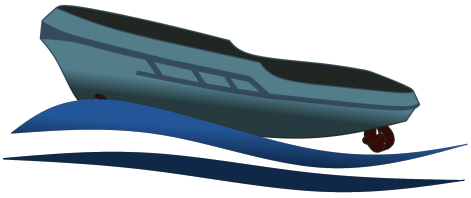
Användarhandledning

Oscar Hermansson, Oskar Jonsson, Jonatan Siönäs,
Axel Ståhlbom, Johannes Wenngren, Tim Wiik

8 december 2021

Version 1.0



Status

Granskad	Oscar, Jonatan	2021-11-30
Godkänd		

Projektidentitet

Beställare: Fredrik Ljungberg, Linköpings universitet
E-post: fredrik.ljungberg@liu.se

Kund: Jonas Linder, ABB Corporate Research
E-post: jonas.x.linder@se.abb.com

Handledare: Gustav Zetterqvist
E-post: gustav.zetterqvist@liu.se

Kursansvarig: Daniel Axehill och Gustaf Hendeby
E-post: daniel.axehill@liu.se, gustaf.hendeby@liu.se

Projektdeltagare

Namn	**Ansvar**	**E-post**
Oscar Hermansson	Dokumentansvarig	osche354@student.liu.se
Oskar Jonsson	Hårdvaruansvarig	oskjo275@student.liu.se
Jonatan Siönäs	Designansvarig	jonsi043@student.liu.se
Axel Ståhlbom	Mjukvaruansvarig	axest992@student.liu.se
Johannes Wenngren	Projektledare	johwe475@student.liu.se
Tim Wiik	Testansvarig	timwi572@student.liu.se

INNEHÅLL

DOKUMENTHISTORIK

Version	Datum	Utförda förändringar	Utförda av	Granskad
0.1	2021-11-23	Första utkast	Samtliga	Oscar
0.5	2021-11-28	Andra utkast	Samtliga	Johannes
1.0	2021-11-30	Första revision	Samtliga	Oscar, Jonatan

1 INTRODUKION TILL FARKOSTEN

Det här dokumentet utgör användarmanual för projektet ”Modellbaserad styrning av småskalig ytfarkost” i kursen TSRT10. I kapitlen nedan beskrivs först hur farkosten startas och sen hur varje delsystem är uppbyggt och används. Figur 1 visar hur farkosten ser ut och bilderna kommer från Zetterqvist och Steens examensarbete [1].

Figur 1: Bild av systemet

2 UPPSTARTSPROCESS

Nedan följer en kort sammanfattning som beskriver hur farkosten startas. För en mer utförlig beskrivning, se projektets git repo.

Koppla in ett batteri i båten för ström och kontrollera att routern har internetuppkoppling. Farkosten är föreberedd för att automatiskt koppla upp sig mot routern. Koppla upp en extern dator till routern (RT_labb, lösenord: Vc6u4E0G) och använd SSH (Secure Shell) med IP adress: 192.168.1.120 för att kunna skicka kommandon till farkosten trådlöst. Farkosten går även att använda med Raspberry Pi:en kopplad till en skärm tillsammans med mus och tangentbord för underlättad kodskrivning eller utförande av enkla tester. I Raspberry Pi:ens *bash rc* har kommandot

```
sudo pigpiod
```

lagts till vilket aktiverar I/O-portarna automatiskt när en terminal öppnas. Ifall Raspberry Pi:en skulle behöva nollställas eller liknande måste kommandot skickas manuellt innan körning. När det grafiska gränssnittet används, i en terminal lokaliserad i hemmappen, kan filen *launch_withGUI.sh* köras för att starta båtens programvara. Använd följande kommandon för att göra detta

```
cd git/boat/

./launch_withGUI.sh
```

Ifall farkosten vill startas utan det grafiska gränssnittet kan istället följande kommandon användas:

```
cd git/boat/

./launch.sh
```

Utan det grafiska gränssnittet antas RTK-antennen vara vid blå lagunen. Se till att den tillhörande handkontrollen är påslagen och säkerhetsbrytaren är frånslagen. Öppna en ny terminal för att skicka kommandon till farkosten som till exempel

```
rostopic pub /ship_command boat/ShipCommand "{velocity_1: 700, angle_1: 90}"
```

vilket ändrar hastigheten och vinkelutslaget för roderpropeller 1 till angivna värden.
För att läsa data som publiceras på ett specifikt *topic* används kommandot

```
rostopic echo /topic
```

för att skriva ut datan som publiceras på *topic*. En lista med tillgängliga *topics* hittas genom att skriva

```
rostopic list
```

För att få realistiska värden från magnetometern behöver den kalibreras innan körning. Detta görs genom att öppna en ny terminal och skriva in

```
cd git/boat/calibration
```

följt av

```
python3 calibrate_magnetometer.py
```

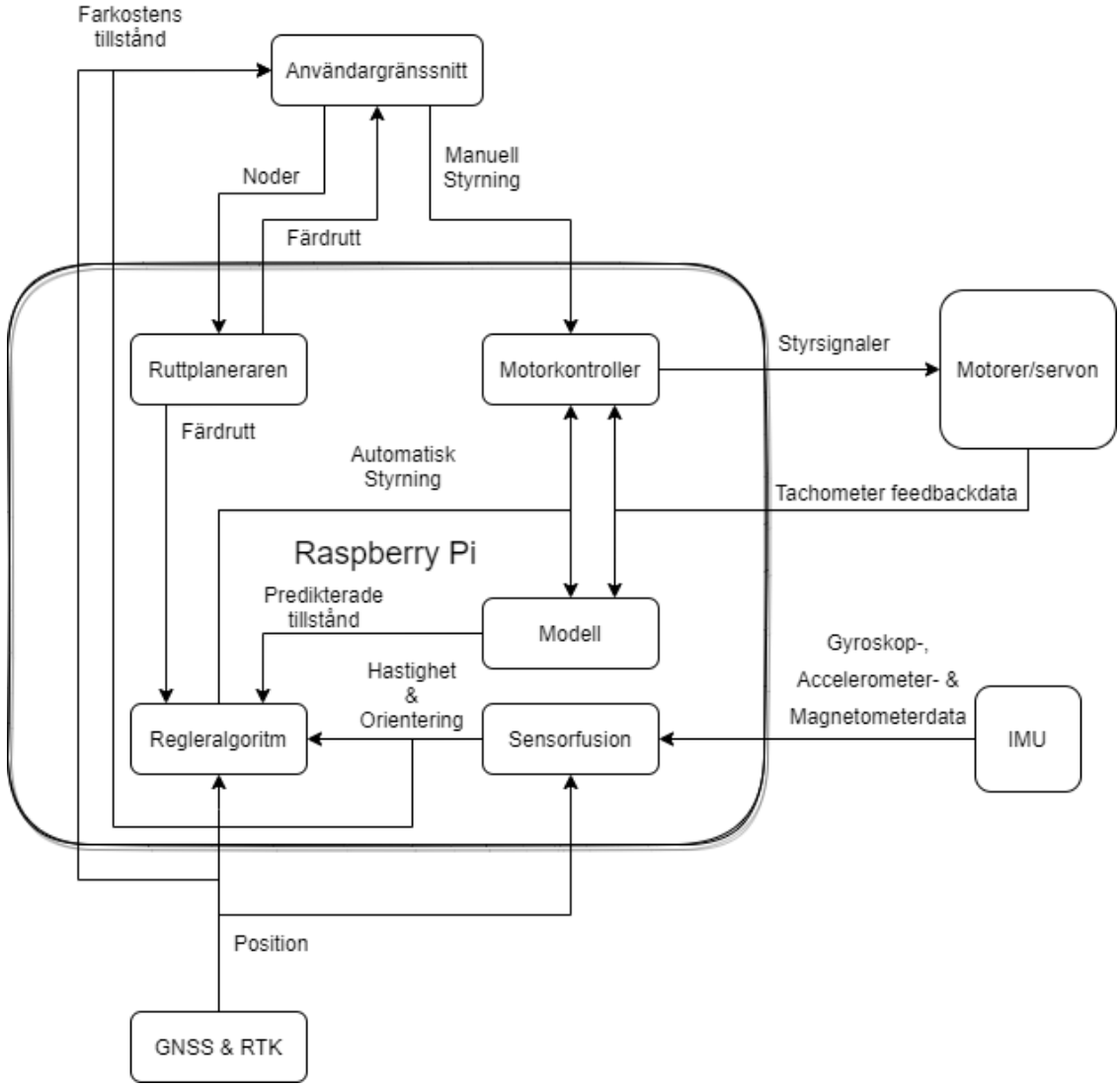
för att starta kalibreringen. Rotera sedan farkosten ett varv i det horisontella planet och stäng av kalibreringen genom att använda

```
ctrl+c
```

