UNIVERSITATEA "POLITEHNICA" DIN BUCUREŞTI
FACULTATEA DE ELECTRONICĂ, TELECOMUNICAŢII ŞI TEHNOLOGIA INFORMAŢIEI

SISTEM ROBOTIC AERIAN AUTONOM. SISTEM DE TRANSMISIE ȘI DE ANALIZĂ A DATELOR VIDEO

PROIECT DE DIPLOMĂ

prezentat ca cerinţă parţială pentru obţinerea titlului de
Inginer în domeniul *Electronică şi Telecomunicaţii*
programul de studii de licenţă *Electronică Aplicată(ETC - ELA)*

Conducători ştiinţifici
Ș.L. Dr. Ing. Horia CUCU
Prof. Dr. Ing. Corneliu BURILEANU

Absolvent
Arlind LIKA

București
2016

Anexa1

Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Departamentul Electronică Aplicată si Tehnologia Informaţiei

Aprobat Director de Departament :
Prof. Dr. Ing. Sever PAŞCA

TEMA PROIECTULUI DE DIPLOMĂ
a studentului LIKA P. Arlind, grupa 444B

1. Titlul temei: **Sistem robotic aerian autonom. Sistem de transmisie şi de analiză a datelor video.**

2. Contribuţia practică, originală a studentului va consta în (în afara părţii de documentare):

Proiectul are ca scop obţinerea unui quadcopter care să fie telecomandat şi să transmită date la staţia de recepţie unde acestea vor fi prelucrate. Proiectul va consta în următoarele etape:

- *Asamblarea quadcopterului şi verificarea funcţionării acestuia.*
- *Realizarea unei comunicaţii între microcontrolerul responsabil de menţinerea stabilităţii şi cel responsabil de autonomia robotului*
- *Integrarea şi utilizarea în sistem a senzorilor şi a camerei video pentru achiziţia de date; transmisia şi prelucrarea acestora*
- *Simularea software a unei telecomenzi pentru rutina de calibrare, testele preliminare şi controlarea quadcopterului.*

3. Proiectul se bazează pe cunoştinţe dobândite în principal la următoarele 3-4 discipline: *Microcontrolere, Programarea Calculatoarelor, Prelucrarea digitală a semnalelor, Sisteme de comunicaţie.*

4. Proprietatea intelectuală asupra proiectului aparţine: *Laborator de Cercetare Speech and Dialogue (SpeeD), studentului Lika Arlind.*

5. Realizarea practică rămâne în proprietatea: *Laborator de Cercetare Speech and Dialogue (SpeeD), studentului Lika Arlind.*

6. Locul de desfăşurare a activităţii: *UPB-ETTI, Sala 3 de Calculatoare.*

7. Data eliberării temei: *August 2015.*

CONDUCĂTOR LUCRARE:

Ş.L. Dr. Ing. Horia CUCU

Prof. Dr. Ing. Corneliu BURILEANU

STUDENT:

Arlind LIKA

Anexa 5

Declaraţie de onestitate academică

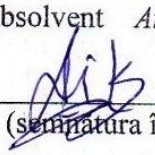
Prin prezenta declar că lucrarea cu titlul "*Sistem robotic aerian autonom. Sistem de transmisie şi de analiză a datelor video*", prezentată în cadrul Facultăţii de Electronică, Telecomunicaţii şi Tehnologia Informaţiei a Universităţii "Politehnica" din Bucureşti ca cerinţă parţială pentru obţinerea titlului de *Inginer* în domeniul *Electronică, Telecomunicaţii şi Tehnologia Informaţiei* programul de studii *Electronica Aplicată* 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, *27.06.2016*

Absolvent *Arlind LIKA*



(semnătura în original)

Cuprins

LISTĂ DE FIGURI

LISTĂ DE ACRONIME

A

ARM = Advanced RISC Machine

B

BEC = Battery Eliminator Circuit

C

CH = Channel

D

DH = Data High

DHCP = Dynamic Host Configuration Protocol

DL = Data Low

E

ESC = Electronic Speed Controller

G

GND = Ground

GPIO= General Purpose Input/Output

GUI = Graphical User Interface

H

HD = High-Definition

HDMI = High-Definition Multimedia Interface

HTML = Hyper Text Markup Language

HTTP = HyperText Transfer Protocol

I

IDE = Integrated Development Environment

IEEE = Insitute of Electrical and Electronics Engineers

IP = Internet Protocol

J

JPEG = Joint Photographic Experts Group

JSP = JavaServer Pages

L

LED = Light Emitting Diode

LiPO = Litium polimer

M

M-JPEG = Motion JPEG

P

PAN ID = Personal Area Network ID

PCB = Printed Circuit Board

PHP = Hypertext Preprocessor

PPM = Pulse Position Modulation

PWM = Pulse Width Modulation

R

RAM = Random Accecs Memory

RPi = Raspberry Pi

RPM = Revolutions Per Minute

RSSI = Received Signal Strength Indication

Rx = Recepție

S

SoC = System On Chip

SSH = Secure Shell

SSID = Service Set Identifier

U

UART = Universal Asynchronous Receiver/Transmitter

URL = Uniform Resource Locator

USB = Universal Serial Bus

T

TTL = Transistor-Transistor Logic

Tx = Transmisie

V

VNC = Virtual Network Control

W

Wi-FI = Wireless

WLAN= Wirelss Local Area Network

WSGI = Web Server Gateway Interface

INTRODUCERE

MOTIVAȚIA

Quadcopterele, numite și vehicule aeriene fără pilot, au un potențial larg în îndeplinirea sarcinilor pe cât de importante, pe atât de periculoase. Acestea sunt utilizate frecvent în armată, în agricultură, în acitvitațile sportive și în diferite proiecte de cercetare. Progresele în domeniul electronicii și apariția echipamentelor necesare pentru controlul quadcopterelor au permis producerea în masă a acestor vehicule. Cercetări legate de dezvoltarea lor se fac în continuare cu scopul de a îmbunătăți performanțele acestor echipamente.

Aceasta popularitate și necesitate pentru quadcoptere a fost una dintre motivele pentru care eu am ales această tema. Această lucrare include toate etapele necesare pentru construirea unui quadcopter capabil să zboare și să transmită în timp real imagini sau video folosind o cameră montată pe șasiu. Proiectul însuși necesită cunoștințe din multe domenii diferite începând cu mecanica, electronica, programarea și comunicațiile; acesta fiind un alt motiv de alegere a temei.

Există foarte multe aplicații în care quadcopterele pot fi folosite: ca echipamente de supraveghere în timpul evenimentelor publice sau demonstrații, în operațiunile de recuperare în zonele în care este greu de ajuns, pentru securitate la o proprietate privată. În toate exemplele enumerate de mai sus, avantajul principal este decolarea verticală și utilizarea în aproape orice mediu pentru mai multe sarcini.

Figura 1 Quadcopter Elev-8 cu HoverFly Open

OBIECTIVELE

Această lucrare își propune următoarele obiective:

- Asamblarea quadcopterului și verificarea funcționării acestuia.
- Realizarea unei comunicații între microcontrolerul responsabil de menținerea stabilității și cel responsabil de autonomia robotului.
- Simularea software a unei telecomenzi pentru testele preliminare și controlarea quadcopterului
- Integrarea și utilizarea în sistem a senzorilor și a camerei video pentru achiziția de date; transmisia și prelucrarea acestora.

Pentru realizarea proiectului se vor folosi piese dintr-un kit de dezvoltare de la Parallax[1]. Componentele necesare pentru construirea quadcopterului trebuie aranjate și lipte conform regulilor. Schema și cadrul folosit vor fi prefabricate. Tot prefabricate sunt și piese cum ar fi motorarele și ESC-urile,legăturile electrice ale acestora fiind realizate personal în laborator. Un PCB pentru fixarea componentelor în cadrul quadcopterului va fi proiectat în programul PROTEUS.

Cerințele menționate mai sus trebuie îndeplinite pentru a proiecta un prototip cât mai performant în lumea reală. O cameră pentru achiziția imaginilor trebuie integrată în sistem. Poziția ei se va calcula având în vedere unghiul si direcția din care se va face achiziția video. Un sistem de comunicație care va ajuta în controlul quadcopterului de la distanță se va implementa. Trebuie luată în considerare toată greutatea adaugată (baterie mai mare, microcontroller în plus, camera și alte component introduse) pentru a nu afecta în mod semnificativ controlabilitatea și timpul de zbor al quadcopterului. La sfârșit, folosind doar fluxul video, utilizatorul ar trebui să fie capabil să controleze quadcopterul de la distanță folosind tastatura unui laptop. În stația de bază (PC, Laptop) se va face transmisia video live și prelucrarea imaginilor.

Desfășurarea activității (testele preliminare și realizarea proiectului) se va face în laboratorul de cercetare SpeeD.

STRUCTURA

Această lucrare este compusă din 5 capitole. După această parte introductivă, vor urma noțiuni de teorie legate de quadcopter.

În Capitolul 1 se va descrie quadcopterul utilizat, Parallax Elev-8[1] cu specificațiile lui. Fiind un produs comercial, acesta oferă un acces foarte limitat la software, prin urmare vom trata microcontrolerul de zbor ca o cutie închisă cu care putem să comunicăm prin anumite semnale. Componentele hardware folosite vor fi prezentate cu proprietațile lor.

În Capitolul 2 vor fi discutate noțiuni teoretice despre imaginea digitală și metode folosite pentru prelucrarea imaginii și detecția de obiect.

În Capitolul 3 se vor introduce tehnologiile software folosite în acest proiect. Se vor prezenta aplicațiile folosite și tehnologiile de programare necesare pentru controlarea sistemului și pentru transmisia și comunicația cu el.

În Capitolul 4 se va introduce partea practică și procedurile necesare pentru pregătirea quadcopterului de zbor în teren deschis. Se vor descrie toate testele necesare pentru controlul lui și pentru realizarea comunicației la distanță.

În Capitolul 5 se va face transmisia video live pe o pagina web locală și prelucrarea frame-urilor obținute. O prelucare mai detaliată a imagini va avea loc în stația de bază.

În sfârșit vor fi puse în evidență concluziile sistemului studiat și tot așa se va face o prezentare legată de îmbunătățirea sistemului din toate punctele de vedere.

CAPITOLUL 1

SISTEM AUTONOM DE TIP QUADCOPTER

1.1 Despre quadcopter

Un quadcopter este un vehicul aerian cu 4 elice care poate zbura. Cele 4 elice sunt plasate în exteriorul motoarelor de tip brushless care în funcție de ajustarea vitezei lor, pot face manevre în spațiul tridimensional. Motoarele sunt fixate pe un cadru cu configurație de tip X (există și alte configurații de tip +).

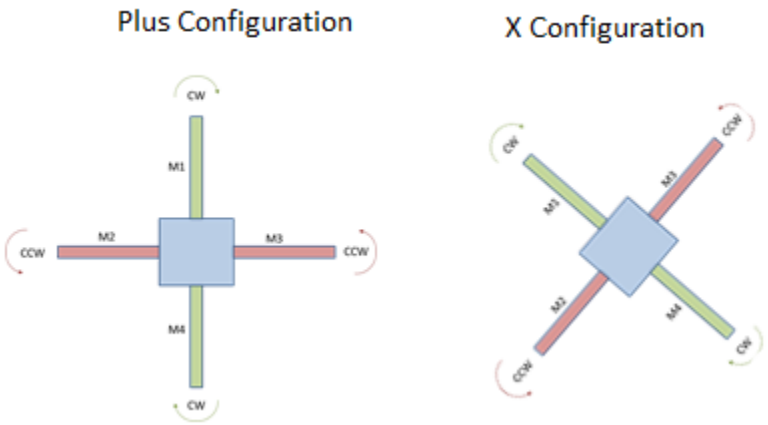


Figura 1-1 Tipuri de configurații la un quadcopter

Un quadcopter cu configurație de tip X are stabilitate și accelerație mai mare și oferă vizibilitate mai bună pentru imagine fără a fi obturată de aripile cadrului. [2]

Quadcopterul este un concept popular pentru drone, datorită capacităților sale. Avantajul majore ale dronelor este posibilitatea de a fi manevrate în spații foarte mici. În prezent quadcopterele au devenit mai accesibile și sunt dotate cu cameră video, senzori, GPS și alte facilități. Și, desigur, comunicația fără fir oferă acces și control la distanțe foarte mari.

Mișcarea în diferite direcții se realizează ajustând viteza pe fiecare motor. Motoarele sunt puse simetric față de centrul de greutate și se rotesc 2 în sensul orar și 2 în sensul invers.

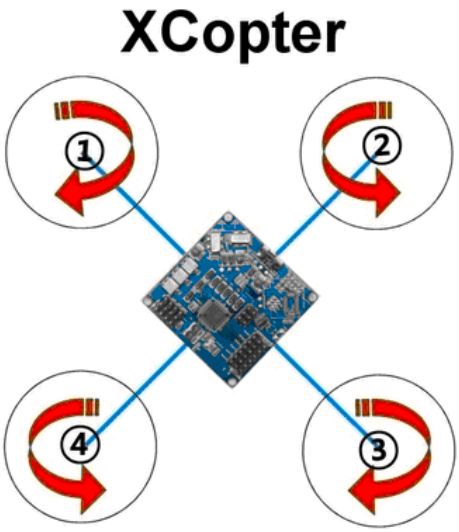


Figura 1-2 Poziția motoarelor pe quadcopter

Modelul de la Parallax[1] este un quadcopter comercial și în mod normal controlarea lui se face cu o telecomandă radio. În proiectul nostru, vom folosi un microcontroler și tastatura unui laptop pentru a simula comportamentul asemănător între stația de tansmisie și cea de recepție.

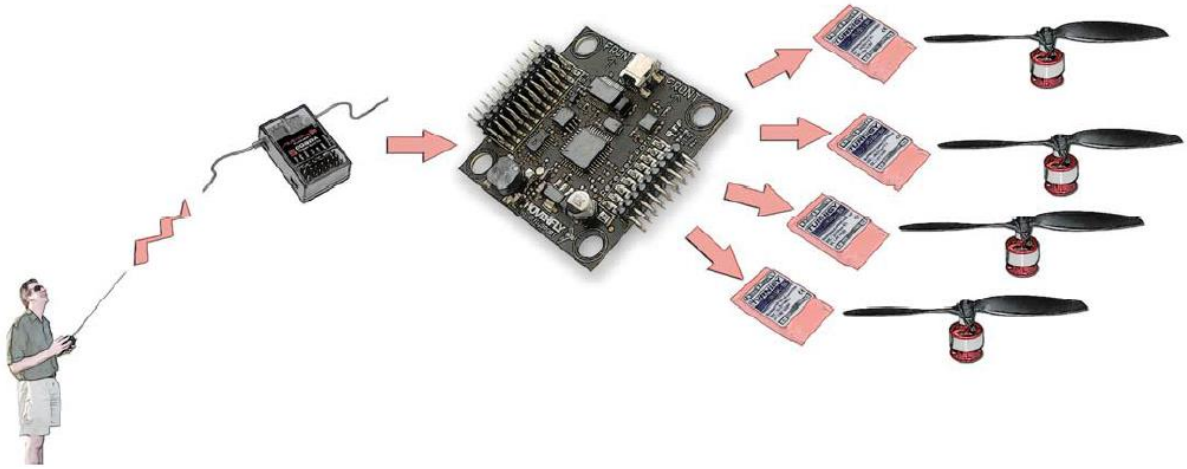


Figura 1-3 Comanda quadcopterului Elev-8

În continuare vom păstra componentele hardware ale quadcopterului (microcontrolerul de zbor, ESC-urile, motoarele, cadrul) și vom crea un software pentru control. Acest software trebuie să fie capabil să se comporte ca un controler radio și să pastreze în mod continuu comunicația cu quadcopterul. Prin apăsarea diferitelor taste, trebuie să realizăm manevre în toate direcțiile posibile.

1.1.1 Schema bloc

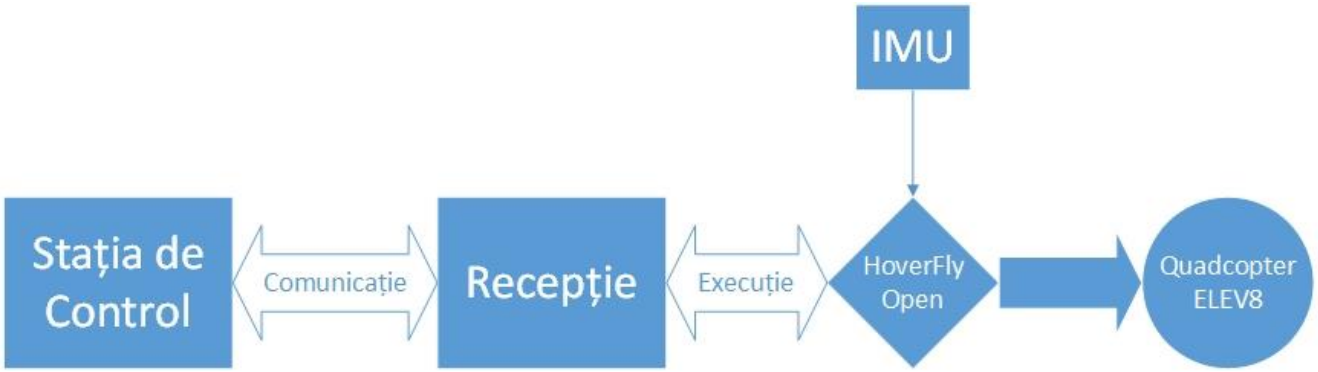


Figura 1-4 Schema bloc

În acest proiect, schema de mai sus reprezintă modul de funcționare al quadcopterului. Se va stabili o comunicație între stația de bază și cea de recepție. Stația de bază va fi un calculator pe care îl vom folosi pentru transmiterea comenzilor la un microncontroler care va juca rolul stației de recepție. Acest microcontroler va recunoaște comenzile și va genera mai departe semnalele necesare. În cazul nostru, execuția se referă la generarea semnalelor care vor fi recunoscute de controlerul de zbor. Acest

controler, în funcție de semnalele primite și informația primită de la senzorii interni ai sistemului, va transmite către quadcopter semnalele necesare pentru funcționarea acestuia.

Acest mod de funcționare va ramane valabil în orice moment de timp al zborului. Comunicația stabilită va ajuta în controlul quadcopterului. Pentru crearea unui sistem cât mai bun, această comunicație trebuie să fie fără fir și cu raza de funcționare cât mai mare.

1.1.2 Schema principală și arhitectura

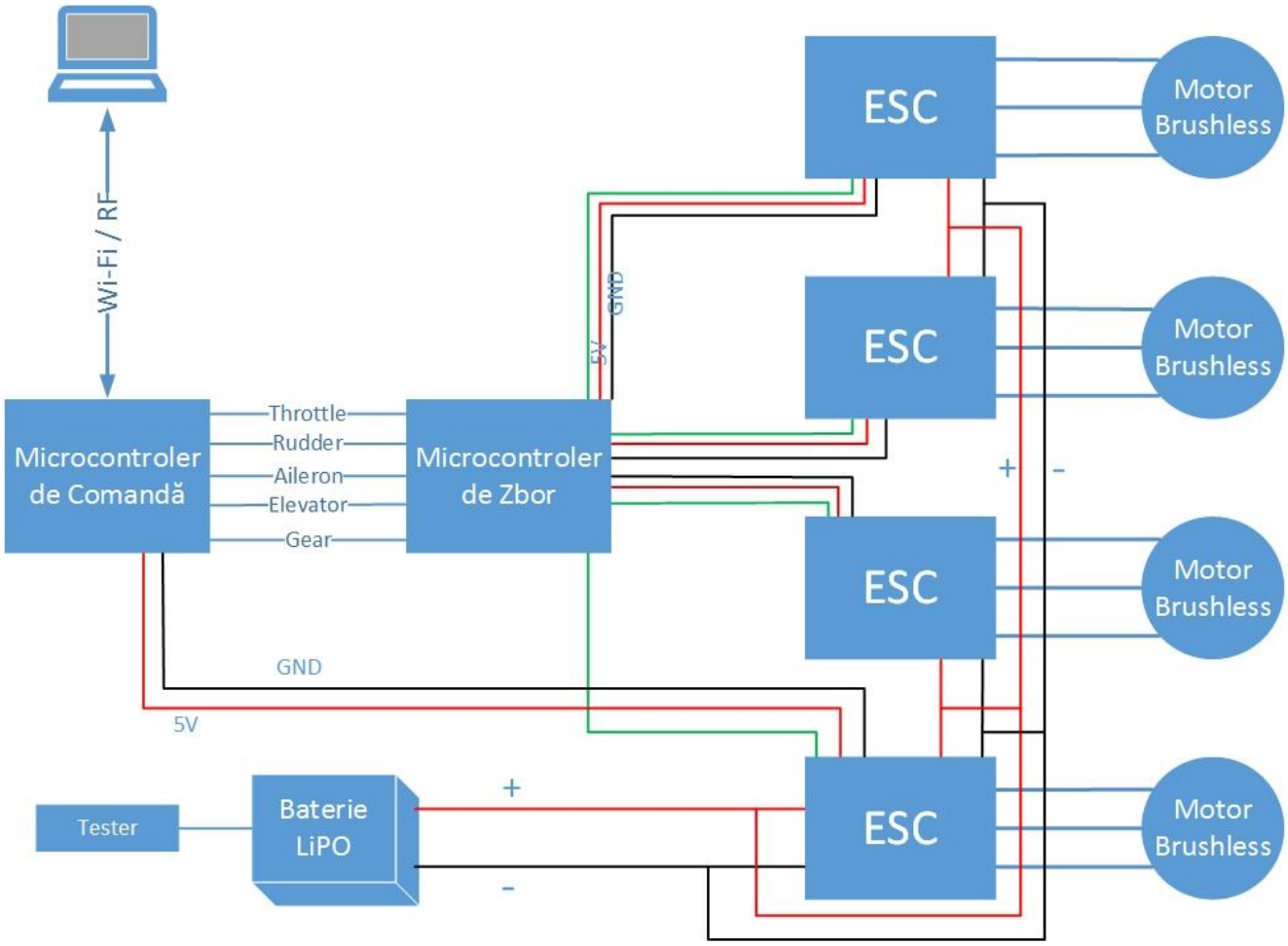


Figura 1-5 Schema și arhitectura quadcopterului

HoverFly Open[3] este microcontrolerul responsabil pentru stabilizarea quadcopterului. El va comunica cu un alt microcontroler (Arduino, Raspberry Pi) prin care va prelua comenzi pentru controlul quadcopterului. Pentru realizarea mișcării, este necesară înclinirea întregului sistem, doarece elicele sunt fixate pe cadru. Cu ajutorul componentelor numite ESC (Electronic Speed Controller), motoarele își vor schimba viteza pentru realizarea deplasării. ESC-urile vor fi comandate de controlerul de zbor HoverFly Open.

Transmiterea comenzilor de pe calculator se va face folosind module de comunicație XBee sau prin WiFI. Aceste comenzi vor ajunge la microcontrolerul de comandă care va genera semnale de tip PWM în funcție de starea în care se află. Aceste semnale vor ajunge la HoverFly Open care le va folosi mai departe pentru control.

