Sveučilište u Zagrebu
Fakultet strojarstva i brodogradnje

Tomislav Tomašić, Andrea Demetlika

SAMOBALANSIRAJUĆI MOBILNI ROBOT

Zagreb, 2011.

Ovaj rad izrađen je na Fakultetu strojarstva i brodogradnje, Zavodu za robotiku i automatizaciju proizvodnih sustava, pod vodstvom prof. dr. sc. Mladena Crnekovića i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2010./2011.

Sadržaj

Popis slika

Popis oznaka

Oznaka	Opis	Jedinica
$\mathbf{b}$	Koeficijent otpora gibanju,	Ns/m
$\mathbf{F_H}$	Horizontalna sila između kotača i osovine motora,	N
$\mathbf{F_V}$	Vertikalna sila između kotača i osovine motora,	N
$\mathbf{F_{PH}}$	Sila trenja,	N
$\mathbf{F_{PV}}$	Vertikalna sila podloge na kotač,	N
$\mathbf{g}$	Zemljina gravitacija,	m/s^2
$\mathbf{J_k}$	Moment inercije kotača oko osi rotacije,	kgm^2
$\mathbf{J_n}$	Moment inercije njihala oko osi rotacije,	kgm^2
$\mathbf{K_a}$	Pojačanje armature,	A/V
$\mathbf{K_{ch}}$	Pojačanje choppera,	V/V
$\mathbf{K_e}$	Pojačanje EMS,	Vs/rad
$\mathbf{K_m}$	Pojačanje momenta,	Nm/A
$\mathbf{K_r}$	Pojačanje redukcije,	rad/rad
$\mathbf{K_{rf}}$	Pojačanje redukcije i gubitaka trenja,	Nm/Nm
$\mathbf{L}$	Udaljenost od z-osi do težišta robota,	m
$\mathbf{m_k}$	Masa kotača,	kg
$\mathbf{m_n}$	Masa njihala,	kg
$\mathbf{M_M}$	Moment motora,	Nm
$\mathbf{r}$	Polumjer kotača,	m
$\mathbf{x}$	Pomak po x-osi,	m
$\dot{\mathbf{x}}$	Brzina po x-osi,	m/s
$\ddot{\mathbf{x}}$	Ubrzanje po x-osi,	m/s^2
$\mathbf{x_n}$	Pomak težišta njihala po x-osi u odnosu na os rotacije kotača,	m
$\mathbf{y}$	Pomak po y-osi,	m

$\dot{\mathbf{y}}$	Brzina po y-osi,	m/s
$\ddot{\mathbf{y}}$	Ubrzanje po y-osi,	m/s^2
$\mathbf{y_n}$	Pomak težišta njihala po y-osi u odnosu na os rotacije kotača,	m
$\mathbf{\Theta_n}$	Kut nagiba njihla,	rad
$\dot{\mathbf{\Theta}}_\mathbf{n}$	Kutna brzina njihla,	rad/s
$\ddot{\mathbf{\Theta}}_\mathbf{n}$	Kutna akceleracija njihala,	rad/s^2
$\ddot{\mathbf{\Theta}}_\mathbf{k}$	Kutna akceleracija kotača	rad/s^2

Poglavlje 1

Uvod

U ovom radu opisan je postupak izrade samobalansirajućeg mobilnog robota na dva kotača. Konstrukcija robota je sama po sebi nestabilna i teži prevrtanju oko osi rotacije kotača. Djelovanjem motora robot se pokreće u odgovarajući smjer i time vraća u uspravni položaj. Princip rada robota odgovara onome kod popularnog električnog vozila tvrtke Segway.

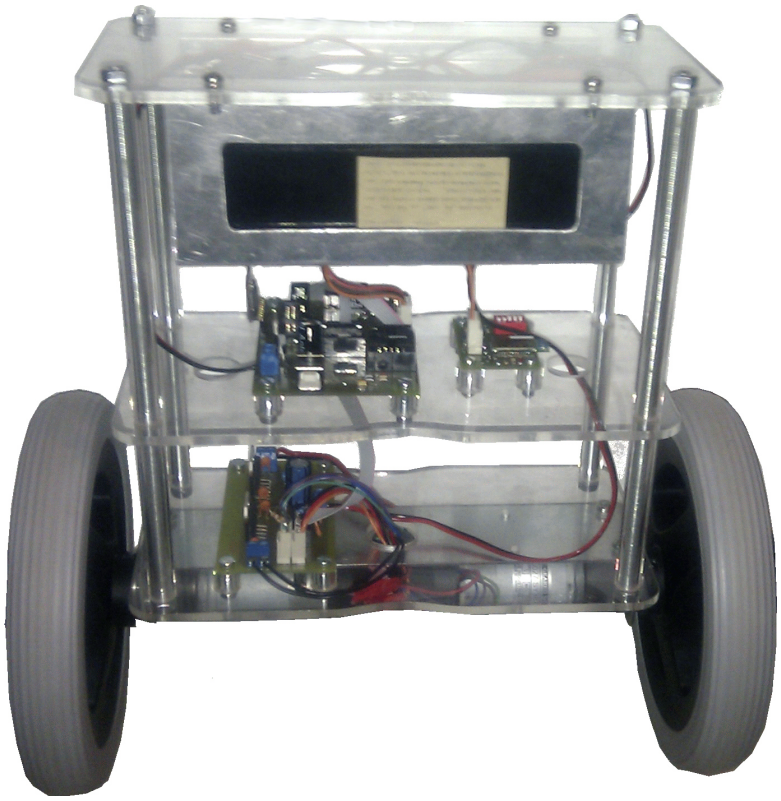
Izrada ovog projekta obuhvaća rješavanje širokog spektra zadataka; projektiranje i izradu mehaničke konstrukcije i tiskane pločice, simulaciju matematičkog modela, implementaciju regulacijskog algoritma u mikrokontroler, te realizaciju komunikacije robota s računalom i mobitelom. Upravo ta integracija strojarskih, elektroničkih, automatičarskih i informatičkih disciplina, značajna za područje mehatronike, glavni je razlog odabira ovog projekta. Samobalansirajući mobilni robot predstavlja dobru platformu za projektiranje i ispitivanje naprednih regulacijskih i estimacijskih algoritama, te kao takav može se koristiti za daljnja znanstvena istraživanja.

Projekt je započet izradom matematičkog modela dinamike robota i motora, njegove linearizacije, i modeliranja u *Simulinku*. Kako bi se bolje mogla vizualizirati dinamika i stabilnost robota, napravljena je veza *Simulink* modela sa 3D modelom u *Matlab-ovom* alatu VRML.

Da bi se na kotače djelovalo pravilnim iznosom momenta potrebna je točna informacija o trenutnom kutu nagiba. Ona se dobiva kombiniranjem zašumljenih izlaza senzora akcelerometra i žiroskopa pomoću Kalmanovog filtra.

Kako bi se proučilo što više regulacijskih metoda, projektirano je nekoliko vrsta regulacije. Prvi je PID regulator koji regulira sustav samo po kutu nagiba robota, a glavni nedostatak je da ne regulira položaj. Dalje je izveden LQR regulator koji vrši regulaciju po kutu i po

položaju, uz pretpostavku da su točno poznata sva stanja sustava. Nakon toga spajanjem LQR regulatora sa estimatorom stanja (Kalmanovim filtarom) dobiven je LQG regulator koji regulira sustav u kojem, uz robota i motore, postoji i žašumljeno djelovanje senzora. Osim navedenih napravljena je i regulacija neizrazitim (engl. *fuzzy*) regulatorom.



Slika 1.1: *Samobalansirajući mobilni robot na dva kotača*

Mehanička konstrukcija sastavljena je od tri pleksiglas ploče od kojih svaka nosi jedan dio konstrukcije (motore, baterije, ...). Svaki dio konstrukcije modeliran je u CAD alatu i kasnije izrađen.

Algoritam regulacije, zajedno sa ostalim procesima, nalazi se u mikrokontroleru smještenom na upravljačkoj jedinici. Upravljačka jedinica preko PWM signala upravlja pločicom koja ima funkciju motor kontrolera, čime je odvojen energetski od upravljačkog dijela.

Upravljanje robota izvedeno je preko bluetooth komunikacije. Napravljena je mogućnost očitavanja telemetrije i upravljanja koristeći komunikaciju između robota i računala, te robota i mobitela.

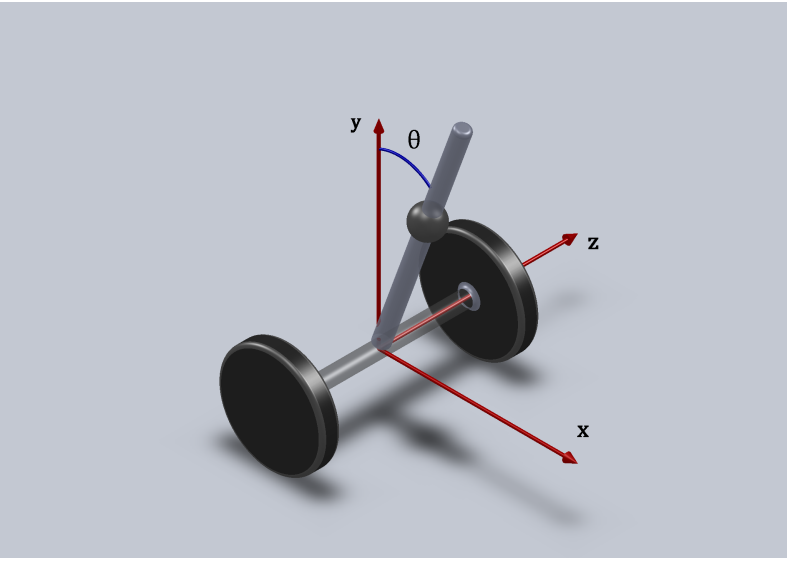
Poglavlje 2

Matematički model

U ovom poglavlju izveden je matematički model samo-balansirajućeg mobilnog robota te provedena je linearizacija sustava u svrhu projektiranja regulatora. Daljnja izlaganja u ovom poglavlju temelje se na referencama [1, 2, 3] u kojima je ova tema detaljno razmatrana.

2.1 Izvod matematičkog modela

Kod izvođanja matematičkog modela mobilni robot se u mislima rastavi na dva dijela; kotače i ostalu konstrukciju (njihalo). Zatim se za svaki dio postavlja suma sila i momenata oko određene točke.



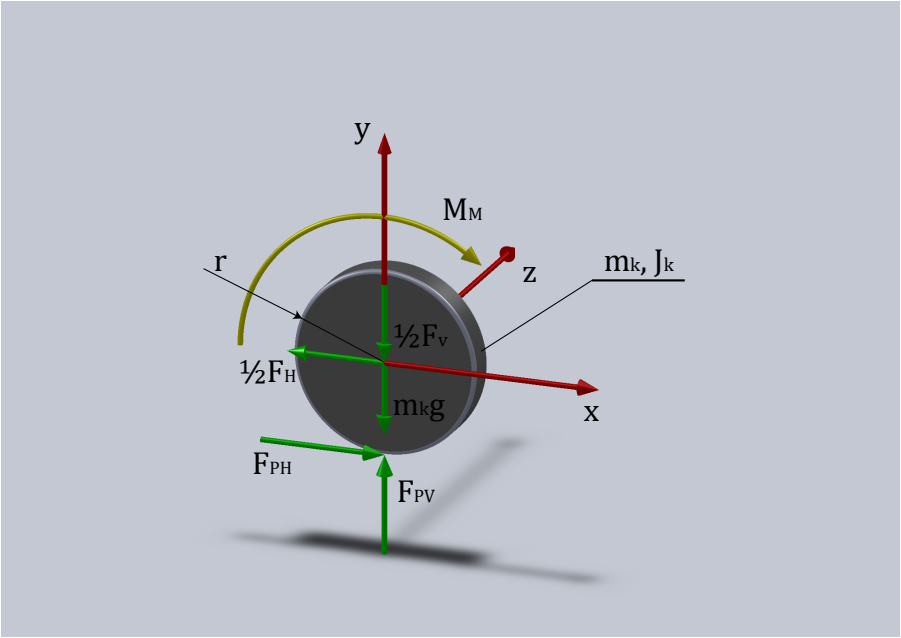
Slika 2.1: *Pojednostavljeni prikaz mobilnog robota*

Jednadžbe za kotač:

$$\sum F_x = 0: \qquad m_k\ddot{x} = -\frac{1}{2}F_H + F_{PH} - b\dot{x}, \tag{2.1}$$

$$\sum F_y = 0: \qquad m_k\ddot{y} = F_{PV} - \frac{1}{2}F_V - m_k g, \tag{2.2}$$

$$\sum M_T = 0: \qquad J_k\ddot{\Theta}_k = M_M - F_{PH}r. \tag{2.3}$$



Slika 2.2: *Sile i momenti na kotačima*

Pomoću sljedećih jednadžbi izrazimo kutno ubrzanje kotača $\ddot{\Theta}_k$:

$$x = r\Theta_k, \tag{2.4}$$

$$\dot{x} = r\dot{\Theta}_k, \tag{2.5}$$

$$\ddot{x} = r\ddot{\Theta}_k. \tag{2.6}$$

Iz (2.6) slijedi

$$\ddot{\Theta}_k = \frac{\ddot{x}}{r}, \tag{2.7}$$

te se uvrsti u (2.3)

$$J_k\frac{\ddot{x}}{r} = M_M - F_{PH}r, \tag{2.8}$$

nakon čega se dobije izraz za silu F_{PH}

$$F_{PH} = \frac{M_M}{r} - \frac{J_k}{r^2}\ddot{x}. \tag{2.9}$$

Kad se (2.9) uvrsti u (2.1) i (2.3) i izluči dobije se:

$$(m_k + \frac{J_k}{r^2})\ddot{x} = \frac{M_M}{r} - \frac{1}{2}F_H - b\dot{x}, \tag{2.10}$$

$$J_n\ddot{\Theta}_n = -F_H L\cos\Theta_n + F_V L\sin\Theta_n - 2M_M. \tag{2.11}$$

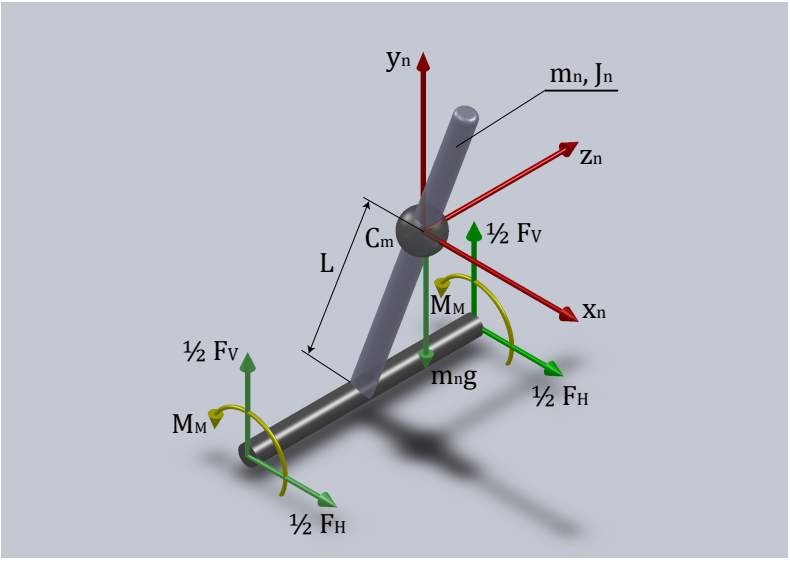
Nadalje treba pronaći sile F_H i F_V, što se dobije iz jednadžba sila za njihalo:

$$\sum F_x = 0: \qquad m_n\ddot{x}_n = F_H, \tag{2.12}$$

$$\sum F_y = 0: \qquad m_n\ddot{y}_n = F_V - m_n g, \tag{2.13}$$

te izlučivanjem F_V slijedi

$$F_V = m_n\ddot{y}_n + m_n g. \tag{2.14}$$



Slika 2.3: *Sile i momenti njihala*

Zatim je potrebno pronaći vezu pomaka kotača i težišta njihala s kutom nagiba robota, kako bi dobili potrebne sile F_V i F_H. Iz slike 2.1 može se zaključiti da je:

$$x_n = x + L\sin\Theta_n, \tag{2.15}$$

čijim dvostrukim deriviranjem dobijemo željenu vezu varijabli

$$\dot{x}_n = \dot{x} + L\cos\Theta_n\dot{\Theta}_n, \tag{2.16}$$

$$\ddot{x}_n = \ddot{x} + L\cos\Theta_n\ddot{\Theta}_n - L\sin\Theta_n\dot{\Theta}_n^2. \tag{2.17}$$

Potom se isti postupak sprovede za pomake s obzirom na y os:

$$y_n = L\cos\Theta_n, \tag{2.18}$$

$$\dot{y}_n = -L\sin\Theta_n\dot{\Theta}_n, \tag{2.19}$$

$$\ddot{y}_n = -L\sin\Theta_n\ddot{\Theta}_n - L\cos\Theta_n.\dot{\Theta}_n^2. \tag{2.20}$$

Uvrštavanjem (2.17) u (2.12) i (2.20) u (2.14) dobiju se izrazi za silu F_H i F_V:

$$F_H = m_n(\ddot{x} + L\cos\Theta_n\ddot{\Theta}_n - L\sin\Theta_n\dot{\Theta}_n^2), \tag{2.21}$$

$$F_V = m_n(-L\sin\Theta_n\ddot{\Theta}_n - L\cos\Theta_n\dot{\Theta}_n^2 + g). \tag{2.22}$$

Zatim se ovi izrazi za sile F_H i F_V uvrste u (2.10) i (2.11), te sređivanjem slijedi:

$$(J_n + L^2m_n)\ddot{\Theta}_n = -Lm_n\ddot{x}\cos\Theta + gLm_n\sin\Theta - 2M_M, \tag{2.23}$$

$$(m_k + \frac{J_k}{R^2} + \frac{1}{2}m_n)\ddot{x} = \frac{M_M}{R} - \frac{1}{2}m_nLcos\Theta\ddot{\Theta} + \frac{1}{2}m_nLsin\Theta\dot{\Theta}^2 - b\dot{x}. \tag{2.24}$$

2.2 Linearizacija modela

Linearizacijom se dobije linearni model sustava koji za vrijednosti kuta oko radne točke odgovara nelinearnom modelu, a njegovom primjenom se višestruko pojednostavlja postupak sinteze regulatora. Za radnu točku odabirana je vrijednost kuta nagiba $\Theta = 0$.

