Samo-balansirajuća robotska kolica

Prendivoj, Dominik

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij elektrotehnike

Završni rad

SAMO-BALANSIRAJUĆA ROBOTSKA KOLICA

Rijeka, rujan 2023. Dominik Prendivoj

0069089488

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij elektrotehnike

Završni rad

SAMO-BALANSIRAJUĆA ROBOTSKA KOLICA

Mentor: doc. dr. sc. Ivan Volarić

Rijeka, rujan 2023.

Dominik Prendivoj

0069089488

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 20. ožujka 2023.

Zavod: **Zavod za automatiku i elektroniku**
Predmet: **Elementi automatizacije postrojenja**
Grana: **2.03.03 elektronika**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Dominik Prendivoj (0069089488)**
Studij: Sveučilišni prijediplomski studij elektrotehnike

Zadatak: **Samo-balansirajuća robotska kolica / Self-balansing robotic car**

Opis zadatka:

Potrebno je izraditi sustav koji se sastoji od vertikalnih robotska kolica koja stoje na dva kotača, te održavaju ravnotežu u uspravnom položaju. Pomak robotskih kolica iz ravnotežnog položaja mjeri se pomoću akcelerometra (npr. MPU-6050 modul) spojenog na Arduino platformu. Na Arduino također su spojena dva istosmjerna motora za pogon robotskih kolica, dok je pomoću H-mosta (npr. L298N modul) moguće kontrolirati smjer i brzinu vrtnje istosmjernih motora. Mikrokontroler je potrebno isprogramirati na način da DC motor pokreće kolica u smjeru suprotnom od pomaka akcelerometra, održavajući kolica u uspravnom ravnotežnom položaju.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Dominik Prendivoj

Zadatak uručen pristupniku: 20. ožujka 2023.

Mentor:

Doc. dr. sc. Ivan Volarić

Predsjednik povjerenstva za završni ispit:

Prof. dr. sc. Dubravko Franković

IZJAVA

Izjavljujem da sam samostalno izradio ovaj završni rad pod nazivom „Samo-balansirajuća robotska kolica“ uz pomoć stečenog znanja tijekom školovanja, navedene literature i mentora doc. dr. sc. Ivana Volarića.

Rijeka, rujan 2023.

Dominik Prendivoj

0069089488

Sadržaj

1. UVOD

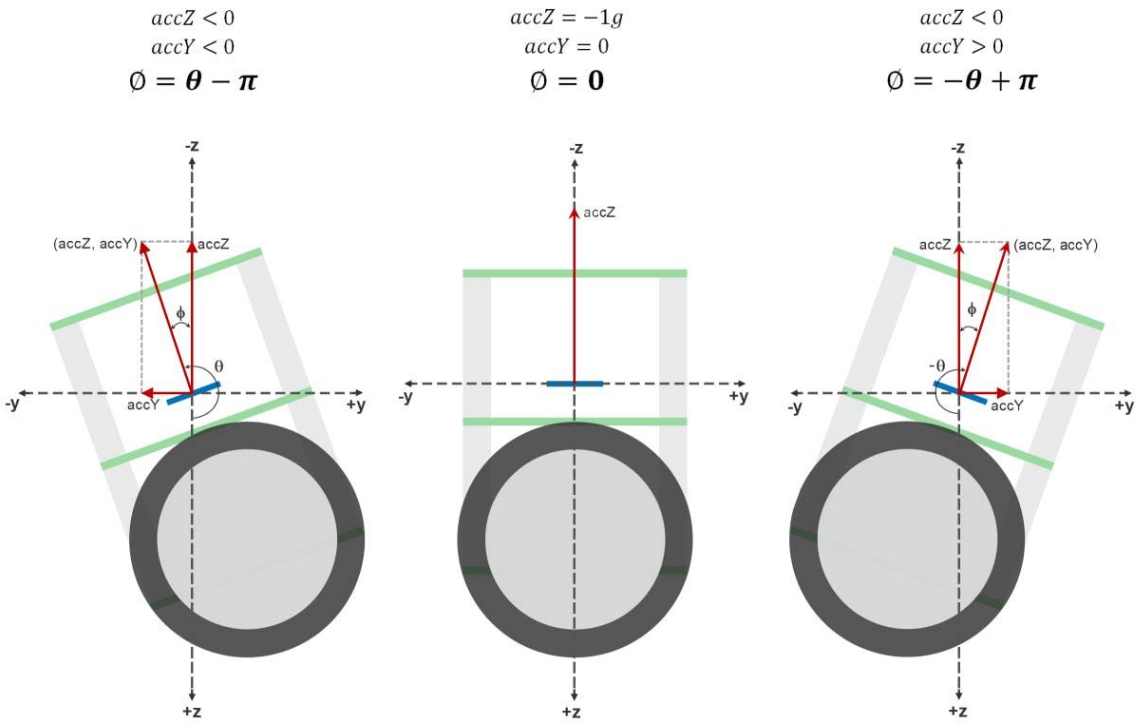
Cilj izrade ovog samo-balansirajućeg robota ja prikaz važnosti robotike i automatizacije u suvremeno doba. Zbog sve bolje i brže komunikacije u svijetu raste i potreba za bržim i boljim pametnim uređajima, postrojenjima i slično. Tu nastupaju robotika i automatizacija kao rješenje za potrebe današnjeg tržišta.

Robotika se zasniva na mehatronici koja objedinjuje znanja iz strojarstva, elektrotehnike, elektronike, automatizacije i računalstva. Najraširenija grana robotike je industrijska robotika koja sve više zamjenjuje čovjeka u rutinskim poslovima i poslovima koji su opasni po život. O sve većoj potražnji robota govori i podatak da je u svijetu 2007. godine bilo oko 6,5 milijuna robota, a 2011. oko 18 milijuna. Najveći broj tih robota korišten je u automobilskoj i elektroničkoj industriji. U Hrvatskoj se može naći industrijskih robota, ali se više proizvode roboti za specijalne primjene (npr. razminiranje) i robotske komponente za druge proizvođače [1].

Automatizacija se bavi upravljanjem nekog procesa ili sustava i zamjenjuje čovjeka u previše složenim i repetivnim poslovima. Razvijanje raznih strojeva započeto je prvom industrijskom revolucijom u periodu mehanizacije prije otprilike dva stoljeća. Razvojem strojeva došlo je do potrebe za njihovim upravljanjem jer su neki strojevi bili previše složeni za čovjeka i tako je započelo doba automatizacije i druge industrijske revolucije. Kako bi se neki proces upravljao treba ga regulirati da bi bio učinkovit. Zadatak regulacije je održavati proces od vanjskih i unutarnjih poremećaja tako da se pomoću povratne veze vraća izlazna vrijednost na ulaz i uspoređuje s zadanim vrijednostima na ulazu i zatim po potrebi regulira. Danas se automatizacija koristi u svim postrojenjima, a jedan od najvećih stupnjeva automatizacije imaju energetska postrojenja čak do 90%. U procesnoj industriji kao što je petrokemija stupanj automatizacije iznosi između 60 i 80%, dok u metaloprerađivačka industrija ima i danas mali stupanj automatizacije [2].

2. PRINCIP RADA I PRIMJENA

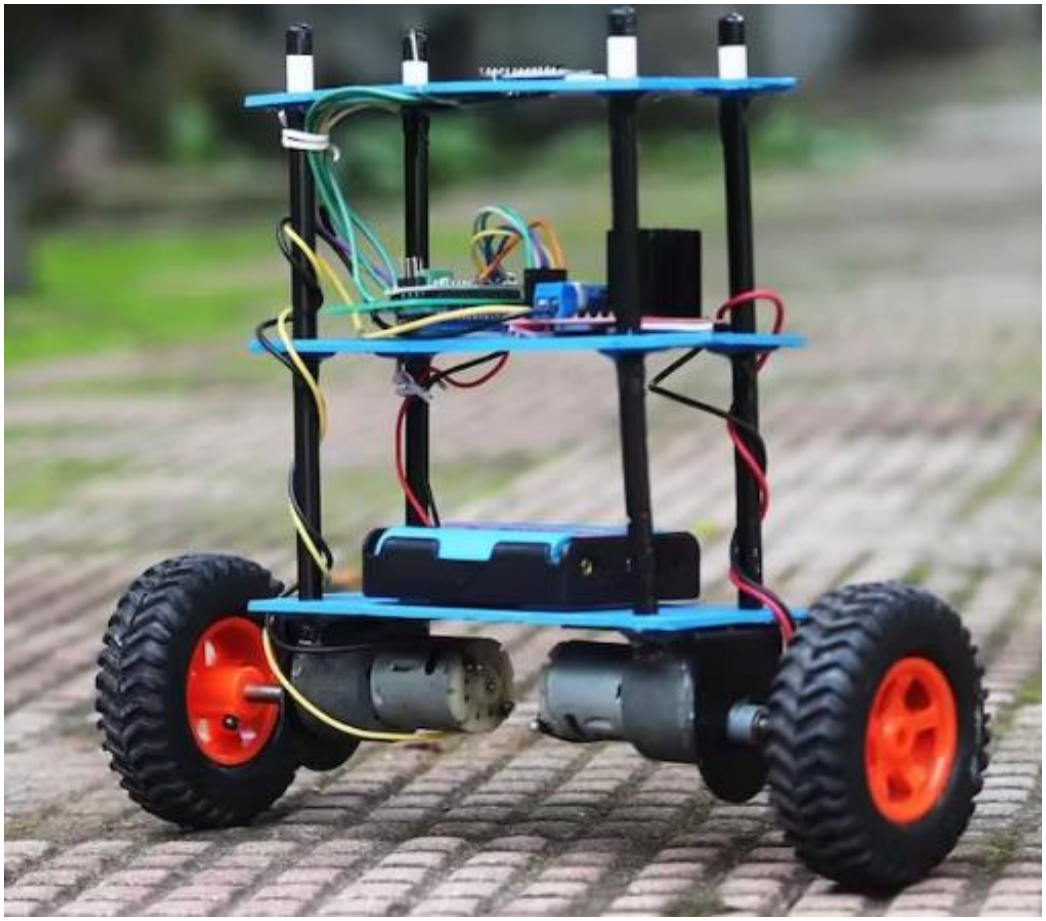
Princip rada samo-balansirajućeg robota zasniva se na PID regulaciji koja je opisana u nastavku. Samo-balansirajući robot je robot kojemu konstrukcija stoji na dva kotača i time teži prevrtanju. Kako se ne bi prevrnuo balansiraju ga dva motora koja djeluju u suprotnom smjeru od smjera prevrtanja. Težište robota spušta se što niže zbog lakšeg balansiranja tako da se na dnu nalaze motori i upravljač za motore, a mogu biti i baterije ukoliko su teške. Na vrhu konstrukcije nalazi se senzor pomaka jer je vrh konstrukcije najudaljeniji od osi kotača oko koje se okreće robot i tu senzor najbrže osjeti pomak robota [3].



