Universitatea
Ștefan cel Mare
Suceava

Facultatea de Inginerie Electrică
și Știința Calculatoarelor

Controlul unui dispozitiv mobil de la distanță folosind interfață web

Profesor coordonator:

Conf.dr.ing. Ovidiu-Andrei Schipor

Student:

Rebeca-Ligia Purice

Suceava, Iulie 2018

Cuprins

Lista de abrevieri

AWS – Amazon Web Services

iOS – iPhone Operating System

HTML – Hypertext Markup Language

CSS – Cascading Style Sheet

DSL – Domain-specific Language

WSL – Windows Subsytem for Linux

API – Application Programming Interface

Wi-Fi – Wireless Fidelity

RC – Radio Control

DC – Direct Curent

IP – Internet Protocol

PAN – Personal Area Network

LAN – Local Area Network

IEEE – Institute of Electronical and Electronics Engineers

CRC – Cyclic Redundancy Check

TCP – Transfer Control Protocol

OSI – Open Systems Interconnection

WEP – Wired Equivalent Privacy

WPA – Wi-Fi Protected Access

Lista de figuri

1. Introducere

Am ales să rezolv această temă în vederea susținerii lucrării de diplomă deoarece aceasta integrează atât o parte software cât și o parte hardware și permite utilizarea cunoștințelor acumulate în cadrul mai multor materii studiate în facultate. Am fost conștientă că alegând această temă spre implementare nu va fi ușor pentru mine deoarece trebuia să studiez mai multe subiecte noi, care nu au fost integrate în programa studiată la cursuri. Însă având experiența realizării proiectelor în cadrul cursurilor și laboratoarelor știam că voi asimila mult mai ușor noțiunile noi dacă lucrez eu însumi să caut detalii și modalități de rezolvare pentru a realiza un proiect funcțional.

Putem observa foarte ușor că tehnologia este într-o continuă dezvoltare și dorim să ne asigurăm un anumit grad de comoditate atunci când interacționăm cu diferite dispozitive sau aplicații. Pentru programatori și pentru cei care interacționează foarte des cu acest fel de „insrumente" este foarte tentant să încerce să îmbunătățească modul de viață în acest sens deoarece dețin cunoștințele necesare pentru acest lucru, iar dacă nu le dețin încă și le pot însuși repede cu suficient interes și muncă.

Ideea de la care am pornit a fost să creez o aplicație care să permită controlul de la distanță asupra unui dispozitiv mobil (o mașină în cazul nostru). Am ales ca mijloc de control o interfață web deoarece este portabilă pe mai multe tipuri de dispozitive și sisteme de operare, oferind astfel posibilitatea acoperirii unui numar mai mare de utilizatori ai aplicației. Această interfață va putea fi rulată pe orice telefon mobil, iar controlul propiu-zis al mașinii va fi realizat înclinând telefonul într-una din cele 4 direcții de mers posibile pentru o mașină- stânga, dreapta, înainte sau înapoi. Datele generate de senzorii de mișcare din componenta fizică a telefonului vor fi preluați și transmiși către circuitul care coordonează activitatea mașinii. Așadar, ceea ce reprezintă deocamdată o condiție în a fi utilizator al aplicației este accesul la un smartphone care să aibă sistemul de operare Android și care la fabricare a fost dotat cu accelerometru. Cu timpul intenționez să dezvolt funcționalitatea aplicației și pentru sistemul de operare iOS. Este adevărat că o pagină web poate fi deschisă pe mai multe sisteme de operare- Windows, Linux, etc. Dar nu este comod să utilizăm un laptop pentru a direcționa mașina, deoarece funcționalitatea aplicației necesită o înclinare a dispozitivului responsabil cu determinarea direcției pentru a coordona mașina pe traiectoria dorită. În funcționarea aplicației intenționez să implementez soluțiile pentru a asigura un grad de confort ridicat utilizatorului, iar în opinia mea utilizarea unui dispotiv de dimensiuni mici pentru specificarea direcției satisface acest aspect.

După stabilirea în linii largi a modului de funționare al aplicației am considerat oportun ca fiecare utilizator să poată fi identificat unic în vederea accesului la aplicație. Nu este de dorit ca oricine să poată avea acces la aplicațiile instalate pe telefonul unei anumite persoane. De aceea, pentru a putea accesa interfața responsabilă de coordonarea mașinii, pentru a asigura un grad ridicat de securitate, am implementat un mod de autentificare mai sigur, mai rapid și mai comod pentru utilizator decât majoritatea metodelor de autentificare folosite în prezent. Suntem obișnuiți ca atunci când accesăm un cont să fim nevoiți să introducem un nume de utilizator și parola asociată pentru a accesa componenta principală a

aplicației. Un alt mod de autentificare utilizat în mai puține domenii este autentificarea folosind imaginea feței. Dacă numele de utilizator și parola pot fi transmise sau „furate”, lucrurile nu stau la fel și în vederea feței. Astfel, prin implementarea aplicației utilizând conceptele amintite mai sus cred că mi-am îndeplinit scopul de a asigura un grad de încredere ridicat pentru folosirea acestei interfețe, dar și o interacțiune facilă și comodă pentru a realiza coordonarea unui dispozitiv mobil de la distanță.

2. Fundamente teoretice și tehnice

În prezentarea succintă realizată în capitolul anterior am amintit faptul că interfața este constituită din mai multe pagini web, fiecare dintre ele fiind responsabilă de anumite acțiuni pentru a asigura funcționalitatea finală a aplicației. În continuare vom prezenta noțiunile teoretice esențiale înțelegerii funcționării acestui proiect.

2.1. Tehnologii Web

2.1.1. HTML, JavaScript, CSS

Abrevierea HTML provine de la sintagma HyperText Markup Language:

1. **HyperText** indică metoda pe care o folosim pentru a naviga pe web. Accesând anumite texte cu format special numite hyperlink-uri putem trece dintr-o pagină în alta. „Hyper" denotă faptul că dintr-un anume loc putem accesa orice altă adresă, iar ordinea de navigare nu este prestabilită.
2. **Markup** semnifică modul în care influențează tag-urile HTML textele scrise în interiorul lor.
3. **HTML** este un limbajde descriere, deoarece asemenea altor limbaje are cuvinte cheie și sintaxă proprie.

O pagină web este un document scris în descriere HTML care este interpretat de un browser. De asemenea, una dintre definițiile tradusă din limba engleză spune ca o pagină web este un set de date sau de informații care este proiectată pentru a fi vizualizată ca o parte a unui website[1]. Așadar, o pagină web este diferită de un website prin faptul că cel din urmă este tocmai o colecție de pagini web. O pagină web poate fi statică sau dinamică. O pagină statică va prezenta același conținut de fiecare dată când va fi accesată, iar o pagină dinamică poate avea conținut diferit la fiecare accesare.

Modul în care este stilizată o pagină web este precizat în fișerul CSS, care este apelat în interiorul fișierului HTML. Standardul CSS conține specificații tehnice cu privire la organizarea în pagină. Într-un fișier CSS pot fi stilizate majoritatea elementelor care se regăsesc în fișierul HTML și care sunt vizibile utilizatorului. Elementele pot fi identificate în CSS prin id-ul lor, prin nume de tag-uri sau prin nume de clase asociate acestora. Orice pagină web este afectată de cel putin un script CSS, chiar și în cazul în care realizatorul paginii nu a menționat acest lucru. Acest script este cunoscut și sub denumirea de **User Agent StyleSheet**, deoarece în absența specificării unui anumit mod în care se dorește aranjarea în pagină browser-ul are nevoie totuși de un script implicit care să se ocupe de acest lucru.

Funcționalitatea propriu-zisă a unei pagini web este asigurată de fișierele JavaScript. În cadrul acestora se scrie codul corespunzător acțiunilor care așteptăm să fie realizate la accesul unui buton, al unui link sau la declanșarea anumitor evenimente.

JavaScript este abreviat JS și este un limbaj de nivel înalt. Este suportat de majoritatea browser-elor incluzând Firefox, Safari, Chrome, Internet Explorer, etc. Oferă posibilitatea paginilor web de a deveni interactive. JavaScript a fost realizat pentru a funcționa împreună cu structura HTML a unei pagini. De obicei codul este organizat în funcții care asigură

caracteristica interactivă de care am amintit mai sus. Pot fi realizate validări pentru anumite câmpuri din formularele paginlor web, calcule, pot fi afișate mesaje în interiorul paginii rezultate în urma unor prelucrări. Elementele din interiorul paginii sunt accesate prin intermediul id-ului lor pentru a putea fi prelucrate.

Prin urmare, se poate afirma că cele trei componente care stau la baza creării paginilor web sunt foarte strâns îmbinate deoarece:

1. HTML descrie pagina per ansamblu (ce imagini vom regăsi pe pagină, ce text);
2. CSS este utilizat pentru a personaliza vizual pagina (specificarea culorilor, mărimea textului);
3. JavaScript reprezintă componenta dinamică a paginii web care face ca mare parte din componentele paginii să fie programabile.

2.1.2. Node.js

Node.Js a fost creată de către Ryan Dahl în anul 2009 și este o platformă construită pe motorul V8 al Chrome și se bazează pe un model event-driven și non-blocking I/O, folosind callback-uri. Este construită pe runtime-ul JavaScript de la Chrome, favorizează modularitatea, iar aplicațiile sale single-thread sunt scrise în JavaScript[2]. Prima versiune era suportată doar de sistemele de operare Linux și Mac OS X.

2.1.3. Ruby

Ruby este un limbaj de programare dinamic și open source care se concentrează pe simplitate și productivitate. Inițial a fost creat pentru programarea de jocuri. Are o sintaxă elegantă ușor de înțeles. Conține foarte multe cuvinte cheie în limba engleză ceea ce atrage chiar și programatorii începători să-l folosească.

Pentru construirea limbajului Ruby, creatorul său a îmbinat părți din limbajele sale favorite: Smalltalk, Ada, Eiffel, Lisp și Perl. Astfel a rezultat un nou limbaj care îmbină programarea funcțională cu programarea imperativă. Programarea funcțională pune accent pe calcul în evaluarea funcțiilor matematice evitând starea, iar programarea imperativă pune accent pe schimbările de stare. Yukihiro „Matz" Matsumoto, autorul limbajului, a spus deseori că „nu încearcă să facă Ruby să fie simplu, ci natural"[3]. O altă idee care aparține lui Matz cu privire la acest limbaj este: „Ruby este simplu în aparențe, dar este foarte complex în interior, precum corpul omenesc."[4]

În cadrul limbajului Ruby, totul este privit ca un obiect. Pentru orice parte de cod și de informație care alcătuiește un cod Ruby putem accesa proprietățile și acțiunile predefinite acestora. Programarea orientată pe obiecte numește proprietățile instanțe ale variabilelor, iar acțiunile sunt denumite metode. Numerele și alte tipuri priimitive nu sunt considerate obiecte în alte limbaje de programare. Dar, după cum am precizat la începutul prezentării acestui subcapitol, în Ruby se regăsesc influențe din limbajul Smalltalk care determină asocierea de instanțe și metode tuturor tipurilor definite de Ruby.

Este văzut ca un limbaj flexibil, deoarece utilizatorul poate face modificări în definiția cuvintelor cheie, poate elimina sau rescrie părți importante care stau la baza definirii limbajului. Programatorului nu îi este astfel restricționată creativitatea în scrierea codului în Ruby.

Caracteristica sa dinamică denotă faptul că aceeași porțiune de cod poate însemna lucruri diferite în funcție de contextul în care este introdusă. Acesta poate constitui un dezavantaj în cadrul aplicațiilor complexe deoarece este mai dificil de urmărit și de corectat erorile. Așadar este necesar un grad de experiență ridicat pentru a stabili un mod eficient de a organiza codul. De asemenea, procesarea codului scris în Ruby poate fi încetinit datorită flexibilității sale deoarece dispozitivul pe care este rulat codul trebuie să facă multe referințe pentru a fi sigur care este definiția exactă a codului scris. Acest lucru afectează în mod negativ performanțele codului Ruby.

În privința sistemului de operare utilizat pentru rularea aplicațiilor Ruby este recomandat Linux sau WSL. Majoritatea librăriilor Ruby sunt create pentru a fi rulate pe Linux și pot interveni erori în timpul execuției pe Windows.

În concluzie, optarea pentru scrierea codului în Ruby prezintă un avantaj datorită flexibilității codului, datorită carcateristicii dinamice și a cuvintelor cheie ușor de înțeles, însă trebuie acordată o atenție sporită în cazul programelor de mari dimensiuni.

2.1.3.1. Sinatra

Sinatra este un DSL și totodată o librărie utilizată pentru crearea rapidă a aplicațiilor web, folosind Ruby, cu un minim de efort[5]. Este o alternativă pentru alte framework-uri care stau la baza creării aplicațiilor web cum ar fi:

- Ruby on Rails;
- Nitro;
- Merb;
- Camping.

Autorul acestui framework este Blake Mizerany. Sinatra este o librărie simplu de folosit și flexibilă. Deși framework-urile menționate mai sus au la bază standardul model-view-controller, Sinatra nu respectă această organizare.

În crearea unui proiect Ruby utilizând framework-ul Sinatra este necesară instalarea gem-ului framework-ului:

gem install sinatra

Rularea proiectului se va realiza prin următoarea comandă în consolă:

ruby <nume_proiect>.rb

Pagina web corespunzătoare fișierului Ruby rulat va fi disponibilă la link-ul *http://localhost:4567*.

În Sinatra putem vorbi despre rute (routes). O rută reprezintă o metodă HTTP care se potrivește cu sintaxa unui link URL.

get '/' do

<afișează conținut pagină>

end

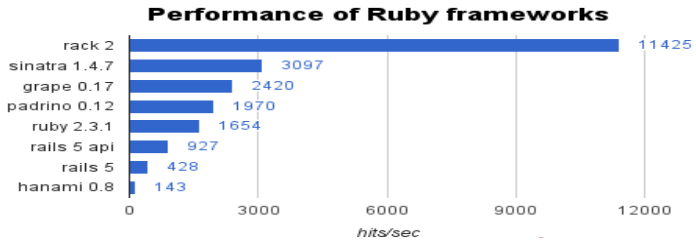
Sintaxa de mai sus setează conținutul paginii care va fi afișat în momentul în care este accesat link-ul asociat paginii principale a proiectului. După caracterul *'/'*, între apostroafe, putem specifica orice cuvânt cheie dorim pentru a preciza pagina pe care dorim să o accesăm sau pentru a specifica o operație ce trebuie efectuată și instrucțiunile sale în blocul respectiv.

