Hochschule
München
University of
Applied Sciences

DEPARTMENT OF APPLIED SCIENCES AND MECHATRONICS

BERICHT DATENKOMMUNIKATION

RFID-Zutrittskontrolle mittels ESP32 & RC522

Name	Matrikelnummer
Ettl, Alexander	29349325
Welzel, Tim	26502625

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Konzept

Abbildung 1: elektronisches Türschloss [1]

1.1 Projektübersicht

Dieses Projekt implementiert ein elektronisches Zutrittskontrollsystem basierend auf RFID-Technologie (Radio-Frequency Identification). Das System ermöglicht die kontaktlose Identifikation von Personen mittels RFID-Karten und gewährt oder verweigert den Zutritt basierend auf einer zentral verwalteten Berechtigungsliste.

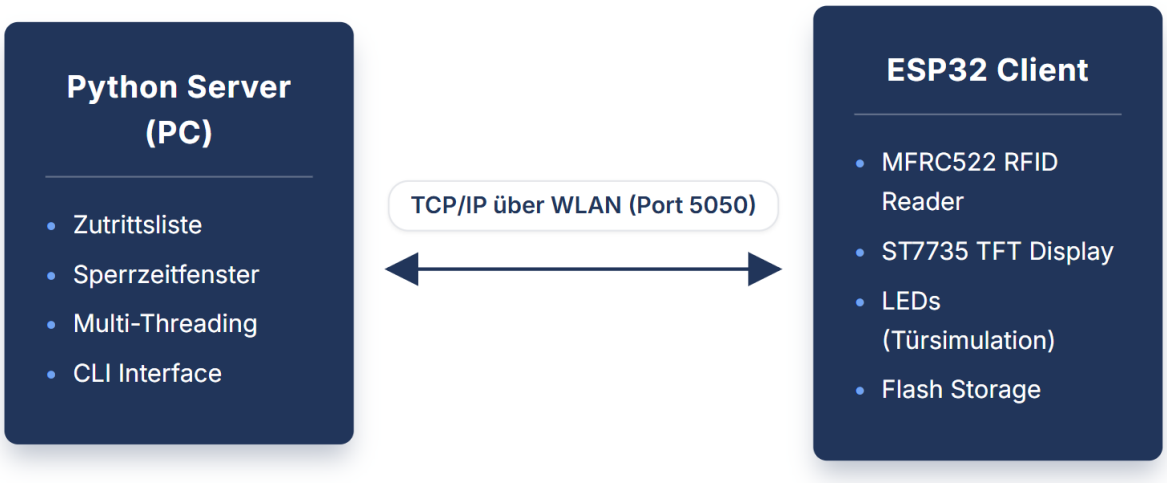

Abbildung 2: Client-Server-Modell [eigene Darstellung]

Die Architektur folgt dem klassischen Client-Server-Modell:

- Der Server läuft auf einem PC und verwaltet die Zutrittsliste sowie Sperrzeitfenster
- Der Client ist ein ESP32-Mikrocontroller, der RFID-Karten liest und mit dem Server kommuniziert

1.2 Anforderungen gemäß Lastenheft

Tabelle 1: Anforderungen aus Lastenheft [2]

Anforderung	Beschreibung
RFID-Lesung	Kontaktloses Lesen der eindeutigen Kartenidentifikation (UID)
Zutrittsprüfung	Abgleich der UID mit der serverseitigen Berechtigungsliste
Sperrzeitfenster	Zeitbasierte Zutrittsbeschränkungen (z.B. nachts)
Offline-Modus	Lokale Fallback-Liste bei Serverausfall
Visuelle Rückmeldung	LED- und Display-Anzeige des Zutrittsstatus
Synchronisation	Automatische Aktualisierung der lokalen Liste

1.3 Systemarchitektur

Das System besteht aus zwei Hauptkomponenten, die über WLAN (TCP/IP) miteinander kommunizieren:

- Python-Server (PC): Zutrittsliste, Sperrzeitfenster, Multi-Client-Unterstützung, CLI-Interface
- ESP32-Client: RFID-Reader (MFRC522), TFT-Display (ST7735), LEDs (Türsimulation), Lokale Liste (Flash)

1.4 Betriebsmodi

Das System unterstützt zwei Betriebsmodi:

Online-Modus: ESP32 ist mit dem WLAN verbunden und erreicht den Server. Jede RFID-Abfrage wird in Echtzeit an den Server gesendet. Der Server prüft UID, Sperrzeitfenster und sendet ALLOW oder DENY.

Offline-Modus: Bei Netzwerkausfall oder Server-Unerreichbarkeit. ESP32 nutzt die lokal im Flash gespeicherte Berechtigungsliste. Periodische Reconnect-Versuche alle 60 Sekunden.