1.1.3 Schema de funcționare prin stări

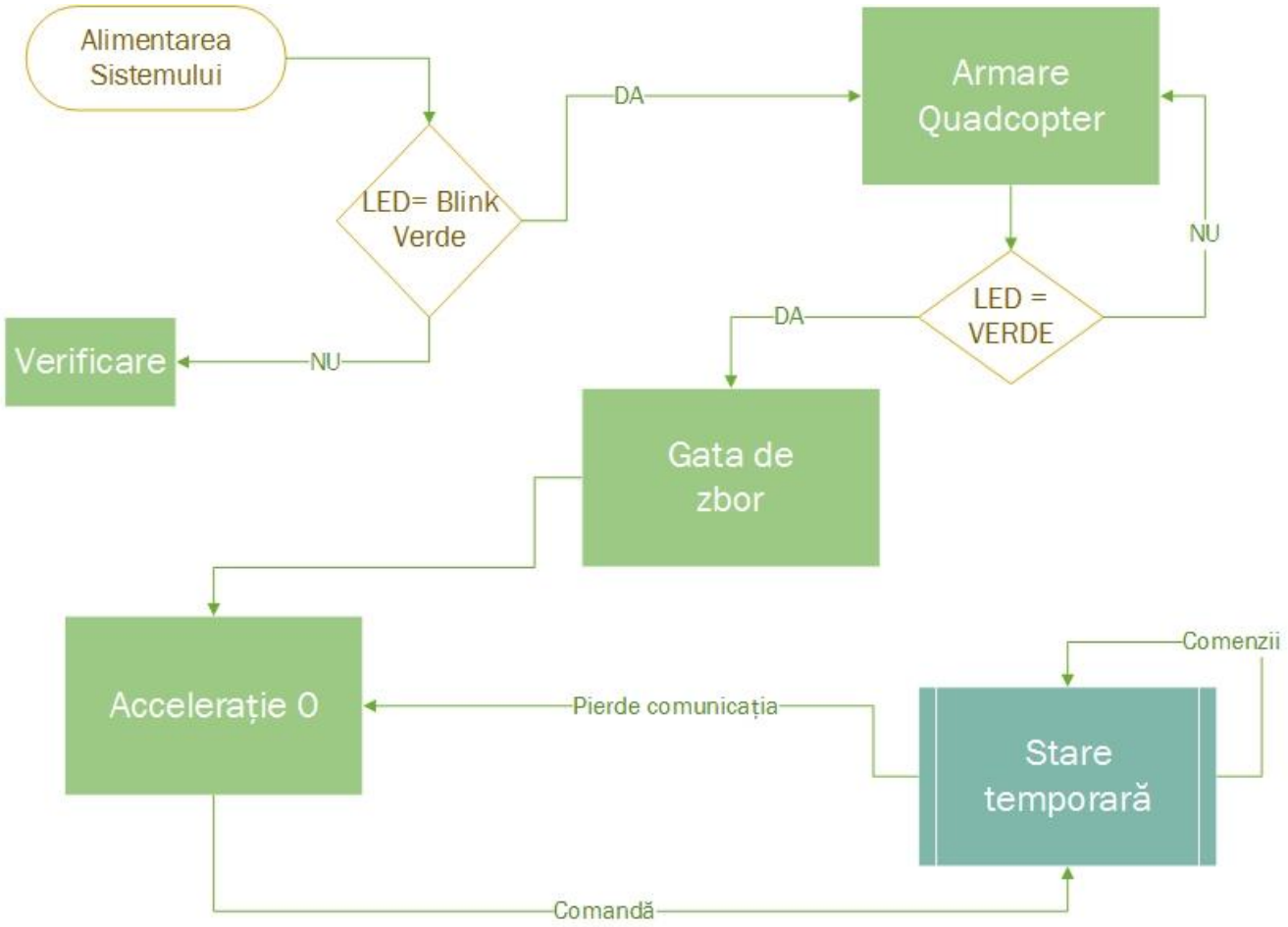


Figura 1-6 Schema de funcționare prin stări

Pentru a ridica quadcopterul de la sol, există o procedură care trebuie urmărită și în care se verifică pas cu pas dacă condițiile necesare pentru zbor sunt îndeplinite. După alimentarea bateriei, se va verifica dacă semnalele trimise de către microcontrolerul de zbor sunt corecte. Dacă da, atunci quadcopterul este gata de armat. În caz contrar, trebuie revizuit programul astfel încât să îndeplinească cerințele necesare.

Dupa faza de armare, quadcopterul este gata și microcontrolerul de zbor va aștepta în continuare comenzi pentru controlarea sistemului.

Din motive de siguranță, zborul va începe mereu cu semnalul de accelerație la valoarea minimă și în caz că la un moment dat se va pierde legatura între stația de control și microcontrolerul de comandă, quadcopterul va primi din nou starea de la început fără a avea posibilitatea de a crea probleme exterioare grave.

1.2 Microcontrolerul de zbor

HoverFly Open[3] este un microcontroler de zbor produs de firma Parallax și are capacitatea de a controla 4 sau mai multe motoare. Acesta se alimentează direct de la ESC și nu are nevoie de o baterie separată.

Funcția principală a acestui microcontroler este monitorizarea și controlarea motoarelor pentru menținerea stabilității. Pentru a face aceasta, el folosește datele provenite de la giroscop, accelerometru și senzor de presiune.

Figura 1-7 Microcontroler HoverFly Open

Microcontrolerul acesta va fi privit în continuare ca pe o entitate funcțională la care nu avem acces, dar putem să îi transmitem semnale de intrare și să folosim semnalele de ieșire pentru control. Intrarea acestui microcontroler este constituită din 5 canale, fiecare fiind responsabil pentru o manevră de zbor.

Figura 1-8 Canalele de comandă ale microcontrolerului de zbor

A = Este canalul de Aileron (Roll=Ruliu)

R = Este canalul de Rudder (Yaw=Girație)

T = Este canalul de Throttle (Accelerație)

E = Este canalul de Elevator (Pitch=Tangaj)

G = Este canalul responsabil pentru menținerea altitudinii(Gear)

Pinul negru al fiecărui canal reprezintă masa, pinul roșu alimentarea iar cel alb reprezintă pinul responsabil pentru control. Pe acest pin se transmite semnalul de tip PWM care controlează mai departe mișcarea quadcopterului.

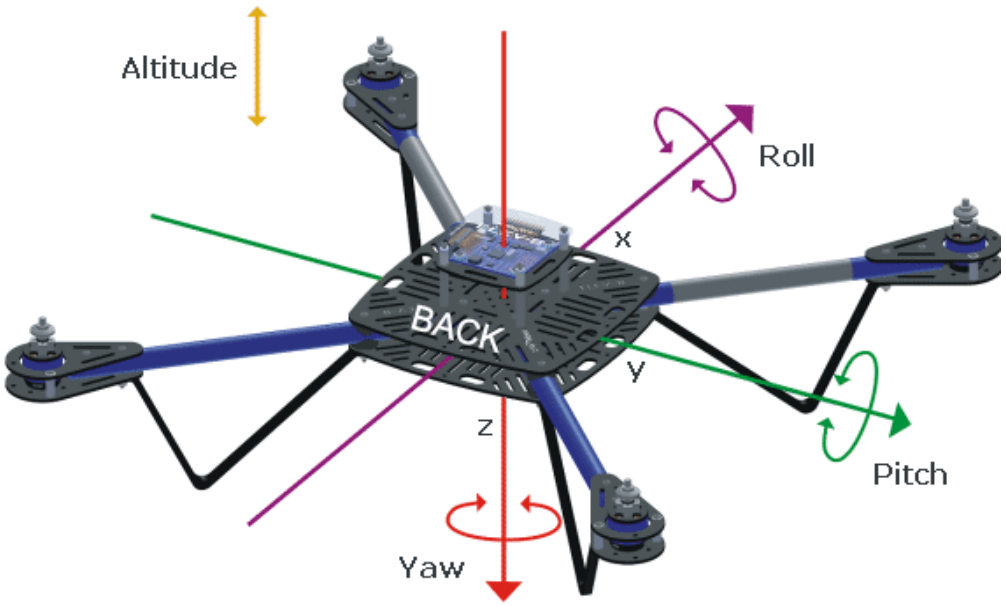


Figura 1-9 Direcțiile de mișcare ale unui quadcopter

Canalul de Throttle de pe microcontroler este responsabil pentru schimbarea altitudinii. Canalul Yaw descrie mișcarea după axa z, canalul Pitch după axa y și canalul Roll după axa x. Pentru realizarea deplasării se folosește comanda de Pitch care înlină quadcopterul după axa y și având canalul de Throttle activat, se realizează mișcarea înainte sau înapoi.

Toate cele 5 canale lucrează cu semnale de tip PWM[4] cu frecvența *f=50Hz* si cu facor de umplere de la 5% pana la 10%.

$$f=50Hz \Rightarrow T=20ms$$

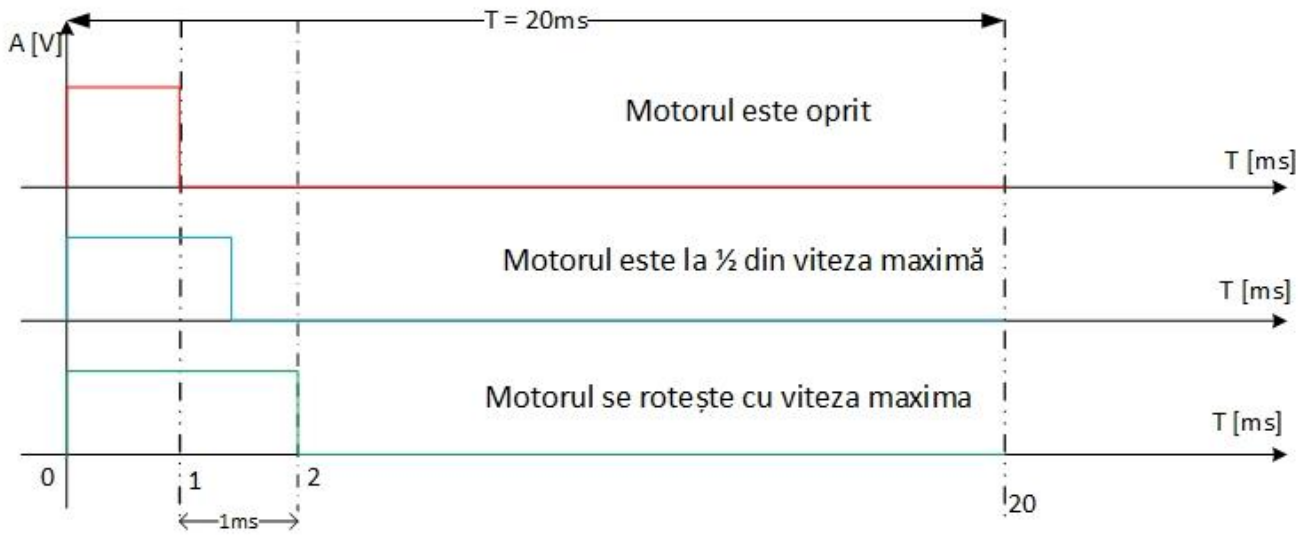


Figura 1-10 Semnale de tip PWM pentru controlul motoarelor

După cum se observă, este nevoie de 5 canale separate pentru a trimite comenzi de mișcare. În cazurile când quadcopterul se comandă folosind un controler radio, el folosește doar un canal unind toate semnalele într-unul singur numit PPM(Pulse Position Modulation)[5]. Semnalele separate folosesc doar intervalul 0-2ms pentru a trimite informația necesară așa că pentru a nu pierde cele 18ms, semnalul de tip PPM separă perioada lui în intervale mai mici de 2ms, fiecare corespunzând unui canal diferit. În felul acesta, semnalul PPM folosește toată perioada de 20ms și poate să comande în paralel pana la 10 canale diferite. În cazul nostru avem nevoie doar de 5 canale.

Pentru a simula controlerul radio, se va folosi același principiu și semnalele se vor genera de la microcontrolerul de comanda (Arduino și Raspberry Pi)

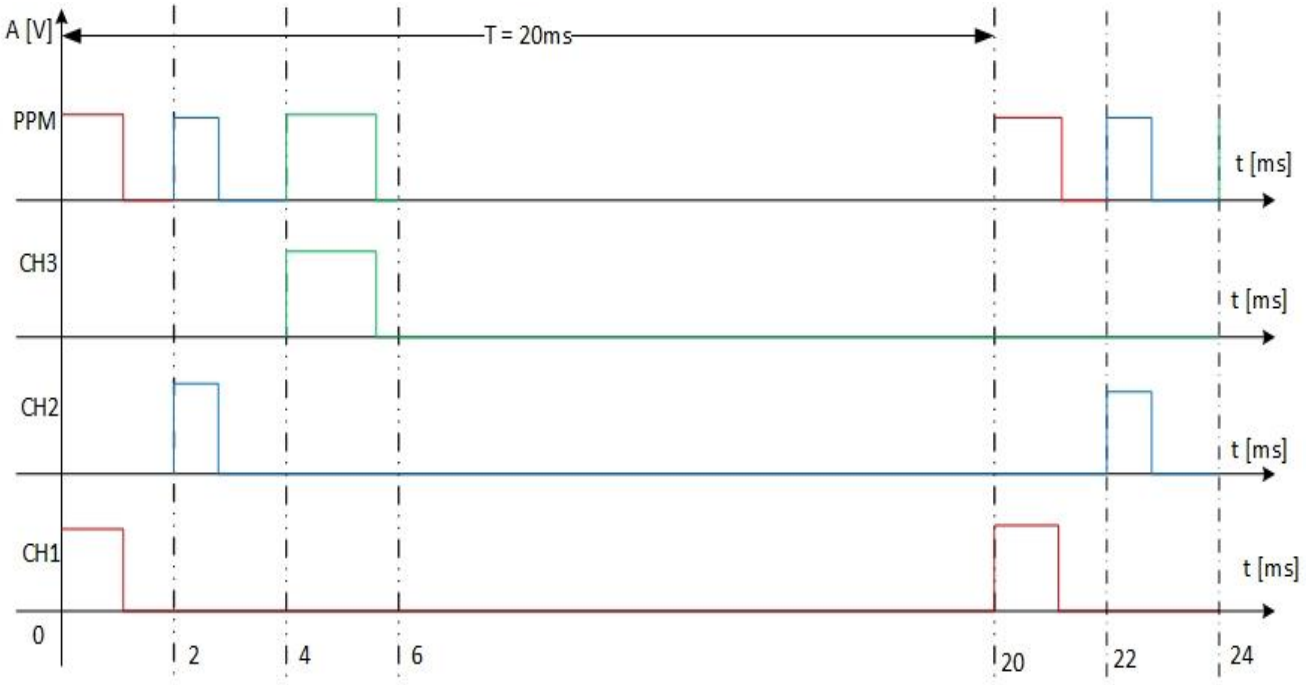


Figura 1-11 Semnale de tip PPM convertite în PWM

1.3 Microcontrolerul de comandă

1.3.1 Arduino Uno

Așa cum s-a precizat mai sus, pentru trimiterea comenzilor la HoverFly Open am folosit la testare o placă de dezvoltare de tip Arduino Uno[6]. Arduino este o platformă de calcul fizic bazată pe o placă cu multe intrări/ieșiri și un mediu de dezvoltare care implementează limbajul de procesare. Acest limbaj include mai multe funcții din limbajul C, dar și biblioteci specifice folosite doar în Arduino IDE. Platforma oferă posibilitatea de a face orice proiect în care sunt implicite intrări diferite, și aplicația dezvoltată în Arduino IDE ajută la controlarea ieșirilor în funcție de acele intrări. Această abordare

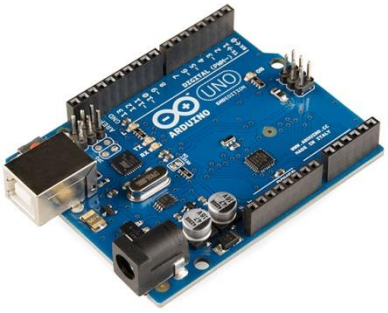


Figura 1-12 Arduino Uno

permite clafisicarea aplicațiilor conform rolul lor, ca sisteme autonome sau sisteme dependente și de control.

La baza acestei plăci de dezvoltare este un microcontroler ATMega 328P care funcționează pe 16MHz, cu memorie FLASH de 32kB. Are 14 pini digitali de ieșire și 6 pini de intrare de tip analog. Comunicația cu calculatorul este de tip UART TTL (cu pinii Tx și Rx, unde primul este de transmisie și al doilea de recepție).

În proiectul nostru am ales să folosim Arduino pentru o soluție mai bună și mai simplă. El oferă avantaje majore pentru că:

- Ambele direcții de dezvoltare hardware și software sunt ușor accesibile și pot fi ușor personalizate și extinse în funcție de proiect.
- Oferă intrări digitale și analogice, o interfață serial și ieșiri digitale cu semnale de tip PWM.
- Este foarte ușor de folosit, se conectează și comunică cu calculatorul folosind un cablu de tip USB. Se alimentează la o tensiune de 5V.
- Este realtiv ieftin și open-source, cu multe programe/coduri deja funcționale.

1.3.2 Raspberry Pi 3

Raspberry Pi [7] este un minicalculator care poate fi conectat la mai multe periferice și poate funcționa la fel de bine ca un calculator normal. El oferă multe posibilități pentru programare fiindcă poate să ruleze și compileze mai multe limbaje cum ar fi Python, C, C++. În proiectul nostru, avantajul principal este folosirea interfeței Wi-FI prin care putem să comunicăm și să primim/transmitem diferite informații fără a avea nevoie de legături fizice.

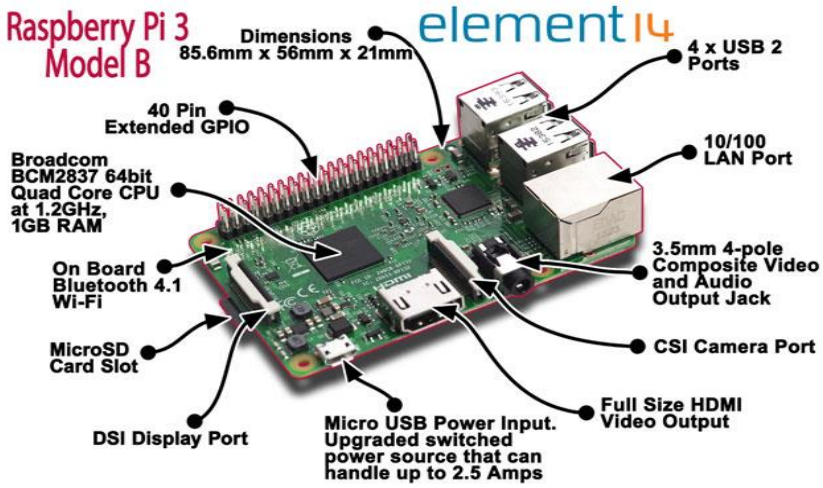


Figura 1-13 Raspberry Pi 3

Nucleul dispozitivului este sistemul de tip SoC de la BroadCom BCM 2837 cu un procesor de tip ARM QuadCore cu frecvența de lucru 1,2GHz și memorie RAM de 1GB. Raspberry Pi folosește sistemul de operare Raspbian și profită de multe pachete și biblioteci funcționale în Debian/Linux.

Raspberry Pi se alimentează la 5V direct de la calculator cu cablu de tip USB sau folosind o baterie așa cum este cazul nostru. În mod normal consumă un curent de 700mA care ajunge la 1000mA sau mai mult în funcție de dispozitivele periferice conectate la el. Portul de HDMI este folosit pentru a se conecta la ecrane cu rezoluție mare, iar în ceea ce privește camera HD, aceasta poate fi conectată printr-un cablu cu interfața serială. Portul de Ethernet se folosește pentru conexiunea la internet iar porturile USB pentru echipamente periferice care pot fi adăugate în sistem.

Porturile GPIO [8] sunt folosite ca pini generali de intrare și ieșire. În total sunt 40 de pini GPIO în care sunt incluși și pinii pentru comunicația UART care este controlată intern de către sistemul de operare.

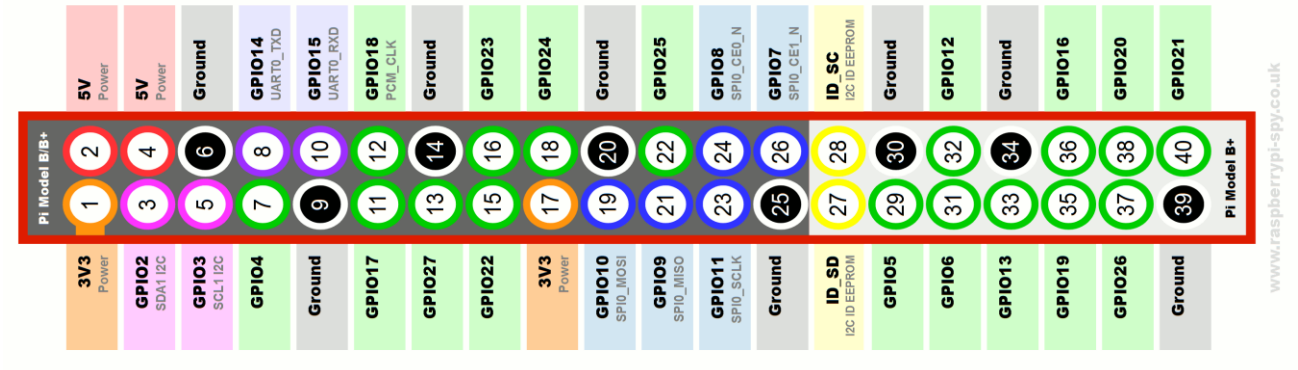


Figura 1-14 Pini de GPIO la RPi 3

1.4 Controlerul de viteză al motoarelor ESC

ESC este componenta electronică care controlează viteza motoarelor. În acest proiect sunt folosite 4 ESC-uri de la GemFan, fiecare responsabil pentru un motor. Ele se alimeantează la baterie cu cele 2 fire de intrare și la cele 3 fire din partea opusă se conectează motorul de tip brushless. Dacă motorul nu se rotește în direcția dorită, atunci este necesar doar să schimbăm 2 din cele 3 fire între ele. Această componentă conține un circuit integrat inteligent care schimbă viteza motoarelor în funcție de comanda primită. Rolul cel mai important al său este de a converti curentul continuu din baterie în curent alternativ. Cele 3 fire legate la motor conduc curent alternativ în defazaj de 120°. Schimbând factorul de umplere al semnalului PWM primit, ESC modifică viteza motorului.

Figura 1-15 Electronic Speed Controller (ESC)

Aceste piese trebuie calibrate astfel încât să cunoască valorile minime și maxime care corespund unui semnal de tip PWM. Semnalul de comandă în acest caz va fi trimis de Hoverfly Open folosind bara cu

3 pini, unde firul negru este ground, roșu este alimentarea și alb este comanda. ESC-urile conțin si un circuit electronic numit BEC (Battery Eleminator Circuit) [9] care va fi folosit pentru a alimenta cele 2 microcontrolere fără a fi nevoie de altă baterie. Acest circuit BEC generează la pinul de ieșire 5V (necesar pentru alimentarea unui microcontroler).

