IIoT Gefahrenbereich-Warnsystem

Modul: Busines Process Automation (BPA)

Studiengang: Angewandte Informatik

Professor: Prof. Dr. Dirk Reichelt

Teammitglieder:

- Alexander Schulz (55297)
- Janne Puschke (55436)
- Magnus Andreas Böhne (56315)
- Matthias Kernke (53513)

Inhaltsübersicht

1. Einführung

In modernen Produktionsanlagen können Störungen in Druckluftsystemen gefährliche Situationen verursachen, insbesondere wenn Mitarbeitende keine unmittelbare Kenntnis dieser Probleme haben. Dieses Projekt nutzt die Kombination aus Echtzeit-Sensorüberwachung, Positionsdatenerfassung und haptischem Feedback, um eine proaktive Warnlösung zu schaffen. Das System wurde für das IIoT Testbed der HTWD Dresden entwickelt und demonstriert die praktische Anwendung von IIoT-Technologien zur Erhöhung der Betriebssicherheit. Bei erkannten Druckluftstörungen und gleichzeitiger Anwesenheit eines Mitarbeiters im Gefahrenbereich erfolgt eine Warnung über einen speziellen Handschuh mit Vibrationsfeedback, was die Reaktionszeit verkürzt und potenzielle Unfälle verhindert.

2. Architekturkonzept und Schnittstellen

2.1. Systemüberblick

Das IIoT Gefahrenbereich-Warnsystem dient dem Schutz von Mitarbeitern in der Fertigungsumgebung des IIoT Testbeds der HTWD Dresden. Das System erkennt Gefahrenbereiche, die durch technische Störungen entstehen, und warnt Mitarbeiter über einen tragbaren Handschuh mit Vibrationsfeedback.

Das Hauptszenario ist die Erkennung eines Druckabfalls in der Druckluftversorgung, der eine potenzielle Gefahr darstellt. Wenn ein Mitarbeiter einen solchen Gefahrenbereich betritt, erhält er eine Warnung.

2.2. Systemarchitektur

Die Systemarchitektur folgt einem Container-basierten Ansatz mit microservice-orientierten Komponenten, die über standardisierte Protokolle kommunizieren.

