Grafičko korisničko sučelje za upravljanje, prikupljanje i prikaz signala iz ADCS prototipa

Lukić, Neven

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:295975

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-19**

Repository / Repozitorij:

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1316

GRAFIČKO KORISNIČKO SUČELJE ZA UPRAVLJANJE, PRIKUPLJANJE I PRIKAZ SIGNALA IZ ADCS PROTOTIPA

Neven Lukić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1316

GRAFIČKO KORISNIČKO SUČELJE ZA UPRAVLJANJE, PRIKUPLJANJE I PRIKAZ SIGNALA IZ ADCS PROTOTIPA

Neven Lukić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1316

Pristupnik:	**Neven Lukić (0036541471)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Josip Lončar

Zadatak: **Grafičko korisničko sučelje za upravljanje, prikupljanje i prikaz signala iz ADCS prototipa**

Opis zadatka:

Sustav za određivanje i upravljanje orijentacijom satelita (engl. Attitude Determination and Control System, ADCS) je kompleksan sustav koji se sastoji se od više senzora, aktuatora i upravljačke jedinice. Zadatak ovoga završnog rada je razviti i implementirati grafičko korisničko sučelje koje omogućuje komunikaciju i upravljanje ADCS prototipom. Sučelje omogućuje povezivanje s ADCS prototipom putem Bluetooth veze, prikazivanje signala senzora inercijske mjerne jedinice (akcelerometra, magnetometra i žiroskopa), prikazivanje orijentacije ADCS prototipa, daljinsko upravljanje zamašnjacima i koračnim motorima ADCS prototipa, te kalibraciju prototipa. Grafičko korisničko sučelje je potrebno implementirati koristeći se Python bibliotekom PyQt vodeći računa o skalabilnosti i ograničenjima vezanim uz rad u stvarnom vremenu. Potpunu funkcionalnost grafičkog sučelja je potrebno demonstrirati u laboratorijskim uvjetima koristeći se razvijenim ADCS prototipom.

Rok za predaju rada: 14. lipnja 2024.

Sadržaj

1. Uvod

1.1. Mali sateliti

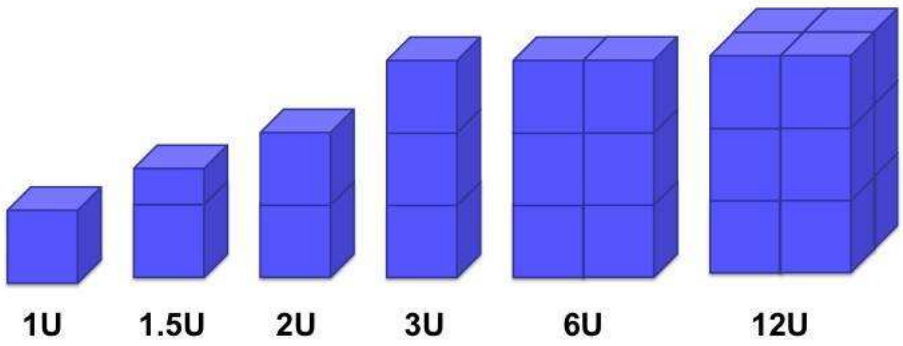
Veličina i cijena satelita danas iznimno varira i oni dolaze u raznim formatima. Od manjih satelita koje možemo držati pa do satelita koji su veličine autobusa. NASA (engl. *North American Space Agency*) kao male satelite (eng. *SmallSats*) kategorizira one koji imaju masu manju od 180 kilograma i nisu veći od otprilike polovice kuhinjskog hladnjaka [1]. Podjelu je moguće vidjeti u tablici 1.1. Točna podjela satelita na kategorije može varirati od izvora do izvora pa tako na primjer FAA (engl. *Federal Aviation Administration*) kao malene satelite kategorizira one koji imaju masu manju od 1200 kilograma.

1.2. CubeSat sateliti

Sateliti CubeSat spadaju u kategoriju nanosatelita i imaju standardiziranu veličinu i oblik. Standardna veličina CubeSat satelita je "jedna jedinica" ili "1U" gdje "U" dolazi iz engl. *Unit*. Jedna jedinica je 10x10x10 centimetara. Jedinice se mogu kombinirati i spajati u različite konfiguracije satelite veličine 2U, 3U, 6U i 12U koje su prikazane na slici 1.1. Format satelita je razvijen na Američkim sveučilištima Polytechnic State University at San Luis Obispo (Cal Poly) i Stanford University u 1999. Format je standardiziran kako

Naziv kategorija	Masa satelita
Minisatellite	100 - 180 kilograma
Microsatellite	10 - 100 kilograma
Nanosatellite	1 - 10 kilograma
Picosatellite	0.01 - 1 kilograma
Femtosatellite	0.001 - 0.01 kilograma

Tablica 1.1. Podjela malih satelita na kategorije

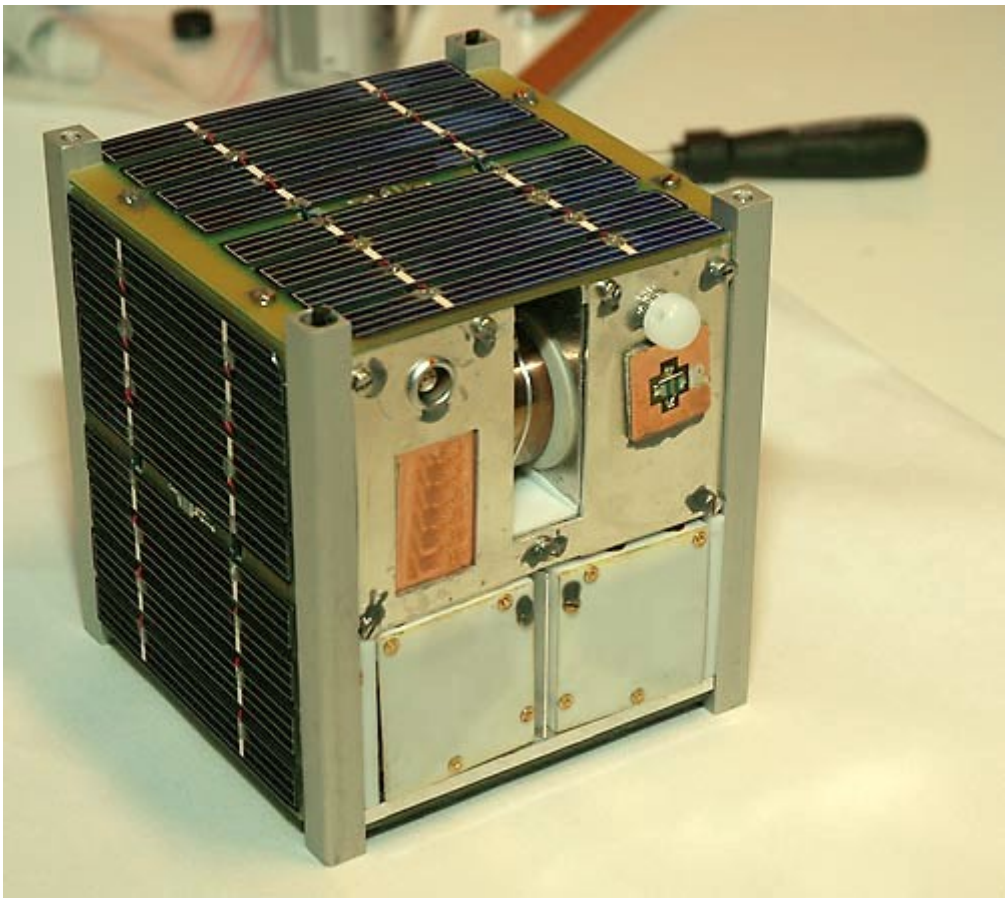


Slika 1.1. Prikaz raznih konfiguracija CubeSat satelita

bi pružio platformu za istraživanje svemira koja je dostupna obrazovnim ustanovama i manjim laboratorijima. Standardizirani oblik i dostupnost satelita su rezultirali naglim razvojem tehnologije i industrije CubeSat satelita te sada postoji zasebna grana znanosti, regulacije i istraživanja koja se bave isključivo CubeSat satelitima.

