MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

AUTOMATIZAREA ȘI CONTROLUL INTELIGENT AL JALUZELELOR

PROIECT DE DIPLOMĂ

Autor: **Claudiu-Ionuț BĂRĂIAN**

Conducător științific: **SL.dr.ing. Ioan Valentin SITA**

2024

MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Vizat,

DECAN
Prof. dr. ing. Mihaela DÎNȘOREANU

DIRECTOR DEPARTAMENT AUTOMATICĂ
Prof. dr. ing. Honoriu VĂLEAN

Autor: **Claudiu-Ionuț BĂRĂIAN**

Automatizarea și controlul inteligent al jaluzelelor

1. **Enunțul temei:** *Proiectul vizează dezvoltarea unui sistem automatizat de control al jaluzelelor, utilizând plăci NodeMCU, motor pas cu pas, senzor de lumină și matrice LED, controlate printr-o aplicație Android.*

2. **Conținutul proiectului:** *Pagina de prezentare, Declarație privind autenticitatea proiectului, Sinteza proiectului, Cuprins, Introducere, Studiu bibliografic, Analiză, proiectare, implementare, Bibliografie, Anexe.*

3. **Locul documentării:** *Universitatea Tehnică din Cluj-Napoca, alte locuri dacă este cazul*

4. **Consultanți:** *SL.dr.ing. Ioan Valentin SITA*

5. **Data emiterii temei:** 03.11.2023

6. **Data predării:** 11.07.2024

Semnătura autorului _____

Semnătura conducătorului științific _____

MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Declaraţie pe proprie răspundere privind

autenticitatea proiectului de diplomă

Subsemnatul(a) **Claudiu-Ionuț Bărăian**,
legitimat(ă) cu CI seria CJ nr. 536818 , CNP 5020107125779,
autorul lucrării:

AUTOMATIZAREA SI CONTROLUL INTELIGENT AL JALUZELELOR

elaborată în vederea susţinerii examenului de finalizare a studiilor de licenţă la **Facultatea de Automatică şi Calculatoare**, specializarea **Automatică şi Informatică Aplicată,** din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea iulie 2024 a anului universitar 2023-2024, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

În cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data
11.07.2024

Prenume NUME
Bărăian
(semnătura)

MINISTERUL EDUCAȚIEI
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

SINTEZA

proiectului de diplomă cu titlul:

Automatizarea şi controlul inteligent al jaluzelelor

Autor: **Claudiu-Ionuţ BĂRĂIAN**

Conducător ştiinţific: **Titlu. ing. Claudiu-Ionuţ BĂRĂIAN**

1. Cerinţele temei: Dezvoltarea unui sistem automatizat de control al jaluzelelor, care să ofere multiple moduri de funcţionare adecvate utilizatorului şi să poată menţine o luminozitate constantă în încăpere, indiferent de perturbările exterioare, favorizând utilizarea luminii naturale.

2. Soluţii alese: Utilizarea plăcilor NodeMCU pentru conectivitate şi control, motor pas cu pas 28BYJ-48, driver ULN2003, matrice LED MAX7219 şi senzor de lumină DFR0026, integrându-le într-o aplicaţie Android dezvoltată cu Kotlin şi Jetpack Compose.

3. Rezultate obţinute: Control precis al jaluzelelor, ajustarea automată a luminii în funcţie de condiţiile ambientale, interfaţă utilizator modernă şi intuitivă, economii de energie prin utilizarea optimizată a luminii naturale şi artificiale.

4. Testări şi verificări: Testarea individuală şi integrată a componentelor hardware şi software, verificarea conectivităţii şi funcţionării corecte a sistemului, ajustarea performanţei pentru a asigura o operare conformă cu cerinţele temei.

5. Contribuţii personale: Dezvoltarea şi integrarea modulelor hardware şi software, proiectarea şi implementarea interfeţei Android, optimizarea sistemului pentru performanţă şi eficienţă, testarea şi validarea funcţionalităţilor implementate.

6. Surse de documentare: Literatura de specialitate privind automatizarea locuinţelor, documentaţia tehnică a componentelor hardware utilizate, tutoriale şi resurse online pentru NodeMCU şi Kotlin.

Semnătura autorului _____

Semnătura conducătorului ştiinţific _____

Cuprins

1 Introducere

1.1 Context general

Încă din cele mai vechi timpuri, omul a căutat să-şi îmbunătăţească nivelul de confort în mediul în care trăieşte. Locuinţa reprezintă pentru fiecare individ un spaţiu esenţial, unde îşi petrece majoritatea timpului. Această dorinţă de a crea un ambient confortabil şi sigur în propria casă a condus la dezvoltarea continuă a tehnologiilor de automatizare.

În prezent, progresele tehnologice permit implementarea sistemelor inteligente în locuinţe, sporind confortul şi eficienţa energetică. Automatizarea jaluzelelor, de exemplu, permite reglarea automată a cantităţii de lumină naturală din încăpere, contribuind astfel la crearea unui mediu de viaţă optim şi la reducerea consumului de energie.

Pe lângă confortul oferit, eficienţa energetică a locuinţei a devenit un aspect prioritar. În contextul actual, caracterizat de o preocupare în creştere pentru sustenabilitate şi reducerea consumului de energie, sistemele automate de control al jaluzelelor reprezintă o soluţie eficientă pentru îmbunătăţirea acestui aspect. Implementarea unui sistem automatizat de jaluzele nu doar că asigură un nivel optim de lumină naturală, dar contribuie şi la reducerea consumului de energie electrică necesară iluminatului. Astfel de sisteme inteligente devin din ce în ce mai accesibile şi populare, oferind oportunităţi considerabile pentru îmbunătăţirea calităţii vieţii şi protecţia mediului înconjurător.

În această lucrare, am dezvoltat un sistem automatizat pentru jaluzele, folosind un motor pas cu pas controlat de o placă NodeMCU, echipată cu chip Wi-Fi ESP8266. Sistemul include, de asemenea, un senzor de lumină conectat la o altă placă NodeMCU şi un LED ataşat la o a treia placă NodeMCU. Controlul se realizează printr-o aplicaţie Android, care permite utilizatorilor să comute între modurile automat, manual şi de noapte.

Modul automat permite setarea intensităţii luminii dorite în cameră prin intermediul unui glisor, iar sistemul va ajusta jaluzeaua pentru a atinge valoarea dorită. Deoarece jaluzeaua nu se deschide foarte rapid, LED-ul va suplimenta lumina până când jaluzeaua ajunge la poziţia necesară, favorizând astfel lumina naturală. Modul manual permite utilizatorului să seteze procentajul de deschidere atât pentru jaluzea, cât şi pentru LED. În modul de noapte, jaluzeaua şi LED-ul se închid complet.

Placa conectată la LED are rolul de server HTTP principal, primind comenzi de la aplicaţia Android şi informaţii de la placa NodeMCU conectată la senzorul de lumină şi de la cea conectată la motor. Placa conectată la motor funcţionează, de asemenea, ca server HTTP, primind comenzi de la placa server principal. Acest aranjament permite un control precis şi eficient al luminii naturale şi artificiale în locuinţă, îmbunătăţind astfel confortul şi eficienţa energetică.

1.2 Obiective

Obiectivul principal al acestei lucrări este realizarea unui sistem automatizat de control al jaluzelelor, care să ofere utilizatorilor un control precis şi eficient al luminii naturale în locuinţă. Pentru realizarea acestui obiectiv, am urmărit câteva aspecte esenţiale:

- Aspecte teoretice privind funcţionarea motorului pas cu pas şi conectarea acestuia cu placa NodeMCU: A fost necesară o înţelegere a principiilor de funcţionare ale motorului pas cu pas 28BYJ-48 şi a modului în care acesta poate fi controlat prin intermediul plăcii NodeMCU. Aceasta a inclus configurarea şi programarea motorului pentru a asigura mişcări precise ale jaluzelei.
- Analizarea interacţiunii dintre controlerul principal şi celelalte module (senzor de lumină şi motorul pas cu pas): Sistemul presupune o coordonare eficientă între placa server principal, senzorul de lumină şi motorul pas cu pas. A fost esenţială sincronizarea între toate modulele, pentru o bună funcţionare a sistemului.
- Evaluarea şi operarea sistemului automatizat în modurile automat, manual şi de noapte: Fiecare mod de operare are specificităţile sale şi a necesitat o proiectare atentă pentru a asigura funcţionarea corectă a sistemului. Modul automat ajustează jaluzeaua şi LED-ul pentru a menţine nivelul dorit de lumină. Modul manual permite controlul direct al utilizatorului asupra poziţiei jaluzelei şi intensităţii LED-ului, iar modul de noapte asigură închiderea completă a jaluzelei şi stingerea LED-ului pentru a crea un mediu optim pentru somn.
- Modul în care comenzile sunt transmise şi executate între aplicaţia Android şi sistemul hardware: Proiectarea unui sistem de control fiabil implică dezvoltarea unei aplicaţii Android care să poată trimite comenzi precise către plăcile NodeMCU. Aceasta include asigurarea unei comunicări stabile şi rapide prin intermediul reţelei Wi-Fi.
- Proiectarea şi implementarea unui sistem robust şi eficient, utilizând plăci de dezvoltare NodeMCU şi resurse software adecvate: Alegerea plăcilor NodeMCU şi a resurselor software necesare (biblioteci de control pentru motorul pas cu pas, senzori şi LED-uri) a fost esenţială pentru dezvoltarea unui sistem eficient şi fiabil. Implementarea a presupus atât programarea microcontrollerelor cât şi dezvoltarea aplicaţiei Android pentru a asigura o interfaţă de utilizare intuitivă.

În scopul realizării acestui obiectiv, s-a construit un sistem automatizat cu ajutorul a 3 plăci de dezvoltare NodeMCU, un motor pas cu pas 28BYJ-48, senzor de lumină DFRobot şi o matrice LED MAX7219, asigurându-se astfel o interfaţă de control eficientă şi intuitivă prin intermediul unei aplicaţii Android. Acest sistem permite utilizatorilor să beneficieze de un control optim al luminii naturale şi artificiale, contribuind la crearea unui mediu confortabil şi eficient energetic în locuinţă.

1.3 Specificații

Lucrarea de față descrie procesul de construire a unui sistem automatizat pentru controlul jaluzelelor, care asigură un nivel optim de confort și eficiență energetică în locuință. Sistemul este format din următoarele componente principale:

- Motor pas cu pas 28BYJ-48: Utilizat pentru controlul precis al poziției jaluzelei.
- Plăci NodeMCU: Trei plăci NodeMCU, fiecare având un rol specific (control motor, senzor de lumină, control LED).
- Senzor de lumină DFRobot: Măsoară intensitatea luminii din cameră și trimite datele către placa server.
- Matrice LED MAX7219: Suplimentează lumina până când jaluzeaua ajunge la poziția necesară sau atunci când lumina naturală nu este suficientă.

Funcționalități principale:

- Modul automat: Utilizatorul poate seta intensitatea luminii dorite în cameră prin intermediul unui glisor în aplicația Android. Sistemul ajustează jaluzeaua pentru a atinge valoarea dorită, iar LED-ul suplimentează lumina până când jaluzeaua ajunge la poziția necesară.
- Modul manual: Utilizatorul poate seta manual procentajul de deschidere al jaluzelei și intensitatea LED-ului prin aplicația Android.
- Modul de noapte: Jaluzeaua și LED-ul se închid complet pentru a asigura un mediu întunecat pentru somn.

Arhitectura sistemului:

- Placa server principal (NodeMCU): Controlează intensitatea LED-ului și gestionează request-urile HTTP.
- Senzor de lumină (NodeMCU): Trimite date legate de intensitatea luminii către placa server principal.
- Control motor (NodeMCU): Controlează motorul pas cu pas conform comenzilor primite de la placa server principal și trimite înapoi către aceasta poziția actuală a jaluzelei.
- Aplicație Android: Interfața principală de control pentru utilizator, care permite comutarea între moduri, setarea intensității luminii și ajustarea poziției jaluzelei și a LED-ului.

Acest sistem demonstrează beneficiile utilizării tehnologiilor de automatizare în crearea unui mediu de locuit confortabil și eficient din punct de vedere energetic. În plus, utilizarea unei aplicații mobile pentru controlul sistemului asigură o interfață user-friendly, facilitând astfel utilizarea și configurarea sistemului de către orice utilizator.

2 Studiu bibliografic

2.1 Automatizarea clădirilor

Automatizarea clădirilor, cunoscută şi sub denumirea de Building Automation Systems (BAS), a evoluat semnificativ odată cu progresul tehnologic. BAS sunt implementate pentru a controla şi monitoriza diverse sisteme ale clădirilor, cum ar fi iluminatul, climatizarea (HVAC), securitatea şi alte funcţii esenţiale. Aceste sisteme sunt esenţiale pentru îmbunătăţirea eficienţei energetice, confortului şi siguranţei în clădiri. Studii recente arată că utilizarea BAS poate reduce semnificativ consumul de energie şi emisiile de carbon, contribuind la obiectivele de sustenabilitate globală. [1]

Am ales să automatizez jaluzelele unei camere utilizând un motor pas cu pas, senzor de lumină şi plăci NodeMCU pentru a crea un sistem eficient şi integrat de control al iluminatului natural. Această soluţie contribuie direct la îmbunătăţirea eficienţei energetice, reducând necesitatea iluminatului artificial şi optimizând utilizarea luminii naturale. În plus, utilizarea unui motor pas cu pas asigură precizia necesară pentru ajustarea fină a poziţiei jaluzelelor, iar plăcile NodeMCU permit o integrare uşoară şi eficientă din punct de vedere al costului a componentelor IoT.

2.2 Senzori de lumină

Senzorii de lumină sunt esenţiali în diverse aplicaţii de automatizare, inclusiv în sistemele de iluminat automat, dispozitivele mobile şi în gestionarea resurselor energetice. Tehnologiile actuale pentru senzori de lumină includ fotodiode, fototranzistoare şi senzori bazate pe tehnologia CMOS, care oferă măsurători precise ale intensităţii luminii ambientale. Dezvoltările recente în domeniul senzorilor de lumină au condus la îmbunătăţirea sensibilităţii şi a intervalului dinamic, permiţând integrarea lor eficientă în sistemele de automatizare a locuinţelor. [2]

Integrarea unui senzor de lumină în proiectul de automatizare a jaluzelelor permite ajustarea automată a poziţiei jaluzelelor pentru a menţine nivelul dorit de iluminare naturală în încăpere. Aceasta nu doar îmbunătăţeşte confortul, dar şi contribuie la eficienţa energetică prin reducerea necesităţii iluminatului artificial. Utilizarea unui senzor de lumină de înaltă precizie asigură că sistemul poate reacţiona rapid şi precis la schimbările în intensitatea luminii ambientale, oferind astfel o soluţie eficientă şi performantă.

2.3 Motoare pas cu pas

Motoarele pas cu pas sunt utilizate pe scară largă în aplicaţiile care necesită control precis al mişcării, cum ar fi imprimantele 3D, echipamentele CNC şi sistemele de automatizare. Aceste motoare funcţionează prin deplasarea în paşi discreţi, permiţând un control precis al poziţiei şi vitezei. Tehnologiile actuale în controlul motoarelor pas cu pas includ drivere avansate care permit micro-pasificarea, reducerea vibraţiilor şi îmbunătăţirea performanţelor dinamice. [3]

Am ales să folosesc un motor pas cu pas 28BYJ-48 5V pentru automatizarea jaluzelelor datorită preciziei şi fiabilităţii sale. Capacitatea motorului de a efectua mişcări precise în paşi mici este esenţială pentru ajustarea fină a poziţiei jaluzelelor, asigurând astfel că nivelul de lumină din încăpere poate fi controlat exact aşa cum doreşte utilizatorul. De asemenea, motorul pas cu pas este compatibil cu plăcile NodeMCU şi poate fi controlat eficient prin intermediul acestora, ceea ce simplifică implementarea şi reduce costurile.

Pentru controlul motorului pas cu pas, am utilizat driverul ULN2003. Acesta este un driver popular pentru motoarele pas cu pas datorită capacităţii sale de a gestiona curenţi mai mari şi de a oferi protecţie la suprasarcină. Driverul ULN2003 permite conectarea facilă a motorului 28BYJ-48 la placa NodeMCU, asigurând astfel un control precis şi fiabil al motorului. Utilizarea acestui driver îmbunătăţeşte performanţa generală a sistemului şi contribuie la stabilitatea şi durabilitatea acestuia.

2.4 Plăci de dezvoltare, NodeMCU

NodeMCU este o platformă de dezvoltare open-source bazată pe modulul ESP8266, care oferă capacităţi Wi-Fi şi de procesare pentru aplicaţii IoT. NodeMCU este cunoscută pentru costul său redus şi flexibilitatea în integrarea cu diverse senzori şi actuatori. Platforma suportă programarea în Lua şi Arduino IDE, oferind o curba de învăţare uşoară şi o comunitate vastă de dezvoltatori care contribuie cu resurse şi biblioteci utile. [4]

Am ales plăcile NodeMCU pentru acest proiect datorită costului redus, flexibilităţii şi capacităţilor integrate de reţea Wi-Fi. NodeMCU permite o comunicare eficientă între componentele sistemului şi aplicaţia Android, facilitând transmiterea datelor şi controlul în timp real al jaluzelelor şi LED-ului. În plus, compatibilitatea cu Arduino IDE simplifică procesul de programare şi implementare a firmware-ului necesar pentru funcţionarea sistemului.

2.5 Limbaje de programare folosite

2.5.1 C++

Limbajul specific Arduino, bazat pe C++, este larg utilizat pentru programarea plăcilor de dezvoltare microcontroller, inclusiv NodeMCU. Acesta permite scrierea de cod simplu şi eficient pentru controlul hardware-ului, fiind extrem de popular în comunitatea DIY şi în proiectele de automatizare. Bibliotecile disponibile pentru Arduino simplifică

interacţiunea cu diverse componente, cum ar fi motoarele pas cu pas, senzori şi module de comunicare wireless. [5]

Utilizarea limbajului Arduino şi C++ pentru programarea plăcilor NodeMCU este justificată de necesitatea unui control precis şi robust al motorului pas cu pas şi de integrarea cu alte componente, cum ar fi senzorul de lumină şi matricea LED. Limbajul Arduino oferă o sintaxă uşor de învăţat şi utilizat, care permite dezvoltarea rapidă şi eficientă a codului pentru controlul hardware-ului.

