UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Tehnologia Informației

LUCRARE DE DIPLOMĂ

Coordonator științific:
conf. dr. ing. Andrei STAN

Absolvent:
Daniel BUDU

Iași, 2024

UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Tehnologia Informației

Sistem eHealth pentru analizarea mișcării

LUCRARE DE DIPLOMĂ

Coordonator științific:
conf. dr. ing. Andrei STAN

Absolvent:
Daniel BUDU

Iași, 2024

DECLARAŢIE DE ASUMARE A AUTENTICITĂŢII PROIECTULUI DE DIPLOMĂ

Subsemnatul Budu Daniel,
legitimat cu CI seria B nr. 33034996 , CNP 2006033044275
autorul lucrării Sistem eHealth pentru analizarea mișcării,
elaborată în vederea susţinerii examenului de finalizare a studiilor de licenţă, programul de studii CALCULATOARE organizat de către Facultatea de Automatică şi Calculatoare din cadrul Universităţii Tehnice „Gheorghe Asachi" din Iaşi,
sesiunea iulie a anului universitar 2024, luând în considerare conţinutul Art. 34 din Codul de etică universitară al Universităţii Tehnice „Gheorghe Asachi" din Iaşi (Manualul Procedurilor, UTI.POM.02 - Funcţionarea Comisiei de etică universitară), declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române (legea 8/1996) şi a convenţiilor internaţionale privind drepturile de autor.

Data

01.07.2024

Semnătura

Cuprins

Sistem eHealth pentru analizarea mișcării

Daniel BUDU

Rezumat

Această lucrare se concentrează pe dezvoltarea unui sistem eHealth inovator pentru detectarea și analiza mișcărilor simple ale mâinilor. Acest proiect utilizează senzorul de accelerometru MPU 9DOF Click (MPU-9250) în tandem cu microcontrolerul Raspberry Pi Pico W. Senzorul colectează date despre accelerațiile pe cele trei axe și le transmite prin interfața I2C către Raspberry Pi Pico W, care integrează un model de machine learning pentru detectarea tipului de mișcare efectuată de utilizator.

Încorporarea conceptului „Internet of Things" (IoT) în acest proiect evidențiază simplitatea și eficiența comunicării între dispozitivele precum Raspberry Pi Pico W și o interfață grafică implementată pe un laptop sau alt dispozitiv similar.

Aplicația eHealth dezvoltată poate asista utilizatorul în refacerea formei fizice a mâinilor sau în realizarea unor exerciții de încălzire musculară specifică mâinilor. Prin utilizarea datelor colectate și analizate în timp real, sistemul oferă feedback util contribuind astfel la îmbunătățirea performanței și sănătății mâinilor utilizatorului.

Introducere

Într-o societate modernă în care timpul a devenit una din cele mai prețioase resurese ale noastre și tehnologia a avansat până la punctul în care omenii sunt conectați in mod constant online si au acces instantaneu la informații, nevoia de sisteme avansate de monitorizare a sănătății nu a fost niciodată mai evidentă.

În acest context, am decis să dezvolt un sistem de eHealth pentru analiza comportamentului ca parte a tezei de licență. Scopul acestui proiect este de a transforma dispozitivele obișnuite în intrumente interactive care monitorizează și analizează mișcarile utilizatorului, oferindu-le astfel informații utile pentru a le înbunatați sănătatea și calitatea vieții.

În zilele noastre, din ce în ce mai aglomerate și solicitante, timpul este o resursă limitată și este important să eficientizăm fiecare activitate. Un rol cheie va fi jucat de Internet of Things(IoT), care conectează milioane de dispozitive care pot comunica între ele și pot automatiza diverse procese.

Dispozitivele IoT schimbă modul în care interacționăm cu mediul înconjurator, deoarece pot colecta si transmite date in timp real, fără a necesita intervenția continuă a utilizatorului. Sistemul de analiză a comportamentului eHealth monitorizează comportamentul utilizatorilor și oferă analize detaliate pe baza tehnologiei IoT și tehnici avansate de inteligență artificială.

Modul de viață alert și solicitant pe care îl trăim impune necesitatea unor soluții inovatoare pentru gestionarea sănătății și a bunăstării personale. Sistemele de monitorizare a sănătății bazate pe tehnologia IoT pot aduce beneficii semnificative, permițând utilizatorilor să își monitorizeze activitatea fizică. Astfel, proiectul își propune să transforme activitățile de zi cu zi și exercițiile fizice într-un proces monitorizat și analizat, oferind utilizatorilor informații valoroase pentru îmbunătățirea sănătății și a calității vieții.

Sistemul de eHealth propus constă dintr-un dispozitiv portabil format din două subsisteme, câte unu pentru fiecare mână. Fiecare subsistem este echipat cu un microcontroler Raspberry Pi Pico W și un senzor de mișcare MPU 9DOF pentru a monitorizarea în timp real a măscărilor. Aceste date sunt ulterior analizate de un model TinyML, antrenat pentru a recunoaște și clasifica diverse tipuri de mișcări ale mâinilor. Datele colectate sunt transmise prin TCP la un server central, unde sunt analizate și afișate utilizatorului printr-o interfață grafică.

Obiectivul principal al aplicației este dezvoltarea unui dispozitiv capabil sa monitorizeze și să analizeze mișcarile utilizatorului în timp real. Această inițiativă se bazează pe necesitatea crescândă de a integra tehnologia în viața cotidiană pentru a monitoriza și a inbunatați sănatatea. Sistemul propus va utiliza senzori de mișcare avansați pentru a capta date precise despre activitățile fizice ale utilizatorului.

Monitorizarea mișcărilor în timp real nu doar că oferă o imagine clară a nivelului de activitate fizică. Prin implementarea acestui sistem, utilizatorii vor putea să-și gestioneze mai bine rutina zilnică și să facă ajustările necesare pentru a-și îmbunătăți starea generală de sănătate.

Lucrarea este structurată în cinci capitole, fiecare abordând aspecte teoretice și practice esențiale pentru implementarea proiectului:

Capitolul 1 - Aspecte Teoretice: Acest capitol oferă o privire de ansamblu asupra domeniului de aplicabilitate al proiectului, evidențiind obiectivele propuse și oferind o comparație cu alte dispozitive similare și ideile de bază ale acestora.

Capitolul 2 - Proiectarea Sistemului: Analizează partea de proiectare a întregului sistem, atât din punct de vedere hardware cât și software, prezentând specificațiile tehnice ale componentelor și utilizarea acestora în cadrul proiectului.

Capitolul 3 - Implementarea Sistemului: Descrie procesul de implementare a dispozitivului, instalarea și modificările făcute componentelor inteligente, atașarea senzorilor și realizarea

aplicației pentru mobil.

Capitolul 4 - Testarea și Rezultatele Obținute: Prezintă testarea dispozitivului, rezultatele obținute, precum și eventualele defecte și îmbunătățiri identificate pe parcurs.

Capitolul 5 - Concluzii și Perspective de Dezvoltare: Încheie lucrarea cu idei de dezvoltare și extindere ale proiectului prezentat, evidențiind potențialul de aplicare în diverse domenii și propunând direcții viitoare de cercetare.

Capitolul 1. Fundamentarea teoretică și documentarea bibliografică

1.1. Domeniul și contextul abordarii temei

Termenul „Internet of Things”(IoT) se referă la dispozitive cu senzori, capabilități de procesare, software și alte tehnologii care conectează și schimbă date cu alte dispozitive și sisteme prin rețea de comunicații. Acestea includ inginerie electrică, comunicații și informatică. Termenul „Internet of Things” este uneori considerat înșelător, deoarece nu necesită dispozitive pentru a se conecta la internetul public. Tot ce trebuie să facă este să se conecteze la rețea și să fie adresabil individual.

Domeniul IoT a evoluat prin convergența mai multor tehnologii, cum ar fi calculatoarele omiprezente, senzorii avansați, sistemele încorporate puternice și invațare automată. Domeniile vechi, cum ar fi sistemele încorporate, rețelele de senzori fără fir și automatizarea lucrează impreună și independent pentru a permite dezvoltarea IoT.

Pe piața de consum, tehnologia IoT este cel mai frecvent asociată cu produse de „casă inteligentă”, cum ar fi corpuri de iluminat inteligente, termostate, sisteme de securitate și alte aparate electrocasnice, care pot fi controlate sau folosite prin intermediul unui smartphone sau a unui dispozitiv „inteligent”.

„Internet of Things” a devenit o tehnolofie esențială care transformă și înbunătățește multe domenii de activitate prin capacitatea sa de a conecta și comunica date între dispozitive.

Această revoluție tehnologică a condus la numeroase inovații și aplicații. A avut un impact semnificativ asupra diferitelor industrii. De la gestionarea eficientă a resurselor în orașele inteligente până la optimizarea proceselor industriale, IoT și-a dovedit utilitatea și versatilitatea. În continuare, vom explora câteva dintre domeniile cheie în care IoT a avut succes, evidențiind beneficiile și potențialul acestei tehnologii în fiecare domeniu.

- **Philips HealthSuite**
  Philips HealthSuite este o platformă IoT care colectează și analizează date de la dispozitive medicale conectate, cum ar fi monitoarele de ritm cardiac și tensiune arterială. Platforma oferă date în timp real pentru pacienți și profesioniștii din sănătate. Beneficiile aduse de acest sistem sunt înbunătațirea monitoriizarii pacienților, optimizarea tratamenului poate chiar și prevenirea bolilor.[1]

- **Orașe inteligente (Barcelona)**
  Barcelona este unu din cele mai dezvoltate orașe inteligente din lume. Proiectele IoT din acest oraș, includ gestionarea deșeurilor, iluminatul stradal, parcari inteligențe și soluții inteligente de irigații. Aceste etape de dezvoltare au adus o mulțime de beneficii caȘ reducerea consumului de energie, inbunatațirii calitații vieții și gestionarea mai eficientă a resurselor, care devine din ce în ce o problemă mai mare.[2]

- **General Electric (GE) Predix Platform**
  GE a dezvoltat platforma Predix pentru a colecta și analiza datele de la echipamentele industriale, platforma este utilizată în diverse industrii, aviație, industrie electrică și manufactură. Beneficiile aduse de această platformă sunt urmatoarele: optimizarea operatiilor, creșterea eficienței și reducerea timpului de nefuncționalitate.[3]

- **Automobile conectate(Tesla)**
  Mașinile Tesla sunt echipate cu o gamă largă de senzori și conectivitate IoT care permit actualizări software over-the-air, conducere autonomă și monitorizarea performanței vehiculului. Acest sistem vine și cu o gamă larga de beneficii ca creșterea siguranței, îmbunătățirea

experienței de conducere și actualizări continue ale vehiculelor fără a fi necesar un service fizic.[4]

- **John Deere**
  John Deere utilizează IoT pentru a dezvolta tractoare și echipamente agricole inteligente. Acestea sunt echipate cu senzori și conectate la internet pentru a monitoriza și optimiza activitățile agricole. Beneficiile utilizarii IoT sunt creșterea productivității agricole, reducerea costurilor și îmbunătățirea sustenabilității. [5]

- **Amazon Go**
  Amazon Go este un magazin fără casieri care utilizează o combinație de viziune computerizată, senzori și IoT pentru a permite clienților să facă cumpărături fără a trece printr-o casă de marcat. Beneficii: experiență de cumpărături rapidă și convenabilă, reducerea costurilor operaționale și colectarea de date valoroase despre comportamentul consumatorilor. [6]

1.2. Tema propusă

Această lucrare de licență își are radacinile în cunoștințele acumulate în cadrul a trei discipline fundamentale ale facultații: Arhitectura Calculatoarelor, Proiectarea Sistemelor Digitale și Sisteme Încorporate. Aceste materii m-au inspirat să dezvolt o temă de cercetare inovatoare în domeniul eHealth, concentrată pe analiza mișcării într-un context de utilizare medicală și de sanătate personală.

- **Arhitectura Calculatoarelor** a furnizat baza necesară pentru a înțelege structurile și principiile de funcționare ale sistemelor informatice, esențiale pentru proiectarea eficientă a soluțiilor eHealth. Cunoștințele acumulate în acest domeniu m-au pregătit să abordez aspectele tehnice complexe ale sistemului propus, asigurându-mă că acesta va funcționa optim și va oferi rezultate precise și fiabile în analiza mișcării.

- **Proiectarea Sistemelor Digitale** mi-a oferit cadrul necesar pentru a înțelege procesele de proiectare a componentelor digitale integrate, esențiale în dezvoltarea dispozitivelor eHealth. Această disciplină m-a familiarizat cu tehnologiile de senzori și cu procesele de optimizare a performanței sistemelor digitale, aspecte critice în implementarea soluțiilor de monitorizare a mișcării.

- **Sistemele Încorporate** au reprezentat puntea între teoria digitală și aplicațiile practice, concentrându-se pe integrarea și programarea sistemelor cu resurse limitate. Studiul acestui domeniu m-a pregătit pentru a gestiona eficient resursele hardware disponibile și pentru a dezvolta soluții inteligente pentru analiza mișcării într-un sistem eHealth.

Prin integrarea acestor domenii, am fost capabil să definească și să dezvolt o temă de licență care explorează potențialul tehnologiilor moderne de analiză a mișcării în contextul sănătății și al îmbunătățirii calității vieții. Această lucrare nu numai că reflectă cunoștințele teoretice acumulate, dar și demonstrează aplicabilitatea lor practică într-un proiect real. Subiectul ales nu doar că explorează frontierelor tehnologice actuale, dar și aduce o contribuție semnificativă în domeniul eHealth, facilitând monitorizarea mișcării pentru îmbunătățirea stării generale de sănătate.

Această combinare a cunoștințelor pe care le-am dobândit în facultate reprezintă fundamentul solid pe care îmi voi construi proiectul de licență. Ea oferă o înțelegere completă despre cum teoria poate fi aplicată în practică într-un domeniu inovator și cu impact mare, cum ar fi analiza mișcării în sistemele eHealth.

1.3. Tipuri de produse existente

Senzorii de accelerometru au revoluționat diverse domenii prin capacitatea lor de a detecta și analiza mișcările în mod precis și eficient. Aceste tehnologii au fost integrate într-o gamă largă de aplicații, oferind soluții inovatoare pentru sănătate. În acest subcapitol, vom explora câteva dintre cele mai semnificative aplicații ale senzorilor de accelerometru în analiza mișcărilor, evidențiind beneficiile și impactul lor în diferite contexte.

