Drona cu sistem de echilibrare automat

Introducere

Proiectul **Drone Stabilizer** reprezintă construcția unei drone (quadcopter), bazată pe motoare cu perii (brushed), capabilă să își mențină echilibrul în aer în mod autonom. Nucleul proiectului din punct de vedere software este implementarea unui algoritm de control de tip **PID (Proportional-Integral-Derivative)** care procesează datele de la un senzor inerțial pentru a ajusta viteza motoarelor în timp real.

Functionalități principale:

- stabilizarea automată pe axele Pitch, Roll și Yaw folosind giroscopul și accelerometrul
- recepția comenzilor de zbor (throttle, direcție) prin protocol wireless de 2.4GHz
- controlul turației celor 4 motoare prin semnale PWM transmise către punțile H
- monitorizarea orientării dronei în timp real
- gestionarea energiei pentru a asigura stabilitatea electronicii în timpul căderilor de tensiune ale motoarelor

Laboratoare folosite: UART (Lab 1), Intreruperi (Lab 2), Timere/PWM (Lab 3), SPI (Lab 5 - NRF24), I2C (Lab 6 - MPU).

Descriere generală

Schema Bloc

Module si interactiuni:

Drona:

- **ATmega328PB:** unitatea centrală de procesare. Citește datele de orientare de la MPU6050, calculează unghiurile de înclinare, rulează algoritmii PID de stabilizare și ajustează viteza motoarelor în timp real prin semnale PWM. Primește comenzi de la distanță prin modulul NRF24 și gestionează comenziile primite de la remote controller.
- **MPU6050:** senzorul IMU (Inertial Measurement Unit) care combină un giroscop pe 3 axe cu un accelerometru pe 3 axe. Giroscopul măsoară viteza unghiulară (grade/secundă), iar accelerometrul măsoară forțele liniare, inclusiv gravitația. La pornire, MCU-ul efectuează o calibrare pe 1000 de eșantioane pentru a elimina offset-urile statice. În zbor, datele brute sunt combinate printr-un filtru

complementar: 96% giroscop (precizie pe termen scurt) și 4% accelerometru (corecție drift pe termen lung), rezultând unghiurile de roll, pitch și yaw. Comunicare prin I2C. Timer2 genereaza intreruperi o data la o milisecunda, cand se ruleaza PID-ul pentru a se corecta directia dronei.

- **NRF24:** modulul de comunicație wireless care operează pe banda de 2.4GHz. Drona funcționează în modul receptor (listening), așteptând comenzi de la un remote controller. Prima comandă obligatorie este m — aceasta declanșează calibrarea MPU și permite pornirea motoarelor. Ulterior, avem comenzi de urcare, coborare, stanga, dreapta, fata si spate, dar si o comanda pentru oprire instanta a elicelor. Comunicare cu MCU-ul prin SPI.
- **Drivere MX1616H (x2):** drivere de motor cu punte H dublă, fiecare controlând câte două motoare brushed. Primesc semnalul PWM de la MCU prin registrele OCR (Output Compare Register) ale timerelor hardware (Timer0 pentru motoarele stânga, Timer1 pentru motoarele dreapta) și translatează comenzile digitale în tensiune aplicată direct pe bobinele motoarelor. Direcția de rotație (CW/CCW) este setată la inițializare prin pinii de direcție ai porturilor C și D.
- **4 motoare brushed:** dispuse în configurație quad — față-stânga (CCW), față-dreapta (CW), spate-stânga (CW), spate-dreapta (CCW). Sensurile alternante de rotație sunt esențiale pentru anularea momentului girroscopic și controlul yaw-ului. Viteza fiecărui motor este ajustată independent de algoritmul de stabilizare, câte 15 unități PWM per pas, clamped între 0 și 255.
- **LiPo 1S 3.7V:** alimentează direct driverele MX1616H și motoarele. Tensiunea de 3.7V este ridicată la 5V prin buck converter pentru alimentarea MCU-ului, MPU6050 și NRF24.

