SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Marko Car, Antun Ivanović

Prošireno korisničko sučelje za upravljanje robotskim manipulatorom u letu

30. travnja 2014.

Ovaj rad izrađen je u Laboratoriju za Robotiku i Inteligentne Sustave Upravljanja, na Zavodu za Automatiku i Računalno Inženjerstvo pod vodstvom prof. dr. sc. Stjepana Bogdana i predan je za natječaj za dodjelu Rektorove nagrade u akademskoj godini 2013/2014.

Sadržaj

Popis slika

Popis tablica

1. Uvod

Robotika je višedisciplinarna znanstvena grana koja objedinjuje znanja iz područja mehanike, elektronike, računarstva i automatike [1]. Najčešće se pod pojmom "robota" podrazumijeva industrijski robot koji se još naziva robotski manipulator ili robotska ruka. U takvom, klasičnom poimanju robota, isti se modelira u obliku lanca krutih članaka, koji su međusobno povezani pokretljivim zglobovima, a pričvršćeni su na nepomično postolje. Razvojem mobilne robotike, robotski manipulatori dobili su novu dimenziju pokretljivosti, koja im je omogućila pokretljivost u dvodimenzionalnom prostoru. Ovaj rad nastoji doprinijeti novom, diskretnom skoku u istraživanju robotskim manipulatorima, omogućujući im novu dimenziju prostornog kretanja integrirajući ih u bespilotne letjelice.

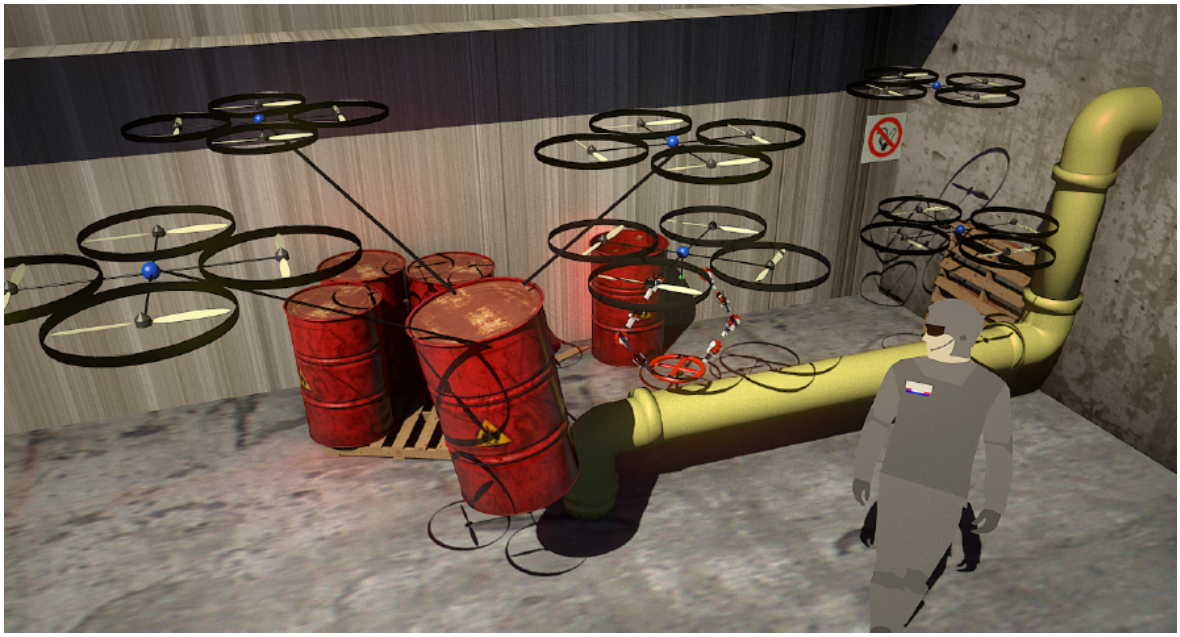
Značenje robotike u modernom društu je iz dana u dan sve veće zbog ubrzanog napretka znanosti i tehnologije. Poseban fokus istraživanja u robotici stavljen je na umjetne spoznajne sustave i autonomne robote koji će biti sposobni za rad u dinamičkim i nedeterminističkim okruženjima [3]. Istraživački programi imaju za cilj razvoj autonomnih sustava za rad u opasnom i nepristupačnom okruženju na misijama pretraživanja i spašavanja, borbe protiv požara i nadgledavanju granica. U takvim se misijama vrlo često koriste zračni robotski sustavi, odnosno bespilotne letjelice. Zbog ograničene veličine bespilotnih letjelica, i ograničene nosivosti, takvi su robotski sustavi, uglavnom usredotočeni na izgradnju mapa prostora, lokalizaciju cilja i navigiranje robotskih agenata na zemlji. Unatrag posljednjih dvadeset godina, istraživanje bespilotnih letjelica ograničeno je na izbjegavanje interakcije s okolinom, no danas se sve češće provode istraživanja koja uključuju manipulatore. Razvojem bespilotnih letjelice pomiču se granice njihove nosivosti, te se omogućuje da se priključivanjem robotskih manipulatora na konstrukciju letjelice pomaknu granice njihovih mogućnosti.

Često se kod robota primjećuje antropomorfna građa, pa nerjetko i sami zlgobovi robota odgovaraju ljudskim zglobovima, poput ramena, lakta, struka itd. Odabrana konstrukcija robotskih manipulatora izrađena je tako da sadrži dva manipulatora sa zglobovima ramještenim po uzoru na ljudsku ruku.

Cilj ovog rada uključuje dizajn i implementaciju proširenog korisničkog sučelja pomoću kojeg će jedna osoba moći upravljati bespilotnom letjelicom i na njoj pričvršćenim robotskim manipulatorom. Osmišljeno je višekanalno upravljanje letjelicom putem glasovnih naredba i bežićne igraće palice te upravljanje dvostrukim robotskim manipulatorom putem Kinecta. Istraživanja pokazuju kako upravljanje glasom daje dobre rezultate u vidu brzine i točnosti izvođenja zadatka [2]. Također, glasovno upravljanje omogućuje korištenje svih funkcija letjelice, a da su, pri tome, ruke slobodne te se pogled ne miče s područja u kojem upravljamo letjelicom. Veza između operatera i robotskih manipulatora postignuta je praćenjem pokreta tijela operatera koristeći Kinect te transformacijom tih pokreta u upravljačke veličine robotskog manipulatora, čime je omogućeno prirodno i intuitivno upravljanje robotskim manipulatorom. Za razvoj aplikacija korištena je razvojno okruženje ROS.

Provedena su ispitivanja proširenog upravljačkog sučelja na problemu otvaranja/zatvaranja ventila za što je osmišljen scenarij zadatka u kojem operater mora uzletjeti bespilotnom letjelicom, sletjeti na ventil te uz pomoć dvostrukog robotskog manipulatora okrenuti ventil, a na kraju misije sigurno uzletjeti s ventila i prizemljiti letjelicu. Izrađene su dvije bespilotne letjelice: bespilotna letjelica s dvostrukim robotskim manipulatorom i letjelica s kamerom; te je izrađen ventil. U cilju sigurnog

testiranja regulatora i proširenog korisničkog sučelja, razvijeno je virtualno okruženje u Gazebo ROS simulatoru. Korišteni su već postojeći paketi za simulaciju dinamike tijela bespilotne letjelice, poremećaje uzrokovane vjetrom i aerodinamike propelera. Na taj se način u realnim uvjetima može očekivati slično ponašanje kao u simulatoru. Nakon testiranja u simulatoru provedena su ispitivanja na stvarnom sustavu.



2. Opći i specifični ciljevi rada

U početku razvoja računala interakcija između čovjeka i računala podrazumijevala je korištenje tipkovnice i miša. Razvojem tehnologije ta interakcija se proširila na razna područja poput igraćih palica, igraćih volana itd. Daljnjim razvojem tehnologije omogućena je intuitivnija komunikacija između čovjeka i stroja putem pokreta i gesti te glasovnim naredbama.

Igraće palice predstavljaju skupinu ulaza u sustav čiji je princip rada poznat većini korisnika, a ukoliko nije, zbog jednostavnosti izvedbe, veoma se brzo može naučiti. Većina igraćih palica ima dva tipa ulaza: analogni i digitalni. Analogni ulazi mogu poprimiti mnogo vrijednosti što omogućava doziranje željene razine upravljačke veličine. Digitalni ulazi mogu poprimiti samo dvije vrijednosti, jesu li uključeni ili ne. Igraća palica odabrana je jer je njome lako upravljati letjelicom te istovremeno mijenjati postavke leta.

Prepoznavanje glasa (eng. *voice recognition*) omogućuje interakciju između čovjeka i stroja na čovjeku poznatom terenu. Svakodnevna komunikacija s drugim ljudima uvelike doprinosi intuitivnosti ovakve interakcije. Operater za interakciju mora naučiti naredbe, odnosno ključne riječi, kako bi na pravilan način mogao ostvariti željeno vladanje sustava. Pritom naredbe moraju biti opisne kako ne bi došlo do konflikata u komunikaciji. Također, skup naredbi koji se koristi ne smije zadavati veće probleme kod prilagodbe jer se time povećava kompleksnost interakcije. Osim toga, kod odabira skupa naredbi cilj je i brzo savladavanje i učenje tog skupa.

Prepoznavanje položaja ljudskog tijela u prostoru operateru pruža intuitivan i jednostavan način komunikacije sa strojem. Raznim pokretima i gestama, poput onih kojima se služi u svakodnevnom govoru, mogu se zadati upravljačke naredbe stroju. Ovakva interakcija, kao i prepoznavanje govora, ne smije imati kompleksne pokrete i geste koje se prepoznaju. Također, skup naredbi mora se lako naučiti te biti što intuitivniji kako bi se izbjegli konflikti pri upravljanju.

U ovom radu cilj je spojiti igraću palicu, prepoznavanje govora te pokrete i geste u jedno upravljačko sučelje. Pritom se iz svakog područja uzimaju prikladni dijelovi te se njihove funkcije koriste u proširenom korisničkom sučelju. U eksperimentalnoj provjeri otvaranja/zatvaranja ventila koriste se sve tri komponente sučelja. Glasovnim upravljanjem odabiru se letjelice te se zadaju naredbe za uzlijetanje te slijetanje. Igraćom palicom upravlja se letjelicom dok se letjelica ne pozicionira iznad ventila. Pokretima ruku upravlja se dvostrukim robotskim manipulatorom pri čemu se trenutno stanje ruku operatera preslikava na manipulator. Ovaj način spajanja omogućava jednostavno i intuitivno upravljanje pojedinim dijelovima sustava čime se smanjuje razina stresa operatera, a samim time se povećavaju izgledi za uspješno izvršenje zadatka.

3. Materijali

U ovom poglavlju opisani su korišteni uređaji za komunikaciju i upravljanje.

3.1. Robot Operating System (ROS)

ROS (eng. *Robot Operating System*) je fleksibilna razvojna cjelina (eng. *framework*) koja se koristi za razvoj robotskih aplikacija. ROS je skup alata, paketa i konvencija kojima je cilj pojednostaviti izradu kompleksnih i robusnih robotskih aplikacija na različitim platformama. Zajedničko je djelo Sveučilišta u Stanfordu, MIT-a i njemačkog Tehničkog sveučilišta u Münchenu.

Glavna značajka ROS-a su gotove biblioteke za komunikaciju između pojedinih čvorova odnosno dijelova sustava. Protokoli komunikacije su skriveni od korisnika čime se pojednostavljuje njihovo korištenje pri izradi aplikacija zbog čega korisnik može veoma brzo i jednostavno povezati dijelove jednostavnih i kompliciranih sustava.

Za potrebe rada korištena je verzija ROS Hydro Medusa.

3.2. Turtle Beach Ear Force P11

Kako bi se moglo upravljati bespilotnom letjelicom putem glasovnih naredba potrebno je koristiti mikrofon. U radu su korištene slušalice s mikrofonom Turtle Beach Ear Force P11, prikazane na slici 1. Specifikacije slušalica dane su tablicom 1.

Tablica 1: Specifikacije slušalica Turtle Beach Ear Force P11 [34]

Specifikacije slušalica	
driver	$50mm$
osjetljivost	$120dB$
frekvencijski raspon	$20Hz - 20kHz$
dužina kabela	$3.7m$
specifikacije mikrofona	
frekvencijski raspon	$50Hz - 15kHz$

Slika 1: Slušalice Turtle Beach Ear Force P11

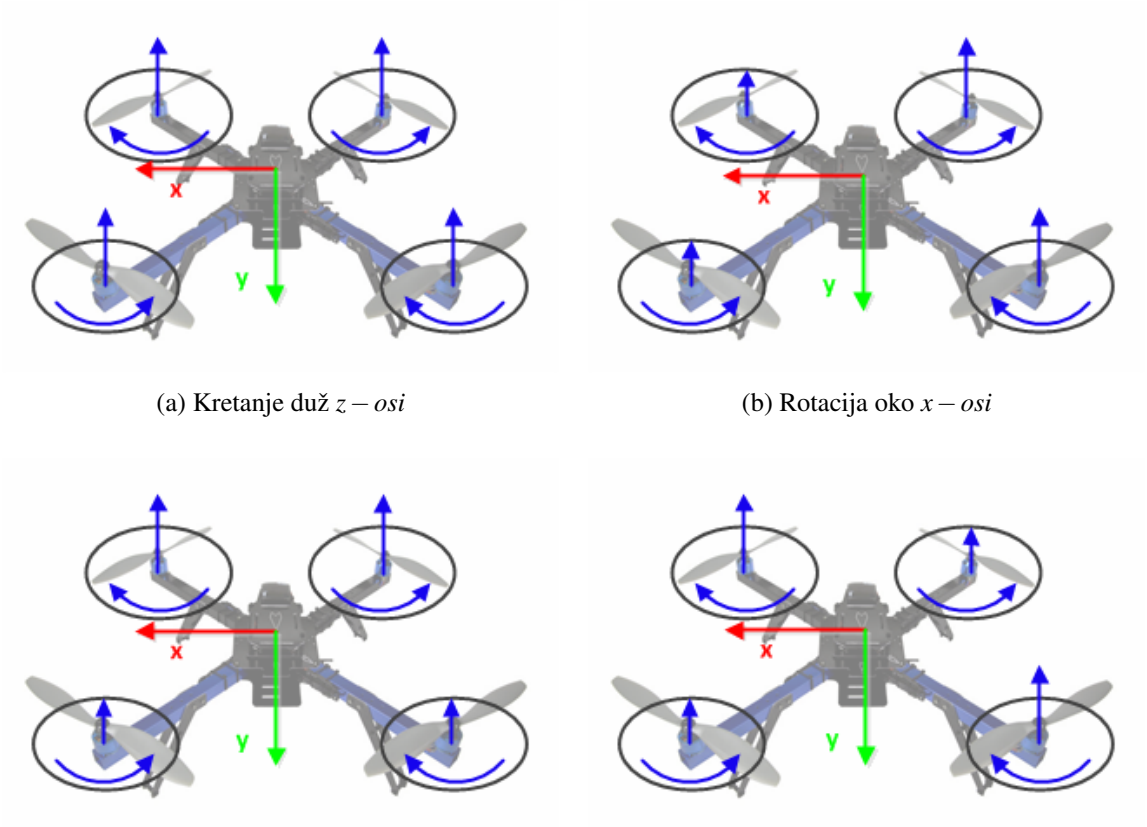
3.3. Bespilotna letjelica - Arducopter

Arducopter je bespilotna rotorska letjelica s četiri pogonska motora (eng. *Quadrotor*) prikazana slikom 2, tvrtke *3D Robotics*. U središtu križne konstrukcije nalazi se tijelo letjelice dok su motori s propelerima postavljeni na krajevima. Na tijelu letjelice nalazi se sva potrebna elektronika za upravljanje i komunikaciju te priključak za baterije. Tijelo letjelice može se nadograditi ukoliko postoji potreba za dodatnom elektronikom. Između tijela letjelice i motora nalaze se nožice kako se pri slijetanju ne bi oštetila letjelica.

Slika 2: Prikaz bespilotne letjelice Arducopter

3.3.1. Princip rada rotorske letjelice s četiri pogonska motora

Rotorska letjelica s četiri pogonska motora može se kretati na četiri različita načina: ubrzanjem duž $z-osi$ te rotacijom oko $x-osi$, $y-osi$ i $z-osi$ prema slici 3. Dijagonalni parovi rotora okreću se u istom smjeru, jedan par okreće se u smjeru kazaljke na satu, a drugi u smjeru suprotnom od kazaljke



(a) Kretanje duž $z-osi$

(b) Rotacija oko $x-osi$

(c) Rotacija oko $y-osi$

(d) Rotacija oko $z-osi$

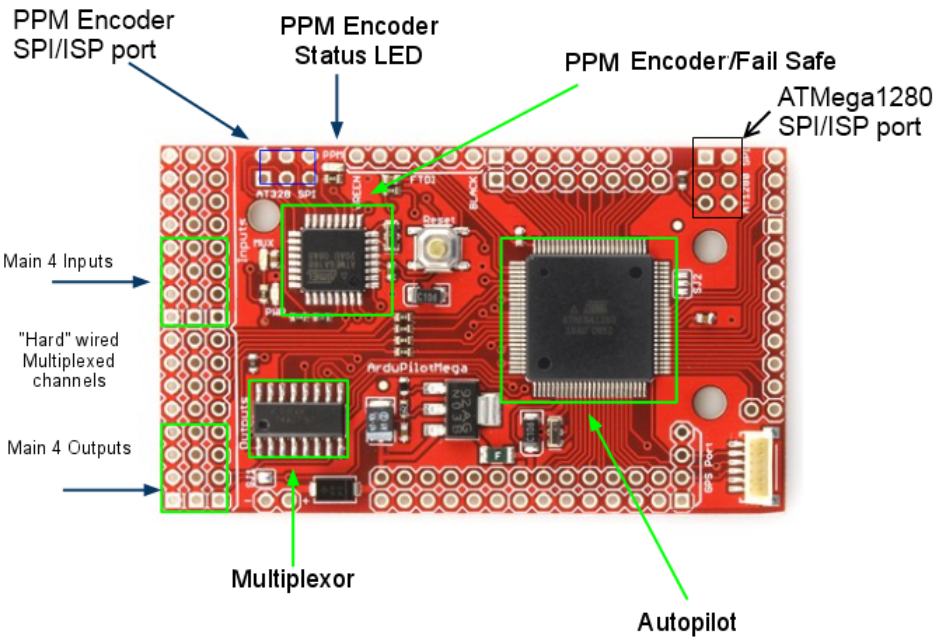
Slika 3: Prikaz osnovnih kretnji letjelice

na satu. Na taj način poništavaju se momenti koji bi u protivnom zakretali letjelicu oko $z-osi$.

Letjelica može biti konfigurirana na dva načina: **+** i **X** konfiguracija. Za kretanje duž $z-osi$ potrebno je povećati ili smanjiti brzine sva četiri motora, slika 3a. Kod zakretanja oko $z-osi$ jedan dijagonalni par motora se ubrzava dok se drugi usporava čime se može postići zakretanje letjelice u oba smjera, slika 3d. Kretanje duž $z-osi$ i zakretanje oko $z-osi$ ne ovise o konfiguraciji letjelice. Zakretanje oko osi x i y mijenja se promjenom konfiguracije. Kod **+** konfiguracije jedan se motor dijagonalnog para ubrzava dok se drugi usporava što za posljedicu ima zakretanje letjelice oko željene osi uz zadržavanje visine i orijentacije. Kod **X** konfiguracije upravljanje je nešto složenije jer je potrebno upravljati brzinama sva četiri motora. Zakretanje oko željene osi postiže se ubrzanjem dva motora koji nisu dijagonalni par te usporenjem preostala dva motora. Jednako kao kod **+** konfiguracije, u ovom se slučaju zadržava visina i orijentacije pri kretanju.

3.3.2. Elektronika letjelice Arducopter

Elektronika letjelice sastoji se od upravljačkog te IMU (*Inertial Measurement Unit*) dijela. Upravljački dio, APM (*ArduPilotMega*), prikazan je slikom 4.



Slika 4: Upravljačka pločica letjelice Arducopter

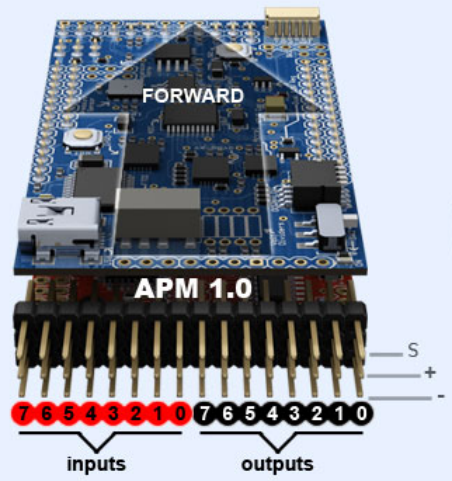
Verzija upravljačke pločice korištene u ovom radu je APM 1.0, [12]. Centralna procesorska jedinica pločice je tipa *ATmega2560* odnosno 8-bitni mikrokontroler tvrtke *Atmel*, [15]. Maksimalan radni takt mikrokontrolera iznosi *16MHz* uz *8KB* radne memorije te *4KB* EEPROM (*Electrically Erasable Programmable Read-Only Memory*) memorije. Na EEPROM memoriju zapisuju se postavke kalibracije letjelice. Također, 86 ulazno-izlaznih pin-ova čini ovaj mikrokontroler pogodnim za primanje, obradu i slanje velike količine podataka. Napajanje mikrokontrolera mora biti između $4.5V - 5.5V$ što se postiže ugrađenim regulatorom napona *LM317*.

APM pločica ima osam ulaza te isto toliko izlaza, slika 5. Pri tome gornji red pinova predstavlja ulazno-izlazne signale dok srednji i donji red napajanje od $5V$. Za spajanje RC (*Radio Control*) modula na pločicu koriste se ulazi $0-3$ čiji je opis dan tablicom 2. Dodatno se mogu koristiti ulazi 4 i 5 ako postoji potreba za zadržavanje pozicije putem GPS-a (*Global Positioning System*) odnosno mijenjanjem moda rada letjelice.

Drugi dio elektronike je već ranije spomenuta IMU pločica [14] odnosno *ArduPilotMega Shield*, slika 6, koja se spaja s APM pločicom. Podatci sa senzora pločice koriste se u upravljačkom dijelu te su nužni za pouzdan rad letjelice. Na samoj pločici nalaze se slijedeći senzori: senzor za mjerenje tlaka (eng. *pressure sensor*) koji služi za procjenu visine letjelice, dvoosni žiroskop za x i y osi, jednoosni žiroskop za $z-os$ i troosni akcelerometar. Također, koristi se i troosni magnetometar spojen

Tablica 2: Opis ulaznih pin-ova APM pločice [17]

Ulaz	Opis	Naziv (eng)
0	Kanal za referencu kuta valjanja (*roll*)	*Aileron*
1	Kanal za referencu kuta poniranja (*pitch*)	*Elevator*
2	Kanal za referencu gasa	*Throttle*
3	Kanal za referencu kuta zakretanja (*yaw*)	*Rudder*

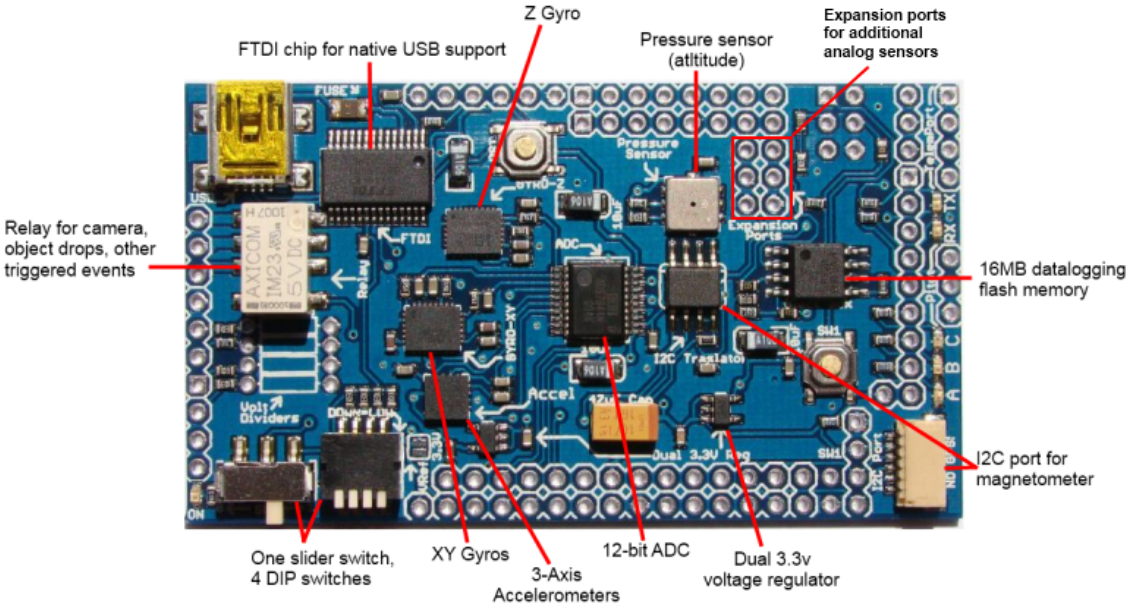


Slika 5: Ulazno-izlazni pin-ovi APM pločice

preko I2C ulaza. Pločica podržava ugradnju dodatnih senzora poput sonara, senzora za mjerenje brzine vjetra te kamere za računanje optičkog toka. Od ostalih komponenata na pločici se nalazi FTDI sklop za komunikaciju preko USB porta, 12-bitni analogno-digitalni pretvornik, regulator napona za generiranje napajanja od $3.3V$ te $16MB$ memorije za spremanje podataka sa senzora.

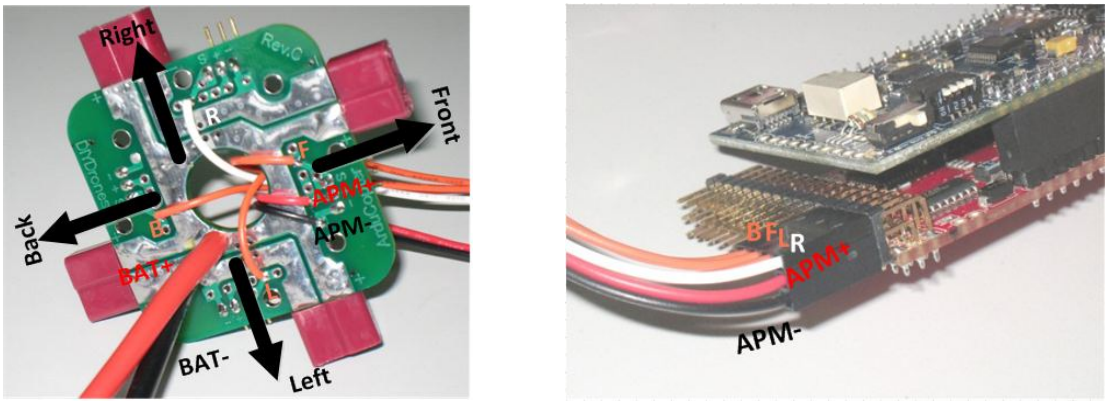
Za reguliranje brzine vrtnje svakog motora zadužena su četiri ESC (*Electronic Speed Controller*) sklopa, slika 7. Na ulaz tih sklopova dolazi istosmjerni napon iz kojeg se generira trofazni izmjenični napon za pokretanje motora. Struja koja može teći kroz sklop, da ne bi došlo do oštećenja, iznosi $20A$, a maksimalna struja koja može teći iznosi $25A$ kroz period od $10s$.

ESC sklopovi spajaju se na posebnu pločicu na koju dolazi napajanje s baterije, slika 8a. Izlazi $0-3$ APM pločice spajaju se na pločicu napajanja te se prosljeđuju na ESC sklopove, slika 8b. Na taj način ESC sklopovi dobivaju upravljačku naredbu o željenoj brzini vrtnje pojedinog motora. Upravljačka se veličina zatim prevodi u odgovarajući napon na stezaljkama motora. Također, napajanje APM-a može biti izvedeno preko pločice napajanja kako je prikazano slikom 8



Slika 6: IMU pločica Arducopter-a

Slika 7: ESC sklop za generiranje trofaznog izmjeničnog napona

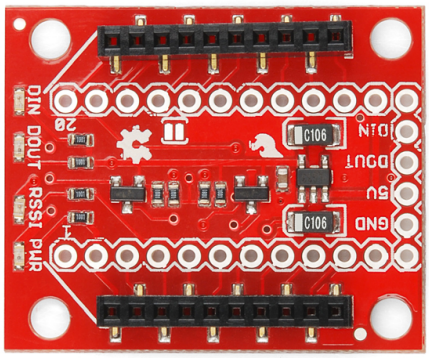


(a) Pločica napajanja (b) Spajanje s APM-om

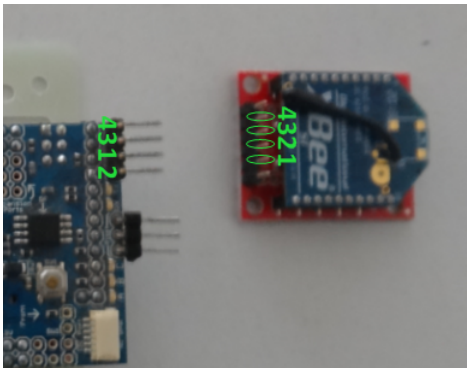
Slika 8: Ispravno spajanje pločice napajanja i APM-a

Upravljačke naredbe se na letjelicu mogu slati putem XBee modula. U tom slučaju potrebno je upariti dva modula koja se koriste za komunikaciju te priključiti jednog od njih na letjelicu. Za spajanje modula na letjelicu korištena je *XBee Explorer Regulated* pločica prikazana na slici 9. Ispravno

spajanje XBee modula s IMU pločicom Arducopter-a prikazano je slikom 10.



Slika 9: Pločica *XBee Explorer Regulated*



Slika 10: Spajanje XBee pločice na IMU pločicu

3.3.3. Motori

Letjelicu pogone četiri bezkolektorska istormjerna motora tipa *AC2836-358* [21], slika 11. Zbog malih dimenzija $(28 \times 36mm)$ te male mase $(70g)$ ovi motori su idealni za primjenu na bespilotnim letjelicama. Maksimalna snaga motora iznosi $P_{max} = 248W$, a postiže se za približne vrijednosti napona $U \approx 12V$ i struje $I \approx 20A$. Ovisnost brzine vrtnje o naponu iznosi $n[RPM] = U[V] \cdot 880[RPM/V]$ (*RPM - Rounds Per Minute*).

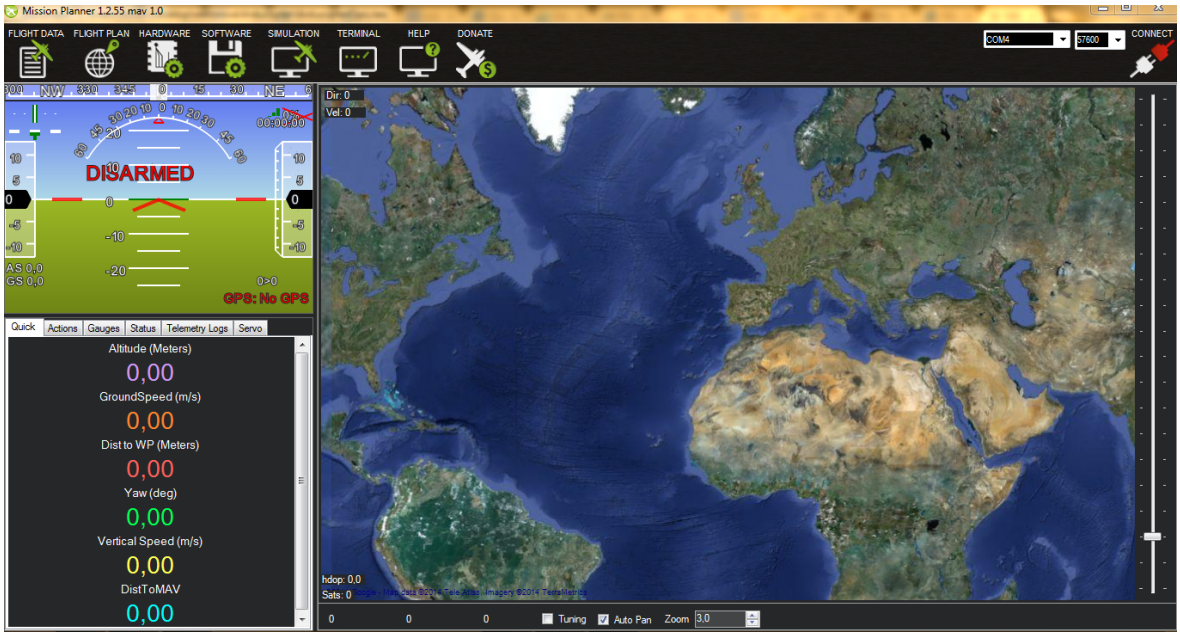
3.3.4. Spajanje s računalom

Za spajanje letjelice s računalom koristi se program *Mission Planner* putem *Mavlink* protokola. Za ispravnu komunikaciju potrebno je podesiti parametre u gornjem desnom kutu sučelja, slika 12. Lijevi parametar odnosi se komunikacijski port na računalu, koji će program sam detektirati. U slučaju više spojenih uređaja korisnik mora odabrati letjelicu za ispravan nastavak spajanja. Drugi parametar odnosi se na brzinu slanja podataka odnosno *baudrate*. Ako se za spajanje koristi USB kabel, parametar se postavlja na $baudrate = 115200 simbol/s$, dok se u slučaju spajanja putem XBee modula postavlja na $baudrate = 57600 simbol/s$. Nakon toga potrebno je kliknuti na opciju *Connect*

Slika 11: Motori korišteni na letjelici

čime počinje komunikacija između računala i letjelice.

Na samom početku *Mission Planner* dohvaća sve parametre zapisane u memoriji letjelice. Ako je letjelica spojena na računalo po prvi puta potrebno je učitati odgovarajući *firmware* na pločicu što se može uraditi odabirom kartice "HARDWARE" na gornjem izborniku. U okviru ovog rada odabran je *firmware* za *ArduCopter* u **X** konfiguraciji. Odabirom kartice "SOFTWARE" dolazi se do postavki letjelice, koje je moguće promijeniti. Također, na ovoj kartici moguće je kalibrirati senzore letjelice što je nužnost prije leta. Odabirom kartice "TERMINAL" dolazi se do postavki letjelice u komandnoj liniji. Odabirom kartice "FLIGHT DATA" moguće je postaviti letjelicu u stanje pripravnosti za let te podesiti igraću palicu za pokretanje letjelice.



