



Sistem smart house controlat de la distanță

LUCRARE DE LICENȚĂ

Absolvent: **Mihai Gligor**

Coordonator științific: **Conf. Dr. Ing. Adrian PECULEA**

2023

Cuprins

Capitolul 1. Introducere	1
Capitolul 2. Obiectivele proiectului.....	3
Capitolul 3. Studiu bibliografic.....	5
3.1. Smart homes	5
3.2. Internet of Things si integrarea acestuia in sistemele smart home	6
Capitolul 4. Analiză și fundamentare Teoretică	8
4.1. Cazuri de utilizare.....	8
4.1.1. Autentificare	10
4.1.2. Vizualizare parametrii	10
4.1.3. Control încuietori.....	11
4.1.4. Control ferestre	11
4.1.5. Control lumini.....	11
4.1.6. Activare/dezactivare senzori.....	12
4.1.7. Setare prag de activare a alarmei pentru detectorul de fum.....	12
4.1.8. Afisare mesaj pe LCD	12
4.2. Protocoale de comunicare utilizate	13
4.2.1. Protocol de comunicare intre hardware si platforma web	13
4.2.2. Protocoale de comunicare intre module hardware	14
4.3. Componente fizice utilizate	16
4.3.1. Arduino Uno	16
4.3.2. Lolin NodeMCU ESP8266	20
4.3.3. Motor Micro Servo SG-90.....	21
4.3.4. Senzorul ultrasonic HC-SR04	22
4.3.5. Senzorul de fum si gaz MQ-2.....	23
4.3.6. Senzorul de temperatura LM35	23
4.3.7. Ecran LCD 20x4 2004A	24
4.3.8. Modul Serial I2C	24
4.3.9. Buzzer pasiv	25
4.3.10. Fire de legătură (Jumper wires)	25
4.3.11. LED	26
4.3.12. Rezistenta.....	26

4.4. Tehnologii utilizate	28
4.4.1. Arduino IDE	28
4.4.2. Visual Studio Code	29
4.4.3. Hostinger	30
4.5. Soluția propusă	32
4.5.1. Perifericele	32
4.5.2. Modulele hardware	32
4.5.3. Platforma web	33
4.5.4. Baza de date	33

Capitolul 5. Proiectare de detaliu și implementare.....34

5.1. Transmiterea datelor	34
5.1.1. Comunicarea între platforma web și microcontrolerul ESP8266	34
5.1.2. Comunicarea între cele două microcontrolere	36
5.2. Procesarea datelor	40
5.2.1. Procesarea informațiilor pe platforma web	40
5.2.2. Procesarea informațiilor în microcontrolere	42
5.3. Funcționalitățile fizice ale sistemului	44
5.3.1. Controlul încuietorilor și ferestrelor	44
5.3.2. Alarma	45
5.3.3. Monitorizarea proximității	46
5.3.4. Monitorizarea calității aerului	48
5.3.5. Monitorizarea confortului termic	49
5.3.6. Trimiterea mesajelor	50
5.3.7. Controlul iluminării imobilului	51
5.4. Soluția propusă	52
5.4.1. Perifericele	52
5.4.2. Modulele hardware	53
5.4.3. Platforma web	53
5.4.4. Baza de date	53

Capitolul 6. Testare și validare55

6.1. Testarea funcționalităților propuse	55
6.1.1. Autentificare	55
6.1.2. Testarea motoarelor servo	59
6.1.3. Testarea alarmelor	60
6.1.4. Testarea controlului luminilor	60
6.1.5. Testarea trimiterii mesajelor	60

6.1.6. Testarea monitorizării temperaturii	61
6.2. Componente si funcționalități testate care nu au fost introduse in proiectul final	61
6.3. Motivația alegerii configurației curente.....	62
Capitolul 7. Manual de instalare si utilizare	63
Capitolul 8. Concluzii.....	68
8.1. Rezultate obținute	68
8.2. Dezvoltări ulterioare	68
Bibliografie.....	69

Capitolul 1. Introducere

Sistemul smart house controlat de la distanță este un proiect inovator care abordează problema costurilor ridicate asociate sistemelor smart home. Proiectul a fost inițiat ca o lucrare de licență, dar poate fi adaptat la nevoile individuale ale oricărui potențial utilizator, în special entuziaștilor și studenților cu profil tehnic.

Componentele utilizate în proiect sunt alese astfel încât să fie accesibile din punct de vedere financiar, fără a compromite funcționalitatea și performanța. Acest fapt oferă posibilitatea de a crea un sistem smart home cu costuri reduse, fără a sacrifica calitatea sau capacitatea de control.

Sistemul smart house controlat de la distanță poate fi adaptat în funcție de cerințele fiecărui utilizator. Prin adăugarea și integrarea de module suplimentare, utilizatorii pot extinde funcționalitatea sistemului în diferite moduri care corespund cu viziunea lor asupra controlului imobilului.

Proiectul este compus din două părți principale, hardware-ul și platforma online de gestionare a datelor și comenzilor. Pe scurt pentru partea hardware a lucrurilor, sunt utilizate două sisteme esențiale: Arduino Uno și NodeMCU ESP8266. Placa Arduino Uno este o platformă de dezvoltare care se bazează pe un microcontroler, care permite interacțiunea cu diverse componente și senzori. Este folosit pentru reglarea și supravegherea diverselor periferice din casă, cum ar fi sistemele de securitate, detectoarele de fum, senzorii de temperatură și senzorii de proximitate. Placa NodeMCU ESP8266 este o altă platformă de dezvoltare care controlează diverse aspecte ale sistemului, cum ar fi luminile și alarmele. Cu toate acestea, scopul său principal este acela de a asigura conectivitatea prin WiFi pentru a transmite date cruciale către platformele online. Microcontrolerele comunică între ele schimbând informații și comenzi printr-o conexiune serială.

Platforma online este găzduită de serviciul de hosting Hostinger¹. Aceasta se ocupă de managementul datelor colectate de la diferitele periferice, de procesarea acestora și de trimiterea comenzilor înspre partea fizică. Platforma are rolul de a memora datele relevante prin intermediul unei baze de date și de a facilita interacțiunea utilizatorului cu sistemul smart house. Prin intermediul unei interfețe simple și intuitive compuse din diferite butoane, textbox-uri și secțiuni pentru afisarea informațiilor, utilizatorii controlează și monitorizează diferite aspecte ale locuinței lor, precum temperatura, securitatea sau iluminarea în diferite părți ale imobilului.

Prin integrarea hardware-ului și a platformei online în același pachet, sistemul smart house controlat de la distanță oferă o soluție completă și eficientă pentru gestionarea și controlul locuinței. Utilizatorii pot beneficia de funcționalități, precum monitorizarea în timp real și control de la distanță, totul prin intermediul unei configurații simple de implementat și a unei platforme online ușor de folosit.

Aceste funcționalități pot să fie folosite de către una sau mai multe persoane care au acces la platforma online care găzduiește interfața de control.

Alternativele existente, precum SmartThings² sau Home Assistant³ oferă funcționalități mai complexe precum comenzi vocale, aplicații dedicate, sisteme modulare compatibile cu diverse produse existente deja pe piață. Însă problema pe care o aduc acestea din punct de vedere al prețului întregului sistem și din punct de vedere al invadării spațiului

¹ <https://www.hostinger.ro/>

² <https://www.smarthings.com/>

³ <https://www.home-assistant.io/>

personal duc la o creștere continuă a cererii pe piață a unei soluții pentru care Sistemul Smart House controlat de la distanță este construit.

Motivația din spatele acestui proiect este reprezentată de către dorința de a experimenta cu un proiect care conține atât parte software cât și parte hardware, și de a crea o conexiune între cele două, astfel aplicând o gamă mai vastă de cunoștințe dobândite pe parcursul celor 4 ani de studenție.

O altă motivație este reprezentată de dorința de a dezvolta o soluție inovatoare și accesibilă pentru controlul și automatizarea locuințelor. Astăzi, sistemele smart home sunt tot mai populare și cerute, însă mulți utilizatori se confruntă cu costuri ridicate asociate cu implementarea acestor tehnologii în casele lor. Prin urmare, am ales să abordez această problemă prin dezvoltarea unui sistem smart home cu costuri reduse, fără a compromite calitatea și funcționalitatea.

Sistemele de control și automatizare cunoscute sub numele de "Smart Home" au devenit o realitate tot mai prezentă în locuințele moderne. Această dezvoltare a avut loc în concordanță cu creșterea nevoilor și cerințelor proprietarilor de locuințe, care doresc să aibă un control mai mare asupra mediului în care trăiesc.

Prin intermediul tehnologiei moderne, instalațiile electrice convenționale sunt înlocuite treptat cu sisteme interconectate și inteligente, capabile să analizeze diferiți parametri ai unei clădiri și să ia decizii în funcție de aceștia. Aceste sisteme pot fi programate să ofere un nivel ridicat de automatizare, permițând locatarilor să controleze diverse aspecte ale locuinței lor, cum ar fi iluminatul, temperatura, siguranța și multe altele.

Un exemplu practic al unui Smart Home ar putea fi următorul: sistemul poate detecta prezența unei persoane într-o cameră și poate ajusta automat temperatura ambientală la un nivel confortabil. De asemenea, poate monitoriza și gestiona consumul de energie, reducând astfel costurile și având un impact pozitiv asupra mediului.

Dezvoltarea sistemelor de control și automatizare pentru Smart Home continuă într-un ritm accelerat. Tehnologii precum Internet of Things (IoT), inteligența artificială și machine learning contribuie la îmbunătățirea funcționalității și a eficienței acestor sisteme.

Unul dintre aspectele cheie ale sistemelor smart house este controlul de la distanță. Prin utilizarea dispozitivelor mobile, a computerelor sau a altor dispozitive conectate la internet, utilizatorii pot accesa și controla diferite echipamente și sisteme din casă.

Un subdomeniu important în crearea acestor produse este cel al automatizării. Sistemele smart house pot fi programate să desfășoare acțiuni în mod automat în funcție de anumite condiții sau evenimente. De exemplu, pot ajusta automat temperatura ambientală în funcție de prezența sau absența persoanelor într-o cameră sau pot activa luminile exterioare atunci când senzorii detectează că se întunecă.

Securitatea este un alt aspect crucial în domeniul smart house. Sistemele de securitate pot include camere de supraveghere, alarme, senzori de mișcare și de fum, acces controlat prin intermediul codurilor sau al amprentelor digitale și alte tehnologii de protecție. Acestea asigură siguranța locuinței și a locatarilor, iar utilizatorii pot monitoriza și controla aceste sisteme de la distanță.

Eficiența energetică și gestionarea resurselor sunt, de asemenea, aspecte semnificative pentru produsele de tip smart house. Sistemele pot monitoriza și controla consumul de energie al diferitelor echipamente și pot oferi recomandări pentru a reduce pierderile și a optimiza utilizarea resurselor.

Capitolul 2. Obiectivele proiectului

Scopul acestui proiect constă în crearea unui sistem de tip smart house simplu și accesibil, care să ofere o experiență ușoară și convenabilă utilizatorilor. Prin utilizarea componentelor accesibile din punct de vedere al disponibilității și costului, proiectul își propune să elimine barierele financiare și să ofere posibilitatea unui număr mai mare de persoane să beneficieze de avantajele unui sistem smart home.

Un aspect esențial al acestui proiect este ușurința cu care poate fi utilizat. Sistemul smart house este proiectat astfel încât să fie intuitiv, permițând utilizatorilor, indiferent de nivelul de experiență tehnică să interacționeze cu el fără dificultăți. Prin crearea unei interfețe simplificate și prin furnizarea unei documentații adecvate, se facilitează înțelegerea și utilizarea sistemului, eliminând astfel necesitatea unei expertize avansate în domeniul tehnologic.

Obiectivul principal al sistemului este de a se concentra exclusiv pe implementarea funcționalităților de bază, ceea ce duce la reducerea nivelului de complexitate. Alegerea de a se axa doar pe lucrurile esențiale are mai multe avantaje.

În primul rând, aceasta permite simplificarea procesului de creare, instalare și întreținere a sistemului smart house. Prin eliminarea funcționalităților complexe și avansate, se reduce numărul de componente și conexiuni necesare, ceea ce duce la o implementare mai rapidă și mai eficientă. Astfel, se reduc și costurile asociate cu achiziționarea și instalarea echipamentelor.

În al doilea rând, fiind un sistem simplu și ușor de înțeles, acesta nu necesită o expertiză tehnică avansată pentru a fi utilizat. Utilizatorii pot beneficia de avantajele sistemului smart home fără a se simți copleșiți de complexitatea tehnologică sau nevoia de a depune efort pentru a învăța cum funcționează sistemul. De asemenea, eliminarea funcționalităților complexe reduce și riscul de erori și probleme tehnice. Cu un sistem simplu și cu mai puține componente și interconexiuni, există mai puține puncte potențiale de eșec sau de incompatibilitate între diferitele elemente ale sistemului.

Cu toate acestea, sistemul dispune de suficiente funcționalități care să îi permită să îndeplinească o gamă largă acțiuni folositoare. Deși se axează pe funcționalități de bază, acestea sunt concepute astfel încât să ofere utilizatorilor un set util și variat de opțiuni.

Obiectivele sistemului din punct de vedere a cerințelor funcționale sunt monitorizarea, controlul și automatizarea.

Prin monitorizare, sistemul are ca obiectiv să ofere utilizatorilor informații în timp real despre diversele aspecte ale locuinței, cum ar fi temperatura ambientală, nivelul de dioxid de carbon, starea încuietorilor, starea luminilor din casă și proximitatea unor obiecte sau persoane față de ușă. Aceste informații furnizate în timp real permit utilizatorilor să aibă o înțelegere clară a situației și să ia decizii informate în ceea ce privește confortul, securitatea și starea locuinței lor. Prin monitorizarea temperaturii ambientale, utilizatorii pot fi conștienți de nivelul de confort termic din interiorul locuinței. Monitorizarea nivelului de dioxid de carbon este importantă pentru a asigura un mediu sănătos în interiorul casei, iar utilizatorii pot lua măsuri pentru aerisi spațiul în cazul în care este depășit nivelul maxim recomandat. Starea încuietorilor și senzorul de proximitate pot fi monitorizate pentru a oferi utilizatorilor informații despre accesul în locuință, precum și pentru a detecta orice activități suspecte sau neautorizate. Monitorizarea stării luminilor permite utilizatorilor să știe dacă luminile sunt aprinse sau stinse în diferitele camere și să le controleze de la distanță în funcție de nevoie.

Controlul reprezintă capacitatea sistemului de a permite utilizatorilor să gestioneze diferitele dispozitive și să efectueze acțiuni specifice în locuință, cum ar fi controlul iluminatului, reglarea temperaturii sau activarea/dezactivarea sistemului de alarmă. Astfel, utilizatorii pot avea control total asupra mediului lor de locuit. Automatizarea în cadrul proiectului de bază se referă la activarea automată a alarmei în două situații. Prima situație este atunci când senzorul de proximitate detectează apropierea cuiva în timp ce sistemul de securitate este activat. În acest caz, sistemul va declanșa imediat alarma pentru a alerta utilizatorii despre potențiala intruziune sau activitate suspectă. A doua situație este atunci când nivelul de dioxid de carbon depășește un anumit prag. În acest caz, sistemul va declanșa alarma pentru a avertiza utilizatorii despre această situație și pentru a le permite să ia măsuri imediate pentru a îmbunătăți calitatea aerului din interior.

Obiectivele sistemului din punct de vedere a cerințelor non funcționale sunt utilitatea, performanța, stabilitatea și fiabilitatea.

Utilitatea se referă la ușurința cu care poate fi utilizat sistemul de către oricine. Acest lucru înseamnă că utilizatorii ar trebui să aibă acces rapid și ușor la toate funcțiile și comenzile sistemului prin intermediul interfeței. Butoanele, meniurile și controalele ar trebui să fie plasate strategic și să fie ușor de găsit și utilizat. Utilizatorii ar trebui să poată accesa și controla luminile, temperatura, încuietorile, alarmele și alte dispozitive sau funcționalități ale sistemului cu doar câteva apăsări. Interfața ar trebui să fie atât de intuitivă și prietenoasă încât orice utilizator, indiferent de nivelul său de experiență tehnică, să poată interacționa cu sistemul fără dificultăți. Utilizatorii nu ar trebui să fie nevoiți să parcurgă manuale sau să primească instrucțiuni complexe pentru a utiliza sistemul. Operațiunile ar trebui să fie clare ușor de înțeles și simplu de pus în practică.

Performanța este un element cheie al sistemului smart house și se referă la rapiditatea și eficiența cu care acesta răspunde la comenzile utilizatorilor și furnizează informații relevante. Componentele relativ simple ale sistemului ajută la reducerea timpului de răspuns. Cu toate acestea, mai există modalități prin care performanța poate fi îmbunătățită. Aceasta poate fi realizată prin optimizarea codului și a algoritmilor utilizați pentru procesarea comenzilor și actualizarea informațiilor. Stabilizarea sistemului reprezintă, de asemenea, un aspect crucial în obținerea unei performanțe ridicate.

Un sistem smart house stabil și fiabil asigură utilizatorilor o funcționare corectă și lipsită de probleme. Aceasta înseamnă că dispozitivele și perifericele conectate la sistem trebuie să fie compatibile și să lucreze în mod armonios împreună, evitând erorile și problemele tehnice. De asemenea, sistemul trebuie să fie capabil să facă față sarcinilor și cerințelor utilizatorilor într-un mod constant. Indiferent dacă este vorba de controlul iluminatului, gestionarea temperaturii sau monitorizarea securității, sistemul smart house ar trebui să ofere performanță și stabilitate pe termen lung.

Sistemul smart house este conceput să fie accesibil utilizatorilor din orice loc, prin intermediul unei pagini web dedicate. Această pagină poate fi accesată prin intermediul unui browser web, atâta timp cât există o conexiune la internet și utilizatorul dispune de un cont valid. Prin intermediul paginii web, utilizatorii pot accesa diverse funcționalități și setări ale sistemului smart house.

Capitolul 3. Studiu bibliografic

Studiul acestui domeniu s-a rezumat în mare la acumularea informațiilor referitoare la două subiecte importante și anume Smart Homes și, ceea ce poate fi considerat un subdomeniu al acestui subiect, IoT (Internet of Things). Cele două susțin în unison fundamentul creării proiectului curent.

3.1. Smart homes

Proiectarea unui sistem Smart House este o temă extrem de actuală, care a fost intens studiată și abordată în diferite studii, cărți și articole de specialitate. Pentru a realiza o lucrare legată de proiectarea unui sistem Smart House, a fost necesar să se efectueze o documentare îndelungată și să se studieze materiale bibliografice diverse și relevante. Accesul la aceste surse de informații a fost ușor de obținut, având în vedere resursele disponibile în prezent. Această cercetare a fost efectuată pentru a avea o înțelegere mai profundă a subiectului abordat și a proiectării sistemului Smart House în cadrul lucrării propuse.

Datorită faptului că aplicația propusă în lucrarea de față se bazează pe un controler Arduino este necesar să se menționeze un articol din 2014 [1] care face referire la sistemele SH mici care utilizează acest tip de controller. Sistemul propus de autori este capabil să monitorizeze și să controleze luminile, temperatura din interior, alarmele și alte dispozitive electronice de menaj, iar funcționarea acestuia este gestionată de un dispozitiv conectat la o rețea care suportă HTML5. Monitorizarea temperaturii este realizată cu același senzor prezent în lucrarea de față adică LM35, cu diferența că implementarea din sistemul prezent în articol prezintă și un ventilator care este activat în funcție de confortul termic actual.

Într-un articol publicat în 2004 [2], s-a realizat un scurt rezumat al cercetărilor efectuate până în acel moment în domeniul „smart home networks” și s-a evidențiat stadiul acestora. Astfel această lucrare a contribuit la conturarea viitoarelor modele de îngrijire și gestionare a locuinței, cunoscute sub denumirea de sisteme Smart Home. Grupurile de cercetare de la MIT, Siemens, Cisco, IBM, Xerox sau Microsoft menționate în articol au utilizat până și tehnici de inteligență artificială (AI) în proiectarea și monitorizarea programelor și rețelelor propuse. Acest nivel de complexitate, deși depășește domeniul propus pentru această lucrare, este o viitoare posibilă adădire la următoarele iterații ale proiectului.

În ceea ce privește sistemele integrale Smart House, fiecare dintre companiile care dețin grupuri de cercetare au elaborat câte un proiect propriu. Lucrarea din referință îi menționează pe cei de la Cisco cu Internet Home, pe cei de la Colorado Boulder cu Adaptive House, pe KTH cu comHOME, pe Microsoft cu EasyLiving, grupul de la MIT cu House of the Future și pe cei de la Philips cu proiectul House of the Near Future, Smart Connections.

Într-un studiu realizat în 2006 [3], s-au explorat principiile de utilizare a sistemelor Smart Home (SH) din perspectiva familiilor, luând în considerare obiceiurile, activitățile zilnice și nivelul de cunoștințe al acestora. Un aspect important abordat în studiu a fost diferența între programarea unui sistem pentru un utilizator final comparativ cu instituția familială. O altă temă abordată a fost reprezentată de discrepanța între nevoia de control asupra dispozitivelor integrate de obicei în sistemele SH și nevoia de control asupra propriei vieți în general, care reprezintă o dorință fundamentală a oricărei persoane. Acest studiu susține că există un prag care odată depășit transformă confortul unui smart home dintr-un ajutor menit să crească nivelul de trai al utilizatorului într-un element care îl determină pe acesta să se simtă

„la mila” sistemului. Astfel, pentru a satisface nevoile utilizatorului, sistemele SH ar trebui să fie capabile să detecteze, să anticipeze și să răspundă la activitățile desfășurate în locuință, fără a invada spațiul privat al acestuia prin a se rezuma la capacități folositoare dar non-invazive. Acest aspect este integrat și în lucrarea prezentă.

3.2. Internet of Things si integrarea acestuia in sistemele smart home

În [4] autorii descriu noțiunea de Smart Home (SH) ca fiind o "aplicație de tip IoT (Internet of Things) care utilizează Internetul pentru a monitoriza și controla aparaturile casnice printr-un sistem de automatizare a locuinței" și propun un fel de prototip hibrid (cu funcționalitate atât locală cât și de la distanță) cu interfață atractivă și interactivă ce poate fi gestionată atât de pe telefonul mobil cât și de pe laptop. Integrarea parțială a acestei idei în proiectul curent este reprezentată de posibilitatea accesării sistemului prin intermediul unui browser web. În secțiunea C în a doua parte a lucrării, aceiași autori includ și un rezumat sub forma unui tabel al sistemelor Smart Home menționate în aceste lucrări. Fiecare variantă de sistem Smart Home prezentată în această ilustrație se concentrează asupra unor aspecte specifice și abordează tema în moduri diverse.

TABLE I
SUMMARY OF SH SYSTEMS

SH System	Focus Criteria					Wireless Interface	Controller	Real Implementation	Web-based Smartphone	Google Assistant
	Indoor Control	Outdoor Control	Safety Security	Monitoring	Energy Management					
Anandhavalli, et al. [7]	✓	✓				Bluetooth/GSM	PIC	✓	✓	
Davidovic and Labus [8]	✓			✓		Bluetooth/WiFi	Raspberry Pi		✓	
David, et al. [9],	✓	✓	✓	✓	✓	Bluetooth/WiFi	Arduino Mega		✓	✓
Kodali and Soratkal [10]	✓					WiFi	NodeMCU		✓	✓
Jabbar, et al. [3]	✓			✓		WiFi	Arduino Mega		✓	
Imran, et al. [12]	✓					Ethernet	Arduino Mega			✓
Ozeer, et al. [13]	✓	✓		✓	✓	Fog-IoT	Raspberry Pi	✓		✓
Konidala, et al. [15]			✓			RFID	PC Server		✓	
Gupta and Chhabra [34]	✓	✓	✓	✓	✓	Ethernet	Galileo board		✓	✓
Ganesh [35]	✓	✓				GSM	8051 μ c			✓
Bhat, et al. [36]			✓		✓	WiFi	PC Server			
Badabaji and Nagaraju [38]				✓	✓	GSM/ WiFi	PC Server			✓
Kaur, et al. [39]	✓		✓		✓	GSM	Arduino		✓	
IoT@HoMe	✓	✓	✓	✓	✓	WiFi	NodeMCU	✓	✓	✓

Figura 3.1. Tabelul cu rezumatul funcționalităților sistemelor SH menționate

Cronologic, sistemul propus în 2011 [5] deținea doar o interfață wireless de tip RFID, un PC Server controller, putea fi manevrat de un smartphone și prezenta un anumit grad de securitate.

În 2015, articolele de specialitate ne aduc în prim-plan trei variante de sisteme SH cu interfață wireless de tip Bluetooth, GSM sau WiFi cu mențiunea că opțiunea de bluetooth era prezenta la toate trei, GSM la doar una dintre ele [6], iar WiFi la următoarele două [7], [8] cu control interior și adaptabilitate la smartphone. Sistemul propus de Anandhavalli și colegii lui [6] putea fi implementat real având un controller de tip PIC și putând fi controlat atât din interior, cât și din exterior, iar cel prezentat în articolul [7] oferea monitorizare și control interior conținând un controler Raspberry Pi. Prototipul recomandat de Davidovic și ceilalți [8] era la momentul respectiv cel mai bine reprezentat din punct de vedere al parametrilor și specificațiilor, acesta fiind unul Web-based și deținând un controller Arduino Mega prin care se oferea posibilitatea de monitorizare, dublul control (interior și exterior) și se garanta siguranța integrând totodată un sistem de management al energiei.

În 2016, sistemele SH proiectate de Kodali [9], de Imran [10], Gupta [11], Kaur [12] și colegii lor sunt prezentate în articole și demonstrații la diferite conferințe de specialitate. Dintre acestea, sistemul cu cele mai multe criterii de referință îndeplinite este cel propus de Gupta și Chhabra. Utilizând o interfață wireless de tip Ethernet și un controller de tip Galileo board, sistemul era unul Web based, putea fi controlat atât interior, cât și exterior, oferea securitate, posibilitatea de monitorizare, un management al energiei și adaptabilitate pentru telefonul mobil.