Rutele vor fi evaluate în ordinea specificării lor în fișier.

Imaginea următoare specifică utilitatea framework-ului Sinatra și realizează o paralelă cu alte două framework-uri:

Figură 2-1 Sinatra vs Padrino vs Ruby on Rails[6]

După cum se poate vedea, este de preferat a se utiliza Sinatra în realizarea aplicațiilor mai simple, deoarece organizarea codului și structura fișierelor pot duce la anumite scăderi ale performanțelor. În imaginea următoare este realizată o comparație a perfomanțelor pentru Sinatra și alte 6 framework-uri:



Figură 2-2 Perfomanțele framework-urilor Ruby[7]

2.1.4. Amazon Rekognition API

Un alt aspect teoretic pe care doresc să-l prezint în acest capitol este Amazon Rekognition API. Este un instrument esențial în dezvoltarea proiectului meu deoarece prin intermediul acestuia am reușit să realizez autentificarea utilizatorului în aplicație prin intermediul feței.

Aș dori să prezint câteva dintre serviciile pe care le poate oferi acest API deoarece m-a uimit multitudinea de domenii pentru care a fost dezvoltat. Ca o prezentare generală se poate spune că prin intermediul Rekognition API-ului este diminuat considerabil efortul și implementarea unui cod pentru lucrul cu imagini și analiza videourilor dintr-o aplicație. Furnizând o imagine sau un video acestui API, poate identifica oameni, obiecte, texte, situații, diverse activități și chiar conținuturi inadecvate din anumite imagini sau videoclipuri. API-ul Rekognition de la Amazon dispune de o acuratețe a analizei fețelor foarte înaltă. Poate analiza, compara și detecta fețe pentru o multitudine de scopuri, dintre care amintesc identificarea utilizatorilor sau stabilirea numărului de persoane prezente într-o fotografie.

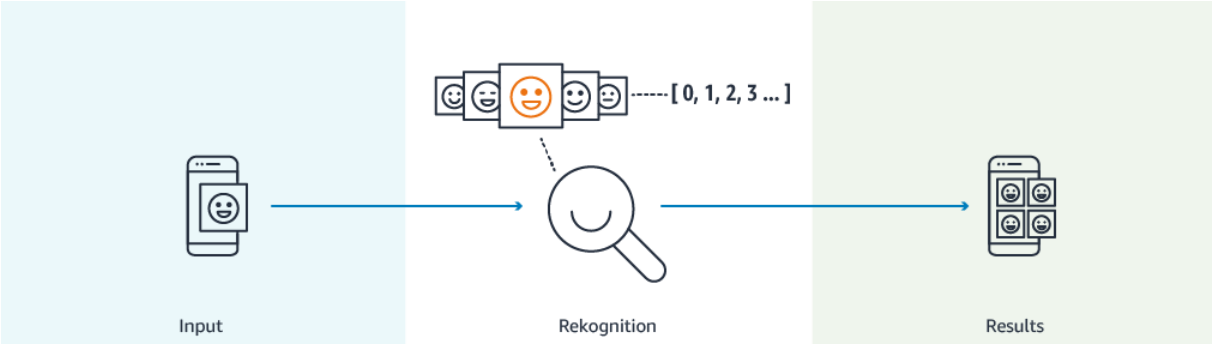
Avantaje ale utilizării Rekognition API:

- Încărcarea de fotografii și de videoclipuri pentru analiză este ușor de înțeles și de realizat;
- Acest serviciu este îmbunătățit permanent; este antrenat pentru a face față celor mai noi tipuri de informații, antrenarea presupunând adăugarea de noi date pentru crește perfomanțele în a recunoaște fețe, obiecte, activități sau situații.
- Timpul de răspuns necesar execuției unor cereri este în cele mai multe cazuri constant, indiferent de volumul de cereri pe care un utilizator le solicită.
- Analiza pe un videoclip este realizată în timp real cu ajutorul serviciului Amazon Kinesis Video Streams. Același lucru se poate spune și despre analiza imaginilor încărcate pe Amazon S3.
- Plata pentru acest serviciu se contorizează în funcție de numărul de imagini sau de minutele dintr-un videoclip supuse analizei și în funcție de caracteristicile stocate pentru fiecare înregistrare a unei fețe necesare recunoașterii faciale.

Rekognition API poate realiza următoarele acțiuni asupra imaginilor și videoclipurilor:

- Detecția obiectelor (masă, chei, mașină), scenelor (plajă, parcare) și a activităților. Cea din urmă este specifică pentru analiza videoclipurilor.
- Recunoașterea feței – permite identificarea unei persoane dintr-o fotografie sau dintr-un videoclip utilizând o colecție personală de imagini ale fețelor înregistrate anterior analizei.
- Analiza facială – la încărcarea imaginii este realizată o analiză a atributelor feței pentru a specifica dacă persoana din imagine este veselă sau tristă sau pentru a identifica prezența anumitor elemente în cadrul feței. Sunt analizați în acest caz ochii, gradul de fericire, existența ochelarilor și altele. Un lucru interesant dacă realizăm această analiză asupra videoclipurilor este că poate fi determinată și schimbarea care intervine după un timp, cum ar fi stările emoționale succesive prin care trece o persoană.
- Detectarea conținutului periculos sau inadecvat.
- Recunoașterea celebrităților.
- Detectarea textului dintr-o imagine.

În continuare doresc să prezint schema care stă la baza funcționării recunoașterii feței, acea parte a Rekognition API-ului pe care am folosit-o și în realizarea proiectul de diplomă:



Figură 2-3 Schema funcționării Rekognition API pentru recunoașterea feței[8]

Inițial se creează o colecție care să înregistreze caracteristicile fiecărei fețe pe care dorim să o stocăm. Aceste caracteristici se numesc metadate, iar memorarea acestora se realizează utilizând funcționalitatea IndexFaces API. Când se dorește a afla cine este persoana dintr-o imagine se utilizează funcționalitatea SearchFaces care va întoarce ca rezultat acele metadate pentru care este găsit un indice de confidență ridicat, în ordine descrescătoare indicelui.

Voi explica succint ce funcționalitate îndeplinește fiecare element din schema prezentată mai sus:

- Input – imaginea încărcată prin intermediul aplicației care preia fotografii de la utilizator;
- Rekognition – caută în colecția de fețe similarități cu grad mare de potrivire pentru fața detectată din imaginea încărcată de utilizator și întoarce un șir de metadate asociate fețelor din colecție pentru care se găsește o potrivire, în ordinea similarităților;
- Rezultate – utilizatorul poate vedea rezultatele obținute în urma căutării.

2.2. Comunicația fără fir

Tipuri de comunicații fără fir:

- infraroșu – este o tehnologie folosită pentru transferul de date având o rază de acoperire mică, iar pentru conectarea între dispozitive este necesar ca acestea să nu se afle în deplasare sau să existe obstacole în câmpul lor vizual. Un avantaj al acestei tip de comunicație este consumul redus de energie.
- bluetooth (PAN) – comunicarea este posibilă atunci când dispozitivele bluetooth se află în aceeași rază de acțiune. Acest tip de comunicație este des întâlnită, fiind folosită pentru schimb de informații între telefoane mobile, laptop-uri, calculatoare, camere digitale etc. Securitatea folosirii acestui tip de comunicație între telefoanele mobile stă în faptul că schimbul de date nu se poate face fără un acord de transfer.
- Wi–Fi (LAN) – Wireless Fidelity: rețele care folosesc unde electromagnetice din domeniul radio. Este cel mai răspândit tip de rețea, deoarece undele radio trec prin pereți şi alte obiecte solide. Este o tehnologie construită pe baza standardelor de comunicaţie din familia IEEE 802.11 utilizate pentru rețele locale de comunicaţie fără

fir la viteze echivalente cu cele ale rețelelor Ethernet. Raza de acoperire a unei rețele fără fir poate fi limitată la nivelul unei camere sau poate fi mai mare.

Avantaje ale comunicațiilor fără fir:

- Mobilitate: asigură conectarea fără dificultăţi atât a clienţilor staţionari, cât şi a clienţilor mobili.
- Scalabilitate: suportă conectarea unui număr mare de echipamente noi şi sporirea razei de acţiune.
- Flexibilitate: oferă conectivitate fără întreruperi clienţilor.
- Costuri reduse: costurile echipamentelor scad pe măsură ce tehnologia avansează.
- Fiabilitate în condiţii grele: uşor de instalat în condiţii de urgenţă.

Dintre aceste tipuri de comunicare fără fir, pentru proiectul constuit am ales să folosesc comunicarea prin Wi-Fi deoarece plăcuța Raspberry Pi Zero W este prevăzută cu un astfel de modul.

2.2.1. Comunicarea prin Wi-Fi

Standardul IEEE 802.11 descrie protocoale de comunicație aflate la nivelul gazdă-rețea al Modelului TCP/IP, respectiv la nivelurile fizic și legătură de date ale Modelului OSI. Aceasta înseamnă că implementările IEEE 802.11 trebuie să primească pachete de la protocoalele de la nivelul rețea (IP) și să se ocupe cu transmiterea lor, evitând eventualele coliziuni cu alte stații care doresc să transmită[9].

Spre deosebire de Ethernet, mediul de transmisie aduce probleme de securitate suplimentară. Dacă în Ethernet, accesul la cablu se putea restricționa prin ascunderea sau asigurarea zonelor prin care trece acesta, undele radio sunt mult mai dificil de controlat. Există mecanisme de bruiaj, care generează un zgomot electromagnetic ce acoperă frecvențele folosite de rețelele 802.11, dar acestea nu pot funcționa perfect, fără a afecta comunicațiile legitime sau fără a lăsa breșe prin care se poate obține acces în rețea. Cum la nivel fizic securitatea este dificil de asigurat, pentru obținerea unui nivel de securitate acceptabil este obligatorie criptarea datelor și controlul accesului la nivelele superioare celui fizic.

WEP

Prima tehnică de criptare a cadrelor la nivelul legătură de date a fost WEP (Wired Equivalent Privacy), numele sugerând că a fost gândită cu scopul de a obține o securitate a legăturii de date echivalentă cu cea a unei rețele Ethernet. Această tehnică a fost folosită din 1997 până când a fost spartă în 2001 și a încetat să mai fie considerată sigură din 2005 odată cu publicarea standardului de securitate IEEE 802.11i.

WEP folosea algoritmul RC4, cu o cheie constantă de-a lungul transmisiunii, în variantele pe 64 de biți (cheie de 40 de biți și vector de inițializare de 24) sau de 128 de biți (cheie de 104 biți și vector de inițializare de 24), controlul integrității datelor realizându-se printr-o sumă de control CRC. În modul de lucru cel mai sigur, cel cu cheie partajată, autentificarea stațiilor se făcea printr-un mecanism de challenge: după ce o stație anunță că dorește să se autentifice, punctul de acces alege aleator un text clar și îl trimite stației; stația criptează textul primit și îl trimite înapoi punctului de acces; punctul de acces decriptează mesajul și îl compară cu cel trimis inițial, permițând sau respingând accesul în consecință. După permiterea accesului, transmisia cadrelor se face criptat cu cheia rețelei.

WPA și WPA2

Ca răspuns la spargerea WEP, Wi-Fi Alliance a produs în 2003 specificația WPA (Wi-Fi Protected Access), în care a adresat problemele primare ale WEP. În WPA, s-a păstrat algoritmul de criptare simetrică RC4, dar s-a introdus în schimb TKIP (Temporary Key Integrity Protocol), o tehnică de schimbare a cheii de criptare pe parcursul sesiunii de lucru și s-a înlocuit suma de control CRC-32 din WEP cu algoritmul Michael, deoarece cu CRC recalcularea sumei de control unui cadru alterat nu necesita cunoașterea cheii de criptare.
IEEE a preluat specificația WPA și a elaborat în 2004 pe baza ei standardul IEEE 802.11i, standard care stabilește o politică de criptare cunoscută sub numele de WPA2. În WPA2, algoritmul de criptare RC4 este înlocuit și el cu mai puternicul algoritm AES, iar suma de control a cadrului este calculată cu ajutorul CCMP, un cod mai sigur decât CRC și decât algoritmul Michael[10].

WPA și WPA2 pot funcționa în două moduri distincte. Cel mai simplu dintre acestea, folosit în general la rețele personale (casnice sau ale unor firme mici), presupune configurarea stațiilor cu ajutorul unei parole de acces, parolă din care se calculează cheile de criptare cu ajutorul funcției PBKDF (Password-Based Key Derivation Function). În celălalt mod, WPA2 autentifică stațiile de lucru cu ajutorul unui server RADIUS[11].

2.3. Raspberry Pi Zero W

2.3.1. Sisteme de operare – Raspbian Stretch

Raspberry Pi este o placă de dezvoltare de tip SBC (Single Board Computer) . Nefiind altceva decât un calculator de dimensiuni mai mici, pe parcursul timpului au fost dezvoltate diferite sisteme de operare compatibile cu acesta, dintre care amintim: Raspbian, RaspBMC, Pidora, RISC OS, NetBSD, Plan 9, Slackware Linux, Android, Firefox OS, OpenELEC, XBMC, Gentoo Linux, FreeBSD etc.

Un avantaj important al unei plăci Raspberry Pi comparativ cu un calculator personal (PC) îl reprezintă prețul redus, lucru ce se reflectă în puterea mică de calcul a acesteia.

În vederea realizării proiectului de diplomă am ales să rulez pe plăcuța Raspberry Pi Zero W sistemul de operare Raspbian Stretch. Raspbian este un sistem de operare special optimizat pentru Raspberry Pi, iar Stretch este numele dat versiunii dezvoltate pentru Debian9.

