Universitatea Tehnică „Gheorghe Asachi” din Iași
Facultatea de Automatică și Calculatoare
Departamentul de Automatică și Informatică Aplicată

Lucrare de licență

Proiectarea și dezvoltarea unui robot mobil de tip pendul inversat

Autor
Popovici Mircea-Alexandru

Profesor coordonator
Șef lucrări dr. ing. *Brăescu Florin Cătălin*

Iași, 2020

Cuprins

Figuri

Tabele

Capitolul 1. Introducere

1.1. Scopul lucrării

Lucrarea documentează procesul de proiectare și dezvoltare a unei platforme robotice mobile de tip pendul inversat. Platforma pune la dispoziție funcționalitățile necesare pentru a facilita implementarea și testarea unor algoritmi de control specifici problemei pendulului inversat.

1.2. Cercetări actuale

Dacă termenul de „robot”, cu semnificația sa modernă, a fost popularizat de scriitorul și biochimistul Isaac Asimov într-o scurtă povestire intitulată „Fuga în cerc” (engl. *Runaround*), publicată abia în 1942, ideea din spatele noțiunii de robot este o temă de gândire ce a preocupat marile minți pe parcursul întregii istorii cunoscute a omului, de la mitologia greacă și legendele meselor automate a lui Hefaistos, la automatele umanoide dansatoare ale dinastiei chineze Zhou și de la cavalerul mecanic al lui Leonardo da Vinci și rața lui Jacques de Vaucanson la Elektro, robotul umanoid prezentat de Statele Unite la Târgul Internațional din 1939. În a doua jumătate a secolului trecut, apariția și evoluția tranzistoarelor a făcut posibilă apariția unor progrese incredibile într-o varietate de domenii științifice. În speță progresele făcute în domeniul electronicii digitale au făcut ca noi și noi metode de control și comandă a diverselor dispozitive să fie dezvoltate, iar o dată cu aceasta și interesul pentru roboți a crescut în lumea ingineriei. O nouă zonă de studiu, cea a roboticii, s-a distins deci în această perioadă ca o ramură interdisciplinară a ingineriei și a informaticii. Să spunem că roboții și robotica au evoluat de atunci ar fi o afirmație cât se poate de modestă. În ultimii 50 de ani roboții au produs o a treia revoluție industrială, au revoluționat domeniul medicinei și în speță al chirurgiei, au ajuns în casele omului de rând și chiar în industria divertismentului. Putem afirma deci că robotica este în continuare un domeniu de actualitate, de interes, mai ales considerând strânsa legătură pe care o are cu domeniul inteligenței artificiale, un domeniu aflat în plină dezvoltare.

Cea mai relevantă categorie de roboți pentru lucrarea și aplicația curentă este cea a roboților cu capabilitatea de menținere a propriului echilibru (engl. *Self-balancing robots*). În această categorie se regăsesc modele de roboți proiectate încă de la jumătatea secolului trecut, când a fost și dezvoltat un astfel de sistem sub forma unui cart cu patru roți, cu o tijă atașată printr-o articulație de rotație cu un grad de libertate. Scopul sistemului era ca deplasând corespunzător cartul să se mențină tija și o eventuală greutate plasată la capătul ei liber în poziție verticală (**Figura 1.1**). Între exemplele mai recente se găsesc roboții de tipul „ballbot”, primul astfel de sistem funcțional fiind construit la Universitatea „Carnegie Mellon” din Statele Unite în 2005 [1] (**Figura 1.2**). Dacă modelele prezentate până acum provin din mediul academic, este important de precizat că astfel de sisteme robotice sunt folosite și în mediul industrial sau în domeniul transporturilor. Corporații multinaționale precum Amazon folosesc roboți personalizați pentru a transporta diverse mărfuri între anumite

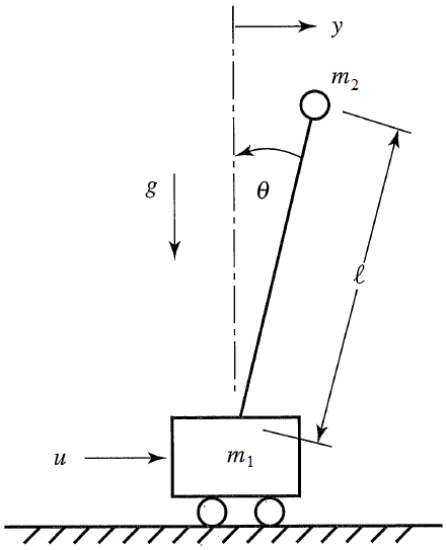


Figura 1.1 – Modelul robotul de tip cart cu tijă

puncte din halele lor, iar în rândul tinerilor sistemele de tip „Segway” sau „hoverboard”, care în esență funcționează pe aceleași principii de bază, devin tot mai populare.

Toate modelele și sistemele menționate împărtășesc un set de proprietăți comune și funcționează în virtutea acelorași principii de bază, care sunt definite de modelul pendulului inversat. Simplitatea modelului pus în discuție face ca problema de reglare ce i se atașează să fie accesibilă tuturor inginerilor cu pregătire în domeniul teoriei sistemelor. Aceasta nu ar trebui confundată cu a afirma că problema de reglare este banală, ci din contră. Problema controlului unui pendul inversat este frecvent menționată ca un etalon [2] în formarea inginerilor în domeniu și poate că adevărata ei frumusețe constă în faptul că ea poate fi abordată prin metode diverse, de la cele mai simple la cele mai avansate. Astfel că un proaspăt inginer poate ataca problema folosind inițial o metodă cu un regulator PID, urmând ca pe măsură ce acumulează experiență să revină asupra temei, spre a-și îmbunătăți abilitățile prin exersarea unor metode mai avansate de control.

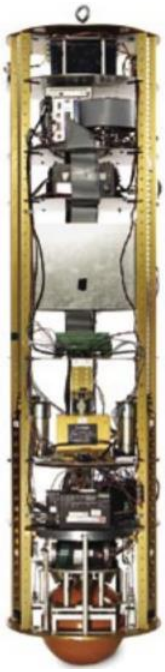


Figura 1.2 – Ballbot

1.3. Descrierea aplicației

În contrast cu aplicația clasică, ce presupune atașarea unui pendul inversat pe o șină culisantă sau pe un vehicul cu patru roți, platforma robotică dezvoltată în cadrul aplicației prezentate în această lucrare presupune ca întregul corp să constituie greutatea pendulului, iar echilibrarea acestuia să se facă prin acționarea a două motoare conectate la două roți laterale. În acest scop am proiectat un cablaj imprimat personalizat, echipat cu un microcontroler, un modul de senzori inerțiali, un modul de comunicație fără fir, drivere pentru motoare de curent continuu și regulatoare de tensiune. Pentru suportul fizic am proiectat un șasiu personalizat ce acomodează cablajul și motoarele robotului, iar apoi am calculat diverși parametri ai platformei robotului cu care am dezvoltat un model intrare-stare-ieșire. Pentru componentele electronice amintite am dezvoltat biblioteci de cod și ulterior am dezvoltat o aplicație demonstrativă formată dintr-un program pentru robot, unul pentru o placă Arduino și unul pentru calculatorul personal.

O descriere mai detaliată a părților componente a proiectului se găsește în capitolele următoare, după cum urmează:

- O privire asupra câtorva noțiuni teoretice necesare pentru înțelegerea lucrării
- Prezentarea structurii hardware a aplicației
- Structura software a aplicației
- Concluzii

Capitolul 2. Considerații teoretice

2.1. Pendul inversat

Modelul pendulului inversat, ce a fost anterior menționat, se referă, în varianta sa simplă, la un model derivat din cel al pendulului clasic. În esență, acesta poate fi construit ca un pendul obișnuit, cu singura restricție fiind ca greutatea sa să fie atașată articulației printr-un suport rigid. Diferența majoră constă, însă, în faptul că centrul de greutate al pendulului trebuie adus și menținut deasupra articulației de care este fixat, ceea ce îl face instabil în mod natural. Așadar, asupra unui pendul inversat este nevoie să fie aplicate forțe externe pentru a compensa lipsa de stabilitate. De aceea nu putem vorbi despre un pendul inversat fără a vorbi despre sistemul fizic din care face parte, ce face posibil controlul său.

O caracteristică importantă a pendulului este că destabilizarea nu se produce liniar. Viteza unghiulară a greutății fixate crește în timp, ceea ce înseamnă că pendulul deviază tot mai rapid față de poziția verticală de echilibru. Pentru un sistem ca cel din **Figura 2.1**, această evoluție neliniară poate fi ușor observată, considerând accelerația unghiulară $\ddot{\theta} = \frac{g}{l}\sin\theta$. [3, 4, 5]

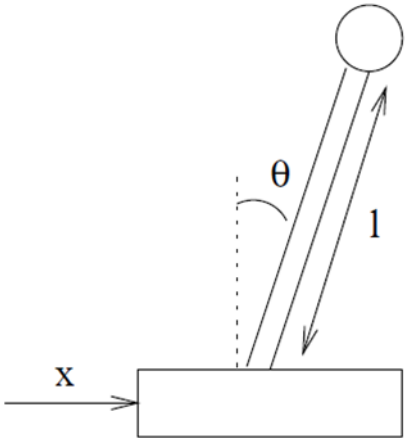


Figura 2.1 – Reprezentarea simplificată a unui sistem generic cu pendul inversat [4]

După cum am mai menționat, pentru a descrie comportamentul unui pendul inversat trebuie să descriem metoda prin care compensăm instabilitatea sa inerentă. În realitate, pendulul inversat nu este deci discutat ca un concept independent, ci ca o componentă a unui sistem specializat. Între modelele consacrate de sisteme cu pendul inversat se găsesc:

1. Căruciorul cu tijă
2. Acrobot / Pendubot
3. Pendulul Furuta
4. Pendulul cu roată reactivă
5. Sistemul „Beam-and-ball”
6. Robotul mobil cu două roți, cu auto-balansare

Aplicația curentă este un exemplu clar de sistem din ultima categorie, dar prezintă similitudini în funcționare cu pendulul Furuta [6] și căruciorul cu tijă. Mai specific, dacă limităm mișcarea robotului doar înainte și înapoi, atunci el poate fi modelat asemănător cu un sistem de cărucior cu tijă, iar dacă studiem comportamentul robotului în timpul virajelor, putem constata prezența unor similarități cu pendulul Furuta.

2.2. Giroscop & Accelerometru

Cea mai simplă construcție pe care o putem numi „giroscop” este un disc fixat prin centrul său cu o tijă. Dacă tija este la rândul ei fixată pe un inel la ambele capete ale sale, iar discul se poate roti și poate lua orice orientare raportat la tijă, atunci întreg ansamblul poate fi considerat a fi un giroscop cu un grad de libertate. De cele mai multe ori, atunci când vorbim despre un giroscop clasic însă, ne referim la unul cu trei grade de libertate, format din două sau trei inele de mărimi

diferite și un disc fixate într-un mod similar, ca cel descris anterior. În virtutea legii a treia a lui Newton, funcționarea unui giroscop are la bază conservarea momentului cinetic unghiular. Când discul din mijloc este în rotație, orientarea axei sale nu este afectată de mișcarea sau rotația inelului de care este fixat.

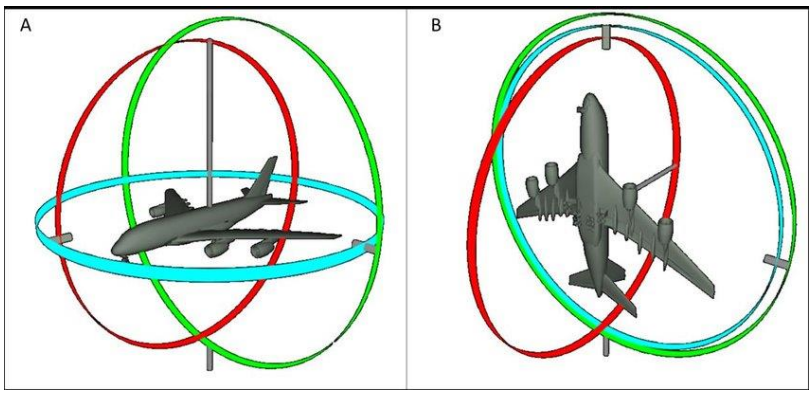


Figura 2.2 – Problema de blocaj gimbal în lucrul cu unghiuri Euler (A. fără blocaj, B. Unghiurile de rulație și girație în blocaj) [7]

Mai mult decât o simplă curiozitate, un astfel de dispozitiv poate fi folosit pentru a deduce orientarea unui obiect față de o referință, ceea ce se și întâmplă în practică. Giroscopul este afectat, însă, de o problemă, numită *blocaj gimbal* (engl. *Gimbal Lock*) [7], care apare atunci când două dintre inelele unui giroscop se aliniază, dispozitivul pierzând astfel un grad de libertate. În **Figura 2.2** poate fi observat efectul unui astfel de blocaj: atunci când inelul corespunzător unghiului *pitch* (cian) se aliniază cu inelul *yaw* (verde), o eventuală schimbare a unghiului *roll* (roșu) produce același efect ca o variație a unghiului *yaw*. Soluțiile pentru a rezolva acest neajuns diferă în funcție de modul de implementare al giroscopului.

În electronică sunt populare giroscoapele de tip MEMS (engl. *Micro-electromechanical System*), care folosesc un ansamblu de condensatoare pentru a măsura efectul forțelor Coriolis asupra unor elemente vibratoare. Aceste dispozitive sunt și ele afectate de blocajul gimbal, iar problema este cel mai adesea rezolvată prin fuzionarea datelor citite cu cele ale unui senzor accelerometru.

Un accelerometru este un dispozitiv ce calculează accelerația obiectului de care este atașat raportat la accelerația gravitațională, în stare de repaos el măsurând o accelerație de 1g ($g = 9.81 m/s^2$). În electronică, din nou, sunt folosite accelerometre de tip MEMS, de cele mai multe ori în același integrat cu un senzor giroscopic și un magnetometru.

Capitolul 3. Structura hardware a platformei robotice

Procesul de proiectare și dezvoltare al unei aplicații în domeniul roboticii implică în genere utilizarea unui cumul de echipamente, componente mecanice, electrice și electronice și alte resurse, fie ele hardware sau software. Capitolul curent detaliază asupra aspectelor palpabile fizic, asupra structurii hardware a aplicației, cu accent pe platforma robotică.

Pentru a putea vorbi despre aceasta, trebuie să precizăm un set de specificații ce stau la baza alegerii sau proiectării părților componente ale platformei:

1. Robotul trebuie să dispună de suficientă putere de procesare pentru a își executa toate procesele în timpi suficient de scurți, care să permită implementarea unui regulator robust.
2. Robotul trebuie să dispună de capabilități senzoriale adecvate pentru a putea pune la dispoziție toate datele relevante funcționării unui algoritm de reglare.
3. Robotul trebuie să aibă o autonomie de funcționare cât mai mare, deci trebuie alese și folosite componentele în mod adecvat spre a se ajunge la un raport optim între energia disponibilă și consum.

În linii mari, platforma robotică este compusă dintr-un cablaj electronic imprimat personalizat, echipat cu capabilități senzoriale, cu un procesor adecvat aplicației, cu dispozitive de control a motoarelor și capabilități de comunicare la distanță. Dacă acest cablaj este „creierul" platformei, interacțiunea cu spațiul fizic se realizează prin două motoare dotate și ele cu senzori sub forma unor codificatoare cu efect Hall. Robotul prinde formă prin montarea și fixarea componentelor precizate pe un șasiu personalizat. În cele ce urmează vor fi descrise toate aceste piese, împreună cu nevoile în virtutea cărora au fost alese.

3.1. Placa de dezvoltare

După cum am menționat anterior, cablajul imprimat împreună cu dispozitivele cu care acesta este echipat formează de fapt componenta activă și interactivă a robotului. Înainte de a prezenta felul în care circuitul permite buna funcționare a tuturor componentelor electronice incluse, va fi detaliat rolul și comportamentul individual al fiecărei astfel de piese.

3.1.1. Regulatoarele de tensiune

Buna funcționare a platformei robotice este asigurată de piese ce ajută fie prin aportul la logica de funcționare, fie prin interacțiunea directă cu mediul înconjurător. Cele două categorii de piese au cerințe diferite pentru a opera corespunzător, motiv pentru care am dotat placa dezvoltată cu două regulatoare de tensiune: unul pentru circuitele logice și unul pentru motoare. De cele mai multe ori, circuitele logice funcționează la tensiuni de 3 – 3.6V sau 5V. În cazul curent, toate componentele logice operează corespunzător la 3.3V, iar motoarele funcționează la o tensiune nominală de 6V, cele două regulatoare satisfăcând întocmai aceste cerințe.

Cele două opțiuni principale pentru obținerea tensiunilor căutate sunt regulatoarele lineare de tip LDO (engl. *Low-dropout regulator*) și regulatoarele cu funcționare în comutație (engl. *Switching DC-DC*). LDO-urile sunt mici ca dimensiuni și nu produc niciun zgomot, însă disipă în mod constant putere, ceea ce înseamnă că risipesc o cantitate semnificativă de energie sub formă de căldură. Prin comparație, regulatoarele în comutație sunt mai mari ca dimensiuni și pot induce mici zgomote, atât audio, cât și în măsurătorile senzorilor pe care îi alimentează. Totuși, acestea

au marele avantaj că nu consumă energie în gol, ceea ce se traduce printr-o eficientizare a consumului total de energie al platformei robotice.

Pentru că una din specificațiile robotului cere ca acest consum să fie cât mai mic, am optat pentru folosirea a două regulatoare cu funcționare în comutație, produse de OKYSTAR, plecând de la cipul MP1584EN al companiei Monolithic Power Systems Inc.. Regulatoarele alese prezintă și avantajul suplimentar că tensiunile lor de ieșire pot fi reglate printr-un potențiometru, ceea ce permite înlocuirea unora dintre componente cu altele, ce operează la alte tensiuni. Aceasta înseamnă, de exemplu, că utilizatorul poate opta pentru alte motoare decât cele propuse în această descriere, singura modificare necesară fiind de a regla tensiunea de ieșire a regulatorului corespunzător, între 0.8 – 20V.

În aplicația curentă, cele două regulatoare sunt alimentate de la un acumulator Li-Ion (acumulator litiu-ion) cu două celule, cu o capacitate de 2.6Ah (putere de 19.24Wh), tensiune nominală de 7.4V și tensiunea de tăiere a alimentării de 8.4V. Regulatoarele pot fi opțional alimentate de la o sursă fixă sau un alt fel de acumulator, atât timp cât acestea pot furniza o tensiune între 6 – 28V, însă particularitățile de proiectare a robotului impun ca tensiunea maximă de alimentare să nu depășească 12V.

Frecvența de comutare a regulatoarelor este, în mod normal, 1MHz și poate urca până la 1.5MHz, deci acesta este intervalul de frecvențe în care trebuie urmărită și apariția perturbațiilor în măsurători.

3.1.2. Senzorii

Una dintre specificațiile platformei robotice se referă la capabilitatea de a colecta periodic date despre propriul unghi de înclinare, iar pentru a satisface această cerință sistemul include un modul PCB GY-251, dotat cu circuitul integrat MPU6050, produs de TDK InvenSense [8]. Circuitul integrat în cauză este format dintr-un senzor giroscopic MEMS (engl. *Micro-electromechanical System*) pe trei axe și un accelerometru MEMS pe trei axe, un senzor de temperatură cu ieșirea digitală și un procesor de mișcare digital cu tehnologia MotionTracking™. Procesorul menționat are o serie de funcții, precum detecția căderii în gol sau a unor gesturi, dar cel mai important pentru aplicația curentă este că procesorul poate filtra datele de la cei doi senzori și, mai mult, poate elimina blocajul gimbal prin fuzionarea datelor obținute de la aceștia. Mai multe detalii despre cum este folosit acest procesor vor fi date în capitolul următor.

Figura 3.1 – Modulul MPU-6050

Interfațarea cu microcontrolerul se face prin protocolul de comunicație I^2C (engl. *Inter-Integrated Circuit*). Datele preluate de către microcontroler de la cei doi senzori pot fi utilizate pentru a implementa un algoritm de reglare.

3.1.3. Modulul de comunicație

Pentru comunicarea cu aplicații externe, platforma robotică este echipată cu un modul de emisie-recepție bazat pe cipul nRF24L01+, produs de Nordic Semiconductor [9], alimentat la tensiuni între 1.9V și 3.6V, cu un consum maxim de 13.6mA măsurat la recepție, cu emisie și recepție la frecvențe de 2.4 – 2.525GHz și viteză de transfer de 1Mbps sau 2Mbps, selectabilă. Circuitul poate comunica pe distanțe de până la 80m în câmp

Figura 3.2 – Modulul de comunicație nRF24L01+

deschis, pe 125 de canale, are două moduri de funcționare la consum redus de energie și se poate interfața cu microcontrolerul prin protocol SPI (engl. *Serial Peripheral Interface*). Modulul în forma sa din **Figura 3.2** include un cristal de cuarț de 16MHz și o antenă potrivită și garantează posibilitatea comunicării corecte cu un al doilea modul identic.

3.1.4. Driverele de motor

Fiecare dintre cele două motoare este comandat de un driver Pololu [10], construit în jurul circuitului integrat MAX14870 [11], produs de Maxim Integrated și conceput special pentru comanda motoarelor de curent continuu. Driverul suportă motoare ce funcționează la tensiuni între 4.5V și 36V și curenți de până la 1.7A în regim normal, adică mai mult decât suficient pentru motoarele alese. Partea logică a driverului suportă tensiuni între 3V și 5V, ceea ce permite interfațarea cu microcontrolerul dsPIC33 fără componente adiționale.

Circuitul primește la o intrare un semnal PWM care, practic, dictează valoarea tensiunii cu care driverul alimentează mai departe motorul, tensiune care se transpune în viteza de rotație a motorului comandat. Pe pinul DIR, driverul primește un semnal ce dictează sensul curentului pe circuitul de alimentare al motorului și deci direcția de rotație a motorului comandat. Suplimentar, driverul oferă o ieșire digitală, activă pe nivel de tensiune scăzut. Această ieșire trece în „0" logic, atât timp cât sesizează o eroare de funcționare. Modulul periferic PWM al procesorului poate folosi această informație pentru a opri sau relua generarea de semnal modulat, în funcție de starea motoarelor, fără a fi nevoie de intervenția aplicației utilizator și deci fără a consuma timpi de procesare suplimentari.

