UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DOMENIUL: Calculatoare și Tehnologia Informației

SPECIALIZAREA: Tehnologia Informației

**Localizarea unui dispozitiv portabil prin intermediul unei aplicații mobile folosind comunicarea prin aer**

**KeyFinder**

LUCRARE DE DIPLOMĂ

Coordonator științific

Ș.l.dr. Nicolae Botezatu

Absolvent

Bogdan-Iulian Iurea

Iași, 2021

Introducere

Capitolul 1. Fundamentarea teoretică și documentarea bibliografică pentru tema propusă

* 1. Tema propusă

Scopul temei propuse este acela de a oferi unei persoane posibilitatea să-și găsească obiectele ce pot fi pierdute prin atașarea unor dispozitive de localizare de acestea. Acest dispozitiv poate fi contactat folosind unde radio, astfel prin utilizarea unui alt dispozitiv ce are rolul de emițător se va putea trimite un semnal de activare al procesului de localizare al dispozitivului receptor. La momentul recepționării semnalului, dispozitivul va iniția rutina de localizare prin emiterea unor semnale sonore și luminoase ajutând la găsirea obiectului pierdut.

Pentru ca semnalul de activare să poată fi emis către dispozitivul ce trebuie localizat se va pune la dispoziție o aplicație mobilă ce va permite vizualizarea dispozitivelor receptor din apropiere. Dintre dispozitive se poate alege unul pentru a crea o asociere, asocierea fiind realizată prin verificarea unei chei unice de securitate cunoscută de dispozitivul receptor și care trebuie să fie introdusă de către utilizator. Dacă există dispozitive cu care s-au realizat asocieri și care să fie și în proximitate se oferă posibilitatea de a trimite semnalul de activare către unul dintre acestea.

Având în vedere cele enunțate în paragraful anterior, se deduce faptul că dispozitivul emițător este reprezentat de un telefon inteligent(eng.: smartphone). Dispozitivul receptor va avea la bază un microprocesor ce integrează module Bluetooth și Wi-F și va conține o diodă și un buzer(eng.: buzzer) prin care se vor emite semnalele de localizare. Concomitent, acesta va avea și un buton prin care se va afișa starea dispozitivului(dacă este conectat sau nu). Comunicarea dintre dispozitive va fi realizată prin intermediul standardului tehnologic Bluetooth cu energie redusă.

1.1.1. Obiective propuse

Luând la cunoștință scopul temei propuse enunțat la subcapitolul 1.1, obiectivul principal este acela de oferi posibilitatea găsirii obiectelor pierdute așa încât un utilizator să nu se simtă deranjat de dispozitivul atașat de obiect. Astfel, produsul final va trebui să fie de mici dimensiuni și să asigure un nivel de confort satisfăcător pentru utilizator.

La primirea semnalului de activare, dispozitivul va alterna fiecare culoare a diodei și va emite un semnal sonor. Totodată, la apăsarea butonului din componență, acesta va arăta că dispozitivul este pregătit de conectare prin emiterea unui semnal luminos de culoare verde sau va arăta că dispozitivul este deja conectat prin emiterea unui semnal luminos de culoare albastră.

Pentru aplicația mobilă, obiectivele propuse sunt următoarele:

* Aplicația trebuie împărțită într-o parte ce conține strict vizualizarea dispozitivelor din proximitate și modalitatea de asociere cu acestea și într-o parte ce conține doar vizualizarea dispozitivelor deja asociate și operațiunile ce pot fi executate de acestea;
* Căutarea dispozitivelor receptor să fie posibilă prin simpla apăsare a unui buton;
* Utilizatorul să fie anunțat despre operațiunile ce necesită timp pentru a-și termina execuția(de exemplu: căutarea dispozitivelor din jur) prin diverse metode(cutii dialog cu mesaje de atenționare, loader-e etc.);
* La finalul fiecărei căutări să se poată vizualiza dispozitivele receptor din proximitate prin dispunerea informațiilor acestora într-o listă ce va fi afișată pe ecran;
* Pentru a se realiza o asociere cu un dispozitiv, utilizatorul va trebui să introducă cheia unică de securitate a dispozitivului și apoi să aștepte rezultatul validării acesteia. Validarea cu succes va avea ca efect adăugarea dispozitivului în lista de dispozitive asociate, iar în cazul în care validarea a eșuat, utilizatorul va fi anunțat în legătură cu acest lucru;
* Dispozitivele asociate să fie afișate indiferent dacă acestea sunt în proximitate sau nu, dar totodată, să existe o modalitate de diferențiere între ele;
* Pentru un dispozitiv asociat să existe o modalitate de a elimina asocierea, iar pentru cel aflat și în proximitate să se ofere posibilitatea emiterii semnalului de activare al acestuia.

1.1.2. Justificarea abordării

Tema a fost abordată deoarece pachetul de cunoștințe necesar dezvoltării unui astfel de proiect cuprinde o mare parte din programa domeniului studiat. În acest mod, în proiect sunt folosite elemente învățate din discipline precum: electrotehnică, dispozitive electronice și electronică analogică, electronică digitală, sisteme cu microprocesoare, rețele de calculatoare, programare orientată pe obiecte, paradigme de programare, programare Web și programarea dispozitivelor mobile.

Aceasta combină elemente ce țin de dezvoltarea aplicațiilor mobile și dezvoltarea sistemelor cu microprocesor, existând posibilitatea ca produsele finale să poată comunica prin intermediul comunicării „prin aer”. În acest mod, tema cuprinde caracteristici specifice domeniului **IoT**, un domeniu de actualitate, fapt ce aduce un plus în vederea luării deciziei de a aborda un astfel de proiect.

Nu în ultimul rând, tema a fost aleasă și pentru utilitatea oferită a unui astfel de dispozitiv. Prin intermediul acestuia se poate economisi timp, astfel utilizatorul poate fi ușurat de nivelul de stres pe care l-ar dobândi în urma căutării unui obiect pierdut.

1.2. Domeniul și contextul abordării temei

Considerând obiectivele anterior enunțate și scopul temei propuse, se deduce că proiectul face parte din domeniul **IoT**(Internet of Things) pentru că se dorește ca un dispozitiv să fie controlat de la distanță prin intermediul unei aplicații mobile folosind comunicarea „prin aer”.

Internet of Things, tradus Internetul lucrurilor face referință la multitudinea de dispozitive ce pot comunica între ele, colectând și schimbând date. Prin aceste dispozitive se dorește ușurarea responsabilităților activităților de zi cu zi, scopul final fiind acela de a îmbunătăți calitatea vieții(QoL – Quality of Life). Din acest de domeniu fac parte aplicații/produse precum:

* Accesorii inteligente(ceasuri, brățări etc.) ce pot informa purtătorul despre nivelul de stres acumulat, temperatura pielii, locația actuală ș.a.m.d. În urma analizei informațiilor primite se pot lua decizii ce pot întreține calitatea vieții purtătorului(de exemplu, se pot lua decizii în privința continuării antrenamentului sportiv, necesității de a reduce nivelul de stres etc.);
* Dispozitive ce pot fi localizate prin GPS/Bluetooth(eng.: tracking device) prin intermediul unui alt dispozitiv mobil sau a unei aplicații Web;
* Aparatură casnică inteligentă(frigidere/mașini de spălat/aspiratoare inteligente) ce poate fi controlată din telefon/calculator.

Pentru a crea aplicații ce țin de domeniul IoT sunt necesare cunoștințe în domeniile dezvoltării de aplicații mobile/Web, dezvoltării de sisteme încorporate și comunicării în rețea.

1.2.1. Sisteme încorporate

Un sistem este un aranjament în care toate unitățile sale lucrează împreună în conformitate cu un set de reguli. De asemenea, acesta poate fi definit ca un mod de lucru/organizare sau o modalitate de a face una sau mai multe sarcini în conformitate cu un plan fix. Într-un sistem, toate subcomponentele depind una de cealaltă.

Un sistem încorporat poate fi considerat un sistem hardware cu software încorporat în acesta. Poate fi un sistem independent sau poate face parte dintr-un sistem mai mare. Un sistem încorporat este un microcontroler sau un sistem bazat pe microprocesor care este proiectat pentru a efectua o anumită sarcină. Un sistem încorporat are trei componente(hardware, software și R.T.O.S. – Real Time Operating System). Un sistem de operare în timp real supraveghează aplicația ce este executată și oferă un mecanism de control al proceselor, definind modul în care funcționează sistemul.

Așadar, un sistem încorporat are la bază un microcontroler/microprocesor ce poate fi programat și care poate utiliza(dacă este cazul) un sistem de control în timp real.

Pentru a programa un microcontroler sunt necesare cunoștințe ale limbajelor de programare „low level” deoarece memoria pusă la dispoziție este limitată, iar limbajele de nivel înalt nu sunt eficiente din punct de vedere al utilizării memoriei.

1.2.2. Tehnologia Bluetooth

Bluetooth-ul este un standard al tehnologiei fără fir cu rază de acțiune scurtă care elimină necesitatea de a folosi cabluri pentru a comunica cu mai multe dispozitive electronice. Funcționalitatea acestuia este foarte asemănătoare cu cele ale rețelei mobile sau ale tehnologiei Wi-Fi, dar diferă prin faptul că el este destinat comunicării pe rază scurtă având sarcini relativ simple, în timp ce tehnologiile anterior menționate sunt destinate conectării în masă a mai multor dispozitive într-o rețea largă sau chiar la internet.

