Modeliranje i upravljanje mobilnim inverznim njihalom

Pošta, Dejan

Modeliranje i upravljanje mobilnim inverznim njihalom

Pošta, Dejan

VELEUČILIŠTE U KARLOVCU
Preddiplomski studij mehatronike

MODELIRANJE I UPRAVLJANJE MOBILNIM INVERZNIM NJIHALOM

MODELING AND CONTROL OF THE MOBILE INVERTED PENDULUM

Završni rad

Dejan Pošta
Karlovac, 2019.

IZJAVA:

Izjavljujem da sam ja – student Dejan Pošta, OIB: 74171065181, matični broj: 0112616037, upisan na stručni studij mehatronike akademske godine 2018./2019., radio ovaj rad samostalno, koristeći se znanjem stečenim tijekom obrazovanja, te uz stručnu pomoć i vođenje mentora Denisa Kotarskog, mag. ing.mech., kojem se zahvaljujem na pruženoj pomoći.

U Karlovcu, 2019.

Dejan Pošta

_____

SAŽETAK

U ovom završnom radu prikazano je modeliranje inverznog njihala te sinteza linearnog algoritma upravljanja mobilnim invernim njihalom. Provedena je linearizacija matematičkog modela inverznog njihala te je cijeli sustav realiziran. Na prototipu PID regulator je podešen te su analizirana tri različita pogonska podsustava za robot.

SUMMARY

In this bachelor degree final report, modelling and linear controller synthesis is shown. Linearization of mathematical model of the inverted pendulum is conducted and whole system is realized. On the prototype the PID controller was tuned and three different drive systems were analyzed.

KLJUČNE RIJEČI

Mobilno inverzno njihalo, linearizacija modela, PID regulator, orijentacija robota

KEY WORDS

Mobile inverted pendulum, linearization, PID controller, robot attitude

Sadržaj

Popis slika

Popis tablica

Popis grafova

1 Uvod

U ovom završnom radu opisan je postupak modeliranja, upravljanja i izvedbe mobilnog inverznog njihala. Inverzno njihalo je popularan problem u dinamici i teoriji upravljanja te se koristi kao mjerilo za ispitivanje upravljačkih strategija. Osim što se koristi u robotici, brojne su komercijalne primjene sustava temeljenih na inverznom mobilnom njihalu poput hoverboard-a (Slika 1a) ili segway-a (Slika 1b). Mobilno inverzno njihalo je inherentno nestabilan sustav i teži prevrtanju oko osi rotacije oslonca. Problematika sustava zahtjeva integraciju više disciplina mehatronike. Teorija upravljanja, elektronika i mehanika samo su neke od disciplina kroz koje se ovaj projekt proteže.

Slika 1 - Komercijalne izvedbe sustava inverznog njihala

Cilj ovog završnog rada je matematičkim modelom opisati robot, razraditi sustav upravljanja, obrade signala, mehaničke konstrukcije te izraditi mobilno inverzno njihalo tj. robot koji će se aktivno stabilizirati koristeći elektromotorni pogon i PID regulator.

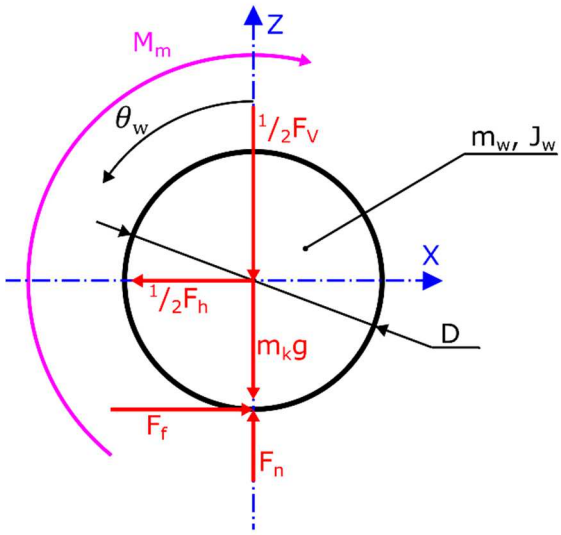
Sustav upravljanja ovog robota je baziran na PID regulatoru. Upravljački algoritam koji se nalazi na Arduino modulu neprestano mjeri trenutnu orijentaciju robota i izračunava korekcijski parametar pomoću kojeg stabilizira robot. Senzorski podsustav koji čini inercijska jedinica služi za mjerenje trenutne orijentacije robota. Upravljački podsustav iz inercijske jedinice dobije neobrađenu informaciju o orijentaciji, odnosno nagibu robota te nad njom vrši digitalnu obradu signala u svrhu dobivanja upotrebljive informacije. Zatim upravljački algoritam PID regulatora izračunava korekcijski parametar iz trenutnog podatka o orijentaciji. Na kraju upravljački podsustav koristi korekcijski parametar za pogon elektromotora koji ispravljaju robot.

2 Matematički model mobilnog inverznog njihala

U ovom poglavlju prikazan je izvod matematičkog modela mobilnog inverznog njihala. Ovaj korak je važan kod projektiranja upravljačkog sustava i regulacije jer nam predočava ponašanje sustava, te omogućuje odabir optimalnog načina upravljanja sustavom (u ovom slučaju robota). Kod kompleksnijih sustava je ovaj korak neophodan za realizaciju rješenja, no za jednostavnije sustave moguće je iz iskustva predočiti upravljački sustav ili gotovo rješenje. Zbog pojednostavljenja modela neki parametri su izostavljeni što je čest slučaj i u praksi za parametre koji su irelevantni ili malo utječu na sustav. Izvod modela temelji se na objavljenim radovima [6, 7].

2.1 Nelinearni model robota

Model robota je podijeljen na model kotača robota i na model šasije (konstrukcije) robota. Za svaki od ta dva dijela modela, postavljaju se sume sila i momenata (2.1 do 2.3).



Slika 2 - Kotač sa označenim silama i momentima

Kotač robota odvojen je od ostatka robota te su prikazane sile i momenti (engl. *free body diagram)* na slici 2. Postavljene su jednadžbe sume sila i momenata

$$\sum F_x = 0: \qquad m_w \ddot{x} = -\frac{1}{2}F_h + F_f - b\dot{x}, \tag{2.1}$$

$$\sum F_z = 0: \qquad m_w \ddot{z} = F_n - \frac{1}{2}F_v - m_w g, \tag{2.2}$$

$$\sum M_w = 0: \qquad J_w \ddot{\theta}_w = M_m - F_f R, \tag{2.3}$$

gdje je:

- m_w – masa kotača,
- F_h - horizontalna sila između kotača i osovine motora,
- F_v - vertikalna sila između kotača i osovine motora,
- b - koeficijent otpora gibanju,
- F_f – sila trenja,
- F_n - vertikalna sila podloge na kotač
- J_w - moment inercije kotača oko osi rotacije,
- M_m – moment motora,
- R – polumjer, odnosno D promjer kotača.

Translacijska brzina kotača jednaka je

$$\dot{x} = R\dot{\theta}_w = \frac{D}{2}\dot{\theta}_w, \tag{2.4}$$

gdje je $\dot{\theta}_w$ je kutna brzina kotača. Slijedi da je translacijsko ubrzanje kotača jednako

$$\ddot{x} = R\ddot{\theta}_w = \frac{D}{2}\ddot{\theta}_w. \tag{2.5}$$

Sada je moguće iz formule 2.5 izraziti kutno ubrzanje kotača

$$\ddot{\theta}_w = \frac{\ddot{x}}{R}. \tag{2.6}$$

Uvrštavanjem izraza 2.6 u 2.3 dobije se

$$J_w \frac{\ddot{x}}{R} = M_m - F_f R, \tag{2.7}$$

iz čega možemo izraziti silu trenja

$$F_f = \frac{M_m}{R} - J_w \frac{\ddot{x}}{R^2}. \tag{2.8}$$

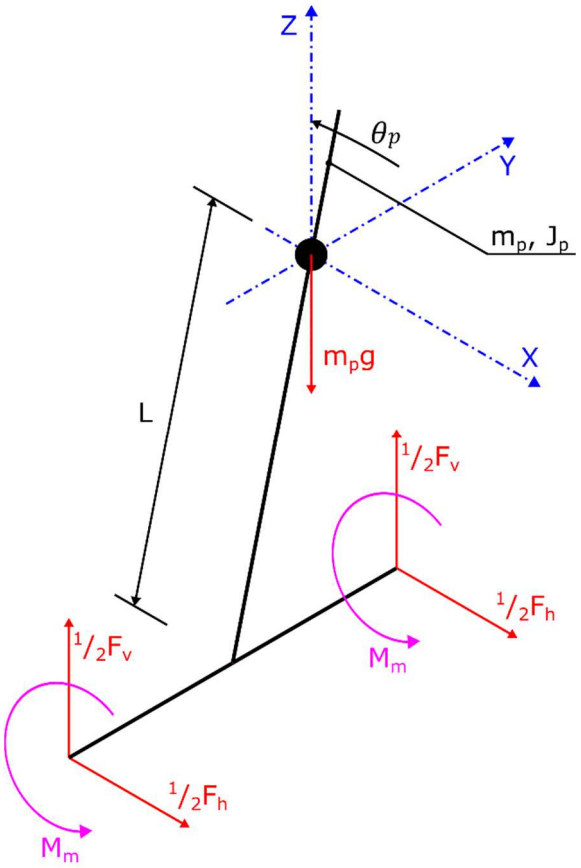
Uvrštavanjem izraza 2.8 u 2.1 dobije se

$$m_w\ddot{x} = -\frac{1}{2}F_h + \frac{M_m}{R} - J_w \frac{\ddot{x}}{R^2} - b\dot{x}. \tag{2.9}$$

Izlučivanjem translacijskog ubrzanja slijedi

$$\ddot{x}\left(m_w + \frac{J_w}{R^2}\right) = \frac{M_m}{R} - \frac{1}{2}F_h - b\dot{x}. \tag{2.10}$$

Zatim se šasija robota odvoji od ostatka robota te se prikažu sile i momenti (slika 3).



Slika 3 - Šasija robota sa označenim silama i momentima

Postavljene su jednadžbe sume sila i momenata za šasiju robota

$$\sum F_x = 0: \qquad m_p \ddot{x}_p = F_h, \tag{2.11}$$

$$\sum F_z = 0: \qquad m_p \ddot{z}_p = F_v - m_p g. \tag{2.12}$$

gdje je m_p masa šasije (robota). Izlučivanjem vertikalne sile u jednadžbi 2.12, dobije se

$$F_v = m_p(\ddot{z}_p + g). \tag{2.13}$$

Suma momenata oko težišta šasije robota jednaka je

$$\sum M = 0: \qquad J_p \ddot{\theta}_p = -2M_m - F_h L cos(\theta_p) + F_v L sin(\theta_p), \tag{2.14}$$

gdje je:

- J_p – moment inercije njihala oko osi rotacije,
- θ_p - kut nagiba njihala,
- L – udaljenost od osi kotača do težišta robota.

Za dobivanje potrebnih vertikalnih i horizontalnih sila, potrebno je pronaći vezu pomaka kotača i težišta njihala sa kutom nagiba njihala

$$x_p = x + L\sin(\theta_p). \tag{2.15}$$

Dvostrukim deriviranjem jednadžbe 2.15 dobije se željena veza varijabli,

$$\dot{x}_p = \dot{x} + L\cos(\theta_p)\dot{\theta}_p, \tag{2.16}$$

$$\ddot{x}_p = \ddot{x} + L\cos(\theta_p)\ddot{\theta}_p - L\sin(\theta_p){\dot{\theta}_p}^2. \tag{2.17}$$

Zatim se postupak ponavlja za vertikalnu komponentu

$$z_p = L\cos(\theta_p), \tag{2.18}$$

$$\dot{z}_p = -L\sin(\theta_p)\dot{\theta}_p, \tag{2.19}$$

$$\ddot{z}_p = -L\sin(\theta_p)\ddot{\theta}_p - L\cos(\theta_p){\dot{\theta}_p}^2. \tag{2.20}$$

Uvrštavanjem izraza 2.17 u 2.11 i 2.20 u 2.13 dobivamo jednadžbe za horizontalnu i vertikalnu silu

$$F_h = m_p\left(\ddot{x} + L\cos(\theta_p)\ddot{\theta}_p - L\sin(\theta_p){\dot{\theta}_p}^2\right), \tag{2.21}$$

$$F_v = m_p\left(-L\sin(\theta_p)\ddot{\theta}_p - L\cos(\theta_p){\dot{\theta}_p}^2 + g\right). \tag{2.22}$$

Jednadžbe 2.21 i 2.22 se zatim uvrštavaju u jednadžbe 2.10 i 2.14

$$\ddot{x}\left(m_w + \frac{J_w}{R^2}\right) = \frac{M_m}{R} - \frac{1}{2}m_p\left(\ddot{x} + L\cos(\theta_p)\ddot{\theta}_p - L\sin(\theta_p){\dot{\theta}_p}^2\right) - b\dot{x}, \tag{2.23}$$

$$\begin{aligned} J_p\ddot{\theta}_p = {} & -2M_m - m_p\left(\ddot{x} + L\cos(\theta_p)\ddot{\theta}_p - L\sin(\theta_p){\dot{\theta}_p}^2\right)L\cos(\theta_p) \\ & + m_p\left(-L\sin(\theta_p)\ddot{\theta}_p - L\cos(\theta_p){\dot{\theta}_p}^2 + g\right)L\sin(\theta_p). \end{aligned} \tag{2.24}$$

Sređivanjem jednadžbi 2.23 i 2.24 dobiju se diferencijalne jednadžbe mobilnog inverznog njihala 2.25 i 2.26:

$$\ddot{x}\left(m_w + \frac{J_w}{R^2} + \frac{1}{2}m_p\right) = \frac{M_m}{R} - \frac{1}{2}m_pL\cos(\theta_p)\ddot{\theta}_p + \frac{1}{2}m_pL\sin(\theta_p){\dot{\theta}_p}^2 - b\dot{x}, \tag{2.25}$$

$$\ddot{\theta}_p(J_p + m_pL^2) = -2M_m - \ddot{x}m_pL\cos(\theta_p) + gm_pL\sin(\theta_p). \tag{2.26}$$

2.2 Linearizacija modela

Upravljački sklop ovog robota će koristiti linearni PID regulator. PID regulator nije optimalan sustav regulacije jer je linearan te su njegove performanse u nelinearnim sustavima ograničene. Ovaj sustav je nelinearan, no ako se linearizira oko radne točke i drži u određenim granicama možemo uspješno regulirati sustav. Radna točka inverzno njihala je u/oko θ = 0, ovisno koliko toleriramo grešku aproksimacije malih kutova.

Za aproksimaciju malih kutova vrijede izrazi 2.27 i 2.28.

$$\sin\theta = \theta \tag{2.27}$$

$$\cos\theta = 1 \tag{2.28}$$

Također se može zanemariti kvadrat brzine rotacije (izraz 2.29).

$$\dot{\theta}^2 = 0 \tag{2.29}$$

Koristeći izraz 2.27 može se dobiti izraz za grešku aproksimacije 2.28 za sin θ.

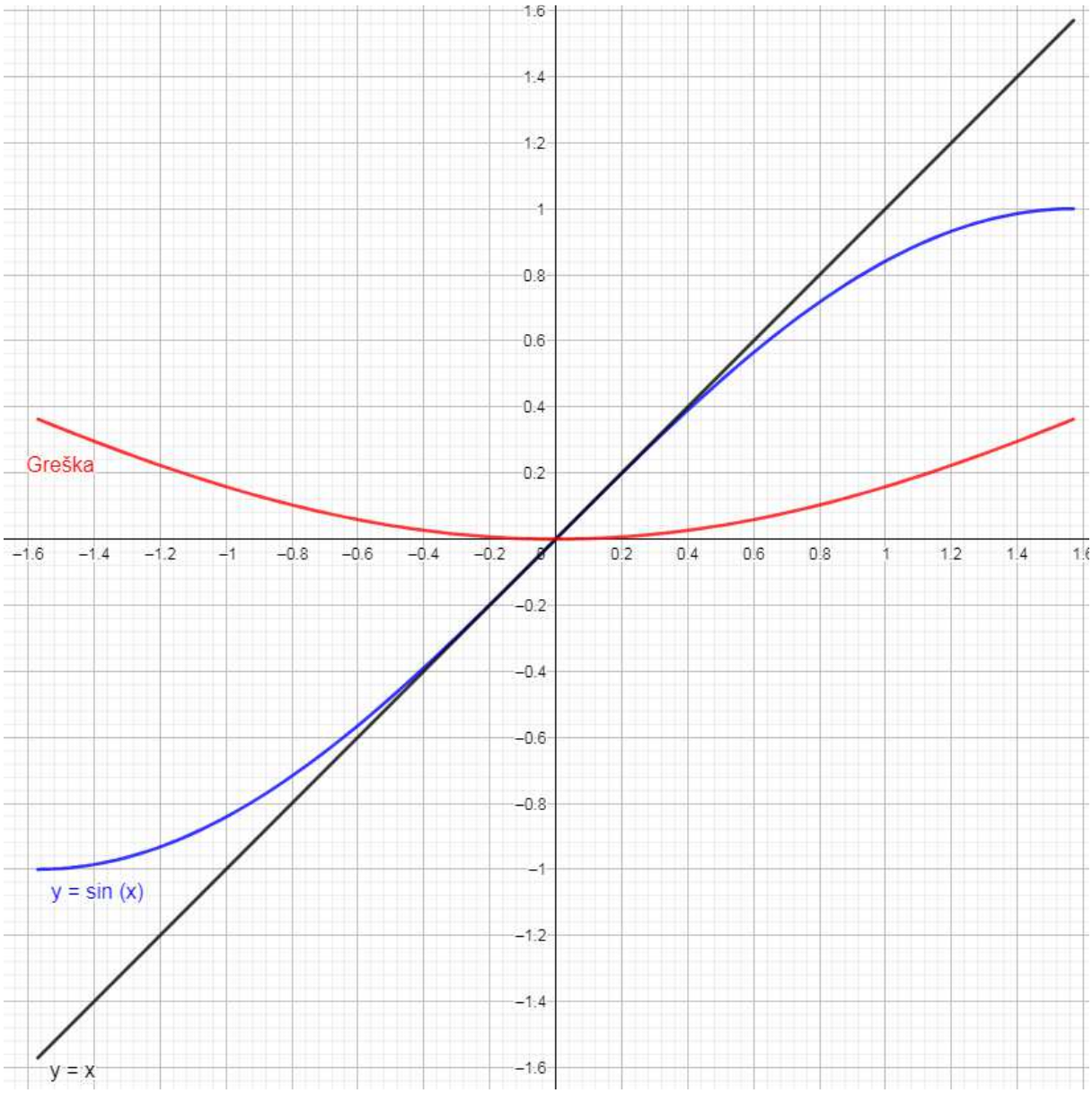
$$Greška(x) = 1 - \frac{\sin x}{x} \tag{2.30}$$

Vidljivo je da je najveće odstupanje na $x = \pm\frac{\pi}{2}$. Ako se taj podatak uvrsti u 2.30 dobijemo 2.31. Greška za sin θ je računata samo kao primjer odstupanja.

$$Greška\left(\frac{\pi}{2}\right) = 0{,}3634 \tag{2.31}$$

To je greška od čak 36.34 %, no to je greška za kut od 90°. Ako se robot drži u granicama od otprilike ±15°, maksimalna greška će biti 1.14%. To je jako malo odstupanje a drastično pojednostavljuje sustav.

