Cuprins

Listă de figuri **iii**

1. Introducere **1**
1.1 Actualitatea și necesitatea temei 1
1.2 Problema propusă 2
1.3 Obiectivele lucrării 3
1.4 Structura lucrării 3

2. Arhitecura hardware **5**
2.1 Placa de dezvoltare Arduino Uno 5
2.2 Pachetul de sinzori accelerometru, giroscop și magnetometru AltIMU-10 v4 7
2.3 Conectarea AltIMU-10 v4 la Arduino Uno 8

3. Arhitectura software **10**
3.1 Platforma de dezvoltare Unity 3D 10
3.1.1 Editorul Unity 11
3.1.2 Crearea unei animații în Unity3D 12
3.1.3 Controlul unei animații în Unity 12
3.2 Limbajul de programare specific plăcii de dezvoltare Arduino 14
3.2.1 Structura limbajului arduino 14
3.2.2 Vizulaizarea datelor în ediorul arduino 15
3.3 Platforma Matlab 15
3.3.1 Structura sistemului MATLAB 17
3.4 Conexiunea dintre Arduino și calculator 18
3.4.1 Conexiunea dintre placa de dezvolatre Arduino și Unity 19
3.4.2 Conexiunea dintre placa de dezvolatre Arduino și Matlab 20

4. Convențiile orientării **21**
4.1 Unghiurile lui Euler 21
4.2 Cuaternionii 22
4.2.1 Norma unui cuaternion 23
4.2.2 Înmulțirea 24
4.2.3 Vectorul de rotație 24
4.2.4 Derivata unui cuaternion 25

4.2.5 Determinarea unghiurilor lui Euler pe baza cuaternionilor . . . 25

5. Filtrarea datelor **26**
5.1 Filtrul Kalman . . . 26
5.1.1 Predicția . . . 26
5.1.2 Măsurarea . . . 27
5.1.3 Asocierea datelor . . . 27
5.1.4 Corecție . . . 28
5.1.5 Vectorul de stare . . . 28
5.1.6 Predicția modelului și măsurătorile modelului orientării . . . 28
5.2 Dezavantajele folosirii filtrului Kalman liniar . . . 29
5.3 Filtrul Kalman extins . . . 29
5.3.1 Vectorul de stare . . . 29
5.3.2 Formularea modelului predicției . . . 29
5.3.3 Formularea modelului măsurătorilor . . . 30
5.3.3.1 Maparea parametrilor accelerometrului . . . 31
5.3.3.2 Maparea parametrilor magnetometrului . . . 31
5.3.4 Matricele de covarianță . . . 32
5.3.5 Ecuațiile filtrului Kalman extins . . . 33

6. Determinarea orientării pe baza filtrului AHRS **34**
6.1 Filtrul AHRS . . . 34
6.2 Principiile folosite în filtrul AHRS . . . 35
6.2.1 Orientarea pe baza vitezei unghiulare . . . 35
6.2.2 Orientarea pe baza vectorului măsurătorilor . . . 35
6.2.3 Fuziunea celor două metode de determianre a orientării . . . 39
6.2.4 Compensarea deformării magnetice . . . 40
6.2.5 Controlul implicării erorii furnizate de giroscop . . . 41

7. Rezultate experimentale **43**
7.1 Analiza datelor citite de senzorii inerțiali . . . 44
7.1.1 Analiza parametrilor în regim static . . . 44
7.1.2 Analiza datelor de intre în regim dinamic . . . 45
7.2 Rezultatele algoritmilor de determinare a orientării . . . 46
7.2.1 Rezultatele determinării orientării pe baza unghiurilor lui Euler și a filtrului AHRS . . . 47
7.2.1.1 Rezultatele determinării orientării în regim static . . . 47
7.2.1.2 Rezultatele determinării orientării în regim dinamic . . . 49

8. Concluzii şi dezvoltări ulterioare **52**
8.1 Concluziile proiectului . . . 52
8.2 Direcții de dezvoltare ulterioară . . . 53

Listă de figuri

2.1 Explicarea pinilor plăcii Arduino Uno 6
2.2 Altimu-10 v4 8
2.3 Conectarea AltIMU-10 v4 la Arduino Uno 9
3.4 Animație Unity 11
3.5 Editorul Unity 12
3.6 Uneltele *Transform* și *Rigidbody* 13
3.7 Confirmarea atașării scriptului 13
3.8 Datele procesate de arduino și afișate prin modulul serial 16
3.9 Editorul Matlab 18
3.10 Conexiunea dintre placa Arduino și calculator 19
3.11 Conexiunea dintre placa Arduino și calculator 20
4.12 Reprezentarea unghiurilor Tait-Bryan. Unghiul Ψ corespunde rotației în jurul axei Z. Unghiul Φ corespunde rotației în jurul axei X. Unghiul Θ corespunde rotației în jurul axei Y 22
4.13 Orientarea vectorului ${}^{A}\hat{r}$ ce se rotește în cu unghiul Θ în sistemul de coordonate A raportat la sistemul de coordonate B 23
6.14 Schema bloc de determinare a orientării pe baza fuzionării celor două metode 40
6.15 Schema bloc de determinare a orientării pe baza fuzionării celor două metode, compensarea distorsiunilor magnetice(Group1) și compensarea erorii giroscopului (Group2). 42
7.16 Parametrii giroscopului în regim staționar 44
7.17 Parametrii accelerometrului în regim staționar 45
7.18 Datele furnizate de giroscop în regim dinamic 45
7.19 Datele furnizate de accelerometru în regim dinamic 46
7.20 Reprezentarea unghiului în raport cu axa x 48
7.21 Reprezentarea unghiului în raport cu axa y 48
7.22 Reprezentarea unghiului în raport cu axa z 49
7.23 Reprezentarea orientării în raport cu axa x 50
7.24 Reprezentarea orientării în raport cu axa y 50
7.25 Reprezentarea orientării în raport cu axa z 51
7.26 Reprezentarea orientării în 3D 51

1. Introducere

Actualitatea și necesitatea temei
Problema propusă
Obiectivele lucrării
Structura lucrării

1.1 Actualitatea și necesitatea temei

Studiile privind orientarea corpurilor au reprezentat și reprezintă o atracție pentru o serie de domenii precum navigație [4], industria aerospațială [9], [12], industria animațiilor [20], jocurile video, ergonomie, sport [21] sau medicină [22]. Aceste studii sau bazat pe captarea datelor inerțiale ale corpului și implementarea acestora în algoritmi matematici ce determină orientarea acestuia.

Un domeniu în care sunt folosite intens aceste studii este industria aerospațială, unde dispozitivele inerțiale (IMU-Inertial Measurements Unit) sunt principalele componente ale sistemului de navigație și orientare. Prelucrarea măsurătorilor captate de acestea determinând viteza, orientarea și forțele gravitaționale ce acționează asupra aeronavei. Un dispozitiv IMU este compus dintr-un accelerometru ce realizează măsurători ale accelerației pe axele spațiale, un giroscop ce determină rotațiile în jurul celor trei axe și un magnetometru ce este folosit la calibrare si la filtrarea erorilor [12].

Alte proiecte în care se folosesc intens studiile de determinare a orientării în spațiul tri-dimensional, sunt proiectele în care sunt analizate mișcările corpului uman in cadrul mersului. În cadrul acestor proiecte sa folosit un pachet de senzori (IMU) precum cei folosiți în industria aerospațială, ce generează date ce au ca scop urmărirea poziției corpului in cadrul deplasării [7], [8].

1.2 Problema propusă

Analizând aplicațiile realizate pe baza orientării se observă că majoritatea aplicațiilor folosesc un pachet de senzori IMU (Inertial Measurements Unit). Acest dispozitiv hardware are la bază un senzor de giroscop tri-axial și un senzor de accelerometru tri-axial și mai nou s-a implementat și un senzor de magnetometru tri-axial. Această îmbunătățire adusă acestei structuri hardware conduce la o structură hibridă numită MARG (Magnetic, Angualr Rate, Gravity), ce are performanțe mult mai bune. Având în vedere că bazele studiilor bazate pe senzorii inerțiali au fost puse folosind senzorul IMU, în literatura de specialitate ambii senzori poartă această denumire.

Pentru o imagine mai clară a determinării orientării, pe baza parametriilor determinați de un senzor inerțial se poate realiza o detaliere a parametriilor măsurați de fiecare componentă a acestuia. Accelerometrul măsoară, precum îi spune și numele, accelerația incluzând și componenta gravitațională ce influențează măsurătorile accelerometrului [11]. Giroscopul realizează detectarea vitezei ungiulare realizate de corp, viteza unghiulară fiind asociată în acest caz cu rotația corpului în jurul unei axe de coordonate. Cea mai recentă componentă a senzorilor IMU, magnetometrul realizează măsurători asupra intensității și direcției câmpului magnetic, în special ale câmpului magnetic exercitat de Pământ asupra senzorului [22].

Pentru a determina orientarea în spațiul tri-dimensional este nevoie ca senzorul inerțial să realizeze o măsurare pentru fiecare axă spațială. Astfel pentru o integrare eficientă atât hardware cât și din punct de vedere al implementării software se folosesc senzori ce realizează măsurători pe toate cele trei axe spațiale [14].

Accelerometrul tri-axial este un traducător 3D ce e compus din trei traducătoare dispuse conform dispunerii sistemului de coordonate spațial. Din punct de vedere electronic este un dispozitiv ce convertește accelerația într-un semnal electric, putând măsura atât accelerația statică cât și accelerația dinamică, menționând faptul că accelerația dinamică este afectată și de alte forțe fizice, nu doar de gravitație precum e afectată accelerația statică. În cadrul măsurătorilor accelerometrului intervine un factor perturbator foarte important generat de influența gravitației asupra accelerației, aceasta influențând în principal accelerația pe axa z. Datele furnizate de accelerometru sunt organizate sunt sub forma unui vector cu trei elemente (1.1), [21].

$$a = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix} \tag{1.1}$$

Giroscopul tri-axial determină vitezele unghiulare în jurul celor trei axe spațiale, dar măsurătorile sale sunt afectate de erori ce vor trebui compensate pentru determinarea

unei orientări corecte. Vectorul ieșirilor giroscopului este notat astfel (1.2), [4].

$$S_w = \begin{bmatrix} w_x & w_y & w_z \end{bmatrix} \tag{1.2}$$

Magnetometrul tri-axial realizează detectarea direcției și intensității câmpului magnetic, parametri returnați de acesta fiind de forma expresiei (1.3). În cazul magnetometrului trebuie specificat că generează erori prin bruiajul realizat de componentele magnetice din vecinătatea sa [5], [15].

$$m = \begin{bmatrix} m_x & m_y & m_z \end{bmatrix} \tag{1.3}$$

1.3 Obiectivele lucrării

Există multe metode ce descriu orientarea unui obiect pe baza parametrilor citiți de un senzor IMU și raportat la sistemul de referință terestru. În acest proiect sunt folosite metoda unghiurilor lui Euler și cuaternionii relativi la orizontală și la câmpul magnetic [13], determinarea direcției prin matricea cosinusului (DCM), calcularea orientării pe baza filtrării Kalman sau metoda filtrului AHRS (Altittude and Heading Reference System) [9], [12].

Având în vedere existența unui număr mare de algoritmi matematici trebuie făcută o evaluare în vederea alegerii celui mai eficient. Încă din etapa de documentare se observă că referințele metodei matricei DCM [18] prezintă performanțe slabe, dar această metodă împreună cu metoda unghiurilor lui Euler [6] returnează performanțe satisfăcătoare cu un timp de răspuns foarte bun. Cele mai bune referințe sunt asupra metodei filtrului Kalman [**?**] aplicat asupra cuaternionilor relativi la orizontală și la câmpul magnetic [8], dar prezintă un volum de calcul ridicat ce face o implementare dificilă și un timp de răspuns ridicat. Referințe bune există și asupra modelului AHRS [1], [3] ce a fost implementat cu scces în aplicații cu un timp de răspuns scurt, generând performanțe asemănătoare filtrării Kalman.

1.4 Structura lucrării

1. Introducere

În acest capitol se realizează o introducere în domneniul orientării 3D. Expunându-se domeniile în care este folosit acest principiu, tipurile de proiecte și actualitatea lor în aplicațiile curente. Este prezentată structura senzorilor ce captează parametrii ce determină orientarea și algoritmii matematici ce rezolvă problema expusă.

2. Arhitectura hardware

Acest capitol expune structura hardware folosită în scopul măsurării parametrillor de accelerometru, giroscop și magnetometru. Structura hardware este alcătuită dintr-o placă de dezvoltare Arduino Uno detaliată în acest capitol, la care se conectează un pachet de senzori inerțiali IMU.

3. Arhitectura software

Capitolul 3 descrie componentele software ce se doresc a fi folosite în scopul realizării unui algoritm ce calculează orientarea pe baza parametriilor inerțiali, orienatre implementată într-o aplicație 3D.

4. Convențiile orientării

Acest capitol face o introducere în cadrul principilor orientării punându-se accent pe metoda de descriere a spațiului tri-dimensional sub forma unghiurilor lui Euler, respectiv sub forma cuaternionilor.

5. Filtrarea datelor

Capitolul 5 descrie modalitățile de filtare a datelor provenite de la senzor, cu scopul filtrării perturbațiilor și compensării erorilor, pentru determinarea unei orientări precise.

6. Determinarea orientării pe baza filtrului AHRS

Acest capitol detaliază un model de filtru derivat din filtrul Kalman expus în capitolul anterior, pe baza căruia se face o orientare precisă și mai rapidă decât filtrarea Kalman.

7. Rezultate experimentale

Utilitatea acestui capitol este de a observa rezultatele acțiunilor produse de implemenatre a algoritmilor software de citire și analiză a datelor inerțiale, procesarea acestora și expunerea modelului orientrării.

8. Concluzii și dezvoltări ulterioare

Capitolul final al proiectului face o concluzionare a studiilor și rezultatelor obținute în cadrul proiectului, observându-se o vedere generală a soluției de rezolvare a problemei. Pe baza cunoștințelor dobândite și analizării necesităților și tendințelor actuale se face o expunere a viitoarelor dezvoltări ale proiectului.

2. Arhitecura hardware

Placa de dezvoltare Arduino Uno
Pachetul de sinzori accelerometru, giroscop și magnetometru AltIMU-10 v4
Conectarea AltIMU-10 v4 la Arduino Uno

Componetele dispozitivului hardware sunt compuse dintr-o placa de dezvoltare Arduino Uno, care are ca scop citirea parametrilor furnizați de pachetul de senzori AltImu-10 v4 ce conține senzori de accelerometru, giroscop și magnetometru.

2.1 Placa de dezvoltare Arduino Uno

Arduino UNO este o platformă de procesare disponibilă tuturor utilizatorilor, bazată pe software și hardware flexibil și simplu de folosit. Constă intr-o platformă de mici dimensiuni (6.8 cm / 5.3 cm – în cea mai des întâlnită variantă) construită în jurul unui procesor de semnal și este capabilă de a prelua date din mediul înconjurator printr-o serie de senzori și de a efectua acțiuni asupra mediului prin intermediul luminilor, motoarelor, servomotoare, și alte tipuri de dispozitive mecanice. Procesorul este capabil să ruleze cod scris într-un limbaj de programare care este foarte similar cu limbajul C++ [23].

Specificații :

- microcontroler: ATmega328;
- tensiune de lucru:5V ;
- tensiune de intrare (recomandat):7-12V;
- tensiune de intrare (limita):6-20V;
- pini digitali:14(6 generând ieșlire PWM);
- pini analogici:6;
- intensitatea curentului de ieșire:40 mA;

- intensitate curentului de ieșire pe 3.3V: 50 mA;
- memoria flash: 32 KB (ATmega328) 0.5 KB pentru bootloader;
- EEPROM: 1 KB (ATmega328);
- Frecvența generatorului de impulsuri:16 MHz.

O descrierea detaliată a plăcii de dezvoltare Arduino Uno se poate face pe baza imaginii următoare, unde sunt evidențiate componentele principale.

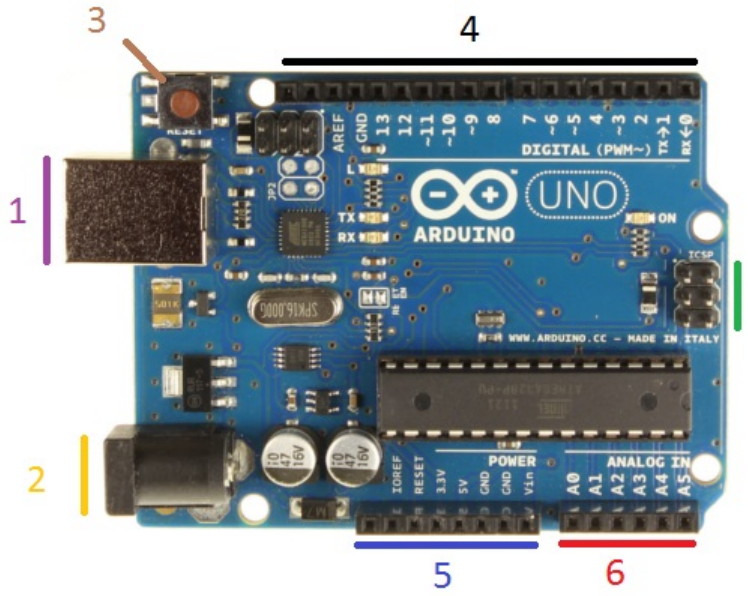

FIGURA 2.1: Explicarea pinilor plăcii Arduino Uno

Componentele principale vizibile în imaginea următoare au caracteristici ce sunt necesare în cadrul realizării unei structuri hardware pe baza acestei platforme electronice. Secțiunile evidențiate în imagine sunt:

1. Conector de USB de Tip B (Mamă), pentru alimentarea cu 5V și transmisii de date;
2. Port de alimentare placă Arduino, recomandat este ca tensiunea de alimentare să fie între 7V-12V, putându-se formula o serie de observații:
   - în cazul în care se alimentează Arduino Uno prin conectorul USB din altă sursă în afară de PC(de exemplu un încărcător), trebuie măsurată tensiunea de alimentare deoarece alimentarea de la conectorul USB nu trece prin stabilizatorul de 5V al plăcii;
   - în momentul când placa Arduino Uno este conectată la PC pentru programare sau comunicare prin intermediul monitorizării seriale este recomandată deconectarea alimentării externe din conectorul 2;

3. Buton de reset, resetează placa fără însă a se pierde programul scris, acesta reîncepând de la funcția *void setup()* ;
4. Șir pini digitali, conține 14 intrări/ieșiri din care 6 pot fi folosite ca ieșiri PWM (3,5,6,9,10,11), o detaliere se face a acstora se face astfel:
   - pinii 0(rx), 1(tx) sunt folosiți pentru a primi (rx) și transmite (tx) date. În cazul în care programul folosește funcția de comunicare serială (se poate indentifica în program prin „Serial.begin„) aceste ieșiri nu pot fi folosite;
   - pinul 13 are conectat un led (L pe placă) în serie cu o rezistentă. LED-ul ne indică starea pinului: aprins pentru HIGH, și stins pentru LOW;
   - GND este conectat la masă.
5. Șir pini alimentare:
   - IOREF: poate măsura tensiunea la care operează microcontrolerul (3v3 sau 5V)
   - RESET: resetează placa când acesta este conectat la masă (GND)
   - Vin: rin intermediul acestui pin se poate alimenta placa Arduino Uno, respectându-se tensiunea recomandată de alimentare (în cazul în care alimentăm placa prin acest pin, tensiunea trece prin stabilizator).
6. Șir pini analogici: A0...A5. Valoarea acestori pini se poate citi cu funcția *analogRead()*, din citire rezultă numere între 0-1023 pentru 0V – 5V.