Figura 3.3 – Funcțiile pinilor driverului MAX14870

3.1.5. Microcontrolerul

Dispozitivele ce intră în componența platformei robotice folosesc diverse protocoale de comunicare și semnale digitale, folosirea unui algoritm de control impune procesarea numerică a datelor, iar prezența motoarelor implică necesitatea existenței hardware-ului specializat pentru

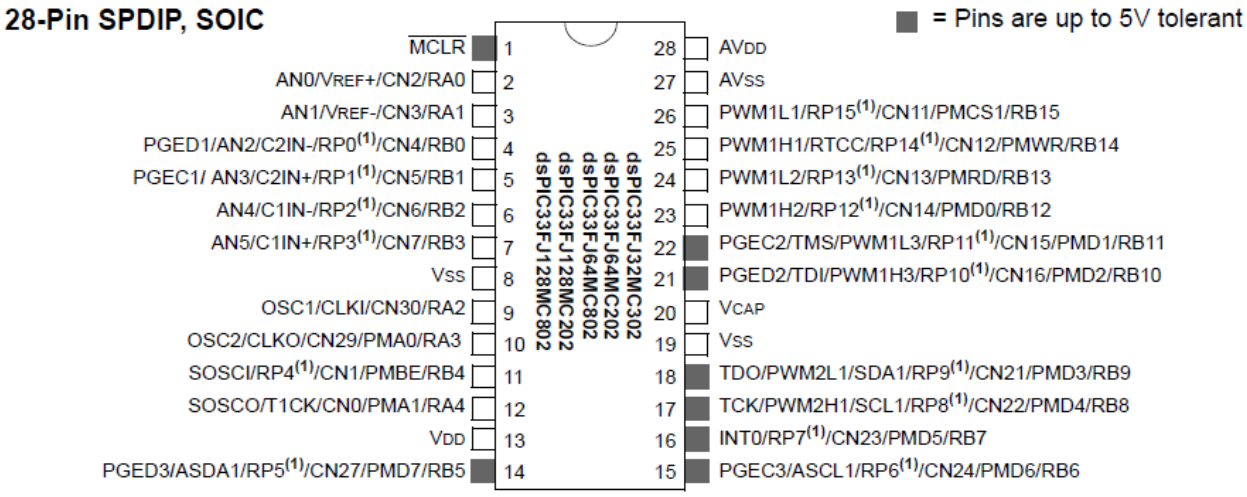


Figura 3.4 – Configurația pinilor microcontrolerului dsPIC33FJ128MC802

comanda lor. Pentru a putea conecta împreună toate componentele este deci nevoie ca procesorul să dispună de capabilități de interfațare cu toate protocoalele folosite de celelalte componente, iar pentru rularea unui algoritm de control este nevoie ca procesorul să fie suficient de puternic. De aceea, pentru aplicația curentă am ales un procesor numeric de semnal, produs de Microchip Technology Inc., dsPIC33FJ128MC802 [12].

Microcontrolerul este conceput cu o arhitectură Harvard, cu magistrale separate pentru program și memorie de date, ceea ce permite lucrul simultan cu instrucțiuni organizate în cuvinte de 24 de biți și date organizate în cuvinte de 16 biți. Dispozitivul se alimentează la o tensiune între 3 și 3.6V, dispune de 128Ko de memorie RAM (engl. *Random-Access Memory*) de date, 128Ko de memorie Flash de program, o arhitectură de funcționare pe 16 biți și viteză de procesare maximă de până la 80 MIPS (engl. *Millions of Instructions Per Second*), folosind un cristal extern de cuarț. Caracteristica RISC a procesorului, împreună cu proprietățile amintite anterior, se traduc prin capabilitatea procesorului de a rula programe de complexitate ridicată și de a face deci cu succes față atât aplicațiilor din mediul industrial, al automatizărilor, cât și aplicațiilor de uz general.

Relevant pentru proiectul în discuție, constând într-o aplicație experimentală cu senzori și motoare de curent continuu, este că procesorul este suficient de rapid pentru a efectua calculele în timp real, are suficienți pini pentru a asigura toate conexiunile necesare și dispune de toate perifericele specializate de care are nevoie:

1. Procesorul este capabil să producă semnale PWM (engl. *Pulse-Width Modulation*) printr-un modul periferic specializat întocmai pentru comanda motoarelor. Acesta are mai multe moduri de funcționare și capabilități adiționale. De exemplu, perifericul poate fi programat ca atunci când detectează un semnal specific, apărut în urma unei defecțiuni, primit de la driverul de motor, să oprească generarea semnalului modulat și eventual să ridice o cerere de întrerupere în acest caz, urmând ca la dispariția semnalului să reia funcționarea normală, fără intervenția aplicației utilizator.
2. Procesorul include un periferic specializat pentru citirea, decodarea și filtrarea datelor primite de la codificatoarele de poziție pentru motoare. Alternativ, aproape orice pin de-al său poate fi folosit pentru a ridica o cerere de întrerupere software, prin care poate fi implementată aceeași funcționalitate.
3. Procesorul dispune de periferice specializate pentru comunicarea prin protocoalele I^2C, UART (engl. *Universal Asynchronous Receiver-Transmitter*) și SPI. Dintre acestea, perifericele I^2C și SPI sunt intens utilizate în aplicația curentă pentru a interfața controlerul cu celelalte componente.
4. Procesorul include un convertor analog-digital (engl. *Analog-Digital Converter*, abreviat ADC) accesibil prin unul din porturi. Acesta este folosit pentru a controla alimentarea platformei.
5. O ultimă capabilitate de interfațare folosită de aplicație este cea de primire a întreruperilor externe. Spre deosebire de întreruperile software menționate la punctul 2 al acestei liste, întreruperile externe sunt mai puține la număr, sunt fixate pe anumiți pini, pot fi mascate sau nemascate individual și au niveluri de prioritate programabile individual.

Nu în ultimul rând, unul dintre porturile procesorului este format din pini reconfigurabili, ceea ce permite ca diverse intrări sau ieșiri ale perifericelor să fie asignate convenabil pinilor de pe acest port. Această funcționalitate ajută în două feluri: inversarea rolurilor anumitor pini

ușurează proiectarea circuitului imprimat și, într-o aplicație generală, poate într-o oarecare măsură permite schimbarea procesorului original cu unul ușor diferit.

Funcționalitatea fiecărui pin este descrisă pe scurt în tabelul de mai jos. În capitolele următoare, fiecare dintre funcționalități va fi discutată mai în detaliu, cu accent pe semnificația particulară și modul de utilizare punctual al fiecăreia în cadrul platformei robotice dezvoltate. În **Anexa 2** se găsește un tabel cu mai multe detalii despre funcțiile fiecărui pin.

Tabel 3.1 – Funcțiile pinilor procesorului dsPIC33FJ128MC802 folosite

Nr. pin	Funcționalitate	Tip pin	Descriere
1	$\overline{\text{MCLR}}$	Intrare	Folosit pentru resetarea procesorului în timpul programării
2	AN0	Intrare	Folosit ca intrare pentru convertorul analog-digital
3	RA1	Intrare	Pin de intrare generic
4	PGD	Intrare	Folosit pentru primirea datelor în timpul programării
	RB0	Ieșire	Folosit ca pin de ieșire generic în timpul funcționării
5	PGC	Intrare	Folosit pentru primirea semnalului de ceas în timpul programării
	RB1	Ieșire	Folosit ca pin de ieșire generic în timpul funcționării
6	RP2 (PWM1FAULT)	Intrare	Pin cu re-mapare configurat ca intrarea FAULT a perifericului PWM1
7	RP3 (PWM2FAULT)	Intrare	Pin cu re-mapare configurat ca intrarea FAULT a perifericului PWM2
8	VSS	Alimentare	Conectat la masă
9	RA2	Intrare	Pin de intrare generic
10	RA3	Intrare	Pin de intrare generic
11	RP4 (SPI1MOSI)	Ieșire	Pin cu re-mapare configurat ca ieșire de date pentru perifericul SPI1
12	RA4	Intrare	Pin de intrare generic
13	VDD	Alimentare	Conectat la tensiunea de alimentare de 3.3V
14	RP5 (SPI1SS)	Ieșire	Pin cu re-mapare configurat ca ieșire de selecție a dispozitivului *slave* pentru perifericul SPI1
15	RP5 (SPI1MISO)	Intrare	Pin cu re-mapare configurat ca intrare de date pentru perifericul SPI1
16	INT0	Intrare	Întrerupere externă INT0
17	SCL1	Ieșire	Pinul semnalului de ceas generat de perifericul I2C1
18	SDA1	Intrare / Ieșire	Pinul semnalului de date pentru perifericul I2C1
19	VSS	Alimentare	Conectat la masă
20	VCAP	Alimentare	Pin conectat la regulatorul intern al nucleului procesorului
21	RP10 (SPI1SCK)	Ieșire	Pin cu re-mapare configurat ca ieșirea semnalului de ceas pentru perifericul SPI1
22	RP11 (INT1)	Intrare	Pin cu re-mapare configurat ca întreruperea externă INT1

23	PWM1H2	Ieșire	Pin de ieșire al semnalului PWM generat de perifericul PWM1, activ pe nivelul logic „1”
24	RB13	Ieșire	Pin de ieșire generic
25	PWM1H1	Ieșire	Pin de ieșire al semnalului PWM generat de perifericul PWM1, activ pe nivelul logic „1”
26	RB15	Ieșire	Pin de ieșire generic
27	AVSS	Alimentare	Conectat la masă
28	AVDD	Alimentare	Conectat la tensiunea de alimentare de 3.3V

3.1.6. Cablajul imprimat

Pentru ca piesele descrise până în acest punct să lucreze împreună, este nevoie de un suport adecvat, de trasee electrice alese corespunzător și de componente pasive potrivite, având în vedere particularitățile fiecărui semnal și a fiecărei componente. În plus, trebuie luat în considerare că, mai ales în dezvoltarea unei aplicații experimentale, în crearea unui prototip, există riscul ca unele componente să cedeze. Din aceste considerente am ales ca pentru componentele electronice să proiectez un cablaj imprimat personalizat, care, în loc să încorporeze în mod direct circuitele integrate discutate anterior, pune la dispoziție socluri potrivite pentru module PCB prefabricate care le includ. Această abordare oferă o serie de avantaje atât pentru autor în calitatea de proiectant, cât și pentru utilizator:

1. Modulele PCB dispun de conectori pentru interfațare, ce sunt așezați astfel încât ușurează accesul la diversele semnale. Dacă un integrat MAX14870 are, de exemplu, pini de semnal de jur împrejurul capsulei de siliciu, modulul PCB care îl include se interfațează prin două barete de pini, dintre care una este folosită pentru alimentare și conexiunea cu motoarele, iar cealaltă pentru schimbul de informații cu microcontrolerul.
2. Modulele PCB includ toate componentele pasive necesare bunei funcționări a integratelor pe care le adăpostesc. De exemplu, modulul de comunicație nu doar că include condensatoare de decuplare pentru a asigura buna alimentare a integratului, dar include un traseu electric expus, ce servește drept antenă, și pasivele potrivite pentru respectiva antenă. Prin aceasta, modulul simplifică procesul de proiectare, elimină necesitatea calculelor, simulării și testării pentru antenă și garantează că platforma robotică poate comunica fără probleme cu un circuit dotat cu un modul similar sau identic.
3. Modulele PCB lasă spațiul de pe placă, de sub ele, liber. Deci, în aceste spații pot fi plasate componentele ce trebuie montate direct pe placă, ceea ce se traduce printr-o economie de spațiu semnificativă.
4. Dacă cele menționate până acum sunt avantaje din punctul de vedere al proiectantului, cel mai mare avantaj este, de fapt, un beneficiu pentru utilizator. Modulele PCB pentru care am proiectat placa se găsesc în comerț în număr mare și la prețuri scăzute, ceea ce înseamnă că, în cazul unei defecțiuni apărute la una dintre componente, aceasta poate fi foarte ușor înlocuită.

5. Nu în ultimul rând, pentru că modulele folosite sunt populare atât în aplicații industriale, cât și în proiecte personale, utilizatorul poate găsi ușor informații relevante despre cum acestea pot fi folosite.

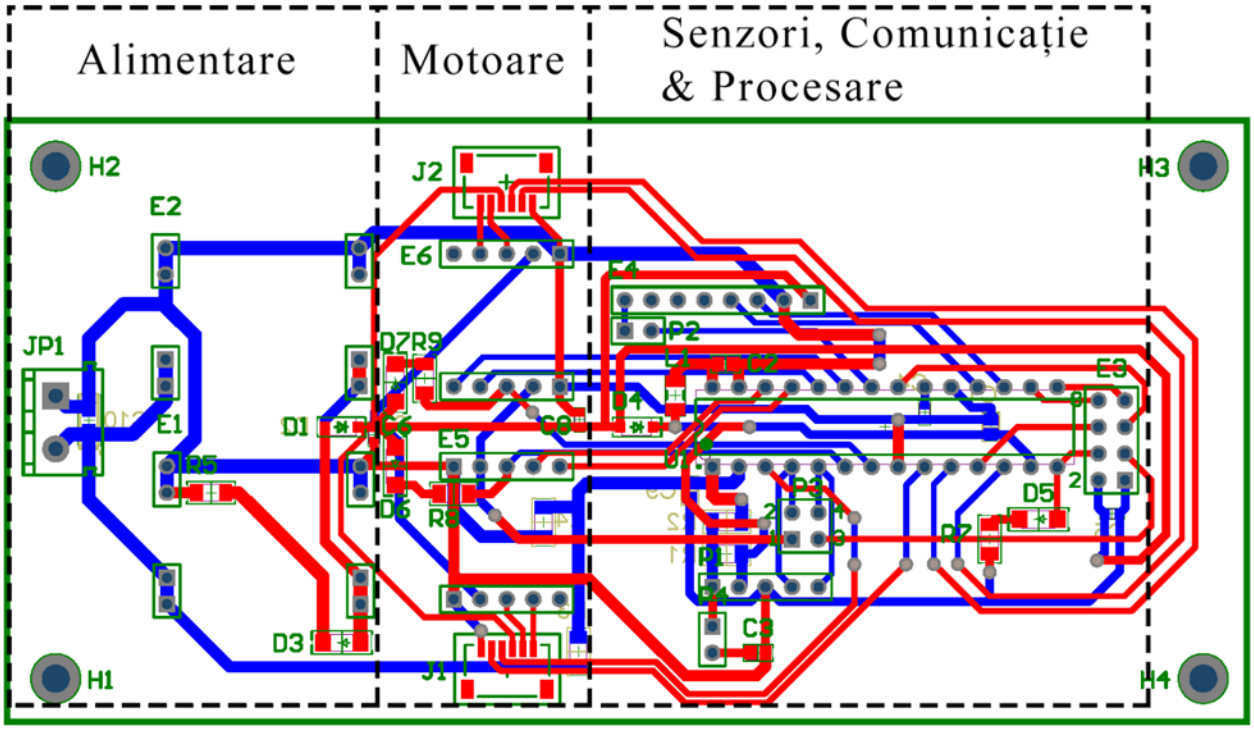


Figura 3.6 – Organizarea cablajului imprimat. Vedere în 2D.

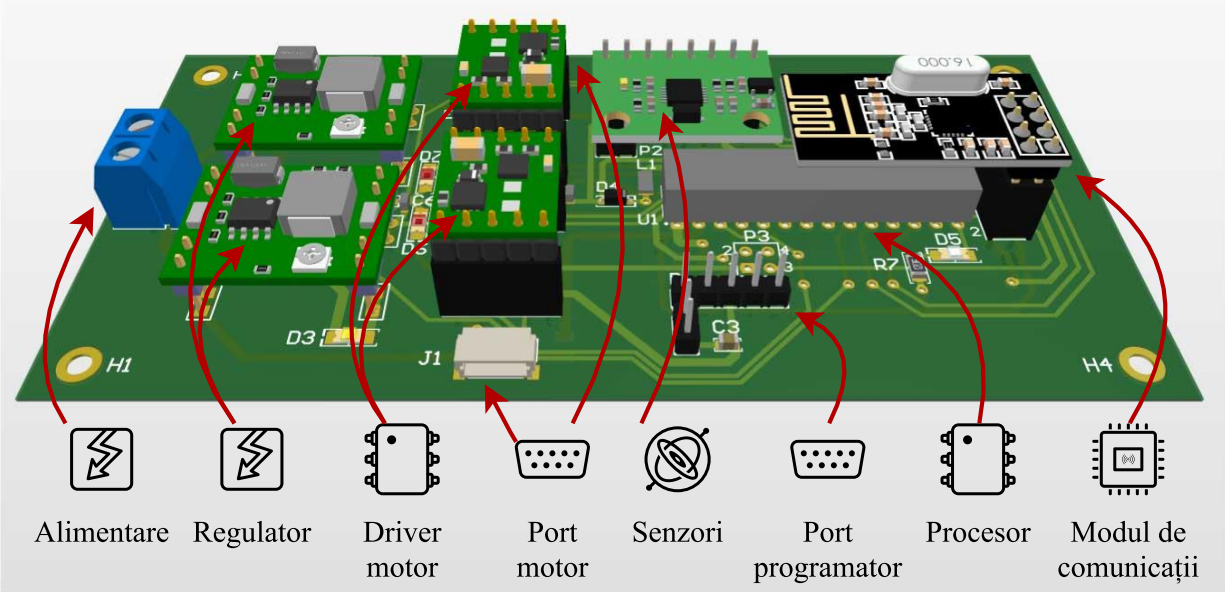


Figura 3.5 – Organizarea cablajului imprimat. Vedere în 3D din lateral.

Placa dezvoltată ia în considerare și formatul fizic al robotului, de aceea are o formă dreptunghiulară, și este organizată pe secțiuni. După cum poate fi observat în **Figura 3.6** și **Figura 3.5**, mufa de alimentare este poziționată în partea stângă a plăcii, iar regulatoarele de tensiune sunt

plasate în imediata proximitate a conectorului de alimentare. Urmează secțiunea pentru controlul motoarelor. Cele două drivere de motor sunt poziționate simetric față de axa plăcii, fiecare fiind însoțit de câte un LED ce semnalează apariția defecțiunilor și de câte un conector foarte compact, de tipul JST-SH, pentru a face legătura cu motoarele și codificatoarele cu care acestea sunt dotate. În porțiunea rămasă a plăcii se găsesc senzorul MPU6050 și modulul de comunicație, poziționate direct deasupra procesorului. Felul în care este plasat modulul MPU nu doar că economisește spațiu și oferă un aspect ordonat plăcii, dar face ca axele senzorului să coincidă cu axele robotului, ceea ce înseamnă că datele obținute de la senzor pot fi folosite în aplicație fără a mai necesita reinterpretări sau calcule cu matrice de rotație.

O privire mai clară asupra conexiunilor între componente, făcând abstracție de poziționarea lor pe PCB, se găsește în **Anexa 1**, prin schema plăcii. Per ansamblu, există două tipuri de conexiuni pe cablajul imprimat: trasee de alimentare și trasee de semnal.

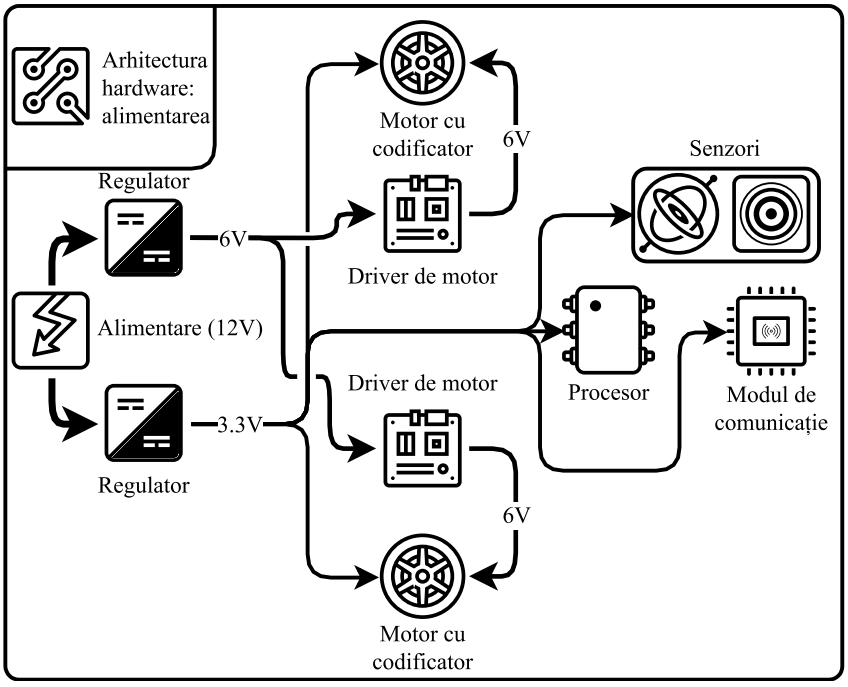


Figura 3.7 – Schema simplificată a alimentărilor

Pe placa dezvoltată există o alimentare de 6V, care este folosită de driverele de motor pentru a produce comanda motoarelor. Regulatorul este protejat de curenții de întoarcere printr-o diodă Schottky, ce suportă curenți constanți de până la 3A, suficienți pentru motoarele folosite. Din borna diodei se disting două trasee de alimentare, câte unul pentru fiecare motor, ceea ce înseamnă că acestea sunt conectate în paralel, un aspect important pentru că astfel o cădere de tensiune sau o defecțiune pe unul dintre trasee nu va afecta și motorul conectat la celălalt traseu. Alimentarea componentelor cu logică digitală se face din regulatorul de 3.3V, de altfel protejat printr-o diodă Schottky. Din aceasta pleacă în paralel trasee către codificatoare, către driverele de motor și un traseu care ulterior se împarte la rândul său către procesor, modulul de comunicație și modulul cu senzori. Înainte de alimentarea procesorului, pe placă se găsește încă o diodă Schottky, pentru a preveni alimentarea întregului circuit de către programator în timpul operațiilor de programare sau depanare.