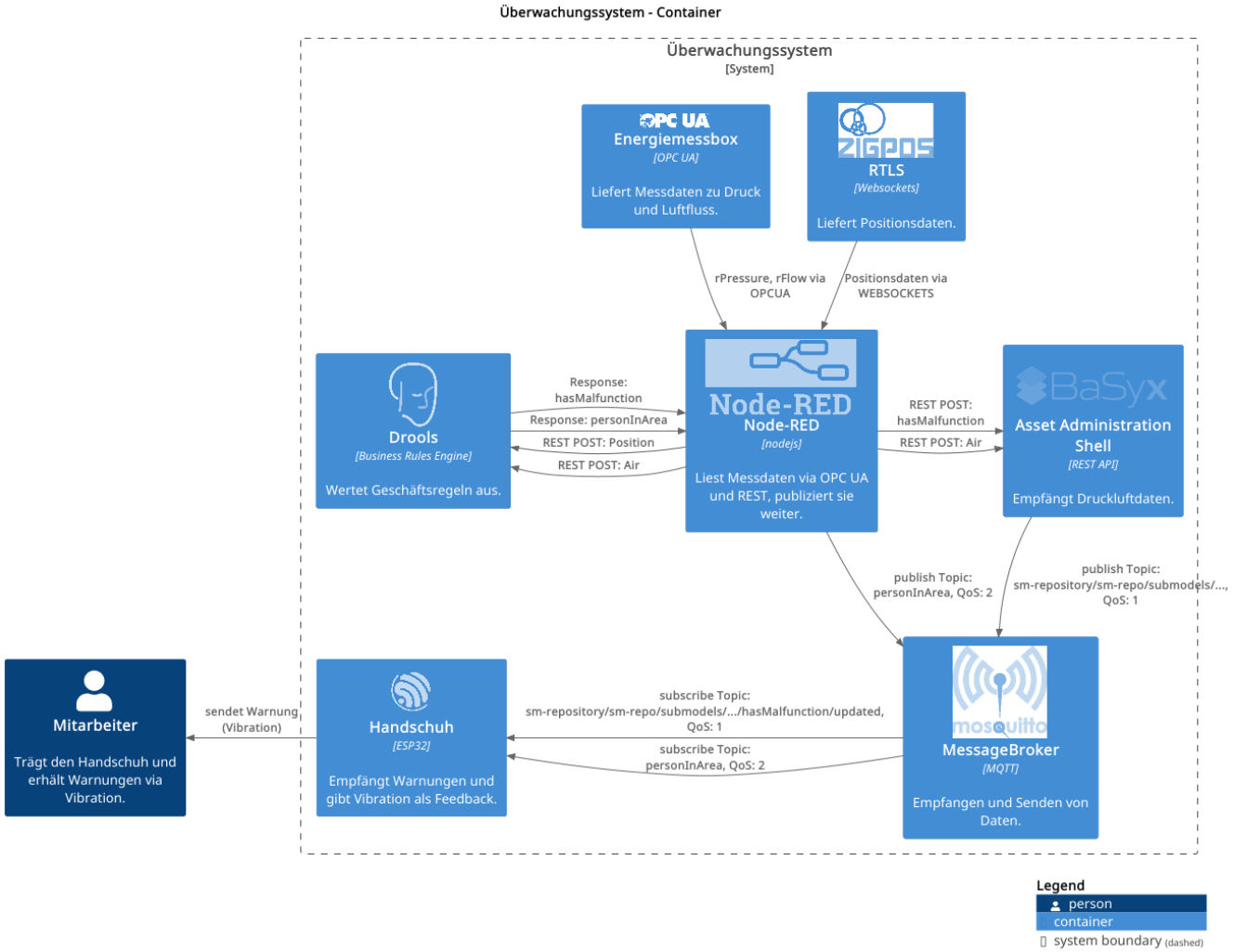

Abbildung 1. Systemdiagramm

2.2.1. Hauptkomponenten

Komponente	Beschreibung	Technologie
Energiemessbox	Erfasst Messdaten zu Druck und Luftfluss	OPC UA
ZIGPOS RTLS	Liefert Positionsdaten der Mitarbeiter	WebSockets
Node-RED	Zentraler Datenintegrations- und -verarbeitungsservice	Node.js
Drools	Business Rules Engine zur Auswertung von Geschäftsregeln	Java, Quarkus
Asset Administration Shell (AAS)	Modelliert das Druckluftsystem als Asset nach Industrie 4.0 Standards	BaSyx Framework
MQTT Message Broker	Ermöglicht die asynchrone Kommunikation zwischen Komponenten	Eclipse Mosquitto
Handschuh	IoT-Gerät, das Warnungen empfängt und Vibrationsfeedback gibt	ESP32

2.3. Datenfluss und Schnittstellen

2.3.1. OPC UA Schnittstelle zur Energiemessbox

Die Energiemessbox stellt Druckluftdaten über das OPC UA Protokoll bereit. Node-RED ruft die Daten alle 50 Millisekunden ab. Folgende Variablen werden abgefragt:

- `rPressure`: Aktueller Druck in Bar (NodeId: ns=2;s=|var|CECC-LK.Application.GVL.rPressure)
- `rFlow`: Aktueller Luftstrom in Bar (NodeId: ns=2;s=|var|CECC-LK.Application.GVL.rFlow)

2.3.2. WebSocket-Schnittstelle zum ZIGPOS System

Das ZIGPOS Real-Time-Location-System (RTLS) sendet Positionsdaten über eine WebSocket-Verbindung. Der Ablauf ist:

1. Authentifizierung über REST API, um einen API-Schlüssel zu erhalten (gültig für 60 Minuten)
2. Aufbau einer WebSocket-Verbindung mit dem API-Schlüssel
3. Empfang von Positionsdaten über das Topic "POSITIONS"
4. Filterung der Daten nach dem Tag "Martin Schmidt"

2.3.3. REST-Schnittstelle zu Drools

Node-RED sendet zwei Arten von Daten an die Drools Rules Engine:

1. **Druckluftdaten**: Druck und Durchfluss für die Erkennung von Störungen
2. **Positionsdaten**: Koordinaten des Mitarbeiters für die Erkennung, ob sich die Person im Gefahrenbereich befindet

Node-RED speichert die zuletzt empfangenen Positionsdaten zwischen, um sicherzustellen, dass Drools auch dann Positionsdaten erhält, wenn sich die Person im Gefahrenbereich befindet und keine Bewegung stattfindet (keine neuen Datem vom ZIGPOS). Im Node-RED Flow ist ein Join-Knoten integriert, der die Daten nur dann an Drools weiterleitet, wenn sowohl Positionsdaten als auch Druckluftdaten gleichzeitig vorliegen, oder nach spätestens 0,5 Sekunden alle verfügbaren Daten sendet.

