Universitatea POLITEHNICA din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

# Sistem multiprocesor de zbor tridimensional cu şase elice verticale

## Proiect de diplomă

Prezentată ca cerinţă parţială pentru obţinerea titlului de
*Inginer* în domeniul *Electronică şi Telecomunicaţii*
*specializarea Electronică Aplicată*

Conducător ştiinţific
Prof. Univ. Dr. Ing. Corneliu BURILEANU

Absolvent
Adrian-Ioan LIŢĂ

2012

# Declaraţie de onestitate academică

Prin prezenta declar că lucrarea cu titlul "*Sistem multiprocesor de zbor tridimensional cu şase elice verticale*", prezentată în cadrul Facultăţii de Electronică, Telecomunicaţii şi Tehnologia Informaţiei a Universităţii POLITEHNICA din Bucureşti ca cerinţă parţială pentru obţinerea titlului de *Inginer* în domeniul *Electronică şi Telecomunicaţii*, programul de studii *Electronică Aplicată şi Ingineria Informaţiei* este scrisă de mine şi nu a mai fost prezentată niciodată la o facultate sau instituţie de învăţământ superior din ţară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referinţe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi fac referinţă la sursă. Reformularea în cuvinte proprii a textelor scrise de către alţi autori face referinţă la sursă. Înţeleg că plagiatul constituie infracţiune şi se sancţionează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor şi măsurătorilor pe care le prezint ca fiind făcute de mine, precum şi metodele prin care au fost obţinute, sunt reale şi provin din respectivele simulări, experimente şi măsurători. Înţeleg că falsificarea datelor şi rezultatelor constituie fraudă şi se sancţionează conform regulamentelor în vigoare.

Bucureşti, *Iulie 2012*
*Adrian-Ioan LIŢĂ*

_____

(semnătura în original)

# Cuprins

# Lista figurilor

# Lista tabelelor

# Lista acronimelor

ABS = Acrylonitrile butadiene styrene
ARM = Advanced RISC Machines – firmă ce deţine licenţa arhitecturii de procesoare cu set redus de instrucţiuni omonime
BEC = battery eliminator circuit
BEMF = back electro-magnetic force
BLDC = brushless direct current motor
DMA = direct memory access
ESC = electronic speed controller
$I^2C$ = Inter-Integrated Circuit, magistrală de date sincronă de viteză relativ mică
ICE = in-circuit emulator
KB = kilobyte: 210 bytes = 8 x 210 biţi
$K_v$ = constanta de flux [RPM/V]
MOS = Metal Oxid Semiconductor
PIC = marcă înregistrată Microchip® de microcontrolere
PID = proportional-integral-derivativ
PWM = pulse width modulation
RAM = random access memory
$R_{DS}$ = rezistenţa drenă-sursă a tranzistorului MOS în conducţie
RPM = rotaţii pe minut
SPI = serial peripheral interface – magistrală de date similară $I^2C$
UART = Universal Asynchronous receiver/transmitter
ZigBee = protocol de reţea wireless cu autoconfigurare

# Capitolul I Introducere

Încă din secolul 11 umanitatea a simţit nevoia de a avea dispozitive care să realizeze anumite sarcini în mod automat, şi care, pentru a nu inspira un sentiment de teamă, să arate cât mai uman posibil. Astfel în 1088, în China a fost construit primul ceas mecanizat care avea figurine înfățișând oameni (manechini) care indicau ora şi băteau clopotele. Câteva decenii mai târziu, unul dintre primii roboţi având formă umană a fost schiţat de însuşi Leonardo Da Vinci în jurul anului 1495 (1).

Revenind mai aproape de timpul nostru, la sfârşitul celui de-al Doilea Război Mondial s-au pus bazele inteligenţei artificiale, cercetarea fiind făcută de mai multe grupuri de interes. Pe de-o parte au fost ingineri precum Norbert Wiener, Alan Turing şi Claude Shannon iar pe de altă parte psihologi precum Walter Pitts şi Warren McCulloch (2).

În 1971 Ted Hoff împreuna cu echipa sa de la Intel au creat primul microprocesor care avea dimensiuni similare cu cele de astăzi, şi care singur era mult mai puternic decât ENIAC (computerul electromecanic al armatei americane folosit în al Doilea Război Mondial). De acolo progresul tehnologic a explodat, ajungându-se în zilele noastre la microprocesoare de uz general comerciale cu puteri de peste 80 000 de milioane de instrucţiuni pe secundă.

De asemenea, nevoia de a avea roboti s-a amplificat şi nevoia lor de a arăta cât mai uman a putut lua alte direcţii. Începând cu anii `80 Texas Instruments a făcut cercetare în crearea unor bombe şi rachete inteligente din punct de vedere electronic. Pe sectorul industrial roboţii au luat încet dar sigur locul oamenilor în producţie, lărgind astfel posibilităţile şi scăzând costurile. Lucruri pe care oamenii nu le pot face fizic, roboţii le fac, cu precizie maximă (3).

Alte aplicații ale roboţilor au fost automatizările. Încet s-a început cu automatizarea manevrelor avioanelor, a elicopterelor, apoi a maşinilor, şi odată cu aceasta, şi a modelelor de avioane, elicoptere si maşini.

Proiectul de hexa sau quadcopter 100% electric nu a fost niciodată realizat la scară mare până la sfârşitul lui 2011 când s-a realizat un dispozitiv ce putea ridica un om cu masa de 80 Kg. La scară de model, proiectul este foarte popular printre amatori precum şi printre câteva firme de nişă ce produc aparate de zbor pentru armată şi pentru fotografie. Dintre firmele de nişă putem enumera Mikrokopter[1] (quad-, hexa- si octo- copteruri cu preţuri variind între 1500 şi 3000 € având performanţe relativ bune – octocopterul poate ridica şi ţine o sarcină de până la 4Kg, cu un timp de zbor de aproximativ 10 minute sub această sarcină) şi Draganfly[2] (aparate profesionale destinate domeniului poliţienesc şi militar cu preţuri de peste 10 000 $ având performanţe relativ slabe pentru preţul cerut, dar calitatea fiind foarte ridicată).

Hexacopterul prezentat în această lucrare este realizat cu un buget de sub 2000 €, şi are toate capabilităţile de supraveghere, fotografie şi transport al unui copter profesional. Din punct

[1] www.mikrokopter.de/
[2] http://www.draganfly.com/

de vedere al sarcinii, aparatul poate ridica până la 4 Kg, dar sub această sarcină timpul de zbor scade sub 10 minute. Pentru un zbor fără sarcină suplimentară, timpul ţintit de zbor este de 25-30 de minute.

În Figura I-1 este prezentată schema mecanică a hexacopterului la scară. Pentru a defini exact cotele, diametrul unei elice este de aproximativ 28 cm şi diametrul mare al hexagonului central este de 25 cm.

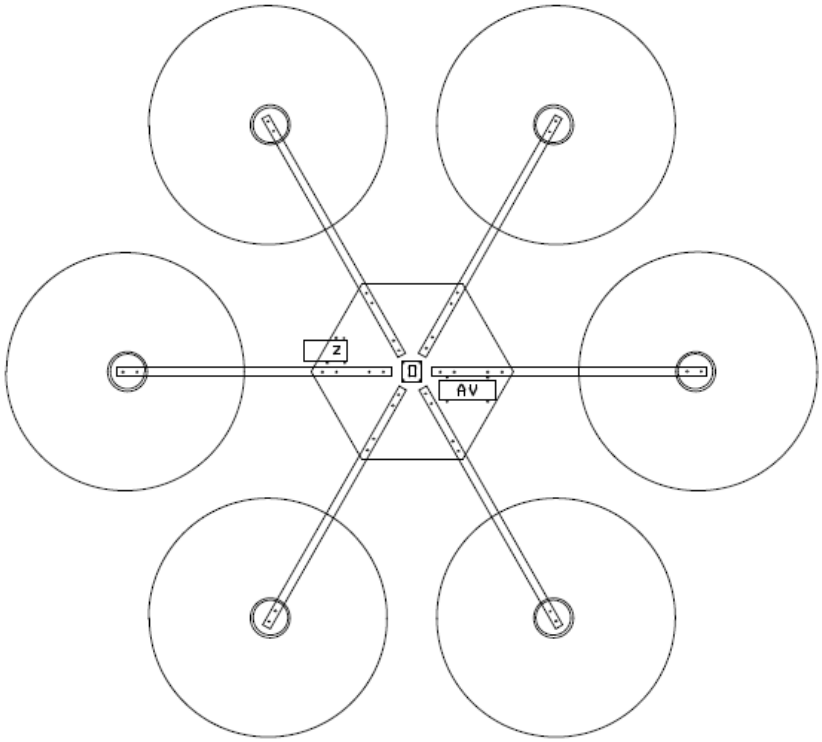


Figura I-1 Hexacopterul la scară

În schemă, pătratul central marcat cu „O” reprezintă senzorul de orientare, dreptunghiul marcat cu „Z” reprezintă transceiver-ul ZigBee iar dreptunghiul marcat „AV” reprezintă transmiţătorul audio-video. Barele din fibră de carbon ce susţin motoarele cu elice direct-drive au o lungime de 35 cm. Motoarele au un diametru de 4.5 cm şi stau pe un suport din aluminiu cu diametrul de 5 cm. Suportul este folosit pentru a adapta prinderea pe bara din fibră de carbon la prinderea motorului.

Figura I-2 Primul test al ansamblului

Figura I-3 Imagini de la primul test

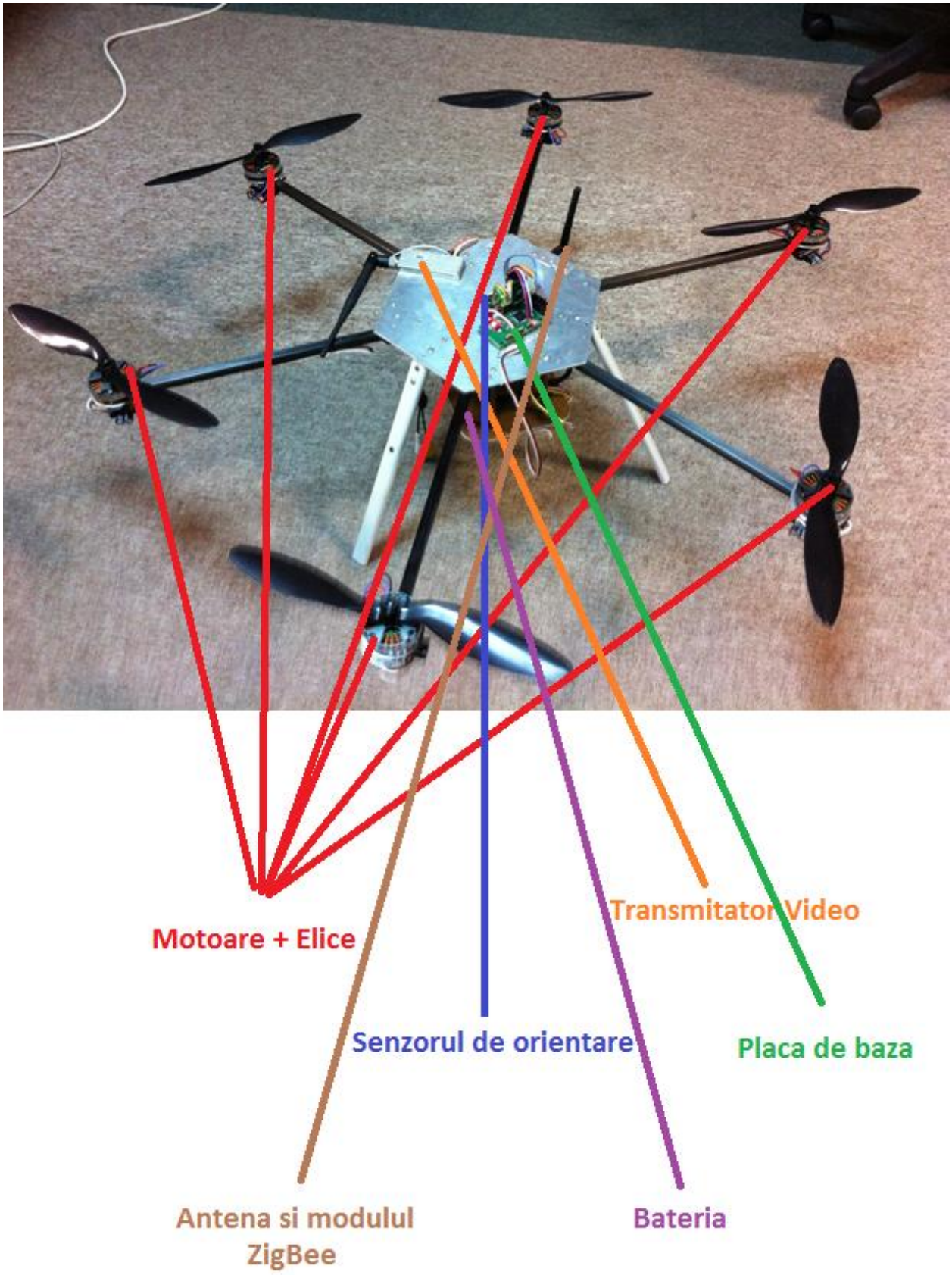


Figura I-4 Imagine descriptivă a componentelor principale

Construcţia totală a aparatului (doar parte hardware, fără soft) a durat aproximativ 5 luni deoarece majoritatea componentelor au fost comandate din străinătate şi transportul durează în general destul de mult.

În construcţie am folosit materiale precum aluminiu, otel, fibră de carbon, silicon (hot glue), bandă dublu adezivă, cauciuc şi plastic.

Controllerele de motoare au fost făcute manual: PCB-ul prin metoda de transfer UV al cablajului, iar componentele au fost lipite apoi cu un ciocan de lipit. Tot acest proces a durat o săptămână pentru toate cele 6 controllere.

Placa de bază a fost făcută la fabrică deoarece metoda prin UV de realizare de cablaje pe care am folosit-o nu poate realiza distanţe foarte mici între pini, şi microcontrolerul folosit are distanţa între padurile pinilor (clearance) de 200 de microni. Microcontrolerul este unul dintre cele mai performante microcontrolere din categoria low-cost, pe 32 de biţi, cu arhitectură MIPS® M4K®. Acesta vine într-o capsulă TQFP cu lungime de 12 mm şi 100 de pini.

Pentru programarea microcontrolerului am folosit uneltele de dezvoltare puse la dispoziţie de Microchip® gratuit pe situl http://www.microchip.com: MPLAB X® IDE si Microchip C32. MPLAB X® IDE este un mediu de dezvoltare bazat pe celebra platformă open source Netbeans, care permite dezvoltarea şi editarea facilă a programelor şi algoritmilor prin sugerarea de text (autocompletion) şi care, cu ajutorul compilatorului pe 32 de biţi Microchip C32 compilează şi construieşte un program din limbaj C în cod maşină specific microcontrolerului PIC32® folosit în proiect. De asemenea, tot în acest mediu de dezvoltare este posibilă programarea precum şi debuggingul software-ului pe microcontroler cu ajutorul unui programator sau emulator. Unealta folosită de mine pentru programare şi debugging a fost Microchip Real ICE® (in-circuit emulator capabil să facă tracing în unitatea de procesare a microcontrolerului).

Din microcontroler am folosit modulele UART, modulul $I^2C$, timere, precum şi întreruperile disponibile pentru aceste module pentru a putea scrie un algoritm performant de control al hexacopterului.

Dezvoltarea software-ului a durat aproximativ 3 luni, fiind începută imediat după terminarea proiectării parţii hardware, dar înainte de a o avea realizată fizic 100%. Primii paşi au reprezentat configurarea perifericelor microcontrolerului şi stabilirea unei structuri schelet, precum şi a câtorva maşini de stare pe care acesta să ruleze.

Un challenge l-a reprezentat decodificarea informaţiei de la senzorul de orientare, deoarece informaţia din datasheet nu corespundea în totalitate cu realitatea. Pentru aceasta am căutat forumuri şi persoane care au folosit un astfel de senzor pentru aplicaţii proprii.

# Capitolul II Hexacopter: analiză hardware și software

## 1. Sisteme utilizate în proiectarea aparatului

Un hexacopter este în fond un elicopter telecomandat cu 6 elice verticale. Ca oricare elicopter, acesta are nevoie în primul rând de motoare și elice. Pentru a ușura proiectarea, toate motoarele și elicele sunt identice. Sub acest aspect, dacă am fi ales toate elicele să se rotească într-un singur sens pentru a împinge aerul în jos, acestea ar fi impus un moment rezultant întregului aparat, moment căruia nu aveam un alt moment de sens opus, iar rezultatul ar fi fost că aparatul s-ar fi rotit continuu. Din această cauză a fost aleasă următoarea configurație: trei elice care se rotesc în sens trigonometric și trei în sens opus trigonometric, toate șase având aceleași proprietăți: lungime de 11 inch și pitch de 4.7 inch.

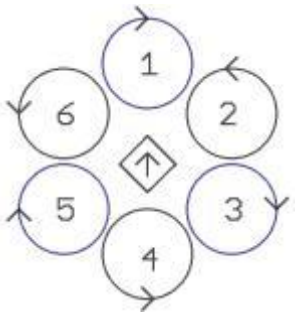


Figura II-1 Sensuri elice [5]

Motoarele folosite sunt motoare fără perii (brushless) construite special pentru aplicații de tip quadcopter sau hexacopter. Sunt motoare outrunner (cu rotor exterior) aplatizate, de turație relativ mică (aproximativ 530 de rotații pe volt aplicat în condiții de comutație corectă).


Figura II-2 Motorul folosit [4]

Specificațiile motorului oferite de producător sunt: rezistența fazei de 0.075Ω, curent consumat în gol de 0.5A, diametrul motorului de 45mm, masa de 67g, constanta de flux $K_v$ de 530RPM/V, tensiunea aplicată între 11.1 si 18.5V și o putere nominală de 150W. Axul motorului are diametru de 3mm (standard folosit pentru elice de mici dimensiuni din aeromodelism), dar producătorul oferă și un adaptor pentru elice cu axul de 5mm (standard folosit în elice de dimensiuni medii din aeromodelism).

De asemenea, vânzătorul a făcut o serie de teste cu diferite tipuri de elice, și rezultatele sunt prezentate în tabelul următor:

| Motor 4006D | | | | | | |
|---|---|---|---|---|---|---|
| **Elice** | **Tensiune [V]** | **Curent [A]** | **Putere [W]** | **Ridică [g]** | **RPM** | **Eficiență [g/W]** |
| GWS 9050 | 11.5 | 3 | 34.5 | 288 | 5776 | 8.347 |
| | 15.3 | 4.7 | 71.91 | 490 | 7448 | 6.814 |
| | 19.2 | 6.8 | 130.56 | 734 | 9030 | 5.621 |
| GWS 1060 | 15.3 | 6.4 | 97.92 | 622 | 7144 | 6.352 |
| | 19.2 | 9 | 172.8 | 900 | 8590 | 5.208 |
| GWS 1047 | 11.5 | 5.2 | 59.8 | 475 | 5380 | 7.943 |
| | 15.3 | 8 | 122.4 | 785 | 6820 | 6.413 |
| GWS 1147 | 11.5 | 6.85 | 78.775 | 611 | 5095 | 7.756 |
| | 15.3 | 10.5 | 160.65 | 990 | 6365 | 6.162 |
| GWS Q1280 | 11.5 | 8.6 | 98.9 | 669 | 4793 | 6.764 |
| | 15.3 | 13.3 | 203.49 | 1037 | 5880 | 5.096 |
| APC 12x6E | 11.5 | 10.65 | 122.475 | 780 | 4496 | 6.368 |
| | 15.3 | 16.35 | 250.155 | 1175 | 5350 | 4.697 |

Tabelul II-1 Teste cu diferite elice şi motorul ales [4]

Pentru o eficiență crescută a puterii, precum și pentru a crește masa utilă pe care hexacopterul o poate ridica, am folosit elice de tip GWS 1147. Acestea au o lungime de 11 inch (28 cm) și pas (pitch) de 4.7 inch (12 cm). Elicele utilizate sunt puse alternativ una rotindu-se în sens trigonometric, cealaltă în sens orar.

Figura II-3 Elicele folosite [4]

Elicele sunt făcute din ABS (plastic dur) și sunt suficient de puternice să nu sufere deformări în timpul zborului, dar pot suferi stricăciuni ireparabile dacă în timpul rotirii se lovesc de alte obiecte.

Pentru a controla motoarele fără perii, se folosesc controllere specializate (BLDC controllers), construcția și funcționarea acestora fiind detaliate în subcapitolul următor.

În afară de partea care execută zborul propriu-zis (motoare și elice), aparatul are nevoie și de un senzor de orientare, pentru a cunoaște cât mai exact poziția. Am folosit un sistem inerțial de măsură (senzor de orientare) tridimensional fabricat de firma CH Robotics și comercializat de Pololu. Acest senzor este de fapt un sistem complex format din nouă senzori (trei accelerometre, trei giroscoape și trei magnetometre) și un procesor ARM de 32 de biți capabil să calculeze poziția cu o frecvență de 500Hz sau de 1KHz. Ieșirea senzorului este digitală, acesta comunicând cu placa de bază a hexacopterului printr-o interfața UART-TTL. La ieșire, în funcție de comanda


Figura II-4 Senzorul de orientare UM6-LT [10]

sau de configurația pe care o are, senzorul poate întoarce fie cele trei unghiuri euleriene (roll, pitch si yaw), fie informații brute de la senzori (accelerații, unghiuri).