În același timp, Bluetooth-ul cu energie redusă(eng.: Bluetooth Low Energy) este tot un standard tehnologic de comunicare „prin aer” cu origini moștenite de la Bluetooth-ul „clasic”, dar care aduce îmbunătățiri asupra consumului de energie cu costul reducerii cantității de date trimise. Dispozitivele care adoptă standardul cu energie redusă pot intra într-un mod inactiv(eng.: sleep) până la sosirea unui nou eveniment de conectare, reducând semnificativ cantitatea de energie consumată.

Bluetooth-ul, din punct de vedere al implementării hardware este compus din două părți, una analogică radio și una digitală. Partea digitală este numită Host Controller(H.C.) și conține interfețele cu mediul gazdă, un procesor și modulul de procesare al semnalului digital(Link Controller). Pe nucleu sunt executate instrucțiuni care permit descoperirea și comunicarea cu alte dispozitive prin intermediul protocolului de gestiune al legăturilor(eng.: Link Manager Protocol - L.M.P.). La nivelul software, pentru a asigura compatibilitatea între dispozitive cu implementări hardware diferite se utilizează o interfață comună între dispozitivul gazdă și nucleul Bluetooth, astfel protocoalele de nivel superior sunt mascate de serviciile din banda de bază cu ajutorul protocolului de adaptare și control al legăturilor logice(eng.: Logic Link Control and Adaptation Protocol – L.2.C.A.P.).

Standardul încorporează mai multe implementări ale protocoalelor de comunicare, astfel sunt definite protocoale ce stau la baza acestuia(L.M.P. – stabilește și controlează legătura dintre dispozitivele Bluetooth, L.2.C.A.P. – maschează serviciile din banda de bază, S.D.P. – tabelează serviciile expuse ale altor dispozitive), protocoale de înlocuire a cablurilor(R.F.C.O.M.M. – asigură existența unei legături între două dispozitive folosind frecvența radio), protocoale de control telefonic(T.C.S.B.I.N., H.T.T.P., F.T.P.) și protocoale adoptate(P.P.P., T.C.P./I.P., U.D.P.).

Oferă posibilitatea de a conecta mai multe dispozitive electronice fără a exista un intermediar datorită reglementărilor pe care le aduce. Tehnologia aduce propriile implementări asupra menținerii relației de conectivitate dintre dispozitive, oferind posibilitatea alegerii protocolului de comunicare între emițător și receptor. Prin urmare, sunt definite seturi de profiluri Bluetooth, adesea numite servicii sau funcții care expun încapsulări ale funcționalității unui anumit dispozitiv.

Un profil Bluetooth este o specificație prin care se definesc aspecte asupra menținerii unei conexiuni fără fir între dispozitive. Specificația profilului trebuie să conțină un minim de informații asupra utilizării stivei de protocoale Bluetooth cu scopul de a asigura interoperabilitatea între dispozitive. De-a lungul timpului au fost create și standardizate mai multe profiluri Bluetooth, dintre care cele cu numărul cel mai mare de aplicații în momentul actual sunt:

* A2DP – Advanced Audio Distribution Profile: are ca scop de trimiterea în flux a fișierelor audio;
* FTP – File Transfer Profile: se ocupă de transferul de fișiere;
* BPP – Basic Printing Profile: are ca rol imprimarea documentelor;
* HFP – Hands-Free Profile;
* GAP – Generic Access Profile: controlul accesului;
* GATT – Generic Attribute Profile: transferul cantităților mici de date eficient din punct de vedere al consumului de energie(Bluetooth Low Energy).

1.2.3. Aplicații mobile

O aplicație mobilă este un program de calculator sau o aplicație software concepută pentru a rula pe un dispozitiv mobil, cum ar fi un telefon, o tabletă sau un ceas. Aplicațiile au fost inițial destinate pentru a spori productivitatea, dar cererea pentru aplicații a cauzat extinderea rapidă în alte domenii(cum ar fi jocuri mobile, automatizări etc), așa încât acum există milioane de aplicații disponibile.

În momentul actual cele mai folosite sisteme de operare destinate dispozitivelor mobile sunt Android(dezvoltat de Google) și iOS(dezvoltat de Apple). O aplicație mobilă nu poate funcționa pe ambele sisteme de operare deoarece fișierele binare rezultate în urma compilării unui proiect Android nu sunt compatibile cu sistemul iOS și nici invers. Programul rezultat este numit aplicație nativă deoarece este destinat doar platformei pentru care a fost creat.

Pentru a crea o aplicație pentru Android sunt necesare mediul de dezvoltare integrat(I.D.E. – Integrated Development Environment) Android Studio, pachetul de dezvoltare Android pentru limbajele Java/Kotlin sau pachetul de dezvoltare nativ pentru limbajele C/C++. Simultan, pentru a crea o aplicație iOS sunt necesare mediul de dezvoltare integrat Xcode și pachetul de dezvoltare iOS. Aplicațiile native iOS pot fi scrise folosind limbajele Swift sau Objective C.

Datorită incompatibilității aplicațiilor între platforme au fost create noi medii de dezvoltare prin care să fie posibilă scrierea aplicațiilor mobile prin utilizarea unui singur cod sursă, programele rezultate fiind numite aplicații multi-platformă(eng.: Cross-Platform). Dintre aceste medii de dezvoltare, cele mai utilizate sunt:

* React Native – Mediu de dezvoltare creat peste librăriile native ce face legătura între codul sursă scris și librăriile native ale fiecărei platforme, astfel oferind o experiență și o performanță apropiată de aplicațiile native;
* Flutter – Mediu de dezvoltare ce are ca scop evitarea folosirii librăriilor native prin utilizarea propriei mașini virtuale prin care sunt gestionate evenimentele sistemului(gesturi, animații etc.) cât și desenarea elementelor din interfață;
* Apache Cordova – Mediu de dezvoltare ce permite dezvoltarea aplicațiilor Web hibride pentru aplicații mobile folosind tehnologii Web. Majoritatea mediilor de dezvoltare destinate creării de aplicații mobile prin intermediul tehnologiilor Web au la bază această tehnologie(de exemplu: Ionic Framework, Quasar Framework etc.);
* Ionic Framework – Oferă instrumente si servicii pentru dezvoltarea de aplicații mobile, aplicații desktop și aplicații Web progresive bazate pe tehnologii si practici moderne de dezvoltare Web folosindu-se de tehnologii Web. Ca rezultat, aplicațiile nu sunt native deoarece structura acestora este afișată prin vizualizări Web, dar nu sunt nici aplicații Web deoarece acestea pot fi împachetate și au acces și la librăriile native. Acest tip de aplicații sunt numite aplicații multi-platformă hibride.

1.3. Realizările actuale pe aceeași temă. Analiza comparativă a tipurilor de produse/aplicații existente din categoria temei

De-a lungul timpului au apărut în comerț diferite abordări în ceea ce prevede tema propusă, dezvoltatorii venind cu diferite soluții pentru a rezolva problema discutată. Astfel, dintre aplicațiile existente, enumerăm: „Orbit”, „chipolo”, „tile”, „MYNT”, „Samsung Galaxy SmartTag”, „Apple AirTag”, „eSky Wireless Key Tracker” și altele.

Aplicațiile anterior enumerate au același scop, de a localiza un obiect pierdut prin atașarea de acesta a unui dispozitiv ce poate fi localizat, fiecare putând avea funcționalități suplimentare. Astfel, aplicațiile pot diferi în ceea ce presupune cantitatea și calitatea funcționalităților oferite, cât și prin modalitatea de a găsi un dispozitiv(folosirea unei telecomenzi, a unei aplicații mobile ce poate afișa distanța față de dispozitiv/locația acestuia, etc.)

Având în vedere produsele existente din categoria temei, proiectul dezvoltat are funcționalități și caracteristici asemănătoare celor aflate pe piață, asemănările dintre acestea fiind nivelul de portabilitate al dispozitivului receptor, modalitatea de contactare a acestuia și funcția de localizare. Elementele diferite sunt reprezentate de opțiunile suplimentare pe care celelalte produse le oferă, precum localizare GPS, localizare GPS prin realitate augmentată, localizare bidirecțională etc.

Ceea ce este diferit în abordarea temei este modalitatea de a crea o asociere între dispozitivul receptor și dispozitivul mobil. Pentru ca dispozitivul să poată fi localizat este necesară autorizarea prin validarea unei chei de securitate. Cheia este cunoscută de dispozitivul receptor și de deținătorul dispozitivului, astfel accesul este interzis pentru cei ce nu cunosc cheia.

1.3.1. tile

Este o companie ce oferă dispozitive localizabile prin tehnologia Bluetooth Low Energy, permițând deținătorului localizarea unuia prin utilizarea unei aplicații mobile. Dimensiunile dispozitivului sunt relativ mici, acesta venind în diferite forme, precum breloc, sticker sau card.

La nivel de funcționalitate, dispozitivul poate fi localizat prin semnale sonore, locație GPS, dar și prin analizarea distanței față de acesta. Accesul asupra funcționalităților este pus la dispoziție printr-o aplicație mobilă sau prin intermediul asistenților virtuali(Alexa, Google etc.).