2 Hardware

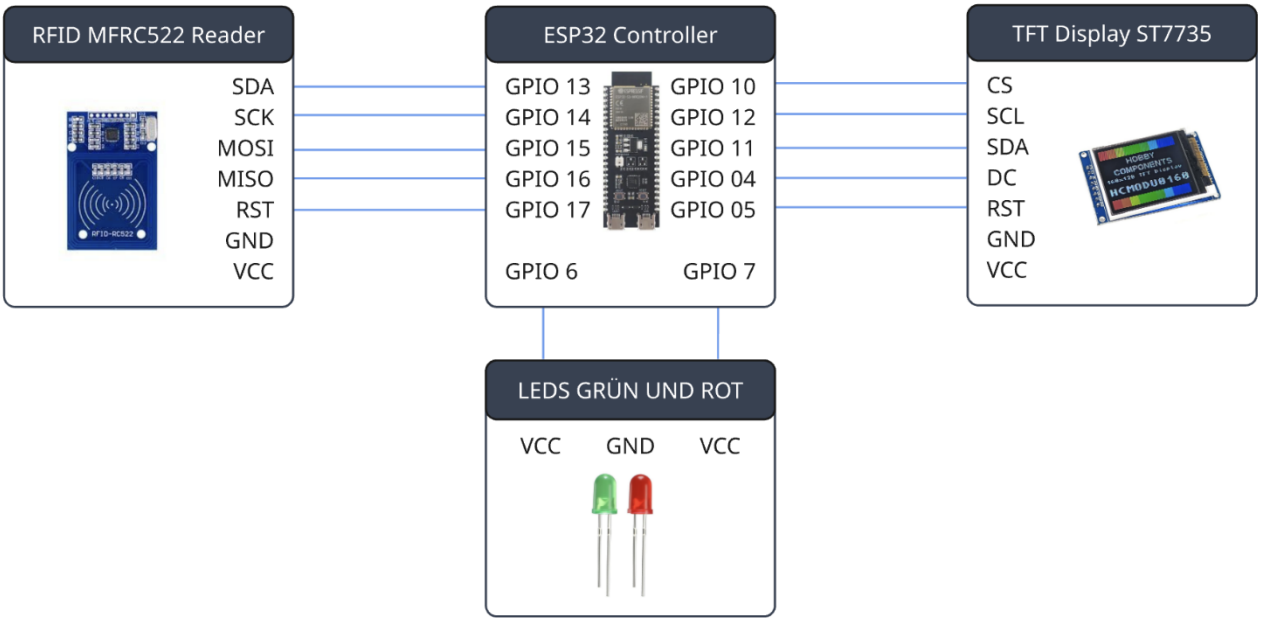

Abbildung 3: vereinfachter Schaltplan [eigene Darstellung]

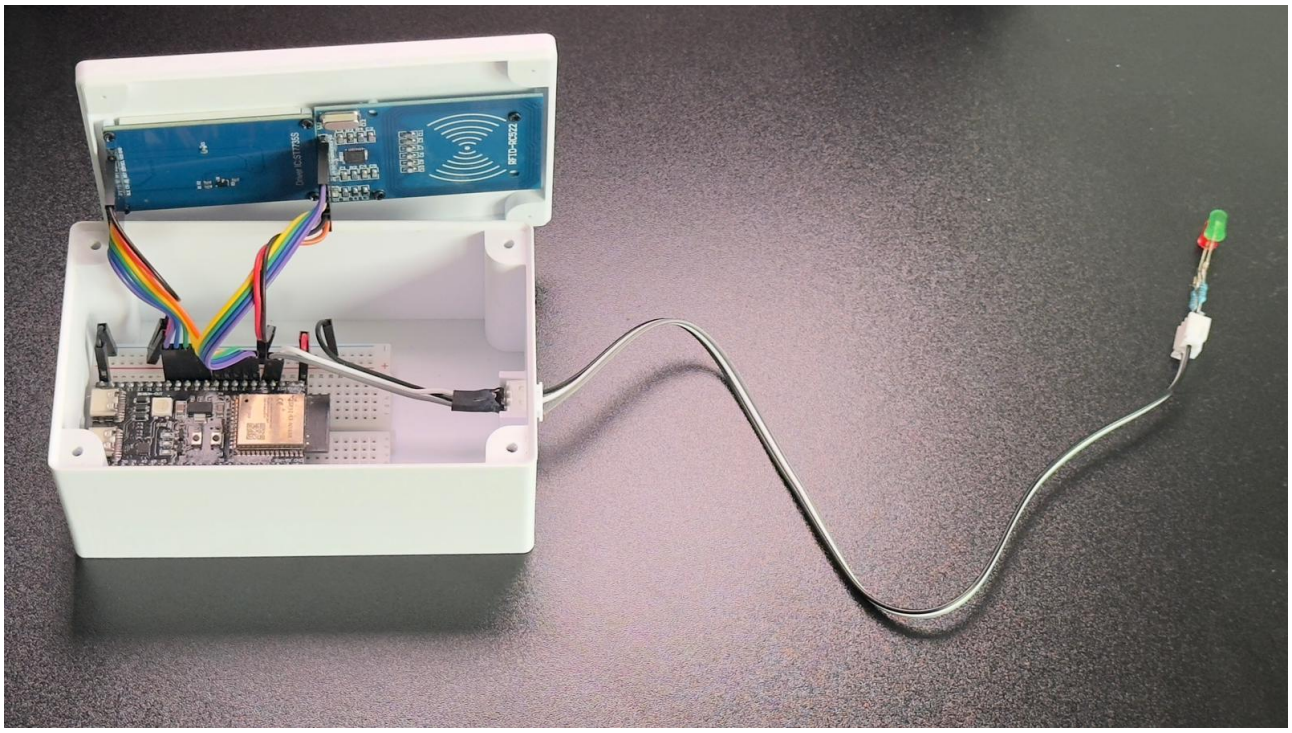

Abbildung 4: Komponenten in Gehäuse [eigene Darstellung]

2.1 ESP32-Mikrocontroller

Der ESP32 ist ein leistungsstarker 32-Bit-Mikrocontroller der Firma Espressif mit integriertem WLAN und Bluetooth.

Tabelle 2: ESP32 Übersicht

Eigenschaft	Wert
Version	ESP32-S3-N16R8
Prozessor	Xtensa LX7 Dual-Core
RAM	512 KB SRAM + 8 MB PSRAM
Flash	16 MB
WLAN	802.11 b/g/n, 2.4 GHz
GPIO	34 programmierbare Pins
SPI	4 Hardware-SPI-Schnittstellen
Betriebsspannung	3.3V

2.2 RFID-Reader MFRC522

Der MFRC522 ist ein RFID-Lese-/Schreibmodul der Firma NXP für die Frequenz 13.56 MHz nach ISO/IEC 14443 A/MIFARE.