Slika 12: Početni ekran sučelja programa *Mission Planner*

3.3.5. Kalibriranje letjelice

3.3.5.1. Kalibracija ESC sklopova

Nakon sastavljanja letjelice potrebno je izvesti početnu kalibraciju letjelice. Prvo je potrebno kalibrirati ESC sklopove [16] u koje se kalibracijom upisuju minimalna i maksimalna vrijednost gasa (eng. *Throttle*). Nekalibrirani ESC sklopovi puštaju zvučne signale u jednakim vremenskim razmacima. U tom slučaju, za kalibriranje, potrebno je pratiti slijedeće korake:

1. Isključiti napajanje i USB priključak
2. Uključiti RC odašiljač i postaviti gas na maksimum
3. Spojiti napajanje. U ovom trenutku, ovisno o modelu, ESC sklopovi bi trebali ispustiti dva kratka zvučna signala.
4. Pričekati da se sustav podigne
5. Ostaviti gas na maksimumu i iskopčati napajanje. Ponovo ukopčati napajanje, u ovom trenutku APM šalje signal s RC odašiljača direktno na ESC sklopove. Nakon kratkog vremena ESC sklopovi puštaju dva kratka zvučna signala nakon kojih slijedi kratka stanka. Unutar stanke potrebno je pomaknuti gas na minimum nakon čega slijedi još jedan zvučni signal.
6. Prije gašenja dodati malo gasa kako bi se potvrdila kalibracija te provjeriti pale li se motori sinkronizirano.
7. Ako se motori pale sinkronizirano kalibracija je uspješna, u protivnom ponovo proći postupak.

Opisan postupak kalibracije nije bilo moguće provesti bez RC odašiljača i prijemnika.

3.3.5.2. Kalibracija akcelerometra

Odabirom kartice "HARDWARE" dolazi se do izbornika u kojem je moguće kalibrirati senzore letjelice [18], a samim time i akcelerometar. Korištenjem grafičkog sučelja akcelerometar se može kalibrirati uz pomoć čarobnjaka. Kalibracija se sastoji od šest koraka u kojima se letjelica postavlja u različite položaje u kojima se pamti trenutno stanje akcelerometra:

1. Položiti letjelicu na ravnu podlogu
2. Postaviti letjelicu tako da prednji dio gleda prema dolje
3. Postaviti letjelicu tako da prednji dio gleda prema gore
4. Postaviti letjelicu na lijevi bok
5. Postaviti letjelicu na desni bok
6. Položiti letjelicu na ravnu podlogu na stražnju stranu

3.3.5.3. Kalibracija magnetometra

Kao i za akcelerometar, izbornik za kalibriranje magnetometra nalazi se pod karticom "HARDWARE" te se kalibracija izvodi uz pomoć čarobnjaka. Pokretanjem kalibracije prikupljaju se podatci s magnetometra kroz vrijeme $T = 60s$ na temelju kojih se računaju odmaci magnetometra za trenutnu lokaciju. Za vrijeme kalibracije potrebno je pomicati i rotirati letjelicu.

3.3.6. Povezivanje s ROS-om

Za pokretanje letjelice putem ROS-a korišten je paket *roscopter*. Paket nudi mogućnost upravljanja letjelicom te dohvaćanja podataka s letjelice. Čvor se može pokrenuti naredbom:

```
rosrun roscopter roscopter.py --device=/dev/ttyUSB1 --baudrate=57600
--enable-control=true
```

pri čemu argument `--device` označava komunikacijski uređaj, `--baudrate` specificira brzinu prijenosa, a `--enable-control` omogućava ili zabranjuje slanje upravljačkih naredbi na letjelicu.

Upravljačke se naredbe šalju na temu *rc* koju stvara čvor *roscopter* te poprimaju vrijednosti u intervalu $1000 - 2000$. Na temu *rc* šalju se podatci u obliku niza od osam cjelobrojnih podataka. Prva četiri podatka niza su redom: zakret oko $y - osi$ (*pitch*), zakret oko $x - osi$ (*roll*), gas (*throttle*) te zakret oko $z - osi$ (*yaw*).

3.3.7. jDrones Arducopter

Osim letjelice Arducopter tvrtke *3D Robotics* u ovom radu korištena je i letjelica Arducopter tvrtke *jDrones*. Na obje letjelice korištena je ista elektronika. Dvije su razlike: tijelo letjelice (eng. *frame*) te motori, slika 13.

Slika 13: Bespilotna letjelica Arducopter tvrtke *jDrones*

3.3.7.1. Motori

Motori na *jDrones Arducopter*, slika 14, letjelici nešto su slabiji od onih na *3DRobotics Arducopteru*. Specifikacije motora dane su tablicom 3.

Tablica 3: Specifikacije motora *AC2830-358* [31]

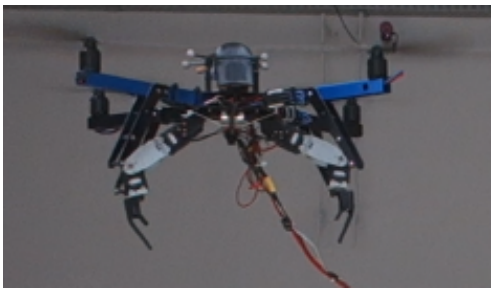
Dimenzije	$28 \times 30mm$
Masa	$52g$
Brzina vrtnje	$U[V] \cdot 850[RPM/V]$
Maksimalna snaga	$200W$

Slika 14: Motor *AC2830-358*

3.4. Bespilotna letjelica s dvostrukim robotskim manipulatorom

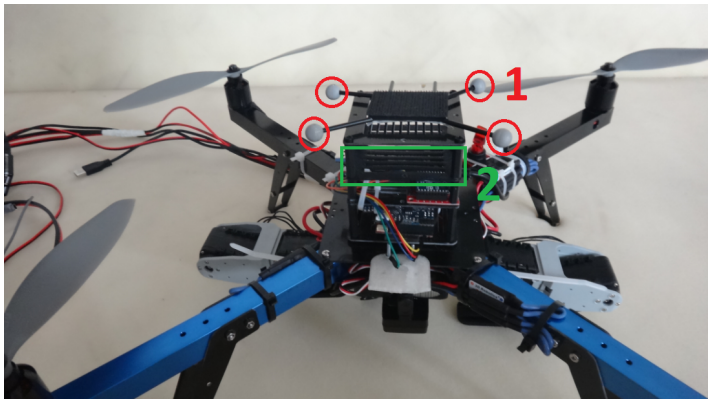
Za potrebe testiranja korisničkog sučelja potrebno je bilo modificirati letjelicu Arducopter tvrtke *3DRobotics* ugradnjom dvostrukog robotskog manipulatora i ArDrone Parrot kamera. Dvostruki robotski manipulator je potreban kako bi letjelica mogla uhvatiti ventil te ga otvoriti/zatvoriti. Na slici 15 prikazana je modificirana bespilotna letjelica s dvostrukim robotskim manipulatorom. Na donju stranu letjelice pričvršćena je ArDrone Parrot kamera kako bi osoba koja upravlja letjelicom imala vizualnu povratnu vezu o udaljenosti od ventila te koliko je ventil otvoren/zatvoren. Uz ArDrone Parrot kameru potrebna je i ArDrone Parrot matična ploča kako bi bilo moguće dobiti sliku s kamere. Odabrana je ta kamera zbog malih dimenzija, dobre kvalitete slike i lakog povezivanja s upravljačkim računalom. Detaljnije informacije o kameri i povezivanju kamere mogu se pronaći u [36]. Za dobivanje slike s kamere korišten je ROS paket *ardrone_autonomy* [32]. Potrebno je bilo izraditi plastičnu

pločicu na kojoj je pričvršćena matična ploča te je montirati na bespilotnu letjelicu. Plastična pločica, prikazana na slici 16, modelirana je u 3D CAD programu AutoCAD, dimenzija je 75x75 mm, te je isprintana 3D printerom.



(a) Manipulator spreman za upravljanje

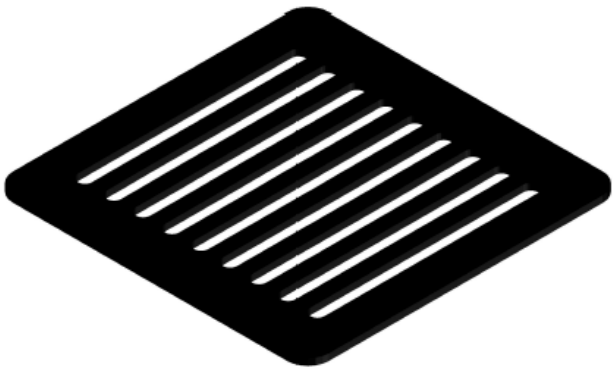
(b) Manipulator u položaju pogodnom za let



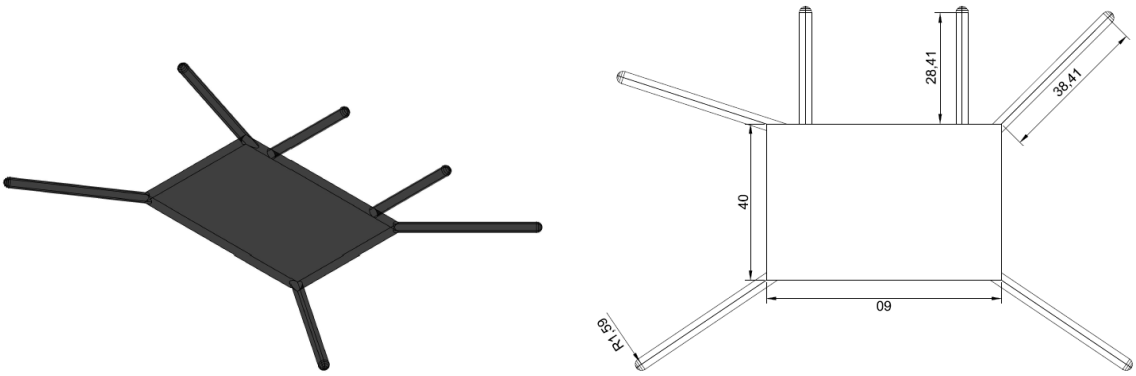
(c) Krupniji prikaz bespilotne letjelice

Slika 15: Prikaz bespilotne letjelice s dvostrukim robotskim manipulatorom

Na slici 15c crvenom bojom (broj 1) označeni su reflektirajući markeri koji se koriste kako bi se pratila pozicija i orijentacija letjelice sustavom OptiTrack. Zelenom bojom (broj 2) označena je plastična pločica na kojoj se nalazi matična ploča ArDrone Parrota. Reflektirajući markeri nalaze se na plastičnom nosaču koji je također modeliran u AutCAD-u i isprintan 3D printerom. Plastični nosač prikazan je na slici 17, dimenzije su izražene u milimetrima.



Slika 16: Plastična pločica modelirana u AutoCAD-u



(a) Realističan izgleda plastičnog nosača (b) Dimenzije plastičnog nosača

Slika 17: Plastični nosač za reflektirajuće markere modeliran u AutoCAD-u

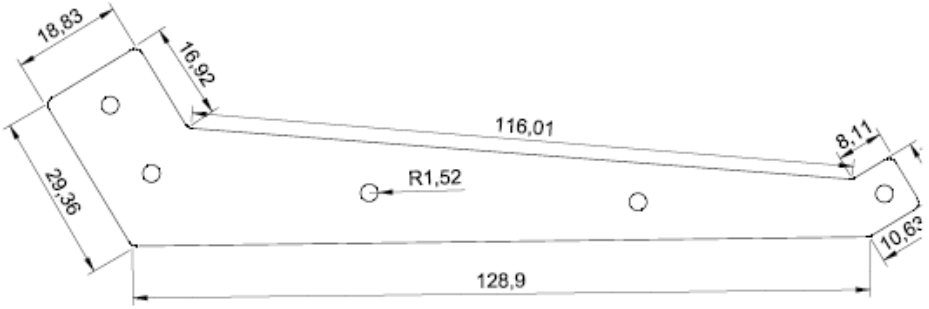
3.5. Bespilotna letjelica s kamerom

Druga bespilotna letjelica koja je modificirana je ArduCopter bespilotna letjelica tvrtke jDrones. Provedene su dvije modifikacije: dizajnirane su nožice kako bi slijetanje letjelice bilo što elegantnije, montirana je Logitech kamera. Bespilotna letjelica je prikazana na slici 18. Nožice su modelirane u 3D CAD programu AutoCAD te su ispirntane 3D printerom. Slika 19 prikazuje dizajn i dimenzije nožica.

Slika 18: Bespilotna letjelica s kamerom

Isprintano je 8 takvih nožica te su u paru montirane na bespilotnu letjelicu. Između svakog para nožica umetnuta je spužva. Na letjelicu su postavljena tri reflektirajuća markera kako bi je bilo moguće pratiti sustavom za praćenje pokreta OptiTrack. Reflektirajući markeri postavljeni su na plastičan nosač prikazan na slici 17.

(a) Realističan izgleda plastične nožice



(b) Dimenzije plastične nožice

Slika 19: Plastična nožica modelirana u AutoCAD-u

3.6. Ventil

Izrađen je željezni ventil koji je montiran na plastične cijevi. Ventil se ne može zatvoriti ili otvoriti već sadrži ležajeve kako bi ga letjelica mogla zavrnuti. Slikom 20 prikazana je izvedba ventila.

Slika 20: Metalni ventil montiran na plastične cijevi

3.7. XBee

XBee modul [11] je uređaj za bežičnu komunikaciju tvrtke *Digi International*. Uređaj koristi 802.15.4 IEEE standard za bežičnu komunikaciju, koji je osmišljen za primjene s malom brzinom prijenosa te smanjenom potrošnjom energije. Frekvencije na kojima se može odvijati komunikacija su: $868 - 868.6MHz$, $902 - 928MHz$ te $2.400 - 2.3835GHz$ pri čemu je zadnji pojas frekvencija najčešće korišten. Unutar samog pojasa postoji više komunikacijskih kanala u razmaku od $5MHz$ počevši od $2.405GHz$ do $2.480GHz$.

U današnje vrijeme XBee modul se sve češće koristi zbog male potrošnje energije, relativno male cijene te raznih primjena. Velika je prednost što na samom uređaju postoje digitalni ulazi i izlazi zbog čega u jednostavnim primjenama nema potrebe za korištenjem mikrokontrolera. Također, na uređaju se nalaze i PWM (*Pulse Width Modulation*) izlazi.

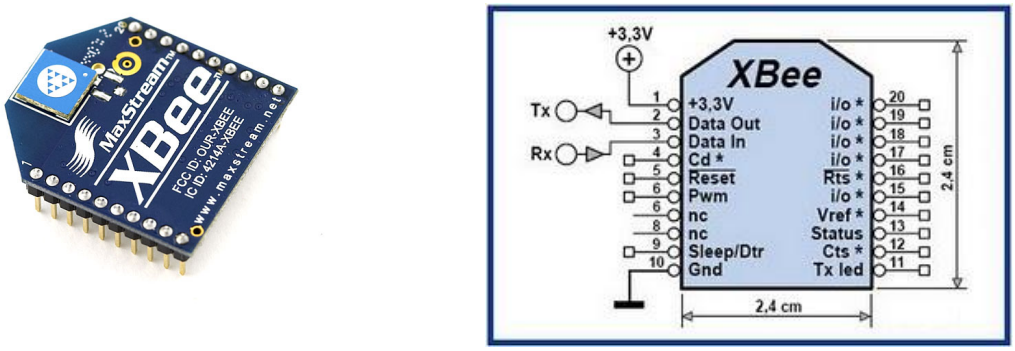
3.7.1. Opis uređaja

Na slici 21 prikazan je uređaj s pripadnim rasporedom pinova. Funkcije pojedinih pinova dane su tablicom:

Tablica 4: Funkcije pinova XBee modula

Pin	Funkcija
1	Napajanje modula $U = 3.3V$
2	UART izlaz
3	UART ulaz
4	Digitalni izlaz 8
5	Resetiranje modula
6-7	Izlazi PWM signala
8	Rezerviran pin
9	Digitalni ulaz 8
10	Uzemljenje
11, 15-20	Analogni ulazi ili digitalni izlazi/ulazi
12	Digitalni izlaz 7
13	Postavljanje pločice u štedni/aktivni mod rada
14	Referentni napon za analogne i digitalne ulaze

Postoje dvije različite inačice modula: XBee i XBee-PRO. Glavna razlika između inačica je potrošnja energije i domet komunikacije. Kod osnovne varijante XBee modula domet iznosi $30m$, dok kod XBee-PRO modula domet iznosi $90m$ u zatvorenom te $90m$ odnosno $1600m$ na otvorenom prostoru. Maksimalna struja koju osnovni modul može trošiti iznosi $45mA$ pri naponu $3.3V$, a snaga odašiljača iznosi $1mW$. Kod XBee-PRO varijante maksimalna struja iznosi $250mA$ pri naponu $3.3V$, a snaga odašiljača iznosi $63mW$. Kako u ovom radu nije bilo potrebe za velikim dometom jer je letjelica uvijek bila blizu upravljačkog računala, odabran je osnovni XBee modul.



(a) Modul (b) Ulazi i izlazi modula

Slika 21: Prikaz XBee uređaja

3.7.2. Podešavanje modula

Za konfiguriranje modula može se koristiti besplatan softver *X-CTU* tvrtke Digi International. Za spajanje XBee modula na računalo koristi se pločica *XBee Explorer USB* koja omogućava komunikaciju preko mini USB kabela. Da bi se preko *X-CTU* uspješno uspostavila komunikacija s modulom potrebno je podesiti parametar *Baud* tako da odgovara vrijednosti zapisanoj na modulu, *Baud Rate* označava brzinu slanja podataka izraženu u simbolima po sekundi. Zato je važno da su na modulu i računalu zapisane iste vrijednosti kako bi se sinkronizirala komunikacija.

Nakon uspješnog spajanja moguće je promijeniti postavke modula na željene vrijednosti. Tvornički postavljen parametar *Baud Rate* na modulu iznosi $BaudRate = 9600 simbol/s$. U postavkama se taj parametar nalazi pod nazivom *Interface Data Rate* te se može postaviti na željenu vrijednost. Da bi se uparila dva *XBee* modula potrebno im je postaviti istu brzinu prijenosa te jednak komunikacijski kanal, koji se u postavkama nalazi pod nazivom *Channel*.

Kod pisanja parametara na modul može doći do greške. U tom slučaju više nije moguće uspostaviti komunikaciju s modulom. Ovaj problem može se riješiti ručnim resetiranjem odnosno kratkim spajanjem pinova 1 i 5 čije su funkcije dane tablicom 4.

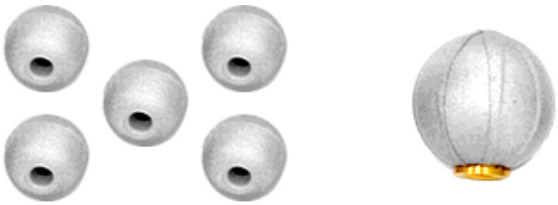
3.8. OptiTrack

OptiTrack je sustav za praćenje pokreta tvrtke NaturalPoint, pruža praćenje pokreta za komercijalne, industrijske i znanstvene uporabe. Dizajniran je za praćenje markera, pokreta tijela i lica. ACROSS centar za istraživanje opremljen je s osam Flex 13 kamera rezolucije 1.3 MP (1280×1024), vidnog polja 56° i frekvencije do 120 sličica po sekundi (eng. *FPS - frames per second*) te kašnjenja od $8.3ms$ (eng. *latency*)[6]. Prostor koji pokrivaju kamere, prikazan na slici 24, veličine je $250 \times 420 \times 850$ kubičnih centimetara [7]. U kombinaciji s OptiTrack algoritmima može pratiti predmet s tolerancijom manjom od $0.5mm$. Flex 13 kamera, prikazana na slici 22, može slati različite tipove predobrađene slike (*object, precision, segment, MJPEG grayscale, raw grayscale*) na računalo što rezultira s efikasnijom obradom slike i kalibracijom sustava. Kamere emitiraju infracrvenu svjetlost valne duljine $850nm$ koja se odbija od posebno dizajniranih markera te se na taj način računa pozicija markera, koji mora biti u vidnom polju dviju ili više kamera.

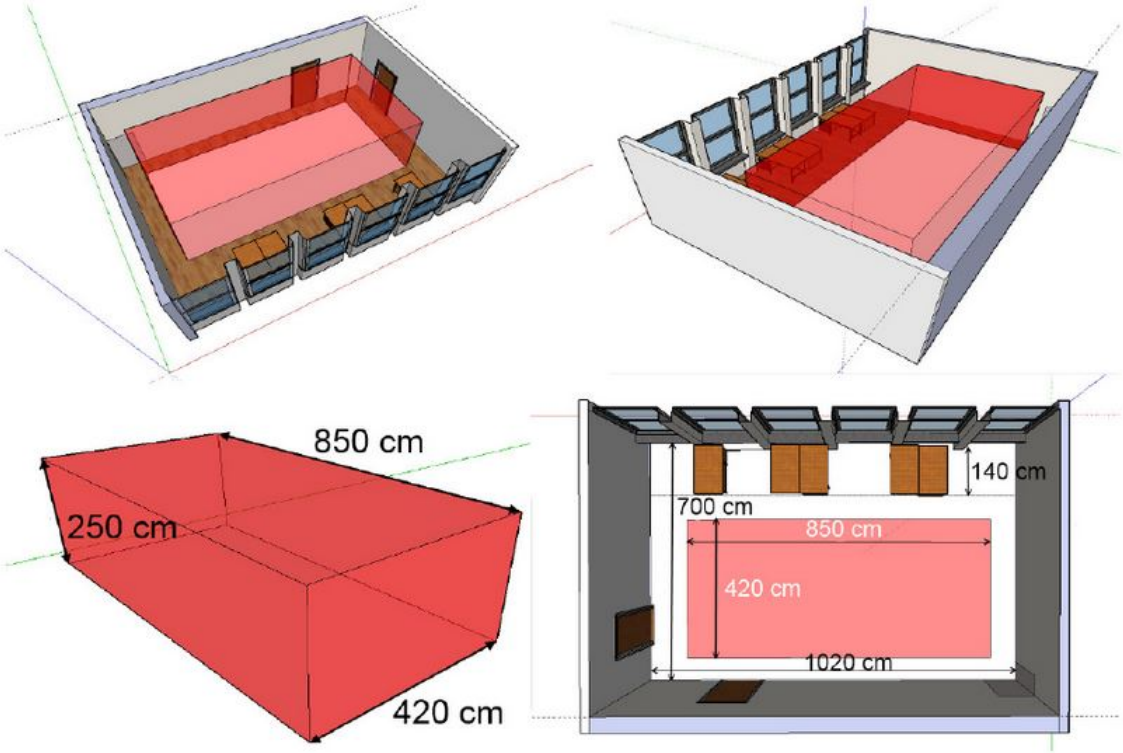
Slika 22: Flex 13 kamera

U centru za istraživanje nalazi se stolno računalo na kojem je instalirana aplikacija Motive. Kamere su spojene na računalo preko USB porta. Podešena je vlastita mreža koja omogućuje slanje podataka između stolnog računala i svakog računala na mreži.

NaturalPoint Motive aplikacija za praćenje pruža kalibraciju i 3D praćenje više objekata istovremeno u realnom vremenu. Kako bi se pratila 3D pozicija i orijentacija objekta, na objekt je potrebno asimetrično postaviti 3 reflektirajuća markera na minimalnu udaljenost od $6mm$. Primjer takvih markera nalazi se na slici 23. Korištenje više od tri markera povećava preciznost.

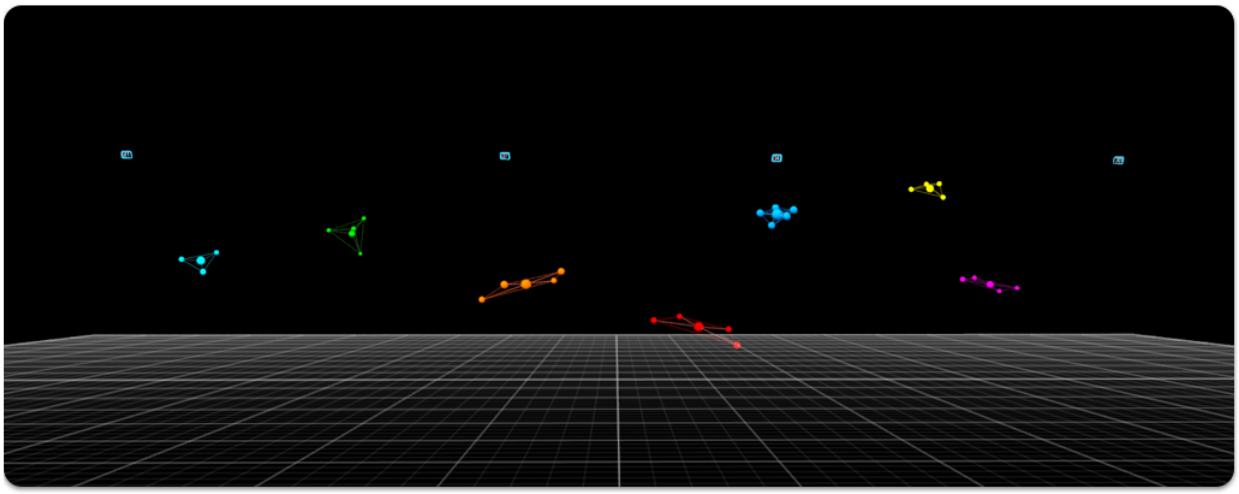


Slika 23: Reflektivni markeri



Slika 24: Područje pokrivenosti kamerama ACROSS centra za istraživanje

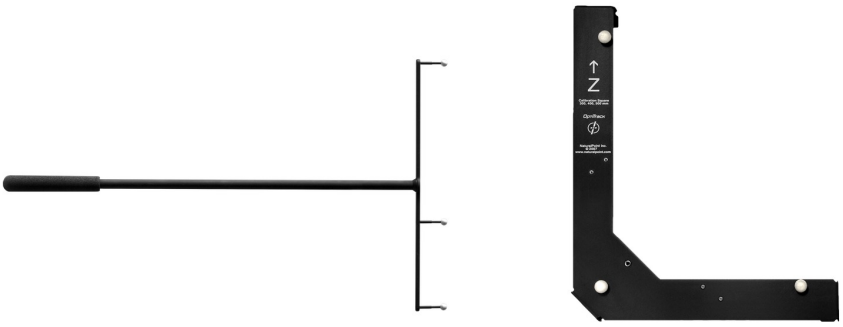
Slika 25 prikazuje istovremeno praćenje više objekata. Svaki objekt ima asimetrično postavljena tri ili više reflektivnih markera i u vidnom je polju dvije ili više kamera.



Slika 25: Istovremeno praćenje više objekata

Prije samog početka praćenja objekta potrebno je sustavu dati informaciju o položaju kamera u prostoru, što se postiže kalibracijom. Prvi korak uspješne kalibracije je micanje ili programsko blokiranje (opcija *block visible*) svih fizičkih elemenata koji bi mogli interferirati s kamerama. Početak kalibracije počinje odabirom opcije *start wanding*. Nakon detektiranja štapa za kalibraciju, na kojem se nalaze tri reflektirajuća markera, prikazanog na slici 26a, sustav će automatski započeti sakupljati uzorke. Koristeći štap za kalibraciju potrebno je sakupiti što više uzoraka. Kada je sakupljen dovo-

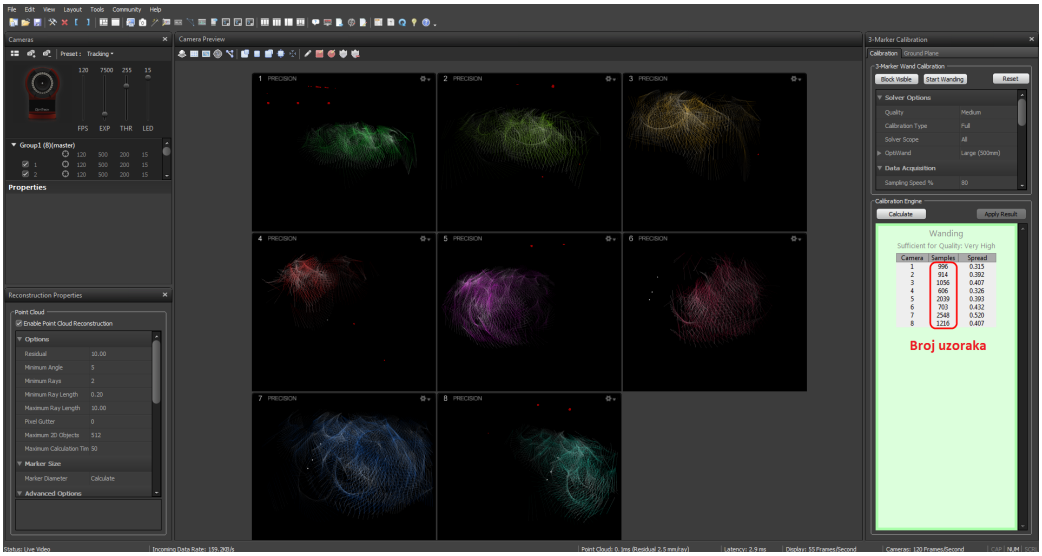
ljan broj uzoraka potrebno je odabrati opciju *calculate* i pričekati da se pojavi *ready to apply* te zatim odabrati opciju *apply result* za spremanje rezultata kalibracije. Slika 27 prikazuje sakupljanje uzoraka za kalibraciju. Slijedeći korak je definiranje tla (eng. *ground plane*) tako da se postavi predmet s tri reflektirajuća markera, prikazan na slici 26b, na poziciju u kojoj želimo ishodište koordinatnog sustava.



(a) Štap za kalibraciju (b) Postolje za umjeravanje koordinatnog sustava

Slika 26: Potrebni predmeti za kalibraciju

Definiranje objekta za praćenje postiže se odabirom reflektirajućih markera u aplikaciji i pritiskom desne tipke koja otvara izbornik u kojem se odabire *rigid body*, a zatim *create from selected markers*. Tok podataka (eng. *data stream*) uključuje se u izborniku *Streaming Properties* odabirom *Broadcast Frame Data*.



Slika 27: Sakupljanje uzoraka za kalibraciju

3.8.1. Povezivanje s ROS-om

Rad s podacima koje šalje Motive aplikacija na mrežu omogućava besplatan paket *ACROSS_Optitrack* koji se može preuzeti sa stranice [8]. Paket sadrži čvor *MOCAPNode* koji šalje primljene podatke u obliku ROS poruke *geometry_msgs/Pose* u temu */Optitrack*. Paket je napisan u C++ programskom

jeziku, koristi NatNet protokol za povezivanje s računalom na kojem se nalazi Motive aplikacija i pružao podatke u stvarnom vremenu. NatNet koristi UDP protokol, prijenos podataka može se obaviti tako da izvorni čvor adresira paket koristeći adresu koja će biti na odredištu, potom šalje paket na mrežu, a potom na odredište (*unicast* prijenos). Podaci se mogu slati i na način da se paket podataka kopira i šalje na specifične podskupove uređaja na mreži, izvor adresira paket koristeći *multicast* adresu, te potom kopira paket i šalje kopije svakom čvoru (korisniku) koji je dio *multicast* adrese (*multicast* prijenos). NatNet server sadrži dvije dretve (eng. *thread*) i dva sučelja (eng. *socket*), jedan za slanje podataka, a drugi za primanje/slanje naredbi. NatNet server i klijent mogu se nalaziti na istom ili različitim računalima, a više NatNet klijenta mogu se spojiti na isti NatNet server. *ACROSS_Optitrack* također nudi mogućnost praćenja više objekata u isto vrijeme na način da se podaci svakog objekta objavljuju u zasebnu temu.

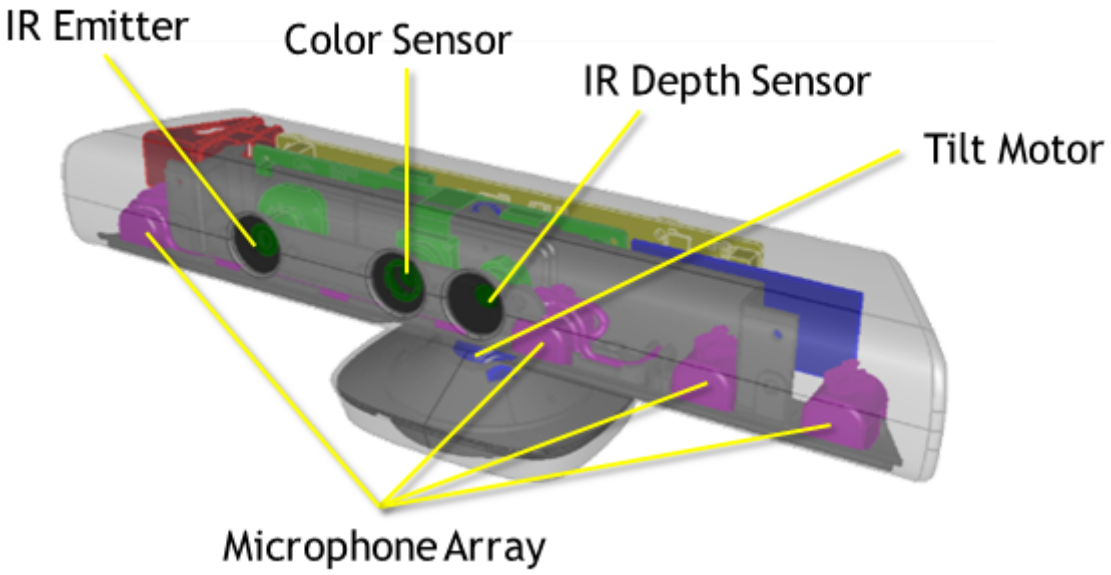
Čvor se pokreče na sljedeći način:

```
rosrun ACROSS_Optitrack MOCAPNode
```

3.9. Kinect

Uređaj Kinect je razvila tvrtka *Microsoft* za *Xbox 360* igraću konzolu. Uređaj je osmišljen kako bi se uz korištenje odgovarajućih algoritama detektirali pokreti čime omogućuje korisniku interakciju bez korištenja upravljača. Pri tome korisnik ne mora razmišljati o upravljačkim naredbama jer jednostavnim i intuitivnim pokretima i gestama ostvaruje interakciju s video igrom. Kinect nije usko vezan uz igraću konzolu *Xbox 360* zbog mogućnosti spajanja s računalom čime se uvelike proširuje primjena.

3.9.1. Opis uređaja



Slika 28: Uređaj Kinect tvrtke *Microsoft*

Na uređaju se nalazi nekoliko senzora [22]. Srž uređaja čine video kamera odnosno *Color Sensor* čije su specifikacije dane tablicom 5 te senzor dubine *Depth Sensor*. Rezolucija video kamere može se podesiti, ali kroz eksperimente je utvrđeno da je rezolucija od $640 \times 480px$ (*pixela*) sasvim dovoljna za detekciju pokreta. Tome u prilog ide i brzina snimanja koja je u tom slučaju $30fps$ (*Frames Per Second*) što znači češće osvježavanje podataka.