1.2.1. Prednosti CubeSat satelita

Standardizirani oblik satelita, njegovih dijelova, ograničenja na volumen i masu donose mnoge prednosti [2]. Razvoj je jednostavniji, kraći i ne zahtijeva toliko resursa u odnosu na razvoj većih satelita. Te činjenice omogućavaju znanstvenim i obrazovnim institucijama da imaju pristup i mogućnost rada s takvim satelitima. Kraći razvojni ciklus omogućava brže iteracije, razvijanje i testiranje novih ideja. Nisu potrebni deseci godina za razvoj i planiranje pa se tako neki sateliti mogu razviti i lansirati unutar dvije godine [3]. Brzi razvoj znanstvenicima omogućava izradu različitih korisnih tereta (engl. *Payload*) koji će raditi neko mjerenje. Tako sateliti mogu imati puno različitih ciljeva. Još jedna posljedica niže cijene razvoja i lansiranja je ta da se može lansirati veći broj satelita koji se organiziraju u konstelacije. Uz sve to, manja masa i veličina kao i manji broj novih i ne verificiranih dijelova smanjuju rizike prilikom lansiranja. Za kraj, kako su CubeSat sateliti manji mogu se inkorporirati u druge misije kako bi popunili prazan prostor



Slika 1.2. Fotonaponske ćelije na CubeSatu

koji nastaje lansiranjem nekog većeg satelita. To smanjuje troškove satelita, ali s druge strane stvara nove rizike pa sateliti mora biti temeljito ispitan prije nego što će se lansirati zajedno s ostalim satelitima, bili oni CubeSat ili nekog značajno većeg formata.

1.2.2. Izazovi CubeSat satelita

Limitirani veličinom, satelit mora biti optimalno ispunjen što dovodi do izazova. Prostor za pohranu energije nije velik, satelit mora podržati razne podsustave koji služe za njegovo održavanja na životu kao i koristan teret koji provodi mjerenje. Kako je energetski budžet vrlo malen, na satelitima su najčešće ograđene fotonaponske ćelije kao što je moguće vidjeti u slici 1.2. Fotonaponske ćelije zauzimaju nekoliko strana satelita, a ako je naš korisni teret neka kamera i misija je snimanje neke točke na Zemlji onda moramo imati sustav za određivanje i upravljanje visinom i orijentacijom satelita (engl. *Attitude determination and control systems - ADCS*).

2. Upravljanje orijentacijom satelita

2.1. Uvod

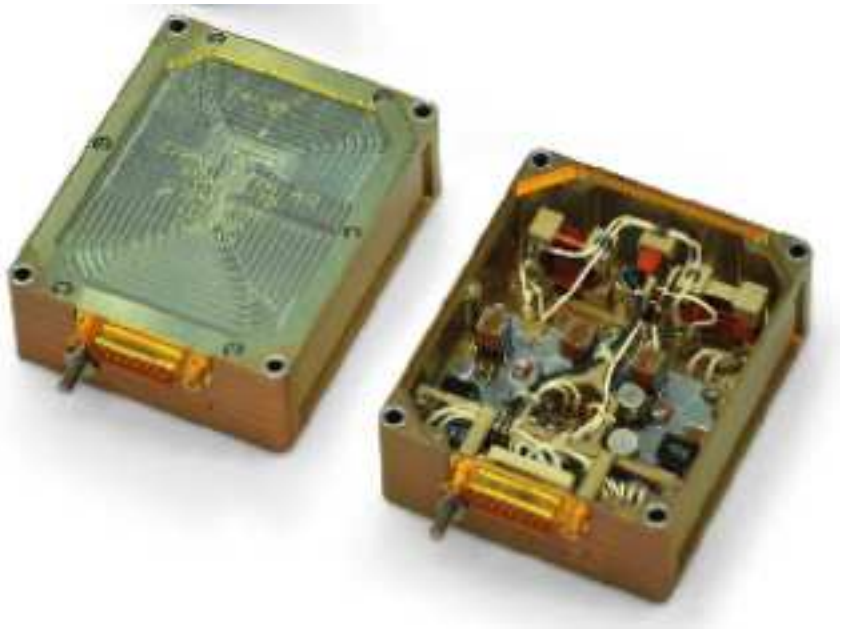
Sustav satelita koji autonomno na satelitu određuje orijentaciju te ju u stvarnom vremenu ispravlja prema ciljevima misije. Također, moguće je zadati nove ciljeve satelitu pa on prema tome mora promijeniti orijentaciju. Uz to ovaj sustav prikuplja podatke koje obrađuje i može ih slati uz ostale telemetrijske podatke na zemaljsku stanicu kako bi voditelji misije znali što se dešava sa samim satelitom.

2.2. Određivanje orijentacije i upravljanje orijentacijom

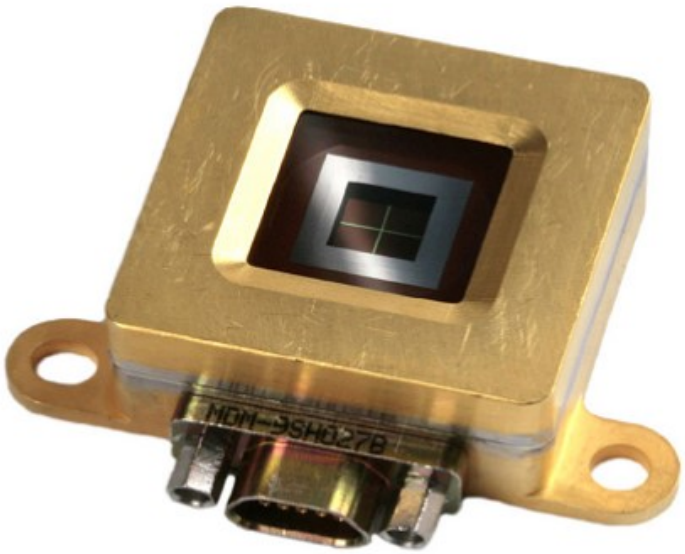
Sustav konstantno dobiva informacije iz nekih senzora poput magnetometra na slici 2.1. ili senzora sunca na slici 2.2. Te informacije se obrađuju, iz njih se određuje trenutna orijentacija, ona se uspoređuje sa željenom te se izračunava pogreška (engl. *error*). To je mjera koliko je trenutna orijentacija daleko od željene. Na temelju izračunate pogreške koriste se aktuatori poput istosmjernih motora (engl. *Direct current motor - DC Motor*) na slici 2.3. ili magnetorkera na slici 2.4. kako bi promijenili orijentaciju satelita.

2.2.1. Načini rada ADCS sustava

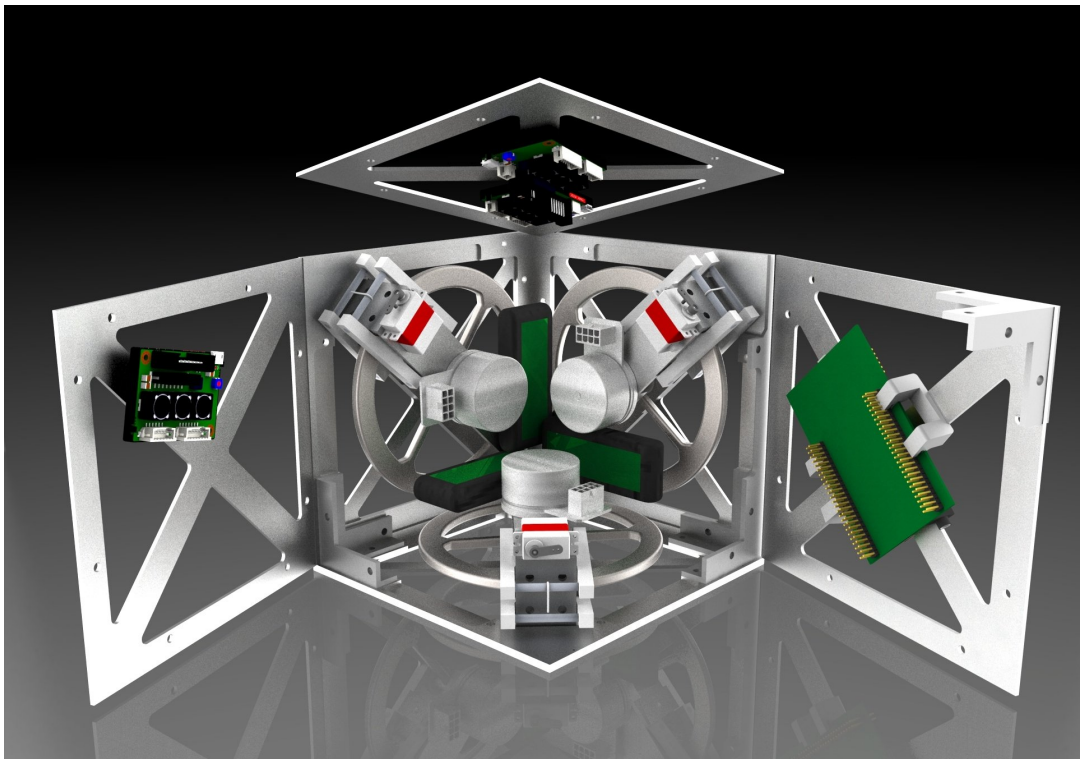
Kako satelit prolazi kroz nekoliko glavnih faza tijekom svojeg postojanja ADCS sustav mora moći raditi u različitim načinima kako bi satelit ispunio svoj zadataka. Glavni načini rada ADCS sustava su manevar stabiliziranja i uklanjanja neželjenih vrtnji (engl. *Detumbling*), aktivno usmjeravanje prema potrebi misije (engl. *Nominal Mode*) i usmjeravanje prema suncu u slučaju potreba punjenja baterija (engl. *Sun Safe Mode*) [4]. Kako bi satelit imao mogućnosti potrebne da ispuni svoj zadatak potrebno je razviti meha-



