Library
Capacity
Measurement
System

LibCap

L. Benner, A. Buss, P. Lange, A. Madhour

Übersicht

1. Businessmodell
2. „Showcase“
3. Methodik
4. Grundlagen
5. Umsetzung
6. Fazit

1. Businessmodell

Why?

- Kein **Detailüberblick** über **Auslastung** von Bibliotheken
- **Reservierungssysteme** nur generell nicht auf **Sitzplatzebene**

What?

- Ermöglichung eines Überblicks über jeden Sitzplatz
- **Kostengünstige On-Premise** Solution

https://www.youtube.com/watch?v=ml8fKBpgCN8

3. Methodik

Theoretische Grundlagen

Hardware:

- ESP32-Microcontroller mit C++
- Sensoren

Middleware:

- Python
- MQTT mit Mosquitto
- Postgres-Datenbank

Web:

- Flask
- Bootstrap

Virtualization:

- Docker

3. Methodik

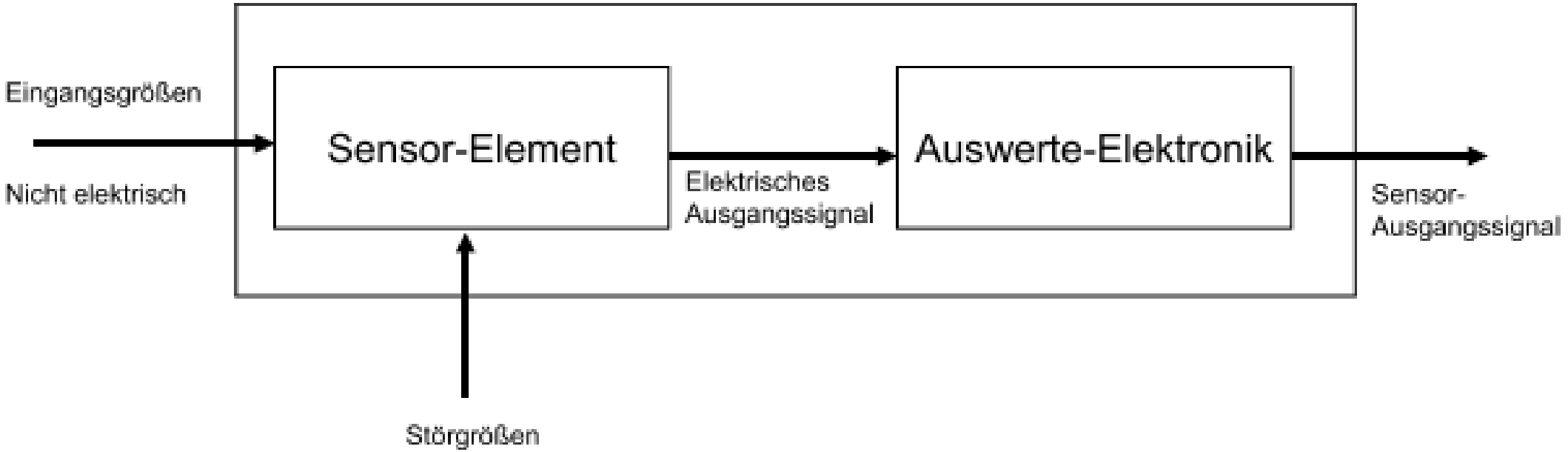
Vorgehensweise

- Parallel
- Kanban Board über Github

Bereiche

- Sensortheorie (Alina)
- Hardware/Arduino (Lukas)
- Middleware (Ayman)
- Web/Frontend (Phillip)

Sensor-System

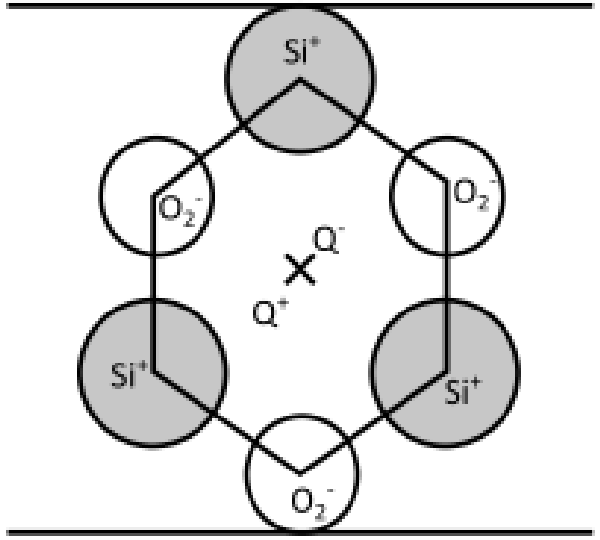


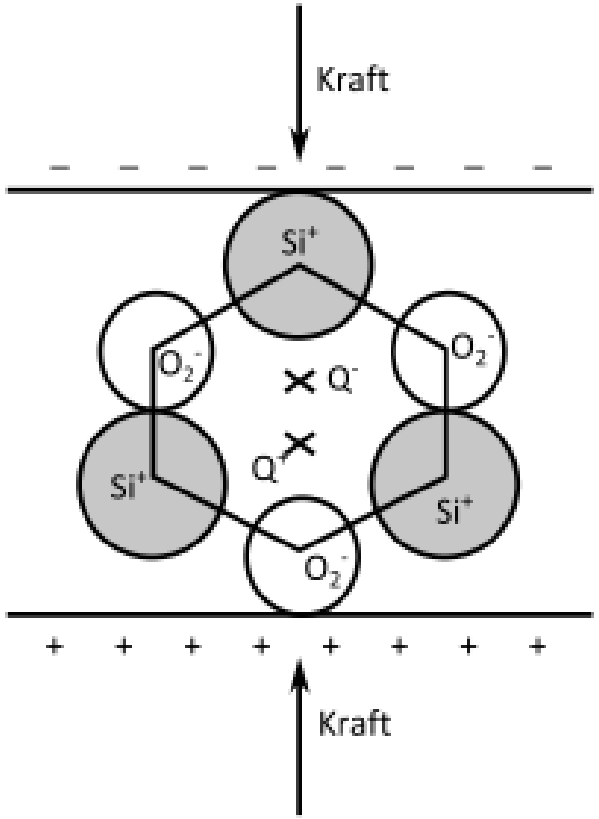
Ultraschall-Abstandssensor

- **Sendet Ultraschall-Wellen** im Ultraschallwellen-Bereich aus (20 kHz bis 10 GHz)
- **Empfängt** reflektierte **Ultraschallwellen**

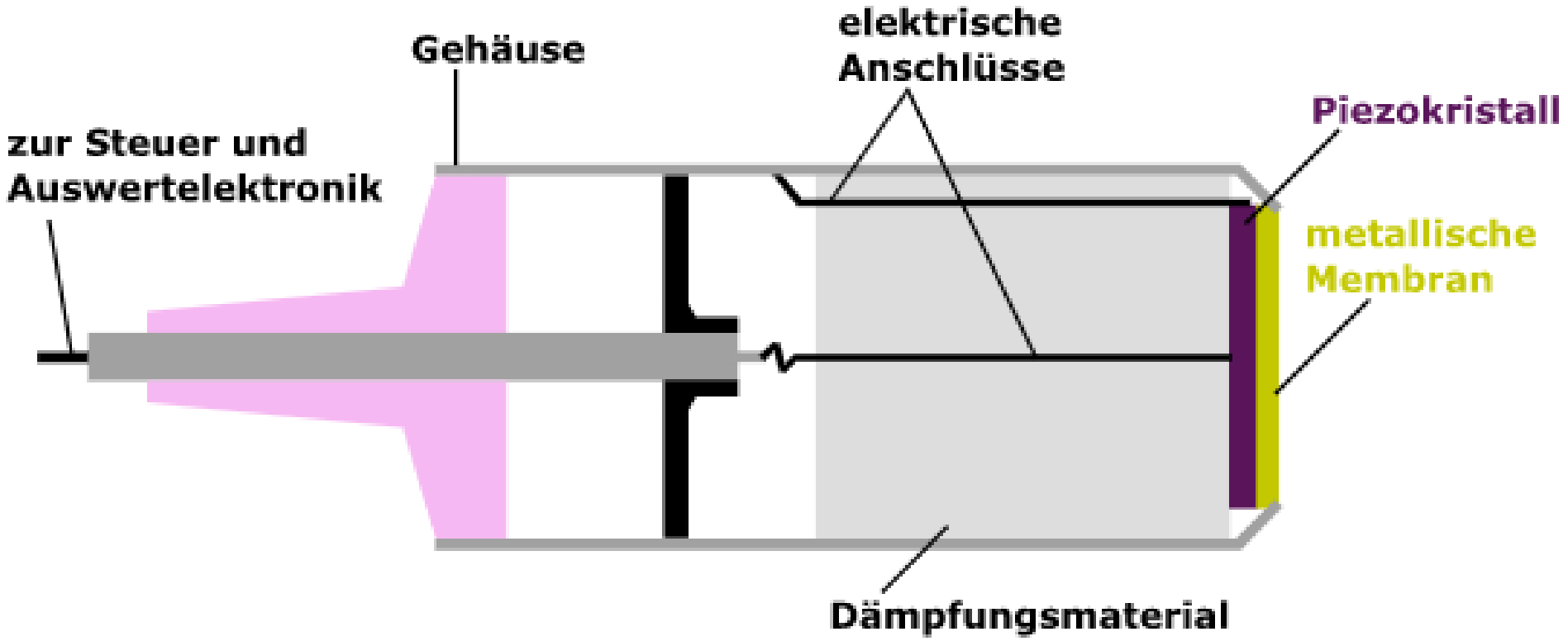
- Nutzt den **piezoelektrischen Effekt**
- Detektiert Objekte und kann Abstände zu diesen ermitteln