Graf 1 prikazuje grešku aproksimacije (crveno, Greška(x)) za izraz 2.27.



Graf 1 - Greška aproksimacije

Ako se primjene izrazi 2.27 do 2.29 na izraze 2.25 i 2.26 dobije se linearizirani matematički model robota 2.32 i 2.33.

$$\ddot{x}\left(m_w + \frac{J_w}{R^2} + \frac{1}{2}m_p\right) = \frac{M_m}{R} - \frac{1}{2}m_p L\ddot{\theta}_p - b\dot{x}, \tag{2.32}$$

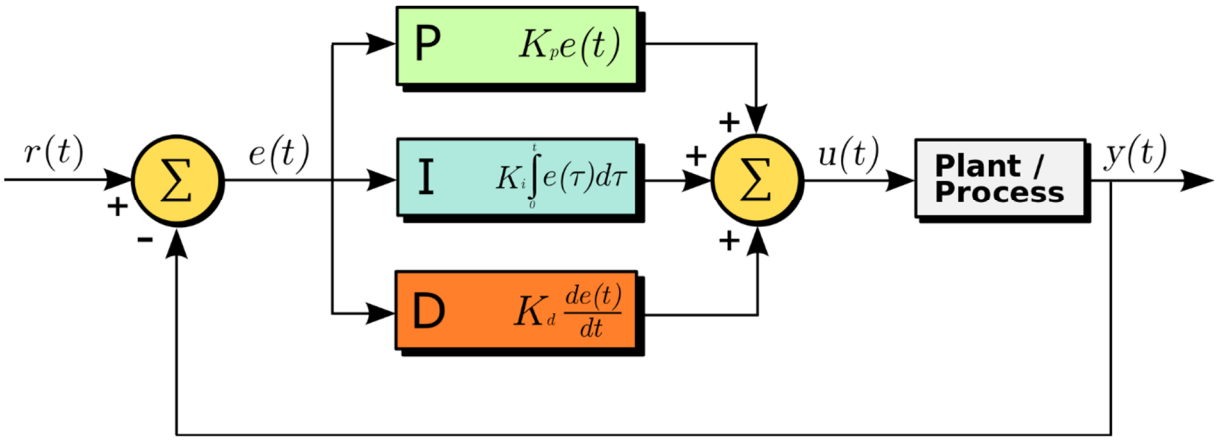
$$\ddot{\theta}_p\left(J_p + m_p L^2\right) = -2M_m - \ddot{x}m_p L + g m_p L\theta_p. \tag{2.33}$$

3 Upravljanje orijentacijom robota

Mobilno inverzno njihalo, osim što je inherentno nestabilni nelinearni sustav, je i podupravljani sustav što znači da robot ima više stupnjeva slobode gibanja od broja aktuatora. Moguće je kombinirati upravljanje orijentacijom i pozicijom robota, a u radu će se zbog kompleksnosti sustava razmatrati upravljanje orijentacijom koje treba osigurati stabilnost robota. S obzirom na resurse i predviđene komponente realnog sustava, odabran je PID regulator koji regulira kut otklona robota tj. njegovu orijentaciju.

3.1 PID regulator

PID regulator je upravljački mehanizam s povratnom vezom u kojem regulator konstantno računa grešku (odstupanje e(t); Slika 4) sustava od zadane vrijednosti i računa korekciju s obzirom na proporcionalnu, integralnu i derivativnu komponentu, po čemu je i dobio i ime. PID regulator (Slika 4) u obzir uzima postavljenu vrijednost r(t), e(t) je odstupanje sustava od postavljene vrijednosti. Dio PID regulatora je i sustav (engl. *plant*) čiji odziv koristi kao povratnu vezu. Ovaj način regulacije je kontinuiran i koristi stalna mjerenja senzorima i radi korekcije na sustav da on ostane na postavljenoj vrijednosti koristeći izvršni element (aktuator; u ovom slučaju pogonski podsustav s dva motora).



Slika 4 – Blok shema PID regulatora

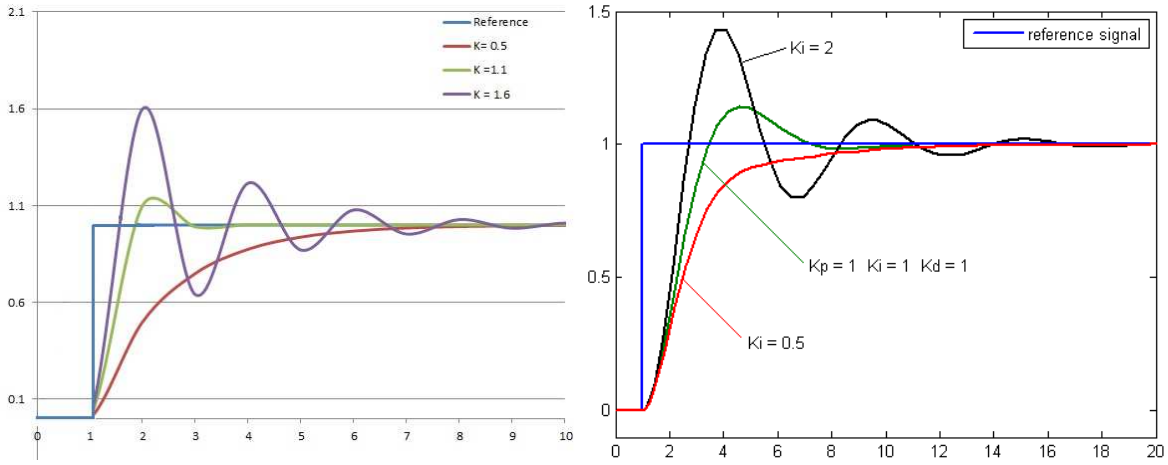
PID regulator u matematičkom obliku prima oblik prikazan na izrazu 3.1.

$$u(t) = K_P e(t) + K_I \int e(\tau)\, d\tau + K_D \dot{e}(t) \tag{3.1}$$

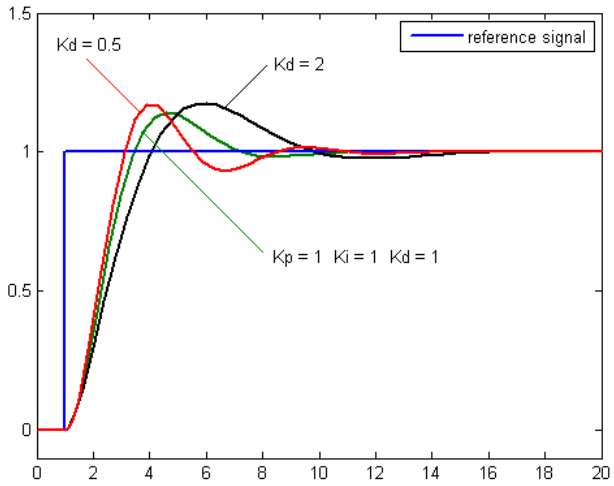
Greška (odstupanje) od zadane vrijednosti r(t) se računa prema izrazu 3.2

$$e(t) = r(t) - y(t) \tag{3.2}$$

PID regulator opisuju 3 parametra (K_P, K_I i K_D; izraz 3.1). Manipulacijom tih parametara se može mijenjati ponašanje PID regulatora (slika 5, slika 6, Tablica 1). Ti parametri se moraju podesiti da bi se uspješno upravljalo sustavom.



Slika 5 – Efekti promjene K_P i K_I parametara na odziv regulatora



Slika 6 – Efekti promjene K_D parametara na odziv regulatora

Parametar	Vrijeme rasta	Prebačaj	Vrijeme smirivanja	Greška ustaljenog stanja	Stabilnost
K_P	Smanjuje	Povećava	Mala promjena	Smanjuje	Degradira
K_I	Smanjuje	Povećava	Povećava	Eliminira	Degradira
K_D	Mala promjena	Smanjuje	Smanjuje	Nema utjecaja u teoriji	Poboljšava ako je K_D mali

Tablica 1 – Utjecaj parametara PID regulatora na ponašanje sustava

Podešavanje PID regulatora je moguće na više načina. U ovom slučaju regulator je podešen Ziegler–Nichols metodom. Podešavanje tom metodom je jednostavno i moguće je dobiti

zadovoljavajuće rezultate koje je po potrebi kasnije moguće finije podesiti. Podešavanje Ziegler–Nichols metodom obavlja se na sljedeći način:

1. K_I i K_D se postave na nulu.
2. K_P se povećava do vrijednosti K_U na kojoj sustav počne oscilirati.
3. Očita se period oscilacije T_U na parametru K_U.
4. Iz tablice (Tablica 2) se očitaju empiričke vrijednosti parametara PID regulatora.

Tip regulatora	K_P	K_I	K_D
P	$0.5\ K_U$		
PI	$0.45\ K_U$	$0.54K_U / T_U$	
PID	$0.6\ K_U$	$1.2\ K_U / T_U$	$3\ K_UT_U / 40$

Tablica 2 - Vrijednosti parametara PID – prema Ziegler–Nichols

PID regulator se također može ručno podesiti (iskustveno) znajući posljedice promjene pojedinog parametra (Tablica 1), ali ta metoda zahtjeva dosta iskustva. PID regulator u ovom projektu je podešen iskustveno.

4 Komponente i podsustavi robota

U ovom poglavlju opisan je sustav robota podijeljen na 3 podsustava: pogonski, upravljački i senzorski podsustav robota. Pod pogonskim podsustavom robota će biti objašnjeni i navedeni pogonski elementi koji su korišteni za izvedbu robota. Pod upravljačkim podsustavom će biti objašnjena uloga upravljačke jedinice i upravljačko sklopovlje, te njihove funkcije. Pod senzorskim podsustavom će biti opisani senzori te elementi obrade signala.

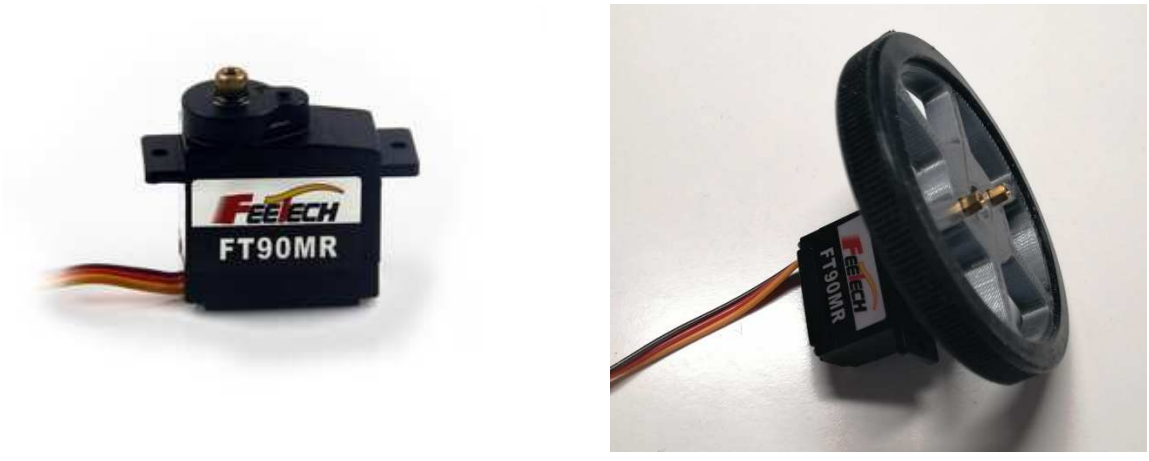
4.1 Pogonski podsustav

Pod pogonskim podsustavom podrazumijevamo aktuatore sa zadaćom propulzije robota. Pogonski sustav se sastoji od pretvornika energije i izvršnog člana. Pretvornik energije u ovim slučajevima su bili elektromotori različitih izvedbi. Izvršni članovi su zupčasti prijenosnik i kotač, koji smanjuju brzinu vrtnje izlaznog vratila motora, te pretvaraju rotacijsko gibanje u linearno gibanje respektivno gledano.

U pogonskom podsustavu kroz razvoj projekta je bilo različitih sustava koji su bili razmatrani i korišteni. Neke od izvedbi podsustava nisu imale zadovoljavajuće performanse. Neki od tih sustava su imali i mane koje je bilo neisplativo ili nemoguće ispraviti.

4.1.1 Izvedba sa servo motorima

Prvi pogonski podsustav koji je korišten na robotu je takozvani servo pogon. Korišten je FT90MR servo motor sa kontinuiranom rotacijom (**Error! Reference source not found.**). Na izlazno vratilo je spojen kotač.



a) b)

Slika 7 - FT90MR servo motor a), servo motor sa kotačem b)

FT90MR ima 3 priključka, od kojih su 2 za napajanje, dok je jedan za upravljački signal koji dijeli nulu sa napajanjem.

Parametar	Vrijednost	Komentar
Brzina praznog hoda	130 min^{-1}	Pri naponu od 6V
Moment pri 0 min^{-1}	1.5 kg/cm	Pri naponu od 6V
Masa	9 g	
Dimenzije [mm]	22.9 × 12.1 × 26.5	Širina, visina, debljina

Tablica 3 – Parametri servo motora

FT90MR servo motor kao upravljački signal koristi pravokutni valni oblik napona kojim se može regulirati njegova brzina. Brzina i smjer vrtnje servomotora se upravlja duljinom poluperioda signalnog višeg napona (5V) pulsno širinskom modulacijom (engl. *pulse width modulation - PWM*). Maksimalna brzina motora je 130 min^{-1} (Tablica 3).

Upravljački signal je bio generiran koristeći Timer1 modul prisutan na Arduinu. Mikroupravljač koji se koristi na Arduinu je Atmega328p, te je u upravljačkom kodu prema dokumentaciji podešen spomenuti timer na sljedeći način:

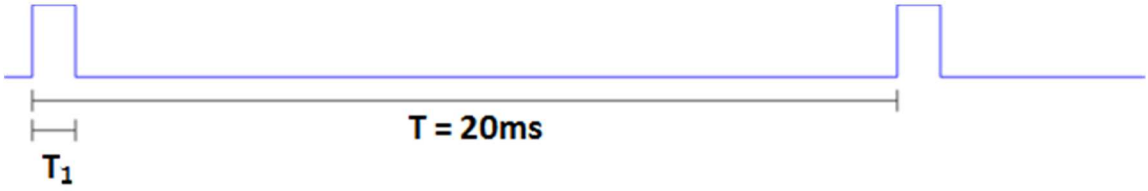
1. Prescaler za timer je 1, tj. Timer1 radi na 16MHz
2. PWM mod rada je 14 (Fast PWM), maksimum je ICR1 registar. (Tablica 15-5 u dokumentaciji za Atmega328p, koja je priložena)
3. OC1A i OC1B se resetiraju na TCNT1 = 0 a postavljaju na OCR1A = 0 ili OCR1B ovisno o priključku.

```
TCCR1A = (1<< COM1A1)| (1<< COM1B1)|(1<<WGM11);
TCCR1B = (1<<WGM13)|(1<<WGM12)|(1<<CS10);
ICR1 = 0x1388; //5000
```

Kod 1 - timer

Frekvencija PWM-a je 50Hz. Željena frekvencija PWM-a se dobije postavljanjem ICR1 registra na željenu vrijednost, koja se izračunava po formuli 4.1.

$$ICR1 = \frac{F_{CPU}}{prescaler \cdot frekvencija} = \frac{16000000}{64 \cdot 50} = 5000. \qquad (4.1)$$



Slika 8 – Duljina poluperioda

T_1 je duljina poluperioda gdje je napon više logičke razine (Slika 8). T je period signala koji iznosi 20ms. Frekvencija signala je 50Hz. Sa podacima o motoru možemo derivirati formulu za brzinu okretaja, znajući korijen funkcije u 1.5ms i maksimalnu brzinu okretaja od 130 min^{-1} na periodu od 2ms (4.2).

$$rpm[min^{-1}] = 260 \cdot T_1 - 390. \tag{4.2}$$

Formula je dobivena uvrštavajući radne točke u jednadžbu pravca. Ovo je idealan slučaj jer ovaj tip servomotora ima mrtvo područje oko 1.5ms, gdje se uopće ne vrti. Mrtvo područje je odprilike 0.5 ms i razlikuje se od motora do motora, ali se može podesiti potenciometrom na motoriću. U radu je softverski rješen problem koristeći zasebno područje za svaki motor.