2.2 Pachetul de sinzori accelerometru, giroscop și magnetometru AltIMU-10 v4

AltIMU-10 v4 este un pachet de senzori inerțiali(IMU) ce este compus dintr-un giroscop tri-axial L3GD20H, un accelerometru tri-axial LSM303D, un magnetometru tri-axial și un barometru digital LPS25H. Primii trei parametrii ai acestui pachet de senzori, reprezintă componentele inerțiale ce pot determina orientarea și poziția sistemului (AHRS), iar citirea presiuni poate fi convertită în altidudine. Printr-un algoritm matematic și un microcontroler sau microprocesor pe baza măsurătorilor făcute de Altimu-10 v4 se poate determina orientarea și poziția unui corp.

Specificații:

- dimensiuni:25mm x 13mm x 3mm ;
- greutate(fără pini):0.8g;
- tensiunea de operare:2.5V -5.5V;

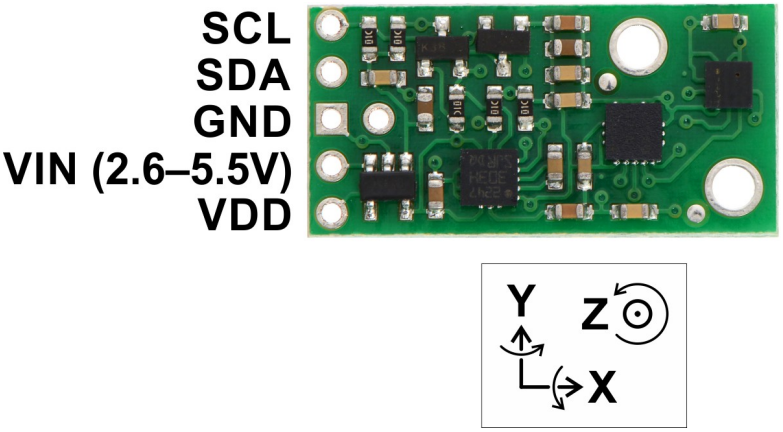

FIGURA 2.2: Altimu-10 v4

- intensitatea curentului de iesire:6mA;
- formatul ieșirii(I^2C):
  - ieșirea giroscopului:citirea unei axe se face pe 16 biți;
  - ieșirea accelerometrului:citirea unei axe se face pe 16 biți;
  - ieșirea magnetometrului:citirea unei axe se face pe 16 biți;
  - ieșirea barometrului:citirea presiunii se face pe 24 biți;
- domeniul sensibilitatății:
  - giroscop:± 245,± 500 rad/s;
  - accelerometru:± 2,± 4,± 6,± 8 g;
  - magnetometrul:± 2,± 4,± 8 gauss;
  - barometrul:260 mbar- 1260 mbar;
- Conectarea: Pentru folosirea acestui pachet de senzori trebuie folosite cel puțin patru conexiuni legate la pinii:VIN, GND, SCL și SDA. VIN va fi conectat la un pin ce are o tensiune între $2.5 - 5.5V$, GND va fi conectat la $0V$, iar pinii SCL și SDA trebuie conectați la magistrala I^2C ce operează pe nivelul logic al pinului VIN [24].

2.3 Conectarea AltIMU-10 v4 la Arduino Uno

Din caracteristicile de conexiune ai pachetului de senzori Altimu-10 v4 se observă că acesta trebuie conectat la o magistrală I^2C.

Magistrala I^2C a fost creata in ani '80 de către Philips semiconductors. Ea își propunea să fie o cale usoara de a conecta microcontroller-ul cu alte circuitele integrate.

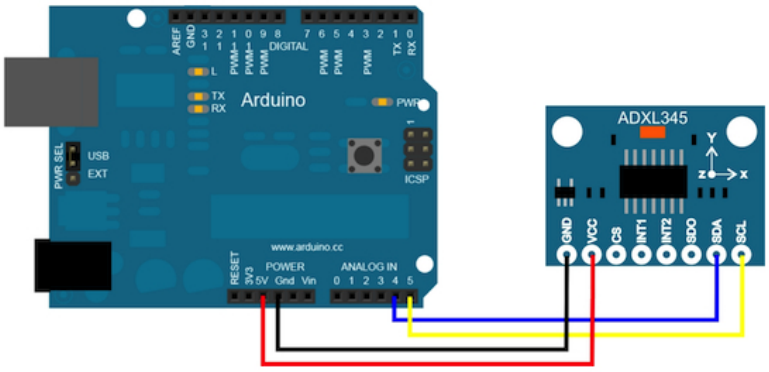

FIGURA 2.3: Conectarea AltIMU-10 v4 la Arduino Uno

Magistrala I^2C consta fizic în două linii active (SDA și SCL) si una de masă (GND) prin care comunică două sau mai multe componente dupa niște reguli precise. Liniile active SDA (Serial Data Line) și SCL (Serial CLock Line) sunt bidirectionale, componentele conectate la magistrala pot funcționa fie ca receptor fie ca emițător (la momente de timp distincte).

Controlul magistralei va aparține (la un moment dat) unui singur dispozitiv Master, care va adresa un singur dispozitiv Slave . Masterul este dispozitivul care va iniția transferul (printr-o condiție de START), va genera impulsurile de ceas pe linia SCL și tot el va încheia transferul generand conditia de STOP. Viteza de transfer pe magistrală poate fi de $100kbit/sec$ sau $400kbit/sec$, iar adresa dispozitivelor I^2C este formată de regulă din 7 biți [24].

3. Arhitectura software

Platforma de dezvoltare Unity 3D
Crearea unei animații în Unity3D
Limbajul de programare specific plăcii de dezvoltare Arduino
Platforma Matlab

3.1 Platforma de dezvoltare Unity 3D

La începutul anilor 2000, trei tineri programatori, fără mulți bani, au început să codeze și să implementeze cea ce avea să devină una din cele mai populare platforme software din industria jocurilor. Acești trei programatori sunt David Helgson, Joachim Ante și Nicholas Francis, iar proiectul lor a fost inspirat din platforma Apple Final Cut Pro. Final Cut Pro oferea realizatorilor amatori de filme instrumente profesionale la un preț redus, pe aceleași coordonate s-a bazat și Unity, vizând dezvoltatorii amatori de jocuri video.

O versiune primitivă de Unity a fost lansată în 2005, versiune ce era compatibilă și pe sistemul Windows, nu doar sistemul de operare Mac pentru care s-a dezvoltat inițial. Din 2008 bazându-se pe un real succes, platforma a devenit mult mai complexă, iar vânzările softului le-a permis să angajeze și o duzină de programatori.

O altă lovitură dată de în 2009 a fost folosirea platformei Unity3D de către Cartoon Network pentru crearea FusionFall, un MMORPG (Massively Multiplayer Online Role Playing Game) pentru copii ce a fost jucat de 8 milioane de persoane. Electronic Arts a folosit în Unity3D în 2009 pentru crearea Tiger Woods PGA Tour Online, chiar Microsoft și Ubisoft au devenit clienți pentru Unity.

Astăzi Unity are peste 300 de angajați în toată lumea și dezvoltă software pentru iOS, Android, Windows, Mac, Linux, Web browsers, PS3, Xbox 360. Unity plănuiește suport și pentru Sony's PlayStation Vita, oferind în acest moment suport pentru Windwos Phone și BlackBerry. Peste 1.8 milioane de programatori folosesc Unity, plug-inul pentru browser a fost insatlat de peste 200 de milioane de ori. Dead Trigger și Dead

Trigger 2, unele din cele mai complexe grafici pentru jocuri au fost dezvoltate pe baza a Unity3D.
Chiar dacă mari nume din industria graficii folosesc Unity, micii dezvoltatori sunt principala mândrie ai dezvoltatorilor acestei platforme, fapt ce se observă din deviza spusă de David Helgson, CEO și co-fondatorul Unity: *"Dorința noastră este să oferim aceleași instrumente și micilor developeri precum granzilor"* [25]

FIGURA 3.4: Animație Unity

3.1.1 Editorul Unity

Editorul Unity are împărțită fereastra principală mai multe secțiuni vizibile imaginea 3.5, ce descrie secțiunile importante.

Editorul Unity oferă un mediu *drag and drop*. Pentru dezvoltarea unui joc nu e nevoie obligatoriu de scrierea uni cod într-un limbaj de programare, dar majoritatea proiectelor necesită deprinderi de programare. Unity oferă o diversitate de limbaje de codare precum C# , JavaScript, sau Boo, pentru dezvoltatorii ce folosesc sintaxa Python. Mediul de dezvoltare este rulat în Mono, o versiune gratis de .NET Framework. Platforma Unity în sine este scrisă în C++,pentru că, din spusele lui Helgson "Animația trebuie să ruleze rapid precum un program C++, dar controlul animației trebuie făcut printr-un limbaj accesibil, oferit de platforma .NET" [25].

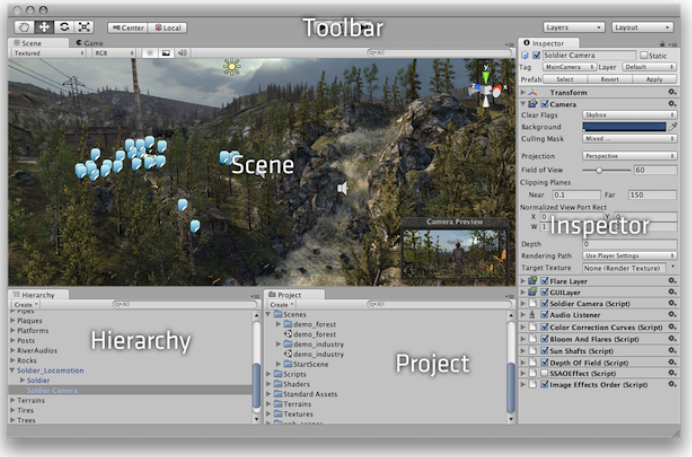

FIGURA 3.5: Editorul Unity

3.1.2 Crearea unei animații în Unity3D

Un joc în Unity este împărțit în mai multe obiecte ale jocului, obiecte ce sunt principalele entități ale animației și au proprietăți speciale, acesta putând fi un personaj, mediu înconjurător sau un efect special. Astfel anumite obiecte dintr-un joc sunt foarte diferite,fiind necesară o repartizare a acestora în containere ce înmagazinează obiecte de același tip.

Folosirea obiectelor

Un obiect dintr-un joc are automat o poziție, rotație și scalare, componente definite de unelta *Transform*, ce determină locația obiectului în spațiul real (3D).

Prin unelta *Rigidbody*, îi putem adăuga obiectului creat anumite caracteristici ce influențează comportaea inerțială din lumea reală, parametrii precum masa sau gravitația, aceștia influnțând comportarea animației spre o comportare realistă.

3.1.3 Controlul unei animații în Unity

Chiar și cele mai ușoare jocuri au nevoie de o secțiune de cod într-un limbaj de programare ce va răspunde la intrările venite de la utilizator și organizează evenimentele jocului într-o succesiune dorită de dezvoltator. Secțiunea de cod poate fi folosită pentru a crea efecte grafice sau chiar implementarea unui algoritm de inteligență artificială pentru personajele jocului.

Codul scris într-un limbaj compatibil .Net creează conexiunea dintre obiectele și acțiunile jocului, implementând o clasă principală de tip *MonoBehaviour*, ce va fi atșată unui obiect din joc.

În cadrul clasei $MonoBehavouir$ există două funcții principale. Funcția $Update$, în care se programează acțiunea obiectului în fiecare frame. În cadrul acestei funcții

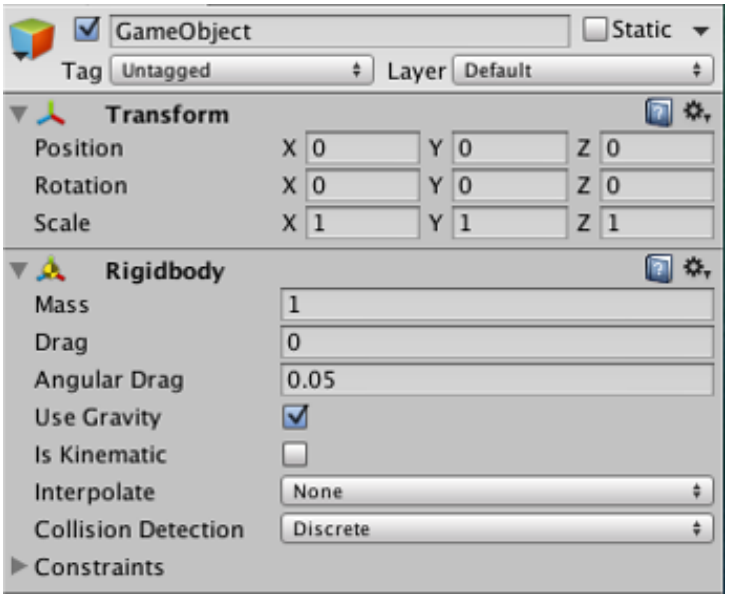

FIGURA 3.6: Uneltele *Transform* și *Rigidbody*

se programează acțiunile de răspuns la intrările introduse de utilizator. Practic, această funcție programează atitudinea obiectului, căreia este asociată, pe tot parcursul jocului. A doua funcție importantă în orice joc, este funcția $Start$ ce este apelată înainte ca jocul să înceapă, realizând toate inițializările stărilor obiectului, parametrii inițiali cu care va rula jocul în cadrul primei rulării a funcției Update, practic a fiecărui frame al jocului [25].

```
using UnityEngine;
using System.Collections;

public class MainPlayer : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```

Atașarea comportării setate de codul scris este asignată foarte ușor în cadrul editorului Unity, prin tragerea link-ului scriptului scrs asupra link-ului obiectului din panoul $Hierarchy$. Astfel în cadrul panoului $Inspector$ se poate observa că obectului i-a fost atașat un script, precum în imaginea următoare:

FIGURA 3.7: Confirmarea atașării scriptului

3.2 Limbajul de programare specific plăcii de dezvoltare Arduino

Pentru a programa microcontroller-ul trebuie să conectată placa Arduino la un computer pe care trebuie instalat mediul de dezvoltare şi driverele necesare. Mediul de dezvoltare este disponibil în mod gratuit pe site-ul producătorului pentru diverse sisteme de operare [23].

Pentru a configura mediul de dezvoltare arduino pentru sistemul de operare Windows se urmează pașii:

- se descarcă aplicaţia de pe pagina producătorului şi se dezarhivează într-un director convenabil;
- se conectează placa Arduino la computer printr-un cablu USB asstfel cel puţin un LED ar trebui să se aprindă pe placă;
- se indică locaţia driverului, în general directorul C:/arduino-1.0/drivers;
- se pornește mediul de dezvoltare executând C:/arduino-1.0/arduino.exe;
- se indică modelul plăcii în meniul Tools > Board;
- se indică portul pe care s-a conectat placa Arduino în meniul Tools > Serial Port;
- se elaborează scriptul ce se dorește a fi încărcat pe microcontroller;
- se încarcă programul pe placă: File > Upload.

3.2.1 Structura limbajului arduino

Limbajul folosit este o variantă simplificată de C/C++, ameliorată cu diverse biblioteci specifice platformei Arduino. Este foarte uşor de folosit pentru oricine are experienţă de programare în orice limbaj cât de cât structurat. De altfel, puteţi constata cât de simplu este analizând exemplul următor, care aprinde și stinge la anumite inervale un led:

```
 1. /*
 2.   Blink
 3.   Turns on an LED on for one second, then off for one second, repeatedly.
 5.   This example code is in the public domain.
 6.  */
 7.
 8. void setup() {
 9.   // initialize the digital pin as an output.
10.   // Pin 13 has an LED connected on most Arduino boards:
```