Piezoelektrische Effekt

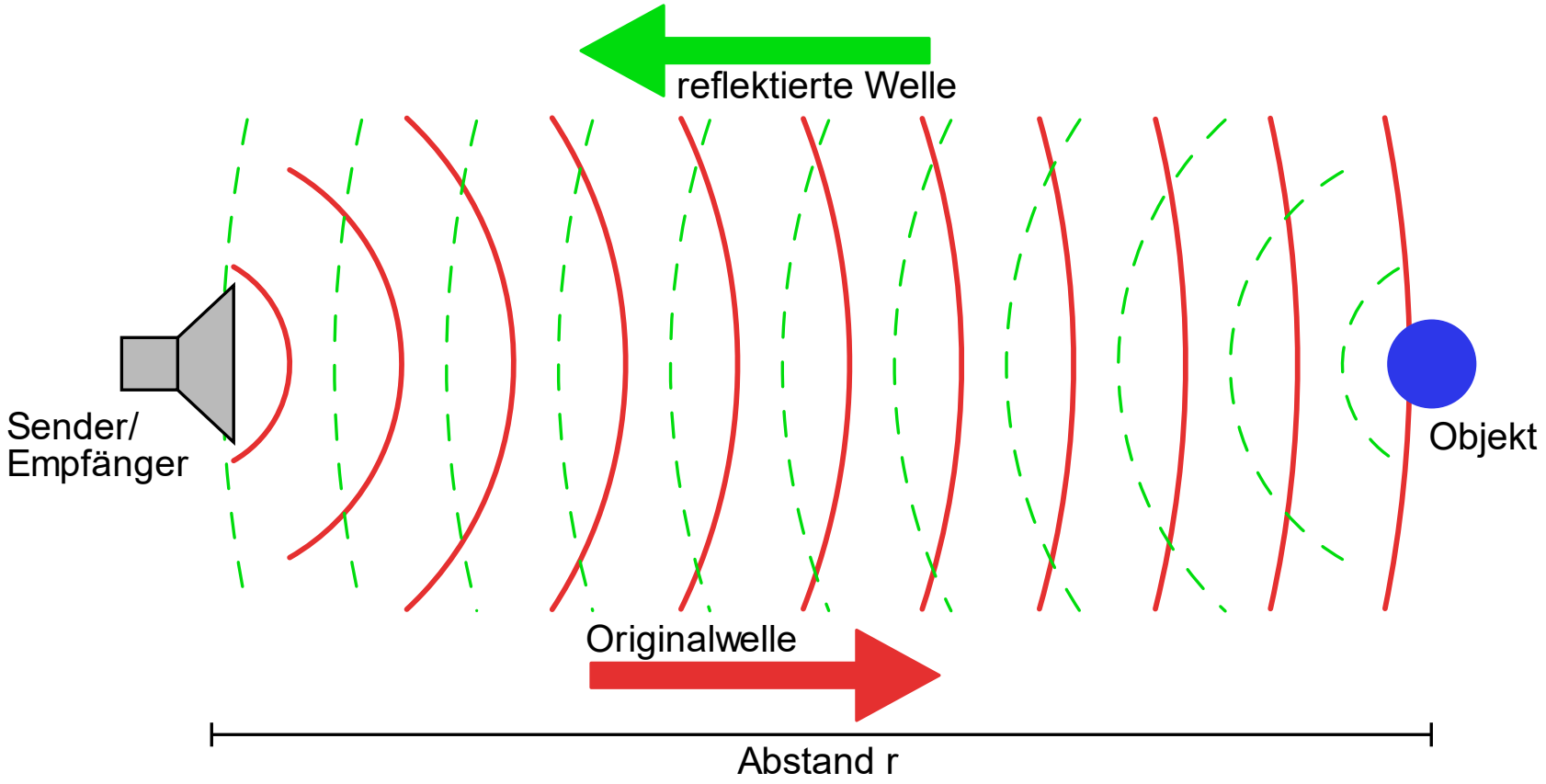




Aufbau eines Ultraschall-Sensors



Funktionsweise von Abstandssensoren

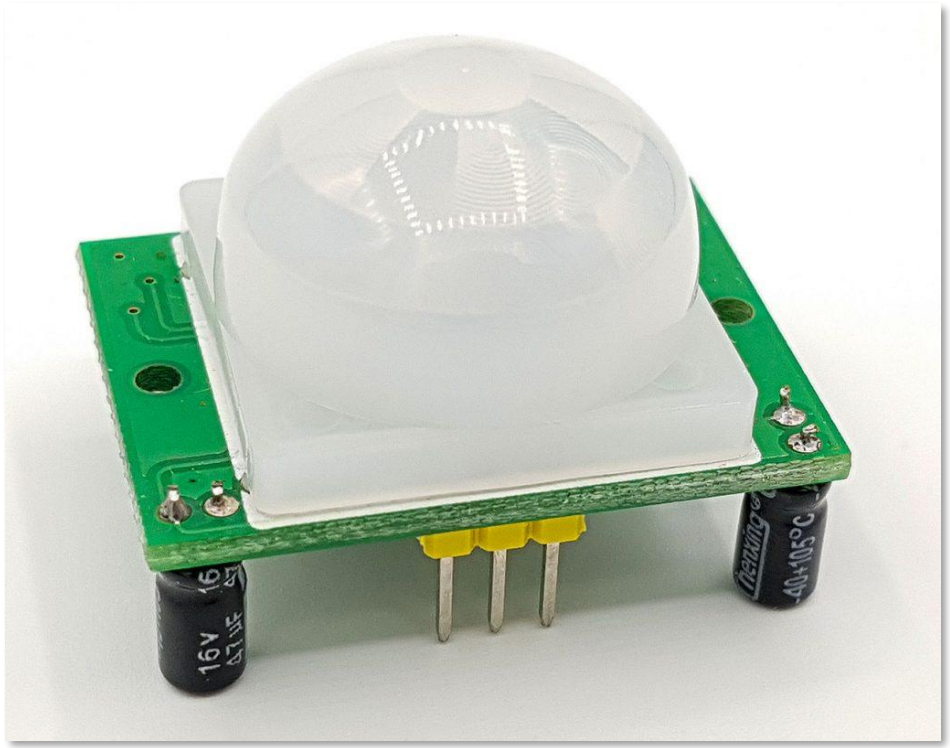


PIR-Bewegungssensor

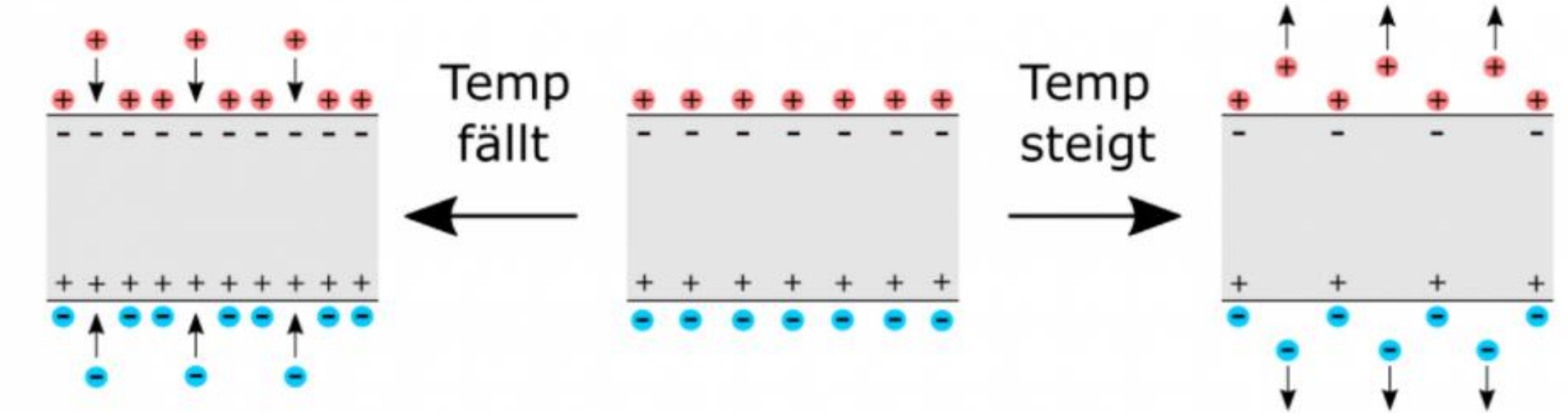
- **P**assiver **I**nfra**r**ot Sensor
- **P**yroelektrischer **I**nfra**r**ot Sensor