Publicațiile din 2017 fac referire la un sistem SH [13] cu interfață wireless GSM și controller de tip 8051 μ c. Anul următor, Jabbar [14], Badabaji [15], Bahat [16] și colegii lor se remarcă prin proiectarea a trei sisteme SH cu interfață wireless GSM sau WiFi și controller-e Arduino Mega sau PC Server, dar care îndeplinesc doar câteva dintre criteriile de referință pe care le propun restul sistemelor în tabelul sinoptic al sistemelor Smart house.

Prototipul descris într-un articol din 2019 [17] de U. Ozeer, L. Letondeur, F.G. Ottogalli, G. Salaün, și J.M. Vincent se apropie cel mai mult de varianta lui Jabbar [4] și a colegilor lui prin faptul că utilizează o interfață wireless de tip Internet of Things. IoT@Home, așa cum l-au denumit ei are o interfață wireless de tip WiFi și un controller NodeMCU, prezintă securitate și siguranță, oferă posibilitatea de monitorizare, management al energiei și dublu control, poate fi implementat real, este web-based, adaptabil pentru smartphone și conține Google Assistant.

Cum, în zilele noastre, Internet of Things joacă un rol extrem de important în ceea ce privește controlul și managementul energiei electrice a unei locuințe care conține un sistem inteligent, câțiva specialiști în domeniu [18] dezvoltă o așa-zisă IoT Smart Household Distribution Board (ISHDB) pentru a monitoriza și controla diferite aparaturi smart de întreținere a locuinței. Funcția principală a plăcii de distribuție proiectate este aceea de a colecta și stoca date referitoare la tensiune, curent și putere, și să le afișeze într-o manieră cât mai atractivă și intuitivă folosind o aplicație mobilă. În acest sens, se folosește un prototip bazat pe Arduino pentru a colecta datele în cloud-ul ThingSpeak prin intermediul unei rețele Wi-Fi. Tehnologia Arduino este folosită pentru automatizarea procesului și reducerea dimensiunii și costului hardware-ului. Prin comparație cu alte sisteme existente, autorii susțin superioritatea metodei propuse din perspectiva costurilor de implementare mai scăzute, scop prezent și în implementarea sistemului prezentat în această lucrare.

Capitolul 4. Analiză și fundamentare Teoretică

Sistemul smart house controlat de la distanță reprezintă un produs care intenționează să implementeze funcționalitățile de bază ale unui sistem de tip smart home prin intermediul unor componente și procese relativ simple comparativ cu competiția astfel încât rezultatul final să poată fi accesibil oricărei persoane, indiferent de expertiza tehnică pe care o are.

Proiectul este compus din două părți principale, hardware și software. Partea hardware este compusă din două microcontrolere pentru control și perifericele care execută comenzile sau monitorizează parametrii doriți de către utilizator.

Partea software care este compusă dintr-o platformă web pe care se poate accesa interfața grafică responsabilă cu trimiterea comenzilor prin intermediul unor butoane sau a unor textbox-uri și cu afișarea parametrilor monitorizați de către partea hardware.

4.1. Cazuri de utilizare

În Figura 4.1. sunt descrise acțiunile pe care le poate lua un utilizator în cadrul sistemului.

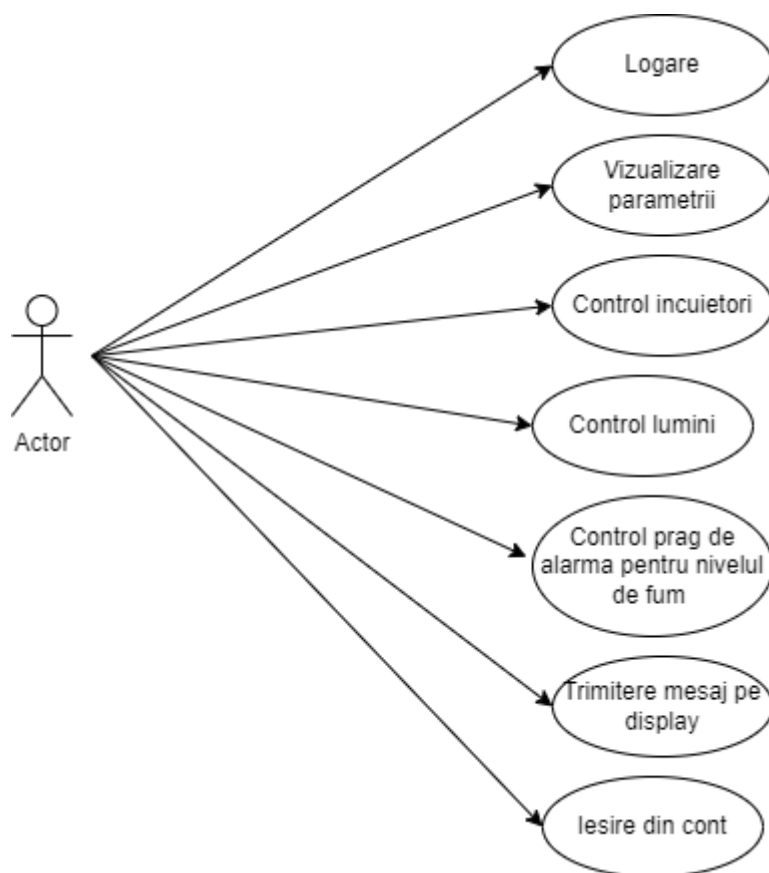


Figura 4.1. Use case

În Figura 4.1. este descris principalul flux de evenimente al sistemului smart house controlat de la distanță. Aceasta diagrama prezintă principalele acțiuni pe care le poate lua utilizatorul în timpul folosirii sistemului.

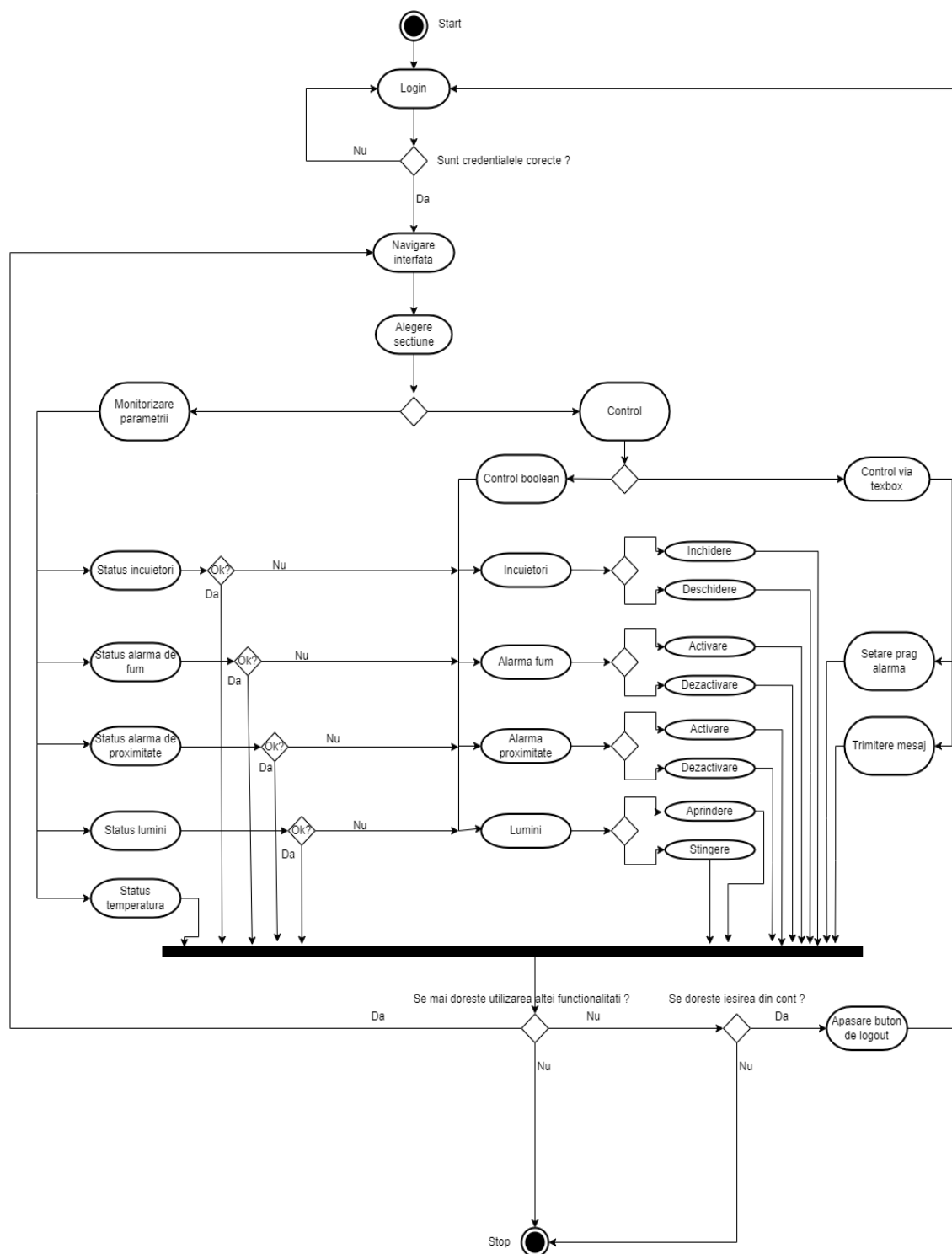


Figura 4.2. Flow Chart

4.1.1. Autentificare

Descriere caz de utilizare:

Utilizatorul introduce credențialele primite in pagina de login

Actor vizat: Utilizatorul

Precondiții:

- Acces la un dispozitiv care dispune de browser web
- Acces la internet
- Acces la credențialele primite de la administratorul sistemului .

Scenariu de utilizare:

- Utilizatorul accesează un browser web
- Utilizatorul introduce adresa la care se afla interfața grafica
- Daca nu a mai fost logat pe dispozitivul respectiv, sau la finalul utilizării a părăsit pagina prin intermediul butonului de logout, acesta va fi redirectionat către pagina de login.
- Vor fi introduse credențialele valide in textbox-urile corespunzătoare
- Se va apăsa butonul de login pentru trimiterea formularului
- Daca credențialele sunt corecte utilizatorul va fi redirectionat către interfața grafica de control.

Scenarii alternative:

- iv a. Se introduce o combinație invalidă de nume de utilizator - parolă
 - iv a. 1) Se reîncarcă pagina fără conținutul aflat in textbox-uri
 - iv a. 2) Se afișează un mesaj de eroare
 - iv a. 3) Se introduc credențialele valide.
 - iv a. 4) Se continua de la pasul v.

Postcondiții:

- Utilizatorul de afla pe pagina interfeței grafice si poate sa folosească funcționalitățile sistemului.

4.1.2. Vizualizare parametrii

Descriere caz de utilizare:

Utilizatorul monitorizează parametrii casei prin intermediul interfeței grafice.

Actor vizat: Utilizatorul

Precondiții:

- Acces la un dispozitiv care dispune de browser web
- Acces la internet
- Acces la credențialele primite de la administratorul sistemului .

Scenariu de utilizare:

- Utilizatorul accesează interfața grafica
- Parcurge interfața pana când ajunge la secțiunea cu parametrul dorit
- Daca senzorul respectiv este activat, utilizatorul vizualizează informația de care are nevoie.

Scenarii alternative:

- iii a. Senzorul care oferă informații cu privire la parametrul dorit este dezactivat.
 - iii a. 1) Se apasă butonul care activează pagina
 - iii a. 2) Se așteaptă câteva secunde
 - iii a. 3) Se da refresh la pagina
 - iii a. 4) Se vizualizează informația dorita

Postcondiții:

- Utilizatorul de afla pe pagina interfeței grafice si poate sa folosească funcționalitățile sistemului.

4.1.3. Control încuietori

Descriere caz de utilizare:

Utilizatorul dorește sa încuie/descuie uși prin intermediul interfeței.

Actor vizat: Utilizatorul

Precondiții:

- Acces la un dispozitiv care dispune de browser web
- Acces la internet
- Acces la credențialele primite de la administratorul sistemului .

Scenariu de utilizare:

- i. Utilizatorul accesează interfața grafica
- ii. Parcurge interfața pana când ajunge la secțiunea responsabila cu securitatea.
- iii. Apasă butoanele corespunzătoare ușii pe care dorește sa o încuie sau descuie.

Postcondiții:

- Utilizatorul completează acțiunea dorita iar aparenta butoanelor se schimba in concordanta cu schimbarea.

4.1.4. Control ferestre

Descriere caz de utilizare:

Utilizatorul dorește sa închidă/deschidă ferestre prin intermediul interfeței.

Actor vizat: Utilizatorul

Precondiții:

- Acces la un dispozitiv care dispune de browser web
- Acces la internet
- Acces la credențialele primite de la administratorul sistemului .

Scenariu de utilizare:

- i. Utilizatorul accesează interfața grafica
- ii. Parcurge interfața pana când ajunge la secțiunea responsabila cu securitatea.
- iii. Apasă butoanele corespunzătoare ferestrelor pe care dorește sa le închidă sau deschidă.

Postcondiții:

- Utilizatorul completează acțiunea dorita iar aparenta butoanelor se schimba in concordanta cu schimbarea.

4.1.5. Control lumini

Descriere caz de utilizare:

Utilizatorul dorește sa aprindă/stingă lumini in anumite camere prin intermediul interfeței.

Actor vizat: Utilizatorul

Precondiții:

- Acces la un dispozitiv care dispune de browser web
- Acces la internet
- Acces la credențialele primite de la administratorul sistemului .

Scenariu de utilizare:

- i. Utilizatorul accesează interfața grafică.
- ii. Parcurge interfața până când ajunge la secțiunea responsabilă cu iluminarea.
- iii. Apasă butoanele corespunzătoare luminilor pe care dorește să le aprindă sau stingă.

Postcondiții:

- Utilizatorul completează acțiunea dorită iar aparența butoanelor se schimbă în concordanță cu schimbarea.

4.1.6. Activare/dezactivare senzori

Descriere caz de utilizare:

Utilizatorul dorește să activeze/dezactiveze senzorii care oferă informații cu privire la anumiți parametri prin intermediul interfeței.

Actor vizat: Utilizatorul

Precondiții:

- Acces la un dispozitiv care dispune de browser web
- Acces la internet
- Acces la credențialele primite de la administratorul sistemului .

Scenariu de utilizare:

- i. Utilizatorul accesează interfața grafică
- ii. Parcurge interfața până când ajunge la secțiunea responsabilă cu securitatea.
- iii. Apasă butoanele corespunzătoare senzorilor pe care dorește să îi activeze sau dezactiveze

Postcondiții:

- Utilizatorul completează acțiunea dorită iar aparența butoanelor se schimbă în concordanță cu schimbarea.

4.1.7. Setare prag de activare a alarmei pentru detectorul de fum

Descriere caz de utilizare:

Utilizatorul dorește să schimbe pragul la care se activează alarma de fum.

Actor vizat: Utilizatorul

Precondiții:

- Acces la un dispozitiv care dispune de browser web
- Acces la internet
- Acces la credențialele primite de la administratorul sistemului .

Scenariu de utilizare:

- i. Utilizatorul accesează interfața grafică
- ii. Parcurge interfața până când ajunge la secțiunea responsabilă cu controlul numeric.
- iii. Introduce valoarea numerică dorită în textbox.
- iv. Apasă butonul change.

Postcondiții:

- Utilizatorul completează acțiunea iar valoarea din textbox este schimbată.

4.1.8. Afișare mesaj pe LCD

Descriere caz de utilizare:

Utilizatorul dorește să afișeze un mesaj pe LCD.

Actor vizat: Utilizatorul

Precondiții:

- Acces la un dispozitiv care dispune de browser web
- Acces la internet
- Acces la credențialele primite de la administratorul sistemului .

Scenariu de utilizare:

- Utilizatorul accesează interfața grafică
- Parcurge interfața până când ajunge la secțiunea responsabilă cu trimiterea textului.
- Introduce propoziția dorită în textbox. Aceasta propoziție poate să aibă un maxim de 35 de caractere.
- Apasă butonul change.

Scenarii alternative:

- a. Propoziția introdusă depășește cele 35 de caractere
 - Se rescrie propoziția astfel încât să îndeplinească constrângerile impuse.
 - Se apasă butonul change
 - Se resetează manual plăcuța Arduino Uno prin apăsarea butonului reset prezent pe aceasta.
 - Se așteaptă câteva secunde până la resetarea sistemului .

Postcondiții:

- Utilizatorul completează acțiunea iar valoarea din textbox este schimbată.

4.2. Protocoale de comunicare utilizate

4.2.1. Protocol de comunicare între hardware și platforma web

4.2.1.1. HTTP

Comunicarea între microcontrolere și platforma web este realizată prin intermediul protocolului HTTP.

HTTP (Hypertext transfer protocol) este protocolul principal utilizat de către WWW(World Wide Web) pentru accesarea informațiilor pe internet. Acesta se bazează pe o structură de tip cerere – răspuns.

Cererea este trimisă de client către serverul web și reprezintă o solicitare pentru obținerea unor resurse identificate prin intermediul unui URL. Cererea conține informații despre metoda utilizată pentru transferul datelor, locația resursei și versiunea protocolului utilizat.

Răspunsul conține versiunea protocolului, un cod de status, un mesaj de status și mai multe headere. Versiunea protocolului specifică formatul și specificațiile utilizate în răspunsul HTTP, codul de stare este un număr care indică rezultatul cererii, headerele sunt componente suplimentare ale răspunsului HTTP și conțin informații adiționale, cum ar fi tipul de conținut returnat, dimensiunea fișierului, setările de cache sau cookie-urile.

Cele mai importante metode HTTP :

- **GET** este folosită pentru a obține informații de pe un server web prin specificarea unui URI (Uniform Resource Identifier) în cerere. Cererile efectuate prin metoda GET sunt destinate să recupereze date și nu au niciun alt efect asupra resurselor de pe server.
- **HEAD** este similară metodei GET, cu excepția faptului că serverul returnează doar linia de stare și secțiunea antet a răspunsului, fără a include corpul de răspuns al resursei solicitate.
- **POST** este utilizată pentru a transfera informații de la client către serverul web cu privire la o resursă specificată. Aceste informații sunt incluse în corpul cererii HTTP

și sunt destinate să fie prelucrate de către server. De obicei, metoda POST este utilizată atunci când se dorește trimiterea de date către server pentru a fi salvate, actualizate sau procesate într-un anumit mod

- **PUT** este utilizată pentru a încărca o resursă specificată de pe client pe serverul web, înlocuind-o în cazul în care deja există. Aceasta necesită specificarea datelor de autentificare și permisiunile adecvate pentru a efectua această operație.
- **DELETE** este utilizată pentru a șterge o resursă specificată de pe serverul web. Rezultatul acestei operații depinde de permisiunile utilizatorului care furnizează datele de autentificare în antetele cererii.

Codurile de răspuns au următoarele interpretări :

- Clasa de status 1xx reprezintă erori informaționale și pot fi ignorate.
- Clasa de status 2xx indică că cererea a fost primită și procesată cu succes.
- Clasa de status 3xx indică redirectionări către alte surse.
- Clasa de status 4xx indică erori ale utilizatorului în formularea cererii.
- Clasa de status 5xx indică erori ale serverului.

Protocolul HTTP este utilizat pentru a permite comunicarea între dispozitivul hardware ESP8266 și platforma online. Platforma trimite un mesaj de echo care conține informații despre starea curentă a butoanelor și a datelor din baza de date necesare microcontrolerului. Acesta, la rândul său, trimite o cerere de tip POST către platformă pentru a prelua aceste date și pentru a comunica platformei starea curentă a senzorilor. Informațiile cu privire la senzori sunt trimise în cererea de tip POST sub forma "info1=value1&info2=value2&info3=value3";

4.2.2. Protocoale de comunicare între module hardware

4.2.2.1. UART

UART (Universal Asynchronous Receiver/Transmitter) este un protocol de comunicare hardware ce se bazează pe transmiterea serială asincronă. Într-o comunicare asincronă, nu există un semnal de sincronizare (clock) care să coordoneze trimiterea și recepționarea biților. Pentru a depăși acest impediment, protocolul UART utilizează o viteză configurabilă pentru transferul datelor (Baud rate).

Protocolul UART este simplu și foarte ușor de implementat deoarece utilizează doar trei fire, un fir comun pentru împământare, unul pentru transmitere și celălalt pentru recepție serială.

Conexiunea între două dispozitive este de asemenea simplă, se leagă dispozitivele la o împământare comună, pinul RX al primului dispozitiv fiind legat la pinul TX al celui de-al doilea iar pinul TX al primului dispozitiv fiind legat la pinul RX al celui de-al doilea după cum este ilustrat în figura 4.2.

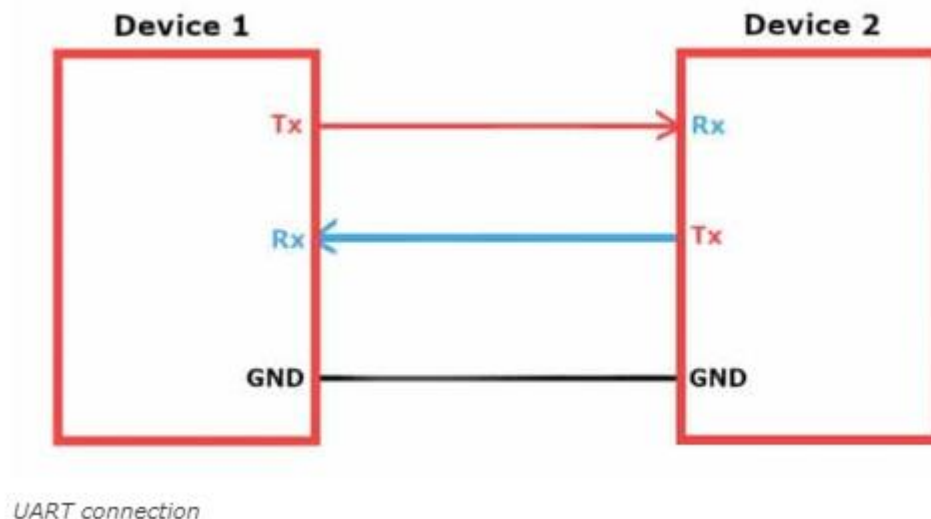


Figura 4.3. Conexiune UART

Sursă imagine:

<https://911electronic.com/uart-serial-communication-interface/>

Dispozitivele UART primesc date în mod asincron, bit cu bit, prin portul lor RX și trimit date în același mod prin portul TX. Pentru o comunicare corectă, este necesar să se configureze același "Baud rate" pe ambele dispozitive, care reprezintă rata de transmitere a datelor în biți pe secundă. Deoarece UART nu utilizează un semnal de clock pentru sincronizare, transmițătorul și receptorul folosesc propriile semnale de clock pentru a coordona transmiterea și recepția datelor. Astfel, sincronizarea baud rate-ului asigură o transmisie și recepție corecte între dispozitive.

Comunicarea UART necesită un fir comun pentru împământare pentru a asigura o referință comună pentru semnalul de date transmis între dispozitive.

Acest protocol este folosit în proiect pentru comunicarea între plăcuțele de dezvoltare Arduino Uno și Lolin NodeMCU.

4.2.2.2. I2C

I2C (Inter-Integrated Circuit) este un protocol de comunicație serială sincronă utilizat pentru transmiterea datelor între diferite dispozitive electronice.

Protocolul I2C implică două linii principale: linia de date (SDA - Serial Data Line) și linia de ceas (SCL - Serial Clock Line).

Linia de date (SDA) în protocolul I2C este utilizată pentru transferul efectiv al datelor între dispozitive în mod serial, fiind de asemenea o linie bidirecțională.

Linia de ceas (SCL) este responsabilă de sincronizarea comunicației între dispozitive. Este o linie unidirecțională, iar semnalele de ceas sunt generate de dispozitivul master.

În timpul comunicației I2C, datele sunt transmise în pachete de biți, iar semnalele de ceas sunt utilizate pentru a sincroniza trimiterea și recepționarea acestor biți pe linia de date.

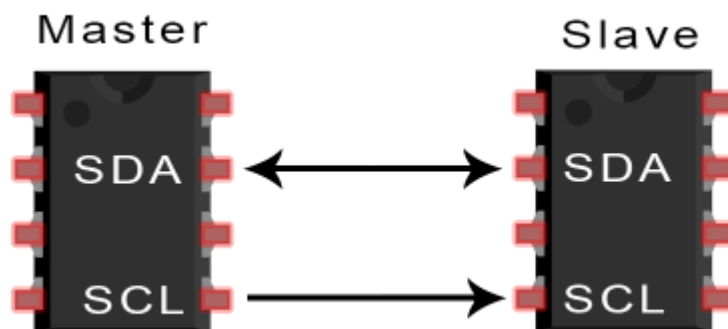


Figura 4.4. Conexiune I2C

Sursă imagine:

<https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/embed/>

În cadrul proiectului comunicația I2C a fost folosită între modulul I2C al display-ului LCD și plăcuța de dezvoltare Arduino Uno, pentru simplificarea conexiunilor.

4.3. Componente fizice utilizate

4.3.1. Arduino Uno

Arduino Uno este o placă de dezvoltare bazată pe procesorul ATmega328P care funcționează la o frecvență de ceas de 16 MHz.