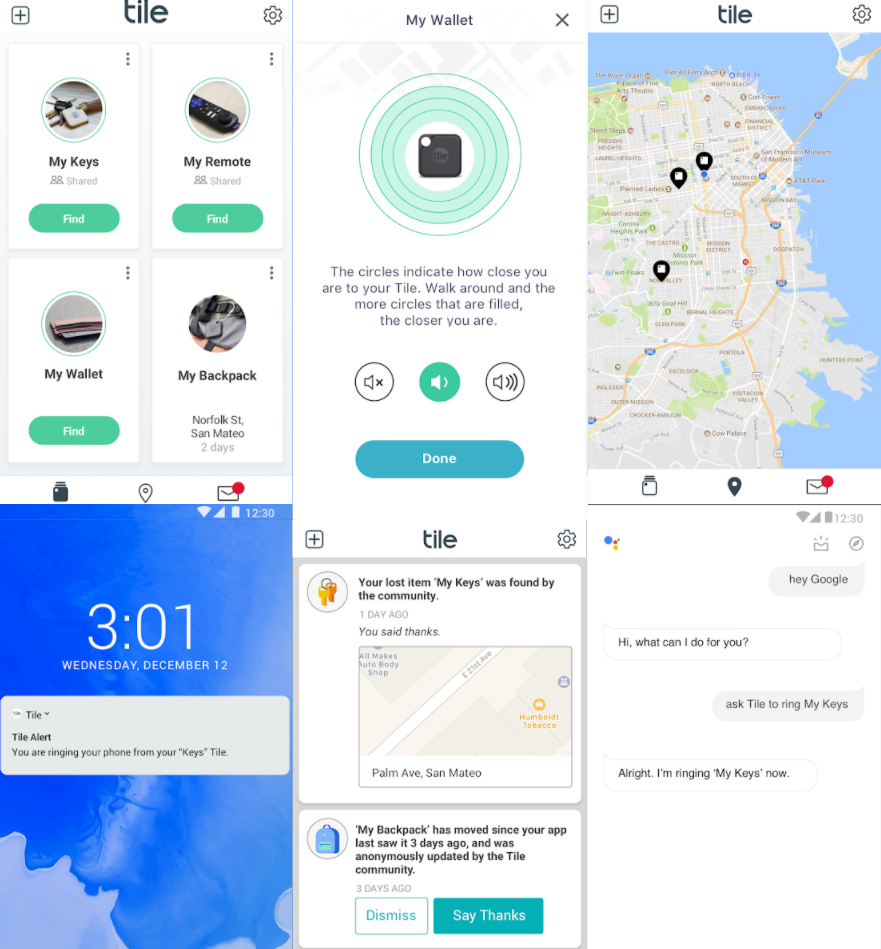


Figura 1.1. Aplicația „tile”



Figura 1.2. Dispozitivul localizabil „tile”

1.3.2. Samung Galaxy SmartTag

Acesta oferă aceleași funcționalități precum cele ale dispozitivului descris anterior, ceea ce îl diferențiază fiind integrarea în ecosistemul SmartThings dezvoltat de Samsung. SmartThings este o aplicație ce permite controlul și gestiunea tuturor obiectelor „inteligente” ce sunt conectate la rețeaua locală, dar și a celor ce sunt conectate prin Bluetooth Low Energy.

Dezvoltatorii oferă posibilitatea de a vizualiza locația dispozitivului și prin vizualizarea poziției acestuia folosind realitatea augmentată. Simultan, o funcționalitate ce este greu de atins este aceea că dispozitivul poate fi localizat și prin comunicarea cu orice alte obiecte inteligente ce fac parte din ecosistem, fapt strict dependent de numărul de produse aflate în uz.

În paralel cu scopul principal, se oferă și posibilitatea de a modifica funcționalitatea butonului dispozitivului. Printre acțiunile pe care le poate modifica un utilizator se numără: controlul altor dispozitive inteligente(cum ar fi pornirea/oprirea unor becuri inteligente, televizoare smart etc.), trimiterea de notificări sau mesaje prestabilite către o persoană, căutarea dispozitivului mobil asociat ș.a.m.d.

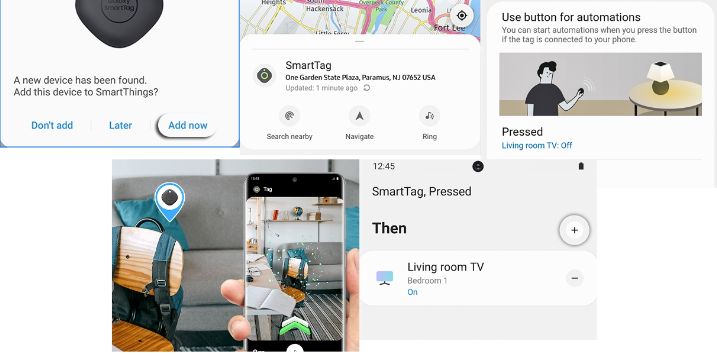


Figura 1.3. Funcționalități ale aplicației SmartThings



Figura 1.4. Dispozitivul localizabil Samsung Galaxy SmartTag

1.5. Specificații privind caracteristicile așteptate de la aplicație

Considerând obiectivele propuse, semnalul sonor al dispozitivului trebuie să poată fi diferențiat foarte ușor de sunetele prezente în mediul înconjurător, astfel, se așteaptă ca acesta să poată fi recunoscut încă de la începutul rutinei de localizare. Simultan, semnalele luminoase trebuie să fie de intensitate maximă așa încât dispozitivul să poată fi localizat și în condiții de iluminare nefavorabile.

Perioada de scanare a dispozitivelor din jur nu trebuie să dureze o perioadă îndelungată de timp, deoarece aceasta ar putea crea o stare de confuzie pentru utilizator. Scanarea nu trebuie să dureze mai mult de 3 sau 5 secunde.

Funcționalitățile trebuie oferite printr-o cale cât mai simplă, așa încât utilizatorul să nu se simtă deranjat de gradul de accesibilitate al acestora. Acțiunile pe care le poate face trebuie scoase cât mai în evidență(butoanele/câmpurile text să fie clare).

Dacă un utilizator este prea depărtat de un dispozitiv, iar dacă aplicația oferă posibilitatea de a-l contacta, comunicarea dintre acestea fiind imposibilă datorită distanței mult prea mari dintre acestea, utilizatorul va trebui anunțat de acest lucru. Distanța de la care un dispozitiv poate fi contactat trebuie să fie asemenea caracteristicilor standardului Bluetooth Low Energy, iar un dispozitiv trebuie să execute rutina de localizare pentru cel puțin 4 secunde.

Aplicația mobilă va trebui să fie ușor de folosit și să nu creeze o stare de confuzie pentru utilizator. Funcționalitățile acesteia trebuie să fie ușor de remarcat așa încât utilizatorul să le poată folosi și fără citirea unui manual de instrucțiuni.

Capitolul 2. Proiectarea aplicației

2.1. Analiza platformelor hardware pe care vor fi executate aplicațiile

O platformă hardware este compusă din seturi de elemente hardware compatibile ce permit executarea programelor software. Fiecare astfel de platformă are definit propriul limbaj(limbajul mașină), iar programele ce rulează pe o astfel de platformă sunt special construite pentru fiecare categorie de procesor. Aceasta definește standardul în jurul căruia poate fi dezvoltat un sistem, este o bază de tehnologii pe care sunt construite alte tehnologii sau procese.

Pentru a îndeplini obiectivele propuse sunt necesare două dispozitive ce pot comunica, fiecare având un rol esențial în proces, unul va trimite instrucțiuni/comenzi, iar altul le va intercepta și interpreta.

Au fost analizate toate opțiunile în vederea selectării sistemelor ce vor putea comunica, iar ca rezultat, au fost alese ca inițiator/emițător mulțimea dispozitivelor mobile ce folosesc ca sistem de operare platforma **Android**, alegere motivată de excelența portabilității unui program pentru o astfel de platformă și de infrastructura de dezvoltare a aplicațiilor destinate acestuia. În același timp, componenta pe post de receptor este reprezentată de microcontrolerul **ESP32**, alegere motivată de independența cipului de periferice, având integrate module de Bluetooth și Wi-Fi, dar și de performanța pe care o are.

**ESP32** este o serie de sisteme „low-cost” cu consum redus de energie, având cip integrat Wi-fi și Bluetooth cu mod dual. Seria ESP32 utilizează un microprocesor Tensilica Xtensa LX6 atât în variante dual-core, cât și în variante single-core și include comutatoare de antenă încorporate, „balun R.F.”, amplificator de putere, amplificator de recepție cu zgomot redus, filtre, și module de gestionare a alimentării.

Dispozitivele mobile pe care este rulează platforma Android pot avea la bază tipuri diferite de procesoare cu arhitectură **A**.**R**.**M**.(Qualcomm Snapdragon, Samsung Exynos, HiSilicon Kirin etc) cu un număr de nuclee cuprins între unul și opt. Memoria RAM este în general cuprinsă între 0.5-8 GO. Acestea încorporează o gamă variată de module hardware(camere foto, modul Bluetooth, modul Wi-Fi etc.).

Utilizarea tehnologiei Bluetooth în schimbul tehnologiilor cu funcționalități asemănătoare este motivată de existența unei comunicări între dispozitive fără a utiliza intermediari care să poată manipula pachetele de date trimise, astfel este asigurată securitatea la nivel de transfer al datelor. Simultan, înlocuirea Bluetooth-ului „clasic” cu Bluetooth-ul Low Energy a fost motivată de consumul scăzut de energie al tehnologiei succesoare.

2.2. Modulele proiectului

Un modul este o parte separabilă logic a unui program. În acest mod, proiectul este format din două programe software și o componentă hardware. Unul din programele software este destinat dispozitivelor mobile, iar cel de-al doilea este destinat microcontrolerului ESP32. Componenta hardware este reprezentată de microcontroler și ansamblul elementelor de circuit conectate la pinii de intrare/ieșire ai acestuia.

Prin aplicația mobilă se dorește accesarea componentei Bluetooth a dispozitivului, așa încât prin intermediul interfeței cu utilizatorul să se poată realiza descoperirea, asocierea și trimiterea comenzilor către dispozitivele-receptor din proximitate. Aceasta va împărțită în 3 module, primul(„components”) ce conține elementele pur vizuale ale aplicației, al doilea(„logic”) ce cuprinde implementările funcționalităților programului, iar al treilea(„screens”) realizează cuplarea modulelor.

