Digitalizacija zrakoplovnog instrumenta za ugradnju u simulator za obuku pilota jedrilice

Delija, Viktoria

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:798851

Rights / Prava: Attribution 4.0 International/Imenovanje 4.0 međunarodna

Download date / Datum preuzimanja: **2024-05-26**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Viktoria Delija

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Prof. dr. sc. Mladen Crneković, dipl. ing.

Student:

Viktoria Delija

Zagreb, 2023.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru prof.dr.sc. Mladenu Crnekoviću na pomoći i savjetima tijekom pisanja završnog rada te asistentu Marinu Lukasu mag. ing. na ideji za temu za rad kao i za smjernice, savjete i pomoć u izradi završnog rada.

Viktoria Delija

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 23 – 6 / 1**	
Ur.broj: **15 - 1703 - 23 -**	

ZAVRŠNI ZADATAK

Student: **Viktorija Delija** JMBAG: **0036503244**

Naslov rada na hrvatskom jeziku: **Digitalizacija zrakoplovnog instrumenta za ugradnju u simulator za obuku pilota jedrilice**

Naslov rada na engleskom jeziku: **Digitalization of aircraft instument for aplication in glider pilot training simulator**

Opis zadatka:

Dio obuke pilota aviona provodi se na simulatorima leta kako bi se uštedjeli resursi zrakoplova te pilotima omogućilo školovanje na području izvanrednih postupaka. Školovanje pilota jedrilica na simulatorima pojavilo se tek nedavno te je aktualan predmet razvoja.

Kako bi simulator letenja zadovoljio potrebe školovanja, unutrašnjost simulatora mora biti identična unutrašnjosti zrakoplova. Jedan od izazova kod konstrukcije simulatora je i konstrukcija potpuno vjernih funkcionalnih replika zrakoplovnih instrumenata. Moguće rješenje tog problema je digitalizacija postojećih zrakoplovnih instrumenata.

U radu je potrebno:

- osmisliti i projektirati elektronički sklop sa mikrokontrolerom za komunikaciju između računala i upravljačkog sklopa koračnog motora,
- projektirati tiskanu pločicu za osmišljeni elektronički sklop,
- konstruirati kućište za elektronički sklop koje je prilagođeno za ugradnju u zrakoplovni instrument,
- ugraditi koračni motor za pomicanje kazaljke na zrakoplovnom instumentu,
- uspostaviti komunikaciju između mikrokontrolera i programa za simulaciju letenja Condor2 putem serijske veze te NMEA protokola te upravljati gibanjem koračnog motora za pomicanje kazaljke.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
30. 11. 2022.

Zadatak zadao:
Prof. dr. sc. Mladen Crneković

Datum predaje rada:
1. rok: 20. 2. 2023.
2. rok (izvanredni): 10. 7. 2023.
3. rok: 18. 9. 2023.

Predviđeni datumi obrane:
1. rok: 27. 2. – 3. 3. 2023.
2. rok (izvanredni): 14. 7. 2023.
3. rok: 25. 9. – 29. 9. 2023.

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA III

POPIS OZNAKA IV

POPIS TABLICA IV

SAŽETAK V

SUMMARY VI

1. UVOD 1

1.1. Ideja 1

2. ELEKTRONIČKI SKLOP 2

2.1. Mikrokontroler 2

2.2. Ostale komponente 3

2.2.1. Koračni motor 3

2.2.1.1. Načelo rada koračnog motora 4

2.2.2. Servomotor 5

2.2.2.1. Načelo rada servomotora 6

3. KONSTRUKCIJA 7

3.1. Brzinomjer 7

3.2. Variometar 7

3.3. Visinomjer 8

4. PRORAČUN SERVOMOTORA 10

5. ELEKTRIČNA SHEMA 17

6. PROGRAM ZA SIMULACIJU LETENJA 18

7. ARDUINO KOD 21

8. ZAKLJUČAK 24

LITERATURA 25

PRILOZI 26

PRILOG I 27

PRILOG II 29

POPIS SLIKA

Slika 1. Mikroupravljačka pločica 2
Slika 2. Raspored ulazno/izlaznih pinova 3
Slika 3. Koračni motor x27.168 4
Slika 4. Bipolarni koračni motor 4
Slika 5. Raspored pinova na koračnom motoru x27.168 5
Slika 6. Servomotor FS 155 BB MG 5
Slika 7. Dijelovi servomotora 6
Slika 8. Brzinomjer 7
Slika 9. Variometar 7
Slika 10. Konstrukcija poklopca variometra 8
Slika 11. Visinomjer 9
Slika 12. Konstrukcija visinomjera 9
Slika 13. Komparator 10
Slika 14. Mjerenje ponovljivosti s komparatorom 11
Slika 15. Zračnost unutar zupčanika 12
Slika 16. Kut zakreta servomotora 12
Slika 17. Kut zakreta visinomjera 13
Slika 18. Zglobni četverokut 14
Slika 19. Proračun zglobnog četverokuta 14
Slika 20. *MATLAB* program za provjeru duljina zglobnog četverokuta 16
Slika 21. Električna shema 17
Slika 22. *Condor Soaring* simulator 18
Slika 23. UDP postavke 19
Slika 24. Uspostava čitanja i slanja podataka 19
Slika 25. Dobiveni podaci iz simulatora 19
Slika 26. Dobivanje vrijednosti iz niza podataka 20

POPIS OZNAKA

α_1 - kut motora

L_1 - duljina kraka motora

x_1 - pomak motora

α_2 - kut kazaljke

L_2 - duljina kraka kazaljke

x_2 - pomak kazaljke

θ_1 - ulazni kut zglobnog četverokuta

θ_4 - izlazni kut zglobnog četverokuta

r_1, r_2, r_3, r_4 - duljine krakova zglobnog četverokuta

POPIS TABLICA

Tablica 1. Specifikacije SBC-NodeMCU-ESP32 razvojne pločice 2
Tablica 2. Specifikacije FS 155 BB MG servomotora 5
Tablica 3. Odstupanje od zadanog položaja servomotora 11
Tablica 4. Mjerenje kuta zakreta kazaljke visinomjera 13

SAŽETAK

Ovaj rad se može podijeliti u tri dijela: prvi je prilagodba samih instrumenata kako bi se mogli ugraditi motori za upravljanje kazaljkom te konstruiranje i izrada svih potrebnih dijelova za ispravan rad.

Drugi dio je odabir upravljačke pločice i svih potrebnih dijelova koji su prilagođeni za svaki instrument te izrada ispravne električke sheme koja ima sve potrebne komponente.

Treći dio je prijenos informacija iz simulatora na upravljaču pločicu putem Bluetooth komunikacije i sam način slanja i obrade potrebnih podataka za prikazivanje ispravnih vrijednosti na instrumentima, kao i programiranje upravljačke pločice.

U radu su korišteni programi *Altium Designer* i *SolidWorks*, kao i programsko sučelje *Arduino*.

Ključne riječi: mikrokontroler, NMEA, simulator letenja, zglobni četverokut

SUMMARY

This thesis can be divided into three parts: the first part involves adapting the instruments themselves to accommodate the installation of motors for controlling the gauge and designing all the necessary components for proper operation.

The second part is the selection of the microcontroller board and all the necessary parts for each instrument, as well as the creation of a correct electrical scheme that includes all the required components.

The third part is the transfer of information from the simulator to the microcontroller board via Bluetooth communication and the method of sending and processing the necessary data to display correct values on the instruments, as well as programming the microcontroller board.

The programs used in this work include *Altium Designer* and *SolidWorks*, as well as the *Arduino* software interface.

Keywords: microcontroller, NMEA, flight simulator, four bar linkage

1. UVOD

U svrhu školovanja pilota jedrilica dio obuke provodi se na simulatorima leta. Simulatori leta su koristan alat u obuci pilota jer omogućuju kontroliran okoliš u kojem piloti mogu usavršiti svoje vještine bez korištenja resursa stvarne jedrilice kako bi se eliminirao rizik te smanjila cijena sveukupne obuke. Omogućuje budućim pilotima upoznavanje sa procedurom leta od početka do kraja, instrumentima koji se nalaze u unutrašnjosti zrakoplova, upoznaje ih s načinom navigiranja jedrilice te s još mnoštvo aspekata leta koji su potrebni za upravljanje pravom jedrilicom. Zato je od velike važnosti da letenje pomoću simulatora bude što vjernije unutrašnjosti zrakoplova, kako bi piloti imali što realniju sliku upravljanja jedrilicom, da bi doživjeli okoliš leta te učili na greškama na sigurnan način. Jedan od načina izvedbe simulatora je digitalizacija postojećih zrakoplovnih instrumenata, brzinomjera, visinomjera i variometra.

