MINISTERUL EDUCAȚIEI ŞI CERCETĂRII AL REPUBLICII MOLDOVA
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică şi Microelectronică
Departamentul Ingineria Software şi Automatică
Programul de studii: Tehnologia informaţiei

Sistem portabil pentru monitorizarea factorilor care influenţează sănătatea

Practica de licenţă

Student: _____ **Pleşu Cătălin, TI-206**

Coordonator întreprindere: _____ **Munteanu Eugeniu, programator analist**

Coordonator de licenţă: _____ **Secrieru Adrian, asist.univ.**

Coordonator universitate: _____ **Cojocaru Svetlana, asist.univ.**

Chişinău, 2024

CUPRINS

ABREVIERI

- IoT (eng. Internet of Things) - internetul lucrurilor, rețeaua de dispozitive fizice interconectate care comunică și cooperează;
- IT (eng. Information Technology) - Tehnologia Informației, utilizarea calculatoarelor și a tehnologiei pentru stocarea, procesarea și transmiterea informațiilor;
- PM1 (eng. Particulate matter 1) - particule în suspensie cu diametrul de 1 micrometru sau mai mic, care pot afecta sănătatea umană,
- PM2.5 (eng. Particulate Matter 2.5) - particule în suspensie cu diametrul de 2.5 micrometri sau mai mic, care pot avea efecte nocive asupra sănătății umane;
- PM10 (eng. Particulate Matter 10) - particule în suspensie cu diametrul de 10 micrometri sau mai mic, care pot afecta calitatea aerului și sănătatea umană;
- VOCs (eng. Volatile Organic Compounds) - Compuși Organici Volatili, substanțe chimice care pot emana vapori și care pot avea efecte nocive asupra sănătății umane;
- ANT+ (eng. Advanced and Adaptive Network Technology) - tehnologie de comunicație wireless utilizată în special în dispozitivele de fitness și sănătate pentru a transmite date către alte dispozitive compatibile;
- Wi-Fi (eng. Wireless Fidelity) - tehnologie de rețea fără fir care permite dispozitivelor să se conecteze la internet și să comunice între ele într-o rețea locală;
- FPGA (eng. Field-Programmable Gate Array) - matrice de porți programabilă, dispozitive hardware care pot fi reconfigurate după producție pentru a îndeplini diverse funcții;
- ASIC (eng. Application-Specific Integrated Circuit) - circuit integrat specific aplicației, o formă de cipuri electronice pecializate proiectate pentru o anumită aplicație;
- VHDL (eng. VHSIC (eng. Very High-Speed Integrated Circuit Hardware Description Language) Hardware Description Language) - , limbaj de programare utilizat pentru proiectarea circuitelor digitale;
- C - limbaj de programare de nivel înalt utilizat în dezvoltarea software-ului de sistem și de aplicații;
- C++ - limbaj de programare de nivel înalt și orientat pe obiecte, derivat din limbajul C, utilizat în special pentru dezvoltarea de software complex;
- OS (eng. Operating System) - sistem de operare, software-ul de bază care controlează hardware-ul unui computer și permite utilizatorilor să ruleze aplicații și să interacționeze cu sistemul;
- RTOS (eng. Real-Time Operating System) - sistem de operare în timp real, un tip de sistem de operare care oferă garanții de timp în executarea sarcinilor;

- OTA (eng. Over-the-Air) - actualizare prin aer, procesul de actualizare a software-ului dispozitivelor sau a echipamentelor fără a fi necesară o conexiune fizică;
- ZigBee - standard de comunicare fără fir utilizat în rețelele locale, în special în aplicații de automatizare a clădirilor și a casei inteligente;
- LoRaWAN (eng. Long Range Wide Area Network) - rețea de arie largă cu distanțe mari, tehnologie de comunicație wireless de lungă distanță și de consum redus de energie, folosită pentru conectarea dispozitivelor IoT la internet;
- AMQP (eng. Advanced Message Queuing Protocol) - protocol avansat de mesagerie de tip mesaj-coloană, pentru transmiterea mesajelor între aplicații;
- MQTT (eng. Message Queuing Telemetry Transport) - protocolul de transport al mesajelor mașină la mașină, utilizat în special pentru dispozitive IoT pentru a trimite și a primi date într-o manieră eficientă;
- HTTP (eng. Hypertext Transfer Protocol) - protocolul de transfer hipertext, utilizat pentru transmiterea informațiilor pe internet;
- CoAP (eng. Constrained Application Protocol) - protocolul constrâns de aplicație, pentru aplicațiile web de tip restful pe dispozitivele iot cu resurse limitate;
- DDS (eng. Data Distribution Service) - serviciul de distribuție a datelor, o tehnologie de middleware utilizată pentru transmiterea și schimbul de date între dispozitivele conectate la rețea;
- LwM2M (eng. Lightweight M2M) - protocol și servicii de administrare a dispozitivelor IoT utilizate pentru gestionarea și monitorizarea dispozitivelor IoT în rețelele LPWAN;
- IO (eng. Input/Output) - intrare/ieșire, interfața prin care dispozitivele comunică cu mediul lor exterior;
- HG - Hotărâre Guvernamentală;
- CSV (eng. Comma-Separated Values) - valori separate prin virgulă, un format de fișier utilizat pentru stocarea și schimbul de date tabelare;
- SQLite - O bibliotecă de programare care oferă o bază de date relațională ușoară și fără server;
- URL (eng. Uniform Resource Locator) - localizator uniform de resurse, adresa web care specifică locația unei resurse de pe internet;
- API (eng. Application Programming Interface) - interfață de programare a aplicației, un set de reguli și protocoale care permit diferitelor componente de software să comunice între ele;
- CI (eng. Continuous Integration) - integrare continuă, practica de a automatiza testarea și integrarea codului nou într-un proiect software în mod frecvent;
- CD (eng. Continuous Delivery) - livrare continuă, o practică în dezvoltarea software-ului care implică automatizarea procesului de livrare a software-ului la utilizatori;

- UML (eng. Unified Modeling Language) - limbajul de modelare unificată, un set de notări şi reguli pentru a descrie şi a modela structura şi comportamentul sistemelor software;
- SysML (eng. Systems Modeling Language) - limbajul de modelare a sistemelor, o extensie a uml utilizată pentru modelarea şi analiza sistemelor fizice şi tehnice;
- AADL (eng. Architecture Analysis and Design Language) - limbajul de analiză şi proiectare a arhitecturii, un limbaj de modelare utilizat în ingineria sistemelor embebuite şi în timp real;
- OPM (eng. Object Process Methodology) - metodologia de proces orientată pe obiecte, o metodologie de dezvoltare a sistemelor care se concentrează pe modelarea obiectelor şi a interacţiunilor dintre ele.

INTRODUCERE

În această lucrare vor fi prezentate rezultatele analizei domeniului de monitorizare a mediului. Rezultatele comparării a trei sisteme existente, Atmotube PRO, Garmin Tempe şi DT-8820. Au fost definite scopul, obiectivele şi specificaţiile tehnice ale proiectului. A fost proiectat sistemul utilizând limbajul de modelare unificat UML.

Practica a avut loc în perioada 29 ianuarie – 9 februarie şi 19 februarie – 1 martie. Conform contractului de angajare, practica a avut loc la compania Arobs Software S.R.L care pe piaţa din Republica Moldova este specializată în domeniul auto. În special programarea sistemelor încorporate din care sunt constituite autoturismele.

Ca activităţi practice pe lângă documentarea pentru realizarea acestui sistem a fost demarată şi operaţiunea de realizare a sistemului. La moment sistemul se află încă în etapa de dezvoltare, dar până la vara s-ar putea chiar să fie finalizat.

1 ANALIZA DOMENIULUI

Monitorizarea mediului cuprinde metode şi strategii pentru găsirea, analizarea şi stabilirea parametrilor mediului. Aceasta poate duce la reglarea nivelurilor de poluare în cazurile când calitatea mediului nu este satisfăcătoare [1]. Factorii de bază pe care se concentrează această disciplină sunt:

- aerul;
- apa;
- solul;
- gălăgia;
- biodiversitatea.

Dintre factorii enumeraţi mai sus, aerul şi gălăgia sunt centrul atenţiei pentru această teză.

Dintre tendinţele tehnologice în domeniul de monitorizare al mediului cele mai interesante sunt teledetecţia şi internetul lucrurilor. Teledetecţia presupune utilizarea de sateliţi, drone, avioane sau alte dispozitive pentru a capta imagini sau a efectua măsurători ale suprafeţei sau atmosferei pământului. Internetul lucrurilor (IoT) este reţeaua de obiecte fizice interconectate, cum ar fi senzori, dispozitive sau maşini, care pot comunica şi schimba date [2]. Dintre aceste tehnologii, IoT este mai iefină şi scalabilă, ceea ce ar permite de a monitoriza mediul în timp real.

Internetul lucrurilor prin automatizare, poate genera o cantitate de mai mare de date valoroase pentru a ajuta la luarea deciziilor. Totodată facilitează interacţiunea dintre oameni şi maşini, dar şi între maşini. Dintre problemele cu care se întâmpină acest domeniu sunt asigurarea asigurarea interoperabilităţii, scalabilităţii, fiabilităţii, securităţii, confidenţialităţii şi eticii dispozitivelor şi sistemelor IoT. Interoperabilitatea se referă la capacitatea diferitelor dispozitive şi sisteme de a comunica şi de a lucra împreună fără probleme. Scalabilitatea se referă la capacitatea de a gestiona un număr tot mai mare de dispozitive şi date fără a compromite performanţa sau calitatea. Fiabilitatea se referă la capacitatea de a funcţiona corect fără erori sau defecţiuni. Securitatea se referă la capacitatea de a proteja dispozitivele şi datele împotriva accesului neautorizate sau atacurilor. Confidenţialitatea se referă la capacitatea de a respecta drepturile şi preferinţele utilizatorilor cu privire la datele lor personale. Etica se referă la capacitatea de a adera la principiile şi valorile morale atunci când proiectaţi şi utilizaţi dispozitive şi sisteme IoT [3].

Tehnologia informaţiei (IT) este un set de domenii conexe care cuprind sisteme informatice; software; limbaje de programare; procesarea datelor şi stocarea informaţiilor [4]. Prin urmare IoT este un subdomeniul al tehnologiei informaţiei având ca conceptele fundamentale:

- date;
- dispozitive;
- analiza;

- conectivitate.

1.1 Importanţa temei

Monitorizarea parametrilor mediului este un domeniu important. Fiind că calitatea acestor parametri influenţează direct sănătatea, nu doar a oamenilor, dar ar oricărui organizam viu. Spre exemplu, anual 9 milioane de oameni mor din cauza poluării [5]. Majoritatea ţărilor nu depun eforturi considerabile pentru a rezolva această problemă [5]. Prin urmare este important ca măcar să conştientizăm nivelul de poluare la care suntem expuşi.

Poluare nu poate fi evitată prin izolare de la lumea externă. Fiind că populaţia umană deja face acest lucru, populaţia globală petrece în jur de 90% în interior. Analizând aceste informaţii, putem deduce că nivelul poluării în interior nu este mai mic ca nivelul de poluare în exterior, ba chiar poate fi mai mare aşa cum morţile cauzate de poluare sunt în creştere [5] iar oamenii îşi petrec din ce în ce mai mult timp în interior.

O soluţie evidentă, pentru a reduce efectele negative ale poluării asupra sănătăţii, este de a petrece mai mult timp în natură. Filosoful Friedrich Nietzsche a spus: "O viaţă sedentară este adevăratul păcat împotriva Duhului Sfânt.". Acesta a ameliorat simptomele unei boli pe care le avea, făcând nişte drumeţii în natură.

Însă există posibilitatea ca mediul în care are loc odihna să fie poluat, sau să nu fie benefic sănătăţii. Prin urmare monitorizarea mediului, este un instrument indispensabil pentru protejarea sănătăţii. Monitorizarea în timp real, cu feedback este şi mai bună, aşa cum nu este necesar de a investiga constant dacă starea mediului este satisfăcătoare. În caz că un parametru nu este corespunzător normelor, sistemul va înştiinţa utilizatorul.

1.2 Sisteme similare cu proiectul realizat

Viziunea proiectului care va fi realizat este un sistem portabil, care permite utilizatorilor să se informeze despre parametrii mediului în care se află. Astfel au fost identificate 3 sisteme deţin parţial capabilităţile sistemului dorit. Aceste sisteme sunt:

- Atmotube PRO [6];
- Tempe [7];
- DT-8820 [8].

Pentru a analiza aceste sisteme, vor fi analizate următoarele calităţi ale acestora:

- parametrii măsuraţi;
- funcţionalităţi;
- dezavantaje;
- durata de viaţă a bateriei;
- preţul.

Atmotube PRO, reprezentat în figura 1.1 este o staţie meteo portabilă capabilă şi să monitorizeze calitatea aerului. Aceasta poate monitoriza următorii parametri:

- particule PM1, PM2.5, PM10;
- compuşi organici volatili (VOCs);
- temperatura aerului;
- umiditatea relativă;
- presiunea barometrică.

Funcţionalităţi-le principale ale acestuia sunt monitorizarea parametrilor menţionaţi anterior, această monitorizare are loc în timp real, iar datele sunt transferate pe telefonul mobil pentru a fi afişate prin intermediul Bluetooth.

Printre dezavantajele acestui sistem se numără faptul că deşi poate fi considerat portabil, are dimensiuni destul de mari 86 x 50 x 22 mm. Ceea ce l-ar face incomod de purtat în mai multe situaţii.

Durata de viaţă a bateriei este de 10 zile. În comparaţie cu, celelalte sisteme identificate aceasta tot poate fi considerată un dezavantaj.

Preţul acestuia este în jur de 3400 de lei. Ceea ce îl face inaccesibil pentru un număr mare de personae. Totuşi preţul poate fi atribuit unui număr mai mare de parametri monitorizaţi, şi faptului că dispozitivul poate să se conecteze la un telefon mobil.

Figura 1.1 - Atmotube PRO

Garmin Tempe, reprezentat în figura 1.2 este un mic senzor de temperatură, care se evidenţiază prin dimensiunea sa mică de 18 x 12 x 18 mm. Acesta poate monitoriza doar temperatura.

Funcţionalitatea sa principală este monitorizare temperaturii. Afişarea are loc pe alt dispozitiv, care trebuie să fie compatibil cu protocolul ANT+.

El are două dezavantaje majore. Cum a fost menționat mai sus, datele pot fi afișate doar pe un dispozitive Garmin compatibil prin tehnologia ANT+. Al doilea dezavantaj este că poate monitoriza doar temperatura aerului.

Bateria acestuia are o durată de viață extraordinară de aproximativ un an. Asta se datorează faptului că dispozitivul are limitările menționate mai sus.

Acesta este cel mai accesibil sistem în această listă. Prețul acestuia este în jur de 800 de lei. Totuși considerând numărul de parametri măsurați, nu poate fi considerat un preț rezonabil.

Figura 1.2 - Garmin Tempe

DT-8820, reprezentat în figura 1.3 este un multimetru capabil să măsoare intensitatea luminii, intensitatea sunetului, temperatura aerului şi umiditatea relativă.

Spre deosebire de celelalte dispozitive, DT-8820 dispune de un ecran pentru a afișa datele măsurate. Şi chiar are şi butoane fizice pentru a selecta modul al acestuia de lucru.

Dintre dezavantaje se numără lipsa conectivității, acesta poate fi utilizat ca un multimetru obișnuit, fără posibilitatea de a colecta datele. Dimensiunea acestui dispozitiv 251 x 64 x 40 mm, tot este un dezavantaj, deşi poate fi considerat portabil, el nu poate fi considerat purtabil.

Acest dispozitiv foloseşte baterii de 9V, şi fiind ca nu este utilizat în continuu, asta ar presupune că durata de viață a unei baterii ar putea fi chiar mai mare de un an.

