NTNU
Norwegian University of Science and Technology

Digital Twin Development

Condition Monitoring and Simulation Comparison for the ReVolt Autonomous Model Ship

Alexander Danielsen-Haces

Master of Science in Cybernetics and Robotics
Submission date: June 2018
Supervisor: Morten Breivik, ITK
Co-supervisor: Tom Arne Pedersen, DNV GL

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Preface

This thesis concludes the author's 2 year Masters degree in Cybernetics and Robotics from the Norwegian University of Science and Technology (NTNU).

The ReVolt Model Ship, colloquially refered to as ReVolt, is an experimental autonomous ship, being developed by DNV GL (Den Norske Veritas - Germanischer Lloyd). Working on ReVolt was chosen as autonomous vehicles has always been a personal interest. The Digital Twin concept was chosen as one of several challenging aspects of the vehicle that DNV GL and NTNU wanted to research and develop. It is a technology that has been pivotal in the other sectors and I am excited to help bring that technology to autonomous marine vehicles.

I would like to thank my project supervisors, Morten Beivik and Tom Arne Pedersen, whose guidance has been crucial in the development of this project and paper. Without their assistance, this project would not be possible. I am extremely grateful to the many people at DNV GL who have answered my questions, and provided invaluable help to the development of this project. This includes, but is not limited to Kristoffer Eide and Levi Jamt. I would also like to thank my friends and family for being an important support system encouraging me during stressful times and celebrating with me during my accomplishments.

Several people and groups have helped make this thesis a reality. DNV GL has been vital in providing assistance with ReVolt. Specifically, they have developed the ReVolt Simulator and helped guide me in the use of it. Stat Towing Tank built the ReVolt Test Model, sponsored by DNV GL. This was further developed by Kjetil Muggerud and Henrik Alfheim during their masters project, specifically building a Robot Operating System (ROS) code base to work off of. Albert Havnegjerde and Vegard Kamsvåg have also been developing on this code base concurrently with my developments.

In regards to the testing with ReVolt, Tom Arne Pedersen from DNV GL was critical in organizing the follow boat, while Stefano Bertelli was critical in organizing transport to and from the harbour. Fellow master student, Albert Havnegjerde, executed several of his tests concurrently with me, and provided help during the process.

DNV GL also provided a playground subscription of Microsoft's Azure Cloud Services for me to work on during this project.

Omega Workshop provided quick 3D printing services for the encoder brackets. Both Omega Workshop and Ascend NTNU allowed loan of their tools during this thesis, specifically used for mounting and wiring the encoders.

Glenn Angell and the mechanical workshop was vital in diagnosing and fixing the port side motor.

Last, but not least, bi-weekly meeting with my supervisors Morten Breivik and Tom Arne

Pedersen were imprtant in providing guidance and direction for this thesis.

This thesis is a continuation of the specialization project [1] from the Fall of 2017. That project involved the development of much of the Digital Twin infrastructure for ReVolt. While this thesis also continued that development, it also involved exploring several of the use cases of a Digital Twin system, including condition monitoring and system identification.

In addition, the following software was used and/or developed in:

- **Unity 3D Editor** - Development of ReVolt's Visualization tool
- **Solidworks** - CADing of the encoder mounts.
- **Robot Operating System (python and C++)** - ReVolt's software framework - azure connection and encoder drivers
- **Arduino** - encoder drivers.
- **Matlab** - building of Condition Monitoring Neural Network, data visualization for report.
- **DNV GL ReVolt Simulation** - Connected to Unity Visualization, and test day data run through Simulation.
- **Microsoft Azure Cloud Services** - handling of incoming data from ReVolt Model to Azure Services
- **java** - Turning ReVolt data into TSQ files for running in the simulator.

For more detail regarding the full list of work done in this thesis refer to section 1.4.

Trondheim, 2018–06–17

....

Alexander Danielsen-Haces

Abstract

Digital Twin technology is becoming an integral part of simulation, testing and operation of semi and fully autonomous vehicles. This technology shows great potential for decreasing testing time, improving cost efficiency and decreasing environmental impact of autonomous vehicles. This thesis includes development towards a digital twin system for DNV GL's concept test ship, ReVolt.

Several practical elements of the digital twin system have been implemented. The digital twin system now includes encoders which measure the speed of the stern motors, a 4G capable boat and a more user friendly visualization in Unity. Field tests were conducted with the objective of collecting crucial data from ReVolt. This data is crucial to test and refine the various digital twin use cases.

The first use case explored is condition monitoring. Several condition monitoring algorithms are researched and two algorithms have been developed and implemented. These produced results that can inform if one of the motors experiences a fault.

The second use case explored was system identification. Data collected from the ReVolt Model during the test day has also been compared with the ReVolt Simulator. Comparisons were made both with respect to the thruster dynamics and the overall boat dynamics. System identification theory is discussed as this can be used to improve the models in the ReVolt simulation.

Sammendrag

Digital tvilling (Digital Twin) teknologi begynner å bli en standard del av simulering, testing og drift av delvis og helt autonomt kjøretøy. Denne teknologien viser stort potensial for en reduksjon av testingstid, bedre kostnadseffektivitet og redusert miljøpåvirkning av autonome fartøy. Denne oppgaven omfatter utvikling mot et digitalt tvillingsystem tvilling system for DNV GL's concept testskip, ReVolt.

Flere praktiske elementer i det digitale tvillingsystem tvilling systemet har blitt implementert. Det digitale tvillingsystemet tvilling systemet inneholder nå encodere som måler hastigheten på sternmotorer, en 4G-kababel båt og en mer brukervennlig visualisering i Unity. Feltforsøk ble gjennomført med målet om å samle viktige data fra ReVolt. Disse dataene er avgjørende for å teste og finjustere de forskjelllige digitale tvillings bruksomgangene.

Den første brukssaken som utforskes er tilstandsovervåking. Flere tilstandsovervåkning algoritmer er undersøkt og to algoritmer er utviklet og implementerte. Disse algoritmene produserte resultater som kan informere om en av motorene opplever en feil.

Den andre brukssaken som ble utforsket var systemidentifikasjon. Data samlet fra ReVolt Modelen i løpet av testdagen er også sammenlignet med ReVolt Simulator. Sammenligningene ble gjort både med hensyn til thrusterdynamikken og den samlede båtdynamikken. Systemidentifikasjonsteori diskuteres, da dette kan brukes til å forbedre modellene i ReVolt-simuleringen.

Contents

Acronyms

ANN	Artificial Neural Network
ARX	Auto-Regressive model with eXogenous inputs
DNV GL	Den Norske Veritas Germanisher Llyod
IoT	Internet of Things
JSON	JavaScript Object notation
KNN	K Nearest Neighbors
NTNU	Norwegian University of Science and Technology
PCINT	Pin Change Interrupt
PWM	Pulse Width Modulation
ROS	Robot Operating System
SQL	Structured Query Language
SVM	Support Vector Machines

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background: Digital Twin

The growth of autonomy is inevitable, however, still quite new. It is developing rapidly in the aviation, automotive and maritime industries. There are cars, planes and boats that are capable of driving themselves, but society is not confident enough in these systems to completely switch to unmanned systems.

In order to help analyze and refine the many facets of an unmanned, autonomous vehicles, the increasing amount of data available is used. As the world grows ever more connected and technology is able to collect and store larger and larger amounts of data, a possibility opens to produce incredibly detailed vehicle simulations. These simulations are not based entirely on first principle physics models, but rather take massive amounts of data collected from the vehicles' sensors and combine them with the existing models to describe a much clearer picture of the system. These simulations are meant to be continuously connected to the physical vehicle and mimic, in real-time, every aspect of the physical vehicle. They are therefore dubbed 'digital twins.' Furthermore, these digital twins live in simulated, virtual worlds, which are also meant to emulate real world conditions for the vehicle(s).

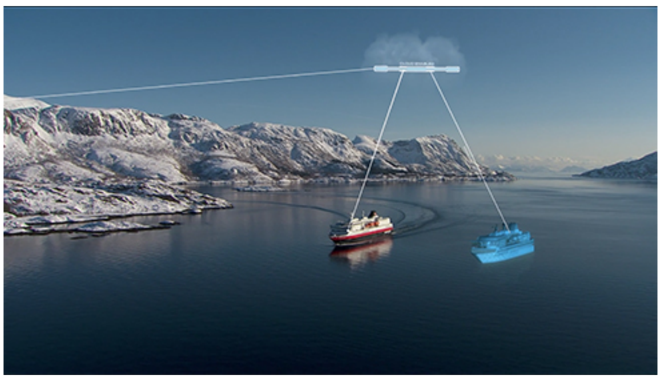


Figure 1.1: Digital Twin Illustration [2]

For ships, this type of simulation can lead to advancements in safety and operational efficiency, while achieving the goals of reducing operational cost and environmental impact.

In the early phases of a project, testing of the control systems can be aided by the data retrieved from a test model. By testing via simulation in the digital world, testing can be executed both quicker and help create more accurate models. This testing can create a safer vehicle, as the 'digital twin' has completed safety critical tests or high risk maneuvers several thousand times in varying conditions before these same tests and maneuvers are attempted on the physical vehicle.

During the operational phase of a ship, the vehicle's historical data can be accrued. This data can include everything from state data such as position, speed and loading, to environmental data such as wind and water parameters. This data can be used to make predictions about the lifespan of different components under certain conditions, and identify possible points of failure in order to save money and increase efficiency. Furthermore, examination of similar data can lead to reduced environmental effects. For example, optimal driving speeds can be determined based on examining the relationship between environmental/-loading data and battery usage data. This can be adjusted continually over the lifetime of the vehicle.

1.2 Motivation: ReVolt

In their press releases [3] [2], DNV GL has explained that ReVolt was born out of the increasing need to decongest roads of land-based transport vehicles. Currently, the maritime solution to transport of goods short distances is more expensive and more dangerous than land-based transport. As an emission free, unmanned vessel, ReVolt aims to reduce the price, risk and environmental impact below that of land-based solutions.