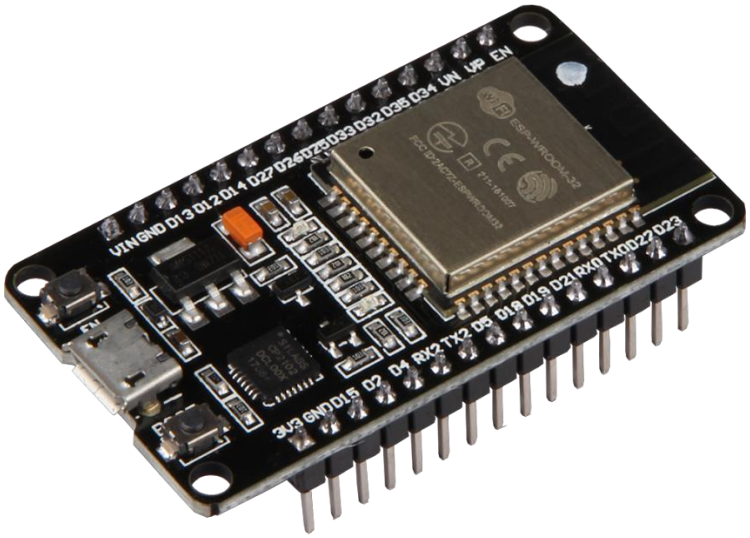
1.1. Ideja

Za konstrukciju ispravnog simulatora leta, potreban je mikrokontroler koji će upravljati s tri motora koji kontroliraju kazaljke na zrakoplovnim instrumentima. U ovom radu koristi se razvojna pločica na koju su spojena dva koračna motora te servomotor. Kako bi motori mogli upravljati kazaljkama potrebno je konstruirati kućišta za svaki zrakoplovni instrument te ugraditi motore u instrumente. Podaci za upravljanje instrumentima dobivaju se iz programa za simulaciju leta *Condor 2*, te putem Bluetooth veze ti podaci se šalju na motore.

2. ELEKTRONIČKI SKLOP

2.1. Mikrokontroler

Mikroupravljačka pločica korištena u radu je SBC-NodeMCU-ESP32 proizvođača Joy-it. Glavna karakteristika pločice je mogućnost povezivanja preko Wi-Fi mreže te ugrađeni Bluetooth modul, što je glavni razlog odabira pločice, pošto se komunikacija programa za simulaciju leta i mikrokontrolera odvija preko Bluetooth veze. [1]

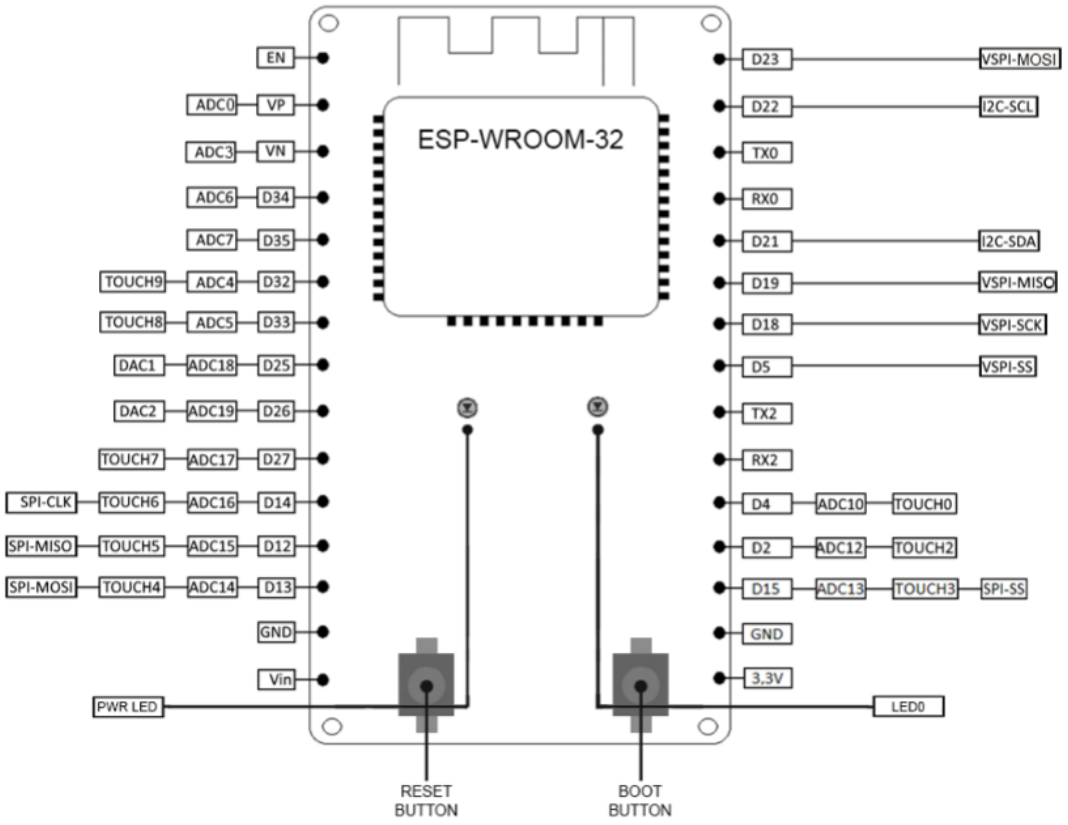


Slika 1. Mikroupravljačka pločica

Najvažnije tehničke specifikacije mikroupravljačke pločice su 4 MB memorije, 512 kB SRAM memorije, te radi na frekvenciji 2,4 GHz. Pločica ima 21 ulazno/izlaznih pinova te podržava I2C, SPI i UART komunikaciju. [1]

Tablica 1. Specifikacije SBC-NodeMCU-ESP32 razvojne pločice

Mikrokontroler	NodeMCU ESP32
SRAM	512 kB
Frekvencija	2,4 GHz
Bluetooth povezivanje	Klasično/LE
Komunikacija	UART/I2C/SPI/DAC/ADC
Radni napon	3,3 V
Temperatura rada	-40°C – 125°C
Dimenzije	48x26x11,5 mm
Težina	10 g



Slika 2. Raspored ulazno/izlaznih pinova

2.2. Ostale komponente

2.2.1. Koračni motor

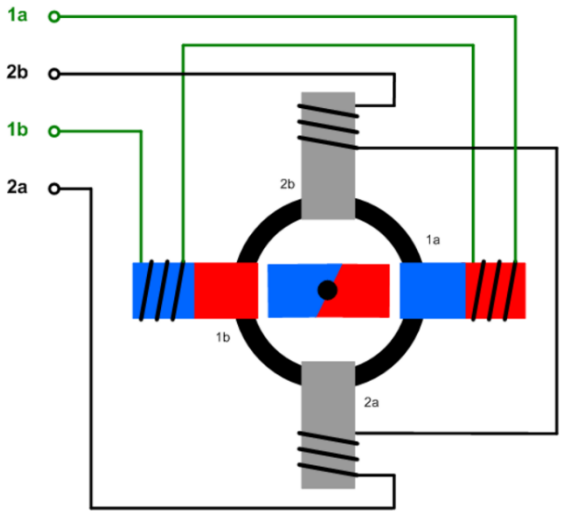
Koračni motor koji se koristi za brzinomjer i variometar simulatora leta je x27.168 proizvođača Adafruit, koji se često koristi za upravljanje brzinomjera na automobilima. Visoke je preciznosti sa kutom zakreta od 315 stupnjeva te se svakim korakom zaokrene za 1/2 stupnja. [3]

Motor je odabran radi lakog upravljanja te visoke preciznosti pozicioniranja koja je potrebna za mjerenje brzine te kako skale korištenih instrumenata također imaju zakret od 315 stupnjeva, lako je odrediti zakret koračnog motora da odgovara pokazivaču brzine. Povlači struju od 15 do 20 mA te pošto struja iz svakog I/O pina mikroupravljačke pločice iznosi 40 mA, koračni motor se može spojiti na pločicu bez potrebe korištenja drivera.

Slika 3. Koračni motor x27.168

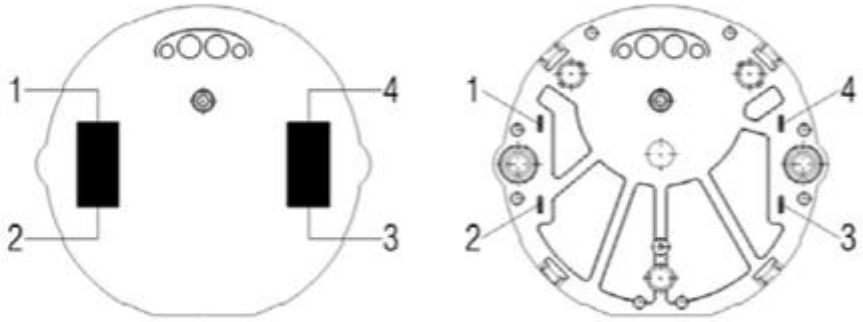
2.2.1.1. Načelo rada koračnog motora

Koračni motor se okreće tako da, kada struja teče kroz namotaje statora, stvara se magnetsko polje oko tih namota. Odabrani motor je bipolarni motor, što znači da ima dva polariteta te se rotacija magnetskog polja postiže promjenom polariteta napona na krajevima namotaja. Rotor se sastoji od trajnih magneta s sjevernim i južnim polovima, dok stator ima četiri elektromagnetska namota (1A, 1B, 2A, 2B), koji su upareni, pri čemu svaki par čini jednu fazu motora. Kada struja teče kroz elektromagnetske namote, stvaraju se magnetska polja.



