Smart Safe Box

Autor: Emilian Horduna

Grupa: 332CD

Introducere

Am realizat un seif electronic care se deschide prin introducerea unui cod PIN. În plus, am integrat un sistem de închidere automată, care blochează ușa seifului la 5 secunde după ce a fost închisă, pentru un plus de siguranță.

Scopul acestui proiect este să creez un design eficient și practic pentru a-mi păstra obiectele importante în siguranță. M-am gândit la un sistem care să fie ușor de folosit, dar și suficient de sigur pentru a preveni accesul neautorizat.

Inițial, pornisem de la ideea de a construi doar un mecanism inteligent de blocare a unei uși, dar pe parcurs m-am decis să dezvolt ceva mai complet și mai util – așa a apărut ideea acestui seif electronic.

Descriere generală

Proiectul constă în realizarea unui seif electronic inteligent controlat de o placă Arduino UNO R3. Seiful oferă o metodă de autentificare pentru deschidere: introducerea unui cod PIN printr-o tastatură matricială (keypad).

După ce ușa seifului este închisă, sistemul va aștepta automat 5 secunde, iar apoi va activa un servomotor care acționează un zăvor mecanic, blocând ușa. Acest mecanism previne uitarea ușii deschise și oferă un plus de securitate.

Pe parcursul utilizării, utilizatorul este asistat de un ecran LCD 1602, care afișează informații relevante precum: introducerea codului, starea de acces, erori sau confirmări. Codul PIN tastat va fi afișat parțial sau complet, în funcție de logica de securitate aleasă.

Pentru a indica starea seifului, sistemul utilizează două LED-uri:

- LED verde – indică faptul că seiful este deblocat;

- LED roșu – indică faptul că seiful este blocat.

În plus, un buzzer piezoelectric emite sunete diferite pentru a semnala:

- introducerea unui cod corect;

- introducerea unui cod greșit;

- blocarea automată a ușii.

Actor → Keypad 4×4 → Arduino UNO R3: Utilizatorul introduce un cod PIN folosind tastatura matricială 4×4. Tastatura este conectată la 8 pini digitali ai plăcii Arduino și transmite caracterele tastate către microcontroler pentru validare.

Arduino UNO R3 → LCD 1602: Pe ecranul LCD sunt afișate mesaje precum „Introduceți codul”, „Acces permis” sau „Cod greșit”. LCD-ul este conectat prin interfața I2C la pinii A4 (SDA) și A5 (SCL).

Arduino UNO R3 → Buzzer pasiv: Buzzerul pasiv este utilizat pentru a oferi feedback sonor (sunet scurt pentru cod corect, ton de eroare pentru cod greșit etc.). Este comandat prin semnal generat cu funcția tone() și conectat pe un pin digital (D10).

Arduino UNO R3 → LED roșu + LED verde: Cele două LED-uri indică starea sistemului: LED-ul roșu este aprins când seiful este blocat, iar LED-ul verde când este deblocat. Ambele LED-uri sunt conectate pe pini digitali și sunt comandați prin semnal simplu GPIO.

Arduino UNO R3 → Servomotor → Zăvor ușă: Servomotorul SG90 este folosit pentru acționarea mecanică a zăvorului ușii seifului. În urma unui cod corect, Arduino trimite un semnal PWM pe pinul D11 către servomotor, care deblochează mecanismul. După 5 secunde, un timer software (implementat prin millis()) declanșează automat reînchiderea ușii prin rotirea servo-ului în poziția de blocare.

