Samostabilzirajuća platforma s tri stupnja slobode gibanja

Vugić, Martin

Master's thesis / Diplomski 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:235:387675

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Martin Vugić

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor: Student:

Prof. dr. sc. Željko Šitum, dipl. ing. Martin Vugić

Zagreb, 2023.

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

Zahvaljujem se svom mentoru prof. dr. sc. Željku Šitumu što mi je omogućio i pomogao da napišem ovaj rad, te svojoj obitelji i prijateljima koji su mi pomagali i davali podršku pri studiranju.

Martin Vugić

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

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

DIPLOMSKI ZADATAK

Student: **MARTIN VUGIĆ** Mat. br.: 0035202291

Naslov rada na hrvatskom jeziku: **Samostabilizirajuća platforma s tri stupnja slobode gibanja**

Naslov rada na engleskom jeziku: **Self-stabilizing platform with three degrees of freedom**

Opis zadatka:

Za održavanje stabilnog ravnotežnog položaja kamere tijekom snimanja ili nekog drugog predmeta koristi se motorizirana platforma s tri stupnja slobode gibanja izrađena u obliku kardanskog zgloba (eng. gimbal). Operater može čitavi sklop proizvoljno zakretati u prostoru, a korištenjem povratnih veza po kutevima zakreta platforme u tri osi održava se njezin horizontalni položaj. U većini slučajeva, mjerenje kuteva zakreta platforme ostvaruje se korištenjem žiroskopa/akcelerometra postavljenog na sredini platforme. Korištenjem tri odvojena motora za stabilizaciju platforme mehanizam reagira na zakretanje sklopa, a upravljački algoritam izračunava upravljačke signale koji se dovode na pogonske aktuatore, kako bi se poništilo neželjeno zakretanje platforme. Stoga, ovakav mehanizam predstavlja ogledni primjer korištenja složenog algoritma automatske regulacije u tipičnom mehatroničkom sustavu.

U radu je potrebno:
- projektirati i izraditi motoriziranu platformu s tri stupnja slobode gibanja, čije se zakretanje vrši pomoću električnih motora,
- opisati korištene komponente pogonskog, upravljačkog i mjernog dijela sustava,
- razmotriti moguće načine regulacije sustava, načiniti sintezu regulatora i izvršiti simulaciju procesa te odabrati najpovoljnije rješenje s obzirom na praktičnu izvedbu,
- razvijene regulacijske algoritme eksperimentalno provjeriti na izrađenoj maketi sustava.

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

Zadatak zadan:
17. studenog 2022.

Rok predaje rada:
19. siječnja 2023.

Predviđeni datum obrane:
23. siječnja do 27. siječnja 2023.

Zadatak zadao:
prof. dr. sc. Željko Šitum

Predsjednica Povjerenstva:
prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS TEHNIČKIH CRTEŽA

BROJ CRTEŽA	**Naziv iz sastavnice**
ST_PLAT-000-001	Uška za platformu
ST_PLAT-000-002	Platforma
ST_PLAT-000-003	Držač servomotora
ST_PLAT-000-004	Kućište
ST_PLAT-000-005	Poklopac kućišta
ST_PLAT-001-000	Troosna samostabilizirajuća platforma

POPIS OZNAKA

Oznaka	Jedinica	Opis
$\alpha,\beta,\gamma,\theta,\phi$	° ili rad	kut
U	V	napon
C	Ah	kapacitet baterije
I	A	jakost el. struje
t	s	vrijeme
m	kg	masa
ω	rad/s	kutna brzina
s	mm	dužina
g	ms^{-2}	gravitacijsko ubrzanje
f	Hz	frekvencija
C	F	električni kapacitet

Kratice	Opis
IMU	Inertial Measurement Unit
MEMS	Micro-electro-mechanical systems
PWM	Pulse-width Modulation
DC	Direct Current
EEPROM	Electrically Erasable Programmable Read-Only Memory
DMP	Digital Motion Processor
VDD	Voltage (at) drain
CPU	Central Processing Unit
LSB	Least significant bit

SAŽETAK

Prilikom kretanja, odnosno hodanja, trčanja ili pri jednostavnom pomaku rukom, otežano je održavati predmet koji nosimo u horizontalnoj ravnini zbog samih pokreta ljudskog tijela koji se pri tome obavljaju. Primjerice, fotografima uvelike pomažu stabilizirajuće platforme (eng. *gimbal*) kod obavljanja njihovog posla zbog toga što će korištena kamera uvijek biti horizontalno postavljena bez obzira na pokrete koje rade ili u kojoj se poziciji nalaze kako bi kadar, pa u konačnici slika bila zadovoljavajuća. Troosna stabilizirajuća platforma ili gimbal stabilizator je uređaj koji služi za kontrolu orijentacije u 3D prostoru kako bi predmet ostao u horizontalnoj ravnini. Primjena stabilizirajuće platforme, osim u navedenom primjeru fotografiranja, koristi se i u filmskoj industriji, na brodu (kompas, brodski elementi poput pećnice, stolova i dr.), pa čak i u medicini.

U ovom radu bit će izrađena troosna samostabilizirajuća platforma koja je upravljana korištenjem *Arduino*-a, *open-source* platforme za kreiranje elektroničkih prototipova bazirana na sklopovlju i programskom paketu koji je fleksibilan i jednostavan za korištenje, 3D modeliranja, 3D printanja, te korištenjem ostalih komponenti potrebnih za sustav poput servomotora, razvojne ploče, pretvarača napona i dr.

Ključne riječi: Stabilizirajuća platforma; mikrokontroler Arduino; žiroskop i akcelerometar MPU-6050; 3D modeliranje;

SUMMARY

While moving, i.e. walking, running or simple hand movements, it's difficult to keep object steady in horizontal position due to movements of the human body. For example, stabilizing platforms (Gimbals) help photographers during their work by stabilizing camera in horizontal position regardless of the movements they are making or in which position they are in order for the frame, therefore the image, to be significantly better. A three-axis stabilizing platform or gimbal is a device used to control the orientation and rotation of the object in 3D space so it can remain in horizontal position. The application of the gimbal, apart from the aforementioned example in photography, is also used in the film industry, on ships (compass, ship elements such as ovens, tables, etc.), and even in medicine.

In this paper, a three-axis self-stabilizing platform will be created using Arduino, an open-source platform for creating electronic prototypes based on a circuit and a software package that is flexible and easy to use, 3D modeling, 3D printing, and using other components necessary for a system such as a servo motor, a development board, a voltage converter, etc.

Key words: Self-stabilizing platform; Arduino microcontroller; Gyroscope and accelerometer MPU-6050; 3D modelling;

1. UVOD

Aktivna stabilizacija postoji već jako dugo kod fotografiranja i u filmskoj industriji, gdje se stabilizirajuće platforme koriste za stabilizaciju kamera prilikom kreiranja fotografskih, odnosno videozapisa. Stabilizacija kamere danas je dostupna i kod amaterskih fotografa i filmaša pojavom praktičnih i komercijalno dostupnih troosnih stabilizirajućih platformi, koje ćemo u ovom radu često nazivati gimbal stabilizatori radi pojednostavljenog izraza koji je već opće prihvaćen u svijetu. Kako danas veliki broj ljudi koristi pametne mobitele s dobrim kamerama ili kamere manjih proporcija, vrlo je jednostavno i pristupačno montirati i pričvrstiti kameru za troosni gimbal stabilizator (slika 1.) i dobiti relativno kvalitetne video zapise ili fotografije bez prevelikih trzaja kamere dok se čovjek kreće prilikom snimanja.

Slika 1 Gimbal stabilizator [1]

Snimatelji i fotografi su od davnih dana tražili načine za stabilizaciju svojih uređaja. To je bilo poprilično teško s glomaznim i staromodnim aparatima, ali kako su ti fotoaparati postali sve manjih dimenzija, bilo ih je lakše i stabilizirati. Osim običnih tronožaca ili postolja za kamere, među prvim pokušajima stabilizacije kamere bio je aeroskop (eng. *aeroscope*) 1910. godine.

Aeroskop (slika 2.) je vrsta kamere koja u sebi sadrži stlačeni zrak za pravljenje filmova i slika, ali isto tako je bila i prva uspješno napravljena kamera s ručnim upravljanjem [2]. Pokreće ju stlačeni zrak koji se prije snimanja upumpava u kameru jednostavnom ručnom pumpom.

Snimajući aeroskopom, snimatelj nije morao okretati ručicu za pomicanje materijala za snimanje što je pridonijelo stabilizaciji jer je mogao upravljati objema rukama, držeći kameru i kontrolirati fokus za razliku od ostalih kamera tog doba.

Slika 2 Aeroskop [2]

U filmskoj industriji 70-ih, prije pojave prvih pravih stabilizacija kamere, redatelj je imao izbor za snimanje pokretnih snimaka poput [3]:

1. Kamera koja se može montirati na kolica (engl. Dolly), nosač s kotačima koji se kotrlja po tračnicama ili glatkoj površini (slika 3)

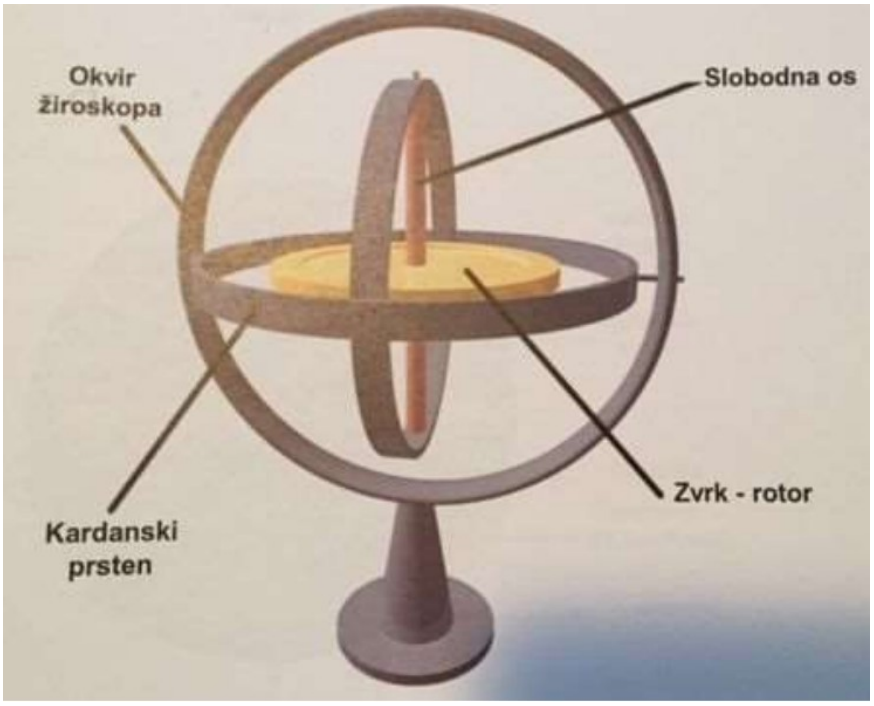
2. Kran ili ruka s protuutegom na koju se montira kamera radi okomitih i vodoravnih pomicanja
3. Kamera manjih dimenzija koja se drži s rukama ili uz pomoć ručke

Slika 3 Kolica za kameru (eng. *Dolly*)

Neki od navedenih načina se koriste i danas. Najveća revolucija stabilizacije kamera pojavila se 1975. godine od tvrtke Steadicam koja je napravila stabilizirajuću vestu (slika 4) za koju možemo reći da je preteča današnjih stabilizatora manjih dimenzija.

Slika 4 Steadicam stabilizirajuća vesta [4]

No, stabilizirajuće veste i gimbal stabilizatori novijeg doba ne mogu funkcionirati bez instrumenta koji se zove žiroskop. Žiroskop služi za stabilizaciju unutrašnje platforme (zvrk ili rotor) oko osi simetrije i ovješen je tako da os rotacije može slobodno mijenjati svoj pravac u prostoru. Građen je od jednog, dvaju ili triju prstena koji se nazivaju kardanski prsteni ili „gimbali“ koji funkcioniraju kao zakretni nosači kako bi omogućili stabilizaciju unutrašnje platforme (objekta) prilikom zakretanja osi. Na slici 4 prikazan je troosni žiroskop u kardanskom ovjesu.



Slika 4 Žiroskop u kardanskom ovjesu [5]

Matematičke osnove žiroskopa (slika 4) postavio je Leonhard Euler 1765. uz pomoć *Euler*-ovih kutova, a Léon Foucault konstruirao je napravu 1851. godine koja mu je služila za demonstraciju činjenice da Zemlja djeluje kao zvrk, nazivajući je žiroskop [6].

Kod stabilizirajućih vesti, osim same veste, operator nosi pojas koji je pričvršćen za izoelastičnu ruku. Sve je to povezano sa višeosnim kardanskim spojevima, koji čine žiroskop, sa jako niskim trenjem spojeva kako bi kamera „klizila“. Da bi kamera ostala uspravna, moraju se staviti protuutezi kako bi dno sustava bilo malo teže, okrećući se oko kardanskog zgloba. Zbog same težine prijašnjih kamera, dodajući tome ostale komponente veste uz dodatne utege za balans, cijeli sklop je imao veliku masu koju nije lako pomaknuti malim pokretima tijela operatora.

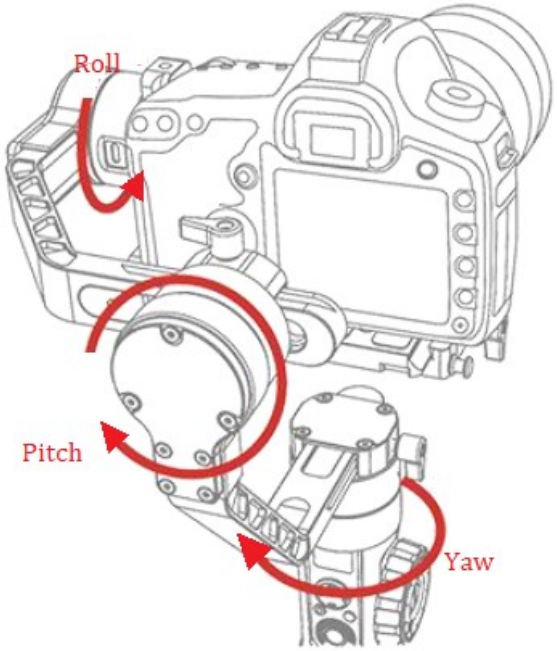
Rastuća popularnost praktičnih DSLR (engl. *Digital single-lens reflex*) i GoPro kamera (slika 5) koje teže manje, svega nekoliko kilograma, omogućila je savršenu priliku za razvoj i popularnost ručnih gimbal stabilizatora. Iako stabilizirajuće veste tvrtke Steadicam imaju svoje

prednosti jer se oslanjaju na inerciju operatora, gimbal stabilizatori su puno manji, praktičniji i lakši, te ih je daleko jednostavnije koristiti uz mogućnost višeosne stabilizacije [7].

Slika 5 GoPro kamera s gimbal stabilizatorom [8]

2. RAZRADA IDEJE

Da bismo krenuli sa razradom ideje, prvo moramo postaviti koordinatni sustav stabilizirajuće platforme koja se može vidjeti na slici 5.



Slika 6 Koordinatni sustav stabilizirajuće platforme [9]

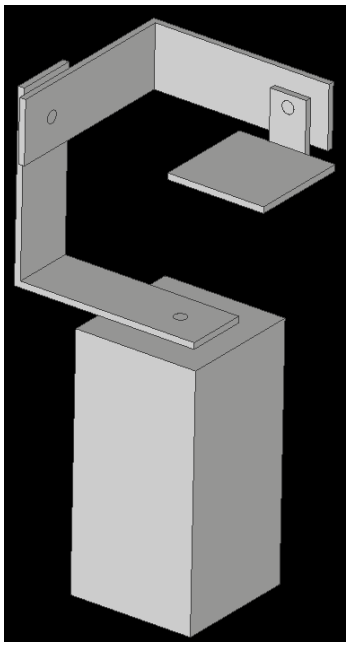
Koordinatne osi označili smo sa učestalim izrazima koji se koriste za stabilizirajuće platforme i za sustave slične uređajima poput dronova ili aviona. Možemo ih svrstati na:

- Yaw – Z-os (skretanje)
- Pitch – Y-os (poniranje)
- Roll – X-os (valjanje)

Nakon postavljanja koordinatnog sustava, moramo konstruirati cjelokupni sklop stabilizirajuće platforme. Kućište mora biti dovoljno veliko kako bi stale sve komponente korištene u sustavu, a to su:

- mikrokontroler ,
- žiroskop i akcelerometar,
- pretvarač napona,
- litijske baterije,
- servomotori,
- mnogi drugi elementi poput žica, prekidača, konektora i dr.

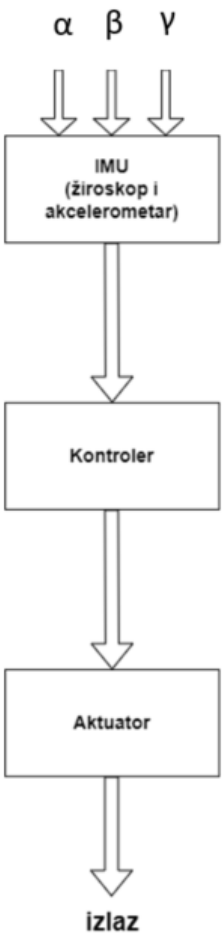
Uz kućište, konstruirat će se i držači motora, kao i postolje. Skica konstrukcije sklopa prikazana je na slici 7.



Slika 7 Skica konstrukcije troosne samostabilizrajuće platforme

Osim samog 3D modeliranja, provjerit će se i ispravnost inercijalne mjerne jedinice (eng. *inertial measurement unit* – IMU) zbog toga što često znaju biti neispravni. To ćemo provjeriti putem 3D vizualizacije preko programskog sučelja *Processing* na eksperimentalnoj pločici koja se najčešće koristi za *Arduino* projekte. Prije same 3D vizualizacije, kalibrirat ćemo sustav zbog čestog pojavljivanja pogreške senzora uslijed nakupljanja malih vrijednosti izlaznih signala koji nemaju korelaciju s ulaznim signalom (engl. *drift*). Kalibraciju vršimo uz pomoć dvije metode koje ćemo navesti kasnije u radu, te ćemo izabrati metodu koja se prilikom eksperimentiranja pokaže najboljom.

Shematski sustav sklopa može se vidjeti na slici 8. Sklop reagira na tri ulazna signala, odnosno kutova zakreta koje mjeri senzor (žiroskop i akcelerometar). Nakon toga mikrokontroler obrađuje i modulira primljene signale za upravljanje aktuatorima.



Slika 8 Shematski prikaz rada sklopa

3. IZBOR KOMPONENATA

Odabrane komponente koje ćemo koristiti u izradi ovog projekta za samostabilizirajuću platformu s tri stupnja slobode su:

1. Za mikrokontroler koji će vršiti obrađivanje primljenih podataka odabran je *Atmega328P* mikrokontroler koji se nalazi na razvojnoj pločici *JOY-IT Atmega328 NANO* zbog dovoljne količine digitalnih ulaznih i izlaznih pinova, te prihvatljive cijene u odnosu na *Arduino NANO* prema čijem principu je napravljen.

2. Za određivanje kuteva zakreta, odnosno skretanje, poniranje i valjanje (*yaw*, *pitch* i *roll*) koristit će se *MEMS* modul *MPU-6050* koji ima funkciju žiroskopa i akcelerometra.

