UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DOMENIUL: Calculatoare și Tehnologia Informației

SPECIALIZAREA: Tehnologia Informației

**Dispozitiv localizabil prin intermediul unei aplicații mobile**

**KeyFinder**

LUCRARE DE DIPLOMĂ

Coordonator științific

Ș.l.dr. Nicolae Botezatu

Absolvent

Bogdan-Iulian Iurea

Iași, 2021

Declarație pe proprie răspundere

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 de găsi obiectele frecvent 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 ales 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 electroluminiscentă și un buzer(eng.: buzzer) prin care se vor emite semnalele de localizare. Simultan, va pune la dispoziție 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 electroluminiscente și va emite un semnal sonor în ritmul aprinderii diodei. 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 strict 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 posibilitatea de a elimina asocierea, iar pentru cel aflat în proximitate să se ofere și posibilitatea emiterii semnalului de activare al acestuia.

1.1.2. Justificarea abordării

Această temă 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.

Tema propusă 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”, astfel tema se încadrează în caracteristicile domeniului **IoT,** un domeniu de actualitate, aducând un plus în vederea luării deciziei abordării unui astfel de proiect.

1.2. Domeniul și contextul abordării temei

Considerând obiectivele enunțate la capitolul 1.1.1. 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. Astfel, se pot lua decizii în urma informațiilor oferite(dacă antrenamentul sportiv poate continua, dacă este necesar să se reducă starea de agitație etc.);
* Dispozitive ce pot fi localizate prin GPS/Bluetooth(eng.: tracking device) folosind o aplicație mobilă/Web;
* Aparatură casnică inteligentă(frigidere/mașini de spălat/aspiratoare inteligente) ce poate fi controlată prin telefon/calculator;

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

1.2.1. Sisteme încorporate

Un sistem este un aranjament în care toate unitățile sale asambla lucra împreună în conformitate cu un set de reguli. Acesta poate fi, de asemenea, definit ca un mod de lucru, organizare sau 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 ca 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 microcontroller/microprocesor ce poate fi programat și care poate utiliza un sistem de control în timp real.

Pentru a programa un microcontroller 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 pot fi 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 în 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

1.4. Specificații privind caracteristicile așteptate de la aplicație

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 platformei hardware pe care va fi executată aplicația

O platformă hardware este compusă din seturi de elemente hardware compatibile ce permit executarea de aplicații 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 obiectivul propus sunt necesare două dispozitive ce pot comunica, fiecare având un rol esențial în proces, unul ce va trimite instrucțiuni/comenzi, iar altul ce le va intercepta și interpreta. Au fost analizate toate opțiunile în vederea selecției 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 microcontroller-ul ESP32, alegere motivată de independența cipului de periferice, având integrate module de Bluetooth și Wi-Fi.

Principalul motiv al alegerii tehnologiei Bluetooth în schimbul tehnologiilor cu funcționalități asemănătoare este acela că aceasta oferă posibilitatea existenței unei conexiuni între dispozitive fără a mai exista intermediari care pot manipula pachetele primite, astfel este asigurată securitatea la nivel de transfer al datelor. Concomitent, utilizarea Bluetooth-ului Low Energy în locul Bluetooth-ului „clasic” a fost motivată de faptul că deși cel clasic oferă posibilitatea de a transfera pachete mari de date într-o perioadă mai scurtă, acesta consumă mai multă energie față de succesorul său.

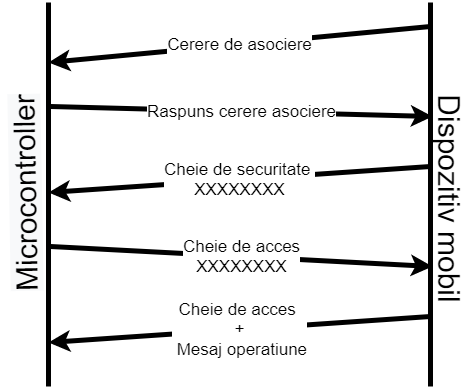
2.2. Modulele proiectului

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 microcontroller-ului ESP32. Componenta hardware este reprezentată de microcontroller 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.