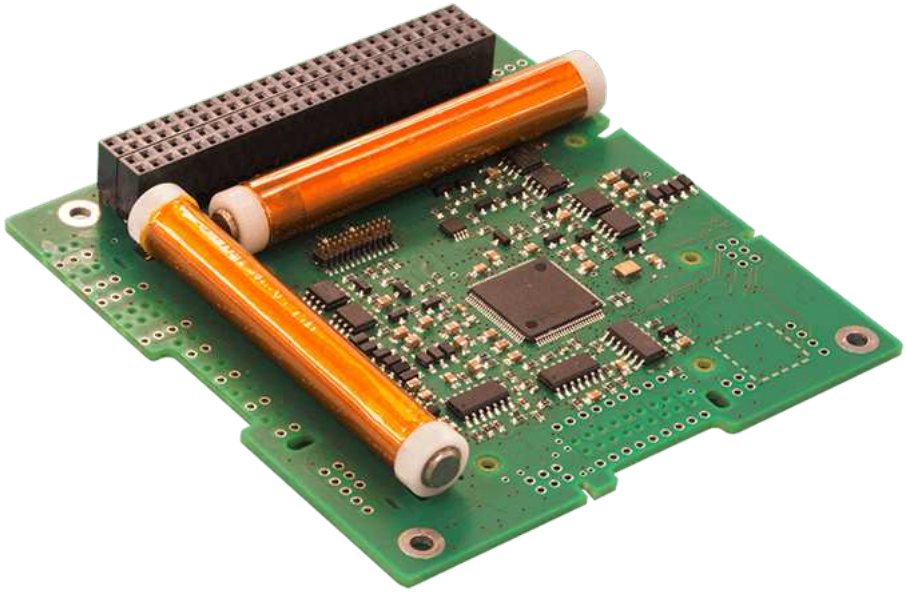
Slika 2.1. Magnetometar



Slika 2.2. Senzor sunca



Slika 2.3. Zamašnjaci s pogonom DC Motora



Slika 2.4. Magnetorker

ničke dijelove, elektroniku, softver za elektroniku i upravljačke algoritme za određivanje i upravljanje orijentacijom.

2.3. Trenutno razvijeni sustav

Ovaj rad je dio veće cjeline, a to je projekt pod trenutnim vodstvom doc. dr. sc. Josipa Lončara na kojem sudjeluju i ostali studenti a cilj je izraditi vlastiti ADCS sustav od osnovnih komponenti i demonstrirati ispravno ponašanje sustava. Ovo spominjem zato što je korisno imati širu sliku cijelog projekta kako bi mogli uklopiti ovaj rad u ostatak priče. Trenutni prototip vidljiv na slici 2.5. Paralelno s razvojem programske podrške za upravljanje orijentacijom ostatka tima je radio, i u ovom trenutku još uvijek radi, na razvoju sljedeće verzije ADCS sustava koja će nam omogućiti upravljanje orijentacijom oko sve tri osi rotacije.

2.3.1. Mehanički dizajn

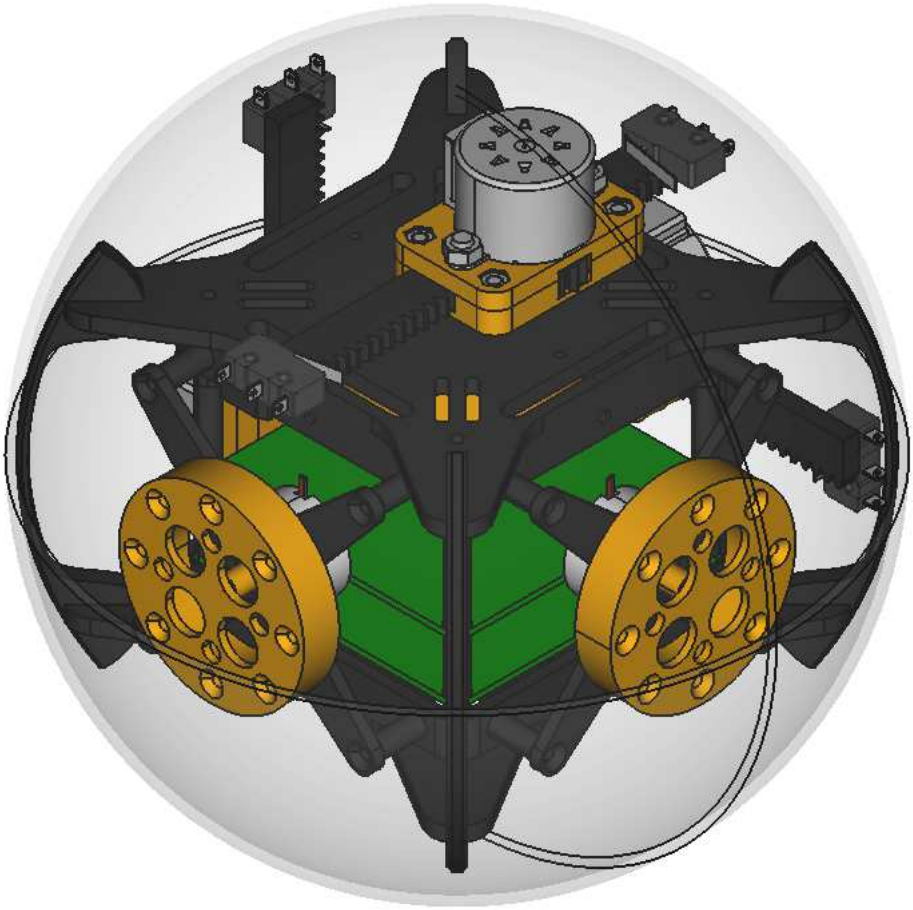
Prozirna sfera vidljiva na slici 2.5. je dio eksperimentalne postave koja nam omogućuje simuliranje uvjeta bez trenja. Naime, postolje koje drži plastičnu sferu odgovara zaobljenosti sfere te u sredini sadrži malu rupicu. Kroz tu rupicu se ispuhuje zrak koji dolazi iz kompresora te se stvara efekt lebdenja. Zrak stvara zračni jastuk između sfere i postolja te se s time smanjuje trenje na gotovo zanemariv iznos. Ta postava nam je bitna za simuliranje pravog okruženja zato što u vakuumu svemira nemamo trenje.

Verzija 2 ADCS sustav, vidljiva na slici 2.6., se sastoji od okvira u kojem se nalaze komponente: tiskane pločice sa senzorima, mikroračunalom i potrebnim hardverom za upravljanje aktuatorima, tri DC motori koji služe kao aktuatori (svaki za jednu os), tri koračna motori za balansiranje satelita i baterije.

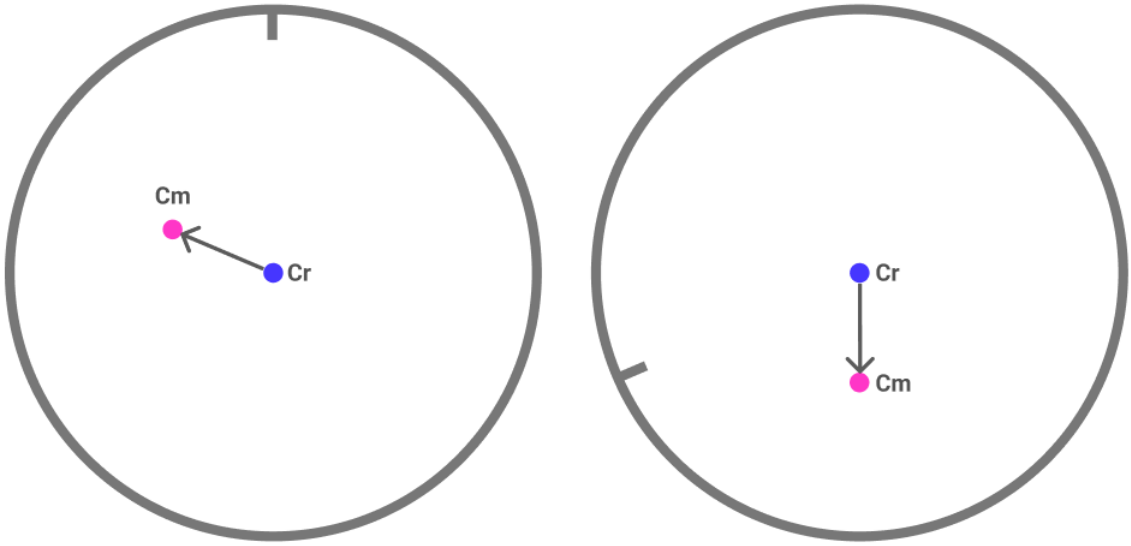
2.3.2. Senzori

Kao senzor za mjerenje podataka koristi se inercijalna mjerna jedinica (engl. *Inertial measurement unit - IMU*) koja mjeri troosne podatke gravitacijske akceleracije, kutne akceleracije, iznos magnetometara i iznos temperature koji nam u ovom trenutku nije toliko bitan za ostvarivanje zadatka.

