

SISTEM DIGITAL PENTRU SIGURANȚĂ ȘI CONFORT CASNIC

Candidat: Dalia GOGOAȘ

Coordonator științific: dr.ing. Alexandru TOPÎRCEANU

Sesiunea: Iunie 2024

REZUMAT

Această lucrare reprezintă un prototip al unui produs inovator, conceput pentru a sporii siguranța și confortul în propria casă prin digitalizare. Proiectul este împărțit în două capitole principale: circuitul care efectuează descurierea fizică a ușii și aplicația mobilă care monitorizează și controlează parțial acest circuit.

Mecanismul de deschidere a ușii se bazează pe un senzor de amprentă plasat în exteriorul casei, un buton plasat pe partea interioară și aplicația mobilă. Comunicarea dintre circuit și aplicație se realizează prin conceptul de IoT (Internet of Things), utilizând un modul Bluetooth integrat în arhitectura circuitului. Acest tip de comunicare a fost ales deoarece permite o interacțiune eficientă pe distanțe scurte, ideală pentru utilizarea în interiorul casei sau în proximitatea acesteia.

Proiectul include și o funcționalitate care permite vizualizarea istoricului deschiderii ușii de la orice distanță, folosind o bază de date Firebase accesibilă prin internet. Astfel, utilizatorii pot monitoriza în timp real starea ușii, indiferent de locație, atâta timp cât au acces la internet. Această caracteristică este esențială pentru securitate, oferind informații precise despre momentul deschiderii ușii, utile în cazul unei spargerii.

Circuitul este controlat de un Arduino Nano, iar aplicația mobilă este dezvoltată în Android Studio folosind Java și XML. Implementată în Java, aplicația se bazează pe principii de Programare Orientată pe Obiecte.

În urma realizării acestui proiect am reușit să îndeplinesc obiectivele propuse lăsând loc pentru posibile îmbunătățiri la nivelul securității în faza de autentificare. Concluziile pe care le-am tras în urma procesului de implementare a proiectului atât hardware cât și software sunt următoarele: a fost mult mai greu decât ma așteptam și am reușit să îmi depășesc anumite limite, un adevărat inginer automatist trebuie să cunoască atât partea hardware cât și software pentru a excela în domeniu și în cele din urmă, când vine vorba despre securitate și nu numai este nevoie de multă imaginație pentru a acoperii toate cazurile posibile de interacțiune cu prototipul.

CUPRINS

1	Introducere	6
1.1	Domeniul proiectului	6
1.2	Specificații generale ale dezvoltării proiectului	6
1.3	Obiective	7
2	Prototipuri asemănătoare	8
2.1	August Wi-Fi Smart Lock	8
2.1.1	Specificații tehnice	8
2.1.2	Puncte Forte și Limitări	8
2.2	Yale Assure Lock SL	9
2.2.1	Specificații tehnice	9
2.2.2	Puncte Forte și Limitări	9
2.3	Schlage Encode Smart WiFi Deadbolt	10
2.3.1	Specificații tehnice	10
2.3.2	Puncte Forte și Limitări	10
2.4	Kwikset Kevo Smart Lock	11
2.4.1	Specificații tehnice	11
2.4.2	Puncte Forte și Limitări	11
2.5	Încuietori inteligente DIY pe baza Arduino	12
2.5.1	Specificații tehnice	12
2.5.2	Puncte Forte și Limitări	12
2.6	Compararea încuietorilor inteligente similare	14
3	Tehnologii folosite	16
3.1	Internet of Things (IoT)	16
3.2	Ingineria Mecanică	17
3.2.1	Arduino IDE	17
3.3	Dezvoltarea Software	18
3.3.1	Android Studio	18
3.3.2	Java	19
3.3.3	Programarea Orientată pe Obiecte	20
4	Specificații Tehnice	21
5	Implementare	22
5.1	Componentele Hardware	22
5.1.1	Microcontroler	22
5.1.2	Servo Motor	23
5.1.3	Componente Adiționale ale Circuitului	28
5.2	Componenta Software	34

5.2.1	Principii OOP folosite	34
6	Fluxul de funcționare al aplicației	39
6.1	Autentificare	39
6.2	Controlul și monitorizarea Circuitului	42
7	Testare	55
7.1	Scenarii de testare	55
7.1.1	Activarea și dezactivarea Rețelei Bluetooth pe Telefonul Mobil	55
7.1.2	Conectarea la dispozitiv	56
7.1.3	Deblocarea Ușii prin Amprentă	57
7.1.4	Deblocarea ușii prin buton	58
7.1.5	Deblocarea ușii prin aplicație	58
8	Concluzii	60
8.1	Concluzii pe baza realizării proiectului	60
8.2	Posibile viitoare îmbunătățiri ale proiectului	60
9	Bibliografie	61

LISTĂ DE FIGURI

5.1	Arduino Nano v.3 (sursa: [8])	23
5.2	Arduino Nano v.3 (sursa: [16])	25
5.3	Proiectarea roților (sursa: [7])	26
5.4	Proiectarea mânerului (sursa: [7])	26
5.5	Sistemul de rotire vizualizat din față	27
5.6	Sistemul de rotire vizualizat din spate	28
5.7	Comutator magneti (sursa: [17])	29
5.8	Buton (sursa: [18])	29
5.9	Senzor de amprentă (sursa: [19])	30
5.10	Buzzer (sursa: [21])	31
5.11	Schema bloc a circuitului	33
6.1	LogIn	40
6.2	Register	40
6.3	Validare date	41
6.4	Pagina Principală	47
6.5	Activare Bluetooth	48
6.6	Activarea butonului de dezvăluire a potențialelor dispozitive	48
6.7	Lista dispozitivelor	49
6.8	Încercarea de vizualizare a potențialelor dispozitive	49
6.9	Dezactivare Bluetooth	50
6.10	Istoric	50
6.11	Eroare la conectare	51
6.12	Niciun dispozitiv conectat	51
6.13	Mesaj trimis cu succes	52
6.14	Notificare în cazul de amprentă străină	52
6.15	Notificare pentru deschiderea ușii	53
6.16	Notificare pentru închiderea ușii	53
6.17	Notificare în afara aplicației	54

LISTĂ DE TABELE

2.1	Compararea caracteristicilor principalelor tipuri de încuietori inteligente	14
-----	---	----

LISTĂ DE FRAGMENTE

5.1	Funcțiile de rotație	24
5.2	Validarea amprentei	31
5.3	Întregul ciclu al circuitului	32
5.4	Interfața DeviceServiceCallback	35
5.5	Gestionarea datelor privind utilizatorul actual	35
5.6	Abstractizare	36
5.7	Implementarea Interfeței DeviceServiceCallback	36
5.8	Utilizarea Interfeței DeviceServiceCallback	37
5.9	Moștenirea	37
5.10	Singleton	38
6.1	Deblocarea ușii de către circuit	43
6.2	Conectarea la încuietoare	44
6.3	Deblocarea ușii din intermediul aplicației	45
6.4	Obiectul de tip BroadcastReceiver	45
6.5	Funcția de registerReciver()	45
6.6	Funcția de creare a notificărilor	46
6.7	Funcția de reciveMessage()	47

1. INTRODUCERE

1.1 DOMENIUL PROIECTULUI

Domeniul acestui proiect se încadrează în sfera tehnologiei IoT (Internet of Things) și a securității inteligente pentru locuințe. Proiectul combină aspecte de inginerie mecanică, electronică și dezvoltare software pentru a crea un sistem de încuietoare inteligentă care sporește confortul și siguranța utilizatorilor.

Securitate Inteligentă pentru Locuințe reprezintă un domeniu în creștere rapidă, care îmbină tehnologii avansate pentru a oferi soluții de securitate eficiente și convenabile. Aceasta implică utilizarea dispozitivelor conectate la internet, care permit monitorizarea, controlul și gestionarea securității casei în timp real, de la distanță. Acestea sunt câteva aspecte esențiale ale securității inteligente pentru locuințe:

- Componenta tehnologică a acestor sisteme include dispozitive conectate (IoT) precum camere de supraveghere, senzori de mișcare, alarme și încuietori inteligente, care comunică între ele prin internet. Automatizarea și controlul de la distanță permit utilizatorilor să monitorizeze și să gestioneze securitatea locuinței lor prin aplicații mobile sau alte platforme online, indiferent de locație. În plus, inteligența artificială (AI) și tehnologiile de machine learning sunt folosite pentru a analiza datele colectate de la dispozitivele de securitate, identificând modele și comportamente suspecte și îmbunătățind astfel răspunsul la potențiale amenințări.
- Dispozitivele cheie în cadrul sistemelor de securitate inteligentă includ încuietorile inteligente, care permit accesul în locuință prin metode avansate, cum ar fi cititoarele de amprente, codurile PIN, aplicațiile mobile și recunoașterea facială. Camerele de supraveghere integrate în rețelele IoT oferă monitorizare video în timp real, înregistrări și alerte instantanee. De asemenea, senzorii de mișcare și contact detectează mișcarea sau deschiderea ferestrelor și ușilor, trimițând alerte către utilizatori sau declanșând alarmele.
- Securitatea datelor devine o preocupare majoră, deoarece dispozitivele conectate la internet sunt predispuse la hacking și acces neautorizat, fiind esențială protecția împotriva acestor amenințări. Compatibilitatea dispozitivelor reprezintă o altă provocare, deoarece integrarea diferitelor dispozitive de securitate într-un sistem poate fi dificilă, necesitând standarde și protocoale comune. În cele din urmă, costurile inițiale pentru implementarea unui sistem de securitate inteligent pot fi semnificative dar vor fi amortizate pe termen lung prin beneficiile oferite.

1.2 SPECIFICAȚII GENERALE ALE DEZVOLTĂRII PROIECTULUI

Proiectul a fost dezvoltat începând cu componenta hardware care efectuează deblocarea fizică a ușii. Întreaga sa activitate este controlată de microcontrolerul Arduni Nano

care a fost programat folosind Arduino IDE specific pentru toate plăcuțele Arduino. Aplicația mobilă care constituie componenta Software a proiectului a fost dezvoltată sub forma unei aplicații android folosind AndroidStudio ca și mediu de dezvoltare. Acesta este structurat special pentru dezvoltarea acestui tip de aplicații punând la dispoziție toate condițiile necesare pentru un proces optim și rapid de implementare. Limbajul de programare folosit pentru dezvoltarea funcționalităților a fost Java iar pentru interfața cu utilizatorul, am folosit XML. În cursul implementării au fost urmate principii de Programare Orientată pe Obiecte pentru a realiza un cod ușor de înțeles, reutilizat și ușor de extins pentru viitoare noi îmbunătățiri. Legătura dintre aceste două mari componente a fost realizată prin conceptul de IOT folosind comunicarea prin Bluetooth.

1.3 OBIECTIVE

Înainte de a începe dezvoltarea fizică a proiectului am stabilit câteva obiective pe care acesta să le îndeplinească:

- a. Dezvoltarea și implementarea funcționalităților de deblocare a ușii prin:
 - senzor de amprentă pentru autentificare biometrică
 - buton pentru deblocarea manuală a ușii din interiorul casei.
 - comenzi de deblocare a ușii prin intermediul unei aplicații mobile
- b. Monitorizarea stării ușii prin instalarea și configurarea magneților pe ușă pentru a detecta starea deschisă/închisă și activarea automată a mecanismului de închidere a ușii bazată pe starea detectată de magneți.
- c. Prevenirea accesului neautorizat la controlarea și monitorizarea încuietorii prin implementarea unui sistem de autentificare securizat.
- d. Crearea unei funcționalități în aplicație care permite utilizatorilor să vizualizeze istoricul deschiderilor și să stocheze și gestioneze securizat datele referitoare la deschiderea ușii.
- e. Controlul Bluetooth-ului de pe dispozitivul personal prin aplicația mobilă.
- f. Implementarea unei metode de conectare simplă și intuitivă între aplicația mobilă și încuietore.

2. PROTOTIPURI ASEMĂNĂTOARE

2.1 AUGUST WI-FI SMART LOCK

August Wi-Fi Smart Lock este o soluție de blocare inteligentă de ultimă generație, care permite utilizatorilor să-și controleze și să-și monitorizeze ușa de la distanță. Spre deosebire de versiunile anterioare, acest model integrează Wi-Fi direct în dispozitiv, eliminând necesitatea unui bridge separat pentru conectivitate la internet.

2.1.1 SPECIFICAȚII TEHNICE

Wi-Fi Integrat, permite controlul de la distanță fără a fi necesar un hub suplimentar.

Control prin Aplicație oferă posibilitatea utilizatorului de a bloca sau debloca ușa de oriunde prin intermediul aplicației August.

Această încuietore este compatibilă cu Amazon Alexa, Google Assistant, Apple HomeKit și multe alte platforme de automatizare a casei.

Încuietorea efectuează blocarea și deblocare automată a ușii folosind localizarea telefonului. Ușa se blochează automat când pleci și se deblochează când ajungi acasă.

Partea de securitate conține și verificare biometrică prin amprentă sau recunoaștere facială.

Pentru o îmbunătățire a interacțiunii această încuietore prezintă și un senzor care monitorizează dacă ușa este deschisă sau închisă și trimite alerte corespunzătoare

Utilizatorii pot seta alarme pentru a fi notificați când anumite persoane intră sau ies din casă sau dacă ușa rămâne deschisă.

Aplicația oferă un feed de activitate, unde utilizatorii pot vedea istoricul deschiderilor și închiderilor ușii.

Instalarea este simplă și durează aproximativ 10 minute. Nu necesită înlocuirea întregului mecanism al încuietorii, ci doar a părții interioare. Kit-ul include toate componentele necesare, iar aplicația August oferă instrucțiuni detaliate pas cu pas pentru instalare și configurare

2.1.2 PUNCTE FORTE ȘI LIMITĂRI

Avantaje principale ale acestei încuietori sunt:

- Instalare rapidă și ușoară.
- Nu necesită schimbarea cheilor existente.
- Conectivitate directă prin Wi-Fi și compatibilitate cu multiple platforme smart home.
- Funcții avansate de securitate și verificare biometrică.

Dezavantaje principale ale acestei încuietori sunt:

- Cost relativ ridicat comparativ cu alte încuietori inteligente.
- Funcționalitățile automate de blocare/deblocare pot fi uneori imprecise, depinzând de localizarea telefonului.
- Conectivitate directă prin Wi-Fi și compatibilitate cu multiple platforme smart home.
- Necesitatea de a achiziționa separat tastatura pentru acces suplimentar, ceea ce crește costurile.

Resurse: [1]: Informații detaliate despre specificațiile tehnice și avantajele Yale Assure Lock SL.

2.2 YALE ASSURE LOCK SL

Yale Assure Lock SL este o încuietoare inteligentă avansată care oferă utilizatorilor posibilitatea de a înlocui complet mecanismul de închidere al ușii, asigurând accesul fără cheie prin intermediul unui ecran tactil și diverse opțiuni de integrare smart home.

2.2.1 SPECIFICAȚII TEHNICE

Încuietoarea prezintă un ecran tactil care permite eliminarea completă a cheilor fizice, oferind un acces modern și securizat.

Integrare cu Smart Home necesită module suplimentare pentru compatibilitate cu platforme precum Z-Wave, Zigbee, Apple HomeKit și August Connect.

Alimentare este bazată pe baterii. Folosește patru baterii AA cu o durată de viață de până la un an.

Acces se face prin coduri Oferind posibilitatea de a crea coduri de acces pentru oaspeți.

Instalare este relizată de către utilizator și necesită înlocuirea întregului mecanism al încuietorii, dar poate fi instalată fără unelte speciale, doar cu o șurubelniță.

