



Flanders
State of
the Art

16_107_3
FHR reports

ROS-free running system Towing Tank for Manoeuvres in Shallow Water, Ostend

Parameter variations and validation

DEPARTMENT
MOBILITY &
PUBLIC
WORKS

www.flandershydraulicsresearch.be

ROS-free running system, Towing Tank for Manoeuvres in Shallow Water, Ostend

System snapshot 2022/11/14

Van Zwijnsvoorde, T.; Villagómez, J.; Delefortrie, G.

Legal notice

Flanders Hydraulics Research is of the opinion that the information and positions in this report are substantiated by the available data and knowledge at the time of writing.
The positions taken in this report are those of Flanders Hydraulics Research and do not reflect necessarily the opinion of the Government of Flanders or any of its institutions.
Flanders Hydraulics Research nor any person or company acting on behalf of Flanders Hydraulics Research is responsible for any loss or damage arising from the use of the information in this report.

Copyright and citation

© The Government of Flanders, Department of Mobility and Public Works, Flanders Hydraulics Research 2022
D/2022/3241/204

This publication should be cited as follows:

Van Zwijnsvoorde, T.; Villagómez, J.; Delefortrie, G. (2022). ROS-free running system, Towing Tank for Manoeuvres in Shallow Water, Ostend: System snapshot 2022/11/14. Version 3.0. FHR Reports, 16_107_3. Flanders Hydraulics Research: Antwerp

Until the date of release reproduction of and reference to this publication is prohibited except in case explicit and written permission is given by the customer or by Flanders Hydraulics Research. Acknowledging the source correctly is always mandatory.

Document identification

Customer:	Flanders Hydraulics Research	Ref.:	WL2022R16_107_3
Keywords (3-5):	ROS, free-running, FML		
Knowledge domains:	Harbours and waterways		
Text (p.):	24	Appendices (p.):	2
Confidential:	☒ Yes	Released as from:	01/01/2025

Author(s):	Van Zwijnsvoorde, T.
------------	----------------------

Control

	Name	Signature
Reviser (internal):	Villagómez, J.	Getekend door: Jose Villagomez Rosales Getekend op: 2023-01-06 11:36:26 +01:0 Reden: Ik keur dit document goed <i>Jose C. Villagomez</i>
Reviser(external):	Delefortrie, G.	Getekend door: Guillaume Delefortrie (Sig) Getekend op: 2023-01-06 13:23:41 +01:0 Reden: Ik keur dit document goed <i>Guillaume Delefortrie</i>
Project leader:	Wolput, B.	Getekend door: Ben Wolput (Signature) Getekend op: 2023-01-11 12:41:40 +01:0 Reden: Ik keur dit document goed <i>Ben Wolput</i>

Approval

Head of Division:	Bellafkih, K.	Getekend door: Abdelkarim Bellafkih (Sig) Getekend op: 2023-01-06 11:57:53 +01:0 Reden: Ik keur dit document goed <i>Abdelkarim Bellafkih</i>
-------------------	---------------	--



Abstract

The Towing Tank for Manoeuvres in Shallow water has been completed in 2020. Awaiting the completion of the towing carriage, a stand-alone state-of-the-art free running system has been developed. This tender was won by Kapernikov, an IT specialist who developed the software and hardware. The free running system is capable of knowing the position of the ship in the tank, based on lidar, IMU and camera technology. The position feedback is given as input to control algorithms, which return rudder and propeller setpoints. The system as a whole runs as a ROS – Robot Operating System – environment. This report gives an overview of the ‘as-in’ at the end of 2022, where not all work has been delivered by Kapernikov. However, a demo-version is available, which is shown to potential clients and project partners and fully described in this report. As the system is a work-in-progress, with possible extensions to the current project scope, learning points and possible actions are included in this document.

Contents

Abstract	III
Contents	IV
List of figures	VI
1 Underlying documentation	1
2 Introduction.....	2
3 System description	3
3.1 Ship control.....	3
3.2 Position measurement	4
4 Hardware and software equipment	5
4.1 Ship hardware.....	5
4.2 Towing tank hardware.....	6
4.3 Software	7
4.3.1 Kapernikov (ROS).....	7
4.3.2 FH (C#)	8
4.4 Status hardware and software integration	9
5 System set-up and operation	10
5.1 Kapernikov Azure structure.....	10
5.2 Deployment flow	11
5.2.1 Wall PC (PCROS2).....	11
5.2.2 Ship	12
5.3 Test set-up for client demonstration.....	16
5.4 Stop a test / recording	18
5.5 Restarting test	18
5.6 End of testing day	19
5.7 Download recording from IPC to ROSPC	19
5.8 IPC memory	19
6 ShaITT - WLTN – VO communication.....	20
7 Key dates, lessons learned, action points, memo's.....	21
7.1 Memo's.....	21
7.1.1 M16_107_15.....	21
7.1.2 M16_107_17.....	21
7.1.3 M16_107_19.....	21

7.1.4	M16_107_22.....	21
7.1.5	M16_107_23.....	21
7.1.6	M16_107_24.....	21
7.1.7	M16_107_25.....	21
7.2	Key dates	22
7.3	Lessons learned	22
7.4	Action points Kapernikov.....	23
7.5	Action points FH	23
Appendix 1 : M16_107_24		A1

List of figures

Figure 1 : Description of free-running system (ROS environment) - high quality image can be found on Kapernikov Azure, here	3
Figure 2 : Ship hardware – KVLCC2 model – as equipped on 2022/11/14 Top : Equipped KVLCC2 top view Bottom : left to right IPC, camera, IMU, NAV350.....	5
Figure 3 : Detail Aruco-markers (camera integration)	6
Figure 4 : Detail reflectors tube (NAV350)	6
Figure 5 : Measured positions from turning circle test, mail Hongwei He 07 september 2022. ; Dx,Dy, u,v, Dpsi, {EKF} output ROS system (Kalman-sensor-fusion-filter) {Cal} : differentiation of positions by Hongwei He. Graphs copied from excel sheet 'TurningCircleSezp7_SHIP_42_position.xlsx', mail of 2022/09/14.	7
Figure 6 : Content of experiment folder 'EXP33CsharpSim', located op shore PC (ROSPC).	8
Figure 7 : Content of subfolder 'behaviour0' of 'EXP33CsharpSim', located on shore PC (ROSPC).....	8
Figure 8 : Navigation on Kapernikov sleeptank azure page – accessing the wiki page.....	10
Figure 9 : Left - Command window showing the set-up of the remote control system W1 Right – Command window showing uploading of behaviours W2	12
Figure 10 : Command windows showing behaviours when starting the system W3,W4,W5	13
Figure 11 : Command windows showing behaviours after completing full set-up of experiment (after completing of W7) W3,W4,W5 Data shown in window is written from C# behaviour and can thus be different then the one shown on this example.	13
Figure 12 : Command windows for activating ship and making conneciton with the rosboard (W6,W7)	14
Figure 13 : Appareance of the rosboard after sucessfully completed the deployment flow.....	15
Figure 14 : Planned path and followed track following EXP33CsharpSim behaviours and config files (figure made by Hongwei He based on output data test).	16
Figure 15 : Unstable positioning system when starting the experiment (example).	16

1 Underlying documentation

Bestek en offerte

[1] *“Ontwikkeling van een meetsysteem voor de bepaling van de positie van vrijvarende scheepsmodellen voor de sleeptank van het Maritiem Onderzoekscentrum”*, Bestek nr. FH_2020_01

[2] *“Maritiem Onderzoekscentrum – Raming budget voorwaardelijke gedeelte meetsysteem vrijvarende scheepsmodellen – volgens bestek FH 2020_01”*

Rapporten Kapernikov

[3a] *“FH_2020_01 Positiebepaling vrijvarende scheepsmodellen”*; Verhagen, B., Langsweirdt, P. ; 2021/04/28 (hoofdrapport)

[3b] *“2021-04-28-FH-Proeftypes.pdf”*, bijlage bij 3a

[3c] *“2021-04-28-FH_2020_01-Meetprincipes.pdf”*, bijlage bij 3a

[3d] *“2021-04-28-FH_2020_01-Cameraseselectie.pdf”*, bijlage bij 3a

Sites

Kapernikov Teams site

Kapernikov Azure site <https://dev.azure.com/Kapernikov/Sleeptank>

2 Introduction

Kapernikov (IT-consultant) is developing a free-running system for FH – Flanders Hydraulics. This system is developed at FML – Flanders Maritime Laboratory –, in the Towing Tank for Manoeuvres in Shallow Water ShaTT. The technology, which works with open-source ROS (Robot Operating System) software, is generally applicable for any water basin where free-running tests want to be conducted, given that the necessary equipment (basin and ship) is present.

The system can be described as :

- A system which can be operated as a standalone system. No towing carriage is needed to run the system.
- A system which is capable of controlling a ship (rudder/propeller), measuring positions accurately and making the output available for post-processing.
- A system which has the potential to be used to control different ships in the same physical basin, interacting with each other.

The system has no official working title although it has been submitted to the Blue Innovation Award 2022 contest under the name of ASHIP – Autonomous Ship Innovation Platform. In practice, terms like ‘free running system’, ‘ROS system’ can be used, which is also done throughout this report for readability. Note that it is possible that the towing carriage, which is going through SAT – Site Acceptance Tests – as we speak, could perform free running tests. Throughout this report, the free running system is always exclusively the ROS based free running system, not a system which is part of the towing carriage.

At the moment of finalising this report, the system is already operated by FH, through a deployment flow which was made available by Kapernikov. There are however a few issues (interface, sensor fusion details) which should still be addressed. A follow-up report will be produced including these parts. For now, the discussion is held based on the ROS-system status as per 2022/11/14.