- **Thalmic Labs Myo Armband:** Este un dispozitiv purtabil care utilizează senzori de mișcare, inclusiv accelerometru, giroscop și electromiograf (EMG), pentru a detecta și interpreta gesturile mâinilor și brațelor utilizatorului. Acesta poate fi folosit pentru controlul gestual al computerelor, dispozitivelor mobile și a altor tehnologii. [7]

Figura 1.1. Thalmic Labs Myo Armband

- **Apple Watch Series 6:** Apple Watch utilizează un senzor accelerometru și giroscop pentru a detecta mișcările mâinilor și pentru a monitoriza activitatea fizică a utilizatorului. Aceste date sunt utilizate pentru a calcula pașii, distanța parcursă, ritmul cardiac și alte metrici relevante pentru sănătate. [8]

Figura 1.2. Apple Watch Series 6

- **Wearable Accelerometer and Gyroscope Sensors for Estimating the Severity of Essential Tremor:** Sistemul se bazează pe senzori purtabili care sunt atașați pe încheieturile mâi-

nilor sau alte zone afectate. Acești senzori includ accelerometre, care măsoară accelerația liniară, și giroscoape, care măsoară rata de rotație. Dispozitivele sunt ușor de purtat și nu interferează cu activitățile zilnice ale utilizatorului, asigurând confort și comoditate. Accelerometrele și giroscoapele din aceste dispozitive portabile sunt capabile să detecteze mișcările fine și frecvente asociate cu tremorul esențial. Accelerometrele înregistrează variațiile de accelerație pe diferite axe, în timp ce giroscoapele măsoară rotațiile și oscilațiile. Prin colectarea acestor date, dispozitivele oferă o imagine detaliată și precisă a mișcărilor involuntare ale pacientului. Un avantaj major al acestui sistem este capacitatea de a furniza feedback instantaneu utilizatorilor și medicilor. Informațiile analizate sunt accesibile prin intermediul aplicațiilor mobile sau platformelor web, permițând pacienților să-și monitorizeze starea în timp real și să observe efectele tratamentului. Medicii pot utiliza aceste date pentru a ajusta regimurile de tratament și pentru a lua decizii informate bazate pe date obiective, îmbunătățind astfel calitatea îngrijirii. Prin îmbunătățirea monitorizării și ajustării tratamentului, aceste tehnologii contribuie la creșterea calității vieții pacienților și la optimizarea îngrijirii medicale. [9]

Figura 1.3. Cuantificarea tremorului în boala Parkinson

- **Nintendo Wii Remote (Wiimote):** Este un controler de jocuri inovator lansat împreună cu consola Wii în 2006 de către Nintendo. Acesta a fost remarcabil prin utilizarea unui senzor de accelerometru incorporat, care permitea utilizatorilor să interacționeze cu jocurile prin mișcări fizice. Accelerometrul din Wiimote detecta schimbările de accelerație pe trei axe (X, Y, Z), permițând jucătorilor să simuleze gesturi precum mișcări de balansoar, lovire, aruncare sau rotire. Această tehnologie a adus o nouă dimensiune în experiența de joc, făcând jocurile mai interactive și mai captivante prin utilizarea mișcărilor corpului în loc de simple apăsări de butoane. În contextul Internet of Things (IoT), Wiimote exemplifică modul în care senzorii de accelerometru pot fi utilizați în dispozitive inteligente pentru a monitoriza și a răspunde la mișcările utilizatorilor în timp real. Această capacitate de interacțiune bazată pe gesturi a fost revoluționară pentru industria jocurilor și a deschis calea pentru alte aplicații IoT care utilizează senzori similari pentru monitorizarea mișcărilor în diverse domenii, cum ar fi sănătatea, fitnessul, realitatea augmentată și mai mult.

Figura 1.4. Nintendo Wii Remote

1.4. Prezentare comparativă privind realizările actuale pe aceeași temă

Aplicația mea va atinge unele parți din proiectele menționate în subcapitolul anterior, dar va fi o aplicație pentru antrenament, recuperare. Aplicația va permite selectarea unui exercițiu pe care ai vrea să-l practici, selectarea mâinii cu care ai vrea sa faci acest exercițiu îcât și selectarea unui interval de timp, pentru a practica ecest exercițiu.

Un plus al aplicației poate fi considerat intefața grafica destul de intuitiva și ușor de folosit. Pe langă acest beneficiu adus de aplicație, dispozitivele care vor fi fixate pe mâini nu vor fi conectate prin nici un fir la laptop pentru a transmite datele, toată comunicația se face wireless, astfel dispozitivul este ușor de folosi.

- Folosirea Raspberry Pi Pico W aduce foarte multe puncte forte, dimensiunile reduse fac posibilă integrarea acestuia in proiecte mici, consumul redus de energie electrica fac posibilă alimentarea acestuia de la o baterie, puterea de procesare care este destul de semnificativă, chiar și pentru proiecte mai mari.

- Utilizarea senzorului MPU 9DOF Click aduce și el o mulțime de beneficii în dezvoltarea proiectului eHealth. Modulul este compact și usor de integrat în dispozitive portabile, consumul redus de energie electică de asemenea este un beneficiu major.

- Atașarea proiectului pe o mănușă combină tehnologia avansată cu un design ergonomic, oferind utilizatorului confort în utilizare și este o soluție practică pentru monitorizarea mișcarii mâinilor.

Capitolul 2. Proiectarea aplicației

2.1. Componente hardware

2.1.1. Raspberry Pi Pico W

Raspberry Pi Pico w este o variantă a plăcii de dezvoltare Raspberry Pi Pico, ce vine cu suport pentru conectivitate wireless, de unde și sufixul "W" din denumire. Iată câteva detalii importante despre Raspberry Pi Pico W:

- **Microcontroler RP2040**
  Echipat cu un microcontroler dual-core ARM Cortex-M0+ cu o frecvență de 133 MHz, Raspberry Pi Pico W oferă un echilibru optim între performanță și eficiență energetică pentru aplicații embedded.

- **Conectivitate Wireless**
  Integrează o interfață wireless 2.4GHz, oferind suport pentru Wi-Fi și Bluetooth Low Energy (BLE). Acest lucru permite comunicația fără fir cu alte dispozitive și rețele, extinzând considerabil posibilitățile de conectivitate ale plăcii.

- **Memorie Flash**
  Dispune de 2 MB de memorie flash pentru stocarea programelor și datelor. Capacitatea generoasă de stocare permite implementarea de aplicații complexe și gestionarea eficientă a datelor.

- **Interfețe de Comunicare**
  Include o gamă variată de interfețe de comunicare, precum GPIO (General Purpose Input/Output), I2C, SPI, UART, PWM, ADC etc. Aceste interfețe permit conectarea ușoară la o diversitate de periferice și senzori, facilitând extinderea funcționalităților plăcii.

- **Alimentare Flexibilă**
  Poate fi alimentat fie direct prin portul Micro USB Type-B, fie prin pini GPIO, cu o tensiune de operare cuprinsă între 1,8V și 5,5V. Această flexibilitate în alimentare o face adecvată pentru o varietate de aplicații și scenarii de utilizare.

- **Suport Software**
  Beneficiază de suport extins din partea comunității Raspberry Pi, cu o gamă largă de librării și exemple de cod disponibile. Acest suport simplifică dezvoltarea de proiecte și permite utilizatorilor să exploateze pe deplin capabilitățile plăcii.

- **Dimensiuni Compacte**
  Cu dimensiuni de 21mm × 51mm și un design subțire de 1mm, Raspberry Pi Pico W este extrem de compactă și poate fi integrată ușor în proiecte cu spațiu limitat.

- **Securitate și Fiabilitate**
  Integrează funcții de securitate și protecție, asigurând o operare stabilă și sigură în diferite medii și aplicații embedded.

- **Preț Accesibil**
  Este un produs accesibil ca preț, fiind o opțiune populară pentru proiecte de hobby, educaționale și prototipuri în domeniul electronicelor.

Raspberry Pi Pico W găsește aplicații în diverse domenii și proiecte, inclusiv:

- **IoT și Automatizare**
  Folosit pentru implementarea sistemelor IoT (Internet of Things) și automatizarea diverselor dispozitive și procese.
- **Monitorizarea și Controlul Dispozitivelor**
  Utilizat pentru monitorizarea și controlul dispozitivelor prin intermediul conectivității wireless, oferind flexibilitate în gestionarea diverselor scenarii.
- **Sisteme de Senzori și Colectare de Date**
  Integrat în sisteme de senzori pentru colectarea și analiza datelor de la diferite senzori ambientali și industriali.
- **Educație și Training**
  Este o platformă ideală pentru educație în domeniul electronicelor și programării, fiind accesibilă și ușor de înțeles pentru studenți și entuziaști.
- **Prototipare Rapidă**
  Folosit pentru dezvoltarea rapidă de prototipuri și demonstrații pentru diverse proiecte și idei inovative.

Raspberry Pi Pico W este o soluție puternică și versatilă pentru dezvoltatorii care doresc să integreze funcționalități wireless în aplicațiile lor embedded. Cu o combinație echilibrată de performanță, conectivitate extinsă și suport software bogat, acesta oferă o bază solidă pentru proiecte în domeniul IoT, al sistemelor înglobate și al automatizării.[10]

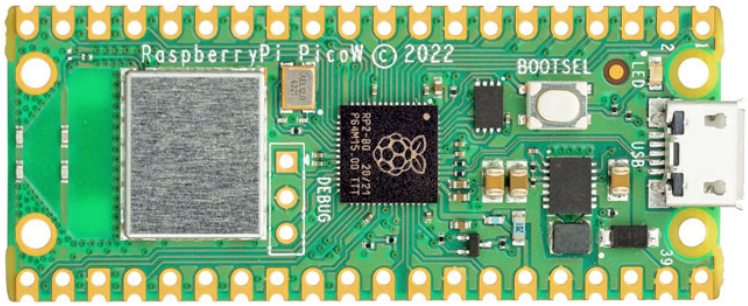

Figura 2.1. Raspberry Pi Pico W

2.1.2. MPU 9DOF Click

MPU 9DOF Click este un modul senzorial care include un senzor de mișcare cu 9 grade de libertate (DOF - Degrees of Freedom). Acesta combină un giroscop cu 3 axe, un accelerometru cu 3 axe și un magnetometru cu 3 axe, permițând măsurarea mișcării și orientării în spațiu pe toate cele trei axe (X, Y și Z). Câteva din caracteristicile senzorului sunt:

- **Giroscop cu 3 axe**
  Măsoară viteza unghiulară în jurul celor trei axe (X, Y și Z). Este util pentru detectarea rotației și stabilizarea mișcării.

- **Accelerometru cu 3 axe**
  Măsoară accelerația liniară în cele trei direcții (X, Y și Z). Este folosit pentru a detecta mișcarea liniară și gravitația.
- **Magnetometru cu 3 axe**
  Măsoară câmpul magnetic în cele trei direcții (X, Y și Z). Este folosit pentru a determina orientarea în raport cu câmpul magnetic al Pământului, funcționând practic ca o busolă.
- **Interfață I2C/SPI**
  Modulul poate comunica cu un microcontroller sau un alt dispozitiv prin intermediul unei interfețe I2C sau SPI, oferind flexibilitate în integrarea cu diverse platforme.
- **Precizie și stabilitate**
  Combinarea celor trei tipuri de senzori permite o măsurare precisă și stabilă a mișcării și orientării, făcând acest modul ideal pentru aplicații de navigație, robotică, realitate augmentată și multe altele.
- **Dimensiuni compacte**
  Modulul este de obicei mic și ușor de integrat în diverse proiecte electronice.

Figura 2.2. MPU 9DOF Clik

Modulul MPU 9DOF Click este foarte versatil și poate fi utilizat într-o gamă largă de aplicații care necesită măsurători precise ale mișcării și orientării.[11]

2.1.3. Cablul USB Type-A la Micro USB Type-B

Cablul USB Type-A la Micro USB Type-B este un element esențial pentru conectarea și programarea Raspberry Pi Pico W, oferind atât funcționalități de transfer de date, cât și de alimentare.

- **Conector USB Type-A**
  Conectorul USB Type-A este partea standard a cablului care se conectează la portul USB al laptopului sau PC-ului. Este compatibil cu porturile USB 2.0 și USB 3.0, oferind flexibilitate în utilizare.
- **Conector Micro USB Type-B**
  Conectorul micro USB Type-B este partea mai mică a cablului care se conectează la dispozitivul Raspberry Pi Pico W. Este compatibil cu majoritatea dispozitivelor care utilizează standardul micro USB pentru alimentare și transfer de date.

- **Compatibilitate**
  Funcționează cu majoritatea sistemelor de operare, inclusiv Windows, macOS și Linux, fără a necesita drivere suplimentare pentru funcționarea de bază.

2.1.4. Cabluri conexiune (jumper wires)

Aceste cabluri sunt utilizate pentru a realiza conexiuni între diferite componente pe o placă de prototipare sau direct între module. Sunt realizate din cupru pentru a oferi o conductivitate electrică bună, izolate cu un strat de plastic pentru a preveni scurcircuitele și a proteja firele de deteriorare.

2.1.5. Acumulatori

Ultra Fire BRC 18650 6800mAh 3.7V Li-ion, sunt acumulatorii pentru alimentarea lui Raspberry Pi Pico W, datorită tensiunei de alimentare oferite sunt compatibili cu acest microcontroller care accepta tensiune de alimentare între 1,8 V și 5,5 V.

2.2. Comonente software

2.2.1. Visual Studio Code (VS Code)

Folosirea editorului Visual Studio Code (VS Code) pentru programarea microcontrollerului Pico W în C/C++ ar fi o alegere excelentă datorită numeroaselor caracteristici si avantaje pe care le oferă. Câteva din aceste avantaje sunt:

- **Suport pentru C/C++**
  VS Code oferă suport pentru C/C++ prin intermediul extensiilor specializate cum ar fi "C/C++" de la Microsoft, care oferă autocompletare, evidențierea sintaxei și multe alte funcționalități.

- **Integrare Git**
  VS Code are o integrare excelenta cu Git, astfel oferă ușurință in descărcarea comonentelor necesare pentru programarea microcontrollerului Pico W, cum ar fi (pico-sdk, pico-examples).

- **Disponibilitate**
  VS Code este disponibil pe mai multe platforme cum ar fi: Windows, Linux, macOS.

- **Terminal integrat**
  VS Code include un terminal integrat care îți permite să rulezi comenzi direct din editor, ceea ce facilitează interactiunea cu alte toolchain-urile de dezvoltare.

- **Interfață prietenoasă**
  VS Code are o interfață prietenoasă și intuitivă, care poate fi personalizată în funcție de preferințele tale. Poți ajusta layout-ul, temele, și keybindings pentru a-ți optimiza fluxul de lucru.

