Univerzitet "Džemal Bijedić"
Fakultet Informacijskih Tehnologija

Završni rad

Samobalansirajući robot

elektronički bazirani projekat za održavanje robota na dva točka uspravnim

Profesor:
Doc.dr. Elmir Babović

Student:
Adi Šoše, IB160034

Akademska godina: 2018/2019

Mostar, 2019. godina

IZJAVA O AUTORSTVU

Ja, **ADI (AMIR) ŠOŠE**, student Fakulteta informacijskih tehnologija, Univerziteta "Džemal Bijedić" u Mostaru, pod punom moralnom, materijalnom i krivičnom odgovornošću,

Izjavljujem

da je rad pod naslovom

SAMOBALANSIRAJUĆI ROBOT

u potpunosti rezultat sopstvenog istraživanja, gdje su korišteni sadržaji (tekst, ilustracije, tabele itd.) drugih autora jasno označeni pozivanjem na izvor i ne narušavaju bilo čija vlasnička ili autorska prava.

U Mostaru, 17.09.2019.

Adi Šoše, IB160034

Sadržaj

1 Sažetak

2 Uvod

U nadolazećem tekstu imat ćete priliku da se detaljnije upoznate sa teoretskim i praktičnim procesom izrade samobalansirajućeg robota. U prvih nekolika sekcija, čitaoc će se upoznati sa tehnologijama i uređajima korištenim u svrhe izrade diplomskog rada. Također, osvrnut ćemo se na osnove elektronike koje je neizbježno razumijeti da bi se shvatio praktični dio rada. Još jedna stavka o kojoj ću pisati jesu matematičke kontrolne petlje koje su jedan od glavnih aspekata samobalansirajućeg robota.

Osnovni cilj ovog projekta jeste uspješna izrada robota koji se može uspravno balansirati na dva točka i kretati se pomoću komandi koje će primiti putem bluetooth veze.

Nakon teoretskog dijela rada detaljnije ću govoriti o samoj Android aplikaciji kao i najzanimljivijem dijelu projekta - izradi robota koji se kontroliše pomoću Arduino mikrokontrolera. Bitno je naglasiti da se radi o multidisciplinarnom radu koji integriše polja matematike, fizike, elektronike, komokunikacijskih protokola, mehanike, inžinjeringa i razvoja mobilnih aplikacija.

3 Arduino

Arduino je elektronički bazirana platforma otvorenog koda[1]. Radi se o ploči koja na sebi najčešće ima Atmel-ov 8-bitni AVR mikrokontroler. Iako postoji više vrsta Arduino ploča (Uno, Nano, Mega, Leonardo itd.), koncept programiranja njihovog ponašanja je isti. Međutim, ono što čini razlike među ovim verzijama, jeste količina radne memorije, kao i broj ulazno/izlaznih pinova. Ono što čini ovu platformu veoma popularnom jeste njena pristupačnost, kako cijenom, tako i stepenom potrebnog predznanja iz polja elektronike i integralnih kola.

Arduino, kao platforma, nije namijenjen za rješenja u produkciji i masovnu proizvodnju, već za izradu prototipa uređaja ili projekte koji spadaju u kategoriju hobija. Osobina, koja za to ima najveći značaj, je opća namjenjenost Arduina što ga u proizvodnji čini skupljim od ploča koje su napravljene da služe samo jednoj svrsi.

Obzirom sam kroz razvoj ovog projekta koristio Arduino Mega, sve buduće reference će se odnositi na Mega model.

3.1 Proces kompajliranja

Programski jezik u kojem se pišu Arduino datoteke koje sadrže izvorni kod je C++. Međutim većina standardnih biblioteka su preuzete iz C programskog jezika, zbog male količine radne memorije kontrolera. Arduino datoteke je moguće prepoznati po njihovoj ".ino" eksteniziji. Ove datoteke se još nazivaju i skicama.

Nakon što se pokrene proces kompajliranja projekta, Arduino okruženje pravi male promjene u kodu, kako bi se nakon toga mogao proslijediti gcc i g++ kompajleru. U ovoj fazi se sve datoteke u direktoriju kombinuju u jednu i na njen početak se dodaje `#Include<Arduino.h>` zaglavlje. Zatim slijedi povezivanje koda sa standardnim Arduino bibliotekama, a nakon toga i sa ostalim bibliotekama uključenim u direktorji skice. Kako bi bila uključena u proces prevođenja, biblioteka (njen .cpp i .h dokument) se mora nalaziti na `libraries\{NazivBiblioteke}` putanji.

Kada je skica povezana i prevedena, vrijeme je da takva bude prebačena u Arduino memoriju gdje će se izvršavati.

[1] https://github.com/arduino

3.2 Pisanje koda

Kao što je spomenuto u sekciji 3.1, Programski jezik koji koristimo pri programiranju Arduino kontrolera je C++. Najjednostavnija ".ino" datoteka se sastoji iz dvije funkcije:

1. Setup
2. Loop

Funkcija "Setup" se izvršava samo jednom pri pokretanju kontrolera i služi za inicijalizaciju komponenti i objekata. Druga funkcija, pod nazivom "Loop", je sama srž načina rada ovih ploča. Kod koji se nalazi unutar ove funkcije će se ciklično izvršavati sve dok je Arduino upaljen. Taj kod je sekvenca koja predstavlja i ponašanje isprogramirane ploče u upotrebi.

3.3 Razvojno okruženje

3.3.1 Arduino IDE

Arduino posjeduje svoje oficijelno okruženje pod nazivom "Arduino IDE". Ono dolazi u dvije verzije - online[2] i verzija koju instaliramo na lokalni računar. U ovom tekstu ću se fokusirati na offline okruženje, jer se uz njegovu instalaciju automatski instaliraju i Windows driveri potrebni za prebacivanje koda na Arduino..

Pri kreiranju prve skice ispred korisnika se nalazi "prazna" datoteka.