2.2.2 PUNCTE FORTE ȘI LIMITĂRI

Avantaje principale ale acestei încuietori sunt:

- Design modern și elegant.
- Eliminarea cheilor fizice.
- Opțiuni de integrare smart home cu o compatibilitate largă prin module suplimentare.
- Instalare relativ simplă, deși necesită înlocuirea mecanismului procesul este bine detaliat și accesibil pentru utilizatorii obișnuiți.

Dezavantaje principale ale acestei încuietori sunt:

- Costuri suplimentare pentru accesorii cum ar fi modulele pentru integrare smart home care sunt vândute separat, crescând costul total.

- Durata de viață a bateriilor deși este decentă, utilizatorii trebuie să fie atenți la înlocuirea bateriilor pentru a evita blocarea accesului.

Resurse: [2]: Informații detaliate despre specificațiile tehnice și avantajele Yale Assure Lock SL.

2.3 SCHLAGE ENCODE SMART WIFI DEADBOLT

Schlage Encode Smart WiFi Deadbolt este o soluție avansată de blocare inteligentă care permite utilizatorilor să controleze și să monitorizeze ușa de la distanță, oferind o combinație de securitate de top și funcționalități moderne.

2.3.1 SPECIFICAȚII TEHNICE

Opțiunea de Wi-Fi integrat, permite conectarea directă la rețeaua Wi-Fi a casei, fără a fi necesar un hub suplimentar.

Controlul se realizează prin intermediul aplicației oferind posibilitatea utilizatorului de a bloca sau debloca ușa de oriunde folosind aplicațiile Schlage Home sau Key by Amazon.

Această încuietore este compatibilă cu Amazon Alexa și Google Assistant pentru control vocal.

prezența unui ecran tactil, oferă acces fără cheie și permite crearea și gestionarea a până la 100 de coduri de acces unice.

Alimentare se realizează pe bază de baterii utilizând 4 baterii AA, care sunt ușor de înlocuit.

Securitate este certificată cu gradul 1 comercial ANSI/BHMA, oferind cea mai înaltă protecție pentru o încuietore rezidențială.

Instalare se realizează de către utilizator într-un mod ușor folosind doar o șurubelniță, datorită tehnologiei *Snap'n Stay* care fixează broasca pe ușă.

2.3.2 PUNCTE FORTE ȘI LIMITĂRI

Avantaje principale ale acestei încuietori sunt:

- Design elegant și durabil, disponibil în diverse finisaje și stiluri pentru a se potrivi cu orice decor.
- Funcții avansate de securitate care includ criptare securizată și tehnologii de alarmă integrate pentru protecție împotriva străinilor.
- Control la distanță, permite monitorizarea și gestionarea de la distanță a accesului prin intermediul aplicațiilor dedicate.
- Accesul prin coduri oferă posibilitatea de a crea coduri temporare sau permanente pentru oaspeți, cu programare individuală.

Dezavantaje principale ale acestei încuietori sunt:

- Prețul ridicat, fiind una dintre cele mai scumpe încuietori rezidențiale.

- Instalare complexă, deși este concepută pentru instalare personală, necesită înlocuirea întregului mecanism al broaștei, ceea ce poate fi dificil pentru utilizatorii fără experiență.
- Dimensiune mare, designul poate fi considerat voluminos comparativ cu alte încuietori inteligente.

Resurse:

- [3]: Informații detaliate despre specificațiile tehnice și avantajele Schlage Encode Smart WiFi Deadbolt.
- [4]: Descriere oficială și caracteristici ale produsului Schlage Encode Smart WiFi Deadbolt.

2.4 KWIKSET KEVO SMART LOCK

Kwikset Kevo Smart Lock este o soluție inovatoare de blocare inteligentă care utilizează tehnologia Bluetooth pentru a permite utilizatorilor să deschidă ușa printr-o simplă atingere, oferind o combinație de confort și securitate modernă.

2.4.1 SPECIFICAȚII TEHNICE

Tehnologie Bluetooth, permite deschiderea ușii prin simpla atingere a unui smartphone compatibil.

Control prin Aplicație oferă posibilitatea utilizatorului de a gestiona accesul prin intermediul aplicației Kevo, permițând trimiterea de eKeys electronice pentru acces temporar, programat sau permanent.

Integrare cu Smart Home necesită Kevo Plus, un gateway Bluetooth care permite integrarea cu Amazon Alexa, Ring, Skybell și termostatele Honeywell.

Alimentare este relizată pe bază de baterii utilizează 4 baterii AA, care sunt ușor de înlocuit.

Backup Keyhole și Funcția SmartKey Re-Keying, oferă posibilitatea utilizării unei chei fizice de rezervă și reconfigurarea rapidă a cilindrului pentru a accepta o nouă cheie.

Instalare este realizată de către utilizator fiind ușor de realizat cu ajutorul ghidului interactiv disponibil în aplicația Kevo și instrucțiunilor detaliate incluse în pachet.

2.4.2 PUNCTE FORTE ȘI LIMITĂRI

Avantaje principale ale acestei încuietori sunt:

- Design modern și elegant disponibil în două opțiuni (rotund și pătrat).
- Tehnologia Touch-to-Open oferă un acces convenabil fără chei fizice.
- Controlul accesului prin eKeys permite trimiterea de chei electronice temporare, programate sau permanente.
- Integrare cu dispozitive smart home, inclusiv Amazon Alexa, Ring, Skybell și Honeywell.

Dezavantaje principale ale acestei încuietori sunt:

- Necesitatea unui gateway Kevo Plus pentru control de la distanță și integrare smart home, ceea ce crește costurile totale.
- Funcționalitățile avansate necesită un abonament Kevo Plus.
- Lipsa unei tastaturi fizice pentru acces prin coduri poate fi un dezavantaj pentru utilizatorii care preferă această metodă de acces.
- Oaspeții trebuie să descarce aplicația Kevo și să permită accesul la locația dispozitivului pentru a utiliza eKeys.

Resurse:

- [5]: Informații detaliate despre specificațiile tehnice și avantajele Kwikset Kevo Smart Lock.
- [6]: Descriere oficială și caracteristici ale produsului Kwikset Kevo Smart Lock.

2.5 ÎNCUIETORI INTELIGENTE DIY PE BAZA ARDUINO

Încuietorile inteligente DIY pe baza platformei Arduino oferă o soluție flexibilă și accesibilă pentru securizarea ușilor, permițând utilizatorilor să creeze sisteme personalizate de blocare și deblocare prin intermediul programării și componentelor electronice.

2.5.1 SPECIFICAȚII TEHNICE

Microcontrolerul Arduino este inima sistemului, controlând toate componentele electronice și gestionând logica de blocare/deblocare.

Controlul prin aplicație sau interfață web permite utilizatorilor să blocheze sau să deblocheze ușa de la distanță.

Tehnologia de recunoaștere a amprentelor este adesea integrată, folosind senzori de amprentă pentru a oferi acces biometric securizat.

Modulul Bluetooth sau Wi-Fi este utilizat pentru comunicarea fără fir între microcontroler și dispozitivele mobile.

Alimentare poate fi realizată pe bază de baterii sau prin sursă de curent, în funcție de designul specific al proiectului.

Instalare de către utilizator este relativ simplă pentru cei cu cunoștințe de bază în electronică și programare, necesitând asamblarea componentelor și scrierea codului necesar pentru funcționarea încuietorii.

2.5.2 PUNCTE FORTE ȘI LIMITĂRI

Avantaje principale ale acestor încuietori sunt:

- Flexibilitate ridicată în design și funcționalități, permițând personalizarea în funcție de nevoile specifice ale utilizatorului.

- Costuri reduse comparativ cu soluțiile comerciale, datorită utilizării componentelor accesibile și a platformei open-source Arduino.
- Posibilitatea de a învăța și de a dezvolta abilități în electronică și programare prin construirea și configurarea sistemului.

Dezavantaje principale ale acestor încuietori sunt:

- Necesitatea unor cunoștințe tehnice în electronică și programare, ceea ce poate fi o barieră pentru utilizatorii fără experiență.
- Potențialele probleme de fiabilitate și securitate, în funcție de calitatea implementării și a componentelor utilizate.
- Lipsa suportului profesional, utilizatorii fiind responsabili de întreținerea și repararea sistemului în caz de probleme.

Resurse:

- [7]: Informații detaliate despre realizarea unui sistem de încuietoare inteligentă pe bază de Arduino.
- [8]: Detalii despre utilizarea plăcii Arduino Nano pentru proiecte DIY de încuietori inteligente.
- [9]: O colecție de proiecte DIY bazate pe Arduino, inclusiv încuietori inteligente.

2.6 COMPARAREA ÎNCUIETORILOR INTELIGENTE SIMILARE

Tabelul 2.1: Compararea caracteristicilor principalelor tipuri de încuietori inteligente

Tipul	Caracteristici			
	August Smart	Yale Assure SL	Schlage Encode Smart WiFi Deadbolt	Kwikset Kevo Smart
Conectare	Wi-Fi	Necesită module suplimentare	Wi-Fi	Bluetooth
Control	aplicația August	aplicația Yale	aplicațiile Schlage Home și Key by Amazon	aplicația Kevo
Compatibilitate Smart Home	Alexa, Google Assistant, Apple HomeKit	Z-Wave, Zigbee, HomeKit, August Connect	Alexa, Google Assistant	Alexa, Ring, Skybell, Honeywell
Acces fără Cheie	Da, blocare/deblocare automată	Da, ecran tactil	Da, ecran tactil	Da, Touch-to-Open
Securitate Biometrică	amprentă, Face ID	Nu	Nu	Nu
Senzor Ușă	Da	Nu	Nu	Nu
Coduri	Nu	Da coduri pentru oaspeți	Da	Da, eKeys electronice
Alimentare	4 baterii AA	4 baterii AA	4 baterii AA	4 baterii AA
Instalare	Instalare fără înlocuirea mecanismului	Instalare DIY, necesită înlocuirea mecanismului	Instalare DIY, ușoară	Instalare DIY, ghid interactiv
Preț	Relativ ridicat	Ridicat	Foarte ridicat	Moderat

Analiza comparativă prezentată în tabelul 2.1 oferă o perspectivă detaliată asupra principalelor tipuri de încuietori inteligente disponibile pe piață. Iată câteva concluzii importante trase din această comparație integrând și încuietori inteligente DIY pe baza de Arduino:

Conectare: August Wi-Fi Smart Lock și Schlage Encode Smart WiFi Deadbolt se remarcă prin Wi-Fi integrat, eliminând necesitatea unui hub suplimentar. Yale Assure Lock SL necesită module suplimentare, iar Kwikset Kevo Smart Lock și încuietorile Arduino DIY folosesc Bluetooth, cu opțiuni de Wi-Fi pentru Arduino.

Control: Toate modelele permit controlul prin aplicație, oferind flexibilitate în gestionarea accesului. Încuietorile Arduino DIY oferă posibilitatea de a crea aplicații

personalizate.

Compatibilitate Smart Home: August și Schlage sunt compatibile cu Alexa și Google Assistant, oferind o integrare extinsă. Yale necesită module suplimentare, iar Kwikset și Arduino DIY necesită uneori gateway-uri suplimentare.

Acces fără Cheie și Securitate Biometrică: Toate modelele oferă acces fără cheie, însă doar August integrează securitate biometrică. Încuietorile Arduino DIY pot fi personalizate pentru a include recunoașterea amprentelor.

Coduri: Yale și Schlage permit crearea de coduri pentru oaspeți, iar August oferă senzorul DoorSense™ pentru monitorizarea stării ușii. Încuietorile Arduino DIY pot fi configurate pentru a utiliza senzori de ușă și coduri de acces configurabile.

Instalare și Costuri: August și Kwikset sunt ușor de instalat, în timp ce Yale și Schlage necesită înlocuirea mecanismului. Încuietorile Arduino DIY necesită cunoștințe de bază în electronică. Schlage este cel mai scump, iar soluțiile Arduino sunt cele mai accesibile, dar necesită achiziționarea componentelor și cunoștințe tehnice.

În urma analizării tabelului 2.1 am ajuns la următoarele concluzii:

- August Wi-Fi Smart Lock este ideal pentru securitate biometrică și integrare smart home.
- Yale și Schlage oferă funcționalități avansate, dar la un cost mai mare. Kwikset este accesibil și ușor de utilizat.
- Arduino DIY este potrivit pentru entuziaștii de electronică și programare, oferind flexibilitate și costuri reduse.

3. TEHNOLOGII FOLOSITE

3.1 INTERNET OF THINGS (IOT)

Conexiunea dintre componenta Hardware și cea Software a fost realizată prin punând în aplicare conceptul de IOT. Istoria Internet of Things (IoT) începe în anii '80 cu dispozitive conectate experimental, cum ar fi un automat Coca-Cola, de băuturi răcoritoare de la Universitatea Carnegie Mellon. Termenul "Internet of Things" a fost introdus în 1999 de Kevin Ashton de la MIT. În anii 2000, tehnologiile de comunicație wireless și microelectronică au avansat, iar IoT a început să fie adoptat pe scară largă în diverse industrii în anii 2010. Odată cu apariția rețelilor 5G în anii 2020, IoT a devenit esențial pentru dezvoltarea orașelor inteligente și a vehiculelor autonome, oferind noi oportunități pentru inovație și eficiență.

Există mai multe căi de comunicare care permit dispozitivelor să se conecteze și să schimbe date. Fiecare metodă are propriile avantaje și dezavantaje, fiind potrivite pentru diferite aplicații:

- Wi-Fi este o tehnologie de rețea, larg răspândită, care permite conectarea dispozitivelor la internet și între ele într-o rețea locală, oferind viteză mare de transfer a datelor, dar având un consum ridicat de energie și o rază de acțiune limitată.
- Bluetooth este o tehnologie de comunicare utilizată pentru conectarea pe distanțe scurte între dispozitive, având un consum redus de energie (mai ales Bluetooth Low Energy - BLE). Datorită acestor aspecte acest tip de conexiune este ideal pentru dispozitive portabile și de proximitate având ca și dezavantaje o rază de acțiune scurtă și viteză de transfer a datelor mai mică comparativ cu Wi-Fi.
- Rețelele mobile (2G, 3G, 4G, 5G) permit comunicații pe distanțe lungi și sunt utilizate frecvent pentru dispozitive care necesită conectivitate constantă și mobilitate, oferind o acoperire largă și o viteză mare de transfer a datelor (în special 4G și 5G). Ca și dezavantaj al acestui tip de comunicare este prezența unui consum ridicat de energie.
- Zigbee este un standard de comunicație wireless destinat aplicațiilor cu consum redus de energie și rată de date redusă, utilizat frecvent în automatizările de casă, având avantajul unui consum foarte redus de energie. Un alt avantaj al acestui tip de comunicare este utilizarea rețelilor de tip mesh care extind raza de acțiune, dar cu dezavantajul unei viteze reduse de transfer a datelor și al unei complexități mai mari în configurarea rețelei.
- LoRaWAN (Long Range Wide Area Network) este un protocol de rețea pentru comunicații pe distanțe lungi, cu consum redus de energie, utilizat în special în aplicațiile industriale și de monitorizare a mediului. Are ca și avantaj principal o distanță mare de comunicație fiind ideal pentru aplicații de senzori distribuiți, dar cu dezavantajul unei viteze de transfer a datelor foarte redusă și al unei complexități în instalare și gestionare.