Ifall matrisen som skrivs ut i terminalen visar *NaN*, gör om kalibreringen. Under kalibreringen bör det inte finnas ytterligare elektronik som telefoner, datorer eller smarta klockor i båtens omedelbara närhet.

Farkosten kan styras manuellt med handkontroll eller genom SSH. Dessa två lägen kan väljas genom en spak på handkontrollen, se avsnitt 3.3 för vidare beskrivning.

Figur 2 ger insikt i hur farkosten fungerar. Den visar hur de olika modulerna på farkosten kommunicerar med varandra och vilken information som skickas.



Figur 2: Flödesschema för systemets olika moduler.

3 FARKOSTENS HÅRDVARA

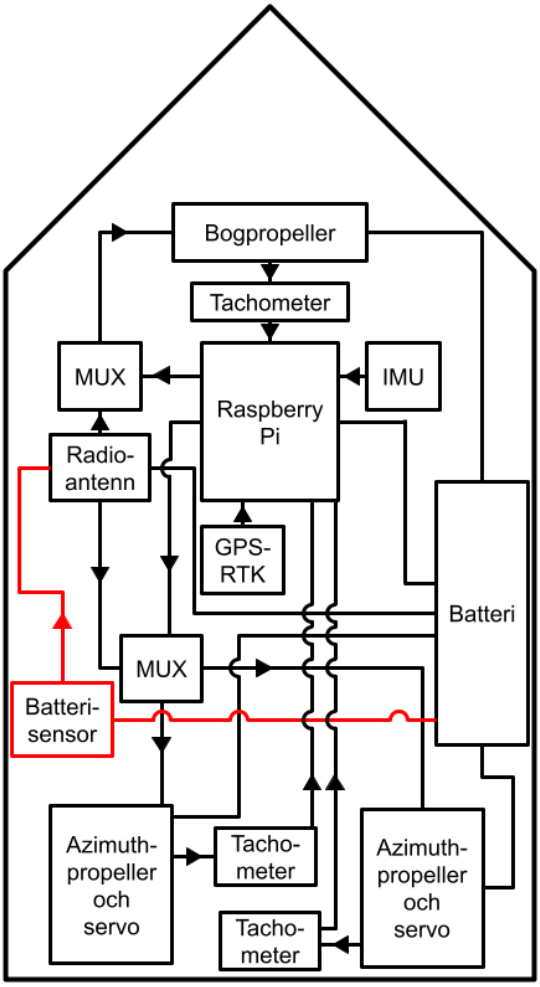
Nedan ges förklaringar på vad farkosten olika hårdvarukomponenter används till.

3.1 Komponenter

Farkostens komponenter ser ut som följande

- Två stycken roterbara roderproppellrar och tillhörande servon. Roderpropellrarna drivs av varsin borstlös motor av typen ”G-Power BL-Motor, type 3536”.
- En bogpropeller driven av en borstlös motor av typen ”G-Power BL-Motor, type 3536”.
- En GNSS med RTK (Real Time Kinematic). Modulen är en u-blox ZED-F9P och har en MagmaX2 extern antenn.
- En IMU (Inertial Measurement Unit) av typen MPU9250 för mätning av vinkelhastighet, acceleration och magnetfält
- Tachometer för propellrarna.
- En X8R - FrSky RC mottagare för användning vid manuell styrning av skeppet.
- En Raspberry Pi 4
- Flera litiumjonbatterier med spänningen på 11.1 V och kapacitet på 2200 mAh (Turnigy 2200mAh 3S 25C LiPo Pack), varav endast ett åt gången kopplas in och strömförsörjer båten.
- Propellermotorerna har varsin ESC ("Electronic Speed Controller"eller elektrisk hastighetsregulator på svenska) av typen Graupner HoTT +T 35 BEC.
- En spänningsregulator som genererar 5.0 volt (och max 5 Ampere) till Raspberry Pi:en.
- En spänningsregulator som genererar 7.0 volt (och max 20 Ampere) till servomotorerna.
- En FrSky lipo sensor för övervakning av batterinivån i farkosten.

Hur dessa komponenter är sammankopplade syns i Figur 3.



Figur 3: Systemskiss av farkostens komponenter

3.2 Raspberry Pi

Farkostens Raspberry Pi använder sig av ROS 1 (Robot Operating System version *noetic*) och hanterar all mjukvara för ingående moduler som sensorfusion, reglering och kommunikation med en extern dator. Hur mjukvaran används förklaras tydligare i kapitel 4, 6 och 7.

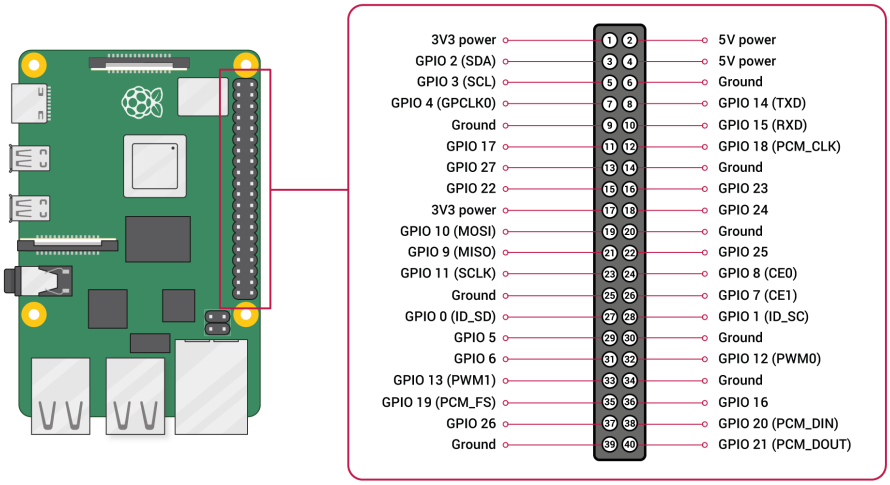
För att kunna koppla upp en extern dator till Raspberry Pi behöver man ange processorns IP-adress.

```
Raspberry Pi IP: 192.168.1.120
```

I Tabell 1 och Figur 4 redovisas alla anslutningar till Raspberry Pi:ens in- och utgångar samt GPIO pinnar. Jord (GND) ansluts mellan varje modul som kommunicerar med Raspberry Pi:en. Notera även att ESC betyder ”Electronic Speed Controller” eller elektrisk hastighetsregulator på svenska.