This report features the following chapters

Chapter 2	Introduction
Chapter 3	System description
Chapter 4	Hardware and software equipment
Chapter 5	System set-up and operation
Chapter 6	ShaTT – WLTN – VO communication
Chapter 7	key dates, lessons learned, action points, memo’s

3 System description

The two main functionalities which were required were already listed in the introduction : a system that can measure the ship's position and which allows us to control the ship (rudder, propeller), based on the measured position. Figure 1 shows the architecture of the system. The figure is available on the Kapernikov [Azure site](#), for which Kapernikov needs to give individual permission to access the site.

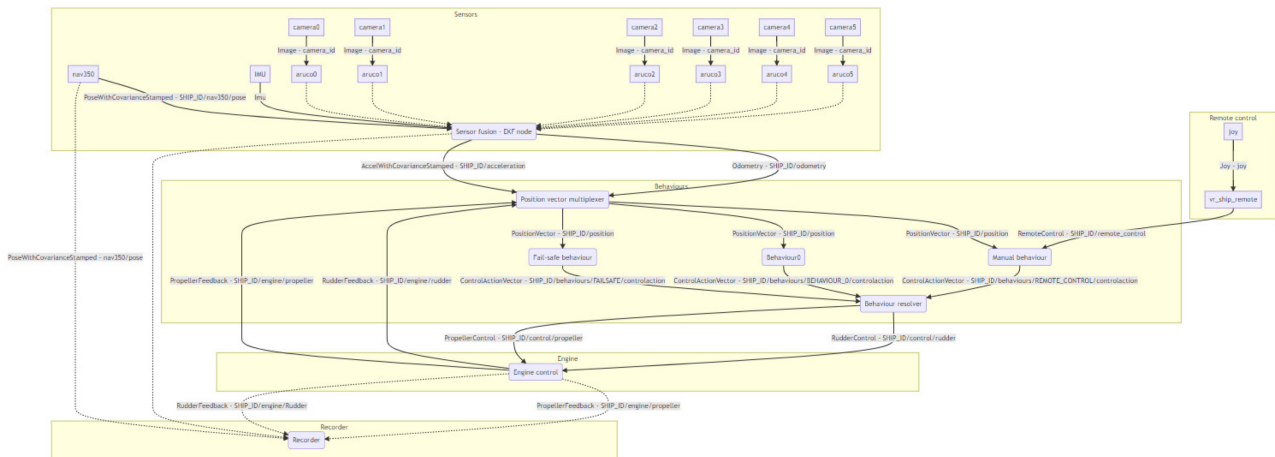


Figure 1 : Description of free-running system (ROS environment) - high quality image can be found on Kapernikov Azure, here

3.1 Ship control

As was already shown in Figure 1, ROS is capable of integrating all different components of the system and allows to transfer data (ROS-packages) between the different nodes of the system. In FH's current daily operation in the Antwerp facility, Towing Tank for Manoeuvres in Confined Water (ConfTT), where the carriage is used to conduct free running tests, C# based ship control algorithms are used. These require as input ship position (and velocity, acceleration) and time and deliver as output setpoints for rudder and propeller. Kapernikov has written an interface which allows C# to interface with ROS, allowing FH to use the same coding language (C#) in both towing tank facilities. Following remarks are made :

- At ConfTT, the free-running system is operated in symbiosis with the towing carriage. The carriage can accelerate the ship (captive), but also grab the ship after the test (captive deceleration). As the ROS system is built as stand-alone, this captive acceleration and deceleration is not part of the system's description. This means that some parts of the C#-code used in ConfTT, which express links and conditions for the carriage to release – catch the ship, are removed from the code for ShalTT ROS system.
- Despite having a similar structure (input-output), the existing C# routines from ConfTT needed to be rewritten to work well with the ROS interface as developed by Kapernikov. More details can be found in section 4.3.2.

3.2 Position measurement

Several methods exist to measure the position (velocity, acceleration) of the ship. Kapernikov did a preliminary desktop study [3a,b,c,d] where they identified the different technologies which can be used for the current application. Important here is that the most well-known technique to measure positions – GPS – cannot be used as the GPS signals are blocked by the concrete infrastructure.

Based on the study [3a,b,c,d], three different systems were selected

- *NAV350*, a laser scanner (lidar)
- *IMU*, Inertial Measurement unit
- *Cameras*

All three systems (hardware) have been purchased by FH according to the specifications from [3a,b,c,d] and have been installed on the ship (hardware). For the NAV350 and camera systems, hardware needed to be installed on the towing tank side walls. As per 2022/11/14, only the IMU and NAV350 are actively used to generate position and time measurements, they are the main focus of this report. For the camera system, there is referred to [3c,d] for more background. The hardware components for the camera system are described in this document as they are part of the ship and towing tank outfitting as per 2022/11/14

4 Hardware and software equipment

4.1 Ship hardware

Specifically for the ROS free-running system as developed by Kapernikov, the following equipment has been installed, in line with the documentation found in [3a,b,c,d], see Figure 2.

- *NAV350*, a laser scanner. One unit, installed at midship. Sufficiently high above the keel in order to have vision of the reflector units installed at the tank wall.
- *IMU*, Inertial Measurement unit. Installed close to the centre of mass of the ship (x, y), at the keel.
- *Cameras*, six cameras. One looking head (0°), one looking behind (180°), four looking laterally (70°, 110°, 250°, 290°).

The following hardware has been added to the ship's equipment, in order to successfully perform the free running tests.

- Rudder and propeller, including *CAN-bus* (Controller Area Network) have been installed
- Four batteries (2 times 2 batteries in series 12V -> 24V). Choice to install this amount of battery power (more than 48 h continuous operation when fully charged) was made as the ship needed to be ballasted after all, so the batteries are just part of the weight.
- An IPC (Industrial PC)
- Two water evacuating pumps, needed to evacuate the water which inevitably enters the ship through the shaft opening
- Fuses (rudder, propeller, pumps, other)
- Several ethernet connections
- Wifi receiver and sender station

On 2022/11/22, Bart van den Broeck measured the position of the equipment in the ship model. He will make a 3D cad model out of his measurements.

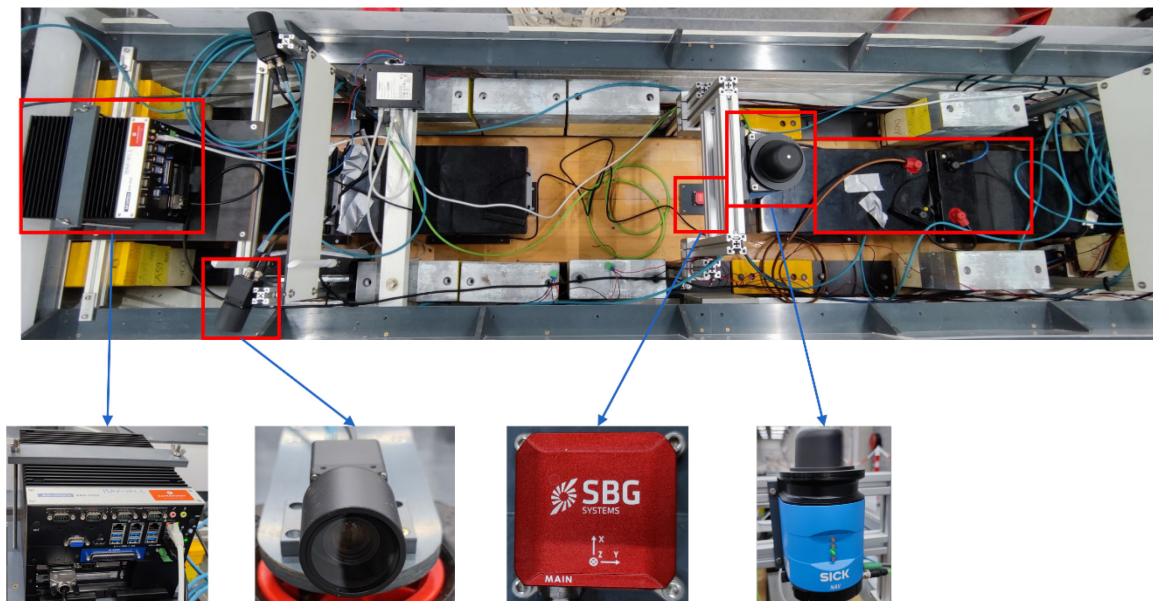


Figure 2 : Ship hardware – KVLCC2 model – as equipped on 2022/11/14

Top : Equipped KVLCC2 top view

Bottom : left to right IPC, camera, IMU, NAV350

4.2 Towing tank hardware

Two of the three position measurement devices (NAV350 and cameras) require additional hardware to be installed on the towing tank wall. Details can be found in [3 a, c].

For the camera system, Aruco markers are needed to detect the position of the ship (Figure 3). The size and shape of the markers have been developed in function of the camera specs and the tank size. The use of small and large markers is done in order to have sufficiently sharp images from close by and far away in all positions. Kapernikov initially made a plan of Aruco placement [3a,c]. During the course of 2022, the Aruco markers needed to be removed and put back several times. As they are positioned now, this is not according to Kapernikov's plan as attached to [3a,c] . FH has sent the new list of Aruco markers to Kapernikov, including their measured position. FH will develop an updated plan with Aruco marker positions.