U upravljačkom djelu mrtvo područje je ispravljeno izrazom 4.3:

$$izlaz[\mu s] = \frac{rpm + 390}{260} + sgn(rpm) * 500. \tag{4.3}$$

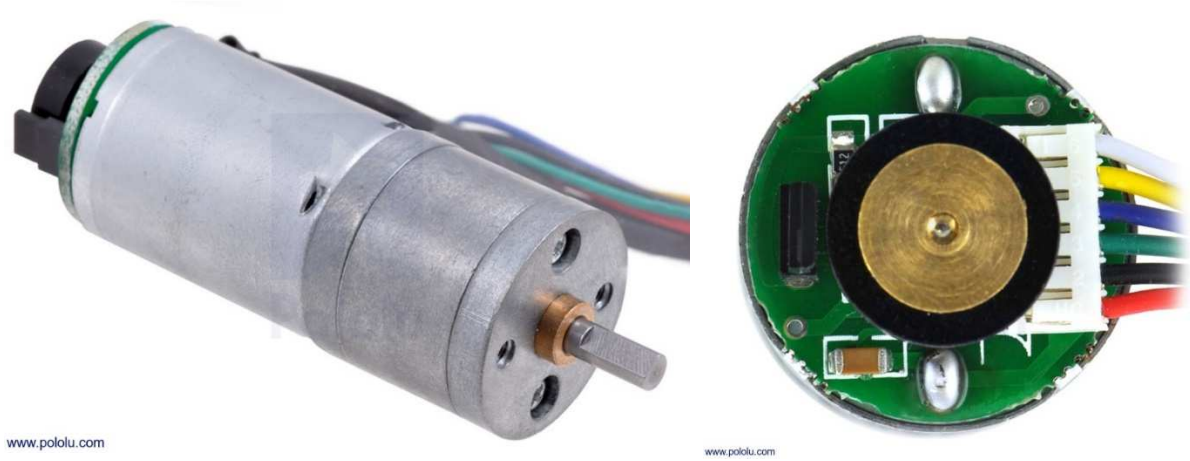
Koristeći kotač promjera 60mm dobijemo maksimalnu brzinu robota od 40.84 cm s^{-1} (izraz 4.4).

$$v = 130min^{-1} \cdot \pi \cdot 6cm = 2450.44\ \frac{cm}{min} = 40.84\ \frac{\text{cm}}{s}, \tag{4.4}$$

što je dovoljna brzina. U ranim stadijima dizajna robota to je bila prihvatljiva brzina. No taj pogon se ipak pokazao nedovoljnim zbog premalog ubrzanja, tj motori nisu imali dovoljno brz odziv da isprave robot od padanja. Također prisutnost mrtvog područja je otežala upravljački sustav jer su motorići imali različita mrtva područja, što je zahtjevalo dodatna empirička rješenja.

4.1.2 Izvedba s istosmjernim motorima

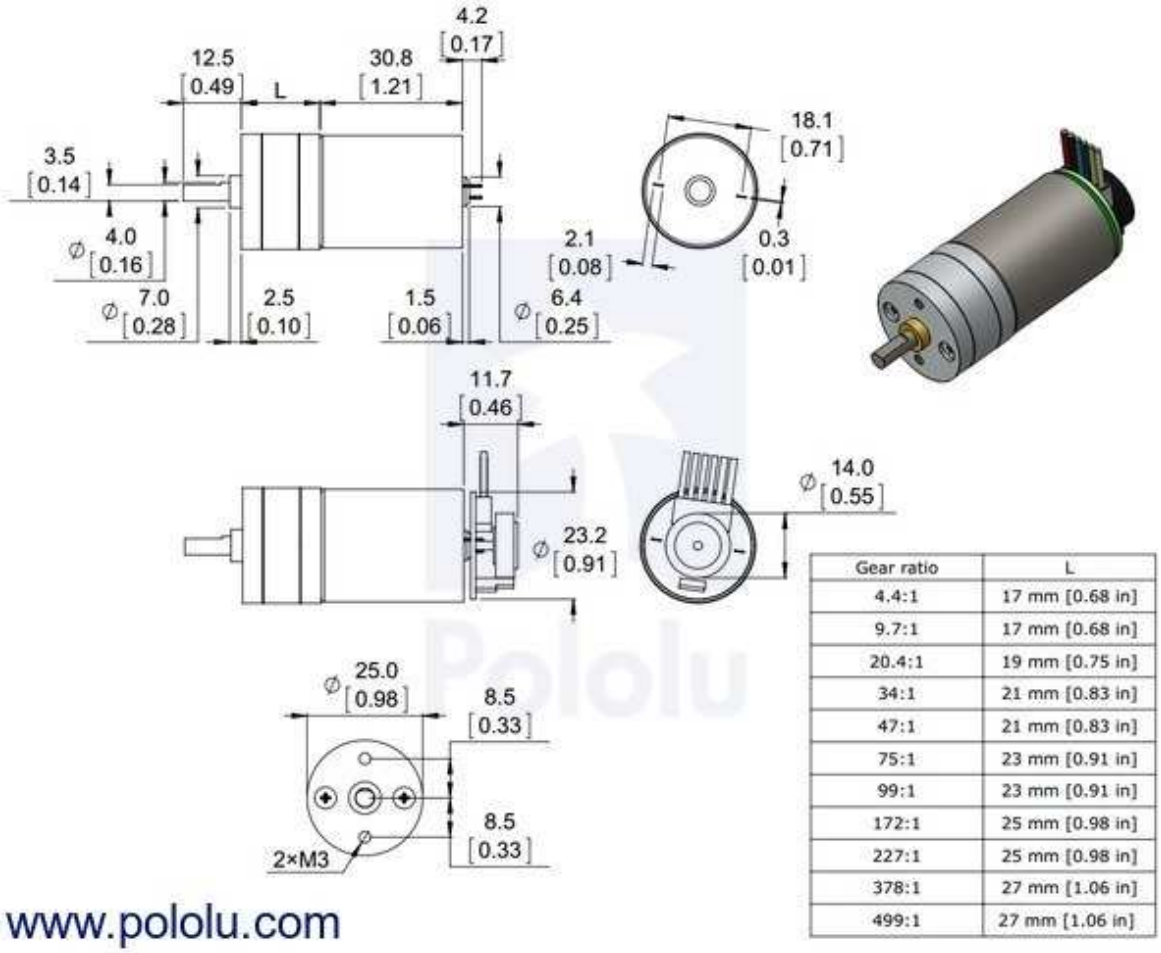
Sljedeći pogonski podsustav koji je korišten na robotu je sustav s istosmjernim motorom. Ovaj istosmjerni motor raspolaže sa većim momentom i brzinom praznog hoda. Korišteni motor je RK370SD (Slika 9), koji je spojen na višestupanjski zupčasti reduktor sa redukcijom od 20.4:1. Na slici 9b prikazan je enkoder motora (64 pulsa po okretaju).



a) b)

Slika 9 – Istosmjerni motor RK370SD

Enkoder je spojen izravno na vratilo elektromotora i ima rezoluciju od 64 impulsa po revoluciji. Enkoder je baziran na Hall-ovom efektu. NA slici 10 prikazane su dimenzije (gabariti) aktuatora.



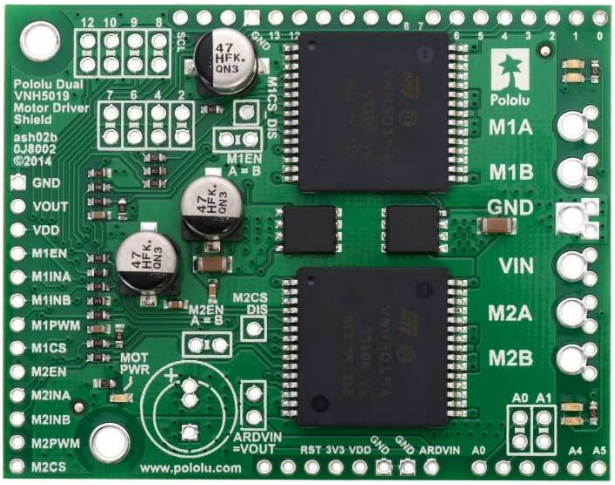
Gear ratio	L
4.4:1	17 mm [0.68 in]
9.7:1	17 mm [0.68 in]
20.4:1	19 mm [0.75 in]
34:1	21 mm [0.83 in]
47:1	21 mm [0.83 in]
75:1	23 mm [0.91 in]
99:1	23 mm [0.91 in]
172:1	25 mm [0.98 in]
227:1	25 mm [0.98 in]
378:1	27 mm [1.06 in]
499:1	27 mm [1.06 in]

Slika 10 - Dimenzije DC motora

Parametar	Vrijednost	Komentar
Brzina praznog hoda	290 min^{-1}	Pri naponu od 6V
Moment pri 0 min^{-1}	2.4 kg/cm	Pri naponu od 6V
Struja pri 0 min^{-1}	2.4 A	Pri naponu od 6V
Masa	9 g	
Omjer redukcije	20.4:1	
Rezolucija enkodera	64	Impulsa po revoluciji motora

Tablica 4 – Podaci korištenog aktuatora.

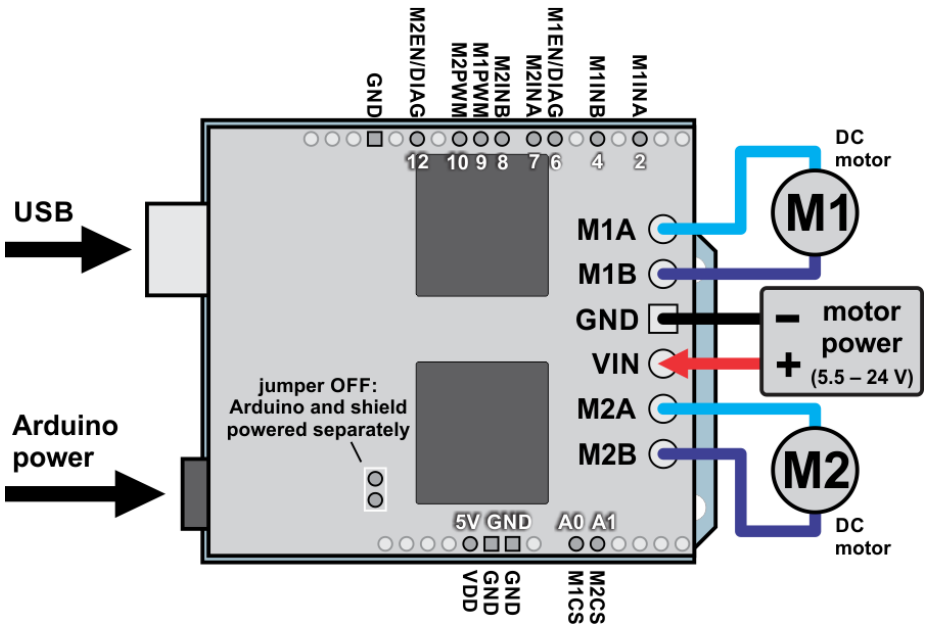
Prethodni pogonski podsustav (kontinuirani servo) je koristio integrirani upravljački sklop (tzv. driver) koji je upravljao i napajao motore. Korišteni upravljački sustav ne može bez posebnog sklopa napajati i upravljati elektromotorima. Da bi se promijenio smjer rotacije istosmjernim elektromotorima, treba zamijeniti polaritet napajanja, a za promjenu brzine okretaja vratila treba mijenjati napon. Za upravljanje motorima korištena su dva VNH5019 H-mosta (Slika 11).



Slika 11 - VNH5019 modul

M2INA, M2INB, M1INA i M1INB služe sa upravljanje smjerom rotacije motora. Jedan od ulaza(INA/INB) treba biti na logičkoj jedinici ako želimo da se motor vrti u tom smjeru. U slučaju da su i jedan i drugi ulaz na visokoj logičkoj vrijednosti, namot motora se stavljen u kratki spoj (kočenje).

Za regulaciju brzine motora koristi se PWM signal. Kod regulacije broja okretaja pod opterećenjem potrebno je regulirati brzinu sa povratnom vezom. Pošto opterećenje motora nije veliko, za pojednostavljenje upravljanja upravljano je motorom bez povratne veze.



Slika 12 - Shema spajanja motora na VNH5019

Korišteni VNH5019 modul montira se izravno na korišten upravljački sklop (Arduino, **Error! Reference source not found.**). Modul koristi napajanje Arduino-a za napajanje logičkog sklopovlja čipa, te koristi vanjski napon od 6V spojen na VIN priključak. Vanjsko napajanje koristi istu nulu kao i ostatak sustava robota. Sklop na pločici ima kondenzatore za stabilizaciju napajanja logičkog sklopovlja. Dodana su još 2 vanjska kondenzatora za eliminaciju visokofrekventnih šumova koji mogu znatno utjecati na cijeli električni sustav robota.

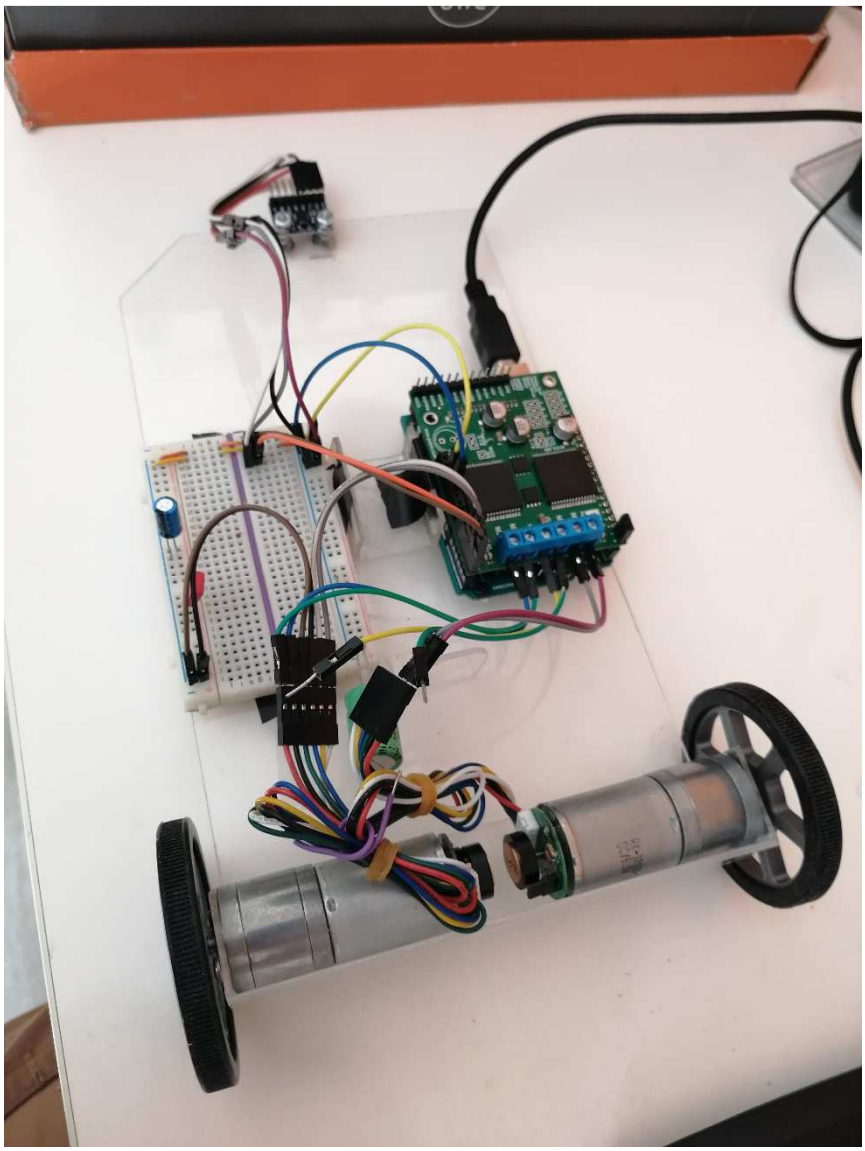
Na robot su također montirani kotači promjera 60mm, dajući maksimalnu brzinu robota od 91.11 cm/s (izraz 4.5).

$$v = 290\ min^{-1} \cdot \pi \cdot 6cm = 5466.37 \frac{cm}{min} = 91.11\ \frac{cm}{s}. \tag{4.5}$$

Brzina okretaja vratila kao funkcija ulaznog signala moduliranog pulsno-širinskom modulacijom (PWM; izraz 4.6)