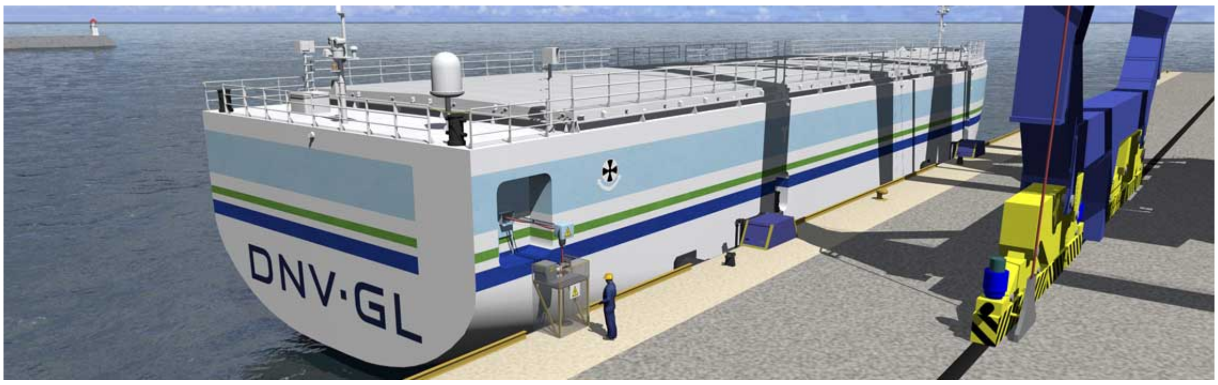


Figure 1.2: ReVolt docked. Source: [2]

One of the biggest issues in shipping industries is the safety record. More than 900 fatalities are reported every year, with a majority having a direct link to human error. This is 90 higher than in land-based shipping. By creating autonomous vessels, the human error is nearly completely eliminated. However, an autonomous vehicle is not only a safer vehicle, but also more cost effective. Without ship operators, operational labor costs are reduced. And furthermore, more of the ship can be devoted to cargo, rather than including a structure for crew facilities.

ReVolt is also an electric vehicle, powered by a 3000 kWh battery which leads to a range of 100 nautical miles per battery charge. The environmental impact of electric vehicles is far less than that of traditional vehicles, especially in countries which produce vast amounts of clean energy (for example Norway). As an electric vehicle, the number of moving parts is greatly reduced compared to diesel engines. This further reduces costs associated with the operational life-cycle of the ship. With all of these factors in mind, the cost savings for ReVolt are estimated to be 34 million USD in a 30 year life-time [3] [2].

The digital twin paradigm is therefore an important step in realizing ReVolt and its goals. The Digital Twin systems various use case are illustrated in fig. 1.3.

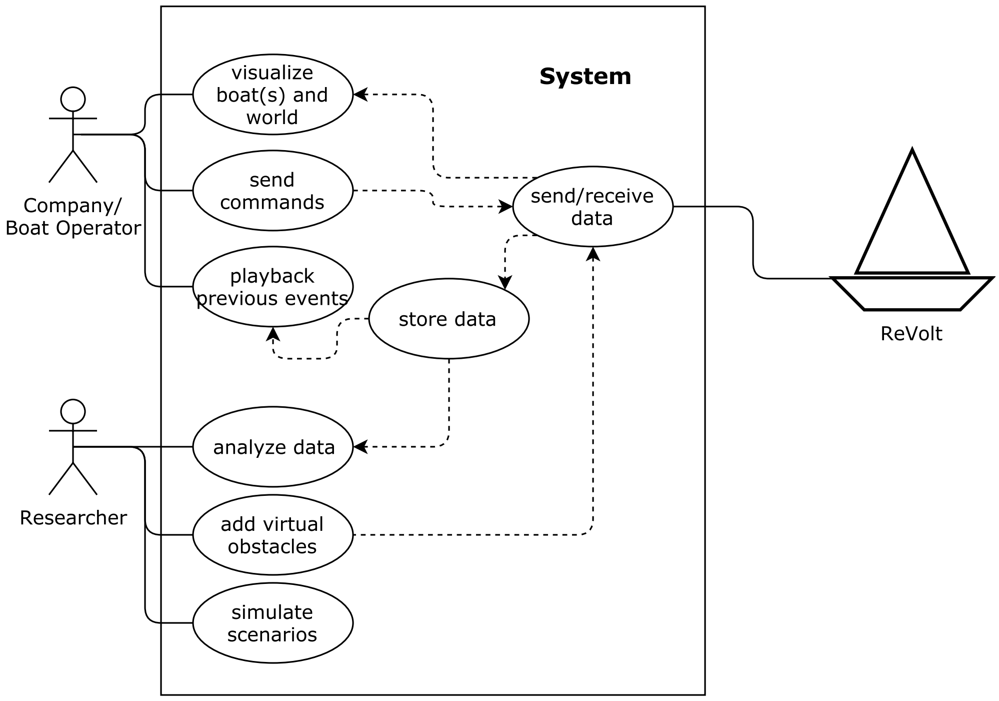


Figure 1.3: Potential use cases for ReVolt Digital Twin system

For the purposes of this thesis, the ReVolt model will refer to the ReVolt model boat that DNV GL bought from Stadt Towing Tank. The ReVolt simulation will refer to the simulation of ReVolt developed by DNV GL. The ReVolt visualization will refer to the Unity generated visualization of ReVolt and the world around it developed during the specialization project last semester. When a qualifier is omitted, ReVolt will refer to the ReVolt model from this point forward.

1.2.1 Previous Work

The ReVolt model is a 1:20 scale model of ReVolt. DNV GL has since partnered with NTNU to develop the ReVolt model as a test platform for the autonomous functions of the vehicle. One important step in the automatization of the ReVolt Model was the development of the Dynamic Position System, which was done in the Spring of 2017 [4].

During the summer of 2017, three students continued developing the ReVolt model test platform as interns for DNV GL. The three students worked in preparation for their own project and master's thesis. This work includes overhauling the code structure and documentation, making improvements to existing systems, and preparing ReVolt to be an IoT device.

For the code base, several improvements were made. All scripts in the ReVolt source code were converted into classes, and a central node was created for handling all control modes (heading, autopilot, dp, etc.) in order to improve scalability of the system. The code base was also connected to the git version control system, so that it was easier for several people to work on concurrently. Finally, the online user manual for ReVolt was updated to reflect all changes from the summer.

Several existing functions were improved upon during the summer. The stepper motors were overhauled by greasing the exterior and brushing the contact surfaces, as they were sticking quite often. Current measurement sensors were added to the thrusters in order to help in speed and battery loading estimation. The RC remote control handling was improved by adding a dead-zone near the neutral values, to prevent accidentally leaving something on when it was meant to be off. Finally, the emergency stop button function was improved to ensure the ReVolt reverted to a neutral state when pressed, to prevent unwanted motion when depressed.

In order to connect ReVolt to the internet, and interact with it as an IoT device, a 4G router was installed to the model. In addition, development began on a application in Unity to visualize ReVolt.

During the Fall 2017 semester, the three summer students each began work on their own additions to the ReVolt system. Albert Havnegjerde developed a Remote Control & Monitoring station to connect to and control ReVolt from a remote location [5]. Vegard Kamsvåg began development on a sensor system, fusing multiple sensors, in order to detect the environment around ReVolt [6]. Alex Danielsen began work on creating a digital twin of ReVolt, including a visualization application in Unity and data storage/processing pipeline in Azure [1].

1.3 Problem Formulation

The problem of this master's thesis is three fold. The first is specific improvements to the system developed in the Fall of 2017. The second is to create a Internet of Things (IoT) style condition monitoring system for the thrusters of ReVolt. The third is to develop metrics for determining the accuracy of the ReVolt simulation in comparison to the the ReVolt model.

The specific objectives are:

1. Improvements to System
   - (a) Unity Application Improvements
     - i. Switching between several viewpoints in the 3D virtual world (e.g. several fixed views and a human operable view)
     - ii. Create a test environment in the Unity App which is identical to the ocean inlet testing area
   - (b) Define data to be sent and stored
   - (c) Add 4G functionality to the ReVolt Model
   - (d) Install encoders on the ReVolt Model and create a driver to measure motor speed
2. Condition Monitoring
   - (a) Create a benchmark test for the ReVolt Model to run and collect data
   - (b) Develop a platform and user interface for analyzing trends real-time
   - (c) Implement monitoring and alerts for the thruster system of the ReVolt Model
3. Comparison of Model and Simulator
   - (a) Run the Simulator and the ReVolt Model with identical inputs and compare data such as thrust input signals, maneuvering capability, etc.
   - (b) Explain theory for adjusting simulator parameters based on differences between the ReVolt Model and the Simulator, with focus on the thruster system

1.4 Contributions

The main contributions are:

- Add encoders to determine thruster RPM on the stern thrusters

- Add 4G capability to the ReVolt model to allow data collection to the Azure platform when not connected to WiFi
- Improve the usability of the Unity Application including adapting the camera controls and developing a map of Trondheim harbour
- Execute tests in the Trondheim harbour to collect data for developing condition monitoring algorithms and for comparing with the ReVolt simulation
- Develop and implement condition monitoring algorithms for the stern thrusters
- Assessment of the simulator compared to data collected from the tests in Trondheim harbour
- Suggestion of future work to improve the ReVolt's digital twin features

1.5 State of the Art

While the focus of this report is ships, specifically DNV GL's ReVolt test platform, the idea of a digital twin living in a virtual world is not confined to the maritime industry.

Industrial processes such as pumps [7], wind farms [8] , and glass production [9] are all developing digital twins for their machinery. However, perhaps the most interesting and most advanced industry which has developed digital twin and virtual world technology is the autonomous automotive industry. Waymo, the Alphabet (Google's parent company) subsidiary, has developed an extensive and detailed set of simulated maps for "digital" cars to drive in. This virtual world with digital twin cars is a way to log exponentially more miles than physical vehicles could [10]. This is especially important for Almotive, who is a smaller autonomous car company. They cannot afford the same fleet as larger competitors such as Waymo and Tesla, so it is necessary for many tests to occur virtually in order to stay competitive. Furthermore, while most miles driven in real life are quite strait forward, with no unexpected surprises, it is possible in the virtual world to force difficult scenarios that are not yet solved. This allows time to be used on developing solutions to problematic situations instead of wasting a lot of time on simple, obstacle free, straight driving, which has already been solved [11]. Waymo has taken this a step further, and has developed virtual maps based on places and scenarios that the real cars have needed human intervention. For example, a multi-lane roundabout in Dallas once caused an issue for a Waymo car, and the car had to be controlled by the driver. This roundabout was added to the virtual world, and using the data from the car, they were able to recreate the situation and adjust the control parameters until the digital twin successfully handled the scenario. After the scenario is solved in the virtual world, they move to their real world testing facility in Castle, California. Castle is an entire town (of only roads) built for testing the vehicles. However, due to the virtual world testing, test time at Castle is vastly reduced and the whole process is streamlined [10]. This exact principle and testing cycle can be implemented by autonomous maritime vehicles, in order to speed up the time frame for testing.

