



UNIVERSITATEA DIN
BUCUREŞTI



FACULTATEA DE
MATEMATICA ŞI
INFORMATICA

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

TITLUL LUCRĂRII DE LICENȚĂ

Absolvent

 Numele studentului

Coordonator științific

Titlul și numele profesorului coordonatorului

București, iunie 2021

Rezumat

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce vitae eros sit amet sem ornare varius. Duis eget felis eget risus posuere luctus. Integer odio metus, eleifend at nunc vitae, rutrum fermentum leo. Quisque rutrum vitae risus nec porta. Nunc eu orci euismod, ornare risus at, accumsan augue. Ut tincidunt pharetra convallis. Maecenas ut pretium ex. Morbi tellus dui, viverra quis augue at, tincidunt hendrerit orci. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam quis sollicitudin nunc. Sed sollicitudin purus dapibus mi fringilla, nec tincidunt nunc eleifend. Nam ut molestie erat. Integer eros dolor, viverra quis massa at, auctor.

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce vitae eros sit amet sem ornare varius. Duis eget felis eget risus posuere luctus. Integer odio metus, eleifend at nunc vitae, rutrum fermentum leo. Quisque rutrum vitae risus nec porta. Nunc eu orci euismod, ornare risus at, accumsan augue. Ut tincidunt pharetra convallis. Maecenas ut pretium ex. Morbi tellus dui, viverra quis augue at, tincidunt hendrerit orci. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam quis sollicitudin nunc. Sed sollicitudin purus dapibus mi fringilla, nec tincidunt nunc eleifend. Nam ut molestie erat. Integer eros dolor, viverra quis massa at, auctor.

Cuprins

1	Introducere	4
2	Preliminarii	5
3	Hardware	6
3.1	Ciclocomputer-ul	6
3.1.1	Asamblarea Ciclocomputer-ului	6
3.1.2	Ajustări hardware pentru alimentare sigură	7
3.1.3	Procesul iterativ de dezvoltare hardware	7
3.1.4	Modelare 3D	8
4	Software	10
4.1	Detectia de semne	10
4.1.1	Algoritmul de detectie	11
4.1.2	Etape preparatorii si logging	12
4.1.3	Arhitectura programului	13
4.2	Ciclocomputer-ul	14
4.2.1	Citirea de date de la senzori	14
4.2.2	Stocarea curselor biciclistului pe microSD	15
4.2.3	Transmiterea de informatii catre biciclist	16
5	Concluzii	17
	Bibliografie	18

Capitolul 1

Introducere

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean dignissim metus justo, nec pharetra mauris tincidunt id. Praesent semper turpis quis faucibus pulvinar. Fusce ut justo nisi. Praesent vehicula blandit erat, sed dignissim justo bibendum lobortis. Vivamus fringilla, elit at pulvinar imperdiet, dui elit lobortis sapien, et vehicula urna ex et velit. Sed efficitur, neque sed egestas lobortis, diam leo pellentesque sem, nec gravida est nunc efficitur orci.

Capitolul 2

Preliminarii

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean dignissim metus justo, nec pharetra mauris tincidunt id. Praesent semper turpis quis faucibus pulvinar. Fusce ut justo nisi. Praesent vehicula blandit erat, sed dignissim justo bibendum lobortis. Vivamus fringilla, elit at pulvinar imperdiet, dui elit lobortis sapien, et vehicula urna ex et velit. Sed efficitur, neque sed egestas lobortis, diam leo pellentesque sem, nec gravida est nunc efficitur orci.

Capitolul 3

Hardware

3.1 Ciclocomputer-ul

Ciclocomputer-ul este modulul principal in jurul căruia este bazat tot proiectul. Acesta are la bază o placă de dezvoltare **SparkFun Thing Plus - ESP32 WROOM**. Scopul său principal este să adune date despre călătoriile biciclistului și să afișeze în timp real viteza, distanța parcursă precum și timpul petrecut de utilizator pe bicicletă.

