UNIVERSITATEA POLITEHNICA BUCUREȘTI
FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI
DEPARTAMENTUL INGINERIA INFORMAȚIEI

Smarthome cu MCU Infineon/STM (core ARM)

Casă inteligentă – Senzori fără fir pentru automatizarea casei

PROIECT DE DIPLOMĂ

Coordonator științific:
Ș.l. dr. ing. Călin BÎRĂ

Absolvent:
Robert-Valentin ENE

BUCUREŞTI
2022

Universitatea "Politehnica" din Bucureşti **Anexa 1**
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Program de studiu **INF**

TEMA PROIECTULUI DE DIPLOMĂ

a studentului **ENE C. Robert-Valentin, 444A**

1. Titlul temei: Smarthome cu mcu Infineon/STM (core ARM)

2. Descrierea temei și a contribuției personale a studentului (în afara părții de documentare):
Sistemul embedded bazat pe microcontroler Infineon sau STM cu display, care este capabil să învețe coduri de telecomandă IR și să le reproducă la cerere (eg. pentru a stinge televizorul, pentru a porni/opri A/C etc), poate citi temperaturi de la senzori fără fir (nu neapărat wi-fi), poate aprinde/stinge becuri conform detecției prezenței / ușilor deschise etc. Dacă se va folosi mediul Arduino de dezvoltare, se vor scrie și bibliotecile necesare (nu doar call-uri la biblioteci existente). În urma analizei cerințelor, se poate schimba producătorul microcontroler-ului cât timp are nucleu ARM.

3. Discipline necesare pt. proiect:
PC, POO, SDA, AMP

4. Data înregistrării temei: 2021-12-15 09:04:03

Conducător(i) lucrare,
Ş.L.Dr.Ing. Călin BÎRĂ

Student,
ENE C. Robert-Valentin

Director departament,
Ş.L. dr. ing Bogdan FLOREA

Decan,
Prof. dr. ing. Mihnea UDREA

Cod Validare: **082a12c330**

Anexa 5

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul SMARTHOME CU MCU INFINEON/STM (CORE ARM), prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității "Politehnica" din București ca cerință parțială pentru obținerea titlului de *Inginer/ Master* în domeniul *Calculatoare şi tehnologia informaţiei*, programul de studii *Ingineria informaţiei* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, *22.06.2022*

Absolvent *Robert-Valentin ENE*

(semnătura în original)

CUPRINS

CUPRINSUL TABELELOR PREZENTE ÎN LUCRARE

CUPRINSUL FIGURILOR PREZENTE ÎN LUCRARE

SINOPSIS

Automatizarea casei și întreaga idee despre case inteligente devine din ce în ce mai importantă și mai căutată în secolul 21. Obiectul principal al acestui proiect este acela de a dezvolta și a crea un sistem inteligent prin care putem acumula informații de la diferiți senzori din casă pe telefon și de a controla funcționalități ale casei noastre direct de pe telefonul nostru.

Tehnologiile folosite în acest proiect sunt: protocolul de comunicație fără fir ESP-NOW, care este un protocol de transmisie fără fir creat de Espressif între mai multe plăci de dezvoltare ESP32; diferite protocoale de transmisie serială - UART, I2C; baza de date pentru acumularea datelor și protecția acestora; o aplicație de tip *front-end* în care vom avea interfața utilizatorului.

Aplicația propusă permite controlul facil al utilităților din casă: stingerea/aprinderea becurilor, pornirea/oprirea aerului condiționat, pornirea/oprirea televizorului, precum și informarea constantă a utilizatorului cu privire la starea casei: aflarea temperaturii și umidității din casă; cunoașterea nivelului de gaz din incăpere. De asemenea, nu se folosesc telecomenzile aferente fiecărui dispozitiv, ci sistemul este capabil să învețe și să reproducă semnalele infraroșu de la telecomanda fizică, și să folosească, în schimb, butoanele din aplicația web destinate controlului fără fir al dispozitivelor infraroșu din casa inteligentă. Toate acestea sunt implementate în rețeaua proprie a utilizatorului, fără a fi nevoie de fire și rețele fizice complexe.

1 INTRODUCERE

Oamenii, de multe ori, au tendința de a face lucrurile mai ușoare, și nu este un lucru de condamnat. Se poate observa cum lumea din zilele noastre se îndreaptă din ce în ce mai mult spre tehnologie, și acesta este un lucru extraordinar. Dacă tehnologia se dezvoltă, este bine să o aplicăm și să o folosim în mod direct în viețile noastre. Oamenii sunt destul de ocupați la locul de muncă încât nu mai doresc să aibă stresul provocat de anumite neajunsuri în propria casă. Din aceste motive (și din altele pe care le voi prezenta în continuare), aplicațiile pentru case inteligente au o importanță deosebită, acestea influențând semnificativ starea și emoțiile ființei umane.

1.1 Context

Proiectul propus are în vedere crearea unui sistem prin care putem controla propria casă doar din telefonul mobil, fiind mai ușor și mai la îndemână. Automatizarea casei este de multe ori numită pur și simplu „casă inteligentă”. Pentru realizarea acestui proiect se folosește tehnologie IoT – Internet of Things (Figura 1 Componente IoT). Aceste sisteme IoT sunt formate atât din componente hardware cât și din componente software, toate lucrând împreună pentru a aduce la finalitate un produs dezvoltat din punct de vedere al tehnologiei și al științei.

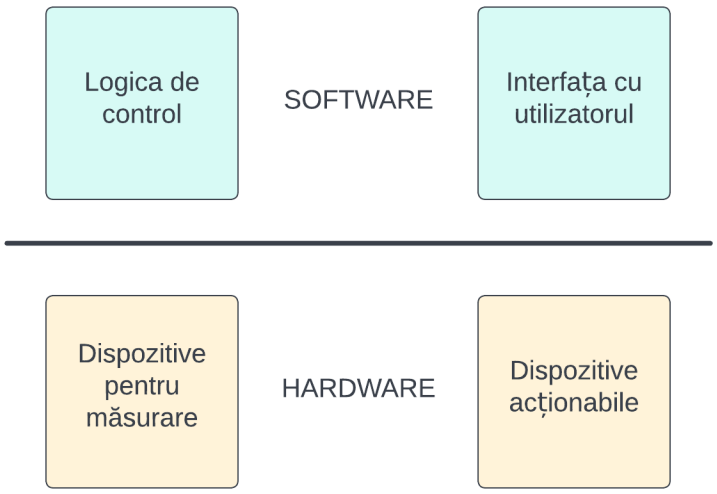

Figura 1 Componente IoT

1.2 Problema

În România încă nu este atât de extins acest domeniu de case inteligente deoarece soluțiile apărute pe piață sunt destul de scumpe și greu de implementat, mai ales dacă locuința este deja construită. Principalele probleme sunt lipsa banilor, fără de care nu prea se poate construi un ecosistem prietenos prin care să se introducă funcționalități inteligente la propria casă. Sistemele apărute pe piață, sunt într-adevăr inteligente, dar numai un Hub inteligent pentru controlul luminilor costă destul de mult.

O altă problemă este că atunci când se dorește a se implementa un sistem inteligent pentru locuință, trebuie să fie introduse tot felul de cabluri. Probabil este nevoie de canale de cabluri prin pereți și este greu să se ia această decizie luând în considerare implicațiile.

1.3 Obiective

Obiectivul principal al acestui proiect este oferirea unei soluții complete pentru crearea unei case inteligente, cu nivel de finanțe redus, care să îndeplinească cerințele unui sistem automatizat și inteligent (Figura 2 Utilități în casa inteligentă). Pornind de la configurarea unei camere cu diferiți senzori care transmit starea casei aproape în fiecare moment al zilei, această soluție se poate răspândi în întreaga casă. Implementarea proiectului este facilă, iar aplicația poate fi ușor de accesat și folosit în viața de zi cu zi de către utilizatori, astfel devenind o soluție de luat în calcul când vine vorba de automatizarea unor funcționalități într-o casă inteligentă.

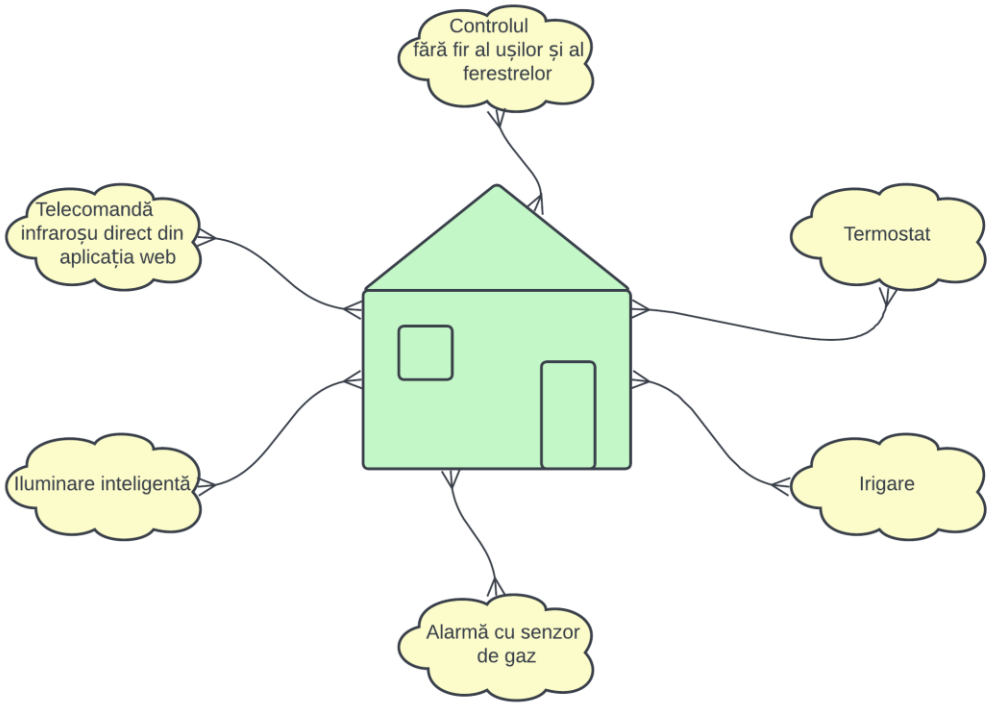

Figura 2 Utilități în casa inteligentă

1.4 Soluția propusă

Pentru a atinge obiectivele acestui proiect, soluția propusă este alcătuită din trei componente: o aplicație web care are în vedere informarea utilizatorilor cu privire la starea casei și în același timp controlul diferitelor funcții pe care casa inteligentă le pune la dispoziție, o componentă hardware pentru măsurarea indicatorilor cu ajutorul senzorilor și o aplicație server pentru colectarea și gestionarea datelor.

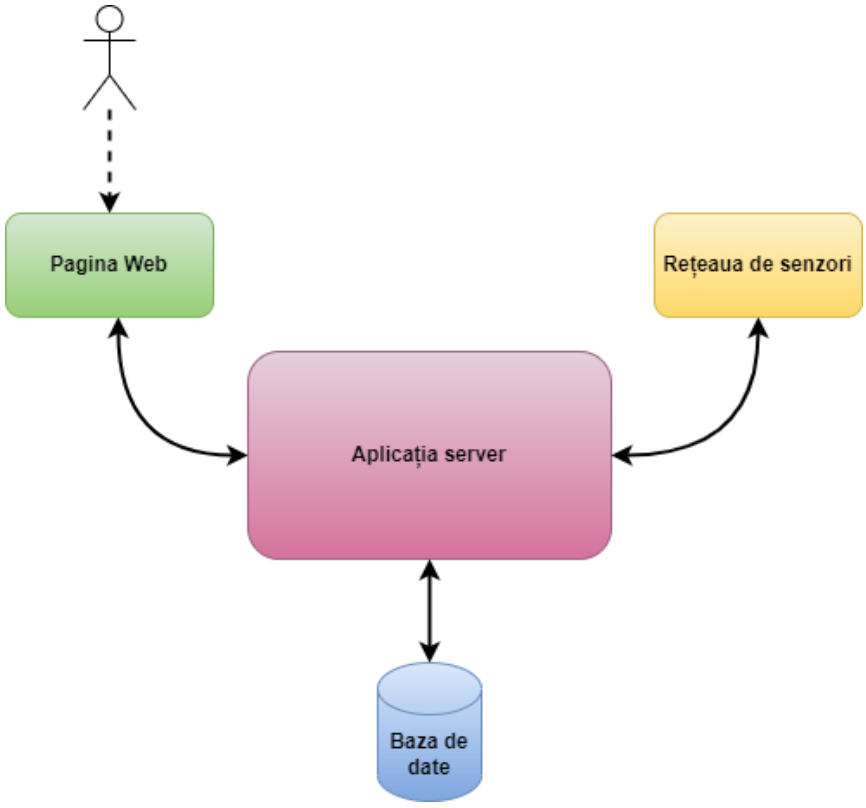

Figura 3 Arhitectura soluției propuse

Prototipul hardware este un dispozitiv propus pentru măsurarea diferitelor informații și date provenite de la senzori. Acesta va fi construit dintr-o stație de bază – un kit de dezvoltare Nucleo-64, senzori pentru măsurarea luminozității din încăpere (KY-018), măsurarea temperaturii și a umidității (ambele putând fi preluate de la un singur senzor – DHT11), măsurarea nivelului de gaz din încăpere (MQ-2).

Pe lângă senzorii care transmit date despre starea casei, avem și componente hardware care ajută la automatizarea locuinței – senzor PIR pentru facilitarea aprinderii luminii la detectarea prezenței, senzori IR care detectează prezența unor unde infraroșu, iar sistemul poate învăța coduri de la telecomanda TV/AC și poate transmite mai departe acele coduri direct din aplicația web, motoare pas-cu-pas care ajută la închiderea/deschiderea ușilor/ferestrelor în cazul în care utilizatorii pleacă de acasă, sau se detectează o cantitate semnificativă de gaz sau de fum.

Aplicația server, detaliată în capitolul 5 – Soluția propusă, este responsabilă pentru colectarea datelor. Aceasta primește în mod constant informații utile de la senzori și le publică la endpoint-ul programat. Aplicația server interoghează prototipul hardware, acesta efectuând măsurători la intervale de timp bine stabilite, iar fiecare senzor ajunge să trimită datele către server.

Aplicația web este destinată oricărui utilizator, având în vedere un public larg. Aceasta îndeplinește mai multe roluri. Un prim rol este cel de informare: utilizatorii se pot documenta, într-o pagină dedicată, cu privire la starea locuinței, temperatura și umiditatea din încăpere și nivelul de gaz sau fum și impactul pe care acesta l-ar putea avea asupra sănătății acestora. Există în aceeași pagină dedicată o secțiune unde utilizatorul poate urmări grafice cu valorile senzorilor în ultimele 2 ore, astfel putând să aibă o privire de ansamblu a istoricului casei.

1.5 Structura lucrării

În continuare vom aprofunda motivațiile acestui proiect în capitolul 2: vom răspunde la întrebări precum „de ce este nevoie de o asemenea aplicație”, „ce valoare aduce utilizatorului aplicația”, „care este impactul pe care o astfel de implementare o are în viața de zi cu zi” și „de ce soluțiile existente nu sunt suficiente pentru împlinirea scopului acestui proiect”.

În capitolul 3 vom analiza soluțiile deja existente pe piață în acest sens, care sunt avantajele și dezavantajele acestora, cu ce diferă soluțiile deja existente cu sistemul propus în această lucrare, apoi în capitolul 4 vom prezenta ce tehnologii au fost folosite în realizarea acestui proiect, urmând ca în capitolul 5 să fie prezentată soluția propusă pentru a acoperi neajunsurile soluțiilor existente prezentate în urmă cu 2 capitole. Vom vedea care este arhitectura soluției propuse, cum intercomunică componentele ce alcătuiesc întreg sistemul propus, cum sunt construite și ce rol îndeplinesc aceste componente și, în final, ce aduce în plus această soluție față de cele prezentate în capitolul 3.

În capitolul 6 vom evalua aplicația din mai multe perspective și vom vedea dacă aceasta și-a atins scopul și poate deservi utilizatorii în sensul anunțat anterior. La final, în urma acestor evaluări vom trasa concluziile acestui proiect (în capitolul 7).