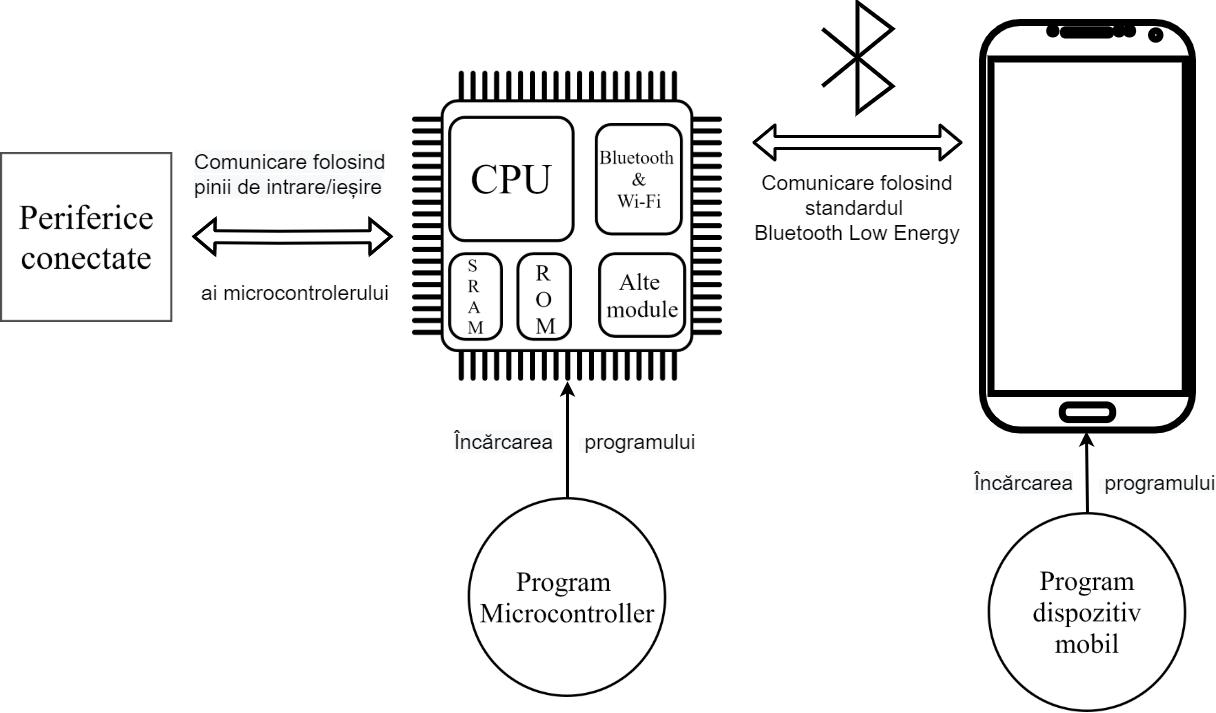
Programul pentru microcontroller va aștepta cererile de asociere, va interpreta mesajele primite de la dispozitivele asociate, va lua măsuri în funcție de natura mesajului primit și va gestiona evenimentele de întrerupere apărute în perioada de execuție. Așadar, acesta va cuprinde logica de inițializare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

Pentru a contura comunicarea între dispozitivul mobil și microcontroller a fost creată următoarea diagramă de activități(Figura 1.1).



Comunicarea dintre dispozitive

În ceea ce privește modulul hardware, se ia în considerare siguranța implementării circuitului electric din punct de vedere electrotehnic. Prin acesta se permite crearea evenimentelor de întrerupere ale programului microcontroller-ului(prin apăsarea unui buton), cu scopul de a afișa starea curentă a modulului Bluetooth(dacă există asocieri sau nu) prin semnale luminoase(se folosește un L.E.D.). Simultan, este folosit un buzzer care alături de dioda electroluminiscentă vor ajuta la localizarea dispozitivului.



Interacțiunea dintre modulele proiectului

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. În timpul perioadei de studiu a opțiunilor luate în considerare, aplicațiile Android au următoarele avantaje:

* 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 ca aplicația să poată fi fragmentată 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 poate fi amânată.

2.3.2. Avantaje și dezavantaje ale microcontroller-ului ESP32

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

* Wi-Fi și Bluetooth integrat, module ce reduc necesitatea de periferice;
* 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.;
* Memoria flash este criptată;
* Consum de energie redus.