1.5 Motoarele brushless

Motoarele de tip brushless sunt utilizate în vehiculele de zbor datorită arhitecturii mai simple și duratei mai mari de viață. Ele sunt alimentate de la ESC-uri de la care primesc curent alternativ. Rotația se realizează injectând current la 2 din cele 3 înfășurări imobile pe rând și în mod periodic. [10]

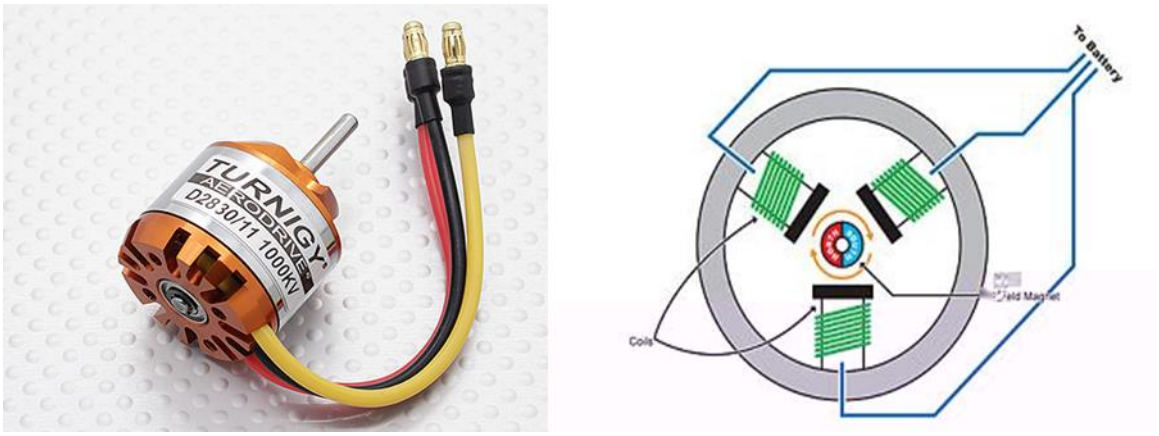


Figura 1-16 Motoare de tip Brushless

Cele 3 conductoare din ESC furnizează curent alternativ motorului și în funcție de frecvența de funcționare se definește și viteza motorului. Pentru o viteză mai mare, ESC-ul primește un semnal PWM cu factor de umplere mai mare și dă la ieșire un curent mai mare, ceea ce face motorul să se învârtă mai repede.

1.6 Baterie LiPO 3S

Bateria de tip LiPO este una dintre cele mai importante componente din proiect. Ea asigură autonomia quadcopterului și este folosită pentru alimentarea întregului sistem. Durata de zbor a quadcopterului depinde de capacitatea bateriei.

Această baterie, numită baterie litiu-polimer este reîncarcabilă și compusă din 3 celule. Fiecare celulă furnizează tensiunea nominală de 3,3-3,7V. Având în serie 3 celule, bateria ajunge la valoarea de 11,1V (valoarea necesară în proiectul nostru).

Pentru încarcarea bateriei se va folosi un încărcător special și starea bateriei va fi monitorizată în continuare folosind un tester care afișează la fiecare moment de timp statusul bateriei.

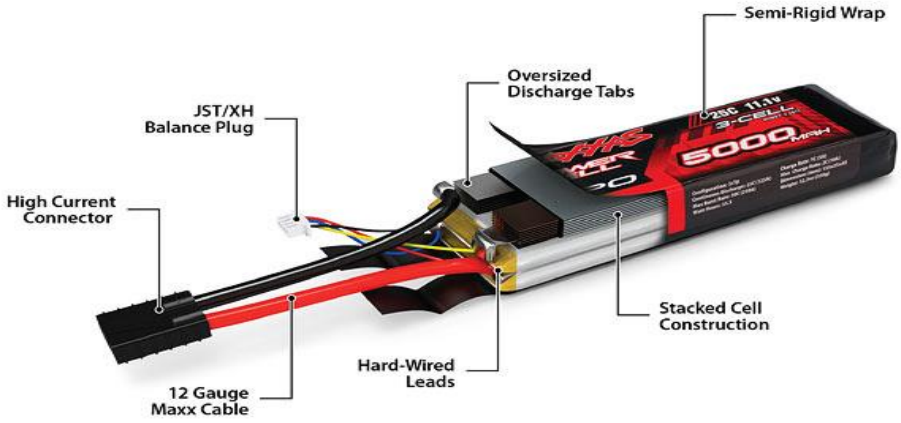


Figura 1-17 Baterie de tip LiPO

1.7 XBee Seria 1

XBee [11] este un modul de radio-frecvență care va fi folosit pentru comunicația între calculator și microcontrolerul de comandă (Arduino).

Figura 1-18 Module de comunicație XBee S1

Acest modul lucrează la frecvența de 2.4GHz și funcționează conform protocolului 802.15. Puterea de transmisie este de 1mW(0dB) ceea ce înseamnă că într-un teren deshis comunicația se poate pierde foarte ușor. Un alt motiv care afectează raza de comunicație este antena integrată de tip PCB. Acesta este și motivul pentru care, pentru a face sistemul să fie controlat de la distanțe mai mari, trebuie schimbate și modulele de comunicație cu XBee-uri profesionale având antene de tip fir cu amplificatoare mult mai puternice.

1.8 Camera RPi

Camera RPi va fi folosită în acest proiect datorită integrării simple în microcontrolerul de Raspberry Pi 3. Acest modul poate fi accesat folosind anumite librării aparținând mai multor limbaje de programare

sau folosind aplicația Video4Linux. Cu această camera se va face capturarea imaginii și transmisia video folosind comenzi simple de la Raspberry Pi. Conectarea camerei cu RPi se face prin intermediul interfeței SCI când minicalculatorul este nealimentat. Aceasta se face pentru a preveni erori sau defecte ulterioare în conexiune sau în dispozitivele folosite.

Figura 1-19 Camera RPi Vision 2

1.9 **Elicele**

Elicele sunt componentele folosite pentru ridicarea quadcopterului de la sol. Ele sunt montate în cuplul motorului și urmăresc rotația lui. Sunt folosite 2 elice care se rotesc în sens orar și 2 elice în sens antiorar. Ele sunt plasate în puncte situate diametral opus pentru a ușura efectul giroscopic al fiecărui motor. În felul acesta se păstrează stabilitatea quadcopterului.

Figura 1-20 Elice pentru quadcopter

CAPITOLUL 2
PRELUCRAREA IMAGINII

2.1 Noțiuni teoretice despre imagine

O imagine este un semnal continuu bidimensional care este definit ca o funcție de două variabile continue: *f(x,y).* Imaginea digitală reprezintă imaginea reală cu o mulțime de valori numerice finite, codificate dupa un anumit sistem. [12]

Din punct de vedere teoretic putem privi imaginea ca având un suport finit cu un număr nelimitat de linii și coloane, însă, în practică, dimensiunea imaginii este întotdeauna limitată, fiind redată în planul discret ca o matrice cu un număr finit de linii și coloane. Acest mod de privire împarte imaginea în unități foarte mici numite pixeli și identificate prin două coordonate plane.

Figura 2-1 Imagini digitale

Fiecare pixel al unei imagini este asociat pe de-o parte cu poziția sa relativă în imagine și deține pe de altă parte valorile caracteristice ale semnalului de lumină emis. Semnalele digitale pot fi clasificate în:

- alb/negru sau binare, cu un bit per pixel
- în nuanța de gri, cu 8 sau mai mulți biți per pixel
- color, cu 8 sau mai mulți biți pentru fiecare culoare de bază

Imaginile color pot fi descrise în trei moduri diferite:

- RGB (Red,Green,Blue) prin culorile de bază
- HSV (Hue,Saturation,Value) prin nuanță, saturație și valoare
- HSL (Hue,Saturation,Lightness) prin nuanță, saturație și luminozitate.

Pentru a calcula intensitatea comună a unei imagini color și a trece în nivele de gri se folosește relația de mai jos:

$$I = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

După tipul datelor în structură, imaginile pot fi împărțite în:

- imagini scalare, în care fiecare componentă este un scalar (imaginile monocrome, în general alb-negru și imaginile cu nivele de gri)

- imagini vectoriale, în care fiecare componentă este un vector de numere (imaginile color în care vectorul are trei elemente ce corespund celor 3 culori de bază). Pentru imaginile multicomponentă, vectorul asociat fiecărui punct din imagine are mai multe elemente.

Pentru a trece de la modelul teoretic al imaginii, semnal bidimensional în spațiu continuu, la reprezentarea unui semnal bidimensional în spațiu discret trebuie aplicate modelului continuu două tipuri de operații: operația de eșantionare și operația de cuantizare. Eșantionarea este consecința limitării rezoluției în domeniul spațial și în domeniul timp, iar cuantizarea este consecința datorată limitării rezoluției intensității, a modului de reprezentare al informației, al numărului de biți necesari pentru reprezentarea unui pixel în imagine.

Așadar, pentru digitalizarea unei imagini, respectiva imagine este eșantionată, fiind reținute numai anumite puncte de interes din imagine și apoi fiecare eșantion este cuantizat în funcție de numărul de biți folosiți.

Având în vedere informațile de mai sus și modelele de reprezentare a imaginilor, se pot efectua diferite operații cu ele: modificarea contrastului, operații de filtare și diferite transformate geometrice.

2.1.1 Stocarea imaginilor

Stocarea imaginii se poate face în memoria de lucru pentru un timp scurt (doar pe durata prelucrări efective) sau stocare de lungă durată în fișiere.

La stocarea de scurtă durată pentru prelucrare rapidă, se vor folosi structuri de date bidimensionale prin intermediul unor variabile. Declararea lor se va face dinamic (declararea statică nu este convenabilă din cauza dimensiunilor mari ale imaginilor). În funcție de limbajul de programare folosit, stocarea se va face corespunzator cu funcții care generează matrici sau vectori de date.

Stocarea de lungă durată presupune scrierea informaților pe fișiere. Pentru realizarea acesteia este necesar ca șirul de octeți reprezentativ al imaginii să conțină informațiile acesteia cum ar fi: dimensiunile, tabelul de culoare,posibilitatea de indexare și metoda de compresare. Dimensiunea imaginii este legată de rezoluția ei, numărul de pixeli și adâncimea culorii (biți de culoare per pixel).

Formaturile imaginii s-au făcut cunoscute după extensia fișierelor ce conțin imaginile stocate. Cele mai folosite formaturi sunt: BMP, TIF, GIF, JPEG, etc. [12]

2.1.2 Tehnici de operare cu imagini

În tehnicile de operare cu imagini, elementele de bază care identifică și care schimbă în funcție de operație sunt: contrastul, histograma și geometria imaginii. [13]

- Geometria imaginii include toate transformările geometrice care i se aplică unei imagini. O transformare geometrică presupune preluarea informației (poziția fiecărui pixel) de la

imaginea sursă și maparea în imaginea destinație. Operații geometrice care pot fi făcute cu imaginile sunt: scalare, rotație, reflexie,translație, etc.

- Contrastul determină gradul de vizibilitate al obiectelor din imagine, reprezentând intervalul dintre zonele întunecate și zonele luminoase. În percepția vizuală a lumii reale, contrastul este determinat de diferența dintre culoare și luminozitatea unui obiect și alte obiecte din interiorul aceluiași câmp vizual.

- Histograma unei imagini reprezintă modul de distribuție al culorilor în imagine. Cu alte cuvinte, este o funcție care asociază fiecărei culori din imagine frecvența de apariție în imagine. Această informație este redată sub forma unui graf unde pe axa abscisei sunt culorile posibile în imagine, iar pe axa ordonatei se vor regăsi frecvențele de apariție ale fiecărei culori de pe axa abscisei.

Figura 2-2 Histograma imaginilor

Există diferite tipuri de operații cu imaginile:

- Operații punctuale, prin care se realizează o corespondența între valoarea veche și noua valoare pentru fiecare pixel.
- Operații locale, prin care valoarea pixelului este obținută din vechea valoare a pixelului și din valorile unor pixeli vecini.
- Operații integrale, în care noua valoare a unui pixel este dependentă de valorile tuturor pixelilor din imagine.

Operații cu imaginile:

- Modificarea contrastului: se consideră o îmbunătățire a unei imagini, nu i se adaugă nicio informație nouă fața de cea curentă, ci doar este prezentat altfel conținutul inițial al acesteia. Modificarea contrastului poate fi liniară sau neliniară, prima transformă liniar contrastul pe porțiuni alese sau dorite de utilizator, iar a doua realizează o modificare neuniformă a

contrastului în jurul unei valori sau pe întregul interval. Folosind metoda neliniară, contrastul se poate mări într-un interval și se poate micșora pe un alt interval folosind același funcție de modificare.

- Operația de filtrare: este considerată și ea o metodă de îmbunătățire a imaginii. În funcție de filtrele aplicate, imaginea capătă noi proprietăți. Filtrarea de netezire, de exemplu, elimină micile diferențe dintre valorile pixelilor aparținând unei aceleiași regiuni. Această metodă este folosită des pentru eliminarea zgomotului alb aditiv gaussian suprapus imaginii.
  Filtrarea de contrastare are ca obiectiv îmbunătățirea perceperii vizuale a contururilor obiectelor. Aceasta se poate realiza prin modificarea valorilor pixelilor aflați de o parte și de alta a unei frontiere comune.

2.2 Detecția de obiect

Detecția de obiect presupune identificarea unui obiect sau a mai multora într-o imagine digitală folosind diferite metode și diferiți algoritmi. Detecția și identificarea obiectelor în mediul robotic ajută în decizii autonome ale sistemelor robotice. Folosindu-se de aceste decizii, sistemele robotice sunt capabile să genereze un raspuns pentru situații anume. Ocolirea obstacolelor, urmărirea obiectelor/persoanelor, supravegherea terenurilor sunt unele aplicații care necesită cunoștințe despre sistemele de detecție și de identificare.

Detecția și identificarea sunt două concepte care diferă unul de celălalt. Ambele sunt legate de verificarea obiectelor pe imagini, dar prima (detecția) doar verifică dacă există sau nu obiect în imagine și daca da, ieșirea ei sunt poziția și o forma apropiată a obiectului. Identificarea presupune în primul rând detecție dar și o verificare mai extinsă pentru a recunoaște tipul sau modelul corect al obiectului. Ieșirea unui sistem de identificare este un raspuns care face diferențierea cu alte modele apropiate ale obectului și cunoaște exact proprietățile lui. Pe de altă parte, ieșirea unui sistem de detecție este doar existența sau nu a unui obiect în imagine și poziția lui.

2.2.1 Detecția mișcării

Cea mai bună metodă pentru detecția mișcării este implementarea unui algoritm care face segmentarea imaginii și folosește tehnica de diferență. Astfel, detecția de mișcare începe prin observarea diferențelor dintre două imagini. Schimbarea în acest caz este tradus ca o mișcare. O metodă de detecție este de a calcula diferența zonelor respective la două imagini succesive sau de a calcula diferența dintre o imagine la momentul de timp t și imaginea de fundal. În ambele cazuri se face extragerea unei imagini din cealaltă și se observă în felul acesta diferența lor.[14]

Figura 2-3 Detecția mișcării folosind diferența dintre imagini

Fiecare metodă are avantajele și dezavantajele ei. În cazul în care diferența de pixeli este făcută cu imaginea din fundal, atunci trebuie luată o secvență de imagini a fundalului astfel încât modelul perfect să includă și alte modele mai deformate (cu luminozitate mai mare sau într-un mediu mai închis). Această metodă este foarte folosită când camera de luat vederi este într-o poziție fixă și mediul supravegheat de aceasta rămâne neschimbat. Introducerea unui obiect în raza de lucru înseamnă schimbarea imaginii de fundal. În acest caz și daca nu există mișcare, algoritmul va semnaliză detecție de mișcare din cauza obiectului rămas oprit (diferența cu fundalul va exista mereu din cauza obiectului).

Această problemă nu va exista când diferența se face cu imaginea anterioară. Teoretic algoritmul va funcționa la fel, doar că imaginea de fundal în acest caz va fi actualizată după fiecare pas. Introducerea unui obiect imobil în urmatoarea parcurgere nu va mai fi semnalizată doarece imaginea anterioară și cea de după vor conține ambele informația despre acel obiect.

În ambele cazuri se va face doar verificarea mișcării (dacă există sau nu mișcare într-o anumită zonă) și nu identificarea obiectului care acționează. Recunoașterea obiectului necesită mai multe date și parametrii și de asemenea o bază de date la intrarea sistemului pentru clasificarea obiectelor.

Așa cum a fost menționat mai sus, detecția de mișcare începe cu segmentarea imaginilor. Din imaginea de fundal, imaginea curentă și cea anterioară se extrag parametrii pentru fiecare pixel. Dacă imaginea de fundal este definită ca B și imaginea capturată în momentul de timp t ca C(t) atunci pentru fiecare pixel se va executa operația matematică de diferență. Daca parametrul extras reprezintă nivelul de gri, atunci pentru fiecare pixel vom avea:

$$P[Z(t)] = P[C(t)] - P[B]$$

unde Z(t) va fi imaginea de diferență. Mai departe această imagine se va binariza pentru a elimina erorile mici apărute din diferite motive. La sfârșitul operației vom avea o imagine care arată daca a existat mișcare și în caz afirmativ în ce zonă s-au produs acele mișcări.

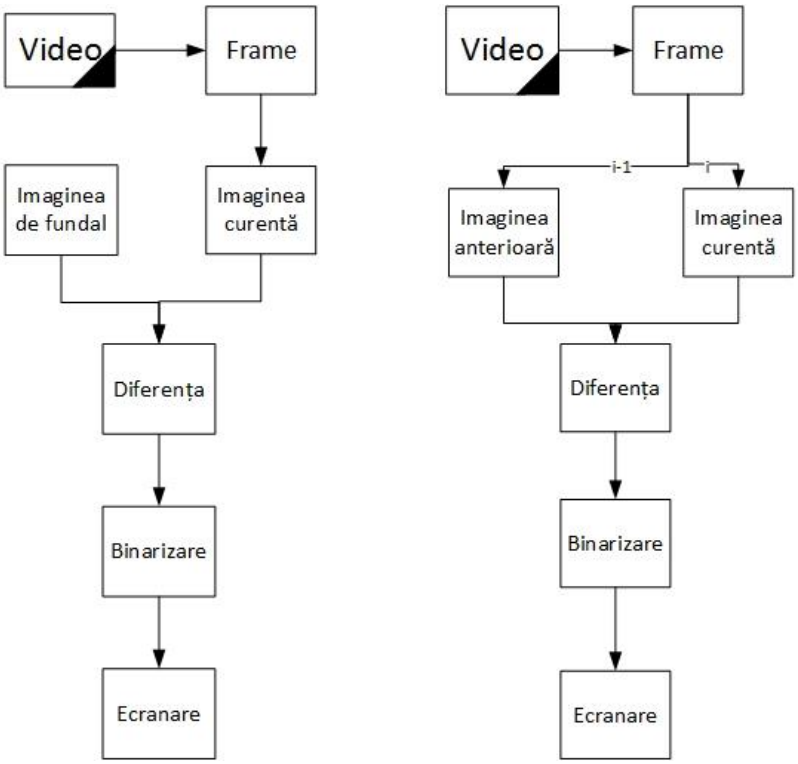


Figura 2-4 Procesul de detecție a imaginii

2.2.2 Detecția de contur

Detecția de contur este numele atribuit metodelor matematice care au ca scop identificarea punctelor într-o imagine digitală la care luminozitatea imaginii se schimbă brusc sau, în mod mai formal, are discontinuități. Acele puncte unde luminozitatea imaginii se schimbă drastic, de obicei sunt organizate într-un set de linii curbate numit margine.

Un model de margine tipic poate să fie, de exemplu, granița dintre un bloc de culoare roșie și un bloc de culoare verde. Însa, o linie poate fi un număr mic de pixeli de o culoare diferită decât fundalul imaginii. Pentru o linie, prin urmare, poate fi o pereche de margini, însemnând un contur pe fiecare parte a liniei.

Majoritatea metodelor folosite pentru detecția de contur urmăresc urmatoarele etape: [15]

1) Detectează pixeli de front: Pixel de front este acel pixel care își schimbă intensitatea brusc în comparație cu pixeli vecini. Dar nu orice pixel care îndeplinește această condiție poate fi pixel de front. Există și cazul când acesta este un pixel de zgomot și din cauza aceasta înainte de detecție se realizează operația de netezire a imaginii.
2) Se elimină pixelii care nu sunt de front. După etapa de netezire, mulți pixeli care se consideră zgomot, nu mai îndeplinesc criteriile necesare pentru a fi pixeli de contur.
3) Se conectează pixelii de front pentru a crea conturul. Pentru această operație se folosesc două metode diferite:

a. Metoda locală, care folosește relațiile între pixelul respectiv și pixelii vecini. În acest caz conturul este creat iterativ, urmărind punctele de interes identificate una după alta.
b. Metoda globală, care folosește informații generale învățate, cum ar fi recunoașterea formelor geometrice ale contururilor sau ecuației matematice.

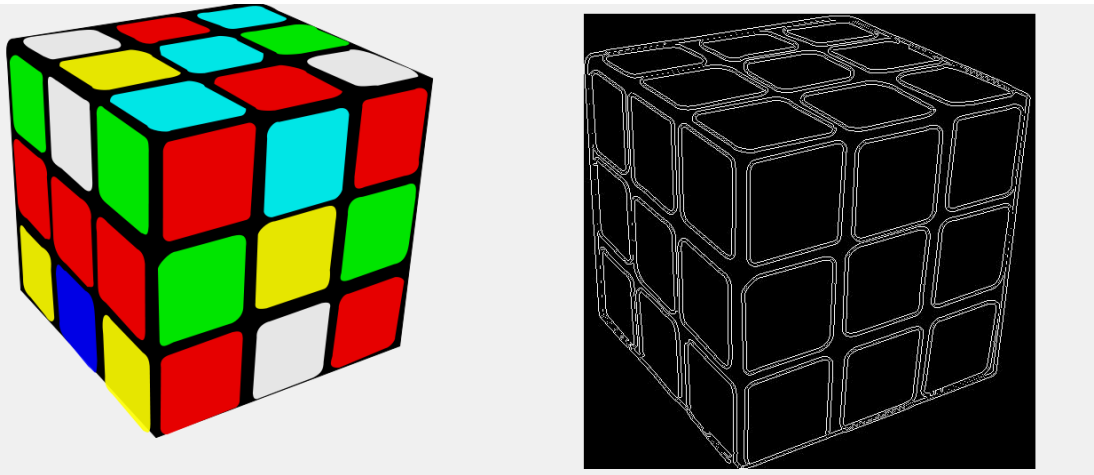


Figura 2-5 Detecția de contur

Majoritatea algoritmilor de detecție a contururilor folosesc prima derivată și a doua derivată din funcția imaginii doarece, într-un punct de margine:

I) Derivata I are un minim sau maxim local în acel punct.
II) Derivata a II-a trece prin 0.

În cazul imaginii digitale care are funcția exprimată în coordonate bidimensionale, prima derivată corespunde Gradientului iar a doua derivată corespunde Laplacianului.

