Robotski insekt sa šest nogu

Trgovac, Patrick

Master's thesis / Diplomski rad

2022

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:235:020065

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Trgovac Patrick

Zagreb, 2022

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:
Dr. sc. Tomislav Stipančić

Student:
Trgovac Patrick

Zagreb, 2022./2023

IZJAVA

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

Zahvaljujem mentoru doc. dr.sc Tomislavu Stipančiću i Leonu Korenu, mag.ing.mech., na svim savjetima koje su mi pružili, uloženom vremenu te pomoći koju su mi pružili prilikom izrade ovog rada.

Također se zahvaljujem i svojoj obitelji koja me podržava tijekom cijelog mog školovanja.

Trgovac Patrick

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: 602-14/22-6/1	
Ur. broj: 15-1703-22-	

DIPLOMSKI ZADATAK

Student: **PATRICK TRGOVAC** Mat. br.: 0035205454

Naslov rada na hrvatskom jeziku: **Robotski insekt sa šest nogu**

Naslov rada na engleskom jeziku: **Six-legged robotic insect**

Opis zadatka:

Osim za industrijsku primjenu, roboti se koriste u najrazličitije svrhe. Domena primjene robota definira kako taj robot treba izgledati te koje sposobnosti treba posjedovati. Tako roboti mogu biti stacionarni ili mobilni. Za robote koji se kreću po nepristupačnim ili neravnim terenima nije dovoljno samo da imaju kotače pomoću kojih ostvaruju kretanje. Takvi su roboti vrlo često zmijoliki ili posjeduju različit broj nogu kako bi se mogli kretati kroz neuređenu, neravnu i nehomogenu okolinu.

U radu je potrebno:

- Osmisliti i prilagoditi konstrukcijsko rješenje za robotskog insekta sa šest nogu.
- Dizajnirati te izraditi fizičke komponente koje čine tijelo robota.
- Odrediti, prilagoditi i povezati elektroničke komponente uključujući upravljačko računalo, motore, senzore, ožičenje te druge komponente.
- Razviti programsku podršku koja omogućuje upravljanje robotom. Programska podrška uključuje upravljačke mehanizme implementirane direktno na robotu te funkcionalnu web stranicu koja omogućuje upravljanje robotom s udaljenog mjesta.

Razvijeno cjelovito rješenje robotskog insekta potrebno je eksperimentalno evaluirati.
U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan: 29. rujna 2022.

Rok predaje rada: 1. prosinca 2022.

Predviđeni datum obrane: 12. prosinca do 16. prosinca 2022.

Zadatak zadao:

doc. dr. sc. Tomislav Stipančić

Predsjednica Povjerenstva:

prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS TEHNIČKE DOKUMENTACIJE

BROJ CRTEŽA	Naziv
01-10-2022	Tijelo
02-10-2022	ZglobA_desni
03-10-2022	ZglobA_lijevi
04-10-2022	ZglobB
05-10-2022	ZglobC_desni
06-10-2022	ZglobC_lijevi
07-10-2022	Drzac
08-10-2022	Pomocni element 1
09-10-2022	Pomocni element 2
10-10-2022	Pomocni element 3
20-10-2022	Pauk sklop

POPIS OZNAKA

Oznaka	Mjerna jedinica	Opis oznake
N	N	sila iz podloge
W	N	težina zgloba
W_4	N	težina tijela dok su tri noge podignute
T	Nm	moment zgloba
L	m	udaljenost
Θ	rad	Kut u radijanima
t	h	vrijeme u satima
C_b	mAh	kapacitet u mAh
I_R	mA	potrošnja struje robota
p_{wx}	mm	Udaljenost po x osi
p_{wy}	mm	Udaljenost po y osi
p_{wz}	mm	Udaljenost po z osi
a_2	mm	Stranica a_2
a_3	mm	Stranica a_3
q_2	deg	Kut q_2
q_3	deg	Kut q_3
c_2	deg	Kosinus kuta q_2
c_3	deg	Kosinus kuta q_3
s_2	deg	Sinus kuta q_2
s_3	deg	Sinus kuta q_3
s	m	Udaljenost u metrima
v	m/s	Brzina zvuka u metrima u sekundi
t	s	vrijeme u sekundama
a	mm	Udaljenost središta elipse od krivulje po x osi
b	mm	Udaljenost središta elipse od krivulje po y osi
c	mm	Udaljenost središta elipse od krivulje po z osi
d	mm	Udaljenost središta elipse od ishodišta kordinatnog sustava po x osi
h	mm	Udaljenost središta elipse od ishodišta kordinatnog sustava po y osi
k	mm	Udaljenost središta elipse od ishodišta kordinatnog sustava po z osi
r	mm	polumjer kružncie
a_1	deg	gornja granica kružnice u stupnjevima
a_2	deg	doljnja granica kružnice u stupnjevima
b_1	deg	gornja granica kružnice u stupnjevima
b_2	deg	doljnja granica kružnice u stupnjevima

Sažetak

U ovom radu je prikazano kako napraviti robota. Takvi roboti mogu biti različitog tipa pa je važno napomenuti da je ovo heksapod, odnosno pauk sa šest nogu.

Pokazano je kako izraditi konstrukciju nogu da bude kompatibilna sa motorima, kako povezati zglobove te kako ih povezati sa tijelom.

U drugom dijelu se nalaze proračuni za statički moment nogu kada su tri noge u zraku. Proračunata je inverzna kinematika koja je potrebna za kontrolu noge te je izračunata potrebna struja za robota.

U trećem dijelu je objašnjeno koje mikroprocesore koristiti i zašto njih. Također je objašnjeno kako riješiti problem sa velikim strujama i koje motore koristiti. Kako riješiti problem sa PCA9685 koji ne može dati dovoljno struje svim motorima.

Zadnji dio je kontrola robota. Kod je pisan u Arduino IDE. Objašnjeno je kako povezati više mikroprocesora, kako iskoristiti *websockete* i zašto, te kako izraditi jednostavnu internet stranicu u Javascriptu. Također je objašnjeno kako kontrolirati noge pauka i kako iskoristiti inverznu kinematiku za to.

Ključne riječi: Pauk, robot, robot insekt, robot sa šest nogu, heksapod, arduino, ESP-32CAM, NodeMCU, PCA9685, MG996R, Javascript

Summary

In this thesis, it is shown how to make a Spider robot. Spider robots can be different so it is important to say this is a hexapod, a spider robot with six legs.

The design of the robot is shown how to create legs to be compatible with motors, how to connect different parts of the legs and how to connect them with the body.

In the second part are shown calculations for static torque of the legs when three of the legs are in the air. Inverse kinematics is also shown because it is needed for the control of the robot. And some simple calculations for the current that is needed to power the robot.

The third part is electronics, what microprocessors are used and why them. How to solve the problem with too big currents and what motors are used. How to solve the PCA9685 problem with currents is also addressed.

The last part is control of the robot. The code is written in Arduino IDE. Connecting multiple microprocessors, how to use WebSockets and why, and how to create a simple webpage in Javascript. Details around controlling movements of spider legs and how to use inverse kinematics for it.

Key words: Spider, robot, robot insect, robot with six legs, hexapod, arduino, ESP-32CAM, NodeMCU, PCA9685, MG996R, Javascript

1. UVOD

Priroda je danas veliki izvor ideja za inžinjere. Skoro svakodnevno mogu se vidjeti objave o nekom novom robotu koji ima neki oblik životinje. Roboti u obliku pasa koji služe kao pomoć, ili igračke za djecu, roboti za istraživanje dubina u obliku ribe, hobotnice, jegulje ili neke druge morske životinje, ili u obliku velike mačke da se postignu što veće brzine, ili roboti u obliku ptice koji također lete po zraku kao i original. Inžinjeri gledaju u prirodu zbog velike učinkovitosti živih bića koja se može vidjeti u svim životinjama, npr. mala riba koja postiže velike brzine pod vodom što je gotovo nemoguće za današnje inžinjere, ili sova koja ima veliku snagu da pobjedi pljen te vid koji može dostići i nekoliko kilometara, ili mačka koja može vrlo precizno i daleko skočiti sobzirom na svoje tijelo i mnogi drugi primjeri koji se mogu pronaći.

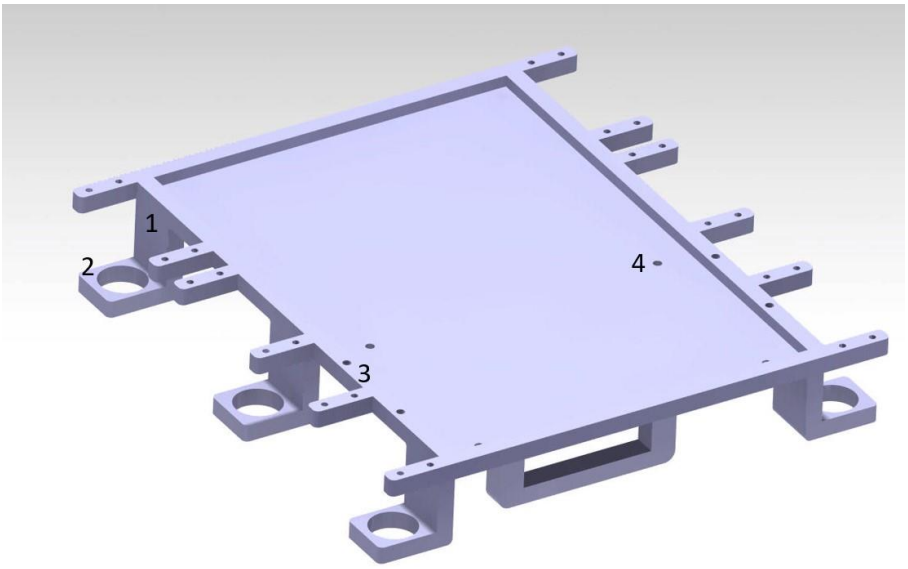
U ovom projektu izabran je pauk kao ideja zbog njegovih zanimljivih nogu. Dok većina bića ima noge koje idu od kukova prema zemlji, pauk ima noge koje idu od kukova na stranu pa tek od koljena idu prema zemlji. Razlog zašto je to zanimljivo je jer s takvom vrstom nogu je jednostavnije napraviti robota, iako ima više nogu i više posla za njihovu izradu, noge pauka riješavaju jedan problem koji se javlja kod većine dvonožnih pa čak i četveronožnih robota, a to je ravnoteža. Iako većina robota danas pobijedi taj problem još uvijek je to zahtjevan zadatak koji zahtjeva puno više rada. Također paukove noge lakše mogu preći preko prepreka te se pritom ne zaplesti i pasti zbog lošeg dizajna ili programiranja. Sama priroda je svojim dizajnom nogu riješila puno problema na koje inžinjeri nailaze oponašajući bilo ljudske ili neke druge slične noge. Ovim se projektom to sve želi pokazati na maloj i jednostavnoj razini.

2. KONSTRUKCIJA

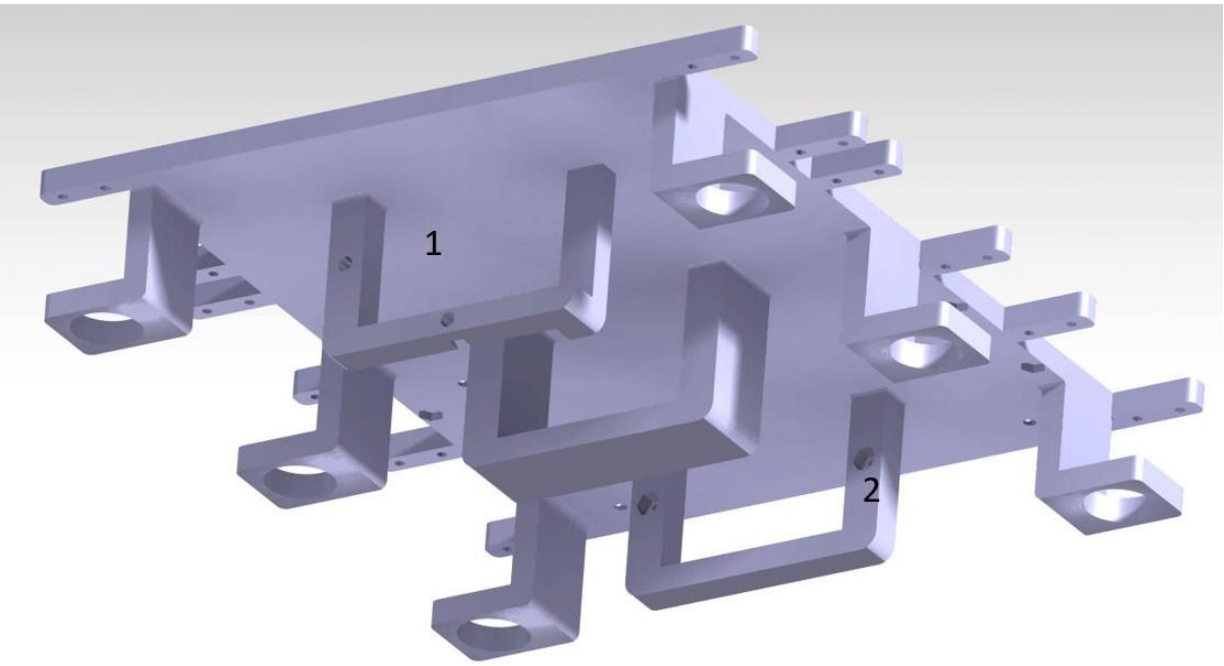
Konstrukcija robota se sastoji od tijela, zglobova te pomoćnih elemenata. Tijelo je glavni dio na koji se spajaju svi ostali dijelovi. Svaka od 6 nogu se sastoji od 3 zgloba koji se spajaju jedan na drugoga. Pomoćni elementi su elementi za držanje senzora i baterije na mjestu.

2.1. Tijelo

To je kao i kod živog pauka glavni dio robota za koji se spajaju noge, te svi elektronički elementi se nalaze na njemu. Sa svake strane nalaze se tri mjesta za motore, svako mjesto ima dvije grede na kojima se nalaze po 2 provrta za vijke. Njima se motor pričvrsti za Tijelo. Također sa doljnje strane nalazi se provrt u koji dolazi ležaj te prostor za bateriju. Zadnja strana za bateriju napravljena je na način da žice iz baterije mogu nesmetano proći. Na okomitim gredama nalaze se provrti za vijke i matice koji služe za povezivanje sa pomoćnim elementom koji spriječava ispadanje baterije. Gornji dio tijela je upušten, tj. ima rub da elektronički elementi i bez ikakvog pričvrćivanje ne odvoje se od Tijela. Prednja strana će se prepoznati po četiri provrta na upuštenom dijelu i četiri provrta na rubu. Provrti na upuštenom dijelu su za vijke koji drže PCB pločicu, a provrti na rubu drže pomoćni element na koji dolazi motor za senzor. Na slikama 2.1 i 2.2 se može vidjeti model Tijela u CATII.



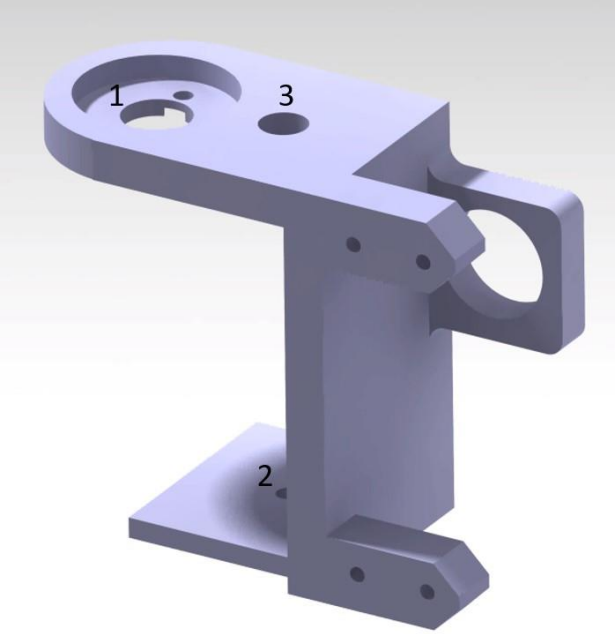
Slika 2.1 Tijelo, 1-Mjesto za motor, 2-provrt za ležaj, 3-provrti za pomoćni element, 4-provrt za držanje PCB pločice



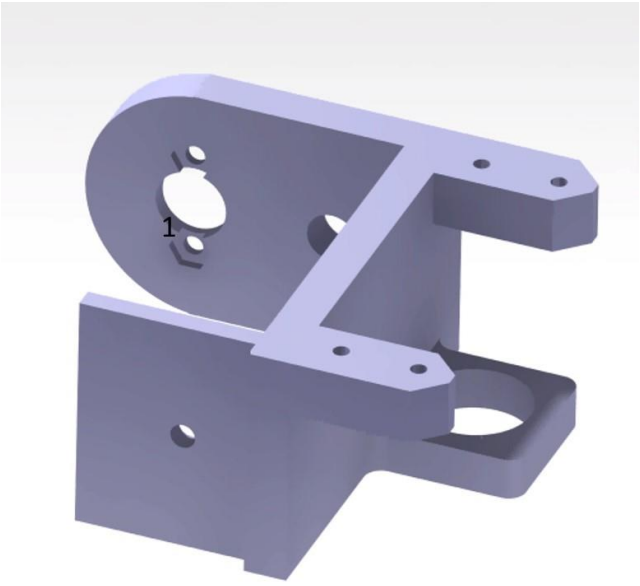
Slika 2.2 Tijelo, 1-Mjesto za bateriju, 2-provrt za vijak i maticu

2.2. Prvi zglob

Prvi zglob je zglob koji se spaja na motor koji je pričvršćen za tijelo. Sa gornje strane ima mjesto za ručicu motora. Ručica motora je element koji se spaja na osovinu motora, te preko tog elementa se sa osovine na idući element prenosi moment. Ručica motora na sebi ima 4 provrta kroz koje prolaze vijci i pričvršćuju ručicu za sljedeći element. Mjesto za ručicu je upušteno te ima dva provrta za vijke. Zbog malih dimenzija ručice odlučeno je da će se ručica i Prvi zglob povezivati samo sa dva vijka da bi se izbjegla dodatna naprezanja i ojačala konstrukcija zgloba. Provrt sa doljnje strane je oblikovan za maticu. Matica se utiskuje u provrt te tako se sprječava njezino okretanje odnosno slučajno odvrtanje te se štedi na prostoru. Ukupna dubina provrta u koji se stavlja ručica je 6 mm, što znači da su potrebni vijci duljine 6 mm da se ručica poveže sa Prvim zglobom. Provrt koji se nalazi prije mjesta za ručicu je provrt za odvijač kojim se olakšava pričvršćivanje vijka. Sa doljnje strane Prvog zgloba dolazi pomoćni element „Držać“. Provrt koji se može vidjeti na slici 2.3 pod brojem 2 služi za vijak koji pričvršćuje Držać i Prvi zglob. Kao i Tijelo, Prvi zglob ima mjesto za motor na kojega će se spajati sljedeći zglob. Pa tako su grede, provrti na njima te provrt za ležaj jednaki kao i na tijelu. Da bi se smanjio moment, udaljenost između mjesta za ručicu i mjesta za sljedeći motor mora biti što manja što je razlog ovakvog dizajna.



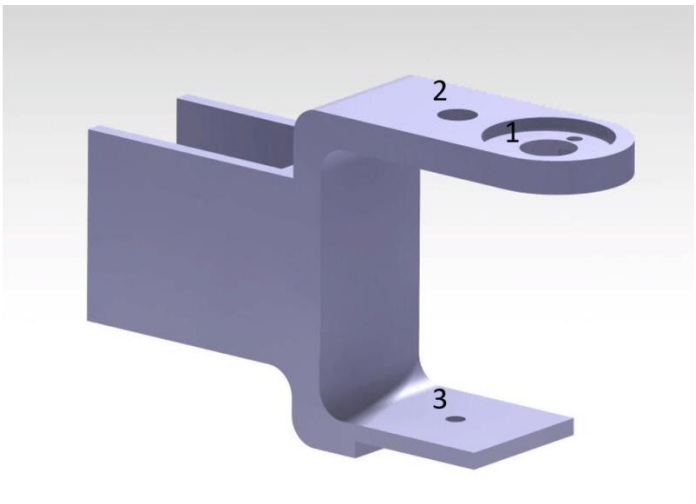
Slika 2.3 Prvi zglob, 1-mjesto za ručicu, 2-provrt povezivanje sa držačem, 3-provrt za odvijač



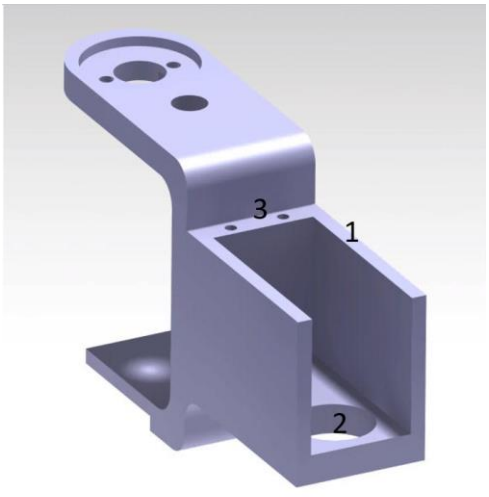
Slika 2.4 Prvi zglob, 1-provrt za matice

2.3. Drugi zglob

Drugi zglob kao i Prvi ima utor za ručicu motora sa gornje strane, a sa doljnje provrt za vijak. Provrt za odvijač se nalazi na istom mjestu kao i kod Prvog zgloba da bi se olakšalo stezanje vijka. No za razliku od Prvog zgloba, Drugi ima nešto veću duljinu jer prilikom okretanja preko ručice motora mora se moći zaokrenuti 180 stupnjeva oko osovine motora. Druga strana ima mjesto za sljedeći motor no tu se motor nalazi između dvije površine. Povezuje se sa dva vijka samo sa jedne strane. Također ispod mjesta za motor se nalazi provrt za ležaj. Drugi zglob se može vidjeti na slikama 2.5 i 2.6



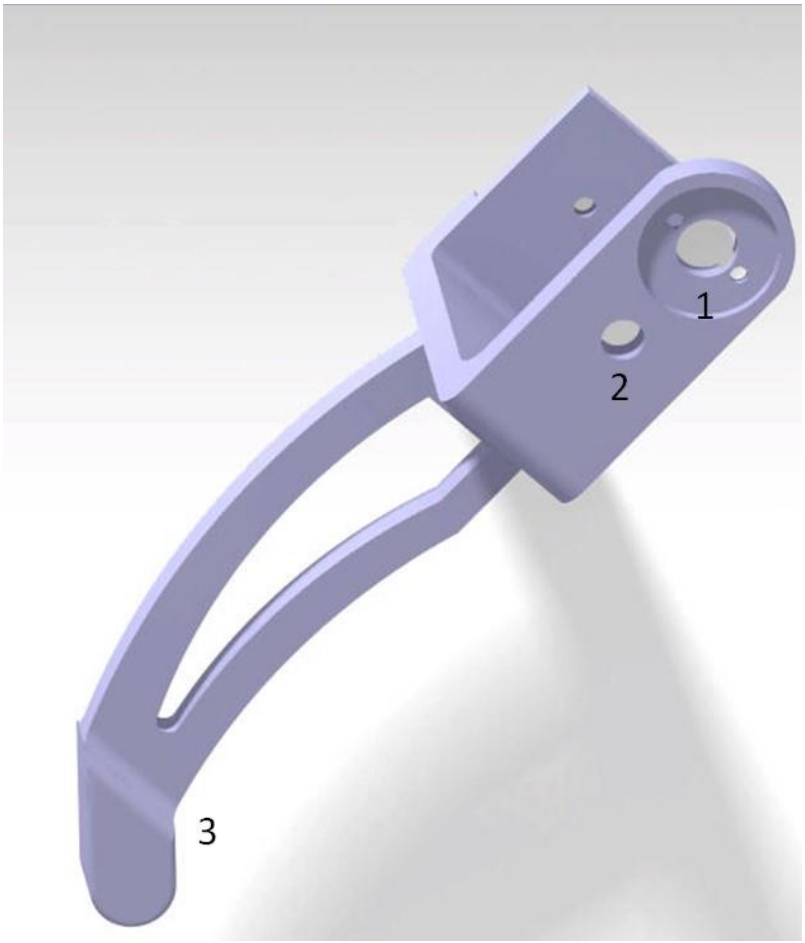
Slika 2.5 Drugi zglob, 1-provrt za ručicu motora, 2-provrt za odvijač, 3-provrt za vijak



Slika 2.6 Drugi zglob, 1-Mjesto za motor, 2-provrt za ležaj, 3-Provrti za vijke koji drže motor

2.4. Noga

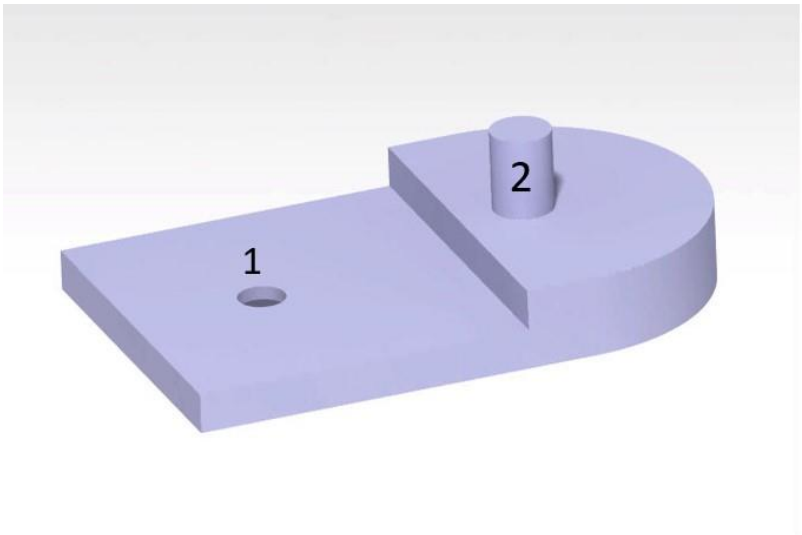
Noga sa gornje strane ima provrt za ručicu i provrt za odvijač kao prethodni zglobovi. Sa doljnje strane nalazi se dio koji dolazi u kontakt sa podlogom. Na mjestu gdje dolazi u kontakt sa podlogom noga je zaobljena da bi mogla stajati pod nekim drugim kutom oko osovine motora, tj. da se ne bi izgubio oslonac ako bi se pauk nagnuo u neku stranu. U sredini prednjeg dijela je izrezan veliki komad materijala da bi se olakšao komad.



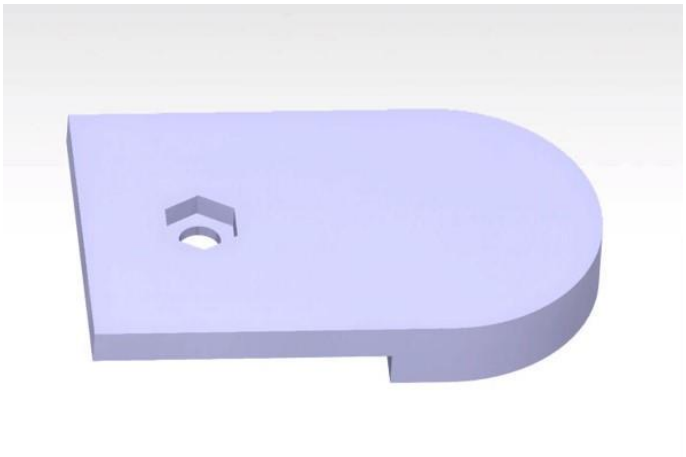
Slika 2.7 Noga, 1-Provrt za ručicu motora, 2-provrt za odvijač, 3-podloga

2.5. Držač

Držač je pomoćni dio koji se vijkom povezuje sa zglobovima i tijelom. To je na zglobovima provrt za vijak ispod provrta za odvijač. Njime se osigurava da se zglob okreće oko jedne osi. On je zapravo osovina sa druge strane. Ima provrt za vijak koji je sa doljnje strane oblikovan za maticu. Te ima valjak koji ulazi u unutarnji promjer ležaja. Držač se može vidjeti na slikama 2.8. i 2.9.



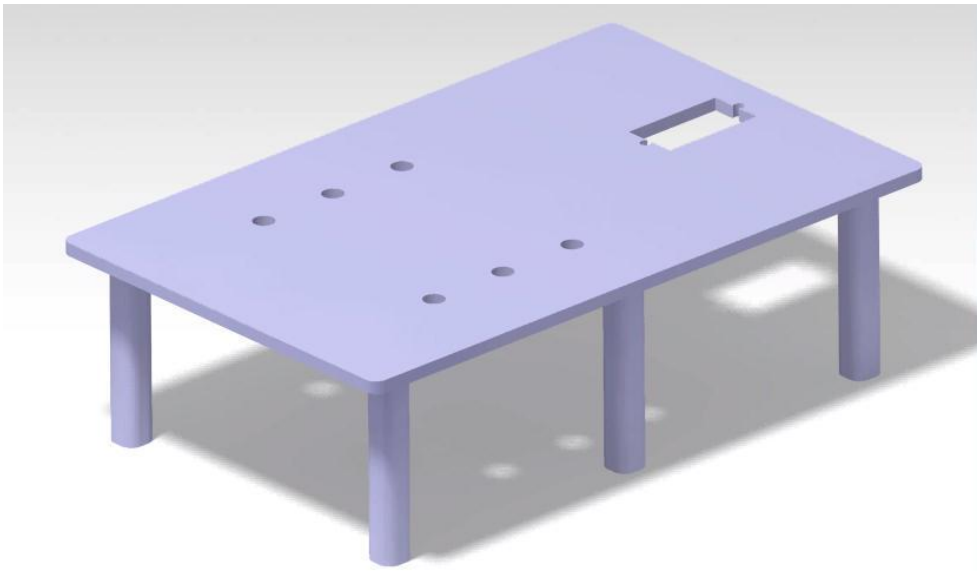
Slika 2.8 Držač, 1-Provrt za vijak, 2-Valjak za unutarnji promjer ležaja



Slika 2.9 Držač, Provrt za maticu

2.6. Pomoćni element 1

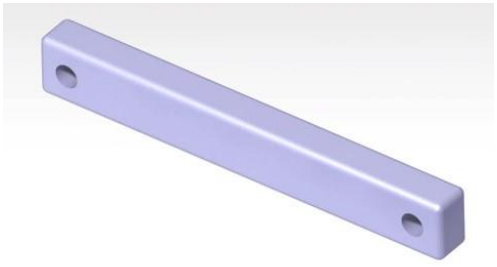
Prvi pomoćni element služi za držanje motora koji će okretati senzor. Tako umjesto stavljanja tri senzora, sa svake strane po jedan, stavit će se samo jedan. Također služi za držanje Razdjelnika. Kroz 6 provrta prolaze mali svornjaci koji drže razdjelnik na mjestu. Pomoćni element 1 se može vidjeti na slici 2.10



Slika 2.10 Pomoćni element 1

2.7. Pomoćni element 2

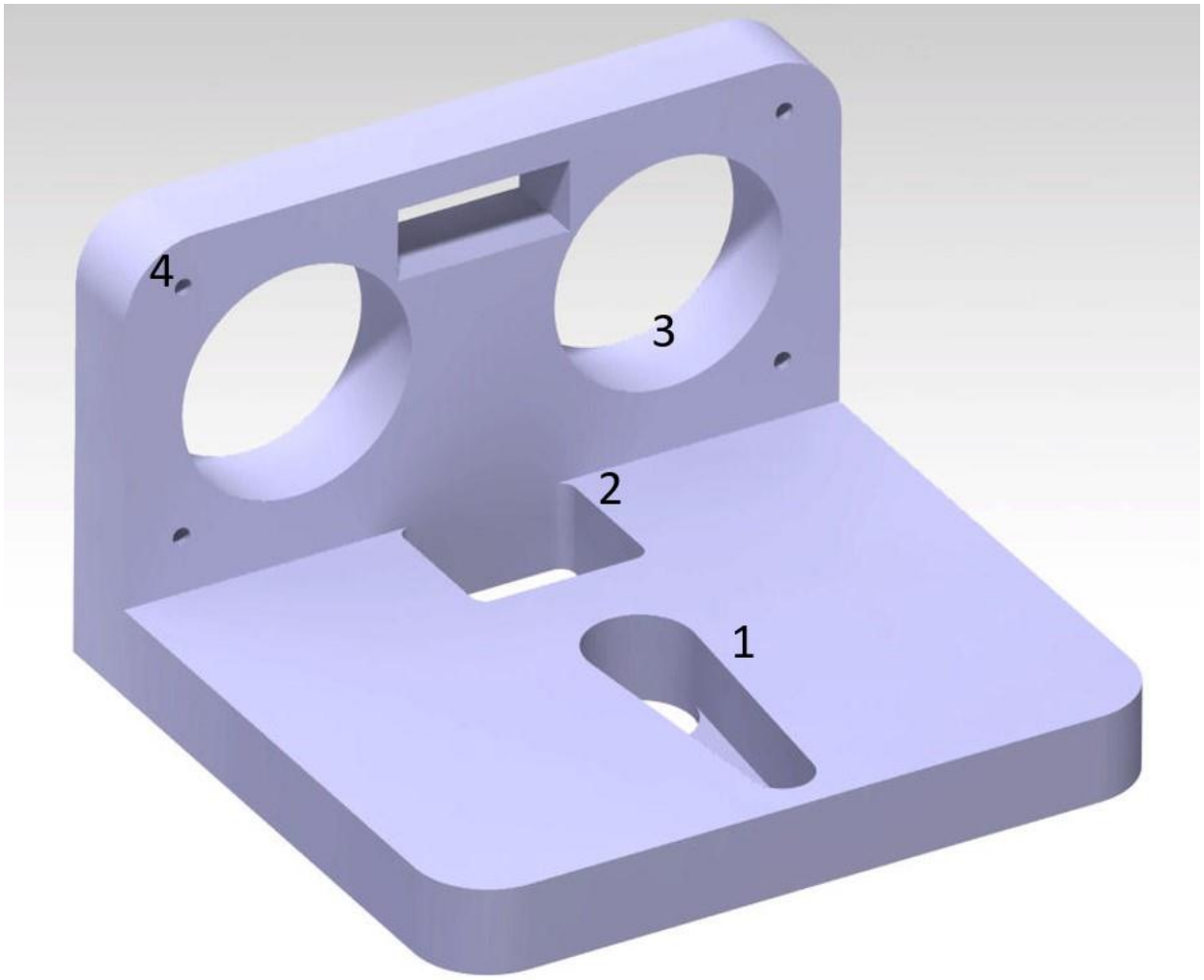
Pomoćni element 2 se stavlja da bi zadržao bateriju na mjestu. Ima dva M3 provrta za pričvršćivanje. Može se vidjeti na slici 2.11.



Slika 2.11 Pomoćni element 2

2.8. Pomoćni element 3

Drugi pomoćni element služi za držanje senzora na motoru. Ovdje je ručica za motor drugačijeg oblika jer je ovdje druga vrsta motora. Na mjestu gdje je veći promjer provrta prolazi osovina motora te se sve okreće oko tog dijela. Za razliku od prijašnjih ručica ovdje nisu potrebni provrti za vijke jer ručica ostaje na mjestu zahvaljujuci svojem obliku te tako i prenosi moment. Ispred provrta za ručicu nalazi se pravokutni provrt kroz koji prolaze kablovi senzora. Na vertikalnom dijelu ima dva okrugla provrta velikog promjera koji su prolazi za zvučnike senzora. Dok se u kutovima nalaze provrti za vijke koji drže senzor za pomoćni element. Pomoćni element se može vidjeti na slici 2.12.



Slika 2.12 Pomoćni element 2, 1-Provrt za ručicu motora, 2-Provrt za kablove senzora, 3-Provrti za zvučnike senzora, 4-provrti za vijke

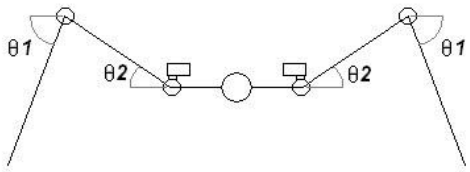
3. PRORAČUN

Potrebno je napraviti dva proračuna. Prvi je proračun momenta noge, da li će robot moći stajati ako su tri noge podignute u zrak. A drugi je proračun potrošnje struje da bi se vidjela kakva je potrebna baterija.

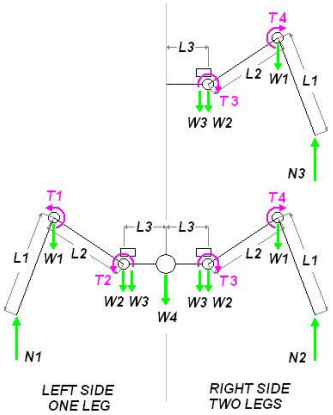
3.1. Proračun momenta nogu

Ovaj proračun je napravljen prema [4]. S obzirom na to da je ovo robot sa 6 nogu potrebno je provjeriti koliki su momenti na zglobovima kada su tri noge u zraku.