Structura de trasee de semnal este sintetizată în **Figura 3.8**. Există un traseu direct de la conectorul de alimentare la microcontroler, folosit pentru a inactiva toate componentele atunci când bateria este descărcată. Pentru implementarea protocolului I^2C este nevoie de două trasee, pentru SPI este nevoie de patru conexiuni, iar pentru ICSP (engl. *In-Circuit Serial Debugger*) este nevoie de două trasee de semnal și încă unul care supra-alimentează procesorul. Cele două module ce comunică prin SPI și I^2C completează protocoalele folosite prin utilizarea întreruperilor. Cu alte cuvinte, există două trasee suplimentare, unul între procesor și modulul de senzori și unul între

procesor și modulul de comunicație, rezervate pentru întreruperi externe. Nu în ultimul rând, modulul de comunicație mai are nevoie de o conexiune directă pentru un semnal ce permite sau inhibă transferul mesajelor prin protocolul fără fir.

Placa este prevăzută cu patru găuri ce permit fixarea ei pe șasiu folosind șuruburi.

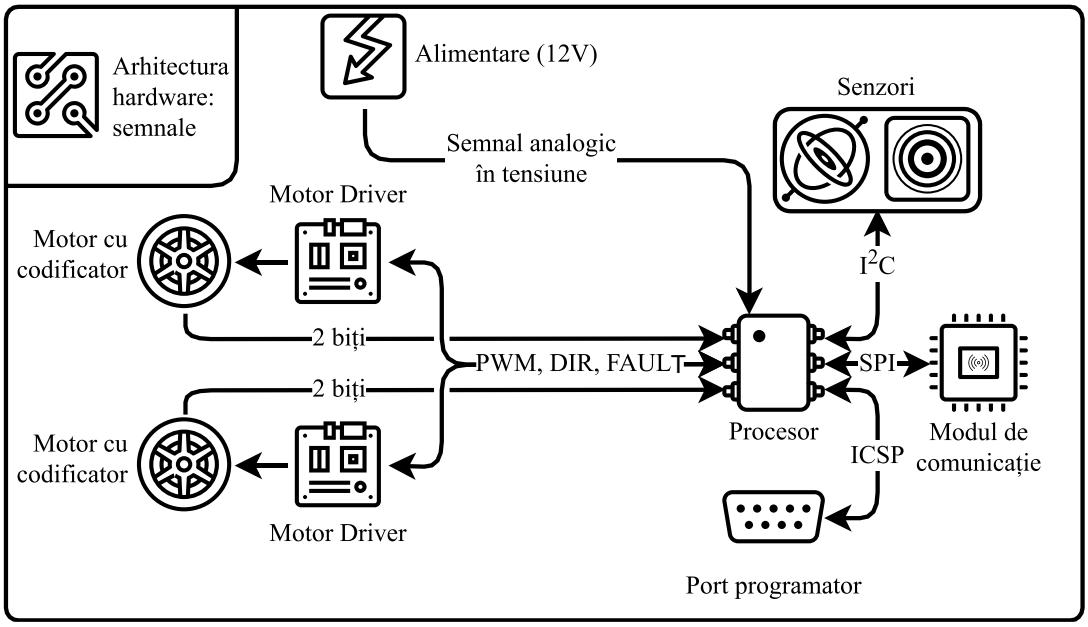


Figura 3.8 – Sinteza semnalelor dintre componentele platformei robotice

3.2. Motoarele și codificatoarele

În genere, există câteva mari categorii de motoare: de curent continuu, de curent alternativ, pas cu pas și servomotoare.

Motoarele de curent alternativ nu sunt viabile pentru proiectul în discuție, din cauza dimensiunilor mari, atât a lor, cât și a driverelor de care au nevoie, și din cauză că întreaga platformă robotică este alimentată la curent continuu și nu posedă capabilitatea de conversie în curent alternativ. Motoarele pas cu pas sunt probabil cel mai ușor de comandat, pot avea dimensiuni reduse, însă la fiecare mișcare acestea produc vibrații, care perturbă senzorul giroscopic și accelerometrul, un efect foarte grav care poate face ca datele citite de la senzori să devină inutilizabile în sensul stabilizării robotului pe o perioadă mai lungă de timp. Din motive evidente, ce țin de proprietățile și capabilitățile servomotoarelor, nici acestea nu reprezintă o variantă viabilă pentru aplicația propusă.

Figura 3.9 – Motorul Pololu

Am ales, deci, pentru platforma robotică micro-motoarele de curent continuu, de mare putere, cu perii, alimentare la 6V și cu reductor metalic cu factor de 100.37:1, produse de Pololu [13, 14]. Varianta folosită are un consum maxim de 1.5A și poate genera un cuplu de 1.6kg·cm. Modelul dispune de o tijă prelungită, astfel că pe fiecare dintre cele două motoare folosite poate fi montat un codificator de cuadratură. Codificatorul înregistrează conform factorului de scalare a motorului o sută de rotații complete pentru fiecare rotație a unei roți fixate pe axul motorului.

Motorul ales are dimensiuni foarte reduse, are un consum mic de energie, are o perioadă de viață lungă și poate dezvolta forțe suficient de mari pentru a deplasa întregul robot în condiții normale. Un avantaj suplimentar al motorului ales este că acesta face parte dintr-o gamă de produse cu proprietăți foarte variate, dar dimensiuni identice. Utilizatorul poate, așadar, înlocui cu minim efort motoarele cu unele alimentate la 12V sau cu unele cu raportul reductorului diferit.

Figura 3.10 – Exemplu de conectare a codificatorului la motorul Pololu

Pentru toate motoarele din gamă, producătorul lor pune la dispoziție și câteva tipuri de codificatoare [15], care nu doar că oferă informații despre starea motoarelor, ci și ușurează conectarea la acestea.

Principiul de funcționare al codificatoarelor folosite se bazează pe efectul Hall. Tijei motorului i se atașează un disc ce conține un mic magnet, iar PCB-ul codificatorului conține doi senzori cu efect Hall. La trecerea magnetului peste senzorii Hall apare o diferență de potențial, care produce trecerea ieșirilor senzorilor în stare activă. Intuitiv, codificatorul oferă două semnale de ieșire, câte unul pentru fiecare senzor Hall. Semnalul de ieșire poate fi privit ca o variabilă de doi biți, ce poate lua patru valori. La o rotație completă a extensiei tijei, și deci a discului atașat de aceasta, ieșirile senzorilor vor trece prin toate cele patru combinații posibile. Considerând și factorul de scalare al motorului, se poate aproxima că o rotație completă a axului principal al motorului va produce schimbarea valorii obținute, concatenând cei doi biți, de 400 de ori. Codificatorul furnizează astfel date despre orientarea tijei motorului la fiecare moment de timp. Adițional, prin folosirea datelor, împreună cu un numărător programat ca timer pe procesor, se pot calcula:

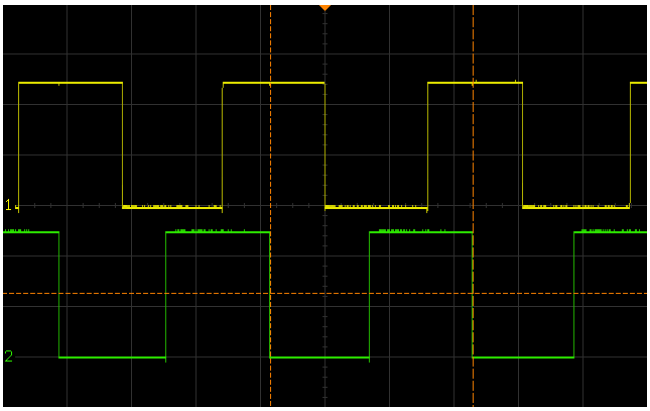


Figura 3.11 – Forma semnalelor de la ieșirea unui codificator când motorul de care este atașat se află în rotație

1. accelerația unghiulară a fiecărui motor
2. viteza de rotație a fiecărui motor
3. poziția relativă a platformei robotice față de punctul de plecare

Robotul dezvoltat poate implementa funcționalitatea de menținere a propriului echilibru și fără aceste traductoare, însă datele obținute de la ele pot fi folosite pentru a rafina algoritmul de reglare, pentru a implementa deplasarea după comanda utilizatorului și, dacă sunt trimise mai departe la o aplicație externă, pentru a implementa controlul unei întregi echipe de roboți.

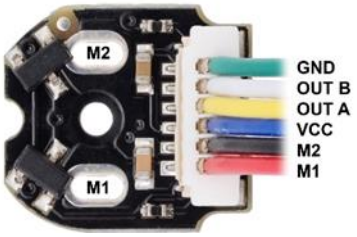


Figura 3.12 – Conexiunile între PCB și codificator

Fixarea mecanică a unui codificator ca în **Figura 3.10** se realizează lipind cu un aliaj de staniu bornele M1 și M2 din **Figura 3.12** de bornele de alimentare ale motorului. În acest fel, nu mai este nevoie de conexiuni separate pentru alimentarea motoarelor, pentru că legătura poate fi făcută chiar prin codificatoare. Ele dispun de un conector JST-SH care oferă încă un avantaj, prin faptul că toate conexiunile necesare cu placa pot fi făcute printr-un singur cablu, de dimensiuni foarte reduse. Cu alte cuvinte, folosind codificatoarele alese reducem

numărul de fire necesare pentru interfațarea cu motoarele și traductoarele lor la două cabluri foarte compacte și asigurăm rezistența mecanică a conexiunilor între placă și ansamblul motor-traductor.

3.3. Șasiul

La fel de important ca și placa dezvoltată, ce face posibilă funcționarea robotului, este și șasiul folosit. El trebuie să acomodeze toate părțile componente ale robotului și într-un fel în care acesta să poată fi folosit pentru a crea un pendul inversat. De aceea am dezvoltat un șasiu personalizat, modular, ce poate fi ușor obținut, folosind o imprimantă 3D, format din trei piese: un corp principal, un suport pentru acumulator și o piesă plastică pentru a limita căderea robotului la o poziție în care placa să fie aproximativ paralelă cu planul pe care se situează robotul. Dimensiunile celor trei piese și o serie de randări din diverse perspective sunt disponibile în **Anexa 3**. La piesele proiectate pentru aplicația curentă se adaugă și carcasele pentru motor, puse la dispoziție de producătorul motoarelor, și care se găsesc în două mărimi.

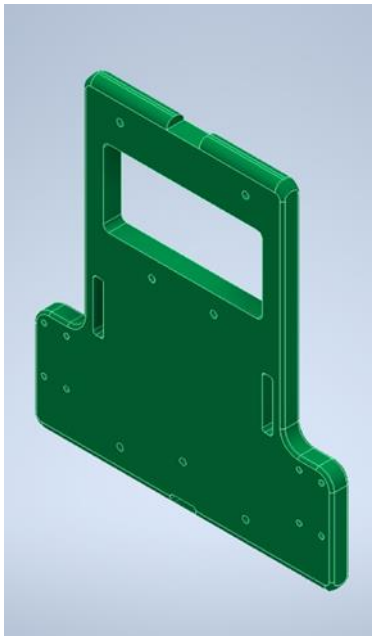


Figura 3.13 – Corpul principal al șasiului personalizat. Proiecție izometrică 3D

Dintre cele trei piese proiectate, doar corpul principal este necesar, celelalte componente putând fi eliminate în funcție de nevoile utilizatorului. Corpul principal, vizibil în **Figura 3.13**, dispune de găuri pentru fixarea plăcii personalizate pe acesta, găuri pentru fixarea suportului de acumulator și găuri pentru fixarea motoarelor, folosind oricare dintre cele două modele de carcase disponibile pentru ele. În partea superioară există un spațiu conceput pentru piesa plastică ce limitează căderea. Pentru a putea organiza cablurile și pentru a le putea restrânge mișcările ce pot afecta performanțele de funcționare a robotului, corpul este prevăzut cu două fante laterale și una inferioară, pentru

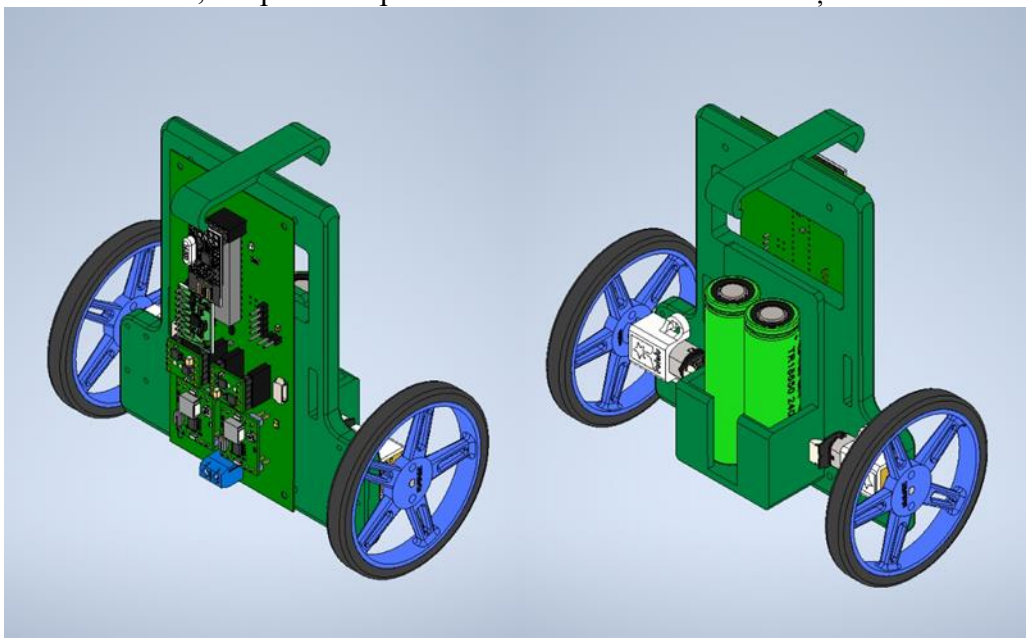


Figura 3.14 – Proiecția izometrică 3D a simulării platformei robotice asamblate

cablurile de legătură cu motoarele, respectiv pentru firele de alimentare. În partea superioară se observă prezența unei secțiuni decupate, concepută astfel pentru a reduce din masa șasiului, fără a îi afecta rezistența structurală în mod catastrofal. Reducerea masei șasiului în partea sa superioară este importantă pentru că ajută la coborârea poziției centrului de greutate al sistemului. Pentru ca robotul să fie considerat a respecta modelul de pendul inversat, centrul său de greutate trebuie să fie deasupra axului roților, însă cu cât acesta este mai aproape de ax, cu atât efortul necesar pentru a stabiliza întregul robot scade. Deci, un centru de greutate apropiat de axa de rotație poate ușura procesul de proiectare al unui algoritm de reglare și poate reduce necesarul de energie al robotului.

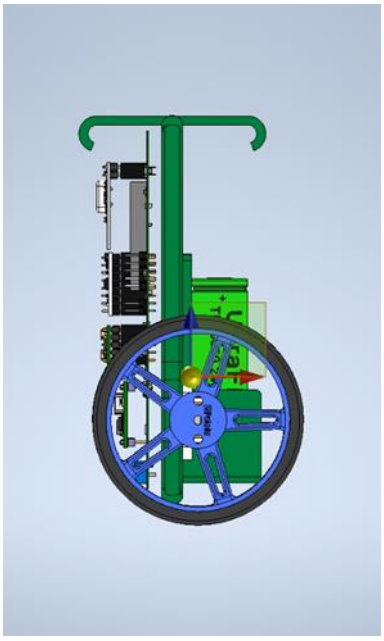


Figura 3.15 – Poziția centrului de greutate al robotului. Vedere din profil

Suportul pentru acumulator poate adăposti două baterii ce respectă standardul de dimensiune 18650, un standard comun pentru acumulatorii de tip Li-Ion. Suportul se poate fixa prin șuruburi pe corpul principal.

Șasiul este conceput pentru cazul în care roțile robotului au un diametru minim de 7cm. Producătorul motoarelor pune la dispoziție și o gamă întreagă de roți, compatibile complet cu gama de motoare folosite, ceea ce înseamnă că utilizatorul poate înlocui fără prea mult efort roțile, cu unele de diametru cel puțin egal cu cel menționat.

3.4. Modelul matematic al platformei robotice

Cunoscând toate piesele ce intră în structura fizică a robotului, utilizatorul poate dezvolta modele matematice prin diverse metode, pe care mai departe le poate folosi pentru a proiecta regulatoare. Pentru aceasta am pus la dispoziție un cumul de parametrii relevanți ai robotului, cu eventuale informații privind felul în care aceștia au fost calculați. În plus, am creat și pus la dispoziție și modelul intrare-stare-ieșire al robotului, ce poate fi direct utilizat pentru a simula comportarea sistemului și pentru a proiecta un regulator pentru acesta.

Pe baza parametrilor dați în **Tabel 3.2** și vizualizați în **Figura 3.16**, poate fi dezvoltat modelul dinamic prin metoda Lagrange [16, 17] sau metoda Kane. [17]

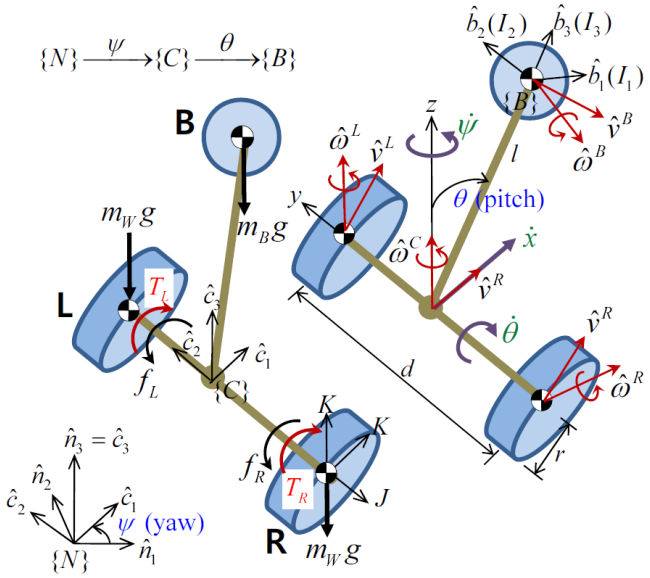


Figura 3.16 – Reprezentarea schematizată a platformei robotice împreună cu parametrii ei

Tabel 3.2 – Parametrii platformei robotice

Parametru	Descriere
$d = 129{,}369\text{mm}$	Distanța dintre centrele de greutate ale celor două roți
$l = 14{,}318\text{mm}$	Distanța între axa roților și centrul de greutate al corpului (lungimea pendulului)
$r = 35\text{mm}$	Raza roților
$m_B = 231{,}6504\text{g}$	Masa corpului pendulului, fără roți
$m_W = 14{,}1748\text{g}$	Masa unei roți
$J = 173{,}6413\text{g} \cdot \text{mm}^2$	Momentul inerțial al unei roți, raportat la axa sa de rotație. Calculat ca $J = m_W \cdot r^2$
$K = 86{,}8207\text{g} \cdot \text{mm}^2$	Momentul inerțial al unei roți, perpendicular pe axa sa de rotație. Calculat ca $K = \frac{1}{2} \cdot m_W \cdot r^2 = \frac{1}{2} \cdot J$
$I_1 = 492944{,}14\text{g} \cdot \text{mm}^2$ $I_2 = 245127{,}512\text{g} \cdot \text{mm}^2$ $I_3 = 312187{,}238\text{g} \cdot \text{mm}^2$	Momentele inerțiale ale corpului pendulului, raportate la centrul său de greutate. Acestea sunt obținute prin descompunerea formei robotului pe fiecare axă în primitive, cu formule cunoscute și însumarea ulterioară a rezultatelor acestora

Pentru a obține modelul intrare-stare-ieșire [18] este nevoie să cunoaștem un număr suplimentar de parametri:

Tabel 3.3 – Parametrii motoarelor alese

Parametru	Descriere
$b \cong 0\text{Nms/rad}$	Coeficientul de vâscozitate al motorului. (frecare) Are valori neglijabile, deci a fost eliminat pentru a ușura calculele.
$R = 4\Omega$	Rezistența internă a motorului. Calculată după legea lui Ohm ca $R = \frac{V}{I_{blocare}}$, cu $V =$ tensiunea nominală de alimentare a motorului și $I_{blocare} =$ curentul consumat de motor la blocare.
$K_e \cong 0.1736\text{Vs/rad}$	Constanta forței electromotoare. Calculată ca $K_e = V/v$, cu $V =$ tensiunea nominală de alimentare a motorului și $v =$ viteza maximă de rotație a motorului.
$K_m \cong 0.1046\text{Nm/A}$	Constanta momentului forței motorului. Calculată ca $K_m = \frac{M_{blocare} \cdot g}{I_{blocare}}$, unde $M_{blocare} =$ momentul forței la blocare (engl. *stall torque*), $g =$ accelerația gravitațională.

Cu acești parametri am obținut, în cele din urmă, modelul intrare-stare-ieșire al sistemului:
$\begin{cases}\dot{\xi} = A\xi + Bu \\ y = C\xi + Du\end{cases}$, cu $\xi = [x \quad \dot{x} \quad \theta \quad \dot{\theta}]^T$

$$A = \begin{bmatrix}0 & 1 & 0 & 0 \\ 0 & \alpha & \beta & -r\alpha \\ 0 & 0 & 0 & 1 \\ 0 & \gamma & \delta & -r\gamma\end{bmatrix}, B = \begin{bmatrix}0 \\ \alpha\varepsilon \\ 0 \\ \gamma\varepsilon\end{bmatrix}, C = \begin{bmatrix}0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1\end{bmatrix}, D = \begin{bmatrix}0 \\ 0\end{bmatrix}$$ și

$$\alpha = \frac{2(Rb - K_e K_m)(m_B l^2 + m_B r l + I_2)}{R(2(I_2 J + J l^2 m_B + I_2 m_W r^2 + l^2 m_B m_W r^2) + I_2 m_B r^2)}$$

$$\beta = \frac{-l^2 m_B^2 g r^2}{I_2(2J + m_B r^2 + 2m_W r^2) + 2J l^2 m_B + 2l^2 m_B m_W r^2}$$

$$\gamma = -\frac{2(Rb - K_e K_m)(2J + m_B r^2 + 2m_W r^2 + l m_B r)}{Rr(2(I_2 J + J l^2 m_B + I_2 m_W r^2 l^2 m_B m_W r^2) + I_2 m_B r^2)}$$

$$\delta = \frac{l m_B g(2J + m_B r^2 + 2m_W r^2)}{2I_2 J + 2J l^2 m_B + I_2 m_B r^2 + 2I_2 m_W r^2 + 2l^2 m_B m_W r^2}$$

$$\varepsilon = \frac{K_m r}{Rb - K_e K_m}$$

Modelul rezultat:

$$A = \begin{bmatrix}0 & 1 & 0 & 0 \\ 0 & -0{,}6197 & -76{,}463 & 0{,}0217 \\ 0 & 0 & 0 & 1 \\ 0 & 65{,}2209 & 5994{,}5 & -2{,}2827\end{bmatrix}, B = \begin{bmatrix}0 \\ 0{,}1249 \\ 0 \\ -13{,}1576\end{bmatrix}, C = \begin{bmatrix}0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1\end{bmatrix}, D = \begin{bmatrix}0 \\ 0\end{bmatrix}$$

Urmărind matricele de controlabilitate și de observabilitate, ajungem la concluzia că sistemul este complet controlabil, însă doar parțial observabil.