3. Servomotori koji će pomicati držače i platformu u tri ravnine. Koristit će se tri servomotora *TowerPro MG996R 180° Robotic* koji mogu vršiti zakrete do 180°.

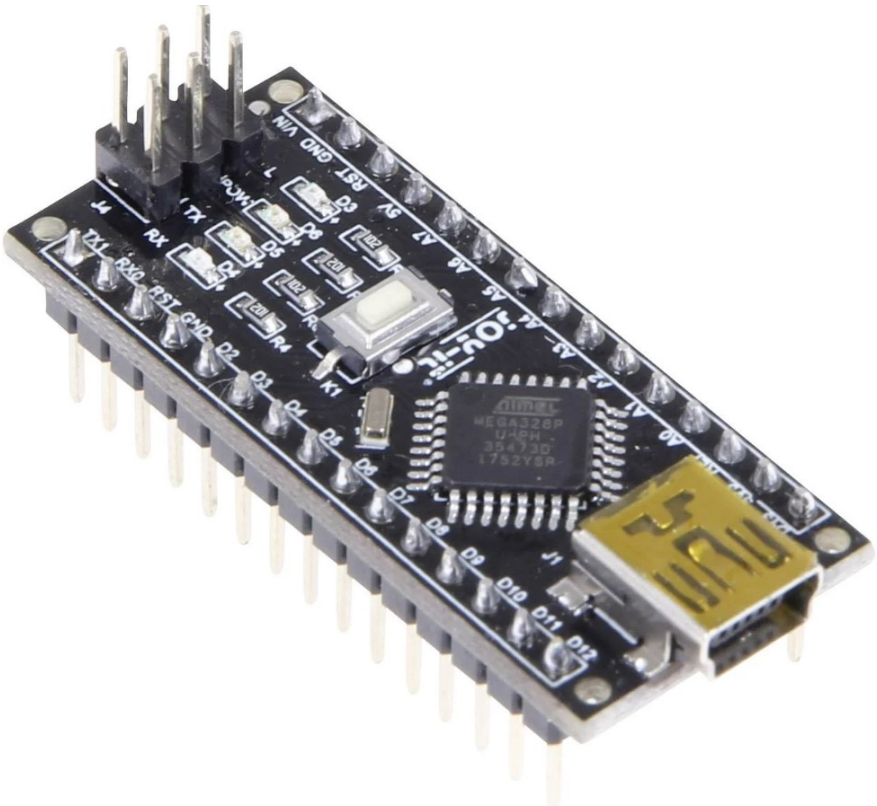
4. Pretvarač napona koji će smanjivati ulazni napon, odabran je LM2596S zbog prihvatljive cijene i malih dimenzija.

5. Kamera GoPro koja će služiti samo za videoprikaz eksperimenta.

6. Dvije litijeve baterije 18650 napona 3,7 *V* i 2200 *mAh* proizvođača *CAMELION BATTERIEN* i držač litijevih baterija.

7. Od ostalih komponenata koristit će se spojne žice koje su kompatibilne za *Arduino* projekte, konektori, eksperimentalna ploča i dr.

3.1. Razvojna pločica JOY-IT Atmega328P

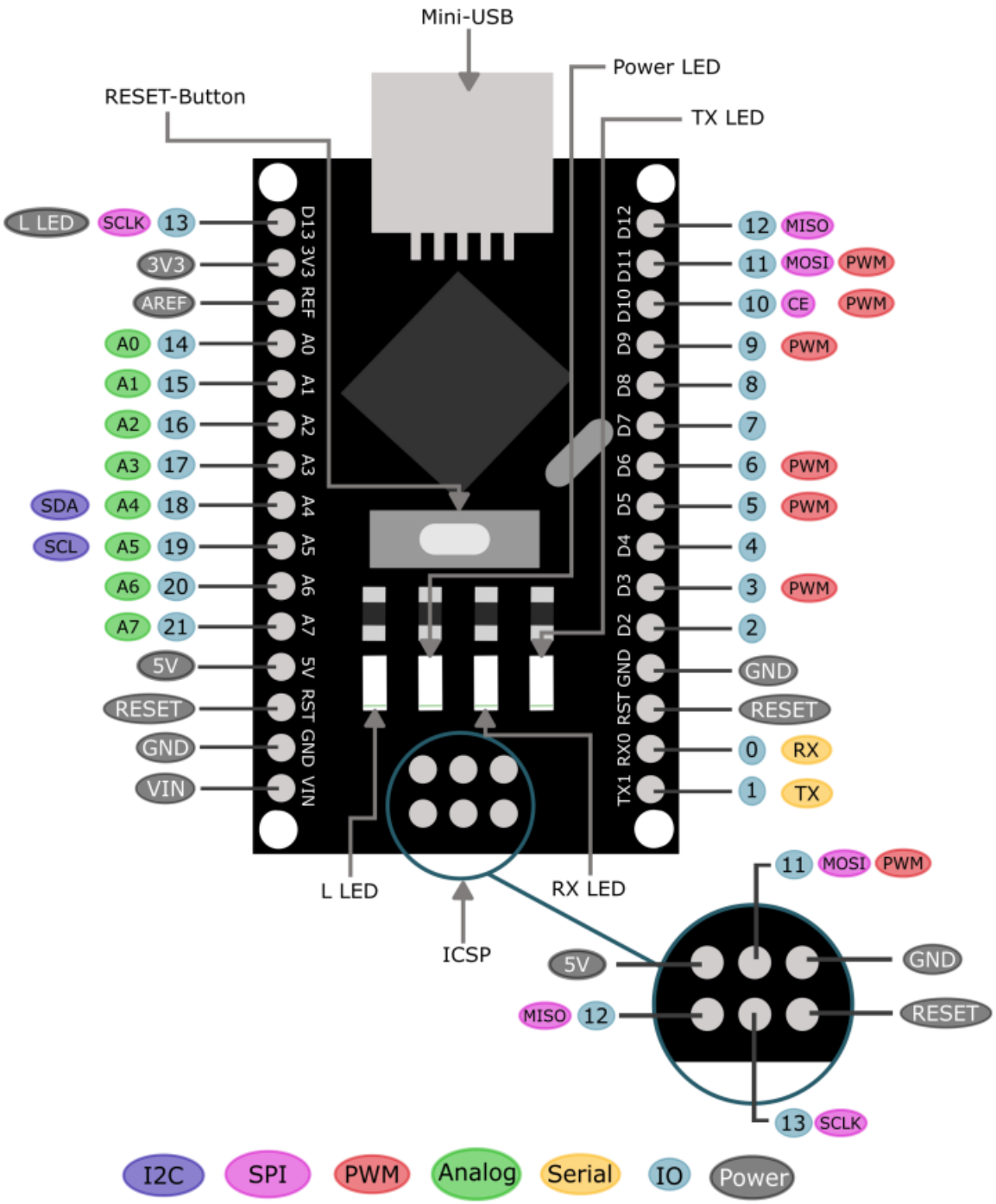


Slika 9 JOY-IT ATmega328P-AU razvojna ploča [10]

Razvojna ploča JOY-IT Atmega328P u sebi sadržava Atmega328P mikrokontroler koji je *Arduino* kompatibilan. Osim mikrokontrolera, sadržava 8 analognih i 22 digitalna pina (6 s *PWM*). Također, sadržava i integrirano USB sučelje za napajanje strujnog kruga i prijenos programskog koda. Programira se pomoću besplatnog i *open-source* programa *Arduino IDE* uz pomoć dodatno instaliranih *biblioteka* (eng. *library*) koje ćemo navesti kasnije. Jezik za programiranje baziran je na *C*++ jeziku [11].

Tablica 1 Razvojna pločica JOY-IT Atmega328P – tehničke specifikacije [11]

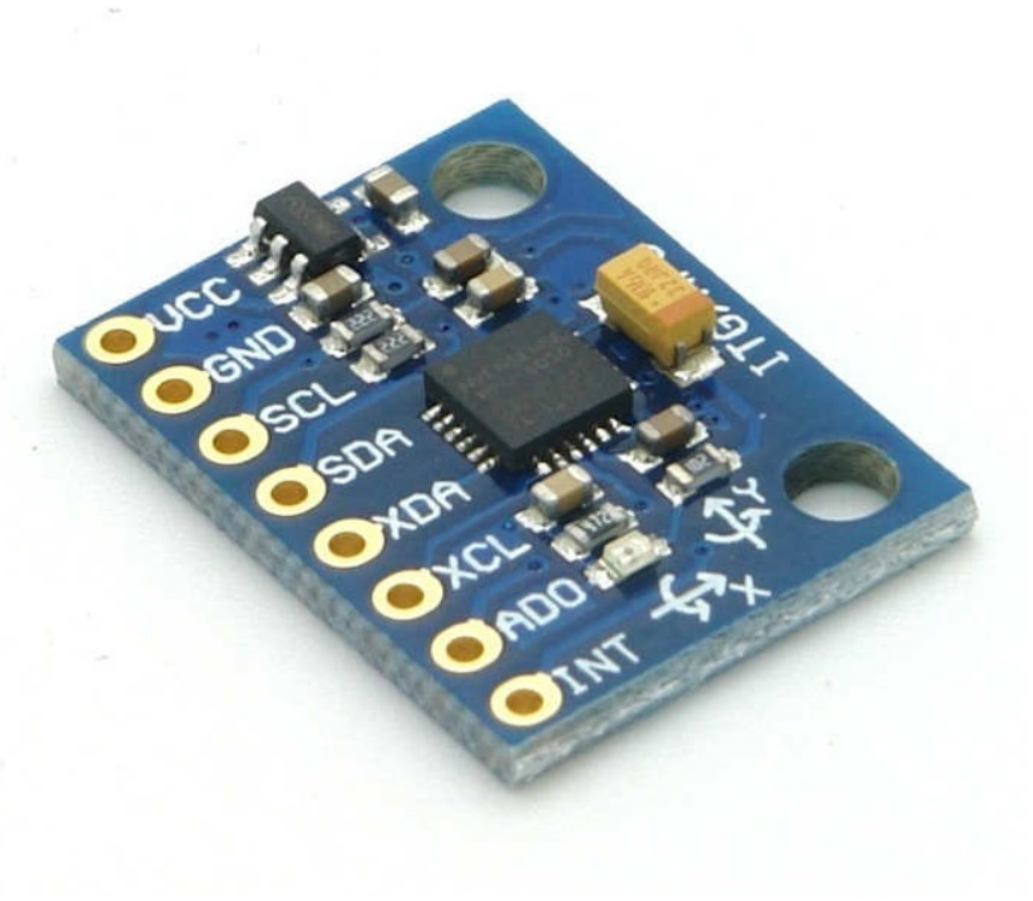
Mikrokontroler	Atmega328P-AU
Radni napon	5V
Flash memorija	32 kB (2 kB za Bootloader)
SRAM	2kB
Frekvencija takta	16 MHz
Analogni pinovi	8
Digitalni pinovi	22 (6 s PWM)
EEPROM	1 kB
DC struja po I/O pinu	40 mA
Ulazni napon	7-12 V
Potrošnja struje	19 mA
Dimenzije	18 x 45 mm
Težina	8 g



Slika 10 Shematski prikaz JOY-IT Atmega328p-AU razvojne pločice [11]

3.2. Žiroskop i akcelerometar MPU-6050

MPU-6050 (slika 11) je 6-osni MEMS integrirani uređaj za praćenje kretanja, određivanja pozicije i orijentacije. Uređaj kombinira 3-osni akcelerometar i 3-osni žiroskop kako bi eliminirao potrebu za korištenjem posebnih žiroskopa ili akcelerometara. Također, u sebi sadrži i DMP (eng. *Digital Motion Processor*) koji rasterećuje zahtjeve za intenzivnim računanjem obrade pokreta (eng. *Motion Processing*). To je, zapravo, ugrađeni procesor MPU-6050 modula koji kombinira podatke koji dolaze iz akcelerometra i žiroskopa kako bi olakšali, a samim time i točnije odredili poziciju i orijentaciju predmeta. Dodatno, MPU-6050 ima *VLOGIC* referentni pin (uz analogni pin *VDD* koji služi za logičko referentno i analogno napajanje uređaja) i podržava komunikaciju samo preko I^2C serijskog sučelja [12].

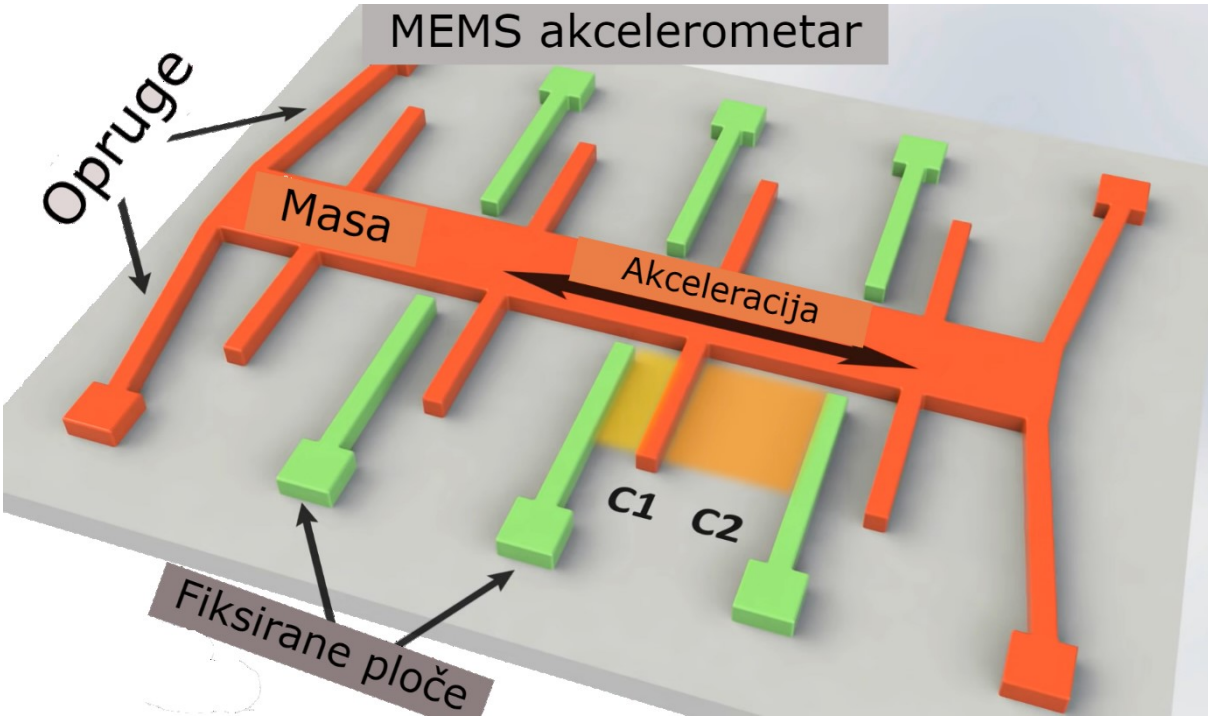


Slika 11 MPU-6050 žiroskop i akcelerometar [13]

3.2.1. MEMS senzori

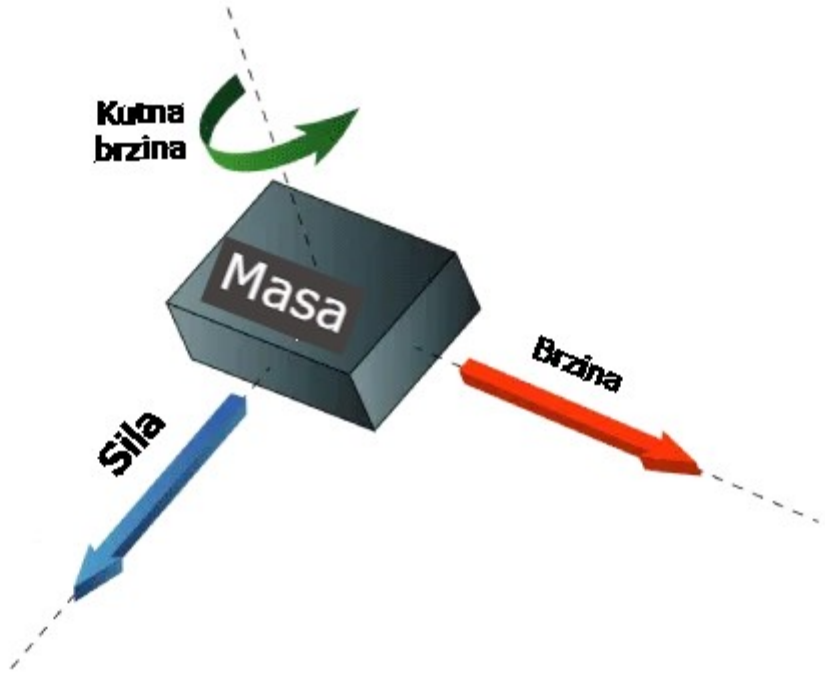
Kako bi pobliže objasnili kako MPU-6050 funkcionira, ukratko ćemo objasniti što su to zapravo MEMS senzori. MEMS (eng. *Micro Electro-Mechanical System*) su vrlo mali sustavi ili uređaji sastavljeni od mikro komponenti veličine 0,001 mm do 0,1 mm. Ove komponente izrađene su od silicija, polimera, metala i/ili keramike, a obično se kombiniraju s *CPU*-om (mikrokontrolerom) za kompletiranje sustava.

MEMS akcelerator mjeri ubrzanje mjerenjem promjene kapacitivnosti. Njegovu mikrostrukturu možemo zamisliti kao što je prikazano na slici 12. Ima masu pričvršćenu za oprugu koja je ograničena na kretanje u jednom smjeru, dok joj je vanjska ploča fiksirana. Dakle, kada nastupi ubrzanje u određenom smjeru, masa će se pomaknuti i kapacitet (C1 i C2) između ploča i mase će se promijeniti. Ova promjena u kapacitetu će se izmjeriti, obraditi i odgovarat će određenoj vrijednosti ubrzanja [14].



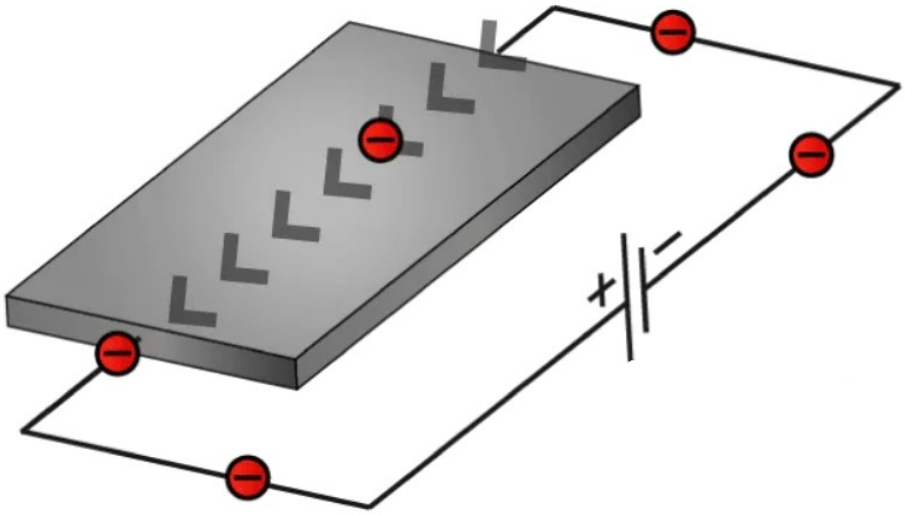
Slika 12 Prikaz rada MEMS akcelerometra [14]

MEMS žiroskop mjeri kutnu brzinu pomoću Coriolisovog efekta. Kada se masa kreće u određenom smjeru određenom brzinom i kada se primijeni vanjska kutna brzina kao što je prikazano na slici 13 zelenom strelicom, pojavit će se sila prikazana plavom strelicom, a crvenom strelicom je prikazana brzina koja će uzrokovati okomito pomicanje mase. Dakle, slično akcelerometru, ovaj pomak će uzrokovati promjenu kapacitivnosti koja će biti izmjerena, obrađena i odgovarat će određenoj kutnoj brzini [14].