Potrebne skice se mogu vidjeti na slikama 3.1 i 3.2.



Slika 3.1 Kutevi između nogu



Slika 3.2 Sile i momenti

Potrebno je napomenuti da je $N_2=N_3$.

Gdje je:

N	N	sila iz podloge
W	N	težina zgloba
W_4	N	težina tijela dok su tri noge podignute
T	Nm	moment zgloba
L	m	udaljenost
Θ	rad	Kut u radijanima

Podaci potrebni za proračun:

L_1=0.155 m

L_2=0.085 m

L_3=0.083 m

W_1=0.8336 kg*m/s_2

W_2=0.8829 kg*m/s_2

W_3=0.8336 kg*m/s_2

W_4=17.4618 kg*m/s_2

Θ_1 =1.57 rad

Θ_2=0.52 rad

Θ_3=0.52 rad

Prva jenadžba prema skicama glasi:

$$N_1 + 2 * N_2 = W_4 + 6 * (W_1 + W_2 + W_3) \tag{3.10}$$

Druga jenadžba za moment lijeve noge glasi:

$$T_{LF} = -W_1 * L_1 * Cos\Theta_1 - W_2 * (L_1 * Cos\Theta_1 + L_2 * Cos\Theta_2) - W_3 * (L_1 * Cos\Theta_1 + L_2 * Cos\Theta_2) - W_4 * (L_1 * Cos\Theta_1 + L_2 * Cos\Theta_2 + L_3) - 2 * W_3 * (2 * L_3 + L_1 * Cos\Theta_1 + L_2 * Cos\Theta_2) - 2 * W_2 * (2 * L_3 + L_1 * Cos\Theta_1 + L_2 * Cos\Theta_2) - 2 * W_1 * (2 * L_3 + L_1 * Cos\Theta_1 + L_2 * Cos\Theta_2) + 2 * N_2 * (2 * L_3 + 2 * L_1 * Cos\Theta_1 + 2 * L_2 * Cos\Theta_2) \tag{3.11}$$

Treća jenažba za moment koljena glasi:

$$T_{Knee} = T_1 - N_1 * L_1 * Cos\Theta_1 - W_2 * L_2 * Cos\Theta_2 - W_3 * L_1 * Cos\Theta_1 - W_4 * (L_2 * Cos\Theta_2 + L_3) - 2 * W_3 * (L_2 * Cos\Theta_2 + 2 * L_3) - 2 * W_2 * (L_2 * Cos\Theta_2 + 2 * L_3) - 2 * W_1 * (2 * L_2 * Cos\Theta_2 + 2 * L_3) + 2 * N_2 * (2 * L_2 * Cos\Theta_2 + 2 * L_3 + L_1 * Cos\Theta_1) \tag{3.12}$$

Četvrta jenadžba za moment kuka glasi:

$$T_{Hip} = T_2 - N_1 * (L_1 * Cos\Theta_1 + L_2 * Cos\Theta_2) + W_2 * L_2 * Cos\Theta_2 - W_4 * L_3 - 2 * W_2 * L_3 - 2 * W_3 * L_3 - 2 * W_1 * (2 * L_3 + L_2 * Cos\Theta_2) + 2 * N_2 * (L_2 * Cos\Theta_2 + 2 * L_3 + L_1 * Cos\Theta_1) \tag{3.13}$$

Uvrštavanjem brojeva u (3.11) dobiva se:

$$N_2 = 6.7\text{N} \tag{3.14}$$

Uvrštavanjem (3.14) u (3.10) dobiva se:

$$N_1 = 19.35\text{N} \tag{3.15}$$

Uvrštavanjem brojeva te (3.15) u (3.12) dobiva se:

$$T_1 = 0.02\text{Nm} \tag{3.16}$$

Uvrštavanjem (3.14) i (3.15) u (3.13) dobiva se:

$$T_2 = 0.3\text{Nm} \tag{3.17}$$

Izabrani motor koji mora podnesti ovaj moment je MG996R, a njegov moment je 1 Nm.

Pa faktor sigurnosti S pokazuje da moment motora je zadovoljavajuć.

$$S = \frac{T_M}{T_2} = \frac{1}{0.3} = 3.3 \tag{3.18}$$

3.2. Proračun potrebne struje

Tablica 3.1 Popis dijelova za izračun potrošnje struje

Ime	Potrošnja struje po kom (mA)	Broj komada	Ukupna potrošnja (mA)
NodeMCU	200	1	200
ESP 32 Cam	200	1	200
MG996R	900	18	16200
Sg90	250	1	250
HC-SR04	15	1	15

Ukupna potrošnja:

$$U_p = 500 + 500 + 16200 + 250 + 15 = 17465\ \text{mA}$$

Prema izračunatoj potrošnji potrebna je struja minimalno 18A te što veći kapacitet da bi baterija duže trajala. Prema proračunu izabrana baterija je: Goldbat Lipo 3S 5200mAh 80C, slika 3.3. Ovakvom baterijom se osigurava da u bilo kojome trenutku robot može tražiti 18A struje i da se pritom baterija ne zagrije. Struja od 18A je teoretska maksimalna struja koju robot može tražiti ali u većini slučajeva tražena struja je između 10 i 12A. Prema tome ako se želi znati vrijeme koje će biti potrebno da se baterija isprazni koristi se formula:

$$t = \frac{C_b}{I_R} = \frac{5200}{12000} = 0.43\text{h}$$

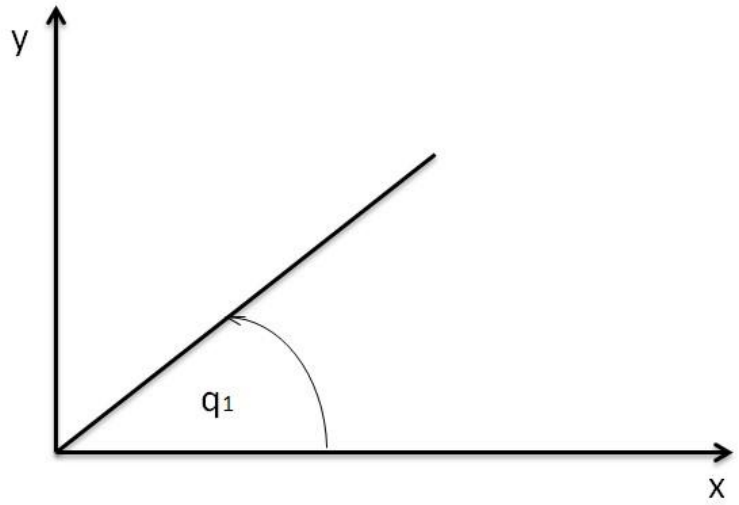
Gdje je:

t	h	vrijeme u satima
C_b	mAh	kapacitet u mAh
I_R	mA	potrošnja struje robota

Slika 3.3 Goldbat Lipo 3S baterija [3]

3.3. Inverzna kinematika

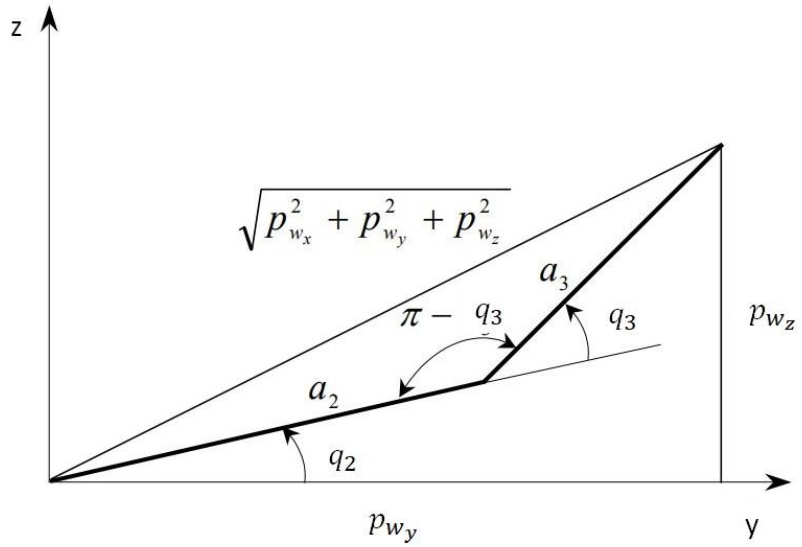
Inverzna kinematika je napravljena prema [10.],



Slika 3.4 Skica za XY ravninu

Sa slike 3.4 se dolazi do prve jenadžbe za q_1 koja glasi:

$$q_1 = \text{Atan2}\left(\frac{y}{x}\right) \tag{3.19}$$



Slika 3.5 Skica za YZ ravninu

Gdje je:

p_{wx}	mm	Udaljenost po x osi
p_{wy}	mm	Udaljenost po y osi
p_{wz}	mm	Udaljenost po z osi
a_2	mm	Stranica a_2
a_3	mm	Stranica a_3
q_2	deg	Kut q_2
q_3	deg	Kut q_3
c_2	deg	Kosinus kuta q_2
c_3	deg	Kosinus kuta q_3
s_2	deg	Sinus kuta q_2
s_3	deg	Sinus kuta q_3

Kao što je prikazano na slici 3.5. koristi se kosinusov poučak da se dobije sljedeća jenadžba:

$$p^2{}_{w_x} + p^2{}_{w_y} + p^2{}_{w_z} = a^2{}_2 + a^2{}_3 - 2a_2a_3\cos(\pi - q_3) \tag{3.20}$$

Nakon sređivanja se dobiva:

$$p^2{}_{w_x} + p^2{}_{w_y} + p^2{}_{w_z} = a^2{}_2 + a^2{}_3 - 2a_2a_3 \tag{3. 21}$$

Iz (3.21) se dobivaju sljedeći izrazi:

$$c_3 = \frac{p^2{}_{w_x} + p^2{}_{w_y} + p^2{}_{w_z} - a^2{}_2 + a^2{}_3}{2a_2a_3} \tag{3. 22}$$

$$s_3 = \pm\sqrt{1 - c_3{}^2} \tag{3. 23}$$

Izraz za q_3 se tada dobiva:

$$q_3 = \mathrm{Atan2}(s_3, c_3) \tag{3. 24}$$

Ti izrazi se dalje mogu zapisati na sljedeći način:

$$p_{w_z} = (a_2 + a_3c_3)s_2 + a_3c_2s_3 \tag{3. 25}$$

$$p_{w_z} - (a_2 + a_3c_3)s_2 = a_3c_2s_3 \tag{3. 26}$$

Kvadriranjem lijeve i desne strane jenadžbe (3.26), te sređivanjem izraza se dobiva:

$$s_2{}^2 - \frac{p_{w_z}(a_2 + a_3c_3)s_2}{p^2{}_{w_x} + p^2{}_{w_y} + p^2{}_{w_z}} + \frac{\left(p^2{}_{w_z} - a_3{}^2s_3{}^2\right)}{p^2{}_{w_x} + p^2{}_{w_y} + p^2{}_{w_z}} = 0 \tag{3. 27}$$

Rješavanjem jenadžbe (3.27) se dobivaju sljedeći izrazi:

$$s_2 = \frac{p_{w_z}(a_2 + a_3 c_3) - a_3 s_3 \sqrt{p^2{}_{w_x} + p^2{}_{w_y}}}{p^2{}_{w_x} + p^2{}_{w_y} + p^2{}_{w_z}} \quad (3.\ 28)$$

$$c_2 = \frac{a_2 + a_3 c_3 \sqrt{p^2{}_{w_x} + p^2{}_{w_y}} + a_3 s_3 p_{w_z}}{p^2{}_{w_x} + p^2{}_{w_y} + p^2{}_{w_z}} \quad (3.\ 29)$$

Izraz za q_2 se tada dobiva:

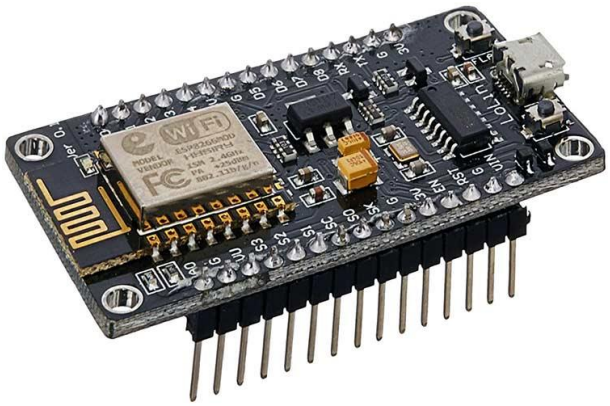
$$q_2 = Atan2(s_2, c_2) \quad (3.3)$$

4. ELEKTRONIKA

Za upravljanje robotom potrebne su nam elektroničke komponente koje će se moći programirati i tako pokretati robota. Kao procesor se koriste NodeMCU te ESP 32 Cam koji će upravljati motorima MG996R, motorom Sg90 te senzorom HC-SR04. Sve to će biti na PCB pločici koja se izrađuje da bi sve bilo kompaktnije i kvalitetnije povezano. Kao izvor se koristi baterija Goldbat Lipo 3S 5200mAh koja daje 11.1V što je više nego što smiju primiti nabrojane komponente, pa će se za smanjivanje napona koristiti pretvarač napona.

4.1. NodeMCU ESP 8266

NodeMCU je mikroprocesor koji se koristi za neke manje, često hobističke projekte. Koristi se zbog jednostavnog programiranja u Arduino IDE te mogućnosti za spajanje preko Wi-Fi. Također ima i veliki broj pinova koji mu daju mogućnost za spajanje više različitih modula, senzora ili drugih elektroičkih komponeti u isto vrijeme. Izabran je u ovom projektu jer programiranjem u Arduino IDE pojednostavljuje upravljanje, mogućnošću spajanja na Wi-Fi omogućuje kontrolu preko interneta, odnoso preko različitih uređaja te ima dovoljno pinova za sve ostale elektroničke komponente. NodeMCU se može vidjeti na slici 4.1.



Slika 4.1 NodeMCU ESP 8266 [5]

4.2. ESP 32 CAM

ESP 32 CAM je drugi mikroprocesor u ovom projektu. Kao i NodeMCU koristi čip ESP 8266, no on na sebi već ima ugrađeu kameru i port za SD karticu što NodeMCU nema. Kao i NodeMCU ima mogućnost povezivanja preko Wi-Fi te se može programirati u Arduino IDE. Zbog ugrađene kamere nema tako veliki broj pinova što smanjuje neke mogućnosti kada se

koristi sam. Iz tog razloga u ovom projektu se koristi u kombinaciji sa NodeMCU, gdje njihovim međusobnim povezivanjem se dobiva sve potrebno za ovaj projekt. ESP 32 CAM je prikazan na slici 4.2.

Slika 4.2 ESP 32 CAM [6]

4.3. Motor MG996R

Za pokretanje zglobova koriste se motori MG996R. Prema proračunu potrebno je barem 0.4 Nm za održavanje, a MG996R ima moment od 1 Nm što ga čini idealnim za ovaj projekt. MG996R je servo motor sa metalnim zupčanikom te mogućnošću okretanja od 0 do 180 stupnjeva. Upravljanje kutom osovine se izvodi preko PWM signala koji se dobiva iz mikroprocesora. Također motor se mora spojiti i na izvor od barem 4.8V. Moguće ga je spojiti i do 6.5V što daje mogućnost na male promjene brzina okretanja ovisno o izabranom izvoru. U ovom projektu svi motori su spojeni na 6V. Motor se može vidjeti na slici 4.3.

Slika 4.3 Motor MG996R [7]

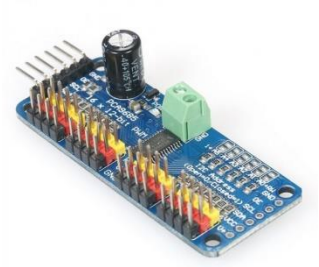
4.4. Motor Sg90

Motor Sg90 je mali motor sa momentom od 0.25Nm. Kao i MG996R može se kretati između 0 i 180 stupnjeva, a upravlja se sa PWM signalom. Zupčanici nisu metalni već su izrađeni od nekog polimera zbog čega je znatno slabiji i brže se troši. Izvor na koji se spaja je 6V, iako se može spojiti na izvor između 4.8 i 6.5V. Potrebna struja za njegovo kretanje je između 100 i 250 mA. U ovom projektu koristi se za okretanje senzora što sa svojim specifikacijma vrlo lako izvodi. Motor Sg90 je prikazan na slici 4.4.

Slika 4.4 Motor Sg90 [8]

4.5. PCA 9685

PCA 9685 je Adafruitov 16 kanalni 12 bitni PWM driver, odnosno to je PCA pločica koja ima 16 izlaza za PWM signal. Na ulazu ima SDA i SCL signal koji se prima od mikroprocesora te ih zatim pretvara u PWM. Ima dva ulaza za napon, jedan napaja samu pločicu dok drugi napaja motore. Svi izlazi za PWM uz sebe imaju i uzemljenje te pin za naponski izvor. Može se priključiti na napone do 6V te podnesti maksimalnu struju od 25mA po pinu. Također moguće je spojiti više PCA9685 zajedno pa se tako može proširiti do 64 izlaza za PWM signal. PCA9685 se može vidjeti na slici 4.5.



Slika 4.5 Adafruitov PWM driver PCA 9685 [11]

4.6. Senzor udaljenosti

Za senzor udaljenosti izabran je HC-SR04 zbog jednostavnog korištenja, male cijene i dobre kvalitete. HC-SR04 je senzor koji mjeri udaljenost tako da šalje zvučni val do prepreke, on se odbija od nje te se vraća do senzora. Nakon što se vrati zna se vrijeme koje je bilo potrebno za taj put te se zna brzina putovanja vala. Iz tih podataka možemo izračunati udaljenost prema formuli za brzinu koja je:

$$v = \frac{s}{t} \tag{4.1}$$

Dok formula za udaljenost je:

$$s = v * t \tag{4.2}$$

Gdje je:

s	m	Udaljenost u metrima
v	m/s	Brzina zvuka u metrima u sekundi
t	s	vrijeme u sekundama

Vrijeme unutar Arduino IDE je u mikrosekundama, a udaljenosti su male pa je jednostavnije izražavati u centimetrima pa tako brzina zvuka koja je 340 *m/s* se pretvara u 0.034 *cm/µs* da bi formula bila ispravna. Ne smije se zaboraviti da val putuje do prepreke i nazad pa da bi se dobila udaljenost do prepreke mora se podjeliti sa 2 kako bi se dobila ispravna udaljenost. Nakon toga formula glasi:

$$s = \frac{0.034 * t}{2} [cm] \tag{4.3}$$

Senzor HC-SR04 ima 4 pina, osim 2 pina za ulaz ima i pinove Trig i Echo. Pin Trig se stavlja u vioko stanje na 10 μs za koje će vrijeme poslati 8 ciklusa zvučnih valova te će biti primljeni od strane Echo pina. Echo pin će na izlazu dati vrijeme u mikrosekundama koje je valu trebalo da napravi taj put. Senzor se može vidjeti na slici 4.6.

Slika 4.6 Senzor HC-SR04 [9]

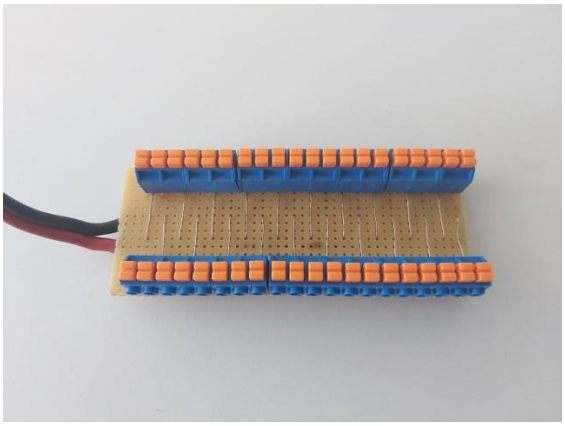
4.7. Pretvarač napona

Pretvarač napona služi za spuštanje napona baterije na potreban napon motora. U ovom slučaju napon baterije je 11.1V, dok za ovaj projekt potreban napon je 6V. Prema specifikacijma ima učinkovitost od 96% kada se napon od 24V spušta na 12V pri struji od 20A. Također na sebi ima dva hladnjaka za bolje hlađenje. Može se vidjeti na slici 4.7.

Slika 4.7 Pretvarač napona [2]

4.8. Razdjelnik

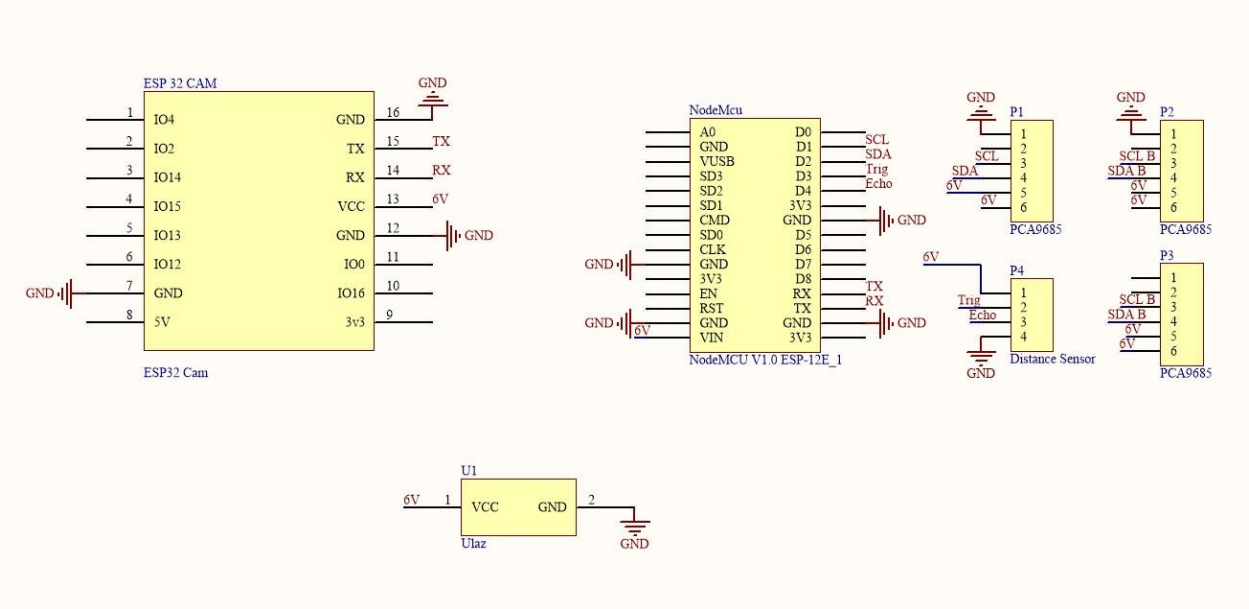
Za ovaj projekt je posebno izrađen da bi se smanjila struja koja prolazi kroz žice. Pa tako kroz sredinu razdjelnika idu dvije velike žice koje podnose struju od 25A, a iz njih idu male žice koje podnose struju od 1A te će na njih biti spojeni motori. Može se vidjeti na slici 4.8.



Slika 4.8 Razdjelnik

4.9. PCB pločica

Izrađena je PCB pločica da bi se izbjegle žice te olakšalo spajanje komponenti. Na slici 4.9. može se vidjeti schema pločice. Sa ulaza dolazi 6V te se spaja na sve dijelove kao što je prikazano na slici 4.9. NodeMCU je spojen sa ESP 32 Cam, tako da se spoji RX sa NodeMCU na TX od ESP 32 Cam, te TX sa NodeMCU na RX od ESP 32 Cam. PCA9685 je spojen na NodeMCU, tako da SCL sa NodeMCU spojimo na SCL na PCA9685, te isto tako i sa SDA signalom. PCA9685 pod oznakom P2 je zapravo kraj prvog PCA9685 koji se spaja na isti način sa drugim PCA9685 koji je označen sa P3. Senzor udaljenosti spaja se sa NodeMCU tako da se njegovi pinovi spoje na drugi i treći pin senzora. U ovom projektu to su pinovi D3 i D4 te pin D3 je proglašen TrigPin, a D4 je proglašen EchoPin.



Slika 4.9 Elektronička schema

5. SKLAPANJE

Prema [1], montaža ili sklapanje, jest svaka djelatnost kojoj je cilj spajanje dvaju ili više objekata u cjelinu, određene namjene. Montaža je zastupljena u svim ljudskim djelatnostima, od industrije (građevinarstvo, strojarstvo, elektronika, brodogradnja...), do kućanstva. Budući da je montaža završna aktivnost u proizvodnji, svi propusti, greške i nedostaci prethodnih faza proizvodnje akumuliraju se u njoj. Stupanj složenosti montažnih radova, ovisno o proizvodu (čiji su gabariti raspona od minijaturnih elektroničkih proizvoda do tankera), ishodi različitim izvođenjem montaže. Upravo raznolikost pojavnosti proizvoda i njihove značajke (male količine, veliki obujmi, težine i broj ugradbenih elemenata, složeni geometrijski oblici ugradbenih elemenata...), kao i činjenica da se automatizacija montaže suočava s posebno složenom problematikom zamjene ljudskog rada, zasnovanoga na iznimnim motoričkim, osjetilnim i mentalnim sposobnostima, uvjetuju da se montaža i danas izvodi najčešće ručno, uz korištenje jednostavnoga alata.

Pa tako i u ovom projektu montaža se odvija ručno, a ovdje je opisana na što jednostavniji način te opisani redosljed spajanja je najjednostavniji od mogućih redoslijeda spajanja.

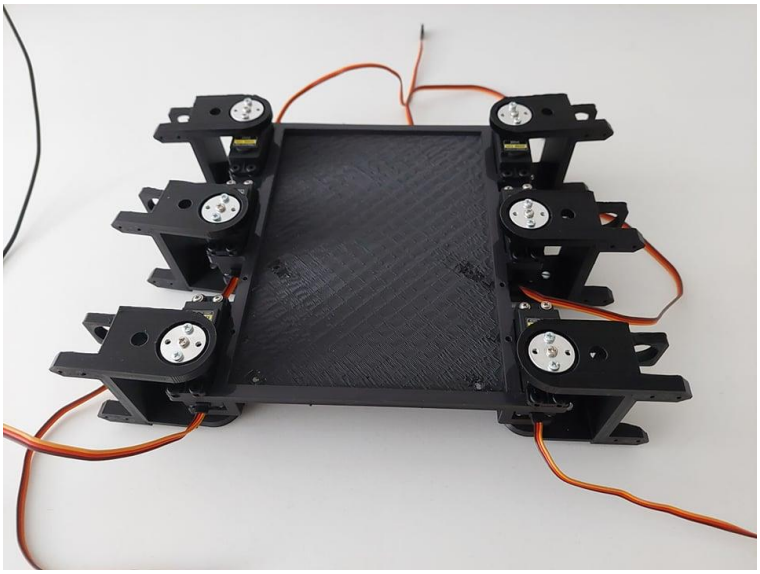
Tablica 5.1 Popis dijelova za sklapanje

Redni broj dijela	Ime	Broj komada
1.	Tijelo	1
2.	Prvi zglob	6
3.	Drugi zglob	6
4.	Noga	6
5.	Držač	18
6.	Pomoćni element 1	1
7.	Pomoćni element 2	1
8.	Pomoćni element 3	1
9.	MG996R	18
10.	Sg90	1
11.	HC-SR04	1
12.	Vijak i matica M3x8	18
13.	Vijak i matica M3x6	18
14.	Vijak i matica M2.5x6	36
15.	Vijak i matica M2.5x10	12
16.	Metalna ručica za MG996R	18
17.	Vijak 2x12 DIN 7982	64
18.	Ležaj 625-2RS	18

Napomena: Prije sklapanja potrebno je motore namjestiti na željeni kut sa Arduino IDE.

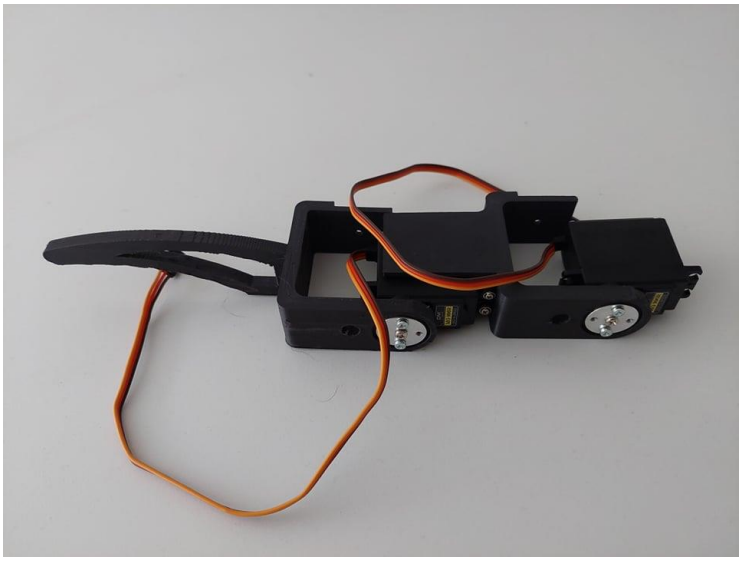
Sklapanje kreće sa (2). Na pripadajuće mjestu stavlja se (16) te pričvršćuje sa (14). Zatim se (9) umeće u (16) te se pričvršćuje sa (13). Kada je motor namješten na željeni kut treba paziti da ga se stavi ispravno sobzirom na taj kut. Tako se olakšava programiranje. Ovaj proces ponoviti za sve (2).

Nakon sklapanja (2) i (9), oni se zajedno umeću na predviđeno mjesto na (1). (2) se orjentira tako da provrt za ležaj gleda prema zadnjoj strani od (1). (1) i (9) se učvršćuju sa (17). Stavlja se (18). Zatim se povezuju (1), (2) i (5), tako da se (5) umetne u (18) te se (5) pričvrsti sa (12) za (1). Ovaj proces se ponavlja za sve (2). Na kraju postupka spoj izgleda kao na slici 5.1.



Slika 5.1 Spoj Prvog zgloba i Tijela

Daljni postupak je sklapanje (3) i (4). Prije sklapanja potrebno je utvrditi koja (4) ide na koju stranu od (1), (4) mora uvijek biti orjentirana kao na slici 5.2. Na odgovarajuće mjesto umeće se (16) u (4) te se pričvršćuje sa (14). Umetanje (9) u (16) te povezivanje sa (13). Stavljanje (18) u (3). Nakon toga se spojeni (4) i (9) stavljaju u (3) te se (3) i (9) povezuju sa (17). Sljedeće je spajanje (3), (4) i (5). Najprije se (5) umeće u (18), a zatim se (5) i (3) pričvršćuju sa (12). Zatim se (16) stavlja u (3) te povezuje sa (14) te se odmah može umetnuti (9) i povezati sa (13). Isti postupak se ponavlja za sve preostale zglobove.



Slika 5.2 Spoj Drugog zgloba i Noge

Zatim slijedi sklapanje spojenih (3) i (4) sa (2). (9) koji se drži za (3) se stavlja u (2) te se pričvršćuje sa (17). Stavlja se (18) u (2). Umetanje (5) u (18) te povezivanje (5) i (2) sa (12). To se ponavlja dok se sve ne sklopi.

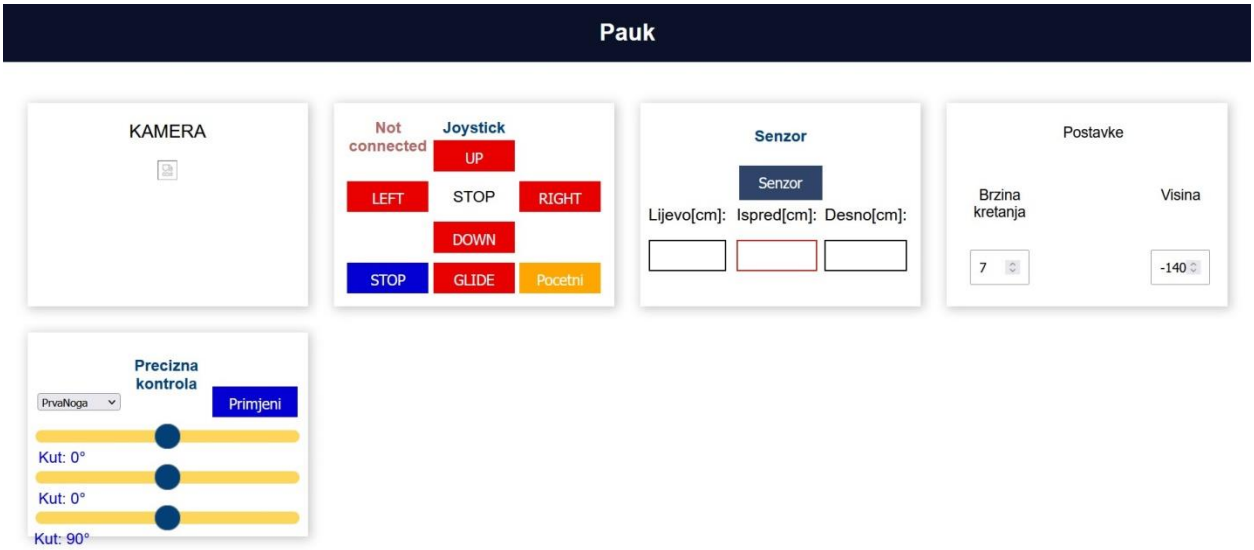
Na kraju (11) se umeće u (6). Ručica koja je došla sa (10) se umeće u (6). Nakon toga se (10) umeće u ručicu i povezuje. Umetanje spojenog (10) u (8). (8) se stavlja na (1) te povezuje sa (15). Nakon stavljanja baterije povezuje se (1) i (7) sa (15) što je kraj sklapanja.

6. SOFTVER I UPRAVLJANJE

Softver je treći veliki dio potreban za rad jednog robota. Bez kvalitetnog softera i dobro izrađene konstrukcije za robota neće biti dobar za rad. No za kvalitetan softver potrebno je puno znanja i iskustva. U ovom projektu napravljen je jedan osnovni softver koji pokriva sve potrebno za rad robota.

6.1. Komunikacija

Komunikacija je najbitniji dio softvera. Da bi korisnik mogao upravljati robotom mora postojati komunikacija između robota i čovjeka. U ovom slučaju je napravljna internet stranica preko koje korisnik može zadavati naredbe robotu. Internet stranica je izrađena sa Javascriptom.

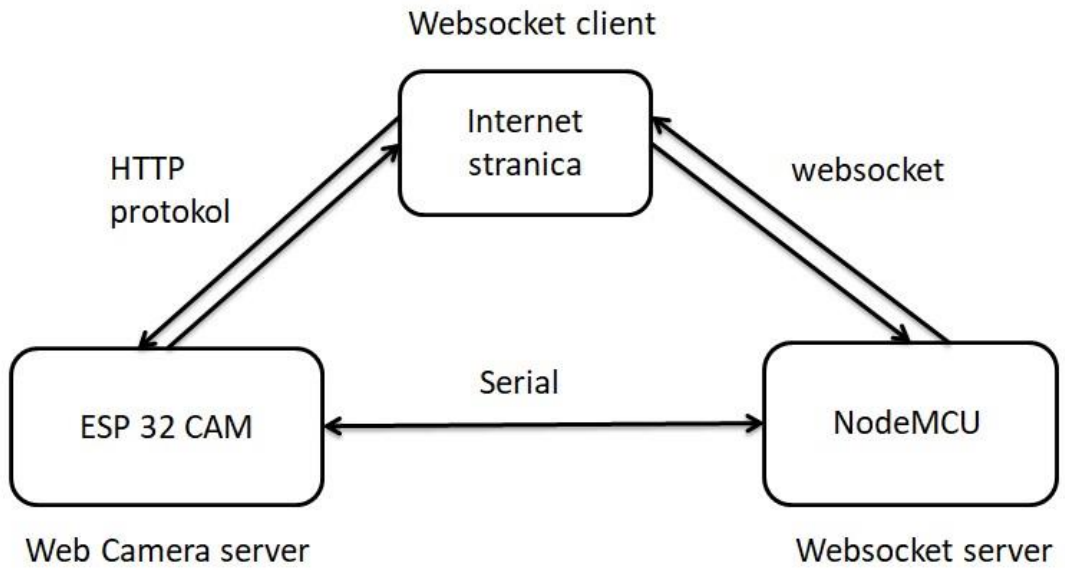


Slika 6.1 Internet stranica

Na slici 6.1. se može vidjeti raspored stranice. U prvom prozoru se nalazi kamera koja se nalazi na prednjoj strani robota. U drugom prozoru se nalazi upravljanje robota. Klikom na gumb se zadaje smjer kretanja robota, ako se stisnu dva pravca, npr. „*UP*“ i „*RIGHT*“ tada će se robot kretati naprijed desno, odnosno po luku će se kretati udesno. Ako se stisne samo gumb za desno, tada će se robot na mjestu okretati oko vlastite osi. Uključivanjem „*GLIDE*“ opcije robot se više neće zakretati već će se translatirati po zadanom pravcu. Sljedeći prozor je napravljen za očitavanje udaljenosti od prepreke koju daje senzor. Prozor „*Postavke*“ služi za upravljanje brzinom kretanja te određivanje udaljenosti tijela od podloge. U zadnjem prozoru se nalazi fina

kontrola robota. Izabere se željena noge te se stisne na gumb „*Primjeni*“, nakon toga je moguće kontrolirati svaki zglob izabrane noge posebno.