$W_c = [B \quad AB \quad A^2B \quad A^3B]$, $\text{rank}(W_c) = 4 \Rightarrow$ Sistemul este complet controlabil.

$W_o = [C \quad CA \quad CA^2 \quad CA^3]$, $\text{rank}(W_o) = 1 \Rightarrow$ Sistemul este parțial observabil.

3.5. Alte aspecte hardware ale aplicației

Pe lângă platforma robotică, aplicația include o placă de dezvoltare Arduino Uno, dotată cu un modul de emisie-recepție nRF24L01+. Placa Arduino este conectată la un calculator personal, care, la rândul său, preia comenzi de la un controller Microsoft Xbox One Wireless sau oricare alt model compatibil cu XInput. Mai multe detalii despre aceste resurse vor fi date în capitolul următor, ce detaliază asupra aspectelor software ale aplicației.

Capitolul 4. Structura software a aplicației

După cum am mai precizat pe parcursul lucrării, aplicația are ca piesă centrală platforma robotică, însă cuprinde și o serie de programe dezvoltate pentru resursele hardware auxiliare, enunțate în ultimul paragraf al capitolului anterior. **Figura 4.1** prezintă schematizat legăturile dintre echipamentele hardware și componentele software prin care aceste legături se stabilesc.

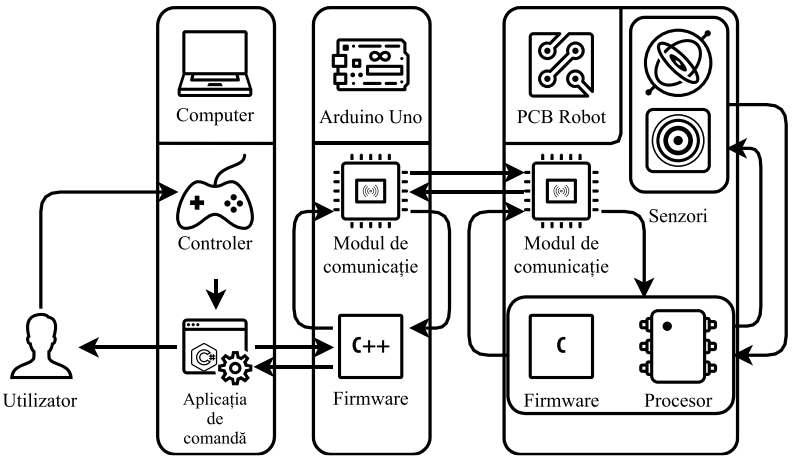


Figura 4.1 – Diagrama relațiilor dintre componentele aplicației

În ansamblu, structura software a aplicației este formată din patru programe. Dintre acestea, unul este firmware-ul modulului de comunicație, cu care vine preprogramat și care nu face obiectul discuției. Pe celelalte trei programe le-am proiectat și dezvoltat special pentru aplicația curentă.

Pe scurt, aplicația de comandă, ce rulează pe un calculator personal, reprezintă interfața de control pe care o are utilizatorul. Prin aceasta, utilizatorul poate trimite comenzi către platforma robotică și poate primi înapoi informații despre starea platformei și parametrii săi relevanți. Pentru că un calculator personal nu se poate interfața direct cu un modul nRF24L01+, am ales să folosesc o placă Arduino, dotată cu un astfel de modul și al cărei rol este de a facilita schimbul de mesaje între aplicația de pe computerul personal și robot. Nu în ultimul rând, firmware-ul robotului este conceput pentru a putea primi comenzi externe, pentru a putea returna date despre propria stare și pentru a implementa efectiv funcționalitățile robotului.

În cele ce urmează, voi detalia asupra structurii și rolului fiecărui program menționat, dar și asupra programelor auxiliare dezvoltate. Acestea din urmă au ajutat fie în proiectarea, fie în dezvoltarea structurii hardware și a celor trei programe principale amintite.

4.1. Firmware-ul platformei robotice

Dintre cele trei programe, cu siguranță cel mai important și complex este acesta. Firmware-ul disponibil pentru microcontrolerul dsPIC este astfel conceput încât pune la dispoziție toate funcționalitățile necesare pentru ca platforma să fie folosită în mod independent și extinde aceste facilități cu unele ce permit comunicarea cu aplicații externe. Programul este scris în limbajul C și compilat cu C30 [19], un compilator mai vechi al companiei Microchip. Deși compania pune la dispoziție un compilator mai nou, numit xc16 și compatibil cu întreaga gamă de procesoare numerice de semnal dsPIC, am optat pentru compilatorul mai vechi deoarece acesta este mai stabil și include o serie de biblioteci care fac mai ușoară folosirea perifericelor de care dispune procesorul. Compilatorul xc16 oferă și el această ultimă facilitate, însă numai pentru un număr restrâns de modele PIC și dsPIC, din care cel ales nu face parte.

Firmware-ul este compus dintr-un număr de module:

1. Un modul folosit pentru inițializarea microcontrolerului, ce monitorizează starea acestuia și a bateriei și oprește execuția programului, în cazul apariției unei erori sau în cazul bateriei scăzute.
2. Un modul de interfațare cu motoarele și codificatoarele.
3. Un modul de interfațare cu circuitul nRF24L01+.
4. Un modul de interfațare cu circuitul MPU6050.

Deoarece funcționarea firmware-ului este complet dependentă de structura hardware, pe parcursul prezentării modulelor se vor mai găsi și detalii hardware, relevante pentru a înțelege efectul executării codului descris.

4.1.1. Inițializarea microcontrolerului și operații auxiliare

Atunci când este alimentat, microcontrolerul trece printr-o perioadă de inițializare în care nu poate primi comenzi din partea aplicațiilor externe. În acest interval este fixată sursa de ceas a procesorului, sunt inițializate perifericele și controlerul de întreruperi externe.

Pentru aplicația curentă, am dorit ca microprocesorul să funcționeze cât mai aproape de frecvența sa maximă de 40MIPS (engl. *Million Instructions per second*). De aceea am folosit ca sursă de ceas oscilatorul FRC (engl. *Fast Resistor-Capacitor*) a microcontrolerului, care furnizează o frecvență de 7,37MHz, împreună cu o buclă PLL (engl. *Phase-locked loop*). Frecvența de operare care rezultă are valoarea:

$$F_{CY} = \frac{F_{OSC}}{2} = \frac{1}{2} F_{IN} \frac{M}{N_1 N_2} = \frac{1}{2} F_{FRC} \frac{M}{N_1 N_2} = \frac{1}{2} \cdot 7{,}37\text{MHz} \cdot \frac{65}{3 \cdot 2} = 39{,}936\text{MHz}$$

Am calculat valorile parametrilor M, N_1, N_2 folosind un script MATLAB disponibil în **Anexa 4.1**. Folosind scriptul pe care l-am creat, un utilizator trebuie doar să introducă frecvența de funcționare dorită și i se returnează parametrii PLL ce pot asigura cea mai apropiată frecvență, atunci când este folosit oscilatorul FRC. Mai departe, acest script poate fi folosit și pentru a ajuta la stabilirea ratei de transfer a perifericelor I^2C.

Utilizatorului i se pun la dispoziție o serie de facilități pentru a selecta o sursă diferită de ceas. În fișierul **defines.h** există câteva definiții pentru a stabili sursa de ceas și câteva definiții pentru a stabili parametrii buclei PLL în cazul în care aceasta este folosită:

```
    #define FRC 7370000UL   //  Fast RC Ocsillator - 7.37MHz
    #define LPRC 32768UL    //  Low-Power RC Oscillator - 32.768Khz

    //  Use PLLcalc Matlab Script to calculate the best PLL Parameters
    #define PLL_M   65UL    //  PLL Multiplier real value
    #define PLL_N1  3UL     //  PLL Divider 1 real value
    #define PLL_N2_2        //  PLL Divider 2 set on 1/2 factor
//    #define PLL_N2_4      //  PLL Divider 2 set on 1/4 factor
//    #define PLL_N2_8      //  PLL Divider 2 set on 1/8 factor

    //  Defines for actual value of PLL N2 divider bits depending on selected c
onfig
    #ifdef PLL_N2_2
    #define PLL_N2  0x00    //  PLL Divider 2 register value
    #elif defined(PLL_N2_4)
```

```
    #define PLL_N2  0x01    //  PLL Divider 2 register value
    #else
    #define PLL_N2  0x11    //  PLL Divider 2 register value
    #endif

    //  Select appropriate oscillator. Uncomment to select

//    #define USE_FRC         //  Use Fast RC Oscillator
    #define USE_FRCPLL      //  Use Fast RC Oscillator with PLL
//    #define USE_LPRC        //  Use Low-Power RC Oscillator

    //  Define Oscillator frequency according to selected clock source

    #ifdef USE_FRC
    #define F_OSC FRC                           //  Oscillator frequency
    #elif defined(USE_FRCPLL)
    #ifdef PLL_N2_2
    #define F_OSC ((FRC/PLL_N1)*PLL_M)/(2UL)    //  Oscillator frequency
    #elif defined(PLL_N2_4)
    #define F_OSC ((FRC/PLL_N1)*PLL_M)/(4UL)    //  Oscillator frequency
    #else
    #define F_OSC ((FRC/PLL_N1)*PLL_M)/(8UL)    //  Oscillator frequency
    #endif
    #elif defined(USE_LPRC)
    #define F_OSC LPRC
    #else
    #define F_OSC 0UL
    #endif

    //  Define frequencies

    #define FCY     F_OSC/2         //  Device Operating Frequency = Selected O
scillator Frequency / 2
```

Secvența de cod ilustrată determină apelul funcției de inițializare a buclei PLL atunci când ea trebuie folosită, oferă parametrii pentru setarea acesteia și oferă o serie de valori ce sunt utilizate mai departe pentru a seta corespunzător frecvențele de funcționare a perifericelor. Totodată, folosind valorile definite în această secvență, se garantează satisfacerea unor cerințe de funcționare a modulelor externe microcontrolerului, de exemplu prin implementarea corectă a unor timpi de așteptare între anumite operații. Prin simpla schimbare a definiției active din *USE_FRCPLL* în *USE_LPRC,* utilizatorul poate scădea frecvența de funcționare a procesorului la 16,384KHz. Dacă aplicația în care platforma este folosită se poate descurca la această frecvență, prin selectarea sursei LPRC se poate economisi energie și deci se poate lărgi perioada de autonomie de funcționare a robotului.

Inițializarea perifericelor PWM, I²C și SPI va fi discutată mai târziu, împreună cu respectivele componente pentru care sunt folosite. Pe lângă perifericele amintite, mai este inițializat convertorul analog-digital, cu un singur canal activ și conversie pe 12 biți. Prin acesta se poate obține de pe pinul AD0 valoarea tensiunii de la acumulator. Robotul este proiectat pentru a fi alimentat la tensiuni de până la 12V, însă pinii microcontrolerului suportă doar tensiuni de până la 3.6V, cu excepția unui număr mic de pini ce suportă 5V. Pentru că referința de tensiune a convertorului este alimentarea procesorului, pe care am fixat-o în jurul valorii de 3.3V, legătura dintre acumulator și convertor se face printr-un divizor de tensiune. După formula $V_{AN0} = \frac{R_4}{R_3+R_4} V_{CC}$, tensiunea ajunsă pe pinul convertorului este un sfert din tensiunea acumulatorului. Cunoscând aceste date, se poate afla tensiunea de alimentare și se poate opri execuția programului atunci când bateria este descărcată, pentru a o proteja. În aplicația curentă am folosit un acumulator care, în condiții normale, furnizează 7.4V, așadar, atunci când tensiunea scade sub 7V, procesorul dezactivează perifericele, modulul de comunicație, driverele de motoare și senzorii și apoi intră într-un mod de consum redus, în care oscilatorul FRC nu mai este funcțional.

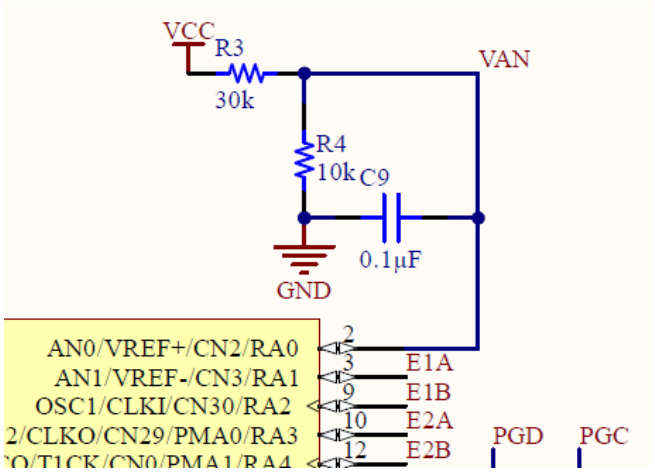


Figura 4.2 – Extras din schema cablajului imprimat. Vedere asupra divizorului de tensiune conectat la convertorul ADC al microcontrolerului

4.1.2. Controlul motoarelor și al codificatoarelor

Motoarele de curent continuu ale robotului sunt controlate în mod indirect de microcontroler prin driverele specializate descrise în capitolul anterior. Acestea alimentează motoarele corespunzător, în funcție de o serie de semnale primite de la procesor.

În primul rând, un astfel de driver este activ doar atât timp cât semnalul de intrare de pe pinul EN# se află în „0” logic. Cât timp semnalul este setat pe „1” logic, indiferent de valorile celorlalte intrări, driverul nu va comanda motoarele. Această funcționalitate este folosită atunci când microcontrolerul detectează descărcarea bateriei.

Sensul de rotație al motorului este stabilit prin semnalul DIR, iar viteza de rotație este stabilită printr-un semnal PWM, generat de microcontroler. Pentru aceasta, microcontrolerul folosește un pin de uz general și un pin de ieșire al perifericului PWM1. La alimentare, în microcontroler este apelată funcția de inițializare a perifericului, ce dispune de trei canale, cu un total de șase ieșiri, din care doar două sunt active, restul fiind folosite pentru alte funcții. Perifericul ce se ocupă de generarea semnalului PWM este setat la o frecvență de funcționare egală cu cea a procesorului. Schimbarea setărilor modulului și forțarea manuală a ieșirilor de semnal se poate produce doar la începutul unei noi perioade a semnalului.

Unul din pinii reconfigurabili ai microcontrolerului este folosit pentru a primi semnalul FAULT1. Atunci când motorul 1 se supraîncălzește sau suferă o altă defecțiune, sau atunci când la driverul său apare o problemă, semnalul FAULT al driverului trece din „1” logic în „0” logic, ceea ce determină aprinderea unui LED roșu cât timp defecțiunea nu este rezolvată, oprirea

generării de semnal PWM pentru amândouă motoarele și apariția unei cereri de întrerupere pe microcontroler. Utilizatorul poate folosi această întrerupere pentru a semnala către o aplicație externă apariția defecțiunii. Pentru motorul rămas, aceeași funcționalitate este implementată integral prin software.

Pentru a facilita comanda motoarelor, în codul sursă pus la dispoziție pentru robot este implementată o funcție ce ajustează comanda motoarelor în funcție de câțiva parametri de intrare.

```
// In defines.h

    #define MOTOR_RATIO 400         //  Motor encoder-shaft rotation ratio
    #define FAULT1  _RB2            //  PWM1 FAULTA pin
    #define FAULT2  _RB3            //  PWM1 FAULTB pin
    #define DIR1    _RB15           //  Motor 1 direction pin
    #define DIR2    _RB13           //  Motor 2 direction pin
    #define DC_CAP  100             //  Duty cycle cap

// In motors.c

/**
 * Use this function to adjust the parameters of one of the motors.
 *  PWM must be active and FAULT signals must not be active for the
 *  effects to be visible.
 * @param motor         <code>false</code> for motor 1, <code>true</code> for m
otor 2
 * @param duty_cycle    PWM duty cycle for selected motor valued from 0 - DC_CA
P (capped at DC_CAP)
 * @param dir           <code>true</code> for forward rotation, <code>false</co
de> for backward rotation
 */
void MotorAdjust(bool motor, uint8_t duty_cycle, bool dir)
{
    duty_cycle = duty_cycle > DC_CAP ? DC_CAP : duty_cycle;

    switch (motor)
    {
        case false:     //  Motor 1
            P1DC1 = P1TPERbits.PTPER * DC_CAP / duty_cycle;
            DIR1 = dir;
            motor1.dir = dir;
            break;
        default:        //  Motor 2
            P1DC2 = P1TPERbits.PTPER * DC_CAP / duty_cycle;
            DIR2 = -dir;
            motor2.dir = dir;
```

```
            break;
    }
}
```

Viteza maximă de rotație a motoarelor este de 330rpm, care corespunde cu o alimentare constantă de 6V, și în consecință cu un semnal PWM cu factor de umplere 100%. Pentru a determina parametrul *duty_cycle* al funcției, potrivit pentru viteza pe care dorim să o comandăm, putem considera evoluția vitezei motorului ca fiind o funcție liniară dependentă de tensiunea de alimentare:

$$v = v_{max} \cdot \frac{V}{V_{nominal}}, V = V_{nominal} \cdot \frac{T_{dc\%}}{100} \Rightarrow v = v_{max} \cdot \frac{T_{dc\%}}{100}$$

unde $v =$ viteza unghiulară a motorului, $v_{max} =$ viteza unghiulară maximă a motorului, $V =$ tensiunea de alimentare pe motor, $V_{nominal} =$ tensiunea nominală de alimentare a motorului, $T_{dc\%} =$ factorul de umplere al semnalului PWM exprimat în procente.

Deci, pentru aplicația curentă:

$$v = 330\text{rpm} \cdot \frac{T_{dc\%}}{100} \Rightarrow T_{dc\%} = \frac{10v}{33\text{rpm}}$$

Pentru că parametrul *duty_cycle* nu poate lua valori fracționare, trebuie aproximată valoarea obținută la cel mai apropiat întreg:

$$duty\_cycle = \text{round}(T_{dc\%})$$

Dacă motoarele sunt comandate prin apelarea funcției puse la dispoziție, informațiile extrase de la codificatoare sunt disponibile utilizatorului prin intermediul unor structuri globale, în care se găsesc date despre:

- numărul de rotații complete al roților atașate fiecărui motor
- numărul de rotații parțiale al fiecărei roți (în aplicația curentă, o rotație completă a axului de la ieșirea reductorului este echivalentă cu aprox. 400 de rotații parțiale înregistrate de codificator)
- direcția de rotație a motorului
- factorul de umplere al semnalului PWM ce comandă motorul

Datele de la codificatoare sunt actualizate în așa-numita întrerupere de notificare a schimbării valorii de intrare (engl. *Input Change Notification*) de care dispune microcontrolerul. Aceasta este o întrerupere ce poate apărea de la oricare din pinii CNx setați corespunzător, iar cererea este ridicată oricând valoarea de pe unul din pinii vizați se schimbă. În aceeași întrerupere este tratată și apariția unei defecțiuni pe motorul 2.

4.1.3. Interfața cu modulul de emisie-recepție

Modulul ales este unul foarte popular în mediul industrial, dar mai ales în proiectele hobbyiștilor, iar de regulă acest fapt face ca diverse exemple de utilizare a componentei împreună cu plăcile Arduino să fie la un click distanță. Pentru modulele nRF24L01 și nRF24L01+ există o bibliotecă de cod, „RF24” [20], bine documentată, ce implementează într-un mod comprehensibil o interfață cu plăcile discutate. Având ca sursă de inspirație respectiva bibliotecă și urmărind manualele de utilizare a microcontrolerului [12] și a circuitului nRF [9], am creat o bibliotecă de cod proprie, ce oferă funcționalități similare.

Pentru a putea înțelege funcționarea circuitului nRF și logica secvențelor de cod din firmware dedicate acestuia, trebuie înțeleasă diagrama automatului de stare a modulului,

disponibilă în **Anexa 5**, și cele două protocoale de comunicație pe care acesta le folosește: SPI și Enhanced ShockBurst™.

4.1.3.1. SPI

SPI reprezintă, de fapt, nivelul fizic al protocolului de comunicație dintre microcontroler și nRF. În general, el este folosit între un dispozitiv *master* și unul sau mai multe dispozitive *slave*. Protocolul SPI necesită cel puțin patru conexiuni fizice pentru următoarele semnale:

- SCK – Semnal de ceas, transmis de către dispozitivul *master* tuturor dispozitivelor *slave*. Comunicația prin SPI este sincronă cu semnalul SCK.
- MISO (engl. *Master Input Slave Output*) – Semnal prin care dispozitivul *master* primește date de la dispozitivele *slave*.
- MOSI (engl. *Master Output Slave Input*) – Semnal prin care dispozitivul *master* trimite mesaje către dispozitivele *slave*.
- SS# (engl. *Slave Select*) sau CSN (engl. *Chip Select Not*) – Semnal prin care dispozitivul *master* poate selecta dispozitivul *slave* cu care dorește să comunice. Tot prin acest semnal se marchează începutul și finalul unei tranzacții.

Într-un sistem cu mai multe dispozitive *slave*, protocolul SPI impune existența câte unui semnal SS# distinct între *master* și fiecare dintre dispozitivele *slave*. Transferul de date este bidirecțional, efectuat prin două fire, ceea ce înseamnă că într-o tranzacție, în același timp în care *master*-ul trimite date prin MOSI, dispozitivul *slave* îi trimite înapoi date prin MISO. De aceea, atunci când unul dintre dispozitive doar așteaptă să primească date, el va trimite, de fapt, în gol un număr de octeți egal cu cel pe care încearcă să îi citească. Pentru că dispozitivul *slave* trebuie selectat de către *master* pentru a putea comunica, devine evident faptul că un schimb de date nu poate fi inițiat decât de *master*. În cazul punctual al platformei robotice, microcontrolerul este dispozitivul *master*.