Slika 2.5. Verzija 1 prototipa ADCS sustava



Slika 2.6. 3D model 2. verzije prototipa ADCS sustava



Slika 2.7. Prikaz utjecaj centra mase izvan centra rotacije

2.3.3. Aktuatori

Unutar sustava se nalaze tri DC motora na koje su pričvršćeni zamašnjaci. Aktivacijom DC motora, kreću se rotirati i zamašnjaci koji imaju neku masu. Zbog očuvanja kutne količine gibanja rotacija zamašnjaka uzrokuje rotaciju samog satelita u suprotnom smjeru. Na taj način imamo kontrolu orijentacije satelita.

Kako testiranja radimo na Zemlji i ne možemo lagano i praktično simulirati beztežinsko stanje moramo se pobrinuti za gravitaciju. Ovdje nam je to bitno zato što da bi dobili točno upravljanje orijentacije centar mase nam mora biti u centru rotacije satelita. Na slici 2.7. lijevo je prikazan sustav u kojem centar mase (Cm na ilustraciji) nije u centru rotacije (Cr na ilustraciji). Centar mase je fiksiran oblikom sfere u kojoj se sam satelit nalazi i on je smješten u centar te sfere. Sustav želi doći u stabilno stanje, a to stanje je da se centar mase žali smjestiti direktno ispod centra rotacije i time ćemo neželjenu rotaciju. Nakon stabiliziranja sustava, naš ADCS će se neželjeno zarotirati što je prikazano na slici 2.7. desno. Dio ovog problema možemo izbjeći pomnim dizajniranjem tako da masu u sustavu rasporedimo podjednako no to neće uvijek biti moguće niti dovoljno. Za preostalo balansiranje tu su koračni motori na kojima je pričvršćen mali zupčanik. Taj zupčanik u suradnji s drugim linearnim zupčanikom vidljivim na vrhu sustava u slici 2.6. omogućuje linearno pomicanje tog koračnog motora. To nam omogućava precizno pomicanje mase unutar sustava kako bi centar mase namjestili unutar centra rotacije.

2.3.4. Ugradbeni programska podrška

Ugradbena programska podrška je odgovorna za čitanje podataka iz senzora, njihovu obradu i slanje programskoj podršci putem tehnologije Bluetooth. Uz to, ugradbeni softver je zaslužan za primanje upravljačkih naredbi od programske podrške, njihovo interpretiranje i izvršavanje potrebnih zadataka poput ubrzavanja DC motora ili pomicanja koračnih motora.

U ovom projektu je odabrano da će se upravljački algoritmi izvršavati na udaljenom računalu koje će s pomoću Bluetootha komunicirati s ADCS sustavom. Tako možemo brže razvijati i testirati različite upravljačke algoritme bez da moramo rastavljati ADCS sustav, prepisivati MathLab ili Python kod u C ili C++ i reprogramirati mikrokontroler.

3. Programska podrška za upravljanje, prikupljanje i prikaz signala iz ADCS prototipa

3.1. Opis

Programska programska podrška je ostvarena u obliku grafičkog korisničkog sučelja (engl. *Graphical user interface - GUI*). Program je samostalna cjelina koja podržava mogućnosti ostvarivanje Bluetooth veze, prikupljanja i obrade podataka koji dolaze od ADCS sustava, grafički prikaz i obradu prikupljenih podataka. Uz sve to tu je i dio koji korisnicima omogućuje slanje korisničkih signala prema ADCS sustavu kao i podrška za kasnije implmentaciju izvršavanja automatskih algoritama upravljanja. Program je realiziran u Python programskom jeziku koristeći biblioteku PyQt6 za izradu korisničkog sučelja. Svrha ovog programa je da omogući efikasan način skupljanja i vizualizacije podataka kao i mjesto za razvoj novih algoritama za upravljanje orijentacijom. U programu je moguće unijeti opcije za ostvarivanje Bluetooth veze, upravljanje tom vezom, prikupljanje i parsiranje dolaznih podataka, vizualizacija podataka, unos parametara potrebnih za upravljanje ADCS sustavom kao, i slanje upravljačkih naredbi putem iste Bluetooth veze.

3.2. Korištene tehnologije

3.2.1. Python

Programski jezik Python je vrlo brz za izradu prototipa, dovoljno zreo za izradu cijelih programa, u ovom slučaju grafičkog korisničkog sučelja, i kasnije dovoljno fleksibilan za uređivanja kroz novije verzije. Jednostavna i pristupačna sintaksa omogućuje visoku čitljivost. Nudi podršku za objektno oblikovanu paradigmu što olakšava izradu većih

projekata. No ima i svoje nedostatke, jezik nije statički tipiziran što pri velikom projektu može stvarati probleme u razvoju. Bez korištenja `typing` biblioteke podrška za tipove je dosta slaba i iskustvo za programere nije odlično. Biblioteka `typing` nudi način da se varijable dodatno anotiraju što današnji pametna ugrađena razvojna okruženja (engl. *Integrated development environment - IDE*) znaju prepoznati i javiti kojeg je tipa trenutna varijable ili koje sve tipove neka funkcija očekuje. To je svakako unapređenje, ali iz osobnog iskustva pri velikom projektu i dalje zna doći do gdje se anotacije tipova gube ako se na primjer, radi višestruko nasljeđivanje ili neki drugi način pisanje kompliciranijeg koda gdje IDE ne može razumjeti semantiku vašeg koda.

3.2.2. PyQt6 i Qt

PyQt6 je naziv biblioteke koja je omotač C++ biblioteke Qt verzije 6 [5]. Pozivom funkcija iz biblioteke PyQt6 pozivaju se funkcije iz slično zvane C++ biblioteke. Qt biblioteka radi na više platforma (engl. *Cross platform*) i nudi visoko aplikacijsko programsko sučelje (engl. *Application programming interface - API*) za interakciju s modernim sustavima desktop i mobilnih uređaja. Qt omogućava laganu interakciju sa servisima lokacije, multimedije, upravljanje Bluetooth vezama i izradu nativnog grafičkog korisničkog sučelja [6]. Kako svaka od ovih funkcionalnosti može biti implementirana različito na različitim operacijskim ustavima, Qt se brine da programer ne mora mislit o tome već samo razvija svoju aplikaciju kao da cilja jednu platformu.

PyQt6 je omotač te nam omogućava da koristimo Python programski jezik za razvoj naših programa, a da ne moramo brinuti o platformama ili C++ kodu. Također, C++ kod je iznimno performantan i puno brži za razliku od Pythona pa je odlično da se s stvarima vezanim uz hardver bavi C++ a ne Python. S ovime imamo brzinu i jednostavnost razvoja u Pythonu, a opet imamo i brzinu izvođenja C++ koda koji obavlja za nas teški posao komunikacije s operacijskim sustavom.

3.3. Arhitektura projekta

Program se sastoji od dva velika dijela. To je korisničko sučelje s kojim se korisnik susreće i gdje može vidjeti podatke. Drugi je poslovne logike do koje korisnik nema direktan pristup. Tamo se nalazi kod za spajanje na ADCS sustav, kod za primanje, obradu i

pohranu podataka te kod za slanje naredbi ADCS sustavu. Ne želimo da korisnik mora pisati svoje programe kako bi napravio neku jednostavnu naredbu već mu tome služi korisničko sučelje.

