IT22A WPF Projektarbeit

ESCAPEROOM WEBSITE

VERSION 1.1.5

04.06.2024

Vorgestellt von
Bohn Matthias

Inhaltsverzeichnis

Escaperoom website

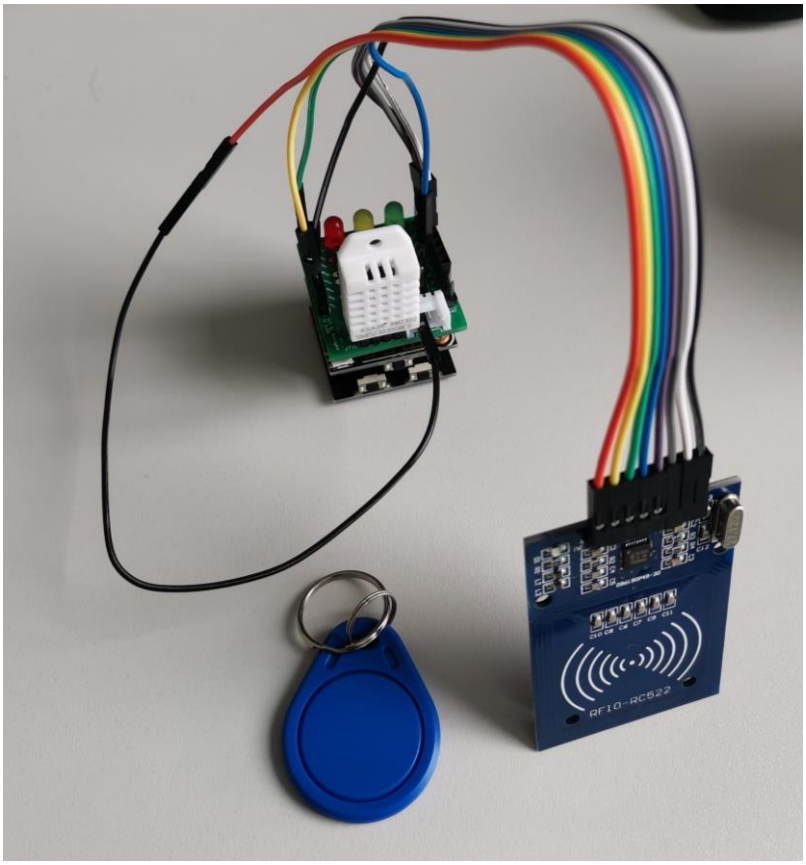
Kurzbeschreibung des Produktes

Dieses Projekt besteht aus einem Escape Room, der auf einer Website dargestellt wird und von einem Raspberry Pi gehostet wird. Der Raspberry Pi fungiert als Webserver mit Apache 2 und betreibt zusätzlich einen MQTT-Server. Ein ESP8266 steuert mehrere Sensoren und Aktoren, darunter LEDs, Taster, einen LDR-Sensor, ein RFID-Chip Lesegerät und einen DHT-22 Sensor.

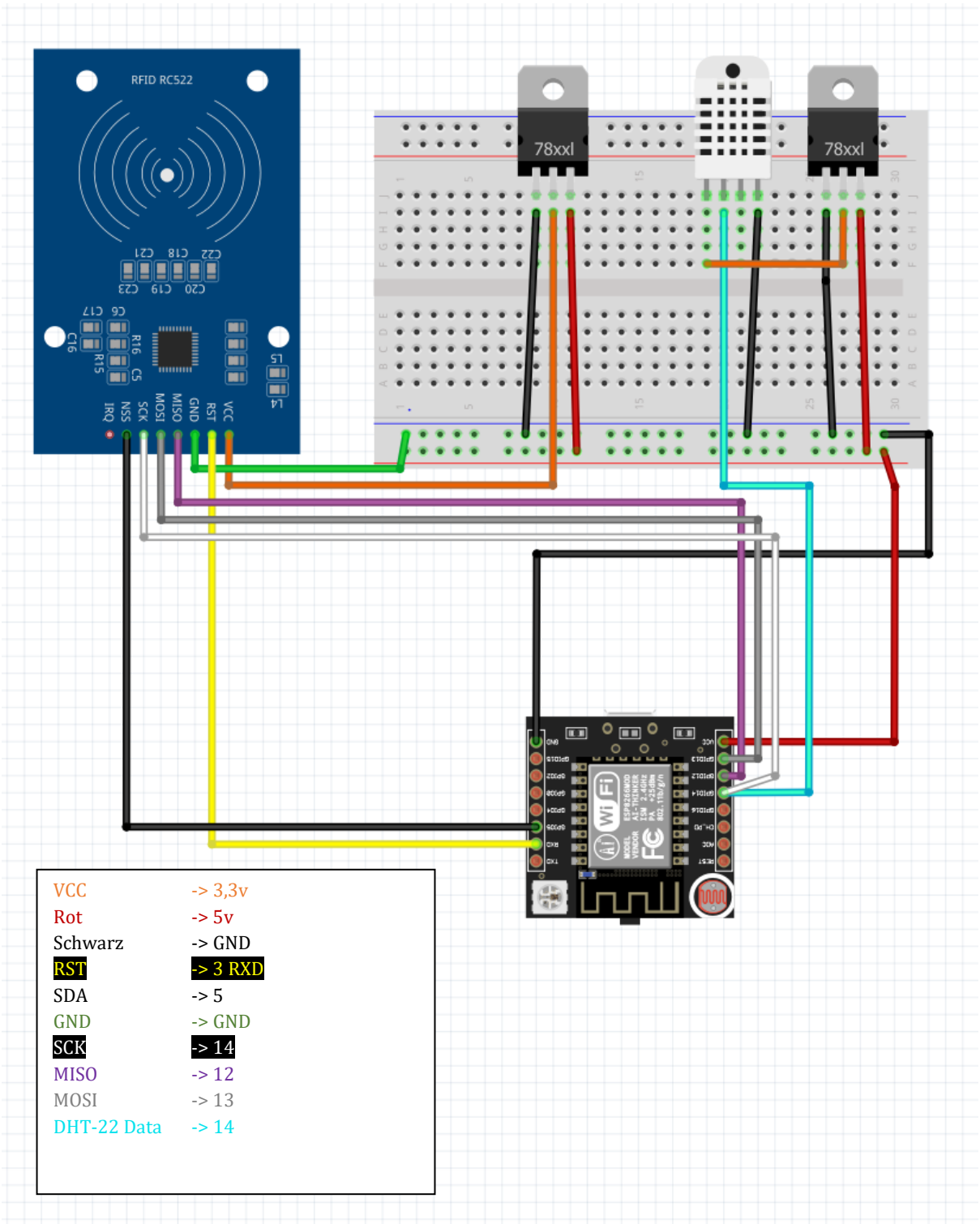
Die Website simuliert ein Haus mit drei Räumen und einem Flur. Der Spieler muss verschiedene Rätsel lösen, die durch die Sensoren und Aktoren gesteuert werden, um von einem Raum zum nächsten zu gelangen und um schlussendlich wieder aus dem Haus zu entkommen. Die Kommunikation zwischen den Sensoren/Aktoren und der Website erfolgt über das MQTT-Protokoll.

Komponenten und deren Zusammenspiel visualisieren

Realbild:

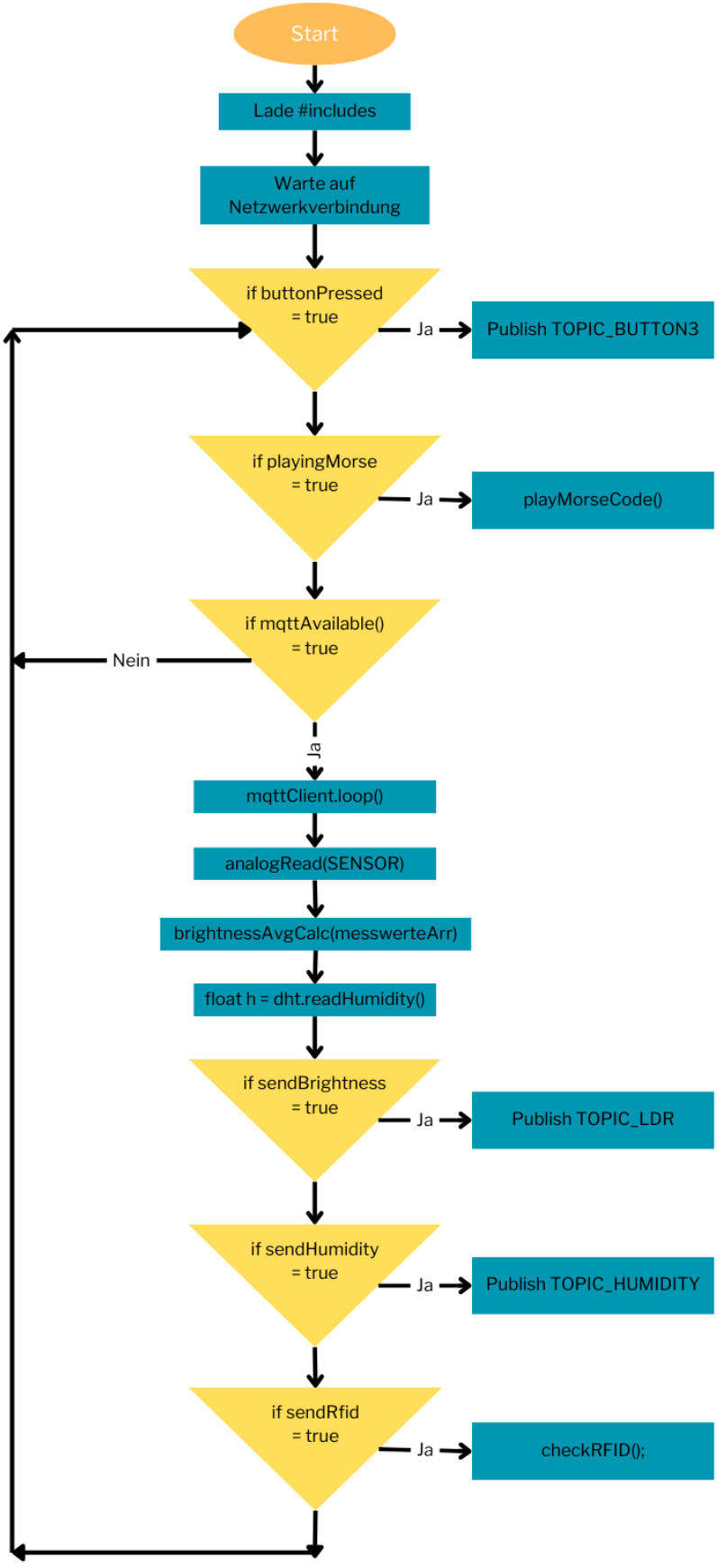


Schaltplan:

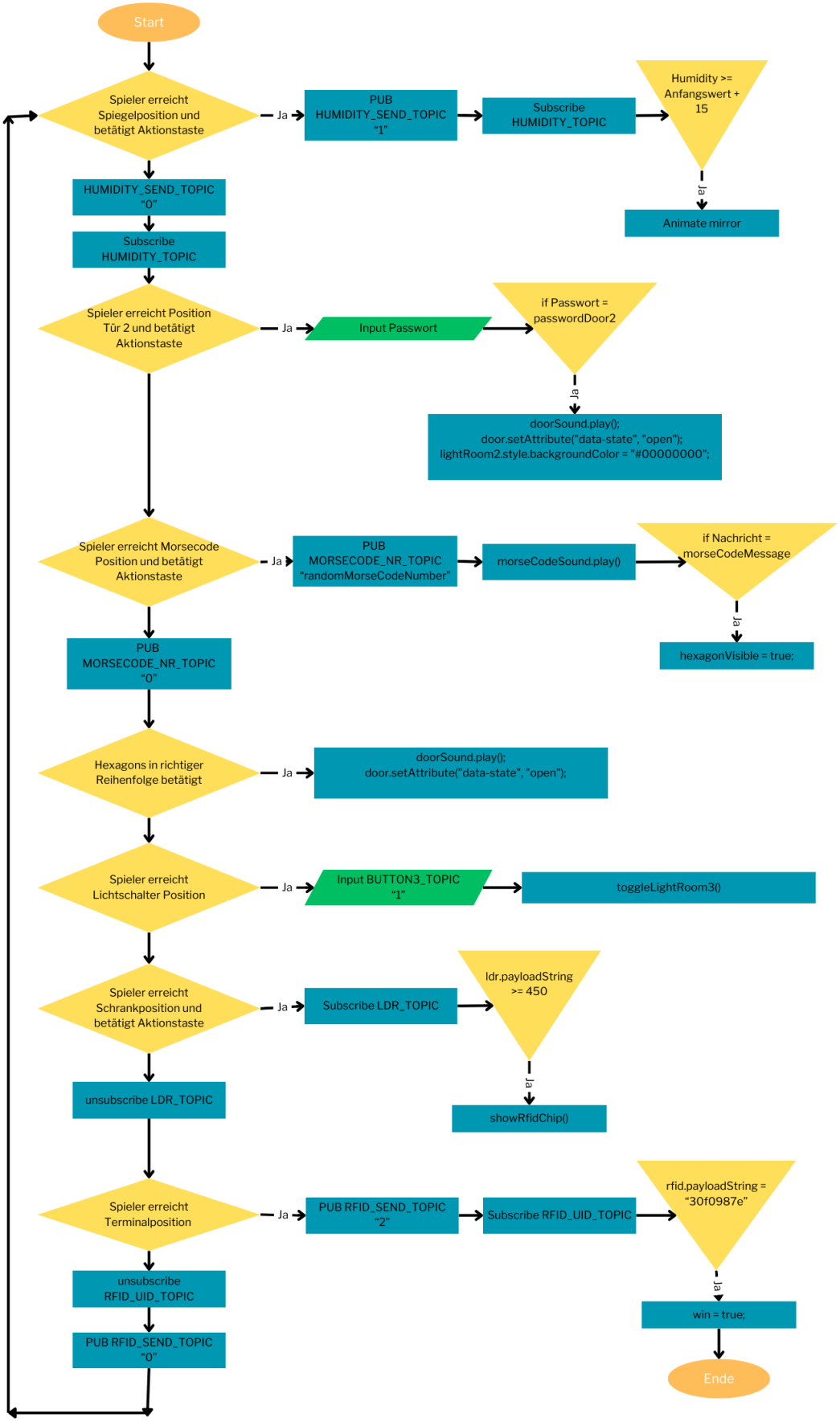


VCC	-> 3,3v
Rot	-> 5v
Schwarz	-> GND
RST	-> 3 RXD
SDA	-> 5
GND	-> GND
SCK	-> 14
MISO	-> 12
MOSI	-> 13
DHT-22 Data	-> 14

PAP ESP:


Start
Lade #includes
Warte auf Netzwerkverbindung
if buttonPressed = true
Ja
Publish TOPIC_BUTTON3
if playingMorse = true
Ja
playMorseCode()
if mqttAvailable() = true
Nein
Ja
mqttClient.loop()
analogRead(SENSOR)
brightnessAvgCalc(messwerteArr)
float h = dht.readHumidity()
if sendBrightness = true
Ja
Publish TOPIC_LDR
if sendHumidity = true
Ja
Publish TOPIC_HUMIDITY
if sendRfid = true
Ja
checkRFID();

PAP Javascript / HTML:



Beschreibung der Kommunikationsprotokolle

MQTT-Protokoll:

- Anfrage und Antwort:
  - Der ESP8266 sendet Sensordaten (z.B. Luftfeuchtigkeit, Taster Zustand) an den MQTT-Server auf dem Raspberry Pi.
  - Die Website abonniert diese Daten und verwendet sie zur Steuerung der Rätsel.
- Nachrichtenaufbau:
  - MQTT-Nachrichten bestehen aus einem Topic und einer Payload.
  - Topics sind hierarchisch strukturiert, z.B. Sensor/dht22 für den DHT-22 Sensor.
  - Payload enthält die eigentlichen Daten, z.B. { "humidity": 60 }.
- Technische Eigenschaften:
  - MQTT läuft über TCP/IP.
  - Standard-Port für MQTT ist 1883.
- QoS (Quality of Service) Stufen:
  - QoS 0: Zustellung nach besten Möglichkeiten
  - QoS 1: Mindestens einmalige Zustellung
  - QoS 2: Genau einmalige Zustellung

http-Protokoll:

- Anfrage und Antwort:
  - Der Browser des Benutzers sendet eine http-Anfrage an den Webserver (Apache 2) auf dem Raspberry Pi.
  - Der Webserver antwortet mit den HTML-, CSS-, und JavaScript-Dateien, die die Website darstellen.
- Nachrichtenaufbau:
  - http-Anfragen bestehen aus einem Request-Line (z.B. GET /index.html http/1.1), Header-Feldern und optional einer Nachricht.
  - HTTP-Antworten bestehen aus einer Statuszeile (z.B. http/1.1 200 OK), Header-Feldern und dem eigentlichen HTML-Inhalt.
- Technische Eigenschaften:
  - http läuft über TCP/IP, normalerweise auf Port 80.
- Verbindungsaufbau:
  - Client-Server-Modell, bei dem der Client (Browser) Anfragen stellt und der Server (Raspberry Pi) antwortet.

Tabelle der wichtigsten Protokollparameter:

Protokoll	Port	Transport Layer	Sicherheit
MQTT	1883	TCP/IP	Optional
Http	80	TCP/IP	Keine

Installation von git unter Linux

Führe die folgenden Befehle aus, um git auf einem Linux-System zu installieren:

sudo apt update

sudo apt install git

MQTT auf dem Raspberry Pi installieren

Installiere das MQTT-Paket (Mosquitto) auf dem Raspberry Pi:

sudo apt update

sudo apt install mosquitto mosquitto-clients

Starte den Mosquitto-Dienst und stelle sicher, dass er beim Systemstart aktiviert ist:

sudo systemctl start mosquitto

sudo systemctl enable mosquitto

Apache2-Konfiguration

Installation von Apache2

sudo apt update

sudo apt install apache2

Füge die folgende Konfiguration zur /etc/apache2/sites-available/000-default.conf hinzu, um den Zugriff auf das Verzeichnis der CGI-Skripte zu ermöglichen:

```
<VirtualHost *:80>
    # ...

    DocumentRoot /home/mqtt/EscapeRoom

    # ...

    <Directory "/home/mqtt/">
        AllowOverride None
        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Require all granted
    </Directory>

    ProxyPassMatch          "/ws$"  ws://127.0.0.1:9001

</VirtualHost>
```

Nach den Änderungen muss der Apache Server neu gestartet werden:

service apache2 restart

Dateien von Git herunterladen

Lade die Watch-Pulse Dateien in deinen Projekt-Ordner herunter

cd /home/mqtt/

sudo git clone https://github.com/lilbone/EscapeRoom.git
Anschließend musst du noch den Benutzer anpassen:

sudo chmod -R +x /home/mqtt/EscapeRoom

Arduino-Bibliotheken installieren

Um den Sketch escapeRoomSketch.ino auf deinem ESP8266 auszuführen, musst du die folgenden Arduino-Bibliotheken installieren:

- DHT Sensor Library von Adafruit:
  - Suche in der Bibliotheksverwaltung nach "DHT sensor library" und installiere sie.
- ESP8266WiFi:
  - Diese Bibliothek ist normalerweise mit der Installation des ESP8266-Pakets in der Arduino-IDE enthalten.
- PubSubClient:
  - Suche in der Bibliotheksverwaltung nach "PubSubClient" und installiere sie.
- MFRC522:
  - Suche in der Bibliotheksverwaltung nach "MFRC522" und installiere sie.
- Streaming:
  - Suche in der Bibliotheksverwaltung nach "Streaming" und installiere sie.

ESP8266 vorbereiten

- Taster / LED / Senor-Erweiterung aufstecken
- DHT-22 mit Sensor-Schnittstelle verbinden
- RFID Chip-Reader wie folgt verbinden:
  - Orange 3,3v -> 3,3v
  - Gelb RST -> 3 RXD
  - Schwarz SDA -> 5
  - Grün GND -> GND
  - Weiß SCK -> 14
  - Lila MISO -> 12
  - Grau MOSI -> 13

Beschreibung eines besonderen Teils der Umsetzung

RFID-Chip-Reader und Verwendung der Sensoren zur Lösung von Rätseln:

Ein herausragendes Merkmal dieses Projekts ist die Implementierung des RFID-Chip-Readers. Der RFID-Reader am ESP8266 liest den RFID-Chip und sendet die Daten über MQTT an die Website. Sobald der Chip korrekt erkannt wird, kann der Spieler die letzte Tür im Escape Room öffnen.

Darüber hinaus wurde großer Wert darauf gelegt, die Sensoren und Aktoren nicht nur zur Datenerfassung, sondern als aktive Bestandteile der Rätsel zu nutzen:

DHT-22 Sensor: Erkennt Atemfeuchtigkeit, um ein beschlagenes Spiegelbild und ein verborgenes Passwort anzuzeigen.

LED und Ton: Werden verwendet, um Morsecode zu signalisieren, den die Spieler entschlüsseln müssen.

Taster und LDR-Sensor: Interaktionen mit diesen Komponenten schalten Licht ein oder machen den RFID-Chip sichtbar.

Diese kreative Nutzung der Komponenten sorgt für ein interaktives und immersives Erlebnis im Escape Room.

Programm-code ESP

```
/* ####################################################################
 Filename       : escapeRoomSketch.ino
 Author         : Bohn Matthias
 Date           : 26.05.2024
#################################################################### */
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <MFRC522.h>
#include "Streaming.h"
#include "wifi.h"
#include "params.h"
#include "functions.h"

DHT dht(DHT_PIN, DHT_TYPE);  // Initialisierung des DHT-Sensors

void setup() {
  Serial.begin(9600);
  Serial << endl
         << "Start Escape-Room" << endl;

  Serial << F("Version: 1.1.5") << endl
         << F("Build: ") << F(__TIME__) << F("  ") << F(__DATE__) << endl
         << F(__FILE__) << endl;

  pinMode(LED_YELLOW, OUTPUT);
  pinMode(TASTER_3, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(TASTER_3), handleButtonPress,
FALLING);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, psk);
  // wait for AP association
  Serial << "Warte auf Verbindung..." << endl;

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial << ".";
  }

  Serial << endl
         << "Mit " << ssid << "verbunden" << endl
         << endl
         << "IP-Addresse: " << WiFi.localIP() << endl
         << endl;

  //DHT Sensor
```

```
  pinMode(DHT_POWER, OUTPUT);     // Konfiguriere den DHT-Stromversorgungspin
als Ausgang
  digitalWrite(DHT_POWER, HIGH);  // Schalte die Stromversorgung für den DHT-
Sensor ein
  dht.begin();                    // Initialisierung des DHT-Sensors

  mqttClient.setServer(MQTT_BROKER, PORT);

  mqttAvailable();
  mqttClient.setCallback(callback);

  Serial << "Setup abgeschlossen" << endl;
}


void loop() {
  int messwerteArr[ANZAHL_MESSWERTE];
  int brightnessAvg = 0;
  unsigned long currentMillis = millis();  // Aktuelle Zeit abrufen

  if (buttonPressed) {
    buttonPressed = false;
    publishData(TOPIC_BUTTON3, "1");
  }

  // Wird kontinuierlich aufgerufen und prüft den Morse-Code-Status
  if (playingMorse) {
    playMorseCode();
  }

  if (currentMillis - previousMqttMillis >= 200) {
    previousMqttMillis = currentMillis;
    if (mqttAvailable()) {

      unsigned long start = millis();
      do {
        mqttClient.loop();  // Verarbeite den Eingangs-Nachrichtenstapel
      } while (millis() - start < 200);

      if (currentMillis - previousMillis >= interval) {
        for (int i = 0; i < ANZAHL_MESSWERTE; i++) {  // Messwerte erfassen
          messwerteArr[i] = analogRead(SENSOR);
        }

        brightnessAvg = brightnessAvgCalc(messwerteArr);  //
Durchschnittshelligkeit berechnen

        h = dht.readHumidity();     // Luftfeuchtigkeit lesen

        if (sendBrightness) {
```

```
          if (abs(brightnessAvg - brightness) > 50) {
            brightness = brightnessAvg;
            publishData(TOPIC_LDR, String(brightnessAvg));
          }
        }

        if (sendHumidity) {
          Serial << h << ", " << humidity << endl;
          if (abs(round(h) - humidity) > 2) {
            humidity = round(h);
            publishData(TOPIC_HUMIDITY, String(humidity));
          }
        }
      }

      if (sendRfid) {
        SPI.begin();         // SPI-Bus initialisieren
        mfrc522.PCD_Init();  // MFRC522 initialisieren

        checkRFID();
      }
    }
  }
}
```

```
/* ###################################################################
 Filename      : functions.h
 Author        : Bohn Matthias
 Date          : 26.05.2024
################################################################### */
#include "WString.h"
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

#include "params.h"

// Interrupt Service Routine (ISR) für das Drücken des Buttons
void IRAM_ATTR handleButtonPress() {
  buttonPressed = true;
}

// Funktion zum Finden des Index des maximalen Werts im Array
int findMaxIdx(const int messwerteArr[]) {
  int maxValue = messwerteArr[0];               // Initialisierung des
maximalen Werts mit dem ersten Wert im Array
```

```
  int idx = 0;                                    // Index des maximalen Werts
initialisieren
  for (int i = 1; i < ANZAHL_MESSWERTE; i++) {  // Schleife durch alle
Messwerte
    if (messwerteArr[i] > maxValue) {           // Überprüfen, ob der aktuelle
Wert größer als der bisherige maximale Wert ist
      maxValue = messwerteArr[i];               // Aktualisierung des
maximalen Werts
      idx = i;                                  // Aktualisierung des Index
des maximalen Werts
    }
  }
  return idx;  // Rückgabe des Index des maximalen Werts
}

// Funktion zum Finden des Index des minimalen Werts im Array
int findMinIdx(const int messwerteArr[]) {
  int minValue = messwerteArr[0];               // Initialisierung des
minimalen Werts mit dem ersten Wert im Array
  int idx = 0;                                  // Index des minimalen Werts
initialisieren
  for (int i = 1; i < ANZAHL_MESSWERTE; i++) {  // Schleife durch alle
Messwerte
    if (messwerteArr[i] < minValue) {           // Überprüfen, ob der aktuelle
Wert kleiner als der bisherige minimale Wert ist
      minValue = messwerteArr[i];               // Aktualisierung des
minimalen Werts
      idx = i;                                  // Aktualisierung des Index
des minimalen Werts
    }
  }
  return idx;  // Rückgabe des Index des minimalen Werts
}

// Funktion zur Berechnung des Durchschnitts der Helligkeitswerte
int brightnessAvgCalc(const int messwerteArr[]) {
  int AVG = 0;                                   // Durchschnittsvariable
initialisieren
  const int idx_max = findMaxIdx(messwerteArr);  // Index des maximalen Werts
finden
  const int idx_min = findMinIdx(messwerteArr);  // Index des minimalen Werts
finden

  for (int i = 0; i < ANZAHL_MESSWERTE; i++) {  // Schleife durch alle
Messwerte
    if (i != idx_max && i != idx_min)           // Überprüfen, ob der Index
nicht dem Index des maximalen oder minimalen Werts entspricht
    {
      AVG += messwerteArr[i];  // Wert zum Durchschnitt hinzufügen
    }
```

```
  }
  return AVG / (ANZAHL_MESSWERTE - 2);  // Durchschnitt berechnen und
zurückgeben
}

// Funktion zum Veröffentlichen von Daten über MQTT
void publishData(String topic, String payload) {
  mqttClient.publish(topic.c_str(), payload.c_str());
  Serial << "PUBLISH: Topic = " << topic << " Payload = " << payload << endl;
}

// Funktion zur Überprüfung des RFID-Tags
void checkRFID() {
  if (mfrc522.PICC_IsNewCardPresent()) {  // Überprüfen, ob eine neue Karte
vorhanden ist
    if (mfrc522.PICC_ReadCardSerial()) {  // Überprüfen, ob die Kartendaten
gelesen werden können
      Serial << "Card UID: ";
      for (byte i = 0; i < mfrc522.uid.size; i++) {  // Schleife durch die
UID-Bytes
        if (mfrc522.uid.uidByte[i] < 0x10) {
          Serial << " 0";
        } else {
          Serial << " ";
        }
        Serial << _HEX(mfrc522.uid.uidByte[i]);  // Ausgabe der UID-Bytes in
Hexadezimalformat
      }
      Serial << endl;
      // Hier könntest du die UID auch an MQTT senden
      String uidString = "";
      for (byte i = 0; i < mfrc522.uid.size; i++) {  // Erstellen des UID-
Strings
        uidString += String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : "") +
String(mfrc522.uid.uidByte[i], HEX);
      }
      publishData("esp/rfid/uid", uidString);  // UID über MQTT senden
      sendRfid = false;
      mfrc522.PICC_HaltA();  // Stop reading
    } else {
      Serial << "Error reading card." << endl;
    }
  } else {
    Serial << "No card present." << endl;
  }
}

// Funktion zum Abspielen des Morse-Codes
void playMorseCode() {
  unsigned long currentMorseCodeMillis = millis();
```

```
  static bool isSymbolSpace = false;

  if (playingMorse && currentMorseCodeMillis - previousMillisMorseCode >=
waittime) {
    previousMillisMorseCode = currentMorseCodeMillis;
    char symbol = morseBuffer[morseIndex];

    if(prev_sym == '.' || prev_sym == '-'){
      digitalWrite(LED_YELLOW, LOW);  // LED ausschalten nach einem Symbol
      publishData(TOPIC_LAMP_STATUS, "0");
      waittime = WAIT_TIME;
      prev_sym = ' ';
    } else if (symbol == '.') {
      digitalWrite(LED_YELLOW, HIGH);  // LED für Punkt-Symbol einschalten
      publishData(TOPIC_LAMP_STATUS, "1");
      prev_sym = symbol;
      waittime = DOT_TIME;
      morseIndex++;
    } else if (symbol == '-') {
      digitalWrite(LED_YELLOW, HIGH);  // LED für Strich-Symbol einschalten
      publishData(TOPIC_LAMP_STATUS, "1");
      waittime = HYPHEN_TIME;
      prev_sym = symbol;
      morseIndex++;
    } else if (symbol == ' ') {
      digitalWrite(LED_YELLOW, LOW);  // LED ausschalten für Leerzeichen
      publishData(TOPIC_LAMP_STATUS, "0");
      waittime = SPACE_TIME;
      prev_sym = ' ';
      morseIndex++;
    } else {
      digitalWrite(LED_YELLOW, LOW);  // LED ausschalten bei unbekanntem
Symbol
      playingMorse = false;
      publishData(TOPIC_LAMP_STATUS, "0");
      morseIndex = 0; // Reset Morse index
    }
    if (morseIndex >= strlen(morseBuffer)) {
      morseIndex = 0; // Wiederholen des Morse-Codes
      isSymbolSpace = false; // Startet mit Buchstabenpause beim Wiederholen
    }
  }
}

// Callback-Funktion für eingehende MQTT-Nachrichten
void callback(char* c_topic, byte* payload, unsigned int length) {
  // Konvertiere Payload in String
  String msg, topic;
  for (byte i = 0; i < length; i++) {
    msg += char(payload[i]);
```

```
}
topic = String(c_topic);
Serial << "CALLBACK: Topic = " << topic << " Payload = " << msg << endl;

// Verarbeite empfangene Topics
if (topic == TOPIC_LAMP) {
  if (msg == "1") {
    digitalWrite(LED_YELLOW, HIGH);
    publishData(TOPIC_LAMP_STATUS, "1");
  }
  if (msg == "0") {
    digitalWrite(LED_YELLOW, LOW);
    publishData(TOPIC_LAMP_STATUS, "0");
  }
}

if (topic == TOPIC_SEND_HUMIDITY) {
  Serial << "send humidity " << msg << endl;
  if (msg == "1") {
    sendHumidity = true;
    publishData(TOPIC_HUMIDITY, String(humidity));
  }
  if (msg == "0") {
    sendHumidity = false;
  }
}

if (topic == TOPIC_SEND_LDR) {
  Serial << "send brightness " << msg << endl;
  if (msg == "1") {
    sendBrightness = true;
    publishData(TOPIC_LDR, String(brightness));
  }
  if (msg == "0") {
    sendBrightness = false;
  }
}

if (topic == MORSECODE_NR_TOPIC) {
  if (msg == "1") {
    strncpy(morseBuffer, SOS, sizeof(morseBuffer) - 1);
    playingMorse = true;
  } else if(msg == "2"){
    strncpy(morseBuffer, SEK, sizeof(morseBuffer) - 1);
    playingMorse = true;
  } else if(msg == "3"){
    strncpy(morseBuffer, NSA, sizeof(morseBuffer) - 1);
    playingMorse = true;
  } else {
    playingMorse = false;
```

```
      morseIndex = 0;
      digitalWrite(LED_YELLOW, LOW);
      publishData(TOPIC_LAMP_STATUS, "0");
    }
  }

  if (topic == RFID_SEND_TOPIC) {
    if (msg == "2") {
      sendRfid = true;
    } else {
      sendRfid = false;
    }
  }
}

// Funktion zur Überprüfung der MQTT-Verfügbarkeit und Verbindung
boolean mqttAvailable() {
  while (!mqttClient.connected()) {  // Überprüfen, ob der MQTT-Client
verbunden ist
    Serial << "connecting to MQTT-Broker: ";
    Serial << MQTT_BROKER << endl;
    mqttClient.connect("ESP-Client_xyz");  // Verbindung zum MQTT-Broker
herstellen
    mqttClient.subscribe(TOPIC_LAMP);  // Abonnieren der benötigten Topics
    mqttClient.subscribe(TOPIC_SEND_HUMIDITY);
    mqttClient.subscribe(TOPIC_SEND_LDR);
    mqttClient.subscribe(MORSECODE_NR_TOPIC);
    mqttClient.subscribe(RFID_SEND_TOPIC);
  }
  return mqttClient.connected();  // Rückgabe des Verbindungsstatus
}

#endif
```

```
/* ###################################################################
 Filename      : params.h
 Author        : Bohn Matthias
 Date          : 26.05.2024
################################################################### */
#ifndef PARAMS_H
#define PARAMS_H

// MQTT-Parameter
const char* MQTT_BROKER = "192.168.43.133";
const uint16_t PORT = 1883;
```

```
WiFiClient espClient;
PubSubClient mqttClient(espClient);

// MQTT-Themen
#define TOPIC_LAMP "esp/lighting/LED_YELLOW"
#define TOPIC_LAMP_STATUS "esp/lighting/LED_YELLOW_status"
#define TOPIC_LDR "esp/brightness"
#define TOPIC_SEND_LDR "esp/brightness/send"
#define TOPIC_HUMIDITY "esp/humidity"
#define TOPIC_SEND_HUMIDITY "esp/humidity/send"
#define MORSECODE_NR_TOPIC "morsecode/nr"
#define TOPIC_TEMPERATURE "esp/temperature"
#define TOPIC_BUTTON3 "esp/btn3"
#define RFID_SEND_TOPIC "esp/rfid/send"

// Variablen für RFID
#define RST_PIN D2  // RST-PIN für RC522
#define SS_PIN D1   // SDA-PIN für RC522
volatile bool sendRfid = false;
MFRC522 mfrc522(SS_PIN, RST_PIN);  // Erstellen einer MFRC522-Instanz
/*
Orange 3,3v    -> 3,3v
Gelb RST       -> 3 RXD
Schwarz SDA    -> 5
Grün GND       -> GND
Weiß SCK       -> 14
Lila MISO      -> 12
Grau MOSI      -> 13
 */

#define TASTER_3 2
volatile bool buttonPressed = false;

//Variablen für DHT22
#define DHT_TYPE DHT22  // Typ des DHT-Sensors
#define DHT_PIN 14      // Pin, an dem der DHT-Sensor angeschlossen ist
#define DHT_POWER 4    // Pin zur Stromversorgung des DHT-Sensors
float h;

#define LED_YELLOW 13

// Variablen für LDR
const int SENSOR = 0;             // Analog-Pin, an dem der LDR (Light
Dependent Resistor) angeschlossen ist
const int ANZAHL_MESSWERTE = 25;  // Anzahl der Messwerte zur
Durchschnittsbildung
const int HYSTERESE = 10;         // Hysterese für den LDR-Wert
bool leaveHyst = true;            // Variable zur Überwachung der Hysterese
volatile bool sendBrightness = false;
```

```
// Globale Variablen
volatile bool sendHumidity = false;
int brightness = 0;
int humidity = 0;

unsigned long previousMillis = 0;
unsigned long previousMqttMillis = 0;
const long interval = 200;
unsigned long previousMillisMorseCode = 0;

// MorseCode Variablen
#define DOT_TIME 50
#define HYPHEN_TIME 1000
#define SPACE_TIME 500
#define WAIT_TIME 50
#define SOS " ...  ---  ...     "
#define SEK " ...  .  -.-     "
#define NSA " -.  ...  .-     "
int morseIndex = 0;
bool isLetterSpace = false;
volatile bool playingMorse = false;
int waittime = 0;
char prev_sym = ' ';
char morseBuffer[50];

#endif
```

```
/* ####################################################################
 Filename       : wifi.h
 Author         : Bohn Matthias
 Date           : 26.05.2024
#################################################################### */
#include <ESP8266WiFi.h>

const char* ssid = "MrRobot";       // WIFI SSID for station mode
const char* psk = "Pa55wortBohn";  // WIFI PSK

const char* ap_ssid = "EscapeRoomWifi";  // WIFI SSID for AP mode
const char* ap_psk = "EscapeRoomWifi";    // WIFI PSK
```

Programm-code HTML

```
<!DOCTYPE html>
<html lang="de">

<head>
   <meta charset="UTF-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
   <title>Escape Room</title>
   <link rel="stylesheet" href="/css/main.css" />
   <link rel="stylesheet" href="/css/room1.css" />
   <link rel="stylesheet" href="/css/room2.css" />
   <link rel="stylesheet" href="/css/room3.css" />
   <link rel="shortcut icon" href="#" type="image/x-icon" />
</head>

<body>
   <div id="escaperoom">
      <div class="jumbotron"></div>

      <div class="door door-master" data-state="close"></div>
      <div id="exit">Exit</div>

      <div id="mirror">
         <div id="player-background"></div>
         <p>PW2<br />trowssap</p>
      </div>

      <div id="room1" class="room small-room">
         <div class="door door-1" data-state="open"></div>
         <div id="mirrorLine"></div>
      </div>
      <div id="pc-table">
         <div class="pc"></div>
      </div>
      <div id="pc-chair"></div>
      <div id="mirror-puzzle-help"></div>
      <div id="morse-code"></div>
      <div id="rommNr1"><p>1</p></div>


      <div id="room2" class="room small-room">
         <div class="door door-2" data-state="close"></div>
      </div>
      <div id="morse-code-device"></div>
      <div id="lightRoom2" class="light small-room"></div>
      <div id="lightSwitchLabel"></div>
      <div id="rommNr2"><p>2</p></div>
      <div id="pc-table2">
         <div id="printer"></div>
```

```
        <div id="notes"></div>
        <div id="morseCode-puzzle-help"></div>
      </div>


      <div id="room3" class="room big-room">
        <div class="door door-3" data-state="close"></div>
      </div>
      <div id="lightRoom3" class="light"></div>
      <div id="lightSwitch"></div>
      <div id="table-dining"></div>
      <div id="schrank"></div>
      <div id="wardrobe-open">
        <div id="wardrobe-light"></div>
        <div id="rfid-chip"></div>
      </div>
      <div id="reader"></div>
      <div id="alarmLamp"></div>
      <div id="rommNr3"><p>3</p></div>
      <div id="wardrobe-puzzle-help"></div>
      <div id="candle"></div>

      <div id="hexagon1"></div>
      <div id="hexagon2"></div>
      <div id="hexagon3"></div>

      <div id="player"></div>
    </div>

    <div id="control">
      <div id="time">
        <h2>Time: 00:00</h2>
      </div>
      <div id="control-box">
        <h2>Steuerung</h2>

        <p class="control-text">
          <img src="images/control/arrow-keys-left.png" width="30" alt="" />
Links
        </p>

        <p class="control-text">
          <img src="images/control/arrow-keys-right.png" width="30" alt=""
/> Rechts
        </p>

        <p class="control-text">
          <img src="images/control/arrow-keys-up.png" width="30" alt="" />
Oben
        </p>
```

```
            <p class="control-text">
               <img src="images/control/arrow-keys-down.png" width="30" alt="" />
Unten
            </p>

            <hr />

            <p class="control-text">
               <img src="images/control/space_bar.png" height="30" alt="" />
Aktion
            </p>

            <p class="control-text">
               <img src="images/control/mouse.png" height="30" alt=""
/>Aufnehmen<br>Rucksack vergrößern
            </p>

            <p class="control-text"></p>
         </div>

         <div id="items">
            <hr />
            <h2>Rucksack</h2>
            <div id="morse-code-bag">
               <div class="morse-code-image"></div>
            </div>
            <div id="lightSwitchLabel-bag">
               <div class="lightSwitchLabel-image"></div>
            </div>
            <div id="rfid-chip-bag">
            </div>
         </div>
      </div>
      <script src="js/mqttws31.js"></script>
      <script src="js/mqtt.js" defer></script>

      <script src="js/main.js" defer></script>
      <script src="js/room1.js" defer></script>
      <script src="js/room2.js" defer></script>
      <script src="js/room3.js" defer></script>
      <script src="js/escaperoom_1.js" defer></script>
</body>

</html>
```

Programm-code CSS

```css
/* ########################################################################
 Filename      : main.css
 Author        : Bohn Matthias
 Date          : 26.05.2024
######################################################################## */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
}
body {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background-color: #f0f0f0;
}
#escaperoom {
  width: 600px;
  height: 600px;
  background-color: #cbd2d6;
  background-image: url("../images/general/wood-floor-hallway.jpg");
  background-size: 150px 150px;
  position: relative;
  border: 8px solid #000;
  display: flex;
  justify-content: center;
  align-items: center;
}

.jumbotron {
  z-index: 10;
  position: absolute;
  top: 150px;
  max-width: 450px;
  padding: 10px;
  background: steelblue;
  border-radius: 8px;
  box-shadow: snow 0px 0px 26px 5px;
  display: none;
  gap: 8px;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}

.room {
```

```
  background-color: #666;
  background-image: url("../images/general/wood-floor.jpg");
  background-size: 400px 400px;
  position: absolute;
  border: 8px solid #000;
}
.small-room {
  width: 252px;
  height: 252px;
}
.big-room {
  width: 252px;
  height: 600px;
}
#room1 {
  top: -8px;
  left: -8px;
}
#room2 {
  bottom: -8px;
  left: -8px;
}
#room3 {
  top: -8px;
  right: -8px;
}
#player {
  width: 35px; /* Gesamtbreite des Spieler-Elements einschließlich des
zusätzlichen Bildes */
  z-index: 3;
  height: 35px;
  background-image: url("../images/player/player.png"); /* Hintergrundbild des
Spielers */
  background-size: 35px 35px; /* Größe des Hintergrundbildes für den Spieler
*/
  background-position: left center; /* Position des Spieler-Bildes */
  position: absolute;
  top: 0px; /* Anpassen, um die gewünschte vertikale Positionierung zu
erreichen */
  left: 280px; /* Anpassen, um die gewünschte horizontale Positionierung zu
erreichen */
}

#player::after {
  content: ""; /* Fügt ein Pseudoelement hinzu */
  z-index: 2;
  display: none;
  width: 10px;
  height: 20px; /* Höhe des zusätzlichen Bildes */
```

```
  background-image: url("../images/player/exclamation-mark.png"); /*
Hintergrundbild des zusätzlichen Bildes */
  background-size: cover; /* Größe des Hintergrundbildes für das zusätzliche
Bild */
  position: absolute;
  top: -10px; /* Anpassen, um die gewünschte vertikale Positionierung zu
erreichen */
  left: -10px; /* Anpassen, um das zusätzliche Bild neben dem Spieler
anzuzeigen */
  animation: wobble 0.3s infinite alternate; /* Animation hinzufügen */
}
#player.show-after::after {
  display: block; /* Zusätzliches Bild anzeigen, wenn die Klasse 'show-after'
vorhanden ist */
}

@keyframes wobble {
  from {
    transform: rotate(-5deg);
  }
  to {
    transform: rotate(5deg);
  }
}

#hexagon1 {
  width: 35px; /* Gesamtbreite des Spieler-Elements einschließlich des
zusätzlichen Bildes */
  height: 35px;
  display: none;
  background-image: url("../images/general/hexagon-gray.png"); /*
Hintergrundbild des Spielers */
  background-size: 35px 35px; /* Größe des Hintergrundbildes für den Spieler
*/
  background-position: left center; /* Position des Spieler-Bildes */
  position: absolute;
  top: 540px;
  left: 290px;
}
#hexagon2 {
  width: 35px; /* Gesamtbreite des Spieler-Elements einschließlich des
zusätzlichen Bildes */
  height: 35px;
  display: none;
  background-image: url("../images/general/hexagon-gray.png"); /*
Hintergrundbild des Spielers */
  background-size: 35px 35px; /* Größe des Hintergrundbildes für den Spieler
*/
  background-position: left center; /* Position des Spieler-Bildes */
  position: absolute;
```

```
  top: 180px;
  left: 20px;
}
#hexagon3 {
  width: 35px; /* Gesamtbreite des Spieler-Elements einschließlich des
zusätzlichen Bildes */
  height: 35px;
  display: none;
  background-image: url("../images/general/hexagon-gray.png"); /*
Hintergrundbild des Spielers */
  background-size: 35px 35px; /* Größe des Hintergrundbildes für den Spieler
*/
  background-position: left center; /* Position des Spieler-Bildes */
  position: absolute;
  top: 530px;
  left: 20px;
}

.light {
  position: absolute;
  background-color: #000000f3;
  z-index: 3;
}

.door {
  position: absolute;
  background-color: #b12121;
  border-radius: 4px;
  transition: background-color 1.3s;
  z-index: 5;
}
.door-1 {
  bottom: calc(50% - 35px);
  right: -9px;
  width: 10px;
  height: 70px;
}
.door-2 {
  top: -9px;
  left: calc(50% - 35px);
  width: 70px;
  height: 10px;
}
.door-3 {
  bottom: calc(50% - 35px);
  left: -9px;
  width: 10px;
  height: 70px;
}
.door-master {
```

```
  left: calc(50% - 35px);
  top: -9px;
  width: 70px;
  height: 10px;
}
div[data-state="open"] {
  background-color: #568f44; /* Hier die gewünschte Hintergrundfarbe einfügen
*/
}

#exit {
  position: absolute;
  left: calc(50% - 17px);
  top: -32px;
  font-size: 18px;
  font-weight: 700;
  font-family: cursive;
}

#time{
  margin-bottom: 10px;
  color: #4682b4;
}

#control {
  margin-left: 30px;
}

#control-box {
  display: flex;
  flex-direction: column;
  gap: 8px;

  .control-text {
    display: flex;
    align-items: center;

    img{
      margin-right: 10px;
    }
  }
}

#items {
  margin-top: 20px;
  display: flex;
  gap: 8px;
  flex-direction: column;
  align-items: center;
}
```

```css
#morse-code-bag {
  position: relative;
  background-image: url("../images/room1/morse-codes.png");
  background-size: cover;
  width: 40px;
  height: 60px;
  display: none;
}

.morse-code-image {
  position: absolute; /* Positionierung relativ zum Elternelement */
  top: 50%;
  left: 50%;
  transform: translate(-500px, -400px);
  background-image: url("../images/room1/morse-codes.png");
  background-size: cover;
  width: 300px;
  height: 450px;
  z-index: 10;
  opacity: 0; /* Das Bild ist standardmäßig unsichtbar */
  transition: opacity 0.3s; /* Übergangseffekt für die Sichtbarkeit */
  pointer-events: none;
}

#morse-code-bag:hover .morse-code-image {
  opacity: 1; /* Bild wird sichtbar, wenn über das Elternelement geschwebt
wird */
}

#lightSwitchLabel-bag {
  position: relative;
  background-image: url("../images/room2/light-switch-horizontal.png");
  background-size: cover;
  width: 100px;
  height: 29px;
  display: none;
}

.lightSwitchLabel-image {
  position: absolute; /* Positionierung relativ zum Elternelement */
  top: 50%;
  left: 50%;
  transform: translate(-500px, -200px);
  background-image: url("../images/room2/light-switch-horizontal.png");
  background-size: cover;
  width: 200px;
  height: 58px;
  z-index: 10;
  opacity: 0; /* Das Bild ist standardmäßig unsichtbar */
```

```
  transition: opacity 0.3s; /* Übergangseffekt für die Sichtbarkeit */
  pointer-events: none;
}

#lightSwitchLabel-bag:hover .lightSwitchLabel-image {
  opacity: 1; /* Bild wird sichtbar, wenn über das Elternelement geschwebt
wird */
}

#rfid-chip-bag {
  position: relative;
  background-image: url("../images/room3/rfid-chip.png");
  background-size: cover;
  width: 58px;
  height: 60px;
  display: none;
}
```

```
/* ########################################################################
 Filename       : room1.css
 Author         : Bohn Matthias
 Date           : 26.05.2024
######################################################################## */
@keyframes backgroundFadeIn {
  0% {
    background: linear-gradient(to bottom, #b3b1b1, #837e7e);
  }
  100% {
    background: linear-gradient(to bottom, #e6e5e5, #919191);
  }
}

@keyframes textFadeIn {
  0% {
    color: rgba(240, 248, 255, 0);
  }
  100% {
    color: rgba(255, 255, 255, 0.45);
  }
}

@keyframes playerOpacity {
  0% {
    opacity: 1;
  }
  100% {
```

```
    opacity: 0.3;
  }
}

#mirror {
  z-index: 10;
  position: absolute;
  display: none;
  top: 50px;
  left: 45px;
  height: 200px;
  width: 150px;
  background: linear-gradient(to bottom, #b3b1b1, #837e7e);
  border: 6px solid #457eb6;
  border-radius: 2px;
  overflow: hidden;
}
.animateMirror {
  animation: backgroundFadeIn 4s forwards;
}

#mirror p {
  font-weight: 800;
  text-align: center;
  margin-top: 30px;
  font-family: cursive;
  font-size: 20px;
  color: rgba(240, 248, 255, 0);
}
.animate-mirror-p {
  animation: textFadeIn 4s forwards;
}

#player-background {
  background-image: url(../images/player/player.png);
  background-size: 100px 100px;
  background-position: center;
  background-repeat: no-repeat;
  width: 100%;
  height: 100%;
  filter: blur(1px);
  opacity: 1;
  position: absolute;
  top: 85px;
}
.animate-layer-background {
  animation: playerOpacity 3s forwards;
}

#mirrorLine {
```

```
  position: absolute;
  background-color: #457eb6;
  top: 0;
  left: 160px;
  width: 55px;
  height: 6px;
}

#pc-table {
  position: absolute;
  z-index: 1;
  background-image: url("../images/room1/pc-table.png");
  background-repeat: no-repeat;
  background-size: cover;
  top: 0;
  left: 5px;
  width: 88px;
  height: 35px;
}

#pc-table .pc {
  background-image: url("../images/room1/pc.png");
  background-size: cover;
  background-repeat: no-repeat;
  position: relative;
  left: 12px;
  width: 66px;
  height: 32px;
}

#pc-chair {
  z-index: 0;
  position: absolute;
  background-image: url("../images/room1/pc-chair.png");
  background-repeat: no-repeat;
  background-size: cover;
  top: 20px;
  left: 30px;
  width: 36px;
  height: 34px;
}

#mirror-puzzle-help {
  z-index: 3;
  position: absolute;
  background-image: url("../images/room1/email.png");
  background-repeat: no-repeat;
  background-size: cover;
  display: none;
  top: 3px;
```

```
  left: 60px;
  width: 18px;
  height: 18px;
  border-radius: 50%;
  animation: wobble 0.3s infinite alternate;
  box-shadow: -1px 1px 5px 1px;
}

#morse-code {
  position: absolute;
  background-image: url("../images/room1/morse-codes.png");
  background-repeat: no-repeat;
  background-size: cover;
  top: 173px;
  left: 178px;
  width: 31px;
  height: 46px;
}

#rommNr1 {
  position: absolute;
  font-weight: bold;
  top: 59px;
  left: 235px;
  color: #faebd7;
}
```

```
/* ######################################################################
 Filename       : room2.css
 Author         : Bohn Matthias
 Date           : 26.05.2024
###################################################################### */
#morse-code-device {
  position: absolute;
  background-image: url("../images/room2/morse-code-device.png");
  background-repeat: no-repeat;
  background-size: cover;
  top: 400px;
  left: 200px;
  width: 36px;
  height: 57px;
}

#lightRoom2 {
  bottom: -8px;
  left: -8px;
  /* background-color: transparent; */
```

```
}

#lightSwitchLabel {
  position: absolute;
  background-image: url("../images/room2/light-switch.png");
  background-repeat: no-repeat;
  background-size: cover;
  top: 390px;
  left: 10px;
  width: 13px;
  height: 50px;
}

#pc-table2 {
  position: absolute;
  background-image: url("../images/room1/pc-table.png");
  background-repeat: no-repeat;
  background-size: cover;
  top: 549px;
  left: 100px;
  width: 88px;
  height: 35px;

  #notes {
    background-image: url("../images/room2/notes.png");
    width: 23px;
    height: 24px;
    background-repeat: no-repeat;
    background-size: cover;
    position: relative;
    top: -18px;
    left: 8px;
  }

  #printer {
    background-image: url("../images/room2/printer.png");
    width: 40px;
    height: 25px;
    background-repeat: no-repeat;
    background-size: cover;
    position: relative;
    top: 8px;
    left: 43px;
  }

  #morseCode-puzzle-help {
    position: relative;
    background-image: url("../images/room2/prints.png");
    background-repeat: no-repeat;
    background-size: cover;
```

```
    display: block;
    top: -45px;
    left: 54px;
    width: 17px;
    height: 20px;
    animation: wobble 0.3s infinite alternate;
    box-shadow: -1px 1px 5px 1px;
  }
}

#rommNr2 {
  position: absolute;
  font-weight: bold;
  transform: rotateZ(-90deg);
  z-index: 3;
  top: 333px;
  left: 161px;
  color: #faebd7;
}
```

```
/* ########################################################################
 Filename      : room3.css
 Author        : Bohn Matthias
 Date          : 26.05.2024
######################################################################## */
#table-dining {
  position: absolute;
  background-image: url("../images/room3/table-dining.png");
  background-repeat: no-repeat;
  background-size: cover;
  top: 370px;
  left: 430px;
  width: 106px;
  height: 170px;
}

#schrank {
  position: absolute;
  background-image: url("../images/room3/Schrank.png");
  background-repeat: no-repeat;
  background-size: cover;
  top: 0px;
  left: 430px;
  width: 109px;
  height: 33px;
}
```

```css
#wardrobe-open {
  position: absolute;
  z-index: 11;
  background-image: url("../images/room3/Schrank-open.png");
  background-repeat: no-repeat;
  background-size: cover;
  display: none;
  top: -15px;
  left: 360px;
  width: 220px;
  height: 220px;
  transition: all 1.5s;

  #wardrobe-light {
    background-color: black;
    width: 126px;
    height: 152px;
    position: relative;
    top: 34px;
    left: 49px;
    transition: all 2s;
  }

  #rfid-chip {
    background-image: url("../images/room3/rfid-chip.png");
    background-repeat: no-repeat;
    background-size: cover;
    border-radius: 8px;
    width: 29px;
    height: 30px;
    position: relative;
    top: -50px;
    left: 96px;
    opacity: 0;
    transition: all 1s;
    animation: wobble 0.3s infinite alternate;
  }
  #rfid-chip:hover {
    background-color: #fafad2;
  }
}

#lightRoom3 {
  top: 0px;
  left: 348px;
  width: 236px;
  height: 584px;
}
```

```css
#lightSwitch {
  position: absolute;
  background-color: #ffffff; /*#fdf300*/
  top: 340px;
  left: 335px;
  width: 5px;
  height: 15px;
  z-index: 1;
}

#alarmLamp {
  position: absolute;
  background-image: url("../images/room3/revolving-light-red.png");
  background-size: cover;
  top: 125px;
  left: 348px;
  width: 25px;
  height: 25px;
  z-index: 1;
}

#reader {
  position: absolute;
  background-image: url("../images/room3/reader.png");
  background-size: cover;
  top: 155px;
  left: 348px;
  width: 35px;
  height: 44px;
}

#rommNr3 {
  position: absolute;
  font-weight: bold;
  top: 232px;
  left: 339px;
  color: #faebd7;
}

#wardrobe-puzzle-help {
  position: absolute;
  background-image: url("../images/room3/info.png");
  background-repeat: no-repeat;
  background-size: cover;
  display: none;
  top: 390px;
  right: 95px;
  width: 12px;
  height: 25px;
  animation: wobble 0.3s infinite alternate;
```

```
}

#candle {
  position: absolute;
  background-image: url("../images/room3/candle.png");
  background-repeat: no-repeat;
  background-size: cover;
  top: 446px;
  right: 91px;
  width: 19px;
  height: 21px;
}
```

Programm-code Javascript

```
/* ######################################################################
 Filename       : main.js
 Author         : Bohn Matthias
 Date           : 26.05.2024
###################################################################### */
// Initialisierung von Variablen
let playerSize = 35; // Größe des Spielers in Pixel
let win = false; // Zustand des Spiels: gewonnen oder nicht

let themeSoundIsPlaying = false; // Zustand der Hintergrundmusik: spielt oder
nicht

// Laden und Einstellen der Audiodateien
const stepSound = new Audio('../sounds/step2.mp3');
stepSound.volume = 0.4;
const doorSound = new Audio('../sounds/dooropened.mp3');
doorSound.volume = 0.8;
const hexagonSound = new Audio('../sounds/hexagon.mp3');
hexagonSound.volume = 0.9;
const hexagonOffSound = new Audio('../sounds/hexagon-off.mp3');
hexagonSound.volume = 0.9;
const newNotificationSound = new Audio('../sounds/new-notification.mp3');
newNotificationSound.volume = 0.9;
const lightAmpSound = new Audio('../sounds/light-amp.mp3');
lightAmpSound.volume = 0.7;

// Starten der Hintergrundmusik, falls sie nicht bereits spielt
if (!themeSoundIsPlaying) {
    // playThemeSound(); // Kommentar entfernt, da Funktion nicht definiert
    themeSoundIsPlaying = true;
}
```

```
// Funktion zum Abspielen des Schrittgeräusches
function playStepSound() {
    stepSound.play();
}

// Funktion zum Abspielen der Hintergrundmusik
function playThemeSound() {
    const themeSound_1 = new Audio('../sounds/Final Fantasy V - A
Presentiment.mp3');
    themeSound_1.volume = 0.2;
    themeSound_1.loop = true; // Dauerschleife aktivieren

    themeSound_1.play(); // Wiedergabe starten
}

// Funktion zum Überprüfen, ob eine Tür geöffnet ist
function canMoveThroughDoor(number) {
    const door = document.querySelector(`.door-${number}`);
    return door.dataset.state === "open";
}

// Funktion zur Überprüfung von Kollisionen mit Objekten im Raum
function checkCollisionWithObjects(playerPosition, playerPositionBefore, room)
{
    // Auswahl der Objekte im aktuellen Raum
    let roomObjects;
    if (room === 1) {
        roomObjects = room1Objects;
    } else if (room === 2) {
        roomObjects = room2Objects;
    } else if (room === 3) {
        roomObjects = room3Objects;
    } else {
        roomObjects = []; // Leere Liste, falls kein Raum gefunden wurde
    }

    // Überprüfung jeder Objektposition im aktuellen Raum
    for (const object of roomObjects) {
        // Berechnung der Objektgrenzen
        const objectLeft = object.left;
        const objectRight = object.left + object.width;
        const objectTop = object.top;
        const objectBottom = object.top + object.height;

        // Überprüfung, ob der Spieler mit dem Objekt kollidiert

        if (playerPositionBefore.left >= objectRight && playerPosition.left <
objectRight && playerPosition.top + playerSize > objectTop &&
playerPosition.top < objectBottom ) {
```

```
            // Spieler kollidiert mit dem Objekt von rechts
            playerPosition.left = objectRight;
            return true;
        } else if (playerPositionBefore.left + playerSize <= objectLeft &&
playerPosition.left + playerSize > objectLeft && playerPosition.top +
playerSize > objectTop && playerPosition.top < objectBottom ) {
            // Spieler kollidiert mit dem Objekt von links
            playerPosition.left = objectLeft - playerSize;
            return true;
        } else if (playerPositionBefore.top >= objectBottom &&
playerPosition.top < objectBottom && playerPosition.left + playerSize >
objectLeft && playerPosition.left < objectRight ) {
            // Spieler kollidiert mit dem Objekt von unten
            playerPosition.top = objectBottom;
            return true;
        } else if (playerPositionBefore.top + playerSize <= objectTop &&
playerPosition.top + playerSize > objectTop && playerPosition.left +
playerSize > objectLeft && playerPosition.left < objectRight ) {
            // Spieler kollidiert mit dem Objekt von oben
            playerPosition.top = objectTop - playerSize;
            return true;
        }
    }

    // Keine Kollision mit Objekten
    return false;
}

// Funktion zur Überprüfung von Kollisionen mit Gegenständen im Raum
function checkCollisionWithItems(playerPosition, playerPositionBefore) {
    // Auswahl der Gegenstände im aktuellen Raum
    let items = itemObjects;
    // Überprüfung jeder Gegenstandsposition im aktuellen Raum
    for (const item of items) {
        // Berechnung der Gegenstandsgrenzen
        const itemLeft = item.left;
        const itemRight = item.left + item.width;
        const itemTop = item.top;
        const itemBottom = item.top + item.height;

        // Überprüfung, ob der Spieler mit dem Gegenstand kollidiert
        if ( playerPositionBefore.left >= itemRight && playerPosition.left <
itemRight && playerPosition.top + playerSize > itemTop && playerPosition.top <
itemBottom ) {
            document.getElementById(item.backpackId).style.display = "block";
// Gegenstand in den Rucksack verschieben
            document.getElementById(item.id).style.display = "none"; //
Gegenstand aus dem Raum entfernen
            return true;
```

```
        } else if ( playerPositionBefore.left + playerSize <= itemLeft &&
playerPosition.left + playerSize > itemLeft && playerPosition.top + playerSize
> itemTop && playerPosition.top < itemBottom ) {
            document.getElementById(item.backpackId).style.display = "block";
// Gegenstand in den Rucksack verschieben
            document.getElementById(item.id).style.display = "none"; //
Gegenstand aus dem Raum entfernen
            return true;
        } else if ( playerPositionBefore.top >= itemBottom &&
playerPosition.top < itemBottom && playerPosition.left + playerSize > itemLeft
&& playerPosition.left < itemRight ) {
            document.getElementById(item.backpackId).style.display = "block";
// Gegenstand in den Rucksack verschieben
            document.getElementById(item.id).style.display = "none"; //
Gegenstand aus dem Raum entfernen
            return true;
        } else if ( playerPositionBefore.top + playerSize <= itemTop &&
playerPosition.top + playerSize > itemTop && playerPosition.left + playerSize
> itemLeft && playerPosition.left < itemRight ) {
            document.getElementById(item.backpackId).style.display = "block";
// Gegenstand in den Rucksack verschieben
            document.getElementById(item.id).style.display = "none"; //
Gegenstand aus dem Raum entfernen
            return true;
        }
    }

    // Keine Kollision mit Gegenständen
    return false;
}
```

```
/* ######################################################################
 Filename       : mqtt.js
 Author         : Bohn Matthias
 Date           : 26.05.2024
###################################################################### */
// Globale Variablen
var client = null; // MQTT-Client
var led_is_on = null; // Status der LED, benötigt für led_toggle()
let firstHumidityPub = true; // Status für die erste Veröffentlichung der
Luftfeuchtigkeit

// Konfigurationen
const HOSTNAME = "192.168.43.133";
const PORT = "80";
const PATH = "/ws";
```

```
const CLIENTID = "mqtt_js_" + parseInt(Math.random() * 100000, 10);

const LDR_TOPIC = "esp/brightness"; // Thema für den Helligkeitssensor
const TOPIC_SEND_LDR = "esp/brightness/send";

const HUMIDITY_TOPIC = "esp/humidity"; // Thema für die Luftfeuchtigkeit
const HUMIDITY_SEND_TOPIC = "esp/humidity/send"; // Thema zum Senden der
Luftfeuchtigkeit

const TEMPERATURE_TOPIC = "esp/temperature"; // Thema für die Temperatur

const RFID_SEND_TOPIC = "esp/rfid/send"; // Thema zum Senden des RFID
const RFID_UID_TOPIC = "esp/rfid/uid"; // Thema zum Empfangen der RFID-UID

const TOPIC_LAMP = "esp/lighting/led_yellow"; // Thema für die rote LED
const LAMP_STATUS_TOPIC = "esp/lighting/led_yellow_status"; // Thema für den
Status der roten LED

const BUTTON3_TOPIC = "esp/btn3"; // Thema für den dritten Button

const MORSECODE_NR_TOPIC = "morsecode/nr"; // Thema für den Morsecode

let humidity = 0; // Variable zur Speicherung der aktuellen Luftfeuchtigkeit
let firstHumidity = 0; // Variable zur Speicherung der ersten gemessenen
Luftfeuchtigkeit

window.onload = connect(); // Wenn die Webseite vollständig geladen ist, wird
connect() aufgerufen

// Hauptfunktion zum Herstellen der Verbindung zum MQTT-Broker
function connect() {
  // Client einrichten
  client = new Paho.MQTT.Client(HOSTNAME, Number(PORT), PATH, CLIENTID);
  console.info(
    "Verbindung zum Server wird hergestellt: Hostname: ",
    HOSTNAME,
    ". Port: ",
    PORT,
    ". Client ID: ",
    CLIENTID
  );

  // Callback-Handler setzen
  client.onConnectionLost = onConnectionLost; // Bei Verbindungsverlust
  client.onMessageArrived = onMessageArrived; // Bei eintreffenden Nachrichten

  // Optionen für die Verbindung setzen
  var options = {
    onSuccess: onConnect, // Nach erfolgreicher Verbindung wird onConnect
aufgerufen
```

```
    onFailure: onFail, // Bei Fehlschlagen der Verbindung
  };

  // Client mit den oben festgelegten Optionen verbinden
  client.connect(options);
  console.info("Verbindung wird hergestellt...");
}

// Funktion, die bei erfolgreicher Verbindung zum MQTT-Broker aufgerufen wird
function onConnect(context) {
  console.log("Client verbunden");

  // Optionen für das Abonnieren von Themen
  options = {
    qos: 0,
    onSuccess: function (context) {
      console.log("> SUB-ACK");
    },
  };
  // Relevante Themen abonnieren
  client.subscribe(LAMP_STATUS_TOPIC, options);

  // Nachricht mit Wert 0 senden
  message = new Paho.MQTT.Message("0");
  message.destinationName = RFID_SEND_TOPIC;
  message.retained = true;
  console.log("< PUB", message.destinationName, "0");
  client.send(message);
}

// Funktion, die aufgerufen wird, wenn die Verbindung zum MQTT-Broker
fehlschlägt
function onFail(context) {
  console.log("Verbindung fehlgeschlagen");
}

// Funktion, die aufgerufen wird, wenn die Verbindung zum MQTT-Broker verloren
geht
function onConnectionLost(responseObject) {
  if (responseObject.errorCode !== 0) {
    console.log("Verbindung verloren: " + responseObject.errorMessage);
    window.alert("Verbindung verloren: " + responseObject.errorMessage);
  }
}

// Funktion, die aufgerufen wird, wenn eine MQTT-Nachricht eintrifft
function onMessageArrived(message) {
  console.log("> PUB", message.destinationName, message.payloadString);
```

```
  // Aktualisieren der Elemente der Webseite basierend auf dem Thema der
Nachricht
  if (message.destinationName == LDR_TOPIC) {
    if (message.payloadString >= 450) {
      showRfidChip(true);
    } else {
      showRfidChip(false);
    }
  } else if (message.destinationName == LAMP_STATUS_TOPIC) {
    // Status der LED basierend auf der Nachricht aktualisieren
    if (message.payloadString == "1") {
      morseCodeSound.play();
      led_is_on = true;
    } else {
      morseCodeSound.pause();
      morseCodeSound.currentTime = 0;
      led_is_on = false;
    }
  } else if (message.destinationName == HUMIDITY_TOPIC) {
    // Erste Veröffentlichung der Luftfeuchtigkeit speichern
    if (firstHumidityPub) {
      firstHumidity = message.payloadString;
      firstHumidityPub = false;
    }
    humidity = message.payloadString;
  } else if (message.destinationName == BUTTON3_TOPIC) {
    if (message.payloadString == "1") {
      toggleLightRoom3(); // Licht im Raum 3 umschalten
    }
  } else if (message.destinationName == RFID_UID_TOPIC) {
    // Aktion ausführen, wenn die richtige RFID-UID empfangen wird
    if (message.payloadString == "30f0987e") {
      win = true;
      lightAmpSound.play();
      document.getElementById("alarmLamp").style.backgroundImage =
"url('../images/room3/revolving-light-green.png')";
    }
  }
}

// Funktion zum Abonnieren eines Themas
function subscribe_topic(topic) {
  options = {
    qos: 0,
    onSuccess: function (context) {
      console.log("> SUB-Ack " + topic);
    },
  };
  console.log("> SUB " + topic);
  client.subscribe(topic, options);
```

```
}

// Funktion zum Umschalten der LED
function led_toggle() {
  var payload;
  if (led_is_on) {
    payload = "0";
    led_is_on = false;
  } else {
    payload = "1";
    led_is_on = true;
  }

  // Nachricht mit dem neuen LED-Status senden
  message = new Paho.MQTT.Message(payload);
  message.destinationName = TOPIC_LAMP;
  message.retained = true;
  console.log("< PUB", message.destinationName, payload);
  client.send(message);
}
```

```
/* ########################################################################
 Filename       : escaperoom_1.js
 Author         : Bohn Matthias
 Date           : 26.05.2024
######################################################################## */
// Initialisiere die Spielerposition
let playerPosition = {
    top: 0,
    left: 280,
};

// Variablen zur Verfolgung des aktuellen Zustands
let jumbotronVisible = false;
let actualRoom = 0;
let hexagonVisible = false;
let hexagon1Active = false;
let hexagon2Active = false;
let hexagon3Active = false;
let seconds = 0;

// Variablen für die Rätsel
let puzzleSeconds = 0;
let mirrorPuzzle = false;
let mirrorPuzzleFirstHelp = false;
let morseCodePuzzle = false;
```

```
let morseCodePuzzleFirstHelp = false;
let lightSwitch3Puzzle = false;
let lightSwitch3PuzzleFirstHelp = false;
let hexagonPuzzle = false;
let hexagonPuzzleFirstHelp = false;
let wardrobePuzzle = false;
let wardrobePuzzleFirstHelp = false;

let updateTimeInterval;

// Liste der Gegenstandsobjekte
const itemObjects = [
    { id: 'morse-code', backpackId: 'morse-code-bag', top: 178, left: 183,
width: 21, height: 36 },
    { id: 'lightSwitchLabel', backpackId: 'lightSwitchLabel-bag', top: 390,
left: 10, width: 13, height: 50 },
];

// Jumbotron-Element
let jumbotronElem = document.querySelector(".jumbotron");

// Willkommensnachricht im Jumbotron anzeigen
jumbotronElem.innerHTML = `
   <h2>Willkommen</h2>
   <p>Du bist in einem alten, verlassenen Herrenhaus gefangen. Um zu
entkommen, musst du eine Reihe kniffliger Rätsel lösen. Nutze die versteckten
Hinweise und zeige,
   dass du scharfsinnig genug bist, um den Weg nach draußen zu finden. Deine
Zeit läuft - kannst du das Geheimnis des Hauses lüften und
entkommen?</p><p>Tipp: Nutze den ESP und den Computersound um die Rätsel zu
lösen.</p>
   <p style="display: flex; gap: 10px;"><img
src="images/control/space_bar.png" height="20" alt=""><span>Drücke die Space
Taste zum Starten</span></p>
`;
jumbotronElem.style.display = "flex"; // Jumbotron sichtbar machen
jumbotronVisible = true; // Zustand aktualisieren

// Funktion zur Zeitformatierung
function pad(value) {
    return value.toString().padStart(2, '0');
}

// Funktion zur Zeitaktualisierung
function updateTime() {
    // Berechne Minuten und Sekunden
    const minutes = Math.floor(seconds / 60);
    const secs = seconds % 60;

    // Formatiere die Zeit
```

```
    const formattedTime = `Time: ${pad(minutes)}:${pad(secs)}`;

    // Aktualisiere die Zeitanzeige
    document.querySelector("#time h2").textContent = formattedTime;

    // Überprüfe die Zeit für die jeweiligen Rätsel und zeige Hilfetexte an
    const mirrorPuzzleHelpElem = document.getElementById("mirror-puzzle-
help");
    if (!mirrorPuzzle) {
        if (puzzleSeconds == 90 && !mirrorPuzzleFirstHelp) {
            mirrorPuzzleFirstHelp = true;
            newNotificationSound.play();
            mirrorPuzzleHelpElem.style.display = "block";
        }
    } else if (!lightSwitch3PuzzleFirstHelp && !mirrorPuzzleFirstHelp) {
        mirrorPuzzleHelpElem.style.display = "none";
    }

    const morseCodePuzzleHelpElem = document.getElementById("morseCode-puzzle-
help");
    if (!morseCodePuzzle && mirrorPuzzle) {
        if (puzzleSeconds == 90 && !morseCodePuzzleFirstHelp) {
            morseCodePuzzleFirstHelp = true;
            newNotificationSound.play();
            morseCodePuzzleHelpElem.style.display = "block";
        }
    } else {
        morseCodePuzzleHelpElem.style.display = "none";
    }

    const lightSwitch3PuzzleHelpElem = document.getElementById("mirror-puzzle-
help");
    if (!lightSwitch3Puzzle && mirrorPuzzle && morseCodePuzzle &&
hexagonPuzzle) {
        if (puzzleSeconds == 60 && !lightSwitch3PuzzleFirstHelp) {
            lightSwitch3PuzzleFirstHelp = true;
            newNotificationSound.play();
            lightSwitch3PuzzleHelpElem.style.display = "block";
        }
    } else if (!lightSwitch3PuzzleFirstHelp && !mirrorPuzzleFirstHelp) {
        lightSwitch3PuzzleHelpElem.style.display = "none";
    }

    const wardrobePuzzleHelpElem = document.getElementById("wardrobe-puzzle-
help");
    if (!wardrobePuzzle && mirrorPuzzle && morseCodePuzzle && hexagonPuzzle &&
lightSwitch3Puzzle) {
        if (puzzleSeconds == 60 && !wardrobePuzzleFirstHelp) {
            wardrobePuzzleFirstHelp = true;
            newNotificationSound.play();
```

```
            wardrobePuzzleHelpElem.style.display = "block";
        }
    } else if (!wardrobePuzzleFirstHelp && !mirrorPuzzleFirstHelp) {
        wardrobePuzzleHelpElem.style.display = "none";
    }

    // Erhöhe die Sekunden
    seconds++;
    puzzleSeconds++;
}

// Funktion zum Verstecken des Jumbotrons
function hideJumbotron() {
    const jumbotron = document.querySelector(".jumbotron");
    if (jumbotron) {
        jumbotron.style.display = "none";
        jumbotronVisible = false;

        // Entferne den Event-Listener, um Mehrfachausführungen zu verhindern
        document.removeEventListener("keydown", hideJumbotron);
    }

    // Starte den Timer und aktualisiere die Zeit jede Sekunde
    updateTimeInterval = setInterval(updateTime, 1000);

    // Rufe die Funktion sofort auf, um den initialen Wert zu setzen
    updateTime();
}

// Event-Listener hinzufügen, um das Jumbotron zu verstecken
document.addEventListener("keydown", hideJumbotron);

// Event-Listener für Tastatureingaben hinzufügen
document.addEventListener("keydown", function (event) {
    const player = document.getElementById("player");

    // Vorherige Spielerposition speichern
    let playerPositionBefore = {
        top: playerPosition.top,
        left: playerPosition.left,
    };

    // Spielerbewegung basierend auf der gedrückten Pfeiltaste
    switch (event.key) {
        case "ArrowUp":
            playerPosition.top -= 15;
            playStepSound();
            break;
        case "ArrowDown":
            playerPosition.top += 15;
```

```
            playStepSound();
            break;
        case "ArrowLeft":
            playerPosition.left -= 15;
            playStepSound();
            break;
        case "ArrowRight":
            playerPosition.left += 15;
            playStepSound();
            break;
    }

    // Spielerposition begrenzen
    if (playerPosition.top < 0) playerPosition.top = 0;
    if (playerPosition.top > 550) playerPosition.top = 550;
    if (playerPosition.left < 0) playerPosition.left = 0;
    if (playerPosition.left > 550) playerPosition.left = 550;

    // Kollision mit den Wänden und Objekten in den Räumen überprüfen
    checkMoveRoom1RightWall(playerPositionBefore, playerPosition);
    checkMoveRoom1BottomWall(playerPositionBefore, playerPosition);
    checkMoveRoom2RightWall(playerPositionBefore, playerPosition);
    checkMoveRoom2TopWall(playerPositionBefore, playerPosition);
    checkMoveRoom3LeftWall(playerPositionBefore, playerPosition);
    const isColliding = checkCollisionWithObjects(playerPosition,
playerPositionBefore, actualRoom);

    // Spielerposition aktualisieren, wenn keine Kollision vorliegt
    player.style.top = playerPosition.top + "px";
    player.style.left = playerPosition.left + "px";

    // Kollisionen in den Räumen überprüfen
    checkRoom1MirrorPos(playerPosition, playerPositionBefore);
    if ((!mirrorPuzzle && mirrorPuzzleFirstHelp) || (!lightSwitch3Puzzle &&
lightSwitch3PuzzleFirstHelp)) {
        checkRoom1PcPos(playerPosition, playerPositionBefore);
    }

    checkRoom2DoorPos(playerPosition, playerPositionBefore);
    checkRoom2MorseCodePos(playerPosition, playerPositionBefore);
    if (!morseCodePuzzle && morseCodePuzzleFirstHelp) {
        checkRoom2TablePos(playerPosition, playerPositionBefore);
    }

    checkRoom3LightSwitchPos(playerPosition, playerPositionBefore);
    checkRoom3WardrobePos(playerPosition, playerPositionBefore);
    checkRoom3ReaderPos(playerPosition, playerPositionBefore);
    if (!wardrobePuzzle && wardrobePuzzleFirstHelp) {
        checkRoom3TablePos(playerPosition, playerPositionBefore);
    }
```

```
    // Kollision mit dem Hexagon überprüfen
    checkHexagonPos(playerPosition);

    // Kollision mit Gegenständen überprüfen
    checkCollisionWithItems(playerPosition, playerPositionBefore);
});

// Funktion zum Deaktivieren des Hexagons nach Ablauf der Zeit
function clearHexagon(elem, hexagonActive) {
    if (!hexagon1Active || !hexagon2Active || !hexagon3Active) {
        hexagonOffSound.play();
        elem.style.backgroundImage = "url('/images/general/hexagon-
gray.png')";
        if (hexagonActive == 1) {
            hexagon1Active = false;
        } else if (hexagonActive == 2) {
            hexagon2Active = false;
        } else {
            hexagon3Active = false;
        }
    }
}

// Funktion zum Überprüfen der Position des Hexagons
function checkHexagonPos(playerPosition) {
    if (hexagonVisible) {
        if (actualRoom == 0 && !hexagon1Active && playerPosition.left >= 280
&& playerPosition.left <= 295 && playerPosition.top >= 515 &&
playerPosition.top <= 535) {
            const hexagon1Elem = document.getElementById("hexagon1");
            hexagon1Elem.style.backgroundImage =
"url('/images/general/hexagon-blue.png')";
            hexagonSound.play();
            hexagon1Active = true;
            setTimeout(() => clearHexagon(hexagon1Elem, 1), 15000);
        }
        if (actualRoom == 1 && !hexagon2Active && playerPosition.left >= 10 &&
playerPosition.left <= 25 && playerPosition.top >= 155 && playerPosition.top
<= 180) {
            const hexagon2Elem = document.getElementById("hexagon2");
            hexagon2Elem.style.backgroundImage =
"url('/images/general/hexagon-green.png')";
            hexagonSound.play();
            hexagon2Active = true;
            setTimeout(() => clearHexagon(hexagon2Elem, 2), 3000);
            if (hexagon1Active && hexagon2Active && hexagon3Active) {
                hexagonPuzzle = true;
                hexagonPuzzleFirstHelp = false;
                puzzleSeconds = 0;
```

```
                let door = document.querySelector(".door-3");
                doorSound.play();
                door.setAttribute("data-state", "open");

                if (lightRoom3State) {
                    const lightRoom3 = document.getElementById("lightRoom3");
                    lightRoom3.style.backgroundColor = "transparent";
                }
            }
        }
        if (actualRoom == 2 && !hexagon3Active && playerPosition.left >= 10 &&
playerPosition.left <= 25 && playerPosition.top >= 505 && playerPosition.top
<= 530) {
            const hexagon3Elem = document.getElementById("hexagon3");
            hexagon3Elem.style.backgroundImage =
"url('/images/general/hexagon-red.png')";
            hexagonSound.play();
            hexagon3Active = true;
            setTimeout(() => clearHexagon(hexagon3Elem, 3), 9000);
        }
    }
}
```

```
/* ######################################################################
 Filename       : room1.js
 Author         : Bohn Matthias
 Date           : 26.05.2024
###################################################################### */

// Variable zur Überprüfung, ob der Spiegel im Raum 1 sichtbar ist
let mirror1Visible = false;

// ID des Intervalls für die Feuchtigkeitsüberprüfung
let intervalIdHumidity;

// Objekte im Raum 1 mit ihren Positionen und Abmessungen
const room1Objects = [
  { id: "pc-table", top: 5, left: 10, width: 78, height: 25 },
  { id: "mirror", top: 0, left: 160, width: 55, height: 6 },
  { id: "pc-chair", top: 20, left: 35, width: 26, height: 29 },
];

// Funktion zur Überprüfung der Position des Spiegels im Raum 1
function checkRoom1MirrorPos(playerPosition, playerPositionBefore) {
  const playerElement = document.querySelector("#player");
```

```
  if (actualRoom == 1 && playerPosition.left >= 156 && playerPosition.left <=
186 && playerPosition.top >= 6 && playerPosition.top <= 21) {
    playerElement.classList.add("show-after"); // Füge eine Klasse hinzu, um
das zusätzliche Bild anzuzeigen

    // Abonniere das Thema, um die Feuchtigkeitsdaten zu erhalten
    subscribe_topic(HUMIDITY_TOPIC);

    // Füge den Event-Listener für das Tastaturereignis "keydown" hinzu
    document.addEventListener("keydown", showMirror1);
  } else if (actualRoom == 1 && playerPositionBefore.left >= 156 &&
playerPositionBefore.left <= 186 && playerPositionBefore.top >= 6 &&
playerPositionBefore.top <= 21 && (playerPosition.left < 156 ||
playerPosition.left > 186 || playerPosition.top < 6 || playerPosition.top >
21)) {
    playerElement.classList.remove("show-after"); // Entferne die Klasse, um
das zusätzliche Bild auszublenden

    // Entferne den Event-Listener
    document.removeEventListener("keydown", showMirror1);
    document.querySelector("#mirror").style.display = "none";
    document.querySelector("#mirror").classList.remove("animateMirror"); //
Entferne die Animationsklasse
    document.querySelector("#mirror p").classList.remove("animate-mirror-p");
// Entferne die Animationsklasse für den Text
    document.querySelector("#player-background").classList.remove("animate-
layer-background"); // Entferne die Animationsklasse für den Hintergrund

    // Abonnement vom Thema HUMIDITY_TOPIC kündigen
    client.unsubscribe(HUMIDITY_TOPIC, {
      onSuccess: function () {
        console.log("Abonnement von " + HUMIDITY_TOPIC + " gekündigt");
      }
    });

    // Sende Nachricht mit Wert 0, um die Feuchtigkeitsübertragung zu stoppen
    message = new Paho.MQTT.Message("0");
    message.destinationName = HUMIDITY_SEND_TOPIC;
    message.retained = true;
    console.log("< PUB", message.destinationName, "0");
    client.send(message);

    clearInterval(intervalIdHumidity);

    mirror1Visible = false; // Aktualisiere den Zustand auf unsichtbar
  }
}

// Funktion zur Anzeige des Spiegels im Raum 1
function showMirror1(event) {
```

```
  // Überprüfe, ob die gedrückte Taste die "Space"-Taste ist
  if (event.code === "Space") {
    // Wenn der Spiegel sichtbar ist, setze das Display auf "none"
    if (mirror1Visible) {
      document.querySelector("#mirror").style.display = "none";
      document.querySelector("#mirror").classList.remove("animateMirror"); //
Entferne die Animationsklasse
      document.querySelector("#mirror p").classList.remove("animate-mirror-
p"); // Entferne die Animationsklasse für den Text
      document.querySelector("#player-background").classList.remove("animate-
layer-background"); // Entferne die Animationsklasse für den Hintergrund

      // Abonnement vom Thema HUMIDITY_TOPIC kündigen
      client.unsubscribe(HUMIDITY_TOPIC, {
        onSuccess: function () {
          console.log("Abonnement von " + HUMIDITY_TOPIC + " gekündigt");
        }
      });

      // Sende Nachricht mit Wert 0, um die Feuchtigkeitsübertragung zu
stoppen
      message = new Paho.MQTT.Message("0");
      message.destinationName = HUMIDITY_SEND_TOPIC;
      message.retained = true;
      console.log("< PUB", message.destinationName, "0");
      client.send(message);

      clearInterval(intervalIdHumidity);

      mirror1Visible = false; // Aktualisiere den Zustand auf unsichtbar
    } else {
      // Andernfalls setze das Display auf "block", füge die Animationsklassen
hinzu und aktualisiere den Zustand auf sichtbar
      document.querySelector("#mirror").style.display = "block";

      firstHumidityPub = true;

      // Sende Nachricht mit Wert 1, um die Feuchtigkeitsübertragung zu
starten
      message = new Paho.MQTT.Message("1");
      message.destinationName = HUMIDITY_SEND_TOPIC;
      message.retained = true;
      console.log("< PUB", message.destinationName, "1");
      client.send(message);

      // Überprüfe die Feuchtigkeit und führe die Animation aus
      checkHumidityAndAnimate();

      mirror1Visible = true;
    }
```

```
  }
}

// Funktion zur Überprüfung der Feuchtigkeitsdaten und Ausführung der
Animation
function checkHumidityAndAnimate() {
  // Setze ein Intervall, das alle 200ms überprüft
  intervalIdHumidity = setInterval(() => {
    if (humidity > firstHumidity + 15) {
      // Stoppe das Intervall, wenn die Bedingung erfüllt ist
      clearInterval(intervalIdHumidity);

      // Führe die Animationsbefehle aus
      document.querySelector("#mirror").classList.add("animateMirror");
      document.querySelector("#mirror p").classList.add("animate-mirror-p");
      document.querySelector("#player-background").classList.add("animate-
layer-background");
    }
  }, 200);
}

// Funktion zur Überprüfung der Position des PCs im Raum 1
function checkRoom1PcPos(playerPosition, playerPositionBefore) {
  if (playerPosition.left >= 0 && playerPosition.left <= 88 &&
playerPosition.top >= 30 && playerPosition.top <= 60) {

    // Füge den Event-Listener für das Tastaturereignis "keydown" hinzu
    if (!mirrorPuzzle) {
      document.addEventListener("keydown", showMirrorPuzzleInfo);
    } else if (mirrorPuzzle && morseCodePuzzle && !lightSwitch3Puzzle) {
      document.addEventListener("keydown", showLightSwitch3PuzzleInfo);
    }
  } else if (actualRoom == 1 && playerPositionBefore.left >= 0 &&
playerPositionBefore.left <= 88 && playerPositionBefore.top >= 30 &&
playerPositionBefore.top <= 60 && (playerPosition.left < 0 ||
playerPosition.left > 88 || playerPosition.top < 30 || playerPosition.top >
60)) {
    // Ursprüngliche Stile wiederherstellen, wenn das Jumbotron ausgeblendet
wird
    jumbotronElem.style.display = "none";
    jumbotronElem.style.background = "steelblue";
    jumbotronElem.style.borderRadius = "8px";
    jumbotronElem.style.boxShadow = "snow 0px 0px 26px 5px";
    jumbotronElem.style.alignItems = "center";
    jumbotronElem.style.border = "none";

    jumbotronVisible = false;

    // Entferne den Event-Listener für das Tastaturereignis "keydown"
    if (!mirrorPuzzle) {
```

```
      document.removeEventListener("keydown", showMirrorPuzzleInfo);
    } else if (mirrorPuzzle && morseCodePuzzle && !lightSwitch3Puzzle) {
      document.removeEventListener("keydown", showLightSwitch3PuzzleInfo);
    }
  }
}

// Funktion zur Anzeige der Spiegelrätsel-Informationen
function showMirrorPuzzleInfo(event) {
  if (event.code === "Space") {
    let jumbotronElem = document.querySelector(".jumbotron");

    if (jumbotronVisible) {
      // Ursprüngliche Stile wiederherstellen, wenn das Jumbotron ausgeblendet
wird
      jumbotronElem.style.display = "none";
      jumbotronElem.style.background = "steelblue";
      jumbotronElem.style.borderRadius = "8px";
      jumbotronElem.style.boxShadow = "snow 0px 0px 26px 5px";
      jumbotronElem.style.alignItems = "center";
      jumbotronElem.style.border = "none";

      jumbotronVisible = false;
    } else {
      // Zeige die Informationen zum Spiegelrätsel an
      const today = new Date().toLocaleDateString();
      const mail = `
        <p><b>NEUE E-MAIL</b></p>
        <p><b>Von:</b> ESCAPE ROOM SYSTEM</p>
        <p><b>An:</b> SPIELER</p>
        <p><b>Datum:</b> ${today}</p>
        <br>
        <p><b>Nachricht:</b></p>
        <p>Was unsichtbar ist, wird sichtbar, wenn der Atem der Natur es
berührt.</p>
      `;
      jumbotronElem.innerHTML = mail;

      jumbotronElem.style.display = "flex";
      jumbotronElem.style.background = "#e6e5e5";
      jumbotronElem.style.borderRadius = "0";
      jumbotronElem.style.border = "2px solid";
      jumbotronElem.style.boxShadow = "snow 0px 0px 8px 0px";
      jumbotronElem.style.alignItems = "flex-start";

      jumbotronVisible = true;
    }
  }
}
```

```
// Funktion zur Anzeige der Lichtschalter 3 Rätsel-Informationen
function showLightSwitch3PuzzleInfo(event) {
  if (event.code === "Space") {
    let jumbotronElem = document.querySelector(".jumbotron");

    if (jumbotronVisible) {
      // Ursprüngliche Stile wiederherstellen, wenn das Jumbotron ausgeblendet
wird
      jumbotronElem.style.display = "none";
      jumbotronElem.style.background = "steelblue";
      jumbotronElem.style.borderRadius = "8px";
      jumbotronElem.style.boxShadow = "snow 0px 0px 26px 5px";
      jumbotronElem.style.alignItems = "center";
      jumbotronElem.style.border = "none";

      jumbotronVisible = false;
    } else {
      // Zeige die Informationen zum Lichtschalter 3 Rätsel an
      const today = new Date().toLocaleDateString();
      const mail = `
        <p><b>NEUE E-MAIL</b></p>
        <p><b>Von:</b> ESCAPE ROOM SYSTEM</p>
        <p><b>An:</b> SPIELER</p>
        <p><b>Datum:</b> ${today}</p>
        <br>
        <p><b>Nachricht:</b></p>
        <p>Finde den dritten von drei, um den Pfad zu erleuchten.</p>
      `;
      jumbotronElem.innerHTML = mail;

      jumbotronElem.style.display = "flex";
      jumbotronElem.style.background = "#e6e5e5";
      jumbotronElem.style.borderRadius = "0";
      jumbotronElem.style.border = "2px solid";
      jumbotronElem.style.boxShadow = "snow 0px 0px 8px 0px";
      jumbotronElem.style.alignItems = "flex-start";

      jumbotronVisible = true;
    }
  }
}

// Funktion zur Überprüfung der Bewegung an der rechten Wand im Raum 1
function checkMoveRoom1RightWall(playerPositionBefore, playerPosition) {
  if (playerPositionBefore.left >= 244 && playerPosition.left < 244 &&
playerPosition.top < 244) {
    if (!(playerPosition.top >= 85 && playerPosition.top <= 115 &&
canMoveThroughDoor(1))) {
      playerPosition.left = 244;
      actualRoom = 0; // Setze den Spieler zurück in Raum 0
```

```
    } else {
      playerPosition.left = 201;
      actualRoom = 1; // Setze den Spieler zurück in Raum 1
    }
  } else {
    if (playerPositionBefore.left <= 201 && playerPosition.left > 201 &&
playerPosition.top < 244) {
      if (!(playerPosition.top >= 85 && playerPosition.top <= 115 &&
canMoveThroughDoor(1))) {
        playerPosition.left = 201;
        actualRoom = 1; // Setze den Spieler zurück in Raum 1
      } else {
        playerPosition.left = 244;
        actualRoom = 0; // Setze den Spieler zurück in Raum 0
      }
    }
  }
}

// Funktion zur Überprüfung der Bewegung an der unteren Wand im Raum 1
function checkMoveRoom1BottomWall(playerPositionBefore, playerPosition) {
  if (playerPosition.left < 244 && playerPositionBefore.top <= 202 &&
playerPosition.top > 202) {
    playerPosition.top = 202; // Setze den Spieler zurück an die obere Grenze
der unteren Wand
  }
  if (playerPosition.left < 244 && playerPositionBefore.top >= 244 &&
playerPosition.top < 244) {
    playerPosition.top = 244; // Setze den Spieler zurück an die untere Grenze
der unteren Wand
  }
}
```

```
/* ####################################################################
 Filename       : room2.js
 Author         : Bohn Matthias
 Date           : 26.05.2024
#################################################################### */
// Liste der Objekte im Raum 2
const room2Objects = [
  { id: "morse-code-device", top: 405, left: 205, width: 26, height: 47 },
  { id: "pc-table2", top: 554, left: 105, width: 78, height: 30 },
];
// Passwort für Tür 2
const passwordDoor2 = "trowssap";
```

```
// Mögliche Morsecode-Nachrichten
const morseCodeMessage = ["sos", "sek", "nsa"];

// Zufällige Auswahl einer Morsecode-Nachricht
const randomMorseCodeNumber = Math.floor(Math.random() * 3) + 1;

// Morsecode-Sound
const morseCodeSound = new Audio('../sounds/morsesound.mp3');
morseCodeSound.volume = 0.2;

// Funktion zur Überprüfung der Bewegung an der oberen Wand des Raums 2
function checkMoveRoom2TopWall(playerPositionBefore, playerPosition) {
  // Raum 2 Obere Wand
  if (
    playerPosition.left < 244 &&
    playerPositionBefore.top <= 306 &&
    playerPosition.top > 306
  ) {
    if (
      !(
        playerPosition.left >= 85 &&
        playerPosition.left <= 115 &&
        canMoveThroughDoor(2)
      )
    ) {
      playerPosition.top = 306;
      actualRoom = 0;
    } else {
      playerPosition.top = 348;
      actualRoom = 2;
    }
  }

  if (
    playerPosition.left < 244 &&
    playerPositionBefore.top >= 348 &&
    playerPosition.top < 348
  ) {
    if (
      !(
        playerPosition.left >= 85 &&
        playerPosition.left <= 115 &&
        canMoveThroughDoor(2)
      )
    ) {
      playerPosition.top = 348;
      actualRoom = 2;
    } else {
      playerPosition.top = 306;
      actualRoom = 0;
```

```
    }
  }
}

// Funktion zur Überprüfung der Bewegung an der rechten Wand des Raums 2
function checkMoveRoom2RightWall(playerPositionBefore, playerPosition) {
  // Raum 2 Rechte Wand
  if (
    playerPositionBefore.left >= 244 &&
    playerPosition.left < 244 &&
    playerPosition.top > 306
  ) {
    playerPosition.left = 244;
  }
  if (
    playerPositionBefore.left <= 201 &&
    playerPosition.left > 201 &&
    playerPosition.top > 306
  ) {
    playerPosition.left = 201;
  }
}

// Funktion zur Überprüfung der Position der Tür im Raum 2
function checkRoom2DoorPos(playerPosition, playerPositionBefore) {
  const playerElement = document.getElementById("player");
  if (playerPosition.left >= 70 && playerPosition.left <= 130 &&
playerPosition.top >= 290 && playerPosition.top <= 306 &&
!canMoveThroughDoor(2)) {
    playerElement.classList.add("show-after"); // Füge eine Klasse hinzu, um
das zusätzliche Bild anzuzeigen

    // Füge den Event-Listener für das Tastaturereignis "keydown" hinzu
    document.addEventListener("keydown", showDoor2PwDialog);
  } else if (playerPositionBefore.left >= 70 && playerPositionBefore.left <=
130 && playerPositionBefore.top >= 290 && playerPositionBefore.top <= 306 &&
(playerPosition.left < 70 || playerPosition.left > 130 || playerPosition.top <
290 || playerPosition.top > 306)) {
    playerElement.classList.remove("show-after"); // Entferne die Klasse, um
das zusätzliche Bild auszublenden

    // Entferne den Event-Listener
    document.removeEventListener("keydown", showDoor2PwDialog);
    document.querySelector(".jumbotron").style.display = "none";
    jumbotronVisible = false; // Aktualisiere den Zustand auf unsichtbar
  }
}

// Funktion zur Anzeige des Passwortdialogs für Tür 2
function showDoor2PwDialog(event) {
```

```
  if (event.code === "Space") {
    let jumbotronElem = document.querySelector(".jumbotron");

    if (jumbotronVisible) {
      jumbotronElem.style.display = "none";
      jumbotronVisible = false;
    } else {
      jumbotronElem.innerHTML = `
        <h2>Passwort</h2>
        <input type="password" name="door2pw" id="door2pw" />
      `;

      // Ensuring the focus is set after the element is rendered and visible
      setTimeout(() => {
        const inputElem = document.getElementById("door2pw");
        // Set focus to the input field
        inputElem.focus();

        inputElem.addEventListener("change", (e) => {
          let inputValue = e.target.value.toLowerCase();

          if (inputValue == passwordDoor2) {

            mirrorPuzzle = true;
            mirrorPuzzleFirstHelp = false;
            puzzleSeconds = 0;

            // Zugreifen auf das Tür-Element
            const door = document.querySelector(".door-2");
            // Zugrifff auf die Raumhelligkeit
            const lightRoom2 = document.getElementById("lightRoom2");
            // Ändern des data-state-Attributs auf "open"
            doorSound.play();
            door.setAttribute("data-state", "open");

            lightRoom2.style.backgroundColor = "#00000000";
          }
        });
      }, 0);

      jumbotronElem.style.display = "flex";
      jumbotronVisible = true;
    }
  }
}

// Variable zur Überprüfung, ob der Morsecode bereits gesendet wurde
let messageSent = false;

// Funktion zur Überprüfung der Position des Morsecode-Geräts im Raum 2
```

```
function checkRoom2MorseCodePos(playerPosition, playerPositionBefore) {
  const playerElement = document.querySelector("#player");
  if (actualRoom == 2 && playerPosition.left > 160 && playerPosition.left <=
170 && playerPosition.top > 395 && playerPosition.top < 428) {
    playerElement.classList.add("show-after"); // Füge eine Klasse hinzu, um
das zusätzliche Bild anzuzeigen

    // Überprüfe, ob die Nachricht noch nicht gesendet wurde
    if (!messageSent) {
      // Senden der Nachricht
      message = new Paho.MQTT.Message("" + randomMorseCodeNumber);
      message.destinationName = MORSECODE_NR_TOPIC;
      message.retained = true;
      console.log("< PUB", message.destinationName, "" +
randomMorseCodeNumber);
      client.send(message);

      // Markiere, dass die Nachricht gesendet wurde
      messageSent = true;
    }

    // Füge den Event-Listener für das Tastaturereignis "keydown" hinzu
    document.addEventListener("keydown", showRoom2MorseCodeDialog);
  } else if (actualRoom == 2 && playerPositionBefore.left >= 160 &&
playerPositionBefore.left <= 170 && playerPositionBefore.top >= 395 &&
playerPositionBefore.top <= 428 && (playerPosition.left < 160 ||
playerPosition.left > 170 || playerPosition.top < 395 || playerPosition.top >
428)) {
    playerElement.classList.remove("show-after"); // Entferne die Klasse, um
das zusätzliche Bild auszublenden

    // Sende Nachricht mit Wert 0
    message = new Paho.MQTT.Message("0");
    message.destinationName = MORSECODE_NR_TOPIC;
    message.retained = true;
    console.log("< PUB", message.destinationName, "0");
    client.send(message);

    // Setze messageSent zurück auf false
    messageSent = false;

    // Entferne den Event-Listener
    document.querySelector(".jumbotron").style.display = "none";
    document.removeEventListener("keydown", showRoom2MorseCodeDialog);
  }
}

// Funktion zur Anzeige des Dialogs für den Morsecode im Raum 2
function showRoom2MorseCodeDialog(event) {
  if (event.code === "Space") {
```

```
    let jumbotronElem = document.querySelector(".jumbotron");

    if (jumbotronVisible) {
      jumbotronElem.style.display = "none";
      jumbotronVisible = false;
    } else {
      jumbotronElem.innerHTML = `
        <h2>Nachricht?</h2>
        <input type="text" name="morseCodeMessage" id="morseCodeMessage" />
      `;

      jumbotronElem.style.display = "flex";
      jumbotronVisible = true;

      // Ensuring the focus is set after the element is rendered and visible
      setTimeout(() => {
        const inputElem = document.getElementById("morseCodeMessage");
        inputElem.focus();

        inputElem.addEventListener("change", (e) => {
          let inputValue = e.target.value.toLowerCase();
          if (inputValue == morseCodeMessage[randomMorseCodeNumber - 1]) {

            morseCodePuzzle = true;
            morseCodePuzzleFirstHelp = false;
            puzzleSeconds = 0;

            hexagonOffSound.play();
            document.getElementById("hexagon1").style.display = "block";
            document.getElementById("hexagon2").style.display = "block";
            document.getElementById("hexagon3").style.display = "block";
            hexagonVisible = true;
          }
        });
      }, 0);
    }
  }
}

function checkRoom2TablePos(playerPosition, playerPositionBefore) {
  if (actualRoom == 2 && playerPosition.left >= 130 && playerPosition.left <=
160 && playerPosition.top >= 504 && playerPosition.top <= 519) {

    // Füge den Event-Listener für das Tastaturereignis "keydown" hinzu
    document.addEventListener("keydown", showMorseCodePuzzleInfo);

  } else if (actualRoom == 2 && playerPositionBefore.left >= 130 &&
playerPositionBefore.left <= 160 && playerPositionBefore.top >= 504 &&
playerPositionBefore.top <= 519 && (playerPosition.left < 130 ||
```

```
playerPosition.left > 160 || playerPosition.top < 504 || playerPosition.top >
519)) {
    // Ursprüngliche Stile wiederherstellen, wenn das Jumbotron ausgeblendet
wird
    jumbotronElem.style.display = "none";
    jumbotronElem.style.background = "steelblue";
    jumbotronElem.style.borderRadius = "8px";
    jumbotronElem.style.boxShadow = "snow 0px 0px 26px 5px";
    jumbotronElem.style.alignItems = "center";
    jumbotronElem.style.border = "none";

    jumbotronVisible = false;

    // Entferne den Event-Listener für das Tastaturereignis "keydown" hinzu
    document.removeEventListener("keydown", showMorseCodePuzzleInfo);

  }
}
function showMorseCodePuzzleInfo(event) {
  if (event.code === "Space") {
    let jumbotronElem = document.querySelector(".jumbotron");

    if (jumbotronVisible) {
      // Ursprüngliche Stile wiederherstellen, wenn das Jumbotron ausgeblendet
wird
      jumbotronElem.style.display = "none";
      jumbotronElem.style.background = "steelblue";
      jumbotronElem.style.borderRadius = "8px";
      jumbotronElem.style.boxShadow = "snow 0px 0px 26px 5px";
      jumbotronElem.style.alignItems = "center";
      jumbotronElem.style.border = "none";

      jumbotronVisible = false;
    } else {
      const today = new Date().toLocaleDateString();
      const faxMessage = `
        <p><b>FAX-NACHRICHT</b></p>
        <p><b>Von:</b> ESCAPE ROOM SYSTEM</p>
        <p><b>An:</b> SPIELER</p>
        <p><b>Datum:</b> ${today}</p>
        <br>
        <p><b>Nachricht:</b></p>
        <p>Blinkende Lichter tanzen im Rhythmus der Punkte und Striche.</p>
        <p>Die Antwort liegt in den alten Mustern.</p>
        <p>Erkenne 3 Zeichen, um den Weg zu finden.</p>
      `;
      jumbotronElem.innerHTML = faxMessage;

      jumbotronElem.style.display = "flex";
      jumbotronElem.style.background = "#e6e5e5";
```

```
      jumbotronElem.style.borderRadius = "0";
      jumbotronElem.style.border = "2px solid";
      jumbotronElem.style.boxShadow = "snow 0px 0px 8px 0px";
      jumbotronElem.style.alignItems = "flex-start";

      jumbotronVisible = true;
    }
  }
}
```

```
/* ######################################################################
 Filename      : room3.js
 Author        : Bohn Matthias
 Date          : 26.05.2024
###################################################################### */
// Liste der Objekte im Raum 3
const room3Objects = [
   { id: "table-dining", top: 380, left: 440, width: 86, height: 150 },
   { id: "schrank", top: 0, left: 435, width: 99, height: 28 },
   { id: "reader", top: 160, left: 348, width: 30, height: 34 },
];

let rfidCount = 0;
let lightRoom3State = false;

// Funktion zur Überprüfung der Bewegung an der linken Wand des Raums 3
function checkMoveRoom3LeftWall(playerPositionBefore, playerPosition) {
   // Überprüfung der linken Wand im Raum 3
   if (playerPositionBefore.left <= 305 && playerPosition.left > 305) {
      if (!(playerPosition.top >= 260 && playerPosition.top <= 290 &&
canMoveThroughDoor(3))) {
         playerPosition.left = 305;
         actualRoom = 0;
      } else {
         playerPosition.left = 348;
         actualRoom = 3;
      }
   }

   // Überprüfung der linken Wand im Raum 3 (umgekehrte Richtung)
   if (playerPositionBefore.left >= 348 && playerPosition.left < 348) {
      if (!(playerPosition.top >= 260 && playerPosition.top <= 290 &&
canMoveThroughDoor(3))) {
         playerPosition.left = 348;
         actualRoom = 3;
      } else {
```

```
            playerPosition.left = 305;
            actualRoom = 0;
        }
    }
}

// Funktion zur Überprüfung der Position des Lichtschalters im Raum 3
function checkRoom3LightSwitchPos(playerPosition, playerPositionBefore) {
    const playerElement = document.querySelector("#player");
    // Überprüfung, ob der Spieler sich in der Nähe des Lichtschalters befindet
und ob er sich im Raum 3 befindet
    if (playerPosition.left > 300 && playerPosition.left <= 305 &&
playerPosition.top >= 320 && playerPosition.top <= 340 && actualRoom == 0) {
        playerElement.classList.add("show-after"); // Klasse hinzufügen, um
zusätzliches Bild anzuzeigen

        // Event-Listener für das Tastaturereignis "keydown" hinzufügen, um die
Taste zu lesen
        readButton(true);
    } else if (actualRoom == 0 && playerPositionBefore.left > 300 &&
playerPositionBefore.left <= 305 && playerPositionBefore.top >= 320 &&
playerPositionBefore.top <= 340 && (playerPosition.left < 300 ||
playerPosition.left > 305 || playerPosition.top < 320 || playerPosition.top >
340)) {
        playerElement.classList.remove("show-after"); // Klasse entfernen, um
zusätzliches Bild auszublenden

        readButton(false);
    }
}

// Funktion zur Überprüfung der Position des Schranks im Raum 3
function checkRoom3WardrobePos(playerPosition, playerPositionBefore) {
    const playerElement = document.querySelector("#player");
    // Überprüfung, ob der Spieler sich in der Nähe des Schranks befindet und
ob er sich im Raum 3 befindet
    if (playerPosition.left > 440 && playerPosition.left < 495 &&
playerPosition.top >= 20 && playerPosition.top < 40 && actualRoom == 3) {
        playerElement.classList.add("show-after"); // Klasse hinzufügen, um
zusätzliches Bild anzuzeigen

        // Sende Nachricht mit Wert 0
        message = new Paho.MQTT.Message("1");
        message.destinationName = TOPIC_SEND_LDR;
        message.retained = true;
        console.log("< PUB", message.destinationName, "1");
        client.send(message);

        // Füge den Event-Listener für das Tastaturereignis "keydown" hinzu
        document.addEventListener("keydown", showWardrobe);
```

```
    } else if (actualRoom == 3 && playerPositionBefore.left > 440 &&
playerPositionBefore.left < 495 && playerPositionBefore.top >= 20 &&
playerPositionBefore.top < 40 && (playerPosition.left < 440 ||
playerPosition.left > 495 || playerPosition.top < 20 || playerPosition.top >
40)) {
        playerElement.classList.remove("show-after"); // Klasse entfernen, um
zusätzliches Bild auszublenden

        // Entferne den Event-Listener
        document.removeEventListener("keydown", showWardrobe);

        // Sende Nachricht mit Wert 1
        message = new Paho.MQTT.Message("0");
        message.destinationName = TOPIC_SEND_LDR;
        message.retained = true;
        console.log("< PUB", message.destinationName, "0");
        client.send(message);

        const wardrobeElem = document.getElementById("wardrobe-open");
        wardrobeElem.style.display = "none";

        readLdr(false);
    }
}

// Funktion zum Lesen der Taste
function readButton(event) {
    if (event) {
        subscribe_topic(BUTTON3_TOPIC); // Funktion zum Abonnieren des Themas
aufrufen
    } else {
        // Abonnement vom Thema BUTTON3_TOPIC kündigen
        client.unsubscribe(BUTTON3_TOPIC, {
            onSuccess: function () {
                console.log("Abonnement von " + BUTTON3_TOPIC + " gekündigt");
            }
        });
    }
}

// Funktion zum Umschalten des Lichts im Raum 3
function toggleLightRoom3() {
    const lightRoom3 = document.getElementById("lightRoom3");
    const lightSwitchRoom3 = document.getElementById("lightSwitch");

    // Berechnet den aktuellen Hintergrundstil des Lichtschalters
    const computedStyle = window.getComputedStyle(lightSwitchRoom3);
    const backgroundColor = computedStyle.backgroundColor;

    if (!lightRoom3State) {
```

```
      // Wenn der Lichtschalter aus ist, ändere ihn zu gelb und schalte das
Licht ein
      lightRoom3State = true;
      lightSwitchRoom3.style.backgroundColor = "#fdf300";
      if (canMoveThroughDoor(3)) {
         lightRoom3.style.backgroundColor = "transparent";
         lightSwitch3Puzzle = true;
         lightSwitch3PuzzleFirstHelp = false;
      }
   } else {
      // Wenn der Lichtschalter an ist, ändere ihn zu weiß und schalte das
Licht aus
      lightRoom3State = false;
      lightSwitchRoom3.style.backgroundColor = "#ffffff";
      lightRoom3.style.backgroundColor = "#000000f3";
   }
}

// Funktion zum Anzeigen des Schranks
function showWardrobe(event) {
   // Überprüft, ob die gedrückte Taste die Leertaste ist
   if (event.code === "Space") {
      const wardrobeElem = document.getElementById("wardrobe-open");
      wardrobeElem.style.display = "block"; // Schrank anzeigen
      readLdr(true); // Beginnt, den LDR-Sensor zu lesen
   }
}


// Funktion zum Lesen der Helligkeit
function readLdr(event) {
   if (event) {
      subscribe_topic(LDR_TOPIC); // Funktion zum Abonnieren des Themas
aufrufen
   } else {
      // Abonnement vom Thema LDR_TOPIC kündigen
      client.unsubscribe(LDR_TOPIC, {
         onSuccess: function () {
            console.log("Abonnement von " + LDR_TOPIC + " gekündigt");
         }
      });
   }
}

// Funktion zur Anzeige des RFID-Chips
function showRfidChip(state) {
   const rfidElem = document.getElementById("rfid-chip");
   const wardrobeLightElem = document.getElementById("wardrobe-light");

   if (state) {
```

```
      // Licht des Schranks ausblenden
      wardrobeLightElem.style.opacity = "0";

      // Puzzle-Status aktualisieren
      wardrobePuzzle = true;
      wardrobePuzzleFirstHelp = false;
      puzzleSeconds = 0;

      if (rfidElem) {
         // RFID-Chip sichtbar machen
         rfidElem.style.opacity = "1";

         // Event-Listener hinzufügen, um den RFID-Chip beim Klicken zu
entfernen
         rfidElem.addEventListener("click", (e) => {
            rfidElem.remove();
            document.getElementById("rfid-chip-bag").style.display = "block";
         });
      }

   } else {
      // Licht des Schranks wieder einblenden
      wardrobeLightElem.style.opacity = "1";

      if (rfidElem) {
         // RFID-Chip unsichtbar machen
         rfidElem.style.opacity = "0";
      }
   }
}


// Funktion zur Überprüfung der Position des rfid Lesegeräts im Raum 3
function checkRoom3ReaderPos(playerPosition, playerPositionBefore) {
   const playerElement = document.querySelector("#player");
   // Überprüfung, ob der Spieler sich in der Nähe des rfid
Lesegeräts  befindet und ob er sich im Raum 3 befindet
   if (playerPosition.left >= 377 && playerPosition.left < 385 &&
playerPosition.top >= 140 && playerPosition.top <= 170 && actualRoom == 3) {
      playerElement.classList.add("show-after"); // Klasse hinzufügen, um
zusätzliches Bild anzuzeigen

      if (document.getElementById("rfid-chip-bag").style.display != "") {
         // Sende Nachricht mit Wert 0
         message = new Paho.MQTT.Message("2");
         message.destinationName = RFID_SEND_TOPIC;
         message.retained = true;
         console.log("< PUB", message.destinationName, "2");
         client.send(message);
```

```
            subscribe_topic(RFID_UID_TOPIC);
        }

        // Füge den Event-Listener für das Tastaturereignis "keydown" hinzu
        document.addEventListener("keydown", checkRfid);
    } else if (actualRoom == 3 && playerPositionBefore.left >= 377 &&
playerPositionBefore.left < 385 && playerPositionBefore.top >= 140 &&
playerPositionBefore.top <= 170 && (playerPosition.left < 377 ||
playerPosition.left > 385 || playerPosition.top < 140 || playerPosition.top >
170)) {
        playerElement.classList.remove("show-after"); // Klasse entfernen, um
zusätzliches Bild auszublenden

        // Sende Nachricht mit Wert 0
        message = new Paho.MQTT.Message("0");
        message.destinationName = RFID_SEND_TOPIC;
        message.retained = true;
        console.log("< PUB", message.destinationName, "0");
        client.send(message);

        // Entferne den Event-Listener
        document.removeEventListener("keydown", checkRfid);
        document.querySelector(".jumbotron").style.display = "none";
        jumbotronVisible = false; // Aktualisiere den Zustand auf unsichtbar
    }
}

function checkRfid(event) {
    if (event.code === "Space") {
        let jumbotronElem = document.querySelector(".jumbotron");

        if (jumbotronVisible) {
            jumbotronElem.style.display = "none";
            jumbotronVisible = false;
        } else {

            if (win && mirrorPuzzle && morseCodePuzzle && hexagonPuzzle &&
lightSwitch3Puzzle && wardrobePuzzle) {

                doorSound.play();
                clearInterval(updateTimeInterval);

                // Zugreifen auf das Tür-Element
                const door = document.querySelector(".door-master");
                door.setAttribute("data-state", "open");

                const minutes = Math.floor(seconds / 60);
                const secs = seconds % 60;

                // Formatiere die Zeit
```

```
            const formattedTime = `${pad(minutes)}:${pad(secs)}`;

            jumbotronElem.innerHTML = `
                  <h2>👌 Mission Erfolgreich</h2>
                  <h3>Du hast alle Rätsel erfolgreich gelöst</h3>
                  <h3>Deine Zeit: ${formattedTime}</h3>
               `;

            jumbotronElem.style.display = "flex";
            jumbotronVisible = true;

         } else if (rfidCount >= 2) {
            jumbotronElem.innerHTML = `
                  <h2>!Mission gescheitert!</h2>
                  <h3>Das Haus bleibt nun für immer verschlossen!</h3>
               `;

            jumbotronElem.style.display = "flex";
            jumbotronVisible = true;
         } else {

            jumbotronElem.innerHTML = `
               <h2>!Warnung!</h2>
               <h3>Du hast noch ${2 - rfidCount} Versuche</h3>
            `;

            rfidCount++;

            jumbotronElem.style.display = "flex";
            jumbotronVisible = true;
         }
      }
   }
}

function checkRoom3TablePos(playerPosition, playerPositionBefore) {
   if (actualRoom == 3 && playerPosition.left >= 435 && playerPosition.left <=
500 && playerPosition.top >= 330 && playerPosition.top <= 345) {

      // Füge den Event-Listener für das Tastaturereignis "keydown" hinzu
      document.addEventListener("keydown", showWardrobePuzzleInfo);

   } else if (actualRoom == 3 && playerPositionBefore.left >= 435 &&
playerPositionBefore.left <= 500 && playerPositionBefore.top >= 330 &&
playerPositionBefore.top <= 345 && (playerPosition.left < 435 ||
playerPosition.left > 500 || playerPosition.top < 330 || playerPosition.top >
345)) {
      // Ursprüngliche Stile wiederherstellen, wenn das Jumbotron ausgeblendet
wird
      jumbotronElem.style.display = "none";
```

```
      jumbotronVisible = false;

      // Entferne den Event-Listener für das Tastaturereignis "keydown" hinzu
      document.removeEventListener("keydown", showWardrobePuzzleInfo);

   }
}
function showWardrobePuzzleInfo(event) {
   if (event.code === "Space") {
      let jumbotronElem = document.querySelector(".jumbotron");

      if (jumbotronVisible) {
         // Ursprüngliche Stile wiederherstellen, wenn das Jumbotron
ausgeblendet wird
         jumbotronElem.style.display = "none";

         jumbotronVisible = false;
      } else {
         const today = new Date().toLocaleDateString();
         const Message = `
         <p><b>Hinweis:</b></p>
         <p>In der Dunkelheit liegt ein Geheimnis, erleuchte es, um es zu
sehen.</p>
       `;
         jumbotronElem.innerHTML = Message;

         jumbotronElem.style.display = "flex";

         jumbotronVisible = true;
      }
   }
}
```