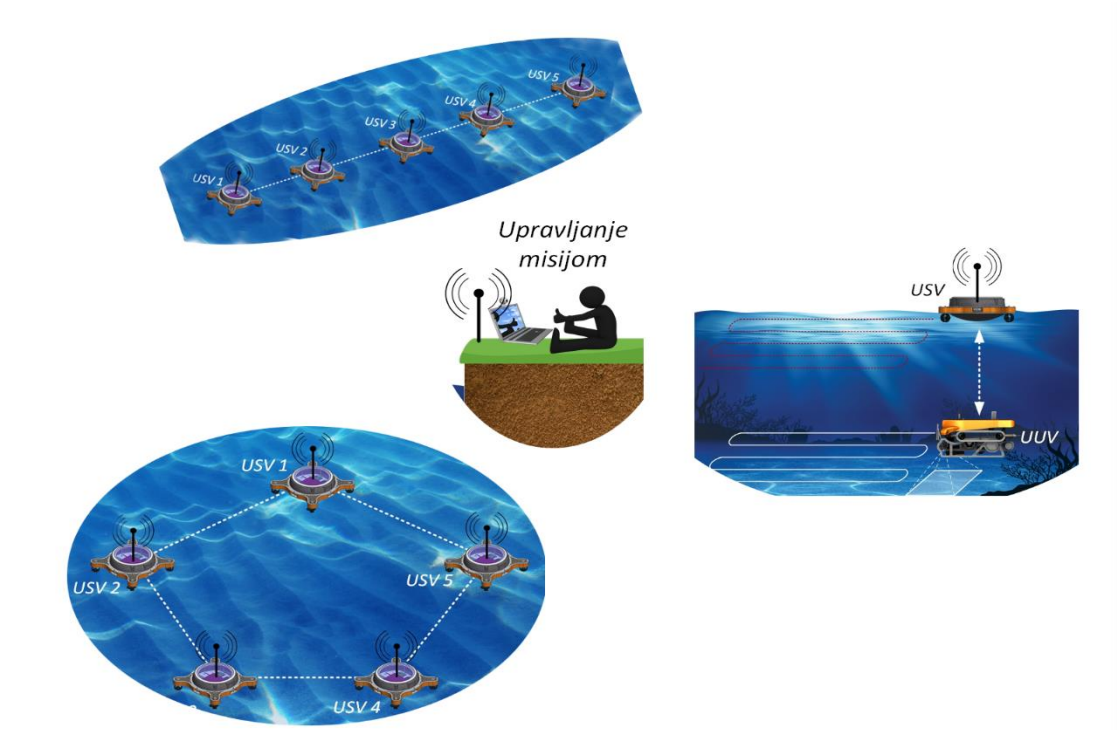
CroMarX

Kooperativna robotika u nadzoru i istraživanju mora

D5.2 Softver za upravljanje misijom i korisničko sučelje



Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva

Laboratorij za podvodne sustave i tehnologije

31. kolovoz 2018.

Sadržaj

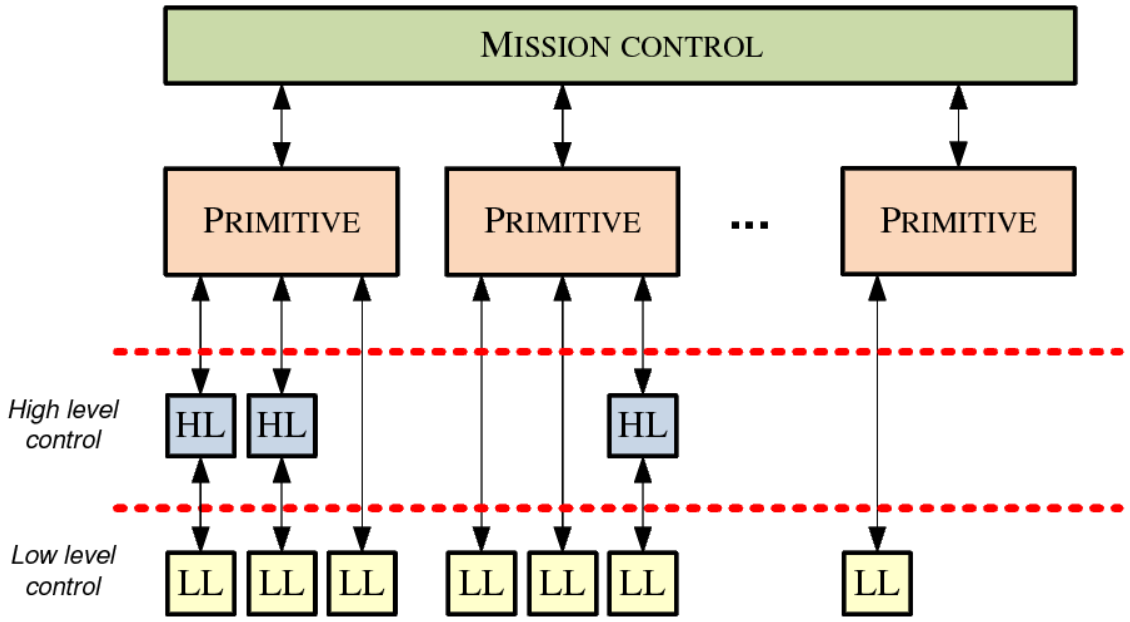
Uvod

Upravljanje i nadzor više pomorskih vozila predstavlja izazov zahvaljujući njihovoj prostornoj raspodjeli i nevidljivosti u podvodnom okolišu. Tijekom izvođenja eksperimenata u morskom okruženju, od velike je važnosti imati programsku potporu za upravljanje misijom sa sposobnošću i) pružanja informacija o i prikaz pozicija svih vozila uključenih u eksperiment, s naglaskom na podvodna vozila koja nisu vidljiva s kopna; ii) funkcioniranja kao korisniku prilagođen centar za upravljanje i davanje naredbi pojedinim vozilima kao i flotama vozila, te upravljanje tijekom misije.