```
11.   pinMode(13, OUTPUT);
12. }
13.
14. void loop() {
15.   digitalWrite(13, HIGH);   // set the LED on
16.   delay(1000);              // wait for a second
17.   digitalWrite(13, LOW);    // set the LED off
18.   delay(1000);              // wait for a second
19. }
```

Au fost definite două funcţii, $setup()$ (liniile 8-12) şi $loop()$ (liniile 14-19). Aceste două funcţii trebuie să fie prezente în orice program.
Funcţia $setup()$ este executată o singură dată, la iniţializarea plăcii (de fiecare dată când este alimentată, de fiecare dată când încărcaţi un program nou şi de fiecare dată când resetaţi placa).În programul nostru, funcţia setup() face un singur lucru: declară pinul 13 (adică pinul digital 13) ca pin de ieşire. Dacă vă amintiţi de mai sus, pinul 13 este conectat şi la LED-ul de pe placă.

Funcţia $loop()$ se execută apoi la infinit, fără pauză. Dacă se doresc pauze sau încetiniri ale programului se pot scrie în acestă secțiune. În exemplul de mai sus funcţia $loop()$ realizează următoarele:

- linia 15: scrie "1" la pinul 13 (adică din acest moment pinul respectiv va fi alimentat cu 5V);
- linia 16: aşteaptă 1000 de milisecunde, adică o secundă; nu uitaţi, pinul 13 este alimentat, deci LED-ul de pe placă este aprins;
- linia 17: scrie "0" la pinul 13 (adică din acest moment pinul respectiv nu va mai fi alimentat);
- linia 18: aşteaptă din nou o secundă (dar de data asta pinul 13 nu mai este alimentat, deci LED-ul este stins).

3.2.2 Vizulaizarea datelor în ediorul arduino

Arduino permite vizualizarea datelor procesate de mocrocontroller prin intermediul unui modul serial, care se sincronizează prin intermediul codului, unde se setează o viteză de transfer a datelor prin comunicație serială. Astfel utiliztorul poate avea acces la datele procesate de arduino pe calculatorul propriu, modalitate foarte eficientă de evaluare a comportării procesului.

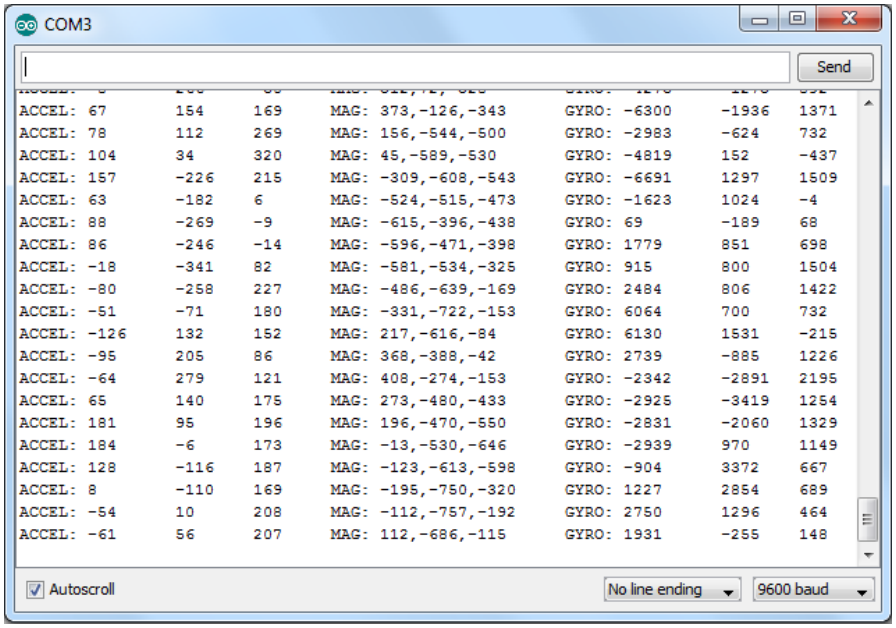

FIGURA 3.8: Datele procesate de arduino și afișate prin modulul serial

3.3 Platforma Matlab

MATLAB® (MATtrix LABoratory) este un pachet de programe de înaltă performanţă, interactiv, destinat calculului matematic, ştiinţific şi ingineresc. MATLAB integrează calcul, programare şi vizualizare, într-un mediu de lucru prietenos, soluţionarea problemelor presupunând folosirea notaţiilor matematice clasice. Utilizarea programului MATLAB include:

- Matematică şi calcul numeric
- Programare şi dezvoltare de algoritmi
- Modelare şi simulare
- Analiză de date, exploatarea rezultatelor şi vizualizare
- Grafică ştiinţifică şi inginerească
- Dezvoltare de aplicaţii software, incluzând construcţie de interfeţe grafice cu utilizatorul (GUI)

MATLAB este un produs al companiei americane The Mathworks, Inc. [http://www.mathworks.c şi lucrează sub Windows, Unix, LINUX şi Machintosh. MATLAB include toate facilităţile unui limbaj complet de programare, admiţând interfeţe cu limbajul de programare C, C++ şi FORTRAN.

MATLAB a cunoscut o puternică evoluţie în decursul ultimilor ani, reprezentând astăzi în mediile universitare o unealtă standard de calcul, fiind asociată diverselor cursuri introductive sau avansate în matematică, ştiinţă şi inginerie. În industrie, MATLAB este recunoscut ca un mijloc de investigaţie numerică performant, utilizat în sprijinul unei activităţi de cercetare, dezvoltare şi analiză de înalt nivel.

Versiunea completă a pachetului de programe MATLAB conţine o întreagă familie de module specifice, denumite tool-box-uri, respectiv blockset-uri, care permit rezolvarea unor aplicaţii din diverse domenii cum ar fi: maşini, aparate şi acţionări electrice, control de sistem, aplicaţii DSP, procesarea materialelor şi electro-tehnologii, procesare de semnal, mecanică, industria aeronautică şi de automobile, statistică, finanţe şi multe altele.

Aceste module sunt colecţii de funcţii MATLAB (M-files), uşor de asimilat, care extind puterea de calcul a pachetului de programe MATLAB în vederea rezolvării unor clase particulare de probleme. Colecţia de module MATLAB conţine: Simulink, DSP, Control System, SimPowerSystems, SimMechanics, Data Acquisition, Fuzzy Logic, Image Processing, Partial Differential Equations, Neural Network, Optimization, System Identification, Financial, Statistics, Communications, Database, Virtual Reality [26].

3.3.1 Structura sistemului MATLAB

Mediul de dezvoltare

Acesta este alcătuit dintr-un set de unelte care facilitează folosirea funcţiilor şi fişierelor MATLAB. Multe dintre acestea reprezintă de fapt interfeţele grafice şi includ fereastra principală MATLAB sau MATLAB Desktop, fereastra de comenzi sau Command Window, fereastra ce memorează istoria comenzilor sau Command History, şi browser-ele de Help, Workspace, Files, Search Path.

Biblioteca de funcţii matematice MATLAB

Aceasta constă într-o vastă colecţie de algoritmi de calcul, pornind de la funcţii elementare precum sumă, sinus, cosinus şi aritmetică complexă, pană la funcţii mai sofisticate precum inversare de matrici, calcul de valori proprii, funcţii Bessel, şi transformata Fourier.

Limbajul MATLAB

Limbajul MATLAB este un limbaj matrice/vector de înalt nivel ce include instrucţiuni de control al buclelor, funcţii, structuri de date, comenzi de intrare/ieşire şi instrucţiuni de programare orientată pe obiecte. Limbajul MATLAB permite atât ”programarea superficială” pentru crearea rapidă a unor mici programe de calcul specifice, cât şi "programarea în detaliu" în vederea dezvoltării unor programe complexe de nivel superior.

Handle Graphics®.

Handle Graphics reprezintă sistemul de grafică MATLAB şi include atât comenzi de înalt nivel pentru vizualizarea 2D şi 3D a datelor, procesare de imagini, animaţie şi

grafică, cât şi comenzi de jos nivel ce permit personalizarea completă a reprezentărilor grafice şi construirea integrală a interfeţelor grafice (GUI) pentru aplicaţiile MATLAB.
MATLAB Application Program Interface (API).

Aceasta este o bibliotecă ce permite scrierea programelor C şi Fortran ce interacţionează cu MATLAB. Biblioteca conţine facilitaţi de apel de subrutine din MATLAB (dynamic linking), de apelare a MATLAB-ul ca pe o maşină de calcul, şi de citire şi scriere de fişiere MAT-files [26].

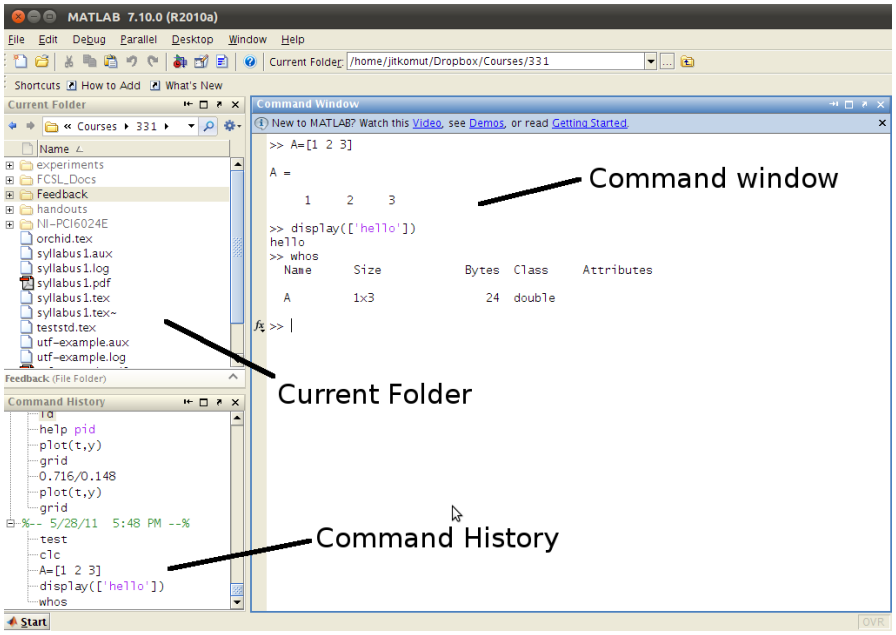

FIGURA 3.9: Editorul Matlab

3.4 Conexiunea dintre Arduino și calculator

Placa de dezvoltare Arduino prezintă facilități gândite pentru utilizatorii care se simt mult mai confortabil programând într-un limbaj de nivel înalt (Java, C#, Phyton, Ruby, PHP) . Aceaste facilități sunt realizate de comunicația serială , dintre plăcile Arduino și calculatoarele clasice, prin intermediul căreia datele captate de la senzori și/sau datele procesate de microcontroller-ul plăcii arduino vor fi transferate calculatorului unde, cu ajutorul unor limbaje de programare de nivel înalt, se pot realiza aplicații mult mai complexe, atât la nivel hardware cât și software. Tot aceași facilitate poate fi adusă prin realizarea unei comunicații de tip ethernet, dar această facilitate ridică costurile structurii hardware, și este dependentă de conexiunea la internet.

Datele sunt transmise în flux continuu, într-un format de tip string separat printr-un caracter, în general virgulă.La nivelul limbajului de programare folosit pe calculator

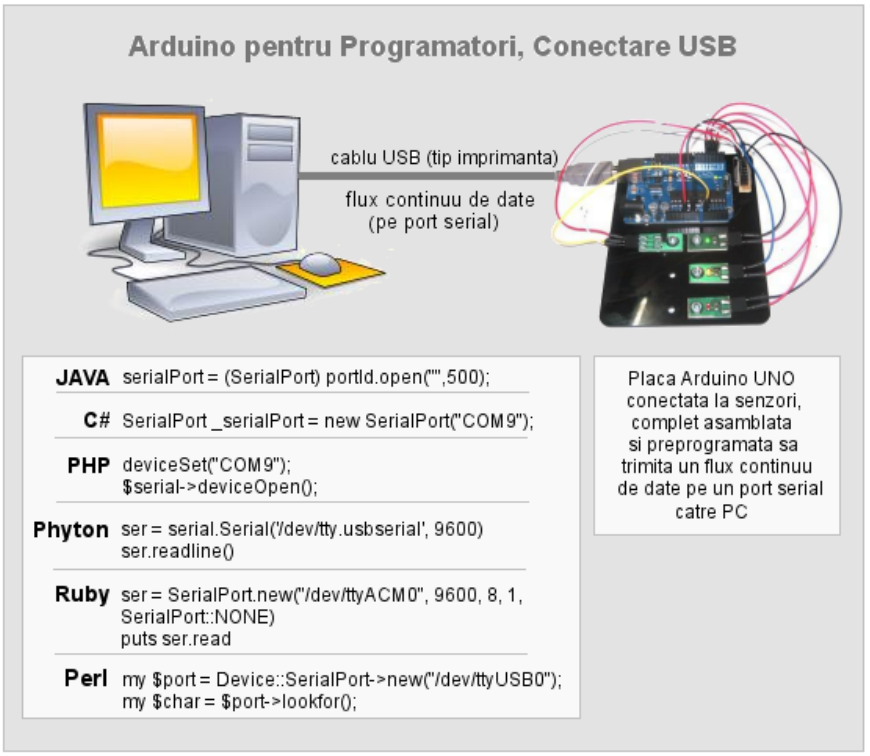

FIGURA 3.10: Conexiunea dintre placa Arduino și calculator

trebuie făcută citirea datelor și interpretarea acestora în funcție de semnificația acestora [23].

3.4.1 Conexiunea dintre placa de dezvolatre Arduino și Unity

Comunicarea dintre Arduino și Unity este realizată cu ajutorul librăriei CmdMessenger, ce face legătura între Arduino și limbajele .Net, implicit și editorul specific platformei Unity, Mono, editor ce e compatibil cu implementările C# [25].
Această librărie facilitează:

1. Trimiterea și primirea comenzilor în ambele sensuri.
2. Adăugarea de argumente comenzilor.
3. Generarea de funcții de reacție ca răspuns la anumite comenzi primite.
4. Transferul datelor bidirectional.

Formatul codului de citire a datelor prin intermediul portului serial, în cadrul editorului Mono, arată astfel:

```
using System;
using System.IO.Ports;

class MainClass
```

```
{
        public static void Main(string[] args)
        {
                SerialPort sport = new SerialPort("/dev/ttyUSB0", 9600);

                if (sport != null)
                {
                        if (sport.IsOpen)
                                {
                                        sport.Close();
                                }
                }

                Console.WriteLine("Arduino Serial Test");

                sport.Open();