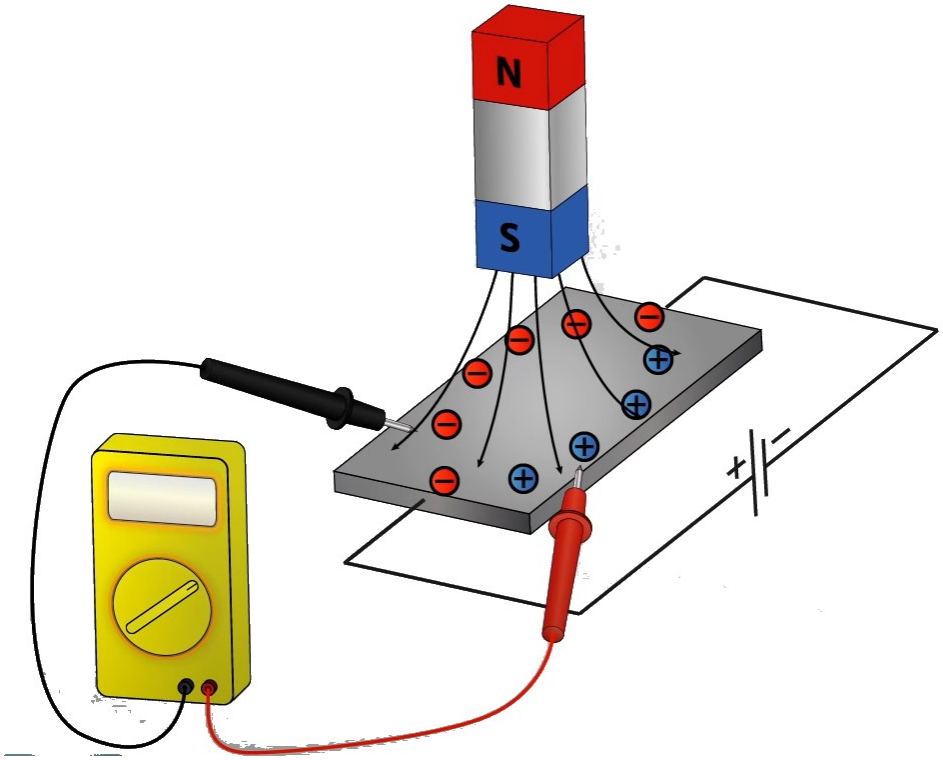
Slika 13 Prikaz rada MEMS žiroskopa [14]

MEMS magnetometar mjeri zemljino magnetsko polje koristeći *Hall*-ov učinak ili magnetski otporni učinak. Gotovo 90% senzora na tržištu koristi *Hallov* efekt i on funkcionira na sljedeći način. Ako imamo vodljivu ploču kao što je prikazana na slici 14 i postavimo da kroz nju teče struja, elektroni bi tekli ravno s jedne na drugu stranu ploče.



Slika 14 Vodljiva ploča

Ako dovedemo neko magnetsko polje blizu ploče, poremetili bismo ravnomjerni tok i elektroni bi skrenuli na jednu stranu ploče, a pozitivni polovi na drugu stranu ploče. To znači da ako stavimo voltmetar između ove dvije strane dobit ćemo neki napon koji ovisi o jačini magnetskog polja i njegovom smjeru (slika 15.). Ostalih 10% senzora na tržištu koristi magnetno-otporni učinak. Ovi senzori koriste materijale koji su osjetljivi na magnetsko polje, obično se sastoje od željeza (Fe) i nikla (Ne). Dakle, kad su ti materijali izloženi magnetskom polju, oni mijenjaju svoj otpor. U modulu MPU-6050 nije zastupljen magnetometar.



Slika 15 Mjerenje napona vodljive ploče pomoću voltmetra

Tablica 2 Raspon mjerenja i osjetljivosti akcelerometra i žiroskopa [12]

Žiroskop		Akcelerometar	
Raspon mjerenja [°/*sec*]	Osjetljivost [*LSB*/°/*sec*]	Raspon mjerenja [*g*]	Osjetljivost [*LSB*/*g*]
± 250	131	± 2	16384
± 500	65.5	± 4	8192
± 1000	32.8	± 8	4096
± 2000	16.4	± 16	2048

U tablici 2. navedeni su osnovni podaci za raspon mjerenja i osjetljivost za svaki od mogućih slučajeva žiroskopa i akcelerometra koje nam nudi MPU-6050. Broj okretaja po minuti [*rpm*] žiroskopa moguće je izračunati prema jednadžbi (1):

$$\frac{\boldsymbol{x}}{\mathbf{360°}} = \boldsymbol{\omega}\ [\mathbf{rad/s}] \qquad \textbf{(1)}$$

gdje je:

- x – raspon mjerenja
- ω – kutna brzina žiroskopa mjerena u radijanima po sekundi

Prema navedenoj jednadžbi, vrijednost kutne brzine bit će $\omega = 0.694$ rad/s. Gledajući logički, nikad nećemo dosegnuti takvu kutnu brzinu stabilizirajuće platforme, stoga množemo uzeti najveću osjetljivost kako bi dobili najbolja očitanja, a da pritom ne dođe do pogrešnih očitavanja. Istim načinom razmišljanja možemo doći i do izbora za osjetljivosti akcelerometra znajući da korištenjem našeg uređaja, nikad nećemo dostići veliku brzinu pomicanjem stabilizirajuće platforme od svoje ishodišne horizontalne ravnine. Koristeći jednadžbu (2), eksperimentalnim pokušajima nikad nismo dostigli vrijednost varijable „y“ da bude veća ili manja od ± 2 g, stoga smo i u slučaju akcelerometra uzeli najveću osjetljivost radi boljih očitavanja i detekcije pomicanja u ishodišnoj ravnini [15].

$$\frac{a}{b} = \pm y\ [\mathrm{g}] \qquad \textbf{(2)}$$

gdje je:

- a – očitavanje senzora pomakom od ishodišne horizontalne ravnine
- b – izabrana osjetljivost
- y – vrijednost raspona mjerenja akcelerometra

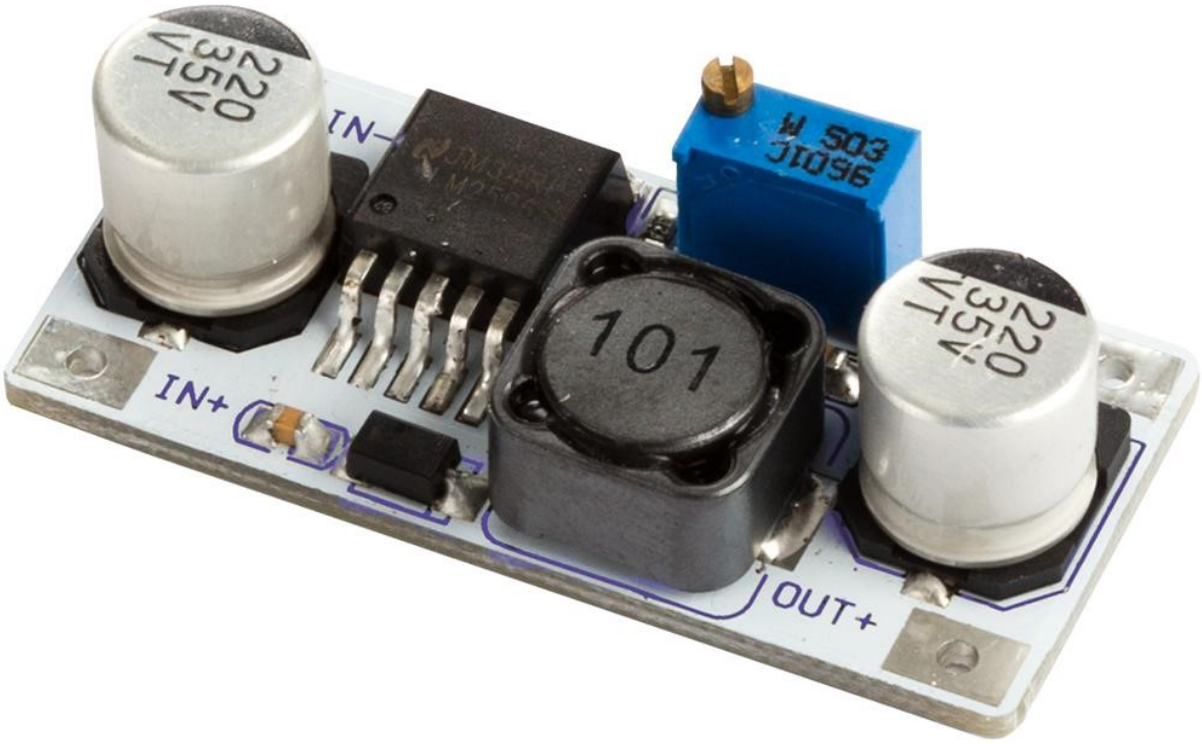
Tablica 3. pokazuje popis korištenih registara za rad senzora, te njihov kratki opis.

Tablica 3 Registri korišteni za rad MPU-6050 senzora

Registar	Oznaka	Opis
1B	GYRO_CONFIG	Podešavanje raspona mjerenja žiroskopa i pokretanje „self-testa“
1C	ACCEL_CONFIG	Podešavanje raspona mjerenja akcelerometra i pokretanje „self-testa“
3B, 3C. 3D, 3E, 3F, 40	ACCEL_OUT	Pohranjivanje izmjerenih vrijednosti akcelerometra u navedenim registrima
43, 44, 45, 46, 47, 48	GYRO_OUT	Pohranjivanje izmjerenih vrijednosti žiroskopa u navedenim registrima
6B	PWR_MGMT_1	Registar koji omogućava uključenje uređaja, podešavanje oscilatora ili resetiranje cijelog senzora

3.3. Podesivi DC-DC step-down pretvarač napona LM2596S

Step-down pretvarač napona LM2596S je DC-DC konverter (eng. *Buck converter*) koji snižava ulazni napon (napajanje) na izlazu (opterećenje). Imaju široku primjenu u niskonaponskim izvedbama kao što su *Arduino* projekti. U našem slučaju, pretvara napon od 7,4 V na 5 V koji se podešava putem trimera na pločici. Vrlo je praktičan jer izlazni napon uvijek ostaje isti i malih je dimenzija [16].



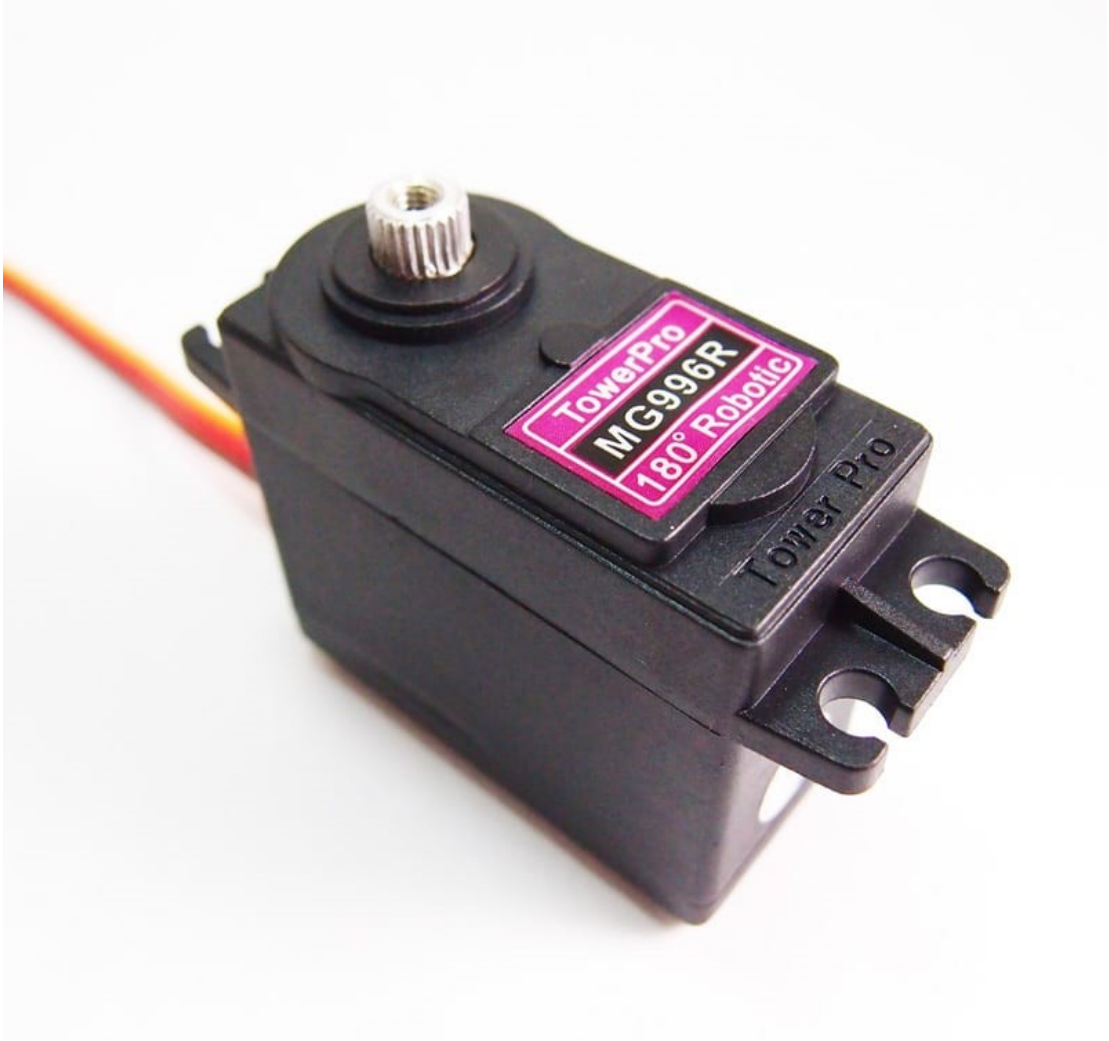
Slika 16 Podesivi DC-DC step-down modul LM2596S [16]

Tablica 4 Tehnički podaci podesivog DC-DC step-down modula LM2596S

Ulazni napon	3 do 40 V DC
Izlazni napon	1,25 do 35 V DC
Maks. izlazna struja	2,5 A
Čip	LM2596S
Dimenzije	49 x 26 x 12 mm

3.4. Servomotor MG996R TowerPro 180° Robotic

Motor oznake MG996R TowerPro (slika 13) je servomotor s metalnim zupčanikom i s maksimalnim zakretnim momentom od 11 kg/cm. Motor se okreće od 0° do 180° na temelju radnog ciklusa *PWM*-a koji se dovodi na njegov signalni pin [17]. Često je korišten u *Arduino* projektima zbog svoje male težine i malih dimenzija. Njegove karakteristike prikazane su u tablici 5.



Slika 17 Servomotor MG996R TowerPro 180° Robotic [18]

Tablica 5 Karakteristike MG996R TowerPro 180° Rotation Servomotora [19]

Težina	55 g
Dimenzije	40,7 x 19,7 x 42,9
Radni napon	4,8 – 6,6 V
Okretni moment	9,4 kg/cm (4,8 V); 11 kg/cm (6,0 V)
Brzina	0,19 sec / 60° (4,8 V); 0,15 sec / 60° (6,0 V)
Vrsta zupčanika	metalni zupčanik

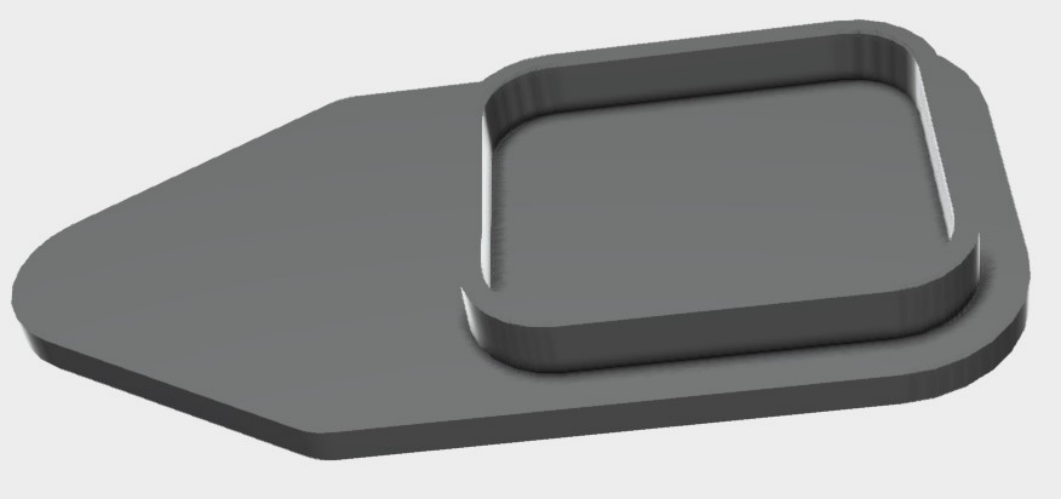
4. PROJEKTIRANJE I ISPITIVANJE RADA SUSTAVA

4.1. 3D modeliranje konstrukcije

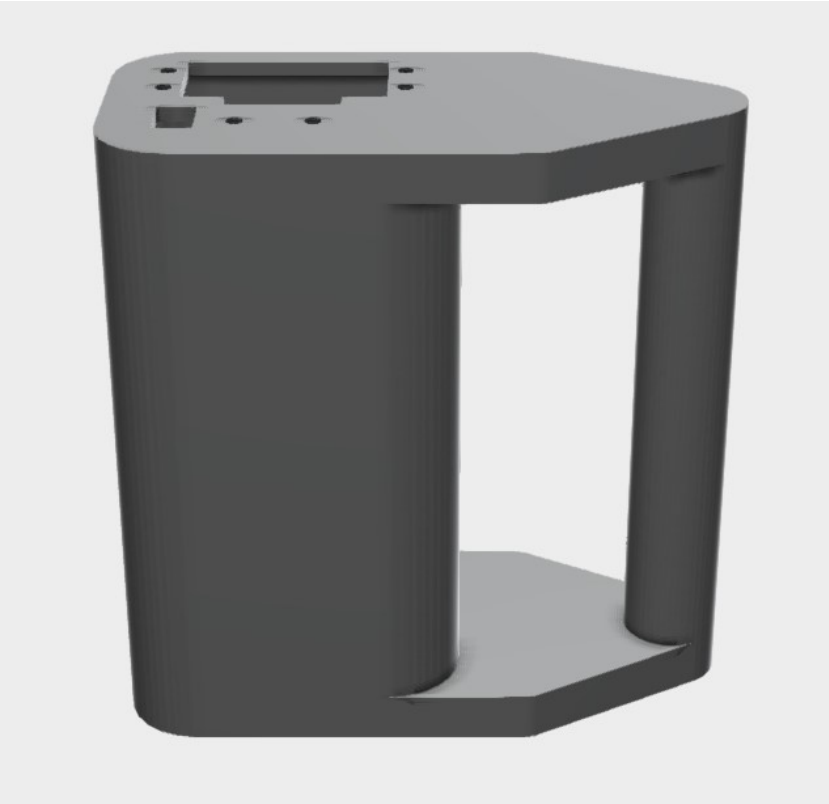
Modeliranje sklopa troosne samostabilizirajuće platforme napravljeno je u programu AutoDesk Inventor 2022. Model je konstruiran tako da bude manjih dimenzija nego uobičajeni gimbal stabilizatori. Svaka komponenta je napravljena 3D printanjem. Sklop se sastoji od:

- podnožja (slika 18)
- kućišta s drškom (slika 19)
- 2 držača servomotora (slika 20)
- uške za držanje platforme (slika 21)
- platforma (slika 22)

Model je konstruiran tako da bude manjih dimenzija nego uobičajeni gimbal stabilizatori. Kućište je kvadratnog oblika s drškom radi lakšeg manevriranja gimbala. Iznutra je šuplje kako bi stale sve potrebne komponente. Osim toga, na površini kućišta napravljeni su otvori kako bi se mogli postaviti MPU-6050, servomotor i prekidač. Držači servomotora napravljeni su pravokutnog oblika s otvorima dovoljno velikim kako bi se mogli ugraditi preostala dva servomotora. Platforma je pravokutnog oblika dovoljno velika za kamere manjih dimenzija poput kamere *GoPro* koja će biti prikazana u videozapisu ovog rada. Spojevi između zupčanika servomotora i držača motora drugih osi spojeni su plastičnim prirubnicama koje se dobiju u paketu uz servomotor s vijcima. Ostale komponente koje je bilo potrebno vijčano spojiti, uzeti su M3 vijci i matice. Cijeli model sklopa prikazan je na slici 23, a na slici 24 prikazan je sklop nakon 3D printanja.