The concept Digital Twin has been around for at least 15 years. However, there are three technologies that are now available in order to allow for the creation of digital twins as envisioned. The first is high fidelity simulation tools. Instead of just being able to model a 3D design, one can now implement and simulate the physics of the design in great detail. This combined with a basis of operational data allows for models more accurate than ever before [12]. Several programs are available for these simulations, including gaming programs such as Unreal Engine [13] and Unity 3D [14] for visualization of simulations, CAD programs such as AutoDesk's AutoCAD [15] and Dassault Systems' SolidWorks [16] for building up the physical models, and simulation programs such as Dassault Systems' Dymola [17] and Mathworks' Simulink [18] for the mathematical models. Many solutions will take advantage of several programs, in addition to custom solutions.

The second is the scale of connected sensors, the Internet of Things. Never before has data transfer been so cheap and so easy from anywhere in the world (and in outer-space) as it is today. Data can be collected on nearly every aspect of a system, sent to the cloud, processed in the cloud and store as a basis for developing better future models [12]. Many services exists for transferring data to the Internet, and processing, analyzing, storing and sending the data. These include Microsoft Azure, Amazon Web Services and Google, which all offer a variety of cloud computing services.

Finally, the tools for developing predictive models based on these now available large quantities of data are in development. Techniques in machine learning and big data analytics are vital and being constantly improved and developed in order to be able to properly use the data streaming in from the IoT devices [12]. Techniques for analyzing big data include Principle Component Analysis (PCA), Principle Component Regression (PCR) and Principle Least Squares Regression (PLSR). PCA is a variable reduction method. This is important when dealing with large quantities of data, as it can help reduce the data to show which variables contribute most to the variability in the system, what important relationships there are between variables, and differentiate between variables that have a structure and those which are just noise. PCR and PLSR both create a predictive model of a particular output variable. For example, if you were interested in how the other ship parameters affect battery life, then battery life would be the output variable. In PCR, the residual error between the data points and the model is minimizes, while in PLSR, the co-variance between the input and output variables is maximized. This leads to different behavior with respect to how outliers affect the system and how small groups of data affect the system [19].

With these technologies in place, several features have been identified as potential, but not necessary components of a digital twin and its corresponding virtual world. These features were suggested for use for the structural systems in U.S. Air Force vehicles, but the principles apply to many digital twins. First, a high fidelity mathematical and physical model is needed as a means of simulation for the digital twin within the virtual world. A detailed sensor system is essential to collect historical data for use in analysis. A system to process the data and continually assess and estimate critical mission or operational parameters is important for many applications. It is useful to integrate (collect, store, and process) data from an entire fleet, for systems where there are more than one of the same type of system in operation at the same time. Finally, an anomaly detection and correct system is a useful feature for reducing costs [20].

Few examples were found of digital twins being used in the maritime industry, and none so far in the autonomous surface vehicle sector. This technology is essential to the rapid development of safe, reliable and efficient autonomous ships.

1.6 Outline

The structure of the thesis is as follows. **Chapter 2** discusses the digital twin infrastructure and improvements made to it during the course of this thesis. **Chapter 3** discusses the theory, implementation and results of a condition monitoring system for ReVolt's stern motors. **Chapter 4** illustrates a comparison between the ReVolt Simulator and the ReVolt Model and discusses the use the theory of system identification as it is used for digital twin applications. **Chapter 5** concludes this thesis and includes possible future work on the digital twin infrastructure.

Chapter 2

Digital Twin Infrastructure for Revolt

2.1 Digital Twin System Overview

The Digital Twin system for ReVolt has been developed both through the specialization project [1] and this thesis. This system is a complex network including the ReVolt Model, the ReVolt Simulator, a ReVolt visualization app, and Microsoft's Azure Services [21].

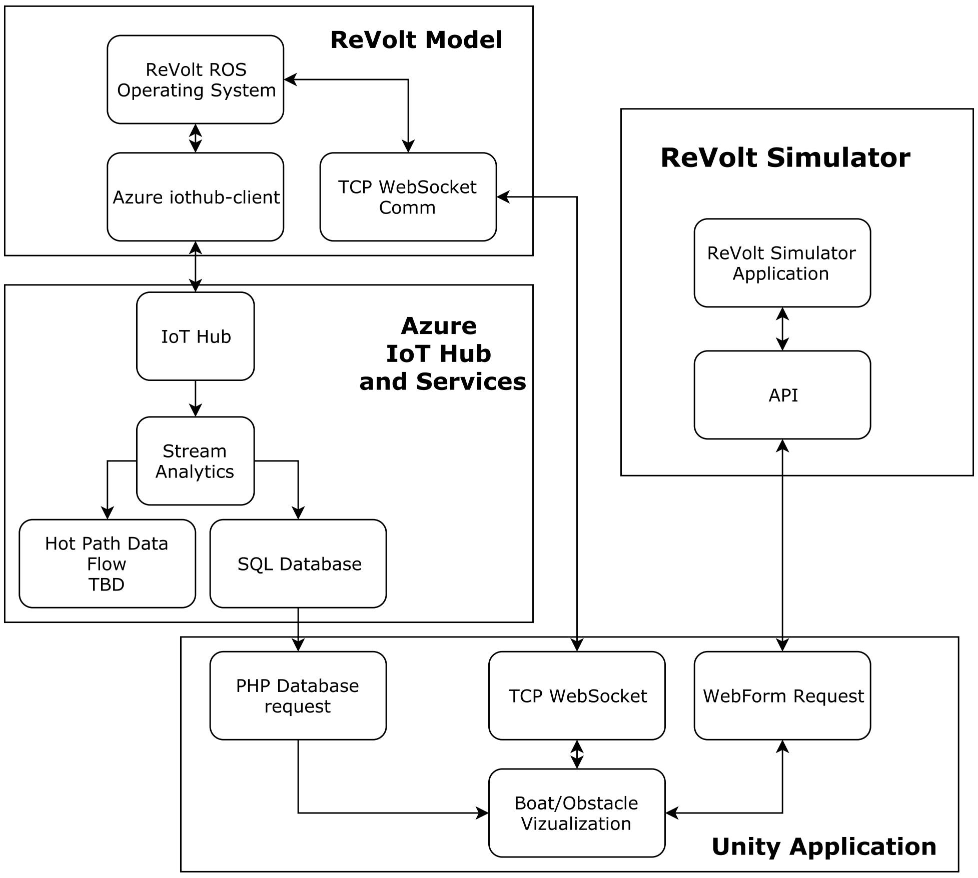


Figure 2.1: Overview of the main parts of the project and the connections between them.

Figure 2.1 shows the 4 main segments of the digital twin system, as well as the connections between them.

1. Unity Application

Unity is a 3d game creation engine. Useful Unity features for this project include a high fidelity physics engine, a UI builder to allow for interactivity, and the rich open source community. Games created in Unity can be deployed as applications to many different hosts, including as a WebApp to WebGL [14].

ReVolt's Unity application mimics the physical system from data that it receives, thereby serving as a visualization of ReVolt. This data can come from either the ReVolt Model, the ReVolt Simulator, or both. In addition, virtual object locations, generated in the ReVolt Simulator, can be transmitted from the application to the ReVolt Model.

2. Azure IoT Hub

The Azure suite is Microsoft's solution for cloud computing. It contains a host of options for data acquisition, storage, processing and analytics, and visualization and display. Each option is called a resource and several resources can be strung together into a pipline [21].

The pipeline (fig. 2.2) for ReVolt includes the Azure IoT Hub, Azure Stream Analytics, SQL Database, and Web App resources. These are used to aquire, process and store the data appropriately. Data, both archived or real-time, can be sent to the Unity Application in order to drive the virtual boat and enhance the world around the virtual boat (e.g. obstacles and waves).

3. Simulator Connection

In the absence of the ReVolt Model, or when simulator testing is desired, the Unity Application is able to connect to the ReVolt simulator, and collect data from the simulator in order to drive the virtual ReVolt and enhance the world around the virtual boat (e.g. obstacles and waves).

4. ReVolt Model

The ReVolt Model will send and receive important telemetry data to and from the Azure IoT Hub and the Unity Application.

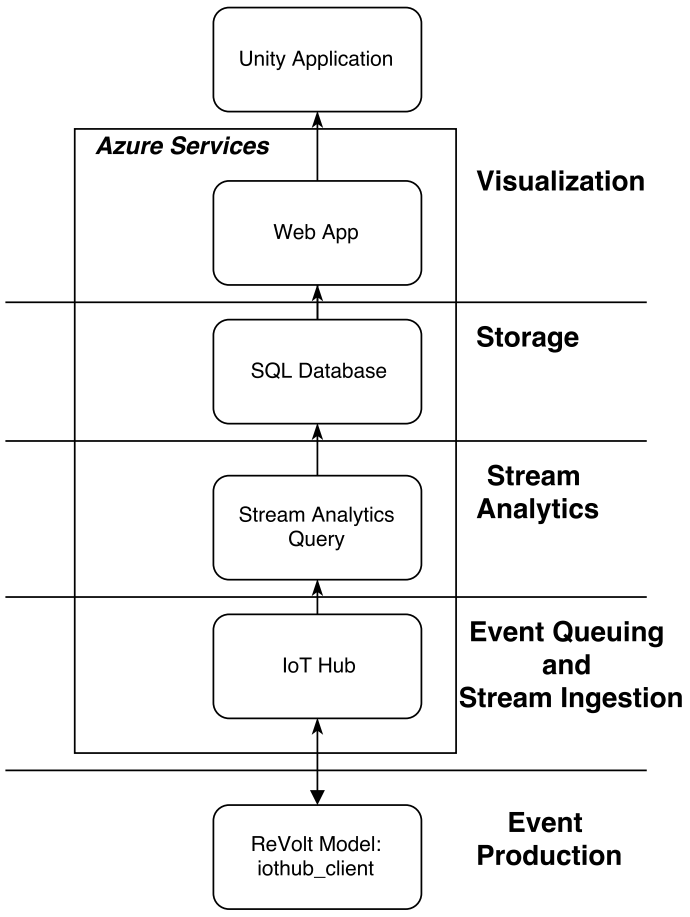


Figure 2.2: Azure pipeline of services for long term storage of data and playback mode

2.2 Unity Application Development

Over the course of this thesis the following developments were made to improve the Unity Application section of the Digital Twin system.

2.2.1 Adjustable Camera Viewpoints

With the inclusion of obstacle boats, and the later potential for a fleet of ReVolt ships, it was important to be able to switch the view between the different boats. These boats and obstacles are followable objects. Quick keys were added to the Unity visualization, so that the user could cycle through the obstacle boats and ReVolt at will. The N key (for 'next') switches to the next object that the camera can follow.

In addition, camera navigation was added with the mouse and arrow keys, so the user can decide to roam with the camera at will, if the snapped positions were insufficient. The Y button will switch between a Camera mode that follows a folloawable object and a free camera controlled completely by the arrow-keys and the mouse. These controls are summarized in fig. 2.4.