Tabelle 3: MFRC522 RFID Übersicht

Eigenschaft	Wert
Frequenz	13.56 MHz
Protokoll	ISO 14443 A/MIFARE
Lesereichweite	ca. 2-5 cm
Schnittstelle	SPI (bis 10 Mbit/s)
Betriebsspannung	3.3V

Die Kommunikation erfolgt über SPI mit folgender Pinbelegung:

Tabelle 4: MFRC522 RFID Pinouts

MFRC522-Pin	ESP32-GPIO	Funktion
SDA (CS)	GPIO 13	Chip Select
SCK	GPIO 14	Serial Clock
MOSI	GPIO 15	Master Out, Slave In
MISO	GPIO 16	Master In, Slave Out
RST	GPIO 17	Reset

2.3 TFT-Display ST7735

Das ST7735 ist ein 1.8" TFT-Farbdisplay mit einer Auflösung von 128×160 Pixeln.

Tabelle 5: TFT-Display Übersicht

Eigenschaft	Wert
Auflösung	128 × 160 Pixel
Farbtiefe	18-bit (262.144 Farben)
Schnittstelle	SPI
Größe	1.8 Zoll Diagonale

Die Pinbelegung für das Display:

Tabelle 6: TFT-Display Pinouts

ST7735-Pin	ESP32-GPIO	Funktion
SCL (SCK)	GPIO 12	Serial Clock
SDA (MOSI)	GPIO 11	Serial Data
RST	GPIO 05	Reset
DC	GPIO 04	Data/Command
CS	GPIO 10	Chip Select

2.4 LEDs zur Türsimulation

Zwei LEDs simulieren den Zustand einer elektrischen Türverriegelung:

Tabelle 7: LED Pinouts

LED	ESP32-GPIO	Bedeutung
Grün	GPIO 06	Zutritt gewährt
Rot	GPIO 07	Zutritt verweigert

2.5 Verdrahtungshinweis

Wichtiger Hinweis: MFRC522 und ST7735 nutzen separate SPI-Busse, da sie unterschiedliche Taktgeschwindigkeiten und Konfigurationen erfordern. Der ESP32 unterstützt mehrere Hardware-SPI-Schnittstellen, wodurch beide Geräte parallel betrieben werden können.

3 Softwarearchitektur

3.1 Server

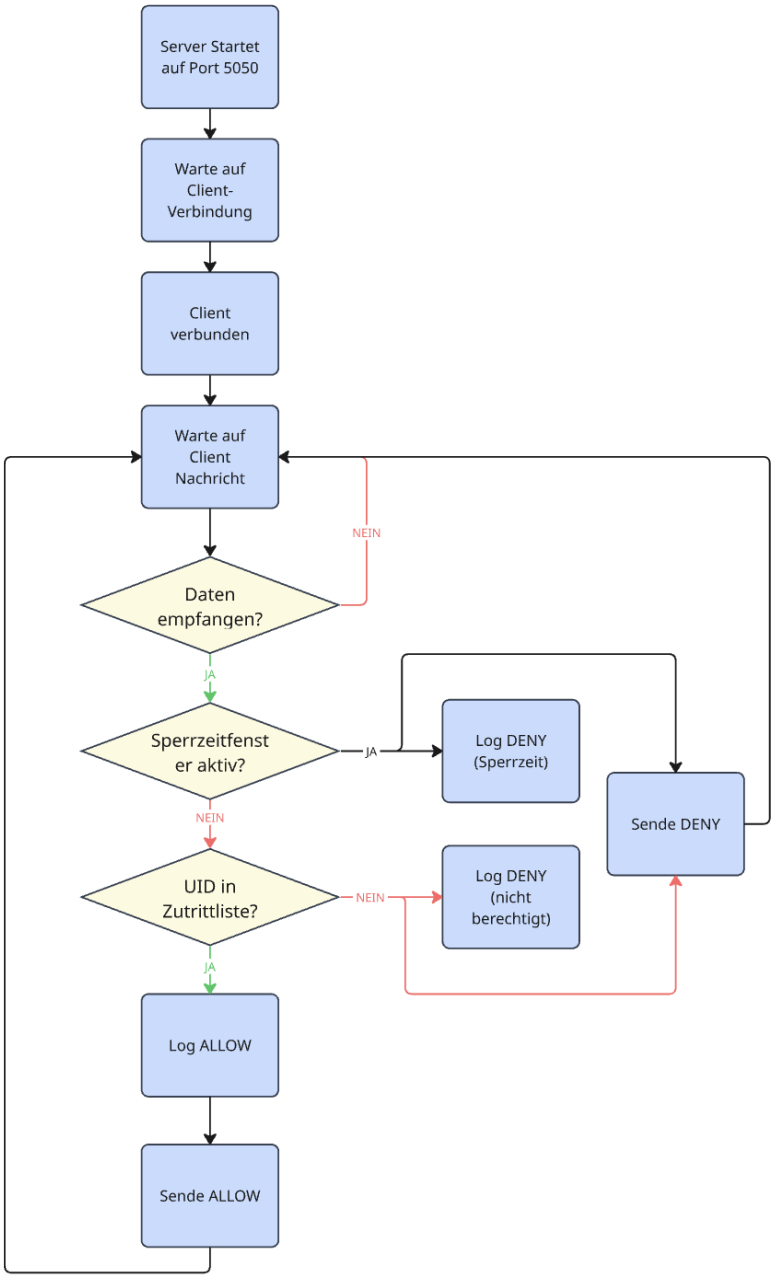

Abbildung 5: Flowchart Server [eigene Darstellung]

Der Server ist in Python implementiert und läuft auf einem PC. Er bildet das zentrale Element der Zutrittskontrolle.