Slika 2.1. Bočni prikaz robota

U ovom radu koriste se dva DC motora s reduktorom, a njihovo upravljanje vrši se pomoću H-mosta L298N. Pomak vrha robota se mjeri pomoću senzora MPU6050 i podatci s tog senzora šalju se na razvojnu pločicu Arduino UNO koja te podatke obrađuje i pomoću H-mosta L298N upravlja s motorima.

Primjena samo-balansirajućih robota nije previše popularna jer takvi roboti troše mnogo energije u odnosu na ono što pružaju. Ipak samo-balansirajuće robote možemo naći u nekim primjerima transporta. Balans nekog robota može se lako postići dodavanje kotača i nema potrebe za kompliciranim upravljanjem, ali ono što čini ovaj robot popularnim je njegov program koji se koristi kod drugih robota za balansiranje i služi kao podloga za testiranje programa [3].

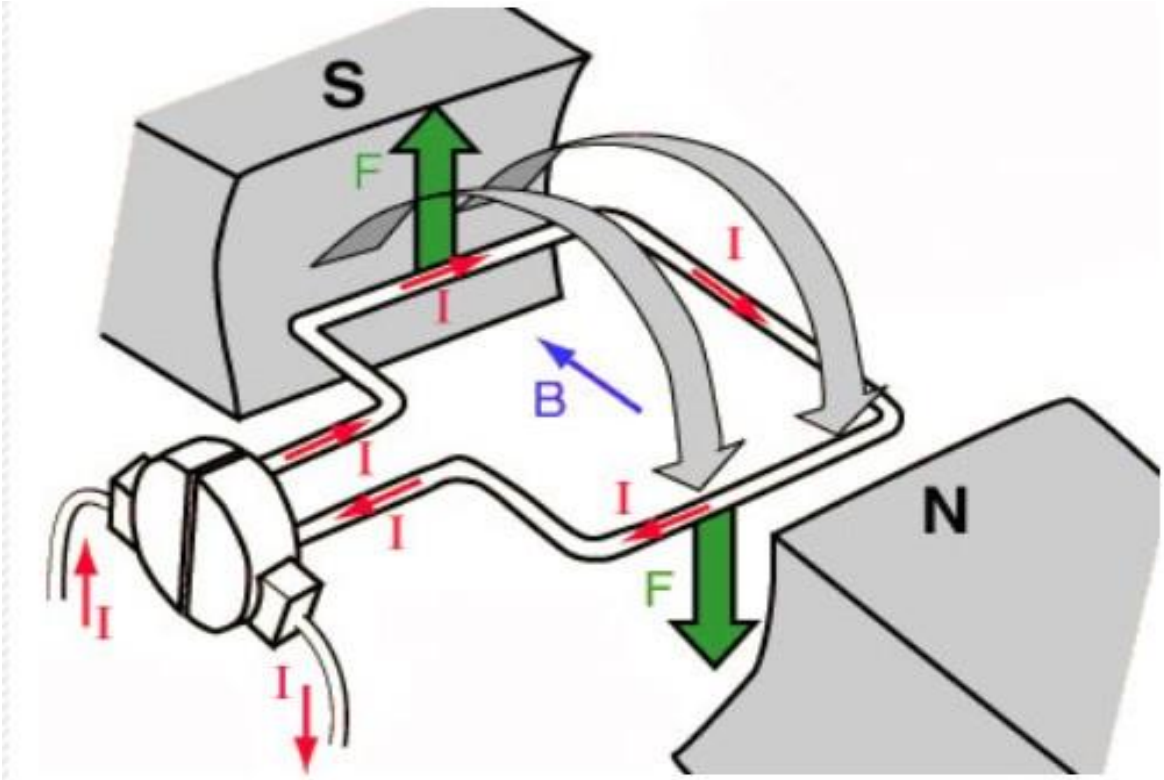


Slika 2.2. Primjer samo-balansirajućeg robota

3. KOMPONENTE SAMO-BALANSIRAJUĆEG ROBOTA

3.1. DC motori s redukcijom

Istosmjerni (DC) motor je elektromehanički stroj koji istosmjernu električnu energiju pretvara u kružno gibanje, a može raditi i obratno. Kao i svaki motor ima rotor i stator, ali ima i komutator ili popularnije kolektor. Ovisno o spajanju kolektora s uzbudom na statoru dijeli se na istosmjerne motore s nezavisnom, serijskom, paralelnom ili složenom uzbudom. Radi na principu Lorencove sile koja se javlja na vodič kad njime protječe struja i nalazi se u magnetsko polju. Na statoru se nalaze namoti kroz koje prolazi istosmjerna struja. Svaki namot stvara sjeverni ili južni magnatski pol ovisno o smjeru namatanja i tako nastaje konstantno uzbudno magnetsko polje statora. Na rotoru se isto nalaze namoti koji su spojeni na istosmjerno napajanje preko kolektorskih četkica. Kad kroz rotorski namotaj proteče struja dolazi do pojave Lorencove sile na vodič i zakretanja. Kad se rotor okrene, struja na četkicama je i dalje istog smjera, ali se zbog zakreta rotora, promijenio smjer struje kroz rotorske namotaje, stvarajući suprotno polarizirano magnetsko polje omogućavajući nastavak zakretanja rotora u istom smjeru [4].



Slika 3.1. Princip rada istosmjernog motora

Istosmjerni motori imaju dva izvoda koja se spajaju na istosmjerno napajanje. Ako zamijenimo plus i minus napajanja motor će se početi okretati u suprotnom smjeru, a ako smanjimo napon napajanja motoru će se smanjiti brzina. Te karakteristike čine ovaj motor lako upravljivim i zato je ovaj motor i dalje u širokoj upotrebi iako ima kolektor i četkice koje se troše i stvaraju pad napona.

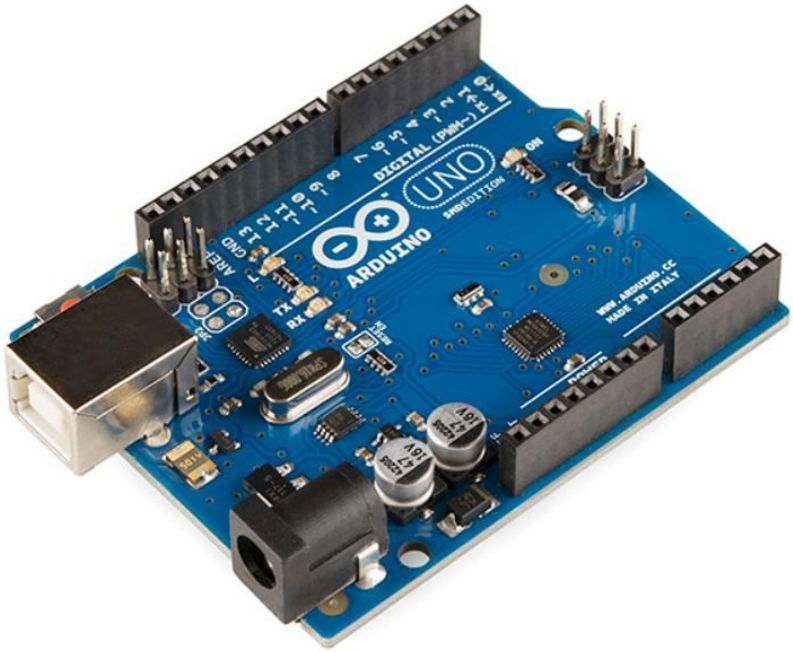
Kod malih DC motora stator je napravljen od permanentnih magneta i napaja se samo rotor motora. Takav motor je korišten i u ovome radu, ali s dodatkom reduktora broja okretaja jer se kotači robota trebaju sporo okretati, a broj okretaja ovakvog motora iznosi nekoliko tisuća.

Izabran je DC motor s rektorom koji radi na 6 V. Ovaj motor ima dovoljno jak moment za održavanje balansa robota i zato nema potrebe za jačim. Radi od 1.5 V do 12 V, ali najpovoljnije radi od 3 V do 6 V. Broj okretaja na izlazu redukcije je 130 okretaja po minuti što je dovoljno sporo za kontrolirano upravljanje robota. Maksimalna struja motora iznosi 250 mA pri maksimalnom naponu od 12 V, a minimalna 70 mA pri naponu 3 V. Dimenzije ovog motora su male, a težina motora je samo 80 grama što ga čini odličnim za ovog robota [5].

Slika 3.2. Izabrani DC motor s redukcijom

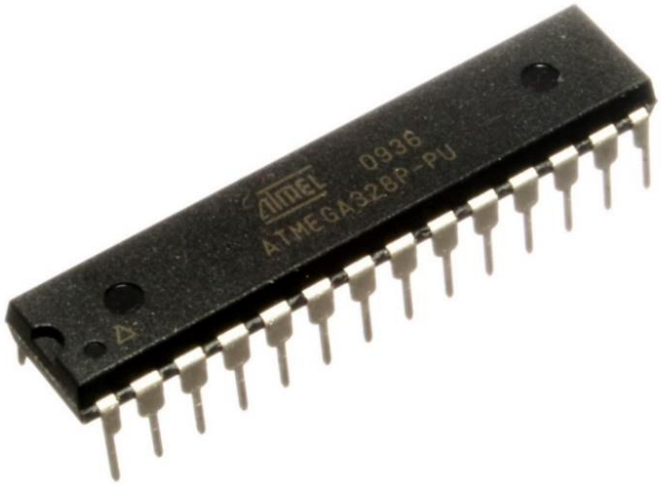
3.2. Arduino Uno R3

Arduino Uno R3 je razvojna pločica talijanske tvrtke Arduino koja je prvenstveno razvijena za početnike u elektronici i za male hobi projekte, ali se često koristi kao probna elektronička pločica u mnogim projektima zbog svoje jednostavnosti i lakog programiranja. Također cijena Arduino Uno R3 i drugih proizvoda tvrtke Arduino je vrlo pristupačna. Mnogo senzora i upravljačkih pločica su kompatibilni s Arduino proizvodima, pa je to još jedan razlog široke upotrebe u mnogim radovima kao što je i ovaj [6].