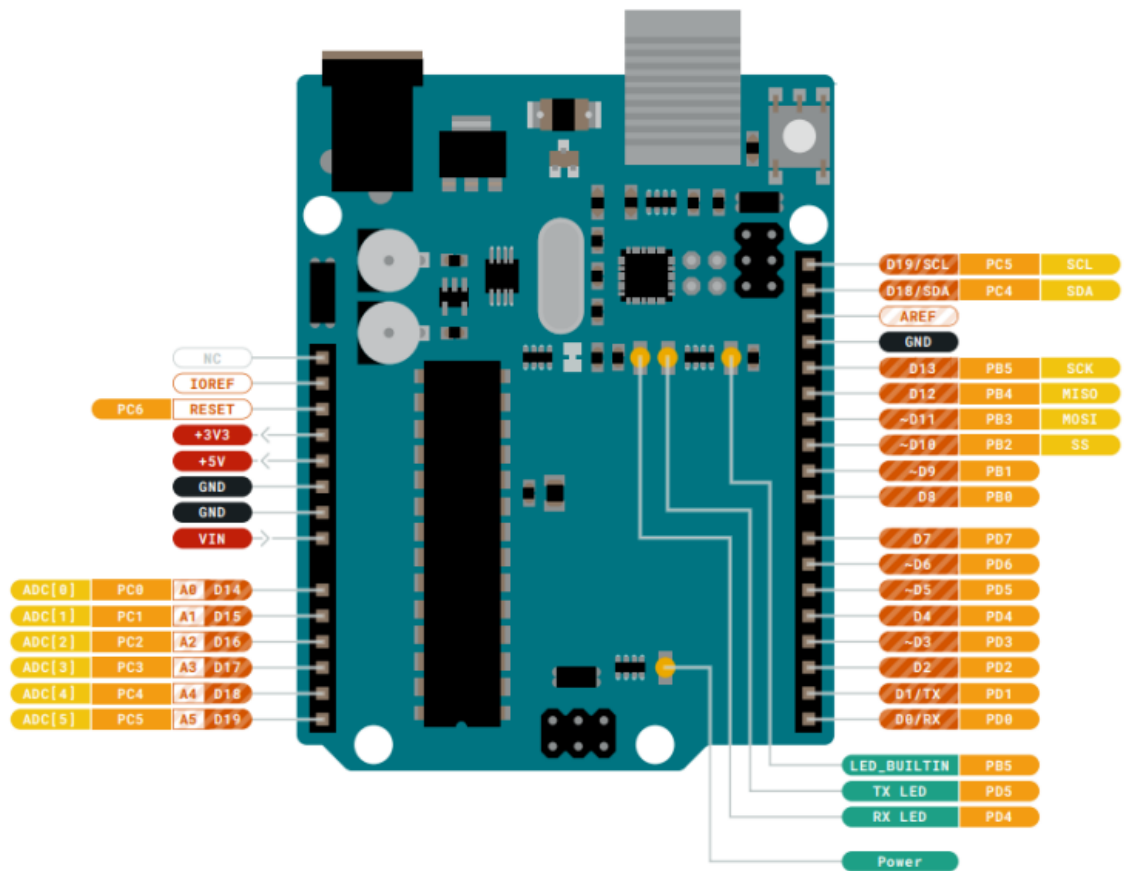


Figura 4.5. Arduino Uno

Sursă imagine:

<https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>

Pinout :



Pinout

Figura 4.6. Pinout Arduino Uno

Sursă imagine:

<https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>

5.1 JANALOG

Pin	Function	Type	Description
1	NC	NC	Not connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog/GPIO	Analog input 0 /GPIO
10	A1	Analog/GPIO	Analog input 1 /GPIO
11	A2	Analog/GPIO	Analog input 2 /GPIO
12	A3	Analog/GPIO	Analog input 3 /GPIO
13	A4/SDA	Analog input/I2C	Analog input 4/I2C Data line
14	A5/SCL	Analog input/I2C	Analog input 5/I2C Clock line

5.2 JDIGITAL

Pin	Function	Type	Description
1	D0	Digital/GPIO	Digital pin 0/GPIO
2	D1	Digital/GPIO	Digital pin 1/GPIO
3	D2	Digital/GPIO	Digital pin 2/GPIO
4	D3	Digital/GPIO	Digital pin 3/GPIO
5	D4	Digital/GPIO	Digital pin 4/GPIO
6	D5	Digital/GPIO	Digital pin 5/GPIO
7	D6	Digital/GPIO	Digital pin 6/GPIO
8	D7	Digital/GPIO	Digital pin 7/GPIO
9	D8	Digital/GPIO	Digital pin 8/GPIO
10	D9	Digital/GPIO	Digital pin 9/GPIO
11	SS	Digital	SPI Chip Select
12	MOSI	Digital	SPI1 Main Out Secondary In
13	MISO	Digital	SPI Main In Secondary Out
14	SCK	Digital	SPI serial clock output
15	GND	Power	Ground
16	AREF	Digital	Analog reference voltage
17	A4/SD4	Digital	Analog input 4/I2C Data line (duplicated)
18	A5/SD5	Digital	Analog input 5/I2C Clock line (duplicated)

Figura 4.7. Descrierea pinilor placii Arduino Uno

Sursă imagine:

<https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>

Din punct de vedere al puterii electrice necesare, microcontrolerul Arduino Uno are nevoie de o putere de input recomandată cu o valoare aflată undeva în intervalul 7-12 de volți, însă limitele pe care le poate suporta se extind până la intervalul de 6-20 de volți. Voltajul la care aceasta operează este de 5 volți pentru o compatibilitate mai bună cu posibilele periferice.

Curentul direct suportat de pinii de input și output este de 20 de miliamperi. Pentru pinul de 3.3 volți acest curent crește la 50 de miliamperi.

Din punct de vedere al memoriei, versiunea care folosește procesorul ATmega328P are o memorie flash de 32 KB o memorie SRAM de 2 KB și o memorie EEPROM de 1 KB.

ATMEGA328P Pinout

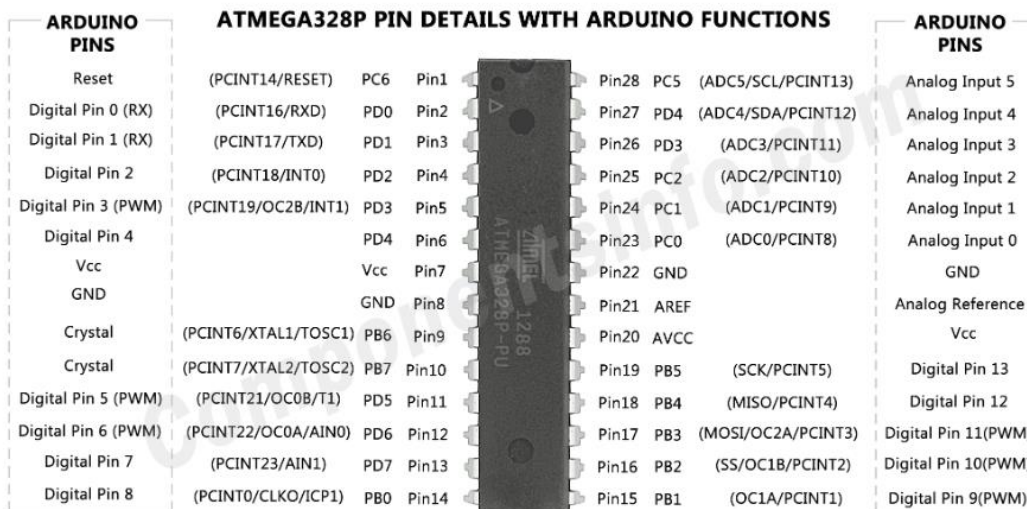


Figura 4.8. ATMEGA328P Pinout

Sursă imagine:

<https://www.componentsinfo.com/atmega328p-pinout-configuration-datasheet/>

Partea de securitate si fiabilitate este asigurata de doua funcționalități si anume Brown Out Detection (BOD) si Power On Reset (POR).

Brown Out Detection reprezintă caracteristica microcontrolerului de a detecta dacă tensiune de alimentare scade sub un anumit prag predefinit. Aceste scăderi de tensiuni pot să apară în diverse situații precum fluctuații în rețeaua care asigură curent plăcuței sau conectarea la aceasta a unei componente care prezintă un consum ridicat.

Atunci când Brown Out Detection este activat, microcontrolerul sau circuitul integrat poate lua măsuri un urma detectării pentru a proteja sistemul împotriva unor eventuale probleme sau erori. De exemplu, poate opri temporar sau reduce activitatea anumitor componente sau poate întrerupe funcționarea sistemului înainte ca tensiunea să devină insuficientă pentru a asigura funcționarea corectă a circuitelor.

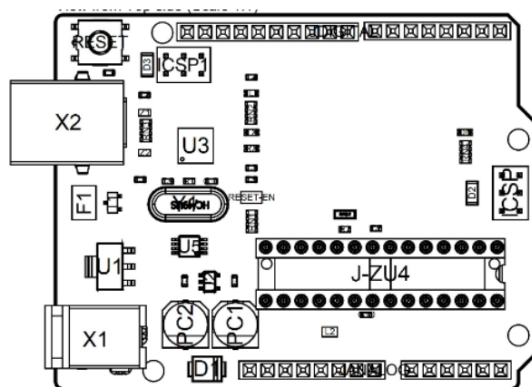
Power On Reset asigura o inițializare stabila si fără erori a sistemului in momentul alimentarii acestuia. Aceasta funcționalitate este obținută prin intermediul unui circuit care monitorizează tensiunea de alimentare si generează un semnal de resetare atunci când aceasta ajunge la nivelul corespunzător pentru a asigura o funcționare corecta a sistemului.

Din punct de vedere al topologiei vor fi menționate câteva componente folosite in timpul procesului de dezvoltare, printre care :

- Conectorul USB de tip B, folosit atât pentru alimentarea plăcuței cat si pentru încărcarea codului necesar pentru funcționare.
- Microcontrolerul ATmega328P care procesează informațiile si comenzile din cod si care sta la baza plăcuței Arduino Uno.
- ATMEGA16U2 Module folosit ca si un convertor USB to Serial.

Majoritatea componentelor ramase in topologie au ca rol reglarea si controlul curentului electric.

Top view



Board topology

Ref.	Description	Ref.	Description
X1	Power jack 2.1x5.5mm	U1	SPX1117M3-L-5 Regulator
X2	USB B Connector	U3	ATMEGA16U2 Module
PC1	EEE-1EA470WP 25V SMD Capacitor	U5	LMV358LIST-A.9 IC
PC2	EEE-1EA470WP 25V SMD Capacitor	F1	Chip Capacitor, High Density
D1	CGR4007-G Rectifier	ICSP	Pin header connector (through hole 6)
J-ZU4	ATMEGA328P Module	ICSP1	Pin header connector (through hole 6)
Y1	ECS-160-20-4X-DU Oscillator		

Figura 4.9. Topologia placutei Arduino Uno

Sursă imagine:

<https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>

4.3.2. Lolin NodeMCU ESP8266

Lolin NodeMCU ESP8266 este o placa de dezvoltare ideala pentru proiecte de tip IoT care combina funcționalitățile unui Arduino cu conectivitatea WiFi. Aceasta are la baza microcontrolerul ESP-8266 care folosește o viteză de ceas de 80MHz.

Pinout :

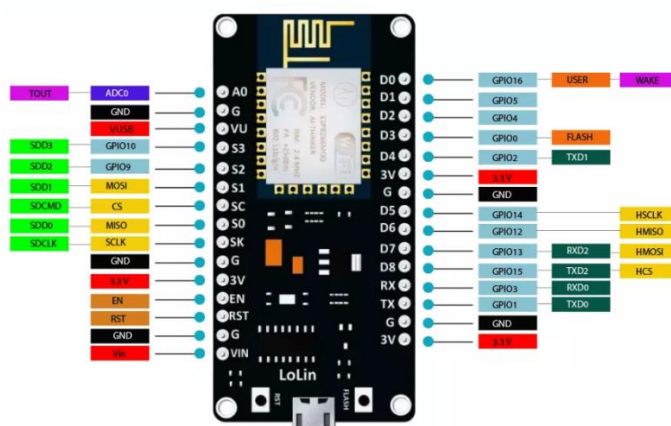


Figura 4.10. Pinout Lolin NodeMCU ESP8266

Sursă imagine:

<https://www.theengineeringprojects.com/2018/08/esp8266-pinout-datasheet-features-applications.html>

Pini importanți :

Label	GPIO	Input	Output	Notes
D0	GPIO16	no interrupt	no PWM or I2C support	HIGH at boot used to wake up from deep sleep
D1	GPIO5	OK	OK	often used as SCL (I2C)
D2	GPIO4	OK	OK	often used as SDA (I2C)
D3	GPIO0	pulled up	OK	connected to FLASH button, boot fails if pulled LOW
D4	GPIO2	pulled up	OK	HIGH at boot connected to on-board LED, boot fails if pulled LOW
D5	GPIO14	OK	OK	SPI (SCLK)
D6	GPIO12	OK	OK	SPI (MISO)
D7	GPIO13	OK	OK	SPI (MOSI)
D8	GPIO15	pulled to GND	OK	SPI (CS) Boot fails if pulled HIGH
RX	GPIO3	OK	RX pin	HIGH at boot
TX	GPIO1	TX pin	OK	HIGH at boot debug output at boot, boot fails if pulled LOW
A0	ADC0	Analog Input	X	

Figura 4.11. Descrierea pinilor placutei Lolin NodeMCU ESP8266

Sursă imagine:

<https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/>

Placa de dezvoltare Lolin NodeMCU ESP8266 este alimentată și operează la o tensiune de 3.3 volți însă poate fi alimentată și la 5 volți prin cablul micro USB care servește și ca un mediu de încărcare a programelor în microcontroler. Tensiunea de input poate varia între 4.5 și 10 volți.

Microcontrolerul ESP8266 are o memorie flash de 4MB și o memorie SRAM de 64 KB.

Această placă de dezvoltare conține un convertor USB to Serial (CH340G), interfețe pentru comunicarea serială cu RX și TX, I2C, SPI și UART și Built-In WiFi 802.11 b/g/n.

4.3.3. Motor Micro Servo SG-90

SG-90 este un motor servo mic și ușor care operează la o tensiune de 5+ volți are un cuplu de 2.5kg/cm, viteza de operare este de 0.1s/60°, și are o rotație de 180 de grade (90 de grade în fiecare direcție). În principal este făcut din plastic cu o greutate de doar 9 grame și are trei fire după cum urmează:

- Maro pentru împământare.
- Roșu pentru tensiunea de alimentare.
- Portocaliu pentru PWM (control).

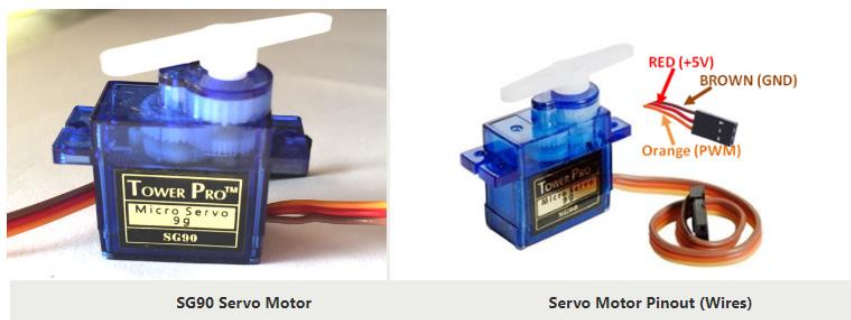


Figura 4.12. SG90 cu pinout

Sursă imagine:

<https://components101.com/motors/servo-motor-basics-pinout-datasheet>

4.3.4. Senzorul ultrasonic HC-SR04

HC-SR04 este un senzor ultrasonic care este folosit la măsurarea distanței sau pentru detecția obiectelor.

Funcționare:

- Senzorul emite un semnal ultrasonic în frecvența de aproximativ 40 kHz. Acest semnal este generat de un transmițător ultrasonic din senzor.
- Semnalul emis se propagă în mediul înconjurător și, atunci când întâlnește un obiect sau o suprafață, este reflectat înapoi către senzor.
- Senzorul HC-SR04 are un receptor ultrasonic care detectează semnalul de revenire sau ecoul.
- Pe baza timpului scurs între emisie și recepția semnalului de revenire, senzorul calculează distanța până la obiectul sau suprafața detectată. Acest calcul se realizează pe baza cunoașterii vitezei de propagare a sunetului în mediul respectiv.
- Senzorul HC-SR04 furnizează distanța măsurată ca rezultat, care poate fi utilizată în funcție de necesitățile proiectului.

Acest senzor operează la o tensiune de 5 volți distanța teoretică de măsurare se afla între 2 și 400 de cm, însă, în realitate distanța practică este restrânsă la intervalul de 2 – 80 cm. Operează la un curent de 15 miliamperi și un unghi de măsurare de 15 grade.



Figura 4.13. HC-SR04 cu pinout

Sursă imagine:

<https://components101.com/sensors/ultrasonic-sensor-working-pinout-datasheet>

4.3.5. Senzorul de fum și gaz MQ-2

MQ-2 este un senzor de gaz care este folosit în special pentru detectarea concentrațiilor de fum și gaze inflamabile. Senzorul este compus dintr-un element senzor ceramic încălzit electric. Acest element are o rezistență electrică inițială și își modifică valoarea în prezența fumului, metanului și butanului fiind conectat la un circuit electronic care măsoară această schimbare. Valoarea rezistenței variază în funcție de concentrația gazului detectat. Circuitul electronic interpretează valorile de rezistență și le convertește într-o valoare digitală sau analogică, care poate fi utilizată.

Acest senzor operează la o tensiune de 5 volți. Output-ul, atât la pinul analog cat și la cel digital este de 5 volți.

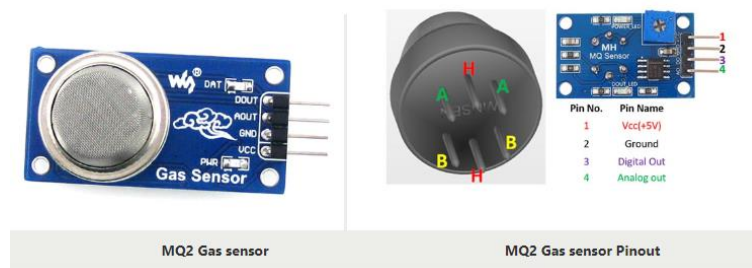


Figura 4.14. MQ-2 cu pinout

Sursă imagine:

<https://components101.com/sensors/mq2-gas-sensor>

4.3.6. Senzorul de temperatura LM35

LM35 este un senzor analogic folosit pentru măsurarea temperaturii. Acesta furnizează o tensiune de ieșire proporțională cu temperatura în grade Celsius.

Senzorul poate măsura temperaturi în intervalul de la -55°C până la $+150^{\circ}\text{C}$, oferă o precizie bună, având o eroare maximă de $\pm 0.5^{\circ}\text{C}$ în condiții normale de funcționare și tensiunea de ieșire a senzorului este proporțională cu temperatura măsurată, cu o raportare de $10\text{ mV}/^{\circ}\text{C}$. De exemplu, la o temperatură de 25°C , tensiunea de ieșire va fi de 250 mV.

Acesta consumă o cantitate mică de curent și necesită alimentare cu o tensiune de 5 volți dar raza de operare se află între 4 și 30 de volți.

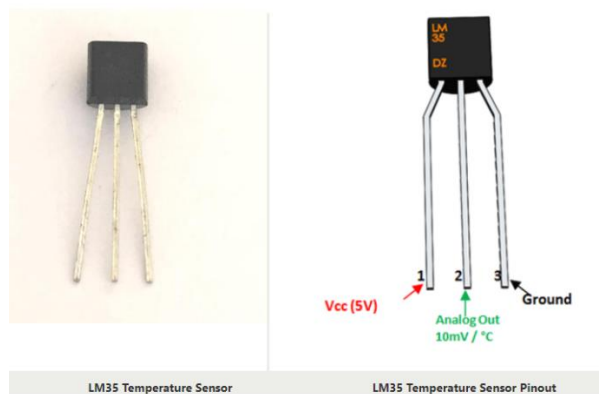


Figura 4.15. LM35 cu pinout

Sursă imagine:

<https://components101.com/sensors/lm35-temperature-sensor>

4.3.7. Ecran LCD 20x4 2004A

LCD 20x 2004A este un ecran cu cristale lichide cu lumina de fond albastra. Poate afișa 4 linii de text și fiecare linie poate avea până la 20 de caractere. Acesta funcționează la o tensiune de alimentare de 5 volți și este compatibil cu un modul de comunicare I2C folosit pentru a simplifica conexiunea cu un microcontroller.

*Principiul de funcționare al unui ecran LCD se bazează pe proprietățile cristalelor lichide care sunt capabile să schimbe alinierea moleculelor din care sunt compuse sub influența unui câmp electric, aliniere care schimbă la rândul ei direcția luminii.

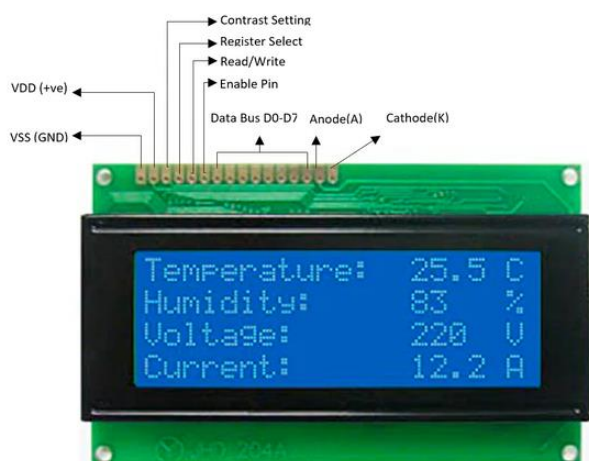


Figura 4.16. LCD 20x4 2004A cu pinout

Sursă imagine:

<https://quartzcomponents.com/products/20x4-graphical-lcd-blue>

4.3.8. Modul Serial I2C

Modulul Serial I2C este un dispozitiv care permite conectarea și controlul unui afișaj LCD utilizând protocolul I2C (Inter-Integrated Circuit). Aceasta înseamnă că modulul oferă o interfață simplificată pentru comunicarea între un microcontroller sau o placă de dezvoltare și ecranul LCD prin intermediul liniei seriale I2C utilizând doar două linii.

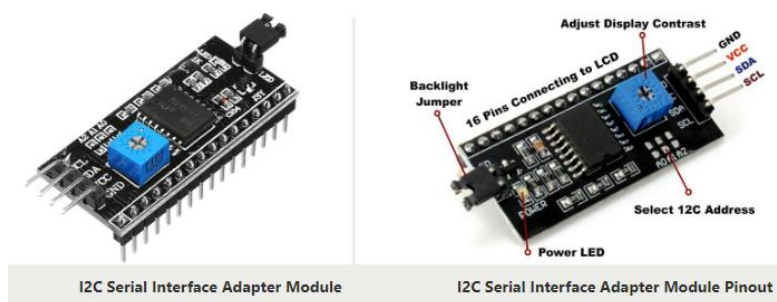


Figura 4.17. Modul Serial I2C cu pinout

Sursă imagine:

<https://components101.com/modules/i2c-serial-interface-adapter-module>

4.3.9. Buzzer pasiv

Buzzerul pasiv este un dispozitiv utilizat pentru a genera semnale sonore. Acesta constă într-un oscilator intern care produce un semnal electric și un element de amplificare care transformă semnalul electric în sunet.

Tensiunea de funcționare este de 5 volți și produce un sunet continuu cu o frecvență de rezonanță de aproximativ 2300 Hz.



Figura 4.18. Buzzer pasiv cu pinout

Sursă imagine:

<https://components101.com/misc/buzzer-pinout-working-datasheet>

4.3.10. Fire de legătură (Jumper wires)

Firul de legătură este un cablu de mici dimensiuni și flexibil, utilizat în domeniul electronic pentru a conecta sau lega componente, module sau puncte de contact între ele.

Acesta constă într-un conductor electric izolat, de obicei din cupru sau aluminiu, învelit într-o înveliș izolan flexibil, precum plastic sau cauciuc. Un capăt al firului este prevăzut cu un conector.

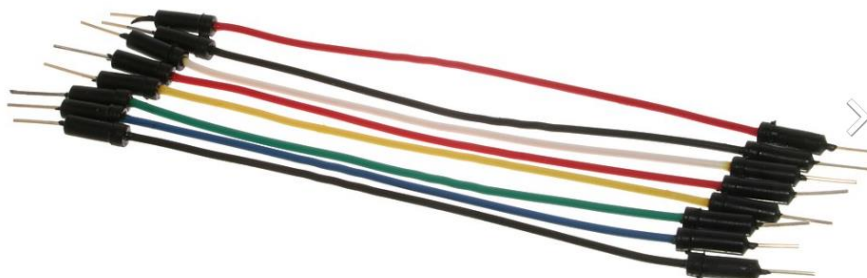


Figura 4.19. Fire de legatură

Sursă imagine:

https://en.wikipedia.org/wiki/Jump_wire

4.3.11. LED

LED (Light-Emitting Diode) este un dispozitiv electronic semiconductor care emite lumină atunci când curentul electric trece prin el. Principiul de funcționare al unui LED se bazează pe efectul de electroluminiscență a materialelor semiconductoare. Când o tensiune este aplicată pe joncțiunea PN a unui LED, electronii și găurile se recombina în materialul semiconductor, eliberând energie sub formă de fotoni (particule de lumină). Deoarece sunt componente relativ sensibile, acestea necesită o rezistență de 220 Ohm legată în serie pentru a reduce riscul de deteriorare.



Figura 4.20. LED

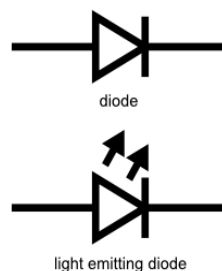


Figura 4.21. Comparatie între dioda simplă și LED

Sursă imagini:

<https://learn.sparkfun.com/tutorials/light-emitting-diodes-leds/all>

4.3.12. Rezistență

Rezistențele sunt componente electronice care sunt concepute pentru a opune trecerea curentului electric și a reduce tensiunea într-un circuit. În acest proiect au fost folosite rezistențe cu valori de 220 de ohmi pentru a asigura integritatea și siguranța LED-urilor folosite.