U tom slučaju vrijede jednadžbe:

$$sin\Theta = \Theta, \tag{2.25}$$

$$cos\Theta = 1, \tag{2.26}$$

$$\dot{\Theta}^2 = 0, \tag{2.27}$$

i njihovim uvrštavanjem u (2.23) i (2.24) slijedi

$$\ddot{x} = \frac{1}{m_k + \frac{J_k}{R^2} + \frac{m_n}{2}}(\frac{M_M}{R} - b\dot{x} - \frac{m_n}{2}L\ddot{\Theta}), \tag{2.28}$$

$$\ddot{\Theta} = \frac{-2M_n - LM_n\ddot{x} + gLM_n\Theta}{J_n + L^2m_n}. \tag{2.29}$$

Uvrštavanjem tih jednadžbi jednu u drugu i njihovim sređivanjem dobije se konačni oblik

$$\ddot{x} = a_{22}\dot{x} - a_{23}\Theta + b_2u, \tag{2.30}$$

$$\ddot{\Theta} = a_{42}\dot{x} - a_{43}\Theta + b_4u, \tag{2.31}$$

gdje su a_{22}, a_{23}, a_{42}, a_{43} koeficijenti matrice sustava, a b_2, b_4 koeficijenti matrice ulaza sustava:

$$a_{22} = \frac{2\{J_n(K_aK_eK_rK_mK_{rf} + bR^2) + Lm_n[bLR^2 + K_aK_eK_rK_mK_{rf}(L + R)]\}}{2J_k(J_n + L^2m_n) + [2L^2m_km_n + J_n(2m_k + m_n)]R^2}, \tag{2.32}$$

$$a_{23} = -\frac{gL^2m_n^2R^2}{J_k(2J_n + 2L^2m_n) + [2L^2m_km_n + J_n(2m_k + m_n)]R^2}, \tag{2.33}$$

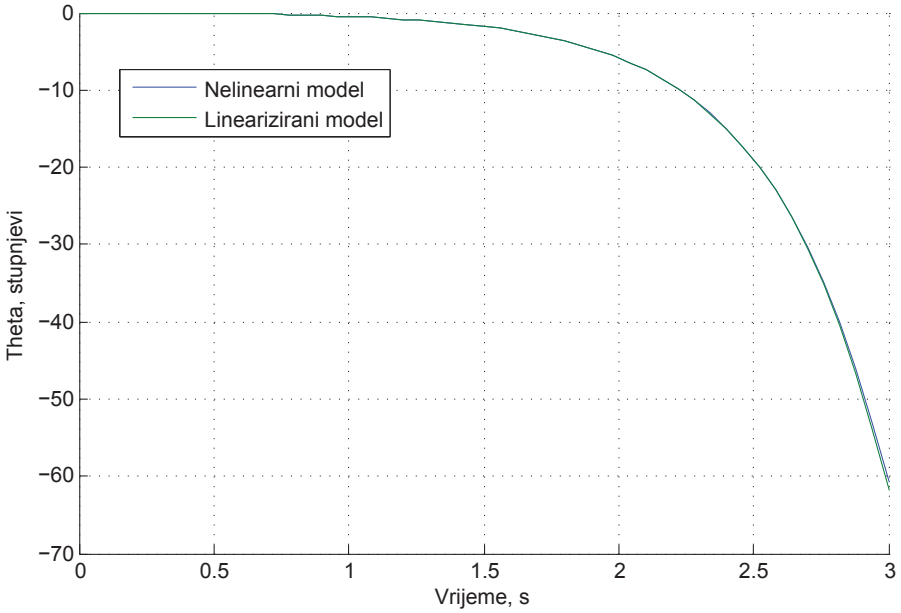
$$a_{42} = \frac{2\{2J_k K_a K_e K_r K_m K_{rf} + R[bLm_n R^2 + K_a K_e K_r K_m K_{rf}(Lm_n + 2m_k R + m_n R)]\}}{R\{2J_k(J_n + L^2 m_n) + [2L^2 m_k m_n + J_n(2m_k + m_n)]R^2\}}, \tag{2.34}$$

$$a_{43} = \frac{gLm_n}{J_n + L^2 m_n} + \frac{gL^3 m_n^3}{2(J_n + L^2 m_n)^2 (m_k + \frac{m_n}{2} - \frac{L^2 m_n^2}{2(J_n + L^2 m_n)} + \frac{J_k}{R^2})}, \tag{2.35}$$

$$b_2 = \frac{2K_a K_{ch} K_m K_{rf} R[J_n + Lm_n(L + R)]}{2J_k(J_n + L^2 m_n) + [2L^2 m_k m_n + J_n(2m_k + m_n)]R^2}, \tag{2.36}$$

$$b_4 = -\frac{2K_a K_{ch} K_m K_{rf}\{2J_k + R[Lm_n + (2m_k + m_n)R]\}}{2J_k(J_n + L^2 m_n) + [2L^2 m_k m_n + J_n(2m_k + m_n)]R^2}. \tag{2.37}$$

Usporedba lineariziranog modela i nelinearnog pokazuje da linearizirani model dobro opisuje ponašanje sustava za male kuteve, što je vidljivo na grafu 2.4.



Slika 2.4: *Usporedba odaziva linearizirano i nelinearizirano sustava na jedinični poremećaj*

Raspis diferencijalnih jednadžbi u obliku prostora stanja:

$$\dot{x} = Ax + Bu \quad , \tag{2.38}$$

$$y = Cx + Du \quad . \tag{2.39}$$

x - vektor stanja

$\dot{x}$ - vektor derivacija varijabli stanja

u - vektor ulaza

y - vektor izlaza

A - matrica koeficijenata sustava

B - matrica ulaza sustava

C - matrica izlaza sustava

D - matrica direktnog preslikavanja ulaza na izlaz

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\Theta} \\ \ddot{\Theta} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & a_{42} & a_{43} & 0 \end{bmatrix}}_{A} \begin{bmatrix} x \\ \dot{x} \\ \Theta \\ \dot{\Theta} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ b_2 \\ 0 \\ b_4 \end{bmatrix}}_{B} u, \tag{2.40}$$

$$\begin{bmatrix} x \\ \Theta \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{C} \begin{bmatrix} x \\ \dot{x} \\ \Theta \\ \dot{\Theta} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{D} u. \tag{2.41}$$

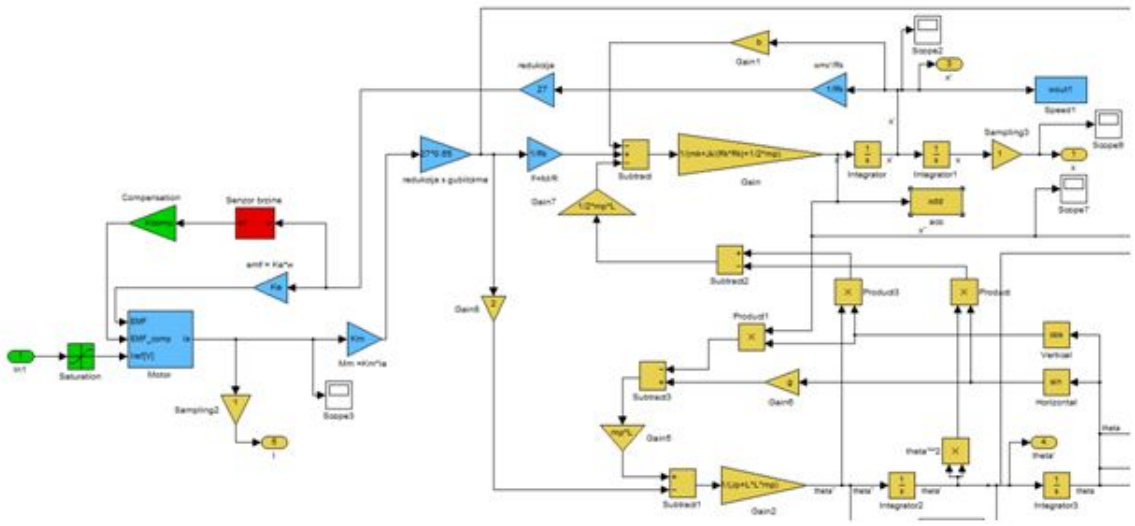
Poglavlje 3

Simulacija modela

3.1 Simulink model

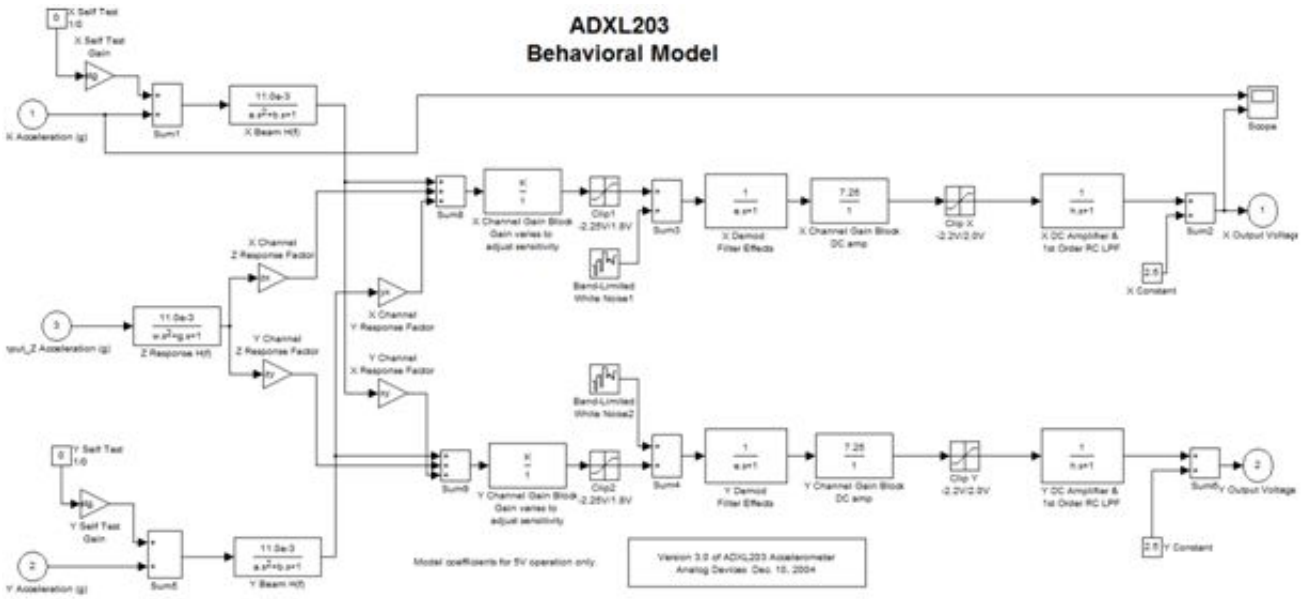
Jednadžbe dobivene izvodom sadrže nelinearne funkcije, pa ih je potrebno modelirati u *Matlab-ovom* alatu za simuliranje i izradu modela - *Simulink* [4], kako bi se dobila točnija dinamika sustava.

Radi preglednosti, u izrađenom modelu blokovi koji opisuju dinamiku pojedinih struktura obojani su različitim bojama. Pa su prema tome blokovi vezani za dinamiku motora obojani plavom, blokovi vezani za dinamiku njihala žutom, blokovi senzora crvenom, te blokovi koji se odnose na obradu u mikrokontroleru zelenom bojom.



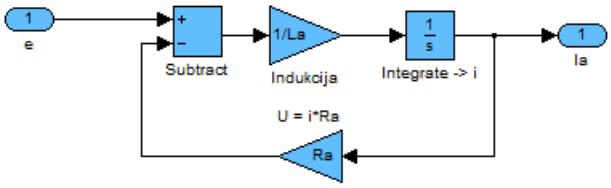
Slika 3.1: *Model njihala motora i senzora*

Model dinamike dvo-osnog akcelerometra tvrtke Analog Devices, ADX203, koji će se koristiti na stvarnom robotu preuzet je sa stranice proizvođača [5]. Da bi se dobila vrijednost akceleracije koju on mjeri potrebno je prvo transformirati vrijednost akceleracije kotača i kuta nagiba njihala u akceleraciju po pojedinim osima zakrenutog akcelerometra i tek te vrijednosti postaviti na ulaz u model akcelerometra.



Slika 3.2: *Model akcelerometra*

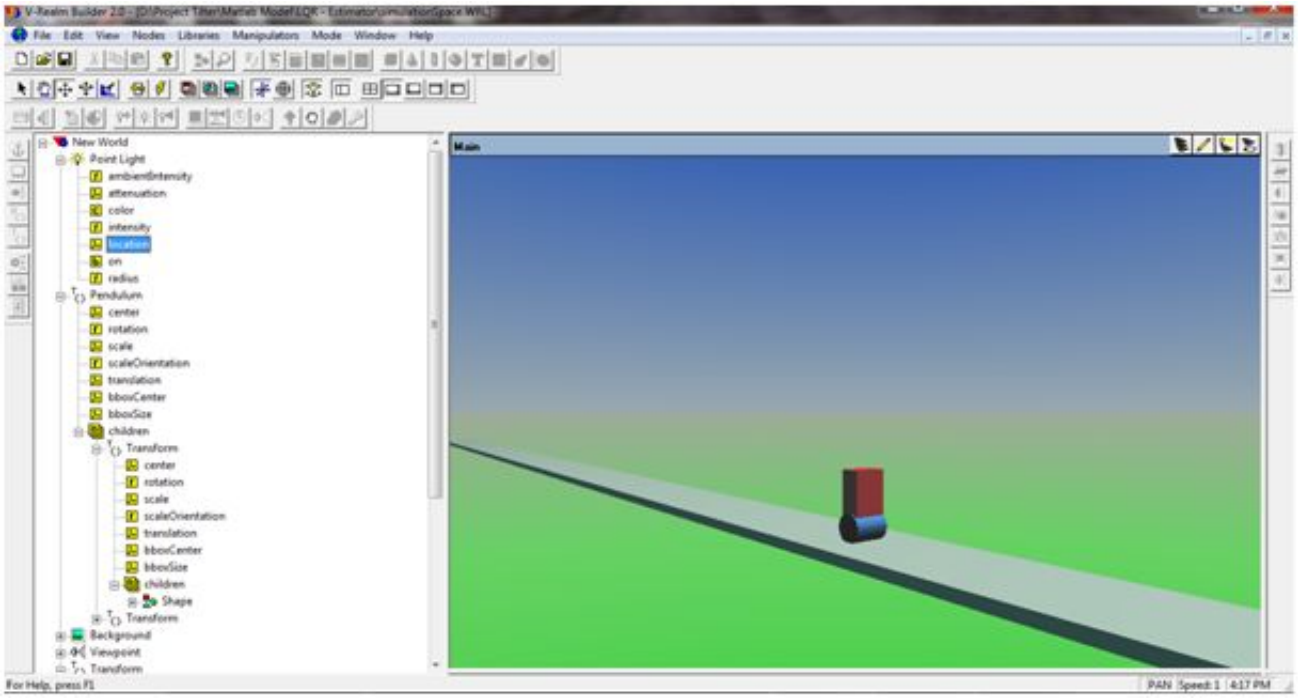
Parametri modela motora odabrani su prema vrijednostima stvarnih kupljenih motora.



Slika 3.3: *Model DC motora u Simulinku*

3.2 VRML simulacija

Kako bi se bolje vizualiziralo gibanje robota izrađen je 3D model sustava u *Matlab-ovom* toolbox-u *VRML (Virtual Reality Modeling Language)*. Za izradu modela korišten je alat *V-Realm Builder* [6] u kojem je definirana pojednostavljena geometrija robota, podloga po kojoj se giba, te pozadina i osvjetljenje. Pozicija i kut nagiba *VRML* modela robota



Slika 3.4: *Alat za izradu VRML modela - V-Realm Builder*

povezani su sa *Simulink* modelom koristeći blokove *Simulink 3D Animator Toolbox-a*. Time se tijekom izvođenja simulacije u 3D okružju može vidjeti točno kretanje i nagib robota.

Poglavlje 4

Projektiranje regulatora

4.1 Osnovni pojmovi

4.1.1 Optimalno upravljanje

U optimalnom upravljanju cilj je projektirati regulator koji daje najbolje moguće performanse u odnosu na neki zadani kriterij [7, 8]. Primjer toga je regulator koji koristi najmanju moguću količinu energije regulacijskog signala da bi doveo izlaz u nulu. U tom slučaju mjera uspješnosti (kriterij optimalnosti) je korištena energija regulacijskog signala.

Optimalnost u odnosu na neki kriterij nije jedina tražena osobina regulatora, mogla bi se također tražiti i stabilnost u zatvorenoj petlji, određeni prebačaj i vrijeme smirivanja, robusnost na zanemarenu dinamiku itd.

LQR i LQG regulatori optimalni su po kriteriju energije. Oni su posebno zanimljivi jer postupak minimiziranja energije automatski daje regulatore koji su stabilni i donekle robusni. Zapravo, regulatori dobiveni ovim postupkom su tako dobri da se koriste čak i kad optimiziranje energije nije traženo. Osim toga, ovaj postupak je primjenjiv na sustave sa više ulaza i izlaza (MIMO sustave) na koje je teško primijeniti klasične modele regulatora.

4.1.2 Upravljivost sustava (engl. *controllability*)

Sustav je upravljiv (engl. *controlable*) ako postoji upravljački vektor u(t) koji može prevesti sustav iz nekog početnog stanja x0(t) u ishodište prostora stanja unutar konačnog vremena t. Upravljivost sustava se provjerava preko Kalman-ovog uvjeta upravljivosti koji

kaže da je sustav upravljiv ako matrica:

$$C = \begin{bmatrix} B & AB & A^2B & ... & A^{n-1}B \end{bmatrix} \tag{4.1}$$

sadržava toliko linearno neovisnih vektora stupaca koliko ima varijabli stanja. U *Matlab-u* se upravljivost sustava provjerava koristeći funkcije *ctrb(A, B)* i *rank(ans)*.

U slučaju robota naredba

```
C=ctrb(A,B)
```

daje matricu:

$$\begin{bmatrix} 0 & 10 & -110 & 2250 \\ 10 & -110 & 2250 & -46230 \\ 0 & -10 & 280 & -5790 \\ -10 & 280 & -5790 & 119170 \end{bmatrix}. \tag{4.2}$$

Njen rang dobiven naredbom *rank(C)* je 4, što odgovara broju stanja sustava, pa je prema tome sustav upravljiv.

