



Universitatea Politehnica 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

<b>1</b>	<b>Introducere</b>	<b>3</b>
1.1	Internet of things . . . . .	3
1.2	Contextul de realizare . . . . .	5
<b>2</b>	<b>Prezentarea sistemelor similare</b>	<b>6</b>
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
<b>3</b>	<b>Fundamente teoretice</b>	<b>11</b>
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	Invertor Schmitt-Trigger . . . . .	15

3.5.14	Diodă . . . . .	15
3.6	Filtrarea semnalelor . . . . .	16
3.7	Histereza . . . . .	17
<b>4</b>	<b>Dezvoltarea soluției</b>	<b>19</b>
4.1	Specificarea cerințelor . . . . .	19
4.2	Arhitectura sistemului . . . . .	20
4.2.1	Modulul senzor . . . . .	21
4.2.2	Modulul de control . . . . .	23
4.3	Proiectare detaliată . . . . .	24
4.4	Implementarea soluției . . . . .	24
4.4.1	Programare modul ESP8266 . . . . .	24
4.4.2	Programare plăcuță Arduino . . . . .	32
4.4.3	Realizarea aplicației web . . . . .	34

# **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 cotidiană. Aceasta 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 unor acțiuni.

Contraște așteptărilor, ideea de Internet of Things nu a apărut recent. Primul dispozitiv care implementă 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 temperatură băuturilor [?]. În anul 1990, inventia lui John Romkey, toaster-ul controlat prin internet [?], 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 [?].

Chiar dacă dispozitivele care au marcat debutul IoT nu oferă funcționalități mult prea utile, în ultimii ani s-a dovedit că ideea de control de la distanță poate fi folosită 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 [?].

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

Industria transporturilor - în acest domeniu, IoT poate fi ilustrat prin transferul de date între un autovehicul și infrastructura, 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 [?].

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 exponentială a numărului de dispozitive interconectate. Conform unei statistică publicate de către Arne Holst, conducătorul echipei de cercetare din cadrul companiei Statista, se apropimează ca în 2030 să se ajungă la un număr mai mare de 25.4 miliarde de dispozitive IoT [?].

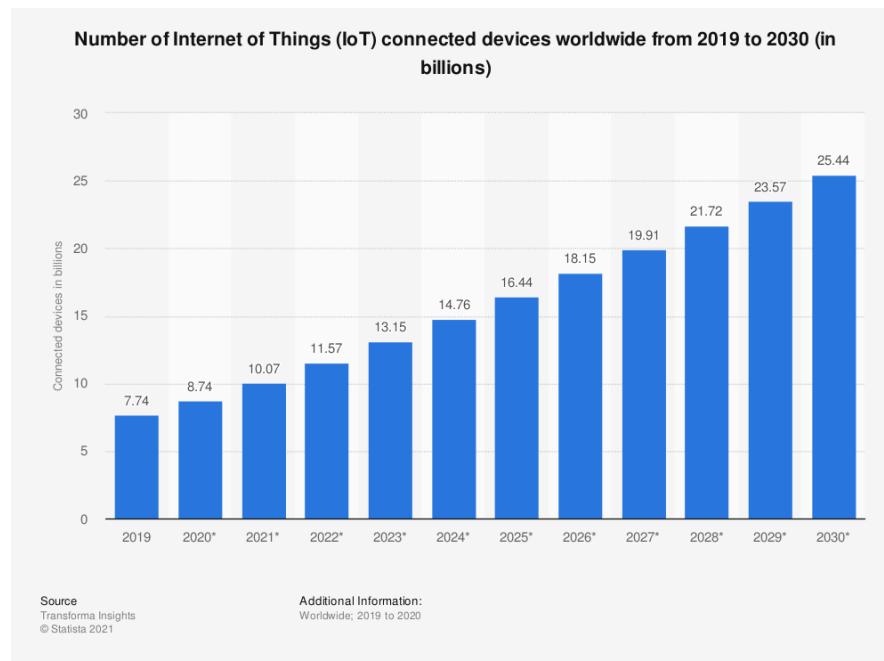


Figura 1.1: Evoluția numărului de dispozitive IoT  
[<https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/>]

