keyestudio - Smart Beach Resort

CHRISTOPH ASTLINGER (2410781003@hochschule-burgenland.at), WOLFGANG GUNDACKER (2410781022@hochschule-burgenland.at), and LISA VAN DEN HEUVEL (2410781017@hochschule-burgenland.at)

Contents

Contents 1
1 Einleitung 2
2 Use Case View 3
2.1 Einzel-Use Cases 3
2.2 Gesamt-Use Case – Smart Beach Resort Management 3
3 Logical View 3
4 Development View 5
5 Process View 5
5.1 Beispielprozess – Sturmwarnung 5
5.2 Kommunikationsparameter 6
5.3 Beispielprozess – Gästewechsel 6
6 Physical View 6
7 Schlussfolgerungen 7
References 9

Authors' Contact Information: Christoph Astlinger (2410781003@hochschule-burgenland.at); Wolfgang Gundacker (2410781022@hochschule-burgenland.at); Lisa van den Heuvel (2410781017@hochschule-burgenland.at).

1 Einleitung

Das Projekt *Smart Beach Resort* untersucht, wie moderne IoT-Technologien in einem praxisnahen Umfeld einsetzbar sind, um den Betrieb von Ferienhäusern effizienter, sicherer und nachhaltiger zu gestalten. An der niederländischen Nordseeküste stehen drei intelligente Modellhäuser, die gemeinsam ein digitales Resort bilden. Die Häuser sind durch ein hybrides Kommunikationssystem miteinander verbunden, das lokale Bluetooth-Verbindungen mit einer Cloud-basierten MQTT-Infrastruktur kombiniert. Jedes Haus erfüllt eine spezifische Funktion und trägt zum Gesamtsystem bei: Überwachung von Wetter und Sturm, Gäste-Komfort und Sicherheit sowie Energie- und Wartungsmanagement. Die Verbindung dieser Einzelfunktionen schafft ein autonom agierendes System, das auf Umweltbedingungen und Nutzeraktionen selbstständig reagiert.

Die physische Umgebung stellt besondere technische Herausforderungen dar. Salzhaltige Luft, hohe Luftfeuchtigkeit, Sandablagerungen und häufige Stürme führen zu Korrosionsgefahr, Sensorbelastung und instabilen Funkbedingungen. Ziel des Projekts ist daher nicht nur die Entwicklung funktionaler Einzellösungen, sondern auch der Nachweis, dass ein IoT-System unter realistischen Umweltbedingungen zuverlässig arbeiten kann. Darüber hinaus soll gezeigt werden, wie drei getrennt entwickelte Einheiten zu einem konsistenten Gesamtsystem verknüpft werden können, das sowohl lokal als auch cloudbasiert funktioniert.

Der Ansatz basiert auf der Nutzung standardisierter IoT-Komponenten. Die Häuser sind mit einem **ESP32**-Controller ausgestattet, der Sensordaten über Temperatur, Luftfeuchtigkeit, Bewegung, Gasgehalt oder RFID-basierte Zugänge erfasst. Die Daten werden über das **MQTT-Protokoll** an den Cloud-Broker **HiveMQ** übertragen und dort verarbeitet. Ein zentraler **Node-RED**-Server übernimmt die Steuerungslogik, aggregiert Ereignisse und reagiert auf erkannte Zustände, beispielsweise durch das Schließen von Fenstern bei Sturm oder das Abschalten der Lüftung bei hoher Feuchtigkeit. Damit entsteht eine klare funktionale Trennung zwischen Geräte-, Kommunikations- und Steuerungsebene.

Die Entwicklung erfolgt dezentral. Jede Teilgruppe arbeitet an einem eigenen Hausmodul, das unabhängig funktionsfähig ist, sich jedoch über ein gemeinsames Topic-System in die Resort-Architektur integrieren lässt. Da die Entwickler an unterschiedlichen Standorten arbeiten, wird die Verbindung über mobile Hotspots realisiert. Dies zeigt, dass auch verteilte Entwicklungsteams IoT-Systeme aufbauen können, ohne auf lokale Netzwerke oder physische Kopplung angewiesen zu sein. Die Cloud-Anbindung erlaubt die Zusammenarbeit in Echtzeit und vereinfacht die Integration der einzelnen Module zu einem gemeinsamen Use Case.