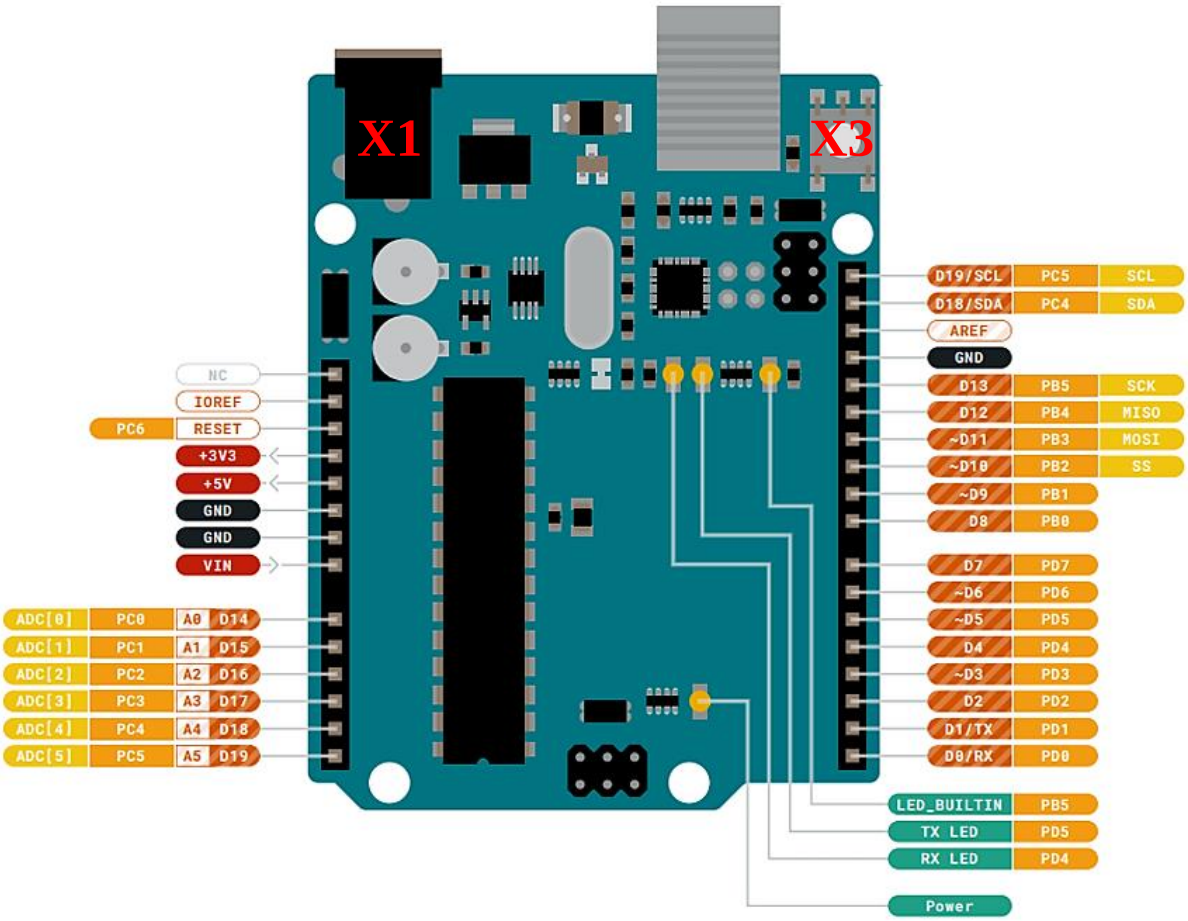
Slika 3.3. Primjer Arduino Uno R3

Glavni mikrokontroler na Arduino Uno R3 pločici je ATmega328p tvrtke Atmel. Mikrokontroler ATmega328p je 8 bitni AVR procesor koji ima maksimalnu brzinu od 20 MHz, 1 KB EEPROM memorije, 2 KB SRAM memorije i 1 MIPS na 1 MHz. Ovaj mikrokontroler se koristi i u autonomnim sustavima kad je potreban jednostavan i jeftin mikrokontroler, a najčešća upotreba je baš kod proizvoda tvrtke Arduino odnosno na razvojnim pločicama Arduino Pro Mini, Arduino Nano i Arduino Uno R3 koji se koristi u ovome radu [7].



Slika 3.4. Mikro kontroler ATmega328p

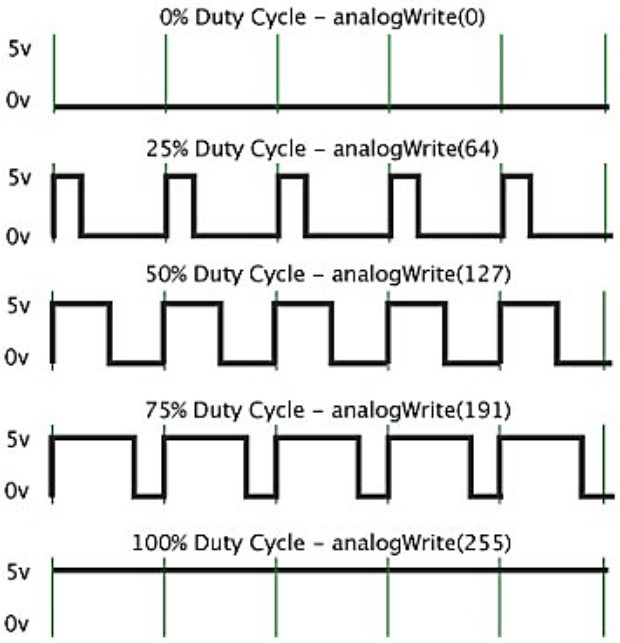
Na Arduino Uno R3 pločici možemo naći mnogo pinova i izvoda pomoću kojih se kontrolira mikrokontroler i pomoću kojih mikrokontroler upravlja drugim komponentama. Prikaz tih pinova prikazan je na slici 3.5. [8].



Slika 3.5. Prikaz pinova Arduino Uno R3 pločice

Na slici 3.5. možemo vidjeti razne LED diode koje služe za prikaz stanja pločice kao što je napajanje ili LED TX i RX koje služe za prikaz prijenosa programa na pločicu i prikaz komunikacije. Zatim imamo prikazane pinove napajanja koji služe za napajanje pločice ili i za napajanje drugih komponenta. Pin VIN i priključak X1 služe za napajanje pločice, a kako bi pločica radila ispravno i bez oštećenja treba imati minimalan napon od 6 V i maksimalni od 20 V, ali preporučeni napon je od 7 V do 12 V. Pločica ima ugrađeni regulator napona koji dovedeni napon na VIN pin pretvara na stabilnih 5 V, pa manji napon od 7 V može rezultirati pretvorbom napona manjom od 5 V zbog padova napona unutar pretvarača, a veći napon od 12 V može jako zagrijati regulator. Na pločici se nalaze 3 GND pina koja služe za uzemljenje odnosno za spajanje minus pola napajanja. Pin +5V ima stabilan napon od 5 V i služi za napajanje drugih komponenti kao što su senzori, a može se i preko njega napajati pločicu, ali treba paziti jer onda treba dovesti stabilnih 5 V jer najveće dopušteno odstupanje je 0.5 V. Pin +3V3 ima stabilan napon od 3.3 V i služi isključivo za napajanje drugih komponenti koje rade na 3.3 V. Pin RESET i tipka X3 služe za resetiranje mikrokontrolera, tj. ponovno pokretanje programa iz početka. Pin IOREF je spojen na 5 V i služi kao referenca za 5 V. Pinovi od A0 do A5 su ulazni analogni pinovi spojeni na 10-bitni analogno-digitalni pretvornik. Pinovi od D0 do D13 su digitalni pinovi i mogu poprimiti vrijednost od 0 V ili 5 V. Digitalni pinovi koji imaju znak ~ su pinovi koji imaju sposobnost PWM (*eng. pulse width modulation*) modulacije [8].

PWM je metoda dobivanja analognih vrijednosti pomoću digitalnih impulsa. PWM se ostvaruje brzim prekidima signala, a srednja vrijednost napona će ovisiti o radnom ciklusu signala [9].



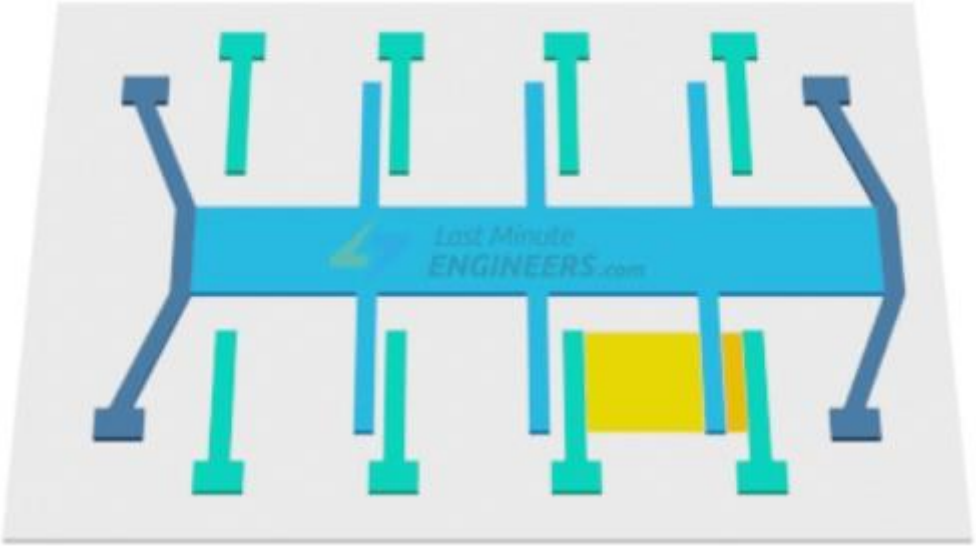
Slika 3.6. PWM modulacija

3.3.MPU6050 senzor

Kretanje MPU6050 senzora se prati pomoću 3-osnog žiroskopa i 3-osnog akcelerometra koji su implementirani u integriranom krugu koji se nalazi na pločici. Na pločici senzora još se nalazi pretvarač koji spušta napon na 3.3 V za napajanje senzora [10].

Slika 3.7. MPU6050 senzor

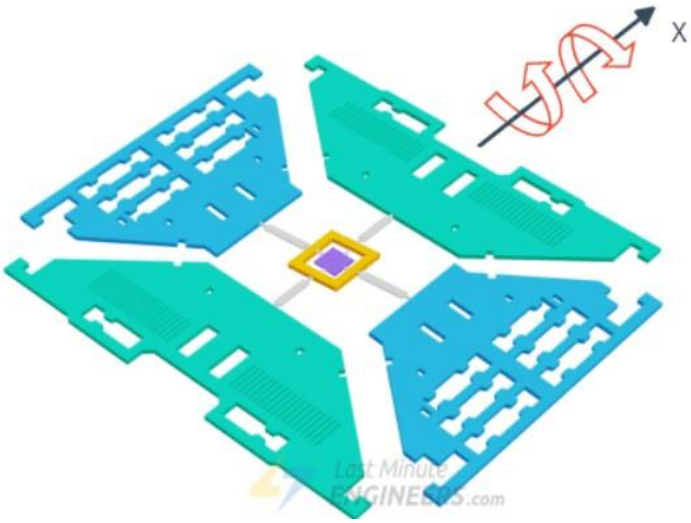
Akcelerometar mjeri ubrzanje senzora na tri osi odnosno ubrzanje u prostoru. Akcelerometar koji se nalazi u ovom senzoru radi na MEMS (*eng. micro electromechanical systems*) principu koji je prikazan na slici 3.8. [11].