- **Documentație excelentă**
  VS Code are o documentație bine pusă la punct, care te poate ajuta să înțelegi și să utilizezi eficient toate funcționalitățile sale.

2.2.2. ARM GNU Toolchain

- **Compatibilitate și Performanță:** ARM GNU Toolchain este optimizat pentru procesoare ARM, asigurând compatibilitate excelentă și performanțe optime pentru microcontrollerul RP2040 din Raspberry Pi Pico W.

- **Standardizare și Maturitate:** Fiind o unealtă matură și bine documentată, ARM GNU Toolchain oferă un set complet de instrumente de dezvoltare, incluzând compilator, asamblor, linker și debugger.

- **Optimizări Avansate**
  Compilatorul oferă numeroase opțiuni de optimizare care îmbunătățesc performanța și eficiența codului.

- **Debugging Detaliat**
  GDB (GNU Debugger) permite depanarea detaliată a codului, esențială pentru dezvoltarea stabilă și robustă a aplicațiilor.

- **Suport Extins pentru Biblioteci**
  Oferă suport pentru diverse biblioteci și framework-uri standard, simplificând integrarea funcționalităților complexe.

- **Comunitate și Resurse Abundente**
  Există o comunitate vastă de dezvoltatori și numeroase resurse online, facilitând suportul și învățarea.

- **Integrare cu Alte Instrumente**
  Se integrează bine cu editori de cod și IDE-uri populare, oferind un flux de lucru eficient.

- **Flexibilitate și Portabilitate**
  Disponibil pe multiple platforme (Windows, Linux, macOS), oferind flexibilitate în dezvoltare.

- **Costuri Reduse**
  Fiind open-source și gratuită, reduce costurile de dezvoltare și elimină preocupările legate de licențiere.

- **Calitate și Fiabilitate**
  Instrumentele sunt bine testate și verificate de comunitatea open-source, asigurând un cod de înaltă calitate.

Utilizarea ARM GNU Toolchain pentru Raspberry Pi Pico W permite dezvoltarea eficientă, performantă și robustă a aplicațiilor embedded, beneficiind de suportul extins și optimizările specifice platformei ARM.

2.2.3. CMake

- **Compatibilitate și Portabilitate**
  CMake este un sistem de build cross-platform care generează fișiere de proiect pentru diferite medii de dezvoltare (Windows, Linux, macOS). Acest lucru asigură compatibilitate și portabilitate pentru proiectele pe Raspberry Pi Pico W.

- **Automatizarea Procesului de Build**
  CMake automatizează procesul de generare a fișierelor makefile și a proiectelor pentru diverse IDE-uri. Aceasta simplifică considerabil configurarea și compilarea proiectelor C/C++.

- **Gestionarea Dependențelor**
  CMake facilitează gestionarea dependențelor externe și a bibliotecilor, permițând includerea ușoară a acestora în proiectul tău. Poți specifica dependențele și CMake le va gestiona automat.

- **Configurabilitate și Flexibilitate**
  CMake permite configurarea detaliată a procesului de build prin fișiere CMakeLists.txt. Poți defini opțiuni de compilare, variabile de configurare și reguli personalizate pentru a satisface nevoile specifice ale proiectului tău.

- **Suport pentru Proiecte Complexe**
  CMake este ideal pentru proiectele complexe care implică mai multe module și biblioteci. Poți organiza codul sursă în subdirectoare și poți gestiona eficient proiectele mari.

- **Integrare cu Alte Instrumente**
  CMake se integrează bine cu alte instrumente de dezvoltare și build, cum ar fi Make, Ninja, și multe IDE-uri populare. Aceasta permite un flux de lucru coerent și eficient.

- **Documentație și Comunitate**
  CMake are o documentație extinsă și o comunitate activă de utilizatori.

- **Detectarea Automată a Configurațiilor Sistemului**
  CMake poate detecta automat configurațiile sistemului și setările necesare pentru compilare, reducând astfel efortul manual de configurare.

- **Suport pentru Proiecte Multi-platformă**
  Astfel proiectele scrise pentru Raspberry Pi Pico W pot fi ușor portate pe alte platforme de dezvoltare.

- **Optimizarea Procesului de Compilare**
  CMake poate optimiza procesul de compilare prin generarea de reguli eficiente de build și prin suportul pentru compilarea paralelă, reducând astfel timpul necesar pentru compilare.

Utilizarea CMake pentru programarea Raspberry Pi Pico W permite automatizarea și optimizarea procesului de build, gestionarea eficientă a dependențelor și configurarea flexibilă a proiectelor, beneficiind de suportul extins și integrarea facilă cu alte instrumente de dezvoltare.

2.2.4. pico-sdk

- **Compatibilitate Directă**
  Pico-SDK este special conceput pentru microcontrollerul RP2040 al Raspberry Pi Pico W, oferind suport complet pentru toate funcționalitățile hardware ale acestuia.

- **Abstractizare a Hardware-ului**
  Pico-SDK oferă un strat de abstractizare a hardware-ului, simplificând accesul la periferice și resurse hardware, cum ar fi GPIO, UART, I2C, SPI, PWM și altele.

- **Biblioteci și Exemple**
  Pico-SDK vine cu un set vast de biblioteci și exemple preconfigurate, facilitând dezvoltarea rapidă a aplicațiilor și învățarea prin practică.

- **Optimizare pentru Performanță**
  Pico-SDK este optimizat pentru performanță, asigurând utilizarea eficientă a resurselor hardware și minimizarea latențelor în accesarea perifericelor.

- **Documentație Extinsă**
  Oferă o documentație detaliată și bine structurată, care acoperă toate aspectele legate de programarea RP2040, de la configurarea mediului de dezvoltare până la utilizarea funcțiilor avansate.

- **Integrare cu CMake**
  Pico-SDK utilizează CMake pentru gestionarea proiectelor, facilitând configurarea și compilarea codului sursă pe diverse platforme de dezvoltare.

- **Suport Comunitar și Actualizări**
  Beneficiază de suport din partea comunității și de actualizări regulate, asigurând îmbunătățiri continue și rezolvarea rapidă a eventualelor probleme.

- **Flexibilitate și Extensibilitate**
  Permite extinderea funcționalităților prin adăugarea de noi biblioteci și module, oferind flexibilitate în dezvoltarea de proiecte complexe și personalizate.

Utilizarea Pico-SDK pentru programarea Raspberry Pi Pico W asigură accesul direct și optimizat la resursele hardware, abstractizare eficientă și documentație extensivă, facilitând dezvoltarea rapidă și performantă a aplicațiilor embedded.

2.2.5. Ninja

- **Viteză și Eficiență**
  Ninja este un sistem de build conceput pentru viteză și eficiență. Este optimizat pentru compilarea rapidă a proiectelor mari, reducând semnificativ timpul de build în comparație cu alte sisteme tradiționale.

- **Paralelizare**
  Ninja suportă compilarea paralelă, folosind eficient toate nucleele procesorului pentru a accelera procesul de build. Acest lucru este deosebit de util pentru proiectele mari și complexe.

- **Integrare cu CMake**
  CMake poate genera fișiere de proiect pentru Ninja, facilitând integrarea ușoară a Ninja în fluxul de lucru existent. Aceasta permite utilizarea beneficiilor ambelor instrumente.

- **Simplitate și Ușurință în Utilizare**
  Fiind un sistem de build minimalist, Ninja are o sintaxă simplă și este ușor de configurat și utilizat. Aceasta reduce complexitatea gestionării procesului de build.

- **Fiabilitate**
  Ninja este un sistem de build robust și fiabil, utilizat pe scară largă în comunitatea dezvoltatorilor pentru proiecte de toate dimensiunile.

- **Diagnosticare Eficientă**
  Oferă mesaje de eroare clare și concise, facilitând diagnosticarea rapidă și corectarea problemelor apărute în timpul procesului de build.

- **Comunitate și Resurse**
  Există o comunitate activă de utilizatori și dezvoltatori Ninja, oferind suport și resurse pentru învățare și rezolvarea problemelor.

Utilizarea Ninja pentru programarea Raspberry Pi Pico W asigură viteză, eficiență, și fiabilitate în procesul de build, beneficiind de suport pentru compilarea paralelă, integrare ușoară cu CMake, și scalabilitate pentru proiecte de toate dimensiunile.

2.2.6. Git pentru Windows

- **Acces Rapid la Codul Sursă**
  Clonarea SDK-ului Pico și a exemplelor prin Git asigură accesul rapid și ușor la toate resursele necesare pentru dezvoltare. Comanda `git clone` permite descărcarea întregului depozit de pe un server Git în câteva secunde.

- **Actualizări și Sincronizare**
  Utilizarea Git permite menținerea SDK-ului și a exemplelor actualizate cu cele mai recente modificări și îmbunătățiri. Comanda `git pull` sincronizează depozitul local cu cel de pe server, asigurând accesul la ultimele versiuni ale codului.

- **Învățare și Experimentare**
  Exemplele incluse în depozitul SDK sunt resurse valoroase pentru învățare și experimentare. Clonarea acestor exemple permite studierea și modificarea lor pentru a înțelege mai bine funcționarea și utilizarea SDK-ului.

- **Versiuni Controlate ale SDK-ului**
  Prin clonarea SDK-ului prin Git, poți alege să lucrezi cu versiuni specifice ale SDK-ului.

- **Documentație Inclusă**
  Depozitul SDK include documentație detaliată despre utilizarea și configurarea acestuia. Clonarea depozitului asigură accesul la această documentație, facilitând configurarea corectă și utilizarea eficientă a SDK-ului.

- **Exemplu Practic de Clonare**

```
git clone https://github.com/raspberrypi/pico-sdk.git
git clone https://github.com/raspberrypi/pico-examples.git
```

  Aceste comenzi vor clona SDK-ul și exemplele Pico în directoarele locale, gata pentru a fi utilizate în proiect.

- **Integrare Ușoară cu IDE-uri**
  SDK-ul și exemplele clonate prin Git pot fi integrate ușor cu diverse IDE-uri și medii de dezvoltare, facilitând configurarea proiectelor și începutul rapid al dezvoltării.

Utilizarea Git pentru clonarea SDK-ului și a exemplelor pentru Raspberry Pi Pico W asigură acces rapid la resurse, actualizări ușoare, gestionarea eficientă a dependențelor, și suport pentru colaborarea eficientă, facilitând dezvoltarea robustă și rapidă a proiectelor în C/C++.

2.2.7. Edge Impulse

- **Edge Impulse**
  Este o platformă de dezvoltare și management pentru aplicații de IoT și dispozitive edge (margine), specializată în analiza datelor și implementarea de modele de învățare automată pe dispozitive mici.

- **Tool-ul Data Forwarder**
  Este o aplicație dezvoltată de Edge Impulse care permite conectarea simplă și eficientă a dispozitivelor hardware (cum ar fi Raspberry Pi Pico W) la platforma Edge Impulse prin intermediul portului serial.

- **Funcționalități Principale**

  - **Conectivitate Simplificată:** Data Forwarder facilitează configurarea și gestionarea fluxului de date între dispozitivul edge și platforma Edge Impulse.
  - **Transmitere a Datelor:** Permite transmiterea datelor colectate de la senzori (cum ar fi accelerometrul) către platforma cloud a Edge Impulse pentru analiză și procesare.
  - **Integrare și Analiză:** Datele colectate sunt analizate în platforma Edge Impulse folosind tehnici de învățare automată pentru a extrage informații utile și a antrena modele predictive.
  - **Dezvoltare Eficientă:** Tool-ul facilitează dezvoltarea rapidă a aplicațiilor IoT și embedded, permițând dezvoltatorilor să utilizeze datele de la senzori pentru a implementa soluții inteligente și adaptate nevoilor specifice.

- **Beneficii:**

  - **Simplificarea Dezvoltării:** Edge Impulse și Data Forwarder oferă un mediu integrat și ușor de utilizat pentru dezvoltarea și testarea aplicațiilor IoT.
  - **Analiză Avansată:** Permite analiza avansată a datelor senzoriale și implementarea de funcționalități de învățare automată pe dispozitive cu resurse limitate.
  - **Suport și Comunitate:** Platforma include suport extins și o comunitate activă de dezvoltatori, facilitând schimbul de cunoștințe și rezolvarea problemelor întâlnite în procesul de dezvoltare.

Utilizarea Edge Impulse și a tool-ului Data Forwarder permite dezvoltatorilor să exploateze la maximum datele senzoriale pentru a crea aplicații inteligente și performante în domeniul IoT și embedded.

2.2.8. PyCharm

- **PyCharm**
  Este un mediu integrat de dezvoltare (IDE) pentru limbajul de programare Python, dezvoltat de JetBrains. PyCharm oferă un set bogat de instrumente pentru dezvoltarea rapidă și eficientă a aplicațiilor Python.

- **Dezvoltarea Interfeței Grafice**
  Utilizând PyCharm, poți scrie cod pentru a crea o interfață grafică (GUI) în Python. PyCharm suportă diverse framework-uri GUI pentru Python, cum ar fi Tkinter, PyQt, Kivy etc., permițându-ți să alegi cea mai potrivită opțiune pentru proiectul tău.

- **Comunicare prin TCP**
  Datorită limbajului python poți ușor să folosești protocoale de comunicatie wireless, de exemplu TCP, pentru a primi date de la un micorocontroler cum ar fi Raspberry Pi Pico W, care are interat un modul de WiFi.

- **Integrare și Utilizare în Interfața Grafică**
  Python permite dezvoltare simplistă și rapida a interfețelor grafice(GUI) cu ajutorul framework-urilor cum ar fi Tkinter.

- **Beneficii ale Utilizării PyCharm:**

  - **Autocompletare și Debugging:** PyCharm oferă funcționalități avansate de autocompletare a codului și de debugging, facilitând dezvoltarea fără erori și depistarea problemelor în mod eficient.

- **Suport pentru Framework-uri:** IDE-ul suportă multiple framework-uri și biblioteci Python, permițând dezvoltarea flexibilă a aplicațiilor GUI în funcție de cerințele proiectului.
- **Integrare cu Alte Instrumente:** PyCharm se integrează bine cu alte instrumente și tehnologii, precum git pentru controlul versiunilor, îmbunătățind fluxul de lucru și eficiența dezvoltării tale.

Utilizarea PyCharm pentru dezvoltarea interfeței grafice în Python și integrarea cu datele primite prin TCP de la Raspberry Pi Pico W facilitează dezvoltarea aplicațiilor interactive și conectate în domeniul IoT și embedded.