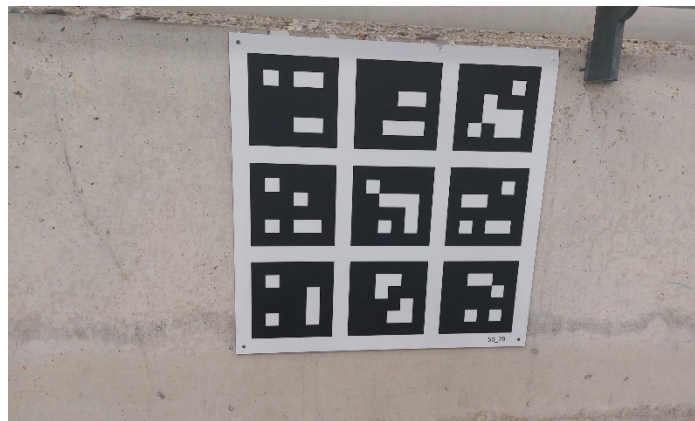


Figure 3 : Detail Aruco-markers (camera integration)

The reflector units are installed to reflect the NAV350 laser beams back to the unit (Figure 4). The round shape and the position (not at the wall, but a few centimetres of the wall) are beneficial for (unique) detection of the unit. The distance between the individual reflectors varies, in order to have a unique position detection. If equidistant reflectors would be used, the ship would not have a unique position, based on the feedback from the NAV350 system.



Figure 4 : Detail reflectors tube (NAV350)

4.3 Software

4.3.1 Kapernikov (ROS)

The ROS environment allows several nodes to communicate with each other by sending containers, which can be logged and saved for post-processing (conversion to .csv files). ROS is one of several systems which could have been used, others include Moos-IVP, which has ship controls as part of the framework. ROS however is a very flexible system structure, with many prebuild algorithms for signal processing. It also allows for flexible interaction with FH-UGent complex control algorithms (C# dll's).

One of the signal processing functions, the main one concerning Kapernikov's work, is the *Kalman-sensor-fusion-filter*, which allows the combine the measurements from the individual hardware components (NAV350, IMU, camera) into an accurate and stable (time stamp) output.

As per 2022/11/14, NAV350 and IMU signals are fused, using the *Kalman-sensor-fusion-filter*. A result of a free running tests where fused signals are shown, is given by Figure 5. The ship performed a turning circle, where u and v should be constant (by definition). The figure shows velocities given in the ROS output Rosbag, tag "EKF". The tag "Cal" shows the values calculated by Hongwei He based on position and time step. The results show that the calculated u and v are constant (when applying running average on the data to get rid of the noise). The EKF results however show that u and v are not constant (also not when taking running average), meaning that the EKF signals are not accurate. Kapernikov already disclosed that it is not uncommon, as the *Kalman-filter* often needs tuning. The data (Rosbag) behind the graphs from Figure 5 has been shared with Kapernikov on 2022/10/28 in order to examine if the filter can be further tuned to enhance the quality of the measurements.

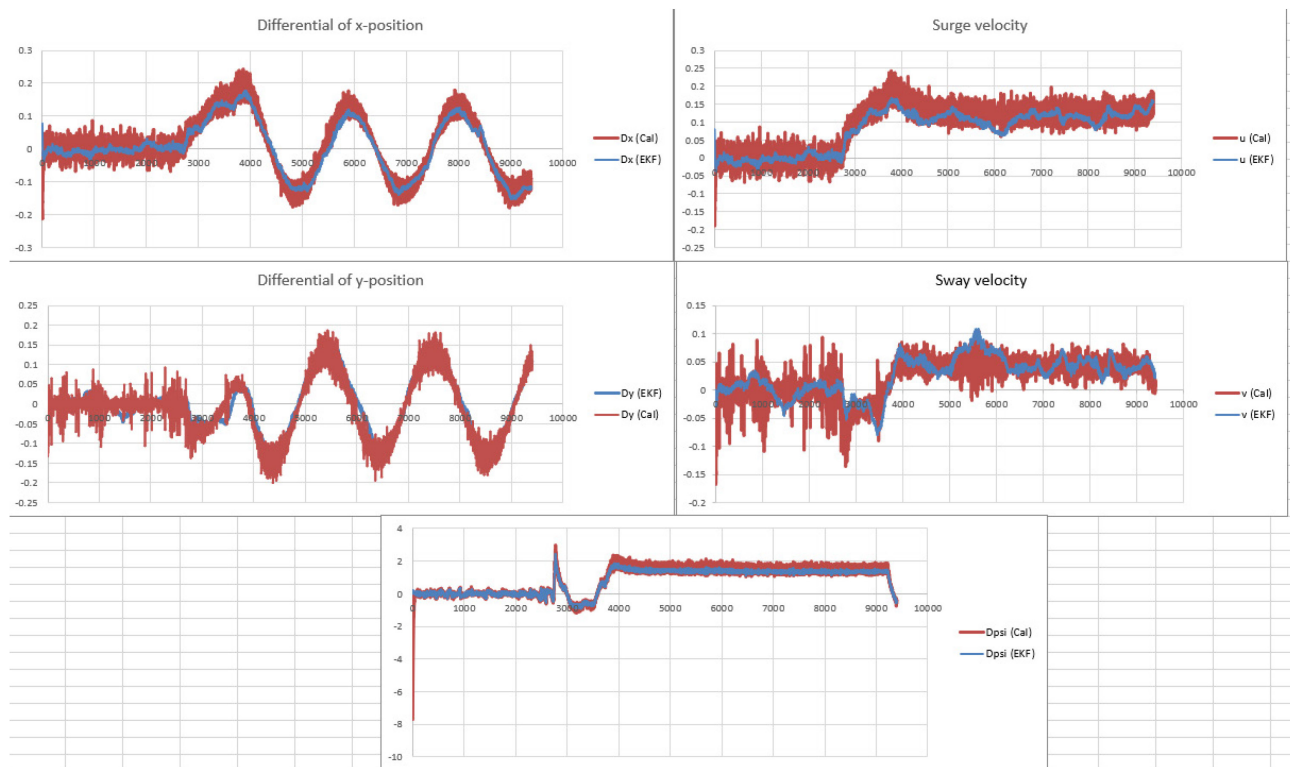


Figure 5 : Measured positions from turning circle test, mail Hongwei He 07 september 2022.
; Dx,Dy, u,v, Dpsi, {EKF} output ROS system (Kalman-sensor-fusion-filter) {Cal} : differentiation of positions by Hongwei He.
Graphs copied from excel sheet 'TurningCircleSezp7_SHIP_42_position.xlsx', mail of 2022/09/14.

4.3.2 FH (C#)

All ship control algorithms are loaded in via predefined structures, where three different control algorithms are defined for each experiment. They are called “behaviour0”, “behaviour1” and “behaviour2”. As implemented now, these behaviours are a start behaviour (acceleration in straight line), a manoeuvre (path following, zigzag) and a crash-stop (hard astern, zero rudder angle) command.

Each behaviour comes with a separate folder, both on ROSPC (shore) and IPC (ship) where the control algorithm is stored. An example of how such folder structure looks is given in Figure 6 and Figure 7, for the experiment which is used for now as demo for visits (see section 5.3 for further details on this experiment). Figure 6 shows the three behaviour folders. The fourth folder “configs” is where the set-up files for the different behaviours are stored (they also need to be loaded to the ship IPC). Figure 7 shows a detail of the content of “behaviour0”. The dll **Behaviour.dll** is the main executable with this exact name (all three folders contain the main executable **Behaviour.dll** (so no addition of 0,1,2 here!)). This main dll can refer to several sub-dlls, where there is no longer a naming convention. The use of SOWXML.dll and Sleptank.Navigation.Controllers.dll, Sleptank.Navigation.Behaviour.Interfaces.dll follows the definitions which are used for the free running system in ConfTT (Antwerp).

 behaviour0	16/11/2022 8:48	Bestandsmap
 behaviour1	16/11/2022 8:48	Bestandsmap
 behaviour2	16/11/2022 8:48	Bestandsmap
 configs	16/11/2022 8:48	Bestandsmap

Figure 6 : Content of experiment folder ‘EXP33CsharpSim’, located op shore PC (ROSPC).

 Behaviour.dll	25/10/2022 15:24	Toepassingsuitbrei...	13 kB
 MathNet.Numerics.dll	25/10/2022 15:24	Toepassingsuitbrei...	1,650 kB
 Sleptank.Navigation.Behaviour.Interface...	25/10/2022 15:24	Toepassingsuitbrei...	5 kB
 Sleptank.Navigation.Behaviour.Interface...	25/10/2022 15:24	Program Debug D...	8 kB
 Sleptank.Navigation.Controllers.dll	25/10/2022 15:24	Toepassingsuitbrei...	22 kB
 Sleptank.Navigation.Controllers.pdb	25/10/2022 15:24	Program Debug D...	13 kB
 Sleptank.Navigation.StartBehaviour.deps	25/10/2022 15:24	JSON File	3 kB
 Sleptank.Navigation.StartBehaviour.pdb	25/10/2022 15:24	Program Debug D...	11 kB
 SOWXML.dll	25/10/2022 15:24	Toepassingsuitbrei...	44 kB
 SOWXML.pdb	25/10/2022 15:24	Program Debug D...	30 kB

Figure 7 : Content of subfolder ‘behaviour0’ of ‘EXP33CsharpSim’, located on shore PC (ROSPC).

When it comes to making dll’s using Visual Studio, you user can choose different build versions. The ‘.NET Standard version (2.0 or 6.0)’ should be chosen as build version.¹

¹ This has in fact been up for discussion many times, where other build versions seem to work as well, also with respect to the discussion dll vs sub-dll. Guillaume Delefortrie (then FH now UGent) initially set up all the behaviours, which Kapernikov deployed. Hongwei He (UGent) has since the initial deployment further developed the C# algorithms. It is an action point for FH to list and store these scripts as well as to check with Kapernikov what the build version requirements are.