Tablica 5: Specifikacije RGB kamere

Brzina snimanja uz rezoluciju $640 \times 480px$	$30fps$
Brzina snimanja uz rezoluciju $1280 \times 960px$	$15fps$
Horizontalno vidno polje	58°
Vertikalno vidno polje	43°

Za mjerenje udaljenosti objekta od Kinect-a koristi se infracrveni laser u paru s monokromnim CMOS fototranzistorom odnosno senzor dubine (*eng. depth sensor*). U starijim verzijama Kinecta bio je ugrađen samo jedan infracrveni laser dok se u novijim verzijama nalaze tri sinkronizirana lasera. Fototranzistor odnosno *IR Depth Sensor* sa slike 28 reagira na podražaje infracrvene svjetlosti. Senzor udaljenosti je konfiguriran za mjerenje udaljenosti u rasponu $1.2 - 3.5m$, no ako je potrebno radno područje se može proširiti. Specifikacije senzora udaljenosti dane su tablicom 6.

Tablica 6: Specifikacije senzora udaljenosti

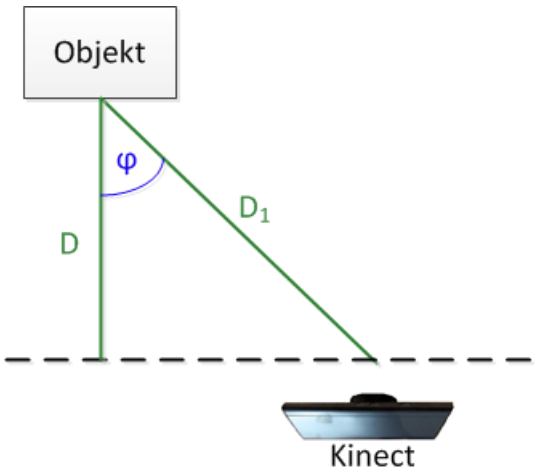
Rezolucija	$320 \times 240px$
Brzina snimanja	$30Hz$
Valna duljina zrake lasera	$830nm$
Snaga lasera	$60mW$
Preporučena konfiguracija raspona udaljenosti	$[1.2, 3.5]m$
Proširena konfiguracija raspona udaljenosti	$[0.7, 6]m$

Mjerenje udaljenosti objekta od senzora udaljenosti radi na sljedećem principu. Infracrveni laser ispaljuje zraku svjetlosti te u tom trenutku počinje mjerenje vremena. Fototranzistor čeka ispaljenu zraku te kada ju prima završava mjerenje vremena. Ako se vremenski interval između slanja i primanja signala označi s Δt uz poznatu brzinu svjetlosti $c = 2.998 \cdot 10^8 \frac{m}{s}$ formula za izračun udaljenosti glasi:

$$D_1 = \frac{c \cdot \Delta t}{2} \tag{1}$$

Dijeljenje s dvojkom nužno je u ovom slučaju jer ispaljena zraka svjetlosti dvostruku udaljenost od uređaja do objekta.

Osim samog mjerenja udaljenosti važno je uočiti razliku između stvarne i procijenjene udaljenosti od uređaja prikazanih slikom 29 .



Slika 29: Prikaz stvarne i procijenjene udaljenosti od uređaja

Procijenjena udaljenost D mnogo je praktičnija za obradu od stvarne udaljenosti. Ta je udaljenost mjerena je od ravnine u kojoj se nalaze *RGB* i dubinska kamera. Na taj se način na svaka četiri *pixela RGB* kamere, pri rezoluciji $640 \times 480px$, pridaje jednaka udaljenost od uređaja. Uz poznate veličine φ, D_1 i D sa slike 29 te se uz poznato mjerenje stvarne udaljenosti dane izrazom (1) dolazi do formule za procijenjenu udaljenost:

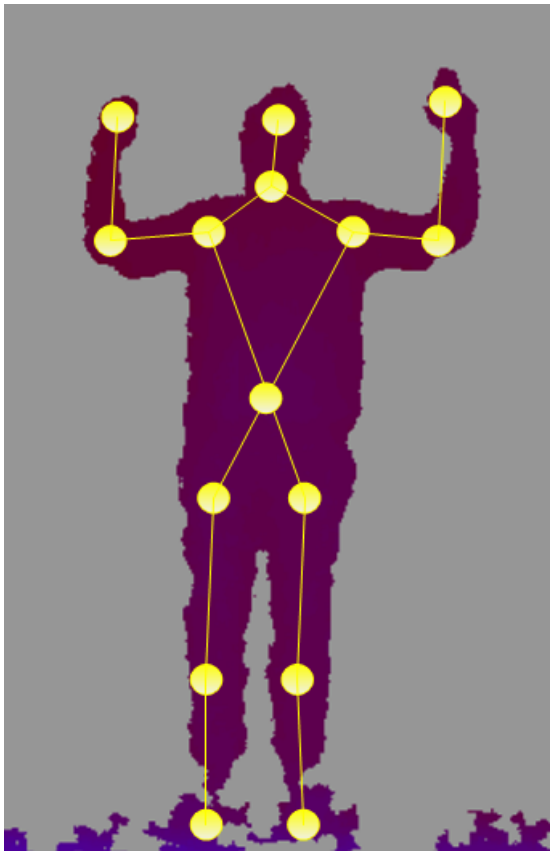
$$\begin{aligned} D &= D_1 \cdot \cos(\varphi) \\ D &= \frac{c \cdot \Delta t}{2} \cdot \cos(\varphi) \end{aligned} \tag{2}$$

Osim kamera, na uređaju se nalaze i četiri mikrofona frekvencije uzorkovanja $f = 16kHz$ te $16-bitnu$ veličinu audio zapisa. U ovom radu mikrofoni nisu korišteni zbog buke izazvane letjelicama.

3.9.1.1. Povezivanje s *ROS*-om

Za slanje podataka o položaju tijela u prostoru koristi se *ROS* odnosno čvor *openni_tracker* [24]. Čvor dohvaća informacije o karakterističnim točkama ljudskog tijela i prosljeđuje ih u *ROS*. Na samom početku čvor detektira korisnika te pokreće kalibraciju. Od korisnika se očekuje da se postavi u položaj Psi (eng. *Pose Psi*) prikazan na slici 30. Preporučena udaljenost od uređaja za vrijeme kalibracije iznosi $d = (2-2.5)[m]$. Ponekad se potrebno pomaknuti nekoliko koraka naprijed ili nazad kako bi se kalibracija uspješno izvršila.

Nakon uspješne kalibracije čvor *openni_tracker* počinje slati podatke tipa *TFMessage* na temu *tf*, na koju se pretplaćuje čvor *TransformToSkeletons*. Podatci su nakon pretvorbe i filtriranja u *TransformToSkeletons* spremni za daljnje korištenje.



Slika 30: Položaj Psi potreban za kalibraciju prije mogućnosti praćenja kostura

3.10. Dvostruki robotski manipulator

Dvostruki robotski manipulator s dva stupnja slobode, korišten u ovom radu, prikazan je slikom 31. Manipulator se sastoji od četiri servo motora tipa *Dynamixel* tvrtke *Robotis*, nosača za ugradnju na letjelicu te dvije hvataljke (eng. *gripper*) prilagođene za hvatanje ventila.

3.10.1. *Dynamixel* servo motori

Za pokretanje dvostrukog manipulatora korišten je *AX-12A* model servo motora [19], 32a. Specifikacije servo motora dane su tablicom:

Slika 31: Dvostruki robotski manipulator

Tablica 7: Specifikacije *Dynamixel* AX-12A servo motora

masa	$54.6g$
dimenzije	$32mm \times 50mm \times 40mm$
rezolucija mjerenja	0.29°
prijenosni omjer	$254:1$
potezni moment (eng. *stall torque*)	$1.5Nm$ pri $U = 12V, I = 1.5A$
brzina bez tereta	$59rpm$
radno područje	$0 - 300^\circ$
raspon temperature	$-5^\circ - 70^\circ$
raspon napona	$9V - 12V$

(a) *Dynamixel* AX-12A (b) *USB2Dynamixel* adapter

Slika 32

Model *AX-12A* odabran je zbog male mase $m = 54.6g$ čime ukupna masa dvostrukog manipulatora iznosi $m \approx 250g$. Za komuniciranje s motorima korišten je *USB2Dynamixel* adapter prikazan slikom 32b koji omogućuje jednostavno povezivanje motora s računalom putem USB kabla.

3.10.2. Povezivanje s ROS-om

Za upravljanje motorima putem ROS-a korišten je gotov paket *dynamixel_controllers*. Čvor *controller_manager.py* služi za inicijalizaciju motora te se kao argumenti predaju postavke komunikacije: USB port na koji je spojen *USB2Dynamixel* adapter, *baudrate*, minimalni i maksimalni indeks motora koje čvor može dodijeliti te frekvencija slanja podataka. Čvor *controller_spawner* stvara teme u koje se šalju reference zakreta pojedinog motora. Kao parametar predaje se *.yaml* datoteka s postavkama motora slijedećeg oblika:

```
JointB1_controller:
    controller:
        package: dynamixel_controllers
        module: joint_position_controller
        type: JointPositionController
    joint_name: JointB1
    joint_speed: 1.47
    motor:
        id: 10
        init: 512
        min: 0
        max: 1023
```

pri čemu je `JointB1_controller` naziv teme koju čvor stvara za taj motor, `joint_name` ime motora, `joint_speed` maksimalna dozvoljena brzina vrtnje, `id` oznaka motora, *init* početna pozicija te `min` i `max` minimalan i maksimalan zakret. Datoteka pritom može sadržavati postavke za više motora.

3.11. Kamera Logitech HD Webcam C270

Zbog potrebe za vizualnom povratnom vezom u radu je korištena *Logitech HD Webcam C270* kamera, slika 33. Specifikacije kamere dane su tablicom 8. Kamera je odabrana zbog svoje malene mase, $m = 73g$, i visoke kvalitete slike.

Tablica 8: Specifikacije kamere Logitech C270 [33]

Specifikacije kamere	
Vidno polje	60°
Žarišna duljina	$4mm$
Rezolucija slikanja	$1280 \times 960px$
Rezolucija snimanja	$640 \times 480px$ uz brzinu snimanja $30fps$

Slika 33: Logitech C270 kamera

3.11.1. Povezivanje s ROS-om

Za povezivanje s ROS-om koristi se paket *usb_cam* [25]. Od ROS parametara za pokretanje koristili su se *video_device* za odabir video porta na koji je spojena kamera te *framerate* za specificiranje brzine snimanja kamere. U tom slučaju čvor se pokreće naredbom:

```
rosrun usb_cam usb_cam_node _video_device:="/dev/video1" _framerate:=30
```

3.12. PlayStation3 Move navigacijska igraća palica

Igraća palica ima 8 digitalna i 4 analogna ulaza, povezuje se s računalom putem bluetootha, a prikazana je na slici 34. Napajana je baterijom koja se može puniti preko usb priključka koji se nalazi na dnu igraće palice.

Slika 34: PlayStation3 Move igraća palica

3.12.1. Povezivanje s ROS-om

Za povezivanje s ROS-om koristi se paket *ps3joy*. Paket se može preuzeti s [27]. Paket *ps3joy* pruža pristup podacima akcelerometra i žiroskopa igraće palice te trenutnom stanju upravljačkog dijela igraće palice. Pokreće se na sljedeći način:

```
rosrun ps3joy ps3joy.py
```

4. Razrada

Vodeća misao kod nastanka ideje za prošireno korisničko sučelje za upravljanje robotskim manipulatorom u letu bila je postići minimalno opterećenje operatera. Upravo zbog toga odabrana su tri upravljačka uređaja:

- Kinect
- PlayStation3 Move igraća palica
- Mikrofon - glasovne naredbe

Svaki od navedenih upravljačkih uređaja povezan je na upravljačko računalo koje komunicira s bespilotnim letjelicama (poglavlje 3.3) putem komunikacijskog uređaja Xbee (poglavlje 3.7). Pozicija svake letjelice kojom se upravlja prati se sustavom za praćenje pokreta OptiTrack koji je opisan u poglavlju 3.8.

Kinect je odabran zbog sposobnosti prepoznavanja položaja ljudskog tijela u prostoru čime se dvostrukim robotskim manipulatorom može postići imitiranje pokreta ruku operatera. Bespilotnom letjelicom upravlja se pomoću PlayStation3 Move igraće palice u kombinaciji s glasovnim naredbama. Ideja je da se glasovne naredbe koriste za odabir letjelice, slijetanje i uzlijetanje letjelice, postavljanje letjelice u pripravnost za let ili u stanje bez mogućnosti leta. PlayStation3 Move igraća palica koristi se za mijenjanje referentne pozicije letjelice u prostoru te, u slučaju opasnosti za letjelicu ili okolinu, za gašenje svih motora letjelice. Idealan je odabir zbog bežične komunikacije te je malih dimenzija. Kako bi se izbjegle nagle promjene brzine bespilotne letjelice, mijenjanje referentne pozicije zamišljeno je na način da se integrira vrijednost predefinirane brzine tijekom pritiska određene tipke. Time promjena pozicije ne će biti step funkcija već rampa. Slikom 35 prikazana je pojednostavljena shema proširenog korisničkog sučelja. Također, dodana je mogućnost mijenjanja referentne pozicije letjelice putem glasovne naredbe kao i mogućnost mijenjanja iznosa predefinirane brzine. Za razvoj aplikacija korištena je razvojna cjelina ROS.

Slika 35: Pojednostavljena shema upravljanja putem proširenog korisničkog sučelja

4.1. Upravljanje bespilotnom letjelicom

Upravljanje bespilotnom letjelicom implementirano je u čvoru *PI_D_controller*, poglavlje 5.5.1. Testirane su dvije strukture upravljanja: jednopetljasta i višepetljasta struktura upravljanja. Informacije o trenutnoj poziciji i orijentaciji letjelice dobivaju se preko sustava OptiTrack, a referentne veličine zadaje operater putem proširenog korisničkog sučelja. Potrebno je bilo izmijeniti postojeći paket *ACROSS_Optitrack* kako bi osim trenutne pozicije i orijentacije letjelice objavljivao i trenutnu brzinu letjelice. Dodan je i argument `-trackables` kojim se predaje datoteka s imenima letjelica koja postaju prefiksi za ROS teme. Brzina letjelice računa se prema sljedećem izrazu:

$$\dot{q}(k) = \frac{q(k) - q(k-1)}{T} \tag{3}$$

gdje je $q(k)$ vektor pozicije letjelice, a T je vrijeme uzorkovanja.

Izračunata brzina letjelice filtrirana je niskopropusnim filtrom drugog reda, izraz (4). Sustav za praćenje pokreta OptiTrack, u trenutcima kada objekt praćenja nije u vidnom polju dvije ili više kamera (ne prati se promjena pozicije objekta), šalje zadnju poznatu poziciju objekta. U trenutku ponovnog praćenja objekta vrijednost brzine može naglo porasti na vrijednost koja nije valjana te time dovesti do mogućeg oštećenja letjelice. Iz tog razloga u slučaju velike promjene brzine (veće od 0.3 m/s) brzina letjelice postavlja se na 0 m/s, izraz (5).

$$\dot{q}(k) = 0.4 \cdot \dot{q}(k) + 0.3 \cdot \dot{q}(k-1) + 0.3 \cdot \dot{q}(k-2) \tag{4}$$

$$\dot{q}(k) = \begin{cases} 0, & \text{za } |\dot{q}(k) - \dot{q}(k-1)| > 0.3 \\ \dot{q}(k), & \text{inače} \end{cases} \tag{5}$$

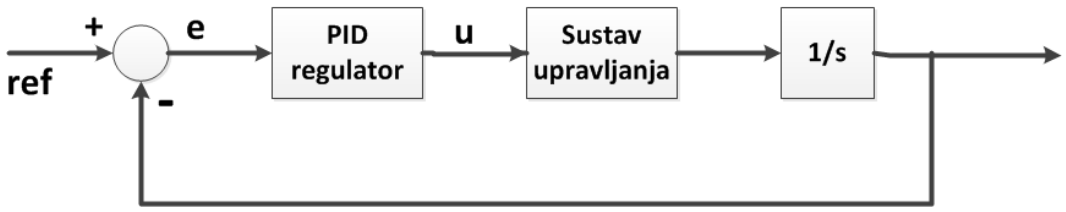
4.1.1. Upravljačka struktura

Razmatrana su dva tipa strukture upravljanja: jednopetljasta i višepetljasta struktura upravljanja. Jednopetljasta struktura upravljanja, slika 36, je struktura upravljanja s jednom povratnom vezom. U našem slučaju zatvorena je povratna veza po poziciji bespilotne letjelice, a u regulacijskom krugu smješten je uobičajeni PID regulator, prikazan na slici 38. Cilj je postići što precizniju regulaciju pozicije letjelice kako bi se mogao dobiti što bolji rezultati kod izvršavanja pojedinih zadataka. No visoke performanse upravljanja složenim sustavima često nije moguće postići korištenjem jednopetljastih struktura upravljanja. Razlog tome ponajprije leži u činjenici da se složenim sustavima nastoji upravljati na temelju samo jedne informacije o sustavu tj. samo na temelju njegove izlazne veličine. Regulator reagira na promjene koje se dogode unutar sustava tek nakon što se njihov efekt registrira na iznosu izlazne veličine. Što dovodi do spore reakcije na djelovanje poremećaja. Iz tog razloga implementirana je kaskadna struktura upravljanja (kaskadna regulacija) prikazana na slici 37.

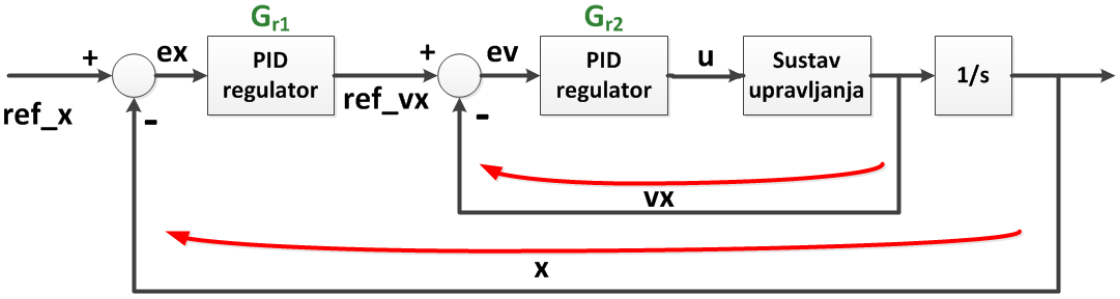
Sustav upravljanja sa slike 37 ima dva regulacijska kruga(petlje):

- 1 - pomoćni regulacijski krug (unutarnji regulacijski krug, podređeni regulacijski krug)

- 2 - glavni regulacijski krug (vanjski regulacijski krug, nadređeni regulacijski krug)

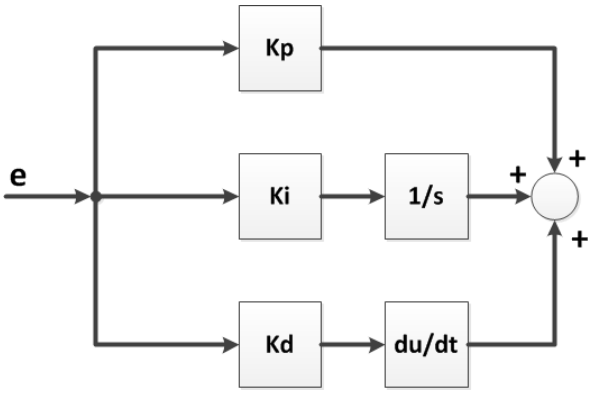


Slika 36: Blokovska shema jednopetljaste strukture upravljanja



Slika 37: Blokovska shema kaskadne strukture upravljanja

U takvoj je strukturi glavni regulator G_{r2} ne djeluje neposredno na sustav (bespilotnu letjelicu) nego tvori referentnu (vodeću) veličinu za podređeni regulator G_{r1}. Takva struktura upravljanja pokazuje određene sličnosti sa strukturom upravljanja zasnovanoj na varijablama stanja - za optimalni regulator stanja postoje povratne veze za sva stanja. Regulatori smješteni u pomoćnom i glavnom regulacijskom krugu uobičajeni su PID regulatori. PID regulator koristi tri osnovna ponašanja: P-proporcionalno, I-integracijsko te D-derivacijsko. Blokovska shema takvog regulatora prikazana je na slici 38.



Slika 38: Blokovska shema PID regulatora

Radi jednostavnijeg upravljanja implementirano je automatsko slijetanje i uzlijetanje letjelice. Uzlijetanje letjelice zamišljeno je na način da se upali regulacija pozicije, postavi referentna vrijednost visine, $z - osi$, na jedan metar te se za početan uvjet integratora regulatora podređene petlje postavi eksperimentalno dobivena vrijednost gasa. Eksperimentalno dobivena vrijednost gasa je ona vrijednost koja je dovoljna da bespilotna letjelica savlada gravitacijsku silu. Slijetanje letjelice osmišljeno

je na način da se referentna veličina visine smanjuje od trenutne visine letjelice do nule u intervalu od $10s$.

Prikazana struktura upravljanja koristi se za regulaciju pozicije i orijentacije letjelice u prostoru te je implementirana u čvoru *PI_D_controller* koji je detaljno opisan u poglavlju 5.5.1.

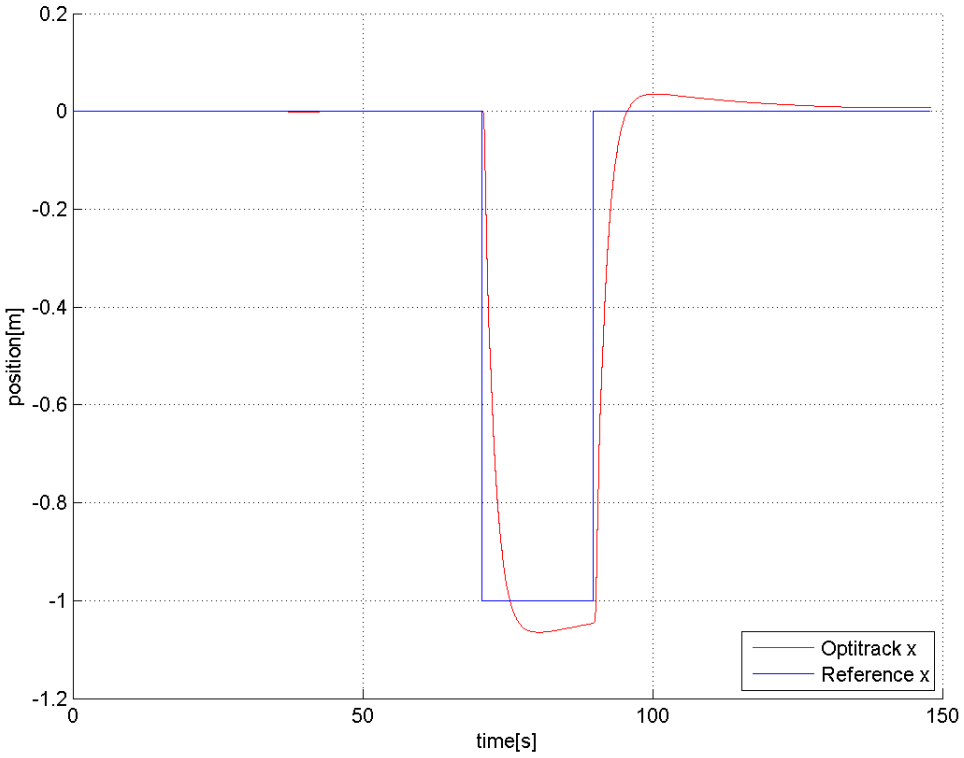
4.1.2. Podešavanje parametara regulatora letjelice u virtualnom okruženju

U virtualnom okruženju korištena je jednopetljasta struktura upravljanja, slika 36. Koristila se povratna veza po poziciji dobivena iz simulacije pri čemu su podešavani parametri jednog PID regulatora. Ugođeni parametri PID regulatora po poziciji te orijentaciji oko $z-osi$ letjelice s dvostrukim manipulatorom dani su tablicom 9:

Tablica 9: Parametri PID regulatora jednopetljaste strukture upravljanja

	x	y	z	yaw
K_P	412	412	170	250
K_I	17	17	15	5
K_D	300	300	100	95

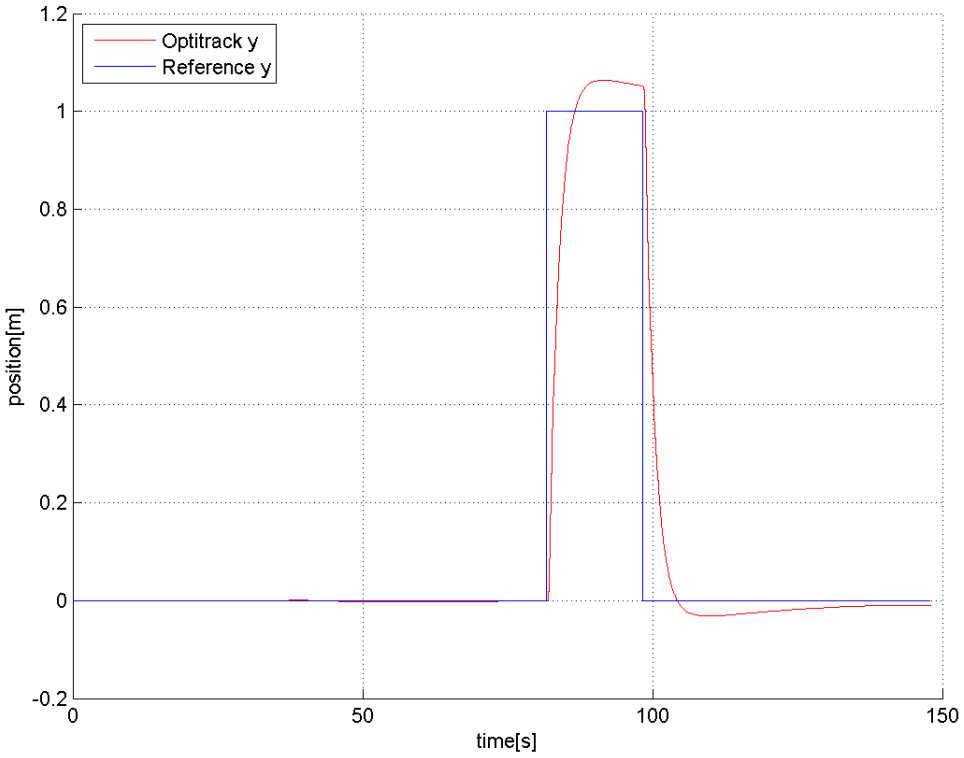
Referentna veličina i odzivi dobiveni jednopetljastom strukturom upravljanja dani su slikama 39-42



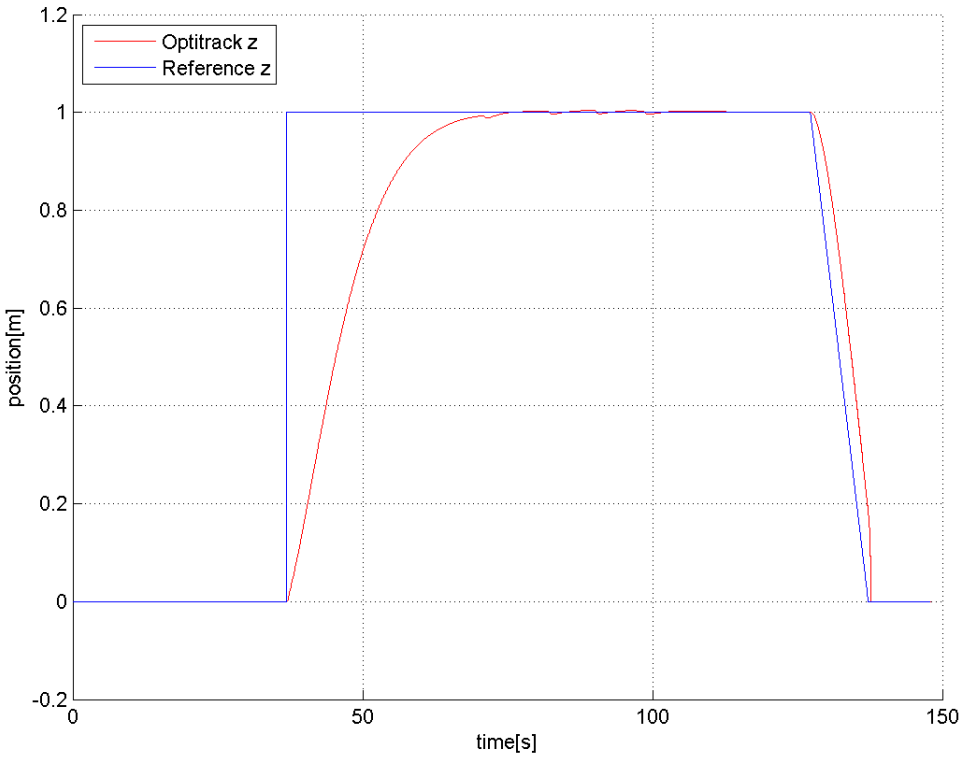
Slika 39: Odziv pozicije na promjenu reference po $x-osi$ uz jednopetljastu strukturu upravljanja

Parametri PID regulatora određeni su eksperimentalno. Na slici 39 vidi se kako odziv veoma dobro prati promjenu referentne veličine te u stacionarnom stanju nema pogreške. Isti zaključak se može izvesti i za $y-os$ jer su parametri regulatora identični. Na slici 41 prikazan je odziv sustava na skokovitu referentnu veličinu duž $z-osi$ gdje sustav također vrlo dobro slijedi referentnu veličinu. Na slici 42 vidi se da u trenutku $t \approx 65s$ i $t \approx 100s$ dolazi do nagle promjene kuta zakreta, što je posljedica promjene referentne veličine $x-osi$ i $y-osi$. Takav je poremećaj regulator u kratkom vremenu uspio kompenzirati.

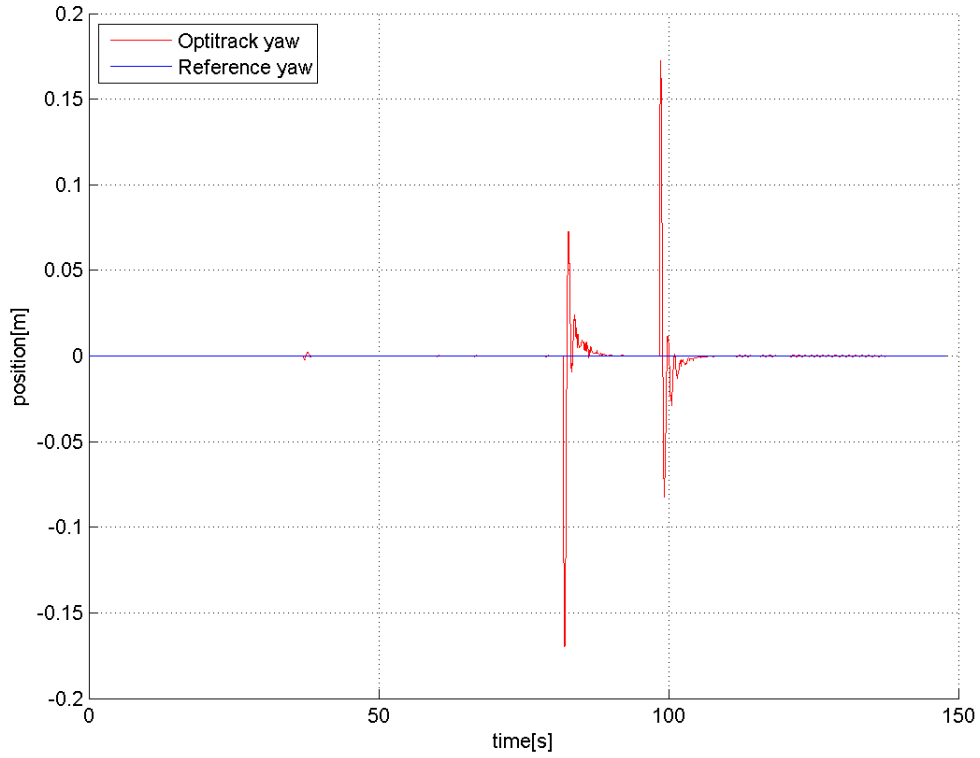
Postignuta je zadovoljavajuća regulacija pozicije i orijentacije jednopetljastom strukturom te nije potrebno određivati parametre regulatora za kaskadnu strukturu upravljanja.



Slika 40: Odziv pozicije na promjenu reference po $y-osi$ uz jednopetljastu strukturu upravljanja



Slika 41: Odziv pozicije na promjenu reference po $z-osi$ uz jednopetljastu strukturu upravljanja



Slika 42: Odziv kuta zakreta *yaw* na promjenu reference uz jednopetljastu strukturu upravljanja

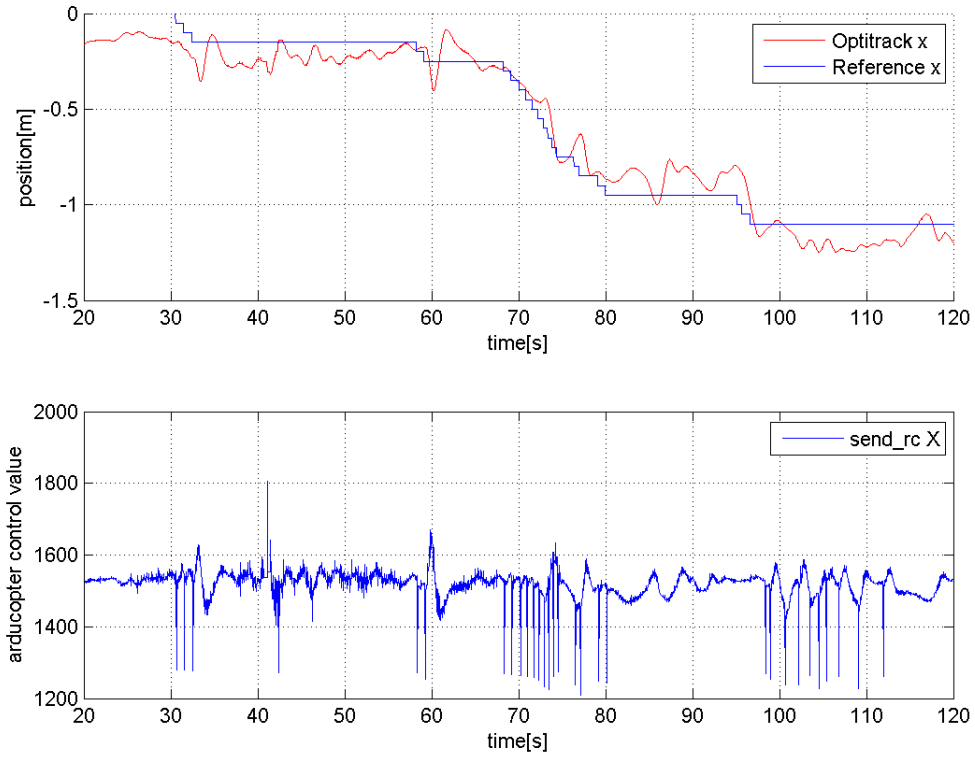
4.1.3. Podešavanje parametara regulatora letjelice u realnom okruženju

Prvotna struktura upravljanja bila je jednopetljasta, slika 36. U tom slučaju koristila se povratna veza po poziciji pomoću OptiTrack sustava pri čemu su podešavani parametri jednog PID regulatora. Parametri ugođenih PID regulatora po poziciji te orijentaciji oko $z-osi$ letjelice s dvostrukim manipulatorom dani su tablicom 10:

Tablica 10: Parametri PID regulatora jednopetljaste strukture upravljanja

	x	y	z	yaw
K_P	275	305	110	400
K_I	10	10	10	0
K_D	240	240	80	0

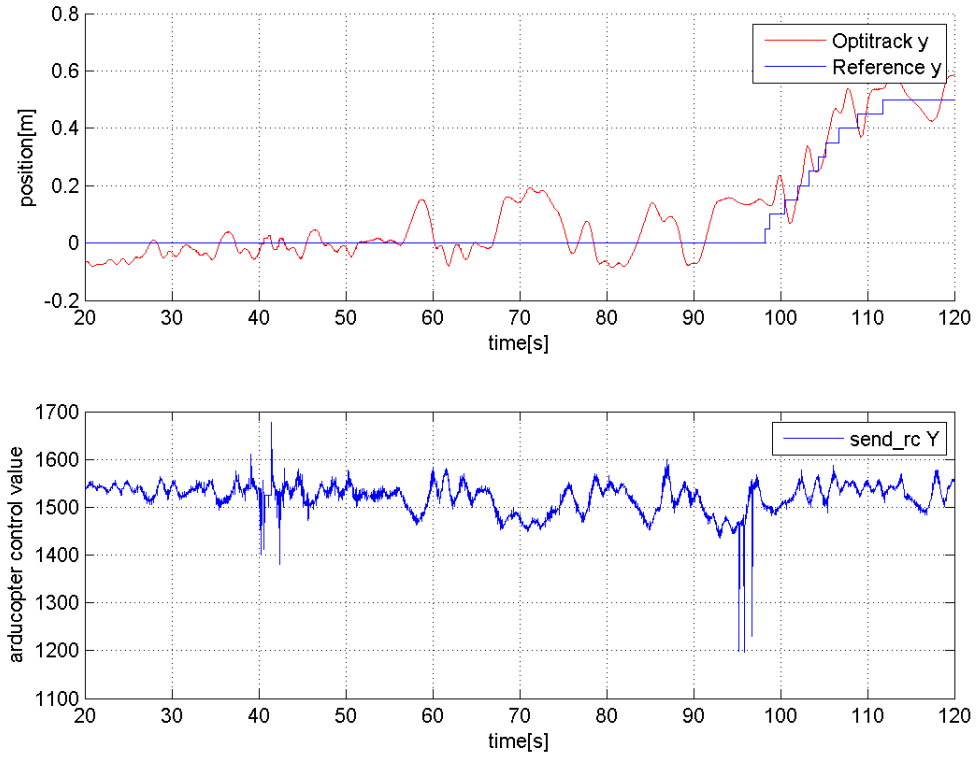
Odzivi dobiveni jednopetljastom strukturom upravljanja dani su slikama 43 - 46.



