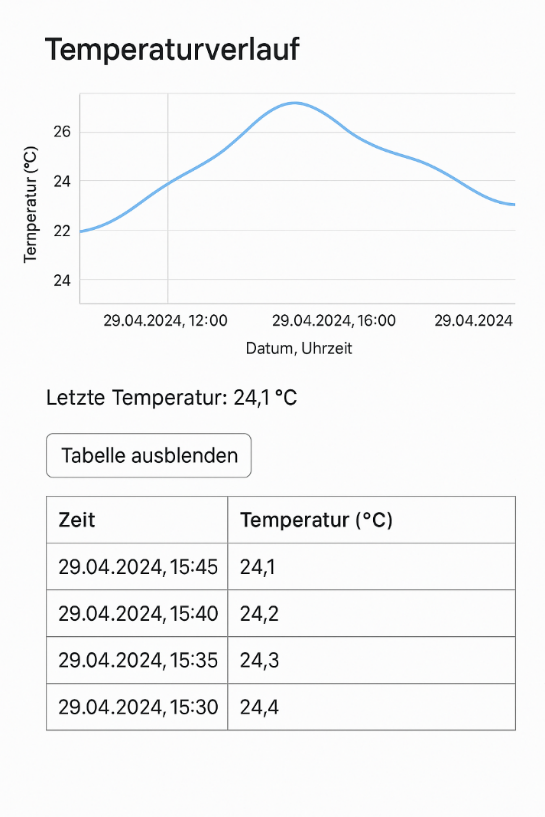
<wert>23.5</wert>

</eintrag>

</temperaturen>

### Codeerläuterungen (weiß noch nich genau)

## Visualisierung

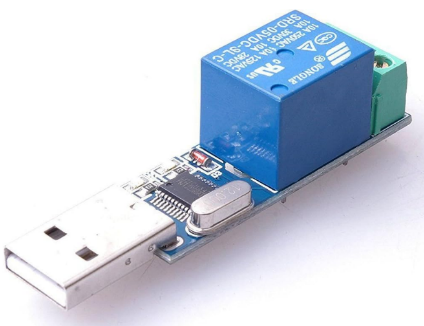
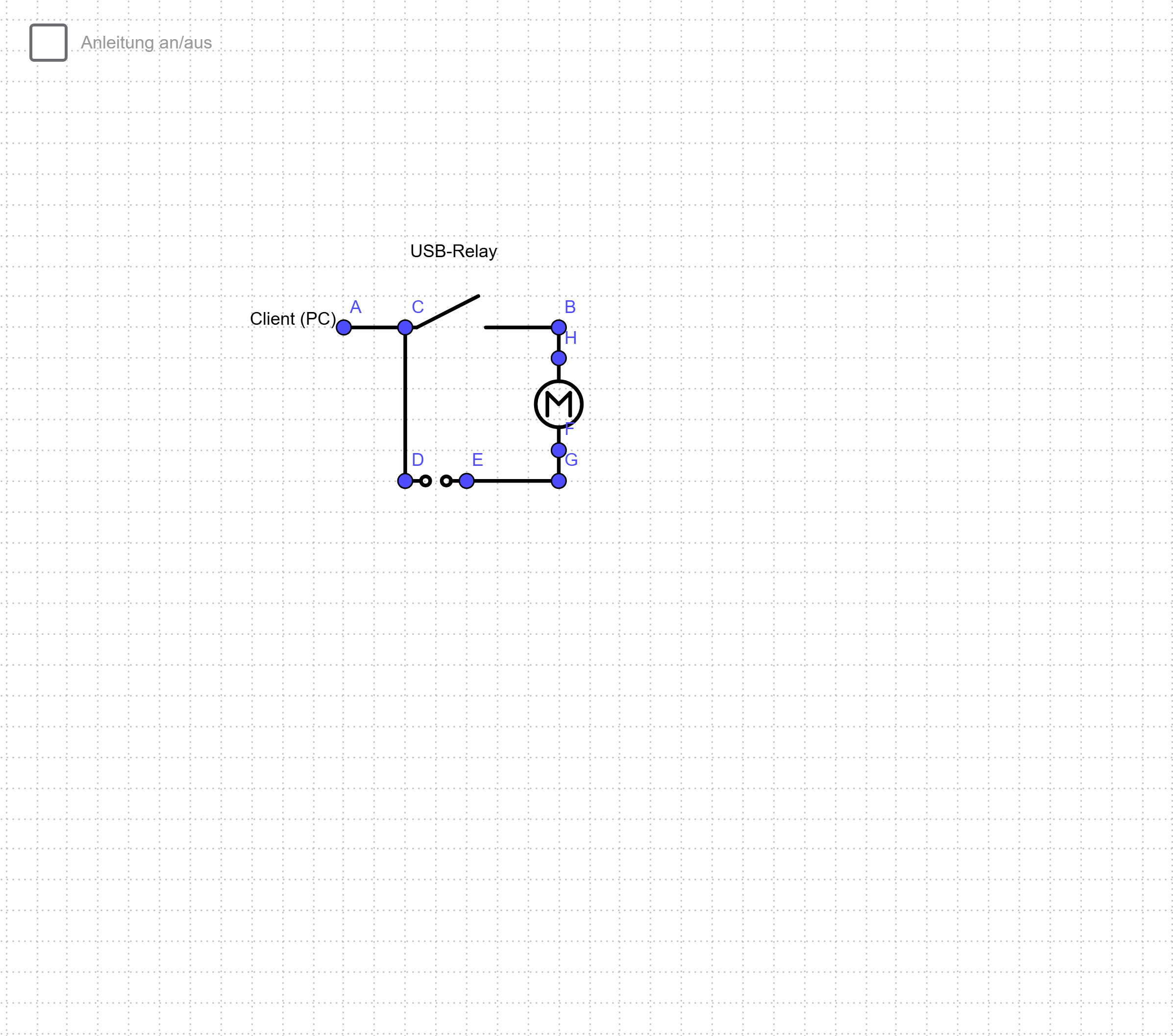
* Beschreibung der Webseite aus espfrontend
* Screenshots der Weboberfläche

