

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. Arhitectura hardware	5
2.1 Placa de dezvoltare Arduino Uno	5
2.2 Pachetul de senzori 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 Vizualizarea datelor în editorul arduino	15
3.3 Platforma Matlab	16
3.3.1 Structura sistemului MATLAB	17
3.4 Conexiunea dintre Arduino și calculator	18
3.4.1 Conexiunea dintre placa de dezvoltare Arduino și Unity	19
3.4.2 Conexiunea dintre placa de dezvoltare 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	24

4.2.5	Determinarea unghiurilor lui Euler pe baza cuaternionilor	25
5.	Filtrarea datelor	26
5.1	Filtrul Kalman	26
5.1.1	Predicţia	26
5.1.2	Măsurarea	27
5.1.3	Asocierea datelor	27
5.1.4	Corecţie	28
5.1.5	Vectorul de stare	28
5.1.6	Predicţia modelului şi 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 predicţiei	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 covarianţă	32
5.3.5	Ecuatiile 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	36
6.2.3	Fuziunea celor două metode de determinare 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 inerţiali	44
7.1.1	Analiza parametrilor în regim static	44
7.1.2	Analiza datelor de între în regim dinamic	45
7.2	Rezultatele algoritmilor de determinare a orientării	47
7.2.1	Rezultatele determinării orientării pe baza unghiurilor lui Euler şi a filtrului AHRS	47
7.2.1.1	Rezultatele determinării orientării în regim static	48
7.2.1.2	Rezultatele determinării orientării în regim dinamic . .	49
8.	Concluzii şi dezvoltări ulterioare	52
8.1	Concluziile proiectului	52
8.2	Direcţii de dezvoltare ulterioară	54

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	Animatie Unity	11
3.5	Editorul Unity	12
3.6	Uneltele <i>Transform</i> și <i>Rigidbody</i>	13
3.7	Confirmarea atașării scriptului	14
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 și 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 în 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 în 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 şi un senzor de accelerometru tri-axial şi mai nou s-a implementat şi un senzor de magnetometru tri-axial. Această îmbunătăţire adusă acestei structuri hardware conduce la o structură hibridă numită MARG (Magnetic, Angular Rate, Gravity), ce are performanţe mult mai bune. Având în vedere că bazele studiilor bazate pe senzorii inerţiali 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 parametrilor determinaţi de un senzor inerţial se poate realiza o detaliere a parametrilor măsurăţi de fiecare componentă a acestuia. Accelerometrul măsoară, precum îi spune şi numele, acceleraţia incluzând şi componenta gravitaţională ce influenţează măsurătorile accelerometrului [11]. Giroscopul realizează detectarea vitezei unghiulare realizate de corp, viteza unghiulară fiind asociată în acest caz cu rotaţia corpului în jurul unei axe de coordonate. Cea mai recentă componentă a senzorilor IMU, magnetometrul realizează măsurători asupra intensităţii şi direcţiei câmpului magnetic, în special ale câmpului magnetic exercitat de Pământ asupra senzorului [22].

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

Accelerometrul tri-axial este un traducător 3D ce e compus din trei traducătoare dispuse conform dispunerii sistemului de coordonate spaţial. Din punct de vedere electronic este un dispozitiv ce converteşte acceleraţia într-un semnal electric, putând măsura atât acceleraţia statică cât şi acceleraţia dinamică, menţionând faptul că acceleraţia dinamică este afectată şi de alte forţe fizice, nu doar de gravitaţie precum e afectată acceleraţia statică. În cadrul măsurătorilor accelerometrului intervine un factor perturbator foarte important generat de influenţa gravitaţiei asupra acceleraţiei, aceasta influenţând în principal acceleraţia pe axa z . Datele furnizate de accelerometru sunt organizate sub forma unui vector cu trei elemente (1.1), [21].

$$a = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix} \quad (1.1)$$

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

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

$$S_w = \begin{bmatrix} w_x & w_y & w_z \end{bmatrix} \quad (1.2)$$

Magnetometrul tri-axial realizează detectarea direcţiei şi intensităţii câmpului magnetic, parametri returnaţi de acesta fiind de forma expresiei (1.3). În cazul magnetometrului trebuie specificat că generează erori prin bruiatul realizat de componentele magnetice din vecinătatea sa [5], [15].

$$m = \begin{bmatrix} m_x & m_y & m_z \end{bmatrix} \quad (1.3)$$

1.3 Obiectivele lucrării

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

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

1.4 Structura lucrării

1. Introducere

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

2. Arhitectura hardware

Acest capitol expune structura hardware folosită în scopul măsurării parametrilor de accelerometru, giroscop şi magnetometru. Structura hardware este alcătuită dintr-o placă de dezvoltare Arduino Uno detaliată în acest capitol, la care se conectează un pachet de senzori inerţiali 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 parametrilor inerţiali, orientare implementată într-o aplicaţie 3D.

4. Convenţiile orientării

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

5. Filtrarea datelor

Capitolul 5 descrie modalităţile de filtrare a datelor provenite de la senzor, cu scopul filtrării perturbaţiilor şi 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 şi mai rapidă decât filtrarea Kalman.

7. Rezultate experimentale

Utilitatea acestui capitol este de a observa rezultatele acţiunilor produse de implementare a algoritmilor software de citire şi analiză a datelor inerţiale, procesarea acestora şi expunerea modelului orientării.

8. Concluzii şi dezvoltări ulterioare

Capitolul final al proiectului face o concluzionare a studiilor şi rezultatelor obţinute în cadrul proiectului, observându-se o vedere generală a soluţiei de rezolvare a problemei. Pe baza cunoştinţelor dobândite şi analizării necesităţilor şi tendinţelor actuale se face o expunere a viitoarelor dezvoltări ale proiectului.

2. Arhitectura hardware

Placa de dezvoltare Arduino Uno

Pachetul de senzori 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 într-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 înconjurător 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 ieșire PWM);
- pini analogici: 6;
- intensitatea curentului de ieșire: 40 mA;

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

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

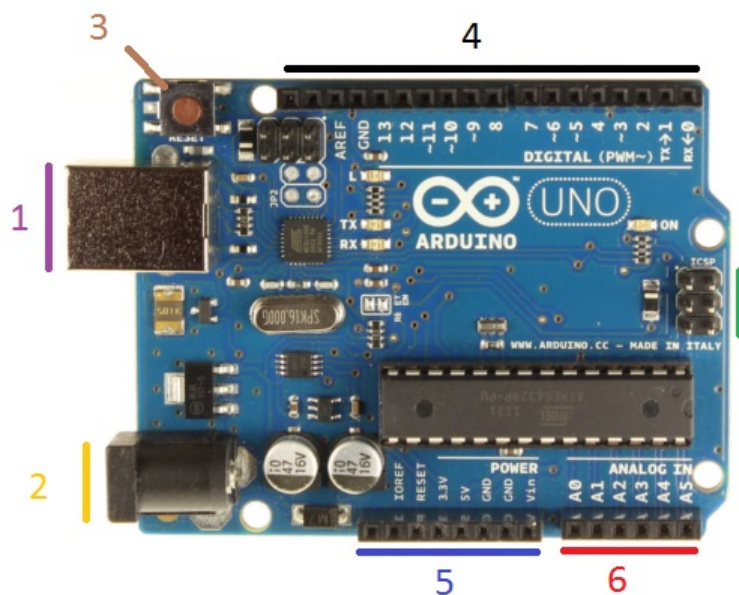


FIGURA 2.1: Explicarea pinilor plăcii Arduino Uno

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

1. Conector de USB de Tip B (Mamă), pentru alimentarea cu 5V şi 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 observaţii:
 - 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 funcţia *void setup()* ;
4. Şir pini digitali, conţine 14 intrări/ieşiri din care 6 pot fi folosite ca ieşiri PWM (3,5,6,9,10,11), o detaliere se face a acstora se face astfel:
 - pinii 0(rx), 1(tx) sunt folosiţi pentru a primi (rx) şi transmite (tx) date. În cazul în care programul foloseşte funcţia de comunicare serială (se poate indentifica în program prin „Serial.begin,„) aceste ieşiri 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, şi stins pentru LOW;
 - GND este conectat la masă.
5. Şir 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. Şir pini analogici: A0...A5. Valoarea acestori pini se poate citi cu funcţia *analogRead()*, din citire rezultă numere între 0-1023 pentru 0V – 5V.

2.2 Pachetul de senzori accelerometru, giroscop şi magnetometru AltIMU-10 v4

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

Specificaţii:

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

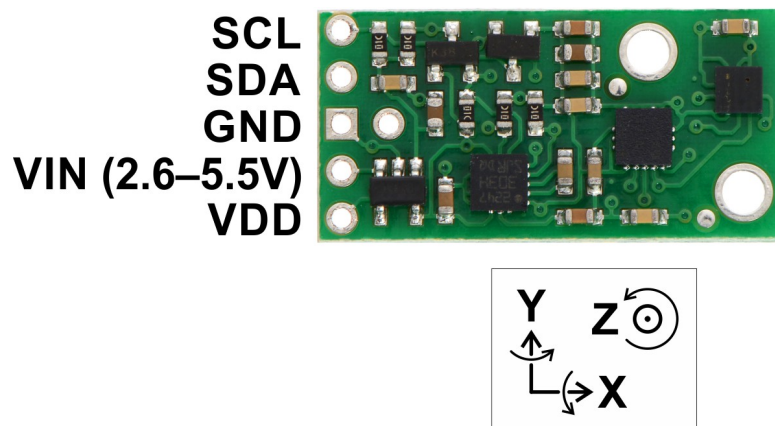


FIGURA 2.2: Altimu-10 v4

- intensitatea curentului de iesire: 6mA;
- formatul ieşirii (I^2C):
 - ieşirea giroscopului: citirea unei axe se face pe 16 biţi;
 - ieşirea accelerometrului: citirea unei axe se face pe 16 biţi;
 - ieşirea magnetometrului: citirea unei axe se face pe 16 biţi;
 - ieşirea barometrului: citirea presiunii se face pe 24 biţi;
- domeniul sensibilităţii:
 - giroscop: $\pm 245, \pm 500$ rad/s;
 - accelerometru: $\pm 2, \pm 4, \pm 6, \pm 8$ g;
 - magnetometru: $\pm 2, \pm 4, \pm 8$ gauss;
 - barometru: 260 mbar- 1260 mbar;
- Conectarea: Pentru folosirea acestui pachet de senzori trebuie folosite cel puţin patru conexiuni legate la pinii: VIN , GND , SCL şi 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 şi SDA trebuie conectaţi 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 creată în anii '80 de către Philips semiconductors. Ea îşi propunea să fie o cale usoară de a conecta microcontroller-ul cu alte circuitele integrate.

