



Universitatea Politehnică Timișoara
Facultatea de Automatică și Calculatoare
Departamentul Calculatoare și
Tehnologia Informației



TERMOSTAT INTELIGENT, CONTROLAT PRINTR-O APLICAȚIE WEB

Proiect de Diplomă

Autor:
Vitomir Dragan

Conducător științific
Prof. Dr. Habil. Ing. Marius Marcu

Timișoara
Iunie, 2021

Cuprins

1	Introducere	3
1.1	Internet of things	3
1.2	Contextul de realizare	5
2	Prezentarea sistemelor similare	6
2.1	Torelectric	6
2.2	Honeywell	7
2.3	Nest	8
2.4	Ecobee	8
2.5	Abordare comparativă a sistemelor prezentate	9
2.6	Comparație cu sistemul prezentat în această lucrare	10
3	Fundamente teoretice	11
3.1	Python	11
3.2	Flask	11
3.3	C++	12
3.4	Arduino IDE	12
3.5	Componente utilizate	13
3.5.1	ESP8266	13
3.5.2	Placă de bază pentru NodeMCU	13
3.5.3	Arduino Uno	13
3.5.4	LCD 1602	13
3.5.5	Senzor DHT11	14
3.5.6	Modul radio frecvență	14
3.5.7	Releu	14
3.5.8	Electrovalvă	15
3.5.9	Pompă de apă	15
3.5.10	Push buton	15
3.5.11	Condensator	15
3.5.12	Rezistență	15
3.5.13	Circuit integrat Schmitt-Trigger	15

3.5.14	Diodă	15
3.6	Filtrarea semnalelor	16
3.7	Histereza	19
4	Dezvoltarea soluției	20
4.1	Specificarea cerințelor	20
4.1.1	Cerințele funcționale ale aplicației web	21
4.1.2	Cerințe nefuncționale ale aplicației web	22
4.1.3	Cerințe funcționale pentru componentele electronice	23
4.1.4	Cerințe nefuncționale pentru componentele electronice	23
4.2	Arhitectura sistemului	24
4.2.1	Modulul senzor	25
4.2.2	Modulul de control	28
4.3	Proiectare detaliată	28
4.3.1	Diagramă de clase	29
4.3.2	Diagrame de secvență	30
4.3.3	Structură software	34
4.4	Implementarea soluției	34
4.4.1	Programare modul ESP8266	35
4.4.2	Programare plăcuță Arduino	42
4.4.3	Realizarea aplicației web	44
4.4.4	Probleme întâmpinate în timpul realizării proiectului	50
4.5	Testarea soluției	52
5	Ghidul utilizatorului	55
5.1	Interfața cu utilizatorul	55
6	Concluzii și direcții de dezvoltare	56

Capitolul 1

Introducere

1.1 Internet of things

Nivelul de evoluție la care a ajuns tehnologia în zilele noastre, dă naștere dorinței de a spori confortul vieții cotidiene prin creșterea gradului de interconectare al dispozitivelor inteligente.

Scopul primordial al conceptului de Internet Of Things este de a face posibilă comunicarea între obiectele care prezintă utilitate în viața de zi cu zi. Acesta reprezintă o rețea vastă de dispozitive interconectate care sunt capabile să ia decizii fără intervenție din exterior. Prin intermediul senzorilor sunt preluate diverse date din mediul înconjurător, date care, în urma prelucrărilor, determină execuția anumitor acțiuni.

Contrar așteptărilor, ideea de Internet of Things nu a apărut recent. Primul dispozitiv care implementa conceptul de IoT a fost lansat în anul 1982, fiind reprezentat de un tonomat de Coca-Cola care putea să trimită informații pe internet referitoare la numărul de doze disponibile și temperatura băuturilor [1]. În anul 1990, invenția lui John Romkey, toaster-ul controlat prin internet [2], stârnește tot mai mult interesul asupra ideii de control de la distanță al dispozitivelor. În decursul anilor, tot mai multe firme producătoare de electronice au încercat să ofere clienților produse care pot transfera date prin intermediul internetului. De exemplu, în anii 2000, firma LG a produs un frigider care se putea conecta la internet prin WiFi [2].

Chiar dacă dispozitivele care au marcat debutul IoT nu oferă funcționalități mult prea utile, în ultimii ani s-a dovedit că transferul de date între dispozitive, fără a utiliza conductori electrici, poate fi folositor chiar și în domenii cheie. Astfel, medicina, armata, transporturile și imobiliarele sunt doar câteva exemple de domenii care au fost influențate de acest concept.

Medicina - se urmărește implementarea conceptului de monitorizare de la distanță, prin intermediul unor aparate care au rolul de a înregistra date legate de starea de sănătate a pacientului și de a le trimite într-o bază de date accesibilă de către personalul

medical [3].

Armata - prezintă dispozitive de spionaj și de supraveghere de ultimă generație, toate acestea fiind posibile datorită ideii de Internet of Things [4].

Industria transporturilor - în acest domeniu, IoT poate fi ilustrat prin transferul de date între un autovehicul și infrastructură, pietoni sau un alt autovehicul. Acest concept poartă denumirea de V2X, iar beneficiul major pe care îl poate aduce este reducerea considerabilă a numărului de accidente [5].

Imobiliare - în zilele noastre, se întâlnește din ce în ce mai des ideea de casă inteligentă. Aceasta se rezumă la faptul că, dispozitivele inteligente din casă comunică atât între ele cât și cu electrocasnicele, reușind să creeze un mediu cât mai confortabil pentru locuitori.

Faptul că domeniul IoT prezintă o utilitate care nu poate fi neglijată, explică creșterea exponențială a numărului de dispozitive interconectate. Conform unei statistici publicate de către Arne Holst, conducătorul echipei de cercetare din cadrul companiei Statista, se aproximează ca în 2030 să se ajungă la un număr mai mare de 25.4 miliarde de dispozitive IoT [6].

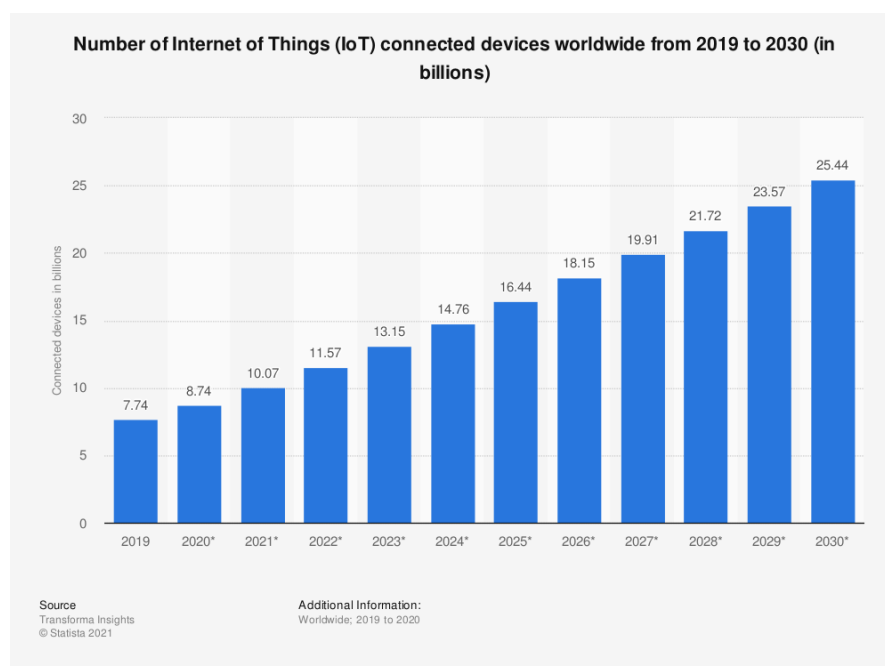


Figura 1.1: Evoluția numărului de dispozitive IoT (sursa: [6])

1.2 Contextul de realizare

Doresc ca prin intermediul proiectului să prezint beneficiile pe care automatizarea și conceptul de IoT le pot aduce. Sistemul creat abordează una dintre problemele care pot interveni la nivelul unei locuințe și se folosește de transferul de date la distanță, între diverse componente electronice, pentru a soluționa disconfortul termic.

Problema pe care încerc să o rezolv, prin intermediul proiectului, am sesizat-o la apartamentul în care locuiesc. Acesta prezintă două camere de dimensiuni diferite, iar în sezonul rece ne confruntăm cu un disconfort termic cauzat de diferențele de temperatură între cele două camere. Termostatul este montat în camera mare, fapt ce determină ca temperatura din camera mică să fie mai mare decât cea setată. Nici mutarea termostatului în camera mică nu reprezintă o soluție, din cauză că timpul necesar încălzirii camerei mari este mai mare decât cel pentru camera mică, ajungându-se ca în camera mare să fie tot timpul mai rece. Prin urmare, această inconveniență m-a motivat să creez un sistem care să reușească să mențină o temperatură constantă în apartament.

La nivel non-tehnic, soluția constă în montarea în fiecare cameră a unor module ce monitorizează temperatura, iar în funcție de aceasta, comandă atât electrovalvele montate pe returul caloriferelor, cât și centrala. Rolul electrovalvei este de a închide sau deschide circuitul de apă din calorifer. De asemenea, este necesar ca lângă centrala termică să se monteze un modul care are rolul de a porni sau de a opri centrala în funcție de comanda primită de la modulele montate în camere. Acest modul se conectează la centrală prin intermediul unor fire, iar transferul de date între modulele din camere și modulul de control al centralei se face prin radio frecvență. Pot fi setate temperaturi diferite pentru fiecare cameră în parte. Setarea se poate face fizic, prin intermediul unor butoane, sau de la distanță, prin intermediul unei aplicații web sau prin comenzi vocale interpretate de Google Assistant. Sistemul prezintă două moduri de funcționare. Primul mod constă în menținerea temperaturii setate, fără a ține cont de oră sau de faptul că ziua curentă este lucrătoare sau nu. Cel de-al doilea mod, oferă posibilitatea utilizatorului de a seta, prin intermediul aplicației web, temperaturi diferite pe anumite intervale orare ale zilei. Sistemul permite setarea a patru intervale orare în timpul zilelor lucrătoare ale săptămânii, iar pentru weekend pot fi setate două intervale.

Capitolul 2

Prezentarea sistemelor similare

În momentul de față, acest subdomeniu se află la un nivel mare de răspândire. Există o serie de producători care comercializează sisteme ce permit controlul la distanță al temperaturii, însă au o deficiență majoră, și anume, prețul ridicat. Voi prezenta mai multe astfel de sisteme, împreună cu detaliile tehnice ale acestora.

2.1 Torelectric

Termostatul produs de firma Torelectric oferă mai multe modalități de reglare a temperaturii [7]:

- Prin aplicație mobilă
- Fizic, utilizând interfața termostatlui
- Prin comenzi vocale, utilizând Google Home și Alexa



Ca și caracteristici principale se pot remarca [7]:

- Conectivitate: Wi-Fi

- Tip alimentare: la retea
- Precizie: ± 2 °C sau ± 1 °C
- Setare temperatură între: 5 - 35 °C
- Temperatură ambientală: 0 - 45 °C

2.2 Honeywell



Termostatul produs de Honeywell se diferențiază față de cel produs de Torelectric prin prezența unor funcționalități inovatoare. Printre particularitățile acestui dispozitiv se remarcă [8]:

- Adaptarea temperaturii în funcție de prezența sau absența unor persoane în locuință
- Programarea temperaturilor pentru șapte zile
- Controlul de la distanță, oferit de aplicația mobilă
- Setarea temperaturii prin comenzi vocale

Comparativ cu primul sistem prezentat, cel de la Honeywell se deosebește prin tehnologia Geofencing. Acesta știe când imobilul nu este locuit, iar ca urmare, va dezactiva încălzirea. De asemenea, este capabil să detecteze momentul în care locuitorii sunt în apropierea casei, reușind să încălzească până la temperatura setată [8]. Un alt avantaj adus de cei de la Honeywell este posibilitatea de a seta temperatura pe decursul a șapte zile, fapt ce asigură un confort sporit, prin adaptarea temperaturii în funcție de diverse intervale orare [8].

Caracteristicile tehnice ale acestuia sunt [8]:

- Destinat: centrale termice
- Suprafața de montare: masă
- Tip alimentare: la rețea
- Conectivitate: Wi-Fi

În continuare, voi prezenta două sisteme care reprezintă apogeul evoluției în acest domeniu. Acestea sunt produse de firme cunoscute precum: Google și Ecobee.

2.3 Nest

Este un termostat inteligent produs de cei de la Google. Acesta oferă funcționalități avansate pentru un management, cât mai eficient, al temperaturii din imobil.

Spre deosebire de sistemele prezentate anterior, termostatele produse de Google Nest integrează tehnologii revoluționare pentru a obține un randament cât mai mare.

Principalele particularități ale acestuia sunt [9]:

- Capacitatea de a se programa automat, în funcție de rutina locuitorilor
- Menține temperatura setată doar dacă există persoane în imobil, altfel trece automat la o temperatură ce asigură un consum cât mai mic de energie
- Control de la distanță de pe laptop, tabletă și telefon
- Salvarea unui istoric al consumului de energie
- Tehnologie Geofencing
- Se pot adăuga senzori pentru mai multe zone

Date tehnice [9]:

- Sursă de energie: baterii
- Tensiune de funcționare: 24 volți
- Afișaj digital
- Conectivitate: WiFi



2.4 Ecobee

Termostatul produs de firma Ecobee concurează cu Google Nest. Acesta oferă, pe lângă funcțiile complexe, posibilitatea de monitorizare detaliată a consumului de energie [10].

Particularități [10]:

- Poate fi controlat prin intermediul unor aplicații precum: Apple HomeKit, Alexa built-in, Google Assistant și SmartThings. De asemenea, oferă posibilitatea de a seta temperatura de pe Android, dar și de pe iOS.

- Implementează un algoritm ce permite controlul temperaturii în diverse locuri din imobil. Se pot conecta mai mulți senzori de temperatura la termostat, iar în funcție de informațiile primite de la aceștia, menține o temperatură constantă în locuință.
- Ecranul se aprinde atunci când detectează o persoană în apropiere, iar pe lângă informațiile legate de temperatură și umiditate, se afișează și vremea pe decursul a cinci zile.
- Este prezentă tehnologia Geofencing

Date tehnice [10]:

- Sursă de energie: baterii sau rețea
- Tensiune de funcționare: 24 volți
- Ecran tactil
- Conectivitate: WiFi



2.5 Abordare comparativă a sistemelor prezentate

Dacă primele două sisteme prezentate reprezentau soluții mai ieftine, ultimele două sunt recunoscute ca fiind cele mai evaluate termostate existente pe piață. Diferențele esențiale sunt vizibile la nivelul funcționalităților oferite, dar și al calității.

Lipsa posibilității de a programa în avans temperaturi pe mai multe intervale orare și zile, de a monitoriza temperatura în mai multe camere, de a stoca un istoric al consumului de energie și de a adapta temperaturile în funcție de rutina locuitorilor, sunt câteva din minusurile termostatelor Torelectric și Honeywell. Google Nest și Ecobee, pe lângă faptul că rezolvă aceste probleme, oferă și o experiență mult mai bună a utilizatorului. Interfețele sunt proiectate în așa fel încât să permită o navigare facilă prin meniurile sistemelor.

În continuare, cu ajutorul datelor preluate din [11], voi prezenta o analiză comparativă a sistemelor oferite de Google Nest și Ecobee.

Termostatul Nest înregistrează fiecare modificare de temperatură pe care utilizatorul o face. După o perioadă de timp, acesta va programa automat temperaturile din imobil în funcție de istoricul de temperaturi setate de utilizator. Ecobee nu prezintă această funcționalitate, dar oferă posibilitatea de a seta temperaturi diferite pe intervale orare diferite.

În ceea ce privește controlul prin comenzi vocale, Nest este compatibil cu un număr mai mic de asistenți virtuali (Google Assistant și Alexa). Ecobee este compatibil și cu Siri, fapt ce reprezintă un avantaj datorită numărului mai mare de potențiali clienți.