Raspberry pi pin	Ansluten till	Kommentar
GPIO 2 (SDA)	IMU SDA	I2C-data till IMU
GPIO 3 (SCL)	IMU SCL	I2C-klocka till IMU
GPIO 0	GNSS-mottagare RX	UART-data från Raspberry Pi till GPS
GPIO 1	GNSS-mottagare TX	UART-data från GPS till Raspberry Pi
GPIO 14	GNSS-mottagare RXD2	RTCM korrektionsdata från Raspberry Pi till mottagaren
GPIO 15	GNSS-mottagare TXD2	RTCM korrektionsdata från mottagaren till Raspberry Pi
GPIO 17	Tachometer 1	Läser av tachometer till azimuthpropeller numer 1
GPIO 27	Tachometer 2	Läser av tachometer till azimuthpropeller numer 2
GPIO 22	Tachometer T	Läser av tachometer till bogpropellern
GPIO 16	ESC till azimuthpropeller numer 1	Styra hastigheten på azimuthpropeller 1
GPIO 20	ESC till azimuthpropeller numer 2	Styra hastigheten på azimuthpropeller 2
GPIO 21	ESC till bogpropeller	Styra hastigheten på bogpropellern
GPIO 19	Servo till azimuthpropeller nummer 1	Styr vinkel på azimuthpropeller 1
GPIO 26	Servo till azimuthpropeller nummer 2	Styr vinkel på azimuthpropeller 2

Tabell 1: Raspberry pi anslutningar

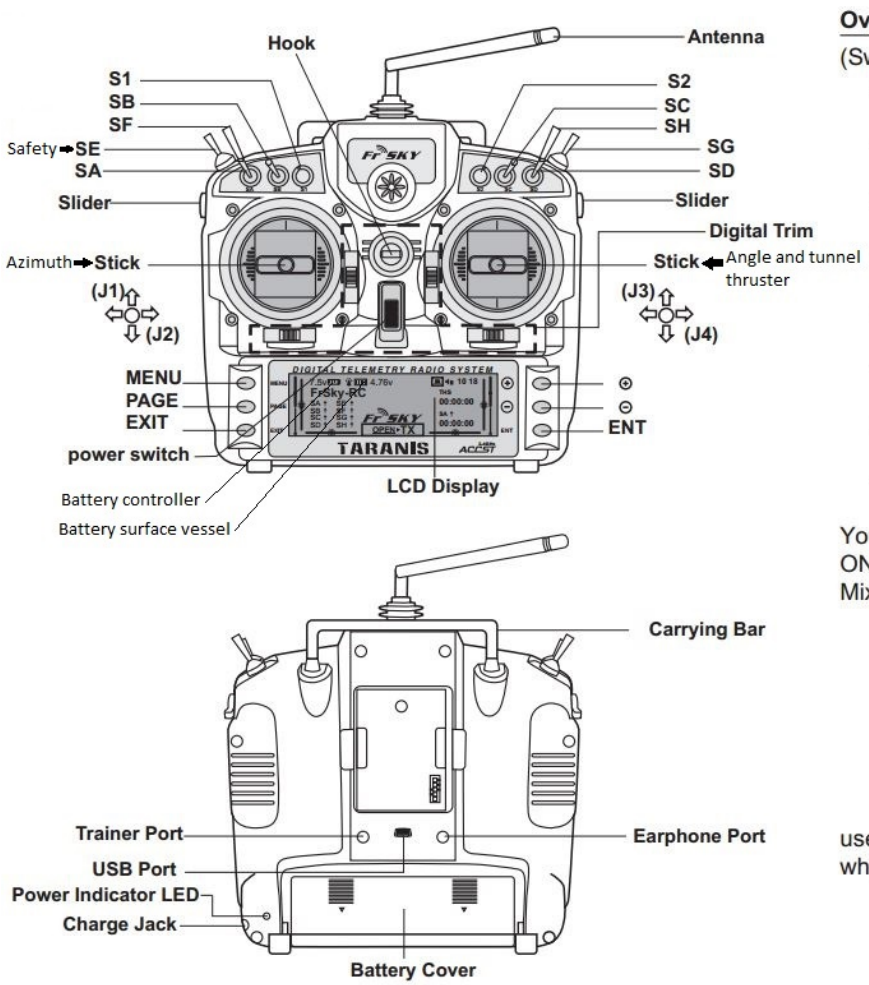


Figur 4: Schema över Raspberry Pi 4b:s pin-positioner.

3.3 Handkontroll

För manuell styrning av farkosten finns en handkontroll med all väsentlig funktionalitet. För att använda handkontrollen och styra farkosten behöver först kontrollen startas genom att skjuta på brytarn på handkontrollens framsida. Handkontrollens skärm kommer börja lysa och olika meddelanden kommer visas. Tryck på '*enter*'-knappen för att gå igenom alla meddelanden tills huvudmenyn visas. I övre vänstra hörnet på skärmen visas handkontrollens- och farkostens batterinivå som bör noga övervakas vid körning. Fartyget bör inte köras om batterinivån är under 10 volt. För att starta manuell styrning måste säkerhetsbrytaren slås av (spak nedåt) vilket är markerat med '*SE*' på kontrollens övre vänstra hörn.

Nu är handkontrollen kopplad till farkosten vilket gör det möjligt att styra farkosten manuellt. Handkontrollens vänstra styrspak kontrollerar roderpropellrarnas hastighet som uppåt för högre hastighet och nedåt för lägre hastighet vilket inkluderar möjligheten att backa. Den högra styrspaken styr vinkelutslaget på roderpropellrarna och rotationshastigheten på bogpropellern. Flytta styrspaken åt höger eller vänster för att ändra vinkelutslaget och få farkosten att svänga höger respektive vänster. Flytta styrspaken upp eller ner för att använda bogpropellern. Figur 5 visar hur handkontrollen ser ut och namnet på respektive knapp. Den ursprungliga bilden kommer från handkontrollens användarmanual [2].



