

Spezifikation für King-Arthur Outdoor Adventure

Inhaltsverzeichnis

1. Einführung	1
2. Excalibur im Stein	2
2.1. Betrieb	4
2.2. MQTT-Anbindung	5
3. Thron zur Krönung	8
3.1. LoRa Kommunikation	8
3.2. MQTT-Bridge	9
4. Webserver	11

1. Einführung

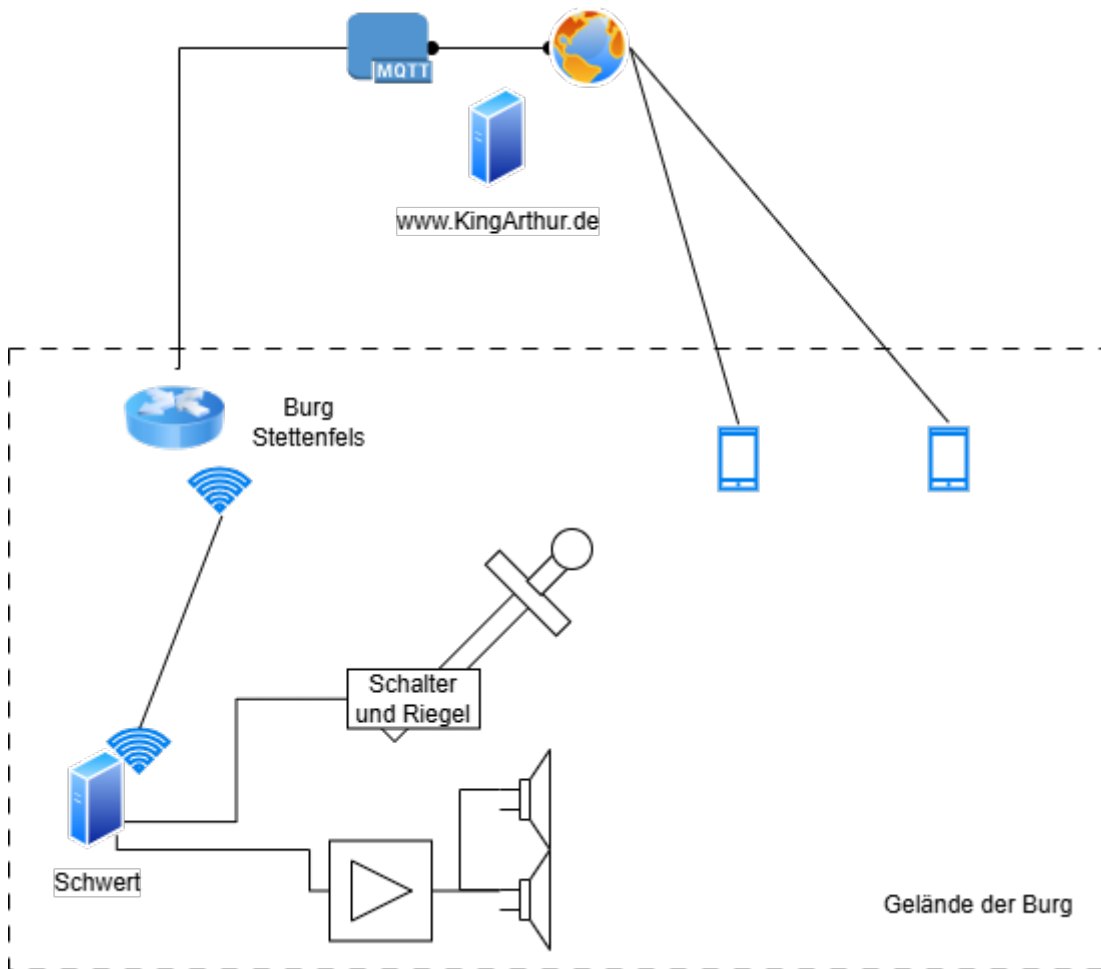
Im Folgenden wird Hardware-Seite der Lösung für die **Outdoor-Tour auf der Burg Stettenfels** beschrieben. Die Tour besteht aus mehreren Spielen von denen zwei mit vernetzten Rechnern realisiert werden.