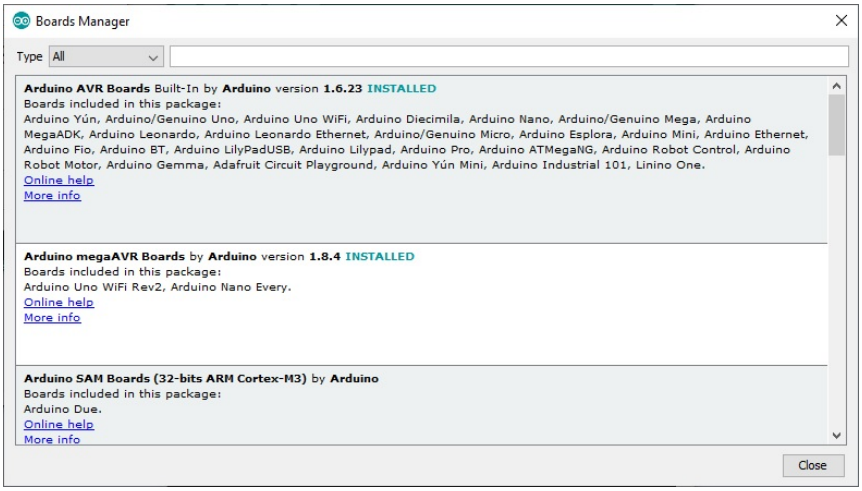
```
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Alati

Boards manager je opcija koja se nalazi u alatnoj traci pod stavkom "Tools". Koristeći ovaj alat, biramo trenutnu arhitekturu ili model ploče na koju ćemo postaviti kod. S obzirom da je program otvorenog koda, kroz ovaj proces se mogu insatlirati i

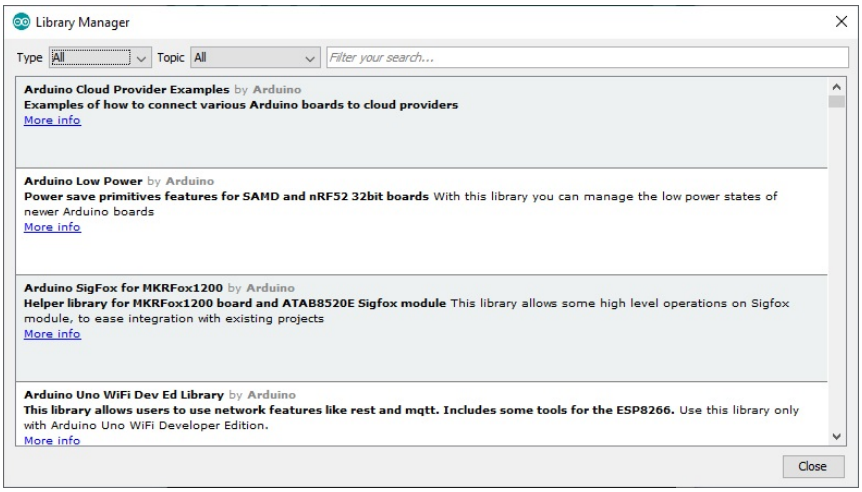
[2] https://create.arduino.cc/editor

datoteke koje nisu ugrađene u IDE, s tim da je u tim slučajevima potrebno obratiti pažnju na sigurnost. Na ovaj način moguće je koristiti ovo razvojno okruženje u svrhu pisanja koda i za druge platforme ili klonove Arduino ploča. Iz prozora je moguće preuzeti i instalirati nove ploče, kao i nadograditi već postojeće.



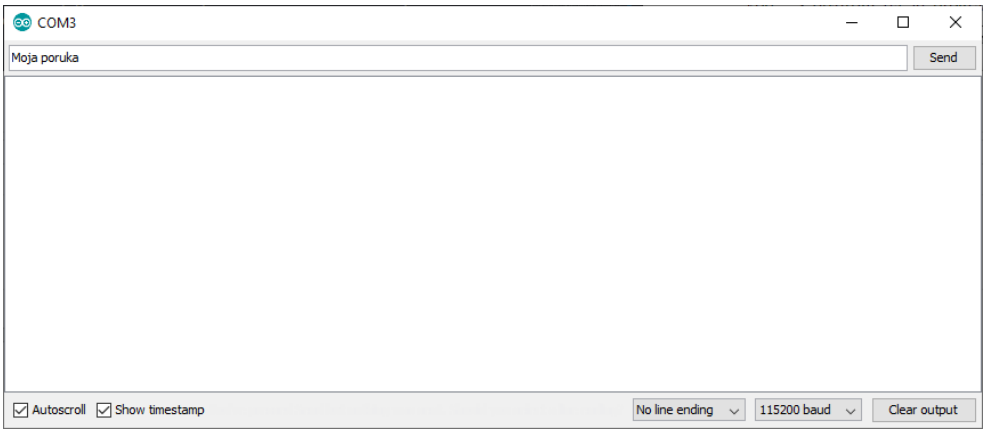
Slika 1: Boards manager prozor

Library manager je ovaj alat se također nalazi pod stavkom "Tools". Arduino IDE već dolazi sa predefinisanim repozitorijem biblioteka koje se nude za preuzimanje i instaliranje. Ukoliko korisnik želi preuzeti biblioteke iz drugih izvora to je moguće uraditi na dva načina: Instalacijom direktno iz ".zip" datoteke ili dodavanjem putanje repozitorija u postavkama okruženja. Brisanje se mora raditi ručno na putanji `%HOMEPATH%\Documents\Arduino\lib`



Slika 2: Library manager prozor

Serial monitor je jedan od načina da se uspostavi komunikacija sa Arduino uređajem dok je isti pokrenut. Kada se ovaj alat pokrene, korisnik može pomoću serijske komunikacije razmjenjivati tekstualne poruke sa Arduinom. Pored toga, na dnu prozora se nalazi padajući izbornik čija trenutno odabrana stavka definiše brzinu komunikacije u bitovima po sekundi.



Slika 3: Serial monitor prozor

3.3.2 PlatformIO

Pored Arduino IDE, programerima na izboru stoji još razvojnih okruženja u kojima mogu razvijati svoje Arduino kodove. Jedno od tih okruženja je PlatformIO.

PlatformIO se instalira u vidu nadogradnje za Visual Studio Code, koji i sam spada u kategoriju programa otvorenog koda. Kao i Arduino IDE, i ovaj alat nudi izbor ploče koja će se prorgamirati, među kojima je, pored samo Arduina, oko 700 ponuđenih[3]. S obzirom da je sada razvojno okruženje Visual Studio Code, to donosi sve njegove prednosti(markiranje sintakse, Intellisense, personalnim postavkama okruženja itd.).

Razlog, pored već navedenih, zbog kojeg sam ja izabrao PlatformIO jeste njegova arhitektura projekta. Naime, u ovoj platformi se koristi standardna arhitektura cpp-a, što je nekome ko dolazi iz tog svijeta, daleko lakše za održavati. Još jedna minorna razlika između ova dva okruženja je to da glavna datoteka sa izvornim kodom u PlatformIO nije .ino, već "main.cpp" u kojoj je obavezno uključiti `Arduino.h` biblioteku.

[3]https://platformio.org/

3.4 Tipovi varijabli

Pošto ploča ima ograničenu radnu memoriju, izbor tipova podataka je veoma bitan zbog njihovog predefinisanog memorijskog prostora koji zauzimaju. Neki od najčešće korištenih tipova su:

- bool (8 bita) - može sadržati jednu od dvije vrijednosti (da ili ne)
- byte (8 bita) - cijeli broj od 0 do 255, bez predznaka
- char (8 bita) - cijeli broj između -127 i 127 kojeg će kompajler pokušati prevesti u karakter
- word (16 bita) - cijeli broj bez predznaka između 0 i 65 535
- int (16 bita) - cijeli broj između -32 768 i 32 767
- long (32 bita) - cijeli broj između 2 147 483 648 i 2 147 483 647
- unsigned long (32 bita) - cijeli broj između 0 4 294 967 295, bez predznaka.
- float (32 bita) - broj sa plutajućom tačkom između -3.4028235E38 i 3.4028235E38

Niz je indeksirana kolekcija varijabli bilo kojeg tipa.

String se ,pored već navedenih tipova, u praksi veoma često koristi kao tip podatka. Ovaj tip podatka nema fiksnu veličinu koju zauzima u memoriji i sastoji se od niza karaktera. Ono što ga čini korisnim su funkcije koje su ponuđene za rad sa stringovima kao što su:

- CharAt(n) - vraća karakter na poziciji n
- IndexOf(c) - vraća prvu poziciju karaktera c u stringu
- Concat(val) - dodaje vrijednost iz varijable val na kraj stringa
- Replace(sub1, sub2) - vrši zamjenu svih instanci sub1 instancom sub2 u stringu
- Substring(n, k) - vraća komad stringa između n i k pozicija
- Lenght() - vraća dužinu stringa

Pokazivači su tipovi podataka koji upućuju na adresu. Diferenciraju se pomoću znaka *. Ako je varijabla x, onda je &x adresa te varijable.

3.5 Konstante

`Arduino.h` biblioteka dolazi sa korisnim konstantama.

- INPUT | OUTPUT - ova konstanta upravlja električno ponoašanje pina, i priprema pin da prihvata ulazne ili šalje izlazne signale.
- HIGH | LOW - odnosi se na pinove i ima može imati različito ponašanje
    1. Kada je pin postavljen kao INPUT, tada se radi o povratnoj vrijednosti koju vraća očitavanje struje sa tog pina. Ukoliko je struja u njemu veća od 3.0V ili 2.0V (u zavisnosti od ploče koja se koristi), vratit će se HIGH, a ukoliko je manje, LOW.
    2. Kada je pin postavljen kao OUTPUT, sa HIGH će se njen izlaz postavlja na 3.3V ili 5V (u zavisnoti od ploče koja se koristi), a sa LOW na 1.5V ili 1.0V.
- e | E - koriste se zbog lakše čitljivosti koda te predstavljaju eksponencijalni dio izraza 10^n koji se množi sa brojem koji mu prethodi.

3.6 Funkcije

Ono što omogćava kontrolisanje ulaza i izlaza Arduino ploče jesu funkcije `Arduino.h` biblioteke.

delayMicroseconds(value) pravi zastoj vremenske dužine value u mikrosekundama.

micros() vraća unsigned long koji predstavlja broj mikrosekundi koje su prošle od paljenja Arduino ploče.

pinMode(pin, INPUT | OUTPUT) upravlja ponašanjem pina i na osnovu konstante INPUT ili OUTPUT određuje da li će pin očitavati ili slati struju na svom kraju.

digitalRead(pin) očitava trenutno stanje pina čije je ponašanje postavljeno na očitavanje i vraća LOW ili HIGH konstantu u zavisnosti od voltaže koju očitava.

digitalWrite(pin, HIGH | LOW) postavlja jačinu protoka struje na izlazu pina na visoko ili nisko.

analogRead(pin) vraća integer vrijednost između 0 i 1024, koja je direktno proporcionalan jačini struje na pinu.

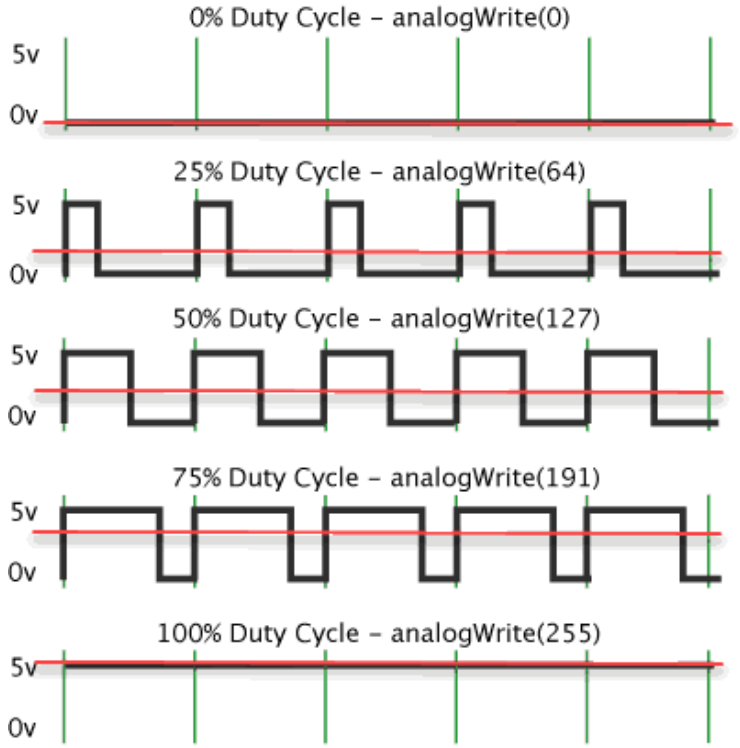
analogWrite(pin, value) radi modulaciju širine pulsa u što simulira jačinu struje na izlazu pina, pošto analogni izlaz na Arduinu ne postoji. Value ima raspon između 0 i 255 što je proporcionalno prividnoj jačini struje koju će pin proizvesti.

Serial.Begin(rate) inicijalizira serijski protok podataka, gdje je rate brzina prenosa podataka u bitima po sekundi.

Serial.Read() vraća prvi bajt serijske komunikacije na ulazu.

Serial.Write(String) šalje podatke u binarnom obliku na serijski izlaz.

attachIntterupt(digitalPinToInterrupt(pin), ISR, mode) Povezuje stanje pina mode (CHANGE, LOW, RISING, FALLING) sa prekidom u izvršenju loop funkcije i poziva ISR (rutina interapcije) koji je funkcija koja ne prima niti vraća parametre.



Slika 4: Primjer modulacije širine pulsa

3.7 Pinovi

Svaka verzija na sebi ima različit broj i mogućnosti pinova, ali svaka posjeduje barem po jedan pin svake vrste (sa izuzecima), kako bi se omogućile jednake funkcionalnosti svih ploča.

Dakle tipovi pinova koji postoje su:

- Digitalni - mogu biti ulazni i izlazni. Postavljaju se i očitavaju samo 2 stanja, pomoću funkcije pinMode() (poglavlje 3.6) se odabire režim u kojem će raditi, a nakon toga se može ili slati struja jačine 1.0V/1.5V ili 3.3V/5V postavljajući digitalWrite() (poglavlje 3.6) HIGH ili LOW respektivno ili očitavati jačina struje koristeći digitalRead() (poglavlje 3.6)
- PWM - digitalni pinovi koji podržavaju Pulse Width Modulation, tj. modulaciju širine pulsa (Slika 4)
- TX - digitalni pin, koristi se za serijsku transmisiju podataka, Serial.Write() (poglavlje 3.6)vrši ispis na ovaj pin.
- RX - digitalni pin, koristi se za serijsko čitanje podataka, Serial.read() (poglavlje 3.6) vrši učitavanje sa ovog pina.
- SCL - linija sata koja se koristi u I^2C komunikaciji (sekcija 7.3.2)
- SCL - linija podataka koja se koristi u I^2C komunikaciji
- INT - digitalni pinovi koji podržavaju prekide, aktiviraju se attachIntterupt() (poglavlje 3.6) funkcijom
- Analogni - pinovi koji imaju mogućnost analognog čitanja jačine struje pomoću analogRead() (poglavlje 3.6) funkcije
- Power - pinovi vezani za ulaz i izlaz struje u ploči
  - 5V - izlaz sa strujom jačine 5V
  - 3.3V - izlaz sa strujom jačine 3.3V
  - GND - uzemljenje
  - VIN - pin kroz koji se Arduino može napajati strujom od 9V
  - RESET - spajanjem na ground restartuje Arduino

3.8 MEGA

Arduino MEGA je jedna od verzija Arduino ploča. Na sebi ima ATMEGA2560 mikrokontroler.

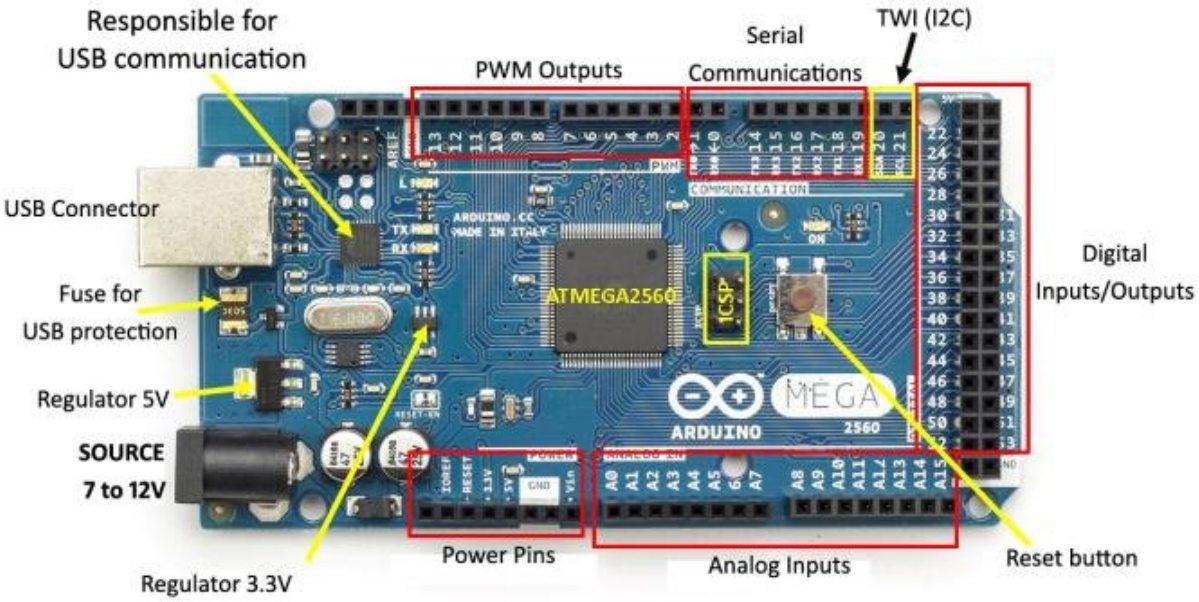
Ploča zahtjeva ulaznu struju od 6V do 20V, s tim da je preporučeno između 7V i 12V. Načini na koje se struja može dovesti u kontroler su kroz USB A interfejs, koji se ujedno

i koristi za povezivanje sa računarom, kroz 2.1mm priključak za napajanje sa pozitivnim centrom ili direktno kroz VIN pin.

Flash memorija	128 KB
Veličina bootladera	8 KB
SRAM	8 KB
EEPROM	4 KB
Brzina sata	16 MHz
Jačina struje I/O pinova	40mA
Jačina struje na 3.3V pinu	60mA

Tabela 1: Arduino MEGA specifikcaije

Na slici 5 je prikazana Arduino MEGA ploča te su pinovi označeni njihovim ulogama.



Slika 5: Šema pinova na ATMEGA2560

Dužina	101.6 mm
VŠirina	53.3 mm

Tabela 2: Arduino MEGA dimenzije

4 HC05

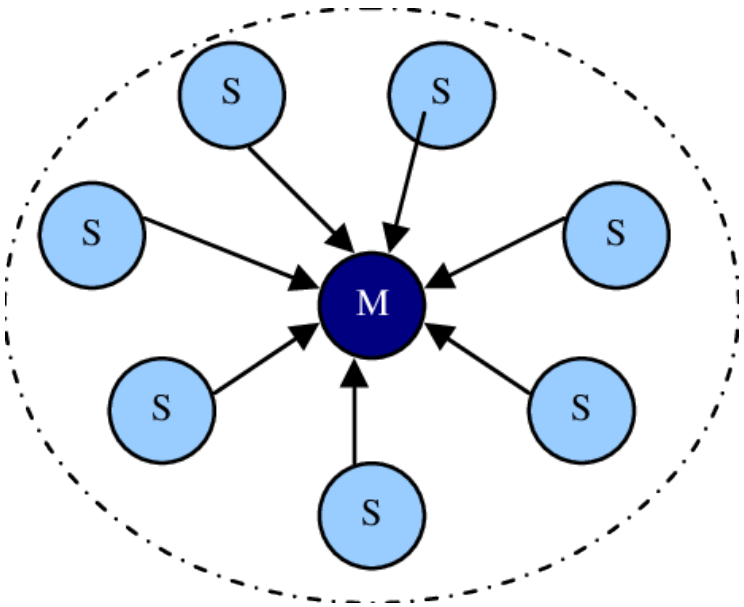
Obzirom da će se robotom moći upravljati pomoću Android aplikacije koja šalje bluetooth pakete, moramo osigurati način da ti paketi budu prihvaćeni od strane mikrokontrolera. To je u projektu postignuto koristeći bluetooth modul pod nazivom "HC-05". Obzirom da je malih dimenzija i zanemarljive težine, HC-05 neće imati značajan utjecaj na sposobnost robota da se balansira.

4.1 Tehnologije

Prije nego što pređem na temu samih specifikacija uređaja, posvetit ću nekoliko paragrafa tehnologijama na kojima se rad modula zasniva.

4.1.1 Piconet

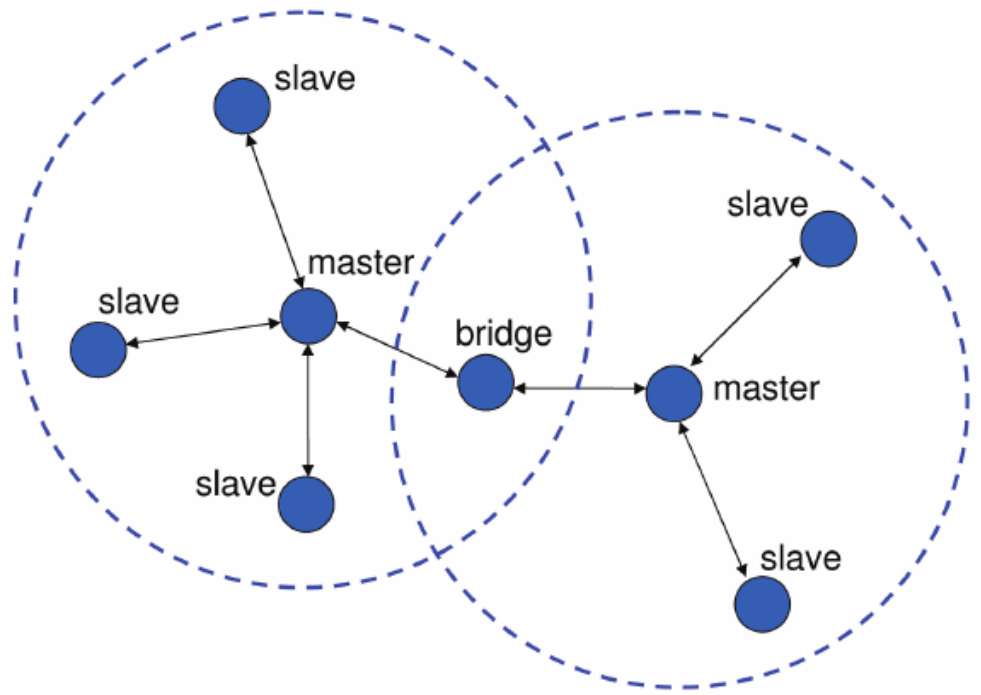
Piconet je ad hoc mreža koja povezuje 2 ili više uređaja povezanih pomoću bluetooth protokola koji imaju usklađen sat, kao i sekvencu skoka. To omogućava *master* uređaju da bude povezan na 7 aktivnih i do 255 neaktivnih *slave* uređaja. Zbog toga što bluetooth sistemi rade na 79 kanala, šansa da dođe do interferencije dva piconeta je oko 1.5%.



Slika 6: Piconet

4.1.2 Scatternet

Scatternet je mreža koja se sastoji od 2 ili više piconeta. Kako bi ovo bilo moguće, jedan slave uređaj mora biti povezan na 2 master uređaja ili se master uređaj iz jedne mreže ponaša kao slave uređaj druge.



Slika 7: Piconet

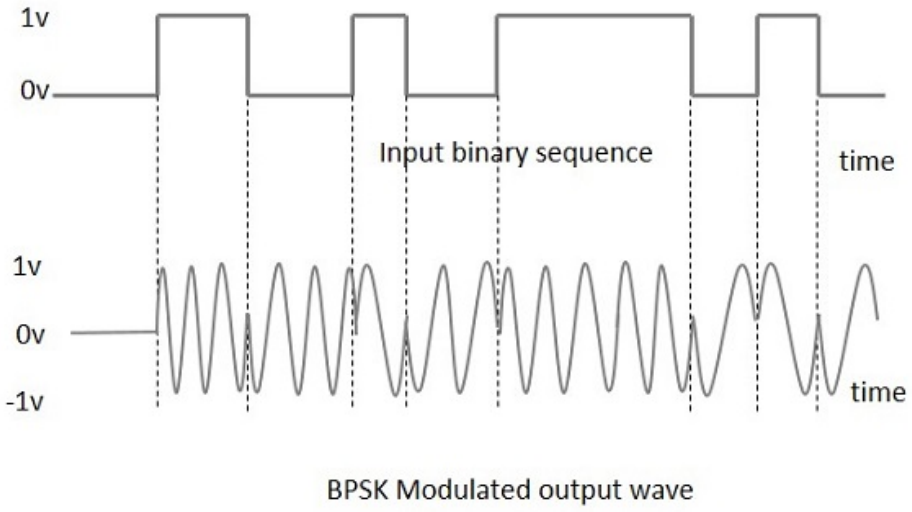
4.1.3 Adaptive Frequency Hopping

Skokovi u frekvenciji su neizbježna karakteristika bluetooth uređaja. Zbog toga što su bluetooth uređaji stalno u pokretu (mobiteli, satovi, slušalice, zvučnici itd.), mora postojati način da se izbjegne interferencija između ovih uređaja. Adaptive Frequency Hopping nastoji riješiti ovaj problem, mijenjajući frekvenciju na kojoj bluetooth uređaj radi. Što se tiče adaptivnosti, taj dio naziva proizlazi iz činjenice da uređaj prvo skenira kanale tražeći kanal sa najmanje prometa.

4.1.4 Phase Shift Keying

Phase-shift keying (PSK) je metoda digitalne komunikacije u kojoj se faza transmitovanog signala mijenja, čime prenosi informaciju. Ovo je moguće postići sa nekoliko metoda, od kojih je najjednostavnija binarni PSK koji koristi dvije suprotne faze od 0°i 180°(slika 8). U ovoj metodi, stanje svakog bita se poredi sa prethodnim te se tako na osnovu njihove

(ne)jednakosti mijenja trenutni bit ili ostavlja istim. Postoje i kompleksnije metode koje uključuju faze od +90°, -90°, ili čak i polovine svih dosadašnjih uglova.



Slika 8: Binarni PSK

4.1.5 Enhanced Data Rate

Enhanced Data Rate (EDR) je tehnologija u kojoj se pomoću PSK modulacione šeme postiže transmisija koja je 2 do 3 puta brža od one koju su prethodne bluetooth tehnologije mogle proizvesti. Pojavljuje se sa Bluetooth 2.0 verzijom kao opcionalna nadogradnja. Koristeći ovu tehonologiju brzina prenosa podataka doseže 2.1 Mbit/s.

4.1.6 BlueCore4-Ext

Ovaj čip ima implementiran bluetooth 2.0 + EDR. Na sebi ima 8 Mbit flash memorije i ima punu podršku za piconet. Uređaj također može služiti kao most za dvije piconet mreže čime nastaje scatternet.

4.2 Specifikacije

HC-05 na sebi ima BlueCore4 eksterni čip i na njemu se zasniva rad modula. Sa time dolazi i kompletna podrška za sve tehnologije koje čip podržava. Pored BlueCore4 mogućnosti, HC-05 podržava Adaptive Frequency Hopping čime se osigurava stabilnost komunikacije. Na sebi ima radio transiver koji radi na 2.4GHz. U mrežama se može ponašati kao master

i slave uređaj. S obzirom da je domet uređaja i do 10 metara, broj scenarija u kojem može biti koristan za komunikaciju je velik.

Potrebna struja	4V-6V
Jačina struje	30mA
Domet	do 10m
Brzina prenosa	do 3Mbps
Piconet uloge	Master, Slave, Master/Slave

Tabela 3: HC-05 specifikacije

Podržane brzine prenosa u bitovima po sekundi su: 9600,19200,38400,57600,115200,230400,460800.

Dužina	27 mm
VŠirina	12.7 mm

Tabela 4: HC-05 dimenzije

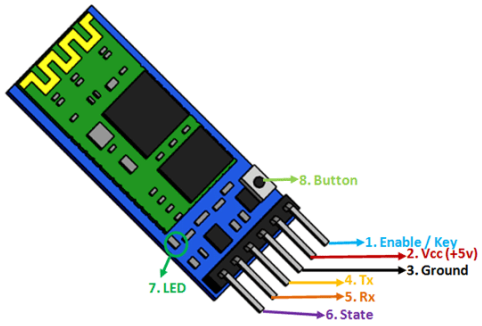
HC-05 također ima sposobnost pamćenja zadnjeg uvezanog uređaja, te automatsko povezivanje ukoliko isti bude pronađen pri paljenju.
Standard na kojem se zasniva bežična komunikacija je IEE 802.15.1[4]

4.3 Pinovi

HC-05 ukupno ima 6 pinova.

- **GND** se spaja na uzemljenje
- **5 VCC** je pin pomoću kojeg se struja (5V) dovodi u modul
- **RX** prima bitove koji će biti poslani uređaju na koji je modul spojen
- **TX** je putanja kroz koju se vrši transmisija bitova koji su primljeni od strane uređaja
- **STATE** pin štalje podatke o trenutnom stanju uređaja (Upaljen, ugašen, konektovan, diskonektovan itd.)
- **ENALBE** pin isključuje uređaj kada je na njega upućena struja od 3.3V

[4]http://www.ieee802.org/15/pub/TG1.html



Slika 9: HC-05 Šema

4.4 AT komande

Kada pri paljenju modula, držimo dugme na njemu pritisnutim, na pin PIO11 spajamo struju. Ovo će uređaj pokrenuti u komandnom načinu rada. Kroz RX se sada neće prihvatati podaci koji će se slati kroz antenu, već komande koje će konfigurisati uređaj. Te komande se nazivaju AT komande.

Kako bi mogli poslati ove komande uređaju, moramo se povezati na njega. To možemo učiniti koristeći Arduino "Serial Passtrough" skicu.

