Samobalansirajući robot na dva kotača

Uroić, Ivan

Undergraduate thesis / Završni rad

2021

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:235:670352

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-04-01**

Repository / Repozitorij:

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

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Ivan Uroić

Zagreb, 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

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

Student:
Ivan Uroić
0035208916

Zagreb, 2021.

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

Zahvaljujem se prof. dr. sc. Mladenu Crnekoviću, dipl. ing. na ukazanom vremenu, savjetima i pruženoj prilici za izradu ovog rada pod njegovim mentorstvom.

Zahvaljujem se također svojoj obitelji i prijateljima na neprestanoj podršci tijekom studija.

Ivan Uroić

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

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 21 - 6 / 1**	
Ur.broj: **15 - 1703 - 21 -**	

ZAVRŠNI ZADATAK

Student: **IVAN UROIĆ** Mat. br.: 0035208916

Naslov rada na hrvatskom jeziku: **SAMOBALANSIRAJUĆI ROBOT NA DVA KOTAČA**

Naslov rada na engleskom jeziku: **SELFBALANCING ROBOT ON TWO WHEELS**

Opis zadatka:

Samobalansirajuće vozilo predstavlja jedan od ideala minimalističke konstrukcije mehaničkog uređaja. Posljedica toga je nestabilan sustav pa je zadatak upravljačke jedinice stabilizacija sustava uz ispunjavanje funkcije vožnje određenom brzinom u zadanom smjeru. To je moguće upotrebom 3-osnog akcelerometra i 3-osnog žiroskopa, mikrokontrolera i teorije vođenja sustava.
U radu je potrebno:

- definirati mehaničku strukturu samobalansirajućeg vozila,
- odabrati motore, senzore i upravljački mikrokontroler,
- izvesti matematički model sustava,
- procijeniti vrijednost predloženog rješenja.

Potrebno je navesti korištenu literaturu i ostale izvore informacija te eventualno dobivenu pomoć.

Zadatak zadan:
30. studenoga 2020.

Datum predaje rada:
1. rok: 18 veljače 2021.
2. rok (izvanredni): 5. srpnja 2021.
3. rok: 23. rujna 2021.

Predviđeni datumi obrane:
1. rok: 22.2. – 26.2.2021.
2. rok (izvanredni): 9.7.2021.
3. rok: 27.9. – 1.10.2021.

Zadatak zadao:

Prof. dr. sc. Mladen Crneković

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

SADRŽAJ

1. UVOD 1
2. STRUKTURA SAMOBALANSIRAJUĆEG VOZILA 3
2.1. Mehanička struktura vozila 3
2.2. Aktuatori 3
2.3. Senzori 3
2.4. Upravljački sustav 3
2.5. Napajanje 4
3. MATEMATIČKI MODEL SUSTAVA 5
3.1. Dinamika pojedinog kotača 5
3.2. Dinamika tijela vozila 6
3.3. Prostor stanja 9
4. REALIZACIJA SAMOBALANSIRAJUĆEG VOZILA 11
4.1. Odabrane komponente 11
4.1.1. Mehanička struktura vozila 11
4.1.2. Aktuatori 14
4.1.3. Senzori 15
4.1.4. Bluetooth uređaj 16
4.1.5. Upravljački sustav 17
4.1.6. Napajanje 19
4.2. Shema spajanja elektroničkih uređaja 20
4.3. Arduino kod 21
4.4. Grafički prikaz vozila u radu 24
4.5. Sklopni prikaz vozila 25
4.6. Aplikacija 28
4.7. Procijenjena vrijednost 30
5. ZAKLJUČAK 31
6. LITERATURA 32
7. PRILOZI 33
7.1. Arduino kod 33
7.2. Kod aplikacije u blok prikazu 38

POPIS SLIKA

Slika 1. Ravnotežno stanje pomoću balastnih voda 2
Slika 2. Simbolički prikaz samobalansirajućeg robota 4
Slika 3. Sile i momenti na pojedinom kotaču 5
Slika 4. Sile i momenti na tijelu robota 6
Slika 5. Prikaz kućišta u Catii 11
Slika 6. Prikaz kućišta u Ultimaker Curi 12
Slika 7. Proces 3D ispisa kućišta 12
Slika 8. Kotači 13
Slika 9. Straight BO Motor 14
Slika 10. MPU 6050 15
Slika 11. HC-05 Bluetooth modul 16
Slika 12. DC Motor Driver Dual H-Bridge L298N 17
Slika 13. Arduino Nano 18
Slika 14. MP0.8-12 JST akumulator 19
Slika 15. Radna karakteristika akumulatora 19
Slika 16. Shema spajanja elektroničkih uređaja 20
Slika 17. Grafički prikaz rada vozila (1) 24
Slika 18. Grafički prikaz rada vozila (2) 25
Slika 19. Usporedba pozicije težišta vozila 26
Slika 20. Prednji prikaz sklopljenog robota 26
Slika 21. Stražnji prikaz sklopljenog robota 27
Slika 22. Prikaz robota s bokocrta 27
Slika 23. Mobilna aplikacija 28

POPIS TABLICA

Tablica 1. Popis kupljenih komponenti 30

POPIS OZNAKA

Oznaka	Jedinica	Opis oznake
F_h	N	Sila u smjeru globalne osi *X*
F_v	N	Sila u smjeru globalne osi *Y*
F_{tr}	N	Sila trenja između pojedinog kotača i podloge
F_N	N	Sila podloge na pojedini kotač
g	m/s^2	Ubrzanje sile teže
J_k	kgm^2	Moment inercije pojedinog kotača, vratila i motora
J_T	kgm^2	Moment inercije tijela robota
Kut	°	Kut nagiba ravnotežnog stanja
L	m	Udaljenost težišta tijela robota od osi rotacije kotača
m_k	kg	Masa pojedinog kotača
m_T	kg	Masa tijela robota svedena u težište
M_k	Nm	Moment na pojedinom kotaču
R_k	m	Radijus kotača
t	s	Vrijeme
v	m/s	Horizontalna brzina gibanja centra kotača
x_1, $\dot{x}_1$, $\ddot{x}_1$	m, m/s, m/s^2	Pozicija, brzina, ubrzanje po osi x_1
x_2, $\dot{x}_2$, $\ddot{x}_2$	m, m/s, m/s^2	Pozicija, brzina, ubrzanje po osi x_2
y_1, $\dot{y}_1$, $\ddot{y}_1$	m, m/s, m/s^2	Pozicija, brzina, ubrzanje po osi y_1
y_2, $\dot{y}_2$, $\ddot{y}_2$	m, m/s, m/s^2	Pozicija, brzina, ubrzanje po osi y_2
θ, $\dot{\theta}$, $\ddot{\theta}$	rad, rad/s, rad/s^2	Kut, kutna brzina, kutno ubrzanje tijela robota
θ_k, $\dot{\theta}_k$, $\ddot{\theta}_k$	rad, rad/s, rad/s^2	Kut, kutna brzina, kutno ubrzanje pojedinog kotača

SAŽETAK

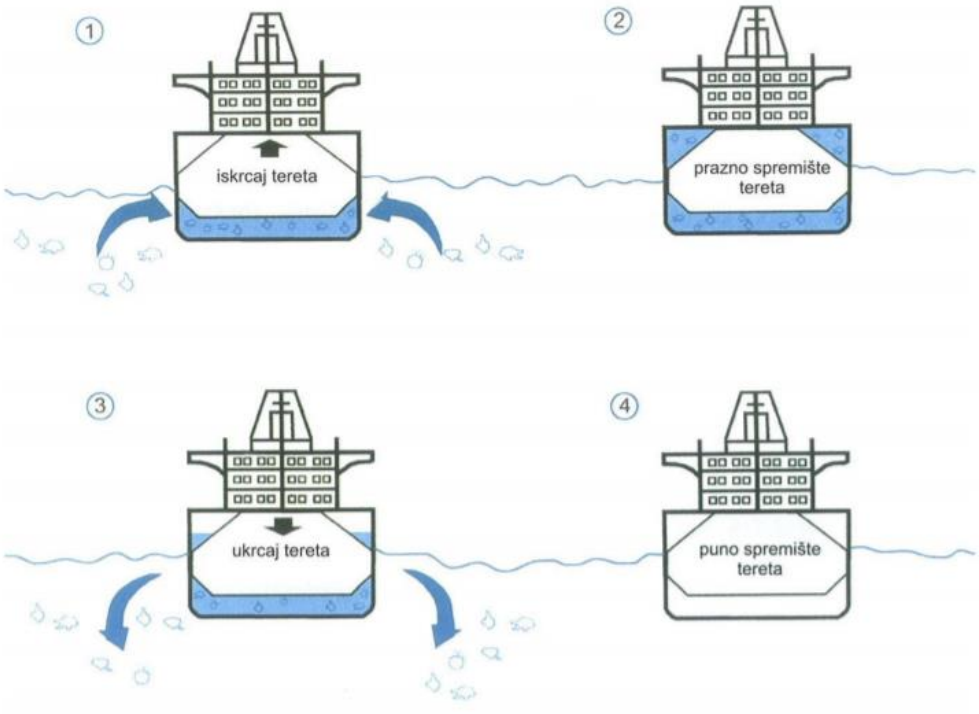
U ovom završnom radu prikazan je razvoj i konstruiranje samobalansirajućeg vozila na dva kotača minimalističke konstrukcije. Nakon objašnjenja sustava, njegovih dijelova i primjene istih, definiran je matematički model sustava vozila. Razumijevanjem matematičkog modela, potrebnih komponenti i načina na koji oni međusobno funkcioniraju, provodi se realizacija projekta. Realizacija projekta zahtjeva odabir potrebnih komponenti, dizajniranje konstrukcije, izrada programskog rješenja za mikrokontroler, također izrada i dizajniranje mobilne aplikacije za upravljanje vozilom, te sklapanje svih navedenih komponenti u jednu cjelinu. Jedan od željenih ciljeva je realizacija projekta uz što manje materijalnih troškova. Tokom izrade ovog projekta korišteni su programski paketi Catia V5R21, Arduino IDE, MIT App Inventor, Ultimaker Cura 4.8.0 i EasyEDA.

Ključne riječi: samobalansirajuće vozilo, samobalansirajući robot na dva kotača, vozilo na dva kotača, Arduino, dinamički sustav, Bluetooth upravljanje.

1. UVOD

Opće nam je poznato kako vozila koja sadrže dva kotača, poput bicikla, motora, romobila predstavljaju mehanički nestabilan sustav. Prema definiciji iz Hrvatske opće enciklopedije, stabilnost je svojstvo mehaničkog, tehničkog ili drugog sustava da samostalno održava ili uspostavlja ravnotežno stanje nakon prestanka djelovanja uzroka koji je ravnotežu poremetio. U mehanici je riječ o statičkoj stabilnosti položaja tijela, a u tehnici o stabilnosti konstrukcija, vozila, zrakoplova i dr. Statička stabilnost svojstvo je održavanja ravnoteže, a pod dinamičkom stabilnošću podrazumijeva se proces vraćanja u ravnotežno stanje, koji se obično odvija na oscilatorni način oko položaja ravnoteže sa sve manjim amplitudama. Kao primjer dinamičke stabilnosti, kod upravljanja motociklom u gibanju, osoba vrši proces vraćanja sustava u stabilno stanje. Osoba to vrši pomicanjem težišta i zakretanjem upravljača (guvernala) motornog vozila.

Kod dinamičke stabilnosti plovila proces vraćanja u ravnotežno stanje vrši automatizirani sustav sa povratnom vezom pomoću balastnih voda. Povratna veza je proces u kojemu informacija o stanju nekog sustava ili jednog njegovog dijela daje zaključak o tome koliko je sustav blizu željenog stanja. Ta informacija dobiva se putem senzora koji mogu biti smješteni na samom uređaju ili u njegovoj okolini. U slučaju manje količine tereta, brod se puni balastnim vodama, dok u slučaju veće količine tereta, količina balastnih voda se smanjuje i time se postiže ravnotežno stanje.



Slika 1. Ravnotežno stanje pomoću balastnih voda

Svako vozilo na dva kotača možemo gledati kao nestabilan sustav koji promatramo po principu obrnutog njihala. Dovesti takav sustav u ravnotežu zahtjeva neprestano ispravljanje. To se može postići korištenjem elektronike, hidraulike ili pneumatike. U ovom radu dinamičku stabilnost vozila na dva kotača vršit će elektronički sustav pomoću povratne veze. Također, bežičnim upravljanjem vozila vršit ćemo gibanje određenom brzinom u određenom smjeru, nakon čega će vozilo i dalje biti u mogućnosti vraćanja u ravnotežno stanje.