- Empfängt Wellen im Infrarotwellenbereich (300 GHz bis 400 THz)
- Sendet kein eigenes Infrarotes Licht aus

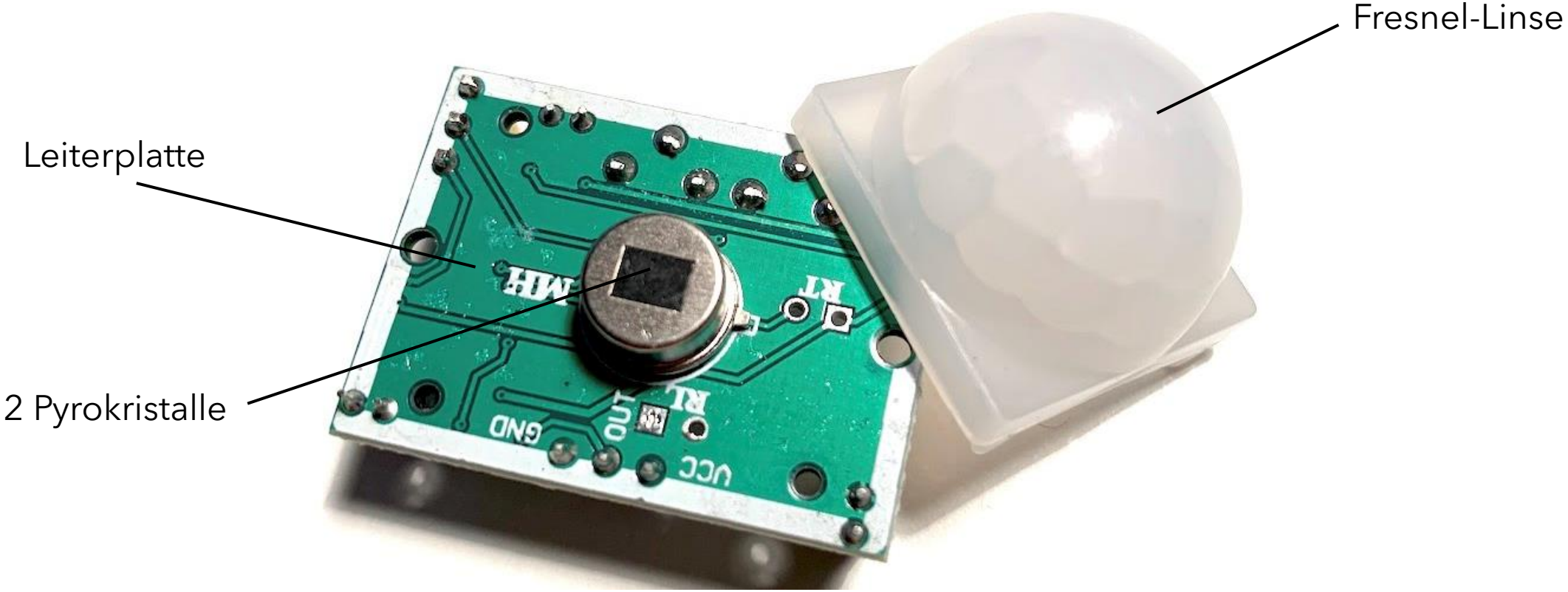
- Nutzt den pyroelektrischen Effekt
- Detektion von Temperaturänderungen



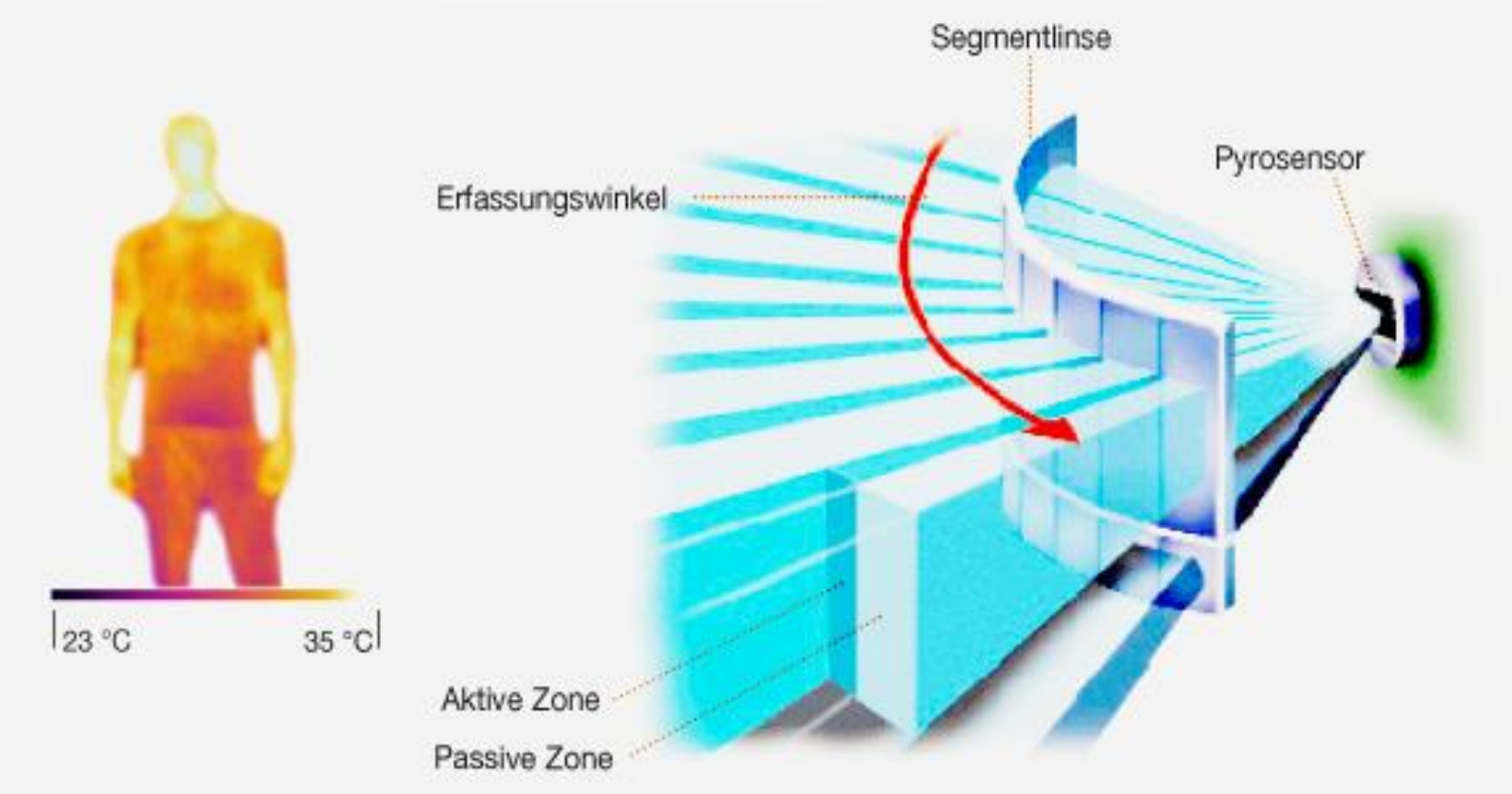
Pyroelektrische Effekt



Aufbau eines PIR-Sensor



Funktionsweise von PIR-Sensoren



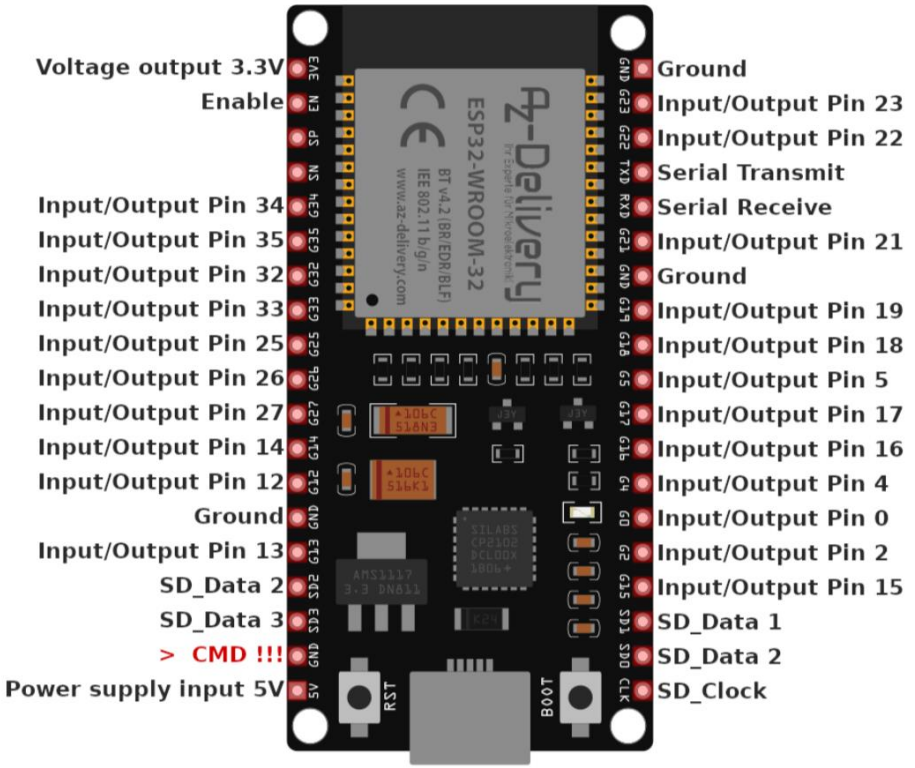
5. Umsetzung

ESP32-Controller

- Output: 3,3V oder 5V
- Input: 5V (über USB)