```
void setup() {
    Serial.begin(9600); // Interna softverska serijska komunikacija
    Serial1.begin(38400); // Komunikacija sa HC-05
    }

    void loop() {
        if (Serial.available())
        Serial1.write(Serial.read());

        if (Serial1.available())
        Serial.write(Serial1.read());
        }
```

Primjer koda 1: Serial passthrough

Nakon što na Arduino pošaljemo ovu skicu, kroz Serial Monitor možemo slati komande direktno na HC-05.

AT	Test komanda
AT+RESET	Reset uređaja
AT+VERSION?	Firmware verzija
AT+ORGL	Vraćanje na fabričke postavke
AT+ADDR?	MAC Adresa uređaja
AT+NAME<Param>	Postavljanje imena uređaja
AT+NAME?	Ime uređaja
AT+RNAME?<Param>	Ime bluetooth uređaja na osnovu adrese
AT+ROLE=<Param>	0-Slave, 1-Master, 2-Slave-Loop
AT+PSWD=<Param>	Postavljanje lozinke
AT+PSWD?	Trenutna lozinka
AT+UART=<Param>,<Param2>,<Param3>	Postavlja baud rate, Stop bit i Parivost
AT+UART?	Baud rate, Stop bit i Parivost
AT+CMODE=<Param>	0-konekcija na fiksnu adresu 1-sve adrese
AT+CMODE?	Trenutni način rada
AT+BIND=<Param>	Postavlja fiksiranu adresu za konekciju
AT+BIND?	Trenutna fiksirana adresa za konekciju
AT+PMSAD=<Param>	Brisanje autentificiranog uređaja
AT+ RMAAD	Brisanje svih autentificiranih uređaja
AT+FSAD=<Param>	Pretraga autentificiranog uređaja
AT+ADCN?	Broj autentificiranih uređaja
AT+MRAD?	Posljednji autentificirani uređaj
AT+LINK=<Param>	Povezivanje s uređajem
AT+DISC	Prekid veze

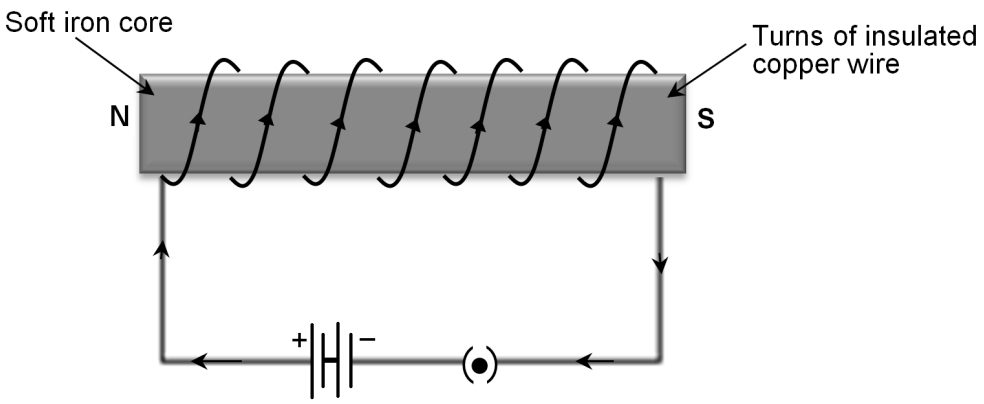
Tabela 5: Tabela AT komandi

5 Koračni motori

Koračni motori su tip DC elektronskog motora bez četkica. Revoluciju dijele u jednake korake odakle i dolazi njihov naziv. Ovi motori zbog toga mogu biti precizno pomjereni i zadržani na željenoj poziciji bez enkodera pozicije.

5.1 Elektromagneti

Elektromagneti su tipovi magneta koji svoja magneta svojstva pokazuju kada kroz njih teče elktrična struja. Jedno od svojstava protoka struje kroz provodnik je magnetno polje koje se stvara oko provodnika. Ovo svojstvo se može multiplicirati zavijanjem izoliranog provodnika u spiralu oko provodničkog jezgra. Za razliku od trajnih magneta, magnetna svojstva ovih uređaja mogu biti aktivirana po potrebi.



Slika 10: Elektromagnet

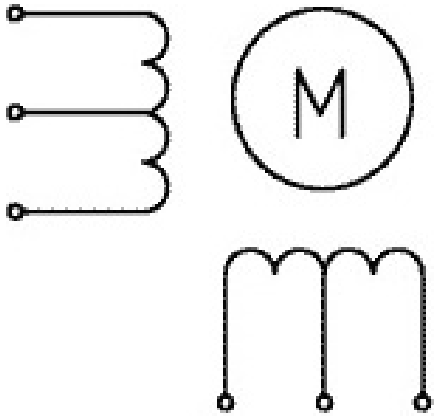
5.2 Proces rotacije ose

Koračni motori su građeni iz okvira koji čine elektromagneti i jezgra - željeznog zupčanika, što je ujedno i trajni magnet. Ovi magneti su najčešće kontrolisani od strane mikrokontrolera ili upravljača koji nije dio samog motora. Kako bi se osovina motora rotirala, potrebno je pustiti protok struje kroz jedan elektromagnet. Nakon što se povučen magnetnom silom zupčanik fiksira na poziciju, te se paljenjem drugog elektromagneta postiže rotacija ose motora na željenu poziciju. Ova rotacija predstavlja koncept koraka gdje prirodni broj N koraka čini jednu revoluciju osovine motora.

5.3 Tipovi

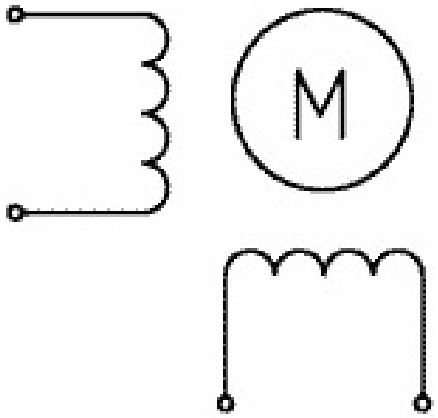
Postoje dva tipa koračnih motora:

Uniopolarni koračni motori za jedan elektromagnet sa središnjim izlazom (eng. center tap). To omogućava da promjenom kraja na kojem je uzemljenje elektromotora, samo polovina elektromagneta bude magnetizirana.



Slika 11: Šema unipolarnog motora

Bipolarni koračni motori imaju samo po jednu žicu na oba kraja elektromotora od kojih je jedna spojena na uzemljenje, a druga na izvor struje. Ovo omogućava jače magnetno djelovanje, kao i kreiranje magnetnog polja u oba smijera na zajedničkom prostoru.



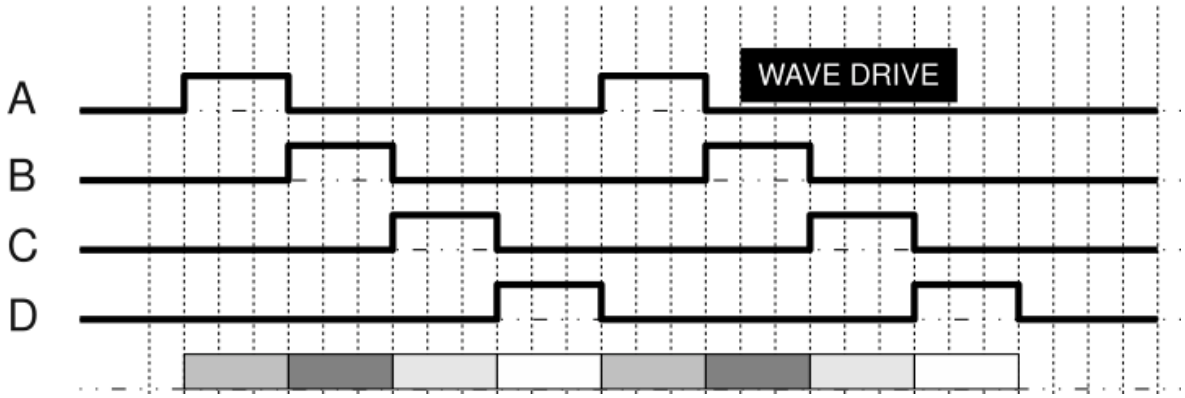
Slika 12: Šema bipolarnog motora

Prednost bipolarnih motora je veća vučna sila i jače magnetno polje, a kodunipolarnih su to manji troškovi zbog jednostavnije kontrole, kao i manji otpor.

5.4 Načini upravljanja

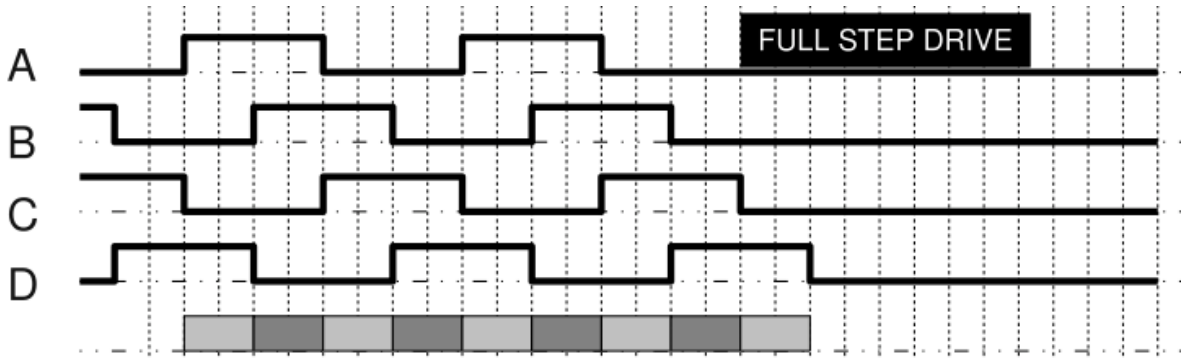
U praksi postoje 4 načina upravljanja koračnim motorima

Valno upravljanje je metoda gdje se u jednom trenutku šalje samo jedna faza. Ovo upravljanje se rijetko koristi jer ima jednak broj koraka kao upravljanje punim korakom, ali samo 50% njegove vučne sile. Ukoliko zupčanik ra rotoru ima 25 zuba, onda će se svaki taj zub pomjeriti pomoću 4 faze. To znači da će se sa 100 koraka napraviti jedna revolucija.



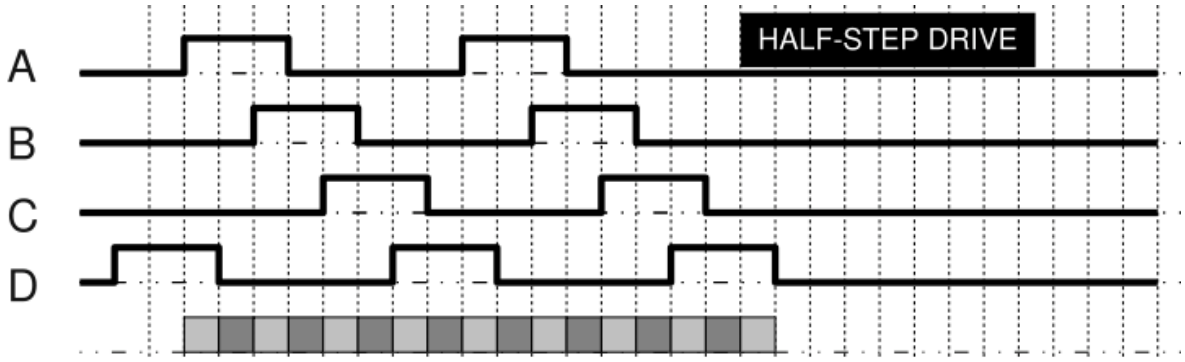
Slika 13: Valno upravljanje

Metoda punog koraka je najčešće korištena metoda upravljanja. U svakom trenutku ima tačno dvije faze. Ovom metodom se izvlači maksimalna vučna sila motora, dok također ima 100 koraka po revoluciji.



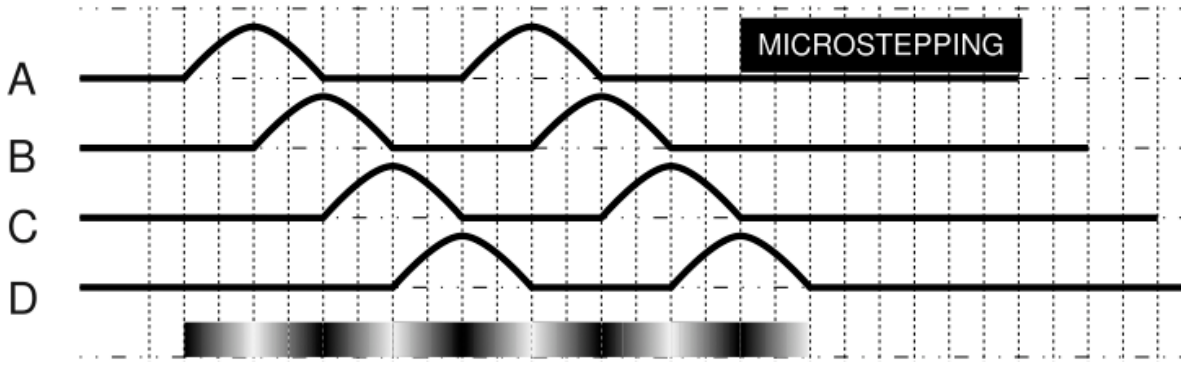
Slika 14: Metoda punog koraka

Polukoračno upravljanje radi promjenu između dvije faze za vrijeme jedne faze. Ovo motoru smanjuje vučnu silu, ali udvostručava rezoluciju. Kako je sada potrebno 8 faza kako bi se zupčanik pomjerio za jedan zub, jedna revolucija zahtjeva 200 koraka.



Slika 15: Polukoračno upravljanje

Mikrokoračanje je metoda u kojoj se sinusnom i kosinusnom funkcijom kreiraju faze. Ovo znatno smanjuje broj koraka u jednoj revoluciji, čime se povećava rezolucija samog motora. U ovim situacijama, ograničenje broja koraka u revoluciji postoji zbog statičkog trenja i zazora.



Slika 16: Mikrokoračanje

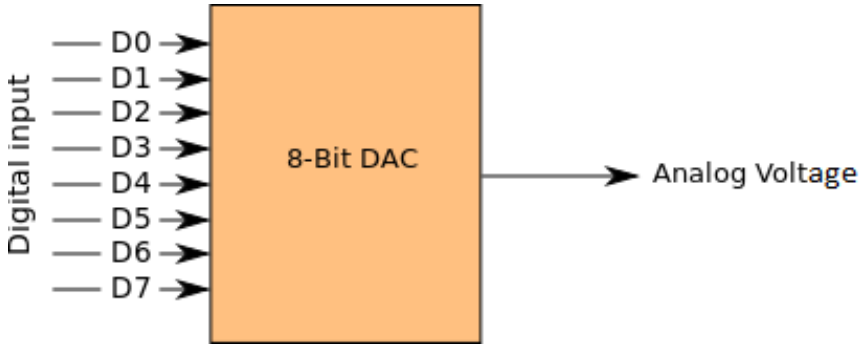
Kako bi se postigle veće brzine kretanja motora, potrebna je akceleracija koja će motor dovesti do pune brzine. U primjeni sam također zapazio da se deakceleracijom pri promjeni smjera postiže bolji rezultat kod promjeni smjera kretanja.

6 A4988

Upravljač motora A4988 je namijenjen za upravljanje bipolarnim koračnim motorima. Pomoću ovog uređaja, motoru motožemo dostaviti i do 2A po namotaju. Međutim, za jačinu struje se ne moramo previše brinuti jer A4988 ima ugrađenu zaštitu od prekomjerne struje (eng. overcurrent protection).

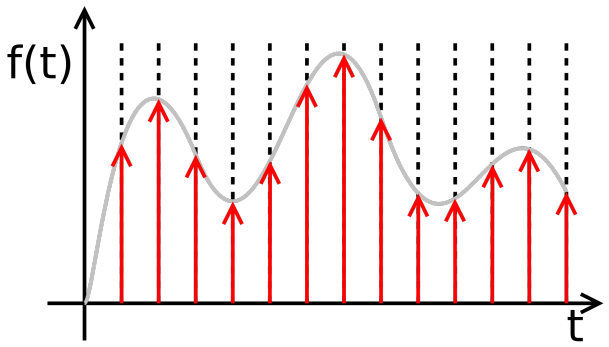
6.1 Digitalno-analogni konverter

U elektronici se dizajneri često susreću sa situacijom da je iz digitalnog potrebno dobiti analogni signal. Ovaj problem se riješava koristeći digitalno-analogne konvertere (eng. Digital-to-analog converter, DAC)



Slika 17: 8-bitni DAC

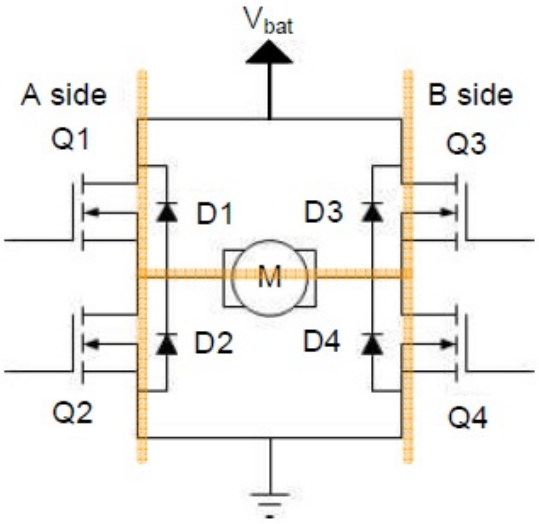
DAC funkcioniše tako što konvertuje abstraktni broj konačne preciznosti u analogni signal čija je voltaža proporcionalna ulaznoj vrijednosti. Idealan DAC konvertuje abstraktni broj u niz impulsa, nakon čega se, koristći linearnu interpolaciju, popunjava signal između impulsa.



Slika 18: Primjer idealne konverzije iz digitalnog u analogni signal

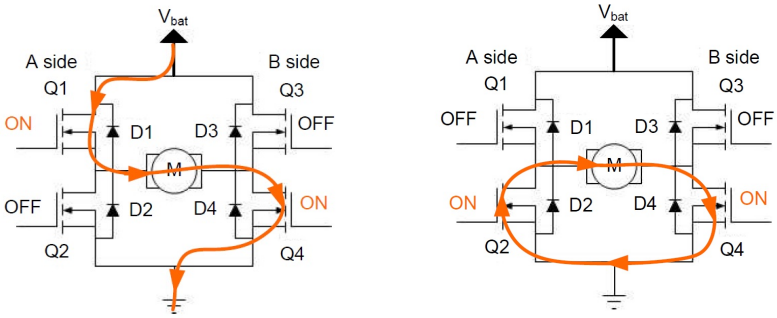
6.2 H-most

H-most je električno kolo čijij je zadatak kontrola toka struje. Najčešće se pojavljuju kao već integrisane komponente. H-most je građen iz četiri prekidača čijim se kombiniranjem u motoru mijenja tok struje.



Slika 19: H-most

H-most ima dva moda u kojima prekidači mogu biti postavljeni, a to su brzo i sporo propadanje (eng. decay). U ovom kontekstu, propadanje prestavlja brzinu kojom se jačina struje u navoju može spustiti na 0.



(a) Brzo propradanje (b) Sporo propadanje

Slika 20: Modovi rada H-mosta

6.3 Miješano propadanje

U svakom koraku motora, protok struje se kontrolira koristeći vrijednost eksternih rezistora čija je uloga opažati ulazni protok struje, referentnu voltažu i izlaznu voltažu digitalno-analognog konvertera koja se kontroliše od strane translatora. Ukoliko je trenutni izlaz DAC-a u niži u odnosu na prošli, onda se mod aktivnog mosta postavlja na mod miješanog propadanja (jedno od svojstava A4988 čipa). Ukoliko je trenutni izlaz DAC-a viši, postavlja se na sporo propadanje. Ovo dinamično mijenjanje modova poboljšava performanse motora u vidu smanjenja distorzije talasa koji prenosi signal.

6.4 Pinovi

A4988 upravljač, na sebi ima 16 pinova:

ENABLE se koristi za privremeno isključenje uređaja. Kada se na njemu nalazi visoki nivo voltaže, uređaj neće primati naredbe.

MS1, MS2, MS3 su pinovi koji određuju rezoluciju mikrokoračanja motora. Šablonskim potavljenjem visokog voltažnog nivoa, upravljač će način rada prebaciti na drugi mikrokoračni mod.

MS1	MS2	MS3	Rezolucija u veličini koraka
N	N	N	1/1
V	N	N	1/2
N	V	N	1/4
V	V	N	1/8
V	V	V	1/16

Tabela 6: Šabloni MS pinova

RESET vraća translator na početno stanje. Dok je visoki nivo voltaže, svi ulazi se ignorišu.

SLEEP se koristi za uštedu energije i postavljajući na njega visoki nivo voltaže, uređaj se stavlja u "sleep mode".

STEP pin - slanjem impulsa voltaže visokog nivoa na ovaj pin zatim vraćanjem istog niski nivo voltaže, upravljač pravi jedan korak na motoru u smjeru zavisnom od stanja DIR pina i veličine zavisne od trenutne rezolucije.

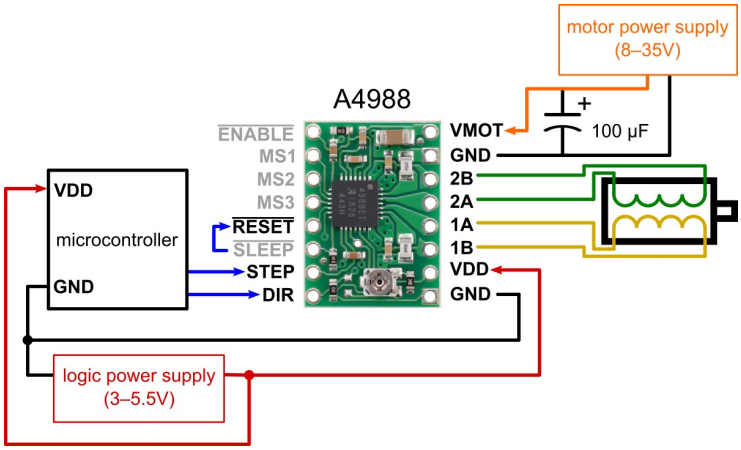
DIR služi za određivanje smjera u kojem će se osa motora okretati. Ovaj pin mora konstantno imati na sebe pušten nivo voltaže gdje visoki i niski određuju smjer.

VMOT je ulaz za struju kojom će se napajati motor. Voltaža ove struje treba biti između 8V i 35V.

VDD predstavlja zvor struje za upravljač.

GND - na upravljaču se nalaze 2 pina za uzemljenje, jedan za struju koja napaja sam uređaj, drugi za struju koja napaja motore.

1A,1B,2A,2B su pinovi koji se povezuju direktno na motor. Parovi za navoje su 2A i 2B, kao i 1A i 1B.



Slika 21: Najjednostavnija šema spajanja A9488

6.5 Specifikacije

Potrebna struja	3V-5V
Maksimalna jačina struje	2A
Termalno gašenje	-20°C, + 80°C
Dužina	do 20.3mm
Širina	do 15.2mm

Tabela 7: A4988 specifikacije

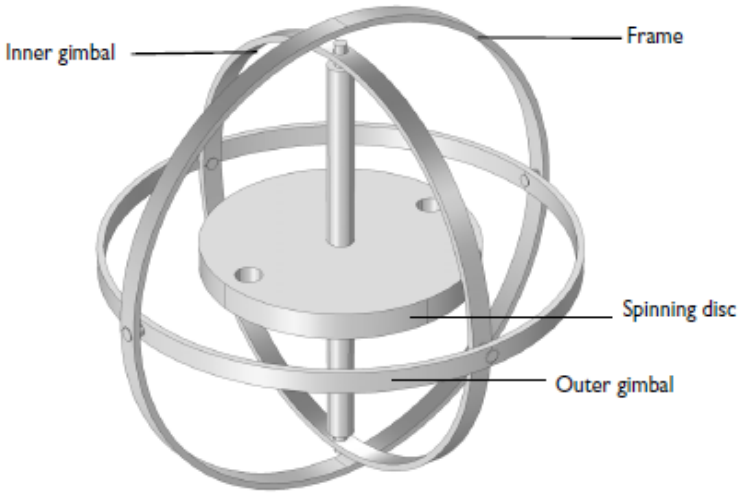
A4988 na sebi ima potenciometar kojim se može kontrolisati jačina struje koja prolazi kroz upravljač.

7 MPU6050

MPU6050 je uređaj za praćenje kretanja i orijentacije tijela na kojem se nalazi. Kombinacija žiroskopa i akcelerometra (koji imaju po 3 ose) sa Digital Motion Processor-om daje vrlo precizne podatke o trenutnom položaju uređaja.

7.1 Žiroskop

Žiroskop je uređaj koji, koristeći zemljinu gravitaciju, određuje svoju orijentaciju. Sastavljen je od diska koji se vrti (rotora), osnove na kojoj se disk nalazi na kojoj se također nalaze tri stacionarna prstena. Kada se orijentacija promijeni, zahvaljujući svojoj rotacionoj sili, rotor ostaje u istom orijentacionom odnosu sa zemljom, što omogućava očitavanje pomjeranja u odnosu na stacionarni dio uređaja oko žiroskopa.



Slika 22: Žiroskop

7.2 Akcelerometar

Akcelerometar je mjerni instrument pomoću kojeg se mjeri promjena u brzini kretanja tijela. Mjerenjem inercijske sile koja djeluje na referentnu masu u akcelerometru, prema drugom Newtonovom zakonu se može izračunati ubrzanje.

$$a = \frac{F}{m}$$

Zbog ovoga će akcelerometar koji je u mirnom stanju očitavati ubrzanje od 9.81m/s^2, a akcelerometar koji pada 0m/s^2.

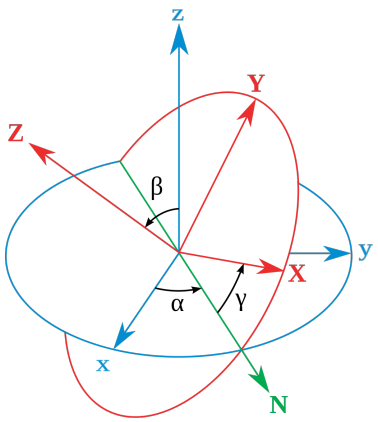
7.3 Digital Motion Processing

Digital Motion Processor (DMP) je interni proceseor koji se nalazi na MPU6050. Nažalost, ovaj procesor je zatvorenog koda i nije dokumentovan. Zbog toga se u Arduino projektima koristi biblioteka "I2Cdevlib"[5] koja je nstala kao proces reverznog inžinjeringa, biblioteke koja preuzima podatke iz DMP-a

7.3.1 Euluerovi uglovi

Eulerovi uglovi se koriste kako bi se predstavila orijentacija tijela u odnosu na fiksirani koordinatni sistem. Ovom metodom se utvrđuje rotacija tijela sa osama x, y, z u tijelo X, Y, Z.Prvo je potrebno odrediti N-osu koja se dobija vektorskim proizvodom $N = Z \times Z$. Zatim se definišu uglovi:

- α kao ugao između x-ose i N-ose
- β kao ugao između z-ose i Z-ose
- γ kao ugao između X-ose i N-ose



Slika 23: Primjer Eulerovih uglova na rotaciji x,y,z →X,Y,Z

Problemi koji se pojavljuju kod ovog pristupa rotaciji tijela su:

- Svaku orijentaciju nije moguće predstaviti koristeći samo jednu jednačinu.
- Tzv. "Gimbal lock" je problem koji nastaje kada ugao između bilo koje dvije ose postane 0°čime se gubi jedan stepen slobode koji rotacija ima.

[5] https://www.i2cdevlib.com/

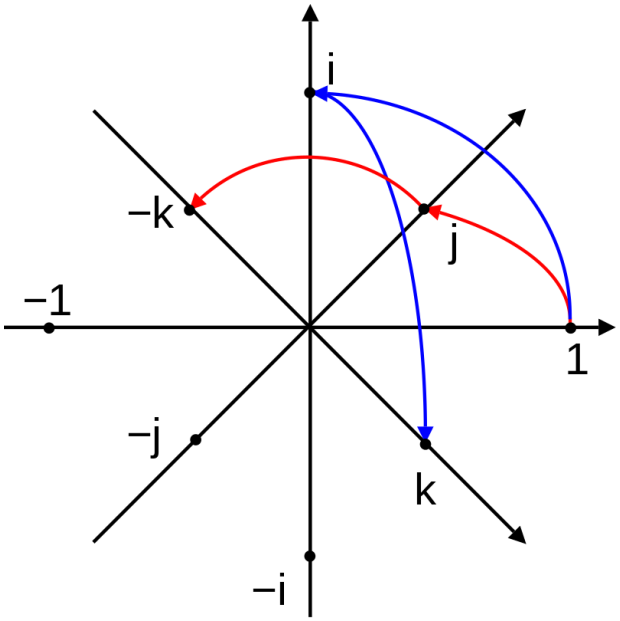
7.3.2 Kvaternioni

Kvaternioni su numerički sistem koji proširuje kompleksne brojeve. Sastavljeni su od imaginarnog i realnog dijela.

$$a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$$

U jednačini iznad, a, b, c i d predstavljaju realne brojeve, dok su i, j i k imaginarni. Pravila koja vrijede za množenje imaginarnih dijelova su:

$$\begin{aligned} ij &= k \\ ji &= -k \\ jk &= i \\ kj &= -i \\ ki &= j \\ ji &= -k \end{aligned}$$



Slika 24: Grafički prikaz proizvoda imaginarnih dijelova i i j

Konjugacija kvaterniona se vrši promjenom predznaka imaginarnom dijelu.

$$q = a + bi + cj + dk$$
$$q* = a - bi - cj - dk$$

Tenzor kvaterniona se dobija formulom:

$$||q|| = \sqrt{a^2 + b^2 + c^2 + d^2}$$

Inverzija kvaterniona se dobija formulom:

$$q^{-1} = \frac{q*}{|q|^2}$$

Rotacija (njeno opisivanje) jeste najčešći vid upotrebe kvaterniona. Rotacija tačke p kvaternionom koji opisuje rotaciju q ,može se riješiti formulom:

$$p' = qpq^{-1}$$

Prednost ove rotacije u odnosu na Eulerovu ili matričnu je ta što imaginarni dio onemogućava gubljenje stepena slobode, kao i to što svaka rotacija može biti opisana jednim kvaternionom rotacije.

7.4 I²C

I²C je serijski komunikacijski protokol zasnovan na master/slave modelu. U ovom modelu, jedan ili više master uređaja mogu kontrolisati jedan ili više slave uređaja. Podaci se prenose kroz dvije putanje:

- SDA (Serial Data) - kroz ovu putanju se šalju komunikacijski paketi
- SCL (Serial Clock) - putanja koja prenosi signal sata

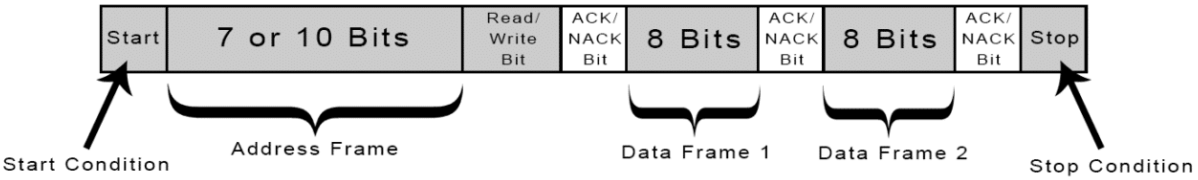
Podaci se kroz SDA prenose bit po bit te su sinkronizirani sa SCL signalima, kontrolisanim od strane master uređaja. Maksimalan broj slave uređaja za jedan master uređaj je 1008, dok je u obrnutoj direkciji maksimalan broj teoretski beskonačan.

7.4.1 Paketi

Komunikacija ovog protokola se zasniva na slanju paketa. Veličina mu nije fiksna, već se dinamički,pomoću početnog i stop signala, određuju početak i kraj mrežnog paketa. Konstrukciju jednog paketa čini 9 okvirova(6 tipova okvira):

- Početni signal - na SDA putanji se prebacuje sa visoke voltažne vrijednosti na nisku, prije nego što se na SCL učini isto.
- Adresni okvir - identificira slave uređaj sa kojim master želi komunicirati (7-10 bita)
- Bit primanja ili slanja - visoka voltažna vrijednost označava da master uređaj od ovog trenutka šalje podatke odabranom slave uređaju, dok niska označava da će početi proces primanja podataka.(1 bit)
- ACK/NACK - nakon svakog okvira se šalje ACK poruka sa informacijom pošiljaocu da li je njegova poruka uspješno primljena(1 bit)
- Okvir sa podacima (8 bit)
- Stop signal - proces obrnut početnom uslovu gdje se prebacuje na visoku voltažnu vrijednost

Okvir sa podacima se šalje nakon što master uređaj dobije pozitivan ACK frame od slave uređaja. Nakon svakog poslanog podatkovnog okvira, trenutačno se prema master uređaju šalje ACK/NACK okvir od strane slave uređaja čija se adresa nalazi u adresnom okvriu.



Slika 25: Primjer I^2Cpaketa

Brzina ove komunikacije može biti između 100kbps i 5Mbps.

Najveća mana ovog protokola je ograničenje okvirnog podatka na 8 bita.

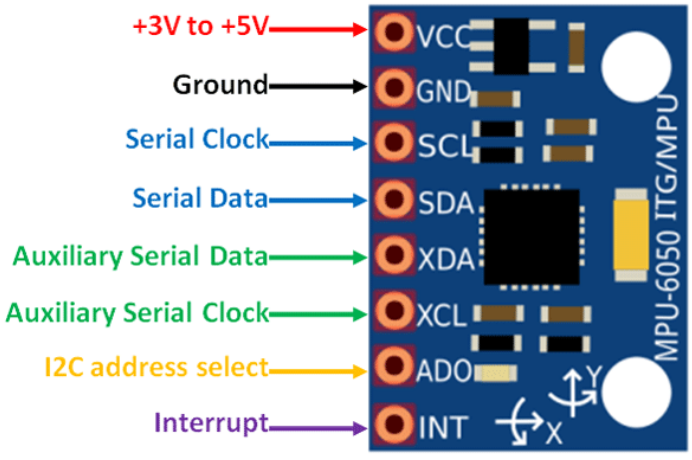
Potrebna struja	3V-5V
Komunikacijski protokol	I^2C
Dužina	21.2mm
Širina	do 16.4mm

Tabela 8: MPU6050 specifikacije

7.5 Specifikacije

7.5.1 Pinovi

- VCC - ulaz za struju
- GND - uzemljenje
- SCL - serijski sat
- SDA - serijska putanja podataka
- XCL - pomoćni sat, koristi se za konekciju na serijski sat drugih I2C uređaja
- XDA - pomoćna putanja podataka, koristi se za konekciju na serijsku putanju podataka drugih I^2Cuređaja
- AD0 - slave adresa uređaja, ovaj pin određuje nulti bit 7 bit-ne adrese i, kada je na njemu visoki stepen voltaže, mijenja adresu uređaja postavivši nulti bit na 1
- INT - digitalni pin koji se koristi za prekide.



Slika 26: Raspored pinova na MPU6050

7.6 I2Cdevlib

I2Cdevlib je biblioteka napisana u C++ programskom jeziku koja olakšava korištenje MPU6050 u Arduino projektima. Neke od funkcija koje ova biblioteka nudi su:

Initialize() postavlja čip u stanje koje je spremno za očitavanje. Ova funkcija postavlja brzinu sata na SCL-u.

dmpInitialize() vrši manipulaciju registara kojom se dmp čip stavlja u funkciju kao i inicijalizacija FIFO buffera, koji čuva podatke prije nego što budu poslati kroz SDA.

Set[X | Y | Z][Gyro | Accel]Offset() postavlja pomak orijentacije uređaja nastalih zbog grešaka u proizvodnji i nesavršene pozicije uređaja na tijelu.

Calibrate[Gyro | Accel]Offset(n) vrši kalibraciju žiroskopa i akcelerometra tako da trenutna pozicija bude nulta za oboje. Broj n predstavlja koliko puta će se ponoviti kalibracija prije uzimanja srednje vrijednosti.

getIntStatus() vraća informaciju o tome da li trenutno postoji signal za prekid koji je došao od MPU6050 uređajai.

dmpGetFIFOPacketSize() vraća veličinu paketa koja se nalazi u FIFO bufferu.

getFIFOCount() vraća broj bita koji se nalazi u FIFO bufferu.

getFIFOBytes(packet, size) upisuje size veličinu paketa iz FIFO buffera u packet varijablu.

dmpGetQuaternion(&q, packet) pretvara podatke iz paketa packet u kvaternion rotacije i pohranjuje ih u objekt tipa quaternion q.

mpu.dmpGetEuler(euler, &q) vrši konverziju kvaterniona rotacije q u Eulerove uglove koje pohranjuje u niz od 3 float varijable euler(α, β, γ).

8 PID kontroler

PID kontroler je kontrolna petlja čiji zadatak je da primijeni preciznu i vremenski prihvatljivu korekciju kontrolnoj funkciji. To postiže računajući vrijednost greške kao razliku ulazne i ciljane vrijednosti, uzevši u obzir proporcionalni, integralni i derivativni dio.

Ukoliko iz jednačine izostavimo neki od dijelova PID-a, onda tu jednačinu oslovljavamo nazivom koji u sebi također ne sadrži taj dio.

Ako su e, r i y, greška, ciljana vrijednost i ulazna vrijednost, respektivno, onda vrijedi da je:

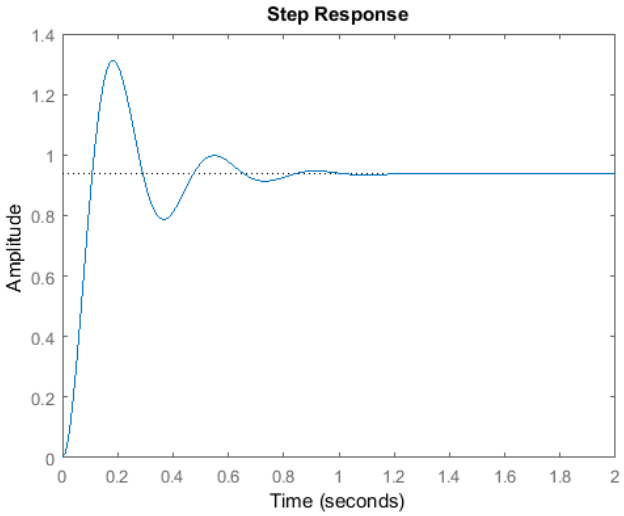
$$e(t) = r(t) - y(t)$$

8.1 Proporcionalni dio

Proporcionalni dio PID kontrolera ima sopstvenu izlaznu vrijednost u uzajamnom odnosu sa greškom. To znači da će se u P kontroleru, primjenjujući izlaznu vrijednost na pokretač, greška proporcionalno smanjivati. Kada se greška smanji na 0, tada će i izlaz biti 0.

$$P = K_p * e(t)$$

Ovaj dio, u teoriji zvuči korisnije nego što jeste, zbog toga što, u stvarnom svijetu, nije optimalno prestati sa primijenjivanjem ispravljačke sile u trenutku kada je ona 0 (npr. tijelo koje ima inerciju, grijač koji nastavlja grijati i kad se ugasi).



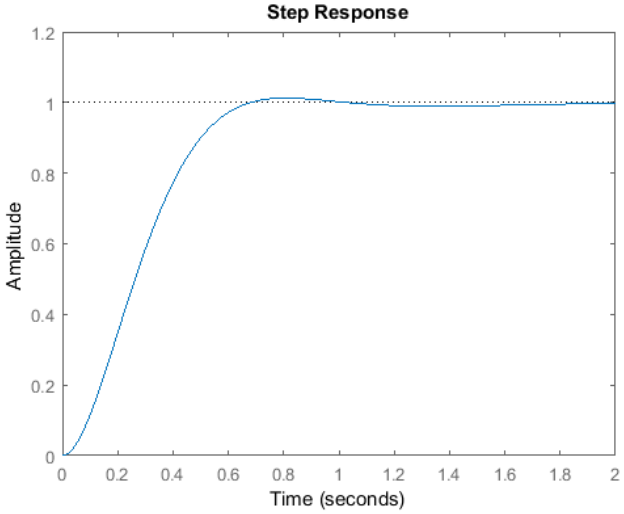
Slika 27: Graf P kontrolera

8.2 Integralni dio

Integralni dio akumulira grešku kroz vrijeme i ima sporiju reakciju od proporcionalnog dijela. Zadatak ovog dijela kontrolera je da eliminiše preostalu grešku i smanji prelijetanje (eng. overshooting).

$$I = K_i * \int_0^t e(t')dt'$$

Kada se greška smanjuje, usporava se rast integranlog dijela. Ovo rezultuje približavanjem proporcionalnog dijela 0 što se kompenzuje greškom akumuliranom u integralnom dijelu.



Slika 28: Graf PI kontrolera

8.3 Derivativni dio

Derivativni dio za zadatak ima da predvidi trend smanjenja greške u idućem koraku koristeći razliku između trenutne i prošle greške. Mjereći koliko brzo se greška u procesu mijenja, nastoji istu eliminisati.

$$D = K_d * \frac{de(t)}{dt}$$

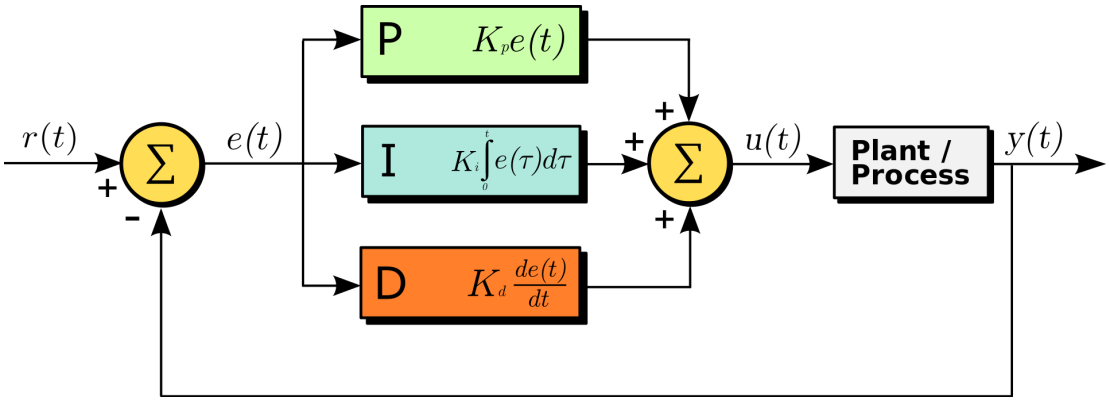
Uređaji koji generišu ulaz za PID često imaju šum. Derivativni dio također kao svoj zadatak ima da eliminiše grešku koja nastaje ovim šumom. Pomoću derivativnog dijela, nastaje tzv. prigušenje koje onemogućuje prelijetanje.

8.4 PID

Kombinujući sva tri dijela koja su dosad opisana, dobija se PID jednačina:

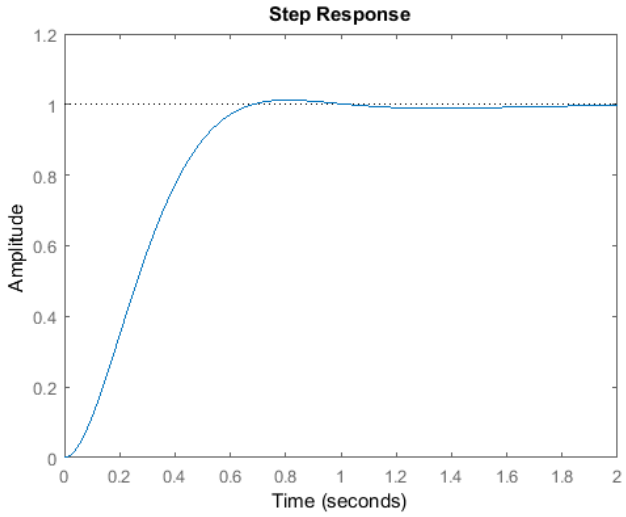
$$u(t) = K_p * e(t) + K_i * \int_0^t e(t')dt' + K_d * \frac{de(t)}{dt}$$

K_p,K_i i K_d se koriste kako bi se pojačao ili smanjio efekt koji proporcionalni, integralni i derivativni dio imaju na izlaz jednačine.



Slika 29: Petlja sa PID kontrolerom

Greška u stanju mirovanja je greška koja nastaje proporcionalnim dijelom jednačine. Ona predstavlja osciliranje oko ciljane vrijednosti koje nastaje kao posljedica konstantnog proporcionalnog dijela koji će premašivati potrebnu vrijednost izlaza. Ovaj problem riješava integralni dio koji preuzima kontrolisanje izlaza sa smanjenjem greške.

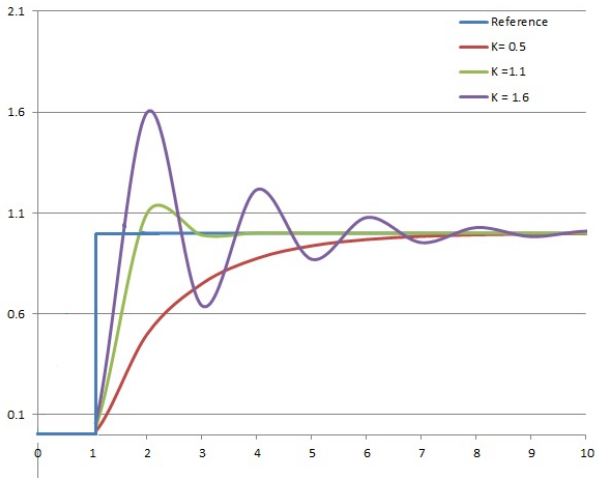


Slika 30: Graf PID kontrolera

8.5 Podešavanje parametara

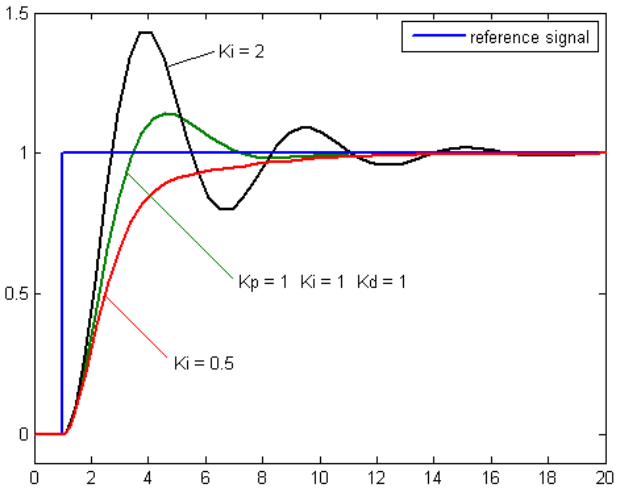
Jedan od najvećih problema sa kojim se dizajneri PID kontrolera susreću jeste podešavanje parametara.

Ručno podešavanje paramtara je metoda koja se započinje tako što se K_i i K_d inicijalno postave na 0. Nakon toga, K_p je potrebno povećavati sve dok petlja ne počne oscilirati. Vrijednost K_p bi trebala biti podešena na jednu polovinu vrijednosti.



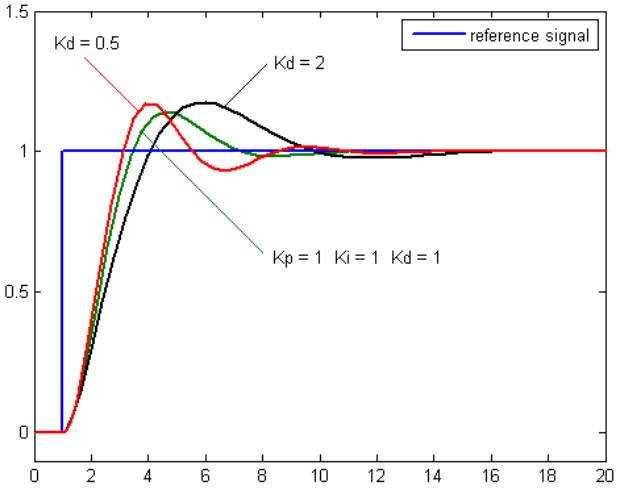
Slika 31: efekt K_p parametra na izlaz

Kada je vrijednost K_p postavljena, prelazi se na podešavanje K_i parametra. Potrebno je parametar povećavati sve dok se svaki pomak ne može korektovati u najmanjem vremenu neophodno kako bi sistem nastavio raditi. Previše K_i vrijednosti dovodi sistem u nestabilno stanje.



Slika 32: efekt K_i parametra na izlaz

Ukoliko je potrebno, K_d treba povećavati dok se vrijeme osciliranje ne smanji na minimum. Brzi sistemi obično imaju podešen ovaj parametar tako da dolazi do malog prelijetanja kako bi se sistem prije doveo u stanje balansa. Neki sistemi nemaju toleranciju na prelijetanje pa se ova metoda ne može primijeniti.



Slika 33: efekt K_d parametra na izlaz

8.6 Modifikacije

8.6.1 Integralni zamah

Integralni zamah (eng. Integral windup) je problem koji nastaje kada pokretač nije u stanju pružiti silu koja mu je zadana izlaznom vrijednošću. Kod ovih slučajeva, integralni dio akumulira grešku što dovodi do prelijetanja. Postoji nekoliko načina da se riješi ovaj problem:

- Isključivanje integralnog dijela sve dok ulazna vrijednost nije u parametrima koji se mogu kontrolisati
- Limitiranje maksimalne vrijednosti integralnog dijela
- Preračunavanje integralnog dijela kako bi ostao u prethodno određenim granicama

8.6.2 Deadband

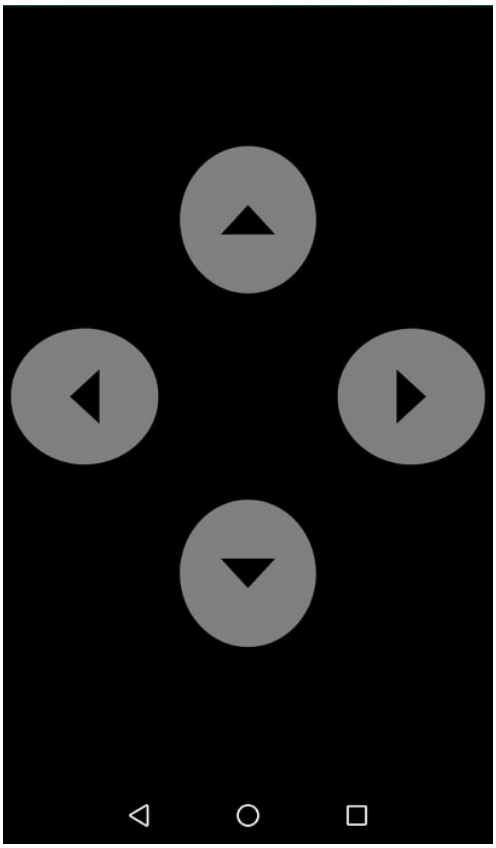
Modifikacija PID kontrolera koja se naziva dedadband znači određivanje zone PID izlaza za koji se isti pretvara u nulu. Ovo se koristi kod sistema koji ne trebaju da reaguju na jako male promjene, tj. kada u sistemu dolazi do kvarova nastalih trenjem.

9 Android aplikacija

U ovom dijelu će biti opisana Android aplikacija, koja je dio samog projekta samobalansirajućeg robota.

9.1 Dizajn

Pošto će se robot moći kretati u 2 smjera, i rotirati, također u 2 smjera, mobilnu aplikaciju će činiti 4 dugmeta, smještena na sredini ekrana.



Slika 34: Dizajn mobilne aplikacije u Adobe XD

Dugmići inicijalno imaju postavljenu providnost na 50%, a 100% samo dok su pritisnuti.

9.1.1 GridLayout

Kako bi dizajn bio responsivan i proporcije ostale jednake na većim i manjim ekranima, korišten je GridLayout. Kada se širina layout-a postavi na `match_parent`, mreža će zauzeti širinu uređaja. U ovom slučaju nam je potrebno 3 kolone i 3 reda

Kolona 0, Red 0	Kolona 1, Red 0	Kolona 2, Red 0
Kolona 0, Red 1	Kolona 1, Red 1	Kolona 2, Red 1
Kolona 0, Red 2	Kolona 1, Red 2	Kolona 2, Red 2

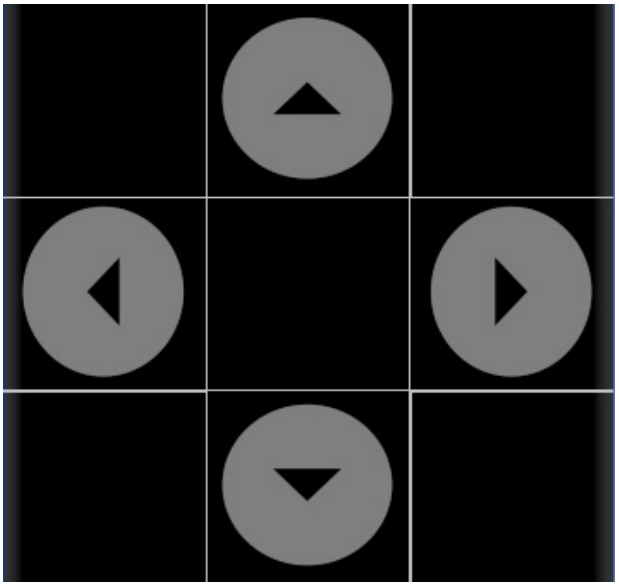
Tabela 9: Izgled 3x3 GridLayout-a