Programul pentru microcontroler va aștepta cererile de asociere, va interpreta mesajele primite de la dispozitivele asociate, va lua măsuri în funcție de natura mesajului primit și va gestiona evenimentele de întrerupere apărute în perioada de execuție. Așadar, acesta va cuprinde logica de inițializare și configurare a componentei Bluetooth, dar și logica de configurare a pinilor de intrare/ieșire ai acestuia.

În ceea ce privește modulul hardware, a fost luată în considerare siguranța implementării circuitului electric din punct de vedere electrotehnic. Prin acesta se permite crearea evenimentelor de întrerupere ale programului microcontrolerului(prin apăsarea unui buton) cu scopul de a afișa starea curentă a componentei Bluetooth(dacă există asocieri sau nu).

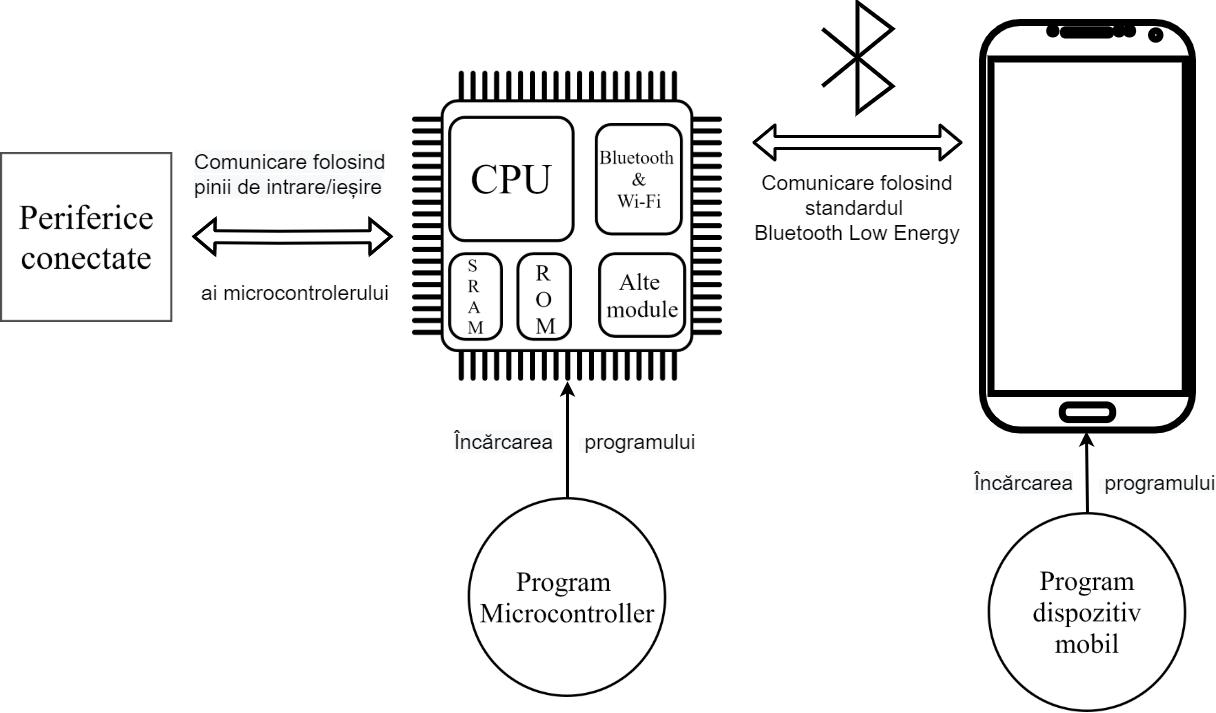


Figura 2.1. Interacțiunea dintre dispozitive

2.3. Avantaje și dezavantaje ale metodei alese

2.3.1. Avantaje și dezavantaje ale aplicațiilor pentru Android

Principalul avantaj al dezvoltării unui program destinat sistemului de operare Android este acela că numărul de dispozitive mobile care folosesc această platformă software este majoritară în raport cu restul platformelor existente pe piață. De-a lungul perioadei de studiu a platformelor dispozitivelor mobile luate în considerare au fost evidențiate următoarele avantaje ale aplicațiilor Android:

* Sistemul de operare are codul sursă accesibil(eng.: open source), lucru benefic pentru crearea aplicațiilor mobile;
* Kit-ul nativ de dezvoltare al software-ului și mediul de dezvoltare sunt bine documentate;
* Existența multiplelor medii terțe ce au ca scop dezvoltarea aplicațiilor native scriind un singur program sau a celor care au ca obiect dezvoltarea unui program ce poate rula pe orice platformă(eng.: cross-platform);
* Infrastructura de dezvoltare nu este costisitoare;
* Rapiditate la nivel de implementare;
* Adaptarea rapidă la mediul de dezvoltare folosit;
* Publicarea aplicației este ușor și rapid de gestionat.

Simultan, cel mai conturat dezavantaj este acela că posibilitatea fragmentării aplicației este mult mai mare datorită numărului larg de dispozitive ce diferă în rezoluție și diagonală. Pentru a asigura compatibilitatea cu toate dispozitivele, perioada de testare a aplicației este lungită considerabil, astfel publicarea aplicației putând fi amânată.

2.3.2. Avantaje și dezavantaje ale microcontrolerului ESP32

Primul avantaj al folosirii acestui microcontroler este acela că cipul integrează un set de module ce permit dezvoltatorilor să creeze aplicații scalabile și adaptabile în timp. Simultan cu această idee, microcontrolerul dispune de mai multe avantaje, precum:

* Module Wi-Fi și Bluetooth integrate;
* Microprocesor cu două nuclee pe 32 de biți ce funcționează la frecvențe între 160 și 240 MHz;
* Pornire securizată ce are ca scop menținerea integrității hardware;
* Pornire prin întreruperi produse la nivelul pinilor de intrare/ieșire;
* 34 de pini de intrare/ieșire programabili;
* Generator de semnale P.W.M.;
* Memorie flash criptată;
* Consum de energie redus.

Dezavantajul scos în evidență este acela că prețul de achiziție al cipului este aproape dublu comparativ cu predecesorul său, microcontrolerul ESP8266, fapt motivat de îmbunătățirile și noile funcționalități aduse.

2.3.3. Avantaje și dezavantaje ale tehnologiei Bluetooth Low Energy

Pe durata studiului a fost conturat evenimentul adoptării standardului tehnologic de cât mai multe companii de profil, lucru datorat avantajelor aduse, dintre care enumerăm:

* Facilitează consumul redus de energie;
* Nu apar interferențe cu alte dispozitive fără fir;
* Conectarea mai multor dispozitive la un singur dispozitiv este posibilă;
* Securitatea transferului de date este asigurată de faptul că nu există intermediari care să transmită datele de la un capăt la altul;
* Compatibilitate asigurată între dispozitivele ce folosesc profilurile existente.

Simultan, au fost identificate și dezavantaje ale acestei tehnologii, dintre care enunțăm:

* Conexiunea dintre dispozitive se face doar pe distanțe scurte, ceea ce duce la pierderea acesteia dacă dispozitivele sunt prea depărtate;
* Lățimea de bandă este mică;
* Dispozitivele rămase pornite ce nu au un cod de securitate pentru asociere pot fi foarte ușor penetrate de atacuri cibernetice, astfel datele stocate pot fi compromise.

2.4. Limite în care metoda aleasă va funcționa

Având în vedere analiza făcută și avantajele și dezavantajele anterior menționate se înțelege că aplicația mobilă este destinată dispozitivelor Android, fapt ce creează limitări din punct de vedere al platformei pe care poate fi lansat în execuție programul. Concomitent, existența unei legături între mecanisme este limitată de distanța dintre acestea, definitivând neputința asocierii dispozitivelor dacă ele nu sunt în proximitate.

Este luată în considerare și capabilitatea tehnologiei Bluetooth de a putea oferi suportul ca un dispozitiv să poată avea mai multe conexiuni. Aceasta nu va li luată în calcul deoarece este neutră din punct de vedere al obiectivelor proiectului, fapt ce duce la o limitare de funcționare.

O altă limitare este aceea că nu toate dispozitivele mobile au module ce încorporează și tehnologia Bluetooth Low Energy, astfel aplicația mobilă fiind destinată doar dispozitivelor ce satisfac acest criteriu.

2.5. Componentele proiectului

O componentă reprezintă o parte a unui întreg. Componentele software sunt părți ale unei aplicații sau ale unui sistem, fiecare având un scop unic, astfel complexitatea unei probleme este împărțită în fragmente ușor de gestionat. O componentă software poate fi implementată independent și este supusă compoziției de către terți. O componentă hardware este o unitate fizică autonomă ce poate fi încorporată într-un sistem complex(de exemplu un microcontroler și modulele încorporate de acesta).

2.5.1. Componentele software

2.5.1.1. Aplicația mobilă

Pentru schițarea aplicației mobile au fost luate în considerare obiectivele principale ale temei propuse, astfel dezvoltarea a pornit cu ideea că prin interfața cu utilizatorul se vor permite:

* descoperirea dispozitivelor din jur;
* asocierea cu unul sau mai multe din dispozitivele descoperite prin autorizare folosind o cheie de securitate;
* contactarea dispozitivelor asociate;
* ștergerea unei asocieri existente;

Astfel s-a creat diagrama de flux(eng.: flow diagram) a ecranelor aplicației, diagramă ilustrată în Anexa 1.1.

2.5.1.1.1. Tehnologia aleasă pentru implementare

Platforma de lucru aleasă pentru program este **React Native** deoarece aceasta pune la dispoziție uneltele necesare dezvoltării de aplicații pentru orice platformă prin crearea unui singur proiect. La etapa de compilare trebuie specificată platforma pentru care se creează respectiva aplicație, mediul apoi făcând legăturile dintre librăriile folosite/implementări și librăriile native ale fiecărei platforme(Android, iOS, Windows, ș.a.m.d.).