                while(true)
                {
                        Console.WriteLine(sport.ReadLine());
                }
        }
}
```

3.4.2 Conexiunea dintre placa de dezvolatre Arduino și Matlab

Pentru a facilita transferul dintre Arduino și platforma Matlab se folosește pachetul MATLAB Support Package.Acest pachet se bazează pe un program ce rulează pe placa Arduino, ce ascultă comenzi venite prin intermediul portului serial, execută comenzile și, la nevoie, returnează un răspuns.
În imaginea următoare se poate vedea o parte din comenzile folosite în cadrul comunicației seriale [26].

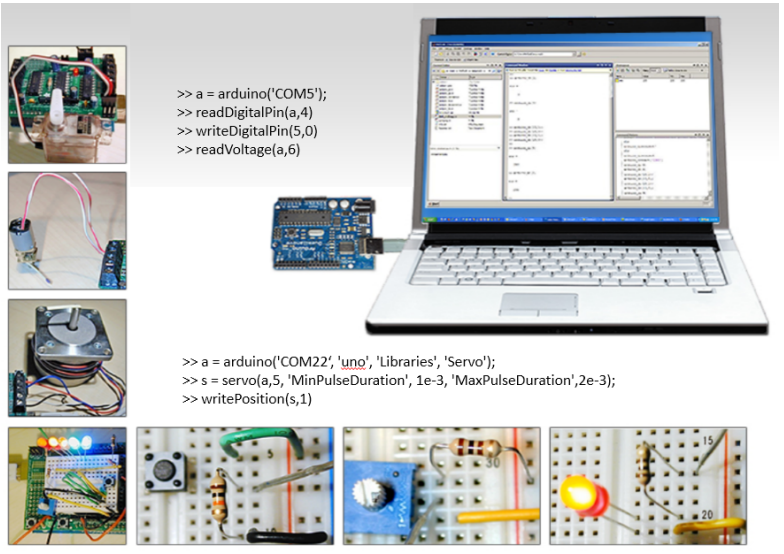

FIGURA 3.11: Conexiunea dintre placa Arduino și calculator

4. Convențiile orientării

Unghiurile lui Euler
Cuaternionii

Există multe metode ce descriu orientarea unui obiect în raport cu un sistem de referință. În acest proiect sunt folosite metoda unghiurilor lui Euler și cuaternionii relativi la orizontală și la câmpul magnetic. O altă metodă folosită des este determinarea direcției prin matricea cosinusului (DCM), dar această metodă este inferioară metodei cuaternionilor.

4.1 Unghiurile lui Euler

Reprezentarea sub forma unghiurilor lui Euler folosește rotația pe cele trei axe ale reprezentării 3D. Rotațiile realizate în funcție de cele trei axe se poate organiza în funcție de reprezentarea Tait-Bryan a unghiurilor sau în funcție de reprezentarea clasică a unghiurilor Euler. În acest proiect e folosită reprezentarea în funcție de $Z - X' - Z''$ a convenței Tait-Bryan, în care se reprezintă rotația în sensul acelor de ceasornic funcție de axa Z (yaw), rotația în funcție de axa Y (pitch), rotația în fucție de axa X (roll).

Această metodă de reprezentare a orientării are o serie de dezavantaje. Cel mai important dezavantaj este reprezentat de blocarea gimbal (Sistemul gimbal este reprezentat de trei inele pivotante ce exemplifică mișcările de rotație ale unui corp în sistemul 3D). Principiul blocării gimbal este reprezentat de cazul în care două inele axiale se plasează pe același plan, moment în care sistemul 3D se transforma într-un model 2D. În exemplul din imaginea 4.12 dacă avionul realizează o miscare de rotație cu 0 radiani față de axa Z, 0.5 radiani în raport cu axa Y și 0 radiani față de axa Z. Axa Z și axa X se afla pe planul xy, astfel avionul nu se mai poate rotii pe planul yz [18], [22].

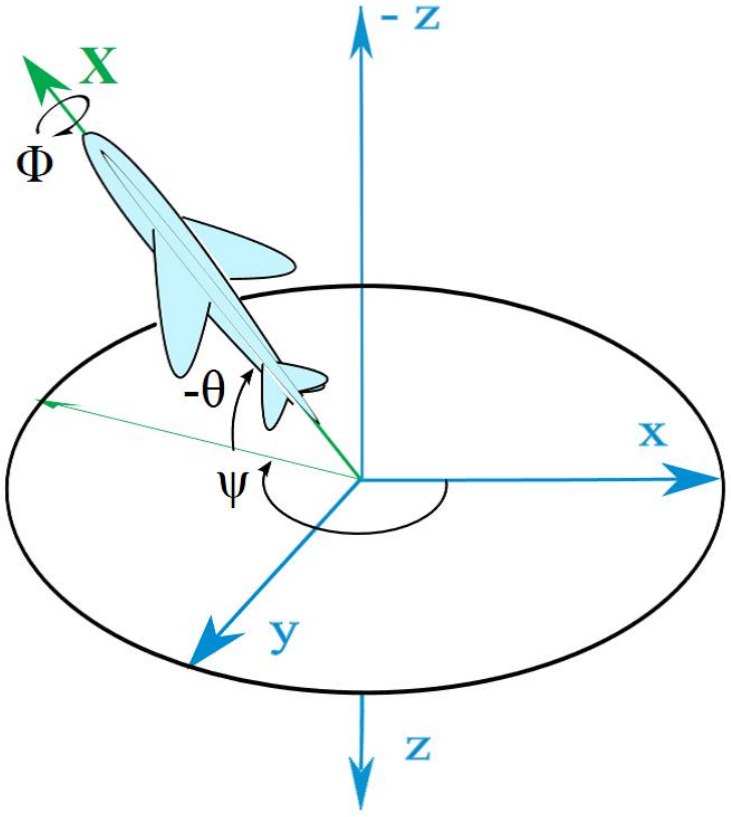

FIGURA 4.12: Reprezentarea unghiurilor Tait-Bryan. Unghiul Ψ corespunde rotației în jurul axei Z. Unghiul Φ corespunde rotației în jurul axei X. Unghiul Θ corespunde rotației în jurul axei Y

4.2 Cuaternionii

Un cuaternion reprezintă o extensie a numerelor complexe fiind o reprezentare patru-dimensională a unui număr complex folosit pentru coordona o reprezentare tri-dimensională sau pentru a reprezenta orientarea unui corp în spațiu. Rotația unui corp cu unghiul Θ în jurul axei A_r din sistemul de coordonate A, dar această rotație poate avea o reprezentare în raport cu un alt sistem de coordonate B, ce este relativ la sistemul A [8].

$$ {}^{A}_{B}\hat{\mathbf{q}} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \end{bmatrix} = \begin{bmatrix} cos(\frac{\Theta}{2}) & -r_x sin(\frac{\Theta}{2}) & r_y sin(\frac{\Theta}{2}) & r_z sin(\frac{\Theta}{2}) \end{bmatrix} \quad (4.4) $$

Astfel în figura 4.13 se poate observa că vectorii $\hat{x}_A$, $\hat{y}_A$, $\hat{z}_A$ și vectorii $\hat{x}_B$, $\hat{y}_B$, $\hat{z}_B$ reprezintă cele trei axe de coordonate ale sistemelor A respectiv B. În ecuația 1.1 cuaternionul $\hat{q}$ ce descrie orientarea corpului, unde r_x, r_y, r_z definesc componentele vectorului ${}^{A}\hat{r}$ in sistemul de coordonate A.

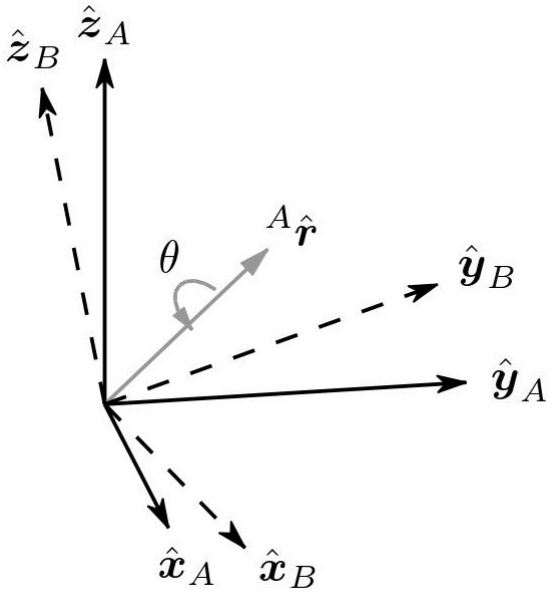

FIGURA 4.13: Orientarea vectorului ${}^A\hat{r}$ ce se rotește în cu unghiul Θ în sistemul de coordonate A raportat la sistemul de coordonate B

În momentul în care avem o unitate a magnitudinii, cuaternionii pot fi folosiți pentru a reprezenta rotația. Pentru reprezentarea rotațiilor se folosesc cuaternionii de rotație, iar pentru reprezentarea orientării relative la un sistem de referința se folosesc cuaternionii de poziție. Spre deosebire de metoda unghiurilor lui Euler, metoda cuaternionilor poate măsura rotațiile în sistemul 3D în fiecare orice situație, evitând blocarea gimbal. Această caracteristică a cuaternionilor duce la o reprezentare matematică a orintării mult mai robustă și fiabilă [12].

4.2.1 Norma unui cuaternion

Norma unui cuaternion este definită ca :

$$|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{4.5}$$

Norma cuaternionului este folosită la normalizarea expresiei acestuia dupa fiecare iterație. Expresia dupa cere se realizează acest lucru este [12]:

$$q = \frac{q}{|q|} \tag{4.6}$$

4.2.2 Înmulțirea

Înmulțirea quaternionilor $p \bigotimes q$ este definită ca o combinație secvențială a două rotații. Înmulțirea cuaternionilor este o operație necomutativă definită ca [12]:

$$\mathrm{p} \otimes \mathrm{q} = Q(p)q = \bar{Q}(q)p \tag{4.7}$$

unde matricea cuaternionilor este definită astfel:

$$\mathsf{Q}(\mathsf{p}) = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \tag{4.8}$$

iar conjugata acesteia este:

$$\bar{\mathsf{Q}}(\mathsf{p}) = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \tag{4.9}$$

4.2.3 Vectorul de rotație

Un vector x poate fi rotit față de sistemul de referință, proces ce e determinat de cuaternionul q ce produce expresia vectorului x' folosind expresia [12]:

$$\begin{bmatrix} 0 \\ x' \end{bmatrix} = q \otimes \begin{bmatrix} 0 \\ x \end{bmatrix} \otimes q^* \tag{4.10}$$

Aceste două înmulțiri de cuaterninoni poate fi combinată sub forma unei matrice $R(q)$ unde:

$$x' = R(q)x \tag{4.11}$$

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_2 - q_0q_3) & 2(q_1q_3 - q_0q_2) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{4.12}$$

4.2.4 Derivata unui cuaternion

Derivata unui cuaternion poate fi făcută cu ajutorul a două ecuații, prima ecuație ia în considerare viteza unghiulară a vectorului de referința în raport cu sistemul de

referință fixat, iar a doua ecuație analizează viteza unghiulară față de sistemul de referința al corpului. În acest proiect s-a folosit a doua ecuație [12]:

$$\dot{\mathsf{q}}_w(q,w) = \frac{1}{2}\begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ w_x \\ w_y \\ w_z \end{bmatrix} \tag{4.13}$$

4.2.5 Determinarea unghiurilor lui Euler pe baza cuaternionilor

Pe baza expresiei orintării pe baza cuaternionilor se poate determina expresia unghiurilor lui Euler în forma Tait-Bryan YPR (yaw=rotația pe axa z, pitch=rotația pe axa y, roll=rotația pe axa x) [1]:

$$yaw = atan2(2 \cdot (q_0 q_3 + q_1 q_2), 1 - 2 \cdot (q_2^2 + q_3^2)) \tag{4.14}$$

$$pitch = atan2(2 \cdot (q_0 q_1 + q_2 q_3), 1 - 2 \cdot (q_1^2 + q_2^2)) \tag{4.15}$$

$$roll = asin(2 \cdot (q_0 q_2 + q_3 q_1)) \tag{4.16}$$

Cu ajutorul acestor ecuații se poate determina și folosii o expresie a unghiurilor lui Euler neafectată de efectul blocării gimbal, generând performanțe mult mai bune în aplicațiile folosite.

5. Filtrarea datelor

Filtrul Kalman
Dezavantajele folosirii filtrului Kalman liniar
Filtrul Kalman extins

5.1 Filtrul Kalman

Filtrul Kalman e un filtru linear și recursiv ce e folosit pentru urmărirea unui proces cu funcție de probabilitate Gaussiană [12].

5.1.1 Predicția

Predicția inferă starea din cadrul i, din informațiile din cadrul anterior $i-1$ şi din modelul dinamic. Modelul dinamic liniar este aplicat estimării anterioare, şi se obține valoarea prezisă pentru vectorul de stare curent. Adițional, valori ale unor mărimi cunoscute pot forma un vector de intrare ui care poate contribui la predicție.

Pe lângă modelul dinamic, exprimat de transformarea liniară F_i , şi de modelul intrării pe care îl exprimăm ca transformarea liniară B_i , există o incertitudine w_i. Această incertitudine (zgomot) exprimă devierea unui sistem real față de modelul dinamic şi de intrare, care nu pot ține cont de orice evoluție. Acest zgomot are media zero, deci nu va influența predicția, care este exprimată de următoarea ecuație [8]:

$$\bar{X}_i = F_i X_{i-1} + B_i u_i \tag{5.17}$$

Pentru a obține matricea de covarianță pentru predicție, vom aplica transformările modelului dinamic matricei de covarianță a stării anterioare, modelul de transformare a intrării pe matricea de covarianță a intrării, si vom adăuga matricea de covarianță a incertitudinii. Notăm cu T_i matricea de covarianță a intrării, si cu Q_i matricea de

covarianță a incertitudinii tranziției. Atunci covarianța predicției este:

$$\bar{P}_i = F_i P_{i-1} X_i F_i^T + B_i T_i B_i^T + Q_i \tag{5.18}$$

5.1.2 Măsurarea

Proces extern filtrului Kalman, măsurãtoarea are ca rezultat unul sau mai mulți vectori de măsurã Y_i^k fiecare cu o matrice de covarianță R_i^k care codificã eroarea de măsurare estimatã (imprecizia senzorului). Relația dintre vectorul de stare si vectorul de măsurã este modelul de măsurare, iar dacã acesta este o transformare liniarã aceasta este descrisã de matricea H_i. Folosind acest model, obținem predicția măsurãtorii, $\bar{Y}_i$ [17].

$$\bar{Y}_i = H_i \bar{X}_i \tag{5.19}$$

Matricea de covarianță pe care o vom asocia lui $\bar{Y}_i$ va fi notată S_i.

$$S_i = H_i P_i H_i^T + R_i \tag{5.20}$$

Matricea S_i nu este matricea de covarianță a măsurãtorii prezise, ci matricea de covarianță a diferenței dintre măsurãtoarea prezisã si o posibilã măsurãtoare realã (matricea de covarianță a $inovarii$, sau a $rezidualului$) – adicã exact matricea necesarã pentru a definii o zonã de cãutare în jurul măsurãtorii prezise $\bar{Y}_i$.

5.1.3 Asocierea datelor

Filtrul Kalman este foarte vulnerabil la asocierea datelor (asocierea măsurãtorilor), din cauza naturii unimodale a funcției de probabilitate în fiecare din fazele de lucru ale filtrului. Acest lucru înseamnã cã vectorul de stare, odatã deplasat spre un indiciu fals, va deveni criteriul de selecție pentru măsurãtorile viitoare, asta însemnând mai multe date greșite incluse în estimare, pânã la devierea totalã de la țintã.
Singura măsură obiectivă a utilității măsurătorii Y_i este verosimilitatea ei, dându-se starea unui obiect urmărit. Această verosimilitate este dată de funcția de probabilitate Gaussiană, centrată în predicția măsurătorii $\bar{Y}_i$, si având matricea de covarianță S_i [17].

$$p(Y_i^k) = \frac{1}{\sqrt{(2\pi)^n |S_i|}} e^{-\frac{(Y_i^k - \bar{Y}_i)^T S_i^{-1} (Y_i^k - \bar{Y}_i)}{2}} \tag{5.21}$$

5.1.4 Corecție

În acest moment toate datele necesare sunt disponibile, și se va calcula noul vector de stare X_i si matricea sa de covarianță P_i.

Prima dată se calculează matricea de amplificare Kalman, K_i, [17]:

$$K_i = \bar{P}_i H_i^T (H_i \bar{P}_i H_i^T + R_i)^{-1} \tag{5.22}$$

Ecuația este echivalentă cu

$$K_i = \bar{P}_i H_i^T S_i^{-1} \tag{5.23}$$

Pasul final este calcularea matricei de covarianță P_i. Majoritatea documentațiilor disponibile dau ecuația următoare pentru calculul P_i.

$$P_i = (I - K_i H_I)\bar{P}_i \tag{5.24}$$

5.1.5 Vectorul de stare

Acest filtru este folosit pentru estimarea rotației în jurul axei $Y(pitch)$ și a rotației în jurul axei $X(roll)$, luând în considerare și erorile de măsurare ale giroscopului pe aceste axe, w_{xb} și w_{yb}:

$$x = \begin{bmatrix} pitch \\ roll \\ w_{xb} \\ w_{yb} \end{bmatrix}_i \tag{5.25}$$

5.1.6 Predicția modelului și măsurătorile modelului orientării

Expresiile *modelului predicție* și *modelului măsurătorilor* pot fi exprimate prin ecuațiile matriceale [1], [12]:

$$\begin{bmatrix} pitch \\ roll \\ w_{xb} \\ w_{yb} \end{bmatrix}_i = \begin{bmatrix} 1 & 0 & -dt & 0 \\ 0 & 1 & 0 & -dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_i \cdot \begin{bmatrix} pitch \\ roll \\ w_{xb} \\ w_{yb} \end{bmatrix}_{i-1} + \begin{bmatrix} dt & 0 & 0 & 0 \\ 0 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}_i \cdot \begin{bmatrix} w_x \\ w_y \\ 0 \\ 0 \end{bmatrix}_i \tag{5.26}$$

5.2 Dezavantajele folosirii filtrului Kalman liniar

Implementarea unui filtru Kalman linear are o serie de dezavantaje. Primul dezavantaj se întâlnește în momentul rotirii unei axe iar concomitent altă axă este deviată de la orizontală, unghiul trigonometric estimat de accelerometru și corespondenta viteză unghiulară măsurată de giroscop nu vor mai fi în același plan. Acest aspect nu reprezintă un defect în momentul în care se operează foarte aproape de orizontală, dar în momentul în care se operează o distanță mare față de orizontală rezultă o sursă importantă de erori.

Pentru a rezolva aceste dezavantaje se folosește implementarea cuaternionilor pe baza comportării modelului orientării. Acest lucru impune folosirea predicției neliniarității modelului măsurat. Acest lucru poate fi făcut prin implementarea unui filtru Kalman extins [12].