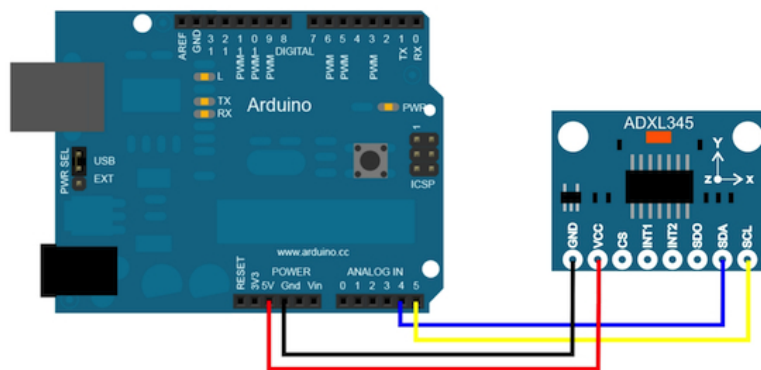


FIGURA 2.3: Conectarea AltIMU-10 v4 la Arduino Uno

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

Controlul magistralei va aparţine (la un moment dat) unui singur dispozitiv Master, care va adresa un singur dispozitiv Slave. Masterul este dispozitivul care va iniţia transferul (printr-o condiţie de START), va genera impulsurile de ceas pe linia SCL şi tot el va încheia transferul generând condiţia de STOP. Viteza de transfer pe magistrală poate fi de 100 kbit/sec sau 400 kbit/sec , iar adresa dispozitivelor I^2C este formată de regulă din 7 biţi [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 ceea 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 planuiește suport și pentru Sony's PlayStation Vita, oferind în acest moment suport pentru Windows Phone și BlackBerry. Peste 1.8 milioane de programatori folosesc Unity, plug-in-ul pentru browser a fost instalat 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 şi co-fondatorul Unity: *"Dorinţa noastră este să oferim aceleaşi instrumente şi 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 unui 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].

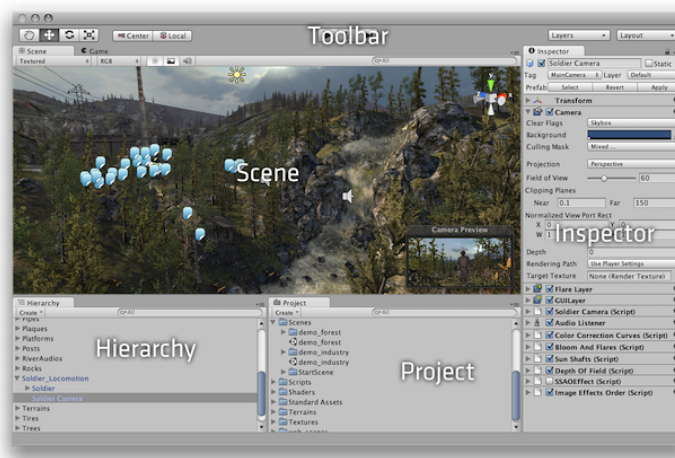


FIGURA 3.5: Editorul Unity

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 comportarea inerțială din lumea reală, parametri precum masa sau gravitația, aceștia influențând 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 atașată unui obiect din joc.

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

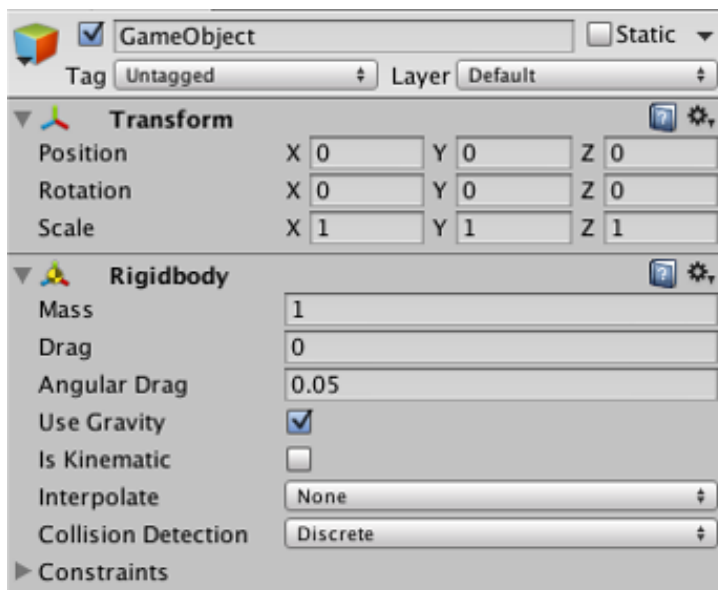


FIGURA 3.6: Uneltele *Transform* şi *Rigidbody*

se programează acţiunile de răspuns la intrările introduse de utilizator. Practic, această funcţie programează atitudinea obiectului, căreia este asociată, pe tot parcursul jocului. A doua funcţie importantă în orice joc, este funcţia *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 () {

    }

}
```

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

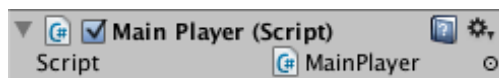


FIGURA 3.7: Confirmarea ataşării scriptului

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 paşii:

- se descarcă aplicaţia de pe pagina producătorului şi se dezarhivează într-un director convenabil;
- se conectează placa Arduino la computer printr-un cablu USB astfel cel puţin un LED ar trebui să se aprindă pe placă;
- se indică locaţia driverului, în general directorul C:/arduino-1.0/drivers;
- se porneşte 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 doreşte 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 şi stinge la anumite intervale un led:

```
1. /*
2.   Blink
3.   Turns on an LED on for one second, then off for one second, repeatedly.
4.   This example code is in the public domain.
5.   */
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 această secţiune. 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 Vizualizarea datelor în ediorul arduino

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

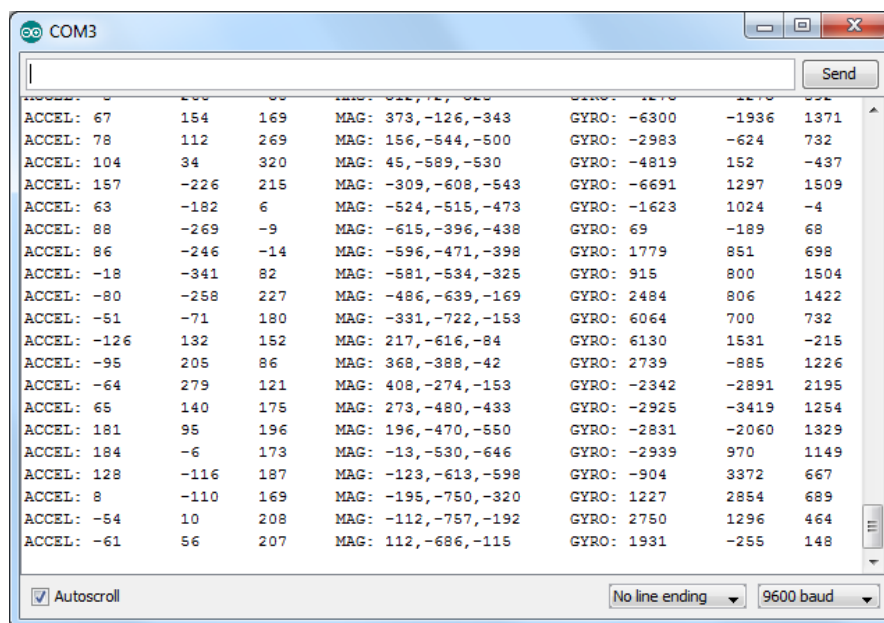


FIGURA 3.8: Datele procesate de arduino şi afişate prin modulul serial

3.3 Platforma Matlab

MATLAB® (MATrix 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 inginerască
- 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.com>] 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, până 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-ului ca pe o maşină de calcul, şi de citire şi scriere de fişiere MAT-files [26].

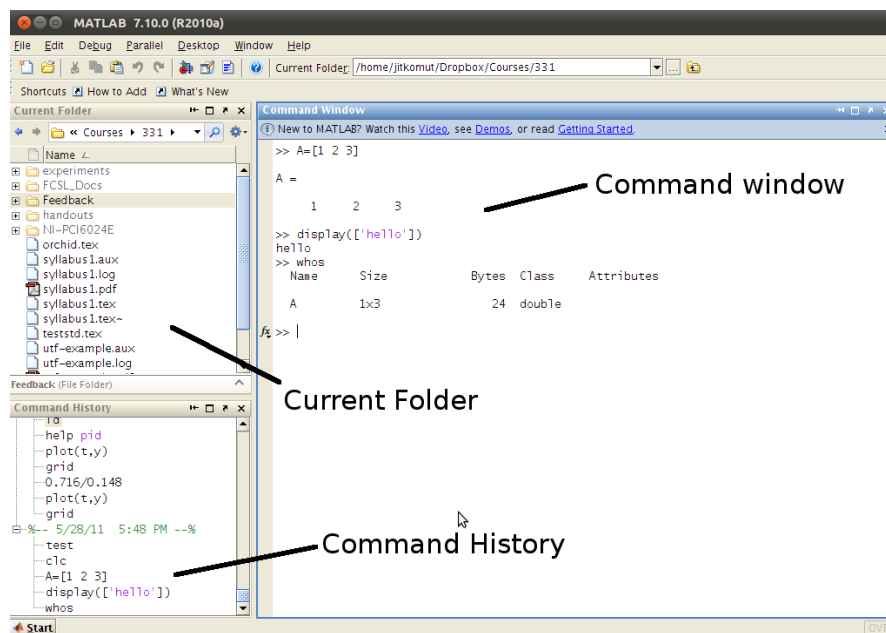


FIGURA 3.9: Editorul Matlab

3.4 Conexiunea dintre Arduino şi calculator

Placa de dezvoltare Arduino prezintă facilităţi gândite pentru utilizatorii care se simt mult mai confortabil programând într-un limbaj de nivel înalt (Java, C#, Python, Ruby, PHP) . Aceste facilităţi sunt realizate de comunicaţia serială , dintre plăcile Arduino şi calculatoarele clasice, prin intermediul căreia datele captate de la senzori şi/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 aplicaţii mult mai complexe, atât la nivel hardware cât şi software. Tot aceeaşi facilitate poate fi adusă prin realizarea unei comunicaţii de tip ethernet, dar această facilitate ridică costurile structurii hardware, şi 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