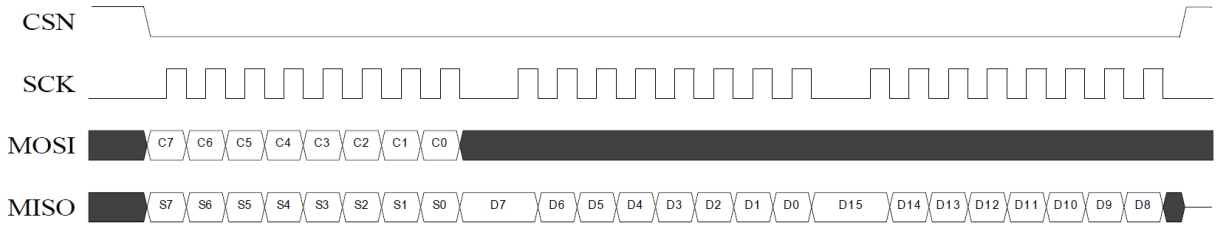


Figura 4.3 – Operație de citire prin SPI

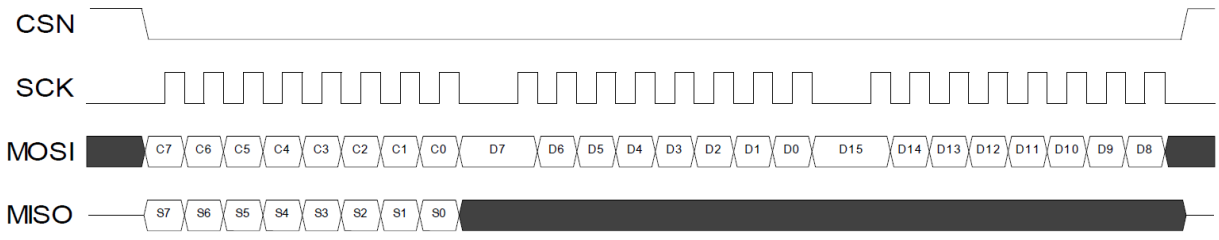


Figura 4.4 – Operație de scriere prin SPI

Utilizarea protocolului SPI de către circuitul nRF permite ca în timpul fiecărei tranzacții între acesta și microcontroler, la finalizarea trimiterii biților de comandă, în registrul perifericului de SPI al procesorului să fie disponibilă valoarea registrului intern STATUS a modulului de comunicație.

4.1.3.2. Enhanced ShockBurst™

Protocolul vizat este cel folosit pentru schimbul de date între modulele de comunicație nRF. Un mesaj construit după regulile sale este format dintr-un preambul, între 3-5o de adresă, un câmp de control de 9 biți, un pachet de date cu lungime de 0-32o, și un CRC (engl. *Cyclic Redundancy Code*) de 8 sau 16 biți. Câmpul de control conține lungimea în octeți a pachetului de date, doi biți ce servesc pentru identificarea pachetului și un bit care, dacă este setat, marchează faptul că pentru mesajul în cauză nu este nevoie ca receptorul să trimită o confirmare. Detalierea asupra caracteristicilor acestui protocol nu importă în contextul curent, dar din descrierea făcută reiese o serie de informații despre capabilitățile de comunicare a platformei robotice:

1. Schimbul de informații dintre robot și o aplicație externă trebuie organizat în mesaje cu până la 32o de date.
2. Mesajele pot avea lungimi fixe sau lungimi variabile.
3. Poate fi eliminată nevoia de mesaje de confirmare
4. Circuitul receptor va semnala primirea unui mesaj nou, doar dacă acesta i se adresează și dacă se potrivește cu CRC-ul de care este însoțit.

4.1.3.3. Protocolul implementat în aplicația curentă

Microcontrolerul comunică prin perifericul său SPI1 cu modulul de emisie-recepție. De aceea, în secvența de inițializare a microcontrolerului este apelată o funcție de inițializare a perifericului, iar ulterior este apelată și funcția de inițializare a circuitului nRF. Aceasta suprascrie valorile unor registre interne ale circuitului nRF, operație disponibilă doar atât timp cât circuitul nu este activ ca receptor sau emițător. În funcția de inițializare se stabilește ca circuitul nRF să opereze ca receptor, dar această setare poate fi schimbată fie direct din timpul inițializării, fie prin apelarea ulterioară a unei funcții puse la dispoziție. Dacă modulul este pornit, atunci prin trecerea semnalului CE în „1” logic se pornește fie emisia, fie recepția, în funcție de modul de funcționare selectat.

În aplicația curentă, modulul de comunicație este folosit ca receptor, acceptă mesaje de dimensiuni variabile și răspunde automat cu mesaje de confirmare ce pot conține încărcături utile de date, cu lungimi variabile între 0 și 32 octeți. Când recepționează un mesaj nou, modulul este setat să producă o cerere de întrerupere prin pinul IRQ. Cererea este preluată de microcontroler prin INT0, căreia îi este atribuită prioritatea maximă. Microcontrolerul citește mesajul primit, comandă golirea structurilor FIFO (engl. *First in, first out*) interne ale modulului și scrie un nou mesaj în coada pentru transmisie. La primirea următorului mesaj, modulul nRF va răspunde cu o confirmare ce conține și datele proaspăt stocate în coada de transmisie.

În acest fel, aplicația externă poate trimite comenzi platformei robotice, poate deduce dacă robotul funcționează și poate primi informații despre starea internă a robotului și valorile citite de la senzori, fără a folosi o structură complicată de mesaje.

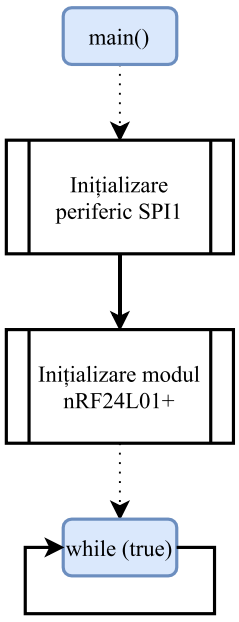


Figura 4.5 – Detaliu din secvența de inițializare a robotului

4.1.3.4. Alte facilități

În vederea satisfacerii cerințelor unor aplicații mai complexe, am implementat biblioteca de cod cu structuri de date ce permit recepționarea datelor de la mai multe module nRF. Cu alte cuvinte, roboții sunt proiectați pentru a putea fi folosiți în echipe de până la 7. Într-o astfel de echipă, fiecare robot dispune de câte un canal de comunicație direct cu fiecare dintre colegii săi. Mai multe detalii despre aceasta și restul funcționalităților sunt disponibile în manualul circuitului nRF, dar este important de reținut că biblioteca implementată este pregătită pentru orice scenariu de funcționare dorit și că aceasta ușurează cu mult procesul de dezvoltare al aplicațiilor.

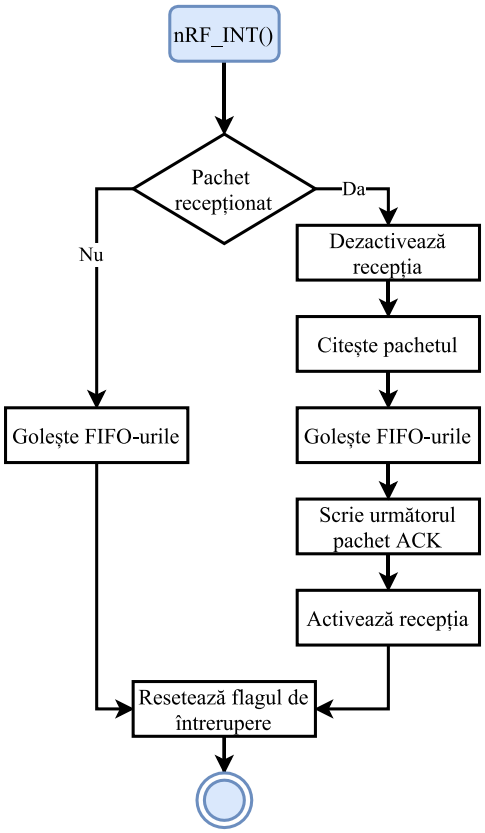


Figura 4.6 – Secvența de citire a mesajelor primite prin circuitul nRF24L01+

4.1.4. Interfața cu senzorii inerțiali

De departe cea mai complexă parte a firmware-ului sunt bibliotecile create pentru interfațarea cu circuitul MPU6050.

Pentru a crea interfața am folosit:

1. documentația circuitului MPU [8]
2. documentația procesorului digital de mișcare cu care acesta este echipat [21]
3. notele de aplicație puse la dispoziție de producător [22, 21]
4. bibliotecile de cod aferente puse la dispoziție de producător
5. ghidul de portare al bibliotecilor menționate [23]
6. documentația, codul pentru Arduino și codul pentru dsPIC30 al bibliotecii I2CDevLib [24]
7. manualul microcontrolerului dsPIC33FJ128MC802 [12]
8. manualul perifericului I^2C al microcontrolerului folosit [25]

În continuare voi prezenta comunicarea procesor – MPU și protocolul I^2C, capabilitățile senzoriale ale circuitului MPU, funcțiile procesorului digital de mișcare și modul de interpretare al datelor citite de pe senzori, în legătură cu structura interfeței software.

4.1.4.1. Protocolul I^2C. Comunicarea procesor – MPU

Protocolul I^2C este unul special conceput pentru comunicarea între circuite integrate, aflate, de regulă, pe același PCB. În mod normal, el este folosit pentru a atașa dispozitive cu viteze de funcționare reduse microcontrolerelor sau microprocesoarelor. Protocolul I^2C este preferat câteodată pentru că acesta poate facilita crearea unei veritabile rețele de dispozitive, folosind o magistrală formată din numai două fire: pentru un semnal de ceas și pentru un semnal de date.

Pentru că dispune de un singur semnal de date, protocolul este în mod inerent unul cel mult half-duplex. Într-un scenariu oarecare, magistrala I^2C poate fi folosită de o rețea formată din unul sau mai multe dispozitive *master* și unul sau mai multe dispozitive *slave*. În cazul particular al aplicației curente, rețeaua este formată doar din microcontroler ca *master* și modulul de senzori inerțiali ca *slave*.

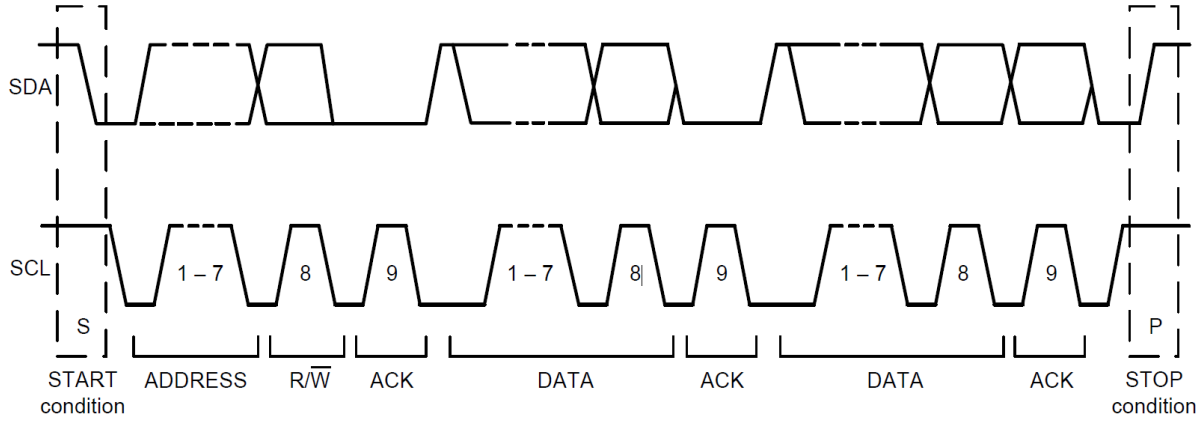


Figura 4.7 – Formele semnalelor într-o tranzacție pe magistrala I²C

Un transfer pe magistrala I²C este inițiat de *master* prin generarea unei condiții de start. *Masterul* începe să genereze un semnal de ceas pe linia SCL, trimite pe linia SDA prin următorii 7 biți adresa dispozitivului cu care dorește să comunice și încheie octetul cu un bit ce marchează dacă dispozitivul adresat urmează să primească date sau dacă acestuia îi sunt cerute date. Dispozitivul *slave* are la dispoziție o perioadă de ceas pentru a ridica linia SDA pe „1" logic pentru a marca recepționarea cu succes a octetului. *Masterul* mai trimite un octet de date și verifică din nou apariția unei confirmări din partea *slave*-ului. *Masterul* poate trimite un număr oarecare de octeți, iar atunci când dorește terminarea tranzacției generează o condiție de stop. Dacă operația este una de citire, atunci dispozitivul *master* generează mesaje de confirmare pentru toți octeții primiți, mai puțin pentru cel cu care dorește ca tranzacția să se încheie. Astfel, dispozitivul *slave* încetează să mai transmită octeți după acesta și *masterul* generează o condiție de stop.

În cazul particular al comunicării între MPU și un procesor, există patru tipuri de tranzacții:

- Citirea unui octet

Tabel 4.1 -Citirea valorii unui registru de 8 biți ai senzorului MPU6050

µC	S	AD+W		RA		RS	AD+R			NACK	P
MPU	-		ACK		ACK			ACK	D		

- Citirea mai multor octeți

Tabel 4.2 – Citirea unui număr de octeți din registrele senzorului MPU6050

µC	S	AD +W		RA		RS	AD +R			ACK		...		NACK	P
MPU	-		ACK		ACK			ACK	D		D		D		

- Scrierea unui octet

Tabel 4.3 – Scrierea unui octet într-un registru intern al MPU6050

µC	S	AD+W		RA		D		P
MPU	-		ACK		ACK		ACK	

- Scrierea mai multor octeți

Tabel 4.4 – Scrierea mai multor octeți în memoria senzorului MPU6050

µC	S	AD+W		RA		D		D	...	D		P
MPU	-		ACK		ACK		ACK				ACK	

Tabel 4.5 – Legenda simbolurilor folosite în Tabelele 4.1 – 4.4

Abreviere	**Descriere**
S	Condiție de start: SDA trece din „1” în „0” cât timp SCL este în „1”
AD	Adresa I²C a modulului MPU
W	Marcajul operației de scriere („0”)
R	Marcajul operației de citire („1”)
ACK	Marcaj de confirmare a recepției („1”)
NACK	Lipsa confirmării recepției („0”)
RA	Adresa unui registru intern al MPU (1 octet)
D	Octet de date
P	Condiție de stop: SDA trece din „0” în „1” cât timp SCL este în „1”

În procesul de implementare a protocoalelor I²C și a comunicației cu senzorul MPU, pentru a putea testa corectitudinea codului scris, am folosit un osciloscop cu decodor de I²C și un analizator logic de semnal produs de Saleae. Am implementat în limbajul C# o aplicație nativă pentru sistemul de operare Microsoft Windows, ce conține o hartă a registrelor interne a MPU6050 și care folosind datele culese de analizatorul logic, poate traduce schimbul de mesaje prin I²C dintre un dispozitiv oarecare și circuitul MPU6050.

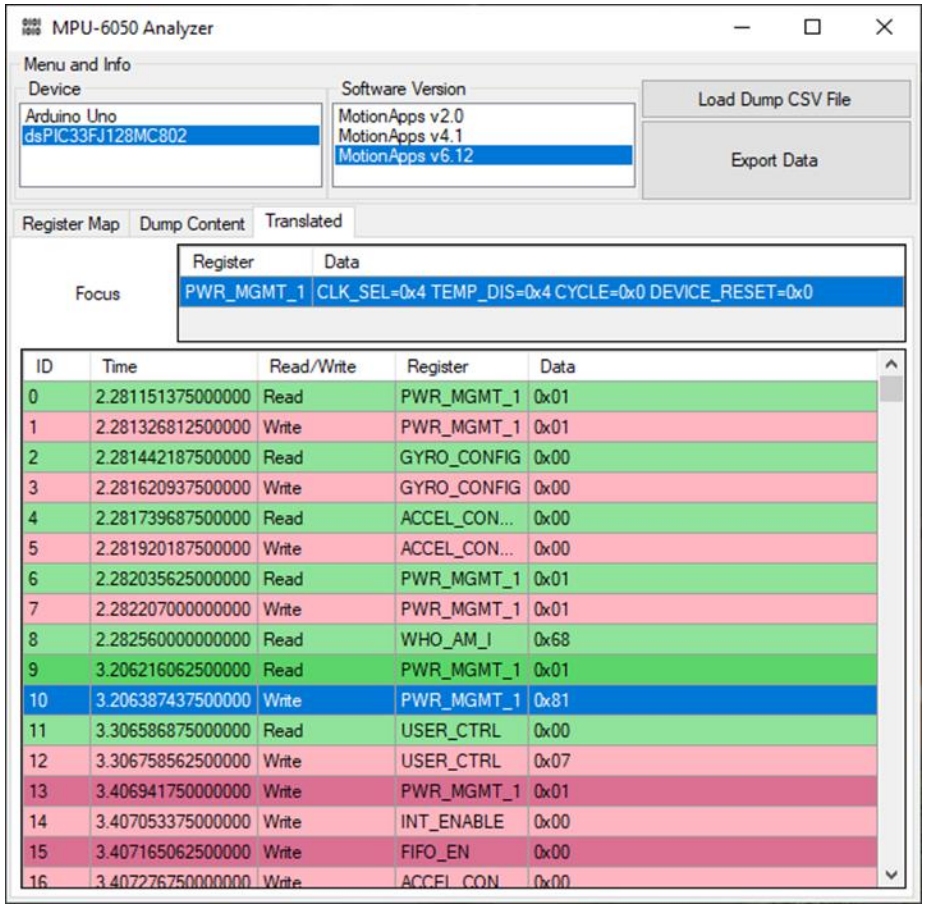


Figura 4.8 – Programul auxiliar dezvoltat pentru depanarea protocolului I²C și a comunicării cu circuitul MPU6050

Revenind la platforma robotică, modulul MPU6050 dispune de două interfețe I²C, din care una este folosită pentru a conecta senzori adiționali la MPU. Legătura fizică între unitatea de procesare a

microcontrolerului și circuitul MPU se face prin interfața I²C principală a MPU-ului și perifericul I2C1 al procesorului. În secvența de inițializare a microcontrolerului este apelată funcția ce setează și pornește perifericul I2C1. Pentru a comunica folosind perifericul, am pus la dispoziție două biblioteci de cod ce implementează protocolul I²C. În aplicația demonstrativă este folosită o librărie mai complexă, formată din fișierele **i2c_header.h** și **i2c_file.c**, adaptată după diverse implementări observate în proiecte ce folosesc gama de procesoare dsPIC33. Am conceput, însă, independent de orice sursă de inspirație, o bibliotecă alternativă, ce utilizează doar funcțiile predefinite ale compilatorului și implementează două funcții, una pentru scriere, una pentru citire, generice. Biblioteca formată din fișierele **dsPIC33_i2c.h** și **dsPIC33_i2c.c** este perfect funcțională și poate fi utilizată ca atare pe orice procesor din gama dsPIC33, și cu minime modificări pe orice procesor de 16biți de tipul PIC/dsPIC.

Modulul MPU poate comunica prin I²C la două viteze:

- 100KHz (mod normal de funcționare)
- 400KHz (modul rapid de funcționare)

Bineînțeles că recomandată este funcționarea la 400KHz, care nu doar că duce la reducerea timpilor cât microcontrolerul se ocupă de comunicarea cu modulul de senzori, dar permite și utilizarea procesorului digital de mișcare integrat în capsula MPU. Pentru a comunica la exact 400KHz, microcontrolerul are nevoie de o sursă de ceas externă, de regulă sub forma unui cristal de cuarț, cu o frecvență egală cu un multiplu al frecvenței căutate. Folosind oscilatorul FRC deci, nu se poate obține exact frecvența dorită pe magistrala I²C, însă prin alegerea corespunzătoare a divizorului din generatorul de semnal de ceas al perifericului I2C1 am obținut o frecvență suficient de apropiată pentru a nu afecta funcționarea.

Pentru calculul valorii potrivite ce trebuie scrisă în registrul I2C1BRG pentru a genera frecvența dorită, un utilizator poate folosi scriptul MATLAB menționat și la calcularea parametrilor buclei PLL. Alternativ, se poate folosi formula de calcul implementată în cod, actualizând definițiile relevante cu valorile potrivite.

```
//  In defines.h
    #define FSCL    400000UL        //  I2C desired baudrate frequency - 400KHz

//  In setup.c
void I2C1_Initialize(void)
{
    ...
    CONFIG2 = FCY / FSCL - 2;
    ...
}
```

4.1.4.2. Operarea normală

Pentru a utiliza circuitul MPU6050 împreună cu microprocesorul dsPIC, am pus la dispoziție două variante:

- O bibliotecă de cod pe care am obținut-o prin portarea codurilor create de Jeff Rowberg [24] pentru Arduino și dsPIC30F pe platforma dsPIC33F. Codul este simplu de înțeles și de folosit, însă biblioteca nu este potrivită pentru aplicațiile complexe, pentru că nu implementează toate funcționalitățile.

- O bibliotecă de cod pe care am obținut-o prin portarea codului sursă pus la dispoziție de producătorul senzorului, InvenSense, pentru un procesor ARM, produs de STMicroelectronics. Deși am redus numărul de fișiere și am renunțat la unele funcționalități, biblioteca aceasta este mult mai complexă, mult mai greu de înțeles și de folosit și mult mai costisitoare ca resurse de memorie și timp de procesare, însă este potrivită pentru aplicații ce folosesc funcțiile avansate ale procesorului DMP.