2 ANALIZA ȘI SPECIFICAREA CERINȚELOR

Dezvoltarea prezentului proiect este motivată din mai multe direcții: din punct de vedere informatic, al integrării tehnologiei în viața de zi cu zi cu scopul de a îmbunătăți viața și de a ne-o face mai ușoară; din punctul de vedere al confortului, de a transforma o casă obișnuită într-una inteligentă, cu diferite funcționalități controlabile din propriul telefon inteligent și din punct de vedere al finanțelor de nivel ridicat necesare pentru implementarea acestui gen de proiecte. Nu în ultimul rând, o problemă este și cea medicală și psihologică din punct de vedere al stresului la care utilizatorul este supus atunci când nu este în controlul propriei locuințe. În continuare vom dezvolta pe rând aceste motivații.

Motivația din punct de vedere informatic este foarte importantă deoarece omul merită să fie la curent cu tot ce se întâmplă în casă. Tehnologia ne ușurează viața și în același timp o îmbunătățește. Având la dispoziție aplicația de informare cu privire la senzorii plasați în interiorul casei, utilizatorii vor fi mult mai liniștiți cu privire la ce se întâmplă în interiorul locuinței, stresul la nivel medical se va micșora, iar oamenii nu vor mai pune atât de mult accent pe panica creată de faptul că propria locuință nu este în siguranță.

Transformarea unei case obișnuite într-una inteligentă este necesară deoarece tehnologia evoluează pe zi ce trece și este păcat să nu o folosim pentru binele propriei ființe. Tehnologia ne-a uşurat şi îmbunătăţit viaţa şi nivelul de trai şi va continua să o facă, și aceasta ne este la îndemână. Deja folosim laptopuri, telefoane inteligente, maşini digitalizate, electrocasnice inteligente. Toate acestea ne-au impactat viaţa personală într-un fel sau altul. Este momentul să integrăm și la nivel de locuință această inteligență. Funcționalitățile pe care le poate oferi o astfel de aplicație sunt simple, dar pot aduce traiul de zi cu zi al vieții la un alt nivel, având siguranța că totul este sub control.

Prezentul proiect propune o soluție pentru a îmbunătăți viața colectivă, în afara spațiului personal al casei, dar nu doar atât. Această soluție permite atât controlul luminilor și diferitelor funcții ale casei din telefon, cât și vizualizarea în timp real a anumitor parametri importanți: temperatura, umiditatea, nivelul de gaz din încăpere, nivelul de luminozitate. Având la dispoziție o aplicație în care informația este doar la un clic distanță, utilizatorul își poate pune încrederea în propria casă și poate pleca la drum fără a mai avea frică pentru ce se poate întâmpla în lipsa acestuia.

Din punct de vedere medical, stresul este unul din factorii care pot duce la îmbolnăvirea pacienților foarte grav. Acesta este un fenomen psihosocial complex care decurge din confruntarea persoanei la nivel psihologic cu diferite sarcini și situații pe care creierul uman le percepe ca fiind dificile, greu de rezolvat, provocând gândire excesivă asupra unor aspecte care, inițial, nu par foarte grave, dar ajung să necesite atât de multă energie și efort depus.

3 STUDIU DE PIAȚĂ / SOLUȚII EXISTENTE

Oamenii interacționează cu mediul din jurul lor prin diferite moduri. Dacă mediul cu care aceștia interacționează este unul prietenos și care răspunde înapoi cu informații, acesta poate aduce un mare avantaj. O aplicație pentru o casă inteligentă introduce această comunicare a utilizatorilor cu propria locuință, și face din aceasta o proprietate cu un mediu prietenos, dând posibilitatea de control total și informare continuă.

Tehnologia a evoluat în ultima perioadă și vedem constant cum știința pune stăpânire pe dorința oamenilor de a se dezvolta și de a implementa noi idei chiar în propriile case. IoT este un domeniu care a luat amploare în ultimii ani, iar în următoarea figură (Figura 4 Evoluția dispozitivelor în tehnologia IoT) se poate observa cât de multe dispozitive au ajuns să poată permite conectarea fără fir a acestora și integrarea tuturor dispozitivelor prezente în case inteligente, și nu numai, împreună, formând astfel ceea ce se numește „Internet of Things”. [1]

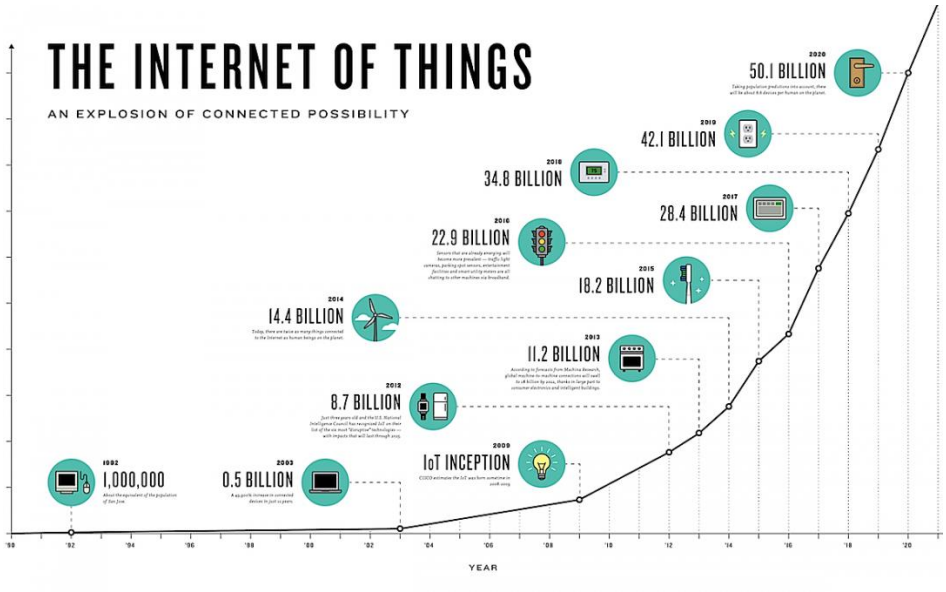

Figura 4 Evoluția dispozitivelor în tehnologia IoT

Una din abordările uzuale cu privire la crearea unui mediu inteligent pentru locuințe este automatizarea funcționalităților prezente deja în interiorul casei. Pot fi utilizați anumiți senzori (senzori PIR) pentru a se reduce costurile privitoare la consumul de curent, sau senzori care să transmită informații utile proprietarilor (senzori de gaz, senzori de temperatură). În același timp, este nevoie de microcontrolere care facilitează comunicația senzorilor cu aplicația pentru casa inteligentă.

Toate acestea sunt integrate într-un sistem inteligent ce poate capta informații și date precise de la senzori, în același timp facilitând interacțiunea utilizatorului cu sistemul. Până acum pe piață au apărut multe soluții pentru problema expusă în această lucrare, dar majoritatea fie au costuri ridicate, fie sunt greu de implementat și necesită schimbări majore în locuință, chiar dacă poate sunt tehnologii avansate, ce conțin integrare cu asistenți vocali precum *Amazon Alexa* sau *Google Assistant*.

Comunicațiile între senzori și microcontroler sau microcontroler și server/aplicație se pot realiza prin mai multe protocoale prezente deja pe piață. Aceste protocoale au diferite avantaje și dezavantaje (Tabelul 1 Avantaje și dezavantaje protocoale de comunicație); unele pot fi comunicații seriale, care necesită prezența unui canal fizic de comunicație (cablu, fire), iar altele pot fi comunicații fără fir și aici se folosesc, din nou, mai multe protocoale și tipuri de comunicație.

În continuare vom prezenta diverse protocoale și tipuri de comunicație, și vom pune în lumină avantajele și dezavantajele acestora.

- Ethernet – Protocol de comunicație standard, care utilizează transmiterea datelor prin cablu, permite transmiterea unor cantități mari de date și acestea pot fi transmise cu ușurință și destul de rapid. Tehnologia Ethernet permite pachetelor de date să fie transmise cu ajutorul cablurilor fizice, create din perechi de fire răsucite. Vitezele de transmisie atinse cu acest protocol sunt de 1Gbps la cablurile de tip CAT5e, iar la cele de tip CAT6a se poate ajunge la viteze de 10Gbps.
- Wi-Fi – Protocol de comunicație standard, care utilizează transmiterea datelor prin aer, permite interoperabilitatea (abilitatea sistemelor sau proceselor de a lucra împreună pentru realizarea unui scop comun) și comunicarea facilă între mai multe dispozitive diferite. Pachetele transmise prin protocolul de comunicație Wi-Fi circulă prin unde radio în spectrul de frecvențe de 2.4GHz sau 5GHz. Vitezele atinse de acest protocol de transmisie de date sunt cuprinse între 10Mbps și 100Mbps. De asemenea, protocolul Wi-Fi este capabil să transmită elemente audio și video la calitate superioară.
- Zigbee – Protocol de comunicație fără fir, dezvoltat pentru a face posibilă comunicația cu o viteză relativ redusă, dar, în același timp, folosind o bandă de frecvență redusă. Acest protocol de transmisie al datelor se folosește pentru comunicații pe distanțe reduse. Undele radio transmise prin acest protocol operează la frecvența de 2.4GHz. Vitezele de transmisie sunt de 250kbps și de obicei se transmit și se primesc diferite comenzi reduse ca dimensiune deoarece frecvența folosită este una lentă (comparativ cu Wi-Fi de 5GHz).
- Bluetooth – Protocol de comunicație fără fir, ce face posibilă comunicația între diferite dispozitive, pe o distanță redusă. Majoritatea dispozitivelor (în special telefoanele inteligente) au la dispoziție și includ acest protocol în arhitectură, ceea ce face din acest protocol unul facil când vine vorba de comunicarea între dispozitive într-o aplicație creată pentru o casă inteligentă. Și acest protocol folosește banda de frecvență de 2.4GHz. Protocolul funcționează pe o setare de *controler/agent*. Variantele Bluetooth 3.0 și Bluetooth 4.0 permit transmiterea datelor la o viteză aproximativă de 24Mbps. [2]

În tabelul de mai jos (Tabelul 1 Avantaje și dezavantaje protocoale de comunicație), voi prezenta avantajele și dezavantajele pe care le oferă fiecare din protocoalele de comunicație prezentate succint mai sus.

Protocol	Avantaje	Dezavantaje
Ethernet	- viteza de transmisie a datelor - distanța facilă pe care se poate face transmisia - securitatea de care dispune rețeaua fizica, pe cablu	- costul ridicat necesar instalării rețelei de cabluri - necesitatea unor switch-uri - planificarea corectă și din timp a casei pentru a introduce rețeaua de cabluri prin perete, pentru a nu fi la vedere
Wi-Fi	- viteza de transmisie a datelor - raza de acoperire - tehnologia și informația este la îndemână - un ruter fără fir este mult mai ieftin decât crearea unei întregi rețele pe cablu care conține switch-uri	- competiția dispozitivelor este foarte mare în aceeași bandă de frecvență: telefoane, tablete, ceasuri inteligente, termostate - consumul de putere este mare la dispozitivele Wi-Fi - la nivel de securitate este necesară o setare corectă și atentă
Zigbee	- rețea flexibilă - necesită consum de putere mic - dispozitivele care folosesc Zigbee sunt pe baterii, ceea ce face ca rețeaua Zigbee să fie redusă din punct de vedere arhitectural și fizic ca dimensiune	- limitat din punct de vedere al accesoriilor disponibile pe piață - nu este atât de popular - nu este sigur din punct de vedere al securității - transmiterea datelor se face cu viteză redusă
Bluetooth	- metodă simplă de împerechere a dispozitivelor - utilizează comunicație care are un consum redus de putere - accesorii reduse din punct de vedere arhitectural și fizic ca dimensiune, majoritatea dispozitivelor utilizând baterii - implementarea este facilă din punct de vedere al costurilor materiale	- utilizează banda de frecvență destul de ocupată: 2.4GHz - comunicațiile se fac pe o distanță redusă - poate pierde conexiunea destul de rapid în condiții precare

Tabelul 1 Avantaje și dezavantaje protocoale de comunicație

După ce am studiat ce protocoale sunt pe piață, ce se poate folosi într-o aplicație de casă inteligentă, se poate face și o comparație (Tabelul 2 Comparație soluții iluminare inteligentă) [3] cu ce soluții există pentru o anumită funcționalitate, ce caracteristici dețin acestea, ce tipuri de comunicație oferă, durata de viață, etc.

Parametru	Iluminare inteligentă		
Produs	LIFX Mini White	LIFX A19+	Philips Hue White and Color + Bridge
Conectivitate	Wi-Fi (nu necesită HUB), 802.11 b/g/n	Wi-Fi (nu necesită HUB), 802.11 b/g/n	Wi-Fi (necesită punte), Zigbee
Integrarea în casa inteligentă	Home Kit, Amazon Alexa, Google Assistant	Home Kit, Amazon Alexa, Google Assistant	IFTT, Logitech, Amazon Alexa, Home Kit, Google Home and Assistant
Activare vocală	Da	Da	Da
Consumul de putere	Echivalentul a 60W, consumă 8W de putere	Echivalentul a 60W, consumă 11W de putere	Echivalentul a 60W, consumă 10W de putere
Durata de viață	22.8 ani	22.8 ani	2.3 ani
Cost	$34/bec	$120/bec	$150/bec

Tabelul 2 Comparație soluții iluminare inteligentă

4 TEHNOLOGII FOLOSITE

4.1 Arduino IDE

Arduino IDE (Integrated Development Environment) este componenta software oficială introdusă de Arduino.cc și este folosită în principal pentru editarea, compilarea și încărcarea codului pe un dispozitiv Arduino. Majoritatea componentelor hardware Arduino sunt compatibile cu acest IDE. Câteva avantaje se pot observa în următoarea figură (Figura 5 Funcționalități Arduino IDE [4]).

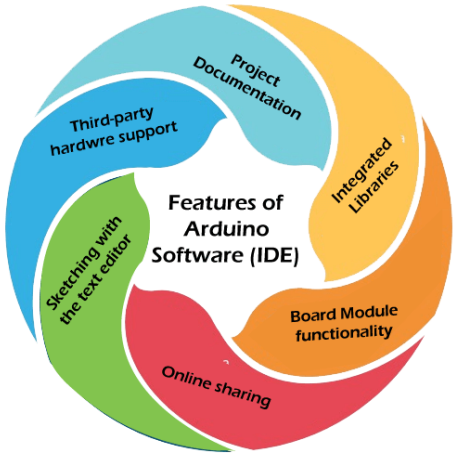

Figura 5 Funcționalități Arduino IDE

Codul în Arduino este scris în limbajul C++, dar pe lângă acesta, are biblioteci, metode și funcții dedicate. C++ este un limbaj de programare citibil de către om, iar atunci când programul sursă este creat, codul este compilat și procesat în cod mașină. [5]

Structura de bază a fiecărui cod scris în Arduino IDE are 4 părți principale:

- Includerea bibliotecilor necesare parcurgerii codului fără posibile erori

- Definirea pinilor folosiți de partea hardware, declararea variabilelor și crearea obiectelor care sunt instanțe ale unor clase create anterior

- Apelarea funcției *setup()* – aceasta definește starea inițiala a plăcii Arduino și se execută doar o singură dată, la inițializarea sistemului. În această funcție se definesc funcționalitățile pinilor declarați anterior utilizând funcția *pinMode()*, starea inițială a pinilor și inițializarea variabilelor

- Apelarea funcției *loop()* – aceasta este obligatorie în orice fișier Arduino (ca și funcția *setup()*), și se execută imediat după ce se termină de executat funcția *setup()*. Funcția *loop()* este funcția main (principală) și, așa cum indică și numele acesteia, se execută continuu, la nesfârșit.

Arduino este o platformă foarte versatilă. Conține suport pentru majoritatea plăcilor de dezvoltare (acestea trebuie instalate din Arduino IDE), o varietate mare de biblioteci ce fac viața dezvoltatorilor mai ușoară – acestea conținând multe funcții, metode și clase care ajută utilizatorii software-ului să ajungă rapid la rezultatul dorit.