```
<GridLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:columnCount="3"
android:rowCount="3">
<!-- Sadržaj GridLayout-a -->
</GridLayout>
```

Primjer koda 2: GridLayout

9.1.2 ImageButton

ImageButton je zapravo obično dugme, kojeg umjesto okvira i teksta u njemu, vizuelno predstavlja slika. Kako bi napravio dizajnirani izgled, u odgovarajuće kolone i redove sam postavio ImageButton-e koji će služiti kao dugmad za upravljanje robotom.



Slika 35: ImageButton-i unutar GridLayout-a

```
        <ImageButton
        android:id="@+id/Right"
        android:layout_row="1"
        android:layout_column="2"
        android:rotation="-90"
        style="@style/DirectionButton"/>
```

Primjer koda 3: ImageButton za dugme "Desno"

Stil za ImageButton je definisan u `style.xml` datoteci. Rotacije slike se može postići postavljanjem atributa rotation iz imenskog prostora "android". Transparentnost se postavlja atributom alpha, također iz imenskog prostora "android". `"Res/drawable"` je putanja direktorija u kojem se nalaze datoteke slika, koje se zatim mogu referencirati iz xml datoteke i postaviti kao slika ImageButtona, background atributom. Atribut gravity postavlja dugme u središte ćelije GirdLayout-a.

```
    <style name="DirectionButton"
    ↪  parent="Widget.AppCompat.ImageButton">
        <item name="android:textColor">#00FF00</item>
        <item name="android:background">
        ↪  @drawable/ic_arrow_drop_down_circle_black_24dp</item>
        <item name="android:alpha">0.5</item>
        <item name="android:layout_width">0dp</item>
        <item name="android:layout_columnWeight">1</item>
        <item name="android:gravity">center</item>
    </style>
```

Primjer koda 4: Still ImageButton-a

9.2 Kod

Pri programiranju Android aplikacija, kod se može pisati u Java ili Kotlin programskom jeziku. Ja sam za izradu ove aplikacije odlučio koristiti Java jezik.

9.2.1 Bluetooth servis

S obzirom na to da će aplikacija sa robotom komunicirati putem bluetooth veze, potrebno je prvo uspostaviti konekciju,a zatim slati podatke putem iste. Kako bi se koristio bluetooth adapter uređaja, potrebno je,u AndroidManifest.xml datoteci, dodati opciju za zahtjevanje permisija pri prvom pokretanju aplikacije.

```
<uses-feature android:name="android.hardware.bluetooth" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission
↪  android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission
↪  android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
↪  android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Primjer koda 5: Zahtjevanje permisija za korištenje bluetooth-a

BluetoothAdapter je objekat koji se nalazi u statičkoj memoriji, u kodu ga je moguće dobiti koristeći `BluetoothAdapter.getDefaultAdapter()`. Adapter posjeduje funkciju `enable` koja će na uređaju upaliti bluetooth, ukoliko već nije upaljen.

BluetoothDevice je tip objekta koji sadrži informacije o uređaju sa kojim želimo komunicirati, objekat uređaja možemo dobiti iz BluetoothAdapter-a na osnovu njegove MAC adrese, koristeći `getRemote` funkciju.