2.3.2. Web Server

Un server web reprezintă un server ce găzduiește paginile web pe care, ulterior, le pune la dispoziție clienților folosind protocolul HTTP. Relația server-client se bazează pe o aplicație care, odată instalată pe server, face posibil transferul paginilor web găzduite – intervine noțiunea de hosting (găzduire), asta deoarece serverul trebuie să dețină datele pe care urmează să le returneze la cerere.

În momentul în care este introdusă o adresă, browser-ul împarte URL-ul în 3 părți:

- Protocolul de transfer hypertext: http - reprezintă limbajul pe care-l folosește browser-ul pentru a comunica cu serverul web.

- Numele de domeniu: www.exemplu.com – tastarea domeniului într-un browser presupune găsirea de către furnizorul de servicii Internet a DNS-ului care are respectivul nume de domeniu și redirecționarea conexiunii spre server unde sunt stocate fișierele ce, ulterior, vor apărea ca un site web.
- Numele fișierului: exemplu.html.

2.3.2.1. Apache

Apache este un server HTTP de tip open source. Apache a jucat și joacă un rol important în dezvoltarea webului, fiind folosit în prezent în circa 65.2 % din paginile web. Apache este un server web cu o contribuție notabilă la dezvoltatea Internetului (world wide web). Apache a reprezentat prima alternativă viabilă la Netscape Communications Corporation, și a evoluat rapid în funcționalitate și performanță ca un rival competitiv pentru alte servere web bazate pe Unix.

Apache este dezvoltat de o comunitate deschisă de programatori sub emblema Apache Software Foundation. Aplicația este disponibilă pentru o mare varietate de sisteme de operare incluzând Unix, FreeBSD, Linux, Solaris, Novell NetWare, Mac OS X, Microsoft Windows si OS/2.

Serverul Apache este caracterizat ca fiind un software gratuit și open source, acesta făcând ca, începând din aprilie 1996, el să fie cel mai popular server HTTP. Cu toate că în noiembrie 2005 a început să piardă din cota de piață, în aprilie 2008 Apache stătea încă la baza a peste 50 % din siturile web, iar în iunie 2013 a ajuns la 65.2%. Apache este folosit pentru 46,91% din totalul domeniului românesc[12].

3. Proiectarea şi implementarea dispozitivului

Pentru realizarea proiectului de diplomă am urmat câteva tutoriale[13] de pe site-ul **w3school.com** unde am putut observa diverse proiecte realizate folosind Raspberry Pi, Node.Js şi socket-uri. De exemplu realizarea unei aplicaţii web de control al unui led (aprindere/stingere), realizarea unei aplicaţii de control a intensităţii luminoase sau chiar aplicaţie de control a culorii unui led RGB. Deoarece Raspberry Pi este un mini calculator, acesta poate fi folosit pentru a crea nenumărate proiecte folosind diverse tehnologii. Ca şi în cazul proiectului pe care am ales să îl realizez, proiectele ce pot fi realizate cu Raspberry Pi pot îmbina partea software cu partea hardware. Pentru paginile care utilizează recunoaşterea feţei am consultat de asemenea un tutorial[14] şi documentaţia de la Amazon[15].

Pentru implementarea dispozitivului mobil m-am folosit de două mari componente: componenta hardware şi componenta software. În acest capitol voi prezenta detaliat din ce este alcătuită fiecare dintre cele două componente, modul de funcţionare al fiecăreia şi modul în care cele două componente comunică.

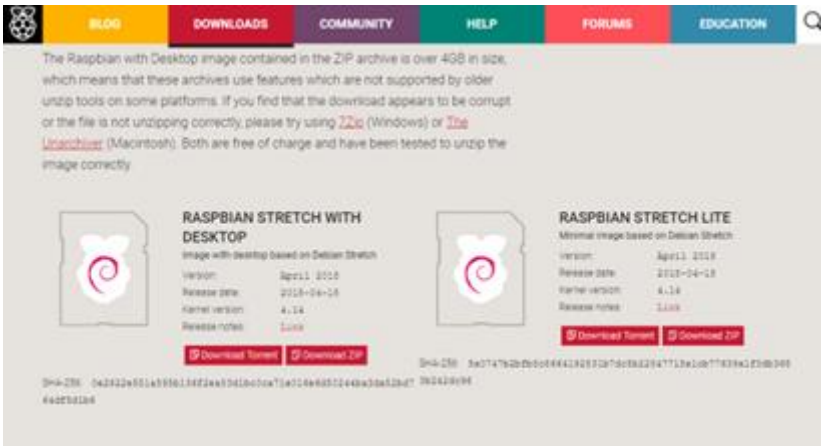
Primul pas în proiectarea dispozitivului a fost alegerea resurselor hardware astfel încât să pot ajunge la îndeplinirea scopului final al proiectului. Al doilea pas a fost alegerea resurselor software astfel încât să pot realiza o comunicaţie facilă între partea hardware şi partea software, dar şi asigurarea unei interacţiuni între utilizator şi dispozitiv cât mai facilă.

3.1. Proiectarea şi implementarea software

În cadrul acestui subcapitol voi prezenta resursele software pe care le-am folosit, atât pentru realizarea interfeţei cât şi pentru controlul dispozitivului şi pentru asigurarea funcţionalităţii proiectului.

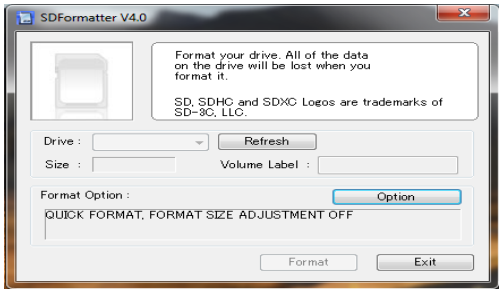
3.1.1. Instalarea resurselor software

Primul pas a fost instalarea sistemului de operare ce va rula pe placa Raspberry Pi Zero W. Pentru aceasta a trebuit descărcată o versiune a sistemului de operare Raspbian Stretch[16] de pe site-ul oficial.



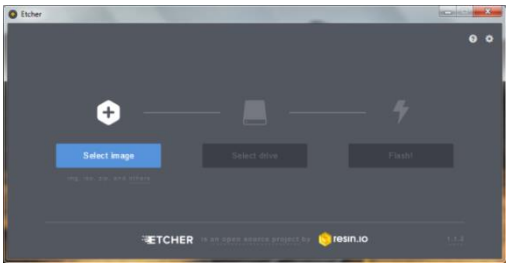
Figură 3-1 Sursă descărcare Raspbian Stretch

Următorul pas a fost formatarea cardului microSD ce urmează a fi folosit ca memorie internă. În acest scop am folosit programul SDFormatter[17]. Pentru acest proiect am folosit un card microSD de la Kingston cu o capacitate de stocare de 16 GB.



Figură 3-3 Formatare card cu SDFormatter

Am ajuns în punctul în care va avea loc instalarea sistemului de operare pe cardul microSD. Pentru a duce la capăt acest pas am folosit un alt tool și anume Etcher[18].



Figură 3-2 Instrumentul Etcher

Pentru conectarea la plăcuță într-o primă fază m-am folosit de un cablu de tip Micro Usb și de terminalul Putty, urmând făcute setările pentru conectarea la rețeaua internet prin Wi-Fi.

Pașii urmați pentru conectarea automată la rețeaua internet au fost următorii:

- Modificarea fișierului config.txt prin adăugarea la sfârșit a următoarei setări:
  dtoverlay=dwc 2
- Modificarea fișierului cmdline.txt adăugând setarea ***modules-load=dwc2,g_ether***
- Folosind Putty m-am conectat la plăcuța Raspberyy Pi Zero W și, intrând în folderul ***/etc/wpa_supplicant/***, am adăugat în fișierul ***wpa_supplicant.conf*** rețeaua la care doresc să se realizeze conectarea:

```
network={
    ssid="NETWORK"
    psk="PASSWORD"
}
```

După salvarea modificărilor făcute se restartează plăcuța pentru a se produce și la nivel de sistem modificările făcute anterior. Din acest moment conectarea la plăcuța Raspberry Pi Zero W se poate face simplu, din terminal, fiind conectați la aceeași rețea de internet.

Un lucru important la acest pas este faptul că putem seta mai multe rețele de internet și să atribuim fiecăreia un anumit grad de prioritate:

```
network={
    ssid="phone_set"
    psk="passwordOne"
    priority=1
}

network={
```

```
    ssid="Home_set"
    psk="passwordTwo"
    priority=2
}
```

Înainte de a instala noile resurse am făcut update și upgrade sistemului de operare pentru a fi sigur de integrarea și funcționalitatea corectă a pachetelor existente deja în sistem și de a le aduce la versiunea cea mai nouă existentă.

Update: `$ sudo apt-get update`

Upgrade: `$ sudo apt-get dist-upgrade`

Pentru a instala Node.js a trebuit într-un prim pas să descărcăm o versiune a acestuia:

```
$ curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
```

Apoi am pornit instalarea:

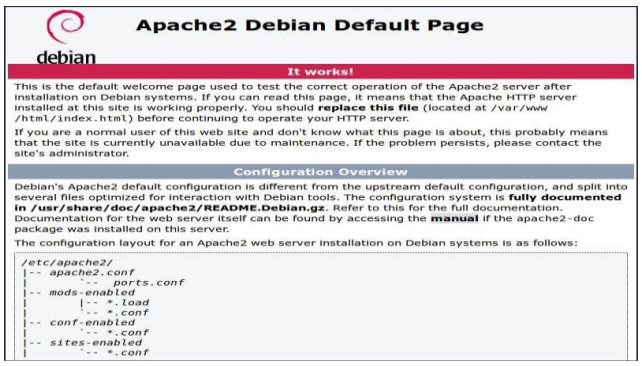
```
$ sudo apt-get install -y nodejs
```

Un lucru opțional pe care l-am putea face pentru a verifica dacă pachetul s-a instalat corect este de a verifica versiunea existentă în sistem folosind comanda ***node –v***.

Pentru a putea folosi plăcuța de dezvoltare Raspberry Pi Zero W ca server web următorul pas care a trebuit făcut a fost instalarea unei versiuni de Apache. Apache a devenit cel mai popular server Web din lume datorită caracteristicilor sofisticate, performanței excelente și a faptului că este gratuit.

```
$ sudo apt-get install apache2 -y
```

După ce am instalat Apache am verificat dacă serverul web a fost creat cu succes.



Figură 3-4 Verificare server Apache

Fișierul sursă pentru pagina web se regăsește în folderul ***/var/www/html***. Am înlocuit codul sursă conținut de acesta cu un cod nou pentru a testa încă o dată funcționalitatea corectă.

```
<html>
<body>
        <h1>Hello World!</h1>
</body>
</html>
```

Rezultatul obținut a fost:

Figură 3-5 Rezultatul paginii web „Hello World!"

3.1.2. Arhitectura interfeței web

După cum am menționat și în primul capitol, crearea unei interfețe ce constă din mai multe pagini web mi s-a părut cea mai potrivită alegere deoarece o interfață web poate fi ușor portată de pe un sistem de operare pe altul fără a necesita modificări semnificative ale codului. De asemenea, nu necesită o instalare a unor resurse software specializate în mod deosebit. Un alt motiv pentru care am optat pentru o interfață web a fost dorințade a asigura funcționarea aplicației cu un minim de efort din partea utilizatorului. În ziua de astăzi majoritatea dintre noi deținem un smartphone, iar folosirea telefonului pentru a interacționa cu dispozitivul prezentat în proiect este un plus adus aplicației. Prin urmare, costurile efectuate în vederea realizării acestui proiect s-au limitat doar la achiziționarea resurselor hardware. De asemenea, am dorit ca utilizatorul să fie întâmpinat de o interfață prietenoasă, intuitivă și ușor de înțeles. Consider că rezultatul final a satisfăcut aceste condiții, iar în continuare voi detalia modul în care a fost gândit fiecare element al interfeței.

Interfața proiectului prezentat este alcătuită din 5 pagini web, fiecăreia dintre aceste pagini fiindu-i atribuită o anumită funcționalitate:

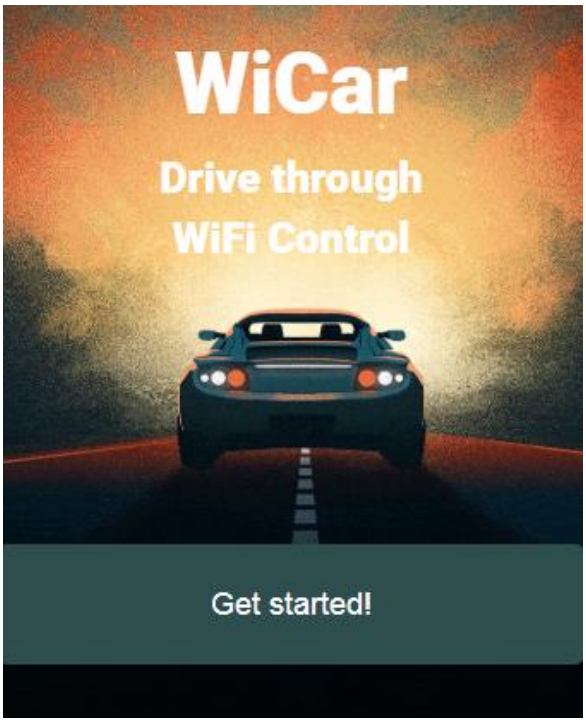
- Prima pagină – întâmpinare și prezentarea scopului aplicației;
- A doua pagină – autentificarea utilizatorului în vederea obținerii accesului la aplicație;
- A treia pagină – posibilitatea înregistrării unui nou utilizator;
- A patra pagină – atenționarea utilizatorului pentru poziționarea landscape a telefonului sau afișarea unei numărătoare descrescătoare dacă telefonul a fost deja poziționat conform cerințelor aplicației;
- A cincea pagină – coordonarea direcției de mers a mașinii.