3.1.1 Konfiguration

Der Server lauscht auf allen Netzwerk-Interfaces (0.0.0.0) auf Port 5050. Die Zutrittsliste enthält die berechtigten Personalnummern als hexadezimale UID-Strings:

```
userID = ['F39A370E', '20047935', '00220394', '72349395']
```

3.1.2 Sperrzeitfenster

Der Server unterstützt zeitbasierte Zugangsbeschränkungen. Das Sperrzeitfenster kann über die CLI konfiguriert werden und blockiert während der definierten Zeit alle Zutrittsversuche, unabhängig von der Berechtigung. Es werden auch Zeitfenster über Mitternacht hinweg unterstützt.

3.1.3 Multi-Threading

Der Server verwendet Threads, um mehrere Clients gleichzeitig zu bedienen. Für jeden verbundenen Client wird ein separater Thread gestartet, der die Kommunikation abwickelt. Dies ermöglicht den gleichzeitigen Betrieb mehrerer Zutrittspunkte.

3.1.4 CLI-Befehle

Tabelle 8: Übersicht der Server CLI-Befehle

Befehl	Funktion	Beispiel
update_local_list	Sendet Zutrittsliste an alle ESP32	
set_lock_start(h, m)	Setzt Beginn des Sperrzeitfensters	set_lock_start(13, 30) ➢ Beginn 13:30 Uhr
set_lock_end(h, m)	Setzt Ende des Sperrzeitfensters	set_lock_end(14, 15) ➢ Ende 14:15 Uhr
clear_lock	Entfernt Sperrzeitfenster	
status	Zeigt aktuellen Serverstatus	
exit	Beendet den Server	
help	Listet alle Befehle auf	

3.2 Client

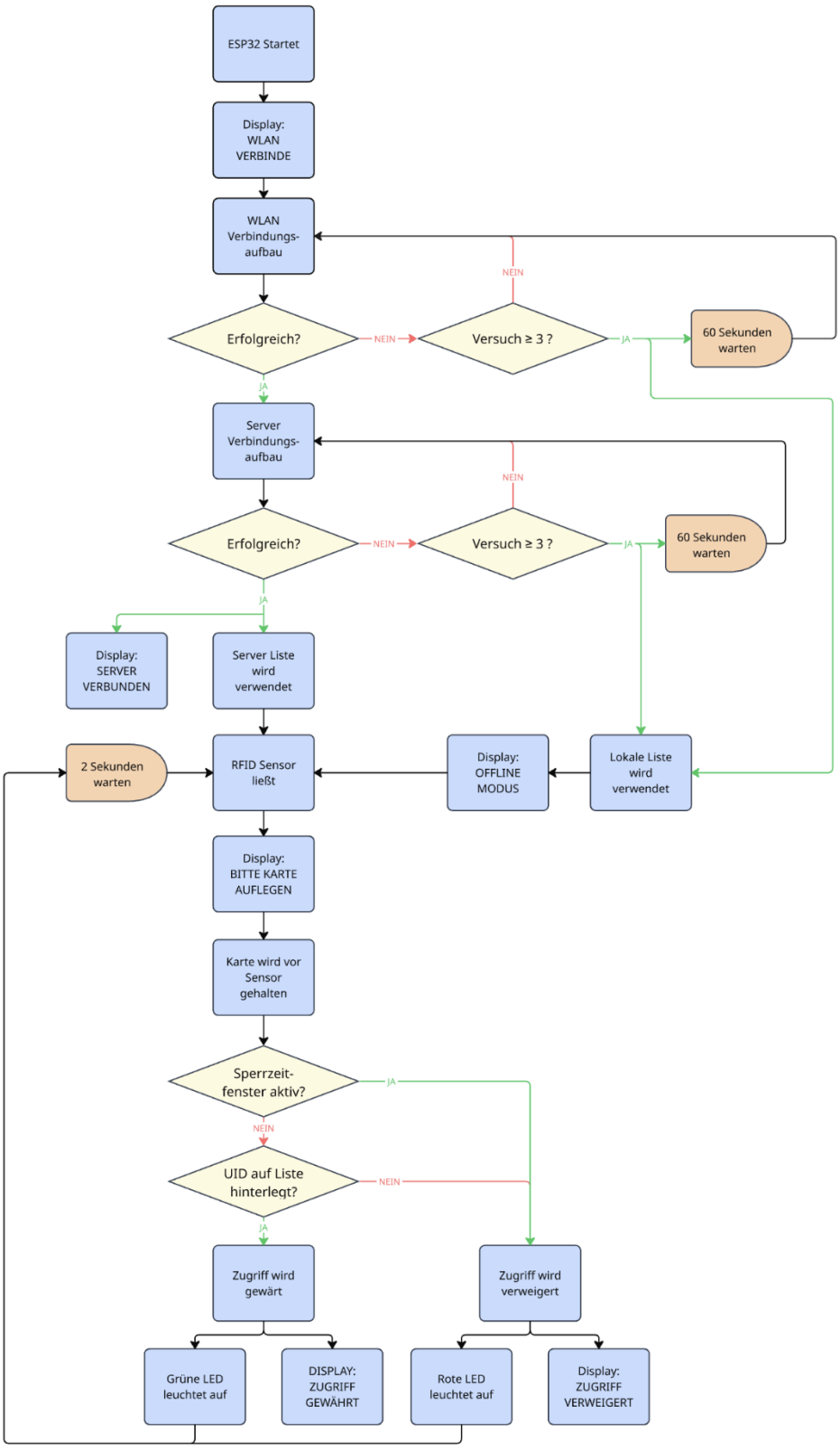

Abbildung 6:Flowchart Client [eigene Darstellung]

Der Client läuft auf dem ESP32 unter MicroPython und besteht aus mehreren Modulen.