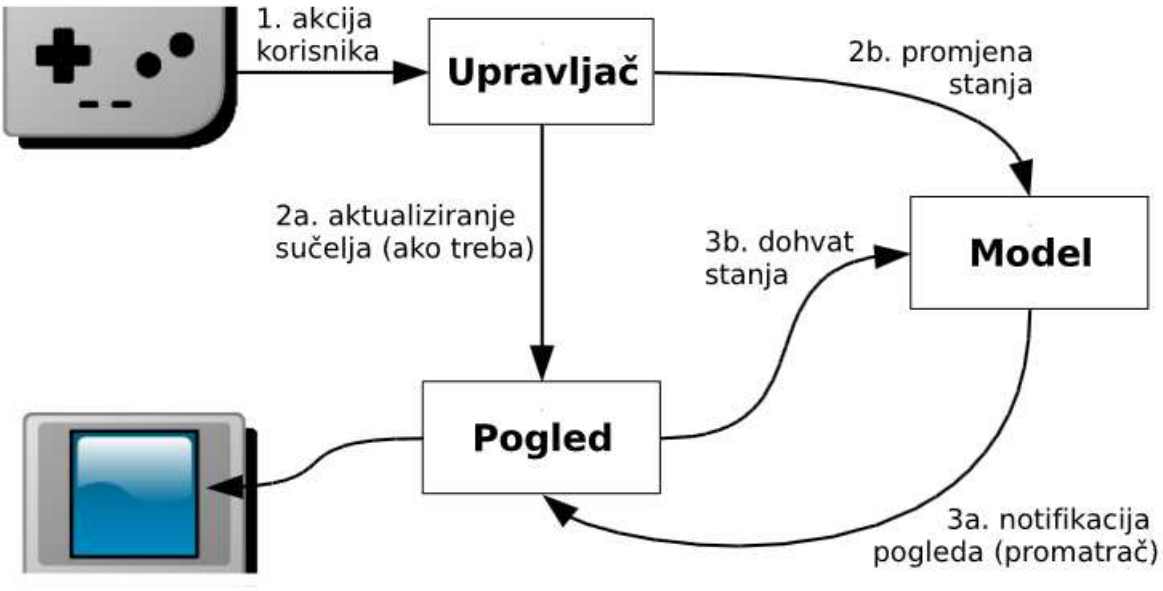
3.3.1. MVC obrazac

Model-Pogled-Upravljač (engl. *Model-View-Controller - MVC*) je arhitektonski oblikovni obrazac koji je korišten kako bi programski kod razdvojio u tri [7], u ovom slučaju dvije, nezavisne cjeline. Tri glavne cjeline čine model koji oblikuje podatke u našem programu, pogled ili pogledi koji su način na koji će ti podatci biti reprezentirani i ono s čime će se korisnik susretati te upravljač koji je oblikovati akcije koje se mogu dogoditi u našem programu. Pogled je prva stvar koju će korisnik vidjeti, taj pogled na neki način iscrtava podatke i potrebne elemente poput gumba za uređenje tih podataka. U trenutku kada korisnik klikne na neki element za mijenjanje podataka ta akcija je ili povezana na upravljač ili je to direktno upravljač koji će tada na temelju zadane akcije i trenutnih podataka ažurirati podatke i na kraju obavijestiti pogled da opet nacrta nove svježe podatke. Struktura slična navedenoj je vidljiva na slici 3.1. Moguće je izvesti razne inačice ovog obrasca pa tako u ovo rješenju imamo zapravo dvije komponente poglede i model-upravljač. Pogledi oblikuju grafičke elemente koje korisnik vidi a oni sadrže reference na ostale poslovne komponente koje imaju ulogu modela i kontrolera zajedno.

Ovo smo iskoristili kako bi odvojili kod za iscrtavanje grafičkog sučelja od koda poslovne logike. Konkretno, kod za spajanje na serijska vrata (engl. *port*) je sasvim odvojen od kod koji crta potrebne grafičke elemente za spajanje na vrata. Definirali smo sučelje koje će razumjeti i kod grafičkog sučelja i kod poslovne logike. Na taj način možemo transparentno zamijeniti oba djela i kod će raditi sve dok oba dijela znaju komunicirati zajedničkim sučeljem. Ovo nam je izuzetno korisno kada dodajem nove mogućnosti ili želimo neki dio koda zamijeniti drugim djelom koda za svrhe testiranja.

3.4. Struktura projekta

Kod programske podrške je podijeljen u nekoliko funkcijskih cjelina za lakšu organizaciju i preglednost. Pregled organizacije projekta kao i opisi direktorija je vidljiv na tablici 3.1. Datoteka `main.py` je glavna datoteka koja pokreće program i definira glavnu aplika-



Slika 3.1. Suradnja MVC komponenata

Naziv	Tip	Opis
config	Direktorij	Konfiguracijske datoteke koje služe za spremanje postavki
core	Direktorij	Funkcije i dijelovi koji nisu vezani uz grafičko sučelje
modules	Direktorij	Veće kompozitne grafičke komponente
stores	Direktorij	Datoteke koje pohranjuju globalne podatke
tabs	Direktorij	Grafičke komponente koje čine glavno sučelje
utils	Direktorij	Dodatne dijeljene funkcionalnosti
validators	Direktorij	Specijalni validatori točnosti upisanog broja
widgets	Direktorij	Posebno napravljene primitivne grafičke komponente
main.py	Datoteka	Glavna datoteka koja pokreče program

Tablica 3.1. Popis cjelina i datoteka koje čine program

ciju kao i strukturu stranica glavne aplikacije. U datotekama `widgets`, `tabs` i `modules` se nalaze grafičke komponente koje se dodaju u stranice. Stranica sadrži ili komponente ili skup komponenata nazvanim modulom. Podatci koji pristižu od ADCS sustava se pohranjuju u datotekama oblikovanim u `stores` direktoriju. Poslovna logika se nalazi u toj datoteci, ali većina upravljača i glavnih funkcionalnosti se nalazi u `core` direktoriju.

3.5. Prijenos podataka između grafičkog sučelja i ADCS-a

Za bežični prijenos podataka koristimo tehnologiju Bluetooth. ADCS sustav ima Bluetooth odašiljač s pomoću kojega može komunicirati s bilo kojim računalom koje ima Bluetooth čip, a to su danas sva računala.

3.5.1. Bluetooth

Bluetooth tehnologija odašilje radio valove na 2.4GHz u industrijsko, znanstveno, medicinskom frekvencijskom području (engl. *Industrial, scientific, and medical - ISM*) [8]. Originalno zamišljena tehnologija kao bežična zamjena za RS-232 podatkovne kablove dana se može koristiti za spajanje periferije, prijenos podataka, prijenosa audio podataka u stvarnom vremenu, spajanju IoT uređaja i još mnogo toga.

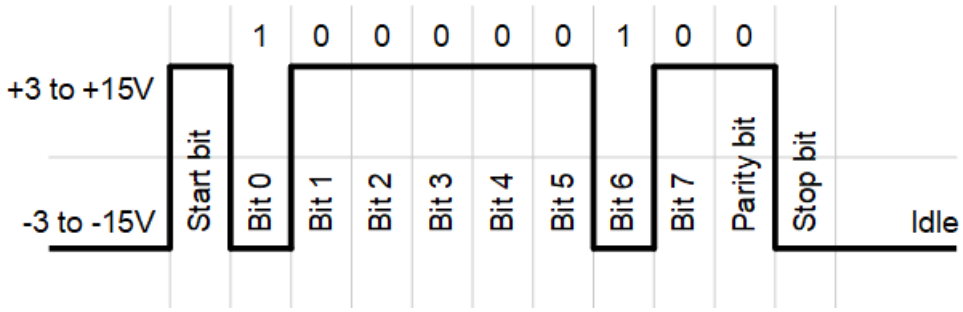
3.5.2. Serijski prijenos podataka

Protokol RS-232 je telekomunikacijski protokol za prijenos podataka između dvije točke. Podatci se šalju bit po bit sekvencionalno (serijski). Svaka poruka se sastoji od početnog bita, poruke, opcionalnog paritetnog bita i stop bita [9]. Start bit označava početak prijenosa i služi za sinkronizaciju prijamnika i pošiljatelja. Nakon toga slijedi neki broj bitova koji prenose podatke. Kako ne znamo koji je to broj, i pošiljatelj i primatelj moraju znati ovu informaciju prije nego što pokrenu bilo kakav prijenos podataka. Nakon bitova poruke dolazi opcionalni paritetni bit, ako je uključen ovaj bit služi za detekciju greške u prijenosu. Na kraju dolazi stop bit koji naponsku razinu vraća u normalno stanje. Naponske razine mogu biti u rasponu od -15 V do +15 V. Naponska razina kada linija nije korištena ili kada dođe stop bit se postavlja u niži dio raspona i signalizira logičku jedinicu dok gornji dio raspona signalizira logičku nulu kao što možemo vidjeti na slici 3.2. Iako je ovo protokol prvo namijenjen za žičanu uporabu, pojavom tehnologije Bluetooth možemo koristiti isti protokol za slanje i primanje podataka samo što naši podatci ne putuju kroz žicu već kroz zrak.

3.5.3. Brzina prijenosa podataka

S obzirom na prirodu zadatak i zahtjeva da obrada podataka i slanje korekcija mora biti ostvareno u pravom vremenu bez kašnjenja moramo biti u stanju primiti i obraditi podatke dovoljno brzo kako bi ispunili taj zahtjev. ADCS podatke sa senzora očitava svakih 50 milisekundi. Ova vrijednost se može mijenjati, ali za potrebe ovog rada nije bila promijenjena. RS232 definira brzinu kao broj bitova koji se može prenijeti u jednoj sekundi (engl. *Baud rate*).