## 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 incerc să o rezolv, prin intermediul proiectului, am sesizat-o la apartamentul în care locuiesc. Aceasta prezintă două camere de dimensiuni diferite, iar iarna ne confruntam 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 reprezinta 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

In 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 deficență 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 [?]:

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



Ca și caracteristici principale se pot remarca [?]:

- Conectivitate: Wi-Fi

- Tip alimentare: La retea
- Precizie:  $\pm 2^{\circ}\text{C}$  sau  $\pm 1^{\circ}\text{C}$
- Setare temperatură intre:  $5 - 35^{\circ}\text{C}$
- Temperatură ambientală:  $0 - 45^{\circ}\text{C}$

## 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 [?]:

- 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. Aceasta 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 [?]. 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 [?].

Caracteristicile tehnice ale acestuia sunt [?]:

- Destinat: centrale termice
- Suprafata 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 prezентate anterior, termostatele produse de Google Nest integrează tehnologii revoluționare pentru a obține un randament cat mai mare.

Principalele particularități ale acestuia sunt [?]:

- 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 [?]:



- Sursă de energie: baterii
- Tensiune de funcționare: 24 volți
- Afisaj 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 [?].

Particularități [?]:

- 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 temperatură 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 [?]:

- 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 evolute 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 temperatură 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 aşa fel încât să permită o navigare facilă prin meniurile sistemelor.

În continuare, cu ajutorul datelor preluate din [?], 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.

In 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 cosumul în decursul a 10 zile. Aceasta trimită 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 oferiti 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 prezентate 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 returnul 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 [?]. 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 [?]:

- Flexibilitatea, poate fi utilizat într-un număr vast de domenii, de la programare web, până la programare pe plăcute. 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 [?].

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 [?].

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 [?]. 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++ [?].

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

- 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 [?].

## 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 [?]. 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 [?].

### 3.5.2 Placă de bază pentru NodeMCU

Aceeași 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 [?].

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

În ceea ce privește pinii prezenti pe placa Arduino Uno, există 6 intrări analogice [?], 14 terminale care pot fi configurate fie ca intrări, fie ca ieșiri [?] 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 [?].

Printre caracteristici se remarcă [?]:

- Tensiune de alimentare: 5 volți
- Luminositatea 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 [?], iar distanța de transmisie diferă în funcție de tensiunea de alimentare a transmițătorului și de calitatea antenelor.

Conform [?], se pot evidenția următoarele caracteristici:

- Distanță 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 intentionez 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 releeului 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 Pompa 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 microcontroler. Prezintă două contacte metalice care închid circuitul în momentul în care se atinge.

### 3.5.11 Condensator

Este o componentă electrică pasivă ce are rolul de a înmagazina tensiune. Aceasta 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 Invertor Schmitt-Trigger

Acesta este utilizat pentru a inversa logica semnalului. Dacă se aplică la intrare un semnal cu nivel logic 1, la ieșirea din invertor va avea nivelul logic 0.

### 3.5.14 Diodă

Limităază 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 percepă ca o apăsare multiplă a butonului. Pentru a rezolva aceasta problemă este necesară crearea unor filtre trece jos [?] 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 condesatorului au fost calculate utilizând ecuația de descărcare a condensatorului [?].