# Schluss

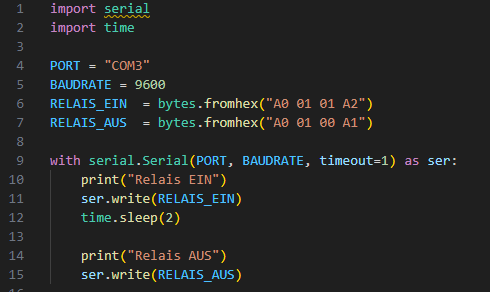
## Zusammenfassung des Projekts

## Mögliche Erweiterungsideen:

* Client-PC steuert über Temperatur-Abfrage (M) Ventilator. Beispielsweise im Sommer wenn es zu heiß wird, geht der Ventilator von alleine an am Arbeitsplatz



über eine serielle Schnittstelle, in Python:



* Datenbank für Langzeitspeicherung

## Reflexion (Was lief gut? Was war schwierig?)

# Kommunikationsplan

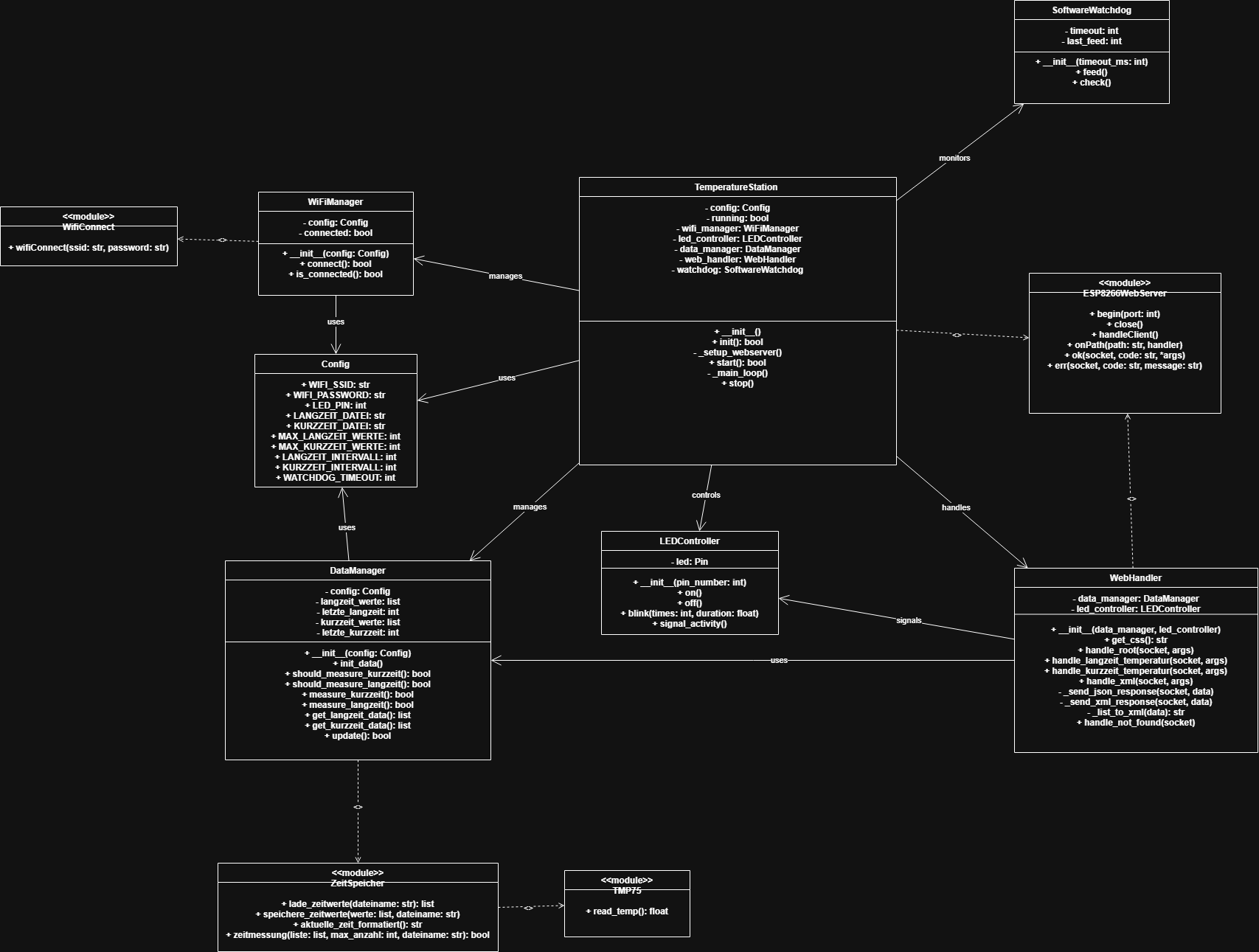