4.1.3 Mjerljivost sustava (engl. *observability*)

Sustav je mjerljiv (engl. *observable*) ako se uz poznato vanjsko djelovanje u(t) i poznate matrice A i C može jednoznačno odrediti početno stanje x0(t) iz izlaznog vektora y(t) promatranog u konačnom vremenskom intervalu t. Mjerljivost se provjerava preko Kalman-ovog uvjeta mjerljivosti koji kaže da je sustav mjerljiv ako matrica:

$$O = \begin{bmatrix} C \\ AC \\ A^2C \\ ... \\ A^{n-1}C \end{bmatrix} \tag{4.3}$$

sadržava toliko linearno neovisnih vektora stupaca koliko ima varijabli stanja. U *Matlab-u* se mjerljivost sustava provjerava koristeći funkcije *obsv(A, C)* i *rank(ans)*.

U slučaju robota naredba

```
O=obsv (A,C)
```

daje matricu:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -20.6 & -0.1 & 0 \\ 0 & 52.4 & 5.8 & 0 \\ 0 & 423.3 & 2.2 & -0.1 \\ 0 & -1077.2 & -5.6 & 5.8 \end{bmatrix}. \tag{4.4}$$

Njen rang dobiven naredbom *rank(O)* je 4, što odgovara broju stanja sustava, pa je prema tome sustav mjerljiv.

Da bi se uspješno primijenile metode optimalnog upravljanja, sustav koji se regulira mora biti upravljiv i mjerljiv.

4.1.4 Stabilnost sustava

Sustav je asimptotski stabilan kad sve svojstvene vrijednosti (eigenvalues) matrice A imaju negativne realne dijelove. Stabilnost se u *Matlab-u* provjerava koristeći funkciju *eig(A)*. U slučaju robota naredba

```
eig(A)
```

daje:

$$\begin{bmatrix} 0 \\ -20.5883 \\ -2.3512 \\ 2.3645 \end{bmatrix}. \tag{4.5}$$

Iz rezultata se zaključuje da je sustav nestabilan, pa je prema tome potrebno projektirati regulator koji će ga činiti stabilnim.

4.2 Kalmanov filtar

4.2.1 Problematika senzora

Za određivanje trenutnog nagiba i regulaciju robota koriste se MEMS žiroskop i akcelerometar. Njihov princip rada temelji se na jednostavnim fizikalnim zakonitostima: inerciji mase u slucaju akcelerometara odnosno Coriolisovu efektu u slucaju žiroskopa. Žiroskop služi za dobivanje kutne brzine robota, koju pokazuje uz visoku točnost te zanemariv šum. Integriranjem te vrijednosti pomoću mikrokontrolera može se dobiti apsolutni kut nagiba. Iz navedenog moglo bi se zaključiti da je za određivanje kuta nagiba dostatno upotrijebiti samo žiroskop, no zbog nakupljanja greške integracije dugoročno se dobiva posve kriva vrijednost kuta. Zbog toga se sustavu dodaje dvo-osni akcelerometar kojim se mjere iznosi ubrzanja u pojedinim osima. Njime se određuje smjer vektora ubrzanja zemljinog gravitacijskog polja to jest određuje se pod kojim je kutom središte zemlje u odnosu na robota (smjer dolje). Za razliku od žiroskopa akcelerometri su podložni utjecaju vibracija koje se manifestiraju kao smetnje, pa su zbog toga kratkoročno nepouzdani. Kako bi se izbjegao drift žiroskopa i uklonio utjecaj smetnji u akcelerometrima, njihovi izlazi se kombiniraju [9] koristeći Kalman-ov filtar.

Općenito, fuzija podataka je proces kombinacije podataka ili informacija kako bi se procijenila ili predvidjela stanja sustava. Podaci se mogu kombinirati s istih ili različitih senzora. Korištenjem većeg broja senzora smanjuje se nesigurnost. Redundantne informacije povećavaju pouzdanost cijelog sustava, dok komplementarne informacije omogućuju percepciju značajki koje nisu vidljive samo jednom senzoru (povećava se observativnost).

Kalmanov filtar je rekurzivni estimator stanja koji omogućuje da se na efikasan način procjene stanja procesa, tako da se minimizira srednja kvadratna pogreška (engl. mean square error). Filtar podržava estimaciju prošlih, sadašnjih te budućih stanja sustava i to radi čak i kada je točno ponašanje modeliranog sustava nepoznato. Svrha Kalmanovog filtra je, uz pomoć poznatog modela sustava i uz prisutan izražen šum u mjerenju senzora, točno određivanje željenih varijabli stanja promatranog procesa.

Na temelju [10, 11] izvedene su temeljnje jednadžbe Kalmanovog filtra.

4.2.2 Estimirani proces

Kalmanov filtar se rabi za estimaciju varijabli stanja $\mathbf{x} \in \Re^n$ diskretno upravljanog procesa kojeg opisuje linearna stohastička jednadžba diferencija:

$$\mathbf{x}(k) = \mathbf{F}(k-1)\mathbf{x}(k-1) + \mathbf{G}(k-1)\mathbf{u}(k-1) + \mathbf{v}(k-1). \tag{4.6}$$

Vrijednost mjerenja opisane su vektorom $\mathbf{z} \in \Re^m$ i njegovo ponašanje je opisano jednadžbom:

$$\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{e}(k). \tag{4.7}$$

Slučajne varijable $\mathbf{v}(k)$ i $\mathbf{e}(k)$ predstavljaju šumove u sustavu; $\mathbf{v}(k)$ šum procesa, a $\mathbf{e}(k)$ šum mjerenja. Pretpostavlja se da su oni međusobno nezavisni i da imaju normalnu raspodjelu vjerojatnosti.

Matrica $\mathbf{F}$ dimenzije n x n predstavlja vezu između vrijednosti varijabli stanja *varijable* u koraku k i $k-1$ za slučaj da nema upravljačkog signala ni šuma u procesu. Matrica $\mathbf{G}$ dimenzije n x l povezuje vrijednost upravljačke varijable $\mathbf{u}$ i varijable $\mathbf{x}$. Matrica $\mathbf{H}$ dimenzije m x n u jednadžbi mjerenja povezuje varijablu $\mathbf{x}$ s mjerenjem $\mathbf{z}(k)$. U primjeni se vrijednosti matrica $\mathbf{H}$ i $\mathbf{F}$ mogu mijenjati u vremenu, no u prvom djelu će se pretpostaviti da su konstantne.

4.2.3 Algoritam diskretnog Kalmanovog filtra

Kalmanov filtar estimira varijable stanja procesa koristeći oblik povratne veze - Povratna veza daje pogrešku procjene vrijedonsti mjerenja, dobivenu na temelju estimiranih varijabli stanja procesa i poznate izlazne jednadžbe. Jednadžbe Kalmanovog filtra dijele se u dvije grupe: jednadžbe koje služe za ažuriranje stanja i jednadžbe koje služe za ažuriranje mjerenja.

Jednadžbe ažuriranja stanja projiciraju trenutno stanje i estimaciju kovarijance greške kako bi se dobile *a priori* (iz modela) estimacije za sljedeći vremenski korak. Jednadžbe ažuriranja mjerenja služe za dobivanje povratne informacije, odnosno za uklapanje novog mjerenja u *a priori* estimaciju kako bi se dobila poboljšana *a posteriori* (stečena promatranjem) estimacija stanja.

Jednadžbe ažuriranja stanja također se mogu promatrati kao jednadžbe predikcije, dok se jednadžbe mjerenja mogu gledati kao jednadžbe korekcije.
Jednadžbe ažuriranja stanja diskretnog Kalmanovog filtra su:

$$\widehat{\mathbf{x}}(k \mid k-1) = \mathbf{F}(k-1)\widehat{\mathbf{x}}(k-1 \mid k-1) + \mathbf{G}(k-1)\mathbf{u}(k-1), \tag{4.8}$$

$$\mathbf{P}(k \mid k-1) = \mathbf{F}(k-1)\mathbf{P}(k-1 \mid k-1)\mathbf{F}^T(k-1) + \mathbf{Q}(k-1). \tag{4.9}$$

Jednadžbe ažuriranja mjerenja diskretnog Kalmanovog filtra su:

$$\mathbf{K}(k) = \mathbf{P}(k \mid k-1)\mathbf{H}^T(k)(\mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}^T(k) + \mathbf{R}(k))^{-1}, \tag{4.10}$$

$$\widehat{\mathbf{x}}(k \mid k) = \widehat{\mathbf{x}}(k \mid k-1) + \mathbf{K}(k)(\mathbf{z}(k) - \mathbf{H}(k)\widehat{\mathbf{x}}(k \mid k-1)), \tag{4.11}$$

$$\mathbf{P}(k \mid k) = (\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k))\mathbf{P}(k \mid k-1). \tag{4.12}$$

Prvo se izračunava Kalmanovo pojačanje $\mathbf{K}(k)$. Zatim se u jednadžbi (4.11) računa *a posteriori* vrijednost stanja procesa tako da se uračuna vrijednost mjerenja $\mathbf{z}(k)$. Posljednji korak je računanje *a posteriori* vrijednosti kovarijance greške pomoću (4.12).

Nakon svakog koraka predikcije i korekcije, algoritam se ponavlja sa prijašnjim *a posteriori* estimacijama koje se koriste za projiciranje ili predikciju novih *a priori* estimacija. Ova rekurzivna priroda algoritma (novo stanje ovisi samo o prethodnom stanju) je jedna od vrlo privlačnih značajki Kalmanovog filtra. Čini praktičnu implementaciju mnogo povoljnijom od na primjer implementacije Wiener-ovog filtra koji istovremeno koristi cijeli skup podataka za svaku estimaciju.

4.2.4 Prošireni Kalmanov filter

Prošireni Kalmanov filtar je nelinearna verzija klasičnog Kalmanovog filtra koji je lineariziran oko trenutne srednje vrijednosti i kovarijance. Prošireni Kalmanov filtar se redovito koristi u teoriji nelinearne estimacije stanja, u navigacijskim sustavima i GPS-u. Nelinearni model procesa ima sljedeću strukturu:

$$\mathbf{x}(k) = \mathbf{f}(x(k-1), \mathbf{u}(k-1), \mathbf{v}(k-1)) \tag{4.13}$$

$$\mathbf{y}(k) = \mathbf{h}(\mathbf{x}(k), \mathbf{e}(k)) \tag{4.14}$$

Gdje su $\mathbf{f}$ i $\mathbf{h}$ nelinearne diferenciabline, monotone funkcije stanja (f i h su "glatke" funkcije).

Funkcija $\mathbf{f}$ se koristi za izračunavanje predviđenih stanja koristeći predviđanja iz prošlog koraka estimacije. Funkcija $\mathbf{h}$ se koristi za izračunavanje predviđenih mjerenja iz predviđenih stanja. No, $\mathbf{f}$ i $\mathbf{h}$ se ne mogu primijeniti direktno na kovarijancu. Umjesto toga

se izračunava matrica parcijalnih derivacija (Jacobian). U svakom koraku Jacobian se računa iz predviđenih stanja. Te matrice mogu se koristiti u jednadžbama Kalmanovog filtra. Ovaj proces u osnovi linearizira nelinearnu funkciju oko trenutno predviđenog stanja.

Jednadžbe ažuriranja stanja proširenog Kalmanovog filtra:

$$\widehat{\mathbf{x}}(k \mid k-1) = \mathbf{f}(\widehat{\mathbf{x}}(k-1 \mid k-1), \mathbf{u}(k-1), 0), \tag{4.15}$$

$$\mathbf{P}(k \mid k-1) = \mathbf{F}(k-1)\mathbf{P}(k-1 \mid k-1)\mathbf{F}(k-1)^T + \mathbf{Q}(k-1). \tag{4.16}$$

Jednadžbe ažuriranja mjerenja diskretnog Kalmanovog filtra:

$$\mathbf{K}(k) = \mathbf{P}(k \mid k-1)\mathbf{H}(k)^T(\mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}(k)^T + \mathbf{R}(k))^{-1}, \tag{4.17}$$

$$\widehat{\mathbf{x}}(k \mid k) = \widehat{\mathbf{x}}(k \mid k-1) + \mathbf{K}(k)(\mathbf{z}(k) - \mathbf{h}(\widehat{\mathbf{x}}(k \mid k-1))), \tag{4.18}$$

$$\mathbf{P}(k \mid k) = (\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k))\mathbf{P}(k \mid k-1). \tag{4.19}$$

Matrica prijelaza i matrica mjerenja definiraju se:

$$\mathbf{F}(k-1) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\widehat{\mathbf{x}}(k-1), \mathbf{u}(k-1)}, \tag{4.20}$$

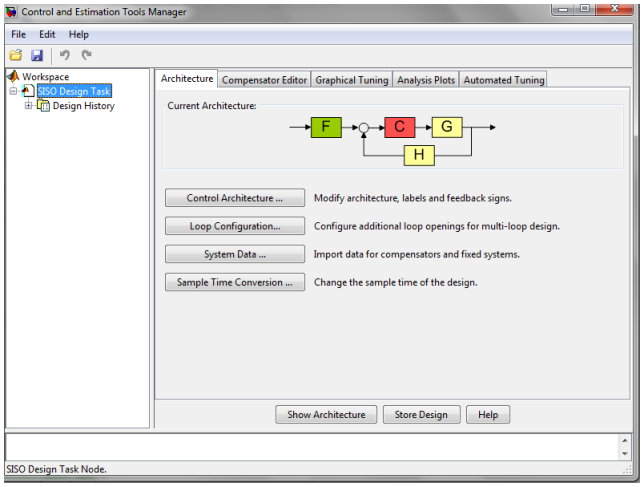
$$\mathbf{H}(k-1) = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\widehat{\mathbf{x}}(k-1)}. \tag{4.21}$$

Za razliku od linearnog, prošireni Kalmanov filtar u osnovi nije optimalni estimator (optimalan je ako su mjerenja i model prijelaza linearni, u tom slučaju je prošireni Kalmanov filtar identičan običnome, dok za nelinearne procese u općenitom smislu nije optimalan). Osim toga ako je početno predviđanje stanja netočno, ili ako je proces krivo modeliran, zbog linearizacije modela procesa estimati varijabli stanja mogu brzo divergirati.

4.3 PID regulator

Prije projektiranja naprednijih oblika regulatora, napravljena je sinteza PID regulatora. PID regulatori omogućuju regulaciju SISO (single input single output) sustava - sustava sa jednim ulazom i jednim izlazom. U ovom slučaju varijabla koja će se regulirati je kut nagiba robota. Projektiranje je izvršeno koristeći root locus metodu u *sisotool* okruženju.

Naredbom sisotool(sys), gdje je sys = ss(A,B,C,D), otvara se sučelje za projektiranje regulacije SISO sustava. U njemu se može mijenjati struktura sustava regulacije, dodavati regulatori, prikazivati bode-ovi dijagrami, root lokus grafovi, odazivi na step, delta funkciju itd.



Slika 4.1: *Sučelje za projektiranje regulacije SISO sustava*

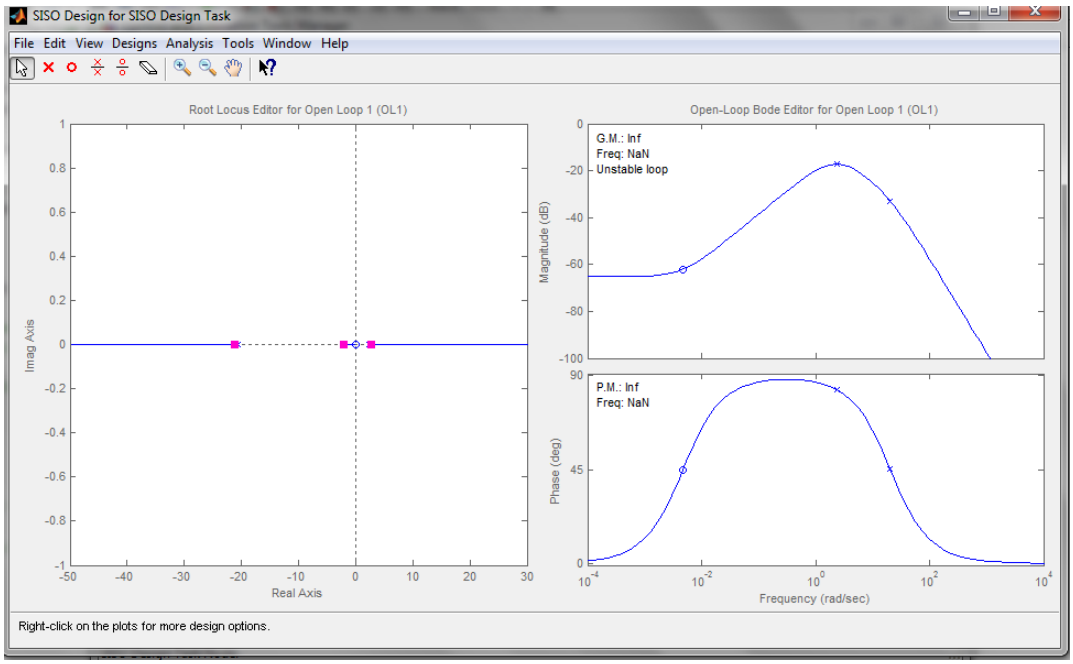
Nakon što se sustavu dodaju kompleksna nula i realni pol

$$K_i \frac{1 + T_p s + T_d s^2}{s}, \tag{4.22}$$

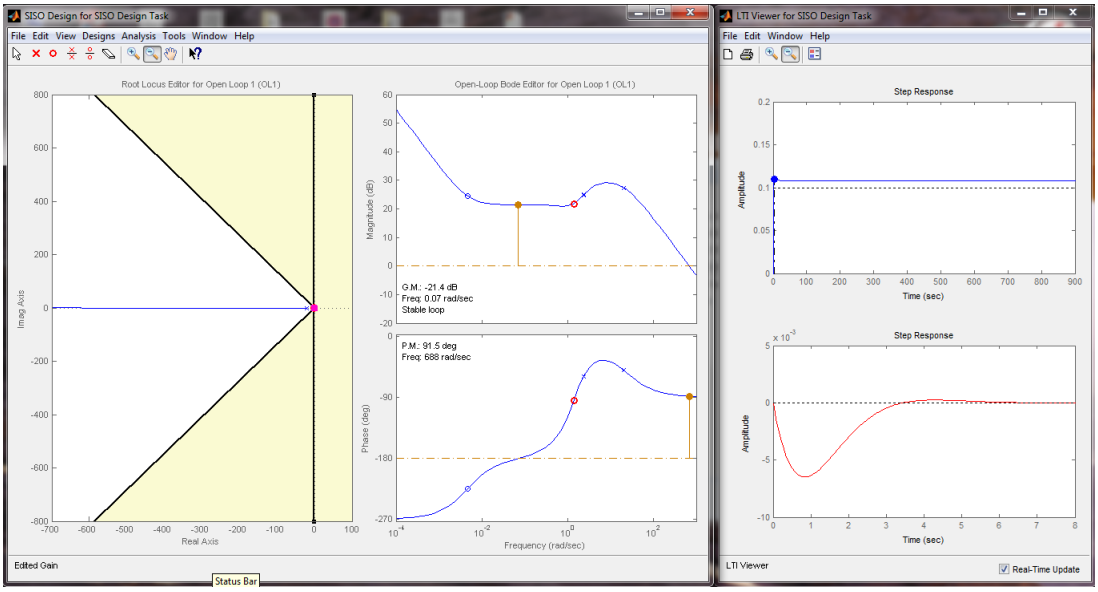
te nakon što se odabere pogodno pojačanje K sustav postane stabilan, što se vidi na slici 4.3. Prigušenje sustava je postavljeno na 0.7071, a prirodna frekvencija na 1.4.