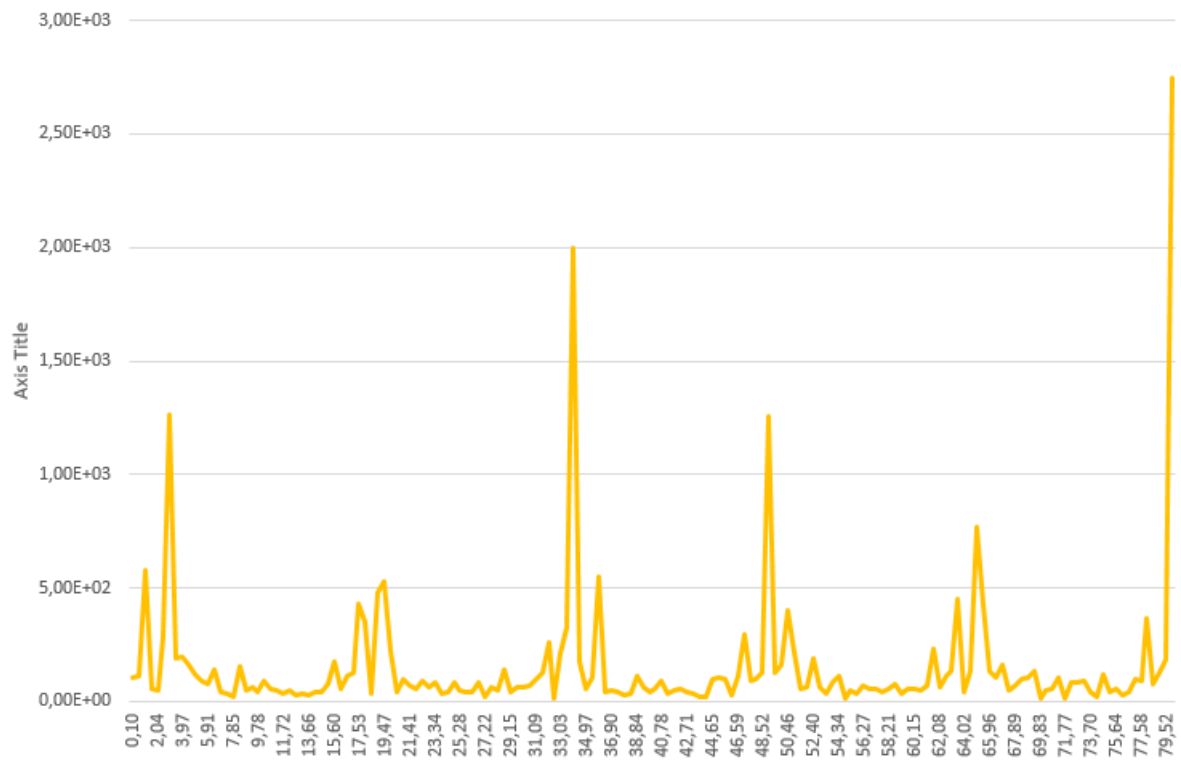
$$rpm[min^{-1}] = 290 \cdot D\ , D \in [0, 1], \tag{4.6}$$

gdje je D radni ciklus PWM signala. Uzevši u obzir idealne uvjete gdje je brzina vratila proporcionalna naponu na elektromotoru.



Slika 13 - Testni robot s DC motorima

PWM signal je generiran kao i za servo. U početku sa frekvencijom od 50Hz, ali vibracije motora kod te frekvencije su ometale rad inercijske jedinice, koja je registrirala vibracije. Vibracije učitane iz inercijske jedinice su analizirane diskretnom fourijerovom transformacijom (Graf 2).



Graf 2 – Analiza vibracija motora

Na grafu je os apscise frekvencija u Hz, a na ordinati magnituda prisutne frekvencije. Vidi se da su dominantne frekvencije harmonici PWM frekvencije. Problem je riješen povećanjem PWM frekvencije na 10kHz. U dokumentaciji motora (RK-370SD) nije naveden induktivitet namota motora, te bez toga podatka nije poznata minimalna primjenjiva frekvencija PWM-a, jer se namot ponaša kao niskopropusni filter.

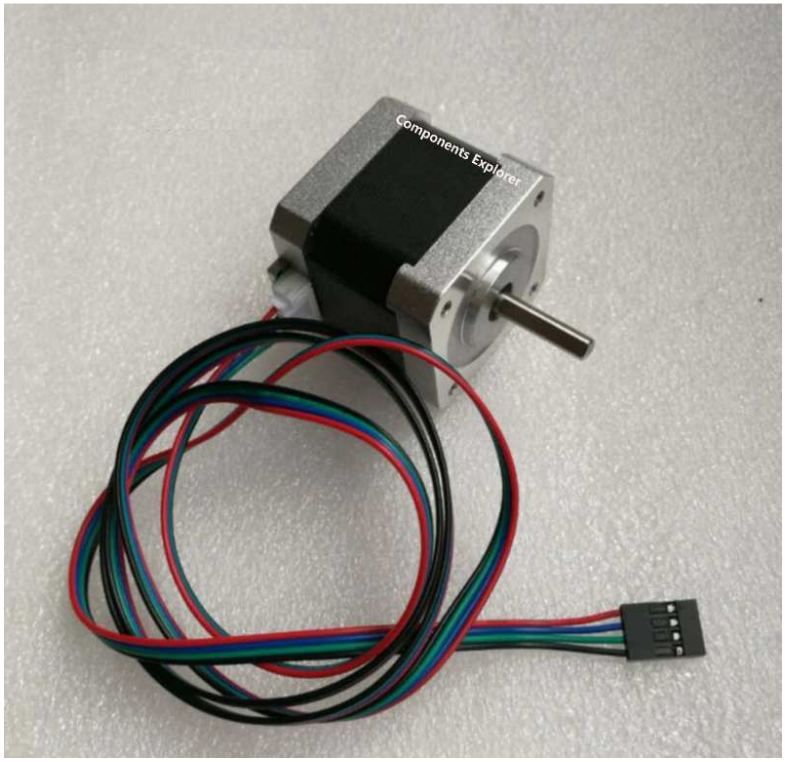
Ovaj način pogona, kao i servo motor imaju mrtvo područje. Mrtvo područje je kompenzirano na isti način kao i kod servo motora, ali DC motoru treba dodatni napon da se pokrene, nakon čega se zbog inercije vrti do poprilično malog napona (kod smanjivanja). To je riješeno tako da je regulator kod paljenja ili promjene smjera motora dao znatno veći napon (ne veći od 6V, ali dovoljan da se pokrene) određeno vrijeme, nakon čega je davao napon u odnosu na željenu brzinu vratila. Taj zadatak je obavljao Timer2, koji je nakon otprilike 100-200 ms smanjio napon na motorima. Taj parametar je bio promjenjiv.

Nakon testiranja pogonskog podsustava s istosmjernim motorima, uočeni su neki nedostaci. Motori su imali dovoljno brz odziv, ali trebalo im je dati viši napon kod pokretanja, što je rezultiralo trzanjem te finalno promašivanjem zadane vrijednosti kuta kad je robot uspravno. Uz taj kardinalan nedostatak motori su i kod 10kHz stvarali vibracije kod samog pokretanja, trzaji kod kretanja su savili konstrukciju robota, i uzrokovali poremećaje vibracije same

konstrukcije, koje su bile dovoljno male frekvencije da prođu kroz niskopropusni filter. Ta frekvencija je prirodna prigušena frekvencija konstrukcije robota, koja bi se smanjila korištenjem kruće konstrukcije (koja bi povećala frekvenciju oscilacije, te bi ta frekvencija prošla teže kroz filtar).

4.1.3 Izvedba sa koračnim motorima

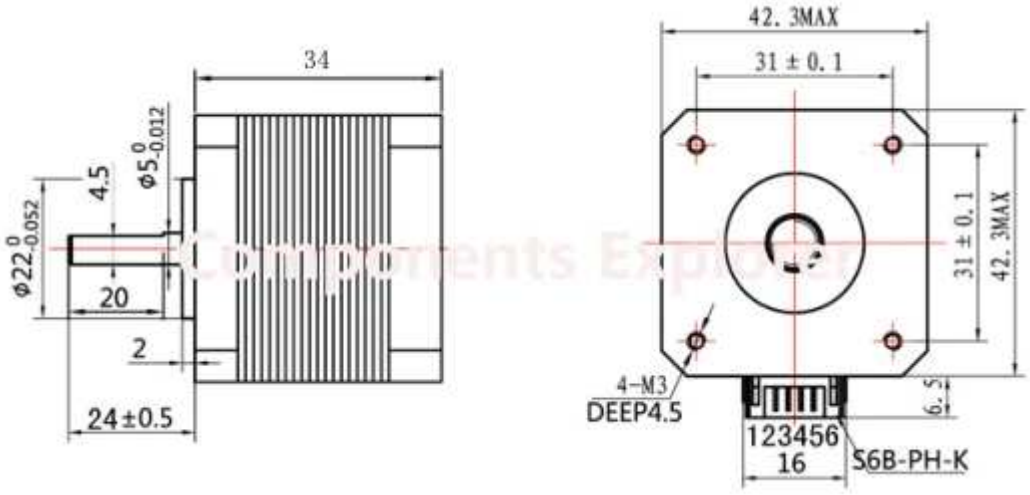
Zadnji pogonski podsustav koji je korišten je koračni motor (engl. *steper motor*). Korišteni motor je unipolarni koračni motor po NEMA17 standardu, koji propisuje oblik prirubnice za montiranje motora. Koračni motor je 17HS1352-P4130 (Slika 14), a dimenzije su prikazane na slici 15.



Slika 14 – Koračni motor – NEMA17

Parametar	Vrijednost	Komentar
Korak	1.8°	200 koraka/revolucija
Nazivni moment	3.2 kg/cm	
Nazivna struja	1.3 A	

Tablica 5 – Podaci o korištenom koračnom motoru



Slika 15 – Dimenzije koračnog motora

Kao i izvedba podsustava s istosmjernim motorom, i ova izvedba zahtjeva zasebni upravljački sklop (tzv. driver). Koračni motori moraju imati posebnu sekvencu uključivanja namota da bi se kontinuirano vrtjeli. Smjer sekvence napona (Slika 16) koji dolazi na namote određuje smjer vrtnje motora. Odabrani koračni motor ima dvije faze te može raditi unipolarno ili bipolarno.

Clockwise Rotation

Index	1a	2a	1b	2b
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1
5	1	0	0	1
6	1	1	0	0
7	0	1	1	0
8	0	0	1	1

Alternate Full-Step Sequence
(Provides more torque)

Two-Phase Sequence

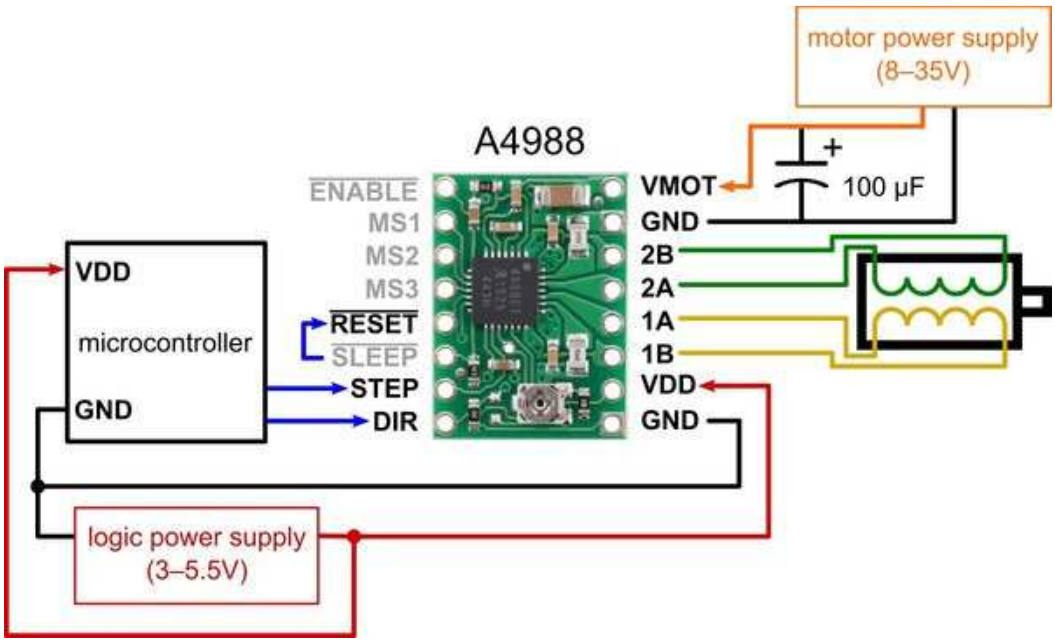
Clockwise Rotation

Index	1a	2a	1b	2b
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1
9	1	0	0	0
10	1	1	0	0
11	0	1	0	0
12	0	1	1	0
13	0	0	1	0
14	0	0	1	1
15	0	0	0	1
16	1	0	0	1

Half-Step Sequence

Slika 16 - Sekvence napona

Modul upravljačkog sklopa motora koji je razmatran i korišten, temelji se na A4988 integriranom sklopu (Slika 17). A4988 zahtjeva samo 2 izvora napajanja (za logiku i motor) te 2 ulazna signala: smjer koraka i korak (DIR i STEP priključci). DIR određuje smjer sekvence, dok rastući brid na STEP priključku pomakne motor za jedan korak.



Slika 17 –Shema spajanja A4988 modula

Ovaj način upravljanja je zahtjevao preinake na generatoru upravljačkog signala. Naime brzina vrtnje koračnog motora je proporcionalna frekvenciji ulaznog signala na STEP priključku. Timer1 je trebalo konfigurirati da radi promjenjivom frekvenijom (Kod 2). Po dokumentaciji za Atmega328p, u tablici 15-5 izabran je CTC način rada (Clear timer on compare) jer se na taj način mogu lagano mjenjati frekvenciju pulseva.

```
TCCR1A = 0;
TCCR1B = (1<<WGM13)|(1<<WGM12)|(1<<CS11)|(1<<CS10);
//prescaler =10
ICR1 = 0x1388; //5000
    OC1A = 100;
    OC1B = 100;
```

Kod 2 – Vidi poglavlje 5.2.2 main.c

Frekvencija se sada može mijenjati mjenjanjem ICR1 vrijednosti. Vrijednost ICR1 registra se može dobiti po formuli 4.7. Brzina se množi sa 2 jer svaki ciklus promjeni logičku razinu na STEP priključku, efektivno djeleći frekvenciju pulseva sa 2.

$$ICR1 = \frac{\frac{F_{CPU}}{prescaler}}{brzina \cdot 2}. \quad (4.7)$$

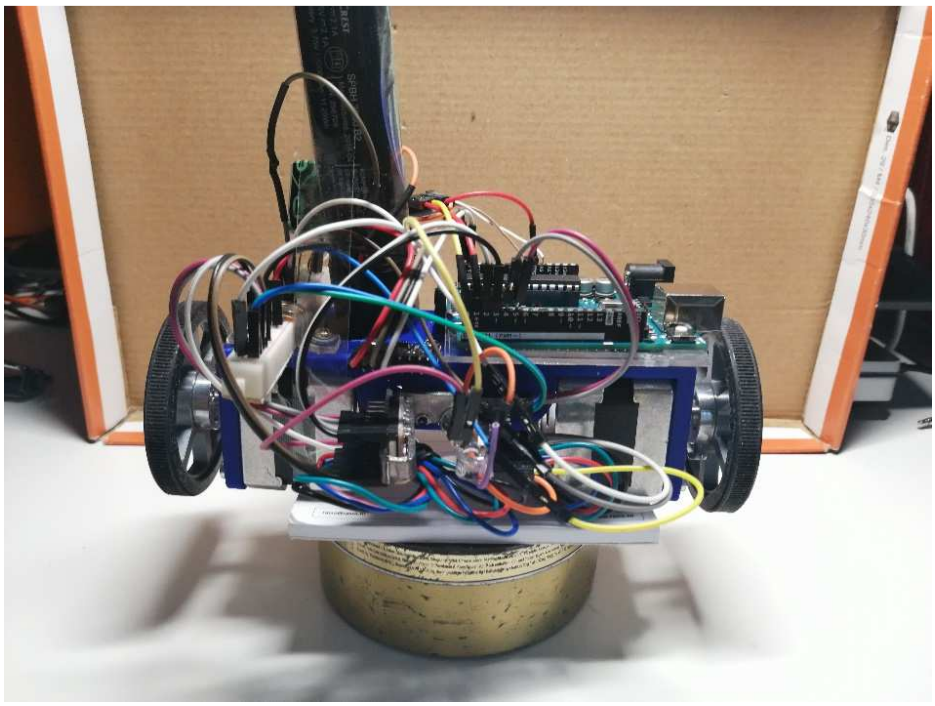
Robot treba skretati, a skrenuti sa koračnim motorima se može samo ako su im brzine različite (diferencijalno skretanje), na neki način se mora promjeniti frekvencija koraka. To je neizvedivo hardverski jer OC1A i OC1B uvijek rade na istoj frekvenciji. Rješenje je da se softverski cjelobrojno djele njihove frekvencije. Frekvencije se mogu djeliti i sa realnim brojem, ali to bi nepotrebno zakompliciralo prekidnu rutinu. Da se ne oduzme previše vremena od digitalne obrade signala i regulacije prioritet je da prekidne rutine budu što kraće i jednostavnije.