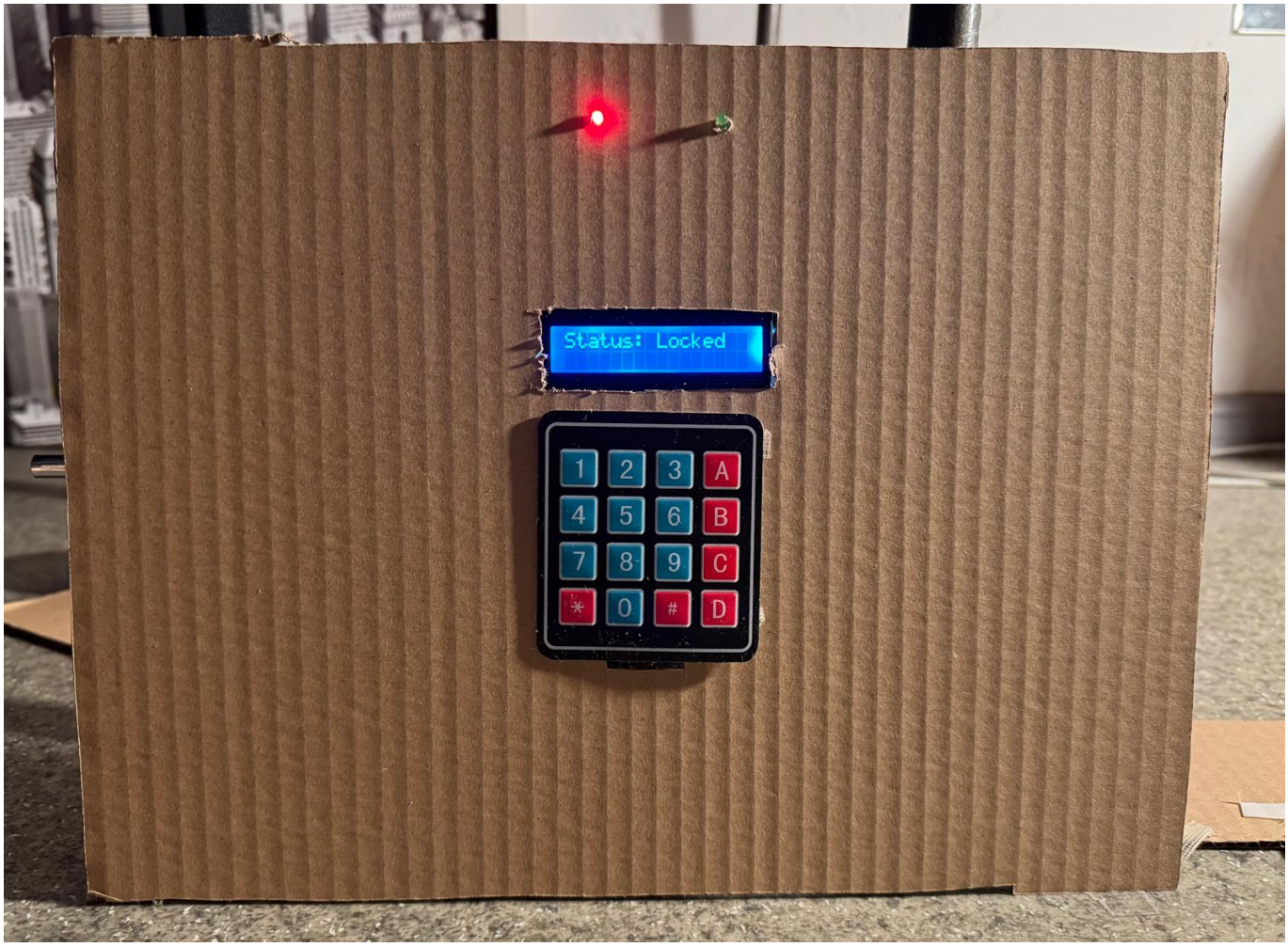
Hardware Design

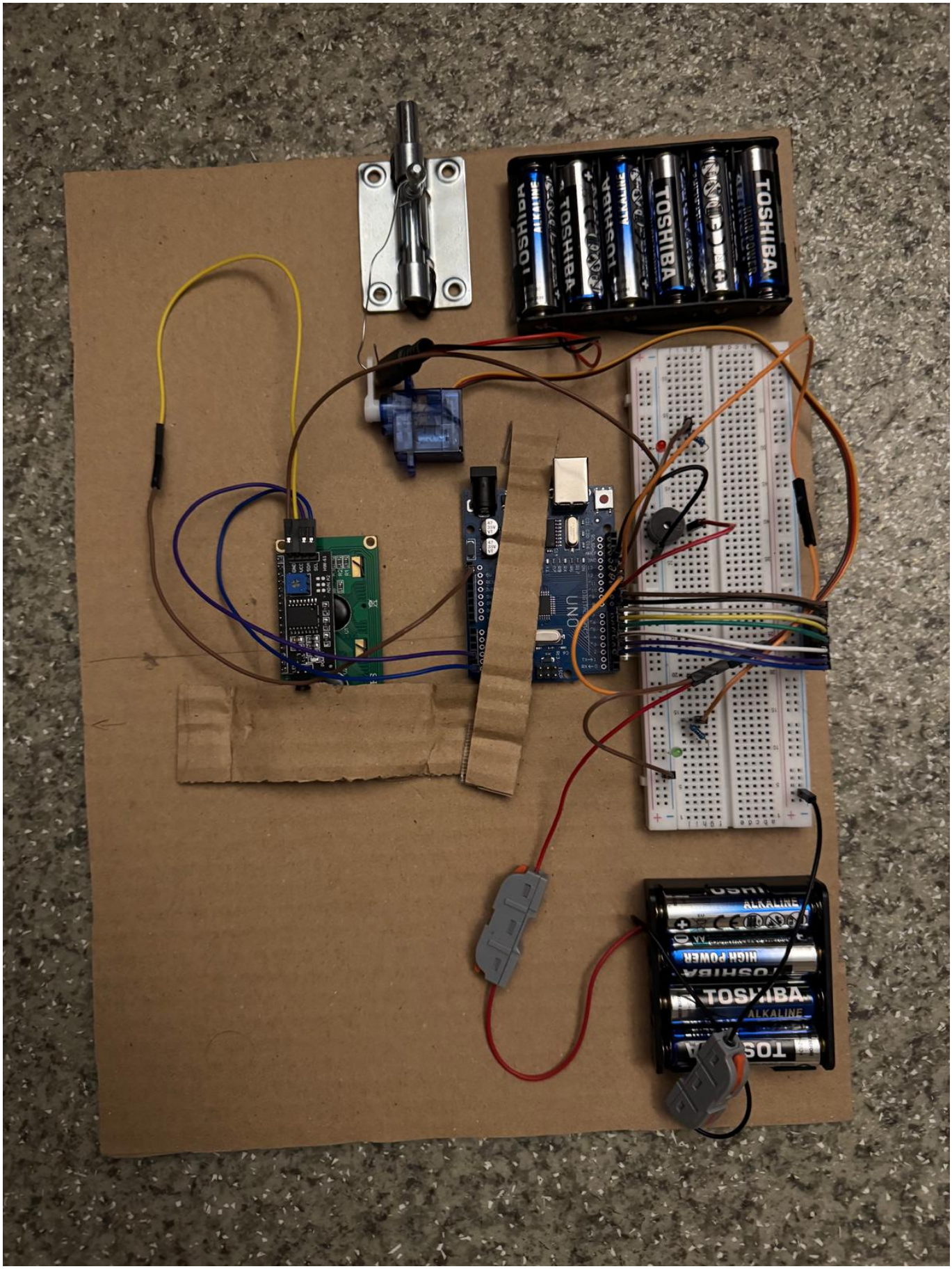
Bill Of Materials:

Nr. crt.	Componentă	Cantitate	Link / Datasheet
1	Arduino UNO R3 ATmega328P	1	[Link](https://ardushop.ro/ro/plci-de-dezvoltare/2282-placa-de-dezvoltare-uno-r3-compatibil-arduino-6427854027122.html)
2	Cablu USB Arduino	1	[Link](https://ardushop.ro/ro/electronica/1167-cablu-usb-a-b-1m-arduino-mega-uno-imprimanta-6427854016515.html)
3	LCD 1602 verde/albastru	1	[Link](https://ardushop.ro/ro/electronica/2305-1306-lcd-1602-verde-albastru.html#/1-culoare-albastru)
4	Modul I2C pentru LCD	1	[Link](https://ardushop.ro/ro/comunicatie/2333-modul-i2c-pentru-lcd-1602-2004-6427854007353.html)
5	Breadboard 830 puncte	1	[Link](https://ardushop.ro/ro/electronica/2297-breadboard-830-puncte-mb-102-6427854012265.html)
6	Tastatură matricială 4×4	1	[Link](https://ardushop.ro/ro/butoane--switch-uri/976-tastatura-matriciala-4x4-6427854013149.html)
7	Kit RFID 13.56 MHz (cititor + card)	1	[Link](https://ardushop.ro/ro/electronica/2309-kit-rfid-1356-mhz-6427854024947.html)
8	Servomotor SG90	1	[Link](https://www.optimusdigital.ro/ro/motoare-servomotoare/26-micro-servomotor-sg90.html)
9	LED-uri 3mm (roșu, verde)	2	[Link](https://ardushop.ro/ro/componente-discrete/64-3-led-3mm.html#/3-culoare-alb)
10	Buzzer piezoelectric activ 3–24V	1	[Link](https://ardushop.ro/ro/componente-electronice/1661-buzzer-piezoelectric-activ-3-24v-hnd-2312-6427854025050.html)