Ulaz PID regulatoru je kut nagiba Θ, a izlaz je upravljačka varijabla u. Slika 4.4 prikazuje položaj PID bloka u modelu.

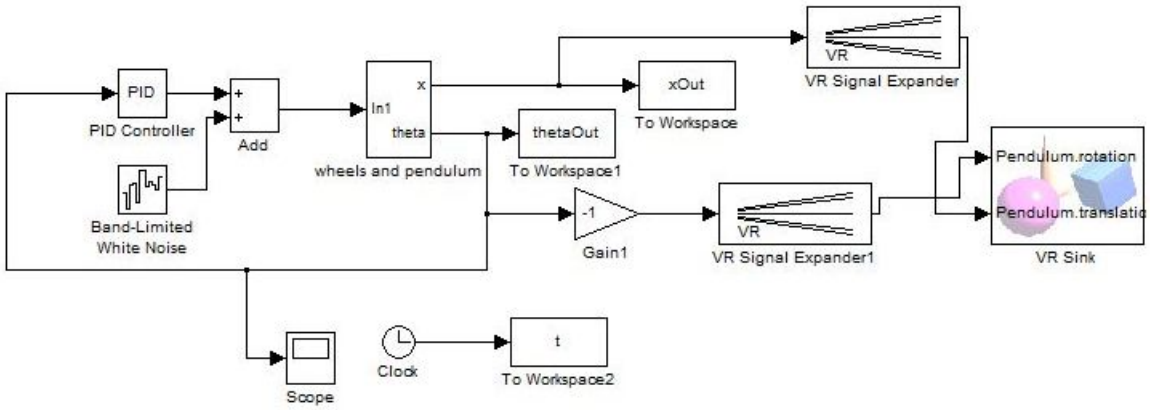
Testiranje dobivenih pojačanja (K_p, K_d, K_i) na nelinearnom simulink modelu pokazuje da projektirani regulator zadovoljava tražena svojstva.



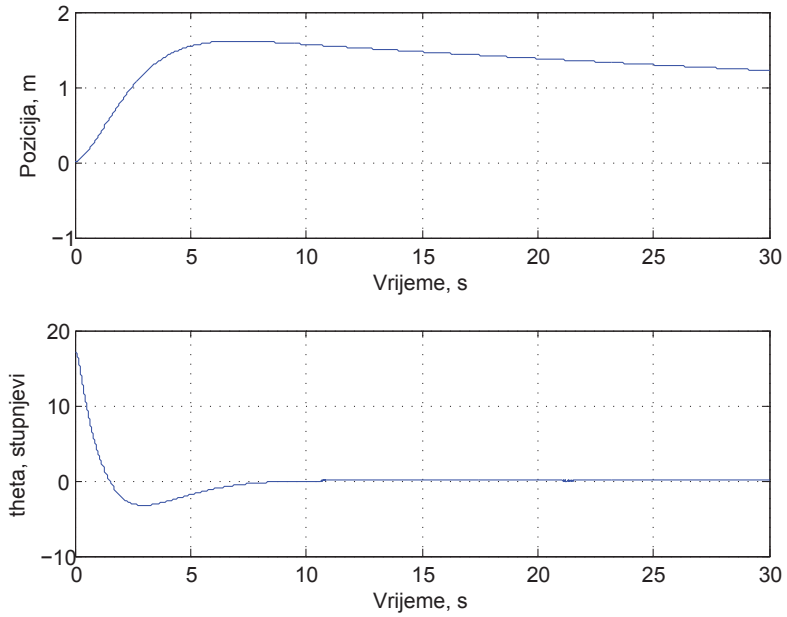
Slika 4.2: *Root locus i bodeovi dijagrami prije dodavanja PID regulatora, root lokus graf pokazuje da sustav ima jedan pol u pozitivnoj polu ravnini*



Slika 4.3: *Root locus, bodeovi dijagrami sa dodanom kompleksnom nulom i realnim polom, te odaziv na step i delta funkciju*



Slika 4.4: *PID regulacijski krug*



Slika 4.5: *Rezultat projektiranja PID regulatora, grafovi prikazuju poziciju i kuta nagiba robota nakon što se robot pokrene sa početnim kutom nagiba od 0.4 radijana*

4.4 LQR regulator

Za razliku od PID regulatora, LQR regulator može regulirati MIMO (engl. *Multiple input multiple output*) sustave – sustave sa više ulaza i izlaza. LQR regulator pretpostavlja da su u svakom trenutku poznate sve vrijednosti stanja sustava.

Za sintezu LQR regulatora potrebno je poznavati matrice stanja sustava [12]. Uz njih potrebno je definirati i matrice Q i R. Matrice Q i R odabiru se prema Brysonovoj metodi koja kaže:

$$Q_{ii} = \frac{1}{maksimalna \quad dozvoljena \quad vrijednost \quad varijable \quad z_i^2}, \tag{4.23}$$

$$R_{jj} = \frac{1}{maksimalna \quad dozvoljena \quad vrijednost \quad varijable \quad u_i^2}. \tag{4.24}$$

Matrica Q je dimenzije 4x4 i označava koliko će sustav težiti regulaciji pojedine varijable. Za nju su odabrane vrijednosti:

$$\begin{bmatrix} \frac{1}{0.095^2} & 0 & 0 & 0 \\ 0 & \frac{1}{0.09^2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{0.06^2} \end{bmatrix}. \tag{4.25}$$

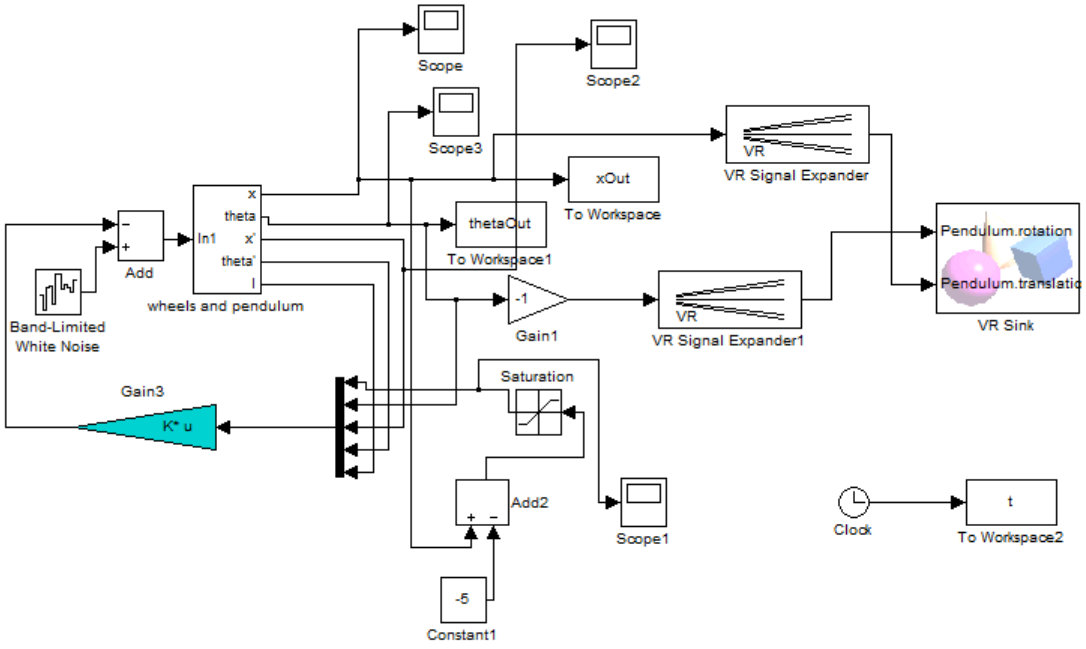
Sustav ima samo jedan ulaz, pa je prema tome dimenzija matice R 1x1. Za nju je odabrana vrijednost [1].

Sljedeći korak kod sinteze LQR regulatora je određivanje pojačanja K. On se određuje koristeći *Matlab-ovu* naredbu K = lqr(A,B,Q,R) koja za ovaj sustav daje vrijednost K:

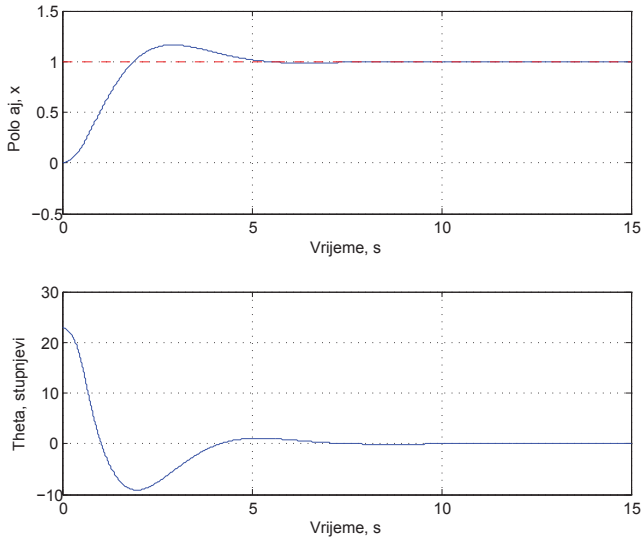
$$\begin{bmatrix} -10.5263 & -54.5593 & -22.1275 & -28.7393 \end{bmatrix}$$

Dobivene vrijednosti primjene se na blok regulatora (gain blok) tako da se kao ulaz postave stanja sustava redoslijedom koji odgovara onom korištenom u sintezi regulatora. Zatim se te vrijednosti matrično množe sa matricom K, pa se na izlazu dobiva vrijednost upravljačke varijable u.

S LQR regulatorom robot može točno regulirati i poziciju i kut nagiba, što se vidi u rezultatu simulacije (slika 4.7).



Slika 4.6: *Blokovski prikaz regulacijskog kruga robota. Plavom bojom označen je blok LQR regulatora*

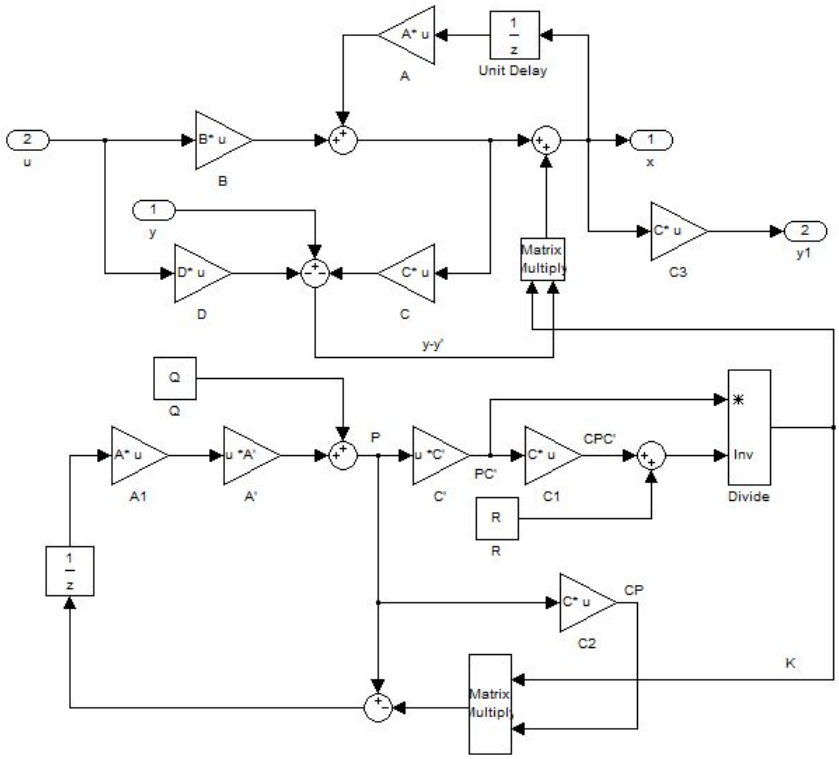


Slika 4.7: *Rezultat sinteze LQR regulatora - zadana je referenca položaja od 1 metra, te početni kut nagiba 0.4 radijana*

4.5 LQG regulator

LQG (Linear Quadratic Gaussian) regulator je proširenje LQR regulatora na sustave u kojima mjerenje svih varijabli stanja nije moguće ili nije praktično, te u kojima je u mjerenju prisutan znatan šum. LQG regulator je kombinacija LQR regulatora i LQE (Linear Quadratic Estimator) - linearnog estimatora stanja, koji je u stvari već spomenuti Kalmanov filtar.

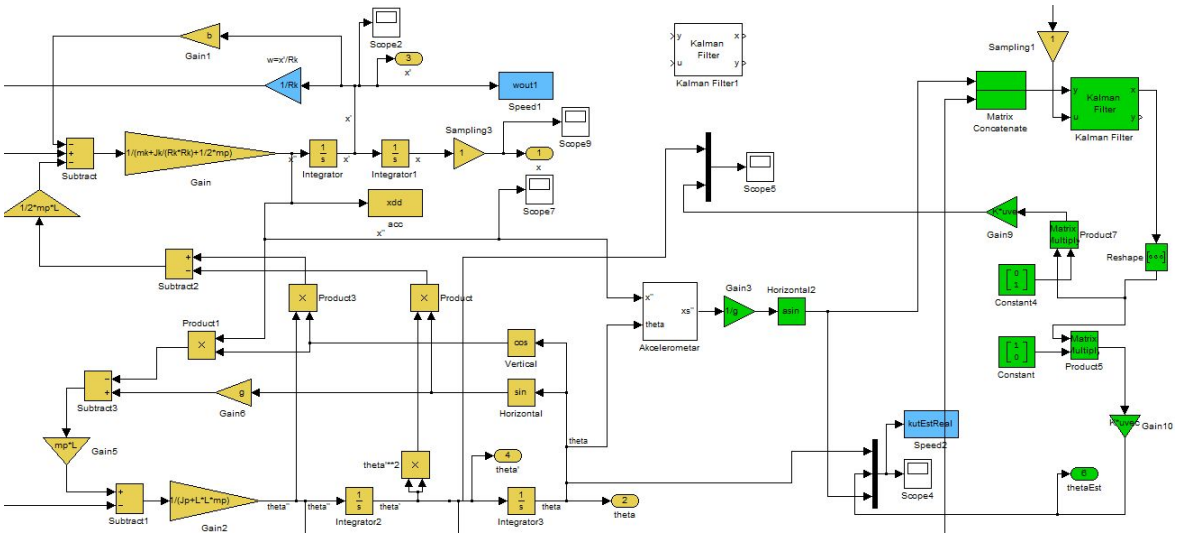
Struktura primijenjenog LQG regulatora slična je kao kod LQR regulatora, samo je između LQR regulatora i sustava dodan blok Kalmanovog filtra. On uzima senzorima zašumljene vrijednosti stanja i estimira njihovu točnu vrijednost. Simulink model samog



Slika 4.8: *Položaj Kalmanovog filtra u krugu regulacije*

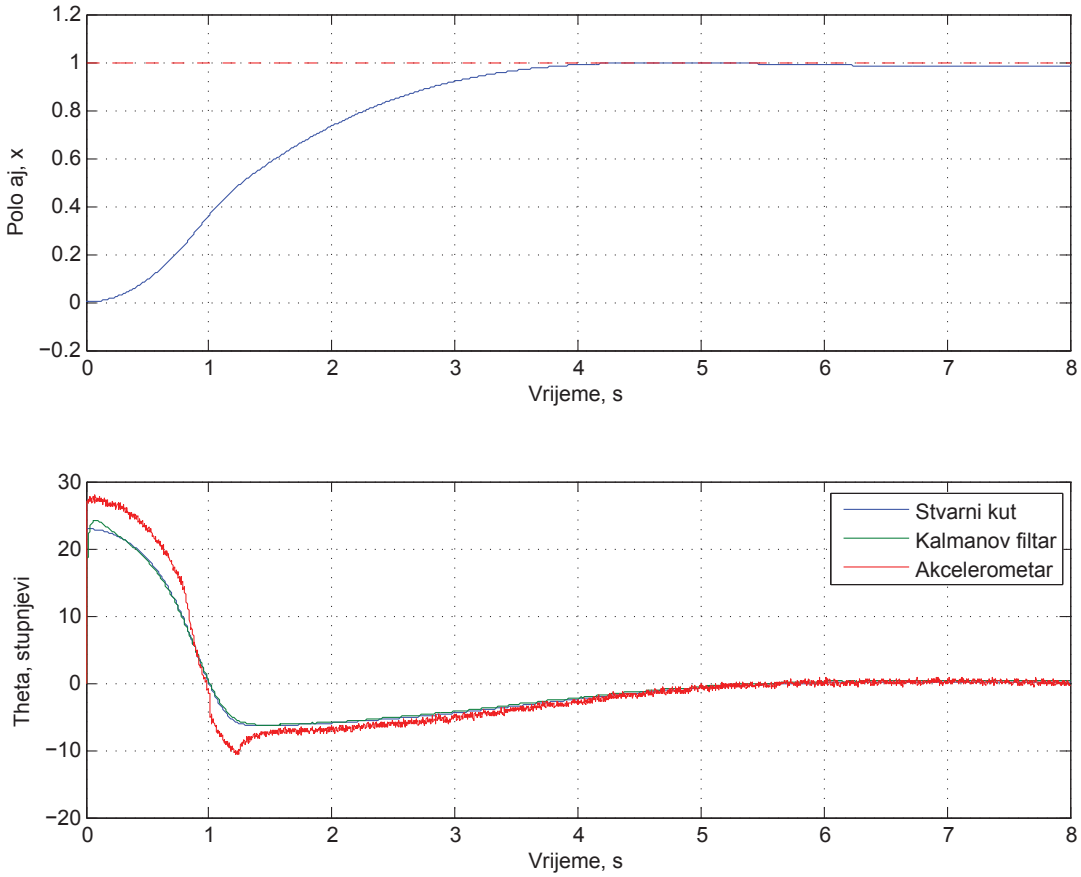
Kalmanovog filtra odgovara jednadžbama danima u poglavlju 2. Ovaj model odgovara vremenski promjenjivom Kalmanovom filtru, dok će se u samom mikrokontroleru primijeniti vremenski ustaljeni oblik. Razlika između ova dva oblika postoji samo u

nekoliko početnih koraka, kasnije je njihovo djelovanje jednako. Zbog ograničenih mogućnosti mikrokontrolera povoljno je primijeniti drugu varijantu jer se kod nje zanemaruje veliki dio matričnog računa. Parametri Kalmanovog filtra postavljeni su tako da se najviše vjeruje vrijednostima dobivenim sa žiroskopa, a manje s akcelerometra.