$$V_{cap} = V_{initial} \left( e^{-\frac{t}{RC}} \right) \quad (3.1)$$

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

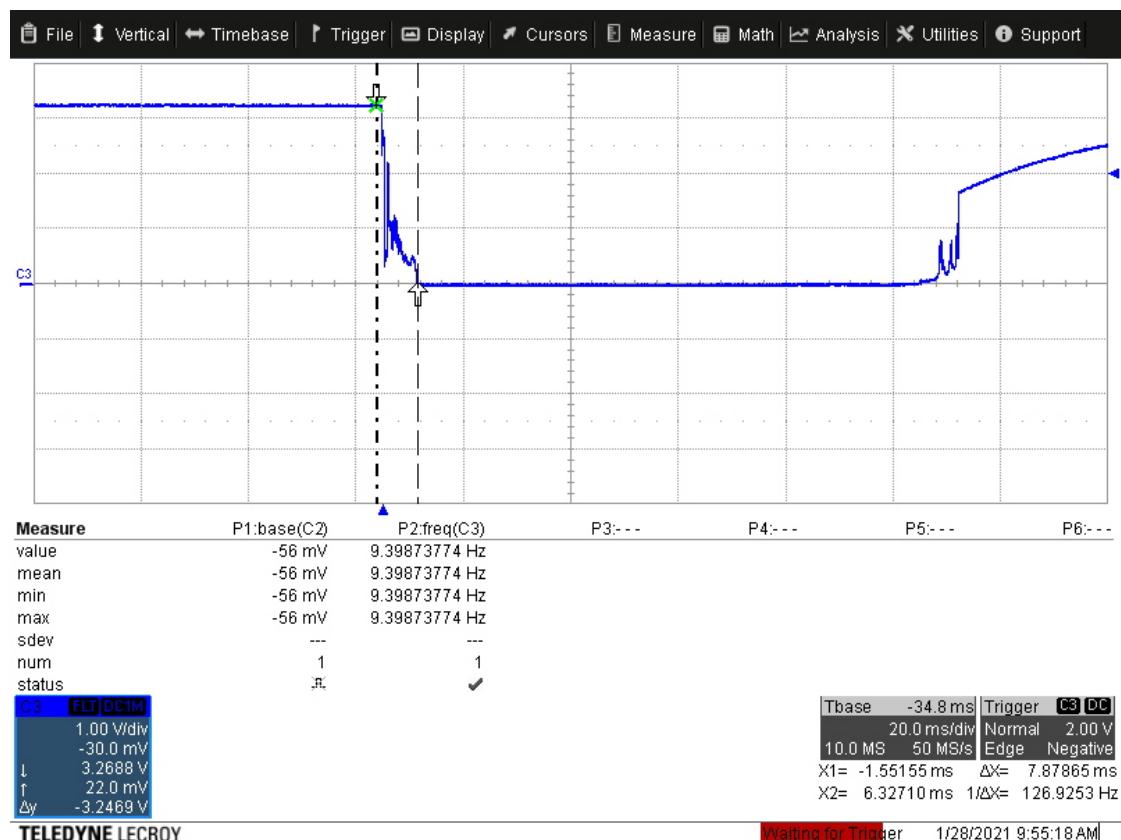


Figura 3.1: Semnal înainte de filtrul trece jos

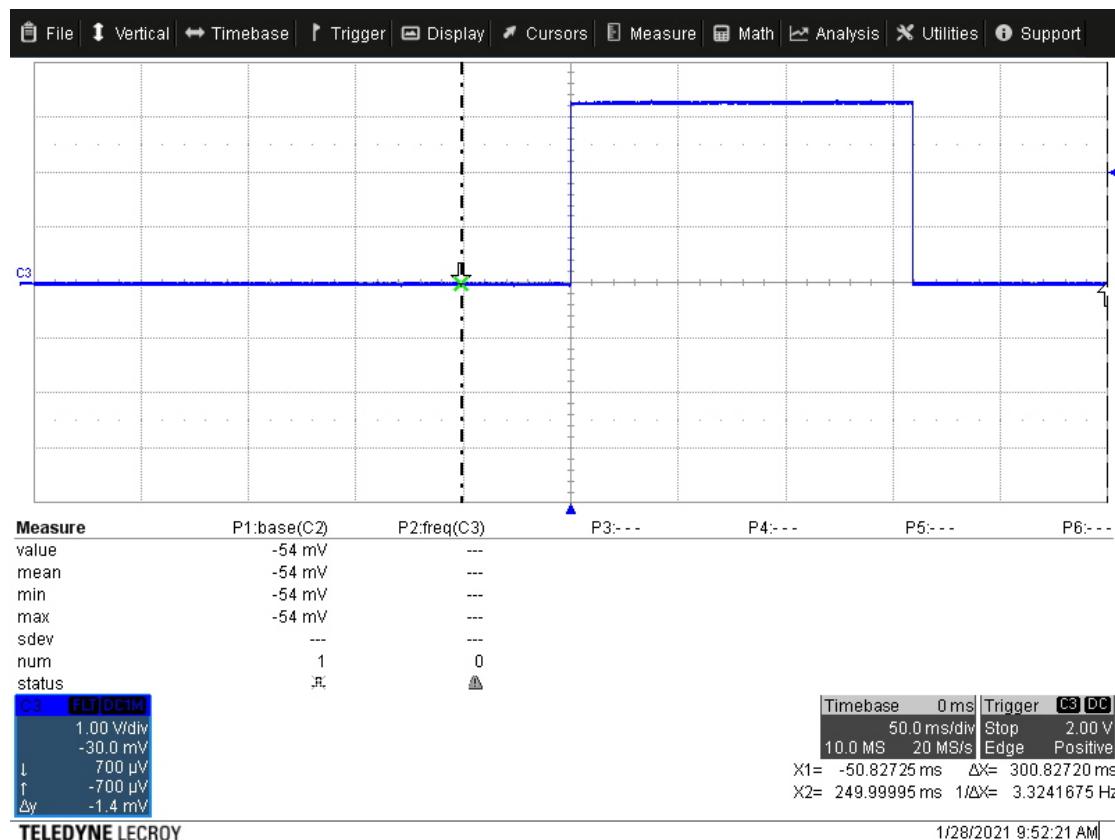


Figura 3.2: Semnal după filtrul trece jos

De asemenea, pentru a obține o filtrare cât mai bună, am utilizat un invertor 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.

### 3.7 Histereza

Pentru a evita crearea unui lanț de porniri și opriri repetitive, pe perioade scurte de timp, cauzate de fluctuațiile rapide de temperatură, este necesară implementarea conceptului de histereză. Aceasta 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 încalzirii 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 valoare 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 prezentarea 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**

...