U izvješću D5.1 „Zahtjevi i arhitektura softvera za upravljanje misijom“ prikazani su zahtjevi na softver za upravljanje misijom kojim se ostvaruju tražene sposobnosti. U ovome izvješću dan je prikaz ostvarenoga napretka u razvoju novoga softvera i integracije postojećega u svrhu nadzora i upravljanja vozilom u misiji, kao i upravljanja misijama za pojedina i skupine vozila.

Softver za upravljanje misijom

Kako bi se mogle izvoditi kompleksne misije na pojedinom vozilu potrebno je integrirati sustav koji prima naredbe od ljudskog operatora i za odgovarajuću naredbu generira referentne veličine za pogonski sustav vozila kojima se ona ostvaruje. Softver za upravljanje misijom mora biti generički kako bi njegova primjena bila moguća na različitim hardverskim platformama. Kako bi se napravila željena apstrakcija hardvera korištena je hijerarhijska upravljačka struktura implementirana u Robot Operating System (ROS) programskom okruženju, prikazana Slikom 1, koja se sastoji od regulatora niske razine (engl. Low level control), regulatora visoke razine (engl. High level control), primitiva i dijela za upravljanje misijom (engl. Mission control). ROS je fleksibilni programski okvir za pisanje softvera za robotske sustave. Sadrži kolekciju alata, biblioteka i konvencija čiji je cilj pojednostavljenje kreiranja kompleksnih i robusnih zadaća koje različite robotske platforme trebaju izvršavati. Iako u svom nazivu sadrži riječi operativni sustav, ROS nije operativni sustav u punom smislu te riječi. ROS pruža neke usluge operativnog sustava kao što su apstrakcija hardvera, kontrola uređaja niske razine, upravljanje paketima, razmjena poruka između procesa, implementacija često korištenih poruka. Također pruža biblioteke i alate za pisanje, pokretanje i izgradnju distribuiranoga koda. Sa svojim osobinama ROS spada u meta-operativni sustav, pa ga možemo shvatiti kao nadogradnju na postojeći operativni sustav. Sav softver razvijen unutar ROS zajednice je otvorenoga koda i razvoju pojedinih komponenti doprinosi veliki broj ljudi diljem svijeta čime je stvorena univerzalna baza softvera koja uvelike olakšava razvoj robota.



Slika 1 Hijerarhijska upravljačka struktura

Regulatori niže i više razine u softverskoj arhitekturi vozila su implementirani kao standardni PID regulatori, i specifični su za pojedinu hardversku platformu, međutim ono što je od glavnoga interesa u ovom izvješću je razina upravljanja misijom. U softverskoj arhitekturi vozila razvijenog unutra Laboratorija za podvodne sustave i tehnologije (LABUST), misija je definirana kao set primitiva odnosno akcija koje je potrebno izvršiti. Primitivi su definirani kombinacijom regulatora visoke i niske razine koje pozivaju te njegovim ulazima dok regulatori visoke razine kao izlaz imaju linearnu kombinaciju brzina. Primjeri regulatora visoke razine i primitiva dani su u Tablici 1 i Tablici 2. Implementacija pojedinih regulatora niske razine, regulatora visoke razine te primitiva je specifična za svako pojedinačno vozilo, dok je dio sustava za upravljanje misijom generičkog karaktera i jednak bez obzira na vozilo na kojem se nalazi.

