Inhaltsverzeichnis

1 Softwarearchitektur

1.1 Übermittlungsprotokoll

Für die Durchführung des Projekts müssen verschiedene Sensor- und Aktorkomponenten (vgl. Kapitel 2) mit einem IoT-Dashboard und Datenbanken (zur Speicherung historischer Daten) verbunden werden.

Zur Verbindung von IoT-Dashboard und Sensorkomponenten muss ein Protokoll ausgewählt werden. Gängige Machine to Machine (M2M)-Protokolle sind:

- MQTT (Message Queuing Telemetry Transport)
- REST (Representational State Transfer)
- XMPP (Extensible Messaging and Presence Protocol)
- AMQP (Advanced Message Queuing Protocol)
- Apache Kafka.

Wichtige Bewertungskriterien der Übertragungsprotokolle sind für unsere Anwendungsfälle:

- Stromverbrauch (aus Nachhaltigkeitsgründen)
- Schwere der Implementierung (aus Umsetzungsaspekten)
- Möglichst kleine Übertragungsdaten (für effektive Datenübertragung)
- Garantie der Nachrichtenübermittlung und des Empfangs (zur Sicherstellung der Übertragung)
- Plattformunabhängigkeit (zur Einbindung weiterer selbstentwickelter Sensor- und Aktorkomponenten)

Lösung: Der direkte Vergleich der verschiedenen Übermittlungsprotokolle zeigt, dass MQTT im Gegensatz zu den anderen Protokollen insbesondere durch seine Plattformunabhängigkeit, seine QoS (Quality of Service; Sicherstellung des Nachrichtenempfangs), sein Subscribe- und Publishing-System für den gewählten Einsatzzweck die beste Lösung darstellt.

Das Ergebnis lässt sich durch die folgenden Quellen nachvollziehen:

- Jasenka Dizdarević, Francisco Carpio, Admela Jukan, and Xavi Masip-Bruin. 2019. A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration. ACM Comput. Surv. 51, 6, Artikel 116 , November 2019). DOI: https://doi.org/10.1145/3292674
- https://www.nabto.com/mqtt-vs-rest-iot/
- https://internetofthingsagenda.techtarget.com/feature/XMPP-IoT-protocol-winner-or-second-place-to-MQTT
- https://www.sic-sales.de/iot-protokolle-mqtt-vs-amqp/
- https://www.kai-waehner.de/blog/2021/03/15/apache-kafka-mqtt-sparkplug-iot-blog-series-part-1-of-5-overview-comparison/

1.2 IoT-Plattform

Zur Sammlung, Speicherung, Steuerung, Aggregation und zum Anzeigen der gemessenen Daten und verbundenen Geräte soll auf eine IoT-Plattform zurückgegriffen werden. Zwingend notwendig dafür ist die Unterstützung des MQTT-Protokolls, welches für die Anwendung in diesem Projekt festgelegt wurde.

Auf dem Markt existieren verschiedene Plattformen wie:

- Thingsboard,
- ioBroker,
- Homeassistant,
- openHAB.

Die folgende Tabelle ermöglicht den Vergleich der Plattformen:

	Thingsboard	**ioBroker**	**Homeassistant**	**openHAB**
Kosten	Kostenfrei in der Community Edition mit eingeschränktem Funktionsumfang; Professional Edition kostet >10$/Monat	Einzelne Cloud-Komponenten sind kostenpflichtig	In der selbstgehosteten Variante kostenfrei	In der selbstgehosteten Variante kostenfrei
Open Source	Ja	Ja	Ja	Ja
Verbreitungsgrad	10600 GitHub Stars	930 GitHub Stars	>48000 GitHub Stars	10607 Github Stars
MQTT-Unterstützung	Teilweise; „Down-/ Uplinkconverter“ nur in der Professional Edition	Ja	Ja	Ja
Android-/iOS-App	Ja	Ja	Ja	Ja

Lösung: Grundsätzlich wäre mit allen o.g. Plattformen die Durchführung des Projektes möglich.