3.2.1 Modulübersicht

Tabelle 9: Modulübersicht des Clients

Modul	Funktion
client_main.py	Hauptprogramm, WLAN-Verbindung
TCP_client.py	TCP-Kommunikation mit Server
rfid_reader.py	Abstraktion des RFID-Readers
mfrc522.py	Low-Level-Treiber für MFRC522
tft_display.py	Display- und LED-Ansteuerung
eeprom_storage.py	Flash-Speicher für lokale Liste
wlan_connect.py	WLAN-Verbindungsmanagement

3.2.2 Programmablauf

Das Hauptprogramm steuert den Systemstart: Zunächst wird versucht, eine WLAN-Verbindung herzustellen (bis zu 3 Versuche). Bei Erfolg wird der Online-Modus mit TCP-Client gestartet, andernfalls der Offline-Modus mit lokaler Liste.

3.2.3 TCP-Client Zustandsmaschine

Der TCP-Client implementiert eine Zustandsmaschine für Online/Offline-Betrieb. Im Offline-Modus werden periodisch (alle 60 Sekunden) Reconnect-Versuche unternommen. Bei erfolgreicher Verbindung wechselt das System automatisch in den Online-Modus.

3.2.4 EEPROM-Speicher

Die lokale Zutrittsliste wird im Flash-Speicher des ESP32 als JSON-Datei gesichert. Diese Liste wird vom Server über das UPDATE_LIST-Kommando synchronisiert und ermöglicht den Offline-Betrieb bei Netzwerkausfall.

4 TCP

4.1 Grundlagen

TCP (Transmission Control Protocol) ist ein verbindungsorientiertes Transportprotokoll der Schicht 4 im OSI-Modell. Es garantiert: Zuverlässige Übertragung (Fehlerkorrektur durch Neuübertragung), Reihenfolgetreue der Pakete und Flusskontrolle.

Im Gegensatz zu UDP bietet TCP eine gesicherte Punkt-zu-Punkt-Verbindung, was für die Zutrittskontrolle essentiell ist, denn eine verlorene „ALLOW"-Nachricht könnte sonst zu unberechtigtem Zutrittsentzug führen.

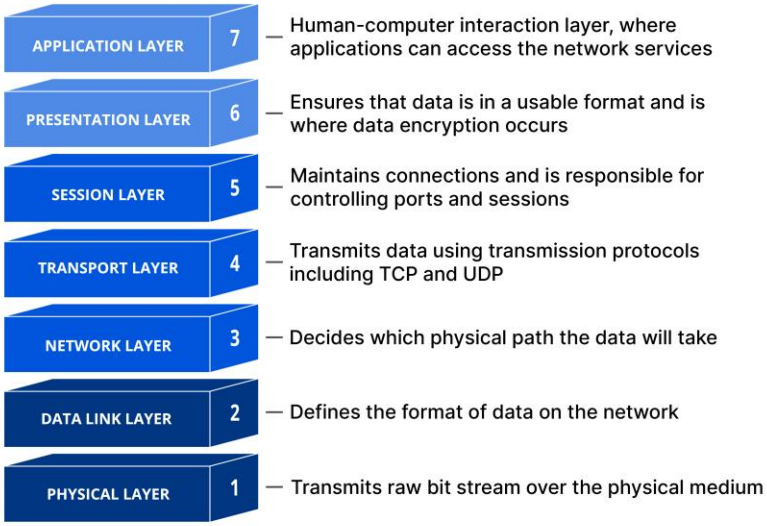

Abbildung 7: OSI-Modell [2]

4.2 Socket-Programmierung

Die Socket-Schnittstelle abstrahiert die TCP/IP-Kommunikation auf Anwendungsebene.

Begriff	Bedeutung
Host	Kommunikationsteilnehmer (PC, ESP32)
Server	Bietet Dienst an, wartet auf Verbindungen

Client	Fordert Dienst an, initiiert Verbindung
Port	16-Bit-Zahl zur Identifikation des Dienstes
Socket	Datenstruktur mit allen Verbindungsinformationen

4.3 Verbindungsaufbau (3-Way-Handshake)

TCP etabliert Verbindungen über einen dreistufigen Handshake: Der Client sendet ein SYN-Paket, der Server antwortet mit SYN+ACK, und der Client bestätigt mit ACK. Danach ist die Verbindung etabliert.

4.4 Protokoll-Design

Die Kommunikation zwischen ESP32 und Server folgt einem einfachen Textprotokoll:

- Client → Server: RFID-UID als Hex-String (z.B. "F39A370E")
- Server → Client: "ALLOW" (Zutritt gewährt) oder "DENY" (Zutritt verweigert)
- Server → Client: "UPDATE_LIST:id1,id2,id3,..." (Neue lokale Liste)