- 512kB RAM
- 240 MHz Taktfrequenz

- WiFi
- Bluetooth

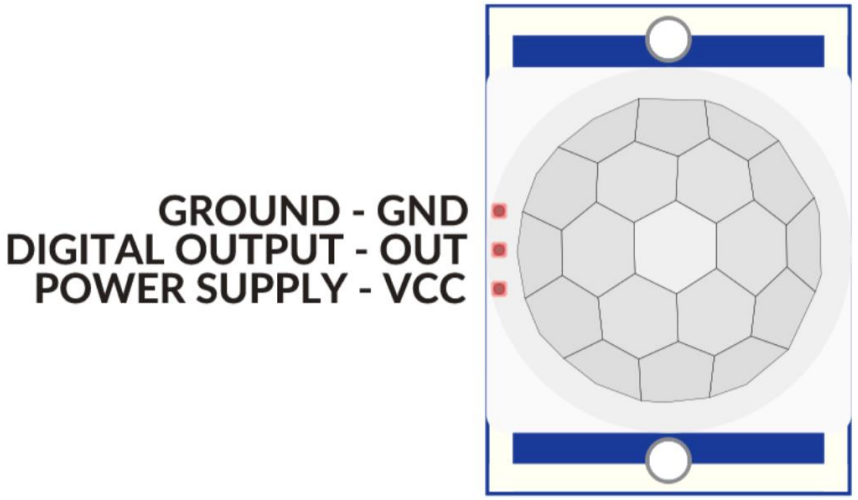


5. Umsetzung

Sensoren

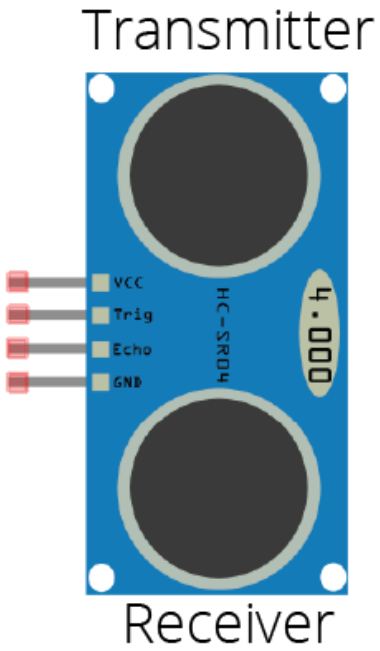
PIR Sensor

- In 3,3V Mode



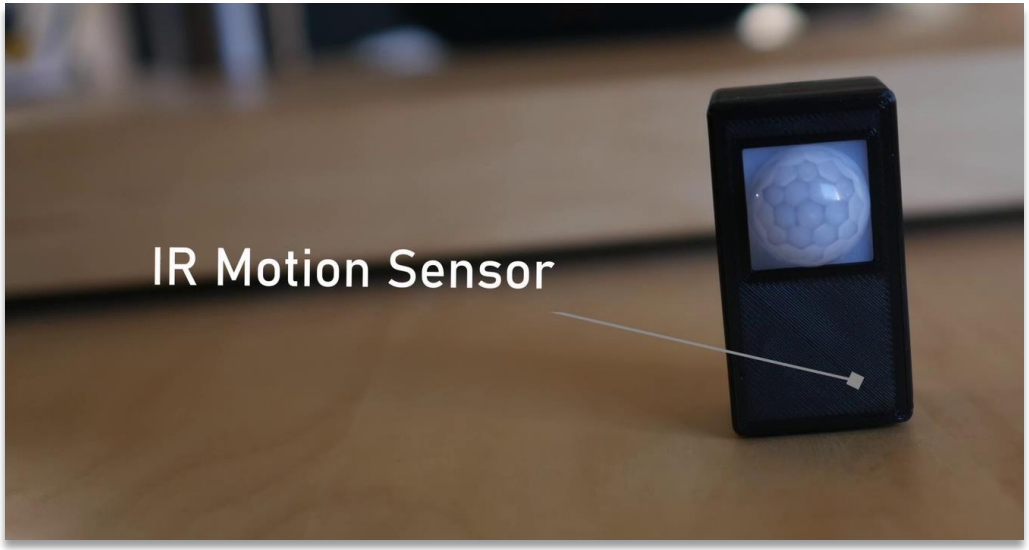
Ultraschall Abstandssensor

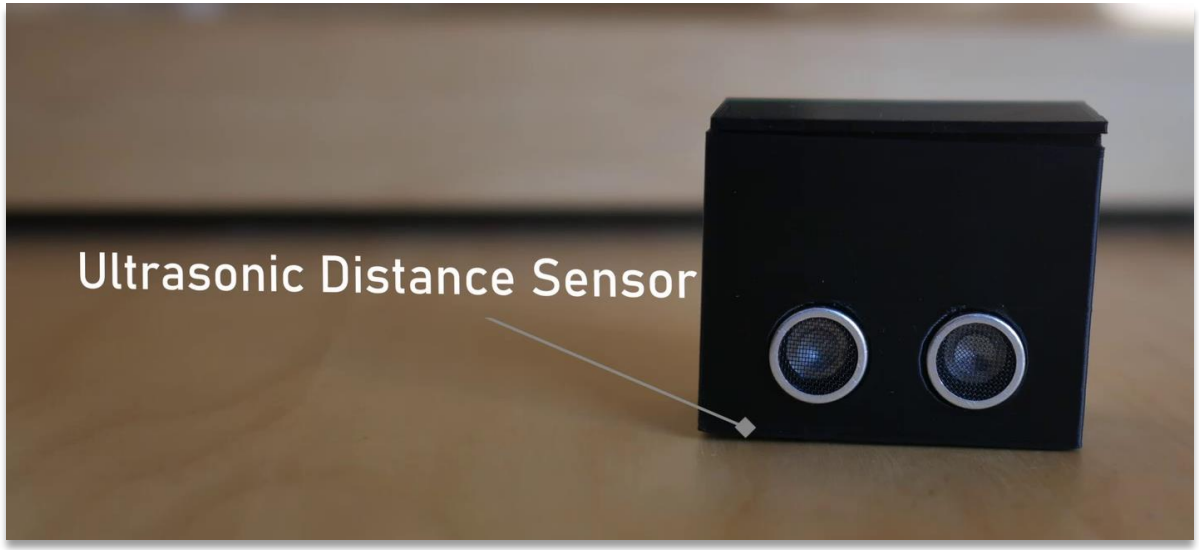
- VCC – 3V Power
- TRIG - Input
- ECHO – Output
- GND - Ground



