Projektiranje kvadkoptera sa sustavom upravljanja na daljinu

Shafi, Suhaib

Istarsko veleučilište
Università Istriana
di scienze applicate

Suhaib Shafi

PROJEKTIRANJE KVADKOPTERA SA SUSTAVOM UPRAVLJANJA NA DALJINU

Završni rad

Pula, srpanj 2024.

Istarsko veleučilište
Università Istriana
di scienze applicate

Suhaib Shafi

PROJEKTIRANJE KVADKOPTERA SA SUSTAVOM UPRAVLJANJA NA DALJINU

Završni rad

JMBAG: 0233009069, redovni student

Studijski smjer: Prijediplomski stručni studij Mehatronike

Predmet: Projektiranje ugrađenih računalnih sustava

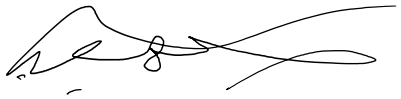
Mentor: Deni Vale, mag. phys., pred.

Pula, srpanj 2024.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Suhaib Shafi, kandidat za prvostupnika inženjera mehatronike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

U Puli, _____ godine

Potpis: 

IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja, Suhaib Shafi, dajem odobrenje Istarskom veleučilištu – Università Istriana di scienze applicate, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom „Projektiranje kvadkoptera sa sustavom upravljanja na daljinu“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama. Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ godine.

Potpis: 

Sažetak

U ovom radu predstavljen je projekt dizajna i izrade kvadkoptera s naglaskom na samostalnu konstrukciju. Detaljno su opisani svi implementirani elementi, uključujući odabir i funkcionalnost svake komponente. Rad detaljno razmatra programskih algoritme koji omogućavaju kvadkopteru funkcionalnosti drona. Analiziran je trošak izrade jednog kvadkoptera, uključujući kontroler i neophodne komponente, te je specificiran broj pojedinih dijelova potreban za sklapanje. Također, opisani su mogući dodaci koji mogu proširiti funkcionalnost drona. U radu su razmotrene metode za regulaciju brzine motora te postupci programiranja motora kako bi se osigurala precizan odziv na upravljačke signale iz kontrolera. Na kraju, istražene su mogućnosti za unaprjeđenje performansi drona i njegove primjene u različitim kontekstima.

Ključne riječi: projektiranje ugrađenih računalnih sustava, kvadkopter, Arduino IDE, daljinsko upravljanje dronom, programiranje

Summary

This thesis presents a quadcopter design and construction project with an emphasis on independent assembly. All implemented elements are detailed, including the selection and functionality of each component. The work thoroughly examines the programming algorithms that enable the drone functionalities of the quadcopter. The cost of manufacturing a single quadcopter, including the controller and necessary components, has been analyzed, and the number of individual parts required for assembly has been specified. Additionally, possible enhancements that could expand the drone's functionality are described. Methods for motor speed regulation and procedures for programming the motors have been considered to ensure precise responsiveness to control signals from the controller. Finally, the possibilities for enhancing the drone's performance and its applications in various contexts have been explored.

Keywords: design of embedded computer systems, quadcopter, Arduino UNO, remote control of the drone, programming

Sadržaj

Popis oznaka i kratica

Oznaka	Opis	Jedinica
U	Napon	Volt (V)
I	Struja	Amper (A)
M	Masa	Kilogram (kg)
V	Brzina	Metar u sekundi (ms^{-1})
G	Gravitacijska akceleracija na površini	Metar po kvadratnoj sekundi (ms^{-2})
A	Ubrzanje	Metar po kvadratnoj sekundi (ms^{-2})
T	Uzgon	Newton (N)
τ	Moment sile	Newton metar (Nm)
I	Moment inercije	Kilogram metar kvadratni (kgm^2)
A	Kutno ubrzanje	Radian po kvadratnoj sekundi (rad/s^2)
F	Sila	Newton (N)
P	Tlak	Pascal (Pa)
R	Kovarijacijska matrica pogreške	Bez jedinice
Kp	Proporcionalni koeficijent	Bez jedinice
Ki	Integralni koeficijent	Bez jedinice
Kd	Derivativni koeficijent	Bez jedinice
t	Vrijeme	Sekunda (s)

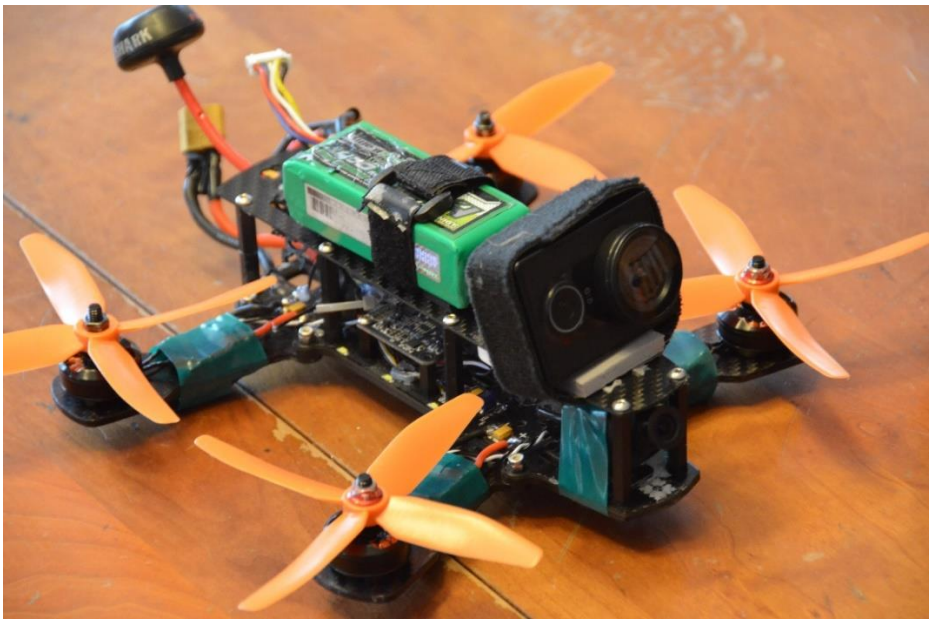
KRATICA	OPIS
GND	Uzemljenje
UART	Univerzalni asinkroni prijemnik-odašiljač
PWM	Modulacija širine impulsa
SPI	Serijsko periferno sučelje
DC	Istosmjerna struja *(engl. direct current)*
VAC	Napon napajanja izmjenične struje
VDC	Napon napajanja istosmjerne struje
ESC	Električni regulator brzine
PCB	Tiskana ploča
DMP	Digitalni procesor pokret
I2C	Međuspojna integrirana komunikacija
PID	proporcionalno-integralno-derivativni
PPM	modulacija pozicije impulsa
GPS	globalni pozicijski sustav

1. Uvod

U suvremenom doba, kvadkopteri, dronovi s četiri propelera, istaknuli su se kao iznimno popularni alati u različitim primjenama, od hobističkih do komercijalnih. Kvadkopteri se koriste za inspekciju infrastrukture kao što su ceste, mostovi, električne vodove, cijevi, dimnjaka i slično. Mogu brzo i sigurno doći do mjesta koja su teško dostupna čovjeku, što štedi vrijeme i resurse u usporedbi s tradicionalnim metodama inspekcije. Koriste u fotografiji i filmskoj industriji za snimanje iz zraka. Oni omogućuju snimanje iz perspektive koja nije moguća iz drugih kutova. Kvadkopteri se koriste u hitnim situacijama za pretraživanje terena, dostavu opreme i pružanje prve pomoći na mjestima gdje je pristup otežan. U poljoprivredi, kvadkopteri se koriste za nadzor usjeva, primjenu pesticida, ili čak za praćenje stanja tla. Kvadkopteri se koriste za istraživanje divljih životinja, nadgledanje ekosustava, te istraživanje terena u teško pristupačnim područjima. Primjer komercijalnog kvadkoptera koji služi istraživanju terena nalazi se na slici 1.

Slika 1. Primjer komercijalnog kvadkoptera. Preuzeto s https://www.pexels.com/photo/quadcopter-flying-on-the-skey-1034812/

Neke tvrtke razmatraju upotrebu kvadkoptera za dostavu paketa u urbanim područjima kako bi smanjile vrijeme isporuke. Kvadkopteri su popularni i kao igračke za zabavu, a postoji i rastuća scena natjecanja u letenju dronova. Primjer takvog hobističkog kvadkoptera nalazi se na slici 2.



Slika 2. Primjer hobističkog kvadkoptera. https://www.netokracija.com/utrke-dronova-148390.

Ovaj rad cilja na samostalno projektiranje i izradu kvadkoptera, s ciljem stvaranja učinkovitog i pristupačnog modela koji se može sastaviti i prilagoditi u inženjerskom okruženju. Cilj projekta nije samo razviti osnovni kvadkopter, već stvoriti platformu koja omogućava inženjerima da implementiraju različite tehnologije, nadogradnje i prilagodbe, čime se povećava funkcionalnost i prilagodljivost u stvarnim operativnim uvjetima. Rad se koncentrira na pojednostavljenje dizajna i optimizaciju troškova izrade, čineći ovu tehnologiju dostupnom širokom spektru profesionalaca u tehnološkom sektoru. Metodologija ovog projekta uključuje detaljnu selekciju i evaluaciju komponenata koje su financijski isplative i široko dostupne, te kroz kombinaciju teorijskog znanja i praktičnih eksperimenata razrađuje pristup konstrukciji kvadkoptera.

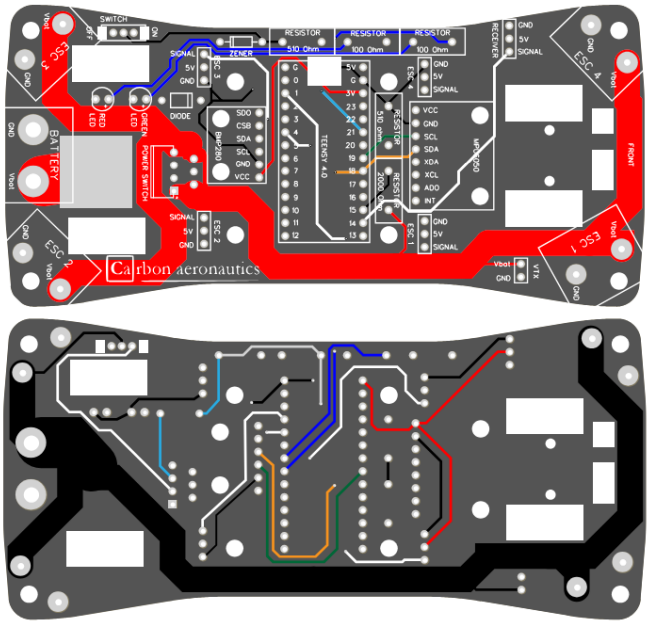
Struktura rada podijeljena je u više poglavlja koja pokrivaju sve od osnovne koncepcije, detaljnog opisa komponenata i sustava, do završnog sklapanja i provođenja testiranja. U drugom poglavlju detaljno su opisane komponente korištene u izradi kvadkoptera. U trećem poglavlju opisan je dizajn pojedinih sustava i njihova uloga u letenju. U idućem poglavlju objašnjeno je letenje kvadkoptera s fizičkog aspekta te su opisane metode ispitivanja performansi letenja drona. U petom poglavlju navedene su korištene biblioteke, opisano je konfiguriranje mikroupravljača te su objašnjeni ključni dijelovi izvornog koda. U šestom poglavlju prikazani su rezultati navedenih ispitivanja. U sedmom poglavlju prodiskutirani su problemi na koje smo naišli prilikom izrade drona i testiranja, njihova rješenja, kao i mogućnosti nadogradnje.

2. Komponente kvadkoptera

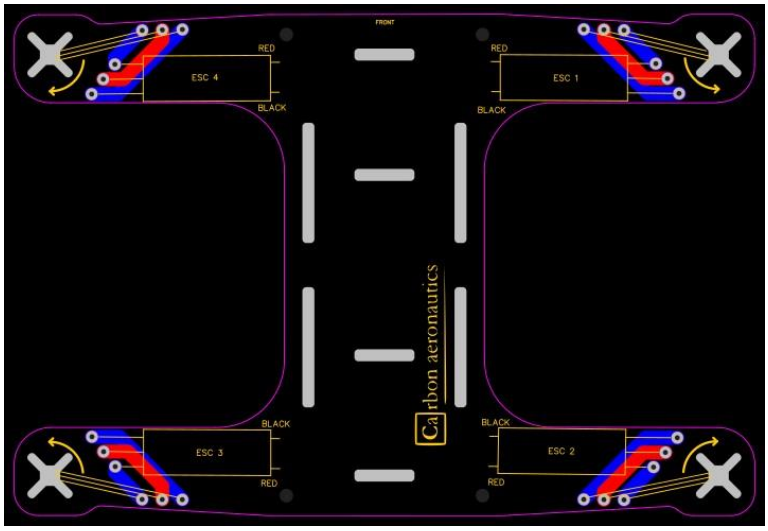
U ovom poglavlju opisane su ključne komponente kvadkoptera, te je pritom analizirana njihova funkcionalnost, tehničke karakteristike i doprinos ukupnoj funkcionalnosti letjelice.

2.1. Okviri tiskanih ploča

Tiskana pločica (*engl. printed circuit board* ili PCB), ključna je komponenta u mnogim elektroničkim uređajima, uključujući i kvadkoptere, jer služi kao platforma za mehaničko i električno povezivanje elektroničkih komponenti. Za potrebe ovog projekta dizajnirane su dvije posebne PCB ploče prikazane na slici 3. i 4. koje omogućuju integraciju svih komponenti u jedinstveno tijelo. Ove ploče sadrže unutarnje bakrene staze koje povezuju sve komponente, omogućavajući njihovu uspješnu međusobnu komunikaciju i napajanje. Dizajn ovih ploča preuzet je iz projekta otvorenog koda (engl. open source project), što je omogućilo korištenje provjerenih rješenja i prilagodbu prema specifičnim zahtjevima našeg kvadkoptera. U ovom projektu, okviri drona su napravljeni od PCB materijala kako bi se smanjila težina drona i omogućilo efikasno spajanje ostalih komponenta.