4.5 Beispielhafte Kommunikation zwischen Client und Server

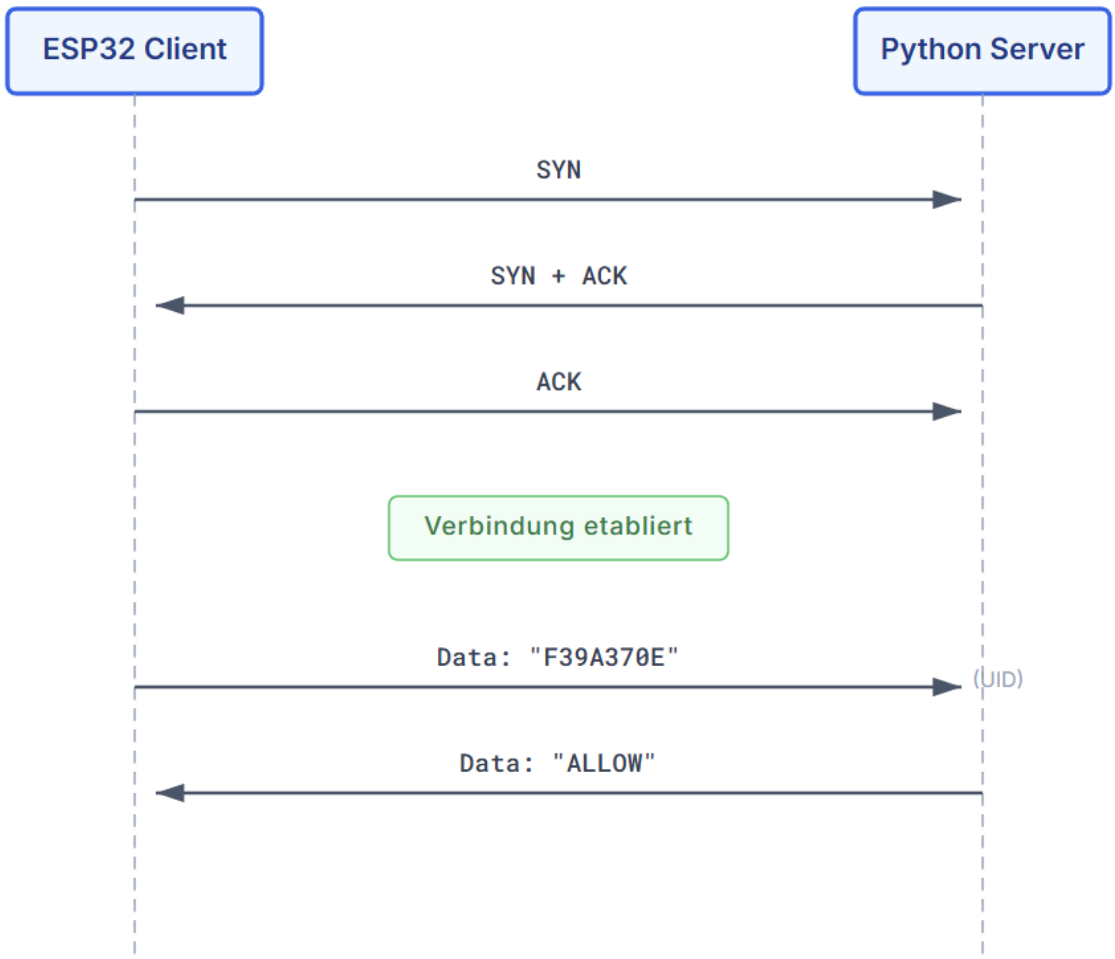

Abbildung 8: Bsp. TCP Kommunikation [eigene Darstellung]

4.6 Fehlerbehandlung

Tabelle 10: Fehlerbehandlung bei TCP

Fehler	Ursache	Reaktion
socket.timeout	Keine Antwort innerhalb 5s	Wechsel zu Offline-Modus
ConnectionResetError	Server/Netzwerk unterbrochen	Reconnect-Versuch
OSError	Allgemeiner Netzwerkfehler	Fallback auf lokale Liste

5 SPI

5.1 Grundlagen

SPI (Serial Peripheral Interface) ist ein synchroner serieller Bus für die Kommunikation zwischen Mikrocontrollern und Peripheriegeräten. Im Gegensatz zu TCP/IP arbeitet SPI auf der physikalischen Ebene (Schicht 1-2 des OSI-Modells, siehe Abbildung 7).

Tabelle 11: SPI Übersicht

Eigenschaft	Wert
Topologie	Master-Slave
Übertragung	Vollduplex (4 Leitungen) oder Halbduplex (3 Leitungen)
Geschwindigkeit	Bis zu mehrere MHz
Adressierung	Über separate CS-Leitung pro Slave
Transportsicherung	Keine

5.2 Signale

SPI verwendet vier Hauptsignale:

Tabelle 12: SPI Signale

Signal	Alternativnamen	Funktion
SCK	SCLK, CLK	Taktsignal vom Master
MOSI	SDO (Master)	Daten: Master → Slave
MISO	SDI (Master)	Daten: Slave → Master
CS/SS	nCS, nSS	Chip Select, aktiviert Slave

5.3 SPI-Modi (CPOL/CPHA)

SPI definiert vier Betriebsmodi durch Kombination von CPOL (Clock Polarity) und CPHA (Clock Phase). In diesem Projekt wird Modus 0 (CPOL=0, CPHA=0) für beide SPI-Geräte verwendet.

5.4 MFRC522 SPI-Kommunikation

Der RFID-Reader verwendet ein Register-basiertes Protokoll. Bei Schreiboperationen wird zuerst ein Adressbyte gesendet (Bit 7=0 für Schreiben, Bits 6-1 für Registeradresse), gefolgt vom Datenbyte. Bei Leseoperationen wird Bit 7 auf 1 gesetzt und das Ergebnis zurückgelesen.

5.5 ST7735 Display-Ansteuerung

Das Display nutzt ebenfalls SPI, jedoch mit separatem D/C-Signal (Data/Command). Bei DC=LOW werden Kommandos gesendet (z.B. Cursor setzen), bei DC=HIGH Daten (z.B. Pixelfarben).