## 4.2 Arhitectura sistemului

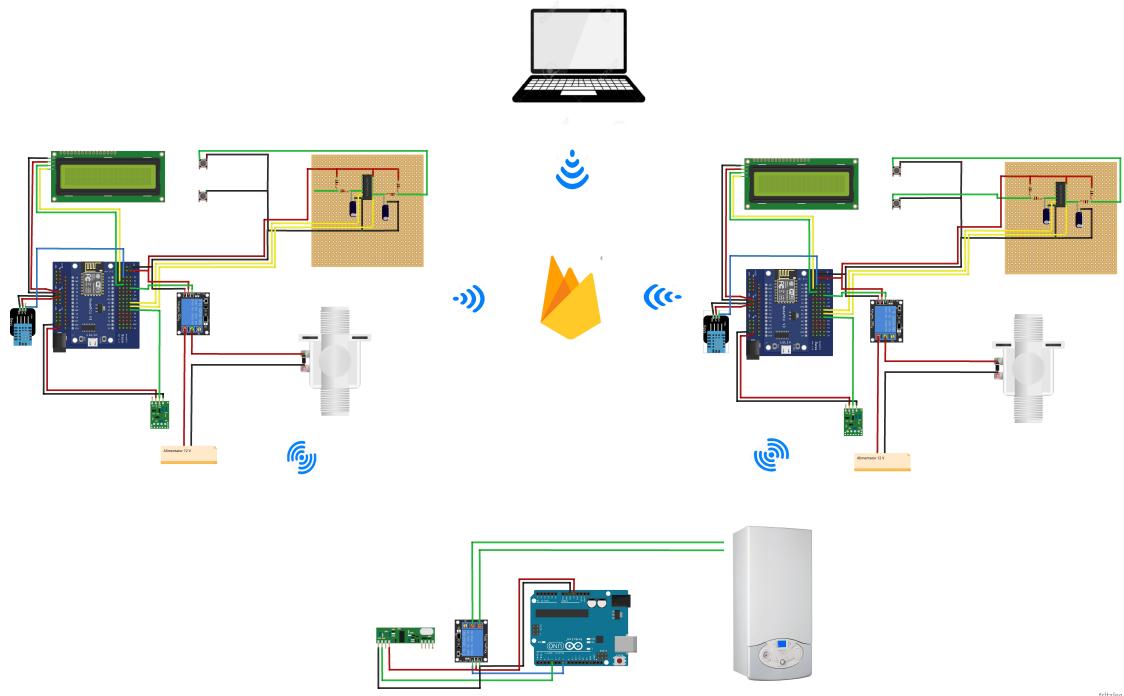


Figura 4.1: Arhitectura sistemului

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 trimită în baza de date. În continuare, aceste valori sunt afișate utilizând aplicația web, făcând posibilă monitorizarea parametrilor ambientali. De asemenea, are acces la temperatura setată, iar în funcție de aceasta și de temperatura curentă din cameră controlează electrovalva și trimită 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, iar în funcție de acestea controlează centrala. Modulul se folosește de logică de tip "SAU", dacă cel puțin un modul senzor trimită 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 intermediuul 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

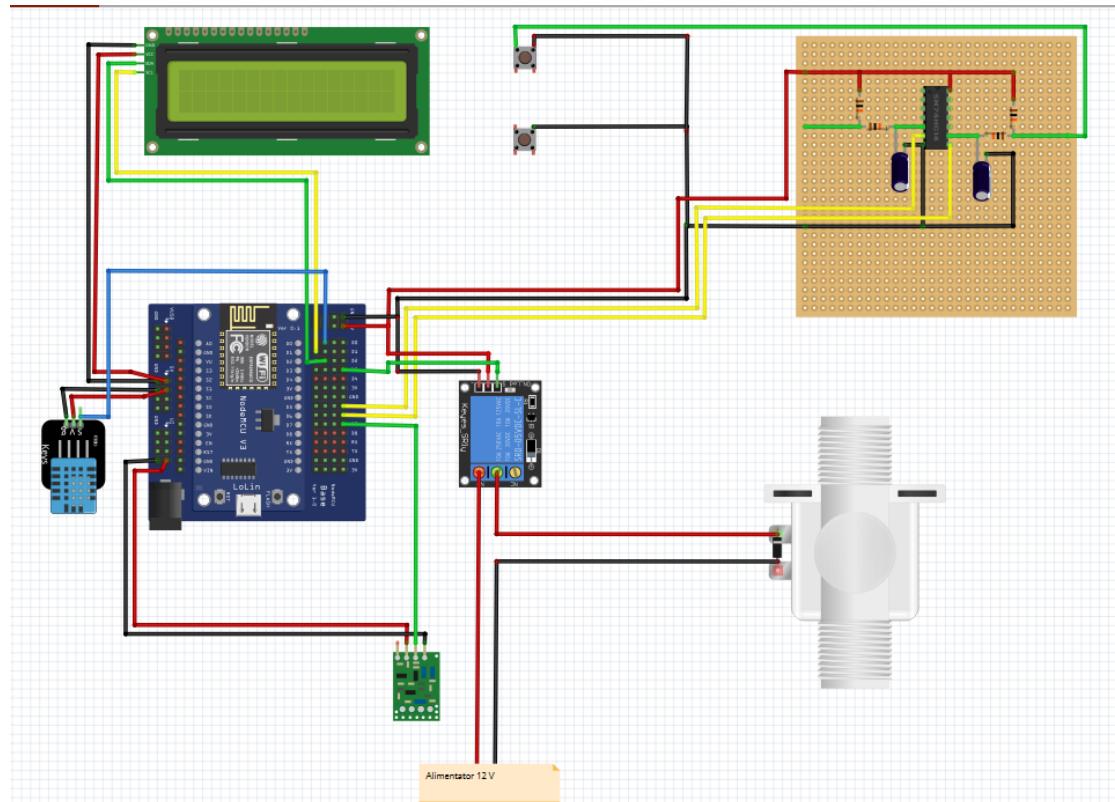


Figura 4.2: Arhitectura modulului senzor

Este format din următoarele componente:

- Modul WiFi NodeMCU
- Placă de bază pentru NodeMCU
- LCD
- Senzor de temperatură și umiditate
- Releu
- Butoane
- Transmițător RF

- Electrovalvă
- Invertor de semnal
- 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 o mufă de alimentare care poate fi conectată la 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 supraalimentare. Astfel, placa de bază pune la dispoziție pini ce oferă tensiuni de ieșire de 3.3 volti, 5 volți și o serie de pini ce sunt conectați direct la mufa 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 utilizati pentru alimentare, VCCC 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 modului 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.