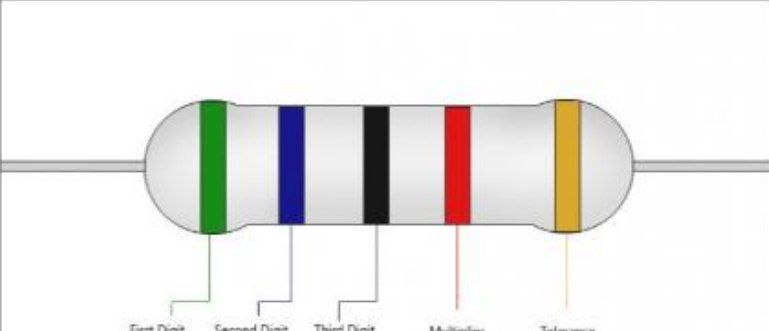


Figura 4.22. Rezistență

Sursă imagine:

<https://www.aliexpress.com/item/1005003363787140.html>

Determinarea rezistenței în funcție de codul de culoare este explicată în Figura 4.18. :



	First Digit	Second Digit	Third Digit	Multiplier	Tolerance
Black	Nil	0	0	1	Nil
Brown	1	1	1	10	±1%
Red	2	2	2	100	±2%
Orange	3	3	3	1000	±3%
Yellow	4	4	4	10000	±4%
Green	5	5	5	100000	±0.5%
Blue	6	6	6	1M	±0.25%
Violet	7	7	7	10M	±0.10%
Grey	8	8	8	100M	±0.05%
White	9	9	9	1G	Nil
Gold	Nil	Nil	Nil	±10	±5%
Silver	Nil	Nil	Nil	±100	±10%

www.CircuitsToday.com

Figura 4.23. Codurile de culoare pentru rezistențe

Sursă imagine:

<https://www.circuitstoday.com/resistor-color-code-chart>

Rezistența de 220 de ohm utilizată pentru proiect va arăta ca în Figura 4.19.

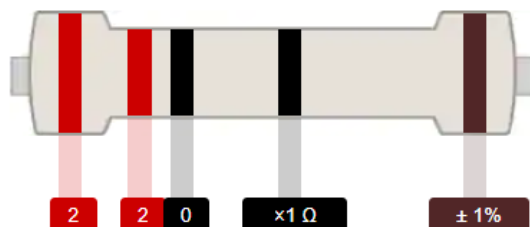


Figura 4.24. Codurile de culoare pentru rezistența folosită în proiect

Sursă imagine:

<https://support.arduino.cc/hc/en-us/articles/360012963800-Where-is-the-220-Ohm-resistor->

4.4. Tehnologii utilizate

4.4.1. Arduino IDE

Arduino Integrated Development Environment (IDE) este un software utilizat pentru programarea și dezvoltarea aplicațiilor pentru plăcile Arduino. Acesta oferă un mediu de dezvoltare ușor de utilizat care ajută la scrierea și încărcarea codului necesar pentru crearea proiectelor care includ microcontrolere.

Acest program furnizează un editor de cod. Limbajul principal folosit este Arduino care reprezintă o variantă modificată a limbajului C/C++. De asemenea acesta verifică sintaxa codului sursă și îl compilează, generând un fișier binar specific microcontrolerelor pe care îl încarcă în plăcile de dezvoltare prin intermediul portului USB. Pentru a ușura munca programatorilor, Arduino IDE pune la dispoziție o colecție de librării software predefinite care conțin funcții care facilitează folosirea diferitelor periferice hardware. Librării noi pot fi adăugate în mediul de lucru prin accesarea meniului Tools -> Manage Libraries... după care se caută în textbox-ul pentru search librăria dorită și se apasă install.

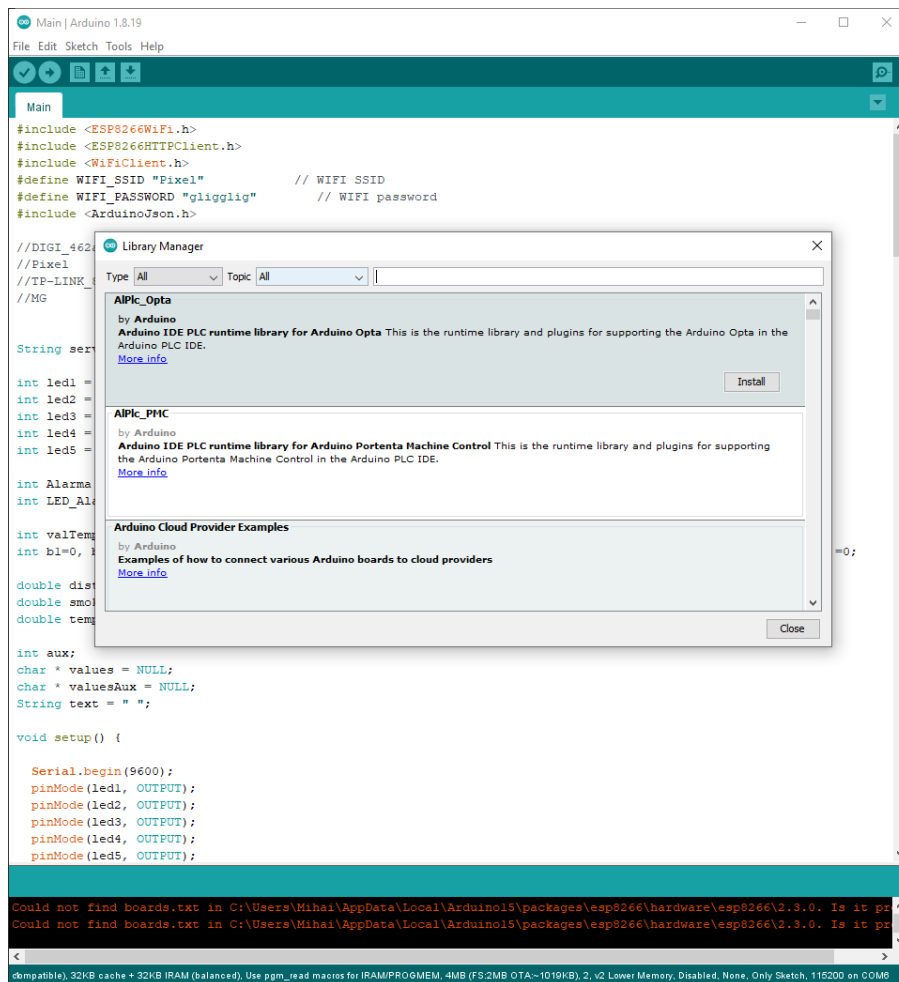


Figura 4.25. Interfața Arduino IDE cu library manager.

Un alt feature al acestui program este reprezentat de către consola de mesaje care afișează rezultatele compilării și încărcării programului, sau, mesaje de informare și erori, în funcție de context.

4.4.2. Visual Studio Code

Visual Studio Code (VS Code) este un editor de cod sursă dezvoltat de Microsoft. Este un instrument utilizat în principal pentru dezvoltarea de software, fiind prezent pe numeroase platforme precum Windows, macOS și Linux.

Acest editor are o interfață utilizator intuitivă și ușor de personalizat, cu funcționalități multiple și ușor de accesat.

Visual Studio Code oferă suport nativ pentru mai multe limbaje de programare, inclusiv JavaScript, TypeScript, Python, C++, C#, Java, HTML, CSS și multe altele. Dintre acestea, pentru proiectul curent au fost folosite limbajele HTML, CSS, PHP și SQL, toate ajutând la dezvoltarea platformei online pe care se afla interfața grafică. Acest editor de cod este extrem de ajustabil atât ca aspect cât și ca funcționalități, având nenumărate extensii folosite și ușor de instalat printr-un simplu search and install. O astfel de funcționalitate folosită pentru proiect este reprezentată de către compilatorul de limbaj PHP care nu era prezent în mod implicit.

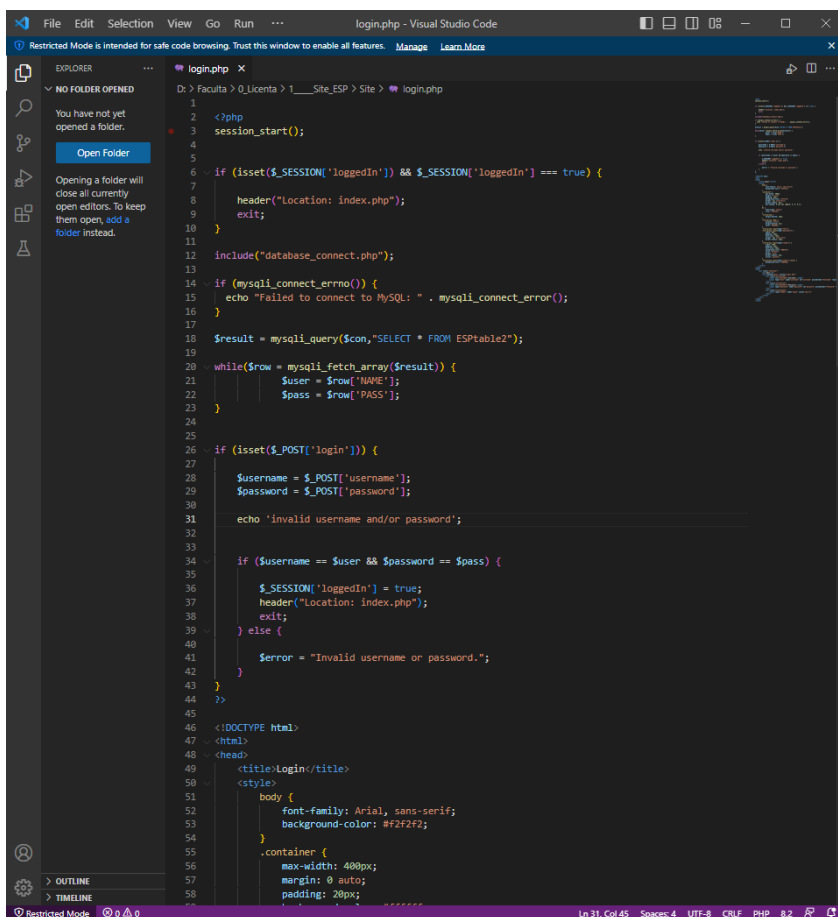


Figura 4.26. Interfața Visual Studio Code

Acesta este capabil de completarea automată a sintaxei, numelor de variabile, funcțiilor și chiar a bibliotecilor externe utilizate în proiect.

Editorul dispune de un sistem de depanare integrat care permite dezvoltatorilor să urmărească și să remedieze erorile în codul lor. De asemenea, oferă posibilitatea de a efectua profilarea aplicațiilor pentru a identifica și optimiza zonele de performanță slabă.

4.4.3. Hostinger

Hostinger International, Ltd este o companie de găzduire web și un registrator de domenii acreditat de ICANN. Fondată în 2004, compania își are sediul în Lituania și are peste 1.000 de angajați. Hostinger este proprietarul mai multor branduri, inclusiv 000webhost, Hosting24, Zyro și Niagahoster, care oferă o varietate de servicii legate de găzduire web și dezvoltare de site-uri.

Hostinger oferă patru tipuri principale de servicii de găzduire web: găzduire comună, găzduire cloud, găzduire VPS și găzduire WordPress. Toate aceste servicii sunt alimentate de serverele LiteSpeed și utilizează sistemul de operare Linux. Totuși, este important de menționat că planul Cloud Startup este alimentat de platforma Google Cloud, nu de serverele interne ale Hostinger.

Pentru a gestiona serviciile de găzduire, Hostinger oferă utilizatorilor un panou de control intern numit hPanel care a fost folosit și pentru managementul platformei online prezente în proiect.

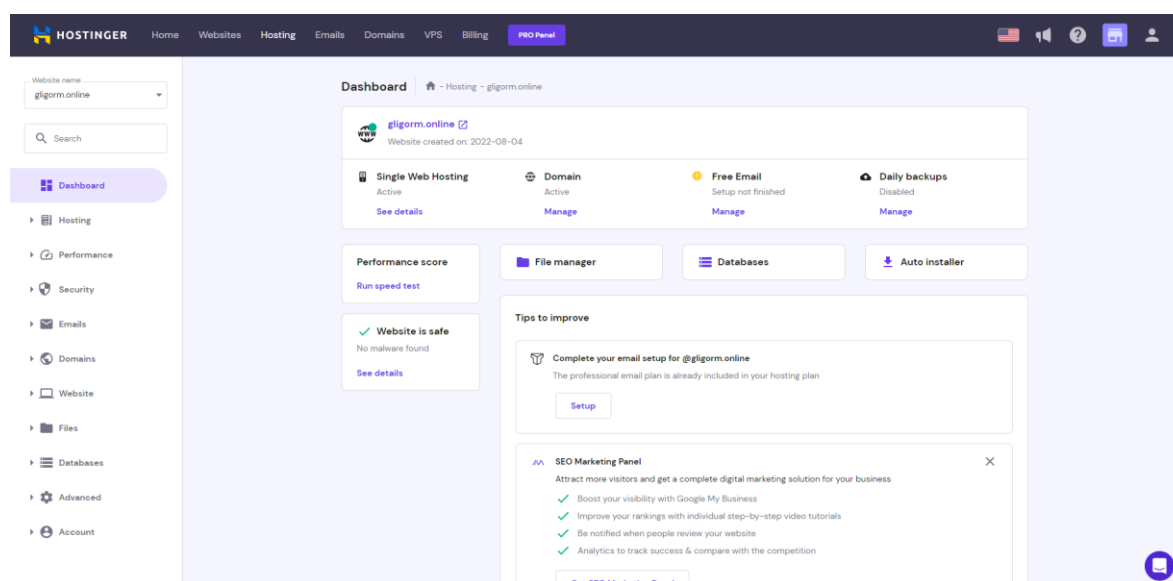


Figura 4.27. hPanel

Managementul fișierelor care compun site-ul a fost realizat în panoul „File Manager”

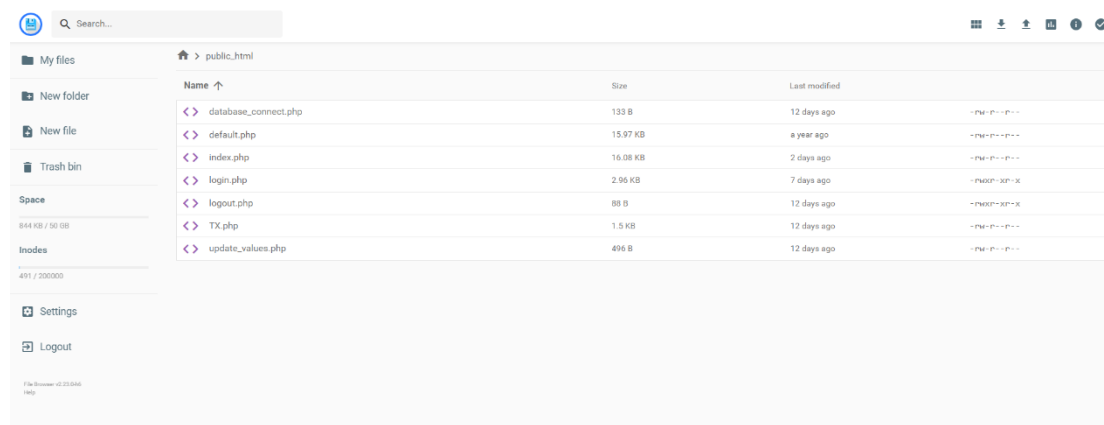


Figura 4.28. File Manager

Baza de date a fost creata in panoul „Databases”

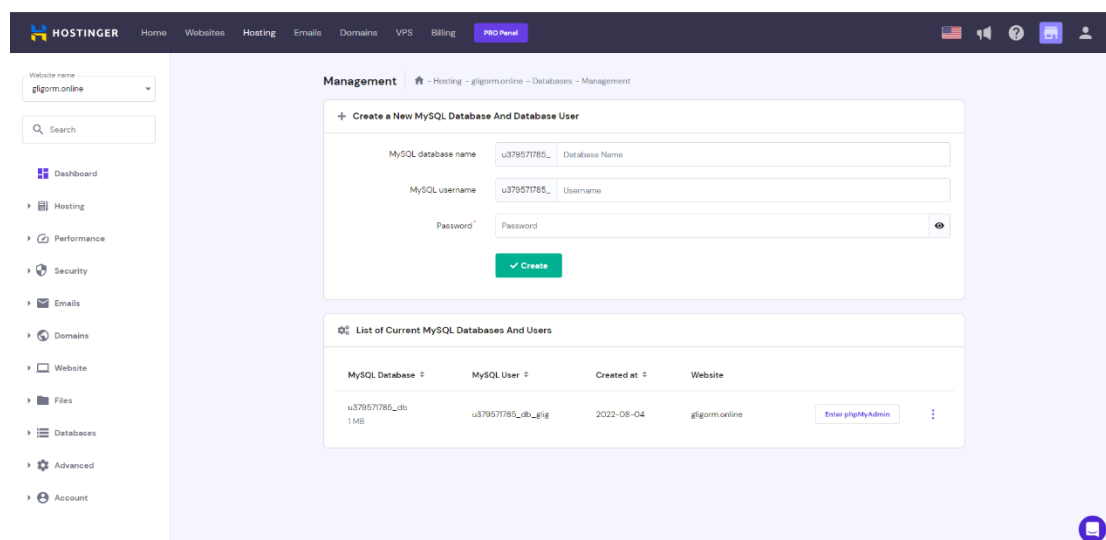


Figura 4.29. Databases

Managementul acestora a fost realizat prin accesarea paginii „phpMyAdmin”

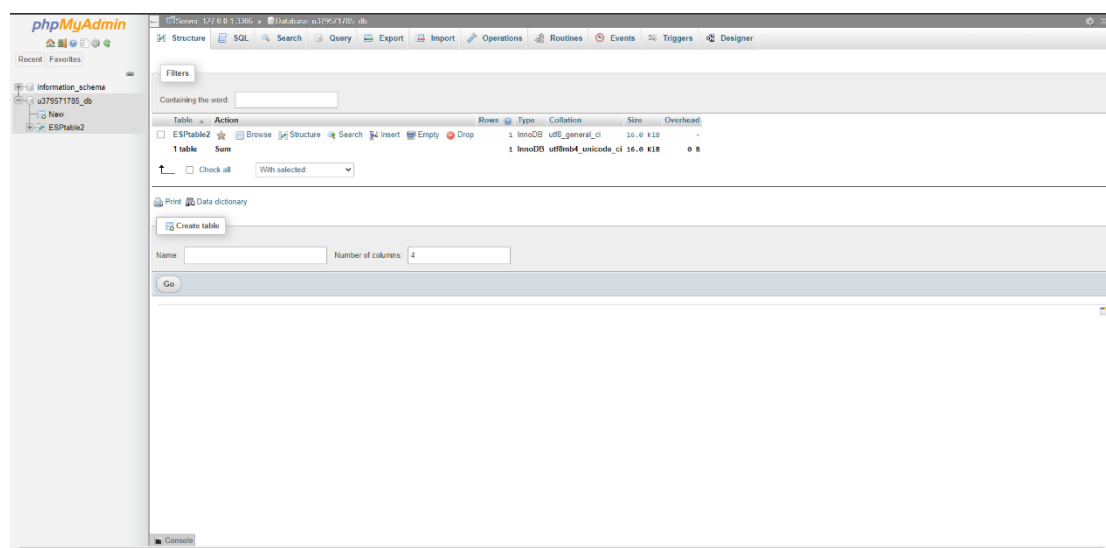


Figura 4.30. phpMyAdmin

Pe lângă serviciile de găzduire web, Hostinger oferă și alte servicii adiționale. Acestea includ găzduirea prin e-mail prin intermediul Titan Email și Google Workspace, înregistrarea și transferul domeniilor, certificatele SSL pentru securizarea site-urilor web și un constructor de site-uri web pentru a ajuta utilizatorii să-și creeze și să-și personalizeze propriile site-uri.

Pentru a asigura o disponibilitate și performanță optimă, Hostinger operează zece centre de date în opt țări din întreaga lume. Aceste țări includ Brazilia, Indonezia, India, Lituania, Olanda, Singapore, Regatul Unit și Statele Unite.

4.5. Soluția propusă

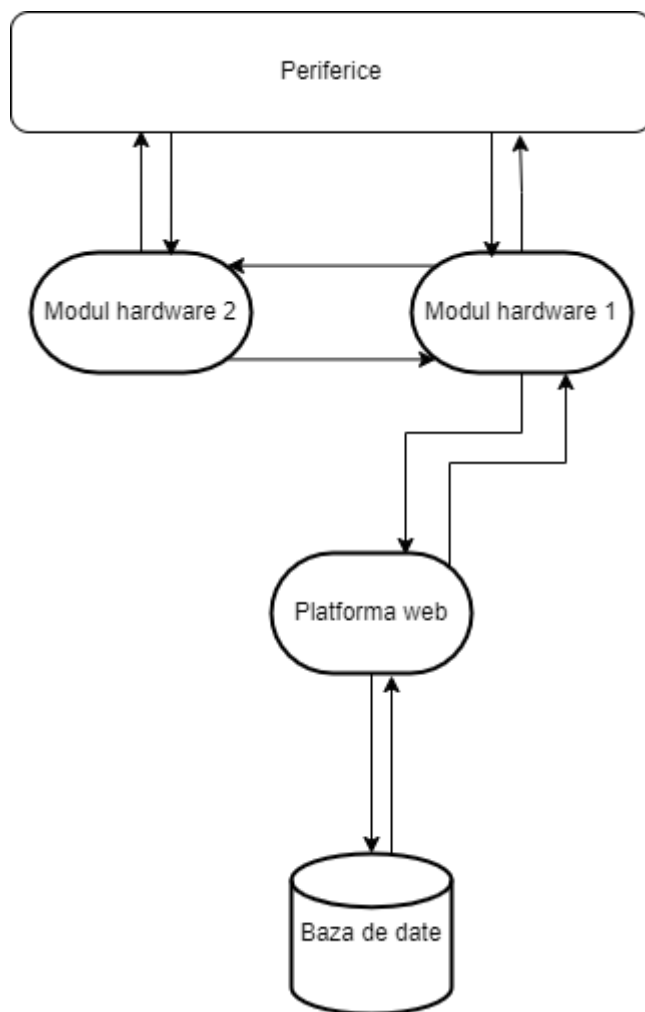


Figura 4.31. Diagrama abstractă a proiectului

4.5.1. Periferecele

Periferecele au rolul de a implementa fizic funcționalitățile sistemului. Acestea reprezintă componente electronice cu diferite proprietăți care servesc ca un mijloc de punere în aplicare a comenzilor primite de la modulele hardware.

Câteva exemple de periferice sunt diferiți senzori, servo-motoare, lumini etc.

4.5.2. Modulele hardware

Modulele hardware au funcția de a recepționa comenzile necesare și de a le transmite către periferice, precum și de a trimite datele obținute către platforma web. Acestea sunt reprezentate de către microcontrolere și sunt responsabile de asemenea și cu procesarea și prelucrarea datelor înainte să fie trimise către server.

Modulele hardware comunica între ele pentru distribuirea informațiilor și comenzilor, cu perifericele pentru implementarea funcționalităților cât și cu serverul pentru trimiterea și recepționarea datelor necesare pentru buna funcționare a sistemului.

4.5.3. Platforma web

Platforma web preia informațiile primite de la microcontrolerele și perifericele conectate și le afișează pe interfața grafică. De asemenea, interpretează schimbările de la butoane și textbox-uri și le transmite atât bazei de date cât și modulelor hardware, acționând ca o mediu principal de control și gestionare.

O altă funcționalitate este reprezentată de autentificarea utilizatorilor și asigurarea securității accesului la sistem. Prin intermediul paginii de login, utilizatorii pot autentifica și obține acces la funcționalitățile sistemului.

4.5.4. Baza de date

Baza de date are rolul de a stoca și gestiona datele sistemului. Aceasta permite accesul rapid la informații, asigură consistența și integritatea datelor și oferă suport pentru interogări și analize. De asemenea, permite sistemului să fie scalabil și performant în fața creșterii volumului de date și a traficului.

Capitolul 5. Proiectare de detaliu și implementare

5.1. Transmiterea datelor

5.1.1. Comunicarea între platforma web și microcontrolerul ESP8266

Comunicarea între cele două părți reprezintă un aspect crucial al integrității și funcționării proiectului. În cele ce urmează voi descrie modul de funcționare al secțiunii de cod care se ocupa cu acest aspect, atât pe partea platformei online, cât și în ceea ce privește microcontrolerul Lolin NodeMCU ESP8266.

5.1.1.1. Platforma web

În cadrul platformei web, fișierul denumit "TX.php" este responsabil cu transmiterea și recepționarea datelor.

Mai întâi se stabilește o conexiune cu baza de date prin includerea fișierului "database_connect.php". După ce a fost stabilită conexiunea, se verifică folosind funcția "isset" dacă au fost primite valori prin intermediul metodei POST. În caz afirmativ, prin intermediul unui query se actualizează tabela din baza de date cu valorile primite. Această secvență de cod se repetă pentru fiecare dintre cele trei valori care trebuie recepționate de la microcontroler.

```
include("database_connect.php");

if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

if (isset($_POST['value1'])) {
    $value1 = $_POST['value1'];
    mysqli_query($con, "UPDATE ESPTable2 SET SENT_NUMBER_1 = '$value1' WHERE id = 99999 AND PASSWORD = '12345'");
}
```

Figura 5.1. Primirea datelor dinspre ESP8266.

Pentru partea de transmitere, se execută un query în care se selectează toate informațiile prezente în baza de date, acestea fiind ulterior atribuite unei variabile. Pasul următor constă în atribuirea informației fiecărei coloane a variabilei corespunzătoare.

