

UNIVERSITATEA POLITEHNICA BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE



PROIECT DE DIPLOMĂ

Învățarea din context pe dispozitive mobile inteligente

Coordonator științific:
S.I. dr. ing. Andrei Olaru

Absolvent:
Maximilian Alexandru

BUCUREȘTI

2017

Abstract.

Învățarea din context pe dispozitive mobile este utilizată la recunoașterea activităților umane de zi cu zi. Numărul dispozitivelor mobile accesibile, ce dețin putere de procesare mare și un număr ridicat de senzori, este în creștere. Din acest motiv, din ce în ce mai multe aplicații se axează pe recunoaștere de activități umane, pentru a oferi o multitudine de beneficii utilizatorului. Acestea variază de la aplicații simple, de exemplu setarea automată a luminozității, la aplicații complexe, ce pot detecta activități de genul a merge, a alerga, a sări, etc.

Scopul acestei lucrări este recunoaștere de activități complexe folosind șase senzori integrați în majoritatea dispozitivelor mobile. Acest lucru este realizat prin implementarea unei aplicații Android. În primul rând, este descrisă structura contextului. Aceasta este formată din valorile a șase senzori diferiți: lumină, proximitate, sunet, gravitație, accelerație și temperatură internă. În al doilea rând, este analizat un algoritm de învățare automată. Acesta este antrenat folosind datele strânse de la senzori și feedback-ul utilizatorului. În final, volumul telefonului este setat automat în funcție de activitatea detectată de către algoritm. Din punct de vedere al rezultatelor, este analizat în amănunt un set de date colectat pe parcursul a două zile. Precizia recunoașterii de activități este mai mare de 90% în condiții ideale.

Cuprins

1. Introducere	7
2. Învățarea din context și recunoașterea activităților umane	9
2.1 State of The art	9
2.2 Abordări existente	10
3. Descrierea soluției propuse.....	15
3.1 Arhitectura implementării	15
3.2 Avantaje și dezavantaje	15
4. Implementarea soluției	17
4.1 Senzorii dispozitivelor mobile Android	17
4.1.1 Senzorul de gravitație	18
4.1.2 Senzorul de accelerație liniară	19
4.1.3 Senzorul de lumină	19
4.1.4 Senzorul de sunet	19
4.1.5 Senzorul de proximitate	20
4.1.6 Senzorul bateriei	20
4.1.7 Citirea senzorilor în Android.....	20
4.2 Setul de date	21
4.3 Structura aplicației Android	22
4.3.1 Interfața aplicației	22
4.3.2 Structura bazei de date	25
4.3.3 Structura internă a aplicației	27
4.4 Algoritmul de recunoaștere de activități	28
4.4.1 Descrierea algoritmului ID3.....	28
4.4.2 Integrarea algoritmului ID3 în aplicație	29
4.4.3 Procesul de recunoaștere de activități	30
4.4.4 Implementarea algoritmului în mediul de dezvoltare Android	31
5. Evaluarea implementării	33
5.1 Evaluarea unui scenariu teoretic	33
5.2 Evaluarea unui exemplu concret de funcționare	35
5.3 Evaluarea consumului de resurse	39
6. Concluzii	41
6.1 Sumar și contribuții	41
6.2 Dezvoltări ulterioare	41

1. Introducere

În ultimii ani, dispozitivele mobile au devenit din ce în ce mai răspândite și folosite în viața de zi cu zi. Acestea sunt capabile să ofere utilizatorului o gamă largă de funcționalități, una mai complexă decât cealaltă. Majoritatea dispozitivelor mobile sunt dotate cu procesoare rapide și memorii de dimensiuni mari, dar și cu metode de eficientizare a consumului de baterie. Din acest motiv, din ce în ce mai mulți senzori sunt integrați în aceste dispozitive, cum ar fi: camere de fotografiat performante, senzori de recunoaștere de amprente, senzori de mișcare, senzori de lumină, etc. Un domeniu care s-a dezvoltat datorită acestor tehnologii este recunoașterea de activități umane. Folosind resursele de procesare performante și senzorii dispozitivelor mobile, aplicațiile sunt capabile să recunoască ce activități execută utilizatorul în timp real cu o precizie deosebit de mare.

Marea majoritate a aplicațiilor de acest gen sunt create pentru a rula pe sistemul de operare Android. Platforma de dezvoltare a aplicațiilor pentru dispozitive Android este open source și oferă toate resursele software necesare pentru a putea accesa și folosi resursele hardware, inclusiv senzorii, în scopul creării de aplicații inovative. Un mare procent din aceste aplicații se axează pe recunoașterea activităților fizice (a alerga, a sta, a merge, etc) și folosesc date provenite de la senzori de mișcare precum accelerometru, giroscop, GPS, etc. Acestea sunt folosite într-o multitudine de domenii existente, inclusiv în domeniu medical. O aplicație din acest domeniu se axează pe monitorizarea activităților pacienților ce suferă de boli precum Alzheimer.

Această lucrare analizează o modalitate diferită de recunoaștere a activităților umane de zi cu zi. Folosirea exclusivă a senzorilor de mișcare implică o procesare destul de complexă a datelor provenite de la aceștia și este limitată la detecția activităților fizice. Soluția propusă și descrisă în această lucrare utilizează o varietate mai mare de senzori, care sunt folosiți într-un mod diferit față de alte abordări existente. Scopul acestei abordări este de a oferi o gamă mai largă de activități ce pot fi recunoscute de către un dispozitiv mobil, fără a fi nevoie de procesări complexe. În plus, utilizatorului i se oferă opțiunea de a asocia anumite acțiuni care să fie executate de către dispozitivul mobil în funcție de activitatea recunoscută. În final, utilizatorul nu trebuie să se îngrijoreze din cauza încălcării intimității personale, deoarece aplicația nu are acces la internet, toate procesările de date personale făcând-se local.

Utilitatea aplicației este descrisă de următorul scenariu posibil: în timpul nopții, dispozitivul mobil detectează că utilizatorul doarme și setează volumul dispozitivului mobil pe silențios. Acest lucru este posibil deoarece contextul în care utilizatorul doarme poate fi descris utilizând senzorul de lumină (întuneric) și microfonul (liniște). Un mare avantaj al aplicației este posibilitatea de a descrie contextul activității în orice mod. Pentru unii utilizatori activitatea de a dormi s-ar putea să aibă alt context (lumină slabă și zgomot ridicat). Utilizatorul este cel care decide caracteristicile contextului unei activități. Din punct de vedere teoretic, utilizatorul poate înregistra oricâte activități dorește atâta timp cât acestea se produc în contexte diferite.

Obiectivele lucrării sunt următoarele: citirea și prelucrarea valorilor a șase senzori integrați pe dispozitivul mobil, implementarea algoritmului de recunoaștere de activități, crearea unei interfețe grafice intuitive pentru folosirea aplicației, stocarea datelor provenite de la senzori într-un mod eficient și

încercarea de a minimiza consumul de energie. Aplicația este concepută să ruleze pe sistemul de operare Android. Se dorește obținerea unei aplicații care să asiste utilizatorul în viața de zi cu zi, oferindu-i o viziune în retrospectivă a activităților pe care acesta le-a executat și posibilitatea de a automatiza anumite funcții ale dispozitivului mobil, precum setarea volumului.

În capitolele ce vor urma, această soluție este descrisă și analizată în detaliu. Capitolul 2 oferă o imagine de ansamblu a domeniului de recunoaștere de activități. Capitolul 3 descrie arhitectura soluției și avantajele/dezavantajele acestei abordări. Modul în care aceasta a fost implementată este descris în detaliu în capitolul 4. Rezultatele obținute și eficiența aplicației este dezbătută în capitolul 5. În capitolul 6 sunt prezentate mai multe modalități de a îmbunătăți aplicația curentă și sunt descrise obiectivele ce au fost atinse pe parcursul acestei lucrări.

2. Învățarea din context și recunoașterea activităților umane

Detecția de activități este din ce în ce mai răspândită datorită creșterii numărului de dispozitive mobile folosite în viața de zi cu zi de un număr foarte mare de utilizatori. Majoritatea companiilor din domeniul dispozitivelor mobile folosesc din ce în ce mai mult recunoașterea de activități pentru a îmbunătăți și ușura viața utilizatorilor.

2.1 State of The art

În momentul de față, învățarea din context folosită la recunoașterea activităților zilnice ale utilizatorului, este realizată, din punct de vedere al hardware-ului, cu senzori individuali sau dispozitive mobile ce conțin o multitudine de senzori. Din punct de vedere software, sunt folosiți algoritmi de învățare automată, precum arbori de decizie, Naïve Bayes, Hidden Markov Chain, rețele neurale, etc.

Descrierea contextului folosind mai mulți senzori separați este foarte eficientă, dar este dificilă de folosit în viața de zi cu zi deoarece necesită numeroase resurse hardware, precum senzori, ce trebuie plasate în anumite regiuni ale corpului sau ale camerei. În plus, fiecare senzor suplimentar duce la creșterea timpului de prelucrare a datelor, prin urmare fiind nevoie de mai multe resurse de procesare. O variantă cu o eficiență mai scăzută, dar ce poate fi implementată fără efort sau resurse suplimentare, este folosirea dispozitivelor mobile, utilizate în fiecare zi de miliarde de utilizatori.

Dispozitivele mobile, mai ales smartphone-urile, conțin o multitudine de senzori ce permit scanarea mediului fără intervenția utilizatorului. Această abordare este limitată de resursele dispozitivului mobil, în mod special de consumul de energie al bateriei. Pe de altă parte, algoritmi de învățare automată, deși limitați de resursele de procesare ale dispozitivului mobil, pot interacționa cu ușurință cu utilizatorul pentru a obține feedback. În plus, datele stocate pe dispozitivul mobil, ce nu provin de la senzori, pot fi folosite pentru a spori acuratețea algoritmilor de recunoaștere a activităților.

Fiecare algoritm de învățare folosit în recunoașterea de activități are avantaje și dezavantaje în funcție de situație. Arborii de decizie sunt simpli de implementat și de înțeles. Petri [1] analizează avantajele și dezavantajele arborilor de decizie. Aceștia pot lucra atât cu valori nominale, cât și cu valori numerice. Pot utiliza seturi de date ce conțin erori sau au valori lipsă. Sunt mai eficienți cu cât numărul de atribute relevante este mic, dar sunt sensibili la fenomenul de overfitting și underfitting din cauza atributelor irelevante și a zgomotului.

Google [2] pune la dispoziție un API de recunoaștere a activităților pentru dezvoltatorii de aplicații Android. Acest API poate detecta opt activități diferite (IN_VEHICLE, RUNNING, etc), toate fiind legate de mișcarea utilizatorului. Sensorii sunt citiți periodic de a lungul unui interval scurt de timp. În plus, sunt folosiți doar senzorii care consumă puțină energie pentru a nu descărca bateria prea repede. Periodicitatea de citire a senzorilor poate fi modificată pentru a spori precizia, dar cu cât aceasta este mai mică cu atât se descarcă mai repede bateria. O altă metodă de conservare a nivelului de încărcare al bateriei este de a înceta citirea senzorilor în momentul în care activitatea detectată este STILL. În general, acest API este folosit pentru a executa anumite acțiuni în funcție de activitățile detectate.

2.2 Abordări existente

Problema recunoașterii activităților poate fi abordată din multe puncte de vedere. Le et al [3] descrie un sistem de recunoaștere a activităților folosind accelerometrul și giroscopul unui telefon mobil. Această abordare necesită ca telefonul mobil să fie poziționat în jurul taliei. Sistemul este capabil să recunoască 6 activități: walking, walking upstairs, walking downstairs, sitting, standing și laying. Spre deosebire de alte sisteme, Le pune accentul pe eliminarea informațiilor irelevante și redundante din setul de date, pentru a spori acuratețea sistemului cu până la 15% și pentru a obține un timp de răspuns de 3-5 ori mai rapid. Acest lucru presupune mai multe etape. În primul rând, datele sunt colectate de la senzori și zgomotul este eliminat cu ajutorul filtrelor. Apoi, datele sunt procesate, fiind selectate doar anumite caracteristici dintre acestea (valoarea medie, deviația standard, etc). Următorul pas constă în eliminarea caracteristicilor irelevante sau redundante care nu contribuie la recunoașterea de activități. Această eliminare duce la reducerea complexității computaționale, rezultând un timp de răspuns mai mic. Ultimul pas de procesare a datelor constă în selectarea exemplelor cu o relevanță mare pentru a fi folosite mai departe de către algoritmul de clasificare.