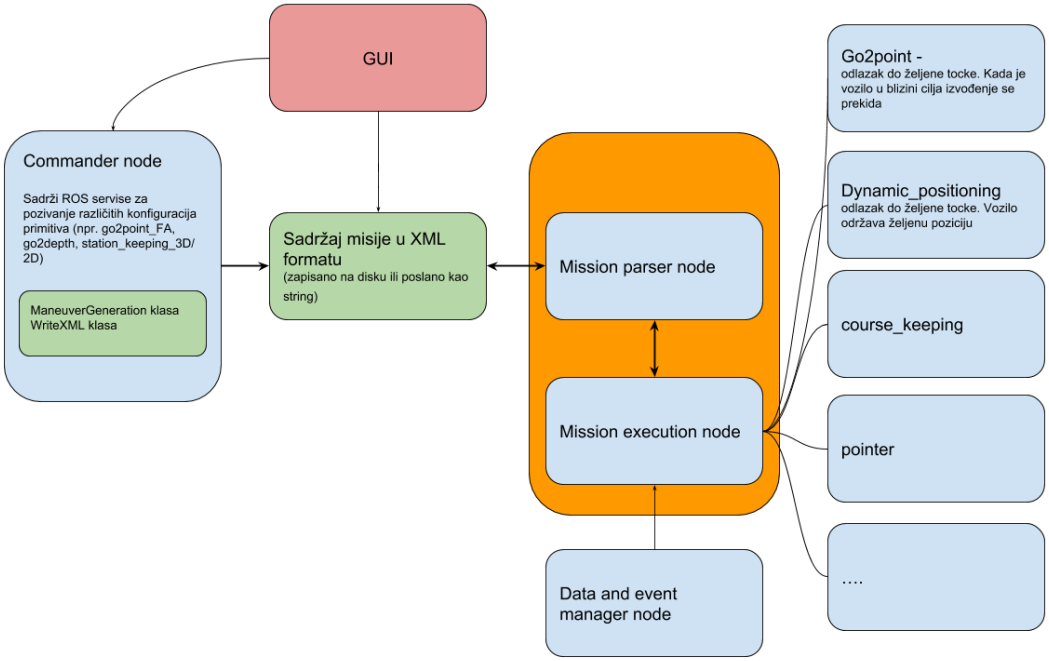
NAZIV	SKRAĆENI NAZIV	IZLAZI
Slijeđenje pravca	LF_FA	u*, v*
Podaktuirano slijeđenje pravca	LF_UA	u*, r*
Regulator zaošijanja	HDG	r*
Dinamičko pozicioniranje	FADP	u*, v*
Regulator dubine	DPT	w*
Regulator visine od morskoga dna	ALT	w*

Tablica 1 Primjer regulatora visoke razine, pri čemu je u brzina napredovanja, v* brzina zanošenja, w* brzina poniranja te r* brzina zaošijanja.*

NAZIV	ULAZI	REGULATORI
go2point_UA	T1, T2	LF_UA; HDG
go2point_FA	T1, T2	LF_FA; HDG
go2point_FA_hdg	T1, T2, Psi*	LF_FA; HDG
dynamic_positioning	T1, Psi*	DP; HDG

Tablica 2 Primjer dostupnih primitiva, pri čemu T1, T2 označavaju početne i završne točke, a Psi referentno zaošijanje.*

Slika 2 prikazuje arhitekturu implementiranoga softvera za upravljanje misijom. Misije se zadaju u XML formatu, koji sadrži informacije o parametrima misije, događajima koji mogu utjecati na izvođenje misije te primitivima koje je potrebno izvršiti. Primjer XML zapisa misije prikazan je u Tablici 3. Misija u XML formatu može piti poslana iz vanjskoga GUI-a, ili od „Commander" upravljačkoga čvora koji može generirati kompleksne misije online na temelju postavljenih zahtjeva i dobivenih mjerenja. Čvorovi „Mission parser" i „Mission execution" zaduženi su za čitanje XML misije, te pozivanje odgovarajućih primitiva. „Mission execution" čvor upravlja tokom misije uzimajući u obzir vanjske događaje. Prilikom izvođenja misije, moguće je da pokrenuti primitivi zahtijevaju iste regulatore niske ili visoke razine, stoga segment upravljanja misijom modelira sve potencijalne konflikte koristeći Petrijeve mreže kako bi se oni izbjegli. Kod upravljanja misijom omogućeno je prebacivanja u ručni režim rada, ako je dostupan, kako bi operater mogao reagirati u nepredvidivim situacijama.



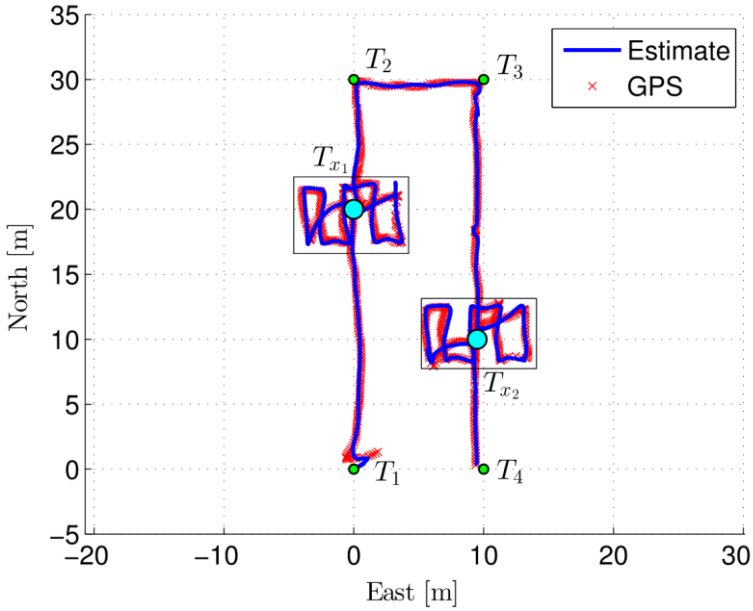
Slika 2 Arhitektura softvera za upravljanje misijom

```
<main>
    <params>
        <param name="paramName" id="paramID">paramValue</param>
        ...
    </params>
    <events>
        <event id="eventID">eventCondition</event>
        ...
    </events>
    <mission>
        <primitive name="primitiveName">
            <id>primitiveID</id>
            <param name="primitiveParamName">paramValue</param>
            ...
        </primitive>
        <primitive name="primitiveName">
            <id>primitiveID_2</id>
            <param name="primitiveParamName">paramValue</param>
            ...
        </primitive>

    </mission>
</main>
```

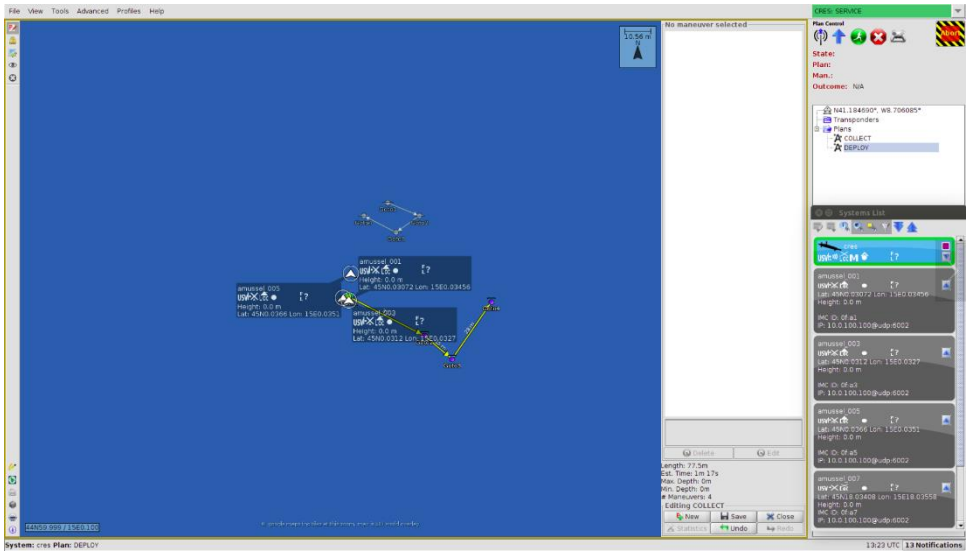
Tablica 3 Primjer XML zapisa misije.

Slika 3 Prikazuje eksperimentalne rezultate gdje sustav za upravljanje misije izvršava "lawnmower" uzorak, te se na vanjski događaj pokreće manji "lawnmower" uzorak, nakon čijeg kraj sustav nastavlja sa većim uzorkom. Upravljanje misijom je navedeno ostvarilo višestrukim pozivanjem "go2point_FA" primitiva. Unutar jedne misije primitiv može biti pozvan više puta s različitim parametrima. Primjerice, "lawnmower" uzorak je moguće ostvariti opetovanim pozivanjem go2point_FA primitiva s različitim ulaznim parametrima.



Slika 3 Primjer kompleksne misije sastavljene od „go2point_FA" primitiva

Za potrebe prikaza vozila i zadavanja misije od strane operatera koristi se Neptus upravljački softver, razvijen od strane LSTS-a, Portugal., prikazan na Slici 4. Razlog korištenja navedene komponente leži u tom što omogućava istovremeni prikaz heterogenih timova autonomnih vozila, dakle ne samo pomorskih vozila, već i zračnih te omogućava kompatibilnost između različiti robotskih platformi, što je bitan preduvjet za suradnju i izvođenje eksperimenta s partnerima na različitim projektima koji koriste potpuno različite softverske arhitekture. Vozila razvijena u LABUST-u koriste Neptus isključivo za prikaz i zadavanje osnovnih manevara. Za sve ostale upravljačke funkcije zadužen je LABUST-ov upravljački softver razvijen u ROS-u.



Slika 4

Kao što je već navedeno, softver za navigaciju, upravljanje i planiranje misije na vozilu implementiran je u ROS (eng. Robot Operating System) distribuiranom okruženju. Integracija sustava implementiranoga u ROS-u i Neptus grafičkog sučelja izvršena je korištenjem Intermodule Communication (IMC) protokola koji se koristi za međusobnu komunikaciju umreženih vozila, senzora i operatera. Protokol ne pretpostavlja specifičnu softversku arhitekturu za klijentske aplikacije. Postoji nativna podrška za različite programske jezike i arhitekture što rezultira optimiran kodom koji se može koristiti za mrežne sustave, ali i za komunikaciju među procesima na istom sustavu. Neptus nativno podržava IMC protokol, dok je unutar LABUST-a razvijen adapter za protokol u ROS-u, čime je omogućena komunikacija Neptusa sa vozilom.

Korisničko sučelje

Korisničko sučelje za konfiguraciju i dijagnostiku izrađeno je u obliku mrežne aplikacije čime je omogućen pristup vozilu korištenjem internetskoga preglednika čime je omogućeno pokretanje korisničkog sučelja na različitim hardverskim platformama poput osobnih računala, tableta ili mobitela, te na različitim operacijskim sustavima koji se pokreću na odgovarajućem hardveru. Korištenjem VPN-a moguće je daljinsko povezivanje s mrežom i u slučajevima kad klijent nije povezan direktno na lokalnu mrežu vozila.

Mrežno grafičko sučelje implementirano je korištenjem više različitih tehnologija. Korišteni su JavaScript/TypeScript, Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) , te razvojni alat Angular. JavaScript je visoki tumačeni jezik (engl. high-level interpreted language) koji se zajedno s HTML-om i CSS-om koristi pri izradi web aplikacija i web stranica. Omogućava interaktivne i dinamične web stranice. Za izradu aplikacije korišten je TypeScript, nadskup JavaScripta, koji znatno nadopunjuje sposobnosti JavaScripta poput mogućnosti korištenja koncepta objektnog programiranja. Sva logika (npr. izračuni pozicija objekta na ekranu) potrebna za rad aplikacije, pisana je u TypeScriptu. HTML se koristi zajedno sa CSS-om za definiranje izgleda web aplikacija i web stranica. Dok se CSS primarno koristi za definiranje izgleda elemenata web aplikacije ili stranice kao što su: boje pozadina elemenata, boja, font, stil, veličina teksta, HTML definira strukturu aplikacije/stranice, definira grupe objekata, format prikaza tih objekata te kako se određeni objekti odnose prema drugim objektima (raspored/pozicija dijelova stranice kao: gumbi, izbornici, itd.). Angular je platforma otvorenog tipa za razvijanje web aplikacija i stranica bazirana na TypeScriptu. Velika prednost navedene platforme je mogućnost pokretanja istoga izvornog koda na različitima hardverskim i softverskim platformama. Angular omogućava laganu integraciju već postojećih komponenti poput Angular Materiala koji definira izgled i animacije nekih često korištenih elementa u izradi web aplikacija i stranica, kao što su: gumbi, izbornici, prekidači, itd.

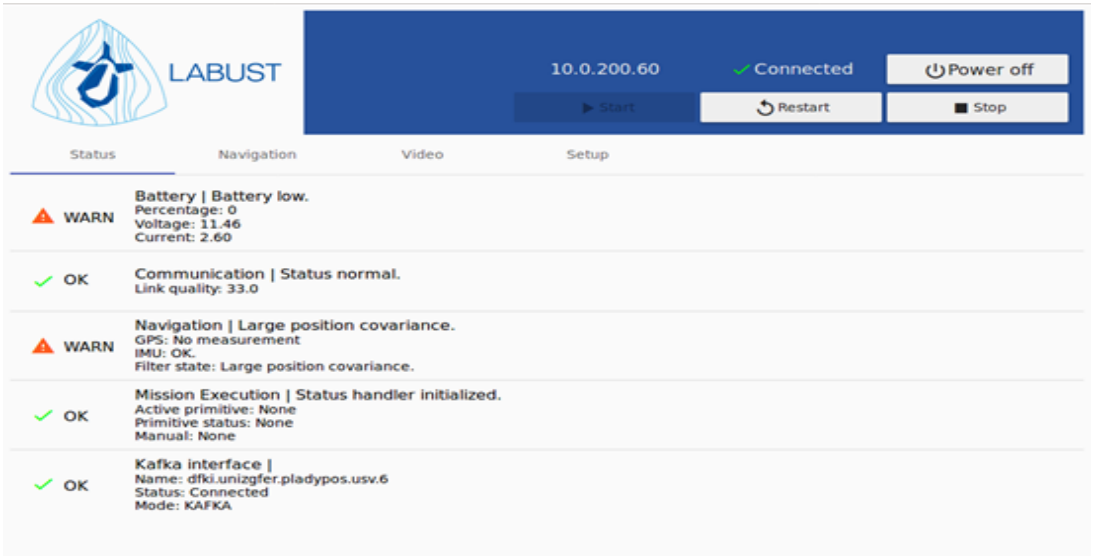
Web sučelje implementirano za nadzor vozila sastoji se od osnovnoga ekrana i niza kartica (eng. Tabs) koje sadrže razne informacije o pojedinim podsustavima vozila. U nastavku su detaljnije pojašnjenje pojedine komponente korisničkoga sučelja.

Status kartica

Status kartica prikazuje dijagnostičke informacije različitih podsustava vozila. Standardni podsustavi koji se nadziru su:

- Baterija – prikaz preostalog kapaciteta, napona i struje baterije
- Komunikacija – prikaz kvalitete Wi-Fi veze
- Navigacija – prikaz statusa navigacijskih senzora i pouzdanosti estimata pozicije
- Izvođenje misije – prikaz statusa misije i moda upravljanja

Osim navedenih podsustava, status se prikazuje i za ostale podsustave vozila ovisno o njegovoj konfiguraciji.

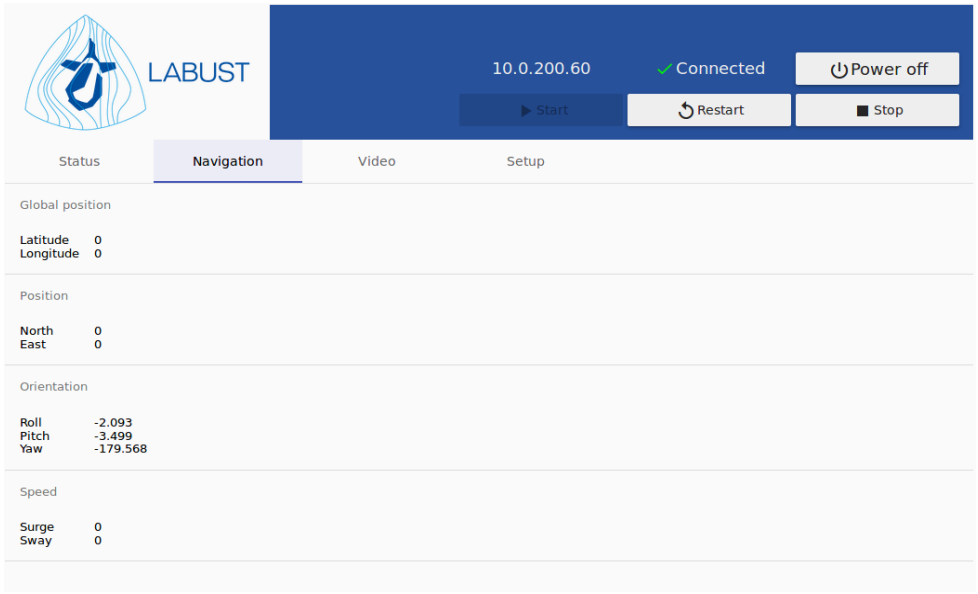


Slika 5 Status kartica

Navigacijska kartica

Navigacijska kartica prikazuje osnovne navigacijske podatke pružene od strane navigacijskoga filtra. Prikazani su sljedeći podaci:

- Geografska širina i dužina u stupnjevima
- Pozicija u lokalnom NED (eng. North-East-Down) koordinatnom sustavu u metrima
- Orijentacija u lokalnom NED koordinatnom sustavu u stupnjevima
- Brzine napredovanja i zanošenja vozila u metrima po sekundi.

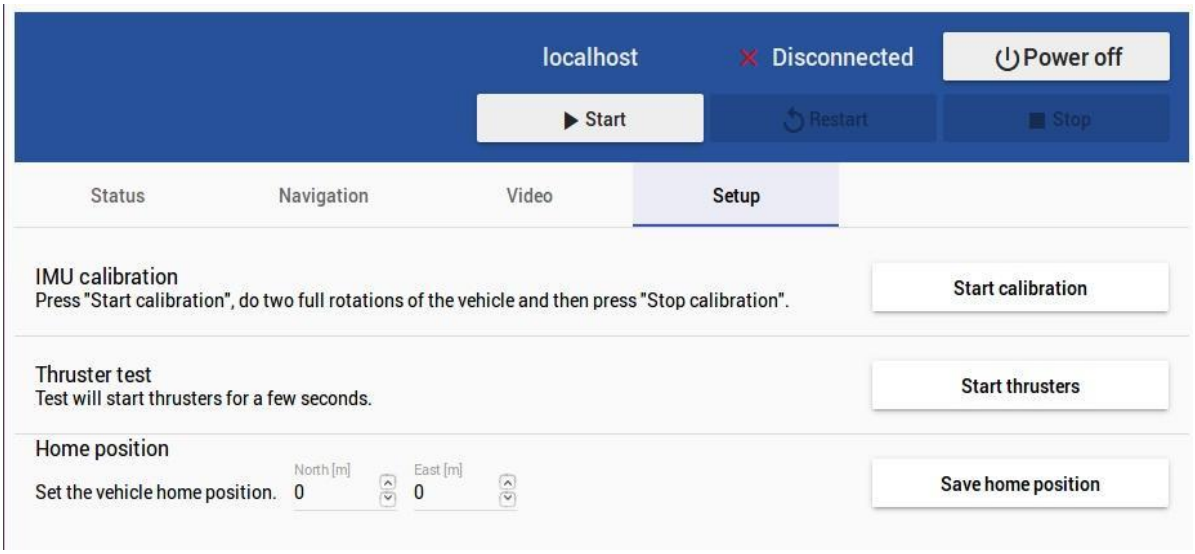


Slika 6 Navigacijska kartica

Konfiguracijska kartica

Konfiguracijska kartica omogućava jednostavno postavljanje i dijagnostiku vozila. Neke od opcija su:

- Kalibracija kompasa
- Dijagnostika pogonskog sustava
- Postavljanje početne pozicija vozila, odnosno lokacije gdje se vozilo vraća u slučaju gubitka komunikacije sa operatorom.



Slika 7 Konfiguracijska kartica

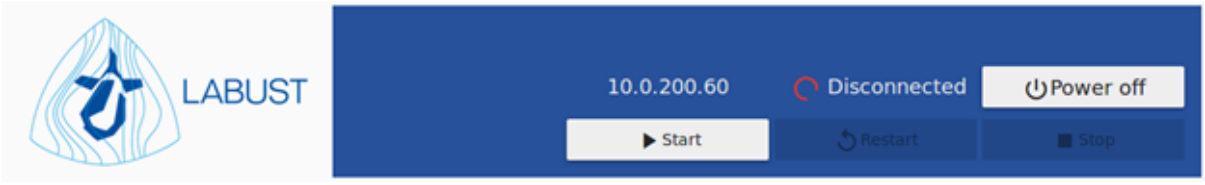
ROS kontrole

Glavne kontrole za pokretanje, zaustavljanje i resetiranje upravljačkoga softvera baziranoga u ROS-u, nalaze se na vrhu upravljačkoga sučelja i stalno su dostupne operateru. Dostupne su sljedeće opcije:

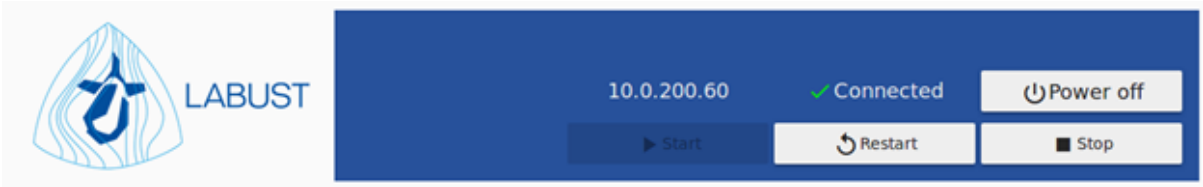
- "Start" - Pokretanje upravljačkoga softvera
- "Restart" - Ponovno pokretanje upravljačkoga softvera
- "Stop" - Zaustavljanje upravljačkoga softvera
- "Power off" - Gašenje glavnoga računala

Na slikama 8, 9, i 10 prikazani su različiti statusi upravljačkoga softvera i gumbe za odabir odgovarajućih opcija. Zatamnjeni gumbi predstavljaju nedostupne opcije u pojedinom slučaju.

Slika 8 ROS upravljački sustav nedostupan



Slika 9 Povezivanje s ROS upravljačkim sustavom

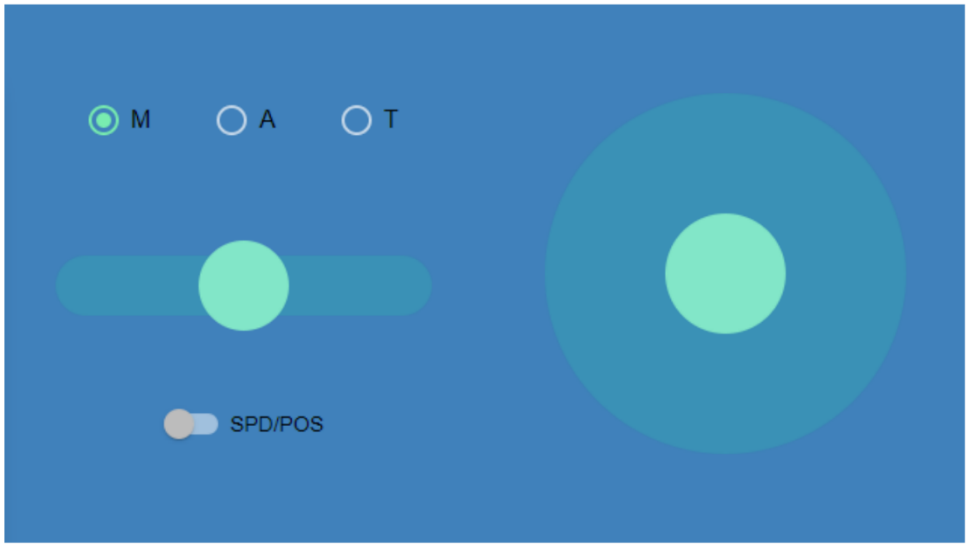


Slika 10 ROS upravljački sustav povezan

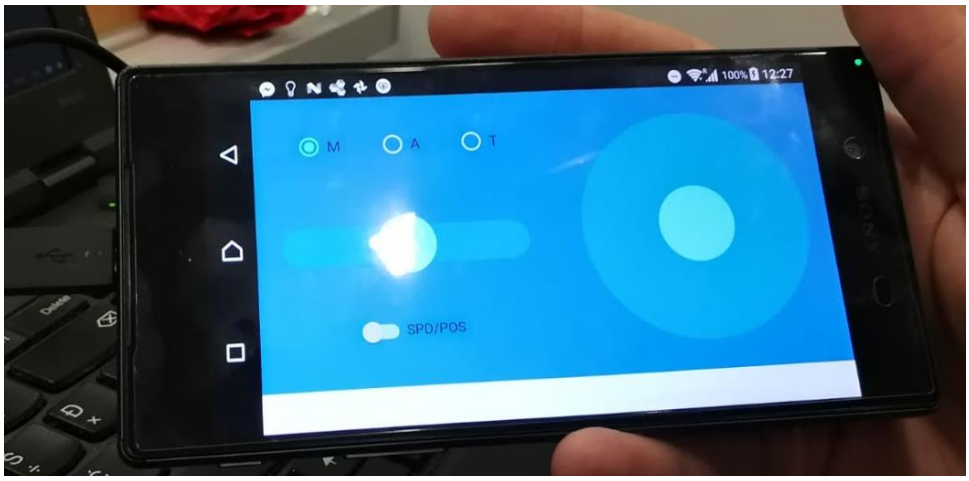
Aplikacija za daljinsko upravljanje površinskim vozilom

Osim softvera za automatsko upravljanje vozilom, razvijen je i sustav za ručno upravljanje vozilom u vidu hibridne web aplikacije. Aplikacija se može izvoditi na mobilnim uređajima, ali se može koristiti i integrirana u predstavljeno web sučelje. Slika 11 prikazuje izgled aplikacije za daljinsko upravljanje, dok su pojedini elementi sučelja prikazani na Slici 13. Slika 12 prikazuje izvođenje aplikacije na Android mobilnom uređaju.

Aplikacijom je moguće odabrati tri načina rada preko tzv. radio tipki (engl. radio button) koje osiguravaju da je aktivan samo željeni način upravljanja. Načini upravljanja su: ručno upravljanje (engl. manual mode - M), automatsko upravljanje (engl. automatic mode - A), praćenje (engl. tracking - T). Ručni način upravljanja omogućava kontroliranje smjera i brzine preko upravljačke ručice (engl. joystick) te kontroliranje rotacije (pozicije oko vertikalne z-osi) preko dvosmjernog klizača (engl. slider). Moguće je odabrati i između dva načina upravljanja rotacijom: držanje određenog kuta ili dinamično definiranje kuta. Osim tehnologija spomenutih u prethodnoj cjelini, korištena je i HammerJS biblioteka. HammerJS omogućava definiranje upravljanja dodirom (engl. touch input) nad elementima.



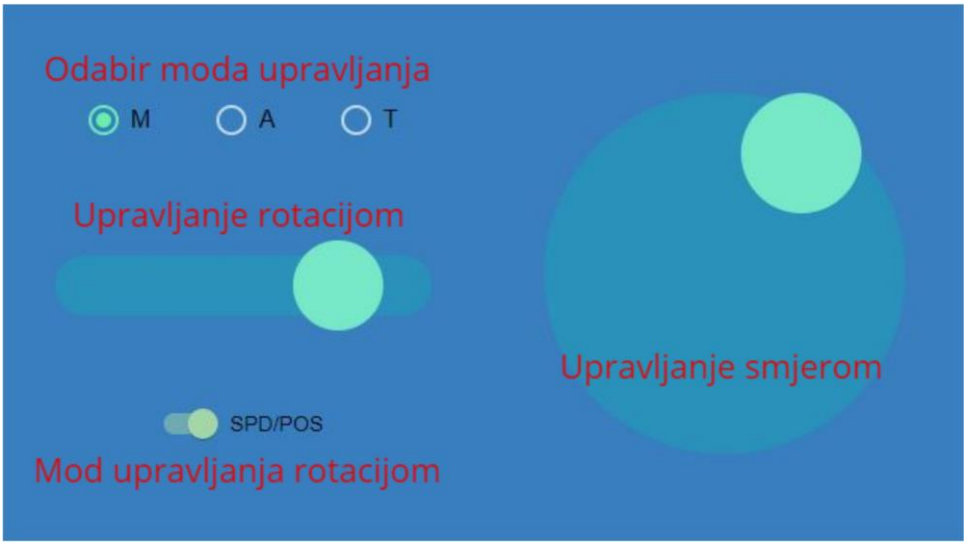
Slika 11 Aplikacija za daljinsko upravljanje površinskim vozilom



Slika 12 Aplikacija za daljinsko upravljanje površinskim vozilom pokrenuta na Android mobilnom uređaju

Dvosmjerni klizač koristi se za definiranje orijentacije kontroliranog vozila. Namješta se u kojem smjeru se želi da pokazuje "prednji" dio kontroliranog vozila (vozilo se okreće oko vertikalne, odnosno z-osi). Kontroler emitira udaljenost od svog centra pretvorenu u kut od -180◦ do +180◦ Ovaj kontroler ima dodatnu funkcionalnost da kada je selektiran POS mode preko prekidača, kontroler zadržava svoju poziciju i nakon što ga je korisnik pustio. To omogućava zadržavanje željenog kuta, odnosno orijentacije vozila.

Upravljačka ručica (engl. Joystick) koristi se za definiranje smjera i brzine kretanja. Tako definirani kontroler mora emitirati signale korisne za kontroliranje željenog vozila. Ovaj kontroler emitira udaljenost od svog centra (radijus) i kut naspram linije koja prolazi kroz njegov centar kako bi se zadala željena referenca upravljačkom sustavu na vozilu.



Slika 13 Aplikacija za daljinsko upravljanje površinskim vozilom sa označenim interaktivnim elementima

Zaključak

U ovom izvješću pokazan je napredak u razvoju softvera za upravljanje misijom i korisničkom sučelju i ostvareni su praktično svi zahtjevi definirani izvješću D5.1 „Zahtjevi i arhitektura softvera za upravljanje misijom“. Razvijeni i integrirani softver omogućava upravljanje i nadzor većeg broja plovila što je potvrđeno istovremenim korištenjem tri vozila na terenskim eksperimentima o čemu će više navedeno biti u izviješću D3.3. Također upravljački softver je neovisan o manevarskim sposobnostima vozila budući da upravljački softver ovisno o konfiguraciji vozila poziva odgovarajući primitiv (podaktuirani ili nadaktuirani “go2point” primitiv) kako bi ostvario željenu misiju. To je moguće zahvaljujući zahtjevu da se kompleksne misije mogu izvršiti kombiniranjem jednostavnih akcija, odnosno primitiva. Zahvaljujući korištenju web tehnologija, pri izradi korisničkoga sučelja, ono se može izvoditi na različitim hardverskim platformama i operativnim sustavima. Također omogućen je jednostavan daljinski pristup, budući da i udaljeni klijent se može povezati na vozilo pod uvjetom da vozilo ima pristup internetu. U današnje vrijeme to je često ostvareno integracijom 3G/4G modula na vozilo. Time se omogućuje i daljinska dijagnostika vozila i dojava kvarnih stanja operateru. Konačno, sav razvijeni softver je temeljen na softveru otvorenoga koda, i dostupan s licencom otvorenoga koda čime se direktno doprinosi razvoju zajednici pomorske robotike.