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

Invertorul de semnal, 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

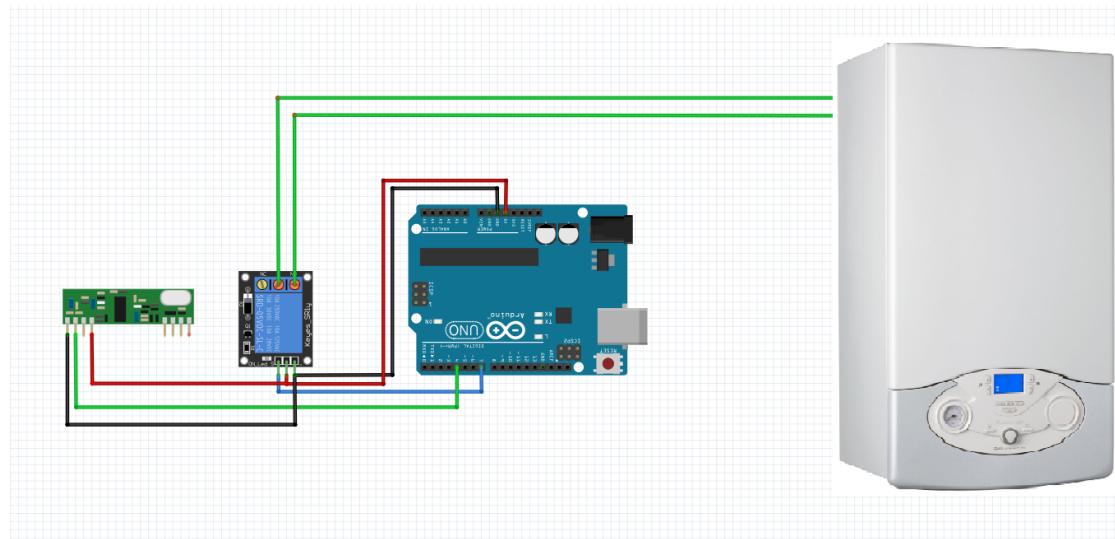


Figura 4.3: Arhitectura modulului de control

Acesta nu prezintă conexiune la internet. Modulul wireless este înlocuit cu o placă 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 volti de la placă Arduino și are rolul de a prelua datele de la transmițătoarele din camere.

## 4.3 Proiectare detaliată

...

## 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 modulu lui. 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 pentru a ști în mod exact data și timpul curent. Acestea sunt necesare 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. Aceasta 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 aşa 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 utilizati pentru a localiza baza de date, respectiv pentru a oferi acces la aceasta.

Setarea temperaturii se realizează prin îintreruperi. Trebuie definit pinul și frontul semnalului pe care se declanșează îintreruperea, 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 îintreruperilor. 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, *increaseDe-*

*siredTemperature* 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 declanșază 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 modulului wireless, conectat sau deconectat 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 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, sămbătă 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 zilelor de la sfârșitul săptămânii, sămbătă și duminică. În cazul în care nu se respectă condiția, se execută secvențele condiționale următoare, împreună cu citirile capitelor 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 conectarea/deconectarea 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 plăcuță 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, iar octetul cel mai puțin semnificativ deține informații legate de natura comenzi. 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 timp mai mare sau egal cu 5 secunde, centrala termică va fi opriță.

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.

Initial, 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 principală 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*. Citirile valorilor umidității și a temperaturii se actualizează în aplicația web doar atunci când apar modificări în baza de date. Astfel, se evită citirile repetitive și consumatoare de resurse.