- Sigfox este o rețea LPWAN (Low Power Wide Area Network) destinată comunicațiilor pe distanțe lungi oferind avantajul unui consum foarte redus de energie și acoperire globală prin rețele dedicate. În schimb dezavantajul acestor rețele este viteza de transfer a datelor foarte redusă și capacitate limitată de trimitere a mesajelor.
- NB-IoT este o tehnologie LPWAN standardizată, dezvoltată pentru a oferi conectivitate prin rețelele celulare existente, având avantajul unui consum redus de energie, acoperire largă și penetrare bună în interiorul clădirilor, dar cu dezavantajul unei viteze de transfer a datelor redusă și al unei latențe mai mari comparativ cu rețelele celulare standard.

Mai multe informații legate de IOT se regăsesc în [10].

3.2 INGINERIA MECANICĂ

Ingineria mecanică este esențială în proiectele IoT, deoarece implică proiectarea și asamblarea componentelor fizice care alcătuiesc dispozitivul. În cazul unui proiect de încuietoare inteligentă, următoarele aspecte sunt cruciale:

- Componentele fizice ale încuietorii, cum ar fi carcasa și suporturile, sunt proiectate folosind software CAD (Computer-Aided Design) și apoi imprimate 3D, permițând personalizarea și adaptarea rapidă a designului pentru diferite tipuri de uși și mânere.
- Servomotoarele sunt motoare mici și precise controlate electronic pentru activarea mecanismului de blocare/deschidere a ușii, sunt integrate prin montarea lor în structura imprimată 3D, conectarea la mecanismul de închidere/deschidere și calibrarea lor pentru a asigura o mișcare precisă și consistentă, prevenind blocajele sau deschiderile accidentale.
- Proiectul poate include senzori de amprentă, comutatoare și alte dispozitive care interacționează cu servomotoarele, acestea fiind poziționate și conectate corespunzător pentru a asigura funcționarea corectă a sistemului. Un exemplu din cadrul proiectului ar fi un comutator magnetic (reed switch) folosit pentru a detecta dacă ușa este închisă și a activa servomotorul pentru blocare.

3.2.1 ARDUINO IDE

Programarea microcontrolerului care controlează to acest ansamblu de componente electronice a fost realizată în Arduino IDE. Este platforma software oficială pentru programarea și dezvoltarea proiectelor bazate pe microcontrolerele Arduino. IDE-ul Arduino este conceput pentru a fi ușor de utilizat, adresându-se atât începătorilor, cât și dezvoltatorilor avansați. Acesta prezintă câteva caracteristici care îl fac să fie atât de ușor de folosit:

- a. Editor de Cod Ușor de Utilizat: Arduino IDE oferă un editor de cod simplu și intuitiv, cu evidențiere a sintaxei, completare automată și diverse funcții de editare care facilitează scrierea și modificarea codului.

- b. **Compilare și Upload Simplificat:** Integrează un compilator care convertește codul scris într-un format executabil de către microcontroler și permite încărcarea facilă a acestuia pe plăcile Arduino prin intermediul unui port USB.
- c. **Biblioteci Extinse:** Arduino IDE include o vastă colecție de biblioteci care permit extinderea funcționalităților standard ale microcontrolerelor, facilitând implementarea rapidă a senzorilor, motoarelor, afișajelor și altor componente.
- d. **Compatibilitate Cross-Platform:** IDE-ul este disponibil pentru diverse sisteme de operare, inclusiv Windows, macOS și Linux, oferind o experiență de utilizare consistentă pe toate platformele.
- e. **Integrare cu Platformele de Dezvoltare Online:** Permite integrarea cu Arduino Web Editor, o alternativă online care oferă sincronizarea proiectelor și acces la ele de pe orice dispozitiv conectat la internet.

Arduino IDE sprijină prototiparea rapidă și dezvoltarea proiectelor de electronică, fiind o unealtă esențială pentru entuziaștii de hardware și profesioniști. Cu o comunitate vastă și resurse online abundente, Arduino IDE continuă să fie un standard de facto în educația STEM și în dezvoltarea de proiecte embedded. Mai multe informații legate de Arduino IDE se regăsesc în [11].

3.3 DEZVOLTAREA SOFTWARE

Dezvoltarea software este crucială pentru a asigura controlul și monitorizarea eficientă a dispozitivului. Aplicația a fost dezvoltată sub forma unei aplicații Android având posibilitatea de a o instala pe un dispozitiv portabil, telefonul mobil.

3.3.1 ANDROID STUDIO

Android Studio este mediul oficial de dezvoltare integrată (IDE) pentru dezvoltarea aplicațiilor Android, creat de Google. Lansat pentru prima dată în mai 2013 la conferința Google I/O, Android Studio este bazat pe IntelliJ IDEA, un IDE foarte popular dezvoltat de JetBrains. Acest mediu de dezvoltare reprezintă un instrument esențial pentru dezvoltatorii Android, combinând un set cuprinzător de funcționalități și instrumente care sprijină întregul ciclu de viață al dezvoltării aplicațiilor, de la scrierea codului și designul interfeței, până la testare, debugging și implementare.

a. Editor de Cod Avansat:

- **Autocompletare:** Oferă sugestii inteligente și completare automată pentru diverse API-uri, facilitând scrierea eficientă a codului.
- **Evidențierea Sintaxei:** Ajută la identificarea rapidă a erorilor și la citirea mai ușoară a codului.
- **Navigare Rapidă:** Permite dezvoltatorilor să se deplaseze rapid între clase, metode și resurse, îmbunătățind productivitatea.

b. Designer de Interfață Vizuală:

- **Layout Editor:** Un editor de interfață drag-and-drop pentru crearea și editarea layout-urilor XML, făcând proiectarea interfețelor mai intuitivă.
- **Preview în Timp Real:** Oferă posibilitatea de a vizualiza modificările de interfață în timp real pe diverse dimensiuni și rezoluții de ecran, asigurând compatibilitatea interfeței cu o gamă largă de dispozitive.

c. Instrumente de Testare și Debugging

- **Android Emulator:** Permite testarea aplicațiilor pe diferite versiuni de Android și configurații hardware, simulând comportamentul real al dispozitivelor.
- **Profiler de Performanță:** Oferă instrumente pentru monitorizarea performanței aplicației, inclusiv utilizarea CPU, memorie și rețea.

d. Sistem de Build Flexibil

- **Gradle:** Sistemul de build bazat pe Gradle permite configurarea avansată a build-urilor și gestionarea dependențelor, facilitând procesele de dezvoltare și implementare continuă.

e. Integrare cu Platformele de Cloud

- **Firebase:** Oferă suport integrat pentru serviciile Firebase, inclusiv autentificare, baze de date în timp real, analytics și notificări push, accelerând dezvoltarea aplicațiilor și îmbunătățind experiența utilizatorului.

Mai multe informații legate de Arduino IDE se regăsesc în [12].

3.3.2 JAVA

Java este un limbaj de programare de nivel înalt, orientat pe obiecte, dezvoltat de James Gosling și echipa sa la Sun Microsystems (acum parte a Oracle Corporation) și lansat pentru prima dată în 1995. Java este cunoscut pentru sintaxa sa clară și robustă, precum și pentru principiul "Write Once, Run Anywhere" (WORA), care permite rularea codului Java pe orice platformă care are instalată o mașină virtuală Java (JVM).

Java este cunoscut pentru portabilitatea sa, datorită codului compilat (bytecode) care poate fi executat pe orice dispozitiv cu Java Virtual Machine (JVM), indiferent de arhitectura hardware sau sistemul de operare. Limbajul promovează programarea orientată pe obiecte (OOP), implementând principii fundamentale precum încapsularea, moștenirea și polimorfismul, care permit dezvoltatorilor să creeze aplicații modulare și reutilizabile.

Java oferă o bibliotecă standard extinsă, cu API-uri diverse ce acoperă o gamă largă de funcționalități, de la structuri de date și colecții, la intrare/ieșire, rețelistică, securitate și interfețe grafice. Modelul său de securitate robust include gestionarea siguranței memoriei și posibilitatea de a rula aplicații în medii izolate (sandbox).

Gestionarea automată a memoriei prin garbage collection elimină necesitatea ca dezvoltatorii să administreze manual alocarea și de-alocarea memoriei. Java suportă programarea multithreaded, permițând dezvoltarea de aplicații care pot efectua mai multe sarcini simultan, contribuind astfel la eficiența și performanța aplicațiilor.

Acest limbaj de programare poate fi folosit în dezvoltarea unei game largi de tipuri de aplicații cum ar fi: Enterprise, Mobile, Web, Desktop și Android. Mai multe informații legate de Arduino IDE se regăsesc în [13].

3.3.3 PROGRAMAREA ORIENTATĂ PE OBIECTE

Programarea Orientată pe Obiecte (OOP) este un model de programare care organizează software-ul în obiecte, instanțe ale unor clase, fiecare cu atribute și comportamente specifice. Acest model de programare este fundamental pentru dezvoltarea de aplicații modulare, reutilizabile și scalabile.

Încapsularea se referă la restricționarea accesului direct la unele dintre componentele unui obiect, ceea ce contribuie la protejarea integrității datelor și ascunderea complexității interne. Atributele unui obiect sunt accesibile și modificate doar prin metode publice (getters și setters), oferind un nivel de control și securitate.

Moștenirea permite unei clase (subclasă) să preia proprietățile și metodele unei alte clase (superclasă), facilitând reutilizarea codului și extinderea funcționalităților. Prin moștenire, clasele pot adăuga sau modifica comportamentele moștenite fără a afecta alte părți ale programului.

Polimorfismul permite obiectelor de diferite tipuri să fie tratate printr-o interfață comună, facilitând interacțiunea și interoperabilitatea între obiecte. Această caracteristică permite utilizarea unor metode cu același nume în clase diferite, dar cu implementări specifice fiecărei clase.

Abstractizarea implică simplificarea complexității prin expunerea doar a detaliilor esențiale și ascunderea implementărilor specifice. Acest principiu se realizează prin intermediul claselor abstracte și interfețelor, care definesc metode fără a furniza implementarea lor detaliată.

Programarea orientată pe obiecte aduce numeroase beneficii, inclusiv:

- **Reutilizarea Codului:** Moștenirea și încapsularea permit reutilizarea eficientă a codului și reducerea redundanței.
- **Modularitate:** Programele sunt structurate în module independente, care pot fi dezvoltate și testate separat.
- **Scalabilitate și Întreținere:** Codul modular și reutilizabil face programele mai ușor de extins și de întreținut.
- **Claritate și Organizare:** Structurarea codului în obiecte și clase face programele mai clare și mai ușor de înțeles.

Programarea Orientată pe Obiecte continuă să fie un standard în dezvoltarea software, datorită capacității sale de a organiza codul într-un mod logic și eficient, facilitând dezvoltarea de aplicații robuste și scalabile. Mai multe informații legate de Arduino IDE se regăsesc în [14].

4. SPECIFICAȚII TEHNICE

Aceste specificații asigură funcționarea corectă, securitatea și utilizarea corectă a prototipului. Iată câteva specificații tehnice:

- Microcontrolerul trebuie să prezinte performanțe adecvate pentru prelucrarea datelor de la senzorul de amprentă și gestionarea comunicațiilor wireless.
- Senzorul de amprentă trebuie să includă o rezoluție minimă de 500 DPI pentru o recunoaștere precisă, un timp de verificare de sub o secundă pentru a asigura acces rapid și o capacitate de stocare de minim 200 de amprente pentru a permite accesul mai multor utilizatori.
- Modulul de comunicare trebuie să fie compatibil cu telefonul mobil pe care se rulează aplicația de monitorizare a încuietorii inteligente, cum ar fi Bluetooth.
- Este nevoie de prezența unui port pentru alimentare a circuitului de tip Micro USB.
- Servo Motorul trebuie să fie capabil să efectueze rotații precise pentru blocarea și deblocarea mecanică a ușii.
- Este nevoie de prezența unui senzor magnetic pentru detectarea stării ușii fie deschisă fie închisă.

Cerințele funcționale ale ansamblului hardware sunt specifice unei încuietori inteligente cum ar fi:

- Citirea amprentelor
- Comunicare cu modulele de rețea
- Gestionarea alimentării
- Notificarea aplicației.

Aplicația mobilă trebuie să conțină o serie de funcționalități, care să fie compatibile cu comportamentul încuietorii inteligente, cum ar fi:

- Autentificarea prin e-mail și parolă pentru un plus de securitate folosind o bază de date.
- Activarea comunicării prin bluetooth.
- Conectarea la încuietoria inteligentă.
- vizualizarea istoricului încuietorii privind momentul și utilizatorul care a efectuat deblocarea ușii.
- Deblocarea ușii prin trimiterea unui mesaj către circuit.
- Afișarea de notificări în afara aplicației pentru o monitorizare continuă a încuietorii.

5. IMPLEMENTARE

Proiectul este împărțit în cele două mari categorii Hardware și Software care împreună alcătuiesc produsul acestei idei. Legătura dintre ele este realizată folosind conceptul de IOT prin comunicarea bluetooth fiind fezabilă în acest context datorita distanței reduse de la care cele două componente comunică, fluxul de date este mic și redus la câteva posibile mesaje doar în cazul unor evenimente specifice.

5.1 COMPONENTELE HARDWARE

Componenta hardware a proiectului este reprezentată de circuitul care efectuează blocarea și deblocarea fizică a ușii. Designul a fost realizat urmând ideea unui prototip cât mai apropiat de potențialul unui produs final al acestei idei. Componentele au fost alese în funcție de simplitatea pe care o oferă pentru a putea simula cat mai ușor posibilele funcționalități.

Un alt criteriu care a contribuit la selecția lor a fost dimensiunea. Fiind un dispozitiv care trebuie atasat unei iale de la ușă care ea în sine este de dimensiuni mici, am încercat să îl fac cât mai mic posibil și din pricina aspectului dar și pentru a simplifica montarea reducând cât de mult posibil intervenția asupra ușii. Am încercat să realizez un design care să fie cât mai general, să se potrivească la cât mai multe modele de iale.

Componentele principale ale unei încuietori inteligente sunt microcontrolerul și servo motorul. Celelalte componente reprezintă partea de personalizare a prototipului fiind posibile nenumarate combinații.

5.1.1 MICROCONTROLLER

În rolul de microcontroler am ales să folosesc o plăcuță Arduino de tip Arduino Nano v.3. Este o placă de dezvoltare mică, completă și compatibilă cu breadboard-ul, bazată pe microcontrolerul ATmega328P.

Această placuță deține următoarele specificații tehnice:

- a. tensiune de operare: 5V
- b. tensiune de intrare (recomandată): 7-12V
- c. tensiune de intrare (limite): 6-20V
- d. pini digitali de I/O: 14 (dintre care 6 pot fi folosiți ca ieșiri PWM)
- e. pini analogici de intrare: 8
- f. curent DC per pin I/O: 40 mA
- g. curent DC pentru pinul 3.3V: 50 mA
- h. memorie flash: 32 KB dintre care 2 KB sunt utilizați de bootloader
- i. SRAM: 2 KB
- j. EEPROM: 1 KB
- k. frecvență de ceas: 16 MHz

Toate acestea sunt reprezentate în Figura 5.1.