Slika 3.8. MEMS akcelerometar [12]

MEMS funkcionira na promjenu kapaciteta, a promjena kapaciteta se mjeri između dvije stacionarne elektrode dok se između njih nalazi pomično tijelo određene mase spojeno na opruge. Ako senzor miruje nema promjene u kapacitetu, a ako se senzor pomakne mijenja se kapacitet C1 i C2 (slika 3.8.) i šalje na digitalni procesor kretanja (DMP) koji obrađuje te podatke i pretvara ih u podatke razumljive Arduinu. Za dobivanje digitalnog izlaza koristi se 16 bitni ADC (analogno-digitalni konverter), a jedinice se mjere u sili gravitacije (g) i MPU6050 senzor ima raspon od +/-2g, +/-4g,+/-8g i +/-16g. Kad je uređaj na ravnoj površini senzor će pokazivati 0g na X i Y osi i +1g na Z osi.

Žiroskop je uređaj koji mjeri rotacijsku brzinu oko osi, najčešće je to oko tri osi odnosno rotacijska brzina u prostoru. Kad se žiroskop okreće unutar senzora dolazi do pojave Coriolisovog efekta koji uzrokuje pomak MEMS sustava unutar senzora. MEMS sustav žiroskopa je kompliciraniji nego kod akcelerometra, ali kao i kod akcelerometra princip rada je isti. Podaci o promjeni kapaciteta se šalju na DMP koji ih obrađuje i šalje dalje. Žiroskop MPU6050 senzora mjeri kutnu brzinu u stupnjevima po jedinici sekunde, a ima raspon od +/-250, +/-500, +/-1000 i +/-2000.



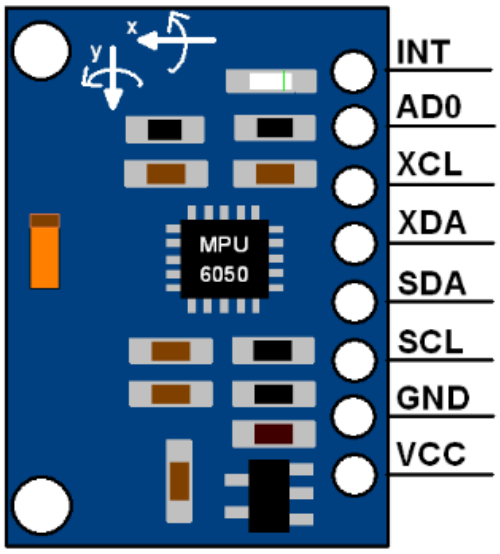
Slika 3.9. MEMS žiroskopa [12]

Komunikacija između MPU6050 senzora i Arduino Uno R3 odvija se pomoću I2C protokola. Protokol I2C koriste mnoge platforme, a Arduino ga koristi za komunikaciju s drugim komponentama koje trebaju prenjet veći broj podataka od obične logičke nule ili jedinice. Za prijenos podataka koristi serijsku sabirnicu koju je 1982. godine razvila tvrtka Philips Semiconductors. Protokol I2C stvoren je za jednostavnu komunikaciju između integriranih krugova u televizoru, ali se 1990. godine počeo šire koristiti.

I2C sabirnica postala je industrijski standard zbog svoje jednostavnosti i niske cijene. Sabirnica koristi samo dvije žice za komunikaciju što je velika prednost nad drugim načinima komuniciranja. Jedna žica se koristi za prijenos serijskih podataka (SDA), a druga za signal radnog takta (SCL). Svi uređaji povezani kroz sabirnicu imaju vlastitu adresu koja je fiksirana hardverom, a ponekad može doći do preklapanja adresa pa je potrebno koristiti još jednu sabirnicu. Sabirnica ima strukturu master-slave i samo uređaj koji je master može započeti komunikaciju kako bi primao i slao podatke na slave uređaj. Jedan master može komunicirati s više slave uređaja, ali uređaji koji su slave ne mogu međusobno komunicirati bez master uređaja. Pošto uređaj koji je master daje radni takt uređaji koji su slave ne trebaju imati svoj zaseban radni takt što je još jedna prednost ovog načina komunikacije.

Nedostataci I2C protokola su spori prijenos jer se podaci prenose serijski jedan po jedan, nije moguće istovremeno slati i primati podatke i ne koriste se metode provjere kako bi se znalo jesu li primljeni bitovi točni [12].

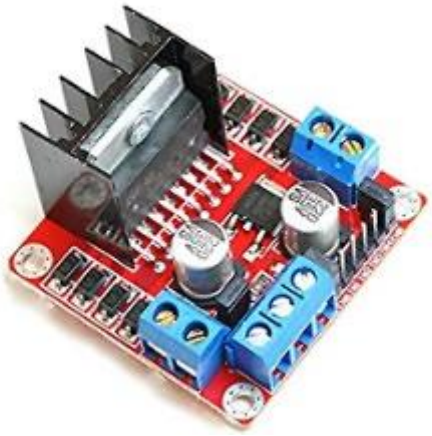
Još jedna bitna stvar kod MPU6050 senzora je raspored pinova i način spajanja. Na slici 3.10. prikazani su izvodi senzora. Pin VCC označava pin napajanja i tu se priključuje napajanje od 5V koje se može dovesti od Arduina, a pin GND označava uzemljenje. Pinovi SDA je pin serijskih podataka, a pin SCL je pin serijskog sata i oni služe za I2C komunikaciju. Pinovi XDA i XCL su pomoćni pinovi za komunikaciju. Pin AD0 je nulti bit u 7-bitnoj adresi senzora i ako se spoji na VCC mijenja adresu senzora. Pin INT je izlazni interupt signal koji se može koristiti prilikom programiranja Arduina [10].



Slika 3.10. Pinovi MPU6050 senzora

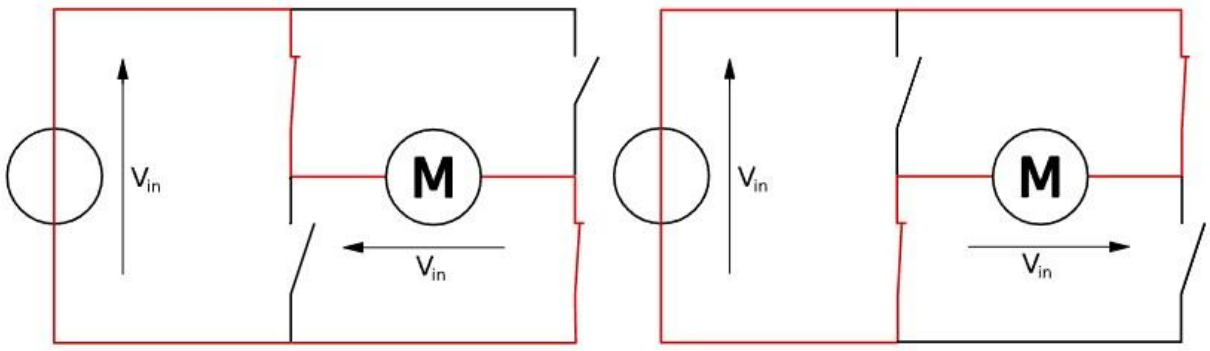
3.4. L298N upravljač

L298N je modul koji služi za upravljanje motorima. Može kontrolirati najviše dva motora i može im kontrolirati smjer okretanja i brzinu vrtnje. L298N služi za upravljanje istosmjernih motora koji zahtijevaju veću struju od one koju Arduino može pružiti. Smjer vrtnje se mijenja pomoću H-mosta, a brzina vrtnje pomoću PWM modulacije koja dolazi preko signala iz Arduino Uno R3 pločice [13].



Slika 3.11. L298N upravljač

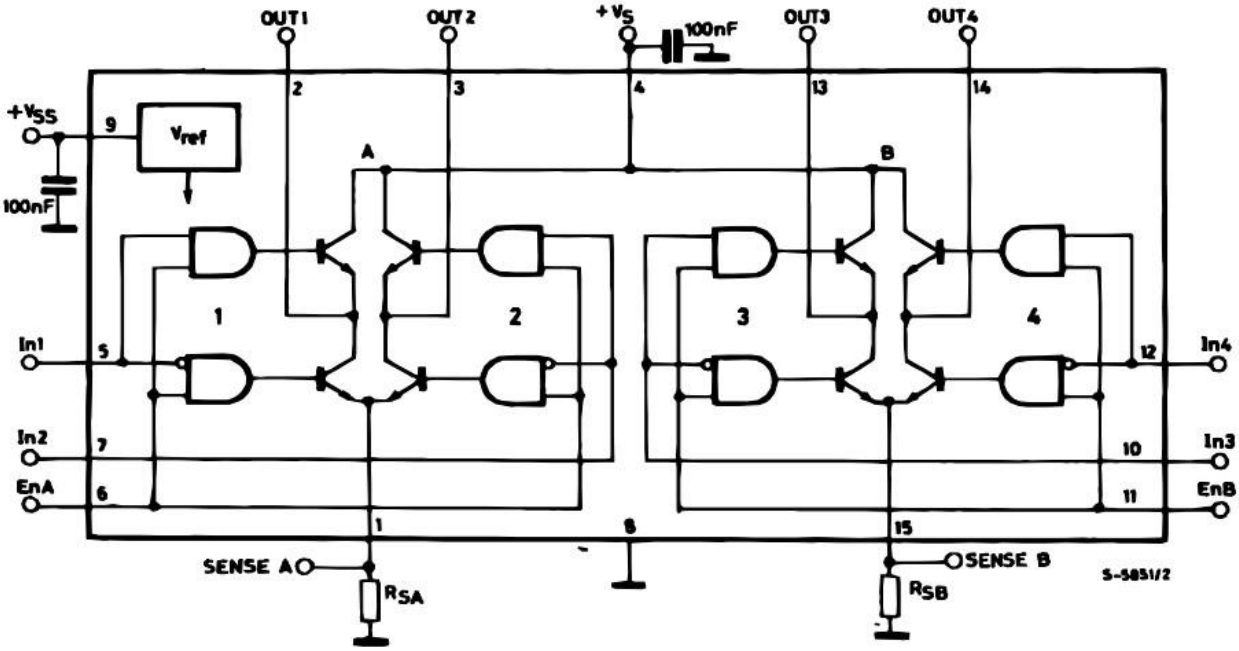
H-most je sklop koji se sastoji od četiri sklopke koje prebacivanjem mogu puštati struju u jednom ili drugom smjeru kroz zavojnicu motora. Sklopke su realizirane pomoću tranzistora ili MOSFET-a koji se nalaze na pločici. Naziv H-most dobio je zato što se na shemama motor koji je u sredini spaja na četiri okolne sklopke [14].



Slika 3.12. H-most [15]

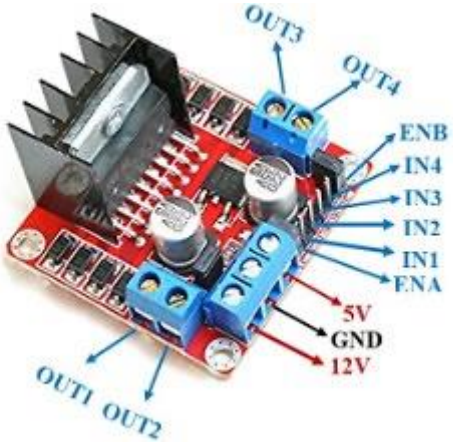
Na slici 3.12. vidi se da kad su zatvorene gornja lijeva sklopka i donja desna sklopka kroz motor teče struja u jednom smjeru, a kad su zatvorene donja lijeva sklopka i gornja desna sklopka struja teče u drugom smjeru.