Acest lucru a ajutat și la organizarea codului, dar și la asigurarea înțelegerii de către utilizator a pașilor care necesită a fi parcurși pentru a ajunge la obiectivul acestei aplicații: coordonarea dispozitivului mobil de la distanță.

Prima pagină este pagina de început căreia nu îi este atribuită altă funcționalitate decât cea de întâmpinare a utilizatorului. Aceasta prezintă pe fundal o imagine de tip gif care accentuează tema aleasă prin reprezentarea unei mașini în mișcare.

Titlul este ales astfel încât să realizeze o sinteză a temei proiectului:

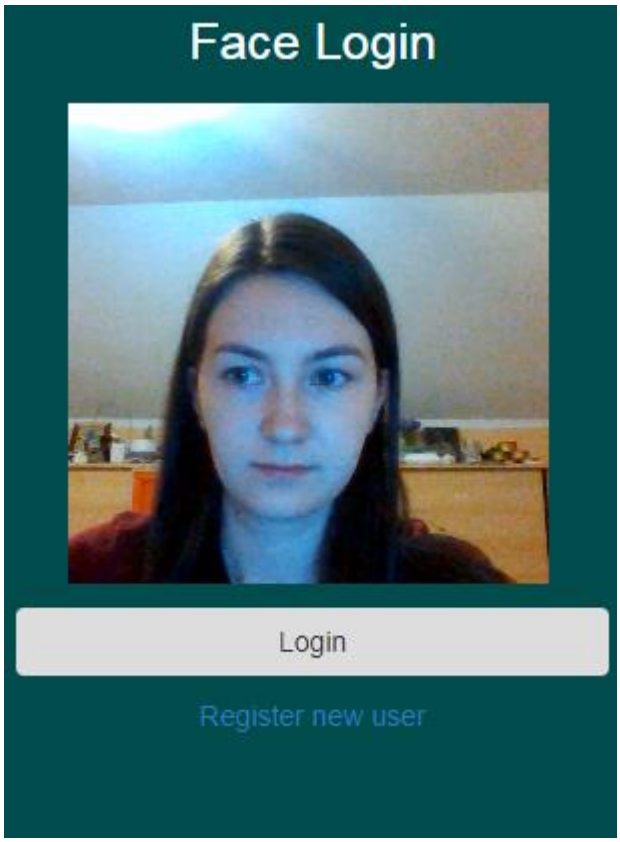
- WiCar – provine de la ideea că dispozitivul utilizat pentru proiect este o mașină, iar „Wi" face trimitere la termenul Wireless (Wi-Fi) specific realizării comunicației fără fir;
- Subtitlul descrie modul de control al mașinii – coordonarea acesteia se realizează prin control Wi-Fi.



Figură 3-6 Pagina 1 a interfeței[19]

Pagina este prevăzută cu un singur buton în partea de jos care permite trecerea la următoarea pagină atunci când utilizatorul consideră că este pregătit să acceseze aplicația propriu-zisă. Am ales să utilizez limba engleză pentru textul prezentat în conținutul paginii deoarece este limba cunoscută și învățată de cei mai mulți utilizatori. Paginile care construiesc interfața aplicației au un design responsive astfel încât la redimensionarea lor aspectul interfeței să păstreze aceeași structură în pagină, elementele din pagină să poată fi vizibile la orice dimensiune a ecranului unui telefon mobil. La accesarea butonului utilizatorul va fi redirecționat pe cea de-a doua pagină.

A doua pagină este cea prin intermediul căreia se decide dacă persoana care utilizează aplicația este un utilizator valid. Titlul sugerează utilizatorului care este scopul afișării acestei pagini. Deoarece autentificarea se realizează pe baza recunoașterii feței a fost nevoie de un container care să captureze imaginile provenite de la camera telefonului. Utilizatorul poate vedea în timp real ce conținut afișează camera.

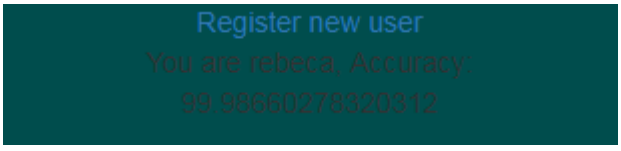


Figură 3-7 Pagina 2 a interfeței

Butonul dispus sub containerul care preia imaginile de la cameră permite verificarea existenței persoanei din imaginea capturată ca utilizator valid.

Următorul element din pagină este un link care va redirecționa spre o pagină asemănătoare ca structură cu pagina actuală care permite înregistrarea unui utilizator.

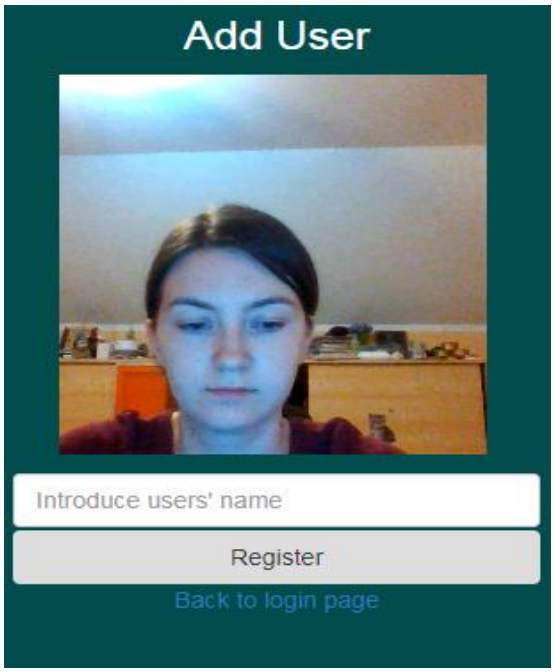
Ultimele elemente nu sunt vizibile inițial. Apariția lor în pagină este determinată de accesul butonului Login, în urma prelucrărilor și analizei efectuate pe imaginea capturată. Conținutul acestor elemente (identificate prin tag-ul) va fi afișat ca în figura de mai jos :



Figură 3-8 Rezultatul autentificării

Este afișat numele utilizatorului și indicele de potrivire între două caracteristici ale fețelor similare. Un alt element care nu poate fi observat vizual este un mesaj vocal redat în urma confirmării existenței unui utilizator.

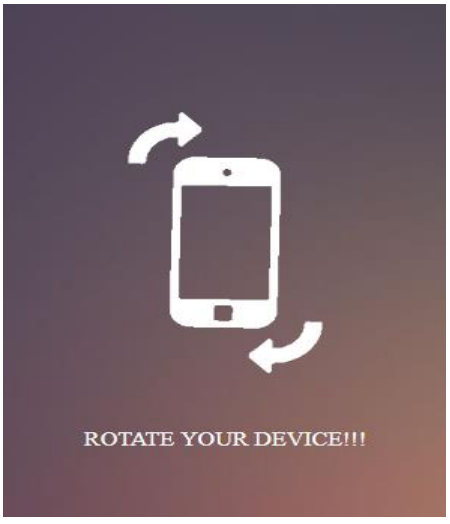
La accesarea link-ului „Register New User” va fi afișată cea de-*a treia pagină*. Aceasta are un conținut similar paginii anterioare.



Figură 3-9 Pagina 3 a interfeței

Elementele noi din această pagină sunt caseta text pentru introducerea numelui utilizatorului care urmează a fi înregistrat și butonul care declanșează adăugarea noului utilizator la colecția de fețe.

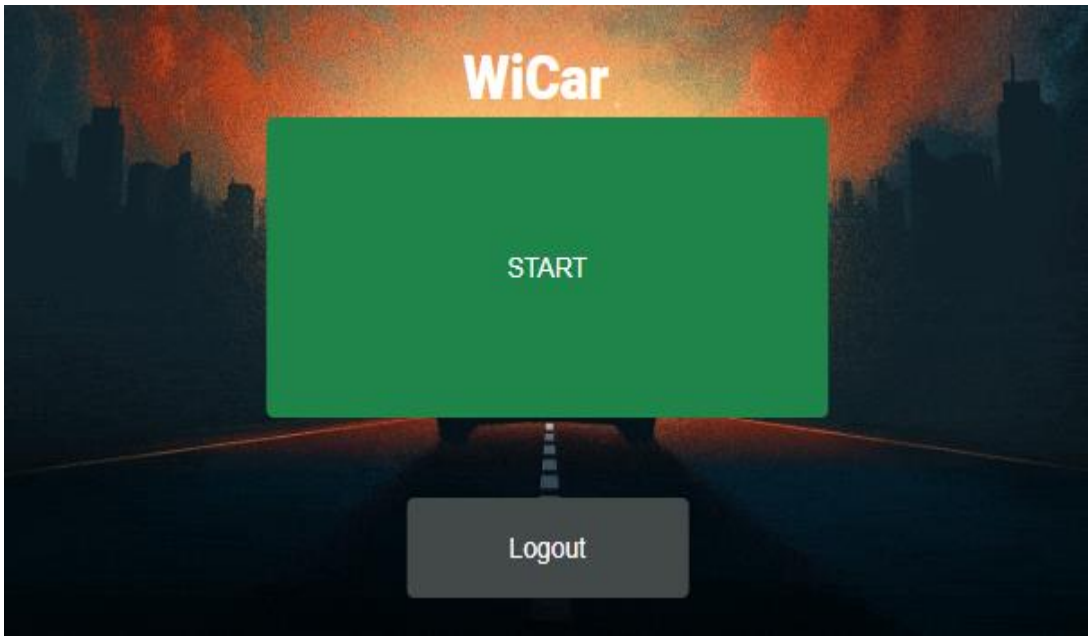
A patra pagină este accesată automat în cazul în care autentificarea din a doua pagină este realizată cu succes. Utilizatorul este redirecționat la pagina în care este specificat modul în care trebuie orientat telefonul pentru a avea rezultatele așteptate în momentul coordonării traiectoriei mașinii. Acest lucru este transmis prin prezentarea unei imagini de tip gif care va prezenta modul în care trebuie rotit telefonul pentru a avea cele mai bune rezultate în utilizarea aplicației. Dacă telefonul este orientat landscape se va declanșa o numărătoare inversă de la 4 până la 0, ceea ce înseamnă că aplicația parcurge etapele normale și avansează în atingerea obiectivelor pentru care a fost dezvoltată. În acest timp utilizatorul se poate pregăti deoarece în câteva secunde va putea „conduce” mașina.



Figură 3-10 Pagina 4 a interfeței (specificare orientare pagină[20])

Figură 3-11 Pagina 4 a interfeței (numărătoare descrescătoare)

Ultima pagină este afișată automat după finalizarea numărătorii descrescătoare din pagina web anterioară, adică după afișarea cifrei 0. Pe această pagină se pot observa două butoane, *Start* și *Logout.* Dacă se apasă butonul *Start*, acesta își va schimba culoarea din verde în roșu și textul în *Stop.*



Figură 3-12 Pagina 5 a interfeței

3.1.2.1.Implementarea interfeței web

În continuare voi prezenta codul sursă care stă la baza funcționării interfeței web. Am văzut în prezentarea anterioară modul în care sunt construite și cum arată fiecare pagină din interfață.

Pentru afișarea și funcționarea primei pagini web am utilizat următorul cod:

```
49      }
50      </style>
51      <script>
52      function pageRedirect() {
53              window.location.replace("http://localhost:4567");
54          }
55      </script>
56      </head>
57      <body>
58
59      <div class="hero-image">
60        <div class="hero-text">
61          <h1 style="font-size:50px">WiCar</h1>
62          <p style ="font-size: 25px">Drive through WiFi Control</p>
63        </div>
64         <button type="button"  class="btn btn-primary btn-lg " onclick="pageRedirect()" id="btn-login">Get started!</button>
65      </div>
66      </body>
67      </html>
68
```

Figură 3-13 Cod sursă index.php

Funcția din interiorul tag-urilor <script></script> este apelată la accesarea butonului *Get started!* Prin urmare, utilizatorul este redirecționat spre cea de-a doua pagină care se găsește la adresa *localhost:4567*. Localhost poate fi înlocuit de un IP local, iar 4567 reprezintă portul deschis pentru ca aplicația să poată rula conform regulilor Sinatra (port standard). Pentru a avea portul deschis este necesar ca înainte de pornirea aplicației să deschidem pagina principala main.rb utilizând comanda *ruby main.rb* în consolă.