La final, se utilizează metoda echo pentru afișarea și trimiterea informațiilor sub forma "###\$variabila1##\$variabila2##\$variabila3##...##\$variabilan~". Acest format facilitează separarea și prelucrarea datelor odată ce sunt recepționate de către microcontroler. Simbolul "~" este utilizat pentru a semnală sfârșitul șirului de caractere trimis.

```
$result = mysqli_query($con,"SELECT * FROM ESPtable2"); |
while($row = mysqli_fetch_array($result)) {
    $b1 = $row['RECEIVED_BOOL1'];
    $b2 = $row['RECEIVED_BOOL2'];
    $b3 = $row['RECEIVED_BOOL3'];
    $b4 = $row['RECEIVED_BOOL4'];
    $b5 = $row['RECEIVED_BOOL5'];
    $b6 = $row['RECEIVED_BOOL6'];
    $b_usa = $row['RECEIVED_BOOL_USA'];
    $b_garaj = $row['RECEIVED_BOOL_GARAJ'];
    $b_geam1 = $row['RECEIVED_BOOL_GEAM1'];
    $b_geam2 = $row['RECEIVED_BOOL_GEAM2'];

    $b_smoke = $row['RECEIVED_BOOL_SMOKE'];

    $n1 = $row['RECEIVED_NUM1'];
    $n2 = $row['RECEIVED_NUM2'];
    $n3 = $row['RECEIVED_NUM3'];
    $n6 = $row['TEXT_1'];

    echo "##$b1##$b2##$b3##$b4##$b5##$b6##$b_usa##$b_garaj##$b_geam1##$b_geam2##$b_smoke##$n1##$n2##$n3##$n6-";
}
}
```

Figura 5.2. Trimiterea datelor inspre ESP8266.

5.1.1.2. Microcontrolerul Lolin NodeMCU ESP8266

În cadrul microcontrolerului Lolin NodeMCU ESP8266, mai întâi trebuie stabilită o conexiune la internet prin intermediul modulului WiFi ESP8266. Pentru crearea acesteia s-a folosit biblioteca „ESP8266WiFi.h” și s-au definit două macro-uri care reprezintă id-ul și parola rețelei la care dorim să conectăm dispozitivul. Apoi trebuie setat modul de funcționare al modulului WiFi al microcontrolerului ESP8266 în modul staționar (station mode) prin funcția „WiFi.mode(WIFI_STA);”. În acest mod, microcontrolerul poate să se conecteze la un punct de acces WiFi (router) folosind credențialele de autentificare (nume rețea și parolă) specificate. Pentru a începe conexiunea se folosește funcția „WiFi.begin(WIFI_SSID, WIFI_PASSWORD);” și se verifică statusul acesteia într-un while prin linia de cod „while (WiFi.status() != WL_CONNECTED)”. Această linie de cod este rulată atât în while-ul din secțiunea de setup care nu permite intrarea în program până la stabilirea unei conexiuni, cât și la începutul fiecărei bucle pentru a monitoriza constant starea rețelei.

```
WiFi.mode(WIFI_STA);
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

while (WiFi.status() != WL_CONNECTED)
{ //Serial.print(".");
  delay(500);
}
```

Figura 5.3. Stabilirea conexiunii WiFi în cadrul microcontrolerului ESP8266.

După ce exista o conexiune stabilită se declară două variabile numite „client” și „http1”, una de tipul „WiFiClient” și cealaltă de tipul „HTTPClient”. Pentru variabila „http1” se folosește funcția begin care are ca parametrii obiectul client prin intermediul căruia se va stabili legătura cu platforma web și URL-ul paginii web responsabile cu primirea și transmiterea de informații. Se adaugă informațiile necesare cu privire la structura informațiilor care se trimit prin intermediul funcției „http1.addHeader("Content-Type", "application/x-www-form-urlencoded");”. Aceasta este utilizată pentru a adăuga un antet (header) în cererea HTTP care specifică tipul de conținut al datelor trimise prin metoda POST. În acest caz, se specifică că datele vor fi trimise în formatul "application/x-www-form-urlencoded".

Formatul "application/x-www-form-urlencoded" este un tip de codificare a datelor utilizat în cererile HTTP, în special în cererile POST. Acest format implică codificarea datelor în perechi cheie-valoare, separate prin caracterul '&' și cu cheia și valoarea separate de caracterul '='. Această codificare este utilizată pentru a transmite datele către server într-un mod structurat și ușor de interpretat. Folosind codificarea menționată mai sus, se pun într-o variabilă de tip „String” valorile pe care dorim să le trimitem. Pentru trimiterea valorilor și recepționarea codului de răspuns s-a atribuit unei variabile de tip „int” funcția POST a variabilei http1 care are ca parametru valorile necesare în formatul menționat anterior. Dacă se primește un răspuns, se atribuie unei valori de tip String rezultatul funcției „http1.getString()”, aceasta reprezentând șirul de caractere trimis de către platforma web, urmând ca acesta să fie procesat.

```
if (WiFi.status() == WL_CONNECTED) {  
  
    WiFiClient client;  
    HTTPClient http1;  
    String postData;  
    http1.begin(client, serverName);  
    http1.addHeader("Content-Type", "application/x-www-form-urlencoded");  
    postData ="value1=" + String(temperatura) + "&value2=" + String(distanța)+ "&value3=" + String(smoke);  
    int httpCode = http1.POST(postData);  
  
    if (httpCode > 0) {  
  
        String payload = http1.getString();  
        ..  
        ..  
    }  
}
```

Figura 5.4. Comunicarea cu platforma în cadrul microcontrolerului ESP8266.

5.1.2. Comunicarea între cele două microcontrolere

Comunicarea între microcontrolerele Lolin NodeMCU ESP8266 și Arduino Uno este realizată prin protocolul UART. Conexiunea fizică între cele două plăci de dezvoltare este ilustrată în figura de mai jos.

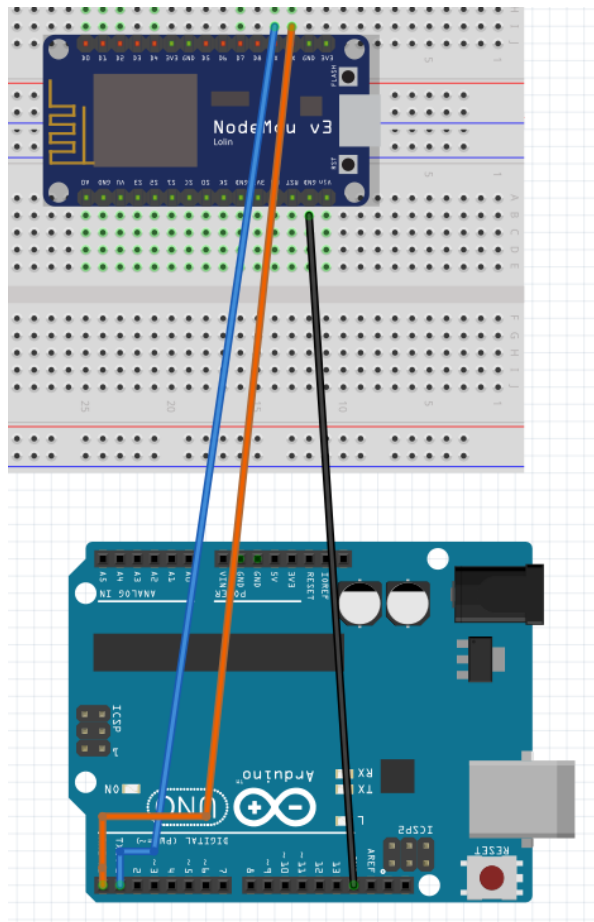


Figura 5.5. Conexiunea seriala între cele două microcontrolere.

Pentru transmiterea corectă și completă a informațiilor, ambele microcontrolere folosesc librăria „ArduinoJson.h” prin intermediul căreia creează documente JSON în care informația va fi stocată.

În cazul plăcuței ESP8266, mai întâi se creează un document dinamic JSON căruia îi sunt atribuite datele care trebuie trimise către Arduino. După atribuire, se folosește funcția „serializeJson(doc, Serial);” pentru serializarea informației și trimiterea acesteia pe portul serial. Următorul pas este parcurgerea unei bucle în care verificăm dacă există informații valabile pe portul serial, informații care reprezintă în cazul nostru răspunsul plăcuței Arduino. Aceste informații sunt atribuite unei variabile și incluse ca parametru al funcției „deserializeJson(doc1, message);”. Se creează un nou document JSON, iar rezultatul acestei funcții este atribuit unei variabile de tipul „DeserializationError”. Dacă există o eroare, o afișăm pe portul serial și dăm return, altfel se procesează informațiile primite.

```
void comunicatie_seriala(int b6,int b_usa,int b_garaj,int b_smoke ,int b_geam1, int b_geam2 ,String text) {
    Serial.flush();
    DynamicJsonDocument doc(1024);

    doc["type"] = "request";
    doc["b6"] = b6;
    doc["b_usa"] = b_usa;
    doc["b_garaj"] = b_garaj;
    doc["b_smoke"] = b_smoke;
    doc["b_geam1"] = b_geam1;
    doc["b_geam2"] = b_geam2;
    doc["text"] = text;

    serializeJson(doc, Serial);
    doc.clear();

    boolean messageReady = false;
    String message = "";

    while (messageReady == false) {
        if (Serial.available()) {
            message = Serial.readString();
            messageReady = true;
        }
    }
    Serial.flush();
    DynamicJsonDocument doc1(256);

    DeserializationError error = deserializeJson(doc1, message);
    if (error) {
        Serial.print(F("deserializeJson() failed: "));
        Serial.println(error.c_str());
        return;
    }
    distanta = doc1["distanta"];
    smoke = doc1["smoke"];
    temperatura =doc1["temperatura"];

    String output = "distanta: " + String(distanta) + "\n";
    output += "Smoke level: " + String(smoke)+ "\n";
    output += "Temperature: " + String(temperatura);

    Serial.println();
    Serial.println(output);
    Serial.println();
    doc1.clear();
    Serial.flush();
}
```

Figura 5.6. Cod pentru stabilirea conexiunii seriale (ESP8266).

În cazul plăcuței Arduino Uno există exact același proces, cu diferența că aici mai întâi se monitorizează portul serial pentru informații, iar apoi se trimit valorile obținute de către senzori prin intermediul obiectului JSON.

Din cauza ordinii de monitorizare se creează un sistem în care fiecare microcontroler trimite informația necesară, după care așteaptă un răspuns din cealaltă parte, sincronizând transmiterea de valori fără să ocupe portul necesar astfel încât să apară conflicte.

```

void receptie_seriala()
{
    Serial.flush();
    while (mReady==false) {
        if(Serial.available())
        {
            msg=Serial.readString();
            mReady=true;
        }

    }
    Serial.flush();
    DynamicJsonDocument doc1(512);
    DeserializationError error = deserializeJson(doc1,msg);
    if(error)
    {
        Serial.println(error.c_str());
    }

    b6 = doc1["b6"];
    b_usa = doc1["b_usa"];
    b_garaj = doc1["b_garaj"];
    b_smoke = doc1["b_smoke"];
    b_geam1 = doc1["b_geam1"];
    b_geam2 = doc1["b_geam2"];
    text = (const char *)doc1["text"];
    doc1.clear();
    Serial.flush();

}

void trimitere_seriala()
{
    DynamicJsonDocument doc(128);
    doc["type"] = "response";
    doc["distanta"] = valDist;
    doc["smoke"] = valSmoke;
    doc["temperatura"] = valTemp;

    serializeJson(doc, Serial);
    Serial.println();
    mReady = false;
    doc.clear();
    Serial.flush();
}

```

Figura 5.7. Cod pentru stabilirea conexiunii seriale (Arduino Uno).

5.2. Procesarea datelor

5.2.1. Procesarea informațiilor pe platforma web

5.2.1.1. Pagina „update_values.php”

Această pagină este responsabilă cu actualizarea bazei de date.

Mai întâi această preia în variabile, prin intermediul metodei POST, o valoare și locația acesteia în baza de date, după care se asigură că există o conexiune cu baza de date prin includerea fișierului "database_connect.php". Dacă conexiunea la baza de date nu este stabilită cu succes, se afișează un mesaj de eroare, în caz contrar se utilizează funcția `mysqli_query()` pentru a executa o interogare prin intermediul căreia se actualizează datele primite prin POST în baza de date. La final, se utilizează funcția `header()` pentru a redirecționa utilizatorul înapoi către pagina "index.php".

```
<?php

include("database_connect.php");

if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

$value = $_POST['value']; //Retinem valoarea
$unit = $_POST['unit'];   //Retinem id-ul
$column = $_POST['column']; //Retinem coloana la care schimbam valoarea

mysqli_query($con,"UPDATE ESptable2 SET $column = '{$value}'
WHERE id=$unit");

header("location: index.php");
?>
```

Figura 5.8. Codul paginii “update_values.php”.

5.2.1.2. Login

La începutul codului, este inițializată o sesiune utilizând funcția "session_start()", după care se verifică statusul acestei sesiuni. Dacă utilizatorul este încă autentificat în aplicație, se trece direct la pagina principală care conține interfața grafică, altfel se include fișierul "database_connect.php" pentru a se stabili o conexiune cu baza de date care conține credențialele valide. Aceste credențiale sunt extrase, după care se verifică dacă utilizatorul a trimis formularul de autentificare prin apăsarea butonului "Log In". Se preiau valorile introduse în câmpurile "username" și "password" și se compară cu valorile stocate în baza de date. În cazul în care acestea sunt corecte, se setează variabila de sesiune `$_SESSION['loggedIn']` la true, semnalând că utilizatorul este autentificat, și se realizează o redirecționare către pagina "index.php" utilizând funcția `header()`. Apelul funcției "exit" asigură oprirea ulterioară a procesării scriptului. În cazul în care utilizatorul introduce un nume de utilizator sau o parolă incorecte, se afișează mesajul "invalid username and/or password".

5.2.1.3. Butoanele și textbox-urile interfeței grafice

Principiul de prelucrare a datelor atât în cazul butoanelor, cât și în cazul text-box-urilor este asemănător. Mai întâi se stabilește o conexiune la baza de date și, utilizând funcția

"mysqli_query()", se realizează o interogare pentru a selecta toate înregistrările din tabela "ESPTable2". Rezultatul interogării este stocat în variabila \$result. Se creează un tabel care conține fie butoane, fie text-box-uri. Următorul pas este realizarea unei bucle "while" prin intermediul căreia se extrag informațiile necesare din baza de date și se stabilește starea curentă a butonului sau valoarea curentă pentru text-box.

În cazul text-box-ului, se preia valoarea scrisă și se actualizează baza de date prin intermediul paginii "update_values.php". Pentru butoane, procesul este asemănător, cu diferența că se verifică dacă butonul este apăsător sau nu, iar în momentul în care acesta este acționat, se actualizează baza de date cu inversul valorii curente.

```
<?php
include("database_connect.php");

if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
$result = mysqli_query($con,"SELECT * FROM ESPTable2");

echo "<table style='font-size: 30px;'"
    <thead>
        <tr>
            <th>Numeric controls</th>
        </tr>
    </thead>

    <tbody>
        <tr>
            <td>Set max CO2 level</td>
        </tr>
    ";

while($row = mysqli_fetch_array($result)) {

    echo "<tr>";

    $coloana7 = "RECEIVED_NUM2";

    $numar_curent2 = $row['RECEIVED_NUM2'];

    echo "<td><form action= update_values.php method= 'post'>
        <input type='text' name='value' style='width: 120px;' value=$numar_curent2 size='15' >
        <input type='hidden' name='unit' style='width: 120px;' value=$unit_id >
        <input type='hidden' name='column' style='width: 120px;' value=$coloana7 >
        <input type= 'submit' name= 'change_but' style='text-align:center' value='change'></form></td>";

    echo "</tr>
    </tbody>";

}
echo "</table>
<br>
";
?>
```

Figura 5.9. Exemplu de procesare a datelor pentru un textbox.

5.2.2. Procesarea informațiilor în microcontrolere

5.2.2.1. Procesarea informațiilor în placa de dezvoltare Lolin NodeMCU ESP8266

Placa de dezvoltare Lolin NodeMCU ESP8266 este responsabilă cu majoritatea procesării datelor la nivelul proiectului deoarece este componenta centrală care face legătura între platforma web și partea fizică.

După cum am menționat și în capitolul de comunicare, aceasta primește datele ca un string de forma: "\$variabila1##\$variabila2##\$variabila3##...##\$variabilan~". Mai întâi, acest string este convertit în "char*" pentru a fi compatibil cu funcția strtok. Se folosește această funcție cu separatorul "##" pentru a prelua prima valoare într-o variabilă. Apoi se intră într-o buclă de tip while, a cărei condiție verifică dacă valoarea citită cu strtok este nulă. Dacă este nulă, se oprește, altfel se incrementează o valoare auxiliară, a cărei valoare a fost inițializată la 0 la începutul buclei principale, pentru a ține evidența numărului de variabilă la care s-a ajuns și se apelează funcția "asignare_valori()". La final, variabilei care reține valoarea curentă ii se atribuie "strtok(NULL, "##");" pentru a trece la următorul element.

```
String payload = http1.getString();
//Serial.println(payload);
values = strtok ((char*)payload.c_str() , "##");

while (values != NULL)
{
    aux++;
    asignare_valori(aux, values);
    values = strtok(NULL, "##");
}
}
```

Figura 5.10. Procesarea datelor și separarea în tokeni.

Funcția "asignare_valori" nu returnează nimic și are ca parametri numărul variabilei curente și valoarea acesteia. Prin intermediul unor if-uri care verifică numărul variabilei, se alege modul de procesare a datelor.

Pentru valorile numerice care trebuie transmise către Arduino, se folosește funcția "atoi()" pentru conversia de la String la număr, și se atribuie variabilei corespunzătoare.

În cazul textului, înainte ca valoarea să fie atribuită unei variabile, se folosește funcția "strtok(values, "~");" pentru a elimina spațiul liber care este transmis după string-ul de date.

Pentru valorile folosite pentru controlul iluminării, se apelează funcția "setBoolAndLED()" care are ca și parametri valoarea care indică starea curentă a LED-ului și LED-ul corespunzător. Această funcție aprinde sau stinge lumina în funcție de starea primită ca parametru.

```
void asignare_valori(int aux, char * values )
{

    if (aux == 1) {
        setBoolAndLED(values, led1);
    }
    if (aux == 2) {
        setBoolAndLED(values, led2);
    }
    if (aux == 3) {
        setBoolAndLED(values, led3);
    }
    if (aux == 4) {
        setBoolAndLED(values, led4);
    }
    if (aux == 5) {
        setBoolAndLED(values, led5);
    }

    if (aux == 6) {
        b6 = atoi(values);
        if (b6 == 1)
        {
            digitalWrite(LED_Alarma, HIGH);
        }
        else
        {
            digitalWrite(LED_Alarma, LOW);
        }
    }

    if (aux == 7) {
        b_usa = atoi(values);
    }
    if (aux == 8) {
        b_garaaj = atoi(values);
    }
    if (aux == 9) {
        b_geam1 = atoi(values);
    }
    if (aux == 10) {
        b_geam2 = atoi(values);
    }
    if (aux == 11) {
        b_smoke = atoi(values);
    }
    if (aux == 13) {
        n2Smoke = atoi(values);
    }
    if(aux == 15){
        valuesAux = strtok (values , "~");
        text = valuesAux;
    }

}
```

Figura 5.11. Functia asignare valori.

5.2.2.2. Procesarea informațiilor în placa de dezvoltare Arduino Uno

Placa de dezvoltare Arduino Uno este responsabilă cu procesarea datelor primite de la senzorii de proximitate și temperatură. Această procesare este descrisă în subcapitolul 5.3 la secțiunile 5.3.3 și 5.3.5. Senzorul de gaz nu necesită o procesare specială a datelor, fiind suficientă citirea valorii cu funcția "analogRead()".

5.3. Funcționalitățile fizice ale sistemului

Când utilizatorul interacționează cu butoanele și câmpurile text, valorile variabilelor corespunzătoare acestora se actualizează în funcție de acțiunea utilizatorului. Aceste valori sunt actualizate și în baza de date de unde sunt luate și mai apoi trimise către ESP8266 prin metoda POST a protocolului HTTP, pentru a fi procesate și atribuite unor variabile din microcontroler și mai apoi folosite. În cazul variabilelor necesare plăcii de dezvoltare Arduino Uno, utilizând un format JSON, aceste valori sunt încapsulate, serializate și transmise prin portul serial folosind protocolul UART. La sosirea datelor în Arduino, acestea sunt deserializate și fiecare element este atribuit unei variabile corespunzătoare. Aceste variabile pot reprezenta statusul butoanelor sau valorile introduse în câmpurile text, permițând controlul sistemului în funcție de acțiunile utilizatorului.

5.3.1. Controlul încuietorilor și ferestrelor

Controlul încuietorilor și ferestrelor reprezintă una dintre funcționalitățile cheie ale sistemului smart house controlat de la distanță. Acest control este realizat prin comenzi către micro servomotoarele SG-90 a căror status stabilește de asemenea și aprinderea LED-ului corespunzător servomotorului în acțiune. Servomotoarele acționează fie ca și un lacăt, fie ca și un mecanism de deschidere și închidere a ferestrelor/ușii de la garaj. În figura 5.1 este prezentată conexiunea perifericelor cu placa de dezvoltare.

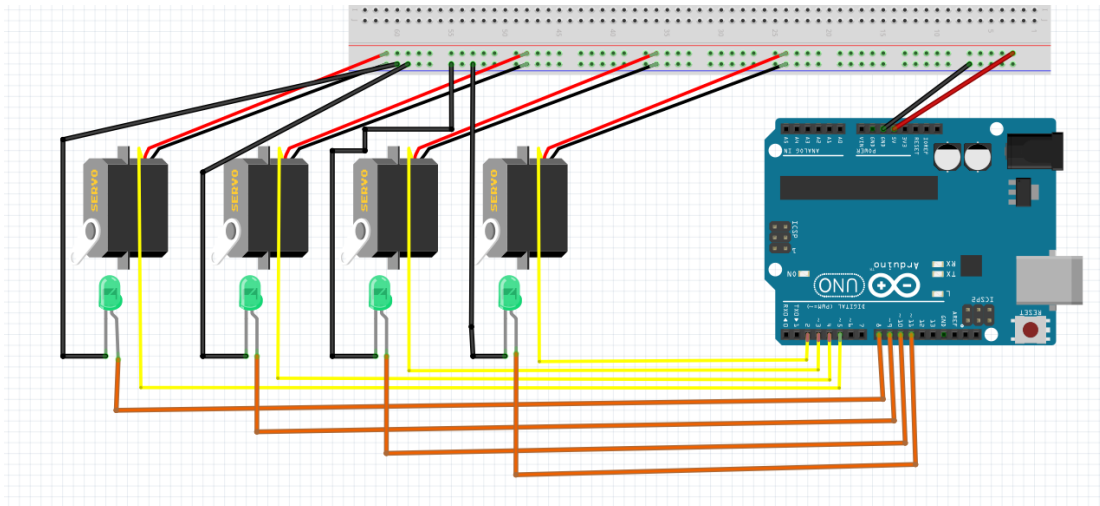


Figura 5.12. Implementarea fizică a controlului încuietorilor și ferestrelor

Pentru controlul micro servomotorului s-a folosit librăria „Servo.h” care permite conectarea unui servomotor la plăcuța de dezvoltare Arduino Uno. S-au definit patru obiecte globale de tip Servo: „servo_usa”, „servo_garaj”, „servo_geam1” și „servo_geam2”. De asemenea, s-au declarat patru constante de tip „int” pentru a specifica pinul la care fiecare servomotor este conectat la placa de dezvoltare. Aceste constante sunt denumite "usa", "garaj", "geam1" și "geam2" și li s-au atribuit valorile 2, 3, 4 și 5, în ordine.

Pentru atașarea servomotoarelor s-a utilizat funcția „attach(pin)”, unde parametrul „pin” a fost înlocuit cu constanta corespunzătoare fiecărui micro servomotor. De asemenea s-a folosit funcția de „pinMode(pin, OUTPUT)” pentru a seta cei 4 pini ai LED-urilor corespunzătoare motoarelor ca si output si funcția digitalWrite() pentru a le inițializa cu valoarea LOW. Variabilele sunt numite LED_usa, LED_garaj, LED_geam1, LED_geam2 si cu valorile 11, 10, 9, 8 în ordine si fiecare dintre acestea este legata în serie la o rezistență de 220 de ohm pentru protecție.

Variabilele în care sunt primite informațiile cu privire la statusul acestor periferice se numesc "b_usa", "b_garaj", "b_geam1", "b_geam2". Când execuția ajunge la funcția "management_date()", valorile acestor variabile sunt verificate folosind "if"-uri. Dacă valoarea este 0, se utilizează funcția digitalWrite(led, LOW) pentru a stinge LED-ul corespunzător motorului, iar apoi se utilizează funcția write(0) pentru a seta numărul de grade și implicit poziția la care dorim să ajungă motorul, în acest caz, poziția fiind 0. În cazul în care valoarea este 1, se setează LED-ul la HIGH și poziția motorului la 90 de grade.

```
//-----//
//----- Motoare servo -----//
if(b_usa==1 ){
    digitalWrite(LED_usa,HIGH);
    servo_usa.write(90);
}else if(b_usa==0){
    digitalWrite(LED_usa,LOW);
    servo_usa.write(0);
}

if(b_garaj==1){
    digitalWrite(LED_garaj,HIGH);
    servo_garaj.write(90);
}else if(b_garaj==0){
    digitalWrite(LED_garaj,LOW);
    servo_garaj.write(0);
}

if(b_geam1==1){
    digitalWrite(LED_geam1,HIGH);
    servo_geam1.write(90);
}else if(b_garaj==0){
    digitalWrite(LED_geam1,LOW);
    servo_geam1.write(0);
}

if(b_geam2==1){
    digitalWrite(LED_geam2,HIGH);
    servo_geam2.write(90);
}else if(b_garaj==0){
    digitalWrite(LED_geam2,LOW);
    servo_geam2.write(0);
}
```

Figura 5.13. Cod pentru controlul servomotoarelor si luminilor asociate

5.3.2. Alarma

Alarma, în proiectul curent, este reprezentată de semnale luminoase și acustice care pot fi activate de către două componente ale sistemului, și anume senzorul de proximitate și

senzorul de fum și gaz, în cazul în care valorile returnate de aceștia depășesc sau se află sub un anumit prag.