CAPITOLUL 3
TEHNOLOGII SOFTWARE UTILIZATE

3.1 Tehnologii de programare

3.1.1 Arduino IDE

Arduino IDE (Integrated Development Environment)[16] denumit și Arduino Software este aplicația oferită de Arduino pentru programarea plăcilor hardware. Acest mediu de programare conține un editor text, o zonă de notificări, consola și bara de instrumente cu butoane pentru funcții speciale și o serie de meniuri pentru configurare. Programele scrise în Arduino IDE se numesc schițe și prin editorul de text pot fi editate și modificate astfel încât să poată fi folosite pentru diferite aplicații. Programele sunt salvate cu extensia *.ino*. Zona de notificări oferă un feedback în timp real iar afișarea erorilor se face la sfârșit. Butoanele de pe bara de instrumente permit verificarea și încărcarea programului, crearea, deschiderea și salvarea schițelor, și deschiderea monitorului de serială.

Software-ul permite lucrul în paralel cu mai multe schițe, acestea fiind coduri de arduino, fișiere de C(cu extensia *.c*) sau fișiere de C++(cu extensia *.cpp*). Arduino include multe funcții din limbajul C dar și biblioteci specifice doar pentru mediul Arduino. Pentru includerea bibliotecilor software în program se folosește instrucțiunea *#include <Nume.h>*.

Monitorul de serială este folosit pentru a trimite sau a recepționa date din dispozitivul hardware. Pentru realizarea comunicației seriale, se folosește funcția *Serial.begin(boud-rate).* La deschiderea comunicației se pot transmite date introducând textul pe monitor sau putem să citim datele primite. Pentru operațiile menționate mai sus folosim funcțiile *Serial.println(), Serial,write()* sau *Serial.read().*

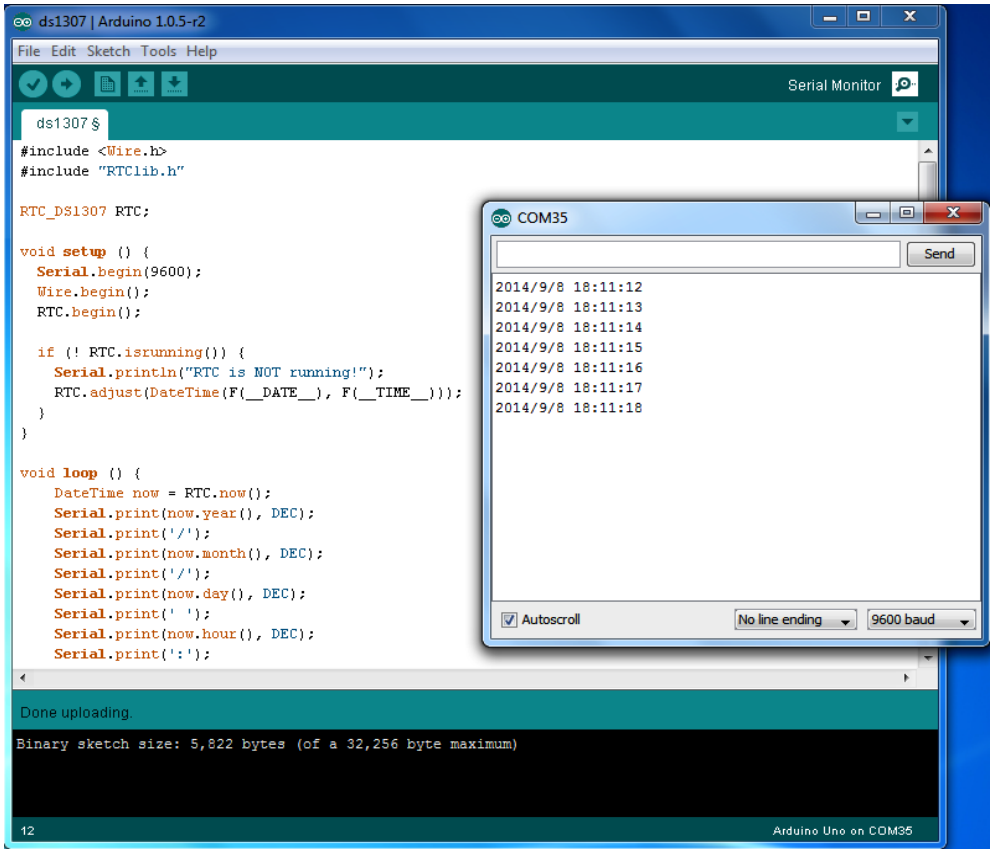


Figura 3-1 Arduino IDE

La încărcare se alege placa de dezvoltare și portul la care aceasta este conectată. Dupa accea, se apasă butonul de *Upload* și programul se va încărca pe microcontroler. La această operație este folosit *Arduino bootloader,* un program încărcat în microcontrolerul de pe placă. Ledurile de Tx și Rx vor semnaliza sfârșitul procesului, iar în caz de erori, acestea se vor afișa în fereastra de notificări din interfața grafică Arduino IDE.

La realimentarea plăcii programul încărcat anterior nu se pierde, acesta fiind salvat în memoria microcontrolerului. Este rolul bootloader-ului să păstreze programul și la alimentare, schița cea mai recent încărcată va fi rulată. Dezavantajul folosirii unui bootloader este faptul că acesta introduce o întârziere în rularea programului, față de folosirea unui programator ce utilizează întreaga memorie flash a cipului.

3.1.2 Python

Python este un limbaj de programare dinamic foarte popular. El oferă posibilitatea de a programa structurat dar și orientat pe obiect. Este un limbaj de scripting, deci este interpretat și nu compilat, economisând astfel mult timp în procesul de dezvoltare și debugging.

Python este folosit foarte mult pentru programarea aplicațiilor web. Mulți dezvoltatori de software lucrează la implementarea limbajului Cpython scris în C. Multe module și biblioteci de Python pot fi de asemenea scrise în *C*, compilate și importate în Python pentru a mări viteza de procesare.

Ca și limbajul de programare C, Python folosește diferite tipuri de date și variabile, folosește structuri condiționale, repetitive și funcții. Funcțiile sunt definite în felul urmator:

```
def Nume(argument):
```

În Python se pot include funcții, variabile și clase definite în alte fișiere. Un astfel de fișier poartă numele de modul. Modulul inclus poate fi prezentat cu alt nume pentru a evita coliziuni de denumiri:

```
import random as rand
```

Folosind limbajul Python se pot realiza ușor operații cu fișiere de tip text și nu numai. Importând diferite biblioteci disponibile gratuit, Python poate să lucreze cu rețeaua folosind protocoale de comunicare prin internet (HTTP, Telnet) sau să reprezinte baza unor platforme de complexitate ridicată pentru programarea Web.

Python oferă și un instrument bun ca editor de text numit PYCharm. Acesta este o platformă de tip IDE folosită pentru programarea avansată cu posibilitatea de a depana, testa și implementa aplicații de la distanță, cu mașini virtuale sau folosind terminalul SSH.

Pentru instalarea limbajului Python pe RPi se va folosi comanda urmatoare:

```
sudo apt-get install python
```

3.1.3 HTML

HTML (HyperText Markup Language)[17] este un limbaj de marcare utilizat pentru dezvoltarea paginilor web care pot fi afișate într-un browser. Scopul HTML-ului este prezentarea informațiilor și documentelor sub formă de text, paragrafe, imagini, tabele, etc. Utilizând un software de redare specializat, numit agent utilizator HTML, se realizează prezentarea documentelor pe o singură pagină. HTML poate fi generat direct folosind tehnologii de codare din partea serverelor cum ar fi PHP sau JSP și este folosit de multe aplicații ca sistem de gestionare a conținutului. [17]

Paginile HTML pot fi citite și editate utilizând orice editor simplu de text și sunt distinse prin extensia lor de tip *.html* sau *.htm*. Paginile sunt formate din etichete, în mare parte din perechi de etichete structurate astfel încât una realizează deschiderea unui bloc de instrucțiuni și alta închiderea lui. Aceste etichete sunt interpretate mai departe de navigatorul web și afișează pe ecran rezultatul lor, ce poate fi accesat prin interfața grafică. Pentru o gamă variată de aplicații în diverse domenii, pagina principală folosită este index.html.

Un document HTML conține următoarele câmpuri importante:

- versiunea documentului
- zona head care este definită de etichetele <head> </head>
- zona corpului care este definită de etichetele <body> </body>

Toate paginile HTML folosesc etichetele <html> </html> la începutul și la sfarșitul textului. În interior se folosesc etichete din categoriile enumerate mai sus. La fel ca la majoritatea limbajelor de programare, și la HTML pot fi introduse comentarii, care bineînțeles nu vor fi afișate de către browser. Comentariile au următoarea sintaxă:

<!—Comentariu care nu se afișează -- >

Având în vedere cele menționate anterior, codul utlizat pentru realizarea unei pagini web cu titlul „Quadcopter Streaming”, ce afișează textul „Licența_2016” este:

```
<html>

        <head>
                <title>Quadcopter Streaming</title>
        </head>
        <body>
                        <p>Licența_2016</p>
        </body>

</html>
```

Pentru optimizarea paginii se folosesc elemente de marcare structurală, pentru prezentare, pentru hiperlink, pentru elemente speciale sau tag-uri specifice. Astfel vom obține o pagină cu un aspect vizual îmbunătățit, dar și cu o organizare mai bine structurată, fiind ușor de accesat și citit de către majoritatea utilizatorilor.

<h: element de marcare> Tiltu </h>
<body: bgcolor,background,etc.. > Conținut </body> //Schimbă structura elementului
<body topmargin="0" leftmargin="0"> </body> //Modifică aranjarea în pagină

3.2 Biblioteci și aplicații

3.2.1 Putty

Putty este un program disponibil gratuit care oferă posibilitatea de a simula un terminal fiind un client pentru SSH, Telnet sau chiar și pentru consola serială. [18]

Putty numeră câteva caracteristici importante:

- Conexiuni de tip Telnet, SSH, Raw sau Serial
- Stochează adresele favorite pentru a le accesa ulterior
- Control asupra portului X11 Forward (folosit pentru accesarea interfețelor grafice)
- Accesarea unui server de oriunde și simularea comportamentului unui client.

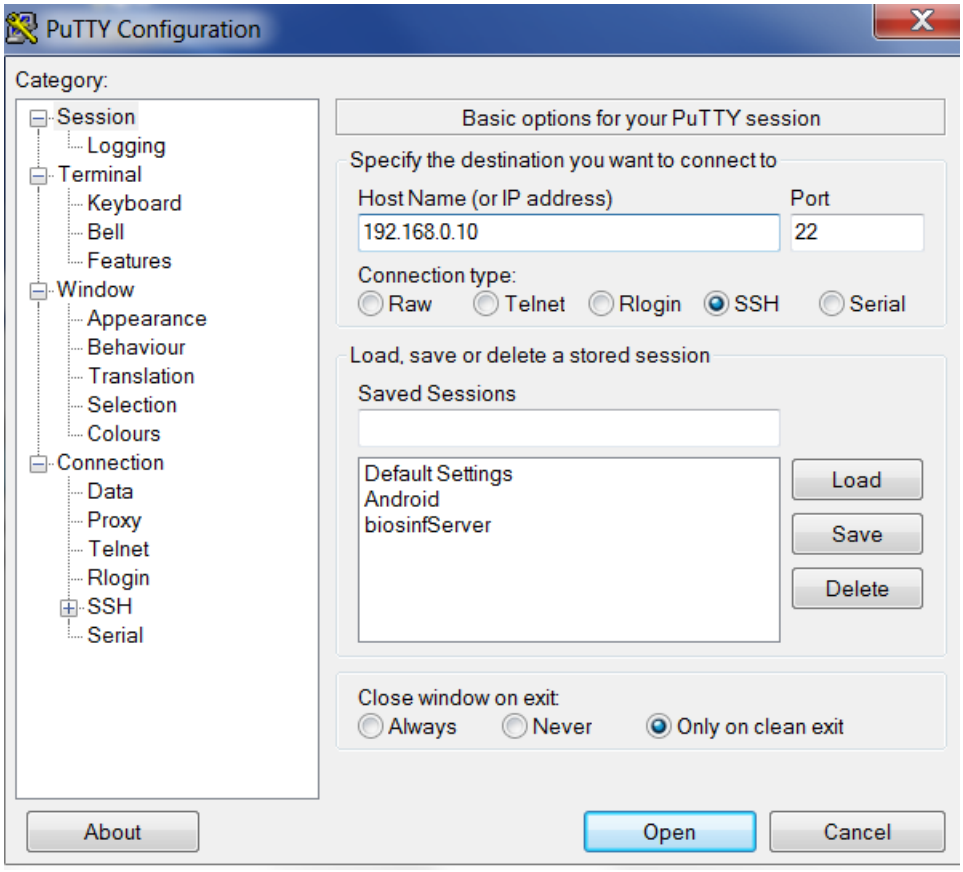


Figura 3-2 Putty

Pentru a accesa resursele din RPi, se va introduce adresa de IP a interfeței dispozitivului și prin portul număr 22 se va realiza conexiunea prin SSH. Adresa de IP se va afla din serverul DHCP care rulează în rețea. Odată introduse aceste date, împreuna cu numele utilizatorului și IP-ul pot fi salvate și

accesate mai rapid la următoarea apelare. La accesare se va introduce parola corespunzătoare utilizatorului.

3.2.2 Flask

Flask este un mediu de lucru în Python bazat pe Werkzeug(bilbiotecă de utlizare a WSGI – Web Server Gateway Interface - pentru Python) și Jinja2 (designer al cadrelor web implementat pentru Python). [19]

Flask este folosit pentru a ușura interfața între server și client și folosește seturi de instrumente și biblioteci pentru a crea o aplicație capabilă să transmită și să raspundă într-o rețea.

- Werkzeug WSGI[20]: este utilizat în crearea aplicaților web folosind limbajul de programare Python și reprezintă o interfață simplă între server și aplicație. Lucrul cu cadrul de web în Python a fost considerat destul de greu și din cauza aceasta WSGI a fost folosit ca o legatură de bază între server și aplicația portabilă. WSGI folosește două parți: Serverul(care poate fi de multe ori un server web cum ar fi Apache) și aplicația (scriptul scris în Python). Pentru a procesa o cerere WSGI, serverul execută aplicația furnizând informația necesară, iar o funcție de apel invers este utilizată pentru aplicație. Aplicația procesează cererea și returnează la server răspunsul folosind funcția de apel invers care a fost furnizată înainte.

```
pip install Werkzeug  //Adaugă componentele în Python
```

- Jinja2[21]: Este un instrument folosit pentru design-ul paginilor web. Programarea paginii web se face în HTML iar Jinja2 ajută în realizarea legăturii între pagina web și aplicație. Jinja permite proiectantului să apeleze funcții cu argumente asupra obiectelor. Acest instrument este considerat șablonul implicit folosit în aplicațiile create cu Flask.

```
pip install Jinja2  //Adaugă componentele în Python
```

În mod implicit, Flask nu include o etapă de acces la baza de date, dar suportă extensii pentru a adăuga o funcționalitate în aplicația dorită. Multe extensii asigură încărcarea, integrarea și validarea bazei de date și diverse tehnologi de autentificare deschise. Prin convenție, șabloanele și fișierele necesare sunt stocate în subdirectoarele aplicației rulate în Python, cu modelele și numelele respective.

În realizarea transmisiei video este nevoie de o funcție generatoare care transmite în continuare frame-urile pe pagina web. Flask oferă suport pentru aceasta prin utilizarea funcțiilor generatoare. Aceaste funcții pot fi întrerupte și reluate, continuând să execute subrutinele lor de dinainte de întrerupere. O funcție de acest gen va fi folosită pentru generarea frame-urilor în continuare. Cea mai bună metodă de transmisie video este numită Motion JPEG și transmite în mod continuu o secvență de imagini independente JPEG.

```
def gen(Camera):
        while True:
```

```
                frame = camera.get_frame()
                yield Image/jpeg
```

Funcția de mai sus va genera în continuu o imagine. Pentru actualizarea imaginii cu cea curentă se va utiliza raspunsul primit de la funcția de generarea a frame-urilor care va schimba imaginea cu cea nouă. Deci funcția returnează controlul execuției până în punctul în care a fost numită. De aceea, *yield* implică faptul că transferul este temporar și voluntar și funcția noastră îl va relua din nou pe viitor (astfel, se crează o buclă ce poate fi întreruptă și reluată fără a strica sau opri subrutina principală de program).

Așa cum se observă, pentru realizarea transmisiei video pe rețea (M-JPEG over HTTP) [22] se folosește Flask, care se ocupă cu deschiderea comunicației și controlarea cererilor de tip GET_REQUEST. La răspunsul acestor cereri se vor încărca pe pagina web frame-urile cu conținutul de tip media (image).

```
return Response(gen(Camera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')
```

După cum se observă, funcția gen(Camera) se ocupă cu captarea frame-urilor, iar ca răspuns primește în continuare un obiect de tip media care se va afișa. Ruta de afișare va fi definită la scriptului programului scris în HTML și va genera o ieșire pe pagina web.

3.2.3 Open CV

Open CV [23] este o bibliotecă cu multe funcții de programare care ajută în prelucrarea imaginilor în timp real. Este folosit foarte mult în calculatoare și servere care se ocupă cu calculul imaginilor vizuale în timp real. Open CV include funcții și implementări mai bune în procesele de recunoaștere și a introdus un concept nou în interfața de programare. Limbajul principal folosit este C++, dar există și versiuni de legătură pentru Python, Java sau MatLab. Toate dezvoltările curente și algoritmii se fac pentru crearea noilor interfețe cu sistemele de operare și pentru minimizarea resurselor folosite în timp real.

Open CV rulează pe diferite platforme printre care și Windows, Linux, Android, etc. În procesul de compilare OpenCV folosește Cmake, o platformă care este concepută pentru realizarea cadrului aplicațiilor care depind de mai multe biblioteci. Platforma are dependențe minime care necesită doar un compilator C++ în propriul sistem de dezvoltare.

3.3 Sistemul de operare Linux

Linux este un sistem de operare foarte des folosit în lumea programatorilor și este dezvoltat și distribuit în continuare ca un software gratis și open-source. La început a fost dezvoltat pentru calculatoarele

generale cu arhitectură bazată pe Intel X86, dar în timp a devenit pentru platformele hardware o opțiune utilizată mai frecvent în comparație cu orice alt sistem de operare. [24]

Linux, în forma sa originală, este cel mai important sistem de operare folosit pe servere și supercalculatoare, fiind folosit foarte puțin la calculatoarele de birou. Linux rulează, de asemenea, pe sisteme integrate, care sunt dispozitive al căror sistem de operare este extrem de ușor adaptabil la sistem. În cazul nostru, sistemul de operare linux este folosit la minicalculatorul RPi 3.

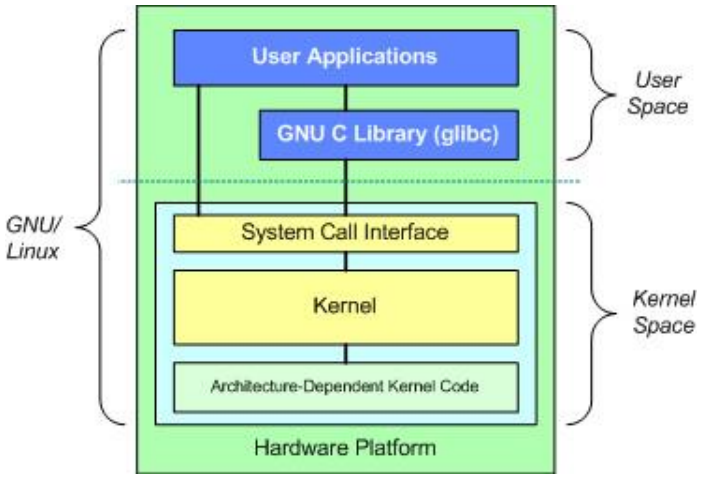


Figura 3-3 Arhitectura de funcționare a sistemului de operare Linux

Un sistem bazat pe Linux este un sistem de operare modular, fapt ce rezultă din proiectarea și din principiile stabilite în Unix. Un astfel de sistem utilizează un nucleu, nucleul Linux, care se ocupă de controlul proceselor, crearea rețelelor, accesul la periferice și sistemul de fișiere. Driverele dispozitivelor pot fi integrate sau adăugate ca module și se încarcă în timp ce sistemul funcționează. Diferite proiecte care crează o interfață de legatură cu sistemul Kernel furnizează o funcționalitate de nivel superior a sistemului. GNU este o parte importantă a sistemului de operare pentru că oferă cele mai comune biblioteci de C. Interfața grafică cu utilizatorul (GUI) este construit pe baza implementării sistemului X Windows.

3.4 Wi-Fi

WiFi este tehnologia care permite dispozitivelor electronice să se conecteze la o rețea locală, fără a avea nevoie de o conexiune cu fir. O rețea WLAN este de obicei protejată cu o parolă și poate fi accesată când dispozitivul respectiv este inclus în raza de funcționare și oferă permisune de access prin introducerea unei parole. Standardul folosit de această tehnologie este IEEE 802.11 și funcționează la 2.4GHz. Pentru protejarea informației în rețea, se folosesc metode de criptare cum ar fi WEP, WPA și WPA2. [25]

Rata de transfer la această tehnologie s-a schimbat în timp, urmărind dezvoltarea continuă atât din punct de vedere al distanței de transmisie cât și al vitezei ajungând astfel ca, în momentul de față, tehnologia 802.11n să funcționeze la o viteză de 450Mbps. Această viteză poate varia din cauza mediului de funcționare sau fenomenelor de natură electrică sau electrostatică apărute în zonă.

CAPITOLUL 4 ASAMBLAREA ȘI TESTAREA QUADCOPTERULUI

Etapa finală pentru pregătirea de zbor a quadcopterului include asamblarea pieselor, calibrarea ESC-urilor și motoarelor, procedura de armare și testarea fără elice. Toți acești pași se vor executa folosind ca microcontroler de comanda Arduino Uno. Programele sunt făcute în limbajul de programare Arduino IDE.

În etapa de testare în teren, arhitectura va ramane aceeași, dar microcontrolerul de comanda va fi înlocuit cu Raspberry Pi 3 datorită conexiunii cu Wi-FI și compabilitatea mai ușoară cu camera de luat vederi. Programul pentru controlul dronei va fi scris în C și rulat pe RPi 3. Comenziile în acest caz vor fi trimise conectând microcontrolerul prin SSH.

4.1 Calibrarea ESC-urilor

Primul pas în pregătirea pentru zbor, este configurarea și calibrarea ESC-urilor. Această etapă presupune sincronizarea celor 4 ESC-uri și cunoașterea semnalului PWM trimis de către acestea (valoarea maximă și valoarea minimă)

Calibrarea lor a fost facută folosind programul din Anexa 1[Calibare]. Fiecare ESC se conectează la Arduino și primește comenzile corespunzătoare. La final, toate cele 4 sunt conectate direct la Arduino (fără intermedierea microcontrolerului de zbor HoverFly Open) pentru a fi sincronizate.

Procedura de calibrare:

- Se conectează componentele necesare conform schemei de mai jos.
- Se rulează programul pe Arduino
- Se trece la starea (a) unde semnalul PWM are valoarea maximă
- Se conectează bateria și se urmăresc semnalele audio de la ESC.
  - ESC intră în modul de programare.
- Se trece la starea (d) unde semnalul PWM are valoarea minimă.
- Se deconectează bateria.