Cea de-a doua pagină, pentru autentificare, are la bază următorul cod pentru afișarea paginii:

```
15      <body style="background-color: 004d4d">
16      <div>
17        <h1 style="font-size: 25px; color: white; margin-top: 1px">Face Login</h1>
18        <div id='camera_stuff'>
19          <div id='message'>
20            <p>Change your browser. It does not support a camera!</p>
21          </div>
22        </div>
23        <br>
24        <p>
25          <button type="button" class="btn" id='search_face' style="width: 100%">Login</button>
26        </p>
27        <a href="register">Register new user</a>
28        <div id='state'></div>
29        <div id='confirm'></div>
30        <img height='30' id='wait_image' src='images/wait.gif' width='30'>
31        <audio id='voice' src='#'></audio>
32        <script src="js/jpeg_camera/jpeg_camera_with_dependencies.min.js" type="text/javascript"></script>
33        <script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
34        <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.17.1/moment.min.js"></script>
35        <script src="js/login.js" type="text/javascript"></script>
36        </div>
37      </body>
38      </html>
39
```

Figură 3-14 Cod sursă login.erb

Un cod asemănător este scris și în fișierul register.erb. Elementul identificat prin id-ul *camera_stuff* va conține containerul pentru a vedea imaginile capturate de cameră, iar cel identificat prin id-ul *message* va afișa mesajul respectiv dacă browser-ul pe care îl folosim nu permite utilizarea camerei. Am utilizat div-uri cu id-urile *state* și *confirm* pentru a primi mesaje despre rezultatele comparației fețelor. Linia 31 prezintă un element al cărui conținut nu va fi vizualizat, ci ascultat. Prin intermediul acestuia utilizatorul este salutat în funcție de

momentul din zi și de numele său. Baza funcționării acestor pagini este cuprinsă în fișierele JavaScript.

```
38  // Compare the captured image of the face to our collection from AWS
39  var search_face = function() {
40    var snapshot = camera_stuff.capture();
41    //setting the api_url for the ruby file for comparing faces
42    var api_url = "/search";
43    $("#wait_image").show();
44    snapshot.upload({api_url: api_url}).done(function(response) {
45        //parsing the response received from the ruby file
46        //it contains three properties: message, id and confidence coefficient
47      var data = JSON.parse(response);
48      if (data.id !== undefined) {
49        $("#confirm").html(data.message + " " + data.id + ", Accuracy: " + data.confidence);
50        //response through a vocal message using the name of the matching face
51        $.post("/voice", {tosay: "Good " + timeOfDay(moment()) + " " + data.id}, function(response)
52        {
53            //set properties for the "voice"
54          $("#voice").attr("src", "data:audio/mpeg;base64," + response);
55          $("#voice")[0].play();
56        });
57        //if login is successfull redirect to the next page after 5 seconds
58        setTimeout(function(){
59            $(document).ready( function() {
60              url = "test";
61              $( location ).attr("href", url);
62              });
63        }, 5000);
64
65
66      } else
67      {
68        $("#confirm").html(data.message);
69      }
70      $("#wait_image").hide();
71      this.discard();
72    }).fail(function(status_code, error_message, response)
73    {
74        //in case of fail show the error message
75      $("#state").html("Login failed with status " + status_code + " (" + error_message + ")");
76      $("#confirm").html(response);
77      $("#wait_image").hide();
78    });
79  };
```

Figură 3-15 Cod sursă login.js

Acesta este fișierul login.js corespunzător fișierului login.erb. La linia 40 se creează o variabilă care să stocheze imaginea capturată de cameră. Pentru aceasta m-am folosit de o librărie *JpegCamera*[21]. Setăm api_url cu eticheta */search*. Acest lucru va folosi la transmiterea imaginii capturate, *snapshot*, către fișierul main.erb care va comunica informația primită API-ului Rekognition, pentru a fi comparată cu fețele existente în colecție. La linia 47 se separă răspunsul primit în 3 variabile, iar dacă a fost găsit un id în colecția de fețe al cărui șir de caracteristici să coincidă cu șirul construit pe imaginea capturată de cameră se va afișa mesajul de confirmare cu numele utilizatorului și procentul de asemănare. Dupa 5 secunde utilizatorul va fi redirecționat pe o nouă pagină (linia 61) pentru a ajunge la scopul final al acestei aplicații. În cazul în care răspunsul primit nu conține un id definit în colecție va fi afișat un mesaj de eroare transmis de fișierul main.rb și nu se va mai face redirecționarea pe următoarea pagină.

Fișierul main.rb coordonează toate operațiile care accesează Rekognition API, permite redirecționarea de la o pagină la alta prin intermediul rutelor și întoarce mesaje specifice

operațiilor efectuate către pagina html. Se poate spune și că realizează comunicația între API-ul Rekognition și fișierele JavaScript.

```
2   require 'rubygems'
3   require 'bundler'
4   Bundler.require
5   require 'sinatra'
6
7   # Load up all our secrets
8   Dotenv.load
9
10  # Set up our AWS authentication for all calls in this app
11  Aws.config.update({
12          :region => 'us-east-1',
13          :credentials => Aws::Credentials.new(ENV['AWS_KEY'],ENV['AWS_SECRET'])
14      })
15
16  # Default collection name
17  USER_COLLECTIONS = "user_collection"
18
19  # The routes
20  get '/' do
21    # Show the main index page
22    erb :login
23  end
24
25  #describing actions for searching a matching face in collection
26  post '/search' do
27    content_type :json
28    client = Aws::Rekognition::Client.new()
29    result = client.search_faces_by_image({
30      collection_id: USER_COLLECTIONS,
31      max_faces: 1,
32      face_match_threshold: 95,
33      image: {
34        bytes: request.body.read.to_s
35      }
36    })
37    if result.face_matches.count > 1
38      {:message => "Too many faces found"}.to_json
39    elsif result.face_matches.count == 0
40      {:message => "No face detected!"}.to_json
41    else
42      # "Message for finding a matching face."
43      {:id => result.face_matches[0].face.external_image_id, :confidence => result.face_matches[0].face.confidence, :message => "You are "}.to_json
44
45    end
46  end
47
```

Figură 3-16 Cod sursă main.rb I

În primul rând a fost necesar să anunț la începutul fișierului cu ce voi lucra. Așadar, a fost nevoie *de rubygems*, pachetul care se ocupă de librăriile necesare acestui limbaj și de configurări, de *bundler* care se asigură că ruby găsește toate resursele *gem* necesare în fișierul Gemfile și de *sinatra,* framework-ul cu ajutorul căruia am realizat aplicația.

La linia 8 am specificat fișierul unde se află credențialele pentru a accesa Rekognition API de la Amazon. Nu pot avea acces la el decât prin crearea unui cont. Am încărcat credențialele și am denumit printr-o constantă numele colecției în care vom încărca metadatele despre imaginile fețelor și totodată locul care va fi accesat pentru a realiza comparația (linia 17).

Apoi am început să setez rutele de care va fi nevoie în timpul execuției aplicației. Prima rută este pentru cea de-a doua pagină, adică atunci când în bara pentru link se va afla *localhost:4567* sau *IP:4567* Pagina care se va afișa va fi *login.erb*. Prima pagină a aplicației, cea de întâmpinare, se va afișa tastând doar IP-ul local.

Apoi am setat ruta pentru *search,* adică apelată la accesarea butonului *Login*. Atunci va fi necesar să fie setat clientul pe serviciul Rekognition, iar în *result* va fi întors rezultatul comparației descris în felul următor: caută în colecția cu numele înregistrat la constanta *USER_COLLECTIONS* și găsește o față al cărei indice de potrivire să fie peste 95% în comparație cu datele imaginii (bytes) preluate din pagina web în execuție. Dacă sunt găsite

mai multe fețe respectând condiția, afișează „Too many faces found!”, dacă nu este găsită nicuna, afișează „No face detected!”. Altfel, dacă este găsită exact o față asemănătoate trimite un mesaj JSON la fișierul JavaScript cu următoarele proprietăți: id-ul feței găsite, procentajul de asemănare și mesajul cu numele utilizatorului.

```
48  #Show the register user page
49  get '/register' do
50      erb :register
51  end
52
53  #Set properties for greeting message
54  post '/voice' do
55    client = Aws::Polly::Client.new()
56    result = client.synthesize_speech({
57      output_format: "mp3",
58      voice_id: "Emma",
59      text: params[:tosay]
60    })
61    Base64.encode64(result.audio_stream.string)
62  end
63
64  #Describing actions for adding a face to collection with a given id-name
65  post '/add/:faceid' do
66    client = Aws::Rekognition::Client.new()
67    result = client.index_faces({
68      collection_id: USER_COLLECTIONS,
69      external_image_id: params[:faceid],
70      image: {
71        bytes: request.body.read.to_s
72      }
73    })
74    "User registered!"
75  end
76
77  #Show the page for "driving" the car
78  get '/test' do
79      erb :test
80  end
81
82  #Go back from the regitering user opage to login page
83  get '/back' do
84      erb :login
85  end
86
```

Figură 3-17 Cod sursă main.rb II

Linia 49 setează ruta la care va fi afișată cea de-a treia pagină. La lina 54 este ruta care face posibilă pronunțarea mesajului de întâmpinare la recunoașterea utilizatorului. Clientul este setat de această dată pe serviciul *Polly*, cel responsabil de mesajele vocale, formatul mesajului va fi mp3, vocea va fi cea asociată pentru *Emma* din API-ul Polly, iar textul este cel primit de la fișierul JavaScript *login.js* prin linia 51 prin atributul *tosay*.

Următoarea rută este responsabilă de adăugarea unui utlizator în colecție. Codul este asemănător celui începând cu linia 26, pentru comparație. Apoi sunt setate și rutele pentru afișarea celorlalte pagini din aplicație.

Numele din interiorul rutelor, cuprinse între apostroafe, sunt setate din fișierul JavaScript prin variabila *api_url* sau prin comanda *$.post...* .

Acesta este codul corespunzător celei de-a patra pagini:

```
<style type="text/css">
    #warning-message { display: none; }
    @media only screen and (orientation:portrait){
        #wrapper { display:none; }
        #warning-message { display:block; }
    }
    @media only screen and (orientation:landscape){
        #warning-message { display:none; }
    }
</style>

<html>
<head>


</head>
<body bgcolor="004d4d">

   <div id="wrapper" class="center2">
    <div style="font-size: 200px; color: white"><span id="timer"></span></div>
   </div>
    <div id="warning-message" class="center-div">
    <div>
            <img src="http://cdn.3dvista.com/brookwood/media/rotate.gif" class="center">
    </div>
    <div style="height: 20px;">
            <p align="center" style="color:white;">ROTATE YOUR DEVICE!!!</p>
    </div>
   </div>

   <script>
    document.getElementById('timer').innerHTML = 4;
    startTimer();

    function startTimer() {
        var m = document.getElementById('timer').innerHTML;
        if(m>0 && window.innerHeight < window.innerWidth){m=m-1}
        if(m==0){
            //window.location.href="http://www.google.com";
        }
        document.getElementById('timer').innerHTML = m ;
        setTimeout(startTimer, 1000);
        if(window.innerHeight > window.innerWidth){
                document.getElementById('timer').innerHTML = 4 ;
        }
    }
```

Figură 3-18 Cod sursă go.erb

Se verifică dacă telefonul este orientat landscape prin compararea înălțimii paginii cu cea a lățimii. Dacă este landscape, valoarea lui *m* va fi decrementată cu o unitate, pornind de la 4 până la 0. Acesta este timer-ul creat, iar fiecare valoarea a lui m până la atingerea intervalului inferior va fi afișată pe ecran.

Codul ultimei pagini va fi prezentat în continuare. Este codul care este responsabil de direcționarea mașinii.

```
var  socket = io.connect('192.168.43.170:8080');
var nIntervId;
var start=0;
$(function()
{
    nIntervId = setInterval(SendToPi ,2500);
});

function logout() {
                start=0;
                window.location.href="192.168.43.170";
}
```

Figură 3-19 Cod sursă app.erb I

```
function inc() {
                var elem = document.getElementById("btn-start");
                if (elem.value == "START") elem.value = "STOP";
                else elem.value = "START";
                if (start%2 == 1) {
                        elem.style.backgroundColor = "#A93226"
                        elem.value = "STOP";
                        elem.innerHTML = 'STOP';
                }
                else {
                        elem.style.backgroundColor = "#1E8449"
                        elem.value = "START";
                        elem.innerHTML = 'START';
                }
                start=start+1;

        }

function SendToPi() {
                if(window.DeviceMotionEvent) {
                        window.addEventListener('devicemotion', function(event) {
                                var x = event.accelerationIncludingGravity.x;
                                var y = event.accelerationIncludingGravity.y;
                                var z = event.accelerationIncludingGravity.z;
                socket.emit('fromclient', {start,x,y,z} );
                        });
                }
}
</script>
```

Figură 3-20 Cod sursă app.erb II

Pentru a controla dispozitivul mobil s-a utilizat sezorul de accelerometru de la smartphone. Drumul parcurs de informație de la senzor la server a fost acesta: în primul rând am scris o funcție *RaspberryPi*, care, printr-un event de tip *window.DeviceMotionEvent* preia cele trei valori de la senzor în trei variabile *x*, *y* și *z*. Funția *RaspberryPi* se apelează la fiecare sfert de secundă, iar valorile *x*, *y* și *z* sunt transmise mai departe folosind un web socket către plăcuța RaspberryPi Zero W. Odată cu aceste trei valori se transmite și o valoare *start* ce are rolul să pună în funcțiune sau să oprească dispozitivul. Valoarea variabilei *start* se incrementează la fiecare apăsare a butonului *START/STOP*.

```
//declare required modules
var app = require('http').createServer(handler)
  , io = require('socket.io').listen(app)
  , fs = require('fs')
  , static = require('node-static')
  , sys = require('util')
  , sleep = require('sleep-ms')
  app.listen(8080);
var Gpio = require('onoff').Gpio;
var FW = new Gpio(19, 'out');
var RET = new Gpio(26, 'out');
var LEFT = new Gpio(6, 'out');
var RIGHT = new Gpio(13, 'out');

//Make a web server on port 8080
//
var file = new(static.Server)();
function handler(request, response) {
  console.log('serving file',request.url)
  file.serve(request, response);
};
console.log('Pi Car we server listening on port 8080 visit http://licenta.local$
lastAction = "";
//If we lose comms set the servos to neutral //
function emergencyStop() {
        //enter 0 point here specific to your pwm control
        FW.writeSync(0);
        RET.writeSync(0);
        console.log('###EMERGENCY STOP - signal lost or shutting down');
}//END emergencyStop
// fire up a web socket server isten to cmds from the phone and set pwm
// accordingly, if using a separate battery pack then disable the
// motor acceleration rate limiting algorithm as this is required when the
// Pi and motors share the same battery.
//
io.sockets.on('connection', function (socket) {
        //got phone msg
        socket.on('fromclient', function (data)
        {
// rate limit accelration only
                // we want normal de-cceleration as this doesn't drain the batt$
          if(data.start%2==0){
              if(data.x < 5) //fwd accel
              {
                      //console.log('inainte');
                      FW.writeSync(1);
                      RET.writeSync(0);
                      FW.writeSync(1);
                      RET.writeSync(0);
                  }else if(data.z < 5) //fwd accel
                  {
//console.log('inapoi');
                      FW.writeSync(0);
                      RET.writeSync(1);

                  }else{
                      //console.log('stop');
                      FW.writeSync(0);
                      RET.writeSync(0);
                  }
                  if(data.y < -2.5)
{
//console.log('stanga');
                      LEFT.writeSync(1);
                      RIGHT.writeSync(0);
                  }else if(data.y > 2.5)
                  {
                      //console.log('dreapta');
                      LEFT.writeSync(0);
                      RIGHT.writeSync(1);
                  }else
{
                      //console.log('nimic');
                      LEFT.writeSync(0);
                      RIGHT.writeSync(0);
                  }

                  clearInterval(lastAction); //stop emergency stop timer
                  lastAction = setInterval(emergencyStop,2000); //set emergency s$
              }else{
                      LEFT.writeSync(0);
RIGHT.writeSync(0);
                      FW.writeSync(0);
                      RET.writeSync(0);
                  }
          });
});//END io.sockets.on
//user hits ctrl+c
//
process.on('SIGINT', function() {
  emergencyStop();
  console.log("\nGracefully shutting down from SIGINT (Ctrl-C)");

  return process.exit();
});//END process.on
```