U kodu (Kod 3) se vidi da zbog logičkog XOR operatora na portu D priključku 2 frekvencija koraka djeli na dva. Varijabla **stepper_Aerror** je cijeli broj sa kojim se djeli varijabla **stepper_Adifference.** Ako je **stepper_Aerror** djeljiv sa **stepper_Adifference** taj puls po redu će biti generiran. Tako ako je **stepper_Adifference** = 1 tada je svaki puls generiran, ako je **stepper_Adifference** =2 tada je svaki drugi (parni) signal generiran itd. Tako je realizirano cijelobrojno djeljenje frekvencije impulsa.

```
uint8_t stepper_Adifference, stepper_Aerror;
ISR(TIMER1_COMPA_vect){
	if((stepper_Aerror % stepper_Adifference) == 0)
		PORTD ^= (1<<2);
	stepper_Aerror++;
}
```

Kod 3 – Vidi poglavlje 5.2.2 main.c

Ovaj pogonski podsustav je zadovoljio potrebe stabilizacije robota. Lako je upravljiv te pruža jak moment na vratilu pri niskim okretajima, omogućujući montiranje kotača na vratilo motora bez reduktora. Još jedna prednost je odsustvo mrtvog područja. Mana motora su dimenzije, kotač promjera 60mm koji je korišten u izvedbama sa sevo i DC motorima ostavlja samo oko 10mm prostora od podloge (slika 18). Masa nije mana u ovom slučaju robota. Sam motor ima dosta velik moment da se pokrene a i dodatna masa daje moment inercije koji čini sam robot tromijim (sporije reagira na poremećaje, tj. sporije pada) što olakšava upravljanje.



Slika 18 – Prva izvedba robota sa koračnim motorima

4.2 Upravljački podsustav

Upravljački podsustav robota je cjelina koja je odgovorna za upravljanje robotom. Upravljački sustav se sastoji od hardvera (Arduino i bluetooth) i softvera (aplikacija). Arduino je modul na kojem se nalazi ATmega328p, mikroupravljač na kojem se odvijaju svi procesi vezani uz upravljanje robotom, obradom signala i komunikacijom s vanjskim svijetom. Bluetooth je modul sa kojim Arduino komunicira s vanjskim svijetom bežično preko Bluetooth protokola, te omogućava daljinsko upravljanje robotom. Aplikacija je treća komponenta upravljačkog podsustava koja rezidira u hardverskom djelu, te se ne mijenja kroz rad robota. Aplikacija definira ponašanje robota te je ključna u ispravnom radu robota.

4.2.1 Arduino upravljačka pločica

Arduino (Slika 19) je modul koji upravlja cijelim robotom. Arduino je razvojna pločica bazirana na Atmelovom ATmega328 mikroupravljaču koja je baš namijenjena za prototipove jer je otvorenog koda (engl. open source) te postoji velika zajednica inženjera, programera i ostalih struka koji ju koriste i svakodnevno unaprijeđuju.

Slika 19 – Arduino Uno

Parametar	Vrijednost	Komentar
Mikroupravljač	ATmgega328p	PDIP
Performanse	1 MIPS/MHz	Većina instrukcija se izvršava u jednom periodu F_CPU
Radna frekvencija procesora(F_CPU)	16MHz	Kvarcni oscilator
Radna memorija	2kB	SRAM
Programska memorija	32kB	FLASH
Trajna memorija	1024B	EEPROM
Broj izlaza	14	6 izlaza je PWM, 4 8-bitnih izlaza I 2 16-bitnih

Tablica 6 – Svojstva Arduino modula

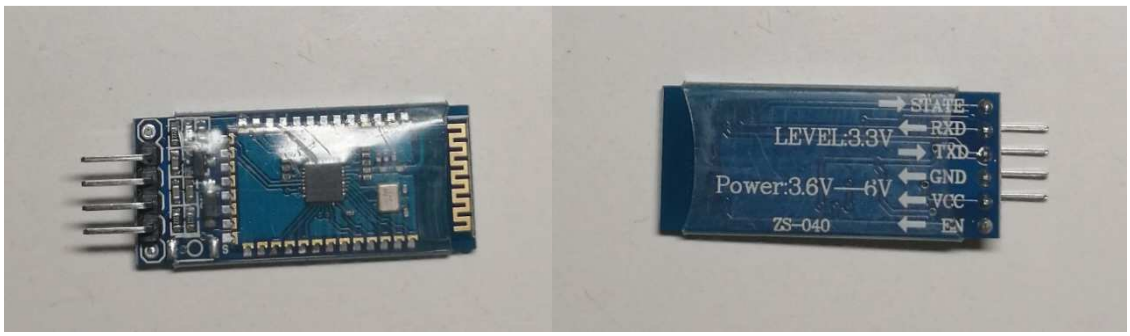
Svi parametri se mogu naći u dokumentaciji za Arduino modul i ATmega328p mikroupravljač, koja je priložena uz ovaj rad.

Arduino se programira preko USB sučelja, što olakšava rad sa mikroupravljačem koji zahtjeva posebni programator koji je integriran na modulu te tipku za resetiranje mikroupravljača. Na modulu se nalaze i dijagnostičke LED diode, dvije su spojene na UART za provjeru serijske komunikacije, jedna koja je spojena na konektor 13 (PORTD5) i može se programirati po želji. Četvrta LED dioda služi za provjeru napajanja modula (svijetli ako je ploča pod radnim naponom).

4.2.2 Bluetooth modul

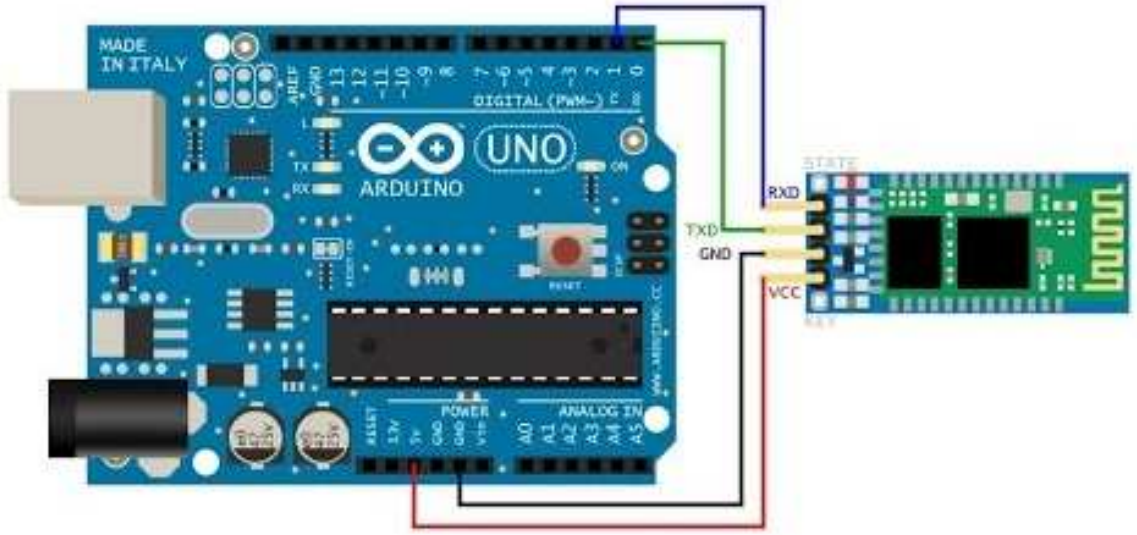
Da robot bude potpuno mobilan, komunikacija se vrši bežično preko Bluetooth protokola. Za komunikaciju Bluetoothom koristi se modul HC-05 (Slika 20).

Modul koristi Bluetooth specifikaciju 2.1 + EDR.



Slika 20 – Bluetooth modul

Bluetooth modul ne treba posebno predstavljanje. Modul sa Arduinom komunicira preko UART-a, što ne zahtjeva ikakve preinake u aplikaciji kod mijenjanja vektora komunikacije (nije potrebno mijenjati aplikaciju da se umjesto Bluetoothom komunicira žičano). Modul se spaja sa Arduino pločicom prema shemi na slici 21.



Slika 21 – Shema spajanja modula na Arduino

4.2.3 Aplikacija za upravljanje

Aplikacija je program pisan u višem programskom jeziku (ovdje C) koji se kompilira u instrukcijski set procesora na kojem će se izvršavati (AVR). Korištena ATmega328p sadrži AVR procesor sa AVR instrukcijskim setom koji je zajednički za mega i tiny serije mikroupravljača, što ne znači da je kompilirana aplikacija univerzalna za sve mikroupravjače bazirane na AVR procesoru (lokacije registara su različite).

Aplikacija je zbog jednostavnosti i efikasnosti pisana u C programskom jeziku. Arduino nudi opciju korištenja C++ programskog jezika za programiranje. Aplikacija je podijeljena u više dadoteka zbog preglednosti koda aplikacije i modularnosti (Tablica 7).

Header datoteka	Source datoteka	Uloga
func.h	func.c	Sadrži funkcije korištene u main.c
i2c_master.h	i2c_master.c	I^2C library, autor: g4lvanix
	main.c	Glavna funkcija, sadrži prekidne rutine
MPU6050raw.h	MPU6050raw.c	MPU6050 library, originalni autor: Davide Gironi
parser.h	parser.c	Sadrži parser i popratne funkcije za interpretaciju serijske komunikacije.
reg.h	reg.c	Sadrži kod za PID regulator
usart.h	usart.c	UART library, autor: jnk0le
usart_config.h		Konfiguracija za UART library

Tablica 7 – Pregled aplikacije

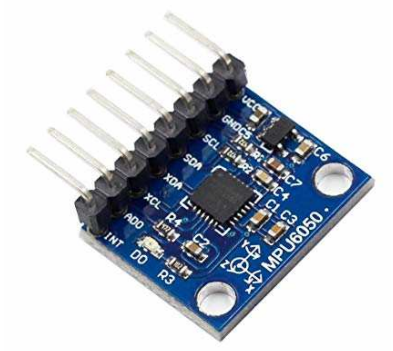
4.3 Senzorski podsustav

Senzorski podsustav čine senzori, tj. osjetila podražaja s okoline robota. Senzori su ključni element kod gotovo svakog uspješnog sustava regulacije i upravljanja. Senzorski podsustav uključuje sama osjetila i pretvarače signala te sklopovlje za obradu signala. Većina obrade signala prisutne u ovom robotu je izvedena digitalno, te odstupaju od klasičnih analognih rješenja. Računala po svojoj digitalnoj prirodi ne mogu obrađivati kontinuirane signale i čiste analogne vrijednosti bez posebnog sklopovlja (postoje i analogna računala koja se rijetko primjenjuju u praksi). Računala mogu obrađivati diskretne vrijednosti koje su fiksne preciznosti zbog kvantizacije te bez problema numerički obrađivati informacije, što je dovoljno.

U ovom poglavlju su napomenute komponente senzorskog podsustava, te algoritme numeričke obrade signala. „Sirovi“ signal (engl. *raw signal*) sa senzora je skoro neupotrebljiv, te će u sljedećim poglavljima biti opisano kako je signal očitan i obrađen da bude koristan.

4.3.1 Inercijska mjerna jedinica - IMU

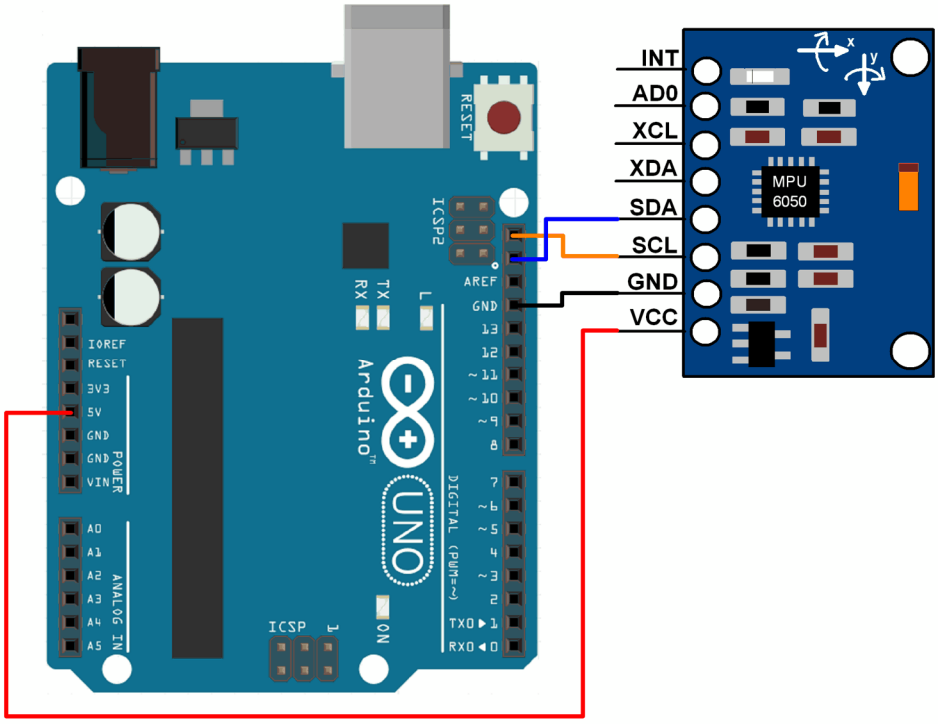
IMU (Eng. Inertial measurement unit) je najvažniji senzor koji je na ovom robotu. On daje informacije o položaju robota u prostoru, što je ovdje prijeko potrebno za vertikalnu stabilizaciju robota (njihala).



Slika 22 - Inercijska jedinica 6050

Korišteni senzor MPU6050 (**Error! Reference source not found.**) je 6-osni inercijski senzor, sa 3 osi rotacije i 3 osi ubrzanja. Za komunikaciju sa upravljačkim sustavom koristi I^2C sabirnicu (poznatu i kao TWI). Također sadrži i DMP (digital signal processor) koji je korišten za obradu signala.

Komunikacija jedinice na I^2C sabirnici se vrši na 400kHz. Konfiguracija sabirnice je implementirana preko posebne biblioteke (library), te neće biti navedena ovdje (priložena je uz upravljački kod). Senzor se spaja sa Arduino pločicom prema shemi na slici 23.



Slika 23 - Shema spajanja senzora na Arduino

IMU ima i posebni izlaz kojim signalizira upravljačkoj jedinici da su mjerenja izvršena i da su podaci spremni za čitanje (INT priključak), no nije korišten jer nije bilo potrebe za bržim mjerenjem.

IMU se prije upotrebe treba inicijalizirati da bi se iz njega dobila točna očitanja. Skoro svi registri inercijske jedinice su postavljeni na vrijednost 0, što olakšava konfiguraciju.

Inicijalizacija počinje postavljanjem registra 0x6B (PWR_MGMT_1) na vrijednost 0. Taj registar upravlja potrošnjom energije modula. Modul je u početku rada ugašen, te postavljanjem PWR_MGMT_1 registra u 0 resetiramo SLEEP bit, te postavimo brzinu rada modula na 8MHz (4.28 Register 107 – Power Management 1, MPU60x0 register map).

```
//#define MPU6050_ADDR 0xD0
i2c_start(MPU6050_ADDR);
i2c_write(0x6B);
i2c_write(0);
i2c_stop();
```

Kod 4 – Vidi poglavlje 5.2.5 MPU6050raw.c

Nakon inicijalizacije (Kod 4) modul je spreman za rad. No za bolje rezultate mjerenja korišten je niskopropusni filter (Kod 5) koji je dio DMP-a prisutnog na modulu.

```
uint8_t mpu6050_setDLPF(uint8_t mode){
	uint8_t reg, regWas;
	i2c_start(MPU6050_ADDR | I2C_WRITE);
	i2c_write(0x1A); //ACX register
	i2c_start(MPU6050_ADDR | I2C_READ);
	_delay_us(10);
	reg = i2c_read_nack();
	regWas = reg;
	reg |= mode;
	i2c_start(MPU6050_ADDR | I2C_WRITE);
	i2c_write(0x1A); //ACX register
	i2c_write(reg); //ACX register
	return regWas;
}
```

Kod 5 – Vidi poglavlje 5.2.5 MPU6050raw.c

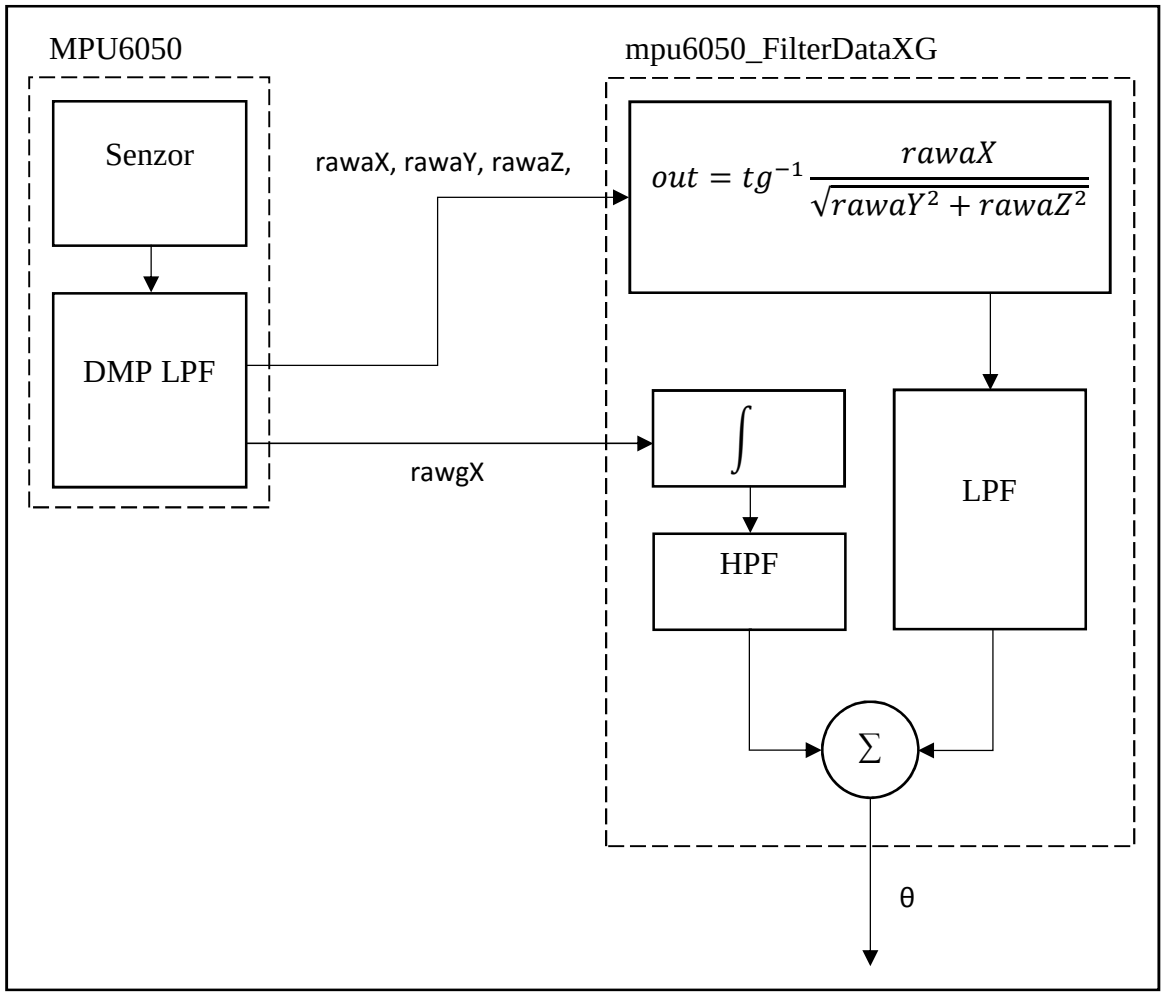
Mode varijablu se odabire preko **DLPF_CFG** parametra (Tablica 8). Izabrano je 10Hz, dovoljno da smanji šum vibracija bez da previše uspori frekvenciju uzorkovanja.