Below some quick notes on how to build a dll on your own machine, starting from an existing dll. The goal was to get the start behaviour running, but under a new project defined as 'Thibautbehaviour'.

- (1) Create new project -> class library -> "Sleeptank.Navigation.ThibautBehaviour"
- (2) Add "public class ThibautBehaviour : INavigationBehaviour"
- (3) Load project Sleeptank.Navigation.Behaviour.Interfaces into VisualBasic -> Build dll
- (4) Rightclick on Sleeptank.Navigation.ThibautBehaviour and add "project_reference" -> Sleeptank.Navigation.Behaviour.Interfaces
- (5) Add "using Sleeptank.Navigation.Behaviour.Interfaces;" on top of class
- (6) Now you will still get errors, but that is normal, as the predefined 'Sleeptank.Navigation.Behaviour.Interfaces' needs two methods to be defined.
- (7) Now copy the content of StartBehaviour class into ThibautBehaviour.cs
- (8) Now two errors are remaining, CourseFollower and AutoBestand, which also needed to be added to the project
- (9) When trying to build AutoBestand you need SOWXML. If you add SOWXML reference to project, you also have AutoBestand
- (10) Build dll from Sleeptank.Navigation.Controllers and add project reference + using statement on top of the page

More information on use of dll's at the ConfTT, in communication with the towing carriage (ATOS) can be found in the following reports

[Delefortrie, G.; Eloot, K.; Mostaert, F. \(2018\). Ship Manoeuvring Models: Free running tests with the KCS. Version 3.0. FHR Reports, 16 023 8. Flanders Hydraulics Research: Antwerp.](#)

[Delefortrie, G.; Eloot, K.; Mostaert, F. \(2020\). Ship Manoeuvring Behaviour: Sub report 13 – Execution of free running tests with the Q-flex. Version 2.0. FHR Reports, 16 023 13. Flanders Hydraulics Research: Antwerp.](#)

It is recommended to update these reports considering the latest implementation of the free running algorithms in the ROS framework of the ShaTT.

4.4 Status hardware and software integration

This section summarizes the status of the software and hardware integration, as per 2022/11/14

- NAV350, IMU and cameras are all installed and working.
- Sensor fusion (Kalman filter) is active and fuses NAV350 and IMU data to give positions and velocities (x, y, dx, dy, psi, r) at a target frequency of 8Hz. This frequency is not yet (stable) at 8 Hz.
- Vertical position (z, dz, trim) are not given in position vector, as cameras need to be active to accomplish this.

In the coming year (2023), working on and testing the accuracy of the system will be a major focus point. In the current scope, it is foreseen that Kapernikov will update the Kalman filter parameters on-site in January 2023.

5 System set-up and operation

In the current status of the system, the system is operated via the *deployment flow*. This name is consistent with the Kapernikov naming which can be found on the Azure site

<https://dev.azure.com/Kapernikov/Sleeptank>

From FH/UGent, the following persons have admission to the Azure website : Thibaut Van Zwijnsvoorde, Stefan Geerts, Hongwei He. When using the PCROS, the azure page is set as bookmark and accessible (already logged in), so you do not need any additional admission to use the system. Bear in mind that the git-repository as well as all most recent documentation is stored on the Azure site.

The azure site in fact shows all components / developments / plans for the free running project (software & hardware developments). For this report, the focus is on helping the FH/UGent user to navigate through the deployment flow as it is present on 2022/11/16. Kapernikov in fact committed to delivering a GUI to start / stop / log tests. This GUI is not available at this time.

All steps indicated here are performed on the Linux configured ROS-PC (PCROS2) on the side of the towing tank. Login for the PC is MOC4ever.

5.1 Kapernikov Azure structure

As said, the azure website contains much information, almost too much for the modal user. It also serves as Github-ish version control system. For working with the system, you need to navigate to the wiki (Figure 8).

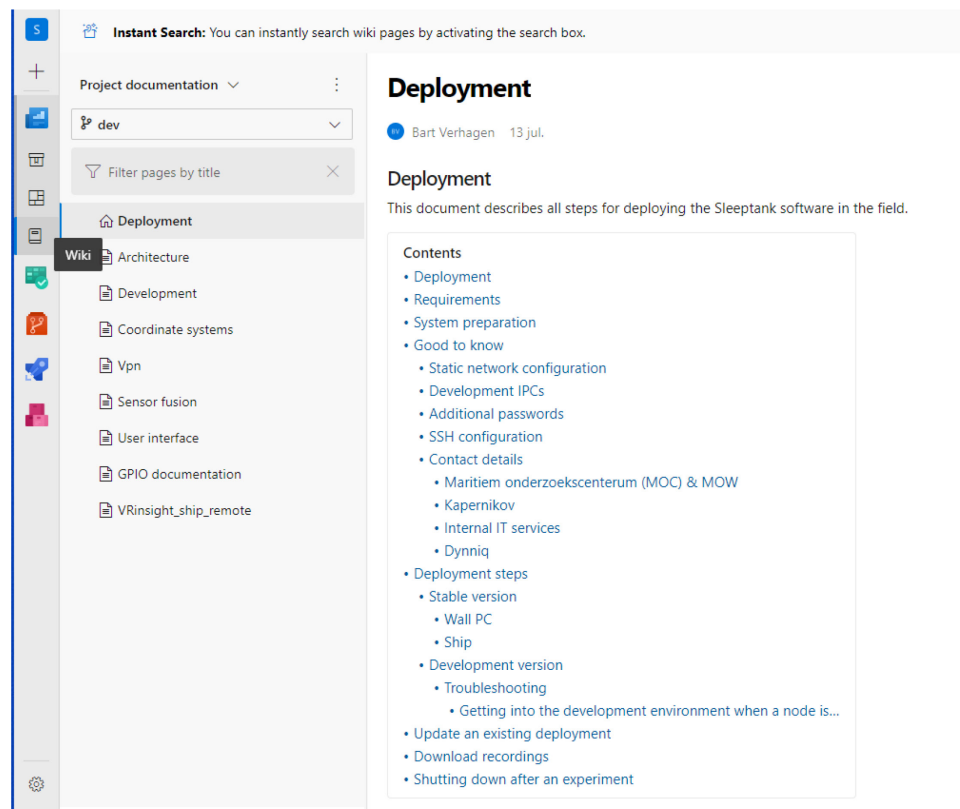


Figure 8 : Navigation on Kapernikov sleeptank azure page – accessing the wiki page.

5.2 Deployment flow

In order to get the system running, you need to follow the *deployment flow*, which is the first item in the wiki page. As the wiki is also version controlled, you should make sure you follow the master-version. In Figure 8 it is set to 'dev' which will not give you the latest stable version. Always check you are in the master branch. The *deployment flow* starts with the following steps

- System requirement
- System preparation
- Good to know
- Development IPC's
- Additional passwords
- SSH configuration
- Contact details

You should not need to do/use/know any of this information to successfully use the system. When doing troubleshooting, some of the information provided in these sections might be useful, for example the IP addresses of the components if one wants to check if the network is responding.

5.2.1 Wall PC (PCROS2)

In the step wall PC, the remote control unit connected to the wall pc is activated. Open a new command window (Ctrl + T) and paste following command (and press enter). This command window is named here **W1**, as there will be more command windows opened to set up the system. Type (or copy/paste) the following line (hotkey to cope using Linux is CTRL+Shift+C):

```
docker login sleeptank.azurecr.io
```

In case the login does not succeed, it is possible that the system got reset and you need to give again the password :

```
docker login --username=mow-infra --password="<docker_registry_password>"  
sleeptank.azurecr.io
```

The password is then found under Additional passwords. The indication <> is used if you need to manually complete the statement, first delete all text within brackets (and including the brackets) and replace with what is requested. In this case, it will be the password c8sbhLrCKIUQiwVJ/WLM3HJ2a1zOsFkZ

Then paste following comment

```
docker pull sleeptank.azurecr.io/runtime
```

then paste

```
docker run "--name=remote_control" --interactive=true --tty=true --privileged --rm --  
net=host "--env=USERNAME=root" "--env=UUID=0" "--env=UGID=0" "--volume=/dev:/dev"  
sleeptank.azurecr.io/runtime ros2 launch sleeptank remote_control
```

The remote control unit should now be successfully activated (Figure 9).

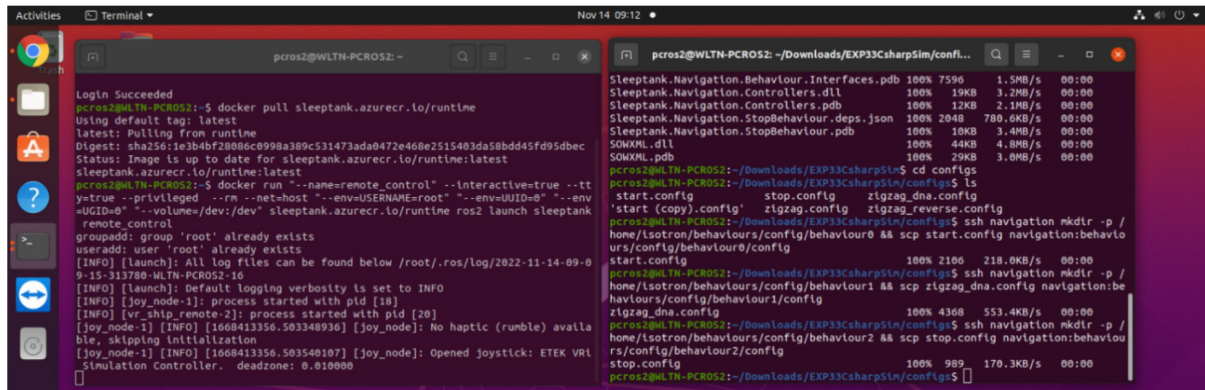


Figure 9 : Left - Command window showing the set-up of the remote control system **W1**
Right – Command window showing uploading of behaviours **W2**

5.2.2 Ship

In the current application, the control system can handle three behaviours (see section 4.3.2). It also needs three (working) behaviours. In case one of the behaviours would crash, the system will no longer respond and the ship will carry on performing the last rudder and propeller command.