Aceasta este implementată prin intermediul a două componente simple și anume un buzzer pasiv și un led roșu. În figura 5.2 este prezentată conexiunea perifericelor cu placa de dezvoltare.

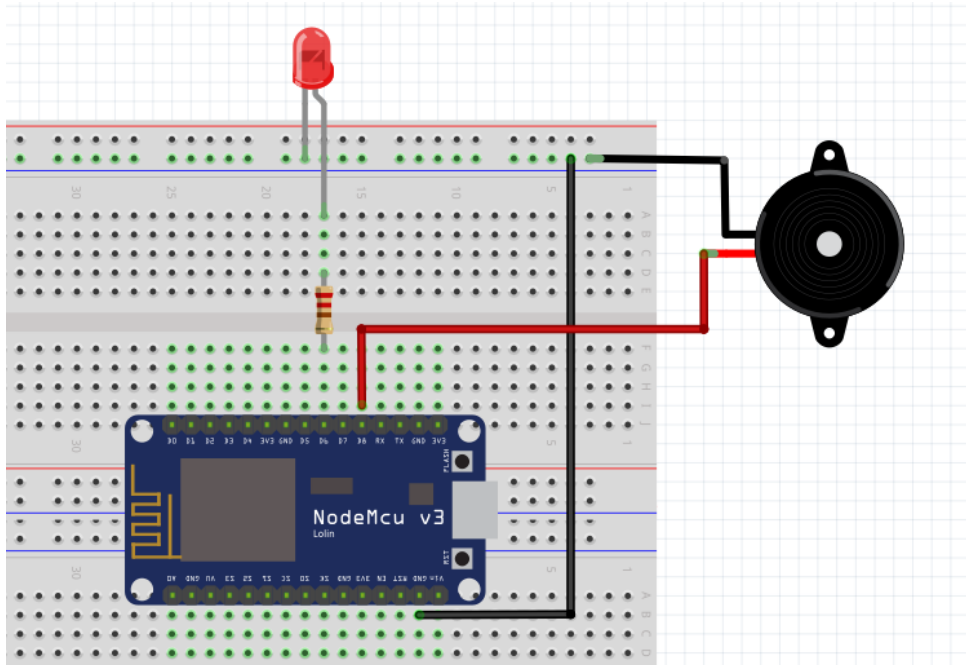


Figura 5.14. Implementarea fizică a alarmei

Pentru controlul acestei funcționalități s-au declarat două variabile globale de tipul "int" numite LED_Alarma cu valoarea D6 pentru LED și Alarma cu valoarea D8 pentru buzzer. Funcția alarma este compusă dintr-un for care merge de la 0 până la 4 în care inițial se aprinde led-ul roșu, după care prin intermediul funcției "tone" se activează buzzerul, se așteaptă 200 de milisecunde prin funcția "delay(200);" apoi se stinge LED-ul și se oprește buzzerul prin funcția "noTone()". După ce se așteaptă încă 200 de milisecunde se reia bucla..

```
//-----  
//-----  
void alarma()  
{  
  for(int i = 0;i<5;i++)  
  {  
    digitalWrite(LED_Alarma,HIGH);  
    tone(Alarma,700);  
    delay(200);  
    digitalWrite(LED_Alarma,LOW);  
    noTone(Alarma);  
    delay(200);  
  }  
}  
//-----
```

Figura 5.15. Cod pentru implementarea alarmei.

5.3.3. Monitorizarea proximității

Monitorizarea proximității este un aspect important al părții de securitate și lucrează în strânsă relație cu funcționalitatea de alarmă. Această monitorizare este realizată de către

senzorul cu ultrasunete HC-SR04 care trimite unde sonore și ascultă după ecoul produs de coliziunea acestora cu un obiect. În funcție de timpul în care ecoul se întoarce, este calculată distanța până la obiectul respectiv. Astfel, acest senzor poate acționa ca un detector de mișcare și proximitate. În figura 5.2 este prezentată conexiunea senzorului cu placa de dezvoltare.

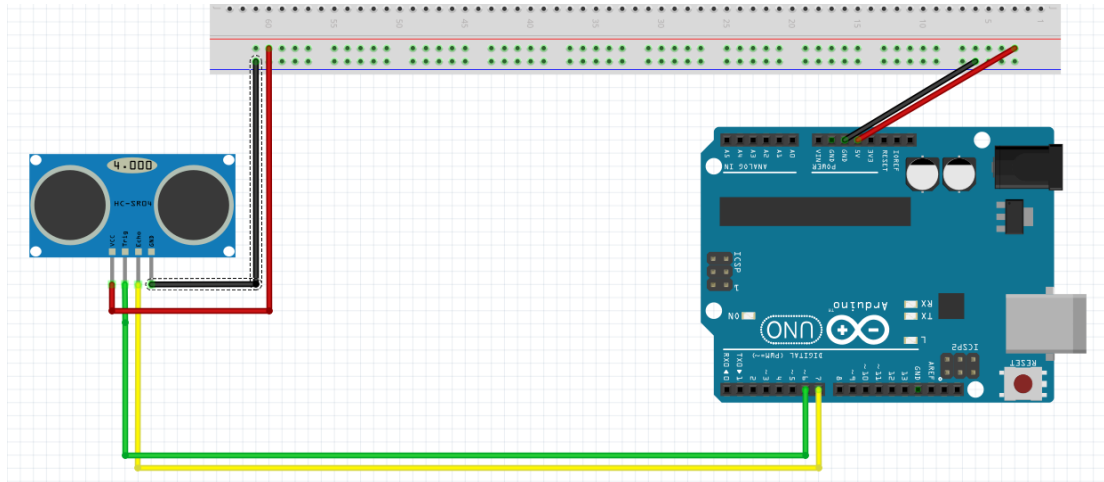


Figura 5.16. Implementarea fizică a senzorului de proximitate

Pentru a primi informații de la senzor, s-au definit două constante globale "trig" și "echo" cu valorile 6, respectiv 7. Apoi, "trig" a fost setat ca output și "echo" ca input prin intermediul funcției "pinMode()".

Variabila în care este primită informația care indică dacă senzorul a fost activat în interfața grafică se numește "b6". Când execuția ajunge la funcția "management_date()", valorile acestei variabile sunt verificate folosind un "if". Dacă aceasta este 0, valoarea distanței este pusă implicit la 0. Dacă valoarea este 1, pinului "trig" îi este atribuită valoarea LOW prin funcția "digitalWrite", după care, după 2 microsecunde contorizate cu funcția "delayMicroseconds(2)", valoarea este pusă la HIGH. După încă 10 microsecunde se pune iar pe LOW. Pentru a obține informația de care este nevoie, se atribuie variabilei "valDist = (pulseIn(echo, HIGH))/29/2;", care reprezintă distanța în centimetri până la cel mai apropiat obiect.

```
//----- Senzor de distanta -----//
//----- Senzor de distanta -----//
if(b6==1)
{
    pinMode(trig, OUTPUT);
    digitalWrite(trig, LOW);
    delayMicroseconds(2);
    digitalWrite(trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(trig, LOW);
    pinMode(echo, INPUT);
    valDist = (pulseIn(echo, HIGH))/29/2;
}
else{
    valDist=0;
}
}
```

Figura 5.17. Cod pentru implementarea senzorului de proximitate.

Această informație este memorată într-o variabilă și atribuită unui element al unui obiect JSON în funcția "trimitere_seriala()". Acest obiect este serializat și trimis înapoi către ESP8266, unde informația este deserializată. Dacă distanța primită prin intermediul obiectului

JSON este mai mică de 12 și mai mare de 0 (care este valoarea primită când sistemul este dezactivat), este apelată funcția "alarma".

5.3.4. Monitorizarea calității aerului

Monitorizarea fumului și gazelor este un aspect important al sistemului de securitate și este strâns legată de funcționalitatea alarmei. Această monitorizare este realizată de către senzorul de fum și gaze MQ2, care detectează prezența acestor substanțe prin intermediul elementelor chimice senzitive. Senzorul MQ2 monitorizează schimbările de rezistență cauzate de prezența fumului sau a gazelor inflamabile. În funcție de nivelul detectat, senzorul schimbă tensiunea de ieșire și valorile acestea pot fi interpretate de sistem. Conexiunea senzorului MQ2 cu placa de dezvoltare este prezentată în figura 5.2.

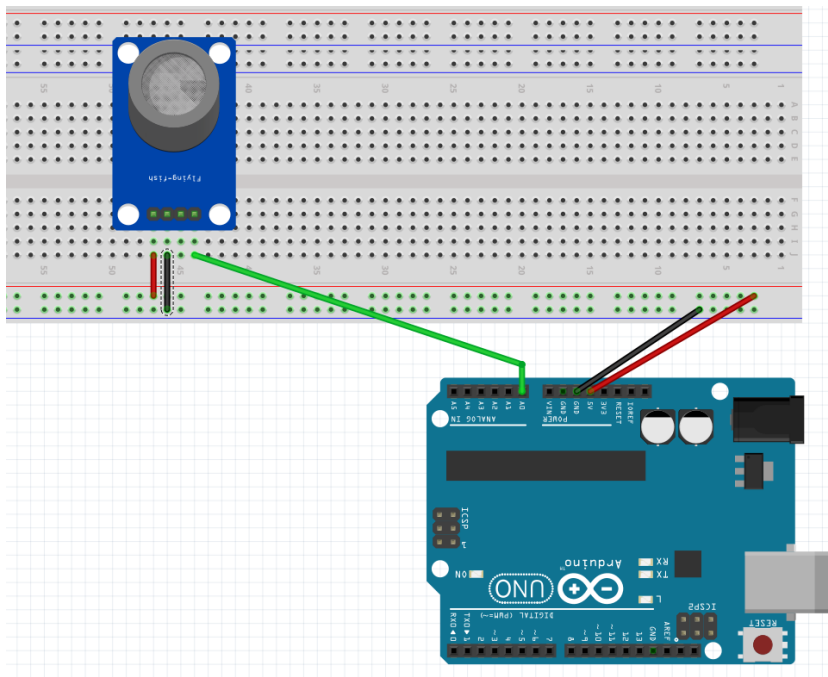


Figura 5.18. Implementarea fizică a detectorului de fum și gaze

Mai întâi, utilizatorul folosește interfața grafică pentru a scrie în textbox valoarea pragului de la care să se activeze alarma în cazul în care detectorul de fum este activat. După scrierea acestei valori, se apasă butonul "change" care va actualiza baza de date, astfel noua valoare va fi trimisă către microcontrollerul ESP8266.

Pentru a primi informații de la senzor, s-a definit o constantă globală denumită "smoke" cu valoarea A0 și a fost setată ca input prin intermediul funcției "pinMode()". Variabila în care este primită informația care ne indică dacă senzorul a fost activat în interfața grafică se numește "b_smoke". Când execuția ajunge la funcția "management_date()", valoarea acestei variabile este verificată folosind un "if". Dacă aceasta este 0, valoarea senzorului este pusă implicit la 0. Dacă valoarea este 1, se atribuie variabilei "valSmoke" rezultatul funcției "analogRead(smoke);".

```
//----- Senzor de fum -----//
//----- Senzor de fum -----//
if(b_smoke==1){
    valSmoke = analogRead(smoke);
}
else{
    valSmoke =0;
}
}
```

Figura 5.19. Cod pentru implementarea detectorului de fum.

Această informație este atribuită unui element al unui obiect JSON în funcția "trimitere_seriala()". Acest obiect este serializat și trimis înapoi către ESP8266, unde informația este deserializată. Dacă valoarea primită prin intermediul obiectului JSON este mai mare decât pragul impus de către utilizator, este apelată funcția de alarmă.

5.3.5. Monitorizarea confortului termic

Monitorizarea temperaturii asigură faptul că utilizatorul este conștient și informat cu privire la confortul termic al imobilului. Această monitorizare este realizată de către senzorul de temperatură LM35, care măsoară temperatura ambientală și o convertește într-un semnal analogic proporțional cu valoarea temperaturii. Semnalul analogic este apoi citit și interpretat de către sistem. Conexiunea senzorului LM35 cu placa de dezvoltare este prezentată în figura 5.2.

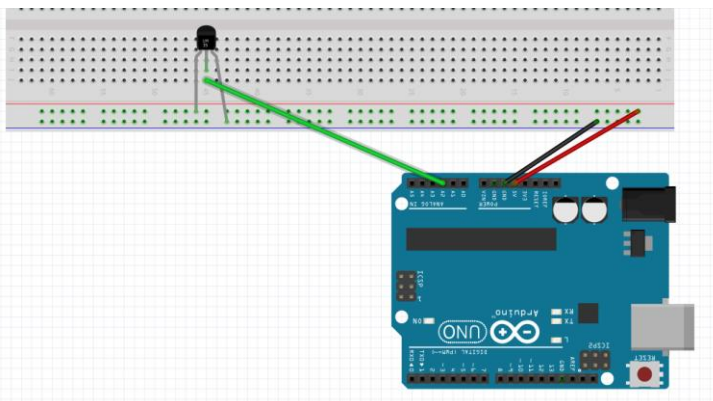


Figura 5.20. Implementarea fizica a senzorului de temperatura

Pentru a primi informații de la senzor, s-a definit o constantă globală denumită "temperatura" cu valoarea A2 și a fost setată ca input prin intermediul funcției "pinMode()".

În funcția "management_date()", este pusă într-o variabilă auxiliară valoarea obținută în urma funcției "analogRead(temperatura);". Apoi, este înmulțită cu (4.3/1023.0), unde 4.3 reprezintă tensiunea de alimentare și 1023 reprezintă intervalul de valori (de la 0 la 1023) al convertorului analog-digital. Valoarea obținută reprezintă milivolții citiți și este înmulțită cu 100 pentru a fi convertită în grade Celsius.

```
//----- Setare temperatura -----//
double ten= analogRead(temperatura);
ten=ten*(4.3/1023.0);
valTemp = ten*100;
//-----//
```

Figura 5.21. Cod pentru citirea temperaturii in grade Celsius

Temperatura este mai departe atribuită unui element al unui obiect JSON în funcția "trimitere_seriala()". Acest obiect este serializat și trimis înapoi către ESP8266, unde informația este deserializată și trimisă pentru a fi afișată pe interfața grafică.

5.3.6. Trimiterea mesajelor

Această funcționalitate este realizată de către LCD 20x4 2004A. Acesta primește un text de maxim 35 de caractere, inclusiv spațiile libere, prin intermediul modulului I2C atașat și îl afișează pentru a transmite mesajul scris de către utilizator. Conexiunea ecranului cu placa de dezvoltare este prezentată în figura 5.2.

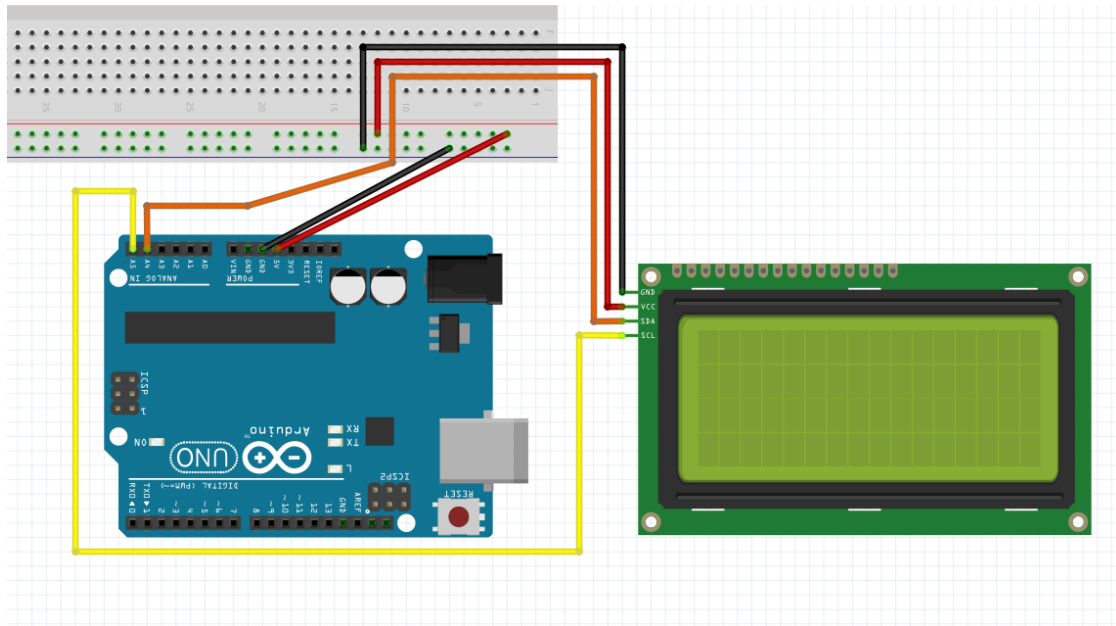


Figura 5.22. Implementarea fizică a ecranului LCD

Este de menționat faptul că limita de 35 de caractere este impusă de dimensiunea redusă a memoriei microcontrollerului, nu de capacitatea ecranului folosit.

Libraria folosită este "LiquidCrystal_I2C.h". S-a declarat o variabilă globală de tipul "LiquidCrystal_I2C" care are forma "lcd(0x27, 20, 4)", unde "lcd" este numele variabilei, "0x27" este adresa I2C, "20" este numărul de coloane, iar "4" este numărul de rânduri al ecranului. În partea de setup, se folosește funcția "init()" pentru inițializarea ecranului și funcția "backlight()" pentru pornirea luminii de fundal.

Informația vine dinspre platforma web în microcontrollerul ESP8266, unde este împărțită în tokeni și atribuită fiecărei variabile corespunzătoare. În cazul textului, înainte ca variabila "text" să primească valoarea respectivă, se apelează funcția strtok având ca parametrii de intrare textul primit de la platforma web și semnul "~". Această secvență de cod este folosită pentru a elimina spațiul liber redundant care vine după text și pentru a eficientiza programul din punct de vedere al memoriei. După ce textul este procesat, este trimis înspre Arduino Uno, unde după deserializare este atribuit altei variabile și convertit în "char*" pentru a putea fi afișat. S-a folosit funcția "setCursor" pentru a pune cursorul pe rândul 1 și coloana 0. După așezarea cursorului, s-a folosit funcția "clear()" pentru a elimina textul afișat anterior și cu funcția "print()" se afișează textul.

```
//----- Setare text -----//
value = (char *)text.c_str();
lcd.setCursor(1,0);
lcd.clear();
lcd.print(value);

}
```

Figura 5.23. Cod pentru afisarea mesajelor pe LCD.

5.3.7. Controlul iluminării imobilului.

Controlul iluminării ajută la simplificarea managementului de energie și la confortul utilizatorului. Acest control este realizat prin comenzi către LED-urile corespunzătoare fiecărei camere. Ele sunt utilizate pentru a ilumina diferite zone sau încăperi în funcție de nevoi și preferințe. Conexiunea LED-urilor cu placa de dezvoltare este prezentată în figura 5.1.

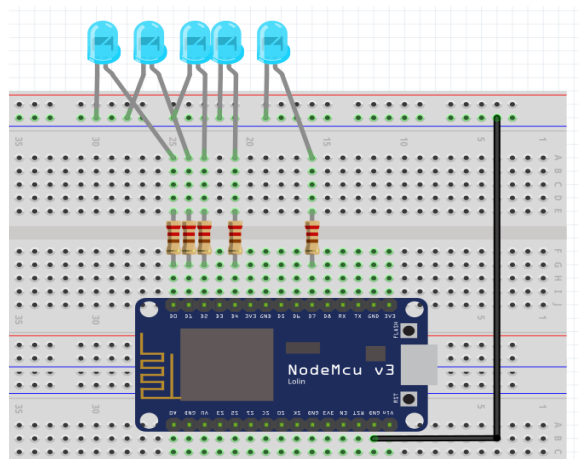


Figura 5.24. Implementarea fizică a iluminării

Implementarea controlului de iluminare este relativ simplă. După ce s-au obținut datele cu privire la starea butoanelor de control, s-a folosit funcția "digitalWrite()" pentru a seta LED-urile în starea potrivită

```
void setBoolAndLED(char *values, int ledAux)
{
    int b=atoi(values);
    if (b == 1)
    {
        digitalWrite(ledAux, HIGH);
    }
    else
    {
        digitalWrite(ledAux, LOW);
    }
}
```

Figura 5.25. Cod pentru aprinderea unui LED.

5.4. Soluția propusă

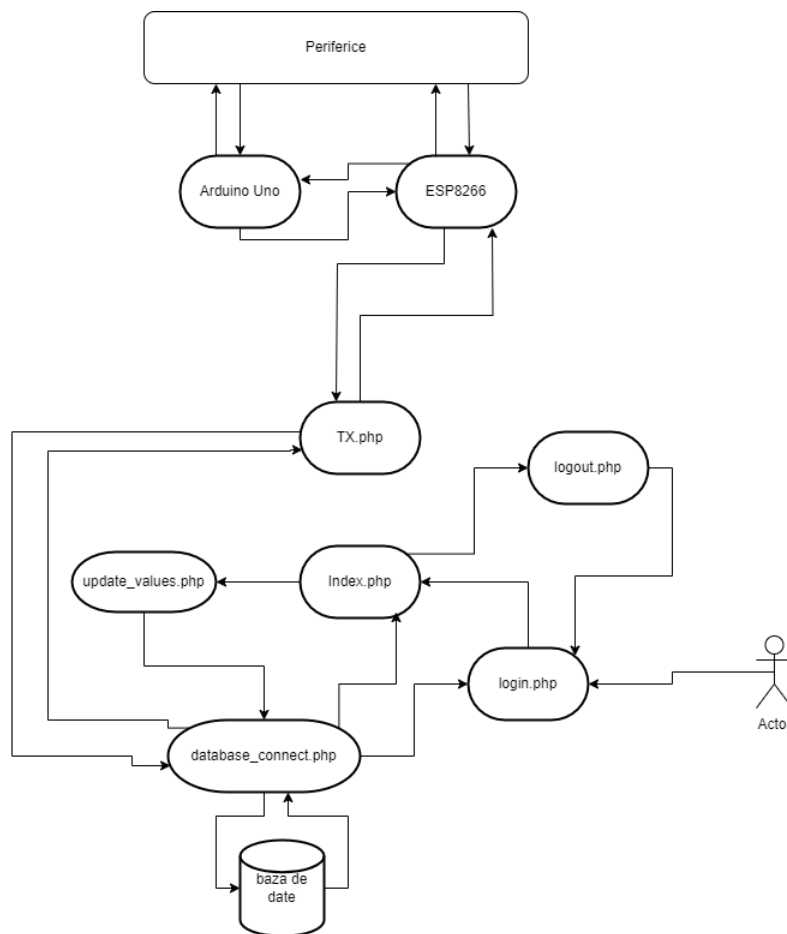


Figura 5.26. Soluția propusă

5.4.1. Perifericele

În cazul proiectului curent, perifericele și rolul lor sunt:

- Micro servomotorul SG-90 are rolul de a acționa încuietoarea de la ușa principală și de a deschide/închide atât ferestrele, cât și ușa de la garaj.
- Senzorul ultrasonic HC-SR04 are rolul de detector de mișcare și senzor de proximitate pentru ușa din față.
- Senzorul de fum și gaz MQ-2 este responsabil cu monitorizarea calității aerului.
- Senzorul de temperatură LM35 este responsabil cu monitorizarea confortului termic.
- Ecranul LCD 20x4 2004A împreună cu modulul serial I2C au rolul de a afișa mesajele scrise de utilizator pe interfața grafică.
- Buzzerul pasiv are rolul de semnal acustic pentru avertizarea utilizatorului în cazul în care cineva se apropie prea mult de ușa din față sau pragul de detectare a gazelor inflamabile și fumului este depășit.
- LED-urile au rol atât de semnal vizual în cazul activării alarmei, cât și pentru iluminare.

5.4.2. Modulele hardware

Există două module hardware utilizate:

Placa de dezvoltare Lolin NodeMCU ESP8266 are patru roluri principale în proiectul curent. Implementarea alarmei, iluminării, asigurarea comunicării între componentele existente și procesarea datelor.

Arduino Uno are ca rol implementarea restului funcționalităților, deoarece are mai mulți pini atât digitali, cât și analogici.