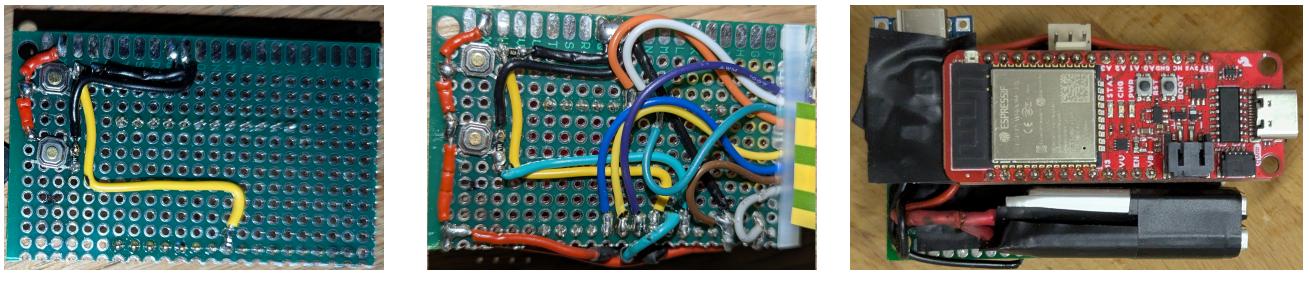
Componentele care alcătuiesc modulul sunt:

- Placă de dezvoltare SparkFun Thing Plus - ESP32 WROOM
- Ecran E-paper
- Senzor sensibil la efectul Hall
- Modulul de incarcare TP4056
- 2 Baterii LiPo 280 mAh

3.1.1 Asamblarea Ciclocomputer-ului

Pentru a asambla proiectul am pornit estimând suprafața necesară a placii perforate pentru a realiza toate conexiunile proiectate, urmand să sectionez placă respectivă folosind un cutter. Urmatorul pas a fost să lipesc 2 sini de pini, conectori mama, pentru a putea face o conexiune non-permanenta cu placă de dezvoltare. Apoi am lipit butoanele și rezistoarele necesare, facând conexiunile cu pinii corespunzători de pe placă de dezvoltare, precum se poate vedea în Figura 3.1a. Am folosit componente SMD pentru a ocupa cât mai puțin spațiu.

O a doua etapă a asamblării proiectului a implicat atât atașarea conectorului cu 3 pini pentru senzorul cu efect hall, care este poziționat în partea din centru-jos a placii perforate, cât și realizarea conexiunilor necesare pentru ecranul E-paper. Pentru a minimiza riscurile



(a) Componente SMD

(b) Conexiuni ecran

(c) Baterie și încărcător

Figura 3.1: Componente electronice pe placă de dezvoltare

ca aceste conexiuni să fie afectate de repede miscari ale ecranului sau schimbari de carcasa, cablurile au fost fixate folosind hot glue și o banda de plastic rigida în capatul din dreapta al placutei de dezvoltare, așa cum se poate observa în Figura 3.1b.

Un ultim pas pentru finalizarea modulului a fost conectarea bateriilor LiPo de 4.2 V în paralel pentru a obține o baterie echivalentă cu aceeași tensiune dar cu o capacitate dubla. Înaintea realizării conexiunii, cele două baterii au fost încărcate la aceeași tensiune pentru a evita o descarcare brusă de curent. Apoi am realizat legatura cu modulul de încarcare, a cărui ieșire este conectată la pin-ul microcontrollerului dedicat bateriilor, V_BATT. De asemenea, am conectat un intrerupator la firul pozitiv dintre încărcător și MCU pentru a putea opri dispozitivul direct din hardware.

3.1.2 Ajustări hardware pentru alimentare sigură

Unele componente au necesitat modificări pentru buna funcționare a proiectului.

Bateriile LiPo folosite au fost reciclate din 2 tigari electronice de unică folosință și nu au un circuit de protecție care să prevină supra-încărcarea, supra-descarcarea sau scurt-circuitarea lor. De asemenea, circuitul de încărcare MCP73831, montat pe placă de dezvoltare folosită, nu este dotat cu astfel de protecții pentru baterie. Prin urmare, am ales un modul de încarcare bazat pe chip-ul TP4056, care este dotat cu protecțiile menționate, și am deconectat de pe placă de dezvoltare modulul implicit.