2.3. Protocoale de comunicație

2.3.1. Transmission Control Protocol (TCP)

TCP este un protocol de comunicație de rețea care operează la nivelul 4(Transport) al modelului OSI, este unul dintre protocoalele principale ale suitei de protocoale Internet, adesea menționat împreună cu IP(Internet Protocol) ca TCP/IP. TCP este un protocol orientat pe conexiune, ceea ce înseamnă că înainte de a începe transferul de date, trebuie stabilită o conexiune între două puncte finale (hosturi). Acest lucru se face printr-un proces numit "handshake în trei pași" (three-way handshake). Protocolul dat de comunicație include algoritmi de control al fluxului de date pentru a preveni copleșirea receptorului cu prea multe date deodată. Mai include si algoritmi de control al congestiei, pentru a ajusta rata de transmitere a datelor în funcție de starea rețelei, prevenind astfel aglomerarea acesteia.

Printre avantajele TCP se numără: fiabilitate mare și corecție a erorilor, livrarea ordonată a pachetelor de date, control al fluxului și al congestiei pentru a menține performanța rețelei. Dezavantajele utilizării TCP sunt overhead suplimentar datorită mecanismului de fiabilitate și control al congestionării și faptul că este mai lent in comparație cu protocoalele fără conexiune, cum ar fi UDP, din cauza stabilirii conexiunii și a confirmarii pachetelor.[12]

2.3.2. Inter-Integrated Circuit (I2C)

I2C este un protocol de comunicație serială folosit pentru a conecta microcontrolerele si periferice într-un sistem electronic. A fost dezvoltat de Philips Semiconductor (acum NXP Semiconductor) în anii 1980 și este folosit pe scară largă în aplicații datorită simplității și eficienței sale.

TCP este esențial pentru funcționarea multor aplicații de rețea care necesită transferuri de date fiabile și ordonate.

Caracteristicele cheie ale I2C sunt:

- Interfața cu două fire, Serial Data Line(SDA) pentru transfer de date și Serial Clock Line(SCL) pentru semnalul de ceas care sincronizează transferul de date.
- Arhitectură Master-Slave, un dispozitiv master inițiază și controlează transferul de date, dispozitivele slave răspund la conezile masterului.
- Fiecare dispozitiv slave are o adresă unică iar masterul transmite adresa dispozitivului destinatar înainte de a iniția transferul de date.
- Transfeul de date poate avea loc în ambele direcții iar datele sunt transferate în mod serial, bit cu bit.
- Rate de transfer:
  - Standard Mode: până la 100 kbps

- Fast Mode: până la 400 kbps
- Fast Mode Plus: până la 1 Mbps
- High-Speed Mode: până la 3.4 Mbps

Avantajele utilizării I2C sunt simplicitatea, deoarece utilizează doar două fire pentru comunicație, reducând complexitatea și costurile de cablare, scalabilitatea, deoarece permite conectarea a numeroase dispozitive slave pe același bus, și versatilitate(suporta atât comunicație sincronă cât și bidirecțională).

Din dezavantajele I2C pot fi enumerate rata de transfer limitată, lungimea limitată a busului de date precum și erori de interferențe.

Protocolul I2C este extrem de util în aplicațiile încorporate datorită simplității sale și capacității de a gestiona multiple dispozitive cu o infrastructură minimală.[13]

2.4. Modelul general al aplicației și interacșiunea dintre ele

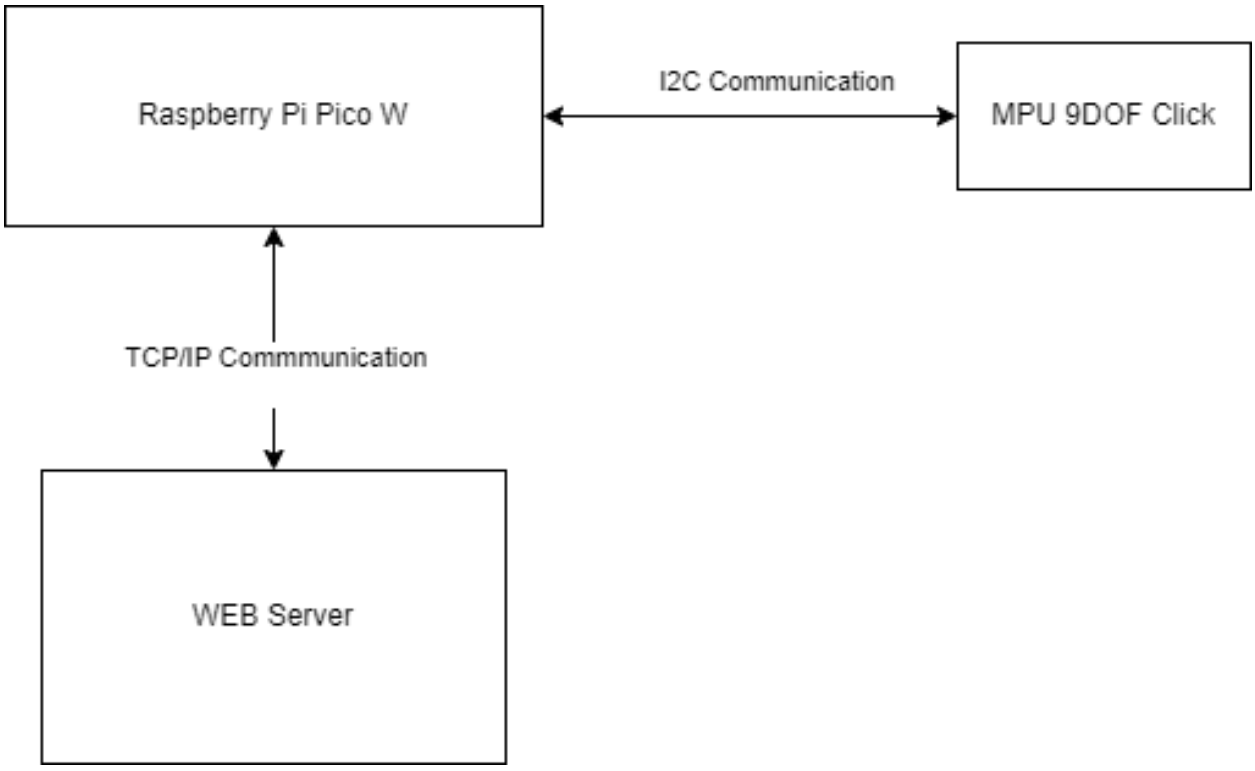

Figura 2.3. Modelul aplicației

Capitolul 3. Implementarea aplicației

3.1. Descrierea implementării

3.1.1. Conexiunile dintre Raspberry Pi Pico W si MPU 9DOF Click

Pentru realizarea acestui proiect am construit simultan 2 produse identice, câte unu pentru fiecare mână. MPU 9Dof Click suportă protocoalele de comunicație I2C și SPI. Pentru realizarea proiectului dat am folosit po=rotocolul de comunicatie I2C chiar daca SPI este mai rapid, viteza oferită de I2C este suficientă pentru citirea datelor de la un senzor de accelerometru într-un context de aplicație embedded. Utilizarea protocolului I2C face integrarea cu Raspberry Pi Pico W simplă și eficientă, utilizand doar doua fire (SDA și SCL), astfel reduce din complexitatea cablajului.

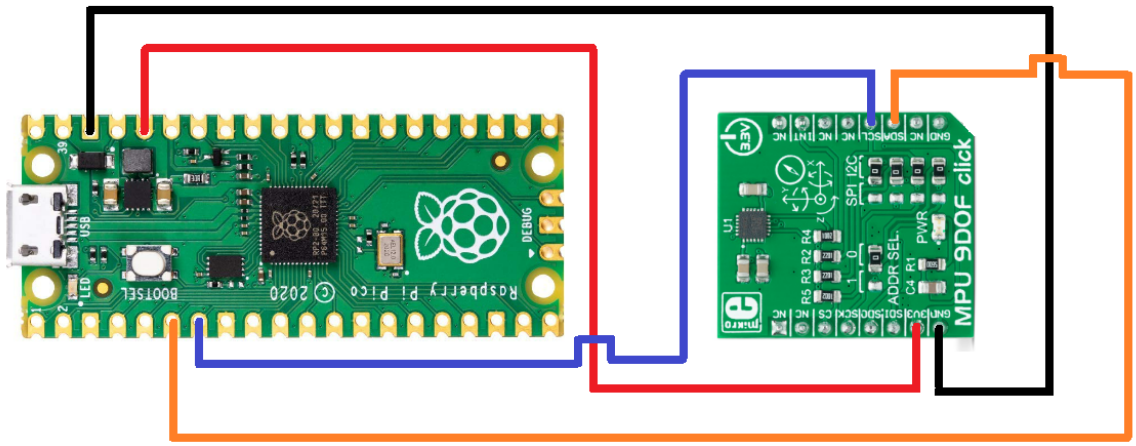

Figura 3.1. Conexiuni între Raspberry Pi Pico W și MPU 9DOF Click

Raspberry Pi Pico W	MPU 9DOF Click
Pin 6 (GP4) SDA	SDA
Pin 7 (GP5) SCL	SCL
GND	GND
3.3V	VCC

Tabelul 3.1. Conexiuni între Raspberry Pi Pico W și MPU 9DOF Click

Raspberry Pi Pico W permite configurarea flexibilă a pinilor GPIO pentru funția I2C. Utilizarea pinilor GP4 (Pin 6) și GP5 (Pin 7) pentru liniile I2C SDA si SCL este comună și bine suportată de bibliotecile de programare disponibile pentru Pio W.

- **GP4 (SDA):** Linia de date I2C este conectată la pinul GP4 deoarece acesta este unul dintre pinii desemnați pentru funcția I2C, asigurând o comunicație stabilă și eficientă cu senzorul.
- **GP5 (SCL):** Linia de ceas I2C este conectată la pinul GP5, care este de asemenea desemnata pentru funcția I2C, asigurând sincronizarea corectă a trasferului de date.

- **3V3 OUT:** Senzorul MPU 9DOF Click operează la 3.3V. Alimentarea corectă este esențiala pentru funcționarea senzorului, de aceea l-am conectat la Pin 36 de pe Pico W, care oferă tensiune de alimentare de 3.3V.
- **GND:** Conexiunea GND asigură referință de potențial comun pentru senzor și microcontroler, poate fi conectat la orice GND de pe Pico W, eu am ales Pin 38.

Alegerea pinilor GP4 si GP5 pentru comunicația I2C pe Raspberry Pi Pico W a fost făcută datorită flexibilității și compatibilității oferite de Pico W pentru această funcție. Protocolul de comunicație I2C simplifică conexiunile și este potrivit pentru comunicația eficienta cu senzorul MPU 9DOF Click, care are integrat in el modulul MPU-9250 produs de InventSense, care combină un girscop de 3 axe, accelerometru pe 3 axe și un manetometru pe 3 axe.

3.1.2. Comunicarea cu modulul MPU-9250

Pentru citirea datelor de pe modulul MPU-9250 am scris o bibliotecă C destul de limitată, citesc doar registrul de accelerometru, de asemenea o mini-biblioteca C++, în care am implementat o clasa „mpu9250”, pentru a face mai simplă citirea datelor de pe senzorul MPU-9250.

Pentru citirea datelor de pe modulul MPU-9250 am urmat câțiva pași relativ simpli:

- **Inițializarea I2C:** Inițializarea comunicației I2C pe pinii GP4 (SDA) și GP5 (SCL)la o frecvența de 100 kHz (Standard Mode).

```
i2c_init(I2C_PORT, 100 * 1000);
gpio_set_function(SDA, GPIO_FUNC_I2C);
gpio_set_function(SCL, GPIO_FUNC_I2C);
gpio_pull_up(SDA);
gpio_pull_up(SCL);
```

- **Adresa MPU:** Adresa I2C a sensorului MPU-9250 este 0x6B.
- **Inițializarea senzorului MPU-9250:** Senzorul este pornit scriind 0x00 în registrul de alimentare (0x6B).

```
uint8_t buf[] = {PWR_MGMT_1, 0x00};//PWR_MGMT_1 = 0x6B
i2c_write_blocking(I2C_PORT, MPU9250_ADDR, buf, 2, false);
```

- **Citirea registrului de accelerații:** Datele sunt citite din registrul de accelerometru (0x3B) și convertite în valori de accelerație în g.

```
uint8_t buffer[6];
uint8_t reg = ACCEL_XOUT_H; // ACCEL_XOUT_H =  0x3B
i2c_write_blocking(I2C_PORT, MPU9250_ADDR, &reg, 1, true);
i2c_read_blocking(I2C_PORT, MPU9250_ADDR, buffer, 6, false);

int16_t raw_ax = (buffer[0] << 8 | buffer[1]);
int16_t raw_ay = (buffer[2] << 8 | buffer[3]);
int16_t raw_az = (buffer[4] << 8 | buffer[5]);

acceleration[0] = raw_ax / 16384.0f;
acceleration[1] = raw_ay / 16384.0f;
acceleration[2] = raw_az / 16384.0f;
```

- **Clasa pentru modulul MPU-9250:**

```
class mpu9250
{
    public:
    float acceleration[3];

    mpu9250(uint8_t SDA, uint8_t SCL);
    void updateData();
    void printData();
};
```

3.1.3. Achiziția datelor pentru antrenarea modelului TinyML

Am folosit componentele hardware enumerate mai sus, cât și biblioteca scrisă pentru achiziția datelor în anumite ipostaze de mișcare a senzorului. Pentru achiziția datelor am scris o funcție în C++ car î-mi preia datele de la modulul MPU-9250 și le transmite pe serială.

```
mpu9250 mpu(4, 5);  // Creates an mpu object

int main()
{
    stdio_init_all();
    sleep_ms(10000);

    while (1)
    {
        mpu.updateData();

        printf(" %f, %f, %f\n",
        mpu.acceleration[0],
        mpu.acceleration[1],
        mpu.acceleration[2]);
        sleep_ms(10);
    }
    return 0;
}
```

Pentru achiziția datelor de la senzor și pentru o eventuală antrenare a unui model de machine learning am folosit Edge Impulse.

Edge Impulse oferă o interfață de linie de comandă (CLI) pentru a gestionarea proiectelor și datelor. În CLI te poți autentifica în contul tău Edge Impulse pentru a accesa un proiect existent de al tău, sau poți creea proiecte noi. Am folosit CLI pentru a încărca datele primite pe seriala de la Pico W, CLI oferă opțiuni pentru etichetarea și organizarea datelor, esențiale pentru modelel de machine learning.