În ceea ce privește partea de acces a aplicației, aceasta este găzduită pe un server, la distanță, prin intermediul platformei *Keroku*. Oricine deține adresa aplicației 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.

Printre funcționalitățile esențiale pe care le deține aplicația sunt:

- pagina principală, ce apare imediat după o conectare cu succes, prezintă rolurile cele mai importante, și anume: posibilitatea de monitorizare a temperaturii și umidității ambientale, dar și reglarea temperaturii dorite. Este construită pentru două camere, dar poate fi extinsă pentru mai multe.
- pagina de programare a intervalelor orare și temperaturile corespunzătoare acestora este, de asemenea, o funcționalitate utilă pentru utilizatori și pentru partea de control al modului de funcționare al sistemului.
- pentru a permite accesul și altor persoane la aplicație, pagina de creare cont oferă posibilitatea de adăugare a noi intrări în baza de date. Astfel, un administrator poate oferi dreptul de acces și altor utilizatori.
  - un administrator are dreptul de a analiza lista persoanelor care dețin un cont și de a adăuga sau retrage anumite drepturi de acces.
  - o funcționalitate utilă pentru persoanele care primesc conturile cu credențialele setate de către un administrator, este posibilitatea de schimbare a parolei vechi. Astfel, parolele rămân confidențiale fiecărui utilizator în parte.

#### 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 corespondenței credențialelor introduse în formular cu cele salvate în baza de date. În cazul unei potriviri, utilizatorul este

redirectionat la calea `/home`, aceasta apartinând paginii principale și permite monitorizarea parametrilor ambientali și setarea temperaturii dorite. Situația în care se introduc credențiale gresite, 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` cripteză 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 intitulate *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.

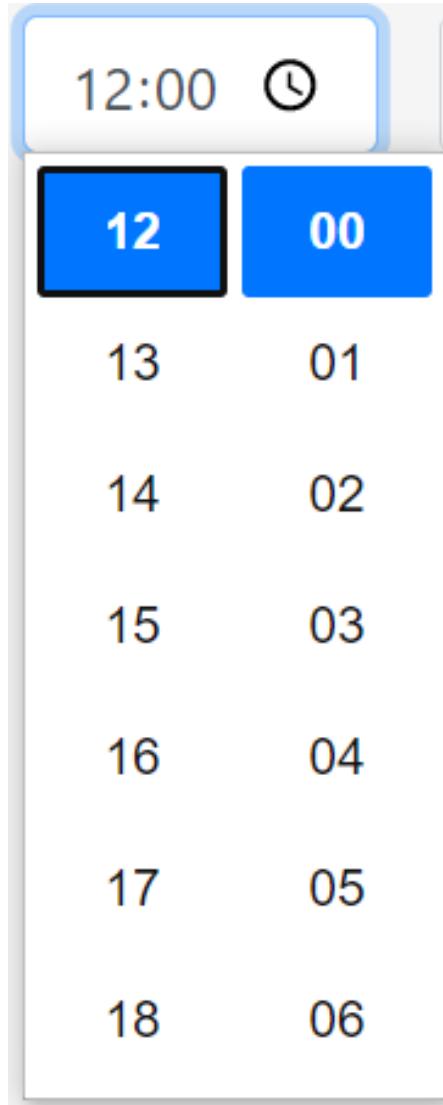


Figura 4.4: Intrare HTML pentru formatul timp

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

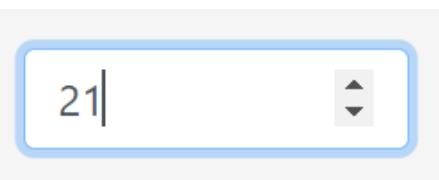


Figura 4.5: Intrare HTML pentru formatul numeric

După ce utilizatorul completează toate câmpurile, este necesar ca aceste valori să fie transmise mai departe partii, din cadrul aplicației web, responsabile cu procesarea informațiilor. Transferul datelor se face prin intermediul unui buton, iar aspectul acestuia este creat în *Bootstrap*.

```
<button type="submit" class="btn btn-primary btn-block">Set  
intervals</button>
```