Slika 4. Bipolarni koračni motor

Kod koračnog motora x27.168 pinovi 1 i 2 čine jednu zavojnicu, a pinovi 3 i 4 drugu zavojnicu.



Slika 5. Raspored pinova na koračnom motoru x27.168

2.2.2. Servomotor

Za pomicanje kazaljke visinomjera korišten je servomotor FS 155 BB MG marke Robbe. Konstruiran je za veliko opterećenje i izdržljivost. Odabran je radi velike preciznosti koja je potrebna za upravljanje kazaljkom. Preporučeni ulazni napon za upravljanje iznost 4.8 do 6V. [10]

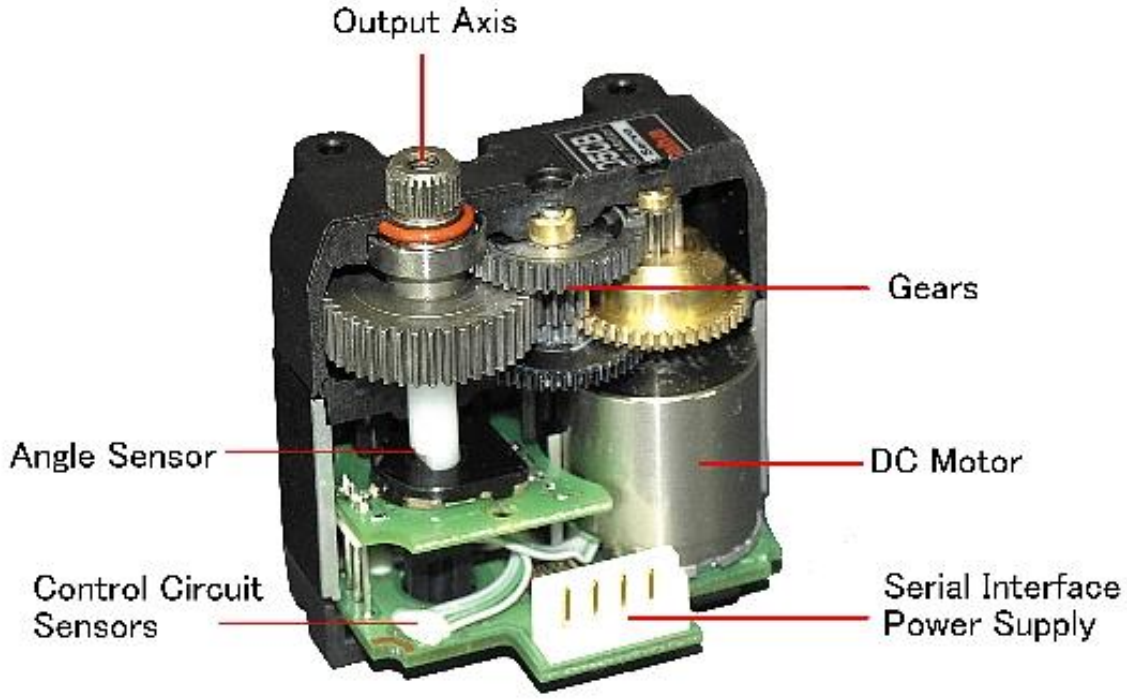
Tablica 2. Specifikacije FS 155 BB MG servomotora

Kutna brzina pri 6 V	12,5 s^{-1}
Okretni moment pri 6 V	1,9 kg/cm
Dimenzije	22,5x11,5x22,5 mm
Težina	12 g

Slika 6. Servomotor FS 155 BB MG

2.2.2.1. Načelo rada servomotora

Servomotor se sastoji četiri glavne komponente: DC motor, senzor pozicije, reduktor i kontrolni krug. Servomotor funkcionira po načelu povratne veze da bi dobili željenu poziciju osovine motora. Kao ulaz dajemo željenu poziciju motora, a izlaz je trenutna pozicija motora. Usporedbom nastaje greška koja pokreće osovinu motora. Osovina motora povezana je sa potenciometrom. Kako se motor pokreće, kut zakreta potenciometra mijenja se proporcionalno s napajanjem, a taj napon uspoređuje se s ulaznim naponom. Postupak se ponavlja sve dok se greška ne smanji na minimum, odnosno dok se servomotor ne pozicionira na željenom mjestu.



Slika 7. Dijelovi servomotora

3. KONSTRUKCIJA

3.1. Brzinomjer

Brzinomjer već ima svoje kućište, koje se rastavlja s prednje strane, čime će ugradnja koračnog motora biti znatno pojednostavljena, tako što će se osovina motora direktno povezati s kazaljkom. Skala brzinomjera nije jednoliko raspoređena, s užim rasponom od 0 do 100 km/h za razliku od ostatka skale, zbog čega će kretanje motora biti prilagođeno da bi brzinomjer pokazivao ispravnu brzinu.

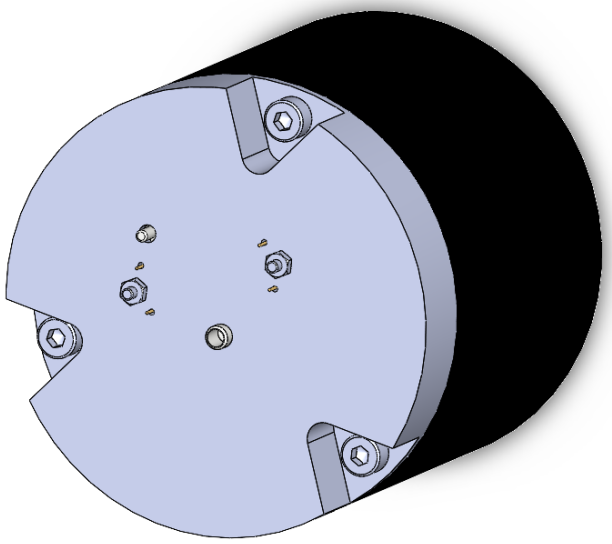
Slika 8. Brzinomjer

3.2. Variometar

Variometar ima ravnomjernu raspoređenu skalu s užim rasponom što se također uzima u obzir kod kontroliranja pomaka koračnog motora.

Slika 9. Variometar

Za variometar je konstruiran poklopac koji se vijcima pričvrsti na kućište instrumenta. Na poklopac je pričvršćen koračni motor koji se spaja na postojeću osovinu koja kontrolira kazaljku. Konstrukcija instrumenata napravljena je pomoću programa *Solidworks*.



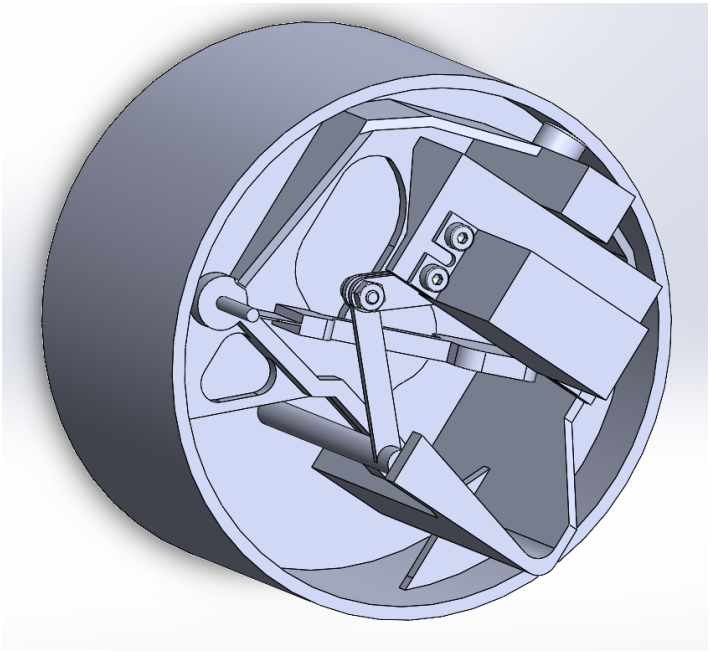
Slika 10. Konstrukcija poklopca variometra

3.3.Visinomjer

Pokretanje kazaljke visinomjera realizirano je korištenjem zupčanika, zbog čega minimalno zakretanje osovine koja utječe na pokrete kazaljke rezultira velikim zakretom na samoj kazaljki. Samim time, pokretanje kazaljke pomoću koračnog motora ne bi bilo moguće izvesti tako da se ostvare glatke kretnje, pošto čak i jednim korakom motora kazaljka izvrši veliki pokret, što bi rezultiralo nepredvidivim trzajima i netočnim pokazivanjima kazaljke. Iz tog razloga servomotor se u instrument ugrađuje skupa sa zglobnim četverokutom, kako bi se svakim korakom servomotora kazaljka stabilnije kretala.