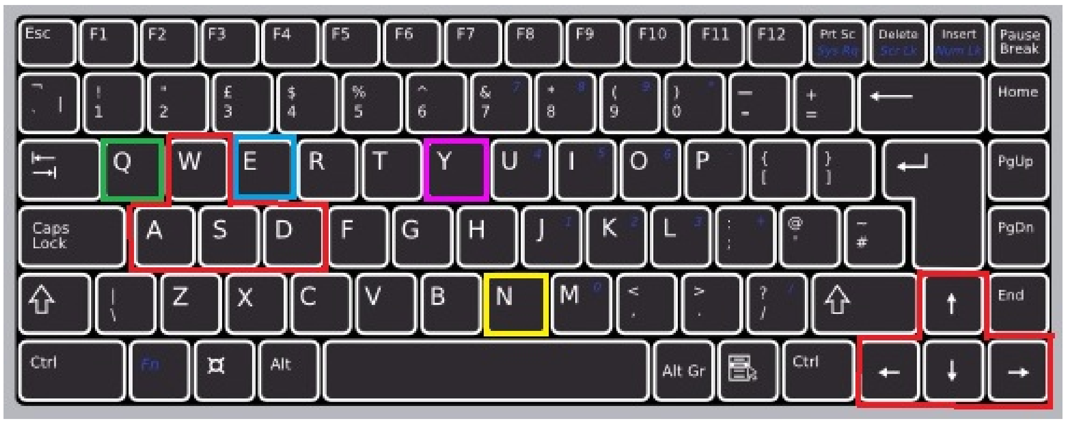
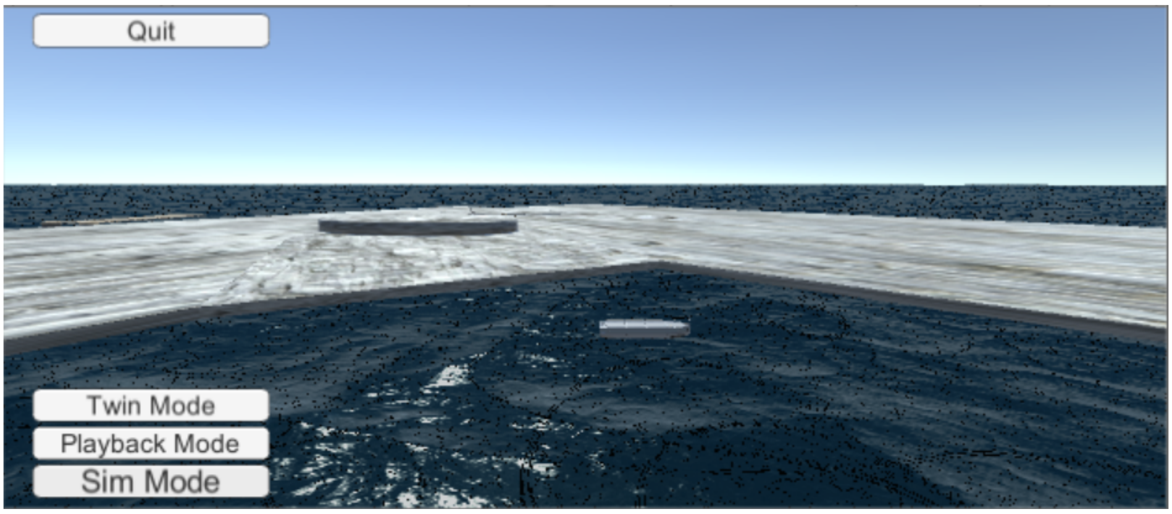


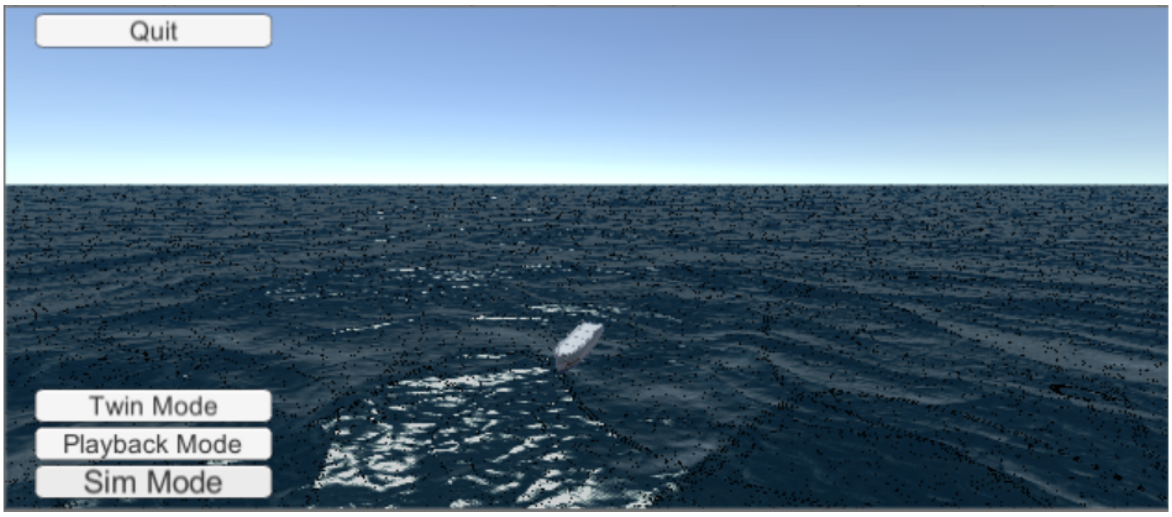
Figure 2.4: Keys to control camera
WASD/Arrows: Movement
Q: Climb
E: Drop
Y: Follow/Manual Camera
Mouse: Camera Angle
N: Next Followable Object
Image: [22]

2.2.2 Harbour Map

In order to use the visualization tool when ReVolt is out in the ocean, a digital world must be created. For this semester the only test area was the Trondheim harbour, so this was created



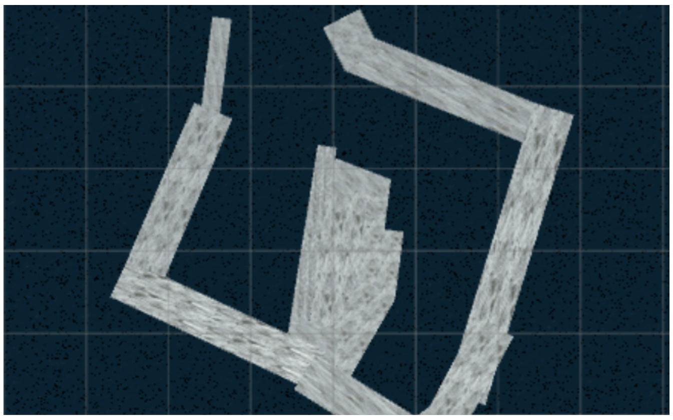
(a) ReVolt near origin point in Trondheim harbor

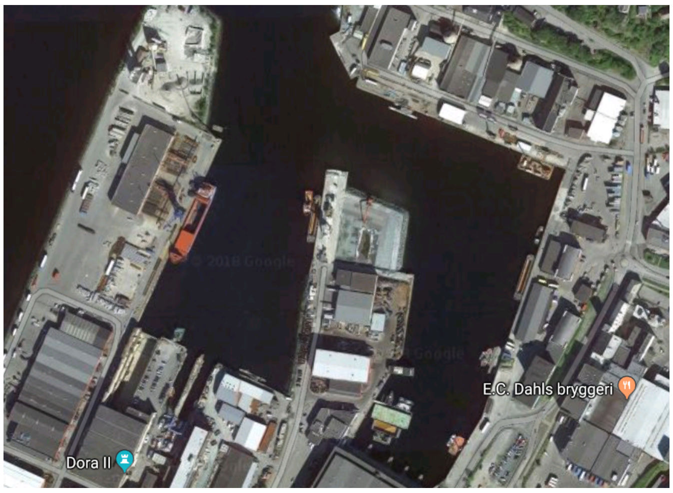


(b) Obstacle Boat several hundred meters away in open ocean

Figure 2.3

in the Unity visualization. No maps were available for the harbour, so several google maps tools [23] were used to determine the length and angle of each side of the harbour. Land objects with these lengths and angles were then positioned in the Unity world at the correct locations to create the outline of the harbour.


(a) Unity Application Map of testing area



(b) Google satellite view of testing area

Figure 2.5

In order to place the ship within this world, the southernmost corner of the harbour was used as an origin for the visualization. Then the ship needed to be placed at a specific coordinate. The x and y distance of the boat's coordinate from the was then determined by drawing a right triangle directly north, and then directly east or west to the coordinate.

In order to use find these distances, the Haversine method was used. The Haversine method takes into account the curvature of the earth and is still well suited for shorter distances where the curvature does not play a large role, to ensure accurate measurements within 0.3% [24].

The Haversine method is as follows in eqs. (2.1) to (2.3):

$$a = \sin^2(\Delta\phi/2) + \cos\phi_1 * \cos\phi_2 * \sin^2(\Delta\lambda/2) \tag{2.1}$$

$$c = 2 * \text{atan2}(\sqrt{a}, \sqrt{1-a}) \tag{2.2}$$

$$d = R * c \tag{2.3}$$

where ϕ is latitude, λ is longitude, R is earth's radius (mean radius = 6371km) [24].

The Haversine method is used to first determine the distance north, or the x distance, to the coordinate directly north, and then the distance east of west, or the y distance, to the coordinate of the boat.

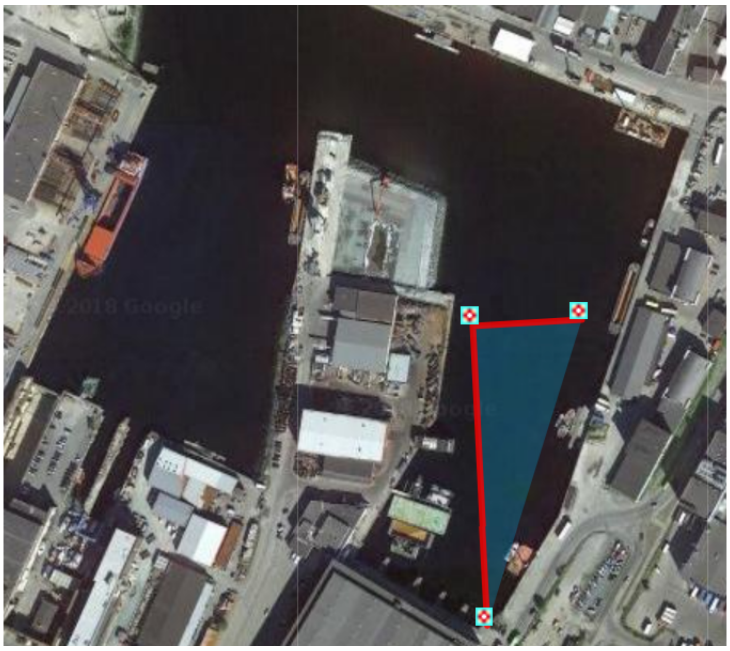


Figure 2.6: Triangle for Calculating x,y position of boat from lat-lon coordinates

2.3 ReVolt Communication Development

Over the course of this thesis, the following developments were made to improve the communication between the components of the Digital Twin system.