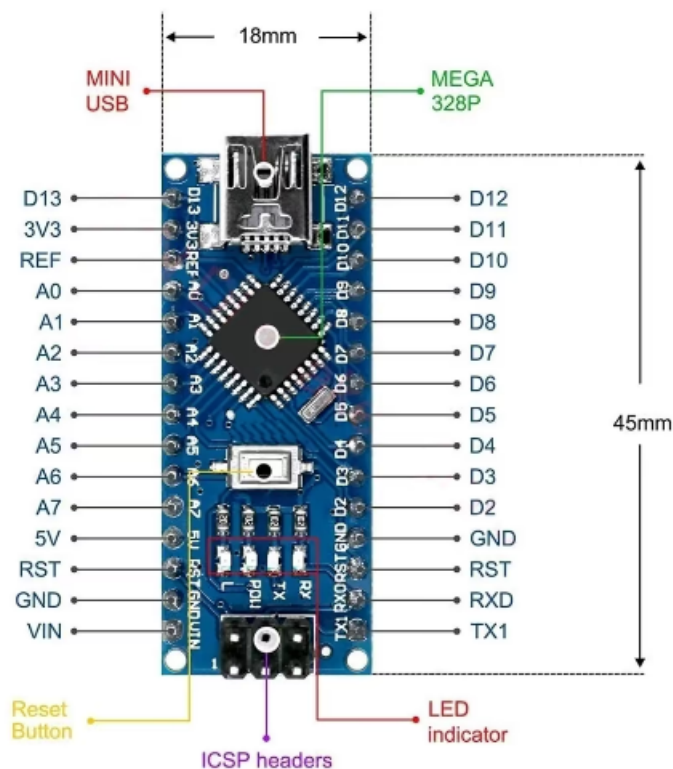


Figura 5.1: Arduino Nano v.3 (sursa: [8])

Arduino Nano utilizează un port Mini-USB atât pentru programare, cât și pentru alimentare. Placa vine cu un bootloader preinstalat, permițând programarea prin intermediul USB fără a necesita un programator hardware suplimentar. Programarea se realizează prin IDE-ul Arduino.

Arduino Nano are câteva caracteristici avantajoase pentru care este mai bună în comparație cu alte plăcuțe. Dimensiunile sale foarte compacte o fac potrivită pentru proiecte care necesită un spațiu redus. Compatibilitatea cu breadboard-ul permite montarea directă pentru prototipare rapidă. De asemenea, Arduino Nano poate fi resetată automat prin software atunci când se face upload-ul unui program, eliminând astfel necesitatea resetării manuale.

Comparativ cu alte plăcuțe Arduino, Arduino Nano este mult mai mică decât Arduino Uno și Arduino Mega, ceea ce o face ideală pentru proiecte cu spațiu limitat. Oferă aproape aceeași funcționalitate ca Arduino Uno, are mai puțini pini de I/O și o memorie puțin mai mică în comparație cu alte plăci mai mari. Toate aceste informații și nu numai se regăsesc la sursa [8].

5.1.2 SERVO MOTOR

Pentru acțiunea de rotire a cheii am ales să folosesc motorul servo SG90 9g, datorită dimensiunii sale compacte, greutății reduse și costului accesibil. Acesta efectuează cele două tipuri de rotații, de deschidere/închidere prin apelarea celor două funcții definite în Fragmentul 5.1. Acest cod a fost preluat din sursa [15].


```
1 void sweepc() //incuiere
2 {
3     if(lock==false)
4     {
5         myservo.attach(5); //This function is used to attach a servo motor
6             to a specific pin on the Arduino board
7         //il porneste
8         //Serial.print(" roteste motor ");
9
10        for (pos = 20; pos <= 180; pos += 1)
11        { // goes from 0 degrees to 180 degrees
12            // in steps of 1 degree
13            myservo.write(pos); // tell servo to go to position in variable '
14                pos'
15            delay(15); // waits 15ms for the servo to reach the position
16        }
17        lock=true;
18        Serial.println("Door is locked!");
19        myservo.detach(); //This function is used to detach a servo motor
20            from the pin it is currently attached to.
21        //il opreste
22    }
23 }
24 void sweepcc() //descuiere
25 {
26     if(lock)
27     {
28         myservo.attach(5);
29         for (pos = 180; pos >= 20; pos -= 1)
30         { // goes from 180 degrees to 20 degrees
31             myservo.write(pos); // tell servo to go to position in variable '
32                 pos'
33             delay(15);
34         }
35         lock = false;
36         //Serial.println("Door is unlocked!");
37         myservo.detach();
38     }
39 }
```

Fragmentul 5.1: Funcțiile de rotație

Acest motor prezintă următoarele caracteristici:

- a. tensiune de operare: 4.8 - 6.0V
- b. curenț: 0.19A@5V, 0.24A@6V
- c. viteza de funcționare: 0.12sec / 60 grade (4.8V) - 0.1 sec / 60 grade (6.0V)
- d. cuplu: 1,6 kg / cm (4,8 V)
- e. interval de temperatură: -30 + 60 °C
- f. servo Tip: servo analogic
- g. dimensiuni: 22.2mm x 11.8mm x 31mm
- h. unghi de rotație: 180° (90° în fiecare direcție)

Motorul servo SG90 are trei fire de conectare:

- a. firul maro pentru GND (împământare)
- b. firul roșu pentru VCC (alimentare)
- c. firul portocaliu pentru semnalul PWM (control)

Toate aceste aspecte fizice sunt reprezentate în Figura 5.1.

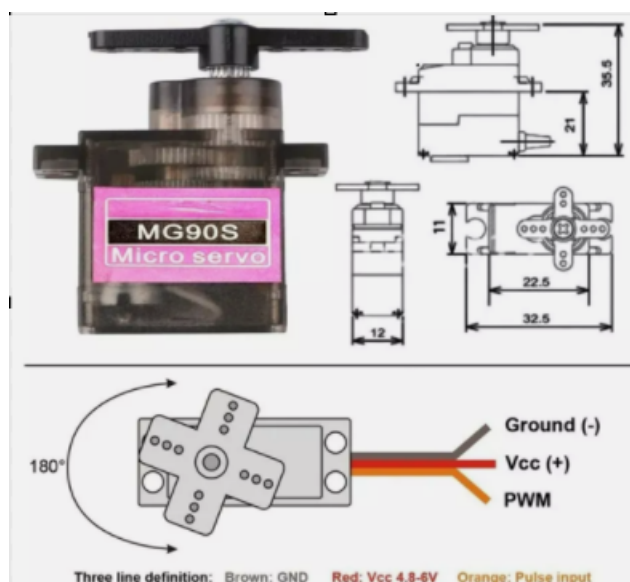


Figura 5.2: Arduino Nano v.3 (sursa: [16])

O limitare a acestui servo motor este rotația. Limita de doar 90° în fiecare direcție nu este suficientă pentru a descuria ușa. Pentru a rezolva această problemă am atașat două roți realizate la imprimantă 3D. Designul roților este reprezentat în Figura 5.3.

Din păcate nu am avut posibilitatea să printez chiar toate componentele iar roțile nu au ieșit chiar marimile potrivite așa că am fost nevoită să aduc câteva ajustări. Roțița cea mică a ieșit puțin prea mare așa că singura soluție la care m-am putut gândi a fost să o topesc puțin într-o parte și astfel am reușit să o potrivesc în design.

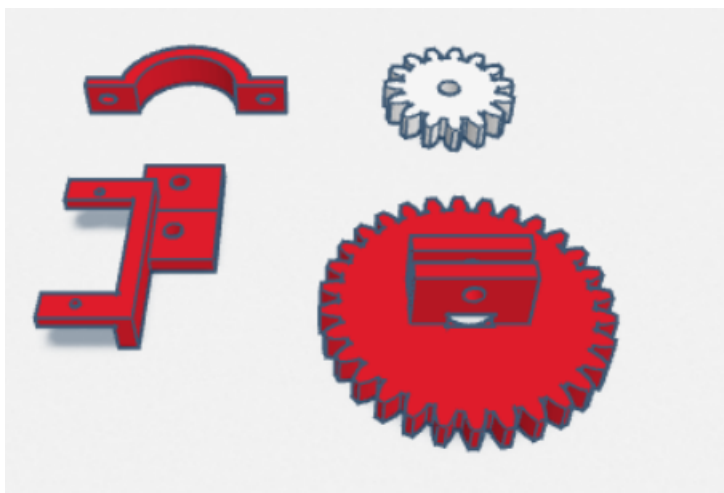


Figura 5.3: Proiectarea roțițelor (sursa: [7])

Pe lângă aceste roțițe a fost nevoie să gasesc o soluție prin care să pot monta motorul pe ușă. El va fi fixat de cleanță printr-un mâner ajustabil în lungime pentru distanța diferită dintre broască și cleanță în funcție de fiecare ușă. Din păcate nu am reușit să găsesc un model care să fie ajustabil pentru orice lungime ci doar pentru 2 iar acest design este reprezentat în Figura 5.4 .



Figura 5.4: Proiectarea mânerului (sursa: [7])

Din cauza mărimii mult prea mari a roțiței mici, din păcate nu a fost suficient să o ajustez doar pe ea. Singura componentă pe care o mai puteam modifica fără să afecteze rotirea a fost mânerul. Acestea fiind spuse am topit puțin din roțiță și puțin din mâner ca să nu modific prea mult componentele și să nu afecteze procesul de rotire.

Toate aceste piese au fost îmbinate și atașate motorului cu ajutorul unor șuruburi de dimensiuni mici iar produsul final se regăsește în Figura 5.5 vizualizarea din față iar apoi în Figura 5.6 se regăsește vizualizarea din spate.

Ceea ce nu am reușit să proiectez și să adaug la acest mecanism este sistemul de fixare a cheii. Acesta trebuia să fie alcătuit din două lamele, de o parte și de alta a mânerului cheii, lipite de roțița cea mare. Cheia fiind introdusă în broasca ușii și fixată între lamele, o dată cu rotirea cestora ar fi fost rotită și ea.

O altă componentă care lipsește din proiectare este partea de prindere a clanței. După cum putem observa în Figura 5.5 în partea de sus a mânerului actual se afla o curbura. Bucata lipsă ar fi fost atașată în continuarea mânerului formând o gaură rotundă unde ar fi fost introdusă clanța de la ușă.

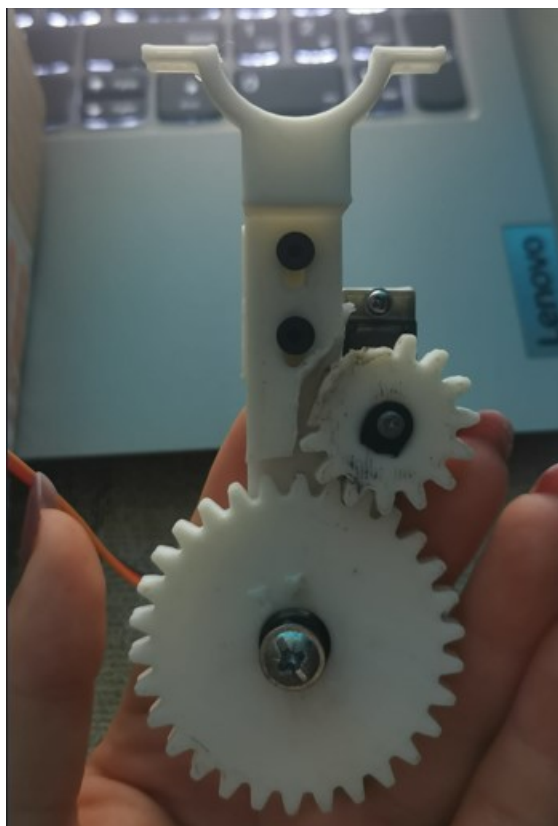


Figura 5.5: Sistemul de rotire vizualizat din față

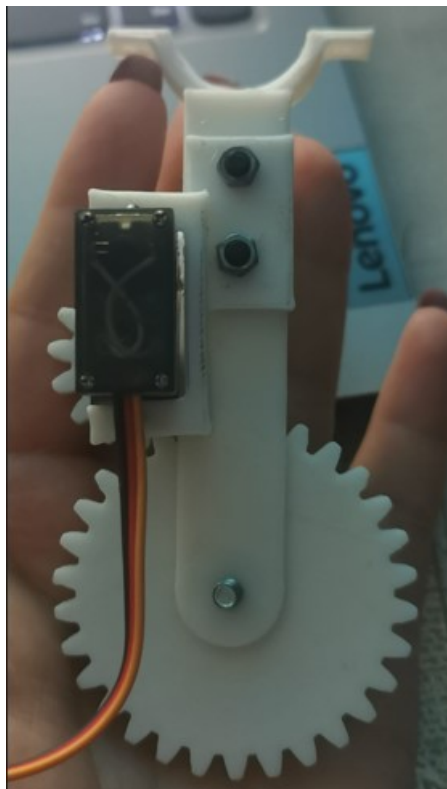


Figura 5.6: Sistemul de rotire vizualizat din spate

5.1.3 COMPONENTE ADIȚIONALE ALE CIRCUITULUI

În această subsecțiune a lucrării vor fi prezentate componentele care fac parte din procesul de personalizare al ideii de încuietoare inteligentă. Fiind un prototip destul de ușor de implementat datorită tehnologiei din zilele noastre, ideea de încuietoare inteligentă nu mai este impresionantă doar prin nume. Acestea fiind spuse am încercat să recreăm această idee cât de original posibil.

Pentru a monitoriza starea ușii, închisă/deschisă, am ales să folosesc un comutator magnetic reprezentat în Figura 5.7. Este conceput ca și un ansamblu mic de comutatoare cu lamelă, special pentru a alerta atunci când ușile, sertarele sau orice altă deschidere se deschide.

O jumătate a ansamblului este fixată de tocul ușii, iar cealaltă este atașată de ușa propriu-zisă. Când setul de comutatoare este separat, contactul este întrerupt și declanșează o alarmă. Când fiecare piesă magnetică se află la o distanță de până la 20mm una de cealaltă, ele completează circuitul cu ajutorul comutatoarelor cu lamelă interne. Ansamblul poate accepta o tensiune maximă de până la 100V la 500mA. În cadrul proiectului nostru declanșarea alarmei constă în notificarea motorului doar pentru rotația de blocare a ușii atunci când magnetii sunt apropiați.



Figura 5.7: Comutator magneti (sursa: [17])

Pentru deblocarea ușii am ales să folosesc un buton, precum cel din Figura 5.8, accesat din interiorul casei și un senzor de amprentă cu cip de colectare a imagini, precum cel din Figura 5.9, accesat din exteriorul casei.

Apăsarea butonului va notifica motorul pentru efectuarea rotației de deblocare a ușii în cazul în care comutatorul magnetic indică faptul că ușa este închisă. Caracteristici tehnice ale butonului:

- un pin pentru conectarea la GND.
- un pin pentru conectarea la un pin digital de pe placuță
- are nevoie de o rezistență de 10k între pinul digital și VCC care îl va face să funcționeze ca un rezistor pull-down.



Figura 5.8: Buton (sursa: [18])

Modulul de amprentă pe care l-am folosit în acest proiect este FPM10A, care este un senzor de amprentă compatibil cu Arduino. Acest modul vine cu memorie FLASH pentru a stoca amprente și funcționează cu orice microcontroler sau sistem cu serial TTL. De asemenea, poate fi adăugat la sisteme de securitate, încuietori de uși, sisteme de pontaj și multe altele. Caracteristici tehnice ale senzorului de amprentă:

- a. tip: optic
- b. lumina de fundal: albastru
- c. interfață: USB2.0 / UART (nivel logic TTL)
- d. suprafața de captare a imaginii: 15×11 (mm)
- e. viteză de verificare: 0,2 sec
- f. viteză de scanare: 0,3 sec
- g. dimensiunea fișierului de caractere: 256 octeți
- h. dimensiunea șablonului: 512 octeți
- i. nivelul de acceptare fals (FAR): 0.0001
- j. rata de respingere rară (FRR): 0,1
- k. rezoluție: 500 DPI
- l. tensiune: 4.2–6VDC (sau 3.3V)
- m. curent favorabil: tipic 75mA
- n. temperatura favorabilă: -20 până la 50 de grade



Figura 5.9: Senzor de amprentă (sursa: [19])

Codul necesar pentru înregistrarea unei noi amprente și verificarea ulterioară a acestora este extras din biblioteca Adafruit și l-am preluat din sursa [20]. În Fragmentul 5.2 este definită funcția de validare a amprente care apelează celelalte funcții necesare pentru verificarea acesteia.

```
1 // returns -1 if failed, otherwise returns ID #
2 int getFingerprintIDez()
3 {
4     uint8_t p = finger.getImage();
5     if (p != FINGERPRINT_OK)
6     {
7         return -1;
8     }
9     p = finger.image2Tz(); //image2Tz() -> converts the captured
        fingerprint image into a fingerprint template
10    if (p != FINGERPRINT_OK) {
11        return -1;
12    }
13    p = finger.fingerFastSearch();
14    if (p != FINGERPRINT_OK) {
15        Serial.println("Stranger at the door"); //aici nu gaseste amprenta
        -> om necunoscut
16        return -1;
17    }
18    // found a match!
19    return finger.fingerID;
20 }
```

Fragmentul 5.2: Validarea amprenteii

În circuit a mai fost adăugat și un buzzer piezoelectric, precum cel din Figura 5.10, pentru a semnaliza prin sunet faptul că urmează să se deblocheze ușa. Acesta este activat o dată cu notificarea motorului și începerea rotației de deblocare a ușii.



Figura 5.10: Buzzer (sursa: [21])

Pentru o înțelegere mai simplă a legăturilor dintre componentele circuitului am realizat o schemă bloc în Figura 5.11 unde sunt surprinse toate interacțiunile dintre acestea iar pe lângă asta, în Fragmentul 5.3 putem observa codul efectuat de către controler pentru a realiza întregul ciclu de funcționare.


```
1 void loop()  
2 {  
3   if(!digitalRead(magnet)){ // daca s-a inchis usa (ultimul ciclu  
        magnetul era "0", iar acum e "1")  
4     sweepc(); // incuie usa  
5     doorStatus=false; // usa e incuiata  
6   }  
7   if(digitalRead(magnet)){  
8     lock = false;  
9   }  
10  if(Serial.available()>0){  
11    msg = Serial.readString(); // Read the message as String  
12  }  
13  if(msg == "Open"){  
14    openDoor(false);  
15    msg = "";  
16  }  
17  if(!doorStatus && (getFingerprintIDez()>=0 || digitalRead(buttonPin)==  
        LOW)) // daca usa e inchisa(/incuiata) si avem buton sau amprenta  
18  {  
19    openDoor(true);  
20  }  
21  delay(50); //don't need to run this at full speed.  
22  lastMagnet = !digitalRead(magnet);  
23 }
```

Fragmentul 5.3: Întregul ciclu al circuitului

După cum se poate observa în schema bloc din Figura 5.11, modulul bluetooth și senzorul de amprentă folosesc protocolul de comunicare UART pentru a interacționa cu plăcuța.

UART (Universal Asynchronous Receiver/Transmitter) este un protocol de comunicare serial utilizat pe scară largă în electronica digitală pentru transferul de date între dispozitive. Acest protocolul include două componente principale:

- Transmițătorul (TX) convertește datele primite de la microcontroler într-un semnal serial și îl transmite către receptor.
- Receptorul (RX) primește semnalul serial și îl convertește înapoi în date paralele, care pot fi prelucrate de microcontroler. Această conversie permite o comunicare eficientă între diferite componente electronice.

Un cadru de date UART este compus din mai multe elemente distincte:

- Un bit de start, care este de obicei un bit 0, inițiază transferul de date.
- Între 5 și 9 biți de date, în funcție de setările de comunicare.
- O paritate opțională poate fi inclusă pentru detectarea erorilor.

- În final, unul sau doi biți de stop (de obicei biți 1) semnalează sfârșitul transferului de date. Acest format permite o structurare clară și robustă a datelor transmise.

Protocolul de comunicare UART funcționează fără a necesita un ceas comun între emițător și receptor. În schimb, utilizează un protocol de sincronizare bazat pe biți de start și stop pentru a coordona transferul de date. Aceasta permite o comunicare flexibilă și simplificată între dispozitive. Toate aceste informații au fost preluate din sursa **UARTWiki**.

O altă tehnică observată în Figura 5.11 este utilizarea unui rezistor Pull-Up. Acest tip de rezistor a fost folosit pentru conectarea butonului dar și pentru conectarea întrerupătorului magnetic la sursa de energie a microcontrolerului.

Un rezistor pull-up este o componentă electronică folosită pentru a stabili o tensiune de referință clară pe un pin de intrare digital al unui microcontroler sau a altui modul. El este conectat între pinul de intrare și tensiunea de alimentare (Vcc). În absența unui semnal de la un comutator sau buton conectat la pin, rezistorul pull-up trage pinul de intrare la un nivel logic înalt (1). Această tehnică este utilizată pentru a preveni starea de „flotare” (unde pinul nu este conectat nici la Vcc, nici la GND), asigurându-se că pinul de intrare este întotdeauna într-o stare definită. Este frecvent utilizat în butoane și comutatoare care, atunci când sunt apăstate, conectează pinul de intrare la masă (GND). Toate aceste informații au fost preluate din sursa [22].

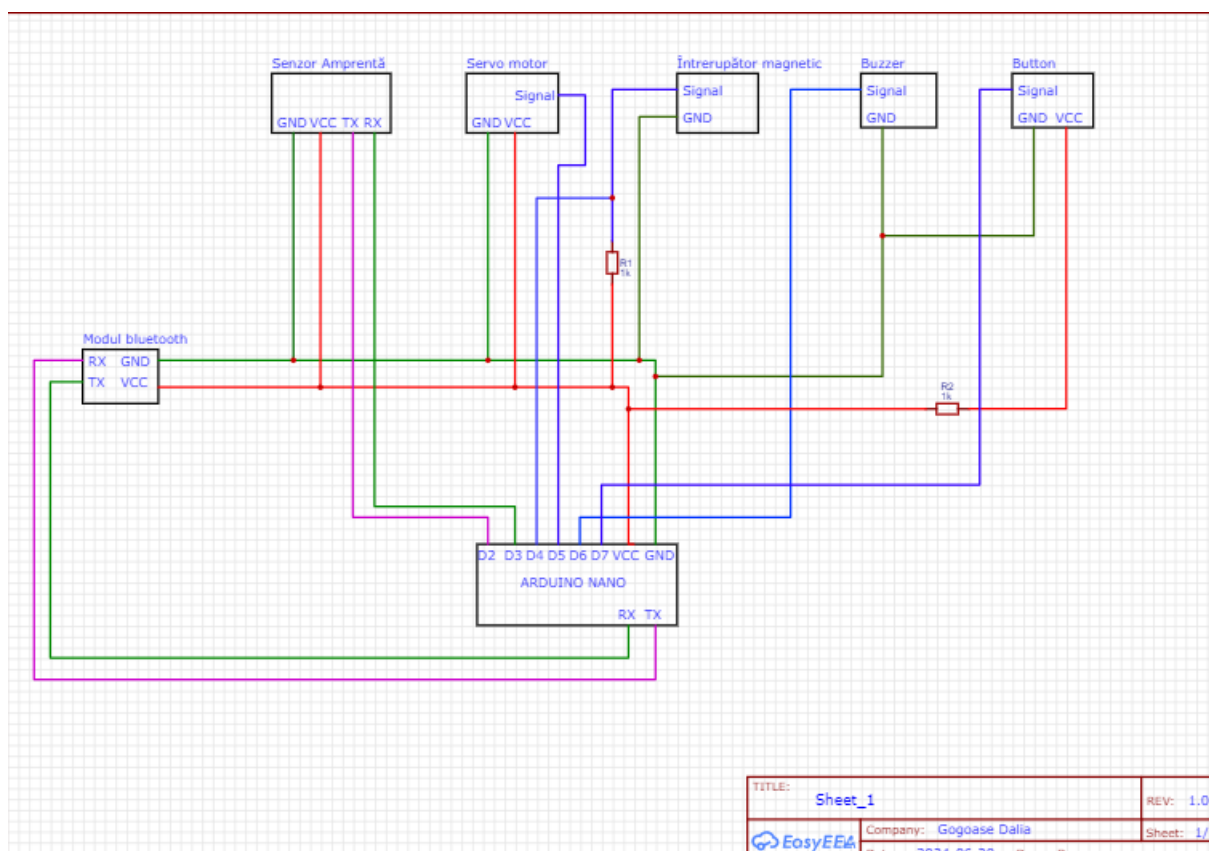


Figura 5.11: Schema bloc a circuitului

5.2 COMPONENTA SOFTWARE

Componenta software a acestui proiect a fost integrată într-o aplicație mobilă deoarece telefonul mobil este un device pe care îl avem mereu cu noi și îl folosim constant. Fiind mereu la dispoziția noastră ne permite monitorizarea constantă a stării ușii noastre prin accesul la internet.

Aceasta are ca și rol monitorizarea totală a circuitului de deblocare a ușii și controlul parțial al acestuia. Când spun monitorizare totală mă refer la faptul că o dată conectați la circuit prin bluetooth aplicația pornește un canal de broadcast prin care va genera notificări legate de activitatea încuietorii inteligente chiar dacă acestea sunt cauzate de evenimente din afara aplicației cum ar fi senzorul de amprentă sau butonul din cadrul circuitului. Când spun controlul parțial mă refer la faptul că din cadrul aplicației, singura influență pe care o putem avea asupra circuitului este de a declanșa deblocarea ușii prin trimiterea unui mesaj către circuit.

5.2.1 PRINCIPII OOP FOLOSITE

Încapsularea se referă la manevrarea datelor folosind metodele care operează asupra acestor date. Este folosită pentru a ascunde valorile sau starea unui obiect în interiorul unei clase, prevenind accesul direct la acestea. De obicei, în clasă sunt furnizate metode accesibile public (așa-numitele getters și setters) pentru a accesa valorile, iar alte clase client apelează aceste metode pentru a recupera și modifica valorile din cadrul obiectului.

Acest principiu este evidențiat prin delararea atributelor fiecărei clase ca fiind `private` pentru a nu putea fi modificate în mod neautorizat, doar funcțiile care lucrează cu aceste obiecte să fie declarate `public`. Putem observa punerea în practică a acestei teoreme în Fragmentul 5.5.

Polimorfismul este utilizarea unui singur simbol pentru a reprezenta mai multe tipuri diferite. Acest principiu permite obiectelor de tipuri diferite să fie tratate într-un mod uniform prin intermediul unei interfețe comune. Este esențial pentru realizarea flexibilității și extensibilității codului.

În cadrul codului, acest principiu este evidențiat printr-o interfață reprezentată în Fragmentul 5.4. Această interfață a fost creată deoarece aveam nevoie de utilizarea unor funcții din `MainActivity` în mai multe clase. Ne fiind o metodă fezabilă instanțierea unei activități am decis să mă folosesc de această interfață. Implementarea funcțiilor are loc în clasa `MainActivity`, după cum se poate observa în Fragmentul 5.7, iar această implementare este folosită de către `DeviceService` precum în Fragmentul 5.7 la linia (6).

Moștenirea se referă la capacitatea unei subclase de a prelua proprietățile și metodele unei superclase. Contribuie la implementarea unui cod reutilizabil, modular și deschis spre extindere.

Acest principiu l-am evidențiat prin extinderea clasei `ComponentActivity` de către `MainActivity` și extinderea clasei `AppCompatActivity` de către activitatea de `LogIn` și `Register` precum apare în Fragmentul 5.9.

Abstractizarea se referă la procesul de reducere a complexității unui sistem prin

ascunderea detaliilor irelevante pentru utilizatori și expunerea doar a aspectelor esențiale. În OOP, abstractizarea se realizează prin utilizarea claselor abstracte și a interfețelor.

În cazul nostru nu a fost nevoie de asemenea abstractizare dar am încercat să respect acest principiu păstrând codul cât mai curat și lipsit de complexitate. Un bun exemplu este în Fragmentul 5.6 la linia (6). În cadrul acestui exemplu am decurs la abstractizare deoarece funcția de *receiveMessage* devenise foarte lungă, greu de înțeles și cu o complexitate foarte mare. Am împărțit conținutul funcției în mai multe funcții de complexități reduse astfel codul este mult mai ușor de urmărit.

```
1 public interface DeviceServiceCallback {  
2     void showToast(String message);  
3     void updatePairedDevicesList(String[] devices);  
4 }
```

Fragmentul 5.4: Interfața DeviceServiceCallback

```
1 public class UserInfo {  
2     private static UserInfo instance = null;  
3     private final String email;  
4     private UserInfo(String email) {  
5         this.email = email;  
6     }  
7     public String getEmail() {  
8         return email;  
9     }  
10    public static void setInstance(UserInfo instance) { UserInfo.  
11        instance = instance; }  
12    public static synchronized UserInfo getInstance(String email) {  
13        if (instance == null) {  
14            instance = new UserInfo(email);  
15        }  
16        return instance;  
17    }
```

Fragmentul 5.5: Gestionarea datelor privind utilizatorul actual

```
1 private void receiveMessage() {
2     Log.e("DeviceService", "Entering receiveMessage function");
3     if (socket != null && socket.isConnected()) {
4         new Thread(() -> {
5             try {
6                 handleMessages();
7             } catch (IOException e) {
8                 e.printStackTrace();
9                 callback.showToast("Failed to receive message");
10            }
11        }).start();
12    } else Toast.makeText(context, "Not connected to any device",
13        Toast.LENGTH_SHORT).show();
14 }
```

Fragmentul 5.6: Abstractizare

```
1 @Override
2 public void showToast(String message) {
3     runOnUiThread(() -> Toast.makeText(this, message, Toast.
4         LENGTH_SHORT).show());
5 }
6
7 @Override
8 public void updatePairedDevicesList(String[] devices) {
9     runOnUiThread(() -> {
10         pairedDevicesAdapter.clear();
11         pairedDevicesAdapter.addAll(devices);
12         pairedDevicesAdapter.notifyDataSetChanged();
13     });
14 }
```

Fragmentul 5.7: Implementarea Interfeței
DeviceServiceCallback

```
1 private void verifyPaierdDevicesList(Set<BluetoothDevice> bt, int index,  
   String[] strings) {  
2     if (bluetoothAdapter.isEnabled()) {  
3       if (bt.size() > 0) {  
4         extractBluetoothDevices(bt, index, strings);  
5         int max = strings.length - 2; // There was always an  
           empty row at the end for some reason;  
6         callback.updatePairedDevicesList(java.util.Arrays.  
           copyOfRange(strings, 0, max));  
7       } else  
8         Toast.makeText(context, "No paired devices found!",  
           Toast.LENGTH_SHORT).show();  
9     } else  
10      Toast.makeText(context, "Bluetooth must be enabled!", Toast.  
           LENGTH_SHORT).show();  
11 }
```

Fragmentul 5.8: Utilizarea Interfeței
DeviceServiceCallback

```
1 public class MainActivity extends ComponentActivity implements  
   DeviceServiceCallback  
2 public class LogIn extends AppCompatActivity  
3 public class Register extends AppCompatActivity
```

Fragmentul 5.9: Moștenirea

Pe lângă aceste principii de OOP, în timpul procesului de implementare am folosit și Singleton. Singleton este un design pattern creațional care asigură că o clasă are o singură instanță și furnizează un punct global de acces la aceasta. Este utilizat în situațiile în care este necesar un obiect unic pentru a coordona acțiunile dintr-un sistem. Toate aceste informații și nu numai, se pot găsi în sursa **SingletonPatternWiki**.

În implementare a fost integrat acest pattern asupra clasei, *DeviceService*. Acest serviciu ține evidența dispozitivului la care suntem conectați și manevrează datele primite de la acesta transformându-le în notificări. Având în vedere faptul că acțiunea de conectare la dispozitiv se face în altă activitate față de cea în care se gestionează accesul la bluetooth și lista de posibile dispozitive, a fost nevoia ca serviciul să aibă o singură instanță accesibil global. Acest lucru a fost posibil adăugând serviciului un atribut care ține evidența instanței curente după cum se poate observa în Fragmentul 5.10. Această tehnică a fost folosită și la gestionarea datelor utilizatorului curent pentru a putea avea acces la ele global indiferent în ce fereastră a aplicației ne aflăm, după cum se poate observa în Fragmentul 5.5.

```
1      public static synchronized DeviceService getInstance (
2          Context context,
3          MessageNotificationService messageService,
4          DeviceServiceCallback callback, FirebaseUser user) {
5          if (instance == null) {
6              instance = new DeviceService(context, messageService,
6                  callback, user);
7          }
8          return instance;
9      }
```

Fragmentul 5.10: Singleton

6. FLUXUL DE FUNCȚIONARE AL APLICAȚIEI

Aplicația mobilă se împarte în 4 activități: *Login*, *Register*, *MainActivity* și *Histsory*. *Login* și *Register* alcătuiesc funcționalitatea de autentificare, *MainActivity* este pagina principală a aplicației iar *Histry* este pagina care ne permite comunicarea cu încuietoarea inteligentă și vizualizarea istoricului.

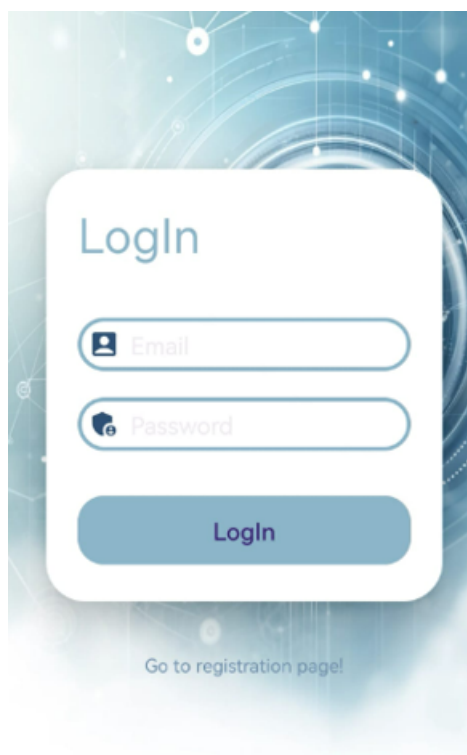
6.1 AUTENTIFICARE

Încuietoarea inteligentă se presupune a fi folosită de întreaga familie, toate persoanele care locuiesc în locuința respectivă. Dat fiind acest aspect, am decis să adaug aplicației posibilitatea de autentificare cu un cont personal pentru a putea ține evidența persoanelor care au acces la ușă. Datorită acestei funcționalități adaugate, atunci când ținem evidența deschiderilor asupra ușii vom putea avea detalii atât despre momentul deschiderii cât și despre persoana care a folosit ușa.

Această funcționalitate de autentificare constă în două acțiuni: autentificare cu un cont deja existent și înregistrarea unui cont nou.

Autentificarea cu un cont deja existent presupune completarea a doua căsuțe de text, una pentru email și una pentru parolă după cum se poate observa în Figura 6.1. După completare acestora, se apasă pe butonul de *Login* și textele vor fi verificate. Dacă sunt probleme, vor fi semnalate pe ecran prin apariția unor *toasts* după cum putem observa în Figura 6.3. După ce acestea sunt validate, aplicația va trece la fereastra principală.

Dacă este necesară crearea unui nou cont, se va apăsa pe textul *Go to registration page!*, care va duce pe pagina de înregistrare. Unde exact ca și pe pagina de autentificare, va fi nevoie de completarea celor două casuțe de text, după completarea lor se va apăsa butonul de *Register* și textele vor fi verificate. Dacă nu îndeplinesc cerințele de formatare, utilizatorul va fi notificat în legătură cu defectul textului printr-un *toast* asemănător cu cel din Figura 6.3. Dacă valorile introduse sunt formatare corespunzător atunci se va deschide fereastra de autentificare unde utilizatorul poate introduce noul cont creat.



The image shows a login form with a white background and rounded corners, centered on a blue background with a network pattern. The form has a title 'Login' in blue. Below it are two input fields: 'Email' with an envelope icon and 'Password' with a key icon. A blue 'Login' button is at the bottom. A link 'Go to registration page!' is at the very bottom.

Login

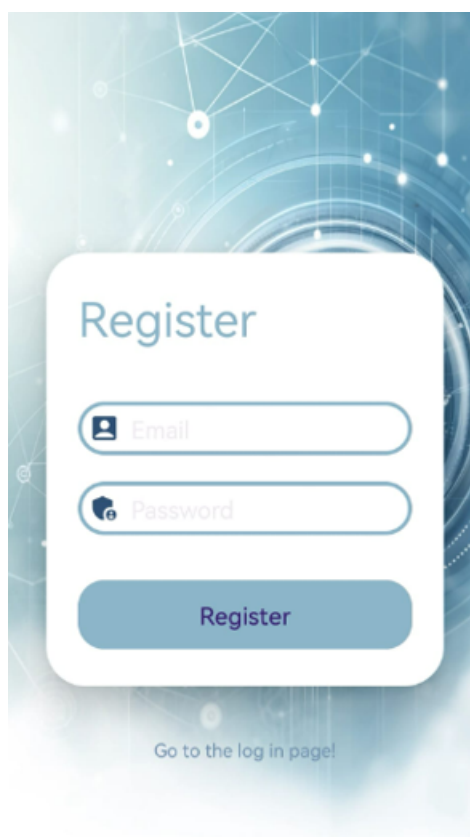
Email

Password

Login

[Go to registration page!](#)

Figura 6.1: Login



The image shows a register form with a white background and rounded corners, centered on a blue background with a network pattern. The form has a title 'Register' in blue. Below it are two input fields: 'Email' with an envelope icon and 'Password' with a key icon. A blue 'Register' button is at the bottom. A link 'Go to the log in page!' is at the very bottom.

Register

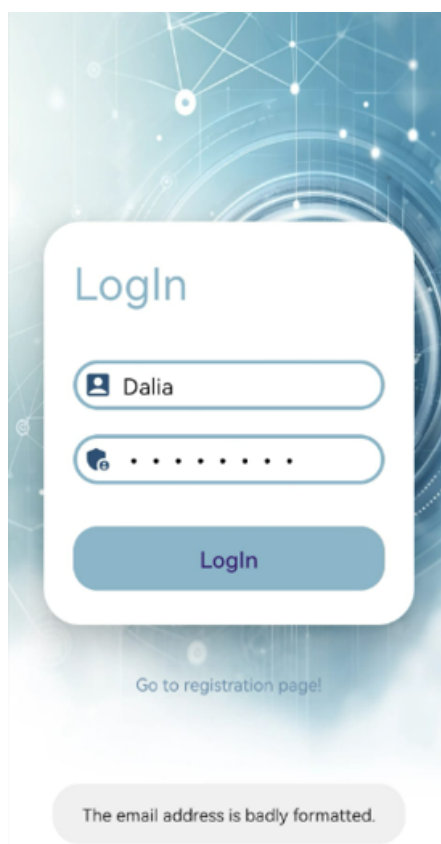
Email

Password

Register

[Go to the log in page!](#)

Figura 6.2: Register



The image shows a mobile application login screen. At the top, the word "Login" is displayed in a blue font. Below it are two input fields: the first contains the name "Dalia" next to a person icon, and the second contains a password represented by dots next to a key icon. A blue "Login" button is positioned below the password field. Underneath the button is a link that says "Go to registration page!". At the bottom of the screen, a grey error message box states "The email address is badly formatted." The background features a blue abstract design with glowing nodes and lines.

Figura 6.3: Validare date

6.2 CONTROLUL ȘI MONITORIZAREA CIRCUITULUI

După autentificarea cu succes a utilizatorului se va deschide fereastra principală din Figura 6.4. După cum putem observa sunt prezente 4 butoane: *Bluetooth On*, *Bluetooth Of*, *Show Paired Devices* și *Log Out*.

După cum sugerează și numele butonului *Log Out*, acesta ne va întoarce la pagina de autentificare din Figura 6.1 pentru cazul în care vrem să ne conectăm cu un alt cont sau să ajungem la pagina de *Register* pentru crearea unui nou cont.

Butoanele de gestionare a interacțiunii bluetooth a telefonului mobil, *Bluetooth On* și *Bluetooth Of* sunt folosite pentru a activa, conform Figura 6.5 și dezactiva, conform Figura 6.9 modulul bluetooth al telefonului. După cum putem observa în Figura 6.4, butonul *Show Paired Devices* este de culoare gri deoarece acesta se află în stare inactivă, nu putem vizualiza lista de potențiale dispozitive dacă rețeaua bluetooth nu este activată. După activare, butonul va deveni de aceeași culoare ca și celelalte iar userul îl va putea accesa. Dacă ulterior se va dezactiva rețeaua bluetooth, butonul nu va deveni la loc de culoare gri dar dacă utilizator îl va apăsa, va apărea un *toast* pe ecran pentru a semnala faptul ca rețeaua trebuie să fie activă precum Figura 6.8.

Dacă rețeaua bluetooth este activă, utilizatorul poate apăsa pe butonul *Show Paired Devices* iar acesta va dezvălui o listă cu potențiale încuietori inteligente la care acesta se poate conecta precum în Figura 6.7. Este de menționat faptul că pentru ca încuietoarea să apară în listă utilizatorul trebuie prima dată să intre în setările propriului telefon și să adauge dispozitivul în lista de *paired devices*.

După apariția listei de dispozitive utilizatorul poate apăsa pe unul dintre ele iar această acțiune va deschide pagina în care se regăsește istoricul ușii respective precum în Figura 6.10. Fiecare înregistrare conține informații legate despre numele încuietorii, momentul în care a avut loc deblocarea și user-ul care a inițiat această comandă din aplicație. Dacă deblocarea nu are loc prin intermediul aplicației atunci în înregistrarea deblocării, în loc de user va fi menționat faptul că a fost deblocată cu ajutorul senzorului de amprentă sau a butonului.

După cum putem observa în Figura 6.10, pe lângă lista de înregistrări se află și două butoane, *Connect* și *Unlock*. Butonul *Connect* este folosit pentru conectarea prin rețeaua bluetooth la dispozitiv. Această acțiune este realizată conform codului prezentat în Fragmentul 6.2. Dacă dintr-un oarecare motiv, fie nu este conectat la o sursă de curent dispozitivul, fie suntem prea departe fie sa defectat, fie sa deconectat între timp, telefonul mobil nu reușește să se conecteze la încuietoare, utilizatorul va fi semnalat prin apariția unui *toast* precum în Figura 6.11. Dacă acțiunea de conectare eșuează vom putea observa cum butonul *Unlock* devine indisponibil pentru utilizator și prinde o culoare gri. După ce se stabilește conexiunea cu dispozitivul, butonul revine la normal și putem trimite comanda de deblocare către încuietoare.

După cum am mai pomenit anterior în lucrare, o dată cu conectarea la un dispozitiv este activată o rețea de *broadcast*. Pentru a crea această rețea ne folosim de un obiect de tipul *BroadcastReceiver* precum este definit în Fragmentul 6.4 și funcția *registerReceiver()* definită în Fragmentul 6.5. Prin această rețea aplicația asculta constant semnale de la încuietoarea inteligentă și le transformă în notificări după cum se poate observa definit în

funcția *recvMessage()* definită în Fragmentul 6.7. Aceste notificări apar atunci când circuitul deblochează ușa, blochează ușa și detectează o amprentă străină, aceste notificări sunt definite în funcția *doorIsOpen()* integrată în funcția de *handleMessages()*. Acestea sunt definite în Fragmentul 6.6.

Notificarea încuietorii inteligente cu privire la deblocarea ușii se realizează prin codul prezent în Fragmentul 6.3 cu ajutorul funcției de *sendMessage()*. Dacă trimiterea mesajului a fost finalizată cu succes pe ecranul telefonului va apărea un *toast* cu un mesaj de confirmare după cum se poate observa în Figura 6.13. Când circuitul detectează mesajul corespunzător acesta aplică rotația de deblocare asupra cheii, așa cum se poate observa în Fragmentul 6.1. După finalizarea procesului de deblocare, circuitul trimite un mesaj către aplicație precum în Fragmentul 6.1 la linia (12). Procesul este finalizat prin apariția unei notificări precum în Figura 6.15. După ce ușa este închisă apropiind cei doi magneti, se efectuează blocarea ușii finalizând acțiunea prin transmiterea unui mesaj care este transformat în notificare precum în Figura 6.16.

Dacă în timp ce suntem conectați la încuietoarea inteligentă, este detectată o amprentă străină, vom fi din nou notificați precum în Figura 6.14.

Toate aceste notificări apar pe ecran chiar dacă nu mai este deschisă aplicația, atâta timp cât suntem conectați la circuit iar serviciul aplicației este încă în desfășurare. Acest lucru poate fi observat în Figura 6.17.

```
1 void openDoor(boolean id) {  
2     digitalWrite(buzzer, HIGH);  
3     delay(200);  
4     digitalWrite(buzzer, LOW);  
5     sweepcc(); // descuie usa  
6     delay(1000);  
7     doorStatus=true; // usa e descuiata  
8     if(id == true) {  
9         Serial.println("Door is unlocked!ID");  
10    }  
11    else{  
12        Serial.println("Door is unlocked!");  
13    }  
14    while(true) {  
15        if(digitalRead(magnet) && lastMagnet) break;  
16    }  
17 }
```

Fragmentul 6.1: Deblocarea ușii de către circuit

```
1  public boolean connectToDevice() {
2      if (bluetoothAdapter.isEnabled()) {
3          connectedDevice = bluetoothAdapter.getRemoteDevice(
4              deviceAddress);
5          try {
6              socket = connectedDevice.
7                  createRfcommSocketToServiceRecord(MY_UUID);
8              socket.connect();
9              Toast.makeText(context, "Connected to " +
10                  connectedDevice.getName(), Toast.LENGTH_SHORT).show();
11              ;
12              receiveMessage();
13              return true;
14          } catch (IOException e) {
15              e.printStackTrace();
16              Toast.makeText(context, "Connection failed", Toast.
17                  LENGTH_SHORT).show();
18              try {
19                  if (socket != null) {
20                      socket.close();
21                  }
22              } catch (IOException closeException) {
23                  closeException.printStackTrace();
24              }
25          }
26      } else Toast.makeText(context, "Bluetooth must be enabled!",
27          Toast.LENGTH_SHORT).show();
28      return false;
29  }
```

Fragmentul 6.2: Conectarea la încuietoare

```
1 public void sendMessage(String message) {
2     if (socket != null && socket.isConnected()) {
3         try {
4             OutputStream outputStream = socket.getOutputStream();
5             outputStream.write(message.getBytes());
6             outputStream.flush();
7             Toast.makeText(context, "Message sent: " + message,
8                 Toast.LENGTH_SHORT).show();
9         } catch (IOException e) {
10            e.printStackTrace();
11            Toast.makeText(context, "Failed to send message", Toast.
12                LENGTH_SHORT).show();
13        }
14    } else Toast.makeText(context, "You must connect to device first
15        !", Toast.LENGTH_SHORT).show();
16 }
```

Fragmentul 6.3: Deblocarea ușii din intermediul aplicației

```
1 BroadcastReceiver discoverReciver = new BroadcastReceiver() {
2     @Override
3     public void onReceive(Context context, Intent intent) {
4         String action = intent.getAction();
5         if (BluetoothDevice.ACTION_FOUND.equals(action)) {
6             BluetoothDevice device = intent.getParcelableExtra(
7                 BluetoothDevice.EXTRA_DEVICE);
8             if (device != null) {
9                 String deviceInfo = device.getName() + "\n" + device
10                    .getAddress();
11                 Log.d("MainActivity", "Discovered device: " +
12                    deviceInfo);
13                 //disciverableDevices.add(device.getName() + "\n" +
14                    device.getAddress());
15                 //discoverableDevacesAdapter.notifyDataSetChanged();
16             }
17         }
18     }
19 };
```

Fragmentul 6.4: Obiectul de tip BroadcastReceiver

```
1 public void registerReceiver(BroadcastReceiver receiver,
2     IntentFilter filter) {
3     context.registerReceiver(receiver, filter);
4 }
```

Fragmentul 6.5: Funcția de registerReciver()

```
1 private void handleMessage() throws IOException {
2     InputStream inputStream = socket.getInputStream();
3     if (inputStream == null) {
4         Log.e("DeviceService", "Input stream is null");
5         return;
6     }
7     BufferedReader reader = new BufferedReader(new InputStreamReader
8         (inputStream));
9     String receivedMessage;
10    while ((receivedMessage = reader.readLine()) != null) {
11        Log.d("DeviceService", "Received message: " +
12            receivedMessage);
13        String finalReceivedMessage = receivedMessage;
14        callback.showToast("Received: " + finalReceivedMessage);
15        Log.d("DeviceService", "Inainte de notificare");
16        dorIsOpen(finalReceivedMessage);
17    }
18    private void dorIsOpen(String finalReceivedMessage) {
19        if (finalReceivedMessage.equals("Stranger at the door")) {
20            setNotificationMessage(finalReceivedMessage, "You might be in
21                danger!");
22        }
23        if (finalReceivedMessage.contains("Door is unlocked!")) {
24            Boolean fingerprint = false;
25            if (finalReceivedMessage.contains("ID")) fingerprint = true;
26            addDataToDataBase(fingerprint);
27            setNotificationMessage(finalReceivedMessage, "");
28        }
29        if (finalReceivedMessage.equals("Door is locked!")) {
30            setNotificationMessage(finalReceivedMessage, "");
31        }
32    }
33 }
```

Fragmentul 6.6: Funcția de creare a notificărilor

```
1 private void receiveMessage() {  
2     Log.e("DeviceService", "Entering receiveMessage function");  
3     if (socket != null && socket.isConnected()) {  
4         new Thread(() -> {  
5             try {  
6                 handleMessages();  
7             } catch (IOException e) {  
8                 e.printStackTrace();  
9                 callback.showToast("Failed to receive message");  
10            }  
11        }).start();  
12    } else Toast.makeText(context, "Not connected to any device",  
13        Toast.LENGTH_SHORT).show();  
14 }
```

Fragmentul 6.7: Funcția de receiveMessage()



Figura 6.4: Pagina Principală

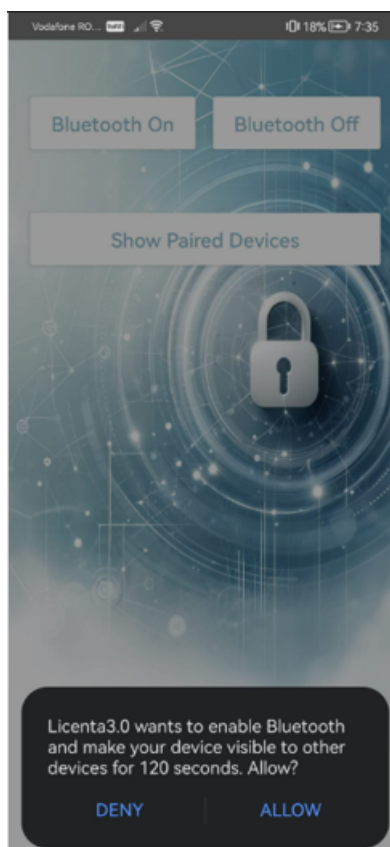


Figura 6.5: Activare Bluetooth



Figura 6.6: Activarea butonului de dezvăluire a potențialelor dispozitive

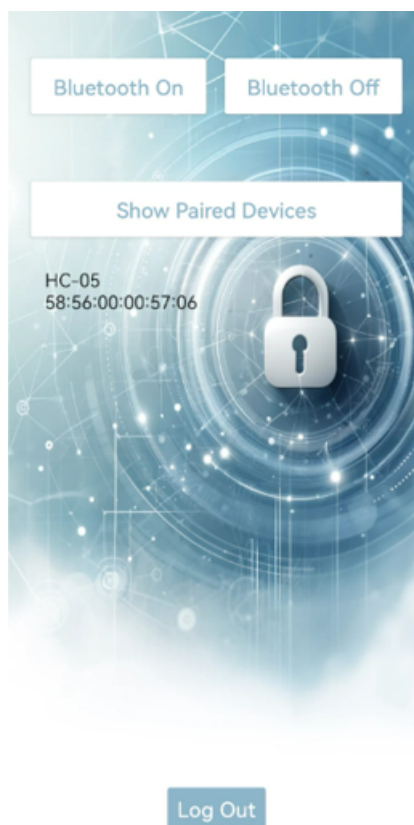


Figura 6.7: Lista dispozitivelor



Figura 6.8: Încercarea de vizualizare a potențialelor dispozitive

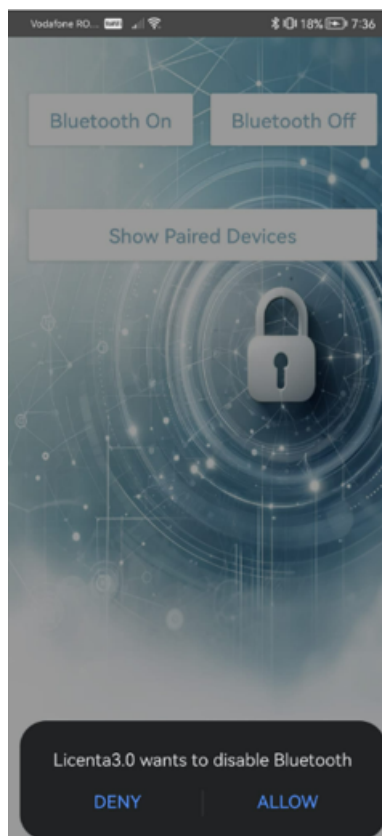


Figura 6.9: Dezactivare Bluetooth



Figura 6.10: Istoric

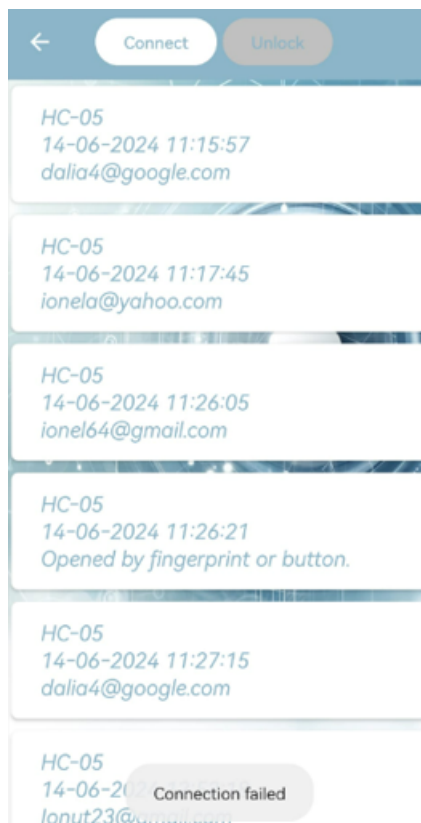


Figura 6.11: Eroare la conectare

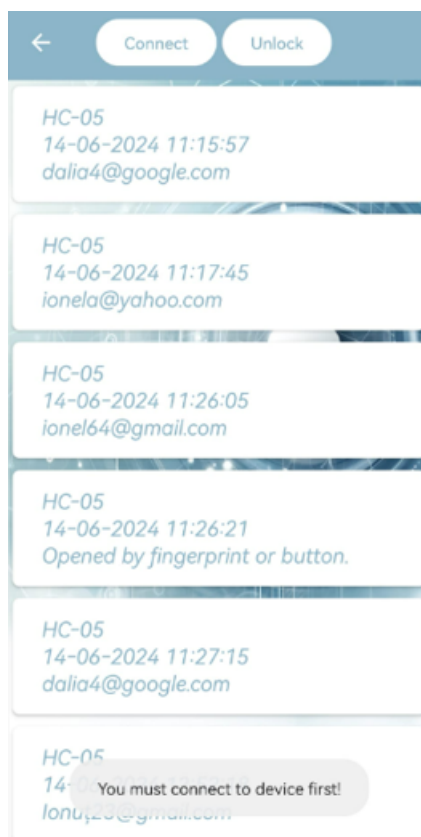


Figura 6.12: Niciun dispozitiv conectat

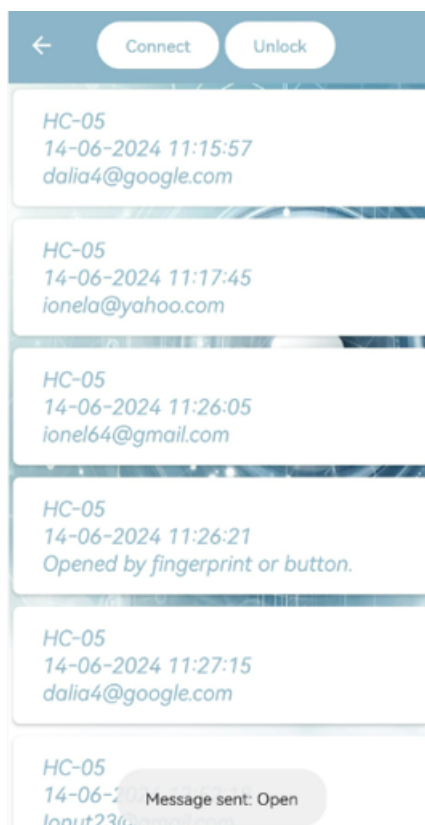


Figura 6.13: Mesaj trimis cu succes

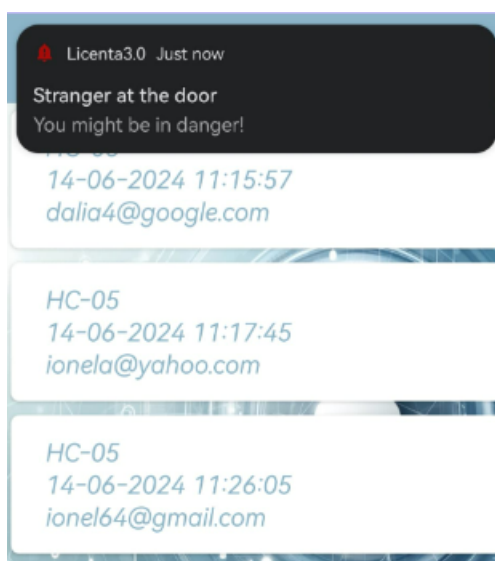


Figura 6.14: Notificare în cazul de amprentă străină

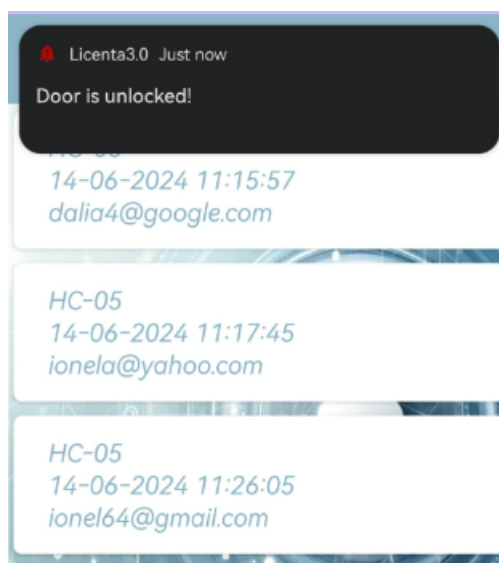


Figura 6.15: Notificare pentru deschiderea ușii

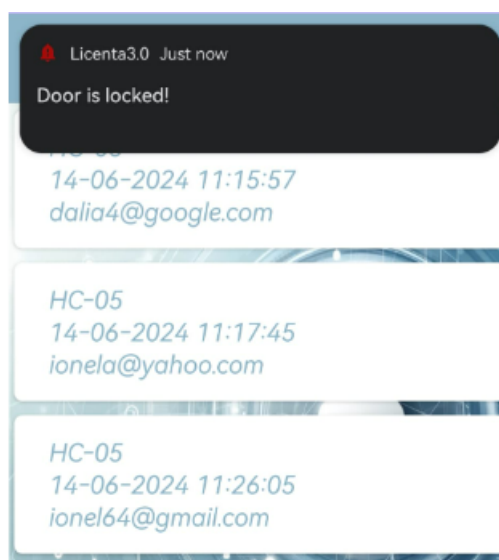


Figura 6.16: Notificare pentru închiderea ușii

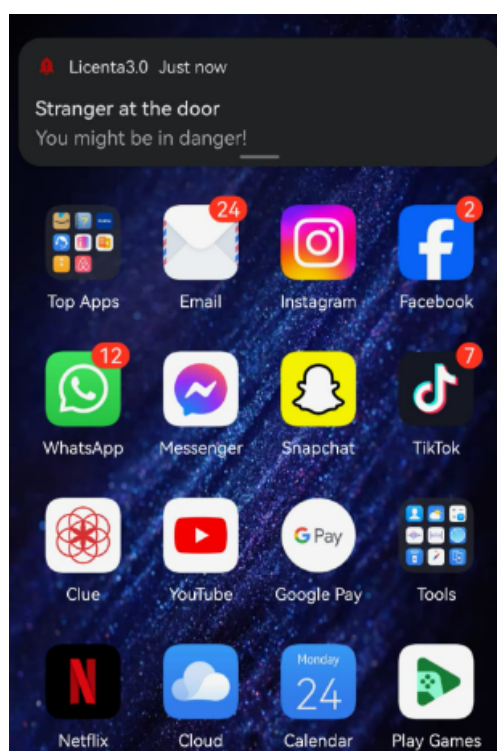


Figura 6.17: Notificare în afara aplicației

7. TESTARE

Testarea este procesul de evaluare a unui sistem pentru a verifica dacă acesta îndeplinește specificațiile și cerințele stabilite și pentru a identifica eventualele defecte sau probleme. Scopul principal al testării este de a asigura calitatea, funcționalitatea, securitatea și performanța produsului. Testând asigurăm că produsul final este fiabil și satisfăcător pentru utilizatori. Testarea poate fi realizată prin diverse metode, manuală sau automatizată, și poate acoperi diferite tipuri de teste:

- Testele funcționale verifică toate funcționalitățile sistemului cu scopul de a confirma funcționarea corectă a acestora conform specificațiilor.
- Testele de integrare asigură că diferite module ale sistemului interacționează corect.
- Testele de performanță Evaluează comportamentul sistemului sub diferite condiții de încărcare.
- Testele de securitate evaluează vulnerabilitățile de securitate privind transferul și accesul la date confidențiale.

Singurul tip de testare efectuat asupra acestui prototip este Testarea Manuală. Toate aceste teste au fost efectuate urmând niște scenarii de test. Scenariile de test reprezintă descrieri detaliate ale situațiilor și condițiilor pe care trebuie să le verifice un tester pentru a valida funcționalitatea, performanța și securitatea unei aplicații software. Ele sunt utilizate pentru a ghida procesul de testare și pentru a asigura că toate aspectele critice ale sistemului sunt verificate. Un scenariu de test este adesea compus din mai multe cazuri de test, fiecare având propriile intrări, condiții de execuție și rezultate așteptate.

Am întocmit câteva scenarii de test pentru a confirma buna funcționare a prototipului dar și pentru identificarea unor posibile defecte care necesită îmbunătățire.

7.1 SCENARII DE TESTARE

7.1.1 ACTIVAREA ȘI DEZACTIVAREA REȚELEI BLUETOOTH PE TELEFONUL MOBIL

a. Precondiții:

- Circuitul este conectat la o sursă de curent.
- Dispozitivul este dăruit în lista de Paired Devices a telefonului.

b. Pași de Test:

- Utilizatorul apasă pe butonul Bluetooth On.
- După apariția unui toast care cere permisiunea activării rețelei, utilizatorul apasă pe butonul Allow.
- Utilizatorul apasă pe butonul Bluetooth Off.

- După apariția unui toast care cere permisiunea dezactivării rețelei, utilizatorul apasă pe butonul Allow.

c. Rezultate Așteptate:

- După apăsarea butonului Bluetooth On, apare pe ecran un toast pentru a cere permisiunea activării rețelei.
- după apăsarea butonului Allow pentru activare, rețeaua bluetooth este activă.
- Butonul Show Paired Devices devine activ.
- după apăsarea butonului Allow pentru dezactivare, rețeaua bluetooth este inactivă.
- La apăsarea butonului Show Paired Devices apare un toast care sugerează faptul că trebuie activată rețeaua bluetooth.

d. Rezultate Obținute:

- După apăsarea butonului Bluetooth On, a apărut pe ecran un toast Pentru a cere permisiunea activării rețelei.
- După apăsarea butonului Allow rețeaua bluetooth a devenit activă.
- După apăsarea butonului Bluetooth Off, a apărut pe ecran un toast Pentru a cere permisiunea dezactivării rețelei.
- după apăsarea butonului Allow pentru dezactivare, rețeaua bluetooth este inactivă.
- La apăsarea butonului Show Paired Devices apare un toast care sugerează faptul că trebuie activată rețeaua bluetooth.

e. Observații:

- Aplicația a reacționat conform rezultatelor așteptate.

7.1.2 CONECTAREA LA DISPOZITIV

a. Precondiții:

- Circuitul este conectat la o sursă de curent.
- Comunicarea Bluetooth trebuie să fie activă pe telefonul mobil.

b. Pași de Test:

- Utilizatorul apasă pe butonul Show Paired Devices.
- După apariția listei de dispozitive utilizatorul apasă pe cel dorit.
- După deschiderea paginii de istoric, se apasă pe butonul Connect.

c. Rezultate Așteptate:

- după apăsarea butonului Connect apare un toast pe ecran cu un mesaj de confirmare a conectivității.

- Butonul `Unlock` devine activ.

d. Rezultate Obținute:

- După apăsarea butonului `Connect`, a apărut pe ecran un toast cu un mesaj de confirmare a conectivității.

e. Observații:

- În urma repetării acestui scenariu am observat că uneori se întrerupe conexiunea cu prototipul iar apăsând pe butonul `Connect` se declanșează un toast care alarmează eșuarea conectivității.

7.1.3 DEBLOCAREA UȘII PRIN AMPRENTĂ

a. Precondiții:

- Circuitul este conectat la o sursă de curent.
- Aplicația mobilă este conectată la dispozitivul de încuietoare inteligentă prin Bluetooth.
- Utilizatorul are o amprentă înregistrată în sistem.

b. Pași de Test:

- Utilizatorul pune degetul cu amprenta deja înregistrată pe ecranul senzorului de amprentă.
- După deblocarea ușii, utilizatorul o deschide și o închide pentru a o bloca la loc.

c. Rezultate Așteptate:

- La amprenta înregistrată buzzerul va fi activat și se va auzi un sunet o dată cu sesizarea motorului care va începe să efectueze rotația de deblocare a ușii.
- Senzorul devine inactiv până la blocarea ușii.
- Pe telefonul mobil conectat la prototip va apărea o notificare care să anunțe deblocarea ușii.
- O dată cu deblocarea ușii apare o nouă înregistrare în lista de istoric a ușii respective.

d. Rezultate Obținute:

- Circuitul a declanșat buzzerul și motorul la detectarea unei amprente corecte.
- După deblocarea ușii a apărut și notificarea de deblocare o nouă înregistrare în istoric urmată de notificarea de blocare după închiderea ușii

e. Observații:

- Circuitul sa comportat exact cum ne am așteptat fara probleme.

7.1.4 DEBLOCAREA UȘII PRIN BUTON

a. Precondiții:

- Circuitul este conectat la o sursă de curent.
- Aplicația mobilă este conectată la dispozitivul de încuietoare inteligentă prin Bluetooth.

b. Pași de Test:

- Utilizatorul apasă pe buton.

c. Rezultate Așteptate:

- La apăsarea butonului buzzerul va fi declanșat și se va auzi un sunet o dată cu sesizarea motorului care va începe să efectueze rotația de deblocare a ușii. Pe telefonul mobil conectat la prototip va apărea o notificare care să anunțe deblocarea ușii.
- O dată cu deblocarea ușii pare o nouă înregistrare în lista de istoric a ușii respective.

d. Rezultate Obținute:

- Circuitul a declanșat buzzerul și motorul la apăsarea butonului.
- După deblocarea ușii a apărut și notificarea de deblocare și o nouă înregistrare în istoric.

e. Observații:

- Circuitul sa comportat exact cum ne-am așteptat, fara probleme.
- Apariția noii înregistrări nu este chiar instantanee ci are o întârziere de câteva secunde.

7.1.5 DEBLOCAREA UȘII PRIN APLICAȚIE

a. Precondiții:

- Circuitul este conectat la o sursă de curent.
- Aplicația mobilă este conectată la dispozitivul de încuietoare inteligentă prin Bluetooth.

b. Pași de Test:

- Utilizatorul apasă pe butonul `Unlock`. din cadrul aplicației

c. Rezultate Așteptate:

- La apăsarea butonului apare un mesaj care confirmă trimiterea mesajului.
- Este delanșat buzzerul împreună cu rotația de deblocare a ușii.
- O dată cu deblocarea ușii pare o notificare care să anunțe deblocarea ușii împreună cu o nouă înregistrare în lista de istoric a ușii respective.

d. Rezultate Obținute:

- Circuitul a declanșat buzzerul și motorul la apăsarea butonului.
- După deblocarea ușii a apărut și notificarea de deblocare și o nouă înregistrare în istoric.

e. Observații:

- Circuitul sa comportat exact cum ne-am așteptat.
- Apariția noii înregistrări nu este chiar instantanee ci are o întârziere de câteva secunde.
- În timpul acestu test am apăsă din nou pe butonul de `Unlock`, fără să aștept blocarea ușii. Această acțiune a blocat circuitul făcându-l să nu mai răspundă la nici un fel de interacțiune. Singurul mod de a ieși din această stare a fost să rulez din nou programul pe plăcuță.

8. CONCLUZII

8.1 CONCLUZII PE BAZA REALIZĂRII PROIECTULUI

Proiectul a reușit să îndeplinească toate obiectivele propuse, demonstrând viabilitatea unei încuietori inteligente controlate și monitorizate prin intermediul unei aplicații mobile.

Combinția între componentele hardware (Arduino Nano, motorul servo SG90, senzorul de amprentă) și software (aplicația mobilă dezvoltată în Android Studio) a fost realizată cu succes. Acest lucru subliniază importanța unei abordări care combină cele două părți în dezvoltarea conceptului de IoT.

Implementarea proiectului a întâmpinat diverse provocări, în special în ceea ce privește integrarea componentelor hardware și dezvoltarea unei interfețe de utilizator prietenoase. Aceste provocări au fost depășite prin ajustări interactive și prin aplicarea principiilor de Programare Orientată pe Obiecte.

Sistemul de autentificare biometrică prin senzorul de amprentă și metodele de autentificare în aplicația mobilă au adăugat un nivel important de securitate. Cu toate acestea, există loc pentru îmbunătățiri suplimentare în ceea ce privește protecția împotriva accesului neautorizat și optimizarea procesului de autentificare.

Comunicarea prin Bluetooth a fost eficientă pentru distanțele scurte specifice mediului casnic.

Acest proiect contribuie la dezvoltarea și popularizarea soluțiilor de securitate inteligente pentru locuințe, evidențiind potențialul tehnologiilor moderne de a îmbunătăți calitatea vieții și siguranța utilizatorilor.

Pe parcursul dezvoltării acestui proiect, am învățat importanța planificării riguroase și a imaginației în abordarea problemelor. Aceste lecții sunt valoroase pentru viitoarele proiecte și pentru cariera mea profesională în domeniul ingineriei și tehnologiei informației.

8.2 POSIBILE VIITOARE ÎMBUNĂTĂȚIRI ALE PROIECTULUI

Adăugarea unei funcționalități care permite utilizatorilor să înregistreze noi amprente direct prin intermediul aplicației mobile. Aceasta va facilita gestionarea accesului și va permite adăugarea ușoară a membrilor noi ai familiei sau a altor persoane autorizate.

Implementarea unei camere în sistemul de încuietore inteligente, care să permită deschiderea ușii prin recunoaștere facială. Această funcționalitate va adăuga un nivel suplimentar de securitate și o altă metodă de deschidere a ușii care ar putea fi mai benefică pentru persoanele cu dizabilități.

Introducerea unui cont de administrator, deținut de proprietarul casei sau persoana care achiziționează încuietorea inteligentă. Acest cont va avea drepturi exclusive de aprobare a noilor conturi înregistrate și de eliminare a unor conturi deja existente. Astfel, se va asigura că doar persoanele autorizate de către el pot avea acces la locuința respectivă prin intermediul încuietorii inteligente.

9. BIBLIOGRAFIE

- [1] Reviews.org, *August Wi-Fi Smart Lock Review*,
Available: <https://www.reviews.org/home-security/august-wi-fi-smart-lock-review/>.
- [2] Yale Home, *Yale Assure Digital Lock*,
Available: <https://www.yalehome.com/au/en/products/smart-products/smart-locks/assure-lock-series/yale-assure-digital-lock>.
- [3] Bob Vila, *A Tested Review of the Schlage Encode Smart Wi-Fi Deadbolt*,
Available: <https://www.bobvila.com/articles/schlage-encode-smart-wi-fi-deadbolt-review/>.
- [4] Schlage, *Schlage Encode™ Smart WiFi Deadbolt*,
Available: <https://www.schlage.com/en/home/smart-locks/encode.html>.
- [5] Reviews.org, *Kwikset Kevo Touch-to-Open Smart Lock: Is It Worth It?*,
Available: <https://www.reviews.org/home-security/kwikset-kevo-review/>.
- [6] Kwikset, *Satin Nickel Kevo Traditional Touch-to-Open Smart Lock, 2nd Gen*,
Available: <https://www.kwikset.com/products/detail/satin-nickel-kevo-traditional-touch-to-open-smart-lock-2nd-gen>.
- [7] Instructables, *Modular Arduino-powered Fingerprint Door Lock*,
Available: <https://www.instructables.com/Modular-Arduino-powered-Fingerprint-Door-Lock/>.
- [8] Kunkune, *Arduino nano v3.0 compatible atmega328p*,
Available: <https://kunkune.co.uk/shop/compatible-with-arduino/arduino-nano-v3-0-compatible-atmega328p/>.
- [9] Arduino, *Arduino Projects Hub*,
Available: <https://create.arduino.cc/projecthub>.
- [10] M. Milenkovic, *Internet of Things: Concepts and System Design*. Springer Cham, 2020, ISBN: 978-3-030-41346-0. DOI: 10.1007/978-3-030-41346-0. [Online]. Available: <https://link.springer.com/book/10.1007/978-3-030-41346-0>.
- [11] S. Monk, *Programming Arduino: Getting Started with Sketches*. McGraw-Hill Education, 2013, ISBN: 978-1259641633. [Online]. Available: <https://www.mheducation.com/highered/product/programming-arduino-getting-started-sketches-monk/M9781259641633.html>.
- [12] L. Darcey and S. Conder, *Android Wireless Application Development Volume I: Android Essentials (3rd Edition)*. Addison-Wesley Professional, 2012, ISBN: 978-0321813831. [Online]. Available: <https://www.informit.com/store/android-wireless-application-development-volume-i-android-9780321813831>.

- [13] J. Gosling, B. Joy, G. Steele, G. Bracha, A. Buckley, and D. Smith, *The Java Language Specification: Java SE, 10th Edition*. Addison-Wesley Professional, 2018, ISBN: 978-0134685991. [Online]. Available: <https://docs.oracle.com/javase/specs/jls/se10/html/index.html>.
- [14] R. Ciesla, "Object-oriented programming (oop)", in *Programming Basics*, Apress, Berkeley, CA, 2021, pp. 43–62. DOI: 10.1007/978-1-4842-7286-2_4. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4842-7286-2_4.
- [15] GoCivici, *Door Lock Project*, Available: <https://github.com/gocivici/door-lock>.
- [16] eBay, *Arduino Nano V3.0 Compatible ATmega328P*, Available: <https://shorturl.at/qyWn2>.
- [17] SparkFun Electronics, *Magnetic Door Switch Set*, Available: <https://www.sparkfun.com/products/13247>.
- [18] Cleste.ro, *Butoane Tactile 6x6x5mm*, Available: <https://cleste.ro/butoane-tactile-6x6x5mm.html>, Accessed: 21-06-2024 [Online].
- [19] Cleste.ro, *Senzor de amprentă*, Available: <https://cleste.ro/senzor-amprenta.html>.
- [20] Adafruit, *Adafruit Fingerprint Sensor Library*, Available: <https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library>.
- [21] Optimus Digital, *Buzzer Pasiv de 5V*, Available: <https://www.optimusdigital.ro/ro/audio-buzzere/634-buzzer-pasiv-de-5-v.html>.
- [22] T. Instruments, "Choosing an appropriate pull-up/pull-down resistor for open drain outputs", *Texas Instruments*, Jan. 2020. [Online]. Available: <https://www.ti.com/lit/an/slva485/slva485.pdf>.

DECLARAȚIE DE AUTENTICITATE A LUCRĂRII DE FINALIZARE A STUDIILOR *

Subsemnatul **ION POPESCU**, legitimat cu **CI** seria **TZ** nr. **100772**, CNP **5000102355779**, autorul lucrării **TITLUL LUCRĂRII DE FINALIZARE A STUDIILOR** elaborată în vederea susținerii examenului de finalizare a studiilor de **LICENȚĂ** organizat de către Facultatea **AUTOMATICĂ ȘI CALCULATOARE** din cadrul Universității Politehnica Timișoara, sesiunea **IUNIE** a anului universitar **2022**, coordonator **dr.ing. MIHAI IONESCU**, luând în considerare conținutul art. 34 din *Regulamentul privind organizarea și desfășurarea examenelor de licență/diplomă și disertație*, aprobat prin HS nr. 109/14.05.2020 și cunoscând faptul că în cazul constatării ulterioare a unor declarații false, voi suporta sancțiunea administrativă prevăzută de art. 146 din Legea nr. 1/2011 – legea educației naționale și anume anularea diplomei de studii, declar pe proprie răspundere, că:

- această lucrare este rezultatul propriei activități intelectuale,
- lucrarea nu conține texte, date sau elemente de grafică din alte lucrări sau din alte surse fără ca acestea să nu fie citate, inclusiv situația în care sursa o reprezintă o altă lucrare/alte lucrări ale subsemnatului.
- sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.
- această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență/diplomă/disertație.

Timișoara,

Data

Semnătura

*Declarația se completează „de mână” și se inserează în lucrarea de finalizare a studiilor, la sfârșitul acesteia, ca parte integrantă.