1. Excalibur im Stein
2. Thron zur Krönung

Ergänzend zu den Hardware-Lösungen wird eine Infrastruktur bereitgestellt, welche - die Webseite für das Spiel hosted und - die Vernetzung zwischen den Geräten realisiert.

2. Excalibur im Stein

Die folgende Abbildung zeigt die angebotene Lösung im Überblick.



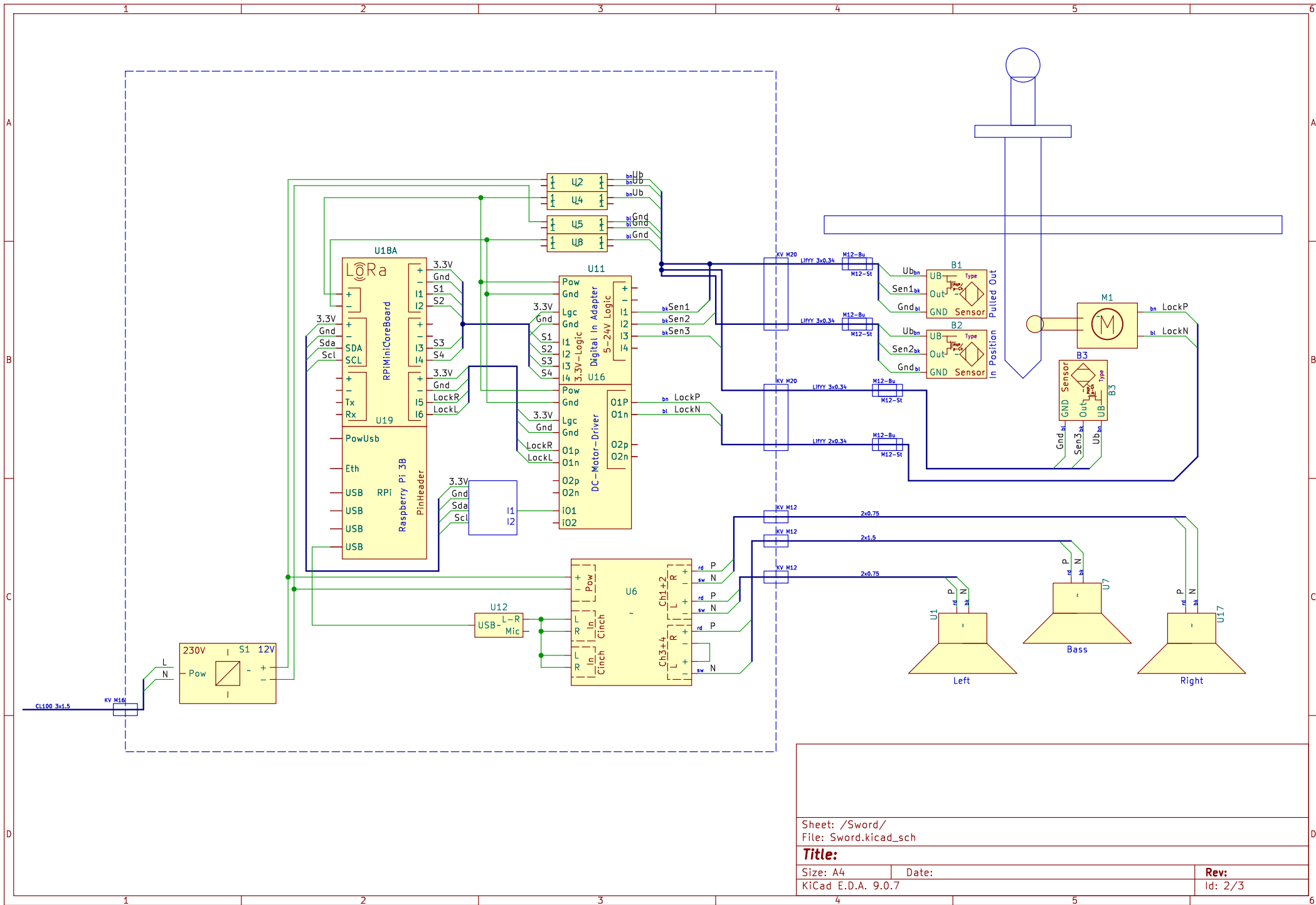
Die Lösung besteht aus einer Hardware und einer Webanwendung mit entsprechender Infrastruktur.