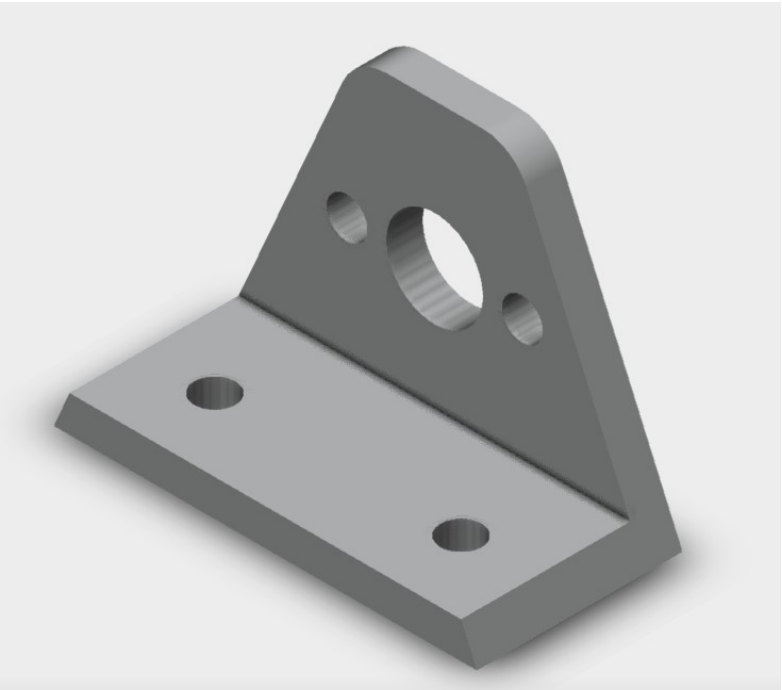


Slika 18 Podnožje



Slika 19 Kućište s drškom

Slika 20 Držač servomotora



Slika 21 Držač platforme

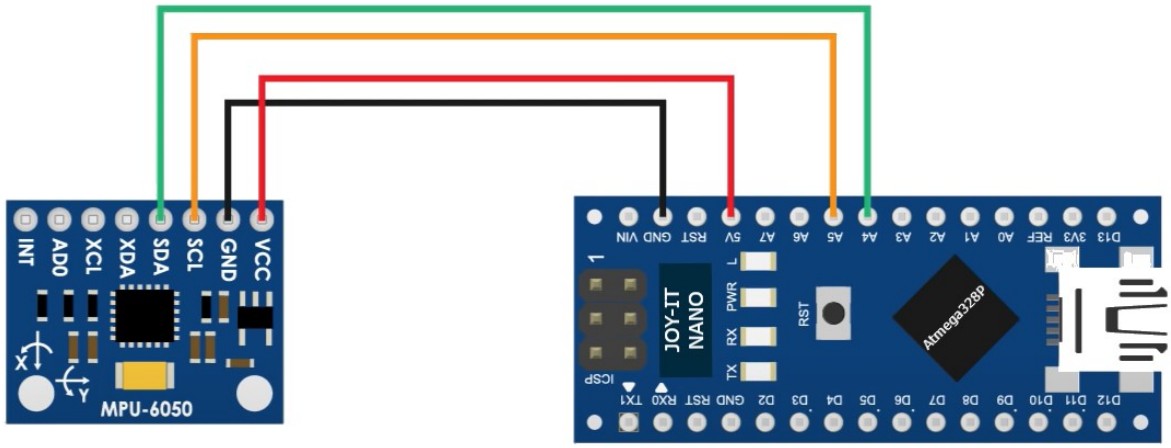
Slika 22 Platforma

Slika 23 Model sklopa troosne samostabilizirajuće platforme

Slika 24 Konstrukcija troosne samostabilizirajuće platforme

4.2. Metoda ručne kalibracije MPU-6050 modula

Kako bi žiroskop i akcelerometar MPU-6050 radio ispravno, kao i većina senzora, prije prve uporabe moramo ga kalibrirati. Ono što želimo napraviti je ukloniti nultu pogrešku. To se odnosi na slučaj kada senzor bilježi mali kut iako je potpuno ravan, odnosno odložen na ravnoj površini. Ova se pogreška može ukloniti primjenom pomaka na neobrađena očitanja senzora akcelerometra i žiroskopa. Pomak treba podešavati dok očitanja žiroskopa ne budu u nuli (bez rotacije), a akcelerometar bilježi ubrzanje zbog gravitacije usmjerene izravno prema dolje. Spajajući modul MPU-6050 s razvojnom pločicom kao što je prikazano na slici 25, koristeći I^2C komunikacijski protokol pokrećemo kod naveden u prilogu pod „METODA RUČNE KALIBRACIJE“. Prvo moramo definirati biblioteku „*Wire.h*“ koja se koristi za I^2C komunikacijski protokol i definirati neke varijable potrebne za pohranu podataka navedene u algoritmu 1.



Slika 25 Shematski prikaz spajanja pri ručnoj kalibraciji MPU-6050 [20]

```
#include <Wire.h>
const int MPU = 0x68; // MPU6050 I2C adresa
float AccX, AccY, AccZ;
float GyroX, GyroY, GyroZ;
float accAngleX, accAngleY, gyroAngleX, gyroAngleY, gyroAngleZ;
float roll, pitch, yaw;
float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;
float elapsedTime, currentTime, previousTime;
int c = 0;
```

Algoritam 1. Definiranje varijabli

U odjeljku za postavljanje (eng. *void setup*), trebamo inicijalizirati „*Wire.h*“ *library* i resetirati senzor kroz registar upravljanja napajanjem. Da bismo to učinili, moramo pogledati tablicu registara senzora navedenu u tablici 3 gdje možemo vidjeti adresu registra.

```
void setup() {
  Serial.begin(19200);          //19200 brzina ispisa (engl. baud rate)
  Wire.begin();                 //inicijalizacija komunikacije
  Wire.beginTransmission(MPU);  //započni komunikaciju sa MPU-6050
  Wire.write(0x6B);             //započni komunikaciju sa registrom 6B
  Wire.write(0x00);          //reset registra (staviti 0 u 6B registar)
  Wire.endTransmission(true);   //završetak komunikacije
```

Algoritam 2. Inicijalizacija

Također, ako želimo, možemo koristiti određene raspone mjerenja i osjetljivosti koje su navedene u tablici 2, ali kako nama odgovara zadani raspon mjerenja i osjetljivosti koji iznosi ±2 *g* za akcelerometar i ±250 °/*sec* za žiroskop, taj dio koda ostavit ćemo komentiranim. U odjeljku petlje (eng. *loop section*) počinjemo s očitavanjem podataka akcelerometra. Podaci za svaku os pohranjeni su u dva bajta ili registra, a adrese tih registara možemo vidjeti iz tablice 2. Kako bismo ih sve pročitali, počinjemo s prvim registrom, a pomoću „*requestFrom()*“ funkcije zahtijevamo čitanje svih 6 registara za X, Y i Z osi. Zatim čitamo podatke iz svakog registra, a budući da su izlazi komplementarni, kombiniramo ih na odgovarajući način kako bismo dobili točne vrijednosti.

```
1 void loop() {
2
3   Wire.beginTransmission(MPU);
4   Wire.write(0x3B);        // započni sa registrom 0x3B (ACCEL_XOUT_H)
5   Wire.endTransmission(false);
6   Wire.requestFrom(MPU, 6, true); //pročitaj sljedećih 6 registara
```

Algoritam 3. Upravljanje registrima

Kako bismo dobili izlazne vrijednosti od -1 *g* do +1 *g*, pogodne za izračun kutova, dijelimo izlaz s prethodno odabranom osjetljivošću navedenoj u tablici 2. što je prikazano u algoritmu 4.

```
1   AccX = (Wire.read() << 8 | Wire.read()) / 16384.0; //vrijednost X-osi
2   AccY = (Wire.read() << 8 | Wire.read()) / 16384.0; //vrijednost Y-osi
3   AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0; //vrijednost Z-osi
```

Algoritam 4. Čitanje vrijednosti akcelerometra

Koristeći navedene jednadžbe (3) i (4) [21], izračunavamo *roll* i *pitch* kutove iz podataka koje nam je dao akcelerometar. Jednadžba iz Arduino koda navedena je pod algoritmom 5.

$$tan\,\phi_{xyz} = \left(\frac{G_{py}}{G_{pz}}\right) \tag{3}$$

$$tan\,\theta_{xyz} = \left(\frac{-G_{px}}{G_{py}sin\,\phi + G_{pz}cos\,\phi}\right) = \frac{-G_{px}}{\sqrt{{G_{py}}^2 + {G_{pz}}^2}} \quad \textbf{(4)}$$

gdje je:

- ϕ_{xyz} – kut *roll* rotacije
- θ_{xyz} – kut *pitch* rotacije
- G_{px}, G_{py}, G_{pz} – izlazne vrijednosti sustava za x,y i z osi

```
1  accAngleX = (atan(AccY / AccZ) * 180 / PI) - 0.58;
2  accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) *  180
   / PI) + 1.58;
```

Algoritam 5. Jednadžbe za izračunavanje vrijednosti akcelerometra

Dodane vrijednosti u algoritmu 5. kao što su -0.58 za varijablu „a*ccAngleX*“ i +1.58 za varijablu „*accAngleY*“ dobili smo pozivajući funkciju „*calculate_IMU_error()*“ navedenoj u prilogu pod programom „RUČNA KALIBRACIJA“ (algoritam 6.), dok je senzor u mirnom položaju. Ovdje radimo 200 očitanja za sve izlaze, zbrajamo ih i dijelimo sa 200. Budući da senzor držimo u ravnom položaju, očekivane izlazne vrijednosti trebale bi biti jednake nuli. Dakle, ovim izračunom možemo dobiti prosječnu pogrešku senzora koje naknadno uvrštavamo u algoritam 5.

```
1 void calculate_IMU_error() {
2
3   while (c < 200) {
4     Wire.beginTransmission(MPU);
5     Wire.write(0x3B);
6     Wire.endTransmission(false);
7     Wire.requestFrom(MPU, 6, true);
8     AccX = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
9     AccY = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
10    AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
11
12    AccErrorX = AccErrorX + (atan(AccY / AccZ) * 180 / PI)
13 +pow((AccZ), 2))) * 180 / PI));
14    AccErrorY = AccErrorY + ((atan(-1 * (AccX) / sqrt(pow((AccY), 2) +
15 pow((AccZ), 2))) * 180 / PI));
16    c++;
17  }
18
19  AccErrorX = AccErrorX / 200;
20  AccErrorY = AccErrorY / 200;
21  c = 0;
22
23  while (c < 200) {
24    Wire.beginTransmission(MPU);
25    Wire.write(0x43);
26    Wire.endTransmission(false);
27    Wire.requestFrom(MPU, 6, true);
28    GyroX = Wire.read() << 8 | Wire.read();
29    GyroY = Wire.read() << 8 | Wire.read();
30    GyroZ = Wire.read() << 8 | Wire.read();
31
32    GyroErrorX = GyroErrorX + (GyroX / 131.0);
33    GyroErrorY = GyroErrorY + (GyroY / 131.0);
34    GyroErrorZ = GyroErrorZ + (GyroZ / 131.0);
35    c++;
36  }
37
38  GyroErrorX = GyroErrorX / 200;
39  GyroErrorY = GyroErrorY / 200;
40  GyroErrorZ = GyroErrorZ / 200;
41
42  Serial.print("AccErrorX: ");
43  Serial.println(AccErrorX);
44  Serial.print("AccErrorY: ");
45  Serial.println(AccErrorY);
46  Serial.print("GyroErrorX: ");
47  Serial.println(GyroErrorX);
48  Serial.print("GyroErrorY: ");
49  Serial.println(GyroErrorY);
50  Serial.print("GyroErrorZ: ");
    Serial.println(GyroErrorZ);
  }
```

Algoritam 6. Funkcija „*calculate_IMU_error*“ [20]

Slično kao za akcelerometar, radimo iste postupke i za žiroskop. Iz tablice 3 pozivamo potrebne registre, iščitavamo podatke, te ih dijelimo s prethodno odabranom osjetljivošću koja je u našem slučaju zadana vrijednost 131 LSB/° koja odgovara rasponu mjerenja od ± 250 °/sec (algoritam 7.)

```
GyroX = (Wire.read() << 8 | Wire.read()) / 131.0;
GyroY = (Wire.read() << 8 | Wire.read()) / 131.0;
GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0;
```

Algoritam 7. Čitanje vrijednosti žiroskopa

Isto kao i kod akcelerometra, ispravljamo izlazne vrijednosti s malim izračunatim vrijednostima koje smo dobili iz funkcije „*calculate_IMU_error()*“ (algoritam 6.). Kako su izlazne vrijednosti u stupnjevima po sekundi, trebamo ih pomnožiti s vremenom da dobijemo samo stupnjeve (algoritam 7.).

```
GyroX = GyroX + 0.56; // GyroErrorX ~(-0.56)
GyroY = GyroY - 2; // GyroErrorY ~(2)
GyroZ = GyroZ + 0.79; // GyroErrorZ ~ (-0.8)

gyroAngleX = gyroAngleX + GyroX * elapsedTime; // deg/s * s = deg
gyroAngleY = gyroAngleY + GyroY * elapsedTime;
yaw =  yaw + GyroZ * elapsedTime;
```

Algoritam 8. Jednadžbe za izračunavanje vrijednosti žiroskopa

Na kraju spajamo podatke akcelerometra i žiroskopa pomoću komplementarnog filtra. Ovdje uzimamo 96% podataka žiroskopa jer su vrlo precizni i ne trpe vanjske sile. Loša strana žiroskopa je to što se kreće ili unosi pogrešku u izlaz kako vrijeme prolazi. Stoga, dugoročno gledano, koristimo podatke iz akcelerometra, 4% u ovom slučaju, što je dovoljno da se eliminira pogreška odstupanja žiroskopa (algoritam 9.). Međutim, kako ne možemo izračunati *yaw* rotaciju iz podataka akcelerometra, ne možemo na njega implementirati komplementarni filtar.

```
gyroAngleX = 0.96 * gyroAngleX + 0.04 * accAngleX;
gyroAngleY = 0.96 * gyroAngleY + 0.04 * accAngleY;
```

Algoritam 9. Komplementarni filtar

Nakon što smo odradili prethodne korake, ispisujemo vrijednosti na serijskom monitoru *Arduino IDE* sučelja pomoću naredbe „*Serial.print*“ koji su prikazani na slici 26 kako bi ustanovili da li naš sustav radi ispravno. Možemo uočiti kako nam vrijednost *yaw* rotacije s vremenom sve više odstupa jer se za njega ne može koristiti komplementarni filtar. To je obično magnetometar koji se može koristiti kao dugoročna korekcija za pomicanje *yaw* rotacije akcelerometra. Isto tako, 3D vizualizacijom u programskom sučelju *Processing* može se primjetiti nedostatak magnetometra. No, MPU-6050 ima značajku koja se zove DMP (eng. *Digital Motion Processor*) koju ćemo opisati u sljedećem poglavlju.

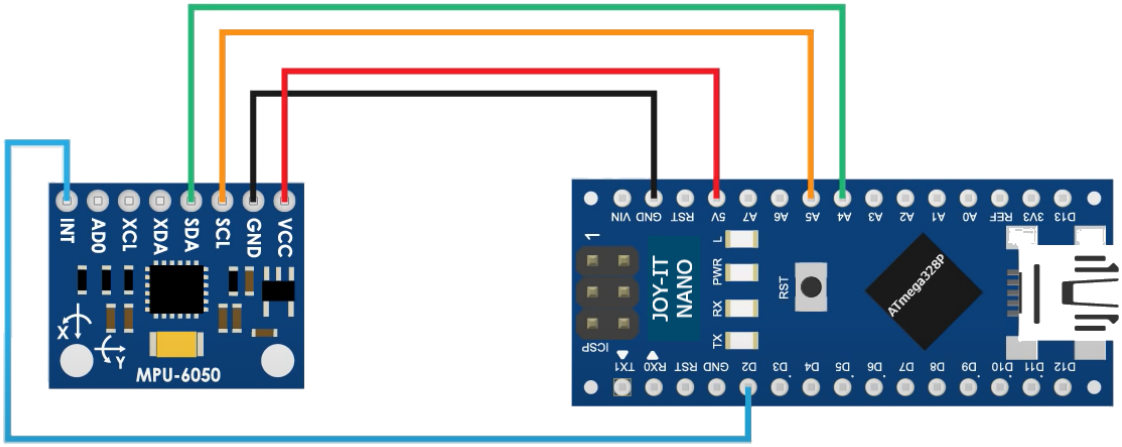
```
ypr  16.60/36.29/-30.24
ypr  16.62/36.27/-30.24
ypr  16.65/36.26/-30.23
ypr  16.67/36.25/-30.22
ypr  16.69/36.25/-30.22
ypr  16.72/36.25/-30.22
ypr  16.74/36.26/-30.23
ypr  16.76/36.28/-30.23
ypr  16.79/36.29/-30.23
ypr  16.81/36.29/-30.24
ypr  16.83/36.29/-30.24
ypr  16.86/36.29/-30.22
ypr  16.88/36.28/-30.21
ypr  16.90/36.27/-30.20
ypr  16.93/36.27/-30.20
ypr  16.95/36.28/-30.21
ypr  16.98/36.28/-30.20
ypr  17.00/36.27/-30.20
ypr  17.02/36.27/-30.20
ypr  17.05/36.26/-30.19
ypr  17.07/36.25/-30.18
ypr  17.09/36.25/-30.18
ypr  17.12/36.25/-30.18
ypr  17.14/36.26/-30.17
ypr  17.16/36.26/-30.18
ypr  17.19/36.26/-30.18
ypr  17.21/36.26/-30.18
ypr  17.23/36.25/-30.19
ypr  17.26/36.27/-30.19
ypr  17.28/36.25/-30.19
ypr  17.30/36.27/-30.20
```

Slika 26 Ispis podataka sa Serial monitora za ručnu kalibraciju

4.3. Kalibracija MPU-6050 modula koristeći DMP procesor

MPU6050 modul sadrži DMP (eng. *Digital Motion Processor*) koji spaja podatke akcelerometra i žiroskopa kako bi minimizirao učinke pogrešaka svojstvenih svakom senzoru. DMP izračunava rezultate u smislu kvaterniona i može pretvoriti rezultate u *Euler*-ove kutove, te također izvoditi druga izračunavanja s podacima.