2. STRUKTURA SAMOBALANSIRAJUĆEG VOZILA

2.1. Mehanička struktura vozila

Mehanička struktura samobalansirajućeg vozila je relativno jednostavna, te se često sastoji od visokokvalitetnog metalnog kućišta koje se može kupiti u trgovini robotike, te dva kotača koja su povezana na elektromotore preko vratila.

2.2. Aktuatori

Aktuatori omogućuju kretanje, odnosno pomicanje kako bi se vozilo moglo postaviti u ravnotežni položaj. Kada je riječ o balansiranju vozila, koriste se električni motori (elektromotori). Ovisno o željenoj preciznosti, snazi, kvaliteti, raspoloživim sredstvima, dostupnosti i namjeni koristimo različite vrste elektromotora. U slučaju kada želimo veću preciznost, odnosno bolju kontrolu, onda uzimamo istosmjerne koračne elektromotore, uz osjetno veće troškove. Međutim, ako nam preciznost nije toliko bitna, te ako želimo funkcionalno i povoljno rješenje, uzimamo obične istosmjerne elektromotore.

2.3. Senzori

Senzor ili mjerno osjetilo je dio mjernoga sustava koji je izravno u dodiru s mjerenom veličinom, te daje izlazni signal ovisan o njezinu iznosu. Zbog prevladavajuće primjene električnih i elektroničkih sustava, većina mjernih osjetila pretvara mjerenu veličinu u električki mjerljiv signal. Često korišteni senzori kod samobalansirajućeg vozila su ultrazvučni senzori koji nam daju informaciju o mogućim preprekama na putu, te akcelerometar i žiroskop koji šalju povratnu informaciju o trenutnom položaju.

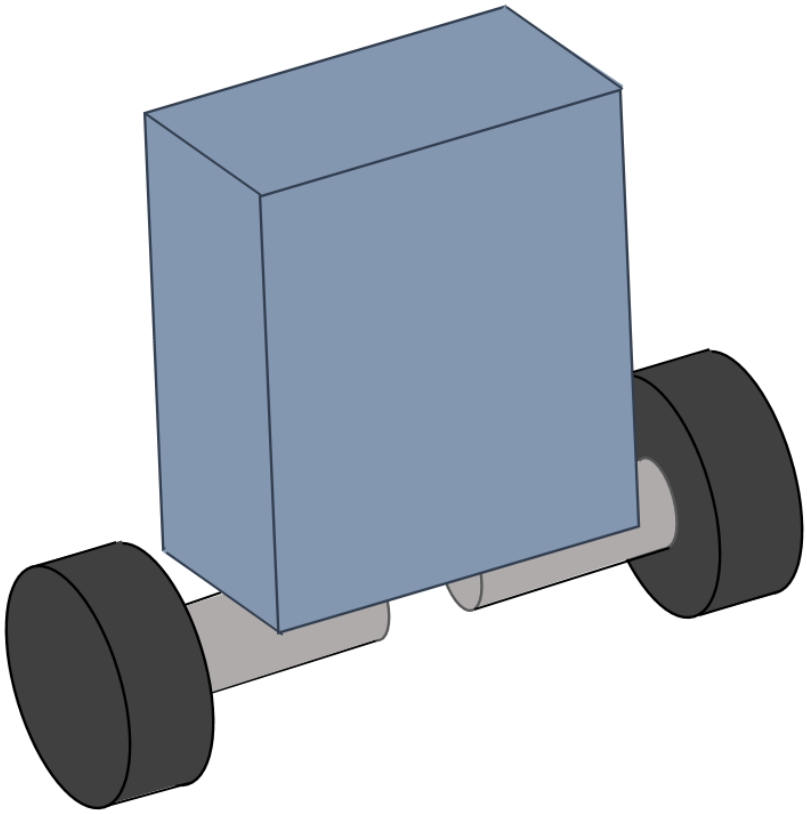
2.4. Upravljački sustav

Upravljački sustav je centralni dio samobalansirajućeg vozila koji se sastoji od mikrokontrolera koji obrađuje povratne informacije sa ugrađenih senzora, kontrolira rad motora, te komunicira i obrađuje podatke s drugih uređaja. Međutim, glavni zadatak mu je održati vozilo u ravnotežnom položaju.

2.5. Napajanje

Veliku ulogu kod samobalansirajućih vozila na dva kotača ima napajanje. Kao i kod drugih električnih vozila napajanju se posvećuje velika pažnja. Zbog bolje ekonomičnosti, kod električnih vozila, najčešće se koriste akumulatori jer oni imaju mogućnost ponovnog punjena. Glavne vrijednosti koje opisuju akumulator su njegov kapacitet (izražen u Ah), napon (izražen u V), veličina, masa, te radna karakteristika. Budući da u ukupnoj masi električnih vozila akumulator zauzima značajan postotak, pozicioniranje njega unutar strukture igra bitnu ulogu.

Implementacija svih navedenih komponenti u jednu cjelinu rezultira izradom samobalansirajućeg robota. Simbolički prikaz robota vidi se na Slici 2.



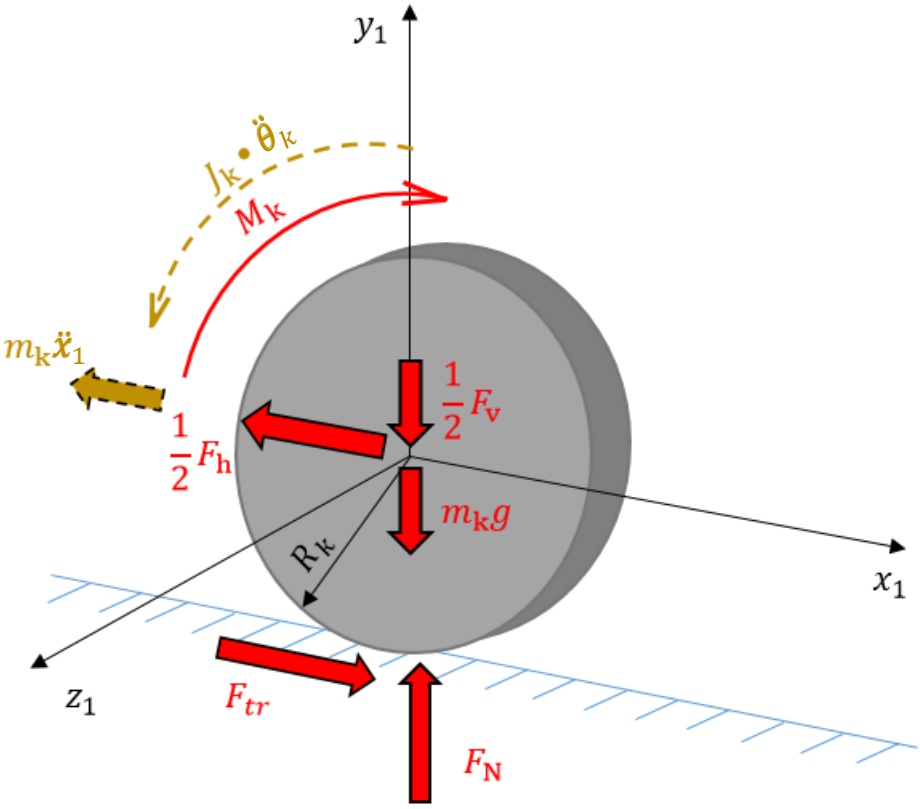
Slika 2. Simbolički prikaz samobalansirajućeg robota

3. MATEMATIČKI MODEL SUSTAVA

Radi boljeg razumijevanja ponašanja sustava kreira se i analizira matematički model istog. Ovim korakom računa se dinamički model pojedinog dijela sustava. Njihovom kombinacijom dobivaju se jednadžbe gibanja koje služe za provedbu simulacije korištenjem programskog paketa.

3.1. Dinamika pojedinog kotača

Sve sile i momenti koji djeluju na pojedini kotač prikazane su na Slici 3.



Slika 3. Sile i momenti na pojedinom kotaču

Kotač je prikazan u prostornom prikazu, te su inercijske sile i momenti prikazani tamno žutom bojom radi lakšeg snalaženja. U ishodištu koordinatnog sustava djeluju reaktivna horizontalna i vertikalna komponenta sile tijela vozila i težina samog kotača. Iznos reaktivnih komponenata sile tijela vozila je podijeljena sa dva budući da se na tijelu nalaze dva kotača. Djelovanjem elektromotora na

kotač javlja se moment M_k. Gibanjem vozila po podlozi, javlja se reaktivna sila podloge i sila trenja. Faktor trenja između kotača i podloge će se postaviti na vrijednost jedan (μ=1) i faktor trenja unutar elektromotora će se zanemarit ($b\dot{x}$=0) radi pojednostavljenja matematičkog modela. Ovime rezultati neće biti izrazito točni, međutim razlika neće biti uvelike značajna. Pomoću Slike 3. izvedeni su sljedeći izrazi.

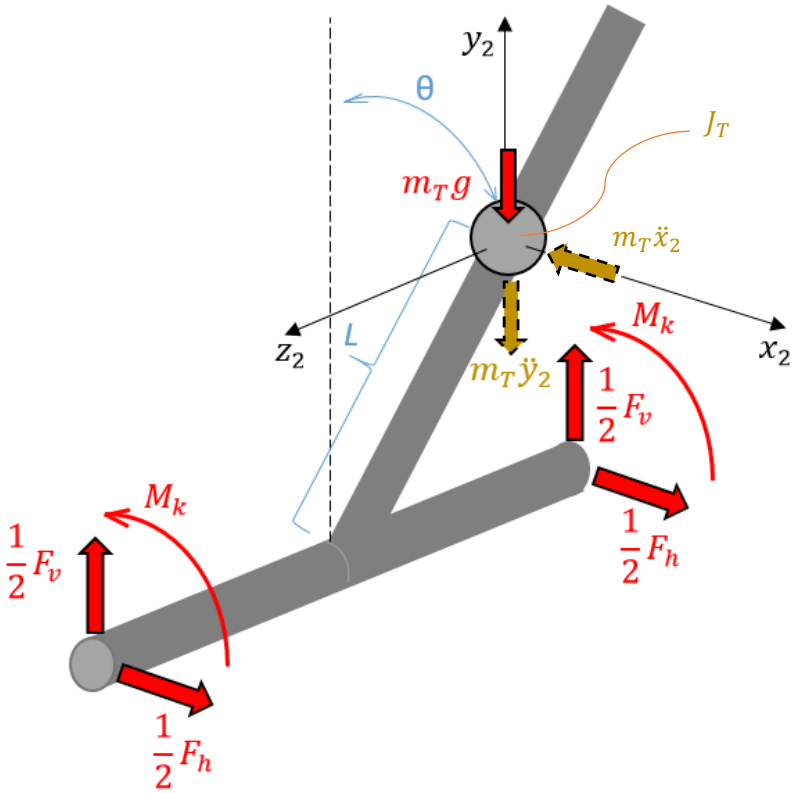
$$\sum F_{x1} = 0: \qquad -\frac{1}{2}F_h + F_{tr} = m_k\ddot{x}_1 \tag{3.1}$$

$$\sum F_{y1} = 0: \qquad -\frac{1}{2}F_v - m_k g + F_N = 0 \tag{3.2}$$

$$\sum M_{z1} = 0: \qquad J_k\ddot{\theta}_k = M_k - F_{tr}R_k \tag{3.3}$$

3.2. Dinamika tijela vozila

Sve sile i momenti koji djeluju na tijelo robota prikazane su na Slici 4.