Slika 3. Gornji PCB.



Slika 4. Donji PCB.

Većina ključnih komponenata smještena na gornjoj PCB ploči, gdje su integrirani senzori, mikroupravljač i glavni čipovi, što omogućava centralizirano upravljanje i obradu podataka. Gornja ploča služi kao glavni komunikacijski čvor i distribucijski centar za signale i napajanje svih elektroničkih komponenata.

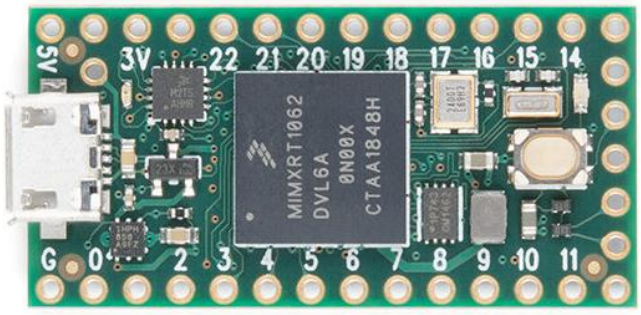
S druge strane, donja ploča je primarno namijenjena za smještaj baterije, motora i regulatora brzine motora. Ovaj raspored osigurava da teži elementi poput baterije budu bliže centru mase kvadkoptera, čime se poboljšava stabilnost leta. Također, regulatori brzine motora su smješteni na donjoj ploči kako bi se minimizirali električni smetnje na signalima koji se upućuju prema motorima. Također, konfiguriran je da podržava dodatne komponente poput kamera i različitih senzorskih modula, čime se omogućuje njihova integracija bez narušavanja aerodinamike ili dodatnog opterećenja na performanse letenja.

Ovaj dizajn dviju ploča omogućava efikasnu organizaciju i distribuciju funkcionalnih grupa komponenata, čime se optimizira performansa kvadkoptera i olakšava održavanje i eventualne nadogradnje sistema.

2.2. Mikroupravljač

Mikroupravljač koji se koristi za ovaj projekt je Teensy 4.0. Ovaj mikroupravljač je odabran zbog svoje visoke brzine i performansi, omogućavajući učinkovito rukovanje složenim proračunima potrebnim za stabilizaciju i upravljanje kvadkopterom. (“CarbonAeronautics” 2023) Teensy 4.0 dolazi s nizom značajki.

Teensy 4.0 ima CPU ARM Cortex-M7 koji radi na frekvenciji od 600 MHz. Memorija uključuje 1024 KB RAM-a i 2048 KB Flash-a. Uređaj ima 40 digitalnih ulaza/izlaza, 14 analognih ulaza i 2 analogna izlaza. USB podržava High-Speed prijenos podataka brzinom od 480 Mbit/s. Komunikacijski protokoli uključuju podršku za I2C, SPI i UART. Primjera navedenog mikroupravljača prikazan je na slici 5.



Slika 5. Primjer Teensy mikroupravljača. Preuzeto s https://www.electronics-lab.com/new-teensy-4-0-fastest-dev-board-powered-arm-cortex-m7/.

Mikroupravljač igra ključnu ulogu u upravljanju kvadkopterom, obavljajući niz funkcionalnosti koje omogućuju stabilan i precizan let. Prva i najvažnija funkcija mikroupravljača je obrada podataka sa senzora. Mikroupravljač kontinuirano prikuplja i obrađuje podatke iz žiroskopa, akcelerometra i barometra. Ovi podaci su ključni za održavanje stabilnosti kvadkoptera tijekom leta jer pružaju informacije o kutnim brzinama, ubrzanju i visini. Nakon što su podaci prikupljeni, mikroupravljač ih koristi za stabilizaciju leta i održavanje ravnoteže kvadkoptera. Precizna obrada ovih podataka omogućuje prilagodbu kontrole i održavanje stabilnog leta.

Druga važna funkcija mikroupravljača je kontrola motora. Mikroupravljač šalje modilacijo širene impulsa (*engl. pulse width modulation* ili PWM) signale elektroničkim regulatorima brzine (ESC) kako bi prilagodio brzinu vrtnje motora. Ovi signali određuju koliko brzo se motori vrte, što direktno utječe na kretanje kvadkoptera. Prilagodba brzine motora omogućuje kvadkopteru različite manevarske sposobnosti, uključujući podizanje, spuštanje, okretanje, te kretanje naprijed, nazad i bočno.

Mikroupravljač je također odgovoran za komunikaciju s prijemnikom. Prima signale iz prijemnika FS-iA6B, koji je povezan s daljinskim upravljačem FlySky FS-i6. Ovi signali uključuju naredbe za kretanje koje šalje operater. Nakon što mikroupravljač primi ove signale, obrađuje ih kako bi izvršio naredbe operatera, uključujući upravljanje smjerom, brzinom i stabilizacijom kvadkoptera.

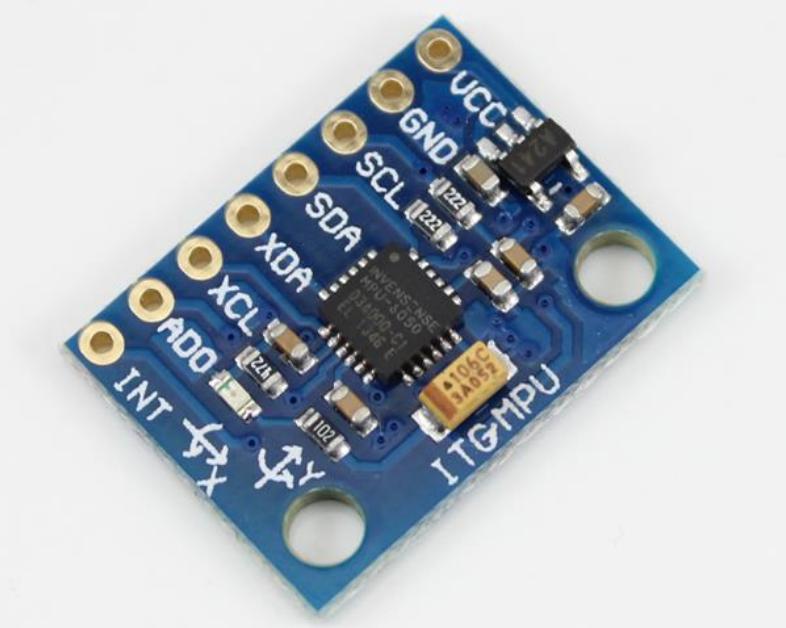
Za precizno upravljanje letom kvadkoptera, mikroupravljač koristi sofisticirane kontrolne algoritme, kao što je PID (Proporcionalno-Integralno-Derivativni) kontroler. PID kontroler omogućuje kontinuiranu prilagodbu kontrole leta na temelju trenutnih podataka sa senzora. Ovaj algoritam omogućuje preciznu kontrolu brzine motora, što je ključno za održavanje stabilnosti i kontrole tijekom cijelog leta. Mikroupravljač neprestano prilagođava brzinu motora na temelju informacija dobivenih od senzora, osiguravajući da kvadkopter može održavati stabilan i siguran let.

Ove funkcionalnosti mikroupravljača ključne su za osiguranje stabilnog, sigurnog i preciznog letenja kvadkoptera, omogućujući mu da uspješno obavlja sve zadatke koje mu je operater postavio.

2.3. Žiroskop i akcelerometar

GY-521 MPU-6050 je ključni senzor u kvadkopteru koji kombinira žiroskop i akcelerometar u jednom modulu. Ovaj senzor omogućuje precizno praćenje orijentacije i ubrzanja letjelice. Žiroskop mjeri kutnu brzinu oko X, Y i Z osi, što pomaže u održavanju stabilnosti kvadkoptera detektiranjem rotacija. Rasponi mjerenja za žiroskop su ±250, ±500, ±1000 i ±2000 °/s. Akcelerometar mjeri linearno ubrzanje duž X, Y i Z osi s rasponima mjerenja od ±2g, ±4g, ±8g i ±16g.

Osim žiroskopa i akcelerometra, MPU-6050 ima i digitalni procesor pokreta (engl. *Digital Motion Processor ili* DMP) koji može obraditi kompleksne proračune pokreta unutar samog čipa, smanjujući opterećenje mikroupravljača. Senzor koristi I2C komunikaciju za prijenos podataka do mikroupravljača, omogućujući brz i pouzdan prijenos informacija. Napaja se naponom od 3.3V do 5V. Kalibracija je potrebna prije upotrebe kako bi se osigurala točnost mjerenja. GY-521 MPU-6050 pruža ključne podatke za stabilizaciju i kontrolu kvadkoptera tijekom leta. Primjer GY-521 MPU-6050 akcelerometra i žiroskopa prikazan je na slici 6.



Slika 6. Primjer GY-521 MPU-6050. Preuzeto s https://www.iccfl.com/product_info.php?products_id=22763.

2.4. Elektromotori i propeleri

Elektromotori i propeleri ključni su komponenti kvadkoptera jer omogućuju potisak potreban za let i manevriranje. U ovom projektu korišteni su Happymodel EX1103 KV5000 motori bez četkica (*engl. brushless*). Ovi motori su odabrani zbog svoje učinkovitosti, pouzdanosti i male težine, što je ključno za održavanje niske mase

kvadkoptera i povećanje trajanja leta. Specifikacije spomenutih motora prikazane su u tabeli 1.

Tabela 1. Specifikacije Happymodel EX1103 KV5000 motora.

KV vrijednost	5000
Napon	2S (7.4 V)
Težina	3.5 g
Tip	bez četkica

Propeleri su odgovorni za stvaranje potiska potrebnog za let kvadkoptera. U ovom projektu koriste se propeleri kompatibilni s EX1103 motorima. Propeleri, ovisno o konfiguraciji, mogu se okretati u smjeru kazaljke na satu (*engl. clockwise* ili CW) ili pak u suprotnom smjeru (*engl. counterclockwise* ili CCW). Tip propelera korišten u ovom projektu je dvokraki i izrađen od plastike. Ovi propeleri pridonose smanjenju ukupne težine letjelice i osiguravaju optimalna aerodinamička svojstva.

Motori i propeleri zajedno stvaraju vertikalni potisak koji omogućuje kvadkopteru da se podiže, spušta i lebdi. Precizna kontrola brzine motora omogućava kvadkopteru da se kreće u svim smjerovima i rotira oko svoje osi. Motori bez četkica, kao što je EX1103, preferirani su zbog svoje visoke učinkovitosti, dugog vijeka trajanja i niskog održavanja. Primjer navedenih motora prikazan je na slici 7., a propelera na slici 8. Kombinacija ovih pažljivo odabranih motora i propelera osigurava da kvadkopter može postići optimalne performanse, uključujući stabilan let, dugotrajnost baterije i visoku upravljivost, što je ključno za uspjeh ovog projekta.

Slika 7. Primjer Happymodel EX1103 KV5000 motora. Preuzeto s https://www.xt-xinte.com/Happymodel-EX1103-1103-7000KV-2-4S-Brushless-Motor-for-Sailfly-X-Larva-X-Toothpick-RC-Racing-Drone-FPV-Models-p660498.html

Slika 8. Primjer Gemfan Hurricane propelera. Preuzeto s https://www.quadkopters.com/product/3-inch-or-less-props/gemfan-hurricane-4024-durable-2-blade-propeller/

2.5 Elektronički regulator brzine

Elektronički regulator brzine (*engl. electronic speed controller* ili ESC) ključna je komponenta koja kontrolira brzinu vrtnje motora u kvadkopteru. U ovom projektu koriste se niskonaponski Airplane A32-6A regulatori brzine. Njegove specifikacije prikazane su u tabeli 2. („vgoodrc“ , bez datuma).

Tabela 2. Specifikacije A32 Airplane ESC-Low voltage A32-6A

Stalna struja	6A
Vrhunac struje (10s)	10A BEC
Izlaz	5.5V/1.5A
Težina	5.5g (s kabelima)
Veličina	25x11mm
PWM	8-18KHz
Programabilnost	Da
Baterija	Lipo 2S

ESC pretvara signale iz mikroupravljača u promjenjive brzine vrtnje motora. Mikroupravljač šalje PWM signale ESC-u, koji zatim regulira količinu struje koja se isporučuje motorima. Na taj način, ESC omogućuje preciznu kontrolu nad brzinom vrtnje motora, što je ključno za stabilizaciju i manevriranje kvadkopterom. Na slici 9. prikazan je niskonaponski A32-6A ESC koji je korišten u izradi drona.

Slika 9. Niskonaponski ESC A32-6A regulator. Preuzeto s http://www.vgoodrc.com/en_product/details-41.html#goods_show

2.6. Baterije

U ovom projektu korištene su dvije različite baterije. Prva baterija koja je korištena bila je Beat LiPo baterija kapaciteta 1300mAh s kontinuiranim pražnjenjem od 35C pri naponu od 7.4V. Ova baterija teži 80 grama i namijenjena je dronovima. Međutim, pokazalo se da je preteška za ovaj kvadkopter, što je negativno utjecalo na performanse leta. Primjer takve baterije nalazi se na slici 10.

Slika 10. Beat LiPo baterija. Preuzeto s https://www.aliexpress.com/item/1005006102714322.html?spm=a2g0o.order_list.order_list_main.97.116218026aGIWp

Zbog prevelike težine prve baterije, odlučeno je kupiti drugu bateriju koja je specifična za airsoft oružje. Druga baterija je Specna Arms LiPo baterija s naponom od 7.4V i kapacitetom od 1300mAh, s kontinuiranim pražnjenjem od 15/30C. Ova baterija teži oko 40 grama, što je značajno manje od prve baterije. Također, priključak je zamijenjen na XT60 kako bi odgovarao potrebama kvadkoptera. Primjer navedene baterije prikazan je na slici 11.