În aplicația curentă am folosit prima bibliotecă menționată, pentru că nu am avut nevoie de funcții avansate ale DMP-ului, însă am avut nevoie ca interacțiunea dintre microcontroler și senzor să fie cât mai redusă. În **Figura 4.9** pot fi observate o serie de mesaje conținând valorile măsurate de giroscop ale orientării exprimate format YPR (engl. *Yaw Pitch Roll*). Mesajele au fost generate folosind biblioteca portată de pe Arduino și au fost trimise de robot unei aplicații externe prin modulul nRF.

```
ypr     0.00    11.81   0.87
ypr     0.00    11.81   0.87
ypr     0.00    11.81   0.87
ypr     0.00    -13.12  -0.87
ypr     0.00    -13.12  -0.87
ypr     0.00    -13.12  -0.87
ypr     -1.75   -21.87  1.75
ypr     -1.75   -21.87  1.75
ypr     -0.87   5.25    0.00
```

Figura 4.9 – Date nefiltrate de la giroscop aflat în poziție orizontală. Date prezentate în format YPR

Modulul MPU6050 integrează doi senzori: un giroscop și un accelerometru. Giroscopul și accelerometrul furnizează date în format digital, rezultatul măsurătorii pe fiecare axă fiind disponibil sub forma unui număr întreg reprezentat pe 16biți în complement față de 2. Sensibilitatea giroscopului atât în aplicația curentă, cât și în mod implicit este de $\pm 250°/\text{sec}$, dar ea poate fi setată și la $\pm 500°/\text{sec}$, $\pm 1000°/\text{sec}$, respectiv $\pm 2000°/\text{sec}$. Giroscopul operează implicit la o rată de eșantionare de 8KHz, dar ea poate fi scăzuta la 1KHz. Sensibilitatea accelerometrului este în mod implicit $\pm 2\text{g}$, dar ea poate fi schimbată la $\pm 4\text{g}$, $\pm 8\text{g}$ sau $\pm 16\text{g}$. Rata de eșantionare a acestuia este fixată la 1KHz. Pentru a seta parametrii menționați și mulți alții, amândouă bibliotecile dispun de funcții specializate.

Cele două biblioteci dispun de funcții de inițializare ale modulului MPU, funcții dedicate diverselor setări, funcții de citire a valorilor măsurate de senzor și funcții de conversie a acestora în diverse formate. În plus, acestea oferă și funcții dedicate lucrului cu procesorul DMP, a cărui utilitate este detaliată în cele ce urmează.

4.1.4.3. Procesorul digital de mișcare

Dispozitivul pus în discuție este un echipament specializat, integrat în capsula de siliciu a senzorului MPU6050. Acesta poate prelua valorile măsurate de senzorii integrați și chiar și valori ale unor senzori externi, conectați prin interfața I^2C secundară a circuitului. Procesorul DMP poate fi folosit pentru a detecta căderea în gol a senzorului și pentru a detecta diverse gesturi, facilități utile, de exemplu, pentru un dispozitiv de tipul telefoanelor mobile. Pentru aplicația curentă însă, marele bonus obținut din folosirea DMP-ului este că prin aceasta se evită nevoia de filtrare manuală a datelor.

Datele legate de axa de ruliu (engl. *roll*) și de axa de tangaj (engl. *pitch*) pot fi satisfăcător filtrate și pe microcontroler, folosind de exemplu un filtru Kalman. Pentru corectarea valorilor pe axa de girație (engl. *yaw*) a giroscopului, este însă nevoie să folosim și datele de la accelerometru, care la rândul lor trebuie mai întâi filtrate. Performanțele filtrelor manuale sunt complet dependente de parametrii aleși de inginerul proiectant, iar întregul proces de filtrare presupune lucrul în aritmetica virgulei mobile, în care operațiile sunt foarte costisitoare ca timp de execuție.

Având în vedere că microcontrolerul are de îndeplinit și alte operații pe parcursul funcționării robotului, obținerea informațiilor de la senzori ar fi întârziată, ceea ce ar scădea performanțele regulatoarelor și ar îngreuna procesul lor de proiectare. În plus, chiar dacă nu este folosit, DMP-ul consumă energie și fiecare operațiune în plus executată de procesor crește și ea consumul de energie, ceea ce înseamnă că filtrarea manuală a datelor ar putea duce la scăderea autonomiei de funcționare a robotului.

Folosind procesorul digital de mișcare deci, reducem munca microcontrolerului, obținem datele filtrate într-un timp mai scurt decât în cazul filtrului implementat pe controler, scădem din consumul total de energie al robotului și obținem datele fuzionate ale giroscopului și accelerometrului sub formă de cuaternioni. Pe scurt, cuaternionii fac parte din categoria numerelor hipercomplexe și sunt numiți astfel pentru că spațiul lor este obținut prin extinderea mulțimii numerelor complexe, prin adăugarea a doi termeni imaginari j și k, ce respectă condiția $i^2 = j^2 = k^2 = ijk = -1$. Noțiunea de cuaternion este una pur matematică, iar reprezentarea este utilizată în practică în special pentru calcule ce implică rotații tridimensionale. Cuaternionii obținuți de la senzor pot fi direct utilizați în algoritmii de reglare, însă bibliotecile de cod create dispun de funcții ce pot converti datele și în alte reprezentări.

În **Figura 4.10** este vizibilă o comparație între datele colectate direct de la giroscop, convertite în format YPR, și datele obținute de la procesorul de mișcare, convertite în același format. Datele nefiltrate și cele de la DMP au fost adunate și trimise simultan către o aplicație externă. Pe parcursul colectării datelor, senzorul a fost staționar într-o poziție aproximativ paralelă cu solul. Am creat un script MATLAB, cu care am citit fișierele ce conțineau datele nefiltrate și

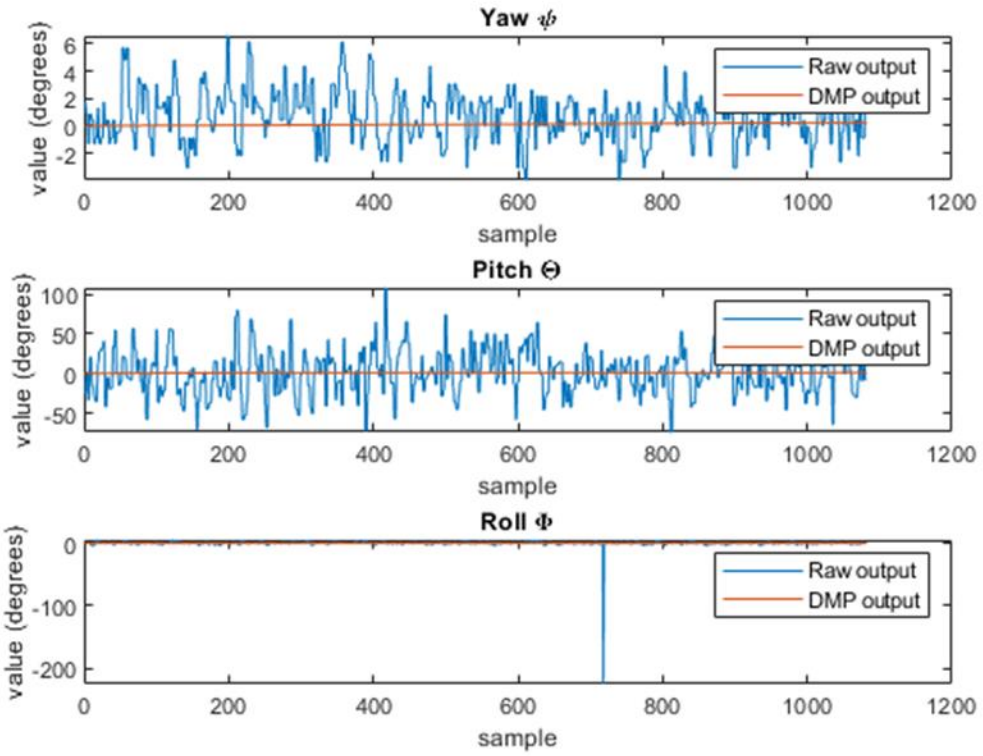


Figura 4.10 – Comparație între rezultatele nefiltrate ale măsurătorilor senzorului MPU6050 și rezultatele filtrate de DMP

datele de la DMP, iar în figură sunt observate reprezentările grafice ale evoluțiilor în timp a valorilor citite. Pentru a pune în evidență și mai bine avantajele utilizării DMP-ului am inclus o serie de grafice ce conțin măsurători și rezultate prezentate în câteva lucrări de diplomă [26, 27]

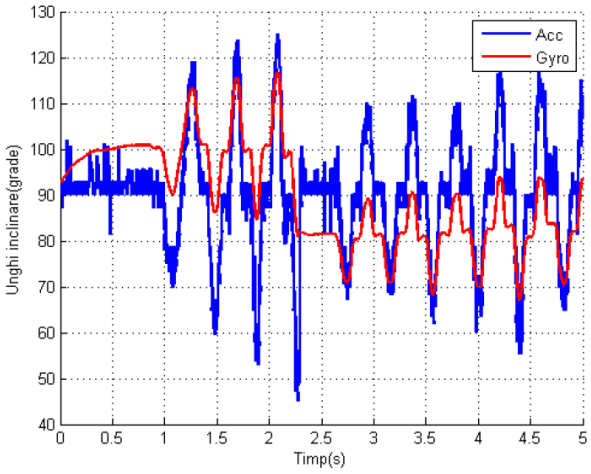


Figura 4.11 – Unghiul de tangaj conform giroscopului și a accelerometrului [26]

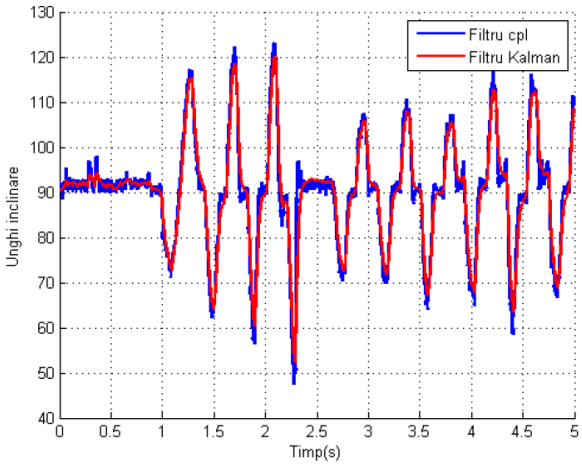


Figura 4.12 – Unghiul de tangaj obținut de filtrul complementar și filtrul Kalman [26]

prezentate în anii anterior în cadrul facultății în care a fost folosit același modul de senzori.

În figurile de mai sus putem observa perturbații foarte mari la măsurare, datorate vibrațiilor puternice a motoarelor pas cu pas folosite și a caracteristicilor inerente ale senzorilor. Filtrul complementar reduce din perturbații, iar filtrul Kalman rafinează și mai mult rezultatele măsurătorilor, însă comparând ieșirea filtrului Kalman cu măsurătorile nefiltrate ale giroscopului devine evident faptul că perturbațiile nu au fost nici pe departe înlăturate. Cu siguranță că parametrii filtrului Kalman ar putea fi și ei acordați mai fin, ceea ce ar reduce și mai mult din perturbații, dar performanțele vor fi în continuare inferioare față de cele ale filtrului implementat de DMP. În graficele discutate este ușor sesizabil și un alt efect specific senzorilor giroscopici și anume așa-numita alunecare în timp. Aceasta se manifestă prin modificarea valorilor măsurate de giroscop în timp, chiar și atunci când orientarea sa nu se schimbă. Dacă un senzor giroscopic este staționar și îi sunt citite datele pe o perioadă de câteva minute, se poate vedea că valorile măsurate tind să „alunece” tot mai departe de valorile reale. Algoritmul folosit în sistemul căruia îi corespund graficele de mai sus compensează pentru această alunecare, însă o face manual, ceea ce nu oferă rezultate perfecte. Efectul poate fi observat și mai bine în **Figura 4.13**. Și în această abordare se observă că filtrul Kalman nu elimină complet perturbațiile.

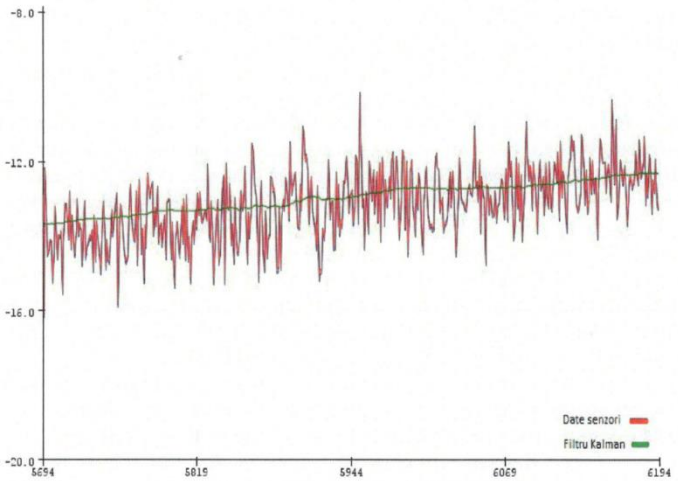


Figura 4.13 – Valorile nefiltrate ale unghiului de tangaj în comparație cu valorile filtrate [27]

Mai mult decât atât, giroscopul suferă de încă o problemă, discutată anterior, numită „blocaj gimbal”. Cele

două lucrări amintite nici nu abordează problema eliminării erorilor apărute ca urmare a acestui blocaj. În practică, acest fapt s-a transpus prin inabilitatea roboților dezvoltați în a executa viraje.

Prin fuzionarea datelor de la giroscop și accelerometru și reprezentarea lor sub formă de cuaternioni, DMP-ul elimină complet problema blocajului gimbal, ceea ce înseamnă că pentru robotul dezvoltat pot fi ușor proiectate regulatoare care să permită menținerea echilibrului și în timpul virajelor. Prin algoritmul de filtrare cu auto-calibrare de care dispune, DMP-ul elimină atât perturbațiile variabile, cât și deviațiile cu efect integrator în timp, produse de efectul de „alunecare". Prin faptul că am făcut posibilă utilizarea cu succes a procesorului DMP, am redus consumul de energie robotului, am scăzut semnificativ volumul de muncă al microcontrolerului, am oferit o metodă facilă prin care un utilizator poate obține date filtrate de o calitate excepțională de la senzor și am extins plaja de aplicații în care poate fi utilizat robotul prin faptul că acesta are acces la toate funcționalitățile suplimentare a procesorului de mișcare.

Dacă cele discutate până acum fac evidente motivele pentru care este recomandată utilizarea DMP-ului, acesta prezintă și câteva dezavantaje. În primul rând, memoria procesorului de mișcare este volatilă, ceea ce înseamnă că firmware-ul acestuia trebuie rescris la fiecare alimentare. Aceasta înseamnă că microprocesorul trebuie să rezerve o zonă de memorie pentru a stoca firmware-ul DMP-ului și, mai mult, la fiecare inițializare el trebuie să consume timpi suplimentari pentru a scrie firmware-ul în memoria procesorului de mișcare. În aplicația curentă, firmware-ul în discuție este reținut în memoria de date a microcontrolerului. Un al doilea neajuns al DMP-ului a intervenit în faza de implementare a bibliotecilor de cod pentru MPU. Complexitatea codurilor sursă puse la dispoziție de producător, dificultatea în procesul de depanare, lipsa unei documentații coerente și separarea documentației procesorului de mișcare de cea a senzorului au făcut ca procesul de portare și adaptare a codului să fie extrem de lent.

4.2. Firmware-ul plăcii Arduino

Deși platforma robotică poate funcționa independent, ea are capacitatea de a primi comenzi și de a returna informații unei aplicații externe. Ca intermediar în comunicarea dintre robot și o aplicație de control și supervizare cu acces la capabilități de procesare superioare, am folosit o placă Arduino Uno, însă în locul acesteia, cu modificări minimale ar putea fi folosită aproape oricare placă din gama Arduino și foarte multe dintre sutele de plăci compatibile cu Arduino dezvoltate de alți producători. Firmware-ul este conceput pentru a demonstra un exemplu de implementare al unui protocol de comunicație între robot și alt dispozitiv, iar pentru aceasta folosește o bibliotecă de cod „RF24" [20], disponibilă și pentru alte platforme populare, ca Raspberry Pi. Acest ultim detaliu înseamnă că, urmărind funcționarea programului discutat în continuare, un utilizator poate implementa ,de exemplu, o aplicație pentru o placă Raspberry Pi, care, datorită capabilităților platformei, s-ar putea ocupa în mod direct atât de comunicația cu robotul, cât și de controlul și supervizarea acestuia.

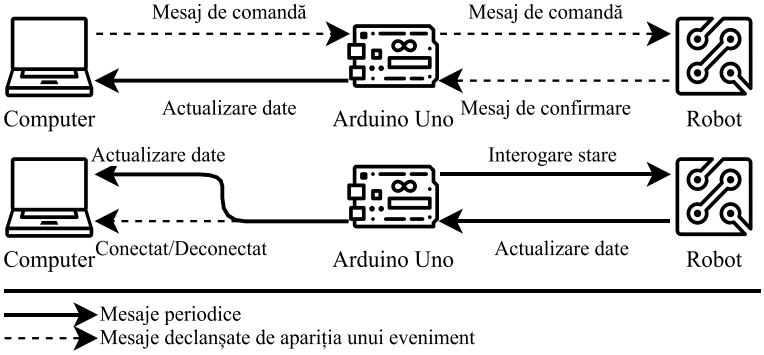


Figura 4.14 – Diagrama simplificată a scenariilor de comunicare pe placa Arduino Uno

În aplicația curentă, placa Arduino Uno, din poziția sa ca interfață între computer și sistemul robotic, are o serie de roluri, observabile în **Figura 4.14**:

1. Atunci când utilizatorul trimite o comandă prin intermediul aplicației de calculator, placa Arduino trimite mai departe comanda către robot și așteaptă un mesaj de confirmare a recepției. Ulterior, placa transmite datele utile din mesajul de confirmare înapoi către aplicația de comandă. În cazul în care nu primește din prima un răspuns, placa reîncearcă trimiterea mesajului de comandă.
2. În mod periodic, în lipsa unui semnal primit din partea aplicației de comandă, placa Arduino trimite un mesaj de interogare a stării către platforma robotică. În cazul în care este recepționat cu succes mesajul, robotul răspunde cu un mesaj conținând date despre valorile filtrate citite de la senzori și date despre starea componentelor platformei robotice.

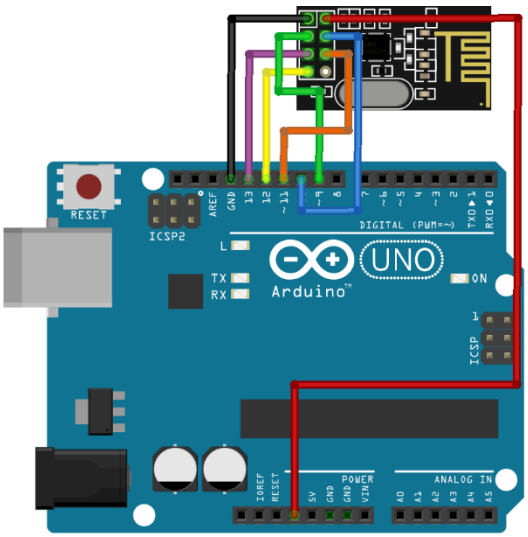


Figura 4.15 – Conexiunea Arduino Uno – nRF24L01+

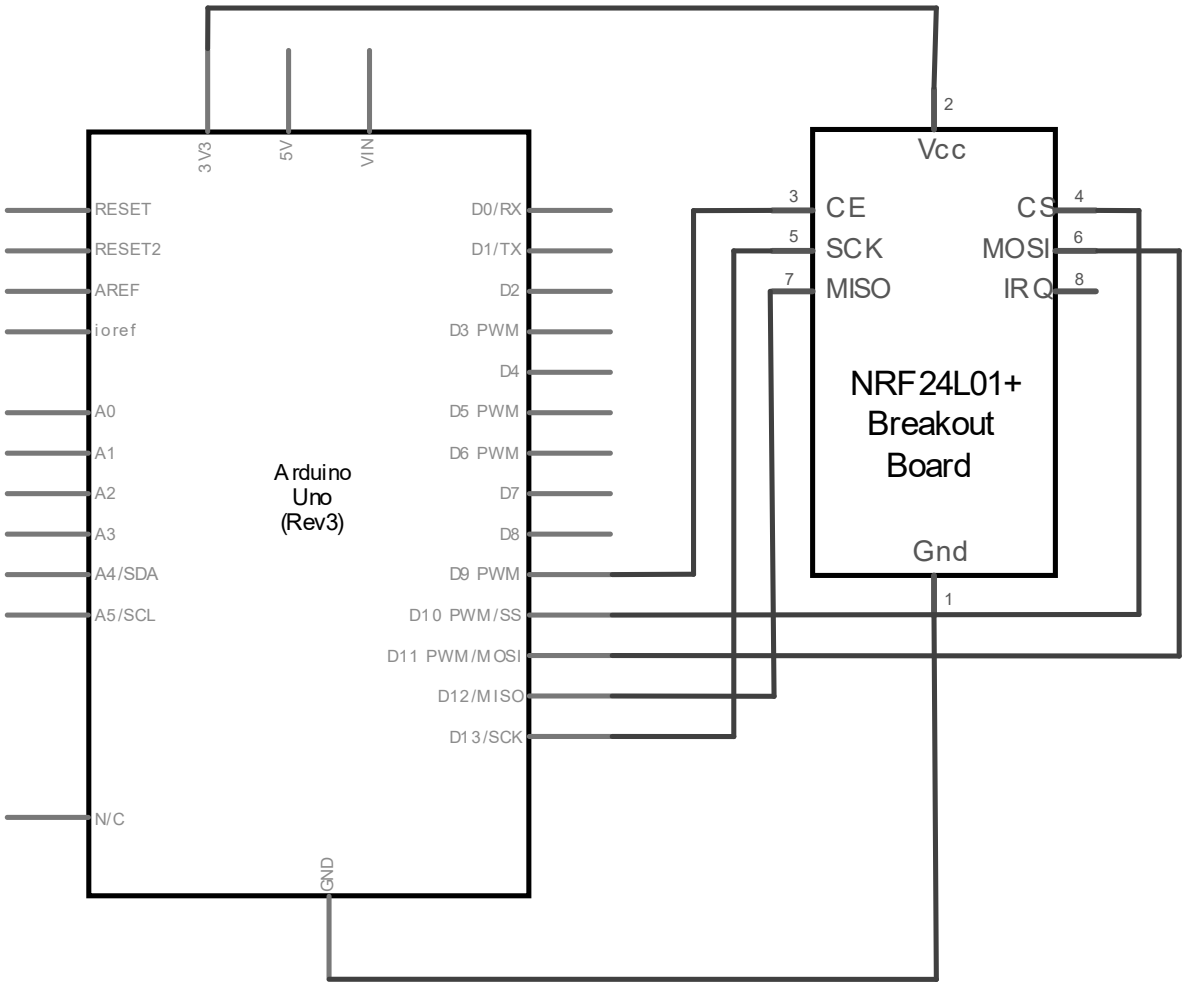


Figura 4.16 – Schema electrică a circuitului format din placa Arduino Uno și modulul de comunicație

3. În funcție de prezența sau absența mesajelor de confirmare din partea robotului, placa Arduino va transmite starea conexiunii cu robotul către aplicația de comandă, atunci când ea se schimbă.

4.3. Aplicația de comandă PC

Figura 4.18 – Indicatorii de stare ai aplicației

Pentru a controla de la distanță platforma robotică, am dezvoltat o aplicație de .NET C#, nativă pentru sistemele de operare Microsoft Windows 7, 8, 8.1 și 10. Deși aceasta nu face obiectul de discuție al lucrării, am pus-o la dispoziție pentru a demonstra de câtă versatilitate și flexibilitate dispune robotul dezvoltat, prin capacitatea sa de a utiliza modulul de comunicație. Așadar, întrucât scopul aplicației este pur demonstrativ, ea nu oferă decât funcționalități limitate, servind ca un punct de plecare pentru un utilizator ulterior.

În forma sa curentă, aplicația folosește un controller Microsoft Xbox One sau un altul compatibil, fiind capabilă să detecteze și să proceseze comenzile primite de la acesta. Aplicația dispune de o serie de indicatori ce arată starea actualizată a conexiunilor cu echipamentele externe: controllerul Xbox, placa Arduino și platforma robotică.

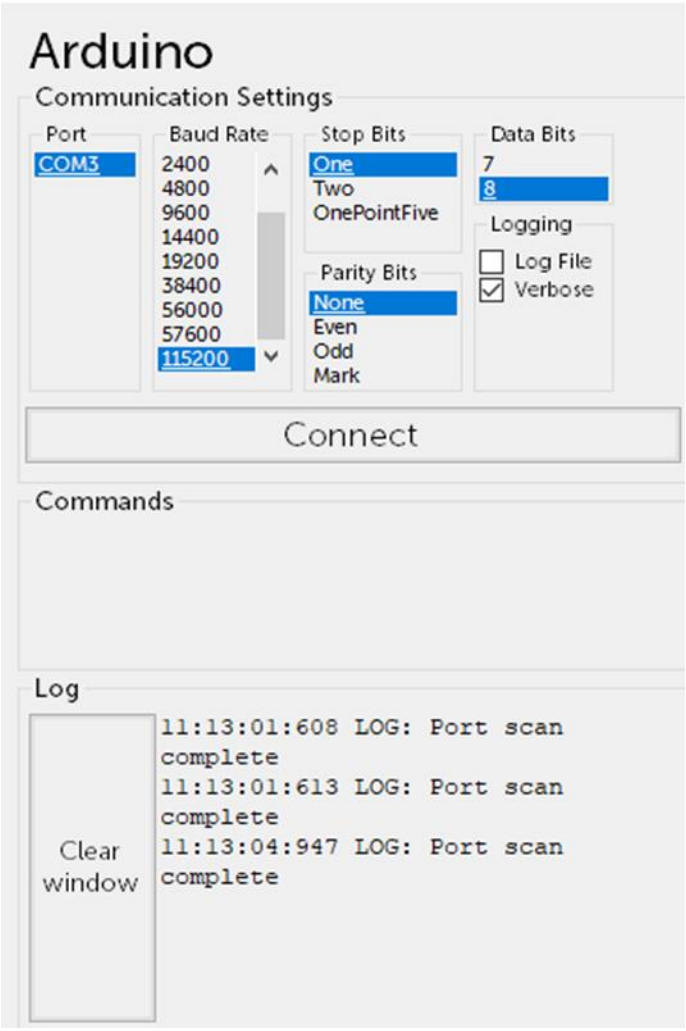


Figura 4.17 – Panoul de control al interacțiunii cu placa Arduino

Prin aplicație, utilizatorul poate face setările necesare pentru a se conecta la placa Arduino. Pentru compatibilitatea cu firmware-ul demonstrativ pus la dispoziție pentru placa Arduino, setările potrivite pot fi văzute în **Figura 4.17**. Prin implementarea opțiunilor pentru parametrii protocolului, aplicația devine compatibilă cu orice echipament cu care utilizatorul decide să înlocuiască placa Arduino, atât timp cât acesta suportă același protocol de bază.

Aplicația arată în fereastra de log-uri informații despre operațiile efectuate intern, despre starea porturilor și a

conexiunii cu placa Arduino, cât și despre mesajele trimise către aceasta. Tipurile de mesaje menționate pot fi ascunse prin dezactivarea opțiunii „Verbose”. Spre deosebire de acestea, mesajele de eroare și mesajele primite înapoi de la robot prin Arduino nu pot fi ascunse.

Comanda platformei robotice se face de către utilizator prin acționarea butoanelor controllerului Xbox. În funcție de elementele acționate pe controller, dacă este deschis canalul de comunicație cu placa Arduino și robotul este detectat, aplicația compune și trimite mesaje de comandă către acesta din urmă folosind placa de dezvoltare ca intermediar.

Capitolul 5. Concluzii

Figura 5.1- Robotul (stânga) în timpul funcționării, alături de placa Arduino, controler și calculatorul ce execută aplicația de comandă

Scopul lucrării, prezentat în capitolul de introducere, a fost acela de a proiecta și dezvolta un robot mobil cu pendul inversat, ceea ce am și făcut, folosind programe și echipamente profesionale și metode moderne, de actualitate.

Am studiat specificul roboților cu pendul inversat și am conceput un cablaj imprimat personalizat, robust și potrivit aplicației. Am dotat cablajul cu o serie de conectori specializați pentru fiecare componentă, ceea ce ușurează testarea funcționalităților acestora și schimbarea lor rapidă în cazul apariției unor defecțiuni. Folosind microcontrolerul Microchip dsPIC33FJ128MC802, dedicat pentru aplicații de control ce integrează motoare, am oferit

posibilitatea de a utiliza algoritmi specifici procesoarelor de tip DSP (engl. *Digital Signal Processor*), implementați optim și direct accesibili prin bibliotecile de cod implicite. Pentru a obține datele necesare într-o aplicație de control am folosit senzorul MPU6050, procesorul digital de mișcare de care acesta dispune și două codificatoare magnetice. Doar cu aceste componente, robotul ar fi deja pregătit să funcționeze independent, dar am reușit să extind și mai mult capacitățile acestuia, prin introducerea modulului de comunicație.

Am pus la dispoziție codul sursă necesar folosirii protocolului SPI, iar pentru protocolul I^2C am dezvoltat nu una, ci două biblioteci complet funcționale ce satisfac cerințele oricărei aplicații.

Și pentru modulul de senzori am pus la dispoziție două biblioteci de cod ce permit atât interacțiuni simple între microcontroler și senzor, cât și folosirea capabilităților extinse ale procesorului digital de mișcare cu care senzorul vine integrat. Aceasta este poate cea mai mare realizare din întregul proiect, întrucât folosirea DMP-ului aduce o multitudine de avantaje care cumulate devin extraordinar de importante. Cel mai relevant este că, prin faptul că am folosit procesorul în discuție, nu a mai fost nevoie să tratez efectele cauzate de blocajul gimbal, de tendința măsurătorilor de a devia în timp și de alte perturbații care altminteri produc probleme ce pot afecta atât procesul de proiectare, cât și funcționarea unor eventuale regulatoare. Spre deosebire de cazul multor alte aplicații, în care sunt folosiți senzorii împreună cu un filtru software, în cazul aplicației curente robotul are acces la măsurători de o acuratețe net superioară, deși nu este implementată nicio metodă de filtrare în program.

Pentru modulul de comunicații am implementat o bibliotecă de cod prin care robotul poate utiliza absolut toate funcționalitățile modulului. Aceasta înseamnă că peste nivelul fizic al protocolului, SPI, caracteristicile nivelurilor stabilite de circuitul nRF24 pot fi modificate pentru a se potrivi cerințelor utilizatorului. Pot fi concepute în consecință și o varietate de versiuni a nivelului de aplicație a protocolului de comunicare cu un program extern. În aplicația demonstrativă sunt folosite de exemplu:

1. Capabilitatea de a schimba frecvența de emisie-recepție
2. Capabilitatea de a transmite mesaje cu lungime variabilă
3. Capabilitatea de a trimite mesaje de confirmare în mod automat
4. Capabilitatea de a trimite date utile prin mesajul de confirmare
5. Capabilitatea de a comunica prin mai multe conexiuni
6. Capabilitatea de a genera cereri de întreruperi la apariția unui eveniment important
7. Capabilitatea de a opri și reporni oricând modulul

În genere, pentru toate componentele am creat biblioteci de cod ce pot fi utilizate pentru a profita de toate funcționalitățile posibile. Prin aplicația demonstrativă am oferit un punct de plecare pentru dezvoltarea unei aplicații mai complexe și un exemplu clar pentru modul de utilizare al bibliotecilor de cod create.

Mai mult decât doar din perspectiva electronică și software, am conceput și șasiul, suportul fizic ce transformă întregul sistem proiectat și dezvoltat într-un veritabil robot mobil. Șasiul este conceput pentru a putea fi cu ușurință imprimat, folosind o imprimantă 3D. Datorită felului în care am dezvoltat șasiul și, în genere, întregul robot, acesta are o serie de proprietăți avantajoase:

1. Costul de producție este relativ scăzut.
2. Robotul are dimensiuni reduse și este ușor de manipulat.
3. Robotul are un aspect ordonat, simetric.
4. Conexiunile dintre componente au rezistențe mecanice bune.

5. Șasiul este modular și adaptat pentru a oferi o oarecare flexibilitate în alegerea unor eventuale piese de schimb.
6. Componentele electronice defectate pot fi ușor înlocuite.
7. Centrul de greutate este apropiat de axul roților, ceea ce simplifică procesul de implementare și acordare al regulatoarelor și, în același timp, reduce efortul motoarelor în timpul funcționării.
8. Consumul de energie este scăzut, ceea ce îmbunătățește autonomia de funcționare
9. Utilizatorul poate afla informații despre starea robotului, chiar și în timpul funcționării independente a acestuia, prin LED-urile cu care platforma este dotată.

Pentru a demonstra faptul că am proiectat corect robotul astfel încât acesta să poată fi folosit într-o problemă de reglare cu pendul inversat, am creat și modelul intrare-stare-ieșire al întregii platforme, din care a rezultat faptul că sistemul este complet controlabil și parțial observabil. Pentru a facilita extinderea aplicației prin crearea unor modele mai complexe, am pus la dispoziție și un set de parametri ai platformei robotice.

Adițional, prin extinderea aplicației folosind placa Arduino și programul pentru calculatorul personal, am pus la dispoziție metode prin care pot fi înregistrate diverse date provenind de pe robot, în timpul funcționării, fără a fi nevoie de conexiuni suplimentare, fizice, care pot altera funcționarea platformei și valorile datelor culese, prin modificarea parametrilor fizici ai platformei. Aplicația externă propusă poate servi și ca un punct de plecare în dezvoltarea unei aplicații de comandă și de supervizare mai complexă.

Nu în ultimul rând, am creat un set de programe auxiliare ce s-au dovedit foarte utile în timpul dezvoltării platformei și aplicației de care ea este însoțită:

1. Un script MATLAB ce ușurează calculul valorilor potrivite pentru unele registre interne ale microcontrolerului dsPIC, pentru a obține frecvența de funcționare dorită și rata de transmisie pe I^2C selectată.
2. Un script MATLAB pentru calculul valorilor matricelor din modelul intrare-stare-ieșire propus.
3. Un script MATLAB ce poate fi utilizat pentru a compara datele obținute direct de la senzorul giroscopic cu datele filtrate de DMP.
4. Un program nativ pentru Windows ce poate prelua datele dintr-un fișier ce conține schimbul de mesaje pe I^2C între un modul MPU6050 și alt dispozitiv și poate interpreta mesajele pentru a le afișa utilizatorului într-un format mult mai lizibil și ușor de înțeles.

Luând în considerare faptul că mi-am îndeplinit obiectivele stabilite, ca direcții viitoare îmi propun să implementez un algoritm de reglare simplu, PID, urmat de o reglare după stare și una bazată pe logică fuzzy. Ca o încununare a muncii depuse, îmi propun să implementez o aplicație de control colaborativ, distribuită, pentru o echipă de roboți identici cu cel dezvoltat.

Bibliografie

[1] R. Hollis, „Ballbots,” *Scientific American,* vol. 295, nr. 4, pp. 72-77, 2006.

[2] O. Boubaker, „The inverted Pendulum: A fundamental Benchmark in Control Theory and Robotics,” în *International Conference on Education and e-Learning Innovations*, Sousse, Tunisia, 2012.

[3] V. Alimguzhin,, F. Mari, I. Melatti, I. Salvo și E. Tronci, „On Model Based Synthesis of Embedded Control Software,” în *10th ACM International Conference on Embedded Software - EMSOFT*, Rome, 2013.

[4] K. Lundberg, „The Inverted Pendulum System,” Noiembrie 2002. [Interactiv]. Available: http://web.mit.edu/klund/www/papers/UNP_pendulum.pdf. [Accesat 20 Iulie 2020].

[5] R. M. Tomazela, *A Combined Model-based Planning and Model-free Reinforcement Learning Approach for Biped Locomotion*, Campinas: Unversidade Estadual de Campinas, 2019.

[6] B. S. Cazzolato și Z. Prime, „On the Dynamics of the Furuta Pendulum,” *Journal of Control Science and Engineering,* vol. 2011, 2011.

[7] H. E. Tannous, „Interactive and Connected Rehabilitation Systems for E-Health,” Université de technologie de Compiègne, Compiègne, 2018.

[8] InvenSense Inc., *MPU-6000 and MPU-6050 Product Specification,* 2013.

[9] Nordic Semiconductors, *nRF24L01 Single Chip 2.4GHz Transceiver - Product Specification,* 2007.

[10] Pololu Corporation, „MAX14870 Single Brushed DC Motor Driver Carrier,” [Interactiv]. Available: https://www.pololu.com/product/2961.

[11] Maxim Integrated Products, Inc., „MAX14870/MAX14872 Compact 4.5V to 36V Full-Bridge DC Motor Drivers,” 2014.

[12] Microchip Technology Inc., „16-bit Digital Signal Controllers with Motor Control PWM and Advanced Analog,” 29 noiembrie 2011. [Interactiv]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/70291G.pdf. [Accesat 01 06 2020].

[13] Pololu Corporation, „100:1 Micro Metal Gearmotor HPCB 6V with Extended Motor Shaft,” [Interactiv]. Available: https://www.pololu.com/product/3075.

[14] Pololu Corporation, *Micro Metal Gearmotors*, Las Vegas: Pololu Corporation, 2019.

[15] Pololu Corporation, „Magnetic Encoder Pair Kit for Micro Metal Gearmotors, 12 CPR, 2.7-18V,” [Interactiv]. Available: https://www.pololu.com/product/3081.

[16] M. S. Mahmoud, „Introduction,” în *Advanced Control Design with Application to Electromechanical Systems*, Dhahran, Butterworth-Heinemann, 2018.

[17] S. Kim și S. Kwon, „Dynamic Modeling of a Two-wheeled Inverted Pendulum Balancing Mobile Robot,” *International Journal of Control, Automation, and Systems,* vol. 13, nr. 4, p. 926–933, 23 Mai 2015.

[18] B. Bonafilia, N. Gustafsson, P. Nyman și S. Nilsson, „Self-balancing two-wheeled robot,” 2013.

[19] Microchip Technology Inc., *MPLAB C30 C Compiler User's Guide,* 2015.

[20] TMRh20, „Optimized high speed nRF24L01+ driver class documentaion,” 2020. [Interactiv]. Available: http://tmrh20.github.io/RF24/.

[21] InvenSense Inc., *Motion Driver 6.12 - Features User Guide,* 2015.

[22] InvenSense Inc., *Motion Driver 6.12 - Getting Started Guide,* 2015.

[23] InvenSense Inc., *Motion Driver 6.12 - User Guide,* 2015.

[24] J. Rowberg, „I2C Device Library,” [Interactiv]. Available: http://www.i2cdevlib.com/. [Accesat 01 iunie 2020].

[25] Microchip Technology Inc., *Inter-Integrated Circuit (I2C),* 2015.

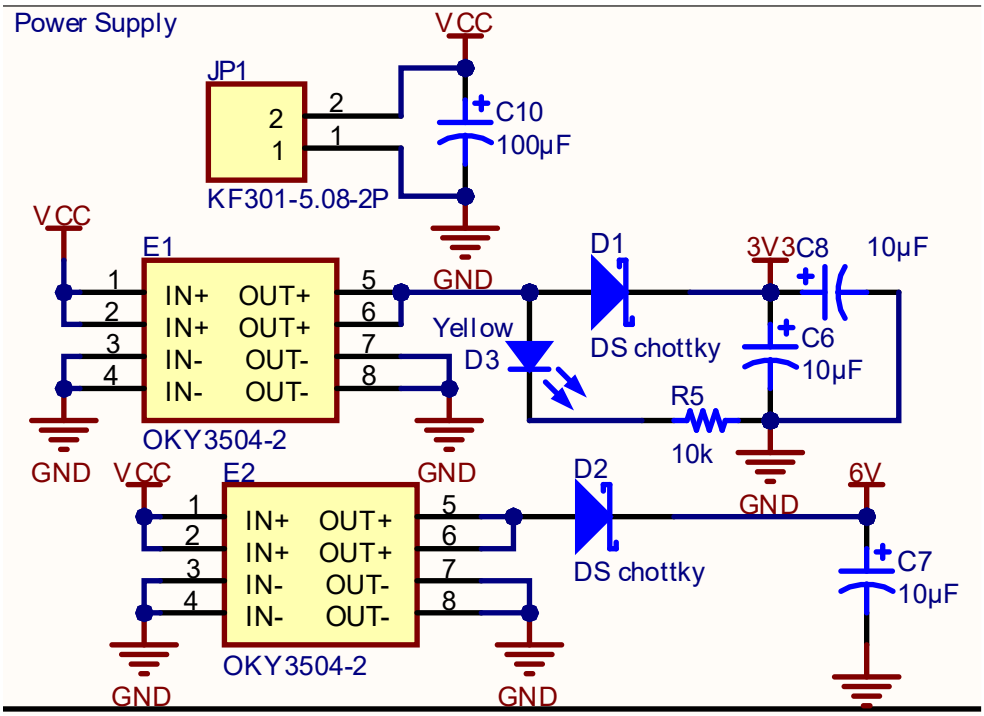
[26] M.-I. Munteanu, „Proiectarea și implementarea unui robot mobil de tip pendul inversat,” Iași, 2017.

[27] V. L. Negoiță, „Proiectarea și implementarea unui robot de tip pendul inversat,” Iași.