Slika 11. Visinomjer

Uz zglobni četverokut, potrebno je konstruirati i postolje servomotora koje se vijcima pričvršćuje na instrument.



Slika 12. Konstrukcija visinomjera

4. PRORAČUN SERVOMOTORA

Za potrebe konstruiranja zglobnog četverokuta, izvodi se mjerenje zračnosti, rezolucije i ponovljivosti servomotora kako bi odredili kut zakreta servomotora. Mjerenje se vrši komparatorom.

Komparator je precizni mjerni instrument koji mjeri odstupanje od određene mjere, a ne samu mjeru. Na referentnoj skali korištenog komparatora jedna puna rotacija strelice iznosi ukupno 1 mm, a svaki pomak iznosi 0,01 mm. Način mjerenja odstupanja izvodi se tako da se krilo servomotora prisloni na mjerno ticalo komparatora te očita vrijednost, pri tome vodeći računa o smjeru rotacije.

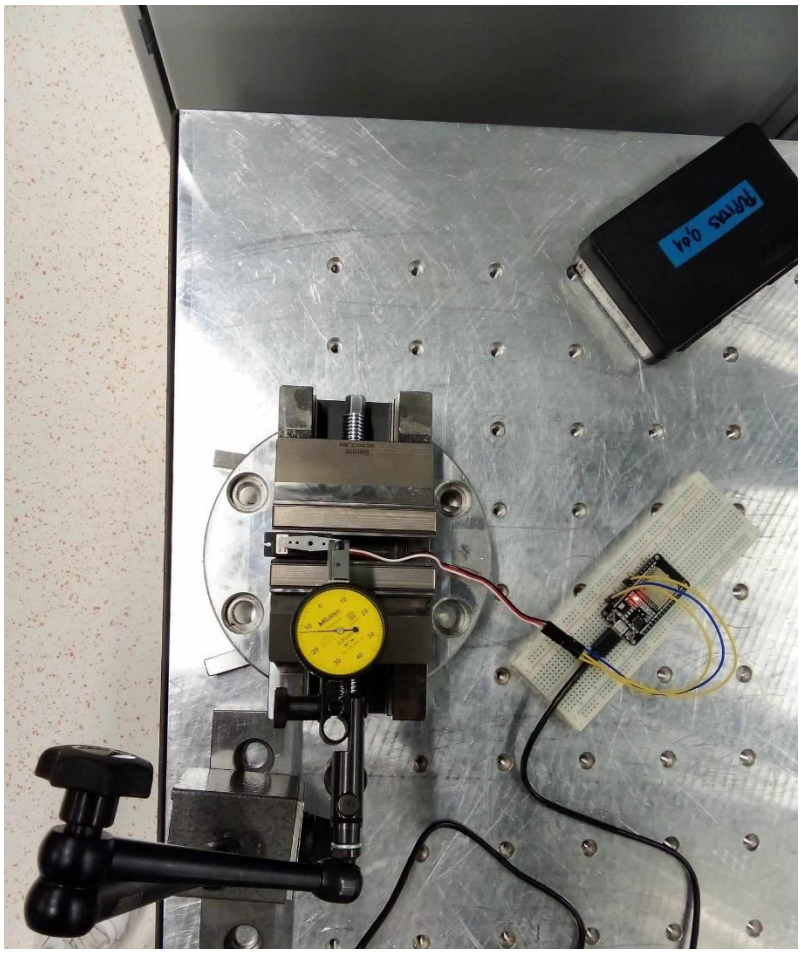
Slika 13. Komparator

Ponovljivost motora je podudaranje rezultata uzastopnih mjerenja istog kuta zakreta izvedenih u istim mjernim uvjetima. Mjerenje ponovljivosti servomotora je izvedeno s tri različita kuta zakreta i za svako je napravljeno devet mjerenja. Kod prvog mjerenja servomotoru je prvo poslana vrijednost kuta zakreta 45, zatim 50 te ponovno 45. Kada se položaj kraka servomotora prvo postavi na 45, taj je položaj na komparatoru postavljen u nulu. Zatim se postavi kut zakreta servomotora na 50, te se vrati na kut zakreta 45. Krajnji položaj servomotora, tj. vraćanje na vrijednost 45 se uspoređuje s početnom vrijednosti i očita se odstupanje.

Kod drugog mjerenja poslana je vrijednost 55, zatim 60 te ponovno 55, te kod trećeg je poslana vrijednost 70, zatim 75 te ponovno 70. Vrijednosti ponovljivosti su prikazane u tablici ispod.

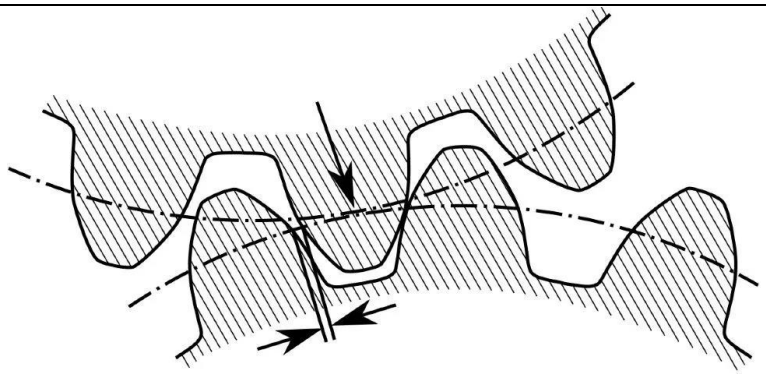
Tablica 3. Odstupanje od zadanog položaja servomotora

	Ponovljivost [mm]								
45-50-45	-0.1	-0.12	-0.13	-0.12	-0.13	-0.1	-0.12	-0.12	-0.12
55-60-55	-0.11	-0.11	-0.115	-0.109	-0.12	-0.10	-0.11	-0.105	-0.1
70-75-70	-0.12	-0.12	-0.13	-0.125	-0.13	-0.12	-0.125	-0.135	-0.13



Slika 14. Mjerenje ponovljivosti s komparatorom

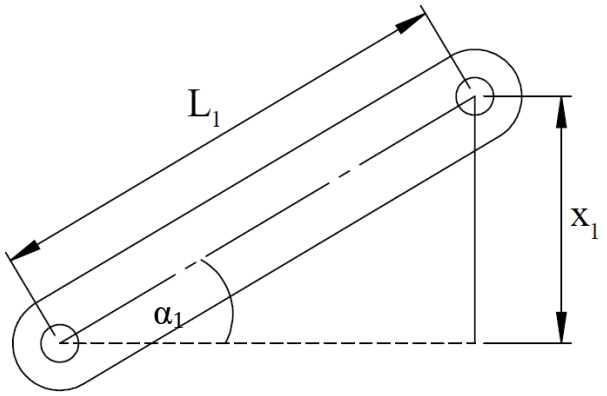
Mjerenjem ponovljivosti vidljivo je da je vrijednost uvijek viša od 0,1 mm. Upravo ta vrijednost govori o vrijednosti zračnosti unutar servomotora. Zračnost označava prazninu koja se može pojaviti unutar mehanizma servomotora, točnije između zubiju zupčanika unutar reduktora. Ta praznina se može očitati na komparatoru kad se promijeni rotacija servomotora, kao u slučaju mjerenja ponovljivosti iznad. [10]



Slika 15. Zračnost unutar zupčanika

Na komparatoru je također izmjerena rezolucija servomotora. Kad se servomotor zakrene za jedan korak, kazaljkom prijeđe 0,25 mm. Uzimajući u obzir zračnost od -0,10 mm, rezolucija servomotora iznosi 0,15 mm.

Rezolucija je veličina potrebna za određivanje kuta koji kazaljka motora prijeđe kad se zakrene za jedan korak.



Slika 16. Kut zakreta servomotora

Krak na kojem je mjerena rezolucija iznosi $L_1 = 12,5\,\text{mm}$, te pomak x_1 iznosi $x_1 = 0,15\,\text{mm}$.