5. Umsetzung

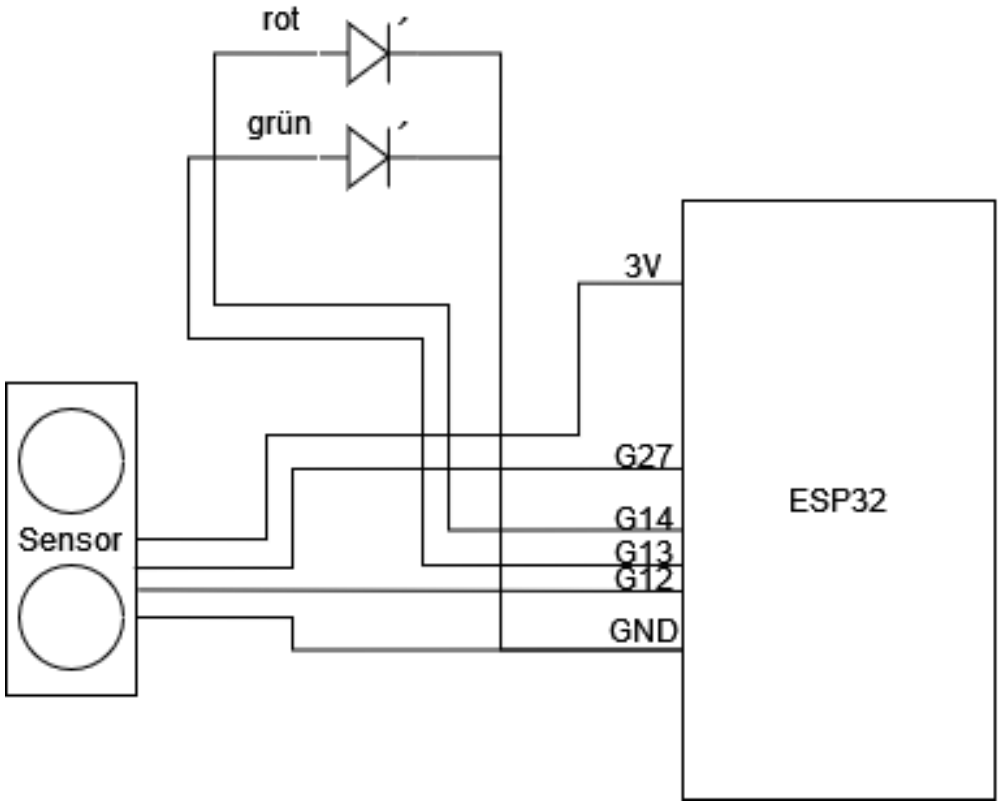
3D gedruckte Gehäuse





5. Umsetzung

Schaltung und Gehäuse



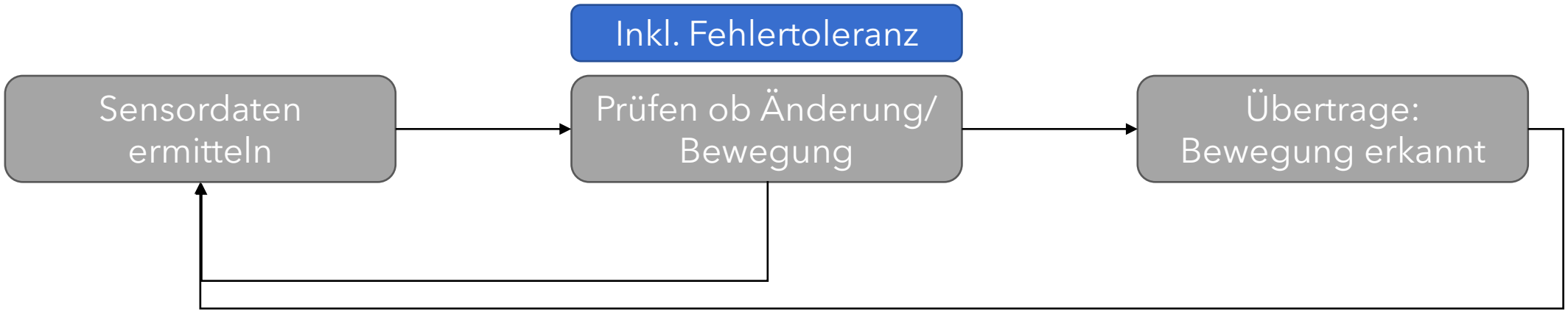
5. Umsetzung

Mikrocontroller-Programmierung

- C++ mit Arduino IDE
- Funktionen:
  - *EspMQTTClient()*
  - *onConnectionEstablished()*
  - setup()
  - loop()

- Hardcodierung der ID
- Minimale Anpassung je Sensortyp
- Für nachfolgende System nicht relevant

Mikrocontroller-Programmierung

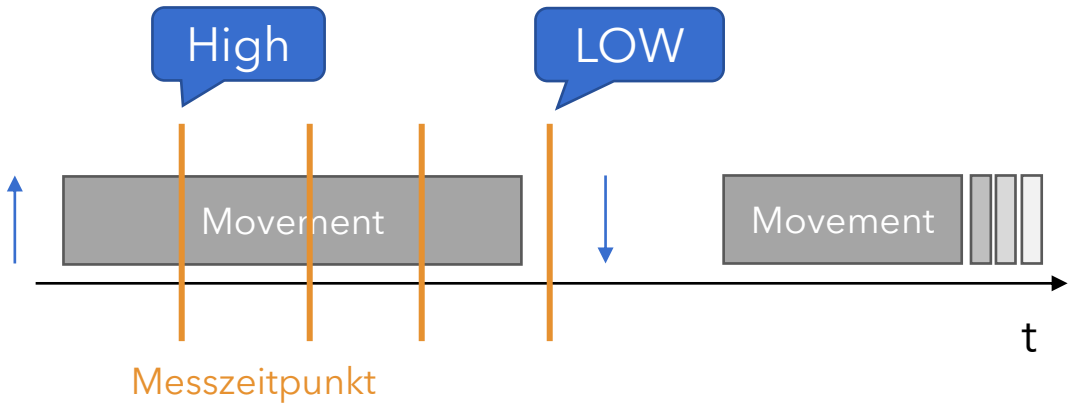


Datenübertragung

- Ziel: Netzwerkverkehr minimieren

→ nur bei **Flankenwechsel** senden

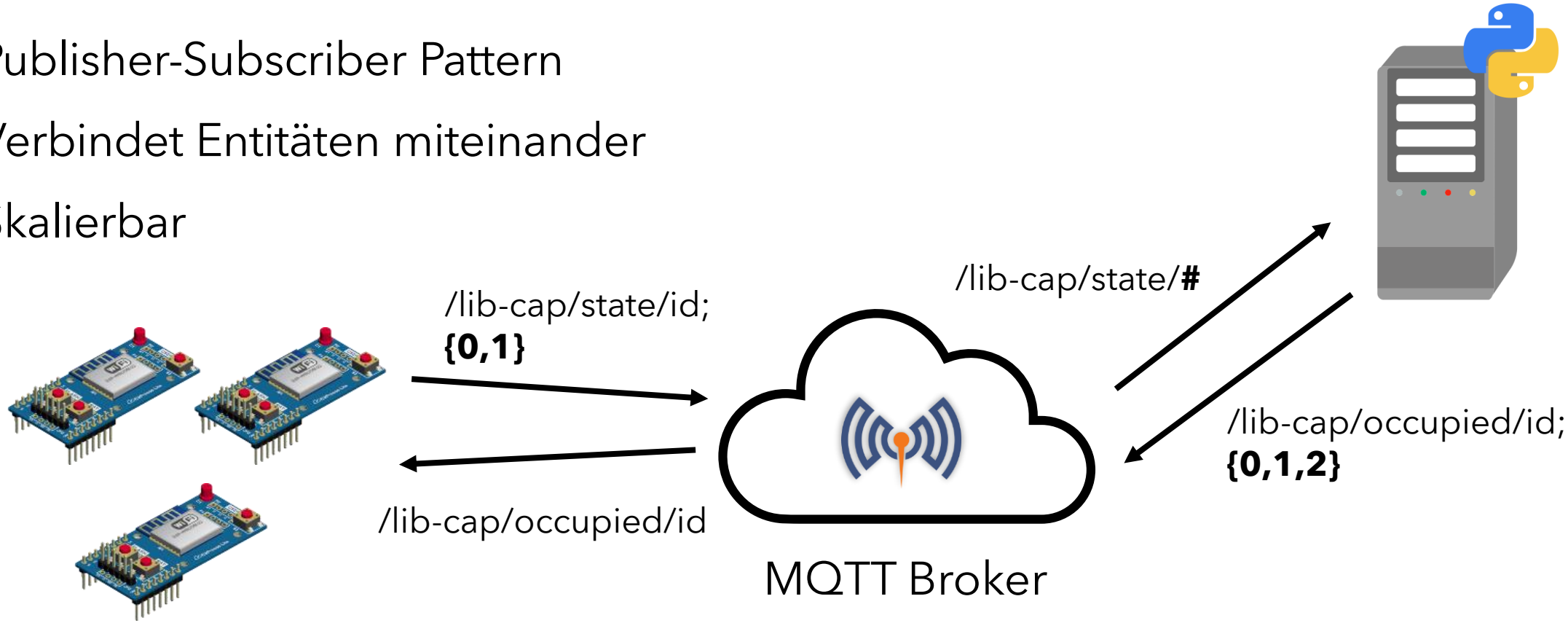
- Vorteil: lange Bewegungen werden nur einmal gesendet