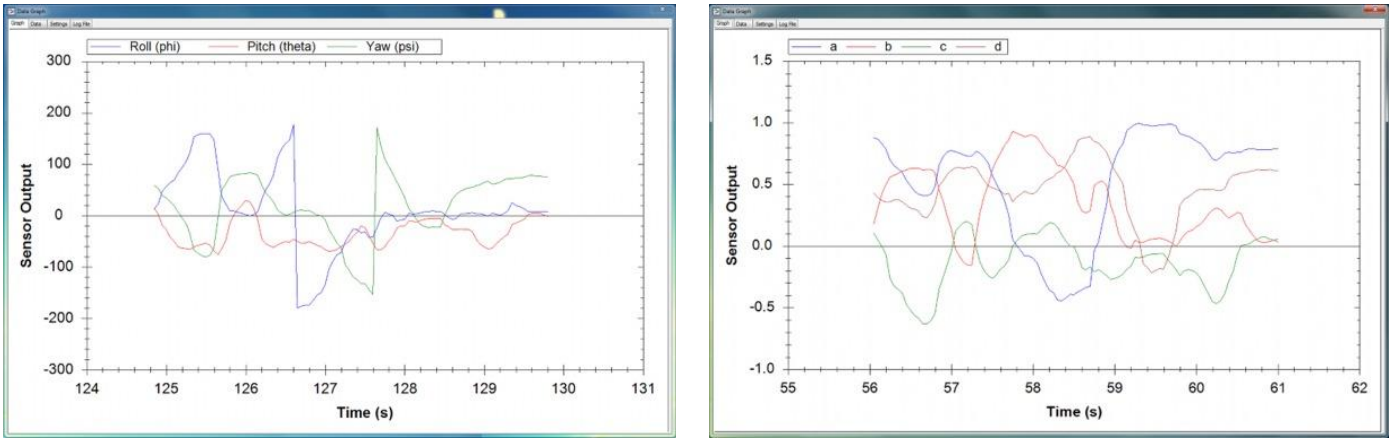


Figura II-5 Ieşirile senzorului: a) unghiurile euleriene b) alte date [10]

Pentru partea de control de la distanță, am folosit două transceivere low-cost de 2.4GHz - Jennic JN5139. Aceste transceivere au o putere de 0.6W [5] și sunt capabile să întrețină o comunicație bidirecționala pe o distanță de până la 4 Km „line of sight". Ele rulează peste un protocol ZigBee modificat de producător, numit JenNet®, și au un procesor ARM de 16MHz care printre altele are un modul UART. Procesoarele transceiverelor au fost programate să transmită informația într-un mod numit UART-over-JenNet®, adică practic să țină locul unui cablu UART/serial foarte lung. Baudrateul UART programat pentru comunicație este de 57600 de biți pe secundă, viteză suficientă pentru a asigura controlul și telemetria hexacopterului.

Figura II-6 Transceiverul ZigBee [5]

Controllerele de motoare, senzorul de orientare, transceiverele, precum și alți senzori și sisteme aflate pe hexacopter sunt legate între ele printr-un punct fix: placa de bază a hexacopterului. Aceasta este construită cu unul dintre cele mai performante microcontrolere Microchip aflate pe piață - PIC32® [6] – care are suficientă viteză și porturi pentru a fi capabil să execute toate operațiile. Acesta este un microcontroler pe 32 de biți și prezintă o frecventă de rulare a instrucțiunilor de 80MHz. Are 512 KB de memorie de program și 128 KB de memorie RAM. Dintre periferice putem enumera 4 canale de SPI, 5 canale de $I^2C$, 16 canale analogice, 85 de pini de intrare-ieșire și 8 canale DMA.

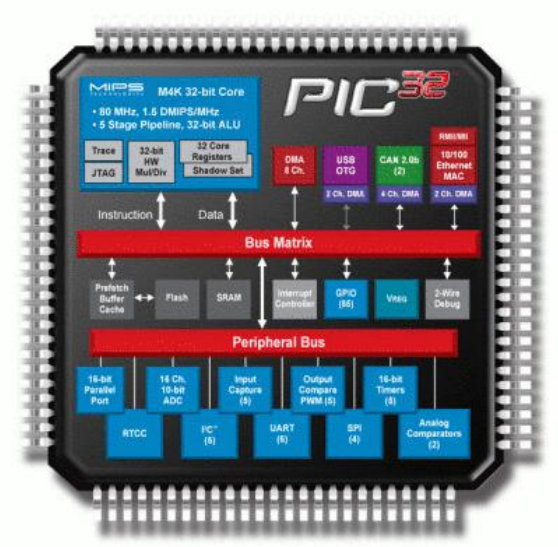


Figura II-7 Schema bloc a microcontrolerului PIC32® [10]

O schemă de principiu a părţii electronice, incluzând placa de bază, este prezentată în Figura II-8.

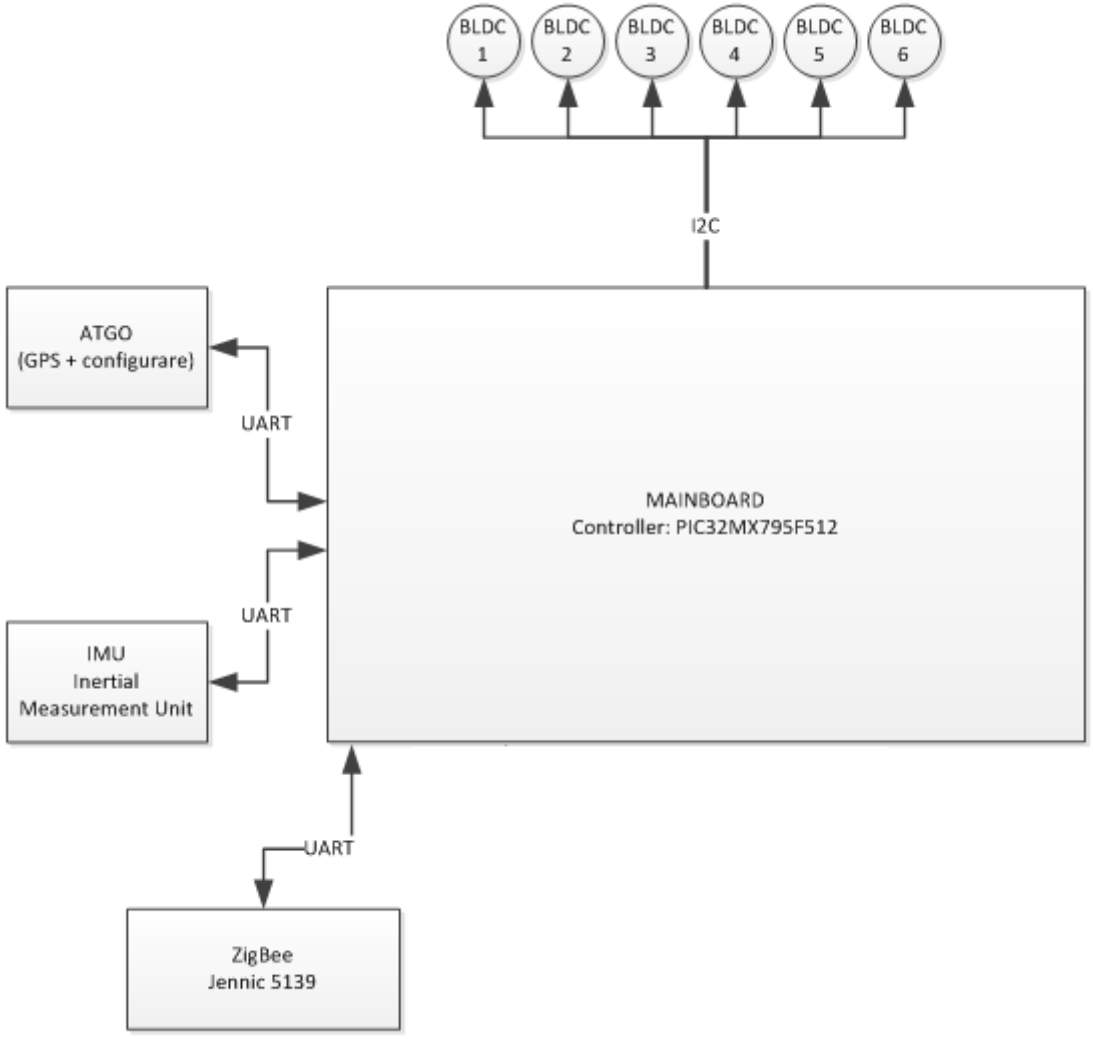


Figura II-8 Schema bloc a sistemului electronic de comandă şi control

Prima versiune a plăcii de bază, numita și versiunea de test, a fost proiectată în Cadence OrCAD 16.2, cu o schemă simplificată, ce avea doar conexiunile către senzori, motoare şi transceiver-ul ZigBee. Acest lucru a fost realizat pentru a testa schema folosită daca este validă, pentru a testa componentele și pentru a putea crea un schelet de inițializare al aplicației. Această placă a fost făcută prin metoda UV, care după câteva încercări nereușite a funcționat. Problema a fost distanța foarte mică dintre pinii microcontrolerului și grosimea foarte mică a acestora, iar în momentul în care transferul termic al cernelii era terminat şi ridicam folia, câteva din trasee rămâneau netransferate. Layoutul acesteia generat în OrCAD 16.2 și o poză cu ea sunt prezentate în Figura II-9 şi în Figura II-10 (mufa asemănătoare cu RJ-45 din poză reprezintă adaptorul de la programatorul Microchip Real ICE® [7] la mufa SIP):

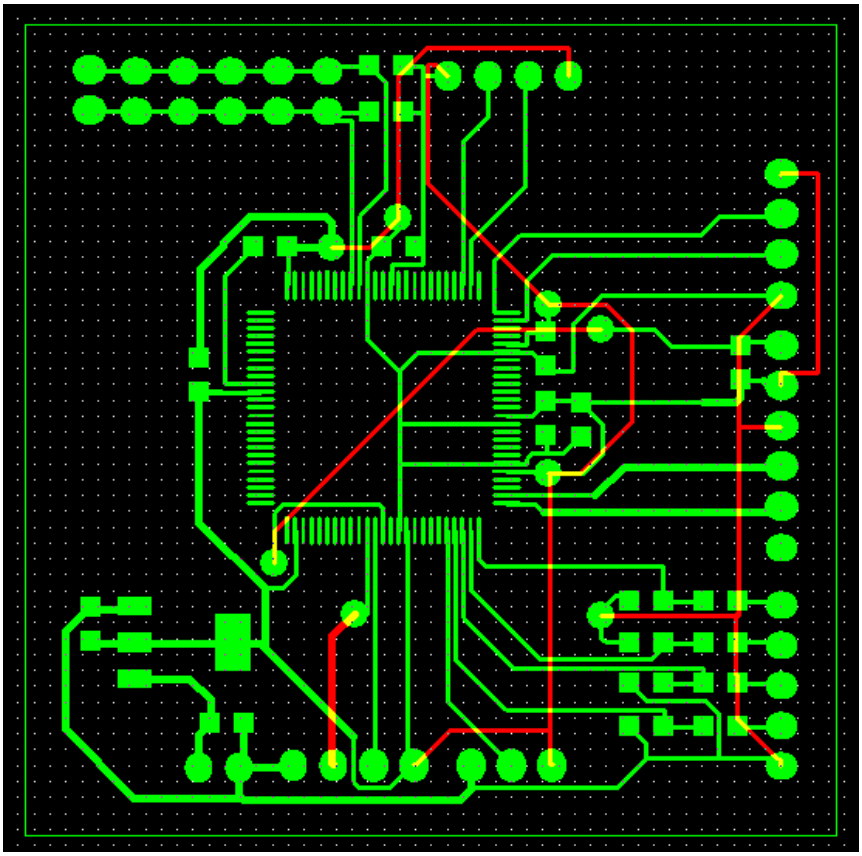

Figura II-9 Layout-ul plăcii de bază cu PIC32®

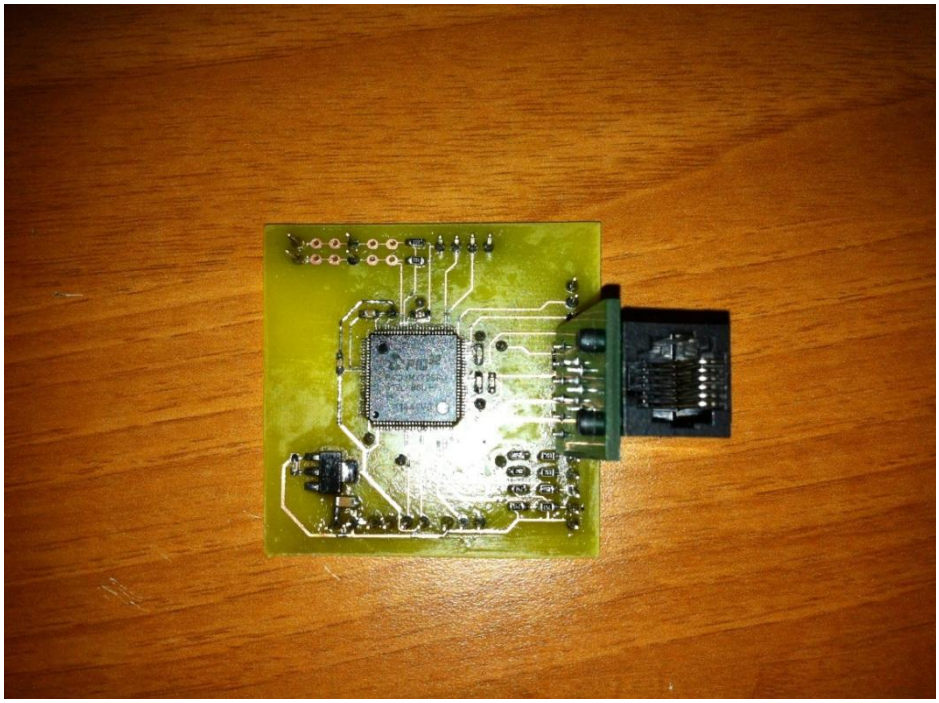

Figura II-10 Placa de bază cu PIC32® şi adaptor pentru Real ICE®

# 2. Controlul fără senzori al motoarelor fără perii[3]

Motoarele fără perii au în general trei faze cu o structură stea sau triunghi. Aceste motoare pot fi de două feluri: inrunner sau outrunner. După cum spune și numele, motoarele inrunner au rotorul la interior și statorul (înfășurările) la exterior, iar motoarele outrunner au rotorul in exterior și înfășurările în interior. Diferențele între inrunner și outrunner sunt în mare parte legate de cuplul și de turația maximă exercitate. Inrunnerele pot dezvolta turații foarte mari, mai ales cu avansare de fază (se poate ajunge până la 5-600.000 RPM scăzând cuplul foarte puternic), în timp ce outrunnerele au în general turații mai mici și cupluri puțin mai mari. Un alt avantaj al outrunnerelor este rotorul exterior, care antrenează curenți de aer în jurul acestuia, curenți care ajută la răcirea înfășurărilor de pe stator.

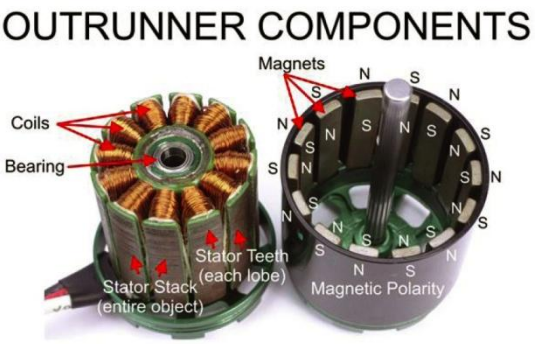


Figura II-11 Motorul outrunner

Motoarele fără perii pot fi cu senzori (Hall) sau fără senzori. Nevoia de senzori este dată de controlul trapezoidal al motorului: controller-ul trebuie să cunoască exact momentul în care trebuie schimbat sectorul de comutație pentru a învârti cu eficiență maximă motorul. Senzorul Hall măsoară fluxul magnetic și atenționează controllerul când rotorul este într-o anumită poziție (de obicei la 30 de grade). Controllerul apoi calculează în cât timp trebuie schimbat sectorul în funcție de aceste 30 de grade.

Un motor fără perii prezintă un număr de perechi de poli (magnetici). Acest număr este foarte important pentru a stabili numărul de rotații al motorului. O rotație mecanică reprezintă o rotație electrică împărțită la numărul de perechi de poli. Pentru a executa o rotație electrică (360 de grade electrice) motorul parcurge 6 sectoare de comutație, fiecare a câte 30 de grade. În fiecare sector, o fază este legată la tensiunea pozitivă, o altă fază este legată la tensiunea negativă (sau masa), și o fază este lăsată liberă. Sensul de rotație și rotația propriu-zisă se fac în funcție de cum sunt comutate aceste sectoare. În Figura II-12 avem reprezentate sectoarele de la 1 la 6. Pentru a roti într-un sens, se parcurg aceste sectoare de la 1 la 6, iar pentru a roti în sens opus, sectoarele sunt parcurse de la 6 la 1.

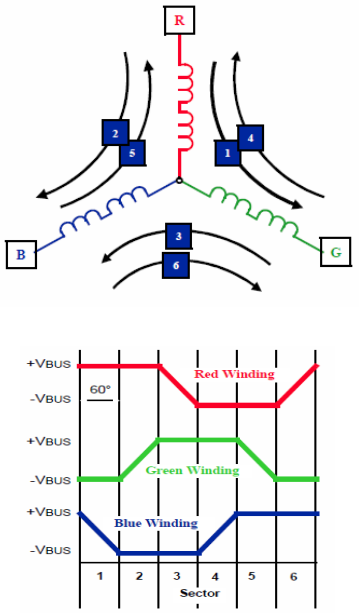


Figura II-12 Control trapezoidal [10]

Controlul fără senzori este posibil tocmai datorită faptului că la un anumit moment, una din faze este lăsată liberă. Tensiunea de pe aceasta fază reprezintă tensiunea forței electromagnetică (Back-EMF) exercitată pe înfășurarea respectivă, datorită rotației exercitate de celelalte două

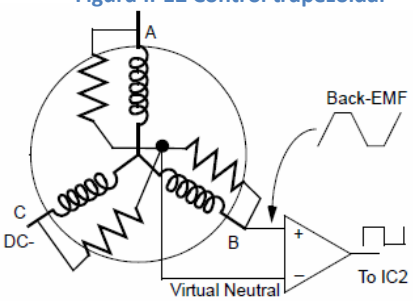


Figura II-13 Reconstrucția neutrului motorului [10]

[3] Capitol scris în colaborare cu Microchip®

înfășurări conectate la tensiune. Un controller de motor fără perii și fără senzori citește (folosind un canal analog) această tensiune și stabilește momentul în care trebuie făcută comutația de sector. Controller-ul își calculează punctul neutru virtual și compară aceste tensiuni. Punctul neutru reprezintă punctul în care toate cele 3 faze ale motorului sunt legate împreună. Acest punct în general nu este disponibil pentru măsură, așa că se reconstruiește. Construirea punctului virtual se poate face în două feluri: fie cu 3 rezistori de aceeași valoare, conectați în stea, fiecare reprezentând cele 3 faze ale motorului, fie direct în soft de către controller. Metoda celor 3 rezistoare este o metodă mai costisitoare, cu rezultate deloc diferite de metoda software. Metoda software presupune faptul că tensiunile pe toate cele 3 faze sunt cunoscute: ($V_{bus+}$), ($V_{bus-}$) și tensiunea de BEMF, măsurată de către controller cu un canal analogic. Tensiunea virtuală din punctul comun al înfășurărilor este media aritmetică a celor 3 tensiuni: ($V_{bus+}$), ($V_{bus-}$) si ($V_{bemf}$).

Metoda folosită în controlul motoarelor dronei este bazată pe detectarea punctului de trecere prin zero al tensiunii de BEMF (zero crossing detection). Trecerea prin zero se referă la momentul în care tensiunea de BEMF ajunge la jumătatea valorii tensiunilor aplicate pe celelalte două faze. În acel moment bine stabilit motorul a terminat de mers exact 30 de grade din sectorul respectiv, rămânându-i de executat exact încă 30 de grade până să trebuiască schimbat sectorul de comutație. Pentru a realiza această măsurătoare, din momentul în care se schimbă un sector, se măsoară (numără) cât timp trece până la detectarea zero-crossing-ului, după care se așteaptă exact aceleași unități de timp, și se schimbă din nou sectorul, după care procesul se reia. O tehnică de mărire a vitezei peste viteza maximă la care poate merge motorul în cuplu maxim este metoda de avansare de fază (phase advance). Avansarea de fază presupune schimbarea sectorului înainte de scurgerea unităților de timp măsurate.

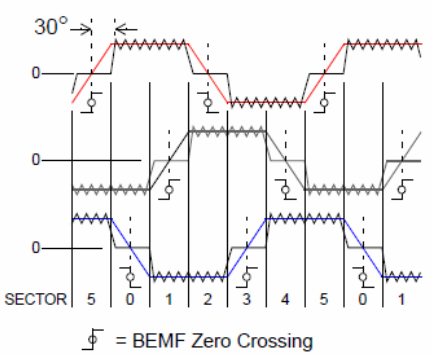


Figura II-14 Principiul de zero-crossing [10]

O altă problemă care apare reprezintă filtrarea tensiunii de BEMF. În practică, pentru a controla turația motorului fără avansare de fază, tensiunile aplicate pe cele două faze nu sunt continue, ci modulate în PWM. Acest lucru induce zgomot și în faza lăsată liberă (cea de pe care se citește tensiunea de BEMF), zgomot care trebuie să nu influențeze măsurătoarea zero-crossing-ului. Filtrarea zgomotului din tensiunea de BEMF se poate face în două feluri: analogic și digital. Filtrul analogic prezintă dezavantajul că faza introdusă în semnalul de BEMF poate fi necunoscută (daca se operează la frecvențe variabile) și dezavantajul complet neglijabil că ocupă spațiu pe PCB. Filtrul digital are avantajul că faza introdusă se cunoaște exact, nu ocupă spațiu suplimentar, și dacă procesorul folosit are suficiente resurse, dezavantajele sunt limitate considerabil. Filtrul introdus în lucrarea de față este un filtru digital, poartă numele de filtru majoritar și este exemplificat în Tabelul II-2 Filtrarea majoritară – funcţia majoritate de mai sus pentru trei intrări de filtrat (A, B și C) și o ieșire (Majoritate).

| A | B | C | Majoritate |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 |

Tabelul II-2 Filtrarea majoritară – funcţia majoritate [10]

Cele trei intrări (A, B si C) nu reprezintă fazele motorului, ci eșantioane consecutive dintr-un singur semnal. În aplicația de față, semnalul filtrat este semnalul comparatorului soft, ce compară tensiunea de BEMF cu tensiunea virtuală din punctul comun al înfășurărilor. Detecția punctului de trecere prin zero apare atunci când la ieșirea filtrului semnalul se inversează (din 0 în 1 sau din 1 în zero, depinzând de sector).