5.3 Filtrul Kalman extins

Filtrul Kalman extins este versiunea neliniară a filtrului Kalman regulat. Această variantă de filtru Kalman folosește matricea Jacobian pentru predicție, iar funcțiile măsurătorilor pentru a liniariza estimarea stării curente și a covarianței. Această implentare permite unui nou set de predicți și de măsurători, pentru a fi folosite în cadrul estimării.

5.3.1 Vectorul de stare

Vectorul de stare reprezintă reprezintă variabilele ce se doresc a fi estimate recursiv cu ajutorul filtrului. În acest proiect vectorul de stare e compus dintr-un cuaternion ce determiă orintarea și erorile vitezelor unghiulare măsurate de giroscop [17].

$$x = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 & q_3 & w_{xb} & w_{yb} & w_{zb} \end{bmatrix}^T \tag{5.27}$$

La momentul inițial vectorul de stare este compus din poziția inițială q =[1 0 0 0] și din eroarea viteze unghiulare care este necunoscută inițial.

$$x = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{5.28}$$

5.3.2 Formularea modelului predicției

În această aplicație filtrul Kalman extins folosește un model al predicție ce este reprezentat printr-o funcție ce descrie următoarea orientare estimată pe baza precedentei

estimări și a vectorului vitezei unghiulare măsurate de giroscop.

$$q_k = q_{k-1} + dt * \dot{q}_k \tag{5.29}$$

unde:

$$\dot{\mathsf{q}}_w(q, w) = \frac{1}{2} \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ q_2 & -q_1 & q_0 \end{bmatrix} \cdot \begin{bmatrix} w_x - w_{xb} \\ w_y - w_{yb} \\ w_z - w_{zb} \end{bmatrix} \tag{5.30}$$

Astfel următoarea predicție poate fi definită, pornind de la expresia predicției filtrului Kalman liniar $\bar{x_i} = F_i x_{i-1} + B_i u_i$, printr-o funcție $f(x_{i-1}, u_i)$ ce va arăta astfel:

$$x_i = f(x_{i-1}, u_i) = \begin{bmatrix} q_0 + \frac{dt}{2} \cdot (-q_1(w_x - w_{xb}) - q_2(w_y - w_{yb}) - q_3(w_z - w_{zb})) \\ q_1 + \frac{dt}{2} \cdot (q_0(w_x - w_{xb}) + q_3(w_y - w_{yb}) - q_2(w_z - w_{zb})) \\ q_2 + \frac{dt}{2} \cdot (-q_3(w_x - w_{xb}) + q_0(w_y - w_{yb}) + q_1(w_z - w_{zb})) \\ q_3 + \frac{dt}{2} \cdot (q_2(w_x - w_{xb}) - q_1(w_y - w_{yb}) + q_0(w_z - w_{zb})) \\ w_{xb} \\ w_{yb} \\ w_{zb} \end{bmatrix} \tag{5.31}$$

Expresia Jacobianului poate fi calculată prin derivarea funcției f în funcție de poziție, rezultând matricea F

$$\frac{\partial f}{\partial x} = F = \begin{bmatrix} 1 & -\frac{dt}{2}(w_x - w_{xb}) & -\frac{dt}{2}(w_y - w_{yb}) & -\frac{dt}{2}(w_z - w_{zb}) & \frac{dt}{2}q_1 & \frac{dt}{2}q_2 & \frac{dt}{2}q_3 \\ \frac{dt}{2}(w_x - w_{xb}) & 1 & -\frac{dt}{2}(w_z - w_{zb}) & \frac{dt}{2}(w_y - w_{yb}) & -\frac{dt}{2}q_0 & -\frac{dt}{2}q_3 & \frac{dt}{2}q_2 \\ \frac{dt}{2}(w_y - w_{yb}) & \frac{dt}{2}(w_z - w_{zb}) & 1 & -\frac{dt}{2}(w_x - w_{xb}) & \frac{dt}{2}q_3 & -\frac{dt}{2}q_0 & -\frac{dt}{2}q_1 \\ \frac{dt}{2}(w_z - w_{zb}) & -\frac{dt}{2}(w_y - w_{yb}) & \frac{dt}{2}(w_x - w_{xb} & 1 & -\frac{dt}{2}q_2 & \frac{dt}{2}q_1 & -\frac{dt}{2}q_0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.32}$$

5.3.3 Formularea modelului măsurătorilor

Modelul măsurătorilor definit în cadrul filtrului Kalman extins ca $y_i = H_i x_i$, putând fi exprimat ca o funcție de orientarea estimată, astfel $h(x_k)$. Această funcție poate fi folosită pentru a mapa starea estimată pe vectorul măsurătorilor z, practic se realizează o comparare a predicție cu măsurătorile reale. În acest caz măsurătorile realizate ce implică parametrii accelerometrului și ai magnetometrului ce vor fi folosiți pentru

predicția curentă. Vectorul măsurătorilor poate fi definit astfel [8]:

$$z = \begin{bmatrix} a_x & a_y & a_z & m_x & m_y & m_z \end{bmatrix}^T \tag{5.33}$$

5.3.3.1 Maparea parametrilor accelerometrului

Vectorul gravitației din sistemul fixat este rotit în cadrul sistemului de coordonate al corpului, fapt reprezentat prin cuaternionul q,mapat in cadrul vectorului accelerometrului. Această aplicație ține cont doar de direcția vectorului gravitației și nu de cea a câmpului magnetic, vectorul este normalizat înnainte de operarea asupra acestuia.

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = (h(x_i)) = R(q) * \overrightarrow{g} = R(q) \cdot \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} -2(q_1q_3 - q_0q_2) \\ -2(q_2q_3 - q_0q_1) \\ -q_0^2 + q_1^2 + q_2^2 - q_3^2 \end{bmatrix} \tag{5.34}$$

5.3.3.2 Maparea parametrilor magnetometrului

Maparea componentelor câmpului magnetic se face similar cu maparea accelerației, rotațile vectorului magnetometrului fiind reprezentate de cuaternionul determinat.

$$\begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} = (h(x_i)) = R(\overrightarrow{q}) \cdot \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} b_x(q_0^2 + q_1^2 - q_2^2 - q_3^2) + 2b_y(q_1q_2 + q_0q_3) + b_z(q_1q_3 - q_0q_2) \\ 2b_x(q_1q_2 + q_0q_3) + b_y(q_0^2 - q_1^2 + q_2^2 - q_3^2) + b_z(q_2q_3 - q_0q_1) \\ 2b_x(q_1q_3 + q_0q_2) + b_y(q_2q_3 - q_0q_1) + b_z(q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \tag{5.35}$$

Folosirea acestui model de citire a valorilor magnetometrului influențează estimările expresilor rotațiilor în jurul axelor $x(roll)$ și $y(pitch)$ determinând erori ale orientării. De asemenea este nevoie de specificarea direcției câmpului magnetic al pământului, care variază considerabil în jurul pământului. Acastă problemă a fost rezolvată folosind soluția prezentată de S.Madgwick [1], unde direcția câmpului magnetic este calculată pe baza aceiași înclinări asupra câmpului măsurat.

Acest procedeu este facut prin rotirea vectorului câmpului magnetic măsurat al sistemului de coordonate al corpului în raport cu sisteul de coordonate fix, asfel rezultând expresia m' [12]

$$\begin{bmatrix} m'_x \\ m'_y \\ m'_z \end{bmatrix} = R^*(\overrightarrow{q}) \cdot \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} = \begin{bmatrix} m_x(q_0^2 + q_1^2 - q_2^2 - q_3^2) + 2m_y(q_1q_2 + q_0q_3) + m_z(q_1q_3 - q_0q_2) \\ 2m_x(q_1q_2 + q_0q_3) + m_y(q_0^2 - q_1^2 + q_2^2 - q_3^2) + m_z(q_2q_3 - q_0q_1) \\ 2m_x(q_1q_3 + q_0q_2) + m_y(q_2q_3 - q_0q_1) + m_z(q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \tag{5.36}$$

Noul câmp de referință e calculat în scopul obținerii aceiași înclinări în cadrul vectorului măsurătorilor.

$$b = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} \sqrt{{m'_x}^2 + {m'_y}^2} \\ 0 \\ m'_z \end{bmatrix} \tag{5.37}$$

Această nouă valoare de referință este substituită în maparea originală, unde termenul $b_y = 0$

$$\begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} = (h(x_i)) = R(\overrightarrow{q}) \cdot \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} b_x(q_0^2 + q_1^2 - q_2^2 - q_3^2) + b_z(q_1q_3 - q_0q_2) \\ 2b_x(q_1q_2 + q_0q_3) + b_z(q_2q_3 - q_0q_1) \\ 2b_x(q_1q_3 + q_0q_2) + b_z(q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \tag{5.38}$$

Aceste două mapări pot fi combinate într-o funcție extinsă a măsurătorilor h și a expresiei Jacobianului H

$$h(x_i) = \begin{bmatrix} -2(q_1q_2 - q_0q_2) \\ -2(q_2q_3 - q_0q_1) \\ -q_0^2 + q_1^2 + q_2^2 - q_3^2 \\ b_x(q_0^2 + q_1^2 - q_2^2 - q_3^2) + b_z(q_1q_3 - q_0q_2) \\ 2b_x(q_1q_2 + q_0q_3) + b_z(q_2q_3 - q_0q_1) \\ 2b_x(q_1q_3 + q_0q_2) + b_z(q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \tag{5.39}$$

$$H = \frac{\partial h}{\partial x} = \begin{bmatrix} 2q_2 & -2q_3 & 2q_0 & -2q_1 & 0 & 0 & 0 \\ -2q_1 & -2q_0 & -2q_3 & -2q_2 & 0 & 0 & 0 \\ -2q_0 & 2q_1 & -2q_2 & -2q_3 & 0 & 0 & 0 \\ -2q_0 & 2q_1 & -2q_2 & -2q_3 & 0 & 0 & 0 \\ 2(q_0b_x - q_2b_z) & 2(q_1b_x + q_3b_z) & 2(-q_2b_x - q_0b_z) & 2(-q_3b_x + q_1b_z) & 0 & 0 & 0 \\ 2(-q_3b_x + q_1b_z) & 2(q_2b_x + q_0b_z) & 2(q_1b_x + q_3b_z) & 2(-q_0b_x + q_2b_z) & 0 & 0 & 0 \\ 2(q_2b_x + q_0b_z) & 2(q_3b_x - q_1b_z) & 2(q_0b_x - q_2b_z) & 2(q_1b_x + q_3b_z) & 0 & 0 & 0 \end{bmatrix} \tag{5.40}$$

5.3.4 Matricele de covarianță

Rezultatele filtrului Kalman estins sunt afectate și de specificațiile matrielor de covarianță Q și R. Acestea sunt matrice diagonală reprezentând variația așteptată a zgomotului în faza predicției și în vectorul măsurătorilor$z$. În această aplicație matricele

de covarianță au fost alese astfel:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.2 \end{bmatrix} \tag{5.41}$$

$$R = \begin{bmatrix} 1 \cdot e6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 \cdot e6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 \cdot e6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \cdot e6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \cdot e6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \cdot e6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \cdot e6 \end{bmatrix} \tag{5.42}$$

5.3.5 Ecuațiile filtrului Kalman extins

Ecuațiile filtrului Kalman extins se pot clasifica în două secțiuni specifice etapelor filtrării.

Ecuațiile predicției

Predicția orinetări: $x_i = f(x_{i-1}, u_i)$

Covarianța predicției: $P = FPF' + Q$

Ecuațiile corecției

Predicția măsurătorii: $y = z - h(x_i)$

Covarianța măsurătorii: $S = HPH' + R$

Matricea de amplificare Kalman: $K = PH'S^{-1}$

Corecția orinetării prezise:$x_i = x_i + Ky$

Corecția covarianței prezise:$P = (I - KH)P$

Cu $F = \frac{\partial f}{\partial x}$ și $H = \frac{\partial h}{\partial x}$.$I$ reprezintă matricea unitate.

Expresia cuaternionului rezultat poate fi convertită în expresia sub forma unghiurilor lui Euler [12].

$$\begin{bmatrix} pitch \\ roll \\ yaw \end{bmatrix} (q) = \begin{bmatrix} atan2(2(q_0q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2)) \\ asin(2 \cdot (q_0q_2 + q_3q_1)) \\ atan2(2 \cdot (q_0q_3 + q_1q_2), 1 - 2 \cdot (q_2^2 + q_3^2)) \end{bmatrix} \tag{5.43}$$

6. Determinarea orientării pe baza filtrului AHRS

Filtrul AHRS
Orientarea pe baza vitezei unghiulare
Orientarea pe baza vectorului măsurătorilor
Fuziunea celor două metode de determianre a orientării
Compensarea deformării magnetice
Controlul implicării erorii furnizate de giroscop

6.1 Filtrul AHRS

Majoritatea senzorilor inerțiali generează date brute ce nu pot fi implementate astfel în aplicații. Obectivul acestui proiect este de a transforma aceste date brute în reprezentări ce pot determia orientarea în spațiul 3D. Pentru a realiza acest lucru, datele provenite de la senzorii inerțiali trebuie procesate, calibrate și stocate pentru a genera date ce pot configura orientarea în spațiul 3D.
Elaborarea unui algoritm de generare a orientării a reprezentat o problemă intens dezbătută realizându-se multe variante de generare a orientării pe baza accelerometrului, giroscopului și magnetometrului. Astfel metodele cele mai folosite în cadrul proiectelor bazate pe determinarea orientării sau bazat pe metodele unghiurilor lui Euler, matricilor de rotație, cuaternionilor sau filtrului Kalman. Aceste prezintă o complextate de operații matematice ce diminuează viteza de execuție, dar au și a dificultate în implementarea software, având și performanțe reduse prin implementarea unei singure metode enunțate mai sus.

O implementare care prezintă o eficiență foarte bună realizând o implementare ce face o simbioză între metodele enunțate anterior este reprezentată de filtrul AHRS

(Altittude and Heading Reference System) dezvoltat de Mahony și perfecționat de Madgwick. Un filtru ce este mult mai rapid și mult mai simplu decât filtrul Kalman, prezentând performanțe asemănătoare poate chiar mai bune.

6.2 Principiile folosite în filtrul AHRS

Implemetarea realizată de Madgwick prezintă două versiuni de filtru AHRS. Prima implementare se adresa dispozitivelor IMU generale ce e compus dintr-un accelerometru și giroscop tri-axial. A doua implementare se referă asupra senzorilor MARG (Magnetic, Angular Rate, and Gravity), senzori ce aduc o îmbunătățire față de senzorii IMU, având în componență un magnetometru tri-axial. Această implementare încorporează componenta magnetică și compensarea erorii generate de măsurătorile giroscopului[1].

6.2.1 Orientarea pe baza vitezei unghiulare

Giroscopul tri-axial va măsura viteza unghiulară a axelor x, y și z ale sistemului de referință al senzorului, vitezele ungiulare se notează w_x, w_y și w_z. Acești parametrii sunt distribuiți într-un vector definit prin ecuația(6.44), expresia cuaternionului descrie viteza de schimbare a orientării a sistemului de coordonate al senzorului raportat la sistemul de coordonate al pământului ${}^{S}_{E}\dot{q}$ ce poate fi calculată prin ecuația (6.45)[3].

$$S = \begin{bmatrix} 0 & w_x & w_y & w_z \end{bmatrix} \tag{6.44}$$

$${}^{S}_{E}\dot{q} = \frac{1}{2} {}^{S}_{E}\hat{q} \bigotimes S_w \tag{6.45}$$