Für die Integration verschiedener Sensoren per MQTT wäre bei Thingsboard die Verwendung der kostenpflichtigen Professional Edition notwendig, bei der die Integration der Sensordaten durch die Programmierung von „Down-/Uplinkconvertern“ im Vergleich zu den anderen Plattformen aufwändiger und kostenpflichtig wäre. Daher wird Thingsboard als Alternative nicht weiter betrachtet.

Auf die Darstellung eines ausführlichen Vergleichs der weiteren Plattformen wird an dieser Stelle verzichtet. Als Grundlage dient hier die Bachelorarbeit „Untersuchung und Vergleich von Open Source Plattformen für das Smart Home“ von Prim Gersbacher, in welchen weiteren Informationen abrufbar sind. Für dieses Projekt wird die Plattform openHAB genutzt, welche laut Gersbacher als bester Kompromiss zwischen Größe und Komplexität gilt. Insbesondere die ansprechende Benutzeroberfläche und eine mit Java angelehnte Programmiersprache (es existiert Java-Programmiererfahrung im Projektteam) sollten die Durchführung des Projektes vergleichsweise einfach durchführbar machen.

Quellen:

- GitHub-Seiten und Webseiten der Dienste
- Pirim Gersbacher: Untersuchung und Vergleich von Open Source Plattformen für das Smart Home, Bachelorarbeit, Hochschule Offenburg, 2018, im Internet: https://opus.hs-offenburg.de/frontdoor/index/index/docId/2805.

1.3 Datenbank

Zur persistenten Speicherung von Sensordaten benötigt openHAB eine Datenbank. Grundsätzlich unterstützt dabei eine Vielzahl an Datenbanken, wie InfuxDB, montoDB, RRDtool, MariaDB, MySQL, DynamoDB, PostgreSQL, etc..

Im Rahmen dieses Projektes wird dabei MariaDB als Datenbank eingesetzt. Dabei handelt es sich um ein freies, Open-Source-Datenbankmanagementsystem (Form von MySQL), welches zu den meistverwendeten Datenbankensystem gehört. Auf dieses System wird zurückgegriffen, da das Projektteam im Vergleich zu den anderen Systemen bereits viele gute Erfahrungen mit MariaDB-Datenbanken sammeln konnte.

1.4 Serverarchitektur

Im Rahmen dieses Projektes wird auf Docker als freies Softwaretool zur Containervirtualisierung zurückgegriffen. Außerdem existieren für alle verwendeten Dienste (wie openHAB, MariaDB) vorkonfigurierte Images, die die Installation dieser Dienste vereinfachen und so die Sicherheit erhöhen. Im Gegensatz zum Einsatz virtueller Maschinen werden außerdem weniger Systemressourcen benötigt. Zudem kann das Projektteam bereits auf viele positive Erfahrungen beim Einsatz von Docker zurückgreifen.

Die vollständige Serverarchitektur ist außerdem im Einsatz- und Verteilungsdiagramm „IT-Systemstruktur“ dargestellt.

Dies ermöglicht die genutzten Dienste sowohl auf macOS, Windows oder Linux-Servern zu betreiben. Grundsätzlich müssen allerdings folgende Hardwareanforderungen mindestens erfüllt werden:

	Arbeitsspeicher	**CPU**	**Speicher**
MariaDB	512 MB	Max. 1 Kern	4 GB
openHAB mit Bindings	512 MB	Max. 1 Kern	100 MB
SWAG (nginx)	256 MB	1 Kern	100 MB
mosquitto (MQTT-Broker)	512 MB	Max. 1 Kern	100 MB
Insgesamt	2048 MB	2 - 4 Kerne	5 GB

Grundsätzlich wird deutlich, dass das System geringe Hardwareanforderungen benötigt. Diese werden beispielsweise durch einen Raspberry Pi 4 abgedeckt. Grundsätzlich kann es bei mehreren Anwendern mit mehreren Geräten sinnvoll sein die IoT-Plattformen auf einem Serversystem zu bündeln, um Synergieeffekte auszunutzen.