Comanda pentru detecția microcontrolerului care trimite date și detecția automată a frecvenței de achiziție a datelor este:

```
edge-impulse-data-forwarder
```

După execuția comenzii poți eticheta datele primite de la senzor, în cazul meu a fost „X, Y, Z”, pentru accelerațiile pe fiecare din cele 3 axe, iar programul îți detectează frecvența de achiziție a datelor, pentru a forma timestampul la fiecare achiziție, un lucru la fel esențial.

Exemplu din Edge Impulse Data Forwarder (CLI):

```
[SER] Detected data frequency: 91Hz
3 sensor axes detected (example values: [-0.029053,-0.216309,1.018799]).
What do you want to call them? Separate the names with ',': X, Y, Z
```

Am decis să antrenez modelul meu de TinyML pentru detectarea starea de pauză cand nici una din accelerații nu se modifică, 3 mișcari simple (acceleratia se modifică doar pe una din axe) cum ar fi: stânga-dreapta, sus-jos, înainte-înapoi și o mișcare un pic mai complexă, accelerațiile se modifică pe 2 axe (mișcarea circulară), pentru fiecare mișcare am preluat datele de la senzor în câte 50 de secunde (5x10 secunde).

Câteva din datele achiziționate de la senzor pentru fiecare mișcare în parte:

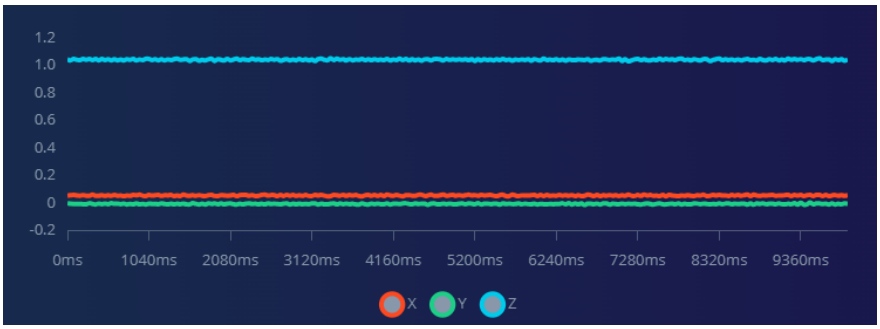

Figura 3.2. Pauză

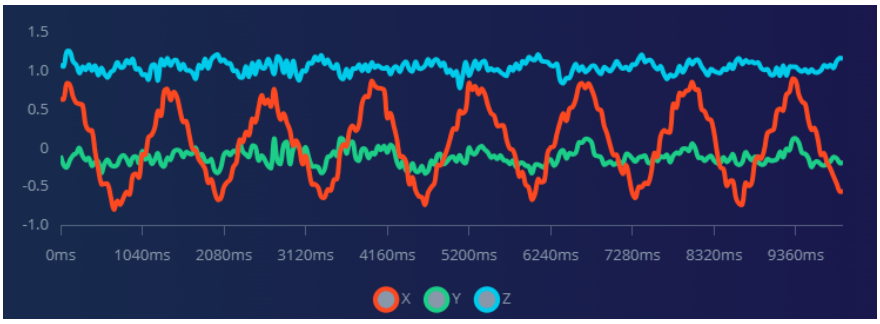

Figura 3.3. Stânga-Dreapta

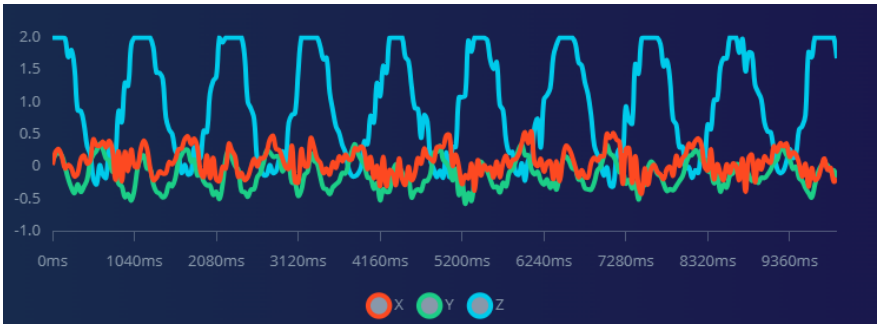

Figura 3.4. Sus-Jos

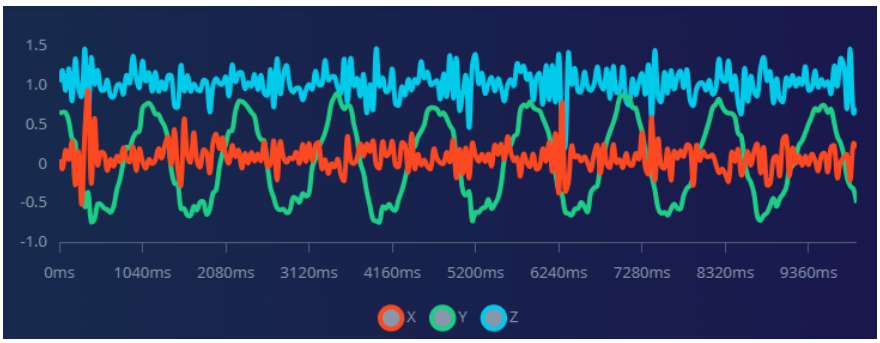

Figura 3.5. Înainte-Înapoi

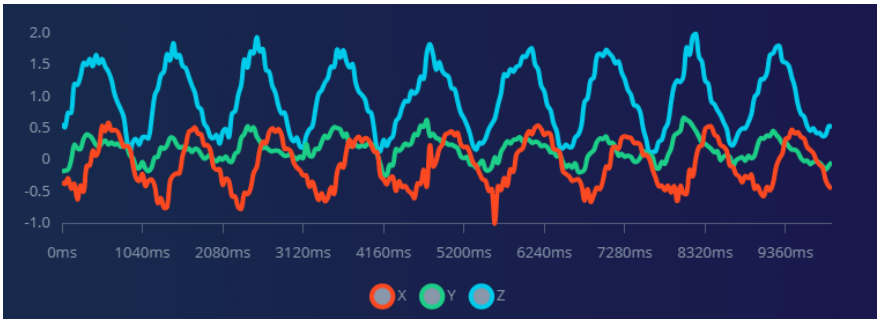

Figura 3.6. Circulară

Din figurile de mai sus se observă cum pentru starea sistemului „Pauză" accelerația rămâne constantă pe toate cele trei axe, pemtru următoarele 3 stari: stânga-dreapta, sus-jos, înainte-înapoi accelerația se modifică considerabil doar pe una din cele 3 axe, X, Z și respectiv Y. Pentru mișcarea mai complexă, circulară sa observă cum accelerația se modifică considerabil pe 2 din cele 3 axe, respectiv X si Z.

3.1.4. Extragerea caracteristicelor spectrale

Procesul din etapa spectrală din Edge Impulse extrage caracteristicile spectrale și are ca scop tramsformarea semnanului de intrare într-o reprezentare numerică care să captureze aspecterelevante ale semnalului în domeniul frecvență. Aceasta este o etapă esențială în prelucrarea semnalelor pentru a extrage informații semnificative care pot fi folosite în algoritmi de clasificare sau detecție.

Edge Impulse parcurge câteva etape in procesul de extragere a caracteristicilor spectrale:

- **Procesarea:** semnalul de intrare este adesea preprocesat pentru a elimina zgomotul și a standardiza amplitudinea sau frecvența. Aceasta poate include filtrarea semnalului sau normalizarea acestuia.

- **Transformarea în domeniul frecvență:** Semnalul preprocesat este apoi transformat din domeniul timpului în domeniul frecvenței folosind transformate de tip FFT (Fast Fourier Transform). Acest proces descompune semnalul în componente sinusoidale cu diferite frecvențe.

- **Extragerea caracteristicilor spectrale:** După transformarea în domeniul frecvenței, sunt extrase caracteristici specifice care descriu semnalul în acest domeniu. Exemple de caracteristici spectrale includ:

  - **Spectrograma:** O reprezentare vizuală a spectrului de putere al semnalului în funcție de timp.

- **Banda de frecvență:** Intervalul de frecvențe în care se concentrează puterea semnalului.
- **Crestături (peaks):** Frecvențele la care semnalul are valori maxime.
- **Energiea spectrală:** Măsura energiei semnalului în diverse benzi de frecvență.

În contextul Edge Impulse, aceste caracteristici spectrale sunt esențiale pentru antrenarea și evaluarea modelelor de ML care rulează pe dispozitive cu resurse limitate, cum ar fi microcontrolerele. Aceste modele pot fi apoi implementate pentru a face inferență pe date noi în timp real, cum ar fi clasificarea activităților bazate pe datele de accelerometru.

3.1.5. Etapa de clasificator din Edge Impulse

În Edge Impulse, etapa de clasificator (classifier) este responsabilă pentru antrenarea și evaluarea modelelor de învățare automată care vor fi utilizate pentru a face predicții pe datele de intrare. Aceasta este o componentă crucială în procesul de dezvoltare a soluțiilor de analiză de date și de luare a deciziilor pe dispozitive integrate (IoT) și la marginea rețelei (edge).

Edge Impulse te lasă să alegi tipul de problemă pe care dorești să o rezolvi, de exemplu clasificare, ceea ce am folosit pentru antrenarea modelului și specificarea setului de date pentru antrenare.

Pentru antrenarea modelului de machine learning am folosit algoritmul de invățare Adam, un algoritm de optimizare avansat. Adam este cunoscut pentru eficiența sa în termeni de performanță și stabilitate în antrenarea rețelelor neuronale.

Pentru algoritmul Adam de învățare am folosit următorii parametri:

- **learning_rate (rata de învățare)**: Controlază cât de mult se ajustează greutățile modelului în funcție de gradienții calculați la fiecare pas de antrenare. În codul tău, este setată la 0.0005.
- **beta_1**: Parametru care controlează rata de decădere a primului moment (media mobilă a gradientului). Este setat la 0.9.
- **beta_2**: Parametru care controlează rata de decădere a celui de-al doilea moment (media mobilă a pătratelor gradientului). Este setat la 0.999.

Beneficiile acestui algoritm de invățare sunt urmatoarele: Adam ajustează rata de învățare pentru fiecare parametru în timpul antrenării, ceea ce ajută la accelerarea procesului de convergență și este unul dintre cei mai utilizați algoritmi de optimizare pentru rețelele neuronale datorită eficienței și fiabilității sale.

În urma antrenării modelului de machine learning am obținut valori ce caracterizează modelul antrenat. Valorile sunt acuratețe de 100%, care este o valoare chiar prea mare și neașteptată, dar asta deoarece am utilizat o valoare de număr de cicli de antrebare de 75 de cicli, iar mișcarile pentru care am antrenat modelul sunt relativ simple. Însă am obținut o valoare de pierderi de 0.3, ceea ce indică faptul ca modelul face încă unele erori.

Matricea de confuzie este un instrument esențial pentru evaluarea performanței unui model de clasificare. Acesta oferă o imagine de ansamblu asupra modelului antrenat, permițând identificarea tipurilor specifice de erori pe care le face modelul.

Matricea de confuzie obținută în urma antrenării modelului este:

	CIRCLE	FRONT-BACK	LEFT-RIGHT	PAUSE	UP-DOWN
CIRCLE	100%	0%	0%	0%	0%
FRONT-BACK	0%	100%	0%	0%	0%
LEFT-RIGHT	0%	0%	100%	0%	0%
PAUSE	0%	0%	0%	100%	0%
UP-DOWN	0%	0%	0%	0%	100%
F1 SCORE	1.00	1.00	1.00	1.00	1.00

Figura 3.7. Matricea de confuzie

Datele din setul de antrenare clasificate după rețeaua neuronală, cu verde sunt cele clasificate corect, iar cu roșu cele clasificate incorect.

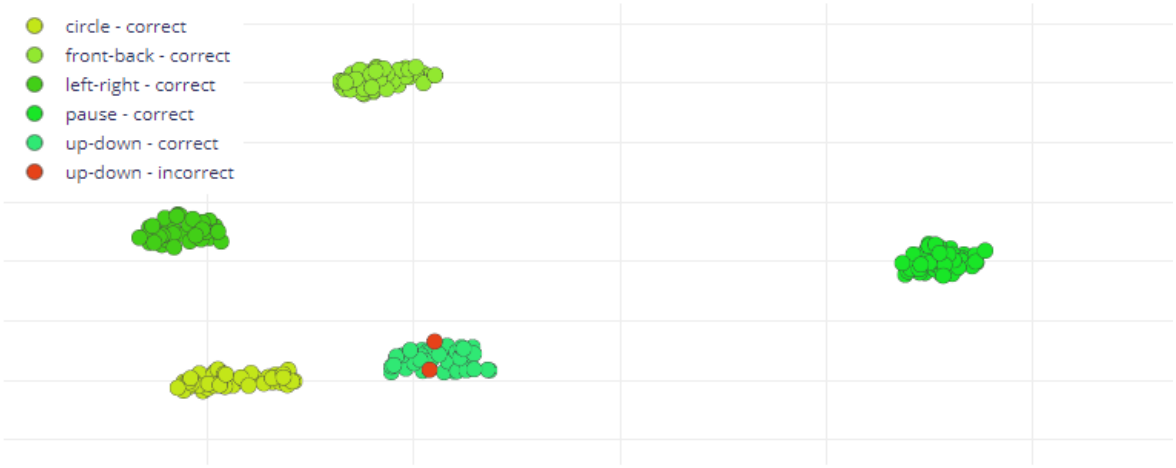

Figura 3.8. Explorarea datelor

De asemenea, Edge Impulse estimează performanțele pentru microcontrolerul pe care îl folosești. Estimările pentru Raspberry Pi Pico RP2040 (Cortex-M0 133MHz) compilat cu Edge Impulse EON sunt:

- **Timpul de deducție:** 3ms.
- **Uilizare de memorie RAM:** 1.4KB.
- **Utilizare memorie Flash:** 16.0 KB.

Aceste valori sunt încurajatoare ținând cont de resursele limitate pe care le are microcontrolerul ales.

De asemenea Edge Impulse îți permite să descarci o arhivă cu o librarie C++ portabilă, care poate fi compilată cu orice compilator modern, astfel am integram modelul de TinyML in proiectul ales.

3.1.6. Integrarea modelului TinyML antrenat pe Pico W

Integrarea modelului pe Raspberry Pi Pico este relativ simplă, datorită faptului că modelul antrenat poate fi exportat ca o librarie C++.