Prețul acestui dispozitiv este în jur de 1200 de lei. Pentru numărul de parametri care ii poate măsura pare a fi un preț mai rezonabil decât celelalte dispozitive.

Figura 1.3 - DT-8820

În urma comparării acestor dispozitive s-a stabilit că caracteristicile unui sistem dorit ar fi ca acesta să poată măsura câți mai mulți parametri posibil, dar cel puțin temperatura, umiditatea relativă, compușii organici volatili, nivelul de gălăgie și nivelul de iluminare. Ideal prețul acestui sistem ar trebui să fie cât mai accesibil. Cel puțin să nu depășească prețul de 600 de lei.

De la un astfel de sistem se urmărește în special ca dimensiunile acestuia să fie mici, cât mai aproape posibil de dimensiunile a Garmin Tempe de 18 x 12 x 18 mm. Și de asemenea cu o durată de viața a bateriei mare, de cel puțin 30 de zile. Dispozitivul trebuie să posede conectivitate prin bluetooh pentru conectarea la un telefon mobil și char și suport pentru Wi-Fi pentru a transmite datele către un server MQTT.

1.3 Analiza tehnologiilor

Un sistem IoT tipic are mai multe niveluri. Aceste niveluri depind unul de altul, importanța lor pentru sistem fiind în ordinea enumerării [9]:

- hardware dispozitiv;
- software-ul dispozitivului;
- protocoale de comunicare;
- platformă cloud.

La nivelul de hardware pe lângă mulțimea de senzori și mecanisme care vor fi utilizate, la baza dispozitivului se află o unitate de procesare. Opțiunile disponibile sunt: FPGA, Microcontroler sau ASIC [10]. Ce ține de dezvoltarea dispozitivului în caz că se optează pentru FPGA sau ASIC, va fi nevoie de scris cod în Verilog sau VHDL ceea ce este diferit de programarea clasică fiind că acest cod nu se execută secvențial dar se transformă în circuite care vor funcționa în același timp. Dacă se optează pentru microcontroler, atunci acesta poate fi programat cu limbaje cum ar fi C sau C++ dar și altele. Compararea

acestor platforme hardware este prezentată în tabelul 1.1, fiecare tehnologie are avantajele și dezavantajele sale. Pentru un sistem care va fi produs pe scara mică, cel mai potrivit pare a fi un microcontroler, așa cum acesta este mai ieftin de procurat în cantități mici, oferă un pic de flexibilitate în utilizarea acestuia însă este mai ușor de utilizat ca celelalte tehnologii.

Tabelul 1.1: Comparare FPGA, microcontroler, ASIC [10]

Criteriu	**FPGA**	**Microcontroler**	**ASIC**
Cost (în cantităţi mici)	moderat	mic	mare
Cost (în cantităţi mari)	moderat	mic	mic
Viteza de procesare	rapid	moderat	foarte rapid
Consumul de energie	moderat	mic	mic
Flexibilitatea	mare	mică	zero
Uşurinţa de utilizare	moderat	uşor	dificil

Software-ul dispozitivului poate fi scris pentru a rula direct pe hardware, sau pentru a utiliza un sistem de operare, de cele mai multe ori, un sistem de operare în timp real [11]. Fiecare abordare are avantajele și dezavantajele sale. Avantajul scrierii codului fără OS este că performanţa acestuia va fi mai mare. Avantajul utilizării unui RTOS este că acesta va garanta procesarea evenimentelor în timpul dorit. O funcţionalitate importantă pe care o poate avea software-ul dispozitivului este programarea OTA, care permite actualizarea software-ului dispozitivului la distanţă [9]. Ce ține de limbajele de programare care pot fi utilizate pentru a scrie acest software, nu există nici o limitare, pot fi utilizate atât limbaje compilate cât și limbaje interpretate. Din limbajele de programare compilate C este cel mai popular, estimându-se că acesta este utilizat în 80% din cazuri. C++ este utilizat din cauza că compilatorul suportă un standard mai nou de cât compilatorul de C. Dintre limbajele mai tinere care au început să fie populare și în domeniul sistemelor încorporate, sunt Rust și Zig [12]. Dintre limbajele interpretate, cele mai utilizate sunt Python și Lua.

Protocoalele de comunicare sunt aceleași ca și în cadrul altor sisteme. Pot fi identificate două categorii principale, protocoale de rețea și protocoale de date [13]. Dintre protocoalele de rețea cele mai utilizate sunt: Wi-Fi; Bluetooh; ZigBee; LoRaWAN. Dintre protocoalele de date cele mai utilizate sunt: AMQP; MQTT; HTTP; CoAP; DDS; LwM2M. La acest nivel nu este dificil de creat protocoale de date proprii.

Platforma în nori este utilizată pentru a prelua, prelucra, analiza și stoca datele. Printre platformele cloud populare se numără: Amazon Web Services; Microsoft Azure. Pe acest tip de platformă dezvoltatorii pot să ruleze aplicaţiile lor proprii. Însă există și platforme cloud care oferă servicii specializate, cum ar fi platforma Adafruit IO care are servicii de broker MQTT.

1.4 Parametrii mediului

Pentru a transforma datele colectate despre parametrii mediului în informaţii utile va fi nevoie de a avea nişte referinţe. Ca referinţă au fost luate HG353 din 2010 [14] şi regulamentul sanitar privind normativele de emitere a zgomotului [15] chiar dacă acesta nu a fost aprobat. Aceste regulamente au ca scop asigurarea condiţiilor de trai inofensive pentru populaţie şi se axează pe parametrii în interior. În continuare vor fi afişate tabele din aceste regulamente care vor fi utilizate ca referinţă.

În tabelul 1.2 sunt prezentate limitele termice minime şi maxime admise pentru diferite rate metabolice. Pentru a obţine metabolismul în waţi este nevoie de a converti cantitatea de Cal care o persoană o utilizează zilnic în Joule înmulţind-o cu 4184, apoi pentru a obţine waţi este nevoie de împărţit energia la numărul de secunde, în cazul dat într-o zi [16]. De asemenea din acest regulament mai poate fi evidenţiat faptul că în perioada rece a anului, se recomandă ca temperatura în interior să fie cu, cel puţin 3 °C mai mare ca în perioada caldă.

Tabelul 1.2 – Limitele termice minime şi maxime admise la posturile de lucru [14]

Metabolismul, (M) W	**Temperatura aerului minimă, °C**	**Temperatura aerului maximă, °C**
$M \leq 117$	18	32
$117 < M \leq 234$	16	29
$234 < M \leq 360$	15	26
$360 < M \leq 468$	12	22
$M > 468$	12	18

Indiferent de anotimp se recomandă ca umiditatea relativă în încăperi să fie menţinută între 30 % şi 70 % [14]. Umiditatea ridicată amplifică sensibilitatea la temperaturi reci, de aceia se recomandă ca dacă umiditatea aerului este mare, temperatura aerului să fie ridicată.

În tabelul 1.3 sunt prezentate nivelurile de iluminare recomandate, pentru lucrărilor cu diferite caracteristici. Aceste recomandări sunt potrivite atunci când contrastul între detaliul observat şi fond este mediu şi când fondul nu este nici luminos nici întunecat.

Tabelul 1.3 – Nivelul de iluminare recomandat, pentru contrastul mediu între detaliu şi fond [14]

Caracteristicile lucrărilor vizuale, pentru distanţa de privire de 344 mm	**Nivelul normat de iluminare, lx**
Lucrări de precizie deosebită cu detalii sub 0,1 mm	1500
Lucrări de precizie foarte mare cu detalii detalii între 0,1 mm şi 0,3 mm	750
Lucrări de precizie mare cu detalii între 0,3 mm şi 0,5 mm	500
Lucrări de precizie medie cu detalii între 0,5 mm şi 0,8 mm	300
Lucrări de precizie mică cu detalii între 0,8 mm şi 1,2 mm	200

În tabelul 1.4 sunt prezentate nivelurile sonore maxime admisibile în încăperile locative, conform legii, după ora 22 este nevoie ca zgomotele emise să nu depăşească 45 dBA. Aceste date sunt

caracteristice pentru apartamente, pentru cămine spre exemplu este permis ca nivelul zgomot să fie mai mare cu 5 dBA în ambele cazuri.

Tabelul 1.4 – Nivelurile sonore maxime ale zgomotului admise în încăperile locative [15]

Intervalul timpului de evaluare	**Nivelurile sonore maxime, dBA**
7.00-22.00	55
22.00-7.00	45

2 SCOPUL, OBIECTIVELE ŞI CERINŢELE SISTEMULUI

Problema existentă în domeniul de monitorizare al mediului este că majoritatea sistemelor existente pentru măsurarea unei varietăţi mari de parametri mai degrabă sunt nişte multimetre decât nişte dispozitive portabile, care să funcţioneze autonom şi doar să înştiinţeze utilizatorii în caz de pericol pentru sănătate. Deşi există sisteme care ar face parţial acest lucru, ele oferă un număr limitat de parametri pe care îi pot monitoriza, sau nu oferă suficiente funcţionalităţi.

Prin dezvoltarea unui sistem portabil pentru monitorizarea a factorilor mediului se urmăreşte scopul de a monitoriza aceşti parametri, pentru a identifica dacă aceştia prezintă un risc pentru sănătatea şi activitatea vitală. După cum a fost stabilit în capitolul precedent, toţi parametrii mediului pot avea un impact asupra sănătăţii şi activităţii vitale. Şi există norme legale ce definesc valorile dorite pentru majoritatea parametrilor, în special cei care pot fi controlaţi în interior. Este râvnită monitorizarea unui număr cât mai mare de parametri ai mediului. Însă din cauza constrângerilor de resurse materiale, dimensiuni ale viitorului sistem şi autonomiei acestuia se va opta ca acest sistem să poată monitoriza cel puţin temperatura, compuşii volatili organici, nivelul de iluminaţie şi nivelul de gălăgie.

Pentru a realiza acest sistem se propun următoarele obiective:

- dezvoltarea dispozitivului de monitorizare a mediului;
- monitorizarea parametrilor doriţi;
- asigurarea portabilităţii dispozitivului;
- dezvoltarea aplicaţiei mobile;
- stabilirea conexiunii între dispozitiv şi aplicaţia mobilă;
- implementarea alarmelor;
- crearea unei interfeţe de utilizator minimalistă;
- monitorizarea mediului în timp real;
- minimizarea costurilor de producţie.

După cum a fost stabilit sistemul de interes este constituit din două elemente, dispozitivul şi aplicaţia mobilă. Aceste elemente la rândul lor pot fi considerate sisteme, aşa cum se doreşte ca ele să poată funcţiona independent unul faţă de altul într-o oarecare măsură. De asemenea un aspect important al sistemului este autonomia faţă de conexiunea la internet, sistemul nu trebuie să depindă de aceasta pentru a funcţiona, această problemă a fost identificată în cadrul aplicaţiei mobile de la Garmin. În subcapitolele ce urmează, vor fi expuse cerinţele funcţionale ce descriu comportamentul şi cerinţele non funcţionale ce descriu calităţile, pentru sistemul în ansamblu. Cerinţele pentru fiecare element în parte pot fi identificate din aceste cerinţe, însă specificarea lor în parte nu oferă multe beneficii proiectului, aşa cum acesta nu este foarte complex.

2.1 Cerinţe funcţionale

- CF1: Sistemul trebuie să fie capabil să monitorizeze parametrii mediului. Acesta va colecta date despre temperatura aerului, umiditatea relativă a aerului, compuşii volatili organici, intensitatea luminii şi nivelul de zgomot. Sistemul poate măsura şi alţi parametri dacă senzorii utilizaţi pot măsura şi alţi parametri. Printre aceşti parametri trebuie să se numere şi nivelul bateriei.
- CF2: Eşantionarea datelor trebuie să aibă o recurenţă stabilită dar configurabilă, pentru fiecare parametru în parte. Configurarea recurenţei trebuie să poată fi efectuată în timpul operaţiei sistemului. Rata de eşantionare configurată trebuie să fie validată, având ca limite admisibile, minimă de 30 secunde şi maximă de 24 de ore. Sistemul trebuie să permită dezactivarea monitorizării parametrilor nedoriţi.
- CF3: Sistemul trebuie să asigure persistenţa datelor. Persistenţa datelor trebuie să fie asigurată local, fără servicii de la al furnizori terţi. Baza de date utilizată trebuie să fie SQLite. Sistemul trebuie să ofere posibilitatea de a exporta datele, fie în formatul SQLite sau CSV.
- CF4: Sistemul trebuie să ofere posibilitatea de a seta alarme. În figura 2.2 este reprezentat modul de configurare dorit al alarmei. Interfaţa de utilizator trebuie să permită selectare unui interval şi alarma să fie activată dacă valoarea actuală este în interiorul sau exteriorul intervalului în dependenţă de setare. Dacă într-o alarmă au fost adăugaţi mai mulţi parametri acesta va avea efectul de **and** în programare. Iar dacă vor fi create mai multe alarme cu parametri diferiţi acestea vor avea efectul de **or**. În caz că există mai multe dispozitive înregistrate în sistem, o alarmă poate fi activată pentru mai multe dispozitive.
- CF5: Sistemul trebuie să ofere posibilitatea de a vizualiza a datelor prin mai multe reprezentări grafice. Este necesară reprezentarea fiecărui parametru într-o listă sau grilă, şi de-asemenea de prezentat grafice liniare sau radar.
- CF6: Componentele sistemului adică dispozitivul şi aplicaţia mobilă trebuie să comunice printr-un protocol propriu de transfer de date, care să funcţioneze la un nivel mai sus de protocolul de reţea Bluetooth. Interacţiunea cu dispozitivul va fi realizată prin intermediul aplicaţiei mobile.
- CF7: Sistemul trebuie să fie inter-operabil cu alte sisteme existente. Pentru realizarea acestei funcţionalităţi sistemul va trebui să ofere o interfaţă ce utilizează protocolul MQTT. În configurarea conexiunii, vor fi prezente câmpuri precum URL-ul, cheia API, numele de utilizator şi parola. Unele câmpuri pot fi lăsate necompletate însă conexiunea trebuieşte validat.
- CF8: Interfaţa de utilizator trebuie să fie minimalistă şi să nu distragă utilizatorii din scopurile lor pentru a utiliza sistemul. În figurile 2.1 şi 2.2 este prezentat un prototip iniţial al paginilor aplicaţiei mobile, realizat în Figma. La lansarea aplicaţiei, va fi afişată o pagină de pornire pe care este scrisă denumirea aplicaţiei mobile. Apoi, urmează pagina de pornire, care va varia în funcţie numărul de dispozitive conectate la aplicaţia mobilă. În cazul în care există mai multe dispozitive

conectate, aplicația va prezenta o pagină de pornire cu o vizualizare succintă pentru toate dispozitivele conectate, oferind astfel o imagine de ansamblu a stării acestora. În situația în care este conectat doar un dispozitiv, interfața va afișa pagina cu detalii specifice despre acel dispozitiv pentru a ușura utilizarea și pentru a furniza o experiență mai concentrată. La apăsarea pe un parametru, trebuie să fie afișat un grafic liniar cu evoluția acestuia în timp. Pe pagina principală, utilizatorii vor avea posibilitatea de a conecta noi dispozitive, facilitând astfel extinderea rețelei de monitorizare. De asemenea, în partea de sus stângă a interfeței va există butonul de setări care deschide pagina de setări, unde vor putea fi adăugate conexiuni MQTT, și va putea fi configurată recurența eșantionării. Pe pagina principală în partea dreaptă sus va fi un buton pentru alarme, care va deschide pagina de alarme, în care vor putea fi create, activate, dezactivate, editate și șterse alarmele.

