

# Smart Lock System

- Sindrilaru Catalina-Maria
- 332CA

## Introducere

Acest proiect implementează o încuietoare inteligentă cu verificare biometrică. Această încuietoare se bazează pe citirea unei amprente și verificarea acesteia pentru controlul accesului într-un spațiu restricționat.

Sistemul pe care am ales să îl creez propune o soluție de securitate ce poate fi integrat în multiple aplicații și medii, de la laboratoare și birouri, la încăperi de depozitare și chiar locuințe.

Scopul proiectului este de a oferi o soluție de securitate și acces controlat, utilizând autentificarea prin amprentă și cod de securitate.

## Descriere generală

Proiectul utilizează un cod de securitate (introdus de administratorul sistemului) pentru a permite înregistrarea unei amprente. După introducerea corectă a codului, utilizatorul este ghidat de mesaje afișate pe ecranul LCD pentru a înregistra o nouă amprentă.

După înregistrarea unei amprente, utilizatorul poate accesa spațiul securizat prin așezarea amprenteii pe cititor.

Proiectul permite înregistrarea mai multor amprente și are o funcționalitate de securitate încorporată, activând un buzzer în cazul în care sunt introduse amprente greșite de trei ori consecutiv. Administratorul are posibilitatea de a șterge toate amprentele înregistrate în sistem, prin introducerea unui alt cod de securitate.



## Hardware Design

## Lista de piese

- Arduino UNO
- BreadBoard
- LCD cu interfata I2C
- Cititor de amprenta cu interfata UART
- Tastatura 4x4
- Buzzer pasiv
- Servo-motor
- Fire

## Design-ul circuitului



## Schema electrica



## Software Design

- Mediu de dezvoltare utilizat: Arduino IDE
- Librării și surse 3rd-party: LiquidCrystal\_I2C.h (pentru LCD), Keypad.h (pentru tastatura), Servo.h (pentru servo-motor), [Adafruit\\_Fingerprint.h](#) (pentru cititorul de amprente)

## Implementare

Complete code on Github: [todo](#)

Pentru a oferi o experienta cat mai buna utilizatorului, am folosit display-ul LCD cu interfata I2C pentru a afisa mesaje ce au ca scop informarea si ghidarea acestuia.

Sistemul verifica in permanenta daca a fost introdus un cod nou. Acest lucru are la baza tastatura, initializata la inceputul programului, 2 coduri de referinta si un sir gol.

```
const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
```

```

    {'*', '0', '#', 'D'}
};

byte rowPins[ROWS] = {13, 12, 10, 9};
byte colPins[COLS] = {8, 7, 6, 5};

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
char correctCodeRegister[] = "147258";
char enteredCode[7] = "";
char correctCodeDelete[] = "258369";

```

Funcția de verificare cod citește un caracter de la tastatură, iar când tasta # este apăsată, verifică validitatea codului. Astfel, tastatura este folosită pentru a pune sistemul în 2 moduri, cel de înregistrare a unei noi amprente și cel de ștergere a tuturor amprentelor ce au fost înregistrate. Astfel, dacă codul introdus este cel corespunzător modului de înregistrare amprentă sau de golire a bazei de date, pe ecran se afișează **Correct Code!** și se începe citirea amprentei sau a golirii bazei de date, după caz. În schimb, dacă codul introdus nu este corect, se afișează **Incorrect Code!**. După ce un # a fost apăsat, șirul ce reține codul tastat se golește.

```

void verify_code() {
    char key = keypad.getKey();
    if (key) {
        if (key == '#') {
            if (strcmp(enteredCode, correctCodeRegister) == 0) {
                registerMode();
            } else if (strcmp(enteredCode, correctCodeDelete) == 0) {
                deleteMode();
            } else {
                incorrectMode();
            }
            memset(enteredCode, 0, sizeof(enteredCode));
        } else {
            if (strlen(enteredCode) < 6) {
                strncat(enteredCode, &key, 1);
            }
        }
    }
}

```

Pentru utilizarea cititorului de amprentă, am folosit biblioteca [Adafruit\\_Fingerprint.h](#), împreună cu câteva dintre funcțiile pe care le pune la dispoziție, de stocare a unei amprente, de golire a bazei de date pentru amprente, precum și de verificare dacă o amprentă a fost deja înregistrată sau nu.

```

SoftwareSerial softwareSerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&softwareSerial);
uint8_t id = 1; // first id