Modulul de încarcare TP4056 respectă sugestia de conectare din fisa tehnică de la secțiunea "Aplicatii Tipice", conexiunile sale fiind facute pentru o baterie de 1Ah. Ciclocomputerul este proiectat pentru a folosi 2 baterii de 280 mAh. Ele sunt conectate în paralel, având o baterie echivalentă de 560 mAh. Prin urmare, rezistența de programare a modulului a fost schimbată de la $1.2K\Omega$ cu una de $2.2K\Omega$ pentru o încarcare potrivită unei baterii cu capacitate mai mică.

3.1.3 Procesul iterativ de dezvoltare hardware

Datorită faptului că acest proiect este în dezvoltare de mai mult de un an, multe modificări software, dar mai ales hardware au fost facute pe fondul utilizării sale în mod



(a) Hardware pentru versiunile 1 si 2



(b) Vedere din laterală a conectorilor



(c) V1 montata pe bicicleta



(d) V2 montata pe bicicleta

Figura 3.2: Versiunile 1 si 2 de hardware, inclusiv carcasele folosite

recurent in viata mea de zi cu zi. Acesta a avut 4 versiuni principale, fiecare suferind diferite modificari mici inainte sa fie abandonate in favoarea unei versiuni mai noi.

Prima dintre acestea separa complet placuta de dezvoltare si ecranul, cele 2 componente avand cate o carcasa separata si fiind conectate folosind legaturi temporare conform ???. O placuta perforata a fost folosita drept adaptor intre periferice si microcontroller, acestea fiind conectate folosind legaturi temporare dupa cum se poate observa in 3.2a si 3.2b. De asemenea, bateria externa folosita era la randul ei prinsa de un suport pentru telefon. Acum acest design avea drept scop principal demonstrarea viabilitatii ideii, ceea ce a si facut.

A doua verisune a construit peste hardware-ul primei, adaugand butoane pentru a interactiona cu meniul, un conector pentru a face legatura cu senzorul, inlocuind cei 3 pini spearati folositi anterior. De asemenea, v2 a adus un sistem mult mai bun de prindere pe ghidon care a fost mostenit parcial si in versiunie ulterioare si a unificat in cadrul modelului 3D ecranul si microcontrollerul. Bateriei i s-a proiectat un suport dedicat, prins pe cadru cu ajutorul unor elastice de par. Aceste modificari pot fi vazute in 3.2d.

Versiunea 3 a fost proiectata din cauza esecului principal al celei anterioare: conexiuni sigure si stabile intre componente folosite. Prin urmare,

3.1.4 Modelare 3D

Pentru a proteja electronicele de socuri, apa si praf, am modelat 3D, folosind Fusion 360, cate o carcasa pentru fiecare versiune majora proiectului. Astfel, au existat o multitudine de variatii ale carcasei pentru a acomoda cat mai bine forma hardware-ului.

In general, procesul de modelare a respectat o serie de pasi. Mai intai am masurat caracteristicile principale ale modulului folosind un subler. Apoi am proiectat o cutie care sa acomodeze hardware-ul, lasand gauri pentru porturi si o deschidere superioara pentru inserarea modulului. Dupa printarea primei variante a versiunii curente, am observat gradul de potrivire dintre carcasa si modul si am facut modificarile necesare asupra modelului 3D pentru a facilita o potrivire cat mai buna.

Capacul a fost proiectat cu un mecanism de prindere care a fost usor imbunatatit de la fiecare versiune. Desi initial era rolul capacului de a tine in loc hardware-ul, in final au fost adaugate la cutia principala 2 carlige de care se poate prinde o banda elastica pentru a pastra modulul nemiscat.

Butoanele folosite sunt proiectate pentru a fi actionate indirect, printr-un mecanism extern. Prin urmare, alaturi de ele a fost proiectat un sistem de ghidare care functioneaza impreună cu niste butoane printate 3D, acestea având o zonă de apăsare accesibilă utilizatorului și o bară prelungită care apasă butonul SMD. Aceasta bara este ghidată cu precizie de un suport lipit în jurul butonului pentru a asigura alinierea corectă la fiecare apăsare.

Pentru a prinde modulul de bicicleta sunt folosite 4 carlige, atașate de fiecare dintre cele 4 colturi inferioare ale carcasei. Folosind 2 elastice de par interconectate, carcasa este atasata de ghidon printr-o metoda simplă dar eficientă.

other
feed-
back

Capitolul 4

Software

Software-ul este o componenta majora in aceasta lucrare, majoritatea timpului petrecut pentru dezvoltarea proiectului fiind alocat acestei sectiuni.