Također je važna i komunikacija između elektroničkih komponenti te između komponenti i internet stranice. Ovdje je ostavljena mogućnost komunikacije serijskom vezom između NodeMCU i ESP 32 CAM-a, ali nije potrebna pa se ne koristi. Prilikom uključivanja robota događa se sljedeće. ESP 32 CAM te NodeMCU se povezuju preko unaprijed zadane lozinke i imena na WiFi. ESP 32 CAM stvara server te koristi HTTP protokol da bi podigao stranicu te poslao video koji snima kamera. NodeMCU stvara *websocket* server te čeka da se netko spoji. Kada se internet stranica podigne, pokuša se spojiti preko *websocketa* sa NodeMCU. Tako postoji stalna veza između internet stranice i NodeMCU dok ESP 32 CAM samo šalje video. Međusobna povezanost može se vidjeti na slici 6.2.



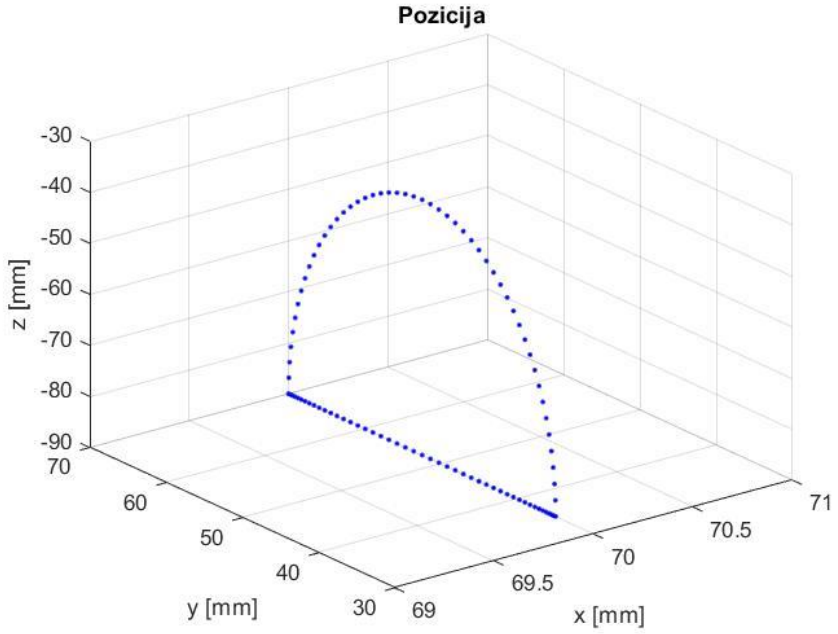
Slika 6.2 Povezanost komponenti

6.2. Kretanje

Da bi se jedan pauk kretao mora se istovremeno kontrolirati 6 nogu. Sobzirom da svaka noga ima 3 stupnja slobode gibanja, cijeli robot ima 18 stupnjeva slobode gibanja. Takav veliki broj stupnjeva slobode gibanja je vrlo kompleksan za kontrolu. Iz toga razloga koristi se inverzna kinematika kojom se dobivaju potrebni kutovi motora. Inverzna kinematika je napravljena za

jednu nogu te se onda promjenom parametara prilagođava i za sve ostale noge. Unošenjem kordinata u inverznu kinematiku dobivaju se kutovi, ali samo za jednu točku u prostoru.

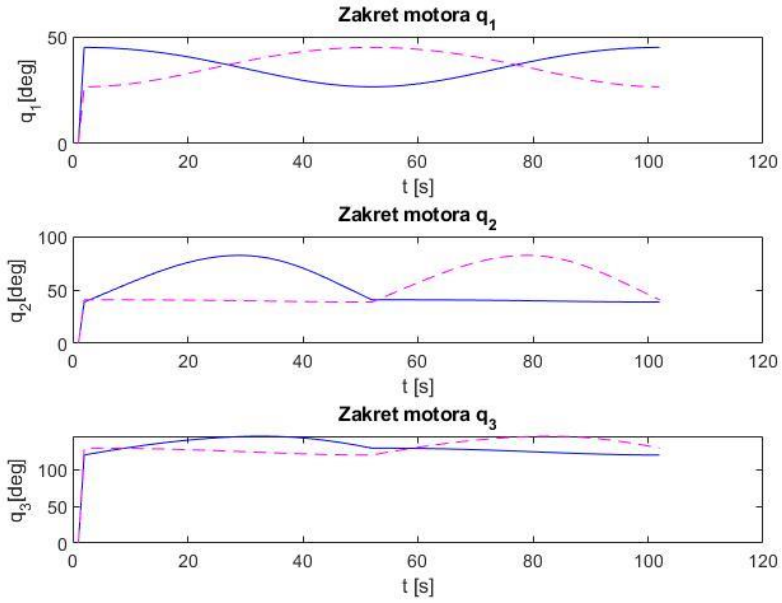
Da bi se robot kretao potrebno je izraditi putanju koju će inverzna kinematika slijediti. Prilikom kretanja robot mora podići nogu u zrak te ju postaviti na neku određenu završnu točku koja se obično nalazi ispred početne točke. Za takvu putanju noge se uzima elipsa.



Slika 6.3 Putanja jedne noge

Nakon što noga dođe u krajnju točku elipse potrebno ju je vratiti u početnu točku. Vraća se po pravcu koji se nalazi ispod krivulje kao što se vidi na slici 6.3. Vraćanje noge po pravcu je zapravo pomicanje tijela unaprijed. Kretanje robota se sastoji od dvije faze, prva je pomicanje noge po elipsi u željenom smjeru, a druga faza je pomicanje tijela. U svakoj fazi se pomiču 3 noge, što znači da su 3 noge u zraku dok su 3 noge na podlozi i pomiču tijelo. Faza traje jedan t.

Na slici 6.3 se može vidjeti pomicanje motora u vremenu.



Slika 6.4 Zakret motora

Da bi se skretalo u stranu potrebno je imati elipsu u „3D“, tj. ravnina u kojoj se nalazi elipsa se mora zakrenuti za određeni kut. Dok pravac postaje kružnica da bi se tijelo zakrenulo u željenu stranu. Kada bi ostao pravac robot bi se kretao u željenu stranu ali bez rotacije, tj. dobila bi se translacija tijela po pravcu, što je i napravljeno u ovom projektu („GLIDE“ opcija). Da bi se tijelo zakretalo oko vlastite osi bez translacije potrebne su kružnice sa središtem u centru tijela. Ako kružnice nisu kocentrične i u središtu tijela dobiti će se neka vrsta translacije.

Parametarske jednadžbe elipse su:

$$x = av * \cos(t * (\frac{\pi}{t_{max}})) + d \tag{6.10}$$

$$y = bv * \cos(t * (\frac{\pi}{t_{max}})) + h \tag{6.11}$$

$$z = cv \cos(t * (\frac{\pi}{t_{max}})) + k \tag{6.12}$$

Gdje su:

$$a = \frac{x_2 - x_1}{2} \tag{6.14}$$

$$b = \frac{y_2 - y_1}{2} \tag{6.15}$$

$$c = \frac{z_2 - z_1}{2} \tag{6.16}$$

$$d = \frac{x_2 + x_1}{2} \tag{6.17}$$

$$h = \frac{y_2 + y_1}{2} \tag{6.18}$$

$$k = \frac{z_2 + z_1}{2} \tag{6.19}$$

a	mm	Udaljenost središta elipse od krivulje po x osi
b	mm	Udaljenost središta elipse od krivulje po y osi
c	mm	Udaljenost središta elipse od krivulje po z osi
d	mm	Udaljenost središta elipse od ishodišta kordinatnog sustava po x osi
h	mm	Udaljenost središta elipse od ishodišta kordinatnog sustava po y osi
k	mm	Udaljenost središta elipse od ishodišta kordinatnog sustava po z osi

Parametarske jednadžbe pravcu su iste kao kod elipse osim z osi koja mora biti ista cijelim pravcem:

$$x = av * \cos\left(t * \left(\frac{\pi}{t_{max}}\right)\right) + d \tag{6.20}$$

$$y = bv * \cos\left(t * \left(\frac{\pi}{t_{max}}\right)\right) + h \tag{6.21}$$

$$z = \frac{z_2 + z_1}{2} \tag{6.22}$$

Parametarske jednadžbe kružnice su sljedeće:

$$x = r * \cos\left(\left(\frac{t}{t_{Maxt}}\right) * \frac{\pi}{2} * a_1 + a_2\right) - x_O \tag{6.23}$$

$$y = r * \sin\left(\left(\frac{t}{t_{Maxt}}\right) * \frac{\pi}{2} * b_1 + b_2\right) - y_O \tag{6.24}$$

$$z = \frac{z_2 + z_1}{2} \tag{6.25}$$

Gdje su:

r	mm	polumjer kružncie
a_1	deg	gornja granica kružnice u stupnjevima
a_2	deg	doljnja granica kružnice u stupnjevima
b_1	deg	gornja granica kružnice u stupnjevima
b_2	deg	doljnja granica kružnice u stupnjevima

Sobzirom da se koristi servo motor koji nema mogućnost provjere pozicije potrebno je pomicati motor u inkrementima, tj. umjesto da se odredi da se pomakne iz početne u završnu točku odjednom, mora se taj put podjeliti na dijelove. Svaki dio puta će trajati neko određeno vrijeme i tako se dobije kontrola brzine motora te precizno slijeđenje krivulje.

7. KOD PROGRAMA

Program je pisan u Arduino IDE, odnosno u C++ jeziku. Programski kod za ovaj projekt je razdvojen u 3 dijela. Prvi dio je kod koji koristi ESP 32 CAM, drugi dio je pisan u Javascriptu te pomoću funkcije prebačen u Arduino IDE, a njime se stvara internet stranica. Treći dio je kod za NodeMCU gdje je glavni dio koda za kontroliranje robota.

7.1. Kod ESP32 CAM-a

Kreće se učitavanjem potrebnih biblioteka te definiranjem varijabli i pinova. Zatim se koristi funkcija *Progmem* koja podiže kod Javascripta na server te stvara internet stranicu prema njemu. Funkcija se otvara sa riječju *rawliteral* nakon koje dolazi zagrada u koju se upisuje kod Javascripta. Nakon što je kod upisan zatvara se zagrada i ponovno se napiše riječ *rawliteral.* Kod za tu fukciju se može vidjeti na slici 7.1.

```
static const char PROGMEM INDEX_HTML[] = R"rawliteral(
<!DOCTYPE html>
<html>
<style>
   html {
```

Slika 7.1 funkcija Progmem – ulaz

```
  console.log("XX"+JSON.stringify(Slider));
}

    </script>
</body>
</html>
)rawliteral";
```

Slika 7.2 funkcija Progmem - izlaz

Funkcija „index_handler“ podiže internet stranicu ako je sve u redu, a funkcija „stream_handler“ šalje video prijenos ako su svi uvjeti zadovoljeni. Funkcije se mogu vidjeti na slici 7.3.

```
static esp_err_t index_handler(httpd_req_t *req){
  httpd_resp_set_type(req, "text/html");
  return httpd_resp_send(req, (const char *)INDEX_HTML, strlen(INDEX_HTML));
}

static esp_err_t stream_handler(httpd_req_t *req){
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
  char * part_buf[64];

  res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
  if(res != ESP_OK){
    return res;
  }


 while(true){
   fb = esp_camera_fb_get();
   if (!fb) {
     Serial.println("Camera capture failed");
     res = ESP_FAIL;
   } else {
     if(fb->width > 400){
       if(fb->format != PIXFORMAT_JPEG){
         bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
         esp_camera_fb_return(fb);
         fb = NULL;
         if(!jpeg_converted){
           Serial.println("JPEG compression failed");
           res = ESP_FAIL;
         }
       } else {
         _jpg_buf_len = fb->len;
         _jpg_buf = fb->buf;
       }
     }
   }
   if(res == ESP_OK){
     size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
     res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
   }
   if(res == ESP_OK){
     res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
   }
   if(res == ESP_OK){
     res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY, strlen(_STREAM_BOUNDARY));
   }
```

Slika 7.3 Funkcije index_handler i stream_handler

Funkcija „Wifi“ povezuje ESP32 CAM bežično sa internetom. Potrebno je definirati ime mreže i lozinku na koju će se ESP spojiti. Nakon što se spoji ispisati će lokalnu IP adresu preko koje se može netko spojiti na njega. Funkcija je ista i za NodeMCU, a može se vidjeti na slici 7.4.

```
void Wifi(){
// Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  Serial.print("Camera Stream Ready! Go to: http://");
  Serial.println(WiFi.localIP());

}
```

Slika 7.4 Funkcija za Wifi

Funkcija „startCameraServer“ definira server, portove te sve potrebne HTTP protokole. Nakon što se podigne server šalje *HTTP GET* protokol da bi primio kod za internet stranicu te prijenos videa. Funkcija „startCameraServer“ se može vidjeti na slici 7.5.

```
void startCameraServer(){
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;
  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = index_handler,
    .user_ctx  = NULL
  };

  httpd_uri_t cmd_uri = {
    .uri       = "/action",
    .method    = HTTP_GET,
    .handler   = cmd_handler,
    .user_ctx  = NULL
  };
  httpd_uri_t stream_uri = {
    .uri       = "/stream",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };
  if (httpd_start(&camera_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(camera_httpd, &index_uri);
    httpd_register_uri_handler(camera_httpd, &cmd_uri);
  }
  config.server_port += 2;
  config.ctrl_port += 2;
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &stream_uri);
  }
}
```

Slika 7.5 Funkcija startCameraServer

Unutar *setupa* se pozivaju sve funkcije, pokreće se kamera te se pokreće server za kameru. Definiraju se pinovi prema preporuci proizviđača. *Setup* se može vidjeti na slici 7.6

```
void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(false);
  Wifi();
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;

  if(psramFound()){
    config.frame_size = FRAMESIZE_VGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

  // Camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
  }
  // Start streaming web server
  startCameraServer();
}
```

Slika 7.6 Setup esp32 Cam-a

7.2. Kod internet stranice

Kod za internet stranicu pisan je u programu Visual Studio Code, jeziku Javascript te nakon toga je prebačen u Arduino IDE kao što je već spomenuto u prošlom poglavlju. Kod Javascripta se sastoji od 3 dijela, koji su *index, script* te *style*.

7.2.1. Index

Index je glavni dio internet stranice, u njega se piše ono što će korisnik kasnije vidjeti na internet stranici ali i definiraju se dijelovi internet stranice koji se kasnije mogu pozvati u *stylu* i *scripti* te koristiti.

Kreće se sa definiranjem stranice sa *DOCTYPE*, zatim se definira naslov stranice što je ovdje „Spider“. Definira se neka opća svojstva te linkovi ako su potrebni. Ovdje su potrebni linkovi za povezivanje sa prijenos videa te povezivanje sa *style* jer u njemu je definiran izgled stranice. Sve se to može vidjeti na slici 7.7.

```
<!DOCTYPE html>
<html>
<head>
    <title>Spider</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/png" href="favicon.png">
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
```

Slika 7.7 Početak stranice

Zatim se definira *body* stranice u kojem se nalazi ono što korisnik vidi. Kao što je prije spomenutu internet stranica je podjeljena u nekoliko prozora. Da bi se to napravilo cijela stranica se stavi u klasu „content“ te zatim podklasu „card-grid“. Tako se kasnije u *stylu* može cijela internet stranica staviti u mrežu, odnosno dobiju se prozori koje je lakše pozicionirati i uređivati. Svaki prozor je klasa pod naziv „card“ te sadrži svoju podklasu za naslov i unutranju mrežu. Kod za prvi prozor se može vidjeti na slici 7.8. Sadrži naslov „Kamera“ te sa *img src* definira da će se tu nalaziti slika ili video, a sa *id* se definira koja slika će biti pozvana. U ovom slučaju postoji samo jedan video pod nazivom „photo“.

```
<body>
    <div class="topnav">
        <h1>Pauk</h1>
    </div>
    <div class="content">
        <div class="card-grid">
            <div class="card">
                <p class="card-title">KAMERA</p>
                <p class="switch">
                    <img src="" id="photo" >
                </p>
            </div>
```

Slika 7.8 Prvi prozor internet stranice

U drugom prozoru se nalazi nekoliko gumbova za kontrolu robota. Gumbove se definira sa *button* te im se pridodaje *id* za raspoznavanje. Svaki gumb može pozivati na nekoliko načina neku funkciju. Fukcije koje Javascript već ima u sebi a služe za pozivanje drugih funkcija su:

onclick, onmousedown, ontouchstart, ontouchstop i druge. *Onmousedown* je funkcija koja poziva drugu funkciju kada se na gumb prtisne mišem dok funkcija *ontouchstart* poziva funkciju kada se gumb pritisne dodirom na ekran. Ako se želi različita funkcija prilikom pritiska na gumb te prilikom otpuštanja gumba potrebno je definirati funkcije za *onmousedown* i *onmouseup.* Ovdje su korištene funkcije *onclick, onmousedown* i *ontouchstart* da bi se robot mogao konotrlirati preko računala ili mobitela. Klasa pod nazivom „Veza“ sa *span id*-om je poveznica da bi se mogao promjeniti tekst „Not connected“ prilikom pozivanja odgovarajuće funkcije. Sve se to može vidjeti na slici 7.9.

```
<div class="card">
        <div class="grid">
            <p class="card-title2"> Joystick</p>
            <p class="Veza"><span id="Spoj">Not connected</span></p>
            <tr><td colspan="3" ><button class="button2" onmousedown="Stop();" ontouchstart="Stop();">STOP</button></td></tr>
            <tr><td colspan="3" ><button id="buttonUp" onclick="moveup()"  ontouchstart="moveup()">UP</button></td></tr>
            <tr><td colspan="3" ><button id="buttonDown" onclick="movedown()"  ontouchstart="movedown()">DOWN</button></td></tr>
            <tr><td colspan="3" ><button id="buttonLeft" onclick="moveleft()"  ontouchstart="moveleft()">LEFT</button></td></tr>
            <tr><td colspan="3" ><button id="buttonRight" onclick="moveright()"  ontouchstart="moveright()">RIGHT</button></td></tr>
            <tr><td colspan="3" ><button id="PocetniPolozaj" onmousedown="Slanje('PocetniPolozaj','PocetniPolozaj')" onmouseup="KlikS
            <tr><td colspan="3" ><button id="GlideButton" onclick="glide()" ontouchstart="glide()">GLIDE</button></td></tr>
            <div class="box4"><span id="Orjentacija">STOP</span></div>
        </div>
</div>
```

Slika 7.9 Drugi prozor internet stranice

U sljedećem prozoru se nalazi gumb za pokretanje senzora. Pritiskom na senzor ispisati će se vrijednosti unutar kućica za svaku stranu. Da bi se to dobilo definira se tri različite klase „box“, svaka sa svojim *id* kako bi ih kasnije mogli urediti po želji. To se može vidjeti na slici 7.10.

```
<div class="card">
    <div class="raspored">
    <p class="card-title3">Senzor</p>
        <tr><td colspan="3" align="center"><button id="button1" onmousedown="getValues();" >Senzor</button></td></tr>
        <div class="state">Lijevo[cm]:<br><p class="box1"><span id="senzor1"></span></p><br></div>
        <div class="state2">Ispred[cm]:<br><p class="box2"> <span id="senzor2"></span></p></br></div>
        <div class="state3">Desno[cm]:<br><p class="box3">  <span id="senzor3"></span></p></br></div>

    </div>
</div>
```

Slika 7.10 Treći prozor internet stranice

Prozor pod nazivom „Postavke“ sadrži dva gumba koji imaju mogućnost da im se promjeni vrijednost. Da bi se dobili takvi gumbovi mora se zadati tip *number* opet sa pripadajućim *id-om.* Ovakav gumb se definira preko funkcije *onchange* koja poziva neku drugu funkciju kada se nešto promjeni, odnosno u ovom slučaju kada se promjeni vrijednost gumba. Mora se zadati vrijednost *value* koja definira početnu vrijednost prilikom otvaranja internet stranice, minimalna te maksimalna vrijednost na koju korisnik može staviti gumb te korak promjene vrijednosti. To se može vidjeti na slici 7.11.

```
        <div class="card">
            <div class="grid2">
                <div class="GlavniNaslov">Postavke</div>
                <div class="Naslov">Brzina kretanja</div>
                <div class="Naslov2">Visina</div>
                <input type="number" id="delay" onchange="Pauza()" value = "7" min="1" max="10" step="0.5">
                <input type="number" id="visina" onchange="Visina()" value = "-140" min="-160" max="-100" step="10">
            </div>
    </div>
```

Slika 7.11 Četvrti prozor internet stranice

U zadnjem prozoru se nalazi jedan padajući izbornik, jedan gumb te tri klizača. Da bi se dobio padajući izbornik potrebno je definirati klasu te *id* pod kojim se onda definira vrijednost svake opcije. Opcije su zapravo ono što se može izabrati u padajućem izborniku. Tako je ovdje pod svaku opciju stavljeno ime jedne noge. Kada se izabere željena noga pritišće se gumb „Primjeni“ koji poziva funkciju istog imena „Primjeni“. Klizač se mora definirati tako da se pod *input type* upiše *range*. Da bi se promjenila vrijednost klizača te upotrijebila poziva se funkcija preko *onchange* funkcije. Također je potrebno definirati maksimalnu, minimalnu i početnu vrijednost te korak promjene. Ispod klizača se stavlja nova klasa „state“ sa *span id* da bi se mogla vidjeti vrijednost klizča prilikom njegovog korištenja. Kod za zadnji prozor se može vidjeti na slici 7.12.

```
<div class="card">
    <div class="grid3">
        <p class="card-title4">Precizna kontrola</p>
        <div class="custom-select">
            <select id = "Odabir">
              <option value="1">PrvaNoga</option>
              <option value="2">DrugaNoga</option>
              <option value="3">TrecaNoga</option>
              <option value="4">CetvrtaNoga</option>
              <option value="5">PetaNoga</option>
              <option value="6">SestaNoga</option>
            </select>
        </div>
        <tr><td colspan="3" ><button class="Primjeni" onmousedown="Primjeni();" ontouchstart="Primjeni();">Primjeni</button></td></tr>
        <div class="sliderCon1"><input type="range" onchange="updateSliderPWM(this)" min="-90" max="90" step="1" value="0" class="slider" id="Value1">
            <p class="state10">Kut: <span id="sliderValue1">0</span>&deg;</p>
        </div>
        <div class="sliderCon2"><input type="range" onchange="updateSliderPWM(this)" min="-90" max="90" step="1" value="0" class="slider" id="Value2">
            <p class="state10">Kut: <span id="sliderValue2">0</span>&deg;</p>
        </div>
        <div class="sliderCon3"><input type="range" onchange="updateSliderPWM(this)" min="0" max="180" step="1" value="90" class="slider" id="Value3">
            <p class="state10">Kut: <span id="sliderValue3">90</span>&deg;</p>
        </div>
    </div>
</div>
</div>
```

Slika 7.12 Peti prozor internet stranice

7.2.2. Script

U početku skripte se nalaze sve potrebne varijable. Gumbovi su zadani kao *boolean* jer za njih je samo potrebno znati da li su pritisnuti ili ne. Definirani su i dva *arraya* u koje se spremaju vrijednosti određenih varijabli. Sve ostale varijable su zadane ili kao *String* ili neka vrsta broja. To se može vidjeti na slici 7.13.

```
var gateway = 'ws://'+'192.168.147.144'+':81/';
var websocket;
var Noga = "PrvaNoga";
let buttonUp = Boolean(false);
let buttonDown = Boolean(false);
let buttonRight = Boolean(false);
let buttonLeft = Boolean(false);
let GlideButton = Boolean(false);
var Orjentacija = "STOP"
var Glide = "OFF"
var Delay = 7;
var Vis = -140;
var Kordinate = {Delay:"0", Z1:"-140", Orjentacija:"STOP", Glide:"OFF"};
var Slider = {Orjentacija:"STOP", BrojSlidera:"0", VrijednostSlidera:"0"};
```

Slika 7.13 Varijable potrebne u skripti internet stranice

Funkcija „onOpen“ javlja da je došlo do povezivanja te mijenja tekst drugog prozora iz „Not connected“ u „Connected“. Dok „onClose“ izvodi obrnute operacije te pokušava ponovno uspostaviti vezu sa pozivanjem funkcije „initWebSocket“ svakih dvije sekunde. Funkcija „initWebSocket“ se pokreće prilikom pokretanja internet stranice zahvaljujući fukciji „onload“. „initWebSocket“ pokreće uspostavljanje veze sa zadanom adresom preko *websocketa*. Veza će biti uspostavljena jedino ako su obje strane povezane na isti „Wifi“ te druga strana je uspjela podići *websocket* server. Također funkcija „initWebSocket“ poziva funkciju „onOpen“ kada se uspostavi veza sa *websoketom*, „onClose“ kadad se zatvori veza te „onMessage“ kada dođe poruka preko *websocketa*. Sve se te funkcije mogu vidjeti na slici 7.14.

```
function onload(event) {
    initWebSocket();
}

function initWebSocket() {
    console.log('Trying to open a WebSocket connection…');
    websocket = new WebSocket(gateway);
    websocket.onopen = onOpen;
    websocket.onclose = onClose;
    websocket.onmessage = onMessage;
}

function onOpen(event) {
    console.log('Connection opened');
    document.getElementById("Spoj").innerHTML = "Connected";
    document.getElementById("Spoj").style.color = '#00df25af';
}

function onClose(event) {
    console.log('Connection closed');
    document.getElementById("Spoj").innerHTML = "Not connected";
    document.getElementById("Spoj").style.color = '#7e000098';
    setTimeout(initWebSocket, 2000);
}
```

Slika 7.14 Funkcija initWebSocket

Funkcija „onMessage“ ima samo jedan zadatak u ovom projektu jer ništa drugo ne može doći osim vrijednosti senzora. Pa tako ona otvara paket te sprema vrijednosti u odgovarajuće varijable i ispisuje te vrijednosti senzora u već prije spomenute kućice ili „box“-ove. Funkcija „getValues“ šalje riječ koju će druga strana prepoznati preko *websocketa.* Funkcija „Stop“ se pokreće kada se pritisne gumb stop. Funkcija šalje preko *websocketa* robotu da se zaustavi, a gumbove koji su aktivni stavlja u neaktivno stanje. Kod se može vidjeti na slici 7.15.

```
function getValues(){
    websocket.send("Values");
}

function Stop(){
    slanjeJson(Delay,"STOP",Glide);
    buttonUp = false;
    color(buttonUp,'buttonUp');
    buttonDown = false;
    color(buttonDown,'buttonDown');
    buttonRight = false;
    color(buttonRight,'buttonRight');
    buttonLeft = false;
    color(buttonLeft,'buttonLeft');


}

function onMessage(event) {
    console.log(event.data);
    const Senzor = JSON.parse(event.data);
    console.log(Senzor);
    console.log(Senzor.Lijevo);
    document.getElementById("senzor1").innerHTML = Senzor.Lijevo;
    document.getElementById("senzor2").innerHTML = Senzor.Desno;
    document.getElementById("senzor3").innerHTML = Senzor.Ispred;
}
```

Slika 7.15 Funkcije stop, getValues i onMessage

Sljedeće funkcije su funkcije koje upravljaju gumbovima. Funkcija „KlikStop“ vraća boju gumba u početno zadanu boju. Dok „moveup“, „movedown“, „moveleft“ i „moveright“ mijenjaju stanje gumba, ako je *True* onda postane *False* i obrnuto, pozivaju funkciju „pisanje“ te mijenaju boju gumba pozivanjem funkcije „color“. „Glide“ radi istu stvar ali mora i provjeriti u kojem je stanju gumb i na temelju toga varijabli „Glide“ dati odgovarajuću vrijednost. To se može vidjeti na slici 7.16.

```
function KlikStop(id_buttona){
    document.getElementById(id_buttona).style.backgroundColor = '';
}

function moveup() {
    buttonUp = !buttonUp;
    pisanje();
    color(buttonUp,'buttonUp');
}

function movedown() {
    buttonDown = !buttonDown;
    pisanje();
    color(buttonDown,'buttonDown');
}

function moveleft() {
    buttonLeft = !buttonLeft;
    pisanje();
    color(buttonLeft,'buttonLeft');
}

function moveright() {
    buttonRight = !buttonRight;
    pisanje();
    color(buttonRight,'buttonRight');
}

function glide() {
    GlideButton = !GlideButton;
    if (GlideButton == false){
        Glide = "OFF";
    }
    else{
        Glide = "ON";
    }
    color(GlideButton,'GlideButton');
    console.log(Glide);
}
```

Slika 7.16 Funkcije za kontrolu gumbova

Funkcija „pisanje“ provjerava koji gumb je pritisnut te na temelju toga šalje preko *websocketa* ključne riječi funkcijom „slanjeJson“. Kod se može vidjeti na slici 7.17.

```
function pisanje(){
if (buttonUp == true && buttonDown==false && buttonLeft == false && buttonRight==false){
    slanjeJson(Delay,"Naprijed",Glide,Vis)
}

else if (buttonUp == true && buttonDown==false && buttonLeft == false && buttonRight==true){
    slanjeJson(Delay,"Lagano desno",Glide,Vis)
}

else if (buttonUp == true && buttonDown==false && buttonLeft == true && buttonRight==false){
    slanjeJson(Delay,"Lagano lijevo",Glide,Vis)
}

else if (buttonUp == false && buttonDown==true && buttonLeft == false && buttonRight==false){
    slanjeJson(Delay,"Nazad",Glide,Vis)
}

else if (buttonUp == false && buttonDown==true && buttonLeft == false && buttonRight==true){
    slanjeJson(Delay,"Nazad desno",Glide,Vis)
}

else if (buttonUp == false && buttonDown==true && buttonLeft == true && buttonRight==false){
    slanjeJson(Delay,"Nazad lijevo",Glide,Vis)
}

else if (buttonUp == false && buttonDown==false && buttonLeft == false && buttonRight==true){
    slanjeJson(Delay,"Strogo desno",Glide,Vis)
}

else if (buttonUp == false && buttonDown==false && buttonLeft == true && buttonRight==false){
    slanjeJson(Delay,"Strogo lijevo",Glide,Vis)
}

else if (buttonUp == false && buttonDown==false && buttonLeft == false && buttonRight==false){
    slanjeJson(Delay,"STOP",Glide,Vis)
}
}
```

Slika 7.17 funkcija pisanje

Funkcija „color“ provjerava u kojem je stanju gumb te na temelju toga mijenja boju gumba. „Pauza“ i „Visina“ spremaju vrijednost gumba u varijablu. „SlanjeJson“ prima vrijednosti varijabli te ih sprema u već napravljeni *array* i šalje preko *websocketa*. Zadnja linija koda unutar funkcije „SlanjeJson“ mijenja tekst u sredini drugog prozora. To je tekst koji govori u kojem se pravcu kreće robot, tj. koji gumbovi su pritisnuti. Te se funkcije mogu vidjeti na slici 7.18.

```
function color(button, id_buttona){

    if (button == true){
        document.getElementById(id_buttona).style.backgroundColor = '#00ff37';
    }
    else{
        document.getElementById(id_buttona).style.backgroundColor = '#e90101';
    }
}

function Pauza(){
    Delay = document.getElementById("delay").value;
    console.log(Delay);
}

function Visina(){
    Vis = document.getElementById("visina").value;
    console.log(Vis);
}

function slanjeJson(x, Orjentacija,Glide,Vis){
    Kordinate['Delay'] = x;
    Kordinate['Z1'] = Vis;
    Kordinate['Orjentacija'] = Orjentacija;
    Kordinate['Glide'] = Glide;
    websocket.send("4a"+JSON.stringify(Kordinate));
    console.log("4a"+JSON.stringify(Kordinate));
    document.getElementById("Orjentacija").innerHTML = Orjentacija;

}
```

Slika 7.18 Funkcije Pauza, Visina, color i slanjeJson

Funkcija „getSelectedText“ pretvara *id* elementa u tekst koji se može spremiti u varijablu. Upravo to se radi sa „Primjeni“. „UpdateSliderPWM“ uzima broj klizača i njegovu vrijednost te ispisuje vrijednost klizača na predviđenom mjestu koje se dobije preko *id* mjesta. Također šalje vrijednost klizača, broj klizača te broj noge preko funkcije „SlanjeSliderVrijednosti“ NodeMCU. „SlanjeSliderVrijednosti“ prima vrijednosti klizača, broj klizača te broj noge i šalje ih *websocketom*. Kod tih funkcija se može vidjeti na slici 7.19.

```
function getSelectedText(elementId){
    var elt = document.getElementById(elementId);

    if (elt.selectedIndex == -1)
        return null;

    return elt.options[elt.selectedIndex].text;
}

function Primjeni(){
    Noga = getSelectedText('Odabir');
    console.log(Noga);
}

function updateSliderPWM(element) {
    var sliderNumber = element.id.charAt(element.id.length-1);
    var sliderValue = document.getElementById(element.id).value;
    document.getElementById("sliderValue"+sliderNumber).innerHTML = sliderValue;
    SlanjeSliderVrijednosti(Noga, sliderNumber, sliderValue);
    console.log(Noga, sliderNumber, sliderValue);
}

function SlanjeSliderVrijednosti(Leg, BrojSlidera, VrijednostSlidera){
    Slider['Orjentacija']=Leg;
    Slider['BrojSlidera']=BrojSlidera;
    Slider['VrijednostSlidera']=VrijednostSlidera;
    websocket.send("XX"+JSON.stringify(Slider));
    console.log("XX"+JSON.stringify(Slider));
}
```

Slika 7.19 Funkcije getSelectedText, Primjeni, updateSliderPWM te SlanjeSliderVrijednosti

7.2.3. Style

Unutar *style*-a se uređuje izgled i pozicija pojeding elementa internet stranice. Kreće se od općeg izgleda cijele stranice sa *html*. Sa *font-family* se uredi izgled teksta te sa *text-align* se tekst stranice poravna da bude u sredini. Sa *display* se mogu dodati neka svojstva izgleda. Kod se može vidjeti na slici 7.20.

```
html {
    font-family: Arial, Helvetica, sans-serif;
    display: inline-block;
    text-align: center;
  }
```

Slika 7.20 style za html

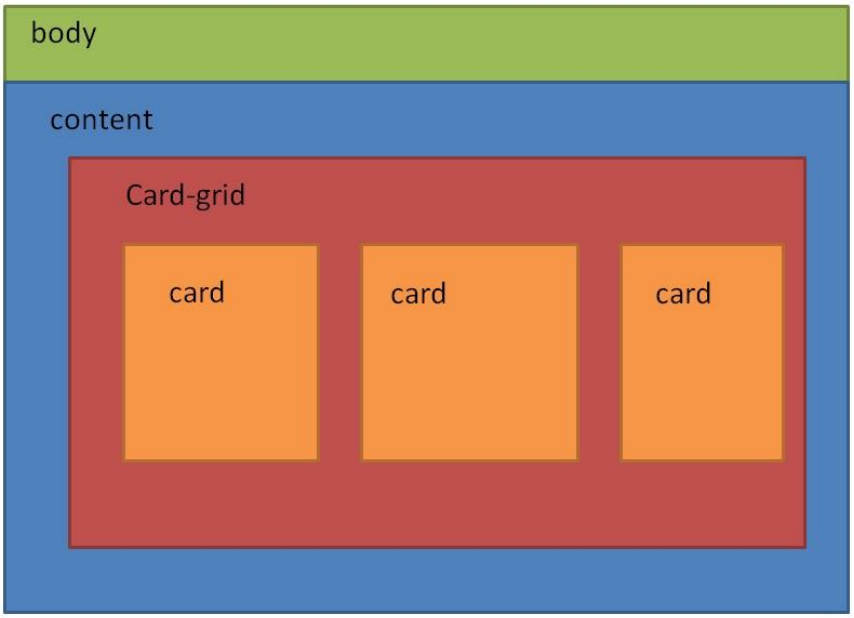
Da bi se dobio izgled gumba kao što je prikazano prije potrebno je urediti nekoliko svojstava. Sobzirom da su gumbovi označeni svaki sa svojim *id*, tada u *style* svaki gumb se poziva tako da se ispred imena stavi znak ljestvi (#). Ako je element označen sa *class* onda se poziva tako da se ispred imena stavi točka. Sa *background-color* se uređuje boja pozadine gumba, tj. boja gumba u ovom slučaju. Ako se žele izraženi rubovi onda se može pod *border* upisati neka debljina ruba. *Padding* određuje udaljenost elementa od nekog zadanog ruba, npr. *padding-left* će pomaknuti element u desno za neku vrijednost. *Font-size* je veličina teksta, dok sa *margin* se određuje površina između nekog ruba i elementa. *Grid-column* i *grid-row* su elementi *grid*-a koji određuju poziciju elementa u mreži. Svi ti elementi su potrebni da bi jedan gumb dobio neki potpuni izgled, kod za jedan gum se može vidjeti na slici 7.21. Svi ostali gumbovi u projektu su na isti način napravljeni.

```
#button1 {
  background-color: #2f4468;
  border: none;
  color: white;
  padding: 10px 20px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 18px;
  margin: 6px 3px;
  cursor: pointer;
  -webkit-touch-callout: none;
  -webkit-user-select: none;
  -khtml-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
  -webkit-tap-highlight-color: rgba(0,0,0,0);
  align-self:centar;
  grid-column: 2 / 3;
  grid-row: 2 / 3;
}
```

Slika 7.21 Style za gumb

Sa *img* je označen element koji sadrži u sebi video koji kamera snima. Sobzirom da se on već nalazi unutar *card* elementa potrebno je samo zadati visinu i širinu elementa. Sa *width* se zadaje

širina, postavlja se *auto* da bi se automatski smanjila u slučaju da se gleda sa nekog manjeg uređaja, ali postavlja se i najveća veličina da se ne može povećati van nekog osovnog sučelja i poremetiti raspored ostalih elemenata na stranici. *Height* ili visina se isto postavlja na *auto*, pa će se tako uvijek prilagoditi širini. Element „h1*"* označava naslov na vrhu internet stranice te dodavanjem svojstva u „h1*"* uređuje se tekst tog naslova. „Topnav" je površina na vrhu internet stranice u kojoj se nalazi naslov. Mali element za sve ostale manje naslove na internet stranici je označen sa „p" te u ovom slučaju samo mijenja veličinu teksta. *Body* je elementa koji označava površinu, a „content" je njegov podelement. *Body* predstavlja cijelu površinu internet stranice, odnosno ekrana, dok „content je površina *body*-a što znači da ako je body 50% ekrana, 100% contenta će isto biti 50% ekrana, a npr. 50% „content će biti 25% ekrana. Pa tako je ovdje samo zadan *padding* u „content" da se dobije odmak za neku površinu od ruba *body*-a. Unutar „contenta" se nalazi „card-grid" koji je mreža. Preko mreže je lakše rasporediti elemente po internet stranici. Unutar „card-grida" se nalaze elementi „card" koji su zapravo prozori koji su se već spominjali u prethodnim poglavljima. Radi lakšeg razumijevanja to se može vidjeti na slici 7.22. Za „card-grid" se ne želi da prekriva cijeli „content" jer možda treba staviti još elemenata uz ili ispod „card-grid". Pa se zato zadaje maksimalna širina i visina te *margin* unutar elementa „card-grid". Također se zadaje grid-gap, odnosno prostor između čelija mreže. Sa *grid-template-column* zadaje se vrsta mreže, pa tako ovdje je postavljena opcija *repeat(auto-fit, minmax(300px, 1fr))* koja omogućuje postavljanje jednakih ćelija te održava raspored ćelija. Kod se može vidjeti na slici 7.23.