Trenutno koristimo brzinu prijenosa od 115200 bitova po sekundi. Podatci koji dolaze



Slika 3.2. Logičke razine protokola RS232

```
counter 0000000
acc (-180.0000, -000.8100, -123.0000)
gyro (-180.0000, -180.0000, -123.0000)
angle (-180.0000, -180.0000, -180.0000)
temp 23.2222
```

Slika 3.3. Primjer formata mjernih podataka

s ADCS-a su predstavljeni u tekstualnom formatu. Primjer podataka koji se šalju od ADCS-a prema korisničkom sučelju su vidljivi na slici 3.3. Broj znakova u primjeru je 146, što je i više nego neki prosječni paket, ali ovdje je uzet nagori slučaj da vidimo hoće li brzina obrade i tada biti zadovoljavajuća. Ostale dogovorene opcije su broj bitova za prijenos jednog znaka je 8, ne koristimo paritetni bit i koristimo jedan stop bit. Za svaki znak to ukupno čini 10 bitova. Ako brzinu prijenosa podijelimo s brojem bitove za jedan znak dobiti ćemo koliko znakova možemo poslati u jednoj sekundi. U ovom slučaju to je:

$$\frac{115200 \text{ bitova/sekundi}}{10 \text{ bitova/znaku}} = 11520 \text{ znakova/sekundi}$$

Podijelimo li sada veličinu najdužeg paketa s brzinom prijenosa znakova, dakle:

$$\frac{146 \text{ znakova}}{11520 \text{ znakova/sekundi}} = 0.0126 \text{ sekundi}$$

Dobivamo vrijeme potrebno za prijenos jednog paketa podataka. To je 12.60 milisekundi. S obzirom na to da podatke sa senzora čitamo svakih 50 milisekundi, ovo bi nam trebalo biti dovoljno vremena za obradu podataka u programu i slanje povratnih upravljačkih naredbi.

3.6. Obrada podataka

Podatci koji dolaze preko serijske konekcije se čitaju redak po redak te se prosljeđuju parseru. Parser pretvara podatke iz tekstualnog oblika u brojeve, za grupira brojeve u cjeline ovisno o kojoj vrsti podataka se radi. Primanje i obrada podataka rade asinkrono. Nakon što se uspostavi serijska konekcija u kodu imamo objekt koji reprezentira tu konekciju. Kod koji se brine o toj konekciji je modeliran kao oblikovni obrazac jedinstveni objekt (engl. *Singleton*) jer ne možemo imati više serijskih konekcija u isto vrijeme. Tom objektu možemo pridružiti funkciju koja će biti pozvana kada stignu neki podatci. To se dešava asinkrono zato što će taj kod biti pozvan u nepoznatim vremenima nakon što stignu podatci. Nakon obrade podataka u parseru, parser obavještava slušatelje da ima novi podatak te im šalje taj podatak. Ovo je postignutu oblikovnim obrascem promatrač (engl. *Observer*) gdje je parser subjekt a zainteresirani dijelovi koda promatraju taj subjekt i čekaju nove podatke. Tako ako je više dijelova zainteresirano za nove podatke možemo lagano dodati novog slušatelja koji će čekati te podatke.

3.7. Pohrana podataka

Kako dobivamo mnogo podataka koje želimo prikazati na grafovima i koristiti za izračune moramo ih negdje pohraniti. Trenutno rješenje je jedan jedinstveni objekt koji modelira naše dijeljeno stanje aplikacije i u njemu se nalaze podatci. Jedinstveni objekt je odabran jer želimo tim podacima pristupiti iz različitih dijelova korisničkog sučelja. Elementi korisničkog sučelja su modelirani kao hijerarhijska struktura stablo i želimo izbjeći da neku referencu moramo prosljeđivati s roditelja na dijete nekoliko generacije. Uz to, za potrebe aplikacije podatke nije potrebno pohranjivati na više mjesta. To bi uzrokovalo veću potrošnju memorije i potencijalno stanje gdje nemamo sinkronizirane podatke u dva različita spremnika.

3.7.1. Obrazac jedinstveni objekt

Obrazac jedinstveni objekt (engl. *Singleton*) je vrlo koristan oblikovni obrazac koji modelira nešto što ne želimo instancirati više puta. U kontekstu ovog programa to su dijeljeno stanje podataka i konekcija sa serijskim vratima. Nema potrebe, a ni smisla imati dva različita objekta koja se brinu o serijskim konekcijama, a niti želimo omogu-

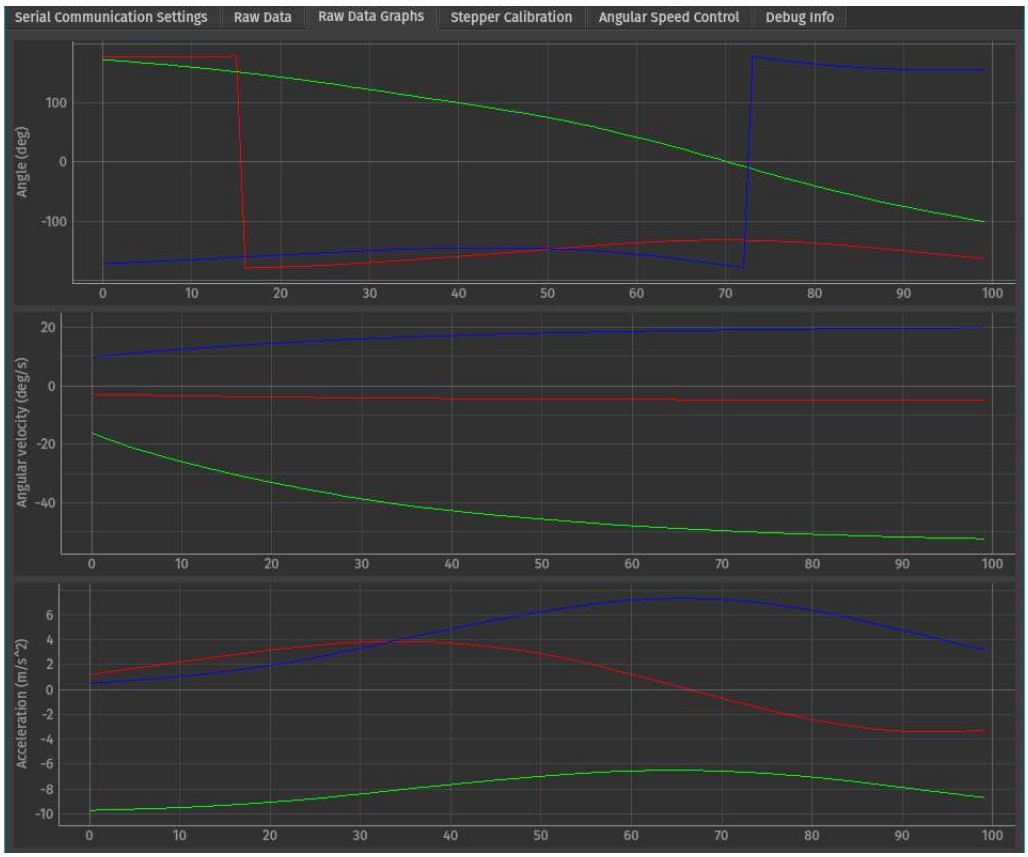
ćiti da imamo dvije konekcije istovremeno. Objekt koji je proglašen jedinstvenim nema javni konstruktor već ima metodu koja dohvaća već stvorenu instancu. Ako instanca tog objekta još ne postoji tada se zove privatni konstruktor koji će napraviti jednu instancu tog objekta. Sad će korisnik može više puta zvati metodu za dohvat objekta obično nazvanu `getInstance()` i dobiti će referencu na isti objekt.