4.1 Detectia de semne

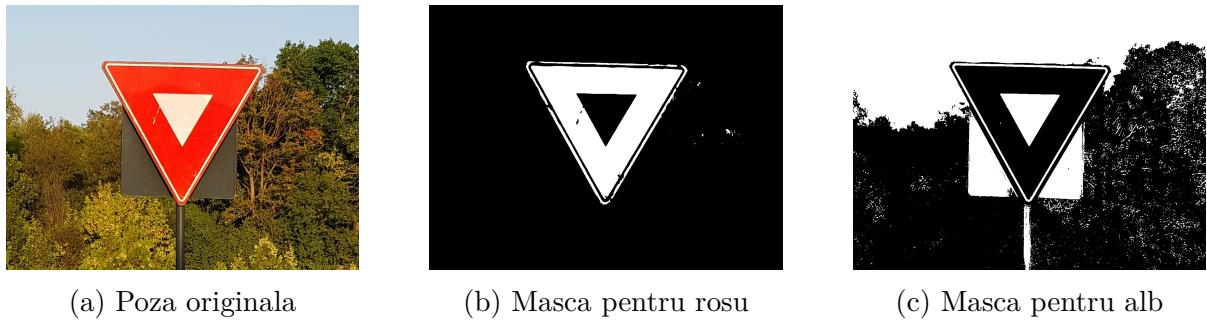
Scopul principal al acestei parti a proiectului este de a detecta semnele pe care un biciclist ar trebui sa le aiba in vedere pentru a circula in siguranta pe carosabil. Desi este importanta cunoasterea tuturor semnelor de circulatie pentru mersul pe strada, am considerat faptul ca urmatoarele 5 semne au o importanta deosebita: Cedeaza trecerea, Oprire, Presemnalizare trecere pietoni, Accesul interzis bicicletelor, Accesul interzis, acestea fiind ilustrate in Figura 4.1.

Incercarea initiala in rezolvarea problemei detectiei eficiente de semne de circulatie a fost implementata in Python 3, pentru a avea avantajul unui limbaj care faciliteaza atat vizualizarea problemei cat si incercarea rapida a mai multor abordari. Ulterior, pentru a mari eficienta algoritmului, codul a fost portat in C++. Acest lucru a micsorat timpul de detectie cu doua ordine de marime. De asemenea cateva dintre procedurile folosite au fost inlocuite pentru a profita de eficienta a diferitelor functii din biblioteca OpenCV[1], ceea ce a adus inca o imbunatatire a timpului de detectie cu 300%.

Scopul principal al acestui modul este sa captureze imagini si sa caute in mod activ



Figura 4.1: Semnele care justifica avertizarea biciclistului



(a) Poza originala

(b) Masca pentru rosu

(c) Masca pentru alb

Figura 4.2: Mastile generate de algoritm alaturi de poza originala

semne de circulatie de interes in cadrul acestora. In momentul unei detectii, prezenata semnului in imagine este transmisa cu ajutorul protocolului de comunicatie serial al ciclocomputer-ului, care avertizeaza sonor si vizual biciclistul de prezenta semnului si o inregistreaza in fisierul CSV cu datele traseului.

4.1.1 Algoritmul de detectie

Principiul de baza pentru gasirea semnelor de circulatie de interes se bazeaza pe compararea unui petic din imagine cu un sablon care contine o versiune curata a felului in care semnul de circulatie trebuie sa arate. O tehnica cunoscuta implica folosirea unor ferestre glisante de mai multe dimensiuni si orientari care parcurg intreaga imagine, dar aceasta metoda ar consuma prea mult timp pentru o detectie in timp real. Prin urmare, principiul urmat este gasirea unor regiuni de interes care sa fie aplicate ulterior peste imaginea cu sablonul, cu scopul de a compara pixel cu pixel cele doua versiuni. Aceste regiuni sunt in cazul acesta determinate de componente conexe rosii.