Das Projekt richtet sich an Betreiber von Ferienanlagen, kleine Hotels und Gebäudeverwalter, die ihre Objekte aus der Ferne überwachen und steuern möchten. Die Kombination aus kostengünstiger Mikrocontroller-Hardware, Cloud-Integration und visueller Steuerungslogik (Node-RED) zeigt, dass sich ein voll funktionsfähiges Smart-Home-System auch ohne teure Speziallösungen realisieren lässt. Der Lösungsansatz kann auf andere Einsatzgebiete wie Studentenwohnheime, Serviced Apartments oder modulare Ferienparks übertragen werden. Wirtschaftlich betrachtet bietet das System eine Reduktion von Personalaufwand und Energieverbrauch sowie eine Erhöhung der Betriebssicherheit und Gästezufriedenheit.

Mehrwert:

- Automatisierte Sturm- und Sicherheitswarnungen für das gesamte Resort
- Energieeffizientes Gebäudemanagement durch bedarfsorientierte Steuerung
- Digitale Check-in/Check-out-Prozesse über RFID-Zugangssysteme
- Vereinfachte Wartungskoordination und Zustandsüberwachung in Echtzeit
- Skalierbarkeit auf größere Ferienanlagen durch Cloud-Integration

Insgesamt zeigt das *Smart Beach Resort*-Projekt, dass ein durchdachtes Zusammenspiel von Sensorik, Cloud-Infrastruktur und Automatisierungslogik eine robuste IoT-Architektur ermöglicht, die sowohl für die Forschung als auch für den praktischen Einsatz in der Gebäudeverwaltung relevant ist. Es dient als Blaupause für verteilte Smart-Home-Systeme und demonstriert, dass Cloud-fähige IoT-Frameworks wie HiveMQ und Node-RED die Grundlage für zukunftsfähige, modulare und wartbare Smart-Environment-Lösungen bilden.

2 Use Case View

2.1 Einzel-Use Cases

Haus 1 – Wetter & Sturm-Überwachung
Sensoren: Temperatur-, Feuchte-, Dampf-, Gas-, Bewegungsmelder.
Aktoren: Buzzer, LCD-Display.
Funktion: Frühwarnung bei Sturm oder Gas, Korrosionsschutz, Sicherheitsalarm.

Haus 2 – Gäste-Komfort & Sicherheit
Sensorik/Aktorik: RFID, Buttons, LCD, RGB-LED, Servo, Yellow-LED.
Funktion: kontaktloser Zugang, Service-Anfragen, automatische Rollos, Beleuchtungssteuerung.

Haus 3 – Energie & Wartung
Sensorik/Aktorik: Motor/Lüfter, Gas-Sensor, Buzzer, LCD, Button.
Funktion: Lüftungssteuerung, Wartungsalarm, Reinigungssignalisierung.

2.2 Gesamt-Use Case – Smart Beach Resort Management

Alle drei Häuser bilden ein gemeinsames IoT-System:

- Zentrales Dashboard (Node-RED) zur Gesamtsteuerung
- Automatische Fenster- und Lüftungssteuerung bei Sturm
- Synchronisierte Reinigungs- und Energieprozesse
- Bluetooth für lokale Haus-zu-Haus-Kommunikation
- Internet-Anbindung über Handy-Hotspots

3 Logical View

Die logische Architektur besteht aus vier Schichten:

(1) **Device Layer:** ESP32-Controller mit Sensor- und Aktorschnittstellen.
(2) **Communication Layer:** MQTT über HiveMQ Cloud.
(3) **Logic Layer:** Node-RED-Flows für Ereignisverarbeitung.
(4) **Application Layer:** Dashboard und mobile App.