L298N sastoji se od duplog H-mosta koji je integriran na čipu Multiwatt 15. Dupli H-most radi na principu uključivanja pinova In1 i In2 za prvi motor i pinova In3 i In4 za drugi motor (slika 3.13.). Ti pinovi su digitalni ulazi što znači da imamo dva stanja (visoko i nisko stanje). Kad je na pinu In1 visoko stanje, a na pinu In2 nisko stanje motor će vrtjeti u jednu stranu, a kad se stanja na pinovima zamjene vrtjeti će u drugu stranu. U slučaju da su oba pina u visokom ili niskom stanju motor se neće vrtjeti. Isto to vrijedi za pinove In3 i In4 koji upravljaju smjerom vrtnje drugog motora.



Slika 3.13. Blok dijagram L298N [16]

Brzina vrtnje motora kontrolira se pomoću PWM modulacije na pinove ENA i ENB (slika 3.14.). Ti pinovi također mogu biti kratko spojeni pomoću premosnica (slika 3.15.), u tom slučaju PWM modulacija nije moguća i motori tada rade punom brzinom, samo im je moguće kontrolirati smjer vrtnje. Smjer vrtnje se kontrolira kako je prije objašnjeno pomoću pinova IN1, IN2, IN3 i IN4 (slika 3.14.), a motori se spajaju ja stezaljke OUT1, OUT2, OUT3 i OUT4. Napajanje motora spaja se na stezaljku 12V, a uzemljenje na stezaljku GND. Stezaljka 5V služi za napajanje elektroničkih sklopova na pločici stabiliziranim naponom od 5 V.

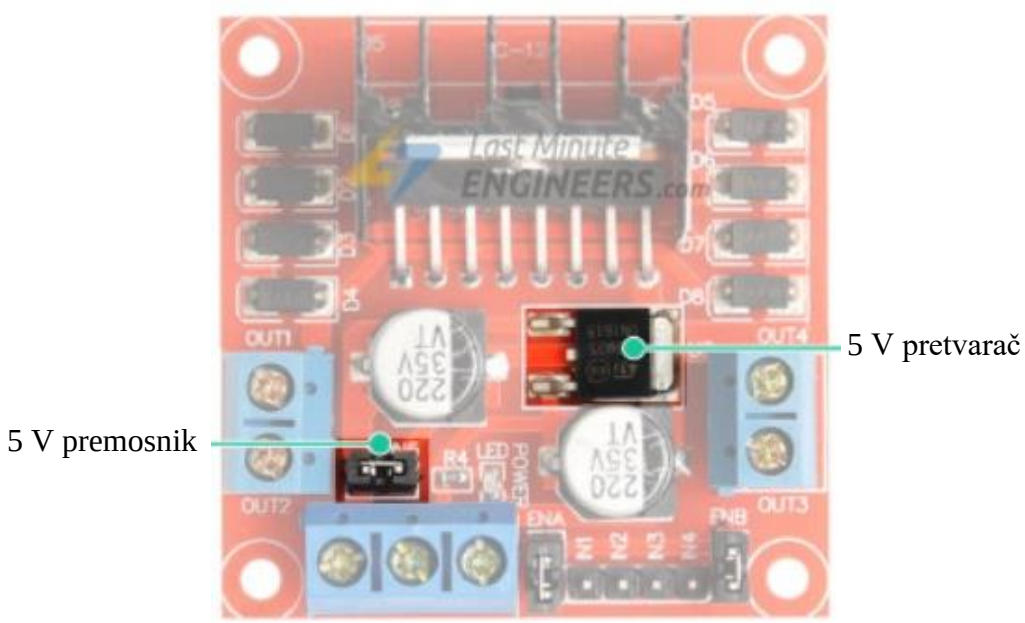


Slika 3.14. Prikaz pinova i stezaljka L298N [13]

Na slici 3.15. prikazani su standardni premosnici koji se koriste za kratko spajanje pinova kod Arduino pločica i drugih elektroničkih pločica. Na slici 3.14. vidi se da su ugrađeni na pinovima ENA i ENB.

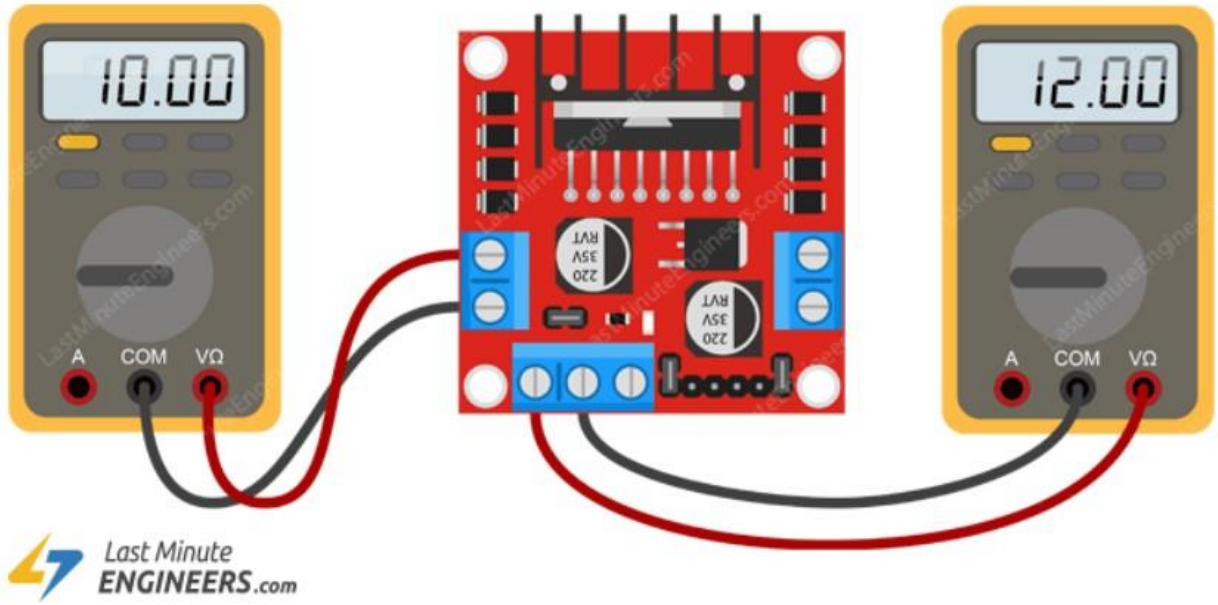
Slika 3.15. Premosnici [17]

Napajanje motora spaja se na stezaljku 12V (slika 3.14.) što znači da se ovaj upravljač napaja s 12 V ili manje, ali to je preporučeno napajanje. Ovaj upravljač može se napajati naponom od 35 V, ali u tom slučaju 5 V regulator koji se nalazi na pločici (slika 3.16.) i služi za napajanje logičkih krugova pločice bi mogao pregoriti. Zato je potrebno skinuti premosnik za 5 V kao što se vidi na slici 3.16. i u tom slučaju treba dovesti stabilan napon od 5 V na stezaljku 5V, a ako imamo napon od 12 V ili manje premosnik možemo ostaviti i u tom slučaju možemo iz stezaljke 5V napajati neke druge komonente stabilnim naponom 5 V [18].



Slika 3.16. Prikaz 5V premosnika i pretvarača

Važno je isto tako znati da postoji pad napona od 2 V koji se izgubi na komponentama unutar integriranog kruga (slika 3.17.). To moramo uzeti u obzir jer ako imamo napajanje od 12 V i motor također radi na 12 V nikad nećemo moći vrtjeti motor punom brzinom. Zato treba izabrati izvor napajanja koji ima 2 V više od potrebnog.

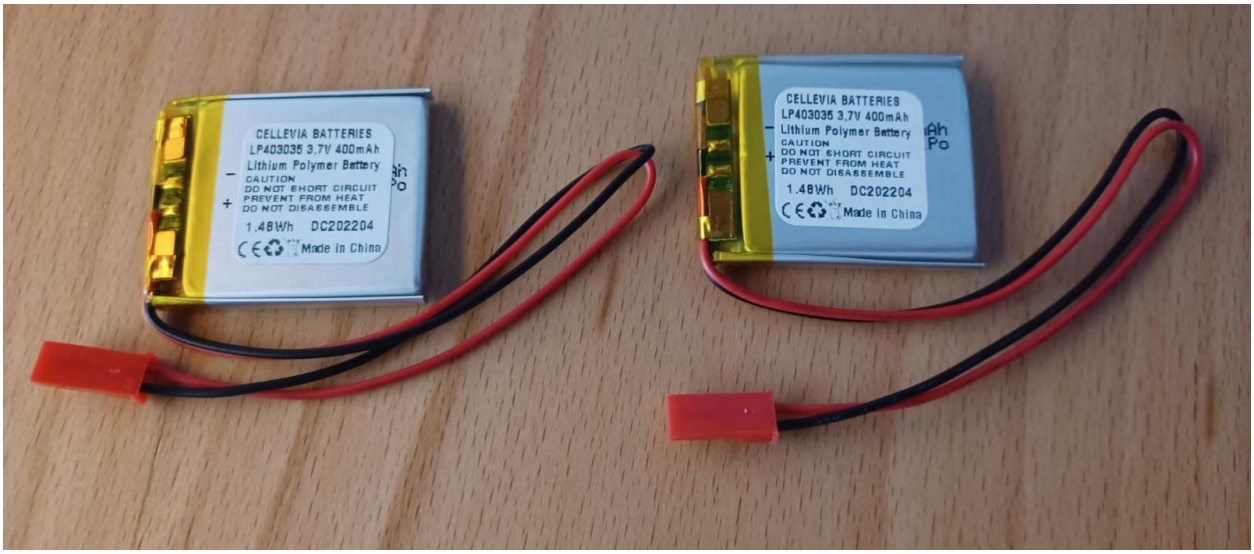


Slika 3.17. Pad napona na L298N

Osim napona bitna nam je i struja koju L298N može dati, a to je maksimalno 2 A po motoru što je i više nego dovoljno jer znamo da naši motori troše 250 mA pri maksimalnom naponu od 12 V.

3.5. Izvor napajanja

Za izvor napajanja koristile su se dvije litijeve baterije spojene u seriju. Razlog spajanja baterija u seriju je zato što svaka baterija ima napon od 3.7 V i kad se to zbroji dobijemo ukupan napon od 7.4 V. Motori se mogu napajati sa 12 V, ali je preporučeno 6 V, pa će se napajati s 6 V odnosno nešto manje jer upravljač L298N ima pad napona od 2 V što znači da će motor od mogućih 7.4 V dobiti 5.4 V i to je dovoljan napon jer je ovaj robot lagan, a motori imaju redukciju što čini situaciju još povoljnijom. Maksimalna struja motora je 250 mA pri 12 V, a pošto se motor napaja s 5.4 V struja je dosta manja. Kapacitet jedne baterije je 400 mAh, a pošto se koriste dvije baterije kapaciteti se zbrajaju i ukupan kapacitet iznosi 800 mAh što je sasvim dovoljno za ove motore [19].