Slika 11. Specna Arms LiPo baterija Preuzeto s https://www.maxairsoft.com/en/specna-arms-lipo-battery-11-1v-15c-1300mah-peq/p-5239.

Korištenjem lakše baterije značajno su poboljšane performanse kvadkoptera, posebno u pogledu trajanja leta i stabilnosti. Ova zamjena omogućila je duže letove i bolje upravljanje, čineći kvadkopter funkcionalnijim za različite zadatke.

2.7. Daljinski upravljač i prijemnik

2.7.1. Karakteristike FlySky FS-i6

Daljinski upravljač i prijemnik ključne su komponente kvadkoptera koje omogućuju operateru da upravlja letjelicom s udaljenosti. U ovom projektu koristi se FlySky FS-i6 daljinski upravljač zajedno s prijemnikom FS-iA6B.

Daljinski upravljač FlySky FS-i6 dizajniran je za jednostavno korištenje i pruža brojne funkcionalnosti koje su ključne za upravljanje letjelicama. Njegove specifikacije navedene su u tabeli 3., a izgled je prikazan na slici 12.

Tabela 3. Specifikacije FlySky FS-i6

Frekvencijski raspon	2.4 GHz
Broj kanala	6 Domet: Do 500 metara
Napajanje	4 AA baterije
Ekran	LCD zaslon za prikaz informacija o letu i postavki
Kompatibilnost	Podržava različite prijemnike FlySky serije
Dimenzije	174x89x190 mm
Težina	392g

Slika 12. Primjer FlySky FS-i6. Preuzeto s https://www.flyrobo.in/flysky_fs-i6_2.4g_6ch_afhds_rc_transmitter/

Upravljač omogućuje kontrolu osnovnih funkcija kvadkoptera, uključujući kretanje naprijed/natrag, kretanje lijevo/desno, penjanje/spuštanje i rotaciju. LCD zaslon prikazuje važne informacije o letu, kao što su stanje baterije, jačina signala i trenutne postavke kanala. Upravljač ima sigurnosnu funkciju (*engl. fail-safe function*) i alarm za nisku bateriju, što osigurava sigurnost tijekom upravljanja kvadkopterom. („flysky“ , 2016)

2.7.2. Funkcionalnosti FS-iA6B

Prijemnik FS-iA6B je uređaj koji prima signale iz daljinskog upravljača i prenosi ih mikroupravljaču kvadkoptera, omogućujući kontrolu motora i drugih funkcija letjelice. Prijemnik prima radijske signale iz daljinskog upravljača i prenosi ih mikroupravljaču. Povezuje se s ESC-ovima kako bi prilagodio brzinu vrtnje motora na temelju kontrolnih signala. Prijemnik se napaja putem BEC izlaza ESC-a ili izravno iz baterije kvadkoptera. („flysky“ , bez datuma). Specifikacije spomenutog prijamnika prikazane su u tabeli 4., a izgleda na slici 13.

Tabela 4. Specifikacije FS-iA6B prijamnika.

Frekvencijski raspon	2.4 GHz
Broj kanala	6
Domet	Do 500 metara
Napajanje	4.0-6.5V (napaja se iz BEC izlaza ESC-a ili izravno iz baterije kvadkoptera)
Povezivanje	PWM ili PPM signal za povezivanje s mikroupravljačem
Dimenzije	47.42x26.14x15.75 mm
Težina	14.9g

Slika 13. Primjer FS-iA6B. Preuzeto s https://alexnld.com/product/flysky-fs-i6-2-4g-6ch-afhds-rc-transmitter-with-fs-ia6b-receiver/

3. Dizajn sustava

Dizajn kvadkoptera je usmjeren na aerodinamičnu učinkovitost s laganim strukturama, koristeći napredne kompozitne materijale za snagu i izdržljivost. Posebna pažnja posvećena je optimizaciji oblika propelera kako bi se smanjio otpor zraka i povećala ukupna stabilnost leta. Uz to, integrirani sustav za smanjenje vibracija osigurava glatku i preciznu kontrolu tijekom složenih manevara.

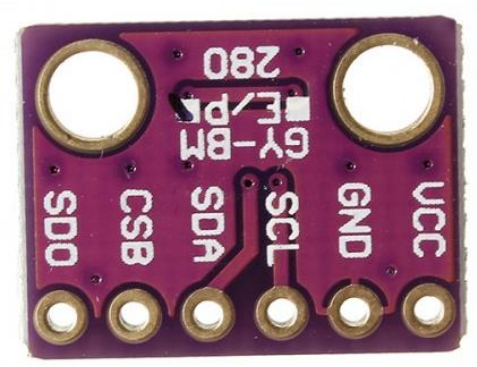
3.1. Sustav napajanja

Kvadkopter je opskrbljen litij-polimer baterijama koje nude ravnotežu između težine i performansi, pružajući dovoljno energije za produženo trajanje leta. (“CarbonAeronautics”, 2023)

3.2. Sustav senzora

Sustav senzora kvadkoptera sastoji se od dva ključna modula: GY-521 MPU-6050 i GY-BME280. GY-521 MPU-6050 kombinira žiroskop i akcelerometar, koji su ključni za praćenje orijentacije i ubrzanja kvadkoptera te za održavanje stabilnosti leta. MPU-6050 također uključuje Digitalni procesor pokreta (DMP) koji omogućuje obradu podataka o pokretima u stvarnom vremenu. („Addicore“ , bez datuma)

GY-BME280 je digitalni senzor koji mjeri temperaturu, barometarski tlak i visinu. Ovaj senzor je važan za precizno određivanje visine kvadkoptera te za kompenzaciju promjena tlaka i temperature tijekom leta. GY-BME280 mjeri temperaturu, barometarski tlak i nadmorsku visinu, napaja se s 3.3V do 5V i komunicira putem I2C sučelja („Protosupplies“ , bez datuma). Ovi senzori zajedno omogućuju kvadkopteru da precizno prati svoju orijentaciju, brzinu i visinu, što je ključno za stabilan i kontroliran let.



Slika 14. Primjer GY-BME280. Preuzeto s https://alexnld.com/product/gy-bmp280-3-3-high-precision-atmospheric-pressure-sensor-module-for-arduino/

3.3. Središnji upravljački sustav

Središnji upravljački sustav kvadkoptera sastoji se od mikroupravljača Teensy 4.0, koji upravlja svim operacijama kvadkoptera, uključujući obradu podataka sa senzora, kontrolu motora i komunikaciju s prijemnikom. Povezivanje senzora, ESC-ova i prijemnika s mikroupravljačem ključno je za funkcionalnost kvadkoptera. Senzori GY-521 MPU-6050 i GY-BME280 povezani su s mikroupravljačem putem I2C sučelja, omogućujući brzu i pouzdanu komunikaciju. Svaki od četiri ESC-a povezan je s mikroupravljačem putem PWM izlaza, što omogućuje preciznu kontrolu brzine motora. Prijemnik FS-iA6B povezan je s mikroupravljačem putem PWM ili PPM signala, omogućujući prijenos kontrolnih signala iz daljinskog upravljača.

Kalibracija i konfiguracija sustava ključni su koraci prije prvog leta kvadkoptera. Prije prvog leta potrebno je kalibrirati žiroskop i akcelerometar kako bi se osigurala točnost mjerenja. Ovaj postupak uključuje postavljanje kvadkoptera na ravnu površinu i pokretanje kalibracijskog postupka. PID kontroler mora se postaviti na odgovarajuće vrijednosti koje se prilagođavaju tijekom testnih letova kako bi se postigla optimalna stabilnost i kontrola. Ovo uključuje fino podešavanje proporcionalnih, integralnih i derivativnih faktora. Također, potrebno je provjeriti da mikroupravljač ispravno prima signale iz prijemnika i da su svi kanali funkcionalni, što se postiže testiranjem komunikacije. Ovi koraci osiguravaju da središnji upravljački sustav kvadkoptera radi ispravno, omogućujući siguran i stabilan let.

3.4. Komunikacijski sustav

Komunikacijski sustav kvadkoptera omogućuje bežičnu kontrolu letjelice putem daljinskog upravljača FlySky FS-i6 i prijemnika FS-iA6B. Ovaj sustav ima ključnu ulogu u prijenosu kontrolnih signala, osiguravanju pouzdanosti i sigurnosti tijekom letenja, te integraciji s ostatkom sustava kvadkoptera.

Funkcionalnosti komunikacijskog sustava uključuju prijenos kontrolnih signala, gdje daljinski upravljač FlySky FS-i6 šalje kontrolne signale prijemniku FS-iA6B. Prijemnik FS-iA6B zatim prenosi te signale mikroupravljaču Teensy 4.0 putem PWM ili PPM signala. Ova komunikacija omogućava mikroupravljaču da upravlja motorima i drugim funkcijama kvadkoptera na temelju zaprimljenih naredbi.

Pouzdanost i sigurnost komunikacijskog sustava osigurana je radom na 2.4 GHz frekvencijskom rasponu, što smanjuje mogućnost smetnji. Uz to, sigurnosne značajke kao što su fail-safe i alarm za nisku bateriju osiguravaju da kvadkopter ostane pod kontrolom čak i u slučaju gubitka signala ili niske baterije. Fail-safe funkcija postavlja kvadkopter u siguran način rada u slučaju gubitka signala, čime se sprječavaju nesreće. Također, moguće je programirati određene reakcije kvadkoptera, poput spuštanja na tlo ili održavanja visine.

Integracija komunikacijskog sustava u kvadkopter uključuje povezivanje prijemnika s mikroupravljačem. Prijemnik FS-iA6B povezan je s mikroupravljačem Teensy 4.0 putem PWM ili PPM signala, što omogućuje prijenos kontrolnih signala koji upravljaju motorima i ostalim funkcijama kvadkoptera. Prije prvog leta, potrebno je kalibrirati daljinski upravljač i prijemnik kako bi se osigurala točnost i pouzdanost kontrolnih signala. Testiranje svih kanala i funkcija osigurava da upravljač ispravno komunicira s kvadkopterom, čime se potvrđuje ispravnost sustava.

Sigurnosne značajke komunikacijskog sustava dodatno povećavaju sigurnost i pouzdanost kvadkoptera. Alarm za nisku bateriju upozorava operatera kada je baterija daljinskog upravljača pri kraju, čime se izbjegavaju neočekivani gubici kontrole. Sigurnosni prekidač sprječava slučajno pokretanje motora, čime se povećava

sigurnost tijekom rukovanja kvadkopterom. Ove sigurnosne mjere osiguravaju da kvadkopter ostane siguran i pod kontrolom u svim uvjetima.

3.5. Konfiguracija motora

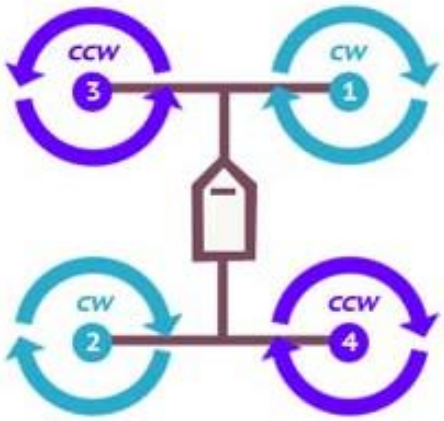
Kvadkopter koristi četiri brushless motora visoke učinkovitosti koji su ključni za stvaranje potiska i manevriranje letjelicom. Svaki motor je kontroliran pomoću elektroničkog regulatora brzine A32 Airplane ESC-Low voltage A32-6A. Najčešće konfiguracije motora uključuju X-konfiguraciju, H-konfiguraciju i + (plus) konfiguraciju.

X-konfiguracija je najčešća konfiguracija motora, gdje su motori raspoređeni u obliku slova "X". Prikazana je na slici 15. Ovaj raspored omogućava uravnotežen potisak i stabilnost leta, što olakšava manevriranje kvadkopterom. Zahvaljujući ovoj konfiguraciji, kvadkopter je sposoban za precizne i brze pokrete, čineći ga pogodnim za različite primjene („circuitdigest“ , bez datum)

Slika 15. Primjer x-konfiguracije. https://circuitdigest.com/article/different-types-of-drone-frames-monocopter-to-octocopter

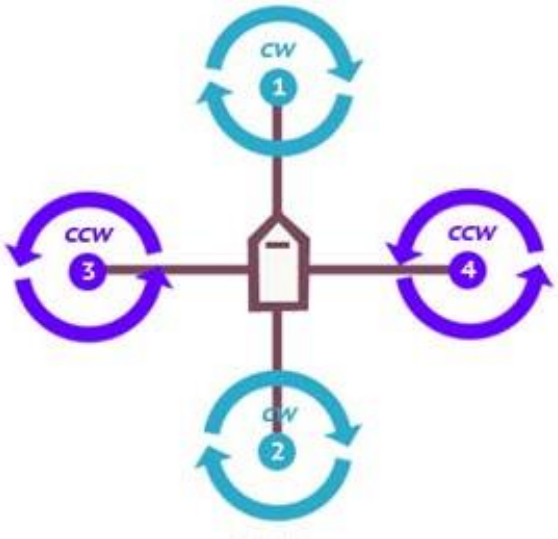
H-konfiguracija raspoređuje motore u obliku slova "H", s produženim bočnim nosačima. Prikazana je na slici 16. Ova konfiguracija pruža dodatnu stabilnost, posebno za kvadkoptere koji nose veći teret ili kamere. U ovom projektu koristi se H-

konfiguracija zbog svoje stabilnosti i sposobnosti nošenja dodatnog tereta. H-konfiguracija je idealna za snimanje iz zraka i druge aplikacije koje zahtijevaju stabilnost i nosivost. („circuitdigest“ , bez datum)



Slika 16. Primjer H-konfiguracije. https://circuitdigest.com/article/different-types-of-drone-frames-monocopter-to-octocopter

(Plus) konfiguracija je rjeđa konfiguracija motora (slika 17.), gdje su motori raspoređeni u obliku znaka "+". Jedan motor nalazi se sprijeda, jedan straga, i dva sa strane. Ova konfiguracija je manje stabilna od X-konfiguracije, ali se ponekad koristi za specifične aplikacije koje zahtijevaju poseban raspored motora. Iako je ova konfiguracija rjeđe viđena, može pružiti prednosti u određenim situacijama koje zahtijevaju specifične letne karakteristike. („circuitdigest“ , bez datum)



Slika 17. Primjer plus-konfiguracije. https://circuitdigest.com/article/different-types-of-drone-frames-monocopter-to-octocopter

Ove različite konfiguracije motora omogućuju prilagodbu kvadkoptera za različite potrebe i aplikacije, osiguravajući optimalne performanse i stabilnost ovisno o specifičnim zahtjevima projekta. U ovom projektu, H-konfiguracija je odabrana zbog svoje sposobnosti pružanja dodatne stabilnosti i nosivosti, što je ključno za zadatke koji uključuju nošenje dodatnog tereta ili snimanje visokokvalitetnih videozapisa iz zraka.