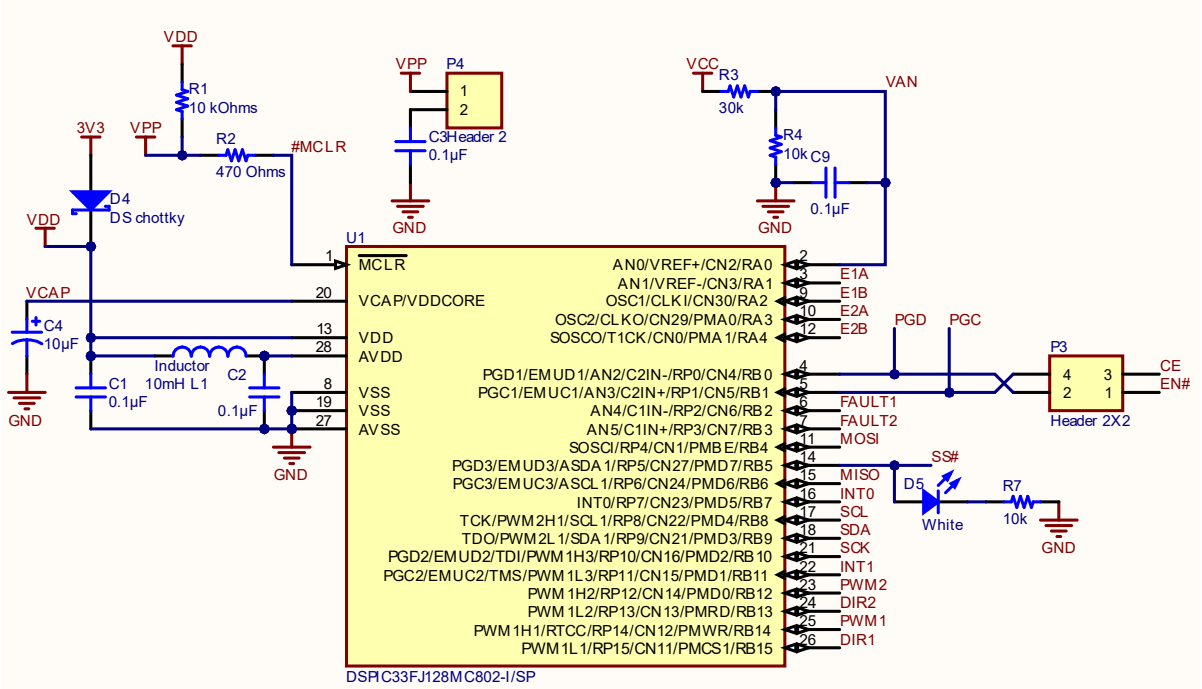
Anexe

Anexa 1. Schema electrică a cablajului imprimat personalizat

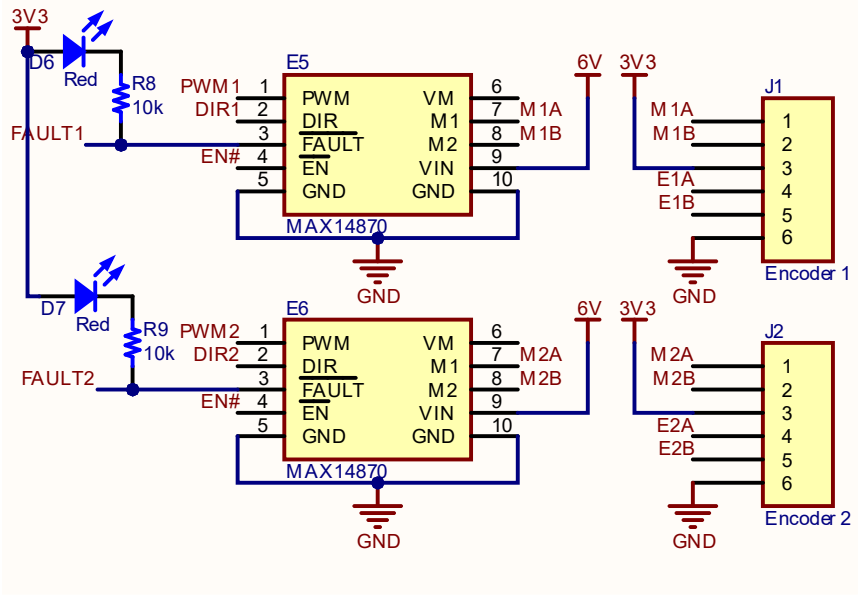
1.1. Alimentare



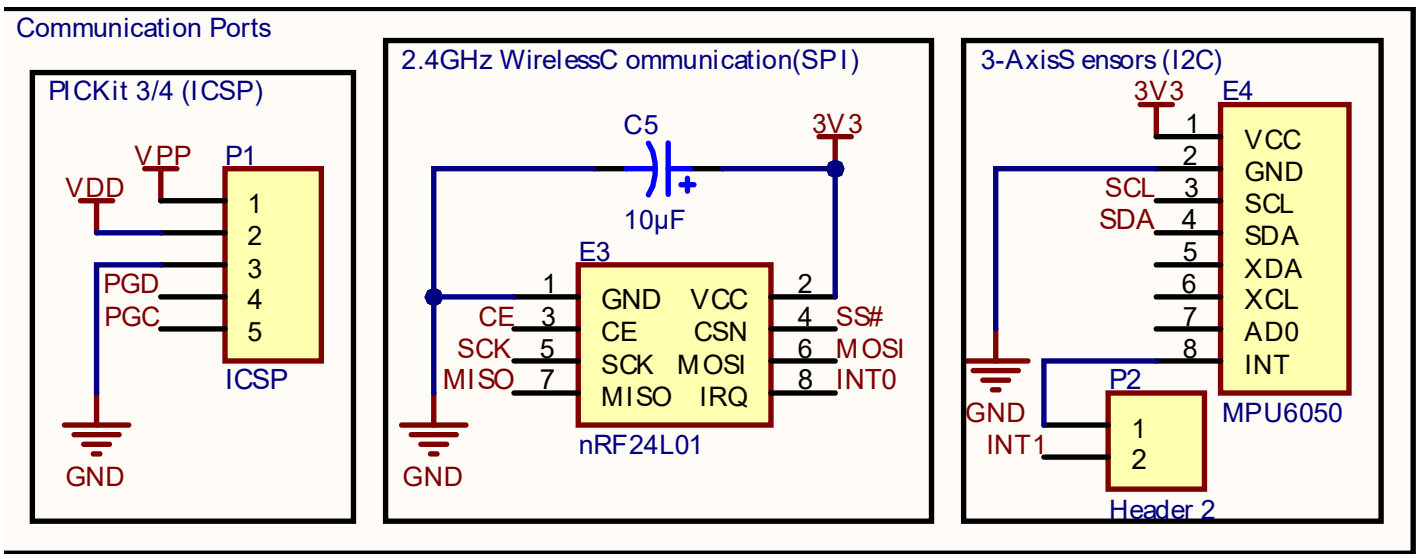
1.2. Microcontroler



1.3. Motoare și codificatoare



1.4. Porturi de comunicație



Anexa 2. Funcțiile detaliate ale pinilor microprocesorului

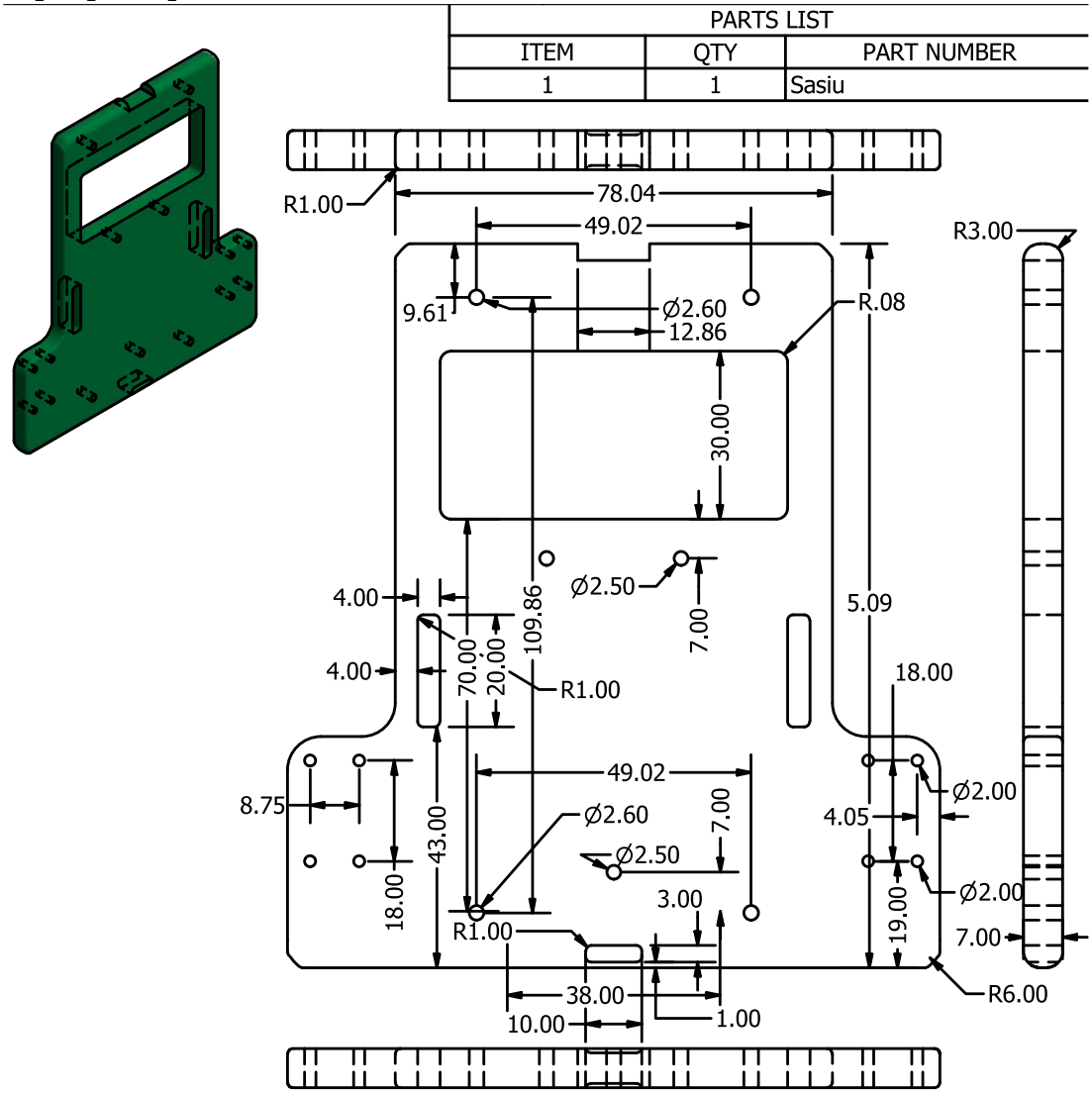
Nr. pin	Funcție	Tip pin	Net	Descriere
1	#MCLR	Intrare	#MCLR	Când este menținut la tensiunea VPP, procesorul este blocat în starea de resetare.
2	AN0	Intrare	VCC	Pin al convertorului ADC conectat la sursa de alimentare printr-un divizor rezistiv.

3	RA1	Intrare	E1A	Semnalul A de ieșire al codificatorului de pe motorul 1.
4	PGD1/RB0	Bidirecțional / Ieșire	PGD/EN#	La programare, semnal de date. În funcționarea normală semnal de activare a driverelor pentru motoare.
5	PGC1/RB1	Intrare / Ieșire	PGC/CE	La programare, semnal de ceas. În funcționarea normală semnal de activare a modulului de comunicație.
6	RP2	Intrare	FAULT1	Semnal de eroare al motorului 1.
7	CN5	Intrare	FAULT2	Semnal de eroare al motorului 2.
8	VSS	Alimentare	GND	Pin de masă.
9	RA2	Intrare	E1B	Semnalul B de ieșire al codificatorului de pe motorul 1.
10	RA3	Intrare	E2A	Semnalul A de ieșire al codificatorului de pe motorul 2.
11	RP4	Ieșire	MOSI	Linie de date unidirecțională de la perifericul SPI1 al microcontrolerului la modulul de comunicație.
12	RA4	Intrare	E2B	Semnalul B de ieșire al codificatorului de pe motorul 2.
13	VDD	Alimentare	VDD	Pin de alimentare de la regulatorul de 3.3V
14	RP5	Ieșire	SS#	Semnal folosit de perifericul SPI1 pentru a începe un schimb de date cu modulul de comunicație.
15	RP6	Intrare	MISO	Linie de date unidirecțională de la modulul de comunicație la perifericul SPI1 al microcontrolerului.
16	INT0	Intrare	INT0	Pin pentru cereri de întrerupere externă primite de la modulul de comunicație.
17	SCL1	Ieșire	SCL	Linia semnalului de ceas generat de perifericul I2C1.
18	SDA1	Bidirecțional	SDA	Linia semnalului de date generat sau primit de perifericul I2C1.
19	VSS	Alimentare	GND	Pin de masă.
20	VCAP	Alimentare	VDD	Pin de alimentare pentru regulatorul intern al microcontrolerului.
21	RP10	Ieșire	SCK	Linia semnalului de ceas generat de perifericul SPI1.
22	RP11	Intrare	INT1	Pin pentru cereri de întrerupere externă primite de la modulul de senzori MPU.
23	PWM1H2	Ieșire	PWM2	Linia de transmisie a semnalului PWM generat pentru motorul 2.
24	RB13	Ieșire	DIR2	Linia semnalului pentru comanda direcției de rotație a motorului 2.
25	PWM1H1	Ieșire	PWM1	Linia de transmisie a semnalului PWM generat pentru motorul 1.

26	RB15	Ieșire	DIR1	Linia semnalului pentru comanda direcției de rotație a motorului 1.
27	AVSS	Alimentare	GND	Pin de masă.
28	AVDD	Alimentare	VDD	Pin de alimentare de la regulatorul de 3.3V.

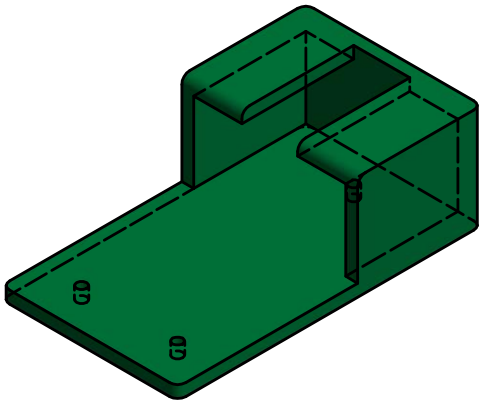
Anexa 3. Detaliile șasiului personalizat

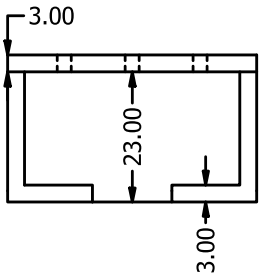
1.1. Corpul principal

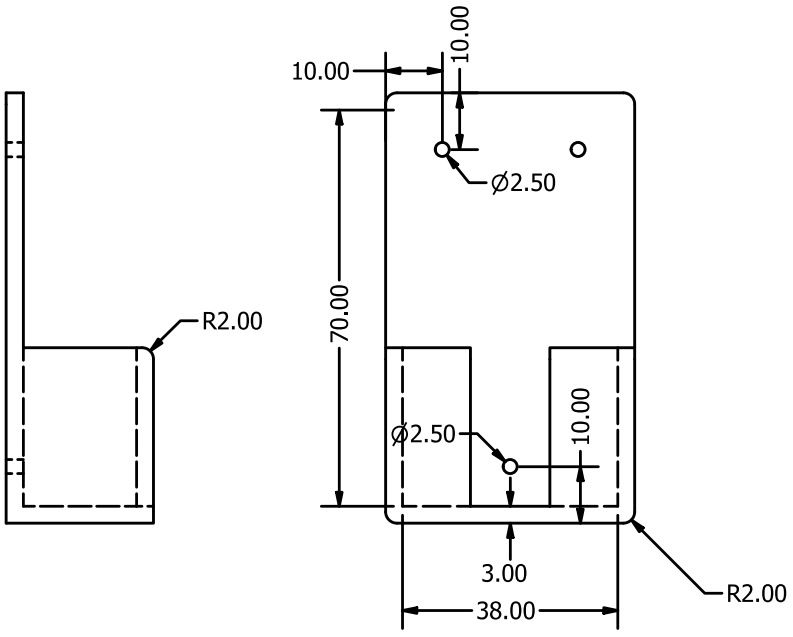


3.5. Suportul de baterie

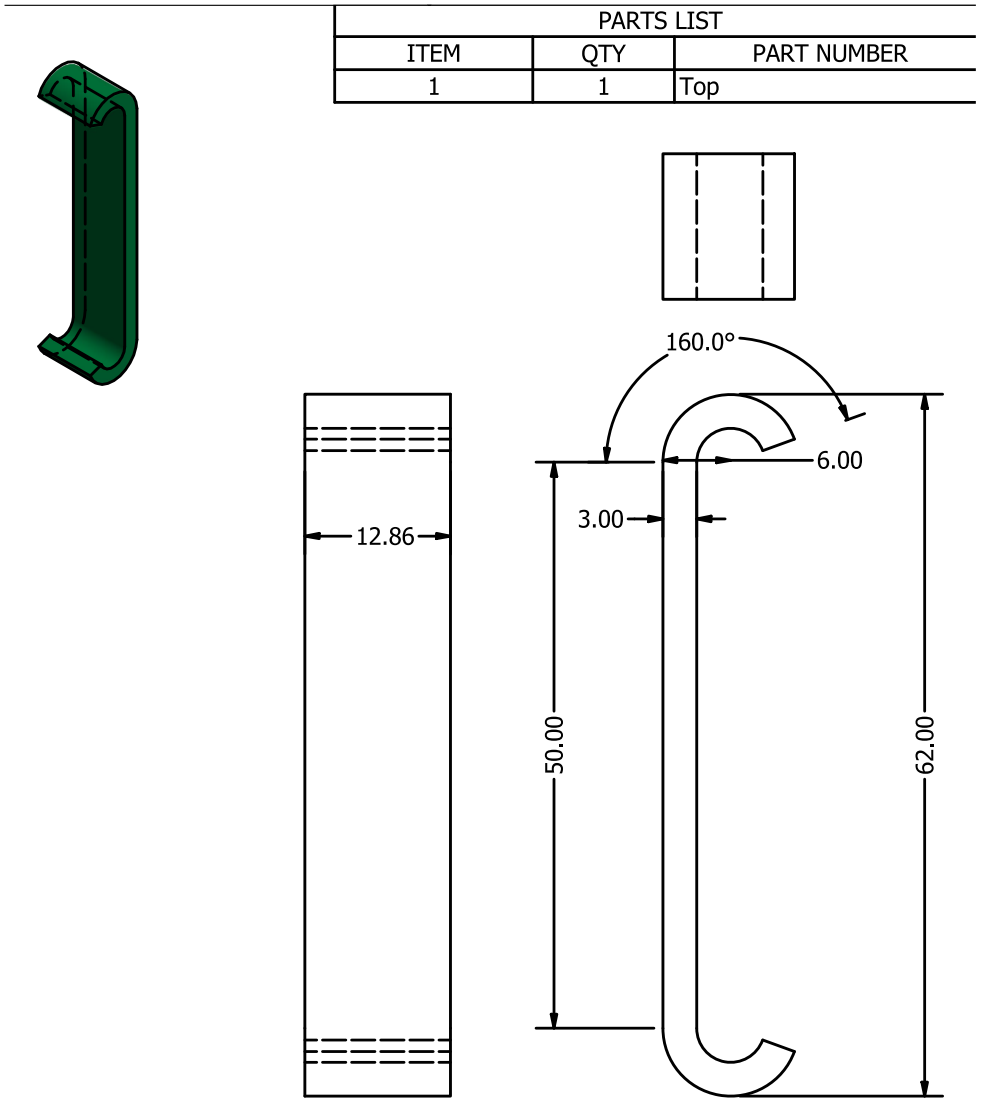
PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	BatteryCase







3.6. Opritor



Anexa 4. Software auxiliar

1.1. Script MATLAB pentru calculul parametrilor potriviți frecvențelor dorite de funcționare a dsPIC33F

```
clear all;
close all;
clc;

disp("Tool for calculating the best parameters for the PLL for the required
wanted frequency.");
disp("Processor frequency can be divided more using DOZE");
disp("Made for dsPIC33FJ128MC802");

KHz = 1000;
MHz = 1000*KHz;
```

```
FOSC = 7.3728 * MHz;
Wanted = input('Wanted FCY(MHz) = ') * MHz;
format , intmax('uint64');

FCY = 0;
difFCY = 80*MHz;

if (Wanted < 6.2513 * MHz)
    disp("Frequency too low for FRC+PLL");
    disp("FRC - 3.6864MHz");
    disp("FRCDIV16 - 230.4KHz");
    disp("FRCDIVN - 3.6864MHz / FRCDIV");
    disp("LPRC - 32.768KHz");
    disp("min(FRC+PLL) - 6.2513MHz");
    return;
end
for N1=0:1:31
    FCY_N1 = FOSC/(N1+2);
    if (FCY_N1 < 0.8 * MHz)
        break;
    end

    for M=0:1:511
        FCY_M = FCY_N1*(M+2);
        if (FCY_M < 100 * MHz)
            continue;
        end
        if (FCY_M > 200 * MHz)
            break;
        end

        for N2=0:1:2
            FCY_N2 = FCY_M/(2^(N2+1));
            if (FCY_N2 > 80*MHz)
                continue;
            end
            if (FCY_N2 < 12.5*MHz)
                break;
            end

            FCY = FCY_N2/2;

            if (abs(FCY - Wanted) < difFCY)
                difFCY = abs(FCY - Wanted);
                FCYMHz = FCY / MHz;
                FCYKHz = FCY / KHz;
                FCYHz = FCY;
                difM = M;
                difN1 = N1;
                difN2 = N2;
            end
        end
    end
end
```

```
disp("M = " + (difM + 2) + " (PLLFBD  = " + difM + ")");
disp("N1 = " + (difN1 + 2) + " (PLLPRE  = " + difN1 + ")");
disp("N2 = " + (2^(difN2 + 1)) + " (PLLPOST = " + difN2 + ")");

if (FCYMHz < 1)
    if (FCYKHz < 1)
        disp("FCY = " + FCYHz + "Hz");
    else
        disp("FCY = " + FCYKHz + "KHz");
    end
else
    disp("FCY = " + FCYMHz + "MHz");
end

Fscl = input('Wanted Fscl(KHz) = ') * KHz;
BRG = floor(FCYHz/Fscl)-2;
disp("BRG = " + BRG);
```

4.7. Script MATLAB pentru calcularea valorilor matricelor modelului intrare-stare-ieșire a robotului

```
clear all;
close all;
clc;

%% Init
d = 117.369 + 12;    % mm
l = 14.318;          % mm
r = 35;             % mm
m_B = 231.6504;      % g
m_w = 14.1748;       % g


% MOI
J = m_w * r^2;       % MOI wheel on axis
K = 1/2 * J;         % MOI wheel perpendicular on axis

I_xx = 492944.14;    % g*mm^2
I_xy = 2026.266;
I_xz = 15533.357;
I_yy = 245127.512;
I_yz = -10805.590;
I_zz = 312187.238;

I1 = I_xx;
I2 = I_yy;
I3 = I_zz;

%% For State Space
U = 6;
```

```
V_rpm = 330;

V = V_rpm * 2*pi/60;

F_stall_kg = 16;
F_stall = 9.80665*F_stall_kg;

Ke = U/V;
Km_mm = F_stall / 1.5;
Km = Km_mm / 1000;

M_b = m_B;
M_w = m_w;

J_b = I_yy * (10 ^(-9));
J_w = J * (10 ^(-9));
L_m = l;

R = 4;
b = 0;
g = 9.81;

L = 10;    % mH, din burta (aproximat dupa alte valori gasite)
L = L / 1000;
r = r / 1000;
L = l / 1000;

alfa = 2 * (R*b - Ke*Km) * (M_b*r*(L^2) + M_b*r*L + J_b) ...    / (R * (2 *
(J_b*J_b+J_w*(L^2)*M_b + J_b*M_w*(r^2) + (L^2)*M_b*M_w*(r^2)) +
J_b*M_b*(r^2)));

beta = -(L^2)*(M_b^2)*g*(r^2) ...
    / (J_b * (2*J_w + M_b*(r^2) + 2*M_w*(r^2)) + 2*J_w*(L^2)*M_b +
2*(L^2)*M_b*M_w*(r^2));

gamma = -2*(R*b-Ke*Km)*(2*J_w + M_b*(r^2) + 2*M_w*(r^2) + L*M_b*r) ...
    / (R*r*(2*(J_b*J_w + J_w*(L^2)*M_b + J_b*M_w*(r^2) +
(L^2)*M_b*M_w*(r^2)) + J_b*M_b*(r^2)));
 delta = L*M_b*g*(2*J_w+M_b*(r^2)+2*M_w*(r^2)) ...
    / (2*J_b*J_w +
2*J_w*(L^2)*M_b+J_b*M_b*(r^2)+2*J_b*M_w*(r^2)+2*(L^2)*M_b*M_w*(r^2));

epsilon = Km*r / (R*b - Ke*Km);

A = [0 1 0 0; 0 alfa beta -r*alfa; 0 0 0 1; 0 gamma delta -r*gamma];
B = [0 alfa*epsilon 0 gamma*epsilon]';
% D = [0; 0];
% C = [0 0 1 0; 0 0 0 1];
D = 0;
C = [0 0 1 0];

Wo = [C C*A C*(A^2) C*(A^3)]';
Wc = [B A*B (A^2)*B (A^3)*B];
```

```
sys = ss(A,B,C,D,-1);
[num, den] = ss2tf(A,B,C,D,1);
G = tf(sys);
Ts = 1/1000;
[numz,denz] = c2dm(num,den,Ts,'zoh');
```

Anexa 5. Diagrama automatului de stare a modulului de comunicație