Slika 4. Sile i momenti na tijelu robota

Budući da na tijelo djeluju dva elektromotora, ukupni moment kojim oni djeluju na tijelo je iznosa $2M_k$. Koordinatni sustav (x_2, y_2, z_2) smješten je u centar težišta tijela od kojeg je os rotacije kotača udaljen za iznos L. Na težište tijela djeluju inercijske sile koje su rastavljene na komponente i prikazane tamno žutom bojom. Prijašnjim prikazom definiraju se jednadžbe u nastavku.

$$\sum F_{x2} = 0: \qquad F_h = m_T \ddot{x}_2 \tag{3.4}$$

$$\sum F_{y2} = 0: \qquad F_v - m_T g = m_T \ddot{y}_2 \tag{3.5}$$

$$\sum M_{z2} = 0: \qquad J_T \ddot{\theta} = -2M_k - F_h l\cos\theta + F_v l\sin\theta \tag{3.6}$$

Uvrštavanjem izraza (3.4) i (3.5) u izraz (3.6) dobije se sljedeći izraz:

$$J_T \ddot{\theta} = -2M_k - m_T \ddot{x}_2 l\cos\theta + m_T \ddot{y}_2 l\sin\theta + m_T g l\sin\theta \tag{3.7}$$

Sljedeći korak je povezivanje definiranje koordinatnih sustava kotača i tijela vozila. To se vrši korištenjem geometrije i trigonometrijskih funkcijama.

$$x_2 = x_1 + l\sin\theta \tag{3.8}$$

$$y_2 = y_1 + l\cos\theta \tag{3.9}$$

Dvostrukom derivacijom izraza (3.8) i (3.9) dobiju se sljedeći izrazi:

$$\ddot{x}_2 = \ddot{x}_1 - l\sin\theta\left(\dot{\theta}\right)^2 + l\cos\theta\ddot{\theta} \tag{3.10}$$

$$\ddot{y}_2 = -l\cos\theta\left(\dot{\theta}\right)^2 + l\sin\theta\ddot{\theta} \tag{3.11}$$

Želi se postići jednadžba ovisnosti derivacije pomaka kotača x_1 i kuta nagiba vozila θ. To se postiže sređivanjem izraza (3.7) koji poprima sljedeći oblik:

$$J_T \ddot{\theta} = -2M_k + m_T g l\sin\theta + m_T l(-\ddot{x}_1\cos\theta + l\ddot{\theta}) \tag{3.12}$$

Gdje je izraz u zagradi dobio oblik sređivanjem izraza (3.13) i uvođenjem pojednostavljenja koje glasi: $\left(\dot{\theta}\right)^2 \approx 0$

$$-\ddot{x}_2\cos\theta + \ddot{y}_2\sin\theta = -\ddot{x}_1\cos\theta + l\ddot{\theta} \tag{3.13}$$

Nastavkom sređivanja izraza (3.12) dobije se sljedeći izraz:

$$(J_T + m_T l^2)\ddot{\theta} = -2M_k + m_T g l\sin\theta - m_T l\ddot{x}_1\cos\theta \tag{3.14}$$

Dobiveni izraz (3.14) je napisan u ciljanom obliku gdje je izražena ovisnost kutne akceleracije $\ddot{\theta}$ o horizontalnoj akceleraciji kotača $\ddot{x}_1$. To predstavlja prvu jednadžbu gibanja sustava.

Sljedeći traženi izraz mora prikazati ovisnost kuta zakreta kotača θ i njihovog horizontalnog pomaka x_1. To se postiže preko geometrije kotača uz uvedenu pretpostavku da nema proklizavanja između kotača i podloge po kojoj se vozilo giba. Izvod traženog izraza je prikazan u nastavku.

$$v = r \bullet \frac{2\pi}{t} = \frac{\theta_k}{t}$$

$$\dot{\theta}_k = \frac{\dot{x}_1}{R_k} \quad / \frac{d}{dt}$$

$$\ddot{\theta}_k = \frac{\ddot{x}_1}{R_k} \tag{3.15}$$

Uvrštavanjem izraza (3.1) i (3.15) u izraz (3.3) dobije se:

$$\frac{J_k}{R_k}\ddot{x}_1 = M_k - \frac{1}{2}F_h R_k - m_k R_k \ddot{x}_1 \tag{3.16}$$

Daljnjim uvrštavanjem izraza (3.4) u izraz (3.16) dobije se:

$$\frac{J_k}{R_k}\ddot{x}_1 = M_k - \frac{1}{2}m_T R_k \ddot{x}_2 - m_k R_k \ddot{x}_1 \tag{3.17}$$

Korištenjem izraza (3.10) u uvođenjem sljedećih pojednostavljenja dobije se izraz druge jednadžbe gibanja:

$$\sin\theta \approx \theta \qquad \ddot{x}_1\cos\theta \approx \ddot{x}_1 \qquad \cos\theta\ddot{\theta} \approx \ddot{\theta}$$

$$\left(\frac{J_k}{R_k} + m_k R_k + \frac{1}{2}m_T R_k\right)\ddot{x}_1 = M_k - \frac{1}{2}m_T R_k l\ddot{\theta} \tag{3.18}$$

Također, prva jednadžba gibanja sada glasi:

$$(J_T + m_T l^2)\ddot{\theta} = -2M_k + m_T g l\theta - m_T l\ddot{x}_1 \tag{3.19}$$

3.3. Prostor stanja

Sustav će biti prikazan u obliku prostora stanja. Opći oblik prostora stanja glasi:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

U ovom zapisu sustava ulazni podaci su vezani uz pomak *x* i kut nagiba θ. Vektor stanja x sastoji se od zavisnih veličina $x, \dot{x}, \theta, \dot{\theta}$. Izlazni vektor y sastoji se od izlaznih veličina. Radi kraćih zapisa i lakšeg snalaženja, uvode se sljedeće supstitucije:

$$E = J_T + m_T l^2, \quad F = m_T g l, \quad G = m_T l,$$

$$H = \frac{J_k}{R_k} + m_k R_k + \frac{1}{2} m_T R_k, \qquad I = \frac{1}{2} m_T R_k l \tag{3.20}$$

Nakon sređivanja jednadžbi i uvođenjem supstitucije, konačan oblik prostora stanja sustava prikazan je u sljedećem izrazu (3.21).

$$\dot{x}_2 = \ddot{x}_1 \,, \;\; \dot{\theta}_2 = \ddot{\theta}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\theta} \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-FI}{EH - GI} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{-HF}{-EH + GI} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \theta \\ \theta_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{E + 2I}{EH - GI} \\ 0 \\ \frac{2H + G}{-EH + GI} \end{bmatrix} u \tag{3.21}$$

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \theta \\ \theta_2 \end{bmatrix}$$

Prema izrazu (3.21) vidi se kako matrice A, B, C i D glase prema sljedećem:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{-FI}{EH - GI} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \dfrac{-HF}{-EH + GI} & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ \dfrac{E + 2I}{EH - GI} \\ 0 \\ \dfrac{2H + G}{-EH + GI} \end{bmatrix}$$

$$C = [0 \quad 0 \quad 1 \quad 0]$$

$$D = [0]$$

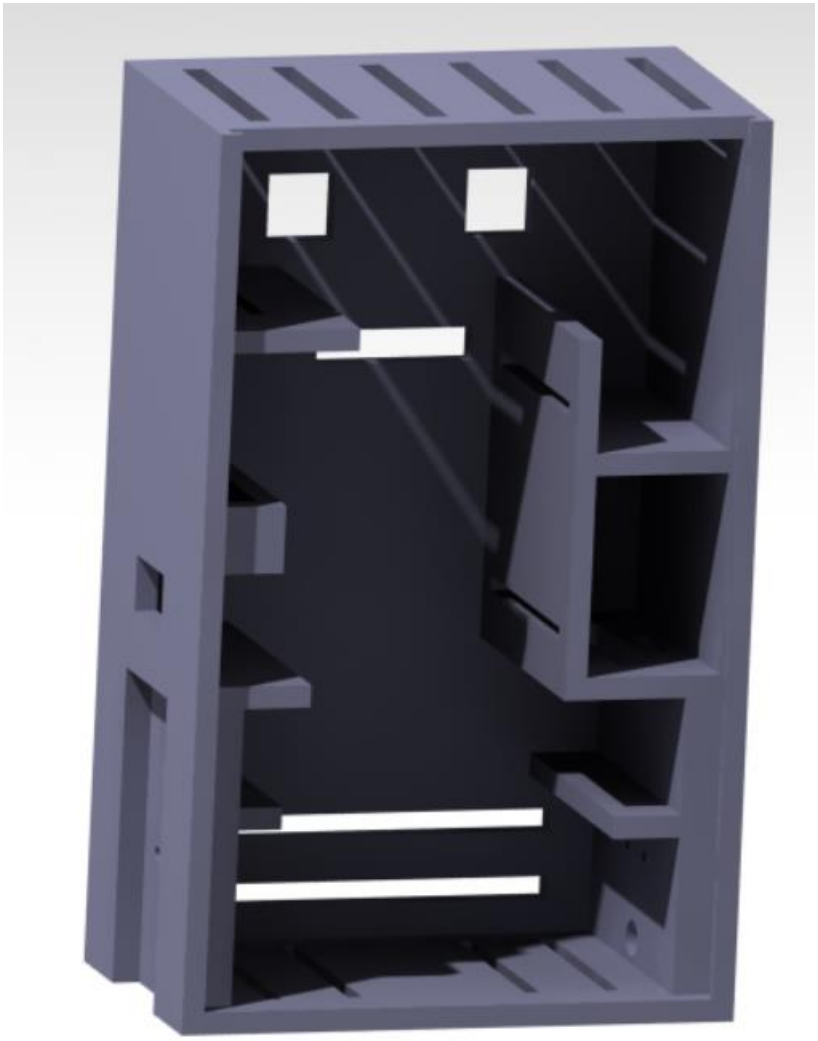
Prostor stanja za bilo koje samobalansirajuće vozilo je određeno. Sustav na svojem izlazu *y* pokazuje iznos kuta nagiba θ na ulazni moment kotača M_k. Budući da se kod realizacije ovog sustava budu koristili obični istosmjerni elektromotori, poznato je kako je moment kojim motori djeluju funkcija ovisna o upravljanoj jakosti struje *M(I)*.

4. REALIZACIJA SAMOBALANSIRAJUĆEG VOZILA

4.1. Odabrane komponente

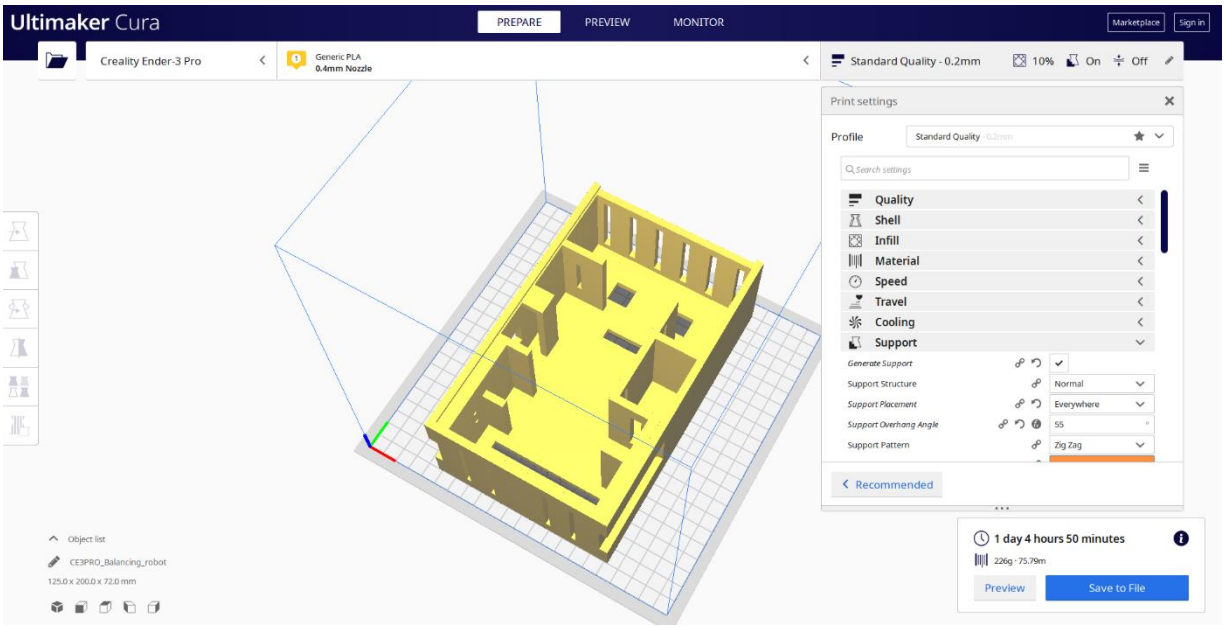
4.1.1. Mehanička struktura vozila

Kućište je konstruirano u programskom paketu Catia V5R21. Oblik, dimenzije i unutarnja struktura prilagođeni su obliku i rasporedu komponenata od kojih se vozilo sastoji, uzimajući u obzir kako bi se težište sklopljenog vozila nalazilo u najpovoljnijem položaju. Sljedeća slika prikazuje „renderani“ prikaz konstruiranog kućišta.