Exemplu de folosire a modelului antrenat:

```
#include "ei_run_classifier.h"
int main() {
    stdio_init_all();
    while(1){
        ei_printf("\nStarting inferencing in 1 seconds...\n");
        sleep_ms(1000);
        cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, 1);
        ei_printf("Sampling...\n");

        float inputBuffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE] = { 0 }; citim valoril

        for (size_t ix = 0; ix < EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE; ix += 3) {
            uint64_t next_tick =
            ei_read_timer_us() + (EI_CLASSIFIER_INTERVAL_MS * 1000);
            mpu.updateData();
            inputBuffer[ix] = mpu.acceleration[0];
            inputBuffer[ix + 1] = mpu.acceleration[1];
            inputBuffer[ix + 2] = mpu.acceleration[2];

            sleep_us(next_tick - ei_read_timer_us());
        }
        signal_t edgeImpulseData;
        int err = numpy::signal_from_buffer(inputBuffer,
        EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &edgeImpulseData);
        if (err != 0) {
            ei_printf("Failed to create signal from buffer (%d)\n", err);
            return 1;
        }
        ei_impulse_result_t outputData = { 0 }ș
        err = run_classifier(&edgeImpulseData, &outputData, debug_nn);
        if (err != EI_IMPULSE_OK) {
            ei_printf("ERR: Failed to run classifier (%d)\n", err);
            return 1;
        }
        float max = 0;
        char state[20];
        for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
            if(outputData.classification[ix].value > max)
            {
                max = outputData.classification[ix].value;
                strcpy(state, outputData.classification[ix].label);
            }
        }
        ei_printf(state);
    }
    return 0;
}
```

3.2. Interfața pentru utilizator

Interfața cu utilizatorul (UI) este un aspect esențial al oricărui proiect deoarece influențează direct experiența utilizatorului final. O UI bine proiectată poate face diferența între succesul și eșecul unui produs.

Interfața proiectată pentru acest proiect ajuta utilizatorul în a practica cateva exerciții cu mainile pentru recuperare sau chiar pentru o incalzire a mușchilor mâinilor.

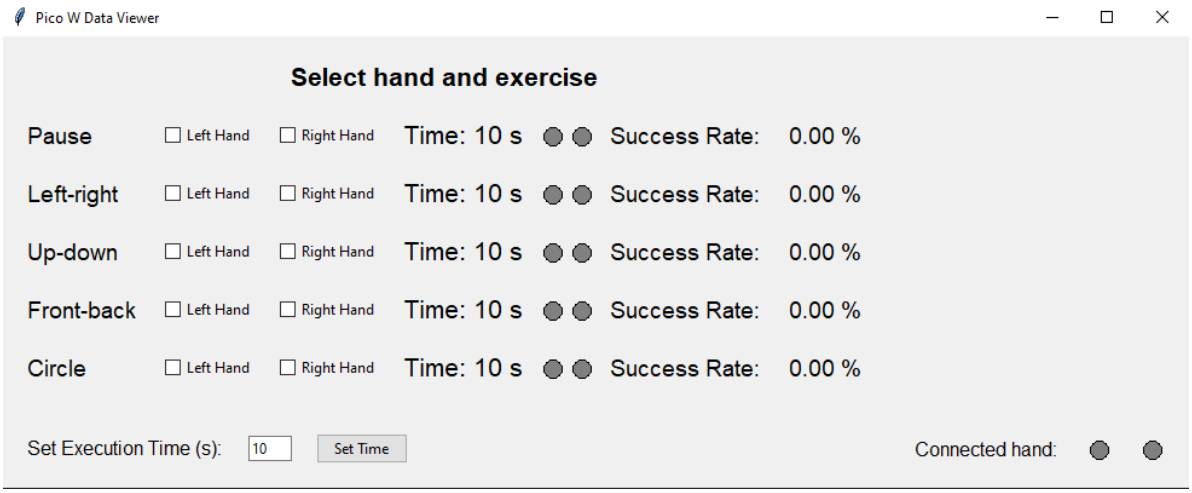

Figura 3.9. Interfață

Interfața este destul de intuitivă, poți seta un timp pentru a practica o anumită mișcare din partea de jos a interfeței.

Pentru fiecare exercițiu poți selecta mâna cu care o sa practici exercițiul dat, la scurgerea timpului se actualizează rata de succes a execuției exercțiului selectat. Becurile din drepul exercitiului selectat se aprind roșu daca mișcarea nu este făcută corect și verde dacă este facut corect, cel din stanga pentru mâna stânga și cel din dreapta pentru mâna dreapta.

În partea dreapta, jos a interfeței sun 2 becuri, care se aprind verde atunci când se conectează oricare din dispozitivele de pe mâna dreaptă sau stângă.

3.2.1. Comunicația dintre dispozitive și interfață

Pe laptop am un server care așteaptă conexiune TCP/IP de la oricare din dispozitive. Serverul ascultă pe adresa 0.0.0.0 pentru noi conexiuni, pe Pico W avem adresa serverului, astfel, cand dispozitivul este porni, acesta incearcă să se conecteze la server.

Partea de server este scrisă in python utilizând biblioteca socket.

Pentru Pico W am folosit bibliotecile:

- "pico/cyw43_arch.h"
- "lwip/dhcp.h"
- "lwip/netif.h"
- "lwip/tcp.h"

Am utilizat comunicație TCP deoarece asigură livrarea fiabilă a datelor prin retransmiterea pachetelor care sunt pierdute sau corupte în timpul transferului.

3.2.2. Integrarea proiectului pe mănușă

Pentru a maximiza comoditatea și a minimiza disconfortul utilizării, am decis să atașăm fiecare dintre cele două dispozitive pe câte o mănușă. Această metodă practică permite utilizatorilor să își păstreze mâinile libere și să se concentreze pe activitățile lor, fără a fi nevoiți să țină dispozitivele constant. Dacă dispozitivele ar fi fost ținute în mână, ar fi putut deveni incomode în timp, afectând astfel eficiența și experiența generală. Montarea pe mănuși asigură o utilizare mai naturală și mai confortabilă, eliminând orice disconfort și asigurând o interacțiune fluidă și neîntreruptă cu tehnologia.

Figura 3.10. Integrarea pe mănușă

Capitolul 4. Testarea aplicației și rezultate experimentale

4.1. Punerea în funcțiune a aplicației

4.1.1. Configurarea Serverului

Partea de server necesită instalarea Python 3.9. În plus, trebuie să fie instalate următoarele biblioteci: `socket` pentru comunicarea în rețea, `threading` pentru gestionarea firelor de execuție concurente și `tkinter` pentru interfața grafică a utilizatorului.

4.1.2. Pregătirea Dispozitivelor de pe Mănuși

Dispozitivele montate pe mănuși trebuie să aibă acumulatorii complet încărcați. Înainte de utilizare, asigurați-vă că porniți întrerupătoarele de pe aceste dispozitive. Pentru ca dispozitivele să comunice corect cu serverul, este esențial să cunoască adresa IP a acestuia.

4.1.3. Utilizarea unei Adrese IP Statice

Pentru a evita problemele legate de modificarea frecventă a adresei IP a serverului, este recomandat să se utilizeze o adresă IP statică. În cazul în care adresa IP a serverului se schimbă, codul sursă de pe dispozitivele montate pe mănuși trebuie actualizat manual cu noua adresă IP. Acest proces implică modificarea codului, recompilarea acestuia și rescrierea fișierului cu extensia `uf2` pe Raspberry Pi Pico W, ceea ce poate fi destul de complicat și consumator de timp.

Prin utilizarea unei adrese IP statice pentru server, se simplifică foarte mult procesul de configurare și întreținere a aplicației, asigurând o conexiune stabilă și fiabilă între dispozitivele de pe mănuși și server. Acest lucru contribuie la o experiență de utilizare mai fluidă și mai eficientă.

4.2. Testarea bibliotecii pentru modulul MPU-9250

Pentru a verifica acuratețea datelor furnizate de senzorul de accelerație, am utilizat un cod dedicat achiziției de date. Acest cod a fost folosit și pentru colectarea datelor necesare antrenării modelului de machine learning. Procesul de testare începe cu inițializarea conexiunii I2C cu senzorul, urmată de activarea senzorului propriu-zis. Datele de la senzor sunt preluate la intervale regulate de 10 milisecunde, asigurând astfel o monitorizare continuă și detaliată a mișcărilor. Această abordare permite obținerea unor seturi de date consistente și precise, esențiale pentru evaluarea și îmbunătățirea performanței modelului de machine learning.

X	Y	Z
0.070312	-0.084961	1.047607
0.067627	-0.083252	1.044434
0.066162	-0.081055	1.040527
0.065918	-0.081055	1.037842
0.074219	-0.081299	1.040283
0.070312	-0.083984	1.039062
0.071045	-0.081055	1.044189
0.068604	-0.083740	1.042480
0.069336	-0.083984	1.041260
0.070068	-0.086182	1.040283
0.069336	-0.084473	1.047363

Tabelul 4.1. Datele de la senzorul de accelerație

Aceste măsurători reflectă valorile obținute pentru axele X, Y și Z ale senzorului la intervale regulate de timp. Informațiile sunt esențiale pentru a evalua comportamentul și răspunsul senzorului în condiții de repaus. Datele colectate oferă o bază solidă pentru analiza ulterioară și pot fi utilizate pentru ajustarea parametrilor sau calibrarea sistemului, asigurând o funcționare optimă în diverse scenarii de utilizare.

4.3. Testarea modelului de TinyML

Pentru evaluarea și testarea modelului meu de TinyML, am ales să utilizez platforma Edge Impulse, datorită modului său simplu și eficient de a testa modelele antrenate. Edge Impulse nu doar că permite o integrare ușoară a modelului, dar și simplifică preluarea datelor de la senzori, necesitând ajustări minime ale codului.

Unul dintre avantajele majore ale utilizării Edge Impulse este reprezentat de interfața sa prietenoasă, care facilitează atât antrenarea modelelor cât și testarea lor. Platforma vine cu un set complet de instrumente care permit gestionarea datelor de antrenament, dezvoltarea și evaluarea modelelor.

În plus, am beneficiat și de interfața de linie de comandă (CLI) oferită de Edge Impulse, care a fost extrem de utilă pentru automatizarea procesului de detectare a frecvenței de achiziție a datelor. Acest lucru mi-a permis să optimizez configurările și să asigur o colectare eficientă a datelor, adaptată nevoilor specifice ale modelului meu.

Tabelul de mai jos prezintă predicțiile modelului TinyML pentru starea "Pauză" în funcție de timestamp-ul înregistrării. Se poate observa că, în majoritatea cazurilor, modelul clasifică corect starea de "Pauză" cu o rată de încredere foarte înaltă, de aproximativ 0.97 sau mai mare. Acest lucru sugerează că modelul este eficient în identificarea și distingerea perioadelor de inactivitate (pauze) în datele de intrare.

Pe masură ce timestamp-ul avansează se observă că variabilele de predicție pentru stările Înainte-Înapoi sau Stânga-Dreapta se modifică foarte puțin până la maxim 0.01, ceea ce indică o rată mare de detecție a starii „Pauză”.

Timestamp (s)	**Circular**	**Înainte-Înapoi**	**Stânga-Dreapta**	**Pauză**	**Sus-Jos**
0	0	0.01	0	0.98	0
1,000	0	0	0	0.99	0
2,000	0	0.01	0	0.99	0
3,000	0	0.01	0	0.98	0
4,000	0	0.01	0.01	0.97	0
5,000	0	0	0	0.99	0
6,000	0	0	0	0.99	0
7,000	0	0	0	0.99	0
8,000	0	0	0	0.99	0
9,000	0	0	0	0.99	0

Tabelul 4.2. Predicții pentru starea „Pauză”

Tabelul de mai jos prezintă predicțiile modelului TinyML pentru starea "Circular" în funcție de timestamp-ul înregistrării. Observăm că modelul atribuie diverse probabilități fiecărei stări enumerate: "Circular", "Înainte-Înapoi", "Stânga-Dreapta", "Pauză" și "Sus-Jos".

Fiind o stare mai complexă, predicțiile pentru starea "Circular" variază semnificativ în funcție de timestamp. La începutul perioadei de observație (timestamp 0), probabilitatea pentru "Circular" este de 0.49, în timp ce pentru "Stânga-Dreapta" este de 0.13 și pentru "Sus-Jos" de 0.36. Pe măsură ce timestamp-ul avansează, se observă fluctuații în distribuția probabilităților

pentru diverse stări.

Timestamp (s)	Circular	Înainte-Înapoi	Stânga-Dreapta	Pauză	Sus-Jos
0	0.49	0	0.13	0	0.36
1,000	0.49	0	0.15	0	0.34
2,000	0.47	0	0.25	0	0.27
3,000	0.43	0.02	0.24	0.01	0.30
4,000	0.42	0.02	0.31	0.03	0.21
5,000	0.52	0	0.10	0	0.36
6,000	0.52	0	0.07	0.01	0.39
7,000	0.48	0.01	0.16	0.01	0.33
8,000	0.53	0	0.12	0	0.34
9,000	0.32	0.32	0.07	0.03	0.26

Tabelul 4.3. Predicții pentru starea „Circular”

4.4. Testarea interfeței cu utilizatorul

Interfața cu utilizatorul joacă un rol crucial într-un proiect deoarece servește ca punte de legătură esențială între utilizator și funcționalitățile aplicației. Ea nu este doar o simplă parte a proiectului, ci un element central care influențează experiența utilizatorului și succesul general al aplicației.

Din figura de mai jos se observă că pentru starea „Pauză” a sistemului, adică menținerea mâinilor nemișcate avem o rată de succes egata cu 95%, deoarece în momentul testării una din mâini a fost mișcată.

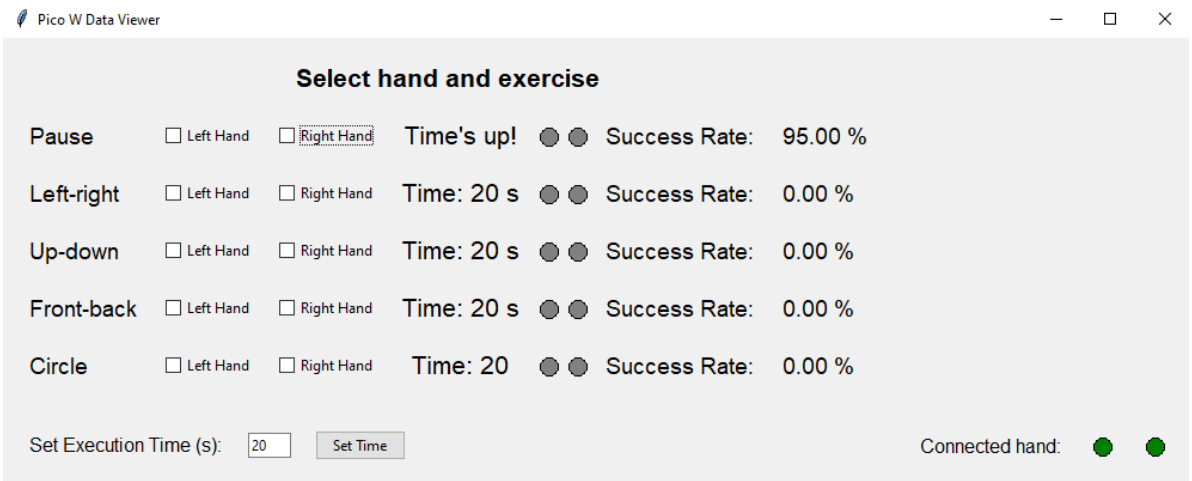

Figura 4.1. Interfața în starea „Pauză”

În figura de mai jos este prezentată starea „Circular” dupa ambele mâini, însă pentru a testa aplicația am efectuat miscarea circulară doar cu una din mâini, cealaltă mână aflânduse in starea de „Pauză” astefl am obținut o rată de succes de 50%.