Telecomanda:

- **Arduino Nano V3:** joaca rolul de **Remote controller**, care este conectat la laptop si primeste comenziile din laptop prin protocolul **UART**, care de asemenea este conectata la un modul NRF24.

Hardware Design

Lista de piese

Nr.	Componentă	Cantitate	Rol în proiect
1	**ATmega328PB**	x1	Flight Controller — unitatea centrală de procesare
2	**MPU6050 — GY-521**	x1	IMU cu giroscop și accelerometru pe 3 axe
3	**MX1616H**	x2	Driver punte H — controlul motoarelor brushed
4	**Motor brushed 820**	x4	Propulsie — configurație quad
5	**NRF24**	x2	Transreceptor wireless 2.4GHz — primire, transmitere comenzi
6	**Buck converter**	x1	Ridicător tensiune 3.7V → 5V pentru partea logică
7	**LiPo 1S 3.7V 500mAh 95C**	x1	Sursă principală de energie
8	**Arduino Uno NANO**	x1	Remote controller

Schema electrica

Pini folositi

Componentă	Pin componentă	Pin ATmega328PB	Explicație
MPU6050	VCC	5V	Modulul este alimentat la 5V
MPU6050	GND	GND	Masă comună
MPU6050	SDA	PC4 / SDA	Pin standard pentru date I2C
MPU6050	SCL	PC5 / SCL	Pin standard pentru clock I2C
NRF24	VCC	3.3V	Modulul necesită alimentare la 3.3V
NRF24	GND	GND	Masă comună
NRF24	CE	PD3	Chip Enable — activare modul RX/TX
NRF24	CSN	PD7	Chip Select — selecție SPI
NRF24	SCK	PB5 / SCK	Clock SPI
NRF24	MOSI	PB3 / MOSI	Date SPI master → slave
NRF24	MISO	PB4 / MISO	Date SPI slave → master
Motor față-stânga (CCW)	IN1	PD2	Control direcție — Driver MX1616H
Motor față-stânga (CCW)	IN2 / PWM	PD5	Semnal PWM — viteză motor
Motor față-dreapta (CW)	IN1	PC0	Control direcție — Driver MX1616H
Motor față-dreapta (CW)	IN2 / PWM	PB1	Semnal PWM — viteză motor
Motor spate-stânga (CW)	IN1	PD4	Control direcție — Driver MX1616H
Motor spate-stânga (CW)	IN2 / PWM	PD6	Semnal PWM — viteză motor
Motor spate-dreapta (CCW)	IN1	PC1	Control direcție — Driver MX1616H
Motor spate-dreapta (CCW)	IN2 / PWM	PB2	Semnal PWM — viteză motor
Buck converter	VIN	VBAT (LiPo)	Intrare 3.7V de la baterie
Buck converter	VOUT	VCC placă	Ieșire 5V pentru partea logică

Pinii PC4 și PC5 au fost aleși pentru MPU6050 deoarece sunt pinii dedicați pentru I2C pe ATmega328PB. Comunicarea se face la 400kHz (fast mode) pentru a minimiza latența citirii datelor IMU.

Pentru NRF24L01+ am folosit pinii hardware SPI (PB3, PB4, PB5) deoarece SPI hardware este semnificativ mai rapid decât bit-banging. CE pe PD3 și CSN pe PD7 sunt pini digitali obișnuiți, aleși să nu intre în conflict cu timerele folosite pentru motoare.

Timer2 este configurat în mod CTC cu un prescaler de 64 și OCR2A = 249, ceea ce generează o întrerupere la fiecare 1ms. În ISR-ul acestui timer se incrementează contorul checkMpu. În bucla principală, la fiecare 50 de incrementări (adică la fiecare 50ms) se citesc datele de la MPU6050, se calculează unghiurile prin filtrul complementar și se rulează cei 3 controlleri PID. Timer2 a fost ales pentru această sarcină tocmai pentru că Timer0 și Timer1 sunt ocupați cu PWM-ul motoarelor.