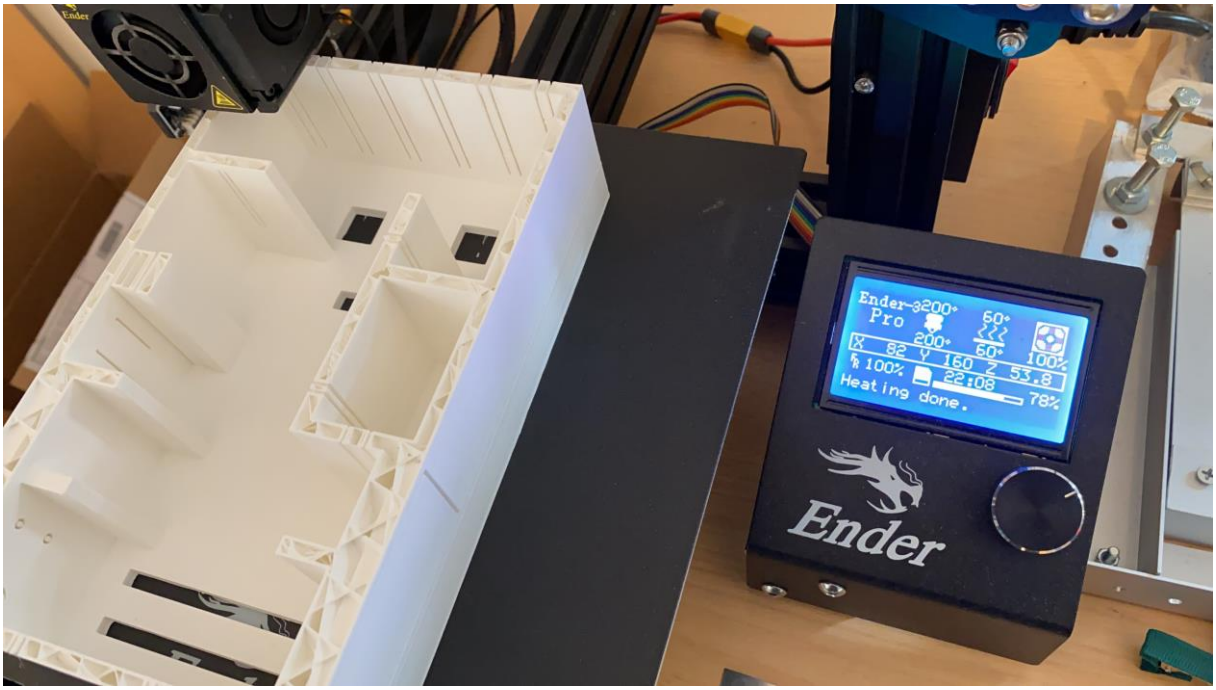


Slika 5. Prikaz kućišta u Catii

Sljedeći korak je pretvaranje konstruiranog kućišta u geometrijski kod koji će printer moći prepoznati i izvršiti (G-code). Korišteni programski paket za tu namjenu je Ultimaker Cura 4.8.0. U navedenom programskom paketu odabire se vrsta materijala (eng. filament), gustoća stjenke, kvaliteta i način 3-D ispisa.



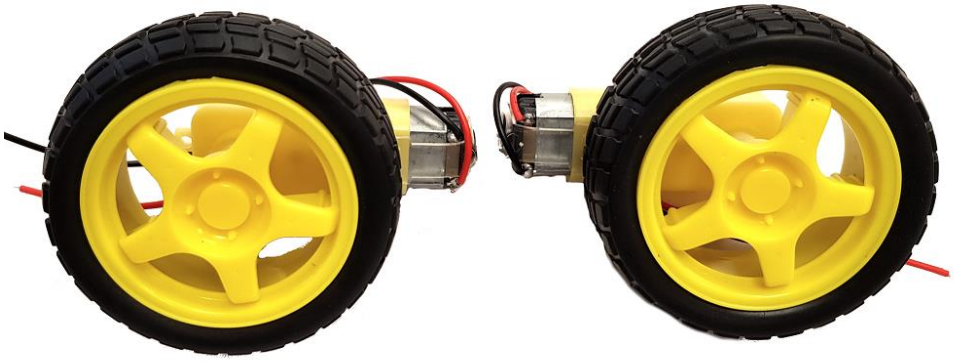
Slika 6. Prikaz kućišta u Ultimaker Curi



Slika 7. Proces 3-D ispisa kućišta

Ispis je trajao približno 29 sati korištenjem PLA materijala (eng. Polylactic acid) s ispunom stjenke 10 %. Zbog korištene ispune i materijala kućište je čvrsto i stabilno, a ujedno i lagano. Za izradu kućišta korišten je 3-D printer Ender 3 Pro od tvrtke Creality.

Preko elektromotora i njegovog vratila pričvršćeni su kotači promjera 65 mm za kućište.



Slika 8. Kotači

4.1.2. Aktuatori

Poznato je da su istosmjerni „Hobby“ elektromotori pristupačni i nepouzdani. Osobno sam htio realizirati vozilo korištenjem dva istosmjerna elektromotora naziva „Straight BO Motor“ koji su namijenjeni za Arduino/Raspberry-PI projekte. Motori ove vrste spajaju se sa upravljačkom jedinicom preko H-mosta sa dvije žice. Izlazna snaga elektromotora ovisi o naponu i jakosti struje. Budući da je motor spojen samo sa dvije žice, jedini način upravljanja je polaritet, duljina trajanja i iznos električnog impulsa.

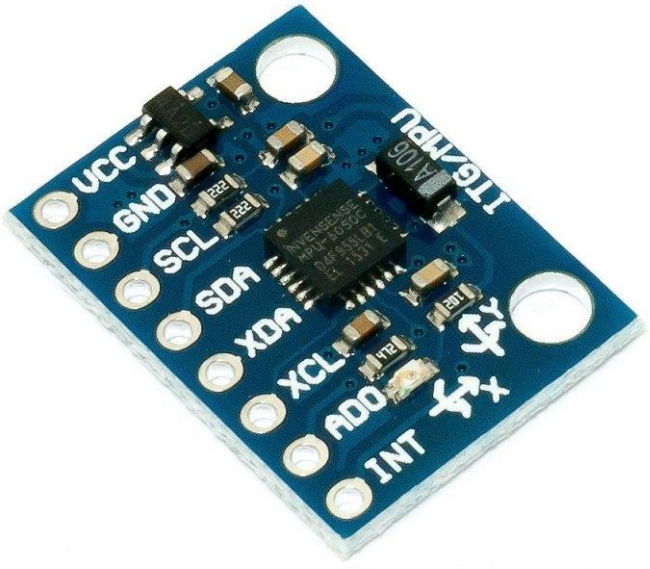
Slika 9. Straight BO Motor

Karakteristike motora su sljedeće:

- Napon: 3 - 6 V
- Brzina vrtnje: 300 o/min
- Dimenzije: 62 mm x 28 mm x 20 mm

4.1.3. Senzori

Za senzor položaja vozila korišten je troosni žiroskop i troosni akcelerometar MPU-6050. Akcelerometar je uređaj koji detektira sile koje djeluju na njega u tri smjera (osi x, y, z). U stanju mirovanja, detektirat će gravitacijsku silu, a prilikom kretanja, detektirat će sve sile koje djeluju na njega. Baš zbog toga što "osjeti" gravitaciju, s ovim senzorom mogu se mjeriti i detektirati udarci, nagibi, kretanje, tapkanje, koraci itd. Žiroskop, s druge strane, je u mogućnosti detektirati orijentaciju i kutnu brzinu te će dati dodatne korisne informacije o položaju objekta na kojemu se nalazi senzor, također u 3 smjera (osi x, y, z). Zbog osjetljivosti akcelerometra na vibracije (smetnje) i zanosnog učinka (eng. drift effect) žiroskopa, njihovi podaci se kombiniraju za bolja očitanja nagiba. Za nagib vozila u stupnjevima, potrebno je obraditi očitane podatke sa senzora koristeći različitih filtera poput „Kalman“ ili „Komplementarnog“ filtera. Međutim, još jedna metoda očitanja podataka koja se može primijeniti, kao na primjer kod odabranog senzora „MPU 6050“ je koristeći ugrađenog procesora - DMP (eng. Digital Motion Processor), koji obrađuje podatke sa žiroskopa i akcelerometra, te ih pretvara u nagib i kutnu brzinu. Podaci se potom obrađuju korištenjem „jrowberg“ Arduino biblioteke kojom se također kontrolira rad senzora. Web poveznica navedene biblioteke nalazi se u popisu literature pod rednim [12.].

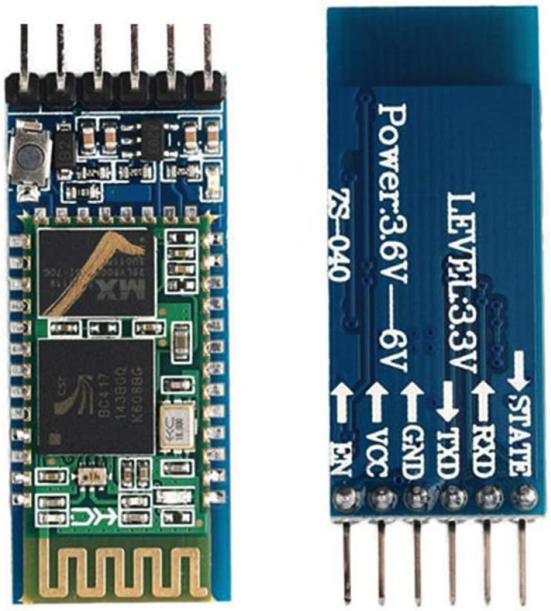


Slika 10. MPU 6050

4.1.4. Bluetooth uređaj

Za komunikaciju mobitela i upravljačke jedinice upotrijebljen je HC-05 Bluetooth modul. Bluetooth je način bežičnog prijenosa informacija. Za prijenos podataka koriste se radio valovi frekvencije od 2.402 GHz do 2.480 GHz. Prednost Bluetooth komunikacije ispred drugih bežičnih načina komunikacije je u vrlo jednostavnom uparivanju uređaja. Bluetooth uređaj šalje vrlo slabi signal (oko 1 mW) što ograničava domet komunikacije (približno 10 m), ali i time smanjuje potrošnju baterije. Bluetooth je namijenjen za prijenos manje količine podataka i za jednostavno upravljanje.

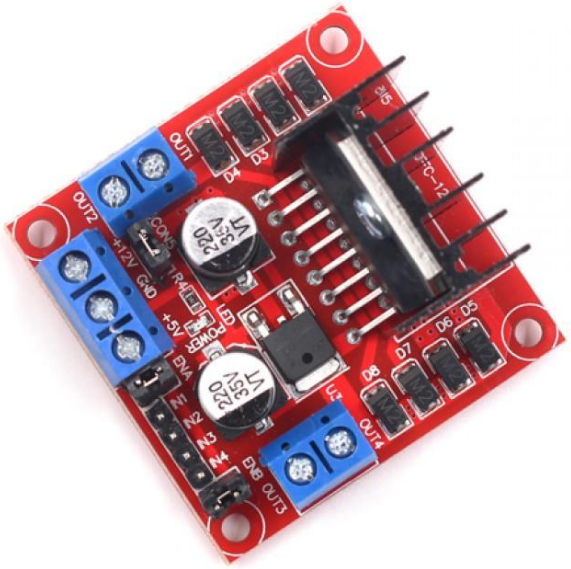
Svrha komunikacije je upravljanje brzine i smjera kretanja vozila. Kako bi se što više rasteretio mikrokontroler, aplikacija sa mobitela, preko Bluetootha, umjesto da kontinuirano šalje trenutno očitane podatke, aplikacija šalje samo promjene podataka. Kao primjer, kada korisnik pritisne tipku za kretanje vozila, pošalje se samo jedna vrijednost sve dok korisnik ne otpusti pritisnutu tipku, čime se vrijednost ponovno mijenja.



Slika 11. HC-05 Bluetooth modul

4.1.5. Upravljački sustav

Za upravljanje istosmjernih elektromotora koristi se „DC Motor Driver Dual H-Bridge L298N“. H-most omogućava zamjenu polariteta na svojim izlazima pa tako omogućava kontrolu vrtnje elektromotora u oba smjera na vrlo jednostavan način. Odabranim H-mostom možemo kontrolirati dva istosmjerna motora u isto vrijeme. U ovisnosti o naredbama mikrokontrolera, H-most pokreće elektromotore u željenom smjeru. On sadrži pretvarač napona sa 12 V na 5 V, što se pokazalo korisnim u realizaciji ovog projekta. S naponom od 5 V napaja se Arduino mikrokontroler.