Tehnologia Geofencing constă în detecția momentului în care imobilul este gol și setarea unei temperaturi ce asigură un consum minim de energie. Totodată, poate detecta dacă locuitorii se apropie de imobil, pentru a ieși din modul economic și a asigura temperatura setată. Geofencing este prezent la ambele termostate, însă la Nest este mai avansat. Poate detecta mai multe telefoane, iar în cazul în care sunt persoane care rămân în locuință și nu au telefonul conectat la Geofencing, prezența acestora este sesizată prin detectoarele de fum Nest, iar modul eco nu va fi activat. Ecobee poate conecta un singur telefon la Geofencing, ceea ce poate reprezenta un dezavantaj.

Monitorizarea consumului de energie se face automat, dar cei de la Ecobee au pus mai mult accent pe această parte, iar raportul pe care termostatul îl face este mult mai detaliat. Se analizează datele în decursul a 18 luni și conține informații legate de: consumul total, influența vremii asupra consumului de energie și compară eficiența imobilului cu celelalte din zonă. Pe de altă parte, Nest poate înregistra doar consumul în decursul a 10 zile. Acesta trimite mail-uri în fiecare lună cu un raport ce conține detalii de consum de energie și face comparație cu luna anterioară.

Monitorizarea temperaturii se face în mai multe zone din imobil prin intermediul unor senzori adiționali. Cele două termostate încearcă să mențină o temperatură medie în locuință. Senzorii oferiți de Ecobee detectează și dacă se află cineva în încăperea respectivă.

2.6 Comparație cu sistemul prezentat în această lucrare

În urma analizei sistemelor existente pe piață, se pot observa două dezavantaje majore. În primul rând, menținerea unor temperaturi exacte în fiecare cameră din imobil este destul de greu de obținut. Chiar dacă se pot adăuga mai mulți senzori de temperatură, termostatele prezentate vor reuși să asigure o temperatură medie la nivel de imobil, și nu o temperatură specifică la nivel de cameră. De asemenea, prețurile acestora sunt destul de ridicate. Prin proiectul de diplomă, am încercat să soluționez aceste probleme. Costurile pieselor pe care le-am utilizat sunt abordabile, iar electrovalvele montate pe returul caloriferelor au rolul de a asigura menținerea temperaturii setate în fiecare cameră.

Capitolul 3

Fundamente teoretice

În acest capitol voi prezenta principalele concepte teoretice utilizate în realizarea proiectului, împreună cu limbajele de programare și framework-urile folosite.

3.1 Python

Python este un limbaj de programare ce a apărut în anul 1991, fiind realizat de către Guido van Rossum [12]. Se bucură de o evoluție fulminantă, ajungând să fie unul din cele mai utilizate limbaje de programare în anul 2020. Creșterea numărului de programatori care aleg să folosească Python, este datorată caracteristicilor precum [12]:

- Flexibilitatea, poate fi utilizat într-un număr vast de domenii, de la programare web, până la programare pe plăcuțe. Funcționează pe o multitudine de platforme, printre care se enumeră: Windows, Mac, Linux și Raspberry Pi.
- În ceea ce privește sintaxa acestui limbaj de programare, este una simplă, care permite scrierea de programe utilizând un număr mai mic de linii de cod.
- Poate fi folosit atât pentru programare procedurală, funcțională, dar și orientată pe obiecte.

3.2 Flask

Este un framework construit pe baza limbajului de programare Python, ce oferă posibilitatea de a dezvolta aplicații web. Faptul că este proiectat ca să fie extins, oferă posibilitatea programatorului de a avea control total asupra aplicației pe care o creează. Prezintă un nucleu robust, care include toate funcționalitățile de bază pe care o aplicație web le necesită, nucleu ce poate fi extins de diverse părți terțe [13].

Pentru a crea aplicații web complexe, utilizarea doar a framework-ului nu este suficientă. Motiv pentru care, flask permite îmbinarea cu limbaje de programare precum javascript, CSS și HTML.

3.3 C++

C++ este un limbaj de programare bazat pe C. Motivele pentru care creatorul acestui limbaj de programare, Bjarne Stroustrup, a decis să folosească limbajul C ca punct de plecare sunt: flexibilitatea și faptul că este un limbaj apropiat de partea hardware, rulează pe multe platforme și se potrivește cu mediul de programare UNIX [14].

Ceea ce aduce nou este posibilitatea de a programa orientat pe obiecte, o programare de tip generic și face posibil conceptul de abstractizare al datelor [14]. Numărul de domenii în care C++ poate fi aplicat crește considerabil datorită introducerii noțiunii de programare orientată pe obiecte. Aceasta implică modelarea unor entități din lumea reală, și interacțiunile acestora, prin intermediul claselor.

3.4 Arduino IDE

Pentru programarea plăcuțelor Arduino, se folosește un mediu de dezvoltare numit Arduino Integrated Development Environment. Acesta suportă limbajele de programare C și C++ [15].

Poate rula pe mai multe platforme precum: Windows, Linux, MAC și Java [15]. De asemenea, este important de menționat faptul că este compatibil cu o serie largă de modele de plăcuțe, câteva exemple fiind [15]:

- Arduino Uno
- Arduino Mega
- Arduino Leonardo
- Arduino Micro

Arduino IDE îndeplinește atât rolul de editor de text, cât și rolul de compilator. Editorul de text reprezintă un suport pentru redactarea codului, iar compilatorul este responsabil de transformarea codului sursă în cod obiect și încărcarea acestuia pe microcontroler [15].

3.5 Componente utilizate

3.5.1 ESP8266

NodeMCU ESP8266 este o placă ce se poate conecta prin WiFi la o rețea de internet. Prezintă o antenă esp8266 ce acceptă standardele 802.11b/g/n și protocolul de securitate WPA/WPA2 [16]. Integrează un modul ADC pe 10 biți, microcontroler pe 32 de biți, ce are un consum redus de energie, și se bazează pe protocolul TCP/IP pentru a face transferul de date [16].

3.5.2 Placă de bază pentru NodeMCU

Are rolul de a alimenta modulul wireless ESP8266 și de a oferi o extensie pentru pinii pe care acesta îi pune la dispoziție. În ceea ce privește alimentarea, se face printr-un conector de tip jack și acceptă valori între 6 și 24 de volți. De asemenea, placa de bază integrează un regulator de tensiune ce convertește valoarea tensiunii de intrare la 5 volți.

3.5.3 Arduino Uno

Arduino aduce pe piață o serie de plăcuțe cu microcontroler, ce pot fi programate pentru diverse aplicații. De asemenea, bibliotecile puse la dispoziție pentru acest tip de sisteme sunt menite să faciliteze procesul de programare al acestora [17].

Programarea plăcuței se face prin intermediul conectorului USB. Aceasta prezintă și un conector separat pentru alimentare care suportă tensiuni în intervalul 7 - 12 volți [17], tensiuni ce vor trece prin regulatorul de tensiune integrat în plăcuță.

În ceea ce privește pinii prezenți pe placa Arduino Uno, există 6 intrări analogice [17], 14 terminale care pot fi configurate fie ca intrări, fie ca ieșiri [17] și o serie de pini ce furnizează tensiune, 3.3 sau 5 volți.

3.5.4 LCD 1602

Oferă posibilitatea de a afișa text pe două rânduri, fiecare rând conținând 16 caractere. Transferul de date se face prin protocolul I^2C , fapt ce reduce considerabil numărul de pini necesari pentru a conecta LCD-ul la Arduino [18].

Printre caracteristici se remarcă [18]:

- Tensiune de alimentare: 5 volți
- Luminozitatea ecranului se poate regla printr-un potențiometru integrat

3.5.5 Senzor DHT11

Prezintă două părți componente. Prima componentă are rolul de a citi umiditatea ambientală. Este alcătuită din doi electrozi care se suprapun peste un substrat. În momentul în care umiditatea substratului se modifică, determină o modificare a rezistenței dintre cei doi electrozi, iar microcontrolerul detectează și interpretează această valoare. Componenta termică este formată dintr-un termistor. Termistorul este un rezistor a cărui rezistență este invers proporțională cu variația temperaturii. Pe baza valorii rezistenței date de termistor, se vor face prelucrări ajungându-se la o valoare validă a temperaturii.

3.5.6 Modul radio frecvență

Pereche formată dintr-un transmițător și receptor. Construite pentru a facilita transferul de date prin radio-frecvență. Acestea funcționează la o frecvență de 433Mhz [19], iar distanța de transmisie diferă în funcție de tensiunea de alimentare a transmițătorului și de calitatea antenelor.

Conform [19], se pot evidenția următoarele caracteristici:

- Distanța de transfer: 20 - 200 metri
- Tensiune de alimentare transmițător: 3.5 - 12 volți
- Rată de transfer: 4 KB/S
- Tensiune de alimentare a receptorului: 5 volți

3.5.7 Releu

Funcționează ca un întrerupător într-un circuit electric. Principiul de bază al unui releu constă în alăturarea unui solenoid și a unor contacte metalice. În momentul în care solenoidul este alimentat, acesta produce un câmp electromagnetic ce va acționa contactele metalice, deschizând sau închizând circuitul. Releele pe care le utilizez funcționează la 5 volți și au o structură puțin mai complexă, structură pe care intenționez să o detaliez. Prezintă trei intrări: VCC, GND și IN, unde pinii de VCC și GND sunt alimentați permanent, iar IN este pinul de semnal. Acesta este legat la baza unui tranzistor, iar în funcție de tensiunea pe care o furnizează, tranzistorul trece în regim saturat sau blocat, funcționând ca un întrerupător pentru solenoid. De asemenea, în paralel cu bobina este montată o diodă ce are ca și scop protejarea tranzistorului de șocurile de tensiune ce pot apărea la deconectarea alimentării bobinei. Este necesară utilizarea releului pentru a putea controla, prin semnale de putere mică, dispozitive ce funcționează la tensiuni și curenți mari.

3.5.8 Electrovalvă

Este un mecanism ce are ca rol deschiderea circuitului de apă în momentul în care este conectat la o sursă de tensiune. Acesta este format dintr-un solenoid și un obturator. Electrovalvele pe care le folosesc sunt de tip normal închis, ceea ce înseamnă că atunci când solenoidul nu este alimentat, obturatorul stă în poziție închis, iar circuitul de apă prin electrovalvă este blocat. În momentul alimentării solenoidului, câmpul magnetic produs de acesta va acționa obturatorul, făcând posibilă trecerea apei prin electrovalvă.

3.5.9 Pompă de apă

Este alcătuită dintr-un motor electric ce funcționează la 12 volți, curent continuu și acționează o paletă ce pune în mișcare apa din circuit.

3.5.10 Push buton

Este un intrerupător, fără reținere, ce are rolul de a trimite semnale către micro-controler. Prezintă două contacte metalice care închid circuitul în momentul în care se ating.

3.5.11 Condensator

Este o componentă electrică pasivă ce are rolul de a înmagazina tensiune. Acesta se utilizează în circuitele electrice pentru a reduce fluctuațiile de tensiune ce pot apărea.

3.5.12 Rezistență

Asemenea condensatorului, rezistența se încadrează în categoria componentelor electrice pasive. Aceasta are rolul de a se opune trecerii curentului electric.

3.5.13 Circuit integrat Schmitt-Trigger

Acesta este utilizat pentru a transforma un semnal analog într-un semnal digital, dar prezintă și rolul de inversor. Dacă se aplică la intrare un semnal cu nivel logic 1, la ieșirea din circuitul integrat va avea nivelul logic 0.

3.5.14 Diodă

Limitează trecerea curentului electric într-un singur sens, de la anod la catod. Prezintă multe utilități în circuitele electrice, un exemplu practic fiind conversia curentului alternativ în curent continuu.

3.6 Filtrarea semnalelor

Butoanele pe care le-am folosit, pentru setarea temperaturii dorite, sunt formate din contacte metalice. În momentul în care acestea se ating, produc o vibrație pe care microcontrolerul o percepe ca o apăsare multiplă a butonului. Pentru a rezolva această problemă este necesară crearea unor filtre trece jos [20] pentru fiecare buton în parte. Rolul acestor filtre este de a permite trecerea semnalelor cu frecvență mică și de a opri semnalele cu frecvențe mari. Pentru aceasta, am utilizat circuite RC serie, iar valorile rezistenței și condensatorului au fost calculate utilizând ecuația de încărcare și de descărcare a condensatorului. De asemenea, pentru a obține o filtrare cât mai bună, am utilizat un circuit integrat Schmitt-Trigger. Caracteristica datorită căreia acesta îmbunătățește filtrarea semnalului este histereza pe care o are între pragul superior, declanșează trecerea pe nivel logic 0, și pragul inferior, declanșează trecerea pe nivel logic 1.

Conform [20], ecuația de încărcare a condensatorului este:

$$V_{cap} = V_{in}(1 - e^{-\frac{t}{RC}}) \quad (3.1)$$

Conform [20], ecuația de descărcare a condensatorului este:

$$V_{cap} = V_{in}(e^{-\frac{t}{RC}}) \quad (3.2)$$

V_{cap} - tensiunea stocată în condensator la momentul t

V_{in} - tensiune de alimentare a circuitului

t - timpul măsurat din momentul aplicării tensiunii de alimentare la bornele circuitului, în cazul ecuației de încărcare a condensatorului. Pentru ecuația de descărcare a condensatorului, reprezintă timpul măsurat din momentul întreruperii alimentării circuitului

R - valoarea în ohmi a rezistenței utilizate

C - valoarea în farazi a condensatorului utilizat

În continuare, pentru a putea aplica formulele, este necesară aflarea timpului cât durează oscilația semnalului. Pentru aceasta, am utilizat un osciloscop ce permite analiza semnalului provenit de la buton.