Timer0 este configurat în mod Fast PWM cu prescaler 8, generând semnal PWM pe OC0A (PD6) și OC0B (PD5) — folosiți pentru motoarele din stânga. Registrele OCR0A și OCR0B controlează direct ciclul de lucru, adică viteza fiecărui motor.

Timer1 este configurat similar în mod Fast PWM pe 8 biți cu prescaler 64, generând semnal PWM pe OC1A (PB1) și OC1B (PB2) — folosiți pentru motoarele din dreapta. Registrele OCR1AL și OCR1BL sunt cele scrise de algoritmul de stabilizare la fiecare iterație PID.

Direcția de rotație a fiecărui motor este setată o singură dată la inițializare prin pinii de direcție ai driverelor MX1616H, nu prin PWM. Algoritmul PID modifică doar viteza (registrele OCR), nu și direcția — motoarele brushed pe o dronă nu își inversează sensul în zbor normal.

Software

Am implementat mai multe functionalitati pentru drona si pentru sistemul de control radio.

Drona

- drona detecteaza inclinarea in timp real prin intermediul senzorului MPU6050 si corecteaza automat pozitia prin controlul individual al celor 4 motoare
- la pornire, motoarele cresc viteza proportional de la 0 la valorile de hover (227/235/245/255) astfel incat drona sa ramana echilibrata pe toata durata accelerarii
- daca drona primeste comanda 'x', toate motoarele sunt oprite imediat indiferent de starea curenta
- sistemul de control ruleaza la 100Hz (o iteratie la fiecare 10ms), asigurand corectii rapide ale pozitiei
- fiecare motor are un trim hardware individual care compenseaza diferentele fizice dintre motoare

Telecomanda

- telecomanda trimite comenzi prin NRF24L01 catre drona sub forma de caractere individuale
- comanda 'm' declanseaza calibrarea MPU6050 (500 de esantioane) dupa care drona este gata de zbor
- comenzile 'k' si 'j' cresc sau scad throttle-ul in trepte de 4 unitati
- comenzile 'w', 'a', 's', 'd' aplica un trim directional pe motoare cu revenire automata la pozitia neutra dupa 200ms de la ultima comanda directionala
- comanda 'x' opreste imediat toate motoarele si reseteaza throttle-ul la 0

Biblioteci si Headere folosite

- **Wire.h** - biblioteca Arduino folosita pentru initializarea comunicatiei I2C cu senzorul MPU6050
- **SPI.h** - biblioteca Arduino folosita pentru comunicatia SPI cu modulul radio NRF24L01
- **nrf24/RF24** - biblioteca clasica folosita pentru initializarea, trimiterea si receptia de informatii prin modulul radio NRF24L01
- **jrowberg/MPU6050** - biblioteca folosita pentru initializarea senzorului inertial si citirea datelor brute de acceleratie si giroscop prin getMotion6()
- **stdint.h** - folosita pentru tipuri de date cu dimensiunea in biti specificata (ex: int32_t, uint8_t)
- **avr/interrupt.h** - folosita pentru definirea ISR-urilor si controlul intreruperilor hardware

Element inedit

Drona foloseste un controler PID implementat exclusiv cu aritmetica in virgula fixa (fixed-point integer), eliminand complet operatiile float din bucla de control. Pe ATmega328P, care nu dispune de FPU hardware, fiecare operatie float consuma 20-100 de cicli de ceas, in timp ce operatiile pe int32_t consuma 2-4 cicli. Aceasta optimizare permite rularea buclei de control la 100Hz in loc de 20Hz, imbunatatind semnificativ stabilitatea dronei.

O alta noutate fata de implementarile clasice este rampa proportionala la pornire: fiecare motor porneste de la o valoare mica si creste cu o rata diferita, calculata astfel incat toate cele 4 motoare sa ajunga la valoarea lor de hover exact simultan, mentinand drona echilibrata pe toata durata accelerarii.