Overview

(Switch Default Settings)

- SA: 3 positions, Alternate; Short Lever
- SB: 3 positions, Alternate; Long Lever
- SC: 3 positions, Alternate; Long Lever
- SD: 3 positions, Alternate; Short Lever
- SE: 3 positions, Alternate; Short Lever
- SF: 2 positions, Alternate; Long Lever
- SG: 3 positions; Alternate; Short lever
- SH: 2 positions, Momentary; Long Lever

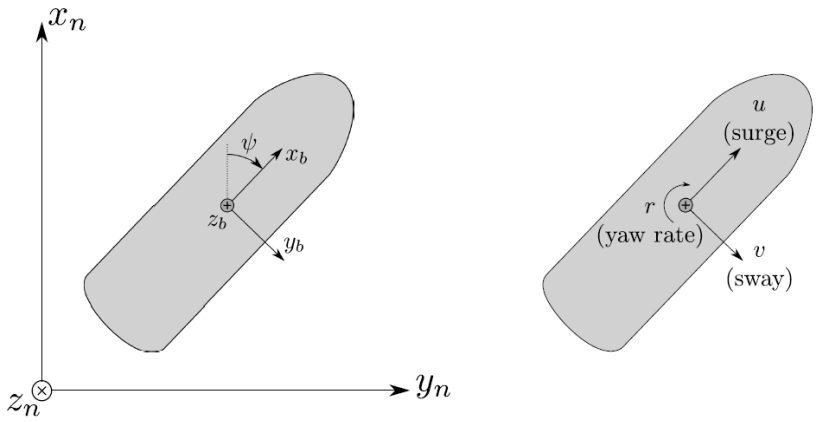
You can choose the Switch and the ON/OFF position in the menu of Mixer.

use earphones with volume control when applied.

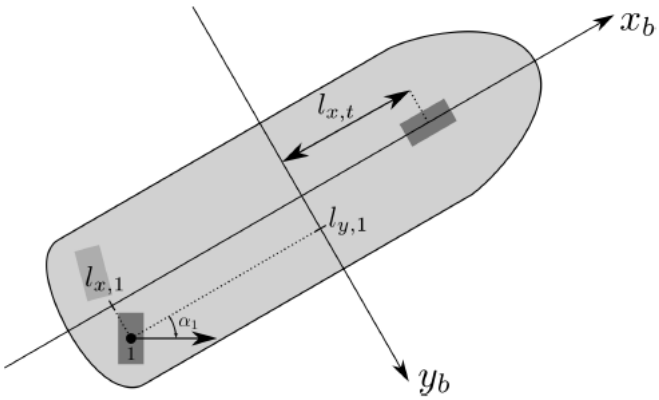
Figur 5: Figur beskrivande hankontrollens utseende

4 MODELL

Modellen används för att prediktera var farkosten kommer vara i framtiden vilket används vid regleringen. Det som modelleras är farkostens position, vinkel och vinkelhastighet genom att ta in rotationshastigheten och vinkelutslag för roderpropeller 1 respektive 2 samt bogpropellerns rotationshastighet. Hur dessa är definierade syns i Figur 6 och 7.



Figur 6: Beskrivning över det lokala koordinatsystemet



Figur 7: Beskrivning över aktuatorernas placering och rotation

För att skatta parametrarna i modellen behövs en datainsamling göras som sedan behandlas i *Matlab*. För att logga datan från en datainsamling, använd ssh för att skicka kommandon till farkosten. För att börja logga data för en datainsamling, öppna en ny terminal och skriv in

```
cd git/boat/catkin_ws
```

följt av

```
./ log_data.sh
```

Skriv in namnet som *bag*-filen där datan sparas kommer ha och tryck på *enter* för att starta loggning av data. När datainsamlingen är färdig, använd

```
ctrl+c
```

för att avsluta loggningen. *bag*-filen sparas i mappen *bagfiles* vilket nås genom att stå i hemmappen och skriva

```
cd git/boat/testdata/bagfiles
```

i en terminal.

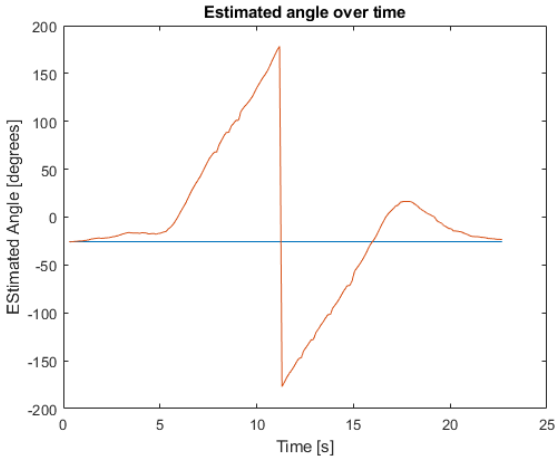
För att använda datan behöver den skickas till en dator med *Matlab* installerat och på denna dator öppna *estimation.m*. Skriv in namnet på *bag*-filen och definiera den som estimeringsdata och gör motsvarande med en annan *bag*-fil för valideringsdata. Starta *estimation.m* vilket kommer plotta estimeringsdatan, skatta parametrarna i modellen, plotta estimeringsresultatet och till sist plotta resultatet av valideringen. Det finns olika varianter av modellen som används och den aktuella modellen ändras genom att ändra

```
given_model
```

till namnet på den önskade modellen i *estimation.m*. Kom ihåg att lägga till eller ta bort initiala parametervärden.

5 SENSORFUSION

Sensorfusionen används för att skatta farkostens vinkel och hastighet (relativt det globalt definierade koordinatsystemet) vilket syns i användargränssnittet och används vid regleringen och modelleringen. Den använder ett Kalman filter som approximerar att farkosten ligger helt plant i vattnet och med mätvärden från *IMU*(*Inertial Measurement Unit*) och *GPS*(*Global Positioning System*) skattas tillstånden. Sensorfusionen publicerar den skattade vinkeln och hastigheterna på *topic /imu/sens_fus*. Använd kommandot som presenterades i uppstartsprocessen för att titta på datan i realtid. Ett exempel på den skattade yaw-vinkeln syns i Figur 8 där farkosten har roterats ett helt varv.



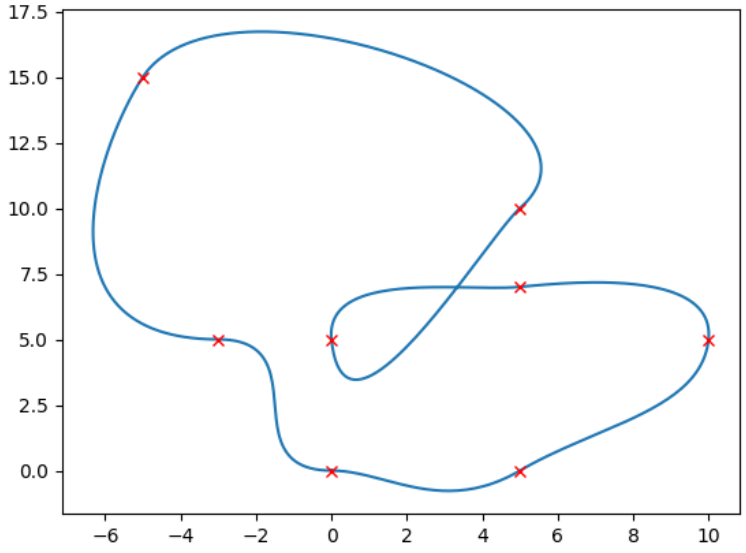
Figur 8: Plott över den skattade yaw-vinkeln under en testkörning där farkosten roterades ett helt varv

6 FÄRDRUTTSPLANERING

Färdruttsplaneringen interpolerar en tredjegradsekvation mellan givna positioner. Funktionen, som återfinns i `boat/catkin_ws/src/boat/scripts/interpolation.py`, kallas enligt:

```
interpolation(node_x,node_y,node_ang,max_c=10,gamma1=1,gamma2=20,N=50,plot_data=False)
```

där `node_x/y/ang` är listor med punkterna som interpolationen ska sammanfoga och `max_c` $\left[\frac{1}{m}\right]$ är den maximalt tillåtna kurvaturen. Eftersom tredjegradsekvationerna kommer bli underbestämda testas olika kombinationer med konstanter och dessa vägs därefter genom en värdefunktion med vikten `gamma1` för längden på den resulterande kurvan och `gamma2` som vikten på den kurvatur som resultatet maximalt uppnår. `N` specificerar hur många delintervall som två punkter kommer kommer delas mellan (fler ger potentiellt bättre resultat, men är tyngre att beräkna) och `plot_data` kan sättas till sann om resultatet ska plottas. Exempel på interpolationen syns i Figur 9.