Za kalibraciju MPU-6050 modula koristili smo *library* za primjenu pokreta „*MPU6050.h*“ i za I^2C serijski komunikacijski protokol „*I2Cdev.h*“ i „*Wire.h*“ [20][21]. Na eksperimentalnu ploču spojili smo razvojnu ploču Atmega328P s modulom MPU-6050 kako je prikazano na slici 27 na ravnu horizontalnu površinu kako ne bi došlo do pogrešnog kalibriranja.



Slika 27 Shematski prikaz spajanja pri kalibraciji s DMP značajkom

Nakon što smo spojili uređaje i priključili na napajanje preko USB priključka, pokrenuli smo kod koji je naveden u prilogu pod nazivom „KALIBRACIJA MODULA MPU-6050“. Kod funkcionira na principu PI (proporcionalno-integralnog) regulatora. Kad promijenimo pomak u MPU-6050 modulu, možemo dobiti trenutne rezultate. To nam omogućuje da koristimo proporcionalno i integralno djelovanje PI regulatora za smanjenje iznosa odstupanja. Integralno djelovanje koristi pogrešku zadane vrijednosti (zadana vrijednost je nula), uzima djelić te pogreške i dodaje je na integralnu vrijednost. Svako očitavanje smanjuje pogrešku u odnosu na

željeni pomak. Što je veća pogreška u odnosu na zadanu vrijednost, to više podešavamo integralnu komponentu. Proporcionalno djelovanje smanjuje šum nastao uslijed integralnog djelovanja. Na kraju svakog seta od 100 očitanja, integralna vrijednost koristi se za stvarne pomake, a posljednje proporcionalno čitanje se zanemaruje zbog činjenice da reagira na svaki šum. Nakon 1000 očitanja, dobit ćemo rezultat kao što je prikazano na slici 28.

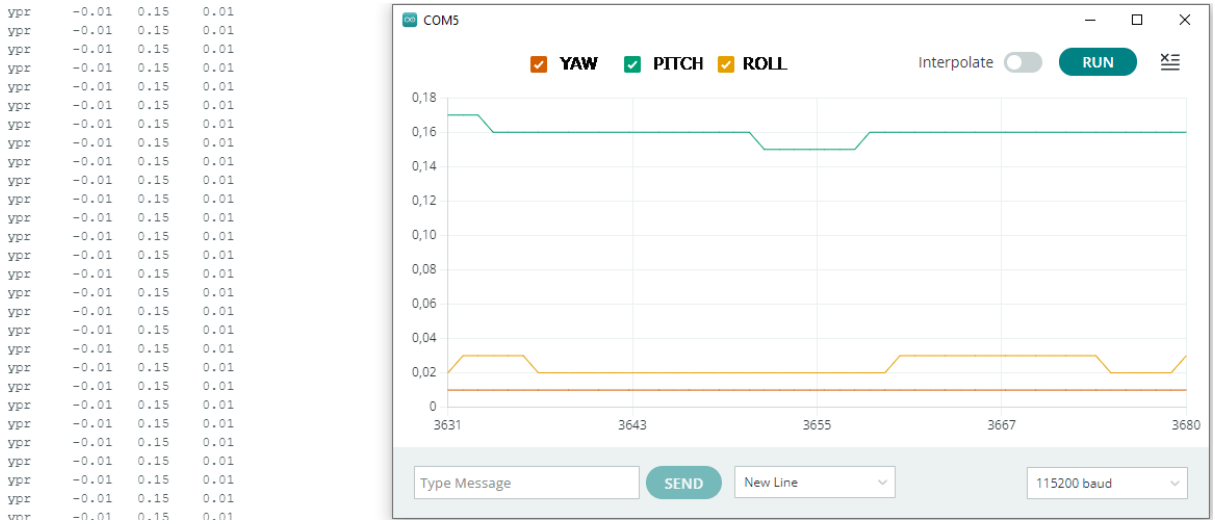
```
inicijalizacija uredaja...
Testiranje konekcije uredaja...
MPU6050 uspjesna konekcija
PID podesavanje tocaka = 100 ocitanja
>......>......
 600 ocitanja
926.00000,      430.00000,      831.00000,      -36.00000,      -34.00000,      -35.00000


>.>.700 ocitanja
926.00000,      430.00000,      831.00000,      -36.00000,      -34.00000,      -35.00000


>.>.800 ocitanja
926.00000,      430.00000,      831.00000,      -35.00000,      -34.00000,      -36.00000


>.>.900 ocitanja
926.00000,      430.00000,      831.00000,      -35.00000,      -34.00000,      -35.00000


>.>.1000 ocitanja
926.00000,      430.00000,      831.00000,      -35.00000,      -33.00000,      -36.00000
```

Slika 28 Prikaz rezultata programa „KALIBRACIJA MODULA MPU-6050“

Nakon tisućitog očitanja, dobijemo konačnu kalibraciju MPU-6050 modula. Svaki stupac odgovara sljedećem:

1. stupac – određivanje odstupanja akcelerometra po X osi
2. stupac – određivanje odstupanja akcelerometra po Y osi
3. stupac – određivanje odstupanja akcelerometra po Z osi
4. stupac – određivanje odstupanja žiroskopa po X osi
5. stupac – određivanje odstupanja žiroskopa po Y osi
6. stupac – određivanje odstupanja žiroskopa po Z osi

U našem slučaju, zanimaju nas odstupanja žiroskopa po X, Y i Z osi i odstupanje akcelerometra po Z osi za 3D vizualizaciju. Na slici 29 prikazan je odziv nakon kalibracije gdje se mogu vidjeti minimalna odstupanja, zanemarujući vanjske utjecaje i horizontalnu podlogu koja nije idealno ravna.



Slika 29 Rezultat kalibracije

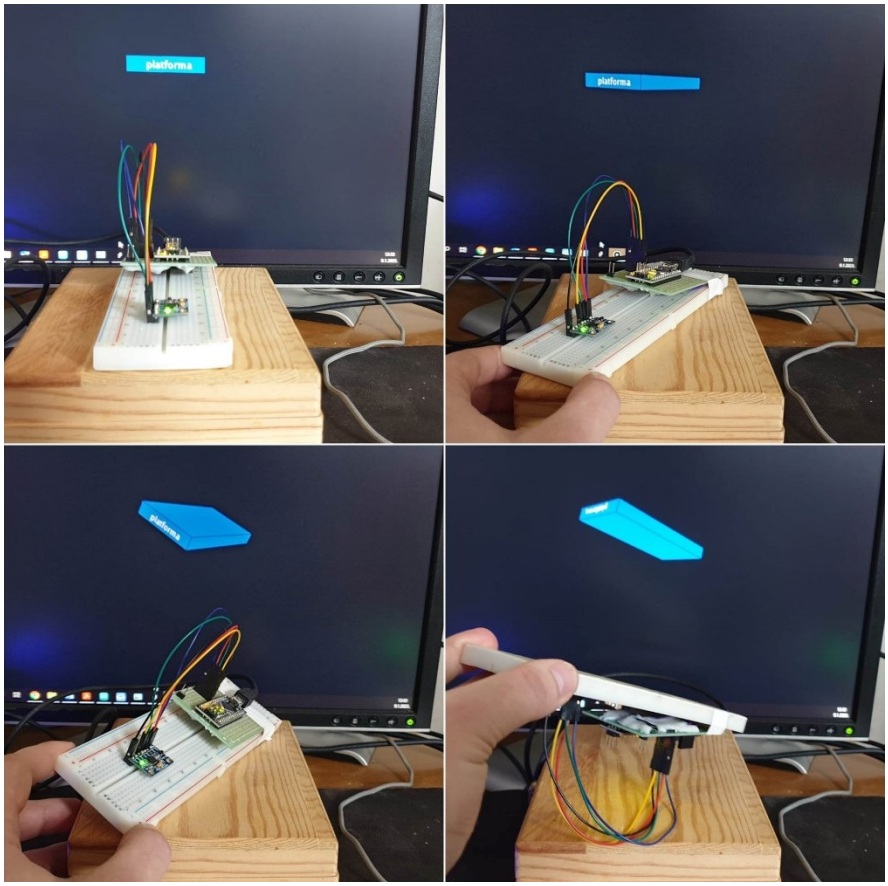
4.4. 3D vizualizacija MPU-6050 modula

Kako bi se najbolje mogla prikazati ispravnost MPU-6050 modula, izrađen je program za 3D vizualizaciju sustava. Program je načinjen u programskom sučelju *Processing* i u *Java* programskom jeziku. Uz pomoć jednog od osnovnih *library*-ija za serijsku komunikaciju „*processing.serial*“ i *library*-ija „*toxi.geom*“ (algoritam 10.) i DMP-a, računamo i ispisujemo kvaternione koji se koriste za predstavljanje orijentacije i rotacije objekata u tri dimenzije. U ovom primjeru koristimo kvaternione za predstavljanje orijentacije i rotacije u odnosu na *Euler*-ove kutove kao u prijašnjem primjeru kalibracije. Nakon toga, dobivamo 3D prikaz kvadra koji reprezentira dimenzije eksperimentalne ploče na koju smo stavili MPU-6050 modul i razvojnu ploču pri izvođenju eksperimenta [21][22][23]. Kod za 3D vizualizaciju može se naći u prilogu pod algoritmom „3D_VIZUALIZACIJA“.

```
1  import processing.serial.*;
2
3  import toxi.geom.*;
4
5
6  Serial port;
7  char[] THREEDVISUALISATION = new char[14];
8  int serialCount = 0;
9
10 int interval = 0;
11
12 float[] q = new float[4];
13 Quaternion quat = new Quaternion(1, 0, 0, 0);
```

Algoritam 10. Uvođenje biblioteka i klase za izračunavanje kvaternion

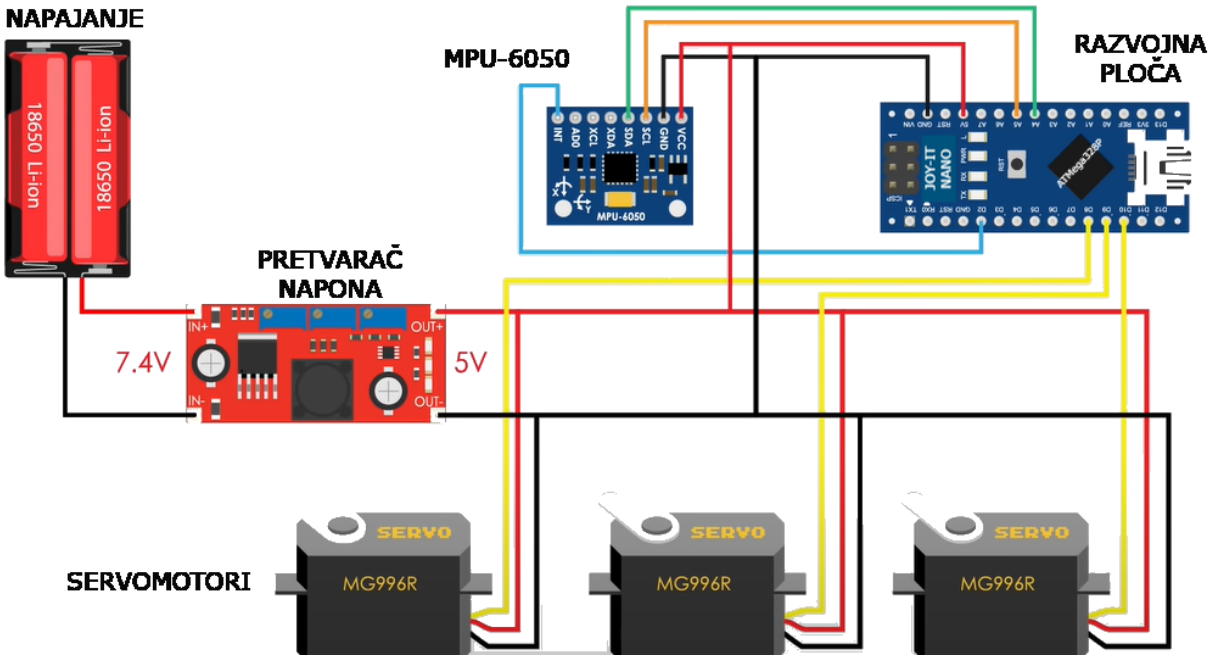
Kao što se može vidjeti iz slike 30, 3D vizualizacija kvadratne platforme radi s minimalnim pogreškama. Korištenjem DMP-a, uz žiroskop i akcelerometar, ispostavilo se puno pouzdanijim i boljim rješenjem nego bez DMP-a jer smo se riješili prekovremenog driftanja *yaw* rotacije koju smo primjećivali prilikom ručne kalibracije i 3D prikaza.



Slika 30 3D vizualizacija MPU-6050 modula

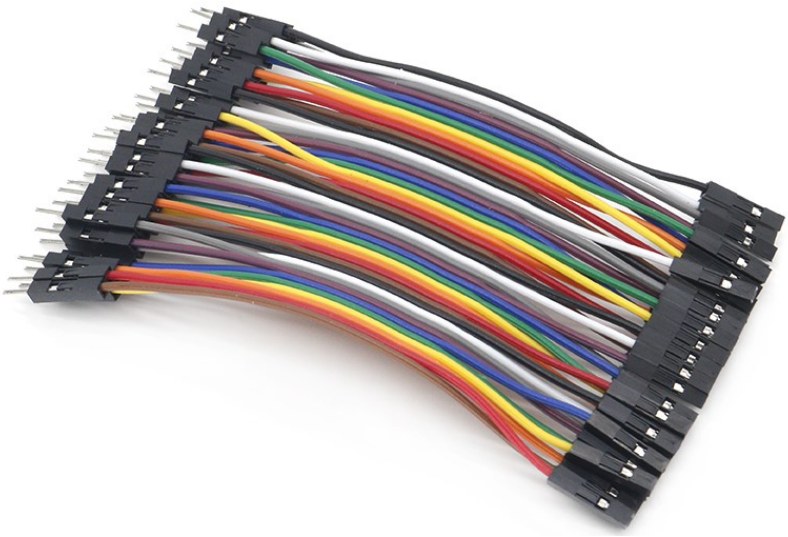
5. IZRADA SUSTAVA

Nakon programiranja sustava i uspješne eksperimentalne analize, možemo započeti s konačnom izradom sustava. Odabrane komponente za izradu troosne samostabilizirajuće platforme spojit ćemo prema slici 31.



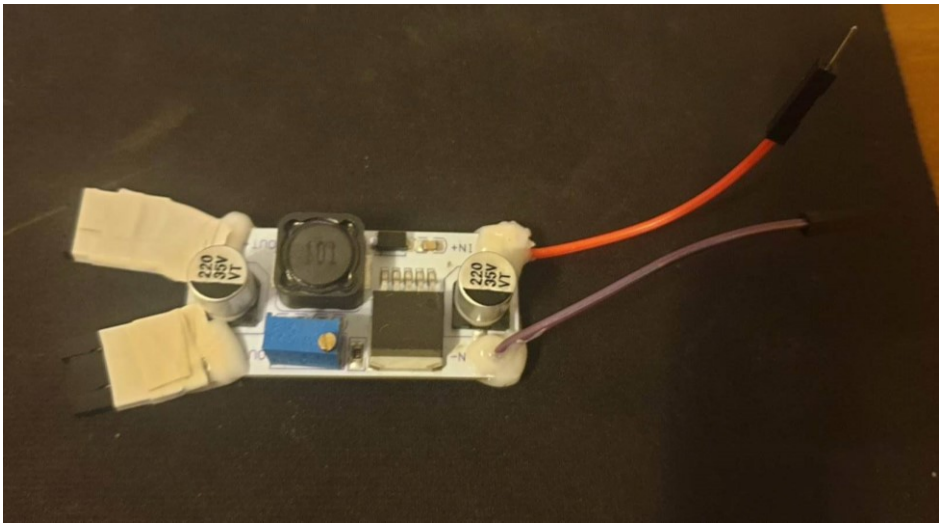
Slika 31 Shematski prikaz spajanja komponenti

Svaki dio spajali smo sa spojnim žicama VELLEMAN C00036 s muškim, ženskim ili muško/ženskim konektorima (slika 32) koje su namijenjene za Arduino projekte radi lakšeg spajanja, i pri potrebi, radi lakšeg rastavljanja.

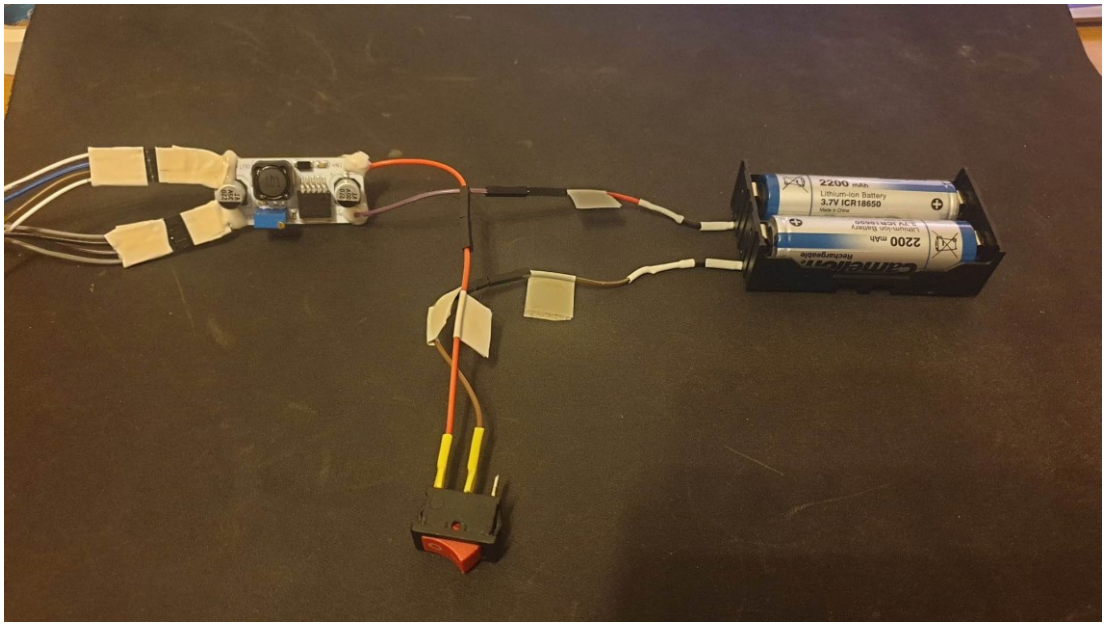


Slika 32 VELLEMAN C00036 žice s muško/ženskim konektorima

Zbog načina spajanja s navedenim žicama, stavljaju se konektori koji su kompatiblni s ulazima. To je učinjeno tako što se uklonio potrebnan broj ženskih konektora sa žica, uklonila se zaštitna izolacija, spojili ih, te zalemili za ulaze i izlaze određenih komponenti koje možemo vidjeti na sljedećim slikama.