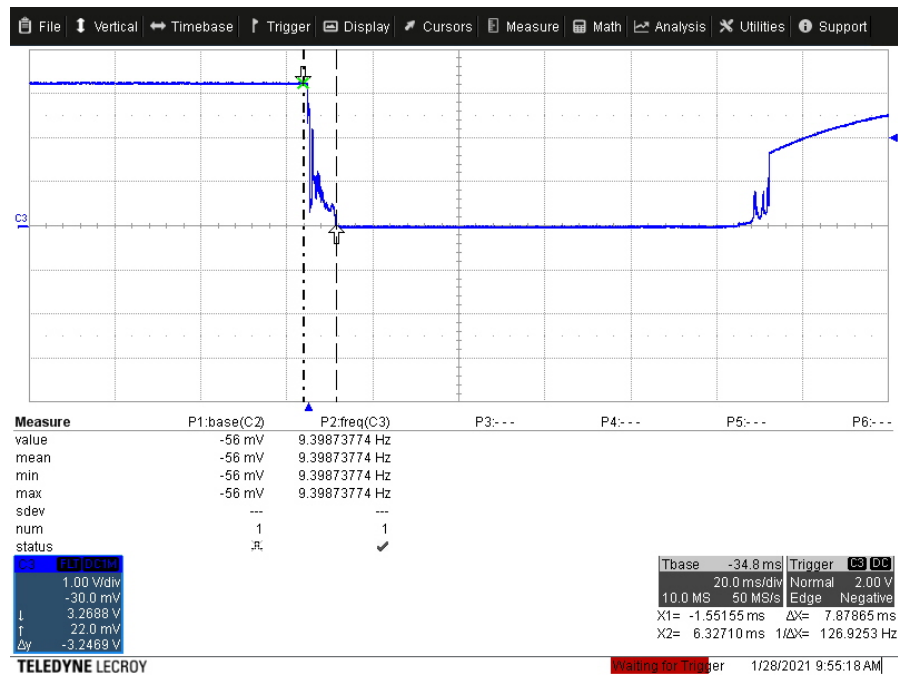


Figura 3.1: Semnal înainte de filtrul trece jos

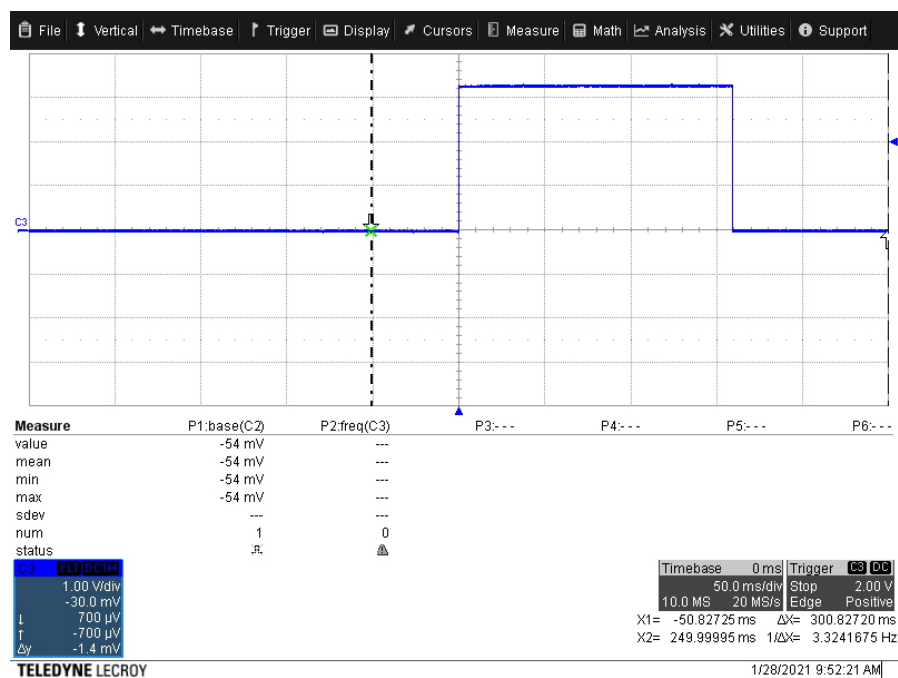


Figura 3.2: Semnal după filtrul trece jos

În urma analizei mai multor semnale de la push buton, cea mai mare perturbație a durat aproximativ 8 ms. În calculul valorii rezistențelor și condensatorului care trebuie folosite în filtrele trece jos, voi folosi un timp de 10 ms. Cele 2 ms în plus reprezintă o marjă de siguranță pentru situațiile în care apar perturbații mai mari decât cele măsurate.

În cele ce urmează, voi prezenta schema filtrului trece jos, iar pe baza acesteia voi detalia calculele realizate pentru a determina valorile componentelor electrice utilizate.

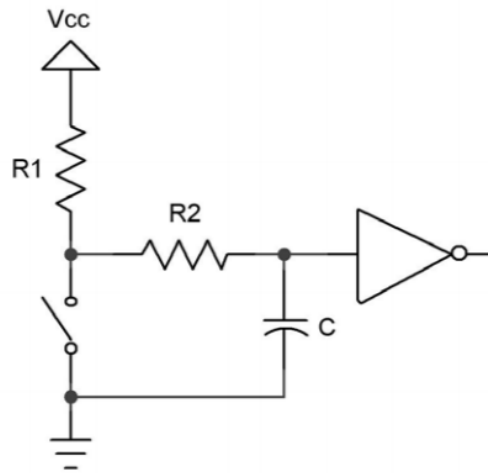


Figura 3.3: Schemă filtru trece-jos (sursa: [20])

În momentul în care întrerupătorul se închide, condensatorul începe să se descarce prin intermediul rezistenței R_2 . Cunoscând acest aspect, voi putea calcula valoarea R_2 utilizând ecuația de descărcare a condensatorului. Valoarea lui t va fi de 10 ms, V_{in} este echivalent cu 3.3 volți (tensiunea la care funcționează modulul WiFi), V_{cap} este egal cu valoarea pragului inferior al inversorului Schmitt-Trigger, iar valoarea condensatorului o aleg de $1 \mu\text{F}$. Mai exact, se calculează rezistența astfel încât, după $t = 10 \text{ ms}$ de la începerea descărcării, tensiunea stocată în condensator să ajungă la valoarea de prag a circuitului integrat.

$$R_2 = \frac{-t}{C \ln\left(\frac{V_{cap}}{V_{in}}\right)} = \frac{-10 \cdot 10^{-3}}{10^{-6} \ln\left(\frac{1}{3.3}\right)} = \frac{-10}{\ln(0.3)} \cdot 10^3 = \frac{-10}{-1.2} \cdot 10^3 = 8.33 \cdot 10^3 = 8.33 \text{ K}\Omega$$

Când se deschide întrerupătorul, condensatorul începe să se încarce prin intermediul rezistențelor R_1 și R_2 . Pornind de la această informație, voi calcula valoarea $R_1 + R_2$ utilizând ecuația de încărcare a condensatorului. Doar valoarea parametrului V_{cap} se modifică, va fi echivalentă cu pragul superior al inversorului.

$$\begin{aligned}
 R_1 + R_2 &= \frac{-t}{C \ln(1 - \frac{V_{cap}}{V_{in}})} = \frac{-10 \cdot 10^{-3}}{10^{-6} \ln(1 - \frac{1.5}{3.3})} = \frac{-10}{\ln(0.54)} \cdot 10^3 = \frac{-10}{-0.61} \cdot 10^3 = 16.39 \cdot 10^3 \\
 &= 16.39 K\Omega \\
 R_1 + R_2 &= 16.39 K\Omega, R_2 = 8.33 K\Omega \Rightarrow R_1 = 8.06 K\Omega
 \end{aligned}$$

3.7 Histereza

Pentru a evita crearea unui lanț de porniri si opriri repetate, pe perioade scurte de timp, cauzate de fluctuațiile rapide de temperatură, este necesară implementarea conceptului de histereză. Acesta constă în stabilirea unei valori de toleranță la temperatura setată, astfel încât diferența de temperatură din momentul în care se trimite comandă pentru oprirea sistemului de încălzire, până la următoarea pornire a acestuia, să fie egală cu dublul valorii histerezei. Cu alte cuvinte, comanda de pornire a încălzirii se va da când temperatura în cameră ajunge să fie mai mică sau egală cu valoarea temperaturii setate, din care se sustrage valoarea histerezei, iar comanda de oprire va fi declanșată când temperatura ambientală este mai mare sau egală cu valoarea temperaturii setate, la care se adaugă valoarea histerezei. În acest fel, timpul între comenzi este mai mare și numărul de cicluri pornit/oprit este mai mic, măsură ce este menită să protejeze sistemele de încălzire.

Capitolul 4

Dezvoltarea soluției

Acest capitol conține o prezentare detaliată a modului în care a fost implementat proiectul. Voi face referiri atât la partea hardware, cât și la partea software.

4.1 Specificarea cerințelor

Redactarea cerințelor sistemului reprezintă o etapă esențială în crearea oricărui proiect. Acestea dictează comportamentul produsului final, fiind necesare atât în faza de dezvoltare, cât și în cea de testare. Cu ajutorul diagramelor cazurilor de utilizare, voi detalia cerințele sistemului descris în această lucrare.

4.1.1 Cerințele funcționale ale aplicației web

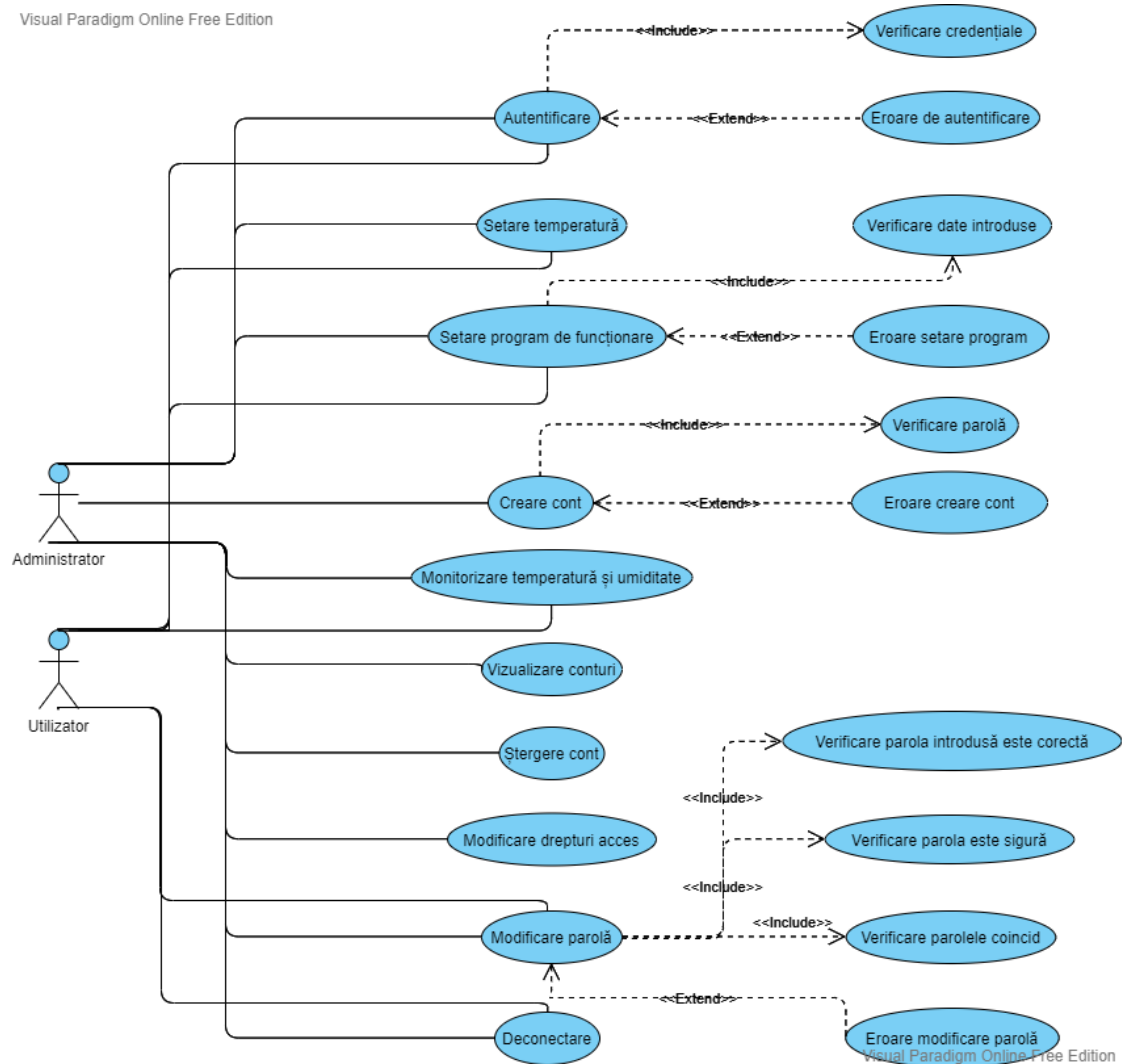


Figura 4.1: Cazurile de utilizare pentru aplicația web

Prima cerință pe care aplicația trebuie să o îndeplinească este de a permite utilizatorilor să se autentifice cu ajutorul credențialelor, indiferent de rolul utilizatorului. De asemenea, trebuie tratate situațiile în care parola este greșită sau contul nu există în baza de date.

Prin intermediul aplicației web, orice utilizator trebuie să poată monitoriza valoarea temperaturii și umidității din imobil.

Setarea temperaturii trebuie să fie posibilă prin intermediul aplicației web, iar această acțiune nu depinde de rolul utilizatorului.

O cerință esențială a sistemului este de a oferi posibilitatea setării, de la distanță, a unui program de funcționare. Această funcționalitate trebuie să fie accesibilă atât pentru conturile cu drepturi de administrator, cât și pentru cele fără. Este necesar ca datele introduse de utilizator să fie verificate și semnalate cazurile în care valorile temperaturilor nu sunt valide sau intervalele nu sunt setate în ordine cronologică.

Pentru a putea gestiona accesul la aplicație, aceasta trebuie să permită administratorilor să creeze, să șteargă conturi pentru alte persoane și să modifice drepturile de acces ale tuturor conturilor salvate în baza de date. Prin urmare, apare nevoia de noi funcționalități, precum: creare cont, ștergere cont și modificare drepturi de acces. La crearea unui cont, se va verifica dacă parola introdusă este una sigură. Validarea se face pe baza a trei criterii: parola trebuie să conțină cel puțin 6 caractere, cel puțin o literă mare și cel puțin o cifră. În cazul în care nu se respectă condițiile, utilizatorul trebuie să fie avertizat, iar contul nu va fi creat.

Cele 3 cerințe menționate mai sus duc la apariția unei noi cerințe. Pentru a facilita ștergerea conturilor și modificarea drepturilor de acces, aplicația web trebuie să pună la dispoziție o pagină care să afișeze toate conturile existente în baza de date. Aceasta trebuie să fie accesibilă doar administratorilor.

De asemenea, din motive de securitate, trebuie ca fiecare utilizator să poată să își modifice parola. Acest caz de utilizare implică mai multe validări. Este necesar să se verifice dacă parola actuală introdusă este cea corectă. După aceea, identic cazului de utilizare modificare parolă, se verifică dacă noua parolă introdusă este una sigură. În cele din urmă, trebuie verificat dacă parola nouă și confirmarea acesteia coincid. Nerespectarea condițiilor duce la afișarea unui mesaj de eroare și la nefinalizarea acțiunii de schimbare a parolei.

După finalizarea acțiunilor dorite, utilizatorii trebuie să se poată deconecta de la aplicație.

4.1.2 Cerințe nefuncționale ale aplicației web

În continuare, voi enumera cerințele legate de securitate:

- Permisunile de acces pot fi modificate doar de administratori.
- Toate parolele trebuie să fie criptate.
- Parolele trebuie să fie complexe.

De asemenea, pe partea de performanță:

- Aplicația web trebuie să poată monitoriza parametrii ambientali și seta temperatura într-un imobil cu două camere.

- Valorile afișate pe pagina principală trebuie actualizate doar în momentul în care acestea se modifică în baza de date. Nu se acceptă ca actualizările să se execute la anumite intervale de timp prestabilite.

4.1.3 Cerințe funcționale pentru componentele electronice

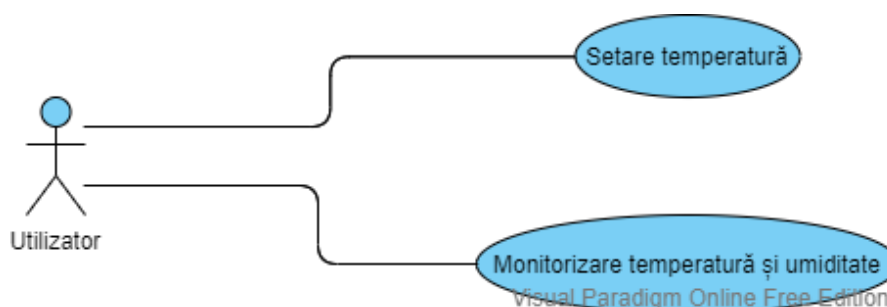


Figura 4.2: Cazurile de utilizare pentru ansamblul de componente electronice

Ansamblul componentelor electronice trebuie să permită modificarea temperaturii setate. Pentru aceasta, se utilizează butoane, iar modificările de temperatură se afișează pe LCD.

Pentru a face posibilă monitorizarea parametrilor ambientali, modulele WiFi trebuie să afișeze pe LCD valorile citite de senzorul de temperatură și umiditate.

Dacă nu există o conexiune la internet în momentul în care sistemul este pornit, este necesar să se considere valoarea prestabilită de 21 °C ca fiind temperatura ce trebuie menținută în imobil. În cazul întreruperii conexiunii la internet în timpul funcționării, trebuie ca ultima valoare a temperaturii dorite, citită din baza de date, să rămână salvată în sistem.