5. Umsetzung

MQTT

- Publisher-Subscriber Pattern
- Verbindet Entitäten miteinander
- Skalierbar



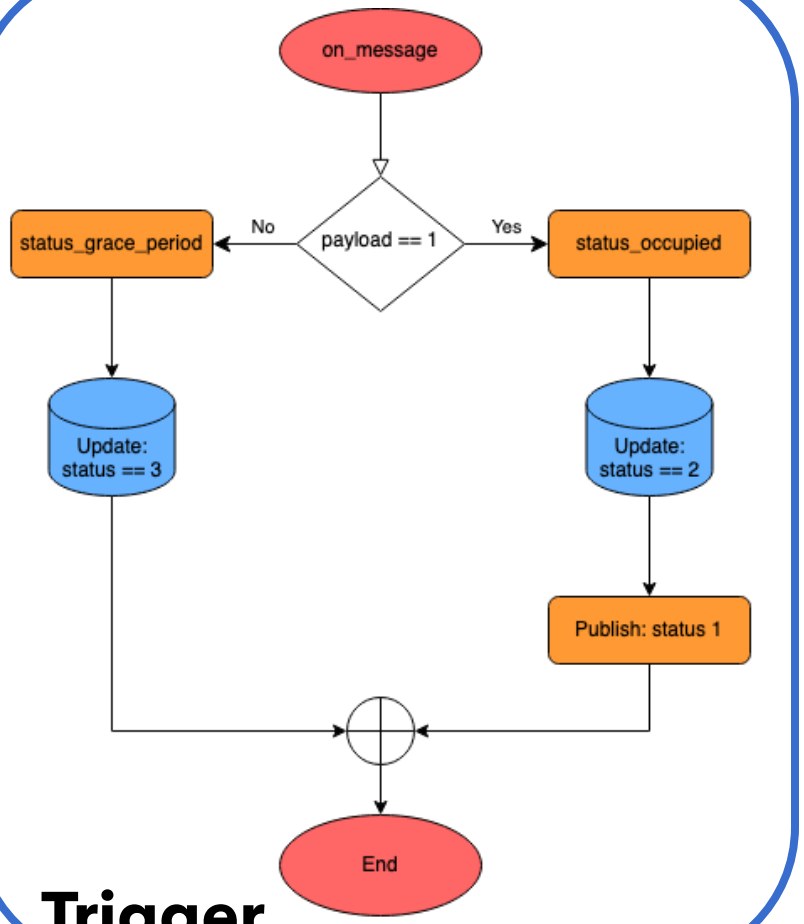
5. Umsetzung

Status

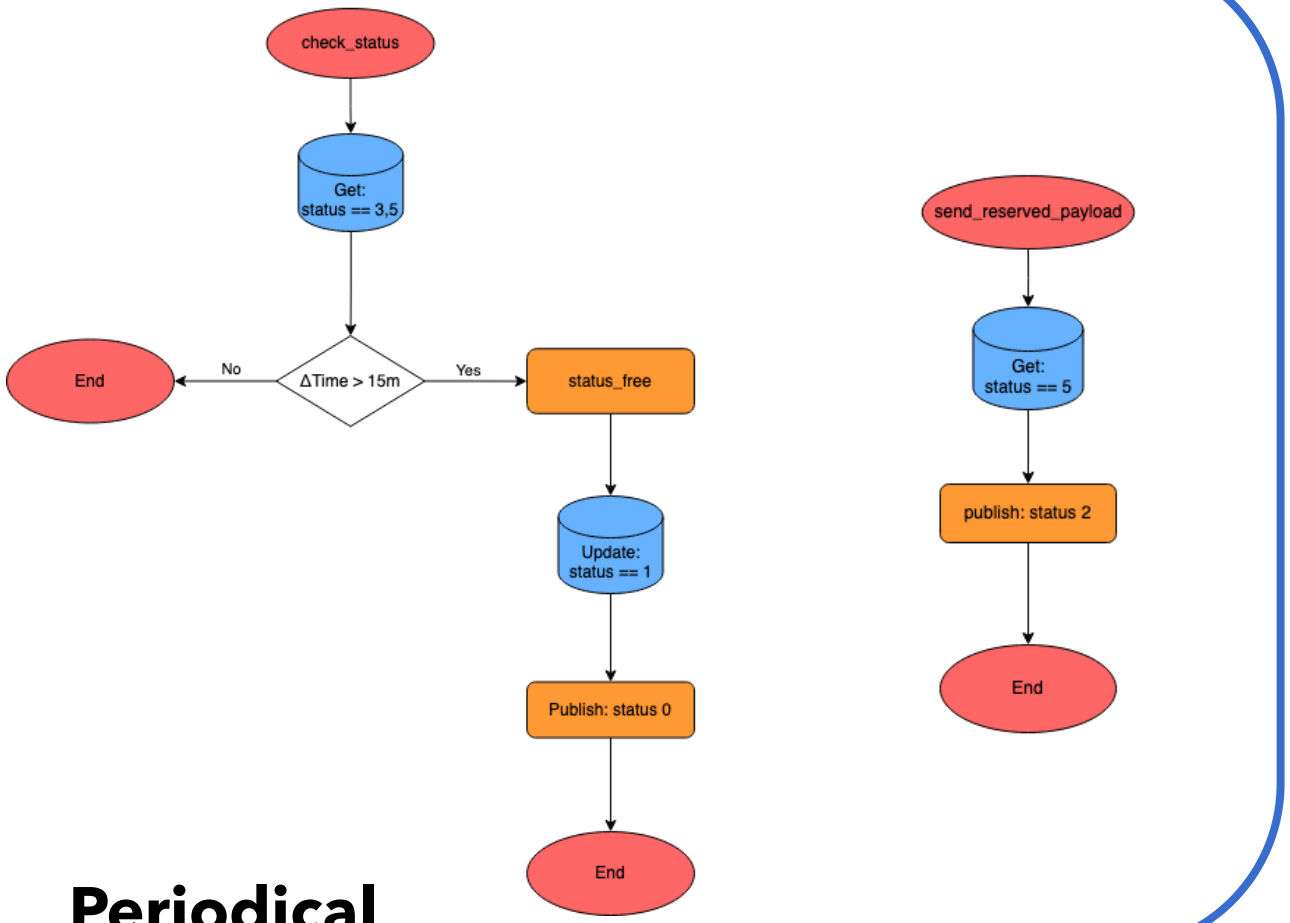
LED-Anzeige	MQTT Payload	Status in Datenbank	Beschreibung
	0	1	Free
	1	2	Occupied
	-	3	Grace Period
	-	4	In Maintenance
	2	5	Reserved