Avantaje ale framework-ului React Native:

* Dezvoltarea unei aplicații React Native este asemănătoare cu dezvoltarea aplicațiilor Web deoarece acesta are ca bază framework-ul React;
* Flexibilitate în ceea ce presupune managementul codului;
* Rapiditatea conversiei proiectului multi-platformă către un proiect nativ;
* Schimbările aduse codului sunt reflectate în pre-vizualizarea aplicației;
* Sprijin deplin din partea dezvoltatorului și a comunității;
* Eficient din punct de vedere al costului și timpului;
* Folosește un limbaj multi-paradigmă;
* Extensibilitatea aplicației pe mai multe platforme.

Totodată, mediul permite împărțirea elementelor vizuale ale aplicației(eng.: view) în componente independente ce pot fi refolosite. O componentă este o clasă sau o funcție **JavaScript** ce are ca scop definirea comportamentului și a aspectului unui obiect de pe interfață. Pentru ca o componentă să fie definită corect, aceasta trebuie să conțină o modalitate prin care să se returneze un obiect **J.S.X.**(**JavaScript** **X.M.L.**). JavaScript X.M.L. este o extensie adusă limbajului JavaScript prin care se poate descrie aspectul interfeței utilizator, în același timp oferind și posibilitatea de a lega variabilele de elementele vizuale(eng.: data binding).

2.5.1.1.2. Modulul pentru funcționalități

Clasa **BtManager** are în vedere utilizarea componentei Bluetooth a dispozitivului mobil. Aceasta conține metode prin care se oferă posibilitățile de a scana dispozitivele din jur, de a crea o asociere cu acestea și de a trimite pachetul de date ce conține mesajul de activare al funcției de localizare a dispozitivului ce urmează să fie contactat.

În câmpul **data** al clasei **BtManager** se încarcă informațiile din fișierul „devices-data.json”, fișier ce conține id-urile serviciilor Bluetooth de interes și câmpuri ce descriu tipul operațiunilor executate de dispozitivele țintă. Câmpul **manger** reprezintă instanța clasei **BleManger** ce aparține pachetului **react-native-ble-plx**. Pachetul oferă o interfață pentru librăriile native ale componentei Bluetooth a oricărui tip de dispozitiv mobil.

Metoda **searchForDevices()**  a clasei **BtManager** returnează un **promise** și are rolul de a descoperi dispozitivele Bluetooth Low Energy din apropiere al căror identificator de serviciu de advertising se găsește în lista de id-uri încărcată la momentul creării instanței **BtManager**. Scanarea se face timp de trei secunde urmând ca apoi să se returneze o listă ce conține detaliile dispozitivelor din proximitate, existând posibilitatea ca aceasta să poată fi și goală în cazul în care nu sunt dispozitive în jur;

Metoda **addDevice()** are ca scop trimiterea cheii de securitate către dispozitivul receptor. Aceasta primește doi parametri, cheia ce trebuie validată și id-ul dispozitivului pentru care se va face asocierea. Va realiza conexiunea între dispozitive și va descoperi serviciile și caracteristicile Bluetooth ale dispozitivului receptor. Apoi va trimite cheia de securitate către dispozitiv succedată de citirea și returnarea rezultatului obținut în urma validării cheii. În cazul în care cheia de securitate este validă funcția va returna codul de acces al dispozitivului sau codul de eroare în cazul în care validarea a eșuat.

Metoda **findDevice()** are în vedere emiterea semnalului de activare al funcției de localizare a dispozitivului receptor, astfel se trimite către dispozitiv codul de acces alături de mesajul specific operațiunii de găsire. Funcția primește doi parametri, id-ul dispozitivului ce trebuie localizat și codul lui de acces obținut la asociere.

Metodele **findDevice()** și **addDevice()** vor arunca erori în cazul în care operația cerută nu poate fi executată(din motive precum: Bluetooth-ul/serviciile de localizare nu sunt pornite, dispozitivul nu este în apropiere sau accesul la serviciile de localizare nu este permis). Pentru a verifica ce fel de eroare trebuie aruncată, a fost creată metoda **throwErrorByType()**,necesitatea acesteia fiind motivată de faptul că managerul Bluetooth din librăria **react-native-ble-plx** aruncă un singur tip de eroare(**BtError**), astfel identificarea acesteia trebuie făcută prin evaluarea mesajului erorii, ci nu prin tipul instanței.

Clasa **StorageManager** are rolul de a accesa spațiul de stocare al dispozitivului mobil, iar prin aceasta se dorește memorarea, preluarea și ștergerea informațiilor dispozitivelor asociate. Informațiile unui dispozitiv sunt stocate/preluate prin intermediul unui model de date(**StoreDeviceDataModel**) ce cuprinde detaliile acestuia. Prin model se dorește crearea unei structuri așa încât informațiile ce urmează a fi memorate/preluate să fie ușor de manipulat.

Pentru a scrie în spațiul de stocare al unui dispozitiv sunt folosite obiecte de tipul cheie-valoare(dicționare), existând o constrângere asupra tipului valorii. Valoarea memorată trebuie să fie un șir de caractere, astfel fiind necesare serializarea și deserializarea modelului.

Metodele clasei **StoreManger**:

* **store()** – are în vedere scrierea în spațiul de stocare a modelului de date al unui dispozitiv, serializându-l și trimițându-l ca valoare. Modelul și cheia la care poate fi găsit sunt primite ca parametri;
* **retrieve() –** returnează informațiile din spațiul de stocare ce aparțin unui dispozitiv asociat, fără a le deserializa prin accesarea valorii cheii primite ca parametru;
* **remove() –** elimină ceea ce se găsește în spațiul de stocare pentru cheia primită ca parametru;
* **getAllDevices() –** returnează informațiile tuturor dispozitivelor asociate sub forma modelului de date **StoreDeviceDataModel**. Acest lucru este făcut prin preluarea informațiilor stocate ale fiecărui dispozitiv, deserializarea și maparea către model. Fiecare model va fi adăugat într-o listă ce urmează să fie returnată la finalul procesării;
* **cleanup() –** elimină toate datele stocate de aplicație.

Atât clasa **BtManager**, cât și clasa **StoreManager** vor avea o singură instanță pe tot parcursul programului, acestea vor trebui implementate conform șablonului **Singleton**. Acest fapt este motivat de dorința de a exista instanțe unice în program care să poată accesa componenta Bluetooth a dispozitivului, respectiv spațiul de stocare al acestuia. Clasele nuvor conține un câmp cu o instanță proprie care să fie returnată atunci când este nevoie de aceasta, fapt neconform cu șablonul de proiectare **Singleton**, funcționalitatea asemănătoare fiind îndeplinită prin crearea unei instanțe și exportarea acesteia(funcționalitate specifică limbajului JavaScript).

2.5.1.1.3. Modulul elementelor vizuale

Scopul logicii stocate în acest sub-modul este de a oferi posibilitatea de a reutiliza și grupa componentele React ce nu au un comportament/stare.

Pentru ca o clasă să poată fi o componentă React validă, aceasta trebuie să extindă clasa **Component** și totodată să suprascrie metoda **render()**(va returna obligatoriu un obiect **J.S.X.**). Unei astfel de componente îi pot fi trimise proprietăți, caracteristică asemănătoare cu pasarea parametrilor unei funcții, valorile proprietăților primite putând fi accesate prin intermediul câmpului **props** al clasei (de exemplu: this.props.<nume\_proprietate>).

Datorită asemănării în implementare a componentelor vizuale, se va descrie o singură componentă ce cuprinde elemente similare cu restul implementărilor, dar și elemente unice, astfel se va descrie clasa **AppOverlay**.

Rolurile proprietăților clasei **AppOverlay**:

* isVisible – determină dacă trebuie afișată componenta la momentul actualizării ecranului;
* onBackdropPress – callback declanșat la apăsarea pe marginea overlay-ului;
* deviceNameValue/securityCodeValue – au rolul de a afișa șirurile de caractere introduse în câmpurile casetelor text la momentul actualizării ecranului;
* onDeviceNameChange/onSecurityCodeChange – callback-uri ce au în vedere gestionarea evenimentelor de schimbare a șirurilor de caractere introduse în câmpurile casetelor text;
* deviceNameErrorMessage/securityCodeErrorMessage – reprezintă mesaje de eroare pentru introducerea invalidă în câmpurile casetelor text;
* deviceId – afișează id-ul dispozitivului selectat;
* onAddPress – callback ce se execută la momentul apăsării pe butonul de adăugare;
* isLoading – proprietate ce determină tipul de vizualizare a butonului(normal sau spinner).

2.5.1.1.4. Modulul de încapsulare a vizualizării și funcționalității programului

În funcție de tipul de platformă pe care este rulată o aplicație mobilă, Android sau iOS, ferestrele principale ale aplicației se numesc activități(eng.: activities), respectiv controlere vizuale(eng.: view controller). Pentru a avea o referință comună asupra ambelor denumiri, elementele vizuale principale ale interfeței vor fi denumite ecrane(eng.: screens).

Așadar, aplicația va fi compusă din două ecrane, unul prin care se permite descoperirea dispozitivelor din proximitate și asocierea cu acestea(clasa **SearchScreen**), iar altul prin care se oferă posibilitatea vizualizării dispozitivelor asociate din apropiere și totodată contactarea acestora(clasa **DevicesScreen**).

Trecerea de la un ecran la altul se va face prin intermediul unui container de navigare(eng.: navigation container) ce va încorpora un navigator de tab-uri (eng.: tab navigator) care va conține referințe către ecranele anterior menționate.