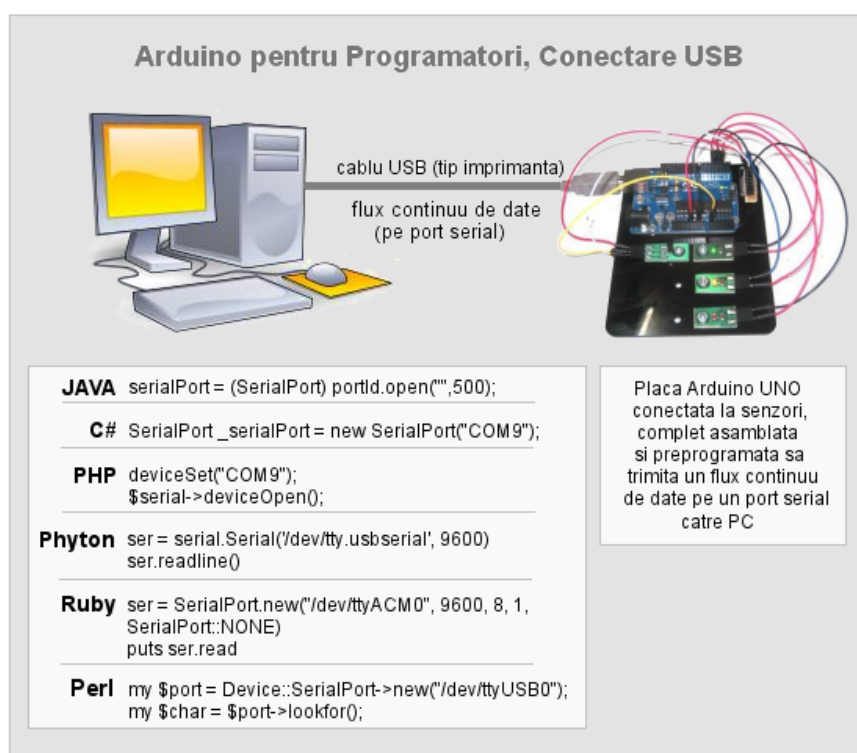


FIGURA 3.10: Conexiunea dintre placa Arduino şi calculator

trebuie făcută citirea datelor şi interpretarea acestora în funcţie de semnificaţia acestora [23].

3.4.1 Conexiunea dintre placa de dezvoltare Arduino şi Unity

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

Această librărie facilitează:

1. Trimiterea şi primirea comenzilor în ambele sensuri.
2. Adăugarea de argumente comenzilor.
3. Generarea de funcţii de reacţie ca răspuns la anumite comenzi primite.
4. Transferul datelor bidirecţional.

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 dezvoltare Arduino şi Matlab

Pentru a facilita transferul dintre Arduino şi platforma Matlab se foloseşte 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 şi, la nevoie, returnează un răspuns.

În imaginea următoare se poate vedea o parte din comenzile folosite în cadrul comunicaţiei seriale [26].

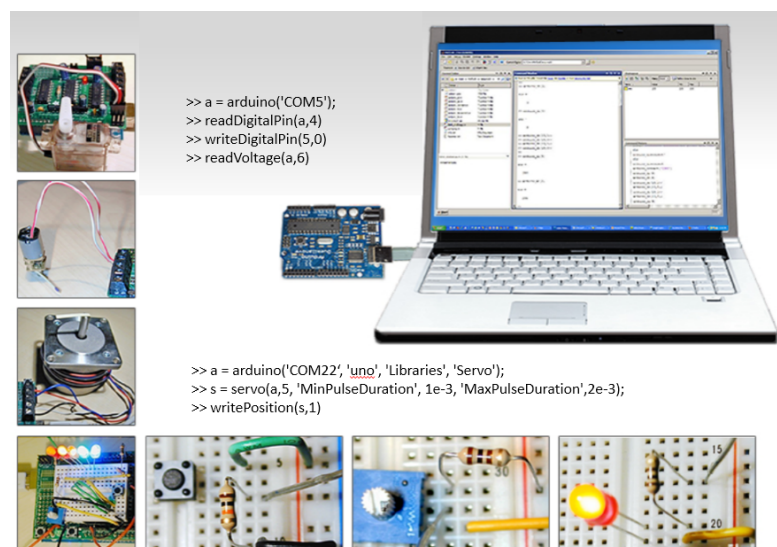


FIGURA 3.11: Conexiunea dintre placa Arduino şi calculator

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 convenției 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 funcție 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 mișcare 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 roti pe planul yz [18], [22].

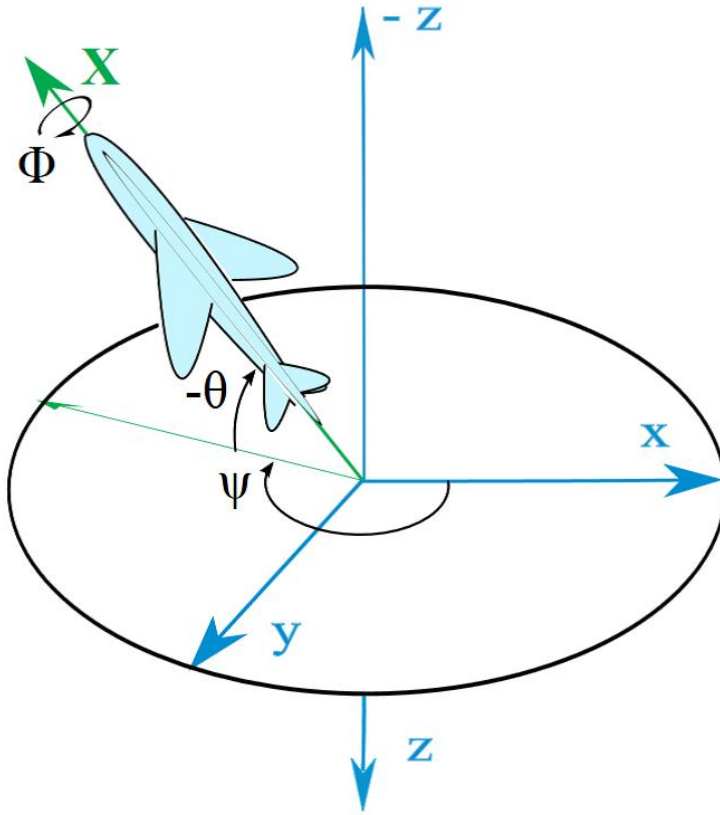


FIGURA 4.12: Reprezentarea unghiurilor Tait-Bryan. Unghiul Ψ corespunde rotaţiei în jurul axei Z . Unghiul Φ corespunde rotaţiei în jurul axei X . Unghiul Θ corespunde rotaţiei în jurul axei Y

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 spaţiu. Rotaţia unui corp cu unghiul Θ în jurul axei A_r din sistemul de coordonate A , dar această rotaţie 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 $\hat{x}_A, \hat{y}_A, \hat{z}_A$ şi vectorii $\hat{x}_B, \hat{y}_B, \hat{z}_B$ reprezintă cele trei axe de coordonate ale sistemelor A respectiv B . În ecuaţia 1.1 cuaternionul \hat{q} ce descrie orientarea corpului, unde r_x, r_y, r_z definesc componentele vectorului ${}^A\hat{r}$ în sistemul de coordonate A .

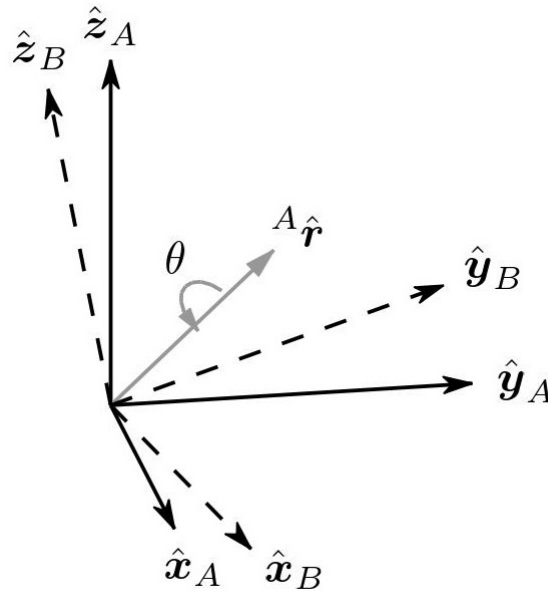


FIGURA 4.13: Orientarea vectorului ${}^A\hat{r}$ ce se roteşte în cu unghiul Θ în sistemul de coordonate A raportat la sistemul de coordonate B

În momentul în care avem o unitate a magnitudinii, cuaternionii pot fi folosiţi pentru a reprezenta rotaţia. Pentru reprezentarea rotaţiilor se folosesc cuaternionii de rotaţie, iar pentru reprezentarea orientării relative la un sistem de referinţă se folosesc cuaternionii de poziţie. Spre deosebire de metoda unghiurilor lui Euler, metoda cuaternionilor poate măsura rotaţiile în sistemul 3D în fiecare oricare situaţie, evitând blocarea gimbal. Această caracteristică a cuaternionilor duce la o reprezentare matematică a orientării mult mai robustă şi fiabilă [12].