O altă utilizare a recunoașterii de activități este monitorizarea oamenilor în vârstă, descrisă de către Sebestyen et al [4]. Pe lângă senzorii telefonului mobil, această abordare folosește și senzori plasați în camere, pe uși și pe mobilier. Senzorii sunt conectați la o placă Intel Galileo ce reprezintă o platformă Arduino. Sistemul se axează pe detectarea de activități complexe, compuse din activități mai simple, de tipul walking, standing, etc. Sistemul ia în considerare și camera în care se află persoana în momentul când detectează activitatea. Spre deosebire de prima abordare, aceasta necesită o infrastructură mai complexă. Este necesar un server central care colectează datele de la toți senzorii din casă, inclusiv de la dispozitivul mobil. Partea de recunoaștere de activități este implementată pe acest server, dar pentru un număr foarte mare de utilizatori, această parte trebuie făcută pe plăcuța Intel Galileo pentru a evita congestionarea serverului. Pentru recunoașterea activităților este folosit un model Markov ascuns. Stările ascunse sunt reprezentate de activitățile ce trebuie recunoscute, alfabetul este reprezentat de observațiile făcute de senzori și dispozitivul mobil. Probabilitățile de tranziție sunt reprezentate de probabilitatea de a trece de la o activitate la alta. Probabilitățile de emisie descriu cât de probabilă este obținerea unor observații în funcție de o activitate, iar probabilitățile inițiale sunt reprezentate de probabilitatea de producere a fiecărei activități înainte de aplicarea algoritmului. Algoritmul folosit la determinarea activităților este algoritmul Viterbi. Acesta returnează calea de activități cu probabilitatea cea mai mare de apariție. Algoritmul este testat pe patru activități ce se produc în timpul dimineții, în bucătărie.

Kim et al [5] analizează recunoașterea de activități și problemele ce trebuie rezolvate pentru a obține rezultate satisfăcătoare. În primul rând, este importantă înțelegerea problemelor principale întâlnite în recunoașterea activităților umane. Persoanele pot executa mai multe activități în același timp, iar algoritmii de recunoaștere a unei singure activități într-un moment de timp nu sunt potriviți pentru a rezolva această problemă. Anumite activități pot fi intercalate. Algoritmii trebuie să poată detecta dacă o activitate este întreruptă pentru o perioadă mică de timp și apoi reluată. O altă problemă esențială este ambiguitatea anumitor activități. De exemplu: folosirea laptopului poate însemna fie muncă, fie joacă, fie altă activitate. O altă problemă apare atunci când un sistem este nevoit să recunoască activitățile mai

multor persoane în același timp, indiferent dacă ele sunt distincte sau nu. Pe lângă aceste probleme, acest articol analizează mai multe tehnici de modelare pentru activitate „eating”. Una din aceste tehnici este folosirea modelelor Markov ascunse, care este foarte eficientă pentru activitățile cu complexitate scăzută, dar nu poate recunoaște activități concurente sau intercalate. O metodă care rezolvă aceste probleme este Emerging Patterns, care folosește câte un vector pentru fiecare activitate pentru a detecta schimbări între două clase de date. O altă metodă este Naïve Bayes, ce asociază fiecărei activități o probabilitate de apariție în funcție de valorile senzorilor.

Eficiența utilizării resurselor este un element cheie atunci când recunoașterea de activități este implementată cu ajutorul unui dispozitiv mobil. Anguita et al [6] analizează recunoașterea de activități fizice folosind un algoritm de Support Vector Machine (SVM) modificat pentru a se mapa mai bine pe hardware și a reduce consumul de resurse fără a pierde acuratețe. Experimentul a constatat în recunoașterea a șase activități fizice folosind un dispozitiv mobile Samsung Galaxy S2 amplasat în jurul taliei.

Sarker et al [7] analizează tradeoff-ul dintre calitatea monitorizării contextului și eficiența folosirii energiei. Monitorizarea continuă a senzorilor are ca rezultat un consum foarte ridicat al bateriei. O altă problemă ce trebuie luată în calcul este reprezentată de fenomenele de over-sensing și under-sensing. Over-sensing duce la creșterea consumului de energie și a overhead-ului calculelor. Under-sensing duce la rezultate cu o acuratețe scăzută. În plus, atât timp cât activitatea utilizatorului nu se schimbă, frecvența de eșantionare trebuie să fie minimă pentru a evita consumul inutil de baterie și procesor. Un alt factor ce influențează consumul de energie este monitorizarea mai multor senzori în același timp. De aceea este nevoie de un mecanism care să modifice în mod dinamic frecvența de eșantionare a senzorilor. Una din aceste metode este „multiplicative increase and multiplicative decrease” (MIMD). Principalele îmbunătățiri aduse de această lucrare sunt: reducere overhead-ului prin folosirea unui algoritm k-means de clusterizare pentru a separa datele utile de cele redundante și implementarea metodei MIMD pentru modificarea frecvenței de eșantionare.

O abordare deosebită a recunoașterii de activități este analizată de Fang et al [8]. Aceștia se axează pe recunoașterea activităților legate de transportul în comun, mai exact de autobuz. Acest lucru este realizat folosind accelerometrul și giroscopul telefonului mobil pentru a detecta activitatea de urcare și de coborâre din autobuz. Spre deosebire de majoritatea metodelor prezentate anterior, telefonul mobil poate fi plasat oriunde pe corp. Pentru a recunoaște cele două activități sunt folosiți clasificatori de bază din toolkit-ul Weka. Folosirea autobuzului este împărțită în patru acțiuni: urcare în autobuz, accelerare, frânare, coborâre din autobuz. Sunt două metode pentru a detecta aceste acțiuni. Prima constă în măsurarea vitezei persoanei atunci când acțiunea de urcare sau coborâre s-a produs. În acest mod, atunci când acțiunea este urmată de accelerare, rezultă urcare, iar atunci când acțiunea este precedată de frânare, rezultă coborâre. Cea de a doua metodă constă în monitorizarea valorilor celor trei axe ale telefonului. Când una din ele trece de un prag, se verifică dacă a avut loc una din cele două acțiuni de urcare sau coborâre.

Un domeniu important în care este folosită recunoașterea de activități este cel medical. Helmy et al [9] a creat aplicația Alzimio care are ca scop supravegherea pacienților care suferă de demență, autism

sau Alzheimer. Aplicația oferă două funcții principale: detectarea de zone sigure folosind geo-fencing și recunoașterea de activități (in-car, on-foot, on-bicycle). Aplicația este concepută să ruleze pe dispozitive mobile Android de tipul low-end, fără a afecta considerabil nivelul de putere consumată. Din punct de vedere al eficienței, Alzimo oferă în condiții optime o acuratețe de 95% și o creștere a consumului de putere de numai 16%. La fel ca la majoritatea abordărilor, senzorii folosiți sunt accelerometrul, giroscopul și senzorul de locație (GPS, WIFI sau turnurile de telefonie).

Există diverse moduri de implementare a recunoașterii de activități pe dispozitivele mobile. Shoaib et al [10] analizează în detaliu o parte din acestea. Primul pas este acela de a colecta date de la senzori cu o anumită frecvență de eșantionare. Următorul pas este preprocesarea datelor colectate prin diverse moduri (de exemplu, eliminarea zgomotului). Apoi, datele sunt segmentate și se extrag anumite valori din acestea. Urmează stadiu de învățare, unde un clasificator este antrenat fie în timp ce datele sunt colectate, renunțând la stocarea acestora, fie la un moment de timp ulterior, datele fiind salvate până în acel moment. În final, clasificatorul antrenat este folosit pentru a recunoaște diversele activități executate de către utilizator. Există două moduri de procesare a datelor colectate de către senzori: pe un server sau pe dispozitivul mobil. Pentru procesarea datelor de către server este necesară o conexiune la internet, iar toate calculele costisitoare din punct de vedere computațional sunt executate pe server. Această abordare permite utilizarea de algoritmi mai performanți și reduce consumul de energie și putere de procesare de pe dispozitivul mobil. Dacă datele sunt procesate local, pe dispozitivul mobil, rezultatele sunt obținute mai rapid, nu este necesară conexiune la internet, dar consumul de energie crește și acuratețea poate scădea în funcție de algoritmi folosiți și activitățile detectate. Majoritatea aplicațiilor de acest gen folosesc arbori de decizie pentru clasificarea datelor. Platforma pe care se implementează cele mai multe lucrări în acest domeniu este Android.

Boukhechba et al [11] propune o altă abordare pentru optimizarea consumului de energie în cadrul recunoașterii de activități fizice. Acesta propune un model de clusterizare adaptivă care modifică algoritmul de clasificare în funcție de nivelul de încărcare al bateriei. Algoritmul de clasificare este bazat pe o versiune a algoritmului k-means ce folosește o fereastră de date de dimensiune variabilă în funcție de comportamentul mișcărilor utilizatorului și nivelul de energie al bateriei. Scopul lucrării este acela de a transforma datele de la GPS în activități umane. Pentru acest lucru sunt folosite trei tipuri de clustere: activități staționare, activități ce necesită mișcare și tranziții între două locuri de interes.

O altă arie de cercetare în acest domeniu este recunoașterea de activități în grup. Abkenar et al [12] descrie un mod de implementare care ia în calcul consumul de energie mare. Sistemul este construit din trei straturi. Primul strat se ocupa cu colectarea de date de la senzori. Cel de al doilea strat folosește aceste date pentru a clasifica activitățile fiecărei persoane, iar ultimul strat se folosește de activitățile individuale ale persoanelor pentru a descrie o activitate de grup.

Este interesant de analizat rolul algoritmilor de învățare automată în domeniul recunoașterii de activități. Rana et al [13] descrie o implementare ce folosește patru senzori plasați în jurul taliei, pe glezna dreaptă, pe brațul drept și pe coapsa stângă. Senzorii sunt folosiți pentru detectarea a cinci activități: sitting, standing, standing up, sitting down și walking. Doar anumite caracteristici sunt stocate pentru fiecare senzor în parte. Scopul lucrării este de a compara patru algoritmi de învățare automată: arbori de

decizie, adaboost, păduri aleatoare de arbori și support vector machine. Adaboost și pădurile aleatoare sunt algoritmi ce folosesc mai mulți clasificatori mai puțin eficienți pentru a obține un clasificator complex și precis. Support vector machine împarte spațiul de ieșire în două semiplanuri distincte, iar punctele noi sunt clasificate în funcție de semiplanul în care se află. Adaboost și pădurile aleatoare cu 10 arbori au peste 99% precizie, pe când arborele de decizie are doar puțin peste 86% precizie.

În general, recunoașterea de activități este implementată folosind senzori de mișcare. Din acest motiv, este necesară o prelucrare complexă a datelor furnizate de acești senzori. În capitolele ce urmează, este descrisă o modalitate mai simplă de recunoaștere a activităților zilnice, folosind o varietate mai mare de senzori.

3. Descrierea soluției propuse

Soluția descrisă în această lucrare, are ca scop detectarea unor activități recurente fără a se baza exclusiv pe senzorii de mișcare. În Figura 1 este descrisă structura implementării soluției. În primul rând, dispozitivul mobil citește mediul înconjurător cu ajutorul unei multitudini de senzori la o anumită frecvență. Ulterior, utilizatorul trebuie să specifice ce activități au avut loc în anumite intervale de timp din trecut. În acest mod, algoritmul este capabil să se folosească de feedback-ul utilizatorului pentru a prezice activități viitoare. În final, utilizatorul poate selecta anumite acțiuni de genul, setează telefonul pe modul silențios, pentru a fi aplicate atunci când este detectată o anumită activitate.

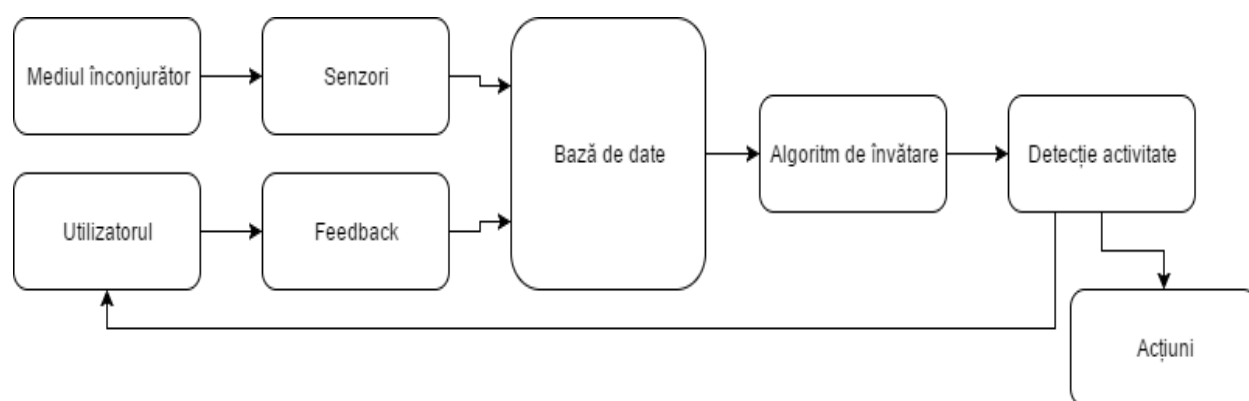


Figura 1. Diagrama de funcționare a aplicației.