2.3.1 4G Connection

In order to become a truly IoT device with a digital twin, ReVolt must be able to communicate with land at all times. In order to accomplish this, a 4G sim card was added to the router of the ReVolt model. This card and subscription are provided through Telia. Only small changes to the router settings were required to set up the 4G network.

2.3.2 Data to Azure

In the fall semester, the framework for sending data to Azure was developed, but the actual data to send had not been decided. The data that is currently sent is described in Table 2.1. As the data is collected to be sent to Azure, it is time-stamped. This is important as the data is asynchronously produced onboard ReVolt.

Table 2.1: Data Overview

Data Name	**Abbreviated Name**	**Decimal format**
Time at startup	start_time	no format
Current time	time	no format
NED-Origin latitude	ned_org_lat	xx.xxxxxxxx
NED-Origin longitute	ned_org_lon	xx.xxxxxxxx
NED-Origin altitude	ned_org_alt	xx.xxxxxxxx
NED-Origin time	ned_org_time	xxxx.xx
Stern starboard measured rpm	str_star_rpm	xxxx.xx
Stern starboard measured rpm time	str_star_rpm_time	xxxx.xx
Stern port measured rpm	str_port_rpm	xxxx.xx
Stern port measured rpm time	str_port_rpm_time	xxxx.xx
Stern starboard thruster effort	str_star_effort	xxxx.xx
Stern starboard thruster effort time	str_star_effort_time	xxxx.xx
Stern port thruster effort	str_port_effort	xxxx.xx
Stern port thruster effort time	str_port_effort_time	xxxx.xx
GPS latitude	gps_lat	xxxx.xx
GPS longitude	gps_lon	xxxx.xx
GPS time	gps_time	xxxx.xx
Heading	heading	xxxx.xx
Control mode	control_mode	xxxx.xx
Dynamic Positioning desired waypoint – north	DP_desWayP_n	xxxx.xx
Dynamic Positioning desired waypoint – east	DP_desWayP_e	xxxx.xx
Dynamic Positioning desired waypoint – heading	DP_desWayP_head	xxxx.xx
Battery voltage	battery_voltage	xxxx.xx
Stern port angle	str_port_angle	xxxx.xx
Stern port angle input	str_port_angle_input	xxxx.xx
Stern starboard angle	str_star_angle	xxxx.xx
Stern starboard angle input	str_star_angle_input	xxxx.xx
Stern Port Angle Time	str_port_angle_time	xxxx.xx
Stern Starboard Angle Time	str_star_angle_time	xxxx.xx
Stern Port Input Angle Time	str_port_angle_input_time	xxxx.xx
Stern Starboard Input Angle Time	str_star_angle_input_time	xxxx.xx

This data is sent in a telemetry message to the Azure IoT Hub. From there it is sent to an event manager which logs the data in an SQL database. Sending data is activated from the command line via the command:

```
rostopic pub \azure_on -r 5
```

where the number at the end (in this case 5) represents the rate at which to send the data. It was discovered that the maximum possible rate of sending telemetry messages is dependent on the number of characters in the telemetry message. A lower rate of sending was required for telemetry messages with more characters. Therefore it was important to format each piece of data to use as few characters as possible, without losing any information. To match the ReVolt Simulator, 20 Hz was the desired rate of tranfer. Unfortunately, the highest send rate with the desired data was 5 Hz. This rate could potentially be increased with a better Azure subscription.

2.4 Encoder Development

During the course of this thesis encoders were added to measure the speed of the stern thrusters. These data are for use in the condition monitoring and simulation comparison in chapter 3 and chapter 4 respectively.

2.4.1 Hardware

The encoders used for measuring the speed of the stern thrusters are the Eagle Tree Optical RPM sensors. In order to connect to the Arduino, the circuit shown in fig. 2.7 was needed in order to ensure a consistent and proper power level to the devices.

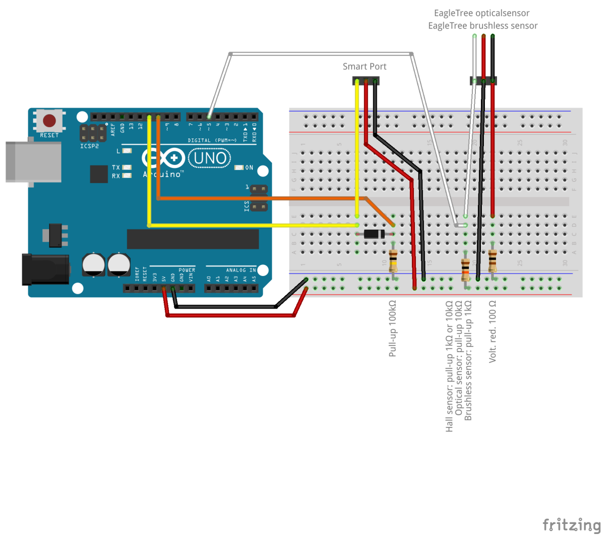

EagleTree opticalsensor
EagleTree brushless sensor
Smart Port
Pull-up 100kΩ
Hall sensor: pull-up 1kΩ or 10kΩ
Optical sensor: pull-up 10kΩ
Brushless sensor: pull-up 1kΩ
Volt. red. 100 Ω
fritzing

Figure 2.7: Encoder circuit [25]

The encoders work by shining UV light at a surface and measuring the level of reflected light. The encoder points at the flat end of the axle, and the axle is colored half white, half black. Whenever the measure of reflected light changes the encoder registers one tick of movement. The mount (shown in fig. 2.8) was 3D printed and fits snugly around the block around the axle and can be mounted with screws once the correct distance from the axle is determined. The encoder fits into the hole on the flat surface of the mount, so as to point at the flat of the axle. It is important that the encoder is the proper distance from the axle (2-3 mm according to the manual). If the encoder is too far, the light will not reflect back and it will never return a reading. If mounted too close, you could risk damaging the encoder if it contacts the axle spinning at high speed.

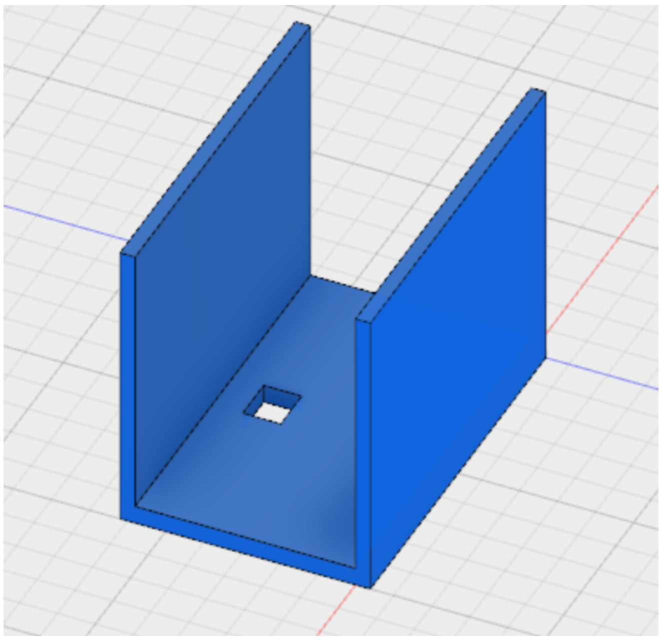


Figure 2.8: 3D printed encoder mount

One main issue with the mounting setup is the wiring. The thrusters can rotate indefinitely, but the wires as is now, can only rotate 1 or 2 full spins before becoming too tight. In the future, it would be wise to add the encoder wires to the slip ring, as is done with the power to the motors.

2.4.2 Software

Interrupt functions are functions that are run based upon an external signal, and run in their own thread regardless of where the main loop of code happens to be. Pin Change Interrupts (PC Interrupts) are interrupt pins that are grouped together on a port. All of the pins on a PC Interrupt port trigger the same, common function when any of the pins trigger an interrupt [26]. The encoders were connected to PC Interrupts ports on the Arduino Mega 2610 which controls the stern thrusters and sensors. The PC Interrupts are grouped into 3 ports of 6-8 pins. Despite each port being connected to a single interrupt function, 3rd party libraries can be used to isolate the individual pins and connect them to their own interrupt function. The PCInt library [27] was used for this purpose, so that the other PC Interrupts can be used in the future. Whenever the encoder detects a change from the black to the white section of the axel, the connected interrupt function is triggered.

This interrupt function accrues the ticks for each encoder. In the main loop of code, the accumulated ticks are converted to rotations per second using eq. (2.4).

$$rps = x\,\text{ticks} * \frac{2\,\text{rotations}}{1\,\text{tick}} * \frac{1}{dt\,\text{sec}} \tag{2.4}$$

where x is the accumulated ticks during the loop of the code, the term $\frac{2\,\text{rotations}}{1\,\text{tick}}$ is to account for the turn-down from the encoder to the propeller and dt is the change in time since the last loop through the code.

The accumulator is reset to zero after each rps calculation.

Chapter 3

Condition Monitoring and Alert System

3.1 Theory

Condition monitoring has been a technique of interest since the 1930's. It is an incredibly valuable feature for any technical system [28]. It includes not only fault detection as the fault occurs, but also prediction of when faults will occur. This makes scheduling of maintenance more efficient, as it can be planned before the device is unusable. This is also more efficient than scheduled maintenance, as checks do not need to be made as frequently [29]. This thesis focuses on condition monitoring of the stern motors, so the theoretical focus is on condition monitoring of AC induction motors.

Based on research about AC motor condition monitoring from [29], many motor faults occur in stages, in which one fault leads to another until the motor is no longer usable. It is therefore possible to monitor the motors for these early faults, and thereby fix the issues before they become damaging. The most common motor faults are: bearing faults (41%), stator faults (37%) and rotor faults (10%). To detect various faults, there have been developed three overall groupings of condition monitoring techniques: signature based, model based, and knowledge based [29].