DLPF_CFG	Accelerometer (F_s = 1kHz)		Gyroscope		
	Bandwidth (Hz)	Delay (ms)	Bandwidth (Hz)	Delay (ms)	Fs (kHz)
0	260	0	256	0.98	8
1	184	2.0	188	1.9	1
2	94	3.0	98	2.8	1
3	44	4.9	42	4.8	1
4	21	8.5	20	8.3	1
5	10	13.8	10	13.4	1
6	5	19.0	5	18.6	1
7	RESERVED		RESERVED		8

Tablica 8 – Konfiguracija DMP LPF pomoću DLPF_CFG parametra

4.3.2 Filtriranje podataka sa senzora

Signal dobiven sa inercijske jedinice je nesavršen i s prisutnim šumovima koje DMP LPF nije eliminirao. U ovom poglavlju će biti objašnjeno filtriranje i obrada dobivenog signala (podatka) u koristan oblik.



Slika 24 - Proces obrade signala inercijske jedinice

Izlazni signal cijelog sustava filtera je kut θ (nagib robota, Slika 24). Iz učitanih sirovih vrijednosti ubrzanja (rawaX, rawaY, rawaZ) se dobiva kut vektora ubrzanja sile teže na X osi. Druga sirova vrijednost kutnog ubrzanja na X osi se prvo integrira, pa integral (kut na X osi) zajedno sa kutom vektora ubrzanja sile teže ide u komplementarni filter. Komplementarni filter se sastoji od niskopropusnog i visokopropusnog filtera sa istom korner frekvencijom, čiji se izlazi zbrajaju. Izlaz komplementarnog filtera je signal kuta nagiba robota θ.

Drugi kompliciraniji izbor za filter je Kalman-ov filter, koji je znatno kompliciraniji od komplementarnog filtera koji je sasvim dovoljan po kriteriju kompliciranost/kvaliteta. U početku su računata sva tri kuta koji opisuju orijentaciju robota, ali kako je projekt napredovao pojednostavljen je da izračunava samo potreban kut.

Postoji mogućnost linearizacije funkcije mpu6050_FilterDataXG (Kod 6) koja koristi trigonometrijske operatore (atg), ali nije implementirana zahtijevajući točan nagib robota. Linearizirana funkcija bi se mogla znatno brže izračunavati.

Filtriranje i obrada dobivenih podataka o orijentaciji (unfilteredData) se vrši na mikroupravljaču (ATmega328p) Arduino modula. Filtriranje i obrada se vrši numerički (Kod 6).

```
void mpu6050_FilterDataXG(int16_t *unfilteredData, float *x_angle){

        *x_angle = atan((unfilteredData[0]/16384.0) /
        sqrt(pow(unfilteredData[1]/16384.0,2) +
        pow(unfilteredData[2]/16384.0,2)))*RAD_TO_DEG;

        *x_angle = 0.98 * (*x_angle + (unfilteredData[4]/131.0) *
        global_dt) + 0.92 * *x_angle;

}
```

Kod 6 – Vidi poglavlje 5.2.5 MPU6050raw.c

Nefiltrirani podaci su skalirani po osjetljivosti senzora koji su navedeni u MPU6050 register map dokumentu (4.4 Register 27 – Gyroscope Configuration; 4.5 Register 28 – Accelerometer Configuration) koji se nalazi u prilogu. Nakon skaliranja sirove vrijednosti se dalje obrađuju.

```
*x_angle = atan((unfilteredData[0]/16384.0) /
sqrt(pow(unfilteredData[1]/16384.0,2) +
pow(unfilteredData[2]/16384.0,2)))*RAD_TO_DEG;
```

Kod 7 – Vidi poglavlje 5.2.5 MPU6050raw.c

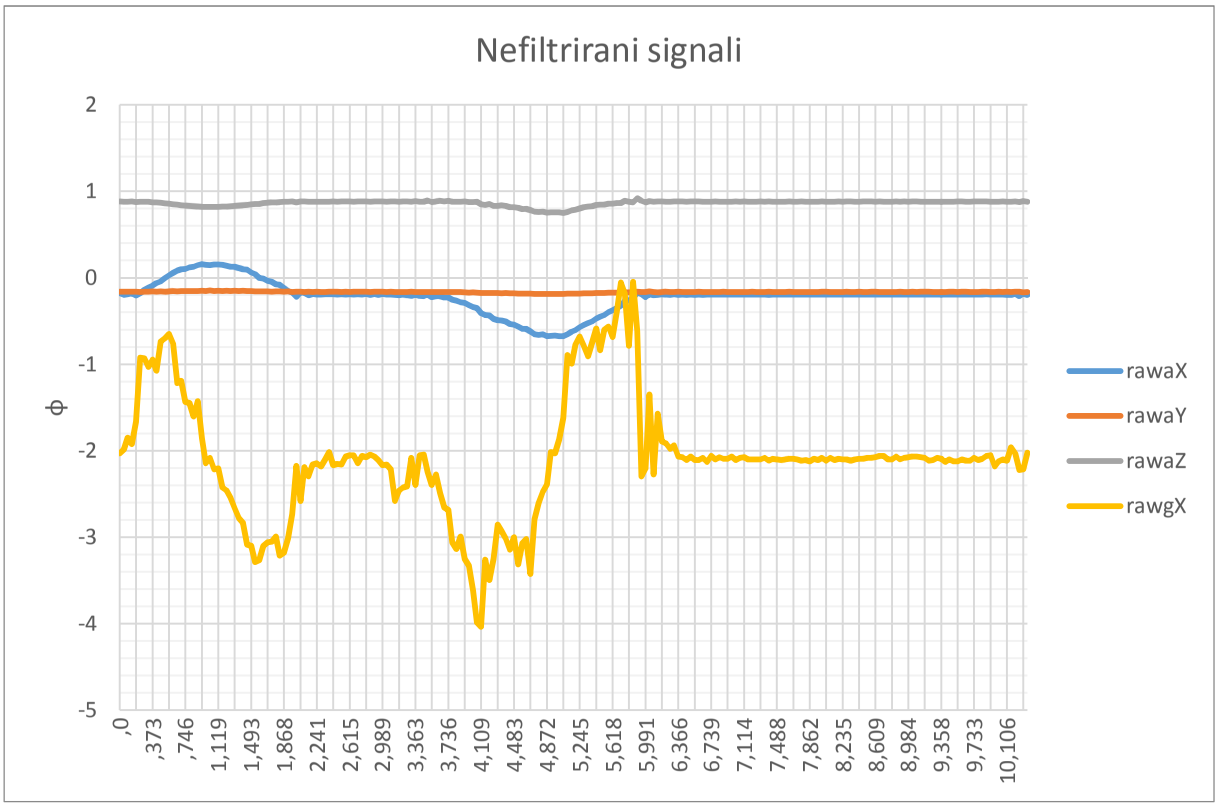
Kod prikazan ***iznad*** (Kod 7) izračunava kut ubrzanja sile teže oko X osi.

```
*x_angle = 0.98 * (*x_angle + (unfilteredData[4]/131.0) *global_dt) +
0.92 * *x_angle;
```

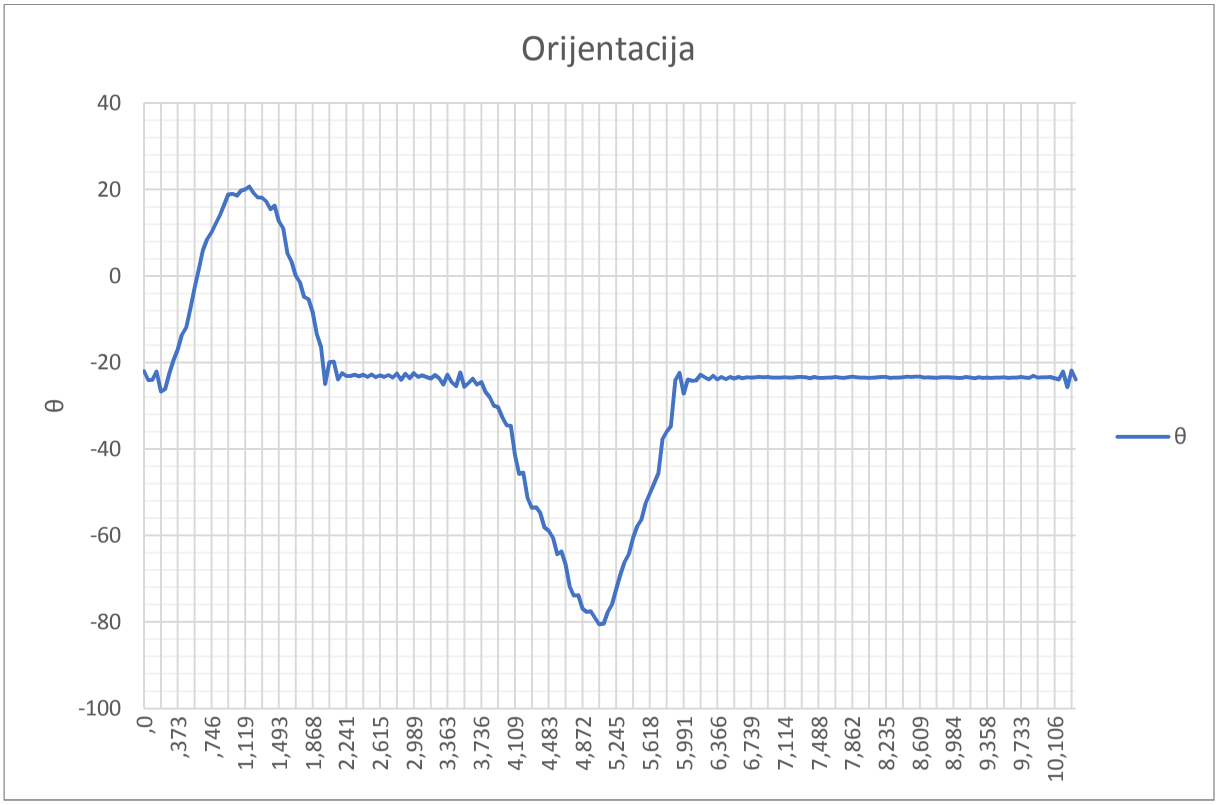
Kod 8 – Vidi poglavlje 5.2.5 MPU6050raw.c

Zadnja komponenta filtritanja je komplementarni filtar (Kod 8). Nakon komplementarnog filtera dobiveni kut se sprema u varijablu tipa float zadanu kao argument funkcije mpu6050_FilterDataXG (Kod 6).

Na grafu 3 su sirovi (prefiks „raw“) podatci sa inercijske jedinice, a na grafu 4 filtriran podatak o orijentaciji θ.



Graf 3 – Nefiltrirani signali sa inercijske jedinice



Graf 4 – Filtrirani podatak o kutu θ.

5 Realizacija sustava robota

Nakon što su svi sustavi robota analizirani i pregledani, vrijeme je da se robot realizira. Ovaj dio je vrhunac dizajna robota jer se svi parametri dizajna i zahtjevi projekta manifestiraju u fizičku izvedbu robota.

Kako je projekt napredovao počevši od modeliranja, razvijao se i prototip robota na kojem su isprobavani sustavi upravljanja te sklopovlje. Taj proces se zove razvoj projekta (eng. *development*).

Ovaj projekt je kroz razvoj doživio mnogo preinaka koje su utjecale na zahtjeve projekta, posebice na korištene komponente i sklopovlje a zatim i na sam upravljački sklop koji se prilagođavao elektronici robota. Pogonski podsustav je komponenta koja je doživjela najviše preinaka, jer je tražen sustav koji će zadovoljavat postavljene zahtjeve projekta (prespori pogonski podsustavi nisu mogli držati robot uspravno, a to je glavni zahtjev).

Program koji čini upravljački podsustav se tijekom razvoja širio u svojim mogućnostima, te od jednostavnog programa za ispitivanje ispravnosti pogonskog podsustava i komunikacijom sa ostalim modulima pretvorio u kompleksnu cjelinu koja efektivno drži robot uspravnim sa serijskim upravljanjem.

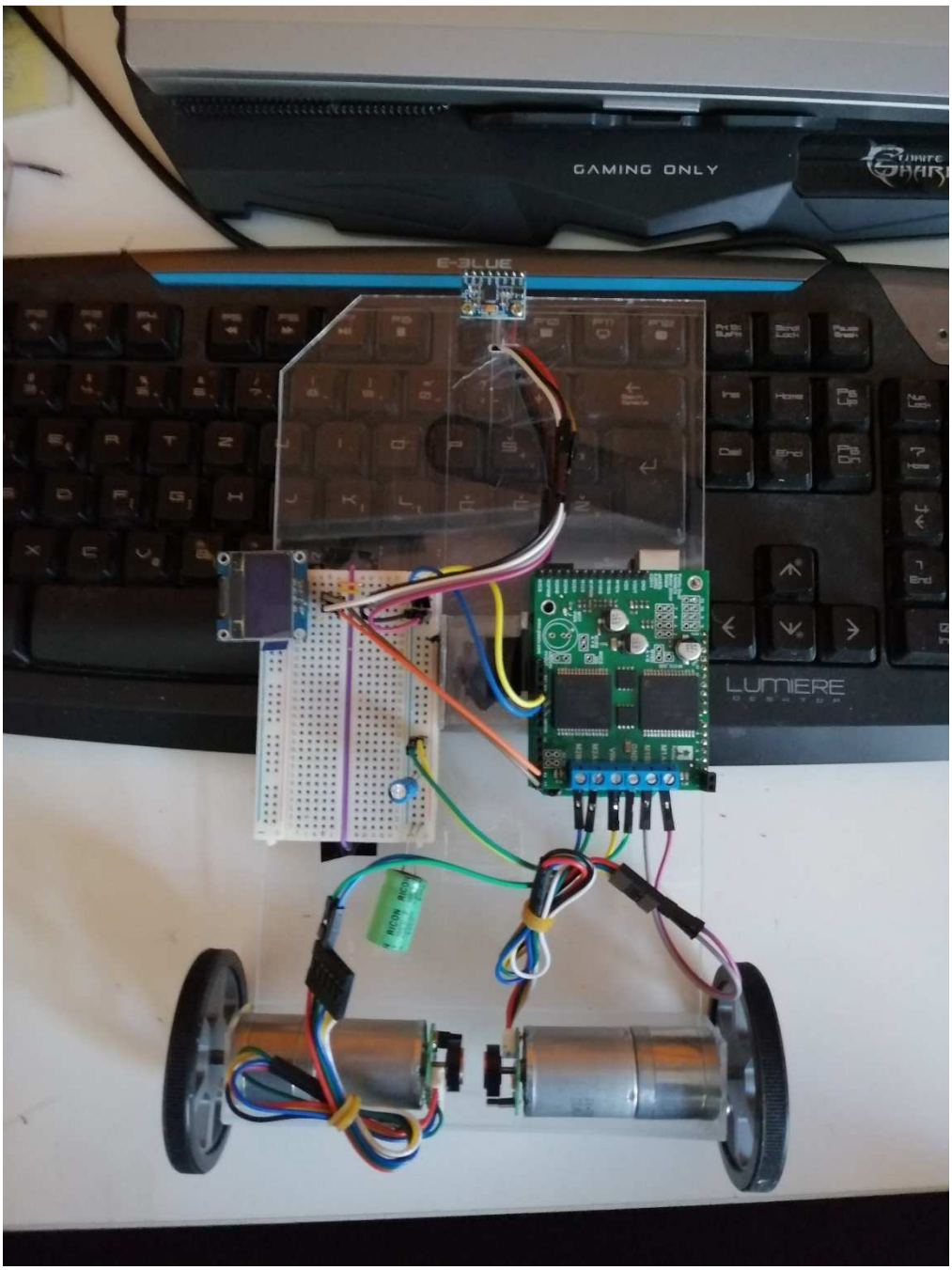
U ovom poglavlju će se opisati mehanička konstrukcija i kod programa robota.