3.1 Arhitectura implementării

Contextul este descris de șase senzori: lumină, proximitate, sunet, accelerație, gravitație și temperatura dispozitivului. La anumite intervale de timp, se execută o citire de context, care este salvată într-o bază de date. Ulterior, utilizatorul dă feedback aplicației, introducând o activitate și intervalul de timp în care aceasta s-a produs. Informațiile din baza de date din intervalul respectiv, sunt prelucrate, etichetate și salvate. Algoritmul se folosește de aceste date pentru a decide ce activitate se execută în funcție de citirile de context ulterioare. Algoritmul de învățare automată folosește arbori de decizie. În plus, utilizatorul poate asocia anumite acțiuni cu anumite activități. De exemplu, atunci când este detectată activitatea „Sleeping”, volumul telefonului să fie setat pe silențios.

3.2 Avantaje și dezavantaje

Un principal avantaj al acestei abordări este flexibilitatea, deoarece utilizatorul este cel care hotărăște ce context descrie fiecare activitate. În plus, activități cu același nume pot să difere în funcție de persoană. De exemplu, „Sleeping” are contexte diferite pentru o persoană care doarme în timpul zilei, când lumina și zgomotul sunt mai ridicate, față de o persoană care doarme noapte, când valorile senzorilor respectivi sunt mult mai mici. Alt avantaj este asocierea acțiunilor cu activități, rezultând o experiență îmbunătățită a utilizării dispozitivului mobil.

Pe de altă parte, această abordare nu poate detecta cu acuratețe orice fel de activitate. În plus, cei șase senzori nu garantează că contextele sunt mereu diferite, chiar dacă activitățile pe care le descriu sunt. Din acest motiv, utilizatorul este nevoit să înregistreze activități pentru care contextele sunt suficient de diferite, pentru a obține rezultate satisfăcătoare. Un alt dezavantaj, este consumul de baterie. Cu cât acuratețea este mai mare, adică citirile de context sunt mai dese, cu atât consumul de baterie crește. În implementarea descrisă mai jos, este folosit un algoritm pentru a decide intervalele la care este citit contextul, pentru a evita un consum prea mare de energie.

Aplicația este folosită pentru a înregistra activitățile zilnice ale utilizatorului și pentru a oferi anumite utilități în funcție de acestea. Înregistrare activităților este folosită pentru putea analiza în retrospectivă structura zile la sfârșitul acesteia. În Figura 2 se observă modul în care este descrisă o zi din punct de vedere al aplicației propuse.

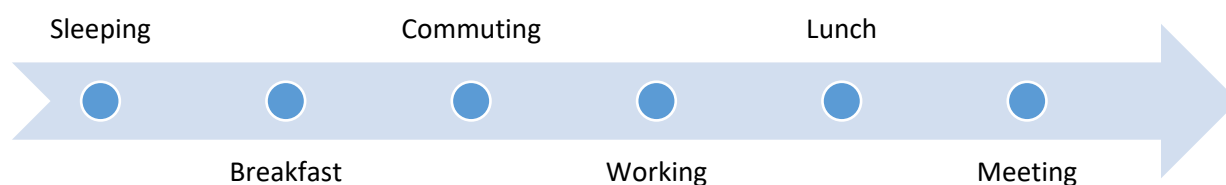


Figura 2. Exemplu de zi descrisă de către aplicație.

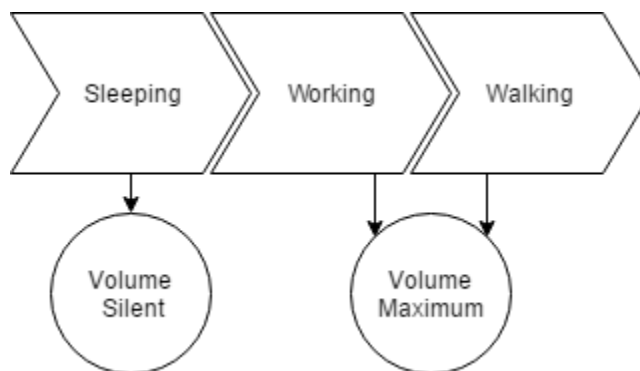


Figura 3. Exemplu de acțiuni ce pot fi oferite în funcție de activitatea curentă.

În funcție de activitatea curentă, aplicația poate modifica în timp real anumite setări ale telefonului, precum nivelul volumului soneriei. Acest lucru este exemplificat în Figura 3, în care activitatea „Sleeping” are asociat volumul silențios, iar activitățile „Working” și „Walking” au asociat volumul la nivel maxim.

4. Implementarea soluției

Soluția a fost implementată pentru dispozitivele mobile care rulează Android. Diagrama de mai jos descrie modulele principale ale aplicației și direcția fluxului de date. În subcapitolele următoare, vor fi prezentate în detaliu aceste module.

4.1 Senzorii dispozitivelor mobile Android

Majoritatea dispozitivelor mobile conțin o multitudine de senzori care sunt capabili să ofere date neprocesate cu o precizie și acuratețe foarte ridicată. Acești senzori sunt împărțiți în 3 categorii: senzori de mișcare (accelerometru, giroscop, etc), senzori de mediu (barometru, lumină, etc) și senzori de poziție (proximitate, magnetometru, etc). Senzorii pot fi hardware, construiți fizic în dispozitiv, sau software, care folosesc date de la unul sau mai mulți senzori hardware.

Tabelul 1

Senzorii folosiți în implementare

Senzor	Descriere	Utilizări
Accelerometru	Măsoară forța de accelerație pe cele 3 axe (x, y, z) în m/s^2 , incluzând forța gravitațională.	Folosit pentru a detecta modul în care se mișcă și poziția dispozitivul mobil.
Accelerație liniară	Măsoară forța de accelerație pe cele 3 axe (x, y, z) în m/s^2 , excluzând forța gravitațională.	Folosit pentru a detecta modul în care se mișcă dispozitivul mobil.
Gravitație	Măsoară forța de gravitație pe cele 3 axe (x, y, z) în m/s^2 .	Folosit pentru a detecta poziția dispozitivului mobil.
Lumină	Măsoară intensitatea luminii ambientale în luși.	Folosit pentru a detecta gradul de iluminare din camera sau locul în care se află dispozitivul mobil.
Microfon	Înregistrează sunetul folosind un obiect de tipul AudioRecord.	Folosit pentru a determina intensitatea zgomotului în decibeli.
Temperatura baterie	Măsoară temperatura bateriei în grade Celsius.	Folosit pentru a detecta cât de intens este folosit dispozitivul mobil sau dacă acesta se află la încărcat.

Senzorii de mișcare folosiți în implementare sunt: accelerometrul și senzorul de gravitație. În descrierea contextului, datele provenite de la senzorul de gravitație sunt folosite fără a fi modificate. Pe de altă parte, datele provenite de la accelerometru sunt transformate în date provenite de la senzorul virtual/software de accelerație liniară prin formula:

$$\text{accelerația liniară} = \text{ieșire accelerometru} - \text{ieșire senzor de gravitație}$$

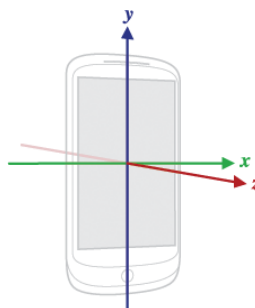


Figura 4. Sistemul de coordonate al dispozitivului mobil Android.¹

Ieșirea celor doi senzori de mișcare este reprezentată de trei valori numerice: forța aplicată pe axa x, pe axa y și pe axa z. În Figura 4 se observă orientarea celor trei axe.

4.1.1 Senzorul de gravitație

Senzorul de gravitație este folosit pentru a detecta poziția dispozitivului mobil. Dacă valoarea corespunzătoare axei z în modul este mai mare decât a celorlalte două axe atunci dispozitivul mobil este poziționat fie cu ecranul în sus, fie cu ecranul în jos. Dacă valoarea este pozitivă, atunci dispozitivul este poziționat cu ecranul în sus, altfel dacă valoarea respectivă este negativă, dispozitivul este poziționat cu ecranul în jos. Folosind același principiu, putem detecta dacă dispozitivul este poziționat vertical (în modul portret) sau orizontal (în modul peisaj). Pentru poziția verticală se verifică dacă valoarea absolută corespunzătoare axei y este mai mare decât valorile celorlalte două axe, iar pentru poziția orizontală se verifică dacă valoarea absolută a axei x este mai mare decât valorile celorlalte două axe.

Tabelul 2

Modelul de reprezentare al senzorului de gravitație

Reprezentare	Descriere	Formulă
Face Up	Valoarea forței de pe axa z este pozitivă și mai mare în modul decât cea de pe axe x și y.	$z > 0, z > x , z > y $
Face Down	Valoarea forței de pe axa z este negativă și mai mare în modul decât cea de pe axe x și y.	$z < 0, z > x , z > y $
Horizontal	Valoare absolută a forței de pe axa x este mai mare decât cea de pe axe y și z.	$ x > y , x > z $
Vertical	Valoare absolută a forței de pe axa y este mai mare decât cea de pe axe x și z.	$ y > x , y > z $

¹ https://developer.android.com/images/axis_device.png

4.1.2 Senzorul de accelerație liniară

Senzorul de accelerație liniară este utilizat pentru a detecta dacă dispozitivul este imobil. Atunci când valorile senzorului pentru cele 3 axe sunt mai mici decât 0.1, rezultă că dispozitivul nu se mișcă. Altfel, rezultă o mișcare necunoscută. Scopul pentru care este folosit acest senzor nu este acela de a recunoaște anumite modele de mișcări.

Tabelul 3

Modelul de reprezentare al senzorilor de mișcare

Reprezentare	Descriere	Formulă
Still	Valoarea forței fiecărei din cele 3 axe (x, y, z) este mai mică sau egală decât 0.1.	$x \leq 0.1, y \leq 0.1, z \leq 0.1$
Moving	Valoarea forței fiecărei din cele 3 axe (x, y, z) este strict mai mare decât 0.1.	$x > 0.1, y > 0.1, z > 0.1$

4.1.3 Senzorul de lumină

Senzorul de lumină indică intensitatea luminii măsurată în lăcși. Pentru a evita o complexitate prea mare rezultată din numărul mare de contexte diferite, valorile senzorului de lumină sunt împărțite în intervale. Cu cât intervalele sunt mai mici și mai dese, cu atât acuratețea algoritmului este mai mare. Tabelul 4 descrie un mod de împărțire a intervalelor eficient și relevant.

Tabelul 4

Modelul de reprezentare al senzorului de lumină

Intervalul de valori	Descriere interval
[0,1)	Noaptea, fără lumină artificială sau naturală.
[1,10)	Cameră cu iluminare artificială scăzută.
[10,100)	Cameră cu iluminare artificială normală.
[100, 1000)	Cameră cu iluminare naturală puternică.
[1000, 5000)	Afară, pe timp de zi, fără expunere directă la soare.
[5000, 100000)	Afară, pe timp de zi, expus la soare direct.

4.1.4 Senzorul de sunet

Senzorul de sunet sau microfonul este folosit pentru a identifica nivelul de zgomot din jurul dispozitivului mobil. Fiecare eșantion dintr-o mostră de sunet este reprezentat de o valoare întreagă între

0 și $2^{15} - 1$. Fiecare citire de context a microfonului conține mai mult eșantioane. Pentru a obține o valoare relevantă a nivelului de zgomot, este calculată valoarea medie pătratică (root-mean-square, RMS) pentru fiecare citire de context:

$$RMS = \sqrt{\frac{\sum_1^n x^2}{n}}, x = \text{valoarea unui eșantion}$$

Valoarea RMS-ului este transformată în decibeli, înainte de a fi salvată și prelucrată de către algoritmul de învățare:

$$RMS_{db} = 20 * \log_{10} RMS$$

La fel ca la senzorul de lumină, pentru a evita o complexitate prea mare, valorile finale furnizate de către microfon sunt transformate în intervale înainte de a fi prelucrate de către algoritm. Prin urmare, există 10 intervale pentru valorile RMS-ului în decibeli:

$$INTERVAL_i = [i, i + 10), \forall i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

4.1.5 Senzorul de proximitate

Senzorul de proximitate este singurul senzor de poziție folosit. Acesta returnează o valoare care reprezintă distanța până la cel mai apropiat obiect. Pentru anumite dispozitive, senzorul de proximitate furnizează o ieșire binară: 0 dacă se află un obiect la o distanță mică de senzor și 9 altfel. Dispozitivul pe care s-a implementat aceasta lucrare conține senzor de proximitate cu ieșire binară. Valoarea 0 este reprezentată de atributul „Far”, iar valoarea 9 de atributul „Near”.

4.1.6 Senzorul bateriei

Senzorul care monitorizează bateria, este folosit pentru a obține temperatura acesteia în grade Celsius. Valorile acestui senzor sunt grupate în 3 intervale: [0, 30) reprezintă temperatura normală a telefonului atunci când nu este folosit, [30, 35) atunci când telefonul este folosit intens și [35, 100) atunci când telefonul este supus la temperaturi foarte ridicate.