Signature based extraction is one method of condition monitoring for induction motors described in X. Liang and K. Edomwandekho [29]. These methods involve analyzing properties of various output signals (motor current, motor speed, etc.). For example, attributes of the current signal in the frequency domain can be used to detect broken rotor bars. These algorithms are used for analyzing a motor running at steady state with constant loads. There also exists algorithms which analyze transient modes, such as start up modes (where the motor is ramping up speed from 0). A start up time of at least 0.5 seconds is required in order to detect broken rotors with these techniques. Signature based methods to detect bearing faults are best detected with vibration and acoustic monitoring. There also exists invasive signature based methods, such as signal injection techniques. These methods involve adding an additional current to the motors to determine the resistance and temperature parameters. The signal injection group of algorithms affects performance, and also requires additional hardware in order to inject the signals and measure the affect of the injected signals [29].

The signature extraction technique requiring steady-state operation to detect bearing faults is not ideal for ReVolt, as the thrusters will be constantly adjusting their speed, and be operating under varying loads. The algorithms analyzing transient modes are also inappropriate as ReVolt's motors take less than 0.5 seconds to ramp up to speed. For the algorithms detecting bearing faults, there are no vibration nor acoustic sensors on-board ReVolt, and it is likely that the operation environment of ReVolt would create too much vibrational and acoustic noise for these techniques to be useful. As far as signal injection techniques, the addition of extra current is also not ideal for ReVolt, as this would affect the boats control systems.

Model based methods are another set of methods for condition monitoring for induction motors described in X. Liang and K. Edomwandekho [29]. Model based approaches use the mathematical models of the induction motors in healthy and faulty states to predict behavior of the motor. The detection algorithms then compares the real data to what the model predicts for various faults to determine if there is a fault and what type of fault is occurring. These models are described in detail in [30] and [29]. Figure 3.1 shows an example of a model built to be used in an algorithm to detect stator inter-turn faults.

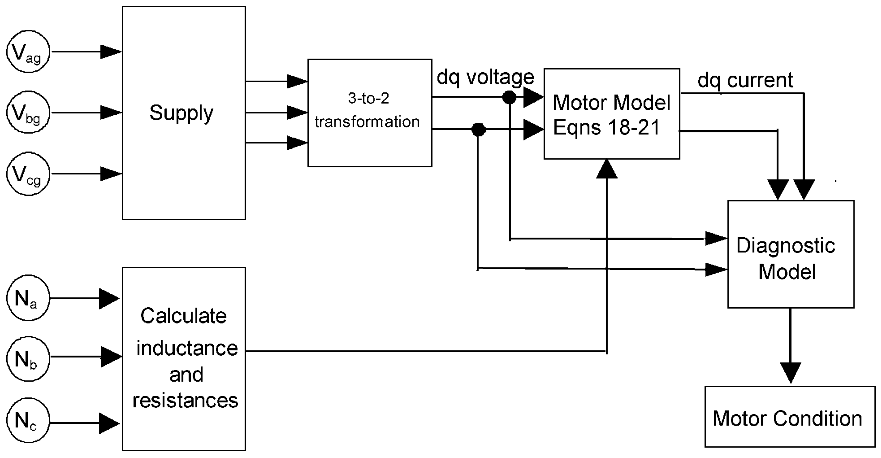


Figure 3.1: Simulink Model for Model Based Condition Monitoring [31]

The final group of methods described in X. Liang and K. Edomwandekho [29] are knowledge based methods. These methods use historical knowledge to determine faults. These methods include k-nearest neighbor (KNN), artificial neural networks (ANN), Support Vector Machines (SVM) and other artificial intelligence techniques. These methods have produced excellent results for various faults. However, they require extensive training data that includes various fault modes. These modes are well suited for the digital twin paradigm, as historical data can be continually collected while in operation, and used to improve the artificial intelligence models [29].

One specific case of the knowledge-based methods is the Artificial Neural Network (ANN). These networks are usually used for image processing, but can be applied to a myriad of problems. Each node of a network has a certain number of inputs, and one output (which can be sent on to several other nodes). These inputs are weighted and summed to determine the output. In some networks, the nodes can include activation function, which turn the output into zero or one depending on the value of the weighted sum. A single node is illustrated in fig. 3.2.

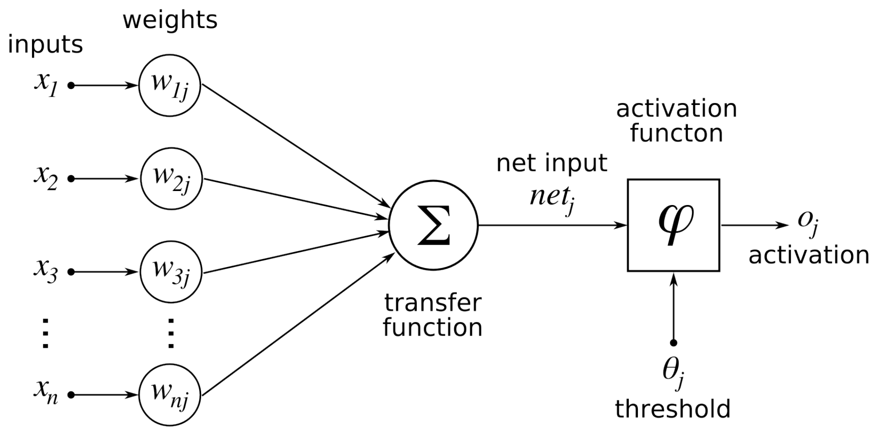


Figure 3.2: Neural Network Node. Transfer Function is often a summing function.

Nodes are typically grouped into layers, and typically, the outputs from one layer become the inputs to the next layer. The final layer of any network is the output layer, which contains the information useful to the user (e.g. fault or no fault, what type of fault, etc.). A full network used for condition monitoring of an induction motor is seen in fig. 3.3. Networks are trained by adjusting the weighting of the inputs at each node. Many training algorithms exist which adjust these weightings based on training data provided to the network. Training data is a set of inputs and their correct, equivalent outputs. The training algorithms adjust the weights of each node to best get each set of inputs the correct set of outputs.

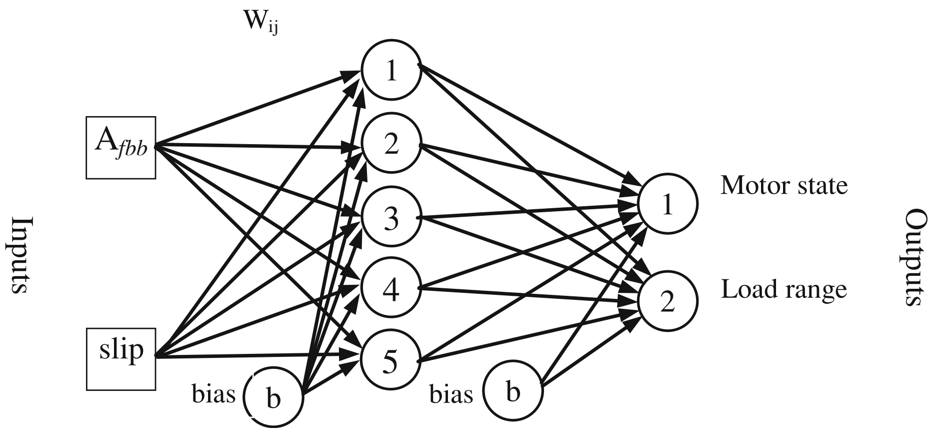


Figure 3.3: Neural Network Condition Monitoring Example [32]

3.2 Simulation

3.2.1 Algorithm Descriptions

The knowledge-based condition monitoring approaches are some of the most promising and fit perfectly with the digital twin model. Two simplified knowledge based methods were implemented. The first algorithm is based only on the data collected from a functioning motor. The second algorithm relies on specific knowledge about functioning and faulty motors. With it, in order to detect specific faults, data needs to be collected with motors that are faulty in specific ways. Since one of the motors was broken during the test day, data for a functioning and a faulty motor was collected. This allowed for the implementation of the following algorithms. Later these algorithms can be built up with more data about specific faults (e.g. 1-2 rotor faults, stator faults, etc.).

Algorithm 1: Acceptable Range Algorithm

The first method creates a model of a functioning motor based on the current (amps) and speed of the motor (RPS) at various setpoints (PWM %).

At each of the setpoints, there are several speeds and currents measured. The average and standard deviation of the speed and current at each setpoint is calculated. A range of acceptable values is then created, encompassing 3 standard deviations (3σ) around the average at each setpoint. A polynomial function is fit from the lower and upper 3σ values to create a zone that represents a normal functioning motor. This zone can be seen in fig. 3.5 and fig. 3.6.

Even with a normally functioning motor there are points that fall outside of the normal zone. Therefore, an alert is only sent when there are over 2 continuous seconds of readings which fall outside of this zone. This prevents false positives for cases when the motor is outside of the defined normal operation for only one or two readings. This is also important for transient states, for example, when the motor is ramping up or down to a certain % PWM output, and it takes time before the speed and current stabilize to a steady state value.

Algorithm 2: Neural Network Algorithm

The second method is similar to the ANN developed in [32]. The input nodes are setpoint (PWM %), motor speed (RPS) and motor current (amps). There is one hidden layer which contains 10 nodes. The single output node is a number representing whether or not a fault has occurred (0 indicating no fault, 1 indicating a fault). When the output is greater than .8 for 0.5 seconds continually, the system triggers an alert. This prevents false positives for cases when the motor is outside of the defined normal operation for only one or two readings. This network needs labeled data from functioning and non-functioning motors in order to train the network.

For both algorithms, when an alert is triggered, the message is sent to the Azure IoT Hub. This alert can then be broadcast to the appropriate personnel.

3.2.2 Testing Method

Since the algorithms are knowledge based, training data from ReVolt in real world conditions is needed in order to inform them. Two separate tests were run in order to collect this data. The details of each test are described in detail in the test plan in appendix A.

The tests were conducted on April 17th, 2018 in the Trondheim harbour (Trondheim havnbasseng). The weather was between 10°C and 14°C, with a breeze of about 13 km/hr. The harbour was very still with little to no wave activity [33].

During the test, the port side motor was experiencing issues. After the test day, the personnel at the electrical workshop at NTNU dissembled, identified and fixed the motor. The issue was identified to be a break in one of the 3 power cables to the motor.

Test 1: Step Inputs Test

This test was designed to get data about how a functioning motor works at steady-state. It involved driving the motors at various setpoints, starting a 50% and incriminating in steps of 10% up to 100%, with enough time to allow for the motor speed to stabilize at each setpoint.

The speed (rps) and the current (amps) of each motor are measured. These responses are plotted along with the input (motor setpoint) in fig. 3.4.

Despite the port side motor experiencing problems during testing, a baseline was able to be determined from the starboard side motor. Later, after the port side motor was fixed, it was determined that the motors behave in a similar enough manner to use the same model for both. The starboard motor of the test-run seen in fig. 3.4 was used to develop the normal ranges.