2 Systemauswahl

2.1 Sensorik

2.1.1 Überwachung der medizinischen Geräte

Ziel: Überwachung der Funktion und Zuverlässigkeit der medizinischen Geräte

> **Grund**: Medizinische Geräte (von Philli) sind nicht für den Dauerbetrieb zugelassen. Da diese im Dauerbetrieb genutzt werden, existiert ein erhöhtes Ausfallrisiko dieser Geräte. Ein Ausfall hätte potenziell tödliche Auswirkungen für die Patienten. Außerdem können so Daten über die Zuverlässigkeit der medizinischen Geräte gewonnen werden, die dem Hersteller zur Zulassung für den Dauerbetrieb zur Verfügung gestellt werden können.

Vorgehensweise: Um einen eventuellen Ausfall zu erkennen, soll die elektrische Leistung der medizinischen Geräte überwacht werden. Eine erhöhte elektrische Leistung oder anormal niedrige Leistung deutet auf einen Defekt der Geräte hin. In dem Fall ist die ordnungsgemäße Funktion der medizinischen Geräte zu überprüfen.

> **Grund**: Dies ist möglich, ohne die Funktionsweise der Geräte zu stören oder direkt in deren Wirkungsweise (wie z. B. bei einer Messung des Luftdurchsatzes) einzugreifen.

Anforderungen:

- Messung ohne Eingriff/Änderung der Funktionsweise der medizinischen Geräte
- Messung nur mit einem zugelassenen Leistungsmessgerät
  - Grund: Die Zuverlässigkeit der medizinischen Geräte ist entscheidend. Die Leistungsmessung soll nicht zu einer erhöhtem Ausfallrisiko führen. Daher soll auf ein Produkt zurückgegriffen werden, welches die einschlägigen Vorschriften der europäischen Union (v.a. Richtlinien 2014/35/EU, 2014/53/EU (RED), 2011/65/EU (RoHS)) erfüllt.
- Eine Messung von Leistungen im Bereich von 0 – 500 W
  - Grund: maximale Leistung der medizinischen Geräte liegt bei 460 W (AIRVO 2)
- Übertragung der Messergebnisse über MQTT-Schnittstelle an MQTT-Broker
  - Grund: MQTT wird als Softwarearchitektur eingesetzt und eine extra Insellösung soll vermieden werden.

Lösung:
Zwischen den Netzstecker der medizinischen Geräte und der Steckdose wird jeweils ein „Shelly Plug S“ geschaltet. Bei dem Shelly Plug S handelt es sich um eine smarte, MQTT-fähige Steckdose mit Leistungsmessung (0 bis 2500 W). Dieser Zwischenstecker erfüllt alle o.g. Rechtsvorschriften (vgl. Konformitätserklärung, https://shelly.cloud/knowledge-base/devices/shelly-plug-s/). Der Zwischenstecker kostet 20,99 €.

2.1.2 Überwachung der Zug-Kräfte

Ziel: Überwachung der Zug-Kräfte am Seil

> **Grund**: Über die Zug-Kräfte am Seil lassen sich Aussagen über das Verhaken des Wagens treffen. Außerdem können diese Daten, wie gezogene Kräfte und Aktivitätszeiten für Ärzte aufbereitet werden.

Vorgehensweise: Zwischen Zug-Seil und Wagen wird ein geeigneter Zug-Kraft-Sensor montiert.

Anforderungen:

- Messbereich im Bereich 0 – 200 N.
- Funktion im Kleinspannungsbereich
- Messung ohne Eingriff/Änderung der Funktionsweise der medizinischen Geräte
- Übertragung per MQTT an MQTT-Broker

Recherche:

1. Lorenz Messtechnik (aus Anfrage)
   Zugkraftsensor K-1107/200N – 656,- Euro/Stk.
   Messverstärker LCV/U5 – 174,- Euro/Stk.
2. RS PRO
   RS PRO Druck, Zug Wägezelle aus Edelstahl, bis 50kg – 770,22 Euro/Stk. zzgl. Messverstärker

Zzgl. Steuergerät zur Übertragung an MQTT-Broker, wie ESP32-Dev-Board

Lösung:

Auf eine Umsetzung der Überwachung der Zug-Kräfte wird im Rahmen des Projektes verzichtet. Eine Marktrecherche zeigt, dass geeignete Sensortechnik den Kostenrahmen des Gesamtprojektes um weiten übersteigt. Daher wird in Rücksprache mit dem Kostenverantwortlichen Marvin Christ, auf eine Umsetzung verzichtet, da die Kosten den Nutzen nicht rechtfertigen. Die Aufbereitung der Aktivitätszeiten für Ärzte wird zudem durch eine Messung der Wagenbeschleunigung (s. unten) anderweitig umgesetzt.