Slika 7.22 Dizajn internet stranice

```
img {  width: auto ;
  max-width: 100% ;
  height: auto ;
}
h1 {
  font-size: 1.8rem;
  color: white;
}
p {
  font-size: 1.4rem;
}
.topnav {
  overflow: hidden;
  background-color: #0A1128;
}
body {
  margin: 0;
}
.content {
  padding: 30px;
}
.card-grid {
  max-width: 1500px;
  min-width: 250px;
  margin: 20px auto;
  display: grid;
  grid-gap: 2rem;
  position: relative;
  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

}
```

Slika 7.23 Body, content i card-grid

„Card“ element se uređuje sa *background-color:white*, *box shadow* se stavlja da prikaze imitaciju sjena oko elementa pa se dobiva izgled 3D prostora. Također je potrebno zadati visinu i maksimalnu visinu te poziciju da je *relative*. Naslov elementa card je pod nazivom „card-title1“ te mu se pridružuje nova veličina teksta, nova boja slova, tekst se podebljava sa *bold* te se zadaje pozicija u centru. To se sve može vidjeti na slci 7.24.

```
.card {
  background-color: white;
  box-shadow: 2px 2px 12px 1px rgba(140,140,140,.5);
  min-height: 250px;
  height: auto-fit, minmax(200px, 1fr);
  position:relative;
}
.card-title1 {
  font-size: 1.2rem;
  font-weight: bold;
  color: #034078;
  box-sizing: content-box;
  align-self:centar;
}
```

Slika 7.24 Card i card-title

Treći prozor ima svoju mrežu pod nazivom „raspored“ da bi se lakše gumb i kućice za upisivanje vrijednosti senzora mogle smjestiti. Pozicija je zadana kao *apsolute*, a visina i širina su 100% tako da bi površina mreže bila preko cijelog „card-a“. *Border* je zadan kao 5px tako da bi se mreža odmaknula od ruba ali je bezbojan pa se ne vidi, dok *padding* je još 10px od ruba. Time se dobiva da niti tekst niti bilo koji element ne može biti na rubu „card-a“. Pod display je zadan *grid*, a *grid-template-column:1fr 1fr 1fr*, što predstavlja tri jednaka retka, dok *grid-template-rows:1fr 1fr 1fr 1fr 1fr* je pet jednakih stupaca. Kućice za upisavanje vrijednosti senzora su defirane sa klasom „box1“, „box2“ i „box3“ te su jednake u svemu osim poziciji. Sa *box-sizing: content box* se defira da će se nešto upisati unutra te se time definira „ponašanje“ širine i visine. Širina i visina su zadane kao *auto*, da bi se promjenile u slučaju da je broj unutra širi od predviđenog. Ali su definirane i maksimalna i minimalna širina i visina da se prilikom smanjenja ili povećanja prozora nebi izgubio izgled. Zadan je mali rub sa *border* dok je unutrašnjost bijela da bi se vidio tekst. Kod za taj dio se može vidjeti na slici 7.25.

```
.raspored{
  position: absolute;
  height: 100%;
  width: 100%;
  box-sizing: border-box;
  border: 5px;
  padding: 10px;
  display: grid;
  grid-template-columns: 1fr 1fr 1fr ;
  grid-template-rows:1fr 1fr 1fr 1fr 1fr;

}
.box1{
  box-sizing: content-box;
  font-size: 20px;
  padding: 5px;
  width: auto;
  height: auto;
  min-width: 80px;
  min-height: 25px;
  max-width: 120px;
  max-height: 25px;
  border: solid 2px black;
  background-color: #ffffff;
}
```

Slika 7.25 style rasporeda i box1

„State“ označava tekst uz kućice u trećem prozoru. Tim elementom mijenja se boja teksta, veličina te se određuje pozicija u mreži. Element „grid“ stvara mrežu sa pet redaka i pet stupaca u drugom prozoru. Dok element „grid2“ stvara mrežu sa tri jednaka stupca i retka u četvrtom prozoru. Njihov kod se može vidjeti na slici 7.26.

```
.state{
  color: #000000;
  font-size: 20px;
  position: absolute;
  grid-column: 1 / 2;
  grid-row: 3 / 4;
}
.grid{
  position: absolute;
  height: 100%;
  width: 100%;
  box-sizing: border-box;
  display: grid;
  grid-template-columns: 1fr 2fr 3fr 2fr 1fr;
  grid-template-rows:40px 50px 50px 50px 50px ;
}
.grid2{
  position: absolute;
  padding: 10px;
  height: 100%;
  width: 100%;
  box-sizing: border-box;
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows:1fr 1fr 1fr ;
}
```

Slika 7.26 Grid, grid2 te state elementi

Sa „box4“ je označen tekst u drugom prozoru koji označa smjer kretanja. Definiran je sa veličinom teksta, bojom te širinom i visinom koje su namještene na *auto*. *Padding-top* je stavljen da ne dođe do dodira sa gornjim gumbom te je pozicija zadana preko *grida*. Elementi „delay“ i „visina“ su elementi gumbova sa strelicma u četvrtom prozoru. Oba imaju zadanu visinu i širinu koje se ne mogu mijenjati no *margin* je zadan sa *auto*. *Padding* je stavljen zbog bolje preglednosti sa lijeve i desne strane te oba imaju zadanu pozicju preko *grida*. Njihov kod se može vidjeti na slici 7.27.

```
.box4{
  box-sizing: content-box;
  width: auto;
  padding-top: 14px;
  height: auto;
  font-size: 1.2rem;
  background-color: #ffffff;
  grid-column: 3 / 4;
  grid-row: 3 / 4;
  text-align: inherit;
}
#delay{
  height:20px;
  width:50px;
  margin: auto;
  padding: 10px 10px;
  text-align: middle;
  font-size: 16px;
  grid-column: 1 / 2;
  grid-row: 3 / 4;
}
#visina{
  height:20px;
  width:50px;
  margin: auto;
  padding: 10px 10px;
  text-align: middle;
  font-size: 16px;
  grid-column: 3 / 4;
  grid-row: 3 / 4;
}
```

Slika 7.27 Elementi box4, delay i visina

Element „Veza“ označa tekst „Not Connected“ odnosno „Connected“ u drugom przoru. Pridodjeljena mu je nova veličina teksta te je tekst označen sa *bold*. Također boja koja je drugačija od drugih te je poravnat u sredinu sa *self-align:centar*. Kao i sve ostalo u tom prozoru i njemu je pozicija određena preko *grida*. „Grid3“ je elemnt koji stvara mrežu u petom prozoru sa tri jednaka stupca te pet redaka u kojem je svaki 50px. Kod se može vidjeti na slici 7.28.

```
.Veza{
  font-size: 1.2rem;
  font-weight: bold;
  color: #7e000098;
  box-sizing: content-box;
  align-self:centar;
  grid-column: 2 / 3;
  grid-row: 1 / 2;
}
.grid3{
  position: absolute;
  padding: 10px;
  height: 100%;
  width: 100%;
  box-sizing: border-box;
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows:50px 50px 50px 50px 50px ;
}
```

Slika 7.28 Elementi Veza i grid3

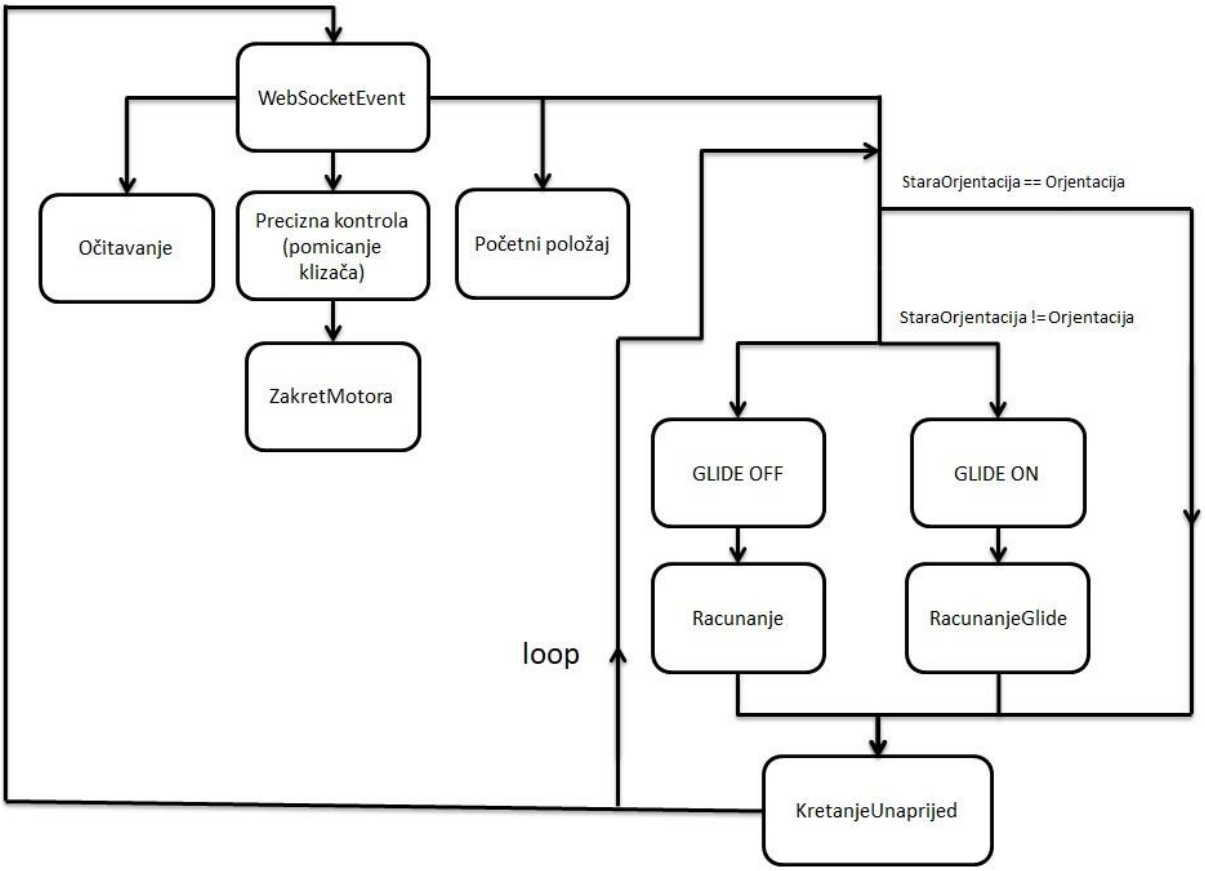
Element „custom-select“ je element koji označava gumb sa padajućim izbornikom. Njemu se definira širina na *auto* te se poravnava u centar sa *self-align:centar* dok pozicija je određena preko *grida*. „Slider“ je element klizača, definira mu se visina na 15px te širina 100% da bi bio širok koliko i element „card“. *Margin* i *border* radius se definiraju radi izgleda klizača. Sljedaći element *slider::-moz-range-thumb* uređuje gumb koji se miče po klizaču. Zadaje se visina i širina od 30px da bi bio okrugao, te se zadaje *border* i *color* da se dobije željeni izgled. Element „SliderCon1“ označava mjesto gdje će biti klizač što znači da je element „slider“ podelement „SliderCon1“. Pa tako postavljanjem pozcije „SliderCon1“ se postavlja „slider“. Njihov kod se može vidjeti na slici 7.29.

```
.custom-select{
  width: auto;
  align-self:center;
  grid-column: 1 / 2;
  grid-row: 2 / 3;
}
.slider {
  -webkit-appearance: none;
  margin: 0 auto;
  width: 100%;
  height: 15px;
  border-radius: 10px;
  background: #FFD65C;
  outline: none;
}
.slider::-moz-range-thumb {
  width: 30px;
  height: 30px;
  border-radius: 50% ;
  background: #034078;
  cursor: pointer;
}
.sliderCon1{
  grid-column: 1 / 4;
  grid-row: 3 / 4;
  padding-top: 10px;
}
```

Slika 7.29 Elementi za uređivanje klizača

7.3. Kod za NodeMCU 8266

NodeMCU sadrži glavni dio program. Nakon što se pritisne gumb na internet stranici šalju se ključne riječi *websocketom* do NodeMCU gdje se nakon toga rade proračuni te pokretanje motora za pomicanje. Dijagram za tok program se može vidjeti na slici 7.30.



Slika 7.30 Dijagram toka programa NodeMCU

Na početku programa unose se potrebne biblioteke te se zadaju sve potrebne varijable. Na slici 7.30. se mogu vidjeti varijable potrebne za slanje odnosno primanje ključnih riječi *websocketom*. Varijabla „DELAY“ označava pauzu između pokreta nogu, njezinom promjenom može se ubrzati ili usporiti hod. „Orjentacija“ sprema ključnu riječ za smjer kretanja. „StaraOrjentacija“ je pomoćna varijabla da bi se smanjilo računanje i ubrzao rad programa. „Glide“ govori da li je „glide“ opcija uključena. Drugi dio slike 7.31. su imena pinova, svaki pin za PWM signal ima ime onog zgloba koji kontrolira.

```
int DELAY = 7;
String Orjentacija = "STOP";
String StaraOrjentacija = "STOP";
String Glide = "OFF";
float Vrijednost_senzora;


//Prva Noga - prednja lijeva strana
int Kuk_1 = 13;
int Zglob_1 = 14;
int Koljeno_1 = 15;
//Druga Noga - prednja desna strana - PWM 2, na drugoj pločici
int Kuk_2 = 4;
int Zglob_2 = 5;
int Koljeno_2 = 6;
//Treca Noga - sredina lijeva strana
int Kuk_3 = 12;
int Zglob_3 = 13;
int Koljeno_3 = 14;
//Cetvrta Noga - srednja desna strana - PWM 2, na drugoj pločici
int Kuk_4 = 0;
int Zglob_4 = 1;
int Koljeno_4 = 2;
//Peta Noga - Zadnja lijeva strana
int Kuk_5 = 8;
int Zglob_5 = 9;
int Koljeno_5 = 10;
//Sesta Noga - Zadnja desna strana - PWM 2, na drugoj pločici
int Kuk_6 = 0;
int Zglob_6 = 1;
int Koljeno_6 = 2;

int Servo_senzora = 8; // PWM 1
```

Slika 7.31 Varijable ključnih riječi te imena pinova

Funkcija „ZakretMotora“ ima više verzija. Svaka kontolira određene pinove. Sobzirom da su dvije PCA9685 pločice postoje 2 *pwm* signala, „pwm1“ i „pwm2“ ovisno na koju pločicu je pin spojen. Funkcija prima vrijednost na koju se motor mora zakrenuti u stupnjevima te pin na kojem se taj motor nalazi. Nakon što primi vrijedost, sa funkcijom *map* se za tu vrijednost pronalazi odgovarajuća vrijednost širine pulsa te nakon toga sa *setPWM* se pokreće motor. Kod tih funkcija se može vidjeti na slici 7.32.

```
void ZakretMotoraPWM_1K(float ang, int servo) {
  int pulse = map(ang, -180, 0, SERVOMIN, SERVOMAX);
  pwm1.setPWM(servo, 0, pulse);
}

void ZakretMotoraPWM_35K(float ang, int servo) {
  int pulse = map(ang, -180, 0, SERVOMIN, SERVOMAX);
  pwm2.setPWM(servo, 0, pulse);
}


void ZakretMotoraPWM_6K(float ang, int servo) {
  int pulse = map(ang, 0, 180, SERVOMIN, SERVOMAX);
  pwm2.setPWM(servo, 0, pulse);
}

void ZakretMotoraPWM_24K(float ang, int servo) {
  int pulse = map(ang, 0, 180, SERVOMIN, SERVOMAX);
  pwm1.setPWM(servo, 0, pulse);
}

void ZakretMotoraPWM_124(float ang, int servo) {
  int pulse = map(ang, -90, 90, SERVOMIN, SERVOMAX);
  pwm1.setPWM(servo, 0, pulse);
}

void ZakretMotoraPWM_356(float ang, int servo) {
  int pulse = map(ang, -90, 90, SERVOMIN, SERVOMAX);
  pwm2.setPWM(servo, 0, pulse);
}
```

Slika 7.32 Funkcija ZakretMotora

Funkcija „Nulti_korak“ se poziva prilikom pritiska na gumb „Pocetni“ te stavlja robota u neki unaprijed zadani početni položaj. Najprije zakreće po 2 zgloba, to su srednji članci nogu, od podloge prema gore da ne dođe do sudara sa podlogom prilikom pomicanja ostalih dijelova. Nakon toga zakreće Kukove, to su prvi članci do tijela, na neki kut gdje se noge ne budu sudarile prilikom pomicanja ostalih dijelova. I na kraju zakreće Koljeno s kojima se podiže u zrak i spreman je za kretanje. Negativne vrijednosti kutova su jer se neki motori zakreću u suprotnom smjeru od pozitivnog smjera. Kod toga se može vidjeti na slici 7.33.

```
void Nulti_korak() {


  ZakretMotoraPWM_124(prednja1Zglob, Zglob_1);
  ZakretMotoraPWM_124(-prednja2Zglob, Zglob_2);
  delay(300);
  ZakretMotoraPWM_356(srednji3Zglob, Zglob_3);
  ZakretMotoraPWM_124(-srednji4Zglob, Zglob_4);
  delay(300);
  ZakretMotoraPWM_356(zadnja5Zglob, Zglob_5);
  ZakretMotoraPWM_356(-zadnja6Zglob, Zglob_6);
  delay(1000);
  ZakretMotoraPWM_124(-prednja1Kuk, Kuk_1);
  ZakretMotoraPWM_124(prednja2Kuk, Kuk_2);
  ZakretMotoraPWM_356(srednji3Kuk, Kuk_3);
  ZakretMotoraPWM_124(srednji4Kuk, Kuk_4);
  ZakretMotoraPWM_356(zadnja5Kuk, Kuk_5);
  ZakretMotoraPWM_356(-zadnja6Kuk, Kuk_6);
  delay(1000);
  ZakretMotoraPWM_1K(-prednja1Koljeno, Koljeno_1);
  ZakretMotoraPWM_24K(prednja2Koljeno, Koljeno_2);
  delay(100);
  ZakretMotoraPWM_35K(-srednji3Koljeno, Koljeno_3);
  ZakretMotoraPWM_24K(srednji4Koljeno, Koljeno_4);
  delay(100);
  ZakretMotoraPWM_35K(-zadnja5Koljeno, Koljeno_5);
  ZakretMotoraPWM_6K(zadnja6Koljeno, Koljeno_6);
  delay(1000);

}
```

Slika 7.33 Funkcija Nulti korak

„Udaljenost“ je funkcija koja upravlja senzorom i daje vrijednost koju je senzor očitao. Na ulazu se mora unijeti na koji je pin spojen *TrigPin* senzora a na koji je spojen *EchoPin* senzora. Zatim se definiraju potrebne varijable, definiraju se pinovi te vrijednost brzine zvuka. Nakon toga se *TrigPin* dovodi napon sa *HIGH* te on šalje zvučni signal. Nakon 10 milisekundi prekida se dovod napona sa *LOW*. Sa *pulseIn* se zatim sa *EchoPin*-a očita vrijeme potrebno za putovanje zvučnog vala te se to ubaci u formulu i dobije se vrijednost udaljenosti prepreke od senzora u centimetrima. Ta vrijednost se spremi u varijablu „distance“ što se može vidjeti na slici 7.34.

```
float Udaljenost(const int trigPin, const int echoPin ) {
  long duration ;
  float distance;
  //define sound velocity in cm/uS
#define SOUND_VELOCITY 0.0343
  // put your main code here, to run repeatedly:
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT);
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distance = (duration * SOUND_VELOCITY) / 2;


  return distance;
}
```

Slika 7.34 Funkcija Udaljenost

Funkcijom „Očitavanje“ se okreće senzor na 3 različite pozicije. Nakon okretanja na poziciju se pokreće funkcija „Udaljenost“ te se sprema vrijednost u varijablu i šalje *websocketom* internet stranici. Kod se može vidjeti na slici 7.35.

```
void Ocitavanje(const int trigPin, const int echoPin) {


  ZakretMotoraPWM_24K(0, Servo_senzora);
  delay(300);
  Vrijednost_senzora = Udaljenost(trigPin, echoPin);
  Serial.println(Vrijednost_senzora);
  delay(200);
  Senzor["Desno"] = Vrijednost_senzora;

  ZakretMotoraPWM_24K(180, Servo_senzora);
  delay(300);
  Vrijednost_senzora = Udaljenost(trigPin, echoPin);
  Serial.println(Vrijednost_senzora);
  delay(200);
  Senzor["Lijevo"] = Vrijednost_senzora;

  ZakretMotoraPWM_24K(90, Servo_senzora);
  delay(300);
  Vrijednost_senzora = Udaljenost(trigPin, echoPin);
  Serial.println(Vrijednost_senzora);
  delay(200);
  Senzor["Ispred"] = Vrijednost_senzora;

  char dataa[150];
  size_t len = serializeJson(Senzor, dataa);
  webSocket.broadcastTXT(dataa, len);

}
```

Slika 7.35 Funkcija Ocitavanje

Funkcije „PrvaNoga“, „DrugaNoga“, „TrećaNoga“, „ČetvrtaNoga“, „PetaNoga“ i „ŠestaNoga“ pokreću nogu prema vrijednostima koje su primljene. Funkcije primaju pozitivne vrijednosti ali neke postaju negativne prilikom pozivanja funkcije „ZakretMotora“. Negativne vrijednosti zapravo označavaju negativan smjer motora. Funkcije su ovako napravljene da se pojednostavi pisanje koda i smanji broj grešaka zbog krivo upisanog smjera. Kod funkcija se može vidjeti na slici 7.36.

```
void Prva_Noga(float q1_naprijed, float q2_naprijed, float q3_naprijed) {

  ZakretMotoraPWM_124(-q1_naprijed, Kuk_1);
  ZakretMotoraPWM_124(q2_naprijed, Zglob_1);
  ZakretMotoraPWM_1K(-q3_naprijed, Koljeno_1);
}

void Druga_Noga(float q1_naprijed, float q2_naprijed, float q3_naprijed) {

  ZakretMotoraPWM_124(q1_naprijed, Kuk_2);
  ZakretMotoraPWM_124(-q2_naprijed, Zglob_2);
  ZakretMotoraPWM_24K(q3_naprijed, Koljeno_2);
}

void Treca_Noga(float q1_naprijed, float q2_naprijed, float q3_naprijed) {

  ZakretMotoraPWM_356(-q1_naprijed, Kuk_3);
  ZakretMotoraPWM_356(q2_naprijed, Zglob_3);
  ZakretMotoraPWM_35K(-q3_naprijed+20, Koljeno_3);
}


void Cetvrta_Noga(float q1_naprijed, float q2_naprijed, float q3_naprijed) {

  ZakretMotoraPWM_124(q1_naprijed, Kuk_4);
  ZakretMotoraPWM_124(-q2_naprijed, Zglob_4);
  ZakretMotoraPWM_24K(q3_naprijed, Koljeno_4);
}

void Peta_Noga(float q1_naprijed, float q2_naprijed, float q3_naprijed) {


  ZakretMotoraPWM_356(q1_naprijed, Kuk_5);
  ZakretMotoraPWM_356(q2_naprijed, Zglob_5);
  ZakretMotoraPWM_35K(-q3_naprijed-15, Koljeno_5);
}

void Sesta_Noga(float q1_naprijed, float q2_naprijed, float q3_naprijed) {


  ZakretMotoraPWM_356(-q1_naprijed, Kuk_6);
  ZakretMotoraPWM_356(-q2_naprijed, Zglob_6);
  ZakretMotoraPWM_6K(q3_naprijed, Koljeno_6);
}
```

Slika 7.36 Funkcije za pokretanje Nogu

Funkcija „Izracun“ je proračun inverzne kinematike. Funkcija prima vrijednosti kordinata x, y i z te ih koristi da bi se izračunali potrebni kutovi koji su spremljeni u varijable „q1_motora“,

„q2_motora“ te „q3_motora“. *SpeedTrig* je biblioteka koja ubrzava računanje trigonometrijskih funkcija. Funkcija „Izracun“ se može vidjeti na slici 7.37.

```
void Izracun(double x, double y, double z) {


  brojnik_q1 = y;
  nazivnik_q1 = x;

  double c3 = (sq(x) + sq(y) + sq(z) - sq(L2) - sq(L3)) / (2 * L2 * L3);
  if ((c3 <= -1) || (c3 >= 1)) {
    Serial.println("Nevaljaju brojevi");
  }
  else {
    nazivnik_q3 = c3;
    double s3 = -sqrt(1 - (c3 * c3));
    brojnik_q3 = s3;

    brojnik_q2 = (L2 + L3 * c3) * z - L3 * s3 * sqrt((x * x) + (y * y));
    nazivnik_q2 = (L2 + L3 * c3) * sqrt((x * x) + (y * y)) + L3 * s3 * z;
  }
  if (x < 0) {
    q1_motora = -degrees(PI - SpeedTrig.atan2(brojnik_q1, nazivnik_q1));
  }
  else {
    q1_motora = degrees(SpeedTrig.atan2(brojnik_q1, nazivnik_q1));
  }
  q2_motora = degrees(SpeedTrig.atan2(brojnik_q2, nazivnik_q2));
  q3_motora = PI - degrees(SpeedTrig.atan2(brojnik_q3, nazivnik_q3));
}
```

Slika 7.37 Funkcija Izracun – racunanje inverzne kinematike

Funkcija „elipsa“ prima kordinate dvije točke te iz njih računa putanju elipse. Sprema kordinatu x u varijablu „PozX“, kordintu y u varijablu „PozY“ te kordinatu z u varijablu „PozZ“. Kod za elipsu se može vidjeti na slici 7.38.

```
void elipsa(double x1, double x2, double y1, double y2, double z1, double z2, double v, int t) {

  double a = (y2 - y1) / 2;
  double b = (z2 - z1) / 2;
  double h = (y2 + y1) / 2;
  double k = (z2 + z1) / 2;
  double c = (x2 - x1) / 2;
  double n = (x2 + x1) / 2;
  int Maxt = 50;

  PozY = a * v * cos(t * (PI / Maxt)) + h;
  PozZ = b * v * sin(t * (PI / Maxt)) + k;


  if (x2 - x1 == 0) {
    PozX = x1;
  }
  else {
    PozX = v * c * cos(t * (PI / Maxt)) + n;
  }
}
```

Slika 7.38 Funkcija za računanje elipse

„Pravac“ je funkcija ista kao elipsa osim što kordinata Z osi je jednaka cijelom putanjom. Također kao „elipsa“ prima dvije točke sa x, y i z kordinatama te daje na izlazu x, y i z kordinate za pravac. Kod se može vidjeti na slici 7.39.

```
void Pravac(double x1, double x2, double y1, double y2, double z1, double z2, float v, int t) {

  double a = (y2 - y1) / 2;
  double b = (z2 + z1) / 2;
  double h = (y2 + y1) / 2;
  double c = (x2 - x1) / 2;
  double n = (x2 + x1) / 2;

  int Maxt = 50;

  PozY = a * v * cos(t * (PI / Maxt)) + h;
  PozZ = b;


  if (x2 - x1 == 0) {
    PozX = x1;
  }
  else {
    PozX = v * c * cos(t * (PI / Maxt)) + n;
  }
}
```

Slika 7.39 Fukcija za računanje pravca

Da bi se dobilo okretanje oko osi robot se mora kretati po kružnici pa se za to koristi funkcija „Kruznica“. Na ulazu prima radijus, z kordinatu najviše i najniže točke elipse, te prima ograničenja za kružnicu. Za nogu nije potrebna cijela kružnica već samo dio. Također prima

vrijednosti pomaka centra po x i y, koji su se u ovom projektu odredili „pješice“. I na izlazu daje vrijednosti x, y i z kordinate. Kod se može vidjeti na slici 7.40.

```
void Kruznica(float r, float z1, float z2, float GornjaGranica, float DoljnjaGranica, float PomakPoX, float PomakPoY, float t) {
  double b = (z2 + z1) / 2;
  int Maxt = 50;

  PozY = r * sin((t / Maxt) * (PI / 2) * radians(GornjaGranica) + radians(DoljnjaGranica)) + PomakPoY;
  PozX = r * cos((t / Maxt) * (PI / 2) * radians(GornjaGranica) + radians(DoljnjaGranica)) + PomakPoX;
  PozZ = b;
}
```

Slika 7.40 Funkcija za računanje kružnice

Funkcija „handleWebSocketMessage“ odlučuje što se dogodi kada dođe poruka kroz *websocket*. Ovdje ima tri opcije, ako dođe riječ „Values“ tada znači da korisnik želi pokrenuti senzor te se pokreće funkcija „Očitavanje“. Ako dođe „XX“ tada je korisnik pomaknuo neki *slider* u zadnjem prozoru i pokreću se funkcije „IzborNoge“, „IzborDijelaNoge“ te „IzborPina“. Ukoliko korisnik pritisne gumb za kretanje tada će se raspakirati paket te će se vrijednosti koje su došle spremiti pod odgovarajuće varijable. Cijeli kod te funkcije se može vidjeti na slici 7.41.

```
void handleWebSocketMessage(uint8_t arg, uint8_t *data, size_t len) {
  message = (char*)data;
  if (message.indexOf("Values") >= 0) {
    Ocitavanje(0, 2);
  }
  else if (message.indexOf("XX") >= 0) {
    String Value = message.substring(2);
    deserializeJson(Slider, Value);
    Orjentacija = Slider["Orjentacija"].as<String>();
    BrojSlidera = Slider["BrojSlidera"];
    VrijednostSlidera = Slider["VrijednostSlidera"];
    IzborNoge();
    IzborDijelaNoge();
    IzborPina();
  }
  else {
    String Value = message.substring(2);
    deserializeJson(Kord, Value);
    DELAY = Kord["Delay"];
    Z1 = Kord["Z1"];
    Orjentacija = Kord["Orjentacija"].as<String>();
    Glide = Kord["Glide"].as<String>();
  }
}
```

Slika 7.41 Funkcija handleWebSocketMessage

Funkcija „webSocketEvent“ će se pokrenuti kada se dogodi nešto vezano uz *websocket*. Ukoliko dođe do spajanja ispisati će poruku da je *websocket* spojen, isto tako ako se odspoji. Također ako dođe poruka kroz *websocket* pokrenuti će funkciju „handleWebSocketMessage“. Kod te funkcije se može vidjeti na slici 7.42.

```
void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t len) {

  switch (type) {
    case WStype_DISCONNECTED:
      Serial.printf("[%u]NodeMCU Disconnected!\n");
      break;
    case WStype_CONNECTED: {
        webSocket.sendTXT(num, "NodeMCU Connected");
        Serial.printf("NodeMCU Connected");
        socket_port = num;
      }
      break;
    case WStype_TEXT: {
        handleWebSocketMessage(num, payload, len);
      }

      break;
  }
}
```

Slika 7.42 Funkcija webSocketEvent

Funkcija „IzborDijelaNoge“ određuje koji članak noge se treba pokrenuti prema tome koji klizač je pomaknut. „IzborNoge“ govori koja noga je izabrana u padajućem izborniku. Dok „IzborPina“ spaja ono što je odlučeno u „IzborDijelaNoge“ i „IzborNoge“ te daje ime pina za pokretanje motora koji se traži. Kodovi tih funkcija se mogu vidjeti na slikama 7.43 i 7.44. Funkcije nisu prikazane u potpunosti na slikama radi njihova obujma.

```
void IzborDijelaNoge() {
  if (BrojSlidera == 1) {
    DioNoge = "Kuk_";
  }
  if (BrojSlidera == 2) {
    DioNoge = "Zglob_";
  }
  if (BrojSlidera == 3) {
    DioNoge = "Koljeno_";
  }
}

void IzborNoge() {
  if (Orjentacija == "PrvaNoga") {
    BrojNoge = "1";
  }
  else if (Orjentacija == "DrugaNoga") {
    BrojNoge = "2";
  }
  else if (Orjentacija == "TrecaNoga") {
    BrojNoge = "3";
  }
```

Slika 7.43 Funkcije IzborDijelaNoge i IzborNoge

```
void IzborPina() {
  if (DioNoge + BrojNoge == "Kuk_1") {
    Pinn = Kuk_1;
  }
  else if (DioNoge + BrojNoge == "Kuk_2") {
    Pinn = Kuk_2;
  }
  else if (DioNoge + BrojNoge == "Kuk_3") {
    Pinn = Kuk_3;
  }
  else if (DioNoge + BrojNoge == "Kuk_4") {
    Pinn = Kuk_4;
  }
  else if (DioNoge + BrojNoge == "Kuk_5") {
    Pinn = Kuk_5;
  }
  else if (DioNoge + BrojNoge == "Kuk_6") {
    Pinn = Kuk_6;
  }
  else if (DioNoge + BrojNoge == "Zglob_1") {
    Pinn = Zglob_1;
```

Slika 7.44 Funkcija IzborPina

Funkcija „Racunanje“ određuje potrebne proračune na temelju željenog pravca kretanja. Na ulazu prima početnu i završnu točku kretanja. Točke su unaprijed zadane za svaki pravac kretanja kao što se može vidjeti na slici 7.45.

```
void Racunanje(double x1, double x2, double x1s, double x2s, double y1, double y2, double y1s, double y2s, double z1, double z2) {


if (Orjentacija == "Strogo lijevo" || Orjentacija == "Strogo desno") {
  x1 = 69;
  x2 = 42;
  x1s = 80;
  x2s = 80;

  y1 = 40;
  y2 = 80;
  y1s = 40;
  y2s = -40;

}
else if (Orjentacija == "Nazad") {
```