Orientarea sistemului de coordonate al senzorului relativ la sistemul de coordonate al pământului la momentul de timp t, ${}^{S}_{E}q_{w,t}$ poate fi determinat prin integrarea expresiei derivatei cuaternionului determinat anterior ${}^{S}_{E}\dot{q}_{w,t}$ este descris prin ecuațile (6.46) și (6.47) furnizate pentru condițiile inițiale știute. În această ecuație, S_{wt} este viteza unghiulară măsurată la timpul t, Δt este perioada de eșantionare, iar ${}^{S}_{E}\hat{q}_{est,t-1}$ este estimarea precedentă a eorientării. Indicele w indică calcularea cuaternionului în funcție de viteza unghiulară[1].

$${}^{S}_{E}\dot{q}w, t = \frac{1}{2} {}^{S}_{E}\hat{q}_{est,t-1} \bigotimes S_{w,t} \tag{6.46}$$

$${}^{S}_{E}q_{w,t} = {}^{S}_{E}\hat{q}_{est,t-1} + {}^{S}_{E}\dot{q}_{w,t}\Delta t \tag{6.47}$$

6.2.2 Orientarea pe baza vectorului măsurătorilor

Accelerometrul tri-axial va măsura mărimea și direcția câmpului gravitației în sistemul de coordonate al senzorului, măsurători afectate de accelerațiile liniare determinate de mișcarea senzorului. Similar magnetometrul tri-axial va măsura mărimea și direcția câmpului magnetic al pământului în sistemul de coordonate al senzorului, afectt de câmpul magnetic local și de perturbații. În contextul implementării unui filtru al orientării, se va admite ca accelerometrul va măsura doar gravitația, iar magnetometrul va măsura doar câmpul magnetic al pământului[1].

Dacă direcția câmpului pamântului este cunoscută în sistemul de coordonate terestru, măsurarea direcție cămpului în cadrul sisemului de coordonate al senzorului va permite calcularea orientării sistemului de coordonate al senzorului în raport cu sistemul de coordonate terestru. Oricum, pentru orice măsurători date, nu va exista o soluție unică a orientării senzorului, existând o infinitate de soluții reperezentate de rotațiile ce se pot realiza pe baza orientării în jurul unei axe paralele cu câmpul. În unele aplicații se folosesc pentru determinarea orientării unghiurile lui Euler, dar acestea generează o soluție incompletă, generând o soluție în care se cunosc două unghiuri, dar al treilea este necunoscut, unghiul necunoscut fiind ungiul în jurul axei paralele cu cămpul magnetic al pământului. Această problemă a fost dezbătută în capitolul 4.2.5, și se numește blocarea gimbal [13]. O soluție pentru rezolvara completă a acestei probleme este dată de folosirea quaternionilor, detaliați în capitolul 4.2.5. Generând formularea optimizării problemei, se consideră o orientare a senzorului, ${}^{S}_{E}\hat{q}$, care se aliniază cu o direcție predefinită a a câmpului în sistemul de coordonate al pământului ${}^{E}\hat{d}$, cu direcția măsurată a câmpului în sistemul senzorului, ${}^{S}\hat{s}$, folosind operația de rotație. Astfel ${}^{S}_{E}\hat{q}$ e posibil să găsească o soluție, unde ecuația (6.48) este funcția obiectiv, iar componenta fiecărui vector e definită în ecuațiile (6.49),(6.50) și (6.51)[1].

$$f({}^{S}_{E}\hat{q}, {}^{E}\hat{d}, {}^{S}\hat{s}) = {}^{S}_{E}\hat{q}^{*} \bigotimes {}^{E}\hat{d} \bigotimes {}^{S}_{E}\hat{q} - {}^{S}\hat{s} \tag{6.48}$$

$${}^{S}_{E}\hat{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix} \tag{6.49}$$

$${}^{E}\hat{d} = \begin{bmatrix} 0 & d_x & d_y & d_z \end{bmatrix} \tag{6.50}$$

$${}^{S}\hat{s} = \begin{bmatrix} 0 & s_x & s_y & s_z \end{bmatrix} \tag{6.51}$$

Există mulți algoritmi de optimizare, dar algoritmul gradientului este unul dintre

cele mai simple de implementat și calculat. Ecuația (6.52) descrie algoritmul gradientului pentru n iterații rezultând orientarea estimată ${}^{S}_{E}q_{n+1}$ bazată pe orientarea inițială ${}^{S}_{E}q_0$ și un pas μ. Ecuația (6.53) calculează soluția definită prin funcția obiect și funcția Jacobian, simplificate sub forma unor matrice coloană (6.54) respctiv o matrice 3x4 (6.55)

$$ {}^{S}_{E}q_{k+1} = {}^{S}_{E}\hat{q}_k - \mu\frac{\bigtriangledown f({}^{S}_{E}\hat{q}, {}^{E}\hat{d}, {}^{S}\hat{s})}{||\bigtriangledown f({}^{S}_{E}\hat{q}, {}^{E}\hat{d}, {}^{S}\hat{s})||}, k = 0, 1, 2...n. \tag{6.52} $$

$$ \bigtriangledown f({}^{S}_{E}\hat{q}, {}^{E}\hat{d}, {}^{S}\hat{s}) = J^T({}^{S}_{E}\hat{q}, {}^{E}\hat{d}) f({}^{S}_{E}\hat{q}, {}^{E}\hat{d}, {}^{S}\hat{s}) \tag{6.53} $$

$$ f({}^{S}_{E}\hat{q}, {}^{E}\hat{d}, {}^{S}\hat{s}) = \begin{bmatrix} 2d_x(\frac{1}{2} - q_3^2 - q_4^2) + 2d_y(q_1q_4 + q_2q_3) + 2d_z(q_2q_4 - q_1q_3) - s_x \\ 2d_x(q_2q_3 - q_1q_4) + 2d_y(\frac{1}{2} - q_2^2 - q_4^2) + 2d_z(q_1q_2 + q_3q_4) - s_y \\ 2d_x(q_1q_3 + q_2q_4) + 2d_y(q_3q_4 - q_1q_2) + 2d_z(\frac{1}{2} - q_2^2 - q_3^2) - s_z \end{bmatrix} \tag{6.54} $$

$$ J({}^{S}_{E}\hat{q}, {}^{E}\hat{d}) = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \tag{6.55} $$

$$ \begin{aligned} a_{11} &= 2d_yq_4 - 2d_zq_3; & a_{23} &= 4d_xq_3 - 2d_yq_2; \\ a_{12} &= 2d_yq_3 + 2d_zq_4; & a_{24} &= -2d_xq_1 - 4d_yq_4 + 2d_zq_3; \\ a_{13} &= 4d_xq_3 - 2d_yq_2; & a_{31} &= 2d_xq_3 - 2d_yq_3; \\ a_{14} &= -4d_xq_4 + 2d_yq_1 + 2d_zq_2; & a_{32} &= 2d_xq_4 - 2d_yq_1 - 4d_zq_2; \\ a_{21} &= -2d_xq_4 + 2d_zq_2; & a_{33} &= 2d_xq_1 + 2d_yq_4 - 4d_zq_3; \\ a_{22} &= 2d_xq_3 - 4d_yq_2 + 2d_zq_1; & a_{34} &= 2d_xq_2 + 2d_yq_3; \end{aligned} $$

Ecuațiile din intervalul (6.52)-(6.55) descriu forma generală a algoritmului aplicabilă unui câmp predefinit în orice direcție. Oricum, dacă direcția câmpului poate fi determinată doar cunoscând componentele de pe una sau două axe principale ale sistemului de coordonate global, astfel ecuațiile se simplifică considerabil. Prin convenție se știe că direcția gravitației definește verticala, axa z precum e definită în ecuația (6.56). Substituind ${}^{E}\hat{g}$ cu vectorul accelerometrului normalizat ${}^{S}\hat{a}$ pentru ${}^{E}\hat{d}$ și pentru ${}^{S}\hat{s}$, ecuațiile (6.54) și (6.55) devin ecuațiile (6.58) și (6.59)[1].

$$ {}^{E}\hat{g} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.56} $$

$$ {}^{S}\hat{a} = \begin{bmatrix} 0 & a_x & a_y & a_z \end{bmatrix} \tag{6.57} $$

$$f({}^S_E\hat{q}, {}^S\hat{a}) = \begin{bmatrix} 2(q_2q_4 - q_1q_3) - ax \\ 2(q_1q_2 + q_3q_4) - ay \\ 2(\frac{1}{2} - q_2^2 - q_3^2) - az \end{bmatrix} \tag{6.58}$$

$$J({}^S_E\hat{q}) = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix} \tag{6.59}$$

Câmpul magnetic al pământului poate fi considerat având componente doar pe axa orizontală și verticală, componenta verticală generând înclinarea câmpului care fluctuează între 65^0 și 70^0 raportat la orizontală[14]. Acesta poate fi reprezentat prin ecuația (6.60). Substituind ${}^E\hat{b}$ și normalizând măsurătorile ${}^S\hat{m}$ pentru ${}^E\hat{d}$ și ${}^S\hat{s}$ în ecuațiile (6.54) și (6.55) devin ecuațiile (6.62) și (6.63) [9].

$${}^E\hat{b} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.60}$$

$${}^S\hat{m} = \begin{bmatrix} 0 & m_x & m_y & m_z \end{bmatrix} \tag{6.61}$$

$$f({}^S_E\hat{q}, {}^E\hat{b}, {}^S\hat{m}) = \begin{bmatrix} 2b_x(0.5 - q_3^2 - q_4^2) + 2b_z(q_2q_4 - q_1q_3) - m_x \\ 2b_x(q_2q_3 - q_1q_4) + 2b_z(q_1q_2 + q_3q_4) - m_y \\ 2b_x(q_1q_3 + q_2q_4) + 2b_z(0.5 - q_2^2 - q_3^2) - m_z \end{bmatrix} \tag{6.62}$$

$$J({}^S_E\hat{q}, {}^E\hat{s}) = \begin{bmatrix} -2b_zq_3 & 2b_zq_4 & -4b_xq_3 - 2b_zq_1 & -4b_xq_4 + 2b_zq_2 \\ -2b_xq_4 + 2b_zq_2 & 2b_xq_3 + 2b_zq_1 & 2b_xq_2 + 2b_zq_4 & -4b_xq_4 + 2b_zq_2 \\ 2b_xq_3 & 2b_xq_4 - 4b_zq_2 & -4q_3 & 2b_xq_2 \end{bmatrix} \tag{6.63}$$

Cum sa enunțat anterior, măsurătorile individuale ale câmpului magnetic al pământului sau a gravitației nu oferă o orientare unică a senzorului. Pentru a obține o orientare unică se combină cele două măsurători.

Pentru optimizarea procesului de obținere a orientării senzorului este nevoie de apelarea în mai multe iterații a ecuației (6.52), pentru a calcula orientarea senzorului la fiecare măsurătoare dăcută de senzor. Pentru eficientizarea algoritmului este nevoie și de ajustarea rata de eșantionare μ la fiecare iterație, în general valoarea optimă cu care este ajustat μ este calculat prin dericvarea funcției f. Această operație prouce un volm ridicat de calcule ce va încetinii procesul, fară a aduce îmbunătățiri considerabile. În cazul acestei aplicații este suficient să se calculeze o iterație la momentul de timp stabilit de convergența reglementată de $\mu_t \geq$ rata de schimbare a orientării.Ecuația (6.64) calculează orientarea estimată ${}^S_E q_{\nabla,t}$ la momentul de timp t bazată pe estimarea precedentă ${}^S_E\hat{q}_{est,t-1}$, și de gradientul funcției obiectiv f, ∇f definit de măsurătorile senzorului ${}^S\hat{a}_t$

și ${}^{S}\hat{m}_t$ la momentul de timp t [9].

$$ {}^{S}_{E}q_{\nabla,t} = {}^{S}_{E}\hat{q}_{est,t-1} - \mu_t \frac{\nabla f}{|| \nabla f||} \tag{6.64} $$

Valoarea optimă a rata de eșantionare μ_t poate fi definită pe baza convergența orientării estimate ${}^{S}_{E}\mathbf{q}_{\nabla,t}$, evitându-se depășirile sau perioadele de eșantionare cu durate ineficient de îndelungate. Astfel μ_t se calculează cu expresia (6.65), unde Δ_t este perioada de eșantionare, ${}^{S}_{E}\dot{q}_{w,t}$ este rata orientării măsurată de giroscop, iar α este augumenatrea μ în concordanță cu zgomotul impus de măsurătorile accelerometrului și magnetometrului[1].

$$ \mu_t = \alpha||{}^{S}_{E}\dot{q}_{w,t}||\Delta_t, \alpha > 1 \tag{6.65} $$

6.2.3 Fuziunea celor două metode de determianre a orientării

Estimarea orientării sistemului de coordonate al senzorului relativ la sistemul de coordonate al pământului ,${}^{S}_{E}q_{est,t-1}$ este obținut din fuzionarea calculelor orientării din expresiile ,${}^{S}_{E}q_{w,t}$ și ${}^{S}_{E}q_{\nabla,t}$, determinate de ecuațiile (6.47) respectiv (6.64). Fuziunea celor două metode este realizată prin ecuația (6.66), unde γ_t și $(1-\gamma_t)$ sunt ponderile asociate fiecărui tip de orientare.

$$ {}^{S}_{E}q_{est,t} = \gamma_t {}^{S}_{E}q_{\nabla,t} - (1-\gamma_t){}^{S}_{E}q_{w,t}, 0 \leq \gamma_t \leq 1 \tag{6.66} $$

Valoarea optimă a γ_t se poate obține prin condiția ca divergența ${}^{S}_{E}q_w$ este egală cu covarianța ponderată a ${}^{S}_{E}q_{\nabla}$, condițiie exprimată în ecuația (6.67), unde $\frac{\mu_t}{\Delta_t}$ este rata covarianței ${}^{S}_{E}q_{\nabla}$, iar β este divergența ${}^{S}_{E}q_w$ exprimată prin valoarea dericvatei cuaternionului corespondent erorii de măsurare a giroscopului.

$$ \gamma_t = \frac{\beta}{\frac{\mu_t}{\Delta_t} + \beta} \tag{6.67} $$

Ecuațiile (6.66) și (6.67) asigură fuziunea optimă între ${}^{S}_{E}q_{w,t}$ și ${}^{S}_{E}q_{\nabla,t}$, asumând că rata covarianței ${}^{S}_{E}q_{\nabla}$ este dependentă de α, adică de zgomot. Încazul în care zgomotul este foarte mare atunci și augumentarea μ definită de expresia (6.65) va crește direct proporțional. O valoare mare a augumentării va face ca termenul ${}^{S}_{E}\hat{q}_{est,t-1}$ folosit în ecuația (6.64) să devină neglijabil, rezulând ecuațile (6.68) (6.69)

$$ {}^{S}_{E}q_{\nabla,t} \approx -\mu_t \frac{\nabla f}{|| \nabla f||} \tag{6.68} $$

$$ \gamma_t \approx \frac{\beta\Delta_t}{\mu_t} \tag{6.69} $$

Înlocuind ecuațiile (6.47), (6.68) și (6.69) în ecuația (6.66) ecuația finală devine (6.70)

$$ {}^{S}_{E}q_{est,t} = \frac{\beta\Delta_t}{\mu_t}(-\mu_t\frac{\bigtriangledown f}{||\bigtriangledown f||}) + (1-0)({}^{S}_{E}\hat{q}_{est,t-1} + {}^{S}_{E}\dot{q}_{w,t}\Delta_t) \tag{6.70} $$

Ecuația (6.70) poate fi simplificată sub forma expresiei (6.71) unde ${}^{S}_{E}\dot{q}_{est,t}$ este estimarea ratei de schimbare a orientării definită în ecuația (6.72) și ${}^{S}_{E}\dot{\hat{q}}_{E,t}$ reprezintă direcția erorii ${}^{S}_{E}\dot{q}_{est,t}$ definită în ecuația (6.73)

$$ {}^{S}_{E}q_{est,t} = {}^{S}_{E}\hat{q}_{est,t-1} + {}^{S}_{E}\dot{q}_{est,t}\Delta_t \tag{6.71} $$

$$ {}^{S}_{E}\dot{q}_{est,t} = {}^{S}_{E}\dot{q}_{w,t} - \beta{}^{S}_{E}\dot{\hat{q}}_{E,t} \tag{6.72} $$

$$ {}^{S}_{E}\dot{\hat{q}}_{E,t} = \frac{\bigtriangledown f}{||\bigtriangledown f||} \tag{6.73} $$

Se poate observa că filtrul calculeaza valoarea schimbării orientării măsurate pe baza gyroscopului, impicând erorile de măsurare ale giroscopului, β, ce vor fi eliminate pe baza direcției erorii estimate a giroscopului, ce a fost calculată cu ajutorul parametrilor accelerometrului și magnetometrului.

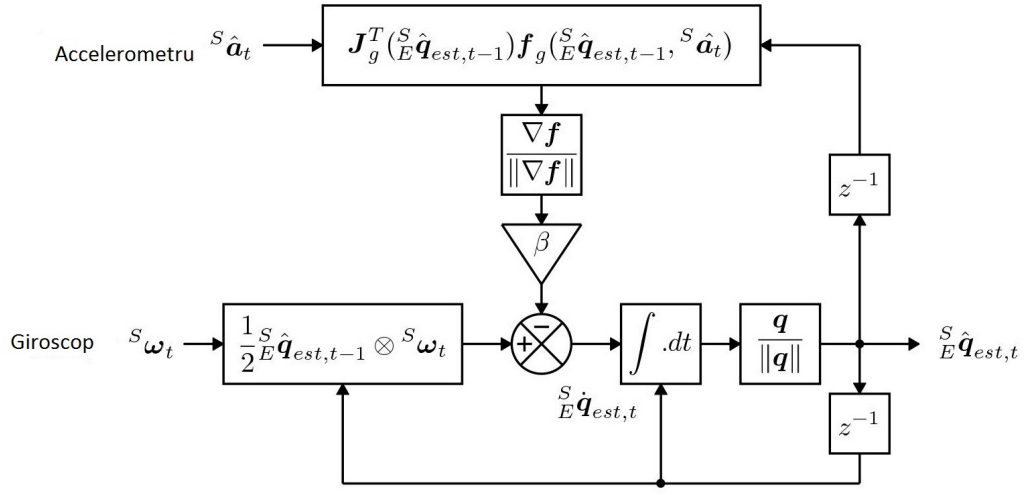

FIGURA 6.14: Schema bloc de determinare a orientării pe baza fuzionării celor două metode

6.2.4 Compensarea deformării magnetice

Măsurătorile câmpului magnetic al pământului vor fi afectate de prezența materialelor feromagnetice din apropierea magnetometrului. Investgigând efectele distorsiunilor generate de comonenta magnetica în determinaea orientări, s-a observat că există numeroase erori incluse de interferențele introduse de componentele metalice [15][1].

Surse interferatoare ce sunt fixe în cadrul sistemuli de coordonate al senzorului pot fi eliminate prin calibrarea senzorilor [16] [10]. Aceste surse generatoare de interferențe pot genera erori de determinare a înclinării planului orizontal relativ la suprafața pământului, erori ce nu pot fi corectate decât raportând la o poziție de referință. Erorile de înclinare ale planului vertical cu pământul pot fi compensate cu ajutorul accelerometrului ce generează poziția senzorului.