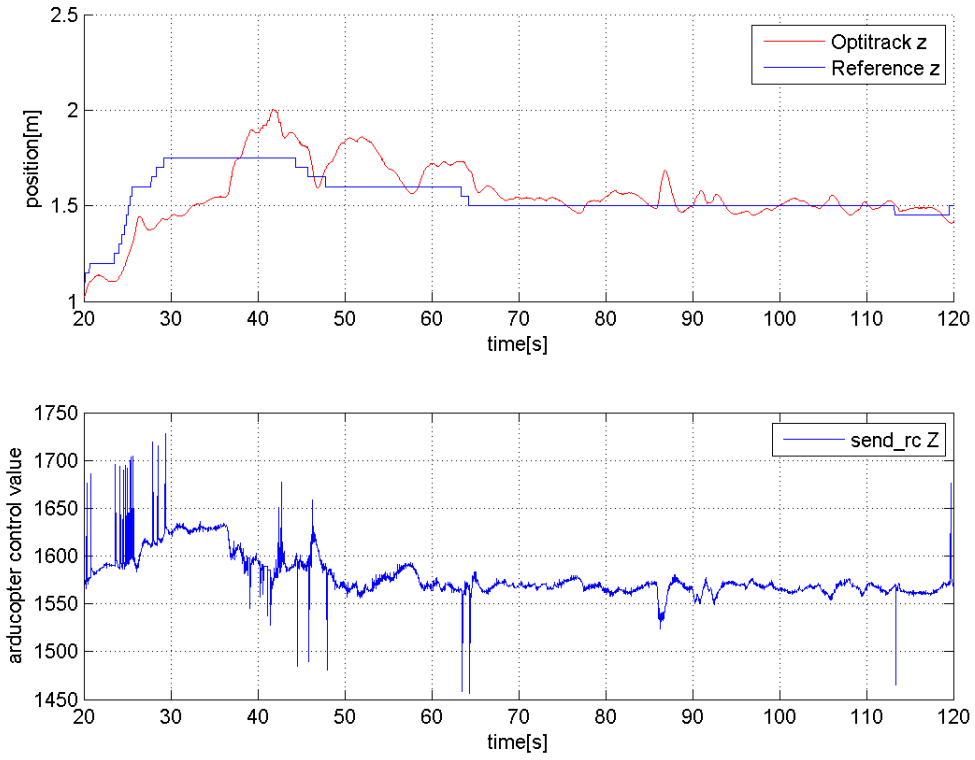
Slika 43: Odziv pozicije na promjenu reference po $x-osi$ uz jednopetljastu strukturu upravljanja

Parametri PID regulatora određeni su eksperimentalno. Kod ugađanja se prvo parametrirala P komponenta regulatora kako bi postigla željena dinamika. D komponenta služi za stabilizaciju sustava jer reagira na odstupanje od reference dok I komponenta otklanja pogrešku u stacionarnom stanju. Na priloženim odzivima prikazana je pozicija i orijentacija letjelice u prostoru te reference željene pozicije i orijentacije. Na donjem grafu prikazana je upravljačka veličina koja se šalje na letjelicu. Na slici 43 može se vidjeti kako letjelica ima malen odmak od reference u ustaljenom stanju, primjerice u vremenskom intervalu $t \in 35s - 55s$. Taj odmak javlja se ponajviše zbog kabla napajanja pričvršćenog na letjelicu. Potreba za kablom napajanja pokazala se opravdanom jer se prelaskom na baterije vrijeme leta uvelike smanjilo pa nije bilo moguće ispitati karakteristike regulacije. Odziv veoma dobro prati promjene reference što znači brz dolazak u željenu poziciju odnosno brzu dinamiku letjelice. Problem se javlja kod upravljačke veličine na čijem se odzivu vide česti "šiljci". Uzrok takvim skokovima upravljačke veličine je D komponenta regulatora koja uzima u obzir trenutnu i prethodnu te razliku dijeli s vremenom uzorkovanja. U trenutcima kada OptiTrack izgubi praćenje letjelice zadržava se stara vrijednost pozicije. Kako se letjelica s vremenom pomakne dolazi do veće razlike između trenutne i prethodne pozicije što je glavni uzrok velikih skokova upravljačke veličine. Za ostale osi mogu se izvesti isti zaključci iako se parametri regulatora razlikuju.

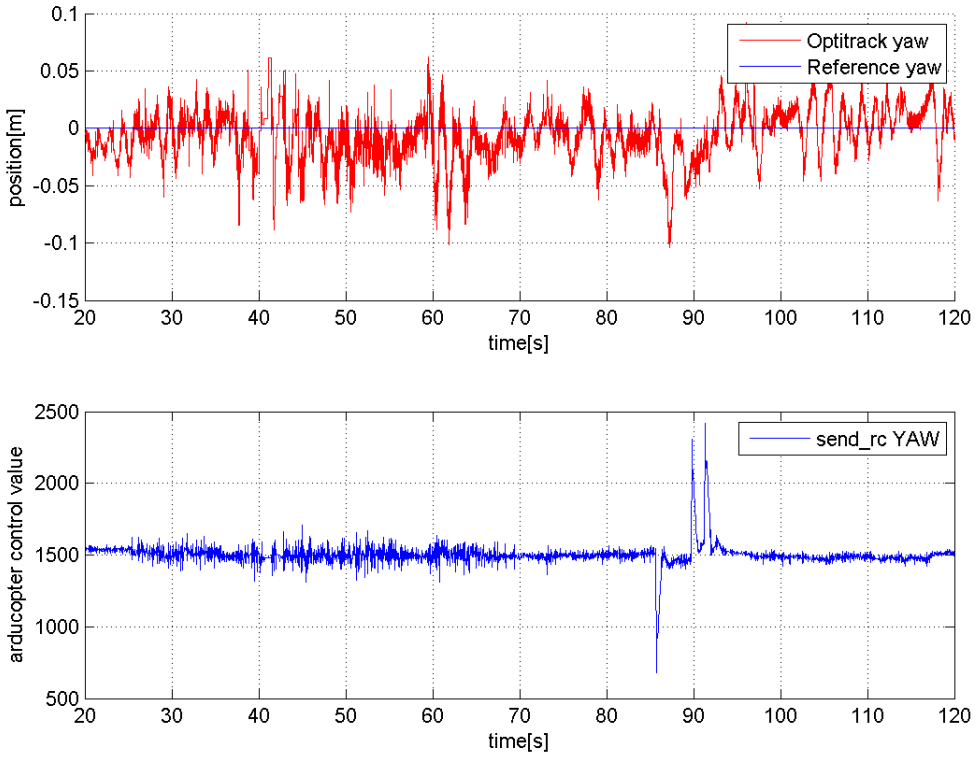
Da bi se postiglo što bolje vladanje, bez skokova upravljačke veličine, a samim time i bez velikih zahtjeva na sustav uvodi se dvopetljasta struktura upravljanja prikazana slikom 37. Unutarnji PID regulator, G_{r1}, zatvoren je u petlji po brzini, a vanjski PID regulator, G_{r2}, zatvoren je po poziciji. Parametriranje regulatora kreće od unutarnje petlje uz ugašenu vanjsku petlju. Referenca brzine ref_{vx}



Slika 44: Odziv pozicije na promjenu reference po $y-osi$ uz jednopetljastu strukturu upravljanja



Slika 45: Odziv pozicije na promjenu reference po $z-osi$ uz jednopetljastu strukturu upravljanja



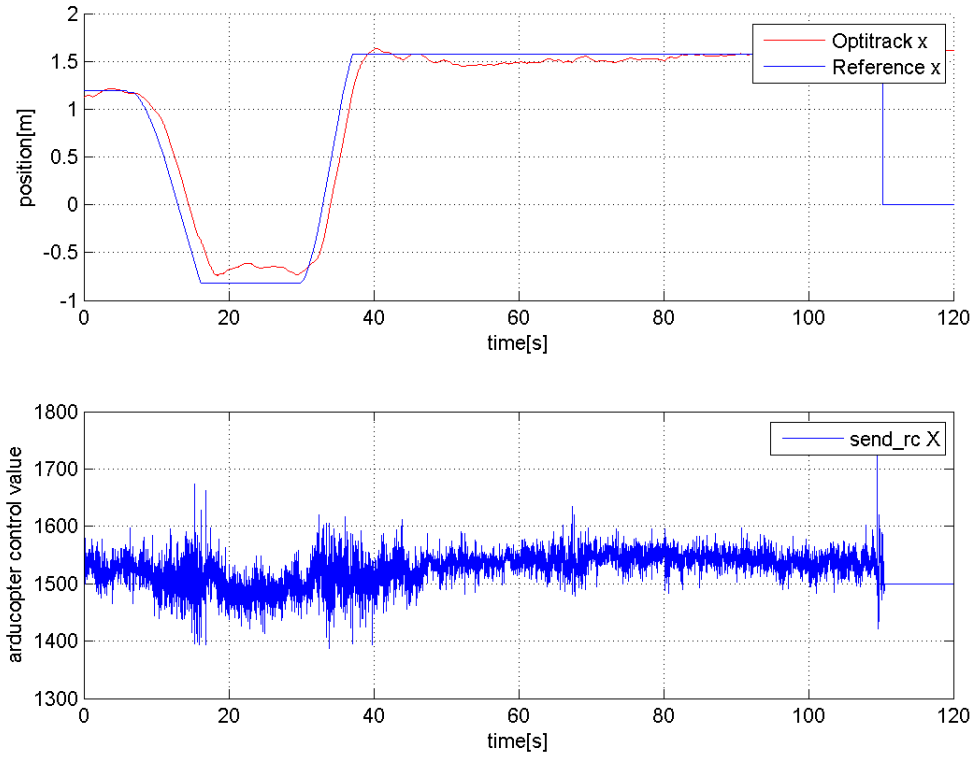
Slika 46: Odziv kuta zakreta *yaw* na promjenu reference uz jednopetljastu strukturu upravljanja

u ovom slučaju jednaka je nuli kako bi se unutarnjom petljom postiglo da letjelica zadržava poziciju, što je posljedica brzine jednake referentnoj. Nakon podešene unutarnje petlje letjelica nije lebdjela na mjestu zbog malih pomaka uslijed nesavršenosti konstrukcije no postignuta je zadovoljavajuća dinamika te se krenulo s parametriranjem vanjske petlje. Parametriranjem vanjskog PID regulatora postiglo se zadržavanje referentne pozicije. Parametri kaskadne regulacije dani su tablicom 11 pri čemu su K_P, K_I i K_D parametri vanjskog PID regulatora, a K_{Pv}, K_{Iv} i K_{Dv} parametri unutarnjeg PID regulatora.

Tablica 11: Parametri regulatora uz kaskadnu regulaciju za letjelicu *3DRobotics Arducopter*

	x	*y*	*z*	*yaw*
K_P	0.7	0.7	110	400
K_I	0	0	10	10
K_D	0.14	0.14	80	70
K_{Pv}	220	220	1	1
K_{Iv}	8	8	0	0
K_{Dv}	21	21	0	0

Odzivi dobiveni dvopetljastom strukturom upravljanja dani su slikama 47 - 50.

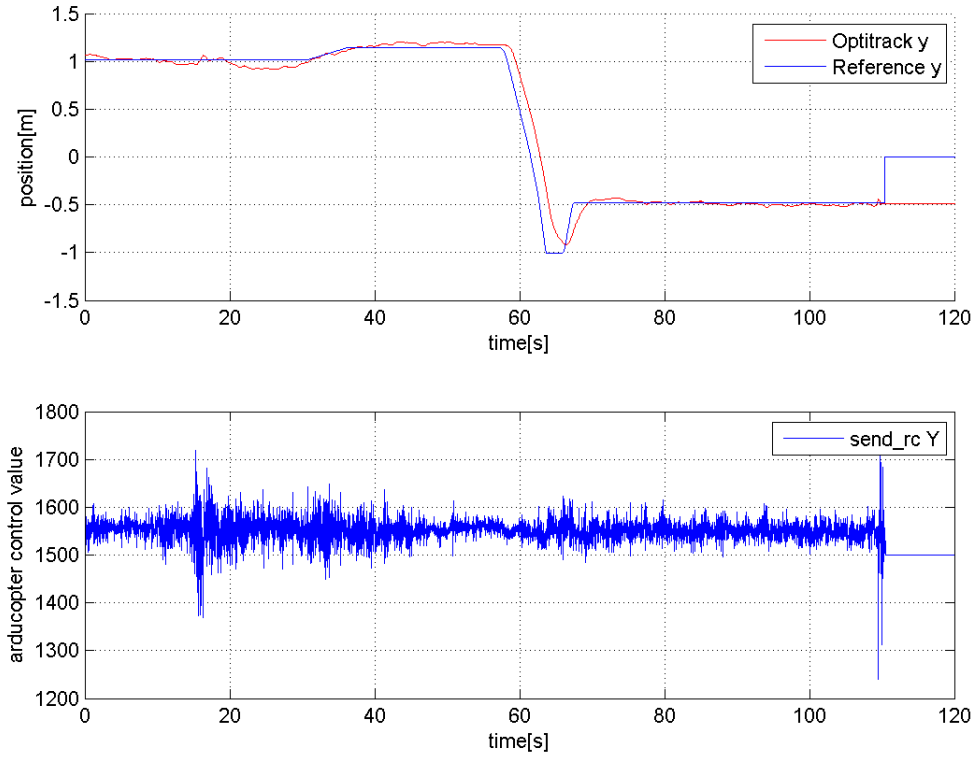


Slika 47: Odziv pozicije na promjenu reference po $x-osi$ uz dvopetljastu strukturu upravljanja

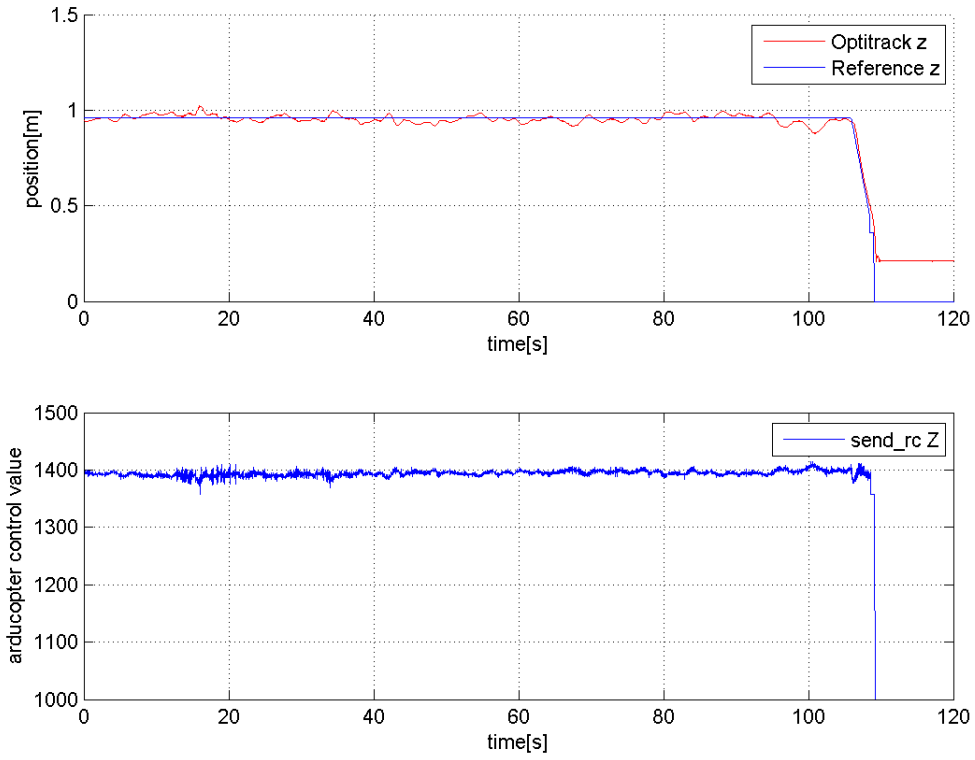
Na slici 47 prikazan je odziv pozicije po $x-osi$ uz dvopetljastu strukturu upravljanja. U stacionarnom stanju odziv dobro prati referencu, no postoji i mali odmak. Taj odmak, kao i kod dvopetljaste strukture, je posljedica kabla koji se ponaša kao konstantna smetnja. S vremenom taj odmak ispravlja I komponenta regulatora. Također, dinamika letjelice je zadovoljavajuća jer veoma dobro prati referentnu veličinu pozicije. Upravljačka veličina nema tolike skokove kao kod jednopetljaste strukture. Razlog takvom ponašanju je nešto drugačiji derivacijski član regulatora. U ovom slučaju diferencira se brzina letjelice, a ne razlika referentne i trenutne veličine regulatora zbog čega nema većih skokova upravljačke veličine.

Iz odziva 43 i 47 mogu se vidjeti dva načina zadavanja referentne pozicije letjelice: skokovita pobuda i rampa. Uz skokovitu promjenu potrebno je nekoliko puta promijeniti referencu kako bi letjelica došla u željenu poziciju ili zadati koordinate željene poziciju. Uz promjenu reference rampom lakše je upravljati letjelicom, a samim time i pozicionirati se u prostoru te je zato odabran ovaj način zadavanja reference.

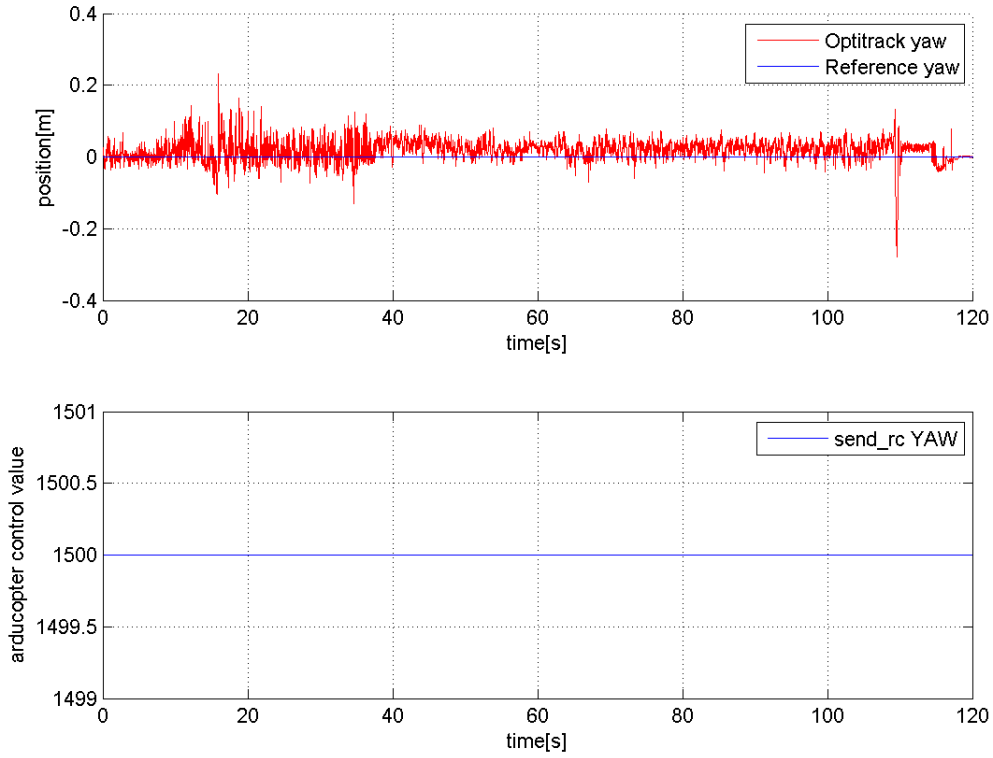
Za potrebe upravljanja letjelicom, na temelju priloženih odziva pozicije uz jednopetljastu i dvopetljastu strukturu upravljanja, odabrana je dvopetljasta struktura upravljanja. Dinamika je u oba slučaja zadovoljavajuća jer letjelica dobro prati referentnu veličinu. U stacionarnom stanju, u oba slučaja, odmak od referentne pozicije nije velik pa obje strukture zadovoljavaju i taj kriterij. Glavni razlog odabira dvopetljaste strukture upravljanja su manji i rjeđi skokovi upravljačke veličine. Naime, skokom upravljačke veličine letjelica veoma naglo mijenja kut zakreta oko određene osi odnosno do-



Slika 48: Odziv pozicije na promjenu reference po $y-osi$ uz dvopetljastu strukturu upravljanja



Slika 49: Odziv pozicije na promjenu reference po $z-osi$ uz dvopetljastu strukturu upravljanja



Slika 50: Odziv kuta zakreta *yaw* na promjenu reference uz dvopetljastu strukturu upravljanja

lazi do "trzaja" kod upravljanja što nikako nije poželjno ponašanje.

Strukturna razlika u parametrima vidljiva je kod PID regulatora za $z-os$. U tom slučaju je unutarnji PID regulator podešen samo s proporcionalnim članom $K_p = 1$. Također, u ovom slučaju programski se isključuje povratna veza po brzini čime se upravljanje svodi na jednopetljastu strukturu. Isti princip vrijedi za kut zakreta *yaw* koji također koristi jednopetljastu strukturu upravljanja. U daljnjim eksperimentima uključeno je rasprezanje, jednadžba (6), te iz tog razloga nije bilo potrebe za uključivanjem regulatora za orijentaciju oko $z-osi$. Rasprezanjem se koordinati sustav letjelice trensformira u koordinatni sustav OptiTracka. Upravljačka veličina za kut zakreta *yaw*, slika 50, je iz tog razloga cijelo vrijeme konstantna.

$$R_z(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Također, za potrebe rada koristila se i letjelica jDrones Arducopter, slika 18. Parametri PID regulatora također su eksperimentalno određeni, tablica 12, uz dvopetljastu strukturu upravljanja.

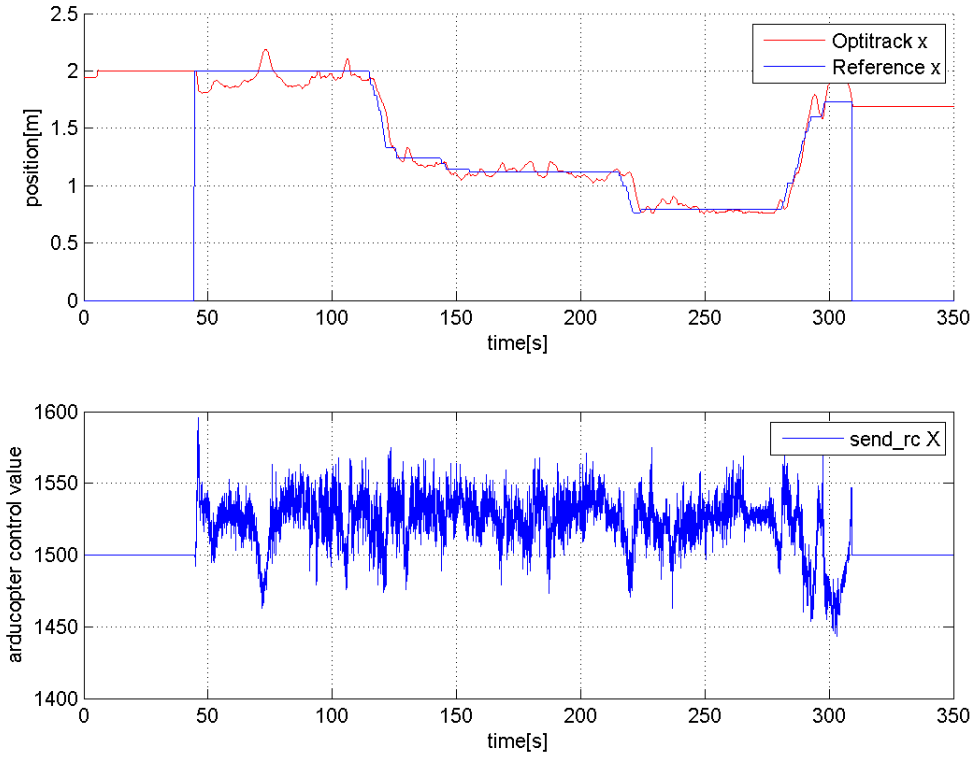
Kako uzlijetanje i slijetanje zahtijevaju mnogo pažnje, izrađene su rutine *takeoff* i *land* kojima operater može jednostavno uzletjeti i sletjeti.

Tablica 12: Parametri regulatora uz kaskadnu regulaciju za letjelicu *jDrones Arducopter*

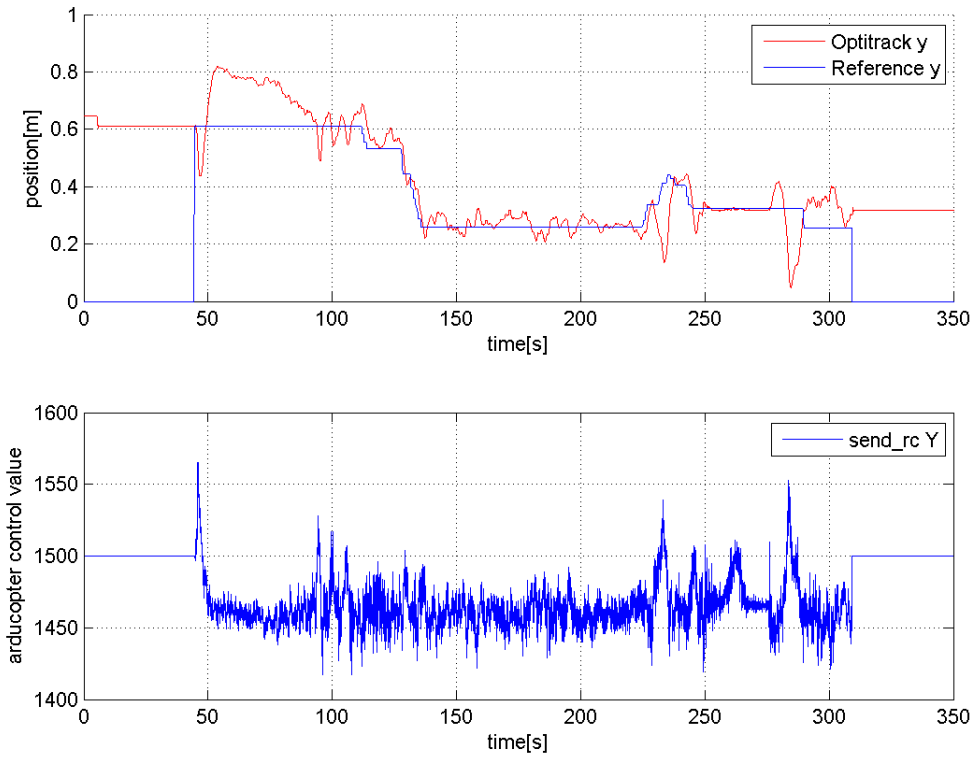
	x	y	z	yaw
K_P	1.7	1.9	115	400
K_I	0.01	0.015	10	10
K_D	0.35	0.35	80	70
K_{Pv}	160	160	1	1
K_{Iv}	8.5	8.5	0	0
K_{Dv}	12	12	0	0

Rutina *takeoff* pokreće se istoimenom glasovnom naredbom nakon čega se letjelica autonomno podiže na visinu od $z = 1m$. To se postiže tako da se na I komponentu, pri uključenju regulatora, postavlja eksperimentalno određena vrijednost iznosa 1400 za letjelicu s dvostrukim manipulatorom te 1320 za letjelicu s kamerom. Odzivi uzlijetanja letjelice s dvostrukim manipulatorom prikazani su na slikama 51 - 53.

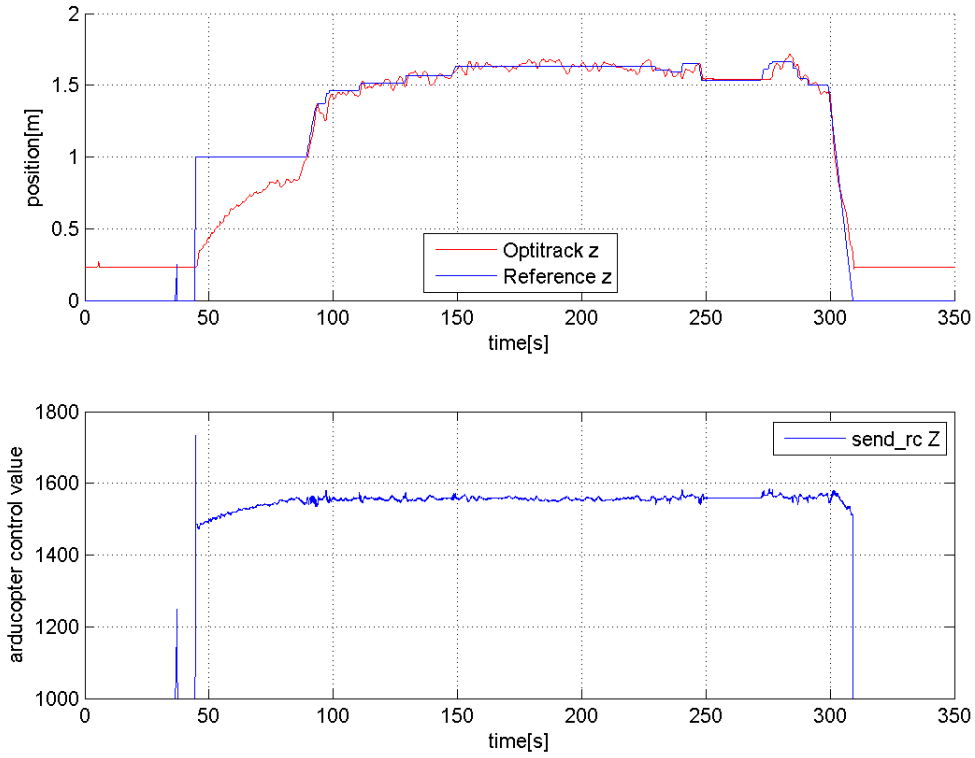
Odzivom 53 prikazane su rutine polijetanja i slijetanja. Naredba za polijetanje dana je nešto prije trenutka $t = 50s$. U tom trenutku dolazi do naglog skoka upravljačke veličine za $z - os$ te polijetanje traje $t_{polijetanje} \approx 40s$. Nakon uspješnog polijetanja letjelica prati prati upravljačke naredbe te u trenutku $t \approx 300s$ dolazi naredba za slijetanje. Rutina slijetanja traje $t_{slijetanje} \approx 10s$ što osigurava kontinuirano smanjenje visine te sigurno i meko prizemljenje što smanjuje izglede za oštećenje letjelice i dvostrukog manipulatora. Po osima x i y se prilikom polijetanja primjećuju odmaci od reference. Takvi odmaci posljedica su niskog leta pri kojem zračne struje otpuhuju letjelicu (eng. *ground effect*). Nakon postizanja dovoljne visine, $z > 0.8m$, letjelica se smiruje po svim osima jer nema poremećaja uzrokovanih zračnim strujama.



Slika 51: Prikaz odziva po $x-osi$ prilikom polijetanja



Slika 52: Prikaz odziva po $y-osi$ prilikom polijetanja



Slika 53: Prikaz odziva po $z-osi$ prilikom polijetanja

4.2. Prošireno korisničko sučelje

U ovom poglavlju detaljnije su objašnjeni elementi proširenog korisničkog sučelja: zadavanje referentne veličine putem glasovnih naredbi i igraće palice te upravljanje dvostrukim robotskim manipulatorom putem Kinecta.

Realizirano prošireno upravljčko sučelje potrebno je testirati na nekom složenom problemu. Odabran je problem otvaranja/zatvaranja ventila iz razloga što je na taj način sustav u potpunom kontaktu s okolinom te je operater u mogućnosti koristiti sve dijelove proširenog korisničkog sučelja.

4.2.1. Prepoznavanje govora

Mogućnost prepoznavanja govora u realnom vremenu otvara široki spektar mogućnosti istraživanja interakcije između čovjeka i računala. Prepoznavanje govora danas se obavlja na mnogo načina, s dobro razvijenim i dobro uhodanim algoritmima, no ni jedna metoda još nije dovedena do vrhunca, sa 100%-tnim prepoznavanjem.

U radu je korišten CMU Pocketsphinx, *open source* alat za prepoznavanje govora, koji je najprikladniji za implementaciju u ROS-u. CMU Pocketsphinx koristi GStreamer kako bi omogućio zaustavljanje i pokretanje prepoznavanja govora te automatski podijelio izgovorene rečenice u riječi,

naredbe.