Controllerele de motoare brushless sunt comandate de placa centrală a hexacopterului printr-o magistrală $I^2C$, fiecare controller având un identificator separat în rețea. Controllerele construite suportă o tensiune de intrare de până la 20 V și curenți de până la 20 A, iar tranzistoarele folosite sunt tranzistoare MOS cu rezistența internă $R_{DS}$ de 3.8 mΩ (canal P) și 3.6 mΩ (canal N). Acest lucru asigură o performanță ridicată în comparație cu alte controllere existente pe piață, care nu coboară rezistența $R_{DS}$ a tranzistoarelor sub 20 mΩ.

In paginile următoare este prezentata schema electronica de funcționare a controller-ului pentru motoare fără perii.

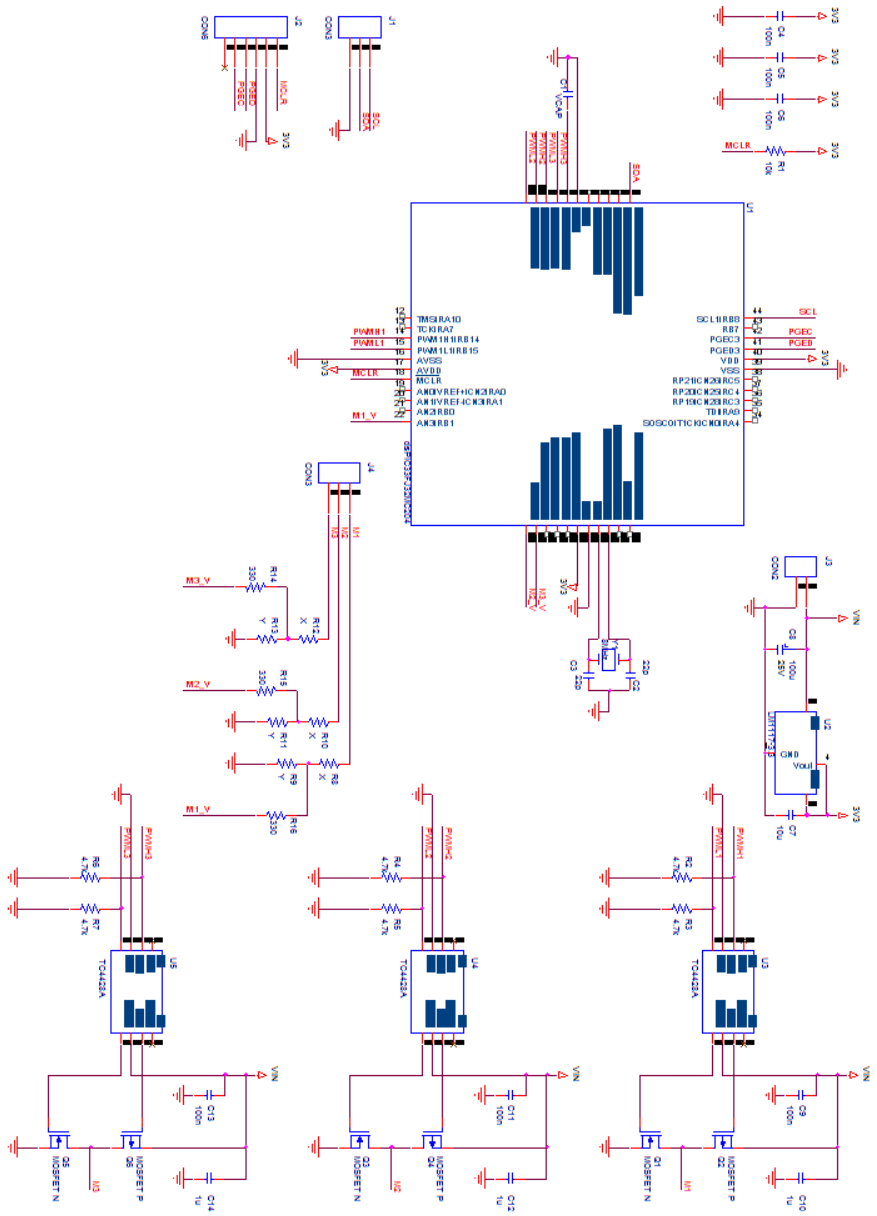

Figura II-15 Schema controllerului de motoare brushless

În partea din stânga a schemei se găsesc conectorii pentru interfața $I^2C$ și interfața de programare ICSP® a microcontrolerului dsPIC®. De asemenea, putem găsi și 3 condensatoare de decuplare, condensatoare care în proiectarea PCB-ului sunt puse foarte aproape de pinii microcontrolerului. Urmează apoi microcontrolerul, dsPIC33FJ32MC204 [8] produs de Microchip, care are nucleu de DSP și modul pentru motor control.

Pentru a alimenta microcontrolerul dsPIC® am folosit un regulator LM1117, regulator ce poate lua ca intrare tensiuni de pana la 18V, reglând la ieșire o tensiune de 3.3V necesară microcontrolerului. Curentul pe care îl suportă regulatorul este de până la 500 mA, dar tot ansamblul de componente ce lucrează pe tensiunea de 3.3V nu depășește 50 mA.

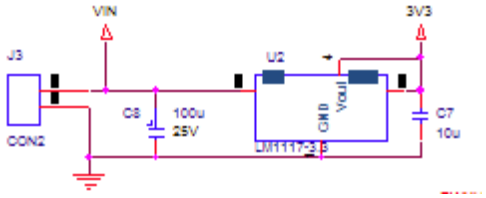


Figura II-16 Alimentarea controllerului

O altă parte importantă a schemei este puntea de control cu 3 brațe a motorului brushless. În Figura II-17 este prezentat doar un braț al punţii. Fiecare braț primește 2 semnale PWM de la microcontroler, semnale destinate pentru controlul celor două tranzistoare MOS. Deoarece microcontrolerul nu suportă curenți mari pe pini, și pentru a putea deschide în totalitate ambii tranzistori la capacitate maximă, a fost introdus un driver high+low side pentru tranzistoare MOS: TC4428A [9] produs tot de Microchip. Spre deosebire de alte drivere, TC4428A are avantajul de a se alimenta direct la tensiunea de alimentare a tranzistoarelor (implicit, a motoarelor). Astfel, fiecare braț are o ieșire controlabilă în punte numită M1, M2 respectiv M3, ieșire pe care se conectează motorul brushless.

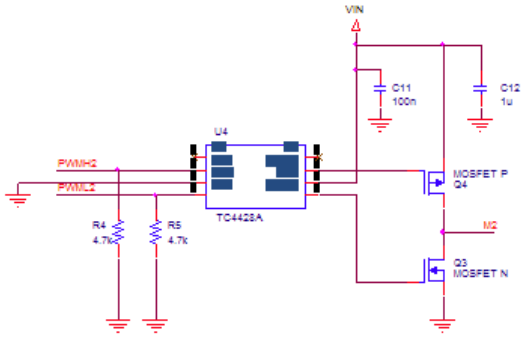


Figura II-17 Braţul punţii controllerului

O ultimă parte importantă a schemei o reprezintă rețeaua de divizoare rezistive care asigură coborârea nivelului de tensiune pentru a citi feedback-ul. Toate cele 3 divizoare rezistive sunt identice și sunt conectate pe cate una din fazele motorului. Acestea asigură coborârea nivelului de tensiune de la maxim 17 la maxim 3.3V pentru a putea fi citită de convertorul analog-digital de care dispune microcontrolerul.

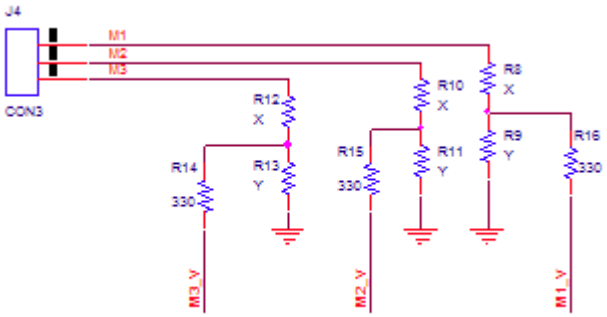


Figura II-18 Divizorul rezistiv al controllerului

În continuare sunt prezentate câteva aspecte practice ale controllerului de motoare brushless realizat: layoutul PCB şi rutinele software.

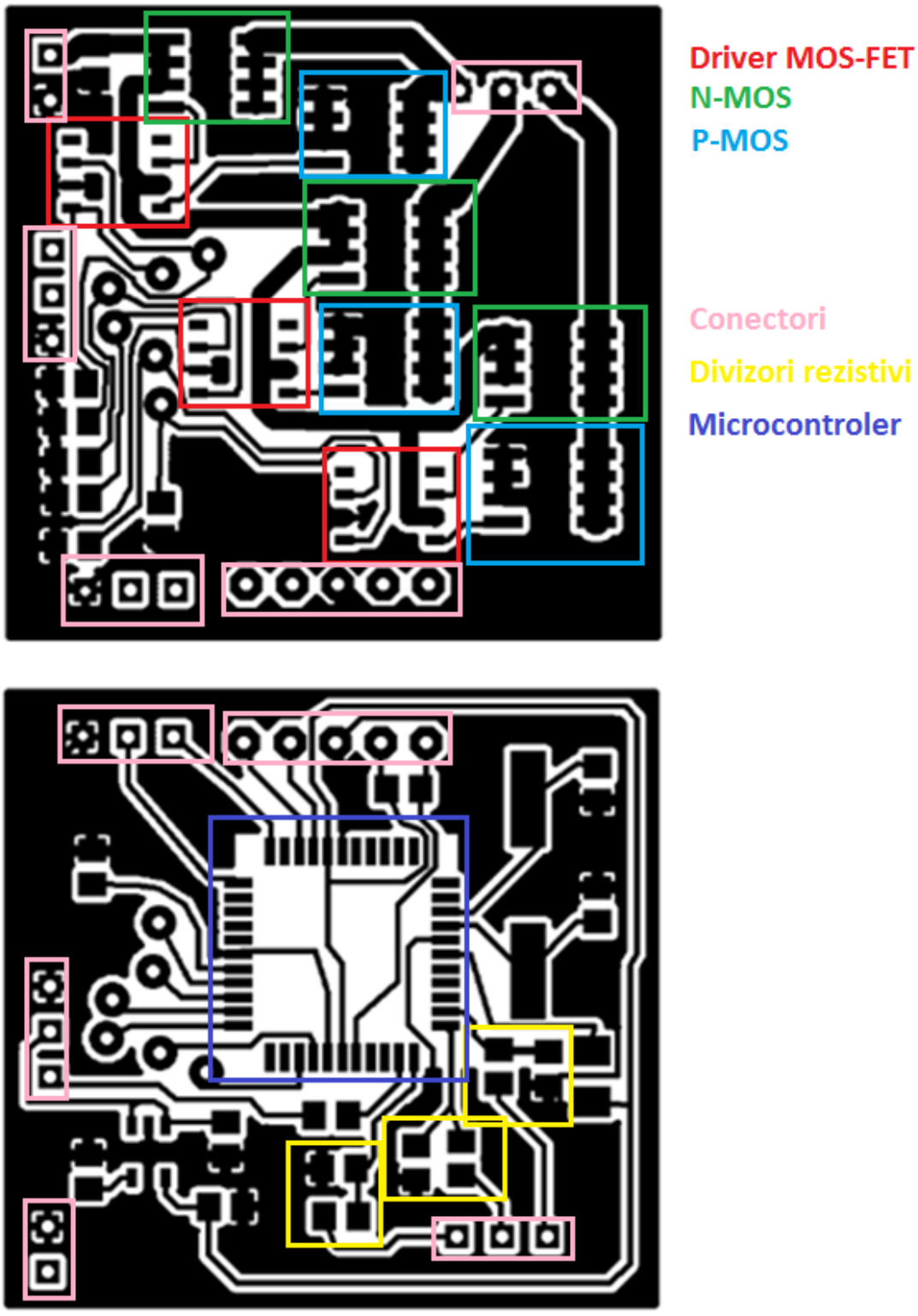


Figura II-19 Layoutul controllerului de BLDC

În Figura II-20 Rutina principală a softului controllerului de motoare brushless şi exemplificarea maşinii de stări, Figura II-21 Rutine pentru deservirea rutinei principale ce realizează controlul efectiv al motorului, Figura II-22 Rutinele de iniţializare şi start ale motorului şi Figura II-23 Rutina principală ce realizează controlul efectiv al motorului este prezentat algoritmul de control sub formă de scheme logice. Schemele sunt în limba engleză și sunt extrase din nota de aplicație AN1160[4] a firmei Microchip, notă de aplicație pe care am dezvoltat-o personal și care este disponibilă public pe situl www.microchip.com. Singura diferență între algoritmul prezentat în scheme și algoritmul prezent în controllerele de pe hexacopter o reprezintă potențiometrul care pe controllerele de la hexacopter este înlocuit cu o variabilă primită prin intermediul magistralei $I^2C$. Acest lucru nu afectează în niciun fel modul de funcţionare sau logica din spatele controllerului.

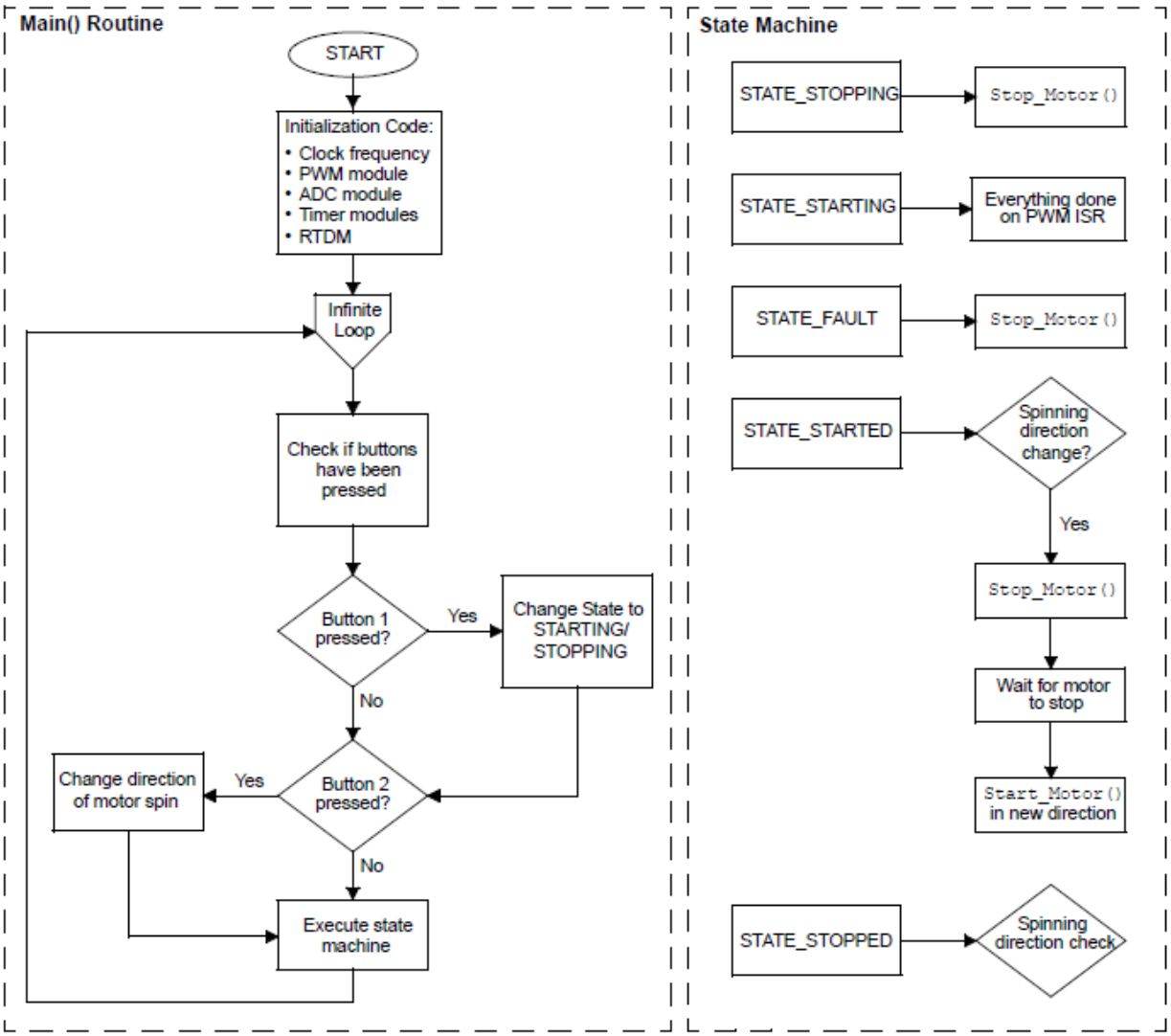


Figura II-20 Rutina principală a softului controllerului de motoare brushless şi exemplificarea maşinii de stări [10]

[4] AN1160 - Sensorless BLDC Control with Back-EMF Filtering Using a Majority Function

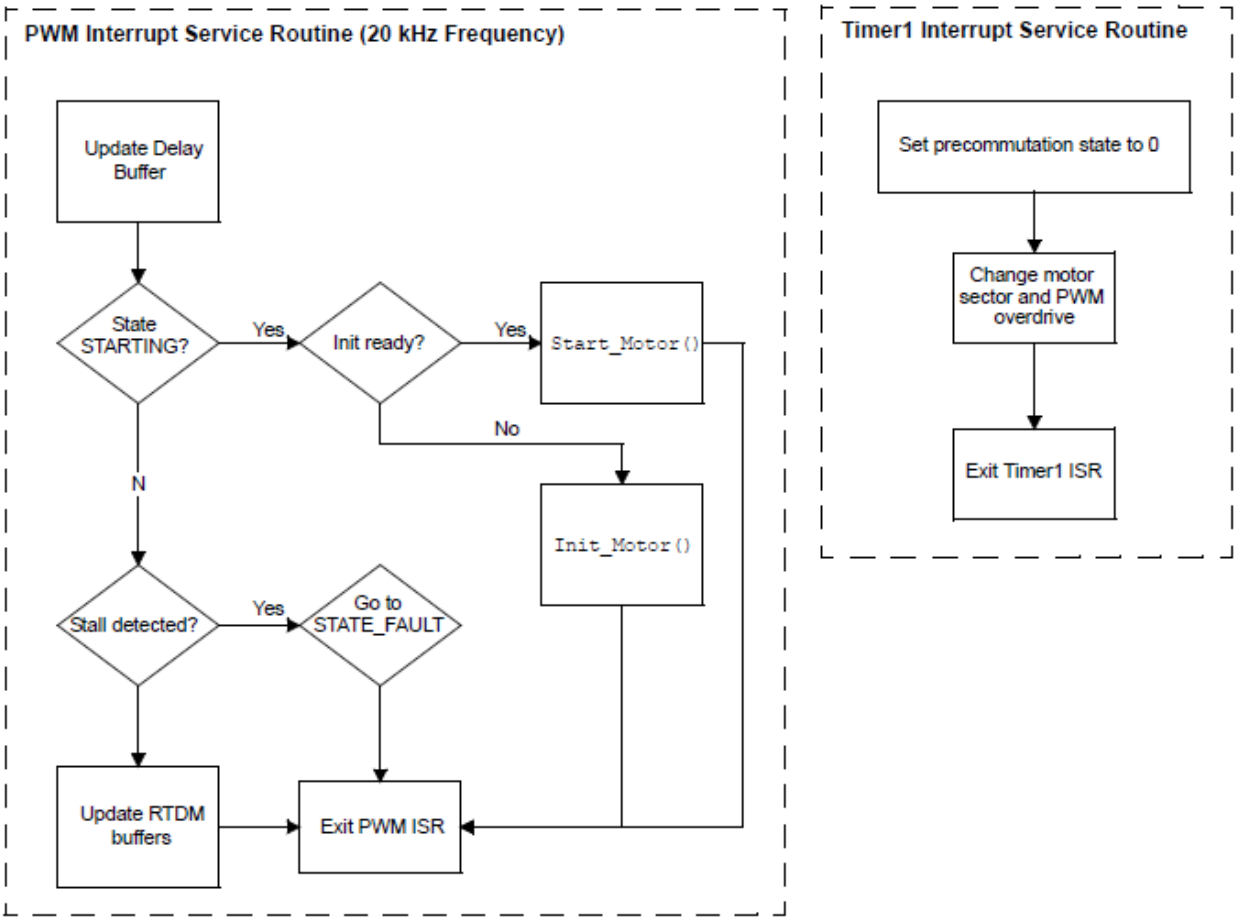


Figura II-21 Rutine pentru deservirea rutinei principale ce realizează controlul efectiv al motorului [10]