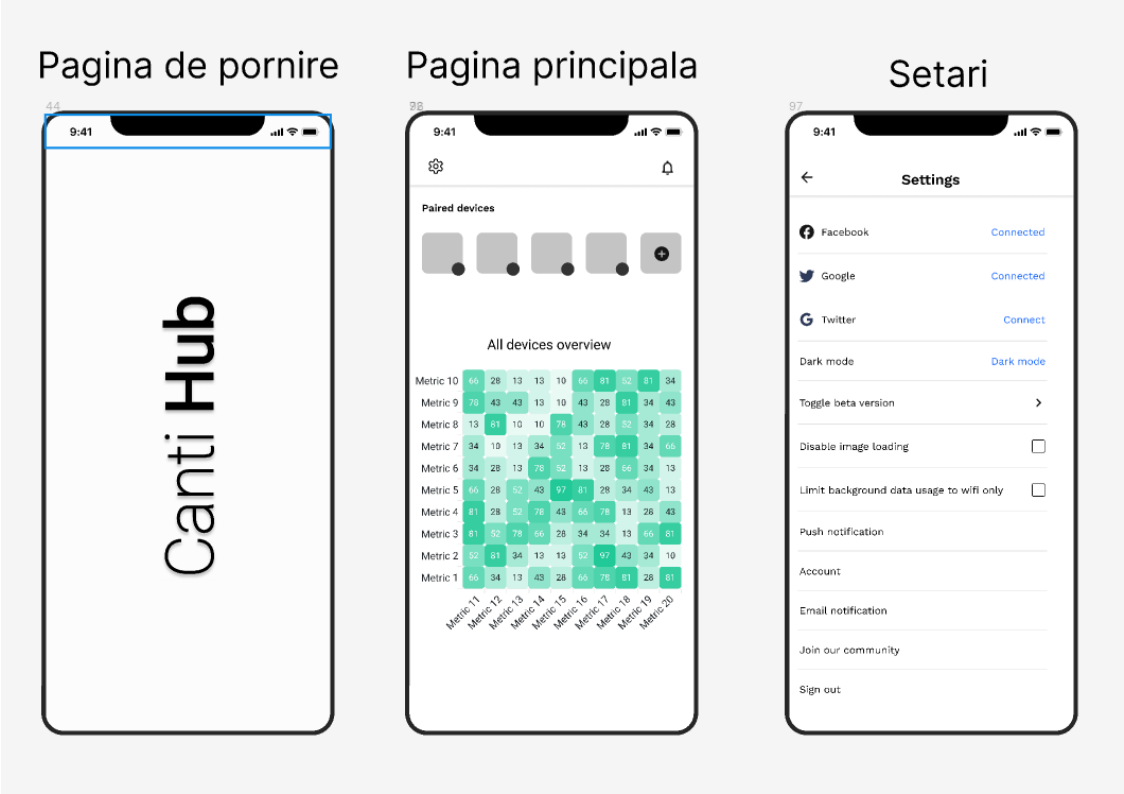

Figura 2.1 - modelul paginii de pornire, principale și setări

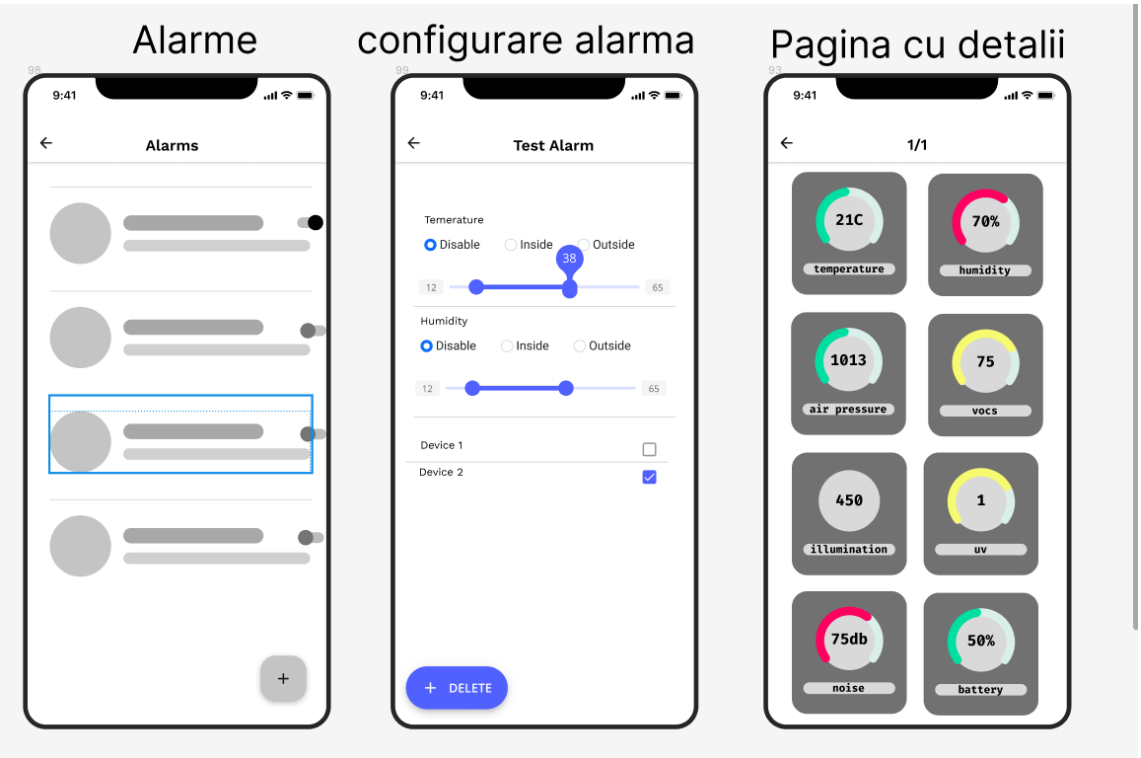

Figura 2.2 - modelul paginii pentru alarme, configurări şi detalii

2.2 Cerinţe non-funcţionale

- CNF1: Sistemul trebuie să aibă o autonomie ridicată. În primul rând, funcționarea sistemului nu trebuie să depindă de conexiunea la internet. În al doilea rând, părțile componente ale sistemului nu trebuie să depindă de conexiunea Bluetooth pentru a funcționa. Sistemul trebuie să aibă o durată optimă de operare, de cel puțin 30 de zile fără a necesita reîncărcarea dispozitivului. În plus, sistemul trebuie să poată detecta şi trata potențialele erori pentru a rămâne funcțional.
- CNF2: Sistemul trebuie să fie scalabil, în sensul că la o aplicație mobilă să poată să fie conectate mai multe dispozitive.
- CNF3: Interoperabilitatea va fi asigurată, prin implementarea unei interfețe MQTT pentru a posta sau prelua date de la sisteme externe.
- CNF4: Asigurarea unui nivel minim de securitate este esențială. Conectarea la dispozitiv trebuie să fie realizată printr-un mecanism de autorizare.
- CNF5: Sistemul trebuie să fie adaptabil. Trebuie să poată suporta diferite versiuni de hardware şi software pentru dispozitivele conectate, iar acestea trebuie identificate şi gestionate corespunzător.
- CNF6: Codul trebuie să fie cât mai portabil pentru a putea fi uşor adaptat şi portat pe alte platforme, în cazul în care este necesar. Este de dorit să se utilizeze un alt limbaj de programare decât C.
- CNF7: Prețul de producție al unui dispozitiv trebuie să fie minimizat, dar totodată să se țină cont de numărul de parametri pe care acesta îi poate măsura.
- CNF8: Dimensiunile dispozitivului nu trebuie să depăşească dimensiunile de 86 x 50 x 22 mm.

- CNF9: Realizarea sistemului trebuie să fie versionată cu ajutorul instrumentului Git. Și să se respecte convenția de mesaje **conventional commit**.
- CNF10: Sursele sistemului trebuie să fie găzduite pe platforma Github, după posibilitate să fie implementate implementate pipeline-uri pentru integrare continuă și livrare continuă. Pentru a facilita procesul de CI și CD pot fi utilizate containere docker.
- CNF11: Realizarea sistemului va avea loc după modelul de dezvoltare secvențial – cascadă. Acesta va permite dezvoltatorului să se concentreze pe o sarcină în loc să fie distras de mai multe activități distincte.
- CNF12: Sistemul trebuie să fie simplu, să nu fie supra-proiectat. Acesta poate fi complicat pe urmă după ce se află într-o stare livrabilă.

3 MODELAREA ŞI PROIECTAREA SISTEMUL

Pentru modelarea sistemului, au fost considerate mai multe limbaje de modelare. Acestea fiind: UML, SysML, Archimate, AADL şi OPM. La prima vedere AADL [17] părea a fi o alegere mai bună, considerând că sistemul va avea şi partea hardware. Acesta ar fi permis abordare de inginerie a sistemelor bazată pe modele, care permite simulări. Totodată acest limbaj oferă atât limbaj textual cât şi grafic. Însă din cauza că pe internet este un număr restrâns de resurse pentru a în învăța nu s-a optat pentru a îl utiliza. A doua ce-a mai bună abordare ar fi de a modela sistemul pe foi de hârtie utilizând un pix, ca limbaj să se folosească ceva improvizat, însă această abordare nu ar fi acceptabilă în ipostaza proiectului respectiv. Ca urmare a acestor fapte s-a optat pentru utilizarea limbajului UML [18] – limbaj de modelare unificat. Deşi iniţial acesta era prevăzut doar pentru sistemele intensive de software, în următoarele reviziuni a fost introdusă posibilitatea de a modela sisteme complete. Deşi unii încă au rămas nemulţumiți şi au creat limbajul derivat SysML care se concentrează pe sisteme dar nu software. Ca instrument pentru modelare sistemului a fost ales Modelio [19], acesta a fost ales datorită faptului că este gratis, cu, codul sursă deschis. Se instalează şi rulează pe maşina locală fără a avea nevoie de acces la internet. Deşi acest instrument nu este perfect şi are probleme cu reprezentarea unor diagrame, faptul că are o interfață de utilizator intuitivă şi simplă îi oferă un avantaj faţă de soluţiile comerciale.

În specificaţiile actuale de UML [18] sunt definite 14 diagrame [20], care sunt împărţite în două tipuri, diagrame comportamentale şi diagrame structurale. Aceste diagrame sunt prevăzute pentru diverse cazuri, de aceia pentru proiectarea sistemului au fost utilizate doar 8 tipuri de diagrame. Aceste diagrame sunt:

- diagrama cazurilor de utilizare;
- diagrama de activitate;
- diagrama de secvenţă;
- diagrama maşină de stări;
- diagramă de comunicare;
- diagramă de clase;
- diagramă de componente;
- diagramă de implementare.

Informația ce-a mai relevantă despre sistem este în diagrama de amplasare, aşa cum acest sistem are şi o parte hardware. Din această poate fi dedus faptul că topologia sistemului va fi stea. Însă din cauza limitărilor limbajului UML, acest lucru ar putea fi sărit cu vederea.

Deşi modelarea are loc înainte de realizare, în timpul dezvoltării pot fi identificate probleme în model, sau limitaţii tehnice. Ca urmare modelul va suferi schimbări, pentru a ilustra structura şi comportamentul sistemului în realitate.

3.1 Descrierea comportamentală a sistemului

Aşa cum a fost menţionat, limbajul UML are un set de diagrame care descriu comportamentul sistemului. Aceste diagrame sunt:

- diagrama cazurilor de utilizare;
- diagrama de activitate;
- diagrama de secvenţă;
- diagrama maşină de stări;
- diagramă de comunicare;
- diagramă de interacţiune;
- diagramă de sincronizare.

Diagrama cazurilor de utilizare, descrie ce face un sistem, însă nu descrie modul de implementare. În această diagramă este prezentat un set de evenimente care pot fi declanşate de către actor. Actorul în această diagramă poate fi o persoană, un rol, sau chiar şi un sistem. Prin această diagramă poate fi prezentată imaginea generală a sistemului. Însă de multe ori este prezentată doar imaginea unui om care ţine nişte balonaşe.

Diagrama de activitate prezintă fiecare pas pentru a atinge un obiectiv în cadrul unui sistem. Această diagramă este potrivită pentru reprezentarea algoritmilor. Fiecare activitate are un punct de intrare şi un punct de ieşire. Iar fiecare acţiune din interior, poate fi la rândul ei un proces.

Diagrama de secvenţă este potrivită pentru a arăta interacţiunea dintre mai multe obiecte sau actori. Aceasta are linii de viaţă din care sunt trase mesaje către alte linii. Această diagramă prezintă în mod cronologic mesajele care au avut loc între obiecte şi actori.

Diagrama maşină de stare este utilizată pentru a modela comportamentul unui obiect complex. Aceasta prezintă care sunt evenimentele care pot schimba starea obiectului şi ce acţiuni sunt întreprinse când o stare este activă.

Diagrama de comunicare pune accent pe comunicarea dintre obiecte. Aici sunt trase asocieri între obiecte, iar mesajele şi direcţiile acestora sun puse lângă linie.

Diagrama de interacţiune adesea este complexă şi este evitat de nespecialişti. Aceasta utilizează aceleaşi elemente ca şi diagrama de activitate cu diferenţa că sunt adăugate elemente cum ar fi interacţiunea, constrângeri de timp sau de durată.

Diagrama de sincronizare cunoscută şi cu numele de diagramă de evenimente prezintă cronologia în are obiectele şi actorii acţionează. Se pune accentul pe durata evenimentelor şi modificările care se produc. Însă din cauza că Modelio nu le suportă. Această diagramă nu a fost utilizată în modelarea sistemului.

3.1.1. Imaginea generală asupra sistemului

Pentru a utiliza sistemul, utilizatorul trebuie să dețină cel puțin un dispozitiv. Apoi acesta va trebui să îl înregistreze în sistem. Odată ce dispozitivul este înregistrat în sistem utilizatorul poate vizualiza datele monitorizate pe acesta. Poate export datele respective. De asemenea mai există și posibilitatea de a seta alarme. Aceasta este imaginea generală a sistemului, reprezentarea ei grafică poate fi vizualizată în figura 3.1.

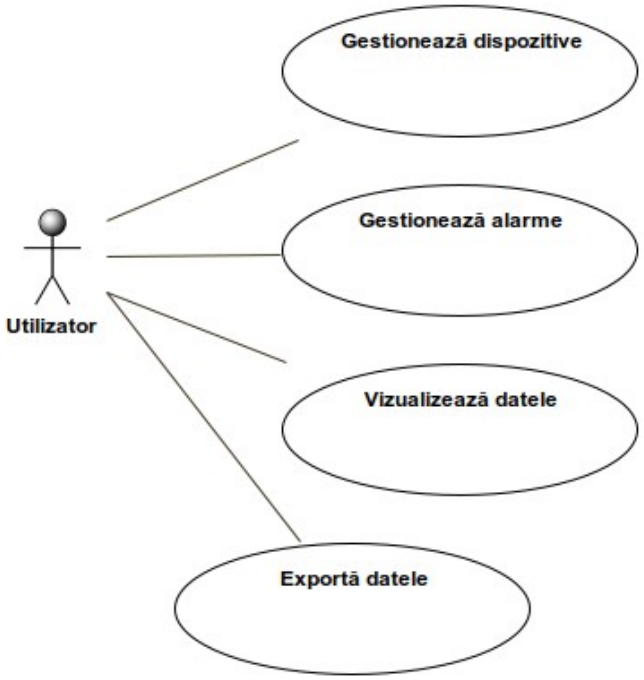

Figura 3.1 - Utilizarea sistemului

Sistemul va permite vizualizarea datelor în mai multe moduri. Cel mai de bază mod va fi fie o listă fie o grilă cu fiecare parametru monitorizat, va fi o pagină dedicată fiecărui dispozitiv. Altă reprezentare va fi o diagramă de tip arie, normalizată astfel în cât să poată fi identificat cu ușurință dacă un parametru este prea mare sau prea mic. În ultimul rând va fi prezentă posibilitatea de a vizualiza fiecare parametru în parte pe un grafic cu linii, pentru a putea vedea cum a evoluat acest parametru. Reprezentarea acestor cazuri de utilizare poate fi vizualizată în figura 3.2.