Glasovno upravljanje implementirano je u čvoru *voice_control_node.py*, a prepoznavanje glasovnih naredba u čvoru *voice_recognition*. Čvorovi su opisani u poglavlju 5.1.2 i 5.2 respektivno. Definiran je skup naredbi koji će biti intuitivan i lako pamtljiv. Takav skup naredbi dan je u poglavlju 5.1.2.2. Naredbama operater može odabirati letjelice, postaviti letjelice u pripravnost za let ili u stanje bez mogućnosti leta, polijetati i slijetati te mijenjati referentne pozicije letjelica. Kao ulaz u čvor za upravljanje bespilotnom letjelicom putem glasovnih naredbi zadaju se imena svih letjelica kojima se želi upravljati. Operater izgovorom imena letjelice odabire letjelicu te ga odabrana letjelica sluša sve dok se ne izgovori naredba koja označava kraj razgovora s letjelicom. Moguće je dati proizvoljno ime letjelici, ali to ime mora razumjeti čvor za prepoznavanje govora.

4.2.2. CMU Pocketsphinx

Pocketsphinx je javno i besplatno dostupna programska biblioteka razvijena na CMU (*Carnegie Mellon University*), čija je središnja biblioteka SphinxBase. Programska biblioteka pisana je u C jeziku i radi na Linux i Windows operacijskim sustavima. Koristi tehnike brzog pretraživanja [4] [5] koje su istražene od CMS-a (*Carnegie Mellon Speech group*).

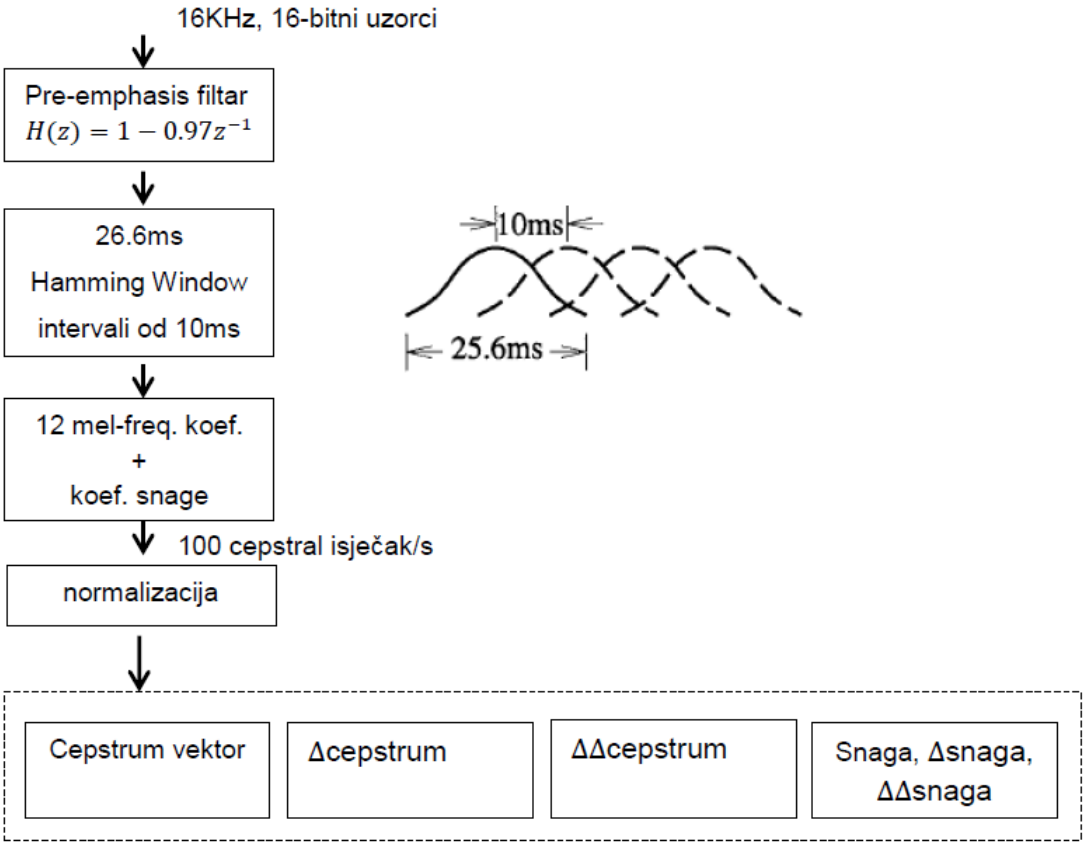
Postoje razni pristupi, tehnike i metode prepoznavanja govora koje su najčešće zasnovane na algoritmima skrivenih Markovljevih modela (eng. *Hidden Markov Models, HMM*), DTW (eng. *Dynamic Time Warping*), neuronskim mrežama i njihovim hibridnim rješenjima. Sphinx koristi tehniku statističkog modeliranja zasnovanu na algoritmima HMM-a. Postoje tri modela koja se koriste kod prepoznavanja govora. To su akustični model, fonetski rječnik te jezični model. Akustični model sadržava akustična svojstva svakog fonema. Fonetski rječnik sadržava samo riječi koje želimo da sustav razumije i objašnjenja od kojih se fonema sastoji koja riječ, a služi kako bi se suzila pretraga. Jezični model također se koristi kako bi se ograničila pretraga te definira koja riječ bi se mogla nadovezati prethodno prepoznatoj riječi, čime se bitno skraćuje proces pretraživanja. Izgovorena riječ se predobrađuje, izrađuje se model te nakon toga uspoređuje sa svim poznatim modelima.

Kako bi se govor pretvorio u tekst potrebno je izraditi modele za riječi. Jedan od ključnih problema u akustičnom modeliranju je odabir jedinice govora. U malim sustavima, nekoliko desetaka riječi, moguće je izraditi model za cijelu riječ, no kako rječnik raste takav pristup postaje neisplativ. Neophodno je prikazati riječ kao skup jedinica te trenirati akustični model tako da se kod izgovora neke nove riječi, ista može opisati već modeliranim jedinicama. Potrebno je sakupiti nekoliko uzoraka svake riječi od različitih govornika kako bi se izgradio model koji je neovisan o govorniku. Fonem je najčešće prihvaćena jedinica te se svaki fonem modelira algoritmima HMM-a. Prirodni govor za razliku od pisanog ne slijedi strogo pravilnu strukturu. Pojedine riječi ne izgovaraju se zasebno već u nepravilnim nizovima, mnoge riječi zvuče slično, stoga izgovor varira ovisno o različitim utjecajima (različite boje glasa među govornicima, neujednačena intonacija i brzina govora ovisna o emocijama govornika i sl.). Svaki fonem je pod utjecajem prethodnog i sljedećeg fonema. Kod malih rječnika u kojima ne postoje riječi koje se slično izgovaraju to ne predstavlja problem, no kod razmjerno velikih rječnika to nije slučaj. Mnogi sustavi upotrebljavaju skup od tri ili čak četiri fonema kako bi se nosili

s takvim problemima. Fonemi se modeliraju na način da se prvo uzorkovani ulazni signal predobrađuje različitim koracima za obradu signala u cepstrum koji sadržava jedan vektor značajki svaki segment, najčešće u trajanju od 10 milisekundi. Cepstrum je rezultat inverzne Fourierove transformacije (IFT) primijenjene na logaritmiranoj Fourierovoj transformaciji orginalnog signala (FT).Na slici (54) prikazani su koraci obrade signala. Protok 16-bitnih uzoraka glasa uzorkuje se frekvencijom od 16KHz te se svaki segment trajanja 10 milisekundi pretvara u MFC vektore (eng. *mel-scale frequency cepstrum*), sastavljene od 12 elemenata i koeficijente snage. Cepstrum vektor i vektor snage se normaliziraju i stvaraju se 4 vektora značajki za svaki segment. Vektori značajki su normalizirani cepstrum vektor, njegova prva i druga diferencija te vektor snage, $x_0(t)$.

$$\begin{aligned} x(t) &= normaliziranicepstrumvektor \\ \Delta x(t) &= x(t+2) - x(t-2) \\ \Delta\Delta x(t) &= \Delta x(t+1) - \Delta x(t-1) \\ x_0(t) &= x_0(t) \end{aligned} \tag{7}$$

Broj elemenata vektora je redom 12,24,12,3 i oni su u konačnici ulazne varijable sustava za prepoznavanje govora.



Slika 54: Obrada signala

HMM predstavlja niz stanja sustava koja su međusobno povezana. U svakom trenutku sustav

može prijeći u neko novo stanje ili može ostati u istome stanju. Promjene stanja nazivaju se tranzicije. Svakoj tranziciji dodijeljena je funkcija kojom se definira vjerojatnost prelaska iz pojedinog stanja. Jedan tranzicijski model predstavlja jedan segment govora, a stanja su mali potprostor sveukupnog značajnog prostora. Oblik potprostora je dovoljno složen da se ne može točno karakterizirati jednostavnim distribucijama.

Kao što je spomenuto, jezični model je potreban u aplikacijama koje koriste opsežan rječnik, kako bi se suzila pretraga. Jezični model definira a priori vjerojatnost niza riječi. Vjerojatnost rečenice (niza riječi $\omega_1, \omega_2, \ldots, \omega_n$) je dana s:

$$P(\omega_1)P(\omega_2|\omega_1)P(\omega_3|\omega_1,\omega_2)P(\omega_4|\omega_1,\omega_2,\omega_3)\cdots P(\omega_n|\omega_1,\ldots,\omega_{n-1}) = \prod_{i=1}^{n} P(\omega_i|\omega_1,\ldots,\omega_{i-1}) \quad (8)$$

U izrazu (8), $\omega_1,\ldots,\omega_{i-1}$ je povijest riječi ili jednostavno povijest od ω_i. U praksi nije moguće dobiti pouzdane vjerojatnosti proizvoljno dugog niza izgovorenih riječi. Te se vjerojatnosti aproksimiraju na sljedeći način:

- Kontekstno neovisna gramatika ili regularna gramatika: takvi jezični modeli koriste se za definiranje forme dobro strukturiranih rečenica ili fraza. Nisu dopuštena odstupanja od propisanih struktura. Budući da su ograničavajuće, formalne gramatike se nikad ne koristi u opsežnim rječnicima.

- N-gram gramatika: unigram, bigram i trigram gramatike definirane su redom (n-grami višeg reda mogu se definirati na sličan način):

  $P(\omega)$ = vjerojatnost pojavljivanja riječi ω

  $P(\omega_j|\omega_i)$ = vjerojatnost ω_j uz uvjet pojavljivanja prethodne riječi ω_i

  $P(\omega_k|\omega_i,\omega_j)$ = vjerojatnost pojavljivanja ω_k uz uvjet pojavljivanja ω_i, ω_j

  Bigram gramatika ne treba sadržavati vjerojatnosti za sve moguće parove riječi, što bi bilo pretjerano za sve aplikacije osim za aplikacije s malim rječnicima. Obično se navode samo najčešći bigrami, te se koristi *backoff* mehanizam kako bi pronašla vjerojatnost unigrama u slučaju da nije pronađen traženi bigram. Drugim riječima, ako je tražena $P(\omega_j|\omega_i)$ i nije pronađena, prelazi se na $P(\omega_j)$.

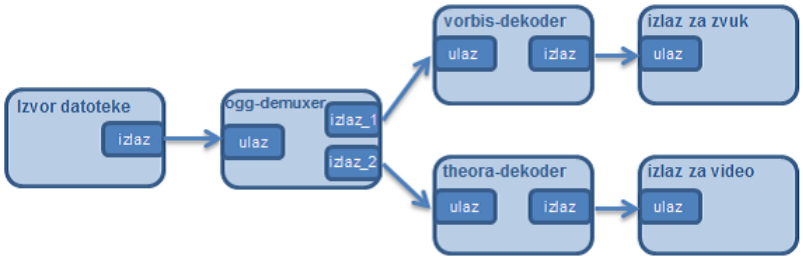
- Klasa n-gram gramatike: n-gram gramatika koja ne mora sadržavati samo *n* riječi već uključuje brojeve, datume, vlastita imena itd. Stvaranje i upotrebljavanje takve gramatike je zahtjevno jer riječi mogu pripadati različitim klasama.

Od svih navedenih gramatika, bigram i trigram gramatika se koriste najčešće iz razloga što se lako treniraju iz velikih količina podataka. Također osiguravaju visoku točnost prepoznavanja govora. Sphinx sustav koristi trigramske jezične modele.

4.2.3. GStreamer

GStremer je alat koji služi za kreiranje aplikacija za protok informacija (eng. *streaming*). Omogućava lako pisanje aplikacija koje se mogu služiti videom ili zvukom. Element je najvažnija klasa

objekta. Obično se kreira lanac elemenata, koji su međusobno povezani, kroz koji teku podatci. Svaki element ima svoju funkciju, bila ona čitanje podataka iz datoteke, dekodiranje ili slanje podataka na zvučnu karticu. Spajanjem nekoliko takvih elemenata stvaramo cjevovod (eng. *pipeline*) koji može obavljati određene funkcije, npr. snimanje videa, prepoznavanje govora i dr. Svaki element ima ulaz (eng. *source pad*) i izlaz (eng. *sink pad*), koji služe za spajanje s ostalim elementima. Ulaz i izlaz se koriste za stvaranje veze i za protok podataka. Oni specificiraju tip podataka koji protječe kroz njih, a njihova veza je dozvoljena samo ako su njihovi tipovi podataka kompatibilni. Elementi GStreamera mogu se grupirati te se takva skupina elemenata naziva kontejner (eng. *bin*). Kontejneri su potklase elemenata. Time se omogućuje da promijenivši stanje kontejnera možemo promijeniti stanje svih elemenata u njemu. Oni također daju povratnu informaciju u slučaju greške ili kraja datoteke. Cjevovod je kontejner najviše razine. On se brine za sinkronizaciju svih elemenata i osigurava sabirnicu za aplikaciju. Jednostavan cjevovod za reproduciranje ogg videa prikazan je na slici 55.



Slika 55: Obrada signala

4.2.4. PlayStation3 Move igraća palica

PlayStation3 Move igraća palica povezuje se s upravljačkim računalom putem Bluetootha. Za čitanje ulaza igraće palice brine se ROS paket *ps3joy* [27]. Paket objavljuje informacije o trenutnim vrijednosti ulaza igraće palice u temu */joy*, a ulazi mogu poprimiti vrijednosti u intervalu od 0 do 1. Igraća palica sadrži 4 analogna i 8 digitalnih ulaza. Upravljanje bespilotnom letjelicom pomoću igraće palice implementirano je u paketu *arducopter_control*, tj. u čvoru *joy_control_arducopter*, koji je detaljno opisan u poglavlju 5.1.1. Definirano je da se PlayStation3 Move igraća palica koristi za sljedeće: gašenje motora letjelice, promjena referentne pozicije/brzine u svim smjerovima, promjena rotacije letjelice, uključenje/isključenje regulacije pozicije.

4.2.5. Praćenje kostura i detekcija poze

Uređaj Kinect najčešće se koristi za praćenje kostura (eng. *skeleton tracking*) i mapiranje prostora. Roboti, neovisno o tome gdje se nalaze, mogu uz pomoć Kinecta stvoriti kartu prostora u kojem se nalaze te se pomoću nje lokalizirati u prostoru. Praćenje kostura omogućuje detekciju pokreta i gesti ljudskog tijela koji se mogu prevesti u upravljačke naredbe za razne robote.

U ovom radu korišteni su *OpenNI* paketi u kombinaciji s *NiTE middleware*. *OpenNI* paketi zaduženi su za komunikaciju između računala i Kinect senzora što podrazumijeva dohvat *RGB* i dubinske

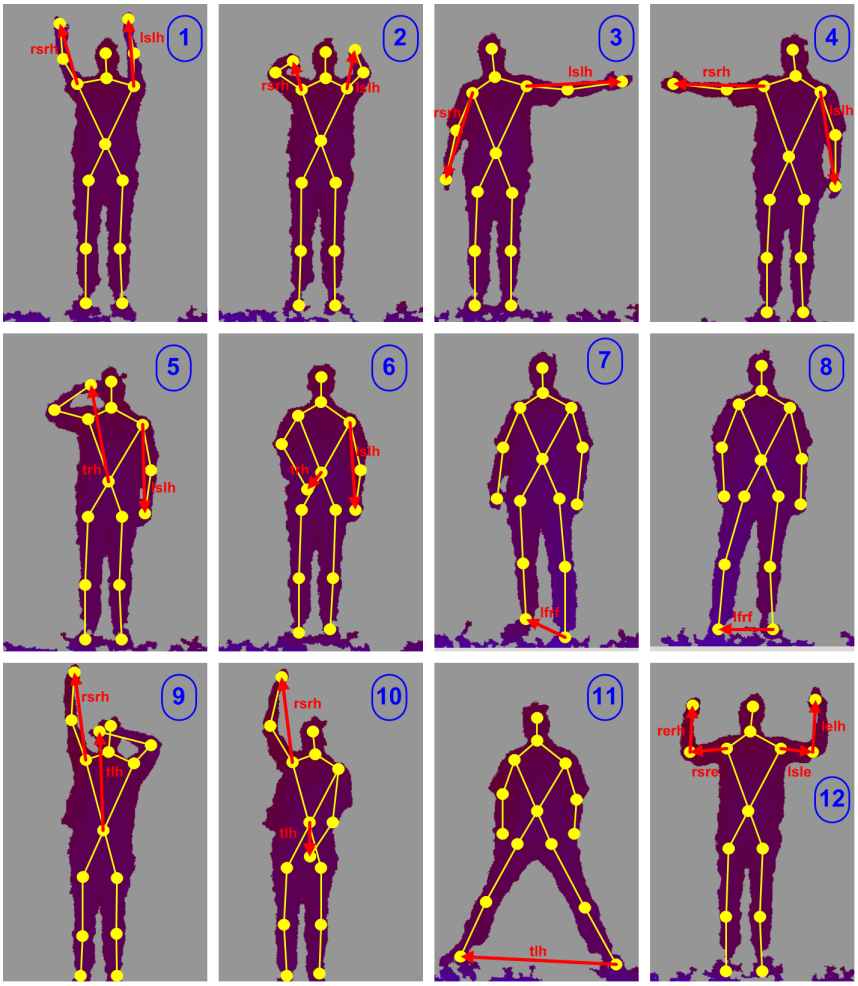
slike te njihovu obradu. Podatci se obrađuju u takozvani oblak točaka (eng. *Point Cloud*) koji sadrži informacije o položaju svakog *pixela* slike u prostoru gledano iz koordinatnog sustava kamere. Na taj način dobiva se trodimenzionalna slika prostora pogodna za daljnju obradu. *NiTE middleware* obrađuje oblak točaka te traži korisnike.

Uz poznate koordinate karakterističnih točaka korisnika dva su pristupa za detekciju poze: čitanje apsolutnih koordinata točaka te pristup pomoću vektora. Čitanje apsolutnih koordinata nije prikladno za implementaciju jer u tom slučaju korisnik uvijek mora biti na istoj relativnoj poziciji od uređaja. Ako se korisnik odmakne od unaprijed definirane pozicije mijenjaju se i koordinate te se iz tog razloga ne detektiraju željeni pokreti. Za detekciju poze korisnika odabrani su vektori definirani karakterističnim točkama ljudskog tijela što pruža robusnu detekciju jer korisnik može upravljati sustavom iz bilo kojeg dijela vidnog polja kamere. U nastavku je dan popis korištenih vektora:

- *rsrh*: Vektor od desnog ramena do desne šake
- *lslh*: Vektor od lijevog ramena do lijeve šake
- *rsre*: Vektor od desnog ramena do desnog lakta
- *lsle*: Vektor od lijevog ramena do lijevog lakta
- *rerh*: Vektor od desnog lakta do desne šake
- *lelh*: Vektor od lijevog lakta do lijeve šake
- *hrh*: Vektor od glave do desne šake
- *hlh*: Vektor od glave do lijeve šake
- *trh*: Vektor od torza do desne šake
- *tlh*: Vektor od torza do lijeve šake
- *lhiplfoot*: Vektor od lijevog kuka do lijevog stopala
- *lhiplknee*: Vektor od lijevog kuka do lijevog koljena
- *lkneelfoot*: Vektor od lijevog koljena do lijevog stopala
- *rhiprfoot*: Vektor od desnog kuka do desnog stopala
- *rhiprknee*: Vektor od desnog kuka do desnog koljena
- *rkneerfoot*: Vektor od desnog koljena do desnog stopala
- *lfootrfoot*: Vektor od lijevog stopala do desnog stopala

Definirani vektori koriste se za detekciju trenutne poze korisnika. Detekcija svake poze podijeljena je na dva dijela. Prvi dio je definiranje karakterističnih vektora koji se uzimaju u obzir pri promatranju poze. Ako se promatraju dvije zrcalno simetrične poze potrebno je obzir uzeti više od jednog vektora kako ne bi došlo do konflikta pri detekciji. Drugi dio detekcije podrazumijeva provjeravanje nalaze li se vektori unutar područja određenog za pojedinu pozu. Ako su pojedini vektori unutar pojedinog područja detektira se odgovarajuća poza korisnika. Također, valja primijetiti kako nisu sve osobe jednake građe što za posljedicu ima različita područja unutar kojih se detektira pojedina poza. Iz tog razloga izrađena je kalibracijska rutina koja omogućava snimanje korisnika u pozi raširenih ruku i u stajaćoj pozi te se na temelju prikupljenih uzoraka spremaju duljine karakterističnih vektora. Spremljene duljine karakterističnih vektora uspoređuju se s duljinama osnovnog korisnika, po kojem su određena područja detekcije, te se na taj način prilagođavaju područja za ostale korisnike. Popis detektiranih poza dan je u nastavku, a slikovni prikaz dan je slikama 56 i 57.

1. *point forward*
2. *point back*
3. *point left*
4. *point right*
5. *right hand high*
6. *right hand low*
7. *step forward left*
8. *step forward right*
9. *rhand up lhand high*
10. *rhand up lhand low*
11. *legs spread*
12. *psi*
13. *left hand high*
14. *left hand mid*
15. *left hand low*
16. *arms spread*
17. *stand straight*
18. *0*



Slika 56: Prikaz poza koje se mogu detektirati

Slika 57: Prikaz poza koje se mogu detektirati

4.2.6. Upravljanje dvostrukim robotskim manipulatorom

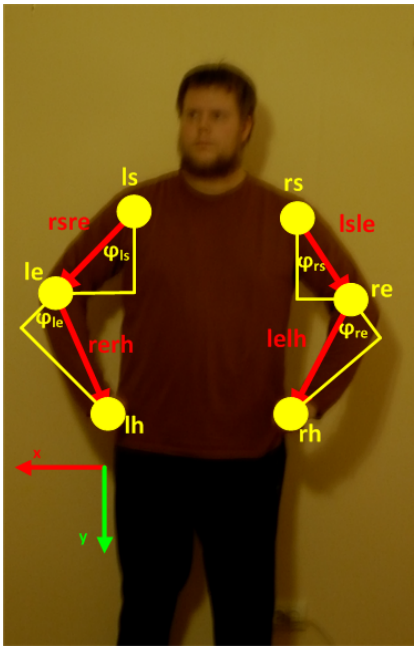
Upravljanje dvostrukim robotskim manipulatorom izvedeno je putem Kinecta. Detekcijom položaja korisnika moguće je izračunati kutove otklona ruku koji se potom preslikavaju na dvostuki manipulator. Pritom su smišljena tri pokreta nogama za promjenu režima rada. Prema slici 56 položaj 8 postavlja manipulator u položaj pogodan za let jer se, prikazan slikom 15b. Osim što je pogodan za let, u ovom položaju se smanjuju rizici od nastanka štete te se iz tog razloga koristi kao položaj(eng. *emergency state*). Položaj 11 omogućuje upravljanje robotskim manipulatorom (eng. *control mode*), a položaj 7 onemogućuje upravljanje čime se zadržava trenutan položaj zglobova (eng. *position lock*). Pokreti nogama za promjene režima rada uvedeni su jer ne ovise o pokretima rukama. Takvo upravljanje implementirano je u čvoru *kinect_arms_controller*, opisanom u poglavlju 5.4.1.

Ideja upravljanja dvostrukim manipulatorom zasniva se na preslikavanju stanja ruku korisnika na sam manipulator. Na taj način postiže se intuitivno upravljanje jer dvostruki manipulator oponaša kretnje korisnika. Na slici 58 može se vidjeti kako zglobovi dvostrukog manipulatora odgovaraju zglobovima korisnika odnosno sastoji se od ramenog zgloba te zgloba lakta i šake. Za kutove zakreta pojedinog zgloba manipulatora dovoljno je proračunati kutove iz poznatih koordinata zglobova ruku korisnika. Formule za izračun kutova ramenih zglobova dane su izrazom (9) te se računaju iz pravokutnih trokuta koje određuju zglobovi ramena i lakta. Kutovi zakreta lakta računaju se prema izrazu (10) odnosno pomoću skalarnog produkta vektora od ramena do lakta i vektora od lakta do šaka.

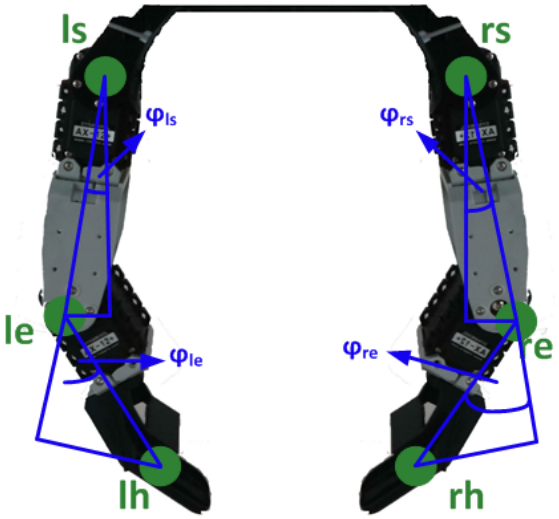
$$\begin{aligned} \varphi_{lijevo\_rame} &= \arctan \frac{-lsle_x}{lsle_y} \\ \varphi_{desno\_rame} &= \arctan \frac{rsre.x}{rsre.y} \end{aligned} \tag{9}$$

$$\begin{aligned} \varphi_{lijevi\_lakat} &= \arccos \frac{\overrightarrow{lsle} \cdot \overrightarrow{lelh}}{\|\overrightarrow{lsle}\| \cdot \|\overrightarrow{lelh}\|} \\ \varphi_{desni\_lakat} &= \arccos \frac{\overrightarrow{rsre} \cdot \overrightarrow{rerh}}{\|\overrightarrow{rsre}\| \cdot \|\overrightarrow{rerh}\|} \end{aligned} \tag{10}$$

Također, veoma je važno obratiti pažnju na radni prostor manipulatora. Ako korisnik izađe iz radnog prostora manipulator mora ostati unutar radnog prostora kako ne bi oštetio nosač ili samu letjelicu. Zbog toga je potrebno provjeravati jesu li kutevi zakreta korisnika unutar radnog prostora te ako nisu postaviti manipulator u graničnu poziciju ovisno položaju korisnika. Iz tog razloga radni prostor ramenih zglobova iznosi 110°, a zglobova lakta 90°.



(a) Zglobovi i kutevi zakreta ruku operatera



(b) Zglobovi i kutevi zakreta dvostrukog manipulatora

Slika 58: Usporedba ruku korisnika i dvostrukog manipulatora

4.3. Problem otvaranja/zatvaranja ventila

Kako bi se testirao rad proširenog korisničkog sučelja osmišljen je scenarij u kojem je potrebno otvoriti/zatvoriti ventil dvostrukim robotskim manipulatorom koji se nalazi na bespilotnoj letjelici. Problem je zamišljen na način da operater mora uzletjeti s bespilotnom letjelicom, doći do ventila, uhvatiti ventil dvostrukim robotskim manipulatorom, otvoriti/zatvoriti ventil te nakon uspješno obavljenog zadatka sletjeti kraj ventila. Zadatak je izveden na dva načina. U prvom načinu operateru je bilo dopušteno direktno gledati u predmet upravljanja i manipulacije, dok u drugom slučaju nije. U drugom slučaju potrebno je bilo uzletjeti drugom letjelicom koja ima montiranu kameru te s njom snimati ventil, a bespilotnom letjelicom s dvostrukim robotskim manipulatorom obaviti zadatak otvaranja/zatvaranja ventila. Operateru je bilo dopušteno gledati samo u slike s kamera bespilotnih letjelica. Ovaj zadatak može se podijeliti na dva dijela. U prvom dijelu testirano je prošireno korisničko sučelje u virtualnom okruženju, a u drugom dijelu na realnim uvjetima. Potrebno je bilo postaviti kontrole igraće palice (kalibracija), izraditi virtualno okruženje, modelirati komponente sustava 3D alatima (bespilotne letjelice, ventil, dvostruki robotski manipulator ...). Potrebno je parametrirati regulatore, odabrati strukturu upravljanja te nakon uspješno obavljenog zadatka u simulatoru krenuti na ispitivanje u realnim uvjetima. Izrađen je ventil, uvedene su potrebne tehničke modifikacije bespilotnih letjelica.

Slika 55 prikazuje PlayStation3 Move igraću palicu na kojoj su tipke označene brojevima. Kontrole igraće palice odabrane su na sljedeći način:

1. sigurnosni gumb (gašenje motora letjelice),
2. promjena referentne pozicije/brzine u smjeru pozitivne $y-osi$
3. promjena referentne pozicije/brzine u smjeru negativne $x-osi$
4. rotacija letjelice oko z osi u negativnom smjeru
5. Pritisak tipke: uključenje regulacije pozicije i orijentacije

   Pomak naprijed: povečanje gasa

   Pomak natrag: smanjenje gasa
6. rotacija letjelice oko $z-osi$ u pozitivnom smjeru
7. promjena referentne pozicije/brzine u smjeru negativne $y-osi$
8. promjena referentne pozicije/brzine u smjeru pozitivne $x-osi$
9. promjena referentne pozicije u smjeru pozitivne $z-osi$
10. promjena referentne pozicije u smjeru negativne $z-osi$

Slika 59: PS3 Move igraća palica

5. Tehnička izvedba

U ovom poglavlju detaljno je objašnjena implementacija proširenog korisničkog sučelja i upravljanja bespilotnom letjelicom te dvostrukim robotskim manipulatorom.

5.1. Paket arducopter_control

Paket *arducopter_control* koristi se za upravljanje ArduCopter bespilotnom letjelicom. Paket sadrži čvor za upravljanje bespilotnom letjelicom putem igraće palice (*joy_control_arducopter*) i čvor za upravljanje bespilotnom letjelicom putem glasa (*voice_control_node.py*). Moguće je postići kombinirano upravljanje putem glasa i igraće palice na način da se izlazi čvora *voice_control_node.py* povežu s ulazima čvora *joy_control_arducopter*. Paket koristi ROS i pisan je u programskom jeziku Python i C++ te radi na linux operativnom sustavu.

5.1.1. Čvor joy_control_arducopter

Primanje podataka s igraće palice te izračun odgovarajuće referentne veličine, provjera ispravnosti dobivenih podataka o položaju letjelice u prostoru glavne su zadaće *joy_control_arducopter* čvora. Čvor je neovisan o izboru igraće palice zbog implementirane kalibracije kojom se podešavaju kontrole upravljanja za pojedinu igraću palicu. Kontrole se mogu podesiti prije samog upravljanja, što je i potrebno napraviti prilikom prvog pokretanja čvora kako bi se stvorila kalibracijska datoteka koja je preduvjet za rad čvora. Kalibracija se pokreće predajom argumenta *-calibrate 1*. Ujedno je potrebno predati lokaciju tekstualne datoteke gdje će se spremiti parametri kalibracije. Proces započinje ispisom poruke u terminalu: "Calibration has started!". Nakon toga slijede upute kojih se valja pridržavati kako bi kalibracija bila uspješna. Pri završetku, ukoliko nije bilo grešaka, ispisuje se poruka u terminalu: "Calibration is completed!". U tom trenutku parametri kalibracije upisuju se u tekstualnu datoteku koja je predana prilikom pokretanja čime su stvoreni svi preduvjeti za rad čvora *joy_control_arducopter*. Ako je predana datoteka prazna ili nepotpuna, kalibracija će se automatski pokrenuti prilikom pokretanja čvora.

5.1.1.1. Ulazno-izlazni podatci

Podatci o stanju igraće palice čitaju se iz teme */joy* koja je tipa poruke *sensor_msgs/Joy*. U toj temi zabilježena je svaka digitalna ili analogna promjena vrijednosti tipke. Ime te teme može se promijeniti predavanjem novog imena preko ROS parametra *joy_topic*. Kombinirano upravljanja putem glasa i igraće palice postiže se predajom ROS parametra *voice_and_joy* vrijednosti 1. Referentna veličina određena glasovnim naredbama prima se u temi *controlValueVoice* tipa poruke *controller/refinv*, a u temu *arm* postavlja se poruka tipa *std_msgs/Empty* ako je letjelicu potrebno postaviti u stanje pripravnosti za let. Ukoliko je odabrano takvo, kombinirano, upravljanje polijetanje i slijetanje letjelice obavlja se autonomno. Informacija o potrebi slijetanja ili uzlijetanja letjelice dobiva se u temama *land* i *takeoff* tipa poruke *std_msgs/Empty*, respektivno. Da bi polijetanje bilo što uspješnije putem ROS parametra *takeoff_value* potrebno je predati vrijednost gasa pri kojem letjelica počinje savladavati gravitacijsku silu. Podatci o poziciji, orijentaciji i brzini letjelice u prostoru primaju se u temi

Optitrack. Poruka u temi je tipa *ACROSS_Optitrack/OptitrackPoseVel*. Ime te teme također se može promijeniti na način da se željeno ime preda preko ROS parametra *mocap_topic*. Stanje bespilotne letjelice (stanje pripravnosti za let ili stanje bez mogućnosti leta) primaju se preko teme *state* tipa poruke *roscopter/State*. Kod pokretanja više instanci čvora, upravljanje s više letjelica, u temi *set_drone* nalazi se poruka tipa *std_msgs/Bool* koja definira je li upravljanje letjelicom dozvoljeno (vrijednost poruke *True*) ili nije dozvoljeno (vrijednost poruke *False*).

Definirani su servis serveri kojim se mijenja referentna vrijednost brzine letjelice, pozicija letjelice u prostoru te servis za generiranje sinusne pobude:

- */reference_value_change* - promjena reference brzine. Servis je tipa *controller/SendFloat*,
- *poseX*, *poseY*, *poseZ* - promijena pozicije letjelice u točku (x,y,z). Servis je tipa *controller/SendFloat*,
- *SineWaveX, SineWaveY, SineWaveZ* - sinusna pobuda. U poruci tipa *contoller/SendSineWave* definira se frekvencija i aplituda te se poziva servis.

Izlazi čvora su dvije teme: *Reference* i *PanicFlag*. Tema *Reference* sadrži referentne vrijednosti pozicije ili brzine, ovisno o tome je li uključena regulacija pozicije, poruke su tipa *controller/refinv*. Druga izlazna tema *PanicFlag*, poruke tipa *std_msgs/Bool*, sadrži informaciju o uključenosti tipke za paniku.