These normal ranges determined from this test are seen in fig. 3.5 and fig. 3.6. For speed (fig. 3.5), a second order function was chosen, as that seemed to follow the trend of the data best. For current (fig. 3.6), a third order function was chosen, as that seemed to follow the trend of the data best. However, this function creates an odd "normal" range at lower motor setpoints (10-30), since this range dips into negative amps. To increase the accuracy of these functions, it would be best to run tests again with steps that start at 10 % and increase from there (e.g. 10%, 20%, 30%, ..., 80%, 90%, 100%) .

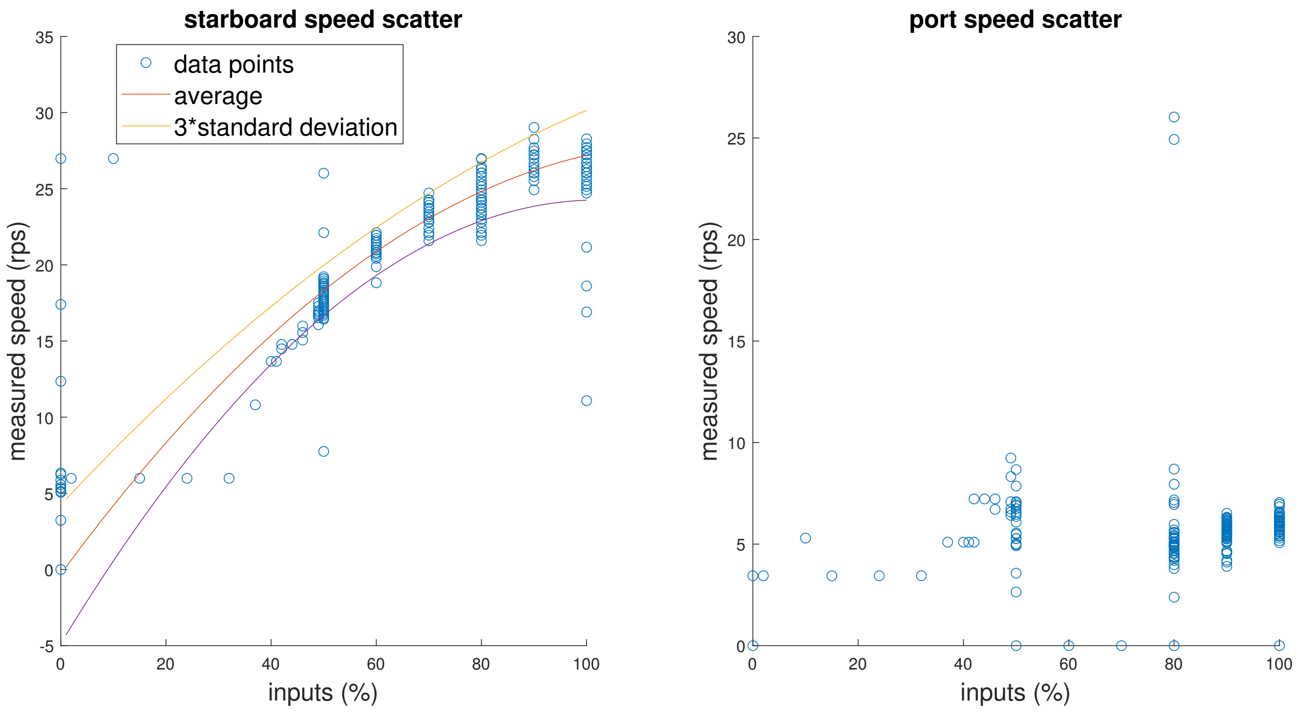


Figure 3.5: Normal RPS range from step input test
port side motor broken, so normal range based on starboard motor

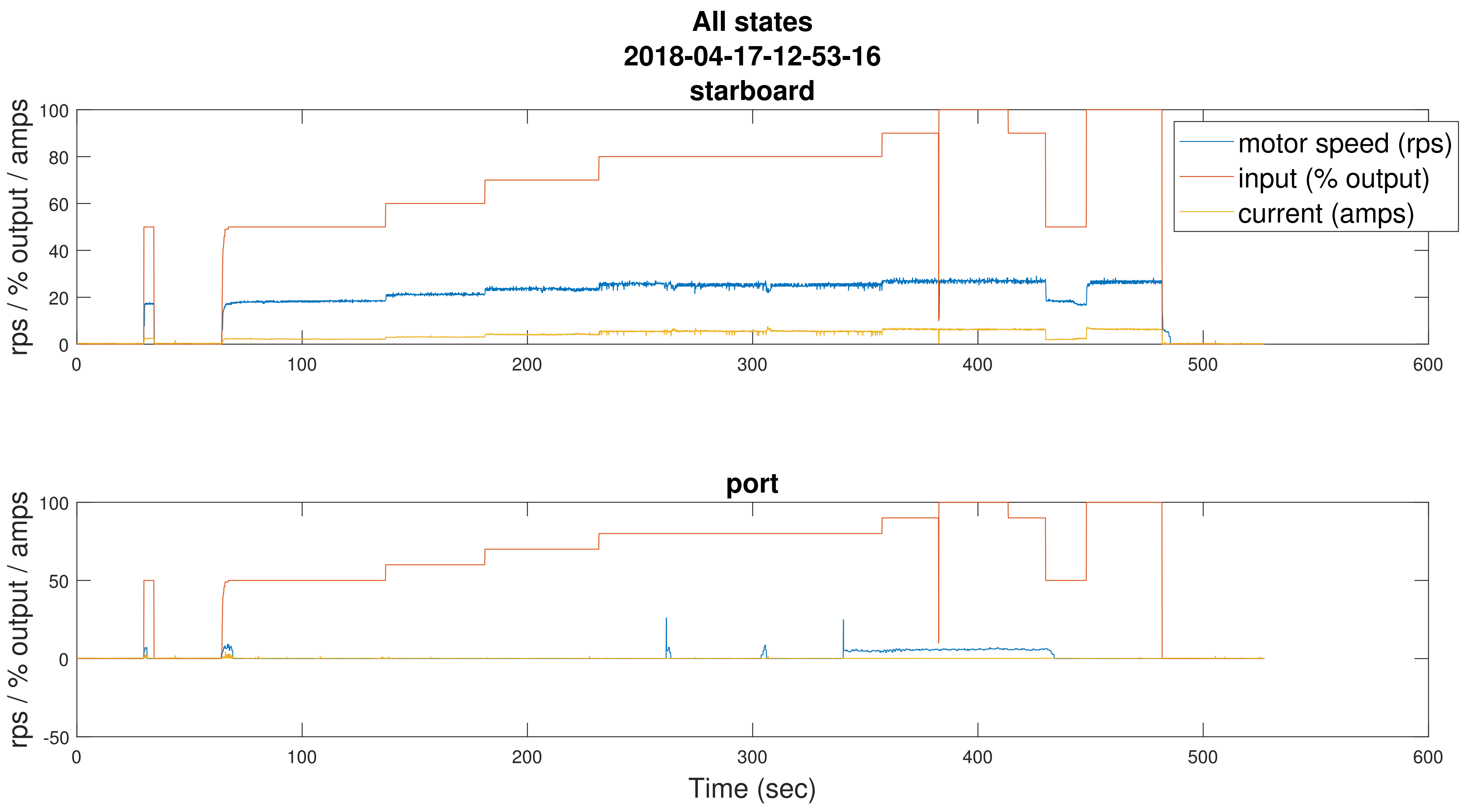


Figure 3.4: Raw input, speed and current motor data. Starboard on top, port on bottom. From step-input test-run.

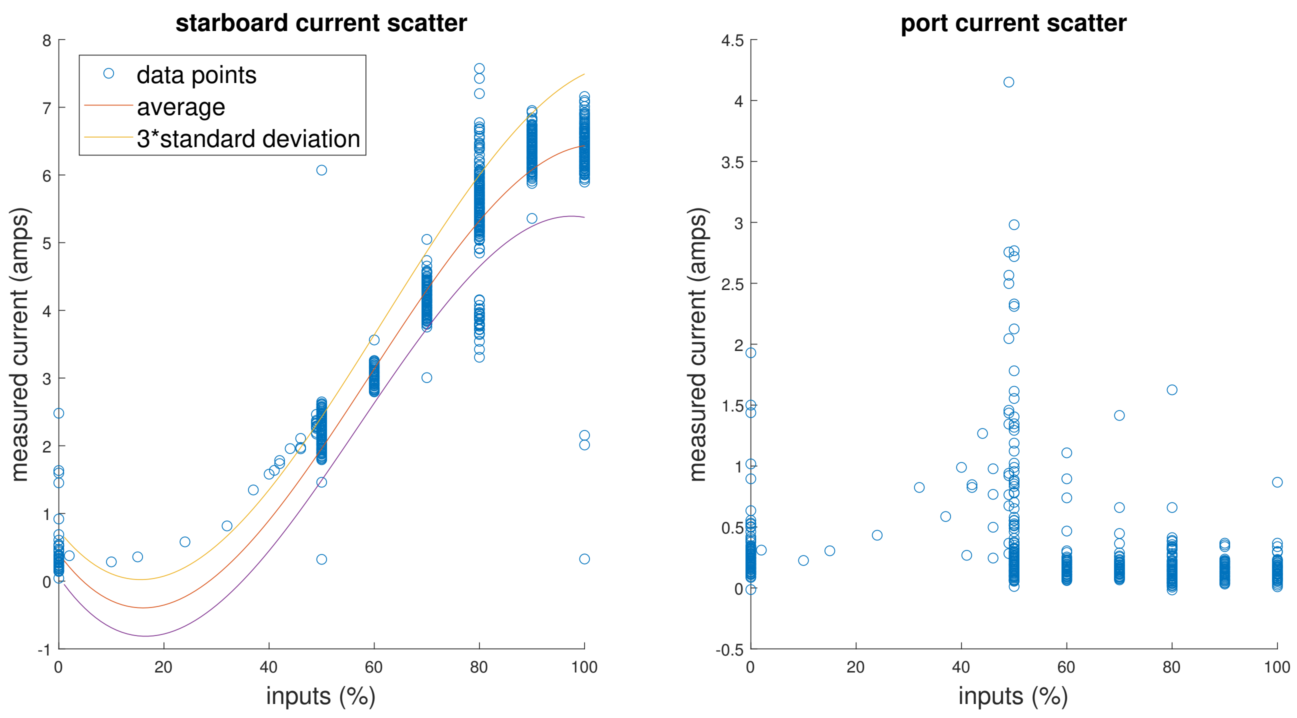


Figure 3.6: Normal current range from step input test
Port side motor was broken, so normal range based on starboard motor

Test 2: 4-corner test

The second test run was the 4 Corner DP test run. This test was used by Alfheim and Muggerud during the development of the DP algorithm for ReVolt [4]. For them, this test was a way to test the various parameters of the DP algorithm, by performing a variety of maneuvers that covers the likely conditions ReVolt could experience during the DP algorithm. This is an appropriate test for collecting data for the training of a Neural Network, as it executes several different maneuvers that would be likely in normal operation of ReVolt. The motors run at various speeds and step between many different speed values as well, so it is likely that this captured a broad range of possible non-faulty motor states. This test is also used in chapter 4 to compare the ReVolt Model to the ReVolt Simulator.