Figură 3-21 Cod sursă app.js

După cum am zis anterior, valorile sunt transmise printr-un socket prin portul *8080* către plăcuță unde sunt recepționate și prelucrate în scriptul *app.js* ce rulează pe server. Scriptul *app.js* este scris in Node.js. Folosind librăria *onoff* am creat 4 variabile cărora le-am asignat câte un pin de la plăcuța Raspberry PiZero W în felul următor:

```
var Gpio = require('onoff').Gpio;
var FW = new Gpio(19, 'out');          //forward
var RET = new Gpio(26, 'out');         //return
var LEFT = new Gpio(6, 'out');         //left
var RIGHT = new Gpio(13, 'out');       //right
```

În funcție de informațiile transmise prin socket s-au realizat următoarele acțiuni:

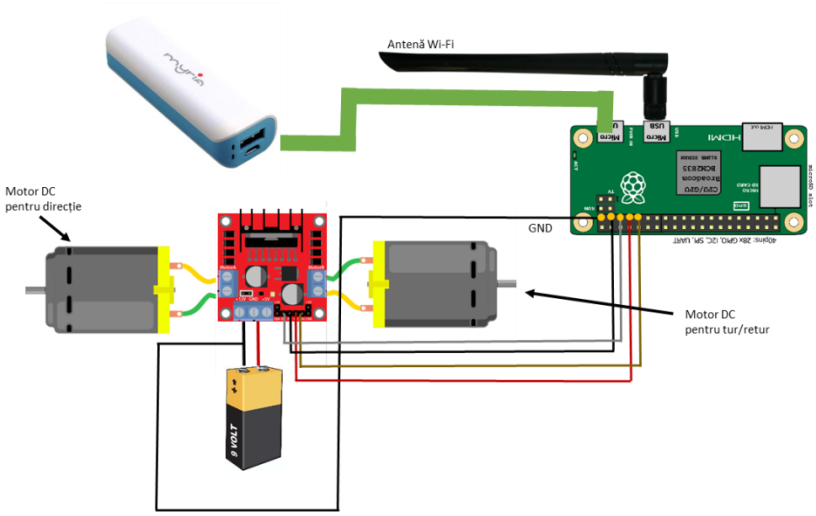
- În cazul în care valoarea lui x era mai mică decât 5 atunci va fi setat 1 pe pinul 19 și 0 pe pinul 26, acest lucru fiind echivalent cu deplasarea înainte.
- Altfel, în cazul în care z<5, atunci valorile semnalelor celor doi pini se inversează, pinul 19 fiind setat pe 0 și pinul 26 pe 1, lucru ce realizează deplasarea înapoi.
- În cazul în care nici una din condițiile anterioare nu este îndeplinită, dispozitivul rămâne în repaus până primește o nouă comandă. Acest lucru face ca pinii 19 și 16 să se întoarcă la starea inițială fiind setați pe 0.
- Direcția de deplasare se face în funcție de variabila y în felul următor: dacă y<-2.5, dispozitivul se deplasează spre stânga (pinul 6 este setat pe 1, iar pinul 13 pe 0), altfel, dacă y>2.5, dispozitivul se deplasează spre dreapta (pinul 6 este setat pe 0, pinul 13 pe 1), altfel dispozitivul se deplasează înainte (atât pinul 6, cât și 13 sunt setați pe 0).
- Însă niciuna dintre aceste acțiuni nu are loc dacă nu este îndeplinită condiția ca *start%2=0*. Este un mecanism simplu de pornire/oprire a dispozitivului ce se bazează pe restul împărțirii la 2 a unei variabile ce se modifică în timp. Dispozitivul nu se pune în mișcare decât în momentul în care butonul *START/STOP* a fost apăsat de un număr impar de ori.

Tot în funcție de valoarea variabilei start se modifică textul și culoarea butonului cu id-ul *button-start*.

Pagina *app.erb* conține o porțiune de cod CSS, porțiunea scriptului ce conține funcții și porțiunea codului scris în HTML.

Pe lângă butonul de *START/STOP*, pagina mai conține un buton de *LOGOUT* care are rolul de a redirecționa utilizatorul la prima pagină de *LOGIN*.

3.2. Proiectarea și implementarea hardware



Figură 3-22 Schema circuitului proiectului

a) Șasiu

Am început proiectarea hardware prin alegerea unei platforme sau șasiu care va fi folosită ulterior pentru așezarea componentelor hardware folosite.

Pentru acest proiect am ales să folosesc un șasiu de la o mașină RC, acest lucru oferind o mobilitatea mai bună dispozitivului întrucât partea mecanică conținând motoarele DC pentru direcție și deplasare era implementată.

Inițial am dorit să construiesc cu blocuri mecano metalice, dar pentru a face șasiul mai ușor și mai aerodinamic și datorită limitărilor hardware, ulterior, am decis folosirea unui șasiu provenit de la o mașinuță RC de la care am îndepărtat partea de control astfel încât să pot implementa ulterior sistemul propriu de control.

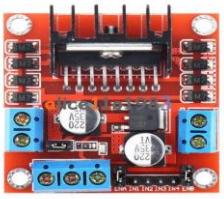
Faptul că șasiul este în totalitate din plastic scade probabilitatea creării unui scurt-circuit între componentele sistemului.

Conexiunile între componentele hardware au fost realizate prin cabluri de tip jumper.

b) Putere de deplasare și direcție

Pentru a da posibilitatea dispozitivului de a se deplasa am folosit cele două motoare DC incorporate în șasiu. Aceste motoare de curent continuu cu perii, au un consum redus de curent și oferă putere și viteză comparabile cu un servomecanism.

Controlul motoarelor s-a realizat folosind un driver și anume modelul L298N[22].



Figură 3-23 Driver L298N

Folosirea unui astfel de model are mai multe avantaje.

În primul rând prețul mic.

În al doilea rând acesta este dispus cu un limitator de tensiune liniar, astfel că în cazul unei alimentări cu tensiune mai mare de 7V nu este necesară alimentarea separată a părții logice.

Cu ajutorul acestui driver este posibil controlul ambelor motoare, atât a celui folosit pentru deplasare, cât și al celui folosit pentru direcție.

Specificații:
- Tensiune motoare: 5-35 V
- Tensiune circuite logice 5V
- Curent motoare: 2A (maxim);
- Curent logică: 36mA;
- Frecvență maximă PWM: 40kHz;
- Dimensiuni: 43 x 43 x 27 mm;

De menționat faptul că L298N poate fi folosit și pentru motoare pas cu pas.

Deși dimensiunea acestuia este mai mare comparativ cu alte drivere, radiatorul cu care este prevăzut ajută prin faptul că disipă o cantitate mare de caldură.

Conexiuni:

- Out 1: terminal motor A
- Out 2: terminal motor A
- Out 3: terminal motor B
- Out 4: terminal motor B
- 5V: 5 V input (dacă sursa folosită este de 7-35 V poate fi folosit ca 5V)
- EnA: PWM pentru motorul A – controlul turației
- EnB: PWM pentru motorul B – controlul turației
- In1: direcție rotație motor A
- In2: direcție rotație motor A
- In3: direcție rotație motor B
- In4: direcție rotație motor B

În dezvoltarea acestui proiect pinii EnA și EnB au fost setați pe enable deoarece nu ne interesează controlul turației celor două motoare DC.

Pentru a controla sensul deplasării setăm In1 pe HIGH, iar In2 îl setăm LOW și dispozitivul se va deplasa înainte. Pentru a inversa sensul setăm LOW, respectiv HIGH pe In1, In2.

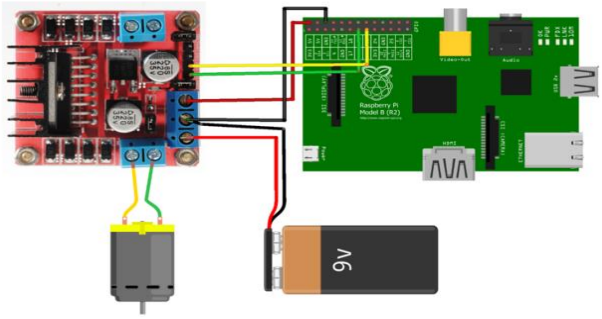
Pentru a controla direcția setăm In3 pe HIGH, iar I4 îl setăm pe LOW și dispozitivul va vira la dreapta. Pentru a vira la stânga setăm LOW, respectiv HIGH pe In3, In4.

Un lucru important de menționat este faptul că GND-ul driverului trebuie conectat cu un pin de GND de la plăcuța Raspberry Pi Zero W.

c) Sursa de curent

Dispozitivul mobil creat are nevoie de 2 surse de curent independente: o sursă de curent ce alimentează driverul L298N folosit pentru controlul motoarelor DC și o sursă de curent pentru alimentarea plăcuței Raspberry Pi Zero W.

Driverul L298N va fi alimentat cu o baterie de 9V ca în figura de mai jos:



Figură 3-24 Circuit cu surse de curent și driver

Alimentarea plăcuței Raspberry Pi Zero W se realizează de la o baterie externă Myria[23].

Specificații:
- Intrare: 5V;
- Ieșire: 5V;
- Capacitate: 2000 mAh;

Figură 3-25 Baterie externă Myria

d) Antenă Wi-Fi

Scopul folosirii unei astfel de antene[24] a fost creșterea semnalului wireless, astfel creștem viteza de transmitere a informației și vom avea un răspuns mai rapid venit din partea dispozitivului mobil.

Specificații:

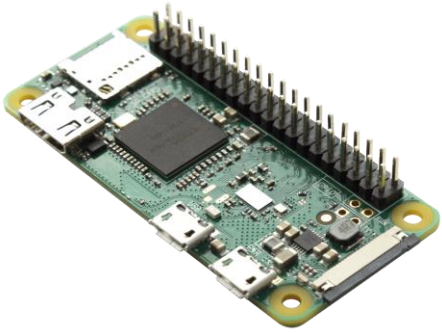
- Frecvență: 2.4G
- Protocol: 802.11n
- Viteza de transmisie: 600 Mbps
- Standard: IEEE 802.11n, IEEE 802.11g, IEEE 802.11b
- Frecvență bandă: 2.4GHz

Figură 3-26 Antenă Wi-Fi

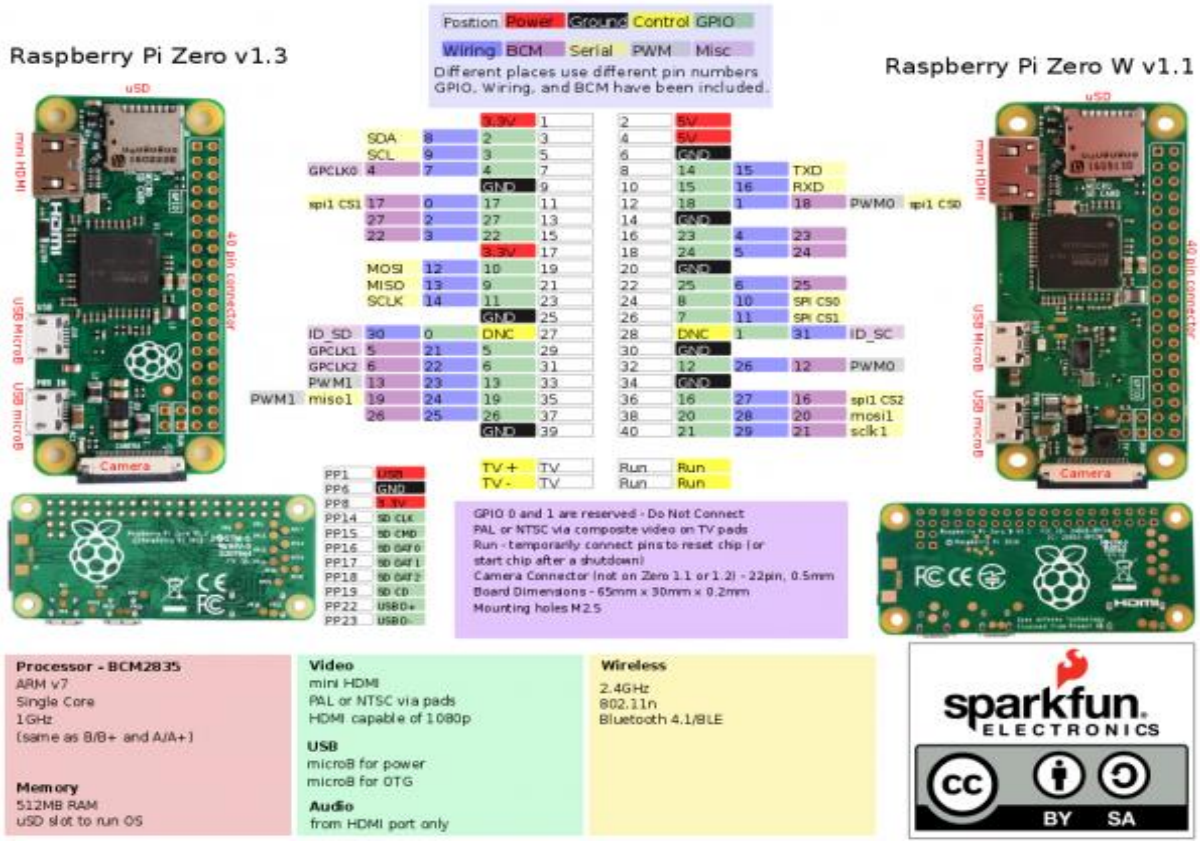
3.2.1. Arhitectura Raspberry Pi Zero W

Raspberry Pi Zero W[25] este unul dintre cele mai mici calculatoare la ora actuală având o configurație ce permite rularea cu ușurință de distribuții Linux și putând procesa algoritmi din cei mai complecși, însă este apropiat și de hardware-ul low level.