Figura 3.2 - Vizualizarea datelor

În figura 3.3 este prezentat mai detaliat modul de gestionare al sistemului. În primul rând, aşa cum a fost menţionat anterior, utilizatorul poate adăuga dispozitive în sistem. Totodată utilizatorul le poate şterge. Un alt aspect important al gestionării sistemului este configurarea recurenţei. Utilizatorul poate configura recurenţa pentru fiecare parametru în parte. Respectând condiţia că aceasta să se încadreze în intervalul 29 secunde, 24 ore. În cadrul acestui sistem, utilizatorul poate gestiona alarme, asta presupune crearea, ştergerea, dezactiva şi activa alarmele. De asemenea el le poate edita. Toate aceste setări sunt transmise şi către dispozitiv dacă bluetoohul este activat şi dispozitivul conectat este în raza de acţiune.

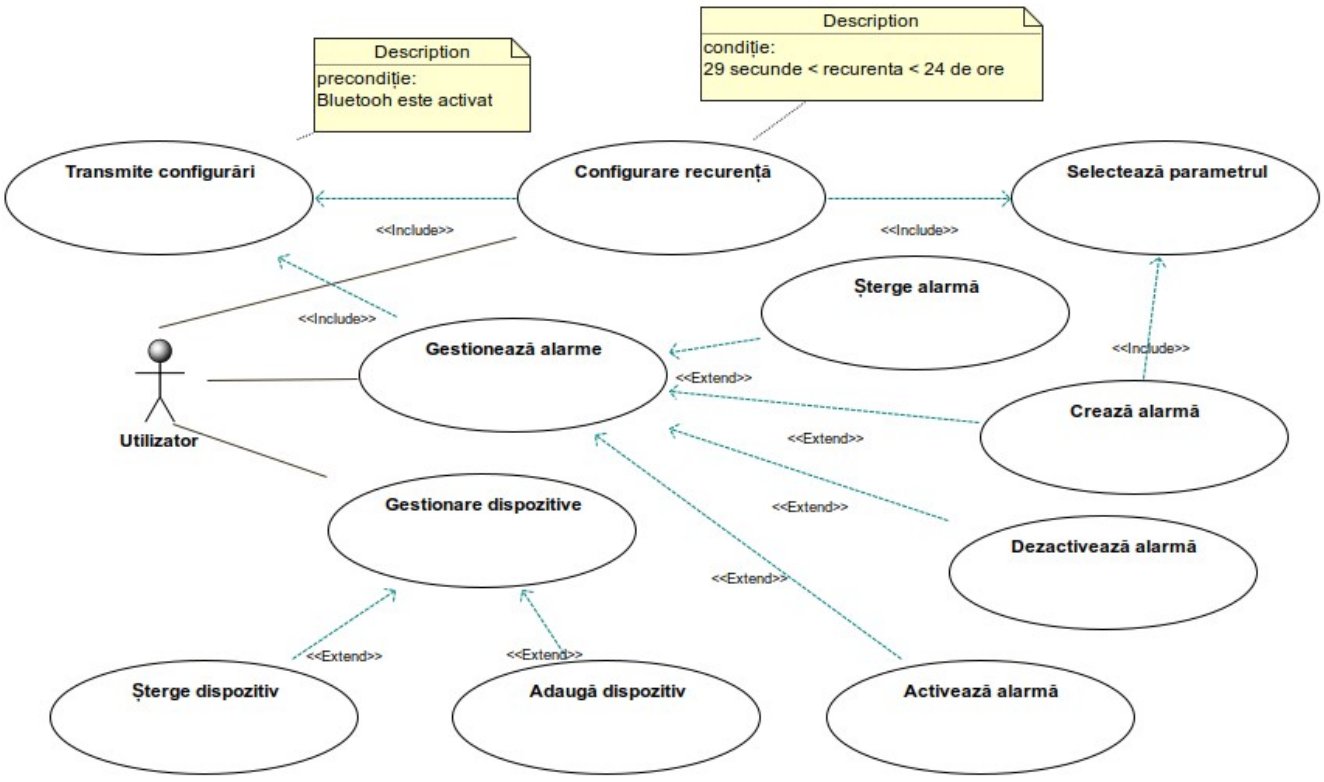

Figura 3.3 - Gestionarea sistemului

În figura 3.4 avem ilustrat faptul că sistemul este inter-operabil. Adică putem conecta sistemul nostru la un alt sistem. Acest lucru va avea loc prin intermediul protocolului de comunicare MQTT. Aplicația va fi principalul publiator către această platformă externă. Pentru ca dispozitivul să poată transmite datele prin MQTT, acesta va trebui să fie în regim staționar. Regimul staționar presupune că dispozitivul este conectat la o sursă de alimentare.

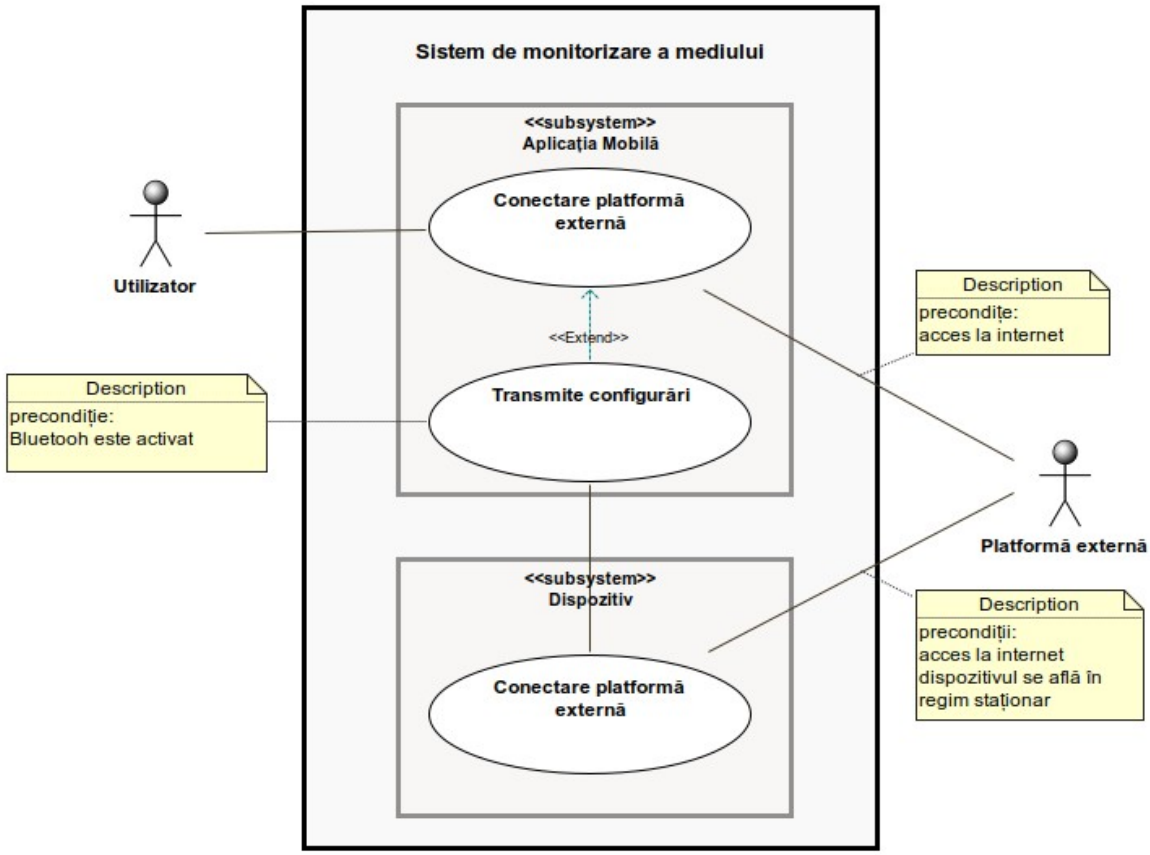

Figura 3.4 - Conectarea la o platformă externă

În figura 3.5 este prezentat modul în care software-ul sistemului poate fi actualizat. În primul rând dezvoltatorul trebuie să publice o versiune nouă de software și să o marcheze corespunzător. Apoi GitHub va compila software-ul fie că acesta este pentru dispozitiv sau telefon mobil. Odată ce utilizatorul va verifica dacă există o nouă versiune software, aplicația îl va înștiința de acest lucru. Versiunea nouă de software va fi descărcată. Utilizatorul va trebui să instaleze manual noua versiune de aplicație mobilă. Noua versiune de software pentru dispozitiv va fi transmisă prin Bluetooth către dispozitiv. Dacă acesta avea cel puțin 70% de baterie, actualizarea va avea loc. Dacă nu utilizatorul va fi înștiințat de acest lucru.

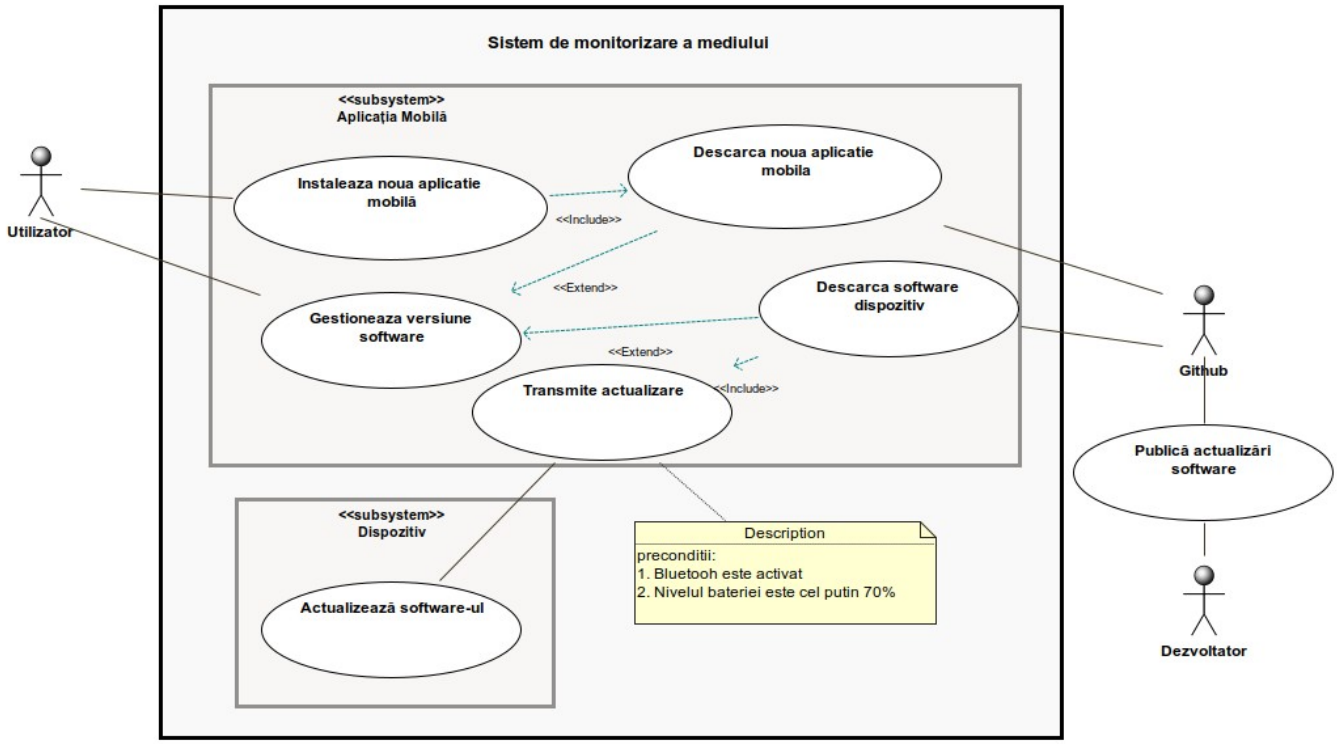

Figura 3.5 - Actualizarea versiunii software

3.1.2. Modelarea vizuală a fluxurilor

Dispozitivul va folosi mai mulți senzori pentru a citi încă și mai mulți parametri. Însă algoritmul pentru această activitate este similar. Primul pas este să fie inițializat senzorul. Aici există posibilitatea ca senzorul să fie defect, ceea ce va genera un mesaj de eroare. În cazul când senzorul este funcțional, se începe resecretizarea parametrului. Se așteaptă finisarea operației. Însă în acest timp pot fi efectuate și alte operațiuni. Spre exemplu pot fi inițializați și alți senzori. Odată ce parametrul a fost convertit, software-ul îl poate citi. În figura 3.6 este reprezentat acest proces.

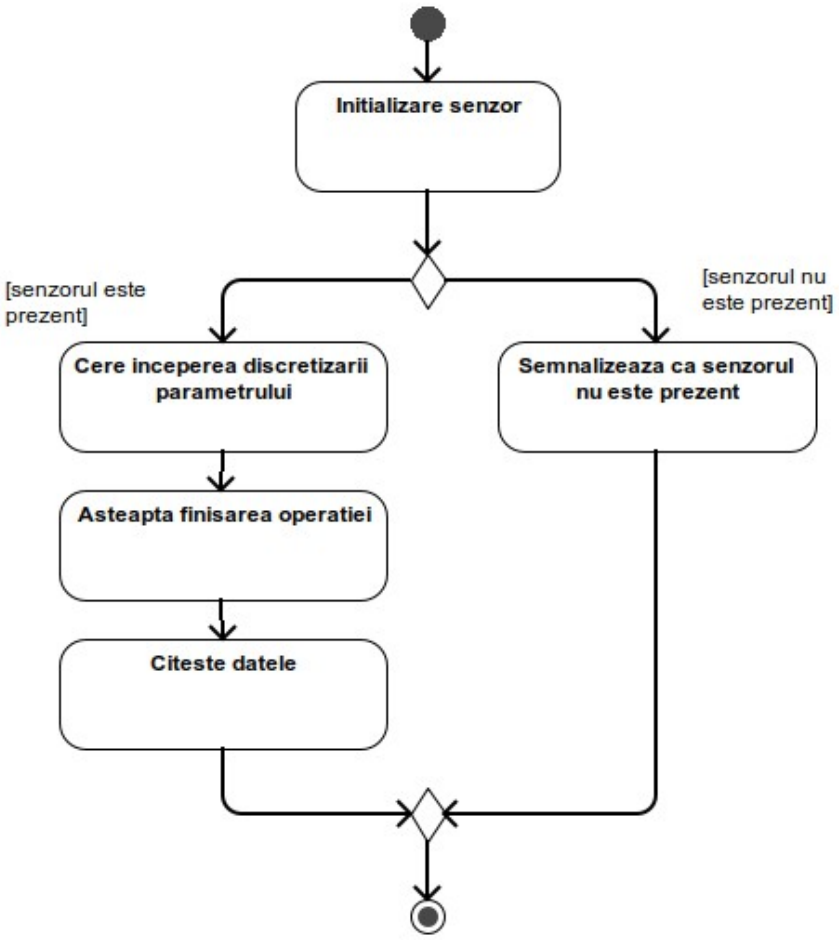

Figura 3.6 - Citirea parametrului

În figura 3.7 sunt prezentați pașii pentru a evalua alarmele. În primul rând sunt determinate care alarme sunt activate, pentru a nu procesa un număr mai mare de alarme. Asta este pe partea de aplicație mobilă, fiind că pe partea de dispozitiv va fi permis de setat doar 3 alarme. Următorul pas este de a identifica dacă condițiile pentru măcar o alarmă sunt îndeplinite. Condiția prezintă un interval în care valoarea parametrului curent trebuie să se încadreze sau să nu se încadreze pentru a genera alarma. Acest comportament este specificat de utilizator. Apoi dacă există măcar o alarmă care să fie activă, va fi generat un semnal de alarmă. Pe aplicația mobilă poate fi prezentată numele alarmei.