Die Umsetzung der Überwachung der Zug-Kräfte kann auf Anfrage und durch zusätzliche Finanzierung im Rahmen eines Updates durchgeführt werden.

2.1.3 Überwachung der Verbrauchsmaterialien

Ziel: Verbrauch des Sauerstoffs, der Nahrung und des Wassers beobachten und detektieren, falls die Verbrauchsmaterialien verbraucht sind.

> **Grund**: Die Verbrauchsmaterialien sind für die medizinische Versorgung der Patienten zwingend notwendig. Falls keine Verbrauchsmaterialien mehr vorhanden sind, besteht eine große Gefahr für die Patienten.

Vorgehensweise: Messen der Gewichte der einzelnen Verbrauchsmaterialien

> **Grund:** Mit zunehmendem Verbrauch nimmt die Masse der Sauerstoffflasche, der Nahrungs- und Wasserbeutel ab. So lässt sich der Verbrauch der Verbrauchsmaterialien messen, ohne diese z. B. durch eingebrachte Füllstandssensoren zu kontaminieren.

Anforderungen:

- Messbereich von 0 – 5,66 kg (gefüllte Sauerstofflasche im Freelox Stroller)
- Genauigkeit ±20 g
- Messung ohne Eingriff/Änderung der Funktionsweise der medizinischen Geräte
- Funktion im Kleinspannungsbereich

Recherche:

- Wägezellen mit Messverstärker und ESP32-Dev-Board

Lösung:

Auf eine Umsetzung der Überwachung der Verbrauchsmaterialien wird im Rahmen des Projektes verzichtet. Im Laufe der Bearbeitung des Projektes wurden neue Informationen der Eltern zur Verfügung gestellt: die medizinischen Geräte überwachen die Füllstände selbstständig und alarmieren, falls die Verbrauchsmaterialien verbraucht sind. Außerdem sei der Verbrauch an Sauerstoff, Wasser und Nahrungsmittel kontinuierlich. Die Alarmierung so gegeben ist und die Erstellung von Verbrauchsstatistiken so nicht weiter sinnvoll.

Als große Herausforderung hat sich außerdem die konstruktive Integration von Wägezellen in den Wagen gezeigt, da die medizinischen Geräte und Verbrauchsmaterialien verschiedene Geometrien besitzen und so individuelle Aufhängungen erarbeitet werden müssen.

2.1.4 Überwachung des Wagens und der Umgebung

Ziele: Detektieren, falls der Wagen umkippt und das Messen der Umgebungstemperatur

Grund: Falls der Gerätewagen umkippt, ist eine Beschädigung der Schläuche oder medizinischen Geräte nicht ausgeschlossen. So geht eine große Gefahr für den Patienten aus. Der Gerätewagen ist zwar möglichst kippsicher ausgelegt. So kann normalen Belastungsszenarien kann das Kippen des Wagens ausgeschlossen werden. Bei anormalen Belastungen, wie dem Herunterfallen über eine Treppe, das einseitige Überfahren hoher Hindernisse oder grober Gewalteinwirkung kann das Kippen allerdings nicht vollständig ausgeschlossen werden. Da wie oben beschrieben vom Kippen eine große Gefahr für den Patienten ausgeht.

Da die medizinischen Geräte nur für Umgebungstemperaturen im Bereich von 20 °C bis 28 °C vorgesehen sind, sollte die Einhaltung dieser Temperaturen überwacht werden.

Vorgehensweise: Überwachung der Winkelgeschwindigkeit des Wagens, Überwachung der Temperatur

Grund: Eine unvorhergesehene Änderung der Winkelgeschwindigkeit in Kipprichtung des Wagens deutet auf ein Umkippen des Wagens hin.