4.2.1 Norma unui cuaternion

Norma unui cuaternion este definită ca :

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

Norma cuaternionului este folosită la normalizarea expresiei acestuia după fiecare iteraţie. Expresia după care se realizează acest lucru este [12]:

$$q = \frac{q}{|q|} \quad (4.6)$$

4.2.2 Înmulţirea

Înmulţirea quaternionilor $p \otimes q$ este definită ca o combinaţie secvenţială a două rotaţii. Înmulţirea quaternionilor este o operaţie necomutativă definită ca [12]:

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

unde matricea quaternionilor este definită astfel:

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

iar conjugata acesteia este:

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

4.2.3 Vectorul de rotaţie

Un vector x poate fi rotit faţă de sistemul de referinţă, proces ce e determinat de quaternionul q ce produce expresia vectorului x' folosind expresia [12]:

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

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

$$x' = R(q)x \quad (4.11)$$

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

4.2.4 Derivata unui quaternion

Derivata unui quaternion poate fi făcută cu ajutorul a două ecuaţii, prima ecuaţie ia în considerare viteza unghiulară a vectorului de referinţă în raport cu sistemul de

referinţă fixat, iar a doua ecuaţie analizează viteza unghiulară faţă de sistemul de referinţă al corpului. În acest proiect s-a folosit a doua ecuaţie [12]:

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

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=rotaţia pe axa z, pitch=rotaţia pe axa y, roll=rotaţia pe axa x) [1]:

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

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

$$roll = asin(2 \cdot (q_0 q_2 + q_3 q_1)) \quad (4.16)$$

Cu ajutorul acestor ecuaţii se poate determina şi folosii o expresie a unghiurilor lui Euler neafectată de efectul blocării gimbal, generând performanţe mult mai bune în aplicaţiile 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 $i - 1$ și din modelul dinamic. Modelul dinamic liniar este aplicat estimării anterioare, și se obține valoarea prezisă pentru vectorul de stare curent. Adicional, 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 , și 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 și 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]:

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

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

covarianţă a incertitudinii tranziţiei. Atunci covarianţa predicţiei este:

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

5.1.2 Măsurarea

Proces extern filtrului Kalman, măsurătoarea are ca rezultat unul sau mai mulţi vectori de măsură Y_i^k fiecare cu o matrice de covarianţă R_i^k care codifică eroarea de măsurare estimată (imprecizia senzorului). Relaţia dintre vectorul de stare si vectorul de măsură este modelul de măsurare, iar dacă acesta este o transformare liniară aceasta este descrisă de matricea H_i . Folosind acest model, obţinem predicţia măsurătorii, \bar{Y}_i [17].

$$\bar{Y}_i = H_i \bar{X}_i \quad (5.19)$$

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

$$S_i = H_i P_i H_i^T + R_i \quad (5.20)$$

Matricea S_i nu este matricea de covarianţă a măsurătorii prezise, ci matricea de covarianţă a diferenţei dintre măsurătoarea prezisă si o posibilă măsurătoare reală (matricea de covarianţă a *inovării*, sau a *rezidualului*) – adică exact matricea necesară pentru a defini o zonă de căutare în jurul măsurătorii prezise \bar{Y}_i .

5.1.3 Asocierea datelor

Filtrul Kalman este foarte vulnerabil la asocierea datelor (asocierea măsurătorilor), din cauza naturii unimodale a funcţiei de probabilitate în fiecare din fazele de lucru ale filtrului. Acest lucru înseamnă că vectorul de stare, odată deplasat spre un indiciu fals, va deveni criteriul de selecţie pentru măsurătorile viitoare, asta însemnând mai multe date greşite incluse în estimare, până la devierea totală de la ţintă.

Singura măsură obiectivă a utilităţii măsurătorii Y_i este verosimilitatea ei, dându-se starea unui obiect urmărit. Această verosimilitate este dată de funcţia de probabilitate Gaussiană, centrată în predicţia măsurătorii \bar{Y}_i , si având matricea de covarianţă S_i [17].

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

5.1.4 Corecţie

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

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

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

Ecuţia este echivalentă cu

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

Pasul final este calcularea matricei de covarianţă P_i . Majoritatea documentaţiilor disponibile dau ecuaţia următoare pentru calculul P_i .

$$P_i = (I - K_i H_i) \bar{P}_i \quad (5.24)$$

5.1.5 Vectorul de stare

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

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

5.1.6 Predicţia modelului şi măsurătorile modelului orientării

Expresiile *modelului predicţie* şi *modelului măsurătorilor* pot fi exprimate prin ecuaţiile matriceale [1], [12]:

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

5.2 Dezavantajele folosirii filtrului Kalman liniar

Implementarea unui filtru Kalman linear are o serie de dezavantaje. Primul dezavantaj se întâlneşte în momentul rotirii unei axe iar concomitent altă axă este deviată de la orizontală, unghiul trigonometric estimat de accelerometru şi corespondenta viteză unghiulară măsurată de giroscop nu vor mai fi în acelaşi 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 distanţă mare faţă de orizontală rezultă o sursă importantă de erori.

Pentru a rezolva aceste dezavantaje se foloseşte implementarea cuaternionilor pe baza comportării modelului orientării. Acest lucru impune folosirea predicţiei neliniarităţi 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 foloseşte matricea Jacobian pentru predicţie, iar funcţiile măsurătorilor pentru a liniariza estimarea stării curente şi a covarianţei. Această implementare permite unui nou set de predicţi şi de măsurători, pentru a fi folosite în cadrul estimării.

5.3.1 Vectorul de stare

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

$$x = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 & \dot{q}_3 & w_{xb} & w_{yb} & w_{zb} \end{bmatrix}^T \quad (5.27)$$

La momentul iniţial vectorul de stare este compus din poziţia iniţială $q = [1 \ 0 \ 0 \ 0]$ şi din eroarea viteze unghiulare care este necunoscută iniţial.

$$x = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \quad (5.28)$$

5.3.2 Formularea modelului predicţiei

În această aplicaţie filtrul Kalman extins foloseşte un model al predicţiei ce este reprezentat printr-o funcţie ce descrie următoarea orientare estimată pe baza precedentei

estimări şi a vectorului vitezei unghiulare măsurate de giroscop.

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

unde:

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

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

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

Expresia Jacobianului poate fi calculată prin derivarea funcţiei f în funcţie de poziţie, rezultând matricea F

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

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 funcţie de orientarea estimată, astfel $h(x_k)$. Această funcţie poate fi folosită pentru a mapa starea estimată pe vectorul măsurătorilor z , practic se realizează o comparare a predicţiei cu măsurătorile reale. În acest caz măsurătorile realizate ce implică parametrii accelerometrului şi ai magnetometrului ce vor fi folosiţi pentru

predicţia curentă. Vectorul măsurărilor poate fi definit astfel [8]:

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

5.3.3.1 Maparea parametrilor accelerometrului

Vectorul gravitaţiei din sistemul fixat este rotit în cadrul sistemului de coordonate al corpului, fapt reprezentat prin cuaternionul q , mapat în cadrul vectorului accelerometrului. Această aplicaţie ţine cont doar de direcţia vectorului gravitaţiei şi nu de cea a câmpului magnetic, vectorul este normalizat înainte de operarea asupra acestuia.

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

5.3.3.2 Maparea parametrilor magnetometrului

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

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

Folosirea acestui model de citire a valorilor magnetometrului influenţează estimările expresiilor rotaţiilor în jurul axelor $x(roll)$ şi $y(pitch)$ determinând erori ale orientării. De asemenea este nevoie de specificarea direcţiei câmpului magnetic al pământului, care variază considerabil în jurul pământului. Acastă problemă a fost rezolvată folosind soluţia prezentată de S.Madgwick [1], unde direcţia câmpului magnetic este calculată pe baza aceiaşi 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, astfel rezultând expresia m' [12]

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

Noul câmp de referinţă e calculat în scopul obţinerii aceiaşi înclinări în cadrul vectorului măsurătorilor.

$$b = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} \sqrt{m_x'^2 + m_y'^2} \\ 0 \\ m_z' \end{bmatrix} \quad (5.37)$$

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

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

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

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

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

5.3.4 Matricele de covarianţă

Rezultatele filtrului Kalman estins sunt afectate şi de specificaţiile matrielor de covarianţă Q şi R . Acestea sunt matrice diagonală reprezentând variaţia aşteptată a zgomotului în faza predicţiei şi în vectorul măsurătorilor z . În această aplicaţie matricele

de covarianţă au fost alese astfel:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.2 \end{bmatrix} \quad (5.41)$$

$$R = \begin{bmatrix} 1 \cdot e6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 \cdot e6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 \cdot e6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \cdot e6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \cdot e6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \cdot e6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \cdot e6 \end{bmatrix} \quad (5.42)$$

5.3.5 Ecuaţiile filtrului Kalman extins

Ecuaţiile filtrului Kalman extins se pot clasifica în două secţiuni specifice etapelor filtrării.

Ecuaţiile predicţiei

Predicţia orinetării: $x_i = f(x_{i-1}, u_i)$

Covarianţa predicţiei: $P = FPF' + Q$

Ecuaţiile corecţiei

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

Covarianţa măsurătorii: $S = HPH' + R$

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

Corecţia orinetării prezise: $x_i = x_i + Ky$

Corecţia covarianţei prezise: $P = (I - KH)P$

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

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

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

6. Determinarea orientării pe baza filtrului AHRS

Filtrul AHRS

Orientarea pe baza vitezei unghiulare

Orientarea pe baza vectorului măsurătorilor

Fuziunea celor două metode de determinare 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. Obiectivul acestui proiect este de a transforma aceste date brute în reprezentări ce pot determina 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 bazate pe metodele unghiurilor lui Euler, matricilor de rotație, cuaternionilor sau filtrului Kalman. Aceste prezintă o complexitate de operații matematice ce diminuează viteza de execuție, dar au și o 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

(Altitude and Heading Reference System) dezvoltat de Mahony şi perfecţionat de Madgwick. Un filtru ce este mult mai rapid şi mult mai simplu decât filtrul Kalman, prezentând performanţe asemănătoare poate chiar mai bune.

6.2 Principiile folosite în filtrul AHRS

Implementarea realizată de Madgwick prezintă două versiuni de filtru AHRS. Prima implementare se adresa dispozitivelor IMU generale ce e compus dintr-un accelerometru şi giroscop tri-axial. A doua implementare se referă asupra senzorilor MARG (Magnetic, Angular Rate, and Gravity), senzori ce aduc o îmbunătăţire faţă de senzorii IMU, având în componenţă un magnetometru tri-axial. Această implementare încorporează componenta magnetică şi 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 şi z ale sistemului de referinţă al senzorului, vitezele unghiulare se notează w_x , w_y şi w_z . Aceşti parametrii sunt distribuiţi într-un vector definit prin ecuaţia(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 ecuaţia (6.45)[3].

$$S = \begin{bmatrix} 0 & w_x & w_y & w_z \end{bmatrix} \quad (6.44)$$

$${}^S_E \dot{q} = \frac{1}{2} {}^S_E \hat{q} \otimes S_w \quad (6.45)$$

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 ecuaţiile (6.46) şi (6.47) furnizate pentru condiţiile iniţiale ştiute. În această ecuaţie, S_{wt} este viteza unghiulară măsurată la timpul t , Δt este perioada de eşantionare, iar ${}^S_E \hat{q}_{est,t-1}$ este estimarea precedentă a eorientării. Indicele w indică calcularea cuaternionului în funcţie de viteza unghiulară[1].

$${}^S_E \dot{q}_{w,t} = \frac{1}{2} {}^S_E \hat{q}_{est,t-1} \otimes S_{w,t} \quad (6.46)$$

$${}^S_E q_{w,t} = {}^S_E \hat{q}_{est,t-1} + {}^S_E \dot{q}_{w,t} \Delta t \quad (6.47)$$

6.2.2 Orientarea pe baza vectorului măsurătorilor

Accelerometrul tri-axial va măsura mărimea şi direcţia câmpului gravitaţiei în sistemul de coordonate al senzorului, măsurători afectate de acceleraţiile liniare determinate de mişcarea senzorului. Similar magnetometrul tri-axial va măsura mărimea şi direcţia câmpului magnetic al pământului în sistemul de coordonate al senzorului, afectat de câmpul magnetic local şi de perturbaţii. În contextul implementării unui filtru al orientării, se va admite ca accelerometrul va măsura doar gravitaţia, iar magnetometrul va măsura doar câmpul magnetic al pământului[1].

