

Software Requirements Specification

für

SensorBear

Version 1.0

Vorbereitet von Damjan Petrovic

Inhaltsverzeichnis

1. EINLEITUNG	3
1.1 ZWECK	3
1.2 DOKUMENTKONVENTIONEN	3
1.3 ZIELGRUPPE UND LESEREIHENFOLGE	3
1.4 PRODUKTUMFANG	3
1.5 REFERENZEN	3
2. GESAMTE BESCHREIBUNG	3
2.1 PRODUKTPERSPEKTIVE.....	3
2.2 HAUPTFUNKTIONEN	4
2.3 BENUTZERKLASSEN UND EIGENSCHAFTEN	4
2.4 BETRIEBSUMGEBUNG	4
2.5 DESIGN- UND IMPLEMENTIERUNGS-CONSTRAINTS.....	4
2.6 BENUTZERDOKUMENTATION	4
2.7 ANNAHMEN UND ABHÄNGIGKEITEN	4
3. EXTERNE SCHNITTSTELLENANFORDERUNGEN	5
3.1 BENUTZERSCHNITTSTELLEN	5
3.2 HARDWARE-SCHNITTSTELLEN	5
3.3 SOFTWARE-SCHNITTSTELLEN	5
3.4 KOMMUNIKATIONSSCHNITTSTELLEN.....	5
4. SYSTEMFUNKTIONEN.....	5
4.1 SYSTEM FEATURE: DATENERFASSUNG	5
4.2 SYSTEM FEATURE: DATENVISUALISIERUNG.....	5
4.3 SYSTEM FEATURE: ADMINISTRATION	5
4.4 SYSTEM FEATURE: BENACHRICHTIGUNGEN	5
5. WEITERE NICHT-FUNKTIONALE ANFORDERUNGEN.....	6
5.1 LEISTUNGSANFORDERUNGEN.....	6
5.2 SICHERHEITSANFORDERUNGEN	6
5.3 SICHERHEITSANFORDERUNGEN (SAFETY)	6
5.4 QUALITÄTSATTRIBUTE.....	6
5.5 GESCHÄFTSREGELN	6
6. WEITERE ANFORDERUNGEN.....	6
ANHANG A: GLOSSAR.....	6

1. Einleitung

1.1 Zweck

Dieses Dokument beschreibt die funktionalen und nicht-funktionalen Anforderungen des sensorbasierten Monitoring-Systems. Ziel ist es, eine präzise Grundlage für Implementierung, Validierung und Wartung des Systems bereitzustellen. Es richtet sich an Entwickler, Tester und Projektbeteiligte.

1.2 Dokumentkonventionen

Dieses Dokument folgt der IEEE 830-Struktur für Software Requirements Specifications. Jede Anforderung erhält eine eindeutige Kennung (z. B. REQ-1).

1.3 Zielgruppe und Lesereihenfolge

Entwickler: zur Implementierung und Schnittstellendefinition

Projektleitung / Lehrer: zur Fortschrittskontrolle und Validierung

Tester: zur Erstellung von Testfällen basierend auf funktionalen Anforderungen

Empfohlene Lesereihenfolge: Kapitel 1–2 für Überblick, Kapitel 3–4 für Funktionalität, Kapitel 5 für Qualitätsanforderungen.

1.4 Produktumfang

Das System dient der kontinuierlichen Erfassung und Visualisierung von Umweltdaten (CO₂, Temperatur, Luftfeuchtigkeit, Lautstärke) in Innenräumen. Jeder ESP32-Sensor sendet Messwerte in festgelegten Intervallen über WLAN oder ein Mesh-Netzwerk an eine zentrale API-Datenbank. Ein Dashboard visualisiert Echtzeit- und Verlaufsdaten, zeigt Warnungen bei Grenzwertüberschreitungen und unterstützt Administratoren bei der Sensorverwaltung.

