Documentație Proiect: Sistem de Control Acces cu ESP32

Purenciu Diana

12 ianuarie 2026

1 Descriere Generală

Acest proiect este un sistem de securitate pentru controlul accesului, gândit să înlocuiască cheile clasice cu metode digitale moderne. Practic, am construit un dispozitiv care permite deschiderea unei uși fie prin introducerea unui cod PIN pe o tastatură, fie prin scanarea amprentei.

Proiectul funcționează pe baza unei comunicări prin internet. Când un utilizator încearcă să intre, ESP32 citește datele (codul sau amprenta) și le trimite imediat către un server central, care rulează pe un calculator.

Serverul este cel care ia decizia finală: verifică într-o bază de date dacă utilizatorul există și dacă codul este corect. După verificare, serverul trimite răspunsul înapoi la ESP32. Dacă accesul este permis, placa afișează un mesaj pe ecranul OLED și aprinde un LED verde. În caz contrar, accesul este respins și se aprinde un LED roșu.

2 Lista Componentelor Hardware

Pentru realizarea montajului fizic au fost utilizate următoarele componente:

- **Placă de dezvoltare:** ESP32 DevKit V1
- **Interacțiune utilizator:** Tastatură matricială 4x4 (Keypad)
- **Afișaj:** Ecran OLED SSD1306 (0.96 inch, interfață I2C)
- **Biometrie:** Senzor optic de amprentă (comunicare Serială)
- **Feedback:** LED Verde (Succes), LED Roșu (Eroare), Buzzer pasiv
- **Elemente pentru conexiune:** Breadboard, fire de legătură, cablu Micro-USB
- **Server:** Laptop/PC pentru rularea aplicației Node.js

3 Funcționalități și Implementare

Sistemul funcționează pe baza unui concept de **Automat de Stări** (State Machine). În cod, am folosit o variabilă globală `mode` care dictează comportamentul plăcuței: 0 (Așteptare), 1 (Logare PIN), 3-5 (Pașii de înregistrare). ESP32 funcționează ca un Client, conectându-se la serverul Node.js pentru a valida datele.

3.1 1. Autentificarea cu cod PIN

- **Funcționare:** Utilizatorul tastează 4 cifre. La fiecare apăsare, cifra este adăugată într-un șir de caractere (*String*) și afișată pe ecran.
- **Implementare tehnică:** Când șirul atinge lungimea de 4, ESP32 construiește manual un pachet JSON (prin concatenare de text: `"{"pin":"1234"}"`) și îl trimite prin metoda `POST` la ruta `/auth`. Răspunsul serverului este citit ca text simplu, iar dacă funcția `indexOf` găsește cuvântul "true" în răspuns, accesul este permis (LED Verde).

3.2 2. Autentificarea cu Amprentă

- **Funcționare:** În meniul principal, tasta 'A' activează citirea. Utilizatorul are 5 secunde să pună degetul.
- **Implementare tehnică:** ESP32 comunică cu senzorul optic prin protocolul Serial (pinii RX2/TX2). Funcția `getFingerprintID` interoghează senzorul dacă a găsit o potrivire în memoria sa internă. Dacă da, senzorul returnează un ID (număr întreg). Acest ID este trimis la server (`/auth`), care verifică în baza de date SQL cui aparține amprenta respectivă.

3.3 3. Înregistrarea unui utilizator nou

- **Pasul 1 (Verificare OTP):** Utilizatorul introduce un cod unic. ESP32 îl trimite la `/verify-otp`. Dacă serverul confirmă, variabila `mode` se schimbă, trecând la pasul următor.
- **Pasul 2 (Setare PIN):** Se introduce noul PIN. Serverul îl salvează și returnează un `user_id`.
- **Parsare:** Deoarece serverul returnează ID-ul noului utilizator în format JSON, am implementat o funcție de parsare manuală care caută textul `"user_id":` în răspuns și extrage numărul care urmează, salvându-l în variabila `currentUserId` pentru a fi folosit la pasul 3.
- **Pasul 3 (Înrolare Amprentă):** ESP32 comandă senzorului să facă două scanări succesive și să memoreze amprenta exact la locația indicată de `currentUserId`. La final, serverul este notificat pentru a face legătura în baza de date.