Dezavantajul scos în evidență este acela că prețul de achiziție al cipului este aproape dublu comparativ cu predecesorul său, microcontroller-ul 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 există interferențe cu alte dispozitive fără fir;
* Permite conectarea mai multor dispozitive la un singur dispozitiv;
* Securitatea transferului de date este asigurată de faptul că nu există intermediari care să transmită datele de la un capăt la altul;
* Compatibilitatea 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 scurtă;
* 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 care 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.

Nu în ultimul rând, se are în vedere și capabilitatea tehnologiei Bluetooth de a putea oferi suportul ca un dispozitiv să poată avea mai multe conexiuni. Acest lucru nu va fi luat în calcul deoarece este neutru din punct de vedere al obiectivelor proiectului, fapt ce duce la o limitare de funcționare.

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. Simultan, o componentă hardware este o unitate fizică autonomă ce poate fi încorporată într-un sistem complex(de exemplu un microcontroller 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 proiectului, 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 și ștergerea unei asocieri existente. Astfel s-a creat diagrama de flux(eng.: flow diagram) a ecranelor aplicației, diagramă ilustrată în Figura 1.3.

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:
* Asemănare cu dezvoltarea aplicațiilor web;
* 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;
* Eficient din punct de vedere al costului și timpului;
* Folosește un limbaj multi-paradigmă;
* Extensibilitatea aplicației pe mai multe platforme;

Atuul acestui framework este acela că 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).