Drools antwortet mit:

- `hasMalfunction`: Boolean-Wert, der angibt, ob eine Störung erkannt wurde
- `personInArea`: Boolean-Wert, der angibt, ob sich die Person im Gefahrenbereich befindet

2.3.4. REST-Schnittstelle zur Asset Administration Shell (AAS)

Node-RED sendet folgende Daten an die AAS, um den digitalen Zwilling des Druckluftsystems zu aktualisieren:

1. Aktuelle Druckluftdaten aus der Energiemessbox
2. Ergebnis der Störungserkennung von Drools (`hasMalfunction`)

Die AAS veröffentlicht Änderungen automatisch auf dem MQTT Broker.

2.3.5. MQTT-Schnittstelle zum Handschuh

Der MQTT Broker dient als zentrale Kommunikationsplattform. Der Handschuh abonniert zwei Topics:

1. `sm-repository/sm-repo/submodels/⋯/hasMalfunction/updated` (QoS 1): Information über erkannte Störungen
2. `personInArea` (QoS 2): Information darüber, ob sich die Person im Gefahrenbereich befindet

Bei positiven Signalen (true) auf beiden Topics vibriert der Handschuh stark für 3 Sekunden. QoS 1 wurde für das hasMalfunction-Topic gewählt, da die Energiemessbox alle 50 Millisekunden neue Daten sendet und ein höherer QoS unnötigen Overhead erzeugen würde. Das personInArea-Topic verwendet QoS 2, um an Randbereichen die korrekte Paket-Reihenfolge sicherzustellen. Der Handschuh gibt außerdem eine kurze, leichte Vibration aus, wenn die Netzwerkverbindung verloren geht.

2.4. Geschäftsregeln

Die Geschäftsregeln werden in Drools durch DMN-Diagramme (Decision Model and Notation) definiert.