Die Elektronik verfügt über folgende Funktionen.

1. Sensor um zu erfassen, ob das Schwert entfernt wurde
2. Sensor um zu erfassen, ob das Schwert vollständig eingesteckt wurde
3. Sensor zum Erfassen ob an dem Schwert gezogen wird
4. Bedienung eines elektrischen Riegels für das Schwert
5. Abspielen von Sounds und Musik

Die Elektronik ist direkt im W-LAN der Burg Stettenfels integriert. Die Elektronik verfügt über ein LoRa Funkmodul um weitere Endgeräte auf dem Gelände anzubinden, die außerhalb des W-LAN-Bereichs positioniert sind. Die Kommunikation wird über MQTT abgearbeitet. Der MQTT-Broker wird auf einem privaten Server gehostet.

Die folgende Seite zeigt den aktuellen Schaltplan der Elektronik

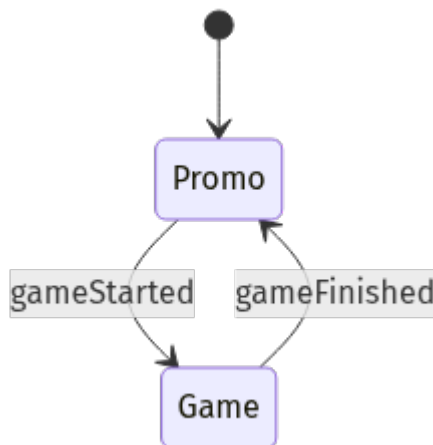


2.1. Betrieb

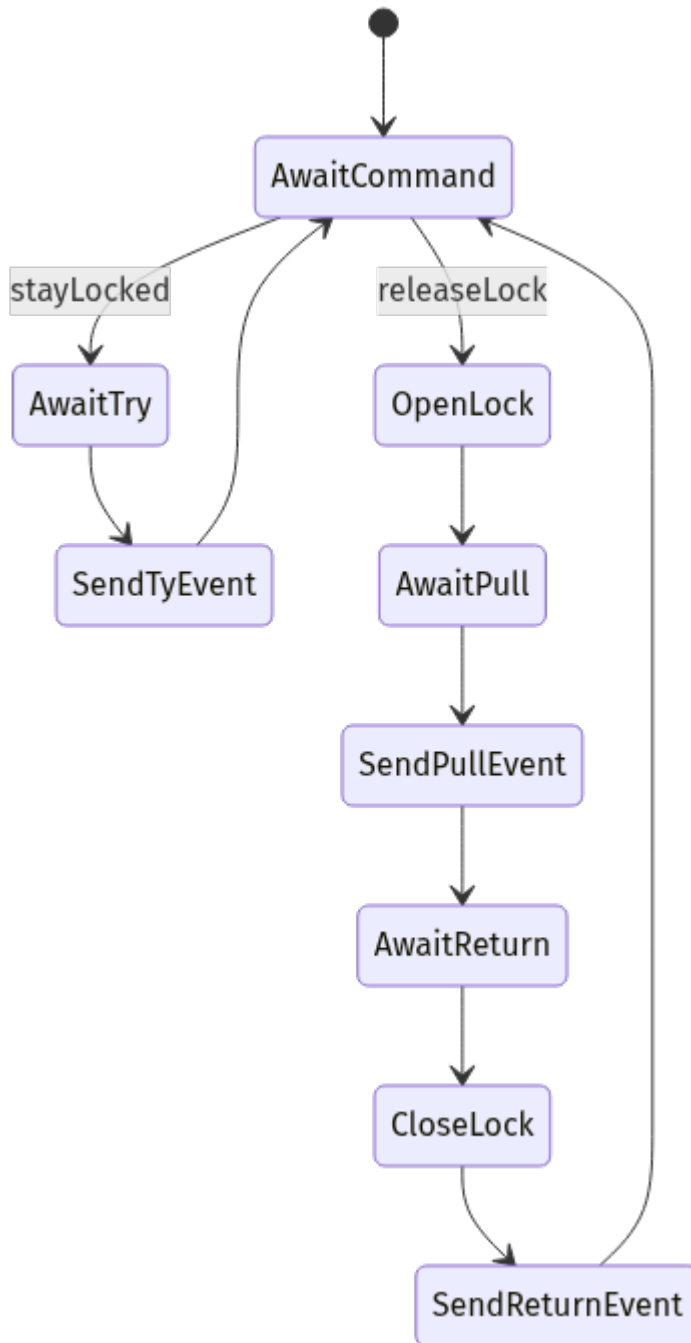
Bis zur Fertigstellung der weiteren Spiele läuft die Station im **PromoOnlyMode**. Dabei gibt die Station die Werbung für die Outdoor-Tour aus, wenn an dem Schwert gezogen wird und reagiert auf die Kommandos die zur Wartung und Konfiguration vorgesehen sind.