Slika 12. DC Motor Driver Dual H-Bridge L298N

Karakteristike H-mosta:

- Max. snaga: 25 W
- Napon: 5-35 V (za motor), 5 V za logiku
- Jakost struje: 2 A
- Dimenzije: 43 mm x 43 mm x 23 mm

Upravljačka pločica je Arduino Nano. Zbog sljedećih karakteristika, Arduino Nano se smatra idealnim izborom za realizaciju ovog projekta.

Slika 13. Arduino Nano

Karakteristike Arduino Nano:

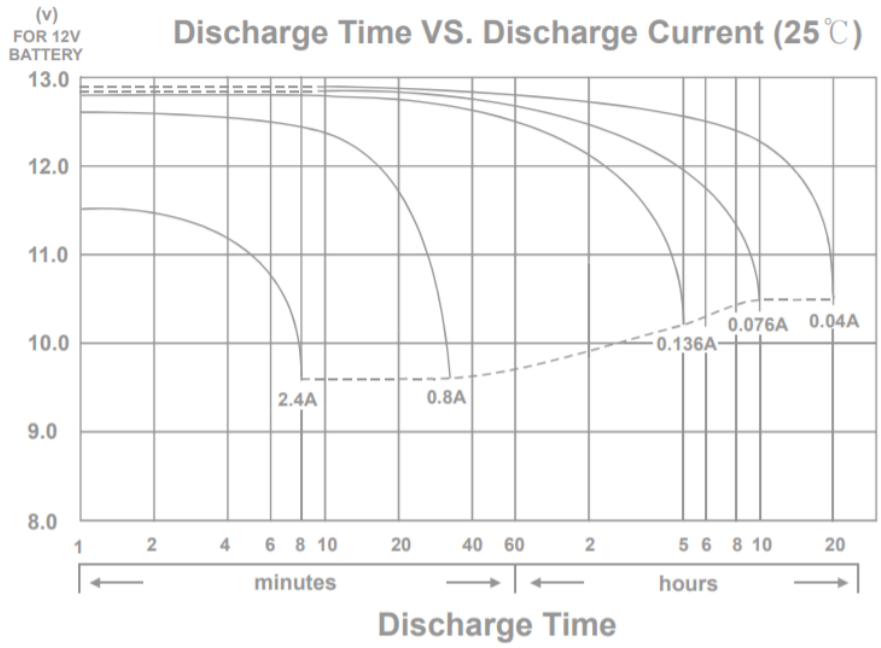
- Mikroprocesor: Atmelov Atmega328
- Broj pinova: 22 I/O pina (14 digitalnih)
- Frekvencija rada: 16 MHz
- Radni napon: 5 V
- Potrošnja struje: 19 mA
- Flash memorija: 32 KB

Za stabiliziranje vozila koristi se otvorena petlja koja se pokreće nakon uspješnog spajanja mobilne aplikacije na HC-05 Bluetooth modul. Bluetooth modul tu informaciju šalje preko „STATE“ pina. Svakim prolaskom kroz petlju vozilo pokuša dovesti u uspravan položaj što omogućavaju podaci dobiveni sa žiroskopa i akcelerometra. Slanjem naredbe za kretanje naprijed ili nazad mijenja se ciljani kut „ravnoteže“. Navedeni kut korisnik zadaje u samoj aplikaciji. Što je kut veći, brzina gibanja će biti veća. Većim kutom nagiba, zbog utjecaja gravitacijskog polja, potreban je veći moment kako bi vozilo ostalo u zadanom položaju. Skretanje se vrši na sličan način, samo što je u funkciji jedan kotač. Na primjer, želimo li da vozilo skrene u desno, u slučaju da je vozilo nagnuto prema naprijed kreće se samo lijevi kotač dok desni miruje, a u slučaju da je vozilo nagnuto prema nazad kreće se samo desni kotač, dok lijevi miruje. Za manevar skretanja u lijevo koristi se isti princip. Time se u isto vrijeme postiže stabilizacija i manevar skretanja vozila.

4.1.6. Napajanje

Korištenjem vanjskog izvora napajanja (ispravljača), pri početnim testiranjima, mjerena je potrošnja električne energije. Na temelju tih mjerenja za izvor napajanja izabrana je akumulatorska baterija „MP0.8-12 JST“ od tvrtke Multipower. Njezina radna karakteristika, kapacitet, izlazni napon, dimenzije i masa činile su ovaj akumulator kao idealan izbor za ovaj projekt.

Slika 14. MP0.8-12 JST akumulator



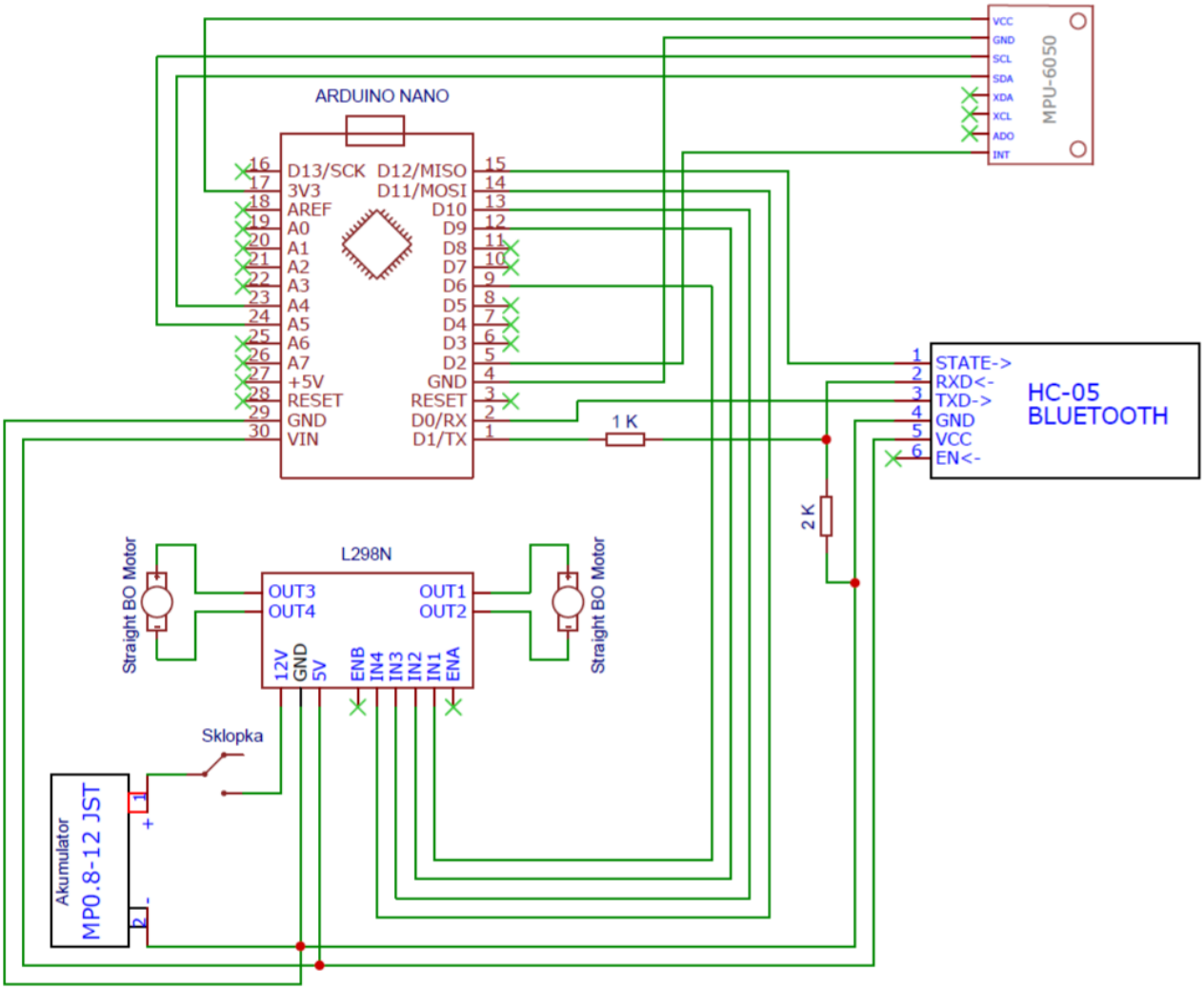
Slika 15. Radna karakteristika akumulatora

Karakteristike MP0.8-12 JST akumulatora:

- Nominalni napon: 12 V
- Kapacitet: 0.8 Ah
- Masa: 370 g
- Dimenzije: 96 mm x 25 mm x 61.5 mm

4.2. Shema spajanja elektroničkih uređaja

Na sljedećoj Slici 16 je prikazan način na koji su međusobno spojeni elektronički uređaji. Za ovaj prikaz korišten je online programski paket EasyEDA. Arduino kod napisan je prema ovoj shemi spajanja.



Slika 16. Shema spajanja elektroničkih uređaja

4.3. Arduino kod

Budući da se radi o jednojezgrenom procesoru, kod mora biti napravljen na efikasan način kako bi procesor mogao primiti i obraditi sve podatke sa senzora, a u isto vrijeme stabilizirati vozilo.

Prilikom uključivanja vozila, program se vrti u petlji (while(!BTconnected)) sve dok se ne uspostavi uspješno povezivanje Bluetooth modula i mobilnog uređaja. To se postiže očitanjem vrijednosti stanja Bluetooth modula (BTpin) i provjerom ako je veza uspostavljena (BTconnected == true). Nakon uspješne uspostave veze, program ulazi u otvorenu petlju i može krenuti sa balansiranjem vozila.

```
pinMode(BTpin, INPUT);
while (!BTconnected)
{
  if (digitalRead(BTpin) == HIGH)
  {
     BTconnected = true;
  }
}
```

U mobilnu aplikaciju unosi se brojčana vrijednost koja predstavlja „brzinu kretanja“ vozila. Inicijalna vrijednost brzine (speedValue) je nula. Pritiskom gumba za kretanje u naprijed aplikacija preko Bluetooth veze šalje niz znakova (String) koji sadrži unesenu brzinu i oznaku za novi red (\n), a pritiskom gumba za kretanje u nazad, šalje negativnu vrijednost. Za skretanje vozila u željenu stranu, aplikacija šalje niz znakova („R\n“ ili „L\n“). Brojčana vrijednost unesene brzine pretvara se u vrijednost željenog nagiba dijeleći s deset da bi dobili finiji raspon kontrole. Na dobiveni željeni nagib dodaje se vrijednost neutralne pozicije iznosa 3.65, što je očitano sa žiroskopa i akcelerometra.

```
if (Serial1.available() > 0)
{
   speedValue = Serial1.readStringUntil('\n');
   switch (speedValue[0])
   {
      case 'R':
        turn = 'R';
        break;
      case 'L':
```

```
        turn = 'L';
        break;
      default:
        turn = 'T';
        setpoint=speedValue.toDouble()/10 + 3.65;
        break;
    }
 }
```

Pri pisanju Arduino koda, korištene su dostupne biblioteke (eng. libraries). Za očitanje i obradu podataka sa žiroskopa i akcelerometra korišten je „jrowberg“ biblioteka, a Web poveznica na nju nalazi se u popisu literature pod rednim brojem [12.]. Inicijalizacija navedene biblioteke u Arduino kodu:

```
#include "MPU6050_6Axis_MotionApps20.h"
```

Dio koda za obradu podataka, koji se nalazi u glavnoj petlji:

```
mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
```

Uz očitane vrijednosti kvaterniona i gravitacije, dobiva se kut u radijanima. Kvaternioni pružaju drugačiji zapis položaja predmeta u prostoru. Budući da su podaci za položaj očitani u radijanima, vrši se pretvorba u stupnjeve:

```
input = ypr[2] * 180 / M_PI;
```

Gotove funkcije PID regulatora preuzete su iz biblioteke „Arduino PID Library“ autora Brett Beauregard. Također, Web poveznica spomenute biblioteke nalazi se u literaturi pod rednim brojem [13.]. Inicijalizacija se postiže ovako:

```
#include <PID_v1.h>
```

PID regulator definira se pomoću postavljanja ulaza (input) koji predstavlja kut nagiba vozila, postavljenog kuta (setpoint) u kojem želimo da vozilo bude nagnuto, te proporcionalnu, integralnu i derivacijsku konstantu (Kp, Ki i Kd).

```
PID pid(&input,&output,&setpoint,Kp,Ki,Kd,DIRECT);
```

Sljedećom naredbom provodi se izračun PID regulatora, te se dobiva vrijednost pokretanja motora (output).

```
pid.Compute();
```