Functionalitati din laborator

- folosesc Timer2 in mod CTC cu prescaler 64 pentru a genera o intrerupere la fiecare 1ms, implementand un systick global folosit pentru masurarea intervalului dintre iteratiile buclei de control
- PWM-ul motoarelor este implementat manual folosind Timer0 (Fast PWM 8-bit pe OCR0A si OCR0B) si Timer1 (Fast PWM 8-bit pe OCR1AL si OCR1BL)
- directia de rotatie a fiecarui motor (CW/CCW) este setata direct prin registrii DDR si PORT la nivel hardware
- modulele folosite comunica prin SPI (NRF24L01) si I2C (MPU6050)
- dt-ul real dintre iteratii este masurat folosind contorul incrementat de ISR, astfel PID-ul este precis chiar daca bucla variaza ca durata

Scheletul proiectului

Proiectul este structurat pe 4 niveluri: main.cpp gestioneaza bucla principala, comenzile radio si masurarea timpului; Drone.cpp orchestreaza logica de zbor, mixarea motoarelor si apelul PID; Engine.cpp abstractizeaza fiecare motor individual cu suport pentru directii diferite de rotatie; PID.cpp implementeaza controlul in feedback cu termenii P, I si D.

Interactiunea dintre module: ISR-ul Timer2 incrementeaza un contor la fiecare 1ms; bucla principala verifica daca au trecut 10ms si apeleaza computeAngles() urmat de stabilize(); stabilize() ruleaza PID pentru roll si pitch, aplica trimurile hardware si trimite valorile finale la fiecare motor prin setSpeed().

Mai multe detalii sunt disponibile in codul sursa al proiectului.

Demo video

Un scurt demo in care prezint functionalitatile proiectului.

7ig5vDZfT6g

Calibrare senzoristica

- pentru a obtine valori precise de roll si pitch, se realizeaza o calibrare automata la pornire prin medierea a 500 de esantioane cu drona complet nemiscata pe o suprafata plana; valorile mediate reprezinta offset-ul sistematic al senzorului si sunt scazute din toate masuratorile ulterioare
- pe axa az se scade suplimentar valoarea de 1.0f reprezentand acceleratia gravitationala, astfel incat axa verticala sa fie referentiata corect
- filtrul complementar combina giroscopul (98%) cu accelerometrul (2%) pentru a elimina drift-ul pe termen lung al giroscopului si zgomotul accelerometrului
- antenele NRF24L01 sunt initializate in modul RF24_PA_LOW pentru a reduce consumul si interferentele la distante scurte de test

Optimizari

- PID-ul foloseste exclusiv aritmetica int32_t cu gainuri scalate cu factorul 100, eliminand toate operatiile float din bucla critica de control
- filtrul complementar foloseste aproximarea ay/az in loc de atan2() si sqrt(), eliminand functiile transcendentale costisitoare pe AVR
- derivata PID este calculata pe masurare si nu pe eroare, prevenind spike-urile la schimbarea brusca a setpoint-ului
- integratorului i se aplica anti-windup prin limitare la +/-5000, prevenind acumularea excesiva cand motoarele sunt saturate
- trimurile hardware per motor sunt aplicate inaintea corectiei PID, astfel incat PID-ul opereaza in jurul unui punct de echilibru real si nu trebuie sa compenseze dezechilibre mecanice

Rezultate Obtinute

Am obtinut o drona capabila sa decoleze si sa zboare controlata prin comenzi radio, cu un sistem de corectie in timp real bazat pe datele furnizate de senzorul inertial MPU6050. Comunicatia radio functioneaza stabil, motoarele raspund corect la comenzi iar filtrul complementar produce valori corecte de roll si pitch validate prin Serial.