5.6 Dual-SPI-Konfiguration

Da MFRC522 und ST7735 unterschiedliche Taktfrequenzen benötigen, verwendet das Projekt zwei separate SPI-Busse:

Gerät	SPI-Bus	Taktrate	Modus
MFRC522	SPI 1	1 MHz	Mode 0
ST7735	SPI 2	20 MHz	Mode 0

6 Funktionen

Bitte scannen Sie den QR Code ein oder folgen Sie dem Link für eine Demonstration der Funktionen (siehe Abbildung 1)

https://www.youtube.com/watch?v=A04lVECOikc

Abbildung 9: Demo YouTube Video [eigene Darstellung]

6.1 RFID-Kartenlesung

Die UID-Extraktion erfolgt in mehreren Schritten:

1. 1. REQA (Request): Weckt Karten im Lesefeld auf
2. 2. Antikollision: Verhindert Konflikte bei mehreren Karten
3. 3. UID-Extraktion: 4 Bytes werden gelesen und hexadezimal formatiert

6.2 Zutrittsprüfung

Die Prüflogik folgt einer definierten Hierarchie:

4. 1. Ist ein Sperrzeitfenster aktiv? → JA: DENY
5. 2. Ist die UID in der Berechtigungsliste? → JA: ALLOW, NEIN: DENY

6.3 Display-Anzeigen

Zustand	Anzeige	LED
Warten	"Bitte Karte auflegen"	-
Zutritt gewährt	"ZUTRITT GEWÄHRT"	Grün
Zutritt verweigert	"ZUTRITT VERWEIGERT"	Rot
Offline-Modus	"OFFLINE MODUS"	-
WLAN-Verbindung	"WLAN Verbinde..."	-
Server verbunden	"SERVER VERBUNDEN"	-

6.4 LED-Türsimulation

Die LEDs visualisieren den Zutrittsstatus für 2 Sekunden: Grüne LED bei gewährtem Zutritt, rote LED bei verweigertem Zutritt.

6.5 Offline-Fallback

Bei Netzwerkausfall wechselt das System automatisch in den Offline-Modus. Die Prüfung erfolgt dann gegen die lokale Flash-Liste. Der Wechsel ist für den Benutzer transparent – nur die Statusanzeige ändert sich von "ONLINE" auf "OFFLINE".

6.6 Listensynchronisation

Der Server kann über den Befehl "update_local_list" die aktuelle Berechtigungsliste an alle verbundenen ESP32-Clients senden. Die Clients speichern diese Liste persistent im Flash und können sie im Offline-Modus nutzen.

6.7 WLAN-Verbindung

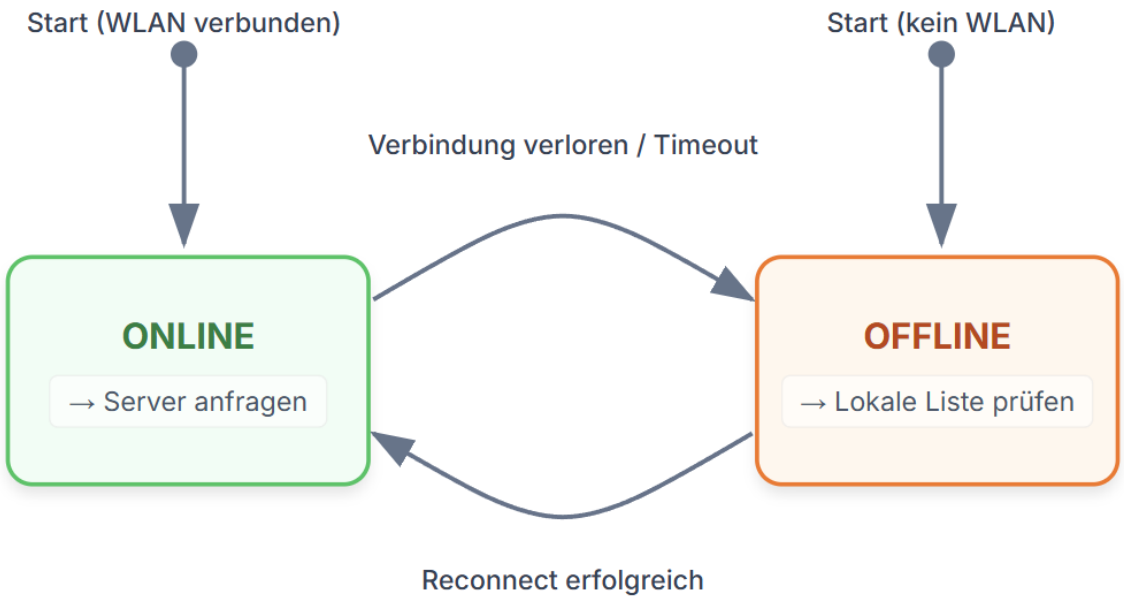

Abbildung 10: Retry-Logik [eigene Darstellung]

Die WLAN-Verbindung erfolgt mit Retry-Logik (3 Versuche) und angepasster Sendeleistung (txpower=5), um Probleme mit der Antennenanpassung zu vermeiden. Bei Verbindungsverlust werden periodische Reconnect-Versuche alle 60 Sekunden durchgeführt.