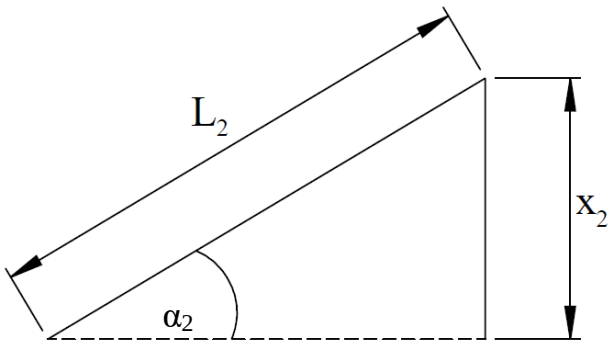
Kut se računa pomoću formule:

$$\alpha_1 = \arctan\left(\frac{x_1}{L_1}\right). \tag{4.1}$$

Uvrštavanjem poznatih veličina u formulu dobijemo zakret servomotora s jednim korakom:

$$\alpha_1 = \arctan\left(\frac{0,15}{12,5}\right) = 0,688°. \tag{4.2}$$

Isti postupak se ponavlja za zakret kazaljke na visinomjeru, gdje se mjeri koliko se zakrene kazaljka ako pokazuje visinu od 100, 200 i 300 metara.



Slika 17. Kut zakreta visinomjera

Kut se računa pomoću formule:

$$\alpha_2 = \arctan\left(\frac{x_2}{L_2}\right). \tag{4.3}$$

Krak L_2 na kojem je mjeren pomak iznosi $L_2 = 6\,\text{mm}$, a x_2 za visinu 100 m iznosi 0,06 mm. Kut za koji se kazaljka okrene iznosi:

$$\alpha_2 = \arctan\left(\frac{0,06}{6}\right) = 0,477°. \tag{4.4}$$

Mjerenjem pomaka zakreta kazaljke za druge vrijednosti visine vidljivo je da se kazaljka pomiče linearno.

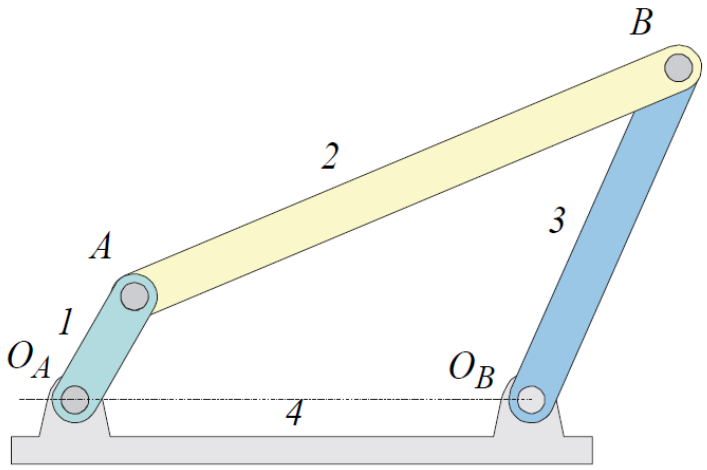
Tablica 4. Mjerenje kuta zakreta kazaljke visinomjera

Prikazana visina [m]	Pomak [mm]	Kut[°]
100	0,06	0,572
200	0,12	1,146
300	0,18	1,718

Kako je cilj da visinomjer pokazuje što točniju vrijednost trenutne visine, jednim korakom servomotora, tj. okretom za 0,688°, želimo da visinomjer napravi pomak za 25 metara, što je četvrtina kuta od 0,572°. Taj kut iznosi 0,143°.

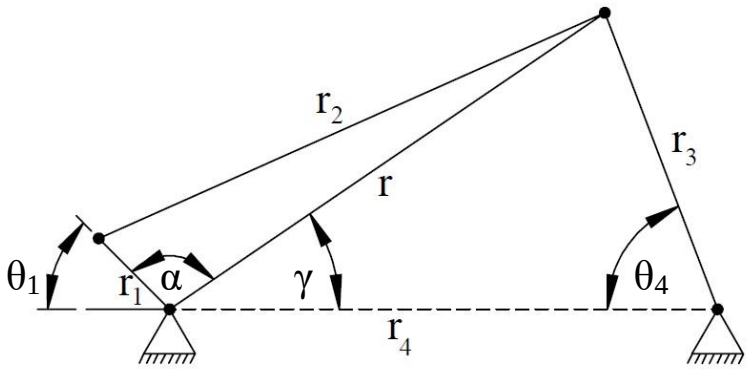
Kako bi se dobile željene vrijednosti, u konstrukciju visinomjera ugrađuje se mehanizam zglobnog četverokuta, kojem je potrebno odrediti vrijednosti krakova kako bi ulazni i izlazni kut bili usklađeni.

Zglobni četverokut je mehanizam koji za članove ima poluge međusobno povezane zglobovima. Promjenom duljina pojedinih članova mogu se ostvariti različita gibanja pogonskog i radnog člana.



Slika 18. Zglobni četverokut

Kod proračuna zglobnog četverokuta za visinomjer, r_1 je pogonski član, r_3 je radni član, r_4 je nepomični član te tražimo duljinu člana r_2. Uz osnovne članove, koriste se još i pomoćne veličine r i kutevi α i γ.



Slika 19. Proračun zglobnog četverokuta

Poznate vrijednosti četverokuta su ulazni kut θ_1 koji iznosi $\theta_1 = 0,688°$, te izlazni kut koji je povećan za 50° radi geometrije zglobnog četverokuta te iznosi $\theta_4 = 50,143°$. Također je

poznata udaljenost između servomotora i kazaljke koja je jednaka $r_4 = 28,74\,\text{mm}$. Uzimamo proizvoljne vrijednosti duljine kraka motora $r_1 = 6\,\text{mm}$, te $r_3 = 25\,\text{mm}$. Traži se duljina r_2 kako bi se zadovoljio početni uvjet.

Pomoću kosinusovog poučka dobije se vrijednost r:

$$r = \sqrt{r_4^2 + r_3^2 - 2r_3 r_4 \cos(\theta_4)} = 23,01\,\text{mm}. \tag{4.5}$$

Kut γ dobije se pomoću sinusovog poučka:

$$\frac{r_3}{\sin\gamma} = \frac{r}{\sin(\theta_4)}, \tag{4.6}$$

te se sređivanjem izraza dobije:

$$\gamma = \arcsin\left(\frac{r_3}{r}\sin(\theta_4)\right) = 56,52°. \tag{4.7}$$

Kut α iznosi:

$$\alpha = 180° - \gamma - \theta_4 = 122,79°, \tag{4.8}$$

te se pomoću kosinusovog poučka dobije vrijednost r_2:

$$r_2 = \sqrt{r_1^2 + r^2 - 2r_1 r\cos\alpha} = 26,7\,\text{mm}. \tag{4.9}$$

Kako se kutevi zglobnog četverokuta ne mijenjaju linearno, potrebno je provjeriti kolika je vrijednost r_2 ako se servomotor zakrene za određeni broj koraka. Maksimalni broj koraka servomotora iznosi 150, te u tom slučaju ulazni kut θ_1 imat će vrijednost:

$$\theta_1 = 150 \cdot 0,688° = 103,2°, \tag{4.10}$$

a izlazni kut će imati vrijednost:

$$\theta_4 = 50° + 150 \cdot 0,143° = 71,45°. \tag{4.11}$$

Uvrštavanjem vrijednosti (4.11) u jednadžbu (4.5) dobije se vrijednost $r = 31,52\,\text{mm}$, te uvrštavanjem u jednadžbu (4.7) dobije se $\gamma = 48,74°$. Dobije se vrijednost r_2 za 150 zakreta motora $r_2 = 26,4\,\text{mm}$.

Vrijednosti za ostale korake motora dobiju se pomoću programa *MATLAB* gdje se uvrštavanjem različitih vrijednosti dobije duljina r_2 kako bi se provjerila točnost rezultata.

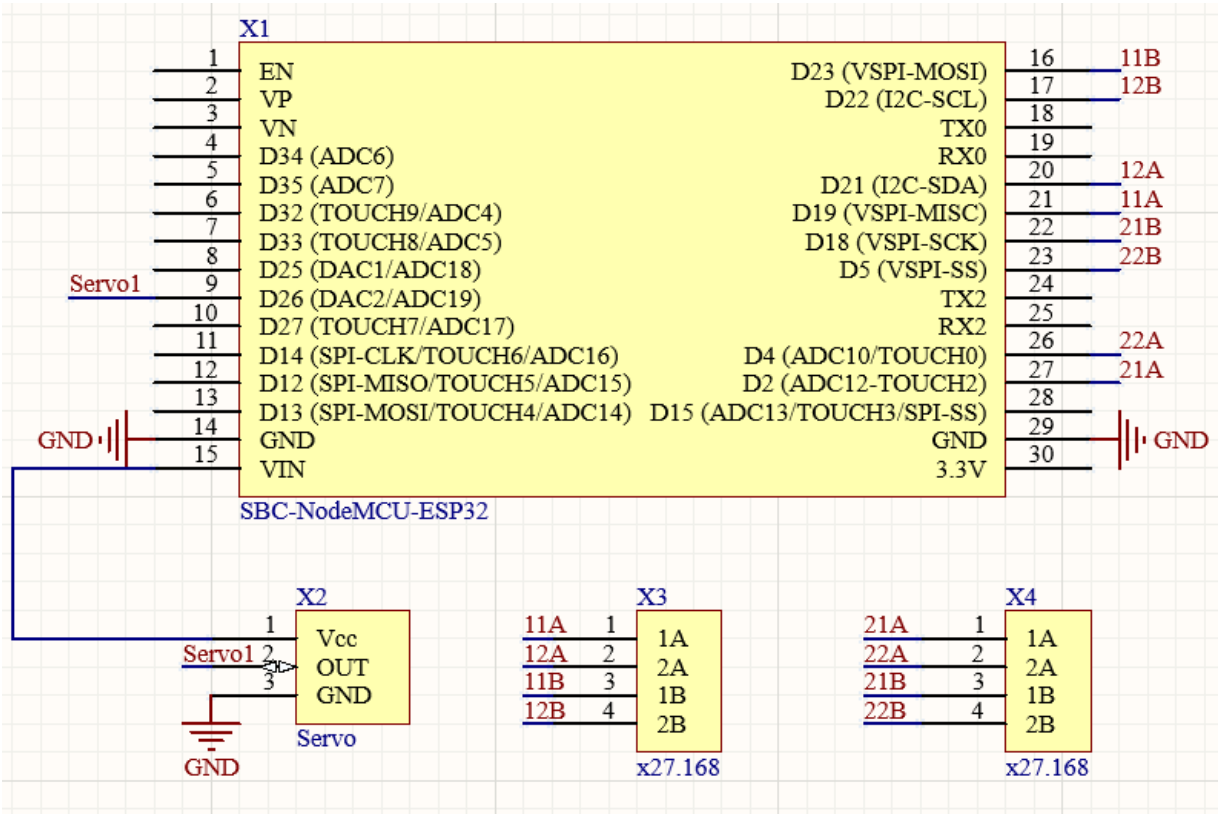
```
clear
format long g

r1=6;
r4 = 28.74;
r3 = 25;

kutkazaljke = atan(0.06/6)/4;
kutmotora = atan(0.15/12.5);

theta1 = 150*kutmotora; %kutmotora pomnoži se s brojem koraka motora
theta4 = 0.872+150*kutkazaljke;

r=sqrt(r4^2+r3^2-2*r3*r4*cos(theta4));
gamma = asin(r3/r*sin(theta4));
alfa = pi-gamma-theta1;
r2 = sqrt(r1^2+r^2-2*r1*r*cos(alfa));
```

Slika 20. *MATLAB* program za provjeru duljina zglobnog četverokuta

Slične vrijednosti se dobiju i za preostale vrijednosti zakreta motora, te uzimamo vrijednost $r_2 = 26,5\,\text{mm}$.

5. ELEKTRIČNA SHEMA

Shema spajanja napravljena je pomoću programa *Altium Designer*. Na mikroupravljačku pločicu SBC-NodeMCU-ESP32 spajaju se dva koračna motora x27.168 i servomotor FS 155 BB MG.



Slika 21. Električna shema

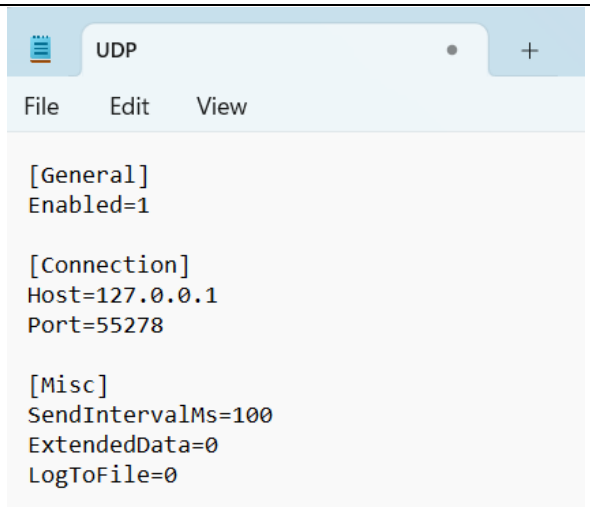
6. PROGRAM ZA SIMULACIJU LETENJA

Podaci za upravljanje gibanjem koračnog motora dobivaju se iz programa za simulaciju letenja *Condor Soaring Simulator*. Program simulira realistično upravljanje jedrilicom na računalu te uključuje veliki izbor jedrilica kao i mnogo lokacija letenja pod raznim vremenskim uvjetima koji su promjenjivi tijekom trajanja simulacije leta. Simulator se uz rekreacijske svrhe može koristiti kao i početna točka za edukaciju pilota. [5]

Jedan od razloga korištenja *Condor Soaring* simulatora u ovom radu je mogućnost dobivanja podataka o brzini, visini i vertikalnoj visini u svakom trenutku simuliranog leta preko NMEA (National Marine Electronics Association) protokola. NMEA je standard u navigaciji koji omogućuje slanje podataka preko GPS uređaja u obliku ASCII linija putem Bluetooth, WiFi i ostalih načina povezivanja. [6]

Slika 22. ***Condor Soaring*** **simulator**

Condor Soaring koristi UDP protokol za slanje podataka o letu. UDP (User Datagram Protocol) je protokol koji ne zahtjeva povezivanje prije prijenosa podataka te nema provjere poslanih podataka što ga čini nepouzdanijim, ali efikasnijim prijenosom. Taj način prijenosa je koristan kod slanja podataka koji se moraju slati velikom brzinom, kao u slučaju programa *Condor Soaring*. Da bismo omogućili prijenos podataka, u postavkama simulacije, *Enabled* mora biti postavljen na vrijednost 1.



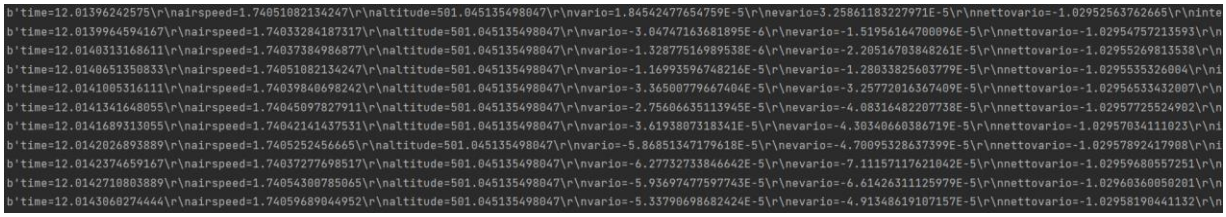
Slika 23. UDP postavke

Pod *Connection* se nalazi IP adresa kao i port gdje simulator šalje podatke. Kako bismo uspješno dobili podatke iz simulatora, potrebno je osmisliti program koji prima podatke na istom portu gdje se šalju podaci iz simulatora. Taj port se naziva *socket*. Program za primanje podataka je napravljen pomoću Python programskog jezika.

```
import socket
import serial

arduinoWrite = serial.Serial("COM7", 9600, timeout=2)
port = 55278
ip = "127.0.0.1"

s = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
s.bind((ip, port))
print("Ready")
```

Slika 24. Uspostava čitanja i slanja podataka

Nakon uspostave primanja podataka, pokretanjem simulatora na konzoli se ispisuju podaci o brzini, visini, vertikalnoj visini te ostalim podacima o trenutnom položaju u simulatoru.



Slika 25. Dobiveni podaci iz simulatora

Podaci se dobivaju u obliku byte varijable. Da bi podaci bili pregledniji, byte podaci se moraju pretvoriti u String, kako bismo mogli pomoću ugrađenih funkcija u Python programskom jeziku za String varijable lakše upravljali podacima i iščitavali potrebne vrijednosti. Kako bismo odredili koji je podatak brzina, visina te vertikalna brzina, String pomoću ugrađene funkcije podijelimo u segmente; kada je očitan novi red sprema se sljedeća vrijednost u listu. Brzinu je sada lako očitati, kako se uvijek nalazi kao drugi element u listi, kao i visinu te vertikalnu brzinu, koji su drugi i treći element liste.

```
airspeedString = values[1]
airspeedFinal = airspeedString[9:]
airspeedInt = airspeedFinal.split('.')
airspeed = airspeedInt[0]
print("Brzina: ", airspeed)
```

Slika 26. Dobivanje vrijednosti iz niza podataka

Nakon što su očitane sve vrijednosti moraju se pretvoriti nazad u byteove kako bi se mogli poslati serijskom vezom na upravljačku pločicu povezanom putem Bluetooth veze sa računalom.

Kako su podaci poslani preko Bluetooth komunikacije oblika byte, nije moguće poslati cjelokupnu očitanu vrijednost brzine, vertikalne brzine i visine iz simulatora, nego se šalje jedna po jedna znamenka. U ovakvom slučaju, kako bismo prepoznali kojoj vrijednosti pripada pojedina znamenka, potrebno je uvesti pomoćne znamenke radi lakšeg snalaženja nakon slanja podataka. Primjer jednog poslanog niza znamenki u ovom slučaju glasi „90S5V1998H“, gdje 90 označava brzinu, nakon čega slijedi slovo „S“ koje označava kraj znamenki koji se odnose na brzinu. Slično se može zaključiti za ostatak niza, gdje „V“ označava vertikalnu brzinu, tj. vario, a „H“ označava visinu. Ovakav niz se šalje preko Bluetooth komunikacije svakih 100 milisekundi.

7. ARDUINO KOD

Potrebne *library* komponente programa su:

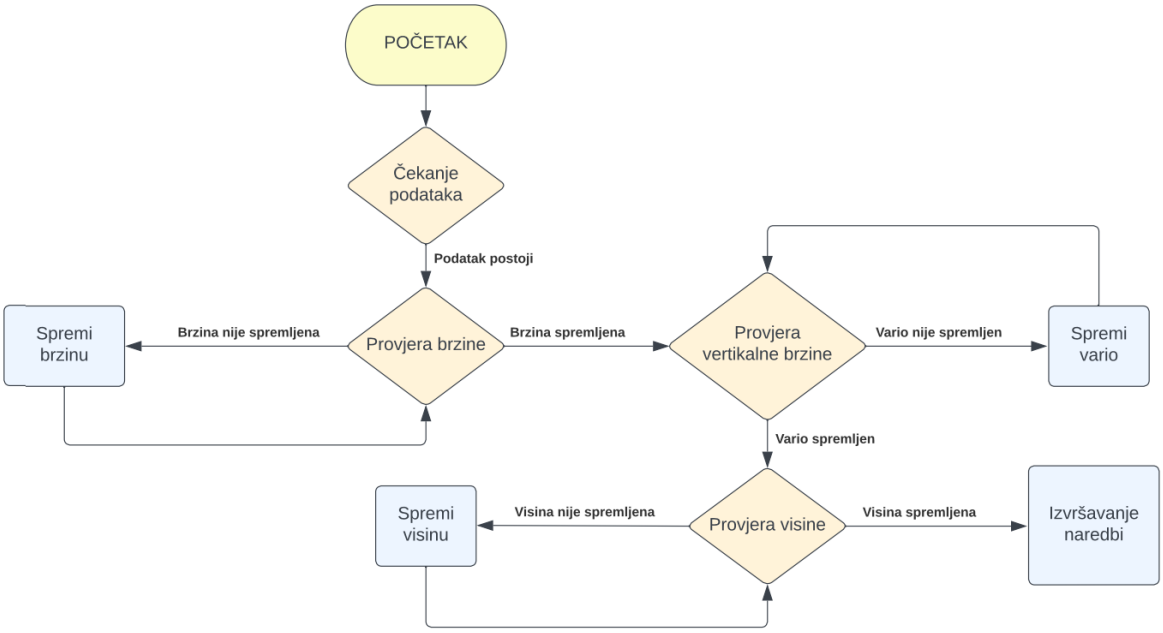
- *Stepper.h* koja sadrži funkcije za upravljanje koračnim motorima, kao i *ESP32Servo* za upravljanje servomotorom
- *BluetoothSerial.h* koja sadrži sve potrebne komponente za uspravljanjem Bluetooth komunikacijom
- *ESP32Servo.h* koja sadrži sve komponente za upravljanje mikroupravljačkom pločicom SBC-NodeMCU-ESP32

Definiraju se pinovi na kojima se nalaze koračni motori i servomotor, zatim te pinove pridružimo motorima pomoću funkcija *Stepper* i *Servo*.

```
#include <Stepper.h>
#include <BluetoothSerial.h>
#include <ESP32Servo.h>

#define  STEPS  600
#define  COIL1  19
#define  COIL2  21
#define  COIL3  22
#define  COIL4  23

#define  COIL5  2
#define  COIL6  4
#define  COIL7  5
#define  COIL8  18

Stepper stepper(STEPS, COIL1, COIL2, COIL3, COIL4);
Stepper stepper2(STEPS, COIL5, COIL6, COIL7, COIL8);
Servo servo1;
int servoPin = 26;
```

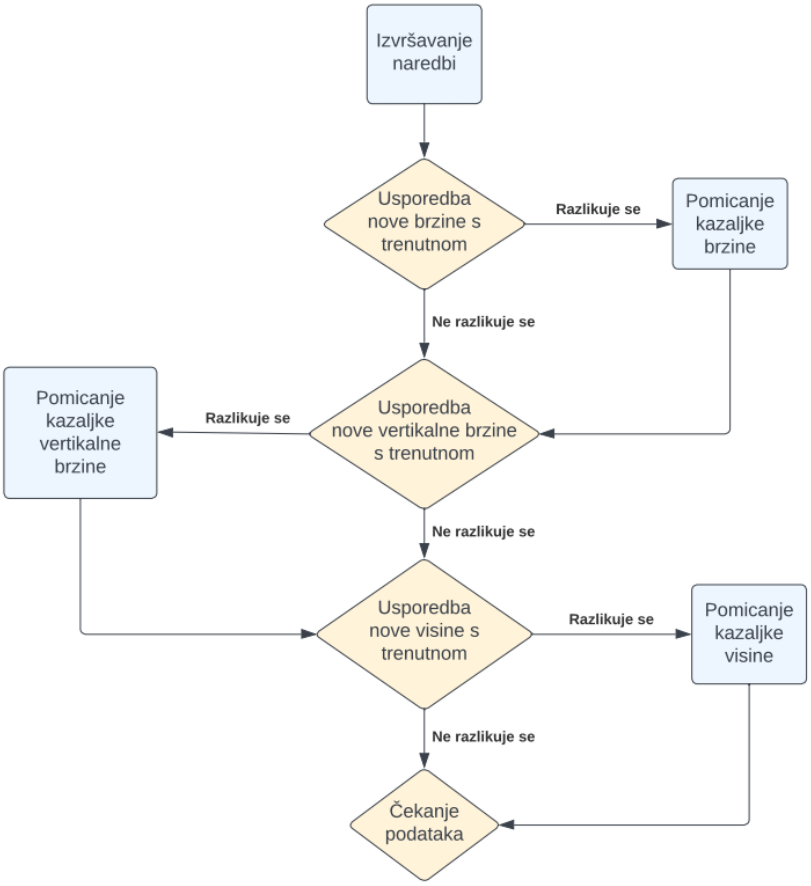
Slika 27. Inicijalizacija Arduino koda

Nakon što se podaci iz simulatora pošalju serijskom vezom preko Bluetooth komunikacije na upravljačku pločicu, poslani byteovi se moraju pretvoriti u brojčane vrijednosti kako bi se koračni motori zakrenuli za odgovarajući kut. To čini prvi dio programa opisan dijagramom ispod.



Slika 28. Tijek prvog dijela programa

Program prvo provjerava postoje li podaci koji se šalju. U tom slučaju prvo se čitaju znamenke koje se spremaju u niz znamenki koji označavaju brzinu, sve dok se ne očitava slovo „S“, što označava kraj čitanja znamenki za brzinu. Sljedeće znamenke se spremaju kao znamenke vertikalne brzine dok se ne očita slovo „V“ te na kraju se spremaju znamenke visine. Kad se očita slovo „H“, niz se očitao do kraja te se dobiveni nizovi pretvaraju u brojčane vrijednosti. Nakon što su dobivene sve tri vrijednosti, program kreće u slanje podataka na motore.



Slika 29. Tijek drugog dijela programa

Dobivene vrijednosti se prvo pretvaraju u broj koraka za koji se motori moraju zaokrenuti da označavaju pravu vrijednost. To se postigne tako da se brojčane vrijednosti skaliraju u odnosu na puni kut zakreta motora. Ta vrijednost čini trenutnu poziciju pojedinog motora te se svaka sljedeća dobivena vrijednost uspoređuje s trenutnom pozicijom te se motor zakrene za tu vrijednost. Dodatno, potrebno je prilagoditi vrijednost brzine zbog nepravilnosti skale.

8. ZAKLJUČAK

U ovom radu prikazano je kako se upravljanje jedrilicom na simulatoru pomoću računala može fizički prikazati na instrumentima koji u pravom vremenu pokazuju točne vrijednosti. Jedna od ograničenja koje je uzeto u obzir koja je posebito važna je brzina slanja podataka instrumentima. Zbog ograničenja na upravljačkoj ploči, bilo je nužno reducirati brzinu slanja podataka, što je rezultiralo sporijim prikazom vrijednosti na instrumentima.