4. Fizika letenja i metode ispitivanja performansi letenja kvadkoptera

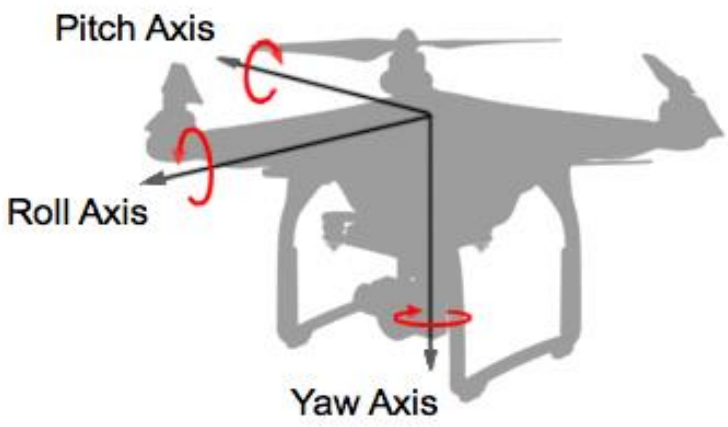
4.1. Fizika letenja kvadkoptera

Fizika letenja kvadkoptera temelji se na principima aerodinamike i mehanike. Kvadkopter koristi četiri rotora za generiranje potiska koji omogućava uzgon, kretanje i stabilizaciju letjelice. (“CarbonAeronautics”, 2023) Osnovni principi uključuju uzgon, rotaciju, nagib i valjanje.

Uzgon se generira rotacijom propelera. Kada se propeleri kvadkoptera okreću, stvaraju uzgon koji djeluje suprotno gravitaciji. Ovaj uzgon omogućuje kvadkopteru da se podiže, spušta i lebdi u zraku. Bez dovoljno uzgona, kvadkopter ne bi mogao ostati u zraku.

Rotacija (*engl. yaw*) kvadkoptera oko vertikalne osi postiže se diferencijalnom brzinom rotacije dijagonalno suprotnih rotora. Kada se brzina rotacije jednog para dijagonalno suprotnih rotora poveća, a drugog para smanji, kvadkopter se rotira oko svoje vertikalne osi. Ova rotacija omogućuje kvadkopteru da mijenja smjer u kojem je okrenut.

Nagib (*engl. pitch*) i valjanje (*engl. roll*) kvadkoptera omogućuju kretanje naprijed/nazad i lijevo/desno. Nagib se postiže diferencijalnom brzinom rotacije prednjih i stražnjih rotora. Kada se brzina rotacije prednjih rotora poveća, a stražnjih smanji, kvadkopter se naginje naprijed. Suprotno, kada se brzina rotacije stražnjih rotora poveća, kvadkopter se naginje nazad. Valjanje se postiže diferencijalnom brzinom rotacije lijevih i desnih rotora. Kada se brzina rotacije lijevih rotora poveća, kvadkopter se naginje desno, i obrnuto („developer.dji“ , bez datuma). Primjer rotacije, nagiba i valjanja prikazan je na slici 18.



Slika 18. Primjer rotacije, nagiba i valjanja. https://developer.dji.com/doc/mobile-sdk-tutorial/en/basic-introduction/basic-concepts/flight-control.html

Osnovne sile i momenti koji djeluju na kvadkopter uključuju težinu, potisak i otpor zraka. Težina (*W*) djeluje prema dolje zbog gravitacije. Ova sila mora biti nadvladana uzgonom kako bi kvadkopter ostao u zraku. Potisak (*T*) djeluje prema gore i stvara se rotacijom propelera. Ova sila omogućava kvadkopteru da se podiže, spušta ili lebdi. Otpor zraka (*D*) djeluje suprotno smjeru kretanja kvadkoptera i ovisi o brzini i površini izloženoj struji zraka. Otpor zraka smanjuje brzinu kvadkoptera i mora se prevladati kako bi se održao željeni smjer i brzina kretanja.

4.2. Metode mjerenja

Ispitivanje performansi letenja kvadkoptera uključuje niz metoda koje omogućuju ocjenu stabilnosti, upravljivosti i efikasnosti letjelice. Glavne metode ispitivanja uključuju levitaciju, odziv na poremećaje, testiranje upravljivosti, testiranje efikasnosti, te testiranje visine i brzine.

Test levitacije provodi se kako bi se osiguralo da kvadkopter može stabilno lebdjeti na jednom mjestu bez značajnih oscilacija. Tijekom ovog testa, mjerenja se provode pomoću žiroskopa i akcelerometra kako bi se ocijenila stabilnost kutova nagiba i valjanja. Cilj je potvrditi da kvadkopter može održavati stabilan položaj u zraku bez neželjenih pomaka.

Odziv na poremećaje testira se tako što se kvadkopter namjerno izlaže malim poremećajima kako bi se ocijenio njegov odziv i sposobnost povratka u stabilno stanje. Tijekom ovog testa koristi se PID kontroler za prilagodbu parametara i poboljšanje stabilnosti letjelice. Ovaj test pomaže identificirati optimalne postavke kontrolera koje omogućuju brz i precizan povratak kvadkoptera u ravnotežu nakon poremećaja.

Testiranje upravljivosti obuhvaća osnovne manevre kao što su kretanje naprijed, nazad, lijevo i desno, te rotaciju oko vertikalne osi (*engl. yaw*). Također se testiraju napredni manevri, uključujući brze promjene smjera, nagle zaokrete i složenije letne operacije. Ovi testovi omogućuju procjenu koliko je kvadkopter upravljiv i kako se ponaša tijekom dinamičkih letnih manevara.

Testiranje efikasnosti uključuje mjerenje potrošnje energije tijekom leta kako bi se ocijenila ukupna efikasnost kvadkoptera. Provode se testovi trajanja leta kako bi se odredilo maksimalno vrijeme leta pri različitim opterećenjima i uvjetima. Ovi testovi pomažu identificirati optimalne načine upravljanja energijom kako bi se produljilo vrijeme leta.

Testiranje visine i brzine provodi se pomoću barometra i akcelerometra za precizno mjerenje visine kvadkoptera tijekom leta. Iako GPS modul nije korišten, brzina kvadkoptera se procjenjuje pomoću drugih senzora, omogućujući ocjenu horizontalne i vertikalne brzine. Ovi testovi osiguravaju da kvadkopter može letjeti na željenim visinama i brzinama te omogućuju prilagodbu kontrolnih algoritama za postizanje optimalnih performansi.

4.3. Fizikalne metode i jednadžbe

4.3.1. Osnovne jednadžbe

Newtonov drugi zakon (uzgon):

$$F = ma$$

gdje je F sila, m masa kvadkoptera, a a ubrzanje. Uzgon T mora biti jednak težini kvadkoptera W za levitaciju:

$$T = mg$$

Moment sile za skretanje, odnosno rotaciju oko vertikalne osi:

$$\tau = I\alpha$$

gdje je τ moment sile, I moment inercije, a α kutno ubrzanje. Diferencijalna brzina rotacije dijagonalno suprotnih rotora stvara moment koji uzrokuje rotaciju kvadkoptera oko svoje vertikalne osi. Ovaj princip omogućava kvadkopteru da mijenja orijentaciju bez promjene svoje pozicije u prostoru.

Kada jedan par dijagonalno suprotnih rotora poveća brzinu rotacije, dok drugi par smanji, stvara se neto moment sile koji uzrokuje rotaciju kvadkoptera. Na primjer, ako se rotor A i D ubrzaju, dok se rotor B i C uspore, kvadkopter će se rotirati u smjeru suprotnom od smjera kazaljke na satu. Ova sposobnost precizne kontrole rotacije ključna je za agilnost i stabilnost kvadkoptera tijekom leta.

4.3.2. PID kontroler

PID kontroler koristi se za stabilizaciju i kontrolu leta kvadkoptera. Njegova glavna funkcija je smanjiti pogrešku između željene i stvarne vrijednosti parametra (npr. položaja ili brzine) kvadkoptera

PID kontroler:

$$u(t) = \mathrm{K_p} e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

gdje je $u(t)$ izlaz kontrolera, $e(t)$ pogreška, $\mathrm{K_p}$ proporcionalni koeficijent, K_i integralni koeficijent, i K_d derivativni koeficijent. Ova jednadžba koristi se za stabilizaciju i kontrolu leta. („Visioli“ , 2009)

4.3.3. Kalmanov filter za procjenu stanja

Kalmanov filter je algoritam za procjenu stanja dinamičkih sustava iz niza nepreciznih i nesigurnih mjerenja. On se koristi za filtriranje šuma iz senzorskih podataka i za pružanje bolje procjene stvarnog stanja sustava.

Ažuriranje predviđanja pomoću Kalmanovog filtera uključuje dva ključna koraka: i) predviđanje stanja i ii) predviđanje kovarijance greške:

$$\hat{x}_k^- = F_{k-1} + Bu_{k-1},$$

$$P_k^- = FP_{k-1}F^T + Q.$$

Ažuriranje mjerenja:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$$P_k = (I - K_k H)P_k^-$$

gdje su $\hat{x}_k$ procijenjeno stanje, P_k kovarijacijska matrica pogreške, P matrica prijelaza stanja, B matrica kontrolnog ulaza, Q procesna kovarijacijska matrica, H matrica mjerenja, R kovarijacijska matrica mjerenja, K_k Kalmanov gain, i z_k mjerenje.

4.3.4. Jednadžba za određivanje visine

Jednadžba za visinu koristeći barometar:

$$h = 44330(1 - \left(\frac{P}{P_0}\right)^{\frac{1}{5.255}})$$

gdje je h visina, P barometarski tlak, i P_0 referentni tlak na razini mora.

5. Programiranje kvadkoptera

Programiranje kvadkoptera uključuje postavljanje i konfiguraciju mikroupravljača, implementaciju kontrolnih algoritama te integraciju sa senzorima i ESC-ovima. (“CarbonAeronautics”, 2023) U ovom projektu korišteni su Teensy 4.0 mikroupravljač, knjižnice BMC i BasicLinearAlgebra, te potrebne knjižnice za rad sa senzorima i komunikacijskim modulima. Programiranje je provedeno u Arduino IDE s dodatkom Teensyduino.

5.1. Postavljanje razvojnog okruženja

Za programiranje kvadkoptera korišteni su Arduino IDE i Teensyduino dodatak. Arduino IDE korišten je za pisanje i učitavanje koda na mikroupravljač Teensy 4.0. Instalirane su potrebne knjižnice BMC i BasicLinearAlgebra, kao i knjižnice za rad sa senzorima poput "Wire.h" za I2C komunikaciju, "MPU6050.h" za žiroskop i akcelerometar, te "Adafruit_BME280.h" za barometar.

5.2. Konfiguriranje

Prvo je potrebno postaviti brzinu I2C komunikacije na 400 kHz, iza čega se inicijalizira I2C sabirnica i konfiguriraju se GY-521 MPU-6050 i GY-BME280. Dio koda odgovaran za navedeno prikazan je ispod:

```
void setup() {
    Wire.setClock(400000);
    Wire.begin();
    delay(250);
    Wire.beginTransmission(0x68);
    Wire.write(0x6B);
    Wire.write(0x00);
    Wire.endTransmission();
    Wire.beginTransmission(0x76);
    Wire.write(0xF4);
    Wire.write(0x57);
    Wire.endTransmission();
    Wire.beginTransmission(0x76);
```

```
    Wire.write(0xF5);
    Wire.write(0x14);
    Wire.endTransmission();
    …
```

Nakon I2C inicijalizacije potrebno je provesti kalibraciju senzora kako bi se osigurala preciznost mjerenja:

```
    for (RateCalibrationNumber=0; RateCalibrationNumber<2000;
      RateCalibrationNumber++) {
        gyro_signals();
        RateCalibrationRoll += RateRoll;
        RateCalibrationPitch += RatePitch;
        RateCalibrationYaw += RateYaw;
        barometer_signals();
        AltitudeBarometerStartUp += AltitudeBarometer;
        delay(1);
    }
    RateCalibrationRoll /= 2000;
    RateCalibrationPitch /= 2000;
    RateCalibrationYaw /= 2000;
    AltitudeBarometerStartUp /= 2000;
}
```

5.3. Implementacija kontrolnih algoritama

PID algoritmi implementirani su za stabilizaciju i kontrolu leta kvadkoptera. PID kontrola koristi povratne informacije sa senzora za kontinuiranu prilagodbu brzine motora kako bi se održala željena orijentacija i položaj letjelice.