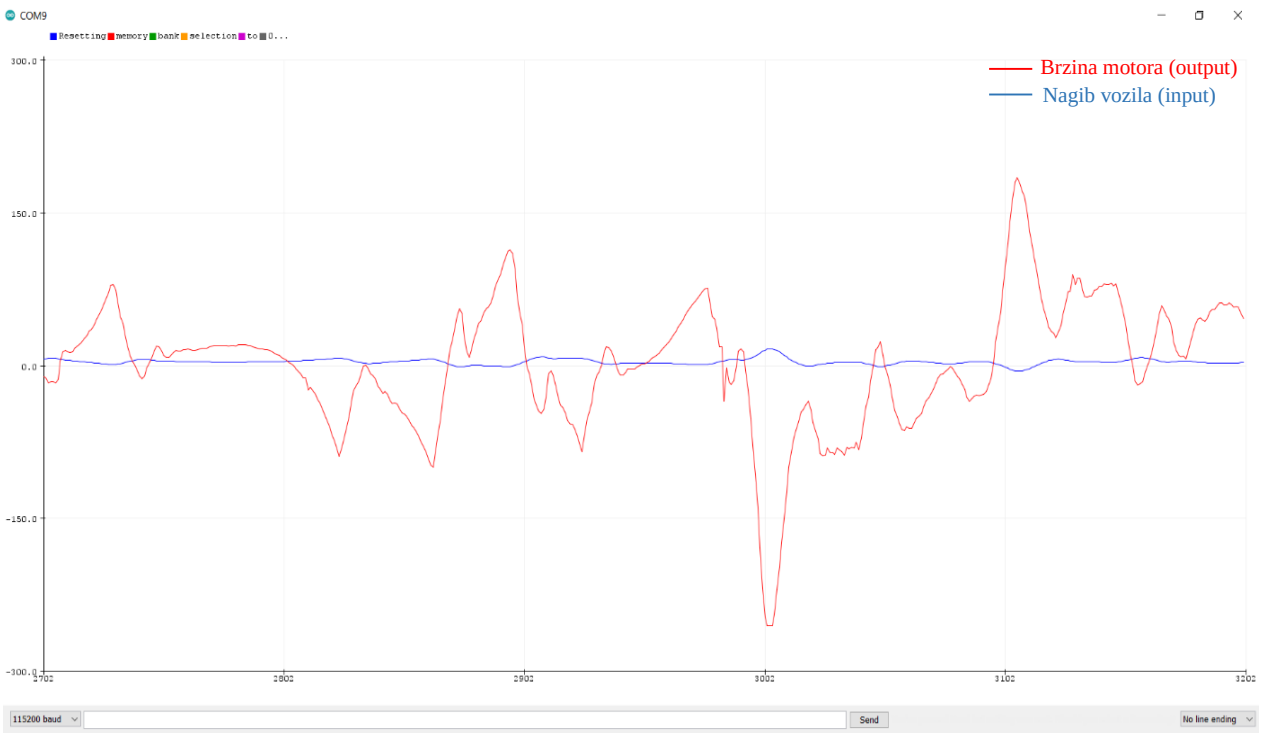
U slučaju kada otklon kuta nagiba (input) prijeđe vrijednost ±50°, motori se isključuju kako pri padu vozila ne bi bili i dalje aktivni. Time se ujedno i sprječavaju moguće dodatne štete. Ako je izlaz na motore pozitivan (output), to znači da robot pada prema naprijed, te se ulazi u funkciju za pomicanje robota u naprijed (Forward()), analogno tome, kada je izlaz na motore negativan, poziva se funkcija gibanja u nazad (Reverse()). To je prikazano u sljedećem dijelu koda.

```
if (input > -50 && input < 50)
 {
   if (output > 0)
     Forward();
   else if (output < 0)
     Reverse();
 }
 else
   Stop();
```

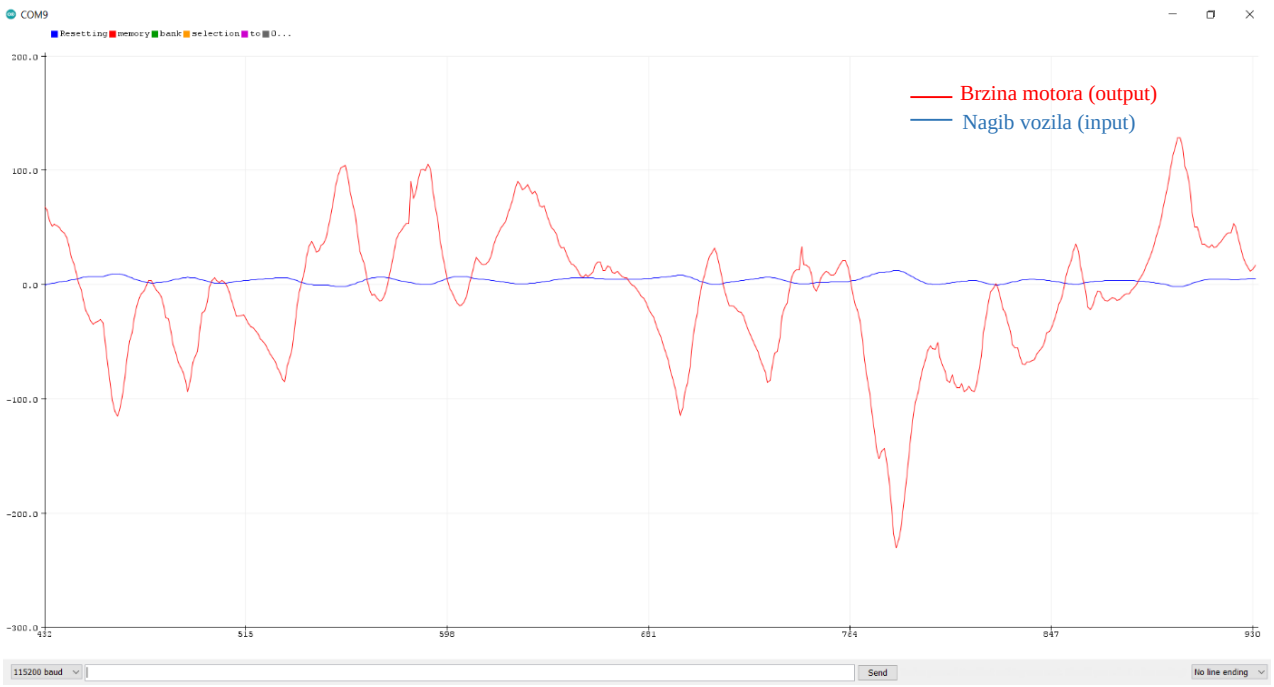
Kompletan Arduino kod nalazi se u prilogu.

4.4. Grafički prikaz vozila u radu

Prilikom izrade vozila, promatrao se odziv (output) na unesenu pobudu (input). Korištenjem Arduino IDE programskog paketa, grafički je prikazan rad vozila pri dostizanju ravnotežnog stanja uspravnog položaja. Promatranjem takvog prikaza pomoglo je pri namještanju i korekciji PID (Proporcialna, integralna i derivacijska) konstanti radi boljeg rada vozila. Na Slici 17 i 18 vidljivo je kako prilikom balansiranja vozila, odziv sustava koji se šalje motorima (output), nalazi se unutar granica ±150 i time se motori okreću relativno sporom brzinom radi korekcije nagiba vozila, međutim namjernim guranjem vozila u proizvoljnu stranu odziv poraste iznad tih granica kako bi motori osigurali vozilo od padanja. Maksimalna moguća vrijednost izračunatog „outputa“ je ±250, budući da je izlaz PID regulatora ograničen na te vrijednosti. Vrijednost „inputa“ (nagiba vozila) prikazana je u stupnjevima i označen je na grafu plavom bojom, a vrijednost „outputa“ odgovara izlaznoj kutnoj brzini motora i označen je crvenom bojom. Na Slici 17 i 18 os apscisa predstavlja broj primljenih i poslanih vrijednosti na mikrokontroleru (što se može gledati kao oznaka vremena), a os ordinata predstavlja vrijednosti „inputa“ i „outputa“.



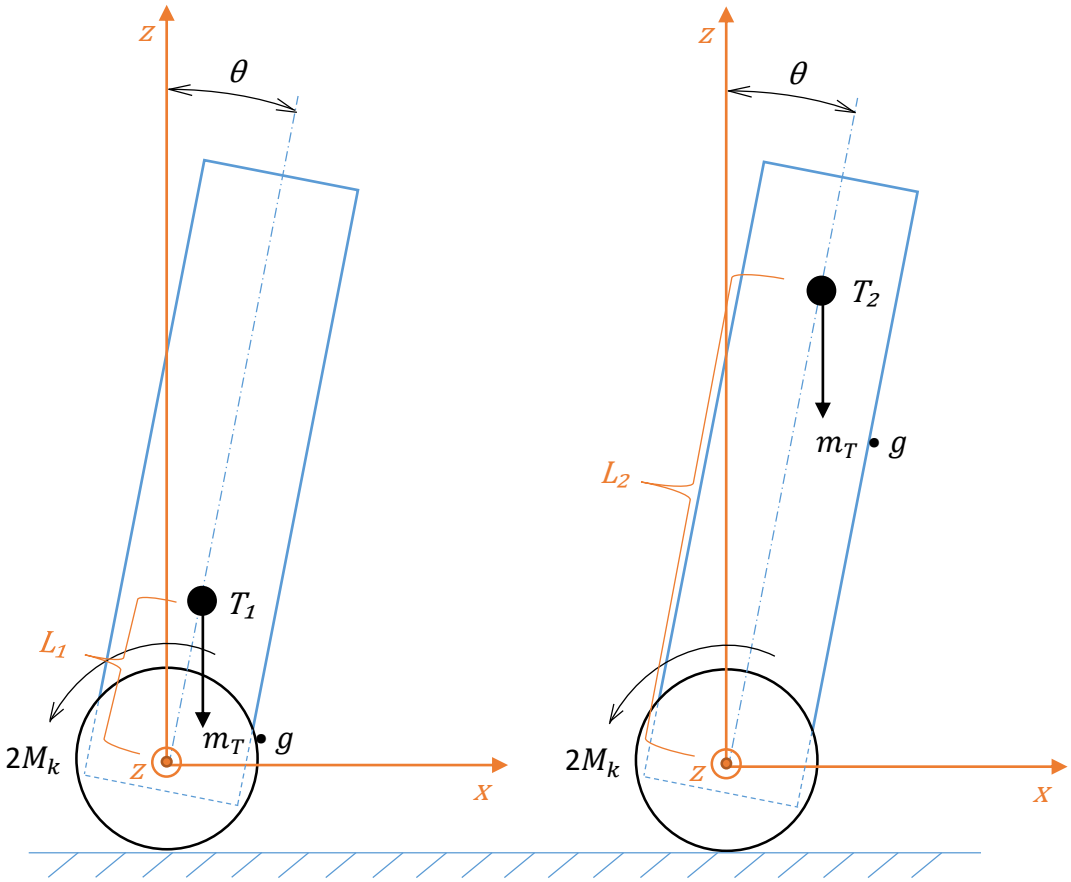
Slika 17. Grafički prikaz rada vozila (1)



Slika 18. Grafički prikaz rada vozila (2)

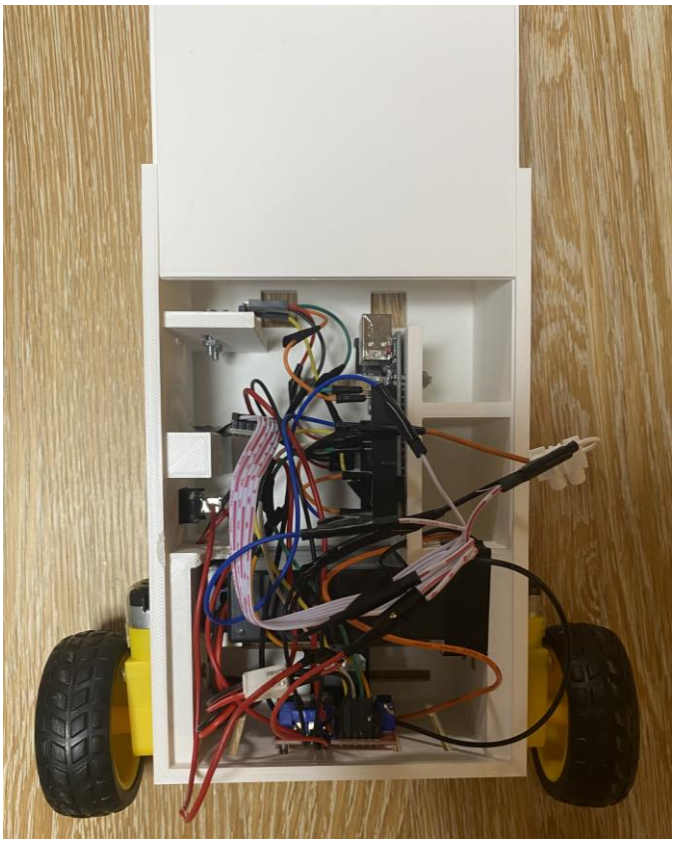
4.5. Sklopni prikaz vozila

Matematički model sustava je dio izrade koji pomaže pri razumijevanju i planiranju i na posljetku realizaciji projekta. Time dobivamo uvid koje sve sile i momente sustav mora savladati prilikom balansiranja. Bitno je zamijetiti kako sile i momenti koji utječu na vozilo nisu samo ovisni o vozilu, već i o mogućim smetnjama koja dolaze iz okoline. Zbog mogućih smetnji, kao na primjer, slučaj kada osoba namjerno gurne vozilo u određenom smjeru, cilj je staviti centar težišta malo iznad osi rotacije kotača. Ovim pozicioniranjem težišta T_1, smanjuje se utjecaj njegovog inercijskog djelovanja (postavljanjem težišta T_2 bliže gornjem vrhu vozila, utjecaj inercije težišta je veći), ali time se olakšava rad motorima. Pozicioniranjem težišta bliže motorima, moment koji motori moraju savladat je manji, ali time se zahtijevaju češće manje korekcije nagiba vozila. Razlika u pozicioniranju težišta vozila prikazana je na Slici 19.