Slika 7.45 Izabrane kordinate prema željenom pravcu – funkcija Racunanje

Nakon izbora točki funkcija ulazi u *for* petlju te opet na temlju željenog pravca provodi potrebne proračune. Nakon provedenih proračuna za elipsu, pravac ili kružnicu sprema te vrijednosti u nove varijable što se može vidjeti na slici 7.46. Nakon što funkcija završi svaka varijabla u sebi sadrži sve točke potrebne za cijelu putanju. Svaka iteracija *for* petlje prvo računa putanju, a zatim računa inverznu kinematiku. Vrijednosti izračunate inverznom kinematikom se također spremaju u varijablu što se vidi na slici 7.47.

```
for (int i = 0; i <= 50; i++) {

  int znakA;
  int znakB;
  if (Orjentacija == "Naprijed" || Orjentacija == "Nazad" ) {
    znakA = i;
    znakB = 50 - i;
  }
  else if (Orjentacija == "Strogo desno" || Orjentacija == "Strogo lijevo"|| Orjentacija == "Lagano desno" || Orjentacija == "Lagano lij
    znakA = 50 - i;
    znakB = i;
  }
  elipsa(x1, x2, y1, y2, z1, z2, 1, i);
    PozXEDp[znakA] = PozX;
    PozYEDp[znakA] = PozY;
    PozZEDp[znakA] = PozZ;
  if (Orjentacija == "Lagano lijevo" || Orjentacija == "Lagano desno"|| Orjentacija == "Nazad desno"|| Orjentacija == "Nazad lijevo") {
    elipsa(67, 40, 10, 46, z1, z2, 1, i);
    PozXEKp[znakB] = PozX;
    PozYEKp[znakB] = PozY;
    PozZEKp[znakB] = PozZ;
  }
  else {
    PozXEKp[znakB] = PozX;
    PozYEKp[znakB] = PozY;
    PozZEKp[znakB] = PozZ;
  }
  elipsa(x1s, x2s, y1s, y2s, z1, z2, 1, i);
    PozXEs[i] = PozX;
    PozYEs[i] = PozY;
    PozZEs[i] = PozZ;
```

Slika 7.46 Funkcija Racunanje izabire proračune za putanju

```
    Izracun(PozXEDp[znakA], PozYEDp[znakA], PozZEDp[znakA]);
      q1_motora_PrednjeNogeE[znakA] = q1_motora;
      q2_motora_PrednjeNogeE[znakA] = q2_motora;
      q3_motora_PrednjeNogeE[znakA] = q3_motora;
    Izracun(PozXEs[i], PozYEs[i], PozZEs[i]);
      q1_motora_SrednjeNogeE[i] = q1_motora;
      q2_motora_SrednjeNogeE[i] = q2_motora;
      q3_motora_SrednjeNogeE[i] = q3_motora;
    Izracun(PozXEKp[znakB], PozYEKp[znakB], PozZEKp[znakB]);
      q1_motora_StraznjeNogeE[znakB] = q1_motora;
      q2_motora_StraznjeNogeE[znakB] = q2_motora;
      q3_motora_StraznjeNogeE[znakB] = q3_motora;
  }
```

Slika 7.47 Funkcija Racunanje provodi proračune za inverznu kinematiku

Funkcija „RacunanjeGlide“ radi na isti način kao „Racunanje“ samo su kordinate točaka izabrane drugačije da bi se dobilo drugačije kretanje. Jedina razlika je što funkcija „RacunanjeGlide“ poziva funkciju „Racunanje“ u slučaju kada je „Glide“ ukljucen a željeni pravac je „Naprijed“ ili „Nazad“. Početak koda funkcije se može vidjeti na slici 7.48.

```
void RacunanjeGlide(double x1, double x2, double x1s, double x2s, double x1z, double

  if (Orjentacija == "Naprijed" || Orjentacija == "Nazad") {
    Racunanje(X1, X2, X1S, X2S, Y1, Y2, Y1S, Y2S, Z1, Z2);
  }
  if (Orjentacija == "Strogo desno" ) {
    x1 = 50;
    x2 = 90;
    x1s = 90;
    x2s = 50;
    x1z = 90;
    x2z = 50;
```

Slika 7.48 Funkcija RacunanjeGlide

Funkcija „Kretanje unaprijed“ je funkcija koja na temelju željenog smjera izabire kojim redom će se korstiti vrijednosti iz varijabli. Unutar funkcije se nalazi *for* petlja koja uzima vrijednosti određene iteracije. Važno je da *for* petlja unutar funkcije „Racunanje“ i „Kretanje unaprijed“ imaju isti broj iteracija. *For* petlja se dijeli na četiri dijela. Prvi i treći dio su pomaci nogu unaprijed dok drugi i četvrti su pomaci tijela unaprijed. Prvi i četvrti dio pomiču prvu, četvrtu i petu nogu, dok drugi i treći dio pomiču drugu, treću i šestu nogu. Cijela *for* petlja traje T. Prvi i drugi dio se odvijaju u isto vrijeme i traju T/2, a nakon toga slijedi treći i četvrti dio

istovremeno koji također traju T/2. Ta funkcija se sastoji od puno linija koda pa se je na slici 7.49. pokazan samo prvi dio.

```
void Kretanje_unaprijed() {


  for (int i = 0; i <= 100; i++) {
    if (Orjentacija == "STOP") {
      break;
    }
    if ((0 <= i) && (i <= 50)) {


      if (Orjentacija == "lijevo" && Glide == "OFF") {

      }
      else if (Orjentacija == "Nazad desno" && Glide == "OFF") {
        Prva_Noga(q1_motora_PrednjeNogeE[50-i], q2_motora_PrednjeNogeE[50-i], q3_motora_PrednjeNogeE[50-i]);
        Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);
        Peta_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i], q3_motora_PrednjeNogeE[i]);
      }
      else if (Orjentacija == "Nazad lijevo" && Glide == "OFF") {
        Prva_Noga(q1_motora_StraznjeNogeE[i], q2_motora_StraznjeNogeE[i], q3_motora_StraznjeNogeE[i]);
        Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);
        Peta_Noga(q1_motora_StraznjeNogeE[50 - i], q2_motora_StraznjeNogeE[50 - i], q3_motora_StraznjeNogeE[50 - i]);
      }
      else if (Orjentacija == "Lagano lijevo" && Glide == "OFF") {
        Prva_Noga(q1_motora_StraznjeNogeE[50-i], q2_motora_StraznjeNogeE[50-i], q3_motora_StraznjeNogeE[50-i]);
        Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);
        Peta_Noga(q1_motora_StraznjeNogeE[i], q2_motora_StraznjeNogeE[i], q3_motora_StraznjeNogeE[i]);
      }
      else if (Orjentacija == "Lagano desno" && Glide == "OFF") {
        Prva_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i], q3_motora_PrednjeNogeE[i]);
        Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);
        Peta_Noga(q1_motora_PrednjeNogeE[50-i], q2_motora_PrednjeNogeE[50-i], q3_motora_PrednjeNogeE[50-i]);
      }
      else if (Orjentacija == "Strogo lijevo" && Glide == "OFF") {
        Prva_Noga(q1_motora_PrednjeNogeE[50-i], q2_motora_PrednjeNogeE[50-i], q3_motora_PrednjeNogeE[50-i]);
        Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);
        Peta_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i], q3_motora_PrednjeNogeE[i]);
      }

    else if (Orjentacija == "Strogo desno" && Glide == "OFF") {
      Prva_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i], q3_motora_PrednjeNogeE[i]);
      Cetvrta_Noga(q1_motora_SrednjeNogeE[50-i], q2_motora_SrednjeNogeE[50-i], q3_motora_SrednjeNogeE[50-i]);
      Peta_Noga(q1_motora_PrednjeNogeE[50-i], q2_motora_PrednjeNogeE[50-i], q3_motora_PrednjeNogeE[50-i]);
    }
    else if((Orjentacija == "Nazad lijevo" || Orjentacija == "Lagano desno") && Glide == "ON"){
      Prva_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i], q3_motora_PrednjeNogeE[i]);
      Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);
      Peta_Noga(q1_motora_StraznjeNogeE[50-i], q2_motora_StraznjeNogeE[50-i], q3_motora_StraznjeNogeE[50-i]);
    }
    else if((Orjentacija == "Lagano lijevo" || Orjentacija == "Nazad desno") && Glide == "ON"){
      Prva_Noga(q1_motora_StraznjeNogeE[i], q2_motora_StraznjeNogeE[i], q3_motora_StraznjeNogeE[i]);
      Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);
      Peta_Noga(q1_motora_PrednjeNogeE[50-i], q2_motora_PrednjeNogeE[50-i], q3_motora_PrednjeNogeE[50-i]);
    }
    else if((Orjentacija == "Strogo lijevo" || Orjentacija == "Strogo desno") && Glide == "ON"){
      Prva_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i], q3_motora_PrednjeNogeE[i]);
      Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);
      Peta_Noga(q1_motora_StraznjeNogeE[i], q2_motora_StraznjeNogeE[i], q3_motora_StraznjeNogeE[i]);
    }
    else {
      Prva_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i], q3_motora_PrednjeNogeE[i]);
      Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);
      Peta_Noga(q1_motora_StraznjeNogeE[i], q2_motora_StraznjeNogeE[i], q3_motora_StraznjeNogeE[i]);
    }

    delay(DELAY);
  }
  if ((0 <= i) && (i <= 50)) {
    Serial.print("-------Povrat_Zglob_2----------------");
    Serial.print("i=");
    Serial.println(i);
    int t = 50 - i;
```

Slika 7.49 Prvi dio funkcije Kretanje unaprijed

Unutar *setupa* koji se izvršava prilikom pokretanja NodeMCU, stavlja se pokretanje *Seriala*, pokretanje *pwm* biblioteke, poziva se funkcija „Wifi“ kao i kod ESP32 CAM te se pokreće *websocket* server i sa *websocket.onEvent(webSocketEvent)* se postavlja da se funkcija „webSocketEvent“ pokrene prilikom nekog *websocket* događaja. *Setup* se može vidjeti na slici 7.50.

```
void setup() {

  Serial.begin(115200);
  pwm1.begin();          // Sends PWM signals.
  pwm1.setPWMFreq(50);   // Makes servos run at 60 Hz rate.
  delay(20);
  pwm2.begin();          // Sends PWM signals.
  pwm2.setPWMFreq(50);   // Makes servos run at 60 Hz rate.
  delay(20);


  Wifi();
  webSocket.begin();
  webSocket.onEvent(webSocketEvent);


}
```

Slika 7.50 Setup programa

Unutar *loop*-a se nalazi funkcija *websocket.loop* koja održava *websocket* vezu. Te glavni dio programa za kretanje. Na temelju izabrane opcije i pravca kretanja pomiče robota u željenom smjeru. Prvi dio provjera da li je koji klizač pomaknut pa pokreće samo traženi zglob. Drugi dio provjerava da li je gumb za početni položaj pritisnut, ako nije tada provjerava da li je „GLIDE“ uključen ili ne te nakon toga ulazi u funkciju „Kretanje unaprijed“. Provjera da li je „Orjentacija“ jednaka „StaraOrjentacija“ služi da se smanji računanje. U slučaju da su jednake onda nema potrebe ponovno računati putanju i inverznu kinematiku nego odmah krene sa kretanjem. Sve se može vidjeti na slici 7.51. i slici 7.52.

```
void loop() {

  webSocket.loop();

  if (Orjentacija == "STOP") {

  }
  else if (Orjentacija == "TrecaNoga" || Orjentacija == "PetaNoga") {
    if (BrojSlidera == 3) {
      ZakretMotoraPWM_35K(-VrijednostSlidera, Pinn);
      Orjentacija = "STOP";
    }
    else {
      ZakretMotoraPWM_356(VrijednostSlidera, Pinn);
      Orjentacija = "STOP";
    }
  }
  else if (Orjentacija == "SestaNoga") {
    if (BrojSlidera == 3) {
      ZakretMotoraPWM_6K(VrijednostSlidera, Pinn);
      Orjentacija = "STOP";
    }
    else {
      ZakretMotoraPWM_356(-VrijednostSlidera, Pinn);
      Orjentacija = "STOP";
    }
  }
  else if (Orjentacija == "DrugaNoga" || Orjentacija == "CetvrtaNoga") {
    if (BrojSlidera == 3) {
      ZakretMotoraPWM_24K(VrijednostSlidera, Pinn);
      Orjentacija = "STOP";
    }
    else {
```

Slika 7.51 Prvi dio loop-a

```
  else {

    if (Orjentacija == "PocetniPolozaj") {
      Nulti_korak();
      Orjentacija = "STOP";
    }
    else {
        if(Orjentacija != StaraOrjentacija){
          if (Glide == "OFF"){
              Racunanje(X1, X2, X1S, X2S, Y1, Y2, Y1S, Y2S, Z1, Z2);
              StaraOrjentacija = Orjentacija;
            }
          else if (Glide == "ON"){
              RacunanjeGlide(X1, X2, X1S, X2S, X1Z, X2Z, Y1, Y2, Y1S, Y2S, Y1Z, Y2Z, Z1, Z2);
              StaraOrjentacija = Orjentacija;
            }
          else{

            }
        }
        else if (Orjentacija == StaraOrjentacija){
        Kretanje_unaprijed();
      }
    }
  }
}
```

Slika 7.52 Drugi dio loop-a

8. ZAKLJUČAK

Nakon izrade ovog projekta može se vidjeti kompleksnost izrade paukovih nogu čak i na ovako jednostavnom robotu. Obično je potrebno nekoliko iteracija izrade konstrukcije prije nego se može koristiti. Proračun će smanjiti potreban broj iteracija ali neke detalje proračun ne može pokazati. Npr. udaljenost između nogu, način spajanja zglobova, raspored zglobova, visina tijela i druge. Nakon izrade pauka može se vidjeti i da bez programiranja i nekih dodatnih elektroničkih komponenti robot će održavati ravnotežu svojim dizajnom kao što je bilo rečeno u uvodu.

Za daljnje unaprijeđenje ovog robota preporučio bih da se pokušaju smanjiti dimenzije elektroničkih dijelova da bi se konstrukcija tijela optimizirala. Iako pauk hoda ovako, sa manjim i lakšim tijelom kretanje bi moglo biti čišće i preciznije. Također bi bilo dobro izraditi tijelo u obliku osmerokuta jer su tada noge na nešto optimalnijem mjestu. Sve to bi riješilo i problem trenja između podloge i noge, gdje trenutno dolazi do laganog proklizavanja, no lakšim tijelom bi se to smanjilo. Također bi bilo dobro promjeniti vrhove nogu koji bi dali veće trenje. Softver je napravljen vrlo jednostavno, bez zajedničkog koordinatnog sustava za sve noge, bez stabilizacije i slično. Za bolje kretanje bi bilo dobro implementirati zajednčki koordinatni sustav koji bi pratio neki pravac ili krivulju prilikom kretanja ili se može izraditi praćenje objekta preko kamere. Također implementirati sustav inkrementa za motore koji se slaže sa zubima motora tako da kretnje budu čiste i brze u isto vrijeme. A na kraju moguće je napraviti i Umjetnu inteligenciju koja bi kontrolirala robota u kretanja, a korisnik samo da zadaje smjer kretanja ili odredište.

Ovakav mali projekt je odličan za mlade inžinjere koji još nemaju isustva. Izradom ovakve konstrukcije može se puno naučiti o konstruiranju. Kako izraditi neki dio da bude jednostavan, lagan i funkcionalan. Prolaskom kroz svaku iteraciju može se vidjeti kako netko napreduje i uči na svojim greškama. Također izrada elektroničke scheme te spajanje svih komponenti je odlično iskustvo kroz koje treba proći svatko sam za sebe. Iako je schema jednostavna može se naići na razne probleme zbog manjka iskustva. Isto tako i programiranje, kroz ovaj projekt se može naučiti puno o komunikaciji između komponenti, o potrebnim funkcijama, izradi internet stranice te mnogo drugih detalja. Iako je ovdje sam program vrlo jednostavan potrebno ga je dobro oraganizirati i označavati da bi se kasnije moglo pronalaziti probleme i popravljati ih. Za nekoga ko tek ulazi u svijet programiranja ovakav mali projekt je odličan da se učvrste osnove te steknu navike koje će kasnije mnogo značiti u izradi većih projekata.

9. LITERATURA

[1]. Jerbić Nikola, Vranješ Kunica, *Projektiranje automatskih montažnih sustava*, Zagreb, 2008.
[2]. https://www.amazon.de/-/en/DANDANdianzi-DC-DC-High-Adjustable-Converter-1-2-35V/dp/B087M6SZQP/ref=sr_1_54?keywords=dc+dc+buck+wandler&qid=1643030337&sprefix=DC+dc+b%2Caps%2C272&sr=8-54, 12.9.2022
[3]. https://www.amazon.de/-/en/GOLDBAT-Connection-Aeroplane-Helicopter-Quadcopter/dp/B08ZXTB4ZM/ref=sr_1_5?crid=I5BHWX2U7G1Y&keywords=goldbat%2Blipo%2B3s%2B80c&qid=1643113594&sprefix=goldbat%2Blipo%2B3s%2B80c%2Caps%2C140&sr=8-5&th=1, 12.9.2022
[4]. https://www.robotshop.com/community/tutorials/show/robot-leg-torque-tutorial, 12.9.2022
[5]. https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet, 12.9.2022
[6]. https://loboris.eu/ESP32/ESP32-CAM%20Product%20Specification.pdf , 12.9.2022
[7]. https://components101.com/motors/mg996r-servo-motor-datasheet , 12.9.2022
[8]. https://datasheetspdf.com/pdf/791970/TowerPro/SG90/1 , 12.9.2022
[9]. https://www.hwlibre.com/hr/hc-sr04/, 12.9.2022
[10]. https://people.etf.unsa.ba/~jvelagic/laras/dok/Inverzna%20kinematika.pdf, 12.9.2022
[11.] https://ardushop.ro/en/home/1329-modul-pca9685-interfata-i2c-16-ch-servo-motor.html, 12.9.2022

10. PRILOZI

10.1. Tehničke specifikacije

Specifikacije motora MG996R prema [7.]:

- Radni napon je +5V
- Struja: 2.5A (6V)
- Zakretni moment: 0.92 Nm (at 4.8V)
- Maksimalni zakretni moment: 1 Nm (6V)
- Brzina je 0.17 s/60°
- Tip zupčanika: Metal
- Rotacija : 0°-180°
- Masa motora : 55gm

Specifikacije motora SG90 prema [8.]:

- Masa: 9g
- Dimenzije: 22.2x11.8x31 mm
- Zakretni moment: 0.18 Nm
- Brzina je 0.1 s/60°
- Radni napon je +4.8V
- Mrtvi prostor: 10µm
- Temperaturni pojas: 0^oC do 55 0^oC

Specifikacije NodeMCU 8266 prema [5.]:

- Mikroprocesor: Tensilica 32-bit RISC CPU Xtensa LX106
- Radni napon: 3.3V
- Ulazni napon: 7-12V
- Digitalni I/O Pinovi (DIO): 16
- Analogni ulazni pinovi (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB

- Clock Speed: 80 MHz
- PCB Antena

Specifikacije ESP 32 CAM prema [6.]:

Model	ESP32- CAM
Paket	DIP-16
Dimenzije	27x40x4.5mm
SPI Flash	32 Mbit
RAM	520KB SRAM + 4M PSRAM
Bluetooth	Bluetooth 4.2 BR/EDR and BLE standard
Wi-Fi	802.11 b/g/n/
Podrška sučelju	UART, SPI, I2C, PWM
Podrška TF kartice	Maksimalna podrška 4G
IO port	9
UART Baundrate	115200 bps
Izlazni format slike	JPEG(OV2640), BMP, GRAYSCALE
Pojas spektra	2412 – 2484MHz
Antena	Na pločici PCB antenna, 2dBi
Snaga odašiljanja	802.11b: 17+2dbM(@11Mbps) 802.11b: 14+2dbM(@54Mbps) 802.11b: 13+2dbM(@MCS7)
Odsjetljivost primanja	CCK, 1Mbps: -90dBm CCK, 11Mbps: -85dBm 6 Mbps (1/2 BPSK)-90dBm
Sigurnost	WPA/WPA2/WPA2 – Enterprise/WPS
Radni napon	5V
Radna temperatura	-20^{o}C do 85^{o}C
Uvjeti skladištenja	-40^{o}C do 90^{o}C, <90%RH

Specifikacije pretvornika prema [2]:

- LED inikator
- Struja: do 20A
- Efikasnost: 96A na pretvorbi sa 24V na 12V pri struji od 20A
- Ulazni napon: 6V do 40V
- Hlađenje: 2 hladnjaka