Slika 3.18. Baterije korištene u izradi

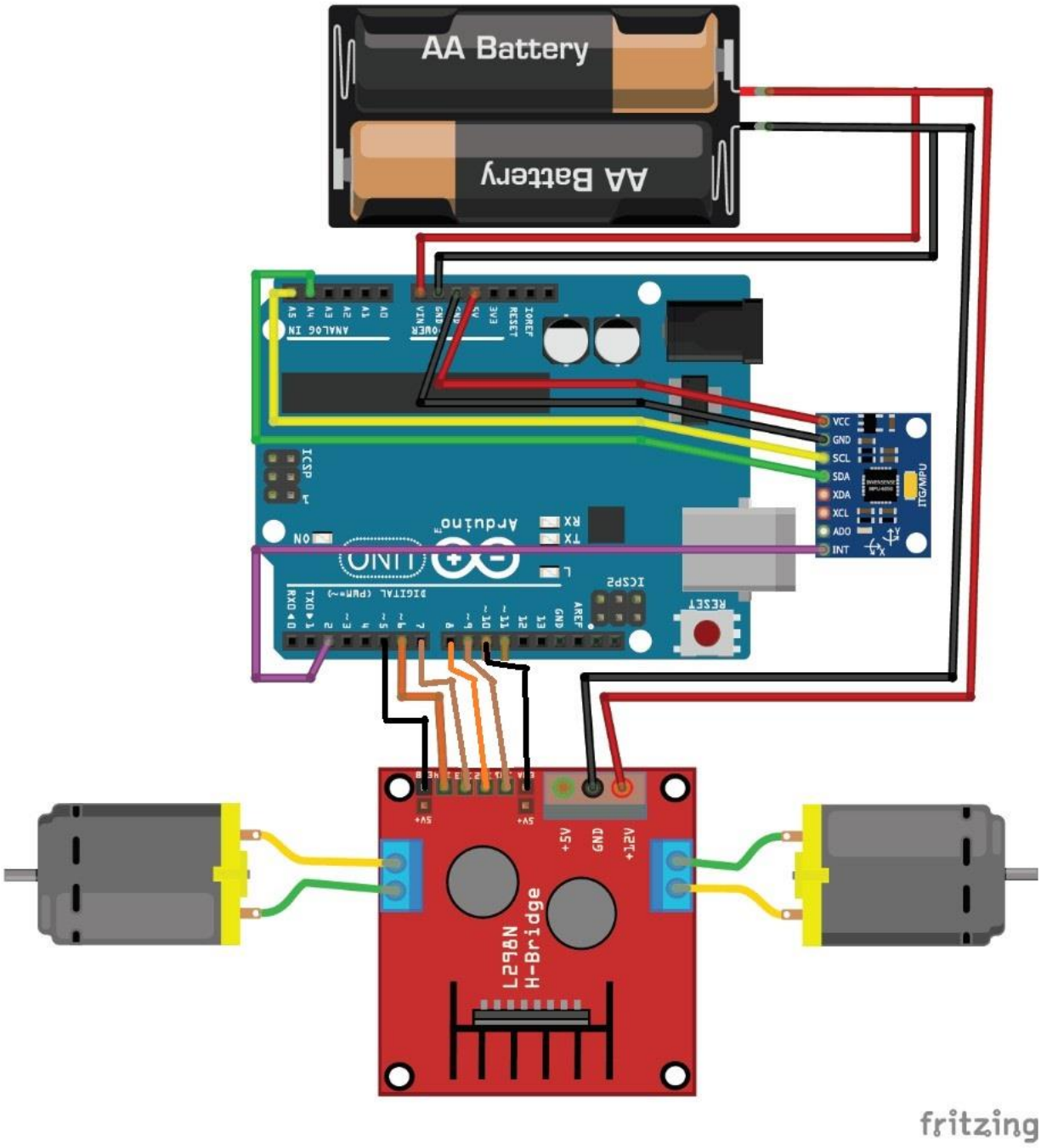
3.6. Ostale komponente

Pri izradi ovog robota još su se koristile i ostale komponente kao što su žice, kotači, prekidač i drvene ploče za tijelo robota. Što se žica tiči nema nikakvih zahtjeva jer je najveća struja oko 100 mA pa su dovoljne tanke žice malog promjera. Isto tako zbog male struje nije bilo posebnih

zahtjeva za prekidač. Kod kotača je bitno da ne kližu odnosno da imaju dobro trenje s površinom kako bi se robot mogao ispravno balansirati. Tijelo robota treba biti što lakše, ali kruto i zato se koriste što tanje drvene ploče.

4. SPAJANJE SAMO-BALANSIRAJUĆEG ROBOTA

Shema spajanja elektroničkih komponenti prikazana je na slici 4.1. Baterije napajaju Arduino Uno R3 pločicu preko pinova VIN i GND. Na pin VIN je spojen pozitivan pol baterije napona 7.4 V, a na pin GND negativan pol. Na pin VIN preporučeno napajanje je između 7 i 12 V što pruža ova baterija.

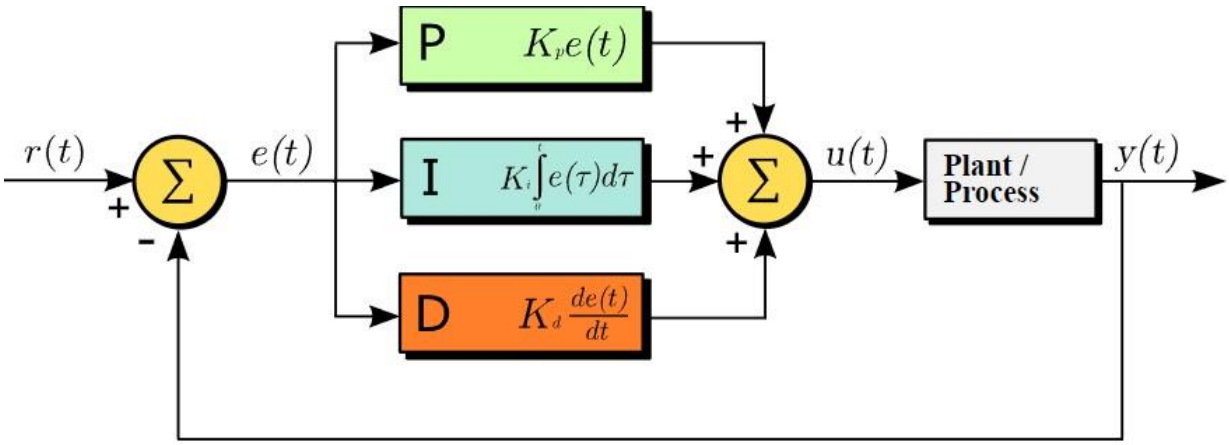


Slika 4.1. Shema spajanja [20]

Baterija također napaja upravljač L298N na stezaljke +12V i GND. Pošto je ulazni napon na upravljaču manji od 12 V nije potrebno posebno napajanje od 5 V za rad elektronike nego se stavlja premosnik i upravljač preko svog pretvarača dovedenih 7.4 V pretvara na 5 V. Senzor MPU6050 se napaja tako da uzimamo 5 V od pinova 5V i GND s Arduino pločice i spajamo ih na pinove VCC i GND na senzoru. Naravno VCC se spaja na pin 5V, a GND na pin GND. Za I2C komunikaciju i prijenos informacija sa senzora na Arduino pin SCL spojen je na analogni ulaz A5, a pin SDA spojen je na analogni pin A4 i pin INT spojen je na digitalni ulaz D2. Pinovi H mosta spojeni su na digitalne ulaze od D5 do D10 i to tako da su pinovi ENA i ENB spojeni na pinove D5 i D10 jer ti pinovi imaju sposobnost PWM modulacije, a pinovi IN1, IN2, IN3 i IN4 su spojeni na pinove D6, D7, D8 i D9. Prvi motor se spaja na stezaljke upravljača OUT1 i OUT2, a drugi na stezaljke upravljača OUT3 i OUT4 (slika 3.14.).

5. PID REGULACIJA

PID regulator ili proporcionalno integralno derivacijski regulator je regulator koji koristi povratnu vezu za prepoznavanje greške i proporcionalne, integralne i derivacijske članove za ispravljanje pogreške. Vrijednost pogreške se stalno iznova izračunava kao razlika između zadane vrijednosti i izmjerene vrijednosti na izlazu koja se vraća pomoću negativne povratne veze. PID regulator je vrlo osjetljiv i točan jer može regulirati vrijednosti pomoću tri parametra i zato je široko korišten u automatizaciji, robotici, elektronici i mnogim drugim granama. Blok dijagram na slici 5.1. prikazuje djelovanje svakog člana na zadanu vrijednost [21].

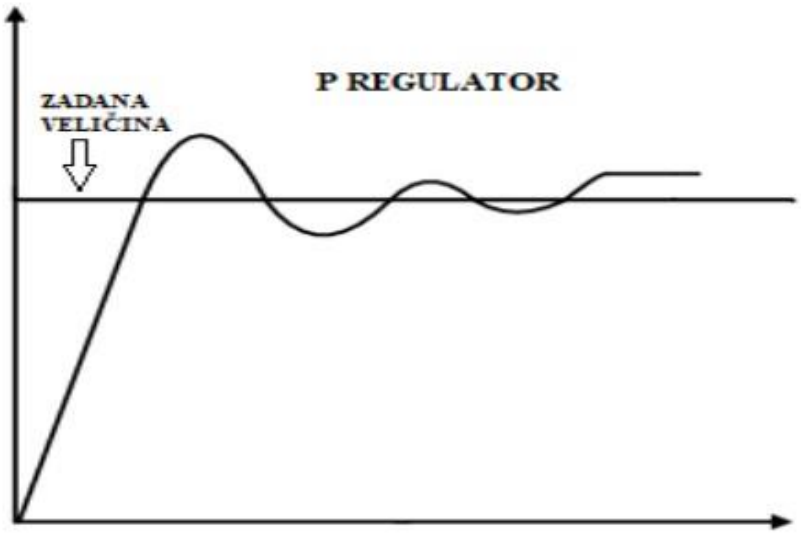


Slika 5.1. Blok dijagram PID regulatora

P član ili proporcionalni član je član koji vrijednost proporcionalno uvećava za faktor Kp tako da ga pomnoži s Kp faktorom. Znači, što je veće odstupanje regulirani signal će biti proporcionalno veći, ali samim korištenjem P regulatora doći će do male konstante pogreške između zadane vrijednosti i vrijednosti na izlazu jer je P regulatoru potrebna pogreška za dobivanje proporcionalnog izlaznog odziva.