4.1.7 Citirea senzorilor în Android

În acest subcapitol, va fi prezentate modul în care pot fi accesați senzorii unui dispozitiv mobil ce rulează sistemul de operare Android. Toate valorile, tuturor senzorilor prezentați mai sus, inclusiv intervalele, au fost ajustate pentru telefonul cu care a fost testată aplicația pe parcursul implementării.

Senzorii de accelerație liniară, gravitație, lumină și proximitate pot fi interogați prin intermediul interfeței *SensorEventListener*. Aceasta pune la dispoziție două metode de callback: una apelată atunci când se schimbă acuratețea senzorului și cealaltă atunci când senzorul înregistrează o nouă valoare. În această implementare vom folosi cea de a doua metodă: *onSensorChanged*. În plus, biblioteca Android pune la dispoziție și o metodă de setare a intervalului la care vor fi interogați senzorii, deși acesta nu este

garantată. Intervalul ales este de 200,000 de microsecunde, reprezentat de constanta `SENSOR_DELAY_NORMAL`.

Pentru microfon este folosit un obiect de tipul *AudioRecord* cu următoarele proprietăți: rata de eșantionare de 44100 Hz, tipul de canal mono și codare PCM pe 16 biți.

Pentru baterie este nevoie de un obiect de tipul *BroadcastReceiver* setat să primească evenimente ce conțin starea bateriei, inclusiv temperatura acesteia.

Toate aceste evenimente sunt controlate de un fir de execuție care citește valorile lor la un interval de timp variabil. Inițial acest interval este de 30 secunde, dar crește până la 5 minute dacă activitatea curentă detectată de către algoritm nu se schimbă. Lungimea intervalului este resetată la 30 de secunde atunci când este detectată o nouă activitate.

4.2 Setul de date

În urma testelor efectuate pentru a evalua aplicația, a fost strâns un set de date folosit la definirea activităților. Acest set de date este alcătuit din două părți: contexte neetichetate și etichetate. Ambele context sunt compuse din șase valori. Diferența între acestea este că contextele neetichetate sunt descrise prin valori numerice, iar cele etichetate sunt descrise prin intervale.

Setul de date compus din contexte neetichetate reprezintă descrierea mediului înconjurător la intervale de 45 de secunde. Acesta are ca atribute intensitatea luminii, proximitatea, nivelul zgomotului, poziția dispozitivului mobil, mobilitatea acestuia și temperatura bateriei. Acest set de date este foarte numeros, având aproximativ 2000 de intrări pe zi. Din acest motiv, în memoria telefonului sunt stocate doar contextele din ziua curentă. Analizând acest set se pot observa toate schimbările minore de context. Valorile a două intrări consecutive din acest set de date diferă într-o proporție foarte mică, de obicei.

Un exemplu din acest set de date este: intensitate lumina 1.2 lx, proximitate aproape, zgomot de 50.4 dB, poziție verticală, în mișcare, temperatura baterie 35.2 °C. Pe baza acestuia, se pot face numeroase deducții. Din poziția verticală rezultă că dispozitivul mobil se află în buzunar, în geantă sau alt loc asemănător. Observație este susținută și de către valoare foarte scăzută a senzorului de lumină și de către senzorul de proximitate. Zgomotul puternic de peste 50 dB și temperatura ridicată a bateriei duce la presupunerea că utilizatorul se află cel mai probabil afară. Utilizatorul este în mișcare, concluzie susținută de accelerometru. Din toate aceste observații putem deduce că utilizatorul execută activitatea de plimbare afară.

Pentru a nu lucra cu 2000 de valori pe zi, există și varianta etichetată a setului de date. Aceasta este salvată în totalitate în memoria dispozitivului mobil. În acest set de date se află tot valorile contextelor, dar acestea sunt agregate în intervale alese în funcție de anumiți factori. În plus, acestea sunt etichetate de către utilizator, reprezentând valorile de referință a contextelor activităților. Folosind acest set de date, se pot deduce activitățile din trecut ale utilizatorului. Fiecare intrare din setul neetichetat poate fi căutată, din punct de vedere al valorilor contextului, în setul etichetat și poate fi dedusă activitatea ce a avut loc. Activitățile ce urmează să fie executate în viitor sunt determinate de către algoritmul de învățare.

Majoritatea contextelor aflate în setul de date curent descriu trei activități: „Walking”, „Working” și „Sleeping”. Numărul intrărilor de descriu activitatea „Sleeping” dintr-o zi este de aproximativ o treime din dimensiunea setului de date, adică aproximativ 600 pe zi. După ce au fost etichetate cu „Sleeping” numărul acestora scade la cel mult 10. „Working” reprezintă cea mai mare parte a intrărilor, deoarece descrie contextul atunci când utilizatorul se află în cameră și nu folosește dispozitivul mobil. Având în vedere ca acest set de date a fost strâns în timpul unei perioade intense din punct de vedere al volumului de muncă efectuat, aproximativ 12 ore au fost petrecute în această stare (1000 de intrări pe zi). Restul intrărilor reprezintă fie activitate „Walking”, fie „None” ce reprezintă contexte ce nu au fost încă etichetate de utilizator. Cu cât utilizatorul efectuează mai multe activități diferite din punct de vedere al contextului, cu atât mai divers o să fie setul de date.

4.3 Structura aplicației Android

Aplicația este formată din următoarele elemente majore: interfața, serviciile, baza de date și algoritmul de învățare automată. În plus, sunt folosite fire de execuție, ascultători pentru senzori, receptori de broadcast-uri și alte funcționalități mai puțin importante oferite de biblioteca Android.

4.3.1 Interfața aplicației

O activitate într-o aplicație Android, reprezintă o fereastră ce conține elemente de interfață cu care poate interacționa utilizatorul. Un fragment este un tip de activitate folosit pentru a crea interfețe complexe și pentru a modulariza codul.

În Figura 8 se observă că aplicația este compusă dintr-o „activitate Android” principală, patru fragmente principale și trei fragmente folosite pentru afișarea bazei de date. Activitatea principală conține o bară de navigare ancorată în partea de jos a ecranului. Fiecare element din bara de navigare este legat de unul din cele patru fragmente principale. Pe lângă elementele de interfață, activitatea principală se ocupă cu pornirea tuturor funcțiilor aplicației: servicii, fire de execuție, receptori de broadcast, etc.

Fragmentul „Home” este afișat atunci când aplicația este deschisă. Acesta poate fi observat în Figura 5. Fragmentul conține un container care afișează activitatea curentă a utilizatorului. Sub acesta, se află un meniu dropdown ce conține toate activitățile utilizatorului cunoscute de aplicație și un buton „Change” folosit pentru a schimba activitate curentă detectată de algoritm. Ultimul element din fragment, este un container de tip *ListView*, folosit pentru a afișa valorile senzorilor în timp real.

Următorul fragment, descris în Figura 6, este denumit „Activities” și conține modalitatea de a trimite feedback algoritmului de recunoaștere a activității umane. În partea de sus sunt trei containere de tip text, unde utilizatorul poate introduce un nume de activitate și intervalul orar în care aceasta a avut loc în format de 24 de ore. A doua jumătate a fragmentului este ocupată de un meniu dropdown folosit pentru a vizualiza logica din spatele alegerilor făcute de algoritmul de învățare.

Al treilea fragment este cel folosit pentru afișarea bazei de date. Acesta la rândul lui, conține trei fragmente, unul pentru fiecare tabel din baza de date. Fiecare din aceste fragmente conține un buton

pentru ștergerea tabelului respectiv și un container de tip listă pentru afișarea intrărilor din tabel. Structura bazei de date va fi detaliat în următorul capitol.

Ultimul fragment este reprezentat de setările aplicației. Cea mai importantă dintre ele, este opțiunea de a porni și opri monitorizarea contextului și recunoașterea de activități. Această opțiune este utilă pentru a nu consuma resursele telefonului atunci când nu este necesară recunoașterea activităților.

Pentru a ușura interacțiunea cu utilizatorul, aplicație conține și o notificare permanentă care afișează activitatea curentă și un buton pentru a putea trimite feedback algoritmului prin schimbarea activității curente. Activitatea afișată de către notificare este actualizată odată pe minut. Interfața notificării poate fi observată în Figura 7.

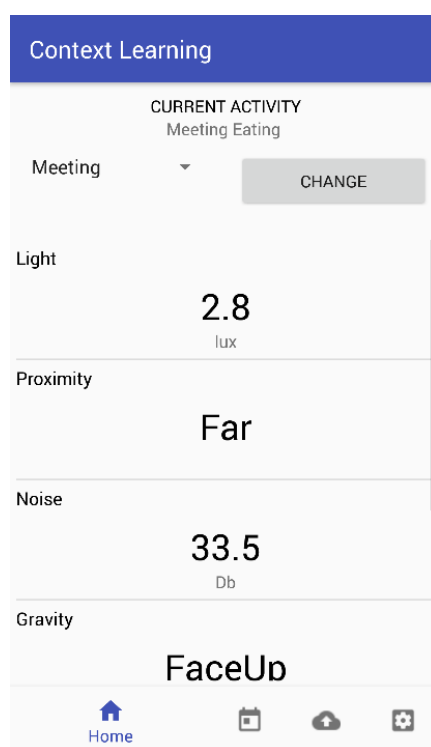


Figura 5. Fragmentul „Home”.

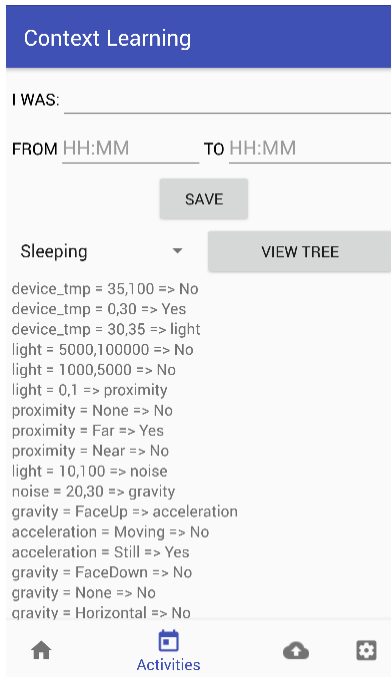


Figura 6. Fragmentul „Activities”, în care este afișat arborele de decizie al activității „Sleeping”.

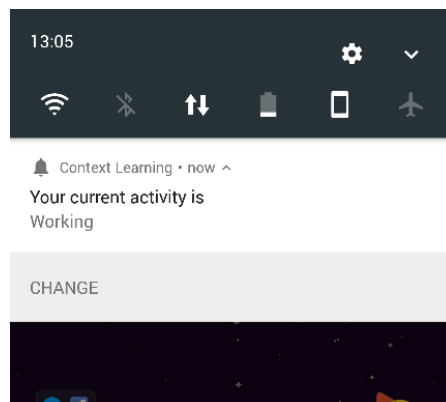


Figura 7. Notificarea ce afișează activitatea curentă: „Working”.

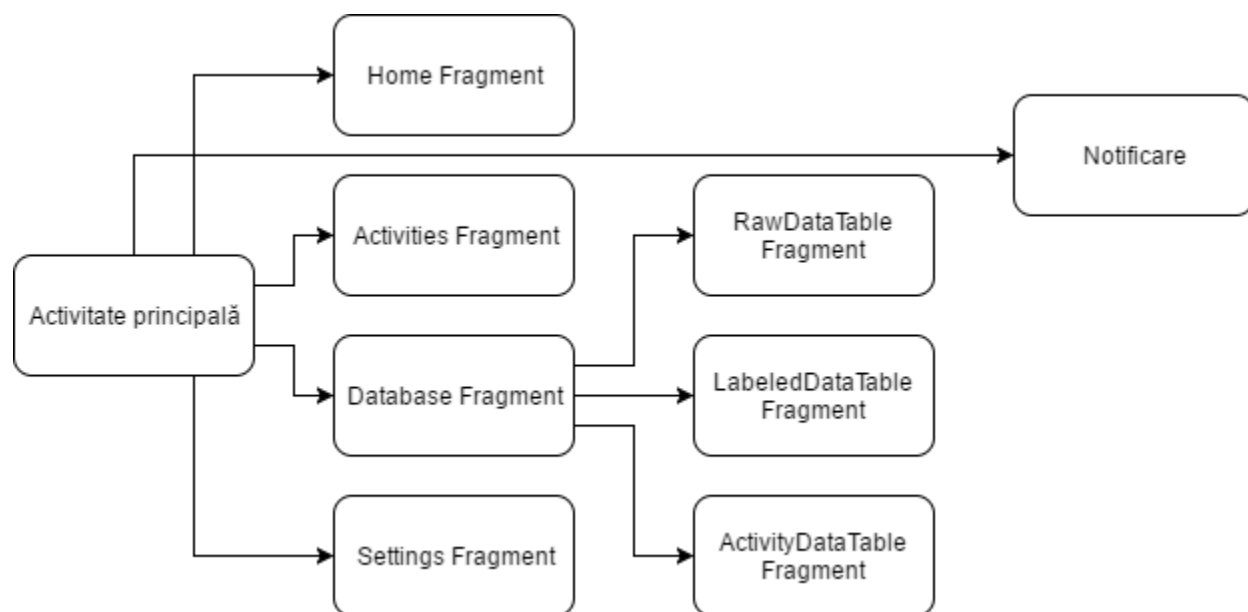


Figura 8. Structura interfeței aplicației.