10.2. Programski kod

NodeMCU 8266 kod:

```
#include <Arduino.h>
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <Math.h>
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <ArduinoJson.h>
//#include <Arduino_JSON.h>
#include <WebSocketsServer.h>
#include <ESPAsyncWebServer.h>
#include <SpeedTrig.h>
Adafruit_PWMServoDriver pwm2 = Adafruit_PWMServoDriver(0x40);   // Initiates library.
Adafruit_PWMServoDriver pwm1 = Adafruit_PWMServoDriver(0x41);   // Initiates library.
const char* ssid = "Patcho";
const char* password = "12345678";
uint8_t socket_port;
String message = "";
WebSocketsServer webSocket = WebSocketsServer(81);
#define SERVOMIN  120  // Minimum pulse length count out of 4096.
#define SERVOMAX  500 // Maximum pulse length count out of 4096.
StaticJsonDocument<300> Senzor; // Allocate a static JSON document
StaticJsonDocument<300> Kord; // Allocate a static JSON document
StaticJsonDocument<100> doc;
StaticJsonDocument<200> Slider;
```

```
float q1_motora;
float q2_motora;
float q3_motora;
//-----------Spremanje pozicije za pomicanje nogu unaprijed po elipsi----------//
float PozXEDp[53];//Za prednje noge
float PozYEDp[53];
float PozZEDp[53];
float PozXEKp[53];//Za zadnje noge
float PozYEKp[53];
float PozZEKp[53];
float PozXEs[53];
float PozYEs[53];
float PozZEs[53];
float q1_motora_PrednjeNogeE[53];
float q2_motora_PrednjeNogeE[53];
float q3_motora_PrednjeNogeE[53];
float q1_motora_SrednjeNogeE[53];
float q2_motora_SrednjeNogeE[53];
float q3_motora_SrednjeNogeE[53];
float q1_motora_StraznjeNogeE[53];
float q2_motora_StraznjeNogeE[53];
float q3_motora_StraznjeNogeE[53];
//-----------Spremanje pozicije za pomicanje tijela unaprijed po pravcu----------//
float PozXPDp[53];
float PozYPDp[53];
float PozZPDp[53];
float PozXPKp[53];
```

```
float PozYPKp[53];
float PozZPKp[53];
float PozXPs[53];
float PozYPs[53];
float PozZPs[53];
float q1_motora_PrednjeNogeP[53];
float q2_motora_PrednjeNogeP[53];
float q3_motora_PrednjeNogeP[53];
float q1_motora_SrednjeNogeP[53];
float q2_motora_SrednjeNogeP[53];
float q3_motora_SrednjeNogeP[53];
float q1_motora_StraznjeNogeP[53];
float q2_motora_StraznjeNogeP[53];
float q3_motora_StraznjeNogeP[53];
//-------------------------------------------------------------------------------//
float brojnik_q1;
float nazivnik_q1;
float brojnik_q2;
float nazivnik_q2;
float brojnik_q3;
float nazivnik_q3;
float L1 = 46;
float L2 = 89;
float L3 = 148.5;
float PozX;
float PozY;
float PozZ;
```

```
float X1 = 80;
float X2 = 80;
float Y1 = 80;
float Y2 = 20;
float Z1 = -140;
float Z2 = -40;
float X1Z = 80;
float X2Z = 80;
float Y1Z = 29;
float Y2Z = -29;
float X1S = 80;
float X2S = 80;
float Y1S = 29;
float Y2S = -29;
int BrojSlidera;
float VrijednostSlidera;
String BrojNoge;
String DioNoge;
int Pinn;
int DELAY = 7;
String Orjentacija = "STOP";
String StaraOrjentacija = "STOP";
String Glide = "OFF";
float Vrijednost_senzora;
//Prva Noga - prednja lijeva strana
int Kuk_1 = 13;
int Zglob_1 = 14;
```

```
int Koljeno_1 = 15;
//Druga Noga - prednja desna strana - PWM 2, na drugoj pločici
int Kuk_2 = 4;
int Zglob_2 = 5;
int Koljeno_2 = 6;
//Treca Noga - sredina lijeva strana
int Kuk_3 = 12;
int Zglob_3 = 13;
int Koljeno_3 = 14;
//Cetvrta Noga - srednja desna strana - PWM 2, na drugoj pločici
int Kuk_4 = 0;
int Zglob_4 = 1;
int Koljeno_4 = 2;
//Peta Noga - Zadnja lijeva strana
int Kuk_5 = 8;
int Zglob_5 = 9;
int Koljeno_5 = 10;
//Sesta Noga - Zadnja desna strana - PWM 2, na drugoj pločici
int Kuk_6 = 0;
int Zglob_6 = 1;
int Koljeno_6 = 2;
int Servo_senzora = 8; // PWM 1
void setup() {
  Serial.begin(115200);
  pwm1.begin();         // Sends PWM signals.
  pwm1.setPWMFreq(50);  // Makes servos run at 60 Hz rate.
  delay(20);
```

```
  pwm2.begin();         // Sends PWM signals.
  pwm2.setPWMFreq(50);  // Makes servos run at 60 Hz rate.
  delay(20);
  Wifi();
  webSocket.begin();
  webSocket.onEvent(webSocketEvent);
}
void loop() {
  webSocket.loop();
  if (Orjentacija == "STOP") {
  }
  else if (Orjentacija == "TrecaNoga" || Orjentacija == "PetaNoga") {
    if (BrojSlidera == 3) {
      ZakretMotoraPWM_35K(-VrijednostSlidera, Pinn);
      Orjentacija = "STOP";
    }
    else {
      ZakretMotoraPWM_356(VrijednostSlidera, Pinn);
      Orjentacija = "STOP";
    }
  }
  else if (Orjentacija == "SestaNoga") {
    if (BrojSlidera == 3) {
      ZakretMotoraPWM_6K(VrijednostSlidera, Pinn);
      Orjentacija = "STOP";
    }
    else {
```

```
    ZakretMotoraPWM_356(-VrijednostSlidera, Pinn);
    Orjentacija = "STOP";
   }
  }
  else if (Orjentacija == "DrugaNoga" || Orjentacija == "CetvrtaNoga") {
   if (BrojSlidera == 3) {
    ZakretMotoraPWM_24K(VrijednostSlidera, Pinn);
    Orjentacija = "STOP";
   }
   else {
    ZakretMotoraPWM_124(-VrijednostSlidera, Pinn);
    Orjentacija = "STOP";
   }
  }
  else if (Orjentacija == "PrvaNoga") {
   if (BrojSlidera == 3) {
    ZakretMotoraPWM_1K(-VrijednostSlidera, Pinn);
    Orjentacija = "STOP";
   }
   else {
    ZakretMotoraPWM_124(VrijednostSlidera, Pinn);
    Orjentacija = "STOP";
   }
  }
  else {
   if (Orjentacija == "PocetniPolozaj") {
    Nulti_korak();
```

```
    Orjentacija = "STOP";
   }
   else {
     if(Orjentacija != StaraOrjentacija){
      if (Glide == "OFF"){
         Racunanje(X1, X2, X1S, X2S, Y1, Y2, Y1S, Y2S, Z1, Z2);
         StaraOrjentacija = Orjentacija;
        }
      else if (Glide == "ON"){
         RacunanjeGlide(X1, X2, X1S, X2S, X1Z, X2Z, Y1, Y2, Y1S, Y2S, Y1Z, Y2Z, Z1, Z2);
         StaraOrjentacija = Orjentacija;
        }
      else{

        }
     }
     else if (Orjentacija == StaraOrjentacija){
     Kretanje_unaprijed();
    }
   }
  }
}
void Nulti_korak() {
  // Ovdje su svi motori stavljanei da se okrecu u pozitivnom smjeru, pa
  // gdje je minus, to znači da je pozitivan smjer za motor, ovakvim rasporedom
  // se ne bude motor zabio u podlogu, ili drugu susjednu nogu
  //Prednje noge
```

```
float prednja1Kuk = 25;
float prednja2Kuk = 25;
float prednja1Zglob = 33;
float prednja2Zglob = 33;
float prednja1Koljeno = 125;
float prednja2Koljeno = 125;
//Srednje noge
float srednji3Kuk = 0;
float srednji4Kuk = 0;
float srednji3Zglob = 33;
float srednji4Zglob = 33;
float srednji3Koljeno = 105;
float srednji4Koljeno = 125;
//Zadnje noge
float zadnja5Kuk = 25;
float zadnja6Kuk = 25;
float zadnja5Zglob = 33;
float zadnja6Zglob = 33;
float zadnja5Koljeno = 125;
float zadnja6Koljeno = 125;
ZakretMotoraPWM_124(prednja1Zglob, Zglob_1);
ZakretMotoraPWM_124(-prednja2Zglob, Zglob_2);
delay(300);
ZakretMotoraPWM_356(srednji3Zglob, Zglob_3);
ZakretMotoraPWM_124(-srednji4Zglob, Zglob_4);
delay(300);
ZakretMotoraPWM_356(zadnja5Zglob, Zglob_5);
```

```
  ZakretMotoraPWM_356(-zadnja6Zglob, Zglob_6);
  delay(1000);
  ZakretMotoraPWM_124(-prednja1Kuk, Kuk_1);
  ZakretMotoraPWM_124(prednja2Kuk, Kuk_2);
  ZakretMotoraPWM_356(srednji3Kuk, Kuk_3);
  ZakretMotoraPWM_124(srednji4Kuk, Kuk_4);
  ZakretMotoraPWM_356(zadnja5Kuk, Kuk_5);
  ZakretMotoraPWM_356(-zadnja6Kuk, Kuk_6);
  delay(1000);
  ZakretMotoraPWM_1K(-prednja1Koljeno, Koljeno_1);
  ZakretMotoraPWM_24K(prednja2Koljeno, Koljeno_2);
  delay(100);
  ZakretMotoraPWM_35K(-srednji3Koljeno, Koljeno_3);
  ZakretMotoraPWM_24K(srednji4Koljeno, Koljeno_4);
  delay(100);
  ZakretMotoraPWM_35K(-zadnja5Koljeno, Koljeno_5);
  ZakretMotoraPWM_6K(zadnja6Koljeno, Koljeno_6);
  delay(1000);
}
void Ocitavanje(const int trigPin, const int echoPin) {
  ZakretMotoraPWM_24K(0, Servo_senzora);
  delay(300);
  Vrijednost_senzora = Udaljenost(trigPin, echoPin);
  Serial.println(Vrijednost_senzora);
  delay(200);
  Senzor["Desno"] = Vrijednost_senzora;
  ZakretMotoraPWM_24K(180, Servo_senzora);
```

```
  delay(300);
  Vrijednost_senzora = Udaljenost(trigPin, echoPin);
  Serial.println(Vrijednost_senzora);
  delay(200);
  Senzor["Lijevo"] = Vrijednost_senzora;
  ZakretMotoraPWM_24K(90, Servo_senzora);
  delay(300);
  Vrijednost_senzora = Udaljenost(trigPin, echoPin);
  Serial.println(Vrijednost_senzora);
  delay(200);
  Senzor["Ispred"] = Vrijednost_senzora;
  char dataa[150];
  size_t len = serializeJson(Senzor, dataa);
  webSocket.broadcastTXT(dataa, len);
}
float Udaljenost(const int trigPin, const int echoPin ) {
  long duration ;
  float distance;
  //define sound velocity in cm/uS
#define SOUND_VELOCITY 0.0343
  // put your main code here, to run repeatedly:
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT);
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
```

```
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculate the distance
  distance = (duration * SOUND_VELOCITY) / 2;
  return distance;
}
void Prva_Noga(float q1_naprijed, float q2_naprijed, float q3_naprijed) {
  ZakretMotoraPWM_124(-q1_naprijed, Kuk_1);
  ZakretMotoraPWM_124(q2_naprijed, Zglob_1);
  ZakretMotoraPWM_1K(-q3_naprijed, Koljeno_1);
}
void Druga_Noga(float q1_naprijed, float q2_naprijed, float q3_naprijed) {
  ZakretMotoraPWM_124(q1_naprijed, Kuk_2);
  ZakretMotoraPWM_124(-q2_naprijed, Zglob_2);
  ZakretMotoraPWM_24K(q3_naprijed, Koljeno_2);
}
void Treca_Noga(float q1_naprijed, float q2_naprijed, float q3_naprijed) {
  ZakretMotoraPWM_356(-q1_naprijed, Kuk_3);
  ZakretMotoraPWM_356(q2_naprijed, Zglob_3);
  ZakretMotoraPWM_35K(-q3_naprijed+20, Koljeno_3);
}
void Cetvrta_Noga(float q1_naprijed, float q2_naprijed, float q3_naprijed) {
  ZakretMotoraPWM_124(q1_naprijed, Kuk_4);
  ZakretMotoraPWM_124(-q2_naprijed, Zglob_4);
  ZakretMotoraPWM_24K(q3_naprijed, Koljeno_4);
```

```
}
void Peta_Noga(float q1_naprijed, float q2_naprijed, float q3_naprijed) {
  ZakretMotoraPWM_356(q1_naprijed, Kuk_5);
  ZakretMotoraPWM_356(q2_naprijed, Zglob_5);
  ZakretMotoraPWM_35K(-q3_naprijed-15, Koljeno_5);
}
void Sesta_Noga(float q1_naprijed, float q2_naprijed, float q3_naprijed) {
  ZakretMotoraPWM_356(-q1_naprijed, Kuk_6);
  ZakretMotoraPWM_356(-q2_naprijed, Zglob_6);
  ZakretMotoraPWM_6K(q3_naprijed, Koljeno_6);
}
void Izracun(double x, double y, double z) {
  brojnik_q1 = y;
  nazivnik_q1 = x;
  double c3 = (sq(x) + sq(y) + sq(z) - sq(L2) - sq(L3)) / (2 * L2 * L3);
  if ((c3 <= -1) || (c3 >= 1)) {
    Serial.println("Nisu dobri brojevi brojevi");
  }
  else {
    nazivnik_q3 = c3;
    double s3 = -sqrt(1 - (c3 * c3));
    brojnik_q3 = s3;

    brojnik_q2 = (L2 + L3 * c3) * z - L3 * s3 * sqrt((x * x) + (y * y));
    nazivnik_q2 = (L2 + L3 * c3) * sqrt((x * x) + (y * y)) + L3 * s3 * z;
  }
  if (x < 0) {
```

```
    q1_motora = -degrees(PI - SpeedTrig.atan2(brojnik_q1, nazivnik_q1));
  }
  else {
    q1_motora = degrees(SpeedTrig.atan2(brojnik_q1, nazivnik_q1));
  }
  q2_motora = degrees(SpeedTrig.atan2(brojnik_q2, nazivnik_q2));
  q3_motora = PI - degrees(SpeedTrig.atan2(brojnik_q3, nazivnik_q3));
}
void elipsa(double x1, double x2, double y1, double y2, double z1, double z2, double v, int t) {
  double a = (y2 - y1) / 2;
  double b = (z2 - z1) / 2;
  double h = (y2 + y1) / 2;
  double k = (z2 + z1) / 2;
  double c = (x2 - x1) / 2;
  double n = (x2 + x1) / 2;
  int Maxt = 50;
  PozY = a * v * cos(t * (PI / Maxt)) + h;
  PozZ = b * v * sin(t * (PI / Maxt)) + k;
  if (x2 - x1 == 0) {
    PozX = x1;
  }
  else {
    PozX = v * c * cos(t * (PI / Maxt)) + n;
  }
}
void Pravac(double x1, double x2, double y1, double y2, double z1, double z2, float v, int t) {
  double a = (y2 - y1) / 2;
```

```
  double b = (z2 + z1) / 2;
  double h = (y2 + y1) / 2;
  double c = (x2 - x1) / 2;
  double n = (x2 + x1) / 2;
  int Maxt = 50;
  PozY = a * v * cos(t * (PI / Maxt)) + h;
  PozZ = b;
  if (x2 - x1 == 0) {
    PozX = x1;
  }
  else {
    PozX = v * c * cos(t * (PI / Maxt)) + n;
  }
}
void Kruznica(float r, float z1, float z2, float GornjaGranica, float DoljnjaGranica, float PomakPoX, float PomakPoY, float t) {
  double b = (z2 + z1) / 2;
  int Maxt = 50;
  PozY = r * sin((t / Maxt) * (PI / 2) * radians(GornjaGranica) + radians(DoljnjaGranica)) + PomakPoY;
  PozX = r * cos((t / Maxt) * (PI / 2) * radians(GornjaGranica) + radians(DoljnjaGranica)) + PomakPoX;
  PozZ = b;
}
void ZakretMotoraPWM_1K(float ang, int servo) {
  int pulse = map(ang, -180, 0, SERVOMIN, SERVOMAX);
  pwm1.setPWM(servo, 0, pulse);
}
```

```
void ZakretMotoraPWM_35K(float ang, int servo) {
  int pulse = map(ang, -180, 0, SERVOMIN, SERVOMAX);
  pwm2.setPWM(servo, 0, pulse);
}
void ZakretMotoraPWM_6K(float ang, int servo) {
  int pulse = map(ang, 0, 180, SERVOMIN, SERVOMAX);
  pwm2.setPWM(servo, 0, pulse);
}
void ZakretMotoraPWM_24K(float ang, int servo) {
  int pulse = map(ang, 0, 180, SERVOMIN, SERVOMAX);
  pwm1.setPWM(servo, 0, pulse);
}
void ZakretMotoraPWM_124(float ang, int servo) {
  int pulse = map(ang, -90, 90, SERVOMIN, SERVOMAX);
  pwm1.setPWM(servo, 0, pulse);
}
void ZakretMotoraPWM_356(float ang, int servo) {
  int pulse = map(ang, -90, 90, SERVOMIN, SERVOMAX);
  pwm2.setPWM(servo, 0, pulse);
}
void Wifi() {
  // Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.print("Go to: http://");
  Serial.println(WiFi.localIP());
}
void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t len) {
  switch (type) {
    case WStype_DISCONNECTED:
      Serial.printf("[%u]NodeMCU Disconnected!\n");
      break;
    case WStype_CONNECTED: {
        webSocket.sendTXT(num, "NodeMCU Connected");
        Serial.printf("NodeMCU Connected");
        socket_port = num;
      }
      break;
    case WStype_TEXT: {
        handleWebSocketMessage(num, payload, len);
      }
      break;
  }
}
void handleWebSocketMessage(uint8_t arg, uint8_t *data, size_t len) {
  message = (char*)data;
  if (message.indexOf("Values") >= 0) {
    Ocitavanje(0, 2);
  }
```

```
  else if (message.indexOf("XX") >= 0) {
    String Value = message.substring(2);
    deserializeJson(Slider, Value);
    Orjentacija = Slider["Orjentacija"].as<String>();
    BrojSlidera = Slider["BrojSlidera"];
    VrijednostSlidera = Slider["VrijednostSlidera"];
    IzborNoge();
    IzborDijelaNoge();
    IzborPina();
  }
  else {
    String Value = message.substring(2);
    deserializeJson(Kord, Value);
    DELAY = Kord["Delay"];
    Z1 = Kord["Z1"];
    Orjentacija = Kord["Orjentacija"].as<String>();
    Glide = Kord["Glide"].as<String>();
  }
}
float Skaliranje(float x, float in_min, float in_max, float out_min, float out_max) {
  float nazivnik = (in_min - in_min) + out_min;
  if (nazivnik == 0) {
    nazivnik = 0.1;
  }
  return (x - in_min) * (out_max - out_min) / nazivnik;
}
void IzborDijelaNoge() {
```

```
  if (BrojSlidera == 1) {
    DioNoge = "Kuk_";
  }
  if (BrojSlidera == 2) {
    DioNoge = "Zglob_";
  }
  if (BrojSlidera == 3) {
    DioNoge = "Koljeno_";
  }
}
void IzborNoge() {
  if (Orjentacija == "PrvaNoga") {
    BrojNoge = "1";
  }
  else if (Orjentacija == "DrugaNoga") {
    BrojNoge = "2";
  }
  else if (Orjentacija == "TrecaNoga") {
    BrojNoge = "3";
  }
  else if (Orjentacija == "CetvrtaNoga") {
    BrojNoge = "4";
  }
  else if (Orjentacija == "PetaNoga") {
    BrojNoge = "5";
  }
  else if (Orjentacija == "SestaNoga") {
```

```
    BrojNoge = "6";
  }
  else {
    BrojNoge = "0";
  }
}
void IzborPina() {
  if (DioNoge + BrojNoge == "Kuk_1") {
    Pinn = Kuk_1;
  }
  else if (DioNoge + BrojNoge == "Kuk_2") {
    Pinn = Kuk_2;
  }
  else if (DioNoge + BrojNoge == "Kuk_3") {
    Pinn = Kuk_3;
  }
  else if (DioNoge + BrojNoge == "Kuk_4") {
    Pinn = Kuk_4;
  }
  else if (DioNoge + BrojNoge == "Kuk_5") {
    Pinn = Kuk_5;
  }
  else if (DioNoge + BrojNoge == "Kuk_6") {
    Pinn = Kuk_6;
  }
  else if (DioNoge + BrojNoge == "Zglob_1") {
    Pinn = Zglob_1;
```

```
  }
  else if (DioNoge + BrojNoge == "Zglob_2") {
    Pinn = Zglob_2;
  }
  else if (DioNoge + BrojNoge == "Zglob_3") {
    Pinn = Zglob_3;
  }
  else if (DioNoge + BrojNoge == "Zglob_4") {
    Pinn = Zglob_4;
  }
  else if (DioNoge + BrojNoge == "Zglob_5") {
    Pinn = Zglob_5;
  }
  else if (DioNoge + BrojNoge == "Zglob_6") {
    Pinn = Zglob_6;
  }
  else if (DioNoge + BrojNoge == "Koljeno_1") {
    Pinn = Koljeno_1;
  }
  else if (DioNoge + BrojNoge == "Koljeno_2") {
    Pinn = Koljeno_2;
  }
  else if (DioNoge + BrojNoge == "Koljeno_3") {
    Pinn = Koljeno_3;
  }
  else if (DioNoge + BrojNoge == "Koljeno_4") {
    Pinn = Koljeno_4;
```

```
  }
  else if (DioNoge + BrojNoge == "Koljeno_5") {
    Pinn = Koljeno_5;
  }
  else if (DioNoge + BrojNoge == "Koljeno_6") {
    Pinn = Koljeno_6;
  }
}
void Kretanje_unaprijed() {
  for (int i = 0; i <= 100; i++) {
    if (Orjentacija == "STOP") {
      break;
    }
    if ((0 <= i) && (i <= 50)) {
      if (Orjentacija == "lijevo" && Glide == "OFF") {
      }
      else if (Orjentacija == "Nazad desno" && Glide == "OFF") {
        Prva_Noga(q1_motora_PrednjeNogeE[50-i], q2_motora_PrednjeNogeE[50-i],
q3_motora_PrednjeNogeE[50-i]);
        Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i],
q3_motora_SrednjeNogeE[i]);
        Peta_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i],
q3_motora_PrednjeNogeE[i]);
      }
      else if (Orjentacija == "Nazad lijevo" && Glide == "OFF") {
        Prva_Noga(q1_motora_StraznjeNogeE[i], q2_motora_StraznjeNogeE[i],
q3_motora_StraznjeNogeE[i]);
        Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i],
q3_motora_SrednjeNogeE[i]);
```

```
    Peta_Noga(q1_motora_StraznjeNogeE[50 - i], q2_motora_StraznjeNogeE[50 - i], q3_motora_StraznjeNogeE[50 - i]);

  }

  else if (Orjentacija == "Lagano lijevo" && Glide == "OFF") {

    Prva_Noga(q1_motora_StraznjeNogeE[50-i], q2_motora_StraznjeNogeE[50-i], q3_motora_StraznjeNogeE[50-i]);

    Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);

    Peta_Noga(q1_motora_StraznjeNogeE[i], q2_motora_StraznjeNogeE[i], q3_motora_StraznjeNogeE[i]);

  }

  else if (Orjentacija == "Lagano desno" && Glide == "OFF") { //Lagano desno

    Prva_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i], q3_motora_PrednjeNogeE[i]);

    Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);

    Peta_Noga(q1_motora_PrednjeNogeE[50-i], q2_motora_PrednjeNogeE[50-i], q3_motora_PrednjeNogeE[50-i]);

  }

  else if (Orjentacija == "Strogo lijevo" && Glide == "OFF") {

    Prva_Noga(q1_motora_PrednjeNogeE[50-i], q2_motora_PrednjeNogeE[50-i], q3_motora_PrednjeNogeE[50-i]);

    Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);

    Peta_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i], q3_motora_PrednjeNogeE[i]);

  }

  else if (Orjentacija == "Strogo desno" && Glide == "OFF") {

    Prva_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i], q3_motora_PrednjeNogeE[i]);
```

```
      Cetvrta_Noga(q1_motora_SrednjeNogeE[50-i], q2_motora_SrednjeNogeE[50-i], q3_motora_SrednjeNogeE[50-i]);

      Peta_Noga(q1_motora_PrednjeNogeE[50-i], q2_motora_PrednjeNogeE[50-i], q3_motora_PrednjeNogeE[50-i]);

    }

   else if((Orjentacija == "Nazad lijevo" || Orjentacija == "Lagano desno") && Glide == "ON"){

      Prva_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i], q3_motora_PrednjeNogeE[i]);

      Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);

      Peta_Noga(q1_motora_StraznjeNogeE[50-i], q2_motora_StraznjeNogeE[50-i], q3_motora_StraznjeNogeE[50-i]);

    }

   else if((Orjentacija == "Lagano lijevo" || Orjentacija == "Nazad desno") && Glide == "ON"){

      Prva_Noga(q1_motora_StraznjeNogeE[i], q2_motora_StraznjeNogeE[i], q3_motora_StraznjeNogeE[i]);

      Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);

      Peta_Noga(q1_motora_PrednjeNogeE[50-i], q2_motora_PrednjeNogeE[50-i], q3_motora_PrednjeNogeE[50-i]);

    }

   else if((Orjentacija == "Strogo lijevo" || Orjentacija == "Strogo desno") && Glide == "ON"){

      Prva_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i], q3_motora_PrednjeNogeE[i]);

      Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i], q3_motora_SrednjeNogeE[i]);

      Peta_Noga(q1_motora_StraznjeNogeE[i], q2_motora_StraznjeNogeE[i], q3_motora_StraznjeNogeE[i]);

    }
```

```
    else {
      Prva_Noga(q1_motora_PrednjeNogeE[i], q2_motora_PrednjeNogeE[i],
q3_motora_PrednjeNogeE[i]);
      Cetvrta_Noga(q1_motora_SrednjeNogeE[i], q2_motora_SrednjeNogeE[i],
q3_motora_SrednjeNogeE[i]);
      Peta_Noga(q1_motora_StraznjeNogeE[i], q2_motora_StraznjeNogeE[i],
q3_motora_StraznjeNogeE[i]);
    }
    delay(DELAY);
  }
  if ((0 <= i) && (i <= 50)) {
    int t = 50 - i;
    if (Orjentacija == "lijevo" && Glide == "OFF") {
    }
    else if(Orjentacija == "Nazad desno" && Glide == "OFF"){
      Druga_Noga(q1_motora_StraznjeNogeP[t], q2_motora_StraznjeNogeP[t],
q3_motora_StraznjeNogeP[t]);
      Treca_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t],
q3_motora_SrednjeNogeP[t]);
      Sesta_Noga(q1_motora_StraznjeNogeP[50-t], q2_motora_StraznjeNogeP[50-t],
q3_motora_StraznjeNogeP[50-t]);
    }
    else if (Orjentacija == "Nazad lijevo" && Glide == "OFF") {
      Druga_Noga(q1_motora_PrednjeNogeP[50 - t], q2_motora_PrednjeNogeP[50 - t],
q3_motora_PrednjeNogeP[50 - t]);
      Treca_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t],
q3_motora_SrednjeNogeP[t]);
      Sesta_Noga(q1_motora_PrednjeNogeP[t], q2_motora_PrednjeNogeP[t],
q3_motora_PrednjeNogeP[t]);
    }
```

```
    else if(Orjentacija == "Lagano lijevo" && Glide == "OFF"){
      Druga_Noga(q1_motora_PrednjeNogeP[t], q2_motora_PrednjeNogeP[t], q3_motora_PrednjeNogeP[t]);
      Treca_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t], q3_motora_SrednjeNogeP[t]);
      Sesta_Noga(q1_motora_PrednjeNogeP[50-t], q2_motora_PrednjeNogeP[50-t], q3_motora_PrednjeNogeP[50-t]);
    }
    else if(Orjentacija == "Lagano desno" && Glide == "OFF"){
      Druga_Noga(q1_motora_StraznjeNogeP[50-t], q2_motora_StraznjeNogeP[50-t], q3_motora_StraznjeNogeP[50-t]);
      Treca_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t], q3_motora_SrednjeNogeP[t]);
      Sesta_Noga(q1_motora_StraznjeNogeP[t], q2_motora_StraznjeNogeP[t], q3_motora_StraznjeNogeP[t]);
    }
    else if(Orjentacija == "Strogo lijevo" && Glide == "OFF"){
      Druga_Noga(q1_motora_StraznjeNogeP[50-t], q2_motora_StraznjeNogeP[50-t], q3_motora_StraznjeNogeP[50-t]);
      Treca_Noga(q1_motora_SrednjeNogeP[50-t], q2_motora_SrednjeNogeP[50-t], q3_motora_SrednjeNogeP[50-t]);
      Sesta_Noga(q1_motora_StraznjeNogeP[t], q2_motora_StraznjeNogeP[t], q3_motora_StraznjeNogeP[t]);
    }
    else if(Orjentacija == "Strogo desno" && Glide == "OFF"){
      Druga_Noga(q1_motora_StraznjeNogeP[t], q2_motora_StraznjeNogeP[t], q3_motora_StraznjeNogeP[t]);
      Treca_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t], q3_motora_SrednjeNogeP[t]);
      Sesta_Noga(q1_motora_StraznjeNogeP[50-t], q2_motora_StraznjeNogeP[50-t], q3_motora_StraznjeNogeP[50-t]);
```

```
    }
    else if((Orjentacija == "Nazad lijevo" || Orjentacija == "Lagano desno") && Glide == "ON"){
      Druga_Noga(q1_motora_StraznjeNogeP[t], q2_motora_StraznjeNogeP[t], q3_motora_StraznjeNogeP[t]);
      Treca_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t], q3_motora_SrednjeNogeP[t]);
      Sesta_Noga(q1_motora_PrednjeNogeP[50-t], q2_motora_PrednjeNogeP[50-t], q3_motora_PrednjeNogeP[50-t]);
    }
    else if((Orjentacija == "Lagano lijevo" || Orjentacija == "Nazad desno") && Glide == "ON"){
      Druga_Noga(q1_motora_PrednjeNogeP[t], q2_motora_PrednjeNogeP[t], q3_motora_PrednjeNogeP[t]);
      Treca_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t], q3_motora_SrednjeNogeP[t]);
      Sesta_Noga(q1_motora_StraznjeNogeP[50-t], q2_motora_StraznjeNogeP[50-t], q3_motora_StraznjeNogeP[50-t]);
    }
    else if((Orjentacija == "Strogo lijevo" || Orjentacija == "Strogo desno") && Glide == "ON"){
      Druga_Noga(q1_motora_StraznjeNogeP[50-t], q2_motora_StraznjeNogeP[50-t], q3_motora_StraznjeNogeP[50-t]);
      Treca_Noga(q1_motora_SrednjeNogeP[50-t], q2_motora_SrednjeNogeP[50-t], q3_motora_SrednjeNogeP[50-t]);
      Sesta_Noga(q1_motora_PrednjeNogeP[50-t], q2_motora_PrednjeNogeP[50-t], q3_motora_PrednjeNogeP[50-t]);
    }
    else {
      Druga_Noga(q1_motora_PrednjeNogeP[t], q2_motora_PrednjeNogeP[t], q3_motora_PrednjeNogeP[t]);
```

```
      Treca_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t],
q3_motora_SrednjeNogeP[t]);

      Sesta_Noga(q1_motora_StraznjeNogeP[t], q2_motora_StraznjeNogeP[t],
q3_motora_StraznjeNogeP[t]);

    }

    delay(DELAY);

  }

  if ((50 <= i) && (i <= 100)) {

    int t = i - 50;

    if(Orjentacija == "Nazad desno" && Glide == "OFF"){

      Druga_Noga(q1_motora_StraznjeNogeE[t], q2_motora_StraznjeNogeE[t],
q3_motora_StraznjeNogeE[t]);

      Treca_Noga(q1_motora_SrednjeNogeE[t], q2_motora_SrednjeNogeE[t],
q3_motora_SrednjeNogeE[t]);

      Sesta_Noga(q1_motora_StraznjeNogeE[50-t], q2_motora_StraznjeNogeE[50-t],
q3_motora_StraznjeNogeE[50-t]);

    }

    else if (Orjentacija == "Nazad lijevo" && Glide == "OFF") {

      Druga_Noga(q1_motora_PrednjeNogeE[50-t], q2_motora_PrednjeNogeE[50 - t],
q3_motora_PrednjeNogeE[50 - t]);

      Treca_Noga(q1_motora_SrednjeNogeE[t], q2_motora_SrednjeNogeE[t],
q3_motora_SrednjeNogeE[t]);

      Sesta_Noga(q1_motora_PrednjeNogeE[t], q2_motora_PrednjeNogeE[t],
q3_motora_PrednjeNogeE[t]);

    }

    else if(Orjentacija == "Lagano lijevo" && Glide == "OFF"){

      Druga_Noga(q1_motora_PrednjeNogeE[t], q2_motora_PrednjeNogeE[t],
q3_motora_PrednjeNogeE[t]);

      Treca_Noga(q1_motora_SrednjeNogeE[t], q2_motora_SrednjeNogeE[t],
q3_motora_SrednjeNogeE[t]);
```

```
    Sesta_Noga(q1_motora_PrednjeNogeE[50-t], q2_motora_PrednjeNogeE[50-t], q3_motora_PrednjeNogeE[50-t]);

  }

  else if(Orjentacija == "Lagano desno" && Glide == "OFF"){

    Druga_Noga(q1_motora_StraznjeNogeE[50-t], q2_motora_StraznjeNogeE[50-t], q3_motora_StraznjeNogeE[50-t]);

    Treca_Noga(q1_motora_SrednjeNogeE[t], q2_motora_SrednjeNogeE[t], q3_motora_SrednjeNogeE[t]);

    Sesta_Noga(q1_motora_StraznjeNogeE[t], q2_motora_StraznjeNogeE[t], q3_motora_StraznjeNogeE[t]);

  }

  else if(Orjentacija == "Strogo lijevo" && Glide == "OFF"){

    Druga_Noga(q1_motora_StraznjeNogeE[50-t], q2_motora_StraznjeNogeE[50-t], q3_motora_StraznjeNogeE[50-t]);

    Treca_Noga(q1_motora_SrednjeNogeE[50-t], q2_motora_SrednjeNogeE[50-t], q3_motora_SrednjeNogeE[50-t]);

    Sesta_Noga(q1_motora_StraznjeNogeE[t], q2_motora_StraznjeNogeE[t], q3_motora_StraznjeNogeE[t]);

  }

  else if(Orjentacija == "Strogo desno" && Glide == "OFF"){

    Druga_Noga(q1_motora_StraznjeNogeE[t], q2_motora_StraznjeNogeE[t], q3_motora_StraznjeNogeE[t]);

    Treca_Noga(q1_motora_SrednjeNogeE[t], q2_motora_SrednjeNogeE[t], q3_motora_SrednjeNogeE[t]);

    Sesta_Noga(q1_motora_StraznjeNogeE[50-t], q2_motora_StraznjeNogeE[50-t], q3_motora_StraznjeNogeE[50-t]);

  }

  else if((Orjentacija == "Nazad lijevo" || Orjentacija == "Lagano desno") && Glide == "ON"){

    Druga_Noga(q1_motora_StraznjeNogeE[t], q2_motora_StraznjeNogeE[t], q3_motora_StraznjeNogeE[t]);
```

```
      Treca_Noga(q1_motora_SrednjeNogeE[t], q2_motora_SrednjeNogeE[t], q3_motora_SrednjeNogeE[t]);

      Sesta_Noga(q1_motora_PrednjeNogeE[50-t], q2_motora_PrednjeNogeE[50-t], q3_motora_PrednjeNogeE[50-t]);

    }

    else if((Orjentacija == "Lagano lijevo" || Orjentacija == "Nazad desno") && Glide == "ON"){

      Druga_Noga(q1_motora_PrednjeNogeE[t], q2_motora_PrednjeNogeE[t], q3_motora_PrednjeNogeE[t]);

      Treca_Noga(q1_motora_SrednjeNogeE[t], q2_motora_SrednjeNogeE[t], q3_motora_SrednjeNogeE[t]);

      Sesta_Noga(q1_motora_StraznjeNogeE[50-t], q2_motora_StraznjeNogeE[50-t], q3_motora_StraznjeNogeE[50-t]);

    }

    else if((Orjentacija == "Strogo lijevo" || Orjentacija == "Strogo desno") && Glide == "ON"){

      Druga_Noga(q1_motora_StraznjeNogeE[50-t], q2_motora_StraznjeNogeE[50-t], q3_motora_StraznjeNogeE[50-t]);

      Treca_Noga(q1_motora_SrednjeNogeE[50-t], q2_motora_SrednjeNogeE[50-t], q3_motora_SrednjeNogeE[50-t]);

      Sesta_Noga(q1_motora_PrednjeNogeE[50-t], q2_motora_PrednjeNogeE[50-t], q3_motora_PrednjeNogeE[50-t]);

    }

    else {

      Druga_Noga(q1_motora_PrednjeNogeE[t], q2_motora_PrednjeNogeE[t], q3_motora_PrednjeNogeE[t]);

      Treca_Noga(q1_motora_SrednjeNogeE[t], q2_motora_SrednjeNogeE[t], q3_motora_SrednjeNogeE[t]);

      Sesta_Noga(q1_motora_StraznjeNogeE[t], q2_motora_StraznjeNogeE[t], q3_motora_StraznjeNogeE[t]);

    }

    delay(DELAY);
```

```
  }
  if ((50 <= i) && (i <= 100)) {
    int t = 100 - i;
    if(Orjentacija == "Nazad desno" && Glide == "OFF"){
      Prva_Noga(q1_motora_PrednjeNogeP[50-t], q2_motora_PrednjeNogeP[50-t],
q3_motora_PrednjeNogeP[50-t]);
      Cetvrta_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t],
q3_motora_SrednjeNogeP[t]);
      Peta_Noga(q1_motora_PrednjeNogeP[t], q2_motora_PrednjeNogeP[t],
q3_motora_PrednjeNogeP[t]);
    }
    else if (Orjentacija == "Nazad lijevo" && Glide == "OFF") {
      Prva_Noga(q1_motora_StraznjeNogeP[t], q2_motora_StraznjeNogeP[t],
q3_motora_StraznjeNogeP[t]);
      Cetvrta_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t],
q3_motora_SrednjeNogeP[t]);
      Peta_Noga(q1_motora_StraznjeNogeP[50-t], q2_motora_StraznjeNogeP[50-t],
q3_motora_StraznjeNogeP[50-t]);
    }
    else if(Orjentacija == "Lagano lijevo" && Glide == "OFF"){
      Prva_Noga(q1_motora_StraznjeNogeP[50-t], q2_motora_StraznjeNogeP[50-t],
q3_motora_StraznjeNogeP[50-t]);
      Cetvrta_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t],
q3_motora_SrednjeNogeP[t]);
      Peta_Noga(q1_motora_StraznjeNogeP[t], q2_motora_StraznjeNogeP[t],
q3_motora_StraznjeNogeP[t]);
    }
    else if(Orjentacija == "Lagano desno" && Glide == "OFF"){
      Prva_Noga(q1_motora_PrednjeNogeP[t], q2_motora_PrednjeNogeP[t],
q3_motora_PrednjeNogeP[t]);
```

```
      Cetvrta_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t], q3_motora_SrednjeNogeP[t]);

      Peta_Noga(q1_motora_PrednjeNogeP[50-t], q2_motora_PrednjeNogeP[50-t], q3_motora_PrednjeNogeP[50-t]);

    }

    else if(Orjentacija == "Strogo lijevo" && Glide == "OFF"){

      Prva_Noga(q1_motora_PrednjeNogeP[50-t], q2_motora_PrednjeNogeP[50-t], q3_motora_PrednjeNogeP[50-t]);

      Cetvrta_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t], q3_motora_SrednjeNogeP[t]);

      Peta_Noga(q1_motora_PrednjeNogeP[t], q2_motora_PrednjeNogeP[t], q3_motora_PrednjeNogeP[t]);

    }

    else if(Orjentacija == "Strogo desno" && Glide == "OFF"){

      Prva_Noga(q1_motora_PrednjeNogeP[t], q2_motora_PrednjeNogeP[t], q3_motora_PrednjeNogeP[t]);

      Cetvrta_Noga(q1_motora_SrednjeNogeP[50-t], q2_motora_SrednjeNogeP[50-t], q3_motora_SrednjeNogeP[50-t]);

      Peta_Noga(q1_motora_PrednjeNogeP[50-t], q2_motora_PrednjeNogeP[50-t], q3_motora_PrednjeNogeP[50-t]);

    }

    else if((Orjentacija == "Nazad lijevo" || Orjentacija == "Lagano desno") && Glide == "ON"){

      Prva_Noga(q1_motora_PrednjeNogeP[t], q2_motora_PrednjeNogeP[t], q3_motora_PrednjeNogeP[t]);

      Cetvrta_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t], q3_motora_SrednjeNogeP[t]);

      Peta_Noga(q1_motora_StraznjeNogeP[50-t], q2_motora_StraznjeNogeP[50-t], q3_motora_StraznjeNogeP[50-t]);

    }

    else if((Orjentacija == "Lagano lijevo" || Orjentacija == "Nazad desno") && Glide == "ON"){
```

```
      Prva_Noga(q1_motora_StraznjeNogeP[t], q2_motora_StraznjeNogeP[t], q3_motora_StraznjeNogeP[t]);

      Cetvrta_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t], q3_motora_SrednjeNogeP[t]);

      Peta_Noga(q1_motora_PrednjeNogeP[50-t], q2_motora_PrednjeNogeP[50-t], q3_motora_PrednjeNogeP[50-t]);

    }

    else if((Orjentacija == "Strogo lijevo" || Orjentacija == "Strogo desno") && Glide == "ON"){

      Prva_Noga(q1_motora_StraznjeNogeP[t], q2_motora_StraznjeNogeP[t], q3_motora_StraznjeNogeP[t]);

      Cetvrta_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t], q3_motora_SrednjeNogeP[t]);

      Peta_Noga(q1_motora_PrednjeNogeP[t], q2_motora_PrednjeNogeP[t], q3_motora_PrednjeNogeP[t]);

    }

    else {

      Prva_Noga(q1_motora_PrednjeNogeP[t], q2_motora_PrednjeNogeP[t], q3_motora_PrednjeNogeP[t]);

      Cetvrta_Noga(q1_motora_SrednjeNogeP[t], q2_motora_SrednjeNogeP[t], q3_motora_SrednjeNogeP[t]);

      Peta_Noga(q1_motora_StraznjeNogeP[t], q2_motora_StraznjeNogeP[t], q3_motora_StraznjeNogeP[t]);

    }

    delay(DELAY);

   }

  }

}

//-----------------------------------------------------MATEMATIKA-----------------------------------------------------//
```

```
void Racunanje(double x1, double x2, double x1s, double x2s, double y1, double y2, double y1s, double y2s, double z1, double z2) {
  if (Orjentacija == "Strogo lijevo" || Orjentacija == "Strogo desno") {
    x1 = 69;
    x2 = 42;
    x1s = 80;
    x2s = 80;
    y1 = 40;
    y2 = 80;
    y1s = 40;
    y2s = -40;
  }
  else if (Orjentacija == "Nazad") {
    x2 = 70;
    x1 = 70;
    x2s = 70;
    x1s = 70;

    y2 = 70;
    y1 = 0;
    y2s = 29;
    y1s = -29;
  }
  else if (Orjentacija == "Lagano lijevo" || Orjentacija == "Lagano desno" || Orjentacija == "Nazad desno"|| Orjentacija == "Nazad lijevo") {
    x1 = 67;
    x2 = 5;
    x1s = 80;
```

```
    x2s = 80;
    y1 = 10;
    y2 = 81;
    y1s = 30;
    y2s = -30;
  }
  else {
  }
  for (int i = 0; i <= 50; i++) {
    int znakA;
    int znakB;
    if (Orjentacija == "Naprijed" || Orjentacija == "Nazad" ) {
      znakA = i;
      znakB = 50 - i;
    }
    else if (Orjentacija == "Strogo desno" || Orjentacija == "Strogo lijevo"|| Orjentacija == "Lagano desno" || Orjentacija == "Lagano lijevo"|| Orjentacija == "Nazad desno"|| Orjentacija == "Nazad lijevo") {
      znakA = 50 - i;
      znakB = i;
    }
    elipsa(x1, x2, y1, y2, z1, z2, 1, i);
      PozXEDp[znakA] = PozX;
      PozYEDp[znakA] = PozY;
      PozZEDp[znakA] = PozZ;
    if (Orjentacija == "Lagano lijevo" || Orjentacija == "Lagano desno"|| Orjentacija == "Nazad desno"|| Orjentacija == "Nazad lijevo") {
      elipsa(67, 40, 10, 46, z1, z2, 1, i);
```

```
      PozXEKp[znakB] = PozX;
      PozYEKp[znakB] = PozY;
      PozZEKp[znakB] = PozZ;
    }
    else {
      PozXEKp[znakB] = PozX;
      PozYEKp[znakB] = PozY;
      PozZEKp[znakB] = PozZ;
    }
    elipsa(x1s, x2s, y1s, y2s, z1, z2, 1, i);
      PozXEs[i] = PozX;
      PozYEs[i] = PozY;
      PozZEs[i] = PozZ;
    Izracun(PozXEDp[znakA], PozYEDp[znakA], PozZEDp[znakA]);
      q1_motora_PrednjeNogeE[znakA] = q1_motora;
      q2_motora_PrednjeNogeE[znakA] = q2_motora;
      q3_motora_PrednjeNogeE[znakA] = q3_motora;
    Izracun(PozXEs[i], PozYEs[i], PozZEs[i]);
      q1_motora_SrednjeNogeE[i] = q1_motora;
      q2_motora_SrednjeNogeE[i] = q2_motora;
      q3_motora_SrednjeNogeE[i] = q3_motora;
    Izracun(PozXEKp[znakB], PozYEKp[znakB], PozZEKp[znakB]);
      q1_motora_StraznjeNogeE[znakB] = q1_motora;
      q2_motora_StraznjeNogeE[znakB] = q2_motora;
      q3_motora_StraznjeNogeE[znakB] = q3_motora;
  }
  if (Orjentacija == "Naprijed" || Orjentacija == "Nazad") {
```

```
    for (int i = 0; i <= 50; i++) {
      Pravac(x1, x2, y1, y2, z1, z2, 1, i);
        PozXPDp[i] = PozX;
        PozYPDp[i] = PozY;
        PozZPDp[i] = PozZ;
        PozXPKp[50 - i] = PozX;
        PozYPKp[50 - i] = PozY;
        PozZPKp[50 - i] = PozZ;
      Pravac(x1s, x2s, y1s, y2s, z1, z2, 1, i);
        PozXPs[i] = PozX;
        PozYPs[i] = PozY;
        PozZPs[i] = PozZ;
      Izracun(PozXPDp[i], PozYPDp[i], PozZPDp[i]);
        q1_motora_PrednjeNogeP[i] = q1_motora;
        q2_motora_PrednjeNogeP[i] = q2_motora;
        q3_motora_PrednjeNogeP[i] = q3_motora;
      Izracun(PozXPs[i], PozYPs[i], PozZPs[i]);
        q1_motora_SrednjeNogeP[i] = q1_motora;
        q2_motora_SrednjeNogeP[i] = q2_motora;
        q3_motora_SrednjeNogeP[i] = q3_motora;
      Izracun(PozXPKp[50 - i], PozYPKp[50 - i], PozZPKp[50 - i]);
        q1_motora_StraznjeNogeP[50 - i] = q1_motora;
        q2_motora_StraznjeNogeP[50 - i] = q2_motora;
        q3_motora_StraznjeNogeP[50 - i] = q3_motora;
    }
  }
```

```
  else{
//----------------------------------------Putanja po kruznici-----------------------------------------------
    float r = 230;
    float GornjaGranica = 20;
    float DonjaGranica = 22;
    float XOdstupanje = -130;
    float YOdstupanje = -85.5;
    float rs = 212.5;
    float GornjaGranicaS = 11;
    float DonjaGranicaS = -8;
    float XOdstupanjeS = -130;
    if (Orjentacija == "Strogo desno" || Orjentacija == "Strogo lijevo") {
      r = 330;
      GornjaGranica = 5.5;
      DonjaGranica = 30;
      XOdstupanje = -216.5;
      YOdstupanje = -125;
      rs = 300;
      GornjaGranicaS = 10;
      DonjaGranicaS = -7.8;
      XOdstupanjeS = -216.5;
    }
    else if (Orjentacija == "Lagano desno" || Orjentacija == "Lagano lijevo"|| Orjentacija == "Nazad desno"|| Orjentacija == "Nazad lijevo") {
      r = 230;
      GornjaGranica = 18;
      DonjaGranica = 25;
```

```
    XOdstupanje = -130;
    YOdstupanje = -85.5;
    rs = 212.5;
    GornjaGranicaS = 11;
    DonjaGranicaS = -8;
    XOdstupanjeS = -130;
  }
  else{

  }
  for (int i = 0; i <= 50; i++) {
    Kruznica(r, z1, z2, GornjaGranica, DonjaGranica, XOdstupanje, YOdstupanje, i);
      PozXPDp[i] = PozX;
      PozYPDp[i] = PozY;
      PozZPDp[i] = PozZ;
    if (Orjentacija == "Lagano lijevo" || Orjentacija == "Lagano desno"|| Orjentacija == "Nazad desno"|| Orjentacija == "Nazad lijevo") {
      Kruznica(215, z1, z2, 8.5, 26, XOdstupanjeS, -85.5, i);
        PozXPKp[50 - i] = PozX;
        PozYPKp[50 - i] = PozY;
        PozZPKp[50 - i] = PozZ;
    }
    else {
      PozXPKp[50 - i] = PozX;
      PozYPKp[50 - i] = PozY;
      PozZPKp[50 - i] = PozZ;
    }
```

```
    Kruznica(rs, z1, z2, GornjaGranicaS, DonjaGranicaS, XOdstupanjeS, 0, i);
      PozXPs[i] = PozX;
      PozYPs[i] = PozY;
      PozZPs[i] = PozZ;
    Izracun(PozXPDp[i], PozYPDp[i], PozZPDp[i]);
      q1_motora_PrednjeNogeP[i] = q1_motora;
      q2_motora_PrednjeNogeP[i] = q2_motora;
      q3_motora_PrednjeNogeP[i] = q3_motora;
    Izracun(PozXPs[i], PozYPs[i], PozZPs[i]);
      q1_motora_SrednjeNogeP[i] = q1_motora;
      q2_motora_SrednjeNogeP[i] = q2_motora;
      q3_motora_SrednjeNogeP[i] = q3_motora;
    Izracun(PozXPKp[50 - i], PozYPKp[50 - i], PozZPKp[50 - i]);
      q1_motora_StraznjeNogeP[50 - i] = q1_motora;
      q2_motora_StraznjeNogeP[50 - i] = q2_motora;
      q3_motora_StraznjeNogeP[50 - i] = q3_motora;
    }
  }
}
//-----------------------------------------------GLIDE-----------------------------------------------------
-------------------------------------------------//

void RacunanjeGlide(double x1, double x2, double x1s, double x2s, double x1z, double x2z,
double y1, double y2, double y1s, double y2s, double y1z, double y2z, double z1, double z2) {
  if (Orjentacija == "Naprijed" || Orjentacija == "Nazad") {
    Racunanje(X1, X2, X1S, X2S, Y1, Y2, Y1S, Y2S, Z1, Z2);
  }
  if (Orjentacija == "Strogo desno" ) {
    x1 = 50;
```

```
    x2 = 90;
    x1s = 90;
    x2s = 50;
    x1z = 90;
    x2z = 50;
    y1 = 50;
    y2 = 50;
    y1s = 0;
    y2s = 0;
    y1z = 50;
    y2z = 50;
  }
    if (Orjentacija == "Strogo lijevo" ) {
    x1 = 90;
    x2 = 50;
    x1s = 50;
    x2s = 90;
    x1z = 50;
    x2z = 90;
    y1 = 50;
    y2 = 50;
    y1s = 0;
    y2s = 0;
    y1z = 50;
    y2z = 50;
  }
  else if (Orjentacija == "Nazad lijevo" || Orjentacija == "Nazad desno") {
```

```
    x1 = 70;
    x2 = 40;
    x1s = 70;
    x2s = 40;
    x1z = 90;
    x2z = 40;
    y1 = 40;
    y2 = 70;
    y1s = -15;
    y2s = 15;
    y1z = 90;
    y2z = 40;
  }
  else if (Orjentacija == "Lagano desno" || Orjentacija == "Lagano lijevo") {
    x1 = 40;
    x2 = 70;
    x1s = 40;
    x2s = 70;
    x1z = 40;
    x2z = 90;
    y1 = 70;
    y2 = 40;
    y1s = 15;
    y2s = -15;
    y1z = 40;
    y2z = 90;
  }
```

```
  else {


  }
if (Orjentacija == "Naprijed" || Orjentacija == "Nazad") {
}
else{
  for (int i = 0; i <= 50; i++) {
    int znakA = i;
    int znakB = 50-i;
    elipsa(x1, x2, y1, y2, z1, z2, 1, i);
      PozXEDp[znakA] = PozX;
      PozYEDp[znakA] = PozY;
      PozZEDp[znakA] = PozZ;
    elipsa(x1z, x2z, y1z, y2z, z1, z2, 1, i);
      PozXEKp[znakB] = PozX;
      PozYEKp[znakB] = PozY;
      PozZEKp[znakB] = PozZ;
    elipsa(x1s, x2s, y1s, y2s, z1, z2, 1, i);
      PozXEs[i] = PozX;
      PozYEs[i] = PozY;
      PozZEs[i] = PozZ;
    Izracun(PozXEDp[znakA], PozYEDp[znakA], PozZEDp[znakA]);
      q1_motora_PrednjeNogeE[znakA] = q1_motora;
      q2_motora_PrednjeNogeE[znakA] = q2_motora;
      q3_motora_PrednjeNogeE[znakA] = q3_motora;
    Izracun(PozXEs[i], PozYEs[i], PozZEs[i]);
      q1_motora_SrednjeNogeE[i] = q1_motora;
```

```
      q2_motora_SrednjeNogeE[i] = q2_motora;
      q3_motora_SrednjeNogeE[i] = q3_motora;
    Izracun(PozXEKp[znakB], PozYEKp[znakB], PozZEKp[znakB]);
      q1_motora_StraznjeNogeE[znakB] = q1_motora;
      q2_motora_StraznjeNogeE[znakB] = q2_motora;
      q3_motora_StraznjeNogeE[znakB] = q3_motora;
  }
    for (int i = 0; i <= 50; i++) {
      Pravac(x1, x2, y1, y2, z1, z2, 1, i);
        PozXPDp[i] = PozX;
        PozYPDp[i] = PozY;
        PozZPDp[i] = PozZ;
      Pravac(x1z, x2z, y1z, y2z, z1, z2, 1, i);
        PozXPKp[50 - i] = PozX;
        PozYPKp[50 - i] = PozY;
        PozZPKp[50 - i] = PozZ;
      Pravac(x1s, x2s, y1s, y2s, z1, z2, 1, i);
        PozXPs[i] = PozX;
        PozYPs[i] = PozY;
        PozZPs[i] = PozZ;
      Izracun(PozXPDp[i], PozYPDp[i], PozZPDp[i]);
        q1_motora_PrednjeNogeP[i] = q1_motora;
        q2_motora_PrednjeNogeP[i] = q2_motora;
        q3_motora_PrednjeNogeP[i] = q3_motora;
      Izracun(PozXPs[i], PozYPs[i], PozZPs[i]);
        q1_motora_SrednjeNogeP[i] = q1_motora;
        q2_motora_SrednjeNogeP[i] = q2_motora;
```

```
      q3_motora_SrednjeNogeP[i] = q3_motora;
    Izracun(PozXPKp[50 - i], PozYPKp[50 - i], PozZPKp[50 - i]);
      q1_motora_StraznjeNogeP[50 - i] = q1_motora;
      q2_motora_StraznjeNogeP[50 - i] = q2_motora;
      q3_motora_StraznjeNogeP[50 - i] = q3_motora;
   }
 }
}
```