Această procedură se repetă pentru toate cele 4 ESC-uri astfel încât piesele sa fie sincronizate între ele.

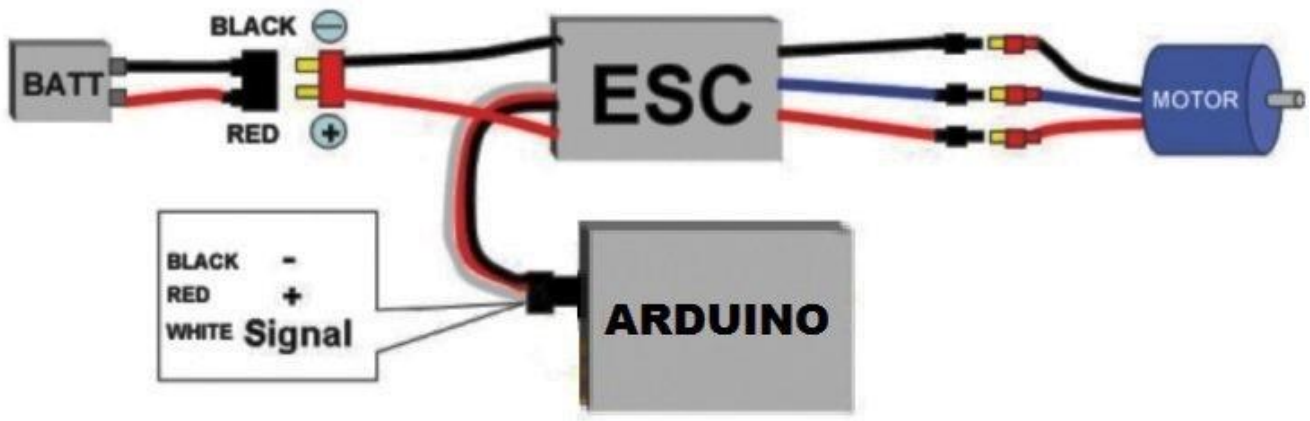


Figura 4-1 Schema de calibrare a ESC-urilor

În timpul acestei proceduri, singurul mod prin care putem să întelegem comenzile și reacția ESC-urilor, este auzind semnalele sonore pe care acestea emit. Fiecare tip de ESC conține proprile semnale și în acest caz este necesară cunoașterea lor.

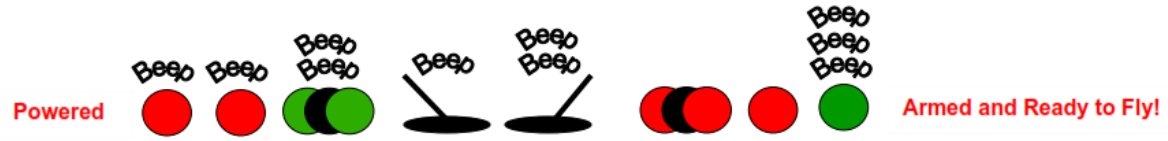
Obervație: Este foarte important și de menționat faptul că toate testele în continuare legate de buna-funcționare a quadcopterului în teren închis să fie făcute fără elice. Aceasta se face pentru motive de siguranță.

4.2 Armarea quadcopterului

Aceasta este prima etapă în pregatirea quadcopterului pentru zbor. Este considerată o etapă de siguranță și de protecție fără ajutorul caruia comenzile nu pot fi recunoscute și nici executate de către quadcopter.

Cum a fost precizat și mai sus, este foarte important ca pentru prima dată, elicele sa fie scoase din motive de siguranță.

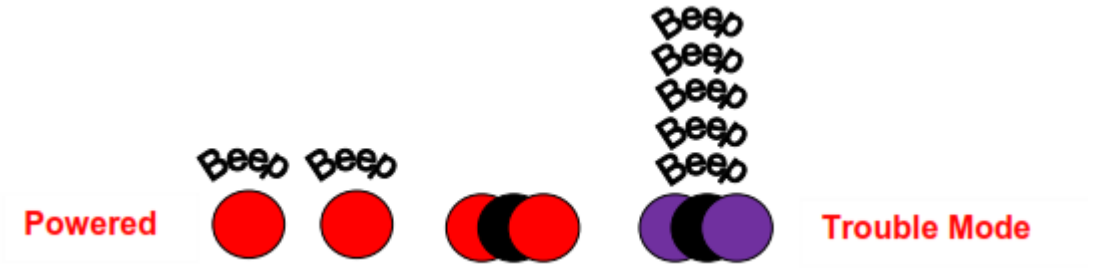
Armarea însăși presupune urmărirea procedurii fixe înainte de fiecare zbor. Urmatorii pașii trebuie executați pe rând:



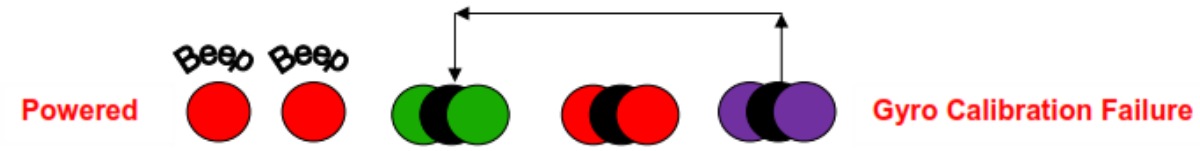
- Stick-ul de accelerație trebuie mutat la valoarea lui minimă
- Se alimentează bateria
- Se așteaptă semnalizarea ESC-urilor iar microcontrolerul de zbor HoverFly va semnala prin aprinderea ledului în culoarea verde.
- Pentru armare, stick-ul de accelerație trebuie mutat în stânga-jos. Aceasta corespunde unei stări unde canalul de Throttle va avea valoarea minimă și cel de Rudder valoarea maximă. Hoverfly va semnala cu un semnal sonor.
- Stick-ul de accelerație trebuie mutat în dreapta-jos, loc ce corespunde stării în care Throttle și Rudder au valoarea minimă.
- Dupa executarea ultimului pas, HoverFly va semnala prin aprinderea LED-ului în culoarea roșie că va face calibrarea senzorilor interni folosiți mai departe. În acest moment, acesta trebuie să rămână nemișcat pentru a face o calibrare corectă.
- La final LED-ul se va aprinde din nou verde și va ramane în această culoare pe durata zborului.

În condițiile în care apar schimbări legate de autonomia quadcopterului, se va schimba și culoarea LED-ului conform cazurilor de mai jos: [3]

A. **Stare nefuncțională:** În acest caz, HoverFly nu recunoaşte unul sau mai multe din cele 5 canale de comandă. Trebuie verificate legaturile și refăcută procedura de armare.



B. **Eșecul de calibrare:** Calibrarea senzorilor externi nu s-a făcut bine.



C. **Menținerea altitudinii în timpul armării:** Această stare nu permite ridicarea quadcopterului de la sol, așa că trebuie repetată procedura de armare.

Ultima etapă importantă este dezarmarea quadcopterului. Ea presupune:

- Mutarea stick-ului de accelerație la valoarea minimă.
- Mutarea stick-ului de accelerație stanga-jos până se aude un singur semnal sonor.
- Mutarea stick-ului de accelerație dreapta-jos până se aude și al doilea semnal sonor.
- Microcontrolerul HoverFly Open este dezarmat.

Procedura de armare și dezarmare este facută folosind programul din Anexa 1 [Armare]. Conform algoritmului gândit, s-au generat semnalele necesare cu ajutorul microcontrolerului de comandă Arduino. Folosind comunicația cu Xbee și cu ajutorul tastaturii, am simulat un controler radio pentru a realiza armarea și dezarmarea quadcopterului.

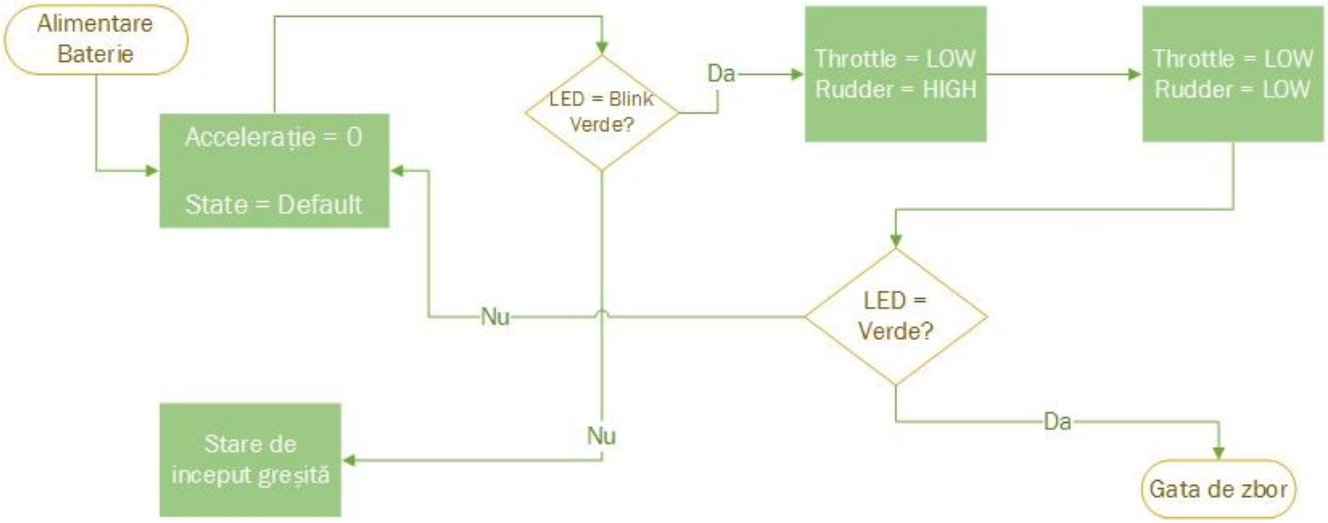


Figura 4-2 Procedura de armare

Pe Arduino este încărcat programul și schimbarea stării se face cu ajutorul tastelor. Alimentarea bateriei pune în funcțiune întregul sistem. Apoi sistemul trece în starea de accelerație minimă și mai departe LED-ul din HoverFly se aprinde intermitent verde. Se execută pe rând stările Throttle_Low_Rudder_High și Throttle_Low_Rudder_Low în care se generează semnalele necesare pentru procedura de armare prin apăsarea tastelor ”d” și ”a”.

După calibrarea senzorilor externi, HoverFly va fi gata de zbor și aprinderea LED-ului cu culoarea verde semnalează aceasta.

4.3 Controlul quadcopterului cu un software

După ce etapele de mai sus au fost parcurse cu succes, quadcopterul este gata să zboare. Pentru controlul lui se va folosi un program pe microcontrolul de comandă care va transmite în continuare semnalele necesare pentru zbor [26]. Programul este gândit să funcționeze cu stări în care putem să schimbăm valoriile semnalelor transmise de către microcontrolerul de zbor.

Așa cum a fost precizat, controlul se face folosind 5 canale prin care se transmit semnale de tip PWM de la microcontrolerul de comandă la cel de zbor. Pentru transmiterea comenzilor de la calculator la microcontrolerul de comandă sunt folosite 2 module XBee care comunică între ele.

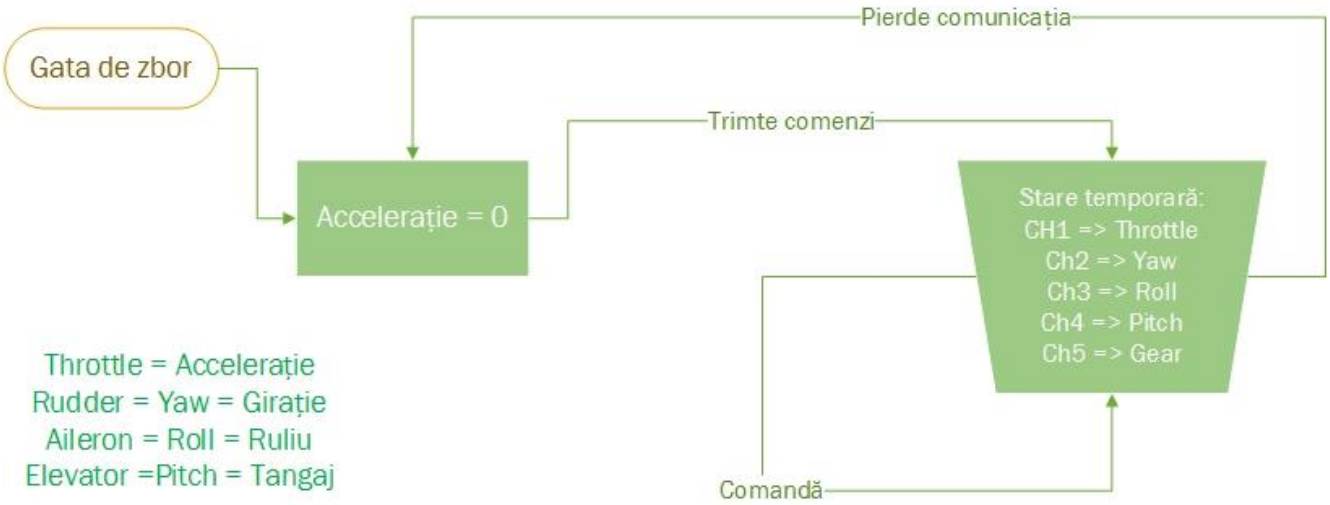


Figura 4-3 Controlul quadcopterului

Cu ajutorul tastelor, putem să trimitem comenzii de către microcontrolerul de comandă și să schimbăm în continuare starea sistemului. O stare presupune generarea tuturor semnalelor necesare pentru zbor și în funcție de factorul de umplere al fiecărui semnal PWM se realizează mișcarea quadcopterului. Fiecare canal este comandat cu ajutorul a 2 taste (una pentru incrementare și alta pentru decrementare). Valoarea minimă este definită pentru factorul de umplere de 5% iar valoarea maximă pentru 10%. Aceasta înseamnă că semnalul este variabil în intervalul 1-2ms și creșterea se face periodic cu un factor de modificare 20-50ns.

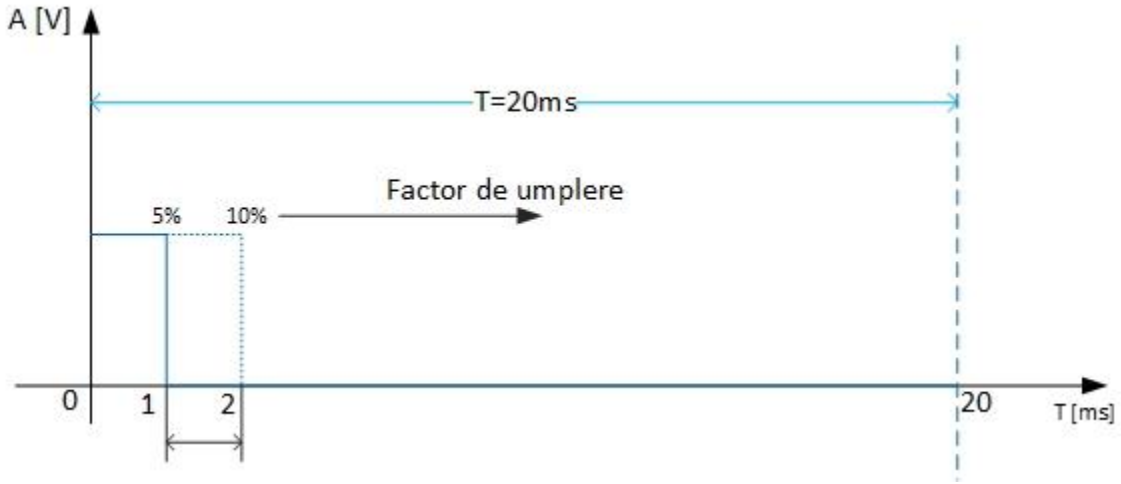


Figura 4-4 Semnalul de control pentru un canal

Canalul de Gear este folosit pentru a activa menținerea altitudinii și de aceea el este un switch între două stări on/off. Pentru starea ”on” va fi folosit factorul de umplere 10% iar pentru starea ”off” cel cu 5%.

4.4 Teste în teren închis fără elice

Primele experimente legate de acest quadcopter au fost desfășurate în interiorul laboratorului. Prima parte a constat în parcurgerea tuturor pașilor de mai sus și schimbarea stărilor cu elicele scoase. Quadcopterul este asamblat și este pus în funcțiune prin alimentarea bateriei. Microcontrolerul de comandă Arduino execută programul încărcat pe el și așteaptă în continuare comenzi. Comunicația între calculator și Arduino a fost realizată cu cablu USB (ceea ce în momentul de fața nu este o problemă).

Se realizează armarea quadcopterului urmărind procedura necesară descrisă în capitolul 4.2 . De fiecare dată când condițiile legate de generarea semnalelor sunt îndeplinite, această procedură se execută cu succes. În felul acesta microcontrolerul de zbor recunoaște semnalele de comandă și este în așteptare. Se fac diferite experimente ajustând viteza de mișcare și se observă comportamentul motoarelor, cum se modifică viteza lor în funcție de semnalele primite.

O primă problemă observată în testele de interior a fost nesicronizarea motoarelor. În urma mai multor teste s-a observat că un motor începea mișcarea mult mai târziu decât celelalte. Primul pas a fost verificarea semnalelor și perioadei/factorului de umplere ale acestora. Am înlocuit acest motor cu unul de rezervă și comportamentul era la fel, mereu era întârziat. Am reconectat motorul cu un alt ESC și am constatat că întârzierea era din cauza necalibrării ESC-ului schimbat înainte.

Am repetat procedura de calibrarea e ESC-urilor unde toate cele 4 ESC-urile folosite au fost sincronizate să recunoască aceleași semnale primite. După ce acest pas a fost realizat cu succes, problema cu motoarele a fost rezolvată.

Pentru a verifica capacitatea dronei de a se ridica de la sol, am creat un sistem cu scopul de a menține drona relativ fixă și a nu îi permite să își crească altitudinea prea mult. Sistemul consta în patru coarde legate în patru puncte diametral opuse de pe trenul de aterizare al dronei într-o parte și la celălalt capat fiind obiecte fixe din cameră. În urma experimentului, am putut să verificăm funționarea corectă a quadcopterului și modificând valoarea semnalului de accelerație (Throttle) am observat intenția sistemului de a se ridica de pe pământ.

4.4.1 Controlul cu Arduino

Testele în interor au avut ca microcontroler de comandă Arduino Uno. Această placă de dezvoltare oferă ușurință în generarea semnalelor de tip PWM. În proiectul nostru, este foarte important să simulăm un controler radio, de accea vom folosi semnalele PWM pentru a construi semnalul de tip PPM necesar pentru controlul quadcopterului.

Pentru generarea semnalului de tip PWM, vom folosi funcția următoare:

```
void generateDefaultPWM(int pin){
 digitalWrite(pin,HIGH);
 delayMicroseconds(H);          //Timpul în care pinul generează semnal cu A=5V
 digitalWrite(pin,LOW);
 delayMicroseconds(L);    }    //Timpul în care pinul generează semnal cu A=0V
```

unde *H+L = Perioada* și *H/Perioadă=Factor de umplere*

În proiectul nostru, pentru a controla quadcopterul, trebuie să trimitem în același timp informația la 5 canale. Pentru a realiza aceasta, avem nevoie de 5 pini digitali de la acest microcontroler. Aceasta presupune folosirea semnalelor de tip PPM ca o singură entitate pe care o s-o considerăm semnal de stare. Aceste semnale vor avea proprietățiile semnalului radio transmis de la controlerul radio.

```
case STATE1:
   generateDutyCyclePWM(throttlePin);          //Generarea semnalului pentru canalul Throttle
   generateDutyCyclePWM(rudderPin);            // Generarea semnalului pentru canalul Rudder
   generateDutyCyclePWM(aileronPin);           // Generarea semnalului pentru canalul Aileron
   generateDutyCyclePWM(elevatorPin);          // Generarea semnalului pentru canalul Elevator
   generateDutyCyclePWM(gearPin);              // Generarea semnalului pentru canalul Gear
   delayMicroseconds(10000);                   //Timpul de așteptare de 10ms
   break;
```

La execuția acestei funcții vom avea la final un semnal cu perioada de 20ms care în primele 10ms va conține 5 semnale cu durată de 2ms corespunzătoare fiecarui canal și 10 ms la sfârșit pentru completarea semnalului PPM.

Fiind o comunicație continuă, aceste funcții vor fi introduse în *void loop() { }* pentru a transmite în continuare semnalele necesare.

Pentru schimbarea stării vom comunica cu Arduino folosind module de comunicație XBee și în funcție de comandă, se va trece în altă stare:

```
if (Serial.available() > 0){                    //Verifică daca s-a primit comanda
        command=Serial.read();                  //Citește valoare comenzii
          switch (command){                     //În funcție de tastă, se ia decizia
                  case 97: //a                  //
                           state = STATE_NEW;   //Se execută starea nouă pentru care se va transmite un
                   break;                       //          alt semnal de tip PPM
```

Programul pentru controlul quadcopterului este inclus în Anexa 1[Control] . Cei 5 pini folosiți au rolul canalelor de transmisie și sunt legați corespunzător cu intrările microcontrolerului de zbor HoverFly Open. La fiecare din ei se generează semnalul PWM folosit mai departe de microcontrolerul de zbor și de ESC-uri.

- Se introduc stările necare pentru armare și o stare temporară care se va schimba în funcție de comenzile primite. Acestor stări le corespund unor anumite configurații de la telecomandă.

De exemplu: THROTTLE_HIGH_RUDDER_LOW simulează telecomanda când stickul de accelerație este în poziția de sus-stânga.

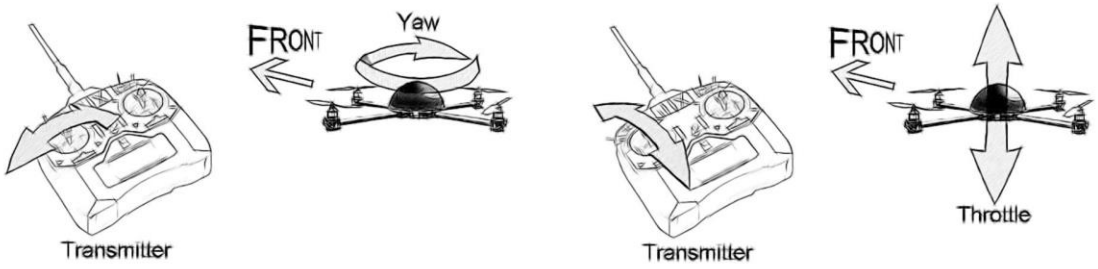


Figura 4-5 Simularea unei telecomenzi

- Se creează funcțiile care vor genera în continuare semnalele de tip PWM. De exemplu funcția de mai jos creează un semnal care va fi generat pe pinul ales și care va schimba factorul de umplere în funcție de variabila x.

```
void generateModifyPWM(int pin){
                digitalWrite(pin,HIGH);
                 delayMicroseconds(1000+x);
                digitalWrite(pin,LOW);
                delayMicroseconds(1000-x);      }
```

- Se deschide comunicația cu calculatorul folosind comunicația Serială. Pinii Tx și Rx vor fi folosiți pentru a trimite și a recepționa informația necesară.