Proporcionalni dio PID funkcije je jednostavno proporcionalan trenutnoj grešci. To znači da će se izlaz povećati proporcionalno grešci *Error*. Veća greška rezultira većim izlaznim signalom, koji pomaže sustavu da brzo reagira. Matematički je to zapisano na sljedeći način:

$$Pterm = P \times Error$$

Integralni dio je odgovoran za eliminaciju stalne pogreške tijekom vremena. Računa se kao suma trenutne i prethodne greške, skalirana s koeficijentom I.

$$Iterm = PrevIterm + I \times (Error + PrevError) \times Timefac/2$$

Također, u konkretnom slučaju koristi se vremenski faktor $Timefac = 0.004$. Ovdje se integralni dio PID funkcije ograničava na vrijednost između -400 i 400 kako bi se spriječio efekt zasićenja (tzv. integralni vjetar).

Derivativni dio je proporcionalan brzini promjene greške. Ova komponenta pomaže u prigušivanju sustava, smanjujući oscilacije. Računa se kao razlika između trenutne i prethodne greške, podijeljena s vremenskim intervalom 0.004:

$$Dterm = D \times \frac{Error - PrevError}{0.004}$$

Izlaz PID funkcije je dan sljedećim izrazom:

$$PIDOutput = Pterm + Iterm + Dterm$$

Pored spomenutog, koje se sprema u 0.-ti element globalnog polja PIDReturn, kao 1. i 2. element polja se redom spremaju pogreška *Error* i *Iterm*. Kod za navedenu PID jednadžbu se nalazi ispod:

```
void pid_equation(float Error, float P, float I, float D, float PrevError,
float PrevIterm) {
    float Pterm = P * Error;
    float Iterm = PrevIterm + I * (Error + PrevError) * 0.004 / 2;
    if (Iterm > 400) Iterm = 400;
    else if (Iterm < -400) Iterm = -400;
    float Dterm = D * (Error - PrevError) / 0.004;
    float PIDOutput = Pterm + Iterm + Dterm;
    if (PIDOutput > 400) PIDOutput = 400;
    else if (PIDOutput < -400) PIDOutput = -400;
    PIDReturn[0] = PIDOutput;
    PIDReturn[1] = Error;
    PIDReturn[2] = Iterm;
}
```

5.4. Kontrola motora i ESC-ova

Brzina motora kontrolirana je putem PWM signala. Konfiguracija PWM signala omogućava precizno upravljanje svakim motorom.

```
void loop() {
    gyro_signals();
    RateRoll -= RateCalibrationRoll;
    RatePitch -= RateCalibrationPitch;
    RateYaw -= RateCalibrationYaw;
    . . .
```

Funkciju *kalman_1d()* pozivamo dva puta. Prvi puta za trenutnu procjenu kuta valjanja (prevrtanja) *KalmanAngleRoll* i trenutnu procjenu neodređenosti kuta valjanja (prevrtanja) *KalmanUncertaintyAngleRoll*. S druge strane varijabla *RateRoll* predstavlja brzinu promjene kuta valjanja (izmjerena od strane žiroskopa), a *AngleRoll* izmjereni kut (od strane akcelerometra). Kao argumente funkcija *kalman_1d()* uzima prethodne vrijednosti *KalmanAngleRoll, KalmanUncertaintyAngleRoll,* te varijable *RateRoll i AngleRoll.* Kao rezultat dobivamo novu, odnosno stabiliziranu vrijednost kuta valjanja *KalmanAngleRoll* i *KalmanUncertaintyAngleRoll*. Kod koji opisuje navedeno nalazi se ispod:

```
    // Kalman filter za stabilizaciju
    kalman_1d(KalmanAngleRoll, KalmanUncertaintyAngleRoll, RateRoll,
AngleRoll);
    KalmanAngleRoll = Kalman1DOutput[0];
    KalmanUncertaintyAngleRoll = Kalman1DOutput[1];
```

U slučaju drugog pozivanja funkcije *kalman_1d()* kao argumente imamo *KalmanAnglePitch* i *KalmanUncertaintyAnglePitch*, te brzinu promjene kuta *RatePitch* izmjerenu i kut nagiba *AnglePitch*, izmjerene od strane žiroskopa, odnosno akcelerometra. Kao rezultat dobivamo novu, odnosno stabiliziranu vrijednost nagiba *KalmanAnglePitch* i neodređenosti nagiba *KalmanUncertaintyAnglePitch*. Kod koji opisuje spomenuto nalazi se ispod:

```
kalman_1d(KalmanAnglePitch, KalmanUncertaintyAnglePitch, RatePitch,
AnglePitch);
    KalmanAnglePitch = Kalman1DOutput[0];
    KalmanUncertaintyAnglePitch = Kalman1DOutput[1];
```

Pogreške brzine valjanja (prevrtanja), nagiba (rotacije oko bočne ili poprečne osi) i skretanja (rotacije oko okomite osi) izračunate su kao razlike željenih i trenutnih vrijednosti navedenih veličina. Odgovarajuće varijable koje se pritom računaju su *ErrorRateRoll*, *ErrorRatePitch* i *ErrorRateYaw*:

```
// PID kontrola za roll, pitch i yaw
    ErrorRateRoll = DesiredRateRoll - RateRoll;
    ErrorRatePitch = DesiredRatePitch - RatePitch;
    ErrorRateYaw = DesiredRateYaw - RateYaw;
```

Kvadkopteri koriste različite kombinacije snage motora za kontrolu svoje orijentacije i visine. PID funkcija omogućava dinamičko podešavanje kontrolnog signala kako bi se postigla njegova željena vrijednost upravljanom sustavu. Putem varijabli *InputPitch* i *InputRoll* kontrolira se nagib kvadkoptera tako što se povećava ili smanjuje snaga suprotnih parova motora. Na primjer, kada se želi kvadkopter nagnuti naprijed, povećava se snaga stražnjim motorima i smanjuje snaga na prednjim motorima. S druge strane varijabla *InputYaw* kontrolira rotaciju kvadkoptera oko vertikalne osi tako što se povećava snaga dva motora koji se nalaze dijagonalno nasuprot jedan drugome, a smanjuje na preostala dva. Naime, za kontroliranje prevrtanja funkcija *pid_equation()* ima sljedeće argumente *ErrorRateRoll, PRateRoll, IRateRoll, DRateRoll, PrevErrorRateRoll, PrevItermRateRoll*, a kao izlaz dobivamo nove vrijednosti *InputRol*, *PrevErrorRateRoll* i *PrevItermRateRoll.*

```
pid_equation(ErrorRateRoll, PRateRoll, IRateRoll, DRateRoll,
PrevErrorRateRoll, PrevItermRateRoll);
    InputRoll = PIDReturn[0];
    PrevErrorRateRoll = PIDReturn[1];
    PrevItermRateRoll = PIDReturn[2];
```

Slično stoji i za kotroliranje nagiba, samo što *pid_equation()* sad ima sljedeće argumente: *ErrorRatePitch*, *PRatePitch*, *IRatePitch*, *DRatePitch*, *PrevErrorRatePitch*, *PrevItermRatePitch*. Kao izlaz dobivamo nove vrijednosti InputPitch, PrevErrorRatePitch i PrevItermRatePitch.

```
pid_equation(ErrorRatePitch, PRatePitch, IRatePitch, DRatePitch,
PrevErrorRatePitch, PrevItermRatePitch);
    InputPitch = PIDReturn[0];
    PrevErrorRatePitch = PIDReturn[1];
    PrevItermRatePitch = PIDReturn[2];
```

Također i za kut skretanja imamo istu PID funkciju, samo što su argumenti funkcije sada varijable *ErrorRateYaw, PRateYaw, IRateYaw, DRateYaw, PrevErrorRateYaw, PrevItermRateYaw*:

```
pid_equation(ErrorRateYaw, PRateYaw, IRateYaw, DRateYaw, PrevErrorRateYaw,
PrevItermRateYaw);
    InputYaw = PIDReturn[0];
    PrevErrorRateYaw = PIDReturn[1];
    PrevItermRateYaw = PIDReturn[2];
```

Slično, izlazi su *InputYaw*, *PrevErrorRateYaw* i *PrevItermRateYaw*. Zatim je potrebno izračunati ulaze za motore kvadkoptera na temelju kontrolnih ulaza *(InputThrottle, InputPitch, InputRoll, InputYaw*). Putem varijable *InputThrottle* povećava sve motorne ulaze proporcionalno kako bi kvadkopter mogao podići. Tada za ulaze motora imamo sljedeći kod:

```
// Ograničenja za motorni ulaz
    MotorInput1 = 1.024 * (InputThrottle - InputPitch - InputRoll - InputYaw);
    MotorInput2 = 1.024 * (InputThrottle + InputPitch - InputRoll + InputYaw);
    MotorInput3 = 1.024 * (InputThrottle + InputPitch + InputRoll - InputYaw);
    MotorInput4 = 1.024 * (InputThrottle - InputPitch + InputRoll + InputYaw);
```

Izračunate ulaze motora moraju se ograničiti tako da ne prelaze određene vrijednosti. Za sva četiri motora to je maksimalna vrijednost 2000. Također, izračunata vrijednost ne smije pasti ispod „idealne“ jer bi to uzrokovalo pad drona, te je stoga izračunate vrijednosti potrebno postaviti u *ThrottleIdle*, koji je u ovom slučaju 1180.

```
// Ograničenja za vrijednosti
    if (MotorInput1 > 2000) MotorInput1 = 1999;
    if (MotorInput2 > 2000) MotorInput2 = 1999;
    if (MotorInput3 > 2000) MotorInput3 = 1999;
    if (MotorInput4 > 2000) MotorInput4 = 1999;
    int ThrottleIdle = 1180;
    if (MotorInput1 < ThrottleIdle) MotorInput1 = ThrottleIdle;
    if (MotorInput2 < ThrottleIdle) MotorInput2 = ThrottleIdle;
    if (MotorInput3 < ThrottleIdle) MotorInput3 = ThrottleIdle;
    if (MotorInput4 < ThrottleIdle) MotorInput4 = ThrottleIdle;
```

Ako je vrijednost gasa pala ispod granice isključivanja, sve varijable koje predstavljaju ulazne signale za motore (MotorInput1, MotorInput2, MotorInput3, MotorInput4) postavljaju se na vrijednost ThrottleCutOff. Ovo će uzrokovati isključivanje svih motora:

```
// Isključivanje motora pri niskom gasu
    int ThrottleCutOff = 1000;
```

```
    if (ReceiverValue[2] < 1050) {
        MotorInput1 = ThrottleCutOff;
        MotorInput2 = ThrottleCutOff;
        MotorInput3 = ThrottleCutOff;
        MotorInput4 = ThrottleCutOff;
        reset_pid();
    }
```

Nakon što se motori isključe, poziva se funkcija *reset_pid()* koja koristi za resetiranje PID kontrolera. To je važno jer će se nakon ponovnog uključivanja motora kvadkopter vjerojatno morati ponovno kalibrirati kako bi se osiguralo stabilno upravljanje. Putem funkcije *analogWrite()* šaljemo PWM signal svakom motoru pojedinačno. Ovdje se na pinu 1 postavlja PWM signal čija je širina impulsa određena vrijednošću *MotorInput1*. Analogno vrijedi za ostale motore. Ovisno o platformi i konfiguraciji, ova vrijednost obično varira od 0 (najmanja snaga) do 255 (najveća snaga). Što je veća vrijednost, to je jači PWM signal i veća snaga koja se šalje na motor spojen na taj pin. Odgovarajući kod prikazan je ispod:

```
// Slanje PWM signala motorima
    analogWrite(1, MotorInput1);
    analogWrite(2, MotorInput2);
    analogWrite(3, MotorInput3);
    analogWrite(4, MotorInput4);
```

Praćenje potrošnje baterije omogućava mjerenje preostalog kapaciteta baterije i signalizaciju kada kapacitet padne ispod određene razine, čemu odgovara sljedeći dio koda:

```
// Praćenje potrošnje baterije
    battery_voltage();
    CurrentConsumed = Current * 1000 * 0.004 / 3600 + CurrentConsumed;
    BatteryRemaining = (BatteryAtStart - CurrentConsumed) / BatteryDefault *
100;
    if (BatteryRemaining <= 30) digitalWrite(5, HIGH);
    else digitalWrite(5, LOW);
```

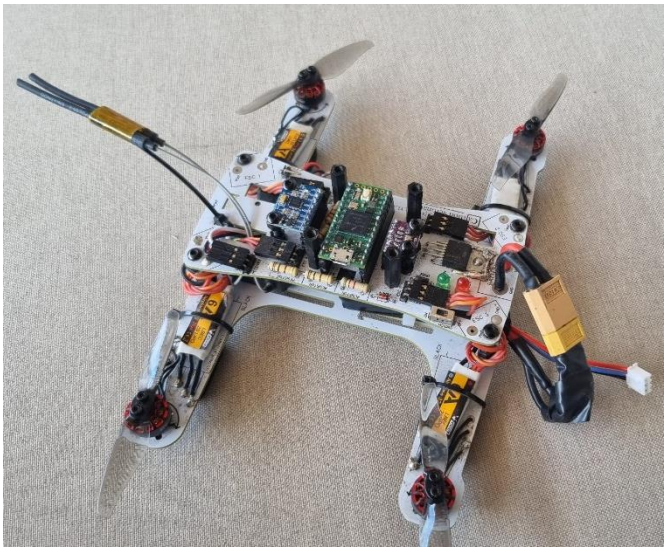
Kontrola vremena petlje osigurava da se određeni dio koda izvršava s određenom frekvencijom, što je u ovoj vremensko osjetljivoj primjeni, odnosno stabilizaciji leta u kvadkopteru od velike važnosti:

```
. . .
// Kontrola vremena petlje
while (micros() - LoopTimer < 4000);
LoopTimer = micros();
}
```