Figura 4.2. Interfața în starea „Circulară”

4.5. Defecte și înbunătățiri

- Unul din defectele majore ale aplicației este faptul ca datele rețelei sunt hardcodate, la fel și IP-ul serverului. Este nevoie de o soluție pentru ca să nu fie nevoie de o intervenție manuală în a modifica codul și pentru a recompila aplicația.
- O alta problemă poate fi considerată greutatea acumulatorilor pentru alimentarea dispozitivelor de detecție a mișcărilor. Ar fi nevoie de acumulatori mai mici, dar capabili sa alimenteze microcontrolerul Raspberry Pi Pico W care necesita o tensiune de alimentare între 1.8V și 5.5V.
- O altă înbunătățire a aplicației poate fi făcută ajustând partea de interfață grafică, pentru a o face mai intuitivă și ușor de folosit.
- Încă o inbunătățire pentru interfața grafică poate fi considerată, posibilitatea utilizatorului de a-și personaliza experiența.

Concluzii

Această lucrare își propune să ilustreze procesul prin care un obiect simplu și banal poate fi transformat într-unul inteligent, cu funcționalități extinse față de cele inițiale. Acest fenomen face parte din domeniul larg recunoscut al Internet of Things (IoT), care a cunoscut o creștere semnificativă în ultimii ani.

Având în vedere evoluția tehnologică și dinamică a sferei internetului, am ajuns la concluzia că acest tip de aplicații eHealth vor fi un succes în viitorul apropiat, ar ajuta foarte mult la economisirea resurselor importante cum ar fi timpul.

Pe viitor ca posibilă direcție de dezvoltare a aplicației aș dori să integrez si giroscopul în acest proiect eHealth de detectare a mișcarilor. Această înbunătățire ar ajuta la antrenarea unor modele de machine learning capabile să detecteze mișcări mult mai complexe, astfel înbunătățind experiența utilizatorului în utilizarea aplicației.

Bibliografie

[1] P. Healthcare, "About healthsuite | philips healthcare," 2023, accessed: 2024-06-28. [Online]. Available: https://www.philips.ro/healthcare/innovation/about-health-suite

[2] J.-R. Ferrer, "Barcelona's smart city vision: an opportunity for transformation," *Field Actions Science Reports*, vol. Special Issue 16, 2017. [Online]. Available: https://journals.openedition.org/factsreports/4367

[3] G. Digital, "Overview | predix platform | ge digital," 2023, accessed: 2024-06-28. [Online]. Available: https://www.ge.com/digital/documentation/predix-platforms/afs-overview.html

[4] H. Rastogi, "Iot in tesla: Applications, benefits and potential risks," 2022, accessed: 2024-06-28. [Online]. Available: https://analyticssteps.com/blogs/iot-tesla-applications-benefits-and-potential-risks

[5] C. Research, "John deere: Pioneering the future with agricultural iot technology," 2022, accessed: 2024-06-28. [Online]. Available: https://www.counterpointresearch.com/insights/john-deere-connected-agriculture/

[6] Digital HEC Montreal, "The impact of the internet of things (iot) on retail operations: A case study of amazon go," https://digital.hec.ca/en/blog/the-impact-of-the-internet-of-things-iot-on-retail-operations-a-case-study-of-amazon-go/, n.d., accessed: 2024-06-28.

[7] A. T. Ergenç, A. Ün, Yıldırım, A. Küçükyılmaz, and C. Ertürk, "Technical features and functionalities of myo armband: An overview on related literature and advanced applications of myoelectric armbands mainly focused on arm prostheses," *ResearchGate*, 2018. [Online]. Available: https://www.researchgate.net/publication/324889539_Technical_Features_and_Functionalities_of_Myo_Armband_An_Overview_on_Related_Literature_and_Advanced_Applications_of_Myoelectric_Armbands_Mainly_Focused_on_Arm_Prostheses

[8] Apple Inc., "Despre actualizările de securitate ale produselor apple," https://support.apple.com/ro-ro/111918, n.d., accessed: 2024-06-28.

[9] S. M. Ali, S. P. Arjunan, J. Peter, L. Perju-Dumbrava, C. Ding, M. Eller, and S. Raghav, "An example article title from ieee xplore," *IEEE Transactions on Example*, vol. 99, no. 1, pp. 1–10, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10304233

[10] Raspberry Pi Foundation, "Raspberry pi pico documentation," https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html, Accessed: 2024-06-30.

[11] Digi-Key Corporation, "Mpu-9dof click manual," https://www.digikey.jp/htmldatasheets/production/1823810/0/0/1/mpu-9dof-click-manual.html, accesat în iunie 2024, accesat în iunie 2024.

[12] Ebics, "Inter integrated circuit (i2c): A guide to understanding and implementing the communication protocol," 2023, accessed: 2024-06-29. [Online]. Available: https://ebics.net/inter-integrated-circuit-i2c/

[13] J. Postel, "Transmission control protocol," RFC 793, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc793

Anexe

Anexa 1. Biblioteca MPU

1.1. mpu9250.c

```
1       #include <stdio.h>
2   #include <string.h>
3   #include "pico/stdlib.h"
4   #include "hardware/i2c.h"
5   #include <math.h>
6   #include "mpu9250.h"
7   #include "pico/binary_info.h"
8
9   #define I2C_PORT i2c0
10  #define MPU9250_ADDR 0x68
11
12  // MPU9250 registers
13  #define WHO_AM_I 0x75
14  #define PWR_MGMT_1 0x6B
15  #define ACCEL_XOUT_H 0x3B
16
17  void mpu9250_reset() {
18      uint8_t buf[] = {PWR_MGMT_1, 0x00};
19      i2c_write_blocking(I2C_PORT, MPU9250_ADDR, buf, 2, false);
20  }
21
22  uint8_t mpu9250_read_byte(uint8_t reg) {
23      uint8_t val;
24      i2c_write_blocking(I2C_PORT, MPU9250_ADDR, &reg, 1, true);
25      i2c_read_blocking(I2C_PORT, MPU9250_ADDR, &val, 1, false);
26      return val;
27  }
28
29  void read_acceleration(float *acceleration) {
30      uint8_t buffer[6];
31      uint8_t reg = ACCEL_XOUT_H;
32      i2c_write_blocking(I2C_PORT, MPU9250_ADDR, &reg, 1, true);
33      i2c_read_blocking(I2C_PORT, MPU9250_ADDR, buffer, 6, false);
34
35      int16_t raw_ax = (buffer[0] << 8  buffer[1]);
36      int16_traw_ay = (buffer[2] << 8buffer[3]);
37      int16_t raw_az = (buffer[4] << 8 | buffer[5]);
38
39      acceleration[0] = raw_ax / 16384.0f;
40      acceleration[1] = raw_ay / 16384.0f;
41      acceleration[2] = raw_az / 16384.0f;
42  }
43
44  void start_i2c(uint8_t SDA, uint8_t SCL)
45  {
46      i2c_init(I2C_PORT, 100 * 1000);
```

```
47      gpio_set_function(SDA, GPIO_FUNC_I2C);
48      gpio_set_function(SCL, GPIO_FUNC_I2C);
49      gpio_pull_up(SDA);
50      gpio_pull_up(SCL);
51
52      mpu9250_reset();
53
54      uint8_t who_am_i = mpu9250_read_byte(WHO_AM_I);
55      printf("MPU9250 WHO_AM_I: 0x%02X\n", who_am_i);
56  }
```

1.2. mpuObject.h

```
1   #include "pico/stdlib.h"
2
3   #ifndef mpuObject_h
4   #define mpuObject_h
5
6   class mpu9250
7   {
8       public:
9       float acceleration[3];
10
11      mpu9250(uint8_t SDA, uint8_t SCL);
12      void updateData();
13      void printData();
14  };
15
16  #endif
```

1.3. mpuObject.cpp

```
1   #include "mpuObject.h"
2   extern "C" {
3       #include "mpu9250.h"
4   }
5
6   mpu9250::mpu9250(uint8_t SDA, uint8_t SCL)
7   {
8       start_i2c(SDA, SCL);
9       read_acceleration(acceleration);
10  }
11
12  void mpu9250::updateData()
13  {
14      read_acceleration(acceleration);
15  }
16
17  void mpu9250::printData()
18  {
19      printf("Acc. X = %f, Y = %f, Z = %f\n", acceleration[0],
        ↪  acceleration[1], acceleration[2]);
20  }
```

Anexa 2. Achizitia datelor

2.1. getData.cpp

```
#include "mpuObject.h"
#include <stdio.h>
#include "pico/stdlib.h"

mpu9250 mpu(4, 5);  // Creates an mpu object

int main()
{
    stdio_init_all();

    sleep_ms(10000);

    while (1)
    {
        mpu.updateData();

        printf(" %f, %f, %f\n", mpu.acceleration[0],
         ↪  mpu.acceleration[1], mpu.acceleration[2]);
        sleep_ms(10);

    }

    return 0;
}
```

Anexa 3. Clasificarea datelor și transmiterea lor la server

3.1. main.cpp

```
#include <stdio.h>

#include "pico/stdlib.h"
#include "pico/cyw43_arch.h"
#include "lwip/dhcp.h"
#include "lwip/netif.h"
#include "lwip/tcp.h"
#include "ei_run_classifier.h"

#include "mpuObject.h"


#define ssid "DIGI-S26z"
#define pass "pR4TAmCwqN"
//#define ssid "Redmi Note 9"
//#define pass "1122334455"


#define SERVER_IP "192.168.1.136"
#define SERVER_PORT 4242

static struct tcp_pcb *tcp_pcb;
static bool connected = false;
```

```
24
25  static bool debug_nn = false;
26
27  mpu9250 mpu((uint8_t)4, (uint8_t)5);
28
29  static err_t on_connect(void *arg, struct tcp_pcb *tpcb, err_t err) {
30      if (err != ERR_OK) {
31          printf("Connection error\n");
32          return err;
33      }
34      printf("Connected to server\n");
35      connected = true;
36      return ERR_OK;
37  }
38
39  static void send_message(struct tcp_pcb *tpcb, char* message) {
40      err_t err = tcp_write(tpcb, message, strlen(message),
        ↪  TCP_WRITE_FLAG_COPY);
41      if (err == ERR_OK) {
42          tcp_output(tpcb);
43          printf("Message sent: %s\n", message);
44      } else {
45          printf("Error sending message: %d\n", err);
46      }
47  }
48
49  int main() {
50      stdio_init_all();
51
52      if (cyw43_arch_init()) {
53          printf("failed to initialise\n");
54          return 1;
55      }
56
57      cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, 0);
58
59
60      //partea de trimitere date
61      cyw43_arch_enable_sta_mode();
62      while (cyw43_arch_wifi_connect_timeout_ms(ssid, pass,
        ↪  CYW43_AUTH_WPA2_AES_PSK, 20000)) {
63          printf("failed to connect\n");
64      }
65
66      printf("Connected to Wi-Fi\n");
67
68      struct netif *netif = &cyw43_state.netif[CYW43_ITF_STA];
69      const ip4_addr_t *ip = netif_ip4_addr(netif);
70      printf("IP Address: %s\n", ip4addr_ntoa(ip));
71
72      tcp_pcb = tcp_new();
73      if (tcp_pcb == NULL) {
74          printf("Error creating PCB\n");
```

```
75          return 1;
76      }
77
78      ip4_addr_t server_ip;
79      ip4addr_aton(SERVER_IP, &server_ip);
80
81      tcp_connect(tcp_pcb, &server_ip, SERVER_PORT, on_connect);
82
83      //partea de achizitie si clasificare a datelor
84      if (EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME != 3) {
85          ei_printf("ERR: EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME should be
            ↪ equal to 3 (the 3 sensor axes)\n");
86          return 1;
87      }
88
89
90      while (true) {
91          cyw43_arch_poll();
92
93          ei_printf("\nStarting inferencing in 1 seconds...\n");
94          sleep_ms(1000);
95          cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, 1);
96          ei_printf("Sampling...\n");
97
98          float inputBuffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE] = { 0
            ↪ }; // buffer-ul in care citim valorile
99
100         for (size_t ix = 0; ix < EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE;
            ↪ ix += 3) {
101             uint64_t next_tick = ei_read_timer_us() +
                ↪ (EI_CLASSIFIER_INTERVAL_MS * 1000);
102
103             mpu.updateData();
104
105             // citirea valorilor propriu-zise
106             inputBuffer[ix] = mpu.acceleration[0];
107             inputBuffer[ix + 1] = mpu.acceleration[1];
108             inputBuffer[ix + 2] = mpu.acceleration[2];
109
110             sleep_us(next_tick - ei_read_timer_us());
111         }
112
113         // transformarea semnalului nostru pentru modelul antrenat
114         signal_t edgeImpulseData;
115         int err = numpy::signal_from_buffer(inputBuffer,
            ↪ EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &edgeImpulseData);
116         if (err != 0) {
117             ei_printf("Failed to create signal from buffer (%d)\n",
                ↪ err);
118             return 1;
119         }
120
121         // comparatia dintre input cu reteaua antrenata
```

```
122             ei_impulse_result_t outputData = { 0 };
123 
124             err = run_classifier(&edgeImpulseData, &outputData,
                ↪  debug_nn);
125             if (err != EI_IMPULSE_OK) {
126                 ei_printf("ERR: Failed to run classifier (%d)\n", err);
127                 return 1;
128             }
129 
130             float max = 0;
131             char state[20];
132             for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
133                 if(outputData.classification[ix].value > max)
134                 {
135                     max = outputData.classification[ix].value;
136                     strcpy(state, outputData.classification[ix].label);
137                 }
138             }
139 
140             ei_printf("    %s: %.5f\n", state, max);
141             #if EI_CLASSIFIER_HAS_ANOMALY == 1
142                 ei_printf("    anomaly score: %.3f\n",
                    ↪  outputData.anomaly);
143             #endif
144             cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, 0);
145 
146             char message[50] = "Stanga -> ";
147             strcat(message, state);
148 
149             if (connected) {
150                 send_message(tcp_pcb, message);
151             }
152         }
153 
154         return 0;
155     }
```