BluetoothSocket je objekat koji predstavlja konekciju sa drugim uređajem. Pozivajući funkciju `createInsecureRfcommSocketToServiceRecord` nad objektom uređaja, kao rezultat ćemo dobiti BluetoothSocket objekat, nakon čega je potrebno pozvati `connect` funkciju na socket objektom kako bi se veza između uređaja mogla uspostaviti. Svaki socket ima dva stream-a podataka, input i output stream. Iz input streama je moguće čitati podatke koji dolaze na adapter, dok se u output stream upisuju podaci koje želimo poslati povezaonm uređaju.

ConnectionThread je klasa koju sam napravio kako bih razdvojio bluetooth funkcionalnosti od UI threada. Kako bih ovo uradio, bilo je potrebno da moja klasa naslijedi Thrad roditeljsku klasu. U konstruktoru kao parametar, ova klasa pirma BluetoothSocket, čiji

outputstream onda smješta u svoju privatnu varijablu. Funkcija kojom se šalju podaci, write, na osnovu String inputa, generiše niz byte-ova i koristeći `write` nad socket objektom, poruku šalje povezanom uređaju.

9.2.2 MainActivity

Aktivnost koja se pokreće pri praljenju aplikacije se naziva "MainActivity". Obzirom da je Java objekto orijentiran programski jezik, aktivnost je javna klasa koja nasljeđuje "AppCompatActivity" klasu.

```
    private char _currentSignal;
    private BluetoothService _bluetoothSerivce;

    public void SendSignal (char Direction){
        if(_currentSignal==Direction)
            return;

        _bluetoothSerivce.Send(Character.toString((Direction)));
        _currentSignal = Direction;
    }
```

Primjer koda 6: SendSignal funkcija u MainActivity klasi

onCreate je funkcija za koju se očekuje da će biti override-ana od strane programera. Ona se poziva svakiu put kada se aktivnost kreira. Prvo što se treba uraditi unutar onCreate funkcije jeste odabir layout-a koji koristi ova aktivnost pomoću setContentView funkcije.

findViewById(id) je funkcija koja vraća view objekat na osnovu njegovog id-a koji je postavljen unutar `xml` datoteke.

Eventi su objekti koji nastaju kada se desi promjena na korisničkom sučelju. Eventima se mogu dodijeliti funkcije koje će se izvršiti kada se event dogodi. Ovi tipovi objekata se najčešće koriste zajedno sa eventListener objektima kojima se odabranim elementima korisničkog sučelja, pridodaju pojedini eventi. Kada se aktivira listener, unutar funkcije se može pristupiti dva objekta, view sa kojim je izvršena interakcija, i tip eventa koji se dogodio. `setOnTouchListener` kao tip eventa može imati `MotionEvent.ACTION_UP`, što označava da dugme više nije u pritisnutom stanju.

```
List<ImageButton> Buttons = new ArrayList<>();
Buttons.add((ImageButton) findViewById(R.id.Left));
Buttons.add((ImageButton) findViewById(R.id.Up));
Buttons.add((ImageButton) findViewById(R.id.Down));
Buttons.add((ImageButton) findViewById(R.id.Right));

for (ImageButton button : Buttons)
button.setOnTouchListener(new View.OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {

                //Dobavljanje objekta koji je pozvao akciju
                ImageButton imgBtn = (ImageButton) v;
                //Stavljanje vidljivosti buttona na 100%
                imgBtn.setAlpha((float) 1.0);

                //Dobavljanje id stringa buttona
                String id =
                ↪  getResources().getResourceName(v.getId());
                int idLocation = id.indexOf('/');
                id = id.substring(idLocation +1,idLocation+2);

                //Slanje signala pomoću bluetootha
                SendSignal(id.charAt(0));

                //Da li je dugme prestalo biti pritisnut?
                if(event.getAction() == MotionEvent.ACTION_UP) {
                        //Varćanje vidljivosti na 50%
                        imgBtn.setAlpha((float) 0.5);
                        //Slanje signala za zaustavljanje
                        SendSignal('S');

                    } return true;}});
```

Primjer koda 7: Dodjeljivanje event objekata ImageButtonima

10 Robot

Upoznavši se sa tehnologijama, komponentama, detaljnim opisom uređaja i PID kontrolerom u dosadašnjem radu, poznato je sve što je potrebno za pravljenje samobalansirajućeg robota.

10.1 Konstrukcija

Konstrukciju robota čine 3 pločice pleksiglasa koje su namontirane na 4 navojne šike. Ispod same konstrukcije se nalaze 2 koračna motora sa točkovima. Pločice su raspoređene u spratove. Na prvome se nalazi matador ploča sa komponentama, na drugom Arduino Mega kontroler i na trećem spratu se ne nalazi ništa, već je tu zbog visine i težine robota.

10.1.1 Navojne šipke

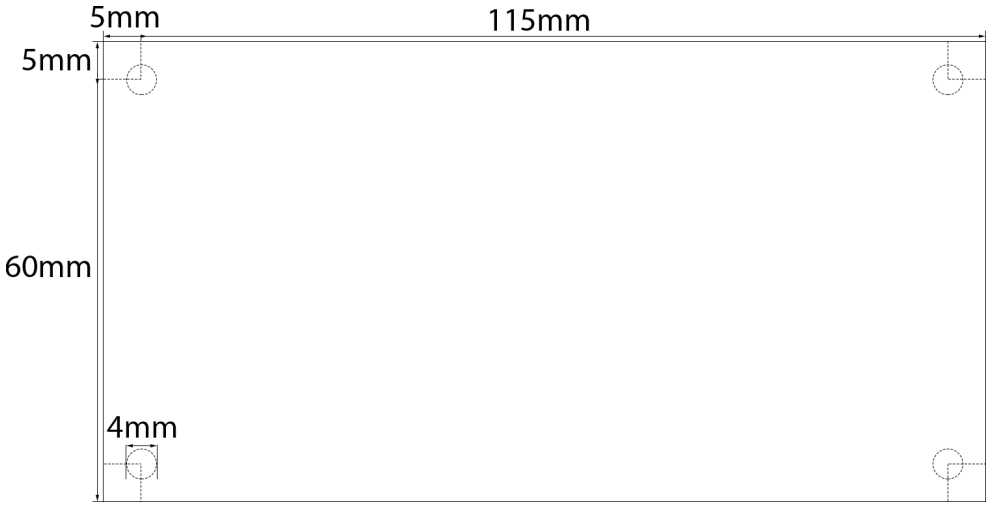
Navojne šipke su podrška čitavog robota. Četiri šipke dužine po 21cm daju robotu visinu koja ga dovodi u stanje disbalansa. Debljina svake šipke iznosi 4mm. U dužini šipke može postojati greška jer su rezane ručno.

Slika 36: Navojna šipka

10.1.2 Pleksiglas

Pločice pleksiglasa na gornja dva sprata robota su debljine 3mm, dok je na prvom spratu debljina pločice 5mm kako bi mogla podnijeti težinu motora i točkova. Sve tri pločice imaju četiri rupe širine 4mm na svojim krajevima, sa namjenom da se kroz njih provuku navojne šipke. Ono što ploče drži u mjestu i onemogućava im klizanje niz navojne šipke, osim samog trenja, su matice, postavljene ispod i iznad svake rupe u pločici (ukupno 24

matice). Pored toga, na pločici koja se nalazi na prvom spratu, su također zabušene rupe veličine 4mm kroz koje su na pločicu pričvršćeni L-profili, koji sa svoje druge strane imaju pričvršćene koračne motore. Srednji sprat na sebi ima zabušene četiri rupe veličine 3mm kroz koje su provučeni šarafi koji drže Arduino MEGA pločicu pričvršćenom na mjestu.



Slika 37: Nacrt pleksiglas pločica sa rupama za navojne šipke

10.1.3 Točkovi

Točkovi koji se nalaze na koračnim motorima su plastični. Težina jednog točka je oko je oko 103g, a prečnik iznosi 144mm. Širina točka iznosi 45mm. Točkovi nisu savršeno proporcionalni, jer im svrha nije kontrolisati delikatnog robota, već se radi o pomoćnim točkovima za dječiji bicikl. Na motore su fiksirani koristeći dvokomponentno ljepilo. Svojim prečnikom, robotu daju dodatnih 75mm visine, što ga dovodi na ukupnih 28.5cm. Početno sam pokušao koristiti druge točkove, prečnika 7cm i težine 150g, ali zbog njihove težine motor nije mogao da se okreće na većim brzinama.

10.2 Komponente

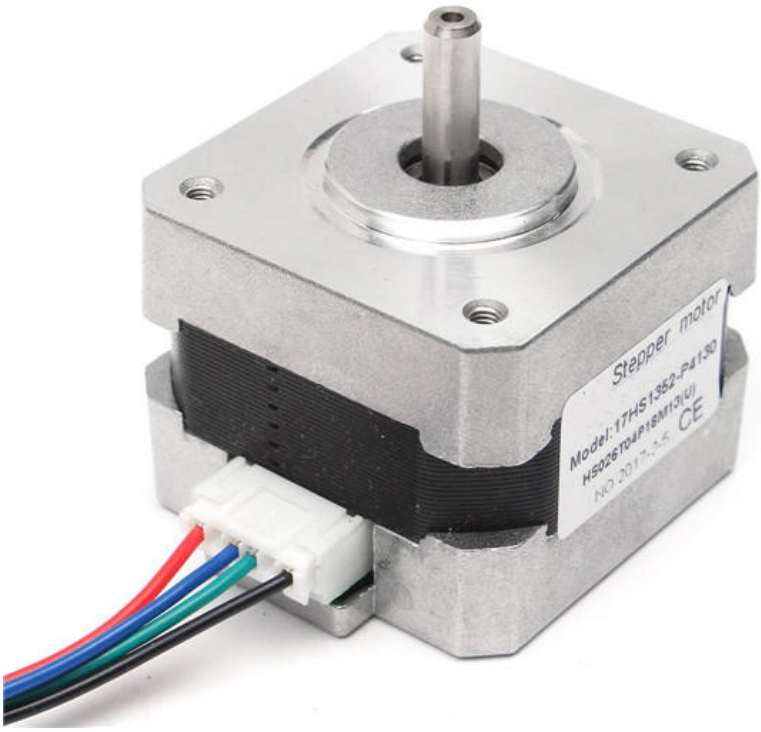
10.2.1 HC-05

Za ostvarivanje veze sa android aplikacijom korišten je HC-05 bluetooth modul (sekcija 4). Pomoću AT komandi (sekcija 4.4), ime modula je postavljeno na "RobotBT", a lozinka na 1511. Također sam koristeći AT komande saznao MAC adresu modula, te je unio kao predefinisanu pri dobijanju BluetoothDevicce objekta u Android aplikaciji (sekcija 9.2.1). Pošto je na modulu uključena opcija automatskog konektovanja na posljednji uređaj, dok lozinka osigurava da je uređaj koji je zadnji povezan Android a aplikacijom za upravljanje,

pri paljenju robota se Android aplikacija i HC-05 povezuju automatski. Brzina prenosa podataka je 9600 bitova po sekundi. HC-05 je spojen na RX1 i TX1 pinove (sekcija 3.7).

10.2.2 Koračni motori

Koračni motori koje sam izabrao za ovaj projekt su Nema 17 motori. Dimenzije ovih motora su 43.18mm x 43.18mm, što je 1.7 inch, odakle i potiče broj 17 u njiovom nazivu. Težina jednog motora je oko 250g. Na poleđini imaju 4 šarafa od kojih su 2 iskorištena kako bi se pričvrstili L-profili.



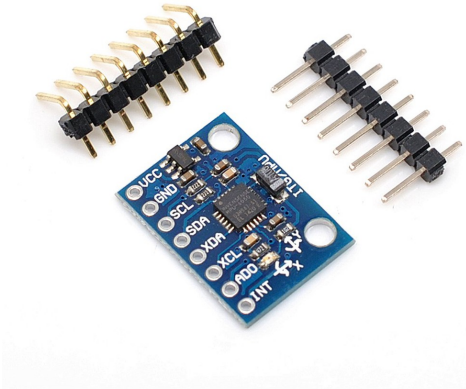
Slika 38: Nema 17 koračni motor

10.2.3 A4988

Za projekt su korištena dva A4988 upravljača(sekcija 6). U ovom slučaju nije bilo moguće koristiti samo jedan zato što postoje situacije kada oba točka ne rade istom brzinom, ili u istom smijeru. Upravljač je postavljen da radi u 1/16 koračnom modu(sekcija 6.4), tako što je na MS1, MS2 i MS3 puštena voltaža visokog nivoa. Pinovi na Arduinu na koje su spojeni DIR i STEP (sekcija 6.4) pinovi su 51, 52, 53 i 54 digitalni pinovi (sekcija 3.7).

10.3 MPU6050

Kako bi robot znao svoju trenutnu orijentaciju u odnosu na zemlju, ugrađen mu je MPU6050 (sekcija 7). Kako bi se uređaj mogao povezati, bilo mu je prvo potrebno zalemiti pinove jer u paketu dođe sa 2 vrste pinova.



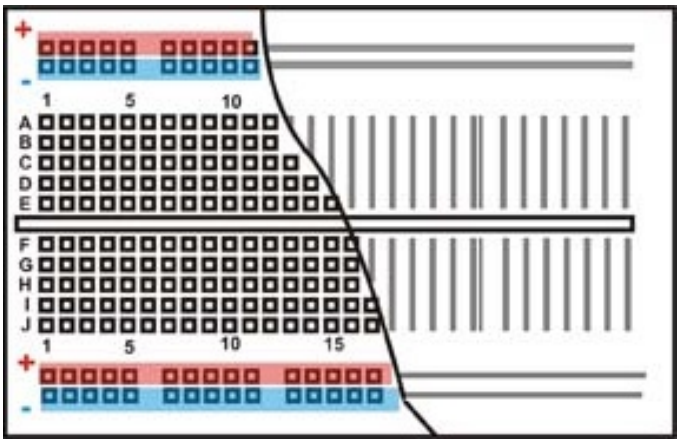
Slika 39: Vrste pinova koje dođu uz MPU6050

Pinovi SDA i SCL (sekcija 7.5.1) su spojeni na Arduino SDA i SCL (sekcija 3.7) pinove koji podržavaju I^2C protokol (sekcija 7.3.2). INT pin je spojen na Arduino pin 2 koji također podržava signale prekida.

10.4 Matador ploča

Matador ploča se koristi kao baza za prototip u elektronici. Razlog zbog kojeg se matador ploče koriste je to što se elektroničke komponente mogu povezati bez ikakvog lemljenja. Ploča se sastoji iz terminala koji se nalaze u sredini, i sabirnice koja se nalazi sa vanjske strane ploče. Terminali se koriste za ožičenje komponenti, dok se sabirnice koriste za dovođenje struje u komponente. Kada na jedan terminal postavimo žicu, svaka druga žica koju postavimo u isti terminal će se ponašati kao da je zaljemljena za prvu žicu ili pin. Sabirnicu možemo prepoznati tako što ide dužinom ploče, dok terminali idu širinom. Uz matador ploču se najčešće koriste jumper kablovi, koji svojim pinovima ostvaruju kvalitetan kontakt sa unutrašnjim dijelom ploče. Zbog lakšeg kabliranja, koristio sam bužire na kablovima. Bužiri su izolacija za kablove koja se zagrijavanjem kontraktuje i u ovom slučaju su korisni kada postoji 2 ili više kablova koji idu u istom smjueru.