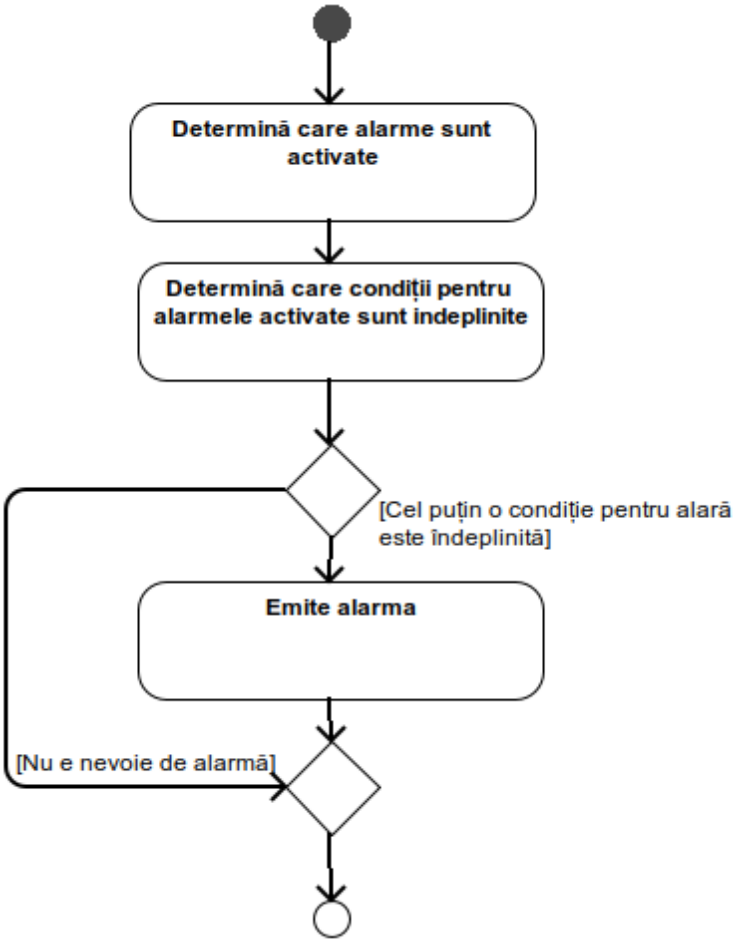

Figura 3.7 - Evaluare alarme

Exportarea datelor este un proces simplu. În primul rând datele sunt extrase din baza de date. Apoi în dependență de formatul selectat de către utilizator, datele sunt transformate într-un fișier de acest format. Formatele necesare de implementat sunt CSV și SQLite, însă se poate de implementat și alte formate. În ultimul rând utilizatorul este întrebat unde dorește să salveze fișierul. Acest proces poate fi vizualizat în figura 3.8.

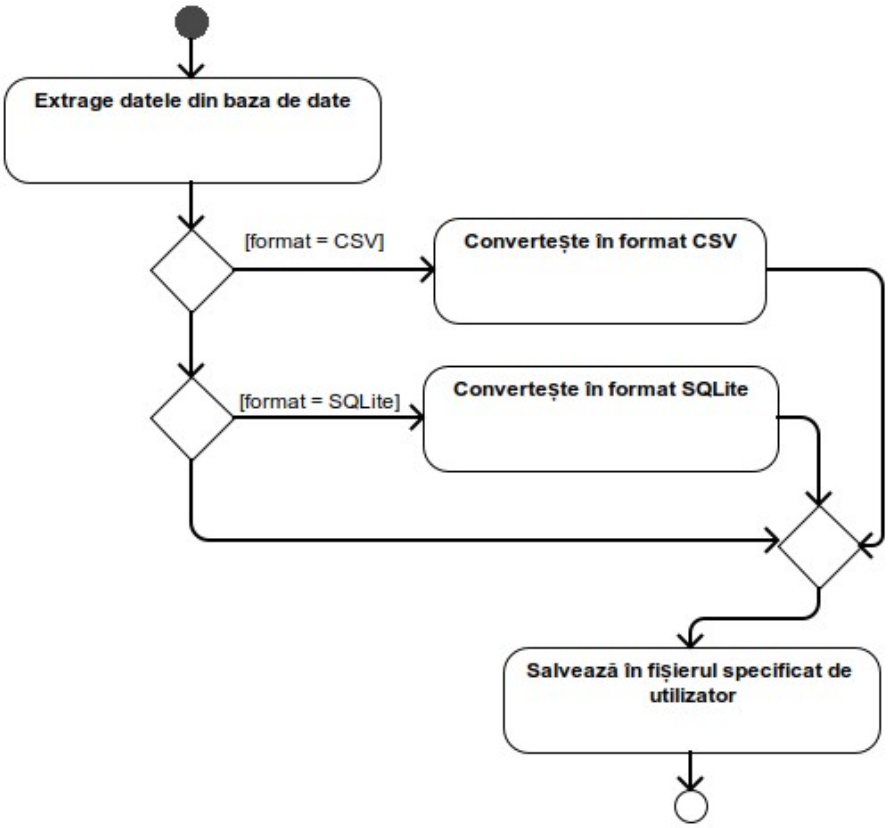

Figura 3.8 - Exportarea datelor

3.1.3. Stările de tranzacţie a sistemului

Dispozitivul poate se poate afla în patru stări majore. Stările sunt Iniţializare, AutoTestare, Operaţie şi Somn. Dispozitivul ajunge în cadrul stării de Iniţializare odată ce acesta este pornit. În cadrul acestei stări microcontrolerul va iniţializa periferialele sale şi senzorii. Dacă iniţializarea va avea loc cu succes dispozitivul va trece în starea de AutoTestare, dacă nu acesta se va stinge. Când se află în starea de AutoTestare microcontrolerul va testa periferialele sale, integritatea senzorilor şi a altor componente. În caz că testarea rezultă în erori dispozitivul se va duce înapoi în sarea de Iniţializare. Însă dacă mai mult de trei cicluri au fost cu eroare, dispozitivul se va stinge. Dacă testarea s-a finisat cu succes, atunci dispozitivul trece în starea de Operaţiune în cadrul căreia el va colecta datele şi le va transmite după posibilitate. Totodată pentru a reduce din consumul de energie, aşa cum de regulă citirea unui parametru durează câteva milisecunde iar restul, cel puţin 29 de secunde nu va fi nimic de făcut, dispozitivul se va duce în starea de Somn. În aceasta majoritatea periferialelor dispozitivului sunt deconectate. Apoi după ce a expirat timpul de inactivitate, dispozitivul se întoarce în starea de Operaţiune. În figura 3.9 pot fi vizualizate aceste stări în mai puţine cuvinte.

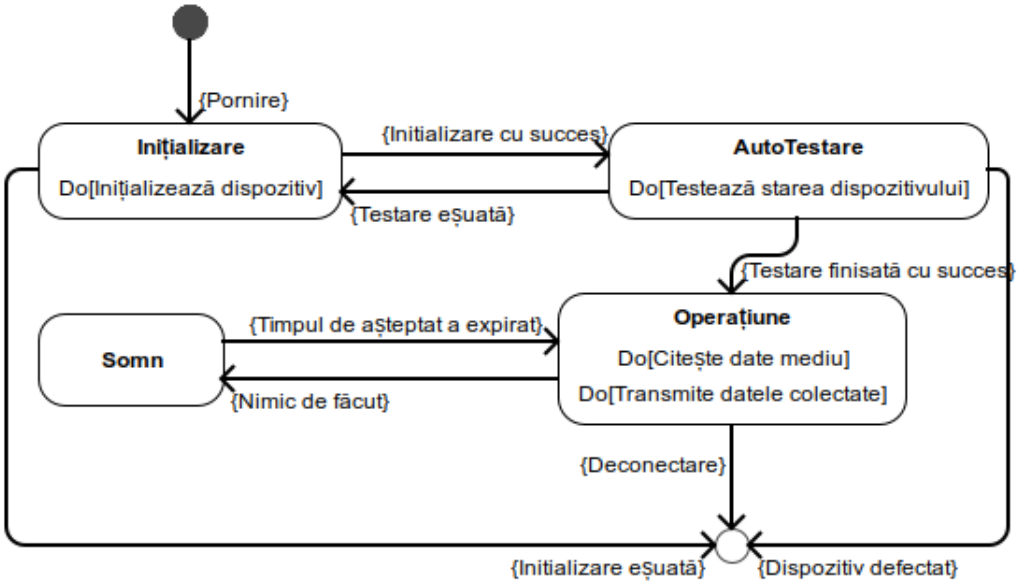

Figura 3.9 - Stările dispozitivului

În figura 3.10 sunt prezentate stările parcurse atunci când dispozitivul se află în starea de Operațiune. În primul rând în starea de Configurare se începe convertirea datelor pe senzori. Apoi se trece în starea de așteptare, în care se așteaptă finisarea convertirilor. După ce convertirea a fost efectuată, datele pot fi citite. Apoi se trece la starea de Procesare, asta ar presupune filtrarea acestora. În caz că este posibil datele sunt transmise. Apoi dispozitivul va trece în starea de Somn.

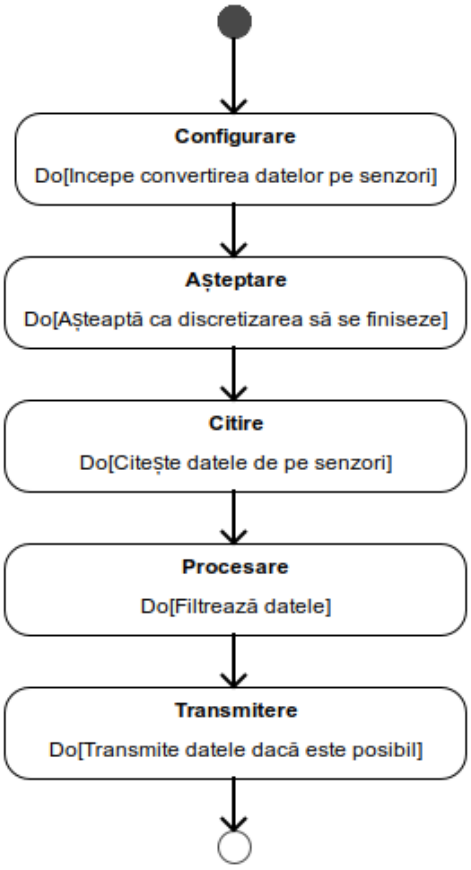

Figura 3.10 - Sub-stările etapei de operațiune

3.1.4. Descrierea scenariilor de utilizare a aplicaţiei

Dispozitivul este un element autonom al sistemului. Odată ce acesta este alimentat, el iniţiază senzorii şi apoi începe procesul de colectare al datelor. Acest proces repetitiv presupune iniţializarea discretizării parametrilor, citirea parametrilor şi salvarea acestora în caz că ei nu pot fi transmişi. Când utilizatorul are nevoie de aceste date, el va accesa aplicaţia mobilă. Aceasta la rândul ei va obţine datele de la dispozitiv şi le va afişa pentru utilizator. Acest scenariu poate fi vizualizat în figura 3.11.

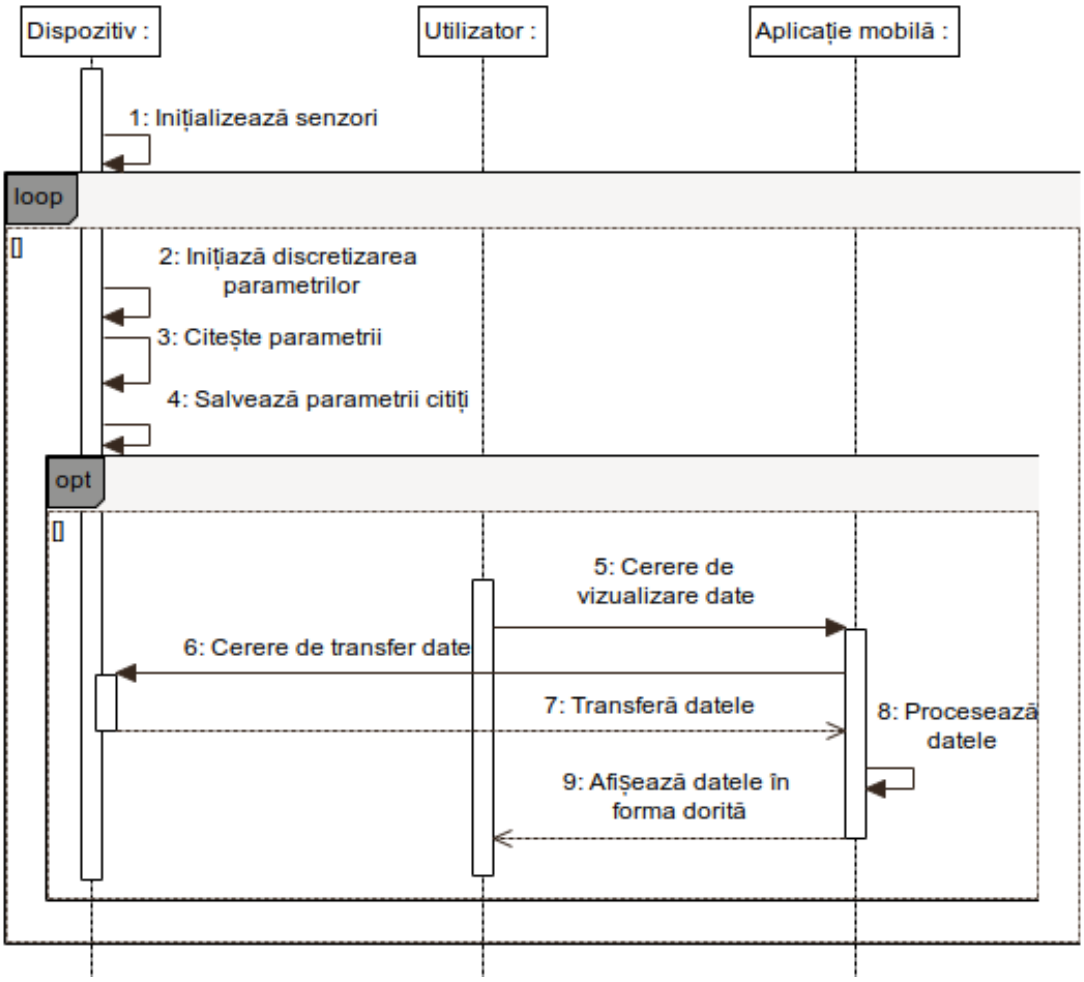

Figura 3.11 - Colectarea datelor

În figura 3.12 este ilustrat modul în care un dispozitiv nou este adăugat în sistem. Această interacţiune este iniţiată de către utilizator. Utilizatorul va trebui să selecteze ce dispozitiv doreşte să adauge în caz că sunt prezente mai multe dispozitive. Apoi aplicaţia va cere, ca utilizatorul să introducă cheia de acces către dispozitiv. Aplicaţia mobilă va transmite cererea de conectare împreună cu cheia de acces către dispozitiv. Acesta va evalua cheia şi în dependenţă de corectitudinea acesteia va accepta sau refuza conexiunea. Utilizatorul va fi înştiinţat de rezultatul acestei operaţii.