Spre deosebire de clasele pur vizuale, clasele ecran sunt proiectate să aibă comportament, ele vor conține variabile de stare(eng.: state variables) prin intermediul cărora este posibilă redesenarea interfeței în momentul în care una din acestea își modifică valoarea/referința.

Stările clasei **SearchScreen** au următorul rol:

* overlayVisible – valoare booleană prin care se determină dacă se va afișa formularul de adăugare al unui dispozitiv;
* isLoading – permite afișarea unui spinner în locul butonului de căutare cât timp se efectuează căutarea dispozitivelor;
* isCheckingSecurityCode – valoare booleană prin care se permite afișarea unui spinner în locul butonului de adăugare din cadrul formularului de asociere cu un dispozitiv cât timp se așteaptă terminarea execuției acțiunii butonului;
* devices – lista dispozitivelor găsite la momentul ultimei scanări;
* deviceId – id-ul dispozitivului pentru care se dorește asocierea;
* deviceName/securityCode – referințe către valorile introduse în câmpurile formularului de adăugare al unui dispozitiv;
* securityCodeErrorMessage/deviceNameErrorMessage – permit afișarea unor mesaje de eroare în cazul în care câmpurile formularului sunt completate necorespunzător.

În corpul metodei **onSearchButtonPress()** se apelează metoda **searchForDevices()** a clasei **BtManager** și se așteaptă căutarea dispozitivelor din proximitate. La finalul căutării, variabila de stare „devices” va fi actualizată cu noile dispozitive găsite în apropiere(lista poate fi goală) cauzând o redesenare a interfeței. Metoda este asociată evenimentului apăsării pe butonul de căutare situat în antetul ecranului.

După inițializarea clasei **SearchScreen**(aceasta urmează să fie afișată) se apelează metoda **onSearchButtonPress()** pentru a afișa dispozitivele receptor din jur.

Metoda **onSubmitAddDevice()** are rolul de a gestiona evenimentele ce pot să apară în momentul în care butonul din formularul de asociere este apăsat. Sunt verificate câmpurile din formular așa încât să fie corect completate și se actualizează variabilele de stare ce au rol de a afișa mesajele de eroare în privința completării incorecte. Apoi se încearcă adăugarea dispozitivului și memorarea cheii de acces a acestuia în cazul în care operația a fost realizată cu succes. Totodată, dacă va apărea o eroare, variabilele de stare „overlayVisible”, „deviceName” și „securityCode” sunt resetate.

Spre deosebire de clasa **SearchScreen**, clasa **DevicesScreen** are un număr mai mic de stări deoarece aceasta nu cuprinde la fel de multe funcționalități. Variabila de stare „devices”este o referință către lista dispozitivelor asociate, iar variabila „isLoading” permite afișarea unui spinner în locul butonului de căutare cât timp se efectuează căutarea dispozitivelor asociate din proximitate.

După inițializarea clasei **DevicesScreen** sunt aduse din spațiul de stocare modelele dispozitivelor asociate și este apelată metoda **refreshDevices()** în cazul în care există asocieri. Metoda apelează funcția de căutare a clasei **BtManager** urmând să filtreze lista obținută prin compararea id-urilor dispozitivelor asociate cu cele ale dispozitivelor din jur. Dacă au fost găsite dispozitive cu id-uri identice, obiectului din variabila de stare ce conține id-ul respectiv i se va seta câmpul „active” pe valoarea „true”. Acest câmp are rolul de a diferenția dispozitivele asociate ce sunt în proximitate de cele care nu, simultan restricționând și accesul asupra funcției de localizare a fiecărui dispozitiv.

Metoda **onFindPress()** este asociată fiecărui obiect din lista-variabilă de stare „devices” și are rolul de a apela funcția de căutare a fiecărui dispozitiv asociat, iar metoda **onDeletePress()** are rolul de a șterge asocierea cu un dispozitiv.

2.5.1.2. Programul pentru microcontroler

2.5.1.2.1. Tehnologia aleasă pentru implementare

Platforma de lucru aleasă pentru program este **Espressif IoT Development** **Framework**(ESP-IDF) deoarece aceasta este interfața de programare oferită de dezvoltatorii microcontrolerului ESP32. Prin aceasta se pun la dispoziție un sistem de monitorizare al aplicației și librăriile „low-level” necesare dezvoltării unui program(Figura 2.3). Această alegere este motivată și de faptul că restul opțiunilor luate în calcul nu au o manevrabilitate completă asupra microcontrolerului, dezvoltatorii acestora oferind interfețe ce încapsulează doar o parte dintre funcționalitățile librăriilor fiecărui cip.

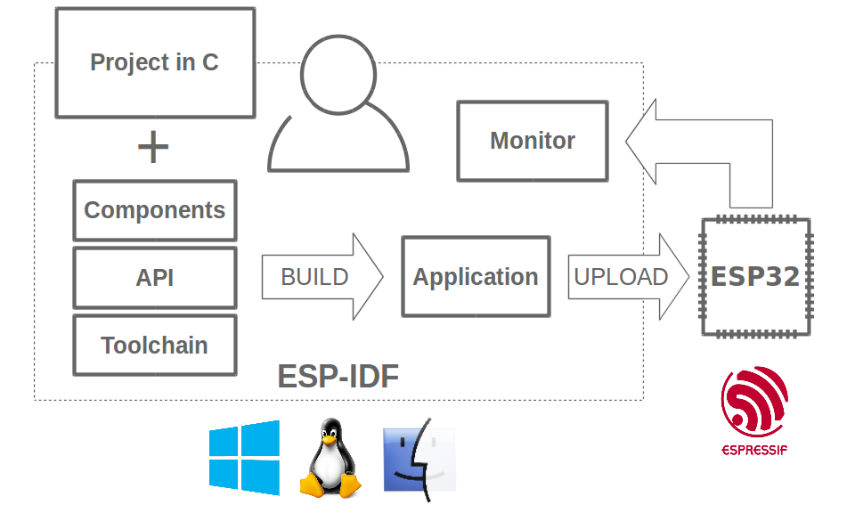


Figura 2.2. Uneltele ESP-IDF

Trebuie luat în vedere și faptul că, deși, platforma este foarte potentă din perspectiva creării unui program, aceasta este și dificil de folosit datorită necesității de a avea cunoștințe asupra limbajului de programare de nivel jos pe care îl are la bază(limbajul **C**).

Mediul de dezvoltare integrat ales pentru a crea proiectul este **PlaformIO**, o extensie adusă editorului **Microsoft Visual Studio Code** ce accelerează și simplifică crearea și livrarea produselor încorporate.

Luând la cunoștință paragrafele anterior enunțate, primul pas în proiectarea programului pentrur microcontroler a fost acela de a crea o diagramă prin care să fie redat fluxul programului(Figura 2.3).

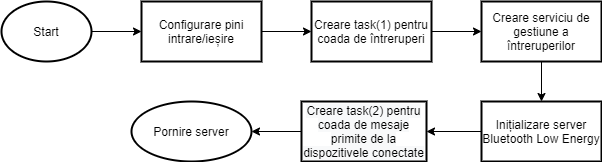


Figura 2.3. Diagrama de flux a programului microcontrolerului

2.5.1.2.2. Configurarea pinilor de intrare/ieșire

Utilizarea pinilor de intrare/ieșire ai microcontroler-ului necesită cunoașterea dispozitivelor legate de aceștia, fapt enunțat în capitolul 2.5.2. Astfel pentru a folosi pinii la care sunt conectate dioda, butonul și buzerul a fost creată funcția **kf\_config\_gpio()**. Funcția conține logica de configurare a pinilor de ieșire așa încât prin aceștia să se poată trimite valori de 1 logic sau semnale **P**.**W**.**M**.(eng.: pulse-width modulation) către periferice. Simultan, aceasta va conține și logica creării evenimentelor de întrerupere la momentul schimbării valorii logice găsite la pinul setat ca intrare.

Modulația în lățime a impulsului este o tehnică pentru obținerea de rezultatelor analogice prin mijloace digitale. Controlul digital este utilizat pentru a crea o undă pătrată(eng.: square wave), un semnal comutat între pornit și oprit.

Controlul intensității luminii emise de diodă nu este necesar, astfel pentru pinii asignați acesteia vor fi trimise doar valori de 1 logic, iar pentru pinul buzerului va fi necesară trimiterea semnalelor **P**.**W**.**M**. deoarece acesta nu poate funcționa altfel.

Pentru a localiza dispozitivul receptor a fost creată funcția **kf\_find()** ce conține logica emiterii într-un mod alternant a semnalelor luminoase și sonore. Aceasta primește un parametru cu rolul de a seta perioada de emitere a semnalelor de localizare.

Modul de funcționare al rutinei de întreruperi este ilustrat prin intermediul unei scheme logice(Figura 2.4). Trebuie avut în vedere faptul că o întrerupere trebuie tratată cât de rapid posibil, astfel, funcția ce are rolul de a gestiona evenimentele apărute trebuie încărcată în memoria RAM pentru a optimiza accesul la aceasta.