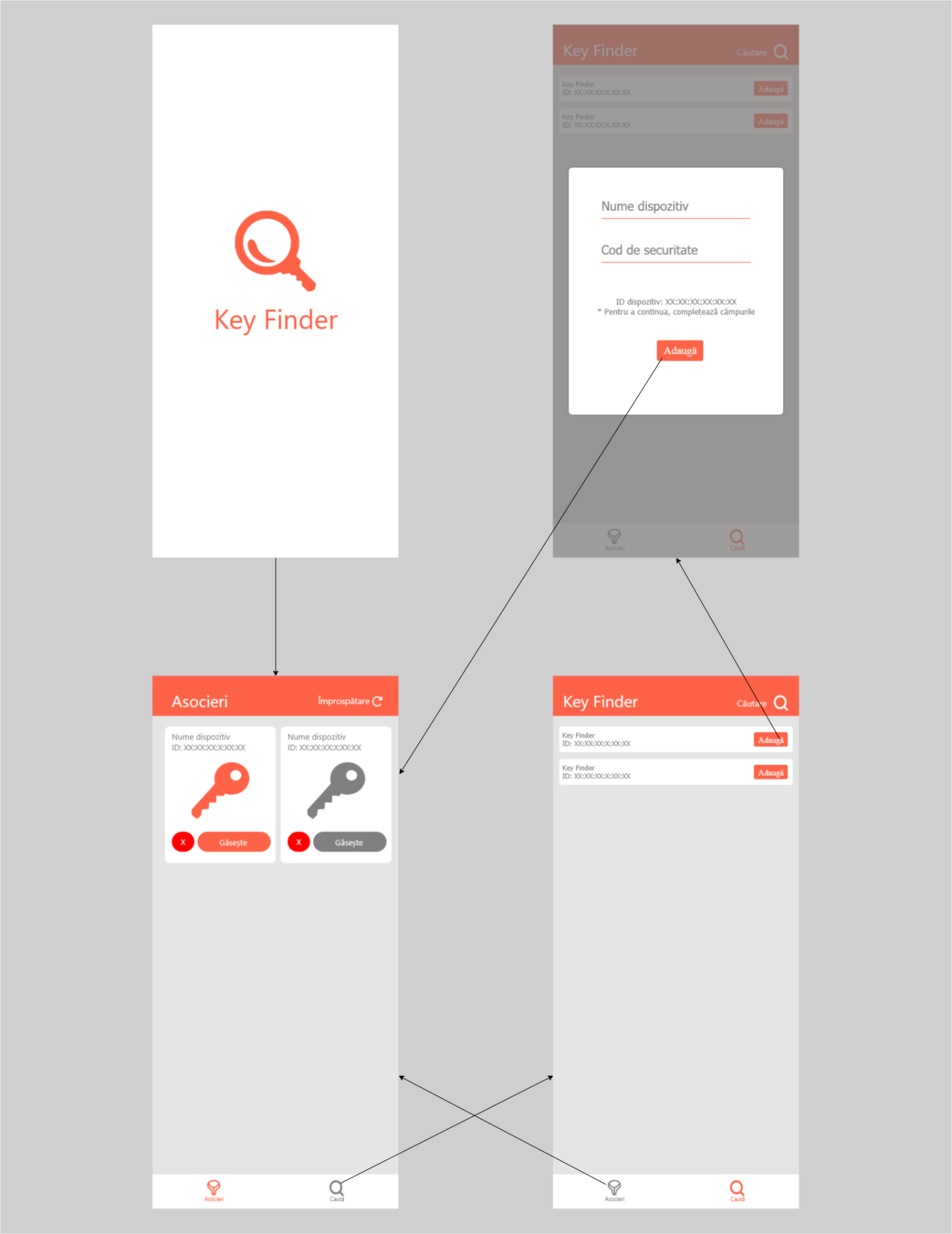


Diagrama de flux a ecranelor aplicației

2.5.1.1.2. Componenta pentru funcționalități

Clasa **BtManager**(Figura 1.4) are o singură instanță pe tot parcursul programului, deoarece se dorește ca o unică entitate din program să poată accesa componenta Bluetooth a dispozitivului. La momentul creării instanței clasei se încarcă datele din fișierul „devices-data.json”, fișier ce conține id-urile serviciilor Bluetooth de interes și câmpuri ce descriu tipul operațiunilor executate de dispozitivele țintă.

Din câte se observă, clasa **BtManager** nu 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 asemănătoarea este îndeplinită prin crearea unei instanțe și exportarea acesteia(funcționalitate specifică limbajului JavaScript).

Metoda **searchForDevices**(Figura 1.5) returnează un **Promise** și are ca rol descoperirea dispozitivelor Bluetooth Low Energy din apropiere al căror identificator de serviciu de advertising se găsește în lista de id-uri de advertisement încărcată la momentul creării instanței **BtManager**. Scanarea se va face timp de trei secunde urmând ca apoi să se returneze lista dispozitivelor din proximitate.

Metoda **addDevice**(Figura 1.6) va primi ca parametri de intrare id-ul dispozitivului cu care se dorește asocierea și cheia de securitate, apoi va realiza conexiunea între capete și va descoperi serviciile și caracteristicile Bluetooth ale dispozitivului receptor. Urmează trimiterea cheii 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 un cod de acces prin intermediul căruia se pot face cereri către serviciul dispozitivului respectiv fără a mai fi nevoie de o viitoare autorizare.

Metoda **findDevice**(Figura 1.7) permite trimiterea mesajului de activare al funcției de găsire a dispozitivului-receptor. Acest lucru se face prin trimiterea unui pachet ce conține codul de acces al dispozitivului și mesajul specific operațiunii de găsire către dispozitivul cu id-ul primit ca parametru.

Se observă că 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).

Totodată, a fost creată și metoda **throwErrorByType**(Figura 1.8) prin care se analizează tipul de eroare aruncat de instanța clasei **BleManager**.Necesitatea acesteia este datorată faptului că la momentul apariției unei erori librăria creează un tip generic de eroare(**BtError**), astfel identificarea erorii trebuie făcută prin evaluarea mesajului acesteia, ci nu prin tipul instanței.

2.5.1.1.3. Componenta elementelor vizuale

Datorită asemănării în implementare a componentelor vizuale, se va prezenta o singură componentă ce va cuprinde elemente similare cu celelalte implementări, dar și elemente unice încorporate, astfel se alege pentru descriere clasa **AppOverlay**(Figura 1.9).

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(asemănător cu declararea obiectului **Overlay**), caracteristică asemănătoare cu pasarea parametrilor unei funcții. Valorile/referințele proprietăților primite pot fi accesate prin intermediul câmpului **props** al clasei (de exemplu: **this.props.<nume\_proprietate>**).