5.1 Mehanička konstrukcija robota

Jedan od glavnih dijelova robota je mehanička konstrukcija. Mehanička konstrukcija je cjelina odgovorna za mehanički integritet robota i sastoji se od fizičkih dijelova. Kod ovog robota mehanička konstrukcija je šasija i pogonski podsustav. Pogonski podsustav je također dio mehaničke konstrukcije jer su na njega izravno pričvršćeni kotači.

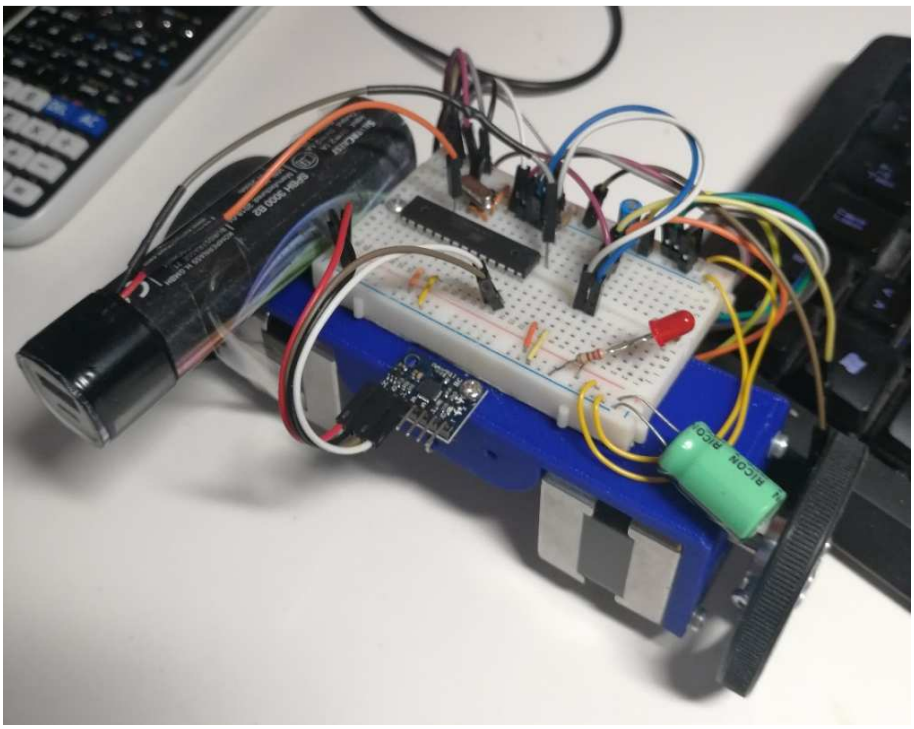
Kroz evoluciju projekta mehanička konstrukcija se bitno mijenjala. Kako su isprobavani razni pogonski moduli pa čak i upravljački moduli, konstrukcija koja ih je držala u cjelini se bitno mijenjala.

Prva verzija robota je bila sa servo motorima. Konstrukcija je bila izrađena od akrilnog stakla (plexiglass). Staklo je rezano skalpelom, lomljeno i lijepljeno superljepilom (cijanoakrilatom). Konstrukcija je bila dovoljno robusna, ali je poslije zamijenjena većom šasijom. Moduli na ovoj konstrukciji su bili učvršćeni izolir-trakom radi lakšeg i efektivnijeg prototipiranja.



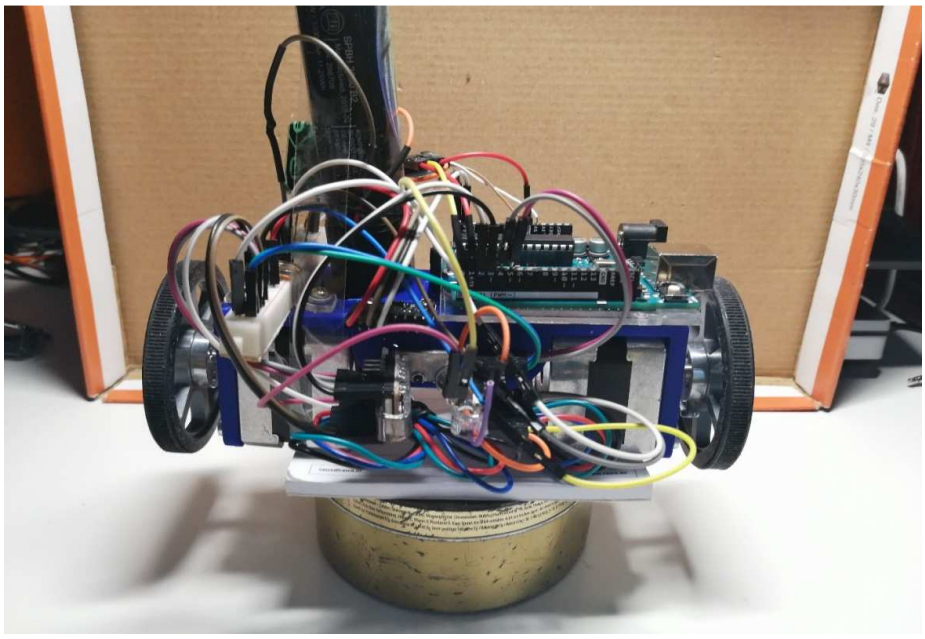
Slika 25 – Izvedba šasije robota sa DC motorima

Nakon što je utvrđeno da su korišteni servo motori koji su korišteni prespori, zamijenjeni su sa istosmjernim motorima. Novi pogonski podsustav je zahtijevao više prostora te je izrađena nova mehaničku konstrukcija (Slika 25). Nova konstrukcija je također izrađena od akrilnog stakla, ali za razliku od prethodne konstrukcije ova je imala termički obrađene elemente. Nosač motora i baterije je bio termoformiran u cilju postizanja veće čvrstoće nego kod lijepljenja superlijepilom. Jedan vidljiv detalj na slici 25 je prisutnost malog OLED ekrana koji je služio za prikaz varijabli prije je implementirana serijska komunikacija.



Slika 26 – Testiranje sustava robota

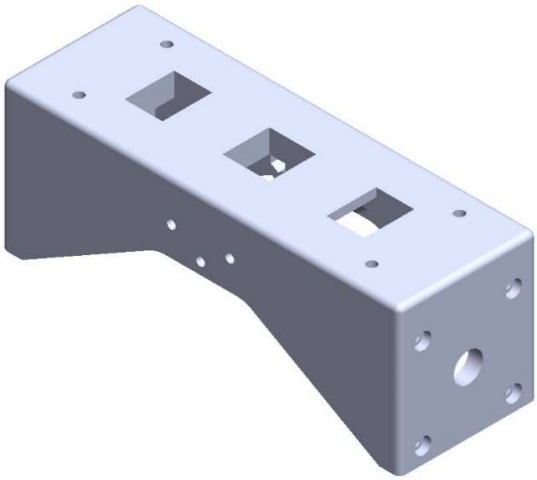
Nakon testiranja robota utvrđeno je da i istosmjerni pogonski podsustav nije dovoljan za stabilizaciju robota te je bilo nužno opet promijeniti mehaničku konstrukciju. Nova mehanička konstrukcija (Slika 26) je bila izrađena iz PLA plastike na 3D printeru FFF (fused filament fabrication) tehnologijom. Na konstrukciji je vidljiva Li-ion baterija na lijevoj strani, te Atmega328p na eksperimentalnoj pločici kao privremena zamjena za Arduino modul.



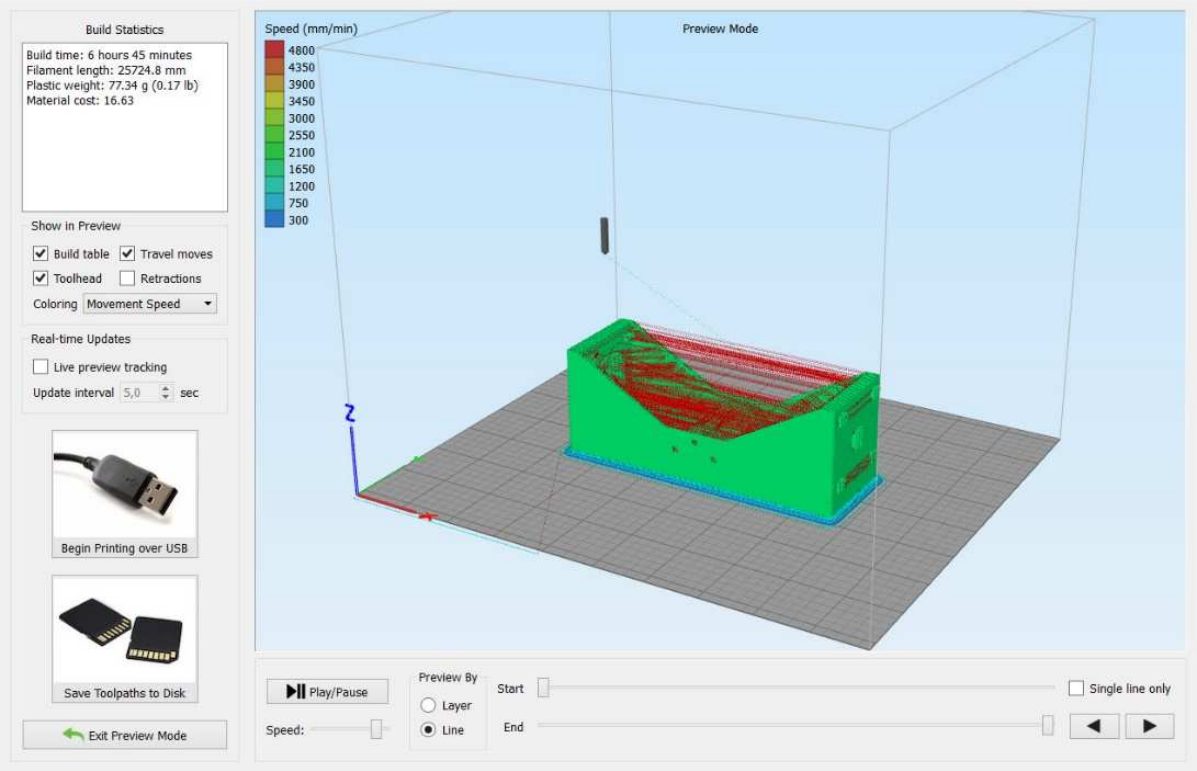
Slika 27 - Testiranje sustava robota

Kako je projekt napredovao i toj su mehaničkoj konstrukciji (Slika 27) dodani elementi od akrilnog stakla (nosači Arduino modula te A4988 modula za pogon koračnih motora). Kasnije su i ti elementi zamijenjeni 3D printanim elementima (Slika 30).

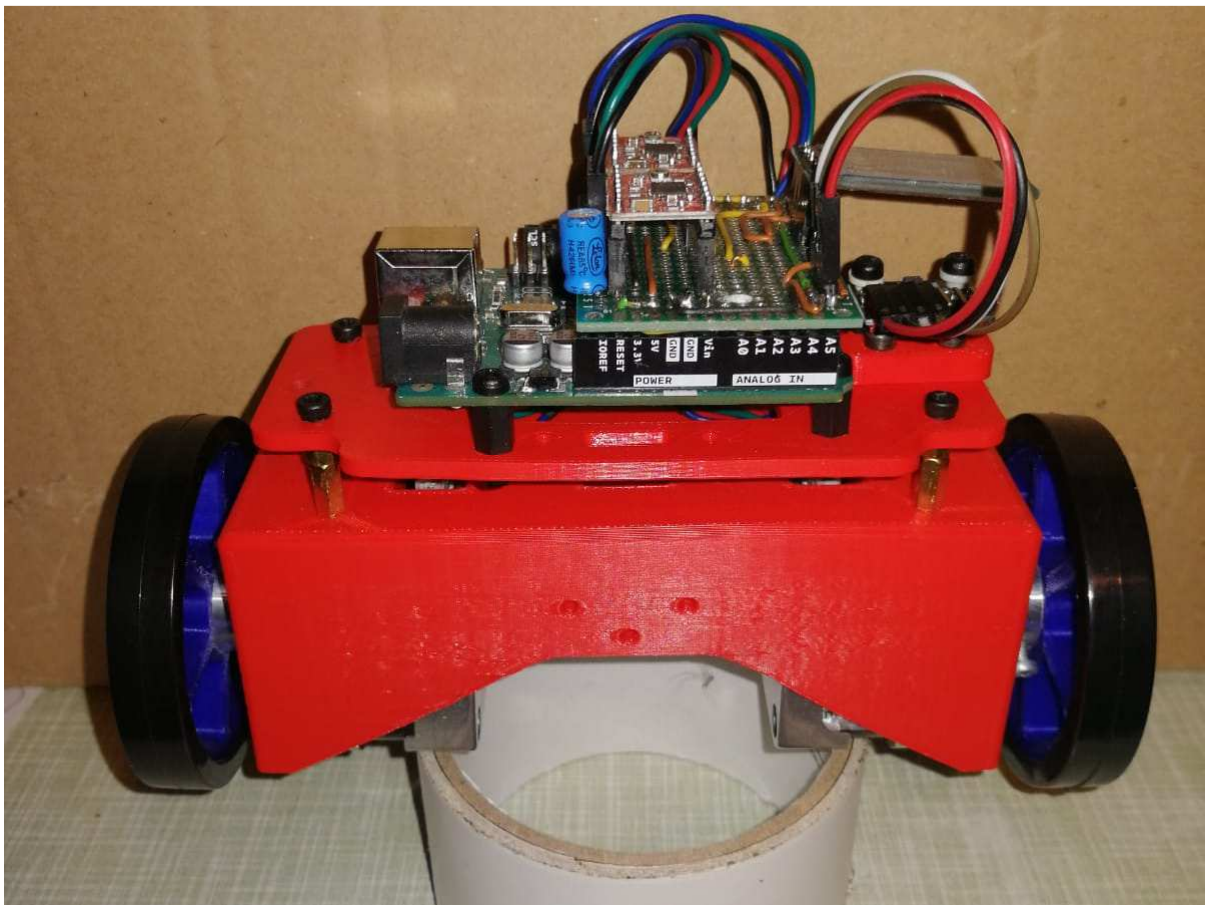
Mehanička konstrukcija šasije (Slika 28 – CAD model) je modelirana u SolidWorks programu. Ovo je glavni dio mehaničke konstrukcije te su na nju vijcima učvršćeni koračni motori i noseći element upravljačke jedinice. Između šasije i nosećeg elementa upravljačke jedinice je mjesto za bateriju koja je osigurana čvrstim dosjedom.



Slika 28 – CAD model šasije robota



Slika 29 – Sučelje 3D printera sa CAD modelom konstrukcije.



Slika 30 – Prototip mobilnog inverznog njihala

5.2 Program (kod) za upravljanje

U ovom poglavlju će bit prikazan kod za upravljanje robota. Pošto je kod poprilično velik neće biti u cijelosti prikazan ovdje te su opisani najbitniji dijelovi. Najbitniji dijelovi su komentirani, a oni očiti nisu.

Sveukupan kod se sastoji od više modula koji sadrže izvornu (source) datoteku i header datoteku. Header datoteka je navedena radi deklaracije funkcija. U poglavljima će biti objašnjene samo izvorne datoteke (source datoteke sa nastavkom .c) ako nije napomenuto suprotno.

U poglavlju 4.2.3 je napomenuta struktura programa (Tablica 7) prema kojoj je kod organiziran. Ovdje neće biti prikazane biblioteke (library) za UART (serijski prijenos) i i2c sabirnicu, no bit će prikazana biblioteka za MPU6050 (autor: Davide Gironi) koja je skraćena na osnovne funkcije. Neki dijelovi koda su referencirani u 4. poglavlju.