Figur 9: Ruttplanerarens interpolation mellan noder.

7 REGLERING

Nedan presenteras de regulatorer och utvecklingsmetoder som finns implementerade samt hur ytterligare regulatorer bör definieras för att passa in i systemet.

7.1 Simuleringsmiljö

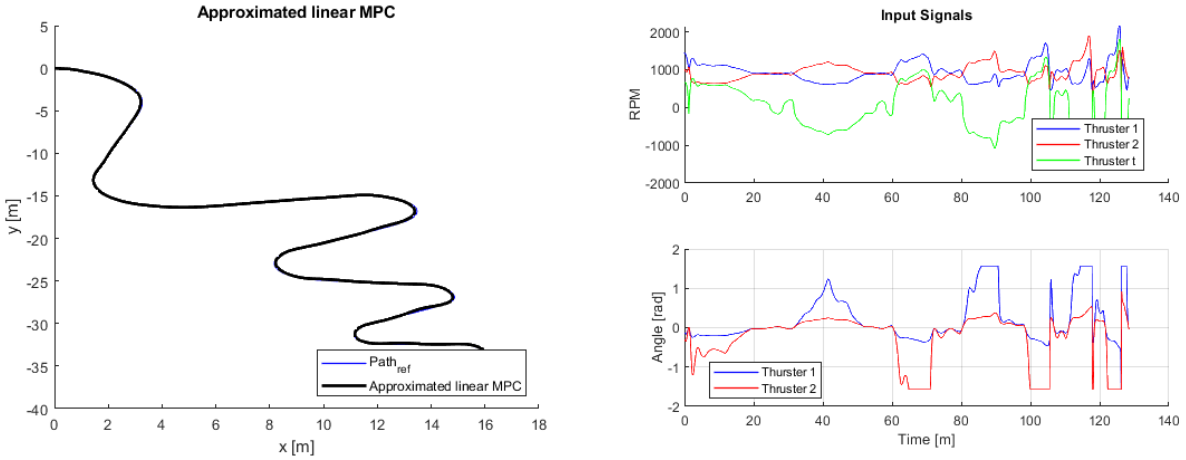
För att testa regleringsprestanda utan att enbart använda hårdvara utvecklades en simuleringsmiljö i MATLAB. Filen `simulation_main.m` skapar ett båtobjekt baserat på modellen av båten, och tillsätter valfri regulator. Banor kan importeras från tidigare testfiler. Nya regleralgoritmer kan enkelt testas i simuleringsmiljön, och kan på så sätt visas vara funktionella innan de implementeras i regulatornoden på båten.

7.2 Implementerade regulatorer

Reglermetoden som används primärt på hårdvaran är MPC (Model Predictive Control) som reglerar farkostens position och vinkel för att följa den bana som erhålles. Även en State-Feedback regulator med konstant hastighet finns implementerad.

7.2.1 *MPC*

MPC bygger på att förutsäga farkostens färdväg för en given styrsignal och sedan ställa ut den styrsignal som ger bäst banföljning. Felet mellan farkostens önskade och faktiska bana minimeras genom kvadratisk programmering i



(a) Banföljning **(b)** Insignaler

Figur 10: MPC i simuleringsmiljön

Python modulen `osqp`. Regulatorn kan finjusteras genom att välja parametrar för matriserna Q_1 och Q_2, där diagonalelementen i matriserna kan ses som straffparametrar vid optimeringen. Elementen i Q_1 straffar avståndsfel, vinkelfel och hastighetsfel medan elementen i Q_2 straffar användandet av styrsignalerna. Notera att för den MPC-regulator som finns implementerad straffar Q_2 krafterna i Newton som orsakas av styrsignalerna. I koden benämns dessa krafter som `cf` och de är definierade enligt:

$$\texttt{cf} = \begin{bmatrix} F_{1x} & F_{1y} & F_{2x} & F_{2y} & F_t \end{bmatrix}^T$$

där x och y index indikerar kraftens komponent i båtens lokala koordinatsystem.

7.2.2 *State Feedback*

Tillståndsåterkopplingen använder sig av referensbanans kurvatur, närmaste avståndet från referensbanan och differensen mellan referensbanans riktning i närmsta punkten och farkostens riktning för att ställa ut en styrsignal till båten. Farkostens roderpropellrar sätts till önskad konstant rotationshastighet. Regulatorn har två parametrar som kan finjusteras, dessa är K_1 och K_2 vilka styr återkopplingen av distansfel respektive riktningsfel.

7.2.3 *Regulatorstruktur*

Regulator-algoritmer kallas genom ROS-noden `ctrl_node.py`. Regleralgoritmen skrivs in som en klass i en egen fil som placeras i mappen `boat/catkin_ws/src/boat/scripts/control` för att sedan importeras till noden. Regulatorer initieras genom att lägga till alternativet i funktionen `set_ctrl_parameters()`. Där kan också regulatorns parametrar ansättas. Regulatorklassen ska innehålla en funktion med följande struktur:

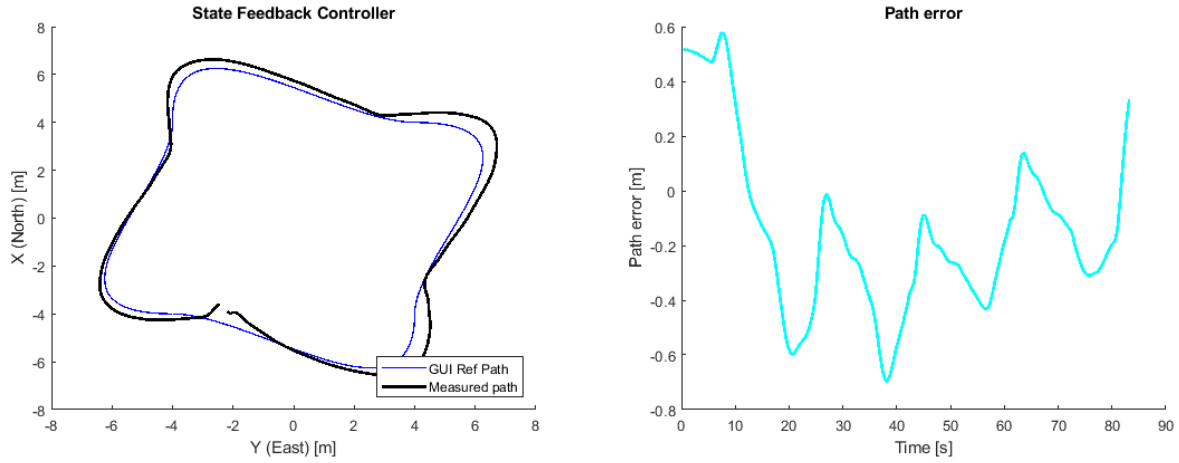
```
u = boat_control(x)
```

Där `x` är tillståndsvektorn för systemet och `u` är önskad styrsignal på formen:

$$\texttt{u} = \begin{bmatrix} n_1 & n_2 & n_t & \alpha_1 & \alpha_2 \end{bmatrix}^T$$

7.2.4 *Regulatorutvärdering*

Regulatorn verkliga prestanda kan visas genom `controller_evaluation.m` skriptet i MATLAB. Referensbana samt uppmätt bana plottas i samma figur, och avståndsfelet under körningen plottas i en separat figur, se Figur 11.