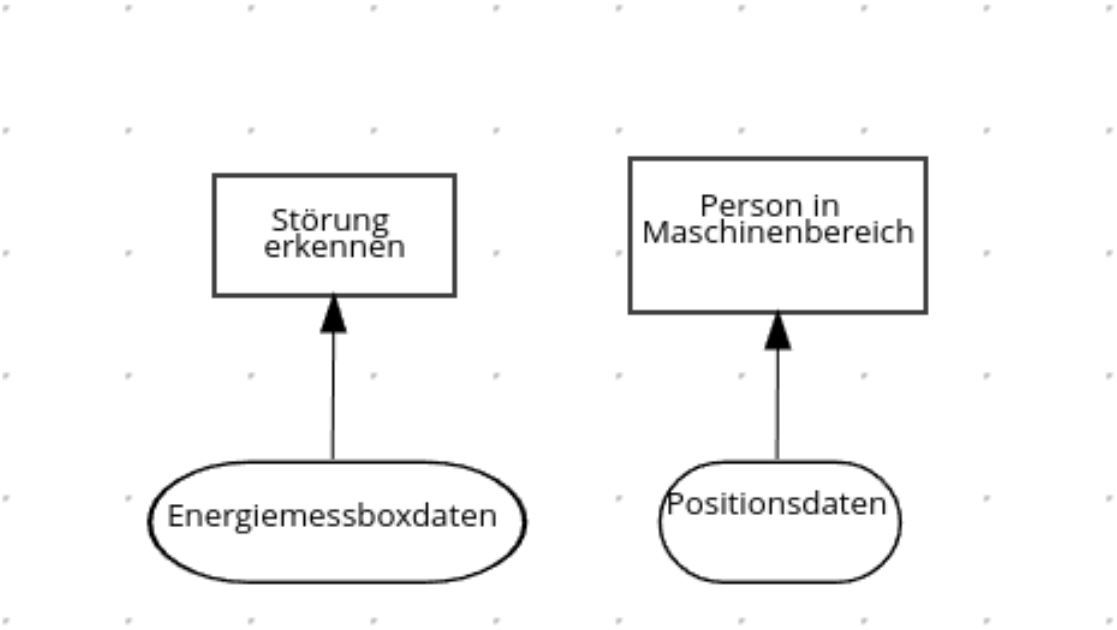

Abbildung 2. DMN Übersicht

2.4.1. Störungserkennung

Eine Störung wird erkannt, wenn:

- Der Luftstrom größer als 10 Bar ist UND

- Der Druck kleiner als 5,5 Bar ist

Die DMN-Hitpolicy für die Störungserkennung ist `FIRST`, sodass nur die erste zutreffende Regel ausgewertet wird.

hasMalfunction *(Decision Table)*

F	airData.airflow *(number)*	airData.pressure *(number)*	hasMalfunction *(decision)*	annotation-1
1	>10	<5.5	true	
2	-	-	false	

Abbildung 3. DMN Störungserkennung

2.4.2. Erkennung von Personen im Gefahrenbereich

Die Erkennung, ob sich eine Person im Gefahrenbereich befindet, erfolgt anhand der Position.

Person in Maschinenbereich *(Decision Table)*

F	Positionsdaten.x *(number)*	Positionsdaten.y *(number)*	Positionsdaten.z *(number)*	Person in Maschinenbereich *(Entscheidung)*	annotation-1
1	<6	>9	-	true	Ausgemessen mit dem Tag
2	-	-	-	false	

Abbildung 4. DMN Personenerkennung im Bereich

3. Entwickelte Artefakte und verwendete Services

3.1. Docker-basierte Systemumgebung

Das gesamte System wird mittels Docker Compose bereitgestellt, was eine einfache Deployment- und Betriebsstrategie ermöglicht. Alle Komponenten sind als Container definiert und können gemeinsam gestartet werden.

```
# Auszug aus docker-compose.yml
services:
  nodered:
    build:
      context: nodered
      dockerfile: Dockerfile
    container_name: nodered
    restart: always
    ports:
      - "1880:1880"
    # ...

  mosquitto:
    image: eclipse-mosquitto:2.0.15
    container_name: mosquitto
    # ...

  drools:
    build:
      context: ./drools
      dockerfile: Dockerfile
    container_name: drools
    # ...

  aas-env:
    image: eclipsebasyx/aas-environment:2.0.0-SNAPSHOT
    container_name: aas-env
    # ...

  aas-registry:
    image: eclipsebasyx/aas-registry-log-mongodb:2.0.0-SNAPSHOT
    platform: linux/arm64 # Hinweis: Die `platform`-Einstellung muss ggf. angepasst
werden, abhängig vom Zielsystem.
    container_name: aas-registry
    # ...

  # weitere Services für AAS, MongoDB, etc.
```

Link zur docker compose file

3.2. Node-RED Flows

Node-RED fungiert als zentrale Integrationsplattform und enthält einen Flows, der die folgenden Aufgaben erfüllen:

1. **Datenerfassung**: Abfrage der Druckluftdaten von der Energiemessbox über OPC UA
2. **Positionserfassung**: Empfang von Positionsdaten vom ZIGPOS-System über WebSockets
3. **Datenverarbeitung**: Kombination der Daten und Weiterleitung an Drools
4. **Regelauswertung**: Interpretation der Drools-Antworten
5. **AAS-Integration**: Aktualisierung der Asset Administration Shell
6. **Benachrichtigung**: Veröffentlichung von Warnungen über MQTT

Link zum Node-RED Flow

3.3. Drools Business Rules Engine

Die Drools-Komponente ist als Quarkus-basierter Microservice implementiert und enthält die DMN-Modelle zur Auswertung der Geschäftsregeln:

1. **Störungserkennung** (DMN): Bestimmt, ob die aktuelle Druckluftsituation eine Störung darstellt
2. **Positionserkennung** (DMN): Bestimmt, ob sich ein Mitarbeiter im Gefahrenbereich befindet

Drools bietet REST-Endpunkte für die Auswertung der Regeln, die von Node-RED aufgerufen werden.