4.3.2 Structura bazei de date

Aplicațiile de pe dispozitivele mobile Android au la dispoziție baze de date de tipul SQLite. Bibliotecile Android oferă numeroase clase și interfețe pentru a lucra cu baze de date SQL. Structura bazei de date folosită în implementare este formată din trei tabele: RawDataTable, LabeledDataTable și ActivitiesDataTable.

Tabelul RawDataTable conține valorile senzorilor prelucrate și momentul de timp la care acestea au fost înregistrate. Prima coloană a tabelului conține timpul în milisecunde la care au fost interogați senzorii. Pentru a simplifica structura tabelului, valori sunt concatenate într-un obiect de tipul *String* și sunt salvate în cea de a doua coloană. Structura celor două coloane este exemplificată în Figura 9. Structura obiectului *String* este următoarea:

```

„INTENSITATEA_LUMINII      PROXIMITATEA      NIVELUL_ZGOMOTULUI
ACCELERAȚIA_LINIARĂ      GRAVITAȚIA      TEMPERATURA_BATERIEI”

```

Tabelul LabeledDataTable conține etichetele date de către utilizator valorilor senzorilor dintr-un interval de timp anume. Înainte de a fi introduse în acest tabel, valorile senzorilor de lumină, sunet și temperatură au fost transformate în intervale. Acest lucru este se poate observa în Figura 10. Prin urmare obiectul *String* ce reprezintă valorile senzorilor în acest tabel are structura asemănătoare cu cel din tabelul anterior, valorile numerice ale senzorilor fiind înlocuite cu intervale.

Tabelul ActivitiesDataTable conține activitatea pe care o execută utilizatorul în funcție de momentul de timp, descrisă în Figura 11. Utilizatorul poate executa mai multe activități simultan. În funcție de datele din acest tabel se decide ce acțiuni trebuie luate pentru a asista utilizatorul în activitatea curentă.

Time: 13:9:34
Sensors: 102.2 Far 38.7 FaceUp Still 30.7

Figura 9. Captură a unei intrări din tabelul RawDataTable.

Time: 13:9:4
Label: Working

Figura 10. Captură a unei intrări din tabelul LabeledDataTable.

Sensors: 100,1000 Far 30,40 FaceUp Still 0,30
Label: Working

Figura 11. Captură a unei intrări din tabelul ActivitiesDataTable.

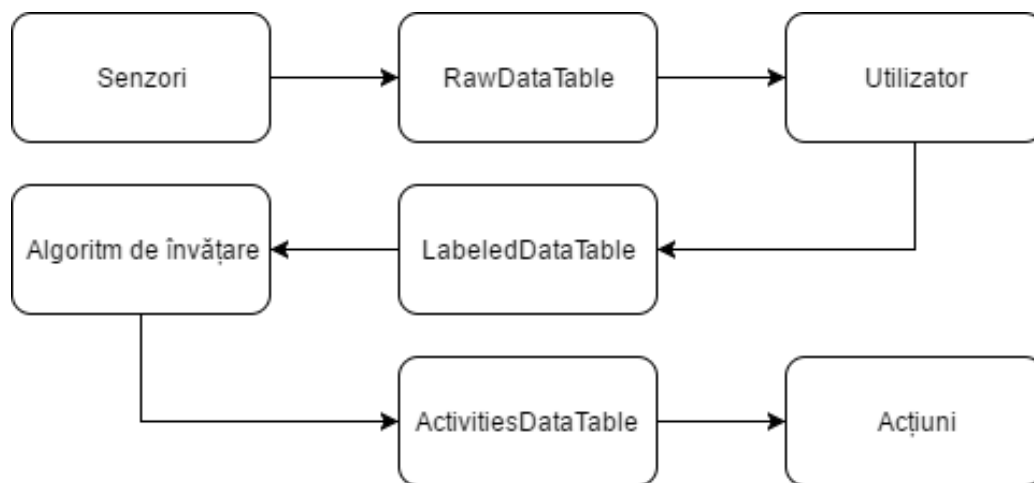


Figura 12. Fluxul de informații al bazei de date.

În Figura 12 se observă că fluxul de informații ce interacționează cu baza de date este următorul: senzorii depozitează informațiile în tabelul RawDataTable. Apoi, utilizatorul etichetează anumite intrări din RawDataTable și le salvează în LabeledDataTable. În final, algoritmul de recunoaștere de activități folosește ca set de antrenare, datele din LabeledDataTable. Ieșirile algoritmului, care reprezintă activitatea curentă, sunt stocate în ActivitiesDataTable, de unde pot fi accesate pentru a îmbunătăți experiența utilizatorului.

4.3.3 Structura internă a aplicației

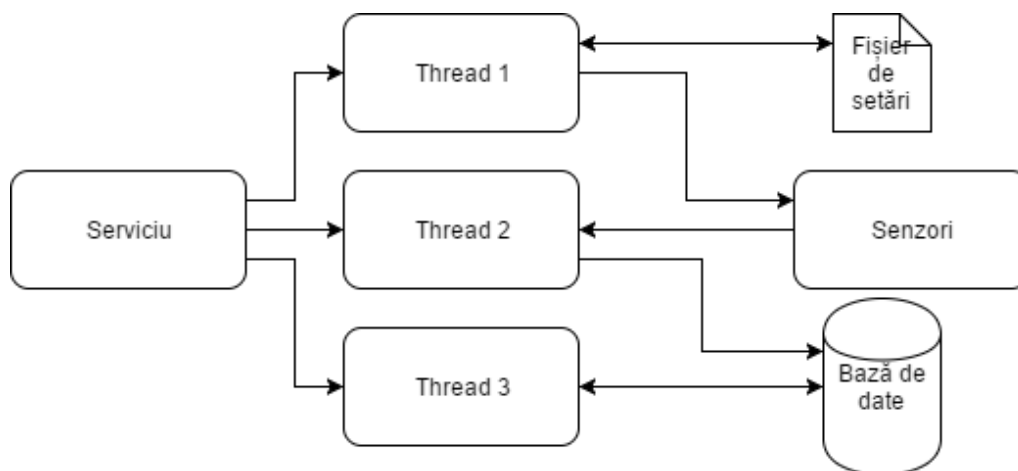


Figura 13. Structura internă a aplicației.

Un serviciul Android, reprezintă o componentă a unei aplicații ce este specializată pe executarea de sarcini în fundal. Aplicația folosește un serviciu care este mereu pornit, chiar dacă aplicația a fost închisă. Acesta pornește trei fire de execuție care au ca scop interacțiunea cu senzorii și cu baza de date. Fluxul de date între aceste elemente este descris în Figura 13.

Primul fir de execuție se ocupă de pornirea și oprirea monitorizării senzorilor. Utilizatorul poate porni și opri monitorizarea din fragmentul de setări. Starea monitorizării este salvată într-un tip fișier special ce memorează intrări de tipul cheie-valoare. La un interval de timp prestabilit, firul de execuție interoghează acest fișier și pornește sau oprește monitorizarea, în funcție de valoarea citită. Acest lucru este necesar, deoarece serviciul continuă să ruleze și când aplicația este oprită. Fără opțiunea de a opri monitorizarea senzorilor, rata de descărcare a bateriei ar crește semnificativ și procesorul ar fi mereu folosit de către aplicație.

Cel de al doilea fir de execuție conține o metodă de callback în care sunt primite valorile senzorilor și în care sunt prelucrate pentru a putea fi folosite ulterior de către algoritmul de învățare. În plus, la un interval de timp datele receptate de către metoda de callback împreună cu timpul curent în milisecunde, sunt adăugate în tabelul RawDataTable din baza de date. Scopul acestui fir de execuție este de a descrie fiecare interval de timp printr-un context format din valorile senzorilor.

Ultimul fir de execuție are ca scop recunoașterea activității curente. În primul rând, acesta actualizează algoritmul de învățare ori de câte ori apare o nouă intrare în tabelul LabeledDataTabel din baza de date. În al doilea rând, la fiecare interval de timp, interoghează algoritmul de învățare pentru a afla activitatea curentă și o stochează în tabelul ActivitesDataTable din baza de date.

4.4 Algoritmul de recunoaștere de activități

Structura algoritmului de recunoaștere de activități este formată din mai mulți arbori de decizie construiți pe baza algoritmului ID3. Un avantaj al algoritmului ID3 este crearea de arbori de decizie de dimensiune minimă. Pentru a îndeplini această condiție este nevoie de o modalitate de selecție a atributelor. ID3 se folosește de teoria informației pentru a decide care atribut este cel mai potrivit să devină nod la un moment de timp. Atributele sunt reprezentate de senzori.

4.4.1 Descrierea algoritmului ID3

În primul rând se calculează cantitatea de informație oferită de fiecare atribut. Această valoare este denumită entropie și are următoarea formulă matematică:

$$Entropy(s) = \sum_c (-p(I) * \log_2 p(I))$$

S = setul de date de antrenare

$p(I)$ = proporția de exemple din S care aparțin clasei I

c = clasele din care fac parte exemplele din setul de antrenare

În al doilea rând este necesar calculul câștigului informațional al unui atribut într-un set de date de antrenare.

$$Gain(S, A) = Entropy(S) - \sum_v \left(\frac{|S_v|}{|S|} * Entropy(S_v) \right)$$

V = valorile atributului A

S_v = setul de exemple din S în care valoarea atributului A este v

$|S_v|$ = cardinalul mulțimii S_v

$|S|$ = cardinalul mulțimii S

Pașii de construire al arborelui de decizie sunt următorii:

1. Dacă toate exemplele din setul de date aparțin aceleiași clase atunci se adaugă un nod frunză cu valoarea clasei.
2. Altfel, se alege atributul cu cel mai mare câștig informațional și se adaugă un nod de decizie cu valoarea atributului.
3. Setul de date este partiționat în n subseturi, n fiind numărul de valori distincte ale atributului selectat la pasul anterior. Fiecare subset conține doar exemple pentru care valoarea atributului este aceeași. Fiecare exemplu aparține unui singur subset.
4. Pentru fiecare subset se apelează recursiv algoritmul începând cu pasul 1.

4.4.2 Integrarea algoritmului ID3 în aplicație

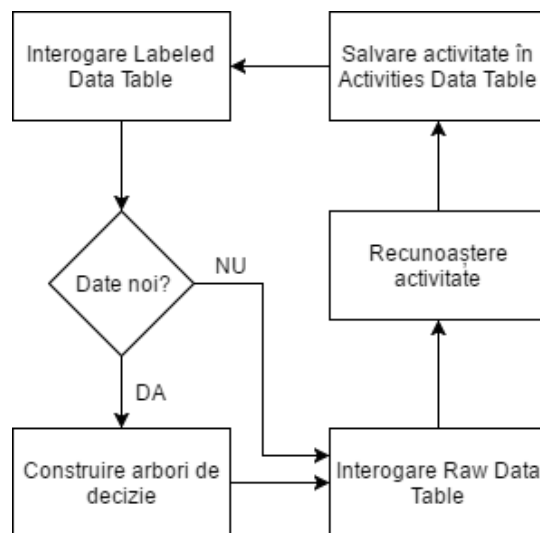


Figura 14. Modul de funcționare al algoritmului de recunoaștere de activități.

Există două abordări valabile pentru implementarea arborilor de decizie în soluția curentă. Prima constă în crearea unui singur arbore de decizie care are ca frunze activitățile ce trebuie recunoscute. Cea de-a doua abordare constă în construirea unui arbore de decizie pentru fiecare activitate în parte cu clasele rezultate „Yes” și „No”. Pentru implementarea actuală a soluției este folosită cea de-a doua variantă. Deși rezultă mai mulți arbori de decizie, aceștia au o dimensiune redusă și pot fi parcurși mai rapid. În plus, este posibilă detectarea fiecărei activități independent de celelalte, ceea ce duce la două mari avantaje: detecția mai multor activități simultan și detecția unei activități anume într-un timp mai scurt, deoarece este parcurs un arbore de decizie de dimensiune mică, cu două clase, în loc de un arbore de decizie cu dimensiune mai mare și cu o clasă pentru fiecare activitate.

În Figura 14 sunt descriși pași necesari construirii arborilor de decizie și folosirii acestora pentru recunoașterea de activități. Arborele de decizie al fiecărei activități este construit folosind setul de date din LabeledDataTable în întregime. Exemplele care sunt etichetate cu activitatea pentru care este construit arborele sunt considerate pozitive, iar toate celelalte exemple sunt considerate negative. Pasul următor constă în aplicarea algoritmului ID3 descris mai sus. Arborii de decizie sunt reconstruiți în momentul în care apare o nouă intrare în tabelul LabeledDataTable. În Figura 15 se observă structura de ansamblu a algoritmului.