(a) Banföljning **(b)** Avståndsfel

Figur 11: Exempel på regulatorutvärdering.

8 ANVÄNDARGRÄNSSNITTET

Det grafiska användargänssnittet kan användas för att underlätta övervakning och testning av farkosten. När användaragränsittet startas kommer det att se ut som i Figur 13, här kan, i det övre högra hörnet, ses vilka data man kan övervaka. Man kan även se farkostens position och tidigare positioner plottas i grafen som tar upp större delen av fönstret, det går även att zooma in och ut som i en vanlig MATLAB-plot, se övre högra hörnet av plot-rutan i Figur 13.

Det första som krävs är att datorn har MATLAB och dess ROS TOOLBOX installerat. Dessutom behöver *Python 2.7* installeras och läggas till i MATLAB:s *path*, detta steg kan verifieras genom MATLAB-kommandot: `pyenv('Version','2.7')` . GUI-koden finns i `/MATLAB/GUI/GUI.mlapp`.

För att kommunikationen mellan datorn och Raspberry Pi:en ska fungera måste man vid första tillfället lägga till 'hostnames' och IP-adresserna för både datorn och Raspberry Pi:en. För att hitta 'hostname' för en Windows-dator öppnar man kommandotolken och skriver 'hostname', Raspberry Pi:en har 'raspberrypi' som 'hostname'. Bådas 'hostname' och IP-adress ska läggas in i /etc/hosts på Raspberry Pi:en och i /windows/system32/driver/etc/host.
Stäng av alla brandväggar på både Raspberry Pi:en och datorn.
Starta om båda för att ändringarna ska bli aktiva.
Skriv `sudo nano ~/.bashrc` i terminalen på Raspberry Pi:en och lägg till följande rader:

```
export ROS_MASTER_URI=http://Ubuntu_PC_IP:11311
ROS_IP=Ubuntu_PC_IP
```

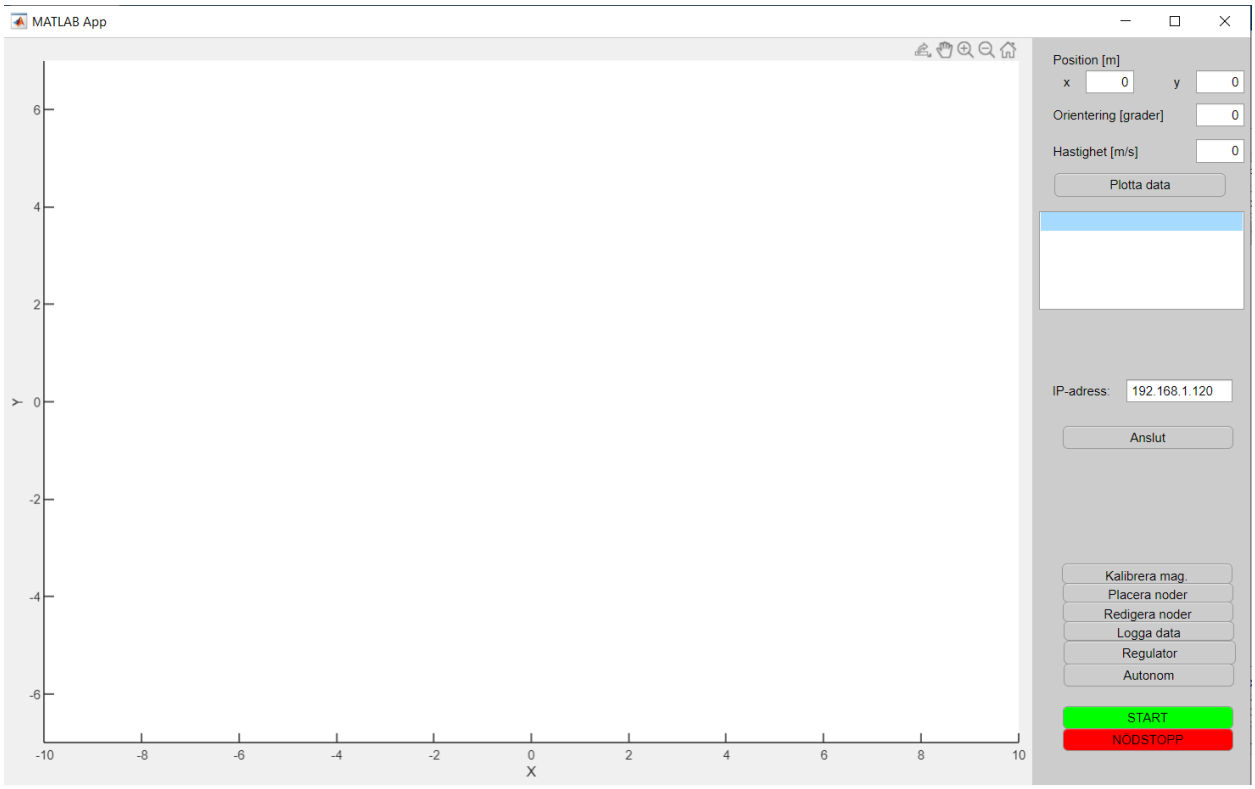
där Ubuntu_PC_IP ersätts med Raspberry Pi:ens IP-adress. Lägg sedan till Windows-datorns IP-adress användargränssnittets kod för startup-funktionen (se Figur 12):
`setenv('ROS_IP','Windows_PC_IP')`
där Windows_PC_IP ersätts med Windows-datorns IP-adress.

```
% Code that executes after component creation
function startupFcn(app)
    % The default IP-address is the one that the Rasberry Pi uses
    % when connected to the router RT_Lab, this will need to be
    % changed when using a different internet access point. The
    % edit field in the GUI can also be used to manually change th
    % IP-address each time the GUI is run.
    app.IPadressEditField.Value = '192.168.1.120';
    app.plotboxratio = app.UIAxes.PlotBoxAspectRatio;
     % Settings for UIAxes
        app.UIAxes.PlotBoxAspectRatioMode = 'manual'
        app.UIAxes.DataAspectRatioMode = 'manual'
        app.UIAxes.DataAspectRatio = [1, 1, 1];
        app.UIAxes.XLim = [-10 10]
        app.UIAxes.YLim = [-7 7]
        app.UIAxes.YLimMode = 'auto'
        app.UIAxes.YLimMode = 'manual'
        app.UIAxes.PlotBoxAspectRatio = app.plotboxratio

    % This will need to be changed when using a new computer or
    % network
    setenv('ROS_IP','192.168.1.238')
    % Ensure that python 2.7 is used
    pyenv('Version', '2.7');
end
```

Figur 12: Koden för startup-funktionen till användargränssnittet.