4.1.4 Cerințe nefuncționale pentru componentele electronice

O serie de cerințe arhitecturale ale sistemului sunt:

- Sistemul trebuie să ruleze într-un regim continuu, zi și noapte.
- Pentru a facilita montajul sistemului, trebuie ca transferul comenzilor între modulele senzor și modulul de control să se facă prin radio-frecvență.
- Distanța maximă de transfer între modulele senzor și modulul de control trebuie să fie de 200 de metri, în spațiu deschis.

Pe partea de siguranță, este necesar ca:

- Centrala termică să fie oprită dacă transferul de date între modulele senzor și modulul de control este întrerupt mai mult de 5 secunde.

În ceea ce privește fiabilitate:

- Sistemul trebuie să poată funcționa și în cazul întreruperii conexiunii la internet.
- Pentru a evita situațiile în care o comandă nu este recepționată, transferul datelor între modulele senzor și modulul de control trebuie să se facă ciclic.

4.2 Arhitectura sistemului

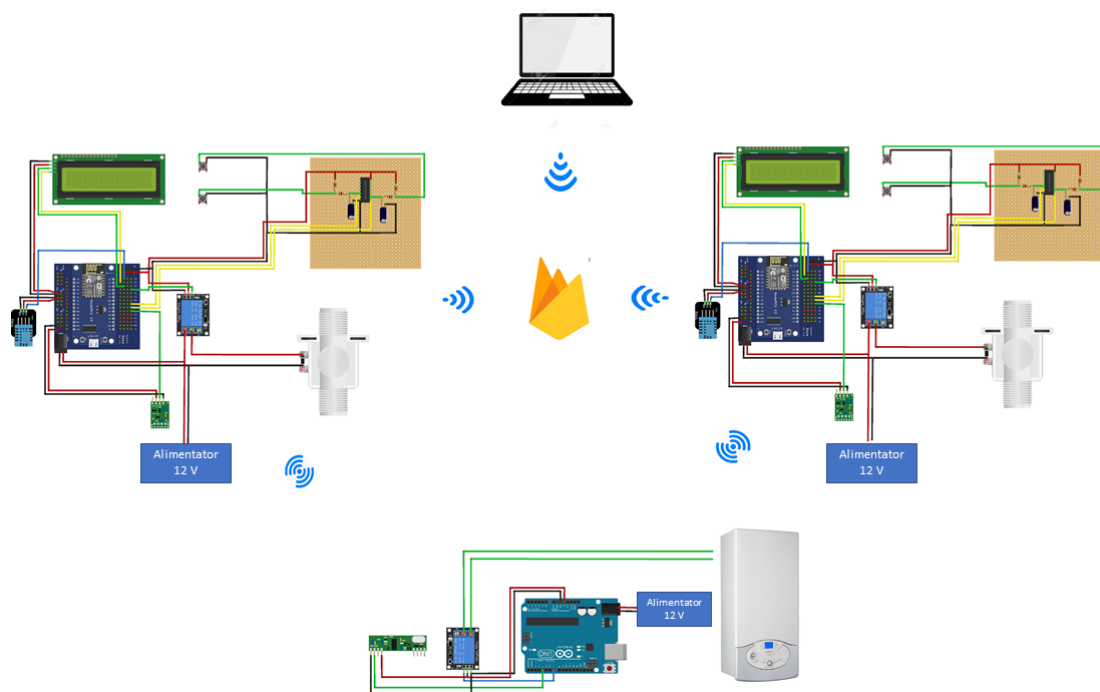


Figura 4.3: Arhitectura sistemului

Sistemul a fost conceput pentru un apartament cu două camere, însă poate fi extins pentru un număr mai mare de încăperi. La nivel macro, este format din trei module, câte un modul senzor pentru fiecare cameră și un modul de control, montat în apropierea centralei termice.

Modulul senzor prezintă o complexitate mai mare, este alcătuit din mai multe componente și îndeplinește o serie de funcționalități. Preia informații precum: temperatura

și umiditatea și le trimite în baza de date. În continuare, aceste valori sunt afișate utilizând aplicația web, făcând posibilă monitorizarea parametrilor ambienali. De asemenea, are acces la temperatura setată, iar în funcție de aceasta, dar și de temperatura curentă din cameră, controlează electrovalvă și trimite comandă de oprire sau pornire a centralei. Un alt rol esențial pe care îl îndeplinește modulul este de a face posibilă setarea unei valori a temperaturii ce urmează să fie menținută în cameră.

Modulul de control este de o complexitate redusă. Primește comenzi de la modulele senzor, le interpretează, iar pe baza acestora controlează centrala. Modulul se folosește de logică de tip "SAU", dacă cel puțin un modul senzor trimite comandă de pornire, atunci centrala termică trebuie să înceapă ciclul de încălzire.

Transferul de date între modulele senzor și modulul de control se face prin intermediul undelor radio, 433 MHz. Placuțele ESP8266 se folosesc de unde Wi-Fi, 2.4 GHz, pentru a se conecta la router.

4.2.1 Modulul senzor

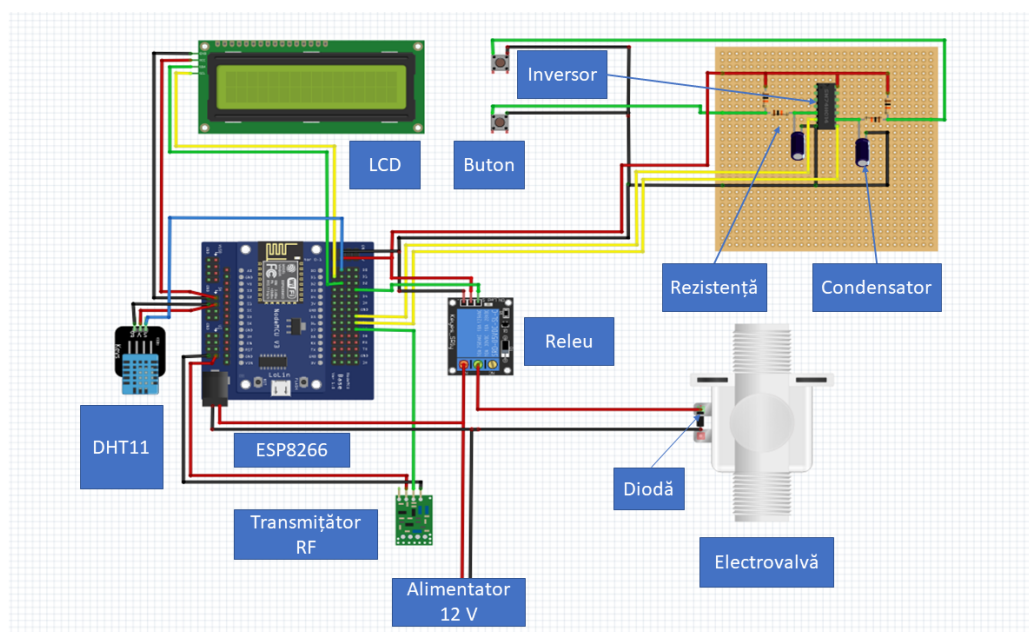


Figura 4.4: Arhitectura modului de senzori

Este format din următoarele componente:

- Modul WiFi NodeMCU
- Placă de bază pentru NodeMCU
- LCD

- Senzor de temperatură și umiditate
- Releu
- Butoane
- Transmițător RF
- Electrovalvă
- Circuit integrat Schmitt-Trigger
- Rezistență
- Condensator
- Alimentator 12 volți

Componenta principală din acest ansamblu este modulul wireless. Se conectează prin WiFi la un punct de acces al rețelei, router-ul în cazul proiectului prezentat, și face transfer de date cu baza de date utilizată. Acesta pune la dispoziție o serie de pini pentru a putea conecta diverse componente auxiliare.

Pinul D0 este utilizat pentru a conecta pinul de date de la senzorul de temperatură și umiditate.

Pinii D1 și D2 se conectează la pinii SCL și SDA pentru a face posibil transferul de date prin protocolul de comunicație I^2C .

Pinul D3 se conectează la pinul IN al releului și are rolul de a controla închiderea sau deschiderea releului, iar pinii D5 și D6 sunt rezervați pentru butoane.

Pinul D7 transferă date la pinul de date al transmițătorului RF.

Placa de bază pentru modulul WiFi are două roluri esențiale în crearea circuitului. Acesta facilitează felul în care se poate alimenta plăcuța wireless, punând la dispoziție un conector de alimentare care este compatibil cu alimentatoare ce furnizează tensiuni între 6 și 24 de volți. Prin intermediul unui regulator de tensiune, va aduce tensiunea de alimentare a modulului WiFi până la 5 volți, prevenind orice posibilitate de supra-alimentare. Astfel, placa de bază pune la dispoziție pini ce oferă tensiuni de ieșire de 3.3 volți, 5 volți și o serie de pini ce sunt conectați direct la conectorul de alimentare și furnizează voltaj egal cu cel dat de alimentator. De asemenea, oferă pentru fiecare pin al modulului ESP8266 încă 4 pini corespondenți, fapt ce reprezintă un avantaj când vine vorba de conectarea unor componente auxiliare.

Pentru a putea afișa temperatura curentă din cameră și temperatura setată, se folosește un display. Transferul de date între LCD și modulul WiFi, la care este conectat, se face prin intermediul protocolului de comunicație I^2C . S-a ales acest protocol deoarece reduce considerabil numărul de fire necesare pentru conectarea și alimentarea LCD-ului, în acest caz fiind necesare doar 4. Prin urmare, display-ul pune la dispoziție următorii pini: VCC, GND, SDA și SCL.

Senzorul de temperatură și umiditate pe care îl utilizez prezintă 3 pini. Doi sunt utilizați pentru alimentare, VCC și GND, iar cel de-al treilea este pinul de date, utilizat pentru transferul de informații între senzor și plăcuța cu microcontroler la care este conectat.

Releul este conectat la modulul wireless și este controlat de acesta prin pinul IN. Rolul releului este de întrerupător în circuitul electric format din electrovalvă și alimentator.

Sunt prezente două butoane ce permit modificarea temperaturii setate. Butonul de sus incrementează temperatura și se conectează la pinul D5 al modului WiFi, iar butonul de jos decrementează valoarea temperaturii și se conectează la pinul D6.

Transmițătorul RF prezintă 3 pini: VCC, GND și DATA. Pinii VCC și GND se conectează la o sursă de tensiune, ieșirile de 12 volți oferite de placa de bază NodeMCU. Pinul de date se conectează la pinul digital D7 al modulului wireless și face posibil transferul de date între acestea. În continuare, datele vor fi transferate la receptorul de tip RF, montat pe modulul de control.

Electrovalva se montează pe returul caloriferelor și este responsabilă de închiderea sau deschiderea circuitului de apă din calorifer. Releul este cel care întrerupe, sau nu, alimentarea electrică a electrovalvei.

Circuitul integrat Schmitt-Trigger, rezistența și condensatorul sunt componente electrice pe care le-am utilizat pentru a filtra semnalul provenit de la buton.

4.2.2 Modulul de control

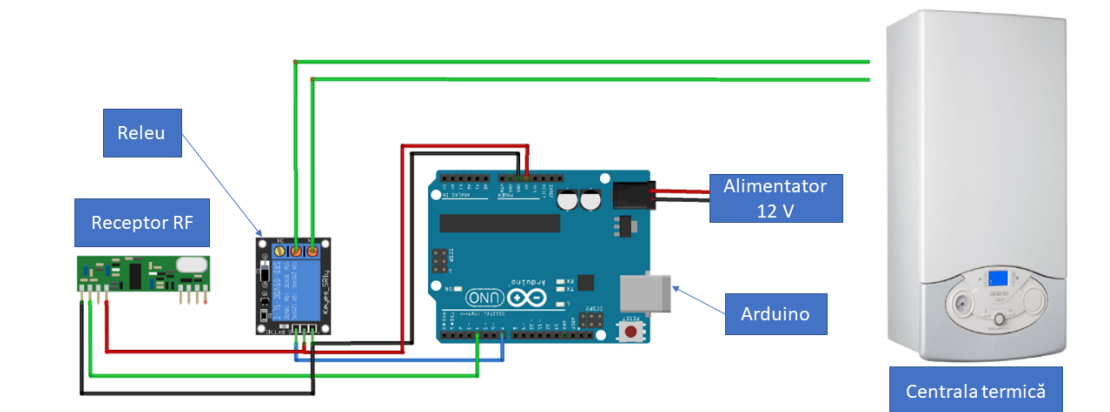


Figura 4.5: Arhitectura modului de control

Acesta nu prezintă conexiune la internet. Modulul wireless este înlocuit cu o plăcuță Arduino Uno, care se ocupă de partea de procesare a datelor primite de la modulele din camere. Prezintă o serie de intrări și ieșiri analogice, dar și digitale.

Pinul digital 4 se conectează la pinul de date al receptorului, făcând posibil transferul de informații între acestea.

Pinul digital 7 se conectează la pinul IN al releului și transmite comanda de închidere sau deschidere a acestuia.

Releul se alimentează la o tensiune de 5 volți, tensiune provenită de la Arduino. În momentul în care releul este pe poziție închis, centrala termică va porni.

Receptorul, asemenea releului, se alimentează cu 5 volți de la plăcuța Arduino și are rolul de a prelua datele de la transmițătoarele din camere.

4.3 Proiectare detaliată

Prin intermediul diagramelor UML voi detalia anumite aspecte ce țin de proiectarea sistemului. De asemenea, voi prezenta structura aplicației și codului aferent părții electronice.

4.3.1 Diagramă de clase

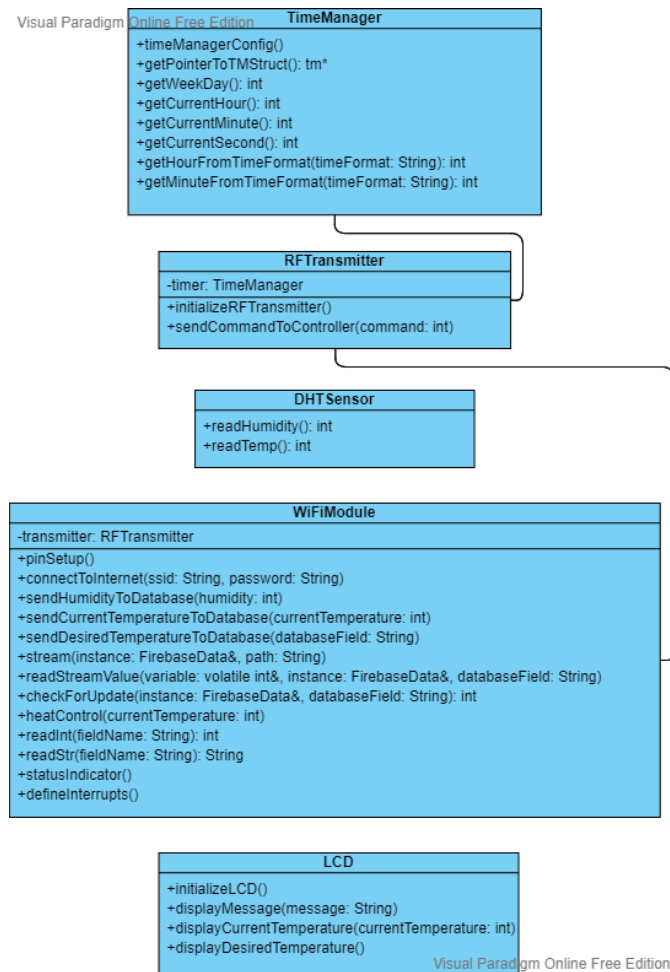


Figura 4.6: Diagramă de clase

Pentru fiecare componentă electronică, ce prezintă funcționalități mai complexe, am creat câte o clasă diferită. Acestea conțin metode ce implementează funcțiile pe care componenta trebuie să le îndeplinească. Cea mai complexă clasă este cea aferentă modulului WiFi, deoarece acesta îndeplinește un rol esențial în cadrul sistemului.

Metoda *heatControl*, din clasa **WiFiModule**, prezintă o instanță a clasei *RFTransmitter* pentru a putea apela *sendCommandToController*. De asemenea, pentru a evita conflictele de date ce pot apărea la transferul comenzilor prin radio-frecvență, metoda *sendCommandToController* va apela *getCurrentSecond* din cadrul clasei **TimeManager**.