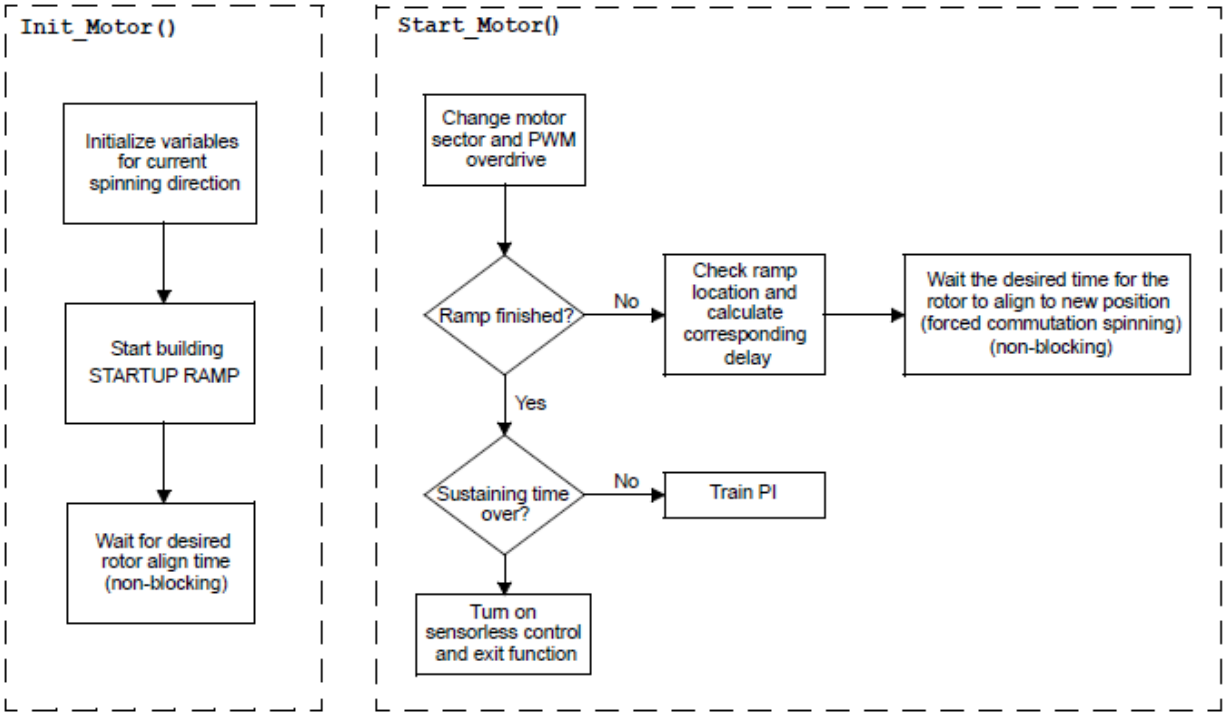


Figura II-22 Rutinele de iniţializare şi start ale motorului [10]

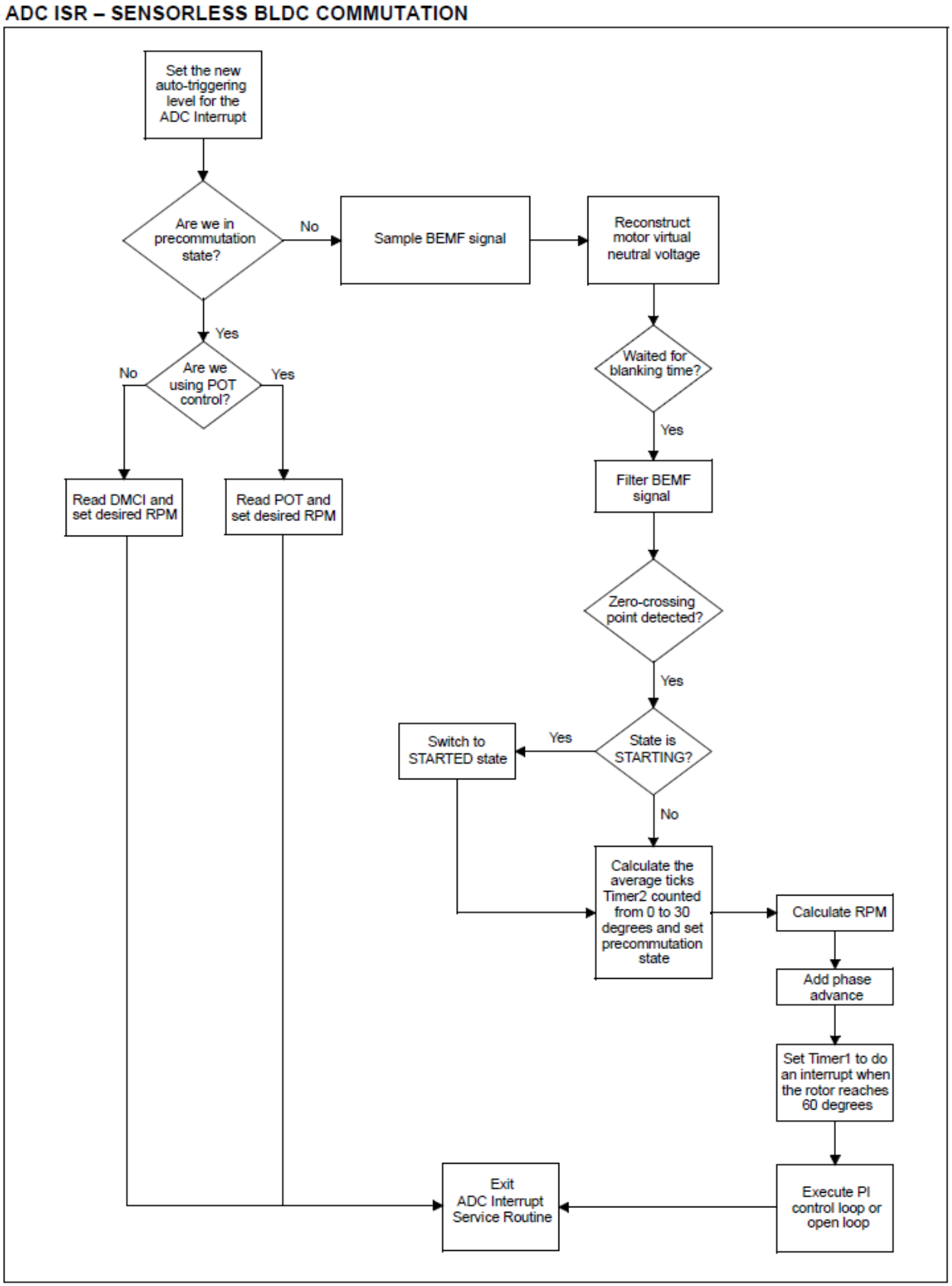


Figura II-23 Rutina principală ce realizează controlul efectiv al motorului [10]

# 3. Descrierea zborului tridimensional cu 6 motoare

Deoarece hexacopterul poate zbura, înseamnă că față de o mașină acesta are cu un grad de libertate în plus. Dar pentru a obține toate cele trei grade de libertate (pe toate cele trei axe reale: X, Y si Z) trebuiesc, pentru început, descrise mișcările de bază ce trebuiesc efectuate. În următoarea analiză considerăm accelerația gravitațională g acționând paralel cu axa Z.

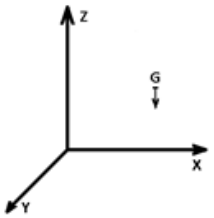


Figura II-24 Spaţiul folosit

În primul rând, hexacopterul are 6 elice, fiecare orientate să sufle aer perpendicular pe suprafața sa. Deci orice mișcare nu se poate face altfel decât prin variația vitezei diferitelor elice. Întâi de toate, vom defini "fața" hexacopterului ca fiind una din cele 6 elice, și pentru demonstrație o vom marca prin culoarea roșie.

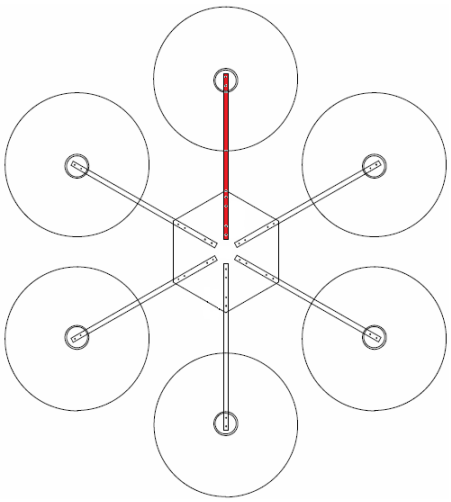

Figura II-25 Vedere de sus

Cum a fost precizat și mai devreme, dacă rotim toate elicele într-un sens (presupunem în sens trigonometric) atunci hexacopterul va obține un moment de rotație în sensul acelor de ceas, și pierdem două avantaje: un grad de libertate, și obținem un aparat care se învârte într-o parte fără control. Totuși acest lucru poate fi foarte ușor evitat folosind elice de sens opus, și învârtind trei elice într-un sens și trei în celălalt sens. Ordinea elicelor sau așezarea acestora nu este importantă pentru a elimina momentul de rotație nedorit.

Dar dacă nu așezăm elicele într-o ordine convenabilă putem pierde cel de-al treilea grad de libertate. De aceea am optat pentru o așezare a elicelor după o regulă foarte simplă: o elice care se învârte în sensul acelor de ceas trebuie sa aibă ca vecini două elice care se învârt în sens trigonometric. Astfel obținem o configurație ce ne permite folosirea tuturor celor 3 grade de libertate, și obținem o idee foarte simplă de cum se obține mișcarea de rotație după axa Z.

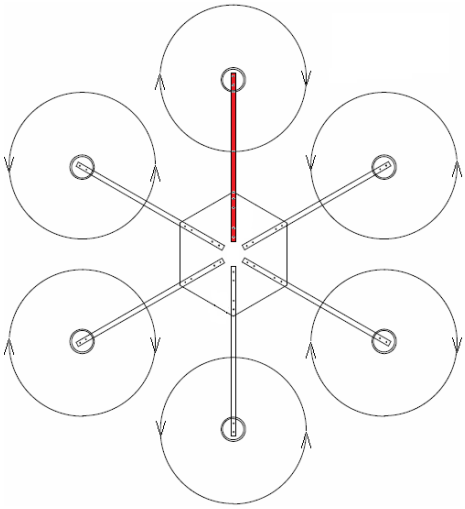

Figura II-27 Vedere de sus a sensurilor de rotaţie ale elicelor

Cele trei mișcări posibile în aer sunt mișcările de rostogolire (roll), înclinație (pitch) și girație (yaw), mișcări ce reprezintă de fapt cei trei parametri care descriu orientarea într-un spațiu euclidian. Deoarece la hexacopter, și în general la aparatele care au un zbor cu elice orizontale, mișcările de rostogolire și înclinație au o parte comună si sunt foarte ușor confundabile, pentru aceste aparate mai avem o mișcare ce reglează altitudinea (take-off / landing).


Figura II-26 Cele trei unghiuri euleriene şi semnificaţia lor [10]

Mișcarea de giratie (yaw) în acest moment este cea mai simplă de explicat datorită formei hexagonale a aparatului și datorită felului în care se învârt elicele.

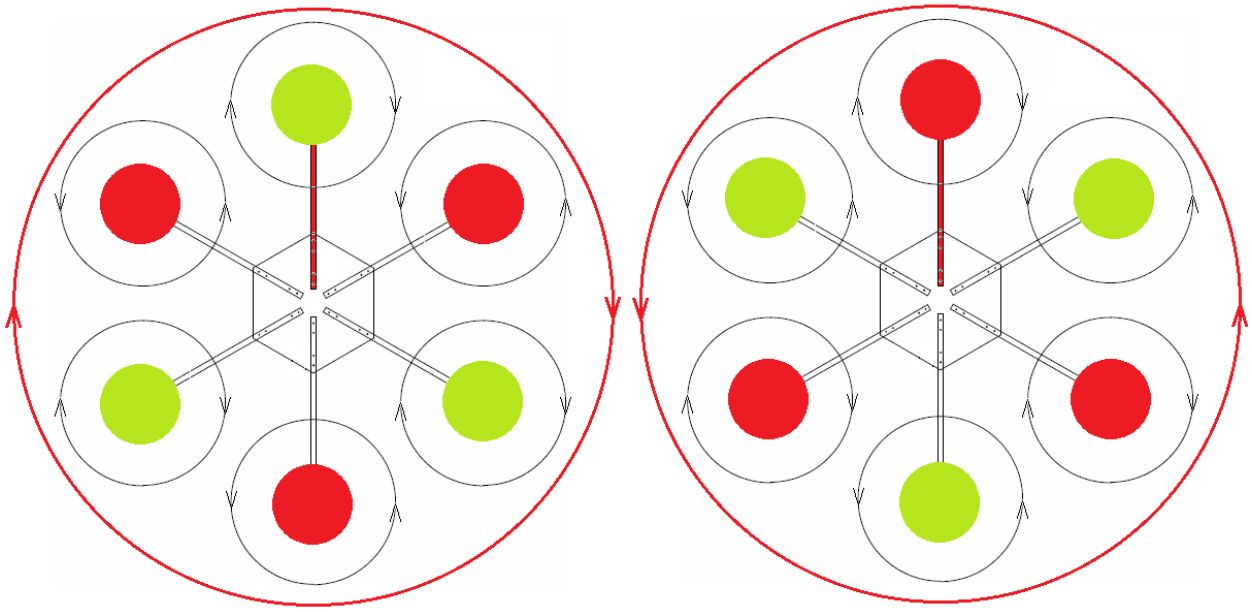

Figura II-28 Mişcarea de giraţie: a) sensul acelor de ceas b) sens trigonometric

Toată ideea mișcării stă în Figura II-28 de mai sus, unde sunt prezentate ambele sensuri de mișcare (sensul acelor de ceas și sensul trigonometric) în funcție de viteza de rotație a elicelor. Întâi de toate, pentru a defini învârtirea normală a elicelor în figură, elicele marcate cu verde se învârt mai încet decât de obicei, iar cele marcate cu roșu se învârt mai repede decât normal.

Spre exemplu, pentru a executa o rotație în sensul acelor de ceas (subfigura din stânga), vom învârti cu o viteză mai mare (puțin) elicele care se învârt în sens trigonometric, și cu o viteză mai mică elicele care se învârt în sensul acelor de ceas.

Pentru a executa mișcarea nu era nevoie decât ca o parte din elice (fie cele de un sens, fie cele de sens opus) să își schimbe turația, dar pentru a nu modifica altitudinea aparatului (a nu afecta un alt grad de libertate), propulsia rezultantă datorită celor două tipuri de elice trebuie echilibrată.

Mișcările de înclinație și rostogolire se pot descrie ca fiind o aceeași mișcare deoarece principiul care stă la baza acestora este identic.

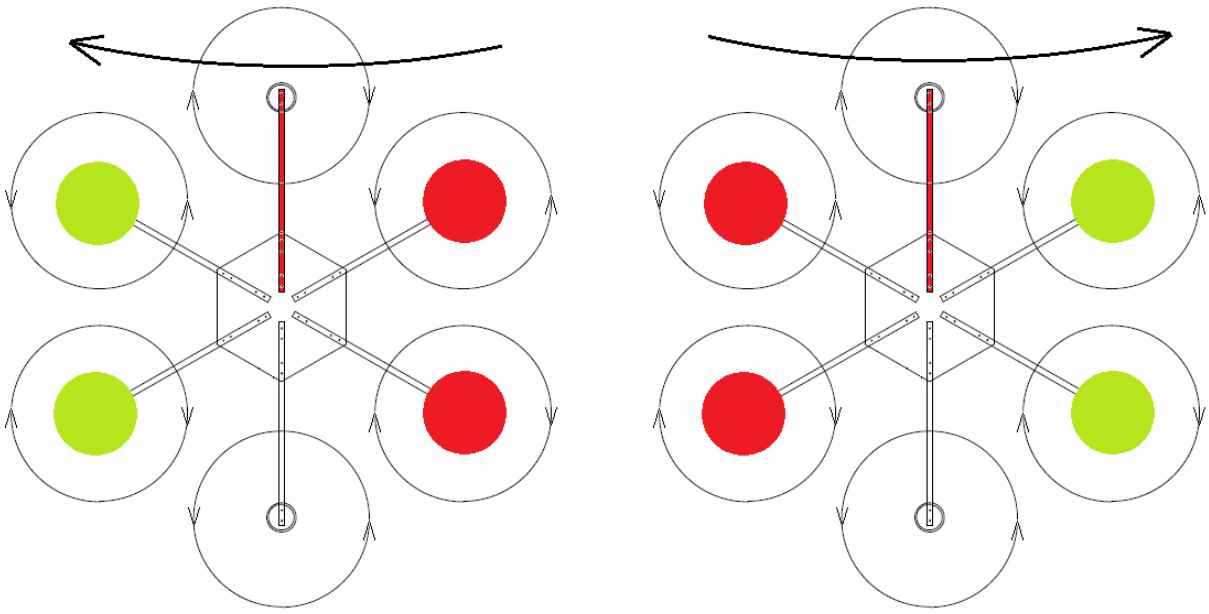

Figura II-29 Mişcarea de înclinaţie

Din nou, elicele cu roșu se învârt mai repede decât normal, iar elicele cu verde se învârt mai încet. Elicele fără culoare păstrează viteza normală de rotație pentru ca mișcarea (rostogolire în acest caz) să nu afecteze și înclinația.

Analog, înclinația se poate realiza astfel:

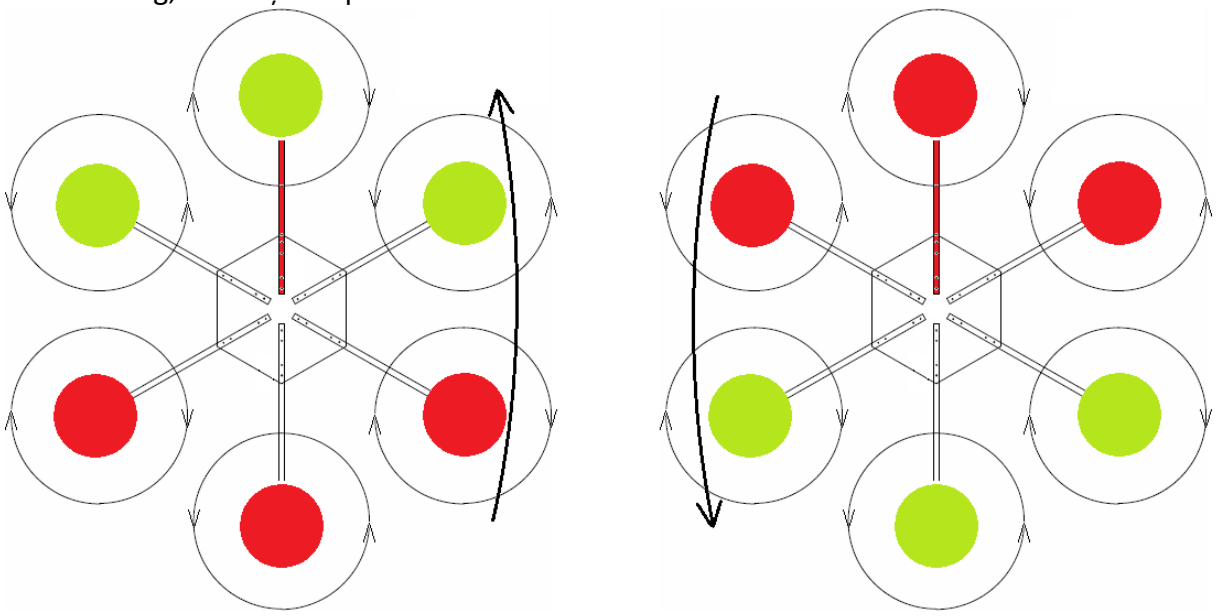

Figura II-30 Mişcarea de rostogolire cu toate cele 6 elice

La înclinație se poate de asemenea realiza mișcarea și cu schimbarea turației a mai puținor elice, fără a afecta dreapta după care se face mișcarea, lucru imposibil rostogolirii.

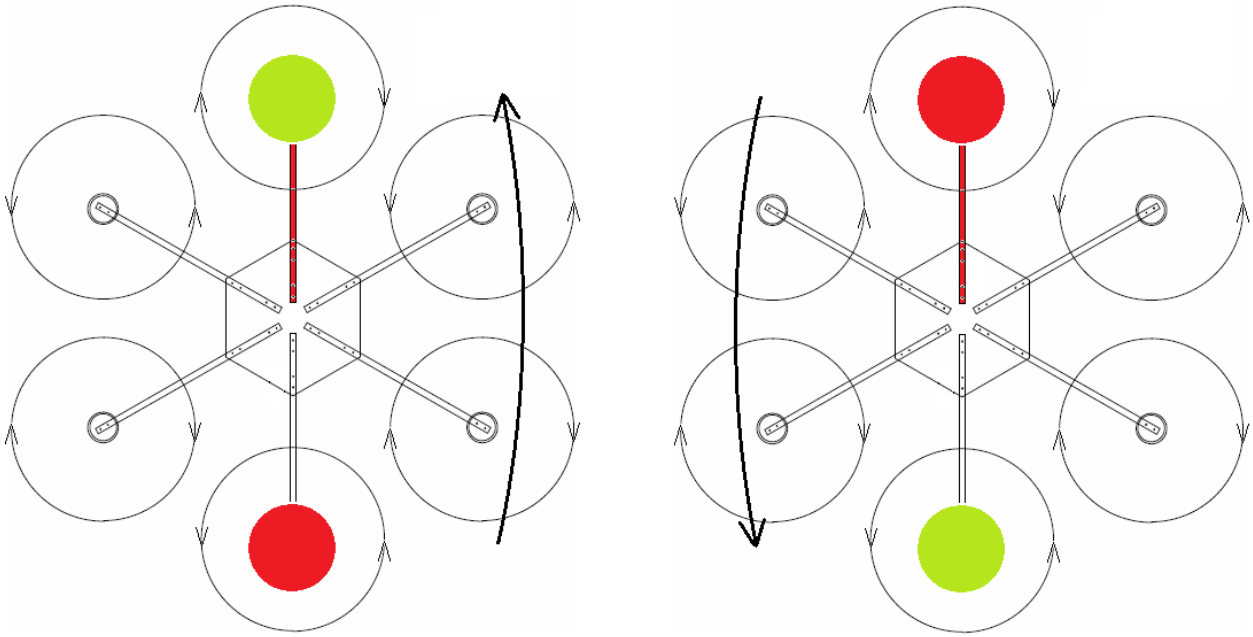

Figura II-31 Mişcarea de rostogolire cu 2 elice

O ultimă mișcare de bază o reprezintă mișcarea de-a lungul axei Z, mișcarea de urcare și coborâre (take-off and landing). Aceasta este una dintre cele mai simple mișcări, și poate fi făcută în diferite feluri: cu 2, cu 4 sau cu 6 elice. Principiul de bază al urcării este că forța de propulsie trebuie să fie mai mare pentru ca accelerația să fie pozitivă. Analog, la coborâre accelerația trebuie să fie negativă. De asemenea, trebuie avut în vedere faptul că trebuie ca în momentul urcării/coborârii (doar mișcarea de baza) aparatul nu trebuie să aibă altfel de înclinație (din acest motiv se poate face doar cu 2, 4 sau 6 elice, și nu și cu 3 si 5).