4.2 Visual Studio Code

Visual Studio Code, denumit și VS Code, este un editor de cod sursă realizat de Microsoft pentru Windows, Linux și macOS. Caracteristicile includ suport pentru depanare, evidențierea sintaxei, completarea inteligentă a codului, fragmente, refactorizarea codului și Git încorporat. [6]

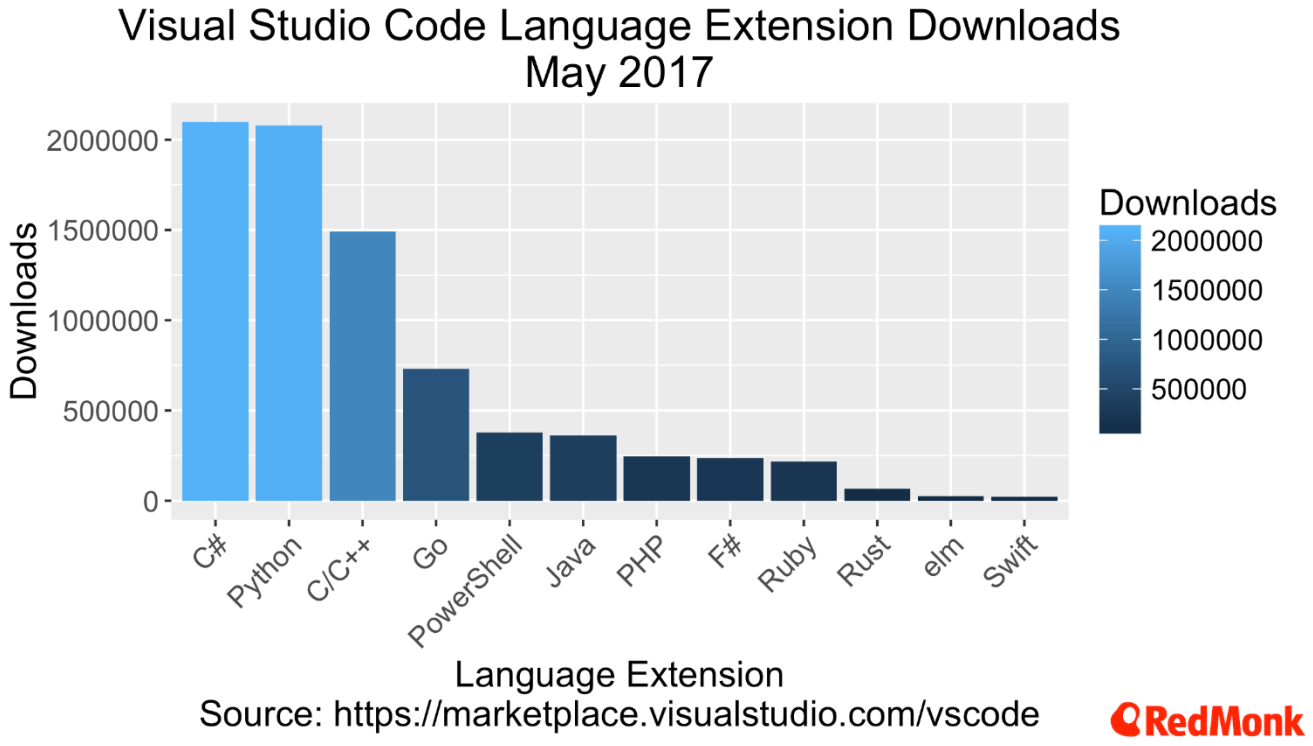

Figura 6 Limbaje folosite și dezvoltate în Visual Studio Code

În Visual Studio Code putem scrie mai multe limbaje de programare precum C/C++, JavaScript, Python și multe altele (Figura 6 Limbaje folosite și dezvoltate în Visual Studio Code). Acesta are o interfață prietenoasă, ușor de înțeles și se dezvoltă în continuu din 2015 până în prezent. La acest proiect au participat mai mulți dezvoltatori dar principalul om care a lucrat a fost Erich Gamma, acesta creând și primul commit pentru acest proiect.

În centrul său, Visual Studio Code are un editor de cod sursă foarte rapid, perfect pentru utilizarea de zi cu zi. VS Code încurajează productivitatea cu evidențierea sintaxelor, potrivirea parantezelor, indentarea automată și multe altele. Comenzile rapide intuitive de la tastatură, personalizarea ușoară și mapările de comenzi rapide de la tastatură permit navigarea cu ușurință în cod.

Pentru o codare serioasă, de asemenea se poate beneficia de instrumente cu mai multă înțelegere a codului decât blocuri de text. Visual Studio Code include suport încorporat pentru completarea codului IntelliSense, înțelegerea și navigarea bogată a codului semantic și refactorizarea codului.

Acest software include un depanator interactiv, astfel încât utilizatorii să poată parcurge codul sursă, să inspecteze variabilele, să vizualizeze stivele de apeluri și să execute comenzi în consolă.

De asemenea, VS Code se integrează cu instrumente de creare și scriptare pentru a efectua sarcini obișnuite, făcând fluxurile de lucru de zi cu zi mai rapide. VS Code are suport pentru Git, astfel încât se poate lucra cu controlul sursei fără a părăsi editorul, inclusiv vizualizarea diferențelor de modificări în așteptare. [7]

La acest proiect, s-a ales folosirea editorului de text Visual Studio Code deoarece acesta integrează perfect tehnologiile necesare pentru dezvoltarea soluției propuse și putem observa și în figura de mai jos (Figura 7 Evoluția popularității editoarelor de text 2015-2019) [8] cum evoluția acestuia este una masivă în ultimii ani, acest editor ajungând mult mai popular decât majoritatea programelor și aplicațiilor existente în acest domeniu.

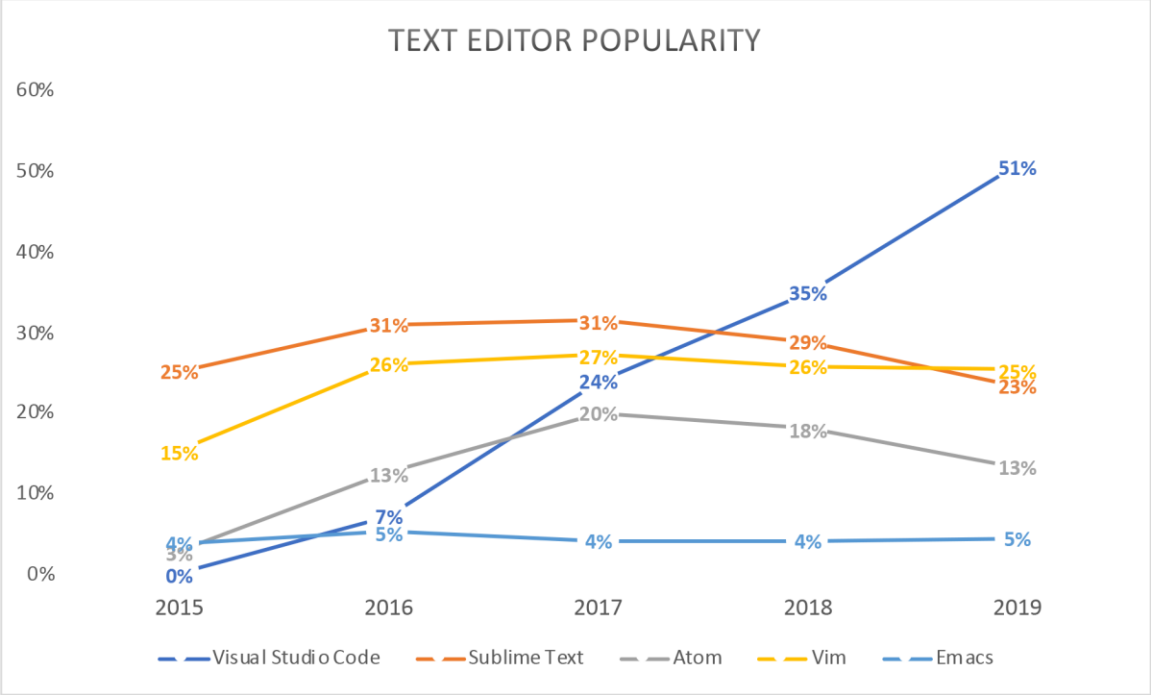

Figura 7 Evoluția popularității editoarelor de text 2015-2019W

5 SOLUȚIA PROPUSĂ

Soluția pe care o propunem este formată din 3 componente principale: aplicația server, aplicația web și o rețea hardware (Tabelul 3 Componente hardware folosite în realizarea proiectului) compusă din senzori care este distribuită în fiecare încăpere din locuință. Rețeaua hardware este alcătuită din componente individuale asemenea prototipului propus în continuare (Figura 8 Schema bloc a sistemului propus). Fiecare dintre aceste componente vor fi analizate, pe rând, în capitolul curent.

Schema bloc a soluției propuse este descrisă în următoarea figură:

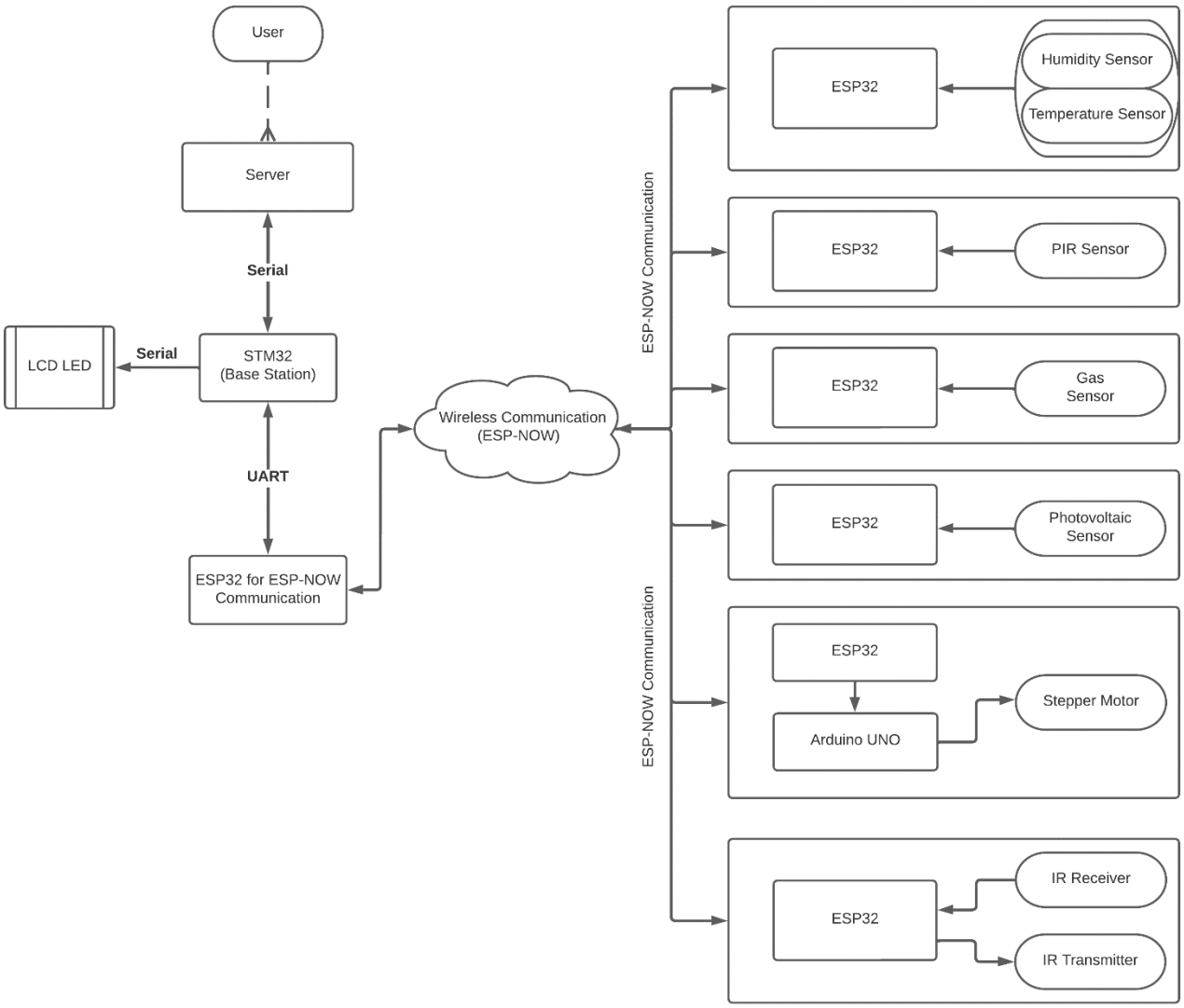

Figura 8 Schema bloc a sistemului propus

Aplicația server va avea ca furnizor de date stația principala (microcontroler STM32) care va trimite informațiile printr-o comunicație serială (UART). Aceste date se transmit server-ului la intervale de timp bine stabilite, iar acesta din urmă stochează datele într-o bază de date timp de 2 ore, acestea fiind accesibile utilizatorului prin clientul web.

Interfața cu utilizatorul va fi simplă, ușor de folosit și va beneficia de funcționalități precum prezentarea sub forma unor grafice a nivelului de gaz din încăpere, nivelului temperaturii și al umidității sau oferirea posibilitatății utilizatorului de a controla unitați de iluminare din locuință.

Scopul atins de componenta hardware	Denumirea componentei hardware
Senzor pentru monitorizarea umidității din încăpere	**Modul DHT11**
Senzor pentru monitorizarea temperaturii din încăpere	**Modul DHT11**
Senzor de mișcare pentru controlul automat al luminilor	**Senzor PIR HC-SR501**
Senzor pentru detecția cantității de gaz	**Senzor MQ-2**
Senzor pentru monitorizarea nivelului luminozității din încăpere	**Modulul KY-018 cu fotorezistor**
Motor pas-cu-pas pentru controlul draperiilor	**Motor pas-cu-pas 28BY J48 + Driver ULN2003**
Senzor de recepție al undelor infraroșu	**Modul receptor infraroșu KY-022**
Senzor de transmisie al undelor infraroșu	**Modul emițător infraroșu KY-005**
Placă de dezvoltare pentru alimentarea și controlul motorului pas-cu-pas	**Arduino UNO R3**
Placă de dezvoltare pentru stația de bază	**Placă de dezvoltare STM32 Nucleo-64 F103Rb**
Placă de dezvoltare pentru fiecare senzor și pentru comunicația acestora (fără fir) cu stația de bază	**Placă de dezvoltare ESP32 CH340C**
LCD LED pentru monitorizarea temperaturii și a umidității din încăpere	**LCD 1602**

Tabelul 3 Componente hardware folosite în realizarea proiectului

5.1 Aplicația server

Aplicația server este construită cu ajutorul NodeJS. Node.js este un mediu de execuție JavaScript și este adoptat masiv în aplicațiile IoT (Internet of Things), deoarece JavaScript este un limbaj de programare rapid, înțeles de majoritatea dezvoltatorilor web și este potrivit pentru crearea unui server pentru o aplicație embedded. [9]

Avantajele framework-ului NodeJS:

- Acesta ajută un dezvoltator web să se concentreze mai mult pe front-end-ul aplicației decât pe partea script-ului de pe server – server-ul fiind destul de ușor de implementat, fiind ajutați și de framework-ul express.js
- Toate straturile și funcțiile conținute de acest framework folosesc același limbaj de programare, JavaScript
- Deoarece serverul de baze de date este implementat pe sistemul dezvoltatorilor, el are control deplin asupra confidențialității și securității acestuia. [10] [11]

Express este cel mai popular framework Node, acesta fiind un cadru pentru aplicații back-end pentru Node.js. Cu ajutorul acestuia se poate crea un server foarte simplu în 3 etape:

- Importarea modulului express pentru crearea unei aplicații de tip Express. Obiectul *app* conține metode care pot fi apelate pentru crearea unor cereri HTTP (HTTP requests).

```
const express = require('express');

const app = express();
```

- Definirea rutei – se folosește metoda app.get() alături de 2 parametri: ruta care este pasată ca un string și o funcție de callback care este invocată mereu când se face un HTTP GET request pe ruta selectată

```
app.get('/temperature-sensor', (req, res) => {

  res.send(dataFromSensor.get("0"));

});
```

- Pornirea server-ului pe un port specificat de dezvoltator și afișarea datelor din funcția anterioară de callback, dacă este cazul

```
app.listen(3000, () =>

  console.log('App listening on port 3000!'),

);
```

Pentru a face posibilă comunicația cu stația principală a componentei hardware (STM32) am apelat la biblioteca *serialport* cu ajutorul căreia am creat un obiect și am pasat parametrii necesari pentru a prelua datele de pe portul serial al laptopului.

```
var port = new SerialPort("COM10", {
  baudRate: 115200,
  dataBits: 8,
  parity: "none",
  stopBits: 1,
  flowControl: false,
});
```

De asemenea, aplicația server comunică cu o bază de date creată în MySQL (Figura 9 Baza de date cu tabelele aferente) folosind instrumentul de proiectare al bazelor de date MySQL Workbench 8.0. Această bază de date conține 2 tabele pentru senzori. Primul tabel este folosit pentru stocarea informațiilor provenite de la senzorii de măsurare – temperatură, umiditate, nivel de gaz în încăpere, luminozitate în cameră. În această tabelă sunt stocate informațiile senzorilor precum: id-ul intrării în baza de date, numele senzorului, câmpurile pe care acesta le măsoară și nu în ultimul rând data și ora la care intrarea în baza de date a fost făcută. Al doilea tabel este folosit pentru a stoca informațiile și comenzile senzorilor IR folosiți în aplicație. Utilizatorul va putea stoca în baza de date comenzile IR ale telecomenzii proprii iar apoi va putea folosi interfața creată (pagina web) pentru a interacționa direct cu dispozitivele IR de pe telefon, fără a mai avea nevoie de telecomanda clasică.