Anforderungen:

- Übertragung der Sensordaten per MQTT-Protokoll an MQTT-Broker
- Funktion im Kleinspannungsbereich
- Messbereich der Winkelgeschwindigkeit: 0 – 200 °/s
- Messbereich der Temperatur: -20 °C – 40 °C
- Messung ohne Eingriff/Änderung der Funktionsweise der medizinischen Geräte

Recherche:

1. **<u>1528-3886-ND (MPU-6050) Sensor</u> und <u>ESP32-Dev-Board</u>**
   - Kombinationssensor zur Messung von Beschleunigung und Winkelgeschwindigkeit aller drei Achsen
   - Erfassungsbereich: ±16 g, ±2000°/s, -40 – 85 °C
   - Protokoll zwischen Sensor und ESP32-Dev-Board: I^2C
   - ESP32-Devboard: WLAN Netzwerkzugang (nötig für MQTT), MQTT- und MPU6050-Bibliotheken
   - Kosten: MPU-6050 – 6,13 €; ESP32-Dev-Board – 11,99€

2. **MIKROE-1577 (MPU-6000) mit ESP32-Dev-Board**
    - Kombinationssensor zur Messung von Beschleunigung und Winkelgeschwindigkeit aller drei Achsen
    - Erfassungsbereich: ±2000°/s, -40 – 85 °C
    - Protokoll zwischen Sensor und ESP32-Dev-Board: I^2C oder SPI
    - WLAN-Netzwerkzugang (nötig für MQTT), MQTT- und MPU6050-Bibliotheken
    - Kosten: MPU-6000 – 34,37 €; ESP32-Dev-Board – 11,99€

3. **ADIS16470/PCBZ-ND Sensor und ESP32-Dev-Board**
    - Kombinationssensor zur Messung von Beschleunigung und Winkelgeschwindigkeit aller drei Achsen
    - Erfassungsbereich: ±40 g, ±2000°/s, -25 – 85 °C
    - Protokoll zwischen Sensor und ESP32-Dev-Board: SPI
    - WLAN-Netzwerkzugang (nötig für MQTT), MQTT- und MPU6050-Bibliotheken
    - Versorgungsspannung Sensor: 3 – 3,6 V
    - Kosten: Kosten: MPU-6000 – 49,35 €; ESP32-Dev-Board – 11,99€

Lösung: Das 1528-3886-ND-Board (mit dem MPU-6050) in Kombination mit einem ESP32-Dev-Board erfüllt alle Anforderungen und ist im Vergleich zu den anderen Optionen am kostengünstigsten. Daher wird diese Lösung ausgewählt.

2.1.5 Überwachung der Wagenbelastungen für Updates und Upgrades

Ziel: Erfassung von Belastungen, die im Betrieb auf den Wagen wirken.

> **Grund**: Zu Beginn des Entwicklungsprozesses hat eine Recherche gezeigt, dass wenig bzw. keine öffentlichen Daten bezüglich Belastungen von solchen Wagen zugänglich sind. Um diesen Wagen in der Zukunft weiterzuentwickeln oder ähnliche Wagen in der Zukunft entwickeln, ist es daher sinnvoll Daten über die Belastungen im Betrieb zu sammeln und aufzuzeichnen.

Vorgehensweise: Messung der Beschleunigungen des Wagens.

> **Grund**: Mit den Beschleunigungen in alle drei Raumrichtungen können Aussagen über Erschütterungen getroffen werden. Außerdem kann mit der Wagenmasse die auf den Wagen wirkende Kraft mit dem zweiten Newtonschen Axiom berechnet werden.

Anforderungen:

- Übertragung der Sensordaten per MQTT-Protokoll an MQTT-Broker
- Funktion im Kleinspannungsbereich
- Messbereich der Beschleunigungen: ± 8g

Recherche:

1528-3886-ND (MPU-6050) Sensor und ESP32-Dev-Board

- Weitere Informationen im Kapitel 2.1.4
- Erfassungsbereich: ±16 g

Lösung: Die Messung der Beschleunigungen in alle drei Raumrichtungen ist mit dem bereits vorgesehenen 1528-3886-ND-Board (MPU-6050-Sensor) möglich. Eine Übertragung mit dem MQTT-Protokoll an den Broker wird über das ESP32-Dev-Board vorgesehen. Da alle Komponenten bereits vorhanden sind und auch diese Anforderungen erfüllen, wird auf eine ausführlichere Marktrecherche verzichtet und die Messung der Beschleunigungen in dieses Sensorsystem integriert.