3.8. Prikaz podataka

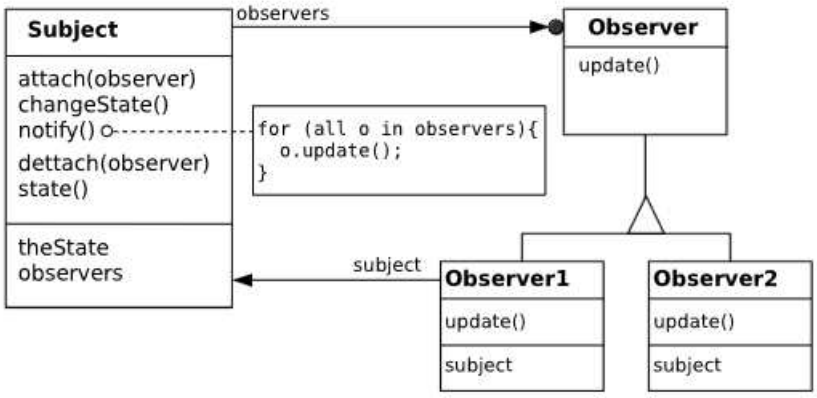
Podatci su prikazani na dva načina, sirovi podatci, dakle tekst koji je pročitan sa serijske veze se ispisuje u jednu tekstualnu list kako bi korisnik mogao odmah vidjeti koji podatci mu dolaze s ADCS-a. Drugi način su grafovi koji prikazuju podatke kroz neki vremenski period, trenutno zadnjih 5 sekundi kao što je vidljivo na slici 3.4. U trenutku izrade nekog grafičkog elementa pomoću biblioteke PyQt6 možemo specificirati početnu vrijednost te komponente. No tu dolazimo do jednog problema. Što ako se ta vrijednost promijeniti, hoće li se i naša komponenta automatski promijeniti kako bi iscrtala novu vrijednost? Odgovor je da neće, ako nije dorađena. Naime, komponente u PyQt6 nisu reaktivne i ne mijenjaju se ako se promjene podatci na koje se referenciraju. Za rješavanje ovog problema nam je pogodan oblikovni obrazac promatrač. Promatrač se sastoji od dva glavna dijela. Subjekta i samog promatrača. U ovom slučaju će neki podatak, recimo trenutna brzina DC motora, biti proglašena subjektom. Grafički element, recimo neki tekstualni element ćemo proglasiti promatračem. Naš subjekt, podatak brzine, će pamtiti tko ga sve sluša i u trenutku kada se on promijeni će javiti svim promatračima da se podatak promijenio.

3.8.1. Obrazac promatrači

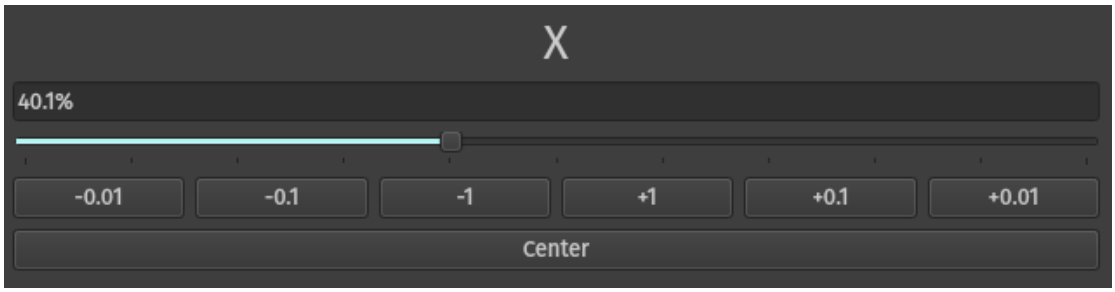
Oblikovni obraza promatrač je vrlo koristan alat kada jedan dio našeg programa mora promatrati drugi dio i reagirati na promijene. Strukturni dijagrama ovog oblikovnog obrasca je vidljiv na slici 3.5. Obrazac se sastoji od tri glavna djela, to su subjekt (engl. *subject*), apstraktni promatrač (engl. *abstract observer*) i konkretni promatrač ili samo promatrač (engl. *observer*). U kontekstu grafičkog korisničkog sučelja promatrači se koriste za održavanje ažuriranosti sučelja s podacima iz modela. Sada možemo razmotriti tipičnu primjenu promatrača u našem programu. Imamo dio sučelja kao što je prikazano



Slika 3.4. Primjer prikaza podatak na grafovima



Slika 3.5. Strukturni dijagram promatrača



Slika 3.6. Primjer GUI-a koji koristi promatrač

na slici 3.6. Vidljiv je element za upis brojeva u kojem se prikazuje neki broj, u ovom slučaju postotak. Taj postotak ovdje označuje koliko će se jedan od koračnih motora pomaknuti od lijeve strane. Ispod toga imamo pomični izbornik, a još ispod toga se nalaze gumbi s kojima možemo mijenjati vrijednost našeg podatka. Upisivanjem broja želimo da nam se promijeni klizni izbornik, pomicanjem kliznog izbornika ili pritiskom na jednog od ponuđenih gumbi želimo da nam se promijeni iznos broja. To smo ostvarili tako da smo naš podatak proglasili subjektom i omotali ga u objekt naziva `ObservableValue`. Nakon kreacije našeg subjekta, tijekom kreiranja korisničkog sučelja možemo na subjekt pretplatiti naše promatrače što možemo vidjeti u nastavku:

```
    self.stepper_value_slider = QSlider(Qt.Orientation.Horizontal)
    ...
    self.stepper_value.add_callback(
        lambda value: self.stepper_value_slider.setValue(
            int(value * stepper_value_slider_scalar)
        )
    )
```

Prvo ćemo napraviti komponentu koja se zove `QSlider` i modelira klizni izbornik, nakon toga ćemo ga još konfigurirati što nam ovdje nije bitno. `stepper_value` je naš subjekt koji omotava vrijednost i na njega pretplaćujemo promatrača. U ovom slučaju to nije zaseban objekt, već je to lambda izraz koji je ekvivalentan objektu promatrača, ali ne moramo pisati svu sintaksu potrebnu za izradu objekta.

3.8.2. Slanje upravljačkih signala satelitu

Korisnik iz korisničkog sučelja može pritisnuti neki gumb kako bi poslao naredbu ADCS sustavu. Jedan takav gumb je gumb "Center" na slici 3.6. Pritiskom na taj gumb ADCS

Naredba	Opis
imu_start_reading()	Pokreni čitanje s IMU jedinice
imu_stop_reading()	Zaustavi čitanje s IMU jedinice
set_stepper(x, n)	Postavi pomak koračnog motora x na vrijednost n%
set_motor_speed(x, n)	Postavi brzinu DC motora broj x na n% maksimalne brzine
stop_motor(x)	Zaustavi DC motor broja x

Tablica 3.2. Popis trenutno podržanih naredbi

sustavu se šalje naredba da taj koračni motor centrira, tj. postavi ga na 50% od krajnje lijeve strane. Slanje naredbi je oblikovani kao zaseban razred čija je svrha omogućiti lagano pisanje koda i pozivanje jednostavnih naredbi, koje se kasnije prevedu u nešto što ADCS sustav razumije, te se na kraju i šalju sustavu putem serijske veze. Motivacija iza ovoga je da možemo korisniku izložiti jednostavno programsko sučelje, a naš razred provodi sve dodatne zadatke koji su potrebni da se ta naredba pošalje. Popis trenutno podržanih naredbi koje GUI nudi korisniku za upravljanje ADCS sustavom su navedene u tablici 3.2. Za izvedbu nekih od navedenih naredbi potrebni su dodatni parametri koji se mogu podesiti kroz postavke te se korisnik neće morati svaki put opterećivati s njima.

4. Verifikacija funkcionalnosti sustava

4.1. Korištene tehnologije

4.1.1. Uvod

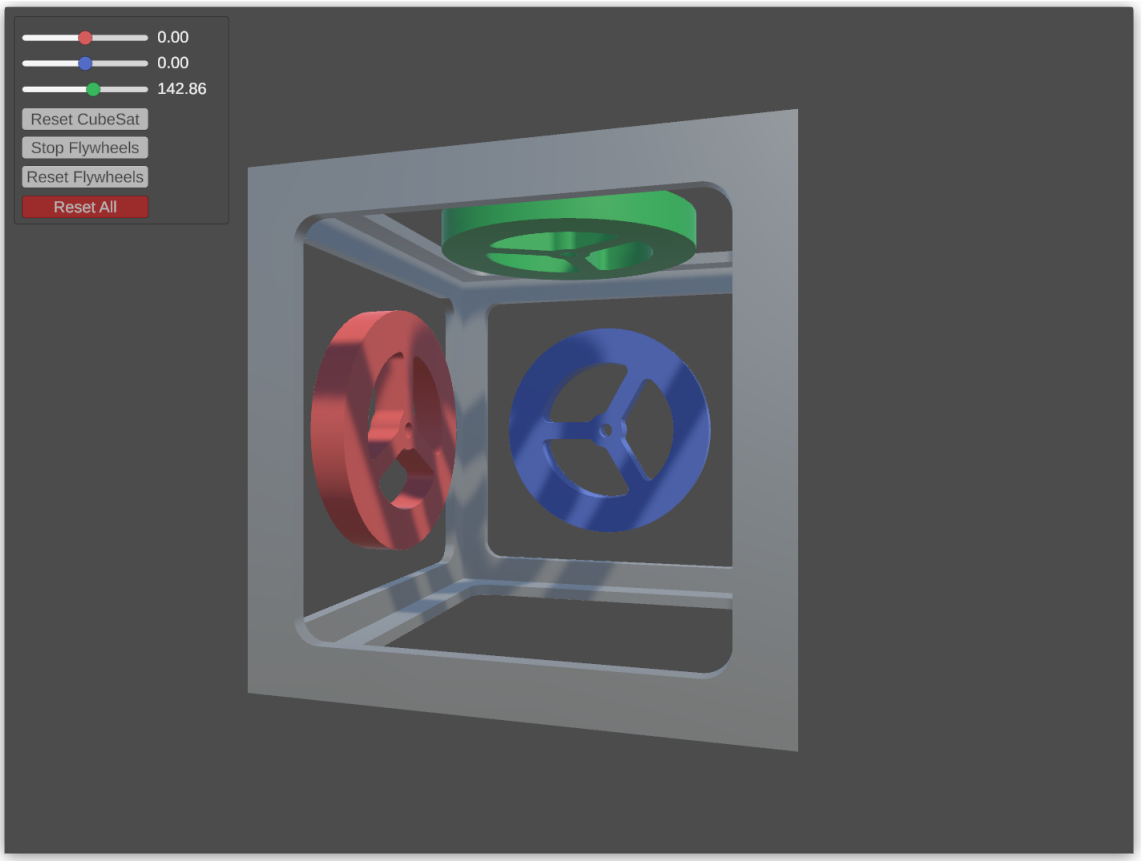
Za vrijeme razvoja korisničkog sučelja razvijeni su dodatni alati za lakšu provjeru rada programa. Jedan alat je simulator napisan u programskom jeziku C koristeći program za izradu video igara Unity. Drugi alat je fizički prototip na razvojnoj pločici koji se sastoji od Bluetooth modula za komunikaciju, Arduino Nano mikrokontrolera i inercijalne mjerne jedinice za sakupljanje podataka.