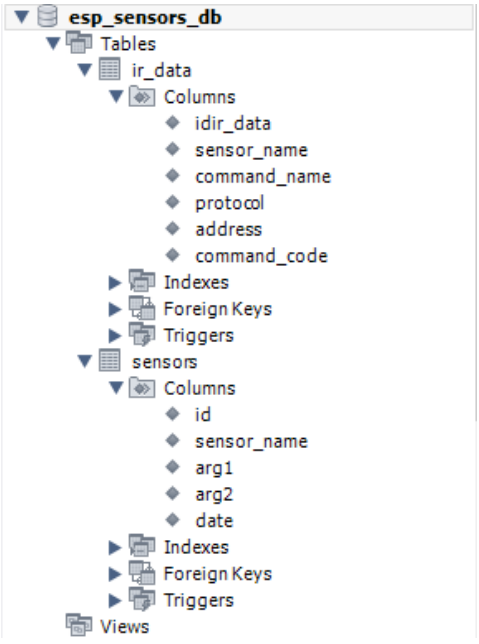

Figura 9 Baza de date cu tabelele aferente

În aplicația server avem și câteva funcționalități ce pot controla baza de date fără a fi nevoie de intervenția utilizatorului direct din MySQL Workbench 8.0. În interiorul server-ului avem câteva rute definite ce controlează anumite funcționalități ale bazei de date:

- Crearea unei noi baze de date
- Crearea unui tabel nou în baza de date
- Inserarea unui rând nou în baza de date
- Ștergerea unui rând din baza de date

Acestea pot fi folosite de către utilizator, oferind acestor rute parametri pentru adăugarea senzorilor și datelor corecte în baza de date. Totuși rutele acestea au fost create pentru a avea date experimentale înainte de conectarea senzorilor la server și conectarea întregii soluții hardware la partea software. Datele se actualizează automat în baza de date și nu este necesară folosirea rutelor amintite anterior, însă pentru un utilizator care vrea să dezvolte aplicația mai mult decât este deja, aceste rute pot fi utile.

5.2 Aplicația web

Interacțiunea cu utilizatorul prezintă un obiectiv important al acestui obiect. Așa cum am amintit în primul capitol, soluția propusă este compusă din cele 3 componente: aplicația server, aplicația web și componenta hardware. În continuare vom prezenta a doua componentă din cele 3 menționate anterior și anume aplicația web. Utilitatea acestei aplicații constă în faptul că utilizatorii vor putea afla informații despre casă și vor putea controla mai multe dispozitive din casa inteligentă alcătuită din întregul sistem.

Tehnologiile folosite pentru dezvoltarea aplicației sunt: Javascript, HTML, CSS. Cele 3 tehnologii lucrează împreună pentru a aduce valoare aplicației finale. HTML este folosit pentru a introduce conținut în pagina web, în timp ce CSS ajută la stilizarea conținutului, făcând aplicația să fie plăcută ochiului uman și, de asemenea, intuitivă la folosire. În timp ce primele 2 tehnologii sunt statice, Javascript aduce un plus aplicației prin faptul că dezvoltă interfața și pe parte dinamică.

Aplicația este formată din două pagini: *Home*, *Room*, ambele încercând să facă din experiența utilizatorului una plăcută. În continuare le vom prezenta pe rând, menționând funcționalitățile fiecăreia și cum se realizează legătura cu aplicația server și cu partea hardware a soluției propuse.

Pagina *Home* (Figura 10 Pagina principală a aplicației web) este formată dintr-un antet care conține un buton ce accesează însăși această pagină. Acest buton cu o iconiță în formă de casă este activ și prezent pe fiecare pagină a aplicației pentru a naviga ușor la pagina de start a interfeței. Pe urmă, în cadranul principal al paginii, avem un fundal reprezentativ pentru aplicația creată și 4 butoane ce reprezintă camerele uzuale într-o casă. Utilizatorul poate accesa oricare din aceste camere apăsând pe butoanele rotunde din centrul paginii.

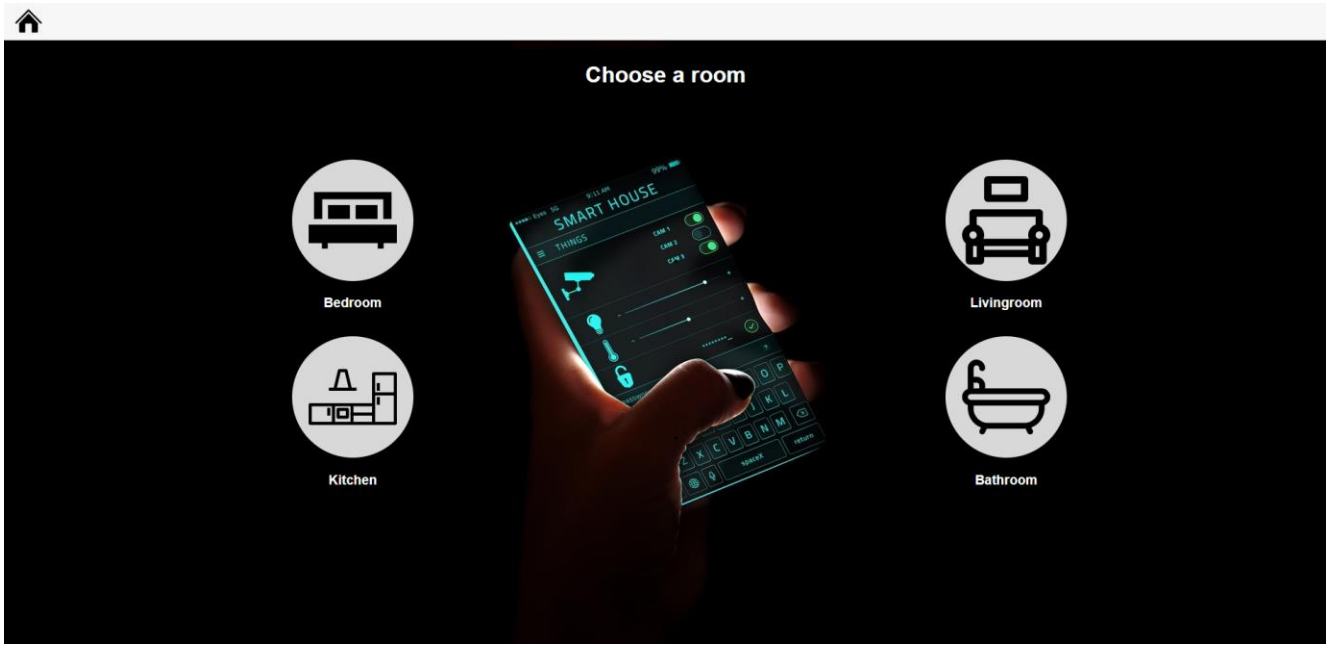

Figura 10 Pagina principală a aplicației web

Navigarea în camera selectată se va face instant iar aplicația va face apelul către a doua pagină creată ce se poate vizualiza (vezi Figura 17 Pagină web completă pentru starea și controlul fiecărei camere). Aceasta este împărțită în 3 categorii. Prima secțiune din această pagină este tot un meniu (Figura 11 Meniul de navigare în diferite camere din pagina *Room*) care poate accesa pagina *Home*, dar pe lângă aceasta, în pagina destinată camerei selectate mai apar încă 4 butoane ce permit navigarea utilizatorului la alte camere fără a se întoarce în pagina de start *Home*. Acestea sunt stilizate cu tehnologia CSS pentru a avea o culoare plăcută ochiului și este oferit și dinamism meniului prin faptul că atunci când cursorul atinge una din opțiuni, aceasta iese în evidență prin sublinierea și colorarea acesteia.

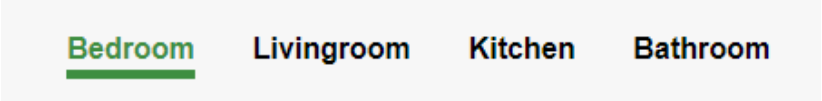

Figura 11 Meniul de navigare în diferite camere din pagina Room

A doua categorie face referire la partea de citire a senzorilor. Secțiunea 2 este una din cele mai importante secțiuni ale proiectului, oferind informații și date de la diferiți senzori ce citesc starea casei inteligente la momentul actual. Se dorește a fi o pagină intuitivă și ușor de folosit, de aceea și această secțiune de afișare a informațiilor este divizată în mai multe părți.

În acest compartiment al paginii utilizatorul poate avea acces la informațiile provenite de la senzorii din camera accesată. Fiecare senzor are diviziunea lui în pagină, informațiile acestora fiind ușor de distins și de interpretat. De asemenea există un titlu cu senzorul care afișează date la fiecare diviziune a paginii. Pe lângă datele provenite de la senzori în format text, utilizatorul poate observa și câteva grafice reprezentative ce reprezintă istoricul pe un timp limitat al unor anumiți senzori (Figura 12 Informațiile curente provenite de la senzorul DHT11 și reprezentarea grafică a istoricului evoluției datelor în încăpere).

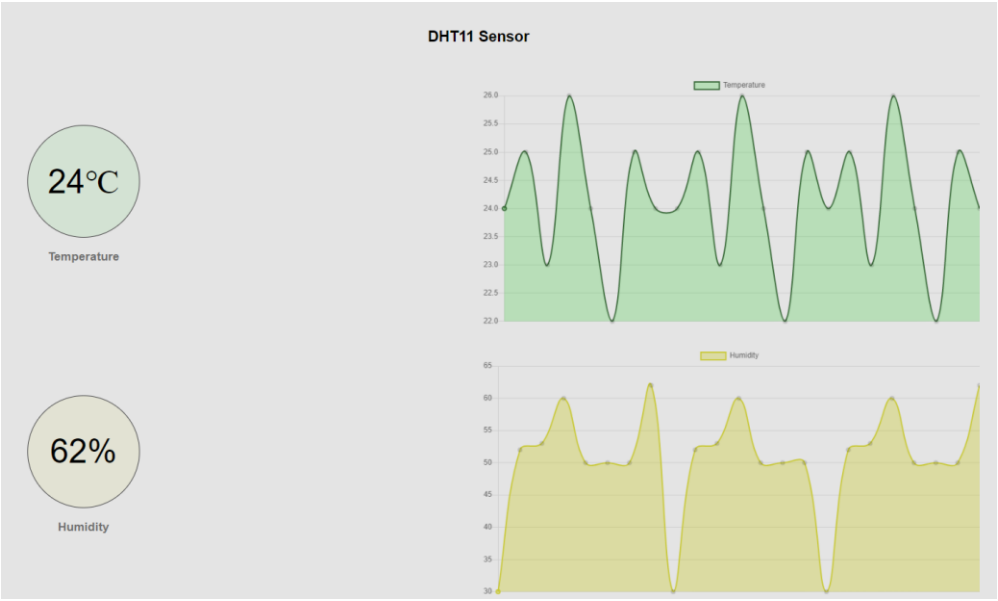

Figura 12 Informațiile curente provenite de la senzorul DHT11 și reprezentarea grafică a istoricului evoluției datelor în încăpere

Datele pe baza cărora se trasează aceste grafice sunt obținute prin apeluri la aplicația server. Pentru fiecare tip de metrică dorit se realizează un apel asincron către un endpoint care conține datele actualizate din baza de date. Un apel asincron inițiază o cerere HTTP către server, însă nu așteaptă ca acesta să răspundă. În schimb, se oferă o metodă care să fie apelată în momentul în care răspunsul server-ului este disponibil. Dacă din anumite motive server-ul nu răspunde în timp util, celelalte funcționalități ale paginii vor fi în continuare disponibile utilizatorului, nealterând funcționarea aplicației.

O a treia categorie creată în pagina curentă este cea care face referire la controlul dinamic pus la dispoziție pentru utilizator. Una din funcționalitățile importante la o casă înteligentă este utilizarea telefonului sau a unui dispozitiv inteligent pentru a controla multiple componente din locuință cum ar fi LED-uri inteligente (Figura 13 Controlul LED-ului din aplicația web). Această secțiune își propune să rezolve o problemă sau o cerință a utilizatorului și anume: „Cum să controlez casa într-un mod inteligent, la o apăsare de pe telefonul mobil?” Acesta poate controla cu ușurință cu ajutorul oricărui dispozitiv cu conexiune la internet mai multe funcționalități conținute de aplicație. Pentru a răspunde la întrebarea adresată anterior, putem spune că utilizatorul va putea folosi cele 2 butoane puse la dispoziție pentru a aprinde sau a stinge un LED.

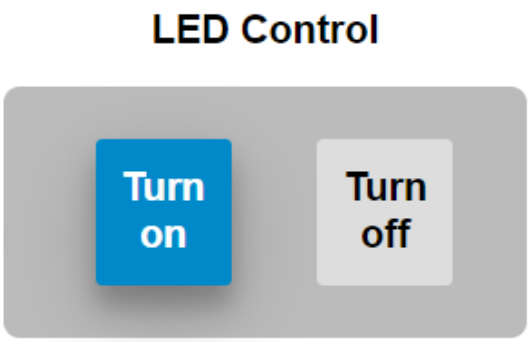

Figura 13 Controlul LED-ului din aplicația web

Această funcționalitate pare simplă, însă este benefică atunci când totul se întâmplă rapid și fără fire. Prin simpla atingere a butonului destinat aprinderii sau stingerii LED-ului, aplicația web trimite o cerere la aplicația server și aceasta deservește rutina prin metode de tip *GET* ce apelează diferite rute definite în aplicația server. La apelul rutei pentru aprinderea LED-ului, aplicația server știe că va trebui să trimită un obiect în format *JSON(JavaScript Object Notation)* pe port-ul USB, iar acest obiect, odată ajuns la plăcuța ESP32 este transmis mai departe plăcuței care este desemnată să aprindă LED-ul. Comunicația între cele 2 componente hardware (cele 2 plăcuțe ESP32) se realizează fără fir prin protocolul de transmisiune al datelor ESP-NOW.

Utilizatorul poate controla un motor pas-cu-pas (Figura 14 Secțiunea de control pentru motorul pas-cu-pas) ce controlează la rândul său o draperie sau o ușă din casa inteligentă. Această funcționalitate este realizată la fel, prin cereri HTTP trimise către aplicația server, iar aceasta la rândul său apelează anumite rute ce conțin metode menite să rezolve problema pusă de utilizator.

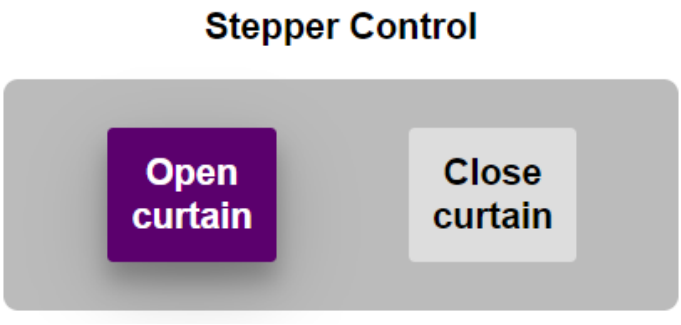

Figura 14 Secțiunea de control pentru motorul pas-cu-pas

Nu în ultimul rând, poate controla butoane ce fac referire la comenzi infraroșu ce pot controla orice dispozitiv care primește astfel de comenzi. Mai jos se poate observa o captură de ecran (Figura 15 Secțiunea de control) în care avem butoanele nedefinite ce pot fi folosite de către utilizator pentru controlul fără fir (direct din aplicația web) al dispozitivelor ce folosesc unde infraroșu.