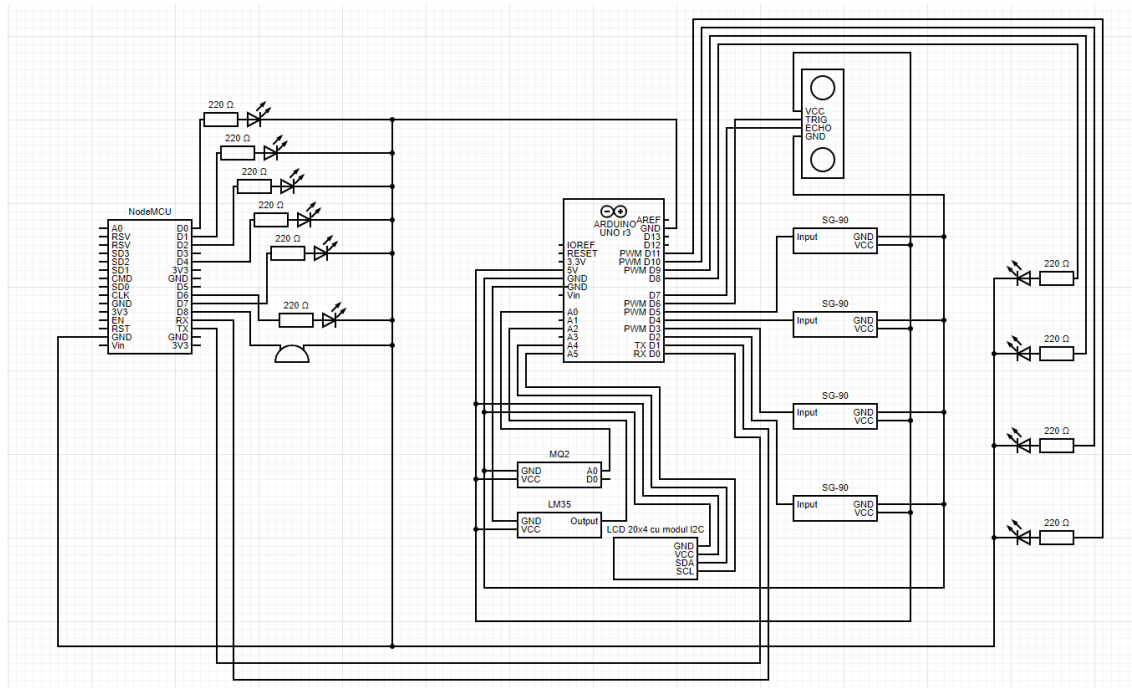


Figura 5.27. Schema circuitului componentelor fizice

5.4.3. Platforma web

Platforma web este compusă din diferite pagini, fiecare cu rolul ei specific:

- login.php are rolul de a verifica dacă există o sesiune curentă. Dacă există, această pagină va redirecționa utilizatorul către interfața grafică, altfel are rolul de autentificare, extrăgând credențialele din baza de date și comparându-le cu cele introduse de utilizator după apăsarea butonului de login.
- index.php este interfața grafică principală prin care utilizatorul poate folosi funcționalitățile sistemului prin intermediul unor butoane și textbox-uri. De asemenea, această pagină conține butonul de logout, care redirecționează utilizatorul către pagina de logout.
- logout.php are doar rolul de a distruge sesiunea curentă și de a redirecționa pagina către login.php.
- database_connect.php are rolul de a stabili o conexiune cu baza de date.
- update_values.php are rolul de a actualiza valorile primite în baza de date.
- TX.php are rolul de a trimite și primi informații de la microcontrolerul ESP8266.

5.4.4. Baza de date

Baza de date este responsabilă cu menținerea consistenței datelor.

Aceasta conține:

- id este coloana care conține cheia primară.
- PASSWORD
- SENT_NUMBER_1...SENT_NUMBER_3 sunt coloane responsabile cu transmiterea de date către platforma web, precum și valori primite de la senzori.
- RECEIVED_BOOL1 ... RECEIVED_BOOL6. De la 1 până la 5, aceste coloane sunt responsabile cu reținerea valorii care indică starea luminilor. Coloana 6 indică starea de activat/dezactivat pentru alarma de proximitate.
- RECEIVED_BOOL_USA coloana responsabilă cu starea ușii.
- RECEIVED_BOOL_GARAJ coloana responsabilă cu starea garajului.
- RECEIVED_BOOL_GEAM1/2 sunt coloane responsabile cu starea celor două geamuri.
- RECEIVED_BOOL_SMOKE coloana responsabilă cu starea de activat/dezactivat pentru alarma de fum și gaze inflamabile.
- RECEIVED_NUM1... RECEIVED_NUM3 coloane responsabile cu trimiterea valorilor numerice din textbox-uri, în produsul final a fost folosită doar RECEIVED_NUM2.
- TEXT_1 este coloana responsabilă cu reținerea ultimului mesaj trimis înspre LCD.
- NAME este coloana care conține numele de utilizator pentru autentificare.
- PASS este coloana care conține parola pentru autentificare.

Capitolul 6. Testare și validare

În acest capitol vom prezenta procesul de testare al funcționalităților.

6.1. Testarea funcționalităților propuse

6.1.1. Autentificare

6.1.1.1. Credențialele greșite

Pentru început se va încerca logarea cu username-ul și parola „admin”

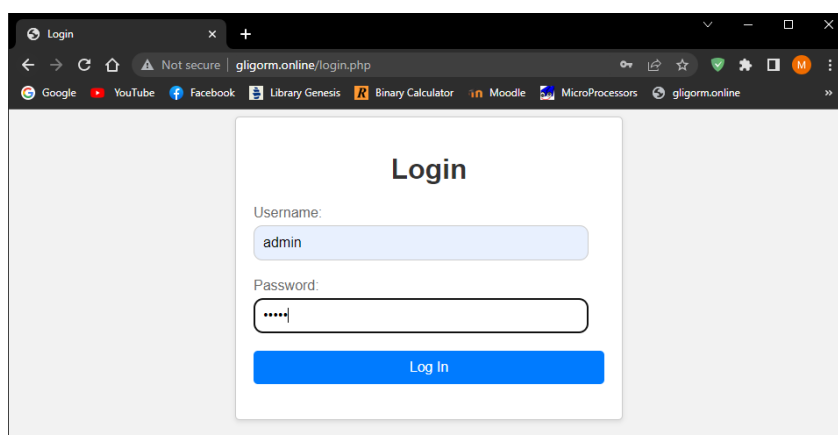


Figura 6.1. Login cu credentiale greșite

După apăsarea butonului de "Log in", din cauza faptului că exista credențiale incorecte, utilizatorul nu va fi redirecționat către interfața grafică, ci pagina de login va fi reîncărcată cu un mesaj de eroare.

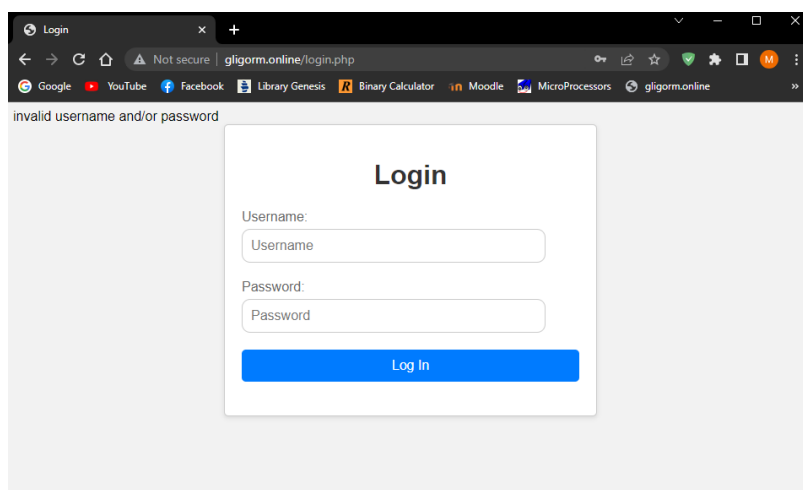


Figura 6.2. Pagina de login după încercarea de autentificare cu credențiale greșite

6.1.1.2. Autentificare de pe diferite browsere

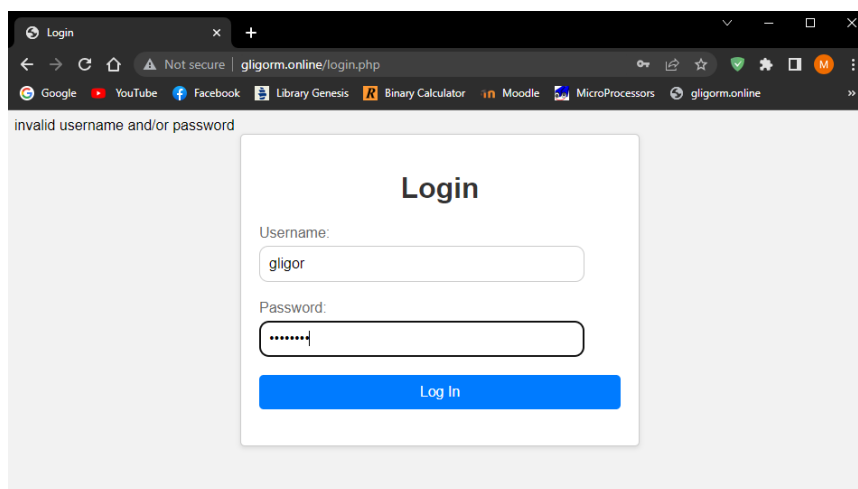


Figura 6.3. Pagina de login in browser-ul Google Chrome

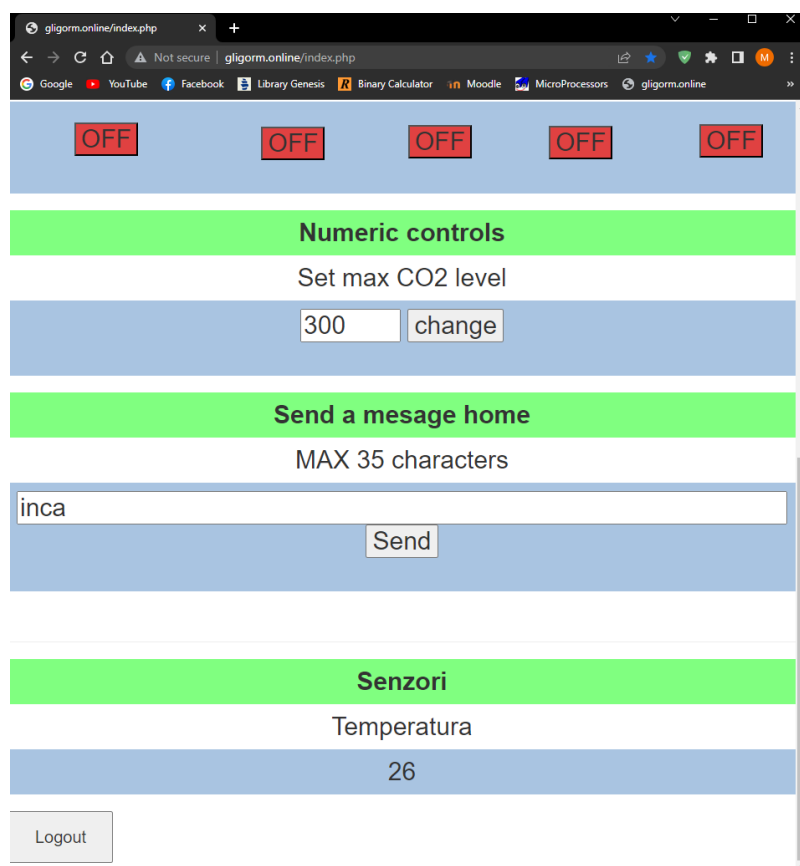


Figura 6.4. Interfata grafica in browser-ul Google Chrome

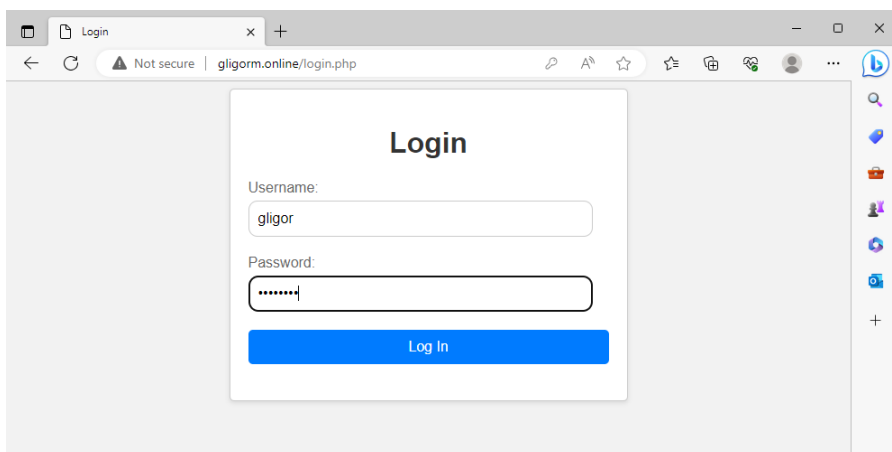


Figura 6.5. Pagina de login in browser-ul Microsoft Edge

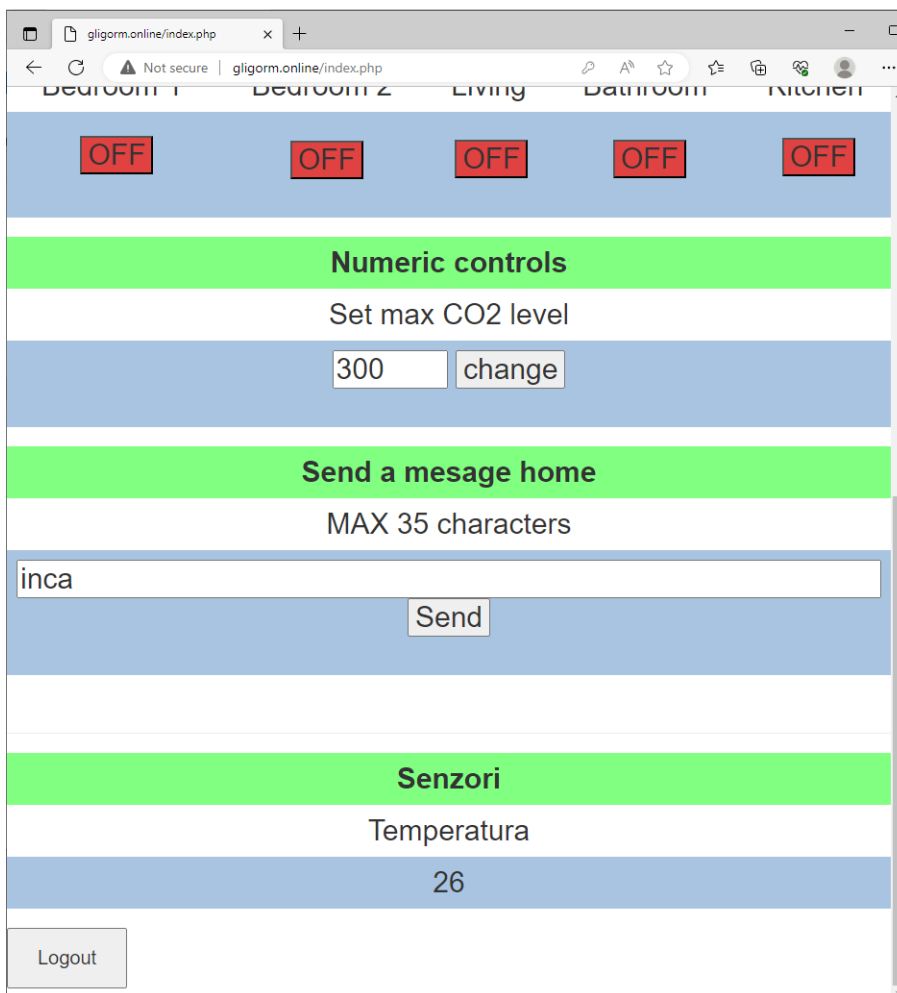


Figura 6.6. Interfata grafica in browser-ul Microsoft Edge

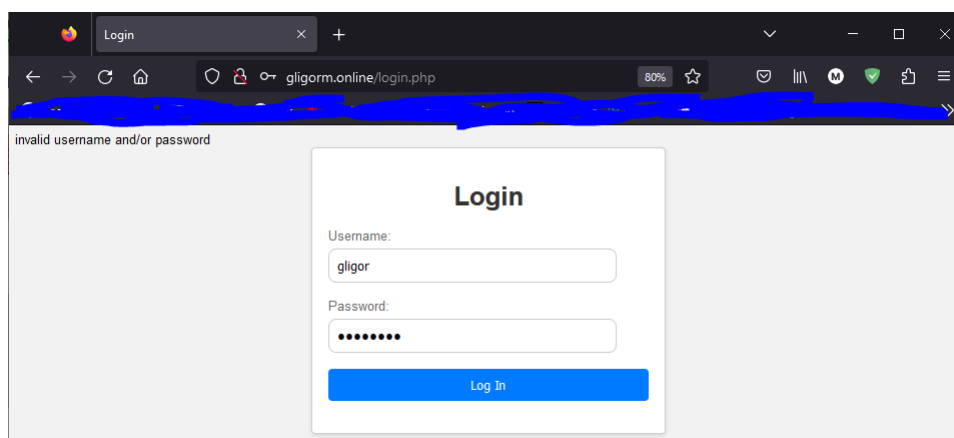


Figura 6.7. Pagina de login in browser-ul Mozilla Firefox



Figura 6.8. Interfata grafica in browser-ul Mozilla Firefox

6.1.1.3. Autentificare de pe diferite dispozitive

Din cauza faptului că interfața grafică și pagina de login se află în domeniul public, acestea pot fi accesate de pe orice dispozitiv care are acces la un browser. Exemplele prezentate în figurile de mai sus reprezintă accesul la site prin intermediul unui laptop. Exemplele din figura următoare reprezintă accesul la platforma web prin intermediul unui smartphone.

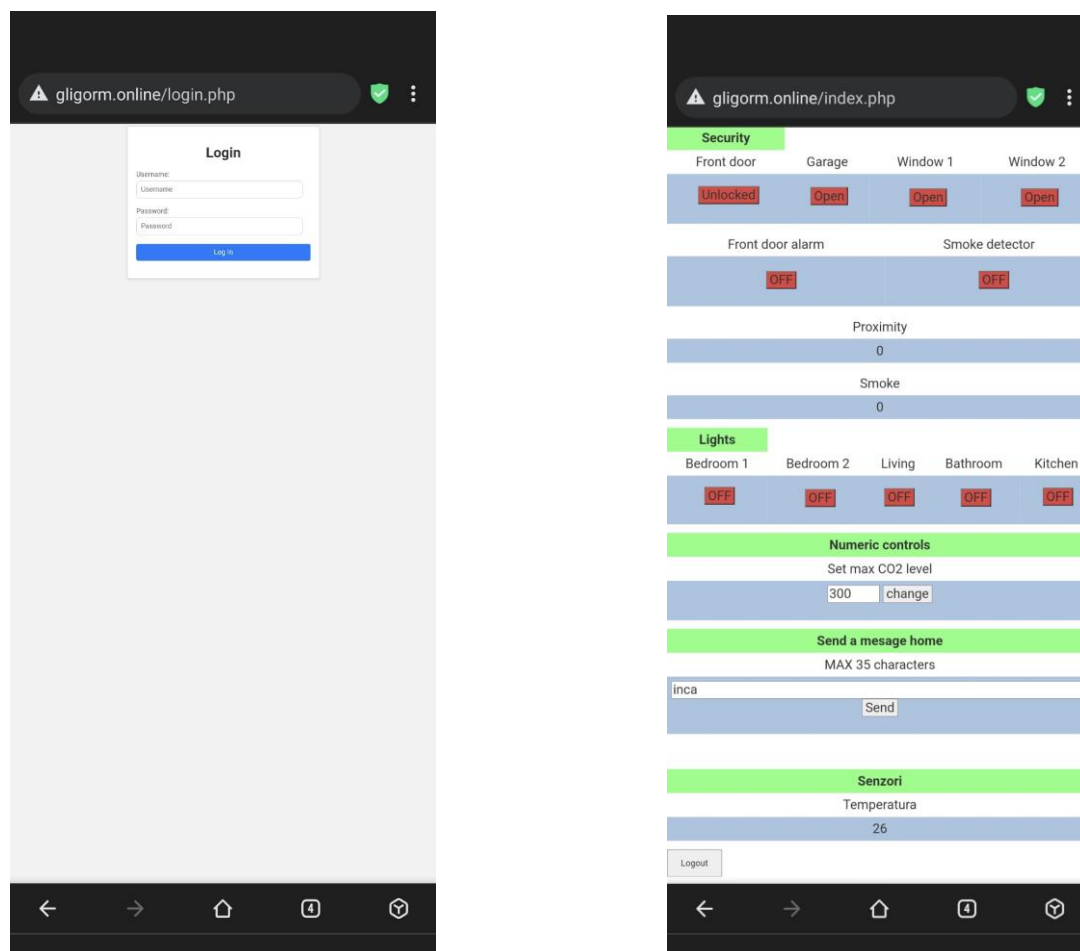


Figura 6.9. Accesul la platforma web prin intermediul unui dispozitiv mobil

6.1.2. Testarea motoarelor servo

S-a realizat o testare inițială a micro servomotoarelor folosind un script separat care conținea doar librăria servo și codul necesar acționării acestora. Acest cod a fost rulat pentru fiecare motor în parte, iar funcționarea corectă a acestora a fost verificată. Apoi, s-a trecut la următoarea etapă de testare, constând în verificarea valorilor primite din interfața grafică. Starea butoanelor de pe interfața grafică a fost verificată și s-a comparat cu valorile primite pe portul serial. Conexiunea între placa ESP8266 și Arduino Uno a fost stabilită, iar valorile primite prin portul serial au fost verificate pentru a se asigura corespondența cu starea butoanelor. Ulterior, s-au integrat părțile de cod necesare din șablonul de testare a servomotoarelor în proiectul final.

Următorul pas a constat în testarea servomotoarelor prin intermediul interfeței grafice. Aceasta a fost realizată prin activarea fiecărui motor în parte pentru a se verifica dacă primește

corect comenzile trimise prin butoane. Apoi, s-au testat combinații de câte două motoare și, în final, s-au acționat simultan toate cele 4 motoare.

6.1.3. Testarea alarmelor

6.1.3.1. Alarma de proximitate

Inițial s-a testat senzorul de proximitate folosind un script separat care conținea doar codul necesar preluării distanței și afișării acesteia pe monitorul serial. Apoi s-a adăugat codul necesar alarmei, adică aprinderea și stingerea LED-ului și acționarea buzzerului când distanța era mai mică decât un prag stabilit. O altă componentă importantă a testării a fost asigurarea transmiterii corecte a datelor atât dinspre interfața grafică către partea fizică, cât și între microcontrolere, deoarece, spre deosebire de motoarele servo, dacă alarma este activată, trebuie să transmită înapoi către ESP8266 datele citite de către senzor. Aceste date au fost testate prin intermediul monitorului serial. După ce s-a asigurat corectitudinea informațiilor transmise, s-au plasat diferite obiecte la diferite distanțe față de senzor și s-a măsurat cu un liniar distanța, după care s-a comparat cu cea afișată pe monitorul serial. De asemenea, s-a testat pentru a determina dacă alarma se activează la distanța stabilită ca prag.

6.1.3.2. Alarma de fum și gaze inflamabile

Inițial s-a testat senzorul de fum și gaze inflamabile folosind un script separat care conținea doar codul necesar preluării valorii numerice și afișării acesteia pe monitorul serial. Apoi s-a adăugat codul necesar alarmei, adică aprinderea și stingerea LED-ului și acționarea buzzerului când valoarea era mai mare decât un prag stabilit. O altă componentă importantă a testării a fost asigurarea transmiterii corecte a datelor atât dinspre interfața grafică către partea fizică, cât și între microcontrolere. Aceste date au fost testate prin intermediul monitorului serial. După ce s-a asigurat corectitudinea informațiilor, s-a lăsat senzorul să ruleze mai mult timp în diferite condiții, precum o cameră cu geamul deschis, apoi o cameră cu geamul deschis, și la final s-a expus senzorul la fum. Astfel, s-a stabilit un prag minim recomandat de siguranță care poate fi aplicat în program fără a declanșa alarma chiar dacă nu există niciun fel de pericol, dar care să asigure în același timp calitatea aerului din încăpere. Desigur, acest prag este ajustabil. De asemenea, s-a testat pentru a determina dacă alarma se activează la depășirea valorii impuse.

6.1.4. Testarea controlului luminilor

Testarea controlului luminilor a fost relativ simplă comparativ cu alte funcționalități, deoarece este implementată direct în ESP8266 și funcțiile folosite sunt simple. Mai întâi s-a verificat prin intermediul monitorului serial dacă valorile primite de la interfața grafică corespund cu starea butoanelor. După aceea, s-a implementat codul care aprinde LED-urile. S-a aprins fiecare LED separat, apoi au fost aprinse pe rând toate și, în final, au fost aprinse în mod aleatoriu, comparând starea LED-ului cu starea butonului corespunzător.

6.1.5. Testarea trimiterii mesajelor

Inițial s-a testat ecranul LCD folosind un script separat care conținea doar codul necesar unei simple afișări. S-au testat trei librării diferite până când a fost găsită librăria potrivită pentru ecranul prezent în proiect. S-a integrat codul necesar în proiectul final și a fost testat din nou cu un simplu print. Următorul pas a fost afișarea textului primit de la interfața grafică. La început au fost eliminate caracterele se spațiu redundante. A fost procesat textul astfel încât să fie trimise înspre Arduino Uno strict caracterele introduse de către utilizator. După aceea, a fost testată lungimea maximă la care poate ajunge șirul de caractere fără a apărea riscul de eroare.

Inițial, s-au introdus valori, atât numerice cât și caractere, care aveau lungimea de 5. Apoi, a fost crescută cu câte un caracter lungimea șirului. La 7 caractere lungime s-a observat apariția unor erori. În urma considerării mai multor posibile soluții, s-a decis reducerea dimensiunii documentului JSON trimis înspre ESP8266 și stabilirea unei dimensiuni de 512 biți pentru fișierul care primește informațiile, astfel a crescut capacitatea afișării de la 7 la 35 de caractere.

6.1.6. Testarea monitorizării temperaturii

Asemănător cu restul componentelor, testarea inițială a avut loc pe un script separat. Inițial s-a folosit doar funcția "analogRead()" pentru a citi valoarea primită de către senzor, după care s-au căutat din mai multe surse prelucrarea corectă a valorilor primite. La final, a fost folosită formula " $(\text{analogRead(temperatura)} * (5/1023.0)) * 100$ ", unde 5 reprezintă sursa de alimentare și 1023 numărul de valori reproduse de către convertorul analog-digital. La final, această valoare, care reprezintă milivolții de output, este înmulțită cu 100 pentru a fi transformată în grade Celsius. După obținerea valorii, aceasta a fost comparată cu valoarea afișată de termometrul din cameră, dar rezultatele au fost inconsistente. O mică diferență între cele două valori era așteptată, însă diferența se apropia de 10 grade Celsius, așa că s-a recitit documentația senzorului și, după o analiză atentă, s-a hotărât să măsurarea tensiunii de alimentare cu ajutorul unui multimetru, iar rezultatul obținut a fost de 4.3 volți. După înlocuirea acestei valori, s-au comparat din nou rezultatele citirii cu termometrul din cameră și valorile au fost aproape identice.