ESP32 CAM kod:

```
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_http_server.h"
// Replace with your network credentials
const char* ssid = "Patcho";
const char* password = "12345678";
#define PART_BOUNDARY "123456789000000000000987654321"
// pločica mora biti AI thinker
#define CAMERA_MODEL_AI_THINKER
//#define CAMERA_MODEL_WROVER_KIT
#if defined(CAMERA_MODEL_WROVER_KIT)
```

```
  #define PWDN_GPIO_NUM    -1
  #define RESET_GPIO_NUM   -1
  #define XCLK_GPIO_NUM    21
  #define SIOD_GPIO_NUM    26
  #define SIOC_GPIO_NUM    27
  #define Y9_GPIO_NUM      35
  #define Y8_GPIO_NUM      34
  #define Y7_GPIO_NUM      39
  #define Y6_GPIO_NUM      36
  #define Y5_GPIO_NUM      19
  #define Y4_GPIO_NUM      18
  #define Y3_GPIO_NUM       5
  #define Y2_GPIO_NUM       4
  #define VSYNC_GPIO_NUM   25
  #define HREF_GPIO_NUM    23
  #define PCLK_GPIO_NUM    22
#elif defined(CAMERA_MODEL_AI_THINKER)
  #define PWDN_GPIO_NUM     32
  #define RESET_GPIO_NUM    -1
  #define XCLK_GPIO_NUM      0
  #define SIOD_GPIO_NUM     26
  #define SIOC_GPIO_NUM     27
  #define Y9_GPIO_NUM       35
  #define Y8_GPIO_NUM       34
  #define Y7_GPIO_NUM       39
  #define Y6_GPIO_NUM       36
  #define Y5_GPIO_NUM       21
```

```
  #define Y4_GPIO_NUM       19
  #define Y3_GPIO_NUM       18
  #define Y2_GPIO_NUM        5
  #define VSYNC_GPIO_NUM    25
  #define HREF_GPIO_NUM     23
  #define PCLK_GPIO_NUM     22
#else
  #error "Camera model not selected"
#endif
static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary=" PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";
httpd_handle_t camera_httpd = NULL;
httpd_handle_t stream_httpd = NULL;
static const char PROGMEM INDEX_HTML[] = R"rawliteral(
<!DOCTYPE html>
<html>
<style>
   html {
    font-family: Arial, Helvetica, sans-serif;
    display: inline-block;
    text-align: center;
  }
  #button1 {
    background-color: #2f4468;
    border: none;
```

```
    color: white;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 18px;
    margin: 6px 3px;
    cursor: pointer;
    -webkit-touch-callout: none;
    -webkit-user-select: none;
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
    -webkit-tap-highlight-color: rgba(0,0,0,0);
    align-self:centar;
    grid-column: 2 / 3;
    grid-row: 2 / 3;
  }
  .button2 {
    background-color: #0400d3;
    border: none;
    color: white;
    height:auto;
    width:auto;
    padding: 10px 20px;
    text-align: center;
```

```
    text-decoration: none;
    display: inline-block;
    font-size: 18px;
    margin: 6px 3px;
    cursor: pointer;
    -webkit-touch-callout: none;
    -webkit-user-select: none;
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
    -webkit-tap-highlight-color: rgba(0,0,0,0);
    grid-column: 2 / 3;
    grid-row: 5 / 6;

  }
  #buttonUp {
    background-color: #e90101;
    height:auto;
    width:auto;
    border: none;
    color: white;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 18px;
```

```
  margin: 6px 3px;
  cursor: pointer;
  -webkit-touch-callout: none;
  -webkit-user-select: none;
  -khtml-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
  -webkit-tap-highlight-color: rgba(0,0,0,0);
  grid-column: 3 / 4;
  grid-row: 2 / 3;
 }
 #buttonDown {
  background-color: #e90101;
  border: none;
  color: white;
  height:auto;
  width:auto;
  padding: 10px 20px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 18px;
  margin: 6px 3px;
  cursor: pointer;
  -webkit-touch-callout: none;
  -webkit-user-select: none;
```

```
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
    -webkit-tap-highlight-color: rgba(0,0,0,0);
    grid-column: 3 / 4;
    grid-row: 4 / 5;
  }
  #buttonRight {
    background-color: #e90101;
    border: none;
    color: white;
    height:auto;
    width:100px;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 18px;
    margin: 6px 3px;
    cursor: pointer;
    -webkit-touch-callout: none;
    -webkit-user-select: none;
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
```

```
    -webkit-tap-highlight-color: rgba(0,0,0,0);
    grid-column: 4 / 5;
    grid-row: 3 / 4;
  }
  #buttonLeft {
    background-color: #e90101;
    border: none;
    color: white;
    height:auto;
    width:100px;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 18px;
    margin: 6px 3px;
    cursor: pointer;
    -webkit-touch-callout: none;
    -webkit-user-select: none;
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
    -webkit-tap-highlight-color: rgba(0,0,0,0);
    grid-column: 2 / 3;
    grid-row: 3 / 4;
  }
```

```
#GlideButton {
  background-color: #e90101;
  border: none;
  color: white;
  height:auto;
  width:100px;
  padding: 10px 20px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 18px;
  margin: 6px 3px;
  cursor: pointer;
  -webkit-touch-callout: none;
  -webkit-user-select: none;
  -khtml-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
  -webkit-tap-highlight-color: rgba(0,0,0,0);
  grid-column: 3 / 4;
  grid-row: 5 / 6;
}
#PocetniPolozaj {
  background-color: #ffa600;
  height:auto;
  width:auto;
```

```
    border: none;
    color: white;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 18px;
    margin: 6px 3px;
    cursor: pointer;
    -webkit-touch-callout: none;
    -webkit-user-select: none;
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
    -webkit-tap-highlight-color: rgba(0,0,0,0);
    grid-column: 4 / 5;
    grid-row: 5 / 6;
  }
  img {  width: auto ;
    max-width: 100% ;
    height: auto ;
  }
  h1 {
    font-size: 1.8rem;
    color: white;
  }
```

```
p {
  font-size: 1.4rem;
}
.topnav {
  overflow: hidden;
  background-color: #0A1128;
}
body {
  margin: 0;
}
.content {
  padding: 30px;
}
.card-grid {
  max-width: 1500px;
  min-width: 250px;
  margin: 20px auto;
  display: grid;
  grid-gap: 2rem;
  position: relative;
  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
}
.card {
  background-color: white;
  box-shadow: 2px 2px 12px 1px rgba(140,140,140,.5);
  min-height: 250px;
  height: auto-fit, minmax(200px, 1fr);
```

```
    position:relative;
  }
  .card-title1 {
    font-size: 1.2rem;
    font-weight: bold;
    color: #034078;
    box-sizing: content-box;
    align-self:centar;
  }
  .card-title2 {
    font-size: 1.2rem;
    height: auto;
    font-weight: bold;
    color: #034078;
    box-sizing: content-box;
    grid-column: 3 / 4;
    grid-row: 1 / 2;
  }
  .card-title3 {
    font-size: 1.2rem;
    font-weight: bold;
    color: #034078;
    box-sizing: content-box;
    align-self:centar;
    grid-column: 2 / 3;
    grid-row: 1 / 2;
  }
```

```
.state {
  font-size: 1.2rem;
  color:#1282A2;
}
.switch {
  padding-left: 5%;
  padding-right: 5%;
}
.raspored{
  position: absolute;
  height: 100%;
  width: 100%;
  box-sizing: border-box;
  border: 5px;
  padding: 10px;
  display: flex;
  flex-direction: column;
  justify-content: space-around;
  display: grid;
  grid-template-columns: 1fr 1fr 1fr ;
  grid-template-rows:1fr 1fr 1fr 1fr 1fr;
}
.box1{
  box-sizing: content-box;
  font-size: 20px;
  padding: 5px;
  width: auto;
```

```
    height: auto;
    min-width: 80px;
    min-height: 25px;
    max-width: 120px;
    max-height: 25px;
    border: solid 2px black;
    background-color: #ffffff;
  }
  .box2{
    box-sizing: content-box;
    font-size: 20px;
    padding: 5px;
    width: auto;
    height: auto;
    min-width: 80px;
    min-height: 25px;
    max-width: 120px;
    max-height: 25px;
    border: solid 2px rgb(199, 19, 19);
    background-color: #ffffff;
  }
  .box3{
    box-sizing: content-box;
    font-size: 20px;
    padding: 5px;
    width: auto;
    height: auto;
```

```
    min-width: 80px;
    min-height: 25px;
    max-width: 120px;
    max-height: 25px;
    border: solid 2px rgb(0, 0, 0);
    background-color: #ffffff;
  }
  .state{
    color: #000000;
    font-size: 20px;
    position: absolute;
    grid-column: 1 / 2;
    grid-row: 3 / 4;
  }
  .state2{
    color: #000000;
    font-size: 20px;
    position: absolute;
    grid-column: 2 / 3;
    grid-row: 3 / 4;
  }
  .state3{
    color: #000000;
    font-size: 20px;
    position: absolute;
    grid-column: 3 / 4;
    grid-row: 3 / 4;
```

```
  }
.grid{
  position: absolute;
  height: 100%;
  width: 100%;
  box-sizing: border-box;
  display: grid;
  grid-template-columns: 1fr 2fr 3fr 2fr 1fr;
  grid-template-rows:40px 50px 50px 50px 50px ;
}
.grid2{
  position: absolute;
  padding: 10px;
  height: 100%;
  width: 100%;
  box-sizing: border-box;
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows:1fr 1fr 1fr ;
}
.box4{
  box-sizing: content-box;
  width: auto;
  padding-top: 14px;
  height: auto;
  font-size: 1.2rem;
  background-color: #ffffff;
```

```
  grid-column: 3 / 4;
  grid-row: 3 / 4;
  text-align: inherit;
}
#delay{
  height:20px;
  width:50px;
  margin: auto;
  padding: 10px 10px;
  text-align: middle;
  font-size: 16px;
  grid-column: 1 / 2;
  grid-row: 3 / 4;
}
#visina{
  height:20px;
  width:50px;
  margin: auto;
  padding: 10px 10px;
  text-align: middle;
  font-size: 16px;
  grid-column: 3 / 4;
  grid-row: 3 / 4;
}
#PocetniPolozaj2 {
  background-color: #0179e9;
  height:auto;
```

```
  width:auto;
  border: none;
  color: white;
  padding: 10px 20px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 18px;
  margin: 6px 3px;
  cursor: pointer;
  -webkit-touch-callout: none;
  -webkit-user-select: none;
  -khtml-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
  -webkit-tap-highlight-color: rgba(0,0,0,0);
  grid-column: 3 / 4;
  grid-row: 3 / 4;
}
.Veza{
  font-size: 1.2rem;
  font-weight: bold;
  color: #7e000098;
  box-sizing: content-box;
  align-self:centar;
  grid-column: 2 / 3;
```

```
  grid-row: 1 / 2;
}
.grid3{
  position: absolute;
  padding: 10px;
  height: 100%;
  width: 100%;
  box-sizing: border-box;
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows:50px 50px 50px 50px 50px ;
}
.card-title4 {
  font-size: 1.2rem;
  font-weight: bold;
  color: #034078;
  box-sizing: content-box;
  align-self:centar;
  grid-column: 2 / 3;
  grid-row: 1 / 2;
}
.custom-select{
  width: auto;
  align-self:center;
  grid-column: 1 / 2;
  grid-row: 2 / 3;
}
```

```
.slider {
  -webkit-appearance: none;
  margin: 0 auto;
  width: 100%;
  height: 15px;
  border-radius: 10px;
  background: #FFD65C;
  outline: none;
}
.slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 30px;
  height: 30px;
  border-radius: 50%;
  background: #034078;
  cursor: pointer;
}
.slider::-moz-range-thumb {
  width: 30px;
  height: 30px;
  border-radius: 50% ;
  background: #034078;
  cursor: pointer;
}
.sliderCon1{
  grid-column: 1 / 4;
```

```
  grid-row: 3 / 4;
  padding-top: 10px;
}
.sliderCon2{
  grid-column: 1 / 4;
  grid-row: 4 / 5;
  padding-top: 10px;
}
.sliderCon3{
  grid-column: 1 / 4;
  grid-row: 5 / 6;
  padding-top: 10px;
}
.state10{
  font-size: 1.2rem;
  color: #0400d3;
  position: relative;
  bottom:40%;
  right:40%;
}
.Primjeni {
  background-color: #0400d3;
  border: none;
  color: white;
  height:auto;
  width:auto;
  padding: 10px 20px;
```

```
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 18px;
  margin: 6px 3px;
  cursor: pointer;
  -webkit-touch-callout: none;
  -webkit-user-select: none;
  -khtml-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
  -webkit-tap-highlight-color: rgba(0,0,0,0);
  grid-column: 3 / 4;
  grid-row: 2 / 3;
}
.Naslov{
  border: none;
  padding: 10px 20px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 18px;
  margin: 6px 3px;
  grid-column: 1 / 2;
  grid-row: 2 / 3;
}
```

```
.Naslov2{
  border: none;
  padding: 10px 20px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 18px;
  margin: 6px 3px;
  grid-column: 3 / 4;
  grid-row: 2 / 3;
}
.GlavniNaslov{
  border: none;
  padding: 10px 20px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 18px;
  font:bold;
  margin: 6px 3px;
  grid-column: 2 / 3;
  grid-row: 1 / 2;
}
    </style>
<head>
    <title>Spider</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
   <link rel="icon" type="image/png" href="favicon.png">
</head>
<body>
      <div class="topnav">
      <h1>Pauk</h1>
   </div>
   <div class="content">
      <div class="card-grid">
         <div class="card">
            <p class="card-title">KAMERA</p>
            <p class="switch">
               <img src="" id="photo" >
            </p>
         </div>
         <div class="card">
               <div class="grid">
                  <p class="card-title2"> Joystick</p>
                  <p class="Veza"><span id="Spoj">Not connected</span></p>
                  <tr><td colspan="3" ><button class="button2" onmousedown="Stop();" ontouchstart="Stop();">STOP</button></td></tr>
                  <tr><td colspan="3" ><button id="buttonUp" onclick="moveup()" ontouchstart="moveup()">UP</button></td></tr>
                  <tr><td colspan="3" ><button id="buttonDown" onclick="movedown()" ontouchstart="movedown()">DOWN</button></td></tr>
                  <tr><td colspan="3" ><button id="buttonLeft" onclick="moveleft()" ontouchstart="moveleft()">LEFT</button></td></tr>
                  <tr><td colspan="3" ><button id="buttonRight" onclick="moveright()" ontouchstart="moveright()">RIGHT</button></td></tr>
```

```
                <tr><td colspan="3" ><button id="PocetniPolozaj"
onmousedown="Slanje('PocetniPolozaj','PocetniPolozaj')"
onmouseup="KlikStop('PocetniPolozaj')"
ontouchstart="Slanje('PocetniPolozaj','PocetniPolozaj')"
ontouchend="KlikStop('PocetniPolozaj')">Pocetni</button></td></tr>

                <tr><td colspan="3" ><button id="GlideButton" onclick="glide()"
ontouchstart="glide()">GLIDE</button></td></tr>

                <div class="box4"><span id="Orjentacija">STOP</span></div>

            </div>

        </div>

        <div class="card">

            <div class="raspored">

            <p class="card-title3">Senzor</p>

                <tr><td colspan="3" align="center"><button id="button1"
onmousedown="getValues();" >Senzor</button></td></tr>

                <div class="state">Lijevo[cm]:<br><p class="box1"><span
id="senzor1"></span></p><br></div>

                <div class="state2">Ispred[cm]:<br><p class="box2"> <span
id="senzor2"></span></p></br></div>

                <div class="state3">Desno[cm]:<br><p class="box3"> <span
id="senzor3"></span></p></br></div>


            </div>

        </div>

        <div class="card">

            <div class="grid2">

                <div class="GlavniNaslov">Postavke</div>

                <div class="Naslov">Brzina kretanja</div>

                <div class="Naslov2">Visina</div>

                <input type="number" id="delay" onchange="Pauza()" value = "7" min="1"
max="10" step="0.5">
```

```
            <input type="number" id="visina" onchange="Visina()" value = "-140" min="-160" max="-100" step="10">
        </div>
    </div>
    <div class="card">
      <div class="grid3">
        <p class="card-title4">Precizna kontrola</p>
        <div class="custom-select">
          <select id = "Odabir">
            <option value="1">PrvaNoga</option>
            <option value="2">DrugaNoga</option>
            <option value="3">TrecaNoga</option>
            <option value="4">CetvrtaNoga</option>
            <option value="5">PetaNoga</option>
            <option value="6">SestaNoga</option>
          </select>
        </div>
        <tr><td colspan="3" ><button class="Primjeni" onmousedown="Primjeni();" ontouchstart="Primjeni();">Primjeni</button></td></tr>
        <div class="sliderCon1"><input type="range" onchange="updateSliderPWM(this)" min="-90" max="90" step="1" value="0" class="slider" id="Value1">
          <p class="state10">Kut: <span id="sliderValue1">0</span>&deg;</p>
        </div>
        <div class="sliderCon2"><input type="range" onchange="updateSliderPWM(this)" min="-90" max="90" step="1" value="0" class="slider" id="Value2">
          <p class="state10">Kut: <span id="sliderValue2">0</span>&deg;</p>
        </div>
        <div class="sliderCon3"><input type="range" onchange="updateSliderPWM(this)" min="0" max="180" step="1" value="90" class="slider" id="Value3">
```

```
                <p class="state10">Kut: <span id="sliderValue3">90</span>&deg;</p>
              </div>


            </div>
        </div>
        </div>
    </div>
</body>
    <script>
var gateway = 'ws://'+'192.168.147.144'+':81/';
var websocket;
var Noga = "PrvaNoga";
let buttonUp = Boolean(false);
let buttonDown = Boolean(false);
let buttonRight = Boolean(false);
let buttonLeft = Boolean(false);
let GlideButton = Boolean(false);
var Orjentacija = "STOP"
var Glide = "OFF"
var Delay = 7;
var Vis = -140;
var Kordinate = {Delay:"0", Z1:"-140", Orjentacija:"STOP", Glide:"OFF"};
var Slider = {Orjentacija:"STOP", BrojSlidera:"0", VrijednostSlidera:"0"};
window.addEventListener('load', onload);
function onload(event) {
    initWebSocket();
}
```

```
function initWebSocket() {
    console.log('Trying to open a WebSocket connection…');
    websocket = new WebSocket(gateway);
    websocket.onopen = onOpen;
    websocket.onclose = onClose;
    websocket.onmessage = onMessage;
}
function onOpen(event) {
    console.log('Connection opened');
  document.getElementById("Spoj").innerHTML = "Connected";
  document.getElementById("Spoj").style.color = '#00df25af';
}
function onClose(event) {
    console.log('Connection closed');
  document.getElementById("Spoj").innerHTML = "Not connected";
  document.getElementById("Spoj").style.color = '#7e000098';
    setTimeout(initWebSocket, 2000);
}
function getValues(){
    websocket.send("W");
}
function Stop(){
    slanjeJson(Delay,"STOP",Glide);
  buttonUp = false;
  color(buttonUp,'buttonUp');
  buttonDown = false;
  color(buttonDown,'buttonDown');
```

```
  buttonRight = false;
  color(buttonRight,'buttonRight');
  buttonLeft = false;
  color(buttonLeft,'buttonLeft');
}
function onMessage(event) {
  console.log(event.data);
    const Senzor = JSON.parse(event.data);
    console.log(Senzor);
  console.log(Senzor.Lijevo);
    document.getElementById("senzor1").innerHTML = Senzor.Lijevo;
    document.getElementById("senzor2").innerHTML = Senzor.Desno;
    document.getElementById("senzor3").innerHTML = Senzor.Ispred;
}
window.onload = document.getElementById("photo").src = window.location.href.slice(0, -1) +
":82/stream";
function Slanje(x,id_buttona){
  console.log(x);
  console.log(id_buttona);
  document.getElementById(id_buttona).style.backgroundColor = '#00ff37';
  slanjeJson(Delay,x,Glide);
}
function KlikStop(id_buttona){
  document.getElementById(id_buttona).style.backgroundColor = '';
}
function moveup() {
  buttonUp = !buttonUp;
```

```
  pisanje();
  color(buttonUp,'buttonUp');
}
function movedown() {
  buttonDown = !buttonDown;
  pisanje();
  color(buttonDown,'buttonDown');
}
function moveleft() {
  buttonLeft = !buttonLeft;
  pisanje();
  color(buttonLeft,'buttonLeft');
}
function moveright() {
  buttonRight = !buttonRight;
  pisanje();
  color(buttonRight,'buttonRight');
}
function glide() {
  GlideButton = !GlideButton;
  if (GlideButton == false){
    Glide = "OFF";
  }
  else{
    Glide = "ON";
  }
  color(GlideButton,'GlideButton');
```

```
  console.log(Glide);
}
function Pauza(){
  Delay = document.getElementById("delay").value;
  console.log(Delay);
}
function Visina(){
  Vis = document.getElementById("visina").value;
  console.log(Vis);
}
function slanjeJson(x, Orjentacija,Glide,Vis){
  Kordinate['Delay'] = x;
  Kordinate['Z1'] = Vis;
  Kordinate['Orjentacija'] = Orjentacija;
  Kordinate['Glide'] = Glide;
  websocket.send("4a"+JSON.stringify(Kordinate));
  console.log("4a"+JSON.stringify(Kordinate));
  document.getElementById("Orjentacija").innerHTML = Orjentacija;
}
function pisanje(){
if (buttonUp == true && buttonDown==false && buttonLeft == false && buttonRight==false){
  slanjeJson(Delay,"Naprijed",Glide,Vis)
}
else if (buttonUp == true && buttonDown==false && buttonLeft == false && buttonRight==true){
  slanjeJson(Delay,"Lagano desno",Glide,Vis)
}
```

```
else if (buttonUp == true && buttonDown==false && buttonLeft == true && buttonRight==false){
  slanjeJson(Delay,"Lagano lijevo",Glide,Vis)
}
else if (buttonUp == false && buttonDown==true && buttonLeft == false && buttonRight==false){
  slanjeJson(Delay,"Nazad",Glide,Vis)
}
else if (buttonUp == false && buttonDown==true && buttonLeft == false && buttonRight==true){
  slanjeJson(Delay,"Nazad desno",Glide,Vis)
}
else if (buttonUp == false && buttonDown==true && buttonLeft == true && buttonRight==false){
  slanjeJson(Delay,"Nazad lijevo",Glide,Vis)
}
else if (buttonUp == false && buttonDown==false && buttonLeft == false && buttonRight==true){
  slanjeJson(Delay,"Strogo desno",Glide,Vis)
}
else if (buttonUp == false && buttonDown==false && buttonLeft == true && buttonRight==false){
  slanjeJson(Delay,"Strogo lijevo",Glide,Vis)
}
else if (buttonUp == false && buttonDown==false && buttonLeft == false && buttonRight==false){
  slanjeJson(Delay,"STOP",Glide,Vis)
}
}
function getRandomIntInclusive(min, max) {
```

```
  const str = (Math.random() * (max - min) + min).toFixed(2);
  return parseFloat(str)
}
function color(button, id_buttona){
  if (button == true){
    document.getElementById(id_buttona).style.backgroundColor = '#00ff37';
  }
  else{
    document.getElementById(id_buttona).style.backgroundColor = '#e90101';
  }
}
function getSelectedText(elementId){
    var elt = document.getElementById(elementId);
    if (elt.selectedIndex == -1)
        return null;
    return elt.options[elt.selectedIndex].text;
}
function Primjeni(){
  Noga = getSelectedText('Odabir');
  console.log(Noga);
  //document.getElementsByClassName("sliderCon1").min = 0;
}
function updateSliderPWM(element) {
    var sliderNumber = element.id.charAt(element.id.length-1);
    var sliderValue = document.getElementById(element.id).value;
    document.getElementById("sliderValue"+sliderNumber).innerHTML = sliderValue;
  SlanjeSliderVrijednosti(Noga, sliderNumber, sliderValue);
```

```
  console.log(Noga, sliderNumber, sliderValue);
}
function SlanjeSliderVrijednosti(Leg, BrojSlidera, VrijednostSlidera){
  Slider['Orjentacija']=Leg;
  Slider['BrojSlidera']=BrojSlidera;
  Slider['VrijednostSlidera']=VrijednostSlidera;
  websocket.send("XX"+JSON.stringify(Slider));
  console.log("XX"+JSON.stringify(Slider));
}
    </script>
</body>
</html>
)rawliteral";
static esp_err_t index_handler(httpd_req_t *req){
  httpd_resp_set_type(req, "text/html");
  return httpd_resp_send(req, (const char *)INDEX_HTML, strlen(INDEX_HTML));
}
static esp_err_t stream_handler(httpd_req_t *req){
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
  char * part_buf[64];
  res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
  if(res != ESP_OK){
    return res;
  }
```

```
while(true){
  fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Camera capture failed");
    res = ESP_FAIL;
  } else {
    if(fb->width > 400){
      if(fb->format != PIXFORMAT_JPEG){
        bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
        esp_camera_fb_return(fb);
        fb = NULL;
        if(!jpeg_converted){
          Serial.println("JPEG compression failed");
          res = ESP_FAIL;
        }
      } else {
        _jpg_buf_len = fb->len;
        _jpg_buf = fb->buf;
      }
    }
  }
  if(res == ESP_OK){
    size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
    res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
  }
  if(res == ESP_OK){
    res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
```

```
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
    }
    if(fb){
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
    } else if(_jpg_buf){
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if(res != ESP_OK){
      break;
    }
    //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}
static esp_err_t cmd_handler(httpd_req_t *req){
  char*  buf;
  size_t buf_len;
  char variable[32] = {0,};

  buf_len = httpd_req_get_url_query_len(req) + 1;
  if (buf_len > 1) {
```

```
  buf = (char*)malloc(buf_len);
  if(!buf){
    httpd_resp_send_500(req);
    return ESP_FAIL;
  }
  if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
    if (httpd_query_key_value(buf, "go", variable, sizeof(variable)) == ESP_OK) {
    } else {
      free(buf);
      httpd_resp_send_404(req);
      return ESP_FAIL;
    }
  } else {
    free(buf);
    httpd_resp_send_404(req);
    return ESP_FAIL;
  }
  free(buf);
} else {
  httpd_resp_send_404(req);
  return ESP_FAIL;
}
sensor_t * s = esp_camera_sensor_get();
int res = 0;

if(res){
  return httpd_resp_send_500(req);
```

```
  }
  httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
  return httpd_resp_send(req, NULL, 0);
}
void startCameraServer(){
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;
  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = index_handler,
    .user_ctx  = NULL
  };
  httpd_uri_t cmd_uri = {
    .uri       = "/action",
    .method    = HTTP_GET,
    .handler   = cmd_handler,
    .user_ctx  = NULL
  };
  httpd_uri_t stream_uri = {
    .uri       = "/stream",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };
  if (httpd_start(&camera_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(camera_httpd, &index_uri);
```

```
    httpd_register_uri_handler(camera_httpd, &cmd_uri);
  }
  config.server_port += 2;
  config.ctrl_port += 2;
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &stream_uri);
  }
}
void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector
  Serial.begin(115200);
  Serial.setDebugOutput(false);
  Wifi();
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
```

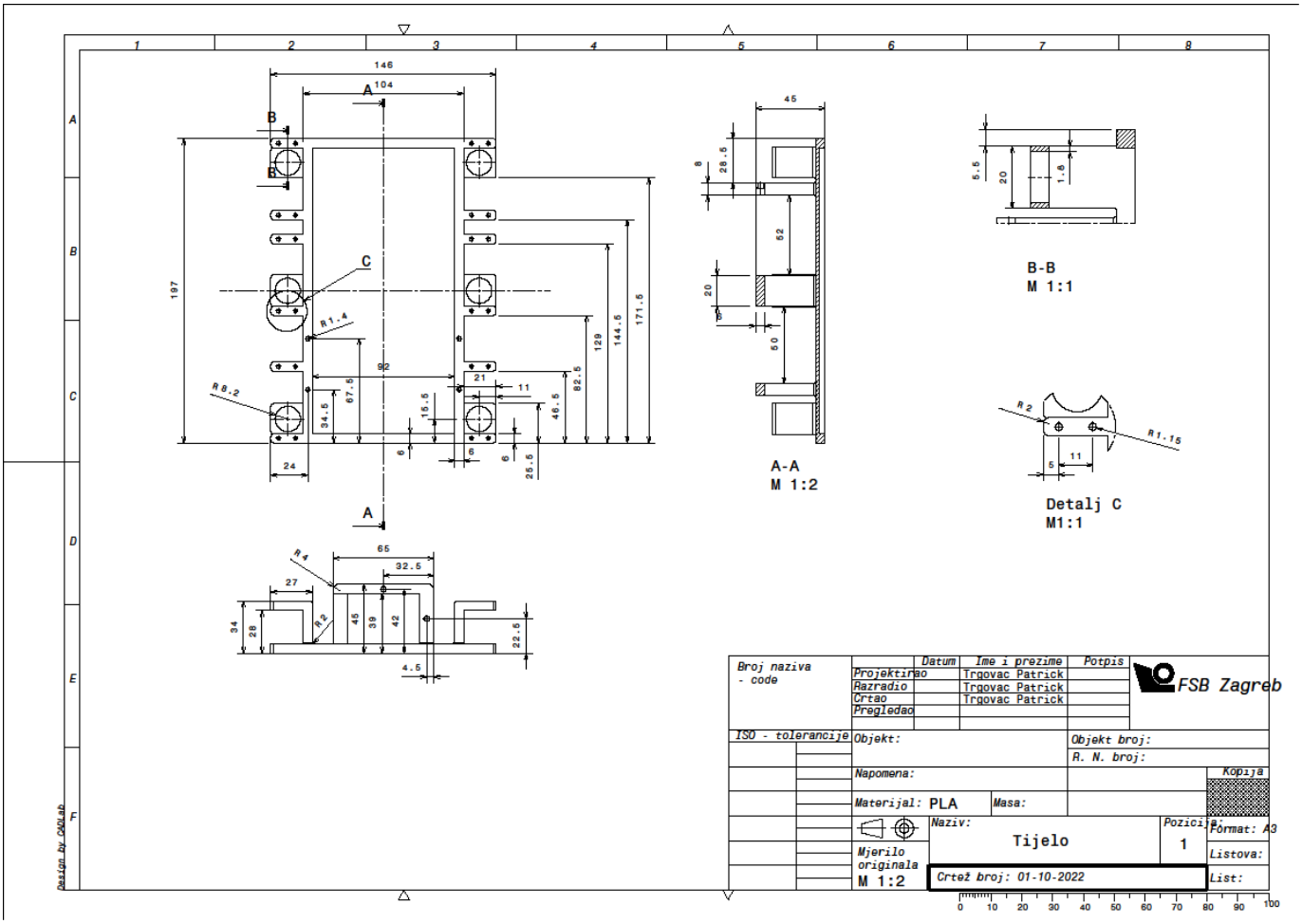
```
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  if(psramFound()){
    config.frame_size = FRAMESIZE_VGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }
  // Camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
  }
  // Start streaming web server
  startCameraServer();
}
void loop() {

}
```

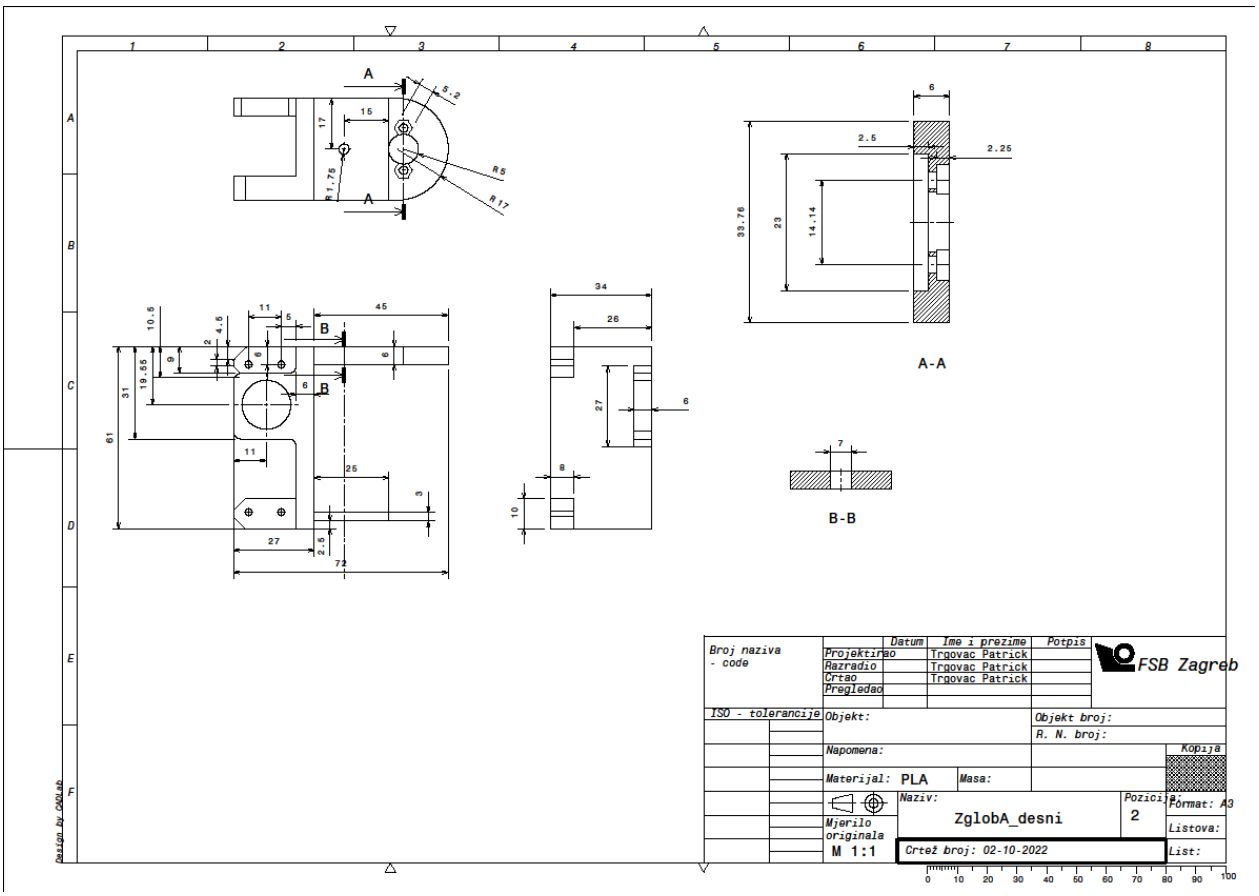
```
void Wifi(){
// Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.print("Camera Stream Ready! Go to: http://");
  Serial.println(WiFi.localIP());

}
```