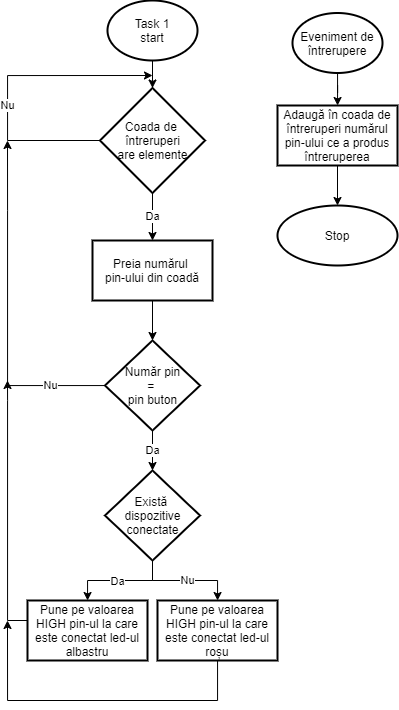


Figura 2.4. Schema logică a rutinei de întreruperi

2.5.1.2.3. Configurarea server-ului Bluetooth Low Energy

Având în vedere API-ul ales pentru programarea microcontrolerului, nu există o modalitate care să facă ușoară utilizarea componentei Bluetooth a dispozitivului, astfel pentru a configura server-ul trebuie:

* inițializată stocarea non-volatilă;
* inițializat și activat controlerul Bluetooth;
* inițializată și activată stiva Bluetooth;
* create funcțiile de gestiune pentru profilul generic de atribut și pentru profilul generic de acces;
* setată limita de date maxim transmise;
* creat task-ul ce gestionează mesajele primite(conform cu Figura 2.5).

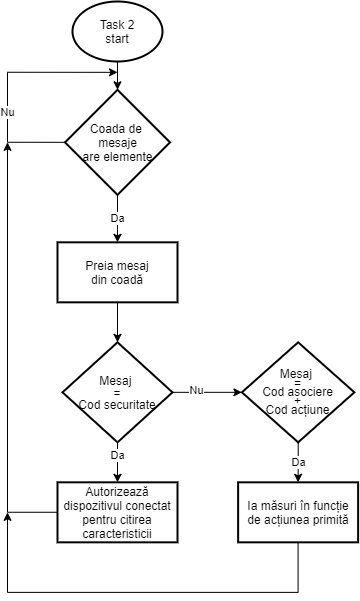


Figura 2.5. Schema logică a task-ului ce interpretează mesajele aflate în coadă

2.5.2. Componenta hardware

Pentru a facilita procesul de dezvoltare al aplicației a fost utilizată o placă de dezvoltare NodeMCU ESP-32S ce are la bază un microcontroler ESP-WROOM-32. Aceasta permite încărcarea și monitorizarea programului microcontrolerului printr-un cablu USB eliminând nevoia de a folosi alte periferice. Totodată, la pinii de intrare/ieșire ai plăcii de dezvoltare au fost conectate un buzzer pasiv, o diodă R.G.B și un buton(Figura 2.7).

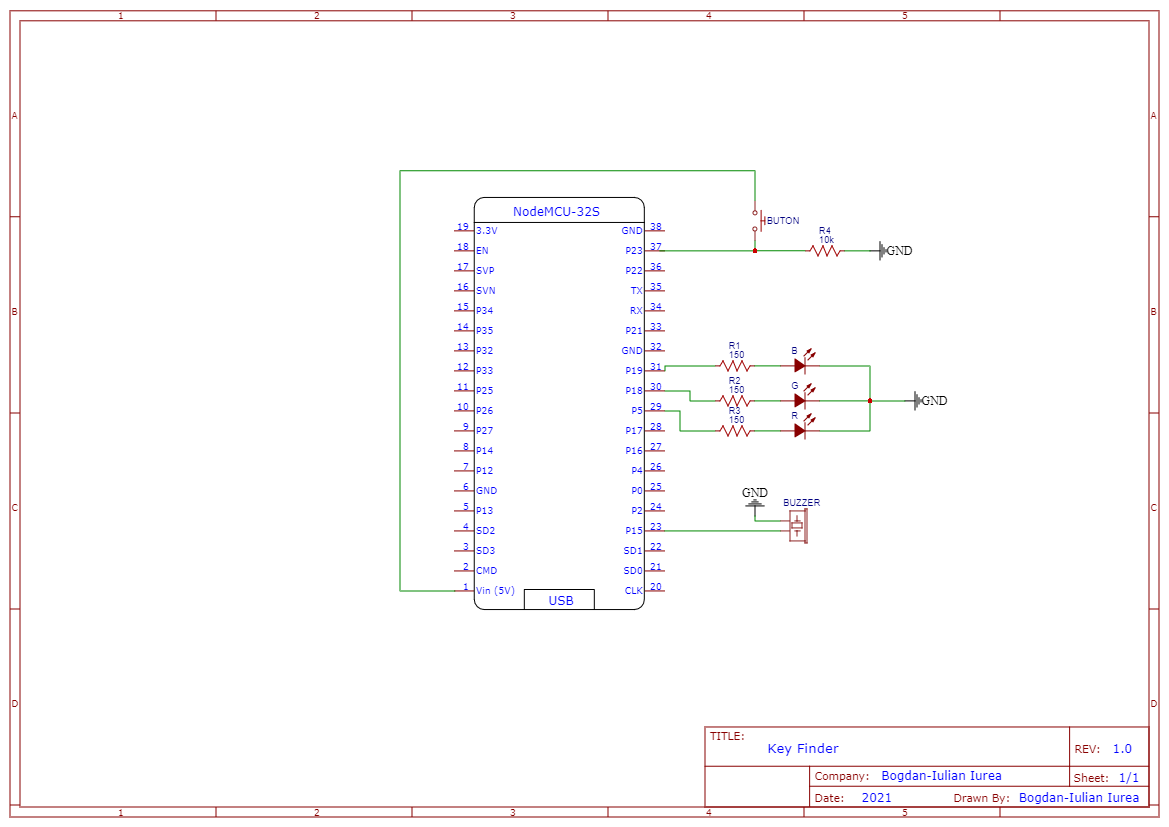


Figura 2.7. Schematica componentei hardware

Capitolul 3. Implementarea aplicației

3.1. Descrierea generală a implementării

3.1.1. Aplicația mobilă

Având în vedere uneltele mediului de lucru folosit pentru a crea aplicația mobilă, au fost abordate paradigma orientată obiect, paradigma funcțională, principiul separării preocupărilor(eng.: **Separation of Concerns – S**.**o**.**C**.), principiul responsabilității, principiul întreținerii și reutilizabilității și principiul coeziunii și al cuplării.

Fiecare funcționalitate a aplicației mobile prezentată la capitolul 2.5.1.1. a fost implementată treptat, urmând diagrama de activități prezentată în Anexa 1.1. Tema de culori a aplicației este încărcată dintr-un fișier JavaScript extern ce conține codurile hexazecimale ale culorilor principale, secundare și suplimentare ale interfeței cu utilizatorul.

Cunoscând faptul că JavaScript este un limbaj ce are un singur fir de execuție, dar care este totodată și asincron, au fost utilizate callback-uri, cu precizarea că au fost evitate implementările asemenea „callback hell” prin utilizarea **promise**-urilor și a **funcțiilor asincrone**.

3.1.2. Programul pentru microcontroler

Configurarea pinilor microcontrolerului se face prin utilizarea unei structuri de configurare(**gpio\_config\_t)** ce are în componență o mască de 64 de biți, fiecare poziție semnificând numărul pinului folosit ca intrare/ieșire. Valoarea(„1 sau 0”) găsită la fiecare poziție arată dacă pinul este folosit sau nu. Structura conține detalii despre activarea întreruperilor, modul de utilizare(intrare/ieșire) și tipul de rezistor folosit pentru fiecare pin menționat în mască. Totodată, în această funcție sunt create coada de întreruperi, serviciul ce adaugă elemente în coadă la momentul producerii unei întreruperi și task-ul ce gestionează elementele aflate în coadă. Configurarea pinului la care este conectat buzer-ul pasiv se face folosind librăriei „ledc” deoarece prin aceasta se oferă o interfață prin care se pot genera semnale **P**.**W**.**M**.

Pentru a crea server-ul Bluetooth Low Energy a fost studiat și adaptat modelul din documentația oferită de dezvoltatorii **ESP-IDF**, astfel server-ul are următoarea configurație:

* Există doar un profil ce expune un serviciu cu o caracteristică ce are un singur descriptor;
* Citirea caracteristicii serviciului de advertising va returna codul de asociere al serviciului dacă cererea a fost precedată de o autorizare realizată cu succes și va reseta variabila ce verifică dacă cererea este autorizată. Dacă cererea nu este autorizată, atunci se va returna mesajul „NO\_DATA”;
* Scrierea pe caracteristica serviciului va avea ca efect trimiterea mesajului scris către un task. Task-ul va interpreta mesajul și va lua măsuri în funcție de natura acestuia. Dacă mesajul este egal cu cheia de securitate, atunci se va pune pe „true” variabila de autorizare a serviciului, iar dacă mesajul este reprezentat de codul de acces concatenat cu mesajul specific operației de localizare, atunci se va iniția rutina de localizare a dispozitivului;
* Evenimentul de deconectare va reseta variabila ce are ca scop memorarea răspunsului validării cheii de securitate;
* Unitatea maximă de transfer este setată la 23 octeți.