Rolurile proprietăților clasei AppOverlay:

* isVisible – proprietate ce determină dacă componenta trebuie afișată la momentul actualizării ecranului;
* onBackdropPress – proprietate ce are rol de callback pentru evenimentul de apăsare pe marginea Overlay-ului;
* deviceNameValue/securityCodeValue – proprietăți primite pentru a afișa valoarea introdusă în casetele text la momentul actualizării ecranului;
* onDeviceNameChange/onSecurityCodeChange – callback pentru a gestiona evenimentele de schimbare a valorilor casetelor text;
* deviceNameErrorMessage/securityCodeErrorMessage – proprietăți trimise pentru a afișa mesaje de eroare pentru valorile invalide introduse în casetele text;
* deviceId – proprietate ce are ca scop afișarea id-ului dispozitivului selectat;
* onAddPress – callback ce gestionează evenimentul de apăsare pe buton;
* isLoading – proprietate ce determină tipul de vizualizare a butonului;

2.5.1.1.4. Componenta 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 controller-e vizuale(eng.: view controller). Pentru a avea o referință comună asupra ambelor denumiri, elementele vizuale ale interfeței vor fi denumite ecrane(eng.: screens).

Așadar, aplicația va fi compusă din două ecrane principale, unul prin care se permite descoperirea dispozitivelor din proximitate și asocierea cu acestea(clasa **SearchScreen**), iar altul prin care se oferă posibilitatea de a vedea care sunt dispozitivele asociate din apropiere și totodată contactarea acestora(clasa **DevicesScreen**). Datorită similitudinii între implementările acestor componente, se va descrie amănunțit un singur obiect-ecran, clasa **SearchScreen**(Figura 1.10).

Trecerea de la un ecran la altul se face prin intermediul unui container de navigare(eng.: navigation container) care încorporează un navigator de tab-uri (eng.: tab navigator) ce are în componență ecranele anterior menționate.

Spre deosebire de clasele pur vizuale, clasele ecran sunt proiectate să conțină variabile de stare(eng.: state) prin intermediul cărora este posibilă redesenarea interfeței în momentul în care una din stări își modifică valoarea.

Stările clasei **SearchScreen** au următorul scop:

* overlayVisible – valoare booleană prin care se determină dacă se va afișa overlay-ul ce conține formularul de adăugare a unui dispozitiv;
* isLoading – valoare booleană prin care se permite afișarea unui spinner în locul butonului de căutare cât timp se efectuează scanarea de dispozitive;
* 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 – variabile ce memorează valorile introduse în câmpurile formularului de adăugare a unui dispozitiv;
* securityCodeErrorMessage/deviceNameErrorMessage – variabile ce permit afișarea unor mesaje de eroare în cazul în care câmpurile formularului sunt completate necorespunzător.

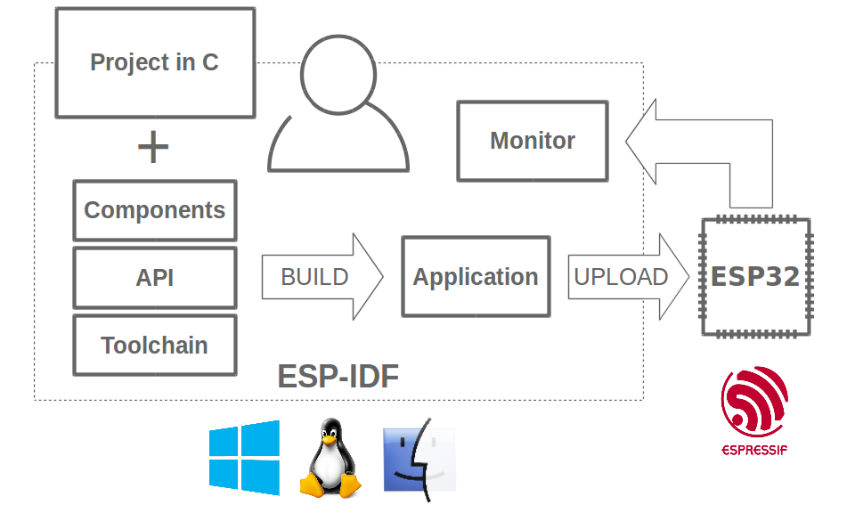
În corpul metodei **onSearchButtonPress** se apelează metoda **searchForDevices** a clasei **BtManager** și se așteaptă scanarea dispozitivelor din proximitate. La finalul scanării, variabila de stare „devices” va fi actualizată cu noile dispozitive găsite în apropiere(lista poate fi goală) și va cauza o redesenare a interfeței.

Metoda **onAddDeviceButtonPress** 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.