In a first step, you need to upload the behaviour from a folder on the PCROS to the IPC on the ship (that is what `ssh navigation` does, connect and work on ship IPC). You can upload all behaviours using one command window. **W2** (Figure 9).

```
cd <path to folder containing the behaviour>
```

```
ssh navigation mkdir -p /home/isotron/behaviours/dll/<behaviour> && scp -r <path to
folder containing the behaviour>/* navigation:behaviours/dll/<behaviour>/
```

Each behaviour comes with a configuration file, in short “config file”, which gives the values for the parameters which are used in the behaviour. These are uploaded next, using the same command window

```
cd <path to the behaviour config>
```

```
ssh navigation mkdir -p /home/isotron/behaviours/config/<behaviour> && scp <config
file> navigation:behaviours/config/<behaviour>/config
```

You should use the same naming convention which is given in the deployment flow, *behaviour0*, *behaviour1*, *behaviour2* for the three behaviours you upload.

The name of the config files <config file> can be chosen by the user, in section 5.3, an example is given of how these <generic statements> need to be replaced for a concrete experiment.

Next up, all behaviours are activated by pasting the command below in three different command windows. It is also advised to use another desktop page to put these three windows. **W3**, **W4**, **W5** (Figure 10). As per 2022/11/14, there is still an error in the deployment flow. The word ‘moc’ needs to be added as indicated below to the command line as given in the deployment flow (and only in this step).

```
ssh -t navigation docker run --name=<behaviour> --interactive=true --tty=true --
privileged --rm --publish=<port>:80 "--
volume=/home/isotron/behaviours/dll/<behaviour>:/Behaviour"
mocsleep tank.azurecr.io/behaviour_dotnet
```

The <port> is used to allocate space in ROS-framework to each behaviour for communication of data. After activating the system (**W7**), the respective command windows will show the output (*print* statement) of the C# algorithms. An example of such data output flow is shown in Figure 11.

```

pcros2@WLTN-PCROS2:~$ ssh -t navigation docker run --name=behaviour0 --interact
ive=true --tty=true --privileged --rm --publish=<port>:80 --volume=/home/isotro
n/behaviours/dll/behaviour0/Behaviour" sleep tank.azurecr.io/behaviour_dotnet
bash: behaviour0: No such file or directory
pcros2@WLTN-PCROS2:~$ ssh -t navigation docker run --name=behaviour0 --interact
ive=true --tty=true --privileged --rm --publish=7000:80 --volume=/home/isotro
n/behaviours/dll/behaviour0/Behaviour" mcsleep tank.azurecr.io/behaviour_dotnet
bash: behaviour0: No such file or directory
Connection to 10.202.18.1 closed.
pcros2@WLTN-PCROS2:~$ ssh -t navigation docker run --name=behaviour0 --interact
ive=true --tty=true --privileged --rm --publish=7000:80 --volume=/home/isotro
n/behaviours/dll/behaviour0/Behaviour" mcsleep tank.azurecr.io/behaviour_dotnet
INFO: Microsoft.Hosting.Lifetime[14]
Now listening on: http://[::]:80
INFO: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
INFO: Microsoft.Hosting.Lifetime[0]
Hosting environment: Production
INFO: Microsoft.Hosting.Lifetime[0]
Content root path: /App/

pcros2@WLTN-PCROS2:~$ ssh -t navigation docker run --name=behaviour1 --interact
ive=true --tty=true --privileged --rm --publish=<port>:80 --volume=/home/isotro
n/behaviours/dll/behaviour1/Behaviour" sleep tank.azurecr.io/behaviour_dotnet
bash: behaviour1: No such file or directory
pcros2@WLTN-PCROS2:~$ ssh -t navigation docker run --name=behaviour1 --interact
ive=true --tty=true --privileged --rm --publish=8000:80 --volume=/home/isotro
n/behaviours/dll/behaviour1/Behaviour" mcsleep tank.azurecr.io/behaviour_dotnet
INFO: Microsoft.Hosting.Lifetime[14]
Now listening on: http://[::]:80
INFO: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
INFO: Microsoft.Hosting.Lifetime[0]
Hosting environment: Production
INFO: Microsoft.Hosting.Lifetime[0]
Content root path: /App/

pcros2@WLTN-PCROS2:~$ ssh -t navigation docker run --name=behaviour2 --interact
ive=true --tty=true --privileged --rm --publish=<port>:80 --volume=/home/isotro
n/behaviours/dll/behaviour2/Behaviour" sleep tank.azurecr.io/behaviour_dotnet
INFO: Microsoft.Hosting.Lifetime[14]
Now listening on: http://[::]:80
INFO: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
INFO: Microsoft.Hosting.Lifetime[0]
Hosting environment: Production
INFO: Microsoft.Hosting.Lifetime[0]
Content root path: /App/

```

Figure 10 : Command windows showing behaviours when starting the system **W3,W4,W5**

```

pcros2@WLTN-PCROS2:~$ ssh -t navigation docker run --name=behaviour0 --interact
ive=true --tty=true --privileged --rm --publish=<port>:80 --volume=/home/isotro
n/behaviours/dll/behaviour0/Behaviour" sleep tank.azurecr.io/behaviour_dotnet
bash: behaviour0: No such file or directory
pcros2@WLTN-PCROS2:~$ ssh -t navigation docker run --name=behaviour0 --interact
ive=true --tty=true --privileged --rm --publish=7000:80 --volume=/home/isotro
n/behaviours/dll/behaviour0/Behaviour" mcsleep tank.azurecr.io/behaviour_dotnet
INFO: Microsoft.Hosting.Lifetime[14]
Now listening on: http://[::]:80
INFO: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
INFO: Microsoft.Hosting.Lifetime[0]
Hosting environment: Production
INFO: Microsoft.Hosting.Lifetime[0]
Content root path: /App/

pcros2@WLTN-PCROS2:~$ ssh -t navigation docker run --name=behaviour1 --interact
ive=true --tty=true --privileged --rm --publish=<port>:80 --volume=/home/isotro
n/behaviours/dll/behaviour1/Behaviour" sleep tank.azurecr.io/behaviour_dotnet
bash: behaviour1: No such file or directory
pcros2@WLTN-PCROS2:~$ ssh -t navigation docker run --name=behaviour1 --interact
ive=true --tty=true --privileged --rm --publish=8000:80 --volume=/home/isotro
n/behaviours/dll/behaviour1/Behaviour" mcsleep tank.azurecr.io/behaviour_dotnet
INFO: Microsoft.Hosting.Lifetime[14]
Now listening on: http://[::]:80
INFO: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
INFO: Microsoft.Hosting.Lifetime[0]
Hosting environment: Production
INFO: Microsoft.Hosting.Lifetime[0]
Content root path: /App/

pcros2@WLTN-PCROS2:~$ ssh -t navigation docker run --name=behaviour2 --interact
ive=true --tty=true --privileged --rm --publish=<port>:80 --volume=/home/isotro
n/behaviours/dll/behaviour2/Behaviour" sleep tank.azurecr.io/behaviour_dotnet
INFO: Microsoft.Hosting.Lifetime[14]
Now listening on: http://[::]:80
INFO: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
INFO: Microsoft.Hosting.Lifetime[0]
Hosting environment: Production
INFO: Microsoft.Hosting.Lifetime[0]
Content root path: /App/

```

Figure 11 : Command windows showing behaviours after completing full set-up of experiment (after completing of W7) **W3,W4,W5**
Data shown in window is written from C# behaviour and can thus be different then the one shown on this example.