3.4 4. Procesarea pe Server (Node.js)

Codul pentru server este scris în JavaScript folosind mediul Node.js și biblioteca Express pentru a gestiona cererile web.

Fluxul de date pe server este următorul:

- **Primirea datelor:** Serverul ascultă permanent pe portul 3000. Când primește un pachet `POST` (de exemplu, un PIN), îl "despachetează" pentru a citi conținutul JSON.

- **Verificarea în Baza de Date:** Datele nu sunt stocate în fișiere text, ci într-o bază de date relațională SQLite. Serverul execută o comandă SQL (de exemplu: `SELECT * FROM users WHERE pin = '1234'`).
- **Decizia:**
  - Dacă baza de date găsește o potrivire, serverul trimite înapoi un răspuns JSON: { `"success": true` }.
  - Dacă nu găsește nimic, trimite: { `"success": false` }.
- Astfel, ESP32 nu trebuie să știe SQL sau logică complexă; el doar așteaptă un simplu "DA" (true) sau "NU" (false) de la server pentru a aprinde LED-ul corespunzător.

3.5 Biblioteci și Resurse Utilizate

1. **Adafruit_SSD1306**: Folosita pentru controlul ecranului OLED și afișarea textului.
2. **Adafruit_Keypad**: Bibliotecă pentru gestionarea matricei de taste (citirea rândurilor și coloanelor).
3. **Adafruit_Fingerprint**: Bibliotecă pentru comunicarea Serială cu senzorul de amprenta.
4. **WiFi.h** și **HTTPClient.h**: Biblioteci folosite pentru conectarea la rețea și efectuarea cererilor HTTP.

Referințe și Tutoriale:

- **Îndrumător Laborator:**:
  `https://mihai.utcluj.ro/wp-content/uploads/dmp/labs/pmp-lab08_new.pdf`
- **Tastatură:**:
  `https://cdn-learn.adafruit.com/downloads/pdf/matrix-keypad.pdf`
- **Ecran OLED:**:
  `https://randomnerdtutorials.com/esp32-ssd1306-oled-display-arduino-ide/`
- **Senzor Amprentă (Sursă 1):**:
  `https://learn.adafruit.com/adafruit-optical-fingerprint-sensor`
- **Senzor Amprentă (Sursă 2):**:
  `https://www.youtube.com/watch?v=rFhhA5h2M6c`
- **Server (Node.js & SQLite):**:
  `https://www.sqlitetutorial.net/sqlite-nodejs/`
- **Server (Node.js & SQLite & Express):**:
  `https://www.youtube.com/watch?v=Hej48pi_lOc`

4 Schema Electrică

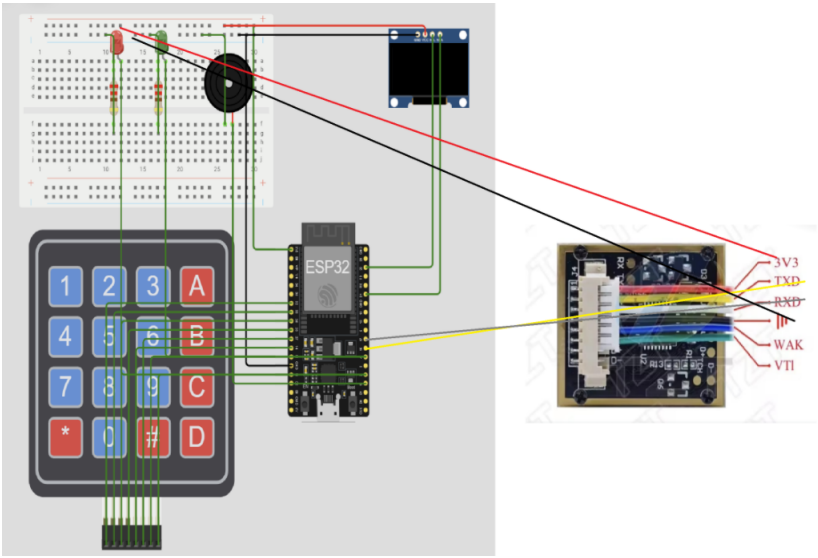

Figura 1: Diagrama de conectare a componentelor în Fritzing