Direcția câmpului magnetic în sistemul de coordonate al pământului la timpul t, ${}^{E}\hat{q}_t$ poate fi calculat pe baza datelor magnetometrului normalizate, ${}^{S}\hat{m}_t$, rotit pe baza orientării senzorului calculată de filtru, ${}^{S}_{E}\hat{q}_{est,t-1}$ descrisă de ecuația (6.74). Această eroare poate fi eliminată dacă direcția de câmpului magnetic de referință terestru ${}^{E}\hat{b}_t$ este egală cu înclinarea (6.75) .

$$ {}^{E}\hat{q}_t = \begin{bmatrix} 0 & h_x & h_y & h_z \end{bmatrix} = {}^{S}_{E}\hat{q}_{est,t-1} \bigotimes {}^{S}\hat{m}_t \tag{6.74} $$

$$ {}^{E}\hat{b}_t = \begin{bmatrix} 0 & \sqrt{h_x + h_y} & 0 & h_z \end{bmatrix} \tag{6.75} $$

Compensând astfel zgomotul parametrilor magnetici, rămân afectate de zgomot datele privind poziția senzorului.

6.2.5 Controlul implicării erorii furnizate de giroscop

Obiectivul acestei sarcini a filtrului AHRS este acela ca valoarea erorii staționare furnizate de giroscop să fie zero. Orice implementare practică cu senzori IMU sau MARG trebuie să țină cont de această condiție, find foarte importantă în cadrul funcționării. Avantajul folosirii bazelor filtrului Kalman face ca modelul AHRS să fie capabil să estimeze eroarea giroscopului, ca o stare adițională pe baza modelului sistemului[**?**], [18]. Mahony, în implementare propusă de el asupra filtrului AHRS [10] arată că implicarea erorii furnizate de giroscop poate fi compensată prin orientarea filtrului spre integrarea erorii în bucla de reglare a ratei de schimbare a orientării. Normalizarea erorii estimate în rata de zchimbare a orientării, ${}^{S}_{E}\hat{q}_E$, va fi exprimată ca eroarea unghiulară pe fiecare axă a giroscopului (6.76). Implicarea erorii furnizate de giroscop ${}^{S}w_b$, este influențată direct de eroare, aceasta poate fi eliminată prin ponderarea cu un coeficient corespunzător ζ. Acest procedeu va genera compensarea măsurătorilor giroscopului ${}^{S}w_c$, e furnizată de ecuațiile (6.77) și (6.78)[1].

$$ {}^{S}w_{E,t} = 2{}^{S}_{E}\hat{q}_{est,t-1} \bigotimes {}^{S}_{E}\hat{q}_{E,t} \tag{6.76} $$

$$ {}^{S}w_{b,t} = \zeta \sum_{t} {}^{S}w_{E,t}\Delta_t \tag{6.77} $$

$$^{S}w_{c,t} = {}^{S}w_t - {}^{S}w_{b,t} \tag{6.78}$$

Rezultatele compensării implicaților aduse de erorile măsurătorilor giroscopului $^{S}w_c$ pot fi înlocuite cu măsurătorile giroscopului ^{S}w în ecuația (6.46). Valoarea erorii unghiulare pe fiecare axă $^{S}w_E$ este egală cu derivata cuaternionului în unitatea de timp [9].

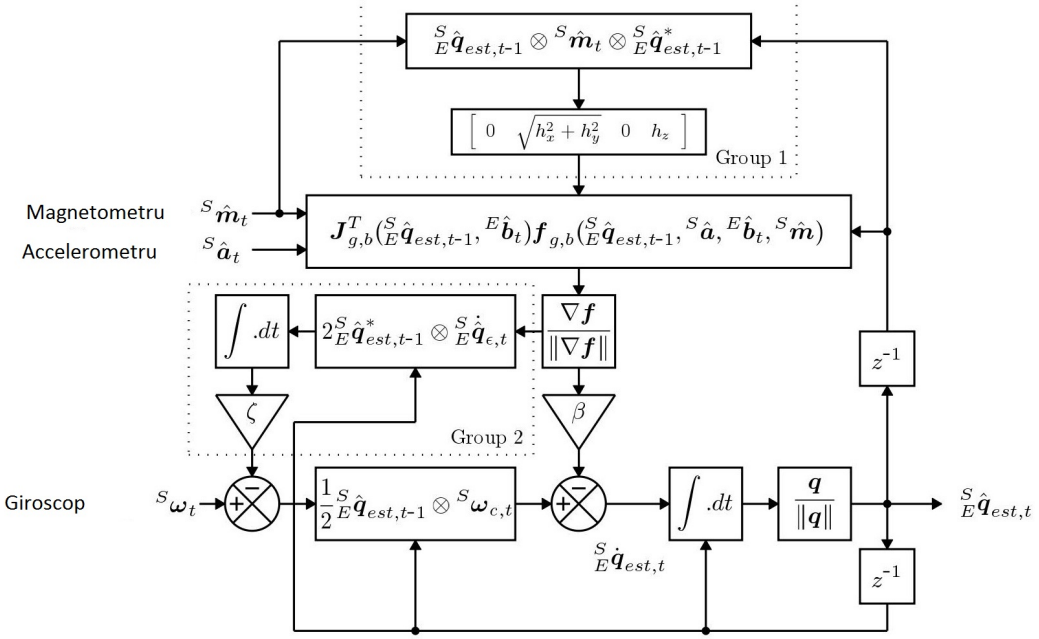

FIGURA 6.15: Schema bloc de determinare a orientării pe baza fuzionării celor două metode, compensarea distorsiunilor magnetice(Group1) și compensarea erorii giroscopului (Group2).

7. Rezultate experimentale

Analiza datelor citite de senzorii inerțiali
Rezultatele algoritmilor de determinare a orientării
Rezultatele determinării orientării pe baza unghiurilor lui Euler și a filtrului AHRS

Principiile de determinare a orientării în spațiul tri-dimensional, enunțate în capitolele anterioare pot fi dovedite prin fuziunea dintre datele captate de structura hardware și procesarea acestora pe baza structurii software cu scopul determinării parametrilor orientativi ce descriu un corp în spațiul terestru.

Pe baza arhitecturi hardware expusă în capitolul 2 ce are ca scop captarea parametilor inerțiali ce pot fi procesați prin mai multe platforme software, platforme ce pot fi rulate la nivelul plăcii de dezvoltare sau la nivelul calculatorului. În funcție de comlexitatea algoritmului și facilitățile dorite folosit se aleg platformele optime și dispozitivele pe care acestea rulează.

Independent de algoritmii de determinare implementați și platforma hardware pe care rulează aceștia, este nevoie de o reprezentre a detelor de intrare, eventual a datelor preliminare, și a rezultatelor procesării pentru a observa comportările sistemului și a face analize asupra acestora. În acest scop, în cadrul acestui proiect sa folosit platforma Matlab expusă în subcapitlolul 3.3, ce citește datele de la nivelul structurii hardware, pe baza comunicației seriale detaliate în subcapitolul 3.4.2 și realizează o reprezentare grafică a acestora.

Pentru simularea orientării determinate de algoritmii de filtrarea și modelare a parametriilor inerțiali sa folosit platforma Unity3D detaliată în subcapitolul 3.1 ce primește datele procesate de algoritmii rulați pe placa de dezvoltare Arduino prin conexiunea expusă în secțiunea 3.4.1.

7.1 Analiza datelor citite de senzorii inerțiali

Asupra datelor citite de senzorii inerțiali sau formulat o serie de formulări matematice a comportării acestora având în vedere toți factorii perturbatori ai mediului înconjurător.

7.1.1 Analiza parametrilor în regim static

În cadrul acestei secțiuni se observă comportărole intrărilor sistemului în cazul în care senzorul nu este deviat din poziția inițială. Astfel rezultă următoarele caracteristici ale datelor giroscopului:

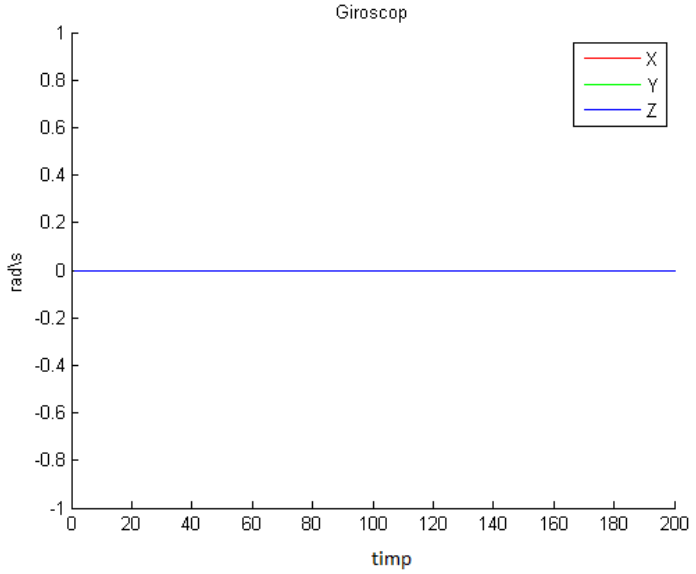

FIGURA 7.16: Parametrii giroscopului în regim staționar

În această reprezenatre a datelor furnizate de giroscop se observă că viteza unghiulară are valoarea zero pe toate cele trei direcții spațiale, giroscopul ne fiind scos din starea inițială.

Aceste caracteristici nu mai sunt valabile pentru valorile accelerometului care generează o repartiție a semnalului vizibilă în figura 7.17. În această figură se observă că se adeveresc principile enunțate în capitolul 1, cu privire cărora accelerația este afectată de zgomote. Principala influență asupra parametriilor accelerației e adusă de influența forței gravitaționale exerciate de Pământ. Precum sa enunțat pe parcursul lucrării se observă că cea mai importantă influență a gravitației este asupra datelor generate de acceleormetru pe axa z, paralelă cu gravitația. Pe acestă direcție senzorul detectează a accelerație constantă, dar diferită cu mult de zero, valoarea reală a acesteia. Prin această comportare se poate observa că, perturbația instalată de gravitație asupra axei

paralele pe direcția acesteia, trebuie compensată în scopul obținerii unei comportări reale.

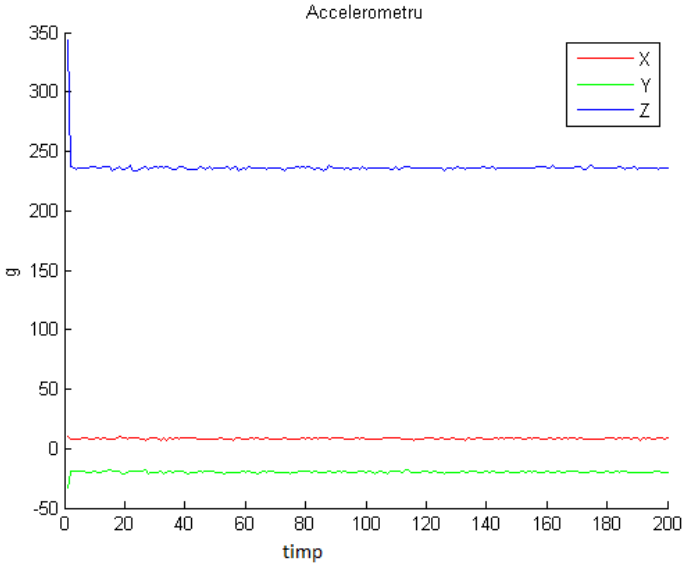

FIGURA 7.17: Parametrii accelerometrului în regim staționar

7.1.2 Analiza datelor de intre în regim dinamic

În această secțiune se încearcă observarea distribuței parametrilor accelerometrului și giroscopului în funcție de orientarea senzorului. Această evaluare este o evaluare pur subiectivă, pe baza repartiției semnalelor se poate observa o orientare posibilă, dar nu sigură sau măsurabilă.

Astfel în imaginea următoare se poate observa pe baza semnalelor provenite de la giroscop un model al rotaților realizat de senzor.

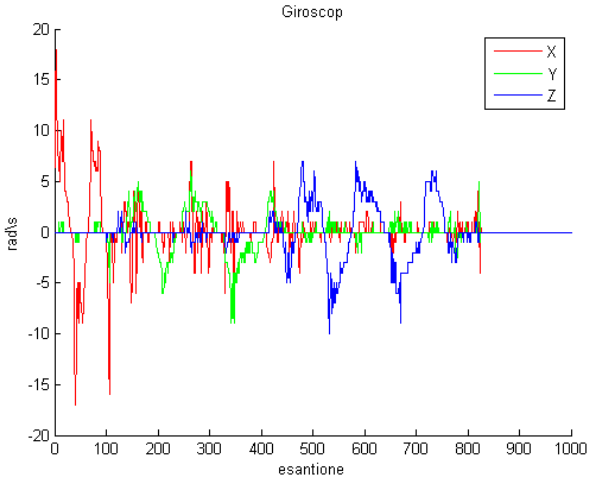

FIGURA 7.18: Datele furnizate de giroscop în regim dinamic

Pe baza imaginii anterioare se poate observa, în diferite secțiuni, o rotație accentuată. Astfel în prima secțiune se observă valori ridicate ale vitezei unghiulare realizate pe axa x, putând presupune că senzorul realizează o rotație în jurul axei x. În a doua secțiune a imaginii se poate observa o fluctuație a valorilor semnalului pe axa y, putând presupune ca sunt rezultate ale rotație în jurul axei z. Ultima secțiune a imaginii conține fluctuații importante ale semnalului furnizat de axa z a giroscopului, comporate ce poate determina o rotație în jurul axei z.

Pe baza presupunerilor anterioare se poate presupune o orientare, operând anumite valori de threshold se poate spune că între anumite valori avem ununghi α în raport cu axa x, dar acesta este o evaluare nesigură și orientativă.

Semnalele echivalente accelerometrului, din modelul mișcărilor dezbătut anterior, sunt reprezentate în imaginea:

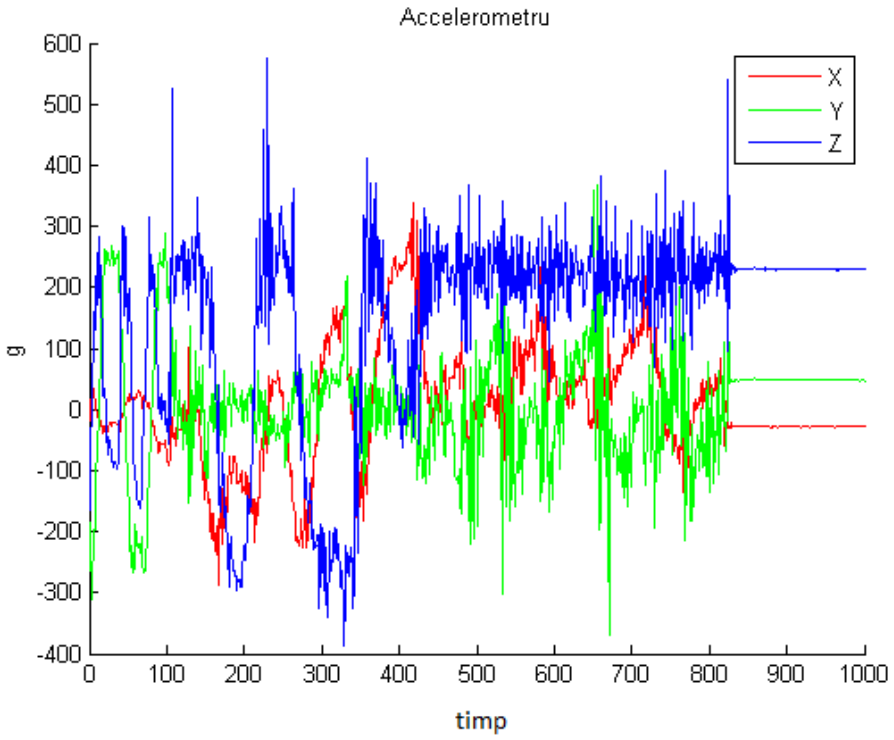

FIGURA 7.19: Datele furnizate de accelerometru în regim dinamic

Din repartiția semnalelor realizată anterior nu se poate determina un model al accelerației, dar se observă fluctuații importante ale valorilor accelerației pe axa z, valori afectate de accelerația gravitațională. În funcție de rotațiile realizate de corp se poate observa că la un moment dat și celealte axe ale sistemului sunt paralele cu vectorul accelerației gravitațiomnale, putându-se presupune existența unor perturbații ridicate ce afectează accelerațiile de pe axele x respectiv z.

7.2 Rezultatele algoritmilor de determinare a orientării

Presupunerile anterioare asupra unei posibile orientări a corpului în spațiul 3D pe baza fluctuaților parametrilor giroscopului reprezinta ipoteze nesigure și imprecise. În scopul formulării unei unei expresii precise și sigure a orientării unui corp sau elaborat o serie de algoritmi matematici ce determină, pe baza parametriilor inerțiali, o formulare veridică a orientării în spațiul tri-dimensional.

O parte din algoritmii matematici, elaborați în decursul timpului, ce pot determina orientarea 3D sau amintit în secțiunea 1.3. Dar principile implementate în acest proiect au fost bazate pe metoda matricei cosinusului (DCM) și unghiurile lui Euler, respectiv metoda filtrului AHRS pe baza cuaternionilor.

7.2.1 Rezultatele determinării orientării pe baza unghiurilor lui Euler și a filtrului AHRS

Reprezentarea sub forma unghiurilor lui Euler a fost detaliată în subcapitolul 4.1, enunțând rotațiile realizate în funcție de cele trei axe , în care se reprezintă rotația în sensul acelor de ceasornic funcție de axa z (yaw), rotația în funcție de axa y (pitch),rotația în fucție de axa x (roll).

Determinarea orientării pe baza unghuirilor lui Euler generează rezultate bune fară un volum mare de calcul, dar principala problemă a acestei reprezentări este generată de defetul principal al acestei metode, acesta este principiul blocării gimbal este reprezentat de cazul în care două inele axiale se plasează pe același plan, moment în care sistemul 3D se transforma într-un model 2D.

În scopul evitării acestui defect al metodei unghiurilor lui Euler s-a încercat implementarea unui filtru AHRS, un filtru dezvoltat din metoda filtrului Kalman și filtrul Kalman extins detaliate în capitolul 6, ce se bazează pe descrierea spațiului tridimensional printr-un vector al cuaternionilor dezbătut în secțiunea 4.2. Această reprezentare a spațiului 3D sub forma de cuaternioni elimină blocarea gimbal, îmbunătațind calitatea determinării orientării.

Filtrul AHRS dezvoltat în capitolul 7 generează o calculare a orientării, implicând și componenta câmpului magnetic al pământului, copmensând foarte bine erorile de măsurare ale giroscopului și reușind să elimine proceseze componenta accelerației gravitaționale în scopul îmbunătățirii performanțelor.

Comparând metoda filtrului AHRS cu metoda filtrării Kalman se observă ca versiunea filtrului AHRS realizată de Madgwick are o viteză de procesare mai bună decât filtrul Kalman, realizănd calcule matematice mult mai reduse, generând rezultate asemănătoare calitativ.

7.2.1.1 Rezultatele determinării orientării în regim static

Rezultatele experimentale pot fi analizate prin reprezentarea rezultatelor generate de cele două metode în același grafic, astfel putându-se observa diferențele acestora.

Imaginile următoare reprezintă detectarea orientării în regim static, paketul de senzori ne fiind scos din echilibru:

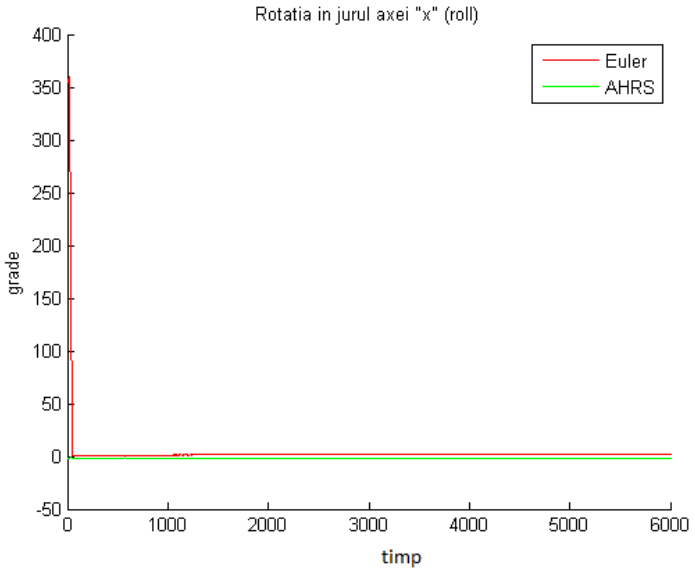

FIGURA 7.20: Reprezentarea unghiului în raport cu axa x

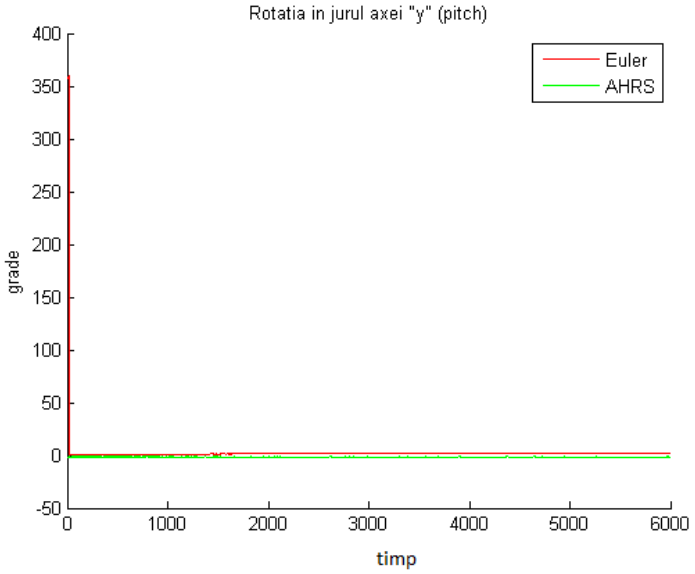

FIGURA 7.21: Reprezentarea unghiului în raport cu axa y

În imaginile precedente se poate observa că, dupa calibrare, cele două metode au o comportatrea asemănătoare, observându-se că senzorul are o poziție apropiată de zero grade în raport cu axa x, și y.

Această comportare va fi diferită în momentul în care se încercă detectare unghiului în raport cu axa z, această reprezentare se poate analiza în imaginea:

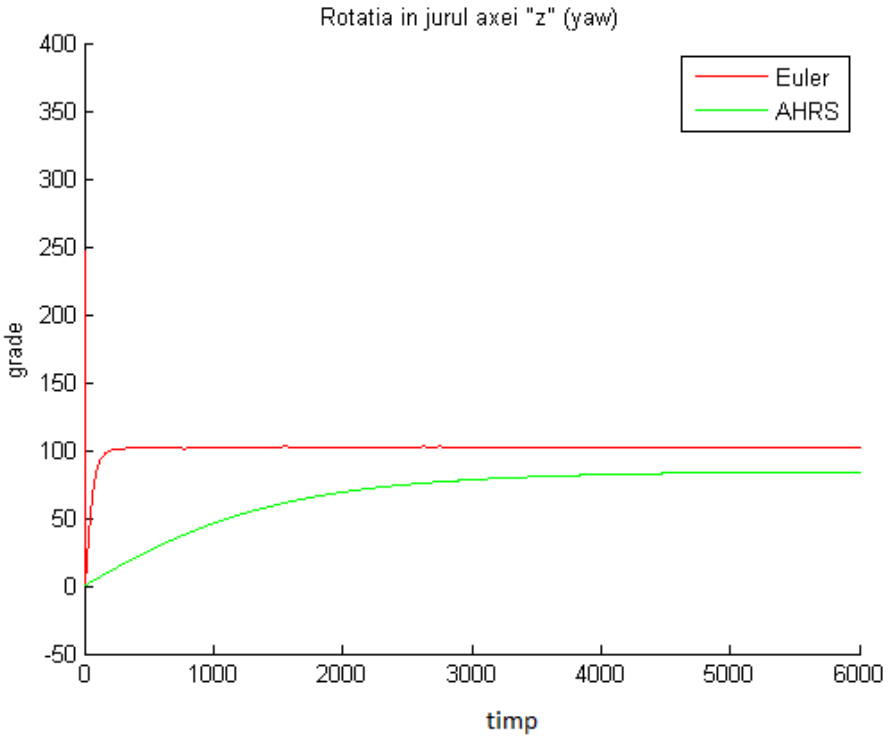

FIGURA 7.22: Reprezentarea unghiului în raport cu axa z

Din reprezentarea orintării pachetului de senzori față de axa z a sistemului spațial se observă că în cazul metodei unghiurilor lui Euler stabilizarea se face mult mai rapid decât timpul de stabilizare al metodei AHRS, dar această are un rezultat mult mai bun decât cel al al metodei unghiurilor lui Euler. În cazul acesta unghiul cu raportat față de axa z este apropiat de 90^0, observându-se că filtrul AHRS se stabilizează la această valoare.

7.2.1.2 Rezultatele determinării orientării în regim dinamic

Determinarea orientării în regim dinamic este mult mai dificilă datorită schimbărilor bruște ale parametrilor senzorilor. Comportarea orientării aplicând cele două metode, duce la detectarea unei orientări similare în raport cu axa x și y, observându-se că orientarea detectată cu metoda unghiurilor lui Euler generează un răspuns mult mai liniar, fără a detecta fluctuațile rapide.