4.3.2 Diagrame de secvență

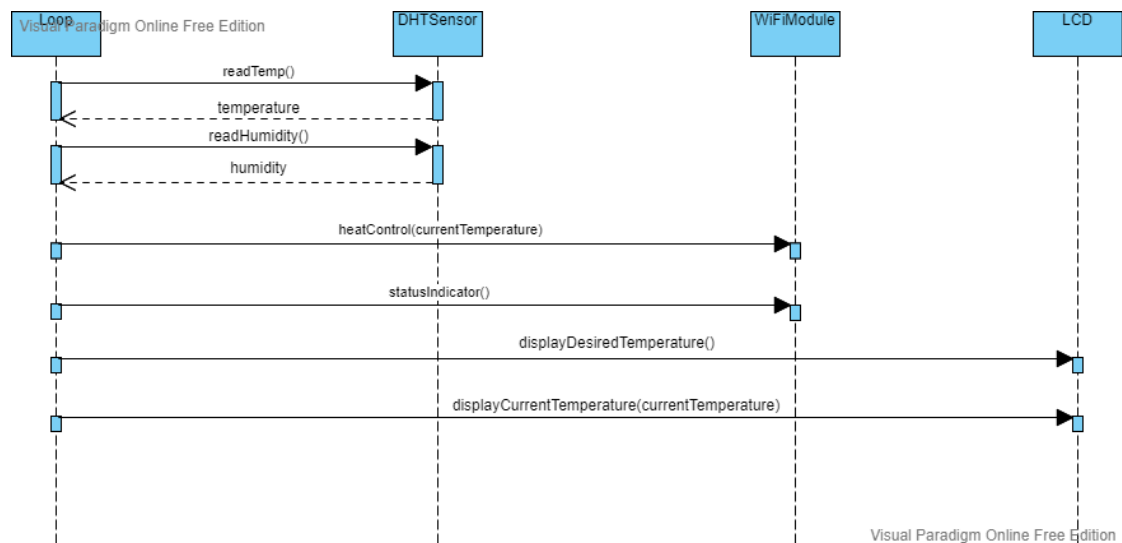


Figura 4.7: Mod de operare fără conexiune la internet

Sistemul trebuie să poată funcționa și în cazul în care are loc o întrerupere la conexiunea de internet. Electronica este proiectată în așa fel încât să poată prelua funcționalitățile de bază ale aplicației, și anume: monitorizarea și setarea temperaturii.

Pentru a obține acest comportament, se produc o serie de interacțiuni între obiecte. Temperatura și umiditatea sunt returnate în urma apelului metodelor `readTemp` și `readHumidity` pe o instanță a clasei `DHTSensor`. În continuare, utilizând un obiect de tip `WiFiModule`, se vor apela metodele responsabile pentru controlul sistemului de încălzire și indicarea statusului plăcuței. Dacă aceasta este conectată la internet, se va aprinde un led incorporat, altfel va rămâne stins. În cele din urmă, se apelează metodele din cadrul clasei `LCD` pentru a afișa valorile temperaturilor.

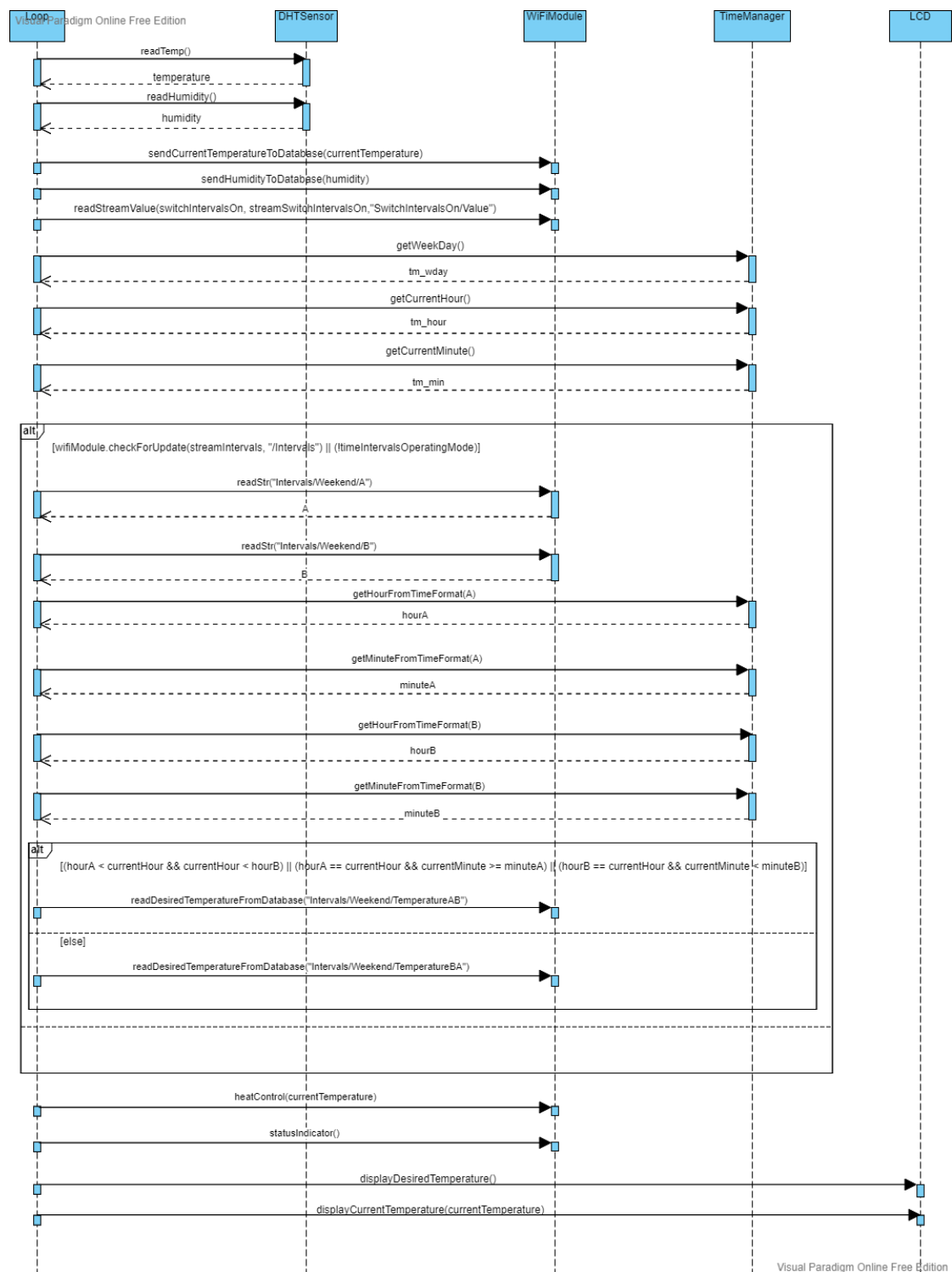


Figura 4.8: Mod de operare pe intervale orare

Diagrama de secvență prezentată este aferentă modului de funcționare pe intervale orare, în timpul zilelor din weekend. Sistemul permite setarea unor temperaturi diferite pe durata a două perioade, valabile doar sâmbăta și duminică. Această funcționalitate este valabilă dacă modul automat este activat și poate fi utilă, deoarece permite setarea unei temperaturi mai mici pe timpul nopții, având ca și efect un consum redus. Pentru zilele lucrătoare, modul de operare este identic, cu mențiunea că sistemul va citi și gestiona patru intervale orare.

Din punct de vedere funcțional, prezintă o complexitate mai mare, fapt ilustrat inclusiv de numărul de interacțiuni între instanțele claselor. Asemenea modului de operare prezentat anterior, apare interacțiunea cu clasa *DHTSensor* pentru a obține valorile umidității și temperaturii ambientale. Deoarece există o conexiune la internet, aceste valori vor fi stocate în baza de date. Motiv pentru care, este necesar apelul metodelor *sendCurrentTemperatureToDatabase* și *sendHumidityToDatabase* pe un obiect al clasei *WiFiModule*. Apelul următor este necesar pentru a actualiza variabila care reține modul actual de funcționare al sistemului.

Pentru a putea stabili în ce interval orar se încadrează sistemul, este necesar să se cunoască ziua, ora și minutul curent. Apelurile metodelor, din clasa *TimeManager*, *getWeekDay*, *getCurrentHour* și *getCurrentMinute* satisfac această necesitate. Dacă există modificări în baza de date, sau dacă s-a făcut o comutare de la modul clasic de operare la cel automat, este necesar să se recitească valorile intervalelor. De aceea, se apelează *readStr("Intervals/Weekend/A")* și *readStr("Intervals/Weekend/B")* pe instanța clasei *WiFiModule*. În continuare, apelurile metodelor din clasa *TimeManager* au rolul de a prelucra intervalele, returnate ca șir de caractere de apelul *readStr*. În urma blocului condițional, se va citi temperatura setată pentru intervalul curent.

Interacțiunile următoare, cu obiecte de tip *WiFiModule*, au rolul de a îndeplini funcții precum reglarea temperaturii și setarea statusului plăcuței ESP8266. Identic modului de operare fără conexiune la internet, metodele din clasa *LCD* sunt utilizate pentru a afișa valorile pe ecran.

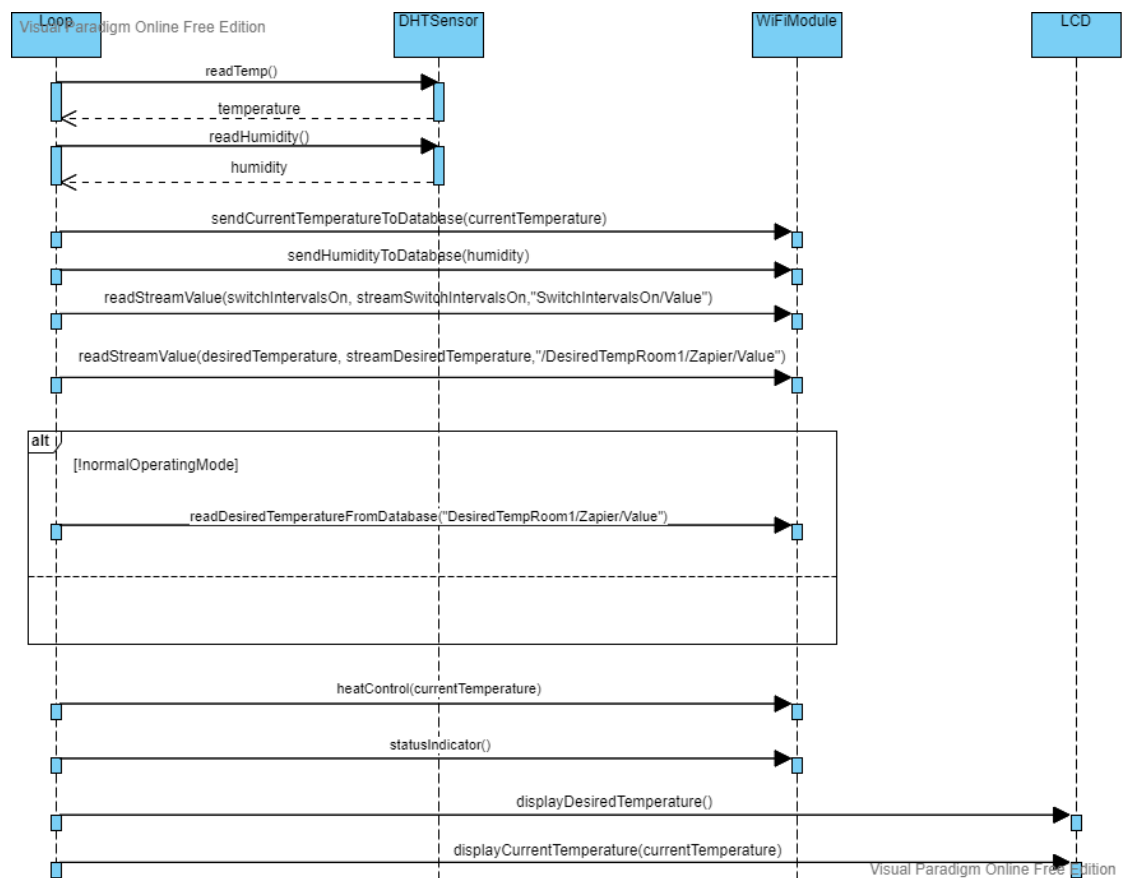


Figura 4.9: Mod operare clasic

Acest mod de operare menține temperatura setată de utilizator, fără a fi influențat de timp sau dată. Spre deosebire de modul de operare fără conexiune la internet, acesta face transfer de informații cu baza de date. Prin urmare, apar noi interacțiuni cu instanțe ale clasei *WiFiModule* pentru: salvarea temperaturii și umidității în baza de date, actualizarea variabilei *switchIntervalsOn* și actualizarea temperaturii setate.

Apelul metodei `readStreamValue(desiredTemperature, streamDesiredTemperature, "/DesiredTempRoom1/Zapier/Value")` va actualiza temperatura dorită doar când apar modificări. Însă, când se face o comutare de la modul automat la cel clasic, se folosește apelul `readDesiredTemperatureFromDatabase("/DesiredTempRoom1/Zapier/Value")`, deoarece este posibil ca datele din baza de date să nu fie modificate. Situație în care funcția `readStreamValue` nu ar returna nicio valoare.

4.3.3 Structură software

Structura aplicației web

Din punct de vedere structural, aplicația web conține două directoare cu implementări specifice părții de interfață cu utilizatorul. Acestea sunt:

- *templates* - alcătuit din fișiere de tip *html*.
- *static* - format din poze ce vor fi afișate în cadrul aplicației, fișiere *javascript* și *css*.

A fost necesară adăugarea directorului *functions*. Acesta este împărțit în module *python*, care implementează funcționalități precum: conversii de tipuri de date, validarea parolei, citirea orei și minutului curent și restricționarea accesului utilizatorilor, care nu sunt administratori, la anumite servicii ale aplicației.

Codul ce implementează logica programului este împărțit în 3 module *python*. Fiecare modul încapsulează funcții specifice categoriei în care se încadrează. De exemplu, modulul *accounts.py* va conține funcțiile pentru gestiunea conturilor din baza de date. Setarea temperaturilor, intervalelor orare și al modului de funcționare al sistemului sunt implementate în cadrul modului *controls.py*. În cele din urmă, funcțiile de bază, *login* și *logout*, sunt conținute în *main.py*.

Structură cod aferent componentei electronice

Modulele de cod ce implementează serviciile oferite de componenta electronică sunt:

- *softwareForESP8266* - conține implementarea tuturor funcționalităților oferite de modulul WiFi.
- *softwareForArduino* - prezintă logica din spatele modului de control al centralei termice.

De asemenea, codul prezent în modulul *softwareForESP8266* este divizat în mai multe clase, conform 4.6

4.4 Implementarea soluției

În acest subcapitol voi prezenta secvențe de cod folosite pentru programarea părții hardware, dar și cod aferent aplicației web.

4.4.1 Programare modul ESP8266

Inițializări

Modulul prezintă o serie vastă de funcționalități și se conectează cu mai multe componente auxiliare. Pentru programarea acestuia am utilizat ArduinoIDE, iar ca și limbaj de programare, C++.

Codul este împărțit în trei fișiere cu extensii diferite: .ino, .cpp și .h. În fișierul cu extensia .h se află prototipurile claselor și al metodelor, iar fișierul .cpp conține implementările acestor metode. Fișierul .ino este alcătuit din două funcții: *setup* și *loop*. Prima este utilizată pentru inițializări și se execută o singură dată, la pornirea modului. În ceea ce privește cea de-a doua funcție, se execută în mod repetitiv până la oprirea alimentării plăcuței WiFi.