  MySerial.begin(9600); este funcția folosită pentru deschiderea comunicației cu calculatorul având ca parametru Baud-Rate=9600b/s. Comunicația serială în acest caz folosește pinii 2 și 4 de la Arduino și pentru definirea acesteia este folosită librăria *<SoftwareSerial.h>* iar declararea pinilor pentru comunicația serială se face cu *SoftwareSerial mySerial(2, 4); // RX, TX*

- În funcția *void loop ()* este introdusă rutina de control a quadcopterului. Toate stările necesare pentru funcționarea corectă sunt incluse. Prima stare STATE_THROTTLE_LOW este necesară pentru rutina de siguranță. Daca se pierde comunicația, programul intră în această stare. Stările THROTTLE_LOW_RUDDER_HIGH și THROTTLE_LOW RUDDER_LOW sunt necesare pentru rutina de armare. Întotdeauna la început este necesară parcurgerea acestei rutine astfel încât quadcopterul să fie gata de zbor.
  Starea STATE_ALL_IDLE este folosită când armarea quadcopterului nu este realizată cu succes. În acest caz, trebuie să reluăm procedura de armare și este foarte important să începem de la starea unde toate stickurile sunt nemișcate. Mai departe, va urma procedura normală de armare.
  Starea de STATE_MODIFY_THROTTLE este starea temporară în care quadcopterul modifică accelerația în funcție de comenzile primite.
- Funcția *if (mySerial.available() > 0)* verifică dacă pe comunicația serială s-a primit informație sau comenzi folositoare. Dacă da, atunci se va citi această informație și se va folosi

mai departe în funcția *switch (command)*. Această funcție va schimba starea care va depinde de comenzile primite, de exemplu :

```
case ASCII_CODE :                    //Tasta apăsată în codul ASCII
        state = STATE_TEMORARY;      //Starea care corespunde comenzilor
break;
```

- Pentru a urmări fiecare intrucțiune și impactul ei la schimbarea stărilor, se va folosi funcția

```
mySerial.println(state);          //Afișează starea curentă
mySerial.println(x);              //Afișează valoare curentă a pasului
mySerial.println('\n' );
```

4.4.2 Comunicația cu Xbee

Următorul pas în proiect este trecerea de la comunicația cu cablu USB la o comunicație fără fir. Această trecere ajută în autonomia sistemului și controlul de la distanță al acestuia.

Modulele folosite sunt XBee Seria 1[11] , module de comunicație radio care creează o rețea între cele două module și realizează comunicația între stația de transmisie și cea de recepție. Configurațiile necesare la ambele module sunt făcute folosind aplicația X-CTU și CoolTherm. Ambele aplicații oferă posibilitatea de a configura module XBee și verifica conexiuniile între ele. Tot așa, aplicațiile poate fi folosite pentru a trimite comenzi în rețea creată între cele 2 module.

Primul pas în folosirea comunicației cu module XBee este configurarea lor. Am folosit aplicația X-CTU în care am definit câmpurile necesare pentru crearea rețelei de comunicații. Câmpurile care definesc comunicația între XBee-urile sunt: [27]

- CH = Este canalul de comunicație în care funcționează modulele de XBee.
- PAN ID = Personal Area Network ID are setată o valoare între 0 și FFFE și este un identificator unic pentru rețea. XBee-urile comunică între ele doar dacă au același PAN ID. Aceasta permite și existența mai multor rețele în aceeași zonă.
- MY Address = Este adresa acestui XBee și este folosită de alte dispozitive pentru a trimite informația.
- Destination Adress High = Este prima jumătate din adresa de 64 de biți către care vrem sa trimitem mesaje.
- Destination Adress Low = Este a doua jumătate din adresa de 64 de biți a adresei destinație

Este foarte important în configurarea XBee-urilor ca identificatorul PAN ID și canalul de comunicație CH să aibă aceleași valori la ambele module. Pentru ca un XBee să comunice cu celălalt trebuie ca adresa lui destinație să fie la fel cu adresa sursă a celuilalt și vice-versa.

Dupa configurare, XBee-urile poate să comunice între ele și următorul pas este fixarea unui modul XBee pe quadcopter iar celălalt va fi conectat la laptop.

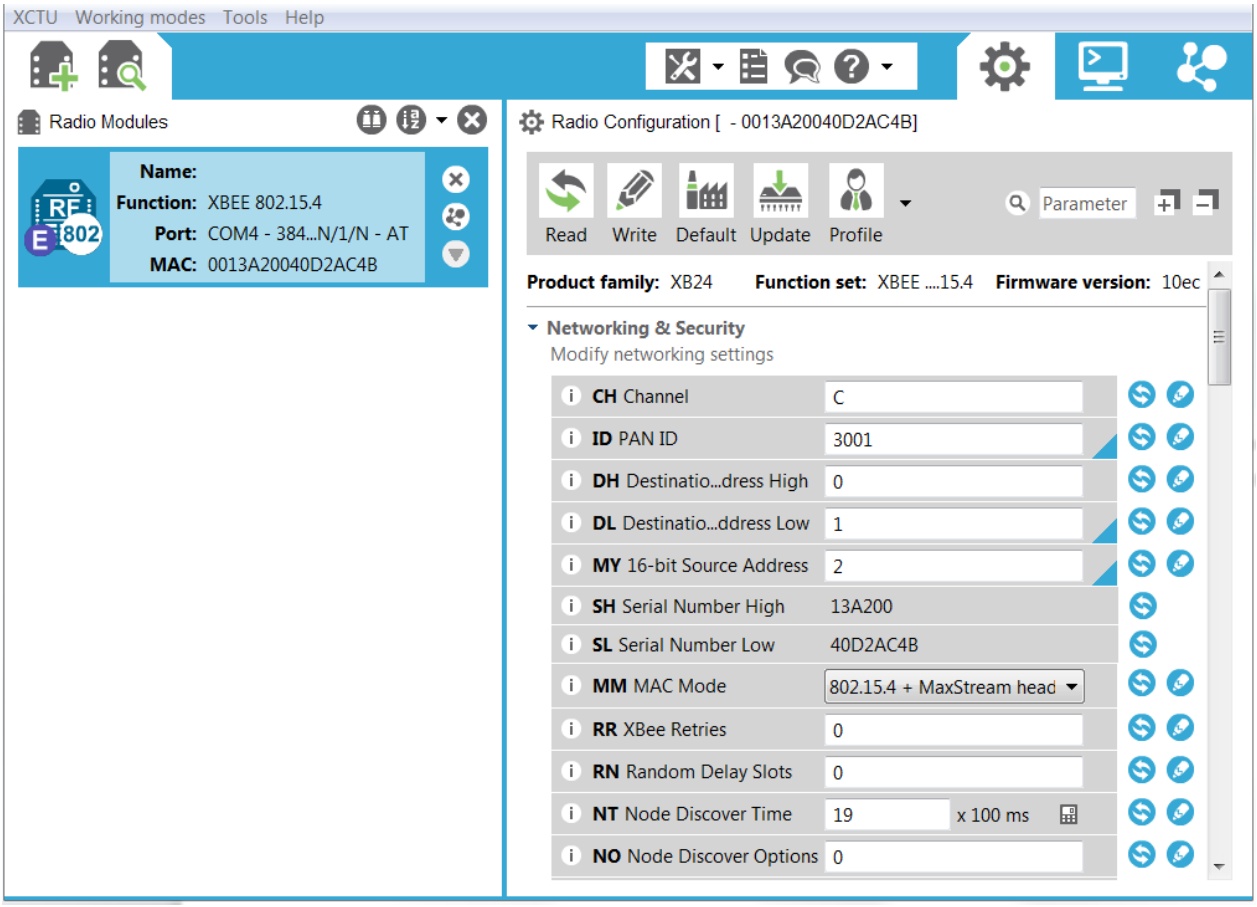


Figura 4-1 Aplicația X-CTU pentru configurarea XBee-urilor

4.4.3 Folosirea unui PCB pentru interconectare

Quadcopterul va fi pus în funcțiune și trebuie să fie un sistem cât mai stabil și cât mai rezistent. Pentru a proteja componentele și pentru a evita firele necesare în conexiuni, se va proiecta și se va printa un shield PCB[29]. Acest shield va conține legaturi electrice interne, găuri de fixare pentru microcontrolerul de comandă, pentru modulul de XBee și pentru alte componente necesare.

Primul pas este proiectarea circuitului în PROTEUS[28], care este un software de design asistat de calculator. Circuitul a fost gândit să facă legăturile între microcontrolerul de comandă și cel de zbor și modulul XBee.

Toate terminațiile conexiunilor au fost proiectate astfel încât conectoarele de diferite mărimi să depindă de numărul pinilor folosiți la fiecare. Canalele de comunicație au nevoie de 3 pini (unu ground, unu alimentare și unu comandă) iar pentru micrcontrolerul de zbor si modulul XBee au fost folosite conectoare de 6 sau 8 pini. Schema electrică este vizualizată în figura de mai jos.

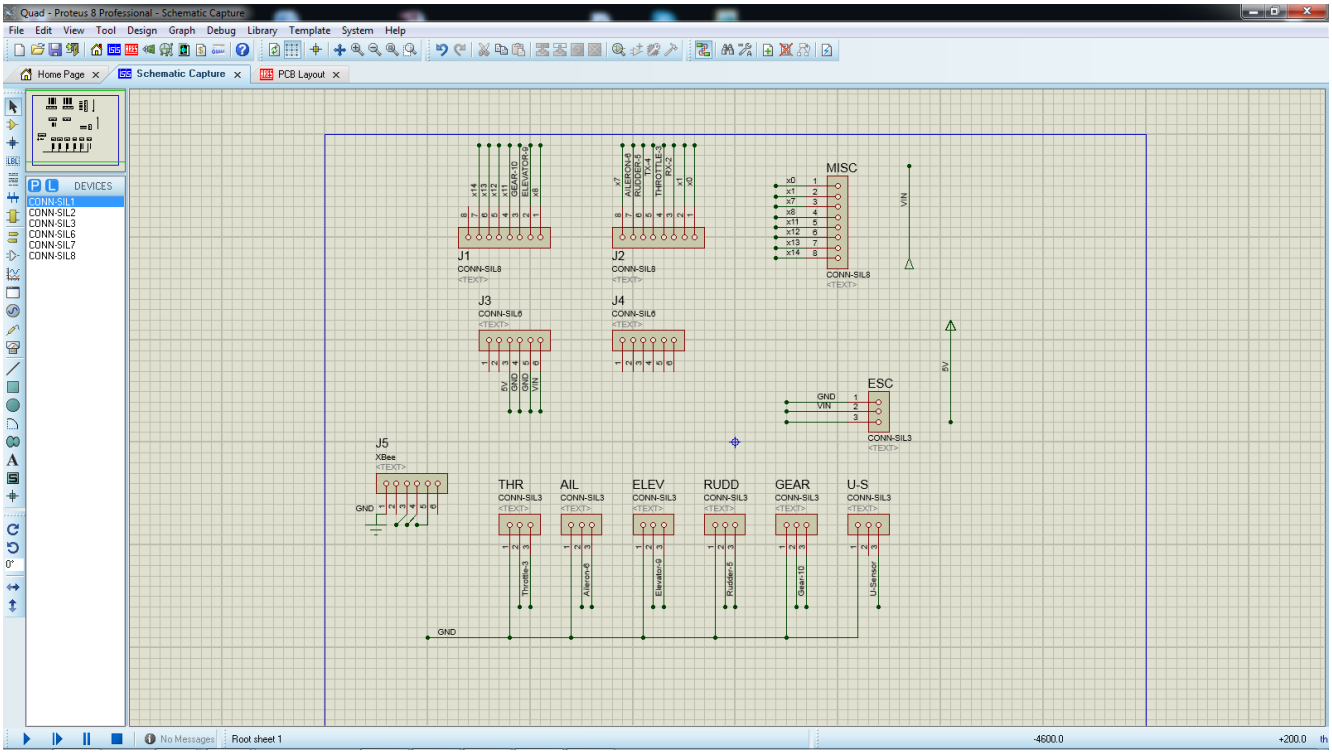


Figura 4-6 Schema electrică a circuitului

Dupa ce schema electrică a fost terminată, circuitul a fost trecut în ARES Proteus pentru definirea layout-ului. În această etapă s-a efectuat proiectarea componentelor și legaturilor pe baza capsulelor care vor fi folosite mai departe pentru printarea circuitului. Traseele au fost proiectate cu dimensiune de 0.5mm pentru a rezista loviturilor fizice și pentru o durată de viața mai mare. Pe layout s-au introdus și etichetele necesare pentru identificarea conectoarelor.

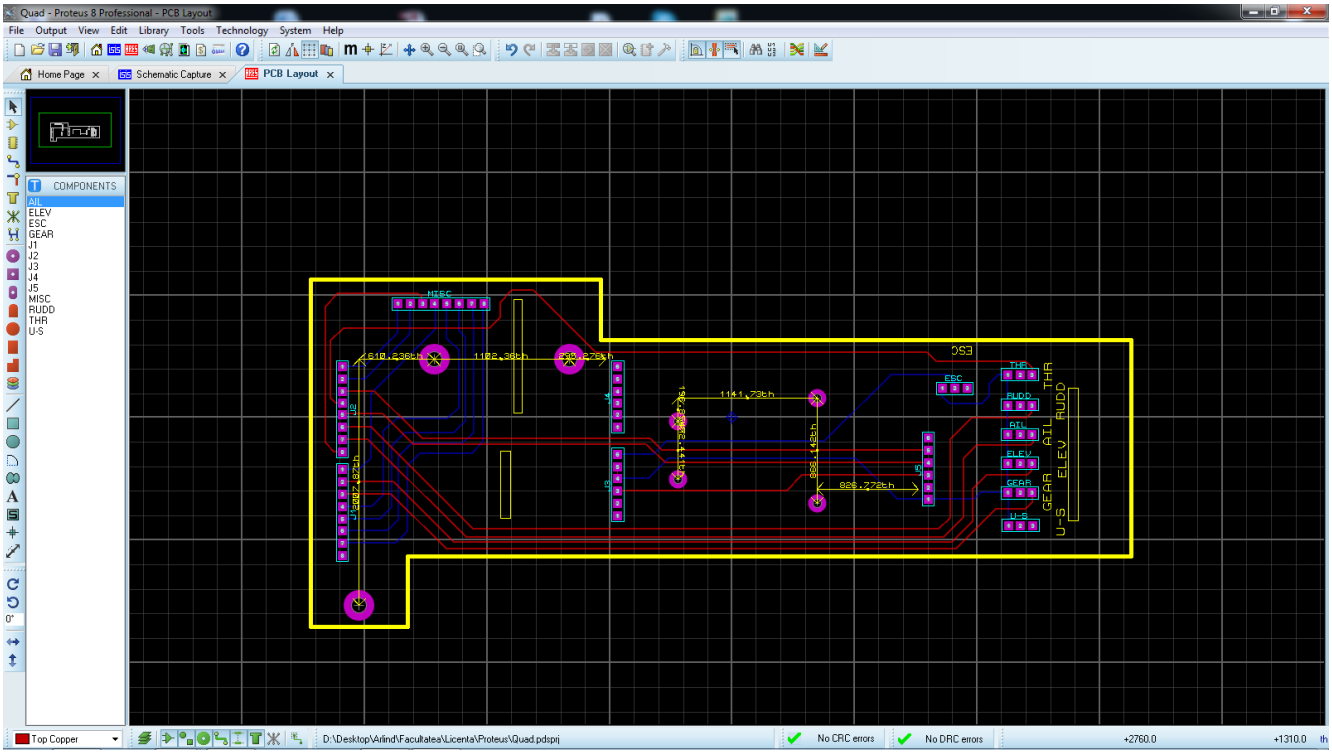


Figura 4-7 Layout-ul circuitului electric

Pentru fixarea componentelor au fost proiectate găuri conform datelor obținute de la aceste componente. Layout-ul a fost lucrat pe ambele parți (Top-Bottom) pentru a ușura proiectarea. Această placă PCB a fost proiectat la laboratorul de CETTI[30] cu ajutorul echipei lor.

Dupa ce PCB-ul a fost proiectat, s-a efectuat fixarea conectorilor folosind o stație de lipit și materialele necesare (fludor, conectoare). Componentele externe au fost fixate pe placă și pentru realizarea conexiunilor finale au fost folosite jumpere între componente și conectoare.

XBee-ul pus pe PCB a fost conectat la Arduino folosind arhitectura următoare:

- Pinul 0 de la XBEE este pinul de GND.
- Pinul 3 de la XBEE este pinul de VCC și se conectează la pinul Arduino 5V.
- Pinul 4 de la XBEE este pinul Tx și se conectează la pinul digital 2 al Arduino.
- Pinul 5 de la XBEE este pinul Rx și se conectează la pinul digital 4 al Arduino.

Cu toate conexiunile făcute, quadcopterul în momentul de fața este mai stabil și componentele pe el sunt mai protejate. Următorul pas este efectuarea testelor în teren deschis și cu elicele montate pe el.

4.5 Teste în teren deschis

Testele în teren deschis presupun folosirea sistemului construit în momentul de fața cu elicele puse și cu comunicația stabilită. Trebuie avut în vedere faptul că acest proces este destul de periculos fiindcă elicele se rotesc la o viteză foarte mare care poate cauza diferite accidentări. Din cauza aceasta, controlul quadcopterului trebuie făcut de la o distanță destul de sigur. Dacă testele se fac pentru prima dată, pasul cu care crește viteza motoarelor (pasul de accelerație) trebuie sa fie mai mic. Asta ajută în urmărirea mai eficientă a modului de comportament al quadcopterului și de asemenea ajută în oprirea mai rapidă a sistemului dacă se observ miscări periculoase a lui.

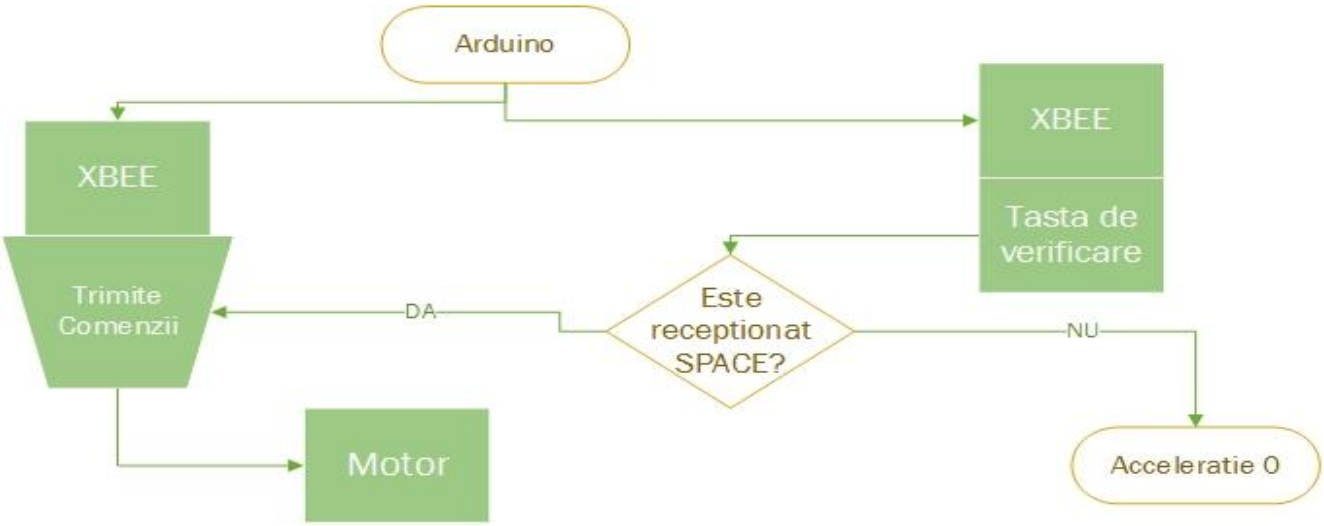


Figura 4-8 Schema de control a comunicației

Primele teste în teren deschis au fost facute într-un teren de fotbal cu spațiu de lucru foarte mare. După ce sistemul a fost alimentat, s-a urmărit procedura de armare și quadcopterul aștepta instrucțiuni de mișcare mai departe. Având în vedere instabilitatea comunicației, am modificat algoritmul astfel încât să oprească quadcopterul în caz de pierdere a semnalului. Pentru aceasta, am inclus în algoritm o

procedură de verificare trimițând la fiecare secundă sau ½ secunde la microcontrolerul de comandă. Dacă acesta raspunde, atunci quadcopterul continuă zborul, iar dacă este întreruptă conexiunea atunci quadcopterul intră în starea cu accelerație 0.

4.5.1 Programarea cu Raspberry Pi 3

Primul pas cu RPi[7] este configurarea corectă pentru a rula și executa programele pe care le vom folosi în continuare. Cum a fost introdus și în capitolul 1.3.2, RPi este un minicalculator care rulează sistemul de operare numit Raspbian Jessie. Acesta este încărcat pe un Card SD care va fi folosit la RPi. Se conectează RPi la calculator folosind un cablu de ethernet și se alimentează prin cablu USB. Pentru a accesa resursele oferite de RPi, ne conectăm la el folosind conexiune SSH și aplicația PUTTY.[18]

La conectarea RPi-ului cu calculatorul, activăm un server DHCP necesar pentru RPi astfel încât el sa-și ia adresa IP necesară. Această adresă se folosește mai departe pentru a se conecta la RPi.

Pasul următor este configuarea conexiuni wireless pentru a descărca resursele folosite mai departe. Acest pas va fi descris detaliat în capitolul 4.5.2.

După conectarea cu Wi-FI, se va face update la sistemul de operare folosind comenzile:

```
sudo apt-get update
sudo apt-get upgrade
```

Se va instala limbajul de programare Python cu pachetele necesare pentru rularea programului în sistemul de operare:

```
sudo apt-get install python-pip
```

În continuare, arhitectura de funcționare va rămâne la fel ca la Arduino, prin umare trebuie să configurăm RPi astfel încât să putem rula programul nostru și controla pinii digitali de ieșire. Pentru a accesa pinii de Input/Output se va folosi o librarie de Python numit python-rpi.gpio:

```
sudo apt-get install python-rpi.gpio
```

Pentru un alt mod de adresare mai apropiat de Arduino IDE, este folosită biblioteca wiringPi[31]. Această librărie este concepută să fie utilizată ușor de cei obișnuiți cu Arduino.