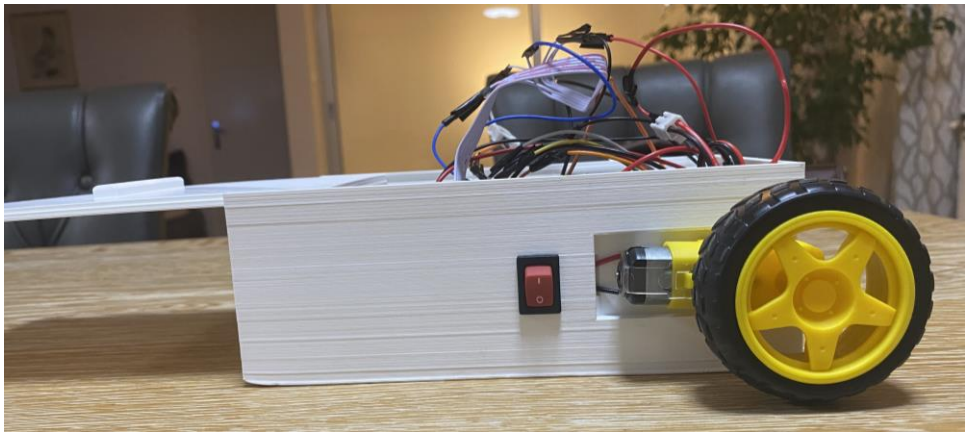


Slika 19. Usporedba pozicije težišta vozila

Slika 20. Prednji prikaz sklopljenog robota



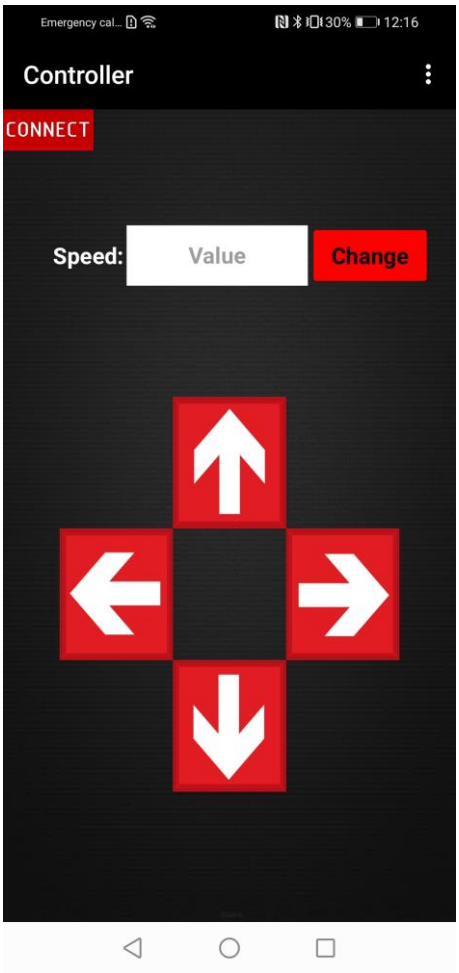
Slika 21. Stražnji prikaz sklopljenog robota



Slika 22. Prikaz robota s bokocrta

4.6. Aplikacija

Zbog rasprostranjenosti mobilnih uređaja, aplikacija je jednostavno primjenjiva i dostupna većini korisnika, te zamjenjuje potrebu za izradom specijalnog upravljača. Aplikacija je napravljena za Android operativni sustav. Koristeći Web programskog paketa „MIT App Inventor“ izrađena je ova mobilna aplikacija „Controller.apk“. Izgled mobilne aplikacije, odnosno korisničkog sučelja prikazan je na Slici 23.



Slika 23. Mobilna aplikacija

Kako bi se vozilo započelo balansirati, potrebno je pritisnuti tipku „CONNECT“ kojom se korisnik povezuje Bluetooth komunikacijom s vozilom. Uspješnim povezivanjem, vozilo kreće sa balansiranjem. Inicijalna vrijednost „Speed: Value“ je nula, što znači da vozilo želi postići uspravno ravnotežno stanje. Međutim, unosom proizvoljnog iznosa „Value“ u rasponu od -50 do +50,

te pritiskom tipke „CHANGE“ postavlja se „brzina“ mogućeg gibanja u naprijed ili u nazad pritiskom strelica smjera. Unosom željene vrijednosti „Speed: Value“, zapravo se mijenja centar željenog ravnotežnog stanja za kut koji poprima vrijednost:

$$Kut = \frac{Value}{10} \quad [°]$$

Iznos proizvoljne brzine „Value“ ne utječe na brzinu gibanja u stranu, već se vozilo orijentira na mjestu radi lakšeg manevriranja i kontrole u prostoru sa mogućim preprekama.

4.7. Procijenjena vrijednost

Ovom tablicom prikazana je vrijednost kupljenih komponenti i ukupni troškovi ovog projekta. Navedene cijene su okvirnog iznosa i razlikuju se ovisno o trgovini.

Tablica 1. Popis kupljenih komponenti

KOMPONENTA	CIJENA
1. Arduino Nano	70,00 kn
2. Motor Driver Dual H-Bridge L298N	45,00 kn
3. MPU 6050 akcelerometar i žiroskop	22,67 kn
4. HC-05 Bluetooth modul	37,80 kn
5. Straight BO Motor s kotačem x 2 kom.	30,00 kn
6. MP0.8-12 JST akumulator	94,50 kn
7. Prekidač	5,00 kn
8. 3D printano kućište (285 g PLA filamenta)	50,93 kn
9. Vijci i matice (M3)	5,00 kn
10. Žice	10,00 kn
Ukupno:	**370,90 kn**

5. Zaključak

Ovaj rad objedinjuje sve faze izrade projekta samobalansirajućeg robota na dva kotača. Pri izradi koristila su se znanja iz različitih kolegija koja su se pohađala tijekom preddiplomskog dijela studija smjera Mehatronike i robotike. Glavni dobitak ovog rada je praktična izrada robota. Korištenjem programskih paketa različite namjene i povezujući njihove rezultate ostvarena je cjelina rada. Ovaj rad je baza ili osnova na kojoj se može vršiti daljnja nadogradnja pri realizaciji samobalansirajućih vozila, a i šire. Praktični primjeri primjene su uređaji slični „Segwayu“ i igračke za djecu/adolescente.

Moguća su poboljšanja i nadogradnje u raznim segmentima. Na primjer, odabir kvalitetnijih DC motora i dodavanje enkodera na izlazno vratilo kako bi se znao točan pomak vozila, poboljšanja u definiranju PID konstanti za minimalno vrijeme odziva uz blagi prijelaz u stanje ravnoteže, te odabir snažnijeg mikroprocesora sa dvije jezgre i implementiranim Bluetooth modulom, itd. Izrada rada ostvarena je uz relativno mali trošak i to zbog odabira jeftinih komponenti, samostalnim dizajniranjem i 3D ispisu kućišta vozila.

Na kraju želim istaknuti da sam tijekom izrade ovog završnog rada proširio svoja znanja na području robotike i mehatronike, te me je dodatno zainteresiralo za daljnja proučavanja i praktičnu primjenu na tom području.

Web poveznica za video prikaz realiziranog samobalansirajućeg robota nalazi se u literaturi pod rednim brojem [17.].

6. Literatura

[1.] Tomašić, T., Demetlika, A., Crneković, M.: Self-balancing mobile robot tilter, Transactions of famena XXXVI-3, 2012.
[2.] Herceg – Rušec M., Završni rad, Samobalansirajuće vozilo, Zagreb, 2020.
[3.] https://www.enciklopedija.hr/Natuknica.aspx?ID=5446
[4.] Hrvatska opća enciklopedija, 1999., Deseti svezak, stranica od 187 do 188.
[5.] https://easyeda.com/
[6.] https://www.chipoteka.hr/
[7.] https://e-radionica.com/hr/
[8.] https://www.enciklopedija.hr/natuknica.aspx?ID=45689
[9.] https://www.omnitron.cz/_dokumenty/3082019111558627/mp08-12jst.pdf
[10.] https://studylib.net/doc/8882587/understanding-quaternions
[11.] https://arduino.stackexchange.com/search?q=arduino
[12.]https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050
[13.] https://github.com/br3ttb/Arduino-PID-Library/blob/master/PID_v1.h
[14.] https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/
[15.] https://create.arduino.cc/projecthub/eEdizon/arduino-self-balancing-robot-482cd7?ref=similar&ref_id=340883&offset=0
[16.] http://appinventor.mit.edu/
[17.] https://youtu.be/UZfp0C76He8

7. Prilozi

7.1. Arduino kod

```
// Self balancing robot Arduino code

#include <SoftwareSerial.h>  // For Bluetooth communication
#include "I2Cdev.h"
#include <PID_v1.h>  // PID library
#include "MPU6050_6Axis_MotionApps20.h"  // from jrowberg
library

MPU6050 mpu;

// HC-05
String speedValue;
char turn = 'T';
SoftwareSerial Serial1(0, 1); //rx tx
boolean BTconnected = false;
const byte BTpin = 12;  // STATE pin - Bluetooth

// MPU control
bool dmpReady = false;    // set true if DMP init was
successful
uint8_t mpuIntStatus;     // holds actual interrupt status byte
from MPU
uint8_t devStatus;        // return status after each device
operation (0 = success, !0 = error)
uint16_t packetSize;      // expected DMP packet size (default
is 42 bytes)
uint16_t fifoCount;       // count of all bytes currently in
FIFO
uint8_t fifoBuffer[64];  // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion
container
VectorFloat gravity;    // [x, y, z]            gravity vector
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll
container and gravity vector

// Tuned values for our specific self balancing robot
double setpoint = 3.65;  // center of stability
// PID values
double Kp = 14.2;  // Proportional
```