Kernkomponenten:

- `SensorManager` – Erfassung und Kalibrierung
- `MQTTClient` – Verbindungsaufbau, Publish/Subscribe
- `RuleEngine` – Regelbasierte Logik (z. B. Sturm)
- `DisplayManager` – Statusausgabe
- `ServiceInterface` – Kommunikation mit App

Fig. 1. Use-Case-Diagramm mit Akteuren (Gast, Host, Reinigungspersonal)

Fig. 2. Abbildung 2: Klassendiagramm mit MQTT-Themen und Zustandsbeziehungen

4 Development View

Das System ist modular entwickelt. Jede Entwickler*in implementiert ein Haus-Modul mit identischer Kommunikationsschnittstelle.

Software-Management:

- Versionskontrolle über GitHub
- Arduino IDE 2.x für ESP32
- Node-RED-Flows als JSON-Dateien

```
/house1/firmware/
/house2/firmware/
/house3/firmware/
/node-red/flows.json
/docs/architecture/
```

Kommunikation: MQTT-Topics wie `resort/house1/telemetry/temp` oder `resort/all/cmd/storm` steuern alle Interaktionen. QoS 1 für sicherheitsrelevante Daten, Retained Status für Systemzustände.

Fig. 3. Komponentendiagramm (3 ESP32 + Node-RED + HiveMQ)

5 Process View

5.1 Beispielprozess – Sturmwarnung

(1) Haus 1 erkennt erhöhte Feuchtigkeit und Temperaturabfall.
(2) MQTT-Nachricht `resort/all/cmd/storm_alert=true`.
(3) Node-RED sendet Befehle:

- `resort/house2/cmd/shutters=DOWN`
- `resort/house3/cmd/ventilation=OFF`

(4) Häuser bestätigen Status über `resort/house/status`.

5.2 Kommunikationsparameter

- Keep-Alive 60 s, Heartbeat 30 s
- QoS 1 für kritische Events
- Nachrichtengröße ca. 150 Byte
- Datenrate ca. 6–12 kB pro Sensor und Stunde

5.3 Beispielprozess – Gästewechsel

(1) RFID-Check-out → Event `checkout`.
(2) Node-RED startet Reinigungstimer für Haus 3.
(3) Reinigung meldet Button-Event `cleaned`.
(4) Node-RED reaktiviert Lüftung und bereitet Check-in vor.

Fig. 4. Sequenzdiagramm für Sturm- und Check-out-Prozesse

6 Physical View

Hardware-Mapping:

Komponente	Beschreibung
ESP32 DevBoard	Controller mit WiFi und Bluetooth
Sensoren	DHT11, Steam, Gas, PIR, RFID, Buttons
Aktoren	Buzzer, Servo, LED, Lüfter
Gateway	Smartphone-Hotspot pro Haus
Broker	HiveMQ Cloud (Serverless)
Controller	Node-RED auf privatem Server

Netzstruktur:

- Jedes Haus → eigenes IoT-Node mit Hotspot-Anbindung.
- Kommunikation über MQTT zu HiveMQ Cloud.
- Node-RED abonniert `resort/#` für zentrale Steuerung.

Fig. 5. Deployment-Diagramm (ESP32 – HiveMQ – Node-RED – App

7 Schlussfolgerungen

Das Projekt zeigt, dass ein verteiltes IoT-System mit mobilen Hotspots zuverlässig funktioniert, wenn ein Cloud-basiertes Kommunikationsframework eingesetzt wird. HiveMQ + Node-RED bietet eine modulare, skalierbare Architektur.

Ergebnisse:

- Sichere, stabile Kommunikation über MQTT Cloud-Broker
- Automatisierte Prozesssteuerung per Node-RED
- Cloud- und On-Premise-Betrieb möglich
- Dezentrale Entwicklung ohne physische Vernetzung

Kosten:

- Hardware: Anschaffung ca. 210 € (3 × KS5009-Kits)
- Cloud-Infrastruktur: 0 €/Monat bzw. Serverkosten bei Neuanmietung

Key Take-Aways:

- MQTT ist das stabilste Kommunikationsprotokoll für IoT.
- Cloud-Broker beseitigen NAT-Probleme bei Hotspots.
- Node-RED vereinfacht Regellogik und Visualisierung.
- System ist erweiterbar und demofähig unter realen Bedingungen.

References