Im **GameMode** ist das Schwert im Spielbetrieb und wartet auf die Kommandos aus der App. Solange kein Spiel läuft darf das Schwert Werbung ausgeben.

Der Zustand des Spiels wird mit dem **set**-Topic übertragen



Die folgende Abbildung zeigt den Spielbetrieb einer Runde



Die Kommandos werden mit dem **set**-Topic übertragen Die Rückmeldungen erfolgen über das **event**-Topic

Soll ein Spieler des Schwert vergeblich zeihen wird das Zug-Ereignis abgewartet und der App gemeldet

Soll ein Spieler das Schwert ziehen dürfen, öffnet die Station den Riegel und wartet bis die Sensoren das Schwert nicht mehr erkennen. Das Ereignis wird gemeldet, und die entsprechenden Sound abgespielt. Danach wartet die Station bis beide Sensoren das Schwert wieder erkennen um den Riegel wieder zu schließen. Die Verriegelung wird erneut gemeldet.

2.2. MQTT-Anbindung

Die Befehle werden von einem externen Server entgegengenommen, welcher auch die Web-

Masken für das Spiel auf dem Smartphone hosted.

Die Station bietet folgende MQTT-Topics zur Kommunikation Jedes Topic hat einen Prefix, der das Spiel eindeutig kennzeichnet. Die Topics werden im folgenden definiert

Prefix

outdoor/king-arthur/sword Prefix ist Teil jedes Topics

2.2.1. Ausgehende Meldungen

Version

- Versionsinformation
- einmalig bei Start der App
- Topic: **<prefix>/ver**
- Message: Codiert als **V<Major>.<Minor>.<Fix>** z.B. **V1.2.3**

Status

- Zustände der Station und der Sensoren
- zyklisch
- Topic: **<prefix>/stat**
- Codiert als als Komma-separierte Key-Value-Liste
- Message: **M:<Mode>,sE:<Val>,sS:<Val>,sP:<Val>,lE:<Val>** z.B. **Mode:Prod, sE:0,sS:0,sP:0,lE:0**
 - M ist der Modus in dem die Station sich befindet
 - Mögliche Werte für Mode: 1=Test, 2=Promo, 3=Game
 - Die Bedeutung der Codes sind
 - sE: Sensor Endposition
 - sS: Sensor StartPosition
 - sP: Sensor für Pull
 - lE: Riegel-Position
 - Mögliche Werte für Val: 0=false, 1=true
- Die Informationen sind für die Wartungsansicht vorgesehen

Event

- Ereignisse an der Station
- interrupt
- Topic: **<prefix>/event**
- Message: **<EventTyp>**
- Mögliche Events sind
 - **pull**: Es wurde an dem Schwert gezogen
 - **pulled-out**: Das Schwert wurde raus gezogen

- **put-in**: Das Schwert wurde zurück gelegt
- **ceremonyDone**: Zeremonie ist abgeschlossen
- Die Nachricht wird nur im Game-Modus gesendet