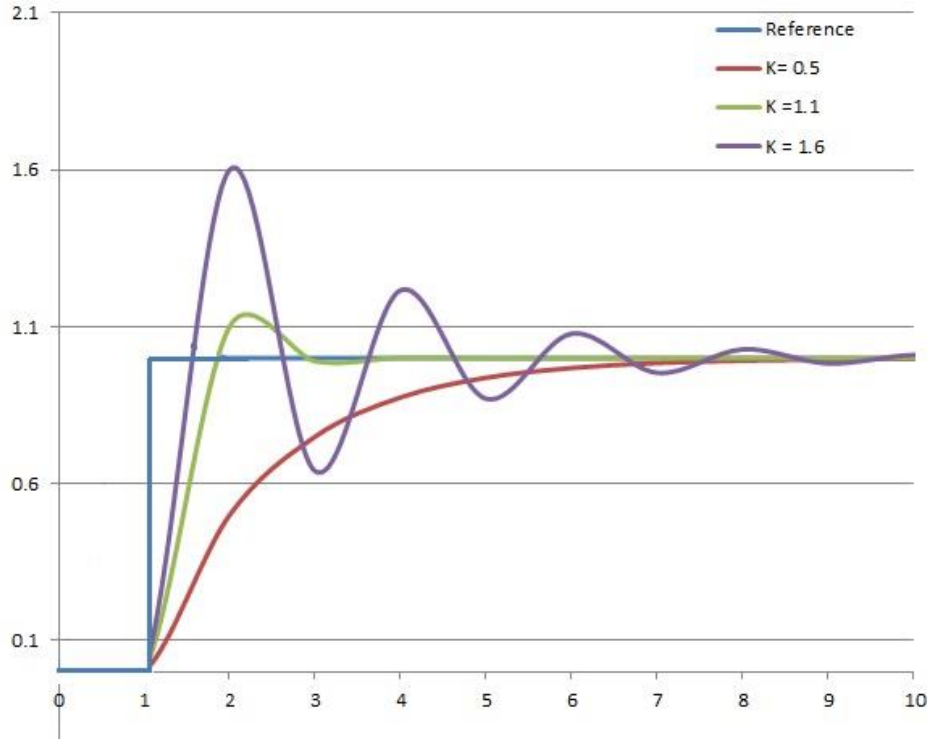
I član ili integracijski član je član koji gleda prošle vrijednosti greške i integrira ih. Integrirani član nastoji ukloniti zaostalu pogrešku, a kad se pogreška ukloni prestati će integrirati. Zato se često koristi u kombinaciji s P članom jer otklanja statičku pogrešku P člana.

D član ili derivacijski član je član koji radi na temelju trenutne promjene. Što je promjena regulirane vrijednosti brža to će brže D član reagirati i prigušiti pogrešku.



Slika 5.2. Prikaz odziva P regulatora [22]

PID regulator kombinira djelovanja svih triju članova i uglađuje njihov rad preko njihovih konstanti Kp, Ki i Kd. Za svaki sustav ove su konstante drugačije i zahtijevaju podešavanje ukoliko želimo imati stabilan sustav. Jedna od dobrih karakteristika ovog regulatora je sve veće povećanje korekcije sa sve većim povećanjem greške. Kvalitetu regulatora gleda se kroz njegovu reakciju na pogrešku. Gleda se brzina odziva, točnost regulacije, stupanj oscilacije sustava, stupanj prelaska zadane vrijednosti i drugi parametri.



Slika 5.3. PID regulacija

Postoje mnogo metoda određivanja regulacijskih konstanti, a u ovom radu su se koristile dvije metode. To su ručna metoda i Ziegler-Nicholsova metoda. Ručno prilagođavanje konstanti je dosta grubo i može uzeti dosta vremena, ali se svejedno koristilo u ovome radu za krajnju finu regulaciju. Ziegler-Nicholsova metoda se izvodi tako da se konstante Ki i Kd postave na 0, a sustav se dovede do granice osciliranja pojačavanjem proporcionalne konstante Kp. Tada se Kp očita i zabilježi kao Ku i očita se period oscilacije Tu. Kad imamo konstante Ku i Tu prema tablici 5.1. izračunaju se vrijednosti konstanti regulatora. Samo-balansirajući robot ne možemo dovesti do same granice oscilacije jer bi se uništio, pa nakon što se približno odrede parametri pomoću Ziegler-Nicholsove metode ručno se prilagođavaju konstante za bolji odziv i rad robota [21].

Tablica 5.1. Ziegler-Nicholsova metoda

Vrsta kontrole	Kp	Ki	Kd
P	0.5 Ku	-	-
PI	0.45 Ku	0.54 Ku/Tu	-
PID	0.6 Ku	1.2 Ku/Tu	3 Ku*Tu/40

6. PROGRAMSKI KOD

Za izradu programskog koda koristi se razvojno sučelje Arduino IDE. To je programski softver tvrtke Arduino i temelji se na programskom jeziku C. Na početku koda uključuju se biblioteke koje se pozivaju pomoću naredbe `#include` i služe za razne namjene. Biblioteke koje se koriste u ovom programu su `PID_v1.h` za izračunavanje PID konstanti i za PID regulaciju, `LMotorController.h` za kontrolu L298N upravljača, `I2Cdev.h` za I2C komunikaciju između senzora i Arduino pločice i `MPU6050_6Axis_MotionApps20.h` za očitanje informacija sa senzora.

```
#include <PID_v1.h>
#include <LMotorController.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
```

U ovom dijelu koda poziva se `Wire.h` biblioteka za I2C komunikaciju kako bi ispravno radila.

```
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif
```

Pomoću naredbe `#define` definiramo neku konstantnu varijablu, a naredbom `MPU6050 mpu` se adresi MPU6050 senzora dodjeljuje ime varijable mpu zbog lakšeg snalaženja u kodu.

```
#define MIN_ABS_SPEED 30
MPU6050 mpu;
```

Ove linije koda provjeravaju i postavljaju se za rad između MPU6050 senzora i Arduino pločice.

```
bool dmpReady = false;
uint8_t mpuIntStatus;
uint8_t devStatus;
uint16_t packetSize;
uint16_t fifoCount;
uint8_t fifoBuffer[64];
```

Sljedeće linije uvode kvaterniorni spremnik koji u sebi sadrži 4 varijable i definira se struktura podataka kako bi se mogla bilježiti pozicija senzora.

```
Quaternion q;
VectorFloat gravity;
float ypr[3];
```

Sada se definiraju ulazne vrijednosti PID regulatora i zadaju se vrijednosti koje PID regulator mora održavati. Isto tako definiraju se vrijednosti konstanti PID regulatora i postavlja se PID regulator za rad.

```
double originalSetpoint = 179;
double setpoint = originalSetpoint;
double movingAngleOffset = 0.1;
double input, output;
int moveState = 0;
double Kp = 94;
double Kd = 1.9;
double Ki = 73;
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);
```

Zatim se postavlja L298N upravljač za rad. Definiraju se digitalni izlazi na Arduino pločici koji su spojeni s upravljačem i pomoću varijabli `double motorSpeedFactorLeft` i `double motorSpeedFactorRight` reguliramo brzinu vrtnje motora na način da vrijednost 1 predstavlja maksimalnu brzinu ili 100% brzine, a ako motori nisu jednakih brzina možemo ih uskladiti.

```
double motorSpeedFactorLeft = 0.5;
double motorSpeedFactorRight = 0.5;

int ENA = 5;
int IN1 = 6;
int IN2 = 7;
int IN3 = 8;
int IN4 = 9;
int ENB = 10;
LMotorController motorController(ENA, IN1, IN2, ENB, IN3, IN4,
motorSpeedFactorLeft, motorSpeedFactorRight);

long time1Hz = 0;
long time5Hz = 0;
```

Sljedećim linijama koda se provjerava stanje INT pina i provjeravaju se promjene na pinu. Zatim se u `void setup` počinje uspostavljati I2C veza. Naredba `void setup` izvodi se samo pri pokretanju koda i zato se tu na početku uspostavlja veza. Nakon što je veza uspostavljena pokreće se MPU6050 senzor.

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

```
void setup()
{

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
  TWBR = 24;
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif

  Serial.begin(115200);
  while (!Serial);

  Serial.println(F("Initializing I2C devices..."));
  mpu.initialize();

  Serial.println(F("Testing device connections..."));
  Serial.println(mpu.testConnection() ? F("MPU6050 connection
successful") : F("MPU6050 connection failed"));

  Serial.println(F("Initializing DMP..."));
  devStatus = mpu.dmpInitialize();
```

Niti jedan MPU6050 senzor nije isti pa se zato postavljaju različiti parametri kako bi potištili te greške i kako bi senzor točno očitavao. Pošto je senzor pokrenut sad se provjerava rad DMP-a i ako sve radi postavljaju se PID postavke za daljnji rad regulatora, postavlja se automatsko pokretanje PID regulatora i zadaju se granice za kontrolu brzine motora.

```
  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(-85);
  mpu.setZAccelOffset(1788);

  if (devStatus == 0)
  {

    Serial.println(F("Enabling DMP..."));
    mpu.setDMPEnabled(true);

    Serial.println(F("Enabling interrupt detection (Arduino
external interrupt 0)..."));
    attachInterrupt(0, dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();

    Serial.println(F("DMP ready! Waiting for first interrupt..."));
    dmpReady = true;

    packetSize = mpu.dmpGetFIFOPacketSize();

    pid.SetMode(AUTOMATIC);
    pid.SetSampleTime(10);
    pid.SetOutputLimits(-255, 255);
```

```
  }
  else
  {

    Serial.print(F("DMP Initialization failed (code "));
    Serial.print(devStatus);
    Serial.println(F(")"));
  }
}
```

Nakon što su se uspostavile i definirale sve veze između senzora i Arduina i postavili parametri PID regulatora može se početi izvršavati glavna petlja programa. U glavnoj petlji provjerava se stanje senzora i gleda se stanje na varijablama prostora koje su definirane na početku. PID regulator izračunava vrijednosti pogreške i zadaje naredbe motorima kako bi uklonio pogrešku. Ostale linije koda služe za provjeru varijabli prostora i ispravnosti koda.

```
void loop()
{

  if (!dmpReady) return;

  while (!mpuInterrupt && fifoCount < packetSize)
  {
    pid.Compute();
    motorController.move(output, MIN_ABS_SPEED);
  }

  mpuInterrupt = false;
  mpuIntStatus = mpu.getIntStatus();

  fifoCount = mpu.getFIFOCount();

  if ((mpuIntStatus & 0x10) || fifoCount == 1024)
  {

    mpu.resetFIFO();
    Serial.println(F("FIFO overflow!"));
  }
  else if (mpuIntStatus & 0x02)
  {

    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

    mpu.getFIFOBytes(fifoBuffer, packetSize);

    fifoCount -= packetSize;

    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
#if LOG_INPUT
    Serial.print("ypr\t");
```

```
    Serial.print(ypr[0] * 180 / M_PI);
    Serial.print("\t");
    Serial.print(ypr[1] * 180 / M_PI);
    Serial.print("\t");
    Serial.println(ypr[2] * 180 / M_PI);
#endif
    input = ypr[1] * 180 / M_PI + 180;
  }
}
```

7. ZAKLJUČAK

Problematika izrade samo-balansirajućeg robota bila je odabir komponenti. Kod odabira komponenti trebalo je paziti da su sve komponente kompatibilne jer imamo puno zahtjeva. Motori trebaju biti dovoljno jaki za okretanje tijela robota, upravljač motora mora moći podržati zahtjeve tih motora. Senzor pokreta mora biti dovoljno osjetljiv jer bi inače robot previše oscilirao i treba imati upravljaču pločicu koja je laka za programirati. Izabrane komponente u ovome radu su dobro radile zajedno u sklopu i robot radi relativno dobro. Za neke bolje rezultate trebao bi se napraviti neki stabilniji i jači robot koji je kontroliran bržom upravljačkom pločicom i osjetljivijim senzorom.