Nedan följer en steg för steg-guide för hur man startar och sedan använder användargränssnittet.



Figur 13: Användargränssnittets utseende vid uppstart

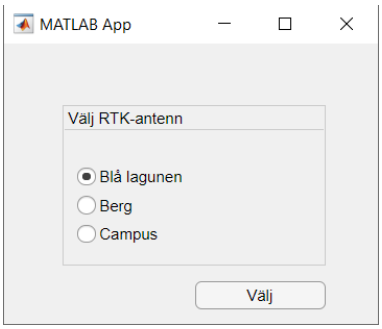
8.1 Steg 1

Det första man måste göra är att ansluta till Raspberry Pi:en, för att åstadkomma detta måste datorn som användargränssnittet används på och Raspberry Pi:en vara anslutna till samma nätverk. Sedan ska man mata in den IP-adress Raspberry Pi:en har på detta nätverk (fältet är förifyllt med adressen som gäller för routern RT_lab) och tryck på anslut.

```
Raspberry Pi IP: 192.168.1.120
Router name: RT_Lab
Router password: Vc6u4E0G
```

8.2 Steg 2

Om anslutningen lyckades kommer en dialogruta där man får välja vilken RTK-antenn man vill använda, se Figur 14.



Figur 14: Dialogrutan för att välja RTK-antenn

8.3 Steg 3

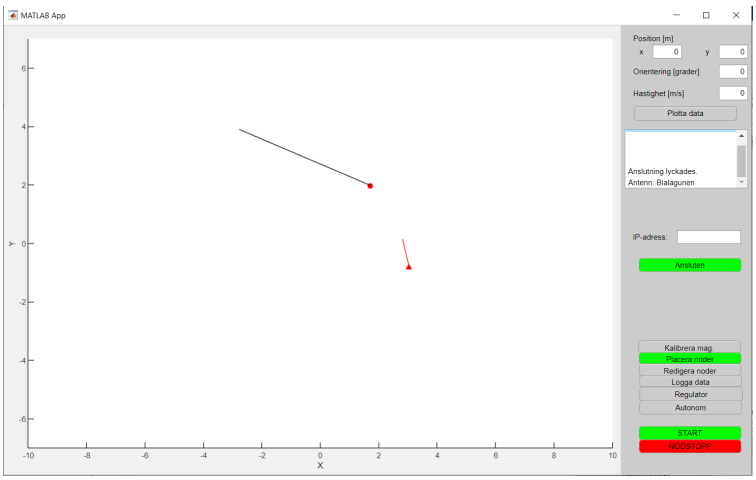
Magnetometern bör kalibreras innan färd påbörjas genom att man trycker på knappen "Kalibrera mag.", rotera farkosten ett varv i det horisontala planet och slutligen tryck på knappen igen för att avsluta kalibreringen. Kontrollera sedan att värdena som dyker upp i rutan är rimliga (t.ex. att det inte finns några NaN"). Det är lämpligt att personen som roterar farkosten inte har någon elektronisk utrustning på sig såsom en mobiltelefon.

8.4 Steg 4

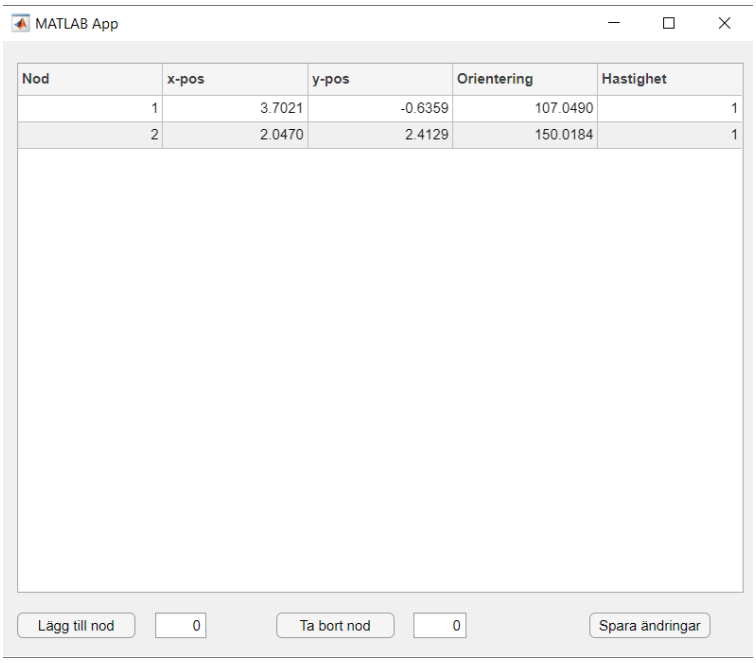
Vill man bara köra manuellt är man klar här, då är det bara att köra med antingen handkontrollen eller med tangentbordet. Egentligen behövs inte kalibrering av magnetometern eller RTK data för att köra manuellt, men det behövs för att få korrekt visualisering i användargränssnittet. Tangentbindningarna är som följande:

- köra framåt/bakåt (1500 rpm): W/S
- svänga vänster/höger (30°): A/D
- bogpropeller vänster/höger (2000 rpm): vänster piltangent/höger piltangent

Vill man köra autonomt längs en bana kan man nu börja placera ut noder genom att klicka på ”Placera noder” och sedan klicka i plot-rutan där man vill ha sina noder. För varje nod man placerar ut får man dessutom välja en riktning för denna som kräver ett extra klick, den svarta linjen som syns i Figur 15 dyker upp när detta sker och indikerar vilken riktning man ger noden. Vill man ändra på, ta bort eller lägga till en nod kan man göra detta genom att trycka på knappen ”Redigera noder”. Då öppnas dialogrutan som ses i Figur 16 och man kan där göra de ändringar man önskar göra sedan trycka på ”Spara ändringar” så uppdateras noderna. För att lägga till eller ta bort en nod behöver man helt enkelt mata in numret på den nod man önskar införa eller ta bort från listan och sedan trycka på knappen ”Lägg till nod” eller ”Ta bort nod”.