Primul pas pentru realizarea detectiei este crearea unor masti binare pentru prezenta fiecarei culori relevante in semnale de circulatie propuse pentru detectie: rosu, alb si negru. Cu ajutorul functiei `cv::inRange` se obtine masca dorita, folosind limitele de hue saturation si value intre care este considerata ca se incadreaza fiecare dintre culori. O parte dintre aceste masti sunt ilustrate in Figura 4.2.

Apoi este folosita functia `cv::connectedComponentsWithStats()` pentru a obtine toate componente conexe din masca rosie, fiecare dintre acesta avand potentialul sa fie un semn de circulatie. Este important de mentionat ca aceasta functie genereaza o matrice unde fiecare componenta conexa are o eticheta unica. Dupa aceea, parcurgand aceste forme rosii unitare se incearca gasirea colturilor relevante pentru fiecare semn, cu scopul de a putea suprapune componenta conexa gasita cu template-ul corespunzator.

De exemplu, pentru semnul Oprire sunt identificate toate cele 8 potentiiale colturi, marand astfel forma poligonului dupa cum se vede in 4.3a. Apoi este calculat patratul care ar putea incadra perfect semnul de stop precum cel din 4.3b. Cu ajutorul celor 4 colturi, patratul este apoi transformat perspectiveal folosind functia `cv::getPerspectiveTransform()`

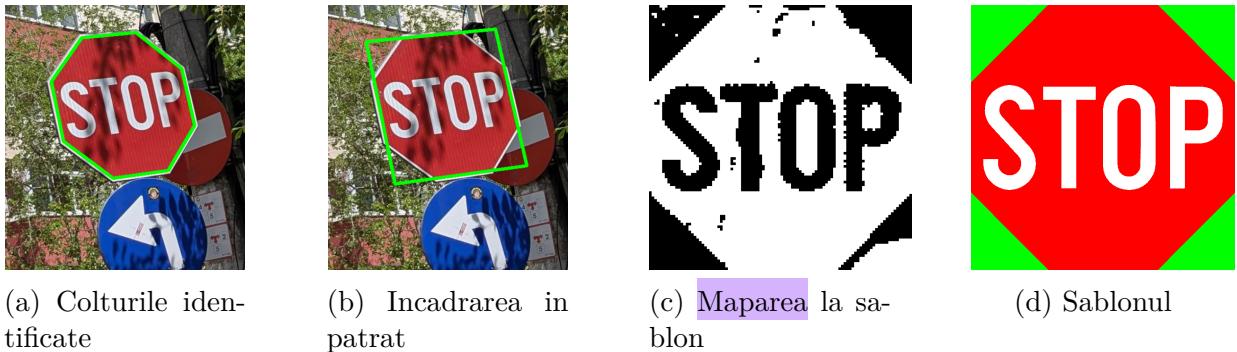


Figura 4.3: Procesul de detectie a semnului Oprire

astfel incat sa corespunda dimensiunilor template-ului cu care urmeaza sa fie comparat (4.3c vs 4.3d). Aceasta transformare este realizata pentru fiecare dintre mastile relevante detectiei. Prin urmare, pentru semnul Oprire este facuta atat pentru masca rosie cat si pentru cea alba.

Avand atat mastile transformate pe dimensiunile sablonului, cat si sablonul care denota cum ar trebui sa arate semnul de circulatie, se face o comparare pixel cu pixel pentru a stabili cat de bine se potriveste potentiala detectie din imagine cu template-ul respectiv.

Pentru a evita computatie inutila, inainte de calcularea patratului care incadreaza semnul si inainte de a face comparatiile pixel cu pixel se verifica unghiurile dintre laturile care formeaza poligonul reprezentativ pentru semnul verificat. De exemplu, pentru Oprire se iau toate unghiurile dintre toate cele 8 laturi si se verifica daca acestea sunt apropiate de 135 de grade, unghiul ideal dintre laturile unui octogon regulat. Daca nu, se considera ca acolo nu exista semnul respectiv.