În continuare, voi prezenta funcția *setup* din fișierul .ino și voi explica rolul apelurilor din cadrul acesteia.

```
#include "WiFiModule.h"

WiFiModule wifiModule;
LCD lcd;
TimeManager timeManager;
RFTransmitter rfTransmitter;
DHTSensor dhtSensor;

FirebaseData streamDesiredTemperature;
FirebaseData streamSwitchIntervalsOn;
FirebaseData streamIntervals;

void setup() {
    timeManager.timeManagerConfig();
    lcd.initializeLCD();
    rfTransmitter.initializeRFTransmitter();
    wifiModule.pinSetup();
    wifiModule.connectToInternet("Asus", "vitomir10");
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
    wifiModule.defineInterrupts();
    wifiModule.stream(streamDesiredTemperature,
        "/DesiredTempRoom2/Zapier/Value");
    wifiModule.stream(streamSwitchIntervalsOn,
        "/SwitchIntervalsOn/Value");
    wifiModule.stream(streamIntervals, "/Intervals");
    delay(500);
```

```
}
```

Pentru început, se creează instanțe pentru clasele ale căror funcționalități urmează să fie utilizate, iar în corpul funcției *setup* se vor apela funcții ce țin de inițializare.

Din clasa *TimeManager* se apelează metoda *timeManagerConfig*, fiind responsabilă de setarea fusului orar. Timpul curent este folosit în modul de funcționare automat, mod în care se ține cont de temperaturile setate de utilizator pe anumite intervale orare.

Până se execută toate funcțiile de inițializare, LCD-ul va fi pornit și va afișa mesajul *"Initializing..."*. Pentru ca acest comportament să fie posibil, este necesar apelul metodei *lcd.initializeLCD()*.

Apelarea metodei *initializeRFTransmitter* din cadrul clasei *RFTransmitter* are ca și scop declanșarea procesului de inițializare al transmițătorului. Acesta reprezintă o precondiție pentru transferul de date între modulele senzor și modulul de control.

Pinii disponibili pe această componentă electronică pot funcționa fie ca intrări, fie ca ieșiri, însă este responsabilitatea programatorului să seteze modul de funcționare a fiecărui pin în parte. Toate aceste configurații le-am grupat în metoda *pinSetup*, al cărei cod îl voi face disponibil.

```
void WiFiModule::pinSetup() {  
    pinMode(RELAY_PIN, OUTPUT);  
    pinMode(INCREASE_TEMPERATURE_PIN, INPUT);  
    pinMode(DECREASE_TEMPERATURE_PIN, INPUT);  
    pinMode(LED_BUILTIN, OUTPUT);  
}
```

Pentru identificarea pinilor am definit macrouri, cu nume sugestive, iar cuvintele cheie *INPUT* și *OUTPUT* au fost predefinite în bibliotecile utilizate.

Majoritatea funcționalităților îndeplinite de acest modul necesită o conexiune validă la internet. Primul pas în acest sens se face prin apelul metodei *connectToInternet*. În cadrul funcției *WiFi.begin()* trebuie menționat numele rețelei la care se conectează și parola. Modulul este programat în așa fel încât timpul alocat conectării la internet să fie de maximum 15 secunde. Dacă acesta este depășit, este posibil să fie o problemă în ceea ce privește conexiunea la internet, iar modulul va funcționa deconectat de la baza de date.

După ce plăcuța WiFi a reușit să se conecteze la internet, următorul pas este identificarea bazei de date. Apelul metodei *Firebase.begin* necesită doi parametri, și anume: adresa bazei de date și cheia secretă. Aceștia sunt utilizați pentru a localiza baza de date,

respectiv pentru a oferi acces la aceasta.

Setarea temperaturii se realizează prin întreruperi. Trebuie definit pinul și frontul semnalului pe care se declanșează întreruperea, dar și numele rutinei de tratare. Toate acestea se realizează în metoda *defineInterrupts*.

Ultimele apeluri din funcția *setup*, *stream(instance, path)*, au rolul de a seta instanța și câmpul din baza de date pentru care se vor monitoriza schimbările de valoare. Modulul face citiri din baza de date doar când sunt date actualizate, astfel reușind să economisească din resursele disponibile.

Întreruperile și tratarea acestora

În continuare, voi prezenta rutinele de tratare a întreruperilor. Acestea sunt simple, nu consumă multe resurse și, ceea ce este esențial în crearea unor astfel de rutine, execuția lor nu durează mult. De asemenea, utilizez două variabile globale, *increaseDesiredTemperature* și *decreaseDesiredTemperature*, pentru a marca faptul că o procedură de tratare a întreruperii a fost accesată.

```
ICACHE_RAM_ATTR void increaseTemperature() {  
    increaseDesiredTemperature = true;  
}
```

```
ICACHE_RAM_ATTR void decreaseTemperature() {  
    decreaseDesiredTemperature = true;  
}
```

În momentul în care este acționat unul din cele două butoane responsabile cu modificarea temperaturii setate, se declanșează execuția unei serii de instrucțiuni. Acestea sunt consumatoare de resurse, motiv pentru care, se execută în interiorul funcției *loop*.

```
if(increaseDesiredTemperature || decreaseDesiredTemperature){  
    if(increaseDesiredTemperature){  
        if(desiredTemperature < MAX_TEMP){  
            desiredTemperature++;  
            lcd.displayDesiredTemperature();  
            if (WiFi.status() == WL_CONNECTED &&  
                (!switchIntervalsOn))  
            {  
                wifiModule.sendDesiredTemperatureToDatabase  
                    ("DesiredTempRoom2/Zapier/Value");  
            }  
        }  
    }  
}
```

```

    }
  }
  increaseDesiredTemperature = false;
} else {
  if (desiredTemperature > MIN_TEMP) {
    desiredTemperature--;
    lcd.displayDesiredTemperature();
    if (WiFi.status() == WL_CONNECTED &&
        (!switchIntervalsOn))
    {
      wifiModule.sendDesiredTemperatureToDatabase
        ("DesiredTempRoom2/Zapier/Value");
    }
  }
  decreaseDesiredTemperature = false;
}
}

```

Pentru început, se verifică dacă de la ultima iterație a fost modificată variabila *increaseDesiredTemperature* sau *decreaseDesiredTemperature*. În caz afirmativ, se face o incrementare sau decrementare a valorii temperaturii setate. Pentru ca rezultatul să se afișeze pe LCD cât mai repede, după ajustarea valorii, se face apelul funcției *displayDesiredTemperature*, din cadrul clasei LCD. Astfel, se obține un timp mic de actualizare și o experiență mai bună cu utilizatorul. De asemenea, se examinează modul de funcționare a modului wireless, conectat, sau nu, la o rețea de internet. În cazul în care există o conexiune validă, valoarea actualizată a temperaturii va fi trimisă în baza de date. O altă condiție necesară este ca modul de funcționare pe intervale orare să fie dezactivat *!switchIntervalsOn*.

Mod de operare presetat de utilizator

Sistemul este setat astfel încât să ofere posibilitatea utilizatorului de a alege patru intervale orare, cu temperaturi diferite, în timpul zilelor lucrătoare. În ceea ce privește ultimele zile ale săptămânii, sâmbătă și duminică, se pot seta doar două intervale orare pe zi, cu temperaturi diferite.

```

if (WiFi.status() == WL_CONNECTED) {
  wifiModule.sendCurrentTemperatureToDatabase(currentTemperature);
  wifiModule.sendHumidityToDatabase(humidity);
  wifiModule.readStreamValue(switchIntervalsOn,

```

```

        streamSwitchIntervalsOn, "SwitchIntervalsOn/Value");
    if (switchIntervalsOn) {
        int weekDay = timeManager.getWeekDay();
        int currentHour = timeManager.getCurrentHour();
        int currentMinute = timeManager.getCurrentMinute();
        if(wifiModule.checkForUpdate(streamIntervals,
            "/Intervals") || (!timeIntervalsOperatingMode))
        {
            if (weekDay == 6 || weekDay == 0) {
                String A = wifiModule.readStr("Intervals/Weekend/A");
                String B = wifiModule.readStr("Intervals/Weekend/B");

                int hourA = timeManager.getHourFromTimeFormat(A);
                int minuteA = timeManager.getMinuteFromTimeFormat(A);
                int hourB = timeManager.getHourFromTimeFormat(B);
                int minuteB = timeManager.getMinuteFromTimeFormat(B);

                if ((hourA < currentHour && currentHour < hourB) ||
                    (hourA == currentHour && currentMinute >=
                     minuteA) ||
                    (hourB == currentHour && currentMinute <
                     minuteB)) {
                    wifiModule.readDesiredTemperatureFromDatabase
                        ("Intervals/Weekend/TemperatureAB");
                    endHour = hourB;
                    endMinute = minuteB;
                } else {
                    wifiModule.readDesiredTemperatureFromDatabase
                        ("Intervals/Weekend/TemperatureBA");
                    endHour = hourA;
                    endMinute = minuteA;
                }
            }
        }
    }
}

```

Primele două instrucțiuni au rolul de a încărca în baza de date valoarea temperaturii și a umidității. Modul de funcționare a sistemului este salvat într-un câmp al bazei de date, numit *SwitchIntervalsOn/Value*. Acesta este monitorizat prin instalarea unui observator pe câmpul bazei de date.

Instrucțiunile ce urmează au rolul de a încadra sistemul în ora, minutul și ziua curentă, iar cu ajutorul acestora, sistemul poate urmări programul de temperaturi setat de către utilizator. Secvența de cod prezentată anterior este utilizată pentru încadrarea zi-

lelor de la sfârșitul săptămânii, sâmbătă și duminică. În cazul în care nu se respectă condiția, se execută secvențele condiționale următoare, împreună cu citirile capetelor intervalelor și valorile temperaturilor corespunzătoare.

```
{
    String A = wifiModule.readStr("Intervals/WorkingDay/A");
    String B = wifiModule.readStr("Intervals/WorkingDay/B");
    String C = wifiModule.readStr("Intervals/WorkingDay/C");
    String D = wifiModule.readStr("Intervals/WorkingDay/D");

    int hourA = timeManager.getHourFromTimeFormat(A);
    int minuteA = timeManager.getMinuteFromTimeFormat(A);
    int hourB = timeManager.getHourFromTimeFormat(B);
    int minuteB = timeManager.getMinuteFromTimeFormat(B);
    int hourC = timeManager.getHourFromTimeFormat(C);
    int minuteC = timeManager.getMinuteFromTimeFormat(C);
    int hourD = timeManager.getHourFromTimeFormat(D);
    int minuteD = timeManager.getMinuteFromTimeFormat(D);

    if ((hourA < currentHour && currentHour < hourB) || (hourA
        == hourB && currentHour == hourA && currentMinute >=
        minuteA && currentMinute < minuteB) || (hourA ==
        currentHour && hourB != currentHour && currentMinute >=
        minuteA) || (hourA != currentHour && hourB ==
        currentHour && currentMinute < minuteB)) {
        wifiModule.readDesiredTemperatureFromDatabase
            ("Intervals/WorkingDay/TemperatureAB");
        endHour = hourB;
        endMinute = minuteB;
    } else if ((hourB < currentHour && currentHour < hourC) ||
        (hourB == hourC && currentHour == hourB &&
        currentMinute >= minuteB && currentMinute < minuteC) ||
        (hourB == currentHour && hourC != currentHour &&
        currentMinute >= minuteB) || (hourB != currentHour &&
        hourC == currentHour && currentMinute < minuteC)){
        wifiModule.readDesiredTemperatureFromDatabase
            ("Intervals/WorkingDay/TemperatureBC");
        endHour = hourC;
        endMinute = minuteC;
    } else if ((hourC < currentHour && currentHour < hourD) ||
        (hourC == hourD && currentHour == hourC &&
```

```

        currentMinute >= minuteC && currentMinute < minuteD) ||
        (hourC == currentHour && hourD != currentHour &&
        currentMinute >= minuteC) || (hourC != currentHour &&
        hourD == currentHour && currentMinute < minuteD)) {
            wifiModule.readDesiredTemperatureFromDatabase
                ("Intervals/WorkingDay/TemperatureCD");
            endHour = hourD;
            endMinute = minuteD;
        } else {
            wifiModule.readDesiredTemperatureFromDatabase
                ("Intervals/WorkingDay/TemperatureDA");
            endHour = hourA;
            endMinute = minuteA;
        }
    }
}

```

Mod clasic de operare

Pe lângă modul automat de funcționare, sistemul este proiectat să poată rula și în mod manual. Acesta presupune menținerea unei temperaturi constante, setate de utilizator, prin butoane, comenzi vocale sau prin intermediul aplicației web. Este important de menționat faptul că acesta consumă mai puține resurse comparativ cu modul de funcționare pe intervale, însă nu oferă același grad de confort. În continuare, voi prezenta o serie de instrucțiuni de cod specifice implementării modului manual.

```

int currentTemperature = dhtSensor.readTemp();
int humidity = dhtSensor.readHumidity();

```

Prima etapă ce se execută în acest mod este citirea umidității și a temperaturii de la senzorul DHT11. Iar funcțiile responsabile pentru aceste funcționalități sunt apelate la începutul funcției *loop*.

```

wifiModule.heatControl(currentTemperature);
wifiModule.statusIndicator();
lcd.displayDesiredTemperature();
lcd.displayCurrentTemperature(currentTemperature);

```

Pentru a face posibilă pornirea/oprirea sistemului de încălzire, este necesar apelul funcției *heatControl*. Acesta trimite comandă atât la electrovalva montată pe calorifer, cât și la centrala termică. De asemenea, este necesar apelul *displayDesiredTemperature* și *displayCurrentTemperature* pentru a afișa temperatura setată și temperatura dorită din cameră. Scopul final este de a permite monitorizarea temperaturii ambientale.

4.4.2 Programare plăcuță Arduino

Este montată în modulul de control și se conectează la componente auxiliare precum: receptor RF și releu, iar prin intermediul acestora face posibil controlul centralei termice. Codul sursă care face posibil tot acest comportament se află pe plăcuța Arduino.

```
unsigned char dataPackage[3];
uint8_t dataPackageLength = sizeof(dataPackage);
if (driver.recv(dataPackage, &dataPackageLength))
{
    dataPackage[2] = '\\0';
    Serial.println((char *) dataPackage);
    int dataPackageInt = atoi((char *) dataPackage);

    if((dataPackageInt/10) == 1){
        commandModule1 = dataPackageInt%10;
        timeFirstModuleSent = millis()/1000;
    }
    else{
        commandModule2 = dataPackageInt%10;
        timeSecondModuleSent = millis()/1000;
    }

    Serial.print("Module1:");
    Serial.println(commandModule1);
    Serial.print("Module2:");
    Serial.println(commandModule2);
    if(commandModule1 || commandModule2){
        digitalWrite(7, LOW);
        Serial.println("On");
    }else{
        digitalWrite(7, HIGH);
        Serial.println("Off");
    }
}
```

```
if(((millis()/1000) - timeFirstModuleSent) >= TIMEOUT &&
    ((millis()/1000) - timeSecondModuleSent) >= TIMEOUT))
{
    digitalWrite(7, HIGH);
    commandModule1 = 0;
    commandModule2 = 0;
}
else if(((millis()/1000) - timeFirstModuleSent) >= TIMEOUT
    && commandModule2 == 0)
{
    digitalWrite(7, HIGH);
    commandModule1 = 0;
}
else if(((millis()/1000) - timeSecondModuleSent) >= TIMEOUT
    && commandModule1 == 0) {
    digitalWrite(7, HIGH);
    commandModule2 = 0;
}
```

Mesajul pe care modulul receptor îl interceptează este format din doi octeți, din care octetul cel mai semnificativ conține numărul de identificare al modulului care a trimis mesajul, iar octetul cel mai puțin semnificativ deține informații legate de natura comenzii. Mesajul recepționat este convertit în întreg datorită faptului că se poate prelucra mai ușor aplicând operații, precum: modulo și div.

Întotdeauna se salvează momentul în care s-au recepționat informații de la fiecare modul în parte. Este importantă menținerea evidenței, pentru a evita situațiile în care apar întreruperi de transfer prin radio - frecvență. Ca și o măsură de prevenție, dacă modulul de control nu primește comenzi de la modulele senzor un timp mai mare sau egal cu 5 secunde, centrala termică va fi oprită.