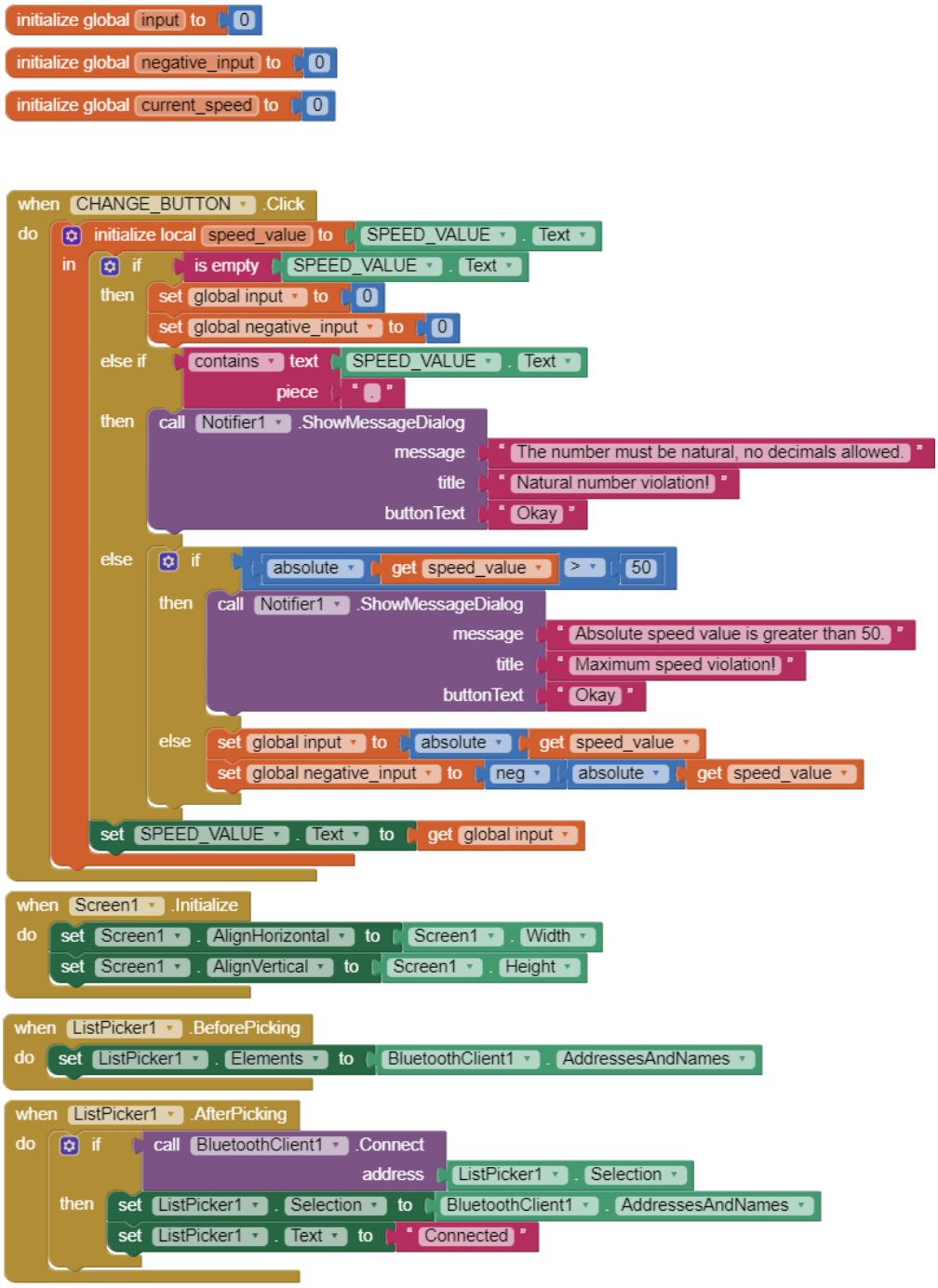
```
double Kd = 0.53;  // Derivative
double Ki = 100;  // Integral

double input, output;
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);

volatile bool mpuInterrupt = false;     // indicates whether
MPU interrupt pin has gone high
void dmpDataReady()
{
  mpuInterrupt = true;
}

void setup() {
  pinMode(BTpin, INPUT);
  while (!BTconnected)  // Bluetooth must be properly
connected in order to continue
  {
    if ( digitalRead(BTpin) == HIGH)  {
      BTconnected = true;  // Ready, continue with the program
    }
  }

  Serial1.begin(9600);  // Bluetooth serial communication

  mpu.initialize();  // initialize MPU device
  mpu.testConnection();  // verify connection
  devStatus = mpu.dmpInitialize();  // load and configure the
DMP

  // Gyro offsets, scaled for min sensitivity
  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(-85);
  mpu.setZAccelOffset(1688);

  // makes sure it worked (returns 0 if so)
  if (devStatus == 0)
  {
    mpu.setDMPEnabled(true);  // turn on the DMP, now that
it's ready
    attachInterrupt(0, dmpDataReady, RISING);  // enable
Arduino interrupt detection
    mpuIntStatus = mpu.getIntStatus();
    dmpReady = true;  // set "DMP Ready" flag so the main
loop() function knows it's okay to use it

    packetSize = mpu.dmpGetFIFOPacketSize();  // get expected
DMP packet size for later comparison
```

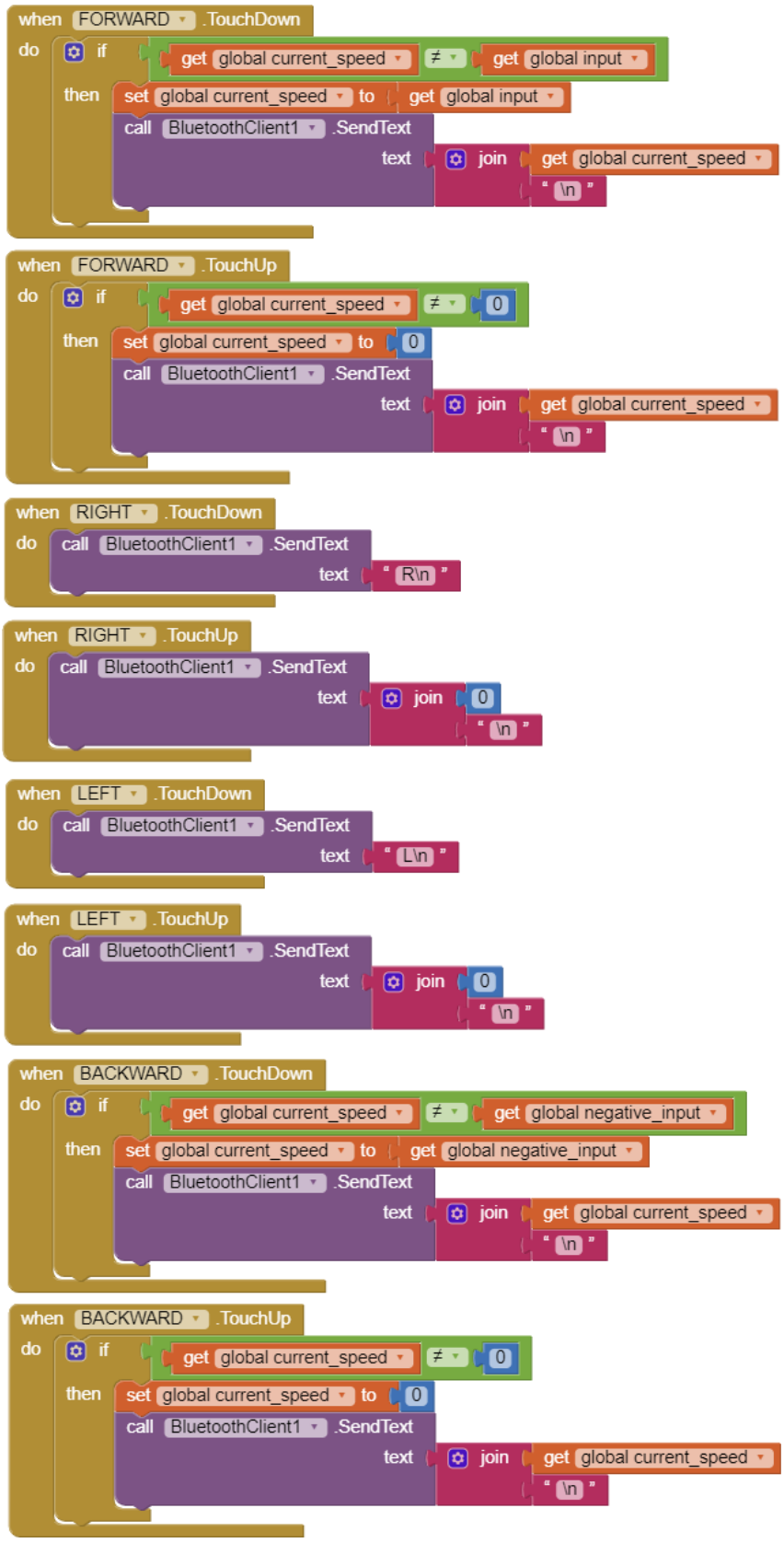
```
    //PID setup
    pid.SetMode(AUTOMATIC);
    pid.SetSampleTime(10);
    pid.SetOutputLimits(-255, 255);
  }

  // Initialize the DC motor output pins
  pinMode (6, OUTPUT);
  pinMode (9, OUTPUT);
  pinMode (10, OUTPUT);
  pinMode (11, OUTPUT);

  // By default turn off both the DC motors
  analogWrite(6, LOW);
  analogWrite(9, LOW);
  analogWrite(10, LOW);
  analogWrite(11, LOW);

}

void loop() {
  if (Serial1.available() > 0)  // If Bluetooth is
communicating
  {
    speedValue = Serial1.readStringUntil('\n');  // Reading
the Value until '\n'
    switch (speedValue[0])  // Read the first character from
the string
    {
      case 'R':
        turn = 'R';
        break;
      case 'L':
        turn = 'L';
        break;
      default:
        turn = 'T';
        setpoint = (speedValue.toDouble() / 10) + 3.65;  //
Added setpoint value for correct calibration
        break;
    }
  }

  if (!dmpReady) return;  // if programming failed, don't try
to do anything

  while (!mpuInterrupt && fifoCount < packetSize)  // wait for
MPU interrupt or extra packet(s) available
  {
```

```
    pid.Compute();  // perform PID calculations and output to
motors

    if (input > -50 && input < 50)  // Robot is tilting
    {
      if (output > 0)  // Falling towards front
        Forward();  // Rotate the wheels forward
      else if (output < 0)  // Falling towards back
        Reverse();  // Rotate the wheels backward
    }
    else  // Center point
      Stop();  // Hold the wheels still
  }

  mpuInterrupt = false;  // reset interrupt flag and get
INT_STATUS byte
  mpuIntStatus = mpu.getIntStatus();

  fifoCount = mpu.getFIFOCount();  // get current FIFO count

  if ((mpuIntStatus & 0x10) || fifoCount == 1024)  // check
for overflow
  {
    mpu.resetFIFO();  // reset and continue
  }
  else if (mpuIntStatus & 0x02)  // check for DMP data ready
interrupt
  {
    while (fifoCount < packetSize) fifoCount =
mpu.getFIFOCount();  // wait for correct available data length
    mpu.getFIFOBytes(fifoBuffer, packetSize);  // read a
packet from FIFO

    fifoCount -= packetSize;  // track FIFO count here in case
there is > 1 packet available

    mpu.dmpGetQuaternion(&q, fifoBuffer);  // get value for q
    mpu.dmpGetGravity(&gravity, &q);  // get value for gravity
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);  // get value
for ypr (yaw, pitch, roll)

    input = ypr[2] * 180 / M_PI;  // Value in degrees

  }
}


void Forward() //Code to rotate the wheels forward
{
  if (turn == 'R')  // Rotate right
```

```
  {
    analogWrite(6, output);
    analogWrite(10, 0);
  }
  else if (turn == 'L')  // Rotate left
  {
    analogWrite(6, 0);
    analogWrite(10, output);

  }
  else  // Drive forward
  {
    analogWrite(6, output);
    analogWrite(10, output);

  }
  analogWrite(9, 0);
  analogWrite(11, 0);
}

void Reverse()  //Rotate the wheels backwards
{
  if (turn == 'R')  // Rotate right
  {
    analogWrite(11, output * -1);
    analogWrite(9, 0);
  }
  else if (turn == 'L')  // Rotate left
  {
    analogWrite(11, 0);
    analogWrite(9, output * -1);
  }
  else  // Drive backwards
  {
    analogWrite(9, output * -1);
    analogWrite(11, output * -1);

  }
  analogWrite(6, 0);
  analogWrite(10, 0);
}

void Stop()  // Stop both wheels
{
  analogWrite(6, 0);
  analogWrite(9, 0);
  analogWrite(10, 0);
  analogWrite(11, 0);
}
```

7.2. Kod aplikacije u blok prikazu




when FORWARD .TouchDown
do if get global current_speed ≠ get global input
then set global current_speed to get global input
call BluetoothClient1 .SendText
text join get global current_speed
" \n "
when FORWARD .TouchUp
do if get global current_speed ≠ 0
then set global current_speed to 0
call BluetoothClient1 .SendText
text join get global current_speed
" \n "
when RIGHT .TouchDown
do call BluetoothClient1 .SendText
text " R\n "
when RIGHT .TouchUp
do call BluetoothClient1 .SendText
text join 0
" \n "
when LEFT .TouchDown
do call BluetoothClient1 .SendText
text " L\n "
when LEFT .TouchUp
do call BluetoothClient1 .SendText
text join 0
" \n "
when BACKWARD .TouchDown
do if get global current_speed ≠ get global negative_input
then set global current_speed to get global negative_input
call BluetoothClient1 .SendText
text join get global current_speed
" \n "
when BACKWARD .TouchUp
do if get global current_speed ≠ 0
then set global current_speed to 0
call BluetoothClient1 .SendText
text join get global current_speed
" \n "