4.1.2. Unity

Unity je set alata za izradu i razvoj video igara. Ovdje je korišten za crtanje 3D objekata koji simuliraju naš ADCS sustav. Originalna ideja je bila i koristiti ugrađenu simulaciju fizike za simuliranje rotacija i inercije, ali zbog određenih limitacija to nije bilo moguće pa je simulacija implementirana ručno. Nije realna pravom svijetu, ali je dovoljno dobra za simuliranje osnovnih funkcija i verifikaciju ispravnosti. Izgled simulatora je vidljiv na slici 4.1.

4.1.3. Arduino

Arduino je skup razvojnih pločica koji omogućuju lagano spajanje s ostalim komponentama i razvoj ugradbene programske podrške za te komponente. Sama Arduino pločica, u ovom slučaju Arduino Nano, se spajaju na razvojnu pločicu s ostalim komponentama. Za spajanje se koriste tanke žice i nije potrebno lemiti što ubrzava razvoj i olakšava promjene. Naravno, ovi benefiti mogu biti i problemi pa u praksi ovo koristimo za izradu



Slika 4.1. Izgled Unity simulacije

prototipa nakon čega se možemo prebaciti na izradu tiskane pločice gdje su sve naše komponente ugrađene.

Testni uređaj je svakih 50 milisekundi očitavao mjerenja s IMU jedinice i obrađivao podatke. Važno je napomenuti da su mjereni podatci temperatura, kutna akceleracija i gravitacijska akceleracija. IMU koji je bio korišten za izradu prototipa nije podržavao mjerenje magnetometra. Također, ti podatci nisu bili direktno slani prema korisničkom sučelju već se koristila C++ biblioteka koja je na temelju kutne i gravitacijske akceleracije izračunala trenutni kut i kutnu brzinu našeg uređaja. Ti podatci uz samu gravitacijsku akceleraciju i temperaturu se šalju prema korisničkom sučelju. Razlog ovog odabira je taj da promjenu u kutu i kutnoj brzini možemo lakše vizualizirati i time lakše pratiti ponaša li se naše sveukupni sustav ispravno.

4.2. Simulirana verifikacija

Simulacija u tehnologiji Unity je olakšala razvoj i pomogla otkloniti mnoge probleme. Uz to s pomoću nje je uspješno testirana komunikacija od strane simuliranog ADCS sustava prema korisničkom sučelju. Ovo simulaciji možemo vjerovati zato što koristi isto sučelje za prijenos podataka korisničkom sučelju. U ovom slučaju to je komunikacija putem serijskih vrata. S obzirom na to da i simulator i fizički sklop koriste isti format možemo ih bilo kada zamijeniti.

4.3. Verifikacija na hardveru

Testiranje na fizičkom prototipu je u ovom slučaju bolje jer je sličnije pravom hardveru na ADCS sustavu. Bluetooth modul za komunikaciju je sličniji onom pravom kao i IMU mjerna jedinica. Također fizički prototip sadrži šum koji rezultira nesavršenim signalom za razliku od simulirane verzije koja daje savršen signal. Fizičkim pomicanjem prototipa možemo vidjeti da se podatci prenose do korisničkog sučelja, tamo se točno obrađuju i prikazuju na grafovima. Time je potvrđena komunikacija od strane prototipa prema korisničkom sučelju. Drugi smjer komunikacije smo potvrdili slanjem naredbi `imu_start_reading()` i `imu_stop_reading()` koje omogućuju ili onemogućuju čitanje i slanje podataka s prototipa.

5. Zaključak

U ovom završnom radu opisani su postupci i tehnologije korištene za izradu grafičkog korisničkog sučelja za primanje, obradu i slanje signala s udaljenog ADCS sustava. Opisan su svi dijelovi grafičkog korisničkog sučelja kao i algoritmi poslovne logike. Spomenuti su oblikovni obrasci koji su bili upotrebljeni kako bi oblikovali skalabilno i modularno rješenje. Opisani i riješeni su izazovi izrade grafičkog sučelja i održavanja njegovog ažuriranosti s podatcima koji je dinamično mijenjaju. Dokumentirani su i popratni alati koji su razvijeni za svrhe verifikacije ispravnosti rada programa. Sljedeći koraci su implementirati mogućnost podrške izvršavanja korisničkih kodova. Tako se grafičko korisničko sučelje može koristit kao razvojni okvir za izradu i testiranje upravljačkih algoritama ADCS sustava oko sve tri osi.

Literatura

[1] What are SmallSats and CubeSats? - NASA. Section: Ames Research Center. [Mrežno]. Adresa: https://www.nasa.gov/what-are-smallsats-and-cubesats/

[2] P. D. P. Kogut. Small satellites: Types, uses, and role in the space industry. [Mrežno]. Adresa: https://eos.com/blog/small-satellites/

[3] CubeSat 101: Basic concepts and processes for first-time CubeSat developers. [Mrežno]. Adresa: https://www.nasa.gov/wp-content/uploads/2017/03/nasa_csli_cubesat_101_508.pdf

[4] Dubravko Babić, Ana Babić, Josip Lončar, i Josip Vuković. Uvod u svemirske tehnologije predavanja. [Mrežno]. Adresa: https://www.fer.unizg.hr/predmet/uust/predavanja

[5] R. C. Limited, "PyQt6: Python bindings for the qt cross platform application toolkit".

[6] M. Fitzpatrick, *Create GUI applications with Python & Qt6*. Martin Fitzpatrick, 2023.

[7] Marko Čupić i Siniša Šegvić. Oblikovni obrasci u programiranju materijali. [Mrežno]. Adresa: https://www.fer.unizg.hr/predmet/ooup/materijali

[8] Bluetooth technology overview. [Mrežno]. Adresa: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

[9] An introduction to RS232 serial port communication. [Mrežno]. Adresa: https://www.anticyclone-systems.co.uk/rs232.php

Sažetak

Grafičko korisničko sučelje za upravljanje, prikupljanje i prikaz signala iz ADCS prototipa

Neven Lukić

Upravljanje i određivanje orijentacije satelita je jedan od ključnih zadataka bilo kojeg satelita. Za potrebe izrade i proučavanja vlastitog sustava određivanja i upravljanjem orijentacijom (ADCS) potrebno je imati dobru platformu kako bi se brzo moglo iterirati kroz razne verzije i isto tako kvalitetno verificirati njihova ispravnost. U ovom radu, prezentirana je programska podrška u obliku grafičkog korisničkog sučelja koja studentima i istraživačima omogućuje sakupljanje, obradu i vizualizaciju podataka, pisanje kontrolnih algoritama i slanje kontrolnih signala ADCS sustavu.

Ključne riječi: Upravljanje orijentacijom satelita, ADCS, CubeSat, Grafičko korisničko sučelje

Abstract

Graphical user interface for controlling, collecting and visualizing data from an ADCS prototype

Neven Lukić

Orientation determination and control are the core tasks of every satellite. For the purposes of creating and researching custom orientation determination and control systems (ADCS), it is beneficial to have a strong platform on which iterations can be made swiftly and at the same time thoroughly verified. In this thesis, the custom support software is presented in the form of a graphical user interface. It serves as a framework for researchers and students to help them with data collection, processing and visualization, writing and running custom control algorithms, and sending control commands to the ADCS system.

Keywords: Orientation control, ADCS, CubeSat, Graphical user interface