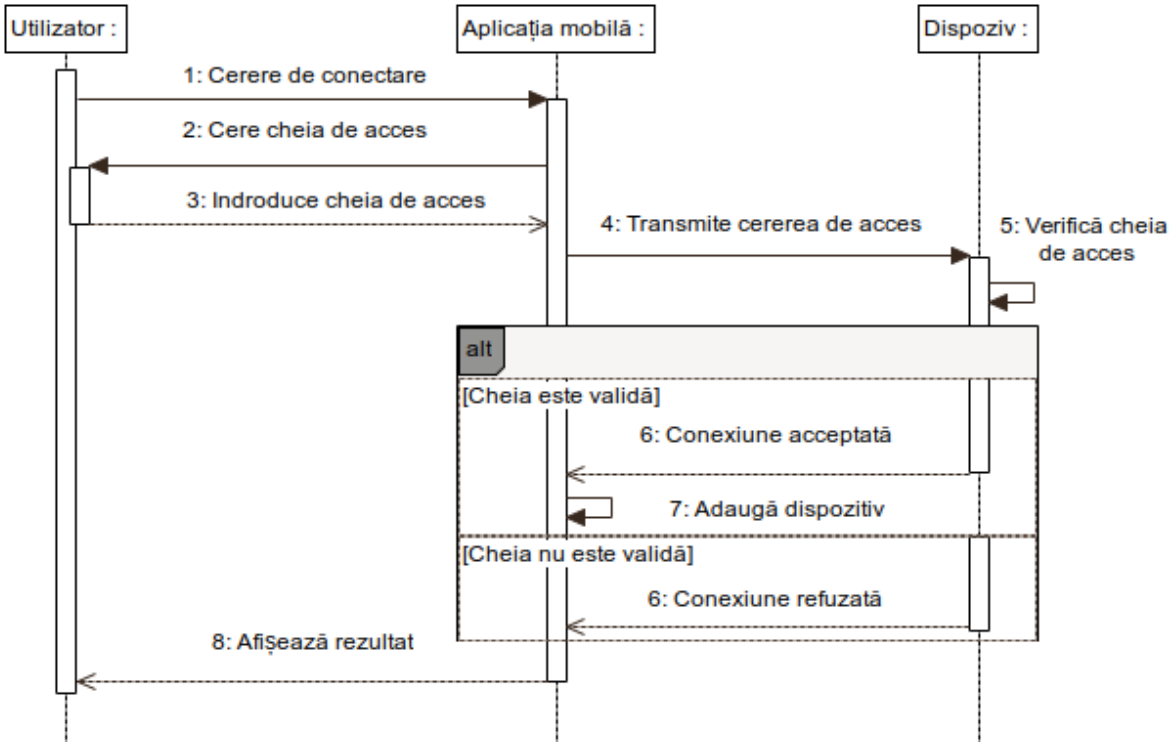

Figura 3.12 - Conectarea unui nou dispozitiv

3.1.5. Fluxurile de mesaje şi legăturile dintre componentele sistemului

În figura 3.13 este prezentată colaborarea dintre componentele fizice ale dispozitivului pentru a colecta datele. În primul rând, microcontrolerul va iniţializa senzorii digitali BME680 şi LTR 390UV. Apoi va citi parametrii respectivi. De pe senzorul BME680 vor fi citite temperatura, umiditatea, presiunea şi compuşii volatili organici. De pe senzorul LTR 390UV va fi citit nivelul de iluminare şi radiaţii ultra violete. În ultimul rând, microfonul ZTS6117 care este un senzor analog, va modifica tensiunea pe un pin, iar microcontrolerul o va citi. Apoi în urma unor calcule va fi identificat nivelul de gălăgie. Pentru a păstra datele, în cazul când nu există o conexiune Bluetooth, microcontrolerul va înscrie aceste date în regiunea de ram a RTC-ului, această regiune de RAM nu pierde alimentarea nici când microcontrolerul este în regim de somn, deci datele nu vor fi pierdute. Când va fi prezentă o conexiune, microcontrolerul va putea citi aceste date şi să le transmite către aplicaţia mobilă.

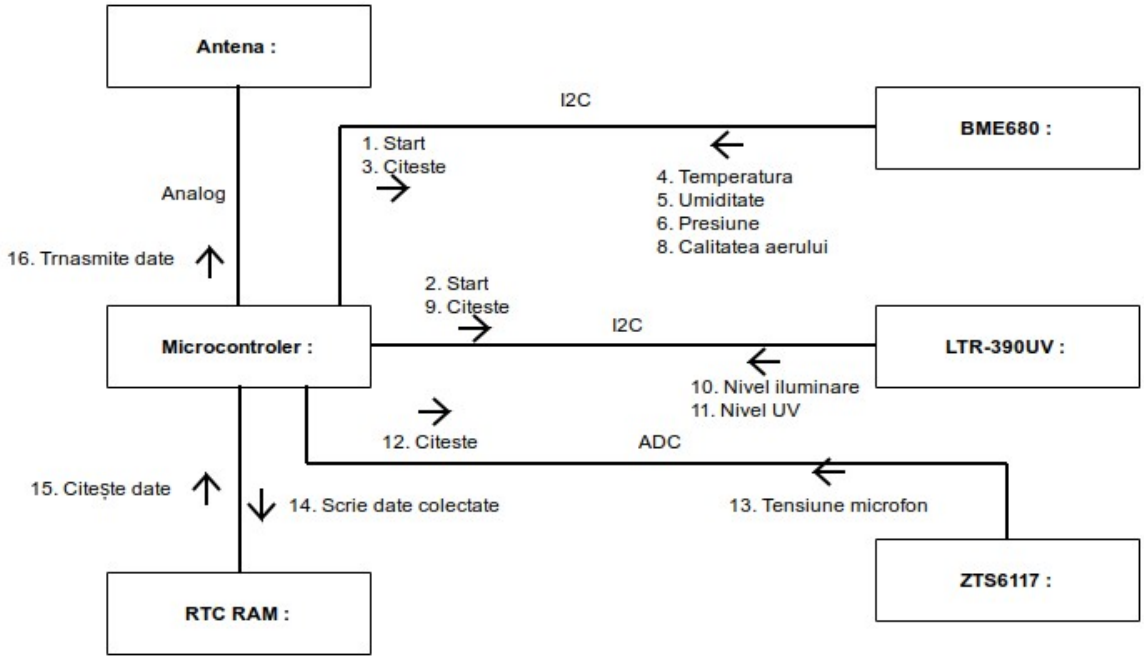

Figura 3.13 - Colectarea datelor

Pentru a adăuga un dispozitiv nou, utilizatorul trebuie să selecteze ce dispozitiv doreşte să adauge în caz că sunt prezente mai multe dispozitive. Apoi aplicaţia va cere, ca utilizatorul să introducă cheia de acces către dispozitiv. Aplicaţia mobilă va transmite cererea de conectare împreună cu cheia de acces către dispozitiv. Acesta va evalua cheia şi în dependenţă de corectitudinea acesteia va accepta sau refuza conexiunea. Utilizatorul va fi înştiinţat de rezultatul acestei operaţii. Acest proces poate fi vizualizat sub formă de colaborare în figura 3.14.

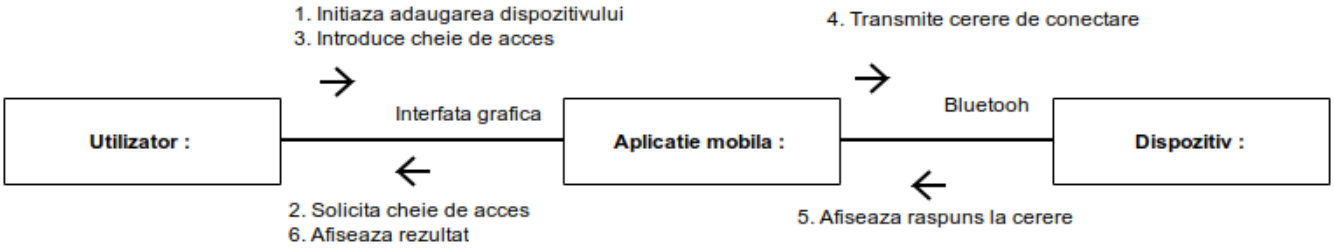

Figura 3.14 - Conectarea unui nou dispozitiv

3.2 Descrierea structurală a sistemului

Pentru a modela structura unui sistem utilizând limbajul de modelare UML, pot fi utilizate următoarele diagrame:

- diagramă de clasă;
- diagramă de obiecte;
- diagramă de componente;
- diagramă de implementare.

- diagramă de structură compozită;
- diagramă de pachete;
- diagramă de profiluri.

Diagrama de clasă este ce-a mai utilizată diagramă structurală în dezvoltarea software. Deşi în principal este utilizată pentru a reprezenta clase, poate fi utilizată şi pentru a reprezenta structuri şi enumeraţii pentru limbajele care nu au clase cum ar fi C. Cu acest tip de diagramă poate fi prezentat designul logic şi fizic al sistemului. Reprezentarea grafică a clasei este un dreptunghi cu trei secţiuni. În prima se regăseşte numele şi stereotipul clasei. În următoarea secţiune sunt prezentate atributele. În ultima secţiune sunt prezentate metodele clasei.

Diagrama de obiecte este utilizată pentru a instalaţia un exemplu al diagramei de clasă. Aceasta are ca scop să valideze corectitudinea diagramei de clasă.

Diagrama de componente prezintă grupările logice ale elementelor. Aici pot fi prezentate modulele software ale sistemului. Dar chiar şi componentele fizice ale acestora şi porturile lor. Fiecare component este reprezentat cu ajutorul unei casete rectangulare. Conectorii sunt interfeţe oferite sau cerute.

Diagrama de structură compozită este utilizată rareori în afara domeniului de dezvoltare software. Deoarece este mai detaliată decât diagrama de clase, descriind mai multe amănunte despre clase şi interacţiunile dintre acestea. Pentru majoritatea cazurilor această diagramă este una în plus.

Diagrama de implementare prezintă componentele hardware şi software ale sistemului. Componentele hardware sunt prezentate ca noduri iar componentele software sunt prezentate ca artefacte. Această diagramă prezintă modul de instalare al sistemului şi cum componentele sistemului sunt legate.

Diagrama de pachete este utilizată pentru a prezenta dependenţele dintre pachetele care compun un model. Scopul principal este de a prezenta relaţia dintre diferitele componente mari care formează un sistem complex.

Diagrama de profiluri este la fel de complexă ca un limbaj. Pentru această diagramă sunt create noi proprietăţi şi semnificaţii pentru diagramele UML existente. Aici se definesc noi stereotipuri. Profilurile sunt utile pentru a particulariza un model UML pentru o platformă specifică, cum ar fi Java sau .Net Framework.

3.2.1. Descrierea structurii statice a sistemului

Pentru implementarea senzorilor, indiferent de limbajul care va fi utilizat poate fi utilizat şablonul de proiectare hardware adaptor [21]. În figura 3.15 este prezentat faptul că toţi senzorii vor moşteni de la clasa senzor. Aceştia vor trebui să ofere o singură metodă pentru citirea parametrilor, însă aceasta va primi tipul parametrului care va fi citit. Astfel se va respecta şi principiul substituirii Liskov [22].

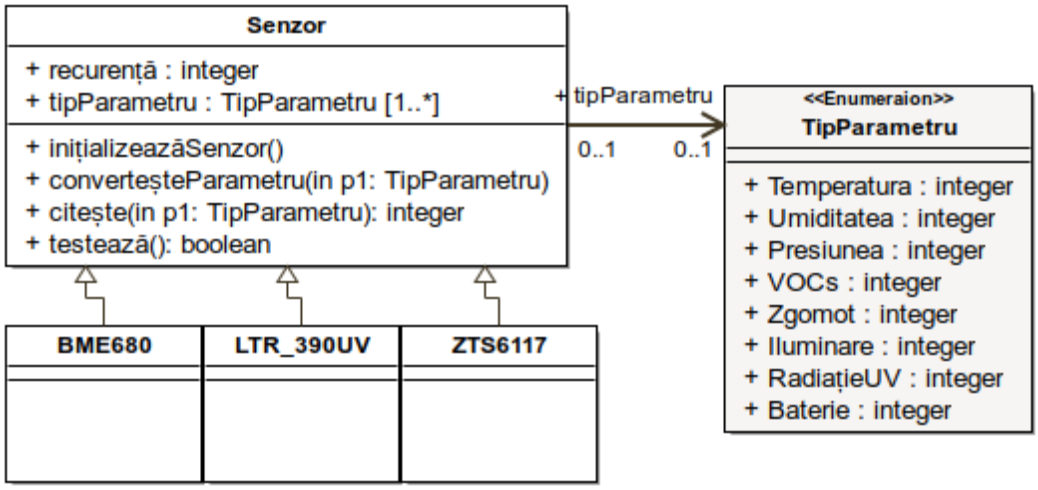

Figura 3.15 - Implementarea senzorilor

În cod va fi prezentă o clasă pentru generarea stărilor dispozitivului. În metodele acesteia se va scrie codul pentru inițializare, auto testare, operațiune și somn. Acestea vor fi implementate cu ajutorul mașinii de stări prezentată în figura 3.9. Cum poate fi observat în figura 3.16 această clasă va interacționa și cu alte clase cum ar fi clasa responsabilă de colectarea datelor, care va avea un șir de senzori. Și clasa de auto testare care va avea metode pentru a testa țâțe părțile componente ale dispozitivului.

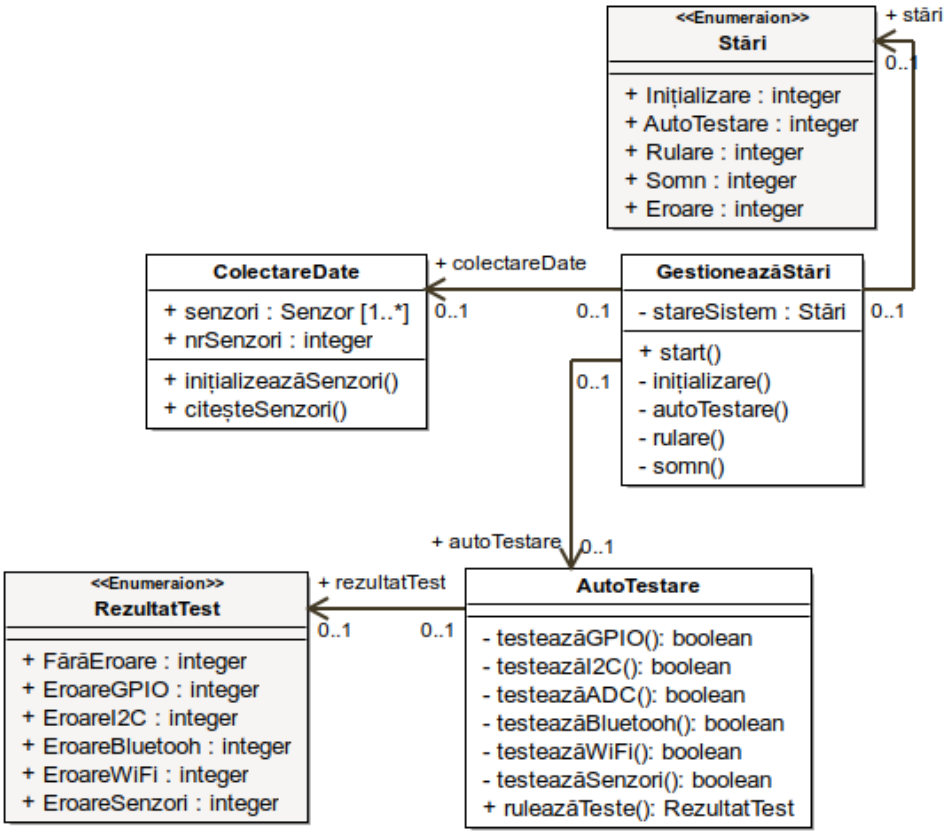

Figura 3.16 - Gestionarea stărilor dispozitivului

3.2.2. Relaţiile de dependenţă între componentele sistemului

Componentele software ale aplicației mobile sunt prezentate în figura 3.17. Acestea sunt logica de aplicație, baza de date, setări, prezentarea și comunicarea. Nivelul de prezentare va oferi o interfață pentru afișarea datelor. Nivelul de comunicare va permite stabilirea conexiunii cu dispozitivul, transmiterea de comenzi și preluarea datelor. Baza de date va permite stocarea datelor colectate pe termen lung. Componenta de setări va permite de a salva setările utilizatorului ca acestea să nu se piardă.