11	Fire de conectare (male-male, male-female)	20	[Link](https://ardushop.ro/ro/9-fire-si-conectori)
12	Rezistențe diverse (220Ω, 1kΩ, 10kΩ)	10	[Link](https://ardushop.ro/ro/cautare?controller=search&s=rezistor)
13	Modul step-down	1	[Link](https://ardushop.ro/ro/display-uri-si-led-uri/1345-modul-indicator-de-tensiune-pentru-acumulatori-6427854019486.html)
14	Suport baterii 18650	1	[Link](https://ardushop.ro/ro/carcase-i-suporturi/47-suport-carcasa-baterii-18650-2buc-6427854018632.html)

Nr. crt.	Componentă	Tip de semnal / protocol	Pini Arduino folosiți
1	Arduino UNO R3 ATmega328P	GPIO, PWM, I2C	—
2	LCD 1602 + Modul I2C	I2C	A4 (SDA), A5 (SCL)
3	Tastatură matricială 4×4	GPIO (Digital Input)	D2, D3, D4, D5, A0, A1, A2, A3
4	Buzzer piezoelectric pasiv	PWM audio / Timere	D10
5	Servomotor SG90	PWM (semnal control servo)	D11
6	LED roșu	GPIO (Digital Output)	D13
7	LED verde	GPIO (Digital Output)	D12




TOSHIBA
ALKALINE
HIGH POWER
UNO

Software Design

Mediul de dezvoltare utilizat pentru implementarea firmware-ului a fost Arduino IDE, cu suport pentru platforma AVR-GCC, țintind microcontrolerul ATmega328P de pe placa Arduino UNO R3. În cadrul proiectului, s-a optat pentru o implementare low-level a protocoalelor, pentru o înțelegere mai profundă a funcționării hardware-ului.

Platformă	**Detalii**
IDE	Arduino IDE
Microcontroler	ATmega328P (pe placa Arduino UNO R3)
Compilator	AVR-GCC (implicit prin Arduino Toolchain)
Programare	În limbaj C/C++ cu acces direct la registre
Biblioteci	Fără librării externe (implementare manuală a I2C, PWM, GPIO, Timere)

Elemente software implementate până în acest moment

• Control tastatură matricială 4×4 (scanare + debouncing software)

• Verificare PIN + contor de greșeli

• Blocare automată a seifului după 10 secunde de la deschidere (millis() – non-blocking)

• Servomotor acționat manual pentru deschiderea/zăvorârea ușii

• LED-uri pentru semnalizare vizuală (GPIO)

• Buzzer cu secvențe sonore diferite pentru stări (corect, greșit, autodistrugere)

• Afișare mesaje pe LCD I2C controlat prin TWI

Corelare cu laboratoare

Laborator	**Concept**	**Aplicație în proiect**
Lab 0	GPIO	Tastatură matricială, LED-uri de stare
Lab 3	PWM & Timere	Servomotor (PWM manual), buzzer (tonuri)
Lab 6	I2C (fără librării)	Comunicare low-level cu LCD 1602 prin TWI

Implementarea logicii

Programul începe prin apelarea funcțiilor de inițializare:

- `initializePins()` configurează LED-urile (roșu și verde), buzzer-ul, pinul de control pentru servomotor și pinii pentru tastatura matricială (ca output pentru rânduri și input cu pull-up pentru coloane).

- `lcd_init()` și `twi_init()` pregătesc interfața LCD 1602 și inițializează comunicația I2C la nivel de registre TWI.

- `setServoPosition(1500)` setează servomotorul în poziția de blocare (90°), la pornirea sistemului.

- Variabilele globale precum `isUnlocked`, `enteredCode`, `unlockTime` sunt resetate la valori inițiale.

După inițializare, programul intră în bucla principală:

1. Se apelează `scanKeypad()` pentru a detecta apăsări pe tastatura 4×4.

2. Dacă utilizatorul apasă `*`, codul introdus este resetat și se afișează mesajul “Enter PIN”.

3. Dacă se apasă o cifră/caracter, acesta este adăugat la `enteredCode`, care este afișat pe LCD.

4. Când lungimea `enteredCode` este egală cu `correctCode`, se compară valorile:

A) Dacă codul este corect:

```
   • Se apelează `unlockSystem()` – servomotorul deschide zăvorul, se aprinde LED-ul verde și buzzerul emite un ton pozitiv.
```

```
   • Se salvează `unlockTime` și se setează `isUnlocked = true`.
```

B) Dacă codul este greșit:

```
   • Se crește contorul `failedAttempts`.
```

```
   • Se afișează mesaj de eroare, se apelează `playToneSequence()` cu tonuri triste.
```

```
   • Dacă `failedAttempts >= 3`, se apelează `handleSelfDestructSequence()` – o secvență cu mesaj animat, countdown, sunet de "alarmă" și flashuri LED.
```

În timpul în care sistemul este deblocat:

- Se afișează un countdown pe LCD (“Auto-lock in: X”).