Pentru descărcarea acestei librării și adăugarea pe RPi se vor folosi comenzile următoare:

```
git clone git://git.drogon.net/wiringPi
cd wiringPi
git pull origin
./build
```

Inițializarea pinilor se face folosind modul de adresare conform biblioteci wiringPi și intrucțiunile folosite sunt următoarele:

- Se face referința conform bibliotecii wiringPi
- Se setează numărul pinului și modul de funcționare (0 dacă pinul va fi folosit ca intrare, 1 dacă va fi folosit ca ieșire și 2 dacă va fi folosit ca un pin de PWM)

```
wiringpi.wiringPiSetupGpio()
wiringpi.pinMode(17, 2)
```

Modul de funcționare al algoritmul va fi la fel ca la Arduino, iar celelalte funcții folosite pentru transformarea codului în platforma Python vor fi:

```
def generateDefaultPWM(pin):                    (1)
        wiringpi.digitalWrite(pin, 1)
        time.sleep(15/10000)
        wiringpi.digitalWrite(pin, 0)
        time.sleep(5/10000)
state==STATE_ALL_IDLE:                          (2)
                generateDefaultPWM(17)
                generateDefaultPWM(18)
                generateDefaultPWM(22)
                generateDefaultPWM(23)
                generateDefaultPWM(24)
                time.sleep(1/100)
```

Prima funcție va fi folosită pentru generarea semnalelor de tip PWM cu perioada de 2ms, iar a doua pentru crearea stărilor care corespund semnalelor PPM generate cu perioada de 20 de ms.

În programul scris pe Arduino, funcția care crează bucla infinită este *void loop(),* iar în Python pentru a crea un mediu asemănător se va folosi funcția *while 1(),* în bucla căruia se va introduce codul de funționare cu toate stările și alte funcții necesare pentru controlul quadcopterului.

Comenzile se vor trimite la RPi conectându-se la el prin wireless și folosind consola de comandă prin aplicația Putty.

4.5.2 Comunicația cu Wi-FI

Comunicația Wi-FI[25] oferă avantaje majore față de cea cu Xbee[11] doarece putem să transmitem mai multe date și informații. Tot așa raza de funcționare este mai mare și poate fi mărită în funcție de routerul folosit. În proiectul nostru, vom avea nevoie de o rază relativ mică, așa că putem să folosim un router mai ieftin sau să creăm o rețea locală între calculator și RPi.

Configurarea: `sudo iwlist wlan0 scan =` Listează toate rețelele aflate de RPi.

`ESSID:"Nume" =` Este numele rețelei la care ne vom conecta.

Pentru adăugarea rețelei în RPi, vom deschide fișierul de configurare și vom introduce datele rețelei.

`sudo nano /etc/wpa_supplicant/wpa_supplicant.conf` = deschide fișierul de configurare în care com adăuga următoarea linie:

```
network={
    ssid="Nume"
    psk="Parolă"
}
```

Dupa ce fișierul a fost salvat, RPi va detecta rețeaua nouă și se va conecta la ea prin wireless folosind un nou IP. Mai departe,folosind PUTTY ne vom conecta la această rețea cu IP-ul nou și toate operațile se vor face de la distanță.

Comunicația wireless va fi folosită în continuare ca mediu de transmisie a datelor de orice fel incluzând informațiile de configurare, datele video etc.

CAPITOLUL 5 TRANSMISIA VIDEO ȘI PRELUCRAREA IMAGINII

Această etapă include toți pașii necesari pentru configurarea corectă a sistemului astfel încât să realizeze transmisia video și prelucrarea imaginii. În sistem se va introduce o cameră compatibilă cu RPi și va fi creat un program software pentru îndeplinirea sarcinilor. Transmisia video se va face pe o pagină web și accesul poate fi făcut de la orice dispozitiv conectat în rețeaua locală, având ruta corect definită.

5.1 Transmisia video live pe pagina web locală

Transmisia video pe pagina locală presupune captarea video cu RPi și accesarea acesteia de pe o pagină web folosind un port în rețeaua de transmisie. Pentru realizarea acestei sarcini se va folosi camera de tip RPi Vision 2. Ea va fi conectată la portul CSI (Camera Serial Interface) în timp ce RPi este nealimentat.

5.1.1 Configurări de bază

Toate comenzile în continuare se vor da la terminalul RPi conectându-se la el prin SSH. Activarea camerei se va face folosind comenzile următoare:

```
sudo apt-get update          //Se va face update și upgrade la software
sudo apt-get upgrade
sudo raspi-config           //Se va deschide fereastra de configurare
```

După ultima instrucțiune se va deschide fereastra de configurare unde se va activa modulul de cameră:

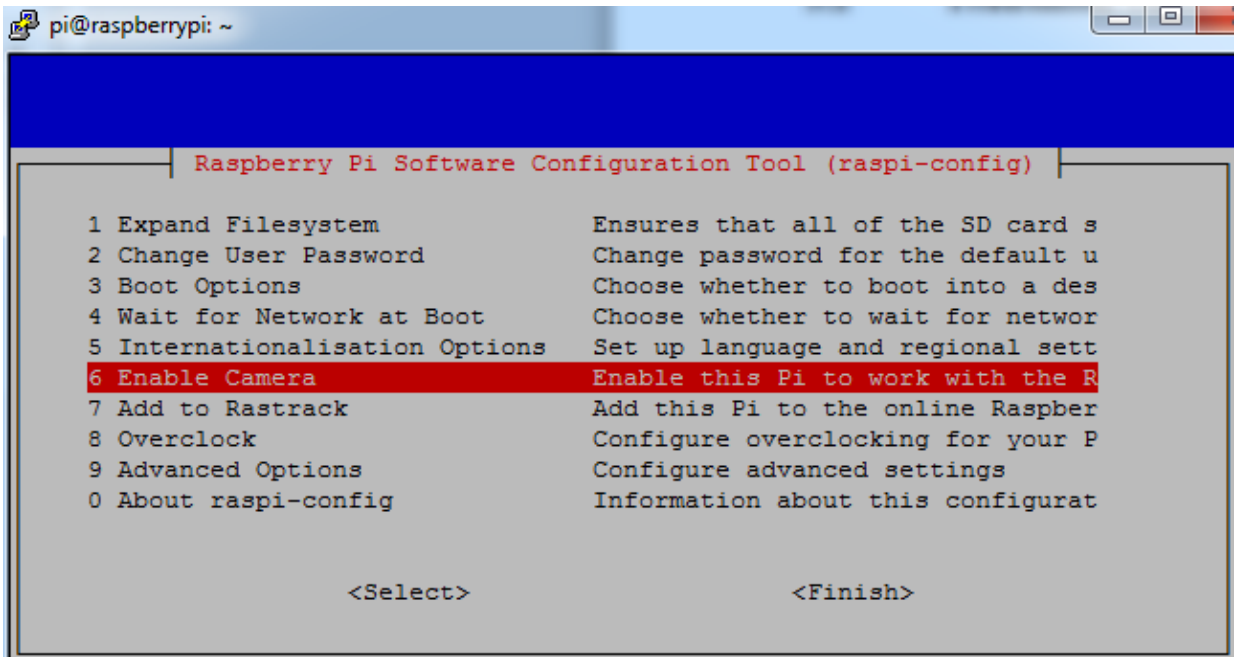


Figura 5-1 Configurarea camerei RPi

Pentru a testa funcționarea camerei, introducem în terminal următoarea comandă:

```
raspistill -v -o test.jpg   //Imaginea de test capturată
```

Aceasta va captura o imagine folosind camera de la RPi și va salva imaginea în directorul de user Pi cu numele test.jpg. Pentru a putea vedea imaginea, avem nevoie de o interfață grafică GUI[32] a sistemului de operare oferit de RPi. De accea în continuare se vor realiza configurările necesare pentru crearea

unei interfețe GUI și includerea tuturor pachetelor și librăriilor necesare pentru realizarea transmisiei video în timp real.

Lucrul cu RPi poate fi dificil câteodată dacă nu există o interfață grafică. De accea se va folosi un sistem care ne permite să controlăm de la distanță interfața grafică a minicalculatorului Raspberry Pi. VNC (Virtual Network Control)[33] este un software care transmite comenzile date de la tastatură sau mouse și primește actualizarea prin rețea de la distanță. Pentru instalarea acestui software se va folosi:

```
sudo apt-get install tightvncserver //Instaleaza software-ul
tightvncserver                      //Crează parola pentru accesare
vncserver :1 -geometry 1920x1080 -depth 24
       //crează serverul 1 cu dimensiunile precizate
```

Pentru accesarea lui, se va folosi aplicația de VNC pe calculator unde se vor introduce IP-ul interfeței a RPi-ului și numărul serverului sub forma: `<IP:x.x.x.x>:1 //Pentru a accesa serverul 1`

Interfața grafică va arată în felul urmator, unde se vede și imaginea de test capturată mai sus.

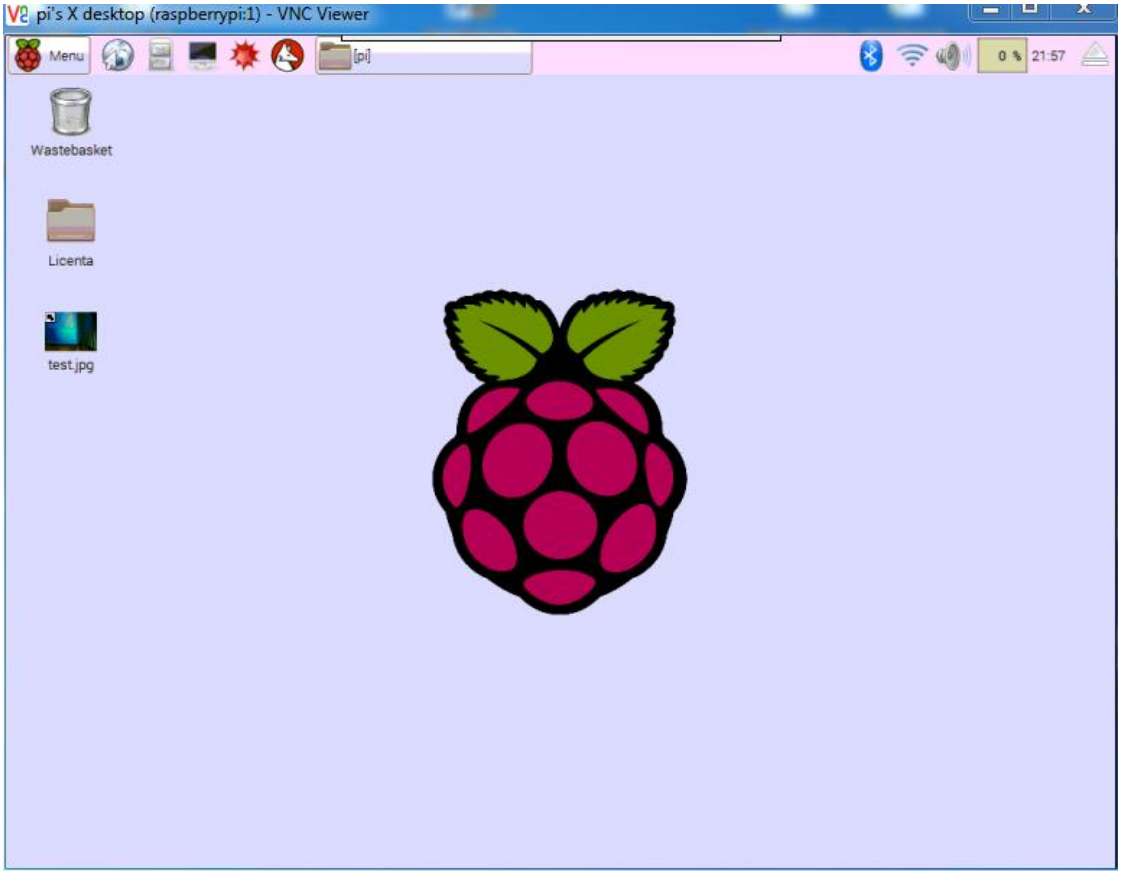


Figura 5-2 Interfața grafică la Raspberry Pi

În continuare se vor instala toate pachetele necesare pentru limbajul de programare Python și pachetele pentru crearea stream-ului video.

```
sudo apt-get update
sudo apt-get install python-pip
sudo apt-get install python-picamera
```

Comenzile de mai sus descarcă și instalează pachetul de Python.

5.1.2 Modul de funcționare a transmisiei video

Pentru crearea paginii web și încărcarea stream-ului pe ea se va folosi un cadru de dezvoltare web numit Flask[19] care permite utilizarea limbajului Python pentru programare și oferă instrumente, biblioteci și tehnologii care ajută în programarea unei aplicații web.

Arhitectura necesară pentru rularea aplicației este următoarea: [34]

```
/Stream_video
    /templates
        Index.html
    app.py
    camera_pi.py
```

Stream_video este folderul care include toate componentele. În templates se găsește șablonul web folosit pentru aplicația care este disponibilă prin intermediul protocolului HTTP. Acestă pagină web este interfața cu utilizatorul și pentru crearea ei este folosit limbajul de marcare HTML.[17]

Pentru generarea șablonului de tip *.html* a fost folosit un template de la Jinja. [21] Codul pentru programarea paginii este prezentat în Anexă. [Anexa-2]

- Eticheta bază de definire este <*html*>.
- Documentul conține două părți importante: *antetul și corpul*. În secțiunea de corp sunt definite informații care apar în șablon, iar cele din secțiunea de antet nu apar.
- Eticheta <*title*> definește titlul paginii.
- Se importă biblotecile de la *jQuery* pentru definirea mai ușoară a stilului folosit.
- Se definește clasa cu atributul *.centered* care va fi folosită pentru alinierea obiectelor în pagină.
- Se creează selectori cu etichetele <h3> și <h4> folosind limbajul CSS.
- Se definește clasa cu atributul *.btn* care va fi atribuită în continuare butonului.
- Eticheta <*body*> definește corpul paginii unde vor fi introduse toate elementele care se vor afișa pe pagină.
- Antetelor li se atribuie clasele definite mai sus și primesc proprietățile lor.
- Se folosește funcția anonimă pentru evenimentul *click* care la apăsarea butonului *Toggle Stream* va afișa imaginile primite de la aplicație.

La sfârșit, pagina web va arăta așa cum este afișată în imaginea de mai jos:

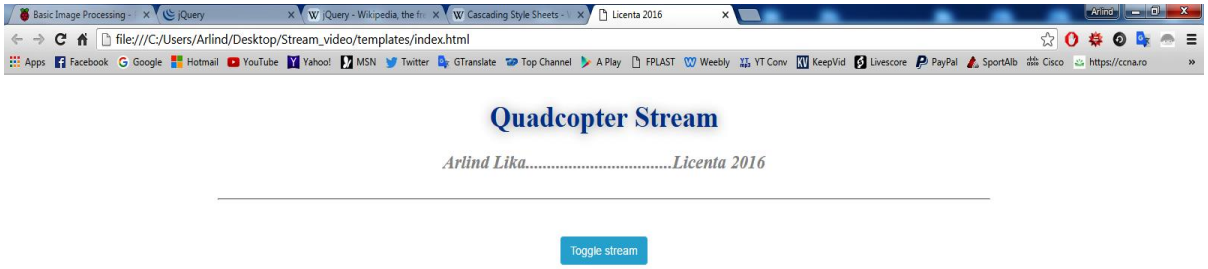


Figura 5-3 Pagina web pentru streaming video

La apăsarea butonului, după ce serverul este activat și paginea este accesată, se vor afișa imaginile de la camera RPi.

Scriptul care realizează transmisia video, *app.py*, este scris în limbajul de programare Python și accesează resursele necesare pentru îndeplinirea sarcinii. Se creează scriptul unde se vor introduce toate funcțiile pentru captarea video și transmisia mai departe pe pagina web locală. În aplicație se va adăuga o nouă rută care va conduce la pagina web creată de noi. Aplicația va rula în portul 5000 și poate fi accesată de orice dispozitiv conectat în aceeași rețea folosind IP-ul interfeței wireless de la RPi, portul și ruta de către aceasta.

O scurtă descriere a aplicației cu funcțiile ei principale este dată mai jos: [Anexa-2] [34]

```
from flask import Flask, render_template, Response          (1)

app = Flask(__name__)                                       (2)

@app.route('/Licenta2016')                                  (3)
def index():
    return render_template 'index.html'

def gen(camera):                                            (4)
                    //Funcția de generarea a frame-urilor

@app.route('/video_feed')                                   (5)
       def video_feed():
                    //Funcția de încărcare a frame-urilor pe pagină web

if __name__ == '__main__':                                  (6)
    app.run(debug=True, host='0.0.0.0')
```

(1) Din pachetul flask se importă bibliotecile necesare pentru proiect. Flask este biblioteca ce realizează conexiunea aplicație-server și deschide ruta necesară pentru aplicația la portul 5000. Render_template este necesară pentru a crea un template folosind mediul de dezvoltare Jinja, în

care este creată pagina web pentru transmisie, iar Response este folosit pentru primirea răspunsului de la funcțiile de generare. O funcție de generare este o funcție specială care poate fi întreruptă și reluată din nou.

(2) Se creează aplicația Flask cu proprietățile enumerate mai sus. Se creează serverul, legatura cu aplicația și ruta de accesare. În spatele acestei aplicații sunt definite toate componentele necesare pentru trecerea informației pe pagina web (protocoale de comunicații și de deschidere a porturilor în rețea)

(3) Se definește extensia din ruta definită *URL <X.X.X.X:Port/.....>* . După ce se accesează această pagină web, serverul primește răspunsul și aplicația încarcă șablonul paginii web. Pagina web *index.html* a fost scrisă și codată în limbajul HTML.

(4) Se definește funcția generatoare care realizează capturarea frame-urilor. Această funcție folosește camera de la RPi și funcții definite pentru a prelua imaginea.

(5) Se realizează încărcarea frame-urilor pe pagina web. După ce pagina web a fost accesată, aplicația va genera frame-urile primite de la *def gen (camera).*

(6) Se rulează aplicația pe rețeaua locală. Orice dispozitiv în rețea poate să acceseze pagina și să vizualizeze conținutul ei.

Funcția care generează frame-urile este dată în Anexă [Anexa 2]. Funcționarea ei este următoarea:

- Se importă bibliotecile necesare (time,io,threading,picamera). Biblioteca *time* ajută în definirea timpilor, *io* exportă funcțiile care transformă imaginea în informație de biți, *threading* ajută în operațiile cu firul de execuție a calculatorului și *picamera* în capturarea imaginii.

- Se creează o clasă cu numele *Camera* care conține obiectele *thread, frame și last_acces*.

- Se definește funcția care generează firul de execuție a camerei. Această funcție creează firul de execuție pentru cameră și începe funcționarea ei.

- Se definesc parametrii pentru imaginea care va fi captată. Dimensiunea ei va fi 640x480 și va fi rotită astfel încât direcția să fie cea dorită.

- Se va captura imaginea și se va salva în format *.jpeg*. După ce este citită, se va elibera locația pentru următoarea imagine. Acest proces va continua până când timpul de acces a ultimului dispozitiv depășește 10 secunde. Firul de execuție se va opri. La următoarea conexiune, se va repeta procedura și în felul acesta pe pagina web va apărea transmisia video live. Pentru definirea frame-urilor pe secundă, am folosit o funcție dinamică care trunchiază fluxul

informației la momentul curent și trece la următorul imediat după. Putem să definim și o transmisie statică a frame-urilor, de exemplu 30fps.

```
@classmethod
  def _thread(cls):
    with picamera.PiCamera() as camera:

      camera.resolution = (640, 480)
      camera.hflip = True
      camera.vflip = True
      camera.framerate = 30

      camera.start_preview()
      time.sleep(2)

      stream = io.BytesIO()
      for foo in camera.capture_continuous(stream, 'jpeg',
                                           use_video_port=True):
        stream.seek(0)
        cls.frame = stream.read()

       stream.seek(0)

       stream.truncate()
```

- Funcția *io.BytesIO()* [35] realizează colectarea informațiilor despre imaginea respectivă. Acesta îi oferă interfeței posibilitatea de a transmite și de a afișa frame-ul respectiv. *Stream* este variabila care stochează această informație și la fiecare transmitere se resetează din nou pentru a prelua noua imagine. Firul de execuție va continua operația folosind camera ca un obiect.
  Pentru realizarea transmisiei, pe aplicația scrisă în scriptul de sus este definită ruta de trecere a imaginilor.

```
@app.route('/video_feed')
def video_feed():
  return Response(gen(Camera()),
          mimetype='multipart/x-mixed-replace; boundary=frame')
```

Această funcție returnează un obiect de tip răspuns multiplu care este generat în mod continuu de către funcția *gen(Camera).* Fără aceasta, pe pagina web se va afișa doar o imagine statică.

În șablonul scris pentru pagina web, afișarea se face folosind:

```
<img src="{{ url_for('video_feed') }}">
```

În Jinja2 delimitatori de tip `{{ ... }}` sunt folosiți pentru a afișa expresii. [21]

Imaginea de mai jos arată pagina web la un dispozitiv mobil inteligent:

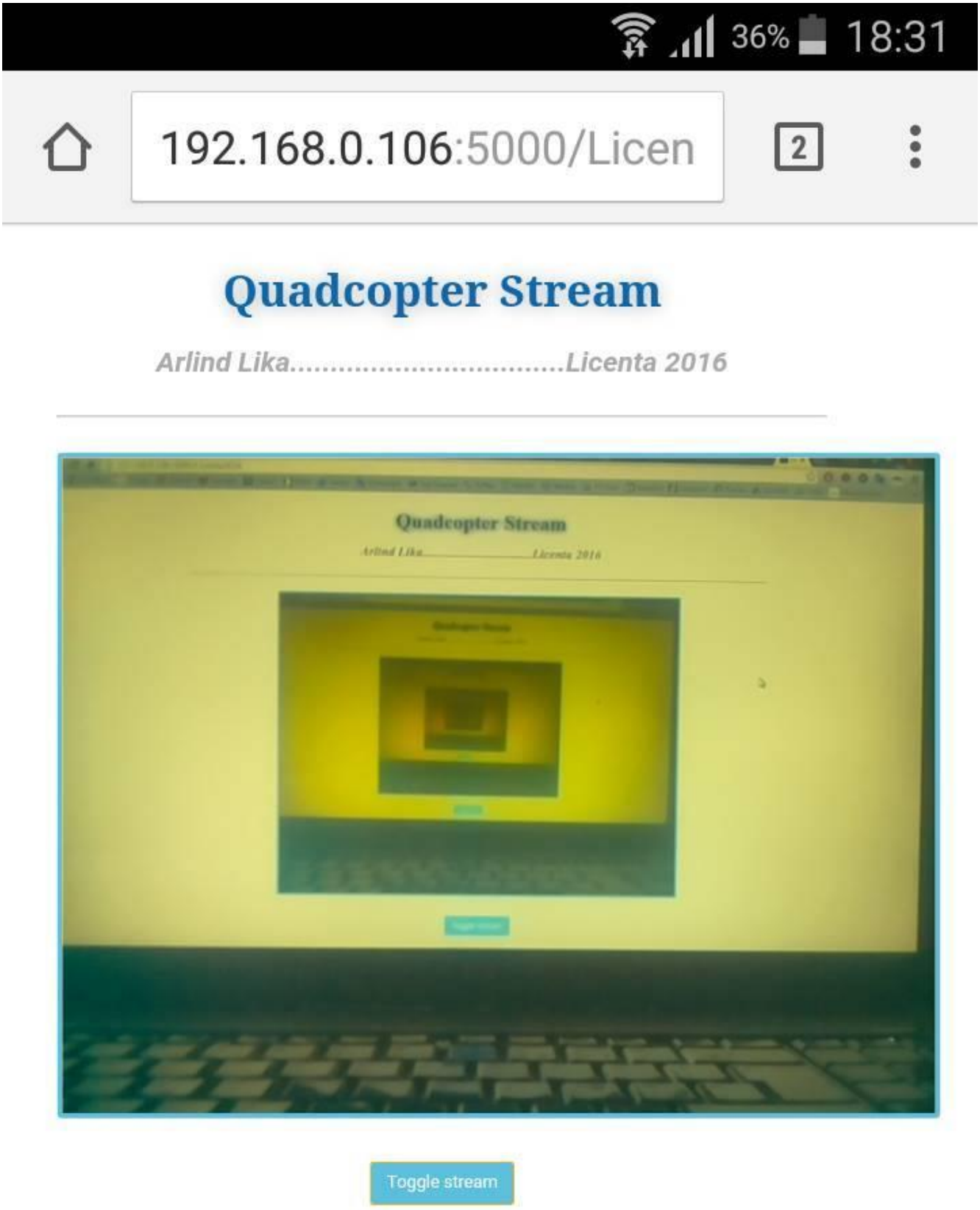


Figura 5-4 Streaming live pe smartphone

5.2 Obținerea și prelucrarea frame-urilor

Pentru obținerea și prelucrarea frame-urilor va fi folosit în continuare un script scris în Python și biblioteca de prelucrarea imaginii *imgproc*[37]. Pentru instalarea acestei biblioteci pe RPi se va folosi comanda următoare:

```
Wget
http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/resources/imgpr
oc.zip
unzip imgproc.zip
cd library
sudo make install
cd ..
```

Această bibliotecă de prelucrare a imaginii conține funcții diferite pentru capturarea imaginii, exemple și coduri sursă, toate în ambele limbaje de bază de programare ale RPi-ului, Python și C.