* Teammitglieder + E-Mail
* Aufgabenverteilung
* Genutzte Kommunikationskanäle

# Literatur- & Quellenverzeichnis

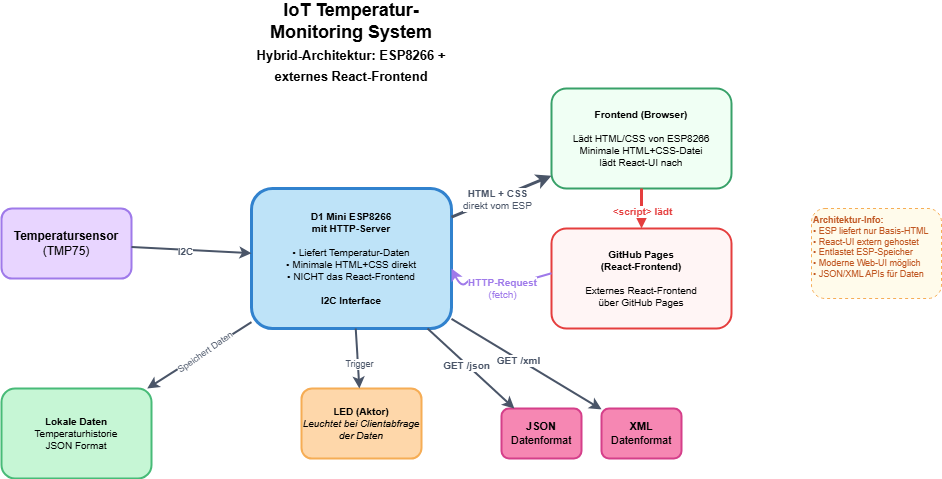
Datenblätter, Webseiten (z. B. ESP8266-Doku, Stack Overflow, GitHub-Links)

# Anhang

* Komplette Quellcodes (als Auszug oder Anhang)
* Datenblätter (z. B. PIR, HC-SR04)
* Schaltpläne
* Diagramme (z. B. Blockschaltbild)
* Optional: Link zum Projektdemo-Video



Klassendiagramm 1



Blockschaltbild 1