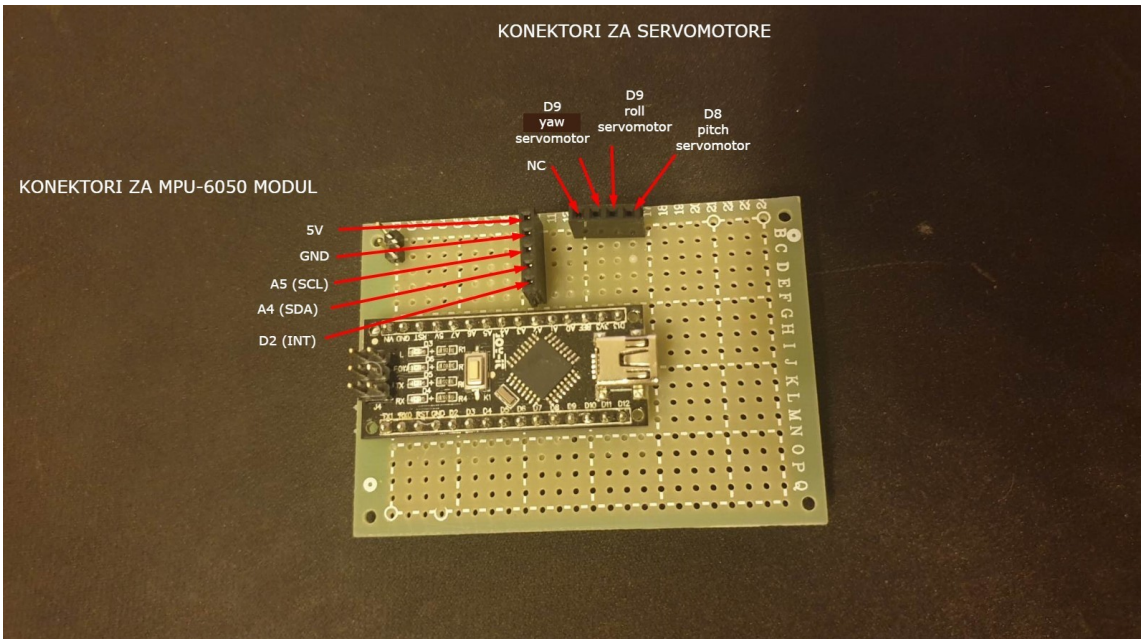


Slika 33 Pretvarač napona s konektorima



Slika 34 Spoj prekidača, napajanja i pretvarača napona

Razvojna pločica JOY-IT Atmega328 zalemljena je za proto ploču radi lakše konekcije komponenti i kako bi spriječili potencijalne kratke spojeve. Spojevi su izolirani koristeći vruće ljepilo. Spoj razvojne ploče s odgovarajućim konektorima za spoj sa servomotorima i modulom MPU-6050 možemo vidjeti na slici 35.



Slika 35 Spoj razvojne ploče s konektorima za MPU 6050 modul i servomotore

Nakon spajanja elektroničkih komponenti, spojili smo preostale komponente koje smo dobili 3D printanjem. Pod to se ubrajaju servomotori za njihove držače, te spoj uške i platforme. Navedene dijelove spojili smo M3 vijcima i maticama, a spojeve između zupčanika servomotora spojili smo plastičnim prirubnicama koje smo dobili u paketu sa servomotorima. Spojevi se mogu vidjeti na sljedećim slikama.

Slika 36 Spoj držača za servomotor i servomotora

Slika 37 Spoj platforme i uške

MPU-6050 modul je jedina elektronička komponenta koju smo učvrstili vijčanim spojem za gornji dio kućišta zbog toga što MPU-6050 modul uvijek mora biti fiksiran kako bi dobili očekivane rezultate stabilizirajuće platforme. U suprotnom, ako MPU-6050 modul nije pričvršćen, odnosno ako se počne pomicati pri korištenju uređaja, stabilizirajuća platforma neće raditi ispravno. Preostale komponente i žice smjestili smo u kućište i zatvorili poklopcem. Konačni izgled troosne samostabilizirajuće platforme možemo vidjeti na slici 38. pri stabiliziranju čaše s vodom.

Slika 38 Troosna samostabilizirajuća platforma

5.1. Cijena izrade sustava

U tablici 6 možemo vidjeti troškovnik izrađenog sustava koji približno iznosi 184,68 *EUR*. Trošak materijala i usluga 3D printanja određena je prosječnom cijenom materijala i usluga obrta 3D printanja iako je rađeno uz pomoć vanjskog suradnika FSB fakulteta.

Tablica 6 Cijena komponenti za izradu sustava

Naziv komponente	Naziv trgovine	Cijena po komadu [EUR]	Broj komada	Cijena ukupno [EUR]
JOY-IT Atmega328p-AU	Chipoteka	15,78	1	15,78
MPU-6050	Mikrotron	10,58	1	10,58
LM2596S	Chipoteka	7,83	1	7,83
MG996R TowerPro servomotor	Mikrotron	11,90	3	23,80
VELLEMAN C000036 spojne žice muško/muški konektori	Chipoteka	4,63	1	4,63
VELLEMAN C000036 spojne žice muško/ženski konektori	Chipoteka	4,63	1	4,63
VELLEMAN C000036 spojne žice žensko/ženski konektori	Chipoteka	4,63	1	4,63
Litijeva baterija 3,7 V 18650 Li-ion	Chipoteka	4,51	2	9,02
M3 vijci i matice	Pevex	1,99	1	1,99
Kućište za litijeve baterije	Chipoteka	3,58	1	3,58
Prekidač VAGA	Bart Elektronika	1,00	1	1,00
Proto ploča	Tevetron	4,00	1	4,00
Materijal i usluge 3D printanja	/	93,21	/	93,21
			Ukupno:	184,68

6. ZAKLJUČAK

U ovom radu izrađena je troosna samostabilizirajuća platforma koja služi za stabilizaciju predmeta pri pomicanju ljudskog tijela kod kretanja. Pri izradi vlastite stabilizirajuće platforme, referirali smo se na stabilizaciju suvremenih kamera manjih dimenzija kako bi kadar, pa u konačnici, fotografija ili videozapis bili zadovoljavajuće kvalitete i orijentacije pri kretanju ljudskog tijela ili ruke kod snimanja.

Sustav se sastoji od razvojne ploče, MPU-6050 žiroskopa i akcelerometra, servomotora, pretvarača napona i izvora napajanja. Ključnu ulogu ovog za stabilizaciju sustava ima MPU-6050 žiroskop i akcelerometar. Navedeni modul ima funkciju senzora kako bi platforma pomoću njegovih izlaznih signala ostala u horizontalnom položaju pri određenim rotacijama. Dijelovi konstrukcije napravljeni su u programu za 3D modeliranje AutoDesk Inventor. Nakon kreiranja modela, dijelovi su izrađeni korištenjem 3D printera.

Prvi problemi nastali su prilikom kalibracije MPU-6050 modula. Kroz duži period vremena, zbog nedostatka magnetometra, počele su se pojavljivati pogreške senzora uslijed nakupljanja malih vrijednosti izlaznih signala koji nemaju korelaciju s ulaznim signalom (engl. drifting), te nam je zbog toga rotacija skretanja (yaw) počela odmicati neovisno o našim pokretima. Nakon istraživanja kako bismo riješili tu problematiku, došli smo do spoznaje o procesoru DMP koji je ugrađen na korištenom MPU-6050 modulu. Ugrađeni procesor također može izračunati i ispisati kvartenione koji se koriste za predstavljanje orijentacije i rotacije 3D objekta u prostoru.

Nakon što smo ponovno kalibrirali sustav s potrebnim bibliotekama za korištenje DMP procesora, signali unosa bili su vjerodostojni izlaznom signalu, a najvažnije od svega, riješili smo se driftanja rotacije skretanja. Nakon rjeđenja problema očitavanja signala sa senzora, načinjen je prikaz 3D objekta kako bismo najbolje prikazali rezultate ponašanja senzora. Nakon uspješne eksperimentalne verifikacije, sve elektroničke komponente smo spojili, izolirali kako ne bi došlo do kratkog spoja u kućištu, te spojili preostale dijelove konstrukcije.

Iako izrađena troosna samostabilizirajuća platforma ne može najpravilnije obnašati funkciju pravog gimbal stabilizatora, najviše zbog toga što su rabljene jeftine komponente s ciljem demonstracije načina rada sustava, a ne njegovo korištenje u profesionalne svrhe, ovaj projekt vjerodostojno prikazuje koja su znanja i vještine potrebne za izradu tipičnog mehatroničkog sustava.

LITERATURA

[1] https://xiaomiplanet.sk/wp-content/uploads/2020/07/xiaomi-mijia-gimbal-recenzia-cover.jpg Pristupljeno 10.12.2022.

[2] https://s.twojahistoria.pl/uploads/2021/10/George_Malins_z_aeroskopem_w_Belgiii_1914.jpg Pristupljeno 13.12.2022.

[3] https://en.wikipedia.org/wiki/Aeroscope Pristupljeno 15.12.2022.

[4] https://en.wikipedia.org/wiki/Steadicam Pristupljeno 15.12.2022.

[5] https://upload.wikimedia.org/wikipedia/commons/thumb/1/16/John_E_Fry _Steadicam_ Operator_UK.jpg/ 1200px-John_E_Fry_Steadicam_Operator_UK.jpg Pristupljeno 16.12.2022.

[6] https://enciklopedija.hr/natuknica.aspx?ID=22090 Pristupljeno 22.12.2022.

[7] https://glidegear.net/blogs/news/a-brief-history-of-the-dslr-camera-gimbal Pristupljeno 23.12.2022.

[8] https://www.captureguide.com/wp-content/uploads/2022/02/inkee-falcon-gimbal.jpg Pristupljeno 23.12.2022.

[9] https://momofilmfest.com/smartphone-gimbal-buyers-guide/ Pristupljeno 23.12.2022.

[10] https://asset.conrad.com/media10/isa/160267/c1/-/hr/1678142_RB_00_FB/image.jpg Pristupljeno 23.12.2022.

[11] https://joy-it.net/files/files/Produkte/ARD_NanoV3/ARD_NanoV3_Datasheet_2022-09-08.pdf Pristupljeno 24.12.2022.

[12] https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf Pristupljeno 24.12.2022.

[13] https://www.electronicwings.com/storage/PlatformSection/TopicContent/138/description/MPU6050%20Module.jpg Pristupljeno 24.12.2022.

[14] https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/ Pristupljeno 26.12.2022.

[15] https://www.youtube.com/watch?v=M9lZ5Qy5S2s&ab_channel=EEEnthusiast Pristupljeno 26.12.2022.

[16] https://cpc.farnell.com/whadda/wpm404/adjustable-voltage-step-down-module/dp/SC18724 Pristupljeno 26.12.2022.

[17] https://www.towerpro.com.tw/product/mg995-robot-servo-180-rotation/ Pristupljeno 26.12.2022.

[18] https://www.makerlab-electronics.com/product/digital-servo-motor-mg996r-180-rotation/ Pristupljeno 26.12.2022.

[19] https://www.handsontec.com/dataspecs/motor_fan/MG996R.pdf Pristupljeno 26.12.2022.

[20] https://howtomechatronics.com/ Pristupljeno 26.12.2022.

[21] http://www.i2cdevlib.com/ Pristupljeno 26.12.2022.

[22] https://www.geekmomprojects.com/mpu-6050-dmp-data-from-i2cdevlib/ Pristupljeno 27.12.2022.

[23] https://mjwhite8119.github.io/Robots/mpu6050 Pristupljeno 27.12.2022.

[24] Adolf Friedrich Ludwig Bredenkamp, *Development and control of 3-axis Stabilised platform*, ožujak 2007., Pristupljeno 27.12.2022.

[25] Ali Algoz, Bakhtiyar Asef Hasnain, *A control system for a 3-axis camera stabilizer,* lipanj 2018., Pristupljeno 27.12.2022.

PRILOZI

I. CD-R disc

II. Programski kod

III. Tehnički crteži

Program „RUČNA KALIBRACIJA“ – *Arduino IDE*

```
// Ukljucivanje eksternih librarija
#include <Wire.h>
//Incijalizacija varijabli
const int MPU = 0x68; // MPU6050 I2C address
float AccX, AccY, AccZ;
float GyroX, GyroY, GyroZ;
float accAngleX, accAngleY, gyroAngleX, gyroAngleY, gyroAngleZ;
float roll, pitch, yaw;
float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;
float elapsedTime, currentTime, previousTime;
int c = 0;
void setup() {
  Serial.begin(19200);
  Wire.begin();                      // inicijalizacija komunikacije
  Wire.beginTransmission(MPU);       // komunikacija s MPU6050 // MPU=0x68
  Wire.write(0x6B);                  // započni komunikaciju s registrom 6B
  Wire.write(0x00);                  // resetiraj - stavi 0 u 6B registar
  Wire.endTransmission(true);
  /*

  // Podesi osjetljivost akcelerometra
  Wire.beginTransmission(MPU);
  Wire.write(0x1C);
  Wire.write(0x10);
  Wire.endTransmission(true);

  // Podesi osjetljivost žiroskopa
  Wire.beginTransmission(MPU);
  Wire.write(0x1B);
  Wire.write(0x10);
  Wire.endTransmission(true);
  delay(20);
  */

  // Pozovi navedenu funkciju
  calculate_IMU_error();
  delay(20);
}
void loop() {

  // === Iščitavanje podataka akcelerometra === //

  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
```

```
  Wire.requestFrom(MPU, 6, true); // komunikacija s sljedećih 6 registara

  AccX = (Wire.read() << 8 | Wire.read()) / 16384.0; // X-axis value
  AccY = (Wire.read() << 8 | Wire.read()) / 16384.0; // Y-axis value
  AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0; // Z-axis value

  // Kalkulacija Roll i Pitch rotacija fiz akcelerometra

  accAngleX = (atan(AccY / AccZ) * 180 / PI) - 0.58; // AccErrorX ~(0.58) See the
calculate_IMU_error()custom function for more details
  accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) * 180 / PI) + 1.58;
// AccErrorY ~(-1.58)

  // === Iščitavanje podataka žiroskopa === //

  previousTime = currentTime;
  currentTime = millis();
  elapsedTime = (currentTime - previousTime) / 1000;
  Wire.beginTransmission(MPU);
  Wire.write(0x43); // registar žiroskopa
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true);
  GyroX = (Wire.read() << 8 | Wire.read()) / 131.0; // za 250°/s raspon mjerenja
dijelimo s 131.0
  GyroY = (Wire.read() << 8 | Wire.read()) / 131.0;
  GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0;

  //Korekcija izlaznih vrijednosti

  GyroX = GyroX + 0.56;
  GyroY = GyroY - 2;
  GyroZ = GyroZ + 0.79;

  gyroAngleX = gyroAngleX + GyroX * elapsedTime;
  gyroAngleY = gyroAngleY + GyroY * elapsedTime;
  yaw =  yaw + GyroZ * elapsedTime;
  //Komplementarni filtar
  gyroAngleX = 0.96 * gyroAngleX + 0.04 * accAngleX;
  gyroAngleY = 0.96 * gyroAngleY + 0.04 * accAngleY;

  roll = gyroAngleX;
  pitch = gyroAngleY;

  // Print the values on the serial monitor printanje vrijednosti na Serial Monitor-u
  Serial.print("ypr  ");
  Serial.print(yaw);
  Serial.print("/");
```

```
  Serial.print(pitch);
  Serial.print("/");
  Serial.println(roll);
}

//Funkcija "calculate_IMU_error()"
void calculate_IMU_error() {

  while (c < 200) {
    Wire.beginTransmission(MPU);
    Wire.write(0x3B);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU, 6, true);
    AccX = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
    AccY = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
    AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0 ;


    AccErrorX = AccErrorX + (atan(AccY / AccZ) * 180 / PI);
    AccErrorY = AccErrorY + ((atan(-1 * (AccX) / sqrt(pow((AccY), 2) + pow((AccZ), 2)))
* 180 / PI));
    c++;
  }


  AccErrorX = AccErrorX / 200;
  AccErrorY = AccErrorY / 200;
  c = 0;

  while (c < 200) {
    Wire.beginTransmission(MPU);
    Wire.write(0x43);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU, 6, true);
    GyroX = Wire.read() << 8 | Wire.read();
    GyroY = Wire.read() << 8 | Wire.read();
    GyroZ = Wire.read() << 8 | Wire.read();
    // Sum all readings
    GyroErrorX = GyroErrorX + (GyroX / 131.0);
    GyroErrorY = GyroErrorY + (GyroY / 131.0);
    GyroErrorZ = GyroErrorZ + (GyroZ / 131.0);
    c++;
  }

  GyroErrorX = GyroErrorX / 200;
  GyroErrorY = GyroErrorY / 200;
  GyroErrorZ = GyroErrorZ / 200;
```

```
  Serial.print("AccErrorX: ");
  Serial.println(AccErrorX);
  Serial.print("AccErrorY: ");
  Serial.println(AccErrorY);
  Serial.print("GyroErrorX: ");
  Serial.println(GyroErrorX);
  Serial.print("GyroErrorY: ");
  Serial.println(GyroErrorY);
  Serial.print("GyroErrorZ: ");
  Serial.println(GyroErrorZ);
}
```

Program „KALIBRACIJA MODULA MPU-6050“ – *Arduino IDE*

```
// Ukljucivanje eksternih librarija
#include "I2Cdev.h"
#include "MPU6050.h"

//Provjera dali je I2CDEV_IMPLEMENTATION jednaka I2CDEV_ARDUINO_WIRE i ako je ukljuci
Wire.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif


MPU6050 accelgyro;

//Incijalizacija varijabli
const char LBRACKET = '[';
const char RBRACKET = ']';
const char COMMA    = ',';
const char BLANK    = ' ';
const char PERIOD   = '.';

const int iAx = 0;
const int iAy = 1;
const int iAz = 2;
const int iGx = 3;
const int iGy = 4;
const int iGz = 5;

const int usDelay = 3150;
const int NFast =  1000;
const int NSlow = 10000;
const int LinesBetweenHeaders = 5;
      int LowValue[6];
      int HighValue[6];
      int Smoothed[6];
      int LowOffset[6];
      int HighOffset[6];
      int Target[6];
      int LinesOut;
      int N;

void ForceHeader()
  { LinesOut = 99; }

void GetSmoothed()
  { int16_t RawValue[6];
```

```
    int i;
    long Sums[6];
    //for petlja koja popunjava Sums na svim mjestima s 0
    for (i = iAx; i <= iGz; i++)
      { Sums[i] = 0; }


    //For petlja koja popunjava accelgyro motion6
    for (i = 1; i <= N; i++)
      {
        accelgyro.getMotion6(&RawValue[iAx], &RawValue[iAy], &RawValue[iAz],
                             &RawValue[iGx], &RawValue[iGy], &RawValue[iGz]);
        if ((i % 500) == 0)
          Serial.print(PERIOD);
        // stopiraj na 3 sekunde
        delayMicroseconds(usDelay);
        //Popunjavanje Sums s pravim vrijednostima (0 + RawValue)
        for (int j = iAx; j <= iGz; j++)
          Sums[j] = Sums[j] + RawValue[j];
      }

    for (i = iAx; i <= iGz; i++)
      { Smoothed[i] = (Sums[i] + N/2) / N ; }
  }

void Initialize()
  {

     //Provjera dali je I2CDEV_IMPLEMENTATION jednaka I2CDEV_ARDUINO_WIRE i ako je
pokreni Wire.begin()
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
     //Provjera dali je I2CDEV_IMPLEMENTATION jednaka I2CDEV_BUILTIN_FASTWIRE i ako je
pokreni Fastwire setup
        Fastwire::setup(400, true);
    #endif

    //Ispis na serial monitor s brzinom od 115200 bauda
    Serial.begin(115200);


    Serial.println("inicijalizacija uredaja...");
    accelgyro.initialize();


    Serial.println("Testiranje konekcije uredaja...");
```

```
    Serial.println(accelgyro.testConnection() ? "MPU6050 uspjesna konekcija" :
"MPU6050 neuspjesna konekcija");
    Serial.println("PID podesavanje tocaka = 100 ocitanja");



        accelgyro.CalibrateAccel(6);
        accelgyro.CalibrateGyro(6);
        Serial.println("\n 600 ocitanja");
        accelgyro.PrintActiveOffsets();
        Serial.println();
        accelgyro.CalibrateAccel(1);
        accelgyro.CalibrateGyro(1);
        Serial.println("700 ocitanja");
        accelgyro.PrintActiveOffsets();
        Serial.println();
        accelgyro.CalibrateAccel(1);
        accelgyro.CalibrateGyro(1);
        Serial.println("800 ocitanja");
        accelgyro.PrintActiveOffsets();
        Serial.println();
        accelgyro.CalibrateAccel(1);
        accelgyro.CalibrateGyro(1);
        Serial.println("900 ocitanja");
        accelgyro.PrintActiveOffsets();
        Serial.println();
        accelgyro.CalibrateAccel(1);
        accelgyro.CalibrateGyro(1);
        Serial.println("1000 ocitanja");
        accelgyro.PrintActiveOffsets();
     Serial.println("\n\n Izvrseno:");
```