Link zum DMN Modell für Störungserkennung

3.4. Asset Administration Shell (AAS)

Die AAS-Komponenten basieren auf dem Eclipse BaSyx Framework und stellen eine Industrie 4.0-konforme Modellierung des Druckluftsystems als digitaler Zwilling bereit. Folgende BaSyx-Services werden verwendet:

1. **AAS Environment**: Hosting der Asset Administration Shell
2. **AAS Registry**: Zentrales Verzeichnis der verfügbaren AAS
3. **Submodel Registry**: Verzeichnis der Submodelle
4. **AAS Discovery**: Service zum Auffinden von AAS-Komponenten
5. **AAS Web UI**: Weboberfläche zur Visualisierung und Interaktion

Link zur AAS Definitionsdatei

3.5. MQTT Message Broker

Der Eclipse Mosquitto MQTT Broker dient als zentrale Kommunikationsplattform und implementiert das Publisher-Subscriber-Muster. Er ermöglicht die lose Kopplung der Systemkomponenten und die zuverlässige Übermittlung von Warnungen an den Handschuh mit konfigurierbaren Quality-of-Service-Levels.

Link zur MQTT Broker Konfiguration

3.6. ESP32-basierter Handschuh

Der Handschuh ist ein tragbares IoT-Gerät, das auf einem ESP32-Mikrocontroller basiert. Es bietet folgende Funktionen:

1. WLAN-Konnektivität für die Verbindung zum MQTT Broker
2. MQTT-Client für den Empfang von Warnungen
3. Vibrationsmotor für haptisches Feedback
4. Batteriebetrieb für mobile Nutzung
5. Fehlertolerante Implementierung (Verbindungsverlusterkennung)

Der Handschuh vibrieret stark für 3 Sekunden, wenn sowohl eine Störung erkannt wurde als auch sich die Person im Gefahrenbereich befindet. Bei Verbindungsverlust gibt der Handschuh eine kurze, leichte Vibration aus.

Link zum Handschuh Quellcode

4. Einrichtung und Inbetriebnahme

4.1. Systemvoraussetzungen

- Docker und Docker Compose (Version 1.29+)
- Netzwerkverbindung zur Energiemessbox (OPC UA)
- Netzwerkverbindung zum ZIGPOS-System
- WLAN-Netzwerk für den Handschuh

4.2. Installation

1. Repository klonen:

   ```
   git clone https://github.com/a-schulz/bpa-energiemessbox.git
   cd bpa-energiemessbox
   ```

2. Konfigurationsdateien anpassen:
   - `pi/basyx/*.properties`: Konfiguration der BaSyx-Services
   - `pi/basyx/*.yml`: Konfiguration der Registry-Services
   - `pi/mosquitto/mosquitto.conf`: MQTT-Broker-Konfiguration
   - `esp32/src/secret.h`: Konfiguration des Handschuhs (WLAN, MQTT-Broker-Adresse)
3. Docker-Images bauen und Container starten:

   ```
   docker-compose up -d
   ```

Hinweis: Die `platform`-Einstellung (Prozessorarchitektur) muss ggf. angepasst werden, abhängig vom Zielsystem.

```
    # ...

  aas-registry:
    image: eclipsebasyx/aas-registry-log-mongodb:2.0.0-SNAPSHOT
    platform: linux/arm64 # <= HIER anpassen, abhängig vom Zielsystem
    container_name: aas-registry
    # ...
```

4.3. Konfiguration der Komponenten

4.3.1. Node-RED

Nach dem Start ist Node-RED unter http://localhost:1880 erreichbar.

1. Überprüfen Sie die Verbindung zur Energiemessbox:
    - Konfigurieren Sie die OPC UA-Verbindung mit der korrekten Serveradresse
    - Testen Sie die Verbindung mit dem "inject"-Knoten
2. Konfigurieren Sie die ZIGPOS-Verbindung:
    - Tragen Sie die API-Zugangsdaten ein
    - Überprüfen Sie die WebSocket-Verbindung
3. Konfigurieren Sie die Drools-Verbindung:
    - Stellen Sie sicher, dass die HTTP-Endpunkte korrekt konfiguriert sind
4. Konfigurieren Sie die AAS-Verbindung:
    - Überprüfen Sie die REST-Endpunkte

4.3.2. Drools

Die Drools-Engine ist nach dem Start unter http://localhost:8080 erreichbar.