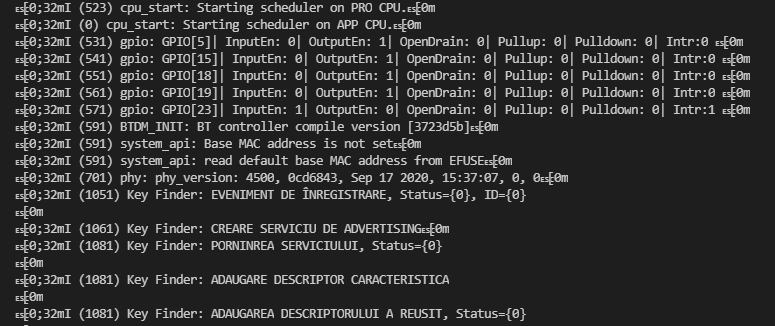


Figura 3.1. Configurarea pinilor de intrare/ieșire și a server-ului Bluetooth Low Energy

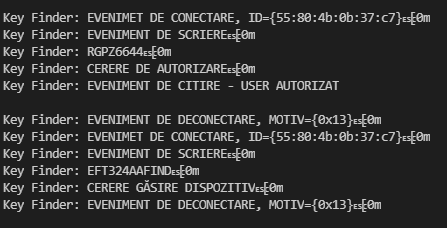


Figura 3.2. Cererea de asociere cu și cererea de localizare

3.1.3. Implementarea componentei hardware

Pentru a implementa componenta hardware a fost urmată schema din Figura 2.7. Astfel cablajul a fost realizat în felul următor(Figura 3.3):

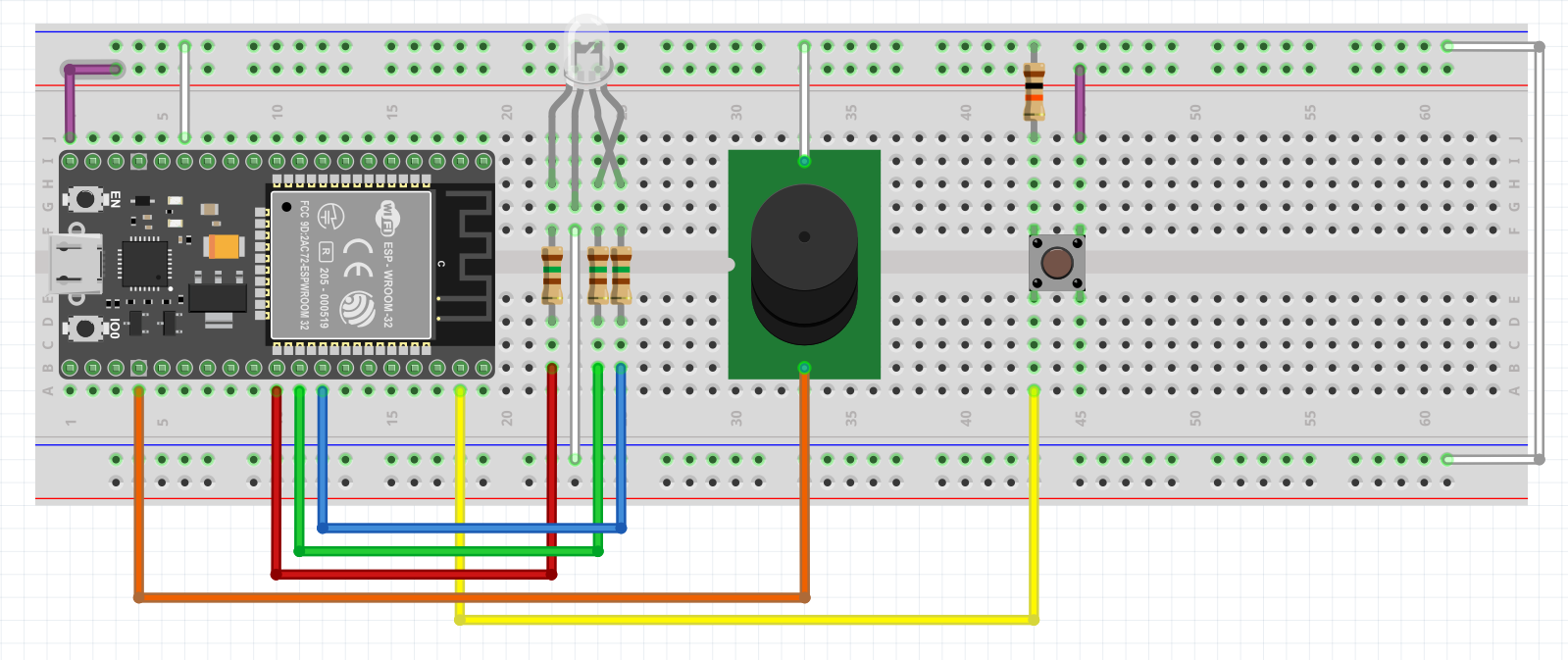


Figura 3.3. Implementarea componentei hardware

3.2. Probleme speciale/dificultăți întâmpinate și modalități de rezolvare

Inițial, aplicația mobilă a fost creată prin intermediul **Expo**, un lanț de unelte construite în jurul mediului **React Native**, deoarece instrumentele cuprinse de mediu sunt mult mai simplu și rapid de folosit în comparație cu cele ale **React Native**. Problema întâmpinată a fost aceea că deși uneltele sunt foarte potente, mediul nu oferă și posibilitatea de a folosi dependințe native. Pentru a rezolva această problemă, proiectul **Expo** a trebui migrat către un proiect **React Native**.

A doua problemă întâmpinată este aceea că erorile **React Native** sunt în unele cazuri extrem de vagi. Spre exemplu, adăugarea unui pachet extern(prin intermediul managerului de pachete **Node**)fără a-i face legătura cu librăriile native poate cauza o eroare fără mesaj, stiva de apeluri fiind greu de citit/urmărit sau chiar neexistând în unele cazuri. Pentru a evita o astfel de situație, pachetele externe trebuie instalate și gestionate prin intermediul comenzilor specifice React Native. Astfel, pentru a adăuga o dependință externă se poate utiliza comanda **react-native install <nume-pachet-node>** sau comanda **npm i <nume-pachet-node>** urmată de comanda **react-native link <nume-pachet-node>**.

A treia problemă întâmpinată este aceea că dezvoltarea aplicațiilor mobile necesită un spațiu de lucru generos deoarece sunt necesare o multitudine de ferestre/unelte deschise pentru a putea avea un control asupra fluxului implementării. Pentru a putea rezolva această problemă este recomandată folosirea unui monitor suplimentar, dar și testarea funcționalităților prin intermediul unui dispozitiv mobil, ci nu printr-un emulator. Simultan, implementările pur **JavaScript** pot fi create/testate și înafara proiectului(în linia de comandă Node sau cea a navigatorului Web), iar componentele React ce nu depind de librăriile native pot fi create/testate într-un proiect Expo separat sau prin intermediul Expo Snack.

O altă problemă întâmpinată a fost aceea că mediul de lucru folosit pentru programarea microcontrolerului ESP32 nu oferă o interfață simplă pentru a inițializa server-ul, astfel a trebuit studiată în detaliu arhitectura standardului Bluetooth Low Energy.

3.3. Idei originale

Având în vedere subiectul temei propuse, originalitatea acesteia nu poate fi adusă în discuție datorită multitudinii de produse/aplicații existente pe piață ce servesc același scop. Ceea ce este implementat este similar ca funcționalitate cu variantele deja existente.

3.4. Comunicarea cu alte sisteme și stocarea informațiilor

Aplicația mobilă comunică cu microcontrolerul folosind comunicațiile prin aer prin intermediul standardului tehnologic Bluetooth Low Energy. Aplicația mobilă poate vizualiza dispozitivele-receptor din apropiere și poate trimite cereri către acestea. Pentru ca dispozitivele să poată comunica, aplicația mobilă va scrie și citi caracteristica serviciului de advertising a dispozitivului receptor.

Asocierea între dispozitive se face în felul următor:

1. Se realizează conexiunea între dispozitive;
2. Aplicația mobilă trimite un mesaj ce conține cheia de acces;
3. Programul pentru microcontroler verifică cheia primită și actualizează variabila de autorizare în funcție de validarea efectuată;
4. Aplicația mobilă să facă o cerere de citire a caracteristicii serviciului dispozitivului receptor;
5. Server-ul returnează codul de acces sau mesajul „NO\_DATA” în funcție de valoarea variabilei de autorizare;
6. Aplicația mobilă va salva codul de acces în spațiul de stocare al dispozitivului pentru a-l folosi pentru viitoare cereri fără a mai fi nevoie de autorizare;
7. Se realizează deconectarea dintre dispozitive.

Pentru a trimite semnalul de activare al funcției de localizare, aplicația mobilă va prelua din spațiul de stocare codul de acces al dispozitivului, va realiza conexiunea cu acesta și apoi va expedia codul alături de mesajul operațiunii de localizare(FIND). La final conexiunea dintre dispozitive este întreruptă.

Pentru a contura comunicarea între dispozitivul mobil și microcontroler a fost creată următoarea diagramă de activități(Figura 3.4).

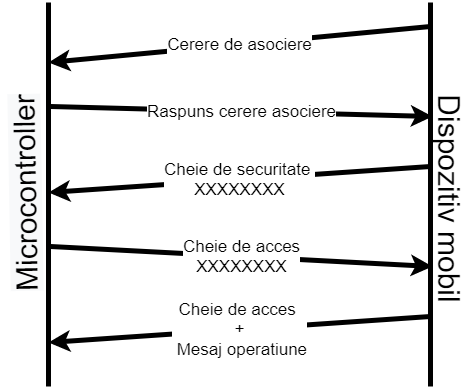


Figura 3.4. Diagramă de activități ce ilustrează comunicarea dintre dispozitive

3.5. Interfața cu utilizatorul

Capitolul 4. Testarea aplicației și rezultate experimentale

4.1. Punerea în funcțiune/lansarea aplicației

4.2. Testarea sistemului

4.2.1. Testarea aplicației mobile

4.2.2. Testarea programului pentru microcontroler

4.2.3. Testarea componentei hardware

4.3. Aspecte legate de încărcarea procesorului, memoriei, limitări în ce privește transmisia datelor/comunicarea

4.4. Aspecte legate de fiabilitate/securitate

4.5. Rezultate experimentale

4.6. Utilizarea sistemului

Concluzii

Bibliografie

Anexe

Anexa 1. Diagrama de activități a aplicației mobile