6. Rezultati

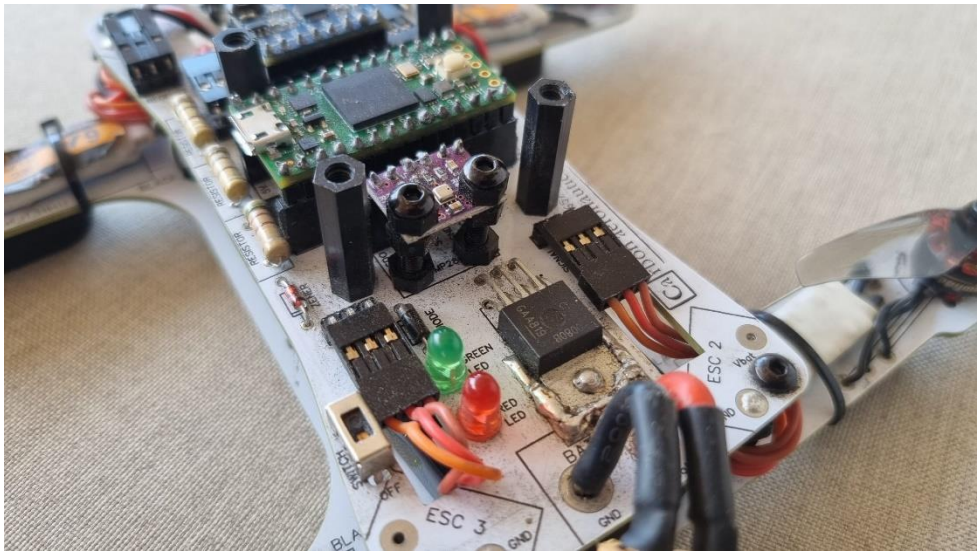
6.1. Stabilnost leta

Kvadkopter je uspio održavati stabilan let bez značajnih oscilacija. Ovo je postignuto preciznom kalibracijom senzora i optimizacijom PID kontrolera. Ispravne postavke PID kontrolera osigurale su uravnoteženu reakciju na promjene smjera i brzine, omogućujući kvadkopteru da stabilno lebdi i učinkovito reagira na kontrolne komande. Iako kvadkopter trenutno leti u prihvatljivom rezultatu, postoji mogućnost daljnjeg poboljšanja stabilnosti kroz dodatne prilagodbe i optimizacije. Na slici 19. prikazan je kvadkopter nakon spajanja svih komponenti.

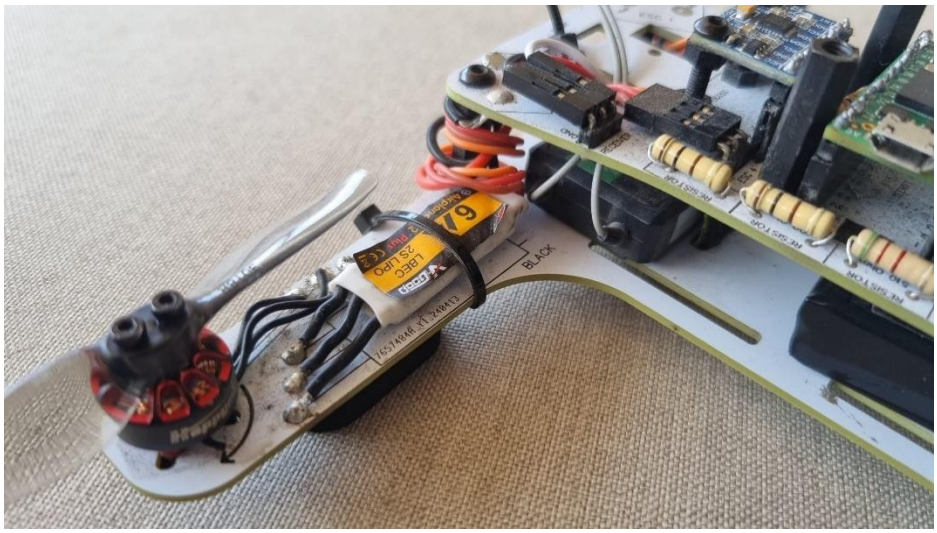


Slika 19. Cijeli kvadkopter nakon spajanja svih komponenti.

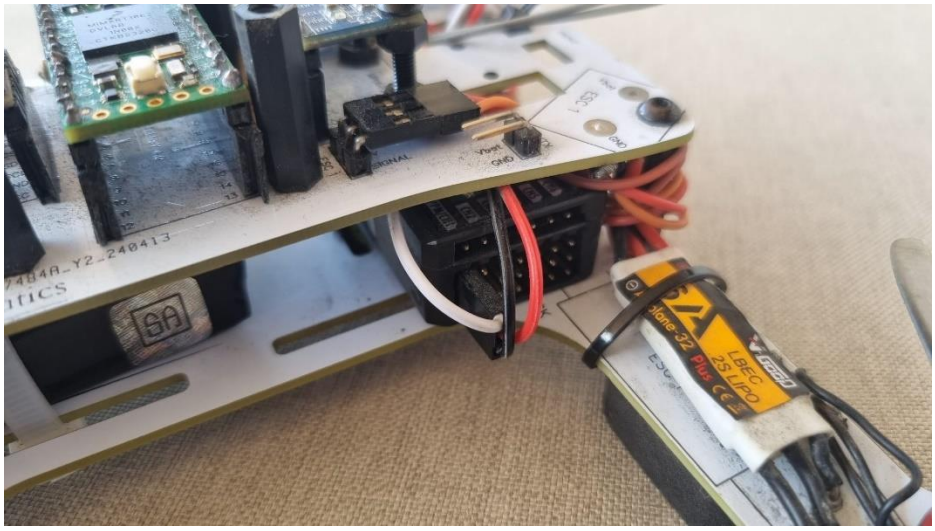
Na slici 19. prikazana je gornja PCB ploča s mikrokontrolerom. Motor i ESC, koji su zalemljeni na donji PCB kvadkoptera, prikazani su na slici 21. Prijemnik FS-iA6B unutar kvadkoptera prikazan je na slici 22.



Slika 20. Gornja strana drona.



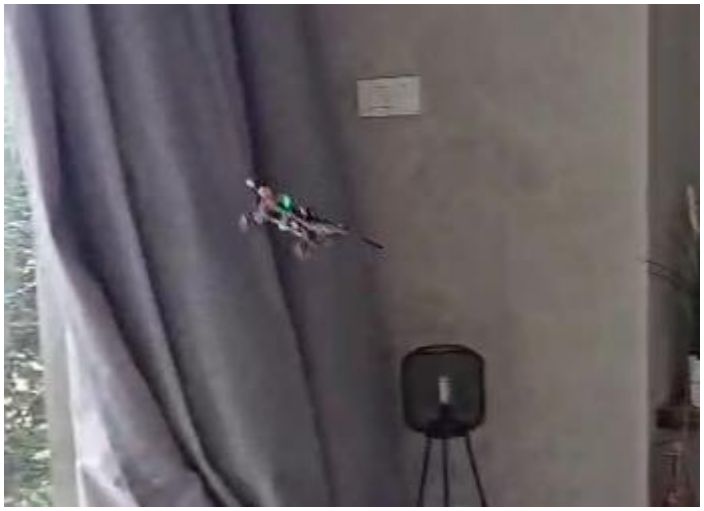
Slika 21. Motor i ESC koji su zalemljeni na donji PCB kvadkoptera.



Slika 22 . Prijemnik FS-iA6B unutar kvadkoptera.

Nekoliko puta je testiran let kvadkoptera na otvorenom. Fotografija uspješnog leta na otvorenom prikazana je na slici 23. U zatvorenom prostoru nešto je teže bilo upravljati dronom pa je bilo slučajeva kad se zabio u prepreke poput zida. Slika 24. prikazuje trenutak prije sudara s preprekom.

Slika 23. Uspješan let kvadkoptera na otvorenom prostoru.



Slika 24. Kvadkopter neposredno prije sudara s preprekom (zidom).

6.2. Kontrola motora

Motori su radili skladno, omogućujući kvadkopteru da precizno izvršava komande za manevriranje. Prijenos PPM signala između prijemnika i mikroupravljača bio je pouzdan i bez smetnji, osiguravajući točan odziv motora. Nakon kalibracije svih motora i zamjene pokvarenog ESC-a, motori su radili na PPM signalima kako bi mogli funkcionirati preko mikrokontrolera. Ovo je rezultiralo preciznim manevrima i stabilnim letom.

Promjena smjera i brzine kvadkoptera testirana je iterativnim prilagodbama PID postavki. Početno testiranje identificiralo je nepravilnosti u letu, poput nestabilnosti i odstupanja od željenog smjera. Nakon svakog leta, PID koeficijenti (proporcionalni, integralni i derivativni) prilagođavani su na temelju zapažanja. Kvadkopter je zatim ponovno pušten u let kako bi se provjerilo poboljšanje.

Ovaj ciklus testiranja i podešavanja ponavljao se dok kvadkopter nije postigao zadovoljavajuće performanse. Fine prilagodbe provedene su kako bi se dodatno optimizirale performanse. Kroz ovaj proces kvadkopter je postigao stabilan let i odgovarajuću kontrolu smjera i brzine, omogućujući pravilno funkcioniranje kroz mnogo pokušaja i pogrešaka.

Motori su kalibrirani pomoću ESC-a koji su bili izravno spojeni na gornji PCB. Proces kalibracije započeo je povezivanjem svakog motora s odgovarajućim ESC-om. Zatim su ESC-ovi konfigurirani za rad u PPM modu, što omogućuje preciznu kontrolu putem mikroupravljača. Svaki ESC je individualno kalibriran kako bi osigurao sinkroniziran rad svih motora. Kalibracija je uključivala postavljanje minimalnih i maksimalnih vrijednosti gasa, čime se osigurava da svi motori pružaju jednak potisak. Nakon kalibracije, motori su testirani kako bi se potvrdila njihova ispravna funkcionalnost i sinkronizacija, što je ključno za stabilan i kontroliran let kvadkoptera.

6.3. Trajanje leta

Kvadkopter je postigao trajanje leta do 10 minuta, što je omogućilo dovoljno vremena za testiranje i korištenje kvadkoptera u različitim uvjetima. Korištenjem lakše baterije od 40 grama, umjesto početne baterije od 80 grama, značajno smo poboljšali performanse letjelice i produljili vrijeme leta.

Tabela 5. Trajanje leta drona sa različitim baterijama

red. br. mjerenja	*baterija*	*trajanje leta*
1	*40 g*	*≈10 min*
2	*80 g*	*8 min 37 s*
3	*80 g*	*8 min 15 s*

6.4. Integracija sustava

Učinkovita integracija svih sustava kvadkoptera omogućila je skladan rad različitih komponenti, uključujući senzore, motore i komunikacijski sustav. Ovo je osiguralo optimalne performanse kvadkoptera i omogućilo preciznu kontrolu i stabilnost tijekom leta. Integracija različitih komponenti bila je besprijekorna, osiguravajući da kvadkopter može ispuniti sve postavljene zadatke s visokim stupnjem pouzdanosti.

7. Diskusija

7.1. Neuspješni rezultati

Tijekom testiranja kvadkoptera naišli smo na nekoliko problema koji su utjecali na performanse letjelice. Ovi problemi uključuju nestabilnost tijekom leta, nepreciznu kontrolu motora i probleme s težinom baterije.

Jedan od glavnih izazova bio je nestabilan let kvadkoptera. Nestabilnost leta bila je rezultat neadekvatno podešenih PID vrijednosti. PID kontroleri, koji su ključni za održavanje stabilnosti leta, nisu bili pravilno podešeni, što je rezultiralo značajnim oscilacijama i gubitkom stabilnosti tijekom letenja. Podešavanje PID kontrolera zahtijeva puno vremena i opsežno testiranje kako bi se postigla optimalna stabilnost.

Drugi problem koji je uočen bila je neprecizna kontrola motora. Na početku, motori nisu bili kalibrirani, što je rezultiralo neujednačenim radom motora. Prilikom kalibracije, primijećeno je da je jedan ESC bio pokvaren, što je dodatno doprinijelo problemima s kontrolom motora. Nakon identifikacije, pokvareni ESC bio je zamijenjen novim iza čega je kalibriran svaki ESC zasebno koristeći kontroler u PPM modu kako bi se osiguralo da svi motori rade skladno i precizno.

Težina baterije također je predstavljala izazov. U početku je korištena baterija koja je težila 80 grama, što je bilo preteško za kvadkopter i sprječavalo ga da stabilno leti. Optimalna težina baterije trebala je biti oko 40 grama kako bi kvadkopter mogao letjeti bez problema. Nakon zamjene baterije lakšom verzijom, performanse leta su se značajno poboljšale.

Ovi neuspješni rezultati pokazali su važnost pravilne kalibracije i odabira komponenti, kao i nužnost detaljnog testiranja i prilagodbe kako bi se osigurale optimalne performanse kvadkoptera.

7.2. Greške koje se mogu izbjeći

Prilikom izrade kvadkoptera, pažljivo i precizno lemljenje glavnih komponenti na gornji PCB je od presudne važnosti, posebno energetski čip koji je ključan za prijenos električne snage. Nakon lemljenja, potrebno je koristiti multimetar kako bi se provjerio protok struje i osiguralo da su svi spojevi ispravni. Ispravno lemljenje Power chipa osigurava pouzdanost i učinkovitost napajanja kvadkoptera.

Nakon lemljenja, bitno je temeljito kalibrirati ESC-ove. Kalibracija ESC-ova je ključna kako bi se osigurala pravilna kontrola motora. Nakon kalibracije ESC-ova, potrebno je testirati let i PID kontroler. Većina vremena bit će posvećena regulaciji PID-a kako bi se postigla optimalna stabilnost drona. Precizno podešavanje PID kontrolera ključno je za stabilan i pouzdan let kvadkoptera.