Samo-balansirajući robot nema neke primjene u praksi, ali zato pojedini dijelovi kao što su senzor ili PID regulator imaju vrlo široku upotrebu. Vrijednosti PID regulatora nisu se određivale idealno zbog konstrukcije robota, ali sa dodatnim podešavanjem robot nije imao jakih oscilacija. Iz ovog rada vidio sam primjenu PID regulacije koja ima dosta prednosti i shvatio njezinu rasprostranjenu upotrebu u svim granama industrije.

Jedan od glavnih ciljeva rada bio je upoznat se bolje sa Arduino Uno platformom i njezinim širokim primjenama kao i bolje upoznavanje s PID regulacijom.

8. LITERATURA

[1] Novaković, B.: „Robotika“, s interneta, https://tehnika.lzmk.hr/robotika/, 27. kolovoza 2023.

[2] Hrvatska enciklopedija: „Automatizacija“, s interneta, https://enciklopedija.hr/natuknica.aspx?ID=4745, 27. kolovoza 2023.

[3] Hackster: „Self-balancing bobot“, s interneta, https://www.hackster.io/marketingmanagerofdattabanur/arduino-self-balancing-robot-e23f9c, 27. kolovoza 2023.

[4] Automatika: „DC elektromotori“, s interneta, https://www.automatika.rs/baza-znanja/mehatronika/dc-elektromotori.html, 28. kolovoza 2023.

[5] Chipoteka: „Motor 3-6 Vdc, sa reduktorom 120:1, 130rpm, OKYSTAR“, s interneta, https://www.chipoteka.hr/motor-3-6-vdc-sa-reduktorom-1201-130rpm-okystar-8090007602, 28. kolovoza 2023.

[6] „Arduino Uno“, s interneta, https://en.wikipedia.org/wiki/Arduino_Uno, 28. kolovoza 2023.

[7] „ATmega328p“, s interneta, https://en.wikipedia.org/wiki/ATmega328, 28. kolovoza 2023.

[8] Arduino: „Arduino Uno R3“, s interneta, https://docs.arduino.cc/static/363e074b8227fe2b7b8bee772b5a3c34/A000066-datasheet.pdf, 28. kolovoza 2023.

[9] Hirzel, T.: „Basics of PWM (Pulse Width Modulation)“, s interneta, https://docs.arduino.cc/learn/microcontrollers/analog-output, 28. kolovoza 2023.

[10] ElectronicWings: „MPU6050“, s interneta, https://www.electronicwings.com/sensors-modules/mpu6050-gyroscope-accelerometer-temperature-sensor-module, 6. rujna 2023.

[11] HowToMehatronics: „MEMS“, s interneta, https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/#google_vignette, 6. rujna 2023.

[12] LastMinuteEngineers: „MPU6050“, s interneta, https://lastminuteengineers.com/mpu6050-accel-gyro-arduino-tutorial/#google_vignette, 6. rujna 2023.

[13] „I2C“, s interneta, https://www.hwlibre.com/hr/i2c-arduino/, 6. rujna 2023.

[14] „L298N Motor Driver Module“, s interneta, https://components101.com/modules/l293n-motor-driver-module, 6. rujna 2023.

[15] HowToMehatronics: „L298N Motor Driver“, s interneta, https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/#google_vignette, 6. rujna 2023.

[16] „H most“, s interneta, https://sr.wikipedia.org/sr-el/%D0%A5_%D0%BC%D0%BE%D1%81%D1%82, 6. rujna 2023.

[17] „L298N datasheets“, s interneta, https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf, 6.rujna 2023.

[18] „Mini Jumper“, s interneta, https://www.ciceksepeti.com/2.54mm-standart-pcb-mini-jumpersont-kcm10400021, 6. rujna 2023.

[19] LastMinuteEngineers: „Interface L298N DC Motor Driver Module with Arduino“, s interneta, https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/, 6.rujna 2023.

[20] „Litijeve baterije“, s interneta, https://www.chipoteka.hr/elektronika-i-alati/elektronika/litijeve-baterije/baterija-litijeva-37v-li-po-400-mah-l403035-2400200187, 6. rujna 2023.

[21] Aswinth, R.: „Self-balancing robot“, s interneta, https://circuitdigest.com/microcontroller-projects/arduino-based-self-balancing-robot, 7. rujna 2023.

[22] „PID regulator“, s interneta, https://en.wikipedia.org/wiki/Proportional%E2%80%93integral%E2%80%93derivative_controller, 7. rujna 2023.

[23] Brljafa, M. (2020). *Sustav automatskog upravljanja temperature* (Završni rad). Pula: Istarsko veleučilište - Universita Istriana di scienze applicate. Preuzeto s https://urn.nsk.hr/urn:nbn:hr:212:921201, 7. rujna 2023.

[24] Arduino: „Arduino IDE“, s interneta, https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics, 8. rujna 2023.

9. POPIS OZNAKA I KRATICA

PID	Proportional-Integral-Derivative
DC	Direct current
V	Volt
A	Amper
EEPROM	Electrically erasable programmable read-only memory
SRAM	Static random-access memory
MIPS	Million instructions per second
MHZ	Mega hertz
PWM	Pulse width modulation
MEMS	Micro electromechanical systems
DPM	Digital motion processor
ADC	Analog-to-digital conversion
I2C	Inter-integrated circuit
SDA	Serial data address
SCA	Serial clock address
MOSFET	Metal oxide semiconductor field effect transistor
IDE	Integrated development environment

10. SAŽETAK I KLJUČNE RIJEČI

Cilj ovoga završnog rada je prikazati proces izrade samo-balansirajućeg robota i korištenja PID regulacije u balansiranju robota. Ovaj robot stoji na dva kotača koja se reguliraju pomoću dva DC motora koja održavaju ravnotežno stanje. Motori se kontroliraju preko upravljača kojeg kontrolira Arduino Uno, a očitanje pomaka u prostoru mjeri se akcelerometrom. Sve korištene komponente objašnjene su u radu, kao i programski kod kojemu je zadatak povezati senzor i Arduino Uno, pa zatim kontrolirati motore. PID regulacija korištena je za balansiranje ovoga robota i prikazana je njezina implementacija u programskom kodu.

Ključne riječi: samo-balansirajući robot, Arduino, DC motori, senzor, H most, PID regulacija

SUMMARY AND KEY WORDS

The goal of this bachelor's thesis is to present the process of designing a self-balancing robot and implementing PID regulation for robot balancing. This robot has two wheels that are regulated by two DC motors which maintain balance. The motors are controlled with a driver module controlled by Arduino Uno, while the displacement is measured by an accelerometer. All used components are described in the paper, as well as the program code which was written to connect the sensor and Arduino Uno and then to control the motors. PID regulation is used for balancing the robot and its implementation in the program code is shown.

Key words: self-balancing robot, Arduino, DC motors, sensor, H bridge, PID regulation

11. DODATAK

Programski kod

```
#include <PID_v1.h>
#include <LMotorController.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

#define MIN_ABS_SPEED 30

MPU6050 mpu;

bool dmpReady = false;
uint8_t mpuIntStatus;
uint8_t devStatus;
uint16_t packetSize;
uint16_t fifoCount;
uint8_t fifoBuffer[64];

Quaternion q;
VectorFloat gravity;
float ypr[3];


double originalSetpoint = 179;
double setpoint = originalSetpoint;
double movingAngleOffset = 0.1;
double input, output;
int moveState = 0;
double Kp = 94;
double Kd = 1.9;
double Ki = 73;
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);

double motorSpeedFactorLeft = 0.5;
double motorSpeedFactorRight = 0.5;

int ENA = 5;
int IN1 = 6;
int IN2 = 7;
int IN3 = 8;
int IN4 = 9;
int ENB = 10;
LMotorController motorController(ENA, IN1, IN2, ENB, IN3, IN4, motorSpeedFactorLeft,
motorSpeedFactorRight);


long time1Hz = 0;
long time5Hz = 0;

volatile bool mpuInterrupt = false;
void dmpDataReady()
{
```

```
  mpuInterrupt = true;
}


void setup()
{

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
  TWBR = 24;
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif


  Serial.begin(115200);
  while (!Serial);


  Serial.println(F("Initializing I2C devices..."));
  mpu.initialize();


  Serial.println(F("Testing device connections..."));
  Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") :
F("MPU6050 connection failed"));


  Serial.println(F("Initializing DMP..."));
  devStatus = mpu.dmpInitialize();


  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(-85);
  mpu.setZAccelOffset(1788);

  if (devStatus == 0)
  {

    Serial.println(F("Enabling DMP..."));
    mpu.setDMPEnabled(true);

    Serial.println(F("Enabling interrupt detection (Arduino external interrupt
0)..."));
    attachInterrupt(0, dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();


    Serial.println(F("DMP ready! Waiting for first interrupt..."));
    dmpReady = true;


    packetSize = mpu.dmpGetFIFOPacketSize();


    pid.SetMode(AUTOMATIC);
    pid.SetSampleTime(10);
    pid.SetOutputLimits(-255, 255);
  }
  else
  {

    Serial.print(F("DMP Initialization failed (code "));
    Serial.print(devStatus);
```

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


void loop()
{

  if (!dmpReady) return;


  while (!mpuInterrupt && fifoCount < packetSize)
  {


    pid.Compute();
    motorController.move(output, MIN_ABS_SPEED);

  }


  mpuInterrupt = false;
  mpuIntStatus = mpu.getIntStatus();


  fifoCount = mpu.getFIFOCount();


  if ((mpuIntStatus & 0x10) || fifoCount == 1024)
  {

    mpu.resetFIFO();
    Serial.println(F("FIFO overflow!"));


  }
  else if (mpuIntStatus & 0x02)
  {

    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();


    mpu.getFIFOBytes(fifoBuffer, packetSize);

    fifoCount -= packetSize;

    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
#if LOG_INPUT
    Serial.print("ypr\t");
    Serial.print(ypr[0] * 180 / M_PI);
    Serial.print("\t");
    Serial.print(ypr[1] * 180 / M_PI);
    Serial.print("\t");
    Serial.println(ypr[2] * 180 / M_PI);
#endif
    input = ypr[1] * 180 / M_PI + 180;
  }
}
```