Slika 4.9: *Simulink model Kalmanovog filtra*

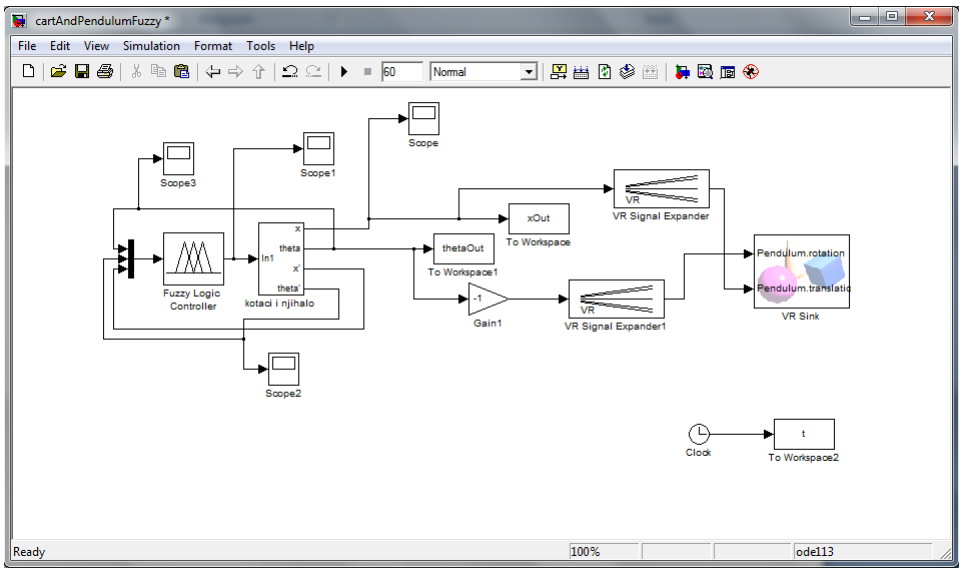
Kako bi se što bolje estimirala vrijednost kuta nagiba, upravljačka varijabla pridružena je vrijednostima dobivenim kombinacijom akcelerometra i žiroskopa, čime se postiže kompenzacija linearne akceleracije robota. Vrijednosti koje se dobiju na izlazu iz Kalmanovog filtra zadovoljavajuće prate vrijednost realnog kuta nagiba, što je vidljivo na slici 4.10.



Slika 4.10: *Rezultat simulacije modela sa LQG regulatorom na jedinični skok reference položaja*

4.6 Neizraziti regulator

U ovom dijelu prikazati će se primjena neizrazitog regulatora na sustavu inverznog njihala. Koristeći *Matlabovo* grafičko sučelje [13] za projektiranje sustava neizrazite logike, napravljen je neizraziti regulator. Ulazi u regulator su vrijednosti sa izlaza senzora, a to su kut nagiba, kutna brzina i linearna brzina. Te vrijednosti se, koristeći ulazne funkcije pripadnosti, fazificiraju, čime postaju neizraziti ulazi. Nakon toga se koristeći definirani skup neizrazitih pravila (engl. *fuzzy rules*) generira neizraziti izlaz. Zatim se taj izlaz defazificira koristeći izlazne funkcije pripadnosti, te se dobije vrijednost koja odgovara naponu kojim se djeluje na motore.



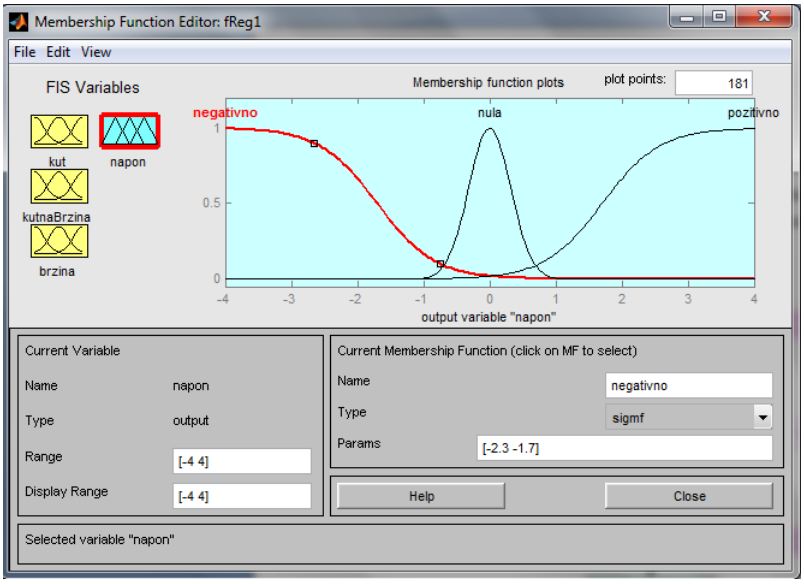
Slika 4.11: *Implementacija neizrazitog regulatora*

Sučelje za projektiranje sustava neizrazite logike se otvara korištenjem *Matlabove* naredbe "fuzzy". U njemu se s lijeve strane nalaze ulazne funkcije pripadnosti, u sredini pravila neizrazitog odličivanja, te sa desne strane izlazne funkcije pripadnosti.

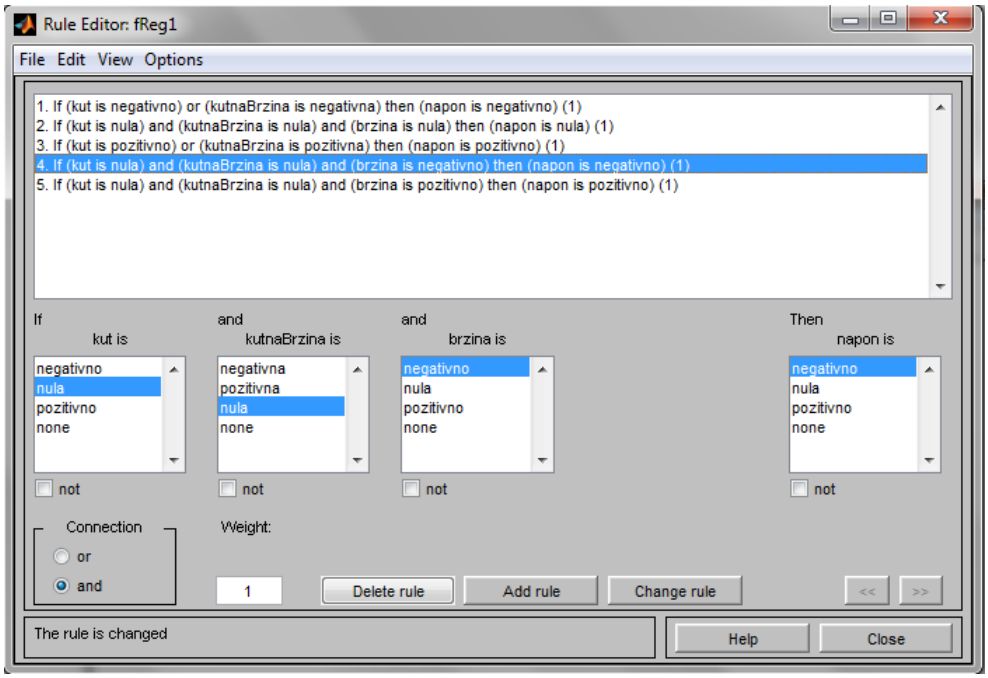
Budući regulator uzima tri ulaza, napravljene su tri funkcije pripadnosti s odgovarajućim neizrazitim skupovima. Za sve funkcije pripadnosti korištene su funkcije tipa *sigmf* i *gaussmf*.

Potom se određuju pravila neizrazitog odlučivanja, pri čemu su definirana pet pravila koja određuju neizraziti izlaz. Njihovom primjenom određuje se koja će funkcija biti aktivna u slučaju da se ostvari zadani uvjet.

Na slici 4.13 označeno je pravilo koje na izlaz pridružuje negativan napon u slučaju da su kut i kutna brzina unutar vrijednosti nula, te da je brzina negativna. To pravilo je potrebno



Slika 4.12: *Definiranje neizrazitih skupova*

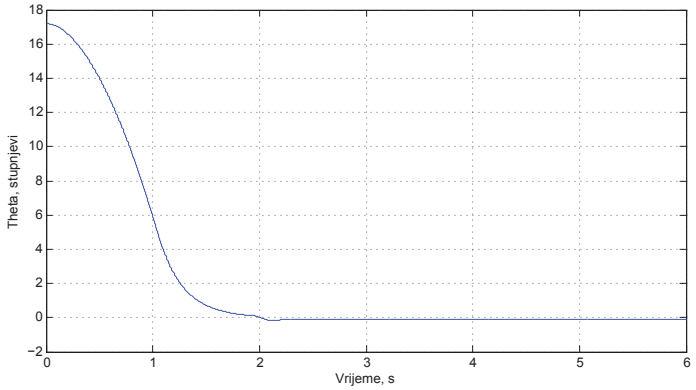


Slika 4.13: *Pravila neizrazitog odlučivanja*

jer bez njega robot uslijed greške kuta nagiba počinje značajno povećavati brzinu u jednom smjeru, što eventualno dovodi do njegovog prevrtanja zbog limita maksimalne brzine.

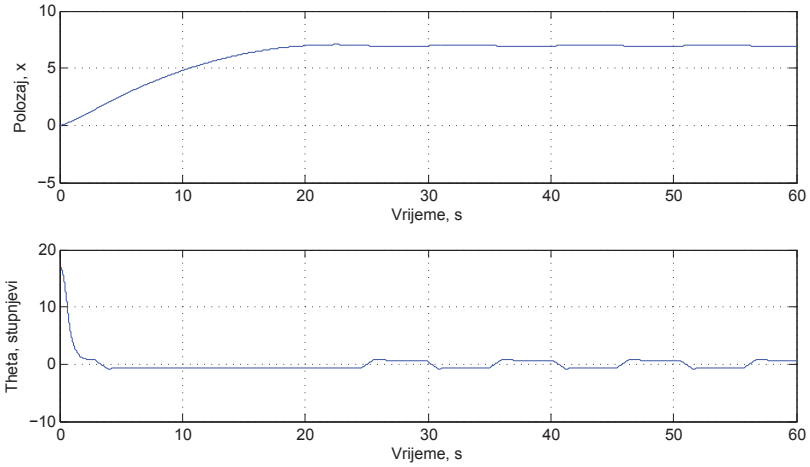
Slika 4.14 prikazuje nagib robota dobiven koristeći projektirani neizraziti regulator. Robot kreće iz početnog nagiba od oko 18°. Unutar četiri sekunde se uspješno stabilizira

na približno 0° bez prebačaja.



Slika 4.14: *Rezultat kontrole robota neizrazitim regulatorom*

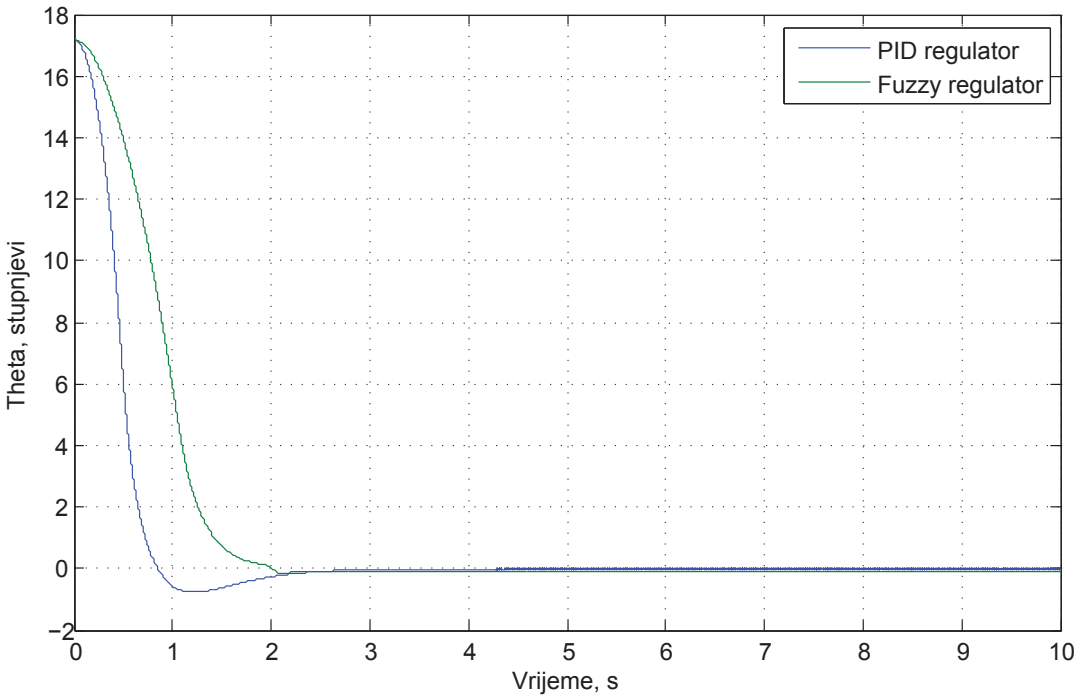
Slika 4.15 prikazuje dugoročne posljedice postojanja regulacijske pogreške nagiba na poziciju robota. Vidi se da pozicija robota zanemarivom amplitudom oscilira oko 7 metara.



Slika 4.15: *Pozicija i nagib robota unutar 60 sekundi*

Neizraziti regulator je uspoređen sa PID regulatorom projektiranim prema root locus metodi. Slika 4.16 prikazuje rezultat simulacija na kojem se vidi da je ponašanje neizrazitog regulatora zadovoljavajuće. Brzina postizanja stacionarne vrijednosti je malo sporija nego kod PID regulatora i iznosi oko 4 sekunde.

Za razliku od PID regulatora, kod neizrazitog regulatora se javlja mala stacionarna pogreška prema slici 4.15, što bi se moglo izbjeći složenijim modelom regulatora.

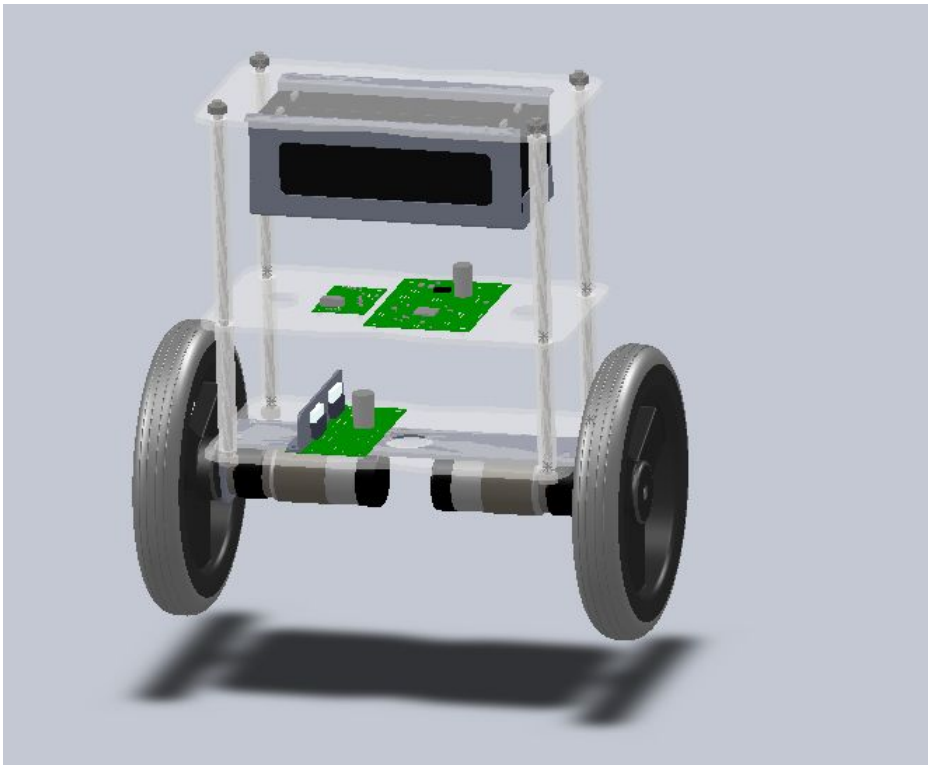


Slika 4.16: *Usporedba PID i neizrazitog regulatora*

Poglavlje 5

Mehanička konstrukcija

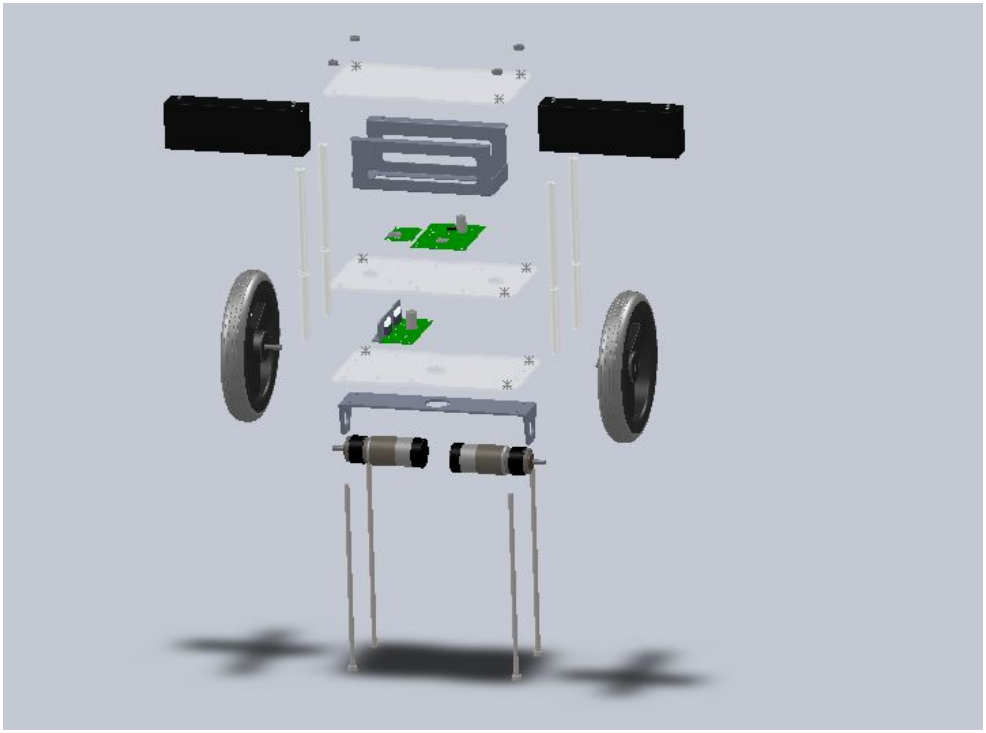
Konstrukcija mobilnog robota modelirana je u programskom paketu *SolidWorks*. Slika 5.1 prikazuje koncept izgleda mobilnog robota sa svim glavnim komponentama. Konstrukcija se sastoji od tri pleksiglas ploče na kojima su postavljeni ostali elementi.