Specificații:
- Procesor SoC BCM2835;
- Frecvență operare procesor: 1 GHz;
- Memorie RAM: 512MB;
- Conectivitate WiFi 2.4GHz 802.11 b/g/n;
- Conectivitate Bluetooth 4.1 + HS Low Energy;
- Chip WiFi/Bluetooth: BCM43438;
- Slot card microSD;
- Conector miniHDMI;
- Conector micro-B USB pentru alimentare;
- Conector micro-B USB pentru date;
- Conector CSI pentru cameră foto/video;
- Pinout de 40 pini GPIO;
- Compatibil cu extensiile HAT/pHAT;
- Dimensiuni: 65 x 30 x 5 mm.



Figură 3-27 Raspberry Pi Zero W



Figură 3-28 Descriere arhitectură Raspberry Pi Zero W

Sursă Figură 3-27[26]

Descrierea pinilor[27]:

GND

Pinii de **GND** de pe Raspberry Pi sunt conectați electric, deci nu contează pe care îl utilizăm când conectăm o sursă de tensiune. În general, se utilizează pinul GND cel mai apropiat de ceilalți pini utilizați.

De exemplu, este o idee bună să utilizăm pinul 17 pentru 3v3 și pinul 25 pentru masă atunci când utilizăm conexiunile SPI, deoarece acestea sunt chiar lângă pinii importanți pentru SPI0.

Raspberry Pi Zero W este prevăzut cu 8 pini de **GND** și anume 6, 9, 14, 20, 25, 30, 34 și 39.

DPI - Display Parallel Interface

Este o interfață paralelă pe 24 de biți, cu 28 de semnale de clock și semnale de sincronizare.

Această interfață permite afișarea paralelă RGB să fie atașată la Raspberry Pi GPIO fie în RGB24 (8 biți pentru roșu,verde și albastru) sau RGB666 (6 biți pentru fiecare culoare) sau RGB565 (5 biți de roșu, 6 verzi și 5 albaștri).

Cei 28 de pini care pot fi folosiți pentru unul dintre modurile de afișare enumerate mai sus sunt: 3, 5, 7, 8, 10, 11, 12, 13, 15, 16, 18, 19, 21, 22, 23, 24, 27, 28, 29, 31, 32, 33, 35, 36, 37, 38 și 40.

GPCLK - General Purpose CLock

Un pin General Purpose CLock poate fi setat pentru a scoate o frecvență fixă fără un control software continuu.

Alte frecvențe pot fi obținute prin setarea unui divizor de clock sub forma SOURCE/(DIV_I + DIV_F/4096).

Pinii plăcuței Raspberry Pi Zero W care pot fi setați astfel sunt 3 la număr (7, 29, 31).

JTAG - Joint Test Action Group

JTAG este o interfață standardizată pentru a face debugging circuitelor integrate pe care le putem utiliza pentru a face debug plăcuței Raspberry Pi.

Raspberry Pi Zero W este prevăzută cu 11 pini de acest tip și anume 7, 13, 15, 16, 18, 22, 29, 31, 32, 33 și 37.

W1-GPIO - One-Wire Interface

Pentru a activa one-wire interface, trebuie adăugată următoarea linie în ***/boot/config.txt***, înainte de repornirea plăcuței Raspberry Pi Zero W: ***dtoverlay=w1-gpio*** sau ***dtoverlay=w1-gpio,gpiopin=x*** dacă se dorește să se utilizeze un PIN personalizat (default este setat BCM4).

Alternativ, se poate activa one-wire interface utilizând **raspi-config** sau următoarele comenzi: ***sudo modprobe w1-gpio.***

Noile kernel-uri (4.9.28 și versiuni ulterioare) ne permit să utilizăm în schimb încărcarea suprapusă dinamică, incluzând crearea mai multor bus-uri de tip 1-Wire pentru a fi utilizate în același timp:

sudo dtoverlay w1-gpio gpiopin=4 pullup=0 # header pin 7
sudo dtoverlay w1-gpio gpiopin=17 pullup=0 # header pin 11
sudo dtoverlay w1-gpio gpiopin=27 pullup=0 # header pin 13

PCM - Pulse-code Modulation

PCM (Pulse-Code Modulation) este o reprezentare digitală a analogului eșantionat. Pe Raspberry Pi este o formă de ieșire audio digitală care poate fi înțeleasă de un DAC pentru sunet de înaltă calitate.

Pini: 12, 35, 38 și 40.

SDIO - SD Card Interface

SDIO este interfața SD host / eMMC de pe Raspberry Pi. Semnalele host SD sunt utilizate în mod normal pentru slotul microSD.

Pinii folosiți în acest scop sunt 13, 15, 16, 18, 22 și 37.

I2C - Inter Integrated Circuit

Pinii I2C ai Raspberry Pi sunt o modalitate extrem de folositoare de a comunica cu diferite tipuri de periferice externe. Pinii I2C includ o rezistență fixă de 1.8 kohms setată la 3.3V. Aceasta înseamnă că nu sunt potrivite pentru utilizarea în scopuri generale IO.

Se poate verifica adresa perifericelor I2C conectate printr-o singură linie:

sudo apt-get install i2c-tools
sudo i2cdetect -y 1

Se poate accesa I2C din Python folosind biblioteca smbus:

```
import smbus
DEVICE_BUS = 1
DEVICE_ADDR = 0x15
bus = smbus.SMBus(DEVICE_BUS)
bus.write_byte_data(DEVICE_ADDR, 0x00, 0x01)
```

Pini folosiți: 3, 5, 27, 28.

SPI - Serial Peripheral Interface

Cunoscut sub numele de magistrală serială cu patru fire, SPI permite conectarea mai multor dispozitive compatibile la un singur set de pini, atribuindu-le diferite pini selectați de chip.

Pentru a comunica cu un dispozitiv SPI, se selectează pinul corespunzător. Implicit, Raspberry Pi Zero W are CE0 și CE1.

```
import spidev
spi = spidev.SpiDev()
spi.open(0, CHIP_SELECT_0_OR_1)
spi.max_speed_hz = 1000000
spi.xfer([value_8bit])
```

Pini utilizați: 11, 12, 19, 21, 23, 24, 26, 35, 36, 38, 40.

UART - Universal Asynchronous Receiver/Transmitter

UART este un protocol de comunicație serială asincronă, ceea ce înseamnă că are octeți de date și transmite biți individuali într-un mod secvențial.

Transmisia asincronă permite transmisia datelor fără ca expeditorul să trimită un semnal de ceas către receptor. În schimb, expeditorul și receptorul sunt de acord asupra

parametrilor de sincronizare în avans și se adaugă biți speciali numiți "biți de pornire" fiecărui cuvânt și utilizați pentru sincronizarea unităților de trimitere și recepție.

UART este utilizat în mod obișnuit pe Raspberry Pi ca o modalitate convenabilă de a controla GPIO-ul sau de a accesa mesajele de boot a kernel-ului din consolă (activată implicit).

Pinii folosiți în acest scop sunt 8 și 10.

WiringPi

WiringPi este o încercare de a aduce simplitatea Arduino-wiring-like și la Raspberry Pi.

Scopul este de a avea o singură platformă comună și un set de funcții pentru accesarea Raspberry Pi GPIO în mai multe limbaje.

WiringPi este o librărie C la bază, dar este disponibilă și pentru utilizatorii Ruby sau Python care pot instala librăria folosind "gem install wiringpi", respectiv "pip install wiringpi2".

Librăria WiringPi2-Python aduce o întreagă serie de funcționalități WiringPi pentru Python, inclusiv caracteristici noi de la WiringPi 2. WiringPi utilizează propria schemă de numerotare a pinilor plăcuței Raspberry Pi Zero W.

Instalarea pentru Python se face foarte ușor efectuând comanda:

sudo pip install wiringpi2

3.2.2. Legătura între componente

Realizarea legăturilor între componentele descrise mai sus și care alcătuiesc partea hardware au fost realizare astfel.

Am menționat mai sus faptul că șasiul dispozitivului a fost luat de la o mașinuță RC de la care a fost scos circuitul ce realiza controlul și interpretarea comenzile primite de la telecomandă. Mașinuța era alimentată inițial cu 4 baterii de 1.5 V care au fost înlocuite ulterior cu o baterie de 9V. Pentru a putea avea control asupra celor două motoare DC acestea au fost conectate la driver-ul L298N astfel: pentru motorul DC folosit pentru deplasare s-au folosit pinii OUT 1 și OUT 2, iar pentru motorul DC folosit pentru direcție s-au folosit pinii OUT 3 și OUT 4.

Alimentarea driver-ului s-a realizat folosind o baterie de 9V , borna pozitivă a bateriei a fost conectată la slotul de 12V al driver-ului L298N, iar borna negativă a fost conectată la GND-ul driver-ului și la pinul GND al plăcuței Raspberry Pi Zero W numerotat cu 39. Este obligatoriu conectarea bateriei și la un pin GND de la plăcuța Raspberry Pi Zero W pentru ca sistemul să poată funcționa.

După acest pas a trebuit să creăm o legătură între driver-ul L298N și plăcuța Raspberry Pi Zero W. În acest scop am folosit patru pini de la plăcuță în modul următor:

Pentru semnalul de control al motorului pentru deplasare am conectat pinul IN 1 de la driver-ul L298N la pinul GPIO19 de la plăcuța Raspberry Pi Zero W și pinul IN 2 la pinul GPIO26.

Pentru semnalul de control al motorului pentru direcție am conectat IN 3 de la driver-ul L298N la pinul GPIO6 și pinul IN 4 la pinul GPIO13 de la plăcuța Raspberry Pi Zero W.

Pinii ENA și ENB de la driver-ul L298N i-am setat pe enable deoarece nu ne interesează momentan un control al turației motoarelor. Însă, folosind acești pini, s-ar putea implementa un algoritm pentru controlul vitezei dispozitivului mobil.

Pentru alimentarea plăcuței Raspberry Pi Zero W s-a folosit o baterie externă Myria, iar conexiunea dintre acestea s-a realizat folosind un cablu Micro Usb.

Cu scopul de a crește semnalul wireless s-a folosit și o antenă care s-a conectat la plăcuță prin portul micro usb și folosind, ca și în cazul alimentării, un cablu Micro Usb.

Cardul microSD folosit pe care s-a instalat anterior sistemului și care este folosit și ca memorie internă a fost introdus în slotul pentru card microSD cu care este prevăzut plăcuța Raspberry Pi Zero W .

Pentru răcirea procesorului plăcuței Raspberry Pi Zero W am folosit un cooler pe care l-am conectat la baterie.

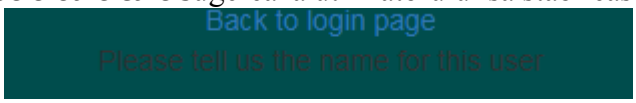
Deoarece am vrut ca bateria să nu se consume pe perioada în care dispozitivul nu este utilizat am folosit un întrerupător.

4. Testarea aplicației și rezultate experimentale

După scrierea codului este de dorit ca aplicația să fie testată pentru a observa modul în care răspunde la anumite acțiuni și pentru a verifica dacă funcționează conform așteptărilor.

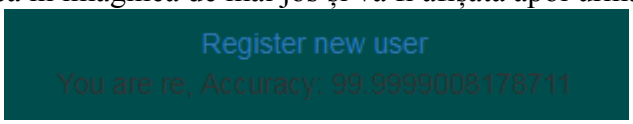
Inițial am dorit să testez partea de autentificare în aplicație.

Dacă doresc să creez un cont, dar nu am ales un nume pentru utilizator, este afișat un mesaj de eroare care sugerează utilizatorului să stabilească un nume.



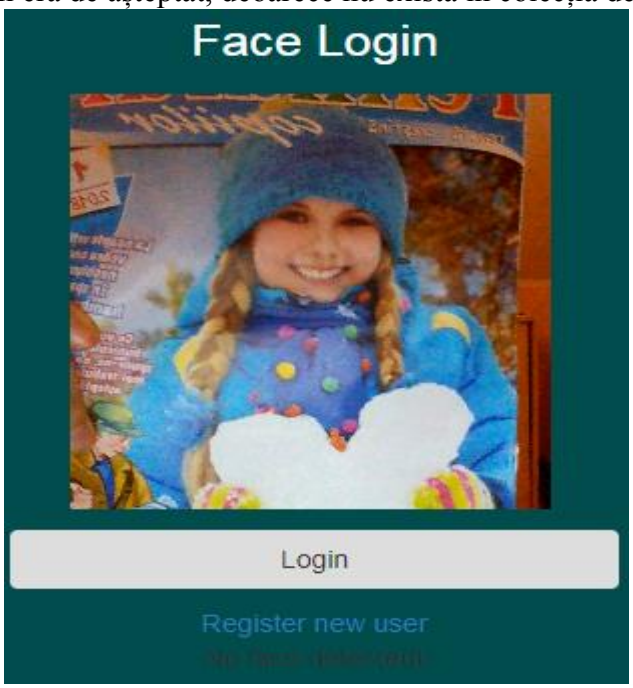
Figură 4-1 Test nume utilizator lipsă

Dacă doresc să accesez aplicația și am asociat un utilizator existent, va apărea un mesaj ca în imaginea de mai jos și va fi afișată apoi următoarea pagină.



Figură 4-2 Test utilizator recunoscut

În momentul în care am pus în fața camerei o față necunoscută, autentificarea a eșuat, așa cum era de așteptat, deoarece nu exista în colecția de fețe.



Figură 4-3 Test utilizator necunoscut

Un pas important a fost configurarea și setarea conexiunii la Wi-Fi. Acest lucru a fost realizat modificând fișierul wpa_supplicant.conf. După realizarea acestot setări conexiunea la plăcuță am realizat-o folosind PuTTy.