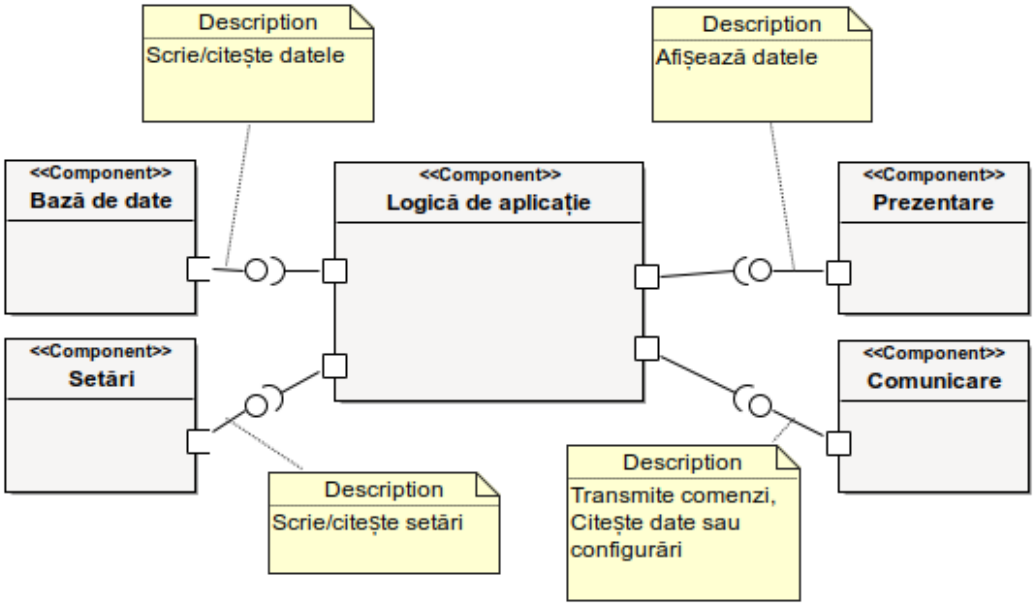

Figura 3.17 - Componentele aplicaţiei mobile

În figura 3.18 sunt prezentate componentele software ale ale dispozitivului. Acestea includ urătoarele componente: aplicație, persistență, comunicare, auto testare, driver LTR 390UV, driver BME680, driver ZTS6117. Componenta de persistență va avea rolul de a stoca setările, în principal cele ce țin de recurența eșantionării parametrilor, conexiuni Wi-Fi și MQTT, dar și stocarea datelor pentru recuperarea lor ulterioară. Componenta de comunicare va permite conectarea la rețele Wi-Fi și Bluetooth, dar și transmiterea datelor prin aceste canale. Componenta de auto testare va testa funcționalitatea dispozitivului și integritatea fiecărui senzor. Driverele pentru fiecare senzor va permite citirea datelor oferite de aceștia dar și identificarea dacă senzorul este în stare funcțională și nu are defecte.

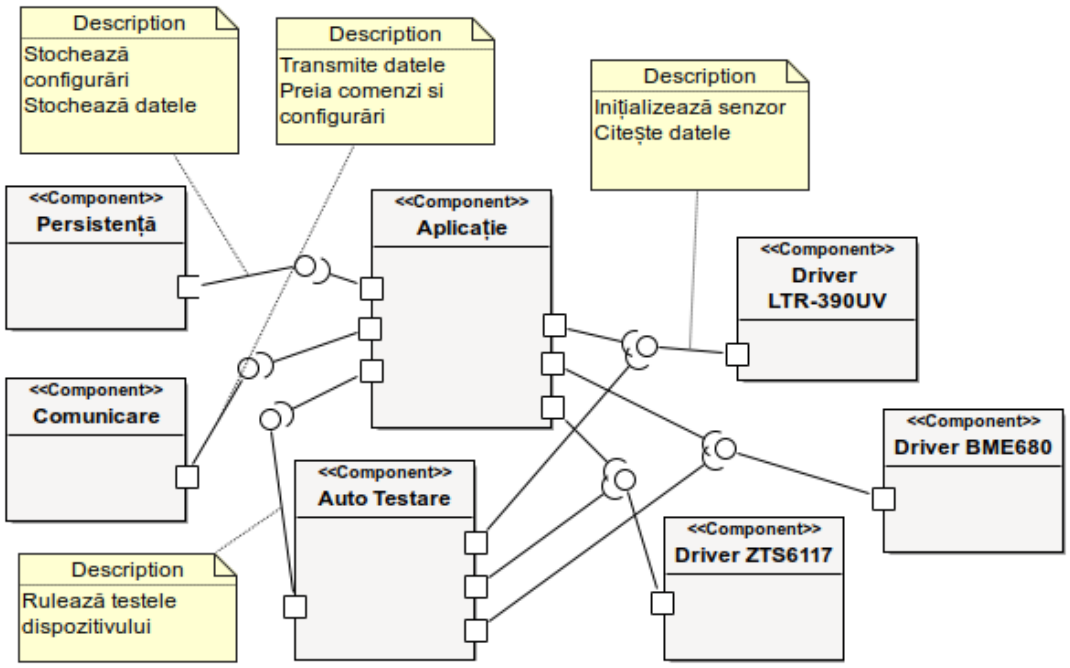

Figura 3.18 - Componentele software-ului dispozitivului

3.2.3. Modelarea echipamentelor mediului de implementare

În figura 3.19 este prezentată implementarea sistemului în ansamblu. Sistemul în stare funcțională este constituit din două elemente un dispozitiv și un telefon mobil. Nucleul dispozitivului este microcontrolerul, care va fula firmware-ul scris pentru acesta. La microcontroler vor fi conectați senzorii respectivi, BME680, ZTS6117 și LTR 390UV de pe care se vor citi parametrii de interes. Pe telefonul mobil, spre deosebire de dispozitiv va fi instalat și un sistem de operare, care va constitui mediul de execuție al aplicației. Acest sistem de operare va permite accesul aplicației la rețeaua Bluetooth, dar și la alte funcții, cum ar fi conexiunea la internet sau afișarea unei interfețe grafice. În cadrul acestui sistem pot fi conectate mai multe dispozitive. Totodată acest sistem ar putea să transmită sau să preia date de la un broker MQTT.

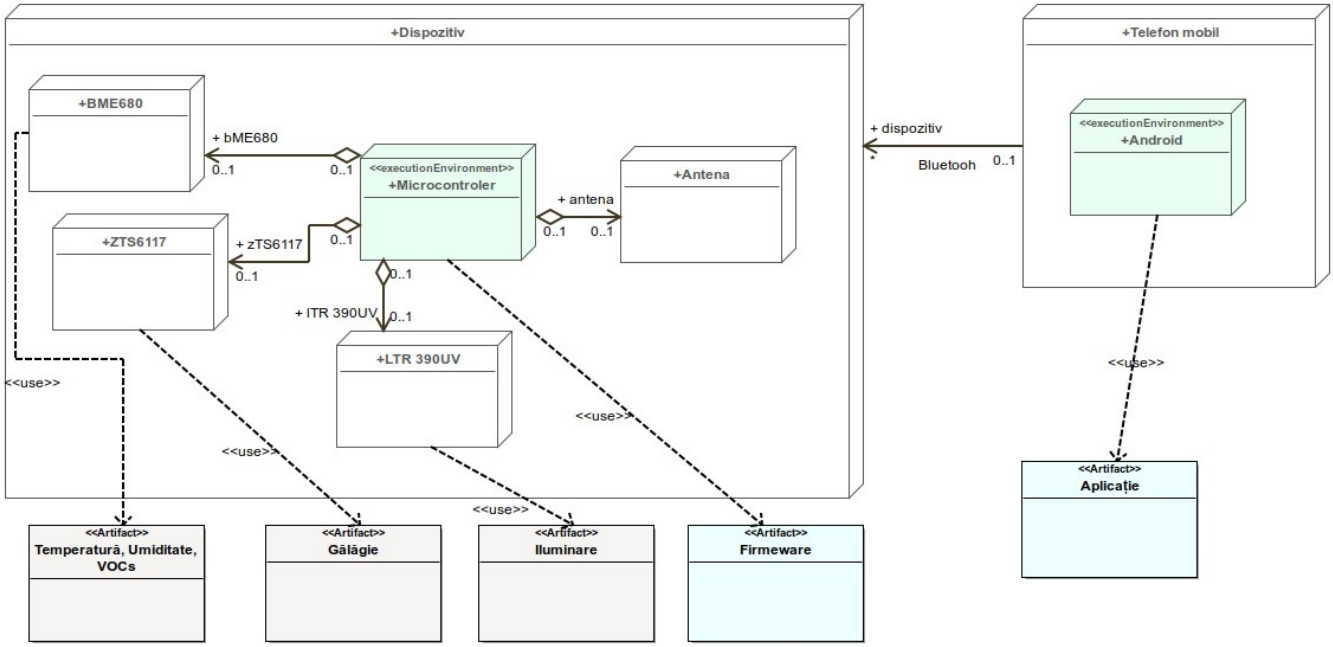

Figura 3.19 - Implementarea sistemului

4 REALIZAREA SISTEMULUI

Pentru realizarea sistemului au fost nevoie de mai multe instrumente software. S-a optat ca toate aceste instrumente să fie gratis pentru a nu fi nevoit a le pirata şi chiar şi mai bine să aibă codul deschis. În continuare este prezentată lista de software utilizată pentru a realiza sistemul:

- Gnu/Linux [23] – sistem de operare;
- terminal – pentru a accesa editorul de text, compilatoarele şi alte programe;
- Neovim [24] – editor de text hiper-extensibil bazat pe Vim [25];
- flutter-tools.nvim [26] – extensie Neovim pentru crearea aplicaţiilor flutter;
- Android Studio [27] – a fost utilizat emulatorul de android oferit de acesta;
- Git [28] – instrument pentru versionarea codului sursă;
- Github [29] – platformă online de găzduire a codului sursă.

Neovim este al doilea cel mai popular editor de text care rulează în terminal din anul 2023 [30]. Printre avantajele acestui editor este modul eficient de a edita textul fiind prezente mai multe moduri de editare, cum ar fi normal, insert, selectare vizuală şi altele. Totodată acest editor are o comunitate activă care creează şi menţin o mulţime de extensii utile. Acest editor poate fi adaptat pentru o mulţime de cazuri de utilizare, iar configurarea acestuia este realizată prin editarea unui fişier, care poate fi copiat şi utilizat şi pe alte calculatoare. Pentru a facilita dezvoltarea unei aplicaţii Flutter, a fost instalată extensia flutter-tools.nvim, care oferă colorarea sintaxei şi alte funcţionalităţi.

Fiind că aplicaţia mobilă este destinată de a rula pe telefoane mobile care de cele mai multe ori au procesoare ARM însă majoritatea calculatoarelor sunt pe platforma x86, a fost nevoie de a instala un emulator android pentru a permite testarea mai rapidă a schimbărilor de cod. Acest emulator a fost obţinut instalând aplicaţia Android Studio.

Git este cel mai popular instrument pentru versionarea codului sursă [31]. Totodată el poate versiona şi alte lucruri, nu doar codul sursă, spre exemplu documente sau imagini care vor fi modificate. Totuşi este mult mai uşor de a lucra cu fişiere textuale decât cu fişiere binare în versionarea acestora, fiind posibilă compararea a două versiuni.

Github este o platformă online care permite găzduirea codului sursă, atât în repozitorii private cât şi în repozitorii publice. Totodată oferă posibilitatea de a crea pipeline-uri pentru integrarea continuă şi plasarea continuă, însă repozitoriile private au nişte limitări în acest domeniu.

Pentru realizarea sistemului au fost utilizate mai multe limbaje de programare, compilatoare şi librării. Limbajele de programare sunt C pentru utilizarea driverelor şi secvenţelor de cod existente pe internet. Zig [32] limbaj de programare şi compilator care permite compilarea atât a codului Zig cât şi C

în acelaşi proiect. Microzig [33] strat de abstractizare unificat pentru mai multe microcontrolere, inclusiv ESP32C3. Limbajul Dart împreună cu framework-ul Flutter pentru crearea aplicației mobile.

Pentru dezvoltarea proiectului a fost utilizat un laptop Lenovo Thinkpad E595 cu procesor AMD Ryzen 7 3700U cu 24 GB de RAM. Dintre perifericele utilizate în-afară de monitorul, tastatura şi touchpad-ul laptopului au mai fost folosite încă 2 monitoare externe. O tastatură ergonomică, despicată şi ortoliniară Dactyl Manuform [34] şi un trackball asemănător cu Logitech MX ERGO [35].

4.1 Aplicaţia mobilă

Prima pagină implementată a aplicației mobile a fost cea de pornire. Pe această pagină este prezentat titlul aplicației rotit cu 90 de grade. Această pagină va fi utilizată pentru a aștepta încărcarea datelor în aplicație. Implementarea vizuală a acestei pagini poate fi vizualizată în figura 4.1.

Figura 4.1 - Pagina de pornire

În figura 4.2 este prezentată o porțiune a paginii principale. Aceasta are o bară pe care în partea stânga se află pictograma setărilor iar în partea dreaptă se află pictograma alarmelor. Acest lucru va face atât setările cât şi alarmele accesibile. Dedesubtul acestei bare se află o listă de dispozitive. Acestea vor fi dispozitivele înregistrate în sistem. Dispozitivele active for fi prezentate cu un disc de culoarea verde iar cele inactive cu un disc de culoarea roşie.

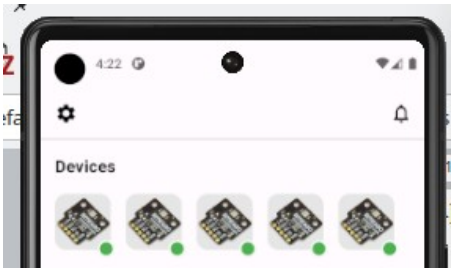

Figura 4.2 - Lista de dispozitive

Modul de a configura o alarmă este prezentat în figura 4.3. Utilizatorul poate introduce un titlu pentru această alarmă, care va fi afişat în caz că este declanşată alarma. În cadrul alarmei curente, utilizatorul va putea indica ce parametru doreşte să fie monitorizat. Pentru fiecare parametru poate fi specificat un interval, şi modul în care se doreşte evaluarea parametrului curent. Dacă aflarea acestuia în interiorul sau exteriorul parametrului va declanşa alarma.