1. Überprüfen Sie die DMN-Modelle:
    - Störungserkennung: http://localhost:8080/q/swagger-ui/ → POST /BPA
2. Testen Sie die DMN-Modelle:
    - Verwenden Sie die Swagger-UI oder Bruno (HTTP Client), um Testdaten zu senden.
    - Bruno Requestdefinitionen

      Beispiel-Request:

      ```
      {
        "airData": {
          "airflow": 5,
          "pressure": 4
        },
        "positionData": {
          "x": 4,
          "y": 5,
          "z": 5
        }
      }
      ```

      Beispiel-Response:

      ```
      {
        "positionData": {
      ```

```
    "x": 4,
    "y": 5,
    "z": 5
  },
  "personInArea": false,
  "airData": {
    "airflow": 5,
    "pressure": 4
  },
  "hasMalfunction": true
}
```

Anpassen der DMN Regeln

Um die Geschäftsregeln anzupassen (z.B. Schwellenwerte für die Störungserkennung oder Koordinaten des Gefahrenbereichs):

1. Öffnen Sie die DMN-Datei im Editor:
   - Pfad: `pi/drools/kogito-examples-main/kogito-quarkus-examples/dmn-quarkus-example/src/main/resources/energiemessbox.dmn`
   - Empfohlene DMN-Editoren: KIE DMN Editor (verfügbar unter https://kie.apache.org/docs/start/download)
2. Bearbeiten der Regeln:
   - Für die Störungserkennung passen Sie die Schwellenwerte für Luftdruck und Luftstrom an
   - Für den Gefahrenbereich definieren Sie die X/Y/Z-Koordinaten des überwachten Bereichs
   - Speichern Sie die Änderungen
3. Nach der Änderung bauen Sie den Drools-Container neu:

```
# Standard Build
docker compose up drools --build

# Bei Problemen mit dem Maven-Cache
docker compose up drools --build --no-cache
```

4. Testen Sie die geänderten Regeln über die Swagger-UI (http://localhost:8080/q/swagger-ui/)

4.3.3. Asset Administration Shell

Die AAS-Web-UI ist unter http://localhost:3000 erreichbar.

1. Überprüfen Sie, ob die AAS für das Druckluftsystem korrekt angezeigt wird
2. Kontrollieren Sie die Submodelle und ihre Eigenschaften
3. In der Web-UI kann auch überprüft werden, ob die Werte korrekt aktualisiert werden.

4.3.4. MQTT-Broker

Der MQTT-Broker kann mit einem MQTT-Client wie MQTT Explorer (http://mqtt-explorer.com/) überprüft werden:

1. Verbinden Sie sich mit dem Broker unter localhost:1883
2. Abonnieren Sie die Topics:
    - `sm-repository/sm-repo/submodels/⋯/hasMalfunction/updated`
    - `personInArea`

4.3.5. Handschuh

Für die Inbetriebnahme des Handschuhs:

1. Laden Sie den ESP Ordner herunter und öffnen Sie es in einer PlattformIO Umgebung.
2. Nennen Sie die dummysecret.h zu secret.h um.
3. Ergänzen Sie die neue secret.h mit den Credentials.
4. Verbinden Sie den ESP mit einem USB-C Kabel mit den PC und laden Sie die Software auf den Chip.
5. Der ESP verbindet sich automatisch, wenn die Verbinung unterbrochen ist vibriert er nur alle Sekunden leicht.
6. Wenn der ESP von Gelb zu Blau wechselt, ist er erfolgreich verbunden.
7. Testen Sie einen Durchlauf.