După ce arborii de decizie au fost construiți este nevoie de o metodă de folosire a acestora pentru a detecta ce activitate se produce la un anumit moment de timp. Se extrage ultima intrare din baza de date din RawDataTable, apoi fiecare arbore de decizie este parcurs și în funcție de intrare se decide dacă activitatea are loc sau nu. Dacă mai mulți arbori de decizie clasifică intrarea ca aparținând activității descrise de ei, atunci sunt detectate mai multe activități concurente. Dacă nici un arbore de decizie nu clasifică intrarea ca aparținând acțiunii respective atunci rezultă activitate „None”. Prin urmare rezultă că există anumite contexte pentru care utilizatorului nu a atașat nici o activitate.

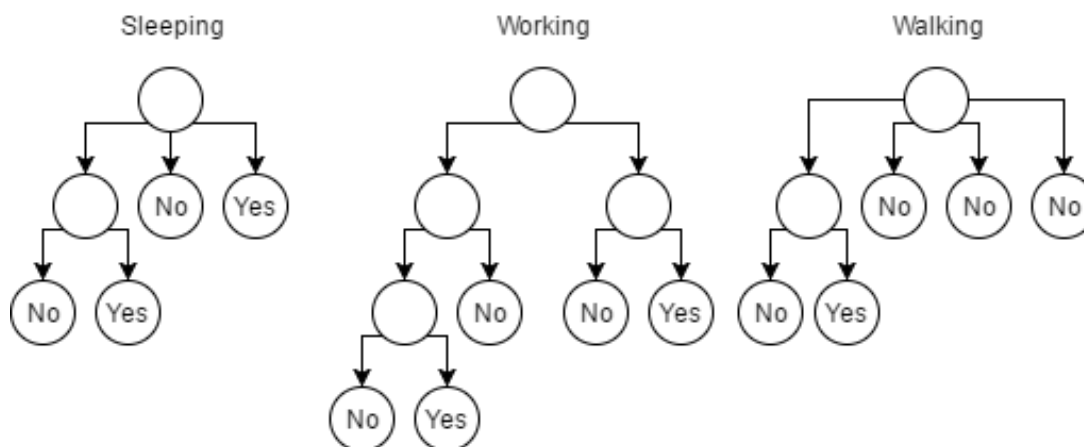


Figura 15. Structura algoritmului. Fiecare activitate împreună cu arborele de decizie ce o descrie.

4.4.3 Procesul de recunoaștere de activități

În funcție de caracteristicile fiecărei activități, arborii de decizie pot varia de la arbori cu un singur atribut la arbori de conțin toate atributele. De exemplu: fie două activități „Sleeping” și „Working”. Se consideră ca activitatea „Sleeping” se produce în următoarele condiții: lumină foarte redus între 0 și 1 lucși, proximitate „Far”, zgomot redus în jur de 25 dB, poziție „FaceUp”, fără mișcare și temperatura bateriei normală în jur de 30 °C. Activitate „Working” are aceleași caracteristici cu excepția luminii care este în jur de 100 lx și zgomotul care are valoarea în jur de 35 dB. Dacă acestea sunt singurele activități înregistrate de către utilizator, atunci, deoarece au cel puțin un atribut distinct (lumina și sunetul), arborii de decizie ce le descriu o să aibă un singur nivel cu un singur atribut, fie lumină, fie sunet. Să presupunem că este ales atributul lumină. Acest lucru simplifică detecția de activități deoarece singurul senzor necesar este cel de lumină. Arborele rezultat este descris în Figura 16.

Considerăm că la exemplul de mai sus mai adăugăm activitatea „Cleaning” care are aceleași caracteristici cu „Working”, în afară de valoarea zgomotului care este în jur de 60 dB. Arborii de decizie sunt recalculați, iar adâncimea lor crește. În Figura 17 se observă creșterea în adâncime a arborilor de decizie atunci când este adăugată o nouă activitate. Nu toți arborii de decizie sunt afectați de adăugarea unei noi activități.

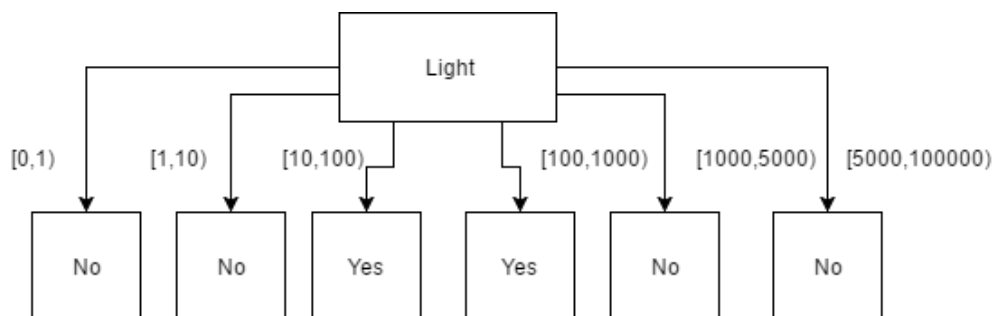


Figura 16. Arborele de decizie al activității „Working”, pentru exemplul descris mai sus.

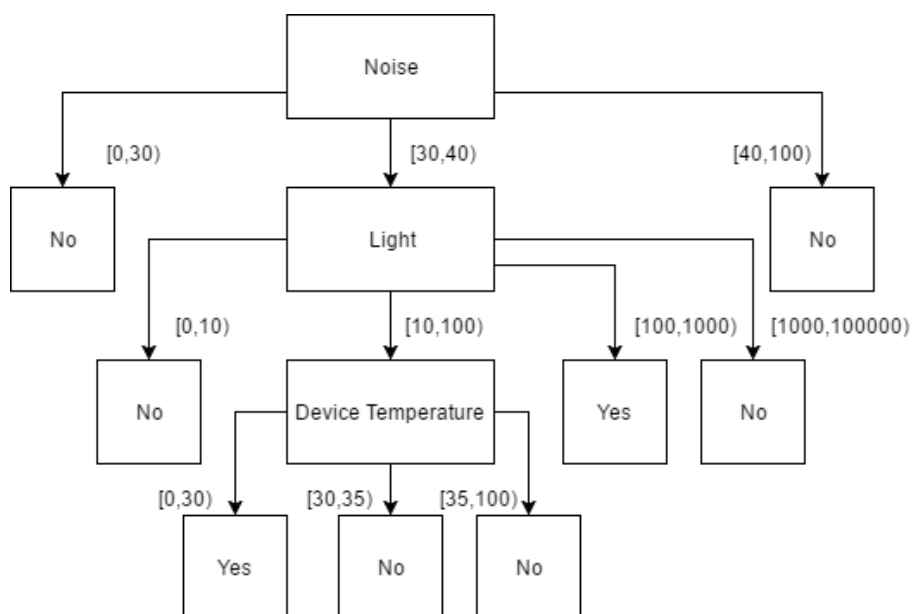


Figura 17. Arborele de decizie al activității „Working” după adăugarea activității „Cleaning”.

4.4.4 Implementarea algoritmului în mediul de dezvoltare Android

Atributele reprezentate de senzorii de lumină, sunet și temperatura bateriei au un număr de valori posibile foarte mare. Deși nu sunt valori continue, datorită limitărilor de reprezentare a numerelor într-o memorie de calculator, pentru algoritmul de arbori de decizie pot fi considerate continue. Din acest motiv, este nevoie de o modalitate de a descrie arborele de decizie fără a crea un nod copil pentru fiecare valoare a acestor atribute/senzori. În schimb, aceste valori sunt transformate într-un număr mic de intervale descris anterior, în secțiunea 4.1. Aceste intervale pot fi modificate în funcție de modelul senzoriilor și de activitățile utilizatorului. Pentru implementarea curent vor fi folosite cele descrise în secțiune menționată anterior.

Stocarea unui arbore de decizie în Java este realizată cu ajutorul clasei *DecisionTree*, care conține un *String* cu numele atributului nodului și un *HashMap* cu chei reprezentate de valorile atributului și valori reprezentate de copii nodului curent, noduri de tipul *DecisionTree*. Această structură permite o parcurgere ușoară atunci când se verifică din ce clasă face parte un exemplu.

Pentru crearea și parcurgerea arborelui de decizie este folosită clasa *DecisionTreeHelper*. Aceasta conține toate metodele necesare construirii unui arbore de decizie pentru o anumită activitate. În această clasă sunt implementate algoritmul ID3, verificarea unui exemplu prin parcurgerea arborelui de decizie și afișarea arborelui de decizie.

5. Evaluarea implementării

În funcție de activitățile create de către utilizator, precizia algoritmului de recunoaștere a acestora diferă. O analiză din punct de vedere teoretic arată importanța duratei și continuității activităților. Pentru a putea evalua corect aplicația este nevoie și de analiza unui exemplu concret pentru a vedea în ce mod se comportă în practică și cum sunt afectate resursele dispozitivului mobil. În secțiunile din acest capitol vom analiza toate aceste aspecte pentru a decide dacă aplicația poate fi folosită în starea curentă.

5.1 Evaluarea unui scenariu teoretic

Considerăm scenariul în care utilizatorul are definite trei activități: „Sleeping”, „Working” și „Walking”. Activitatea „Sleeping” este descrisă de așezarea dispozitivului mobil pe masă, cu fața în sus, în camera în care utilizatorul doarme, în care intensitatea luminii este extrem de scăzută și zgomotul din jur este redus. Dispozitivul nu este mișcat pe parcursul activității, iar temperatura bateriei este normală. Activitatea „Working” este descrisă de așezarea dispozitivului mobil pe masă, cu fața în sus lângă utilizatorul care lucrează. Spre deosebire de „Sleeping”, intensitatea luminii și a zgomotului din încăperea este mai ridicată. Atunci când utilizatorul execută activitatea „Walking”, contextul este complet diferit. Dispozitivul mobil este plasat în buzunarul de la pantaloni, în poziție verticală și se află în mișcare. Cele trei activități au cel puțin un atribut ale căror valori nu se intersectează. Acest lucru elimină ambiguitate și detecția mai multor activități în același timp. În plus, atunci când contextul nu se potrivește cu nici una dintre cele trei activități rezultă activitatea implicită „None”.

Tabelul 5

Caracteristicile celor trei activități

Activitate	Senzor de lumină (lx)	Senzor de proximitate	Senzor de sunet (dB)	Senzor de gravitație	Senzor de accelerație liniară	Senzor de temperatură a bateriei (°C)
Sleeping	[0, 1)	Far	[20, 30)	Face Up	Still	[30, 35)
Working	[10,1000)	Far	[20, 50)	Face Up	Still	[30, 35)
Walking	[0, 1)	Near	[20, 60)	Vertical	Moving	[30, 35)

În condiții ideale, presupunem că dacă în timpul unei activități de durată mai mare de 30 de minute apar activități de durată mai mică de 5 minute, acestea sunt considerate irelevante și sunt înlocuite cu activitatea de durată mare. În acest mod se garantează recunoașterea activităților principale ale utilizatorului, simplificând descrierea unei zile. Contextul ideal al unei zile obișnuite, în care în timpul activităților detectate nu se produce nici o activitate nouă cu durată mai mare de 5 min, este descris folosind cele trei activități în Figura 18.

Activitate reală:	Sleeping	Morning routine	Walking	Working	Walking	Evening routine
Time:	23:00 - 07:00	07:00 - 08:30	08:30 - 09:00	09:00 - 18:00	18:00 - 18:30	18:30 - 23:00
Activitate detectată:	Sleeping	None	Walking	Working	Walking	None

Figura 18. O zi descrisă de aplicație folosind setările ideale descrise mai sus.

Considerând scenariu descris mai sus, rezultă următoarele:

$$p = \frac{t_r}{t_t} * 100 = \frac{18}{24} * 100 = 75 \%$$

p = procentul descris corect de către aplicație

t_r = timpul în care se exactă activitățile detectate corect

t_t = timpul total din cadrul unei zile

Acest procent poate fi îmbunătățit dacă activitățile „Morning routine” și „Evening routine” pot fi descrise de una sau mai multe activități ale căror context diferă prin cel puțin un atribut față de toate celelalte activități. În această situație ideală, procentul de detecție corectă al activităților poate ajunge aproape de 100 %. Într-un scenariu real aceste condiții nu pot fi îndeplinite simultan.