2.5.1.2. Programul pentru microcontroller

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 microcontroller-ului. Prin aceasta se pun la dispoziție un sistem de monitorizare al aplicației și librăriile „low-level” necesare dezvoltării unui program(Figura 1.11). Totodată această alegere este motivată și de faptul că restul opțiunilor luate în calcul nu au o manevrabilitate completă asupra microcontroller-ului, dezvoltatorii acestora oferind interfețe peste librăriile native ce încapsulează doar o parte dintre funcționalități.



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 pe care îl are la bază(limbajul **C**).

Mediul de dezvoltare 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 a fost acela de a crea o diagramă prin care să fie redat fluxul programului(Figura 1.12).

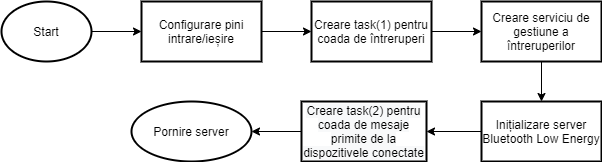
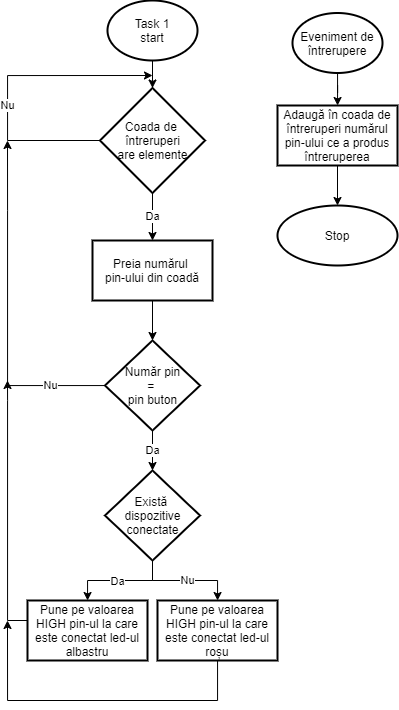


Diagrama de flux a programului microcontroller-ului

2.5.1.2.2. Configurarea pinilor de intrare/ieșire

Utilizarea pinilor de intrare/ieșire ai microcontroller-ului necesită cunoașterea dispozitivelor legate de aceștia, fapt enunțat în capitolul 1.5.2, astfel pentru a folosi pinii la care sunt conectate dioda și butonul a fost creată funcția **kf\_config\_gpio**(Figura 1.13). Configurarea acestora se face prin utilizarea unei structuri de configurație(**gpio\_config\_t)** care are în componență o mască de 64 de biți, fiecare poziție din aceasta semnificând numărul pinului folosit ca intrare/ieșire. Valoarea(„1 sau 0”) găsită la fiecare poziție arată dacă pinul este folosit sau nu.

Simultan, 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 prin folosirea librăriei „ledc” deoarece prin aceasta se pot genera semnale **P.W.M.**, semnale fără de care buzzer-ul pasiv nu poate funcționa.

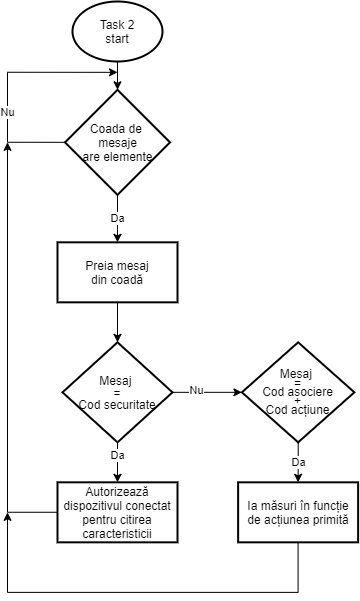


Schema logică a rutinei de întreruperi

2.5.1.2.3. Configurarea server-ului Bluetooth Low Energy

Pentru a crea server-ul, a fost preluat și adaptat exemplul din documentația oferită de dezvoltatorii ESP-IDF, astfel au fost adăugate următoarele funcționalități:

* Citirea caracteristicii serviciului de advertising va returna codul de asociere al serviciului dacă cererea a fost precedată de o autorizare realizată cu succes;
* Trimiterea unui pachet de date către caracteristica serviciului va avea ca efect trimiterea mesajului primit către un task. Task-ul va interpreta mesajul și va lua măsuri în funcție de natura acestuia(conform schemei logice ilustrată în figura 1.15);
* Evenimentul de deconectare va reseta variabila ce are ca scop memorarea răspunsului validării cheii de securitate.