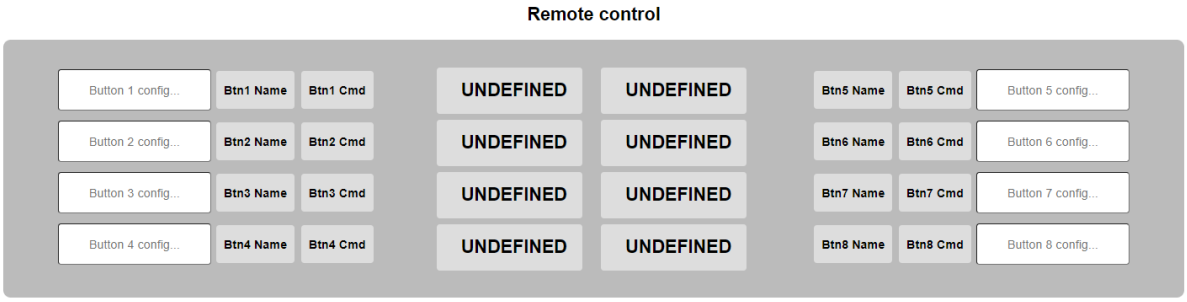

Figura 15 Secțiunea de control pentru telecomanda infaroșu

Butoanele pentru funcțiile de infraroșu sunt fără funcționalitate atunci când aplicația este predată utilizatorului, dar acesta poate crea comenzile dorite de el (Figura 16 Butoane destinate comenzilor infraroșu pentru telecomandă OSRAM) pentru orice telecomandă infraroșu din casa acestuia.

Tot în pagina destinată fiecărei camere, utilizatorul va putea să acceseze o nouă secțiune, editând câmpurile de tip *text* atribuite fiecărui buton și apăsând pe butonul *BtnX Name* de lângă fiecare buton nedefinit. Astfel se vor stoca în baza de date numele comenzilor infraroșu și butoanele se vor configura conform cererii utilizatorului.

Acesta va putea introduce de la tastatură un nume pentru comanda dorită (ex: Aprinde TV), va apăsa butonul de „BtnX Name” și apoi va apăsa butonul de „BtnX Cmd” pentru a putea stoca comanda infraroșu atribuită butonului *X* în baza de date. Utilizatorul va apăsa butonul „Aprinde TV” de pe telecomanda fizică îndreptată către senzorul IR pentru recepția datelor. În urma acestor operațiuni, utilizatorul se poate folosi de noul buton atribuit funcției dorite de acesta.

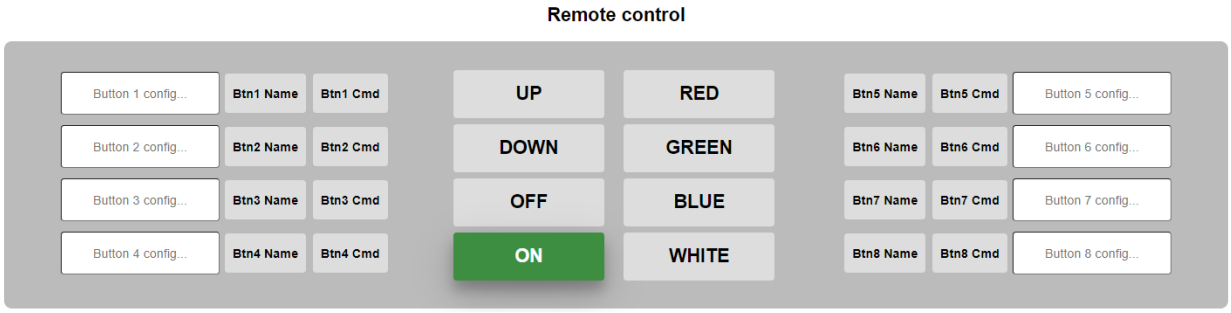

Figura 16 Butoane destinate comenzilor infraroșu pentru telecomandă OSRAM

Figura 17 Pagină web completă pentru starea și controlul fiecărei camere

5.3 Soluția hardware

Din punct de vedere hardware, prototipul propus este format din mai multe componente. Acesta va fi construit dintr-o stație de bază – un kit de dezvoltare Nucleo-64, senzori pentru măsurarea luminozității din încăpere (KY-018), măsurarea temperaturii și a umidității (ambele putând fi preluate de la un singur senzor – DHT11), măsurarea nivelului de gaz din încăpere (MQ-2).

Pe lângă senzorii care transmit date despre starea casei, avem și componente hardware care ajută la automatizarea locuinței – senzor PIR pentru facilitarea aprinderii luminii la detectarea prezenței, senzori IR care detectează prezența unor unde infraroșu, iar sistemul poate învăța coduri de la telecomanda TV/AC și poate transmite mai departe acele coduri direct din aplicația Web, motoare stepper care ajută la închiderea/deschiderea ușilor/ferestrelor în cazul în care utilizatorii vin/pleacă de acasă, sau se detectează o cantitate semnificativă de gaz sau de fum.

Proiectul de față își propune crearea unui sistem bine definit ce poate duce la stabilitatea psihică din punct de vedere al stresului și al neliniștii utilizatorilor cu privire la propria casă. Acest proiect este menit să aducă în casa fiecărui om un sistem inteligent de control al locuinței și o aplicație în care utilizatorul să poată observa în timp real starea casei oferită de senzorii în cauză.

Soluția hardware este una simplă, care necesită costuri reduse de implementare și nu implică schimbări majore în planul casei sau în arhitectura pereților casei. Această soluție se poate implementa într-un timp relativ scurt și poate oferi informații prin comunicație fără fir de la senzori la server.

Componentele hardware au fost gândite în așa fel încât utilizatorii doar să le pornească și acestea să ofere deja informații și date sistemului software. Soluția propusă este ușor scalabilă și poate fi implementată în fiecare încăpere din casă cu ușurință, fără a fi nevoie de mult prea multe schimbări și configurări.

Prin schema bloc prezentată la începutul acestui capitol putem vedea cum fiecare componentă este legată în sistem și comunică fără fir prin protocolul de comunicație creat de Espressif – ESP-NOW.

Cu ajutorul mai multor plăcuțe de dezvoltare ESP32 este posibilă comunicația fără fir între senzori și stația principală (STM32) și astfel este o soluție ușor de implementat care nu necesită tragerea firelor prin casă. Prin faptul că avem și un LCD legat la plăcuța mamă principală a sistemului, se pot observa cu ușurință anumite informații și date provenite de la senzori cum ar fi temperatura sau umiditatea din încapere.

În continuare vom prezenta fiecare componenta a schemei bloc în parte:

5.3.1 Kit-uri de dezvoltare folosite

5.3.1.1 STM32 Nucleo-64 F103

Plăcuța STM32 Nucleo (Figura 18 NUCLEO-F103RB) este o variantă ieftină de dezvoltare a unor proiecte și are o platformă ușor de folosit, de codat și de înțeles. Alimentarea acesteia cu putere poate fi oferită fie din partea calculatorului personal, folosind portul USB, fie de către o sursă de tensiune externă.

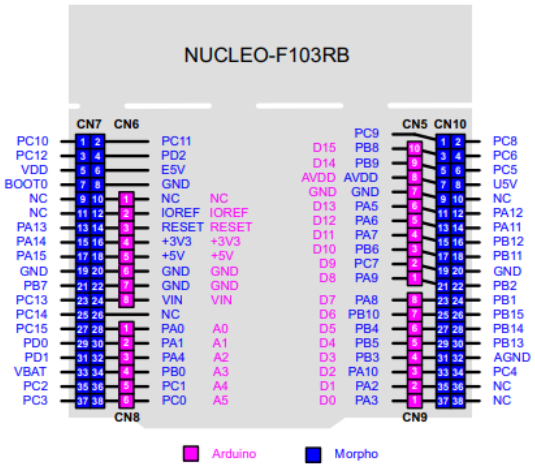

Figura 18 NUCLEO-F103RB

Aceasta este folosită în proiectul de față ca fiind stația principală prin care trec toate informațiile și datele. Nucleo-64 face posibilă comunicația server-ului cu plăcuțele ESP32 printr-o comunicație serială. Aceasta comunică prin interfața USB cu server-ul care este găzduit pe calculatorul personal și transmite datele către baza de date. În același timp, plăcuța de dezvoltare Nucleo-64 face posibilă comunicația cu *controler*-ul ESP32 prin interfața UART. De asemenea, este conectat la aceasta și un LCD pe care se pot citi diferite informații despre starea casei cum ar fi temperatura și umiditatea din încăpere. Protocolul de transmisie a datelor UART (*Universal Asynchronous Receiver-Transmitter*) este folosit de aproape toate microcontrolerele și toate calculatoarele.

Acesta folosește o interfață de comunicație serială care poate transmite și poate recepționa date. Pachetele de date trimise prin acest protocol sunt formate din biți de start și de stop, din biți de date și opțional din biți de paritate care pot fi folosiți pentru detectarea erorilor la transmisia de date. Majoritatea interfețelor UART folosesc 8 biți de date. [12]

Comunicația în acest protocol poate fi de 3 tipuri:

- *simplex* – transmisia se realizează doar într-o singură direcție, fără ca receptorul să poată transmite date înapoi către transmițător
- *full duplex* – ambele dispozitive (și receptorul și transmițătorul) pot transmite și primi date în același timp
- *half duplex* – dispozitivele transmit și primesc date pe rând

5.3.1.2 ESP32 Lolin32

ESP32 (Figura 19 ESP32 Pinout) este un microcontroler puternic SoC (*System on Chip*) cu Wi-Fi 802.11 b/g/n integrat, versiunea Bluetooth 4.2 și o varietate de periferice. Este un succesor avansat al cipului ESP-8266 în primul rând prin implementarea a două nuclee în versiuni diferite de până la 240 MHz. În comparație cu predecesorul său, cu excepția acestor caracteristici, extinde și numărul de pini GPIO (*General Purpose Input Output*) de la 17 la 36, numărul de canale PWM (*Pulse-width Modulation*) la 16 și este echipat cu 4MB de memorie flash.

Microcontrolerele se conectează de obicei cu module IoT și alți senzori inteligenți și oferă date sistemului superior. Plăcuța de dezvoltare folosită în proiect Lolin32 este dezvoltată de firma Wemos și este perfectă pentru a fi folosită în proiecte IoT. Aceasta are un consum redus de putere, dimensiuni reduse și poate integra o multitudine de senzori. Comunicația folosită este ESP-NOW.

ESP-NOW este un protocol de comunicație fără fir care poate face transmisiune de date *peer-to-peer* sau *broadcast*. Acesta este un protocol care necesită un consum redus de putere și poate fi folosit alături de tehnologia *deep-sleep*. Împreună pot reduce consumul de putere semnificativ până la aproximativ 120mA în timpul transmisiei și 0.1mA în timp ce sistemul se află în *deep-sleep*. [13]

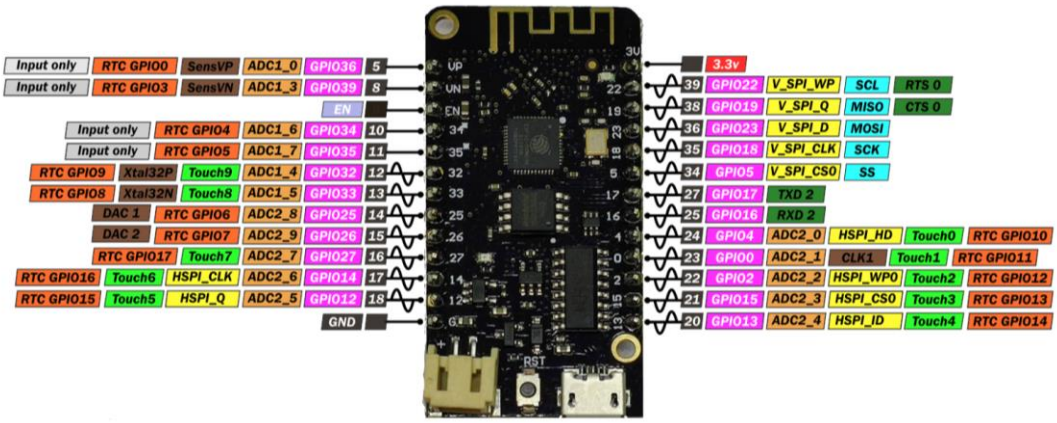

Figura 19 ESP32 Pinout

În proiectul propus s-a ales folosirea plăcuțelor de dezvoltare ESP32 deoarece sunt evoluate ca tehnologie, permit conectarea mai multor senzori la pinii disponibili și sunt dotate cu protocoale de comunicație fără fir. În schema bloc, folosim mai multe astfel de plăcuțe ce conțin diferiți senzori, iar toate acestea comunică fără fir cu stația de bază ce preia informațiile provenite de la plăcuțele *agent* și le transmite server-ului.

5.3.2 Senzori folosiți

5.3.2.1 Senzor de temperatură și umiditate

Senzorul de temperatură și umiditate DHT11 este unul complex cu o ieșire de semnal digital care este calibrată încă din fabrică. Acesta asigură o fiabilitate ridicată și stabilitate excelentă pe termen lung. Senzorul include o componentă de tip rezistiv ce măsoară umiditatea și o componentă NTC pentru a măsura temperatura. Acesta se conectează la un microcontroler (Figura 20 Conectarea senzorului DHT11 la microcontroler) pe 8 biți oferind astfel o calitate excelentă a măsurătorilor, un timp de răspuns redus și de asemenea un cost redus pentru ce oferă comparativ cu produsele de pe piață.

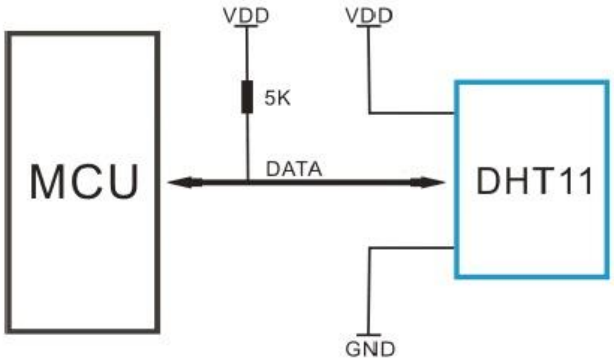

Figura 20 Conectarea senzorului DHT11 la microcontroler

Sursa de tensiune la care se poate alimenta senzorul DHT11 se află între 3 și 5.5V. Când se alimentează senzorul este recomandat ca timp de o secundă să nu se ceară date de la acesta pentru a nu primi o stare instabilă de la inițializare. Procesul de comunicare este pe magistrala simplă și este folosit pentru sincronizarea și comunicația dintre microcontroler și senzor. Un pachet de date complet se transmite în aproximativ 4ms și conține 40 biți de informație.

Când microcontroler-ul trimite un semnal puls de start, senzorul DHT11 își schimbă starea din modul *low-power* în modul *running*, așteptând ca microcontroler-ul să termine pulsul de start și acesta să înceapă transmisia celor 40 biți (Figura 21 Comunicația MCU - DHT11) în care avem incluse informațiile despre temperatură și umiditate. Utilizatorul poate alege ce date preia de la senzor: fie umiditate, fie temperatură, fie ambele. Odată ce datele utile sunt transmise, senzorul intră din nou în modul *low-power* și așteaptă un nou semnal de start de la microcontroler. [14]

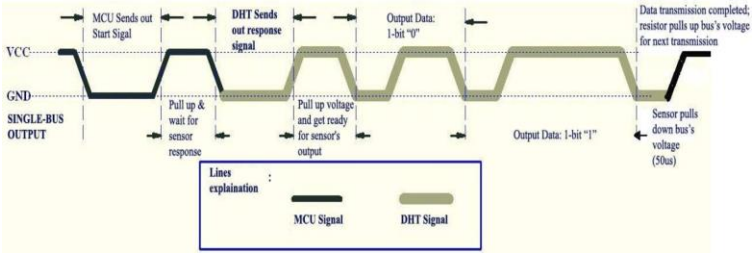

Figura 21 Comunicația MCU - DHT11

5.3.2.2 Modulul PIR HC-SR501 (Passive Infrared Sensor)

Acesta se folosește la detecția mișcării unei persoane aflate în raza senzorului. Este un senzor de dimensiuni reduse, ușor de folosit, redus la capitolele costuri și consum de putere. Acesta este folosit des în aplicații de case inteligente sau chiar în companii pentru a reduce consumul de energie electrică.