2.2 Aktorik

2.2.1 Überblick

Ziel: Alarmierung der richtigen Ansprechpartner bei Fehlfunktionen über mehrere redundanten Systeme im Nah- und Fernbereich des Wagens

> **Grund**: Die medizinische Versorgung der Patienten kann aufgrund vielfältiger Gründe eingeschränkt werden. Dies kann zu lebensbedrohlichen Zuständen führen. Die wichtigsten Gründe können wie oben beschrieben durch die eingesetzten Sensorsysteme detektiert werden. Nach der Detektion ist eine Alarmierung notwendig, um die ordnungsgemäße Funktion der Medizingeräte und des Wagens sicherzustellen und eventuell die Funktionsfähigkeit wiederherzustellen.
>
> Auf mehrere redundanten Systeme sollte zurückgegriffen werden, um auch bei Ausfall eines oder mehrerer Systeme die erforderliche Warnung durchführen zu können. Um mögliche Risikofaktoren und Fehlerursachen zu streuen, sollte auf verschiedene Warnsysteme zurückgegriffen werden.

2.2.2 Akustische Warnung

Ziel: Alarmierung von Personen im unmittelbaren Umfeld des Wagens und angrenzender Bereiche

Vorgehensweise: Alarmierung durch akustisches Warnsignal

> **Grund:** eine akustische Warnung ist im unmittelbaren Umfeld des Wagens und angrenzender Bereiche gut wahrzunehmen, auch wenn der Wagen nicht beobachtet wird.

Anforderungen:

- Lautstärke des Warnsignals: >65 dB (A) (vgl. DIN EN ISO 7731)
- Übertragung der Auslösedaten per MQTT-Protokoll an MQTT-Broker
- Funktion im Kleinspannungsbereich
- Alarmierung ohne Eingriff/Änderung der Funktionsweise der medizinischen Geräte

Recherche:

Keyestudio SC8002B Lautsprechermodul am ESP32-Dev-Board

- Enthält Lautsprecher und Verstärker
- Leistungsaufnahme: 2 W
- Betriebsspannung: DC 5 V
- Kosten: 7,98 €
- Lautstärke: 80 dB (A)

Lösung: Am bereits eingeplanten ESP32-Dev-Board wird die Lautsprechereinheit SC8002B integriert. Diese kann ein ausreichend lautes Warnsignal abspielen.

2.2.3 Visuelle Warnung

Ziel: Redundante Alarmierung von Personen im unmittelbaren Umfeld des Wagens

Vorgehensweise: Visuelle Alarmierung durch Pulsieren einer Lichtquelle in Signalfarbe unabhängig vom ESP32-Dev-Board

> **Grund:** So existiert eine Alarmierungsmöglichkeit, falls Personen im unmittelbaren Umfeld des Wagens keine Audiowarnungen wahrnehmen können (z. B. bei Taubheit oder dem Tragen von Kopfhörern) oder das ESP32-Dev-Board/ der Lautsprecher ausfällt.

Anforderungen:

- Ausreichende Helligkeit
- Unabhängiger Betrieb ohne ESP32-Dev-Board
- Übertragung der Auslösedaten per MQTT-Protokoll an MQTT-Broker
- Alarmierung ohne Eingriff/Änderung der Funktionsweise der medizinischen Geräte

Recherche:

SHELLY DUO E27RG

- MQTT-fähige WLAN-Glühbirne
- Farben mit RGBW-Regelbar
- Genormte E27-Fassung

Lösung: Mit der Shelly Duo E27RG-Glühbirne kann unabhängig vom ESP-32-Board und dem Lautsprecher im unmittelbaren Nahbereich des Wagens redundant erfolgen.