Slika 5.1: *Model mobilnog robota u SolidWorks-u*

Na prvoj ploči postavljeni su motori i sklopovi za njihovu regulaciju. Upravljačka jedinica, senzori nagiba, i komunikacijski modul smješteni su na drugoj ploči. Na trećoj

ploči postavljanje su dvije punjive baterije od 12V, kapaciteta 1.9Ah. Ploče su povezane sa četiri navojne šipke te odvojene cilindričnim separatorima. Cijela konstrukcija izvedena je modularno i lako je proširiva, stoga se uz minimalne preinake može dodati još jedna razina na koju se mogu postaviti dodatni elementi poput kamere, manipulatora, ultrazvučnih senzora, itd.

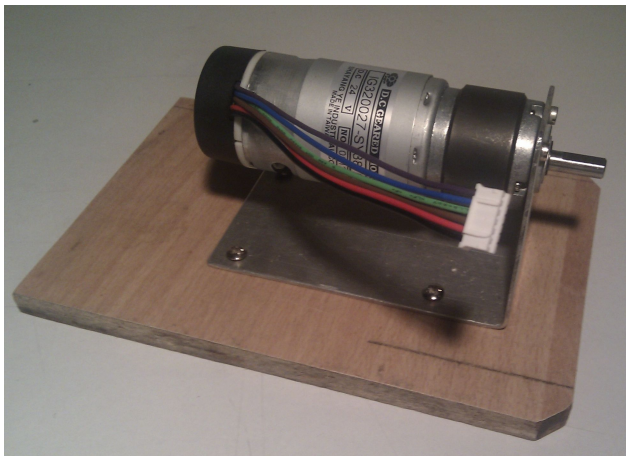


Slika 5.2: *"Eksplodirani" prikaz modela robota*

Za razliku od klasičnih mobilnih robota gdje je cilj postaviti težište što niže radi postizanja bolje stabilnosti i bolje upravljivosti, kod ove vrste robota se postavlja što više. Time je povećan moment inercije robota oko centralne osi kotača, što rezultira povećanom stabilnošću robota zato jer je dinamika donjeg dijela veća od dinamike gornjeg dijela.

Kao aktuatori korištena su dva 24 voltna istosmjerna motora, čiji smjer i brzina se reguliraju preko H-mosta koristeći pulsno širinsku modulaciju (PWM). Sa stražnje strane motora nalaze se magnetski enkoderi koji daju osam impulsa po okretaju motora, odnosno 216 impulsa po okretaju izlaznog vratila. Na prednjoj strani motora spojen je planetarni reduktor s prijenosnim omjerom 1:27 koji služi za ostvarivanje dovoljnog momenta pri niskim okretajima. Svako vratilo motora spojeno je s kotačem promjera 19 cm.

Za testiranje parametra kupljenih motora napravljen je eksperimentalni postav prikazan na slici 5.3.



Slika 5.3: *Postav za testiranje motora*

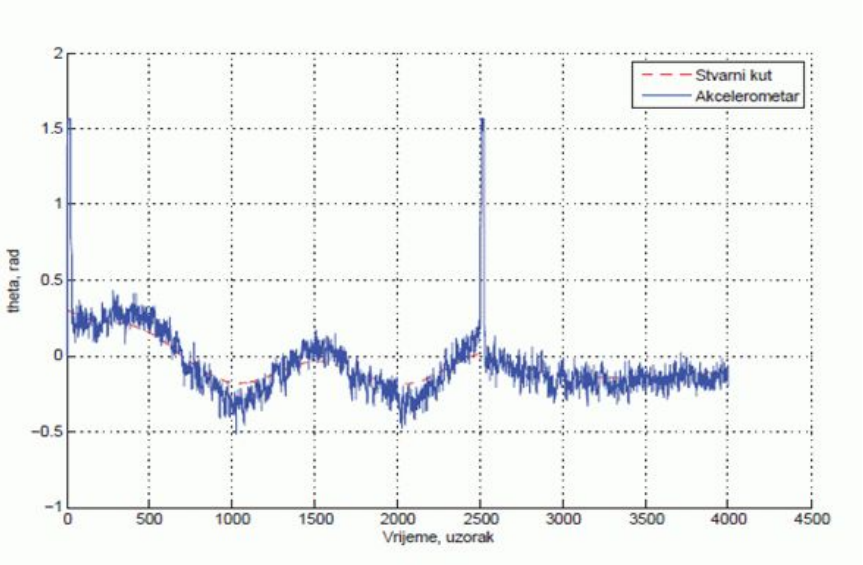
Slika 5.4: *Razvoj mehaničke konstrukcije*

Poglavlje 6

Elektronika

6.1 Senzori

Za određivanje trenutnog nagiba i regulaciju položaja robota koriste se tri vrste senzora: dvo-osni akcelerometar, jedno-osni žiroskop i inkrementalni enkoderi na motorima. Žiroskop služi za dobivanje kutne brzine robota, koju pokazuje uz visoku točnost te zanemariv šum. Integriranjem te vrijednosti pomoću mikrokontrolera može se dobiti apsolutni kut nagiba. Iz navedenog bi se moglo zaključiti da je za određivanje kuta nagiba dostatno upotrijebiti samo žiroskop, no zbog nakupljanja greške integracije dugoročno se dobiva posve kriva vrijednost kuta.



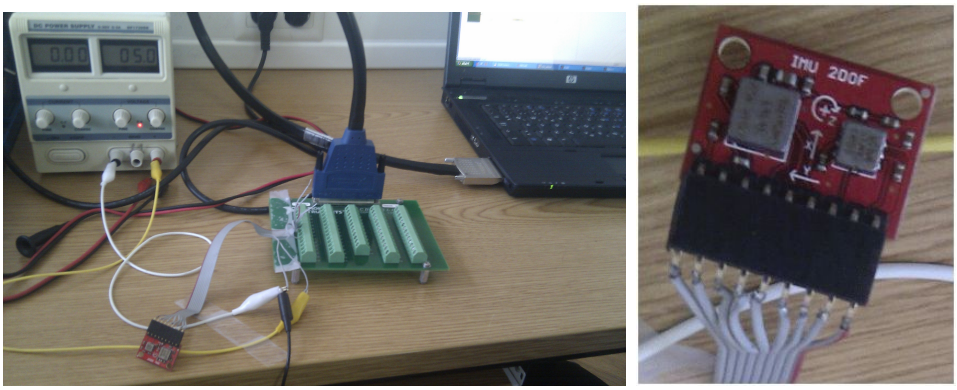
Slika 6.1: *Zašumljeni signal akcelerometra*

Zbog toga se sustavu dodaje dvo-osni akcelerometar kojim se mjere iznosi ubrzanja u pojedinim osima. Njime se određuje smjer vektora ubrzanja zemljinog gravitacijskog polja to jest određuje se pod kojim je kutom središte zemlje u odnosu na robota (smjer dolje). Za razliku od žiroskopa akcelerometri su podložni utjecaju vibracija koje se manifestiraju kao smetnje, pa su zbog toga kratkoročno nepouzdani. Kako bi se izbjegao drift žiroskopa i uklonio utjecaj smetnji u akcelerometrima, njihovi izlazi se kombiniraju koristeći Kalman-ov filtar. Na taj način se rješava problem određivanja nagiba.

Dugoročno gledano uslijed grešaka prilikom regulacije nagiba položaj robota može se znatno promijeniti. Zbog toga se na motore postavljaju inkrementalni enkoderi. Sumiranjem impulsa može se sa zadovoljavajućom točnošću odrediti položaj robota, i prema njemu primijeniti odgovarajuća regulacijska vrijednost.

Za određivanje nagiba odabran je „3DOF IMU combo board“, tiskana pločica koja sadrži dva čipa: ADXL203 dvo-osni akcelerometar i ADXRS613 jedno-osni žiroskop.

Ispravnost senzora je provjerena spajanjem istih s akvizicijskom karticom te dobivanjem željenih signala na računalu.

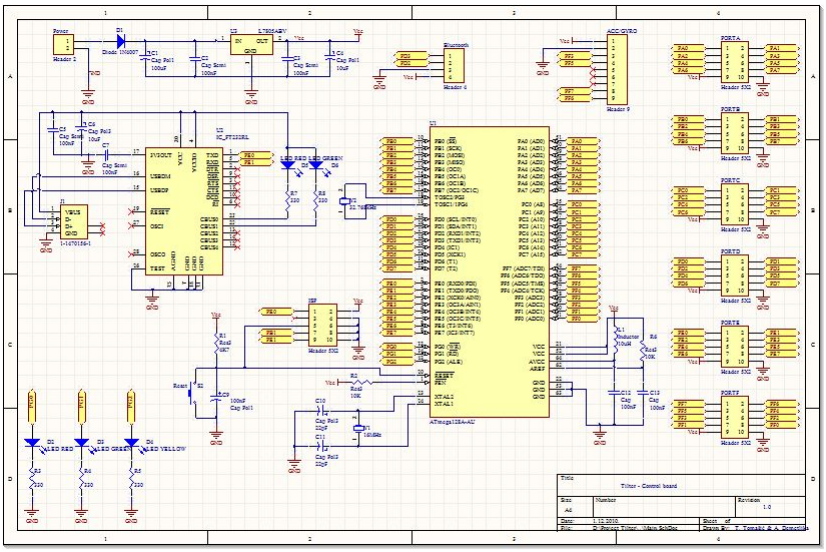


Slika 6.2: *Testiranje senzora*

6.2 Projektiranje i izrada tiskanih pločica

Mobilni robot sadrži tri odvojene tiskane pločice. Projektiranje pločica [14] izvedeno je u programskom alatu *Altium Designer*. Na glavnoj pločici nalazi se regulator napona, koji stabilizira napon sa 12 na 5V, kontektor i čip FT232 za USB komunikaciju, konektori kojima se spaja na druge pločice, te mikrokontroler ATmega128A. Funkcija mikrokontrlera je da vrši A/D pretvorbu signala senzora, na temelju čega izračunava potreban PWM signal pomoću kojeg se upravljaju motori. Osim toga preko njega se vrši komunikacija sa drugim uređajima.

Pločica je izvedena tako da je omogućeno proširivanje dodatnim elementima, poput kamere, ultrazvučnih senzora, itd. Druga pločica predstavlja vezu između upravljačkog i energetskog dijela. Na njoj se nalaze dva H-mosta koji pomoću PWM signala, dobivenog s mikrokontrolera, upravljaju motorima. Projektiranje tiskane pločice obuhvaća definiranje i odabir svih komponenta te izradu električne sheme (slika 6.3).

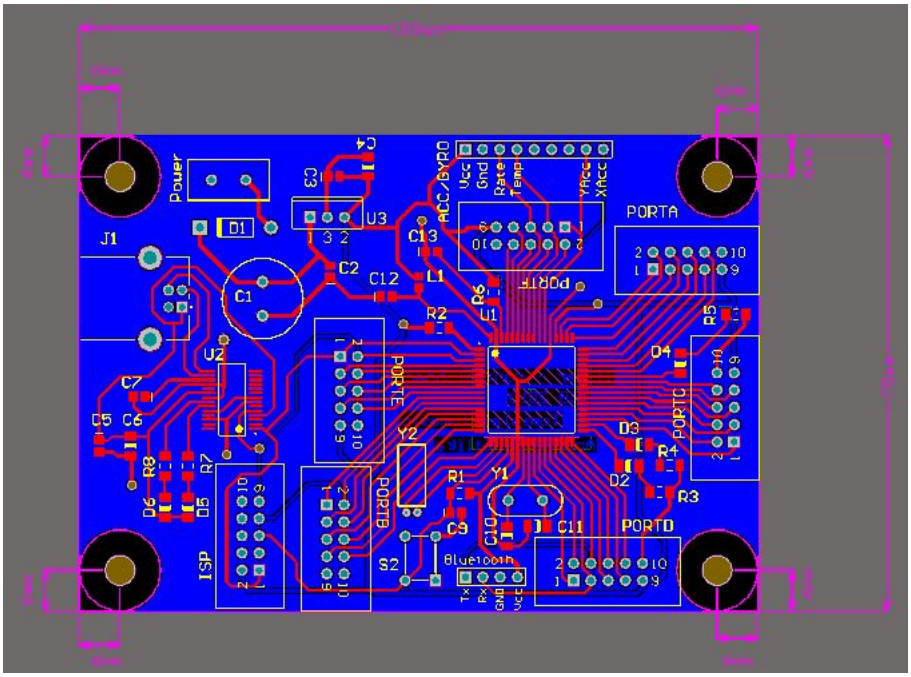


Slika 6.3: *Shematski prikaz glavne tiskane pločice*

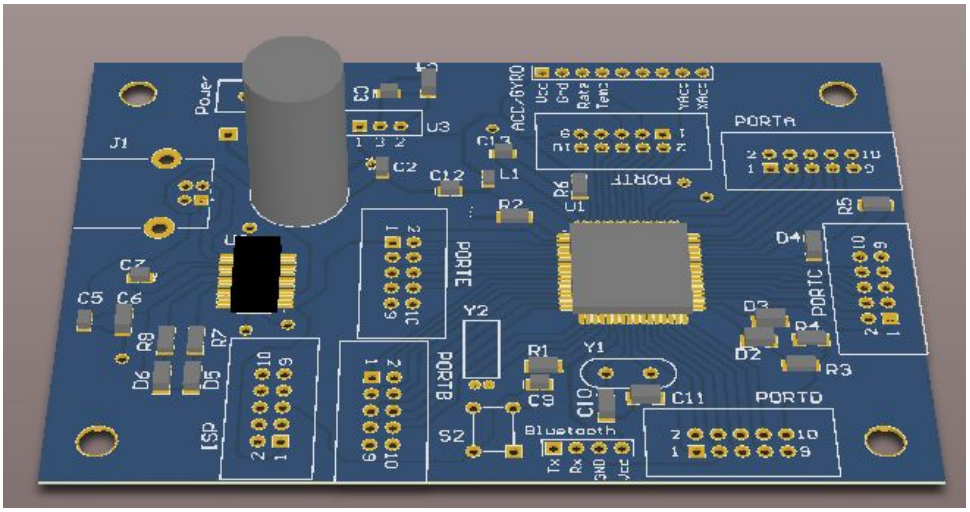
Nakon izrade električne sheme, potrebno je napraviti PCB dokument (slika 6.4) u kojemu se generira dokumentacija za izradu osnovnih maski za tehnološku realizaciju dvostrane tiskane pločice (položajni nacrt komponenata, nacrti tiskanih veza i plan bušenja). *Altium Designer* omogućava i 3D prikaz projektiranih tiskanih pločica što je prikazano slikama 6.5 i 6.6.

Za komunikaciju između robota i računala, odnosno robota i mobitela, odabrana je bluetooth komunikacija. Glavni razlog izbora tog oblika je široka dostupnost uređaja sa ugrađenim bluetooth modulom, poput prijenosnih računala, mobitela, joysticka. Bluetooth

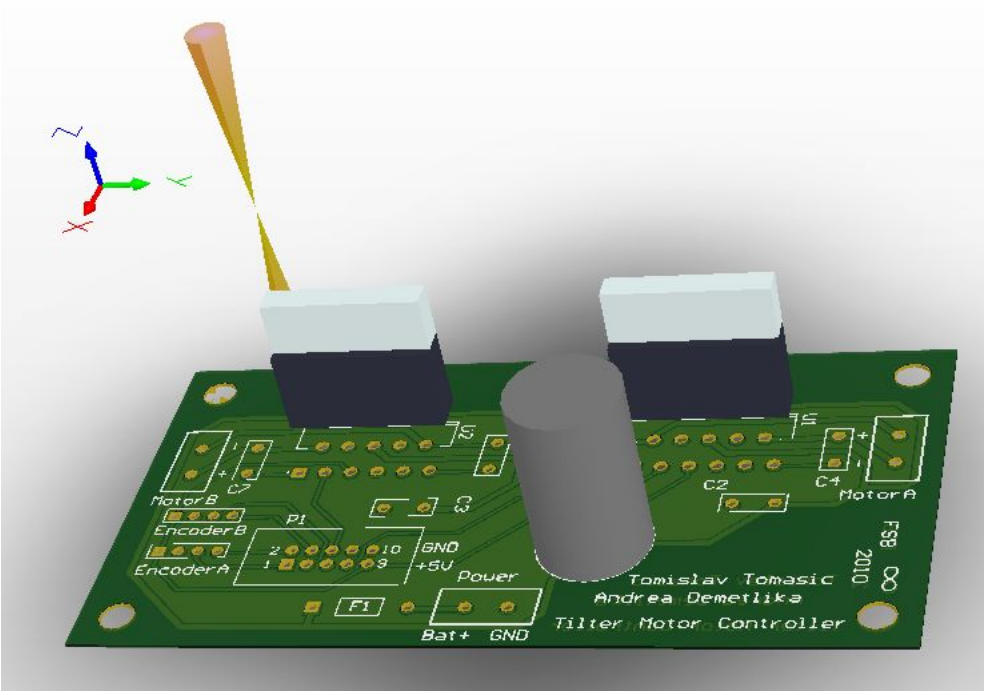
modul (slika 6.7) realiziran je na trećoj pločici, a komunicira sa mikrokontrolerom preko serijske veze.