4.4. Testen des Gesamtsystems

1. Starten Sie alle Services mit `docker-compose up -d`
2. Überprüfen Sie, ob alle Komponenten korrekt gestartet wurden
3. Simulieren Sie eine Druckluftstörung (hoher Luftstrom, niedriger Druck)
4. Bewegen Sie den ZIGPOS-Tag in den Gefahrenbereich
5. Überprüfen Sie, ob der Handschuh vibriert

4.5. Troubleshooting

4.5.1. Neustarten der Services

Falls Probleme mit den laufenden Containern auftreten oder Konfigurationsänderungen wirksam werden sollen, können Sie die Services wie folgt neustarten:

```
# Navigieren Sie zum Projektverzeichnis
cd bpa-energiemessbox/pi

# Stoppen und entfernen Sie alle Container
```

```
docker-compose down

# Starten Sie die Container neu
docker-compose up -d
```

Der Parameter `-d` startet die Container im Hintergrund (detached mode). Entfernen Sie diesen Parameter, um die Container-Logs direkt im Terminal zu sehen.

4.5.2. Allgemeine Probleme

- Docker kann die Images nicht herunterladen oder bauen.
  - Überprüfen Sie Ihre Internetverbindung
  - Stellen Sie sicher, dass Docker korrekt installiert ist
  - Eventuell ist die Zeit des Systems falsch eingestellt (notwendig für TLS/SSL-Verbindungen)
    - Stellen Sie die Uhrzeit manuell ein oder synchronisieren Sie sie mit einem NTP-Server `sudo date --set="YYYY-MM-DD HH:MM:SS"`

4.5.3. Node-RED Verbindungsprobleme

- Überprüfen Sie die OPC UA-Verbindung zur Energiemessbox
- Kontrollieren Sie die ZIGPOS-API-Zugangsdaten
- Prüfen Sie die Drools-Endpunkte mit einem HTTP-Client

4.5.4. MQTT-Verbindungsprobleme

- Stellen Sie sicher, dass der MQTT-Broker läuft
- Überprüfen Sie die Zugriffsrechte und die Konfiguration
- Kontrollieren Sie die Topics und QoS-Einstellungen

4.5.5. Handschuh reagiert nicht

- Überprüfen Sie die WLAN-Verbindung des ESP32
- Kontrollieren Sie die MQTT-Verbindung
- Prüfen Sie die Batteriespannung
- Testen Sie den Vibrationsmotor manuell

4.5.6. AAS Probleme

- unter http://localhost:8081/swagger-ui/index.html findet man die REST Doku der AAS Schnittstelle.
  - wichtige Endpunkte sind folgende:
    - GET /shells : Gibt alle AAS zurück
    - GET /submodels: Gibt alle Submodels zurück

        - GET /submodels/:submodelIdentifier/$value: Gibt alle Werte des Submodels als JSON zurück. Achtung :submodelIdentifier durch echten Wert ersetzen. Dieser ist die id des submodels in UTF8-BASE64-URL enkodiert. Für das Submodel State ist dieser: c3RhdGU=
        - PATCH /submodels/:submodelIdentifier/submodel-elements/:idShortPath/$value: Ändert den Wert: Ändert die Werte eines Submodels in der AAS. :submodelIdentifier ist wieder in UTF8-BASE64-URL enkodiert,:idShortPath ist ein einfacher String. Um z.B. den Werte Pressure im Submodel State zu ändern muss submodelIdentifier:c3RhdGU= und idShortPath:Pressure

4.5.7. Zigpos Probleme

- Für Zigpos sind folgende Endpunkte wichtig:
    - POST /rest/oauth/authorization/unauth-token Zum authentifizieren. Der Body ist url encode und hat folgende Parameter
        - grant_type:password
        - client_id:3039379a-119f-467c-8f94-6ea02b764b99
        - username
        - password
    - GET /rest/devices Um alle devices zu bekommen. Man muss sich vorher authentifizieren und den Token als Bearer Token mitgeben

5. Ausblick

Das System bietet eine solide Grundlage für die Überwachung von Gefahrenbereichen in industriellen Umgebungen. Zukünftige Erweiterungen könnten Folgendes umfassen:

- Integration weiterer Sensoren (z.B. Temperatur, Feuchtigkeit)
- Erweiterung der Regelwerke in Drools für komplexere Szenarien
- Entwicklung einer grafischen Bneutzeroberfläche zum Anpassen der Drools Regeln für die einfachere Benutzung von nicht technischen Benutzern.
- Erweiterung des Feedbacks um audiovisuelle Elemente
- System gegen Netzwerkstörungen absichern