7.3. Moguće nadogradnje

Postoji nekoliko mogućnosti za nadogradnju kvadkoptera koje bi mogle poboljšati njegove performanse i funkcionalnost. Prva mogućnost je ugradnja naprednijih senzora s većom preciznošću, poput inercijske mjerne jedinice (*engl. inertial measurement unit* ili IMU) s više osi ili naprednijih barometara. To bi moglo poboljšati stabilnost i kontrolu leta, omogućujući kvadkopteru da preciznije reagira na promjene u okolini.

Zamjena trenutnih motora jačim motorima može povećati nosivost kvadkoptera. To bi omogućilo letjelici da nosi teži teret ili dodatnu opremu, poput kamera visoke rezolucije, čime bi se proširila funkcionalnost kvadkoptera.

Korištenje baterija s većim kapacitetom može produžiti vrijeme leta kvadkoptera. Iako treba paziti na dodatnu težinu koju donose veće baterije, produženo vrijeme leta omogućilo bi dulje operativne periode bez potrebe za čestim mijenjanjem baterija.

Ugradnja GPS modula može omogućiti napredne funkcije poput autonomnog letenja, povratka na početnu točku i praćenja rute. Ova nadogradnja značajno bi povećala mogućnosti kvadkoptera u pogledu navigacije i sigurnosti.

Nadogradnja komunikacijskog sustava s daljinskim upravljačem koji ima veći domet može omogućiti kontrolu kvadkoptera na većim udaljenostima. To bi operaterima omogućilo da koriste kvadkopter u širem rasponu aplikacija, uključujući one koje zahtijevaju rad na velikim udaljenostima.

Korištenje ESC-ova s većim kapacitetom i boljom kontrolom može poboljšati odziv motora i povećati ukupnu pouzdanost sustava. Bolji ESC-ovi omogućuju finiju kontrolu brzine i momenta motora, što doprinosi stabilnijem letu.

Implementacija sustava za upravljanje kvadkopterom putem mobilne aplikacije može omogućiti lakšu kontrolu i pristup naprednim postavkama. To bi korisnicima omogućilo jednostavnije prilagođavanje postavki i upravljanje letjelicom putem pametnih uređaja.

Poboljšanje i optimizacija softverskog koda za kontrolu kvadkoptera može dodatno povećati učinkovitost i stabilnost leta. Optimiziran softver omogućava brže i preciznije obrade podataka, što rezultira boljim performansama letjelice.

Implementacija naprednijih algoritama za stabilizaciju, poput Kalmanovog filtra ili drugih naprednih kontrolnih algoritama, može značajno poboljšati stabilnost i performanse kvadkoptera. U izvornom kod korišten je samo Kalmanov filter. Neki napredniji algoritmi omogućili bi precizniju kontrolu i stabilizaciju leta, što je ključno za zahtjevne aplikacije. Ove nadogradnje mogu značajno poboljšati performanse, pouzdanost i funkcionalnost kvadkoptera, čineći ga sposobnijim za različite zadatke i izazove.

8. Zaključak

U ovom radu detaljno su analizirani i opisani procesi izrade i testiranja kvadkoptera, uključujući izbor komponenti, integracija sustava, te ispitivanje performansi leta. Kroz ovaj proces suočili smo se s nizom izazova i problema koje smo uspješno savladali, što je rezultiralo stabilnim i funkcionalnim kvadkopterom.

Tijekom testiranja kvadkoptera postigli smo nekoliko ključnih uspjeha. Kvadkopter je uspješno održavao stabilan let bez značajnih oscilacija zahvaljujući preciznoj kalibraciji senzora i optimizaciji PID kontrolera. Kontrola motora bila je precizna i pouzdana, što je omogućilo kvadkopteru da izvršava komande za manevriranje s visokim stupnjem točnosti. Također, postignuto je trajanje leta do 10 minuta, što je omogućilo dovoljno vremena za testiranje i različite operativne zadatke.

Pouzdana komunikacija između daljinskog upravljača i prijemnika osigurala je stabilnu kontrolu nad kvadkopterom, dok su sigurnosne značajke, poput fail-safe funkcije, dodatno povećale sigurnost tijekom leta. Integracija svih sustava kvadkoptera bila je učinkovita, što je omogućilo skladan rad svih komponenti i optimalne performanse letjelice.

Uočeno je nekoliko područja za daljnje poboljšanje. Precizno podešavanje PID kontrolera, optimizacija potrošnje energije i daljnje poboljšanje komunikacijskog sustava ključni su za postizanje još boljih performansi kvadkoptera. Dodatne nadogradnje, poput ugradnje naprednijih senzora, jačih motora, većih baterija, GPS modula i naprednih algoritama za stabilizaciju, mogu značajno poboljšati funkcionalnost i sposobnosti kvadkoptera.

Zaključno, u ovom projektu pokazano je da je napravljeni kvadkopter sposoban za stabilan i pouzdan let, te su identificirani ključni elementi za daljnji razvoj i poboljšanje. Kroz pažljivu analizu i kontinuirano testiranje, kvadkopter može biti prilagođen za širok raspon aplikacija, pružajući vrijedne uvide i alate za budući rad u području zrakoplovstva i robotike.

Literatura

1) Alapić, M., Velčić, I. (2018). Izvod jednadžbi diskretnog Kalmanovog filtera.
2) Addicore (bez datuma). GY-521 MPU6050 3-Axis Gyroscope and Accelerometer IMU. Preuzeto s https://www.addicore.com/products/gy-521-mpu6050-3-axis-gyroscope-and-accelerometer-imu.
3) Banzi, M., Shiloh, M. (2014). Getting Started with Arduino: The Open Source Electronics Prototyping Platform. 3rd Edition. Maker Media, Inc.
4) Beard, R. W., McLain, T. W. (2012). Small Unmanned Aircraft: Theory and Practice. Princeton University Press
5) Carbon Aeronautics (bez datuma). Preuzeto s https://github.com/CarbonAeronautics.
6) Circuit Digest (bez datuma). Everything You Need to Know About the FlySky FS-i6 Transmitter and Receiver for Drone Control. Preuzeto s https://circuitdigest.com/article/everything-you-need-to-know-about-the-flysky-fs-i6-transmitter-and-receiver-for-drone-control.
7) Circuito.io (bez datuma). Teensy Guide. Preuzeto s https://www.circuito.io/blog/teensy-guide/.
8) EPowerHobby (bez datuma). Everything You Need to Know About Electronic Speed Controllers. Preuzeto s https://epowerhobby.com/wp-content/uploads/2019/07/Everything-You-Need-to-Know-About-Electronic-Speed-Controllers.pdf
9) GEPRC (bez datuma). GEP-GR1105 Motor. Preuzeto s https://geprc.com/product/gep-gr1105-motor/.
10) HappyModel (2022). Bassline Spare Part EX1103 KV11000 Brushless Motor. Preuzeto s https://www.happymodel.cn/index.php/2022/09/05/bassline-spare-part-ex1103-kv11000-brushless-motor/.
11) Medium (bez datuma). How to Calibrate Gyroscope with MPU6050. Preuzeto s https://medium.com/@shilleh/how-to-calibrate-gyroscope-with-mpu6050-81b731540c31.
12) Oshwlab (bez datuma). Carbon Aeronautics Quadcopter. Preuzeto s https://oshwlab.com/droneer/Carbon-Aeronautics-Quadcopter.

13) PearsonCMG (bez datuma). Everything You Need to Know About Electronic Speed Controllers. Preuzeto s https://ptgmedia.pearsoncmg.com/images/9780789755988/samplepages/9780789755988.pdf.
14) PJRC (bez datuma). Teensy 4.0. Preuzeto s https://www.pjrc.com/store/teensy40.html.
15) ProtoSupplies (bez datuma). GY-BME280 Pressure, Humidity, Temperature Sensor Module. Preuzeto s https://protosupplies.com/product/gy-bme280-pressure-humidity-temperature-sensor-module/.
16) Squarespace (bez datuma). FS-i6 User Manual 20160819. Preuzeto s https://static1.squarespace.com/static/5bc852d6b9144934c40d499c/t/5c0787e10e2e721a7f17c998/1543997593953/FS-i6+User+manual+20160819.pdf.
17) Visioli, A. (2009). Practical PID Control.