Next up you open a new window **W6** (Figure 12), where you paste

```
ssh navigation
```

```
docker login sleeptank.azurecr.io
```

In case this is unsuccessful, you probably need to give password again (replace < > by password)

```
docker login --username=mow-infra --password="<docker_registry_password>"
sleeptank.azurecr.io
```

Then you paste following commands

```
docker pull sleeptank.azurecr.io/runtime
```

```
docker pull sleeptank.azurecr.io/behaviour_dotnet:latest
```

```
mkdir --parents /home/isotron/recordings && docker run "--name=ship" --interactive=true
--tty=true --privileged --rm --net=host "--env=USERNAME=root" "--env=UUID=0" "--
env=UGID=0" "--volume=/home/isotron/behaviours:/home/isotron/behaviours" "--
volume=/home/isotron/recordings:/home/awesome:rw" sleeptank.azurecr.io/runtime ros2
launch sleeptank ship
```

In a last new command window **W7** (Figure 12), where you paste

```
ssh -t navigation docker exec --interactive=true --tty=true --user root ship ros2 run
sleeptank behaviour_resolver
```

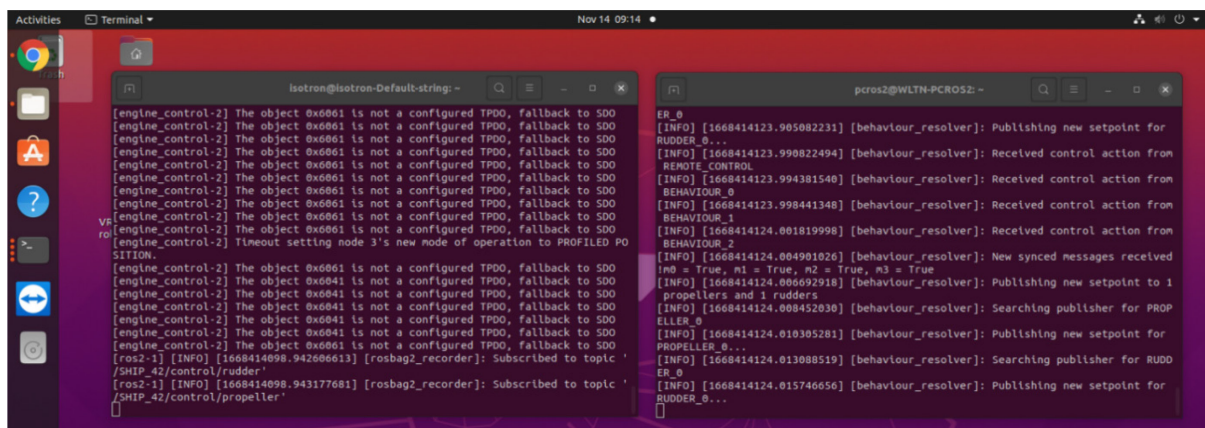


Figure 12 : Command windows for activating ship and making connection with the rosboard (**W6,W7**)

You will immediately see if all steps have been deployed correct when you access the rosboard (ship42) as the first bookmark in the explorer. It should look like in Figure 13.

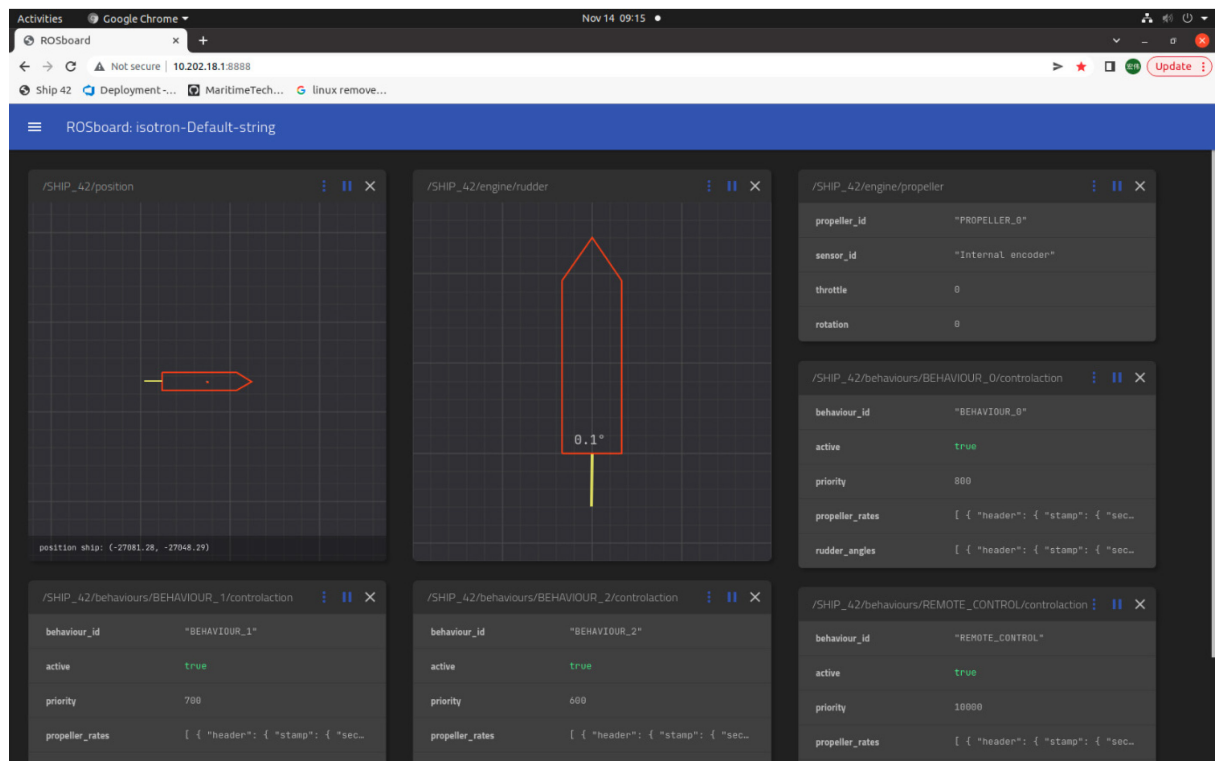


Figure 13 : Appearance of the rosboard after successfully completed the deployment flow.

5.3 Test set-up for client demonstration

On october 13th, Hongwei has presented his demo case for MDK – scheepvaartbegeleiding. This demo is a milestone in the development and will be used by FH to show to potential clients.

The demo case has been described through email by Hongwei He to the FH + UGent team on 2022/10/14, with some further additions by Thibaut Van Zwijsvoorde on 2022/10/14.

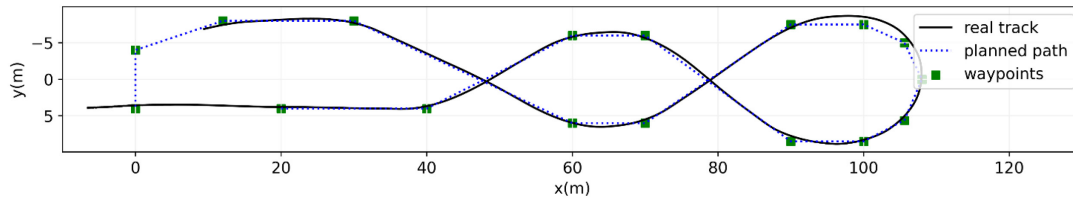


Figure 14 : Planned path and followed track following EXP33CsharpSim behaviours and config files (figure made by Hongwei He based on output data test).

From the deployment flow as described above, the wall PC steps are generic. When uploading the behaviours and configs, one needs to be access (cd) the Downloads\EXP33CsharpSim from the ROSPC (wall PC). The behaviours are named *behaviour0*, *behaviour1*, *behaviour2* as should be. The configs are named *start.config*, *zigzag_dna.config*, *stop.config*. All others steps follow the deployment flow can then be followed to complete the set-up.

The three behaviours are the following

behaviour0 is a start behaviour which accelerates the ship, *behaviour1* is the actual waypoint following, *behaviour2* is a stop behaviour which is activated when approaching the physical harbour element built into the tank for these free running tests.

When the system is deployed according to the deployment flow, remote control is active by default (highest priority number, larger than the one for the C# control behaviours). When pressing *reset*, the control behaviours are activated (or more correct, the remote control is deactivated “False”). This is also the moment when the data starts to be recorded. From this moment on, it is possible to press *pause* to again take manual control and switch back to control behaviour control by pressing *reset*. Note that the data recording is not reset. Only when stopping the experiment following the steps in the next section, the data recording is saved and a ROS-bag is created for post-processing.

In the current version of the system, the positioning of the ship is often unstable at the start of the experiment (after completing last step of deployment flow), see Figure 15. This could overturn the start behaviour (*behaviour0*) to false. In this case, one can manually accelerate the ship until *behaviour1* == true and then press *reset*.

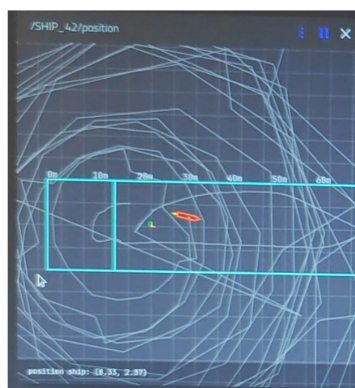


Figure 15 : Unstable positioning system when starting the experiment (example).

Below, all the steps are given, filled in the match the set-up for the demo experiment.

W1

```
docker login sleeptank.azurecr.io
```

```
docker pull sleeptank.azurecr.io/runtime
```

```
docker run "--name=remote_control" --interactive=true --tty=true --privileged --rm --net=host "--env=USERNAME=root" "--env=UUID=0" "--env=UGID=0" "--volume=/dev:/dev" sleeptank.azurecr.io/runtime ros2 launch sleeptank remote_control
```

W2

```
cd EXP33CSharpSim
```

```
ssh navigation mkdir -p /home/isotron/behaviours/dll/behaviour0 && scp -r behaviour0 navigation:behaviours/dll/behaviour0/
```

```
ssh navigation mkdir -p /home/isotron/behaviours/dll/behaviour1 && scp -r behaviour1 navigation:behaviours/dll/behaviour1/
```

```
ssh navigation mkdir -p /home/isotron/behaviours/dll/behaviour2 && scp -r behaviour2 navigation:behaviours/dll/behaviour2/
```

```
cd configs
```

```
ssh navigation mkdir -p /home/isotron/behaviours/config/behaviour0 && scp start.config navigation:behaviours/config/behaviour0/config
```

```
ssh navigation mkdir -p /home/isotron/behaviours/config/behaviour1 && scp zigzag_dna.config navigation:behaviours/config/behaviour1/config
```

```
ssh navigation mkdir -p /home/isotron/behaviours/config/behaviour2 && scp stop.config navigation:behaviours/config/behaviour2/config
```

W3

```
ssh -t navigation docker run --name=behaviour0 --interactive=true --tty=true --privileged --rm --publish=7000:80 "--volume=/home/isotron/behaviours/dll/behaviour0:/Behaviour" mocsleeptank.azurecr.io/behaviour_dotnet
```

W4

```
ssh -t navigation docker run --name=behaviour1 --interactive=true --tty=true --privileged --rm --publish=8000:80 "--volume=/home/isotron/behaviours/dll/behaviour1:/Behaviour" mocsleeptank.azurecr.io/behaviour_dotnet
```

W5

```
ssh -t navigation docker run --name=behaviour2 --interactive=true --tty=true --privileged --rm --publish=9000:80 "--volume=/home/isotron/behaviours/dll/behaviour2:/Behaviour" mocsleeptank.azurecr.io/behaviour_dotnet
```

W6

```
ssh navigation
```

```
docker login sleeptank.azurecr.io
```

```
docker pull sleeptank.azurecr.io/runtime
```

```
docker pull sleeptank.azurecr.io/behaviour_dotnet:latest
```

```
mkdir --parents /home/isotron/recordings && docker run "--name=ship" --interactive=true --tty=true --privileged --rm --net=host "--env=USERNAME=root" "--env=UUID=0" "--env=UGID=0" "--volume=/home/isotron/behaviours:/home/isotron/behaviours" "--volume=/home/isotron/recordings:/home/awesome:rw" sleeptank.azurecr.io/runtime ros2 launch sleeptank ship
```

W7

```
ssh -t navigation docker exec --interactive=true --tty=true --user root ship ros2 run sleeptank behaviour_resolver
```

5.4 Stop a test / recording

When the system is activated (last step of deployment flow), the data recording is not started yet. Only when pressing *reset* (deactivating manual control, behaviour with highest priority kicks in), the recording starts. The recordings continues until the experiment is stopped (Ctrl+C). Pressing *pause* does not lead to stopping of the data recording. It is advised to always stop the experiment (Ctrl+C), for example also before you are having lunch, otherwise there will be too much data recorded (see also section 5.7).

Stopping an experiment is done as follows (order is important).

First W7

Then W6

Then W3,W4,W5 (no particular order here)

The system is now successfully stopped, data recoding is finishing. This means that you can also not control the manually, so make sure the ship is either moored to side, in the harbour or stopped in middle of basin

Do not close the command windows, you can use them again when restarting a test.

5.5 Restarting test

There are two ways to restart a test.

Method 1 : doing the same test, with same config

If you want to repeat the same test set-up, you reactive the behaviours and connections via the command windows, by pressing 'arrow up' in all command windows, in the opposite order of how you stopped the experiment in the previous section.

First W3,W4,W5 (no particular order here)

Then W6

Then W7

This method can should also be used in case you want to take back manual control of the system using the remote control panel.

Method 2 : changes to behaviours and/or configs.

Using command window **W2** you can change behaviours / configs. You only need to reupload the behaviours / configs to the ship which you changed. For example say you change one of the waypoints in the *zigzag_dna.config*, then you do the following command (make sure you are in the correct folder on PCROS).

```
ssh navigation mkdir -p /home/isotron/behaviours/config/behaviour1 && scp  
zigzag_dna.config navigation:behaviours/config/behaviour1/config
```

When testing behaviours / control strategies, it is advised to make new folder on ROSPC (e.g. EXP34 then after EXP33). This is not necessary for the set-up, but can be used for bookkeeping purposes and to immediately find a test back when you want to execute this test. A unique numbering system as present for daily operation at ConfTT is not yet available here, hence the need for manual bookkeeping.

There is a third option, where a change is made to ROS system. It speaks for itself that then the whole system needs to be restarted, by going through all the steps (for example following the example from section 5.3).

5.6 End of testing day

At the end of the testing day, the ship is usually taken out of the basin. Note that the ship could be kept in the basin, please make sure that the system (electricity) stays on, as the water evacuation pumps need to be working. The steps which need to be followed are indicated at the end of the deployment flow. Ctrl +C , Ctrl + D in all windows, followed by the command

```
ssh -t navigation sudo shutdown now
```

5.7 Download recording from IPC to ROSPC

When an experiment has been concluded (=stopping the experiment), the data is collected in a ROSbag, which is saved on the IPC of the ship. The *deployment flow* on the *azure* gives the steps which need to be followed to download the rosbag(s)

Open a new command window (Ctrl+T). Go to a folder where you want to store the data. The following folder is the one which is mounted on WLTN. This means that you can access this folder from VO-laptop if you have the right connection established (see section 6 for a brief overview of what this means).

```
cd /media/sleeptank/PCROS2
```

Check out the rosbags stored on IPC ('ls' command gives list of all contents in location)

```
ssh navigation ls /home/isotron/recordings
```

copy name of rosbag you want to get from IPC and paste it in the following command

```
scp -r navigation:/home/isotron/recordings/<name you found in the previous step>
```

5.8 IPC memory

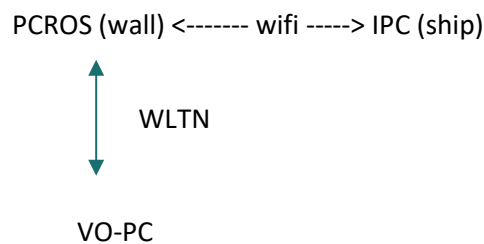
The IPC on board of the ship has limited memory and the ROSbags (certainly for long tests as shown in Figure 14) quickly become large. It is not possible as a 'regular' user to delete ROSbags from the IPC (this was done by Kapernikov to protect user from accidentally deleting data). By using the override command *sudo*, you can delete the files as admin. The structure is similar as is used to access data. Let's say the ROSbag has the name "rosbag2_2022_08_16-09_27_24", then the following command is given to remove this ROSbag from the IPC.

```
ssh navigation rm -r /home/isotron/recordings/ rosbag2_2022_08_16-09_27_24
```

6 ShaITT - WLTN – VO communication

Note : This section is written by a non-IT expert and is meant to give the user of the free running system some information and/or a first “help” when they are looking into data flows.

When looking at hardware and data flows, the free-running system is structured as follows :



The wifi network is built using 6 wifi units which are positioned throughout the basin. One folder on the PCROS (wall), the folder *Sleeptank* is part of WLTN and then thus be accessed from all VO-PC's. The accessibility to this WLTN is done through making a connection via SFTP drive software tool. WL-IT has done this for Thibaut Van Zwijnsvoorde (NAUTICA) and Pieter Hap (SPFO), who have got access via his VO-laptop.

So all files put in the folder *Sleeptank* from the PCROS can be accessed from WLTN-sleeptank, folder MOCROS\PCROS1 or \PCROS2 depending on which ROS PC is used. The pc which is used right now is PCROS2.

7 Key dates, lessons learned, action points, memo's

As the project has had many intermediate milestones, driven by the *agile, sprint* structure followed by Kapernikov, memo's have been used to keep track of the FH side of the project. Also, the free-running system has proven to be a challenging system to build and operate. Hence, a lessons learned section has been added. As with writing this report, the current project for Kapernikov has not been finished, and there is a significant chance of follow-up project, an action point list has been defined.

7.1 Memo's

7.1.1 [M16 107 15](#)

Author : Guillaume Delefortrie. Document where the model scale accuracies required for the free running system are derived.

7.1.2 [M16 107 17](#)

Author : Thibaut Van Zwijnsvoorde. Note written in November 2021 identifying actions for beginning of 2022.

7.1.3 [M16 107 19](#)

Author : Thibaut Van Zwijnsvoorde. Focus on action points free running system first trimester of 2022. Most parts are not relevant (as in dealt with), apart from the discussion on alternative methods to accelerate the system (e.g. air propeller) in the absence of a towing carriage.

7.1.4 [M16 107 22](#)

Author : Thibaut Van Zwijnsvoorde. Description of first start/zigzag/stop tests (2022/02/02) executed.

7.1.5 [M16 107 23](#)

Authors : Thibaut Van Zwijnsvoorde, Guillaume Delefortrie, Bart Verhagen. Action list / need for clarification written by FH to share with Kapernikov (Bart). Kapernikov replied to all points. Some of them are outdated, others are still active. Estimations of time/effort to spend by Kapernikov on extension of the current system can be used to write (potential) new "aanbesteding".

7.1.6 [M16 107 24](#)

Author : Thibaut van Zwijnsvoorde. Description of boundary conditions needed to do free running experiments. Some of this information is now written out in more detail in this report. As it is relevant for daily operation of the system (e.g. which persons have qualifications), is it included here in this report as appendix (Appendix 1).

7.1.7 [M16 107 25](#)

Author : Thibaut Van Zwijnsvoorde. List of remaining action points, is written out in this report (below).

7.2 Key dates

The list below has been based on two excel sheets which have been kept as diary by Thibaut Van Zwijnsvoorde during the daily operation of the system.