Pentru a putea folosi plăcuța de dezvoltare Raspberry Pi Zero W ca server următorul pas care a trebuit făcut a fost instalarea unei versiuni de Apache. După ce am instalat Apache am verificat dacă serverul web a fost creat cu succes. (Figura 3-4)

Fișierul sursă pentru pagina web se regăsește în folderul /var/www/html. Am înlocuit codul sursă conținut de acesta cu un cod nou pentru a testa încă o dată funcționalitatea corectă.

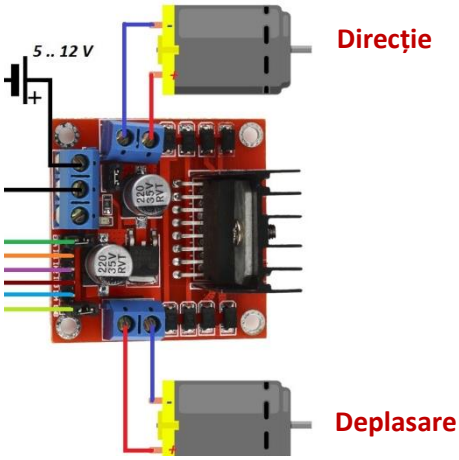
<html>
<body>
<h1>Hello World!</h1>
</body>
</html>

Rezultatul obținut a fost cel din Figura 3-5.

Pentru partea mecanică și anume puterea de deplasare și direcție am folosit șasiul de la mașinuță RC de la care am înlăturat vechiul circuit de control. Pentru deplasare și direcție este folosit câte un motor DC de 4V. Pentru controlul acestora am folosit un driver L298N.

Specificații:

- Tensiune motoare: 5-35 V
- Tensiune circuite logice 5V
- Curent motoare: 2A (maxim);
- Curent logică: 36mA;
- Frecvență maximă PWM: 40kHz;
- Dimensiuni: 43 x 43 x 27 mm;



Figură 4-4 Circuit de testare cu descrierea motoarelor

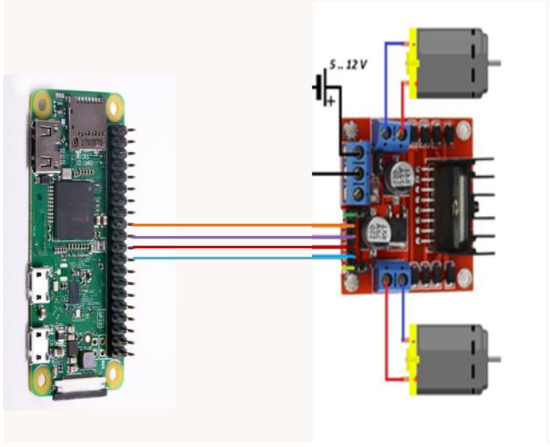
Pentru controlul fiecărui motor s-au folosit câte doi pini, iar alimentarea driverului L298N s-a făcut cu o baterie de 9V.

A urmat scrierea unui script pentru a testa controlul motoarelor.

Pentru fiecare din cele două motoare s-au folosit doi pini de la Raspberry Pi Zero W. Pinii de la driver de PWM au fost setati pe enable deoarece într-o primă fază nu ne vom dori controlul turației motorului de deplasare și nici a celui de direcție.

Scriptul de testare a fost scris în Python:

```
1   import RPi.GPIO as GPIO
2   import time
3   GPIO.setmode(GPIO.BCM)
4   GPIO.setwarnings(False)
5
6   #Run setup
7   GPIO.setup(18,GPIO.OUT)
8   GPIO.setup(19,GPIO.OUT)
9   print "Forward"
10  GPIO.output(18,GPIO.HIGH)
11  GPIO.output(19,GPIO.LOW)
12  time.sleep(2)
13  print "Return"
14  GPIO.output(18,GPIO.LOW)
15  GPIO.output(19,GPIO.HIGH)
16  time.sleep(2)
17  print "Stop"
18  GPIO.output(18,GPIO.LOW)
19  GPIO.output(19,GPIO.LOW)
20
21  #Direction setup
22  GPIO.setup(20,GPIO.OUT)
23  GPIO.setup(21,GPIO.OUT)
24  print "Forward"
25  GPIO.output(20,GPIO.HIGH)
26  GPIO.output(21,GPIO.LOW)
27  time.sleep(2)
28  print "Return"
29  GPIO.output(20,GPIO.LOW)
30  GPIO.output(21,GPIO.HIGH)
31  time.sleep(2)
32  print "Stop"
33  GPIO.output(20,GPIO.LOW)
34  GPIO.output(21,GPIO.LOW)
```

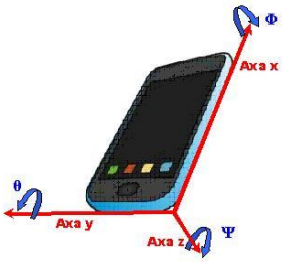


Figură 4-5 Script Python pentru testare și ciruit testare complet

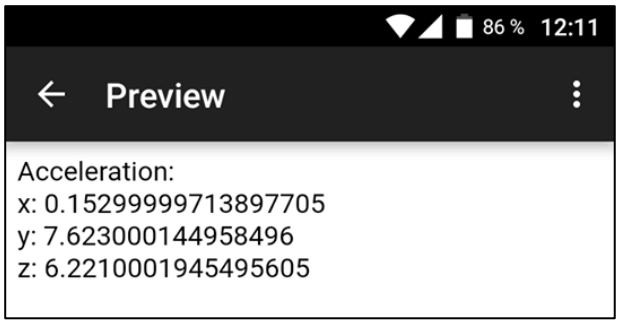
Controlul sistemului mobil va avea loc prin Wi-Fi, astfel pentru o mai bună securitate mi-am propus crearea unei aplicații de conectare bazată pe recunoaștere facială. În acest sens am folosit Rekognition API pe care l-am prezentat în capitolele anterioare.

Pentru controlul mecanismului de deplasare am folosit accelerometrul de la telefon. Pentru aceasta a trebui să studiez modalitatea de funcționare a senzorului și apoi să testez printr-o aplicație simplă funcționalitatea acestuia.

```
<DOCTYPE html>
<html>
<head>
   <script>
   function init() {
     var dataContainerMotion = document.geteElementById('dataContainerMotion');
     // Check for support for DeviceMotion events
     if(window.DeviceMotionEvent) {
     window.addEventListener('devicemotion', function(event) {
               var x = event.accelerationIncludingGravity.x;
               var y = event.accelerationIncludingGravity.y;
               var z = event.accelerationIncludingGravity.z;
               var r = event.rotationRate;
               var html = 'Acceleration:<br />';
               html += 'x: ' + x +'<br />y: ' + y + '<br/>z: ' + z+ '<br />';
               dataContainerMotion.innerHTML = html;
            });
     }
   }
  </script>
  </head>
  <body onload="init()">
    <div id="dataContainerMotion">
      No device motion data
    </div>
  </body>
</html>
```



Figură 4-6 Cod sursă pentru afișarea coordonatelor X, Y și Z și modul de orientarea al axelor



Figură 4-7 Rezultat testare pentru afișarea coordonatelor

5. Manual de utilizare

În acest capitol voi descrie pașii care trebuie parcurși pentru a rula aplicația cu succes.

În primul rând se vor deschide două terminale PuTTy, iar la Hostname vom scrie pi@licenta.local, adică numele cu care identific plăcuța Raspberry Pi Zero W.

Pentru ambele terminale vom folosi comanda *cd /var/www/html* pentru a ajunge în directorul care conține fișierele necesare rulării aplicației.

În primul terminal vom scrie comanda *ruby main.rb* pentru a deschide portul 4567, portul implicit pentru ca Sinatra să poată permite afișarea și buna funcționare a paginilor web.

În al doilea terminal vom scrie comanda *node app.js* care permite primirea și prelucrarea valorilor provenite de la accelerometrul telefonului pentru a direcționa dispozitivul mobil.

Este necesar apoi să deschidem o fereastră Mozilla Firefox, iar în bara de link-uri să scriem IP-ul local. Acesta se găsește tastând în consolă comanda *ping licenta.local*.

În acest moment ar trebui să apară pagina web din Figura 3-6, apoi putem naviga de pe o pagină pe cealaltă cu ajutorul butoanelor dispuse în interfața web.

Click pe butonul *Get started!* va afișa cea de-a doua pagină. Trebuie ca utilizatorul să se vadă în conținutul paginii web în spațiul destinat afișării „camerei”, iar apoi poate da click pe butonul *Login.*

Dacă nu este un utilizator înregistrat, este necesar să acceseze link-ul de sub butonul *Login*, cel pe care scrie *Register new user*. Astfel se va afișa cea de-a treia pagină, în care utilizatorul va introduce întâi numele pe care îl dorește pentru a fi identificat, apoi va apăsa pe butonul *Register*. Dacă operația de adăugare a utilizatorului a fost reușită va fi afișat mesajul *User registered!*. În acest caz poate fi accesat link-ul care conduce spre pagina anterioară, de autentificare și parcurge instrucțiunile specificate anterior, corespunzătoare acestei pagini.

În urma autentificării va fi afișată cea de-a patra pagină care va aștepta până când utilizatorul va orienta telefonul landscape. Atunci când este respectată condiția va începe numărătoarea inversă de la 4 până la 0, apoi va fi afișată ultima pagină.

Dacă utilizatorul va apăsa pe butonul *Start* dispozitivul mobil va începe să vireze sau să meargă înainte sau înapoi, în funcție de orientarea telefonului mobil. La apăsarea butonului *Stop* dispozitivul nu va mai primi comenzi de la telefonul mobil.

Pentru a ieși din aplicație este necesar să fie accesat butonul *Logout.*

6. Concluzii

Recunosc că a fost o provocare pentru mine să realizez această aplicației deoarece nu cunoșteam la început toate limbajele pe care trebuia să le folosesc. Dar la finalizarea acesteia pot spune că mi-am însușit noțiuni de bază pentru limbajul Ruby, nou pentru mine de altfel. A fost necesar să urmăresc tutoriale care să mă direcționeze spre instrucțiunile pe care trebuia să le folosesc. Documentația de la Amazon a fost dificil de înțeles la început pentru mine, dar privind la exemple și paralel la documentație cred că am înțeles modul în care se lucrează cu serviciile puse la dispoziție de Amazon.

De asemenea, nu am fost prietenă cu partea hardware în timpul facultății, am preferat să lucrez mai mult pe dezvoltarea de programe software. Însă cred că este bine ca un programator să cunoască noțiunile de bază și din domeniul hardware. Prin realizarea acestui proiect am dorit să învăț câteva noțiuni elementare și în privința aceasta și cu puțin ajutor pot spune că am început să înțeleg și modul de programare la nivel de începător din acest domeniu.

Obiectivele pe care am reușit să le ating din cele pe care mi le-am propus la început cu privire la acest proiect:

- Am realizat o interfață web prietenoasă și ușor de înțeles;
- Modul de autentificare în aplicație este deosebit de altele, ușor de realizat și mai sigur;
- Este asigurat un confort ridicat în coordonarea unui dispozitiv mobil prin intermediul acestui proiect;
- Am dobândit mai multe cunoștințe atât la nivel de programare software, cât și de programare hardware.

Obiective pe care nu am reușit să le realizez:

- Să detectez dacă în vizorul camerei există o față ale cărei date pot fi prelucrate. Momentan este afișat un mesaj de eroare de la Amazon în cazul încercării autentificării fără ca imaginea capturată să conțină o față;
- Să implementez funcționalitatea aplicației și pentru sistemul de operare iOS.

Chiar dacă nu am reușit să implementez și aceste două soluții, intenționez ca în viitorul apropiat să le duc la îndeplinire și pe acestea, mai ales că cel din urmă nu a fost realizat din cauza timpului care s-a scurs foarte repede de la începutul implementării proiectului până la momentul prezentării proiectului de diplomă.

Bibliografie

[1] https://www.collinsdictionary.com/dictionary/english/web-page
[2] https://www.roweb.ro/ro/tehnologii/node-js
[3] https://www.ruby-lang.org/en/about/
[4] Matz, vorbind în cadrul Ruby-Talk, 12 Mai, 2000
[5] http://sinatrarb.com/intro.html
[6] https://www.wikitechy.com/tutorials/ruby-on-rails/ruby-on-rails-vs-sinatra
[7] http://blog.scoutapp.com/articles/2017/02/20/rails-api-vs-sinatra-vs-grape-which-ruby-microframework-is-right-for-you
[8] https://aws.amazon.com/rekognition/
[9] https://ro.wikipedia.org/wiki/Wi-Fi
[10] https://ro.wikipedia.org/wiki/Wi-Fi
[11] https://ro.wikipedia.org/wiki/Wi-Fi
[12] https://ro.wikipedia.org/wiki/Apache_(server)
[13] https://www.w3schools.com/nodejs/nodejs_raspberrypi_blinking_led.asp
[14] https://hackernoon.com/building-a-face-recognition-web-app-in-under-an-hour-345aa91487c
[15] https://docs.aws.amazon.com/sdkforruby/api/Aws/Rekognition/Types/FaceSearchSettings.html
[16] https://www.raspberrypi.org/downloads/raspbian/
[17] https://www.sdcard.org/downloads/formatter_4/
[18] https://etcher.io/
[19] http://hdgifs.com/gif/hero-car/
[20] http://cdn.3dvista.com/brookwood/index.htm
[21] https://github.com/amw/jpeg_camera
[22] https://www.smart-prototyping.com/L298N-Dual-H-bridge-Motor-Driver-Board
[23] https://altex.ro/acumulator-extern-universal-myria-mu01-bu-2000mah-albastru/
[24] https://www.eletronicshopp.com.br/item/ANTENA-WI%252dFI-COM-ADAPTADOR-USB-PARA-RECEPTORES-VARIADOS.html
[25] https://www.justboom.co/product/raspberry-pi-zero-w-soldered-header/
[26] https://cdn.sparkfun.com/r/600-600/assets/learn_tutorials/6/7/6/PiZero_1.png
[27] https://pinout.xyz/pinout/ground