Erfassung von Knallereignissen

Projektdokumentation: Lärmschutzsystem für Wohngebiete (Knallerfassung)

1. Ausgangslage & Anforderungen

Das Ziel ist die Echtzeitüberwachung von Lärmimmissionen (insb. Böller) in einem Wohngebiet (ca. 1 km²). Die Daten dienen der Stadtverwaltung zur Polizeilichen Lagebeurteilung. Eine personenbezogene Audioaufzeichnung ist strengstens untersagt (Datenschutz), es darf nur die Analyse der Lautstärke stattfinden.

Tech-Stack:

- **Endgerät (Node):** ESP32 DevKitC V4 + MEMS-Mikrofon SPH0645LM4H-B (I²S) + LoRa (868 MHz) + GPS.
- **Server:** Raspberry Pi 4 + SSD.
- **Netzwerk:** MQTT über LoRaWAN.
- **Datenbank:** InfluxDB (Zeitreihen) + PostGIS (Raumordnung) + TimescaleDB.
- **Visualisierung:** Echtzeit-Web-Map (GeoJSON/Leaflet).

---

2. Rechtliche Einordnung & Datenschutz (GDPR)

Wichtig: Audioaufnahmen gelten als „sensible Daten“ und fallen unter Art. 9 DSGVO.

- **Verbot:** Es darf **kein** Audio-File (.wav/.mp3) erzeugt oder gespeichert werden.
- **Erlaubt:** Die Verarbeitung von technischen Messwerten (RMS/Werte) zur Berechnung von Schalldruckpegeln (dB SPL).
- **Maßnahme:** Die ESP32-Firmware wird so programmiert, dass ein FFT (Fast Fourier Transform) auf den Audiostream ausgeführt wird, der Schalldruckpegel berechnet und die rohen Audio-Daten (Puffer) sofort im Speicher verwerfen („Burn and throw“).

---

3. Systemarchitektur

Das System besteht aus drei Schichten: Sensorknoten, Edge-Server und Cloud/Gateway-Server.

3.1 Sensorknoten (Node)

Jeder Knoten ist eine eigenständige Einheit, die Signale aufnimmt, lokal berechnet und über das Netz sendet.

1. **Eingabe:** MEMS-Mikrofon (SPH0645LM4H-B) über I²S.

2. **Verarbeitung:** ESP32 DevKitC (FFT-Berechnung, Triggerlogik).
3. **Ortung:** GPS-Modul (NEO-M8N) für Zeitstempel & Koordinaten.
4. **Funk:** LoRa-Modul (SX1276/SX1262) @ 868MHz.
5. **Versorgung:** Netzteil 5V/2A + Buck-Converter für 3.3V.

3.2 Backend (Raspberry Pi 4)

Zentrale Anlaufstelle für die LoRa-Gateway-Knoten.

1. **LoRa Gateway:** Empfängt 868MHz Signale.
2. **Broker (MQTT):** Mosquitto.
3. **Datenbank:**
   - *InfluxDB/TimescaleDB:* Speichert die Zeitreihe (Zeitstempel, dB-Wert, Node-ID).
   - *PostGIS:* Speichert den Standort der Nodes (geometrische Daten) und verknüpft Events räumlich.
4. **API:** Exponiert Daten via WebSocket oder REST für die Karte.

3.3 Frontend (Browser)

- Verbindung über WebSocket zum Backend.
- Empfang von GeoJSON-Daten.
- Darstellung auf einer Karte (z.B. Leaflet.js).

---

4. Hardware & Schaltungsübersicht

4.1 Sensor-Knoten Blockdiagramm (Textbeschreibung)

```
[Netzteil 5V/2A] --> [Buck-Converter (Step-Down)] --> [ESP32 (3.3V)]
                                                      |
                                                      +-- [SPH0645LM4H-B
(I2S: SCK, WS, SD)]
                                                      |
                                                      +-- [LoRa Module
(TX/RX/SS/D5)]
                                                      |
                                                      +-- [GPS Module
(TX/RX/3.3V)]
```

4.2 Komponentenliste (BOM) pro Knoten

- **ESP32 DevKitC V4:** ~ €10,00 (WLAN/BT & I²S Interface vorhanden)
- **Mikrofon:** SPH0645LM4H-B (InvenSense) -> ~ €5,00
- **LoRa Modul:** RFM95W / SX1276 (868MHz) -> ~ €6,00
- **GPS Modul:** NEO-6M oder NEO-M8N -> ~ €8,00
- **Buck-Converter:** MP1584 oder XL4015 (3.3V Ausgang) -> ~ €1,50

- **Netzteil:** USB Stecker auf DC 5.5x2.1mm (5V/2A) -> ~ €2,00
- **Sonstiges:** Platine, Kabel, Gehäuse, Akku (für Notversorgung) -> ~ €10,00
- **Kosten pro Knoten:** ca. **€42,50**

4.3 Server Komponenten (1 Zentrale)

- **Raspberry Pi 4 (4GB oder 8GB RAM):** ~ €80,00
- **SSD (NVMe oder SATA):** ~ €30,00
- **LoRa HAT (z.B. Dragino LoRa HAT):** ~ €30,00
- **Netzteil RPi:** ~ €10,00
- **Gehäuse & Kühlung:** ~ €15,00
- **Gesamtkosten Server:** ca. **€165,00**

---

5. Software-Logik

5.1 Firmware ESP32 (C++ / Arduino-IDE)

Die Firmware muss Noise Reduction (Dröhnunterdrückung) und FFT verwenden.