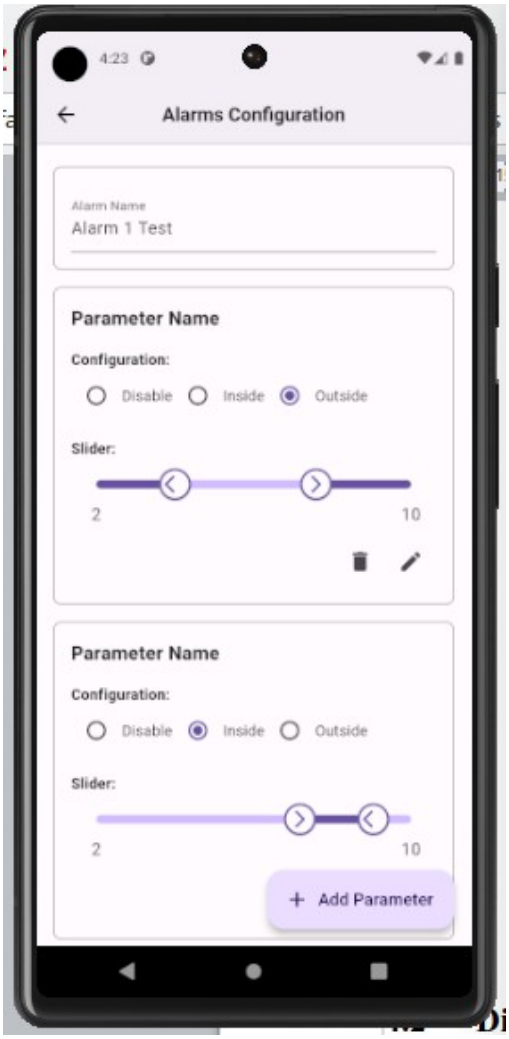

Figura 4.3 - Configurarea unei alarme

Pagina cu setări poate fi vizualizată în figura 4.4. Pe aceasta se regăseşte un link către pagina de Github a proiectului. Următorul element este un comutator care activează verificarea automată a noilor versiuni software pentru aplicaţie, următorul comutator verifică dacă există noi actualizări de firmware pentru dispozitiv. Următorul element din meniu va deschide o pagină pentru configurarea recurenţei

implicite a parametrilor însă utilizatorul va avea posibilitatea să decidă dacă dorește aceasta să fie utilizată pe fiecare dispozitiv în parte. Următorul element va permite de a adăuga conexiuni noi Wi-Fi în cazul când se va dori ca dispozitivul să aibă acces la internet. Apoi în ultimul element vor putea fi setate acreditarea pentru conectarea la un serviciu MQTT.

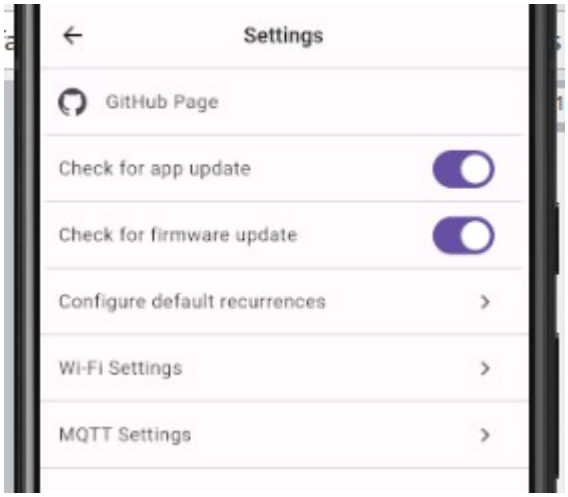

Figura 4.4 - Setările aplicației

Pagina de detalii, va mai trebui să fie actualizată pentru a arăta mai bine. Cel puțin la moment aceasta va putea afișa datele care sunt colectate. Implementarea aceste pagini poate fi vizualizată în figura 4.5.

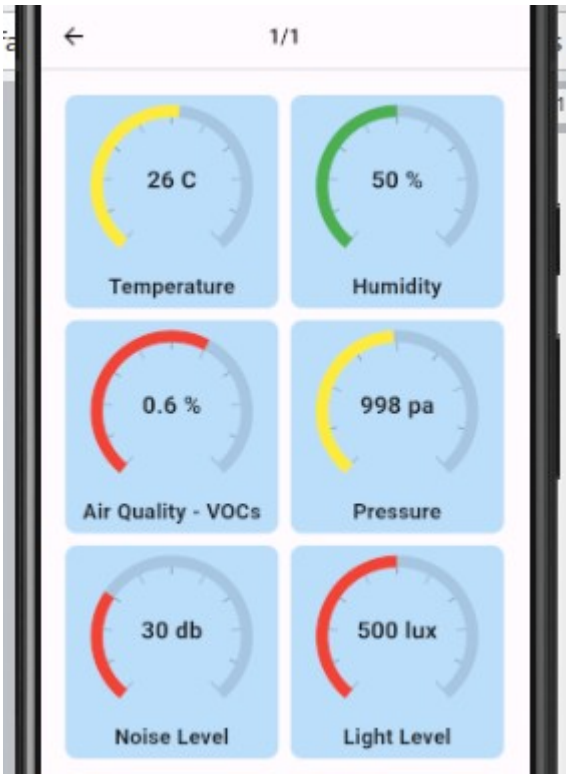

Figura 4.5 - Pagina de detalii

Un detaliu important despre realizarea aplicației este că a fost adăugată localizarea. La momentul aplicația suportă doar limba engleză, dar după necesitate fișierul app_en.arb poate fi tradus în alte limbi cum ar fi româna. Acest lucru este reprezentat în figura 4.6.

Figura 4.6 - Localizarea

În figura 4.7 este prezentat istoricul de mesaje din sistemul Git. La moment au fost făcute doar 23 de comiteri. Și aplicația nu se află într-o stare funcțională însă investind mai mult timp în aceasta, ea va putea fi funcțională.

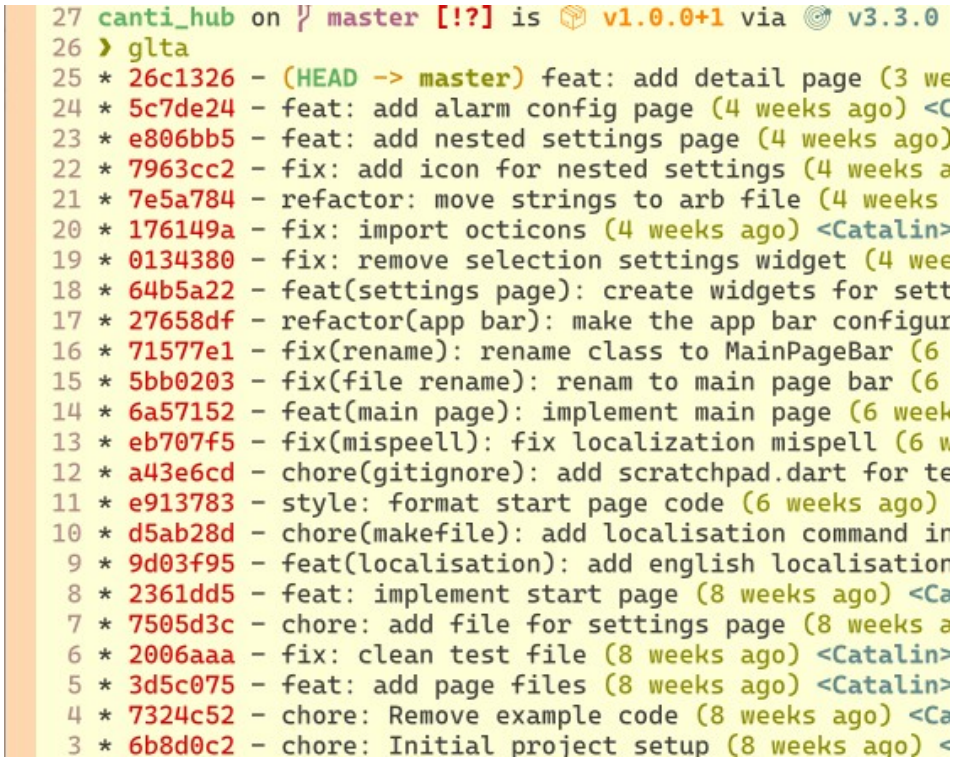

Figura 4.7 - Istoricul de mesaje

CONCLUZII

În cadrul acestei practici a fost efectuate o mare parte din eforturile necesare pentru finisarea proiectului. În principal eforturile ce țin de realizarea raportului. În primul rând a fost analizat domeniul de monitorizare al mediului din care face acest proiect. Și au fost comparate trei sisteme existente asemănătoare cu sistemul dezvoltat, Atmotube PRO, Garmin Tempe și DT-8820. A fost realizate specificațiile sistemului. Și a fost proiectat sistemul. De asemenea s-a început lucrul asupra realizării sistemului.

În urma lucrărilor efectuate, interfața aplicației mobile este aproape finisată, la ea mai trebuie doar de îmbunătățit pagina de detalii, de adăugat tema luminoasă și întunecată adaptivă și de implementat legătura între pagini. La moment obiectivele ce țin de aplicația mobilă sunt parțial îndeplinite find că aplicația nu este într-o stare funcțională.

A fost observat faptul că este mult mai uşor să lucrezi la un proiect care eşti plătit să îl realizezi decât la unul pe care îl faci din inițiativă proprie. Fără de a fi motivat de aspectul material. Altă posibilitate este că munca în echipă este mai benefică decât munca solo fiind că membri echipei te motivează şi îți aduc aminte că trebuie să îți faci sarcinile.

Această practică a avut o importanță dubioasă, în cadrul practicii a fost o pauză de două săptămâni care a făcut dificlă reînceperea lucrărilor în a doua perioadă. De asemenea faptul că se cere de a realiza un videoclip ca rezultat al acestei practici iarăşi, împiedică dedicarea eforturilor pentru finisarea proiectului.

BIBLIOGRAFIE

[1] D. E. Greenfield, „Environmental Monitoring: A Complete Guide”, Sigma Earth. Data accesării: 26 februarie 2024. [Online]. Disponibil la: https://sigmaearth.com/environmental-monitoring-a-complete-guide/

[2] „What are some of the current trends and challenges in environmental monitoring?” Data accesării: 23 noiembrie 2023. [Online]. Disponibil la: https://www.linkedin.com/advice/1/what-some-current-trends-challenges-environmental

[3] „All about Internet of Things (#IoT)”. Data accesării: 26 februarie 2024. [Online]. Disponibil la: https://www.linkedin.com/pulse/all-internet-things-iot-ahmed-karam

[4] „What Is Information Technology? A Beginner’s Guide to the World of IT”, Rasmussen University. Data accesării: 26 februarie 2024. [Online]. Disponibil la: https://www.rasmussen.edu/degrees/technology/blog/what-is-information-technology/

[5] R. Fuller *et al.*, „Pollution and health: a progress update”, *Lancet Planet. Health*, vol. 6, nr. 6, pp. e535–e547, iun. 2022, doi: 10.1016/S2542-5196(22)00090-0.

[6] „Atmotube PRO - Wearable and portable air quality monitor”. Data accesării: 23 noiembrie 2023. [Online]. Disponibil la: https://atmotube.com/atmotube-pro

[7] „tempe™ - Garmin Moldova”. Data accesării: 23 noiembrie 2023. [Online]. Disponibil la: https://garmin.md/produs/010-11092-30/

[8] „DT-8820 - 4 in 1 Multifunction Environment Meter”. Data accesării: 21 februarie 2024. [Online]. Disponibil la: https://www.cem-instruments.com/en/product-id-929

[9] „An Overview of the IoT Tech Stack | IoT Glossary”. Data accesării: 23 noiembrie 2023. [Online]. Disponibil la: https://www.emnify.com/iot-glossary/iot-stack

[10] Russell, „FPGA vs. Microcontroller vs. ASIC”, Nandland. Data accesării: 28 februarie 2024. [Online]. Disponibil la: https://nandland.com/lesson-2-fpga-vs-micro-vs-asic/

[11] M. Lab, „Bare-metal and RTOS Based Embedded Systems”, Microcontrollers Lab. Data accesării: 29 februarie 2024. [Online]. Disponibil la: https://microcontrollerslab.com/difference-bare-metal-and-rtos-based-embedded-systems/

[12] „Why I rewrote my Rust keyboard firmware in Zig: consistency, mastery, and fun”. Data accesării: 29 februarie 2024. [Online]. Disponibil la: https://kevinlynagh.com/rust-zig/

[13] „A 2022 Guide to IoT Protocols and Standards”, Particle. Data accesării: 29 februarie 2024. [Online]. Disponibil la: https://www.particle.io/iot-guides-and-resources/iot-protocols-and-standards/

[14] „HG353/2010 cu privire la aprobarea cerințelor minime de securitate și sănătate la locul de muncă”. Data accesării: 20 februarie 2024. [Online]. Disponibil la: https://www.legis.md/cautare/getResults?doc_id=22129&lang=ro

[15] „Regulamentul sanitar privind normativele de emitere a zgomotului și a vibrației”. Data accesării: 20 februarie 2024. [Online]. Disponibil la: https://gov.md/sites/default/files/document/attachments/intr07_1_18.pdf

[16] M. Gulin, „Answer to «Body Energy Consumption: calculating watts and kwh given calories?»”, Physics Stack Exchange. Data accesării: 27 februarie 2024. [Online]. Disponibil la: https://physics.stackexchange.com/a/702472

[17] „AADL resource pages”. Data accesării: 1 martie 2024. [Online]. Disponibil la: http://www.openaadl.org/

[18] „Unified Modeling Language, v2.5.1”, *Unified Model. Lang.*.

[19] „Modelio Open Source - UML and BPMN free modeling tool”. Data accesării: 1 martie 2024. [Online]. Disponibil la: https://www.modelio.org/index.htm

[20] „Ghidul simplu pentru diagrame UML și modelarea bazelor de date”. Data accesării: 1 martie 2024. [Online]. Disponibil la: https://www.microsoft.com/ro-ro/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling

[21] B. P. Douglass, *Design patterns for embedded systems in C: an embedded software engineering toolkit*, 1st ed. Oxford ; Burlington, MA: Newnes/Elsevier, 2011.

[22] „SOLID: The First 5 Principles of Object Oriented Design | DigitalOcean”. Data accesării: 3 martie 2024. [Online]. Disponibil la: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design
[23] „Linux and GNU - GNU Project - Free Software Foundation”. Data accesării: 3 martie 2024. [Online]. Disponibil la: https://www.gnu.org/gnu/linux-and-gnu.en.html
[24] „Home - Neovim”. Data accesării: 3 martie 2024. [Online]. Disponibil la: https://neovim.io/
[25] „welcome home : vim online”. Data accesării: 3 martie 2024. [Online]. Disponibil la: https://www.vim.org/
[26] Akin, „akinsho/flutter-tools.nvim”. 2 martie 2024. Data accesării: 3 martie 2024. [Online]. Disponibil la: https://github.com/akinsho/flutter-tools.nvim
[27] „Download Android Studio & App Tools”, Android Developers. Data accesării: 3 martie 2024. [Online]. Disponibil la: https://developer.android.com/studio
[28] „Git”. Data accesării: 3 martie 2024. [Online]. Disponibil la: https://git-scm.com/
[29] „Build software better, together”, GitHub. Data accesării: 3 martie 2024. [Online]. Disponibil la: https://github.com
[30] „Stack Overflow Developer Survey 2023”, Stack Overflow. Data accesării: 23 noiembrie 2023. [Online]. Disponibil la: https://survey.stackoverflow.co/2023/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2023
[31] „Stack Overflow Developer Survey 2022”, Stack Overflow. Data accesării: 3 martie 2024. [Online]. Disponibil la: https://survey.stackoverflow.co/2022/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2022
[32] „Home ⚡ Zig Programming Language”. Data accesării: 21 februarie 2024. [Online]. Disponibil la: https://ziglang.org/
[33] „ZigEmbeddedGroup/microzig”. Zig Embedded Group, 3 martie 2024. Data accesării: 3 martie 2024. [Online]. Disponibil la: https://github.com/ZigEmbeddedGroup/microzig
[34] J. Bertrand, „abstracthat/dactyl-manuform”. 2 martie 2024. Data accesării: 3 martie 2024. [Online]. Disponibil la: https://github.com/abstracthat/dactyl-manuform
[35] „Logitech MX ERGO Advanced Wireless Trackball with Tilt Plate”. Data accesării: 3 martie 2024. [Online]. Disponibil la: https://www.logitech.com/en-eu/products/mice/mx-ergo-wireless-trackball-mouse.html