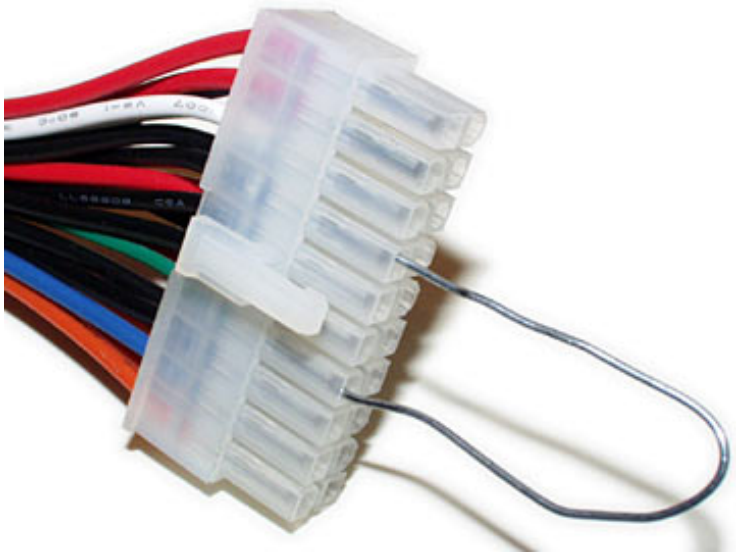
Ja sam u ovom projektu koristio matador ploču sa 60 terminala sa po 5 ulaza, i 4 sabirnice sa vanjskih strana. Dimenzije ploče su 85mm x 55mm.



Slika 40: Matador terminali i sabirnica

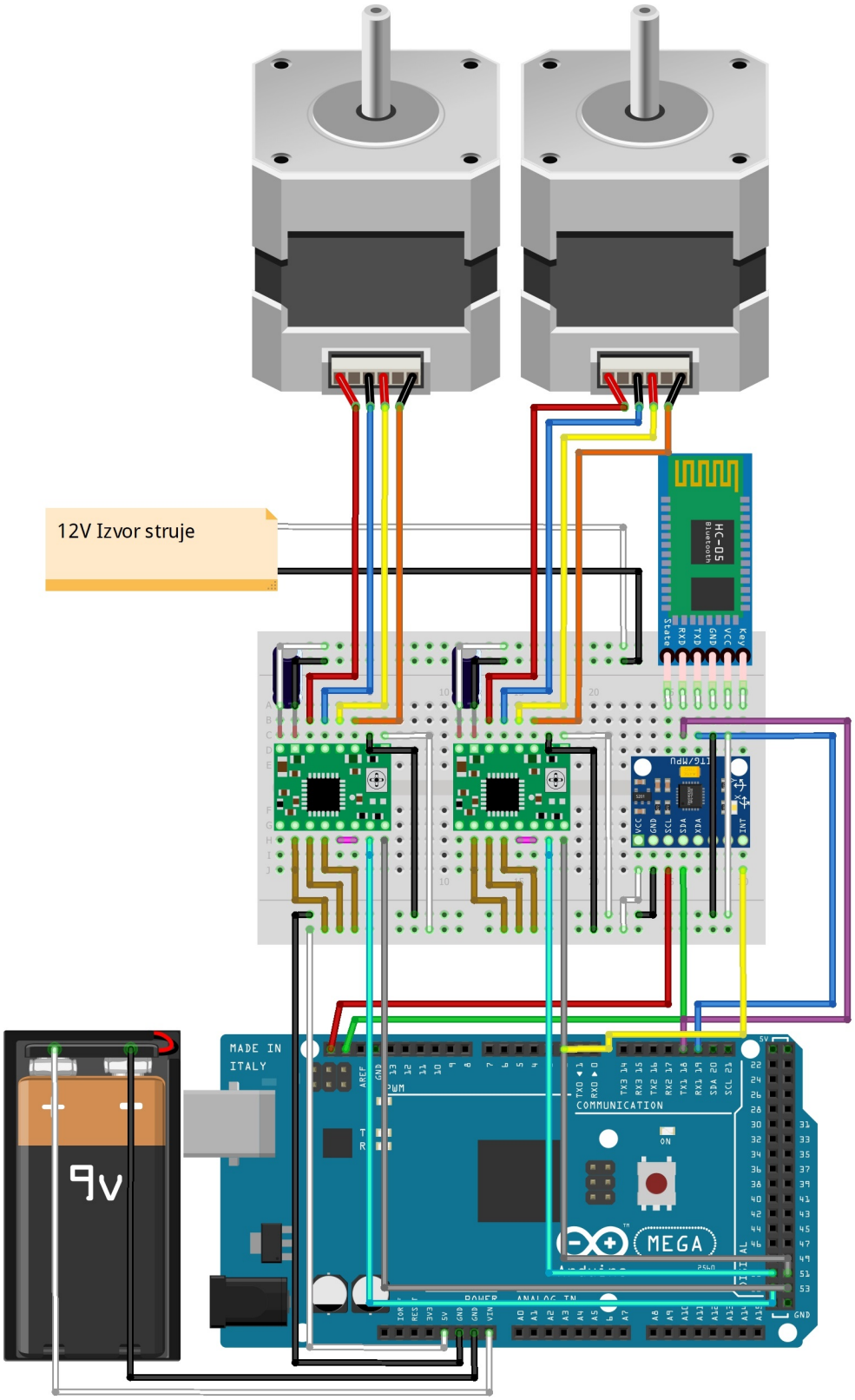
10.4.1 Napajanje

Pošto Arduino ima maksimalan izlaz struje od 5V, a za napajanje A4988 kojem je potrebno između 8V i 35V (Sekcija 6.4) kako bi pokrenuo motore, potrebno je dovesti vanjski izvor struje koji može dostaviti potrebnu voltažu. Pošto nisam bio u mogućnosti na vrijeme doći do baterije koja bi mogla dovesti zahtjevanu struju, iskoristio sam staru napojnu jedinicu iz računara. Način na koji matična ploča pali napojnu jedinicu jeste taj da jedinoj zelenoj žici otvori put do uzemljenja čime napojna jedinica dobija signal da je upaljena. Nakon toga, kroz svaku narandžastu žicu prolazi 3V, crvenu 5V i žutu 12V, dok je svaka crna žica uzemljenje. Ovo sam iskoristio tako što sam na žutu i crnu žicu zalemio bakreni izolirani kabal dužine 3m, te na njegove krajeve zalemio muške jumper kablove koji dovode struju u matador ploču.



Slika 41: Spojena crna i zelena žica kako bi se napojna jedinica upalila

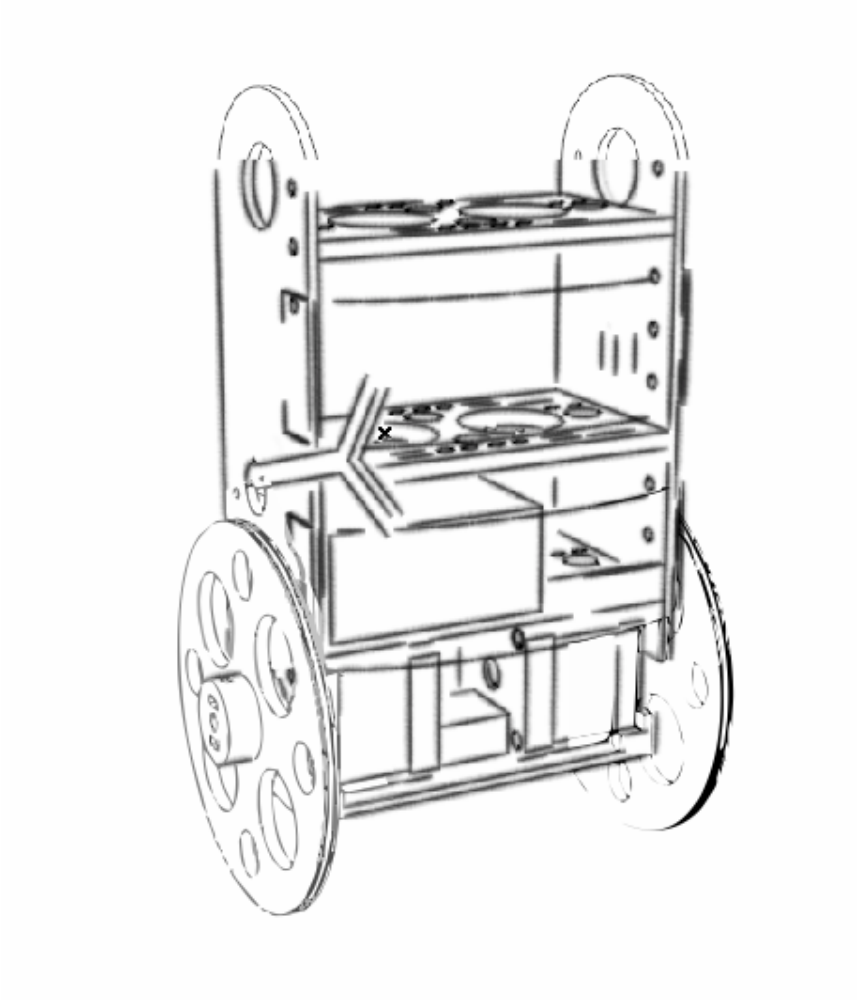
10.5 Kabliranje



Slika 42: Šema kabliranja robota

10.6 Proces balansiranja

U procesu balansiranja robota će biti ključan MPU6050. Očitavanjem njegovih ulaznih podataka o trenutnom položaju robota u prostoru i prosljeđivanjem tih podataka PID kontroleru (sekcija 8), generisat će se izlaz koji će biti proporcionalan brzini motora. Tačnije, kada konstrukcija robota počne padati prema naprijed, PID kontroler će na osnovu podataka iz DMP-a (sekcija 7.3) poslati signali A4988 upravljaču da kretnjom motora pomjeri robot prema naprijed što će težište motora prilagoditi poziciji njegovog gornjeg dijela, te robota zadržati u balansiranom stanju.



Slika 43: Skica sličnog robota

Pravilo podešene K_p, K_i i K_d (sekcija 8.5), omogućit će da se robot i nakon blagog guranja vrati u balansirano stanje. Promjenom ciljane vrijednosti, robot će se kretati naprijed i nazad.

10.7 Arduino program

Ključni dio projekta je sami Arduino program koji će kontrolisati robota.

10.7.1 Biblioteke

Za izradu programa, korištene su funkcije 5 biblioteka.

PID [6]

Ova biblioteka nudi jednostavne funkcije za implementaciju PID kontrolera.

- PID(&y, &u, &r, Kp, Ki, Kd) - konstruktor koj uzime adrese inicijaliziranih varijabli za ulaz, izlaz i ciljanu vrijednost, te parametre za podešavanje
- SetOutputLimits(m,n) - postavlja limite izlazne vrijednosti kako bi se izlaz skalirao
- SetSampleTime(t) - postavlja vremensku varijablu za kontroler u milisekundama
- Compute() - vrši PID kalkulaciju i mijenja vrijednost izlazne varijable

TimerOne [7]

Pošto će u glavnom programu biti potrebno određene događaje izvršavati na osnovu vremenskog parametra, uključena je ova biblioteka koja omogućava podešavanje prekida nakon zadanog vremena i dodjele funkcije koja će se izvršiti na tom prekidu.

- Initialize(t) - postavlja vrijeme koje će proći između izvršenja prekida
- attachInterrupt(void) - zadaje funkciju koja će se izvršiti svakim prekidom.

I2Cdevlib

Ponuđene funkcije ove biblioteke su već opisane u sekciji 7.6.

10.7.2 Motors biblioteka

Prvobitno sam napisao kompletnu biblioteku za akceleraciju i deakceleraciju stepper motora, kako bi se robot fluidnije kretao, ali sam kasnije došao do zaključka da će se akceleracija prirodno dogoditi, jer naginjanjem robota PID izlaz postepeno ubrzava motore ka većim brzinama. Ipak mi je jedan dio te biblioteke poslužio i u finalnom projektu.

[6] https://github.com/br3ttb/Arduino-PID-Library
[7] https://www.arduinolibraries.info/libraries/timer-one

- connectToPins(S,D) - u privatnu varijablu zabilježava pinove ze STEP i DIR na upravljaču motora
- setDirection(D) - Postavlja smjer rotacije motora na osnovu enumeratora
- makeStep() - Radi 1 korak.

```
void Motor::makeStep()
{
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(2);
    digitalWrite(stepPin, LOW);
}

void Motor::setDirection(Direction Direction)
{
    if (Direction == Clockwise)
    digitalWrite(directionPin, HIGH);

    if (Direction == CounterClockwise)
    digitalWrite(directionPin, LOW);
}

void Motor::connectToPins(byte stepPinNumber, byte
↪ directionPinNumber)
{
    stepPin = stepPinNumber;
    directionPin = directionPinNumber;

    pinMode(stepPin, OUTPUT);
    digitalWrite(stepPin, LOW);

    pinMode(directionPin, OUTPUT);
    digitalWrite(directionPin, LOW);
}
```

Primjer koda 8: Funkcije motor klase

10.7.3 Tok programa

Program se može rastaviti na 3 glavna dijela:

1. setup()
2. loop()
3. prekid

Setup()

Unutar setup funkcije se inicijaliziraju uređaji.

1. Inicijalizacija motora
2. Otvaranje HC-05 serijske komunikacije
3. Postavljanje offseta za moju MPU6050 jedinicu
4. Kalibracija MPU6050
5. Preuzimanje veličine FIFO paketa od MPU6050
6. Inicijalizacija vremenskog prekida

Loop()

Unutar loop funkcije se čeka prekid izazvan od strane MPU6050, te se preuzimaju njegovi paketi i vrši PID kalkulacija

1. Ukoliko je broj paketa u FIFO manji od veličine paketa vraća se na početak loop-a
2. Ukoliko je sa HC-05 došao signal, postavlja se u buffer varijablu
3. Varijabla koja označava MPU6050 prekid se postavlja na false
4. Učitavaju se podaci iz FIFO buffera
5. Podaci iz DMP-a se konvertuju u kvaternion, a zatim u eulerove uglove
6. Ugao X ose se postavlja kao ulazna vrijednost za PID kontroler
7. Radi se izračunavanje PID izlaza
8. Postavlja se smjer motora
9. Ukoliko je u bufferu neka od komandi za kretanje, aplicira se njen pomak u PID izlaz
10. Postavlja se vrijeme između koraka za oba motora (brzina motora)

Prekid()

Prekid koji se izvršava na osnovu unutrašnjeg tajmera za zadatak ima pokretanje motora.

1. Računanje trenutnog vremena
2. Računanje vremena od posljednjeg koraka
3. Poređenje vremena sa PID izlazom
4. Ukoliko je vrijeme za sljedeći korak
   (a) Napravi korak
   (b) Postavljanje vremena posljednjeg koraka za motor na trenutno vrijeme

11 Zakljucak

Prilikom pisanja diplomskog rada, imao sam priliku da prođem kroz iznimno zanimljiv i poučan proces koji mi je omogućio da bolje razumijem određene aspekte elektronike, kao i njenu suštinu. Vremenski raspon koji mi je bio potreban za kompletiranje ovog rada je trajao 20 dana. Smatram da je jedna od najvećih prednosti ovog neobičnog projekta mogućnost nadogradnje u budućnosti i, samim tim proširivanje znanja iz navedene oblasti.

Obzirom na to da je robot funkcionalan, zaključio bih da je ovaj rad završen uspješno.Ukoliko ste zainteresovani da temeljitije upoznate korištene metode, čitav projekat, kao i konkretan kod koji je korišten u istom moguće je pronaći na mom GitHub akauntu [8].
Kao što je već navedeno, u budućnosti planiram implementirati nadogradnje na robota, a neke od potencijalnih su:

- Prebacivanje komponenata sa matador-a na PCB
- Izrada kvalitetnije i šire konstrukcije robota
- Unaprijeđenje PID kontroloera
- Zamjena točkova
- Korištenje DC motora
- Instaliranje baterije koja će napajati Arduino i motore

[8] https://github.com/Adi-Sose/FALL-E