```

Astfel, după tastarea corectă a codului pentru înregistrarea unei amprente, se afișează pe ecran mesajul **Enrolling..** și se apelează funcția din biblioteca pentru înrolarea unei noi amprente. Funcția de înrolare folosește un id caruia să îi asocieze amprenta pe care o va înregistra (poate înregistra până la 127 de amprente), așa că am folosit un id care porneste de la 1 și crește de fiecare dată când

o amprenta noua este inregistrata. Aceasta functie presupune asezarea aceluasi deget pe cititor de 2 ori, pentru a verifica corectitudinea amprenteii. Pentru acest proces, am afisat mesaje care sa ghideze utilizatorul, precum **“Place finger!”**, **“Remove finger!”**, **“Place same finger again”**, **“Fingerprints did not match. Introduce code again”**, **“Fingerprints match”**, **“Fingerprint stored”**.

```
void enrollFinger() {
    Serial.println("Enrolling...");
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Enrolling...");
    delay(1000);
    while (!getFingerprintEnroll());
    id++;
    delay(2000);
}
```

Sistemul verifica constant daca a fost asezata o amprenta pe cititor, folosind functia din biblioteca. Astfel, daca citeste o amprenta pe care o are deja stocata in baza de date, afiseaza **“Found a print match!”** si intra in modul de deschidere a usii, iar daca amprenta nu exista in baza de date afiseaza **“Did not find a match!”**, iar usa ramane inchisa.

De fiecare data cand o amprenta gresita este detectata de cititor, se incrementeaza un contor, care se reseteaza atunci cand o amprenta corecta a fost introdusa. Atunci cand sunt introduse 3 amprente gresite (in mod consecutiv), se activeaza buzzerul, ce simuleaza o alarma, timp de 3 secunde.

```
if (failedFingerprintsConsec == 3) {
    failedFingerprintsConsec = 0;
    activateBuzzer();
}
```

```
void activateBuzzer() {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("3 unsuccessful");
    lcd.setCursor(0,1);
    lcd.print("attempts!");

    analogWrite(buzzerPin, 230);
    delay(1000);
    analogWrite(buzzerPin, 0);
    delay(500);
    analogWrite(buzzerPin, 230);
    delay(1000);
    analogWrite(buzzerPin, 0);
    delay(500);
    analogWrite(buzzerPin, 230);
    delay(1000);
    analogWrite(buzzerPin, 0);

    lcd.clear();
}
```

```
}
```