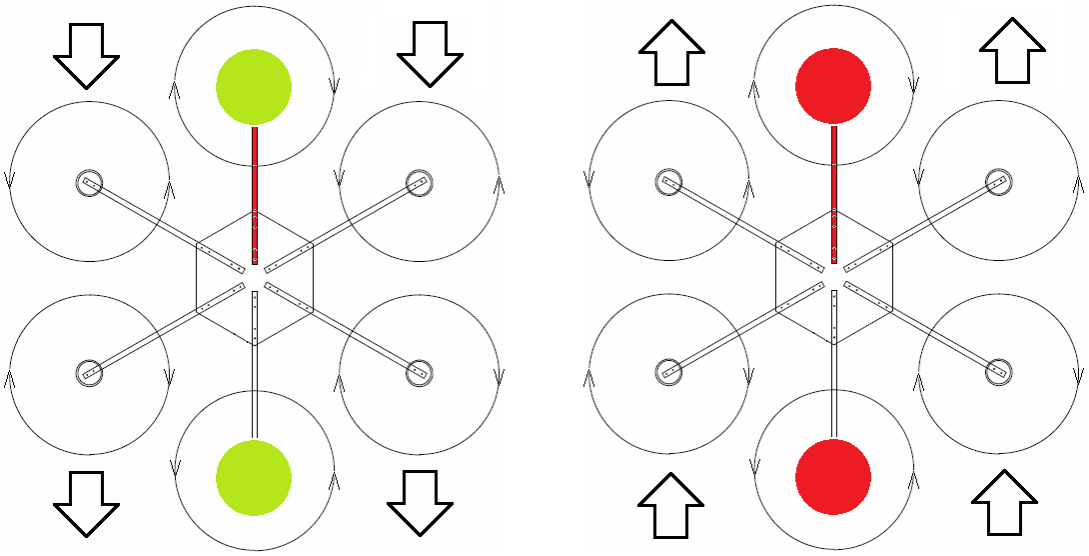

Figura II-32 Mişcarea de urcare şi coborâre cu 2 elice

Pentru cazul în care urcarea/coborârea se face doar cu 2 elice, trebuie avut în vedere că aceste două elice trebuie să fie reciproce (pe aceeași dreaptă care trece prin centrul aparatului). Cu acest criteriu se respectă și criteriul mișcării de girație.

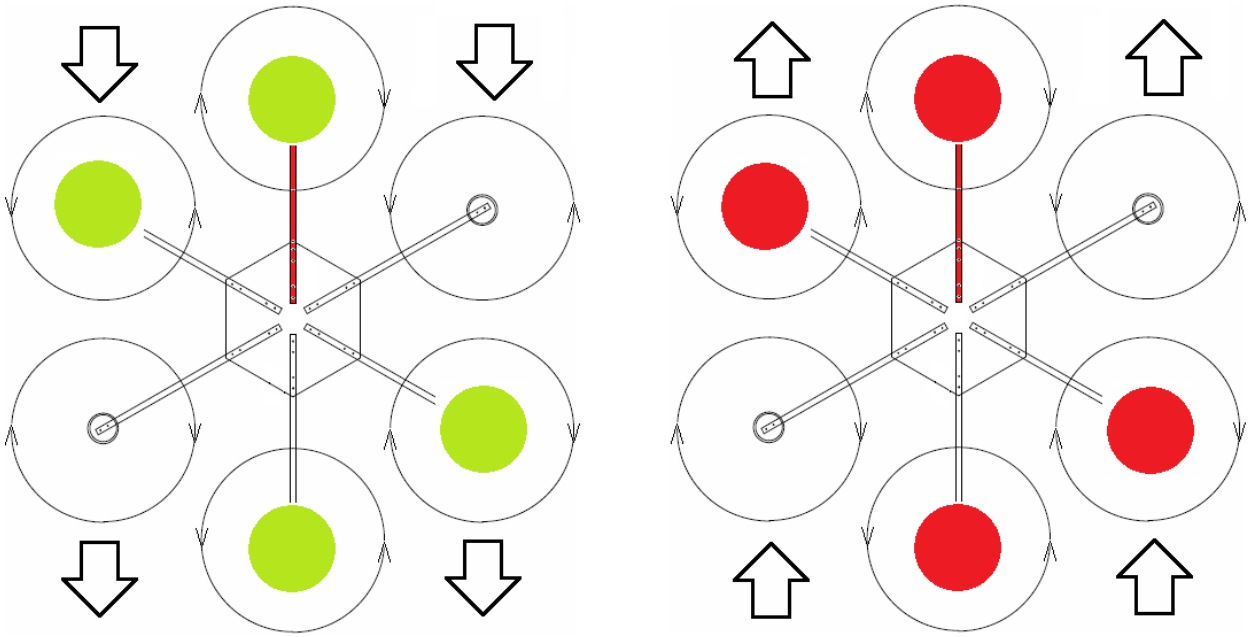

Figura II-33 Mişcarea de urcare şi coborâre cu 4 elice

La fel ca în cazul urcării/coborârii cu 2 elice, în momentul în care folosim 4 elice trebuie să avem în vedere ca acestea să fie reciproce 2 câte 2.

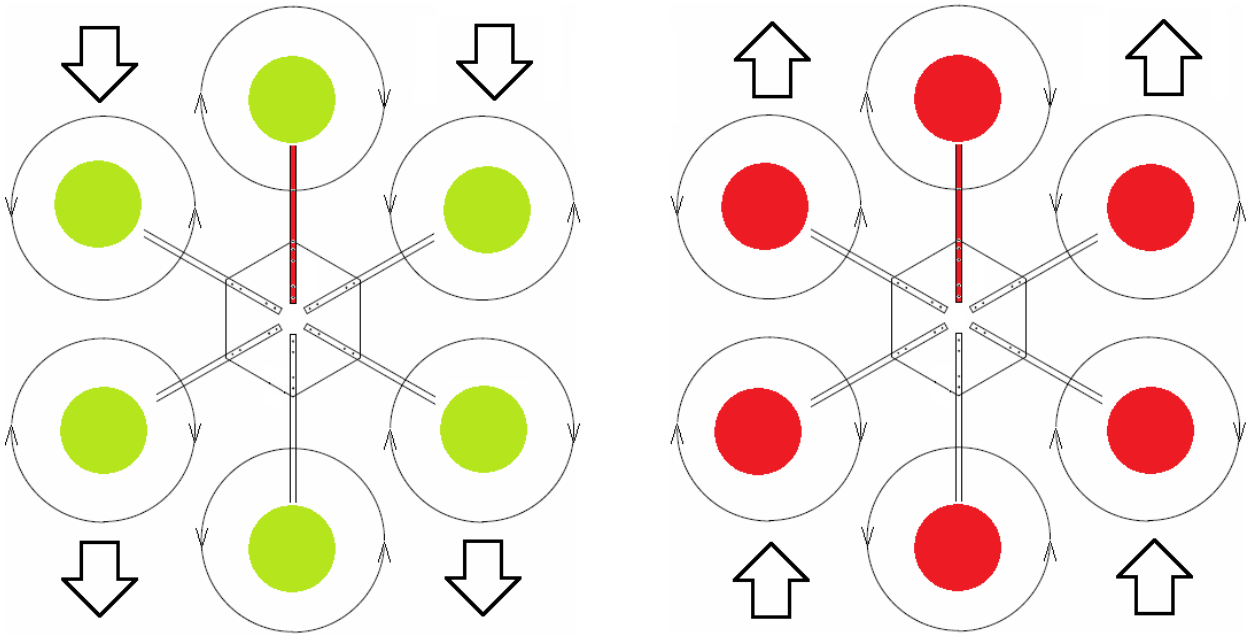

Figura II-34 Mişcarea de urcare şi coborâre cu toate elicele

Pe lângă mișcările de bază, mai sunt mișcările combinate și mișcarea redundantă. Mișcarea redundantă poate fi executată la urcare/coborâre. Această mișcare este prezentată în Figura II-35 Mișcarea redundantă, și poartă acest nume deoarece pentru a executa o urcare spre exemplu, creștem viteza de rotație a 4 elice reciproce, și aparatul va urca. Nu avem nevoie să scădem viteza pe celelalte 2 elice rămase, deoarece singurul lucru care se va întâmpla va fi ca aparatul să urce mai încet, forța de propulsie rezultantă fiind mai mică din

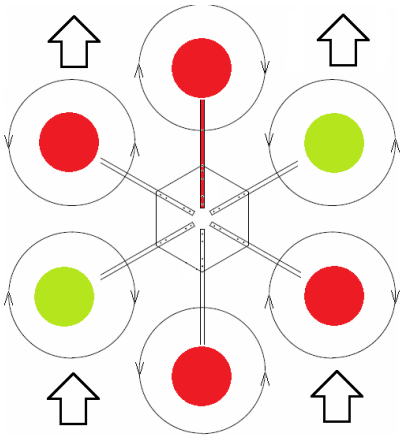

Figura II-35 Mișcarea redundantă

cauza reducerii acesteia pe cele 2 elice rămase. De asemenea, va opune rezistență inutilă și uzură fuzelajului aparatului.

Mișcările combinate sunt cerințe necesare unui aparat bun. Aceste mișcări sunt combinații de rostogoliri cu giratii, rostogoliri cu înclinații, și toate acestea combinate cu urcări și coborâri.

Cea mai utilizată mișcare a aparatului face parte din categoria mișcărilor combinate: mersul înainte. Aceasta este o combinație între mișcarea de înclinație (pitch) și urcare.

Pe motoarele din spate avem o turație mai mare, care va înclina aparatul, reducându-i forța de propulsie în jos. Pe motoarele din față trebuie să readucem forța de propulsie în jos la un nivel considerabil, așa că vom aplica o turație mai mare și pe acestea, dar mai mică decât pe cele din spate, pentru a menține înclinație. Astfel forța rezultantă nu va avea componentă pe axa Z, deci aparatul nu va urca sau coborî.

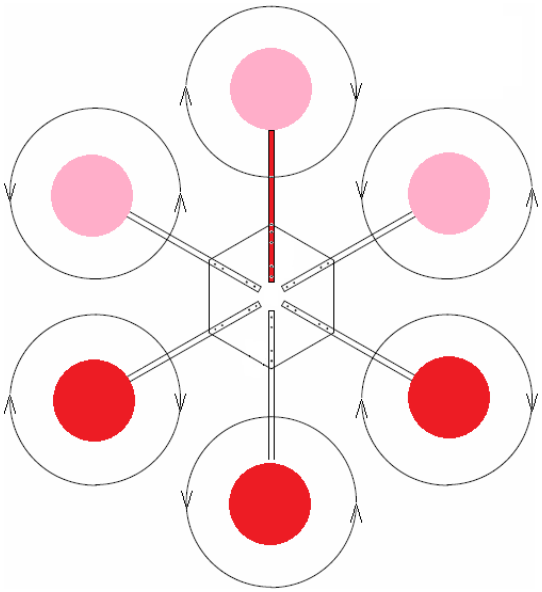

Figura II-36 Mişcarea complexă înainte

# 4. Proiectarea algoritmului pentru realizarea stabilității și controlului

Pentru următoarele pagini vom considera spațiul cartezian $O_{XYZ}$ ca fiind următorul: axa Z paralelă cu accelerația gravitațională și de sens opus. Hexacopterul în momentul în care stă în aer (fără a se mișca într-o anumită direcție) va fi considerat perpendicular pe axa Z.

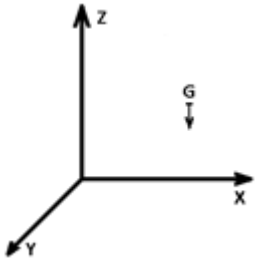

Figura II-37 Spaţiul cartezian

Axele X și Y sunt folosite doar pentru a avea o orientare relativă, iar planul $O_{XY}$ îl vom considera ca fiind planul hexacopterului, punctul O fiind considerat centrul hexagonului ce definește hexacopterul. Sub alte aspecte, în punctul O este montat senzorul de orientare.

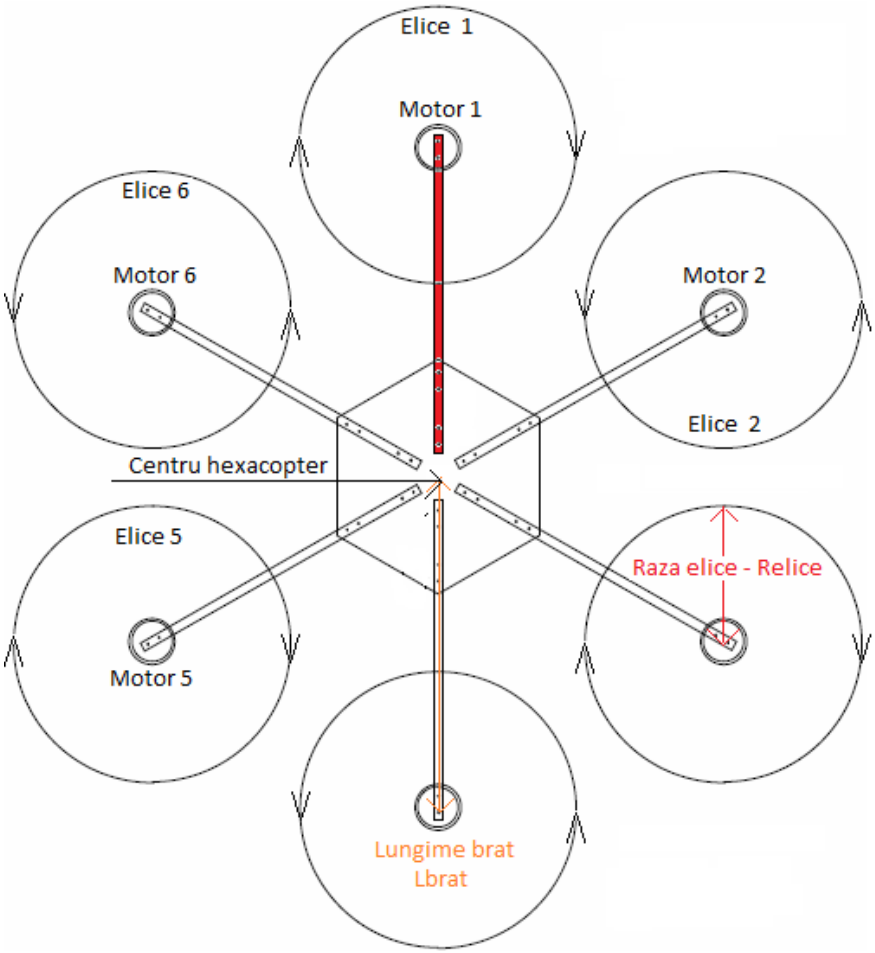

Figura II-38 Proprietăţi fizice ale hexacopterului

În Figura II-38 sunt precizate variabilele fizice prezente la hexacopter:

- Motoarele și elicele cu numerele lor corespunzătoare
- Centrul fizic O al hexacopterului
- Raza elicei: $R_{elice}$
- Lungimea brațului de la centru la motor: $L_{braț}$

Pentru exemplificare și simplificarea calculelor, motorul 1 va fi orientat pe axa Y al planului $O_{XY}$. Dacă ne referim la spațiul $O_{XYZ}$, axa Z va fi întotdeauna perpendiculară pe hexacopter. În Figura II-39 Hexacopterul în planul cartezian, axa Z se presupune că iese din pagină iar hexacopterul este privit de sus.

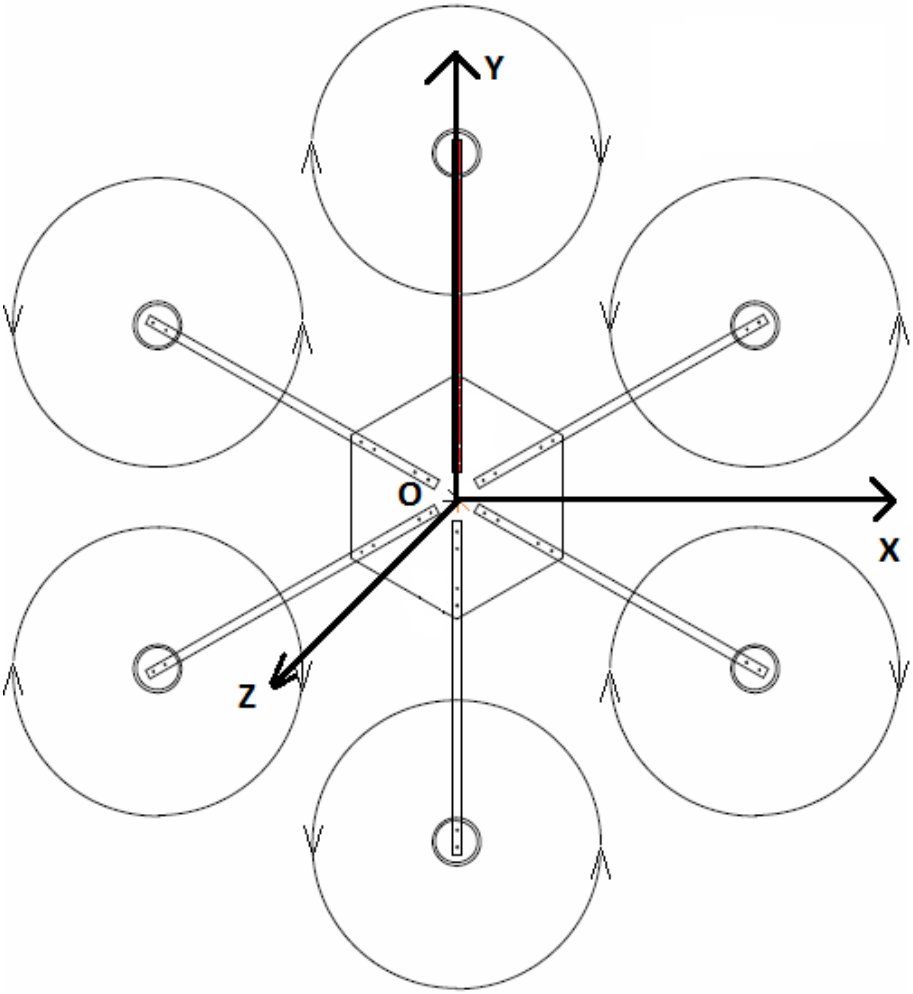


Figura II-39 Hexacopterul în planul cartezian

Având aceste detalii bine stabilite, ne putem gândi la o metodă eficientă de a controla mișcarea pe cele 3 dimensiuni ale hexacopterului. Pornim de la ideea că hexacopterul trebuie să fie un sistem autonom capabil să-și coordoneze mișcările online (online = „la bord”, realizat cu ajutorul capabilităților de procesare ale microcontrolerului) în funcție de senzorul de orientare și de inputul din partea utilizatorului.

Acest input poate fi catalogat în două clase:

- Neautonom
- Autonom

Inputul neautonom se bazează pe controlul hexacopterului strict din partea utilizatorului. Utilizatorul decide cât de sus trebuie să meargă aerodina, direcția în care trebuie să zboare și traiectoria acesteia. Acest input se bazează pe telecomanda acționată direct de utilizator și informațiile de pe telecomandă transmise (digital sau analog) către hexacopter, care după o prelucrare simplă va executa mișcările dictate.

Pe de altă parte, inputul autonom se bazează pe faptul că inputul manual din partea utilizatorului (practic ieșirea telecomenzii) poate fi scriptată. Un bun exemplu de input autonom ar fi ca utilizatorul să-i încarce un traseu de GPS și hexacopterul să îl parcurgă singur, fără alte intervenții.

Fie că ne referim la input autonom, fie că ne referim la input neautonom, hexacopterul trebuie să execute anumite funcții total autonom: stabilitatea și mișcarea.

Prin stabilitate în primul rând ne referim la momentul când hexacopterul se ridică de la sol și nu primește comenzi de mișcare: pur și simplu sta în aer, la punct fix. Stabilitatea este influențată de calitatea senzorului de orientare, și implicit de filtrele pe care acesta le are implementate. O bună stabilitate este atunci când hexacopterul corectează mișcarea în timp util, fără a fi observabil fenomenul. În mișcare aerodina de asemenea trebuie să aibă stabilitate, în sensul că trebuie să-și ruleze traiectoria fără a avea corecții vizibile ochiului.

Dacă ne gândim la microcontroler și la capabilitatea acestuia redusă de procesare (în comparație chiar cu un procesor de uz general slab), putem trage concluzia că stabilitatea poate fi realizată eficient folosind controllere PI sau PID, câte unul pentru fiecare variabilă care trebuie controlată, și fiecare lucrând pe o anumită componentă a rezultantei de mișcare. Controllerele de PI/PID sunt ușor de realizat pe microcontrolere și au o eficiență sporită în momentul în care coeficienții acestora sunt bine calibrați.

Deci pană să ajungem la partea la care ne gândim efectiv la un algoritm de control și la care controlăm rezultanta cu ajutorul buclelor PID, trebuie să gândim hexacopterul din punct de vedere fizic.

Din punct de vedere fizic, hexacopterul poate fi privit ca o cutie neagră la care forțele primare (presupunând că nu avem vânt sau forțe acționate de om) sunt gravitația și forța rezultantă compusă din forțele de împingere ale celor 6 combinații motor-elice.