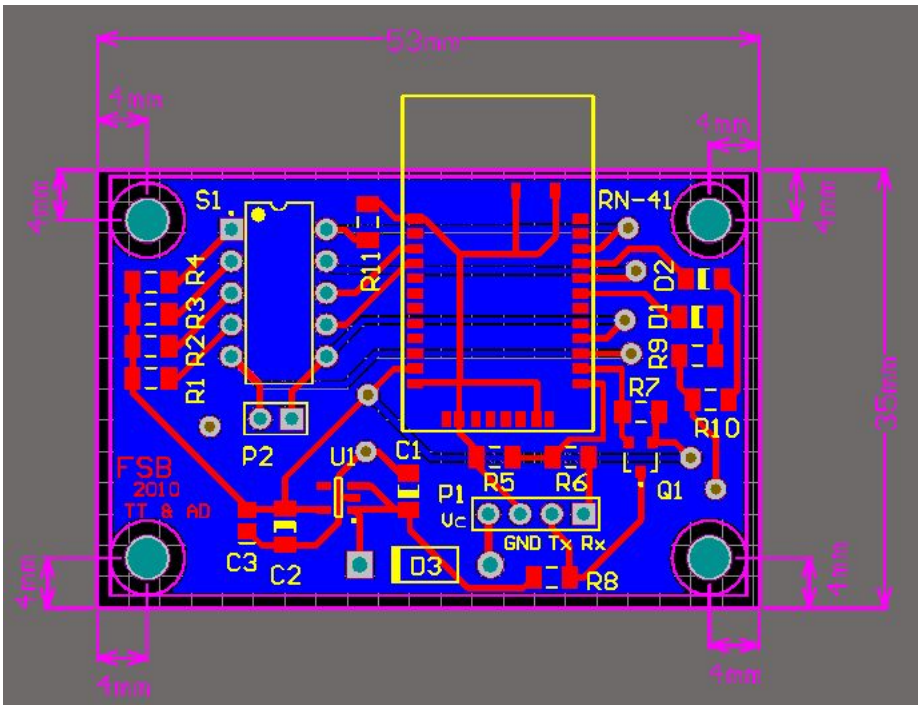
Slika 6.4: *Glavna tiskana pločica*



Slika 6.5: *3D prikaz glavne tiskane pločice*



Slika 6.6: *Motor kontroler*



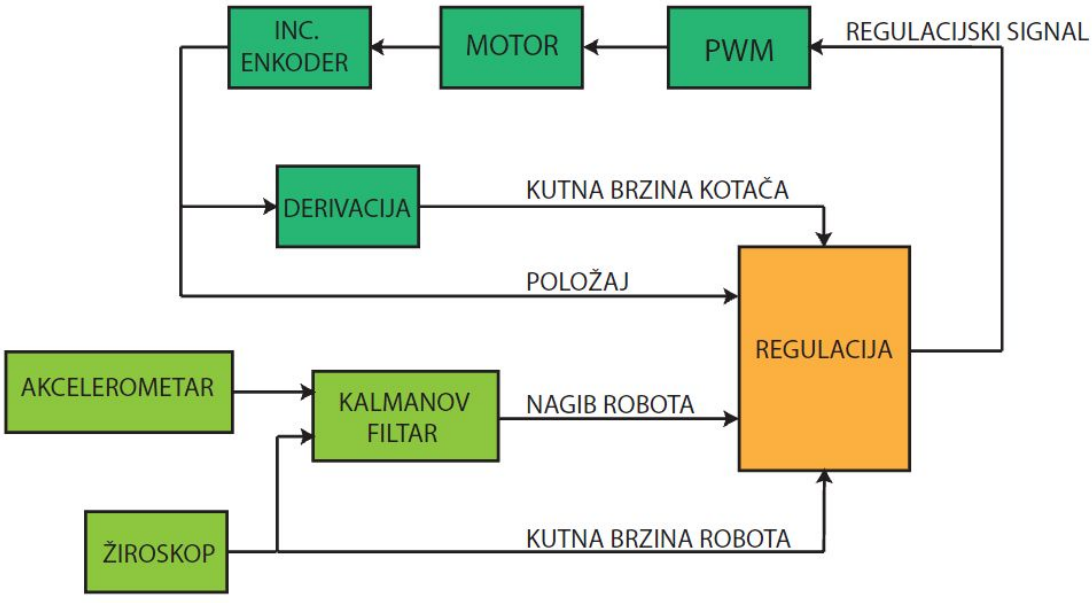
Slika 6.7: *Bluetooth komunikacija*

Poglavlje 7

Razvoj program

7.1 Programiranje mikrokontrolera

Glavni dio programa koji se odvija u mikrokontroleru odnosi se na regulaciju nagiba robota. Prvi korak je analogno digitalna pretvorba analognih izlaza sa akcelerometra i žiroskopa. Kako su izlazi senzora zašumljeni, te imaju probleme koji su detaljnije objašnjeni u prethodnim dijelovima, u drugom koraku izlazi se filtriraju koristeći diskretni Kalmanov filtar. Time se dobivaju točan kut nagiba i kutna brzina.



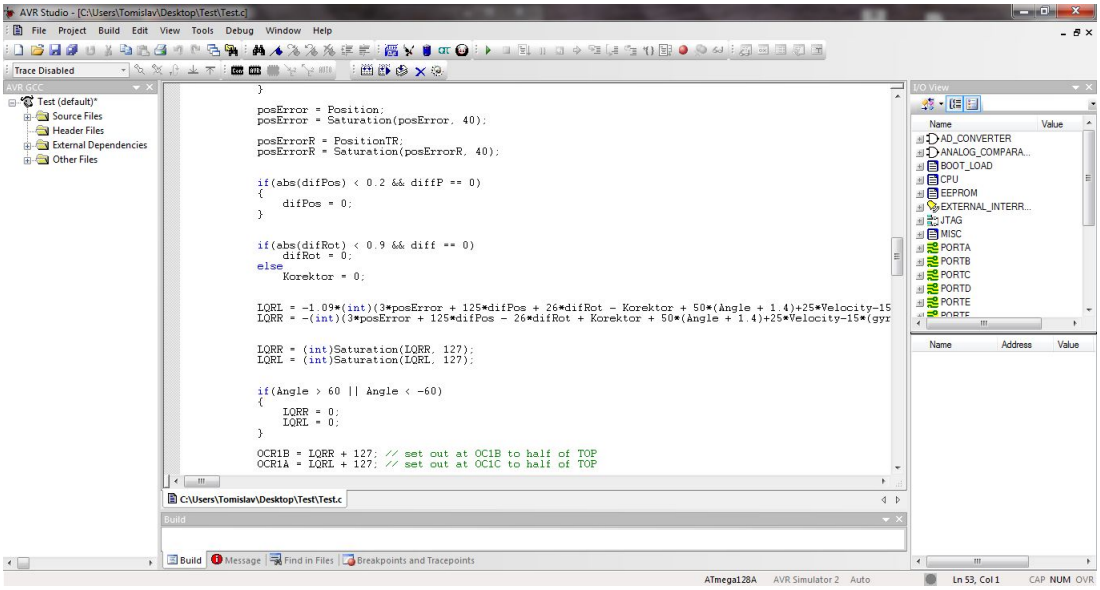
Slika 7.1: *Upravljački algoritam*

Nakon filtriranja, dobivene vrijednosti varijabli stanja robota su ulaz u diskretni LQR regulator koji izračunava vrijednost ”duty cycle-a” PWM signala koji se prenosi na motor kontroler.

Program koristeći vanjske ”interrupt-ove” čita impulse s dva kanala enkodera. Brojanjem impulsa dobiva se relativna pozicija robota u prostoru, a njihovim deriviranjem brzina robota.

Osim regulacije program preko serijske veze komunicira sa bluetoothom, čime je omogućena bežična komunikacija sa vanjskim uređajima. Preko bluetootha može se slati telemetrija na računalo, odnosno upravljati robotom. Na upravljačkoj pločici se nalaze tri LED diode koje služe za prikazivanje statusa programa koji se odvija u kontroleru.

Slika 7.1 shematskim blokovima prikazuje upravljački algoritam mobilnog robota. Kôd je napisan u programskom jeziku *C* unutar *AVR Studio* [15, 16] okruženja (slika 7.2).



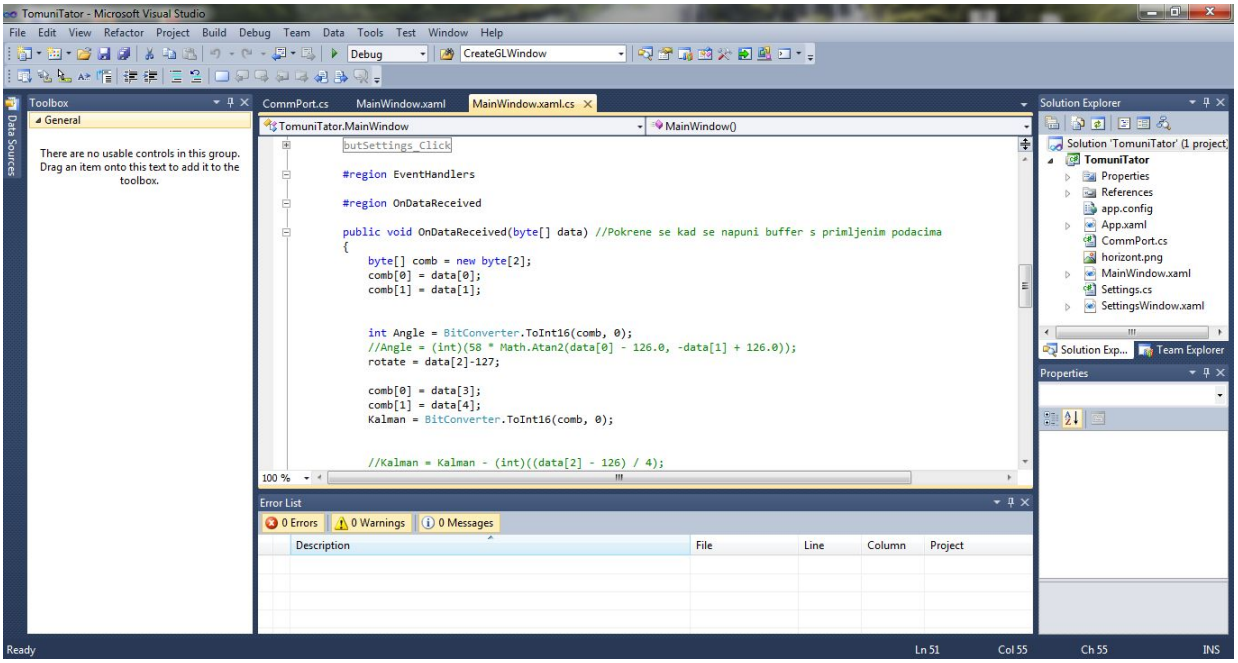
Slika 7.2: *Programski kôd u mikrokontroleru*

7.2 Sučelje na računalu

Da bi se robot upravljao i vizualizirali podaci njegovog trenutnog stanja napravljena je C# [17, 18] aplikacija (slika 7.3) koja preko bluetootha komunicira sa robotom. Ona preko virtualne serijske veze šalje i prima paketiće podataka koji se kasnije prevode u naredbe na robotu, odnosno na kut zakreta umjetnog horizonta na računalu.

Pritiskom tipke smjera na tipkovnici ili gumba u aplikaciji, robot se pokreće u odgovarajućem smjeru. Ako se aktivira gumb "Save Data", moguće je spremanje telemetrije s robota i kasnije analiziranje podataka u *Matlab-u*.

Slika 7.3: *Sučelje računalo-robot*



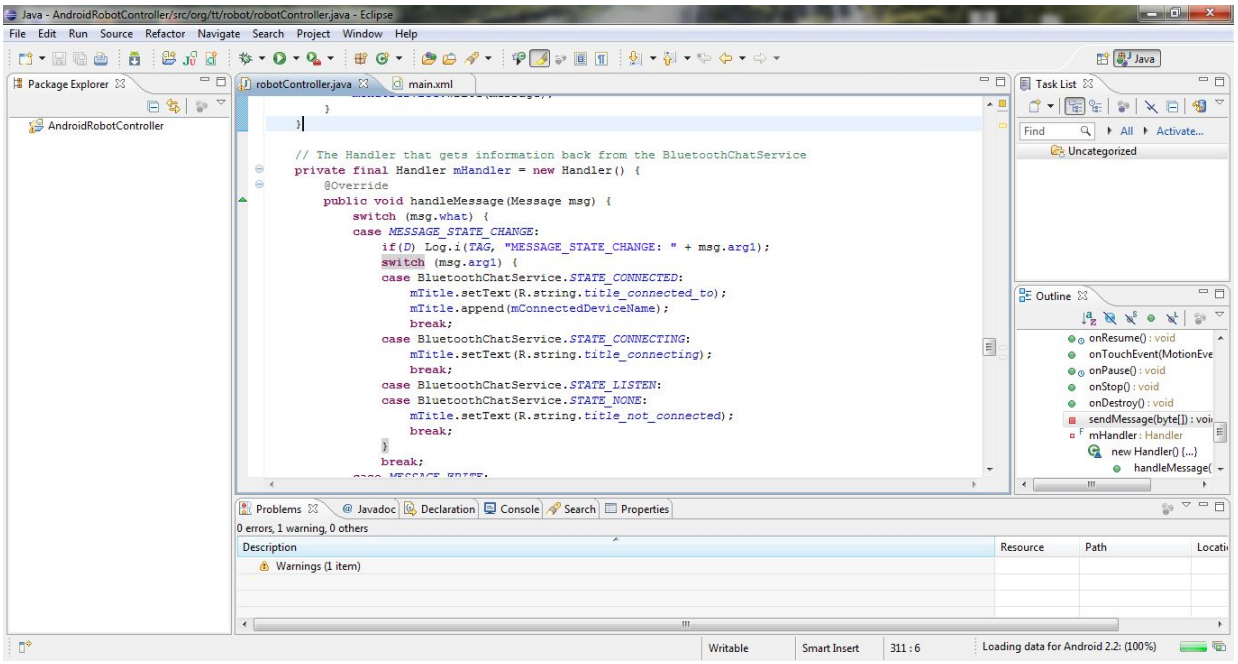
Slika 7.4: *Programsko okruženje Microsoft-ovog alata Visual Studio u kojem je napisan C# kôd*

7.3 Razvoj Android aplikacije

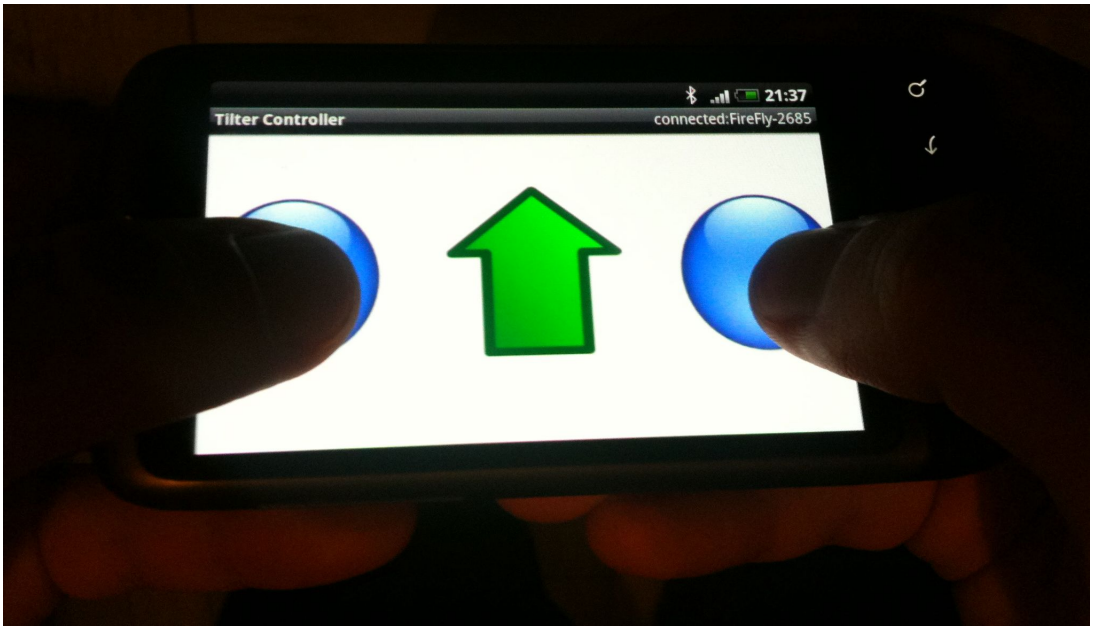
Kako bi se povećala autonomija robota i dala mogućnost upravljanja robota i u situacijama kada računalo nije prisutno, napravljena je *Android* aplikacija [19, 20] koja se odvija na mobitelu. Aplikacija je napravljena u razvojnom okruženju *Java* (slika 7.5).

Aplikacija prilikom nagiba mobitela (slike 7.6 i 7.7), preko akcelerometra ugrađenog u njega, određuje u kojem smjeru će se robot kretati i tu informaciju kao naredbu preko bluetootha šalje robotu.

Kako bi se smanjila mogućnost slučajnog nagiba mobitela u krivom smjeru (a time i vožnje robota u krivom smjeru), napravljen je zaštitni mehanizam u kojem korisnik treba istovremeno pritisnuti dva gumba čime aktivira upravljanje robotom.



Slika 7.5: *Programski kôd za Android aplikaciju*



Slika 7.6: *Prikaz upravljanja robota putem mobitela*

Slika 7.7: *Upravljanje robotom ovisno o nagibu mobitela*

Poglavlje 8

Zaključak

Samobalansirajući mobilni robot je interdisciplinarni projekt na kojem se susreću i integriraju strojarske, elektroničke, automatičarske i informatičke discipline. Pri tome je granice između pojedinih područja sve teže uočiti i upravo ih pojam mehatronike briše.

Robot se za razliku od klasičnih mobilnih robota sastoji od dva kotača, što znači da je njegova konstrukcija sama po sebi nestabilna te teži prevrtanju oko osi rotacije kotača. Djelovanjem motora, pri čemu se moment prenosi na kotače, robot se pokreće u odgovarajući smjer i time vraća u stabilan položaj.

U prvom dijelu rada napravljen je matematički model inverznog njihala i motora. Na temelju matematičkog modela, u drugom dijelu napravljen je *Simulink* model dinamike robota i njegova 3D vizualizacija, što je ostvareno u programskom paketu *Matlab*. To je preduvjet za područje automatike, pri čemu su projektirani ispitani PID, LQR, LQG i neizraziti regulatori, te su prikazani rezultati simulacije njihove primjene. Osim toga u ovom dijelu prikazana je problematika zašumljenih očitanja sa senzora, nakon čega je dano rješenje za njihovo filtriranje i estimaciju primjenom Kalmanovog filtra. Djelovi mehaničke konstrukcije modelirani su u *SolidWorks-u*, nakon čega su i izrađeni.