- Dacă au trecut 10 secunde (`millis() - unlockTime > autoLockDelay`), se apelează `lockSystem()` și sistemul revine în starea de așteptare.

Funcții auxiliare

- `scanKeypad()`

```
Scanează rândurile tastaturii 4x4, setează LOW câte un rând pe rând și citește coloanele. Debouncing software este realizat cu millis().
```

- `setServoPosition(uint16_t position)`

```
Trimite un număr de impulsuri (până la 100) pentru a comanda un servomotor standard SG90, fără folosirea bibliotecii `Servo.h`.
```

- `playTone(uint16_t freq, uint16_t duration)`

```
Generează semnale sonore de frecvență variabilă prin toggling manual al pinului pentru buzzer (fără PWM hardware).
```

- `lcd_init()`

```
Inițializează LCD-ul în mod 4-bit, 2 rânduri, folosind comenzi standard trimise prin I2C cu funcții low-level (`twi_write()`).
```

- `handleSelfDestructSequence()`

```
Afișează un mesaj animat derulant pe LCD, urmat de un countdown și un efect vizual/auditiv (flash LED + sunet de alarmă), apoi revine la `lockSystem()`.
```

Validarea funcțională a fost realizată fizic, cu testarea tuturor scenariilor:

- cod corect → deblocare + ton pozitiv + LED verde

- cod greșit → mesaj eroare + ton negativ + LED roșu

- 3 greșeli → secvență specială de autodistrugere

- blocare automată → verificat după 10 secunde inactivitate

Toate modulele funcționează sincron, fără blocări sau conflicte, datorită utilizării `millis()` pentru temporizări non-blocante și a organizării logice clare între stări (locked/unlocked). `

Componentă	**Metodă de calibrare**	**Rezultat / Valoare finală**
Servomotor	Testare empirică a impulsurilor PWM manuale	600 µs = deblocat (0°), 1500 µs = blocat (90°)
Tastatură 4×4	Debounce software ajustat folosind `millis()`	Timp de debounce: 50 ms
LCD 1602 I2C	Ajustare timpi între comenzi și date, conform datasheet	Funcționare stabilă, fără caractere corupte
Buzzer piezoelectric	Reglare durată și frecvență pentru sunete clare și diferențiate	Ton pozitiv: 262–330 Hz, eroare: 262–175 Hz

Rezultate obținute

• Sistemul funcționează stabil și complet automatizat.

• Codul PIN este verificat corect, cu feedback audio-vizual.

• Servomotorul acționează zăvorul precis, fără librării externe.

• LCD-ul comunică prin I2C low-level (fără Wire.h).

• Blocarea automată funcționează corect cu temporizare non-blocantă.

• Secvența de autodistrugere este declanșată după 3 greșeli.

• Toate componentele au fost testate fizic, cu rezultate conforme.

Concluzii

Proiectul a reușit să integreze cu succes conceptele studiate în laborator, precum GPIO, PWM și I2C, într-un sistem embedded funcțional și bine organizat. Abordarea low-level, fără librării externe, a permis un control complet asupra hardware-ului și o înțelegere aprofundată a comunicației între componente.

Sunt fericit că am reușit să implementez de la zero un seif inteligent folosind Arduino, parcurgând toate etapele esențiale: planificarea logicii, căutarea și achiziția componentelor, documentarea din surse online și vizionarea de tutoriale video pe YouTube.

Experiența a fost una practică, educativă și motivantă, și m-a ajutat să înțeleg mai bine cum se dezvoltă un proiect embedded cap-coadă.

Jurnal

- 26 Aprilie – Alegerea temei proiectului

- 1, 2 Mai – Brainstorming pentru alegerea componentelor

- 3 Mai – Comandarea componentelor hardware

- 6 Mai – Construirea circuitului electronic în Tinkercad (simulare)

- 8 Mai – Recepționarea componentelor fizice

- 10 Mai – Completarea Milestone 1

- 12 Mai – Asamblarea circuitului real

- 12 Mai – Implementarea unui cod funcțional high-level (cu librării)

- 17 Mai – Prezentarea Milestone 2

- 24 Mai – Convertirea codului high-level în cod low-level (fără librării)

- 25 Mai – Finalizarea și prezentarea Milestone 3

- 28 Mai – Participarea la PM Fair

Bibliografie/Resurse

Pagina de GitHub

Demo-Video in development

Demo-Video fully built

Export to PDF