6.2. Componente și funcționalități testate care nu au fost introduse în proiectul final

În procesul de creare al acestui proiect, au fost abordate mai multe idei care nu au fost implementate în rezultatul final din cauza unor circumstanțe variate. În continuare, se vor prezenta câteva dintre acestea.

Microcontrolerul ESP8266 folosit inițial a fost Amica NodeMCU, însă cu acest modul s-au întâmpinat mai multe probleme. S-au încercat mai multe versiuni de cod și când s-a reușit stabilirea unei conexiuni WiFi, s-a observat că, spre deosebire de exemplele online vizionate, aceasta era foarte lentă. Mai târziu, au început să apară erori și după conectare. S-a observat ulterior și faptul că nu recunoștea aproape nicio rețea din apropiere. După câteva săptămâni de depanare, s-a ajuns la concluzia că nu era vorba de o eroare de programare, ci de modulul hardware. După înlocuirea acestuia cu Lolin NodeMCU ESP8266, conexiunile la orice rețea au fost aproape instantane.

Modulul de monitorizare a temperaturii trebuia inițial să conțină și un ventilator care să pornească dacă temperatura depășea un anumit prag. Acest prag era stabilit în interfața grafică, similar cu pragul prezent la monitorizarea calității aerului. În încercarea de a implementa această funcționalitate, inițial s-a introdus textbox-ul necesar, iar pentru ventilator s-a folosit un motor electric DC cu o elice atașată la capăt. Testarea inițială a decurs relativ bine, însă odată introdus în proiectul final, s-a observat că motorul avea nevoie de prea mult curent la pornire, ceea ce crea zgomot electric și perturbă funcționalitatea altor senzori. De exemplu, senzorul de proximitate furniza valori la întâmplare când motorul era pornit și activa alarma fără a fi nevoie.

Servomotorul care acționează ușa de la garaj era inițial controlat de către un modul care recepționa semnale infraroșii de la o telecomandă. La recepționarea unui astfel de semnal, se verifica starea motorului, iar apoi codul se asigura că acesta ajungea în starea opusă. Problema cu această abordare era faptul că orice semnal infraroșu putea acționa sistemul, inclusiv camera oricărui smartphone modern. Din acest motiv, aceste semnale au fost decodificate într-un script

și, dacă codul semnalului corespundea cu codul butonului stabilit pentru controlul servomotorului, era luată o acțiune. Motivul pentru care această funcționalitate a fost eliminată este interferența pe care modulul de recepție a semnalelor infraroșii o avea asupra comunicării seriale. Acest modul se bază, de asemenea, pe protocolul I2C care era folosit pentru LCD. Aceste două dezavantaje au dus la eliminarea acestei componente din proiectul final.

Pentru afișarea mesajelor inițial era planificată folosirea unui LCD 16x2, însă au apărut diverse probleme. În primul rând, a fost dificilă alegerea unei librării care să funcționeze cu ecranul respectiv. După ce a fost găsită librăria potrivită, s-a observat că, deși era posibilă aprinderea luminii de fundal, acesta nu afișa niciun caracter. După schimbarea codului, s-a considerat că există o problemă de hardware. Astfel, s-a folosit un multimetru pentru a verifica dacă, atunci când a fost atașat modulul I2C cu ciocanul de lipit și cositorul, s-au scurtcircuitat pini, însă nu s-au obținut rezultate care să confirme această ipoteză. Ca urmare, ecranul a fost înlocuit cu unul similar, de aceeași dimensiune, care s-a dovedit a prezenta aceeași problemă. S-a decis să se renunțe pentru moment la acea funcționalitate, iar adăugarea unui LCD de dimensiuni 20x4 a fost făcută doar spre sfârșitul proiectului.

Comunicația serială între cele două microcontrolere a trecut, de asemenea, prin mai multe iterații. La începutul proiectului s-a folosit protocolul I2C. Prin intermediul acestuia s-au trimis doar câteva valori booleene, însă s-a observat că atât ecranul LCD, cât și senzorul cu infraroșu se bazau pe același protocol, așa că a fost înlocuit cu protocolul SPI. Din păcate, cu acesta nu s-a reușit stabilirea unei conexiuni stabile, și după mai multe încercări de schimbare a pinilor utilizați, s-a trecut la soluția curentă, și anume protocolul UART.

6.3. Motivația alegerii configurației curente

Configurația curentă a fost influențată de mai mulți factori.

Disponibilitatea componentelor a jucat un rol major în proiectarea acestui sistem deoarece multe dintre acestea erau la îndemână, fiind folosite la proiecte anterioare, cel mai relevant exemplu fiind senzorul cu ultrasunete folosit și la proiectul de la materia Proiectare cu Microprocesoare.

Ușurința integrării acestora în proiectul final a reprezentat un alt factor important. Un exemplu relevant este prezentat în capitolul anterior sub forma senzorului cu infraroșu care, pe lângă faptul că nu era foarte stabil, influența în mod negativ comunicarea dintre microcontrolere și împiedica implementarea ecranului pentru afișarea mesajelor. În contrast, restul componentelor folosite au fost implementate relativ armonios, fără a reduce performanța sau fiabilitatea uneia altele.

De asemenea, componentele au fost alese astfel încât să fie relativ ieftine, dar capabile să îndeplinească funcționalitățile de bază ale unui astfel de sistem. În concluzie, formatul curent a fost influențat atât de ușurința de implementare, cât și de practicitatea perifericelor alese.

Capitolul 7. Manual de instalare si utilizare

Arduino IDE

Următorii pași se execută doar în cazul în care programele necesare nu au fost încărcate în plăcile de dezvoltare sau au fost modificate pentru a adăuga funcționalități.

Pentru descărcarea acestui software se accesează linkul :

<https://www.arduino.cc/en/software>

Aici se selectează opțiunea care corespunde sistemului de operare folosit

Downloads

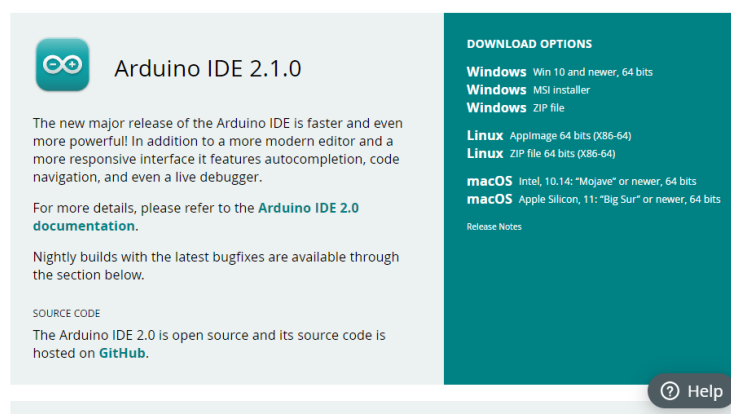


Figura 7.1. Descarcare Arduino IDE

După care utilizatorul va fi redirecționat care pagina de descărcare unde trebuie selectata opțiunea „just download”

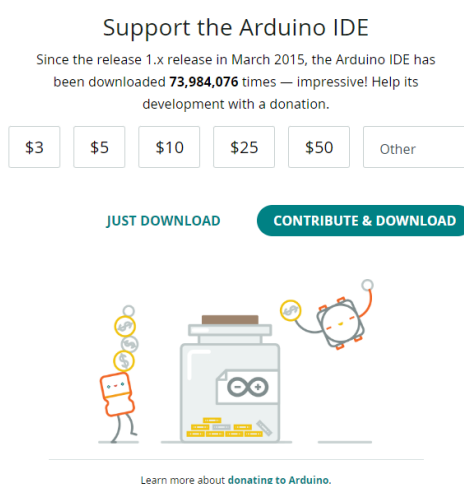


Figura 7.2. Descarcare Arduino IDE 2

Se rulează executabilul si se urmează pașii pana când programul este instalat cu succes. După ce programul a fost instalat, se navighează înspre folderul acestuia si se accesează folderul „libraries”.

Name	Date modified	Type	Size
drivers	30.07.2022 12:23	File folder	
examples	30.07.2022 12:23	File folder	
hardware	30.07.2022 12:23	File folder	
java	30.07.2022 12:23	File folder	
lib	30.07.2022 12:23	File folder	
libraries	06.08.2022 14:57	File folder	
tools	30.07.2022 12:23	File folder	
tools-builder	30.07.2022 12:23	File folder	
arduino.exe	20.12.2021 18:13	Application	72 KB
arduino.l4j.ini	20.12.2021 18:13	Configuration sett...	1 KB
arduino_debug.exe	20.12.2021 18:13	Application	69 KB
arduino_debug.l4j.ini	20.12.2021 18:13	Configuration sett...	1 KB
arduino-builder.exe	20.12.2021 18:12	Application	23.156 KB
libusb0.dll	20.12.2021 18:12	Application exten...	43 KB
msvc100.dll	20.12.2021 18:12	Application exten...	412 KB
msvc100.dll	20.12.2021 18:12	Application exten...	753 KB
revisions.txt	20.12.2021 18:12	Text Document	97 KB
uninstall.exe	30.07.2022 12:23	Application	404 KB
wrapper-manifest.xml	20.12.2021 18:13	XML Document	1 KB

Figura 7.3. Selectarea folderului libraries

În interiorul acestui folder se copiază librăriile oferite de către administratorul produsului.

Adafruit_Circuit_Playground	30.07.2022 12:23	File folder	
Arduino-LiquidCrystal-I2C-library	09.03.2017 08:56	File folder	
Bridge	30.07.2022 12:23	File folder	
Esplora	30.07.2022 12:23	File folder	
Ethernet	30.07.2022 12:23	File folder	
Firmata	30.07.2022 12:23	File folder	
GSM	30.07.2022 12:23	File folder	
IRremote-3.7.1	28.06.2022 21:01	File folder	
Keyboard	30.07.2022 12:23	File folder	
Mouse	30.07.2022 12:23	File folder	
Robot_Control	30.07.2022 12:23	File folder	
Robot_Motor	30.07.2022 12:23	File folder	
RobotIRremote	30.07.2022 12:23	File folder	
SD	30.07.2022 12:23	File folder	
Servo	30.07.2022 12:23	File folder	
SpacebrewYun	30.07.2022 12:23	File folder	
Stepper	30.07.2022 12:23	File folder	
Temboo	30.07.2022 12:23	File folder	
TFT	30.07.2022 12:23	File folder	
WiFi	30.07.2022 12:23	File folder	
IRremote-3.7.1.zip	02.07.2022 12:42	WinRAR ZIP archive	821 KB
LiquidCrystal_I2C-1.1.2.zip	06.08.2022 14:56	WinRAR ZIP archive	9 KB

Figura 7.4. Folderul libraries

În cazul librăriei specifice plăcii de dezvoltare ESP8266, trebuie urmat următorul tutorial : <https://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/>

Odată instalat, se deschid cu acest program fișierele primite.

Mai întâi se va încărca fișierul „Main” in placa Lolin NodeMCU ESP8266.

Pentru aceasta se accesează meniul Tools → Board: → ESP8266 Boards ... → NodeMCU 1.0 (ESP-12E Module) pentru fi selectata placa corecta in timpul compilării programului.

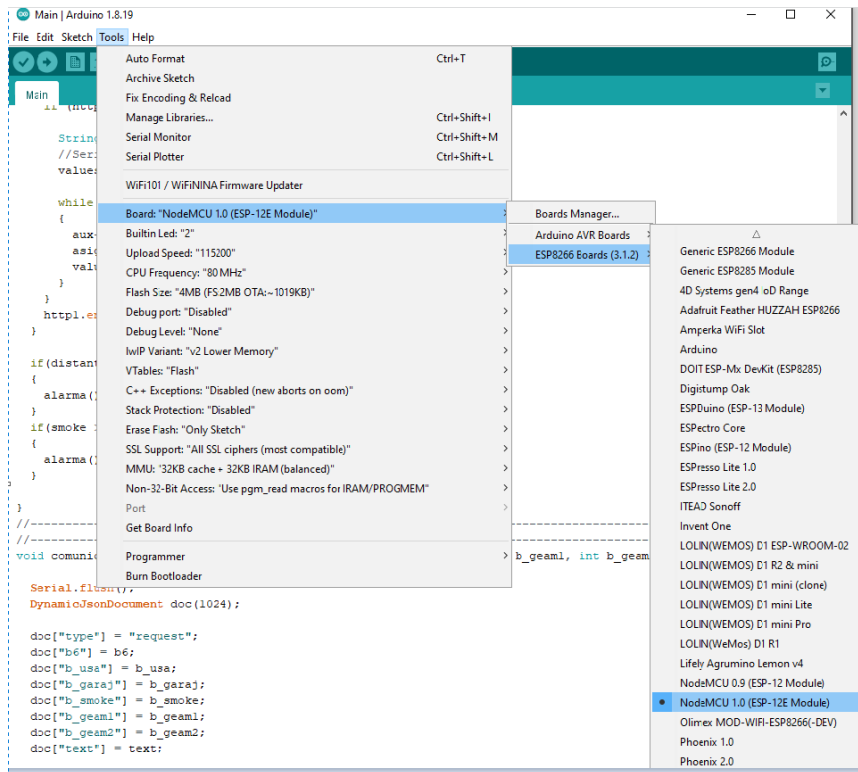


Figura 7.5. Selectarea plăcii NodeMCU

Se introduce într-un port al dispozitivului cablul USB al plăcuței Lolin NodeMCU ESP8266, după care se deconectează firele de la pinii TX și RX. După deconectare, se apasă butonul de upload localizat în bara de sus, așa cum este indicat în imagine.

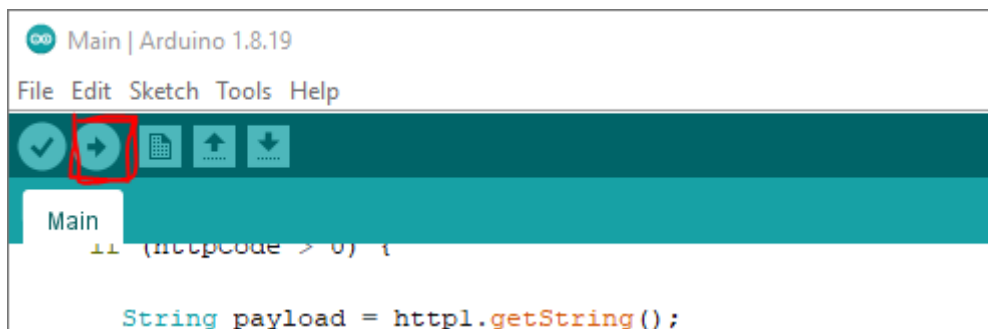


Figura 7.6. Incarcarea programului pe placa de dezvoltare

După încărcarea programului se reconectează firele TX și RX și se deconectează placa de dezvoltare.

Următorul pas este încărcarea programului in placa de dezvoltare Arduino Uno
Mai întâi se deschide fișierul ArduinoUno.ino in Arduino IDE si se selectează in meniu Tools → Board: → Arduino AVR Boards → Arduino Uno

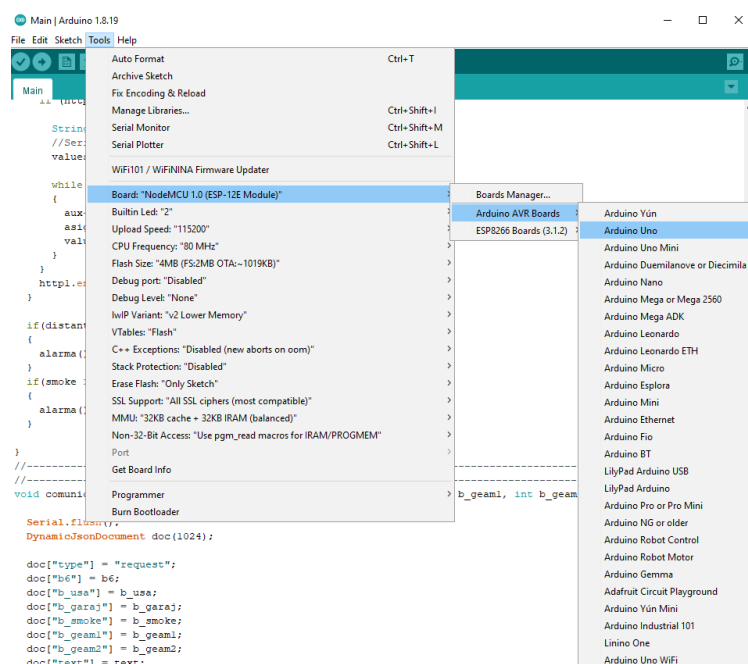


Figura 7.7. Selectarea placii Arduino Uno

Se introduce într-un port al dispozitivului cablul USB al plăcuței Arduino Uno, după care se apasă butonul de upload localizat în bara de sus, așa cum este indicat în figura 7.5. La final, se conectează cablurile USB ale ambelor microcontrolere la o sursă de curent de 5 volți.

Pentru utilizarea funcționalităților acestui produs trebuie urmați următorii pași:

Dacă nu există o autentificare anterioară pe dispozitivul curent, utilizatorul va fi direcționat înspre pagina de autentificare (Figura 7.7.)

<http://glogorm.online/login.php>

Figura 7.8. Pagina de autentificare

După introducerea credențialelor oferite de către administrator, se apasă butonul de login. Dacă credențialele sunt corecte, se va încărca interfața principală la adresa specificată:

<http://glogorm.online/index.php>

Aici utilizatorul poate accesa funcționalitățile sistemului

Security				
Front door	Garage	Window 1	Window 2	
<input type="button" value="Unlocked"/>	<input type="button" value="Open"/>	<input type="button" value="Open"/>	<input type="button" value="Open"/>	
Front door alarm		Smoke detector		
<input type="button" value="OFF"/>		<input type="button" value="OFF"/>		
Proximity				
0				
Smoke				
0				
Lights				
Bedroom 1	Bedroom 2	Living	Bathroom	Kitchen
<input type="button" value="OFF"/>	<input type="button" value="OFF"/>	<input type="button" value="OFF"/>	<input type="button" value="OFF"/>	<input type="button" value="OFF"/>
Numeric controls				
Set max CO2 level				
300 <input type="button" value="change"/>				
Send a message home				
MAX 35 characters				
inca				
<input type="button" value="Send"/>				
Senzori				
Temperatura				
26				
<input type="button" value="Logout"/>				

Figura 7.9. Pagina principala

Pentru acționarea butoanelor este suficient un simplu click, iar aparența butonului se va schimba în funcție de starea curentă.

Pentru setarea nivelului maxim de gaz detectat trebuie introdusă în textbox valoarea dorită, apoi trebuie apăsat butonul "change". Valoarea recomandată este de 300.

Pentru trimiterea unui mesaj pe afișajul LCD se scrie propoziția dorită și se apasă "change". Propoziția nu trebuie să depășească 35 de caractere, inclusiv spațiile libere. În cazul în care această limită este depășită, trebuie rescrisă propoziția astfel încât să îndeplinească constrângerea, după care va fi apăsat din nou butonul "change". După urmarea acestor pași, trebuie apăsat butonul de reset aflat pe placa Arduino Uno.



Figura 7.10. Butonul de reset al placii Arduino Uno

Capitolul 8. Concluzii

8.1. Rezultate obținute

Trăim într-o epocă în care tehnologia este o parte necesară a vieții noastre, iar scopul ei este să ne facă viața mai ușoară. De aceea, multe case au un sistem inteligent responsabil pentru gestionarea lor. Dar, de obicei, acest tip de tehnologie poate fi foarte scump și greu de reparat de unul singur

Obiectivul acestui proiect a fost crearea unui sistem ușor de utilizat, ieftin, dar fiabil, accesibil nu dintr-o aplicație sau un dispozitiv, ci de pe orice dispozitiv care are un browser și o conexiune la internet.

Proiectul smart house controlat de la distanță își propune să ofere clientului posibilitatea de a beneficia de toate funcționalitățile majore ale sistemului smart house la un preț mult mai mic în comparație cu concurența, având același nivel de fiabilitate ca și sistemele complexe.

Funcționalitățile implementate în proiectul final sunt :

- Controlul încuietorii
- Deschiderea și închiderea ferestrelor
- Deschiderea și închiderea garajului
- Alarmă de proximitate cu monitorizarea distanței
- Alarma de fum și gaze inflamabile cu prag de declanșare ajustabil
- Controlul luminilor
- Trimiterea și afișarea mesajelor pe un LCD
- Monitorizarea temperaturii

8.2. Dezvoltări ulterioare

Fiind un sistem relativ modular, proiectul smart house controlat de la distanță are o multitudine de viitoare direcții în care ar putea să se dezvolte. În continuare, se vor prezenta câteva exemple de îmbunătățiri ale sistemului curent.

Sistemul poate fi îmbunătățit în primul rând prin adăugarea unui modul care să permită suprascriserea manuală. Pe scurt, adăugarea unui ecran mic, a unei tastaturi și a unor butoane care să permită utilizarea offline a sistemului. Această utilizare offline poate fi implementată și prin înlocuirea microcontrolerului ESP8266 cu ESP32, care este echipat cu un modul Bluetooth. Această implementare necesită și dezvoltarea unei aplicații dedicate sistemului.

Sistemul de monitorizare a temperaturii poate fi îmbunătățit prin adăugarea unui ventilator. Pentru a integra această funcționalitate, interfața grafică va trebui să fie actualizată cu un textbox în care utilizatorul să poată introduce temperatura dorită. Dacă această temperatură este depășită, ventilatorul se va activa.

Optimizarea sistemului poate fi realizată prin implementarea unor surse externe de curent pentru diferitele periferice atașate microcontrolerelor. Acest aspect ar limita puterea consumată de plăcile de dezvoltare, ar duce la un timp de răspuns mai bun și la citirea mai precisă a valorilor provenite de la senzori.

O altă opțiune pentru îmbunătățirea sistemului poate fi înlocuirea plăcii Arduino Uno cu Arduino Mega, astfel, pe lângă modulele adiționale care pot fi adăugate datorită numărului crescut de pini disponibili, sistemul de trimitere a mesajelor va fi optimizat, deoarece Arduino Mega dispune de mai multă memorie disponibilă pentru primirea și manipularea datelor.

Bibliografie

- [1] A. W. D. A. Adriansyah, „Design of Small Smart Home system based on Arduino,” 2014 Electrical Power, Electronics, Communicatons, Control and Informatics Seminar (EECCIS), 2014.
- [2] D.-Y. L. B. Y. JIANG, „Smart Home Research,” *Sci-hub of open science*, pp. 659-664, 2004.
- [3] M. K. L. C. Y. J. Z. A. K. D. S. Davidoff, „Principles of Smart Home Control,” *Ubiquitous Computing*, pp. 19-34.
- [4] T. K. R. M. R. S. N. Z. N. S. M. Z. M. B. V. S. S. A. W. A. Jabbar, „Design and Fabrication of Smart Home with Internet of Things Enabled Automation System,” *IEEE Open Access Journal*, vol. Vol. XX.
- [5] D.-Y. K. C.-Y. Y. B.-C. L. D. M. Konidala, „Security framework for RFID-based applications in smart home environment,” *Journal of Information Processing Systems*, vol. vol. 7, nr. 1, pp. 111-120, 2011.
- [6] N. S. M. a. P. B. D. Anandhavalli, „Smart Home Automation Control Using Bluetooth And GSM,” *International Journal of Informative and Futuristic Research*, vol. 2, nr. 8, 2015.
- [7] B. D. A. Labus, „A smart home system based on sensor technology,” *Facta Universitatis, Series: Electronics and Energetics*, vol. 29, nr. 3, pp. 451-460, 2015.
- [8] A. C. A. U. E. O. N. David, „Design of a home automation system using arduino,” *International Journal of Scientific & Engineering Research*, vol. 6, nr. 6, pp. 795-801, 2015.
- [9] S. S. R. K. Kodali, „MQTT based home automation system using ESP8266,” *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, pp. 1-5, 2016.
- [10] J. V. V. K. S. D. T. A. S. S. Imran, „Smart Home automation based on IoT using arduino mega,” *nterInational Conference on Current Research in Engineering Science and Technology (ICCREST-2016)*, pp. 2348-8379, 2016.
- [11] J. C. P. Gupta, „IoT based Smart Home design using power and security management,” *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, pp. 6-10, 2016.
- [12] R. S. N. K. P. J. S. Kaur, „Home Automation and Security System,” *Advanced Computational Intelligence: An International Journal (ACII)*, vol. 3, nr. 3, 2016.
- [13] E. Ganesh, „Implementation of IOT Architecture for SMART HOME using GSM Technology,” *International Journal of Computer Techniques*, pp. 2394-2231, 2017.
- [14] M. H. A. N. S. S. A. S. K. M. W. A. Jabbar, „Design and Implementation of IoT-Based Automation System for Smart Home,” *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1-6, 2018.

- [15] V. S. N. S. Badabaji, „An IoT Based Smart Home Service System,” *International Journal of Pure and Applied Mathematics*, vol. 119, nr. 16, pp. 4659-4667, 2018.
- [16] S. B. P. G. O. Bhat, „Implementation of IoT in Smart Homes”.
- [17] L. L. F.-G. O. G. S. J.-M. V. U. Ozeer, „Designing and Implementing Resilient IoT Applications in the Fog: A Smart Home Use Case,” *22nd Conference on Innovation in Clouds, Internet and Networks*, 2019.
- [18] M. O. Q. A. A.-S. S. M. M. M. L. H. M. M. Ahmed, Cost-Effective Design of IoT-Based Smart Household Distribution System, <https://www.mdpi.com/2411-9660/5/3/55>, 2021.