Componenta hardware (Figura 22 Schema modulului PIR HC-SR501) folosește o ieșire digitală (3.3V) care este „true” atunci când este detectată mișcare de către senzor și este „false” când nu se detectează mișcare. Marja de senzitivitate a senzorului poate ajunge până la 7 metri, >120 grade, iar timpul de întârziere poate fi ajustat între 0.8 și 18 secunde. [15]

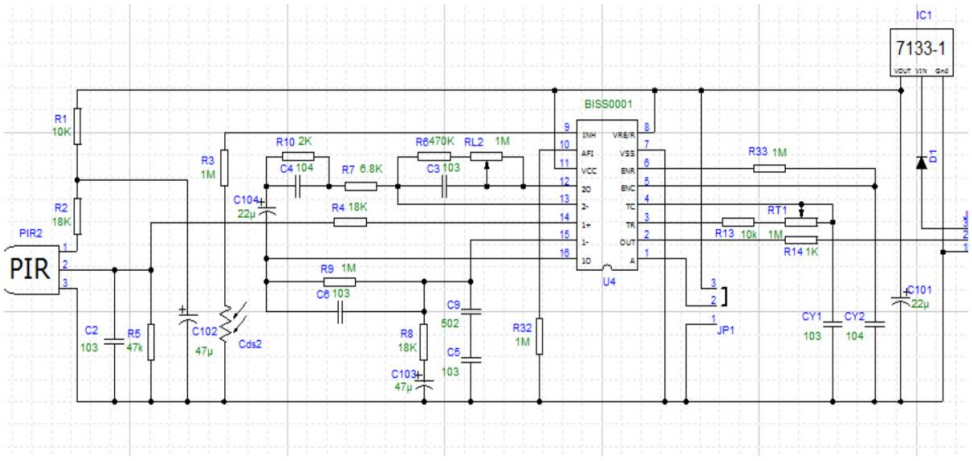

Figura 22 Schema modulului PIR HC-SR501

Tensiunea de alimentare a modulului este între 3.4 și 20V dar, cu ajutorul unui regulator pentru reducere a tensiunii (HT7133) montat pe modulul senzorului, necesar pentru electronica modulului, tensiunea de alimentare ajunge la 3.3V. Senzorul PIR are în componența sa 2 sloturi ce conțin materiale diferite, ambele fiind sensibile la infraroșu (Figura 23 Funcționarea senzorului PIR). Atunci când sezorul este în repaus și nu detectează nimic, înseamnă că cele 2 componente primesc aceeași cantitate de unde infraroșu. În momentul în care senzorul detectează mișcarea și temperatura corpului uman în raza de mișcare se trimite un puls și astfel se transmite semnalul de ieșire către MCU.

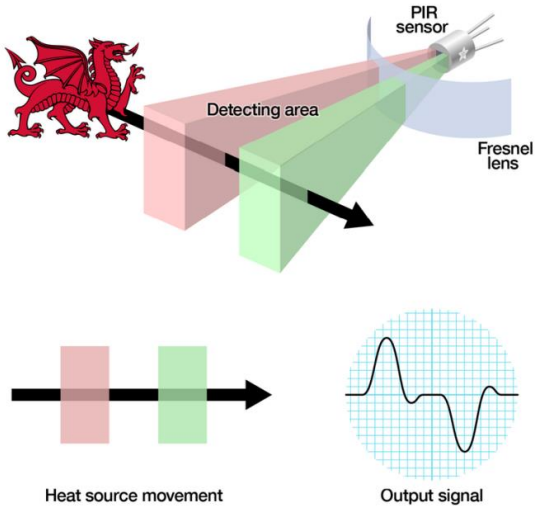

Figura 23 Funcționarea senzorului PIR

5.3.2.3 Senzorul de gaz MQ – 2

Scurgerile de gaze sunt una dintre marile probleme din sectorul industrial, dar nu numai. Putem avea probleme majore și în propria casă dacă avem scurgeri de gaz. Una dintre soluțiile de reducere a pierderilor de gaz este detectarea precoce atunci când acestea apar. Există deja multe tehnologii care ajută la prevenirea daunelor provocate de către scurgerile de gaze, una dintre ele fiind senzorul MQ – 2. Acesta poate detecta mai multe tipuri de gaze (Figura 24 Caracteristica de senzitivitate a senzorului MQ - 2), iar pentru fiecare tip de gaz are o senzitivitate și poate detecta cantități de gaz de la 100 la 10000 ppm (părți pe milion). [16]

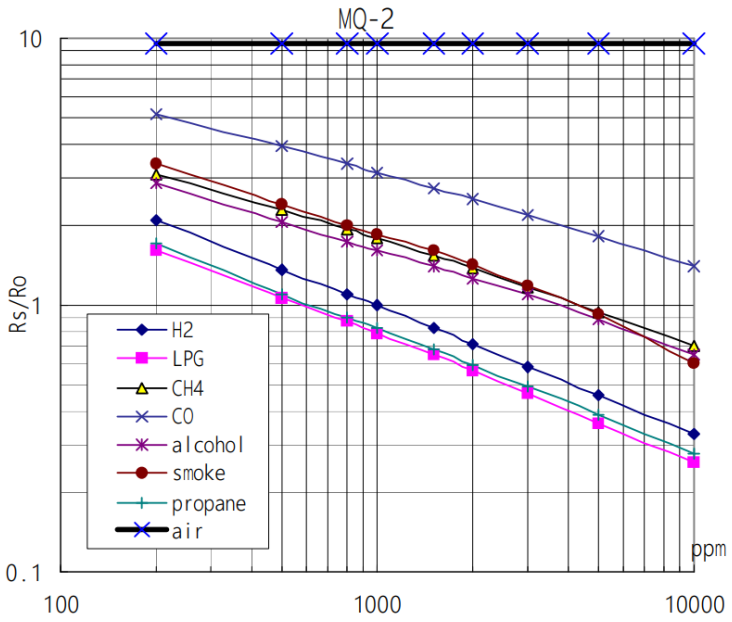

Figura 24 Caracteristica de senzitivitate a senzorului MQ - 2

Folosind un senzor de gaz de tip MQ – 2 ca detector, este de așteptat ca scurgerile de gaz să fie detectate în timp util, ulterior neavând impact mai larg asupra oamenilor. Senzorul MQ – 2 a fost ales deoarece are un preț scăzut (~20 RON) și nu necesită configurare la fiecare alimentare a acestuia cu tensiune. MQ-2 este calibrat și configurat folosind limbajul C, care este implementat prin Arduino IDE. După configurarea senzorului de gaz, este de așteptat ca rezultatele de precizie să ajungă la 80% cu distanța de la senzorul de gaz până la punctul de scurgere de aproximativ 0-10 cm. [17]

În urma configurării, rezistența calculată R0 are valoarea între 0.14-0.30 kOhm și cantitatea de *LPG* (*gaz petrolier lichefiat*) detectată în aer curat este de aproximativ 35 ppm (ceea ce este în regulă; cantitatea de LPG care poate dăuna se poate găsi de la 200 ppm în sus). Atunci când senzorul detectează o cantitate mai mare de 300 ppm, utilizatorul poate observa înroșirea valorii detectate pe pagina destinată camerei curente în dreptul valorii detectate de senzorul de gaz.

5.3.2.4 Modulul cu fotorezistor KY-018

KY-018 este un modul cu ieșire de tip analog, și se poate citi informația oferită de acesta ca voltaj al pinului de semnal atribuit senzorului (Figura 25 Senzorul de lumină KY-018). Acesta poate fi alimentat de la 3.3V la 5V, depinde de aplicația la care este întrebuințat și de sursa de tensiune folosită. Senzorul transmite o tensiune calculată în punctul dintre rezistența plasată pe modulul acestuia și fotorezistor (divizor de tensiune), iar placa de dezvoltare citește această tensiune și o transformă într-o valoare pe 12 biți. Astfel valoarea citită de pinul analog se va găsi între 0 și 4095. [18]

Se poate calcula rezistența cu ajutorul voltajului măsurat pe pinul analog. În funcție de valoarea de prag pe care o setăm din programul software, când valoarea rezistorului ajunge mai jos de acesta, se declanșează un semnal prin care putem aprinde LED-ul și astfel putem avea control asupra luminilor din casă.

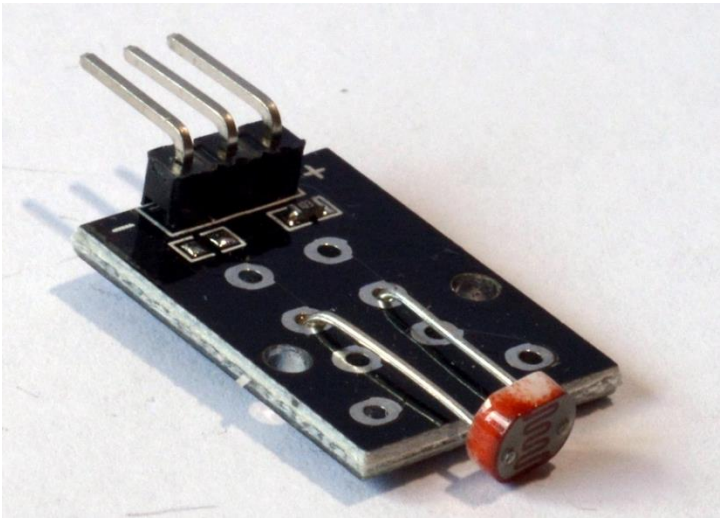

Figura 25 Senzorul de lumină KY-018

Acest modul deja are integrat în componența sa hardware un rezistor variabil SMD R103 (10 kOhm) care își poate modifica rezistența în funcție de nivelul de luminozitate din încăpere. Astfel, dacă luminozitatea este mare, fotorezistorul are rezistența mare, și aproape toată tensiunea cade pe fotorezistor. Cu cât rezistența fotorezistorului este mai mică, cu atât lumina din cameră este la un nivel redus, și astfel întreaga tensiune din punctul de măsurare cade pe rezistorul atașat pe modul (cel de 10 kOhm).

În aplicația propusă modulul KY-018 ajută la controlul unui LED. Acesta este programat să declanșeze un semnal atunci când luminozitatea este mai mică decât PRAG, iar în acel moment, LED-ul se aprinde, știind că luminozitatea este scăzută în acea cameră.

5.3.2.5 Motorul pas-cu-pas

Motorul pas-cu-pas poate fi văzut ca un motor sincron AC dar care are numărul de poli (și pe rotor și pe stator) mai mare (Figura 26 Funcționarea unui motor pas-cu-pas bipolar). Acesta este caracterizat de mai mulți parametri printre care se numără:

- Voltajul: Motorul folosit în aplicația curentă lucrează între 5V și 12V. În funcție de tensiunea de alimentare la care este supus motorul pas-cu-pas, acesta poate produce o putere de rotație mai mare sau mai mică.

- Rezistența: „Rezistența la rotație este o altă caracteristică a unui motor pas-cu-pas. Această rezistență va determina intensitatea curentului motorului, dar va afecta și curba cuplului motorului și viteza maximă de funcționare.” [19]

- Grade pe fiecare pas: Aceasta este probabil cea mai importantă caracteristică atunci când se alege un motor pas-cu-pas. Acest număr reprezintă numărul de grade cu care se va roti tija motorului și astfel putem calcula de câte rotații ale tijei este nevoie pentru a acoperi o rotație completă de 360 grade.

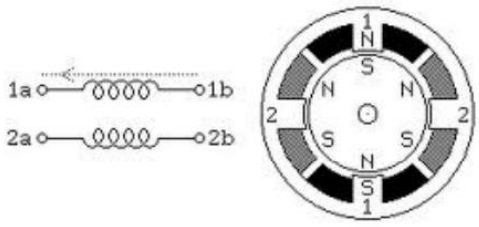

Figura 26 Funcționarea unui motor pas-cu-pas bipolar

În aplicația propusă motorul pas-cu-pas ajută la controlul draperiilor de la ferestrele casei inteligente. Acesta este programat să declanșeze un semnal atunci când utilizatorul apasă pe butonul „Open curtain” sau „Close curtain”, iar în acel moment, motorul pas-cu-pas începe să deschidă sau să închidă, după caz, draperia de la fereastră.

5.3.2.6 Senzorul IR transmițător și IR receptor

Modulul folosit pentru recepția semnalelor infraroșu este KY-022. Acesta are în componența sa un rezistor de 1kOhm și un LED de culoare roșie (Figura 27 Schema modulului KY-022) ce se aprinde în momentul în care se face recepția unei comenzi infraroșu. Rezistorul de 1 kOhm este folosit pentru a reduce curentul care circulă în interiorul modulului pentru a evita distrugerea senzorului infraroșu. [20]

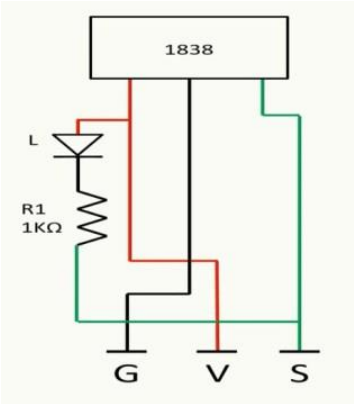

Figura 27 Schema modulului KY-022

Modulul KY-022 se bazează pe senzorul infraroșu VS1838. Acesta se poate alimenta cu o sursă de tensiune între 2.1 și 5.5V, iar fiecare puls primit de la unda infraroșu se află între 400 și 800 microsecunde.

Modulul folosit pentru transmisia semnalelor infraroșu este KY-05. Acesta conține un LED infraroșu de 5mm. Tensiunea de alimentare este între 3.3 și 5V, lungimea de undă la care transmite modulul este de 940nm (aceasta fiind complet invizibilă – vezi Figura 28 Percepția ochiului uman asupra lungimii de undă) și consumul de putere este de aproximativ 90mW.

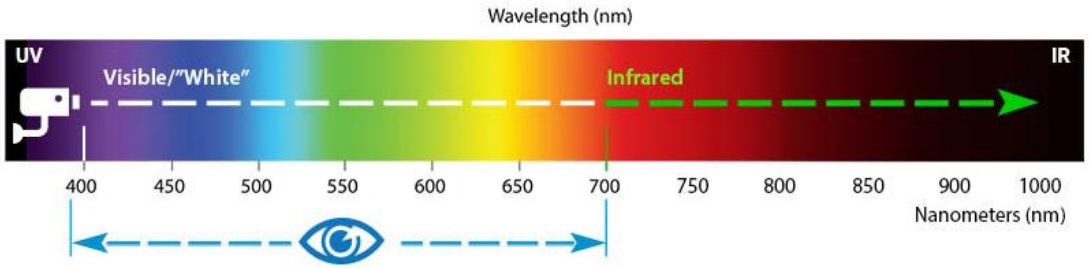

Figura 28 Percepția ochiului uman asupra lungimii de undă

În concluzie, putem spune că soluția hardware îndeplinește cerințele proiectului propus în starea incipientă, produsul finit îndeplinind tema lucrării. Avem un sistem embedded bazat pe un microcontroler STM ce comunică prin interfața serială (*USB*) cu server-ul. Aplicația server este găzduită de calculatorul personal al utilizatorului și astfel datele transmise de microcontrolerul STM ajung să fie stocate în baza de date cu ajutorul rutelor create în interiorul server-ului.

Plăcuța de dezvoltare STM32 Nucleo-64 comunică și cu un LCD LED interfațat pentru ca utilizatorul să poată afla informații utile despre starea casei cum ar fi temperatura din încăpere, nivelul de gaz în ppm sau starea senzorului de mișcare PIR.

Alături de microcontrolerul de bază care ține evidența tuturor datelor ce provin de la sistemul hardware, este și o placă de dezvoltare ESP32 care ține loc de *controler* în schema noastră propusă (un *controler* și mai mulți *agenți*). Aceasta din urmă este legată prin interfața UART la microcontrolerul STM și transmite fiecare informație provenită de la celelalte plăcuțe ESP32.

Fiecare senzor este conectat la propria placă de dezvoltare ESP32, aceasta fiind soluția propusă pentru a avea o conexiune exclusiv fără fir între toți senzorii și stația principală (stația de bază). Comunicația între fiecare *agent* și *controler*-ul de bază se face prin ESP-NOW, protocol de comunicație fără fir conceput de Espressif. Cu ajutorul acestui protocol, este nevoie doar de aflarea adresei *MAC(media access control address)* a fiecărei plăcuțe din schema bloc, și cu ajutorul bibliotecilor dedicate din mediul de dezvoltare Arduino IDE se poate face comunicația în mod asincron.