Priloženi program može sadržavati par promjena, ali se izvršava na isti način

5.2.1 func.c i func. h

Ovaj modul sadrži inicijalizaciju periferije mikroupravljača (`init` funkcija) te funkciju za zadavanje brzine okretaja koračnog motora (`stepper_setSpeed`).

```
#include <math.h>

uint8_t stepper_Adifference, stepper_Bdifference, stepper_Aerror, stepper_Berror;
float stepper_icrTmp, stepper_icrWas;

uint8_t init(void){
	TCCR2B = 6; //16M/256, 62500 Hz.
	TIMSK2 = 1; //ovf int
	DDRD  |= (1<<2)|(1<<3)|(1<<4)|(1<<5);

	//PIN CHANGE INT
	DDRC &= ~(0x0F);
	return 0;
}

void stepper_setSpeed(float speed, int8_t difference){
	stepper_icrTmp = (2000000)/(2*abs(speed)*145);
	ICR1 = stepper_icrTmp;
	//OCR1A = stepper_icrTmp/2;
	//OCR1B = stepper_icrTmp/2;
	OCR1A = 20;
	OCR1B = 20;
	//if(stepper_icrTmp > 12500) //20 speed
	TCNT1 = 0;
	if (speed > 0){
		PORTD |=  (1<<3);
		PORTD &= ~(1<<5);
	}else{
		PORTD &= ~(1<<3);
		PORTD |=  (1<<5);
	}

	if(difference > 0){
		stepper_Adifference = abs(difference);
		stepper_Bdifference = 1;
	}else if(difference == 0){
		stepper_Adifference = 1;
		stepper_Bdifference = 1;
	}else{
		stepper_Adifference = 1;
		stepper_Bdifference = abs(difference);
	}
	//stepper_icrWas = stepper_icrTmp;

}
```

Header datoteka:

```
#ifndef _FUNC_H_
#define _FUNC_H_

uint8_t init(void);

uint8_t timer1_start(uint16_t prescaler, uint8_t pwm_mode, uint8_t input_capture,
uint8_t enableINTovf);

void stepper_setSpeed(float speed, int8_t difference);

#include "func.c"
#endif
```

5.2.2 main.c

Ovo je glavna datoteka koda robota. Ovo je prvi dio programa koji se pokreće (`main` funkcija). Uključivanje ostalih modula koda je izvršeno nakon inicijalizacije varijabli jer moduli ovise o njima. Ovaj modul se sastoji od:

1. Definicija `F_CPU` i uključivanje datoteka za rad sa mikroupravljačem.
2. Inicijalizacije varijabli.
3. Uključivanje ostalih modula koda (`#include`)
4. `main` funkcija.
5. `me_process` funkcija.
6. Prekidne rutine (`ISR`)

Kod sadrži neke varijable koje se ne koriste, te ih je compiler ignorirao.

Definicija *F_CPU* i uključivanje datoteka za rad sa mikroupravljačem, inicijalizacija varijabli i `#include` direktive:

```
#if (F_CPU != 16000000UL)
#undef F_CPU
#define  F_CPU 16000000UL //frekvencija rada CPU-a 16 MHz
#endif

#include <avr/io.h>
#include <util/delay.h>
#include <string.h>
#include <avr/interrupt.h>

#define ME_BUFFER_SIZE 16 //Koristi se u više djelova koda

char me_buffer[ME_BUFFER_SIZE];
char me_displayString[16];
float me_time = 0;

int16_t me_mpu6050_rawdata[6];
float me_mpu6050_fdataXG;
float me_mpu6050_fdata[6];
float global_dt;
float global_MovCommand = 0;
uint8_t global_TrnCommand = 0, global_commandTerminate;

uint8_t global_TCNTWas, global_overflows, me_updateStepperSpeed = 0;

//Include dio
#include "i2c_master.h"
#include "i2c_master.c"
#include "usart.h"
#include "usart.c"

#include "reg.h"
#include "func.h"
#include "MPU6050raw.h"
#include "encoder.c"
#include "parser.h"

void me_process(void);
```

`main` funkcija i `me_process`:

```
int main(void)
{
	uart_init(BAUD_CALC(9600)); // 8n1 serijski uart prijenos
	uart_puts("hello from usart 0\r\n");

	i2c_init(); // inicijalizacija i2c sučelja
	init();
	mpu6050_init();

	mpu6050_setDLPF(6);
	_delay_ms(10); //pričekaj dok se IMU stabilizira

	//Inicijalizacija Timer1 modula na mikroupravljaču:
	TCCR1A = 0;
	TCCR1B = (1<<WGM13)|(1<<WGM12)|(1<<CS11)|(1<<CS10);

	//Izvrši komandu ML, učitaj PID parametre iz EEPROM memorije
	parser_parseString("ML", 2, NULL);

	delay_ms(115);
	sei(); //Uključi prekide (interrupts)

    while (1) //Beskonačna petlja
    {
		if(uart0_AvailableBytes() > 0){ // provjerava ima li naredbi
			_delay_ms(100); // pričeka dok dođe cijela naredba
			uart_gets(me_buffer, ME_BUFFER_SIZE); // učitaj naredbu
			parser_parseString(me_buffer, ME_BUFFER_SIZE, NULL); // izvrši ju
		}
		_delay_ms(50);
	}
}

void me_process(void){
	//Učitavanje vrijednosti i filter:
	mpu6050_readVals(MPU6050_ADDR, me_mpu6050_rawdata);
	mpu6050_FilterDataXG(me_mpu6050_rawdata, &me_mpu6050_fdataXG);
	//PID regulator:
	regulator_pid(me_mpu6050_fdataXG);
}
```

Prekidne rutine:

```
ISR(TIMER2_OVF_vect){
	global_overflows++;
	global_commandTerminate++;
	me_updateStepperSpeed++;
	if(me_updateStepperSpeed > 8){
		stepper_setSpeed(pid_output + global_MovCommand, global_TrnCommand);
		me_updateStepperSpeed = 0;
		if(pid_errorCumulative != pid_errorCumulative)
			pid_errorCumulative = 0; // za djeljenje frekvencije izvršavanja
	}
	if(global_overflows > PID_OVERFLOW_SETPOINT){
		global_dt = 0.011111111;
	//Fiksni dt jer se regulator pokreće u stalnim intervalima frekvencijom od 90 Hz
		me_process();
		global_overflows = 0;
	}
	if(global_commandTerminate > 12){
		global_MovCommand = 0;
		global_TrnCommand = 0;
		global_commandTerminate = 0;
	}
}

//Potrebno za diferencijalno skretanje:
ISR(TIMER1_COMPA_vect){
	if((stepper_Aerror%stepper_Adifference) == 0)
		PORTD ^= (1<<2);
	stepper_Aerror++;
}

ISR(TIMER1_COMPB_vect){
	if((stepper_Berror%stepper_Bdifference) == 0)
		PORTD ^= (1<<4);
	stepper_Berror++;
}
```

5.2.3 parser.c i parser.h

Ovo je modul za izvršavanje naredbi. Taj modul sadrži funkciju koja interpretira naredbe te ih izvršava. Naredbe se sastoje od najviše 2 slova te mogu primati argumente koji se pišu uz slovo bez razmaka (npr. PP1.234 će postaviti KP parametar za PID regulator na vrijednost 1.234).

Višeznakovne konstante se ne koriste zbog nedefiniranog ponašanja, ali su ovdje korištene jer su testirane i rade ispravno. Ne jamči se da će se nekom drugom mikroupravljaču ili compileru ovaj modul isto ponašati, pa je potreban oprez.

Korišteni compiler je avr-gcc (gcc - GNU Compiler Collection).

```
#include <stdlib.h>
float parser_parseString_value;

//Deklaracija PID parametara u EEPROM memoriji.
float EEMEM parser_pidKP, parser_pidKI, parser_pidKD, parser_pidSetpoint;

uint8_t parser_parseString(char *string, uint8_t len, uint8_t *output_data){
	uint16_t parser_parseString_Letter;
	if((*(string + 1) == 0x0A) || (*(string + 1) == 0x0D)) *(string + 1) = 0;
	if((*(string + 1)) != 0) parser_parseString_value = strtod(string + 2, NULL);
	parser_parseString_Letter = (*(string + 1)<<8) + *string;

	switch(parser_parseString_Letter){
		case ParserCommand_Forward:
			global_MovCommand = GLOBAL_MOVCOMMAND_MAGNITUDE;
		break;

		//Neće biti prikazane sve naredbe, ali imaju isti format kao ova gore
		//tj. case naredba: break;

	}
parser_bufferFlush(string, len);
return 0;
}
```

Header datoteka:

```
#include <string.h>
#include <stdlib.h>
#include <avr/eeprom.h>

#define PARSER_BUFFERSIZE 16
#define PARSER_MAXPLACESFORVALUES 6
#define GLOBAL_MOVCOMMAND_MAGNITUDE 2.5
#define GLOBAL_TURNCOMMAND_MAGNITUDE 3

/******************** PARSER COMMAND LIST ***************/
#define ParserCommand_Forward                          'F'
#define ParserCommand_Back                             'B'
//Nisu prikazane sve definicije naredbi
#define ParserCommand_SaveParamsPid                    'PS'
//ParserCommand_SaveParamsPid naredba zapravo glasi SP
//Sve naredbe sa 2 slova su u kodu napisane naopako da se
//ispravno interpretiraju
/******************** PARSER COMMAND LIST END ***************/

uint8_t parser_parseString(char *string, uint8_t len, uint8_t *output_data);
void parser_bufferFlush(char *buffer, uint8_t len);

#include "parser.c"
#endif
```

5.2.4 reg.c i reg.h

Ovaj modul sadrži implementaciju PID regulatora. Ovdje ima puno varijabli koje se više ne koriste.

```
uint16_t pid_deadzoneEliminator = 0;
uint8_t pid_deadzone_startForward, pid_deadzone_startBackward,
pid_deadzone_startedForward, pid_deadzone_startedBackward, pid_deadzoneTimer;
uint8_t pid_overflows = 0, pid_overflowsTCNTWas = 0;
int16_t pid_motorAoffset = 0, pid_motorBoffset = 0;
float pid_azimuth = 0;
float pid_error, pid_errorWas, pid_setpoint = 0, pid_errorCumulative = 0;
float pid_output;
float pid_KP, pid_KD, pid_KI;// i pid_dt kad nije bila konstanta
float regulator_pi_error, regulator_pi_errorCumulative;

void regulator_pid(float input){
//Ovaj dio je za računanje dt-a. Trenutno je dt konstanta pa je ovaj dio koda
//komentiran:
//	pid_dt = (1024*((float)pid_overflows * 256 + TCNT2 - pid_overflowsTCNTWas))
/F_CPU;
//	pid_overflowsTCNTWas = TCNT2;
//	pid_overflows = 0;

	pid_error = pid_setpoint - input; //e(t)
//Integrator:
	pid_errorCumulative = pid_errorCumulative + pid_error * global_dt;

	//ograničavanje integralne komponente:
	if(pid_errorCumulative > 50) pid_errorCumulative = 50;
	if(pid_errorCumulative < -50) pid_errorCumulative = -50;

	pid_output = pid_KP * pid_error + pid_KI * (pid_errorCumulative) + pid_KD *
((pid_error - pid_errorWas)/global_dt);
	pid_errorWas = pid_error; //za D komponentu
}
```

Header datoteka:

```
#ifndef _REG_H_
#define _REG_H_

#define PID_DEADZONEELIMINATOR_DEFAULT 200
#define PID_DEADZONEELIMINATOR_START 350
#define PID_OVERFLOW_SETPOINT 2
#define PID_DEADZONE_SETPOINT 10
#define PID_BOUNDARY 900 //steps/s
#define PID_DEADENING 0

void regulator_pid(float input);
float regulator_pi(float input, float setpoint);

#include "reg.c"
#endif
```

5.2.5 MPU6050raw.c i MPU6050raw.h

Ovo je skraćeni modul za komunikaciju sa MPU6050 IMU modulom. Sadrži filter.

Uključivanje modula, definicije i deklaracije funkcija:

```
#include <math.h>
#include <util/atomic.h>
#define RAD_TO_DEG 57.295779513082320876798154814105
#define FILTER_MOVING_AVERAGE_SIZE 96

uint8_t filter_overflows = 0, filter_overflowsTCNTWas = 0;
float filter_averageArray[FILTER_MOVING_AVERAGE_SIZE];
float filter_average;
uint8_t filter_averageCounter = 0;
```

Funkcija `mpu6050_readVals` za očitavanje IMU modula:

```
void mpu6050_readVals(uint8_t devide_addr, int16_t *__rawdata) {
	uint8_t mpu6050_readVals_i;
	uint8_t mpu6050_readVals_array[14];
	ATOMIC_BLOCK(ATOMIC_FORCEON){ //Ovaj se dio koda ne smije prekinuti.
		i2c_start(devide_addr | I2C_WRITE);
		i2c_write(0x3B); //ACX registar
		_delay_us(10);
		i2c_start(devide_addr | I2C_READ);
		for(mpu6050_readVals_i = 0; mpu6050_readVals_i < 14; mpu6050_readVals_i++){
			if(mpu6050_readVals_i == 13)
			mpu6050_readVals_array[mpu6050_readVals_i] = i2c_read_nack();
			else
			mpu6050_readVals_array[mpu6050_readVals_i] = i2c_read_ack();
		}
		i2c_stop();
	}//Ovdje završava ATOMIC_BLOCK

//Spremi dobivene podatke na adresu __rawdata:
	__rawdata[0] = (((int16_t)mpu6050_readVals_array[0]) << 8) |
mpu6050_readVals_array[1];
	__rawdata[1] = (((int16_t)mpu6050_readVals_array[2]) << 8) |
mpu6050_readVals_array[3];
	__rawdata[2] = (((int16_t)mpu6050_readVals_array[4]) << 8) |
mpu6050_readVals_array[5];
	__rawdata[3] = (((int16_t)mpu6050_readVals_array[8]) << 8) |
mpu6050_readVals_array[9];
	__rawdata[4] = (((int16_t)mpu6050_readVals_array[10]) << 8) |
mpu6050_readVals_array[11];
	__rawdata[5] = (((int16_t)mpu6050_readVals_array[12]) << 8) |
mpu6050_readVals_array[13];
}
```

Filter:

```
void mpu6050_FilterDataXG(int16_t *unfilteredData, float *x_angle){

// izračunapanje kuta θ:
	*x_angle = atan((unfilteredData[0]/16384.0) / sqrt(pow(unfilteredData[1]/16384.0,2)
+ pow(unfilteredData[2]/16384.0,2)))*RAD_TO_DEG;

// Komplementarni filter. Izlaz je spremljen na lokaciju x_angle:
	*x_angle = 0.97 * (*x_angle + (unfilteredData[4]/131.0) * global_dt) + 0.03 *
*x_angle;
}
```

Funkcije za inicijalizaciju inercijske jedinice:

```
void mpu6050_init(void){
	i2c_start(MPU6050_ADDR);
	i2c_write(0x6B);
	i2c_write(0); //turn on
	i2c_stop();
}

uint8_t mpu6050_setDLPF(uint8_t mode){
	uint8_t reg, regWas;
	i2c_start(MPU6050_ADDR | I2C_WRITE);
	i2c_write(0x1A); //ACX register
	i2c_start(MPU6050_ADDR | I2C_READ);
	_delay_us(10);
	reg = i2c_read_nack();
	regWas = reg;
	reg |= mode;
	i2c_start(MPU6050_ADDR | I2C_WRITE);
	i2c_write(0x1A); //ACX register
	i2c_write(reg); //ACX register
	return regWas;
}
```

6 Zaključak

Iz opisanog matematičkog modela robota, vidljiva je nelinearna priroda sustava mobilnog inverznog njihala. S obzirom da je broj aktuatora manji od broja stupnjeva slobode gibanja, sustav je podupravljan iz čega proizlazi da je i strogo spregnut, a što je vidljivo iz diferencijalnih jednadžbi modela. To znači da je za promjenu pozicije robota nužno promijeniti orijentaciju i obrnuto, promjena orijentacije rezultira gibanjem robota.

Razvoj upravljačkog algoritma, digitalna obrada signala, informatika (kod i komunikacija), mehanika i ostale komponente projekta moraju raditi zajedno da se ovakav robot može realizirati. To zahtjeva pomno planiranje međudjelovanja ovih isprepletenih komponenti. Racionalnim i efektivnim rješenjima za probleme kod realizacije ovog robota možemo dobiti veoma precizan i stabilan robot.

Razrada raznih komponenti i sustava ovog robota kroz život projekta su se znatno mijenjali, demonstrirajući evoluciju ideje u završni proizvod. Ispitani su različiti pogoni robota, konstrukcije pa i algoritmi upravljanja i obrade signala. Upravljački algoritam se razvijao i optimizirao tokom projekta, te se morao prilagoditi ostalim promjenama na robotu poput mehaničke konstrukcije ili pogona. Eksperimentiralo se sa različitim parametrima PID regulatora i promatranje utjecaja na te parametre. Realizirana je i serijska komunikacija preko Bluetooth protokola za daljinsku i bežičnu promjenu parametara regulacije i upravljanja.

7 Popis literature

Ovo je popis korištene literature korištene kod izrade ovog završnog rada i realizacije mobilnog inverznog njihala. Korišteni su postojeći radovi kao referenca, te priručnici i dokumentacija proizvođača korištenih modula. Korišteni dokumenti su priloženi uz ovaj završni rad.

[1] Dokumentacija HC-05 Bluetooth modula.
   a) HC05-UserManual.pdf
   b) HC05.pdf

[2] Dokumentacija za NEMA17 koračne motore.
   a) ds-311.pdf

[3] Dokumentacija za MPU6050 IMU.
   a) MPU-6000-Register-Map1.pdf
   b) MPU-6000-Datasheet1.pdf

[4] Dokumentacija za upravljački modul koračnih motora A4988.
   a) A4988.pdf

[5] Dokumentacija za RK370SD istosmjerni motor.
   a) rk_370sd.pdf

[6] Balancing a Two-Wheeled Autonomous Robot, *Rich Chi Ooi, The University of Western Australia, School of Mechanical Engineering, 2003*
   a) Balancing a Two-Wheeled.pdf

[7] Samobalansirajući mobilni robot, *Tomislav Tomašić, Andrea Demetlika, FSB Zagreb, 2011*

[8] Samobalansirajući mobilni robot, *Sebastian Dovičin, FER Zagreb, 2018.*

[9] Balancing robot control and implementation , *Hsin-Chuan Sung, Texas A&M University, 2015*

[10] Two-Wheeled Self-Balancing Robot, *Hana Hellman*

[11] Dokumentacija za ATmega328p mikroupravljač.
   a) Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[12] Dokumentacija za VNH5019 modul.
   a) dual_vnh5019_motor_driver_shield.pdf