**Dokumentation ESP32 Mikrocontroller**

Unser Projektziel ist die smarte Überwachung und Zugangskontrolle zu einem Rechenzentrum.  
Für dieses Vorhaben verwenden wir den **ESP32 Mikrocontroller**, da dieser nativ eine Verbindung über WiFi unterstützt. Darüber hinaus taktet der ESP32 schneller als beispielsweise ein Arduino Uno und bietet zudem mehr Ein- und Ausgänge.

Für unser cyber-physisches System setzen wir mehrere Sensoren und Aktoren ein:

**DHT11**

Der DHT11 ist ein Temperatur- und Luftfeuchtigkeitssensor. Er misst die Temperatur in Grad Celsius sowie die Luftfeuchtigkeit in Prozent. Die Daten werden digital übertragen, eine manuelle Normalisierung oder Kalibrierung war laut unseren Tests nicht notwendig. Die Einbindung des DHT11 in den Mikrocontroller-Code erfolgt über die Verwendung der Bibliothek <DHT.h>.

**LCD1602 (I²C)**

Das **LCD1602** ist ein Liquid Crystal Display, das in diesem Projekt zur Darstellung von Temperatur, Luftfeuchtigkeit sowie Systemzuständen dient. Es wurde über den **SDA- und SCL-Bus** angeschlossen, da es das I²C-Protokoll unterstützt.

**Buzzer V1.2**

Der Buzzer erzeugt hörbare Töne, deren Tonhöhe in Hertz (Hz) im Code definiert wird. Er wird primär für Zutrittssignale sowie für Alarmtöne verwendet. Der Aktor kommuniziert digital; eine zusätzliche Konfiguration ist nicht notwendig.

**LEDs**

Eine grüne und eine rote LED zeigen den aktuellen Systemstatus an:

* **Rot**: System gesichert (Alarm scharf geschaltet)
* **Grün**: System ungesichert (Alarm deaktiviert)  
  Die LEDs sind über 220-Ohm-Widerstände angeschlossen und werden über digitale Pins mit HIGH bzw. LOW geschaltet.

**RFID-RC522**

Der RFID-RC522 Sensor erkennt RFID-fähige Tags oder Geräte. Die Anbindung erfolgt über den **SPI-Bus**. Unsere Zugangskontrolle basiert auf dem Vergleich der übermittelten Tag-IDs mit freigegebenen IDs.

**Präsenzsensor**

Dieser Sensor ist vorgesehen, wurde jedoch noch nicht eingerichtet.

**Verkabelung**

Die genaue Verkabelung der Komponenten ist im folgenden Schaubild dargestellt:

**[Schaubild der ESP32-Verkabelung]**

**Funktionalitäten des Mikrocontrollers**

**Grundfunktion**

Die Kernaufgaben des Mikrocontrollers sind:

* Messung von Temperatur und Luftfeuchtigkeit
* Einlesen von RFID-Karten
* Auslösung eines Alarms bei bestimmten Bedingungen

Ein Alarm wird ausgelöst, wenn:

* ein konfigurierbarer Grenzwert (z. B. Temperatur) überschritten wird
* eine Präsenz erkannt wird, während das System gesichert ist

**Konsistente Konfiguration**

Ein Teil des Speichers des ESP32 wird zur Dateiablage genutzt. Dort befinden sich u. a. die Dateien config.json und access\_IDs.json. Diese persistenten Daten ermöglichen es, Konfigurationen und Zugangs-IDs auch nach einem Neustart des ESP32 zuverlässig zu laden. Änderungen an der Konfiguration oder Zugangsberechtigungen können bequem über ein Webinterface vorgenommen werden.

**WiFi- & MQTT-Verbindung**

Der ESP32 stellt beim Start automatisch eine Verbindung zum konfigurierten WiFi-Netzwerk sowie zum MQTT-Broker her. Die Verbindung wird kontinuierlich überwacht; im Falle eines Abbruchs wird sie automatisch wiederhergestellt.

**Kommunikation über MQTT**

Die gesamte Netzwerkkommunikation des ESP32 läuft über **MQTT**. Es wurde ein passwortgeschützter MQTT-Broker eingerichtet.  
Sowohl die Kommunikation mit der Datenbank als auch die Konfiguration des Systems erfolgen ausschließlich über MQTT.

Da das System eine bidirektionale Kommunikation benötigt, kann der ESP32 sowohl MQTT-Nachrichten senden als auch empfangen und auswerten. Der Datenaustausch zwischen Backend und Mikrocontroller basiert auf MQTT-Kommandos und einer eigens entwickelten Nachrichtenstruktur.