Analizând orientările detectate de filtrul AHRS se poate observa că acesta detectează fluctiații mult mai dese ale miscărilor unghiulare realizate de sistem. Acest poate fi folositor pentru precizie, dar poate fi deranjant pentru stabilitate și constanță.

Analiza realizată anterior se bazează pe următoarele figuri ce descriu comparativ comportarea sistemului aplicând cele două metode:

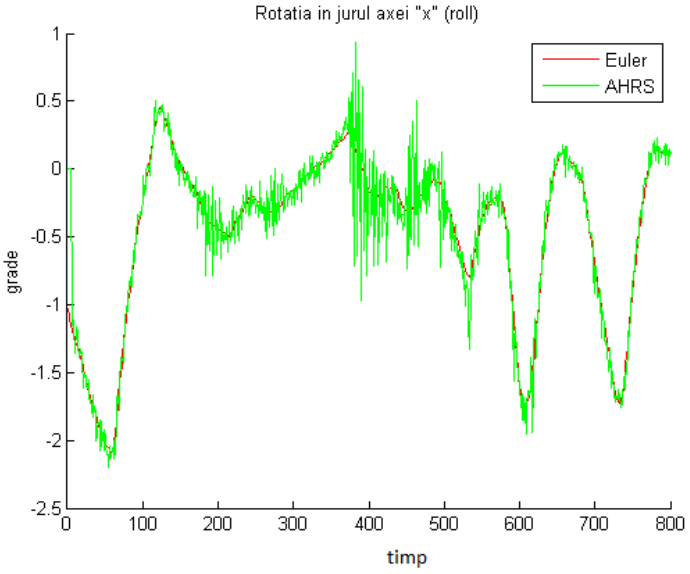

FIGURA 7.23: Reprezentarea orientării în raport cu axa x

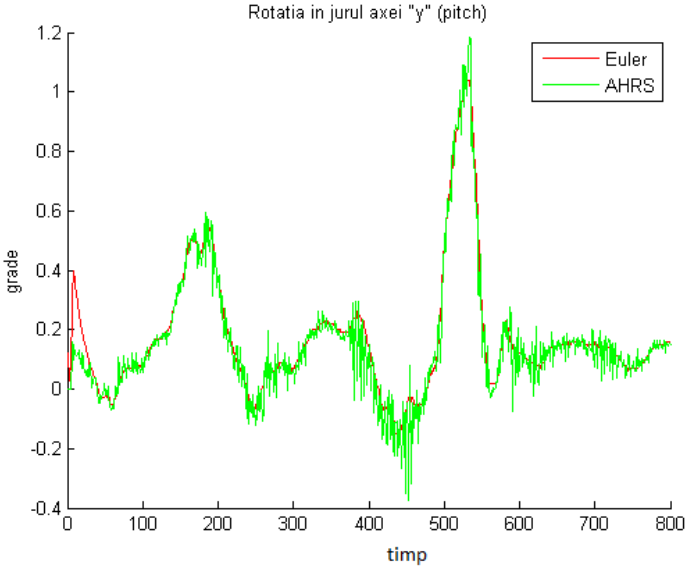

FIGURA 7.24: Reprezentarea orientării în raport cu axa y

Precum în cadrul comportării în regim stabil, reprezentarea orientării dinamice în raport cu axa z a sistemului 3D prezintă soluții diferite genertate de cele două metode. Pe baza comportării semnalului se poate observa că metoda filtrului AHRS are o comportare foarte diferită în raport cu metoda ungiurilor lui Euler, în special în primele perioade de timp, când filtrul AHRS se calibrează și se stabilizează. După stabilizarea modelului AHRS cele două modele genereaza un model asemănător al semnalului, dar

rezultatul filtrului AHRS est mai precis comparativ cu cel al modelului unghiurilor lui Euler.

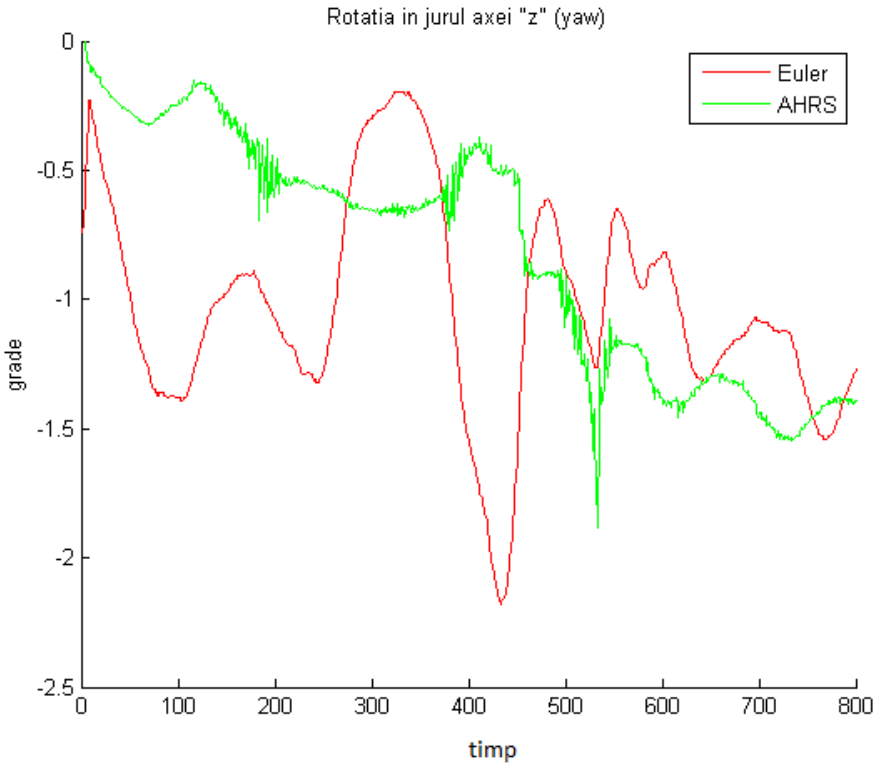

FIGURA 7.25: Reprezentarea orientării în raport cu axa z

Pentru a vizualiza orientarea mai apropiat de realitatea mișcărilor sa realizat o animație în limbajul Matlab care preia datele orientării si le transpune într-o animație ce arată orientarea tridimensională. Un cadru al acestei animații este în imaginea:

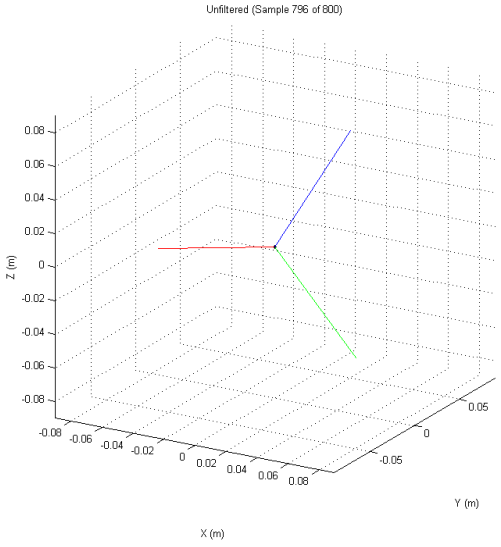

FIGURA 7.26: Reprezentarea orientării în 3D

8. Concluzii și dezvoltări ulterioare

Concluziile proiectului
Direcții de dezvoltare ulterioară

8.1 Concluziile proiectului

Acest proiect a constituit o introducere în domeniul determinării orientării în spațiul tri-dimensional și a raportării corpurilor, aflate în mișcare, la condițiile impuse de forțele terestre. Astfel accelerația gravitațională și componenta câmpului magnetic reprezentând un factor important în cadrul orientării corpurilor.

Demersurile făcute în cadrul acestui proiect au dus la identificarea unui model de structură hardware ce poate capta parametrii de accelerometru, giroscop și magnetometru. Această structură este disponibilă tuturor doritorilor, la prețuri reduse. Placa de dezvoltare Arduino reprezintă o structură hardware robustă, complexă , rapidă și foarte fiabilă, fiind bine documentată de producător, astfel este foarte ușor de învățat structura hardware și facilitățile acesteia. Analizând facilitățile pachetului de senzori inerțiali AltIMU-10, se poate spune că este un pachet de senzori foarte complex, cu performanțe foarte bune, având componente ce pot măsura date la o fregvență ridicată, facilitate ce este foarte folosiroare în aplicațiile în timp real.

Elaborând concluzii asupra structurii software, s-a observat că există foarte multe limbaje de programare ce pot fi folosite în combinație cu placa de dezvoltare Arduino, facilitățile comunicației seriale fiind foarte folositoare, astfel putem spune că programarea software se poate realiza întru-un limbaj de nivel înalt, transferul codului și interpretarea lui în limbaj C/C++ putând fi făcută cu ajutorul unor librării specifice. Această facilitate aduce structurii hardware o flexibilitate importantă, iar structurii software o aplicabilitate complexă.

Cea mai provocatoare parte a acestui proiect a fost reprezentată de determinarea algoritmilor pentru determinarea orientării. Această secțiune a dus la studii amănunțite

a algoritmilor folosiți și dezvoltați în domeniile aplicabile orientării. Astfel primul algoritm studiat a fost algoritmul unghiurilor lui Euler, principiu ce face baza orientării în spațiul 3D și care are performanțe bune vizibiele prin experimentele folosite, fiind implementabil în aplicațiile în timp real, datorită timpului de răspuns foarte rapid și timpului scurt de stanbilizare. Studierea mai amănunțită a acestui principiu a scos la iveală o problemă importantă a acestui algoritm, reprezentată de pringimiul blocării gimbal. Blocarea gimbal face ca sistemul să se blocheze în momentul în care orientarea sistemului și face ca două planuri ale axelor de coordonate să fie paralele, atunci sistemul devenind unul 2D, fiind nevoie de o miscare necontrolată pentru deblocare. Principiul blocării gimbal face ca modelul ungiurilor lui Euler să fie un model problematic în determinarea orientării 3D.

Pentru a evita problemele evidențiate de principiul unghiurilor lui Euler s-a observat că, în multe studii sa folosit reprezentarea spațiului 3D în forma cuaternionilor, o extindere matematică a numerelor complexe, iar orientarea pe baza acestora se face printr-o filtrare Kalman a datelor inerțiale, astfel se reduc influențele gravitației și câmpului magnetic terestru, orientaea fiind determinată mult mai precis prin elaborarea unei estimării a acestei analizând starea precedentă, rezultând un control mult mai bun. Această metodă pare ideală, dar problema ei este reprezentată de complexitatea calculelor și multitudinea de iterații, fapt ce o duce la întârzieri, ce afectează implementarea într-o aplicație în timp real.

Aprofundând studiile sa observat că aceste întârzieri ale filtrului Kalman pot fi evitate printr-un filtru derivat din filtrul Kalman, filtrul AHRS elaborat de Mahony și îmbunătățit de Madgvick, care implementează mai puține calcule și iterații, astfel rezultă un răspuns mai bun al sistemului, la performanțe asemanătoare filtrului Kalman.

Prin implementarea metodelor unghiurilor lui Euler și algoritmului AHRS s-a observat că cele două metode prezintă rezultate asemănătoare în determinarea orientării în funcție de axele x și y, diferența principală este observată la determinarea orientării axei z, în care metoda filtrului AHRS are rezultate mult mai precise, dar cu un timp de stabilizare a semnalului mult mai mare decât cel al metodei Euler.

Pe baza acestor concluzii experimentale se poate spune că metoda unghiurilor lui Euler este cea mai bună pentru determinarea orientării 2D, dar în cazul în care se dorește monitorizarea orientării în funcție de axa z este nevoie de un filtru ce compensează componenta accelerației gravitaționale, a influenție câmpului magnetic și a erorilor giroscopului, cel mai eficient algoritm fiind expus în filtrul AHRS.

8.2 Direcții de dezvoltare ulterioară

Pe baza realizărilor acestui proiect, și obsrvării altor aplicții ale metodelor studiate în această aplicație, dar și prin prisma dezvoltării sistemelor de acest tip se pot trasa mai multe direcții de dezvoltare.

O direcție de dezvoltare ar duce la compnesarea unui neajuns al aplicației curente. Acest neajus este legat de conectarea cablată între calculator și arhitectura hardware. Rezolvarea acestei probleme poate fi făcută prin integrarea unei componente ce furnizează conexiunea la internet, în cadrul plăcii de dezvoltare Arduino. Astfel printr-o conexiune TCP/IP se poate realiza conexiune datelor la nivel de LAN. Tot această direcție de dezvoltarre poate fi făcută prin dezvoltarea aplicație pe o platforma Rasberry Pi, care conține implicit un modul ce permite conexiunea wireless, iar sistemul de operare Linux, folosit de această placă de dezvoltare este mult mai stabil, si permite mai multe facilități legate de tipul conexiunii, menționând și timpul de calcul mult mai eficient.

O altă direcție de dezvoltare este reprezentată de îmbunătățirile ce pot fi aduse, cu scopul implementării în cadrul aplicațiilor Unity3D, astfel se poate dezvolta pe baza algoritmilor dezvoltați în acest proiect, și prin îmbunătățirea structurii hardware, un prototip de mănușă ce va oferii utilizatorului o experiență, în animația 3D, conformă cu mișcările realizate de mâna și degetele sale. Astfel jocul dezvoltat va fi mult mai interactiv, calite ce poate fi sporită prin adăugarea unei componente haptice, în cadrul acestei mănuși. Această componetă oferindu-i utilizatorului un răspuns tactil la acțiunile realizate de el în cadrul jocului.

Precum s-a observat în secțiunea de introducere a proiectului, studiile de determinare a orientării se folosesc intens în proiectele aerospațiale. Pe baza studiilor și cunoștințelor dobândite în acest proiect se poate trasa un obiectiv de realizare unui quadocopter ce va fi controlat prin miscările mâinii utilizatorului. Acestă dezvoltare este una foarte complexă având în vedere că implică și controlul motoarelor, pe baza orientării, dar reprezintă o provocare extrem de interesantă.

Bibliografie

[1] Madgwick, S.O.H. "An Efficient Orientation Filter for Inertial and Inertial/Magnetic Sensor Arrays", 30 aprilie 2010. pp. 1-32

[2] John J. Craig. "Introduction to Robotics Mechanics and Control", Pearson Education International, 2005.

[3] David R. Pratt, Robert B. McGhee, Joseph M. Cooke, Michael J. Zyda. "Flight simulation dynamic modelling using quaternions" Presence, Vol.1, No.4,1994,pp.404-420.

[4] Q. Laddeto, J. van Seeters, S. Sokolowski, "Digital magnetic compass and gyroscope for dismounted soldier position and navigation". Proc.NATO-RTO Meetings, Istambul, 2002.

[5] M. J. Caruso, "Applications of magnetoresistive sensors in navigation systems," Sens. Actuatiors,1997, pp. 15-21.

[6] M. S. Grewal, L. R. Weill, and A. P. Andrews, "Global Positioning Systems, Inertial Navigation and Integration". 2007, pp. 525.

[7] Xi Chen."Human Motion Analysis with Wearable Inertial Sensors" , 2013, pp. 165.

[8] Xiaoping Yun, Eric R. Bachmann."Design, Implementation, and Experimental Results of a Quaternion-Based Kalman Filter for Human Body Motion Tracking".IEEE TRANSACTIONS ON ROBOTICS, VOL. 22, NO. 6, Decembrie 2006, pp.12.

[9] Madgwick, S.O.H. , Harrison A.J.L., Vaidyanathan, R. "Estimation of IMU and MARG Orientation Using a Gradient Descent Algorithm", IEEE Inter. Conf. on Rehabilitation Robotics Rehab , Zurich , Swizerland, 29 iunie – 1 iulie, 2011, pp.179-185.

[10] Mahony R., Hamel T., Pimlin J.M. "Nonlinear Complementary Filters on the Special Orthogonal Group", IEEE Trans. on Automatic Control, 2008, pp.1203-1217

[11] Mark Euston, Paul Coote, Robert Mahony, Jonghyuk Kim, Tarek Hamel."A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV"

[12] Matthew Watson."The Design and Implementation of a Robust AHRS for Integration into a Quadrotor Platform", mai 2013.

[13] Nikolas Trawny ,Stergios I. Roumeliotis."A Tutorial for Quaternion Algebra",Multiple Autonomous Robotic Systems Laboratory, TR-2005-002, Rev. 57 March 2005.

[14] John Arthur Jacobs. "The earth's core", volume 37 of International geophysics series. Academic Press,ediția a 2-a, 1987.

[15] E. R. Bachmann, Xiaoping Yun, C. W. Peterson. "An investigation of the efects of magnetic variations on inertial/magnetic orientation sensors".IEEE Interna- tional Conference on Robotics and Automation ICRA '04, volume 2, aprilie 2004,pp. 1115-1122.

[16] J. F. Vasconcelos, G. Elkaim, C. Silvestre, P. Oliveira, B. Cardeira. "A geometric approach to strapdown magnetometer calibration in sensor frame. In Navigation, Guidance and Control of Underwater Vehicles", volume 2, 2008.

[17] E. Foxlin."Inertial head-tracker sensor fusion by a complementary separate-bias kalman filter". In Proc. Virtual Reality Annual International Symposium the IEEE, 30 Martie -3 Aprilie 1996, pp.185-194,267.

[18] N.H.Q. Phuong, H.-J. Kang, Y.-S. Suh, and Y.-S. Ro. "A DCM based orientation estimation algorithm with an inertial measurement unit and a magnetic compass". Journal of Universal Computer Science, 2009, pp.859-876.

[19] Sarvenaz Salehi, Gabriele Bleser, Norbert Schmitz, Didier Stricker."A Low-cost and Light-weight Motion Tracking Suit".IEEE 10th International Conference on Autonomic and Trusted Computing, 2013, pp.474-479.

[20] Saehoon Yi, Piotr Mirowski, Tin Kam Ho, Vladimir Pavlovicz."Pose Invariant Activity Classification for Multi-Floor Indoor Localization".

[21] Naghshineh, Golafsoun Ameri, Mazdak Zereshki, S.Krishnan, M.Abdoli-Eramaki."Human Motion capture using Tri-Axial accelerometers".

[22] Nguyen Ho Quoc Phuong, Hee-Jun Kang, Young-Soo Suh."A DCM Based Orientation Estimation Algorithm with an Inertial Measurement Unit and a Magnetic Compass".Journal of Universal Computer Science, vol. 15, no. 4 (2009), pp.859-876.

[23] *** “Arduino”, 2008. [Online]. Available: http://www.arduino.cc.

[24] *** “Pololu”, 2010. [Online]. Available: https://www.pololu.com/product/2470

[25] *** “Unity3D”, 2012. [Online]. Available: http://unity3d.com/

[26] *** “Matlab”, 2011. [Online]. Available: http://www.mathworks.com/products/matlab/