Program „OČITAVANJE YAW_PITCH_ROLL_VRIJEDNOSTI“ i „OUTPUT_ZA_VIZUALIZACIJU“ – *Arduino IDE*

```
// Ukljucivanje eksternih librarija
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include <string.h>



//Provjera dali je I2CDEV_IMPLEMENTATION jednaka I2CDEV_ARDUINO_WIRE i ako je ukljuci
Wire.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif


MPU6050 mpu;


//#define IZLAZNE_VRIJEDNOSTI_YAW_PITCH_ROLL
//#define VIZUALIZACIJA


//Definiranje pinova gdje blinka žarulja i interupt pin
#define INTERRUPT_PIN 2
#define LED_PIN 13
bool blinkState = false;

//Incijalizacija varijabli
//unit_8_t -> 8 bitni unsigned integer u rangu od 0 - 255 brojeva
//unit_16_t -> 16 bitni unsigned integer u rangu od 0 - 65535 brojeva

// MPU varijable
bool dmpInitReady = false;
uint8_t mpuIntStatus;
uint8_t devStatus;
uint16_t packetSize;
uint16_t fifoCount;
uint8_t fifoBuffer[64];

// varijable orijentacije i rotacije
Quaternion q;           // [w, x, y, z]         quaternion container

VectorFloat gravVec;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity
vector
```

```
//inicijalizata Visualisation s predefiniranim vrijednostima
uint8_t Visualisation[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };


//detekcija prekida

volatile bool mpuInterrupt = false;
void dmpDataReady() {
    mpuInterrupt = true;
}



//inicijalizacija

void setup() {

    //Provjera dali je I2CDEV_IMPLEMENTATION jednaka I2CDEV_ARDUINO_WIRE i ako je
pokreni Wire.begin() i stavit clock na 400000
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
        Wire.setClock(400000);
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
    //Provjera dali je I2CDEV_IMPLEMENTATION jednaka I2CDEV_BUILTIN_FASTWIRE i ako je
pokreni Fastwire setup
        Fastwire::setup(400, true);
    #endif


    //Ispis na serial monitor s brzinom od 115200 bauda
    Serial.begin(115200);



    // inicijalizacija uređaja
    Serial.println(F("Inicijalizacija I2C uredaja..."));
    mpu.initialize();
    pinMode(INTERRUPT_PIN, INPUT);

    // testiranje konekcije
    Serial.println(F("Testiranje konekcije..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 uspjesno konektiran") :
F("MPU6050 neuspjesno konektiran"));

    // pričekati dok se ne stisne tipka
    Serial.println(F("\nZa pokretanje aktivacije, stisni bilo koju tipku: "));
```

```
  //While koji ceka da se stisne tipka na tipkovnici
  while (Serial.available() && Serial.read()); // empty buffer
  while (!Serial.available());                 // wait for data
  while (Serial.available() && Serial.read()); // empty buffer again

  // inicijalizacija DMP-a
  Serial.println(F("Inicijalizacija DMP-a..."));
  devStatus = mpu.dmpInitialize();


  //Definiranje odstupanja po X,Z,Y osi driftanje
mpu.setXGyroOffset(-31);
mpu.setYGyroOffset(-34);
mpu.setZGyroOffset(-36);
mpu.setZAccelOffset(811);


  if (devStatus == 0) {

      mpu.CalibrateAccel(6);
      mpu.CalibrateGyro(6);
      mpu.PrintActiveOffsets();
      // Paljenje DMP-a
      Serial.println(F("Paljenje DMP-a..."));
      mpu.setDMPEnabled(true);


      Serial.print(F("pokretanje interrupt naredbe (Arduino eksterni interrupt "));
      Serial.print(digitalPinToInterrupt(INTERRUPT_PIN));
      Serial.println(F(")..."));
      attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
      mpuIntStatus = mpu.getIntStatus();


      Serial.println(F("DMP spreman za interrupt..."));
      dmpInitReady = true;


      packetSize = mpu.dmpGetFIFOPacketSize();
  } else {
      //U gornji if ulazi ako je devStatus = 0 sto je uredu
      //Ukoliko je 1,2 ili neki drugi broj ulazi ovdje
      //Ispod su opisi sto moze bit da bi uslo u ovaj else dio


      Serial.print(F("DMP greška u inicijalizaciji (code "));
      Serial.print(devStatus);
```

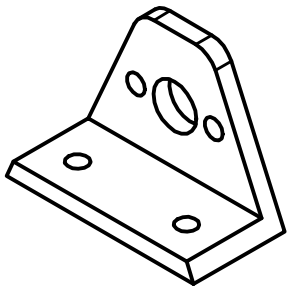
```
        Serial.println(F(")"));
    }


    //Postavljenje led pina da može svjetlit
    pinMode(LED_PIN, OUTPUT);
}



//Postavljanje glavnog programa
void loop() {

    if (!dmpInitReady) return;
    // iščitavaj FIFO
    if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) {

        #ifdef IZLAZNE_VRIJEDNOSTI_YAW_PITCH_ROLL

            //Slanje varijable u fukciju koja popunjava s vrijednostima
            //& označava referencu na varijablu (q, gravVec)
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetGravity(&gravVec, &q);
            mpu.dmpGetYawPitchRoll(ypr, &q, &gravVec);
            //ispis ypr prema formuli (ypr) * 180 / pi
            Serial.print("ypr\t");
            Serial.print(ypr[0] * 180/M_PI);
            Serial.print("\t");
            Serial.print(ypr[1] * 180/M_PI);
            Serial.print("\t");
            Serial.println(ypr[2] * 180/M_PI);
        #endif



        #ifdef VIZUALIZACIJA
            //Prikaži kvaternione za 3D vizualizaciju
            Visualisation[2] = fifoBuffer[0];
            Visualisation[3] = fifoBuffer[1];
            Visualisation[4] = fifoBuffer[4];
            Visualisation[5] = fifoBuffer[5];
            Visualisation[6] = fifoBuffer[8];
            Visualisation[7] = fifoBuffer[9];
            Visualisation[8] = fifoBuffer[12];
            Visualisation[9] = fifoBuffer[13];
            Serial.write(Visualisation, 14);
```

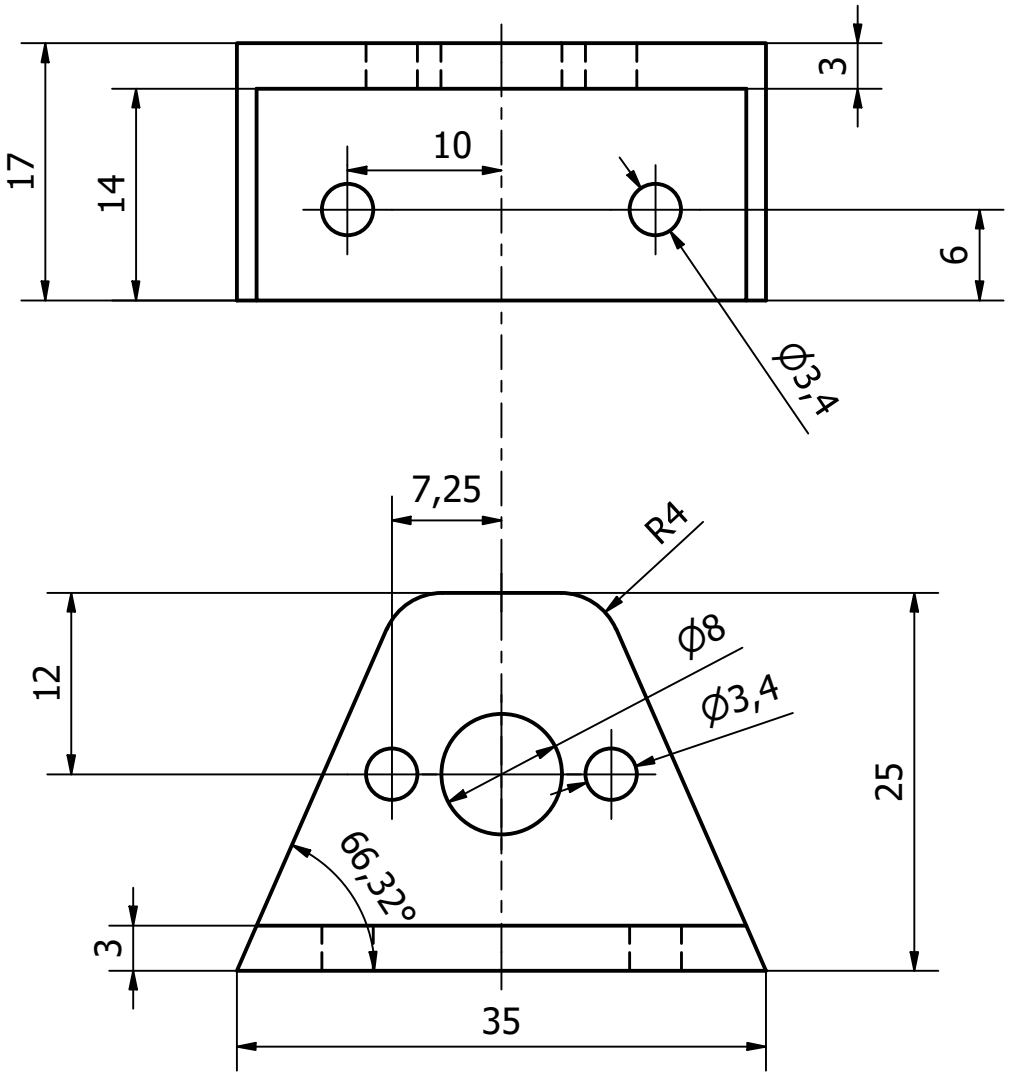
```
            Visualisation[11]++;
        #endif


        blinkState = !blinkState;
        //Paljenje led žarulje na arudino
        digitalWrite(LED_PIN, blinkState);
    }
}
```

Program „3D_VIZUALIZACIJA“ – *Processing Software*

```
// Ukljucivanje eksternih librarija
import processing.serial.*;

import toxi.geom.*;




//Defiranje varijabli
Serial port;
char[] Visualisation = new char[14];
int serialCount = 0;

int interval = 0;

float[] q = new float[4];
Quaternion quat = new Quaternion(1, 0, 0, 0);


void setup() {

    //velicina ekrana
    size(1000, 1000, OPENGL);

    //Dohvacanje prvog porta u listi portova
    String portName = Serial.list()[0];


    //Kreiranje porta s imenom porta i citanjem 115200 bauda
    port = new Serial(this, portName, 115200);

    //slanje bilo kojeg simbola kako bi upalili DMP

    port.write('r');
}

void draw() {
    if (millis() - interval > 1000) {

        port.write('r');
        interval = millis();
    }

    // crna pozadina
    background(0);


    translate(width / 2, height / 2);


    //Okretanje platforme kako se mice plocica uzivo
    float[] axis = quat.toAxisAngle();
```

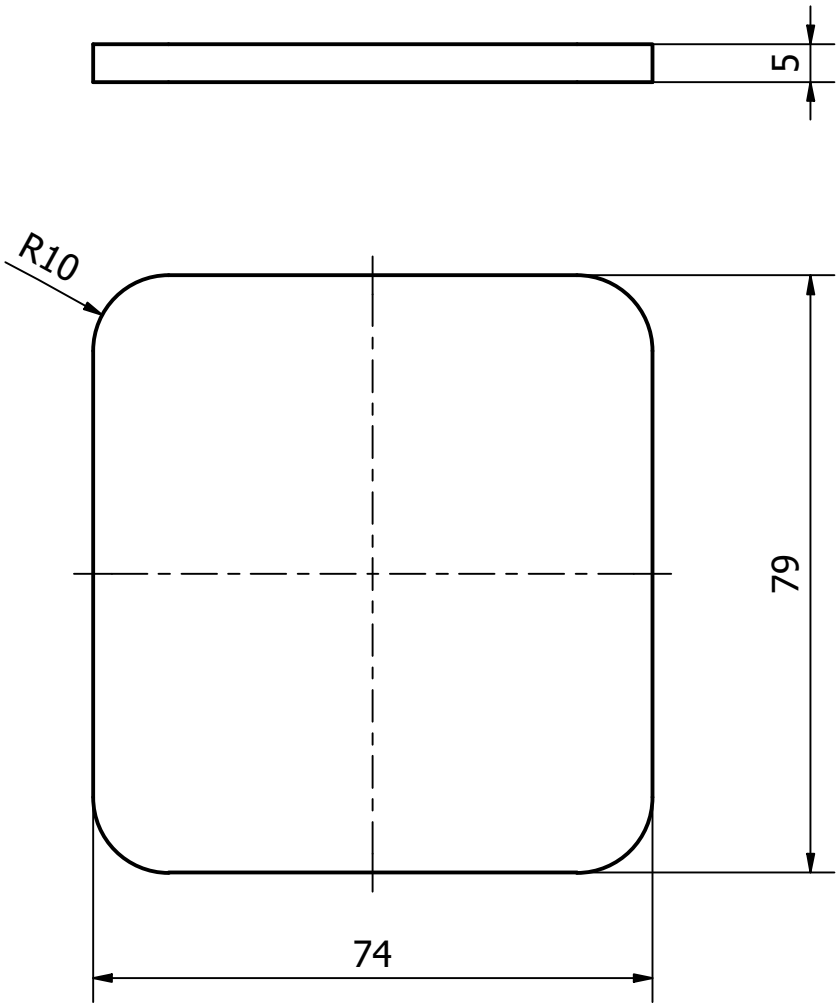
```
    rotate(axis[0], -axis[1], axis[3], axis[2]);

    // uljepsavanje teksta (boja, velicina, pozicija), dodavanje boje platforme
    textSize(30);
  fill(0, 76, 153);
  box (200, 40, 400);
  textSize(25);
  fill(255, 255, 255);
  text("platforma", -50, 10, 201);

}

void serialEvent(Serial port) {
    interval = millis();

    while (port.available() > 0) {
        int ch = port.read();



        //Provjere dali serialCount sadrži znak $
        if (serialCount > 0 || ch == '$') {
            Visualisation[serialCount++] = (char)ch;
            if (serialCount == 14) {
                //Da bi se uslo ovdje treba biti 14 serialCount koji se
inkrementira svakim prolaskom kroz while uvjet
                serialCount = 0;



                //Dobivanje quaterniona
                q[0] = ((Visualisation[2] << 8) | Visualisation[3]) / 16384.0f;
                q[1] = ((Visualisation[4] << 8) | Visualisation[5]) / 16384.0f;
                q[2] = ((Visualisation[6] << 8) | Visualisation[7]) / 16384.0f;
                q[3] = ((Visualisation[8] << 8) | Visualisation[9]) / 16384.0f;
                for (int i = 0; i < 4; i++) if (q[i] >= 2) q[i] = -4 + q[i];


                //Settanje quaterniona u classu Quaternion koja sluzi kako bi se
platforma vrtila u real-timu kako micemo nas arduino u ruci
                quat.set(q[0], q[1], q[2], q[3]);
            }
        }
    }
}
```





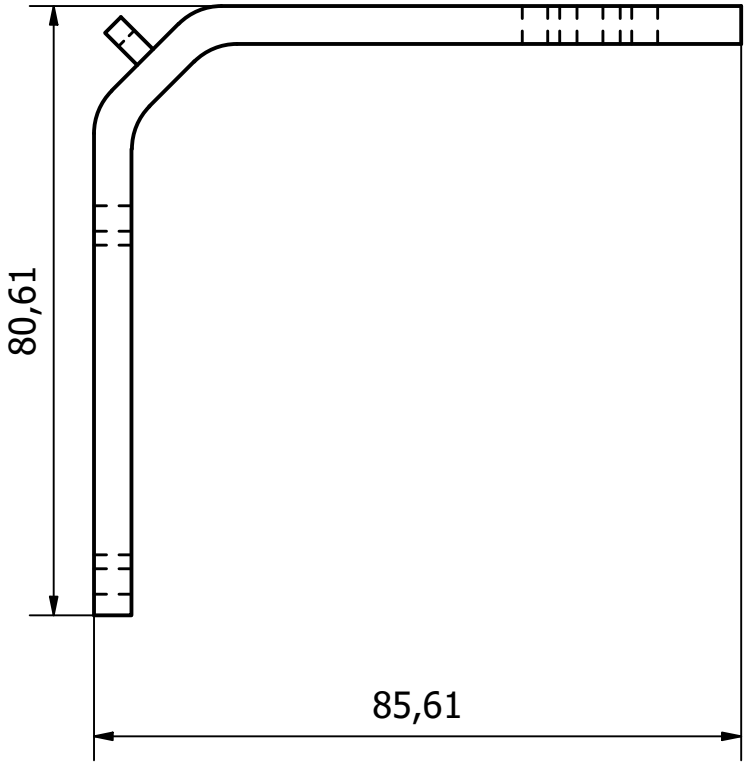
	Datum	Ime i prezime	Potpis	FSB Zagreb
Projektirao		Martin Vugić		
Razradio		Martin Vugić		
Crtao		Martin Vugić		
Pregledao		Željko Šitum		

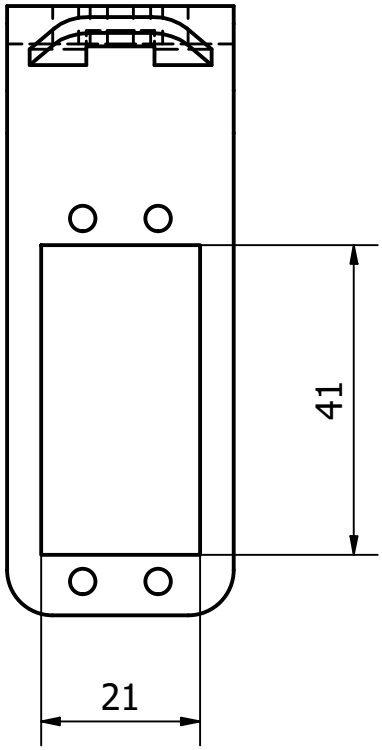
Objekt:		Objekt broj:	
		R. N. broj:	
Napomena:			Kopija:
Materijal: PLA	Masa:		
Mjerilo originala: 2:1	Naziv: UŠKA ZA PLATFORMU	Pozicija: 1	Format: A4
			Listova:
	Crtež broj: ST_PLAT-000-001		List:

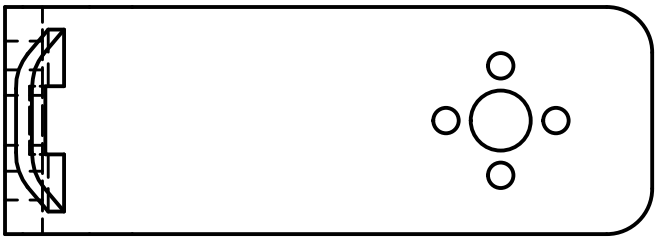


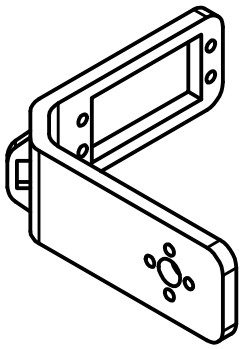
	Datum	Ime i prezime	Potpis	FSB Zagreb
Projektirao		Martin Vugić		
Razradio		Martin Vugić		
Crtao		Martin Vugić		
Pregledao		Željko Šitum		

Objekt:	Objekt broj:	
	R. N. broj:	
Napomena:		Kopija:
Materijal: PLA	Masa:	
Naziv: PLATFORMA	Pozicija: 2	Format: A4
Mjerilo originala: 1:1		Listova:
Crtež broj: ST_PLAT-000-002		List:



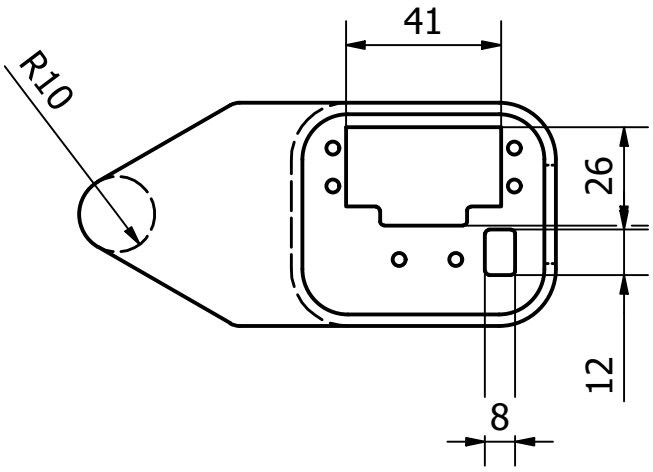


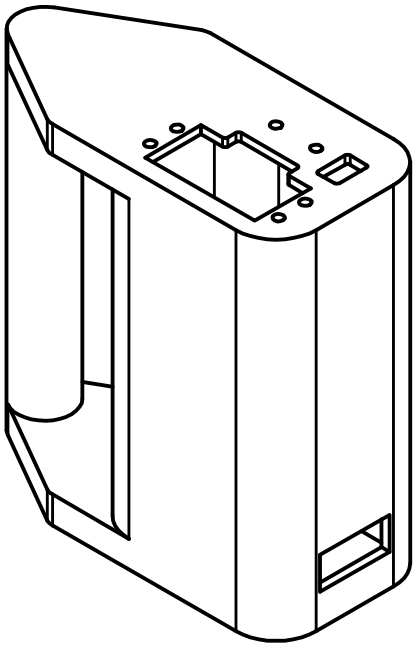


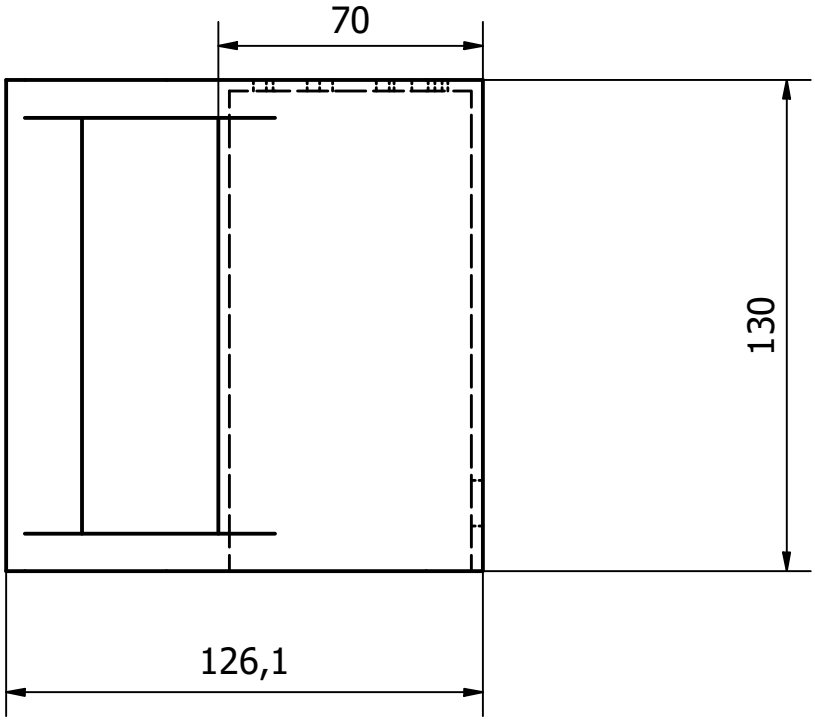


	Datum	Ime i prezime	Potpis	
Projektirao		Martin Vugić		FSB Zagreb
Razradio		Martin Vugić		
Crtao		Martin Vugić		
Pregledao		Željko Šitum		

Objekt:	Objekt broj:	
	R. N. broj:	
Napomena:		Kopija:
Materijal: PLA	Masa:	
Naziv: DRŽAČ SERVOMOTORA	Pozicija: 3	Format: A4
Mjerilo originala: 1:1		Listova:
Crtež broj: ST_PLAT-000-003		List:

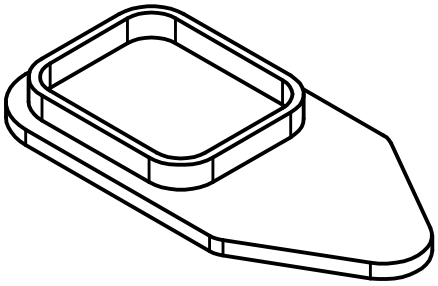


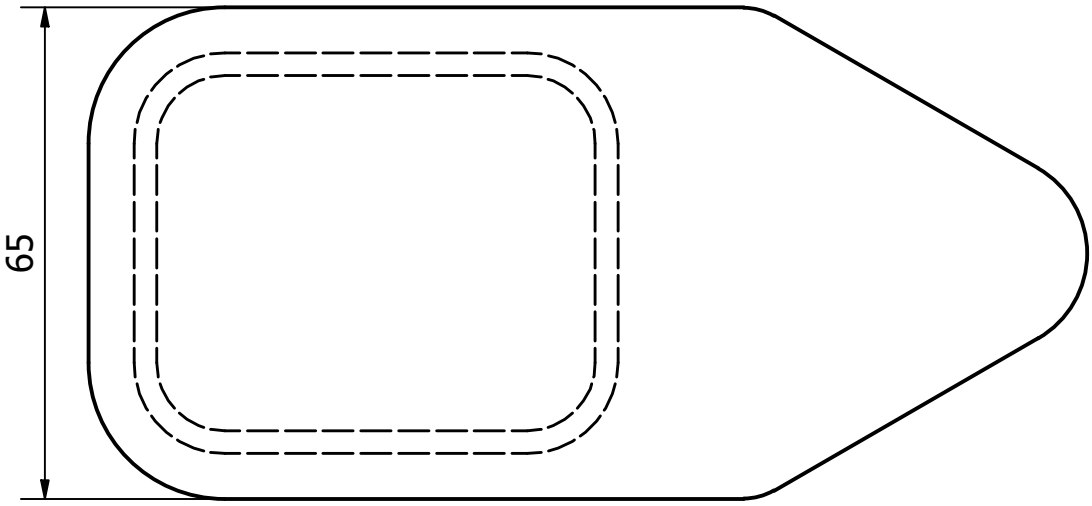


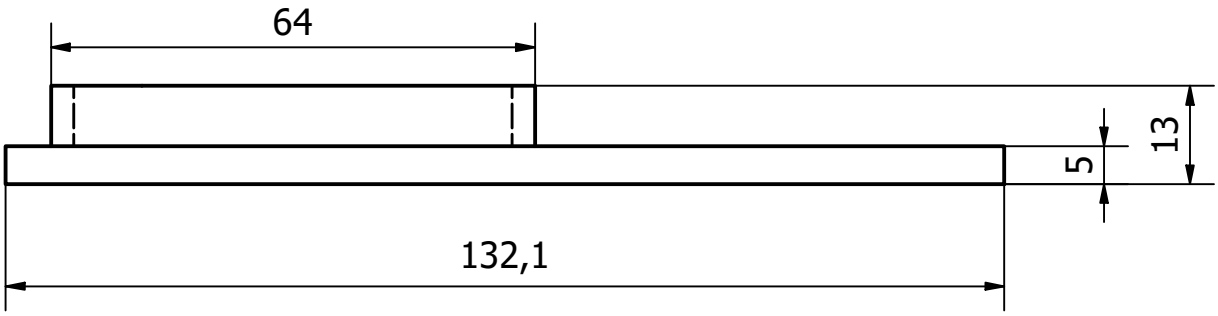


	Datum	Ime i prezime	Potpis	FSB Zagreb
Projektirao		Martin Vugić		
Razradio		Martin Vugić		
Crtao		Martin Vugić		
Pregledao		Željko Šitum		

Objekt:	Objekt broj:	
	R. N. broj:	
Napomena:		Kopija:
Materijal: PLA	Masa:	
Naziv: KUĆIŠTE	Pozicija: 4	Format: A4
Mjerilo originala: 1:2		Listova:
Crtež broj: ST_PLAT-000-004		List:

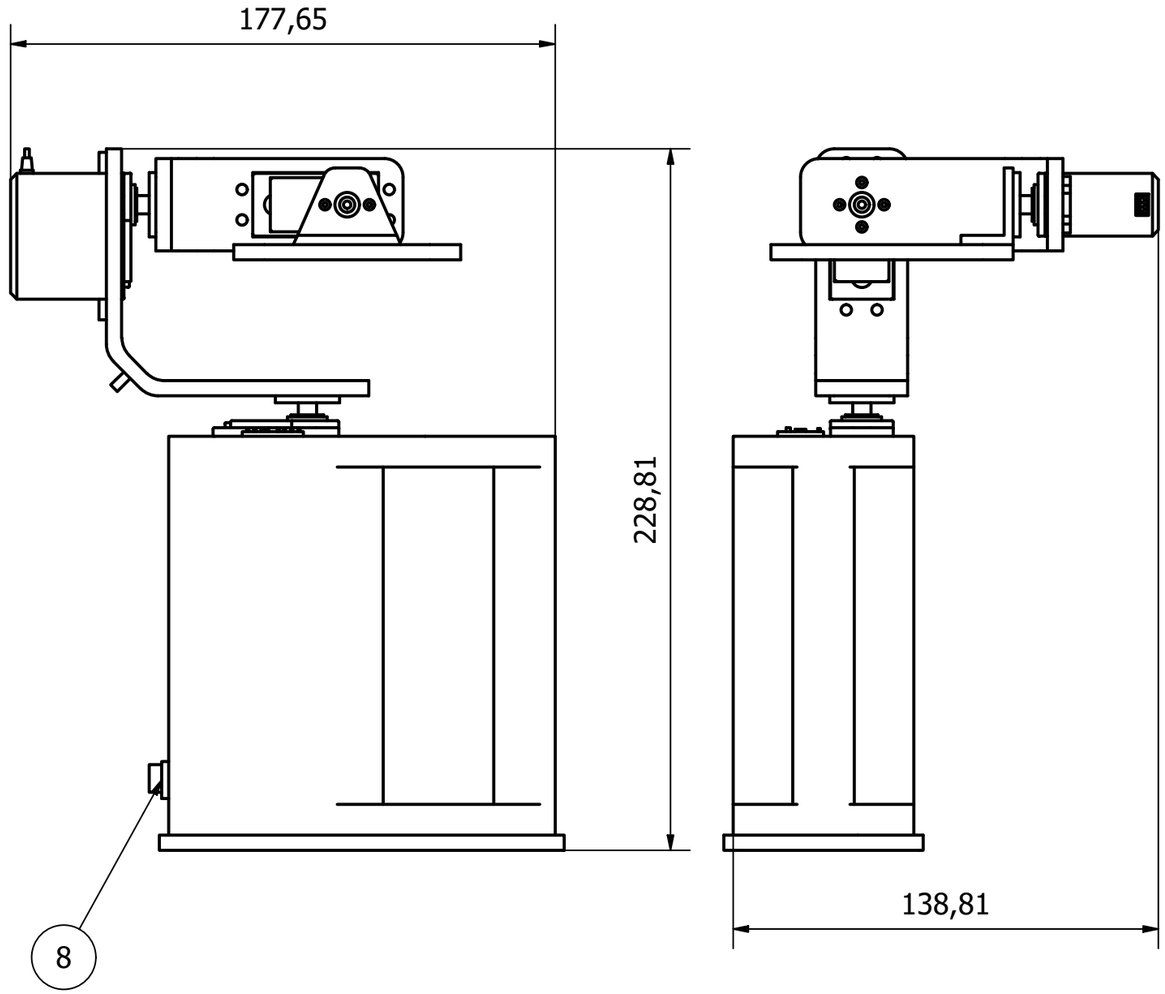


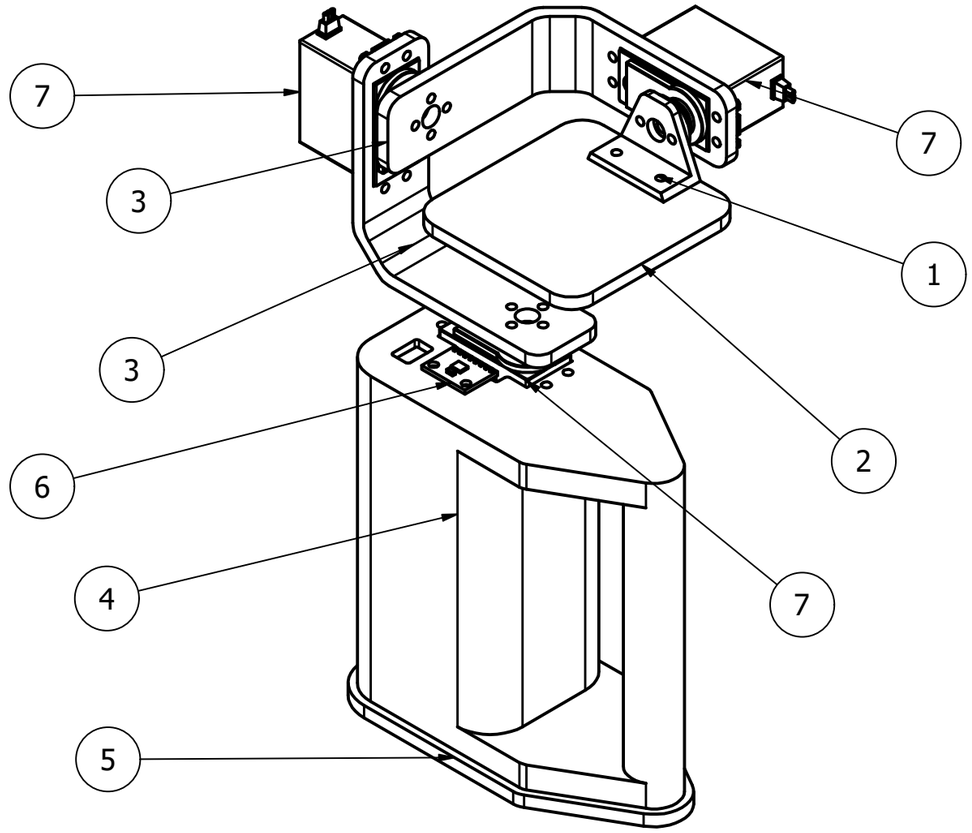


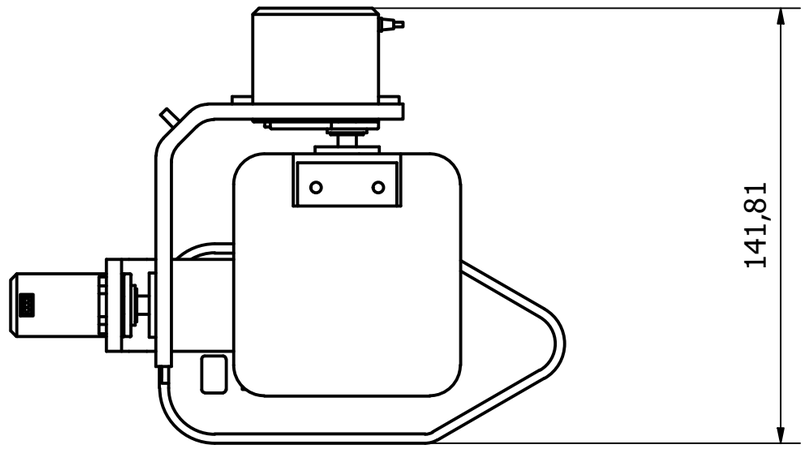


	Datum	Ime i prezime	Potpis	FSB Zagreb
Projektirao		Martin Vugić		
Razradio		Martin Vugić		
Crtao		Martin Vugić		
Pregledao		Željko Šitum		

Objekt:	Objekt broj:	
	R. N. broj:	
Napomena:		Kopija:
Materijal: PLA	Masa:	
Naziv: POKLOPAC KUĆIŠTA	Pozicija: 5	Format: A4
Mjerilo originala: 1:1		Listova:
Crtež broj: ST_PLAT-000-005		List:







Poz	Naziv dijela	Kom	Crtež broj Norma	Materijal	Sirove dimenzije Proizvođač	Masa
8	Prekidač	1				
7	MG996R TowerPro servomotor	3				
6	MPU-6050 modul	1				
5	Poklopac kućišta	1	ST_PLAT-000-005	PLA		
4	Kućište	1	ST_PLAT-000-004	PLA		
3	Držači servomotora	2	ST_PLAT-000-003	PLA		
2	Platforma	1	ST_PLAT-000-002	PLA		
1	Uška za platformu	1	ST_PLAT-000-001	PLA		

Broj naziva - code		Datum	Ime i prezime	Potpis
	Projektirao	20.12.2022	Martin Vugić	
	Razradio	20.12.2022	Martin Vugić	
	Crtao	20.12.2022	Martin Vugić	
	Pregledao	4.1.2023.	Željko Šitum	
	Mentor	4.1.2023.	Željko Šitum	

FSB Zagreb

ISO - tolerancije

Objekt:

Objekt broj:

R. N. broj:

Napomena:

Kopija:

Materijal:

Masa:

Naziv: Troosna samostabilizirajuća platforma

Pozicija:

Format: A3

Mjerilo originala: 1:2

Listova:

Crtež Broj: ST_PLAT-001-000

List: