SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1349

INTEGRACIJA BENTHOS RONILICE I SUSTAVA ZA POZICIONIRANJE

Jerko Tolić

Zagreb, srpanj 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 16. ožujka 2016.

DIPLOMSKI ZADATAK br. 1349

Pristupnik: **Jerko Tolić (0036469526)**
Studij: Elektrotehnika i informacijska tehnologija
Profil: Automatika

Zadatak: **Integracija Benthos ronilice i sustava za pozicioniranje**

Opis zadatka:

Benthos ronilica je daljinski upravljana ronilica za koju postoji upravljački program realiziran u Labview programskom paketu. Desert Star akustički sustav za pozicioniranje omogućuje određivanje lokacije objekata pod vodom. U sklopu diplomskog zadatka potrebno je provjeriti funkcionalnost postojećeg programa za upravljanje Benthos ronilicom te ga modificirati. Ujedno, potrebno je istražiti mogućnosti integriranja Desert Star sustava za pozicioniranje s Benthos ronilicom kako bi se postavili temelji za automatsko upravljanje pozicijom. U sklopu rada potrebno je testirati razvijene programe za upravljanje kao i analizirati kvalitetu akustičkog sustava za pozicioniranje.

Zadatak uručen pristupniku: 18. ožujka 2016.
Rok za predaju rada: 1. srpnja 2016.

Mentor:
Doc. dr. sc. Nikola Mišković

Djelovođa:
Izv. prof. dr. sc. Mario Vašak

Predsjednik odbora za diplomski rad profila:
Prof. dr. sc. Stjepan Bogdan

Ovim putem iskreno se zahvaljujem svima koji su mi svojim sugestijama pomogli pri izradi ovog rada. Posebno se zahvaljujem svojem mentoru,izv.prof.dr.sc.Nikoli Miškoviću, na ukazanom povjerenju, razumijevanju i potpori kroz cjelokupni studij. Također, želio bih se zahvaliti i mag.ing.el. Zdravku Eškinji na pomoći i savjetima oko izrade ovog rada.

Na kraju,hvala mojim roditeljima i braći na pruženoj ljubavi i potpori.

Sadržaj

1. Uvod

U sklopu diplomskog rada potrebno je integrirati *Desert* Star sustav za pozicioniranje s Benthos ronilicom, te realizirati automatsko upravljanje pozicijom ronilice. Benthos ronilica je daljinski upravljana ronilica za koju već postoji upravljački program realiziran u *LabVIEW* programskom paketu, te će iz tog razloga i automatsko upravljanje biti realizirano u *LabVIEWu*. *Desert Star PILOT System* akustički je sustav za pozicioniranje razvijen od strane *Desert Star Systems* LLC, tvrtke iz Sjedinjenih Američkih Država. *PILOT System* omogućuje određivanje lokacije objekata pod vodom uz pomoć tri fiksna, te jednog mobilnog primopredajnika i površinske stanice. Također, u sklopu rada potrebno je doći do matematičkog modela ronilice obradom videa, te je potrebno napraviti Kalmanov filtar za estimaciju pozicije kako bi se ronilicom moglo upravljati i kada mjerenja iz sustava za pozicioniranje nisu dostupna. U radu će se detaljno opisati sustav za pozicioniranje, te pojedini dijelovi sustava. Opisat će se i Benthos ronilica, te će biti opisani svi kodovi korišteni pri realizaciji ovog rada.

2. Desert Star PILOT System

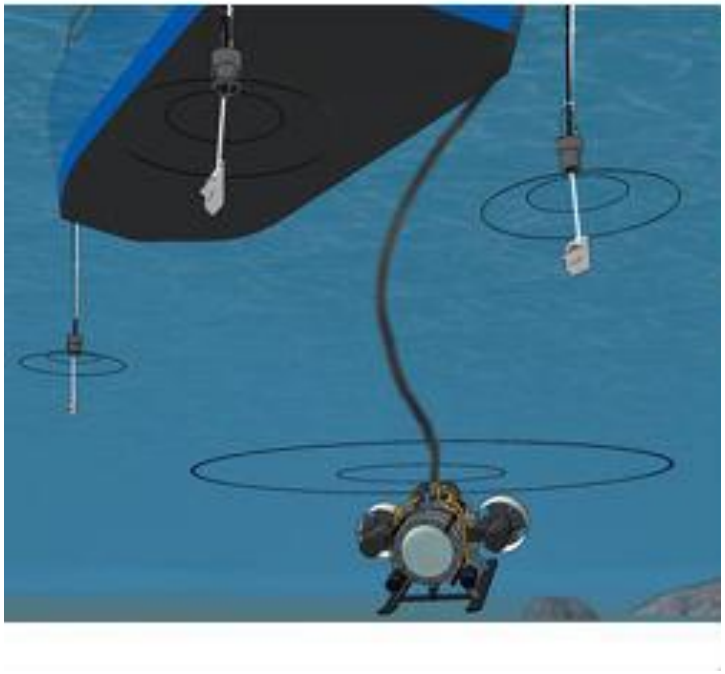
Desert Star PILOT System akustični je sustav za određivanje pozicije objekta pod vodom, te spada u kategoriju SBL (engl. *Short Baseline*) akustičnih sustava za pozicioniranje [1]. Uz SBL sustave postoje još i LBL (engl. *Long Baseline*), te USBL (engl. *Ultra Short Baseline*) sustavi za pozicioniranje, te svaki od navedenih ima svoje prednosti i mane. SBL sustavi se upotrebljavaju u slučaju kada USBL tehnologija ne daje dovoljno precizne rezultate, a postavljanje fiksnih primopredajnika potrebnih za LBL sustav nije praktično [1].

PILOT System prvenstveno služi za praćenje ROVa (engl. *Remotely operated underwater vehicle*) i drugih podvodnih vozila uz pomoć zvuka visoke frekvencije (34 kHZ- 42 kHZ) [2]. Zvuk predstavlja najbolje rješenje za podvodnu lokalizaciju i komunikaciju s obzirom da odlično propagira pod vodom [3] i s obzirom da je korištenje elektromagnetskih valova i svjetla pod vodom onemogućeno [2]. Glavni nedostatak korištenja zvuka pri lokalizaciji jest refleksija signala koja se može podijeliti s obzirom na uzrok refleksije. Tako postoji refleksija uzrokovana značajnom razlikom dva susjedna sloja u vertikalnom profilu vode, te refleksija nastala u ambijentu [3].

2.1. Način rada i karakteristike PILOT sustava

PILOT System koristi tri fiksna primopredajnika (*DSS Standard Sonar Transducer*), te jedan mobilni primopredajnik (*TLT-3*) pričvršćen za ROV ili neko drugo podvodno vozilo. Tri fiksna primopredajnika povezana su kabelom za površinsku stanicu, te se s broda uranjaju u vodu kao što je prikazano na slici 1. Primopredajnik 1 i primopredajnik 2 čine primarni *baseline*, dok primopredajnik 1 i primopredajnik 3 čine sekundarni *baseline*, kao što je prikazano na slici 3. Poželjno je da je duljina primarnog *baselinea* veća od duljine sekundarnog *baselinea,* te da je kut između njih što bliži pravom kutu [2]. Ukoliko su tri primopredajnika postavljena tako da čine liniju, odnosno da je kut između *baselineova* približno 0° ili 180°, *PILOT* sustav neće moći precizno odrediti poziciju objekta budući da ne može odrediti da li se objekt nalazi lijevo ili desno od linije. Također, zbog što bolje preciznosti, mobilni primopredajnik potrebno je postaviti

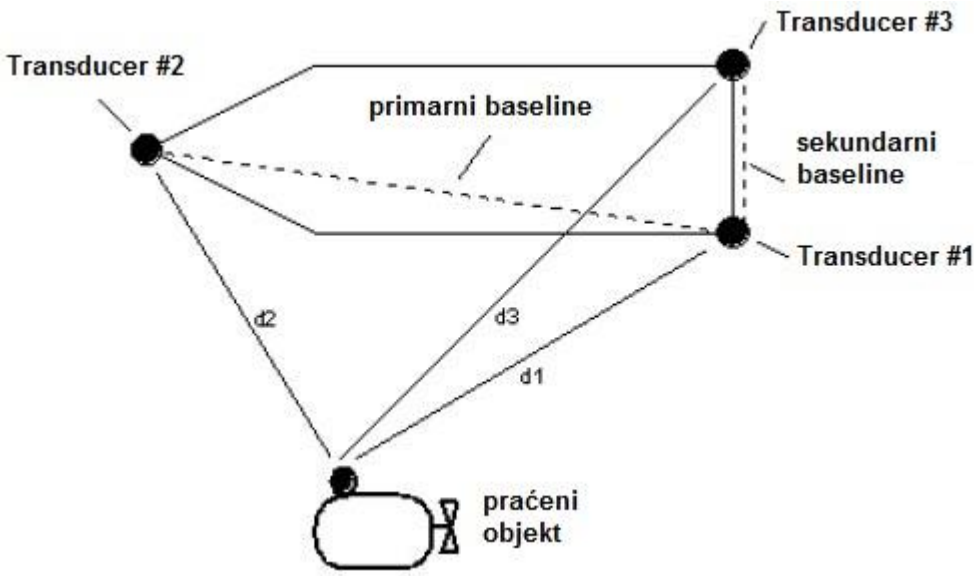
što dalje od propulzora ronilice ili bilo kojeg drugog izvora zvuka, te podalje od površine od koje se zvuk može reflektirati (slika 2).



Slika 1. PILOT System [1]

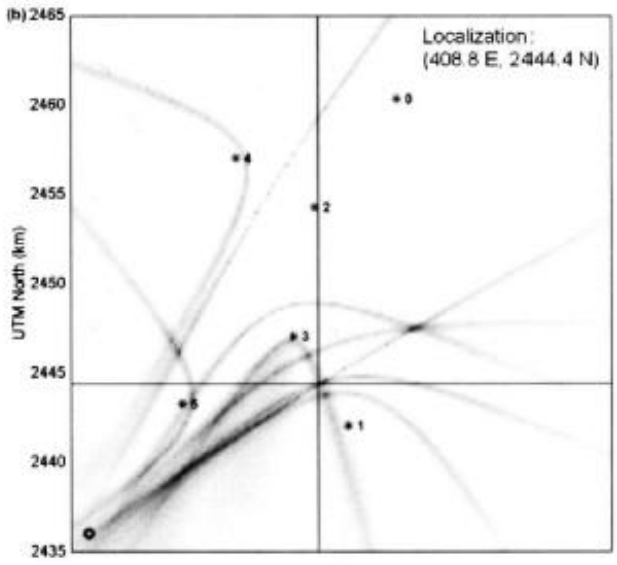
Slika 2. Pravilno postavljanje TLT-3 primopredajnika

Kako bi se odredila pozicija praćenog objekta, površinska stanica preko prvog primopredajnika šalje signal u trajanju dva pinga [2]. Signal propagira kroz vodu te dolazi do mobilnog primopredajnika koji potom odgovara porukom u kojoj su sadržani podaci o trenutnoj dubini dobiveni iz dubinomjera. Navedenu poruku primaju sva tri fiksna primopredajnika, a površinska stanica mjeri vremena između poslanog signala i primljenih signala. Budući je brzina zvuka u vodi poznata, iz dobivenih vremena lako se izračunaju udaljenosti mobilnog od fiksnih primopredajnika (d1,d2 i d3 prikazani na slici 3).



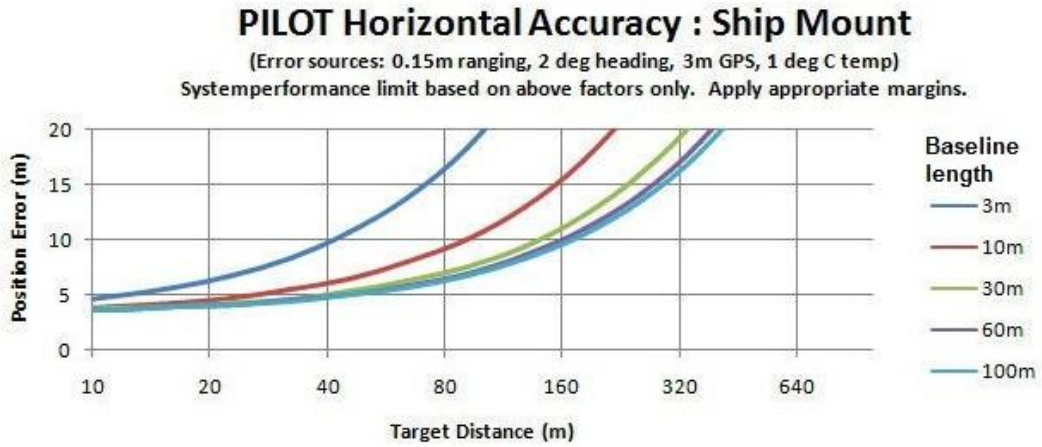
Slika 3. Pravilno postavljanje primopredajnika [2]

Također, iz razlike vremena dolaska signala izračunaju se parametri hiperbole s fokusom u fiksnim primopredajnicima [4]. Sjecišta izračunatih hiperbola određuju poziciju praćenog objekta, kao što je i prikazano na slici 4. Parametri hiperbola izrazito su osjetljivi na preciznost mjerenja razlike vremena dolaska signala pa jako ovise o udaljenosti između fiksnih primopredajnika, frekvenciji uzorkovanja signala i kvaliteti algoritma za izračun navede razlike vremena [3].



Slika 4. Izračunate hiperbole [5]

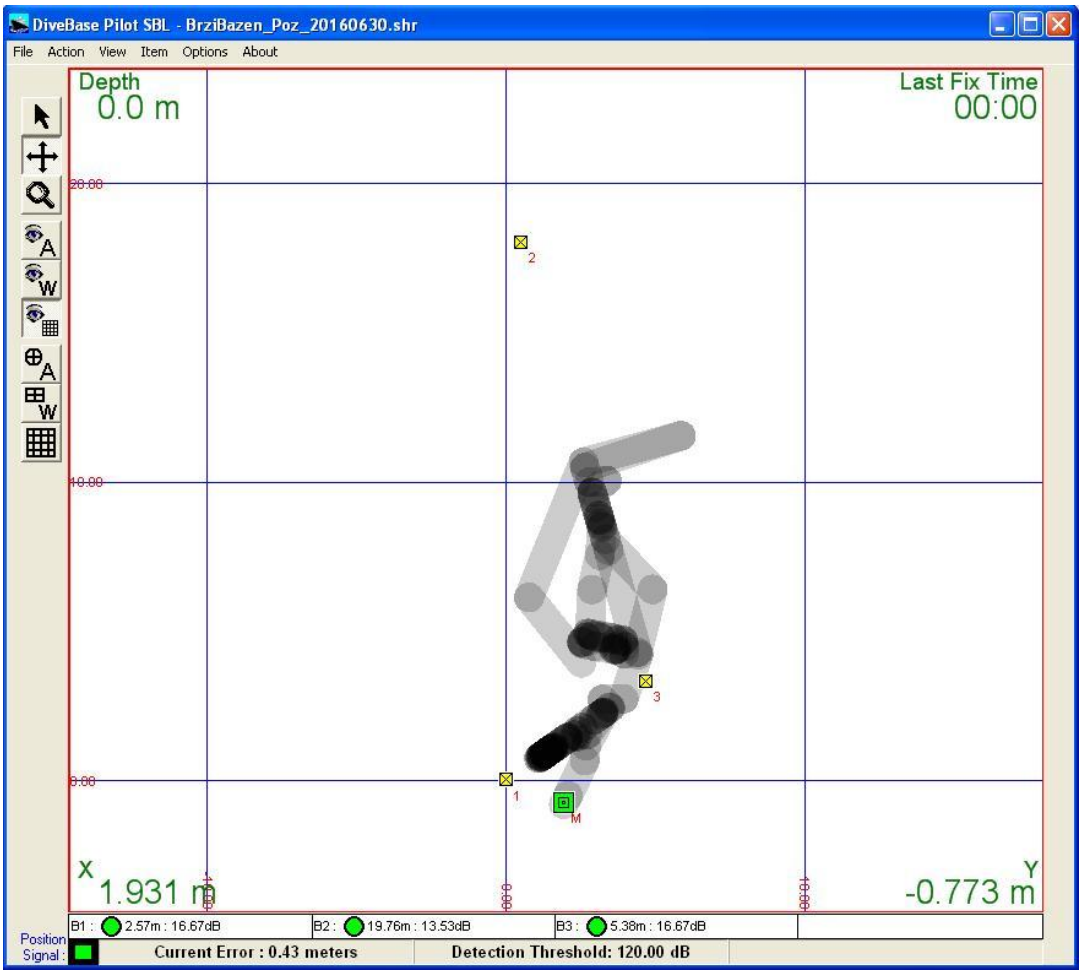
Preciznost mjerenja udaljenosti praćenog objekta kod *PILOT* sustava neovisna je o udaljenosti između primopredajnika, te iznosi +/-0.15 metara. Međutim, preciznost mjerenja orijentacije objekta je veća što je udaljenost između fiksnih primopredajnika veća. Tako za primopredajnike udaljene 3 metra, preciznost iznosi +/-5°, a za primopredajnike udaljene 30 metara iznosi +/-0.5° [2]. Slika 5 prikazuje ovisnost pogreške pozicije o udaljenosti praćenog objekta za različite udaljenosti između primopredajnika. Na slici se može primijetiti da je pogreška pozicije veća za manje duljine *baselinea*, te da pogreška pozicije raste s udaljenošću praćenog objekta od primopredajnika.



Slika 5. Ovisnost preciznosti o udaljenosti praćenog objekta i duljini baselinea [1]

2.2. DiveBase software

DiveBase je programsko okruženje koje se koristi za skupljanje svih podataka iz površinske stanice, podešavanje postavki *PILOT* sustava i prikaz pozicije praćenog objekta. *DiveBase* se može koristiti samo na računalima s Windows XP operacijskim sustavom. Slika ispod (slika 6) prikazuje početni zaslon *DiveBase* programa na kojem se mogu očitati podaci o poziciji objekta (zeleni kvadrat), poziciji primopredajnika (žuti kvadrati), vremenu od zadnjeg osvježavanja mjerenja, te trenutnoj pogrešci.

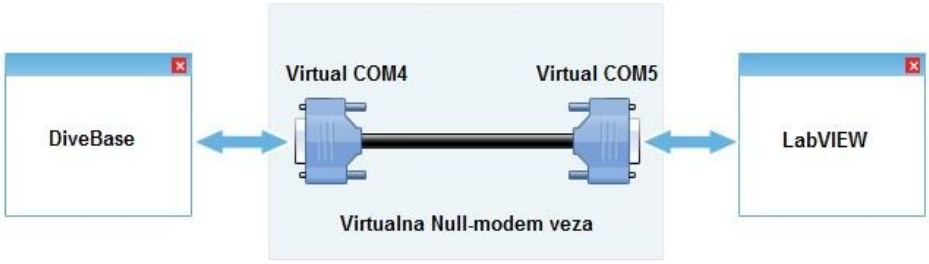


Slika 6. DiveBase software

Postavke *PILOT* sustava, odnosno pozicije fiksnih primopredajnika postavljaju se klikom na *Register and Calibrate Baseline Stations* iz *Action* padajućeg izbornika, a klikom na *Start Real-Time Tracking* započinje praćenje objekta.

2.3. Čitanje podataka iz DiveBasea

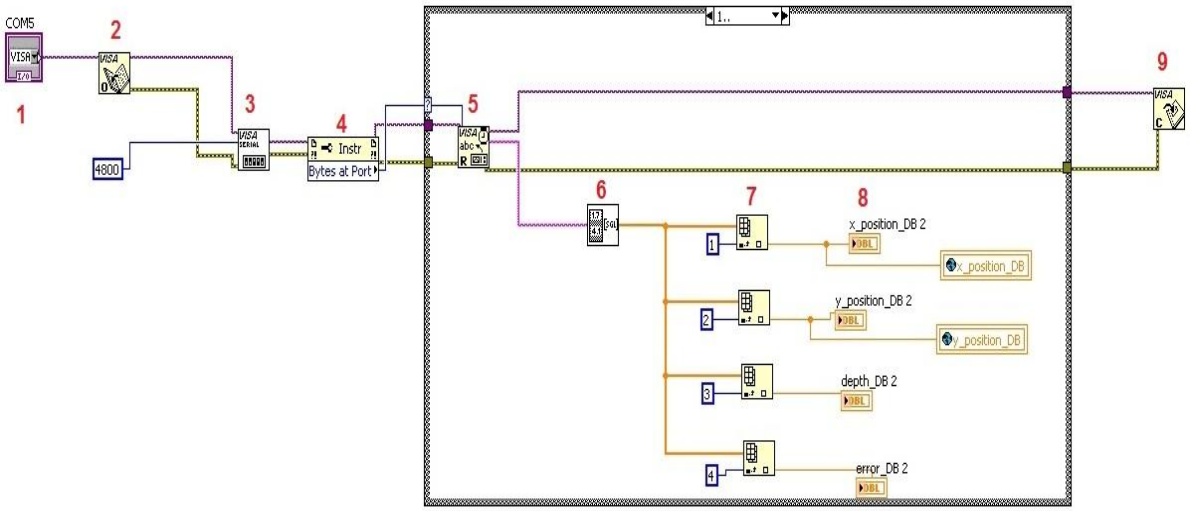
Kako bi se realiziralo automatsko pozicioniranje ronilice, potrebno je omogućiti čitanje podataka iz *DiveBasea*. *DiveBase* nudi mogućnost prebacivanja podataka preko serijskog porta s jednog računala na drugo, no to i nije najzahvalnija opcija budući je nepraktično koristiti dva računala. Puno bolje rješenje je napraviti dva virtualna serijska porta, te ih međusobno upariti. *Null modem emulator* je *open source software* koji nudi mogućnost stvaranja neograničenog broja COM portova, te njihovo međusobno uparivanje. Potrebno je, dakle, instalirati navedeni program, te virtualno upariti dva COM porta. Potom, u *DiveBaseu* treba odabrati jedan od uparenih portova klikom na *Serial Output Com Ports* iz *Options* padajućeg izbornika. Odabrani virtualni port služit će za slanje podataka iz *DiveBasea*. Sljedeće što je potrebno napraviti jest *LabVIEW* VI program za čitanje podataka s drugog virtualnog porta. Shema opisane virtualne veze između portova i korištenih aplikacija dana je na slici ispod.



Slika 7. Shema virtualne Null-modem veze

Slika 8 prikazuje *LabVIEW Block Diagram* programa namijenjenog za čitanje podataka s COM porta. Blok 1 predstavlja *Input/Output Control* u kojem se definira ime porta s kojeg se čitaju podaci. Blok 2 je *VISA Open Function* putem kojeg se otvara komunikacijska linija prema resursu. Sljedeći blok (blok 3) je *VISA Configure Serial Port* , te on služi za inicijalizaciju porta definiranog blokom 1. Blok 4 na svom izlazu daje broj bajtova primljenih preko serijskog porta. Ukoliko je navedeni broj bajtova veći od nule, primljeni podaci se čitaju blokom 5 (*VISA Read*), te se spremaju u *Read Buffer*. Sljedeće što je potrebno napraviti jest

izdvojiti željene podatke o poziciji ronilice iz skupa primljenih podataka. *Extract Numbers.vi* (blok 6) je *LabVIEW* VI program koji pronalazi sve brojeve u zadanom *stringu*, te ih sprema u polje. Blokom 7 (*Index Array*) iz dobivenog polja se izdvajaju podaci o x, y koordinatama ronilice, te podaci o dubini i pogrešci mjerenja, te se spremaju u određene indikatore (blok 8). Blokom 9 (*Visa Close*) zatvara se komunikacija prema resursu.



Slika 8. Read_data.vi

Budući da dubinomjer na dobivenom mobilnom primopredajniku ne radi ispravno, podaci o dubini ronilice se ne uzimaju u obzir prilikom izrade upravljačkog algoritma. Zbog toga će automatsko pozicioniranje ronilice biti realizirano samo u $x - y$ ravnini.

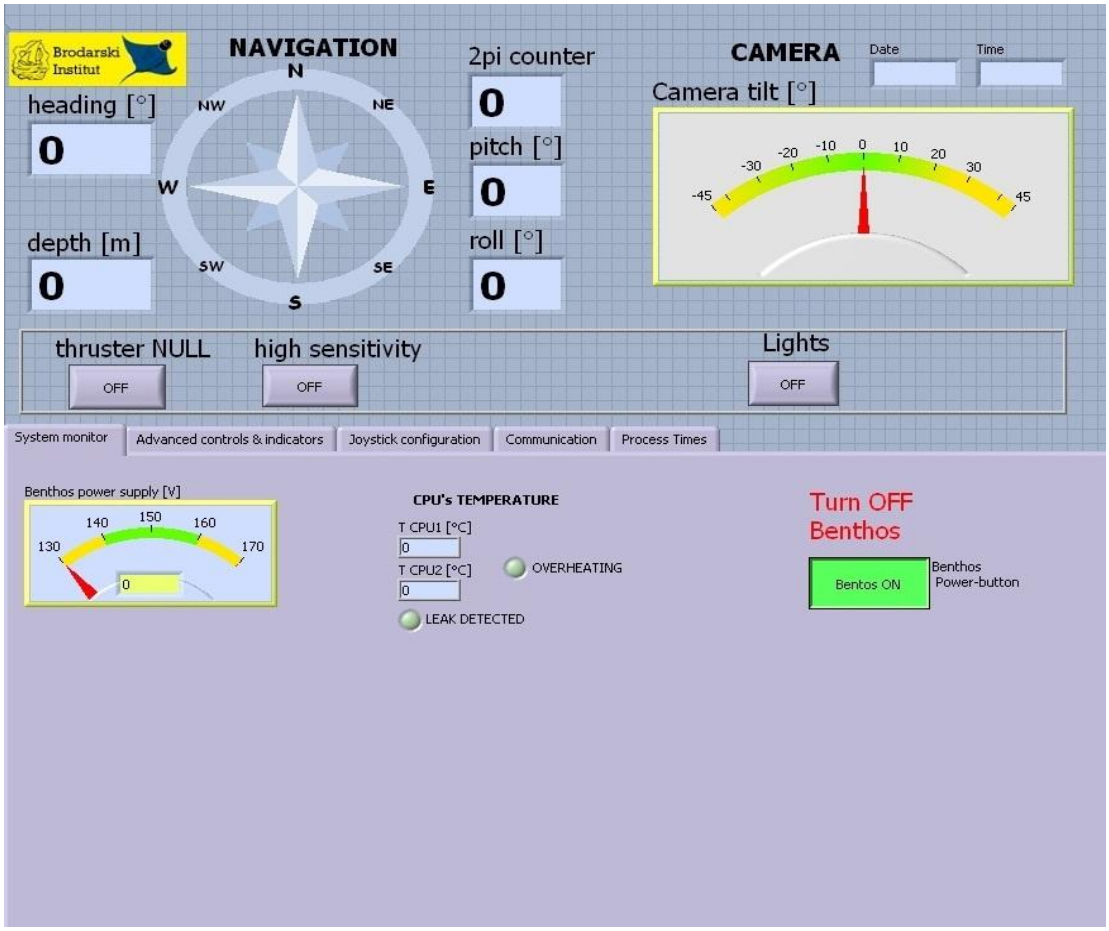
3. Benthos Stingray ronilica

Benthos Stingray je daljinski upravljanja ronilica, proizvod Teledyne Benthos kompanije. Navedena ronilica zbog svojih malih dimenzija (990x457x457 mm) i male težine (32 kg) spada u mini ROV klasu, te se najčešće koristi za inspekciju. Sposobna je ići do 350 metara dubine i postiže unaprijednu brzinu od tri čvora [6]. Pokreću je četiri DC motora snage 373W od kojih se dva koriste za horizontalno gibanje, a po jedan za lateralno i vertikalno gibanje. Ronilica ima četiri stupnja slobode gibanja, napredovanje (engl. *surge*), zanošenje (engl. *sway*), zaranjanje (engl.*heave*), te zaošijanje (engl.*sway*). Također, ronilica je opremljena različitim senzorima kao što su senzor valjanja (engl. *roll*), senzor naginjanja (engl. *pitch*), senzor kursa (engl. *heading*) i senzor dubine. Uz senzore, ronilica je još opremljena i kamerom visoke rezolucije. Napajanje i komunikacija ronilice s površinskom stanicom odvija se preko *tether* kabela. Površinska stanica spojena je na računalo preko RS232 porta.

Slika 9. Benthos Stingray

3.1. Upravljača konzola Benthos ronilice

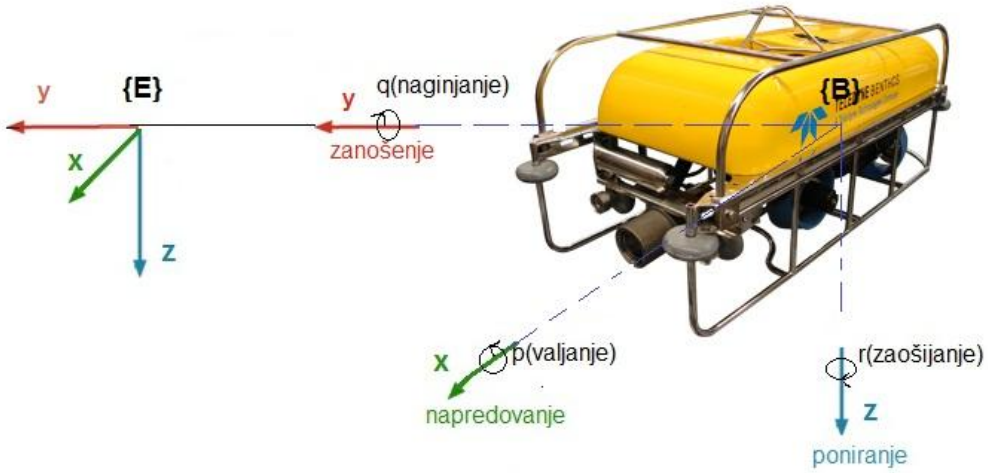
Konzola za Benthos Stingray napravljena je u *LabVIEWu*, te služi za nadzor i upravljanje ronilicom. Prvo što je potrebno napraviti je podesiti modemsku vezu prema uputama iz [10] kako bi ronilica, odnosno površinsku stanica, i računalo uspješno komunicirali. Zatim se pokreće upravljački program *u LabVIEWu,* koji nakon inicijalizacije *joysticka* ulazi u vremensku petlju, koja se poziva svakih 100ms. Unutar jednog poziva, ronilica i računalo razmjenjuju niz *stringova* koristeći UDP (engl. *User Datagram Protocol*) protokol. Ovaj protokol se koristi kada je brzina prijenosa važnija od pouzdanosti prijenosa poruka. Poruke poslane ronilici započinju s BMK#, te završavaju s #KMB, a sadrže naredbe za paljenje, odnosno gašenje svjetla, naredbu za pomicanje kamere, te iznose sila i momenata koji određuju gibanje ronilice. Poruke koje ronilica šalje računalu započinju sa ZMK#, te završavaju s #KMZ, a sadrže informacije dobivene sa senzora ronilice, kao na primjer: dubinu, kurs, nagib ronilice, nagib kamere, temperature procesora, napon napajanja ronilice itd.



Slika 10. Konzola realizirana u LabVIEWu

4. Identifikacija modela ronilice

Kako bi se realiziralo automatsko pozicioniranje ronilice, potrebno je u svakom trenutku znati gdje se ronilica nalazi. *Desert Star PILOT System* daje poziciju ronilice svakih pet sekundi u prosjeku. Međutim, pet sekundi bez informacije o poziciji ronilice je relativno dug period, te može rezultirati oštećenjem ili gubitkom ronilice. Ovaj problem može se riješiti implementacijom Kalmanovog filtra za estimaciju pozicije ronilice. No kako bi se Kalmanov filtar uopće implementirao, potrebno je doći do modela ronilice. Pri izradi modela ronilice definiraju se dva koordinatna sustava, jedan sustav vezan uz Zemlju $\{E\}$ i drugi vezan uz ronilicu $\{B\}$, kao što je prikazano na slici ispod.



Slika 11. Položaj koordinatnih sustava

Koordinatne osi $\{B\}$ i $\{E\}$ sustava međusobno se poklapaju, te su odabrane prema pravilu desne ruke.

Tablica 1 prikazuje notaciju koja se koristi kod plovila, a preuzeta je iz [7].

Tablica 1. Notacija za plovila

Stupanj slobode	Napredovanje	Zanošenje	Poniranje	Valjanje	Naginjanje	Zaošijanje	Def. u
Brzine	u	v	w	p	q	r	$\{B\}$
Pozicije i kutevi	x	y	z	ϕ	θ	ψ	$\{E\}$
Sile i momenti	X	Y	Z	K	M	N	$\{B\}$

Pozicije i kutovi definirani u $\{E\}$ koordinatnom sustavu formiraju vektor η:

$$\eta = [x\ y\ z\ \phi\ \theta\ \psi]^T \tag{4.1}$$

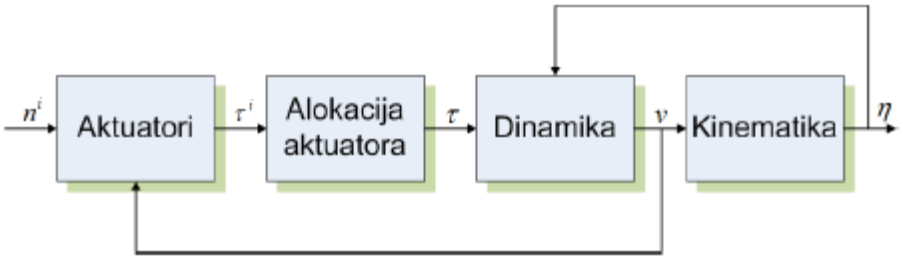
Linearne i rotacijske brzine definirane u $\{B\}$ sustavu formiraju vektor v:

$$v = [u\ v\ w\ p\ q\ r]^T \tag{4.2}$$

Također, sile i momenti definirani u $\{E\}$ koordinatnom sustavu formiraju vektor τ:

$$\tau = [X\ Y\ Z\ K\ M\ N]^T \tag{4.3}$$

Osnovni matematički model ronilice prikazan je na slici 12, te ga se može podijeliti na četiri bloka: aktuatori, alokacija aktuatora, dinamički model i kinematički model. Pojedini blok opisuje međusobnu ovisnost ranije navedenih vektora. Alokacija aktuatora je već ranije napravljena pa će u radu biti opisani samo dinamički i kinematički model.



Slika 12. Blok dijagram matematičkog modela [8]

4.1. Dinamički model ronilice

Dinamički model definira ovisnost brzina o silama i momentima koji djeluju na ronilicu. Potpuni dinamički model ronilice je nelinearan i spregnut, te izrazito složen. Međutim, potpuni model se može pojednostavniti zbog gibanja ronilice malim brzinama. Budući da Benthos ronilica ima četiri stupnja slobode gibanja, dobije se dinamički model opisan s četiri diferencijalne jednadžbe prvog reda :

$$\alpha_u \dot{u} + \beta_u u = X \tag{4.1.1}$$

$$\alpha_v \dot{v} + \beta_v v = Y \tag{4.1.2}$$

$$\alpha_w \dot{w} + \beta_w w = Z \tag{4.1.3}$$

$$\alpha_r \dot{r} + \beta_r r = N \tag{4.1.4}$$

Kao što je već ranije istaknuto, automatsko pozicioniranje bit će realizirano samo u $x - y$ ravnini. Također, zbog jednostavnosti, ronilicom će se upravljati samo pomoću dva aktuatora koja služe za napredovanje i zaošijanje ronilice. Zbog navedenog, izrazi (4.1.2) i (4.1.3) neće se uzimati u obzir.

4.2. Kinematički model ronilice

Kinematički model definira odnos brzina iz $\{B\}$ koordinatnog sustava i pozicija i kutevima iz $\{E\}$ koordinatnom sustavu. Potpuni kinematički model se može pojednostaviti budući će se ronilica gibati samo po površini vode. Dakle potrebno je definirati rotacijsku matricu oko z osi. Kut zaošijanja određuje orijentaciju ronilice, te se smjer kazaljke na satu uzima kao pozitivan smjer rotacije. Pojednostavljeni kinematički model ronilice glasi :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{4.2.1}$$

4.3. Identifikacija modela

Najveći problem za realizaciju automatskog pozicioniranja ronilice predstavlja sporost osvježavanja mjerenja x, y pozicije. Uzme li se obzir da se aktuator za lateralno gibanje ronilice ne koristi, odnosno da je brzina zanošenja uvijek jednaka nuli, prema izrazu (4.2.1), promjena x i y pozicije ovisi samo o brzini napredovanja u i kutu zaošijanja ψ. Potrebno je, dakle, doći do koeficijenata iz izraza (4.1.1). U tu svrhu postavljana je kamera i snimljeno je gibanje ronilice pod utjecajem različitih iznosa sile napredovanja X. Odzivi su snimljeni za $X = 1, X = 0.7, X = 0.6, X = 0.5,\ X = 0.4$ budući da već postojeći upravljački program zahtjeva da se sile i momenti zadaju u rasponu od -1 do 1. Na ronilicu je stavljen crveni marker kako bi se moglo pratiti njeno gibanje. Također, napisana je *Matlab* m-skripta za praćenje crvenog objekta i za računanje brzine istog.

4.3.1. Praćenje crvenog objekta pomoću *Matlaba*

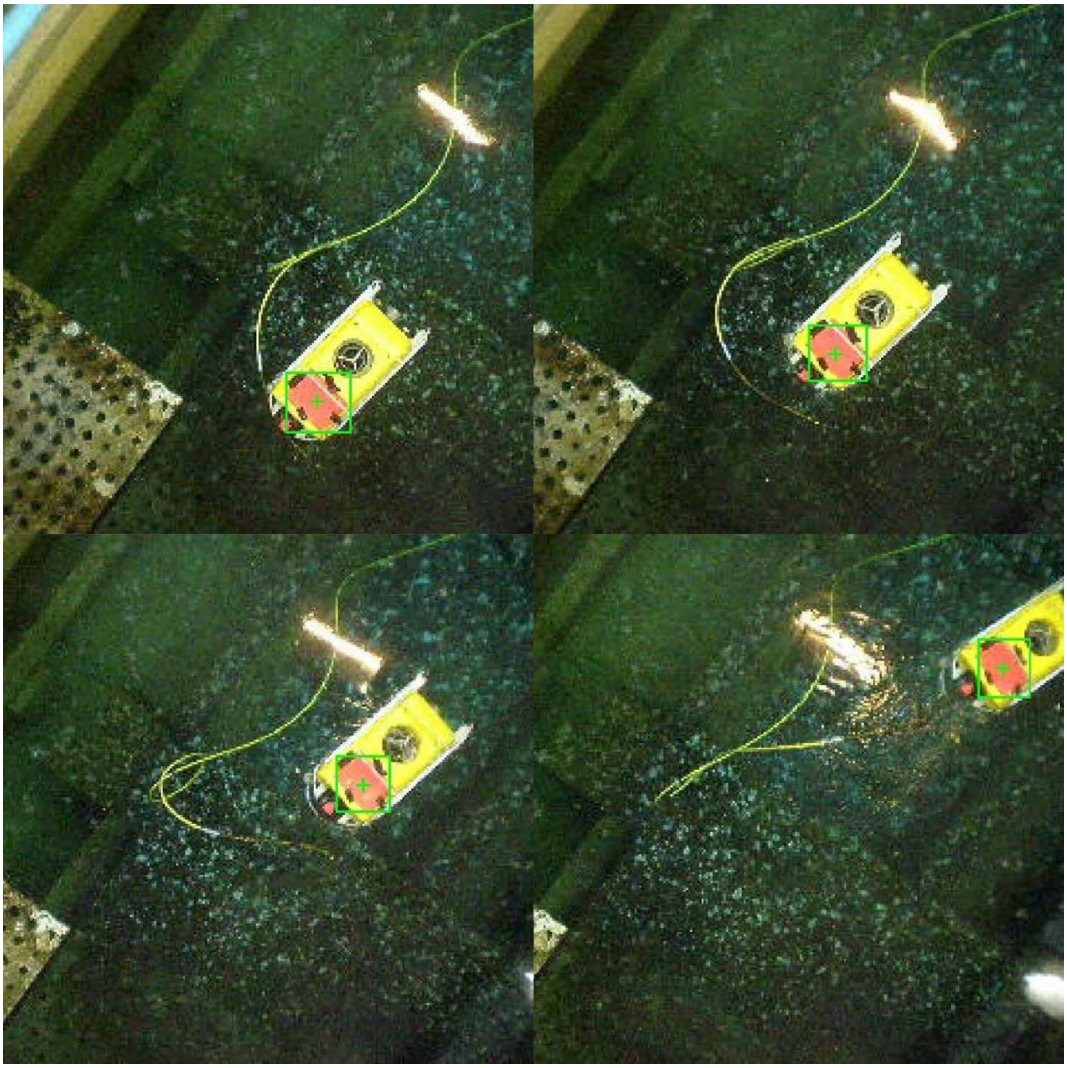
Prvo što je potrebno napraviti jest učitati podatke iz videa u *Matlab workspace*, a za to služi funkcija *VideoReader* koja za ulazni argument prima ime videa. Navedena funkcija vraća varijablu tipa *Multimedia Reader Object* koja u sebi sadrži informacije o trajanju videa, formatu videa, broju slika (engl.*frame*) u videu, frekvenciji slika (engl. *Frame Rate*) itd. Pojedina slika unutar videa se može promatrati kao matrica u kojoj su na mjestu svakog *pixela* zapisane vrijednosti crvene,zelene i plave boje. Kako bi se realiziralo praćenje crvenog markera, potrebno je iz slike izvući dijelove koji u sebi sadrže crvenu komponentu. Sljedeće linije koda pokazuju jedan od načina kako ostvariti navedeno:

```
%prebacivanje rgb slike u grayscale sliku
gray_im=rgb2gray(frame);
%izuzimanje crvenih komponenti iz grayscale slike
red_im = imsubtract(frame(:,:,1), gray_im);
%median filter za flitriranje šuma
red_im = medfilt2(red_im, [3 3]);
```

Dobivenu sliku '*red_im'* je potrebno zapisati u binarnom formatu, te ukloniti sve crvene komponente manje od 150 *pixela* koje mogu predstavljati smetnju pri praćenju markera. Također, potrebno je označiti sve komponente slike koje čine

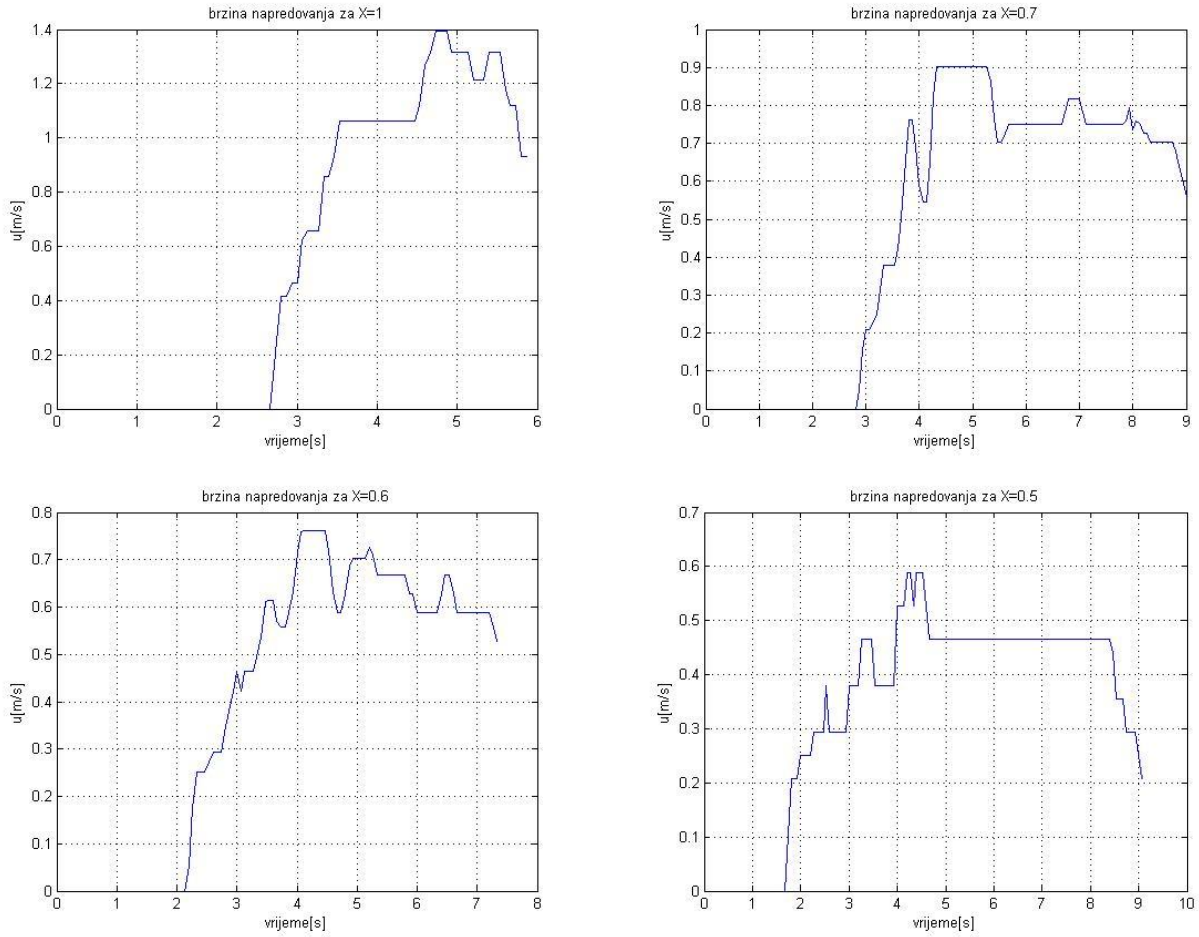
cjelinu i za svaku od njih izračunati centroid. Budući da se pri snimanju videa pazilo da je crveni marker jedini veći crveni objekt u videu, izračunati centroid odgovarat će centroidu markera. Opisani postupak realizira se sljedećim linijama koda, a rezultati su prikazani na slici 13 :

```
% grayscale slika -> binarna slika
red_im = im2bw(red_im,0.15);
% uklanjanje komponenti manjih od 150px
red_im = bwareaopen(red_im,150);
% označavanje povezanih cjelina u slici
L = bwlabel(red_im, 8);
% racunanje cetroida i okvira
tag = regionprops(L, 'BoundingBox', 'Centroid');
```



Slika 13. Praćenje crvenog markera,odnosno ronilice

Brzina ronilice računa se tako da se udaljenost centroida između dva *framea* pomnoži s *frame rateom*. Budući da se radi o malim brzinama, računanje brzine na ovaj način je prihvatljivo. Također, dobivenu brzinu u *pixelima* po sekundi potrebno je prebaciti u metre po sekundi, te upotrijebiti median filtar kako bi se smanjio utjecaj šuma.



Slika 14. Odzivi brzina za X=1, X=0.7,X=0.6,X=0.5

4.3.2. Identifikacija modela pomoću *Matlaba*

Matlab funkcija *procest* na temelju ulaznih parametara estimira model odabrane strukture. U ovom slučaju, na temelju diferencijalne jednadžbe (4.1.1), odabran je model opisan PT1 članom :

$$G(s) = \frac{K}{1+Ts} \tag{4.3.1}$$

Ulazni parametar *process* funkcije mora biti *iddata* objekt, pa je potrebno pozvati *iddata* funkciju čija će izlazna varijabla biti odgovarajućeg tipa. *Iddata* funkcija za ulazne argumente prima izlazni signal sustava, ulazni signal sustava, te vrijeme uzorkovanja. U ovom slučaju izlazni signal sustava odgovara brzini u, ulazni signal sustava odgovara sili X, a vrijeme uzorkovanje jednako je *frame rateu.* Kako su odzivi brzina snimani za pet različitih iznosa sile X, potrebno je stvoriti pet *iddata* objekata i povezati ih u jedan objekt funkcijom *merge*. Novonastali objekt predstavlja ulazni parametar *process* funkcije, te se njenim pozivanjem dobije matematički model koji najbolje opisuje sustav definiran jednadžbom (4.1.1).

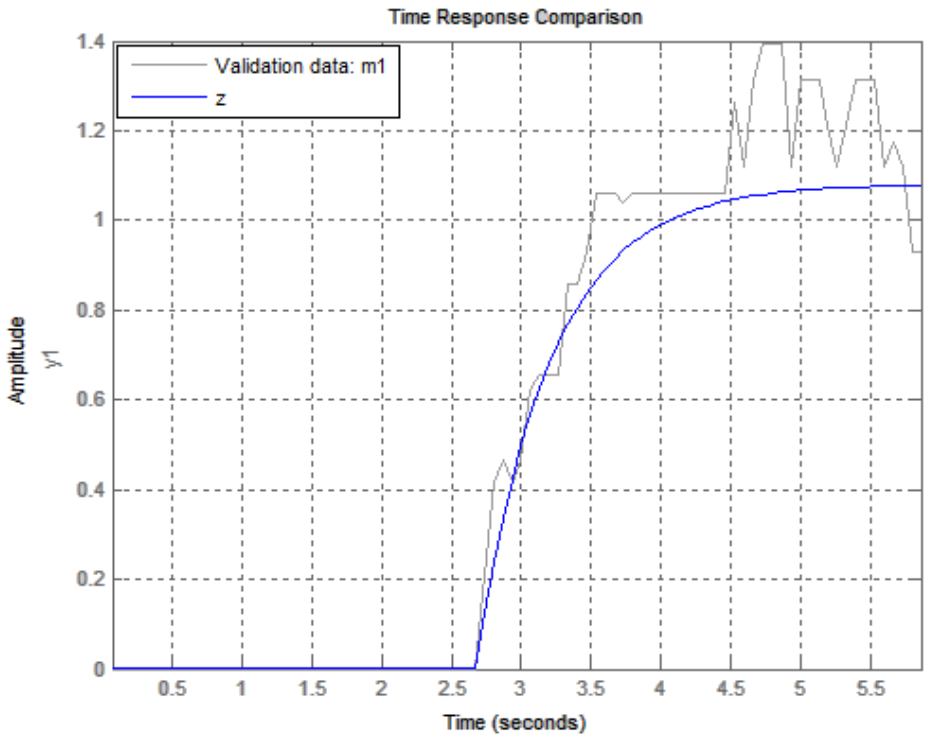
Sljedeće linije koda predstavljaju navedeni postupak estimacije modela:

```
%stvaranje iddata objekata
m1=iddata(vel_n1_filt,u1,1/15);
m4=iddata(vel_n4_filt,u4,1/15);
m5=iddata(vel_n5_filt,u5,1/15);
m6=iddata(vel_n6_filt,u6,1/15);
m7=iddata(vel_n7_filt,u7,1/15);
%povezivanje u 1 iddata objekt
m=merge(m1,m4,m5,m6,m7);
%estimacija PT1 modela
z=procest(m,'P1D')
```

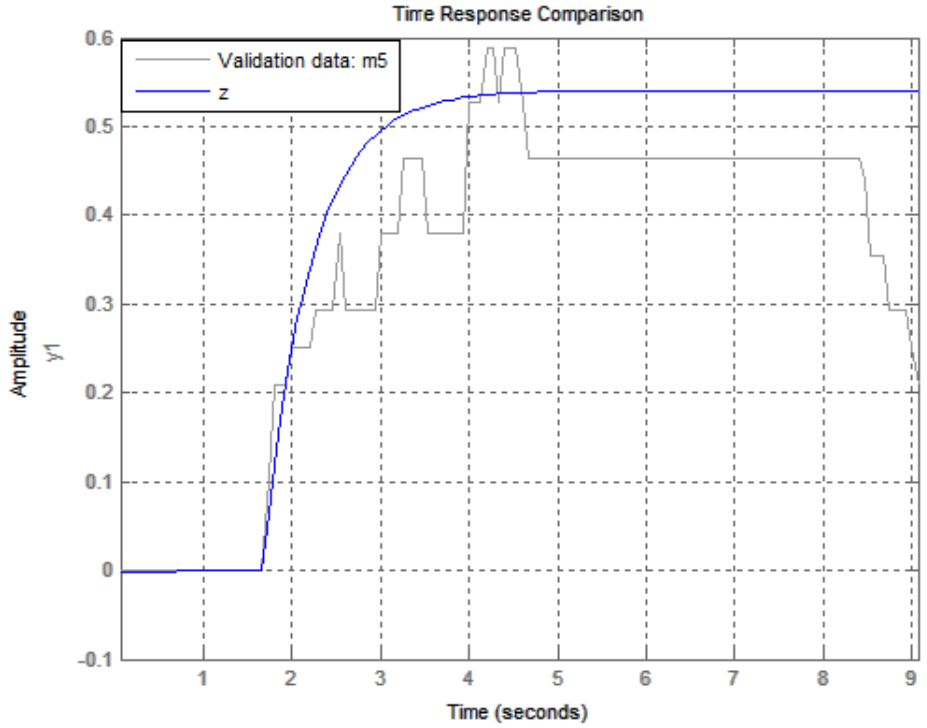
Traženi estimirani model je :

$$G(s) = \frac{1.081}{1+0.5372s} \tag{4.3.1}$$

Usporedi li se dobiveni model sa snimljenim odzivima brzina, najbolje poklapanje se dobije za $X = 1$ i iznosi 79.24%. Za $X = 0.6$, odnosno $X = 0.7$ poklapanje iznosi 78.04% , odnosno 77.58%. Najgori rezultati dobiju se za $X = 0.5$ i iznose 45.18%. Usporedbe odziva za najbolji i najgori slučaj, odnosno za $X = 1$ i $X = 0.5$ prikazane su na slikama ispod.



Slika 15. Usporedba odziva estimiranog modela i odziva brzine za X=1



Slika 16. Usporedba odziva estimiranog modela i odziva brzine za X=0.5

Dobiveni model koristit će se pri implementaciji Kalmanovog filtra. Budući je riječ o diskretnom Kalmanov filtru koji će se pozivati svakih 100ms, potrebno je dobiveni model diskretizirati. Traženi diskretizirani model s vremenom uzorkovanja 100ms ima oblik:

$$G(z) = \frac{0.1836z^{-1}}{1-0.8301z^{-1}} \tag{4.3.1}$$

5. Implementacija Kalmanovog filtra

Kao što je već prije navedeno, mjerenja pozicije dobivena iz *Desert Star PILOT* sustava osvježavaju se u prosjeku svakih pet sekundi. Također, dobivena mjerenja često znaju imati veliku pogrešku što onemogućuje precizno dinamičko pozicioniranje ronilice. Problem sporosti i nepreciznosti mjerenja može se riješiti projektiranjem Kalmanovog filtra (KF). Kalmanov filtar je rekurzivni stohastički estimator koji uz estimaciju varijabli stanja dodatno filtrira zašumljene mjerne signale [9]. Budući da se mjerenja iz senzora kursa osvježavaju prilikom svakog poziva upravljačkog programa (svakih 100ms) i budući je preciznost mjerenja zadovoljavajuća, kut zaošijanja nije potrebno estimirati, te se neće promatrati kao varijabla stanja. Također, već je ranije spomenuto da je brzina zanošenja uvijek jednaka nuli. Imajući navedeno na umu, jednadžbe sustava na temelju kojih će se estimirati stanja sustava napisane su ispod, pri čemu je $T = 100ms$.

$$u(k+1) = 0.8301u(k) + 0.1836X(k) \tag{5.1}$$

$$x(k+1) = x(k) + Tcos(\psi)u(k) \tag{5.2}$$

$$y(k+1) = y(k) + Tsin(\psi)u(k) \tag{5.3}$$

Budući se kut zaošijanja ne estimira, već se uzima direktno iz mjerenja, navedene jednadžbe su linearne pa je korištenje diskretnog Kalmanovog filtra najbolja opcija

u ovom slučaju. Estimacija stanja se provodi rekurzivno u dva koraka, korak predikcije i korak korekcija. Korak predikcije predviđa stanja sustava na temelju jednadžbe stanja sustava, dok korak korekcije korigira stanja dobivena korakom predikcije na temelju novih mjerenja. Zapišu li se jednadžbe promatranog sustava ronilice na sljedeći način [9] :

$$x_k = \Phi_{k-1} x_{k-1} + \Gamma_{k-1} u_{k-1} + w_{k-1} ; \tag{5.4}$$

$$y_k = H_k x_k + v_k ; \tag{5.5}$$

jednadžbe diskretnog Kalmanovog filtra su :

- <u>jednadžbe predikcije:</u>

$$\hat{x}_k^- = \Phi_{k-1} \hat{x}_{k-1}^+ + \Gamma_{k-1} u_{k-1} \tag{5.6}$$

$$P_k^- = \Phi_{k-1} P_{k-1}^+ \Phi_{k-1}^T + Q_{k-1} \tag{5.7}$$

- <u>jednadžbe korekcija:</u>

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{5.8}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k (y_k - H_k \hat{x}_k^-) \tag{5.9}$$

$$P_k^+ = (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T \tag{5.10}$$

w_k i v_k označavaju procesni, odnosno mjerni bijeli šum, matrica Q_k predstavlja nesigurnost modela, a matrica R_k nesigurnost mjerenja. Matrica P_k^- predstavlja nesigurnost stanja, dok K_k označava Kalmanovo pojačanje.

U promatranom slučaju matrica x_k predstavlja vektor stanja $[u(k)\ x(k)\ y(k)]^T$,a matrica u_k predstavlja iznos sile napredovanja X. Vrijednosti ostalih matrica su :

$$\Phi_{\mathrm{k}} = \begin{bmatrix} 0.8301 & 0 & 0 \\ Tcos(\psi_k) & 1 & 0 \\ Tsin(\psi_k) & 0 & 1 \end{bmatrix} \tag{5.11}$$

$$\Gamma_{\mathrm{k}} = \begin{bmatrix} 0.1836 \\ 0 \\ 0 \end{bmatrix} \tag{5.12}$$

$$Q_k = \begin{bmatrix} 0.04 & 0 & 0 \\ 0 & 0.04 & 0 \\ 0 & 0 & 0.04 \end{bmatrix} \tag{5.13}$$

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.14}$$

Matrica H_k realizirana je za dva slučaja, kada postoje valjana mjerenja, te kada mjerenja nisu dostupna ili nisu valjana. Kao što se može vidjeti iz jednadžbi (5.6) - (5.10) , matrica H_k nalazi se samo u jednadžbama korekcije pa u slučaju kada nema valjanih mjerenja, estimacija stanja se vrši samo prema dinamičkom modelu.

$$H_1(k) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.15}$$

$$H_2(k) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.16}$$

Također, kako bi Kalmanov filtar ispravno radio, potrebno je zadati početne uvjete, pa u ovom slučaju vrijedi :

$$\hat{x}_0^+ = \begin{bmatrix} 0 & x_{poc} & y_{poc} \end{bmatrix}^T \tag{5.17}$$

x_{poc} i y_{poc} predstavljaju poziciju ronilice u trenutku pokretanja Kalmanovog filtra. P_0^+ predstavlja nesigurnost početnog stanja:

$$P_0^+ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.18}$$

6. Upravljački algoritam

Već postojeći upravljački program napisan u *LabVIEWu* nadopunjen je tako da podržava i automatsko pozicioniranje ronilice. U postojeći program dodana su tri *SubVI* programa i *Control Button* za odabir načina rada ronilice. Postoji ručni način rada, odnosno upravljanje ronilicom pomoću *joysticka,* i automatski način rada. Prvi *SubVI* potprogram služi za čitanje podataka iz *DiveBase* programa i opisan je u poglavlju 2.3. Izlazi iz ovog *SubVI*, koji ujedno predstavljaju i ulaze u sljedeći *SubVI* u kojem je implementiran Kalmanov filtar, su x i y pozicija dobivena iz *PILOT* sustava. Kalmanov filtar implementiran je u *MathScript* čvoru prema jednadžbama opisanim u poglavlju 5. Treći dodani *SubVI* predstavlja postupak kojim se na temelju estimirane pozicije dobivene Kalmanovim filtrom računa sila napredovanja X, odnosno moment zaošijanja N kako bi se ronilica uspješno pozicionirala u zadanoj točki. Poznajući koordinate ciljne točke i poznajući poziciju ronilice dobivenu iz Kalmanovog filtra može se izračunati traženi kurs (engl.*heading*) ronilice i udaljenost (engl. *distance*) od ciljne točke :

$$heading_{cilja} = \tan^{-1}\left(\frac{y_{cilja} - y_{KF}}{x_{cilja} - x_{KF}}\right) \tag{6.1}$$

$$distance = \sqrt{\left(x_{cilja} - x_{KF}\right)^2 + \left(y_{cilja} - y_{KF}\right)^2} \tag{6.2}$$

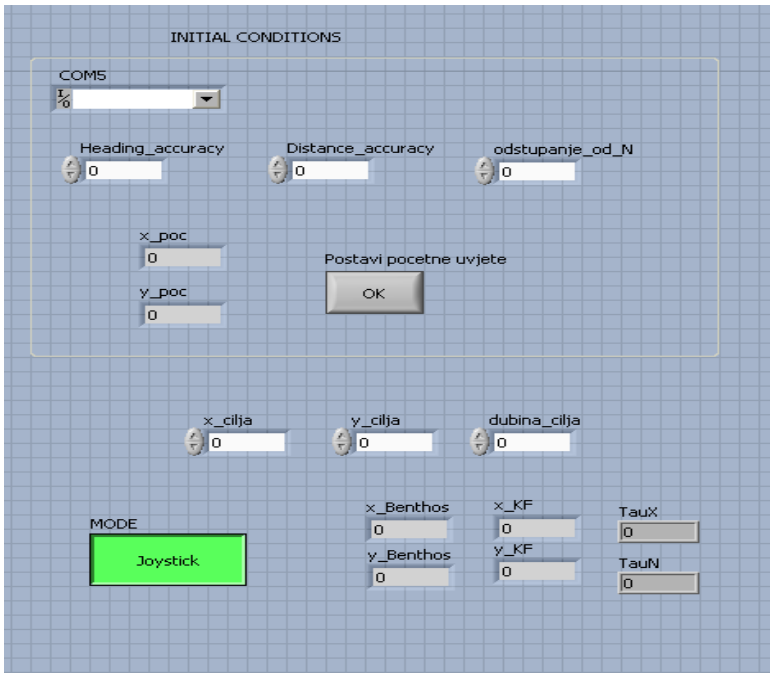
Algoritam automatskog upravljanja ronilicom je napravljen tako da se, sve dok je razlika kursa ronilice i traženog kursa veća od $30°$, zadaje samo moment zaošijanja N. Ukoliko je navedena razlika manja od $30°$, zadaju se i sila napredovanja X i moment zaošijanja N. Način na koji se zadaju sila napredovanja X i moment zaošijanja N je sljedeći :

- ako je udaljenost od cilja veća od tri metra, $X = 3/distance$,
- ako je udaljenost od cilja manja od tri metra, $X = distance/3$
- ako je razlika kurseva veća od 30°, $N = +/-0.6$
- ako je razlika kurseva manja od 30°, $N = razlika/30$

Ovime se osigurava da iznosi sila i momenata nikad nisu veći od 1, kao što to zahtjeva postojeći upravljački program. Također, prilikom zadavanja momenta zaošijanja vodi se računa o smjeru zakretanja, odnosno odabire se smjer za koji je

put zakretanja kraći. Radi prevencije nastanka štene prilikom testiranja algoritma automatskog upravljanja, postavljen je gornji limit sile napredovanja na 0.7. Donji limit postavljen je na 0.3 jer se pokazalo da se za manje iznose sile napredovanja ronilica gotovo pa ne kreće. Iz istih razloga moment zaošijanja je limitiran na iznose od 0.25 do 0.6, odnosno od -0.6 do -0.25. Također, dodane su i dvije varijable, *Heading_accuracy* i *Distance_accuracy* čije vrijednosti određuje sam operater. Navedene varijable određuju preciznost automatskog pozicioniranja ronilice. Ukoliko je udaljenost od cilja manja od *Distance_accuracy*, sila napredovanja se postavlja na nulu. Isto tako, ukoliko je razlika kurseva manja od *Heading_accuracy*, moment zaošijanja jednak je nuli. Prilikom pokretanja upravljačkog algoritma za automatsko pozicioniranje, operater treba učiniti sljedeće:

- odabrati COM port s kojeg se čitaju podaci iz *DiveBasea*
- unijeti vrijednosti za *Heading_accuracy* i *Distance_accuracy*
- ukoliko se y os referentnog koordinatnog sustava ne poklapa sa sjeverom, unijeti kut odstupanja od sjevera
- postaviti početne koordinate ronilice, te unijeti koordinate cilja
- prebaciti u automatski način rada

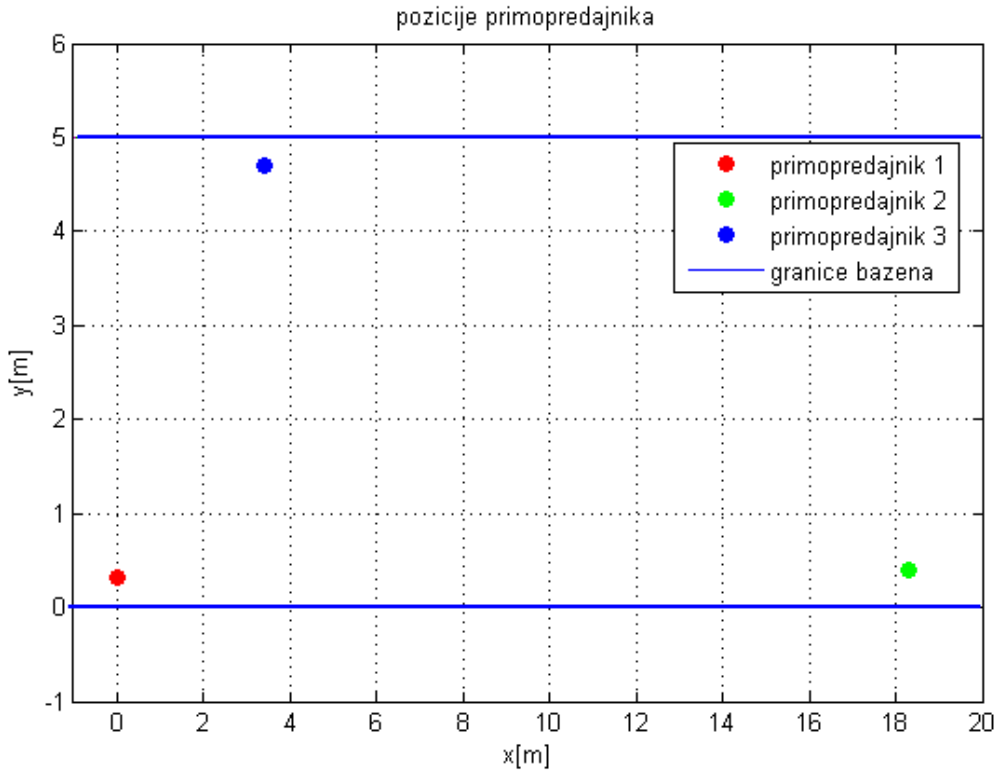


Slika 17. Dodaci konzoli za automatsko upravljanje

7. Rezultati testiranja

Upravljački algoritam se testirao u bazenu na Brodarskom institutu u Zagrebu. Prednost testiranja u bazenu u odnosu na prirodne površine vode je izostanak valova, koji mogu znatno otežati automatsko pozicioniranje. Međutim, bazen ima i svoje nedostatke. Glavni nedostatak jest refleksija zvuka o zidove bazena, koja negativno utječe na preciznost sustava za pozicioniranje. Također, prilikom testiranja nije postojao referentni sustav za praćenje pozicije ronilice pa su podaci o stvarnoj putanji ronilice izostavljeni. Fiksni primopredajnici *PILOT* sustava postavljeni su na sljedeće pozicije (koordinate točaka navedene su u metrima):

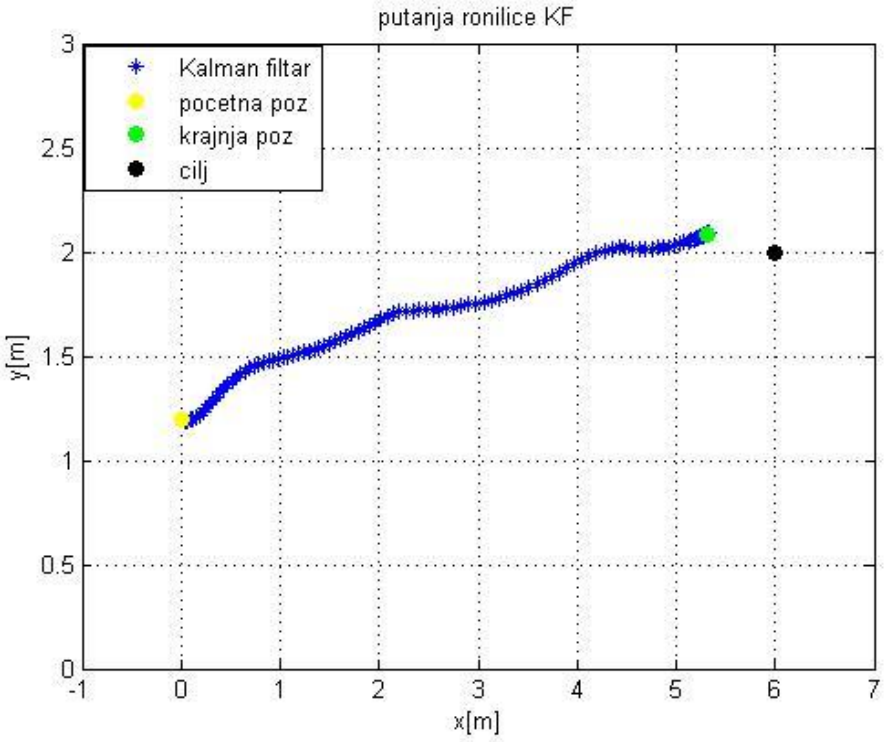
- primopredajnik 1 : (0, 0.32)
- primopredajnik 2 : (18.3, 0.4)
- primopredajnik 3 : (3.4, 4.7)



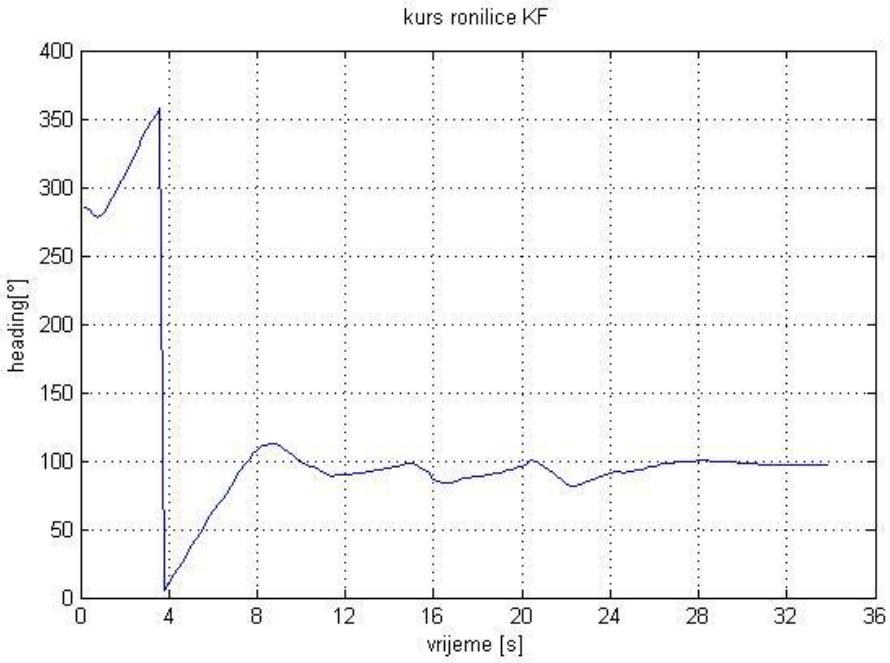
Slika 18.Pozicije primopredajnika

Mobilni primopredajnik postavljen je na ronilicu kao što je prikazano na slici 2.

Prvi eksperiment proveden je bez sustava za pozicioniranje, odnosno ronilica se pozicionirala isključivo pomoću Kalmanovog filtra. Slike ispod pokazuju putanju ronilice i promjenu kursa tijekom eksperimenta. *Heading_accuracy* je postavljen na 8° , a *Distance_accuracy* na jedan metar.



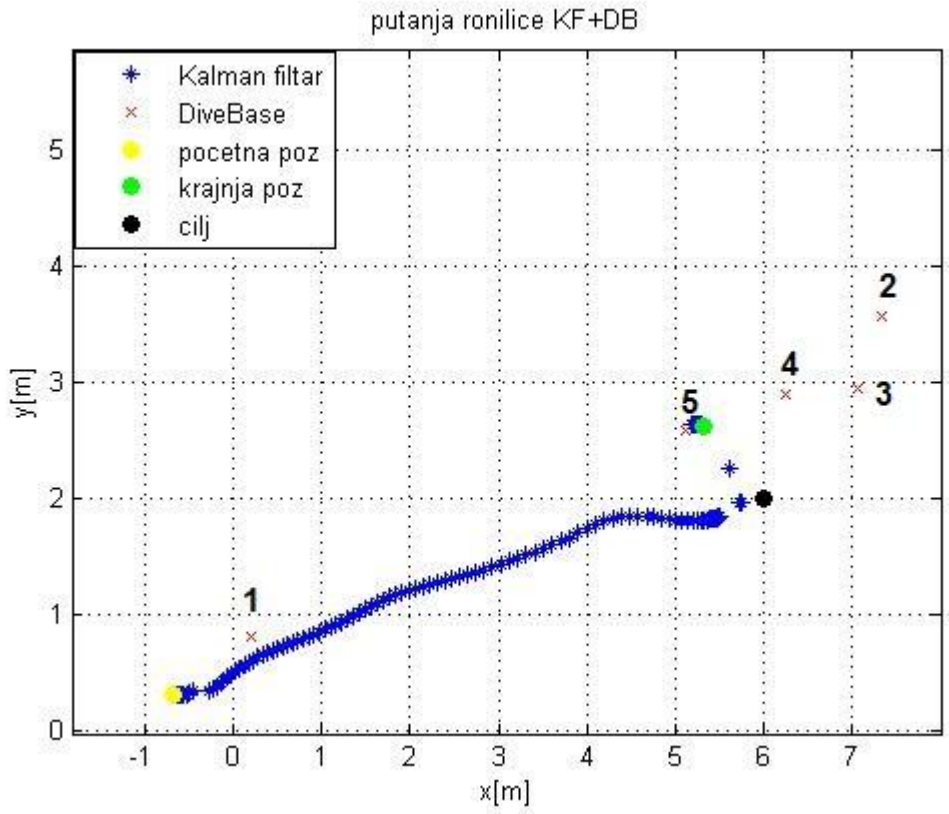
Slika 19. Putanja ronilice KF



Slika 20. Promjena kursaa KF

Kao što se može vidjeti sa slike 20, ronilica se rotirala sve dok nije postigla željeni kurs, te je kasnije taj kurs nastojala održati. Također, sa slike 19 može se primijetiti da ronilica teži pravocrtnom gibanju što je više moguće.

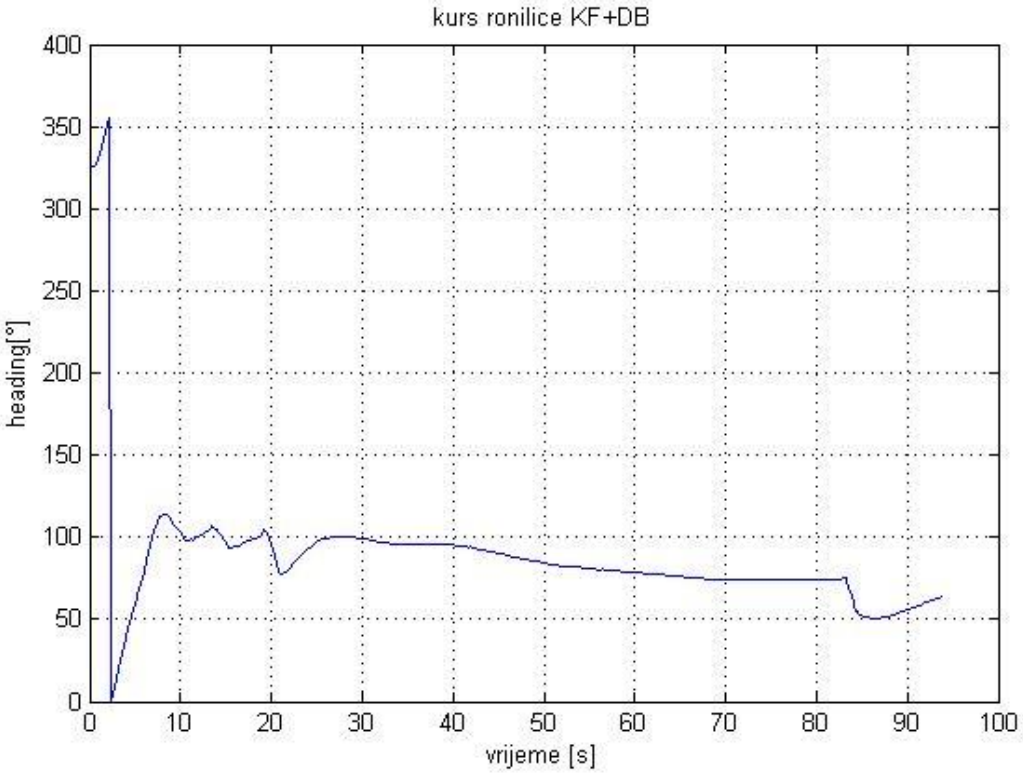
Sljedeći eksperiment proveden je koristeći sustav za pozicioniranje i Kalmanov filtar, a rezultati su prikazani na slici ispod. *Heading_accuracy* i *Distance_accuracy* ostali su isti kao i u prethodnom eksperimentu.



Slika 21. Putanja ronilice, KF i DB

Kao što se može vidjeti sa slike 21, ronilica se uspješno pozicionirala unutar kružnice radijusa jednog metra od ciljne točke. Brojevi iznad pozicija dobivenih iz *DiveBase* sustava označavaju redni broj mjerenja. Sa slike se može primijetiti da je između prvog i drugog mjerenja prošlo dosta vremena. Uslijed većih brzina, *PILOT* sustav nije u mogućnosti dati zadovoljavajuću poziciju, odnosno pogreška mjerenja je prevelika pa se ta mjerenja odbacuju. Upravo je to razlog velike vremenske razlike između prvog i drugog mjerenja. Što je ronilica bliže cilju, to je njena brzina napredovanja manja, pa tako *PILOT* sustav uspijeva dati približnu

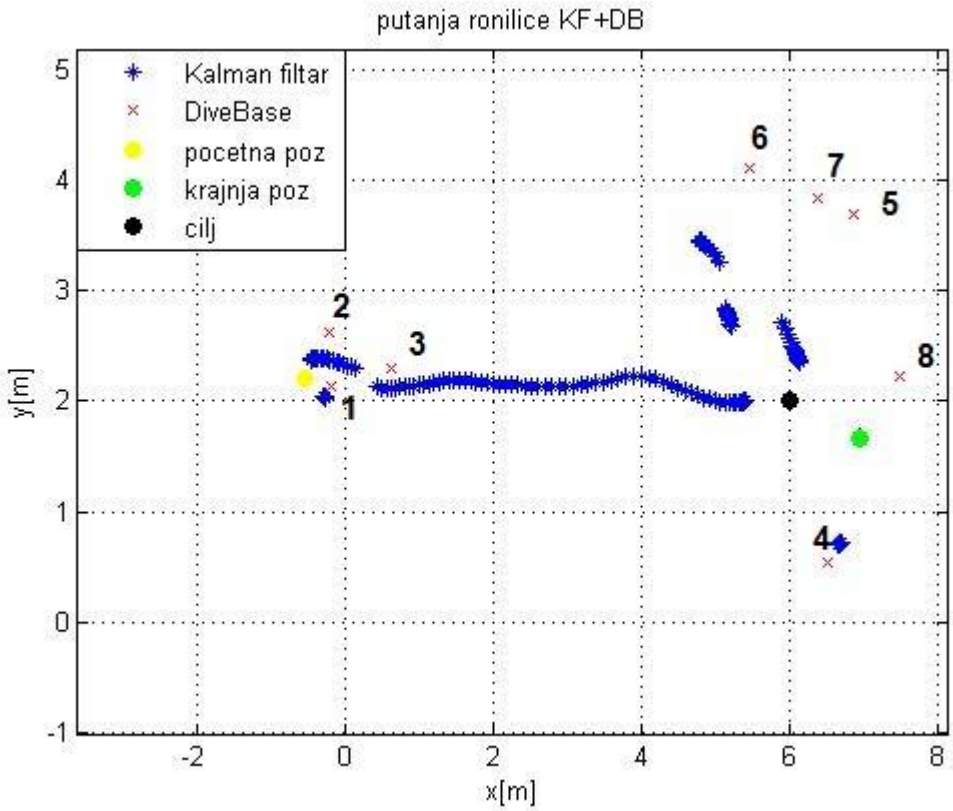
poziciju, ali i dalje nedovoljno preciznu. Tek kad je ronilica stala, *PILOT* sustav je u mogućnosti dati preciznu poziciju (mjerenje 5). Kalmanov filtar se u ovom slučaju pokazao uspješnim jer se ronilica, unatoč sporosti i nepreciznosti mjerenja, uspjela pozicionirati.



Slika 22.Kurs ronilice, KF i DB

Sa slike 22 je vidljivo da ronilica, kao i u prvom eksperimentu, prvo rotira dok ne postigne zadovoljavajući kurs, a potom ga nastoji zadržati.

U sljedećem eksperimentu, također se koristi sustav za pozicioniranje i Kalmanov filtar. *Heading_accuracy* ostaje nepromjenjen, a *Distance_accuracy* iznosi 1.5 metara. Rezultati ovog eksperimenta prikazani su na slici ispod. Ponovno se može primijetiti da uslijed većih brzina dolazi do gubitka mjerenja iz sustava za pozicioniranje (velika vremenska razlika između mjerenja 3 i mjerenja 4). Također, može se vidjeti da uslijed dolaska novih mjerenja Kalmanov filtar brzo korigira stanje. Ronilica se i ovom slučaju uspješno pozicionirala na udaljenosti manjoj od *Distance_accuracy.*



Slika 23. Putanja ronilice, KF i DB, eksperiment 2

8. Zaključak

U sklopu diplomskog rada ispitana je funkcionalnost postojećeg upravljačkog programa u *LabVIEWu*, kao i funkcionalnost *Desert Star PILOT* sustava za pozicioniranje. Komunikacija između računala i ronilice je uspješno uspostavljena, te je uspješno pokrenut upravljački program, koji omogućuje upravljanje ronilicom putem *joysticka. Desert Star PILOT sustav* nije u potpunosti osposobljen budući da dubinomjer na mobilnom primopredajniku ne radi ispravno. Zbog toga je automatsko pozicioniranje realizirano samo u x-y ravnini. Također, zbog sporosti osvježavanja mjerenja iz *PILOT* sustava bilo je potrebno doći do matematičkom modela ronilice, te implementirat Kalmanov filtar. Do matematičkom modela se došlo snimanjem gibanja ronilice za različite iznose sile napredovanja. Dobiveni video podaci su se obradili u *Matlabu,* te je identificiran model ronilice. Na temelju dobivenog modela projektiran je Kalmanov filtar, te je napisan upravljački algoritam za automatsko pozicioniranje ronilice. U svrhu provjere funkcionalnosti dobivenih rješenja, provedeno je nekoliko eksperimenata. Rezultati eksperimenata su zadovoljavajući, te se može zaključiti da su temelji za precizno automatsko pozicioniranje zasigurno postavljeni. Također, može se zaključiti da *PILOT* sustav ne radi baš najbolje u bazenu, a pogotovo ne pri većim brzinama ronilice. Kalmanov filtar pokazao se kao vrlo dobro rješenje budući da se i uslijed nedostatka mjerenja ronilica uspješno pozicionirala. U budućem radu na ovu temu, bilo bi poželjno kada bi se instalirao referentni sustav za praćenje putanje ronilice, kako bi se rezultati upravljačkog algoritma mogli evaluirati. Također, trebalo bi ispitati mogućnost korigiranja mjerenja dobivenih iz sustava za pozicioniranje na temelju podataka o dubini dobivenih iz dubinomjera ronilice. U slučaju da takva mogućnost postoji, automatsko pozicioniranje ronilice u prostoru bi bilo izvedivo.

Vlastoručni potpis:_____

9. Literatura

[1] Desert Star Systems LLC, PILOT System, http://desertstar.com/product/pilot-system/ , 15.04.2016.

[2] Desert Star Systems LLC, PILOT System Manual , http://www.desertstar.com/knowledgebase/doku.php?id=manuals:pilot_system_manual, *PILOT Precision Underwater Acoustic Positioning System Operator's Manual,* 15.04.2016.

[3] Stipanov, M. *Sustavi i metode podvodne lokalizacije.* STO-CMRE. La Spezia,Italija

[4] Gustafsson, F., Gunnarsson, F., "Positioning using timedifference of arrival measurements," Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on , vol.6, no., pp.VI,553-6 vol.6, 6-10 April 2003

[5] Tiemann CO, Porter MB, Frazer LN. Localization of marine mammals near Hawaii using an acoustic propagation model. J Acoust Soc Am. 2004 Jun;115(6):2834-43.

[6] ROVeXchange, *ROV Review - Benthos Stingray* , http://www.rovexchange.com/rov_review_benthos_stingray.php , 11.05.2016.

[7] Fossen, T., Guidance and Control of Ocean Vehicles, John Wiley and Sons, Engleska, 1994

[8] Stilinović,N. *Estimator stanja za autonomnu plovnu platformu*. Diplomski rad. Fakultet elektrotehnike i računarstva, 2012

[9] Perić,N. ,Petrović,I. *Poglavlje 9: Diskretni linearni Kalmanov filtar.* Teorija estimacije. Fakultet elektrotehnike i računarstva , 2014

[10] Bakarić,V., Eškinja,Z.,Vukov,A. *Podvodno vozilo na daljinsko upravljanje Benthos Stingray MK II – Korisničke upute.* Brodarski institut, Zagreb

INTEGRACIJA BENTHOS RONILICE I SUSTAVA ZA POZICIONIRANJE

Sažetak

Benthos ronilica je daljinski upravljana ronilica za koju postoji upravljački program realiziran u *LabVIEW* programskom paketu. *Desert Star* akustički sustav za pozicioniranje omogućuje određivanje lokacije objekata pod vodom. U sklopu diplomskog zadatka ispitana je funkcionalnost postojećeg programa za upravljanje Benthos ronilicom, te je uz pomoć sustava za pozicioniranje realizirano automatsko upravljanje ronilicom. Također, u radu je opisan postupak identifikacije modela iz videa, te je realiziran Kalmanov filter za estimaciju pozicije.

Ključne riječi: sustav za pozicioniranje, ronilica, Kalmanov filtar, identifikacija, estimacija

Integration of Benthos underwater vehicle and positioning system

Summary

Benthos underwater vehicle is remotely operated vehicle and its driver is already realized in *LabVIEW*. Desert Star is acoustic positioning system that is used for determining the location of the underwater object. As part of the assignment the functionality of already existing driver is tested and the autonomy of the vehicle is realized by using positioning system. System identification from video and Kalman filter for position estimation are also described in this paper.

Key words: positioning system, underwater vehicle, Kalman filter, identification, estimation