Tipovi poruka *controller/refinv*, *ACROSS_Optitrack/OptitrackPoseVel* i servisa *controller/SendFloat*, *contoller/SendSineWave* detaljnjije su opisanu u [referenca na to], a tip poruke *roscopter/State* u [rdsa].

5.1.1.2. Metode

Prilikom pokretanja čvora mogu se predati argumenti naredbenog retka *-calibrate* i *-file*. Argumenti se koriste kako bi se definirao put do kalibracijske datoteke (*-file*) i postavila zastavica za pokretanje kalibracije (*-calibrate* 1 - pokreće se kalibracija). Ukoliko je kalibracijska datoteka prazna ili nepotpuna, kalibracija se pokreće neovisno o argumentu *-calibrate*. Koraci kalibracije objašnjeni su u poglavlju 5.1.1.

Upravljačka petlja pokreće se metodom *void run()* i izvršava se frekvencijom od $50Hz$. U svakom prolasku kroz petlju provjerava se stanje bespilotne letjelice (stanje pripravnosti za let ili stanje bez mogućnosti leta), stanje panike, stanja tipaka igraće palice. Ako letjelica nije spremna za let ne može se upravljati istom. Pritiskom na odgovarajuću tipku igraće palice poziva se servis *arm* koji šalje zahtjev za postavljanje letjelice u pripravnost za let. Zatim je potrebno provjeriti zastavicu za paniku. Ta zastavica je uključena ukoliko se pritisne odgovarajuća tipka ili prilikom leta u kojem se regulira pozicija letjelice dolazi do primitka zastarjelih podataka o trenutnoj poziciji letjelice u trajanju više od jedne sekunde. Postavljanje zastavice za paniku rezultira postavljanjem letjelice u stanje bez mogućnosti leta slanjem servisa *disarm*.

Metoda *void run()* poziva metodu *void refCalc()* koja provjerava je li uključena regulacija pozicije po pojedinoj osi te ako je uključena računa referentnu veličinu pozicije letjelice. Vrijednosti tipaka igraće palice mogu se kretati između 0 i 1. Referentna veličina pozicije računa se na način da prvo slijedi provjera je li se radi o analognoj ili digitalnoj tipki ukoliko je tipka analogna trenutna vrijednost tipke igraće palice dijeli s frekvencijom upravljačke petlje i zbraja se na već postojeću vrijednost

referentne pozicije, no ukoliko je digitalna trenutnoj referentnoj poziciji zbraja/oduzima se kvocijent vrijednosti referente brzine i frekvencije upravljačke petlje. Vrijednost referentne brzine može se promijeniti pozivanjem servisa */reference_value_change* iz komandne linije ili nekog drugog čvora. Ako je uključena sinusna pobuda tada se na onu poziciji na kojoj se nalazila letjelica u trenutku uključenja zbraja sinusni signal frekvencije i amplitude definirane pozivom servisa *SineWaveX, SineWaveY* ili *SineWaveZ*. Referentna pozicija i *controll* zastavica kontrole objavljuju se u izlaznu temu */Reference*. Kod isključene regulacije pozicije letjelice u izlaznu temu postavlja se umjesto referentne pozicije trenutna vrijednosti određene kontrolne tipke igraće palice.

Stanja kontrolnih tipaka igraće palice osvježavaju se u metodi *void joyCallback(const sensor_msgs::Joy &msg*. U kalibracijskoj datoteci unaprijed određenim redoslijedom upisani su indeksi tipka koje se koriste za upravljanje te se na taj način zna koju je upravljačku aktivnost potrebno dodijeliti određenoj tipki. Kod pritiska tipke koja je postavljena za uključenje/isključenje regulacije pozicije kao referentna pozicija uzima se trenutna pozicija letjelice.

Pozivanje servisa *SineWaveX* pokreće metodu *bool SineWaveSRV_X(controller::SendSineWave::Request &req, controller::SendSineWave::Response &res)* u kojoj se prihvaća amplituda i frekvencija sinusne pobude te se varijabla vremena postavlja na nulu. Identično vrijedi za generiranje sinusne pobude u smjeru y i z osi, gdje se poziva servis i metoda sa sufiksom -Y, -Z, respektivno.

Metoda *bool SetPositionX(controller::SendFloat::Request &req, controller::SendFloat::Response &res)* pokreće se pozivanjem servisa *poseX* kojim se postavlja željena referentna pozicija letjelice na x osi. Kod pozicioniranja letjelice po y i z osi poziva se servis i metoda sa sufiksom -Y, -Z, respektivno.

Implementirano je automatsko uzlijetanje i slijetanje. Automatsko slijetanje implementirano je na način da se referentna pozicija počevši od trenutne pozicije smanjuje sve do nule u periodu od 10 sekundi. Uzlijetanje je implementirano na način da se u izlaznu temu postave zastavice koje ukazuju na uključenost regulatora pozicije u smjeru x,y i z osi, postavlja se 1 metar kao referentna pozicija u smjeru z osi i poziva servis *takeoff* s vrijednošću koja je predana putem ROS parametra *takeoff_value*.

Metoda *void voiceCallback(const controller::refinv &msg)* implementirana je za slučaj kombiniranog upravljanja putem glasa i igraće palice. Metoda čita podatke s teme *controlValueVoice* i dobivene promijene pozicije zbraja na referentnu poziciju letjelice.

5.1.2. Čvor voice_control_node.py

Upravljanje putem glasa omogućuje paket *voice_recognition* koji filtrira glas u tekst. Čvor *voice_contol_node.py* koristi te naredbe i na temelju njih određuje referentnu veličinu koja se zatim prosljeđuje u regulator. Implementirano je upravljanja s više bespilotnih letjelica i definiran je niz naredbi kojim se postiže što intuitivnije upravljanje. Prilikom pokretanja čvora potrebno je definirati listu imena bespilotnih letjelica kako bi se moglo naređivati letjelicama pojedinačno, a ne svima zajedno. Upotrebom online alata lmtool [37] iz definiranog popisa naredbi generira se jezični model i fonetski rječnik koji se koriste kao ulazi čvora za prepoznavanje govora 5.2.

5.1.2.1. Ulazno-izlazni podatci

Čvor *voice_control_node.py* izvorno se pretplaćuje na izlaznu temu *recognition_output* s porukama tipa *String* koja je izlaz iz čvora za prepoznavanje govora.

Prilikom pokretanja čvora moguće je predati nekoliko ROS parametara:

- *recognition_topic* - ime teme s prepoznatim naredbama,
- *step_value* - fiksna vrijednost promjene pozicije,
- *velocity_value* - vrijednost brzine,
- *step_increment_value* - vrijednost za koju se povećava/smanjuje iznos promjene pozicije,
- *velocity_increment_value* - vrijednost za koju se povećava/smanjuje iznos brzina,
- *drone_names* - lista imena bespilotnih letjelica s kojima se želi upravljati,

ukoliko se ne preda neki od parametara, koriste se predefinirane vrijednosti:

$$\begin{aligned} recognition\_topic &= \text{"}recognition\_output\text{"} \\ step\_value &= 0.05 \\ velocity\_value &= 0.05 \\ step\_increment\_value &= 0.01 \\ velocity\_increment\_value &= 0.01 \\ drone\_names &= \text{"}alpha\_one\text{"} \end{aligned} \tag{11}$$

Za svaku letjelicu s kojom se želi upravljati stvaraju se 4 izlazne teme, čija imena započinju imenom bespilotne letjelice. Referentna pozicija i orijentacija tipa *refinv* (definiranog u tom i tom poglavlju) objavljuje se u temu */controlValueVoice*. U temu */land* tipa poruke *std_msgs/Empty* postavlja se zastavica kada je potrebno uzletjeti, a u temu */land* također tipa poruke *std_msgs/Empty* kada se želi sletjeti. Posljednji izlaz iz čvora je poruka tipa *std_msgs/Bool* koji se objavljuje u temu */set_drone*. Ukoliko se upravlja s više bespilotnih letjelica tada se u temu */set_drone* s prefiksom imena trenutno selektirane letjelice objavljuje vrijednost poruke *True*, a u ostale, ne selektirane, vrijednost poruke *False*.

5.1.2.2. Metode

Implementirane su dvije klase: *voice_publisher* klasa koja se poziva za svaku bespilotnu letjelicu kojom se upravlja kako bi se definirale izlazne teme i pozivi servisa, *voice_control* klasa koja se koristi za transformaciju naredba u odgovarajuću referentnu veličinu. Prilikom pokretanja čvora predaje se ROS parametar s imenima letjelica. Stvara se objekt koji je instanca klase *voice_control*, a u njezinom konstruktoru za svako ime letjelice stvara se objekt koji je instanca klase *voice_publisher*. Time se može zasebno upravljati svakom letjelicom.

recognition_callback(self, msg), metoda klase *voice_control*, izvršava se svaki put kad se primi izgovorena naredba. Ukoliko je primljena naredba *BREAK* glasovno upravljanje je zaustavljeno sve

dok se ne primi naredba *CONTINUE*. Da bi se upravljalo pojedinom letjelicom potrebno je izgovoriti njezino ime, nakon toga letjelica sluša i izvršava naredbe sve dok se ne izgovori riječ *OVER*, ukoliko postoji daljnja želja za upravljanje istom ponovo je ponoviti postupak. Popis, objašnjenje svih naredba i njihova implementacija:

- **BREAK** - pauziranje glasovnog upravljanje, onemogućeno izvršavanje ostatka naredbi osim naredbe CONTINUE,
- **CONTINUE** - nastavak glasovnog upravljanja, omogućeno izvršavanje ostatka naredbi,
- **'ime letjelice'** - odabir letjelice kojom će se upravljati, letjelica sluša sve dok se ne izgovori naredba OVER. U izlaznu temu */set_drone* s prefiksom 'ime letjelice' objavljuje se vrijednost *True*, ako postoji vise letjelica, u temu */set_drone* s prefiksom koji je različit od 'ime letjelice' objavljuje se vrijednost *False*,
- **ADJUST STEP** - podešavanje vrijednosti promjene pozicije, sljedeća je naredba iznos u metrima na engleskom jeziku (0 - 9.9). Npr. ZERO POINT FIVE,
- **INCREASE STEP** - povećanje iznosa promjene pozicije za vrijednost *step_increment_value*,
- **DECREASE STEP** - smanjenje iznosa promjene pozicije za vrijednost *step_increment_value*,
- **ADJUST VELOCITY** - podešavanje vrijednosti brzine, sljedeća je naredba iznos u metrima po sekundi (0 -9.9). Npr. ZERO POINT FIVE,
- **INCREASE VELOCITY** - povećanje iznosa promjene brzine za vrijednost *velocity_increment_value*,
- **DECREASE VELOCITY** - smanjenje iznosa promjene brzine za vrijednost *velocity_increment_value*,
- **OVER** - završeno je naređivanje letjelici, letjelica više ne sluša naredbe,
- **ARM** - postavljanje letjelicu u pripravnost za let. Pozivanje servisa */arm*,
- **DISARM** - postavljanje letjelice u stanje bez mogućnosti leta. Pozivanje servisa */disarm*,
- **LAND** - slijetanje. Objavljivanje poruke tipa *std_msgs/Empty* u temu */land*,
- **TAKEOFF** - uzlijetanje. Objavljivanje poruke tipa *std_msgs/Empty* u temu */takeoff*,
- **MOVE/GO FORWARD** - pomak letjelice naprijed određeni broj metara. Sljedećom naredbom definira se željeni pomak u metrima (npr. ZERO POINT FIVE). Vrijednost pomaka objavljuje se u temu */controlValueVoice*,
- **MOVE/GO BACK** - pomak letjelice iza određeni broj metara. Sljedećom naredbom definira se željeni pomak u metrima (npr. ZERO POINT FIVE). Vrijednost pomaka objavljuje se u temu */controlValueVoice*,
- **MOVE/GO RIGHT** - pomak letjelice desno određeni broj metara. Sljedećom naredbom definira se željeni pomak u metrima (npr. ZERO POINT FIVE). Vrijednost pomaka objavljuje se u temu */controlValueVoice*,

- **MOVE/GO LEFT** - pomak letjelice lijevo određeni broj metara. Sljedećom naredbom definira se željeni pomak u metrima (npr. ZERO POINT FIVE). Vrijednost pomaka objavljuje se u temu */controlValueVoice*,
- **STEP FORWARD** - pomak letjelice naprijed za vrijednost *step_value*. Vrijednost pomaka objavljuje se u temu */controlValueVoice*,
- **STEP BACK** - pomak letjelice iza za vrijednost *step_value*. Vrijednost pomaka objavljuje se u temu */controlValueVoice*,
- **STEP LEFT** - pomak letjelice lijevo za vrijednost *step_value*. Vrijednost pomaka objavljuje se u temu */controlValueVoice*,
- **STEP RIGHT** - pomak letjelice desno za vrijednost *step_value*. Vrijednost pomaka objavljuje se u temu */controlValueVoice*,
- **STEP ROTATE LEFT** - rotacija letjelice lijevo za vrijednost *step_value*. Vrijednost pomaka objavljuje se u temu */controlValueVoice*,
- **STEP ROTATE RIGHT** - rotacija letjelice desno za vrijednost *step_value*. Vrijednost pomaka objavljuje se u temu */controlValueVoice*,
- **TURN VALVE LEFT** - okretanje ventila u lijevo, rotacija u lijevo za $\frac{\pi}{2}$. Vrijednost pomaka objavljuje se u temu */controlValueVoice*,
- **TURN VALVE RIGHT** - okretanje ventila u desno, rotacija u desno za $\frac{\pi}{2}$. Vrijednost pomaka objavljuje se u temu */controlValueVoice*,

Metoda *run(self)* pokreće upravljačku petlju na način da čeka pojavu nove naredbe, kada stigne naredbe tada kreće njezina obrada.

5.1.3. Upute za pokretanje paketa

Prilikom pokretanja čvora *joy_control_arducopter* mogu se predati dva argumenta naredbenog retka (*-calibrate* i *-file*) i četiri ROS parametra (*takeoff_value*, *joy_topic*, *voice_and_joy* i *mocap_topic*). Primjer pokretanja čvora:

```
rosrun arducopter_control joy_control_arducopter -calibrate 1 -file
calibration/ArducopterJoy.txt _mocap_topic:=Optitrack _takoff_value:=1350
_voice_and_joy:=0 _joy_topic:=/joy
```

Čvoru *voice_control_node.py* mogu se predati šest ROS parametara (*recognition_topic*, *step_value*, *velocity_value*, *step_increment_value*, *velocity_increment_value* i *drone_names*. Primjer pokretanja čvora:

```
rosrun arducopter_control voice_control_node.py
_recognition_topic:=recognition_output _step_value:=0.1 _velocity_value:=0.1
_step_increment_value:=0.02 _velocity_increment_value:=0.02
_drone_names:="alpha_one bravo_one"
```

Kombinirano upravljanje zahtjeva pokretanje oba čvora te postavljanje ROS parametra `_voice_and_joy:=1`

5.2. Paket voice_recognition

Paket *voice_recognition* služi za prepoznavanje govora. Sadrži čvor *voice_recognition_node* koji je jednostavan omotač (eng. *wrapper*) oko paketa pocketsphinx napisan u programskom jeziku Python. Koristi gstreamer kako bi omogućio zaustavljanje i pokretanje prepoznavanja govora te automatski podijelio izgovorene rečenice u riječi, naredbe, što je vrlo koristan dodatak pocketsphinxu jer na taj način pocketsphinx dekoder može biti tretiran kao element u cjevovodu, tj. element koji filtrira glas u tekst. Cjevovod koji se koristi za prepoznavanje govora sastoji se od izvora zvuka, koji se prilagođava formatu prikladnom za pocketsphinx, VADER (eng. *Voice Activity Detector*), pocketsphinx i fakesink elementa. VADER element služi kako bi pocketsphinx znao kada rečenica počinje, a kada završava, tj. kada treba obavljati filtriranje.

5.2.1. Ulazno-izlazni podatci

Za uspješno prepoznavanje govora čvoru je potreban jezični model i fonetski rječnik. Lokacija jezičnog modela i fonetskog rječnika čvoru se može predati putem ROS parametara *lm* i *dict*, respektivno. Čvor izvorno prepoznaje engleski jezik no postoji mogućnost prepoznavanja nekog drugog jezika ukoliko se putem ROS parametra *hmm* preda lokacija akustičnog modela tog jezika. Pokretanjem čvora automatski se pokreće prepoznavanje govora. Moguće je pozivom ROS servisa *stop_recognition* zaustaviti, *pause_recognition* pauzirati ili *start_recognition* ponovno pokrenuti prepoznavanje govora. Svaki od tri navedena servisa su tipa *std_srvs/Empty* i pozivaju se bez parametara. Prepoznate riječi, naredbe, su tipa podatka *String* te ih kao takve čvor objavljuje u temu *recognition_output*. Čvoru se može predati ROS parametar *output_topic* kojim se definira ime izlazne teme, no ukoliko se taj parametar ne preda koristi se zadano ime teme *recognition_output*.

5.2.2. Upute za rad s paketom

Prije rada s paketom *voice_recognition* potrebno je instalirati gstreamer i pocketsphinx naredbom: `sudo apt-get install gstreamer0.10-pocketsphinx`. Paket treba smjestiti u radni prostor ROS-a te u direktorij *scripts/voice_recognition/* staviti jezični model i fonetski rječnik. Koristeći online alat lmtool [37] može se iz teksta ili niza naredba generirati fonetski rječnik i jezični model. Pokretanjem skripte *setup.py*, koja dolazi u sklopu paketa, rječnik i model se prebacuju na lokaciju */usr/local/share/pocketsphinx/model/lm/* iz razloga što ih na toj lokaciji pocketsphinx očekuje. Moguće ih je postaviti na neku drugu lokaciju no tada je potrebno čvoru predati potpun put do datoteke putem ROS parametara *lm* i *dict*.

Primjer pokretanja čvora:

```
rosrun voice_recognition voice_recognition_node.py
_lm:=/voice_recognition/lm/model.lm _dict:=/voice_recognition/lm/model.dic
```

5.3. Paket skeleton_tracker

Paket *skeleton_tracker* čine dva čvora: *TransformToSkeletons* i *skeleton_joints*. Zadaća paketa je dohvatiti podatke o položaju tijela, koje šalje čvor *openni_tracker*, te ih obraditi i pretvoriti u pogodniji tip podataka. Osim pretvorbe u sklopu paketa nalazi se čvor za vizualizaciju trenutnog položaja korisnika.

5.3.1. Čvor TransformToSkeletons

Preduvjet za ispravan rad čvora *TransformToSkeletons* je uspješno instaliran i pokrenut čvor *openni_tracker* koji se može besplatno preuzeti na [24]. Čvor *openni_tracker* uzima podatke s kamere Kinecta te pokreće algoritme za detekciju ljudskog tijela. Informacije o koordinatama karakterističnim točaka ljudskog tijela objavljuju se u temu *tf* kao tip poruke *TFMessage*. Zadaća čvora *TransformToSkeletons* je pretvorba tih podataka u drugi tip poruke kako bi ih bilo lakše obraditi.

5.3.1.1. Ulazno-izlazni podatci

Osnovna poruka s kojom radi čvor *TransformToSkeletons* je tipa *SkeletonJoint*. To je složen tip podataka u kojem se nalaze informacije o položaju zgloba u prostoru te njegovoj orijentaciji. Drugi tip poruke je *Skeleton* koji se sastoji od petnaest podataka tipa *SkeletonJoint* koji sadrže informacije o karakterističnim točkama ljudskog tijela u prostoru. U ovom slučaju radi se o petnaest točaka: glava, vrat, ramena, lakti, šake, torzo, kukovi, koljena i stopala. Dakle, *Skeleton* sadrži sve potrebne informacije o položaju ljudskog tijea. Tip podataka *Skeletons* je niz podataka tipa *Skeleton*. Na taj način se omogućuje praćenje više korisnika, najviše petnaest jer je tako određeno u čvoru *openni_tracker*.

5.3.1.2. Parametri

Čvor prima parametar niskopropusnog filtra p kao realan broj u intervalu $p \in [0, 1]$. Ukoliko se parametar ne preda kao argument vrijednost se postavlja na $p = 0.25$.

5.3.1.3. Metode

Ulazni podatci u čvor su tipa *TFMessage* dok su izlazni tipa *Skeletons*. Na taj način šalju se informacije o svim korisnicima čiji se položaj u prostoru u tom trenutku prati. Također, šalju se i informacije o brzini pojedinog zgloba u svim smjerovima.

Metoda *subtraction(self, current, previous)* služi za oduzimanje podataka tipa *Skeletons* te vraća razliku $current - previous$ po svim karakterističnim točkama ljudskog tijela.

Metoda *assign_value(self, input)* služi za dodjelu vrijednosti jedne varijable tipa *Skeletons* drugoj. Metoda je potrebna jer je kod čvora pisan u jeziku *Python* jer su deklarirane varijable pokazivači.

U slučaju da se vrijednost jedne varijable preslika u drugu, bez korištenja metode, pokazivači će pokazivati na istu adresu. Time se vrijednost jedne varijable mijenja pri promjeni druge što dovodi do gubitka podataka.

Metoda *scalar_divide(self, input_value, den)* dijeli sve koordinate podatka tipa *Skeletons* s realnim brojem den.

Metoda *low_pass_filter(self, current, previous, p)* je niskopropusni filtar u ovisnosti o parametru $p \in [0,1]$. Jednadžba filtera glasi:

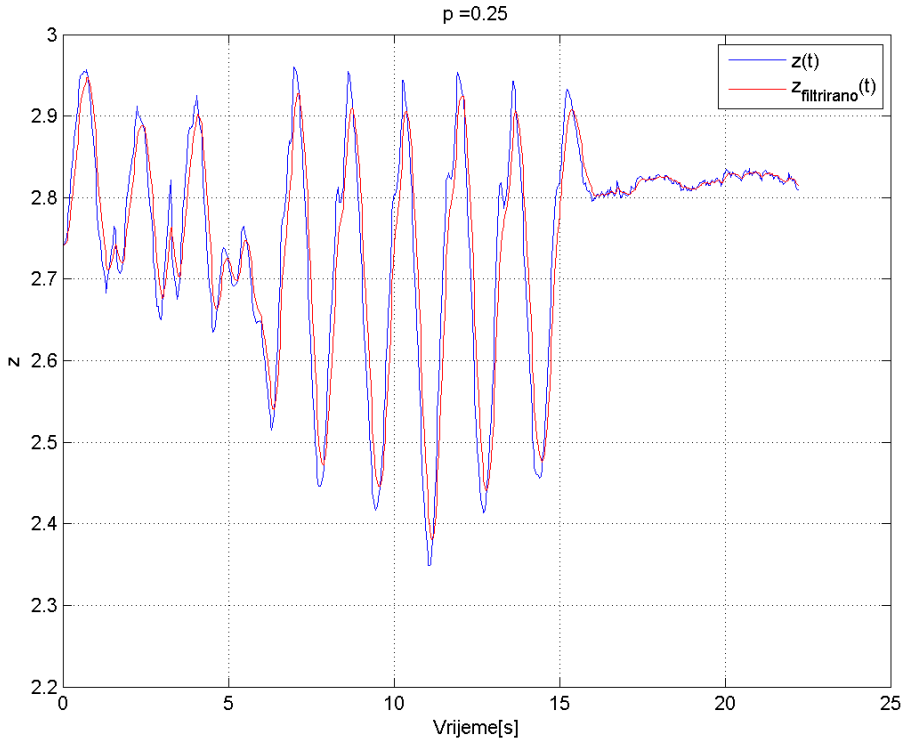
$$current = p \cdot current + (1-p) \cdot previous \tag{12}$$

te radi po svim karakterističnim točkama.

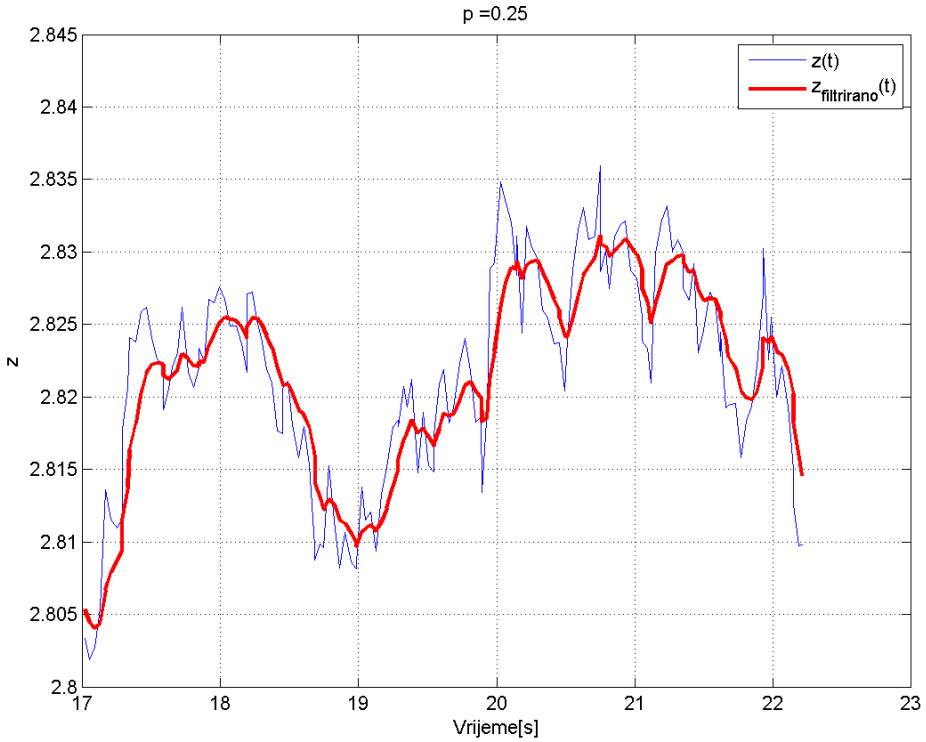
Posljednja funkcija u klasi je *callback(self, data)* koja prima podatke tipa *TFMessage* pretplatom na temu *tf*. Informacija o tipu karakteristične točke koja je proslijeđena u funkciju nalazi se u podatku *data* te je tipa *string*. Dvije su moguće kombinacije znakovnog niza: *točka_korisnik* te *strana_točka_korisnik* pri čemu *strana* može biti lijeva ili desna. Zbog strukture znakovnog niza on se može lako podijeliti po znaku "_" te se tako dolazi do informacije o kojoj se karakterističnoj točki radi. Također, prije slanja podatci se filtriraju niskopropusnim filtrom.

5.3.1.4. Filtriranje podataka

Parametar p niskopropusnog filtra oblika (12) određen je eksperimentalno te iznosi $p = 0.25$. Filtriranjem podataka se ne smije usporiti odziv jer u tom slučaju dolazi do kašnjenja kod brzih reakcija operatera, a istovremeno mora filtrirati visokofrekventni šum, odnosno "šiljke" u odzivu. Na slici 60 može se vidjeti kako filtrirani odziv dobro prati stvarni odziv. Dakle filtar je dovoljno brze dinamike kako se ne bi gubili podatci o brzim kretnjama korisnika. Na slici 61 uvećano je prikazan dio odziva na kojem je korisnik pokušao držati ruku mirnom. U stvarnom odzivu postoje skokovi koje je filtar uspješno prigušio čime podatci postaju pogodniji za daljnju obradu.



Slika 60: Snimljena kretnja korisnika bez i sa filtriranjem



Slika 61: Uvećan posljednji dio odziva sa slike 60

5.3.2. Čvor skeleton_joints

Preduvjeti za rad čvora *skeleton_joints* su instalirana biblioteka *OpenCV* te paketi *openni_camera* i *openni_tracker*. Čvor služi za vizualizaciju položaja karakterističnih točaka korisnika koristeći gotove funkcije za crtanje po slici biblioteke *OpenCV* te informacije o položaju korisnika. Slika korištena za vizualizaciju dolazi s *Kinect*-a te se zato čvor pretplaćuje na temu *rgb/image_raw* kako bi se na toj slici iscrtale karakteristične točke.

5.3.2.1. Tipovi poruka

Čvor koristi dva tipa poruka: *sensor_msgs/Image* i *skeleton_tracker/Skeletons*. Ulazi u čvor su slika s *Kinect* kamere u boji koju objavljuje čvor *openni_node* iz paketa *openni_camera* u temu *rgb/image_raw* te koordinate karakterističnih točaka u prostoru koje objavljuje čvor *TransformToSkeletons* u temu *skeletons/position*. Izlaz iz čvora je obrađena slika objavljena u temi *myimage*.

5.3.2.2. Metode

Funkcija *callback(self, data)* prima sliku s kamere te pomoću funkcije *Circle* iz bibilioteke *OpenCV* crta krugove na mjestima karakterističnih točaka. Također, u funkciji se objavljuje obrađena slika u temu *myimage*.

Funkcija *kinect_callback(self, data)* prima informacije o karakterističnim točkama korisnika. Svaka se točka obrađuje kako bi se mogla nacrtati na izlaznu sliku. Koristeći informaciju o udaljenosti točke od kamere može se izračunati širina i visina slike u metrima za tu udaljenost. Označi li se širina s d, visina s h, horizontalno vidno polje s α, vertikalno vidno polje s β te udaljenost sa z izrazi za izračun širine i visine slike u metrima glase:

$$\begin{aligned} d &= 2 \cdot z \cdot \tan\alpha \\ h &= 2 \cdot z \cdot \tan\beta \end{aligned} \tag{13}$$

Kutevi $\alpha = 58^\circ$ i $\beta = 43^\circ$ preuzeti su iz specifikacija uređaja. Uz poznatu visinu i širinu slike potrebno je još pretvoriti koordinate x i y točke u vrijednosti pogodne za crtanje na sliku. Uz rezoluciju od $640 \times 480 pixela$ dolazi se do izraza:

$$\begin{aligned} x_{px} &= \frac{-x \cdot 640}{d} + 320 \\ y_{px} &= \frac{y \cdot 480}{d} + 240 \end{aligned} \tag{14}$$

pri čemu su x_{px} i y_{px} koordinate karakterističnih točaka u *pixelima*.

5.4. Paket dynamixel_kinect_control

Paket *dynamixel_kinect_control* služi za pokretanje dvostrukog manipulatora s dva stupnja slobode putem *Kinect*-a. Tri su čvora u sklopu samog paketa: *kinect_arms_controller* za obradu položaja korisnika te slanje upravljačkih naredbi na manipulator, *kinect_pose_detect* za detektiranje trenutne poze korisnika i *user_calibration* za stvaranje kalibracijske datoteke korisnika. Za ispravan rad paketa potreban je paket *skeleton_tracker* jer se koristi tip poruke *Skeletons* te da bi se mogle primati koordinate karakterističnih točaka korisnika.

5.4.1. Čvor kinect_arms_controller

Čvor *kinect_arms_controller* skuplja podatke o trenutnom položaju korisnika te preslikava trenutni položaj ruku na dvostruki manipulator s dva stupnja slobode. Također, koristi podatke koje šalje čvor *kinect_pose_detect* te ovisno o pozi korisnika može doći do promjene načina rada.

5.4.1.1. Ulazno-izlazni podatci

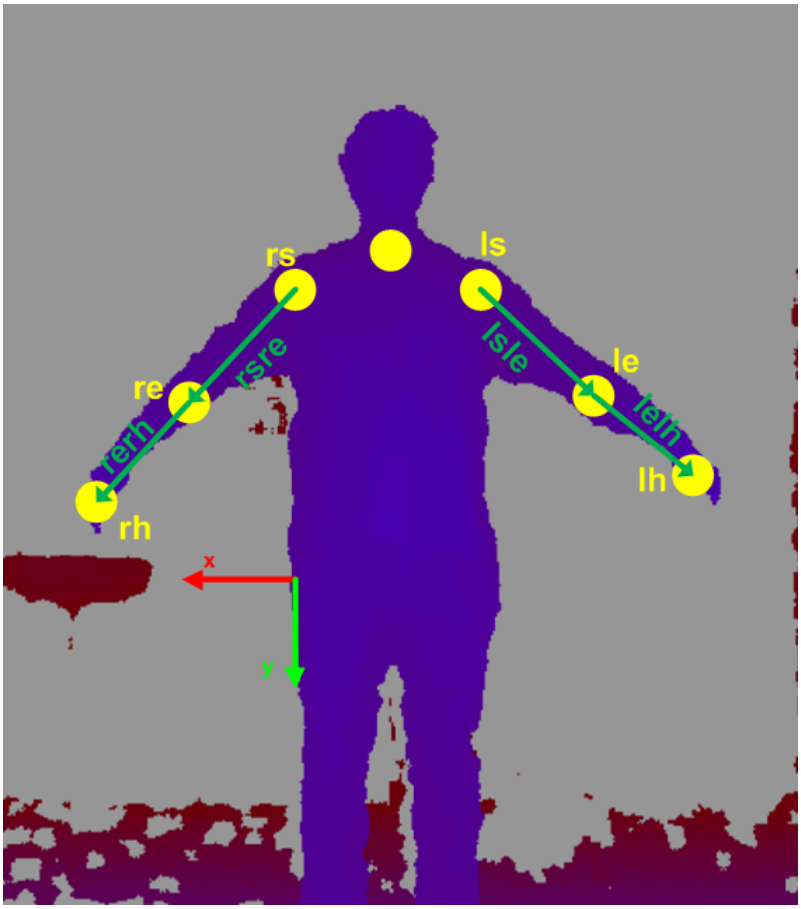
Pretplatom na dvije teme dobivaju se potrebni podatci za upravljanje dvostrukim manipulatorom s dva stupnja slobode. Pretplatom na temu *skeletons/position* dohvaćaju se podatci tipa *Skeletons* o trenutnom položaju korisnika. Druga tema na koju se čvor pretplaćuje je *kinect/pose* s podacima tipa *String* iz koje se čita trenutna poza korisnika.

Izlazi iz čvora su četiri teme za slanje kuteva zakreta motora tipa podataka *Float64*. Teme redom glase: *JointA1_controller/command*, *JointA2_controller/command*, *JointB1_controller/command*, *JointB2_controller/command* te se odnose na lijevo rame, lijevi lakat, desno rame i desni lakat dvostrukog manipulatora, respektivno. Posljednji izlaz iz čvora je poruka tipa *ArmsStatus* koja se objavljuje u temu *arms_data*. *ArmsStatus* je tip poruke u kojem se nalaze dvije zastavice: *EmergencyFlag* odnosno zastavica za izvanredno stanje te *ControlFlag* odnosno upravljačka zastavica.

5.4.1.2. Metode

Funkcija *kinect_callback(self, data)* prima podatke o trenutnom položaju korisnika. Za rad samog čvora nisu potrebni svi podatci već samo položaj ramena, lakta i šaka. Primljeni podatci pretvaraju se u vektore pomoću metode *vector3_maker*. Vektori potrebni za izračun kuteva zakreta motora prikazani su na slici 62. Pritom se vektori definiraju na sljedeći način:

- *rsre* → Vektor od desnog ramena do desnog lakta
- *rerh* → Vektor od desnog lakta do desne šake
- *lsle* → Vektor od lijevog ramena do lijevog lakta
- *lelh* → Vektor od lijevog lakta do lijeve šake



Slika 62: Prikaz vektora ruku potrebnih za izračun kuteva zakreta motora

Metoda *vector3_maker(self, vector_start_point, vector_end_point)* računa konačni vektor kao razliku početne i završne točke:

$$vector_{out} = point_{end} - point_{start} \tag{15}$$

Funkcija *pose_callback(self, data)* prima podatke iz teme *kinect/pose* tipa *String* te mijenja način rada na temelju njihovih vrijednosti. Tri su moguća položaja nogu na koje će se promjeniti način rada opisana u poglavlju 4.2.6.