Pe de altă parte, într-un scenariu real nu sunt asigurate intervale continue de dimensiuni mari în care se execută aceeași acțiune. De exemplu în timpul activității „Sleeping”, pot apărea schimbări neașteptate de context, rezultând un interval fragmentat de activități de tipul „None”. În loc ca activitatea „Sleeping” să aibă loc în tot intervalul 23:00 – 07:00, aceasta are în anumite bucăți ale intervalului respectiv. Acest lucru duce la o acuratețe mai mică a detecției de activități. Să presupunem că în intervalul 23:00 – 07:00, odată pe oră, timp de 10 minute, contextul se schimbă brusc și rezultă activitatea „None”, deși utilizatorul execută activitatea „Sleeping” în realitate. Acest lucru duce la următoarea schimbare a eficienței aplicației:

$$t_r = t_s + t_{wo} + t_{wa} = \left(8 - \frac{8}{6}\right) + 9 + 1 = 16.6$$

$$p = \frac{t_r}{t_t} * 100 = \frac{16.6}{24} * 100 = 70 \%$$

t_s = timpul în care are loc activitate „Sleeping”

t_{wo} = timpul în care are loc activitatea „Working”

t_{wa} = timpul în care are loc activitatea „Walking”

Se observă o scădere cu 5 % a procentului de detecție corectă de activități în cazul în care este afectată doar activitatea „Sleeping”. De aici rezultă că cu cât contextul este mai instabil cu atât scade precizia recunoașterii de activități.

O altă problemă ce poate afecta precizia algoritmului este introducerea unei activități noi care are un context similar cu o activitate existentă. De exemplu să presupunem că activitatea „Working” se produce și atunci când valoarea intensității luminii este în intervalul [0, 1). Acest lucru duce la ambiguitate, rezultând în anumite momente ca utilizatorul execută în același timp „Working” și „Sleeping”. În acest caz acuratețe algoritmului scade la jumătate, deoarece activitatea executată este parțial recunoscută. Cu cât sunt adăugate mai multe activități ce descriu contexte similare cu atât mai mult o să scadă precizia algoritmului.

Din punct de vedere al preciziei cu care sunt recunoscute activitățile sunt folosite următoarele formule matematice:

$$p_{zi} = \frac{\sum t_a - \sum t_a * p_a}{24} * 100 = \frac{\sum t_a * (1 - p_a)}{24} * 100$$

p_{zi} = procentul de timp în care activitățile au fost recunoscute într-o zi

t_a = timpul în care activitatea „a” este recunoscută corect

p_a = procentul de timp în care se produce activitatea „a”, dar apar schimbări de context astfel încât activitatea „a” nu mai este recunoscută

Cu cât p_a scade și t_a crește cu atât mai mult crește procentul de recunoaștere corectă a activităților dintr-o zi. Pentru o precizie bună este nevoie de activități continue de durată cât mai mare și cu un procent de timp în care contextul se schimbă cât mai mic. De aici rezultă că activitatea „Sleeping” ar trebuie în general să îndeplinească aceste condiții, făcând-o potrivită pentru recunoaștere.

5.2 Evaluarea unui exemplu concret de funcționare

Prima activitate căruia i s-a asociat context a fost „Walking”. Descrierile contextelor acestei activități sunt prezentate în tabelul 6. Acestea reprezintă plasarea telefonului în buzunar (proximitate „Near”, iluminare foarte slabă) în poziție verticală sau orizontală. Zgomotul din jur este variabil în funcție de locul unde se execută activitatea. Temperatura dispozitivul este crescută din cauza temperaturii ridicate din ziua în care s-a efectuat experimentul.

Tabelul 6

Contextele activității „Walking”.

Senzor de lumină (lx)	Senzor de proximitate	Senzor de sunet (dB)	Senzor de gravitație	Senzor de accelerație	Senzor de baterie (°C)
[1, 10)	Near	[50, 60)	Vertical	Moving	[35, 100)
[1, 10)	Near	[40, 50)	Vertical	Moving	[35, 100)
[1, 10)	Near	[70, 80)	Vertical	Moving	[35, 100)
[1, 10)	Near	[60, 70)	Vertical	Moving	[35, 100)
[1, 10)	Near	[80, 90)	Vertical	Moving	[35, 100)
[1, 10)	Near	[50, 60)	Horizontal	Moving	[35, 100)

După această asociere dispozitivul este capabil să recunoască activitatea „Walking” și activitatea „None”. Pentru a putea avea o analiză satisfăcătoare și relevantă au mai fost înregistrată activitatea „Working”. Contextul acesteia este reprezentat de aflarea telefonului pe masă cu fața în sus, într-o cameră de cămin (iluminare și zgomot moderat).

Tabelul 7

Contextul inițial al activității „Working”.

Senzor de lumină (lx)	Senzor de proximitate	Senzor de sunet (dB)	Senzor de gravitație	Senzor de accelerație	Senzor de baterie (°C)
[10, 100]	Far	[40, 50]	Face Up	Still	[35, 100]
[10, 100]	Far	[30, 40]	Face Up	Still	[35, 100]

Recunoașterea de activități a fost demarată în timpul nopții. Din acest motiv a mai fost adăugată o activitate, ci anume „Sleeping”. Aceasta este caracterizat de iluminare extrem de mică, zgomot redus și poziționarea telefonului pe masă cu fața în sus.

Tabelul 8

Contextul inițial al activității „Sleeping”.

Senzor de lumină (lx)	Senzor de proximitate	Senzor de sunet (dB)	Senzor de gravitație	Senzor de accelerație	Senzor de baterie (°C)
[0, 1)	Far	[40, 50]	Face Up	Still	[35, 100]
[0, 1)	Far	[30, 40]	Face Up	Still	[35, 100]

Algoritmul a fost pornit la ora 2:40 și a fost ajustat la anumite momente de timp pentru a crește acuratețea de recunoaștere a activităților. În intervalul 2:40 – 5:24 aplicația a detectat activitatea „Sleeping”. Începând cu ora 5:24 algoritmul a început să detecteze timp alternativ activitățile „Sleeping” și „Walking” timp de câteva minute din cauza creșterii intensității luminii din cameră, datoră răsăritului. După aceea s-a stabilizat și a fost detectată activitatea „Walking” în intervalul 5:32 – 7:08. În acel moment, algoritmul a primit feedback de la utilizator care a etichetat contextul din intervalul 2:40 – 7:08 cu activitatea „Sleeping”. Arborii de decizie au fost recalculați, iar activitatea „Sleeping” a început să fie detectată din nou. În intervalul 7:08 – 9:10 activitatea detectată a rămas „Sleeping”, cu mici instabilități. Din cauza unei noi schimbări de context produsă de lumină și zgomot, algoritmul a început să detecteze două activități simultan „Working Sleeping”. La 11:17 utilizatorul a oferit feedback algoritmului și activitatea detectată în continuare a fost schimbată la „Working”. Algoritmul a fost oprit la ora 11:54.

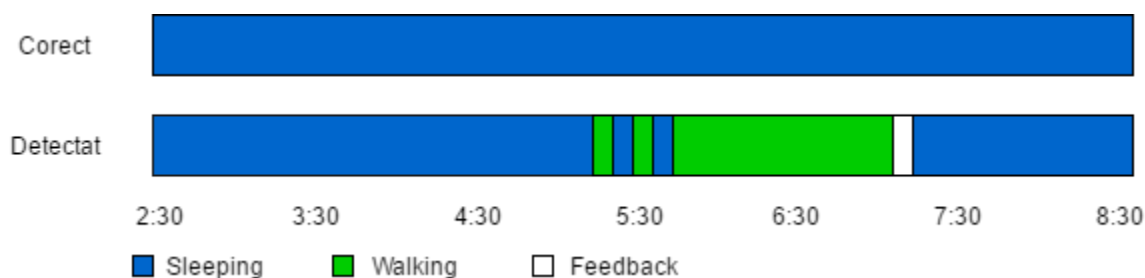


Figura 19. Evoluția algoritmului de recunoaștere de activități în intervalul 2:30 – 8:30.

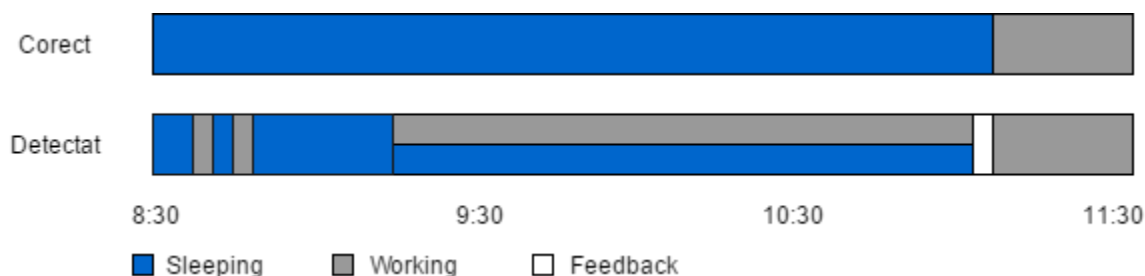


Figura 20. Evoluția algoritmului de recunoaștere de activități în intervalul 8:30 – 11:30.

Se observă din Figurile 19 și 20 modul în care algoritmul învață să recunoască o nouă activitate cu ajutorul feedback-ului provenit de la utilizator. În același timp schimbările de context sunt evidențiate prin alternanța a doua activități într-un interval mic de timp. În plus, feedback-ul utilizatorului poate rezolva problema ambiguității, atunci când două sau mai mult activități sunt detectate simultan.

În momentul în care a fost oferit feedback, activităților respective li s-au adăugat contexte noi. În Figura 19 este exemplificat primul mod de feedback, care constă în etichetarea în masă a tuturor contextelor dintr-un interval din trecut. Acestea au fost etichetate cu activitatea „Sleeping”. În figura 20 este exemplificată cea de a doua metoda de feedback: etichetarea contextului curent cu o nouă activitate. Noua activitate asociată contextului a fost „Working”.

Tabelul 9

Contextul provenit din feedback al activității „Sleeping”.

Senzor de lumină (lx)	Senzor de proximitate	Senzor de sunet (dB)	Senzor de gravitație	Senzor de accelerație	Senzor de baterie (°C)
[1, 10]	Far	[20, 30]	Face Up	Still	[30, 35]
[0, 1]	Far	[50, 60]	Face Up	Still	[30, 35]
[0, 1]	Far	[30, 40]	Face Up	Still	[30, 35]
[1, 10]	Far	[40, 50]	Face Up	Still	[30, 35]
[0, 1]	Far	[40, 50]	Face Up	Still	[30, 35]
[1, 10]	Far	[30, 40]	Face Up	Still	[30, 35]
[0, 1]	Far	[20, 30]	Face Up	Still	[30, 35]

Contextul provenit din feedback al activității „Working”.

Senzor de lumină (lx)	Senzor de proximitate	Senzor de sunet (dB)	Senzor de gravitație	Senzor de accelerație	Senzor de baterie (°C)
[10, 100]	Far	[30, 40]	Face Up	Still	[30, 35]

Acest exemplu arată cum este reglat algoritmul de învățare pentru a atinge o precizie cât mai mare. Utilizările ulterioare ale algoritmului o sa ofere o precizie din ce în ce mai mare. Luând în calcul aceste ipoteze, acuratețea acestui exemplu poate fi considerată a fi minimă.

$$p = \frac{t_d}{t_c} * 100 = \frac{319}{540} * 100 = 59 \%$$

p = procentul de timp în care activitatea a fost recunoscută corect

t_d = timpul în minute în care activitatea a fost recunoscută corect

t_c = timpul în minute în care algoritmul a fost activ

După acest exemplu de reglare al activităților, aplicația a fost pornit în același interval orar în ziua următoare pentru a observa dacă procentul de recunoaștere al activităților s-a îmbunătățit. În intervalul 2:57 – 9:00 a fost detectată activitate „Sleeping” în proporție de 99%. În Figura 21 se observă creșterea intervalului de timp în care aceasta a fost recunoscută față de exemplul anterior. La ora 9:00 contextul s-a schimbat drastic (utilizatorul s-a trezit și a lăsat lumina să intre în cameră). În continuare, de la ora 9:00 la ora 12:00 a fost detectată activitatea „Working” în proporție de 99%, lucru scos în evidență de Figura 22.

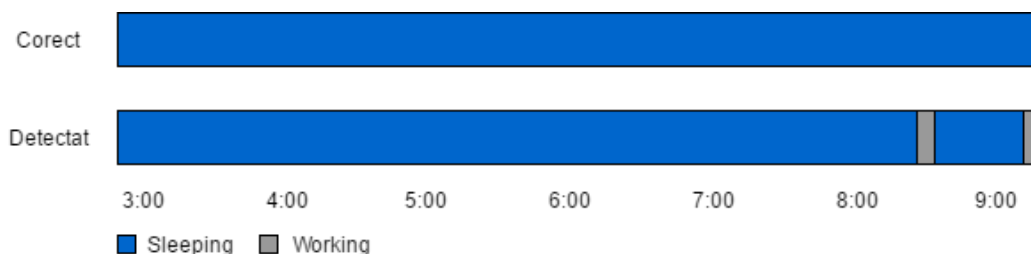


Figura 21. Evoluția algoritmului de recunoaștere de activități în intervalul 3:00 – 9:00.