[Nov 2021 – June 2022](#)

[June 2022 – December 2022](#)

2022/01/27	First zigzag manoeuvre using C# controller
2022/02/02	First start-zigzag-stop (3 behaviours in C#)
2022/02/17	Frist demo (Seafar, Westray, Tresco)
2022/03/03	Thibaut Van Zwijnsvoorde (FH) + Hongwei He (UGent) operate system without Kapernikov's supervision using <i>deployment flow</i> .
2022/03/28	KVLCC2 (demo ship) moved to Antwerp for ballasting
2022/04/26	KVLCC2 returns ballasted (design draft) to Ostend + new IPC on board of ship
2022/06/17&29	data gathering by Kapernikov : generic tests to validate sensor fusion
2022/06/27	Offset in time stamp problem solved by update Kapernikov
2022/06/29	Connection VO-laptop to PCROS2 made using SFTP software from Thibaut's VO-PC.
2022/08/09	Turning circle experiment to check quality of the sensor fusion
2022/09/23	Opening of the system by SG Filip Boelaert during klankbordgroep MASHCON

7.3 Lessons learned

- When using C# control algorithms from ConfTT, they are not immediately ready for ShalTT ROS system. Following points : check units (e.g. propeller rates, rpm or % of max), position vector (given by ROS as input to C#) input format, true/false structures, presence release/grab part (ConfTT),...
- When starting the system, the current status of the software is compared with the server version (ROS,...). If updates are available, they are installed. Some online repositories, e.g. CAN-bus drive provide updates, but the system actually cannot run these updates, so they should be disabled. These types of errors have been encountered but should all have been solved. However, when updates are installed (visible in command window), you should know that there was an update from Kapernikov. If the update is not coming from Kapernikov, it could be other driver update which might cause errors. It is advised to contact Kapernikov whenever updates are installed for ROS system.
- If during start-up the rudder moves to extreme position (+- 75°), reset rudder (disconnect and reconnect fuse) and restart system.
- Check that *deployment flow* is in the 'master' mode and not in the 'dev' mode before starting a test.

7.4 Action points Kapernikov

1. Position vector (output) : Documentation of units / axis system used.
2. Up-to-date version of the deployment flow.
3. A constant & stable timestep should be attained.
4. Delivering the user interface
5. Delivering of post-processing tool, should be part of user interface framework.
6. On Rosboard, axis system displayed on towing tank border is incorrect + sign of the y-coordinate in position is incorrect. Description of Rosboard is not available.
7. At some occasions, the system was drifting (position + heading), the ship went “out of the tank” on the Rosboard display. This has been shared with Bart Verhagen (Kapernikov) on 2022/08/17 by email.
8. When starting up the system, the system oscillates (circles converging to actual position). This influences true/false conditions behaviours.
9. When performing turning circle test, the position data given by ROS seems incorrect, as the speeds (u ,v) do not (average) a constant value when performing a turning circle manoeuvre (shared with Kapernikov, included ROSbags), on 2022/10/28.
10. Accelerations are not given in position vector.
11. No report on the system is available. Azure site (wiki) is useful, but hard to find correct page / section as user (certainly with version control etc.). At the moment, you need someone from Kapernikov to guide you to the section you need.
12. Description of the position vector, including input (NAV350, IMU, camera's) and settings Kalman filter as part of current work (= the point included in this list)

7.5 Action points FH

This list gives the points which should be addressed in the future by FH. Many of these points would probably be delegated to Kapernikov. These points are not part of the current scope, as foreseen within the two ‘bestekken’. Additional budget will need to be available, whether this is by FH, EVFH, VIAIO, Blauwe Cluster or other initiatives. Thibaut Van Zwijnsvoorde will investigate the possibility / potential of starting a VLAIO and/or Blauwe Cluster track in 2023.

- I. The test bookkeeping is right now manual effort, where test behaviours, configuration and output need to be kept in list (e.g. excel). A fixed test definition and method by ROS system to produce unique output file including input parameters and output series is not available (see M16_107_23 for estimation of cost by Kapernikov).
- II. An error in a behaviour (C# code) and/or los wifi-connection and/or crash IPC can cause the ship to continue with last rpm and rudder command. This can lead to unsafe situations for ship / equipment. This has been encountered several times, without any major damages, but of course there is always a risk involved when control cannot be taken.
- III. For each behaviour, the main “dll” needs to be named *behaviour.dll*. This is the one which ROS communicates with. Sub-dlls, called from the main *behaviour.dll* are then not subject to name convention. However, Hongwei He noticed that the sub-dll's gave issues (they were not recognised). This could be due to ROS set-up system, but could also be linked to build version using VisualStudio. The root cause has not yet been found. Underlying, since Guillaume Delefortrie (then FH) made the first code available, the C# algorithms are now all updated by Hongwei He, FH should make sure that all C# algorithms are documented and can be handled by FH. When doing this process, the uncertainty regarding naming / version of sub-dll can be addressed.

IV. Sensor fusion might need further enhancement, depending on the product which Kapernikov will deliver under the current scope (section 7.4).

V. The memory of the IPC is limited and ROSbags are stored on the IPC. They can be removed, using “sudo” master command. It would be easier if they are automatically transferred to ROSPC from the IPC in order to keep the IPC memory available.

VI. The Aruco markers are not positioned according to Kapernikov initial target. The updated position (ID + coordinate) has been communicated with Kapernikov but should be readily available to Kapernikov, FH and any other parties working with the system.

VII. The accuracy of the system should be defined. One possible method is do signal analysis on the test signals. Another approach is to connect to the ship to the towing carriage (rigid in horizontal plane) and compare measured signals by free running system with the known (accurate) positions of the towing carriage.

Appendix 1 : M16_107_24

This memo gives the requirements for performing free running experiments with the ROS controlled ship, as developed by Kapernikov, status 2022/06/24. This guides does not give an in-depth description of the systems, for which there is referred to the [Kapernikov Azure site](#) and upcoming WL report (expected second half of 2022).

Terminology

VO-laptop	standard laptop from Flemish Government with access to VO-net (M,P,W drives)
WLTN	“WL-technisch” network which is a separate network, used for data collection and storage from physical models.
PCROS	Two PC’s (PCROS1 and PCROS2) are available to connect to IPC on board of the ship. At the moment, PCROS2 is used at towing tank side (PCROS1 is in office to side of towing tank). These PC’s are part of WLTN. A connection from VO-laptop is possible but not standard. Password to enter PC’s is ‘MOC4ever’
IPC	Industrial-PC. Unit on board of free running ship, connected to PCROS.

Personal

Needed to run the experiments :

- Minimum 2 people : one person to man the control station and one person to collect the ship and put it in ready position for next experiment
- 3 people is standard crew : one person to man the control station one person to collect the ship and put it in ready position for next experiment and one person to be present around the ship to intervene whenever necessary.

Required qualifications [qualified persons 2022/06/24 added in blue](#)

- person who can operate the *rolbrug*
{[Stefan Geerts](#), [Franky De Schepper](#), [Joeri Tijmsmans](#), [Pieter Hap](#), [Thibaut van Zwijnsvoorde](#)}
- person with suit to enter the basin (can be same person who can operate *rolbrug*)
{[Stefan Geerts](#), [Franky De Schepper](#), [Joeri Tijmsmans](#), [Pieter Hap](#), [Thibaut van Zwijnsvoorde](#)}
- person who can use the Kapernikov free running system
{[Thibaut Van Zwijnsvoorde](#), [Hongwei He](#)}
- person who can connect VO-laptop with WLTN *
- {[Thibaut Van Zwijnsvoorde](#)}
- person who can build behaviours using Visual Studios (C#) *
- {[Guillaume Delefortrie](#), [Hongwei He](#), [Thibaut Van Zwijnsvoorde](#), [Jose Villagómez](#)}

* If all behaviours and configurations (=needed input for experiments) are in place on PCROS, these profiles do not need to be present. During trail period (summer 2022) it is expected that all profiles will need to be present to test the system.

For information on how to follow training for *rolbrug* certification,
contact hilde.buffel@mow.vlaanderen.be (time depends on training dates)

Suits to enter the basin are available in several sizes,
contact pieter.hap@mow.vlaanderen.be for available sizes and hilde.buffel@mow.vlaanderen.be for new orders.

If you want to learn how to work with the Kapernikov free running system, contact thibaut.vanzwijnsvoorde@mow.vlaanderen.be (one day at MOC for training)

If you want your VO-laptop to be connected with WLTN folder (->PCROS1 and PCROS2), contact ian.bruers@mow.vlaanderen.be (15 mins online, Ian takes control if your laptop)

If you want to learn Visual Studies and C#, contact Guillaume.delefortrie@mow.vlaanderen.be (it takes time to master this, ask Guillaume for advice) good sources for tutorials are found [here](#), and [here](#) and [here](#)

Preparation (day before testing)

- Check water level in the tank
- Charge batteries (at least one of the batteries should be fully charged for a full day of sailing)
- Check reflectors and ARUCO-markers. The surface should be clean and they should be attached properly. *When during experiments, markers or reflectors are hit, no problem ! Just contact pieter.hap@mow.vlaanderen.be to make sure it is fixed. Experiments can run with one or 2 units missing.*

Testing day

- Ship is placed into the water
- ROS system is started
- Tests are performed ; with automatic running in one direction and manual control back.
- When ship is back at harbour side, stop propeller and rudder and guide manually into the small harbour, ready for next test to be started.
- Unique naming should be used for data files
- At the end of testing day ship may remain in the water, **HOWEVER**
 - Pumps to evacuate water around shaft need to remain active, otherwise the ships will sink
 - Battery capacity needs to be sufficient to remain active. Only do this with fully charged batteries at the start of the day
- If any issues arise (or questions about the performance of the system), complete the excel log for testing. Contact thibaut.vanzwijnsvoorde@mow.vlaanderen.be in case assistance / feedback from Kapernikov is needed on short term.

DEPARTMENT **MOBILITY & PUBLIC WORKS**
Flanders hydraulics Research

Berchemlei 115, 2140 Antwerp
T +32 (0)3 224 60 35
F +32 (0)3 224 60 36
waterbouwkundiglabo@vlaanderen.be
www.flandershydraulicsresearch.be