Anexa 4. Implementarea interfeței grafice

4.1. main.py

```
 1  import socket
 2  import threading
 3  import tkinter as tk
 4  from tkinter import ttk
 5 
 6  HOST = '0.0.0.0'
 7  PORT = 4242
 8 
 9  exercises = ["Pause", "Left-right", "Up-down", "Front-back",
    ↪  "Circle"]
10 
11  total_number = 0
12  correct_number = 0
13  execution_time = 10
```

```
14
15  # Interval de timeout în secunde
16  timeout_interval = 5
17
18  class ServerApp:
19      def __init__(self, master):
20          self.master = master
21          self.master.title("Pico W Data Viewer")
22
23          main_frame = ttk.Frame(self.master, padding="10")
24          main_frame.grid(row=0, column=0, sticky="nsew")
25
26          self.master.grid_rowconfigure(0, weight=1)
27          self.master.grid_columnconfigure(0, weight=1)
28          main_frame.grid_rowconfigure(0, weight=1)
29          main_frame.grid_columnconfigure(0, weight=1)
30
31          select_label = ttk.Label(main_frame, text="Select hand and
             ↪  exercise", font=("Helvetica", 16, "bold"))
32          select_label.grid(row=0, column=0, columnspan=8, pady=10)
33
34          for exercise_type in exercises:
35              self.create_exercise_row(main_frame,
                 ↪  exercises.index(exercise_type) + 1, exercise_type,
36
                                          ↪  exercise_type.lower().replace("-",
                                          ↪  "_"))
37
38          self.timers = {}
39
40          bottom_frame = ttk.Frame(self.master, padding="10")
41          bottom_frame.grid(row=1, column=0, sticky="ew")
42
43          self.master.grid_rowconfigure(1, weight=0)
44
45          time_label = ttk.Label(bottom_frame, text="Set Execution Time
             ↪  (s):", font=("Helvetica", 12))
46          time_label.grid(row=0, column=0, padx=10, pady=10,
             ↪  sticky="e")
47
48          self.time_entry = ttk.Entry(bottom_frame, width=5)
49          self.time_entry.grid(row=0, column=1, padx=10, pady=10,
             ↪  sticky="w")
50          self.time_entry.insert(0, str(execution_time))
51
52          set_time_button = ttk.Button(bottom_frame, text="Set Time",
             ↪  command=self.set_execution_time)
53          set_time_button.grid(row=0, column=2, padx=10, pady=10,
             ↪  sticky="e")
54
55          right_frame = ttk.Frame(self.master, padding="10")
56          right_frame.grid(row=1, column=1, sticky="se")
57          self.master.grid_columnconfigure(1, weight=0)
```

```
58
59          connected_label = ttk.Label(right_frame, text="Connected
             ↪  hand:", font=("Helvetica", 12))
60          connected_label.grid(row=0, column=0, padx=10, pady=10,
             ↪  sticky="e")
61
62          self.left_connection_bulb = tk.Canvas(right_frame, width=20,
             ↪  height=20)
63          self.left_connection_bulb.grid(row=0, column=1, padx=10,
             ↪  pady=10)
64          self.update_bulb(self.left_connection_bulb, "gray")
65
66          self.right_connection_bulb = tk.Canvas(right_frame, width=20,
             ↪  height=20)
67          self.right_connection_bulb.grid(row=0, column=2, padx=10,
             ↪  pady=10)
68          self.update_bulb(self.right_connection_bulb, "gray")
69
70          # Inițializează timerele de timeout
71          self.left_hand_timer = None
72          self.right_hand_timer = None
73
74          self.server_thread = threading.Thread(target=self.run_server)
75          self.server_thread.daemon = True
76          self.server_thread.start()
77
78      def create_exercise_row(self, parent, row, label_text,
         ↪  exercise_type):
79          label = ttk.Label(parent, text=label_text, font=("Helvetica",
             ↪  14))
80          label.grid(row=row, column=0, padx=10, pady=10, sticky="w")
81
82          left_var = tk.BooleanVar()
83          right_var = tk.BooleanVar()
84          left_check = ttk.Checkbutton(parent, text="Left Hand",
             ↪  variable=left_var,
85                                       command=lambda:
                                          ↪  self.toggle_exercise(exercise_type))
86          right_check = ttk.Checkbutton(parent, text="Right Hand",
             ↪  variable=right_var,
87                                        command=lambda:
                                           ↪  self.toggle_exercise(exercise_type))
88          left_check.grid(row=row, column=1, padx=10, pady=10)
89          right_check.grid(row=row, column=2, padx=10, pady=10)
90
91          setattr(self, f"{exercise_type}_left_var", left_var)
92          setattr(self, f"{exercise_type}_right_var", right_var)
93
94          timer_label = ttk.Label(parent, text=f"Time: {execution_time}
             ↪  s", font=("Helvetica", 16))
95          timer_label.grid(row=row, column=3, padx=10, pady=10)
96          setattr(self, f"{exercise_type}_timer_label", timer_label)
97          setattr(self, f"{exercise_type}_time_remaining",
             ↪  execution_time)
```

```
 98
 99          left_bulb = tk.Canvas(parent, width=20, height=20)
100          right_bulb = tk.Canvas(parent, width=20, height=20)
101          left_bulb.grid(row=row, column=4, pady=10)
102          right_bulb.grid(row=row, column=5, pady=10)
103          setattr(self, f"{exercise_type}_left_bulb", left_bulb)
104          setattr(self, f"{exercise_type}_right_bulb", right_bulb)
105          self.update_bulb(left_bulb, "gray")
106          self.update_bulb(right_bulb, "gray")
107
108          success_rate_var = tk.DoubleVar(value=0.0)
109          success_rate_label = ttk.Label(parent, text="Success Rate:",
              ↪ font=("Helvetica", 14))
110          success_rate_label.grid(row=row, column=6, padx=10, pady=10,
              ↪ sticky="e")
111          success_rate_value = ttk.Label(parent,
              ↪ text=f"{success_rate_var.get():.2f} %",
              ↪ font=("Helvetica", 14))
112          success_rate_value.grid(row=row, column=7, padx=10, pady=10,
              ↪ sticky="w")
113          setattr(self, f"{exercise_type}_success_rate_var",
              ↪ success_rate_var)
114          setattr(self, f"{exercise_type}_success_rate_value",
              ↪ success_rate_value)
115
116      def toggle_exercise(self, exercise_type):
117          for other_exercise_type in ["left_right", "up_down",
              ↪ "front_back", "pause"]:
118              if other_exercise_type != exercise_type:
119                  self.reset_exercise(other_exercise_type,
                      ↪ reset_selection=True)
120
121          left_var = getattr(self, f"{exercise_type}_left_var")
122          right_var = getattr(self, f"{exercise_type}_right_var")
123          if left_var.get() or right_var.get():
124              self.start_timer(exercise_type)
125          else:
126              self.reset_exercise(exercise_type, reset_selection=False)
127
128      def start_timer(self, exercise_type):
129          if getattr(self, f"{exercise_type}_time_remaining") ==
              ↪ execution_time:
130              self.update_timer(exercise_type)
131
132      def reset_exercise(self, exercise_type, reset_selection):
133          global correct_number
134          global total_number
135
136          correct_number = 0
137          total_number = 0
138          setattr(self, f"{exercise_type}_time_remaining",
              ↪ execution_time)
139          timer_label = getattr(self, f"{exercise_type}_timer_label")
```

```
140             timer_label.config(text=f"Time: {execution_time} s")
141
142             if exercise_type in self.timers:
143                 self.master.after_cancel(self.timers[exercise_type])
144                 del self.timers[exercise_type]
145
146             left_bulb = getattr(self, f"{exercise_type}_left_bulb")
147             right_bulb = getattr(self, f"{exercise_type}_right_bulb")
148             self.update_bulb(left_bulb, "gray")
149             self.update_bulb(right_bulb, "gray")
150
151             success_rate_var = getattr(self,
                ↪  f"{exercise_type}_success_rate_var")
152             success_rate_var.set(0.0)
153
154             left_var = getattr(self, f"{exercise_type}_left_var")
155             right_var = getattr(self, f"{exercise_type}_right_var")
156             left_var.set(False)
157             right_var.set(False)
158
159         def update_timer(self, exercise_type):
160             time_remaining = getattr(self,
                ↪  f"{exercise_type}_time_remaining")
161             if time_remaining > 0:
162                 time_remaining -= 1
163                 setattr(self, f"{exercise_type}_time_remaining",
                    ↪  time_remaining)
164                 timer_label = getattr(self,
                    ↪  f"{exercise_type}_timer_label")
165                 timer_label.config(text=f"Time: {time_remaining} s")
166                 self.timers[exercise_type] = self.master.after(1000,
                    ↪  self.update_timer, exercise_type)
167             else:
168                 timer_label = getattr(self,
                    ↪  f"{exercise_type}_timer_label")
169                 timer_label.config(text="Time's up!")
170
171                 exercise_type_lower = exercise_type.lower().replace("-",
                    ↪  "_")
172                 success_rate_var = getattr(self,
                    ↪  f"{exercise_type_lower}_success_rate_var")
173                 success_rate_value = getattr(self,
                    ↪  f"{exercise_type_lower}_success_rate_value")
174
175                 if total_number > 0:
176                     success_rate = (correct_number / total_number) * 100
177                 else:
178                     success_rate = 0.0
179
180                 success_rate_var.set(success_rate)
181                 success_rate_value.config(text=f"{success_rate:.2f} %")
182
183                 left_bulb = getattr(self, f"{exercise_type}_left_bulb")
```

```
184             right_bulb = getattr(self, f"{exercise_type}_right_bulb")
185             self.update_bulb(left_bulb, "gray")
186             self.update_bulb(right_bulb, "gray")
187 
188             left_var = getattr(self, f"{exercise_type}_left_var")
189             right_var = getattr(self, f"{exercise_type}_right_var")
190             left_var.set(False)
191             right_var.set(False)
192 
193     def set_execution_time(self):
194         global execution_time
195         try:
196             new_time = int(self.time_entry.get())
197             execution_time = new_time
198             for exercise_type in exercises:
199                 exercise_type_lower =
                     ↪ exercise_type.lower().replace("-", "_")
200                 setattr(self,
                     ↪ f"{exercise_type_lower}_time_remaining",
                     ↪ execution_time)
201                 timer_label = getattr(self,
                     ↪ f"{exercise_type_lower}_timer_label")
202                 timer_label.config(text=f"Time: {execution_time}")
203         except ValueError:
204             print("Invalid time value")
205 
206     def run_server(self):
207         with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
208             s.bind((HOST, PORT))
209             s.listen()
210             print(f"Server listening on {HOST}:{PORT}")
211 
212             while True:
213                 conn, addr = s.accept()
214                 thread = threading.Thread(target=self.handle_client,
                     ↪ args=(conn, addr))
215                 thread.start()
216 
217     def handle_client(self, conn, addr):
218         print(f"Connected by {addr}")
219 
220         while True:
221             data = conn.recv(1024)
222             if not data:
223                 break
224             message = data.decode()
225             print(f"Received from {addr}: {message}")
226             self.process_message(addr, message)
227 
228         print(f"Disconnected from {addr}")
229         conn.close()
230 
231     def process_message(self, addr, message):
```

```
232             global total_number
233             global correct_number
234             try:
235                 hand, received_exercise = message.split('->')
236                 hand = hand.strip().lower()
237                 received_exercise =
                     ↪  received_exercise.strip().lower().replace("-", "_")
238
239                 for exercise_type in exercises:
240                     exercise_type_lower =
                         ↪  exercise_type.lower().replace("-", "_")
241                     left_var = getattr(self,
                         ↪  f"{exercise_type_lower}_left_var")
242                     right_var = getattr(self,
                         ↪  f"{exercise_type_lower}_right_var")
243                     left_bulb = getattr(self,
                         ↪  f"{exercise_type_lower}_left_bulb")
244                     right_bulb = getattr(self,
                         ↪  f"{exercise_type_lower}_right_bulb")
245
246                     if hand == "stanga":
247                         if left_var.get():
248                             correct_exercise = exercise_type_lower ==
                                 ↪  received_exercise
249                             total_number += 1
250                             if correct_exercise:
251                                 correct_number += 1
252                             self.update_bulb(left_bulb, "green" if
                                 ↪  correct_exercise else "red")
253                         else:
254                             self.update_bulb(left_bulb, "gray")
255                     elif hand == "dreapta":
256                         if right_var.get():
257                             correct_exercise = exercise_type_lower ==
                                 ↪  received_exercise
258                             total_number += 1
259                             if correct_exercise:
260                                 correct_number += 1
261                             self.update_bulb(right_bulb, "green" if
                                 ↪  correct_exercise else "red")
262                         else:
263                             self.update_bulb(right_bulb, "gray")
264
265                 # Resetează timerul de timeout pentru mâna respectivă
266                 if hand == "stanga":
267                     if self.left_hand_timer:
268                         self.master.after_cancel(self.left_hand_timer)
269                     self.left_hand_timer =
                         ↪  self.master.after(timeout_interval * 1000,
                         ↪  self.timeout_hand, "stanga")
270                     self.update_bulb(self.left_connection_bulb, "green")
271                 elif hand == "dreapta":
272                     if self.right_hand_timer:
```

```
273                     self.master.after_cancel(self.right_hand_timer)
274                 self.right_hand_timer =
                     ↪ self.master.after(timeout_interval * 1000,
                     ↪ self.timeout_hand, "dreapta")
275                 self.update_bulb(self.right_connection_bulb, "green")
276
277         except ValueError:
278             print(f"Invalid message format: {message}")
279
280     def timeout_hand(self, hand):
281         if hand == "stanga":
282             self.update_bulb(self.left_connection_bulb, "red")
283         elif hand == "dreapta":
284             self.update_bulb(self.right_connection_bulb, "red")
285
286     def update_bulb(self, bulb, color):
287         bulb.delete("all")
288         bulb.create_oval(5, 5, 20, 20, fill=color)
289
290
291 def main():
292     root = tk.Tk()
293     app = ServerApp(root)
294     root.mainloop()
295
296
297 if __name__ == "__main__":
298     main()
```