Dacă direcţia câmpului pământului este cunoscută în sistemul de coordonate terestru, măsurarea direcţiei câmpului în cadrul sistemului 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 soluţie unică a orientării senzorului, existând o infinitate de soluţii reprezentate de rotaţiile ce se pot realiza pe baza orientării în jurul unei axe paralele cu câmpul. În unele aplicaţii se folosesc pentru determinarea orientării unghiurile lui Euler, dar acestea generează o soluţie incompletă, generând o soluţie în care se cunosc două unghiuri, dar al treilea este necunoscut, unghiul necunoscut fiind unghiul în jurul axei paralele cu câmpul magnetic al pământului. Această problemă a fost dezbătută în capitolul 4.2.5, şi se numeşte blocarea gimbal [13]. O soluţie pentru rezolvarea completă a acestei probleme este dată de folosirea quaternionilor, detaliaţi în capitolul 4.2.5. Generând formularea optimizării problemei, se consideră o orientare a senzorului, ${}^S_E \hat{q}$, care se aliniază cu o direcţie predefinită a câmpului în sistemul de coordonate al pământului ${}^E \hat{d}$, cu direcţia măsurată a câmpului în sistemul senzorului, ${}^S \hat{s}$, folosind operaţia de rotaţie. Astfel ${}^S_E \hat{q}$ e posibil să găsească o soluţie, unde ecuaţia (6.48) este funcţia obiectiv, iar componenta fiecărui vector e definită în ecuaţiile (6.49),(6.50) şi (6.51)[1].

$$f({}^S_E \hat{q}, {}^E \hat{d}, {}^S \hat{s}) = {}^S_E \hat{q}^* \otimes {}^E \hat{d} \otimes {}^S_E \hat{q} - {}^S \hat{s} \quad (6.48)$$

$${}^S_E \hat{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix} \quad (6.49)$$

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

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

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

cele mai simple de implementat şi calculat. Ecuaţia (6.52) descrie algoritmul gradientului pentru n iteraţii rezultând orientarea estimată ${}^S_E q_{n+1}$ bazată pe orientarea iniţială ${}^S_E q_0$ şi un pas μ . Ecuaţia (6.53) calculează soluţia definită prin funcţia obiect şi funcţia Jacobian, simplificate sub forma unor matrice coloană (6.54) respectiv o matrice 3x4 (6.55)

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

$$\nabla f({}^S_E \hat{q}, {}^E \hat{d}, {}^S \hat{s}) = J^T({}^S_E \hat{q}, {}^E \hat{d}) f({}^S_E \hat{q}, {}^E \hat{d}, {}^S \hat{s}) \quad (6.53)$$

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

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

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

Ecuaţiile din intervalul (6.52)-(6.55) descriu forma generală a algoritmului aplicabilă unui câmp predefinit în orice direcţie. Oricum, dacă direcţia câmpului poate fi determinată doar cunoscând componentele de pe una sau două axe principale ale sistemului de coordonate global, astfel ecuaţiile se simplifică considerabil. Prin convenţie se ştie că direcţia gravitaţiei defineşte verticala, axa z precum e definită în ecuaţia (6.56). Substituind ${}^E \hat{g}$ cu vectorul accelerometrului normalizat ${}^S \hat{a}$ pentru ${}^E \hat{d}$ şi pentru ${}^S \hat{s}$, ecuaţiile (6.54) şi (6.55) devin ecuaţiile (6.58) şi (6.59)[1].

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

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

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

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

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

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

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

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

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

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

Pentru optimizarea procesului de obţinere a orientării senzorului este nevoie de apelarea în mai multe iteraţii a ecuaţiei (6.52), pentru a calcula orientarea senzorului la fiecare măsurătoare făcută de senzor. Pentru eficientizarea algoritmului este nevoie şi de ajustarea rata de eşantionare μ la fiecare iteraţie, în general valoarea optimă cu care este ajustat μ este calculat prin dericvarea funcţiei f . Această operaţie prouce un volm ridicat de calcule ce va încetini procesul, fără a aduce îmbunătăţiri considerabile. În cazul acestei aplicaţii este suficient să se calculeze o iteraţie la momentul de timp stabilit de convergenţa reglementată de $\mu_t \geq$ rata de schimbare a orientării. Ecuaţia (6.64) calculează orientarea estimată ${}^S_E\hat{q}_{\nabla,t}$ la momentul de timp t bazată pe estimarea precedentă ${}^S_E\hat{q}_{est,t-1}$, şi de gradientul funcţiei obiectiv f , ∇f definit de măsurătorile senzorului ${}^S\hat{a}_t$

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

$${}^S_E q_{\nabla,t} = {}^S_E \hat{q}_{est,t-1} - \mu_t \frac{\nabla f}{\|\nabla f\|} \quad (6.64)$$

Valoarea optimă a rata de eşantionare μ_t poate fi definită pe baza convergenţa orientării estimate ${}^S_E q_{\nabla,t}$, evitându-se depăşirile sau perioadele de eşantionare cu durate inefficient de îndelungate. Astfel μ_t se calculează cu expresia (6.65), unde Δ_t este perioada de eşantionare, ${}^S_E \dot{q}_{w,t}$ este rata orientării măsurată de giroscop, iar α este augumenatrea μ în concordanţă cu zgomotul impus de măsurătorile accelerometrului şi magnetometrului[1].

$$\mu_t = \alpha \|{}^S_E \dot{q}_{w,t}\| \Delta_t, \alpha > 1 \quad (6.65)$$

6.2.3 Fuziunea celor două metode de determinare a orientării

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

$${}^S_E q_{est,t} = \gamma_t {}^S_E q_{\nabla,t} - (1 - \gamma_t) {}^S_E q_{w,t}, 0 \leq \gamma_t \leq 1 \quad (6.66)$$

Valoarea optimă a γ_t se poate obţine prin condiţia ca divergenţa ${}^S_E q_{w,t}$ este egală cu covarianţa ponderată a ${}^S_E q_{\nabla,t}$, condiţie exprimată în ecuaţia (6.67), unde $\frac{\mu_t}{\Delta_t}$ este rata covarianţei ${}^S_E q_{\nabla,t}$, iar β este divergenţa ${}^S_E q_{w,t}$ exprimată prin valoarea derivatei cuaternionului corespondent erorii de măsurare a giroscopului.

$$\gamma_t = \frac{\beta}{\frac{\mu_t}{\Delta_t} + \beta} \quad (6.67)$$

Ecuaţiile (6.66) şi (6.67) asigură fuziunea optimă între ${}^S_E q_{w,t}$ şi ${}^S_E q_{\nabla,t}$, asumând că rata covarianţei ${}^S_E q_{\nabla,t}$ este dependentă de α , adică de zgomot. Încăzul în care zgomotul este foarte mare atunci şi augumentarea μ definită de expresia (6.65) va creşte direct proporţional. O valoare mare a augumentării va face ca termenul ${}^S_E \hat{q}_{est,t-1}$ folosit în ecuaţia (6.64) să devină neglijabil, rezulând ecuaţiile (6.68) (6.69)

$${}^S_E q_{\nabla,t} \approx -\mu_t \frac{\nabla f}{\|\nabla f\|} \quad (6.68)$$

$$\gamma_t \approx \frac{\beta \Delta_t}{\mu_t} \quad (6.69)$$

Înlocuind ecuaţiile (6.47), (6.68) şi (6.69) în ecuaţia (6.66) ecuaţia finală devine (6.70)

$${}^S_E q_{est,t} = \frac{\beta \Delta_t}{\mu_t} \left(-\mu_t \frac{\nabla f}{\|\nabla f\|} \right) + (1 - 0)({}^S_E \hat{q}_{est,t-1} + {}^S_E \dot{q}_{w,t} \Delta_t) \quad (6.70)$$

Ecuaţia (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 ecuaţia (6.72) şi ${}^S_E \hat{q}_{E,t}$ reprezintă direcţia erorii ${}^S_E \dot{q}_{est,t}$ definită în ecuaţia (6.73)

$${}^S_E q_{est,t} = {}^S_E \hat{q}_{est,t-1} + {}^S_E \dot{q}_{est,t} \Delta_t \quad (6.71)$$

$${}^S_E \dot{q}_{est,t} = {}^S_E \dot{q}_{w,t} - \beta {}^S_E \hat{q}_{E,t} \quad (6.72)$$

$${}^S_E \hat{q}_{E,t} = \frac{\nabla f}{\|\nabla f\|} \quad (6.73)$$

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

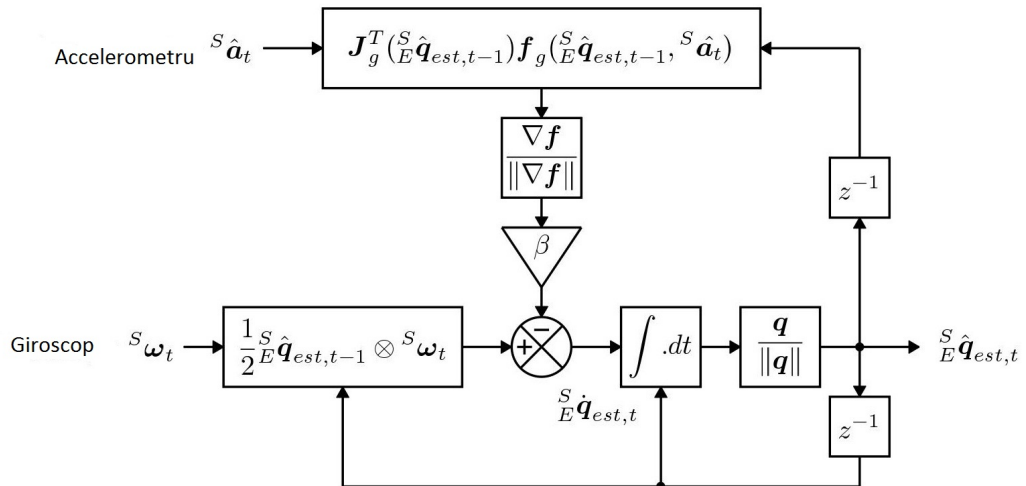


FIGURA 6.14: Schema bloc de determinare a orientării pe baza fuzionării celor două metode

6.2.4 Compensarea deformării magnetice

Măsurătorile câmpului magnetic al pământului vor fi afectate de prezenţa materialelor feromagnetice din apropierea magnetometrului. Investigând efectele distorsiunilor generate de componenta magnetica în determinarea orientării, s-a observat că există numeroase erori incluse de interferenţele introduse de componentele metalice [15][1].