După ce au fost primite comenzi de la ambele module senzor, se execută operație logică de tip SAU, iar în funcție de rezultatul acesteia, va cupla sau decupla releul, ce în continuare controlează centrala termică. Un octet cu valoarea 1 indică pornirea, iar un octet cu valoarea 0 indică oprirea centralei din imobil. De asemenea, ultimele instrucțiuni condiționale din secvența de cod afișată anterior, au rolul de a implementa oprirea de siguranță a sistemului de încălzire.

Inițial, se verifică dacă ambele module senzor nu au trimis comenzi de mai mult de 5 secunde. Dacă nu se respectă prima condiție, se examinează cazul în care primul modul are conexiunea întreruptă, iar cel de-al doilea trimite comandă de oprire a sistemului de încălzire. Iar ultima situație care poate apărea este cea în care conexiunea prin radio frecvență la al doilea modul este inactivă de peste 5 secunde, iar primul modul trimite

comandă de întrerupere a funcționării centralei. Toate aceste situații apar în cazuri de avarie, iar principala cauză este comunicația precară între modulele senzor și modulul de control. Pentru a preveni defectarea centralei termice, este concepută o măsură de siguranță ce constă în oprirea acesteia.

4.4.3 Realizarea aplicației web

Aplicația web face transferul de date cu partea electronică prin intermediul bazei de date *Firebase*. Valorile umidității și a temperaturii se actualizează în aplicația web doar atunci când apar modificări în baza de date. Astfel, se evită citirile repetate și consumatoare de resurse.

În ceea ce privește partea de acces a aplicației, aceasta este găzduită pe un server, la distanță, prin intermediul platformei *Heroku*. Oricine deține adresa aplicației o poate accesa, însă prima pagină care apare la accesare este cea de conectare, iar navigarea poate fi continuată doar în cazul în care utilizatorul deține un cont valid.

Secvențe de cod esențiale din cadrul implementării aplicației web

Pentru a asigura accesul la aplicație doar a persoanelor privilegiate, trebuie creată o pagină de conectare ce să permită verificarea concordanței credențialelor introduse în formular cu cele salvate în baza de date. În cazul unei potriviri, utilizatorul este redirectionat la calea */home*, aceasta aparținând paginii principale și permite monitorizarea parametrilor ambientali și setarea temperaturii dorite. Situația în care se introduc credențiale greșite este tratată printr-un mesaj de avertizare, iar utilizatorului i se oferă posibilitatea reintroducerii datelor de conectare.

Inițial, se preiau informațiile din interfață și se verifică existența utilizatorului cu numele introdus, instrucțiunea ce realizează acest aspect este: `user_to_login = Users.query.filter_by(username=username).first()`. Dacă numele utilizatorului a fost găsit, următorul pas ce trebuie inspectat este potrivirea parolei introduse cu cea salvată în baza de date. Este esențial de luat în considerare faptul că parolele sunt salvate criptate în baza de date, iar înainte de a face comparația cu parola introdusă, este obligatorie și criptarea parolei tastate de utilizator în formularul de conectare.

Logica din spatele criptării și verificării ulterioare a parolei o voi prezenta în rândurile ce urmează.

```
if user_to_login:
    if bcrypt.check_password_hash(user_to_login.password,
                                   password):
```

```
login_user(user_to_login)
return redirect(url_for('home'))
```

Funcția `bcrypt.check_password_hash` criptează parola din formular și face și o comparație cu parola din baza de date. În cazul în care corespondența este verificată, utilizatorul devine înregistrat în aplicația web și dobândește drepturi de acces.

Prin intermediul paginii *home* din cadrul aplicației web, utilizatorul are posibilitatea să salveze în baza de date diverse valori ale temperaturilor ce urmează a fi menținute în imobil. În continuare, voi prezenta funcția care se apelează în această situație. Este implementată în *flask* și se execută în momentul în care se ajunge la calea `/home`.

```
@app.route('/home', methods=['POST', 'GET'])
@login_required
def home():
    status = firebase.get('SwitchIntervalsOn', 'Value')
    if request.method == 'POST':
        variable = request.form.get('outputValue1')
        if variable is not None:
            firebase.put('DesiredTempRoom1/Zapier', 'Value',
                        int(variable))
        else:
            variable = request.form.get('outputValue2')
            firebase.put('DesiredTempRoom2/Zapier', 'Value',
                        int(variable))
    return render_template('controlPage.html', status=status)
```

Se utilizează la setarea temperaturii în baza de date, dar și la citirea câmpului *SwitchIntervalsOn/Value*, ce conține informații referitoare la modul de programare al temperaturilor dorite.

Reprezintă o legătură între partea de interfață și logica din spatele aplicației web. Cu alte cuvinte, valoarea introdusă de către utilizator prin intermediul elementelor de intrare din *html*, este transferată în partea responsabilă cu procesarea informațiilor. Prin intermediul funcțiilor puse la dispoziție de un modul implementat în *python*, valorile introduse se transmit mai departe în baza de date.

Apelul `render_template` este util pentru a prelucra paginile *html*. Acestea pot conține diverse variabile trimise ca și parametri și pot executa structuri de control, toate acestea pentru a facilita felul în care se construiește interfața cu utilizatorul. Variabila

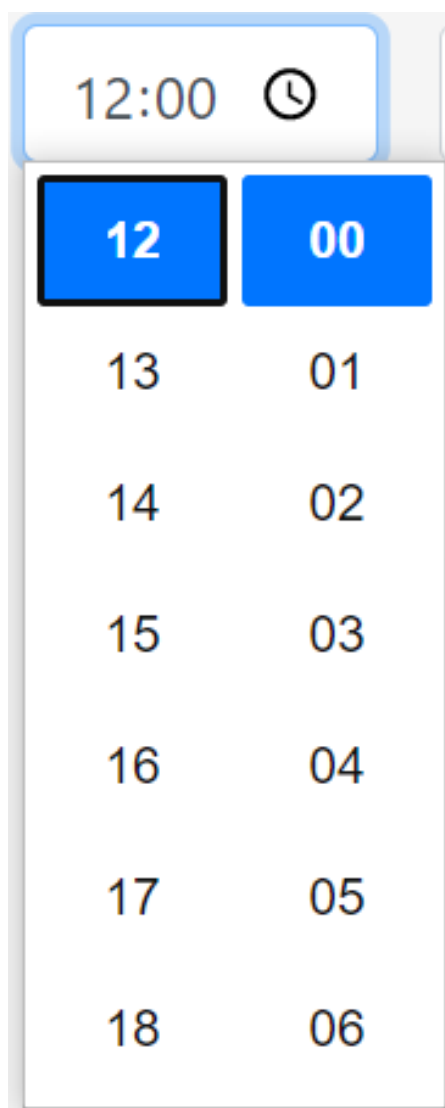
status este transmisă mai departe paginii intitulată *controlPage* și este utilizată pentru a seta poziția butonului ce indică modul de funcționare al sistemului.

În cele ce urmează, voi ilustra modul în care se face o programare a temperaturilor pentru mai multe zile. Voi prezenta traseul pe care informațiile îl parcurg, de la introducerea acestora de către utilizator, până la prelucrarea și salvarea lor în baza de date.

Voi începe prin a expune codul ce face posibilă crearea formularelor utilizate pentru introducerea valorilor temperaturilor și a intervalelor orare. Acestea sunt esențiale pentru interacțiunea între sistem și utilizator.

```
<div class="form-group">
  <input type="time" class="form-control"
    name="firstWorkingDayInterval"
    id="firstWorkingDayInterval" required>
</div>
<div class="form-group">
  <input type="number" class="form-control"
    name="temperatureFirstWDInterval"
    id="temperatureFirstWDInterval"
    placeholder="Temperature 1" required>
</div>
```

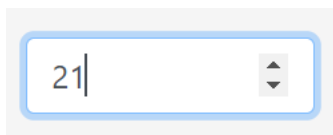
Primul tip de intrare *html* utilizat este cel pentru formatul timp. Este un câmp predefinit, împărțit în două secțiuni, una pentru alegerea orei, iar cealaltă pentru selectarea minutului.



The image shows a time selection interface. At the top, a white box with a blue border contains the text '12:00' and a clock icon. Below this, there are two columns of blue buttons. The first column contains buttons for hours: '12', '13', '14', '15', '16', '17', and '18'. The second column contains buttons for minutes: '00', '01', '02', '03', '04', '05', and '06'. The '12' button is highlighted with a black border.

Figura 4.10: Intrare HTML pentru formatul timp

Cel de-al doilea tip de intrare prezentat, cel de tip numeric, este utilizat pentru setarea valorii temperaturii. Formatul intrării este creat în așa natură încât să permită un reglaj al valorii, la nivel de unitate.



The image shows a numeric input field. It consists of a white box with a blue border containing the number '21' and a vertical cursor. To the right of the box is a small grey button with a double-headed vertical arrow.

Figura 4.11: Intrare HTML pentru formatul numeric

După ce utilizatorul completează toate câmpurile, aceste valori urmează să fie transmise mai departe părții, din cadrul aplicației web, responsabile cu procesarea informațiilor. Transferul datelor se face prin metode de tip *POST*.

Înainte de salvarea informațiilor în baza de date, este necesară o verificare a acestora. Valorile temperaturilor trebuie să se încadreze în intervalul 15 °C și 32 °C, iar orice depășire a acestor limite nu poate fi stocată. De asemenea, ordinea intervalelor orare trebuie să fie setată într-un mod cronologic. În cazul în care aceste condiții nu sunt respectate, se va afișa un mesaj de eroare adaptat contextului, iar utilizatorului i se oferă posibilitatea reintroducerii datelor.

```
MIN = 15
```

```
MAX = 32
```

```
timeObjectA = getTime(a)
```

```
timeObjectB = getTime(b)
```

```
timeObjectC = getTime(c)
```

```
timeObjectD = getTime(d)
```

```
if int(temperatureAB) < MIN or int(temperatureAB) > MAX or
    int(temperatureBC) < MIN or int(temperatureBC) > MAX or
    int(temperatureCD) < MIN or int(temperatureCD) > MAX or
    int(temperatureDA) < MIN or int(temperatureDA) > MAX:
    flash('The values of temperatures should be between
        15 and 32 degrees!', 'warning')
```

```
else:
```

```
    if timeObjectA < timeObjectB < timeObjectC < timeObjectD:
```

```
        try:
```

```
            firebase.put('Intervals/WorkingDay', 'A', a)
```

```
            firebase.put('Intervals/WorkingDay', 'B', b)
```

```
            firebase.put('Intervals/WorkingDay', 'C', c)
```

```
            firebase.put('Intervals/WorkingDay', 'D', d)
```

```
            firebase.put('Intervals/WorkingDay',
```

```
                        'TemperatureAB', int(temperatureAB))
```

```
            firebase.put('Intervals/WorkingDay',
```

```
                        'TemperatureBC', int(temperatureBC))
```

```
            firebase.put('Intervals/WorkingDay',
```

```
                        'TemperatureCD', int(temperatureCD))
```

```
            firebase.put('Intervals/WorkingDay',
```

```
                        'TemperatureDA', int(temperatureDA))
```

```
        except Exception as err:
```

```
            flash('An error occurred while setting intervals:
```

```

        {0}'.format(err), 'warning')
    else:
        flash('Time intervals must be set chronologically!',
              'warning')
    return render_template('schedulingPage.html')

```

Prima instrucțiune condițională utilizată în codul prezentat anterior, are rolul de a determina dacă valorile temperaturilor introduse se încadrează în intervalul acceptat de sistem și de aplicația web.

Pe de altă parte, pentru a putea stabili dacă există o ordine cronologică, se apelează funcția *getTime* cu datele pe care le citim de la intrarea pentru formatul de tip timp. Se creează obiecte ce conțin data, ora și minutul curent pentru fiecare interval în parte. Datorită faptului că aceste obiecte sunt predefinite în *python*, se pot aplica operații de comparație asupra lor.

Implementarea funcției de încapsulare a valorilor intervalelor o voi prezenta în rândurile ce urmează, împreună cu explicațiile de cod aferente.

```

def getTime(timeFormat):
    hour = getHourFromTimeFormat(timeFormat)
    minute = getMinuteFromTimeFormat(timeFormat)

    now = datetime.now(pytz.timezone('Europe/Bucharest'))
    dateTimeObject = datetime(now.year, now.month, now.day,
                               hour, minute)

    return dateTimeObject

```

Apelul *getHourFromTimeFormat* și *getMinuteFromTimeFormat* sunt utilizate pentru a despărți șirul de caractere ce conține ora și minutul setat de către utilizator. Obiectele predefinite în *python* pentru stocarea datei și timpului prezintă ca și câmpuri obligatorii: anul, luna, ziua, ora și minutul. Ca și intrare de la utilizator primim doar ora și minutul, iar pentru crearea obiectului, restul valorilor ne sunt puse la dispoziție prin intermediul unui modul *python*, ce are setat fusul orar actual. În urma apelului funcției *datetime* sunt returnate obiecte ce conțin data curentă, ora și minutul capetelor intervalelor introduse de utilizator. Pe aceste obiecte se vor executa operații de comparație pentru a determina dacă ordinea este una cronologică.

Intr-un final, după ce datele au fost validate, urmează salvarea acestora în baza de date. Pentru a implementa acest aspect, se utilizează suita de apeluri *firebase.put*, care

primesc ca și parametrii: numele câmpului unde urmează să se facă stocarea, împreună cu informațiile aferente.

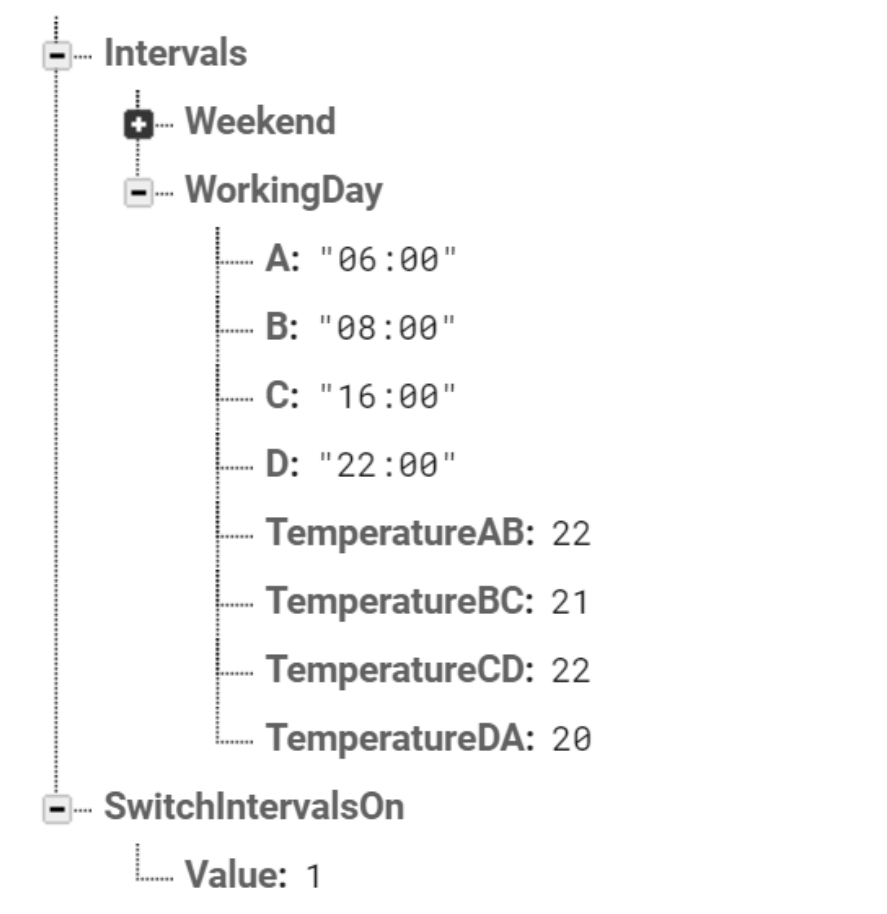


Figura 4.12: Stocare program intervale orare

4.4.4 Probleme întâmpinate în timpul realizării proiectului

Forța contraelectromotoare a bobinei

Electrovalvele sunt formate dintr-un solenoid. Aplicarea tensiunii la bornele acestuia duce la generarea unui câmp electromagnetic, care la rândul său acționează un obturator. Astfel, se realizează închiderea/deschiderea circuitului de apă. Este important de menționat faptul că solenoidul înmagazinează curent, iar întreruperea alimentării electrice a acestuia produce o descărcare, ce se manifestă prin apariția unor fluctuații de tensiune în circuit. Prezența acestor fluctuații poate duce la arderea celorlalte componente electronice, dar și la o uzură accentuată a releelor de comandă. De asemenea,

descărcarea solenoidului produce apariția unor scântei între contactele releului, determinând perturbații în întreg sistemul. Un exemplu de disfuncționalitate cu care m-am confruntat din aceeași cauză, a fost dereglarea temperaturii setate, în momentul în care modulul ESP8266 trimitea comandă de închidere a electrovalvei.