6 EVALUAREA REZULTATELOR

În acest capitol se vor evalua funcționalitățile aplicației pe rând, menționând, acolo unde este cazul, ce funcționează și ce nu, conform așteptărilor (Tabelul 4 Așteptări versus realizări cu privire la proiectul de față), urmând ca în capitolul următor să prezentăm pașii următori de dezvoltare, ce s-ar putea îmbunătăți la aplicație și cum se poate continua proiectul în direcția dezvoltării tehnologiei integrate în lucrare.

Din punct de vedere al interfeței cu utilizatorul, putem spune că aplicația web s-a ridicat la așteptările inițiale: interfața este ușor de utilizat și de înțeles, are o grafică simplă, curată și elementele componente ale vizualului sunt așezate bine în paginile componente ale aplicației web. Aceasta conține două pagini (*Home, Room*), iar pentru fiecare cameră se creează o nouă pagină de tip *Room*. Toate aceste interfețe sunt create pentru a face din interacțiunea cu aplicația web una ușoară, intuitivă și doresc să aducă plusul de care este nevoie într-o aplicație de tipul „casă inteligentă”. Aplicația poate fi asimilată rapid în viața de zi cu zi a utilizatorilor deoarece oferă integrarea componentelor hardware într-un mod simplu, afișând pe ecranul utilizatorului datele necesare cunoașterii stării casei inteligente și a senzorilor din componența camerei.

Din punct de vedere al informațiilor disponibile utilizatorilor, rezultatele sunt satisfăcătoare, însă se pot îmbunătăți anumite aspecte. Așa cum am prezentat la capitolul „Soluția propusă”, interacțiunea utilizatorului cu aplicația se face exclusiv din aplicația web. Acesta poate vizualiza datele provenite de la senzori și poate controla mai multe dispozitive din casă precum LED-uri, motoare pas-cu-pas sau chiar dispozitive ce primesc comenzi infraroșu, utilizatorul putând fi capabil să configureze butoanele din aplicație destinate pentru controlul telecomenzii infraroșu.

Limitarea acestor funcționalități vine din faptul ca utilizatorul trebuie sa fie în aplicația web pentru a afla informații sau pentru a vedea nivelul de gaz din cameră. Acesta va observa doar afișarea unui mesaj de tip text care informează utilizatorul că senzorul de gaz a detectat o valoare care este mai mare decât pragul setat inițial (Figura 29 Notificare în scris pentru valoarea citită de senzorul de gaz). Încă nu a fost implementată o funcție ce poate transmite notificări pe telefonul mobil atunci când senzorul de gaz detectează o cantitate de ppm mai mare decât pragul impus prin programul software.

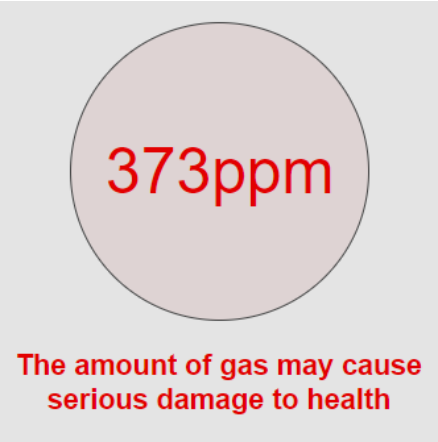

Figura 29 Notificare în scris pentru valoarea citită de senzorul de gaz

Funcționalitatea cea mai importantă a acestei aplicații se află în pagina *Room*, în a treia secțiune, cea care poate controla dispozitivele din casă fără fir. Rezultatele obținute în această pagină sunt foarte satisfăcătoare și ne motivează să dezvoltăm acest produs. Ceea ce aduce în plus față de soluțiile existente este posibilitatea utilizatorului de a introduce propriile comenzi infraroșu la orice telecomandă din casă, numind butoanele după propriul plac și folosind astfel un sistem inteligent pentru controlul dispozitivelor fără fir din casa inteligentă cu un cost redus.

În ceea ce privește adaptarea aplicației web la telefonul mobil, comportamentul paginii este la fel ca pe un ecran mare de computer. Putem spune că acesta este un minus al aplicației, deoarece aceasta nu se adaptează la mărimea ecranului și la rezoluția pe care aplicația este proiectată. Utilizatorul trebuie să se folosească de opțiunea de *zoom* a telefonului pentru a naviga cu ușurință prin paginile create în aplicație. La propunerea inițială această cerință nu era necesară, însă este un obiectiv care ar trebui atins în următoarea perioadă pentru ca aplicația web să fie mai independentă și cu un răspuns mai bun oferit utilizatorilor.

În ceea ce privește baza de date, am obținut o arhitectură performantă și un flux de date eficient. Datele sunt introduse în baza de date de fiecare dată când un senzor transmite pachetul *JSON* la stația de bază. Aceste date sunt păstrate 2 ore în baza de date, urmând ca după cele 2 ore să fie șterse (datele mai vechi de 2 ore nu mai sunt relevante pentru scopul aplicației). Astfel, baza de date rămâne eficientă și rapidă, menținând întotdeauna un număr optim de intrări în tabele.

Din punct de vedere hardware, consider că soluția propusă obține rezultatele dorite în starea incipientă a propunerii proiectului. Fiecare senzor transmite datele fără fir către stația de bază, acestea fiind procesate și transmise pe cale serială către server. Comunicația este una rapidă, și soluția hardware este una scalabilă, protocolul ESP-NOW permițând să se conecteze până la 20 de *agenți* la un *controler*. Aceasta poate fi o limitare, în caz că soluția propusă se dorește a fi folosită în case inteligente mai mari. În acest sens, se vor folosi mai multe stații de bază ce vor fi legate printr-un *hub* la același server.

Aplicația server s-a dovedit o aplicație puternică, scalabilă și una rapidă. Când se inserează date în baza de date se utilizează o singură rută, astfel că putem adăuga oricând alte tipuri de senzor și această operațiune nu va îngreuna în niciun fel complexitatea aplicației server gândite de la început. Comunicația cu stația de bază care se face prin portul serial este una sigură și nu pot fi interferențe, astfel datele provenite de la senzori nu sunt alterate de factori externi.

Ce așteptări au fost în fază incipientă	Ce s-a realizat la finalul proiectului
Sistem embedded	DA, sitemul creat este unul embedded
Sistem bazat pe microcontroler Infineon sau STM cu display	DA, sistemul este bazat pe un microcontroler STM32 Nucleo-64
Sistem capabil să învețe coduri de la telecomanda IR	DA, sistemul este capabil să învețe coduri de la telecomanda IR
Sistem capabil să reproducă la cerere coduri de la telecomanda IR	DA, sistemul este capabil să reproducă coduri de la telecomanda IR prin folosirea butoanelor dedicate ale interfeței utilizatorului
Sistem capabil să citească temperaturi de la senzori fără fir	DA, sistemul este capabil să citească temperatura de la modulul DHT11 ce transmite (fără fir) temperatura către stația de bază
Sistemul poate aprinde/stinge becuri conform detecției prezenței/ușilor deschise	DA, sistemul este capabil să aprinda LED-uri conform detecției prezenței unei persoane în dreptul senzorului de mișcare
Dacă se va folosi mediul Arduino de dezvoltare se vor scrie și bibliotecile necesare	S-a folosit mediul de dezvoltare Arduino, dar fiecare senzor a avut deja biblioteca scrisă pentru funcțiile acestora. S-a scris o clasă de bază ce conține tipurile de senzor enumerate și metodele apelate de aceștia, iar pentru fiecare senzor și modul în parte s-a dezvoltat propriul program software

Tabelul 4 Așteptări versus realizări cu privire la proiectul de față

7 CONCLUZII

În acest capitol vom prezenta obiectivele pe care le-am avut, ce am obținut și cum putem evalua produsul. După cum am amintit și în primele capitole, obiectivul acestui proiect a fost crearea unei soluții embedded pentru monitorizarea stării unei locuințe, și de asemenea controlul anumitor dispozitive. Soluția implementată a trebuit să fie simplă, ușor de folosit de către utilizatori, intuitivă și scalabilă.

Am reușit să dezvoltăm o aplicație server rapidă, scalabilă, capabilă să colecteze date de la stația de bază, aceasta fiind conectată fără fir la toate plăcuțele ESP32 ce conțin senzori. Aplicația este capabilă să colecteze aceste date în mod automat, la fiecare moment de timp în care stația de bază transmite pe conexiunea serială pachete de tip *JSON* provenite de la senzori. Am creat diferite endpoint-uri pentru accesarea datelor colectate după anumite criterii, și aplicația este capabilă să introducă în baza de date și momentul de timp la care sunt introduse datele de la senzor în aceasta. Această funcționalitate ne ajută la crearea unor grafice în timp real, ușor de urmărit și de înțeles de către utilizatori.

Am obținut și un flux de date eficient la nivelul bazei de date prin limitarea timpului maxim de 2 ore în care se pot stoca datele de la senzori și acestea rămân în istoric. Am luat această hotărâre deoarece senzorii transmit date o dată la 30 de secunde și datele mai vechi de 2 ore nu mai sunt relevante pentru folosirea zilnică a aplicației de către utilizator. Astfel, prin eliminarea unor date mai vechi de 2 ore din baza de date, s-a obținut un timp mai rapid de răspuns către server și baza de date rămâne mai eficientă fiind încărcată la nivelul minim.

În ceea ce privește soluția hardware, am creat o arhitectură scalabilă și eficientă. Rețeaua de senzori poate fi amplasată oriunde în interiorul casei, în orice cameră, fiecare senzor având propria lui placă de dezvoltare ESP32 și comunicarea între senzori și stația de bază fiind fără fir. Comunicarea prin protocolul ESP-NOW se realizează la o distanță de până la 220 metri în linie dreaptă, fără obstacole. Totuși aceste date sunt strict experimentale. Este de dorit ca aplicația propusă să nu fie supusă la astfel de distanțe deoarece se pot pierde date pe parcurs. Fiecare componentă hardware trimite asincron datele către stația de bază și astfel plăcuțele de tip *agent* nu depind de nimic altceva decât de momentul de timp la care senzorul captează datele din exterior.

În cele ce urmează vom prezenta câteva idei pentru dezvoltarea ulterioară. Dat fiind faptul că aplicația are un potențial major pentru o casă inteligentă, există multe modalități de integrare ulterioară și dezvoltare de noi funcționalități.

Un prim pas necesar a fost deja amintit în capitolele anterioare: dezvoltarea unui sistem de notificări prin care utilizatorul să poată fi avertizat în timp real prin email/mesaj cu privire la starea senzorilor, mai ales la starea celor care au trecut de pragurile normale de funcționare (senzorul de gaz poate detecta o cantitate mai mare de LPG și aceasta poate dăuna grav sănătății).

Un al doilea pas ulterior va fi integrarea soluției propuse cu un asistent vocal (*Amazon Alexa* sau *Google Assistant*) ce va aduce un mare plus aplicației, utilizatorii nemaifiind nevoiți să folosească aplicația web pentru a afla informații din casa inteligentă sau pentru a controla anumite dispozitive. Prin simpla comandă vocală, acest asistent va face treaba în locul utilizatorului, iar sistemul hardware va răspunde în mod rapid și corespunzător.

De asemenea, o altă funcționalitate ce trebuie implementată în acest sistem este reprezentată de crearea unor scene ce semnifică evenimente sau întâmplări din viața de zi cu zi, iar la fiecare eveniment descris, aplicația să reacționeze într-un anumit fel. De exemplu, dacă utilizatorul dorește să își seteze o scenă de film, sistemul inteligent va putea reduce lumina la un anumit procent în casă, va trage draperiile și va porni televizorul. Sau dacă utilizatorul va dori să plece de acasă, acesta va avea posibilitatea de a stinge fiecare dispozitiv din casă (LED-uri, TV, AC) prin apăsarea unui singur buton pentru a pleca liniștit și fără a avea stresul că a uitat unul din dispozitivele amintite mai sus conectat la priză, acesta consumând curent sau putând produce o catastrofă.

8 CONTRIBUȚII PERSONALE

- Colectarea, analiza și înțelegerea cerințelor
- Proiectarea schemei bloc a sistemului
- Proiectarea design-ului aplicației web din punct de vedere vizual
- Documentarea tehnologiilor necesare pentru realizarea proiectului
- Instalarea tuturor bibliotecilor necesare pentru mediul de lucru
- Crearea mediului de lucru în programul de dezvoltare
- Dezvoltarea codului pentru fiecare senzor în parte:
    - Clasa de bază pentru fiecare agent ce va fi creat (Anexa 1)
    - Agentul pentru monitorizarea temperaturii și a umidității (Anexa 2)
    - Agentul pentru monitorizarea mișcării în încăpere (Anexa 3)
    - Agentul pentru monitorizarea nivelului de gaz (Anexa 4)
    - Agentul pentru monitorizarea nivelului de luminozitate (Anexa 5)
    - Agentul pentru controlul draperiilor (Anexa 6)
    - Agentul pentru transmiterea și recepția datelor infraroșu (Anexa 7)
- Crearea interfeței utilizatorului (Anexa 8)
- Colectarea datelor de către server de la fiecare componentă hardware în parte și introducerea acestora în baza de date (Anexa 9)
- Implementarea și testarea comunicației între toți agenții și server (Anexa 10)

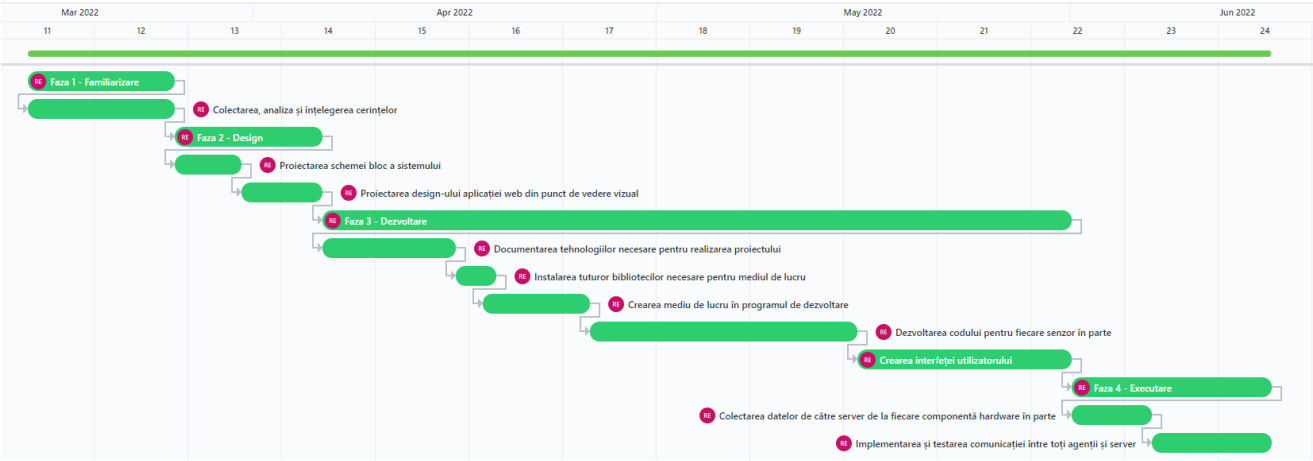

Figura 30 Diagrama Gantt a proiectului

9 BIBLIOGRAFIE

[1] What is the Internet of Things (IoT)? Https://Mesh-Net.Co.Uk, https://www.mesh-net.co.uk/what-is-the-internet-of-things-iot/, accesat la 15.05.2022

[2] Iotpadstaff, I. (2022, June 6). *Best Home Automation Protocols for the internet of things*. IOT & Smart Technology Review & Buyer guide, https://theiotpad.com/tips/home-automation-protocols, accesat la 15.05.2022

[3] Null, C. (2020). Best smart bulbs for your connected home, https://www.techhive.com/article/3129887/best-smart-bulbs.html , accesat la 30.05.2022

[4] *How to install Arduino Software (IDE) on Kali Linux - javatpoint*. www.javatpoint.com. (n.d.), from https://www.javatpoint.com/how-to-install-arduino-software-on-kali-linux , accesat la 12.06.2022

[5] Fezari, Mohamed & Al Dahoud, Ali. (2018). Integrated Development Environment "IDE" For Arduino.

[6] Kleene, R. (2020, September 21). The Era of Visual Studio Code, https://blog.robenkleene.com/2020/09/21/the-era-of-visual-studio-code/, accesat la 16.05.2022

[7] Michael Plainer. (2020). Study of Visual Studio Code

[8] Why Visual Studio Code? (2021, November 3). Https://Code.Visualstudio.Com/, https://code.visualstudio.com/docs/editor/whyvscode, accesat la 25.05.2022

[9] Shah, Hezbullah & Soomro, Tariq. (2017). Node.js Challenges in Implementation. Global Journal of Computer Science and Technology. 17. 72-83.