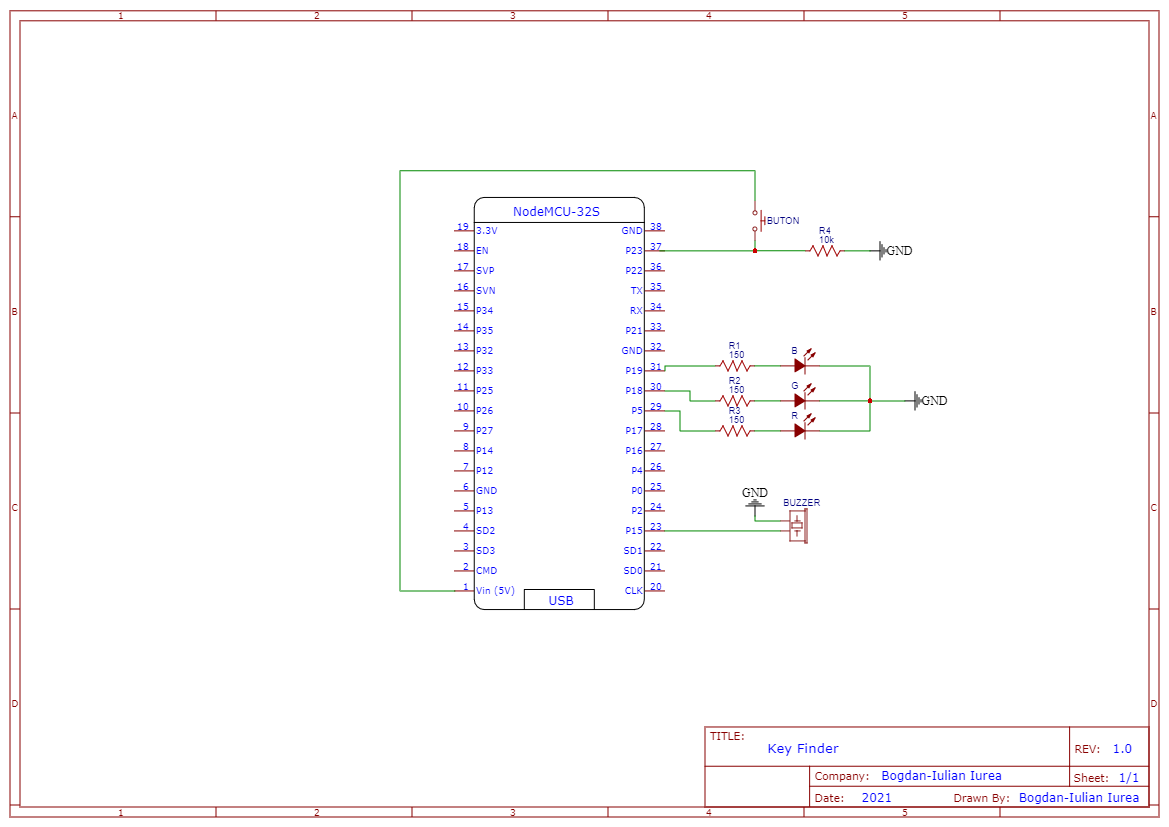


Schema logică a task-ului ce interpretează mesajele aflate în coadă

2.5.2. Componenta hardware

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.

Pentru a facilita procesul de dezvoltare al aplicației a fost utilizată o placă de dezvoltare NodeMCU ESP-32S ce are la bază un microcontroller ESP-WROOM-32. Aceasta permite încărcarea și monitorizarea programului microcontroller-ului printr-un cablu USB eliminând nevoia de a folosi alte periferice. Totodată, la pinii de intrare/ieșire ai plăcii de dezvoltare au fost conectate o diodă R.G.B., un buzzer pasiv și un buton(Figura 1.16).



Schematica componentei hardware

Capitolul 3. Implementarea aplicației

3.1. Descrierea generală a implementării

3.2. Probleme speciale/dificultăți întâmpinate și modalități de rezolvare

3.3. Idei originale, soluții noi

3.4. Comunicarea cu alte sisteme și salvarea/stocarea informațiilor

3.5. Interfața cu utilizatorul

3.6. Implementarea componentei hardware

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 microcontroller

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