Metoda *run(self)* pokreće upravljačku *while* petlju koja se izvodi frekvencijom od $25Hz$. U prvom dijelu petlje računaju se kutevi otklona ruku korisnika. Uzevši u obzir koordinatni sustav i nazive vektora prema slici 62 izrazi, (9) - (10), služe za računanje kuteva otklona lijevog i desnog ramena.

5.4.1.3. Upute za pokretanje

Kako čvor ne prima parametre pokretanje se jednostavno izvodi naredbom:

```
rosrun dynamixel_kinect_control kinect_arms_controller.py
```

5.4.2. Čvor kinect_pose_detect

Čvor *kinect_pose_detect* služi za detekciju trenutne poze korisnika koje se mogu prevoditi u razne upravljačke veličine, promjene načina rada, odabir resursa kojim se želi upravljati itd.

5.4.2.1. Ulazno-izlazni podatci

Čvor se pretplaćuje na temu *skeletons/position* s tipom poruke *skeleton_tracker/Skeletons* iz koje dobiva informacije o trenutnom položaju korisnika.

Detektiranu pozu korisnika čvor šalje na temu *kinect/pose* u obliku podataka tipa *String*.

5.4.2.2. Parametri

Kako je detekcija poza izrađena po referentnom korisniku potrebno je prilagoditi karakterističnе veličine ostalih korisnika u odnosu na referentnog korisnika. Kalibracijska datoteka može se izraditi čvorom *user_calibration* te se kao takva predaje čvoru *kinect_pose_detect* kao parametar *user_dict*. Također, čvor učitava i kalibracijsku datoteku referentnog korisnika.

5.4.2.3. Metode

Funkcija *kinect_callback(self, data)* prima podatke o trenutnom položaju korisnika. Dobiveni podatci koordinata karakterističnih točaka pohranjuju se u varijable tipa *geometry_msgs/Vector3*.

Metoda *vectors(self)* stvara vektore, pomoću metode *vector3_maker*, korištene za provjeru poze korisnika. Iz podataka o položaju korisnika stvara se sedamnaest vektora za provjeru trenutne poze korisnika. Poze su opisane u poglavlju 4.2.5.

Metoda *vector3_maker(self, vector_start_point, vector_end_point, vector_id)* stvara vektor od početne do krajnje točke te ga skalira. Za samo skaliranje potrebna su dva podatka: duljina vektora trenutnog korisnika i duljina vektora referentnog korisnika. Podatci o duljini vektora učitavaju se iz kalibracijskih datoteka pri inicijalizaciji čvora. Skaliranjem oblika:

$$vector_{out} = (point_{start} - point_{end}) \cdot \frac{|vector_{ref,vector\_id}|}{|vector_{user,vector\_id}|} \tag{16}$$

duljina izlaznog vektora je prilagođena duljini vektora referentnog korisnika.

Metoda *run(self)* pokreće upravljačku petlju frekvencije 10Hz. Na temelju iznosa pojedinih komponenti prethodno opisanih vektora detektiraju se poze operatera. Ukupno postoji sedamnaest različitih poza, slike 56 i 57, koje se mogu detektirati te su opisane u poglavlju 4.2.5.

5.4.2.4. Upute za pokretanje

Da bi se čvor mogao pokrenuti mora postojati kalibracijska datoteka referentnog korisnika *default.pkl* koja se nalazi u samom paketu. Ovu datoteku ne preporuča se mijenjati jer su prema njoj izrađena područja za datekciju poze korisnika. Kalibracijska datoteka trenutnog korisnika može se predati kao parametar:

```
rosrun dynamixel_kinect_control kinect_pose_detect.py _user_cal_file:="user_file.pkl"
```

Ukoliko se kalibracijska datoteka korisnika ne preda kao argument čvor će uzeti datoteku *default.pkl*.

5.4.3. Čvor user_calibration

Čvor *user_calibration* koristi se za stvaranje kalibracijske datoteke određenog korisnika, ukoliko je to potrebno. Za kalibraciju se skupljaju duljine vektora, opisanih u prethodnom poglavlju, te se na traži aritmetička sredina uzoraka.

5.4.3.1. Ulazno-izlazni podatci

Čvor se pretplaćuje na temu *skeletons/position*, s podatcima tipa *Skeletons*, iz koje dobiva informaciju o trenutnom položaju korisnika.

Parametri

Čvor prima dva parametra. *sample_number* označava broj uzoraka na temelju kojih će se aritmetičkom sredinom proračunati duljine pojedinih vektora korisnika, a izvorno iznosi $cal\_length = 10$. Drugi parametar, *output_file*, odnosi se na ime kalibracijske datoteke te se zadaje pod navodnicima, a izvorni naziv datoteke glasi: *user.pkl*.

5.4.3.2. Metode

Metode *kinect_callback(self, data)* i *vectors(self)* opisane jednake su istoimenim metodama opisanim u poglavlju 5.4.2. Metoda *vector_length(self, vector)* vraća normu ulaznog vektora.

Funkcija *run(self)* je petlja frekvencije $f = 10Hz$ koja se prekida nakon uspješno završene kalibracije ili na zahtjev korisnika. Da bi se spremile duljine karakterističnih vektora, opisanih u 5.4.2.3, korisnik se mora nalaziti u jednoj od poza 16 ili 17 prikazanih slikom 57. Za provjeru poze gleda se samo ima li korisnik raširene ruke te u tom trenutku sprema duljine karakterističnih vektora gornjeg dijela tijela. Druga poza koja se provjerava jest stoji li korisnik uspravno te, u slučaju da se to ostvari, sprema duljine vektora donjeg dijela tijela. Broj uzoraka koje treba prikupiti definira korisnik pri pokretanju čvora parametrom *sample_number*.

Metoda *save_calibration* pokreće se nakon dovoljno prikupljenih uzoraka te traži aritmetičku sredinu uzoraka za pojedini karakterističan vektor. Rezultati se spremaju u datoteku, koja se također može zadatki kao parametar.

5.4.3.3. Upute za pokretanje

Korisnik može zadati dva parametra pri pokretanju: broj uzoraka za traženje duljine karakterističnih vektora i izlaznu, kalibracijsku, datoteku. Primjer pokretanja, uz *n*-broj uzoraka te *izlazna_datoteka.pkl*-kalibracijska datoteka, glasi:

```
rosrun dynamixel_kinect_control user_calibration.py _sample_number:=n
_output_file:="izlazna_datoteka.pkl"
```

5.5. Paket controller

Paket *controller* služi za reguliranje pozicije i orijentacije letjelice. Sadrži čvor *PI_D_controller* u kojem je implementirana struktura upravljanja sa dva regulacijska kruga (podređenog regulacijskog kruga i nadređenog regulacijskog kruga) koja se još naziva kaskadna struktura upravljanja. Oba regulatora, nadređeni i podređeni, su digitalni paralelni PID regulatori. Čvor se koristi za reguliranje pozicije letjelice i rotacije oko z osi. Kad regulacija nije uključena čvor obrađuje ulazne podatke i objavljuje ih u izlaznu temu. U paketu su definirana tri tipa poruka:

coninv - sastoji se od četiri podataka tipa *uint8*

```
uint8 x_control
uint8 y_control
uint8 z_control
uint8 yaw_control
```

refinv - sastoji se od dvije poruke tipa *Vector3* i jedne poruke tipa *coninv*

```
geometry_msgs/Vector3 position
geometry_msgs/Vector3 orientation
controller/coninv control
```

ControllerOutput - sastoji se od šest poruka tipa *Quaternion*

```
geometry_msgs/Quaternion P
geometry_msgs/Quaternion I
geometry_msgs/Quaternion D
geometry_msgs/Quaternion P_v
geometry_msgs/Quaternion I_v
geometry_msgs/Quaternion D_v
```

i tri tipa servisa koji se sastoje od zahtjeva i odgovora:

SendFloat - sadrži zahtjev tipa *float64*, odgovor je prazan

```
float64 a
--
```

SendString - sadrži zahtjev tipa *string*, odgovor je prazan

```
string a
--
```

SendSineWave - sadrži dva zahtjeva tipa *float64* i jedan zahtjev tipa *bool*, odgovor je prazan

```
float64 omega
float64 amplitude
bool on_off
--
```

5.5.1. Čvor PI_D_controller

U čvor *PI_D_controller* implementirana je struktura upravljanja kojom se postiže što preciznije pozicioniranje bespilotne letjelice. Čvor prima poziciju bespilotne letjelice i orijentaciju oko z osi te na temelju tih veličina računa upravljačku veličinu.

5.5.1.1. Ulazno-izlazni podatci

Ulazni podatci u čvor su referentna pozicija/brzina koja se čita iz teme */Reference* poruke tipa *controller/refinv* te trenutna pozicija, orijentacija i brzina letjelice koje se čitaju iz teme */Optitrack* poruke tipa *ACROSS_Optitrack/OptitrackPoseVel*. Definiran je ROS servis server *takeoff* kojim se postiže automatsko uzlijetanje letjelice. Izlazi iz čvora su tema *ControllerOutput* i tema *send_rc* ili *cmd_vel* ovisno o argumentu naredbenog retka *-msg*. Upravljačka veličina objavljuje se porukom *geometry_msgs/Twist* u temu *cmd_vel* ako je argument *-msg* jednak *Twist* ili ako je argument *-msg* jednak *RC* porukom *roscopter/RC* u temu *send_rc*. Doprinos proporcionalnog P, integracijskog I i derivacijskog D dijela regulatora objavljuju se u temu *ControllerOutput* porukom *controller/ControllerOutput*. Osim argumenta naredbenog retka *-msg* prilikom pokretanja čvoru se mogu predati 76 ROS parametara. Popis ROS parametara i njihovo objašnjenje dano je u nastavku:

- **Kp_x** - Konstanta pojačanja proporcionalnog dijela nadređenog regulatora regulacijskog kruga $x-osi$
- **Kp_y** - Konstanta pojačanja proporcionalnog dijela nadređenog regulatora regulacijskog kruga $y-osi$
- **Kp_z** - Konstanta pojačanja proporcionalnog dijela nadređenog regulatora regulacijskog kruga $z-osi$
- **Kp_yaw** - Konstanta pojačanja proporcionalnog dijela nadređenog regulatora regulacijskog kruga orijentacije $z-osi$
- **Ki_x** - Konstanta pojačanja integracijskog dijela nadređenog regulatora regulacijskog kruga $x-osi$
- **Ki_y** - Konstanta pojačanja integracijskog dijela nadređenog regulatora regulacijskog kruga $y-osi$

- **Ki_z** - Konstanta pojačanja integracijskog dijela nadređenog regulatora regulacijskog kruga $z-osi$
- **Ki_yaw** - Konstanta pojačanja integracijskog dijela nadređenog regulatora regulacijskog kruga orijentacije $z-osi$
- **Kd_x** - Konstanta pojačanja derivacijskog dijela nadređenog regulatora regulacijskog kruga $x-osi$
- **Kd_y** - Konstanta pojačanja derivacijskog dijela nadređenog regulatora regulacijskog kruga $y-osi$
- **Kd_z** - Konstanta pojačanja derivacijskog dijela nadređenog regulatora regulacijskog kruga $z-osi$
- **Kd_yaw** - Konstanta pojačanja derivacijskog dijela nadređenog regulatora regulacijskog kruga orijentacije $z-osi$
- **Kp_vx** - Konstanta pojačanja proporcionalnog dijela podređenog regulatora regulacijskog kruga $x-osi$
- **Kp_vy** - Konstanta pojačanja proporcionalnog dijela podređenog regulatora regulacijskog kruga $y-osi$
- **Kp_vz** - Konstanta pojačanja proporcionalnog dijela podređenog regulatora regulacijskog kruga $z-osi$
- **Kp_vyaw** - Konstanta pojačanja proporcionalnog dijela podređenog regulatora regulacijskog kruga orijentacije $z-osi$
- **Ki_vx** - Konstanta pojačanja integracijskog dijela podređenog regulatora regulacijskog kruga $x-osi$
- **Ki_vy** - Konstanta pojačanja integracijskog dijela podređenog regulatora regulacijskog kruga $y-osi$
- **Ki_vz** - Konstanta pojačanja integracijskog dijela podređenog regulatora regulacijskog kruga $z-osi$
- **Ki_vyaw** - Konstanta pojačanja integracijskog dijela podređenog regulatora regulacijskog kruga orijentacije $z-osi$
- **Kd_vx** - Konstanta pojačanja derivacijskog dijela podređenog regulatora regulacijskog kruga $x-osi$
- **Kd_vy** - Konstanta pojačanja derivacijskog dijela podređenog regulatora regulacijskog kruga $y-osi$
- **Kd_vz** - Konstanta pojačanja derivacijskog dijela podređenog regulatora regulacijskog kruga $z-osi$

- **Kd_vyaw** - Konstanta pojačanja derivacijskog dijela podređenog regulatora regulacijskog kruga orijentacije $z-osi$
- **T** - period uzorkovanja
- **upper_limit_x** - gornja granična vrijednost funkcije zasićenja za $x-os$ (ograničenje)
- **lower_limit_x** - donja granična vrijednost funkcije zasićenja za $x-os$ (ograničenje)
- **upper_limit_y** - gornja granična vrijednost funkcije zasićenja za $y-os$ (ograničenje)
- **lower_limit_y** - donja granična vrijednost funkcije zasićenja za $y-os$ (ograničenje)
- **upper_limit_z** - gornja granična vrijednost funkcije zasićenja za $z-os$ (ograničenje)
- **lower_limit_z** - donja granična vrijednost funkcije zasićenja za $z-os$ (ograničenje)
- **upper_limit_yaw** - gornja granična vrijednost funkcije zasićenja (ograničenje), orijentacija oko $z-osi$
- **lower_limit_yaw** - donja granična vrijednost funkcije zasićenja (ograničenje), orijentacija oko $z-osi$
- **upper_d_x** - gornja granična vrijednost izlaza derivacijskog dijela nadređenog regulatora za $x-os$
- **lower_d_x** - donja granična vrijednost izlaza derivacijskog dijela nadređenog regulatora za $x-os$
- **upper_d_y** - gornja granična vrijednost izlaza derivacijskog dijela nadređenog regulatora za $y-os$
- **lower_d_y** - donja granična vrijednost izlaza derivacijskog dijela nadređenog regulatora za $y-os$
- **upper_d_z** - gornja granična vrijednost izlaza derivacijskog dijela nadređenog regulatora za $z-os$
- **lower_d_z** - donja granična vrijednost izlaza derivacijskog dijela nadređenog regulatora za $z-os$
- **upper_d_yaw** - gornja granična vrijednost izlaza derivacijskog dijela nadređenog regulatora, orijentacija oko $z-osi$
- **lower_d_yaw** - donja granična vrijednost izlaza derivacijskog dijela nadređenog regulatora, orijentacija oko $z-osi$
- **upper_i_x** - gornja granična vrijednost izlaza integracijskog dijela nadređenog regulatora za $x-osi$
- **lower_i_x** - donja granična vrijednost izlaza integracijskog dijela nadređenog regulatora za $x-osi$

- **upper_i_y** - gornja granična vrijednost izlaza integracijskog dijela nadređenog regulatora za $y-osi$
- **lower_i_y** - donja granična vrijednost izlaza integracijskog dijela nadređenog regulatora za $y-osi$
- **upper_i_z** - gornja granična vrijednost izlaza integracijskog dijela nadređenog regulatora za $z-osi$
- **lower_i_z** - donja granična vrijednost izlaza integracijskog dijela nadređenog regulatora za $z-os$
- **upper_i_yaw** - gornja granična vrijednost izlaza integracijskog dijela nadređenog regulatora, orijentacija oko $z-osi$
- **lower_i_yaw** - donja granična vrijednost izlaza integracijskog dijela nadređenog regulatora, orijentacija oko $z-osi$
- **upper_d_vx** - gornja granična vrijednost izlaza derivacijskog dijela podređenog regulatora za $x-os$
- **lower_d_vx** - donja granična vrijednost izlaza derivacijskog dijela nadređenog regulatora za $x-os$
- **upper_d_vy** - gornja granična vrijednost izlaza derivacijskog dijela podređenog regulatora za $y-os$
- **lower_d_vy** - donja granična vrijednost izlaza derivacijskog dijela nadređenog regulatora za $y-os$
- **upper_d_vz** - gornja granična vrijednost izlaza derivacijskog dijela podređenog regulatora za $z-os$
- **lower_d_vz** - donja granična vrijednost izlaza derivacijskog dijela nadređenog regulatora za $z-os$
- **upper_d_vyaw** - gornja granična vrijednost izlaza derivacijskog dijela podređenog regulatora, orijentacija oko $z-osi$
- **lower_d_vyaw** - donja granična vrijednost izlaza derivacijskog dijela nadređenog regulatora, orijentacija oko $z-osi$
- **upper_i_vx** - gornja granična vrijednost izlaza integracijskog dijela nadređenog regulatora za $x-osi$
- **lower_i_vx** - donja granična vrijednost izlaza integracijskog dijela nadređenog regulatora za $x-osi$
- **upper_i_vy** - gornja granična vrijednost izlaza integracijskog dijela nadređenog regulatora za $y-osi$

- **lower_i_vy** - donja granična vrijednost izlaza integracijskog dijela nadređenog regulatora za $y-osi$
- **upper_i_vz** - gornja granična vrijednost izlaza integracijskog dijela nadređenog regulatora za $z-osi$
- **lower_i_vz** - donja granična vrijednost izlaza integracijskog dijela nadređenog regulatora za $z-osi$
- **upper_i_vyaw** - gornja granična vrijednost izlaza integracijskog dijela nadređenog regulatora, orijentacija oko $z-osi$
- **lower_i_vyaw** - donja granična vrijednost izlaza integracijskog dijela nadređenog regulatora, orijentacija oko $z-osi$
- **steady_x** - vrijednost koja se zbraja na upravljačku veličinu x
- **steady_y** - vrijednost koja se zbraja na upravljačku veličinu y
- **steady_z** - vrijednost koja se zbraja na upravljačku veličinu z
- **steady_yaw** - vrijednost koja se zbraja na upravljačku veličinu *yaw*
- **position_controllerX** - zatvaranje(1) ili prekid (0) povratne veze po poziciji ($x-os$)
- **position_controllerY** - zatvaranje(1) ili prekid (0) povratne veze po poziciji ($y-os$)
- **position_controllerZ** - zatvaranje(1) ili prekid (0) povratne veze po poziciji ($z-os$)
- **position_controllerYAW** - zatvaranje(1) ili prekid (0) povratne veze po poziciji (orijentacija oko $z-osi$)
- **velocity_controllerX** - zatvaranje(1) ili prekid (0) povratne veze po brzini ($x-os$)
- **velocity_controllerY** - zatvaranje(1) ili prekid (0) povratne veze po brzini ($y-os$)
- **velocity_controllerZ** - zatvaranje(1) ili prekid (0) povratne veze po brzini ($z-os$)
- **velocity_controllerYAW** - zatvaranje(1) ili prekid (0) povratne veze po brzini (orijentacija $z-os$)
- **r** - faktor zaglađivanja (eng. *smoothing factor*)

Svaki od navedenih ROS parametara osim parametra T može se dinamički rekonfigurirati, tj. moguće ga je promijeniti tijekom rada čvora.

5.5.1.2. Metode

Struktura upravljanja prikazana u poglavlju **??** pokreće se metodom *void run()* i izvršava se frekvencijom $\frac{1}{T}$ gdje je T period uzorkovanja određen ROS parametrom. Svakim prolaskom kroz petlju provjeravaju se podaci u temi *Reference* koja sadržava referentnu poziciju, orijentaciju i zastavicu uključenosti/isključenosti regulatora za svaku os. Na rastući brid zastavice za z os kao početna vrijednost integratora uzima se trenutna vrijednost tj. referentne veličine z. Ako je pozvan servis *takeoff* tada se kao početna vrijednost integratora uzima vrijednost koja je poslana pozivom tog servisa. Sljedi računanje izlaza nadređenog regulatora te zatim izlaza podređenog regulatora. Kako je implementiran paralelan regulator izrazi za računanje P, I i D dijela regulatora dani su izrazom (17), (18) i (19) respektivno. Izlaz iz derivacijskog dijela regulatora filtrira se niskopropusnim filtrom danim izrazom (20).

$$u_p(k) = K_p \cdot e(k) \tag{17}$$

$$u_i(k) = u_i(k-1) + T \cdot K_i \cdot e(k-1) \tag{18}$$

$$u_d(k) = \frac{K_d}{T}(e(k) - e(k-1)) \tag{19}$$

$$y(k) = y(k-1) \cdot (1-r) + u(k) \cdot r \tag{20}$$

Veličina e predstavlja razliku između referentne veličine i mjerene veličine, a parametar r je faktor zaglađivanja. Izlaz iz podređenog regulatora množi se s matricom rotacije R_z danom izrazom (21).

$$R_z(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{21}$$

gdje je ψ orijentacija oko z osi (yaw). Time se koordinatni sustav letjelice transformira u koordinatni sustav sustava za praćenje pokreta OptiTrack. U slučaju da se u temi */Reference* nalazi zastavica koja ukazuje u isključenost regulatora, tada se referentna veličina množi s 500 ili u slučaju z osi s 1000, te joj se dodaje vrijednost *steady_x, steady_y, steady_z,* ili *steady_yaw* ovisno o upravljanoj osi.

5.5.2. Upute za pokretanje paketa

Čvoru se mogu prilikom pokretanja predati već spomenuti ROS parametri i argumenti naredbenog retka. Pokretanje *PI_D_controller* čvora bez parametara dano je u nastavku.

```
rosrun controller PI_D_controller
```

5.6. Povezivanje čvorova

Prošireno korisničko sučelje i upravljanje letjelicom imena "alpha_one" može se pokrenuti nizom sljedećih naredbi:

- `rosrun voice_recognition voice_recognition_node.py`
  - pokretanje glasovnog prepoznavanja
- `rosrun arducopter_control joy_control_arducopter -file calibration/ArducopterPS3Joy.txt _mocap_topic:=alpha_one/Optitrack _takoff_value:=1350 _voice_and_joy:=0 _joy_topic:=/joy`
  - pokretanje upravljanja letjelicom putem igraće palice
- `rosrun arducopter_control voice_control_node.py _drone_names:="alpha_one"`
  - pokretanje glasovnog upravljanja s imenom letjelice "alpha_one"
- `rosrun ACROSS_optitrack MOCAPNode -trackables drone_names.txt`
  - pokretanje čvora za primanje podataka o poziciji letjelice, drone_names.txt je datoteka s imenima letjelice.
- `rosrun ps3joy ps3joy.py`
  - pokretanje čvora za komunikaciju s igraćom palicom
- `rosrun roscopter roscopter_node.py -device=/dev/ttyUSB1 -baudrate=57600 -enable-control=true`
  -pokretanj čvora za uspostavu komunikacije s letjelicom

U datoteci drone_names.txt nalazi se ime letjelice:

`alpha_one`

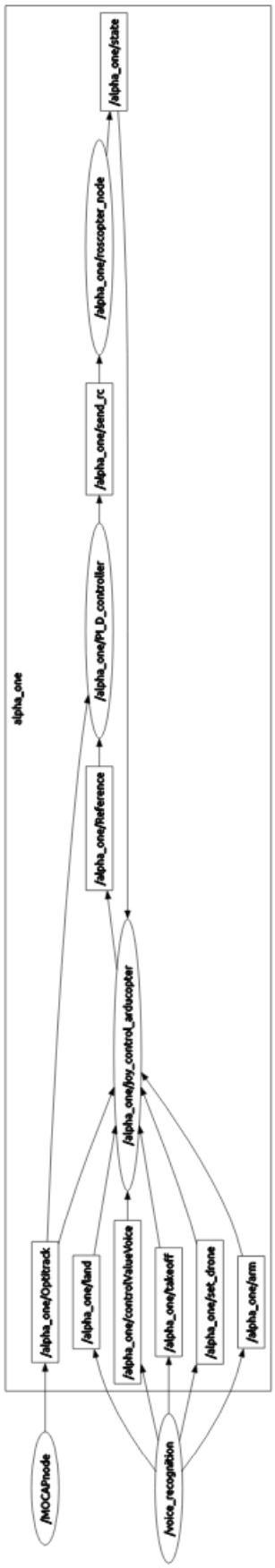
Na taj način pokrenuti su svi čvorovi potrebni za uspostavu komunikacije između proširenog korisničkog sučelja i bespilotne letjelice. Upravljanje dvostrukim robotskim manipulatorom putem kinecta pokreće se na sljedeći način:

- `roslaunch dynamixel_kinect_control controller_manager.launch`
  -inicijalizacija dynamixel motora
- `roslaunch dynamixel_kinect_control arm_controller.launch`
  -stvaranje tema potrebnih za motore
- `rosrun openni_camera openni_node`
  -pokretanje kinect kamere
- `rosrun skeleton_tracker skeleton_joints.py`
  -vizualizacija praćenja kostura
- `rosrun openni_tracker openni_tracker`
  -praćenje kostura

- `rosrun skeleton_tracker TransformToSkeletons.py`
  -pretvorba informacija o položaju kostura u prikladan oblik
- `rosrun dynamixel_kinect_control kinect_pose_detect.py`
  -detekcija poze operatera
- `rosrun dynamixel_kinect_control kinect_arms_controller.py`
  -pokretanje dvostrukog manipulatora

Struktura *launch* datoteka dana je u nastavku:

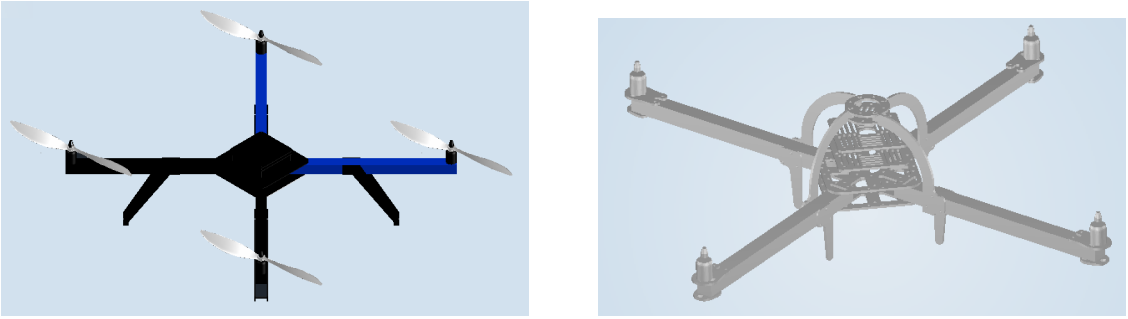
```
controller_manager.launch
<!-- -*- mode: XML -*- -->
<launch>
    <node name="dynamixel_manager" pkg="dynamixel_controllers"
     type="controller_manager.py" required="true" output="screen">
        <rosparam>
            namespace: dxl_manager
            serial_ports:
                pan_tilt_port:
                    port_name: "/dev/ttyUSB0"
                    baud_rate:  250000
                    min_motor_id: 1
                    max_motor_id: 25
                    update_rate: 20
        </rosparam>
    </node>
</launch>

arm_controller.launch
<launch>
    <!-- Start tilt joint controller -->
    <rosparam file="$(find my_dynamixel_tutorial)/tilt.yaml" command="load"/>
    <node name="Joint_controller_spawner"
    pkg="dynamixel_controllers" type="controller_spawner.py"
        args="--manager=dxl_manager
           --port pan_tilt_port
                  JointA1_controller
                  JointB1_controller
                  JointA2_controller
                  JointB2_controller"
           output="screen"/>
<\launch>
```



Povezanost čvorova u ROS-u

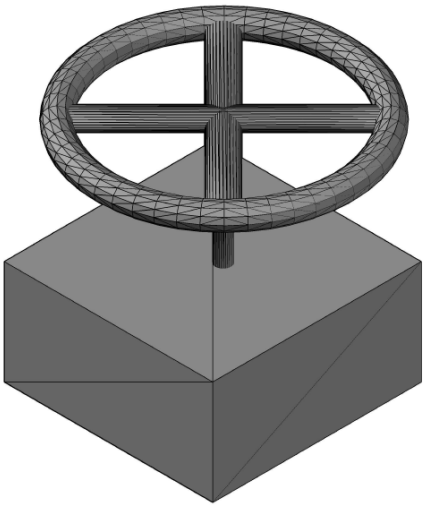
5.7. Virtualno okruženje

Kako bi se testirao rad proširenog korisničkog sučelja dizajnirana je simulacijska scena sa problemom otvaranja/zatvaranja ventila. Virtualno okruženje je Gazebo simulator koji omogućuje brzo testiranje algoritama i dizajniranje robota. Gazebo je odabran iz razloga što pruža mogućnost korištenja ROS-a, što omogućuje testiranje izrađenog proširenog korisničkog sučelja na upravljanju bespilotnom letjelicom s dvostrukim robotskim manipulatorom. Modelirana je bespilotna letjelica ArduCopter tvrtke 3DRobotics i tvrtke jDrones. Model je napravljen u AutoCadu. Na slici 63a nalazi se 3D model bespilotne letjelice tvrtke 3DRobotics, a na slici 63b 3D model bespilotne letjelice tvrtke jDrones. Također su modelirani ventil i članak za dvostruki robotski manipulator te se nalaze na slikama 64, 65, respektivno.



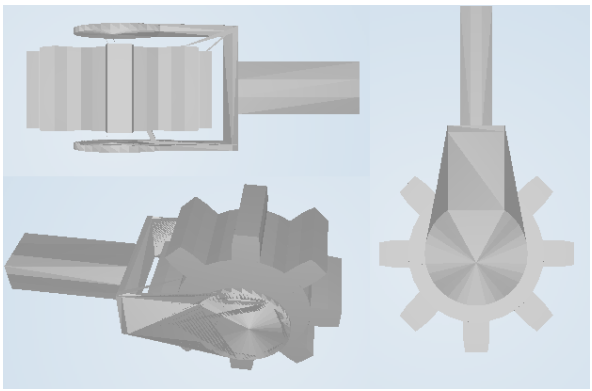
(a) 3D model bespilotne letjelice tvrtke 3DRobotics (b) 3D model bespilotne letjelice tvrtke jDrones

Slika 63: 3D modeli bespilotnih letjelica



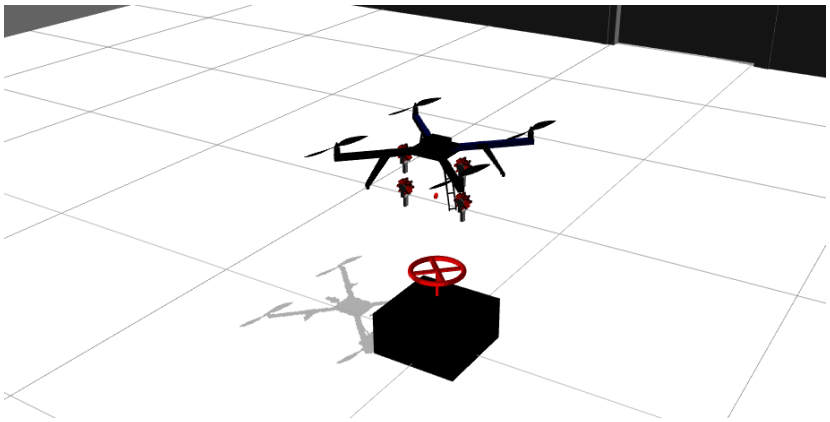
Slika 64: 3D model ventila

Korišten je već gotov paket, *hector_quadrotor*, za simulaciju dinamike tijela bespilotne letjelice, poremećaje uzrokovane vjetrom i aerodinamike propelera. Paket se može preuzeti sa stranice [35]. Korištenjem tog paketa pokušavaju se postići što realniji uvjeti kako bi se u stvarnim uvjetima moglo očekivati slično ponašanje kao u simulatoru.

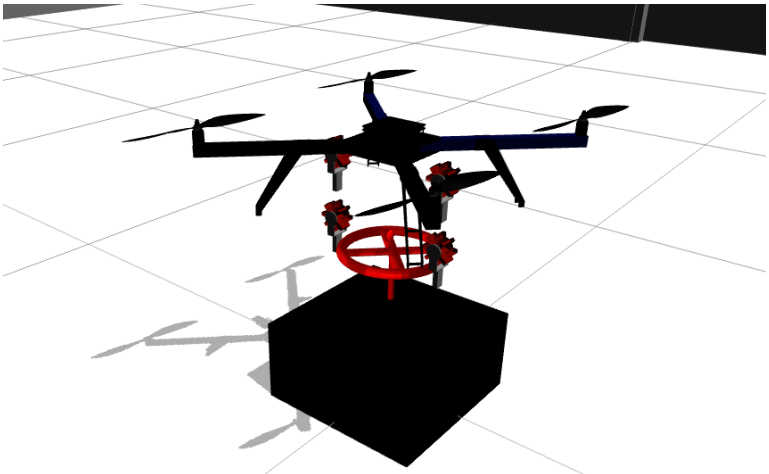


Slika 65: Članak dvostrukog robotskog manipulatora

Simulacijska scena u kojoj se nalazi bespilotna letjelica s dvostrukim manipulatorom i ventil prikazana je na slikama 66 i 67.