Pentru obtinerea scorului de similitudine dintre sablon si zona verificata, este facuta o medie intre scorul de recall pentru toate culorile din semnul de circulatie respectiv. Scorul recall reprezinta raportul dintre detectiile reale identificate si totalitatea pixelilor care trebuiau detectati. De asemenea, intr-o incercare de a elimina cat mai multe fals pozitive, am calculat si cati pixeli rosii cu label-ul corespunzator componentei curente exista in zona verde din template, unde nu ar trebui sa existe astfel de pixeli. Daca raportul dintre numarul celor detectati in zona interzisa si cel al pixelilor interzisi totali depaseste un anumit procent, semnul primeste un scor de 0%.

4.1.2 Etape preparatorii si logging

Inainte de a incepe procedura de identificare de semne, este necesara realizarea unor proceduri de obtinere si pregatire a datelor.

Mai intai se incarca sabloanele in memorie, folosind un std::vector de tipul cv::Mat pentru a le stoca. Modelele au latimea de 100 de pixeli si contin culorile corespunzatoare semnelor pe care le reprezinta in forma lor pura (culoarea rosie contine, in format RGB,



(a) Opreire

(b) Presemnalizare trecere pietoni

(c) Accesul interzis bicicletelor

Figura 4.4: O parte din sabloanele folosite

valorile 255, 0, 0). O parte din sabloane pot fi vazute in 4.4.

Apoi, pentru fiecare poza facuta la rezolutia de 1280x960 se aplica un filtru median de marime 3x3 pentru a scapa de o parte din zgomotul existent la nivel de pixel. Apoi, dupa generarea mastilor se aplica o operatie de erodare, urmata de una de dilatare asupra mastilor rosii obtinute pentru detectie. Astfel sunt eliminate componentelete conexe mici fara a pierde prea mult detaliu din potenialele semne din imagine.

Un alt aspect important este faptul ca exista doua nuante de rosu care sunt luate in considerare, atat pentru un mediu intunecos cat si pentru unul luminos. Prin urmare exista doua masti pentru culoarea rosie asupra carora este realizata detectia separat.

De asemenea, exista functii care printeaza pe imagine atat siguranta detectiilor facute cat si locul in care detectiile au fost realizate, pentru depanare cat si pentru claritate. O constanta PRINT_STATS stabileste daca aceste functii sunt apelate. De asemenea, exista si o functie care se ocupa cu stocarea in flash a imaginilor in care au fost realizate detectii. Aceasta functie este de asemenea prevazuta pentru claritate si vizualizare. Imaginile sunt stocate pe calea `/bike-sys-data/detections`. Exista si un director `/bike-sys-data/control` unde este stocata o imagine la fiecare 10 secunde cu scopul de a obtine si imagini in care, in mare parte, nu exista niciun semn de circulatie. Aceste directoare nu vor fi folosite in cadrul functionarii obisnuite a proiectului, deoarece scopul principal al modulului este sa avertizeze utilizatorul ca exista un semn de circulatie in imagine, fara a fi relevanta pozitia, orientarea sau marimea sa in imagine.

4.1.3 Arhitectura programului

Programul realizat ruleaza sub linux, distributia fiind Raspberry Pi OS pe 64 de biti. Pentru a rula programul imediat ce sistemul e operare este pornit, am creat un cronjob care la @reboot ruleaza executabilul programului.

Acest modul al proiectului de licenta are 3 moduri principale de functionare: loop de detectie, detectie pe un director intreg cat si detectie pe o singura imagine.

Ciclul infinit de detectie are ca scop principal identificarea in timp real a semnelor din imagini si transmiterea detectiilor efectuate prin seriala catre ciclocomputer. Acesta este

proiectat sa functioneze la nesfarsit, fara a se opri cat timp programul nu este intrerupt de sistemul de operare.

Detectia pe director are ca scop aplicarea algoritmului de detectie asupra unui numar mare de imagini, cu scopul de a obtine date utile in identificarea performantei algoritmului de detectie. De exemplu, pornind de la directorul "imagini-test", codul creaza directorul "imagini-test-solved" in care sunt stocate aceleasi imagini din folderul initial, doar ca asupra lor au fost desenate contururile semnelor detectate cat si scorul de siguranta cu care acestea au fost identificate.

De asemenea exista si optiunea de a rula algoritmul de detectie asupra unei singure imagini stocate in flash, care si afiseaza pe loc rezultatul detectiei, fara sa il stocheze. Aceasta portiune de cod are scopul de a analiza pe loc performanta pe o imagine specifica.