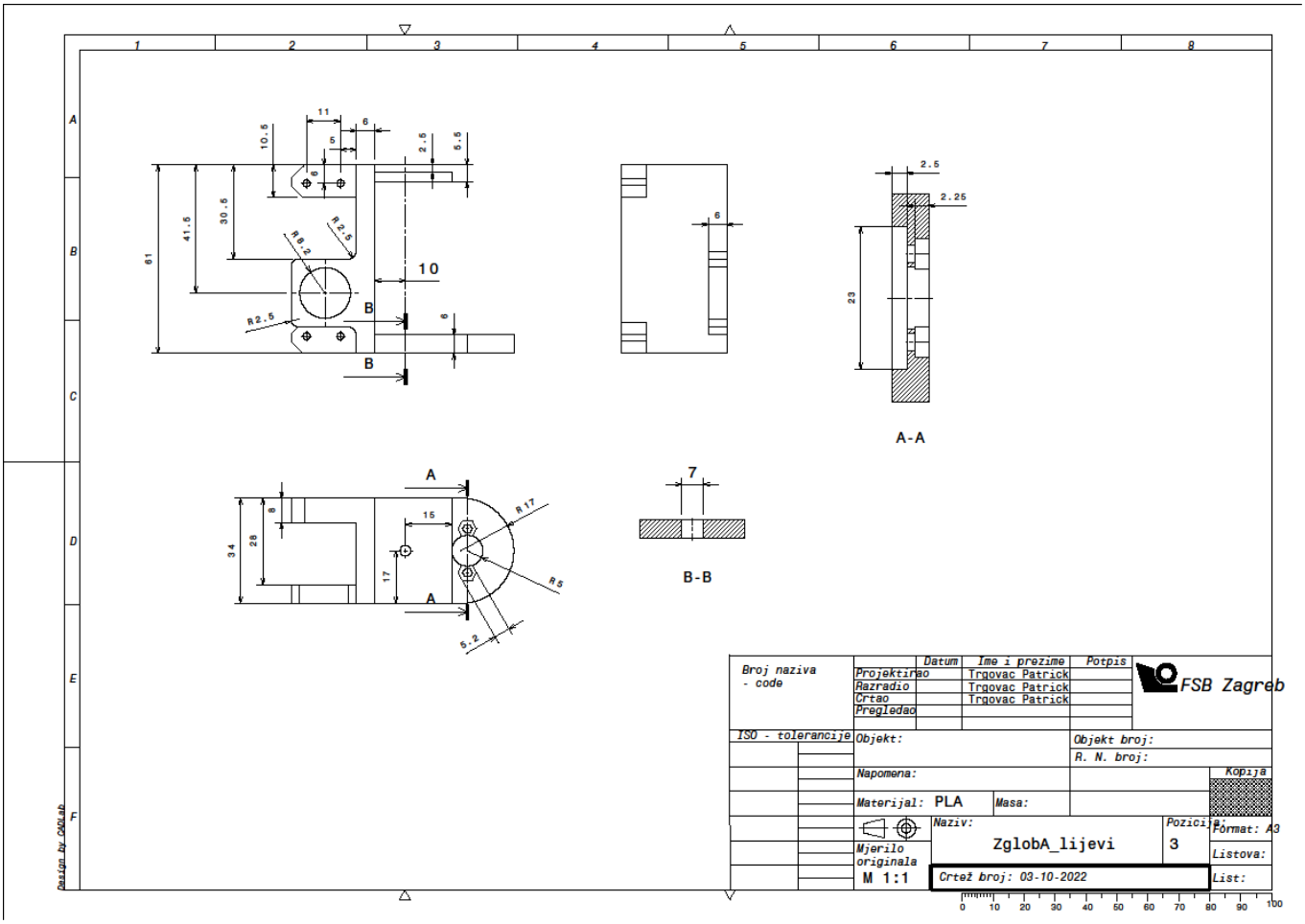
10.3. Tehnička dokumentacija:


146
104
197
92
171.5
144.6
129
82.5
67.5
46.5
34.5
24
R1.4
R8.2
A-A
M 1:2
B-B
M 1:1
Detalj C
M1:1
R2
R1.15
65
32.5
27
R4
R2
34
28
45
39
42
22.5
4.5
Broj naziva - code
Datum
Ime i prezime
Potpis
Projektirao
Razradio
Crtao
Pregledao
Trgovac Patrick
FSB Zagreb
ISO - tolerancije
Objekt:
Objekt broj:
R. N. broj:
Napomena:
Kopija
Materijal: PLA
Masa:
Naziv:
Tijelo
Pozicija:
1
Format: A3
Mjerilo originala
M 1:2
Crtež broj: 01-10-2022
Listova:
List:

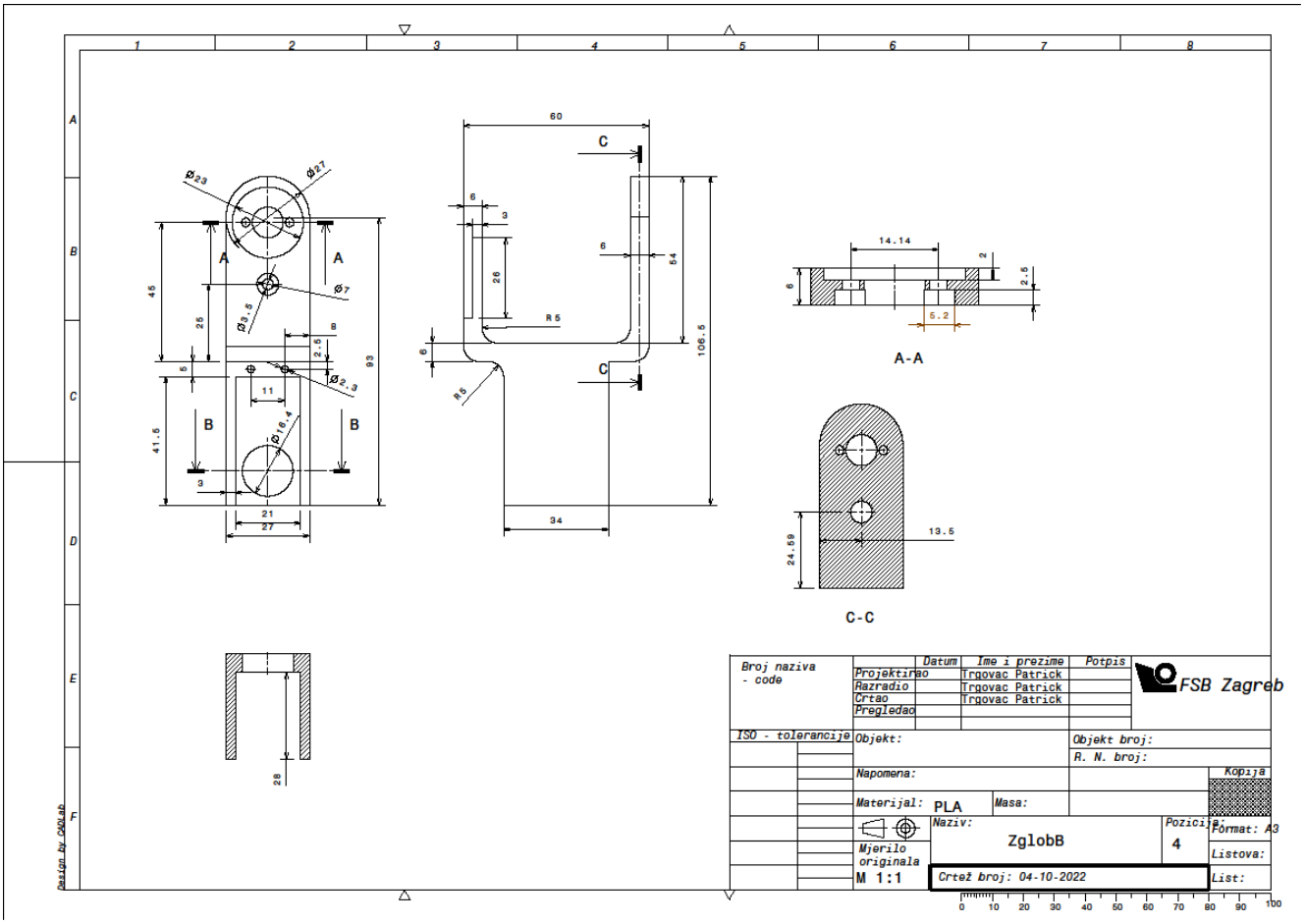
ZglobA_desni:



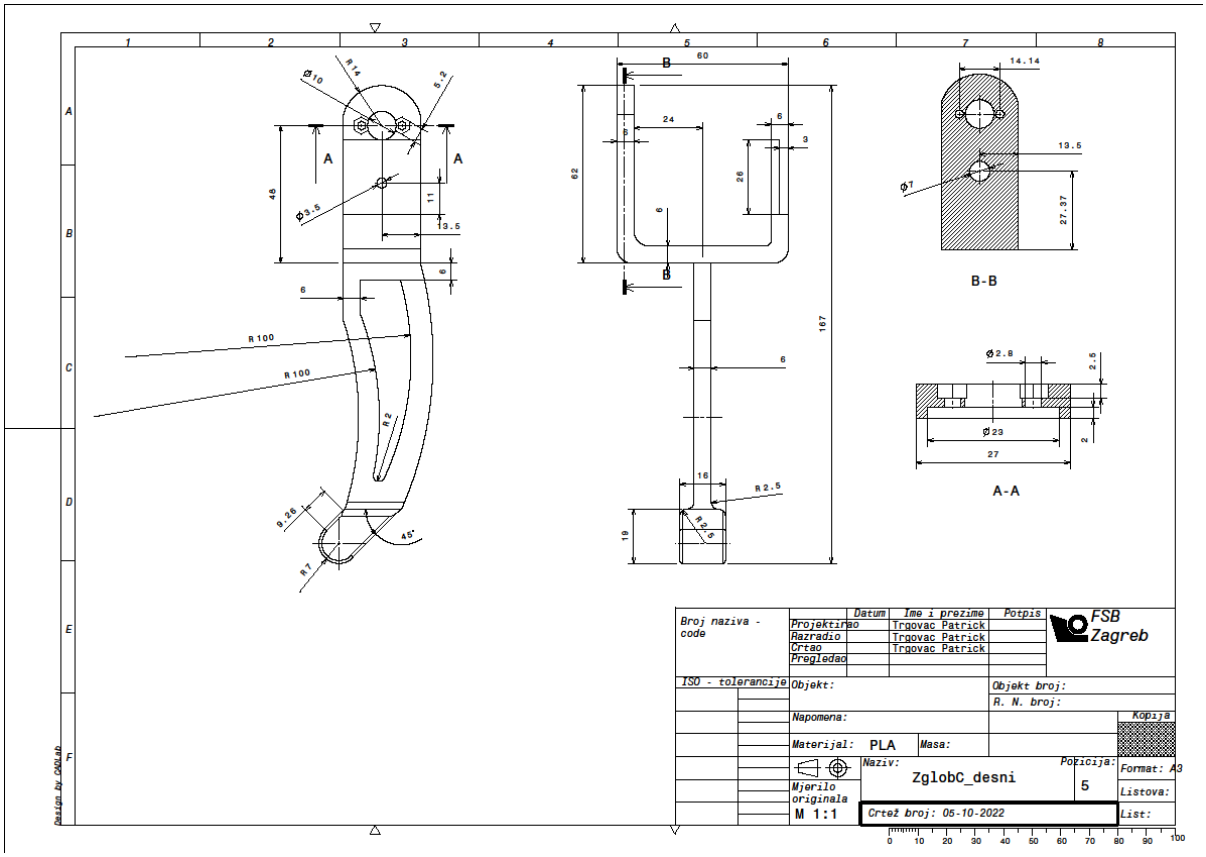
ZglobA_lijevi:



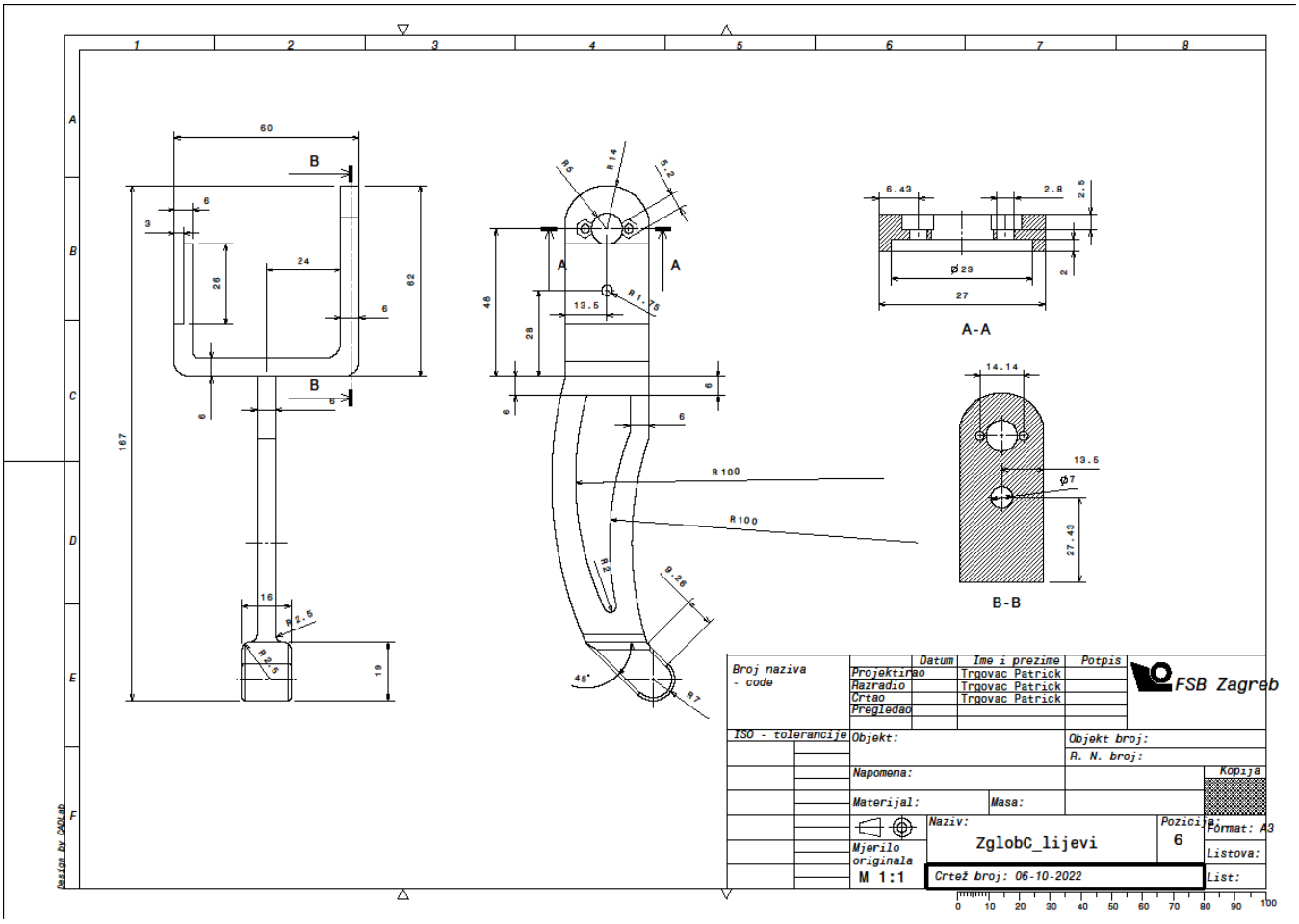
ZglobB



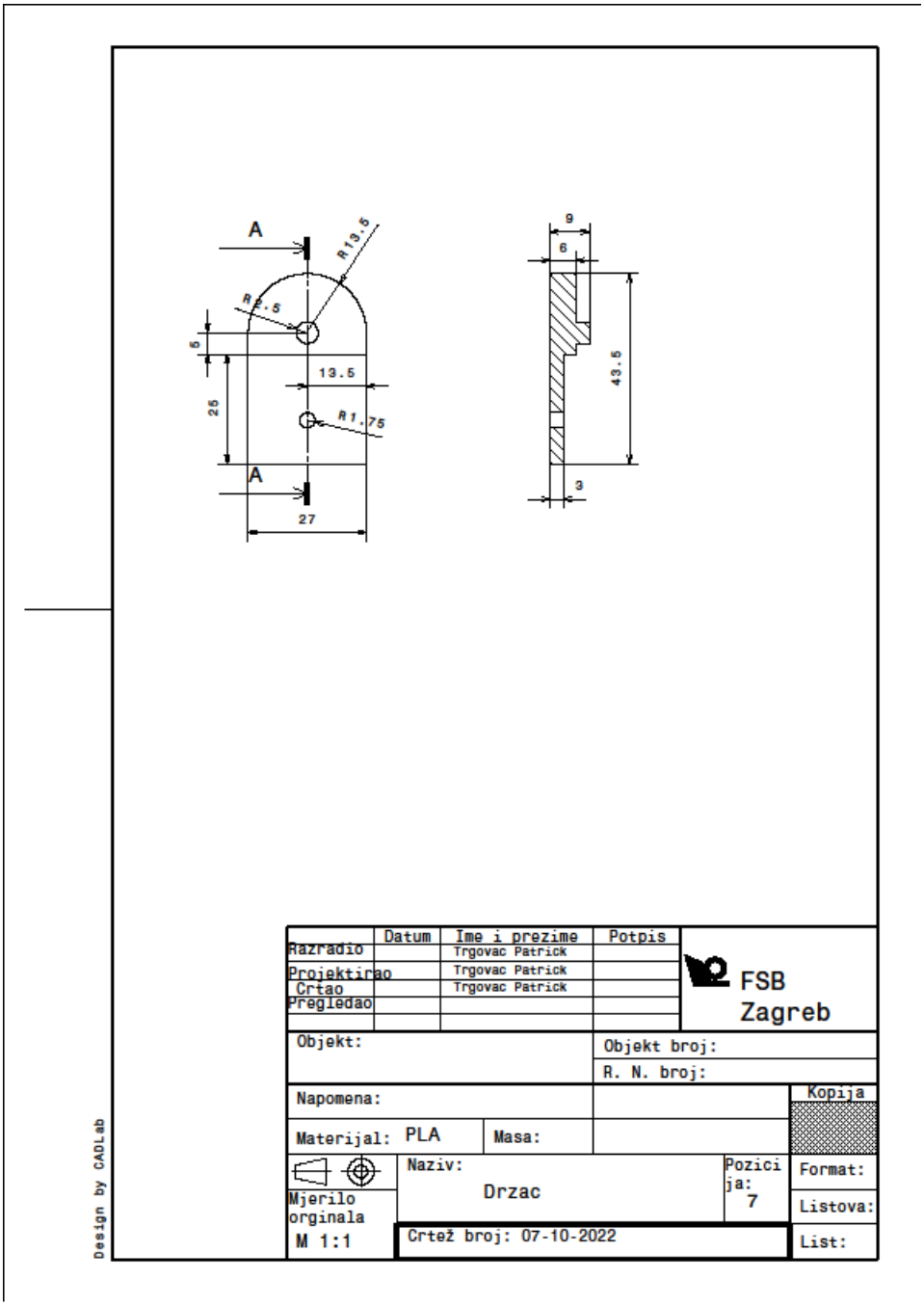
ZglobC_desni



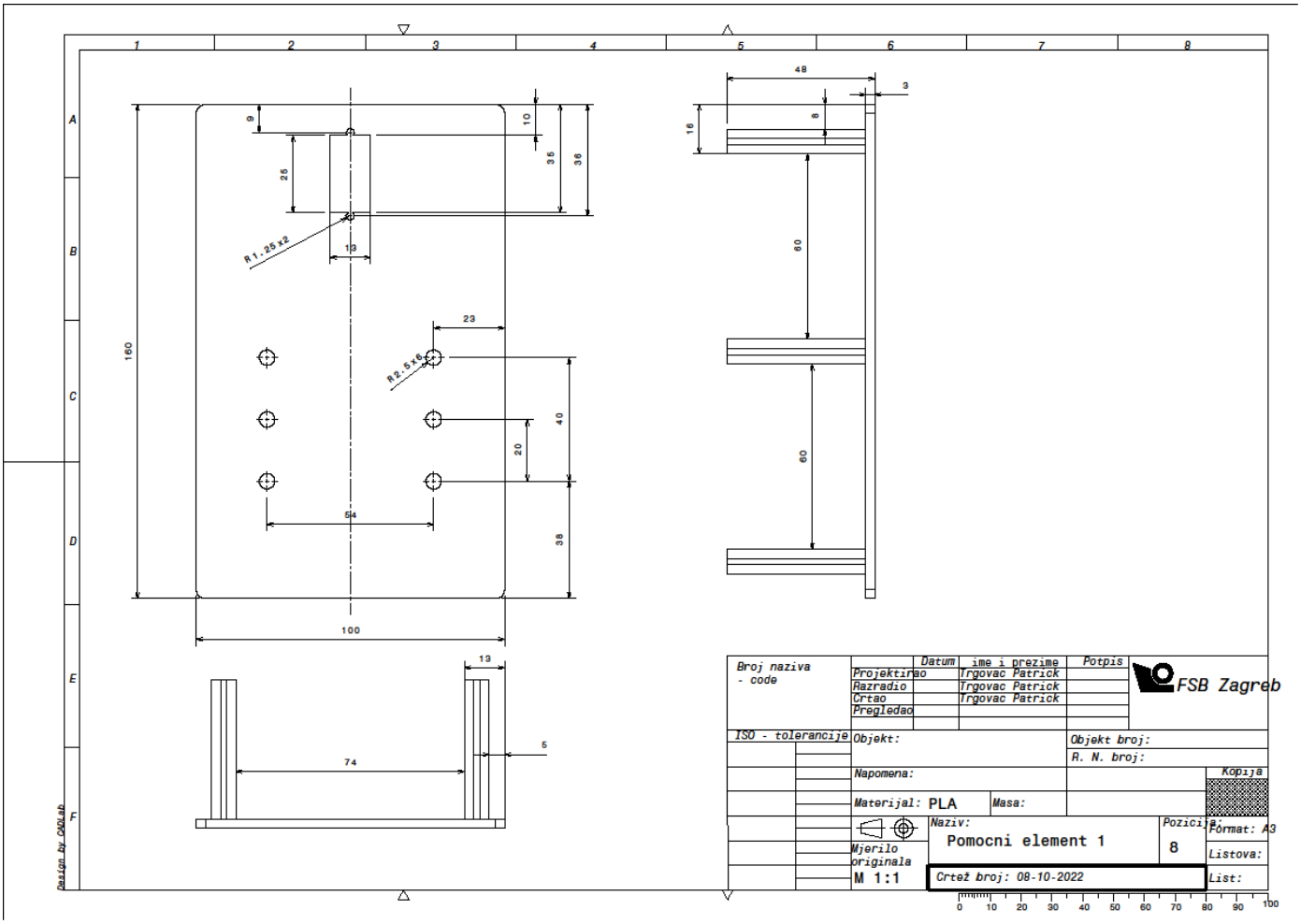
ZglobC_lijevi


A-A
B-B
R100
R100
45°
Broj naziva - code
Datum
Ime i prezime
Potpis
Projektirao Trgovac Patrick
Razradio Trgovac Patrick
Crtao Trgovac Patrick
Pregledao
FSB Zagreb
ISO - tolerancije
Objekt:
Objekt broj:
R. N. broj:
Napomena:
Kopija
Materijal:
Masa:
Naziv:
ZglobC_lijevi
Pozicija:
6
Format: A3
Mjerilo originala
M 1:1
Crtež broj: 06-10-2022
Listova:
List:
Design by CADLab

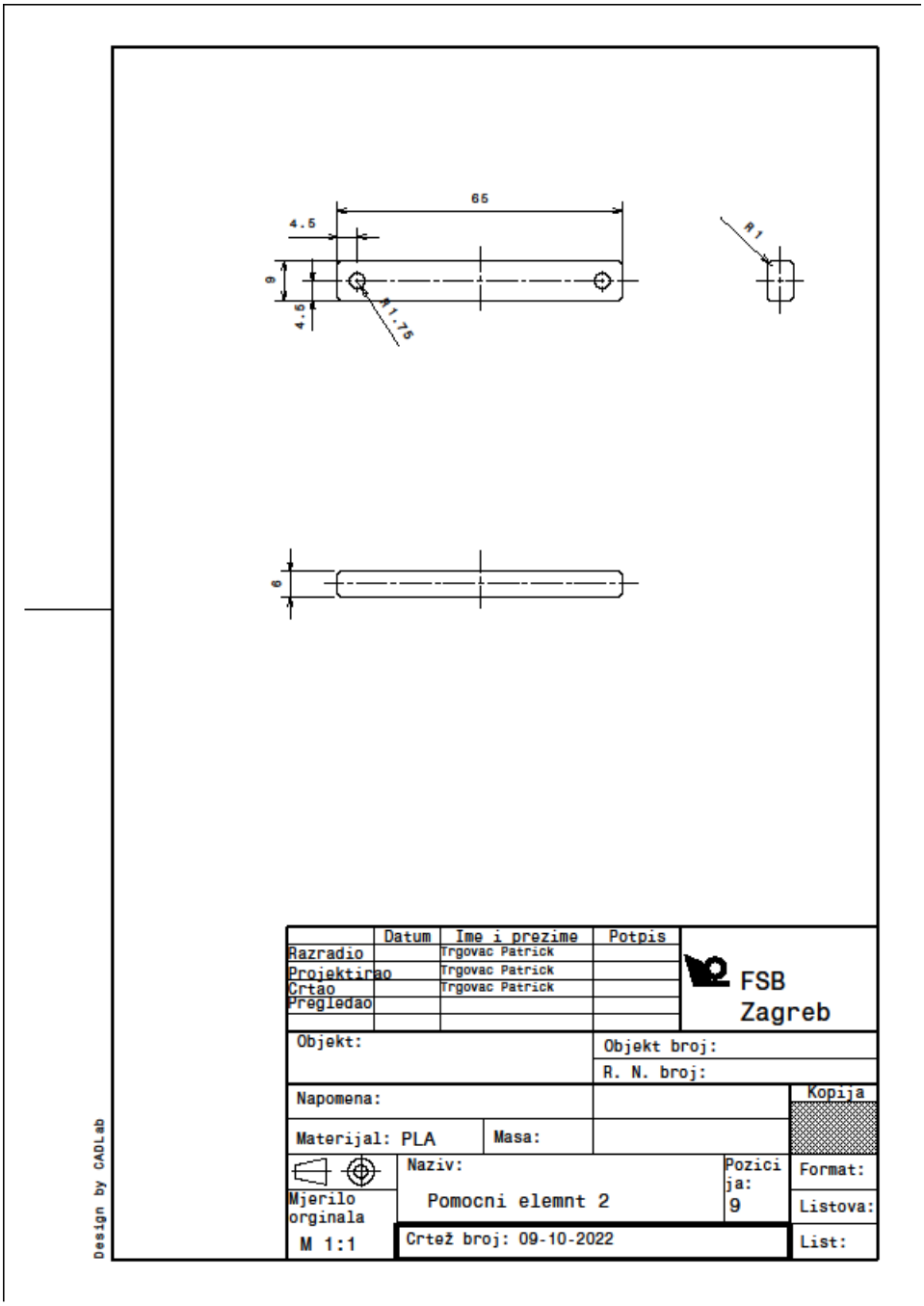
Držač



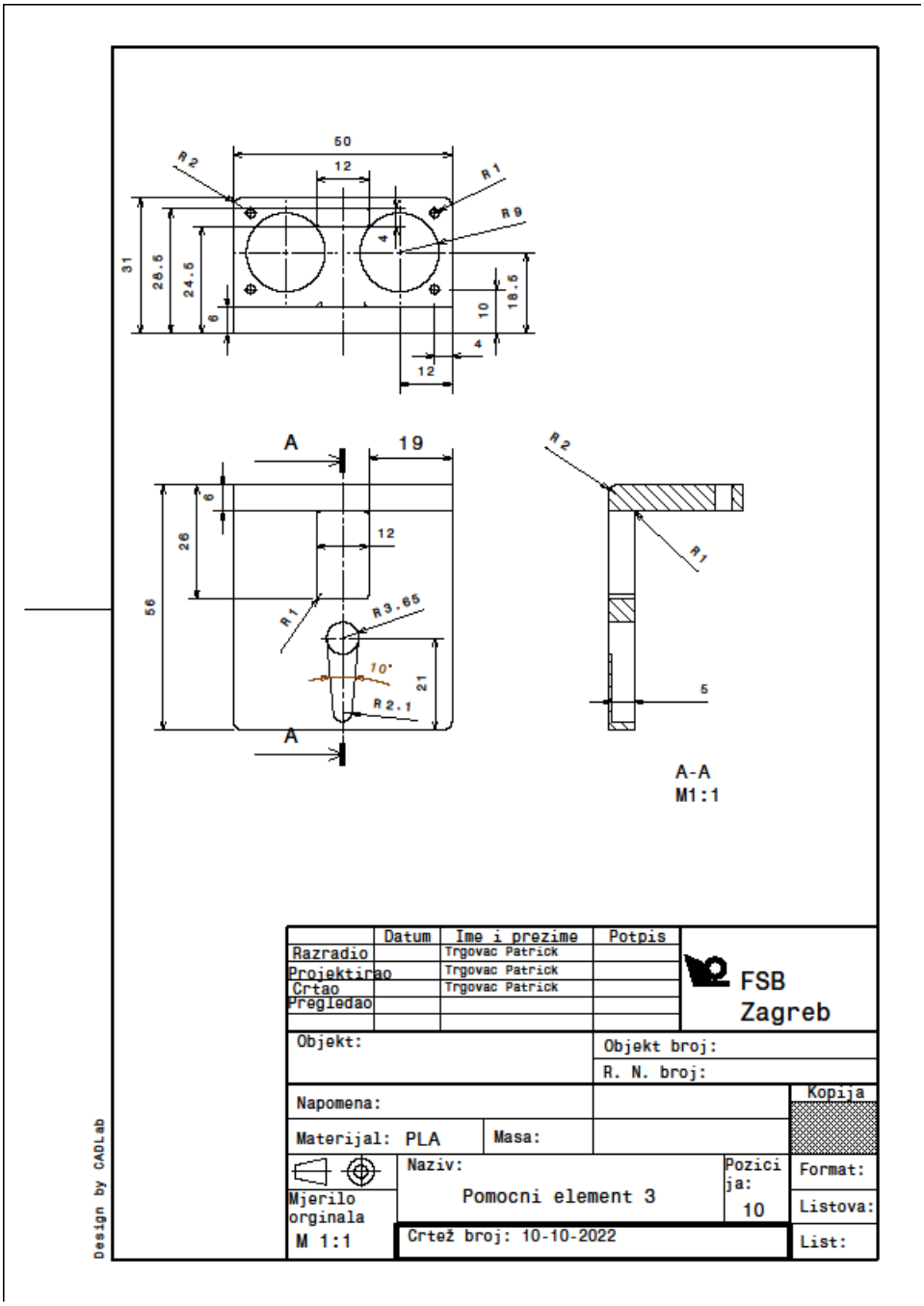
Pomocni element 1



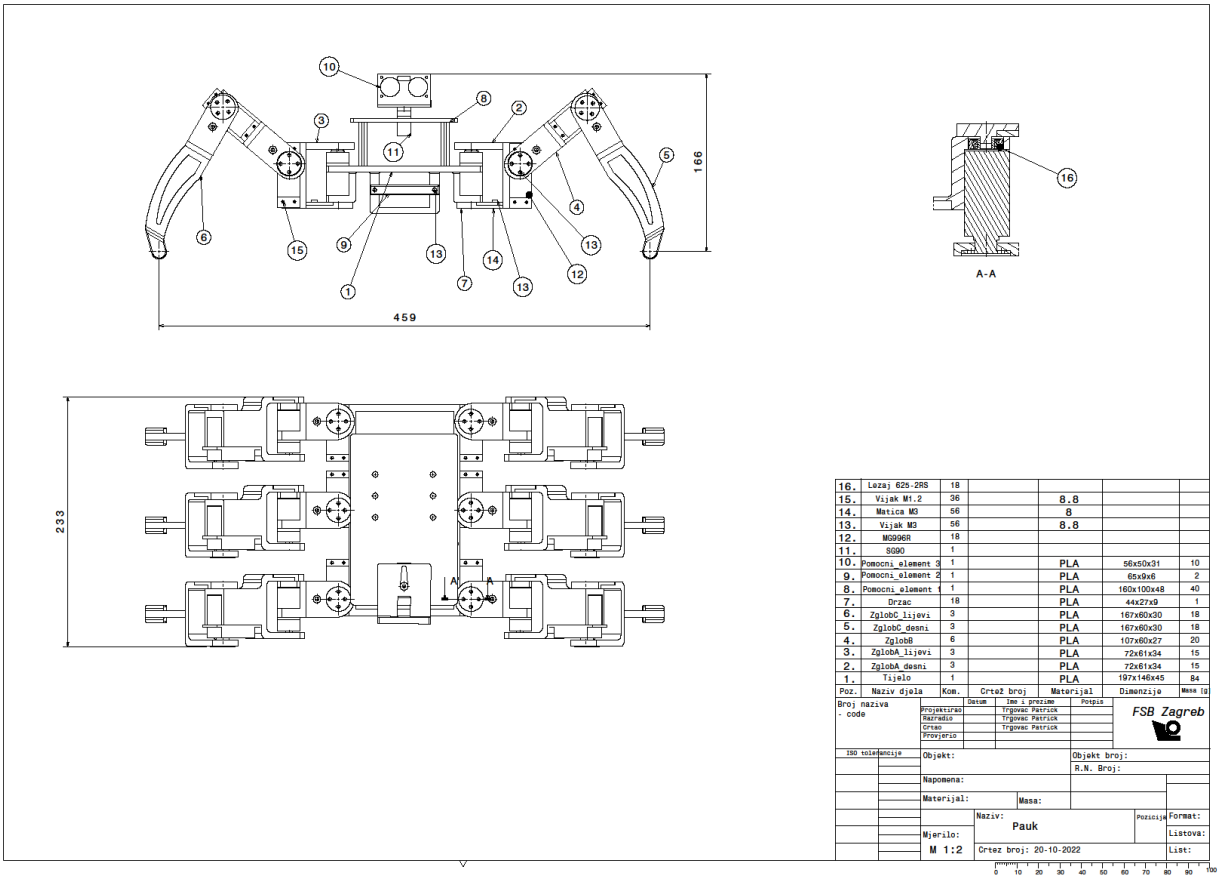
Pomocni Element 2



Pomocni elemnt 3


50
12
R2
R1
R9
31
28.5
24.5
6
4
10
18.6
4
12
A
19
6
26
56
12
R1
R3.65
10°
21
R2.1
A
R2
R1
5
A-A
M1:1
Design by CADLab
Datum
Ime i prezime
Potpis
Razradio
Trgovac Patrick
Projektirao
Trgovac Patrick
Crtao
Trgovac Patrick
Pregledao
FSB
Zagreb
Objekt:
Objekt broj:
R. N. broj:
Napomena:
Kopija
Materijal: PLA
Masa:
Naziv:
Pomocni element 3
Pozicija:
10
Format:
Listova:
Mjerilo orginala
M 1:1
Crtež broj: 10-10-2022
List:

Sklop:



Poz.	Naziv dijela	Kom.	Crtež broj	Materijal	Dimenzije	Masa [g]
16.	Ležaj 625-2RS	18				
15.	Vijak M1.2	36		8.8		
14.	Matica M3	56		8		
13.	Vijak M3	56		8.8		
12.	MG996R	18				
11.	S090	1				
10.	Pomocni_element 3	1		PLA	56x50x31	10
9.	Pomocni_element 2	1		PLA	65x9x6	2
8.	Pomocni_element 1	1		PLA	160x100x48	40
7.	Drzac	18		PLA	44x27x9	1
6.	ZglobC_lijevi	3		PLA	167x60x30	18
5.	ZglobC_desni	3		PLA	167x60x30	18
4.	ZglobB	6		PLA	107x60x27	20
3.	ZglobA_lijevi	3		PLA	72x61x34	15
2.	ZglobA_desni	3		PLA	72x61x34	15
1.	Tijelo	1		PLA	197x146x45	84

Broj naziva - code			
	Projektirao	Trgovac Patrick	
	Razradio	Trgovac Patrick	
	Crtao	Trgovac Patrick	
	Provjerio		
ISO tolerancije	Objekt:	Objekt broj:	
		R.N. Broj:	
	Napomena:		
	Materijal:	Masa:	
	Mjerilo: M 1:2	Naziv: Pauk	Pozicija:
		Crtez broj: 20-10-2022	Format:
			Listova:
			List:

FSB Zagreb