Posibile soluţii pentru implementarea acestei abordări includ utilizarea următoarelor biblioteci dedicate:

- Stepper.h pentru controlul motorului pas cu pas: Permite controlul precis al mişcărilor motorului, esenţial pentru ajustarea poziţiei jaluzelei. [6]
- ESP8266WiFi.h pentru funcţionalităţi de reţea: Facilitează conectarea plăcii NodeMCU la reţeaua Wi-Fi, permiţând comunicarea cu alte dispozitive. [7]
- ESP8266WebServer.h pentru crearea serverului web: Asigură implementarea unui server HTTP pe placa NodeMCU, gestionând cererile HTTP de la aplicaţia Android. [8]
- ESP8266HTTPClient.h pentru cereri HTTP: Permite trimiterea şi primirea cererilor HTTP către şi de la alte dispozitive în reţea. [9]
- LedControl.h pentru controlul matricei LED: Oferă funcţii pentru controlul precis al matricei LED MAX7219, utilizată pentru suplimentarea luminii. [10]

Aceste biblioteci simplifică semnificativ procesul de dezvoltare, oferind funcţii predefinite care reduc complexitatea codului şi îmbunătăţesc stabilitatea şi performanţa sistemului.

2.5.2 Kotlin si Jetpack Compose

Alegerea limbajului Kotlin pentru dezvoltarea aplicaţiei Android este motivată de avantajele sale faţă de Java, inclusiv o sintaxă mai clară şi un sistem de tipuri mai sigur. Kotlin facilitează dezvoltarea de aplicaţii robuste şi uşor de întreţinut, esenţiale pentru controlul unui sistem automatizat de jaluzele. [11]

Jetpack Compose a fost ales pentru dezvoltarea interfeţei de utilizator datorită abordării sale moderne şi eficiente. Componentele UI declarative permit crearea de interfeţe responsive şi intuitive, simplificând codul şi reducând timpul de dezvoltare. În aplicaţia prezentată, componentele Jetpack Compose, cum ar fi “Column”, "Row”, ”Box”, ”Text”, ” Slider”, ” IconButton”, şi ” Surface”, sunt utilizate pentru a construi UI-ul. Aceste componente permit dezvoltarea unei interfeţe elegante şi funcţionale. Utilizarea LaunchedEffect pentru gestionarea operaţiunilor asincrone şi integrarea OkHttpClient pentru comunicarea cu plăcile NodeMCU demonstrează flexibilitatea şi puterea Kotlin şi Jetpack Compose în dezvoltarea aplicaţiilor mobile.

3 Analiză, proiectare, implementare

3.1 Analiza şi proiectarea unui sistem automatizat de control inteligent al jaluzelelor

Pentru a proiecta un sistem automatizat şi inteligent de control al jaluzelelor, este esenţial să analizăm şi să alegem cu grijă fiecare componentă hardware şi software, ţinând cont de nevoile sistemului.

Pentru sistemul de ajustare precisă a jaluzelelor, componenta hardware crucială este motorul. Am optat pentru motorul pas cu pas 28BYJ-48 datorită preciziei şi fiabilităţii sale bine cunoscute, asigurând astfel capacitatea de ajustare fină necesară pentru funcţionarea optimă a sistemului.

Pentru a asigura o conectivitate eficientă şi gestionarea adecvată a sistemului, am decis să plasăm controlerul principal într-o locaţie centrală. Am ales NodeMCU, echipată cu o matrice LED MAX7219, pentru capacitatea sa de a gestiona cererile HTTP şi pentru controlul precis al LED-urilor, asigurând astfel funcţionarea fluentă şi integrarea optimă a tuturor componentelor sistemului.

Pentru a asigura o funcţionare precisă a sistemului, am dispus senzorul de lumină într-o locaţie strategică pentru a măsura cu exactitate intensitatea luminii ambientale. Motorul pas cu pas a fost montat într-o poziţie optimă pentru a permite mişcarea eficientă, dar şi a minimiza dimensiunile ansamblului, în funcţie de datele obţinute de senzorul de lumină, asigurând confortul şi eficienţa energetică dorite.

Pentru a implementa proiectul utilizând NodeMCU şi Arduino IDE, am integrat biblioteci esenţiale pentru gestionarea conectivităţii şi a funcţionalităţilor hardware. Am folosit ESP8266WiFi.h pentru conectarea la reţele Wi-Fi, ESP8266WebServer.h pentru crearea unui server web, şi ESP8266HTTPClient.h pentru efectuarea cererilor HTTP, asigurând astfel comunicarea eficientă şi gestionarea datelor în sistem. În plus, am inclus LedControl.h pentru controlul matricei LED MAX7219 şi stepper.h pentru gestionarea precisă a motorului pas cu pas 28BYJ-48, completând astfel setul de instrumente necesare pentru o automatizare robustă şi funcţionarea optimă a sistemului propus.

Pentru a dezvolta o interfaţă de utilizator intuitivă şi versatilă, am optat să dezvolt o aplicaţie Android folosind Kotlin şi Jetpack Compose în Android Studio. Această combinaţie permite crearea rapidă şi eficientă a unei interfeţe moderne, cu posibilităţi avansate de personalizare şi animaţie, esenţiale pentru o experienţă utilizator fluidă şi atractivă.

Android Studio facilitează dezvoltarea aplicaţiilor Android prin oferirea unui mediu de lucru integrat, care include un simulator de telefon Android pentru testare în timp real. Acest lucru simplifică procesul de dezvoltare, permitând programatorilor să îşi concentreze eforturile pe implementarea funcţionalităţilor şi optimizarea experienţei utilizatorului, fără a fi nevoie de dispozitive fizice în fiecare etapă a procesului de dezvoltare.

Pentru a facilita comunicarea eficientă între modulele NodeMCU şi aplicaţia Android, precum şi între diferitele plăci NodeMCU, am optat pentru utilizarea unei arhitecturi server-client HTTP. Aceasta permite trimiterea şi primirea cererilor HTTP într-un mod rapid şi simplu, asigurând actualizarea în timp real a stării sistemului. Această metodă oferă o soluţie robustă pentru integrarea şi interacţiunea fluidă între componentele sistemului propus.

Prin alegerea atentă a componentelor potrivite pentru integrarea în sistemul propus, am asigurat o funcţionare eficientă şi fiabilă a întregului sistem. Aceste decizii au contribuit semnificativ la facilitarea comunicării rapide şi actualizării în timp real a stării sistemului între modulele NodeMCU şi aplicaţia Android, consolidând astfel performanţa şi utilizarea intuitivă a soluţiei.

3.2 Analiza, proiectarea si implementarea sistemului hardware

3.2.1 Analiza sistemului hardware

Schema bloc prezentată prezentată in Figura 3.1 include trei plăci de dezvoltare NodeMCU, diverse componente conectate, cum ar fi un motor pas cu pas, un LED şi un senzor de lumină, şi interacţiunea dintre acestea. Sistemul este structurat astfel încât să permită controlul precis al poziţiei jaluzelelor şi ajustarea luminii în funcţie de condiţiile de iluminare existente.

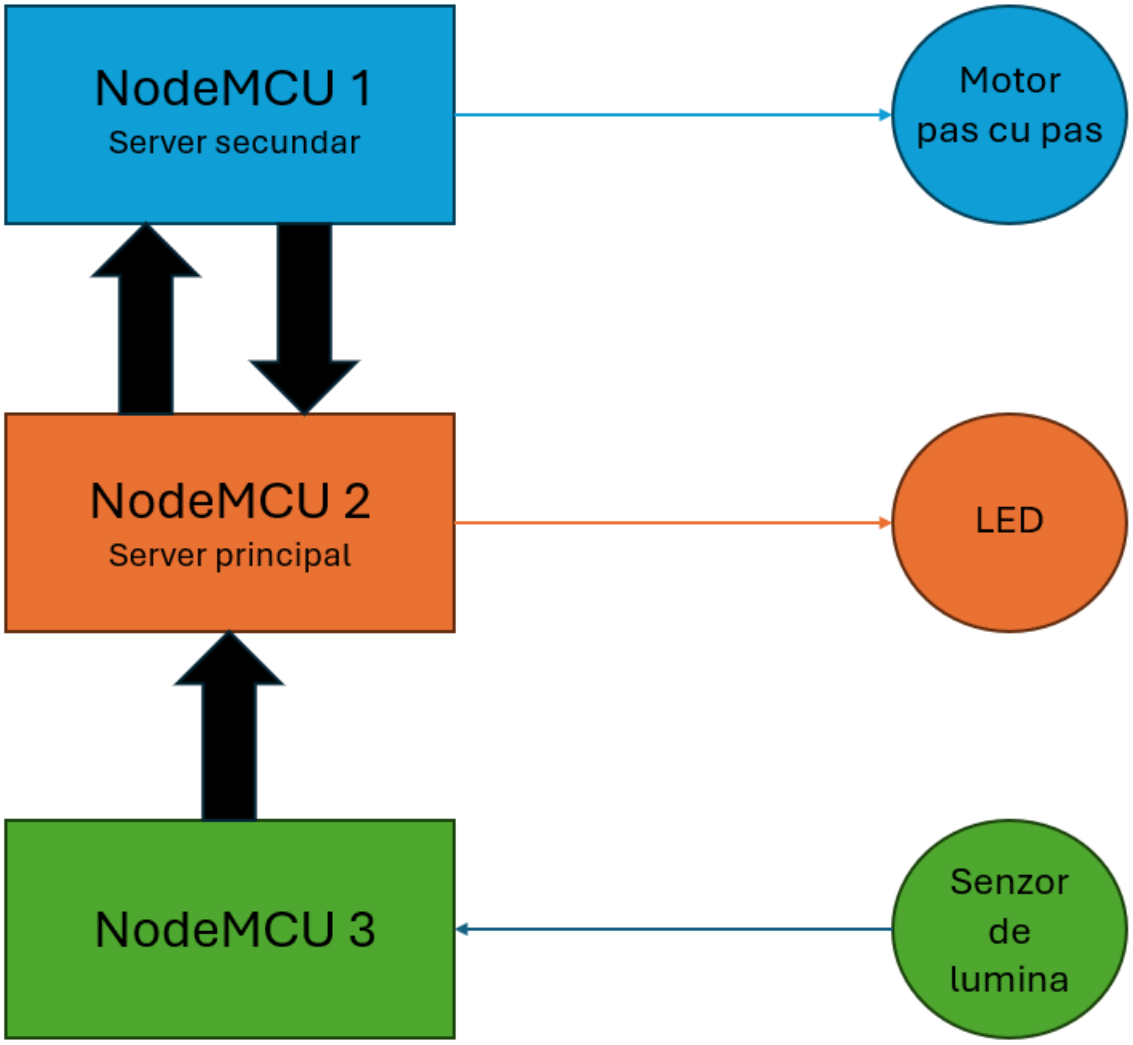


Figura 3.1: Schema bloc a sitemului hardware

NodeMCU 1 funcţionează ca server secundar în sistemul de control automatizat al jaluzelelor. Acesta are rolul principal de a controla motorul pas cu pas, care ajustează precis poziţia jaluzelei pentru a regla lumina naturală ce pătrunde în încăpere. NodeMCU 1 comunică constant cu serverul principal, NodeMCU 2, pentru a primi comenzi specifice de ajustare a jaluzelei şi pentru a raporta starea curentă a acesteia. În plus, NodeMCU 1 monitorizează performanţa motorului pas cu pas, asigurându-se că mişcările sunt precise şi conform specificaţiilor primite. Comunicarea bidirecţională între NodeMCU 1 şi NodeMCU 2 este esenţială pentru actualizări în timp real şi pentru a garanta că poziţia jaluzelei este corectă în funcţie de necesităţile de iluminare. Motorul pas cu pas, componenta esenţială conectată la NodeMCU 1, permite o mişcare precisă şi controlată a jaluzelei, ajustând-o pentru a maximiza confortul şi eficienţa energetică. Această interacţiune coordonată între NodeMCU 1 şi NodeMCU 2 garantează o funcţionare eficientă şi sincronizată a sistemului de automatizare a jaluzelelor, optimizând lumina naturală care intră în încăpere.

NodeMCU 2 funcţionează ca server principal în sistemul de control automatizat. Acesta are rolul de a gestiona comunicarea între toate modulele şi de a coordona funcţionarea generală a sistemului. NodeMCU 2 este responsabil pentru procesarea datelor primite de la alte module şi pentru trimiterea comenzilor necesare pentru ajustări. Acesta controlează intensitatea LED-ului pentru a suplimenta lumina naturală atunci când este necesar, asigurând o iluminare optimă în toate condiţiile. NodeMCU 2 primeşte date de la senzorul de lumină conectat la NodeMCU 3 şi le analizează pentru a decide asupra ajustărilor necesare ale poziţiei jaluzelei şi intensităţii LED-ului. De asemenea, trimite comenzi către motorul pas cu pas controlat de NodeMCU 1 şi către LED. LED-ul, componenta conectată direct la NodeMCU 2, joacă un rol esenţial în ajustarea luminii în funcţie de necesităţi, asigurând un mediu iluminat corespunzător. Coordonarea şi procesarea eficientă a datelor de către NodeMCU 2 sunt esenţiale pentru funcţionarea armonioasă a întregului sistem de automatizare a iluminatului şi jaluzelelor.

NodeMCU 3 funcţionează ca modul de măsurare a intensităţii luminii din cameră. Acesta este responsabil pentru colectarea datelor de la senzorul de lumină conectat şi pentru trimiterea acestor informaţii către serverul principal, NodeMCU 2. Comunicarea între NodeMCU 3 şi serverul principal este esenţială pentru a raporta în timp real datele de iluminare, permiţând ajustări rapide şi precise ale sistemului. NodeMCU 3 monitorizează continuu condiţiile de iluminare ambientală, asigurându-se că datele sunt precise şi actualizate constant. Aceste informaţii permit sistemului să ajusteze automat poziţia jaluzelelor şi intensitatea LED-ului, asigurând astfel o iluminare optimă în funcţie de condiţiile actuale. Senzorul de lumină, componenta conectată direct la NodeMCU 3, joacă un rol crucial în monitorizarea condiţiilor de iluminare ambientală. Datele precise şi actualizate furnizate de NodeMCU 3 sunt esenţiale pentru funcţionarea eficientă şi adaptabilă a sistemului de automatizare a iluminatului şi jaluzelelor, contribuind la un mediu confortabil şi eficient din punct de vedere energetic.

Sistemul de control al jaluzelelor şi iluminării cu LED-uri dispune de trei moduri de funcţionare: automat, manual şi noapte, fiecare având caracteristici şi comportamente specifice pentru a răspunde diverselor nevoi ale utilizatorului.

Modul automat:

În modul automat, utilizatorul poate seta intensitatea dorită a luminii în cameră prin intermediul unui glisor. Sistemul ajustează automat poziţia jaluzelei pentru a atinge nivelul de lumină dorit, favorizând utilizarea luminii naturale. Deoarece jaluzeaua nu se deschide foarte rapid, LED-ul suplimentează temporar lumina până când jaluzeaua ajunge la poziţia necesară. NodeMCU 2, care funcţionează ca server principal, trimite către NodeMCU 1 comenzi de tipul "open", "close" şi "halt" pentru controlul motorului pas cu pas. În acest mod, NodeMCU 1 raportează în timp real poziţia jaluzelei către NodeMCU 2, care este responsabil pentru comanda de oprire a motorului odată ce poziţia dorită este atinsă. Această comunicare constantă asigură ajustări precise şi eficiente ale jaluzelei pentru a menţine nivelul de lumină setat.

Modul manual:

În modul manual, utilizatorul are control direct asupra procentajului de deschidere a jaluzelei şi asupra intensităţii LED-ului. NodeMCU 2 trimite către NodeMCU 1

procentajul la care trebuie să ajungă jaluzeaua, dar în acest mod, NodeMCU 1 nu raportează în timp real poziția jaluzelei înapoi către NodeMCU 2. Acest mod permite utilizatorului să ajusteze setările de lumină și poziția jaluzelei conform preferințelor personale fără intervenția automată a sistemului. Este util atunci când utilizatorul dorește un control direct și specific asupra mediului lor imediat.

Modul de noapte:

În modul de noapte, sistemul este setat pentru a asigura un mediu întunecat, potrivit pentru odihnă. Jaluzeaua și LED-ul sunt închise complet. Similar cu modul manual, NodeMCU 2 trimite la NodeMCU 1 procentajul la care trebuie să ajungă jaluzeaua, în acest caz 0%. NodeMCU 1 nu raportează în timp real poziția jaluzelei către NodeMCU 2, deoarece scopul este de a închide complet jaluzeaua fără necesitatea monitorizării continue. Acest mod simplifică operațiunile sistemului pentru a crea un mediu propice somnului, fără lumini care să perturbe odihna utilizatorului.

Aceste moduri de funcționare oferă flexibilitate și adaptabilitate, permițând utilizatorului să controleze mediul luminos al camerei în funcție de preferințe și necesități specifice. Modul automat maximizează eficiența energetică și confortul, modul manual oferă control personalizat, iar modul de noapte asigură un mediu optim pentru somn.

3.2.2 Proiectarea sistemului hardware

Documentarea proiectării sistemului hardware pentru automatizarea jaluzelei cu rola va include descrierea componentelor utilizate, conexiunile hardware, și explicațiile privind funcționarea și utilizarea fiecărei componente în cadrul sistemului.

Figura 3.2 ilustrează privirea de ansamblu a schemei electrice, evidențiind conexiunile necesare pentru fiecare componentă a sistemului. Se poate observa clar respectarea diagramei bloc prezentată anterior, asigurând astfel coerența și funcționalitatea întregului sistem.

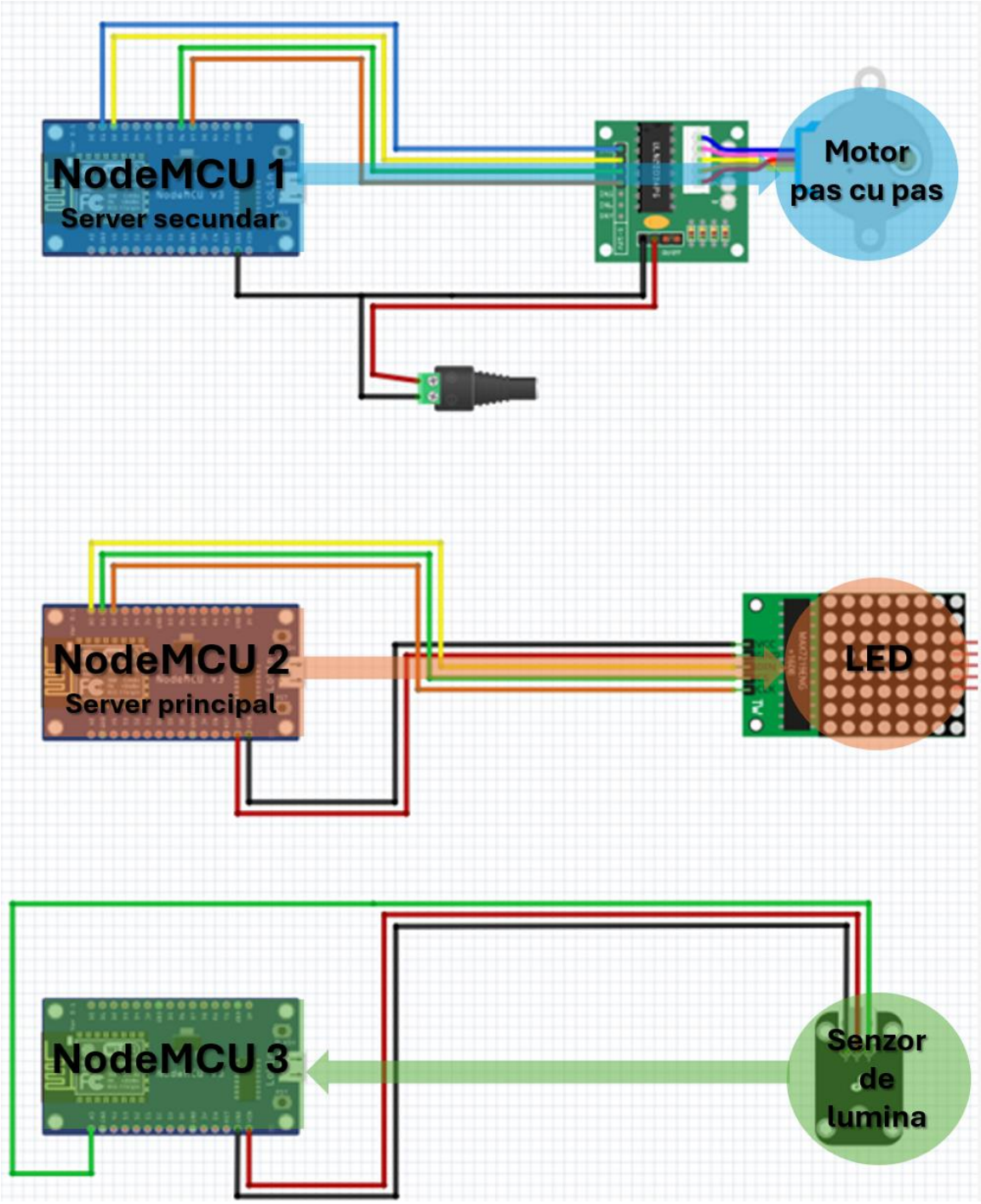


Figura 3.2: Schema electrică a sistemului hardware

3.2.2.1 Ansamblul plăcii NodeMCU și a motorului pas cu pas

Figura 3.2 prezintă un sistem hardware simplu, dar eficient, utilizat pentru automatizarea jaluzelelor prin controlul unui motor pas cu pas. Sistemul include o plăcuță NodeMCU, un driver ULN2003 și un motor pas cu pas 28BYJ-48. Aceasta lucrare explorează conectivitatea și funcționalitatea fiecărei componente, explicând modul în

care acestea interacţionează pentru a realiza o soluţie integrată de automatizare.

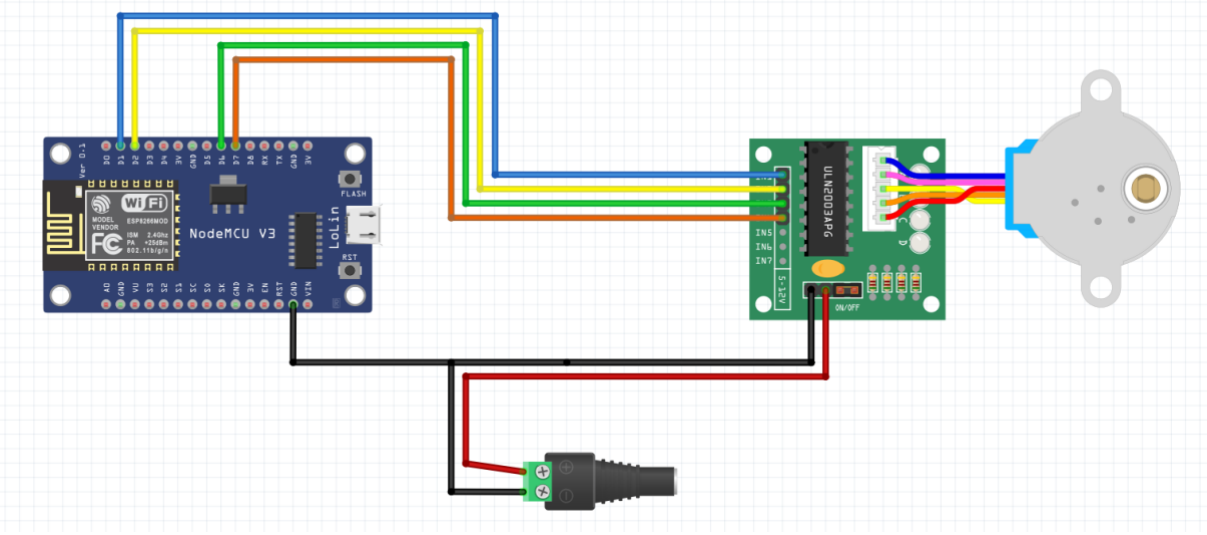


Figura 3.3: Schema electrică a ansamblului plăcii NodeMCU,

motorului pas cu pas 28BYJ-48, şi a driverului ULN2003

Ansamblul descris pentru acţionarea jaluzelelor este construit în jurul unor componente esenţiale care asigură un control precis şi eficient al poziţiei acestora. Componentele principale includ placa NodeMCU, driverul ULN2003 şi motorul pas cu pas 28BYJ-48, integrând tehnologia IoT (Internet of Things) pentru gestionarea şi monitorizarea sistemului prin intermediul unei interfeţe mobile sau web.

Placa NodeMCU, bazată pe chipul ESP8266, este centrală în acest sistem datorită conectivităţii sale Wi-Fi. Acest microcontroller permite comunicarea cu alte dispozitive prin reţea şi este dotat cu pini GPIO (General Purpose Input/Output) care facilitează conexiunea cu diverse module şi senzori, esenţiale pentru adaptabilitatea şi funcţionalitatea într-un sistem IoT (Internet of Things). NodeMCU este alegerea ideală datorită capacităţii sale de a gestiona conexiuni Wi-Fi şi datorită uşurinţei în programare, fiind perfectă pentru proiecte de automatizare de dimensiuni mici şi medii.

Driverul ULN2003 reprezintă un element crucial în arhitectura sistemului, având rolul de a controla motorul pas cu pas 28BYJ-48. Acest driver este echipat cu un array de tranzistoare Darlington, concepute pentru a comuta curenţi mari necesari pentru motoarele pas cu pas. În cazul nostru, conectările între NodeMCU şi ULN2003 sunt configurate astfel: pinul D1 (GPIO5) de pe NodeMCU este conectat la IN1 pe ULN2003, pinul D2 (GPIO4) la IN2, pinul D6 (GPIO12) la IN3 şi pinul D7 (GPIO13) la IN4. Alimentarea motorului pas cu pas 28BYJ-48 se realizează printr-un adaptor 5V şi avem un ground comun între placa NodeMCU şi ULN2003, asigurând astfel o conexiune electrică stabilă şi sigură.

Motorul pas cu pas 28BYJ-48 a fost selectat pentru caracteristicile sale ideale în aplicaţii de precizie, cum ar fi controlul poziţiei jaluzelelor. Acesta funcţionează prin activarea secvenţială a bobinelor sale pentru a genera mişcarea. Specificaţiile tehnice includ un unghi de 5.625 grade per pas şi un raport de reducţie de 1:64, oferind o poziţionare exactă şi repetabilă în cadrul aplicaţiilor practice.

Utilizarea motorului pas cu pas 28BYJ-48 este preferabilă în faţa motoarelor DC în aplicaţii precum controlul jaluzelelor datorită controlului precis al poziţiei. Motoarele DC ar putea necesita un encoder suplimentar pentru feedback-ul necesar poziţionării, în timp ce motorul pas cu pas foloseşte un sistem de control deschis, bazat pe numărul de impulsuri trimise pentru a poziţiona exact. Acest lucru face motorul pas cu pas o soluţie simplă şi eficientă pentru automatizare.

Microstepping-ul este o tehnică utilizată în unele motoare pas cu pas pentru a îmbunătăţi precizia şi a reduce vibraţiile în timpul mişcării. În loc să activeze bobinele în paşi întregi, microstepping-ul împarte fiecare pas în paşi mai mici, permiţând motorului să se deplaseze mai fluid între poziţii. Motorul 28BYJ-48 utilizează un control unipolar, în care fiecare bobină este activată secvenţial într-o direcţie, optimizând astfel controlul şi eficienţa mişcării.

Conectivitatea sistemului este realizată prin NodeMCU, care se conectează la reţeaua Wi-Fi existentă, facilitând comunicarea bidirecţională cu aplicaţia mobilă sau web. Această conectivitate este esenţială pentru controlul la distanţă al jaluzelelor şi pentru transmiterea datelor în timp real către utilizator. Prin intermediul unei interfeţe mobile sau web, utilizatorul poate monitoriza starea jaluzelei şi poate controla deschiderea sau închiderea acesteia în funcţie de preferinţe sau condiţii externe.

Funcţionarea sistemului este determinată de codul încărcat pe NodeMCU, care generează secvenţele de semnale necesare pentru driverul ULN2003. Aceste semnale activează bobinele motorului pas cu pas într-o secvenţă specifică, dictând mişcarea şi poziţionarea precisă a jaluzelei. Controlul numărului şi frecvenţei acestor impulsuri permite ajustarea fină a poziţiei şi a vitezei de mişcare.

Integrarea sistemului în IoT (Internet of Things) extinde funcţionalitatea şi controlul acestuia, permiţând utilizatorului să monitorizeze şi să controleze jaluzelele de la distanţă. Datele privind poziţia jaluzelei şi starea sistemului sunt transmise către utilizator prin intermediul aplicaţiei mobile, oferind un control complet şi feedback imediat asupra operaţiunilor sistemului.

Avantajele sistemului includ precizia şi fiabilitatea controlului motorului pas cu pas, care permite poziţionarea exactă a jaluzelelor în funcţie de preferinţele utilizatorului. Flexibilitatea oferită de NodeMCU în programare şi conectivitate Wi-Fi adaugă posibilităţi extinse de integrare şi control, în timp ce costurile reduse ale componentelor fac soluţia economică şi accesibilă pentru aplicaţii de automatizare rezidenţială sau comercială.

3.2.2.2 Ansamblul plăcii NodeMCU şi a LED-ului

Schema prezentată în figura 3.4 arată modul în care o placă de dezvoltare NodeMCU este conectată la o matrice LED controlată de un circuit integrat MAX7219. Această conexiune permite controlul precis al matricei LED, utilizată pentru a suplimenta lumina sau a afişa diverse informaţii într-un sistem automatizat de control al jaluzelelor. NodeMCU a fost deja discutat anterior, astfel că vom concentra atenţia asupra specificaţiilor şi capabilităţilor MAX7219 şi a conexiunilor sale.

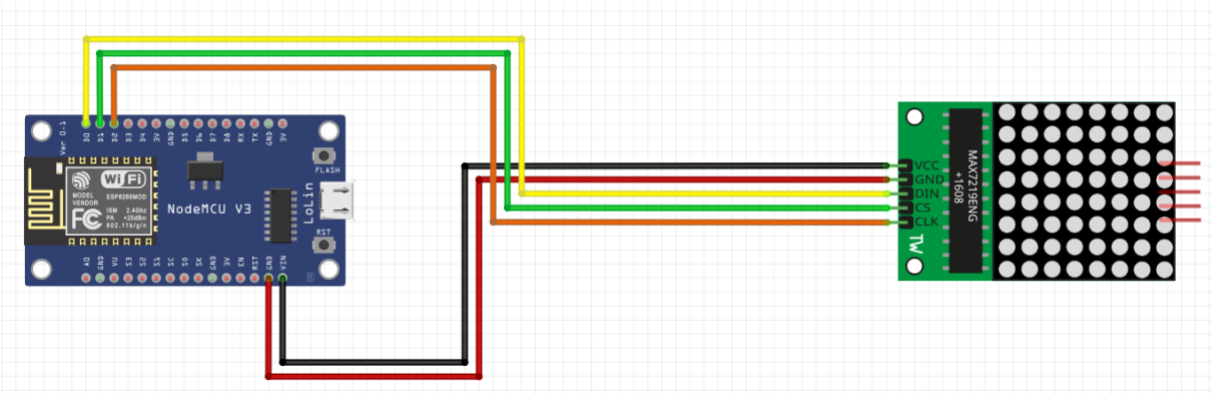


Figura 3.4: Schema electrică a ansamblului plăcii NodeMCU

și a matricei LED controlata de MAX7219

MAX7219 este un circuit integrat de driver pentru afișaj LED, capabil să controleze până la 64 de LED-uri individuale, aranjate într-o matrice de 8x8. Acesta folosește o interfață serială SPI (Serial Peripheral Interface) pentru comunicarea cu microcontroller-ul, reducând numărul de pini necesari la doar cinci: trei pentru date (DIN, CS și CLK), unul pentru alimentare (VCC) și altul pentru masă (GND). Acest design simplifică semnificativ cablajul și controlul afișajelor LED.

Interfața SPI a MAX7219 permite transmiterea datelor seriale de la NodeMCU către circuitul integrat, asigurând astfel un control eficient al fiecărui LED din matrice. Printre principalele caracteristici ale MAX7219 se numără capacitatea de a regla intensitatea luminozității LED-urilor prin modificarea unei valori în registrul de intensitate, modul de scanare limitat care optimizează consumul de energie și funcțiile de detecție a erorilor, utile pentru identificarea LED-urilor defecte sau a conexiunilor slabe.

Conexiunile între NodeMCU și MAX7219 sunt esențiale pentru funcționarea corectă a sistemului. Pinul VCC al MAX7219 este conectat la pinul de 5V de pe NodeMCU, furnizând alimentarea necesară. Pinul GND este conectat la pinul GND de pe NodeMCU pentru a asigura un circuit comun de referință. Pinul DIN de pe MAX7219 este legat la pinul D7 de pe NodeMCU, configurat ca pin de ieșire pentru trimiterea datelor seriale. Pinul CS este conectat la pinul D6 de pe NodeMCU și este utilizat pentru a selecta chip-ul MAX7219 pentru comunicare. În cele din urmă, pinul CLK este conectat la pinul D5 de pe NodeMCU și trimite semnalul de ceas necesar sincronizării datelor între NodeMCU și MAX7219.

În practică, NodeMCU trimite date seriale către MAX7219 prin intermediul pinului DIN, sub controlul semnalelor de ceas (CLK) și chip select (CS). MAX7219 decodează aceste date și le utilizează pentru a controla fiecare LED din matrice. Utilizatorul poate ajusta intensitatea luminii sau poate afișa diverse modele de iluminare prin modificarea datelor trimise către MAX7219. De exemplu, într-un sistem automatizat de control al jaluzelelor, matricea LED poate fi utilizată pentru a suplimenta lumina într-o cameră până când jaluzeaua atinge poziția dorită, asigurând astfel un nivel optim de iluminare.

Utilizarea MAX7219 pentru controlul matricei LED oferă mai multe avantaje semnificative. În primul rând, reduce numărul de pini necesari pentru comunicare, lăsând

astfel mai mulţi pini disponibili pe NodeMCU pentru alte funcţionalităţi. În al doilea rând, MAX7219 gestionează toată logica de control a LED-urilor, simplificând complexitatea codului necesar pe microcontroller. În plus, flexibilitatea şi scalabilitatea sunt îmbunătăţite, deoarece mai multe module MAX7219 pot fi conectate în serie pentru a obţine o intensitate a luminii mai mare.

3.2.2.3 Ansamblul plăcii NodeMCU şi a senzorului de lumina

În figura 3.5 se ilustrează modul în care o placă de dezvoltare NodeMCU este conectată la un senzor de lumină, facilitând măsurarea precisă a intensităţii luminii dintr-o cameră. Această conexiune este esenţială într-un sistem automatizat de control al jaluzelelor, permiţând ajustarea luminozităţii naturală şi artificială în funcţie de condiţiile ambientale. NodeMCU a fost discutat anterior, astfel că accentul va fi pus pe specificaţiile şi capacităţile senzorului de lumină, precum şi pe detaliile conexiunii acestuia cu NodeMCU.

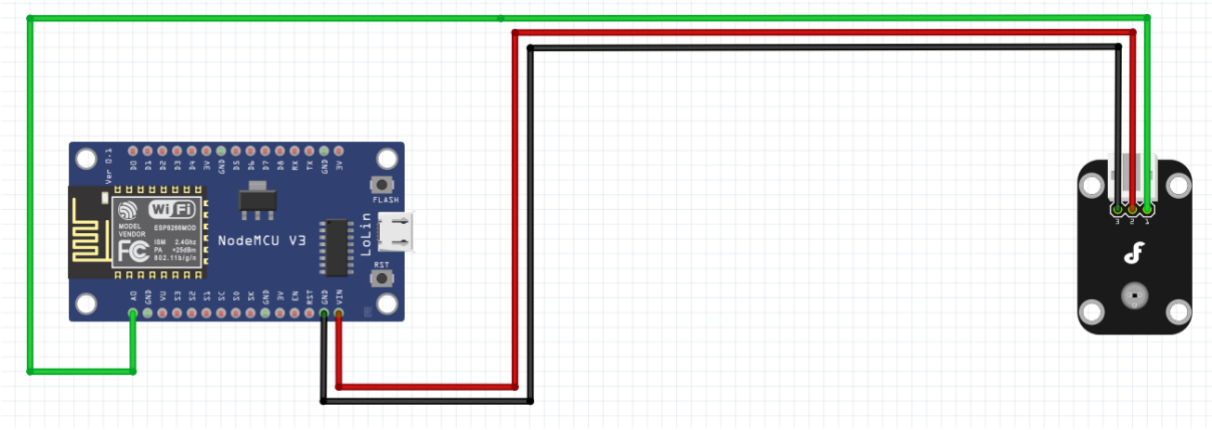


Figura 3.5: Schema electrică a ansamblului plăcii NodeMCU

şi a senzorului de lumină ambientală DFR0026

Senzorul de lumină utilizat în această configuraţie este compact, având dimensiuni de 22x30 mm, ceea ce permite integrarea sa uşoară în diverse aplicaţii electronice. Printre caracteristicile sale remarcabile se numără capacitatea de a măsura intensităţi ale luminii variind de la 1 Lux până la 6000 Lux. Acest interval larg de măsurare îl face potrivit pentru utilizare în diferite medii de iluminare, fie că este vorba despre lumină slabă în încăperi sau lumină puternică în spaţii exterioare. Timpul său de răspuns extrem de rapid, de doar 15 microsecunde, contribuie la obţinerea unor măsurători precise şi aproape în timp real, esenţiale pentru monitorizarea continuă a nivelului de lumină.

Conexiunea fizică între NodeMCU şi senzorul de lumină se realizează prin intermediul pinului A0 de pe NodeMCU, configurat pentru a primi semnale analogice de la senzor. NodeMCU alimentează direct senzorul de lumină, pinul VCC al acestuia fiind conectat la pinul de 5V de pe NodeMCU, asigurând astfel necesarul de alimentare. De asemenea, pinul GND al senzorului este legat la pinul GND al NodeMCU, stabilind un circuit de masă comun necesar pentru o funcţionare corectă şi fiabilă a sistemului.

În cadrul acestei scheme, NodeMCU utilizează datele primite de la senzorul de lumină pentru a transmite ajustările necesare ale iluminării. De exemplu, în cazul în care

intensitatea luminii scade sub un anumit prag prestabilit, NodeMCU-ul configurat ca server principal, poate comanda activarea jaluzelelelor pentru a permite mai multă lumină naturală să pătrundă în cameră sau poate controla luminile artificiale pentru a compensa lipsa luminii naturale. Opus acestei situaţii, în condiţii de lumină intensă, NodeMCU-ul principal poate comanda acţionarea jaluzelelelor pentru a reduce cantitatea de lumină naturală, menţinând un nivel confortabil în încăpere.

Utilizarea senzorului de lumină aduce mai multe avantaje sistemului, inclusiv măsurători precise şi rapide, esenţiale pentru reglarea eficientă a iluminării. Capacitatea sa de a opera într-o gamă largă de intensităţi de lumină îl face extrem de versatil pentru diferite medii de utilizare. Dimensiunile compacte ale senzorului permit integrarea sa simplă în diverse proiecte fără a ocupa mult spaţiu.

3.2.3 Implementarea sistemului hardware

Implementarea sistemului hardware pentru controlul automatizat al jaluzelelor implică integrarea atentă a diferitelor componente şi respectarea strictă a cerinţelor stabilite în etapa de analiză a sistemului hardware. Acest proces asigură că proiectarea realizată este urmată cu precizie, toate conexiunile pinilor plăcilor NodeMCU fiind respectate, pentru a garanta funcţionarea corespunzătoare a sistemului.

În etapa de analiză a sistemului hardware, am stabilit cerinţele fundamentale pentru proiectarea şi implementarea sistemului. Aceste cerinţe includ controlul precis al poziţiei jaluzelei, utilizarea unui motor pas cu pas pentru ajustarea acesteia, măsurarea intensităţii luminii prin integrarea unui senzor de lumină şi suplimentarea luminii prin utilizarea unei matrice LED. De asemenea, am stabilit necesitatea conectivităţii şi controlului centralizat, utilizând plăci NodeMCU pentru a coordona toate componentele şi a permite controlul de la distanţă prin intermediul unei aplicaţii Android.

Proiectarea sistemului hardware a fost realizată astfel încât să răspundă tuturor cerinţelor stabilite în analiza iniţială. Am ales componentele adecvate şi am proiectat conexiunile între acestea pentru a asigura funcţionarea corectă a sistemului.

Implementarea sistemului hardware a inclus, în primul rând, motorul pas cu pas şi conexiunea sa. Motorul utilizat este modelul 28BYJ-48, cunoscut pentru precizia şi fiabilitatea sa. Conectarea acestuia la placa NodeMCU se face prin intermediul unui driver ULN2003, cu conexiunile IN1 de la ULN2003 la D1 de la NodeMCU, IN2 la D2, IN3 la D6 şi IN4 la D7. Alimentarea driverului se realizează de la un adaptor de 5V, iar masa este comună între driver şi placa NodeMCU 1. Această configuraţie permite NodeMCU să controleze rotaţiile motorului pas cu pas pentru a ajusta poziţia jaluzelei în funcţie de necesităţi.

Pentru suplimentarea luminii, utilizăm o matrice LED controlată de un circuit integrat MAX7219, conectată la o altă placă NodeMCU 2, care acţionează ca server principal. Conexiunile între NodeMCU şi MAX7219 sunt realizate astfel: DIN la D0, CS la D1, CLK la D2, VCC la VIN (5V) şi GND la GND. Aceste conexiuni permit NodeMCU să controleze afişajul LED prin intermediul bibliotecii LedControl.h, ajustând intensitatea şi modelele de lumină afişate.

Pentru măsurarea intensității luminii din cameră, utilizăm un senzor de lumină cu dimensiuni reduse, capabil să măsoare lumina în intervalul 1 Lux - 6000 Lux și cu un timp de răspuns de 15 microsecunde. Conexiunile între NodeMCU și senzorul de lumină sunt realizate astfel: Pinul numărul 1 la A0 (pentru feedback de la senzor), pinul numărul 2 la VIN (5V) și pinul numărul 3la GND. Această conexiune asigură alimentarea senzorului de la NodeMCU și trimiterea datelor privind intensitatea luminii la pinul analogic A0 al plăcii NodeMCU.

După implementarea hardware-ului, sistemul a fost supus unor teste riguroase pentru a verifica funcționarea corectă a tuturor componentelor și respectarea cerințelor inițiale. Fiecare componentă a fost testată individual și în contextul întregului sistem pentru a asigura că toate conexiunile și funcționalitățile sunt corecte.

Motorul pas cu pas a fost testat pentru a verifica precizia rotațiilor și capacitatea de a ajusta poziția jaluzelei conform comenzilor primite de la NodeMCU. Testele au inclus verificarea răspunsului la diferite comenzi și asigurarea că jaluzeaua se mișcă în mod consistent și precis. Senzorul de lumină a fost testat pentru a asigura măsurarea precisă a intensității luminii în diferite condiții de iluminare. Testele au inclus expunerea senzorului la diferite niveluri de lumină și verificarea datelor transmise către NodeMCU.

Matricea LED a fost testată pentru a verifica funcționarea corectă a afișajului și capacitatea de a ajusta intensitatea luminii. Testele au inclus afișarea diferitelor modele de lumini și ajustarea luminozității pentru a completa lumina naturală. Întregul sistem a fost testat pentru a verifica conectivitatea și comunicarea între module prin rețeaua Wi-Fi. Testele au inclus trimiterea și primirea comenzilor de la aplicația Android, verificarea răspunsului sistemului și asigurarea unei operări fluide și integrate a tuturor componentelor.

Implementarea sistemului hardware pentru controlul automatizat al jaluzelelor a urmat îndeaproape cerințele stabilite în analiza inițială și proiectarea realizată. Toate conexiunile pinilor plăcilor NodeMCU au fost respectate, asigurând funcționarea corespunzătoare. Prin integrarea motorului pas cu pas, a senzorului de lumină și a matricei LED, am creat un sistem robust și fiabil, capabil să ajusteze poziția jaluzelei și intensitatea luminii în funcție de condițiile de iluminare și preferințele utilizatorului. Testele riguroase au confirmat că sistemul funcționează conform specificațiilor, oferind un control precis și confortabil al iluminării în locuință.

Pentru realizarea conexiunii fizice dintre motorul pas cu pas și jaluzea, este necesar să utilizăm un adaptor special care să faciliteze acest proces. Dat fiind faptul că automatizarea jaluzelelor cu roletă nu este o idee nouă, există deja soluții disponibile pe internet sub forma unor adaptori proiectați special pentru acest scop, după cum se poate vedea in figura 3.5.

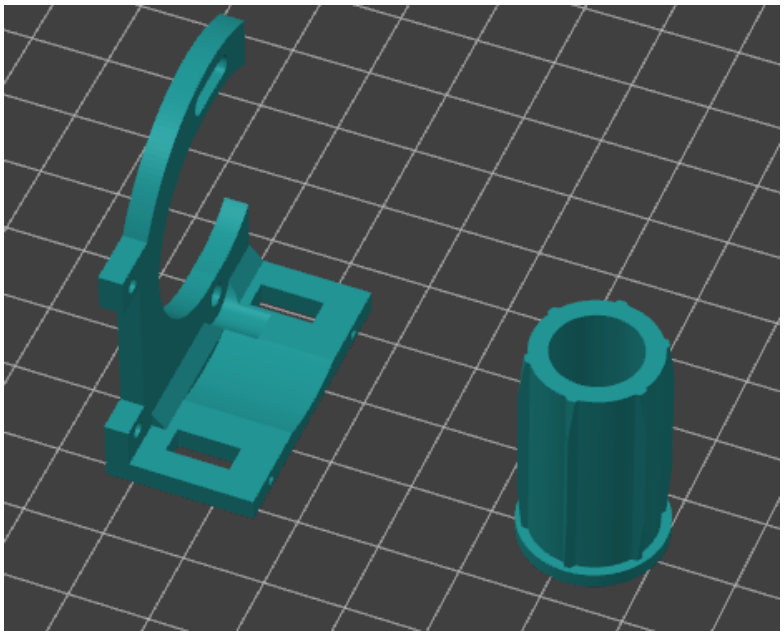


Figura 3.5: Modelele 3d pentru conectarea motorului pas cu pas la jaluzea, și pentru montarea lui pe perete

În vederea implementării acestei soluții, am decis să optăm pentru imprimarea 3D a unui adaptor care să asigure o conexiune robustă și eficientă între motor și mecanismul jaluzelei. Vom utiliza o imprimantă 3D Prusa i3 pentru a crea acest adaptor, care va permite motorului să controleze mișcarea jaluzelei într-un mod precis și fiabil. Prin utilizarea tehnologiei de imprimare 3D oferite de imprimanta Prusa i3, putem personaliza adaptorul pentru a se potrivi perfect specificațiilor mecanismului jaluzelei și ale motorului pas cu pas 28BYJ-48. Alegerea unui model deja existent, disponibil online, ne permite să economisim timp și resurse, beneficiind în același timp de o soluție verificată și testată în practică. [12]

Această abordare academică subliniază importanța utilizării resurselor existente pentru a optimiza procesul de implementare și a asigura funcționalitatea corectă a sistemului automatizat de control al jaluzelelor. Adaptorul imprimat cu imprimanta Prusa i3 nu doar că facilitează conexiunea fizică între motor și jaluzea, dar și contribuie la fiabilitatea și durabilitatea întregului sistem, asigurând o performanță optimă pe termen lung.

Imprimarea 3D a fost realizată utilizând filament din ABS cu un diametru de 1,75 mm. Pentru a putea fi imprimate, modelele 3D au fost încărcate în software-ul PrusaSlicer, unde au fost convertite în cod G. Această conversie este un pas esențial în procesul de imprimare 3D, deoarece codul G oferă instrucțiuni precise imprimantei despre cum să creeze obiectul strat cu strat.

Figura 3.6: Piesele printate cu ajutorul imprimantei 3d Prusa i3

Întregul proces de imprimare a avut o durată de o oră și cinci minute. Figura 3.6 prezintă rezultatul final al imprimării, evidențiind calitatea și detaliile obiectului obținut prin această metodă. În plus, Figura 3.7 ilustrează motorul conectat la adaptorul special.

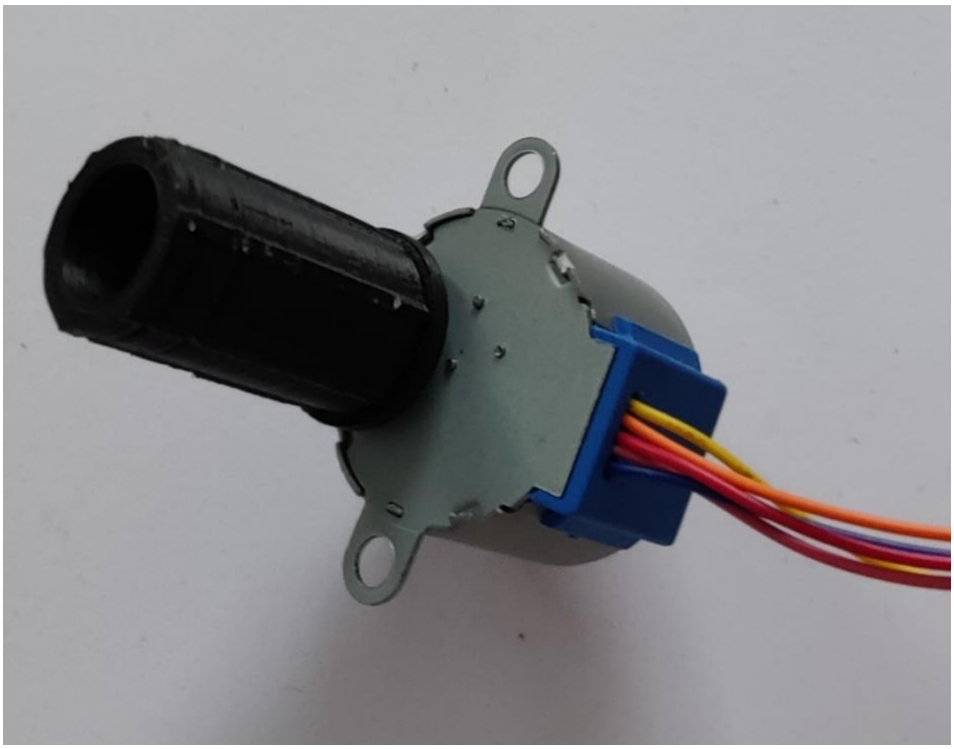


Figura 3.7: Ansamblul motor - adaptor

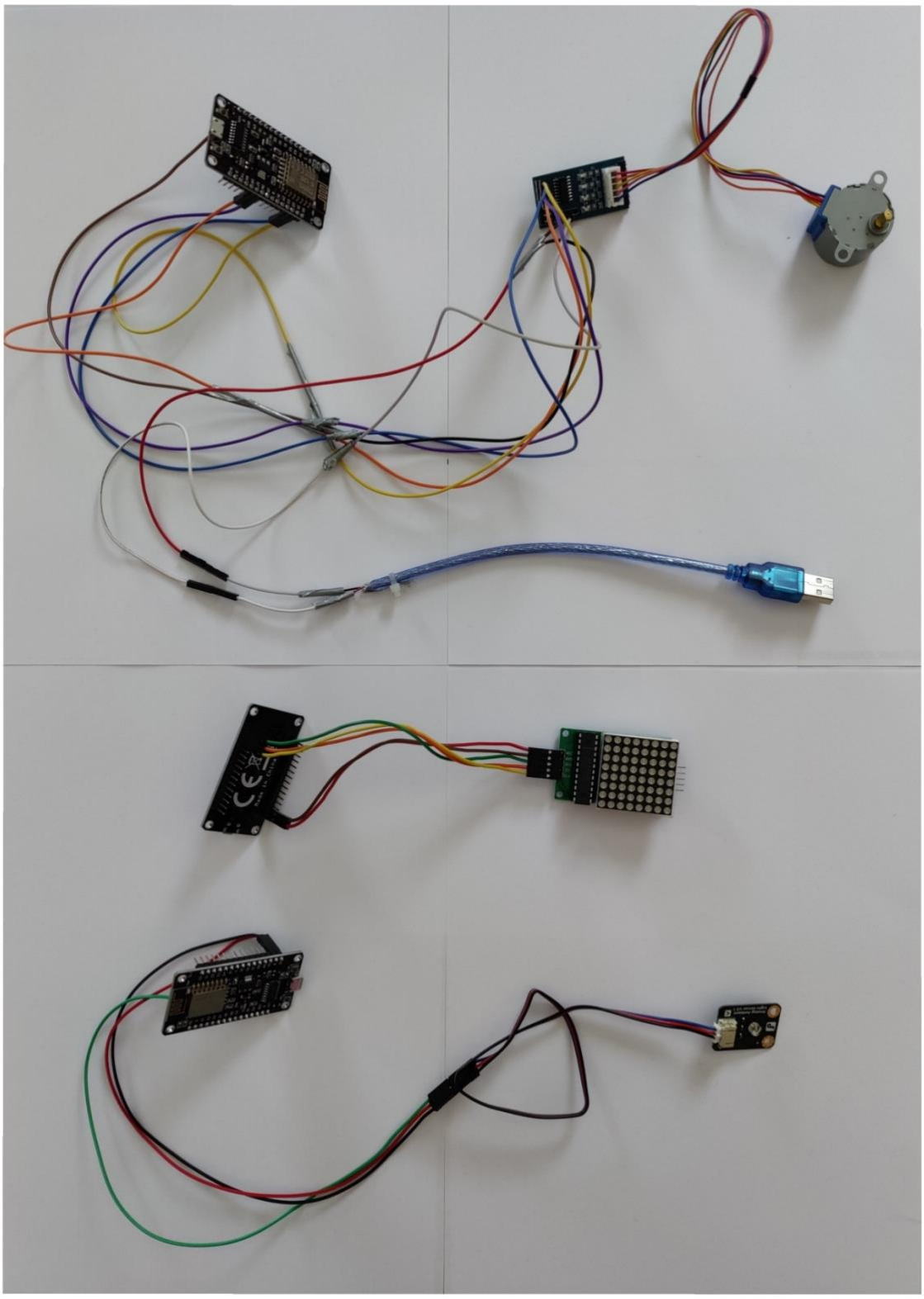


Figura 3.6: ceva

În Figura 3.8 sunt prezentate toate componentele sistemului, inclusiv cele trei plăci NodeMCU, motorul pas cu pas 28BYJ-48, driverul ULN2003, matricea LED MAX7219 și senzorul de lumină ambientală, toate conectate conform specificațiilor și proiectării stabilite anterior.

În Figura 3.9 este ilustrată poziționarea optimă a componentelor în cadrul machetei. Fiind un prototip, nu s-a optat pentru o fixare foarte sigură, deoarece scopul principal este demonstrarea capacităților sistemului de automatizare și control inteligent al jaluzelelor.

Figura 3.9: Macheta pentru demonstrarea capacităților sistemului de automatizare și control inteligent al jaluzelelor

3.3 Analiza, proiectarea si implementarea sistemului software

3.3.1 Analiza sistemului software

Analiza sistemului software este un pas important în vederea proiectării eficiente a unui proiect de automatizare şi control inteligent al jaluzelelor. Acest proces presupune evaluarea nevoilor atât ale utilizatorului, cât şi ale sistemului hardware, în vederea definirii arhitecturii de comunicare a întregului sistem.

Definirea arhitecturii de comunicare implică stabilirea modului în care diferitele module ale sistemului vor comunica între ele. În acest caz, NodeMCU 2 acţionează ca server principal, primind comenzi de la aplicaţia Android şi de la modulul pentru lumină, dar şi comandând modulul pentru controlul motorului, folosind protocoale de comunicaţie HTTP.

Analiza sistemului software asigură o proiectare structurată şi bine planificată, ducând la un sistem robust şi fiabil.

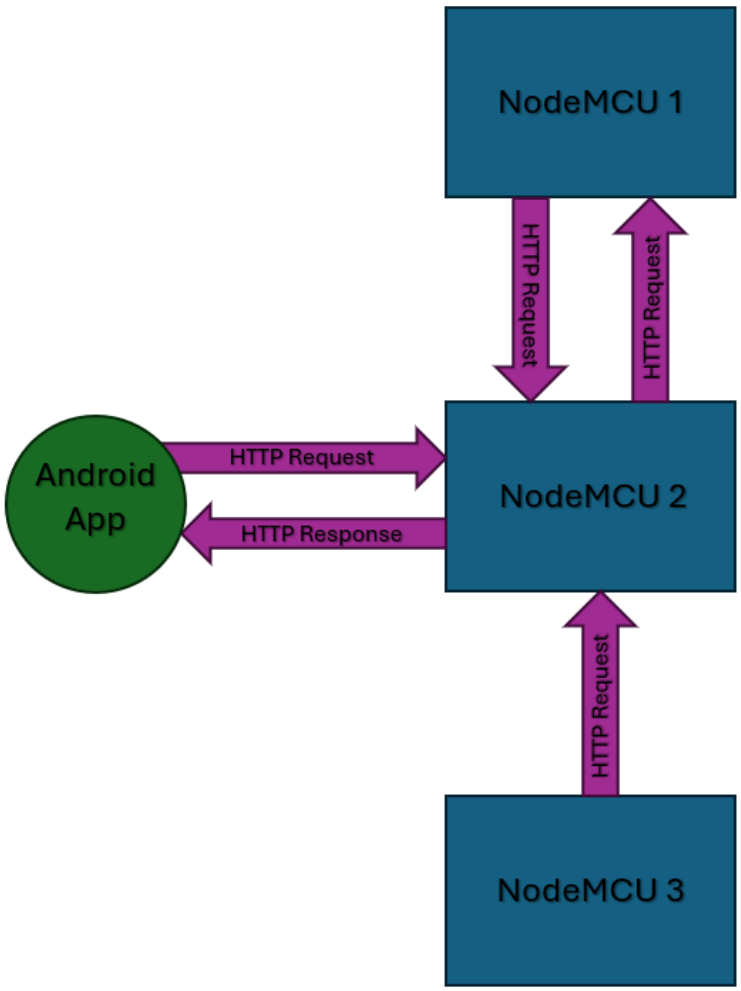


Figura 3.10: Arhitectura modulelor Arduino

În figura 3.10 se poate observa arhitectura şi modul de comunicare între modulele Arduino, dar şi între aplicaţia Android şi serverul principal, reprezentat de NodeMCU 2. Transmiterea datelor se face cu ajutorul cererilor HTTP, precum şi cu răspunsurile la aceste cereri, în cazul comunicării între serverul principal Arduino şi aplicaţia Android, reprezentate în figură cu ajutorul săgeţilor mov.

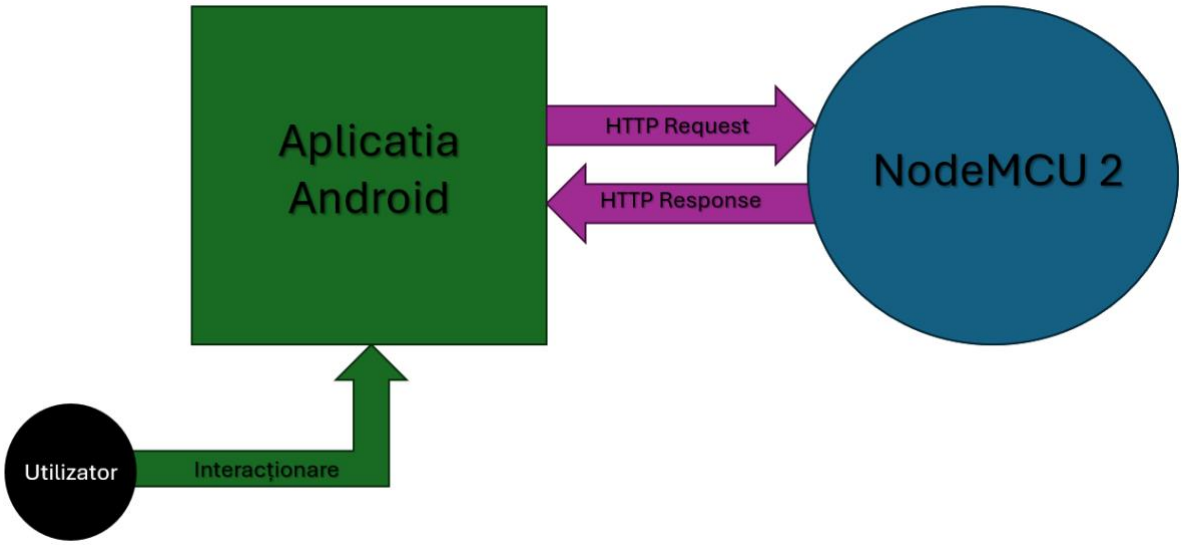


Figure 3.11: Arhitectura modului Android

În figura 3.11 se observă arhitectura şi modul de comunicare al aplicaţiei Android. Utilizatorul interacţionează cu aplicaţia Android, iar, dacă este cazul, aplicaţia creează cereri HTTP către serverul principal, NodeMCU 2. Acesta transmite înapoi către aplicaţie date importante, precum intensitatea LED-ului şi poziţia jaluzelei, în cazul modului de funcţionare manuală.

3.3.2 Proiectarea sistemului software

Proiectarea sistemului software ne aduce mai aproape cu un pas de implementarea unui proiect de automatizare şi control inteligent al jaluzelelor. Acest proces implică definirea arhitecturii software, alegerea tehnologiilor adecvate şi planificarea fluxurilor de date şi comunicaţiei între module.

În primul rând, se stabileşte arhitectura software, care determină posibilele stări în care se vor afla modulele şi secvenţele logice care duc la transpunerea dintr-o stare în alta. În acest caz, NodeMCU 2 acţionează ca server principal, primind comenzi de la aplicaţia Android şi de la modulul pentru lumină, eventual şi de la modulul motorului, şi realizează executarea într-un stil ciclic a programului, în vederea transmiterii de comenzi celorlalte module şi a feedback-ului pentru aplicaţia Android, utilizând protocoalele HTTP.

Alegerea tehnologiilor adecvate este crucială pentru succesul implementării. Pentru programarea plăcilor NodeMCU, se utilizează limbajul C++ şi bibliotecile specifice, precum ESP8266WiFi.h şi ESP8266WebServer.h. Pentru dezvoltarea aplicaţiei Android, se foloseşte limbajul Kotlin şi framework-ul Jetpack Compose, care permite crearea de interfeţe de utilizator moderne şi intuitive.

Planificarea fluxurilor de date implică definirea exactă a modului în care informaţiile vor fi transmise şi procesate între module. De exemplu, senzorul de lumină trimite datele către NodeMCU 2, care apoi ia deciziile necesare pentru ajustarea poziţiei jaluzelei şi intensităţii LED-ului.

Proiectarea sistemului software asigură o implementare coerentă şi eficientă, facilitând dezvoltarea unui sistem robust şi fiabil, capabil să răspundă cerinţelor utilizatorilor şi să funcţioneze eficient în condiţii reale.

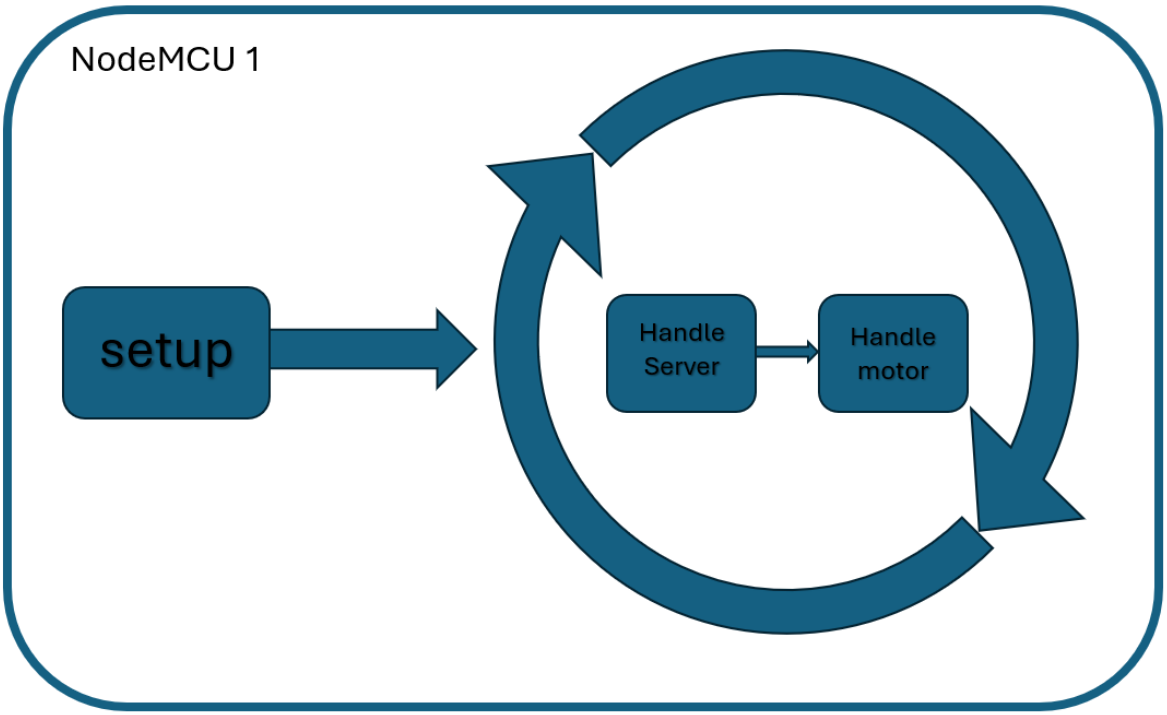


Figura 3.12: Schema logică pentru modulul software al motorului

În figura 3.12 se poate observa modul tipic de funcţionare al unei plăcuţe Arduino, având partea de setup şi loop.

Secţiunea setup este prima care rulează atunci când NodeMCU-ul este pornit sau resetat. Aceasta este utilizată pentru iniţializarea configuraţiilor şi setărilor necesare pentru funcţionarea dispozitivului. În setup, vom include următoarele activităţi:

- Definirea variabilelor necesare programului, cum ar fi adresa IP a serverului principal, numele şi parola adresei Wi-Fi.
- Conectarea la reţeaua Wi-Fi, pe care o vom explica în detaliu la următorul pas.
- Iniţializarea serverului HTTP şi definirea endpoint-urilor.
- Iniţializarea motorului pas cu pas conform documentaţiei librăriei “Stepper.h”.

După ce setup a fost rulat şi configuraţiile iniţiale au fost realizate, NodeMCU-ul intră într-o buclă continuă, numită loop. Această buclă rulează repetitiv şi este utilizată pentru a menţine dispozitivul activ şi receptiv la diferite evenimente. În cadrul buclei loop, avem două componente principale:

- handle server.
- handle motor.

Acestea urmează să fie dezvoltate la următorul pas, implementarea sistemului software.

Din cauza complexităţii modulului server dedicat controlului LED-ului, care se ocupă atât de gestionarea precisă şi în timp real a unei matrici LED, cât şi de luarea deciziilor necesare pentru comanda motorului, am considerat necesară împărţirea logicii de funcţionare în două scheme distincte: setup şi loop.

Figura 3.13: Schema logică pentru metoda setup din modulul software al LED-ului

În figura 3.13 este ilustrat fiecare pas realizat în cadrul metodei setup, pentru a asigura o funcţionare optimă a modulului. Similar cu modulul motorului, începem prin definirea variabilelor esenţiale, care vor fi necesare pentru restul iniţializărilor. Ne conectăm la reţeaua Wi-Fi, ale cărei nume şi parolă le-am stabilit anterior. Ulterior, iniţializăm matricea LED conform specificaţiilor din documentaţia librăriei „LedControl.h”.

În ceea ce priveşte configurarea serverului, vom avea mai multe endpoint-uri, deoarece acest modul funcţionează ca server principal, primind comenzi inclusiv de la aplicaţia Android. După definirea tuturor metodelor asociate acestor endpoint-uri, pornim serverul HTTP. La finalizarea metodei setup, aplicaţia va continua execuţia cu secţiunea loop.

Această divizare clară între setup şi loop permite o structurare logică şi metodică a codului, asigurând astfel o mai bună organizare şi o întreţinere facilă. Procesul de iniţializare cuprinde nu doar stabilirea conexiunii Wi-Fi şi iniţializarea componentelor hardware, dar şi configurarea serverului HTTP pentru a răspunde eficient cerinţelor aplicaţiei Android. Astfel, se garantează o funcţionare robustă şi fiabilă a întregului sistem, capabil să răspundă cerinţelor de control şi automatizare inteligentă a jaluzelelor.

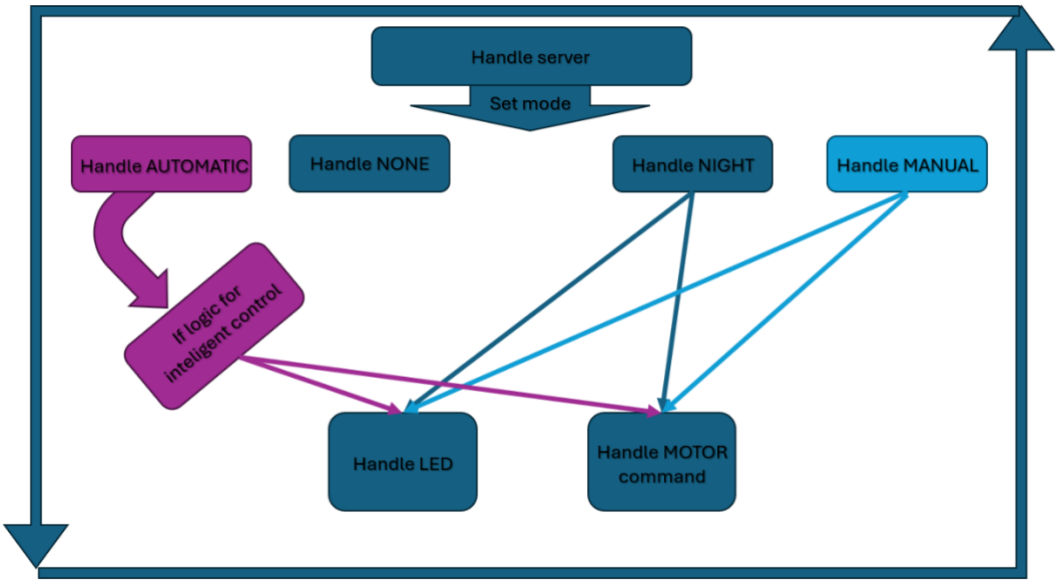


Figura 3.14: Schema logică pentru metoda loop din modulul software al LED-ului

Funcţia loop gestionează multiple moduri de operare distincte pentru sistem, fiecare având cerinţe specifice. Aceste moduri de operare sunt: none, manual, night şi automatic. Fiecare mod este asociat unei funcţii specifice, care implementează comportamentul necesar. În Figura 3.14 este ilustrat modul de funcţionare al funcţiei loop, iar în Figura 3.15 este prezentat echivalentul în pseudocod al procesului de selecţie şi apelare a modului de operare.

```
funcție loop() {
  server.handleClient()

  dacă operatingMode este egal cu "none" atunci
    handle_NONE_mode()
  altfel dacă operatingMode este egal cu "manual" atunci
    handle_MANUAL_mode()
  altfel dacă operatingMode este egal cu "night" atunci
    handle_NIGHT_mode()
  altfel dacă operatingMode este egal cu "automatic" atunci
    handle_AUTOMATIC_mode()
  altfel
    Serial.println("Unknown operating mode")
  sfârșit dacă
}
```

Figura 3.15: Secventa pseudocod pentru funcţia loop al modulului software responsabil de controlul LED şi gestionarea serverului principal

Prima linie din loop() este server.handleClient(). Aceasta se asigură că serverul web integrat gestionează toate cererile HTTP primite. În contextul acestui modul, NodeMCU funcţionează ca un server web care primeşte cereri de la o aplicaţie Android sau de la modulul senzorului de lumină. Fiecare cerere HTTP primită este analizată şi, în funcţie de endpoint-ul solicitat, sunt executate diferite acţiuni, cum ar fi modificarea modului de

operare, actualizarea valorii intensității luminii ambientale sau valorilor cerute pentru poziția jaluzelei și intensitatea LED-ului.

Funcția handle_NONE_mode este responsabilă pentru resetarea tuturor valorilor la stările implicite și oprirea oricărei activități. Acest mod este util pentru a asigura că sistemul se află într-o stare inertă atunci când nu este necesară nicio acțiune, cum ar fi la pornirea sistemului.

În modul manual, funcția handle_MANUAL_mode gestionează atât intensitatea LED-ului, cât și poziția jaluzelei. Intensitatea LED-ului este ajustată pe baza valorii specificate de utilizator și este setată la nivel local de către NodeMCU 2, iar poziția jaluzelei este comandată către NodeMCU 1 conform valorilor primite.

Modul night implică scăderea intensității LED-ului până la oprirea completă a acestuia și comandarea închiderii complete a jaluzelei pentru a asigura întunericul necesar pe timpul nopții.

Modul automat este cel mai complex, implicând ajustarea automată a poziției perdelei și a intensității LED-ului pentru a menține un nivel de lumină ambientală specificat de utilizator, în timp ce este favorizată lumina naturală. Această ajustare se face pe baza valorii citite de senzorul de lumină ambientală, dar și a intensității LED-ului, pentru favorizarea luminii naturale.

Metoda de manipulare a LED-ului este una simplă, deoarece LED-ul este conectat direct la NodeMCU 2. Folosind metodele oferite de biblioteca „LedControl.h”, ajustăm valoarea intensității LED-ului.

Metoda de manipulare a motorului presupune trimiterea unei comenzi către serverul HTTP găzduit de placa NodeMCU 1. Aceasta va prelua integral sarcina de îndeplinire a comenzii sau, alternativ, va trimite înapoi, prin intermediul unei cereri HTTP, valoarea la care a ajuns, așteptând ulterior comanda de oprire, „halt”.

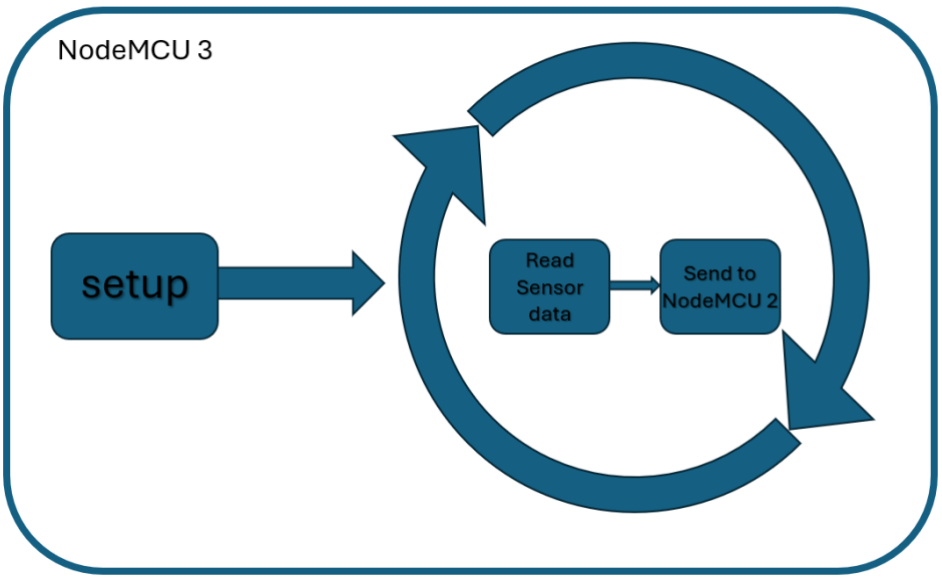


Figura 3.17: Schema logică pentru modulul software al senzorului de lumină ambientală

În figura 3.17 se poate observa din nou modul tipic şi simplu de funcţionare al unei plăcuţe Arduino, având partea de setup şi loop.

Asemenea modulului software pentru NodeMCU 1, in partea de setup vom incepe cu definirea variabilelor necesare programului, cum ar fi adresa IP a serverului principal, numele şi parola adresei Wi-Fi. Dupa care, datorita simplitatii senzorului de lumina ambientala ales, ne ramane doar sa ne conectam la reteaua Wi-fi

După ce secţiunea setup a fost rulată şi configuraţiile iniţiale au fost realizate, NodeMCU-ul intră în funţia loop. Aceasta rulează repetitiv şi este utilizată pentru a menţine dispozitivul activ şi receptiv la diverse evenimente, cum ar fi schimbarile de intensitate a luminii ambientale. În cadrul buclei loop, se regăsesc două componente principale:

- Read Sensor data.
- Send to NodeMCU 2.

Acestea urmează să fie dezvoltate in următorul pas, implementarea sistemului software.

Pentru dezvoltarea aplicaţiilor mobile, alegerea instrumentelor şi a framework-urilor adecvate este crucială pentru succesul unui proiect. Android Studio şi Jetpack Compose sunt două astfel de tehnologii care au câştigat popularitate în rândul dezvoltatorilor Android.

Android Studio reprezintă principalul mediu de dezvoltare integrat (IDE) pentru crearea aplicaţiilor Android, fiind dezvoltat de Google şi oferind un set de instrumente şi caracteristici menite să faciliteze dezvoltarea rapidă şi eficientă a aplicaţiilor mobile. Printre cele mai remarcabile caracteristici se numără editorul de cod avansat, care dispune de funcţionalităţi precum completarea automată, refactorizarea şi navigarea rapidă, toate contribuind la o scriere şi întreţinere mai eficientă a codului. De asemenea, emulatorul Android integrat permite testarea aplicaţiilor pe diverse versiuni şi configuraţii ale sistemului de operare Android, eliminând necesitatea de a dispune de multiple dispozitive fizice.

În plus, Android Studio include instrumente avansate de debugging, esenţiale pentru identificarea şi rezolvarea problemelor de performanţă şi funcţionalitate ale aplicaţiilor. Suportul pentru sistemul de build Gradle adaugă flexibilitate şi eficienţă în gestionarea dependenţelor şi automatizarea proceselor de build. Integrarea cu diverse servicii Google, cum ar fi Firebase, împreună cu actualizările şi suportul continuu din partea comunităţii şi a echipei de dezvoltare Google, consolidează Android Studio ca un mediu de dezvoltare robust şi de încredere pentru dezvoltatorii de aplicaţii Android.

Jetpack Compose reprezintă o revoluţie în designul interfeţelor pentru utilizatori (UI) pentru platforma Android, oferind un framework modern bazat pe un model de programare declarativ. Spre deosebire de abordarea tradiţională, care utilizează XML şi programarea imperativă, Jetpack Compose permite dezvoltatorilor să definească UI-ul într-un mod mult mai intuitiv şi flexibil, folosind limbajul Kotlin.

Unul dintre principalele avantaje ale Jetpack Compose este simplitatea şi concizia pe care o aduce procesului de dezvoltare. Prin eliminarea necesităţii de a scrie fişiere XML

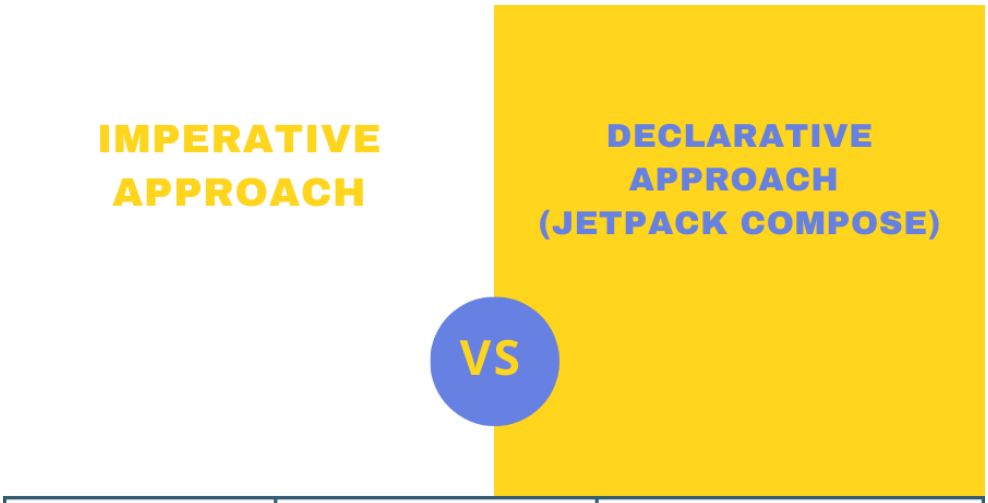
separate pentru layout-uri şi de a le lega cu cod Java sau Kotlin, dezvoltatorii pot defini direct UI-ul în codul Kotlin, reducând astfel considerabil cantitatea de cod necesară.

De asemenea, Jetpack Compose încurajează reutilizarea componentelor UI, cunoscute sub numele de funcţii compozabile (composable functions). Această abordare conduce la un cod mai modular şi mai uşor de întreţinut, facilitând astfel gestionarea şi actualizarea interfeţei pentru utilizator.

Un alt beneficiu semnificativ este suportul pentru actualizări în timp real (Live Updates), care permite ca modificările aduse UI-ului să se reflecte imediat în aplicaţie. Acest lucru reduce considerabil timpul pentru testare şi îmbunătăţeşte productivitatea dezvoltatorilor.

În plus, integrarea uşoară cu Android Studio face din Jetpack Compose un instrument extrem de eficient pentru dezvoltarea aplicaţiilor Android. Android Studio oferă suport complet pentru Jetpack Compose, inclusiv previzualizarea în timp real a UI-ului, facilitând astfel dezvoltarea şi testarea rapidă a interfeţelor utilizator.

Astfel, Jetpack Compose se afirmă ca o soluţie inovatoare şi eficientă pentru dezvoltarea UI-ului pe platforma Android, aducând multiple beneficii care simplifică şi optimizează întregul proces de dezvoltare.



Aspect	Imperative Approach	Declarative Approach (Jetpack Compose)
Programming Style	Step-by-step instructions	High-level descriptions
UI Definition	Explicitly create and manage UI elements	Describe UI structure and interactions
Updates	Manual management of updates	Automatic reconciliation of UI changes
Data Flow	Often complex bidirectional	Unidirectional: data triggers UI updates

Figura 3.18: Programarea imperativă comparată cu cea declarativă, in contextul dezvoltării de interfeţe pentru utilizator [13]

În cadrul designului interfeței pentru utilizator (UI), există două abordări fundamentale: programarea imperativă și programarea declarativă. Diferențele dintre cele doua abordări sunt reprezentate în figura 3.18. Programarea imperativă implică specificarea detaliată a pașilor pe care sistemul trebuie să îi urmeze pentru a realiza o anumită sarcină. Aceasta se traduce, în contextul designului UI, prin definirea explicită a tuturor interacțiunilor și modificărilor de stare.

Avantajele abordării imperative sunt multiple. În primul rând, aceasta oferă un control detaliat asupra fiecărui aspect al interfeței, permițând astfel optimizarea fină a comportamentului și performanței aplicației. De asemenea, flexibilitatea în gestionarea stării este un alt beneficiu important, permițând dezvoltatorilor să gestioneze explicit stările și tranzițiile, ceea ce este deosebit de util în cazul aplicațiilor complexe.

Cu toate acestea, programarea imperativă prezintă și dezavantaje semnificative. Codul poate deveni rapid complex și dificil de întreținut pe măsură ce interfața pentru utilizator devine mai complicată. În plus, necesitatea de a separa layout-urile XML de logica Java/Kotlin poate conduce la un cod mai fragmentat și mai greu de gestionat.

Pe de altă parte, programarea declarativă se concentrează pe descrierea rezultatelor dorite, lăsând detaliile implementării la latitudinea framework-ului. În Jetpack Compose, de exemplu, dezvoltatorii descriu ce ar trebui să afișeze UI-ul în funcție de starea actuală a aplicației. Această abordare oferă avantaje clare: codul declarativ este de obicei mai concis și mai ușor de înțeles, facilitând astfel întreținerea și colaborarea între dezvoltatori. De asemenea, componentele declarative sunt mai ușor de reutilizat și de combinat, ceea ce duce la un design UI mai flexibil și extensibil. Gestionarea automată a stării de către framework reduce riscul de erori și de inconsistențe, simplificând semnificativ procesul de dezvoltare.

Totuși, programarea declarativă are și dezavantaje. Dezvoltatorii pot avea mai puțin control asupra detaliilor fine ale comportamentului UI, ceea ce poate fi o limitare în cazurile foarte specifice. În plus, tranziția de la o abordare imperativă la una declarativă poate necesita o perioadă de adaptare, mai ales pentru cei obișnuiți cu metodele tradiționale.

Un exemplu elocvent al diferenței între aceste două abordări este gestionarea stării în aplicațiile complexe. În programarea imperativă, gestionarea stării implică de obicei multiple variabile și metode care trebuie sincronizate manual. În schimb, Jetpack Compose, prin mecanisme integrate precum remember și mutableStateOf, simplifică foarte mult acest proces în cadrul programării declarative.

Astfel, abordarea declarativă, utilizată cu ajutorul Jetpack Compose, se remarcă prin simplitate și claritate, capturând toată logica UI într-un singur bloc de cod, fără a necesita legături explicite între layout și logica de aplicare, ceea ce evidențiază avantajele acestei metodologii în raport cu programarea imperativă.

Impactul adoptării Jetpack Compose și a abordării declarative asupra dezvoltării aplicațiilor mobile este semnificativ, aducând multiple beneficii atât în termeni de productivitate, cât și de calitate a codului. Reducerea complexității codului și scurtarea

timpului de feedback contribuie la livrarea mai rapidă a funcţionalităţilor, îmbunătăţind astfel calitatea generală a codului produs.

Abordarea declarativă facilitează crearea de componente UI reutilizabile şi modulare, aspect esenţial pentru scalabilitatea şi întreţinerea pe termen lung a aplicaţiilor. Integrarea cu limbajul modern şi expresiv Kotlin adaugă un nivel suplimentar de eficienţă şi claritate în scrierea codului, contribuind la o dezvoltare mai robustă şi mai uşor de gestionat.

În ceea ce priveşte performanţa, deşi iniţial au existat preocupări legate de utilizarea UI-urilor declarative, Jetpack Compose a demonstrat că poate oferi performanţe comparabile sau chiar superioare abordărilor imperative tradiţionale. Acest lucru este posibil datorită optimizărilor interne şi gestionării eficiente a stării şi reîncadrării UI-ului.

Aşadar, adoptarea Android Studio şi Jetpack Compose, împreună cu abordarea declarativă în designul UI, reprezintă un bun pas către modernizarea şi eficientizarea dezvoltării aplicaţiilor mobile. Beneficiile clare în termeni de productivitate, întreţinere şi calitate a codului fac din aceste tehnologii o alegere atractivă pentru dezvoltatorii Android. Deşi abordarea declarativă poate necesita o perioadă de adaptare, avantajele pe termen lung sunt evidente, promiţând un viitor mai luminos pentru dezvoltarea aplicaţiilor mobile.

În acest proiect, am ales să folosesc Android Studio şi Jetpack Compose, adoptând o abordare declarativă pentru designul UI. Aceasta a simplificat semnificativ procesul de dezvoltare, permiţându-mi să ne concentrez pe aspectele esenţiale, cum ar fi crearea unei interfeţe prietenoase şi asigurarea unei conexiuni fiabile cu serverul principal. Jetpack Compose a redus complexitatea iniţială a designului, eliminând necesitatea detaliilor laborioase ale layout-urilor şi interacţiunilor, facilitând astfel crearea rapidă a unor UI-uri moderne şi atractive. Datorită componentei modulare a funcţiilor composable, fiecare parte a interfeţei a putut fi dezvoltată şi testată independent, asigurând astfel consistenţa şi uşurinţa în utilizare. Conexiunea fiabilă cu serverul principal va fi realizată prin cereri HTTP. Alegerea acestei abordări şi a framework-ului asociat în dezvoltarea aplicaţiei Android a eliminat necesitatea unei proiectări la fel de detaliate, dar am ţinut cont de elementele esenţiale: crearea unei interfeţe intuitive şi prietenoase, precum şi asigurarea unei conexiuni clare şi fiabile între aplicaţie şi serverul principal.

3.3.3 Implementarea sistemului software

Implementarea software a modulului motorului pas cu pas, NodeMCU 1, utilizează o placă NodeMCU pentru a coordona funcţionarea motorului pas cu pas 28BYJ-48, care ajustează poziţia jaluzelei. Sistemul primeşte comenzi de la un server şi poate funcţiona atât în mod automat, cât şi în mod manual. Implementarea se bazează pe analiza şi proiectarea realizate anterior, asigurând că toate cerinţele sunt respectate şi că sistemul funcţionează eficient.

Prima etapă a implementării este conectarea plăcii NodeMCU la reţeaua WiFi. Acest lucru permite sistemului să comunice cu serverul şi să primească comenzi de control. În codul ilustrat in figura 3.19, reţeaua WiFi este configurată cu SSID şi parola, iar NodeMCU se conectează la aceasta.

```
const char* ssid = "DIGI-xxxx";
const char* password = "xxxxxxxxx";

void connect_to_wifi(){
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
}
```

Figura 3.19: Metoda folosită de toate plăcile NodeMCU pentru conectarea la rețeaua Wi-Fi

Această secțiune de cod este crucială pentru stabilirea conectivității rețelei, care este esențială pentru funcționarea sistemului. Fără o conexiune stabilă la rețea, NodeMCU nu ar putea primi comenzi sau trimite date despre poziția jaluzelei.

Motorul pas cu pas 28BYJ-48 este controlat de placa NodeMCU prin intermediul unui driver ULN2003. Motorul este configurat pentru a se roti cu o viteză specificată, iar funcțiile step() sunt utilizate pentru a mișca motorul în pași mici. În codul din figura 3.20, motorul este inițializat cu numărul de pași pe revoluție și pinii de control corespunzători.

```
const int NUMBER_OF_STEPS = 1;
const int StepsPerRevolution = 2048;
Stepper myStepper(StepsPerRevolution, D1, D6, D2, D7);

void setup() {
  Serial.begin(115200);
  connect_to_wifi();
  Serial.print("NodeMCU IP Address: ");
  Serial.println(WiFi.localIP());

  // Set the motor speed (RPM)
  myStepper.setSpeed(15);
}
```

Figura 3.20: Inițializarea motorului pas cu pas

Această configurație permite controlul precis al motorului, esențial pentru ajustarea exactă a poziției jaluzelei.

NodeMCU trebuie să gestioneze comenzile primite de la server pentru a mișca motorul în mod corespunzător. Comenzile sunt primite printr-un server web local, care rulează pe NodeMCU. Serverul gestionează solicitările HTTP și trimite răspunsuri în funcție de comanda primită.

```
ESP8266WebServer server(80);

void setup() {
  Serial.begin(115200);
  connect_to_wifi();
  Serial.print("NodeMCU IP Address: ");
  Serial.println(WiFi.localIP());

  // Set the motor speed (RPM)
  myStepper.setSpeed(15);

  // Data receive server NodeMCU
  server.on("/motor", HTTP_GET, []() {
    if (server.hasArg("command") && server.hasArg("percentage")) {
      requestedPercentage = (server.arg("percentage")).toInt();
      motorCommand = server.arg("command");
      Serial.println("Received command " + motorCommand + "\nRequested percentage " + requestedPercentage);
      server.send(200, "text/plain", "Mode received");
    } else {
      server.send(400, "text/plain", "Bad Request");
    }
  });

  server.begin();
  Serial.println("HTTP server started");
}
```

Figura 3.21: Inițializarea si configurarea serverului HTTP în cadrul plăcii NodeMCU 1

În această secțiune de cod, serverul web gestionează o rută: /motor. Ruta /motor primește comenzile pentru motor și procentajul dorit al deschiderii jaluzelei.

Funcția loop() este inima programului Arduino și rulează continuu, actualizând starea sistemului în funcție de comenzile primite și condițiile curente. În această funcție, sistemul decide dacă să opereze în modul automat sau manual și ajustează poziția jaluzelei în consecință. Modul de noapte este tratat identic ca cel manual.

```
void loop() {
  server.handleClient();
  if (requestedPercentage == -1){
    moveMotorAUTOMATIC();
  }else {
    moveMotorMANUAL();
  }
}
```

Figura 3.21: Metoda loop a plăcii NodeMCU 1

Secțiunea de cod din figura 3.21 determină modul de operare al sistemului. Dacă requestedPercentage este -1, sistemul funcționează în modul automat, altfel funcționează în modul manual.

În modul automat, motorul se mișcă în funcție de comanda primită (open sau close). Funcția moveMotorAUTOMATIC() ajustează poziția jaluzelei și actualizează procentajul curent al deschiderii.

```
void moveMotorAUTOMATIC() {
  if (motorCommand == "open") {
    if (currentPosition + NUMBER_OF_STEPS <= maxSteps) {
      myStepper.step(NUMBER_OF_STEPS);
      currentPosition += NUMBER_OF_STEPS;
    }
  } else if (motorCommand == "close") {
    if (currentPosition - NUMBER_OF_STEPS >= 0) {
      myStepper.step(-NUMBER_OF_STEPS);
      currentPosition -= NUMBER_OF_STEPS;
    }
  }
  currentPercentage= (currentPosition * 100) / maxSteps;
  if (oldPercentage != currentPercentage){
    send_curtain_position();
    oldPercentage = currentPercentage;
  }
}
```

Figura 3.22: Operarea motorului în modul automat, de pe placa NodeMCU 1

Această funcţie, ilustrată în figura 3.22, mişcă motorul în paşi mici, adăugând sau scăzând numărul de paşi în funcţie de comanda primită. De asemenea, funcţia actualizează procentajul curent al deschiderii jaluzelei şi trimite această informaţie serverului.

În modul manual, motorul se mişcă pentru a ajunge la procentajul dorit al deschiderii jaluzelei. Funcţia moveMotorMANUAL() ajustează poziţia jaluzelei pentru a atinge procentajul specificat.

```
void moveMotorMANUAL() {
  if (currentPercentage < requestedPercentage){
    if (currentPosition + NUMBER_OF_STEPS <= maxSteps) {
      myStepper.step(NUMBER_OF_STEPS);
      currentPosition += NUMBER_OF_STEPS;
    }
  } else if (currentPercentage > requestedPercentage){
    if (currentPosition - NUMBER_OF_STEPS >= 0) {
      myStepper.step(-NUMBER_OF_STEPS);
      currentPosition -= NUMBER_OF_STEPS;
    }
  }
  currentPercentage= (currentPosition * 100) / maxSteps;
}
```

Figura 3.23: Operarea motorului în modul manual, de pe placa NodeMCU 1

Această funcție mișcă motorul până când procentajul curent al deschiderii jaluzelei corespunde procentajului dorit. Mișcarea se face în pași mici pentru a asigura o ajustare precisă.

Pentru a asigura o funcționare corectă a sistemului, informațiile despre poziția curentă a jaluzelei trebuie trimise periodic către server. Funcția send_curtain_position() se ocupă de acest lucru.

```
void send_curtain_position() {
  // Send the lightPercentage value
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String url = "http://" + String(serverIP) + "/motor?value=" + String(currentPercentage);
    http.begin(wifiClient, url);

    int httpCode = http.GET();

    if (httpCode > 0) {
      String payload = http.getString();
      Serial.println(payload);
    } else {
      Serial.println("Error on HTTP request");
      Serial.println(httpCode);
    }
    http.end();
  }
}
```

Figura 3.24: Metoda responsabilă pentru trimiterea poziției jaluzelei către serverul principal, NodeMCU 2

Această funcție construiește o solicitare HTTP GET și trimite informațiile despre procentajul curent al deschiderii jaluzelei către server. Dacă solicitarea este reușită, serverul primește aceste informații și le poate utiliza pentru a monitoriza și ajusta sistemul în continuare.

Pentru modulul LED, NodeMCU 2, codul începe cu includerea bibliotecilor necesare pentru funcționalitatea WiFi, server web și controlul LED-urilor. Apoi sunt definite variabilele globale pentru gestionarea stării sistemului. Toate acestea se pot observa în figura 3.25.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPClient.h>
#include "LedControl.h"

WiFiClient wifiClient;

const char* ssid = "DIGI-MA4c";
const char* password = "2N9UCsjDFk";
const char* motorIP = "192.168.1.138";

// Variabile de stare și control
int ambientLight = 0;              // [0-100]
int required_ambientLight = 0;     // [0-100]
String motor_command = "halt";     // open, close or halt
int LEDintensity = 0;              // [0-100]
int curtainPosition = 0;           // [0-100], real data from motor
int required_curtainPosition = 0; // [0-100], data from android app
String operatingMode = "none";     // none, manual, automatic, night
int led_intensity = 0;             // [-1 - 15]
const int TOLERANCE = 10;          // Toleranță pentru poziția jaluzelei
```

Figura 3.25: Bibliotecile și variabilele necesare pentru NodeMCU 2

Un obiect LedControl este creat pentru a gestiona matricea LED MAX7219, specificând pinii de conectare, precum in figura 3.26.

```
LedControl lc = LedControl(D0, D2, D1, 1);
byte full[8] = {B11111111, B11111111, B11111111, B11111111, B11111111, B11111111, B11111111, B11111111};
```

Figura 3.26: Variabila şi obiectul necesar controlului LED, NodeMCU 2

Serverul web este configurat pe portul 80 pentru a primi şi gestiona cererile HTTP, similar modului NodeMCU 1 responsabil de controlul motorului. Aceeaşi metodă este utilizată pentru conectarea la reţeaua Wi-Fi.

Funcţia led_setup() iniţializează matricea LED, setând intensitatea şi ştergând afişajul. Funcţia se poate observa în figura 3.26.

```
void led_setup() {
  lc.shutdown(0, false);
  lc.setIntensity(0, 1);
  lc.clearDisplay(0);
  for (int j = 0; j < 8; j++) {
    lc.setRow(0, j, full[j]);
  }
  led_intensity = 0;
}
```

Figura 3.26: Iniţializarea LED-ului, NodeMCU 2

Funcţiile send_motor_command() trimit comenzi către motorul pas cu pas, fie pentru a opri motorul, fie pentru a seta un anumit procentaj de deschidere a jaluzelei.

```
void send_motor_command(String new_motor_command) {
  if (new_motor_command != motor_command) {
    motor_command = new_motor_command;

    if (WiFi.status() == WL_CONNECTED) {
      HTTPClient http;
      String url = "http://" + String(motorIP) + "/motor?" +
                   "command=" + String(motor_command) +
                   "&percentage=" + String(-1);

      http.begin(wifiClient, url);
      int httpCode = http.GET();

      if (httpCode > 0) {
        String payload = http.getString();
        Serial.println(payload);
      } else {
        Serial.println("Error on HTTP request");
        Serial.println(httpCode);
      }
      http.end();
    }
  }
}
```

```
void send_motor_command(int percentage) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String url = "http://" + String(motorIP) + "/motor?" +
                 "command=" + String(motor_command) +
                 "&percentage=" + String(percentage);

    http.begin(wifiClient, url);
    int httpCode = http.GET();

    if (httpCode > 0) {
      String payload = http.getString();
      Serial.println(payload);
    } else {
      Serial.println("Error on HTTP request");
      Serial.println(httpCode);
    }
    http.end();
  }
}
```

Figura 3.27: Metodele folosite pentru trimiterea comenziilor către NodeMCU 1

Funcţia setup() iniţializează comunicaţia serială, conectarea la WiFi, configurarea matricei LED şi configurarea serverului web pentru gestionarea cererilor HTTP. Deoarece acest modul are rol de server principal, există mai multe rute, prezentate în Figura 3.28. Acestea se ocupă atât de comunicarea cu aplicaţia Android, cât şi cu restul modulelor.

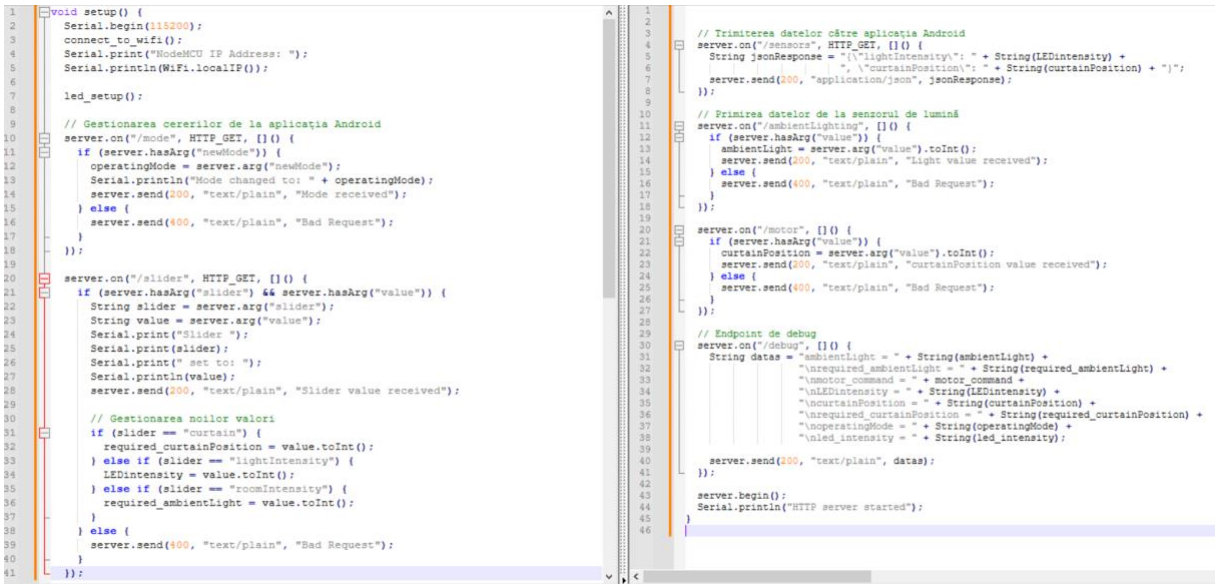
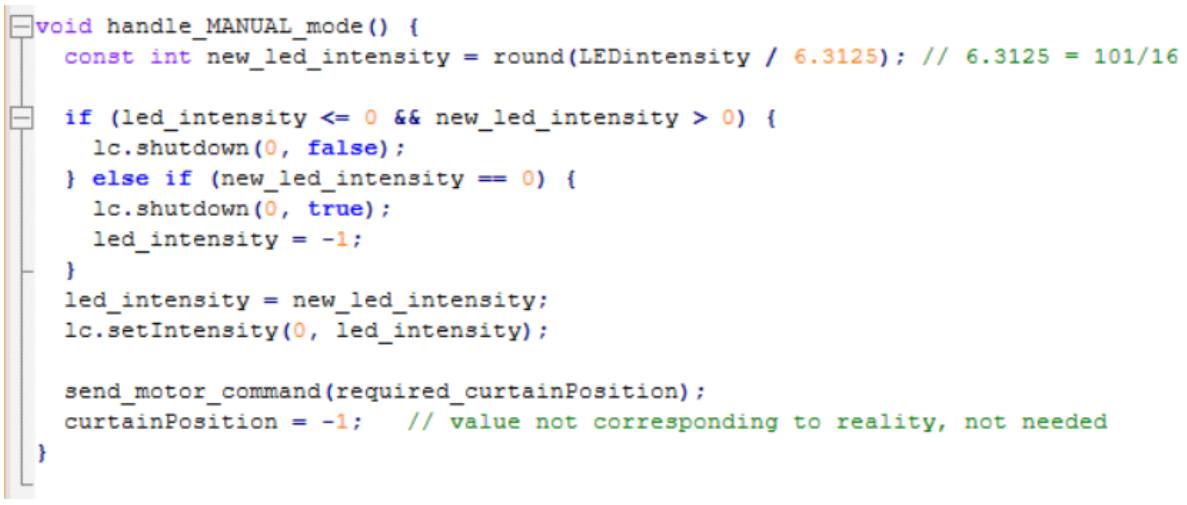


Figura 3.28: Metoda setup pentru serverul principal, NodeMCU 2, şi toate endpoint-urile definite

Funcţia loop() gestionează logica principală a sistemului, bazată pe modul de operare curent, şi este identică cu cea descrisă în etapa de proiectare. După cum am menţionat tot în etapa de proiectare, funcţia handle_NONE_mode reseteaza valorile variabilelor folosite.

Modul manual, reprezentat de metoda handle_MANUAL_mode permite utilizatorului să controleze manual poziţia jaluzelei şi intensitatea LED-ului, şi este ilustrat in figura 3.29.



```
void handle_MANUAL_mode() {
  const int new_led_intensity = round(LEDintensity / 6.3125); // 6.3125 = 101/16

  if (led_intensity <= 0 && new_led_intensity > 0) {
    lc.shutdown(0, false);
  } else if (new_led_intensity == 0) {
    lc.shutdown(0, true);
    led_intensity = -1;
  }
  led_intensity = new_led_intensity;
  lc.setIntensity(0, led_intensity);

  send_motor_command(required_curtainPosition);
  curtainPosition = -1;   // value not corresponding to reality, not needed
}
```

Figura 3.29: Metoda responsabilă de modul manual, din NodeMCU 2

Modul automat, reprezentat de metoda handle_AUTOMATIC_mode, ajustează poziţia jaluzelei şi intensitatea LED-ului pentru a menţine nivelul de lumină dorit. Aceasta este ilustrată in figura 3.30.

```
void handle_AUTOMATIC_mode() {
  if (is_value_tolerated(ambientLight, required_ambientLight)) {
    check_for_halt();
  } else if (ambientLight < required_ambientLight) {
    send_motor_command("close");
    up_LED_intensity();
  } else if (ambientLight > required_ambientLight) {
    send_motor_command("open");
    down_LED_intensity();
  }
}
```

Figura 3.30: Metoda responsabilă de modul automat, din NodeMCU 2

Modul de noapte, reprezentat de metoda handle_NIGHT_mode, închide complet jaluzeaua și stinge LED-ul. Se poate observa în figura 3.31.

```
void handle_NIGHT_mode() {
  down_LED_intensity();
  send_motor_command(100);
}
```

Figura 3.31: Metoda responsabilă de modul de noapte, din NodeMCU 2

Funcția check_for_halt() verifică necesitatea opririi jaluzelei și, în mod specific pentru modul automat de funcționare, prioritizează utilizarea luminii naturale în detrimentul celei artificiale. Este prezentată in figura 3.32.

```
void check_for_halt() {
  String upOrDown = (ambientLight <= required_ambientLight) ? "up" : "down";

  if (upOrDown == "up") {
    if (curtainPosition < 100) {
      if (led_intensity >= 0) {
        send_motor_command("open");
        down_LED_intensity();
      } else {
        Serial.print("Can't supply the required lighting");
        send_motor_command("halt");
      }
    } else {
      send_motor_command("halt");
    }
  } else {
    send_motor_command("halt");
  }
}
```

Figura 3.32: Metoda check_for_halt(), care favorizează lumina naturală

Funcțiile up_LED_intensity() și down_LED_intensity() ajustează intensitatea LED-ului, sunt prezente în figura 3.33.

```
void up_LED_intensity() {
  if (led_intensity == -1) {
    lc.shutdown(0, false);
    led_intensity = 0;
  }
  if (led_intensity <= 15) {
    lc.setIntensity(0, ++led_intensity);
  }
}

void down_LED_intensity() {
  if (led_intensity >= 0) {
    lc.setIntensity(0, --led_intensity);
  } else {
    lc.shutdown(0, true);
    led_intensity = -1;
  }
}
```

Figura 3.33: Metodele utilizate pentru creşterea sau diminuarea intensităţii luminii emise de LED.

Pentru modulul responsabil de citirea valorii luminii ambientale, NodeMCU 3, partea iniţială este mai simplă. Similar cu celelalte module NodeMCU, este implementată metoda de conectare la reţeaua WiFi şi sunt declarate variabilele necesare. De asemenea, au fost incluse bibliotecile necesare pentru funcţionalitatea WiFi şi gestionarea cererilor HTTP.

```
const char* serverIP = "192.168.1.140";  // Replace with the IP address of server NodeMC
const int lightSensorPin = A0;

// for ambientLighting
const int readingsTolerance = 10;
const int required_numberOfReadings = 10;
int numberOfReadings = 0;
float ambientLightValue[10];

void loop() {
  float lightValue = (analogRead(lightSensorPin))/1024.0 * 100;
  ambientLightValue[numberOfReadings++] = lightValue;

  if (numberOfReadings == required_numberOfReadings){
    // calculate the average value of lightPercentage
    float average_ambientLightValue = 0;
    for(int i = 0; i < numberOfReadings; ++i){
        average_ambientLightValue += ambientLightValue[i];
    }
    average_ambientLightValue /= required_numberOfReadings;
    average_ambientLightValue = round(average_ambientLightValue);

//    Serial.println(average_ambientLightValue);


    // remove odd values and calculate again
    for(int i = 0; i < numberOfReadings; ++i){
        if(abs(average_ambientLightValue - ambientLightValue[i] >= readingsTolerance)){
            ambientLightValue[i] = 0;
        }
    }
    average_ambientLightValue = 0;
    for(int i = 0; i < numberOfReadings; ++i){
        average_ambientLightValue += ambientLightValue[i];
    }

    average_ambientLightValue /= required_numberOfReadings;
    average_ambientLightValue = round(average_ambientLightValue);

    send_value_to_server();

    numberOfReadings = 0;
  }
}
```

Figura 3.34: Declararea variabilelor necesare şi citirea valorii luminii ambientale

Secvența de cod responsabilă pentru citirea, calcularea și conversia valorii luminii ambientale în procent este prezentată în Figura 3.34. De asemenea, pentru transmiterea valorii finale către serverul principal NodeMCU 2, se utilizează funcția send_value_to_server, ilustrată în Figura 3.35.

```
void send_value_to_server(){
    if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;
        String url = "http://" + String(serverIP) + "/ambientLighting?value=" + String(average_ambientLightValue);
        http.begin(wifiClient, url);

        Serial.println(url);

        int httpCode = http.GET();

        if (httpCode > 0) {
          String payload = http.getString();
          Serial.println(payload);
        } else {
          Serial.println("Error on HTTP request");
          Serial.println(httpCode);
        }
        http.end();
    }
}
```

Figura 3.35: Metoda utilizată de modulul NodeMCU 3 pentru trimiterea valorii finale către server

Pentru aplicația Android, deși am adoptat abordarea declarativă, lungimea codului este considerabilă. Prin urmare, voi evidenția doar secvențele importante și relevante pentru comunicare, figura 3.36, evitând detaliile legate de codul privind designul interfeței pentru utilizator.

```
@Composable
fun MainScreen(client: OkHttpClient, nodeMcuBaseUrl: String) {
    var activeMode by remember { mutableStateOf( value: "none") } // none, manual, automatic, night
    var showInfoDialog by remember { mutableStateOf( value: false) }
    var showSettingsDialog by remember { mutableStateOf( value: false) }
    var isNightMode by remember { mutableStateOf( value: false) }
    var isDarkTheme by remember { mutableStateOf( value: false) }
    var lastMode by remember { mutableStateOf( value: "none") } // none, manual, automatic

    // Example of switching modes
    fun send_new_mode(newMode: String) {
        if(newMode != activeMode) {
            val url = "$nodeMcuBaseUrl/mode?newMode=$newMode"
            println(url)
            val request = Request.Builder().url(url).build()
            client.newCall(request).enqueue(object : Callback {
                override fun onFailure(call: Call, e: IOException) { e.printStackTrace() }
                override fun onResponse(call: Call, response: Response) { println(response) }
            })
        }
    }

    fun send_slider_data(sliderName: String, newValue: Float) {
        val url = "$nodeMcuBaseUrl/slider?slider=$sliderName&value=${newValue.toInt()}"
        val request = Request.Builder().url(url).build()
        client.newCall(request).enqueue(object : Callback {
            override fun onFailure(call: Call, e: IOException) { e.printStackTrace() }
            override fun onResponse(call: Call, response: Response) { println(response) }
        })
    }

    MyApplicationTheme(darkTheme = isDarkTheme) {...}

}
```

Figura 3.36: Codul sursă relevant aplicației Android

Chiar dacă secvenţele de cod responsabile de designul interfeţei pentru utilizator nu sunt relevante, rezultatul final este esenţial pentru demonstrarea interfeţei prietenoase create. În Figura 3.37 se pot observa două dintre cele trei moduri de funcţionare: manual şi automat. De asemenea, se poate vedea că, la pornirea aplicaţiei, niciun mod nu este activ. Din acest motiv, a fost necesară crearea metodei handle_NONE_mode pentru ca sistemul să aibă o stare iniţială de repaus.

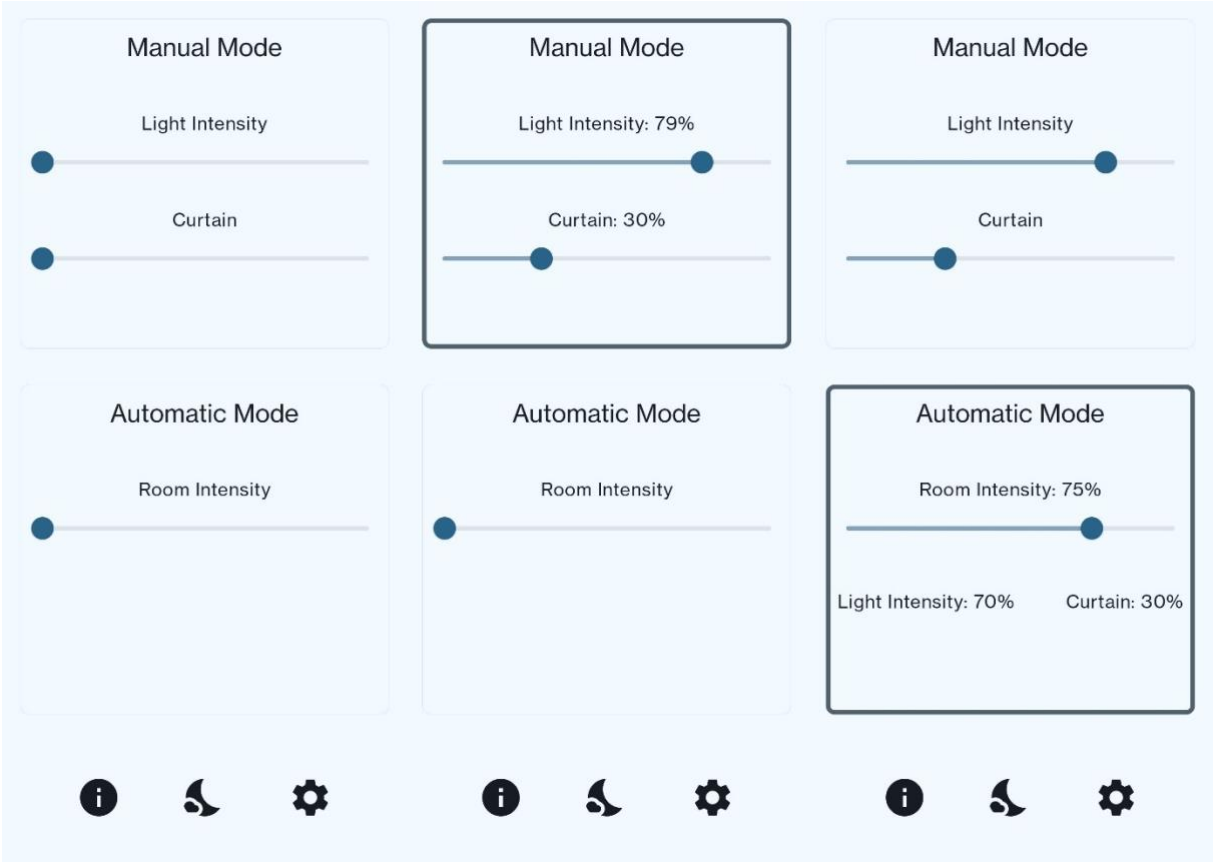


Figura 3.37: De la stânga la dreapta, aplicaţia la pornire, selectarea modului de funcţionare manual şi ajustarea glisoarelor la valorile dorite, selectarea modului automat şi ajustarea glisorului

Se poate observa actualizarea în timp real a câmpurilor „Light Intensity” şi „Curtain” cu date reale transmise de către NodeMCU 2, precum şi păstrarea valorilor glisoarelor, chiar şi atunci când modul de operare este schimbat.

În Figura 3.36 sunt prezentate funcţionalităţile butoanelor situate în partea inferioară a ecranului. Aceste butoane sunt responsabile pentru afişarea informaţiilor privind funcţionarea modurilor de operare, activarea modului de noapte şi, întrucât este o aplicaţie modernă, comutarea între tema întunecată şi tema normală, cu care suntem cu toţii obişnuiţi.

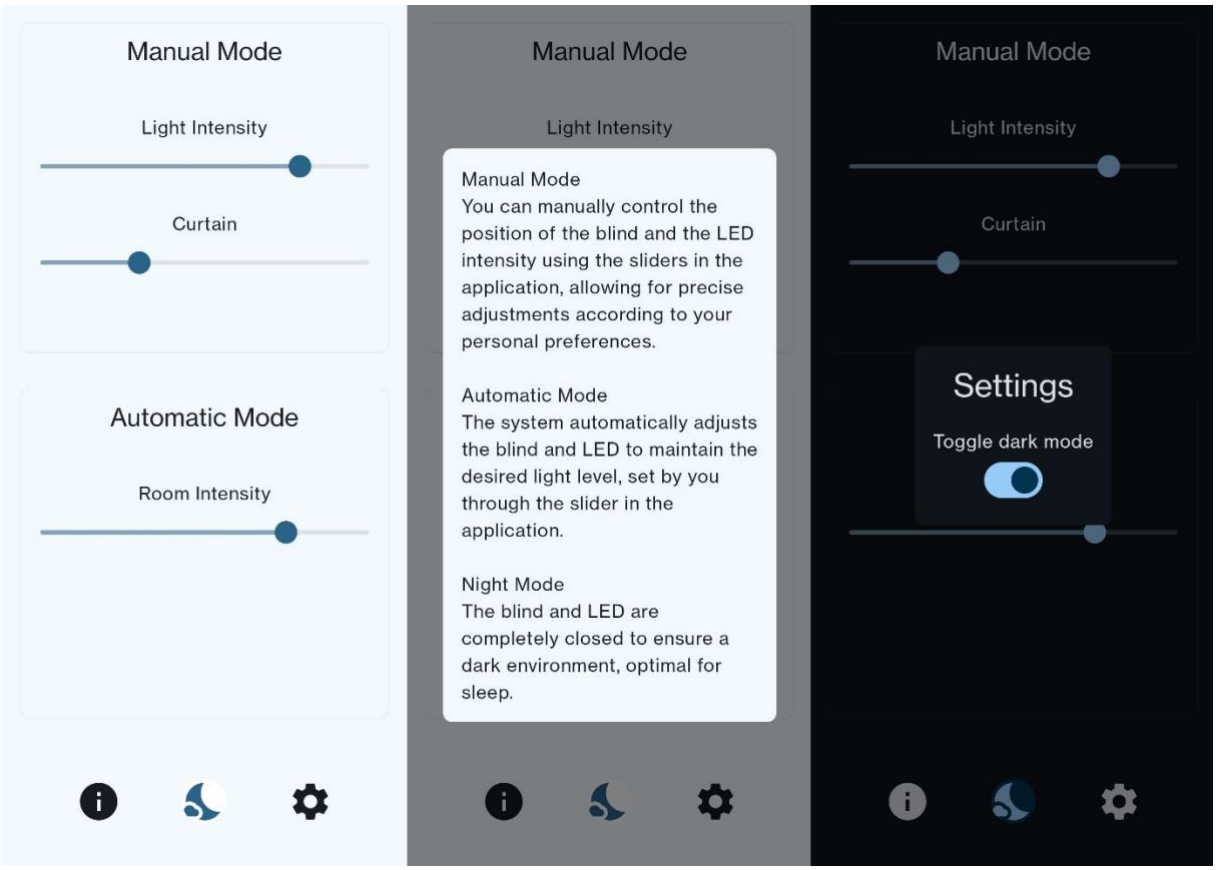


Figura 3.36: De la stânga la dreapta: activarea modului de noapte, declanşarea căsuţei cu informaţii despre modurile de funcţionare şi comutarea între tema întunecată şi cea normală

Implementarea software descrisă mai sus respectă toate cerinţele stabilite în analiza iniţială şi proiectarea sistemului. Fiecare componentă este integrată şi configurată conform specificaţiilor, asigurând funcţionarea la standarde înalte.

Testarea a fost realizată atât individual pentru fiecare modul, cât şi colectiv, în mod practic. Aceasta a inclus testarea efectivă a modurilor de funcţionare şi a dinamicii sistemului.

4 Concluzii

4.1 Rezultate obţinute

În cadrul acestui proiect, am dezvoltat un sistem automatizat de control al jaluzelelor, integrând mai multe componente hardware şi software pentru a oferi utilizatorilor o soluţie inteligentă şi eficientă pentru gestionarea luminii în locuinţă. Sistemul a fost conceput şi implementat respectând cerinţele iniţiale, iar rezultatele obţinute sunt remarcabile.

În introducerea lucrării, am subliniat importanţa confortului şi eficienţei energetice în locuinţele moderne. Automatizarea locuinţelor, şi în special a sistemelor de iluminat, reprezintă un pas esenţial în atingerea acestor obiective. Prin dezvoltarea unui sistem de control automatizat al jaluzelelor, am dorit să demonstrăm cum tehnologiile moderne pot fi utilizate pentru a îmbunătăţi calitatea vieţii şi a reduce consumul de energie.

Studiul bibliografic a evidenţiat progresul tehnologic în domeniul automatizării locuinţelor şi utilizarea plăcilor de dezvoltare precum NodeMCU, a motorului pas cu pas 28BYJ-48, a driverului ULN2003, a matricei LED MAX7219 şi a senzorului de lumină. Am analizat şi utilizarea bibliotecilor software specifice pentru a facilita implementarea acestor componente. Aceste studii ne-au oferit o bază solidă pentru proiectarea şi implementarea sistemului nostru, demonstrând fezabilitatea şi eficienţa utilizării acestor tehnologii.

În etapa de analiză şi proiectare, am stabilit cerinţele hardware şi software pentru sistemul nostru. Am definit modul în care diferitele componente hardware vor fi interconectate şi am selectat tehnologiile adecvate pentru dezvoltarea software. Proiectarea a inclus schema detaliată a conexiunilor şi structura codului necesar pentru a controla fiecare componentă. Această etapă a fost esenţială pentru a asigura că toate părţile sistemului vor funcţiona armonios şi eficient împreună.

Implementarea sistemului a fost realizată cu succes, conform planului stabilit în etapa de proiectare. Am conectat motorul pas cu pas 28BYJ-48 la NodeMCU prin intermediul driverului ULN2003, asigurând controlul precis al poziţiei jaluzelei. Matricea LED MAX7219 a fost conectată la o altă placă NodeMCU pentru a suplimenta lumina atunci când este necesar, iar senzorul de lumină a furnizat date esenţiale pentru ajustarea automată a luminii. Aplicaţia Android dezvoltată cu Kotlin şi Jetpack Compose a oferit o interfaţă intuitivă pentru utilizator, permiţând controlul de la distanţă al sistemului.

Implementarea unui sistem automatizat de control al jaluzelelor, aşa cum a fost descris în acest proiect, reprezintă un exemplu clar al modului în care tehnologiile moderne pot fi utilizate pentru a îmbunătăţi calitatea vieţii şi a eficienţei energetice în locuinţe. Prin integrarea componentelor hardware şi software, şi respectarea analizei şi proiectării detaliate, am reuşit să dezvolt un sistem robust şi eficient, capabil să răspundă nevoilor utilizatorilor.

Rezultatele obţinute evidenţiază controlul precis al jaluzelei prin utilizarea motorului pas cu pas 28BYJ-48, gestionat de NodeMCU, demonstrând capacitatea de ajustare exactă a poziţiei. Senzorul de lumină a oferit date precise, permiţând ajustarea automată a poziţiei jaluzelei şi a intensităţii LED-ului pentru menţinerea unui nivel optim de iluminare. Aplicaţia Android a facilitat interacţiunea intuitivă, oferind moduri de operare variate. Sistemul a demonstrat potenţialul de economisire a energiei prin utilizarea eficientă a luminii naturale. Contribuţia mea include dezvoltarea unui sistem complet de control, integrarea eficientă a componentelor hardware şi software, şi implementarea unei interfeţe de utilizator moderne.

4.2 Direcţii de dezvoltare

Îmbunătăţirea sistemului prin utilizarea inteligenţei artificiale (AI) poate aduce multiple beneficii, inclusiv integrarea algoritmilor de învăţare automată pentru a ajusta automat jaluzelele şi LED-urile, învăţând din comportamentul utilizatorilor şi condiţiile de iluminare. De asemenea, AI poate anticipa nevoile utilizatorilor pe baza istoricului utilizării şi a modelelor de comportament, ajustând proactiv jaluzelele şi iluminatul pentru a maximiza confortul şi eficienţa energetică. Algoritmii AI pot optimiza consumul de energie prin ajustarea dinamică a luminii în funcţie de condiţiile externe şi prezenţa utilizatorilor în cameră, reducând astfel consumul inutil de energie.

Extinderea funcţionalităţilor sistemului poate fi realizată prin integrarea cu alte dispozitive inteligente din locuinţă, precum termostate inteligente, sisteme de securitate şi dispozitive IoT, creând astfel un mediu de locuit complet automatizat şi interconectat. Adăugarea controlului vocal, prin asistenţi virtuali precum Amazon Alexa sau Google Assistant, poate îmbunătăţi experienţa utilizatorilor, oferind un control mai uşor şi mai intuitiv al sistemului. Permiţând utilizatorilor să programeze scenarii complexe şi reguli personalizate pentru ajustarea jaluzelelor şi a luminii, sistemul devine şi mai flexibil şi adaptabil la nevoile individuale.

Îmbunătăţirea securităţii datelor poate fi realizată prin implementarea unor măsuri avansate, cum ar fi criptarea comunicaţiilor şi autentificarea utilizatorilor, protejând astfel datele sensibile şi asigurând confidenţialitatea şi integritatea sistemului. Adăugarea unor mecanisme de redundanţă şi recuperare în caz de defecţiune poate îmbunătăţi fiabilitatea sistemului, asigurând continuitatea funcţionării acestuia în cazul unor defecţiuni hardware sau software.

Concluziile constatate demonstrează viabilitatea şi eficienţa sistemului, în timp ce direcţiile de dezvoltare propuse oferă o viziune clară asupra posibilităţilor de îmbunătăţire şi extindere a funcţionalităţilor acestuia. Integrarea inteligenţei artificiale şi explorarea noilor tehnologii vor juca un rol crucial în evoluţia viitoare a sistemelor de automatizare a locuinţelor, deschizând calea către case inteligente, mai confortabile şi mai eficiente din punct de vedere energetic.

5 Bibliografie

[1] „Adopting Internet of Things for the development of smart buildings: A review of enabling technologies and applications," *Mengda Jia,* 2018.

[2] B. A. Prabowo, „A Review on Light Source Technology," 2018.

[3] V. Athani, Stepper motors: fundamentals, applications and design, 1997.

[4] Y. S. Parihar, „Internet of Things and Nodemcu: A review of use of Nodemcu ESP8266 in IoT products," *JETIR,* vol. 6, nr. 6, p. 4, 2019.

[5] S. Monk, Programming Arduino: getting started with sketches, 2016.

[6] „Stepper.h library," [Interactiv]. Available: https://www.arduino.cc/reference/en/libraries/stepper/.

[7] „ESP8266WiFi.h library," [Interactiv]. Available: https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/src/ESP8266WiFi.h.

[8] „ESP8266WebServer.h library," [Interactiv]. Available: https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WebServer/src/ESP8266WebServer.h.

[9] „ESP8266HTTPClient.h library," [Interactiv]. Available: https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266HTTPClient/src/ESP8266HTTPClient.h.

[10] „LedControl.h library," [Interactiv]. Available: https://www.arduino.cc/reference/en/libraries/ledcontrol/.

[11] S. I. D Jemerov, Kotlin in action, 2017.

[12] „3d print," [Interactiv]. Available: https://github.com/asafteirobert/automatic-roller-blinds-motor/tree/master/3d%20print.

[13] P. TURGUT, „Imperative vs. Declarative UI Development with Jetpack Compose," [Interactiv]. Available: https://pinarturgut09.medium.com/imperative-vs-declarative-ui-development-with-jetpack-compose-d5556cfbefc8.