Pseudocode:

```
void loop() {
  // 1. Audio aufnehmen (z.B. 100ms Block)
  readI2S(audioBuffer);

  // 2. FFT durchführen (z.B. FastLED FFT Library oder CMSIS-DSP)
  float fftOutput[N];
  computeFFT(audioBuffer, fftOutput);

  // 3. RMS/Werte für Bass/Mitteltöne berechnen
  // (Böller haben viel Energie im Bassbereich)
  float loudness = calculateRMS(fftOutput);

  // 4. Pegel in dB umrechnen (Linear -> dB)
  // Voraussetzung: Referenzwert von SPH0645LM4H-B (z.B. 94dB @ 1Vrms)
  float db = linearToDb(loudness);

  // 5. GPS Werte aktualisieren
  readGPS();

  // 6. Trigger-Logik
  if (db > SCHWELLE_DB && isImpulse(audioBuffer)) {
     // Impuls-Erkennung: Kurzes Überschreiten des Limits (Böller!)

     // MQTT Paket bauen
     String payload = "{\"time\":\"" + timestamp + "\",\"lat\":\"" + lat +
"\",\"lon\":\"" + lon + "\",\"val\":" + db + ",\"type\":\"firework\"}";

```

```
        // Senden via LoRa
        LoRaSend(payload);
    }
}
```

5.2 Backend (Python/Node.js)

1. **MQTT Broker (Mosquitto):** Empfängt Payloads.
2. **InfluxDB Ingest:**
   - Tabelle: `measurements`
   - Tags: `node_id`, `location`
   - Fields: `value` (dB), `event_type`
   - Time: Aufnahmezeitpunkt
3. **PostGIS (PostgreSQL):**
   - Tabelle: `sensors`
   - Spalte: `geom` (POINT)
   - Tabelle: `events`
   - Spalte: `event_time`, `event_val`
4. **Websocket Server (Socket.io):**
   - Sobald ein neuer Eintrag in InfluxDB kommt, wird er in ein GeoJSON-Objekt konvertiert (`{type: "Feature", geometry: {type: "Point", coordinates: [lon, lat]}, properties: {db: 100}}`) und an alle verbundenen Clients gesendet.

---

6. Visualisierung (Karte)

Die Web-Oberfläche nutzt **Leaflet.js** (Open Source) für die Kartenansicht.

Funktionsweise:

1. Ein Marker erscheint, sobald ein dB-Wert > 80dB (Schwellenwert) registriert wird.
2. Die Markierung hat die Farbe Rot bei Böllern und Gelb bei starker Umgebungslärme.
3. Ein Klick auf den Marker zeigt Details (Zeitstempel, maximale dB-Lautstärke) in einem Popup an.

GeoJSON Beispiel (das vom Server gesendet wird):

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
```

```
        "type": "Point",
        "coordinates": [12.1234, 48.1234]
      },
      "properties": {
        "time": "2023-10-27T20:15:00Z",
        "db": 110,
        "source": "Sensor_01"
      }
    }
  ]
}
```

7. Implementierungsschritte

1. **Hardware-Phase:**
   - Beschaffung der Komponenten.
   - Löten der Sensorknoten (Entwicklung mit Steckplatine Breadboard möglich).
   - Aufbau des Raspberry Pi Servers.
2. **Software-Phase (Node):**
   - Programmierung des ESP32 (I²S Audio, FFT, GPS, LoRa).
   - Anpassung der Sendeleistung (TX Power) und Datenrate für die 1km² Fläche.
3. **Software-Phase (Server):**
   - Installation von InfluxDB, TimescaleDB, PostGIS und Mosquitto auf dem RPi.
   - Aufsetzen der MQTT-Consumer und Datenbank-Integration.
4. **Visualisierung:**
   - Entwicklung einer einfachen HTML/JS Webseite für die Live-Karte.
5. **Test & Kalibrierung:**
   - Testlauf: Böller simulieren und überprüfen, ob das System innerhalb von Sekunden reagiert.
   - Kalibrierung: Vergleich der gemessenen dB-Werte mit einem Referenz-Messgerät (wegen Richtungsabhängigkeit des Mikrofons).

8. Kostenübersicht (Gesamtsystem)

Komponente	Menge	Einzelpreis	Gesamtpreis
Sensorknoten Hardware	1 Stück	€42,50	€42,50
Raspberry Pi Server	1 Stück	€80,00	€80,00

Komponente	Menge	Einzelpreis	Gesamtpreis
SSD Server	1 Stück	€30,00	€30,00
LoRa HAT Server	1 Stück	€30,00	€30,00
Gesamtkosten			**~ €182,50**

(Preise können variieren, abhängig von Anbieter und Menge)