Surse interferatoare ce sunt fixe în cadrul sistemului de coordonate al senzorului pot fi eliminate prin calibrarea senzorilor [16] [10]. Aceste surse generatoare de interferenţe pot genera erori de determinare a înclinării planului orizontal relativ la suprafaţa pământului, erori ce nu pot fi corectate decât raportând la o poziţie de referinţă. Erorile de înclinare ale planului vertical cu pământul pot fi compensate cu ajutorul accelerometru-ului ce generează poziţia senzorului.

Direcţia 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, ${}^S\hat{q}_{est,t-1}$ descrisă de ecuaţia (6.74). Această eroare poate fi eliminată dacă direcţia de câmpului magnetic de referinţă terestru ${}^E\hat{b}_t$ este egală cu înclinarea (6.75).

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

$${}^E\hat{b}_t = \begin{bmatrix} 0 & \sqrt{h_x^2 + h_y^2} & 0 & h_z \end{bmatrix} \quad (6.75)$$

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

6.2.5 Controlul implicării erorii furnizate de giroscop

Obiectivul acestei sarcini a filtrului AHRS este acela ca valoarea erorii staţionare furnizate de giroscop să fie zero. Orice implementare practică cu senzori IMU sau MARG trebuie să țină cont de această condiţie, fiind foarte importantă în cadrul funcţionării. Avantajul folosirii bazelor filtrului Kalman face ca modelul AHRS să fie capabil să estimeze eroarea giroscopului, ca o stare adiţională 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 schimbare a orientării, ${}^S\hat{q}_E$, va fi exprimată ca eroarea unghiulară pe fiecare axă a giroscopului (6.76). Implicarea erorii furnizate de giroscop ${}^S w_b$, este influenţată 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 ecuaţiile (6.77) şi (6.78)[1].

$${}^S w_{E,t} = 2 {}^S\hat{q}_{est,t-1} \otimes {}^S\hat{q}_{E,t} \quad (6.76)$$

$${}^S w_{b,t} = \zeta \sum_t {}^S w_{E,t} \Delta_t \quad (6.77)$$

$${}^S w_{c,t} = {}^S w_t - {}^S w_{b,t} \quad (6.78)$$

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

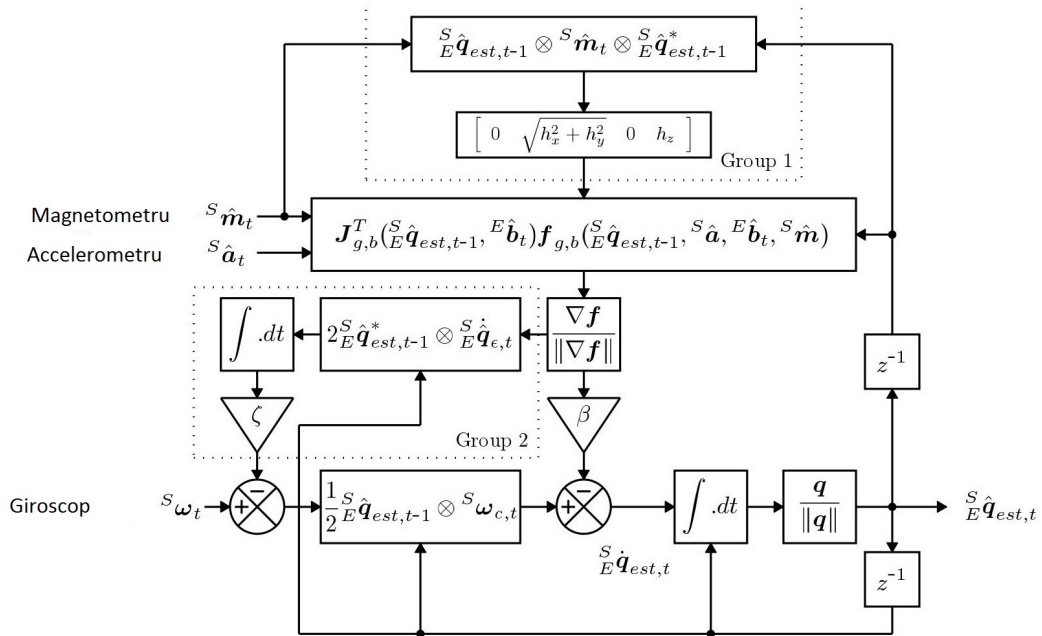


FIGURA 6.15: Schema bloc de determinare a orientării pe baza fuzionării celor două metode, compensarea distorsiunilor magnetice(Group1) şi compensarea erorii giroscopului (Group2).

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 parametrilor 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 complexitatea 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 reprezentare a datelor 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 subcapitolul 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 filtrare și modelare a parametrilor 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 inerţiali

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

7.1.1 Analiza parametrilor în regim static

În cadrul acestei secţiuni se observă comportările intrărilor sistemului în cazul în care senzorul nu este deviat din poziţia iniţială. Astfel rezultă următoarele caracteristici ale datelor giroscopului:

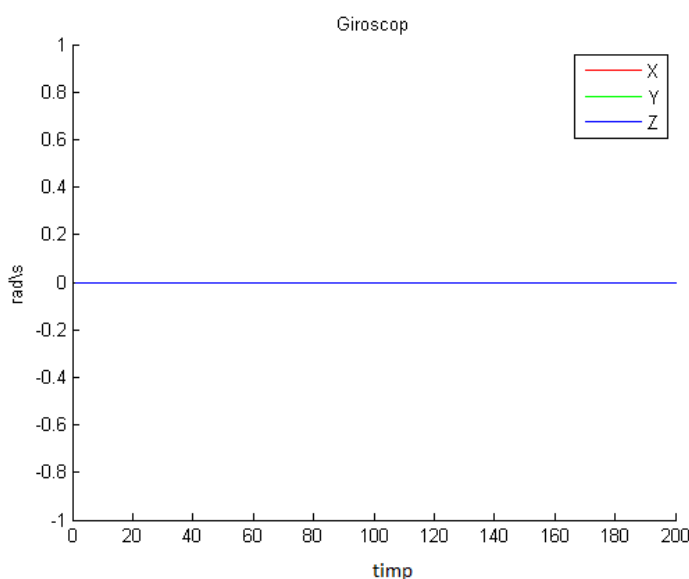


FIGURA 7.16: Parametrii giroscopului în regim staţionar

În această reprezentare a datelor furnizate de giroscop se observă că viteza unghiulară are valoarea zero pe toate cele trei direcţii spaţiale, giroscopul ne fiind scos din starea iniţială.

Aceste caracteristici nu mai sunt valabile pentru valorile accelerometruului care generează o repartiţie a semnalului vizibilă în figura 7.17. În această figură se observă că se adevăresc principiile enunţate în capitolul 1, cu privire cărora acceleraţia este afectată de zgomote. Principala influenţă asupra parametrilor acceleraţiei e adusă de influenţa forţei gravitaţionale exercitate de Pământ. Precum sa enunţat pe parcursul lucrării se observă că cea mai importantă influenţă a gravitaţiei este asupra datelor generate de acceleroimetru pe axa z , paralelă cu gravitaţia. Pe această direcţie senzorul detectează a acceleraţie constantă, dar diferită cu mult de zero, valoarea reală a acesteia. Prin această comportare se poate observa că, perturbaţia instalată de gravitaţie asupra axei

paralele pe direcţia acesteia, trebuie compensată în scopul obţinerii unei comportări reale.

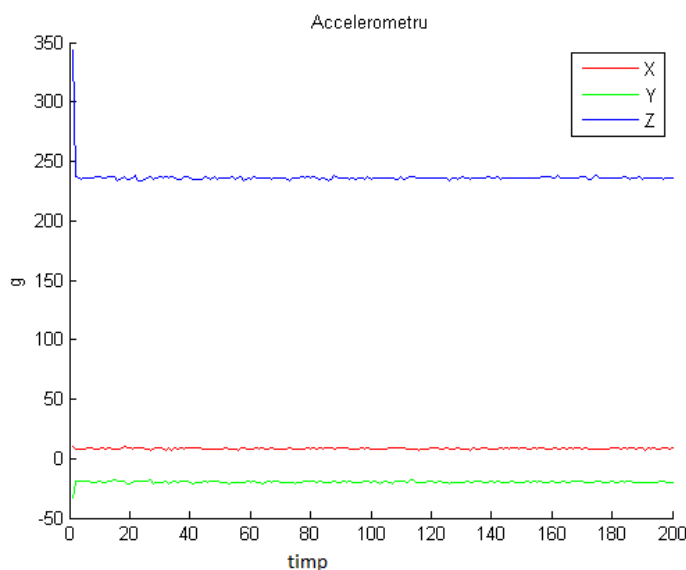


FIGURA 7.17: Parametrii accelerometrului în regim staţionar

7.1.2 Analiza datelor de între în regim dinamic

În această secţiune se încearcă observarea distribuţiei parametrilor accelerometrului şi giroscopului în funcţie de orientarea senzorului. Această evaluare este o evaluare pur subiectivă, pe baza repartiţiei 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 rotaţiilor realizat de senzor.

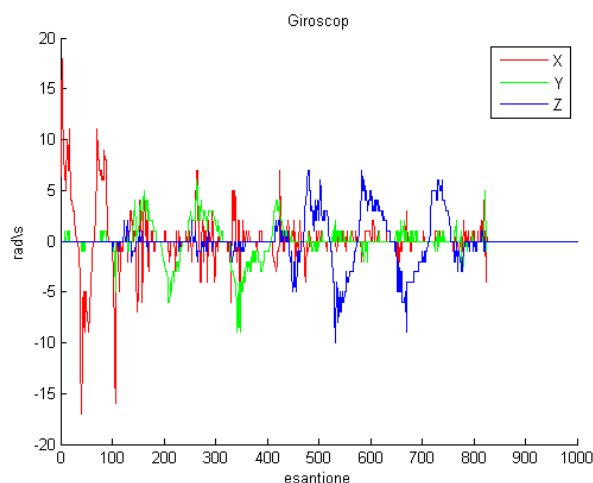


FIGURA 7.18: Datele furnizate de giroscop în regim dinamic

Pe baza imaginii anterioare se poate observa, în diferite secţiuni, o rotaţie accentuată. Astfel în prima secţiune se observă valori ridicate ale vitezei unghiulare realizate pe axa x , putând presupune că senzorul realizează o rotaţie în jurul axei x . În a doua secţiune a imaginii se poate observa o fluctuaţie a valorilor semnalului pe axa y , putând presupune ca sunt rezultate ale rotaţie în jurul axei z . Ultima secţiune a imaginii conţine fluctuaţii importante ale semnalului furnizat de axa z a giroscopului, comporate ce poate determina o rotaţie 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 şi orientativă.

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

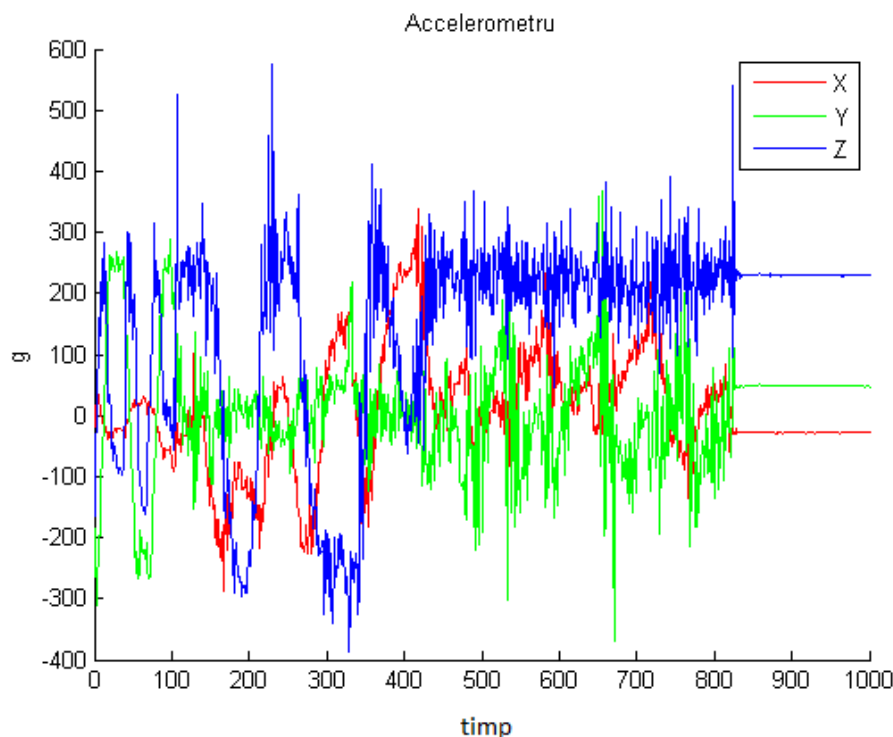


FIGURA 7.19: Datele furnizate de accelerometru în regim dinamic

Din repartitia semnalelor realizată anterior nu se poate determina un model al acceleraţiei, dar se observă fluctuaţii importante ale valorilor acceleraţiei pe axa z , valori afectate de acceleraţia gravitaţională. În funcţie de rotaţiile realizate de corp se poate observa că la un moment dat şi celealte axe ale sistemului sunt paralele cu vectorul acceleraţiei gravitaţionale, putându-se presupune existenţa unor perturbaţii ridicate ce afectează acceleraţiile de pe axele x respectiv z .

7.2 Rezultatele algoritmilor de determinare a orientării

Presupunerile anterioare asupra unei posibile orientări a corpului în spaţiul 3D pe baza fluctuaţiilor parametrilor giroscopului reprezintă ipoteze nesigure şi imprecise. În scopul formulării unei expresii precise şi sigure a orientării unui corp sau elaborat o serie de algoritmi matematici ce determină, pe baza parametrilor inerţiali, o formulare veridică a orientării în spaţiul tri-dimensional.

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

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

Reprezentarea sub forma unghiurilor lui Euler a fost detaliată în subcapitolul 4.1, enunţând rotaţiile realizate în funcţie de cele trei axe, în care se reprezintă rotaţia în sensul acelor de ceasornic funcţie de axa z (yaw), rotaţia în funcţie de axa y (pitch), rotaţia în funcţie de axa x (roll).

Determinarea orientării pe baza unghiurilor lui Euler generează rezultate bune fără un volum mare de calcul, dar principala problemă a acestei reprezentări este generată de defectul principal al acestei metode, acesta este principiul blocării gimbal este reprezentat de cazul în care două inele axiale se plasează pe acelaşi plan, moment în care sistemul 3D se transformă 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 şi filtrul Kalman extins detaliate în capitolul 6, ce se bazează pe descrierea spaţiului tri-dimensional printr-un vector al cuaternionilor dezbătut în secţiunea 4.2. Această reprezentare a spaţiului 3D sub forma de cuaternioni elimină blocarea gimbal, îmbunătăţind calitatea determinării orientării.

Filtrul AHRS dezvoltat în capitolul 7 generează o calculare a orientării, implicând şi componenta câmpului magnetic al pământului, compensând foarte bine erorile de măsurare ale giroscopului şi reuşind să elimine proceseze componenta acceleraţiei gravitaţionale în scopul îmbunătăţirii performanţelor.

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, realizând 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 acelaşi grafic, astfel putându-se observa diferenţele acestora.

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

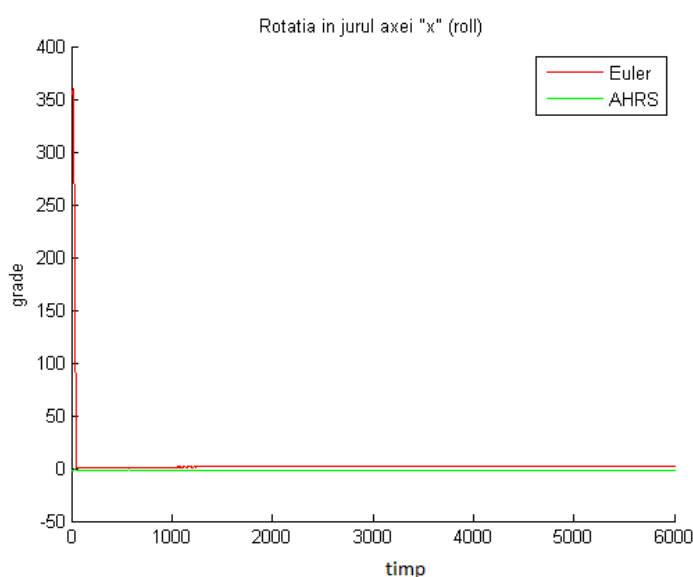


FIGURA 7.20: Reprezentarea unghiului în raport cu axa x

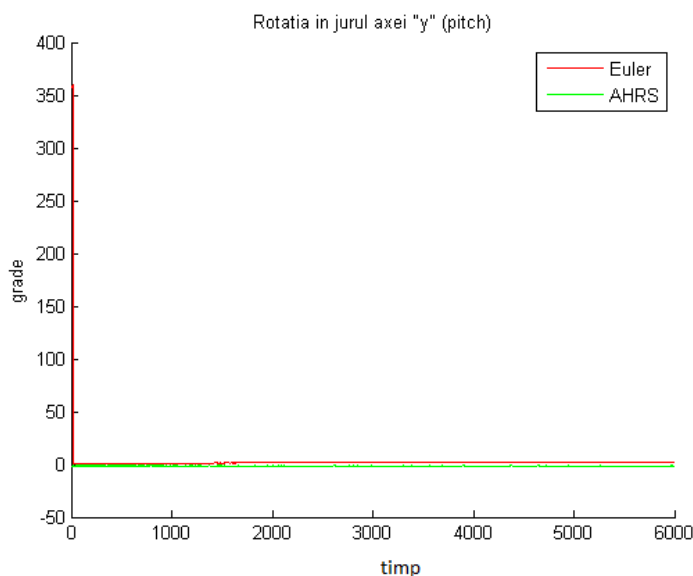


FIGURA 7.21: Reprezentarea unghiului în raport cu axa y

În imaginile precedente se poate observa că, după calibrare, cele două metode au o comportare asemănătoare, observându-se că senzorul are o poziţie apropiată de zero grade în raport cu axa x , şi 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:

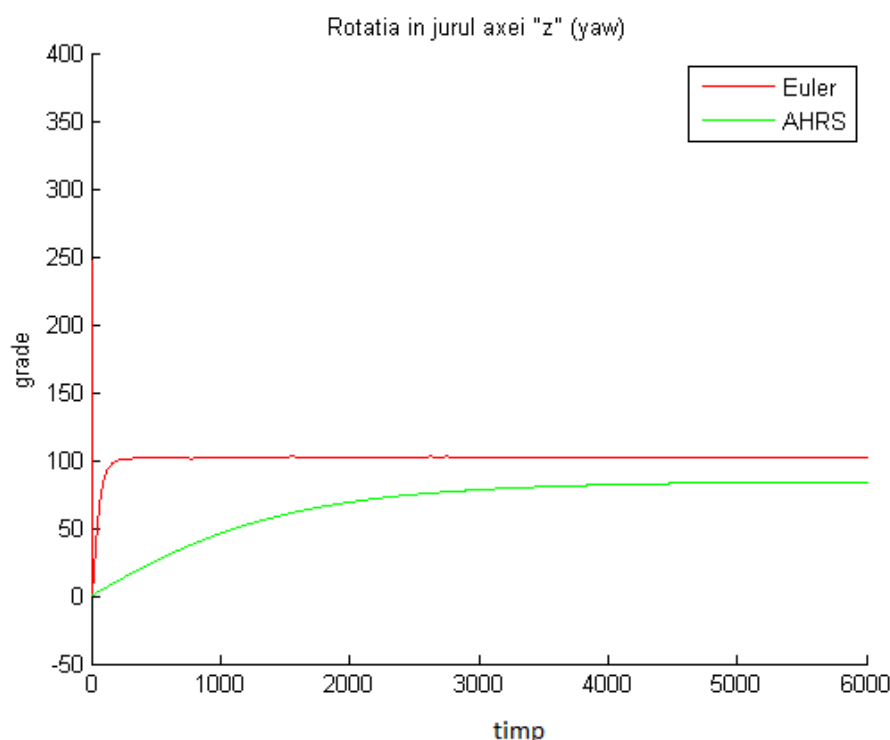


FIGURA 7.22: Reprezentarea unghiului în raport cu axa z

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 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 fluctuațiile rapide.

Analizând orientările detectate de filtrul AHRS se poate observa că acesta detectează fluctuații mult mai dese ale mișcărilor 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:

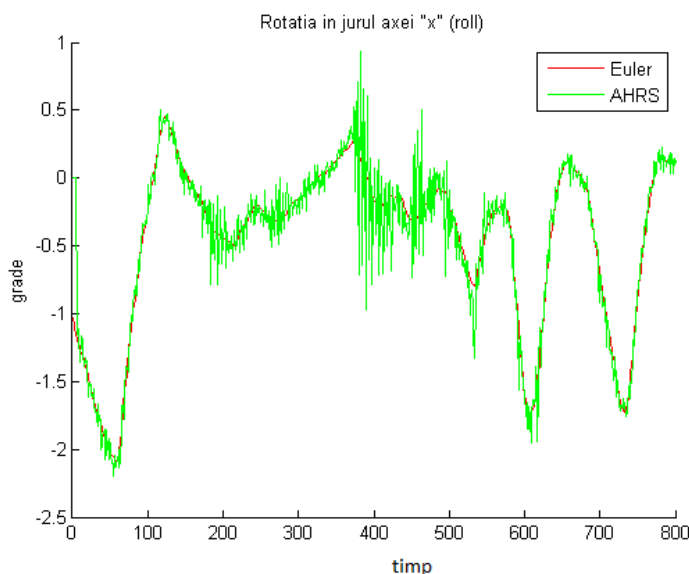


FIGURA 7.23: Reprezentarea orientării în raport cu axa x

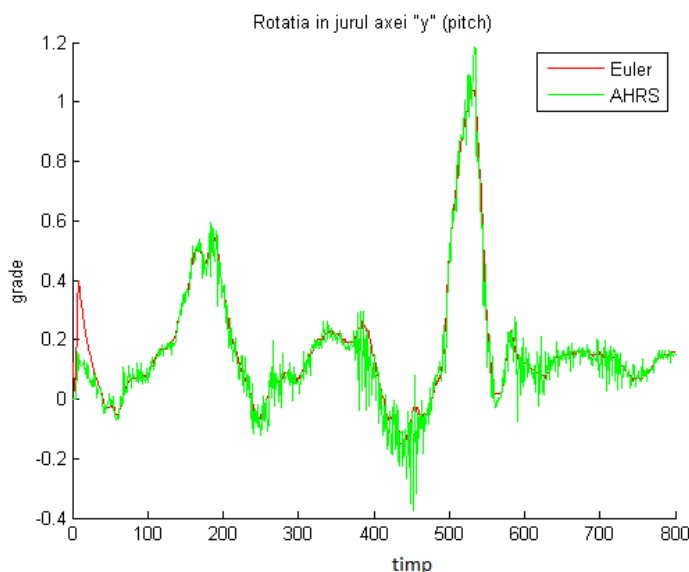


FIGURA 7.24: Reprezentarea orientării în raport cu axa y

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

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

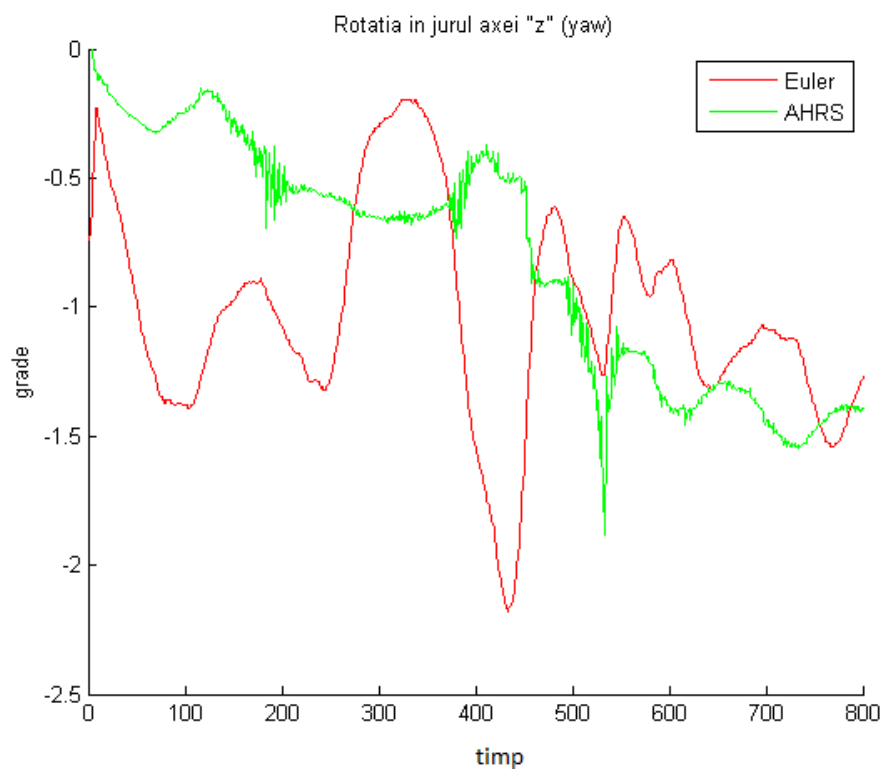


FIGURA 7.25: Reprezentarea orientării în raport cu axa z

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

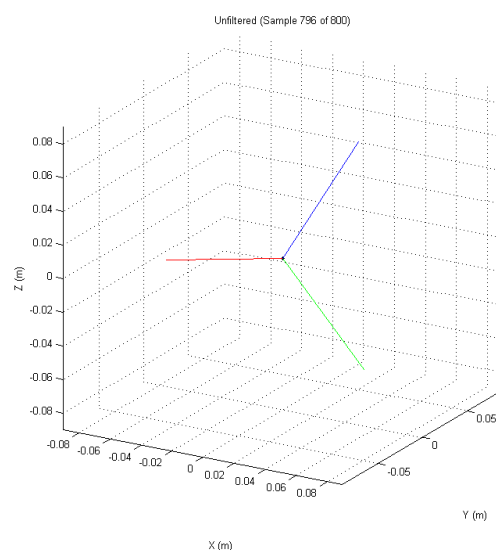


FIGURA 7.26: Reprezentarea orientării în 3D

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 parametri 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 frecvență ridicată, facilități ce este foarte folositoare 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 într-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 folosiţi şi dezvoltati în domeniile aplicabile orientării. Astfel primul algoritm studiat a fost algoritmul unghiurilor lui Euler, principiu ce face baza orientării în spaţiul 3D şi care are performanţe bune vizibile prin experimentele folosite, fiind implementabil în aplicaţiile în timp real, datorită timpului de răspuns foarte rapid şi timpului scurt de stabilizare. Studiarea mai amănunţită 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 şi face ca două planuri ale axelor de coordonate să fie paralele, atunci sistemul devenind unul 2D, fiind nevoie de o mişcare necontrolată pentru deblocare. Principiul blocării gimbal face ca modelul unghiurilor lui Euler să fie un model problematic în determinarea orientării 3D.

Pentru a evita problemele evidenţiate de principiul unghiurilor lui Euler s-a observat că, în multe studii sa folosit reprezentarea spaţiului 3D în forma cuaternionilor, o extindere matematică a numerelor complexe, iar orientarea pe baza acestora se face printr-o filtrare Kalman a datelor inerţiale, astfel se reduc influenţele gravitaţiei şi câmpului magnetic terestru, orientarea fiind determinată mult mai precis prin elaborarea unei estimări a acestei analizând starea precedentă, rezultând un control mult mai bun. Această metodă pare ideală, dar problema ei este reprezentată de complexitatea calculurilor şi multitudinea de iteraţii, fapt ce o duce la întârzieri, ce afectează implementarea într-o aplicaţie 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 şi îmbunătăţit de Madgwick, care implementează mai puţine calcule şi iteraţii, astfel rezultă un răspuns mai bun al sistemului, la performanţe asemănătoare filtrului Kalman.

Prin implementarea metodelor unghiurilor lui Euler şi algoritmului AHRS s-a observat că cele două metode prezintă rezultate asemănătoare în determinarea orientării în funcţie de axele x şi y , diferenţa 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 doreşte monitorizarea orientării în funcţie de axa z este nevoie de un filtru ce compensează componenta acceleraţiei gravitaţionale, a influenţei câmpului magnetic şi a erorilor giroscopului, cel mai eficient algoritm fiind expus în filtrul AHRS.

8.2 Direcţii de dezvoltare ulterioară

Pe baza realizărilor acestui proiect, şi observării altor aplicaţii ale metodelor studiate în această aplicaţie, dar şi prin prisma dezvoltării sistemelor de acest tip se pot trasa mai multe direcţii de dezvoltare.

O direcţie de dezvoltare ar duce la compresarea unui neajuns al aplicaţiei curente. Acest neajus este legat de conectarea cablată între calculator şi 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 direcţie de dezvoltare poate fi făcută prin dezvoltarea aplicaţiei pe o platforma Raspberry Pi, care conţine implicit un modul ce permite conexiunea wireless, iar sistemul de operare Linux, folosit de această placă de dezvoltare este mult mai stabil, şi permite mai multe facilităţi legate de tipul conexiunii, menţionând şi timpul de calcul mult mai eficient.

O altă direcţie de dezvoltare este reprezentată de îmbunătăţirile ce pot fi aduse, cu scopul implementării în cadrul aplicaţiilor Unity3D, astfel se poate dezvolta pe baza algoritmilor dezvoltaţi în acest proiect, şi prin îmbunătăţirea structurii hardware, un prototip de mână ce va oferi utilizatorului o experienţă, în animaţia 3D, conformă cu mişcările realizate de mână şi degetele sale. Astfel jocul dezvoltat va fi mult mai interactiv, calitatea ce poate fi sporită prin adăugarea unei componente haptice, în cadrul acestei mânuşi. Această componentă oferindu-i utilizatorului un răspuns tactil la acţiunile realizate de el în cadrul jocului.

Precum s-a observat în secţiunea de introducere a proiectului, studiile de determinare a orientării se folosesc intens în proiectele aerospaţiale. Pe baza studiilor şi cunoştinţelor dobândite în acest proiect se poate trasa un obiectiv de realizare unui quadcopter ce va fi controlat prin mişcările mâinii utilizatorului. Această dezvoltare este una foarte complexă având în vedere că implică şi 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. Actuators,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 effects of magnetic variations on inertial/magnetic orientation sensors". IEEE International 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. Carneiro. "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, Golafsooun Ameri, Mazdak Zeresheki, S.Krishnan, M.Abdolirramaki. "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/>