2.2.4 Warnung auf Smartphone

Ziel: Redundante Alarmierung von entfernten Personen (z. B. Eltern) die sich nicht im unmittelbaren Nahbereich des Wagens befinden

> **Grund:** Sollten sich wichtige entfernte Personen (z. B. die Eltern) weiter vom Wagen entfernt befinden, könnten diese die oben beschriebenen audio-visuellen Warnsignale nicht war nehmen. Außerdem existiert so eine redundante Alarmierungsmöglichkeit, falls die gesamte Stromversorgung des Wagens nicht mehr gegeben ist.

Vorgehensweise: Alarmierung über einen Smartphone Messengerdienst

> **Grund:** Jedes handelsübliche Smartphone besitzt mit dem eingebauten Akku eine Sicherung vor Stromausfällen. Außerdem erfahren Smartphones und Messengerdienste eine weite Verbreitung und ermöglichen so auch technisch nicht interessierten Menschen den leichten Zugang zu den Warnungen.

Anforderungen:

- Unabhängiger Betrieb ohne ESP32-Dev-Board, MQTT-Glühbirne
- Auslösung der Warnung durch openHAB
- Alarmierung ohne Eingriff/Änderung der Funktionsweise der medizinischen Geräte
- Smartphone App muss über gängige Appstores installierbar sein

Recherche:

- WhatsApp
  - Kostenpflichtiger API-Zugriff für Geschäftskunden
  - Keine einfache Integration in openHAB
  - Höchster Verbreitungsgrad in Deutschland
- Telegram
  - Kostenloser API-Zugriff
  - Telegram-Binding in openHAB ermöglicht einfache Integration in Gesamtsystem
  - Hoher Verbreitungsgrad in Deutschland
- Signal
  - Kostenloser API-Zugriff
  - Integration über Skripte in openHAB
  - Mittel-hoher Verbreitungsgrad in Deutschland
- SMS
  - Integration per IFTTT in openHAB
  - Versand kostenpflichtig
  - Nutzbar ohne App-Installation

Lösung: Zur Alarmierung wird Telegram genutzt, da Telegram einen kostenlosen API-Zugriff bietet und einfach in openHAB integriert werden kann. Außerdem besitzt Telegram eine hohe Verbreitung. Durch das Open-Source Telegram-Binding in openHAB, welches recht weit verbreitet ist, stellt sicher, dass auch in der Zukunft Updates zur Sicherstellung der Telegram-Benachrichtigung eingespielt werden können, im Gegensatz zu den inoffiziellen Signal-Skripten.

2.3 Stromversorgung

Die gewählten Komponenten müssen mit folgender Spannung und Stromstärke versorgt werden:

Komponente	**Spannung**	**Max. Stromstärke**
ESP32-Dev-Board	3,3 – 12 V DC	150 mA
MPU-6050 Sensor (1528-3886-ND-Board)	2,375 – 3,46 V DC	3,9 mA
Keyestudio SC8002B	5 V DC	400 mA
Insgesamt	3,3 V DC und 5 V DC	~600 mA

Das ESP32-Dev-Board bietet grundsätzlich die Möglichkeit, dass man das Board mit einer Spannung von 5 V über den verbauten Micro-USB-Anschluss versorgt. Außerdem besitzt es einen Gleichspannungswandler, so dass über einen Pin 3,3 V bereitgestellt werden können. Über einen anderen Pin können außerdem 5 V für weitere Komponenten zur Verfügung gestellt werden. So ist die Stromversorgung des gesamten Sensorboards sichergestellt. Das Board kann dabei bis zu 150 mA am 3,3 V Output und ~900 mA am 5 V Output zur Verfügung stellen, was die Anforderungen erfüllt.

Das USB Netzteil muss also mindestens 0,6 A zur Verfügung stellen, eingesetzt werden handelsübliche Netzteile mit einer maximalen Stromstärke von 1 A.

Die MQTT-Glühbirne bezieht ihre Spannung über einen E27-Sockel mit Netzspannung. Auch der Shelly Plug S kann als Zwischenstecker seine Stromversorgung über die Netzspannung sicherstellen.