5. Umsetzung

Logik



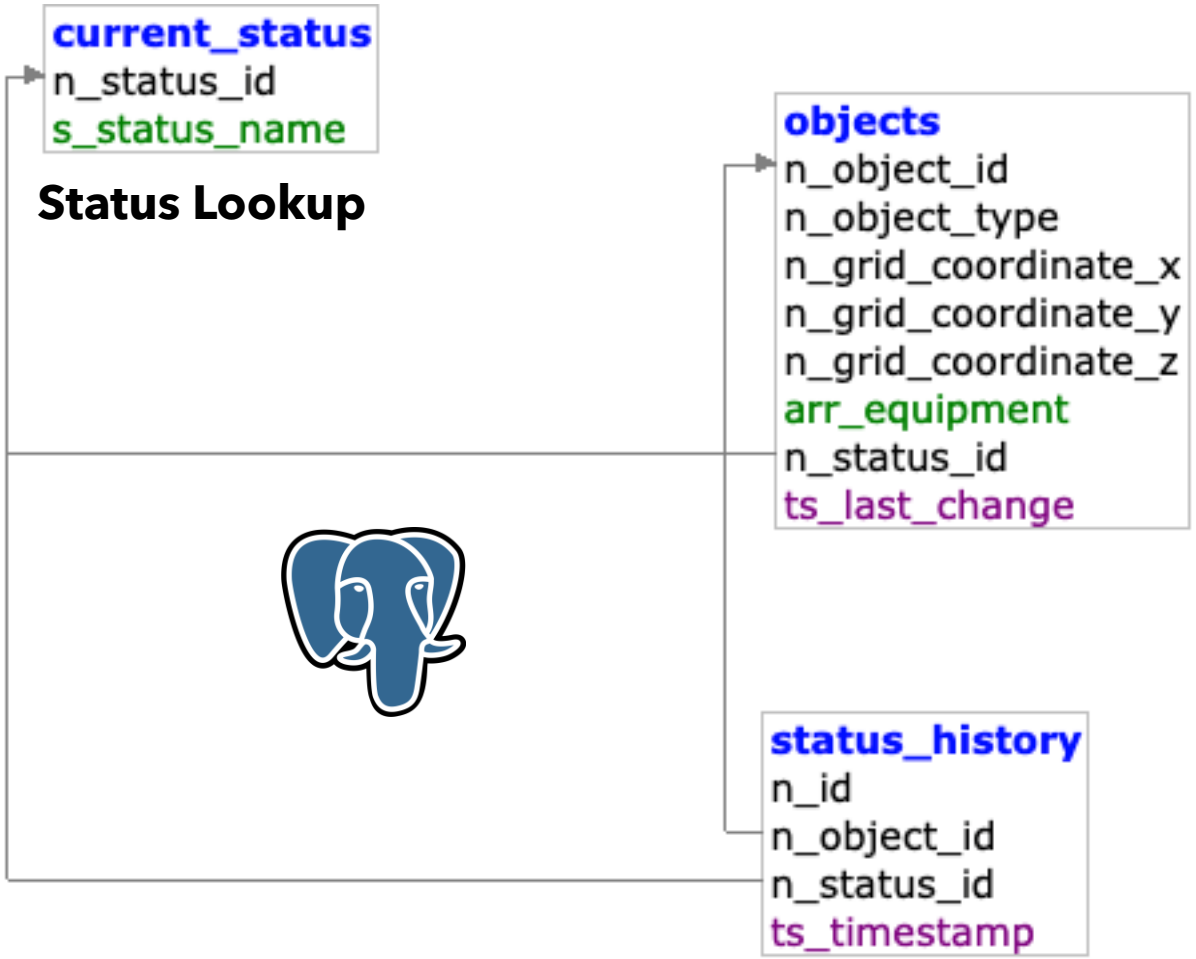
Trigger



Periodical

5. Umsetzung

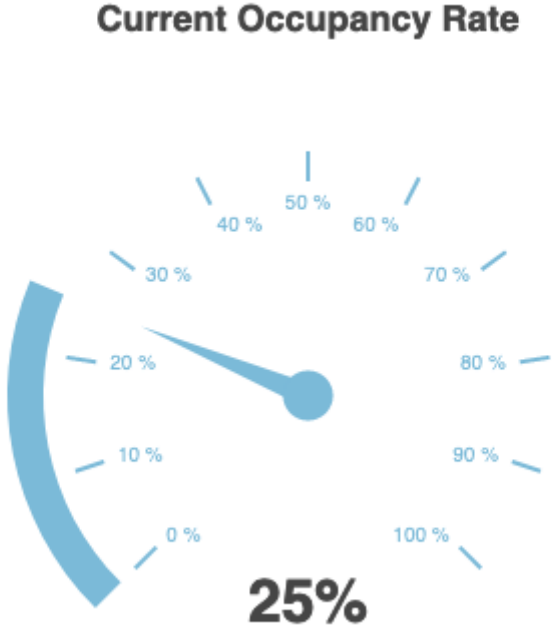
Datenbank

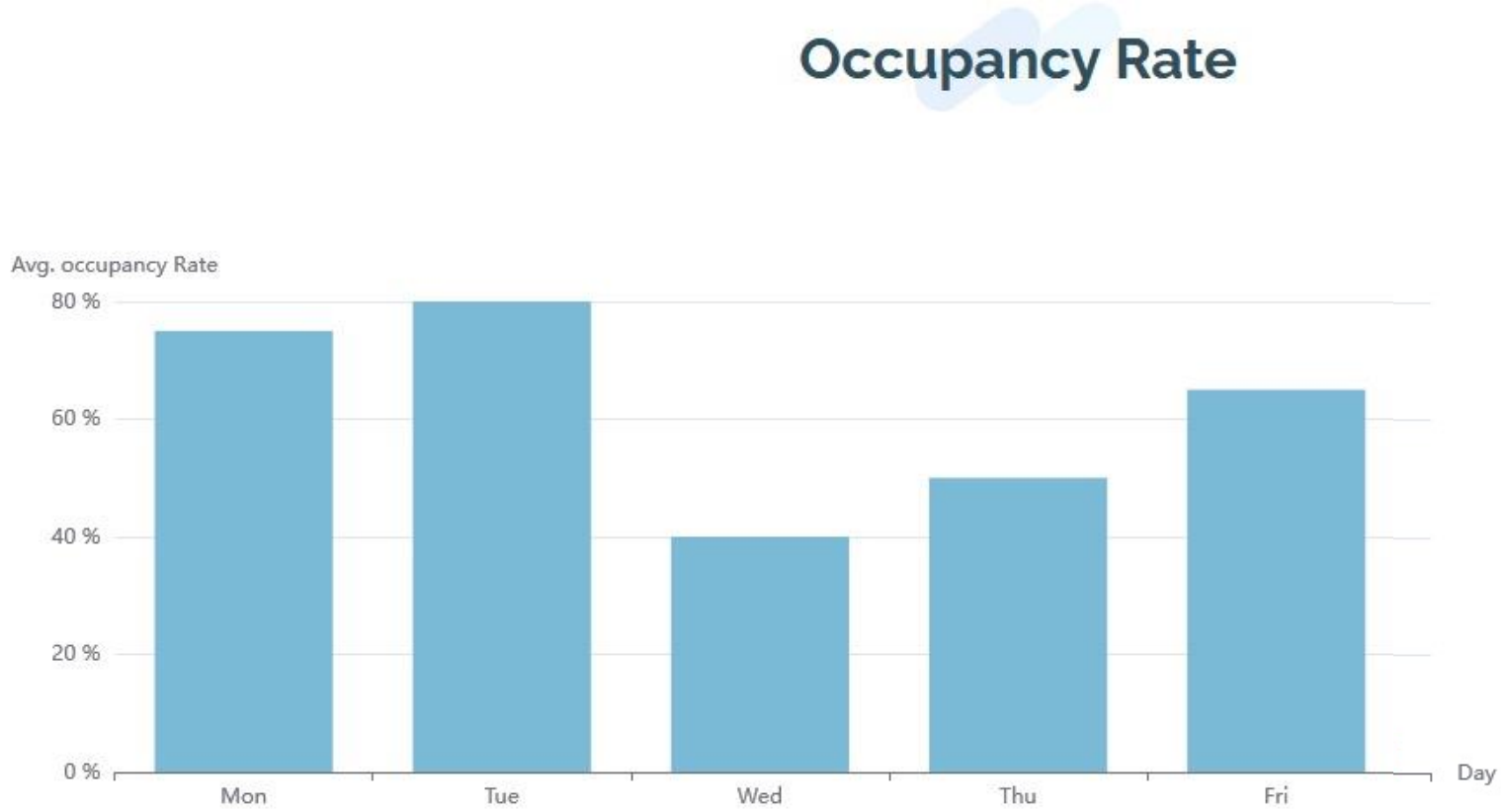


- Web⇔Backend Kommunikation über Datenbank
- „ Lageplan“ in Objects gespeichert
- Trigger bei Update → track changes