[10] Bangare, Sunil & Gupta, S & Dalal, M & Inamdar, A. (2016). Using Node.Js to Build High Speed and Scalable Backend Database Server. nternational Journal of Research in Advent Technology (E-ISSN: 2321-9637). 4. 19.

[11] Express/node introduction - learn web development: MDN. Learn web development | MDN. (n.d.), https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction, accesat la 23.05.2022

[12] Nanda, Umakanta & Pattnaik, Sushant. (2016). Universal Asynchronous Receiver and Transmitter (UART). 1-5. 10.1109/ICACCS.2016.7586376.

[13] Babiuch, Marek & Foltynek, Petr & Smutný, Pavel. (2019). Using the ESP32 Microcontroller for Data Processing. 1-6. 10.1109/CarpathianCC.2019.8765944.

[14] DHT11-Technical-Data-Sheet. Https://Www.Mouser.Com,https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf, accesat la 7.06.2022

[15] PIR Motion Sensor. Https://Cdn-learn.Adafruit.Com, https://cdn-learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf, accesat la 7.06.2022

[16] Technical Data MQ-2 Gas Sensor. Http:/Www.Hwsensor.Com, https://www.mouser.com/datasheet/2/321/605-00008-MQ-2-Datasheet-370464.pdf, accesat la 7.06.2022

[17] Trisnawan, I & Jati, Agung & Istiqomah, Novera & Wasisto, Isro. (2019). Detection of Gas Leaks Using The MQ-2 Gas Sensor on the Autonomous Mobile Sensor. 177-180. 10.1109/IC3INA48034.2019.8949597.

[18] ArduinoModules, Karim, Giugliano, S., Alpha19, Samuelsson, D., Nick, Vladimir, & Pierre.(2021, December 3). *KY-018 Photoresistor Module*. ArduinoModulesInfo, https://arduinomodules.info/ky-018-photoresistor-module/, accesat la 23.06.2022

[19] Roy, Tanu & Kabir, Humayun & Chowdhury, Md. (2014). Simple Discussion on Stepper Motors for the Development of Electronic Device. 5. 1089-1096.

[20] thalo/, P. B. : (2022, April 25). *Ir illuminator - 940nm VS 850nm wavelengths for IR illuminators*. AXTON Illuminators, https://axtontech.com/infrared-850nm-vs-940nm wavelength/, accesat la 23.06.2022

10 ANEXE

- Anexa 1

```
#include "config.hpp"
#include "Motion_Sensor.hpp"
//#include "DHT11_Sensor.hpp"
//#include "Gas_Sensor.hpp"
//#include "LDR_Sensor.hpp"
//#include "IR_Receive.hpp"
//#include "IR_Send.hpp"
#include "esp_now_config.hpp"
#ifdef USE_MOTION_SENSOR
  MotionSensor currentsensor;
#endif

#ifdef USE_GAS_SENSOR
  GasSensor currentsensor;
#endif

#ifdef USE_DHT11_SENSOR
  DHT11Sensor currentsensor;
#endif

#ifdef USE_LDR_SENSOR
  LDRSensor currentsensor;
#endif

#ifdef USE_IRRECV_SENSOR
  IR_Receive currentsensor;
#endif

#ifdef USE_IRSEND_SENSOR
  IR_Send currentsensor;
#endif
```

```
void setup()
{
   Serial.begin(115200);
   esp_now_setup();
   currentsensor.init();

//   esp_sleep_enable_timer_wakeup(10
* uS_TO_S_FACTOR);
}
void loop () {
  currentsensor.measure();

//  currentsensor.sendData();

Serial.println(currentsensor.getJSONst
ring());
  esp_now_send(broadcastAddress,
(uint8_t *)
currentsensor.getJSONstring().c_str(),
currentsensor.getJSONstring().length()
+ 2);
//  Serial.println("Going to
sleep...");
  delay(3000);
//  esp_deep_sleep_start();
}
```

- Anexa 2

```
#pragma once

#include "Sensor.hpp"

#include <Adafruit_Sensor.h>

#include <DHT.h>

class DHT11Sensor : public Sensor {

    float Temperature;

    float Humidity;

    DHT dht = DHT(34, 11);

  public:

    DHT11Sensor() :
Sensor(SensorType::DHT11_SENSOR) {

    }

    void init() {

        dht.begin();

    }

void measure() {

      Temperature =
dht.readTemperature();

      Humidity = dht.readHumidity();

    }

    String getJSONstring() {

      DynamicJsonDocument
jBuffer(1024);

      String jsondata = "";

      jBuffer["sensor_name"] =
"dht11_sensor";

      jBuffer["value1"] = Temperature;

      jBuffer["value2"] = Humidity;

      serializeJson(jBuffer,
jsondata);

      return jsondata;

    };

};
```

- Anexa 3

```
#pragma once

#include "Sensor.hpp"


class MotionSensor : public Sensor {

    int value = 0;

    int state = LOW;

    const int mMotionSensorPin = 34;

    const int led = 22;

  public:

    MotionSensor() :
Sensor(SensorType::MOTION_SENSOR) {


    }


    void init() {

        pinMode(mMotionSensorPin,
INPUT);


//      Set LED to LOW

        pinMode(led, OUTPUT);

        digitalWrite(led, LOW);

    }


    void measure() {


      value =
digitalRead(mMotionSensorPin);   //
read sensor value

      if (value == HIGH) {
// check if the sensor is HIGH

        digitalWrite(led, LOW);   //
turn LED ON

        delay(100);                //
delay 100 milliseconds

if (state == LOW) {

          Serial.println("Motion
detected!");

          state = HIGH;       //
update variable state to HIGH

        }

      }

      else {

          digitalWrite(led, HIGH); //
turn LED OFF

          delay(200);             //
delay 200 milliseconds


          if (state == HIGH){

            Serial.println("Motion
stopped!");

            state = LOW;       //
update variable state to LOW

        }

      }

    }



    String getJSONstring() {

      DynamicJsonDocument
jBuffer(1024);

      String jsondata = "";

      jBuffer["sensor_type"] =
"MotionSensor";

      jBuffer["motion_detection"] =
value;

      serializeJson(jBuffer,
jsondata);

      return jsondata;

    };

};
```

- Anexa 4

```
#pragma once

#include "Sensor.hpp"

#include <MQUnifiedsensor.h>

#define         Board
("ESP-32")

#define         Pin
(34)

#define         Type
("MQ-2")

#define         Voltage_Resolution
(3.3)

#define         ADC_Bit_Resolution
(12) // For ESP32

#define         RatioMQ2CleanAir
(9.83) //RS / R0 = 9.83 ppm

class GasSensor : public Sensor {

    int sensorValue;

    MQUnifiedsensor mq2 =
MQUnifiedsensor(Board,
Voltage_Resolution,
ADC_Bit_Resolution, Pin, Type);

  public:

    GasSensor() :
Sensor(SensorType::GAS_SENSOR) {

    }

    void init() {

        mq2.setRegressionMethod(1);
//_PPM =  a*ratio^b

        mq2.setA(5740.25); mq2.setB(-
2.222);

        /*

        Exponential regression:

        Gas    | a      | b

        H2     | 987.99 | -2.162

        LPG    | 5740.25 | -2.222

        CO     | 36974  | -3.109

        Alcohol| 3616.1 | -2.675S

        Propane| 658.71 | -2.168

        */

        mq2.init();

        mq2.setRL(2);

        float calcR0 = 0;

        for(int i = 1; i<=10; i ++)

        {

          mq2.update(); // Update
data, the arduino will read the
voltage from the analog pin

          calcR0 +=
mq2.calibrate(RatioMQ2CleanAir);

        }

        mq2.setR0(calcR0/10);

        mq2.serialDebug(true);

    }

    void measure() {

      mq2.update();

      sensorValue = mq2.readSensor();


Serial.println(mq2.readSensor());

      mq2.serialDebug();

    }

    String getJSONstring() {

      DynamicJsonDocument
jBuffer(1024);

      String jsondata = "";

      jBuffer["sensor_name"] =
"gas_sensor";

      jBuffer["value1"] = sensorValue;

      serializeJson(jBuffer,
jsondata);

      return jsondata;

    };

};
```

- Anexa 5

```
#pragma once

#include "Sensor.hpp"



class LDRSensor : public Sensor {

    int RVal;

    const int mLDRSensorPin = 34;

    const int led = 22;


  public:

    LDRSensor() :
Sensor(SensorType::LDR_SENSOR) {



    }



    void init() {

        pinMode(led, OUTPUT);

        digitalWrite(led, HIGH);

    }
```

```
void measure() {

      RVal = 4095 -
analogRead(mLDRSensorPin);

      if(RVal < 1000)

        digitalWrite(led, LOW);

      else

        digitalWrite(led, HIGH);

    }



    String getJSONstring() {

      DynamicJsonDocument
jBuffer(1024);

      String jsondata = "";

      jBuffer["sensor_name"] =
"LDR_sensor";

      jBuffer["value1"] = RVal;

      jBuffer["value2"] = 1-
digitalRead(led);

      serializeJson(jBuffer,
jsondata);

      return jsondata;

    };

};
```

- Anexa 6

```
#include <Stepper.h>

#include <esp_now.h>

#include <WiFi.h>

#include <ArduinoJson.h>


// Number of steps per internal motor
revolution

const float STEPS_PER_REV = 32;


//  Amount of Gear Reduction

const float GEAR_RED = 64;


// Number of steps per geared output
rotation

const float STEPS_PER_OUT_REV =
STEPS_PER_REV * GEAR_RED;


// Number of Steps Required

int StepsRequired;


// Pins entered in sequence 1-3-2-4
for proper step sequencing

Stepper steppermotor(STEPS_PER_REV,
36, 34, 39, 35);


String jsondata1;


void OnDataRecv(const uint8_t * mac,
const uint8_t *incomingData, int len)
{

  char* buff = "";

  jsondata1 = "";

  buff = (char*) incomingData;

  jsondata1 = String(buff);

  Serial.println(jsondata1);

  if(jsondata1 ==
"\"stepperState\":1")

  {

    StepsRequired  =
STEPS_PER_OUT_REV * 2;

    steppermotor.setSpeed(1000);

    steppermotor.step(StepsRequired);

  }

  else if(jsondata1 ==
"\"stepperState\":0")

  {

    StepsRequired  = -
STEPS_PER_OUT_REV * 2;

    steppermotor.setSpeed(1000);

    steppermotor.step(StepsRequired);

  }

}

void setup() {

  // Initialize Serial Monitor

  Serial.begin(115200);

  // Set device as a Wi-Fi Station

  WiFi.mode(WIFI_STA);


  // Init ESP-NOW

  if (esp_now_init() != ESP_OK) {

    Serial.println("Error initializing
ESP-NOW");

    return;

  }


esp_now_register_recv_cb(OnDataRecv);

}


void loop() {


}
```

- Anexa 7

```
#pragma once

#include <esp_now.h>

#include <WiFi.h>

#include <ArduinoJson.h>

const uint8_t broadcastAddress[] =
{0x58, 0xBF, 0x25, 0x9D, 0x7E, 0x44};

esp_now_peer_info_t peerInfo;

#ifdef USE_IRSEND_SENSOR

  IR_Send irSend;

#endif

//RECEIVER----------------------------
-----//

String receiveData;

void OnDataRecv(const uint8_t * mac,
const uint8_t *incomingData, int len)
{

  char* buff = "";

  receiveData = "";

  buff = (char*) incomingData;

  receiveData = String(buff);

  Serial.println(receiveData);

  #ifdef USE_IRSEND_SENSOR


    StaticJsonDocument<512> doc;

    if(buff[2] == 'p')

    {

      deserializeJson(doc, buff);

      decode_type_t protocol =
doc["protocol"];

      uint16_t addressAsString =
doc["address"];

      uint8_t command =
doc["command_code"];

      Serial.println(protocol);

      decode_type_t protocol_test =
NEC;

      irSend.sendData(protocol_test,
addressAsString, command);

    }

    else if(buff[2] == 'r')

    {

      Serial.println("receive cmd");

      while(!irSend.irrecv.decode()){}

        if(irSend.irrecv.decode()) {

          irSend.irrecv.resume();

        }

      DynamicJsonDocument
jBuffer(1024);

      String jsondata = "";

      jBuffer["sensor_name"] = "ir";

      jBuffer["protocol"] =
irSend.irrecv.decodedIRData.protocol;

      jBuffer["address"] =
irSend.irrecv.decodedIRData.address;

      jBuffer["command_code"] =
irSend.irrecv.decodedIRData.command;

      serializeJson(jBuffer,
jsondata);

      esp_now_send(broadcastAddress,
(uint8_t *) jsondata.c_str(),
jsondata.length() + 2);

    }

  #endif

}

//----------------------------------
-----//
```

- Anexa 8

```
<header class="main-header">···
</header>
<body>
  <button onclick="topFunction()" id="scrollTopBtn" title="Go to top">
    Top
  </button>
  <section id="key-features">···
  </section>
  <hr>
  <section id="key-features">···
  </section>
  <hr>
  <section id="key-features">···
  </section>
  <h2>LED Control</h2>
  <div id="ledState">
    <script type="text/javascript">···
    </script>
    <ul class="LEDBtns">···
    </ul>
  </div>
  <br /><br />
  <h2>Remote control</h2>
  <ul class="remoteBtns">···
  </ul>
  <br /><br />
  <h2>Stepper Control</h2>
  <div id="stepperState">···
  </div>
  <br /><br />
  <script>
    function getIRData(command_name) {···
  </script>
  <script src="scrollTopBtn.js"></script>
</body>
```

- Anexa 9

```
function addSensorsToDb(dataFromSensor) {
  var today = moment().toDate()
  today = moment(today).format('YYYY-MM-DD HH:mm:ss')
  let post = {
    sensor_name: dataFromSensor.sensor_name,
    value1: dataFromSensor.value1,
    value2: dataFromSensor.value2,
    date: today
  };
  let sql = 'INSERT INTO sensors SET ?'
  let query = db.query(sql, post, err => {
    if(err) {
      throw err
    }
  })
}
```

```
// Get dht11 data
app.get("/dht11-sensor", (req, res) => {
  let sql = `SELECT * from sensors WHERE sensor_name = 'dht11_sensor'`
  let query = db.query(sql, (err, results) => {
    if(err) {
      throw err
    }
    for(let element of results) {
      element.date = moment(element.date).format('YYYY-MM-DD HH:mm:ss')
    }
    res.send(results);
  })
});

// Get gas sensor data
app.get("/gas-sensor", (req, res) => {···
});

// Get LDR data
app.get("/LDR-sensor", (req, res) => {···
});

// Get PIR data
app.get("/PIR-sensor", (req, res) => {···
});
```

- Anexa 10

```
Received data from port: {"sensor_name":"PIR_sensor","value1":0}
Received data from port: {"sensor_name":"dht11_sensor","value1":26,"value2":18}
Received data from port: {"sensor_name":"LDR_sensor","value1":75,"value2":1}
Received data from port: {"sensor_name":"gas_sensor","value1":20}
Received data from port: {"sensor_name":"PIR_sensor","value1":0}
Received data from port: {"sensor_name":"dht11_sensor","value1":27,"value2":17}
Received data from port: {"sensor_name":"LDR_sensor","value1":109,"value2":1}
Received data from port: {"sensor_name":"gas_sensor","value1":20}
Received data from port: {"sensor_name":"PIR_sensor","value1":0}
Received data from port: {"sensor_name":"dht11_sensor","value1":26,"value2":18}
Received data from port: {"sensor_name":"LDR_sensor","value1":47,"value2":1}
Received data from port: {"sensor_name":"gas_sensor","value1":20}
Received data from port: {"sensor_name":"dht11_sensor","value1":27,"value2":17}
Received data from port: {"sensor_name":"PIR_sensor","value1":0}
Received data from port: {"sensor_name":"LDR_sensor","value1":161,"value2":1}
Received data from port: {"sensor_name":"gas_sensor","value1":20}
Received data from port: {"sensor_name":"dht11_sensor","value1":27,"value2":17}
Received data from port: {"sensor_name":"PIR_sensor","value1":0}
Received data from port: {"sensor_name":"LDR_sensor","value1":117,"value2":1}
Received data from port: {"sensor_name":"gas_sensor","value1":20}
Received data from port: {"sensor_name":"dht11_sensor","value1":27,"value2":17}
```