În modul de deschidere a usii, pe care am simulat-o cu ajutorul unui servo-motor (setat inițial la poziția 0), am afișat mesaje precum **“Opening door...”**, **“Door is open!”**, **“Closing door...”**, **“Door is closed!”**. Pentru a simula deschiderea usii, am schimbat poziția servo-motorului de la 0 la 90. Usa rămâne deschisă timp de 3 secunde.

```
void openDoor() {
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Opening door...");
  delay(1500);

  for (;pos<=90;pos+=10) {
    Servo1.write(pos);
    delay(80);
  }

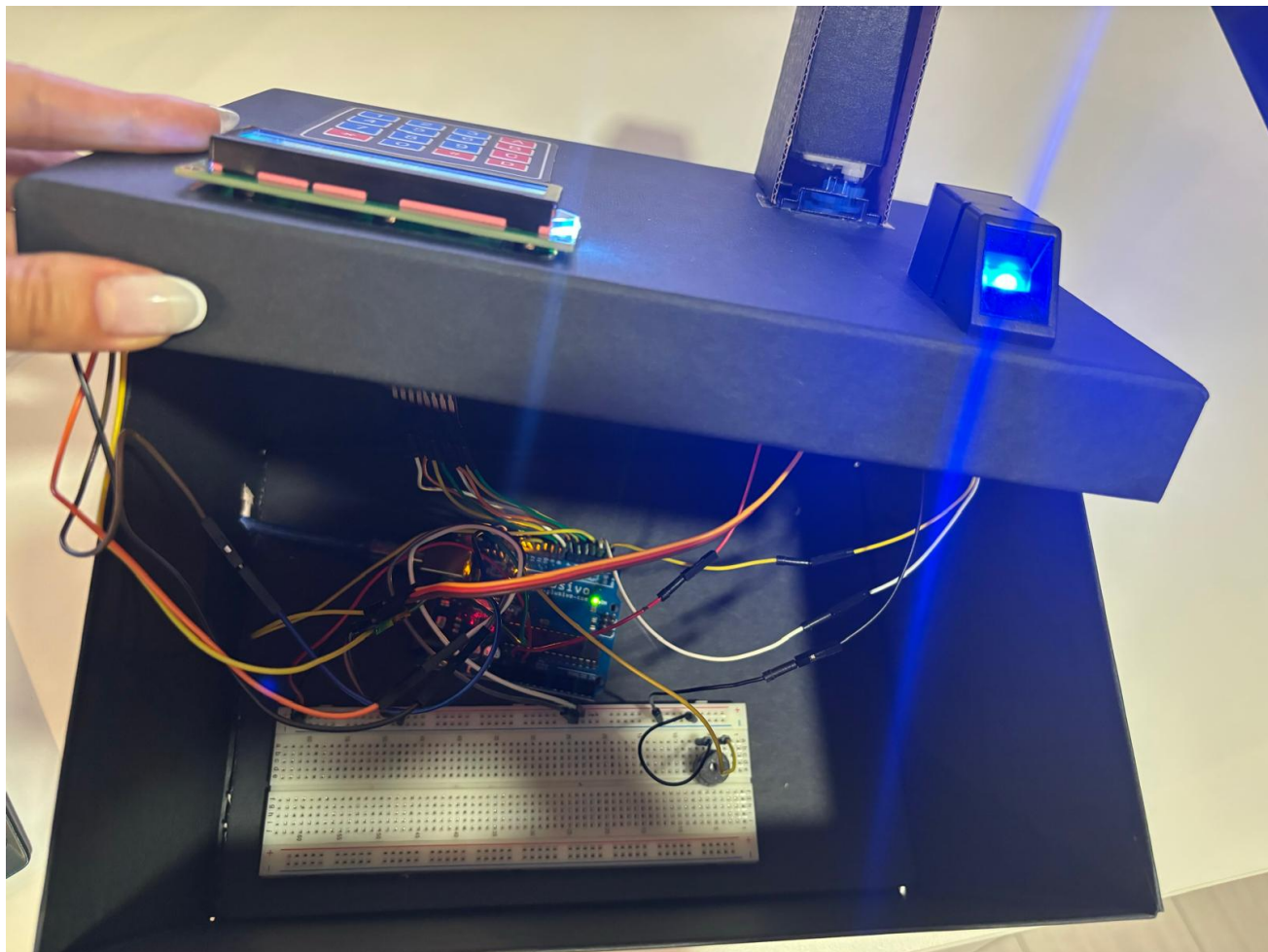
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Door is open!");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Closing door...");
  delay(1500);

  for (;pos>=0;pos-=10) {
    Servo1.write(pos);
    delay(80);
  }

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Door is closed!");
  delay(2000);
  lcd.clear();
}
```

## Rezultate Obținute

Demo: <https://www.youtube.com/watch?v=dleKKA5WVyk>



## Concluzii

Realizarea proiectului mi s-a parut foarte interesanta. Am reusit sa realizez ceea ce mi-am propus. Realizarea design-ului final al proiectului, pentru a avea un aspect placut, ce expune doar piesele ce trebuie sa fie vazute, este cea ce m-a pus cel mai mult in dificultate.

## Download

Github: <https://github.com/CatalinaSindrilaru/Smart-lock-system-Arduino>

[smart\\_lock\\_system\\_sindrilarucatalina.zip](#)

## Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

## Bibliografie/Resurse

- LCD
  - <https://www.geeksforgeeks.org/how-to-interface-i2c-lcd-display-with-arduino/>
  - <https://www.optimusdigital.ro/ro/optoelectronice-lcd-uri/2894-lcd-cu-interfata-i2c-si-backlight-albastru.html>
- Buzzer
  - <https://www.circuitbasics.com/how-to-use-active-and-passive-buzzers-on-the-arduino/>
  - <https://www.circuitgeeks.com/arduino-buzzer-tutorial/>
- Servo-motor
  - <https://www.instructables.com/Arduino-Servo-Motors/>
- Cititor de amprenta
  - <https://ardushop.ro/ro/home/1489-fingerprint-reader.html>
  - <https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library>
- Tastatura
  - <https://circuitdigest.com/microcontroller-projects/interface-4x4-membrane-keypad-with-arduino>
  - <https://www.optimusdigital.ro/ro/senzori-senzori-de-atingere/470-tastatura-matriceala-4x4-cu-conector-pin-de-tip-mama.html>

[Export to PDF](#)

From:

<http://ocw.cs.pub.ro/courses/> - **CS Open CourseWare**

Permanent link:

<http://ocw.cs.pub.ro/courses/pm/prj2024/sseverin/catalina.sindrilaru>



Last update: **2024/05/26 21:13**