4.2 Ciclocomputer-ul

Calculatorul de bicicleta are 4 misiuni principale, fiecare dintre ele ruland in paralel cu celelalte. Acestea sunt urmatoarele: citirea de date de la senzor si calcularea metricilor de cursa, stocarea curselor parcuse de biciclist pe un micro SD, transmiterea informatiilor relevante catre biciclist si comunicarea cu modulul de detectie de semne.

Software-ul scris pentru acest modul ruleaza sub FreeRTOS, un sistem de operare in timp real folosit in cadrul framework-ului ESP-IDF. Comparativ cu versiunea de baza FreeRTOS, implementarea ESP32 faciliteaza suport pentru mai multe nuclee, alocare dinamica a task-urilor pe core-uri cat si mecanisme avansate de sincronizare a proceselor care ruleaza in paralel, precum cozi de asteptare, semafoare si mutex-uri.

4.2.1 Citirea de date de la senzori

Interactiunea dintre senzori si microcontroller este gestionata integral de obiectul HardwareUtility. In cadrul acestui obiect exista structura privata "button" care incapsuleaza starea precedenta a butonului, starea sa curenta cat si momentul ultimei sale detectii. Cele 2 butoane ale dispozitivului, cat si senzorul sensibil la efectul hall pot fi reprezentate cu structura "button". Fiecare dintre periferice are propria metoda care defineste comportamentul sau. Cele doua butoane tin cont de starea precedenta cat si de starea curenta pentru a reprezenta o apasare, iar o apasare foarte scurta este ignorata, pentru a evita apasari accidentale. Pe de alta parte, pentru senzorul montat la roata se tine cont doar de schimbarea de la absenta magnetului la prezenta sa, durata detectiei fiind irelevanta.

Pentru a gestiona toate datele relevante pentru biciclist, am creat structura TripData, care incapsuleaza variabile pentru numarul total de detectii ale magnetului, momentul de timp in care a inceput cursa, momentul de timp al ultimei detectii ale magnetului cat si momentul de timp al detectiei precedente, viteza calculata conform ultimei detectii,

diagrama
pro-
cese

viteza calculata in urma detectiei precedente si viteza medie a unei calatorii.

Obiectul BikeCalc are drept membru o structura de tip TripData si are atributia principala de a calcula parametruii sai. Acest obiect defineste atat o metoda pentru inregistrarea detectiei curente, care actualizeaza parametrii variabilei TripData inainte de a intoarce valorile stocate in aceasta, cat si o metoda pentru aproximarea vitezei curente, care doar intoarce o structura TripData, fara a modifica variabila interna obiectului BikeCalc. Scopul acestieia din urma este de a facilita transmiterea de informatii approximative relevante catre utilizator legate de calatorie fara a avea o ultima detectie recenta. Un exemplu este o incetinire brusca, in urma careia utilizatorul isi schimba viteza de la 10 kmph la 0 kmph in mai putin de distanta perimetrului rotii. Acest lucru ar duce la afisarea vitezei curente de 10 kmph pana la plecarea biciclistului din loc, cand calculatorul ar putea face urmatorul calcul in lipsa unei functii care sa poata aproxima viteza curenta.

De asemenea exista si un obiect de tip Menu in care se tine cont de ecranul care este afisat catre utilizator si in care este stocat un obiect de tip TripData pentru a putea afisa informatiile relevante.

bug?

Avand acest context stabilit, se poate analiza rutina procesului care se ocupa cu gestionarea senzorilor. Aceasta verifica daca a fost detectata o schimbare la nivelul senzorului sensibil la efectul Hall, caz in care detectia este inregistrata iar obiectul de tip TripData este trimis catre procesul care se ocupa cu inregistrarea detectiilor pe microSD cu ajutorul unei cozi de asteptare. De asemenea, se verifica daca a fost apasat vreunul dintre cele 2 butoane disponibile utilizatorului, caz in care obiectul de tip Meniu este actualizat pentru a afisa pagina din meniu dorita de acesta. La fiecare 1.5 secunde cat si in momentul schimbarii variabilei meniu din proces se actualizeaza variabila de meniu globala. Aceasta variabila globala primeste, alaturi de schimbarea de pagina selectata de utilizator, fie datele de cursa cel mai recent detectate, fie o aproximare a acestora, in caz ca nu s-a mai facut nicio actualizare de la ultima detectie.