Osim mehaničke konstrukcije, posebno je važno područje elektronike. Tiskane pločice na kojima se nalaze senzori, mikokontroler, bluetooth modul, H-mostovi i ostale komponente, projektirane su u *Altium Designer-u*. Upravljačku jedinicu predstavlja mikrokontrler koji, na temelju informacija sa senzora i odgovarajućeg upravljačkog algoritma napisanog u programskom alatu *AVR Studio*, proračunava potreban PWM signal čime se određuje moment motora. Veza između robota i računala, te robota i mobitela ostvaruje se bluetooth komunikacijom.

Sučelje između računala i robota programirano je u programskom jeziku *C#*, a omogućava

upravljanje robotom i očitanje telemetrije u realnom vremenu. Koristeći Android aplikaciju programiranu u programskom paketu *Java*, moguće je upravljanje mobilnim robotom ovisno o kutu nagiba mobitela.

Dodatak A

Program u mikrokontroleru

```
1  #define F_CPU 16000000L // important for delay.h
2
3  #include <avr/io.h>
4  #include <util/delay.h>
5  #include <math.h>
6  #include <avr/interrupt.h>
7
8
9  //----------------------------Variables
       -------------------------------------------------
10 volatile int PositionL = 0; //Left wheel position
11 volatile int PositionR = 0; //Right wheel position
12 volatile char command = 0; //Command code
13 char dirL = 0; //Direction of left wheel
14 char dirR = 0; //Direction of right wheel
15
16
17 //----- ------------Function Prototypes
       ---------------------------------------------------------------
18 void Initialize(void); // Initialize stuff
19 void uart_putc(char data); // Send data
20 void SendTelemetry(float AngleM, char gyroRate, float Angle, int
       Position, float Velocity, int LQR, float gyro); //Send telemetry
21 float Saturation(float value, float top);
```

```
22
23 int main(void)
24 {
25   Initialize();
26
27   DDRG = 0xff;
28   PORTG = 0x00;
29   PORTG |= (1<<0);
30
31   char gyroRate = 0;
32   char AccX = 0;
33   char AccY = 0;
34
35   float AngleM = 0;
36   float Angle = 0;
37   float Velocity = 0;
38   int Position = 0;
39   int PositionTR = 0;
40   int Korektor = 0;
41
42   int posError = 0;
43   int posErrorR = 0;
44   float gyro = 0;
45   int LQRR = 0;
46   int LQRL = 0;
47   char count = 0;
48
49   float diff = 0;
50   float diffP = 0;
51
52   float difPos = 0;
53   float difRot = 0;
54
55   float gyroOffset = 0;
56
57   int countRec = 0;
```

```
58  float zero = 2;
59
60  while(1)
61  {
62    //_delay_ms(1);
63
64    ADMUX |= (1 << MUX2) | (1 << MUX1) | (1 << MUX0); //Set ADC Channel
          to 7
65    ADCSRA |= (1 << ADSC); //Start A2D conversion
66
67      AngleM = 59.1*atan2(AccX-127,-AccY+127);
68      //gyro = gyro * 0.95;
69      gyro = gyro + (gyroRate - 127 - gyro)/85.0;
70      Angle = Angle - ((gyro + gyroOffset) / 250.0);
71        Angle = Angle + (AngleM - Angle)/220.0;
72
73    while(ADCSRA & 0b01000000);
74    AccX = ADCH;
75
76    ADMUX &= ~(1<<MUX0);
77    ADMUX |= (1 << MUX2) | (1 << MUX1) | (0 << MUX0); //Set ADC Channel
          to 6
78    ADCSRA |= (1 << ADSC); //Start A2D conversion
79
80      float K = 1;
81      float Ts = 0.009;
82      difPos = difPos + (diffP - difPos * K)*Ts;
83      difRot = difRot + (diff - difRot * K)*Ts*5;
84
85      countRec++;
86
87
88      if(countRec > 100)
89      {
90        diff = 0;
91        diffP = 0;
```

```
92        }
93
94
95        if(command == 91)
96        {
97          command = 0;
98          zero += 0.1;
99        }
100       else if(command == 92)
101       {
102         command = 0;
103         zero -= 0.1;
104       }
105       else if(command == 101 && diff == 0)
106       {
107         countRec = 0;
108         diffP = -1;
109       }
110
111       else if(command == 104 && diff == 0)
112       {
113         countRec = 0;
114         diffP = 1;
115       }
116
117       else if(command == 102 && diffP == 0)
118       {
119         diff = 1;
120         countRec = 0;
121       }
122
123       else if(command == 103 && diffP == 0)
124       {
125         diff = -1;
126         countRec = 0;
127       }
```

```
128
129         if(diff != 0 || diffP != 0)
130         {
131           command = 0;
132           Position = 0;
133           PositionTR = 0;
134         }
135
136         posError = Position;
137         posError = Saturation(posError, 40);
138
139         posErrorR = PositionTR;
140         posErrorR = Saturation(posErrorR, 40);
141
142
143         if(abs(difPos) < 0.2 && diffP == 0)
144         {
145           difPos = 0;
146         }
147
148
149         if(abs(difRot) < 0.9 && diff == 0)
150           difRot = 0;
151         else
152           Korektor = 0;
153
154         //LQR algorithm
155         LQRL = -1.09*(int)(3*posError + 126*difPos + 26*difRot - Korektor
                + 50*(Angle + zero)+25*Velocity-15*(gyro + gyroOffset));
156         LQRR = -(int)(3*posError + 126*difPos - 26*difRot + Korektor +
               50*(Angle + zero)+25*Velocity-15*(gyro + gyroOffset));
157
158
159         LQRR = (int)Saturation(LQRR, 127);
160         LQRL = (int)Saturation(LQRL, 127);
161
```

```
162
163         if(Angle > 60 || Angle < -60)
164         {
165           LQRR = 0;
166           LQRL = 0;
167         }
168
169         OCR1B = LQRR + 127; // set out at OC1B to half of TOP
170           OCR1A = LQRL + 127; // set out at OC1C to half of TOP
171
172       while(ADCSRA & 0b01000000);
173       AccY = ADCH;
174
175       ADMUX &= ~(1<<MUX2);
176       ADMUX |= (0 << MUX2) | (1 << MUX1) | (1 << MUX0); //Set ADC Channel
              to 3
177       ADCSRA |= (1 << ADSC); //Start A2D conversion
178
179         count++;
180         if(count == 30)
181         {
182           count = 0;
183           SendTelemetry(AngleM, gyroRate, Angle*10, Position, Velocity
                *10, PositionTR, gyro*100);
184         }
185
186         Position += PositionL;
187
188         PositionTR += PositionR;
189
190         Korektor += PositionL - PositionR;
191         Korektor = (int)Saturation(Korektor, 50);
192
193
194         Velocity = Velocity + ((PositionL + PositionR)*10 - Velocity)
              *0.01;
```

```
195
196         Position = (int)Saturation(Position, 800);
197         PositionTR = (int)Saturation(PositionTR, 800);
198
199         PositionL = 0;
200         PositionR = 0;
201
202       while(ADCSRA & 0b01000000);
203       gyroRate = ADCH;
204     }
205
206     return 0;
207 }
208
209
210 //-------------------------------------Saturation
      ---------------------------------------------------------------------
211 float Saturation(float value, float top)
212 {
213     if(value > top)
214       value = top;
215     else if(value < -top)
216       value = -top;
217
218     return value;
219 }
220
221
222 //-----------------------------------Send telemetry
      ---------------------------------------------------------------------
223 void SendTelemetry(float AngleM, char gyroRate, float Angle, int
      Position, float Velocity, int LQR, float gyro)
224 {
225     uart_putc(252); // Indicates start of communication
226
227     uart_putc((int)AngleM&0x00FF);
```

```
228   uart_putc(((int)AngleM>>8)&0x00FF);
229
230   uart_putc(gyroRate);
231
232   uart_putc((int)Angle&0x00FF);
233   uart_putc(((int)Angle>>8)&0x00FF);
234
235   uart_putc(Position&0x00FF);
236   uart_putc((Position>>8)&0x00FF);
237
238   uart_putc((int)Velocity&0x00FF);
239   uart_putc(((int)Velocity>>8)&0x00FF);
240
241   uart_putc(LQR&0x00FF);
242   uart_putc((LQR>>8)&0x00FF);
243
244   uart_putc((int)gyro&0x00FF);
245   uart_putc(((int)gyro>>8)&0x00FF);
246
247 }
248
249
250 //-------------------------------------Initialize things
      ------------------------------------------------------------
251 void Initialize()
252 {
253   //---------------------Initialize PWM
        ------------------------------------------------------------------------


254   DDRB |= (1 << DDB5)|(1 << DDB6); //Set pins 5 and 6 as output
255
256   TCCR1A |= (1 << COM1B1)|(1 << COM1A1); // Ask for non-inverted PWM on
        OC1B and OC1C
257   TCCR1B |= (1 << WGM13)|(1 << CS10); // Configure timer 1 for Phase
      and Frequency Correct PWM mode, with no prescaling
258   ICR1 = 255; //Set period to 20ms //// the PWM frequency to 1kHz ////
```

```
    fout = fclk / (2 * N * TOP) ==== 8kHz
259
260 OCR1A = 127;
261 OCR1B = 127; // set out at OC1B to half of TOP
262
263
264 //----------------------Initialize ADC
    ------------------------------------------------------------------
265 ADCSRA |= (1 << ADPS2)|(1 << ADPS1)|(1 << ADPS0); // set prescaler to
    128 - 125KHz sample rate @ 16MHz
266 ADMUX |= (1 << REFS0); //Set ADC reference voltage to AVCC
267 ADMUX |= (1 << ADLAR); //Select 8-bit resolution of ADC
268 ADCSRA |= (1 << ADEN); //Enable ADC
269
270
271 //----------------------Initialize USART1 on port D (Bluetooth)
    ---------------------------------------------------------
272 UCSR1B |= (1<<RXEN1)|(1<<RXCIE1)|(1<<TXEN1); //Rx and Tx enable, 8
    bit characters
273 UCSR1C |= (1<<UCSZ11)|(1<<UCSZ10); //Async operation, no parity, one
    stop bit, 8-bit characters
274 UBRR1H=0x00;
275 UBRR1L=0x67; // Set baud rate to 9600
276
277
278 //----------------------Initialize USART0 on port B (USB)
    ---------------------------------------------------------
279 //UCSR0B |= (1<<RXEN0) | (1<<TXEN0); //Rx and Tx enable, 8 bit
    characters
280 //UCSR0C |= (1<<UCSZ01) | (1<<UCSZ00); //Async operation, no parity,
    one stop bit, 8-bit characters
281   //UBRR0H=0x00;
282 //UBRR0L=0x67; // Set baud rate to 9600
283
284
285 //----------------------Initialize external interrupts
```

```
286   EICRB |= (1<<ISC70)|(1<<RXCIE1)|(1<<ISC41); //set interrupts to fire
          on rise
287   EIMSK |= (1<<INT7)|(1<<INT4); // enable interrupts pins
288   sei(); //enable interrupts
289 }
290
291
292 //-------------------------------------Send over bluetooth
      ------------------------------------------------------
293 void uart_putc(char data){
294   while (!(UCSR1A&(1<<UDRE))); // Wait for previous transmission
295   UDR1 = data; // Send data byte
296 }
297
298
299 //-------------------------------------Bluetooth recive
      ------------------------------------------------------
300 SIGNAL(SIG_USART1_RECV)
301 {
302   command = UDR1;
303 }
304
305
306 //-------------------------------------Left wheel encoder interrupt
      ----------------------------------------
307 ISR(INT7_vect)
308 {
309   if(PINE & (1<<7))
310   {
311     if(PINE & (1<<6))
312       PositionL++;
313     else
314       PositionL--;
315   }
316   else
```

```
317   {
318     if(PINE & (1<<6))
319       PositionL--;
320     else
321       PositionL++;
322   }
323 }
324
325 ISR(INT4_vect)
326 {
327   //if(PINE & (1<<4))
328   //{
329     //PORTG |= (1<<1);
330     if(PINE & (1<<5))
331       PositionR++;
332     else
333       PositionR--;
334   //}
335   //else
336   //{
337     //PORTG &= ~(1<<1);
338   //   if(PINE & (1<<5))
339   //     PositionR--;
340   //   else
341   //     PositionR++;
342   //}
343 }
```

Literatura

[1] D.J. Block, K.J. Astrom, and M.W. Spong. *The Reaction Wheel Pendulum*. Morgan and Claypool Publishers, University of Illinois at Urbana-Champaign, 2007.

[2] F. Grasser, A.D'Arrigo, S.Colombi, and A.Rufer. *A mobile, Inverted Pendulum*. Laboratory of Industrial Electronics Swiss Federal Institute of Technology Lausanne.

[3] T. Žilić, D. Pavković, and D. Zorc. *Modeling and control of a pneumatically actuated inverted pendulum*. ScienceDirect, Zagreb, 2009.

[4] Matlab. *Simulink - Model-Based and System-Based Design*. The MathWorks, 2002.

[5] http://www.analog.com/en/sensors/inertial-sensors/adxl203/products/tools-software-simulation models/index.html. 26.4.2010.

[6] Ligos Corporation. *V·Realm^{TM} Builder - User's Guide and Reference*. 1996.

[7] C. Kilian. *Modern Control Technology*. Delmar Cengage Learning, 2005.

[8] R. F. Stengel. *Optimal Control and Estimation (Dover Books on Advanced Mathematics)*. Dover Publications, 1994.

[9] H.B. Mitchell. *Multi-Sensor Data Fusion*. Springer, New York, 2007.

[10] Jose' Luis Corona Miranda. *Application of Kalman filtering and PID control for direct inverted pendulum control*. Faculty of California State University, Chico, 2009.

[11] G. Welch and G. Bishop. *An Introduction to the Kalman Filter*. Department of Computer Science University of North Carolina at Chapel Hill, 2006.

[12] X. Chen, H. Zhou, R. Ma, F. Zuo, G. Zhai, and M. Gong. *Linear Motor Driven Inverted Pendulum and LQR Controller Design*. College of Engineering China Agricultural University Beijing, China, 2007.

[13] Matlab. *Fuzzy Logic Toolbox 2 - User's guide*. The MathWorks, 2010.

[14] J. Catsoulis. *Designing Embedded Hardware*. O'Reilly Media, 2005.

[15] R. H. Barnett, S. Cox, and L. O'Cull. *Embedded C Programming and the Atmel AVR*. Delmar Cengage Learning, 2006.

[16] John Morton. *AVR: An Introductory Course*. Newnes, 2002.

[17] P. Kimmel. *Advanced C# Programming*. McGraw-Hill/OsborneMedia, 2002.

[18] A. Troelsen. *Pro C# 2010 and the .NET 4 Platform*. Apress, 2010.

[19] J. Steele and N. To. *The Android Developer's Cookbook: Building Applications with the Android SDK (Developer's Library)*. Addison-Wesley Professional, 2010.

[20] M. Gargenta. *Learning Android*. O'Reilly Media, 2011.

Sažetak

Andrea Demetlika, Tomislav Tomašić

Samobalansirajući mobilni robot

Samobalansirajući mobilni robot na dva kotača zvan Tilter radi na principu obrnutog njihala, poput popularnog električnog vozila Segway. Konstrukcija robota je sama po sebi nestabilna, teži prevrtanju oko osi rotacije kotača, pa se djelovanjem motora robot pokreće u odgovarajući smjer i time vraća u uspravni položaj.

Da bi se na kotače djelovalo pravilnim iznosom momenta potrebna je točna informacija o trenutnom kutu nagiba. Ona se dobiva kombiniranjem zašumljenih izlaza senzora akcelerometra i žiroskopa pomoću Kalmanovog filtra. Koristeći te podatke LQR (engl. *Linear quadratic regulator*) regulacijski algoritam proračunava potrebni moment motora koji će djelovati na kotače kako bi robot ostao u ravnoteži. Cijeli algoritam sadržan je u upravljačkoj jedinici robota.

U radu je objašnjen postupak realizacije ovog projekta od projektiranja pojedinih komponenti, preko njihove izrade do integracije elektroničkog, mehaničkog i programskog dijela. Zbog potrebe za korištenjem znanja iz područja mehanike, elektronike, programiranja i regulacije ovaj projekt je izrazito interdisciplinaran i kao takav predstavlja jedan od najpoznatijih mehatroničkih problema.

Robot se može upravljati putem računala ili mobitelom koristeći bluetooth komunikaciju. Telemetrija sa robota može se u realnom vremenu prikazati u grafičkom sučelju na računalu. Prikazuju se trenutne veličine robota kao što su pozicija, brzina, kut nagiba, kutna brzina, i temperatura okoline.

Samobalansirajući mobilni robot predstavlja dobru platformu za projektiranje i ispitivanje naprednih regulacijskih i estimacijskih algoritama, te kao takav može se koristiti za daljnja znanstvena istraživanja.

Ključne riječi: Samobalansirajući, mobilni robot, LQR, Kalmanov filtar, bežična komunikacija.

Abstract

Andrea Demetlika, Tomislav Tomašić

Selfbalancing mobile robot

Selfbalancing mobile robot on two wheels called Tilter works on the principle of the inverted pendulum, similar to the popular electric vehicle Segway. Design of the robot is inherently unstable, without external control it will roll around the axis of rotation of the wheels. Thus driving the motors in the right direction will return the robot to the upward position.

To apply the correct moment to the motors, it is necessary to know the accurate value of robot's tilt angle and position. It can be calculated by combining the noisy accelerometar and gyroscope signal using the Kalman filter algorithm. Using this data LQR (Linear quadratic regulator) control algorithm determines the necessary moment of the motor to act on the wheels to balance the robot. The entire algorithm is contained in the robot control unit.

In this work entire procedure in which the project has been realized is explained, from design of the specific parts, their production, to integration of electronic, mechanical and the programming section. Because of the need to use the knowledge in fields of mechanics, electronics, programming and control this project is extremely interdisciplinary and as such it represents one of the most representative mechatronic problems.

Robot is controlled with a computer or a cell phone using the bluetooth communication. Telemetry from the robot can be displayed in real time on the computer using a graphical interface. The values that can be monitored are position, velocity, tilt angle, angular velocity, and ambient temperature.

Selfbalancing mobile robot is an excellent platform for design and testing of advanced control and estimation algorithms, and as such it can be used for further scientific work.

Key words: Selfbalancing, mobile robot, LQR, Kalman filter, wireless communication.