2.2.2. Eingehende Meldungen

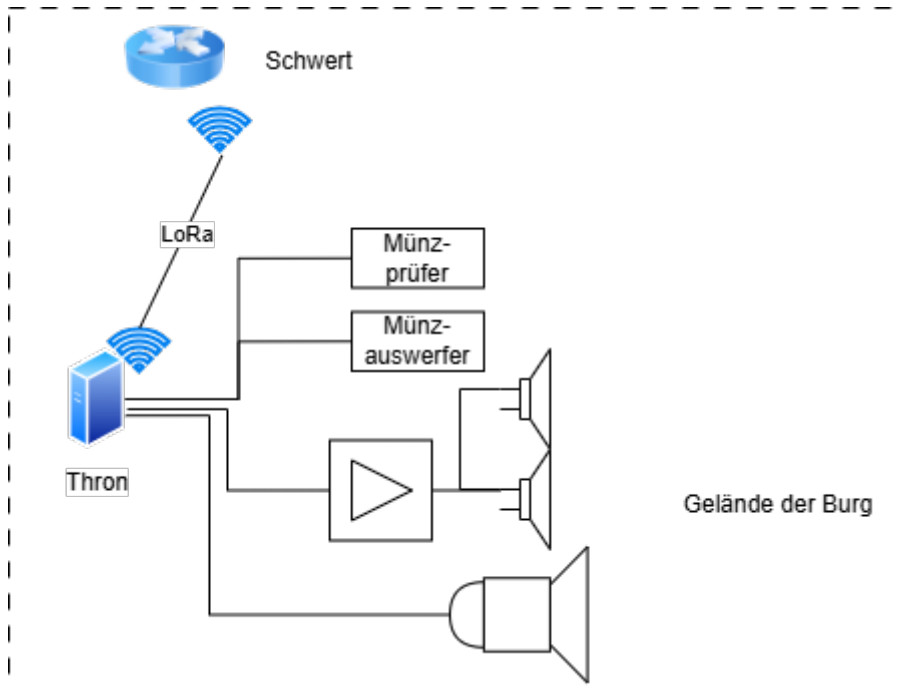
Eingehende Steuerbefehle werden nicht quittiert. Bei der Konfiguration ist **QoS=2** zu wählen, damit sichergestellt ist, dass die Nachricht genau einmal ankommt.

Control

- Steuerbefehl an die Station für den Betrieb und für Wartungszwecke
- interrupt
- Topic: **<prefix>/set**
- Message: **<CommandTyp>**
- Mögliche Kommandos für den Regelbetrieb sind
 - **stayLocked**: Beim nächsten Zugversuch bleibt das Schwert verriegelt
 - **releaseLock**: Schwert wird entriegelt, beim nächsten Zugversuch kann es gezogen werden.
 - **gameStarted**: Wird ein Spiel gestartet verhält sich das Schwert ruhig und wartet auf weitere Befehle
 - **gameFinished**: Wird das Spiel beendet, kann das Schwert wieder den Werbetext spielen.
- Mögliche Kommandos für die Konfiguration
 - **volume:<Val>**: Lautstärke der Station, Wertebereich = **0..100**
 - **promoDelay:<Val>**: Wartezeit nach PromoSound, Wertebereich = **30 .. 300**
- Mögliche Kommandos für die Wartung sind
 - **openLock**: Schwert-Riegel öffnen
 - **closeLock**: Schwert-Riegel Schließen
 - **playSound:<Val>**: Abspielen einer Sounddatei Mögliche Sounds: **Promo, Deny, Chosen**

3. Thron zur Krönung

Die folgende Abbildung zeigt die angebotene Lösung im Überblick.



Die Lösung besteht aus einer Hardware und wird in die Infrastruktur mit integriert.

Die Elektronik verfügt über folgende Funktionen.