Scriptul care realizează prelucrarea imaginilor este prezentat mai jos:

```
from imgproc import *                                                    (1)

cam = Camera(480, 360)                                                   (2)

view = Viewer (cam.width, cam.height, "Text_afișat")                     (3)

while true                                                               (4)
       image=cam.grabImage()                                             (5)

              #Codul pentru prelucrarea imaginii                         (6)

       view.displayImage(image)                                          (7)
```

1. Se importă biblioteca de procesare a imaginii. Această bibliotecă oferă funcțiile de bază pentru accesarea camerei, pentru accesarea datelor imaginii și alte funcții cu complexitate mai mare pentru prelucare.
2. Se realizează accesarea camerei și se setează dimensiunile dorite.
3. Se deschide fereastra de afișare a imaginii cu dimensiunile alese mai sus și cu titlul dorit.
4. Se creează o buclă infinită care va fi folosită pentru captarea imagini, prelucrarea și afișarea ei.
5. Se capturează imaginea.
6. Va fi descris codul pentru prelucrare.
7. Se realizează afișarea imaginii pe fereastra deschisă mai sus.

La rularea acestui script, se va realiza doar capturarea imaginii și afișarea ei. Având în vedere că biblioteca *imgproc* ne oferă acces la datele imaginii, atunci putem să accesăm și să prelucrăm fiecare pixel în parte folosind comanda următoare:

```
for x in range(0, image.width):          //Parcurge toată matricea imaginii
  for y in range(0, image.height):       // ------------------------------
    red, green, blue = image[x, y]       //Extragem valoarea fiecărei culori
```

Un pixel al imaginii este definit ca un vector cu 3 variabile, unde prima este culoarea roșie, a doua este culoarea verde și a treia este culoarea albastru. De exemplu, dacă dorim să schimbăm culoarea unui singur pixel, putem să introducem valorile fiecărei culori:

```
my_image[x, y] = red_value, gree_value, blue_value //Setează valorile
```

Această instrucțiune introduce în poziția respectiva (x,y) noile valori ale pixelului.

La introducerea codului de prelucrare, se va observa o întârziere și imaginile se vor afișa cu o viteză mai mică decât în mod normal. Aceasta se întamplă doarece fiecare pixel al imaginii este testat și prelucrat în parte, un proces care necesită timp. Pentru a accelera procesul, putem sa schimbăm rezoluția imaginii într-una mai mică. Aceasta se va traduce în mai puține operații și mai puțin timp consumat. Schimbarea rezoluției se face schimbând dimensiunile imaginii alese de la început:

cam = Camera(160, 120) sau *cam = Camerea(100, 80)*

Aceasta va micșora dimensiunea imaginii dar și durata procesului, fiindcă la parcurgerea pixelilor vor fi mai puțini de prelucrat.

CONCLUZII

1.Concluzii generale

Obiectivul principal al acestei lucrări a fost dezvoltarea unui sistem de tip quadcopter comandat de la distanță, care să fie capabil să zboare, să transmită informații video și imagini în timp real și să atingă anumite scopuri realizând o prelucare a imaginii. Pentru realizarea întregului proiect a fost nevoie de componente hardware necesare pentru construirea sistemului fizic, de informații cum funcționează un quadcopter astfel încât să realizăm diferite simulări, de resurse software pentru generarea semnalelor de control și realizarea transmisiei video în timp real, și de aplicații pentru configurarea diferitelor piese sau pentru prelucrarea imaginilor.

În etapele de început au fost realizați cu succes pași necesari pentru simularea și configurarea corectă a pieselor. Folosind plăci de dezvoltare și resurse software, care constau în diferite limbaje de programare sau aplicații, a fost simulat cu succes controlul sistemului. În realizarea propriu-zisă, problema a apărut în comunicație, unde echipamentele folosite nu aveau puterea necesară de transmisie la o rază mare în teren deschis, și din cauza microcontrolerul de zbor, care nu făcea ajustările de traiectorie cu suficientă precizie. Toate simulările au avut succes în controlul sistemului și testele în interior arătau un sistem funcțional, iar experimentele în exterior au avut probleme din cauzele de mai sus. Pentru rezolvarea lor, am găsit o soluție fără să se schimbe arhitectura, integrând în sistem un alt microcontroler care realizează comunicația, iar balansarea rămâne o problemă a ansamblului folosit.

În integrarea camerei video, în corpul quadcopterului a fost montată o cameră care realizează captarea imaginilor în timp ce sistemul zboară. Realizarea transmisiei video din această cameră la o stație de bază (în cazul nostru o pagină web) a fost făcută cu succes. Acest pas a constat în folosirea resurselor software necesare pentru transmisie și pentru crearea unei pagini personalizate. Tot așa o prelucrare mai avansată a imaginilor a fost efectuată în stația de bază. Metodele folosite pentru transmitere și prelucrare au fost alese având în vedere dificultățile proiectului și rezultatele obținute.

La sfârșit, produsul final realizează un control corect al componentelor responsabile pentru mișcare, realizează o transmisie în stația de bază a imaginilor obținute cu o întârziere de aproape ½ secunde și realizează și operații necesare pentru prelucrarea acestora.

Per total, au fost studiate metode pentru generarea semnalelor, pentru realizarea comunicației și transmisiei, pentru prelucarea imaginilor și pentru realizarea unor arhitecturi funcționale; au fost folosite cu succes componentele hardware, aplicațiile pentru programarea și configurarea acestora și s-a realizat cu succes programarea și configurarea lor.

2.Contribuții personale

Consider ca această lucrarea include contribuții personale în realizarea practică a proiectului, dar necesită și informații teoretice pentru realizarea ei. Capitolele 4 și 5 descriu în detaliu contribuțiile

personale și pașii care au fost parcurși pentru realizarea proiectului. Sistemul a fost construit de mine începând cu lipirea diferitelor cabluri la componente și realizarea tuturor legăturilor electrice necesare. Diferite teste s-au făcut în continuare pentru verificarea pieselor și realizarea programelor. Am realizat configurarea ESC-urilor, generarea semnalelor PWM cu un microcontroler Arduino, controlarea motoarelor și diferite proceduri de siguranță ale sistemului. Am realizat transmisia și recepția comenzilor de la distanță folosind module de comunicație XBee și microcontrolerul Arduino. Pentru transmisia video, am lucrat cu placa de dezvoltare Raspberry Pi unde am dezvolatat cunoștințele despre sistemul de operare Linux și limbajul de programare Python. Am realizat transmisia video cu ajutorul unei structuri numite Flask și o pagină web folosind limbajul de marcare HTML. Pentru prelucrarea video am folosit diferite biblioteci de la Open CV sau de la Python și operații simple cu ele au fost realizate.

Piesele și echipamentele folosite fac parte din laboratorul de cercatre SpeeD[38] și sistemul contruit va rămâne în continuare în acest laborator pentru dezvoltări ulterioare.

3.Îmbunătățirea sistemului

Așa cum a fost menționate de la început, acest sistem de tip quadcopter are o multifuncționaliate foarte largă. Pentru realizarea unui sistem cât mai stabil și cât mai sigur va exista în continuare loc pentru îmbunătățire. În viitor se dorește ca acest sistem să fie comandat de la distanțe mai mari și să continue prelucarea și identificarea obiectelor. Pentru realizarea acestora, este nevoie de unele piese în plus, mai ales în partea de comunicație de la distanță. Următoarele sarcini ce pot fi îndeplinite:

- Înlocuirea microcontrolerului de zbor sau îmbunătățirea sistemului care este responsabil pentru menținerea stabilități.
- Folosirea unor module cu rază lungă de comunicație pentru realizarea controlului de la distanță.
- Introducerea comenzilor online de pe o pagină web, astfel încât controlul să fie făcut de oriunde atât timp cât microcontrolerul de comandă al quadcopterul este conectat la internet.
- Introducerea unui modul GPS care ajută în autonomia sistemului.
- Realizarea comunicației cu alte unități robotice pentru îndeplinirea sarcinilor mai complexe și urmărirea lor.

Aceste sarcini pot fi îndeplinite și vor ajuta în crearea unui sistem cât mai performant și stabil în mediul exterior.

BIBLIOGRAFIE

[1]Parallax Elev-8 V2: https://www.parallax.com/product/80200 accesat la data 30.09.2015

[2]Configurația de zbor: http://www.coptercraft.com/multirotor-frame-configurations/ accesat la data 16.06.2016

[3]HoverFly Open: https://www.parallax.com/sites/default/files/downloads/80000-Hoverfly-OPEN-Users-Guide-v1.0.pdf accesat la data 30.09.2015

[4] PWM: https://en.wikipedia.org/wiki/Pulse-width_modulation accesat la data 24.06.2016

https://learn.sparkfun.com/tutorials/pulse-width-modulation accesat la data 11.03.2016

[5] PPM: https://oscarliang.com/pwm-ppm-difference-conversion/ accesat la data 11.03.2016

[6] Arduino Uno: https://www.arduino.cc/en/Main/ArduinoBoardUno accesat la data 17.06.2016

[7] Raspberry Pi 3: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ accesat la data 17.06.2016

https://www.raspberrypi.org/documentation/installation/installing-images/README.md accesat la data 30.05.2016

[8] RPi GPIO: https://pimylifeup.com/raspberry-pi-gpio/ accesat la data 10.06.2016

[9] BEC: https://en.wikipedia.org/wiki/Battery_eliminator_circuit accesat la data 24.06.2016

[10] Brushless Motor: http://www.learnengineering.org/2014/10/Brushless-DC-motor.html accesat la data 24.06.2016

[11] XBee: https://en.wikipedia.org/wiki/XBee accesat la data 25.06.2016

https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu accesat la data 15.04.2016

[12] Imaginea Digitală: https://en.wikipedia.org/wiki/Digital_image accesat la data 20.06.2016

[13] Constantin Vertan „*Prelucarea și analiza imaginilor*” Ediția 1999

Brad R., Gellert A., „*Procesarea imaginilor*”, Editura Universității, Lucian Blaga, Sibiu 2003 http://webspace.ulbsibiu.ro/arpad.gellert/html/PI.pdf accesat la data 20.06.2016

[14] Detecția mișcării: http://www.sciencedirect.com/science/article/pii/B9780123965493000045 accesat la data 22.06.2016

[15] Moldoveanu F., „*Detecția frontierelor din imagine (Edge Detection)* ”

[16] Arduino IDE: https://www.arduino.cc/en/Guide/Environment accesat la data 18.06.2016

[17] HTML: http://www.w3schools.com/html/ https://en.wikipedia.org/wiki/HTML accesat la data 15.06.2016

[18] Putty: http://www.putty.org/ accesat la data 26.06.2016

[19] Flask: http://flask.pocoo.org/ https://pypi.python.org/pypi/Flask accesat la data 13.06.2016

https://www.fullstackpython.com/flask.html accesat la 13.06.2016

https://www.raspberrypi.org/learning/python-web-server-with-flask/ accesat la data 13.06.2016

[20] WSGI: https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface accesat la data 13.06.2016

https://www.python.org/dev/peps/pep-0333/ https://wsgi.readthedocs.io/en/latest/ accesat la data 13.06.2016

[21] Jinja2: http://jinja.pocoo.org/docs/dev/ http://jinja2.readthedocs.io/en/latest/templates.html accesat la data 13.06.2016

[22] M JPEG: https://en.wikipedia.org/wiki/Motion_JPEG accesat la data 18.06.2016

[23] Open CV: http://opencv.org/ https://en.wikipedia.org/wiki/OpenCV accesat la data 18.06.2016

[24] Linux: https://www.linux.com/ https://en.wikipedia.org/wiki/Linux accesat la data 23.06.2016

[25] Wi-Fi: https://en.wikipedia.org/wiki/Wi-Fi accesat la data 24.06.2016

[26] Controler Radio: http://adamone.rchomepage.com/guide1.htm accesat la data 25.03.2016

http://www.rc-airplane-world.com/radio-control-gear.html accesat la data 25.03.2016

[27] X-CTU: http://www.libelium.com/development/waspmote/documentation/x-ctu-tutorial/ accesat la data 22.04.2016

[28] Proteus Ares: http://www.theengineeringprojects.com/2013/09/pcb-designing-in-proteus-ares.html accesat la data 23.06.2016

[29] PCB: https://learn.sparkfun.com/tutorials/pcb-basics accesat la data 23.06.2016

[30] CETTI: http://www.cetti.ro/v2/ accesat la data 23.06.2016

[31] wiringPi: http://wiringpi.com/ http://raspi.tv/2013/how-to-use-wiringpi2-for-python-on-the-raspberry-pi-in-raspbian accesat la data 10.06.2016

[32] GUI: https://en.wikipedia.org/wiki/Graphical_user_interface accesat la data 23.06.2016

[33] VNC: https://www.raspberrypi.org/documentation/remote-access/vnc/ accesat la data 01.06.2016

[34] Video Streaming: http://blog.miguelgrinberg.com/post/video-streaming-with-flask

http://raspberrypi.stackexchange.com/questions/42759/streaming-raspberry-pi-camera-to-html-webpage-using-picamera-and-flask accesat la data 14.06.2016

[35] Stream io.BytesIO: https://docs.python.org/3.2/library/io.html#io.BytesIO accesat data la 15.06.2016

[36] Camera Pi: http://picamera.readthedocs.io/en/release-1.10/api_camera.html accesat la data 10.06.2016

[37] Biblioteca Img: https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/image_processing/ accesat la data 19.06.2016

[38] SpeeD : http://speed.pub.ro/

ANEXA 1

Program pentru calibrarea ESC-urilor

```
#include<Servo.h>

int maxi=2000;
int mini=1000;
int pin=9;

 Servo motor;
 void setup() {
  Serial.begin(9600);
  Serial.println("Incepe Calibrarea");
  motor.attach(pin);
}

 void loop() {

    if (Serial.available() > 0){
    command=Serial.read();
    switch (command){
    case 97: //a
      motor.writeMicroseconds (maxi);
      break;
    case 100: //d
      motor.writeMicroseconds(mini);
      break;
 }
}
```

Program pentru armarea quadcopterului

```
int throttlePin=3; //Throttle
int rudderPin=5; //Yaw
int aileronPin=6;  //Roll
int elevatorPin=9;  //Pitch
int gearPin=10;  //Gear

const int STATE_THROTTLE_LOW = 0;
const int STATE_THROTTLE_LOW_RUDDER_LOW = 1;
const int STATE_THROTTLE_LOW_RUDDER_HIGH = 2;
const int STATE_ALL_IDLE = 3;

int state=STATE_THROTTLE_LOW;

int command; // tasta de control

void generateDefaultPWM(int pin){
  digitalWrite(pin,HIGH);
  delayMicroseconds(1500);
  digitalWrite(pin,LOW);
```

```
  delayMicroseconds(500);
}

void generateLowDutyCyclePWM(int pin){
  digitalWrite(pin,HIGH);
  delayMicroseconds(1000);
  digitalWrite(pin,LOW);
  delayMicroseconds(1000);
}

void generateHighDutyCyclePWM(int pin){
  digitalWrite(pin,HIGH);
  delayMicroseconds(2000);
  digitalWrite(pin,LOW);
  delayMicroseconds(0);
}


void setup() {
  pinMode(throttlePin,OUTPUT);
  pinMode(rudderPin,OUTPUT);
  pinMode(aileronPin,OUTPUT);
  pinMode(elevatorPin,OUTPUT);
  pinMode(gearPin,OUTPUT);
  Serial.begin(9600);
  Serial.println("Apasa");
  state=STATE_THROTTLE_LOW;
  Serial.println(state);
}

void loop()  {
  switch (state) {
  case STATE_THROTTLE_LOW:
    generateLowDutyCyclePWM(throttlePin);
    generateDefaultPWM(rudderPin);
    generateDefaultPWM(aileronPin);
    generateDefaultPWM(elevatorPin);
    generateDefaultPWM(gearPin);
    delayMicroseconds(10000);
    break;
  case STATE_THROTTLE_LOW_RUDDER_LOW:
    generateLowDutyCyclePWM(throttlePin);
    generateLowDutyCyclePWM(rudderPin);
    generateDefaultPWM(aileronPin);
    generateDefaultPWM(elevatorPin);
    generateDefaultPWM(gearPin);
    delayMicroseconds(10000);
    break;
  case STATE_THROTTLE_LOW_RUDDER_HIGH:
    generateLowDutyCyclePWM(throttlePin);
    generateHighDutyCyclePWM(rudderPin);
    generateDefaultPWM(aileronPin);
    generateDefaultPWM(elevatorPin);
    generateDefaultPWM(gearPin);
    delayMicroseconds(10000);
    break;
  case STATE_ALL_IDLE:
```

```
   generateDefaultPWM(throttlePin);
   generateDefaultPWM(rudderPin);
   generateDefaultPWM(aileronPin);
   generateDefaultPWM(elevatorPin);
   generateDefaultPWM(gearPin);
   delayMicroseconds(10000);
   break;
                                    }

  if (Serial.available() > 0){
   command=Serial.read();
   switch (command){
   case 97: //a
     state = STATE_THROTTLE_LOW_RUDDER_LOW;
     break;
   case 100: //d
     state = STATE_THROTTLE_LOW_RUDDER_HIGH;
     break;
   case 115: //s
     state = STATE_THROTTLE_LOW;
     break;
   case 98: //b
      state = STATE_ALL_IDLE;
     break;
    }
    Serial.println(state);
    Serial.println(x);
  }
}
```

Program pentru controlul quadcopterului

Programul de controlare va avea înclus și rutina de calibrare. În plus, se va adăuga alte funcții pentru controlul quadcopterului.

```
int x=0  ;                                          //pasul de modificare

void generateModifyPWM     (int pin)  {             //Funcția pentru generarea semanlulu PWM cu factor de umplere variabil
            digitalWrite(pin,HIGH);
            delayMicroseconds(1000+x);              // Crește timpul de HIGH
            digitalWrite(pin,LOW);
            delayMicroseconds(1000-x);              //Scade timpul de LOW
                                              }


void loop()  {
            switch (state) {
                        case STATE_MODIFY_THROTTLE:          //Starea temporară cu semnale PWM variabile
                           generateModifyPWM(throttlePin);
                           generateDefaultPWM(rudderPin);
                           generateDefaultPWM(aileronPin);
                           generateDefaultPWM(elevatorPin);
                           generateDefaultPWM(gearPin);
                           delayMicroseconds(10000);
              break;
            }
```

```
if (mySerial.available() > 0){
   command=mySerial.read();
   switch (command){
          case 99:                                        //c  = increment PWM
                    x=x+25;
                  state=STATE_MODIFY_THROTTLE;
             break;
           case 120:                                      //x  = decrement PWM
                     x=x-25;
                  state=STATE_MODIFY_THROTTLE;
            break;
 }
```

ANEXA 2

Programul pentru pagina web

```
<html>
<head>
          <title>Licenta 2016</title>
          <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.2/jquery.min.js"></script>
          <style type="text/css">

.centered {
   width: 65%;
   margin: 0 auto;
   text-align: center;
}

header > h3 {
   font-size: 36px;
   font-family: serif;
   color: #0D67AF;
   text-shadow: 0px 1px 23px rgba(0,0,0,0.3);
   margin-bottom: -10px;
   margin-top: 30px;
}

header > h4 {
   font-size: 23px;
   font-style: italic;
   color: darkgray;
   font-family: sans;
}
.btn {
   display: inline-block;
   margin-bottom: 0;
   font-weight: 400;
   text-align: center;
   vertical-align: middle;
```

```
    cursor: pointer;
    color: white;
    background-image: none;
    background-color: #5BC0DE;
    border: 1px solid transparent;
    white-space: nowrap;
    padding: 6px 12px;
    margin-top: 36px;
    font-size: 14px;
    line-height: 1.42857143;
    border-radius: 4px;
    -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
}


#stream > img {
    width: 710px;
    height: auto;
    border: 4px solid #5BC0DE;
    margin-top: 20px;
    border-radius: 4px;
}



            </style>


</head>
<body>
            <header class="centered">
                        <h3>Quadcopter Stream</h3>
                        <h4>Arlind Lika.....Licenta 2016</h4>
                        <hr>
            </header>
            <div id="stream" class="centered" hidden>
                        <img src="{{ url_for('video_feed') }}">
            </div>
            <div class="centered">
                        <button id="showStream" class="btn">Toggle stream</button>
            </div>

</body>


<script>

$(function () {
            $("#showStream").click(function () {
                        $("#stream").slideToggle();
            });


});


</script>

</html>
```

Program video-streaming

```
#!/usr/bin/env python
from flask import Flask, render_template, Response

from camera_pi import Camera

app = Flask(__name__)


@app.route('/')
def index():
    return render_template('index.html')


def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')


@app.route('/video_feed')
def video_feed():
    return Response(gen(Camera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')


if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True, threaded=True)
```

Program pentru generarea frame-urilor

```
import time
import io
import threading
import picamera


class Camera(object):
    thread = None
    frame = None
    last_access = 0

    def initialize(self):
        if Camera.thread is None:

            Camera.thread = threading.Thread(target=self._thread)
            Camera.thread.start()


            while self.frame is None:
                time.sleep(0)

    def get_frame(self):
```

```
    Camera.last_access = time.time()
    self.initialize()
    return self.frame

@classmethod
def _thread(cls):
    with picamera.PiCamera() as camera:

        camera.resolution = (640, 480)
        camera.hflip = True
        camera.vflip = True
        camera.framerate = 30


        camera.start_preview()
        time.sleep(2)

        stream = io.BytesIO()
        for foo in camera.capture_continuous(stream, 'jpeg',
                                             use_video_port=True):

            stream.seek(0)
            cls.frame = stream.read()


            stream.seek(0)
            stream.truncate()


             if time.time() - cls.last_access > 10:
                break
    cls.thread = None
```