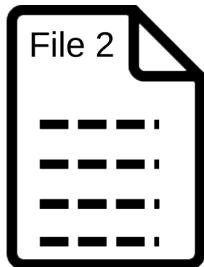
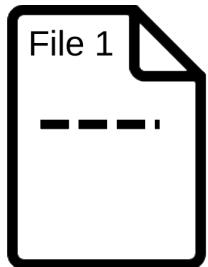
4.2.2 Stocarea curselor biciclistului pe microSD

Pentru a facilita o stocare de date organizata, a fost folosita biblioteca SD.h, care permite crearea si gestionarea de fisiere. Datele sunt stocate in format CSV. De fiecare data cand o calatorie noua este inceputa, este scris in fisierul curent antetul datelor, cu scopul de a delimita cursele. Scrierea la fisiere este facuta prin intermediul protocolului SPI, care este folosit si de procesul care modifica afisarea de pe ecranul dispozitivului. Prin urmare, este folosit un mutex numit g_spimutex pentru a proteja folosirea protocolului.

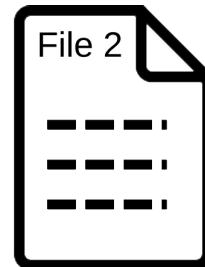
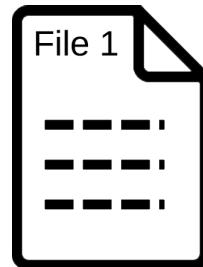
Modul de functionare a scrierii in fisiere este unitar. Mai intai se deschide fisierul in memorie apoi se apendeaza linia CSV curenta si, in final, se inchide fisierul, indiferent daca mai exista si alte linii de adaugat. Motivul pentru care fisierul nu este pastrat in memorie este pentru a permite utilizatorului sa opreasca oricand dispozitivul, fara a risca

cite
sd?

could
be
better?



(a) Fisiere impartite pe calatorii



(b) Fisiere limitate la o marime fixa

Figura 4.5: Optiunile de stocare pentru date de calatorie in flash memory

pierderea de date.

Initial fiecare fisier continea integral cate o calatorie, dupa cum se poate vedea in 4.5a, insa am observat experimental ca dupa un anumit timp fisierele devin prea mari, astfel incetind functionarea calculatorului de bicicleta in momentul deschiderii si ducand la pierderea unor detectii ale senzorului. Prin urmare, toate fisierele sunt limitate la o marime fixa de date, dupa cum indica imaginea 4.5b. Aceste fisiere pot fi reorganizate ulterior pentru a avea cate o cursa per fisier, cu ajutorul antetului mentionat anterior.

Rutina procesului de scriere la fisiere incepe verificand daca exista un microSD conectat. In caz contrar, procesul curent este oprit, intrucat logging-ul este imposibil. Altfel, se identifica cel mai recent fisier si se verifica daca acesta este mai mare decat marimea maxima admisa, caz in care se creaza un fisier nou. Apoi se apendeaza la fisierul curent antetul CSV-ului, care contine "time", momentul detectiei curente si "sign", indexul semnului detectat. In caz ca nu a fost detectat niciun semn, index-ul este -1. Apoi este pornit un ciclu infinit, in care se incearca citirea de la coada de asteptare dintre procesul curent si cel dedicat citirii de la senzor. In caz ca obiectul de TripData citit este diferit de cel citit anterior, continutul sau este stocat in fisierul curent, dupa care se verifica daca marimea fisierului depaseste marimea maxima admisa, caz in care este creat un fisier nou, care devine fisierul curent.

bad
file
names

4.2.3 Transmiterea de informatii catre biciclist

Scopul acestui proces este de a afisa pe ecran informatiile legate de calatoria curenta pe care utilizatorul vrea sa le vada, cat si de a-l avertiza pe biciclist de semnele de circulatie identificate, atat prin printarea semnului detectat pe ecran cat si printr-un semnal audio.

Capitolul 5

Concluzii

Bibliografie

- [1] G. Bradski, „The OpenCV Library”, in *Dr. Dobb’s Journal of Software Tools* (2000).