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

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* zeigt, wie IoT-Technologien ein vernetztes Ferienhaus-Ökosystem bilden können. Drei intelligente Ferienhäuser (hier in Form von keyestudio-Häusern dargestellt) an der niederländischen Nordseeküste werden über Sensorik und Cloud-Kommunikation gesteuert. Besondere Herausforderungen der Umgebung sind hohe Luftfeuchtigkeit, Salz, Sand und Sturm.

Ziel ist es, drei Einzel-Use-Cases zu entwickeln und zu einem Gesamt-Use-Case zu integrieren. Die Lösung adressiert Betreiber, Facility-Manager und Ferienhausvermieter, die ein skalierbares, wartungsarmes System benötigen.

Mehrwert:

- Automatisierte Sturm- und Sicherheitswarnungen
- Energieeffizientes Gebäudemanagement
- Digitale Check-in/Check-out-Abläufe
- Geringe Betriebskosten durch Fernsteuerung

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.

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

(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

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/
```

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

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