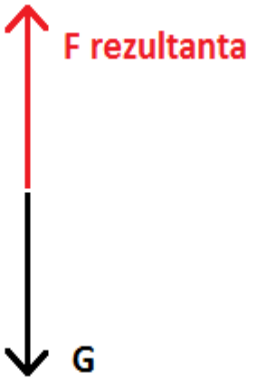


Figura II-40 Forţe pe verticală

Gravitația este cunoscută, știm că acționează pe axa Z. Pentru a combate gravitația, trebuie să răspundem cu o forță egală, de sens opus. Pentru a urca, trebuie să răspundem cu o forță mai mare pentru scurt timp, după care să revenim din nou în echilibru, iar pentru a coborî trebuie ca rezultanta să fie mai mică decât forța gravitațională pentru un interval scurt de timp.

Deci teoretic, pentru a ține hexacopterul în aer la punct fix fără ca alte forțe externe (provocate) să acționeze asupra lui, tot ce trebuie să facem este ca din cele 6 forțe ale motoarelor să compunem o forță rezultantă egală și de sens opus cu greutatea aparatului. Deoarece nu știm și nu putem măsura în zbor masa aparatului, ne vom baza pe accelerație, având posibilitatea folosirii accelerometrului din senzorul de orientare.

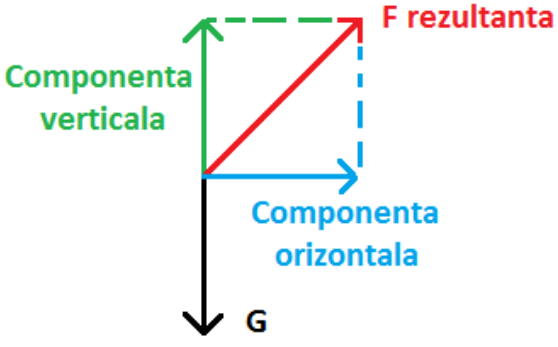


Figura II-41 Forţe în plan

Pentru a putea aplica o direcție de zbor

hexacopterului trebuie să aplicăm o forță rezultantă ca în figura alăturată. Această forță trebuie să aibă o componentă verticală de sens opus forței de greutate și o componentă orizontală care va mișca hexacopterul într-o direcție. Componenta orizontală reprezintă de fapt forța în planul $O_{XY}$.

În afară de forță, mai avem o altă variabilă de care trebuie să ținem cont: momentul rezultant al hexacopterului. Momentul este un factor important, deoarece reprezintă rotația aerodinei. Fiecare motor are un moment propriu, dat de sensul de rotație, viteza de rotație și de proprietățile elicei. Deoarece elicele sunt cvasiidentice (diferă sensul de rotație pentru a împinge), la o aceeași turație fiecare complex motor-elice va avea același cuplu, dar de semn contrar (fiecare motor se învârte în sens opus față de vecinii săi).

Momentul total reprezintă suma momentelor celor 6 motoare. Originea de acționare a vectorului moment folosit în această abordare va rămâne în centrul O al hexacopterului.

O formulă empirică ce poate da sensul de rotație rezultant al întregului aparat, reprezentat de momentul total al acestuia este:

$$M_{total} = \sum_{k=1}^{6} (-1)^k \times RPM_k \times const$$

Ecuația 1 Momentul total cinetic

Formula este empirică și nu reflectă în totalitate toate aspectele fizice, dar are avantajul de a fi ușor de procesat de microcontroler și ușor de controlat rotația hexacopterului. Dacă $M_{total}$ este 0 atunci hexacopterul nu se rotește, iar dacă este diferit de 0 hexacopterul se va roti într-o direcție dependentă de semnul lui $M_{total}$: negativ - aparatul se va roti în sensul acelor de ceas, iar dacă este pozitiv aparatul se rotește in sens trigonometric.

Revenind la forța rezultantă ce acționează asupra hexacopterului, putem scrie o formă simplificată a acesteia. În loc să considerăm o forță ce acționează pe o anumită direcție cu originea în O, putem considera o forță perpendiculară pe planul $O_{XY}$ al hexacopterului, cu originea în punctul O'(x,y), cu x și y relative la O. Această forță raportată la planul perpendicular pe axa gravitației va avea tot două componente: componenta verticală, care trebuie să anuleze gravitația, și o componentă orizontală care va mișca aparatul într-o anumită direcție. Direcția noii forțe va fi dată de punctul O' și centrul de greutate al aparatului.

Noua problemă se rezumă la a calcula punctul O'.

Avantajul noii tehnici de calcul reprezintă simplitatea de calcul din punct de vedere al algoritmului. Punctul O' are două componente x și y, ce pot fi ușor modelate cu ajutorul a două controllere PI/PID.

Notând cu $A_1$ . . . $A_6$ puterile relative exercitate de motoare și elice, putem dezvolta următoarele afirmații:

- $A_1$ . . . $A_6$ corespund variabilelor $BLDC_1$ . . . $BLDC_6$ din softul C
- $A_1$ . . . $A_6$ corespund puterii relative date de controllerele de motoare brushless și forței cu care motoarele și elicele împing în direcția perpendiculară pe planul hexacopterului
- Numerele sunt pe 16 biți

Singura problemă rămâne calcularea punctului O’.

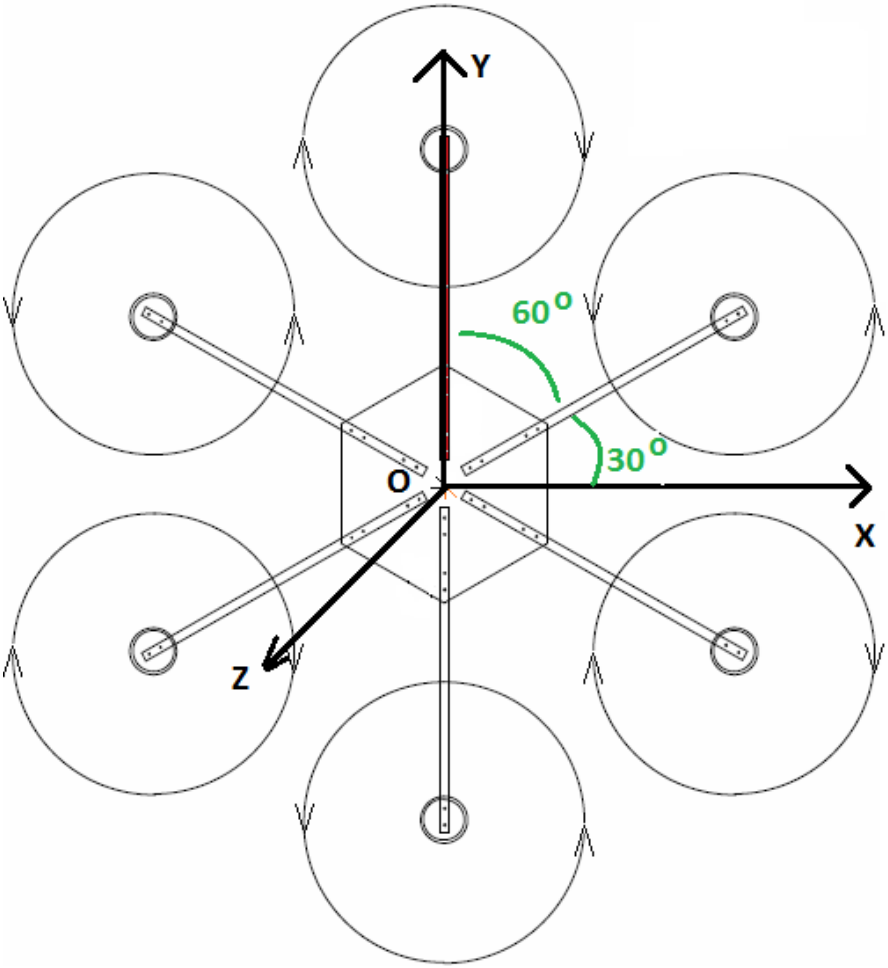


Figura II-42 Metoda de calcul al ponderilor

Deoarece toate brațele și elicele sunt analog egale unul cu celalalt și constante, în formula finală acestea pot fi omise.

$$O'_x = \frac{\sqrt{3}}{2} \times (A_2 + A_3 - A_5 - A_6)$$

$$O'_y = \frac{1}{2} \times (2A_1 + A_2 + A_6 - 2A_4 - A_5 - A_3)$$

$$O'_z = 0$$

Ecuaţia 2 Ponderea fiecărui motor la mişcare

# Capitolul III Implementare, modificări și rezultate ale proiectării

## *1. Controllerele de motoare brushless*

Layerul de deasupra reprezintă partea de putere, mai exact cele 6 tranzistoare MOS-FET – 3 cu canal P și 3 cu canal N, și driverele lor, precum și mufele necesare controllerului:

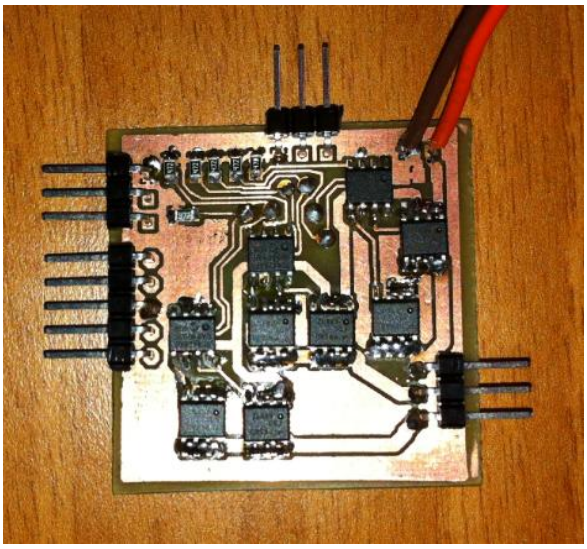

Figura III-1 Layoutul TOP al controllerului de motoare

Layerul de dedesubt reprezintă microcontrolerul, alimentarea acestuia (LDO), cristalul de quartz și divizorul rezistiv împreună cu rezistența limitatoare de curent de la intrarea convertoarelor analog-numerice ale microcontrolerului:

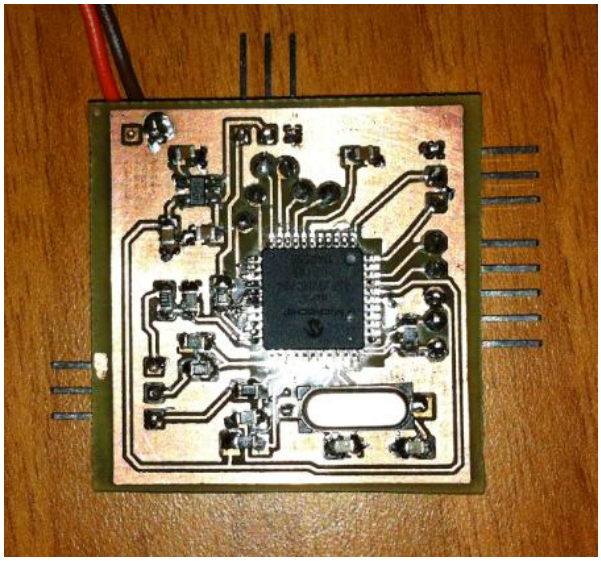

Figura III-2 Layoutul BOTTOM al controllerului de motoare

În Figura III-3 (printscreen de la osciloscop) se pot vizualiza semnalele celor trei faze ale motorului:

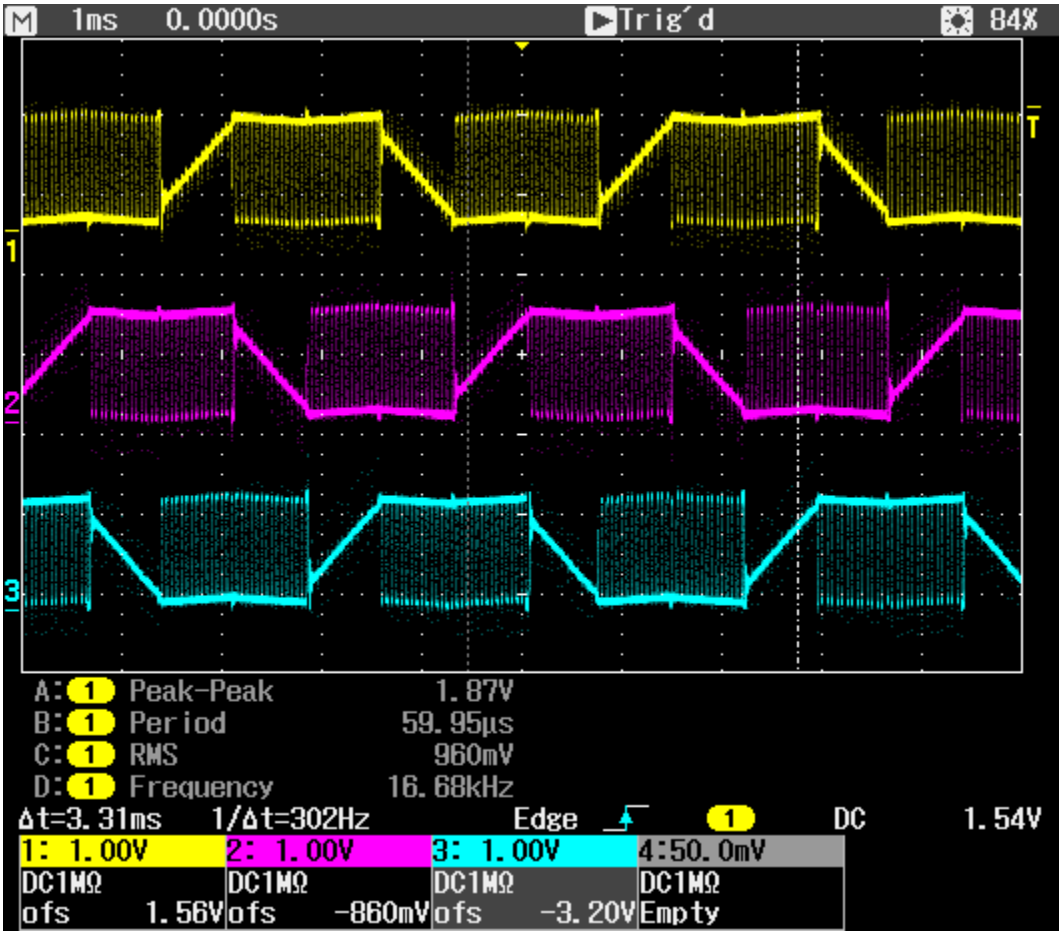


Figura III-3 Tensiunile de BEMF pe cele trei faze

Se poate vedea că factorul de umplere aplicat motorului nu este 100%, și că fiecare semnal este defazat cu 60 de grade față de cel precedent, fapt ce denotă o comutație corectă a sectoarelor. De asemenea se poate observa că forma traseelor respectă forma controlului trapezoidal.

La momentul testării cu motorul folosit pe hexacopter, controllerul s-a comportat foarte bine. Dar în momentul în care am folosit sarcină pe motor şi acesta a început să consume mai mult curent, am observat că tot ansamblul se încălzeşte excesiv la peste 2A consumaţi, lucru inadmisibil din punct de vedere al proiectului – la 2A pe motor aparatul abia dacă ar fi fost capabil să se ridice de la sol fără extra-greutate.

După măsurători şi reverificări am ajuns la concluzia că tranzistorii de tip P se deschid mult mai greu decât estimasem iniţial. Tranzistorii N se deschid în 15 ns, iar tranzistorii P aveau nevoie de aproximativ 250 ns pentru aceasta. Acest lucru se întâmplă din cauza tehnologiei existente, ce oferă avantaje canalului N. Tranzistoarele cu canal P sunt în general mai lente, consumă mai mult, şi sunt mai puţin folosite. În cazul de faţă, singurele soluţii ar fi fost să schimb tranzistoarele P cu altele care au aceeaşi capsulă sau să reproiectez controllerul de motoare. Deoarece nu am găsit tranzistoare MOS-P cu capsulă identică la un preţ acceptabil, şi deoarece reproiectarea controllerului ar fi durat câteva săptămâni bune cu tot cu trimis la fabrică cele 6 plăci, am luat

decizia să folosesc controllere de motoare brushless din comerţ. Controllerele pe care le-am ales se numesc “ESC for brushless motors with BEC”, şi în traducere liberă înseamnă controllere de viteză pentru motoare fără perii cu ieşire de alimentare externă.

Acesta face aproape tot ce făcea şi controllerul proiectat de mine, cu excepţia că nu măsoară turaţia şi nu are o buclă de control al turaţiei, ci pur şi simplu încearcă, după o referinţă să stabilească o tensiune de control al motorului şi să execute comutaţiile de sector corect. Eficienţa acestui controller este destul de ridicată, ceea ce-l face o alegere bună.

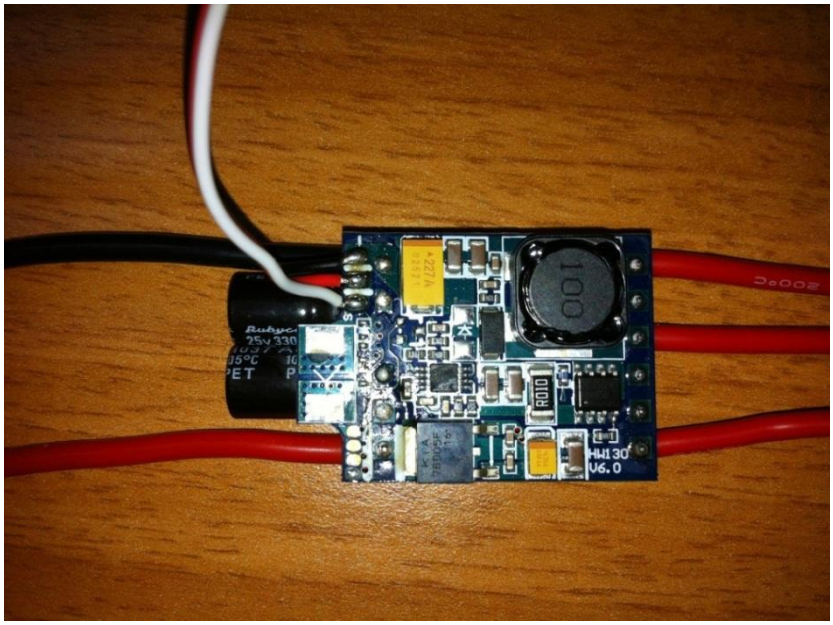

Figura III-4 Noul controller de motoare – Layer-ul pe care se vede sursa în comutaţie destinată circuitului BEC

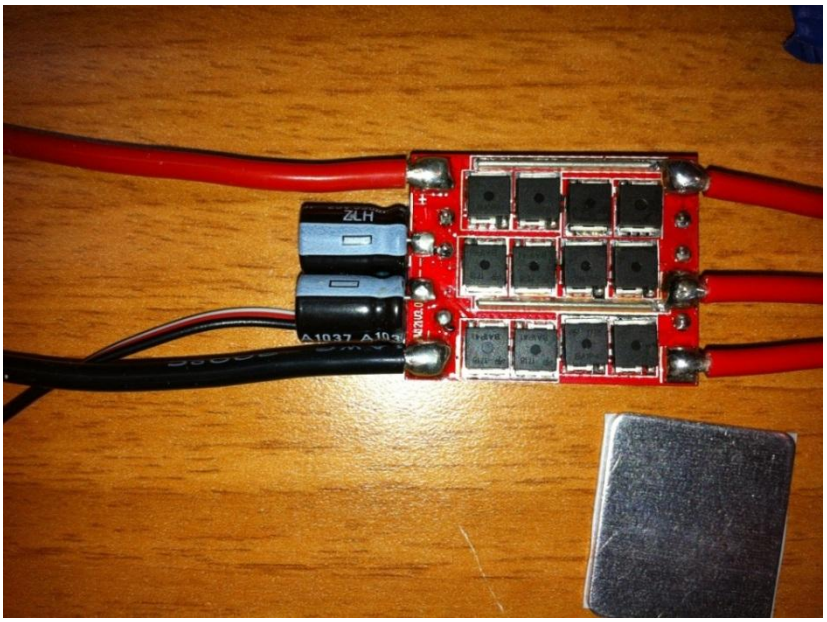

Figura III-5 Noul controller de motoare - Layer pe care se află tranzistoarele de putere, puse câte 2 în paralel

Noul controller de motoare acceptă motoare de până la 20A fără a se încălzi semnificativ, iar comanda acestuia se face prin semnal PWM cu o frecvenţă de 50 Hz şi factor de umplere cuprins între 1 şi 2 milisecunde. La 1 milisecundă motorul stă pe loc, iar la 2 milisecunde acesta are viteza maximă pe care o poate atinge la tensiunea furnizată.

# *2. Reproiectarea plăcii de bază*

Deoarece controllerul de motoare a fost schimbat şi placa de bază iniţială avea ca ieşiri pentru controlul motoarelor doar o magistrală $I^2C$ capabilă să comande toate cele 6 motoare, prima placă realizată nu mai poate fi folosită pe noile controllere deoarece nu are suficienţi pini pentru control scoşi, şi nu are modul ce poate genera semnal PWM din hardware, acesta putând fi realizat doar software. PWM-ul software nu ar fi fost o problemă (deşi cel hardware este mult mai precis, are o rezoluţie mult mai mare şi nu consumă cicli de procesare), dar ar fi trebuit scoase semnale din microcontrolerul PIC32® până la pini, iar aceste semnale trase cu fir subţire. Acest lucru ar fi pus în pericol tot ansamblul din cauza vibraţiilor care ar fi putut desprinde oricând firele lipite. Aşa că am decis să proiectez o nouă placă de bază, lucru relativ uşor de realizat în comparaţie cu un controller de motoare, care are multe piese pe el şi care trebuia să iasă relativ mic ca dimensiuni.

Pentru noua placă de bază am decis să folosesc alt microcontroler – dsPIC33EP512MU810 [(11)], produs tot de către Microchip® - puţin diferit din punct de vedere hardware cu PIC32® prezentat iniţial, care oferă şi un modul PWM. Dezavantajul acestui nou microcontroler este că rulează la o frecvenţă puţin mai mică şi implicit are viteză de procesare mai mică decât microcontrolerul folosit iniţial - 70 MIPS în comparaţie cu 130-140 MIPS – dar acest lucru nu împiedică realizarea funcţionalităţii deoarece pentru o astfel de aplicaţie nu este nevoie de mai mult de 40 MIPS. Prima placă, cea cu PIC32®, am denumit-o versiunea de test.

Celelalte două versiuni ale plăcii de bază sunt versiunile complete și mini. Versiunea completă prezintă mai multe periferice care nu țin de dinamica sau funcționarea hexacopterului în sine, ci de controlul diverselor module și periferice prezente pe acesta. Primul ar fi un controller video de suprascriere (OSD - OnScreenDisplay), care se atașează între ieșirea camerei video și intrarea transmițătorului, și care suprascrie semnalul PAL al camerei cu anumite informații privind tensiunile, curenții, locația și viteza hexacopterului.

Un alt modul ce va fi disponibil pe versiunea completă a plăcii este cel de control al camerelor de fotografiat profesionale (mai exact, timp de expunere și fotografiere) prin mufa standard prezentă pe aceasta, precum și doi conectori pentru servomotoarele ce vor deservi mișcările de înclinație ale camerei foto profesionale. De asemenea, placa completă dispune și de un modul „beacon” de găsire al hexacopterului, precum și de posibilitatea de atașare a unui receiver GPS. Mai exact, acesta va putea traduce informația de la receiverul de GPS și o va putea trimite pe canalul audio al transmițătorului AV.

Versiunea completă a plăcii a fost proiectată dar nu a fost încă executată, deoarece este mult mai complexă decât versiunea mini și necesită mai multe verificări. Versiunea mini este are toate componentele necesare pentru a demonstra capabilitățile hexacopterului. Ambele versiuni (completă și mini) au fost proiectate în Altium Designer 10, un program de proiectare asistată de calculator mai performant decât OrCAD Layout.

Mini este versiunea curentă cu care hexacopterul poate executa toate funcțiile de zbor. Pe această placă putem găsi următoarele componente: mufele de conectare ale perifericelor, microcontrolerul, cristalul extern de quartz și un regulator de tensiune ce asigură tensiunea de 3.3V necesară microcontrolerului, senzorului de orientare și modulului de transmisiune ZigBee.

Layoutul acestei plăci, precum și versiunea 3D a acesteia generată de Altium sunt prezentate în Figura III-6 Layout-ul plăcii de bază în versiunea mini şi Figura III-7 Placa de bază în versiunea mini – model 3D generat de Altium Designer®. Versiunea reală (executată) de cablaj poate fi văzuta în pozele ce prezintă întreg hexacopterul.

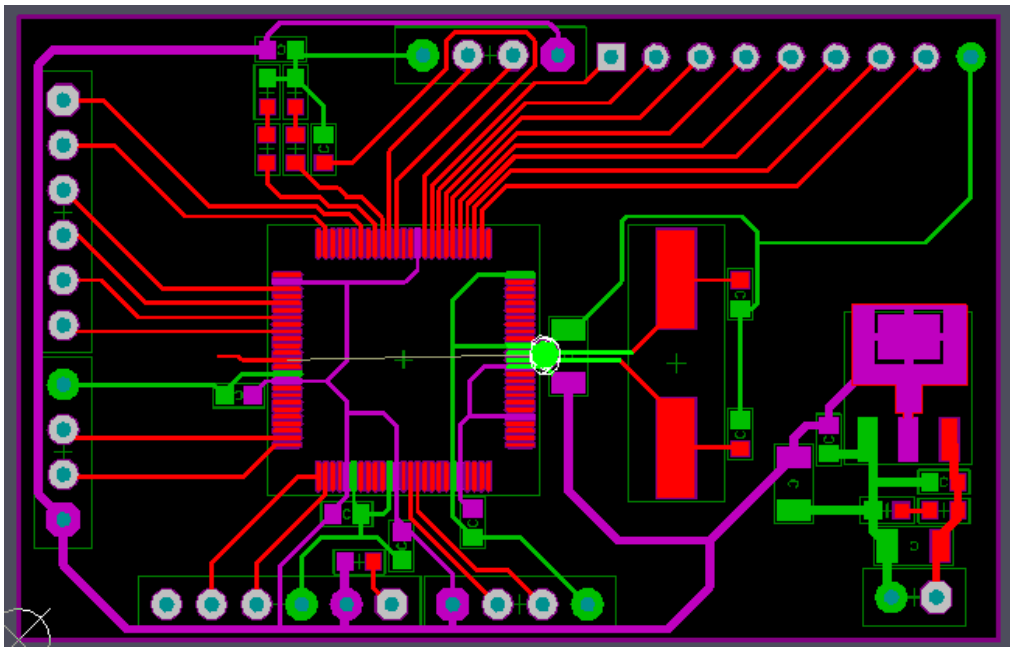

Figura III-6 Layout-ul plăcii de bază în versiunea mini

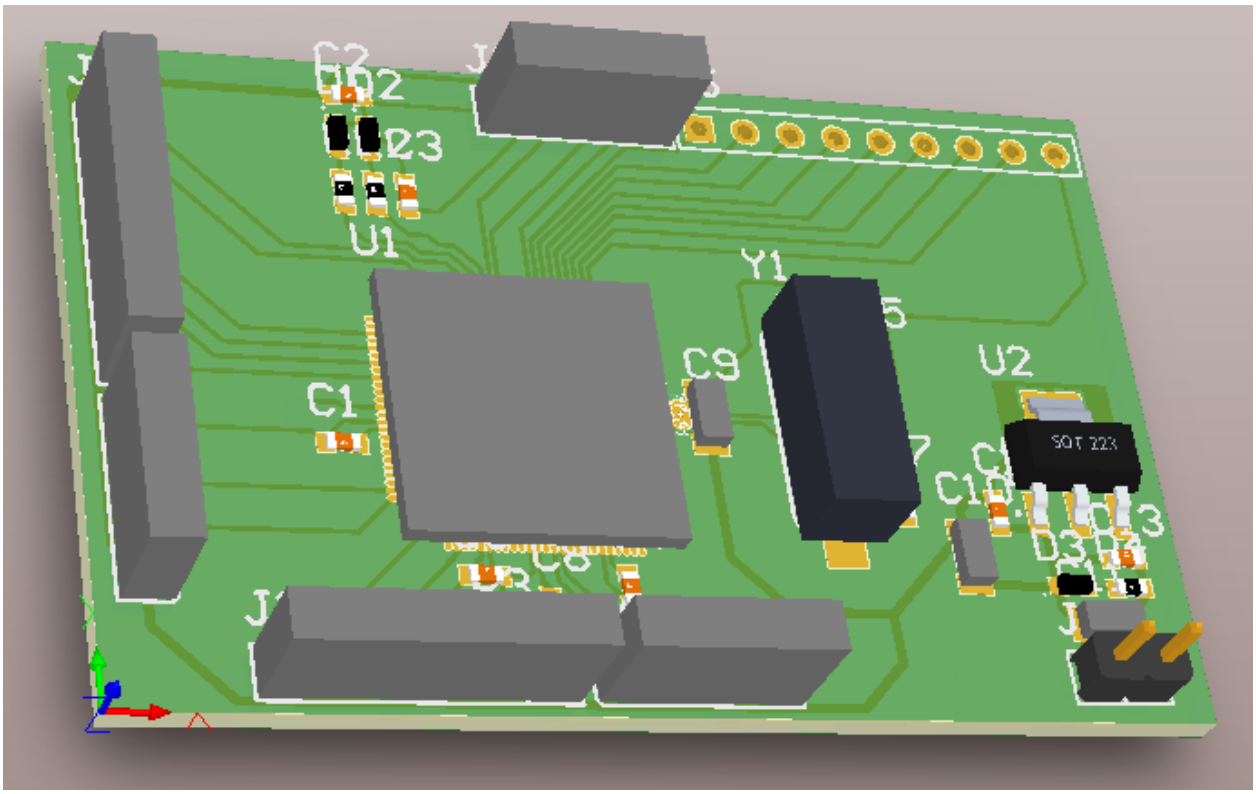


Figura III-7 Placa de bază în versiunea mini – model 3D generat de Altium Designer®

# 3. Folosirea unei telecomenzi

Planul iniţial a fost să folosesc transceiverul ZigBee pentru a controla hexacopterul şi pentru a obţine informaţiile necesare pentru test şi depanare de la acesta. În timpul testelor am observat că este foarte greu să controlezi aparatul folosind nişte numere pe care le trimiţi (în principiu scrise de mână), şi în caz de pericol este bine să ai un control cât mai bun asupra aparatului.

Soluţia a fost folosirea unei telecomenzi clasice de avioane şi elicoptere din comerţ, telecomandă cu un modul radio ce funcţionează 100% analogic şi care, în caz de desincronizare sau pierdere pentru câteva momente a semnalului, nu necesită un timp îndelungat pentru a realiza din nou comunicaţia între transmiţător şi receptor, precum modulele ZigBee.


Figura III-8 Telecomanda folosită pentru control [4]

Telecomanda şi cu receptorul aferent acesteia au 8 canale de transmisie programabile, iar semnalul transmis este un semnal PWM similar celui folosit de noile controllerele de motoare fără perii. Telecomanda are două joystickuri, două potenţiometre şi o mulţime de butoane adiţionale pentru setări. De asemenea aceasta are şi un procesor de semnal care are mai multe moduri de funcţionare definite pentru controlul diverselor aparate (avioane, elicoptere, bărci, etc.), procesorul putând mixa semnalele în funcţie de setarea dorită. Un exemplu de mixare a semnalelor este la un elicopter clasic, atunci când se mişcă de joystickul care deplasează elicopterul înainte, să crească automat şi turaţia rotorul principal pentru a nu pierde din altitudine.

În proiect am folosit telecomanda pe modul brut în care mixarea este dezactivată, şi pe ieşirile receptorului găsesc cele 4 semnale aferente celor 2 joystickuri, plus încă un semnal aferent unui potenţiometru, pentru teste.


Figura III-9 Receptorul aferent telecomenzii [4]

După cum am precizat, receptorul scoate 8 canale din care doar primele 5 sunt folosite, iar ieşirile acestora sunt semnale PWM cu frecvenţa de 50 Hz şi factor de umplere între 1 şi 2 milisecunde. Configuraţia canalelor este următoarea:

- Axa X a joystickului din stânga – canalul 4 (mişcarea de giraţie – yaw)
- Axa Y a joystickului din stânga – canalul 2 (acceleraţia – throttle)
- Axa X a joystickului din dreapta – canalul 1 (mişcarea de rotaţie – roll)
- Axa Y a joystickului din dreapta – canalul 3 (mişcarea de înclinaţie – pitch)
- Potenţiometrul – canalul 5 (folosit la testare)

Noua placă de bază a fost prevăzută cu intrări digitale ce pot măsura astfel de semnale, numite module „Input Compare”. Aceste module folosesc fiecare câte un numărător dedicat şi un detector de salt de tensiune (fie din 0 în 1 – rising, fie din 1 în 0 – falling). Din soft, în momentul în care se detectează un salt din 0 în 1, se porneşte numărătorul, iar când se detectează un salt din 1 în 0 se opreşte numărătorul, se memorează rezultatul şi se resetează numărătorul pentru a putea fi pregătit pentru următoarea numărare. Cunoscând frecvenţa de numărare şi valoarea numărării putem calcula exact factorul de umplere pe semnal. Modulul a fost configurat pentru a scoate numere între 1050 şi 2100 pentru 1 respectiv 2 milisecunde. În Anexa 1 – Cod C destinat telecomenzii, paragraful 1 şi 2 este prezentat codul C aferent iniţializării modulului şi citirii unuia dintre canalele receptorului.

În Figura III-10 şi în Figura III-11 se poate observa efectul mişcării axei Y a joystickului din partea stângă. La minim avem un factor de umplere de 1.05ms iar la maxim de 1.88ms.Toate celelalte semnale sunt centrate (aproximativ 1.45ms) deoarece joystickul prin construcţie stă centrat.

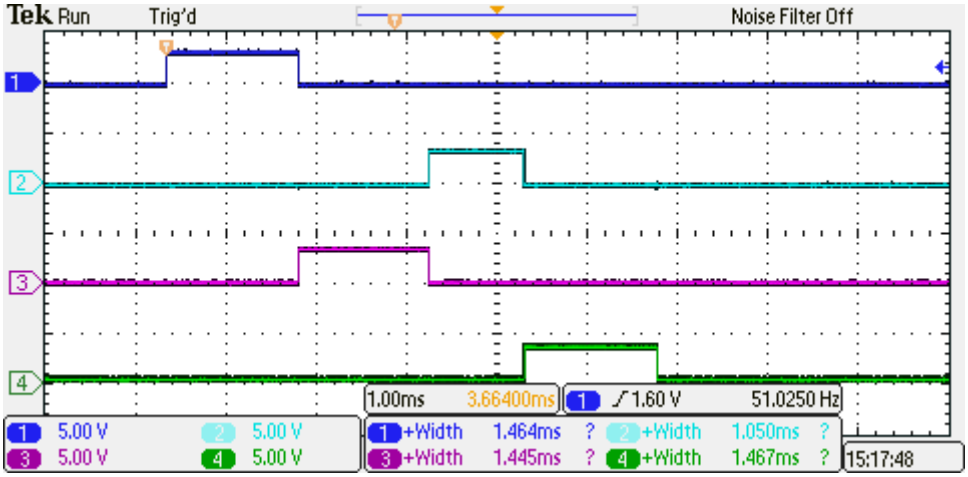


Figura III-10 Semnale de la telecomandă: acceleraţia minimă (2) iar celelalte semnale centrate

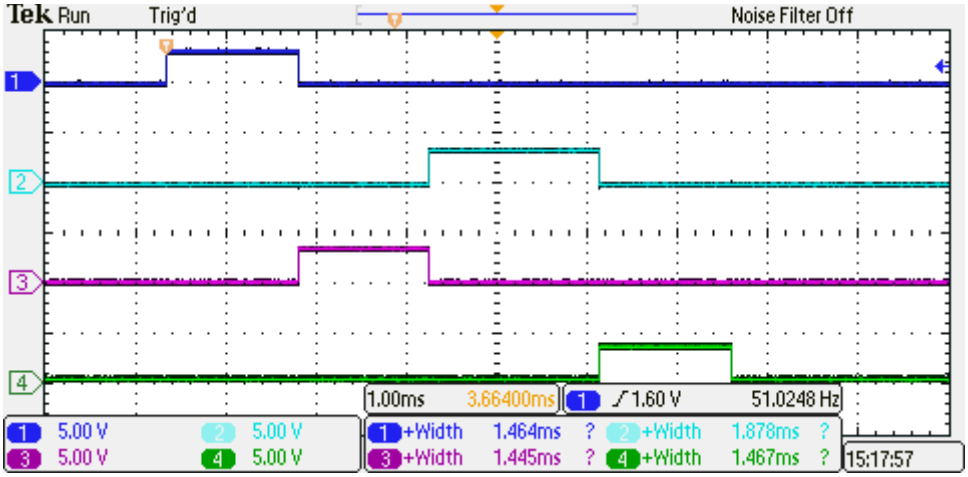


Figura III-11 Semnale de la telecomandă: acceleraţia maximă (2) iar celelalte semnale centrate

Modulul ZigBee folosit iniţial a fost păstrat şi folosit ca utilitar pentru depanare şi colectare a datelor de la bordul aparatului.

# *4. Sisteme şi măsuri de siguranţă "la bord"*

Din teste şi experienţă am realizat că, până a trece la dezvoltarea algoritmului propriu-zis de control şi până a trece la montarea elicelor, o serie de măsuri de siguranţă trebuiesc implementate. Aceste măsuri sunt implementate software şi asigură un minim de siguranţă deopotrivă aparatului şi utilizatorului.

O primă măsura de siguranţă implementată a fost detecţia activă a telecomenzii. Această măsură se bazează pe faptul observat că atunci când telecomanda este oprită, canalul 2 al receptorului este oprit şi nu avem niciun semnal la ieşire. Sistemul este implementat să detecteze lipsa semnalului pe un interval de 50 ms, moment în care marchează telecomanda ca fiind oprită. În momentul în care telecomanda este oprită, computerul trimite permanent un semnal de oprire al motoarelor, indiferent de starea lor anterioară. De asemenea, acest sistem include şi calibrarea automată a telecomenzii în momentul pornirii: se citesc semnalele aferente poziţiei de mijloc al ambelor joystickuri, cu excepţia axei Y a joystickului din stânga, care reprezintă acceleraţia, şi care este scrisă în soft pentru siguranţă.

A doua măsură a fost limitarea acceleraţiei. Din testele în care se computerul plăcii de bază trimitea la toate motoarele aceeaşi putere comandată de cursa axei de acceleraţie s-a observat că motoarele încep să se învârtă la aproximativ un sfert din această cursă, iar undeva sub jumătate de cursă aparatul începe să se ridice. Aşa că outputul motoarelor a fost limitat la 75% din puterea lor totală peste toată cursa axei de acceleraţie.

Figura III-12 Resturi de elice

O ultimă, şi foarte importantă măsură de siguranţă a fost implementată ulterior, după un accident - Figura III-12. Telecomanda are un sistem de verificare a butoanelor la pornire, care nu permite funcţionarea acesteia dacă unul din butoane se află iniţial intr-o altă poziţie decât cea standard, şi din câte se pare, dacă această eroare survine, pe semnalele de ieşire ale receptorului avem factor de umplere de aproximativ 100%, factor pe care microcontrolerul de la bord îl citeşte pe acceleraţie ca însemnând viteză maximă, şi aparatul începe să plece accelerat în sus necontrolabil. Acest lucru a cauzat ridicarea aparatului foarte repede, apoi acesta s-a lovit de picioarele mesei sub care era ţinut din motive de protecţie, tocând practic elicele. Pentru a evita aceste tipuri de accidente a fost introdus un sistem de armare şi dezarmare a aparatului din

telecomandă. Când aparatul este dezarmat, LED1 (albastru) de pe placa de bază este stins, iar când aparatul este armat LED1 este aprins. Armarea se face din potenţiometrul auxiliar aflat pe canalul 5 şi axa de acceleraţie a joystickului din stânga: potenţiometrul trebuie mişcat întâi la maxim, după care la minim, si la sfârşitul acestei mişcări acceleraţia trebuie să fie coborâtă la minim. Dacă la pornirea telecomenzii acceleraţia nu este la minim, iar potenţiometrul este la maxim, simplul fapt că utilizatorul trebuie să mişte potenţiometrul la minim pentru armare îl atenţionează să verifice acceleraţia. Pentru dezarmare este suficient ca potenţiometrul să fie mişcat din poziţia de minim şi aparatul va fi dezarmat. De asemenea, pentru orice oprire a telecomenzii, aparatul se va auto-dezarma. Dezarmarea presupune trimiterea permanentă a semnalelor de stop către motoare.

Codul în limbaj C al acestor sisteme de siguranţă privind telecomanda şi controlul primar (fără automatizare cu feedback din partea senzorului de orientare) al hexacopterului se găseşte în Anexa 1 – Cod C destinat telecomenzii.

O ultimă precizare privind siguranţa este făcută pe baza libertăţii in mişcare pe care aparatul o deţine. Pentru că proiectul este la început şi pentru că nu s-a dispus de un banc specializat de testare, orice fel de mişcare acrobatică (viteză mare de înaintare, dat peste cap, etc.) a fost oprită prin limitarea unghiurilor maxime de rostogolire (roll) şi de înclinaţie (pitch) la 30 de grade.

# 5. Algoritmul de control

Greutatea proiectului constă în proiectarea unui algoritm optim de control capabil să ruleze pe procesoare de viteză mică ce poate menţine stabilitatea şi impune o direcţie de mers întregului aparat. Conform formulelor din Ecuaţia 2, fiecare motor are câte o pondere la direcţia, la sensul şi la forţa de deplasare. De asemenea, din Ecuaţia 1 reiese că fiecare motor are o anumită pondere şi la momentul total cinetic al hexacopterului.

Algoritmul de control este realizat separat pe toate cele 3 axe, şi constă într-un set de trei controllere de tip proporţional-integral (PI) rulează independent unul de celălalt, fiecare încercând să corecteze axa pe care o controlează.

$$Eroarea\ Propoţională = \sphericalangle(dorit) - \sphericalangle(curent)$$
$$Eroarea\ Integrală = Eroarea\ Integrală + Eroarea\ Proporţională$$
$$Ieşirea\ PI = kP \times (Eroarea\ Proporţională) + kI \times (Eroarea\ Integrală)$$

Ecuaţia 3 Controllerul proporţional-integral (PI)

În cazul fiecărei axe, unghiul dorit este setat din telecomandă şi menţinut într-o variabilă. Acesta poate fi modificat şi în software, pentru eventualitatea implementării unui algoritm de zbor total autonom ce nu se foloseşte de telecomandă.

Unghiul curent este dat de către senzorul de orientare, este reprezentat în grade şi are o rezoluţie teoretică de 0.05 grade. Practic însă, senzorul este influenţat de vibraţii şi de curenţii trecuţi prin firele de alimentare al motoarelor. Acest lucru reduce rezoluţia la aproximativ 0.5 grade. Unghiul dat este deja filtrat de către procesorul ARM aflat pe acesta.

Ecuaţia 3 descrie controllerul PI folosit în lucrare. În lucrarea practică, controllerul este iniţializat. Iniţializarea constă în setarea unor parametri iniţiali cum ar fi kP şi kI, precum şi limitele ieşirii controllerului. Limitele şi parametrii sunt tunate pentru a obţine eficienţă maximă în control.

Ieşirea reprezintă corecţia controllerului asupra fiecărei axe. Fiecare motor are o anumită repercusiune raportată la toate axele. Aplicând cinematica inversă, cunoscând ieşirea controllerului PI, putem stabili viteza fiecărui motor ce urmează.

$$BLDC(N) = Acceleraţie + PIoutRoll \times COS(N) + PIoutPitch \times SIN(N) + PIoutYaw \times FactorCuplu(N)$$

Ecuaţia 4 Cinematica inversă a motoarelor

Factorii SIN(N) şi COS(N) sunt calculaţi conform schemei din Figura II-42, iar rezultatul se găseşte în Tabelul III-1 Ponderile axelor asupra motoarelor.

| Motor/Elice | SIN – axa Y - Pitch | COS – axa X - Roll | Factor de cuplu |
|---|---|---|---|
| 1 | 1 | 0 | 1 |
| 2 | 0.5 | 0.866025404 | -1 |
| 3 | -0.5 | 0.866025404 | 1 |
| 4 | -1 | 0 | -1 |
| 5 | -0.5 | -0.866025404 | 1 |
| 6 | 0.5 | -0.866025404 | -1 |

Tabelul III-1 Ponderile axelor asupra motoarelor

Codul C al acestui algoritm este prezentat în Anexa 2 - Cod C al algoritmului de stabilizare.

# Capitolul IV Concluziile lucrării

Lucrarea „Sistem multiprocesor de zbor tridimensional cu şase elice verticale” ce a fost tratată în aceste pagini este o lucrare complexă, cu referire în multe domenii reale: electronică, algoritmică, matematică, mecanică şi aeronautică. Aparatul – hexacopterul – rezultat în urma acestei lucrări prezintă din punct de vedere electronic o complexitate sporită, fiind utilizate elemente din circuitristica digitală, microprocesoare, semnale, telecomunicaţii, proiectare asistată de calculator, precum şi elemente solide de circuitristică de putere.

În această lucrare am realizat aparatul, urmărind paşii necesari: proiectare, construire, asamblare, şi apoi implementare. Proiectarea am realizat-o folosind programele OrCAD şi Altium. OrCAD Layout a fost de asemenea folosit şi pentru proiectarea mecanică a aerodinei, datorită capacităţii acestuia de a lucra cu mărimi fizice. Construirea şi asamblarea a fost realizată prima, pentru ca mai apoi să urmeze implementarea. În paralel am implementat un controller de motoare fără perii eficient, dar care a avut probleme la proiectarea invertorului, acesta neputând susţine curenţii necesari datorită shoot-through-ului cauzat de deschiderea foarte lentă a tranzistorului MOS-P în comparaţie cu tranzistorul MOS-N. Implementarea algoritmului principal de control al aerodinei a constat în mai multe părţi: comunicaţia cu controllerele şi implicit cu motoarele, comunicaţia cu telecomanda, comunicaţia cu senzorul de orientare, comunicaţia cu modulul ZigBee şi implicit cu calculatorul. Cea mai importantă parte a implementării o constă algoritmul de control şi stabilizare a hexacopterului, cele 3 controllere proporţional-integrale şi tunarea acestora.

Materiile ce stau la baza acestei lucrări includ Microcontrolere, materiile de măsurări şi programare, Tehnici CAD, Fizică, Robotică, Automatizări, Testarea echipamentelor şi proceselor, precum şi Dispozitive electronice şi Circuite electronice fundamentale.

Fizica a fost folosită pentru a calcula şi proiecta aparatul în sine, Dispozitivele şi Circuitele electronice pentru a proiecta controllerele de motoare fără perii şi placa de bază, Tehnicile CAD pentru a proiecta layout-ul acestor plăci, Testarea pentru a verifica fiecare placă şi subcircuit de pe aceasta că sunt proiectate şi executate corect, iar apoi, cu ajutorul programării, folosind elemente de robotică şi automatizări am construit întregul algoritm şi proces. Programele în sine au fost făcute cu elemente specifice microcontrolerelor.

O altă analiză a proiectului poate fi făcută din privinţa costurilor totale, pe componente şi sisteme. În Tabelul IV-1 sunt prezentate costurile de bază ale aparatului proiectat excluzând eventualele module ce nu au fost prezentate în lucrare (GPS, video), manoperă şi componentele făcute (placă de aluminiu, suport, prinderi elice) şi componentele oferite gratuit de către Microchip Technology SRL (microcontrolere, servicii de execuţie ale plăcilor electronice, aparate de testare, programare, etc).

| Item | Nume | Buc. | Cost Total | Valuta | Note |
|---|---|---|---|---|---|
| 1 | NX-4006-530kv BLDC Motor | 6 | 648 | RON | HK |
| 2 | Elice 1147 L+R | 12 | 52 | RON | HK |
| 3 | Fibră de carbon | 6 | 55 | RON | HK |
| 4 | Acumulator LiPo 14.8V 6Ah | 1 | 192 | RON | HK |

| | | | | | |
|---|---|---|---|---|---|
| 5 | Încărcător LiPo | 1 | 58 | RON | HK |
| 6 | Telecomanda 8 canale | 1 | 190 | RON | HK |
| 7 | Modul ZigBee NXP Jennic + Antenna | 2 | 220 | RON | Farnell |
| 8 | IMU CH Robotics UM6-LT | 1 | 525 | RON | Pololu |
| 9 | BLDC Controllers 40A | 6 | 420 | RON | HK |
| 10 | Restul pieselor, fire, suruburi, plăci fabrică, etc | 1 | 0 | RON | Fonduri proprii + Support Microchip |
| 11 | Transport total | 4 | 458 | RON | 4 comentzi plasate |
| | | | | | |
| | **TOTAL** | | 2818 | RON | |

**Tabelul IV-1 Costurile totale implicate**

În această lucrare contribuţia mea a constat în analizarea sistemelor asemănătoare, proiectarea unui sistem asemănător dar cu elemente proprii, sinteza în realizarea controlului şi a comenzii electronice, conceperea modelului experimental prin împletirea de elemente existente pe piaţă şi elemente proprii proiectate, precum şi conceperea şi realizarea totală a părţii software a proiectului. Pentru construcţia şi montajul elementelor fizice am consultat diverse persoane pricepute în domeniul mecanic, iar aparatul a ieşit foarte rigid, cu elemente de fibră de carbon şi aluminiu. Cam 40% din timp l-am petrecut proiectând controllerele de motoare fără perii, 20% în construcţia fizică a aparatului, iar restul de 40% în proiectarea plăcilor de bază şi în realizarea şi testarea softwareului.

Pentru viitor i-am prevăzut o nouă placă de bază, capabilă ca pe lângă controlul clasic din telecomandă, să realizeze şi un control autonom, bazat pe coordonate GPS. De asemenea, pe această nouă placă vor fi şi componente de preprocesare a unui semnal video PAL prin suprascriere cu informaţii utile, precum şi circuitristica necesară conectării unei camere foto sau video profesionale şi controlului acesteia. Noua placă va avea un senzor de orientare integrat în capsulă QFN, care este mult mai ieftin decât senzorul folosit în prezent, şi care elimină necesitatea folosirii a încă unei plăci adiţionale. De asemenea un circuit de măsurare a curentului instantaneu consumat şi a timpului de viaţă al bateriei va fi implementată pe această placă. O ultimă îmbunătăţire preconizată este reprezentată de posibilitatea achiziţionării de semnale de la alţi senzori (poluare, gaz, temperatură) printr-o interfaţă serială.

Din punct de vedere al algoritmului, se pot schimba controllerele PI cu PID sau alt tip de controlere, care pot reduce consumul de energie prin verificarea constantă a mişcărilor redundante. Algoritmul poate fi completat, folosind încă un procesor mult mai puternic pentru a adăuga algoritmi de analiză de imagini, pentru a urmări diverse obiective automat, sau pentru analiză si recunoaştere vocală ori acceptare de comenzi.

# Bibliografie

1. History of robots. *Wikipedia.* [Interactiv] http://en.wikipedia.org/wiki/History_of_robots.

2. History of artificial intelligence. *Wikipedia.* [Interactiv] http://en.wikipedia.org/wiki/History_of_artificial_intelligence.

3. Texas Instruments. *Wikipedia.* [Interactiv] http://en.wikipedia.org/wiki/Texas_Instruments.

4. NX-4006-530kv Brushless Quadcopter Motor. *HobbyKing.* [Interactiv] HobbyKing China. http://www.hobbyking.com/hobbyking/store/__17923__NX_4006_530kv_Brushless_Quadcopter_Motor.html.

5. Jennic JN5139. *Jennic.* [Interactiv] NXP Semiconductors. http://www.jennic.com/products/wireless_microcontrollers/jn5139.

6. PIC32MX795F512L. *Microchip.* [Interactiv] Microchip Technology Inc. http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en545660.

7. Real ICE. *Microchip.* [Interactiv] Microchip Technolgy Inc. http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en028120.

8. dsPIC33FJ32MC204. *Microchip.* [Interactiv] Microchip Technology Inc. http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en530334.

9. TC4428A. *Microchip.* [Interactiv] Microchip Technology Inc. http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010673.

10. **Liţă, Adrian-Ioan şi Cheleş, Mihai.** AN1160. *Microchip.* [Interactiv] Microchip Technology Inc. http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en533912.

11. dsPIC33EP512MU810. *Microchip.* [Interactiv] Microchip Technology Inc. http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en554307.

12. MK HexaXL. *Mikrokopter.* [Interactiv] http://www.mikrokopter.de/ucwiki/en/MK-HexaXL.

13. Yaw, Pitch, and Roll. *Collage Factory.* [Interactiv] http://collagefactory.blogspot.ro/2010/06/yaw-pitch-and-roll.html.

14. PIC32® Architecture. *Microchip.* [Interactiv] Microchip Technology Inc. http://www.microchip.com/pagehandler/en-us/family/32bit/architecture.html.

15. CH Robotics UM6-LT. *CH Robotics.* [Interactiv] CH Robotics. http://www.chrobotics.com/index.php?main_page=product_info&products_id=9.

16. Valiant, L. (1984), “Short Monotone Formulae for the Majority Function”, Journal of Algorithms 5: 363–366

17. B. Bose, “Modern Power Electronics and AC Drives”, Prentice Hall PTR, ISBN 0130167436

18. Ata Elahi, Adam Gschwender, "ZigBee Wireless Sensor and Control Network", Pearson Education, 2009

19. Lucio Di Jasio, "Programming 32-Bit Microcontrollers in C: Exploring the PIC32", Newnes, 2008

# Anexa 1 – Cod C destinat telecomenzii

## *1. Iniţializarea modului “Input Capture”*

```
TRISDbits.TRISD7 = 1;       //RX_CH1 input   RP 71
TRISDbits.TRISD6 = 1;       //RX_CH2 input   RP 70
TRISDbits.TRISD5 = 1;       //RX_CH3 input   RP 69
TRISDbits.TRISD4 = 1;       //RX_CH4 input   RP 68
TRISDbits.TRISD13 = 1;      //RX_CH5 input   RPI 77

//*****************************
//          Timer5 for IC module
//*****************************
TMR5 = 0;             // Resetting timer 1
PR5 = 20;
T5CON = 0x8020;   // internal Tcy*64 = 0.91428 uS

//*****************************
//            Input Capture init
//*****************************
IC1CON1 = 0x0C03;
IC2CON1 = 0x0C03;
IC3CON1 = 0x0C03;
IC4CON1 = 0x0C03;
IC5CON1 = 0x0C03;

_IC1IF = 0; // Clear the IC1 interrupt status flag
_IC1IE = 1; // Enable IC1 interrupts
_IC1IP = IC_INTERRUPT_PRIORITY;

_IC2IF = 0; // Clear the IC2 interrupt status flag
_IC2IE = 1; // Enable IC2 interrupts
_IC2IP = IC_INTERRUPT_PRIORITY;

_IC3IF = 0; // Clear the IC3 interrupt status flag
_IC3IE = 1; // Enable IC3 interrupts
_IC3IP = IC_INTERRUPT_PRIORITY;

_IC4IF = 0; // Clear the IC4 interrupt status flag
_IC4IE = 1; // Enable IC4 interrupts
_IC4IP = IC_INTERRUPT_PRIORITY;

_IC5IF = 0; // Clear the IC4 interrupt status flag
_IC5IE = 1; // Enable IC4 interrupts
_IC5IP = IC_INTERRUPT_PRIORITY;

IC1CON2 = 0x0000;
IC2CON2 = 0x0000;
IC3CON2 = 0x0000;
IC4CON2 = 0x0000;
IC5CON2 = 0x0000;
```

# *2. Citirea canalului RX_CH1 cu IC1. Analog oricare RX*

```
void __attribute__ ((__interrupt__, no_auto_psv)) _IC1Interrupt(void) {
        static unsigned int IC1TMR;
        unsigned int tmp1;
        unsigned int calcul1;           /* intermediate value */

        while (IC1CON1bits.ICBNE == 1)
                tmp1 = IC1BUF;              /* take the last value */

        if ((IC1CON1 & 1)==1)           /* rising edge */
        {
                IC1CON1 = (IC1CON1 & 0xfffe);     /* detect next falling edge */
        } else                      /* falling edge */
        {
                IC1CON1 = (IC1CON1 | 0x0001);     /* detect next rising edge */
                calcul1 = tmp1 - IC1TMR;
                rc_x2 = calcul1;   /* variabla rc_x2 tine valoarea ce va fi folosita */

                /* detectam daca acum a fost aprinsa telecomanda si memoram centrul */
                if((rc_initial_load < RC_LOAD_PERIODS)&&(rc_signal_on == 1)) {
                        rc_x2_center = rc_x2;
                        rc_initial_load++;
                }
        }

        IC1TMR = tmp1;
        _IC1IF = 0;
}
```

# Anexa 2 - Cod C al algoritmului de stabilizare

## *1. Codul rulat cu o frecvență de 500 Hz.*

```
desired_roll = (rc_x2 - rc_x2_center) / 2.0;
desired_pitch = (rc_y2 - rc_y2_center) / 2.0;
desired_yaw = (rc_x1 - rc_x1_center) / 2.0;

if(desired_roll > 360)
        desired_roll -= 32768;

if(desired_pitch > 360)
        desired_pitch -= 32768;

if(desired_yaw > 360)
        desired_yaw -= 32768;

ControllerRoll();
ControllerPitch();
ControllerYaw();

BLDC1_DUTY = MIN_DUTY1 + MIN_DUTY_ADD + my_touint(rc_throttle(1) + imu_x_delta(1)
+ imu_y_delta(1) + imu_yaw_delta(1));
BLDC2_DUTY = MIN_DUTY2 + MIN_DUTY_ADD + my_touint(rc_throttle(2) + imu_x_delta(2)
+ imu_y_delta(2) + imu_yaw_delta(2));
BLDC3_DUTY = MIN_DUTY3 + MIN_DUTY_ADD + my_touint(rc_throttle(3) + imu_x_delta(3)
+ imu_y_delta(3) + imu_yaw_delta(3));
BLDC4_DUTY = MIN_DUTY4 + MIN_DUTY_ADD + my_touint(rc_throttle(4) + imu_x_delta(4)
+ imu_y_delta(4) + imu_yaw_delta(4));
BLDC5_DUTY = MIN_DUTY5 + MIN_DUTY_ADD + my_touint(rc_throttle(5) + imu_x_delta(5)
+ imu_y_delta(5) + imu_yaw_delta(5));
BLDC6_DUTY = MIN_DUTY6 + MIN_DUTY_ADD + my_touint(rc_throttle(6) + imu_x_delta(6)
+ imu_y_delta(6) + imu_yaw_delta(6));


//MAX_DUTY_ALLOWED protection
if (BLDC1_DUTY > MAX_DUTY_ALLOWED)   BLDC1_DUTY = MAX_DUTY_ALLOWED;
if (BLDC2_DUTY > MAX_DUTY_ALLOWED)   BLDC2_DUTY = MAX_DUTY_ALLOWED;
if (BLDC3_DUTY > MAX_DUTY_ALLOWED)   BLDC3_DUTY = MAX_DUTY_ALLOWED;
if (BLDC4_DUTY > MAX_DUTY_ALLOWED)   BLDC4_DUTY = MAX_DUTY_ALLOWED;
if (BLDC5_DUTY > MAX_DUTY_ALLOWED)   BLDC5_DUTY = MAX_DUTY_ALLOWED;
if (BLDC6_DUTY > MAX_DUTY_ALLOWED)   BLDC6_DUTY = MAX_DUTY_ALLOWED;
```

## *2. aerodine.h*

```
#ifndef __AERODINE_H__
#define __AERODINE_H__

#include "defs.h"

void ZigBee_WriteString(const char *string);

#define ZERO_DUTY 1100
#define          MAX_DUTY 2182

#define MIN_DUTY1 1299     //1300 on
#define MIN_DUTY2 1302     //1303 on
#define MIN_DUTY3 1300  //1301 on
#define MIN_DUTY4 1298  //1299 on
#define MIN_DUTY5 1303  //1304 on
#define MIN_DUTY6 1299     //1300 on

#define MIN_DUTY_ADD  5   //add some so motors will start and kind of have the same speed

#define MAX_DUTY_ALLOWED        1700
#define MIN_THROTTLE                1300            //sub asta taie tot
#define MAX_THROTTLE_ALLOWED 1500    //cursa 1300 - 2000 --> 0 - 700 --> 500

#define POT_ARMING_LOW  1200
#define POT_ARMING_HIGH 2000

#define ARMING_STATE_UNARMED 0
#define ARMING_STATE_HALFPOT   1
#define ARMING_STATE_ARMED              2

#define RC_STALL_LIMIT              200             //*2 ms = 40 ms - how long will look for RC
controller
#define RC_LOAD_PERIODS             240             //how long to re-update the RC center at the
begining

#define IMU_MAX_XY_DELTA        50
#define IMU_MAX_YAW_DELTA       50

#define PID_ROLL_P   5000
#define PID_ROLL_I    4000

#define PID_PITCH_P 5000
#define PID_PITCH_I  4000

#define PID_YAW_P   5000
```

```
#define PID_YAW_I    4000


extern unsigned int bat_7v4; //instantaneous half voltage
extern unsigned int bat_14v8;          //instantaneous total voltage
extern unsigned int curr_inst;         //instantaneous current

//global variables that indicate controller status
extern unsigned char rc_signal_on;
extern unsigned int rc_signal_timestamp;
extern unsigned int rc_initial_load;

extern unsigned int rc_x1, rc_x2, rc_y1, rc_y2, rc_pot;
extern unsigned int rc_x1_center, rc_x2_center, rc_y2_center;

extern unsigned char hexa_armed;

extern unsigned int hexa_throttle;

extern signed int i_roll,i_pitch,i_yaw;

extern float desired_roll;
extern float desired_pitch;
extern float desired_yaw;

signed int my_toint(float c);
unsigned int my_touint(float c);

//functions for movement on RC
float rc_throttle(unsigned char N);
float imu_x_delta(unsigned char N);
float imu_y_delta(unsigned char N);
float imu_yaw_delta(unsigned char N);

void ControllerRoll();
void ControllerPitch();
void ControllerYaw();


#endif
```

## *3. aerodine.c*

```
#include "aerodine.h"
#include <math.h>

unsigned int bat_7v4;//instantaneous half voltage
unsigned int bat_14v8;          //instantaneous total voltage
unsigned int curr_inst;         //instantaneous current


//global variables that indicate controller status
unsigned char rc_signal_on;
unsigned int rc_signal_timestamp;
unsigned int rc_initial_load;

unsigned int rc_x1, rc_x2, rc_y1, rc_y2, rc_pot;
unsigned int rc_x1_center, rc_x2_center, rc_y2_center;

unsigned char hexa_armed;

unsigned int hexa_throttle;

float mysin[7] = {0, 1, 0.5,          -0.5,                  -1, -0.5,                  0.5};
float mycos[7] = {0, 0, 0.866025404, 0.866025404, 0, -0.866025404, -0.866025404};
float myyaw[7] = {0, 1, -1.0,              1.0,               -1, 1.0,                   -1};

signed int i_roll,i_pitch,i_yaw;

float desired_roll = 0;
float desired_pitch = 0;
float desired_yaw = 0;


unsigned int my_touint(float c) {
        return (unsigned int)ceil(c);
}

int my_toint(float c) {
        return (int)ceil(c);
}

float rc_throttle(unsigned char N) {
        hexa_throttle = rc_y1;

        if(hexa_throttle < MIN_THROTTLE)
                hexa_throttle = 0;
        else
```

```
                hexa_throttle -= MIN_THROTTLE;

        //protect against max throttle - definable
        if (hexa_throttle > MAX_THROTTLE_ALLOWED)
                hexa_throttle = MAX_THROTTLE_ALLOWED;
        return (float)hexa_throttle;
}

float imu_x_delta(unsigned char N) {
        return mycos[N]*i_roll;
}

float imu_y_delta(unsigned char N) {
        return mysin[N]*i_pitch;
}

float imu_yaw_delta(unsigned char N) {
        return myyaw[N]*i_yaw;
}

void ControllerRoll() {
        //ROLL PID
        PIDStructureROLL.qInMeas = my_touint((360+desired_roll));
   //PIDStructureROLL.qInRef = my_touint(360+imu_roll);
   PIDStructureROLL.qInRef = my_touint((360+imu_roll));
        CalcPI(&PIDStructureROLL);
        i_roll =  PIDStructureROLL.qOut;
}

void ControllerPitch() {
        //PITCH PID
        PIDStructurePITCH.qInMeas = my_touint((360+desired_pitch));
   //PIDStructurePITCH.qInRef = my_touint(360+imu_pitch);
   PIDStructurePITCH.qInRef = my_touint((360+imu_pitch));
        CalcPI(&PIDStructurePITCH);
        i_pitch =  PIDStructurePITCH.qOut;
}

void ControllerYaw() {
        //YAW PID
        PIDStructureYAW.qInMeas = my_touint(360+desired_yaw);
   PIDStructureYAW.qInRef = my_touint(360+imu_yaw);
        CalcPI(&PIDStructureYAW);
        i_yaw =  PIDStructureYAW.qOut;
}
```