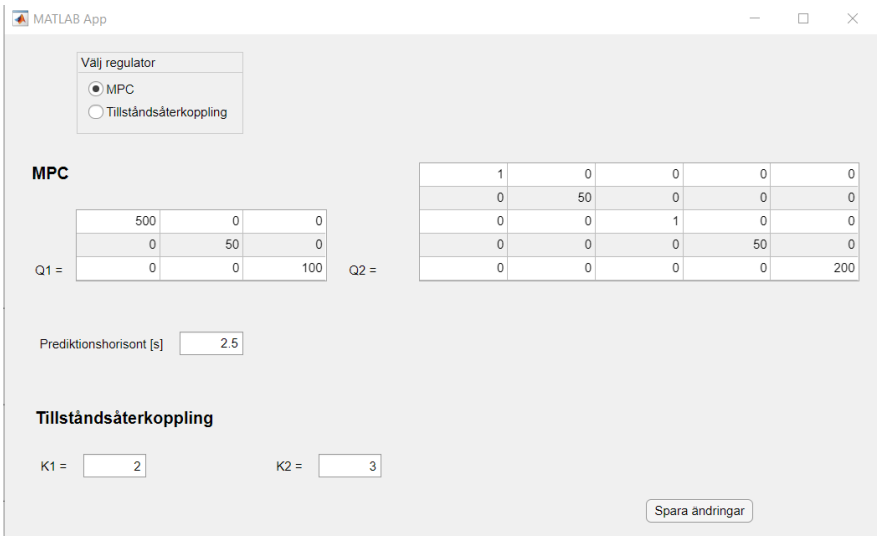
Figur 15: Användargränssnittet när noder placeras ut



Figur 16: Dialogrutan för att redigera noder

8.5 Steg 5

Innan man börjar köra måste man välja vilken sorts regulator man har och dess parametrar, detta görs genom att trycka på knappen ”Regulator” vilket öppnar en dialogruta som kan ses i Figur 17. När man är nöjd med inställningarna klickar man på ”Spara ändringar” så skickas regulatorinställningarna till Raspberry Pi:en. Detta måste även göras om några förändringar görs med färdrutten för att regulatorn ska använda den uppdaterade färdrutten.



Figur 17: Dialogrutan för att välja och justera regulator

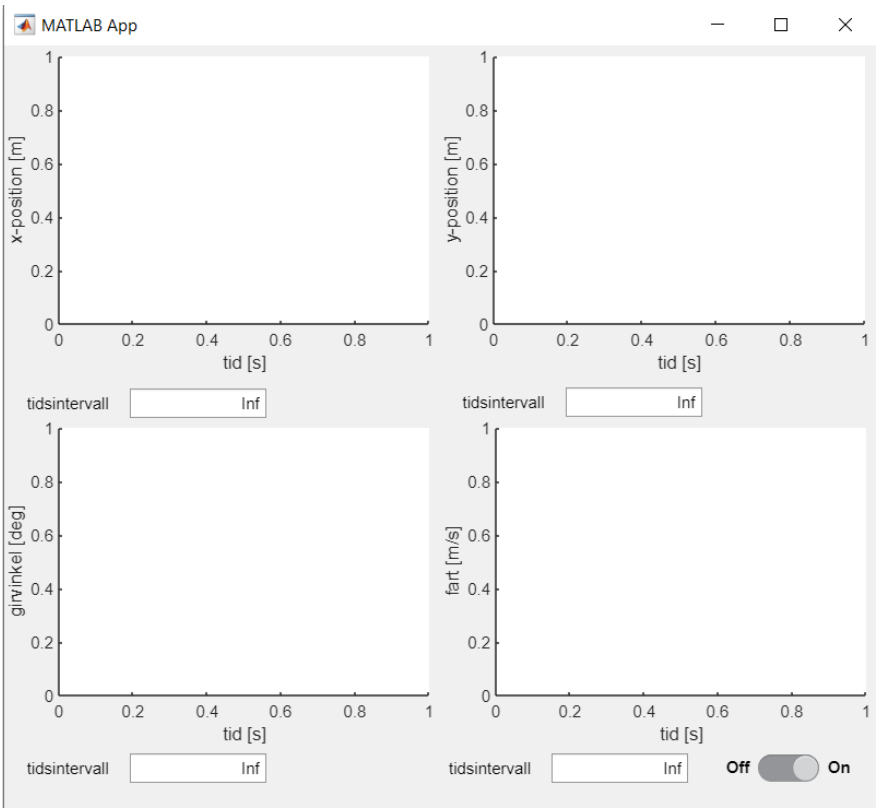
8.6 Steg 6

För att starta en autonom körning behövs nu endast knappen ”Autonom” klickas på så att den blir grön, sedan trycka på knappen ”START”. Knappen ”NÖDSTOPP” stoppar den autonoma körningen samt skickar ett kommando med alla insignaler lika med noll. Denna knapp stannar alltså bara motorerna vilket kan leda till att farkosten fortsätter i den riktning den var på väg, så vill man undvika ett objekt eller strandkant är det lämpligare att manövrera undan med handkontrollen.

8.7 Plotta och logga data

Genom att klicka på knappen ”Plotta data” så öppnas en dialogruta med plottar över de fyra signalerna man kan övervaka, det vill säga: x-position, y-position, orientering och hastighet, se Figur 18. Man kan även ställa in vilket tidsintervall som ska plottas. Värt att notera är att data plottas från och med att man öppnar dialogrutan, inte tidigare.

Vill man logga data så trycker man på knappen ”Logga data”, för att sluta logga data så trycker man igen.



Figur 18: Dialogrutan som visar plottar av data

9 KÄNDA PROBLEM

- Slumpmässig avsaknad av GPS-data.
- Får inte alltid RTK-data vid ett sampel.
- Skalningsmatrisen för magnetometerkalibrering får värde NaN i alla element. När detta sker, upprepa magnetometerkalibreringen tills rimliga värden i matrisens element fås.
- Oregelbundna momentära vinkelutslag och rotationshastigheter för roderpropellerna.
- Vid rotationshastigheter under 1200 rpm på roderpropellerna är mätningarna från tachometrarna inte längre tillförlitliga.
- Användargränssnittet är långsamt från början och blir bara långsammare ju mer som ritas ut på skärmen. Av denna anledning bör det grafiska gränssnittet startas om med jämna mellanrum. En potentiell lösning är även att använda en kraftigare dator.
- Drivremmarna som styr roderpropellerna kan halka vilket orsakar fel styrutslag, detta kan ske vid kraftiga styrutslag eller vid kollision med till exempel sjögräs. För att åtgärda behöver man helt enkelt vrida tillbaka roder-

propellern så många steg som den halkade genom att ställa önskat styrutslag till 0° och sedan vrida tillbaka till 0°.

- Knappen för att logga data i användargränssnittet fungerar endast för att börja logga data, inte sluta. Lösningen på detta problem har varit att helt enkelt logga data genom skriptet `catkin_ws/log_data.sh` istället där man avslutar genom att trycka CTRL + C.
- Kommunikationen mellan GUI:t och båten är i vissa fall känsligt för vilken version av MATLAB som används (i detta projekt användes 2020b).
- Vid autonom körning krävs att båten positioneras efter startnoden innan start för korrekt banföljning.
- Vid autonom körning behöver säkerhetsbrytaren på handkontrollen aktiveras när slutnoden nås. (eftersom regleringen annars försöker återvända till sista punkten på banan)
- Nuvarande implementation av MPC tillåter för höga krafter från bogpropellern. Båten kan i verkligheten inte uppnå den kraft som MPC:n förväntar sig. Detta kan förslagsvis åtgärdas genom att bestraffa, alternativt begränsa, styrsignalen till bogpropellern hårdare.

REFERENSER

[1] G. Zetterqvist och F. Steen, *Modelling and Trajectory Planning for a Small-Scale Surface Ship*, 2021.

[2] *FrSky 2.4GHz ACCESS Taranis X9D Plus 2019/Taranis X9D Plus SE 2019 Manual*, Hämtad: 2021-11-29. URL: https://www.frsky-rc.com/taranis-x9d-plus-2019/.