1.5 Referenzen

IEEE 830-1998 – IEEE Recommended Practice for Software Requirements Specifications

ESP32 Documentation (Espressif Systems)

Next.js / Material UI Developer Guide

PostgreSQL / TimescaleDB Documentation

2. Gesamte Beschreibung

2.1 Produktperspektive

Das Monitoring-System ist ein eigenständiges IoT-System mit drei Hauptkomponenten:

- Sensor Nodes (ESP32)
- Backend (API, PostgreSQL/TimescaleDB)
- Frontend (Next.js Dashboard / Swift / Electron)

Datenfluss: Sensor → API → Datenbank → Dashboard

2.2 Hauptfunktionen

Periodische Erfassung von Sensordaten
Speicherung in zentraler Datenbank
Visualisierung und Verlaufsgrafiken
Grenzwertwarnungen
Verwaltung von Sensoren und Räumen
Exportfunktionen (CSV/JSON)

2.3 Benutzerklassen und Eigenschaften

Admin: Konfiguriert Sensoren, Räume, Grenzwerte

User: Beobachtet Echtzeit- und Verlaufsdaten

2.4 Betriebsumgebung

Sensoren: ESP32 Microcontroller (Arduino / C++)

Server: Node.js API (Express), PostgreSQL / TimescaleDB

Frontend: Next.js Webapp (Material UI)

Kommunikation: WLAN / Mesh Netzwerk, HTTPS-Protokoll

2.5 Design- und Implementierungs-Constraints

Kommunikation ausschließlich über WLAN oder Mesh (kein Ethernet)
Energieversorgung über USB
Hardwarebudget auf Prototyp-Niveau begrenzt
Datenübertragung ≤ 10 s Intervalle
Verwendung von Open-Source-Technologien

2.6 Benutzerdokumentation

Online-Benutzerhandbuch (PDF)
Setup-Anleitung für Sensoren
Administrator-Handbuch

2.7 Annahmen und Abhängigkeiten

Stabile WLAN-Verbindung erforderlich
ESP32-Module verfügen über gültige Firmware
Server ist dauerhaft erreichbar
Browser-kompatibel mit Chrome / Safari / Edge

3. Externe Schnittstellenanforderungen

3.1 Benutzerschnittstellen

Dashboard mit responsivem Design, Anzeige von Live-Daten, Diagrammen und Warnmeldungen. Funktionen: Sensorverwaltung, Export, Benachrichtigungen.

3.2 Hardware-Schnittstellen

ESP32 Sensoren mit CO₂-, Temperatur-, Luftfeuchtigkeits- und Lautstärkesensoren. Kommunikation über I²C, UART oder analoge Pins.

3.3 Software-Schnittstellen

API (REST/JSON) ↔ Frontend

API ↔ PostgreSQL / TimescaleDB

3.4 Kommunikationsschnittstellen

HTTPS / WLAN / Mesh Netzwerk

Datenformat: JSON

Verschlüsselung: TLS 1.3

4. Systemfunktionen

4.1 System Feature: Datenerfassung

REQ-1: System misst alle 10 Sekunden*

REQ-2: Sensoren übertragen Messwerte

REQ-3: Mesh heilt sich selbst bei Ausfall von Sensor

4.2 System Feature: Datenvisualisierung

REQ-4: Dashboard zeigt Echtzeit- und Verlaufsdaten

REQ-5: Warnungen bei CO₂ > 1000 ppm* oder Lautstärke > 70 dB*

REQ-6: Benutzer können Zeiträume auswählen

4.3 System Feature: Administration

REQ-7: Admins verwalten Sensoren und Räume

REQ-8: Grenzwerte und Intervalle anpassbar

REQ-9: Datenexport als CSV/JSON

4.4 System Feature: Benachrichtigungen

REQ-10: Push- oder E-Mail-Benachrichtigungen bei Grenzwertüberschreitung

*Diese Standardwerte können vom Benutzer angepasst werden

5. Weitere nicht-funktionale Anforderungen

5.1 Leistungsanforderungen:

Datenübertragung ≤ 10 s, Dashboard ≤ 2 s, max. 50 Sensoren gleichzeitig.

5.2 Sicherheitsanforderungen:

Authentifizierung, HTTPS

5.3 Sicherheitsanforderungen (Safety):

Keine physischen Schäden, sichere Fehlermeldungen.

5.4 Qualitätsattribute:

Zuverlässigkeit ≥ 95 %, Wartbarkeit (GitHub), Benutzerfreundlichkeit, Portabilität.

5.5 Geschäftsregeln:

Nur Admins dürfen Sensoren ändern.

6. Weitere Anforderungen

Wöchentliche Backups auf GitHub und OneDrive, versionierte Dokumentation.

Zukunft: VOC- und Feinstaubsensoren, RFID-Login, Statistikfunktionen.

Anhang A: Glossar

ESP32 – Mikrocontroller mit integrierten WLAN und Bluetooth Fähigkeiten

ppm – Parts per Million (Maßeinheit CO₂)

Mesh-Netzwerk – dezentrale Netzwerktopologie

API – Schnittstelle zwischen Sensoren, Datenbank und Frontend