Nu am reusit sa finalizez calibrarea gainurilor PID pentru un zbor complet stabil. PID tuning-ul este un proces dificil in absenta unei structuri rigide de testare: fara un cadru care sa tina drona pe loc si sa permita observarea comportamentului la fiecare modificare de gain, este aproape imposibil sa izolezi efectul unui singur parametru. Principalele bottleneck-uri intampinate au fost: motoarele nu sunt identice si introduc dezechilibre greu de compensat doar software, vibratiile motoarelor introduc zgomot in datele IMU afectand termenul D al PID-ului, iar fiecare test necesita resetarea intregului sistem ceea ce ingreuneaza iteratia rapida asupra gainurilor.

Implementarea software actuala este functionala ca platforma, iar o posibila imbunatatire ar putea aduce un zbor complet autonom stabil. As putea spune ca am creat o platforma hardware capabila

care ar putea fi extinsa cu filtre mai avansate (ex. Kalman), autotunare PID sau chiar control de altitudine prin barometru, daca s-ar investi timp suplimentar in calibrare si testare.

Concluzii

Proiectul a fost foarte antrenant si am invatat foarte multe lucruri noi pe parcursul acestuia. A fost prima mea experienta cu constructia unui sistem embedded complex cu mai multe module care interactioneaza in timp real, dar si prima incercare de a implementa un controler PID pe hardware real, unde am realizat ca diferenta dintre teorie si practica este semnificativa.

Timpul a fost destul de limitat, iar PID tuning-ul s-a dovedit a fi cel mai mare obstacol al proiectului. Fara o structura de testare dedicata si fara motoare identice, gasirea gainurilor corecte devine un proces iterativ foarte lent. Daca as relua proiectul, as investi mai mult timp in alegerea unor motoare mai bine matchuite si as construi un cadru de testare fix inainte de a incerca primul zbor liber.

In ciuda faptului ca nu am atins zborul complet stabil, consider ca am construit o platforma hardware si software solida, pe care o inteleg in profunzime, si care poate fi imbunatatita in continuare.

Download

GitHub Repo: https://github.com/stefan18-ux/Drone

Jurnal

Etapa 1 de Hardware

- Am lipit pe o placa de prototipare buck converterul si driverele ca sa aibe gnd si vcc comun de la baterie.
- Am lipit pini pe o placa de prototipare pe care am bagat-o in placuta pentru a putea lipi in continuare fire.
- Am terminat de lipit tot ce era acolo dupa aproximativ 30 ore oneste de munca.
- Am reusit sa ansamblez drona si sa cuprind tot inauntrul ei.
- Momentan am reusit sa setez elicele in sensurile bune, deci se ridica de la sol, dar nu pentru mult timp, trebuie reglata software mult.


WARNING
WARNING
WARNING
WARNING
FUN

Bibliografie/Resurse

Resurse hardware

- **ATmega328PB Xplained Mini** documentatie Microchip: https://www.microchip.com/en-us/development-tool/ATMEGA328PB-XMINI
- **Modul GY-521 (MPU6050)**: https://sigmanortec.ro/Modul-giroscopic-si-accelerometru-3-axe-GY-521-p126016326
- **Driver MX1616H**: De la Victor din laborator
- **Motor coreless 8x20mm 3.7V**: https://sigmanortec.ro/en/mini-coreless-motor-20x85-1mm-shaft-50000rpm-37v
- **Modul NRF24**: https://www.sigmanortec.ro/en/nrf24l01-24ghz-wireless-transceiver-module
- **Modul ridicător tensiune 3.7V → 5V**: https://sigmanortec.ro/en/step-up-modules
- **Baterie LiPo 3.7V 95C 500mah**: https://www.emag.ro/baterie-r-line-gens-ace-tattu-500-mah-3-7-v-95-c-multicolor-taa5001s95jsl/pd/DP84FRMBM/

Resurse software / scheme

- EasyEDA - realizare schema electrica: https://easyeda.com/
- Arduino - citire accurate a sezorilor: https://forum.arduino.cc/t/mpu6050-gyroscope-readings-are-drifting/1113217/10

Export to PDF