1. Freigabe des Münzprüfers
2. Annehmen von Spielmünzen
3. Ausgabe von Souvenir-Münzen
4. Abspielen von Sounds und Musik
5. Steuerung von Licht

3.1. LoRa Kommunikation

Die Befehle werden über LoRa entgegen genommen die über den Schwertaufbau weitergeleitet werden. Statusmeldungen werden ebenfalls über LoRa zurückgemeldet

Befehle

GetVersion: SetState: Active/Passive SetVolume: 0..100

Rückmeldungen

CoinAccepted: Count Live-Signal

Fehlermeldungen

Return-Coins-Empty System-Error

Die Station meldet zyklisch ein Lebenszeichen. Auf Kommando wird der Münzprüfer aktiv

geschaltet und mechanisch freigegeben (Vandalismus-Schutz) Die erfassten Münzen werden zurückgemeldet und können mit der Spieler-Anzahl plausibilisiert werden Etwaige Fehlermeldungen von den Endgeräten werden rückgemeldet

3.2. MQTT-Bridge

Die Protokoll-Brücke welche die LoRa-Befehle auf MQTT übersetzt bietet folgende Kommandos an

Jedes Topic hat einen Prefix, der das Spiel eindeutig kennzeichnet. Die Topics werden im folgenden definiert

Prefix

`outdoor/king-arthur/throne` Prefix ist Teil jedes Topics

3.2.1. Ausgehende Meldungen

Version

- Versionsinformation
- einmalig bei Start der App
- Topic: `<prefix>/ver`
- Message: Codiert als `V<Major>.<Minor>.<Fix>` z.B. `V1.2.3`

Status

- Zustände der Station und der Sensoren
- zyklisch
- Topic: `<prefix>/stat`
- Codiert als als Komma-separierte Key-Value-Liste
- Message: `cA:<Val>,cD:<Val>,cE:<Val>` z.B. `cA:42,cD:0,cE:0`
 - Die Bedeutung der Codes sind
 - cA: Akzeptierte Münzen
 - cD: Fehlversuche mit Münzen
 - cE: Münzschacht ist offen
 - Mögliche Werte für Val: 0=false, 1=true
- Die Informationen sind für die Wartungsansicht vorgesehen

Event

- Ereignisse an der Station
- interrupt
- Topic: `<prefix>/event`
- Message: `<EventTyp>`
- Mögliche Events sind
 - `coin`: Es wurde eine Münze akzeptiert

- **ceremonyDone**: Krönung fertig

3.2.2. Eingehende Meldungen

Eingehende Steuerbefehle werden nicht quittiert. Bei der Konfiguration ist **QoS=2** zu wählen, damit sichergestellt ist, dass die Nachricht genau einmal ankommt.

Control

- Steuerbefehl an die Station für den Betrieb und für Wartungszwecke
- interrupt
- Topic: **<prefix>/set**
- Message: **<CommandTyp>**
- Mögliche Kommandos für den Regelbetrieb sind
 - **closeCoinSlot**: Münzeinwurf schließen (Vandalismus-Schutz)
 - **openCoinSlot**: Münzeinwurf freigeben
- Mögliche Kommandos für die Konfiguration
 - **volume:<Val>**: Lautstärke der Station, Wertebereich = **0..100**
- Mögliche Kommandos für die Wartung sind
 - **playSound:<Val>**: Abspielen einer Sounddatei Mögliche Sounds: **Ceremony**

4. Webserver

Ein im Internet frei erreichbare Maschine erfüllt mehrere Aufgaben für das Projekt.

1. Hosten der App für das Spiel
2. Betrieb des MQTT Brokers

Für die Erfüllung der Aufgabe sind folgende Dienste auf dem Server installiert. . Docker . Reverse Proxy (im Docker) . MQTT-Broker (im Docker) . Web-App (im Docker) . Repository (im Docker) . LogService (im Docker)

Die wesentlichen Dienste sind ausschließlich Docker Container. Die Webseite wird ebenfalls als Docker-Container ausgeliefert. Für die Auslieferung wird ein Repository Bereit gestellt.

Die Kommunikation zwischen App und den vernetzten Geräte läuft über MQTT. Die Kommunikation wird verschlüsselt und wird über eine einfache User/Password Kombination geschützt.

Die Authentifizierung erfolgt über eine User/Passwort-Kombination

```
BrokerUrl:  "mqtt://nerdyssey.de:8883";  
MqttUser:   "exodususer";  
MqttPassword: "will-be-disclosed-later";
```