POPIS SLIKA

POPIS TABELA

PRILOG – IZVORNI KOD

```
#include <Wire.h>
float RateRoll, RatePitch, RateYaw;
float RateCalibrationRoll, RateCalibrationPitch, RateCalibrationYaw;
int RateCalibrationNumber;
#include <PulsePosition.h>
PulsePositionInput ReceiverInput(RISING);
float ReceiverValue[]={0, 0, 0, 0, 0, 0, 0, 0};
int ChannelNumber=0;
float Voltage, Current, BatteryRemaining, BatteryAtStart;
float CurrentConsumed=0;
float BatteryDefault=1300;
uint32_t LoopTimer;
float DesiredRateRoll, DesiredRatePitch, DesiredRateYaw;
float ErrorRateRoll, ErrorRatePitch, ErrorRateYaw;
float InputRoll, InputThrottle, InputPitch, InputYaw;
float PrevErrorRateRoll, PrevErrorRatePitch, PrevErrorRateYaw;
float PrevItermRateRoll, PrevItermRatePitch, PrevItermRateYaw;
float PIDReturn[]={0, 0, 0};
float PRateRoll=0.6; float PRatePitch=PRateRoll; float PRateYaw=2;
float IRateRoll=3.5; float IRatePitch=IRateRoll; float IRateYaw=12;
float DRateRoll=0.03; float DRatePitch=DRateRoll; float DRateYaw=0;
float MotorInput1, MotorInput2, MotorInput3, MotorInput4;
float AccX, AccY, AccZ;
float AngleRoll, AnglePitch;
float KalmanAngleRoll=0, KalmanUncertaintyAngleRoll=2*2;
float KalmanAnglePitch=0, KalmanUncertaintyAnglePitch=2*2;
float Kalman1DOutput[]={0,0};
float DesiredAngleRoll, DesiredAnglePitch;
float ErrorAngleRoll, ErrorAnglePitch;
float PrevErrorAngleRoll, PrevErrorAnglePitch;
float PrevItermAngleRoll, PrevItermAnglePitch;
float PAngleRoll=2; float PAnglePitch=PAngleRoll;
float IAngleRoll=0; float IAnglePitch=IAngleRoll;
float DAngleRoll=0; float DAnglePitch=DAngleRoll;
uint16_t dig_T1, dig_P1;
int16_t dig_T2, dig_T3, dig_P2, dig_P3, dig_P4, dig_P5;
int16_t dig_P6, dig_P7, dig_P8, dig_P9;
float AltitudeBarometer, AltitudeBarometerStartUp;
float AccZInertial;
#include <BasicLinearAlgebra.h>
using namespace BLA;
float AltitudeKalman, VelocityVerticalKalman;
BLA::Matrix<2,2> F; BLA::Matrix<2,1> G;
BLA::Matrix<2,2> P; BLA::Matrix<2,2> Q;
BLA::Matrix<2,1> S; BLA::Matrix<1,2> H;
```

```
BLA::Matrix<2,2> I; BLA::Matrix<1,1> Acc;
BLA::Matrix<2,1> K; BLA::Matrix<1,1> R;
BLA::Matrix<1,1> L; BLA::Matrix<1,1> M;
float DesiredVelocityVertical, ErrorVelocityVertical;
float PVelocityVertical=3.5; float IVelocityVertical=0.0015; float
DVelocityVertical=0.01;
float PrevErrorVelocityVertical, PrevItermVelocityVertical;
void kalman_2d(void){
    Acc = {AccZInertial};
    S=F*S+G*Acc;
    P=F*P*~F+Q;
    L=H*P*~H+R;
    K=P*~H*Invert(L);
    M = {AltitudeBarometer};
    S=S+K*(M-H*S);
    AltitudeKalman=S(0,0);
    VelocityVerticalKalman=S(1,0);
    P=(I-K*H)*P;
}
void barometer_signals(void){
    Wire.beginTransmission(0x76);
    Wire.write(0xF7);
    Wire.endTransmission();
    Wire.requestFrom(0x76,6);
    uint32_t press_msb = Wire.read();
    uint32_t press_lsb = Wire.read();
    uint32_t press_xlsb = Wire.read();
    uint32_t temp_msb = Wire.read();
    uint32_t temp_lsb = Wire.read();
    uint32_t temp_xlsb = Wire.read();
    unsigned long int adc_P = (press_msb << 12) | (press_lsb << 4) |
(press_xlsb >>4);
    unsigned long int adc_T = (temp_msb << 12) | (temp_lsb << 4) |
(temp_xlsb >>4);
    signed long int var1, var2;
    var1 = ((((adc_T >> 3) - ((signed long int )dig_T1 <<1)))* ((signed long
int )dig_T2)) >> 11;
    var2 = (((((adc_T >> 4) - ((signed long int )dig_T1 )) * ((adc_T>>4) -
((signed long int )dig_T1))) >> 12) * ((signed long int )dig_T3)) >> 14;
    signed long int t_fine = var1 + var2;
    unsigned long int p;
    var1 = (((signed long int )t_fine)>>1) - (signed long int )64000;
    var2 = (((var1>>2) * (var1>>2)) >> 11) * ((signed long int )dig_P6);
    var2 = var2 + ((var1*((signed long int )dig_P5)) <<1);
    var2 = (var2>>2)+(((signed long int )dig_P4) <<16);
    var1 = (((dig_P3 * (((var1>>2)*(var1>>2)) >> 13 ))>>3)+((((signed long
int )dig_P2) * var1)>>1))>>18;
    var1 = ((((32768+var1))*((signed long int )dig_P1)) >>15);
```

```
    if (var1 == 0) { p=0;}
    p = (((unsigned long int )(((signed long int ) 1048576)-adc_P)-
(var2>>12)))*3125;
    if(p<0x80000000){ p = (p << 1) / ((unsigned long int ) var1);}
    else { p = (p / (unsigned long int )var1) * 2; }
    var1 = (((signed long int )dig_P9) * ((signed long int ) (((p>>3) *
(p>>3))>>13)))>>12;
    var2 = (((signed long int )(p>>2)) * ((signed long int )dig_P8))>>13;
    p = (unsigned long int)((signed long int )p + ((var1 + var2+ dig_P7) >>
4));
    double pressure=(double)p/100;
    AltitudeBarometer=44330*(1-pow(pressure /1013.25, 1/5.255))*100;
}
void kalman_1d(float KalmanState, float KalmanUncertainty, float
KalmanInput, float KalmanMeasurement) {
    KalmanState=KalmanState+0.004*KalmanInput;
    KalmanUncertainty=KalmanUncertainty + 0.004 * 0.004 * 4 * 4;
    float KalmanGain=KalmanUncertainty * 1/(1*KalmanUncertainty + 3 * 3);
    KalmanState=KalmanState+KalmanGain * (KalmanMeasurement-KalmanState);
    KalmanUncertainty=(1-KalmanGain) * KalmanUncertainty;
    Kalman1DOutput[0]=KalmanState;
    Kalman1DOutput[1]=KalmanUncertainty;
}
void battery_voltage(void) {
    Voltage=(float)analogRead(15)/62;
    Current=(float)analogRead(21)*0.089;
}
void read_receiver(void){
    ChannelNumber = ReceiverInput.available();
    if (ChannelNumber > 0) {
        for (int i=1; i<=ChannelNumber;i++){
            ReceiverValue[i-1]=ReceiverInput.read(i);
        }
    }
}
void gyro_signals(void) {
    Wire.beginTransmission(0x68);
    Wire.write(0x1A);
    Wire.write(0x05);
    Wire.endTransmission();
    Wire.beginTransmission(0x68);
    Wire.write(0x1C);
    Wire.write(0x10);
    Wire.endTransmission();
    Wire.beginTransmission(0x68);
    Wire.write(0x3B);
    Wire.endTransmission();
    Wire.requestFrom(0x68,6);
```

```
    int16_t AccXLSB = Wire.read() << 8 | Wire.read();
    int16_t AccYLSB = Wire.read() << 8 | Wire.read();
    int16_t AccZLSB = Wire.read() << 8 | Wire.read();
    Wire.beginTransmission(0x68);
    Wire.write(0x1B);
    Wire.write(0x8);
    Wire.endTransmission();
    Wire.beginTransmission(0x68);
    Wire.write(0x43);
    Wire.endTransmission();
    Wire.requestFrom(0x68,6);
    int16_t GyroX=Wire.read()<<8 | Wire.read();
    int16_t GyroY=Wire.read()<<8 | Wire.read();
    int16_t GyroZ=Wire.read()<<8 | Wire.read();
    RateRoll=(float)GyroX/65.5;
    RatePitch=(float)GyroY/65.5;
    RateYaw=(float)GyroZ/65.5;
    AccX=(float)AccXLSB/4096-0.05;
    AccY=(float)AccYLSB/4096+0.01;
    AccZ=(float)AccZLSB/4096-0.11;
    AngleRoll=atan(AccY/sqrt(AccX*AccX+AccZ* AccZ))*1/(3.142/180);
    AnglePitch=-atan(AccX/sqrt(AccY*AccY+AccZ* AccZ))*1/(3.142/180);
}
void pid_equation(float Error, float P , float I, float D, float PrevError,
float PrevIterm) {
    float Pterm=P*Error;
    float Iterm=PrevIterm+I*(Error+PrevError)*0.004/2;
    if (Iterm > 400) Iterm=400;
    else if (Iterm <-400) Iterm=-400;
    float Dterm=D*(Error-PrevError)/0.004;
    float PIDOutput= Pterm+Iterm+Dterm;
    if (PIDOutput>400) PIDOutput=400;
    else if (PIDOutput <-400) PIDOutput=-400;
    PIDReturn[0]=PIDOutput;
    PIDReturn[1]=Error;
    PIDReturn[2]=Iterm;
}
void reset_pid(void) {
    PrevErrorRateRoll=0; PrevErrorRatePitch=0; PrevErrorRateYaw=0;
    PrevItermRateRoll=0; PrevItermRatePitch=0; PrevItermRateYaw=0;
    PrevErrorAngleRoll=0; PrevErrorAnglePitch=0;
    PrevItermAngleRoll=0; PrevItermAnglePitch=0;
    PrevErrorVelocityVertical=0; PrevItermVelocityVertical=0;
}
void setup() {
    pinMode(5, OUTPUT);
    digitalWrite(5, HIGH);
    pinMode(13, OUTPUT);
```

```
    digitalWrite(13, HIGH);
    Wire.setClock(400000);
    Wire.begin();
    delay(250);
    Wire.beginTransmission(0x68);
    Wire.write(0x6B);
    Wire.write(0x00);
    Wire.endTransmission();
    Wire.beginTransmission(0x76);
    Wire.write(0xF4);
    Wire.write(0x57);
    Wire.endTransmission();
    Wire.beginTransmission(0x76);
    Wire.write(0xF5);
    Wire.write(0x14);
    Wire.endTransmission();
    uint8_t data[24], i=0;
    Wire.beginTransmission(0x76);
    Wire.write(0x88);
    Wire.endTransmission();
    Wire.requestFrom(0x76,24);
    while(Wire.available()){
        data[i] = Wire.read();
        i++;
    }
    dig_T1 = (data[1] << 8) | data[0];
    dig_T2 = (data[3] << 8) | data[2];
    dig_T3 = (data[5] << 8) | data[4];
    dig_P1 = (data[7] << 8) | data[6];
    dig_P2 = (data[9] << 8) | data[8];
    dig_P3 = (data[11]<< 8) | data[10];
    dig_P4 = (data[13]<< 8) | data[12];
    dig_P5 = (data[15]<< 8) | data[14];
    dig_P6 = (data[17]<< 8) | data[16];
    dig_P7 = (data[19]<< 8) | data[18];
    dig_P8 = (data[21]<< 8) | data[20];
    dig_P9 = (data[23]<< 8) | data[22]; delay(250);
    for (RateCalibrationNumber=0; RateCalibrationNumber<2000;
RateCalibrationNumber ++) {
        gyro_signals();
        RateCalibrationRoll+=RateRoll;
        RateCalibrationPitch+=RatePitch;
        RateCalibrationYaw+=RateYaw;
        barometer_signals();
        AltitudeBarometerStartUp+= AltitudeBarometer; delay(1);
    }
    RateCalibrationRoll/=2000;
    RateCalibrationPitch/=2000;
```

```
    RateCalibrationYaw/=2000;
    AltitudeBarometerStartUp/=2000;
    F = {1, 0.004, 0, 1};
    G = {0.5*0.004*0.004, 0.004};
    H = {1, 0};
    I = {1, 0, 0, 1};
    Q = G * ~G*10*10;
    R = {30*30};
    P = {0, 0, 0, 0};
    S = {0, 0};
    analogWriteFrequency(1, 250);
    analogWriteFrequency(2, 250);
    analogWriteFrequency(3, 250);
    analogWriteFrequency(4, 250);
    analogWriteResolution(12);
    pinMode(6, OUTPUT);
    digitalWrite(6, HIGH);
    battery_voltage();
    if (Voltage > 8.3) { digitalWrite(5, LOW);
        BatteryAtStart=BatteryDefault; }
    else if (Voltage < 7.5) {
        BatteryAtStart=30/100*BatteryDefault ;}
    else { digitalWrite(5, LOW);
        BatteryAtStart=(82*Voltage-580)/100*BatteryDefault; }
    ReceiverInput.begin(14);
    while (ReceiverValue[2] < 1020 || ReceiverValue[2] > 1050) {
        read_receiver();
        delay(4);
    }
    LoopTimer=micros();
}
void loop() {
    gyro_signals();
    RateRoll-=RateCalibrationRoll;
    RatePitch-=RateCalibrationPitch;
    RateYaw-=RateCalibrationYaw;
    kalman_1d(KalmanAngleRoll, KalmanUncertaintyAngleRoll, RateRoll,
AngleRoll);
    KalmanAngleRoll=Kalman1DOutput[0];
    KalmanUncertaintyAngleRoll=Kalman1DOutput[1];
    kalman_1d(KalmanAnglePitch, KalmanUncertaintyAnglePitch, RatePitch,
AnglePitch);
    KalmanAnglePitch=Kalman1DOutput[0];
    KalmanUncertaintyAnglePitch=Kalman1DOutput[1];
    AccZInertial=-sin(AnglePitch*(3.142/180))*AccX
+cos(AnglePitch*(3.142/180))*sin(AngleRoll* (3.142/180))*
AccY+cos(AnglePitch*(3.142/180))* cos(AngleRoll*(3.142/180))*AccZ;
    AccZInertial=(AccZInertial-1)*9.81*100;
```

```
    barometer_signals();
    AltitudeBarometer-=AltitudeBarometerStartUp;
    kalman_2d();
    read_receiver();
    DesiredAngleRoll=0.10*(ReceiverValue[0]-1500);
    DesiredAnglePitch=0.10*(ReceiverValue[1]-1500);
    DesiredRateYaw=0.15*(ReceiverValue[3]-1500);
    DesiredVelocityVertical=0.3*(ReceiverValue[2]- 1500);
    ErrorVelocityVertical=DesiredVelocityVertical- VelocityVerticalKalman;
    pid_equation(ErrorVelocityVertical, PVelocityVertical,
IVelocityVertical, DVelocityVertical, PrevErrorVelocityVertical,
PrevItermVelocityVertical);
    InputThrottle=1500+PIDReturn[0];
    PrevErrorVelocityVertical=PIDReturn[1];
    PrevItermVelocityVertical=PIDReturn[2];
    ErrorAngleRoll=DesiredAngleRoll- KalmanAngleRoll;
    ErrorAnglePitch=DesiredAnglePitch- KalmanAnglePitch;
    pid_equation(ErrorAngleRoll, PAngleRoll, IAngleRoll, DAngleRoll,
PrevErrorAngleRoll, PrevItermAngleRoll);
    DesiredRateRoll=PIDReturn[0];
    PrevErrorAngleRoll=PIDReturn[1];
    PrevItermAngleRoll=PIDReturn[2];
    pid_equation(ErrorAnglePitch, PAnglePitch, IAnglePitch, DAnglePitch,
PrevErrorAnglePitch, PrevItermAnglePitch);
    DesiredRatePitch=PIDReturn[0];
    PrevErrorAnglePitch=PIDReturn[1];
    PrevItermAnglePitch=PIDReturn[2];
    ErrorRateRoll=DesiredRateRoll-RateRoll;
    ErrorRatePitch=DesiredRatePitch-RatePitch;
    ErrorRateYaw=DesiredRateYaw-RateYaw;
    pid_equation(ErrorRateRoll, PRateRoll, IRateRoll, DRateRoll,
PrevErrorRateRoll, PrevItermRateRoll);
    InputRoll=PIDReturn[0];
    PrevErrorRateRoll=PIDReturn[1];
    PrevItermRateRoll=PIDReturn[2];
    pid_equation(ErrorRatePitch, PRatePitch, IRatePitch, DRatePitch,
PrevErrorRatePitch, PrevItermRatePitch);
    InputPitch=PIDReturn[0];
    PrevErrorRatePitch=PIDReturn[1];
    PrevItermRatePitch=PIDReturn[2];
    pid_equation(ErrorRateYaw, PRateYaw, IRateYaw, DRateYaw,
PrevErrorRateYaw, PrevItermRateYaw);
    InputYaw=PIDReturn[0];
    PrevErrorRateYaw=PIDReturn[1];
    PrevItermRateYaw=PIDReturn[2];
    if (InputThrottle > 1800) InputThrottle = 1800;
    MotorInput1= 1.024*(InputThrottle-InputPitch- InputRoll-InputYaw);
    MotorInput2= 1.024*(InputThrottle+InputPitch- InputRoll+InputYaw);
```

```
    MotorInput3= 1.024*(InputThrottle+InputPitch+ InputRoll-InputYaw);
    MotorInput4= 1.024*(InputThrottle-InputPitch+ InputRoll+InputYaw);
    if (MotorInput1 > 2000)MotorInput1 = 1999;
    if (MotorInput2 > 2000)MotorInput2 = 1999;
    if (MotorInput3 > 2000)MotorInput3 = 1999;
    if (MotorInput4 > 2000)MotorInput4 = 1999;
    int ThrottleIdle=1180;
    if (MotorInput1 < ThrottleIdle) MotorInput1 = ThrottleIdle;
    if (MotorInput2 < ThrottleIdle) MotorInput2 = ThrottleIdle;
    if (MotorInput3 < ThrottleIdle) MotorInput3 = ThrottleIdle;
    if (MotorInput4 < ThrottleIdle) MotorInput4 = ThrottleIdle;
    int ThrottleCutOff=1000;
    if (ReceiverValue[2]<1050) {
        MotorInput1=ThrottleCutOff;
        MotorInput2=ThrottleCutOff;
        MotorInput3=ThrottleCutOff;
        MotorInput4=ThrottleCutOff;
        reset_pid();
    }
    analogWrite(1,MotorInput1);
    analogWrite(2,MotorInput2);
    analogWrite(3,MotorInput3);
    analogWrite(4,MotorInput4);
    battery_voltage();
    CurrentConsumed=Current*1000*0.004/3600+CurrentConsumed;
    BatteryRemaining=(BatteryAtStart-CurrentConsumed)/BatteryDefault*100;
    if (BatteryRemaining<=30) digitalWrite(5, HIGH);
    else digitalWrite(5, LOW);
    while (micros() - LoopTimer < 4000);
    LoopTimer=micros();
}
```