Înainte de salvarea valorilor în baza de date, este necesară o verificare a acestora. Condiția primordială care trebuie respectată este ca intervalele orare să fie setate în ordine cronologică. În caz contrar, utilizatorul trebuie avertizat printr-un mesaj și i se oferă posibilitatea reintroducerii datelor.

```
timeObjectA = getTime(a)  
timeObjectB = getTime(b)  
timeObjectC = getTime(c)  
timeObjectD = getTime(d)  
  
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:
    return render_template('schedulingPage.html')

```

---

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 desparti sirul 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, ce are setat fusul orar actual.

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: câmpul din baza de date unde se vor stoca datele și valorile acestora.

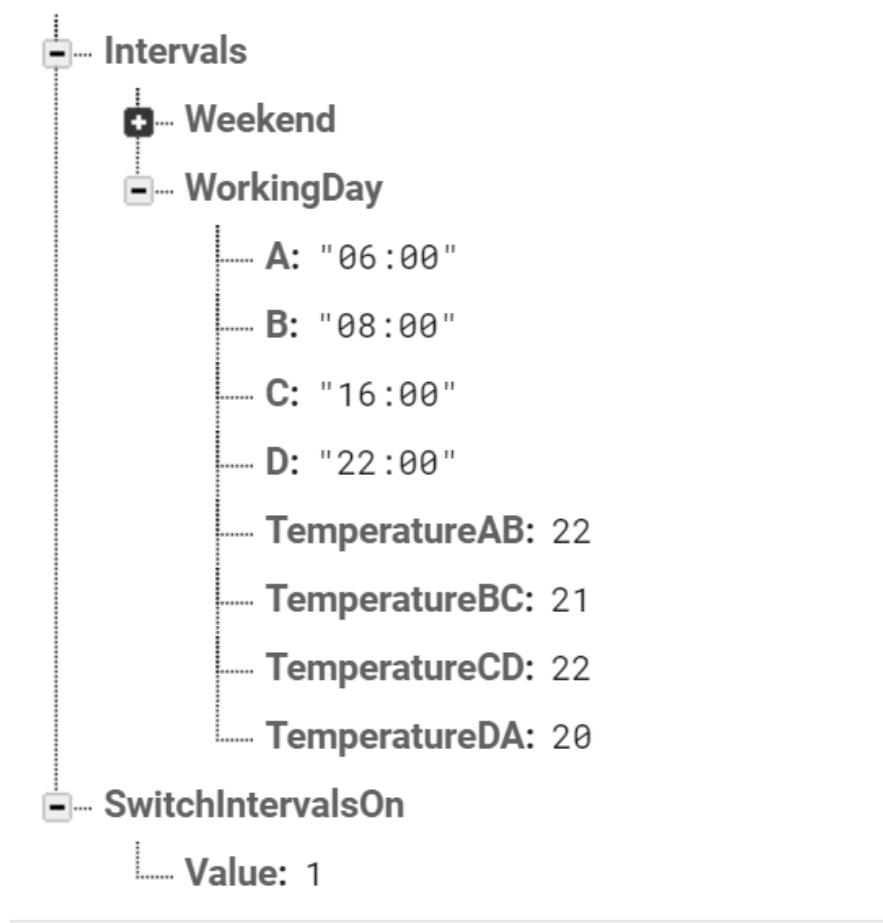


Figura 4.6: Stocare program intervale orare

# Listă de figuri

1.1	Evoluția numărului de dispozitive IoT . . . . .	4
3.1	Semnal înainte de filtrul trece jos . . . . .	16
3.2	Semnal după filtrul trece jos . . . . .	17
4.1	Arhitectura sistemului . . . . .	20
4.2	Arhitectura modulului senzor . . . . .	21
4.3	Arhitectura modulului de control . . . . .	23
4.4	Intrare HTML pentru formatul timp . . . . .	37
4.5	Intrare HTML pentru formatul numeric . . . . .	38
4.6	Stocare program intervale orare . . . . .	40

# **Listă de tabele**