5. Umsetzung

Live-Monitoring



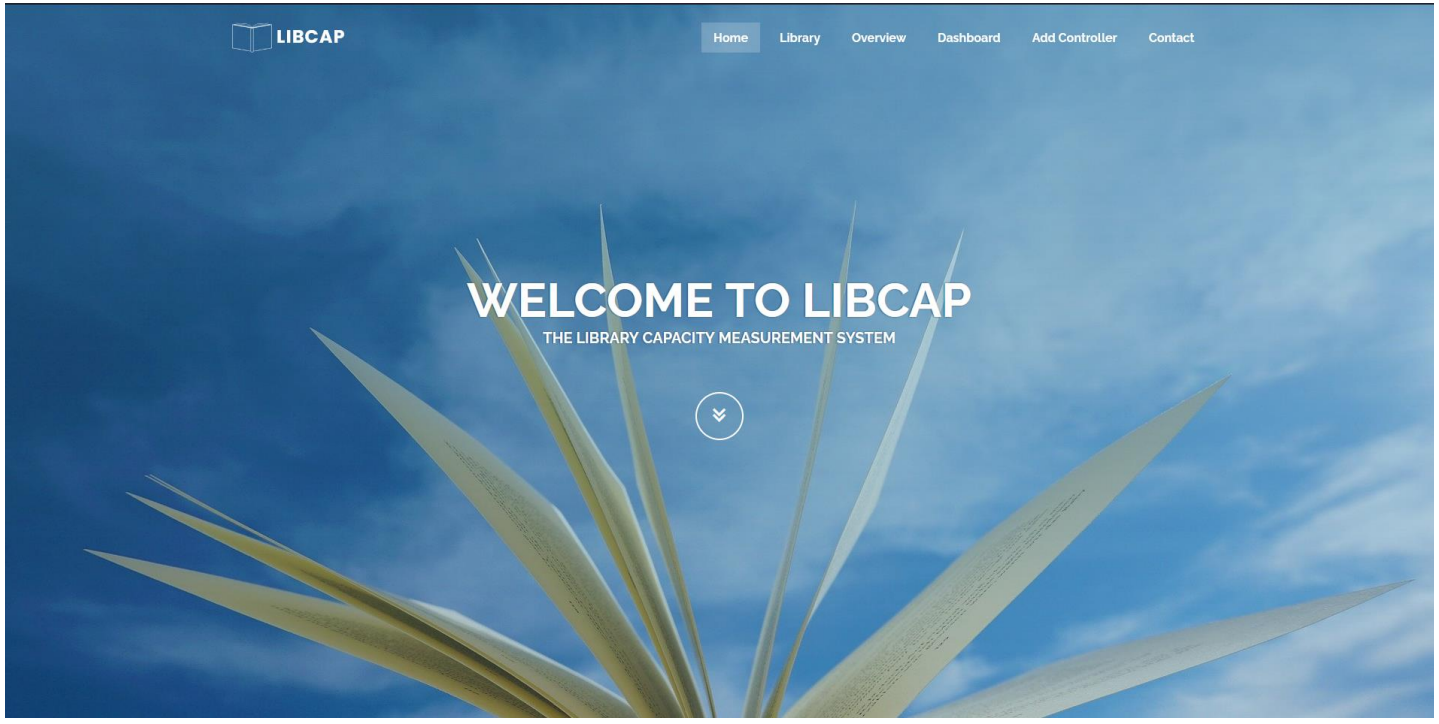


→ Datenbankabfragen

5. Umsetzung

Flask

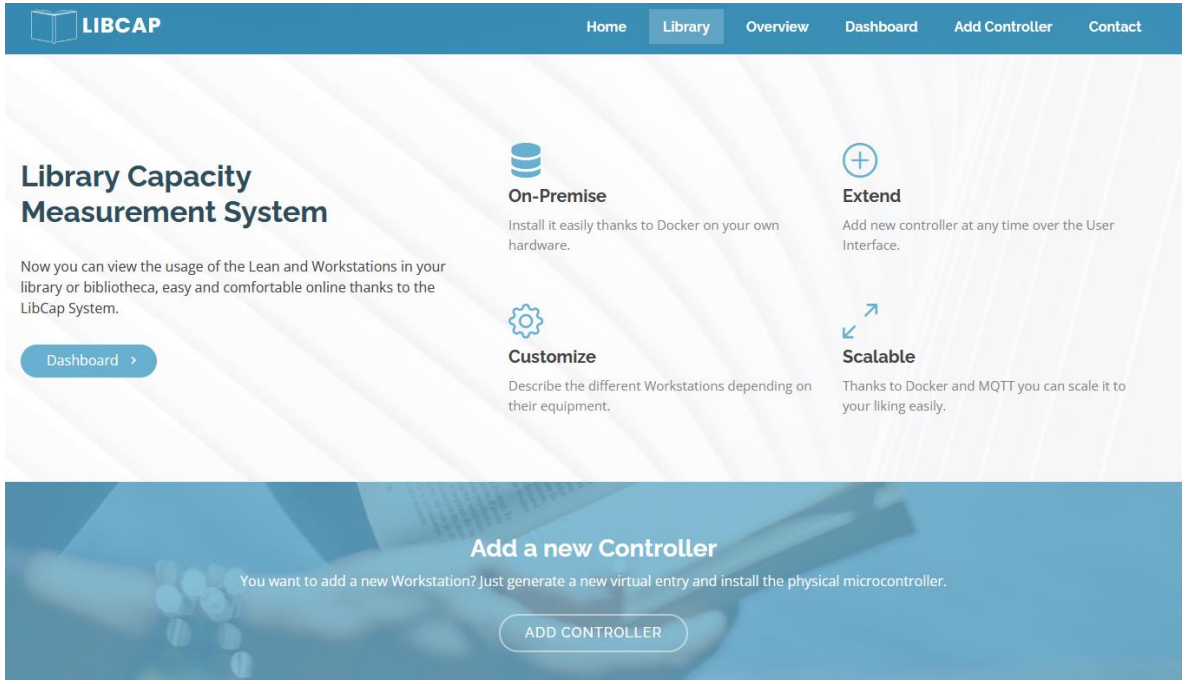
- Dynamisch generierbare Seiten über Jinja
- Keine Interaktivität auf Seite abbildbar über Flask
- Requests / Interaktivität nur abbildbar über HTTP Posts



5. Umsetzung

User Experience

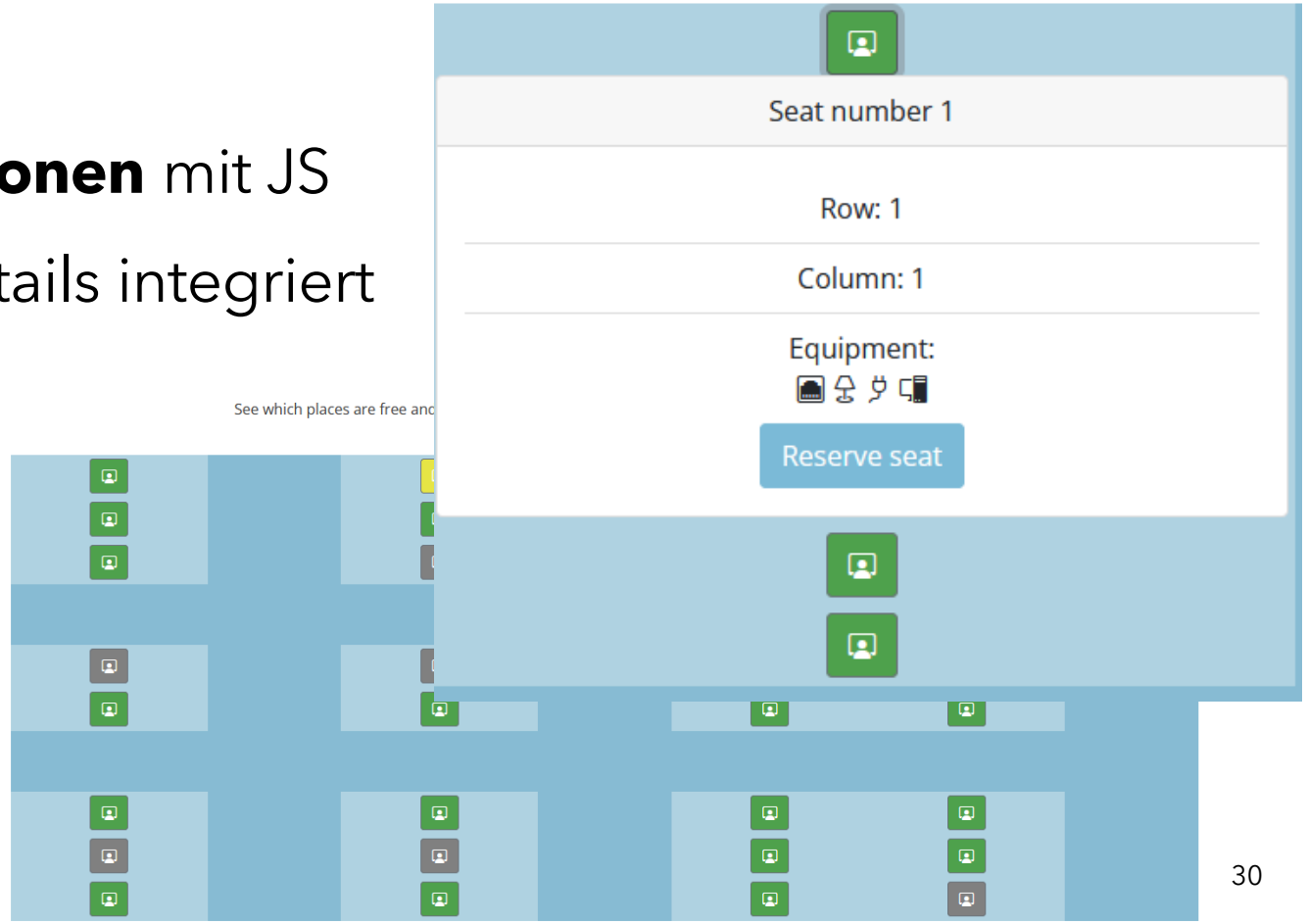
- CSS basierend auf **Bootstrap Vorlage**
- Responsive
- Single Page Webapp als MVP
- Navigation über Ankerpunkte
- Helles Farbschema



5. Umsetzung

Dashboard

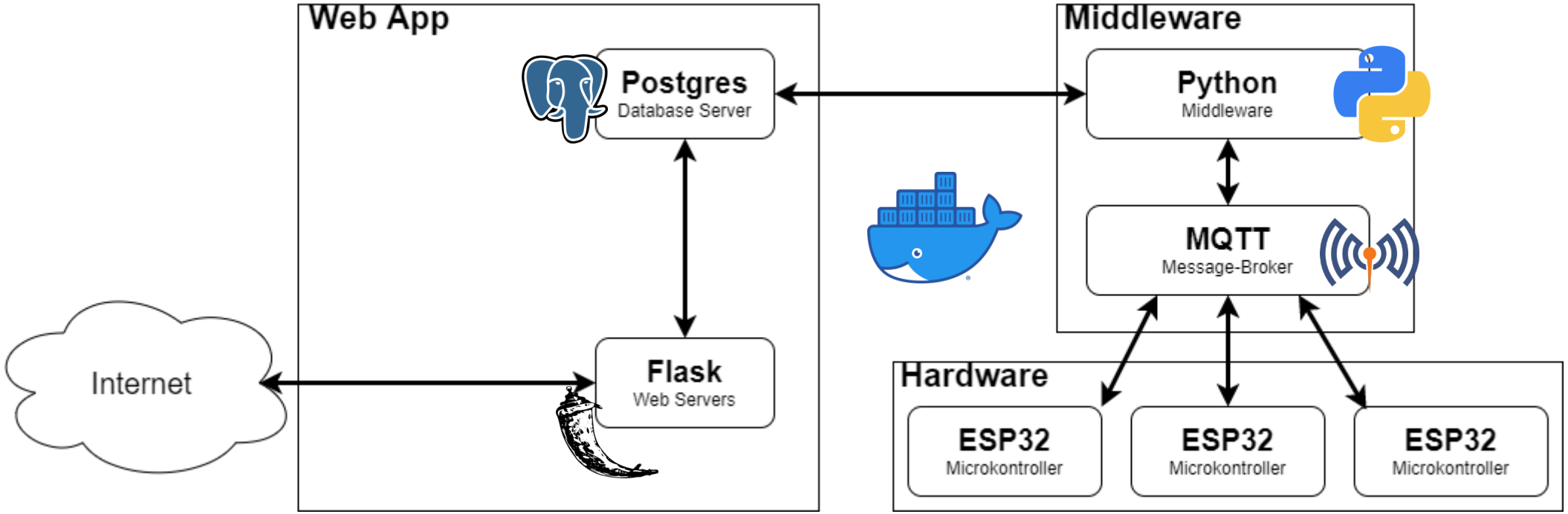
- Mapping aus **Datenbank**
- Fehlende Objekte aufgefüllt
- **Ausklappbare Detail-Informationen** mit JS
- **Reservier-Funktion** direkt in Details integriert



5. Umsetzung

Docker

- Jeder Teil **containerized**
- Einfach skalierbar



5. Umsetzung

Live Demo

6. Fazit

Vorteil:

- Loose Coupling
- Gute **Skalierbarkeit**
- Hohe Kohäsion

Nachteil:

- Security

Learnings:

- Praktische Arbeit mit **Message-Broker**
- Vertiefung in **Containerization**
- Grundsätzliche Funktionsweise von **Sensor-Systemen**
- Arbeiten mit Hardware


Vielen Dank!