Slika 66: Bespilotna letjelica iznad ventila u simulacijskoj sceni



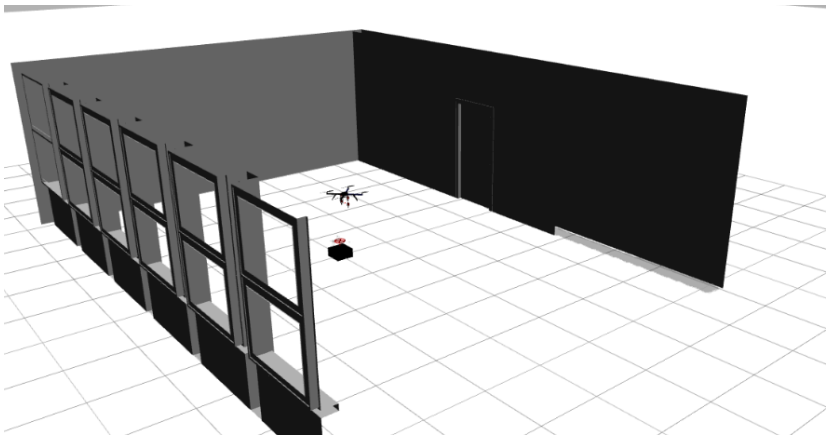
Slika 67: Bespilotna letjelica na ventilu u simulacijskoj sceni

6. Testni scenarij

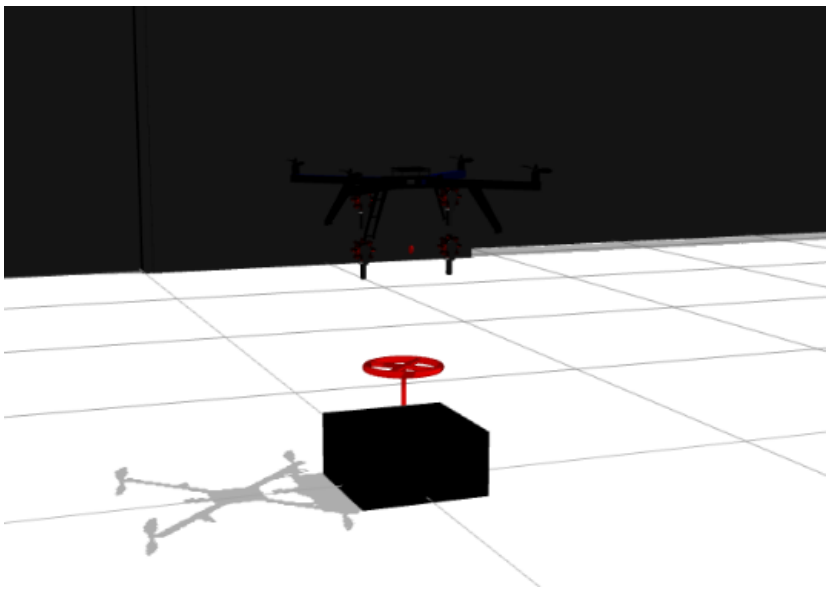
U sklopu ovog rada provedeno je nekoliko eksperimenata. Prvotni eksperimenti odnose se na testiranje proširenog korisničkog sučelja u virtualnom okruženju na problemu otvaranja/zatvaranja ventila. Drugi dio eksperimenata odnosi se na testiranje istog sučelja u realnom okruženju.

6.1. Virtualno okruženje

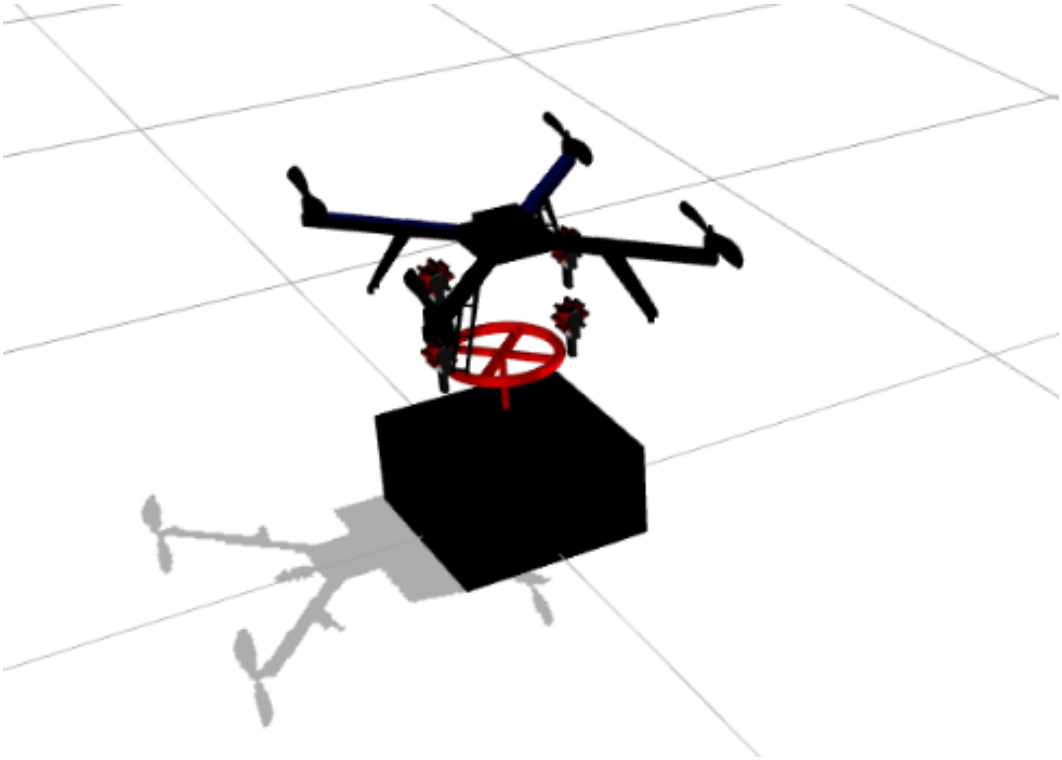
Prošireno korisnicko sucelje testirano je na problemu otvaranja/zatvaranja ventila u virtualnom orkuženju. Simulacijska scena i virtualno okruženje opisani su u poglavlju 5.7. Rezultati eksperimenta snimljeni su te se mogu pogledati na [38]. U nastavku je dano nekoliko slika iz simulacijske scene.



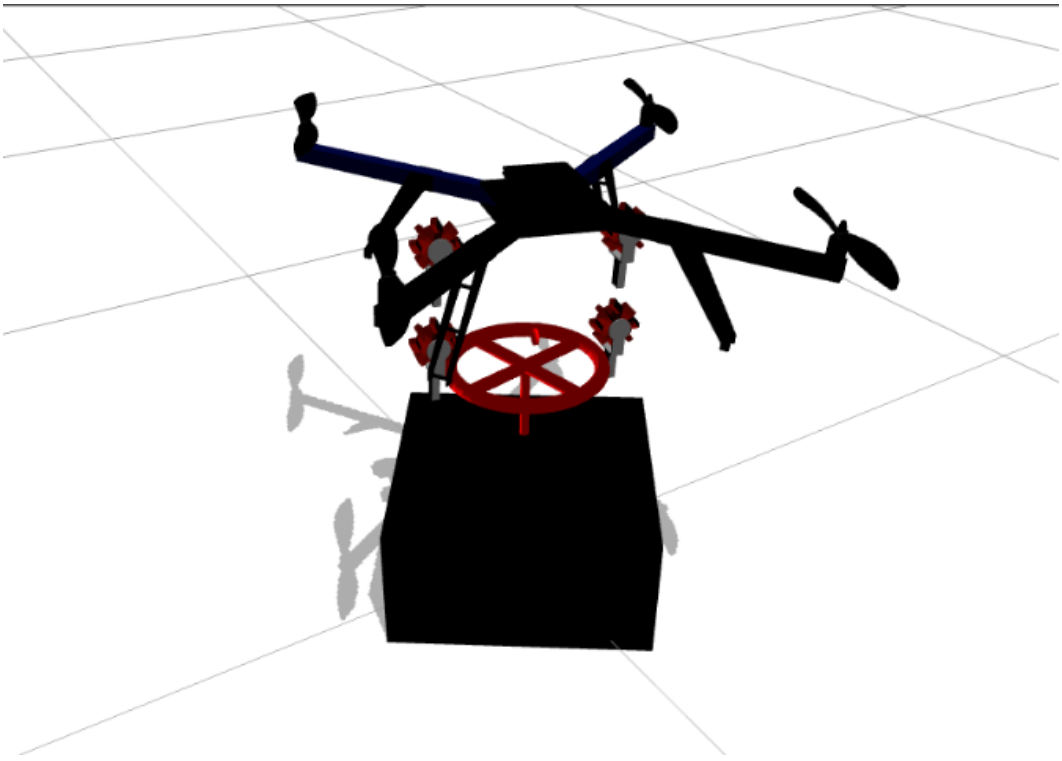
Slika 68: Prikaz simulacijske scene



Slika 69: Pozicioniranje letjelice iznad ventila



Slika 70: Slijetanje na ventila



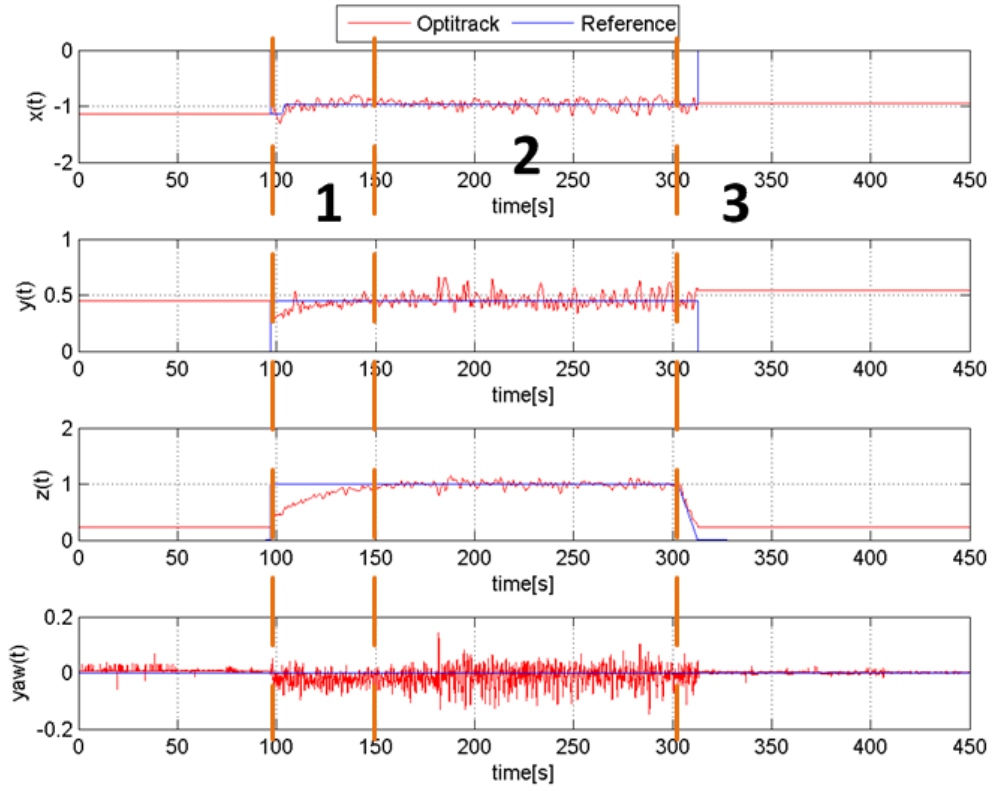
Slika 71: Otvaranje/zatvaranje ventila

Na slici 68 prikazana je simulacijska scena. U simulacijskoj sceni nalazi se bespilotna letjelica i ventil. U poglavlju 5.7 objašnjeno je korišteno virtualno okruženje i izrada simulacijske scene. Slikama 69 i 70 prikazano je pozicioniranje letjelice iznad ventila i slijetanje na ventil. Nakon uspješnog

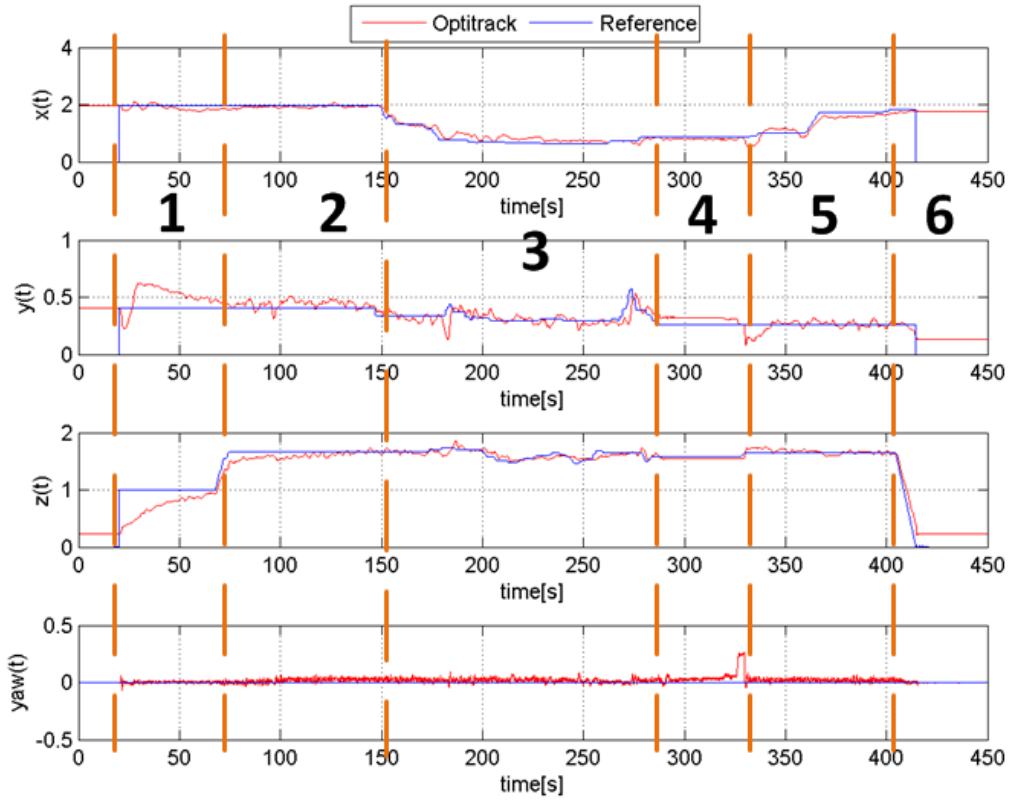
slijetanja na ventil letjelica otvara/zatvara ventil, 71. Problem otvaranja/zatvaranja ventila uspješno je obavljen u virtualnom okruženju te se nakon toga podvrglo testiranje na stvarnom sustavu. Rezultati eksperimenta u virtualnom okruženju mogu se pogledati na [38].

6.2. Realno okruženje

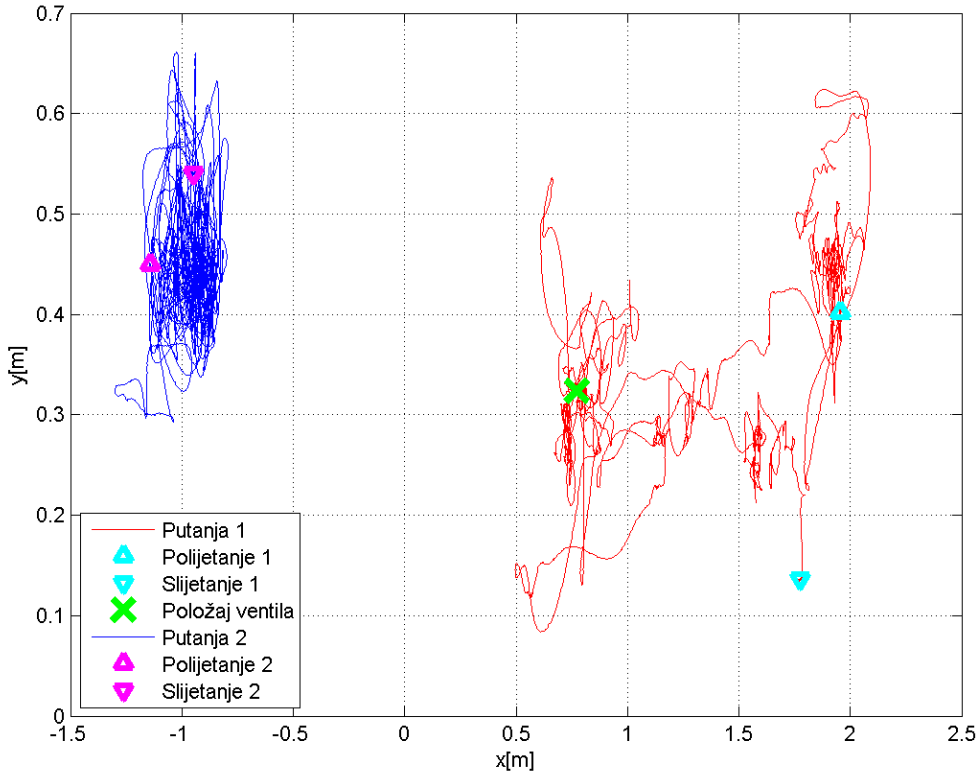
Prošireno korisničko sučelje testirano je u dvije verzije eksperimenta. U prvoj verziji upravljalo se samo letjelicom s dvostrukim manipulatorom pritom gledajući u ventil. Drugu verziju eksperimenta vodila je misao da operater neće moći biti blizu ventila prilikom zavrtanja. U tom slučaju potrebna je vizualna povratna veza o položaju ventila kako bi operater uspješno izvršio zadatak. Vizualna povratna veza izvedena je pomoću dvije kamere: prva kamera nalazi se na letjelici s dvostrukim manipulatorom te pri pozicioniranju iznad ventila snima odozgo čime se dobiva informacija o potrebnom razmaku *grippera* dvostrukog manipulatora, a druga kamera nalazi se na drugoj letjelici koja snima ventil sa strane pružajući informaciju o visini na kojoj letjelica mora biti. U oba slučaja scenarij za letjelicu s dvostrukim manipulatorom je jednak te su stoga prikazani odzivi pozicije letjelica za slučaj korištenja vizualne povratne veze, slike 73 - 74.



Slika 72: Vremenski odziv letjelice s kamerom pri izvođenju eksperimenta zavrtanja ventila



Slika 73: Vremenski odziv letjelice s dvostrukim manipulatorom pri izvođenju eksperimenta zavrtanja ventila



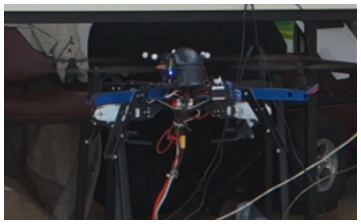
Slika 74: Putanja obje letjelice pri izvođenju eksperimenta zavrtanja ventila

Slike 73 i 72 prikazuju položaj i orijentaciju letjelice s dvostrukim manipulatorom, u daljnjem tekstu prva letjelica i letjelice s kamerom u prostoru, u daljnjem tekstu druga letjelica. Pritom su odzivi podijeljeni u nekoliko dijelova radi lakšeg opisa događaja. U početku se odabire letjelica s dvostrukim manipulatorom te se šalje glasovna naredba za polijetanje. U dijelu **1** odziva 73 prva letjelica polijeće. Nakon što je prva letjelica postavljena na željenu visinu pomoću igraće palice, ona ostaje tamo, dio **2** odziva 73. Za to vrijeme odabire se druga letjelica. U dijelu **1** odziva 72 druga letjelica polijeće te se u dijelu **2** smiruje i zadržava poziciju. Zadaća druge letjelice je snimati ventil kako bi operater što točnije navodio prvu letjelicu prema ventilu te kako bi ga lakše uhvatio dvostrukim manipulatorom. U **3.** dijelu odziva 73 odabrana je prva letjelica te se navodi prema ventilu te započinje upravljanje dvostrukim manipulatorom kako bi se postavio u dobar položaj za prilaz ventilu. Nakon pozicioniranja iznad ventila, što se događa u trenutku $t \approx 200s$, operater pokušava sletjeti na ventil navodeći se slikama s obje letjelice istovremeno upravljajući letjelicom putem igraće palice i dvostrukim manipulatorom putem Kinecta. U **4.** dijelu slijetanje na ventil je uspješno te se upravljanje prebacuje na drugu letjelicu. Vizualna povratna veza s druge letjelice nije potrebna nakon slijetanja na ventil pa se šalje zahtjev za slijetanjem koje se može vidjeti u **3.** dijelu odziva 72. Nakon toga upravljanje se prebacuje na prvu letjelicu te se igraćom palicom šalje naredba zakretanja prve letjelice što se može vidjeti na kraju **4.** dijela odziva. Uspješno zavrnut ventil znači uspješno obavljenu misiju pa se u **5.** dijelu prva letjelica odvaja od ventila te se pozicionira na sigurno mjesto za slijetanje, a pritom operater postavlja dvostruki manipulator u položaj siguran za let. U **6.** dijelu letjelica slijeće čime je zadatak uspješno obavljen te nije izgubljena niti jedna letjelica.

Slika 74 prikazuje putanje obje letjelice u prostoru. Pritom je letjelica s dvostrukim manipulatorom označena jedinicom, a letjelica s kamerom dvojkom u legendi. Iz putanje i odziva 72 može se vidjeti kako letjelica s kamerom ima veće oscilacije oko referentne pozicije od onih opisanih u **??**. Razlog tome je što se eksperiment izvodio s dvije letjelice. Letjelica s dvostrukim manipulatorom nalazila se na većoj visini od letjelice s kamerom te su iz tog razloga zračne struje destabilizirale letjelicu s kamerom zbog čega je u prvom redu došlo do oscilacija.

Na slici 75 prikazana je eksperimentalna izvedba problema otvaranja/zatvaranja ventila koja se u cijelosti može pogledati na [38]. Problem je podijeljen u nekoliko izvedbenih cjelina: polijetanje (slike 75a i 75b), pozicioniranje iznad ventila i slijetanje na ventil (slike 75c - 75e), otvaranje i zatvaranje ventila (slike 75f - 75i) te odvajanje od ventila i slijetanje (slike 75j - 75l).

(a) Početak polijetanja



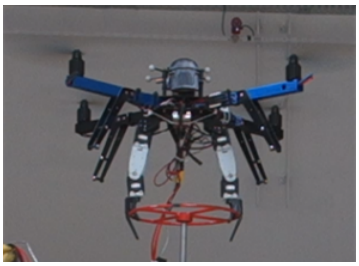
(b) Kraj polijetanja

(c) Približavanje ventilu

(d) Upravljanje manipulatorom

(e) Slijetanje na ventil

(f) Zatvaranje ventila



(g) Vraćanje u početni položaj

(h) Otvaranje ventila

(i) Vraćanje u početni položaj

(j) Odvajanje od ventila

(k) Početak slijetanja

(l) Kraj slijetanja

Slika 75: Eksperimentalna izvedba problema otvaranja/zatvaranja ventila

Zaključak

Prošireno korisničko sučelje razvijeno je kako bi jedan operater mogao upravljati višerobotskim sustavom. Cilj korisničkog sučelja je jednostavno i intuitivno upravljanja svih dijelova sustava. U ovom radu korisničko sučelje sastoji se od tri glavna dijela: glasovno upravljanje, upravljanje pokretima i igraćom palicom. Ispitivanje proširenog korisničkog sučelja provedeno je na problemu otvaranja/zatvaranja ventila. Na taj način testirane su sve komponente korisničkog sučelja te je na taj način letjelica u potpunom kontaktu s okolinom.

Za potrebe ispitivanja izrađene su dvije bespilotne letjelice. Na letjelicu Arducopter tvrtke 3DRobotics ugrađen je dvostuki robotski manipulator. Takav tip manipulatora odabran je zbog velike sličnosti s rukama čovjeka. Upravljanje takvim manipulatorom putem Kinecta pokazalo se iznimno uspješnim jer operater za pokretanje manipulatora koristi svakodnevne pokrete ruku. Montiranjem kamare na drugu bespilotnu letjelicu Arducopter tvrtke jDrones dodana je vizualna povratna veza pomoću koje operater u svakom trenutku dobiva informaciju o položaju letjelice s manipulatorom. Za upravljanje letjelom koristi se glasovno upravljanje u kombinaciji s igraćom palicom. Glasovno upravljanje omogućuje korištenje svih funkcija letjelice, a da su, pri tome, ruke slobodne te se pogled ne miče s područja upravljanja letjelicom. U kombinaciji s igraćom palicom daje odlične rezultate u terminima brzine i točnosti izvođenja zadatka te smanjenog opterećenja operatera prilikom upravljanja takvim višerobotskim sustavom.

Pokazalo se da u simulacijskoj sceni jednopetljasta struktura upravljanja daje dobre rezultate praćenja referentne veličine i brzina odziva sustava, no kod testiranja takve jednopetljaste strukture u realnom okruženju dolazi do odstupanja od tih rezultata. Takva struktura upravljanja zamijenjena je dvopetljastom (kaskadnom) zbog značajne pogreške u stacionarnom stanju te skokova upravljačke veličine. Dvopetljasta struktura upravljanja pokazala je bolje rezultate. Kod pozicioniranja letjelice, pogreška u stacionarnom stanju je manja, nema skokova upravljačke veličine što osigurava mirnije upravljanje letjelicom.

Provedenim testiranjem na problemu otvaranja/zatvaranja ventila uočene su prednosti proširenog korisničkog sučelja. Odabrano upravljanje bespilotnom letjelicom i dvostrukim robotskim manipulatorom putem Kinecta, glasovnog upravljanja i igraće palice omogućilo je operatoru uspješno završiti zadani zadatak zatvaranja ventila. Prethodni eksperimenti, u kojima se koristilo klasično upravljanje s tipkovnicom i upravljačkom palicom, stvaralo je preveliko opterećenje za operatera, koji u konačnici ne bi uspio izvršiti ovako zahtjevan zadatak. Na temelju eksperimentalne analize, može se zaključiti da takvo, prošireno, korisničko sučelje olakšava operateru upravljanje cjelokupnim sustavom. S obzirom na modularnost osmišljenog sustava, kao nastavak ovom radu, predviđa se ugrađivanje dodatnih upravljačkih kanala. Također, aktivno se radi na uvođenju povratne veze stanja operatera, kojom će se utvrditi njegovo psihofizičko stanje, čija će se procjena koristiti kako bi se operateru olakšao teret upravljanja preuzimajući kontrolu nad pojedinim dijelovima sustava.

Literatura

[1] Z. Kovačić, S. Bogdan, V. Krajči. Osnove robotike. Zagreb: Graphis, 2002.

[2] M. Draper, G. Calhoun, H. Ruff, D. Williamson, T. Barry, T. Commanual versus speech input for unmanned aerial vehicle control station operations. Proceedings of the Human Factors and Ergonomics Society, 2003.

[3] Z. Kovačić. Primjena robotskih sustava u modernom vatrogastvu. Vatrogastvo i upravljanje požarima. Vol.III, No. 1, Srpanj 2013, str. 17-18.

[4] M. K. Ravishankar. Efficient Algorithms for Speech Recognition. Doktorska disertacija, Carnegie Mellon University, 1996.

[5] M. Ravishankar. Some results on search complexity vs accuracy. DARPA Speech Recognition Workshop, 1997.

[6] NaturalPoint, Tech Specs, 2014, *A medium volume motion capture camera with excellent precision*, `http://www.naturalpoint.com/optitrack/products/flex-13/specs.html`, 10. travnja 2014.

[7] Matko Orsag, Optitrack Setings, 2013, `https://github.com/morsag/ACROSS_Optitrack/wiki/Optitrack-Settings`, 10. travnja 2014.

[8] Matko Orsag. `https://github.com/morsag/ACROSS_Optitrack`, 10. travnja 2014.

[9] Z. Vukić, Lj. Kuljača. Automatsko upravljanje. Zagreb: Kigen, 2005.

[10] K. Ogata. Modern Control Engineering. Third Edition. New Jersey, USA: Prentice Hall, 1997.

[11] XBee/XBee-Pro RF Modules: Product Manual v1.xEx - 802.15.4 Protocol Minnetonka, Minnesota, USA: Digi International Inc., 2009.

[12] J. Munoz. The ArduPilot Mega Hardware. 11. ožujka 2009. *Technical details of the APM 1 hardware*, `https://code.google.com/p/ardupilot-mega/wiki/Hardware`, 19. travnja 2014.

[13] Completing the ArduPilot Mega hardware. 11. studeni 2012. *Assembling your APM board and IMU*, `https://code.google.com/p/arducopter/wiki/AC2Assembly`, 16. travnja 2014.

[14] The ArduPilot Mega IMU shield Hardware. 12. srpnja 2012. *Technical details of the APM 1 hardware*, `https://code.google.com/p/ardupilot-mega/wiki/IMUHardware`, 19. travnja 2014.

[15] Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V: 8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash. San Jose, California, USA: Atmel Corporation, 2014.

[16] ESC calibration (AKA: setting the end points). 1. travnja 2013. *MultiCopter Instruction Configuration*, `https://code.google.com/p/arducopter/wiki/AC2_ESC`, 17. travnja 2014.

[17] Connecting your RC equipment. 22. studeni 2012. *Tuning Radio*, https://code.google.com/p/arducopter/wiki/Quad_Radio, 17. travnja 2014.

[18] First-time Setup. 9. travnja 2013. *MultiCopter Instructions*, https://code.google.com/p/arducopter/wiki/AC2_First, 17. travnja 2014.

[19] AX-12/ AX 12/ AX-12+/ AX 12+/ AX 12+/ AX-12A: ROBOTIS e-Manual v1.10.00. Robotis, 2010.

[20] ESC 20 Amp with SimonK. https://store.3drobotics.com/products/esc-20-amp-SimonK-1, 19. travnja 2014.

[21] Motor 880Kv AC2836-358 with new prop adapter Datasheet. https://store.3drobotics.com/products/motor-ac2836-358-880kv-1, 19. travnja 2014.

[22] KINECT DEPTH SENSOR EVALUATION FOR COMPUTER VISION APPLICATIONS: Electrical and Computer Engineering Technical Report ECE-TR-6. Aarhus, Danska: Aarhus University, 2012.

[23] Y. A. Girdhar. roscopter. https://code.google.com/p/roscopter/, 18. travnja 2014.

[24] T. Field. openni_tracker. http://wiki.ros.org/openni_tracker, 18. travnja 2014.

[25] B. Pitzer. usb_cam. http://wiki.ros.org/usb_cam, 18. travnja 2014.

[26] M. Quigley, B. Gerkey, K. Watts, B. Gassend. joy. http://wiki.ros.org/joy, 18. travnja 2014.

[27] B. Gassend, M. Wise. ps3joy. http://wiki.ros.org/ps3joy, 18. travnja 2014.

[28] gstreamer. http://gstreamer.freedesktop.org/, 18. travnja 2014.

[29] CMU pocketsphinx. http://cmusphinx.sourceforge.net/, 18. travnja 2014.

[30] pygtk. http://www.pygtk.org/, 18. travnja 2014.

[31] Motor 850Kv AC2830-358 Datasheet. https://store.3drobotics.com/products/motor-ac2830-358-850kv, 19. travnja 2014.

[32] M. Monnajemi. Ardrone Autonomy. https://github.com/AutonomyLab/ardrone_autonomy, 18. travnja 2014.

[33] Logitech HD Webcam C270 http://logitech-en-amr.custhelp.com/app/answers/detail/a_id/17556/~/logitech-hd-webcam-c270-technical-specifications, 18. travnja 2014.

[34] Turtlebeach P11 Earforce http://www.turtlebeach.com/support/entry/830517358/, 19. travnja 2014.

[35] J. Meyer, S. Kohlbrecher. Hector Quadrotor http://wiki.ros.org/hector_quadrotor, 19. travnja 2014.

[36] Parrot ArDrone http://www.ardrone-flyers.com/ar-drone-specs.html, 19. travnja 2014.

[37] Lmtool http://www.speech.cs.cmu.edu/tools/lmtool-new.html, 19. travnja 2014.

[38] Valve turning using dual arm manipulator. https://vimeo.com/93321843, 19. travnja 2014.

Sažetak

Autori: Marko Car, Antun Ivanović

Ime rada: Prošireno korisničko sučelje za upravljanje robotskim manipulatorom u letu

U ovom radu opisan je razvoj proširenog korisničkog sučelja za upravljanje robotskim manipulatorom. Pod pojmom prošireno korisničko sučelje podrazumijeva se spajanje različitih upravljačkih jedinica u jednu cjelinu. Takvim pristupom postiže se jednostavno i intuitivno upravljanje pojedinim dijelovima sustava. Izrađeno sučelje sastoji od tri glavna dijela: mikrofon za glasovno upravljanje, što podrazumijeva odabir letjelica te promjenu postavki leta; Kinect za upravljanje putem kojega se pokreti tijela preslikavaju na dvostruki robotskim manipulator; te bežična igraća palica kao intuitivan pristup upravljanja letjelicom. Fuzijom svih elemenata omogućuje se jednom operateru da upravlja dvostrukim robotskim manipulatorom i bespilotnom letjelicom istovremeno.

U sklopu rada opisana je bespilotna letjelica ArduCopter, princip rada bespilotnih letjelica s četiri pogonska motora, kalibracija ESC sklopova, akcelerometra i magnetometra. U opisu letjelice prikazano je ispravno spajanje elektronike letjelice te komunikacijskog uređaja XBee za koji je naveden postupak promjene postavki. Opisan je sustav OptiTrack za praćenje kretanja letjelice te njegova kalibracija. Opisan je uređaj Kinect korišten za detekciju pokreta operatera te povezivanje uređaja s računalom. Opisan je i sustav prepoznavanja govora te su objašnjeni algoritmi prepoznavanja govora.

Prikazani su postupci realizacije svih dijelova ovog sustava, kao što su odabir naredbi i upravljačkih pokreta, ispitivanje i odabir strukture upravljanja, izrada programske podrške te povezivanje svih dijelova sustava s robotskim operacijskim sustavom (ROS). Izrađeni su paketi za upravljanje svim dijelovima sustava te je opisana tehnička izvedba samih paketa. Za provjeru rada sučelja osmišljen je eksperiment zavrtanja ventila. Provedena je eksperimentalna provjera sučelja u virtualnom okruženju nakon čega je sustav ispitan na realnom sustavu. Za potrebe eksperimentalne provjere u realnom okruženju na letjelicu je ugrađen dvostruki robotski manipulator. Ostvareni cilj eksperimenta bio je sletjeti na ventil pomoću dvostrukog robotskog manipulatora te ga zakrenuti.

Ključne riječi: robotski manipulator, bespilotna letjelica, prošireno korisničko sučelje,

Summary

Authors: Marko Car, Antun Ivanović

Paper: Augmented human machine interface for aerial manipulators

This paper presents an augmented human machine interface with multiple control inputs, used to operate a single aerial manipulator and a fleet of aerial robots. The term augmented human machine interface implies a combination of different control peripherals into a single, multifunctional interface. This approach provides an intuitive control for all parts of the system, and therefore for the whole system. There are three key parts of the interface: a microphone used for voice control, that enables straightforward selection of unmanned aerial vehicles and flight mode selection; Kinect which enables a dual arm manipulator to mimic operator's arm motion through skeleton tracking; and a wireless joystick as an intuitive control device for unmanned aerial vehicles. Combining these elements allows a single operator to control both the dual arm manipulator and unmanned aerial vehicle at the same time.

The paper provides the list of devices used in the implementation, as well as a brief description of how each device is used. Quadrotor mathematical model, specific for the Arducopter quadrotor is provided, along with basic procedures needed for a successful flight, that include calibration of ESC, accelerometer and magnetometer, along with a correct way of wiring vehicle electronics and XBee communication device. Optitrack system used to track the vehicle in an indoor environment is described, with necessary steps to set it up and calibrate for accurate tracking. The Kinect based gesture tracking algorithm is devised and presented, along with Markov Chain based voice recognition algorithm used for voice control.

Procedures to successfully implement such a system are explained for all the system parts(i.e. choice of commands, moves and gestures, control structure, development of software and fusion of all the parts of the system through ROS). Software packages implemented for each part of the system are provided within technical data. Valve twist experiment is derived to examine the performance of the interface. First, the interface was tested within a virtual environment. Afterwards, the experiment was performed in real world experimental environment. Dual arm manipulator was installed on an unmanned aerial vehicle in order to land on valve, perch onto it and turn it. After thorough experimental verification, in both virtual and real world environment, the results of both experiments are compared and discussed.

Keywords: dual arm manipulator, unmanned aerial vehicle, augmented human machine interface