7 Fazit

7.1 Zusammenfassung

Das entwickelte RFID-Zutrittskontrollsystem erfüllt alle im Lastenheft definierten Anforderungen:

Tabelle 13: Soll-Ist Vergleich Anforderungsliste

Anforderung	Status	Umsetzung
RFID-Lesung	✓ Erfüllt	MFRC522 via SPI, UID als Hex-String
Server-Kommunikation	✓ Erfüllt	TCP/IP über WLAN, Port 5050
Zutrittsprüfung	✓ Erfüllt	Serverseitige Liste mit Sperrzeitfenster
Offline-Modus	✓ Erfüllt	Lokale JSON-Liste im Flash
Visuelle Rückmeldung	✓ Erfüllt	TFT-Display + LEDs
Synchronisation	✓ Erfüllt	UPDATE_LIST-Kommando

7.2 Verwendete Protokolle

Protokoll	OSI-Schicht	Anwendung
TCP/IP	Schicht 3-4	Client-Server-Kommunikation
SPI	Schicht 1-2	RFID-Reader, Display
WLAN (802.11)	Schicht 1-2	Drahtlose Vernetzung

7.3 Mögliche Erweiterungen

Für zukünftige Versionen wären folgende Erweiterungen denkbar:

- Verschlüsselung: TLS/SSL für die TCP-Verbindung
- Authentifizierung: Server-Client-Authentifizierung
- Datenbank: Persistente Speicherung der Zutrittsprotokolle
- Web-Interface: Browser-basierte Verwaltungsoberfläche
- Wechsel auf NFC, um Smartphones zum Entsperren zu nutzen

7.4 Lessons Learned

- WLAN-Stabilität: Die Sendeleistung des ESP32 muss oft reduziert werden, um stabile Verbindungen zu erreichen
- SPI-Timing: Unterschiedliche Slaves erfordern oft separate SPI-Konfigurationen
- Offline-Fallback: Ein lokaler Fallback ist für kritische Systeme unerlässlich

8 Anhang

8.1 Pinbelegung (Zusammenfassung)

Tabelle 14: Pinouts

Gerät	Signal	ESP32 GPIO
MFRC522	CS/SDA	13
MFRC522	SCK	14
MFRC522	MOSI	15
MFRC522	MISO	16
MFRC522	RST	17
ST7735	SCK	12
ST7735	SDA	11
ST7735	CS	10
ST7735	DC	4
ST7735	RST	5
LED Grün	-	6
LED Rot	-	7

8.2 Open Source Code auf GitHub

https://github.com/Awzeyy/AccessControl_Projekt_Datenkommunikation

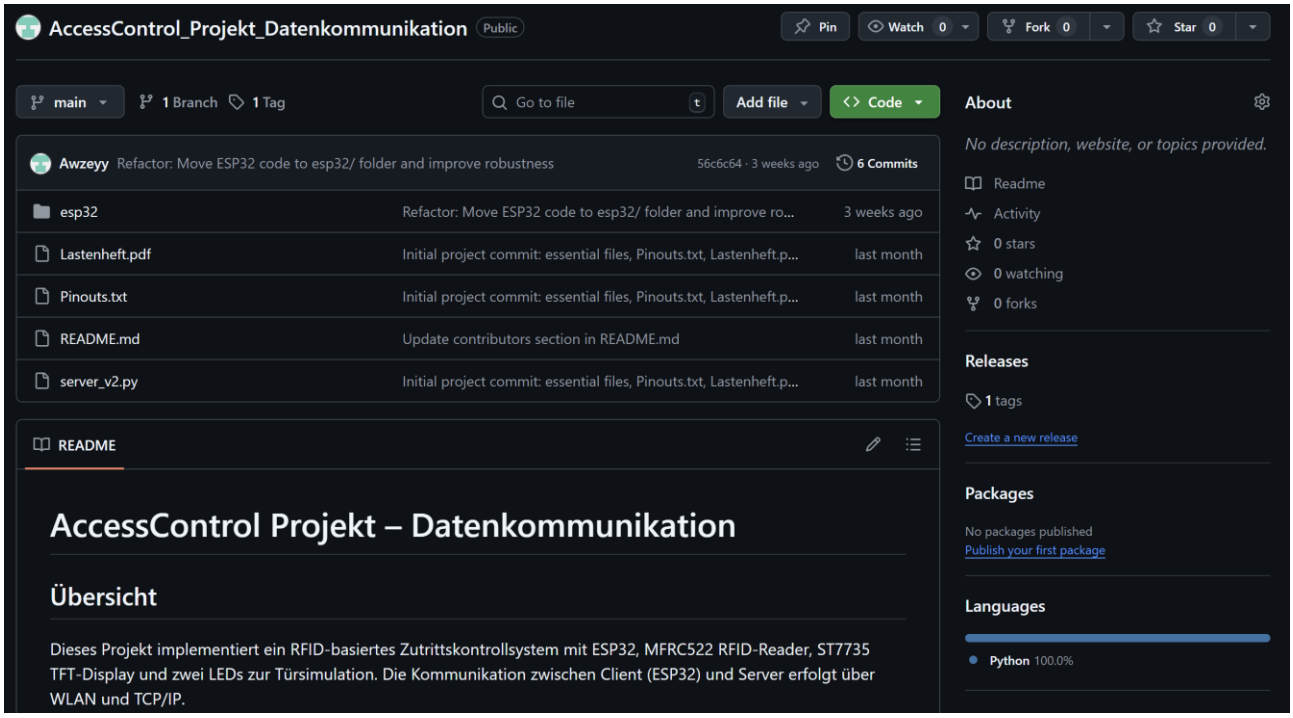

Abbildung 11: Projektseite auf Github [4]

9 Literaturverzeichnis

[1] butterflymx.com, „Abbildung Türschloss mit Hand,“ [Online]. Available: https://butterflymx.com/blog/key-card-door-locks/.

[2] A. Ettl und T. Welzel, *Lastenheft Projektarbeit Datenkommunikation WS 25/26*.

[3] Cloudflare, „Was ist das OSI-Modell?,“ [Online]. Available: https://www.cloudflare.com/de-de/learning/ddos/glossary/open-systems-interconnection-model-osi/.

[4] A. Ettl und T. Welzel, „Github Projekt Datenkommunikation,“ [Online]. Available: https://github.com/Awzeyy/AccessControl_Projekt_Datenkommunikation.