Iako ovaj rad obuhvaća cijeli tok dohvaćanja podataka do njihovog prikaza, moguće ga je nadograditi za neke druge primjene, poboljšati korištenjem bolje upravljačke pločice za optimizaciju cijelog toka ili omogućiti dohvaćanje podataka direktno u Arduino sučelju, što je kompleksije i zahtjeva dodatne dijelove, no smanjuje broj potrebnih programa za obradu podataka.

LITERATURA

[1] https://joy-it.net/en/products/SBC-NodeMCU-ESP32 - 23.04.2023

[2] https://www.conrad.com/p/joy-it-development-board-node-mcu-esp32-modul-1656367 - 23.04.2023

[3] https://www.adafruit.com/product/2424 - 29.05.2023

[4] https://www.aliexpress.com/item/4001007528480.html - 29.05.2023

[5] https://www.condorsoaring.com/about/ - 29.05.2023

[6] https://www.gpsworld.com/what-exactly-is-gps-nmea-data/ - 29.05.2023

[7] https://manualzz.com/doc/12427411/condor-manual-1.1.5--page-1-of-43 - 29.05.2023

[8] https://www.geeksforgeeks.org/user-datagram-protocol-udp/ - 29.05.2023

[9] https://www.robbe.com/de/RC-Elektronik/Servos-Zubehoer/Servos-Digital-High-Voltage/Robbe-Modellsport-FS-155-BB-MG-HV-DIGITAL-SERVO/9111 - 25.07.2023

[10] https://blog.orientalmotor.com/gear-basics-backlash-vs-lost-motion - 26.07.2023

[11] https://industryinsider.eu/maintenance/compressed-air-drives-pneumatics/what-is-gearbox-backlash/ - 26.07.2023

[12] https://www.tme.eu/hr/news/library-articles/page/41861/korachni-motori-vrste-i-primjeri-primjene-korachnih-motora/ - 08.09.2023

[13] https://www.electricaleasy.com/2015/01/how-does-servo-motor-work.html -08.09.2023

[14] Husnjak, M.: Teorija mehanizama, podloge za predavanja, Fakultet strojarstva i brodogradnje, Zagreb, 2009. – 01.09.2023

PRILOZI

I. Python kod

II. Arduino kod

PRILOG I

```
import socket
import serial

arduinoWrite = serial.Serial("COM7", 9600, timeout=2)
port = 55278
ip = "192.168.1.182"

s = socket.socket(family=socket.AF_INET,
type=socket.SOCK_DGRAM)
s.bind((ip, port))
print("Ready")

while True:
    data, addr = s.recvfrom(1024)
    dataString = data.decode('UTF-8')

    values = dataString.split('\n')

    airspeedString = values[1]
    airspeedFinal = airspeedString[9:]
    airspeedInt = airspeedFinal.split('.')
    airspeed = airspeedInt[0]
    print("Brzina: ", airspeed)                          #provjera

    varioString = values[3]
    varioFinal = varioString[6:]
    varioInt = varioFinal.split('.')
    vario = varioInt[0]
    print("Vario: ", vario)                              #provjera

    heightString = values[2]
    heightFinal = heightString[9:]
    heightInt = heightFinal.split('.')
    height = heightInt[0]
    print("Visina: ", height)                            #provjera

    speedMarker = 'S'                    #pretvaranje u byteove
    speedMarkerBytes = speedMarker.encode('UTF-8')
    varioMarker = 'V'
    varioMarkerBytes = varioMarker.encode('UTF-8')
    heightMarker = 'H'
    heightMarkerBytes = heightMarker.encode('UTF-8')
    newText = '\n'
    newTextBytes = newText.encode('UTF-8')

    airspeedBytes = airspeed.encode('UTF-8')
    varioBytes = vario.encode('UTF-8')
    heightBytes = height.encode('UTF-8')
```

```
    arduinoWrite.write(newTextBytes)    #slanje preko Bluetooth
    arduinoWrite.write(airspeedBytes)
    arduinoWrite.write(speedMarkerBytes)
    arduinoWrite.write(varioBytes)
    arduinoWrite.write(varioMarkerBytes)
    arduinoWrite.write(heightBytes)
    arduinoWrite.write(heightMarkerBytes)
```

PRILOG II

```
#include <Stepper.h>
#include <BluetoothSerial.h>
#include <ESP32Servo.h>

#define  STEPS  600
#define  COIL1  19
#define  COIL2  21
#define  COIL3  22
#define  COIL4  23

#define  COIL5  2
#define  COIL6  4
#define  COIL7  5
#define  COIL8  18

Stepper stepper(STEPS, COIL1, COIL2, COIL3, COIL4);
Stepper stepper2(STEPS, COIL5, COIL6, COIL7, COIL8);
Servo servo1;
int servoPin = 26;

BluetoothSerial SerialBT;
void setup() {
  Serial.begin(9600);
  SerialBT.begin("Esp32test");
  stepper.setSpeed(30);
  stepper.step(630);
  stepper2.setSpeed(30);
  stepper2.step(630);
  servo1.attach(servoPin);
}
char c;
String speedms;
String vario;
String height;
char speedMarker = 'S';
char varioMarker = 'V';
char heightMarker = 'H';
boolean speedTrue = false;
boolean varioTrue = false;
boolean heightTrue = false;
int speedInt;
int varioInt;
int heightInt;
int speedkm;
int posSpeed=0;
int posVario=0;
int posHeight=0;

void loop() {
```

```
  while(SerialBT.available()){
    c = SerialBT.read();
    if(c!=speedMarker && c!=varioMarker && c!=heightMarker &&
speedTrue==false){
      speedms.concat(c);
    }
    else if(c==speedMarker){
      speedTrue=true;
    }
    else if(c!=speedMarker && c!=varioMarker &&
c!=heightMarker && speedTrue==true && varioTrue==false &&
heightTrue==false){
      vario.concat(c);
    }
    else if(c==varioMarker){
      varioTrue=true;
    }
    else if(c!=speedMarker && c!=varioMarker &&
c!=heightMarker && speedTrue==true && varioTrue==true &&
heightTrue==false){
      height.concat(c);
    }
    else if(c==heightMarker){
      heightTrue=true;
    }
    else if(varioTrue && speedTrue && heightTrue){
      varioTrue=false;
      speedTrue=false;
      heightTrue = false;
      speedInt = speedms.toInt();
      speedkm = speedInt*3.6;
      varioInt = vario.toInt();
      heightInt = height.toInt();
      /*----------------------BRZINA---------------------*/
      /*Serial.print("Brzina [km/h]: ");
      Serial.println(speedkm);
      Serial.print("Vario [m/s]: ");
      Serial.println(varioInt);
      Serial.print("Visina [km]: ");
      Serial.println(heightInt);
      Serial.print("Brzina poslije mapiranja: ");*/
      if(speedkm<=50){
        speedkm= map(speedkm,0,50,0,20);
      }
      else if(speedkm>50 && speedkm<=70){
        speedkm= map(speedkm,51,70,21,60);
      }
      else if(speedkm>70 && speedkm<=100){
        speedkm= map(speedkm,71,100,61,150);
      }
      else if(speedkm>100 && speedkm<=150){
```

```
        speedkm= map(speedkm,101,150,151,320);
      }
      else if(speedkm>150 && speedkm<=200){
        speedkm= map(speedkm,151,200,321,480);
      }
      else if(speedkm>200 && speedkm<=250){
        speedkm= map(speedkm,201,250,481,630);
      }
      //Serial.println(speedkm);
      while(abs(speedkm - posSpeed)> 2){         //if
diference is greater than 2 steps.
        if((speedkm - posSpeed)> 0){
          stepper.step(-1);
          posSpeed++;
          }
        if((speedkm - posSpeed)< 0){
          stepper.step(1);
          posSpeed--;
          }
      }
      /*----------------------VARIO---------------------*/
      //Serial.print("Vario poslije mapiranja: ");
      if(varioInt>5){
        varioInt=630;
      }
      else if(varioInt<-5){
        varioInt=0;
      }
      else{
        varioInt= map(varioInt,-5,5,0,630);
      }
      //Serial.println(varioInt);
      while(abs(varioInt - posVario)> 2){
        if((varioInt - posVario)> 0){
          stepper2.step(-1);
          posVario++;
          }
        if((varioInt - posVario)< 0){
          stepper2.step(1);
          posVario--;
          }
      }
      /*----------------------VISINA---------------------*/
      Serial.print("Visina poslije mapiranja: ");
      heightInt = map(heightInt, 2750, 0, 40, 150);
      if(heightInt!=0){
        servo1.write(heightInt);
        Serial.println(heightInt);
      }

      vario = "";
```

```
      speedms = "";
      height = "";
      delay(50);
      }
    }
}
```