Vibrațiile contactelor metalice ale întrerupătoarelor

Pentru a putea modifica valoarea temperaturii setate a fi menținute în imobil, se utilizează butoane fizice, legate prin conductori electrici la modulele WiFi. Din punct de vedere funcțional, acestea pot avea comportament neașteptat, cauzat de vibrațiile fine care se produc între contactele metalice ale întrerupătorului. Frecvența cu care microcontrolerul citește semnalul provenit de la buton este suficient de mare, astfel încât, să poată sesiza inclusiv o vibrație ca fiind o apăsare distinctă. Prin urmare, se întâmplă ocazional ca intenția utilizatorului de a incrementa sau decrementa cu o unitate temperatura setată, să se traducă în modificarea acesteia cu mai multe unități. Soluția pe care am implementat-o în această situație, a fost realizarea și montarea unor filtre trece jos pentru fiecare buton în parte. Rolul acestora este de a permite trecerea doar a semnalelor cu frecvențe mici, iar cele de frecvențe mari, perturbațiile, fiind oprite. Capturile de ecran din cadrul capitolului 3, prezintă semnalul înainte de filtrare 3.1, dar și după aceasta 3.2.

Amorsarea instalației cu apă

Pompa de apă pe care am utilizat-o în circuit, nu prezintă funcționalitate de autoamorsare. Prin urmare, prima tentativă de umplere a circuitului cu apă, a constat în desfacerea unei îmbinări a circuitului și introducerea apei cu o seringă prin aceasta. Din cauza faptului că instalația prezintă multe coturi, iar introducerea apei nu o faceam din punctul cel mai înalt, procentajul de umplere al circuitului nu era satisfăcător. Ca și disfuncționalitate, se manifesta prin faptul că pompa de apă de dezamorsa în timpul funcționării. Într-un final, am decis să montez un aerisitor în punctul cel mai înalt al instalației, iar introducerea apei să se facă prin acesta. Astfel, volumul de aer din circuit a fost redus considerabil, ducând la eliminarea situațiilor de dezamorsare a pompei.

Conflict de date apărut la transferul prin radio - frecvență

Ambele transmițătoare utilizate în cadrul machetei funcționează la aceeași frecvență, 433 Mhz. Transferul datelor se face către un singur receptor, montat pe modulul de comandă al centralei termice. Din punct de vedere funcțional, apăreau probleme în momentul în care ambele transmițătoare făceau transfer simultan. Datele nu mai ajungeau să fie recepționate de către receptor, ajungând la pierderea comenzilor. Modulele wireless sunt conectate la un server prin intermediul căruia citesc data și ora curentă.

Utilizând acest aspect, am programat modulele astfel încât să facă transferul de date concurrent. Un modul va transfera la secundă pară, pe când celălalt la secundă impară.

4.5 Testarea soluției

În orice proces ce implică dezvoltare de proiecte de o complexitate mai mare, etapa de testare este una esențială. Aceasta presupune identificarea erorilor sistemului, înainte de a fi pus în producție. Cazurile de test trebuie proiectate astfel încât să cuprindă toate situațiile care pot produce abateri de la comportamentul standard.

În ceea ce privește proiectul prezentat, testarea a vizat două mari componente: aplicația web și instalația pentru circuitul de apă. *Flask* oferă posibilitatea de creare de teste unitare pentru proiectele realizate prin intermediul lui. Pe lângă aceasta, permite și analiza procentului de acoperire al întregii aplicații prin testele scrise.

Voi începe prin a prezenta partea de validare a aplicației web. Am realizat o suită de 50 de teste, pe care am împărțit-o în 3 mari categorii. Prima categorie conține cazurile de test comune pentru toți utilizatorii, funcționalități care sunt valabile indiferent de rolul persoanei autentificate aplicației. Am încercat să abordez o manieră exhaustivă, prin simularea tuturor situațiilor de eroare, dar și de reușită. Printre cazurile mai importante de test care se încadrează în această categorie sunt:

- Verificarea faptului că pentru a putea realiza orice funcționalitate din aplicația web, trebuie să existe un utilizator conectat. Altfel, accesul este obligatoriu să fie interzis.
- Încercarea de conectare utilizând un cont inexistent, trebuie interzisă și avertizată printr-un mesaj de eroare.
- Verificarea funcției de criptare a parolei utilizatorului.
- Verificarea posibilității de conectare la aplicația web.
- Introducerea parolei greșite în momentul conectării.
- Testarea funcționalității de deconectare.
- Activare/Dezactivare mod de lucru cu intervale orare
- Controlul posibilității de setare a intervalelor pentru zile lucrătoare, dar și pentru cele de la sfârșitul săptămânii. De asemenea, cazurile acestea de test se împart în mai multe. Situația în care acestea nu sunt setate în ordine cronologică și situația în care vreo valoare a temperaturii nu se încadrează în intervalul 15 °C - 32 °C.
- Funcția de schimbare a parolei, implică verificarea situațiilor în care: parola curentă nu este validă, parolele introduse nu se potrivesc, noua parolă este prea

scurtă, noua parolă nu conține majusculă și situația în care noua parolă nu conține cifră.

- Verificarea funcției de setare a temperaturii pentru ambele camere.

Cea de-a doua categorie este dedicată testelor unitare pentru utilizatorii care nu au drepturi de administrator. În principiu, aici se verifică faptul că doar administratorii au acces la funcționalitățile privilegiate. Sunt validate cazurile de test precum:

- Un cont fără drept de administrator nu este autorizat pentru crearea de alte conturi.
- Nu este permis accesul pentru a vedea lista de conturi, a șterge un cont, a oferi drepturi de administrator sau a șterge drepturi de administrator.

În cele din urmă, ultima categorie cuprinde testarea doar a funcționalităților dedicate administratorilor. Se verifică:

- Posibilitatea de creare de conturi noi.
- Interzicerea accesului de a crea un cont nou dacă: parolele nu se potrivesc, parola introdusă este prea scurtă, nu conține majusculă sau nu conține cel puțin o cifră.
- Ștergerea unui cont.
- Incercarea de ștergere a unui cont inexistent.
- Acordarea drepturilor de administrator unui anumit cont.
- Acordarea drepturilor de administrator unui cont inexistent.
- Retragera drepturilor de administrator.
- Retragera drepturilor de administrator pentru un cont inexistent.

Este important ca prin testele create să acoperim un procent cât mai mare din întreaga aplicație. Cu toate acestea, trebuie menționat faptul că, procentul de acoperire nu este un indicator al calității testelor, ci doar reprezintă cât de multe funcționalități sunt acoperite prin cazurile de test.

Coverage report: 98%

Module ↑	statements	missing	excluded	coverage
app.py	213	11	0	95%
functions\convert.py	8	1	0	88%
functions\passwordValidation.py	9	0	0	100%
functions\time.py	14	0	0	100%
functions\wrappers.py	9	0	0	100%
unitTest.py	281	0	0	100%
Total	534	12	0	98%

coverage.py v5.4, created at 2021-04-22 14:48 +0300

Figura 4.13: Procent de acoperire

Partea de verificare a instalației de apă a fost una mai simplă. După finalizarea lipiturilor conductelor de cupru și montarea garniturilor la fiecare îmbinare, a fost necesară o verificare a faptului că etanșeitatea este asigurată. Procesul a constat în introducerea unei presiuni de 2 bari, pe care am menținut-o în circuit un timp de 10 minute. În toată această perioadă, am monitorizat, cu ajutorul manometrului pompei de aer, faptul că presiunea rămâne constantă. Pentru a mă asigura că pompa de apă și electrovalvele funcționează corespunzător, am folosit o sursă de tensiune de 12 volți și am alimentat fiecare componentă electrică în parte.

Capitolul 5

Ghidul utilizatorului

În acest capitol voi descrie felul în care utilizatorul poate să interacționeze cu aplicația web, iar în cele din urmă voi prezenta una dintre optimizările esențiale aduse sistemului.

5.1 Interfața cu utilizatorul

Toate paginile au aceeași structură: antet, meniu de navigare și secțiune pentru afișarea conținutului.

Antet-ul conține o imagine și un logo.

Bara de navigare se adaptează în funcție de rolul utilizatorului. Unele butoane devin, sau nu, disponibile.

Secțiunea de conținut, aceasta este diferită la fiecare pagină în parte.

Pagina *Home* – este accesibilă pentru toți utilizatorii și permite monitorizarea și setarea temperaturii.

Pagina *Change password* – permite modificarea parolei pentru fiecare utilizator

Capitolul 6

Concluzii și direcții de dezvoltare

Intenția pe care am avut-o în momentul în care am început să lucrez la acest proiect, a fost să creez un sistem IoT, care să eficientizeze felul în care energia termică poate fi gestionată în cadrul imobilului. Funcționalitatea de bază a sistemului este aceea de a permite setarea temperaturilor diferite pentru fiecare cameră în parte. Comparativ cu celelalte sisteme existente pe piață, acest aspect reprezintă un plus. De asemenea, sunt puse la dispoziție o serie de modalități prin care se pot seta temperaturile din cadrul imobilului:

- Metoda standard, prin intermediul butoanelor fizice atașate de modulul WiFi.
- Prin utilizarea aplicației web.
- Utilizând comenzi vocale, interpretate de Google Assistant.

Am reușit să implemetez majoritatea funcționalităților propuse, însă modul de operare al unora poate fi îmbunătățit. Un exemplu sugestiv în acest sens este partea de control prin comenzi vocale a sistemului. Prin intermediul acestora se poate seta o anumită temperatură, însă nu se poate ajusta temperatura setată prin increment sau decrement de un °C. Setul de comenzi nu este unul complex, iar timpul de răspuns al sistemului este destul de mare.

În ceea ce privește partea de dezvoltare ulterioară a proiectului, țin să menționez că sunt o serie de aspecte a căror implementare poate crește gradul de utilitate și eficiența sistemului.

În varianta actuală, întârzierea între momentul în care se trimite o comandă vocală și momentul în care se primește un răspuns, ajunge să fie de aproximativ un minut. Acest timp poate fi redus dacă se utilizează servicii puse la dispoziție de Google, contracost, ce interpretează comenzile primite de Google Assistant și trimite informațiile interpretate în Firebase.

Pentru a putea modifica datele de conectare la rețea ale modulului wireless, trebuie să se modifice credențialele în codul sursă. O îmbunătățire ar consta în adăugarea unei tastaturi și posibilitatea de a modifica credențialele prin intermediul acesteia.

O altă funcționalitate ce poate fi adăugată este posibilitatea detectării telefoanelor mobile ale locatarilor, iar temperatura din imobil să se adapteze în funcție de prezența sau absența acestora.

Listă de figuri

1.1	Evoluția numărului de dispozitive IoT (sursa: [6])	4
3.1	Semnal înainte de filtrul trece jos	17
3.2	Semnal după filtrul trece jos	17
3.3	Schemă filtru trece-jos (sursa: [20])	18
4.1	Cazurile de utilizare pentru aplicația web	21
4.2	Cazurile de utilizare pentru ansamblul de componente electronice	23
4.3	Arhitectura sistemului	24
4.4	Arhitectura modulului senzor	25
4.5	Arhitectura modulului de control	28
4.6	Diagramă de clase	29
4.7	Mod de operare fără conexiune la internet	30
4.8	Mod de operare pe intervale orare	31
4.9	Mod operare clasic	33
4.10	Intrare HTML pentru formatul timp	47
4.11	Intrare HTML pentru formatul numeric	47
4.12	Stocare program intervale orare	50
4.13	Procent de acoperire	54

Listă de tabele

Bibliografie

- [1] Stephen Ornes. Core concept: The internet of things and the explosion of inter-connectivity. *Proceedings of the National Academy of Sciences*, 113(40):11059–11060, 2016.
- [2] simoniot. <https://www.simoniot.com/history-of-iot/>. [Data accesării: 01.03.2021].
- [3] Cristiano André da Costa, Cristian F. Pasluosta, Björn Eskofier, Denise Bandeira da Silva, and Rodrigo da Rosa Righi. Internet of health things: Toward intelligent vital signs monitoring in hospital wards. *Artificial Intelligence in Medicine*, 89:61–69, 2018.
- [4] L. Yushi, J. Fei, and Y. Hui. Study on application modes of military internet of things (miot). In *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, volume 3, pages 630–634, 2012.
- [5] K. Abboud, H. A. Omar, and W. Zhuang. Interworking of dsrc and cellular network technologies for v2x communications: A survey. *IEEE Transactions on Vehicular Technology*, 65(12):9457–9470, 2016.
- [6] Arne Holst. <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/>. [Data accesării: 01.03.2021].
- [7] Emag. <https://www.emag.ro/termostat-inteligent-universal-pentru-centrala-gaz-wifi-comandat-de-pe-smartphone-internet-compatibil-google-home-si-alexator2000gclw/pd/D0BN1LBBM/>. [Data accesării: 03.03.2021].
- [8] Emag. <https://www.emag.ro/termostat-smart-honeywell-lyric-t6r-wireless-amplasat-pe-masa-comandat-de-pe-smartphone-y6h910rw4055/pd/DP2V6SBBM/>. [Data accesării: 03.03.2021].
- [9] Amazon. <https://www.amazon.com/Nest-T3007ES-Thermostat-Temperature-Generation/dp/B0131RG6VK/>. [Data accesării: 03.03.2021].
- [10] Amazon. <https://www.amazon.com/ecobee3-lite-Smart-Thermostat-Black/dp/B06W56TBLN/>. [Data accesării: 03.03.2021].

- [11] Eric Blank. <https://thesmartcave.com/ecobee-vs-nest/>. [Data accesării: 03.03.2021].
- [12] Guido Van Rossum et al. Python. https://courses.minia.edu.eg/Attach/16028python_lecture1.pdf, 1991. [Data accesării: 15.03.2021].
- [13] Miguel Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.
- [14] Bjarne Stroustrup. *The C++ programming language*. Pearson Education India, 2000.
- [15] Mohamed Fezari and Ali Al Dahoud. Integrated development environment "ide" for arduino. *WSN applications*, pages 1–12, 2018.
- [16] Espressif Systems. https://components101.com/asset/sites/default/files/component_datasheet/ESP8266-NodeMCU-Datasheet.pdf. [Data accesării: 21.03.2021].
- [17] Mircea Popa. Sisteme Încorporate. Curs Anul 3 CTI, Curs 9.
- [18] Handson Technology. http://www.handsontec.com/dataspecs/module/I2C_1602_LCD.pdf. [Data accesării: 20.03.2021].
- [19] Mantech. http://www.mantech.co.za/Datasheets/Products/433Mhz_RF-TX&RX.pdf. [Data accesării: 21.03.2021].
- [20] Jack G Ganssle. A guide to debouncing. *Guide to Debouncing, Ganssle Group, Baltimore, MD, US*, pages 1–22, 2004.