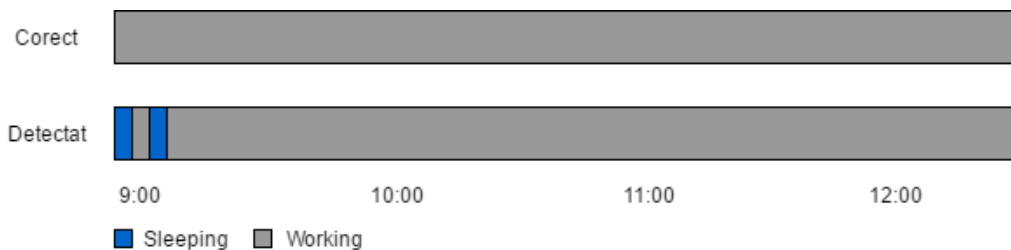


Figura 22. Evoluția algoritmului de recunoaștere de activități în intervalul 9:00 – 12:00.

Se observă o creștere substanțială a precizie algoritmului de recunoaștere de activități. Considerând intervalul de maxim 30 de minute, în care activitățile s-au intercalat și nu au fost recunoscute corect, rezultă:

$$p = \frac{t_d}{t_c} * 100 = \frac{490}{540} * 100 = 90 \%$$

Se observă o creștere semnificativă de 31 % după ce algoritmul a fost reglat prin feedback de către utilizator. În plus, procentul de 90 % indică o eficiență ridicată a algoritmului atunci când activitățile sunt alese corespunzător și când este utilizată funcția de feedback.

5.3 Evaluarea consumului de resurse

Din punct de vedere al resurselor dispozitivului mobil aplicația rulează în condiții satisfăcătoare. În momentul de față dispozitivele mobile au procesoare îndeajuns de puternice pentru a suporta algoritmul implementării care constă în construirea și parcurgerea unor arbori de decizie de dimensiuni reduse. Din punct de vedere al memorie permanente, aplicația stochează datele în trei tabele din aceeași bază de date. Pentru a evita eventuale probleme, datele de la senzori sunt codate și salvate într-un obiect de tipul *String* de dimensiune reduse, lucru care permite crearea unei singure coloane în loc de șase. Pe de altă parte, nivelul de descărcare a bateriei este afectat considerabil, din cauza monitorizării continue a senzorilor.

Tabelul 11

Resursele folosite de către aplicație atunci când este activă recunoașterea de activități.

Timp total	Putere folosită	Timp pe CPU	Memorie RAM	Baterie
9h 10m 47s	667 mAh	3h 0m 12s	35 MB în ultimele 3 ore	20 %

Rezultatele prezentate în tabelul 6 sunt obținute în urma rulării aplicației pe un telefon HTC cu o baterie cu capacitate de 3000 mAh. Senzorii sunt interogați odată la 45 de secunde, iar activitățile sunt recunoscute odată la 60 de secunde. Aplicația a rulat timp de 9 ore consecutive. Din analiza datelor obținute rezultă că aplicație folosește un cantitate mică de memorie RAM și nu afectează performanțele telefonului din acest punct de vedere. Aproximativ o treime din timpul în care aceasta a fost pornită, a fost folosit pentru procesare. Pe de altă parte, consumul de energie este afectat considerabil. Aplicația consumă aproximativ 75 mA / oră. Presupunând ca aplicația rulează timp de 24 de ore, rezultă un consum de 1800 mAh pe zi. Considerând că majoritatea telefoanelor au baterii de cel puțin 3000 mAh, aproximativ 60% din aceasta este folosită pentru recunoașterea activităților. Acest procent este redus dacă presupunem ca pe timpul nopții, aproximativ opt ore, telefonul este la încărcat. În această situație rezultă doar 1200 mAh pe zi, aproximativ 40% din energia totală a bateriei. Deși este o îmbunătățire semnificativă, puterea consumată de aplicație este prea mare.

6. Concluzii

Recunoașterea de activități este folosită din ce în ce mai mult în viața de zi cu zi a utilizatorilor. Majoritatea aplicațiilor se axează pe senzorii de mișcare (accelerometru, giroscop, etc) pentru a detecta activități fizice (walking, sitting, etc). Aplicația descrisă în această lucrare încearcă să ofere o altă viziune asupra senzoriilor dispozitivelor mobile și activităților umane. Deși implementarea curentă oferă rezultate satisfăcătoare, mai este mult de muncă până va putea fi utilizată la capacitate maximă. În subcapitolele următoare vom discuta despre obiectivele atinse în această lucrare, neajunsurile aplicației și funcționalitățile ce ar îmbunătăți implementarea curentă considerabil.

6.1 Sumar și contribuții

În această lucrare, am analizat o nouă abordare a recunoașterii de activități și am implementat-o sub forma unei aplicații pentru dispozitivele mobile ce rulează sistemul de operare Android. Am folosit șase senzori: de lumină, proximitate, sunet, gravitație, accelerație liniară și baterie. Am creat o interfață ușor de utilizat și o bază de date pentru stocarea valorilor senzorilor. Am folosit un algoritm de învățare format din mai mulți arbori de decizie pentru a recunoaște activitățile. Am testat aplicația într-un scenariu teoretic pentru a pune în evidență limitările acesteia. În plus, am testat aplicația în practică de o lungă perioadă de timp pentru a vedea cum se comportă, ce precizie oferă și cât de mult consumă resursele telefonului mobil.

În final, am obținut o aplicație ce permite utilizatorului să asocieze un tip de activitate unui sau mai multor contexte. Activitatea poate să fie de orice tip, nu numai fizică ca în majoritatea implementărilor existente. Aceasta nu este selectată dintr-o listă de activități predefinite, ci este creată de către utilizator. Recunoașterea de activități este realizată pe baza contextelor etichetate cu activități de către utilizator. Aplicația permite setarea volumului automat în funcție de activitate recunoscută. Pe de altă parte, aplicația necesită o cantitate mare de energie, având nevoie de îmbunătățiri în acest domeniu.

6.2 Dezvoltări ulterioare

În primul rând, aplicația are nevoie de mai multe acțiuni, pe lângă setarea automată a volumului, ce ajută utilizatorul atunci când acesta execută o activitate. Implementarea unor notificări inteligente ar aduce o utilitate mare utilizatorului. Acestea constau în setarea volumului soneriei dispozitivului mobil pe silențios și modificarea sa la un nivel mai mare atunci când se primesc două apeluri consecutive într-un interval scurt de timp de la același număr de telefon. Acest lucru poate fi extrapolat la primirea de emailuri de la aceeași adresă. Notificările inteligente ar permite utilizatorului să nu fie deranjat decât dacă este o urgență. O altă acțiune posibilă este personalizarea setărilor telefonului (sonerie, vibrații, wallpaper, etc) pentru fiecare activitate în parte sau atunci când este detectată o anumită tranziție între activități. Platforma Android oferă control destul de larg asupra setărilor telefonului, de unde pot rezulta o multitudine de acțiuni în funcție de activitatea curentă.

O altă modalitate de a îmbunătăți implementarea curentă este prelucrarea mai complexă a datelor venite de la senzori. Intervalele senzorilor de lumină, sunet și temperatură pot fi micșorate pentru o

delimitare mai exactă a activităților. Se pot aplica anumite filtre pe datele provenite de la microfon pentru a obține o mai bună înțelegere a contextului din punct de vedere al sunetului. Accelerometrul și senzorul de gravitație pot fi folosiți pentru a detecta poziții mai precise ale dispozitivului mobil. În plus, pot fi adăugați cu ușurință senzori noi în implementarea curentă, precum giroscop, barometru, busolă, etc.

Fiecărei activități i se poate asocia una sau mai multe locații. Acest lucru duce la o segmentare mai bună a activităților și permite adăugarea activităților cu același nume ce diferă prin locație. Detectarea locației se poate realiza în două moduri în Android. Prima soluție este folosirea GPS-ului ce oferă o precizie mare, dar nu poate fi folosit decât afară și implică un consum mare de baterie. O altă soluție, ce nu este la fel de precisă, este folosirea locației turnului de telefonie la care este conectat dispozitivul mobil. Acesta metodă consumă mai puțină baterie și este suficient de precisă pentru a permite segmentarea activităților.

Din punct de vedere al consumului de energie este nevoie de îmbunătățiri semnificative. Se poate folosi un algoritm de modificare dinamică în funcție de activitățile produse a intervalului de interogare al senzorilor și de acces la baza de date. În plus, se poate folosi istoricul activităților pentru a îmbunătăți acest algoritm. De exemplu dacă în fiecare zi în același interval de timp se produce aceeași activitate, atunci timpul de interogare al senzorilor poate fi crescut semnificativ, deoarece există o șansă foarte mică ca contextul să se schimbe.

Nu în ultimul rând, aplicația ar putea folosi o interfață ceva mai simplă și mai prietenoasă cu utilizatorul. Momentan multe din datele afișate nu sunt necesare utilizatorului de zi cu zi. Acestea sunt utile pentru a scoate în evidență modul de funcționare al implementării curente. În plus, modul de utilizarea al acesteia nu este intuitiv.

Principala problemă a aplicației în starea curentă este continuitatea activităților. Fie activitatea „Sleeping” care se desfășoară pe parcursul nopții timp de opt ore. Există posibilitatea ca în acest interval să se modifice contextul pentru o perioadă mică de timp. De exemplu după patru ore de „Sleeping” este detectată activitatea de „Working” din cauza unei surse de lumină ce a apărut brusc timp de 10 minute, după care se revine la activitatea de „Sleeping”. Este nevoie de un algoritm care decide dacă activitatea de „Working”, care a avut loc doar pentru 10 minute și este precedată și urmată de activitate „Sleeping” ce are loc de o lungă perioadă de timp, este o eroare ce trebuie ignorată sau nu. La ce interval de timp, din momentul în care este detectată o activitate nouă, trebuie considerată ca fiind activitatea curentă. Cum se poate oferi în timp real utilizatorului opțiune de a vedea activitatea curentă, în timp ce este asigurat că nu este o eroare ce are loc doar pentru câteva minute. Mare problemă a aplicației este lipsa unui algoritm de predicție a acestor anomalii ce pot avea loc în timpul unei activități de lungă durată.

BIBLIOGRAFIE

- [1] C. Petri (2010). „Decision Trees”. [paper] Universitatea Babeș-Bolyai, Cluj-Napoca.
- [2] Google Developers, (2014). Activity Recognition Api. [online] Available at: <https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi> [Accessed 19 Jun. 2017].
- [3] T.D. Le, C.V. Nguyen (2015). „Human Activity Recognition by smartphone”. 2015 2nd National Foundation for Science and Technology Development Conference on Information and Computer Science, pp. 219-224.
- [4] G. Sebestyen, I. Stoica, A. Hangan (2016). „Human Activity Recognition and Monitoring for Elderly People”. 2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing, pp. 341-347.
- [5] E. Kim, S. Helal, D. Cook (2010). „Human Activity Recognition and Pattern Discovery”. IEEE Pervasive Comput. 2010.
- [6] D. Anguita, A. Ghio, L. Oneto, X. Parra, J.L. Reyes-Ortiz (2012). „Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support Vector Machine”.
- [7] S. Sarker, A. K. Nath, A. Razzaque (2016). „Tradeoffs Between Sensing Quality and Energy Efficiency for Context Monitoring Applications”. 2016 International Conference on Networking Systems and Security (NSysS).
- [8] L. Fang, S. Yishui, C. Wei (2016). „Up and Down Buses Activity Recognition using Smartphone Accelerometer”. 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference, pp. 761-765.
- [9] J. Helmy, A. Helmy (2016). „The Alzimio App for Dementia, Autism & Alzheimer’s: Using Novel Activity Recognition Algorithms and Geofencing”. 2016 IEEE International Conference on Smart Computing (SMARTCOMP).
- [10] M. Shoaib, S. Bosch, O.D. Incel, H. Scholten, P.J.M. Havinga (2015). „A Survey of Online Activity Recognition Using Mobile Phones”. HCI In Smart Environments, pp. 2060-2085.
- [11] M. Boukhechba, A. Bouzouane, B. Bouchard, C. Guin-Vallerand, S. Giroux (2016). „Energy Optimization for Outdoor Activity Recognition”.
- [12] A.B. Abkenar, S.W. Loke, W. Rahayu, A. Zaslavsky (2016). „Energy Considerations for Continuous Group Activity Recognition Using Mobile”. 2016 IEEE 30th International Conference on Advanced Information Networking and Applications, pp.479-486.
- [13] J.B. Rana, T. Jha, R. Shetty (2015). „Applications of Machine Learning Techniques in Human Activity Recognition”.
- [14] Android Developers, (2014). Sensors overview. [online] Available at: https://developer.android.com/guide/topics/sensors/sensors_overview.html [Accessed 16 Jun. 2017].
- [15] University of Florida (1997). The ID3 Algorithm. [online] Available at: <https://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-papers/2.htm>