With the boat starting due north, at a heading of 0°, the test with the following 5 steps, and describe by Alfheim and Muggerud as [4]:

- 1. Change position x meters due north. This test surge motion ahead.
- 2. Change position x meters due west. This test sway motion to port.
- 3. Change heading 45° counterclockwise. This test yaw motion.
- 4. Change position x meters due south. This test a coupled sway and surge motion to port and astern, respectively.

- 5. Change position x meters due east and change heading 45° clockwise. This test a coupled surge, sway and yaw motion.

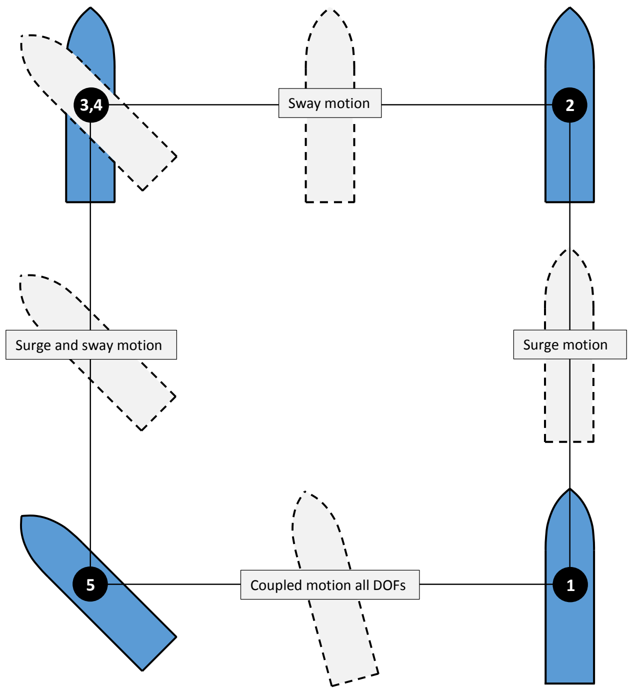


Figure 3.7: 4 Corner Test from DP System [4]

As in the step inputs test, the input of motor setpoint (%) and the outputs of motor speed (rps) and motor current (amps) are recorded during the test for both motors. The inputs and outputs are plotted together in fig. 3.9. The issues with the port side motor are also apparent in this test.

However, since the port side motor was broken during the test day, it was possible to train up a Neural Network by creating training data where all of the inputs that came from the starboard motor resulted in a 1 (representing a working motor), and all of the inputs from the port side resulted in a 0 (representing a faulty motor). The structure of the network is seen in fig. 3.8.

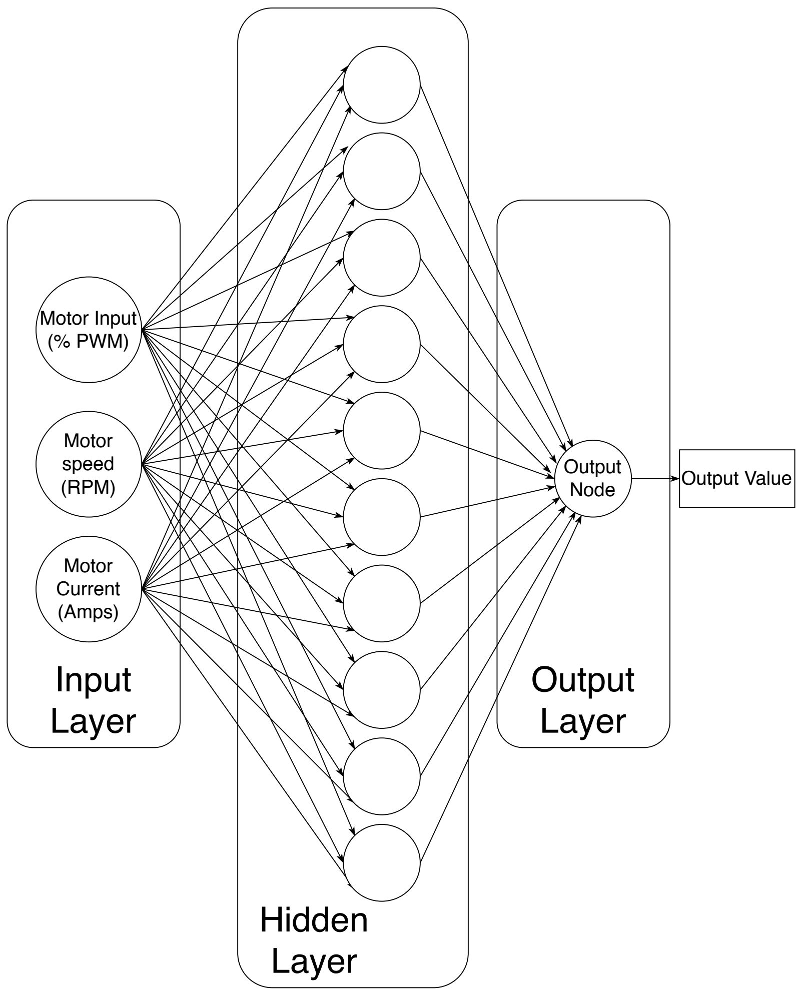


Figure 3.8: Structure of the Neural Network used for Algorithm 2. 1 Hidden Layer of 10 Nodes

3.2.3 Results

Algorithm 1: Acceptable Range Algorithm

The first algorithm was trained on the step inputs test-run and the normal ranges are found in figs. 3.5 and 3.6. The algorithm was then tested against the step inputs test-run, as seen in fig. 3.10. In addition to the inputs and outputs, these figures include information about the alert system. The purple line at 100 indicates when both the current and speed of the motor are outside of the normal range, and is at 0 otherwise. While the green dots at 100 represent an alert that has been tripped, or in other words, when the purple line has been at 100 for over 2 seconds.

This algorithm managed to detect that the starboard motor was working throughout the run, and that the port motor was faulty throughout the run. The only exception being when the port side motor's setpoint was 0, as then it is not possible to detect faulty behavior. This is

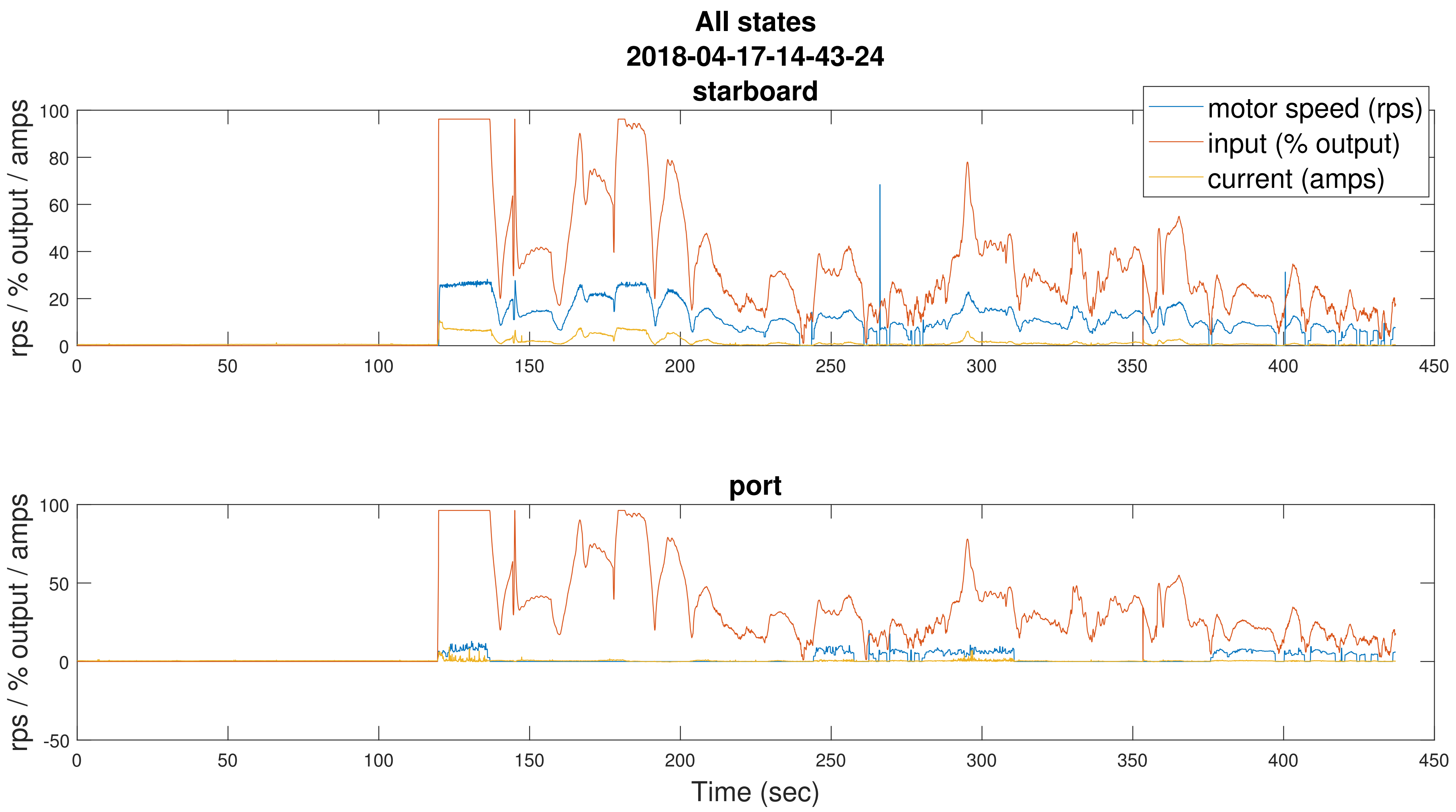


Figure 3.9: Raw input, speed and current motor data. Starboard on top, port on bottom. From DP test-run.

accurate, but also unsurprising, since the normal zones were determined from the starboard motor on this test-run.

More interestingly, this algorithm was also run against the DP test-run, as seen in fig. 3.11. It correctly saw no faults with the starboard side motor. As for the port side motor, it occasionally detected faults, but struggled at lower inputs ($< 50\%$). This is likely due to the training data starting at an input of 50%, and moving up incrementally from there. It would be of value to recollect training data that includes a step test which begins at a lower value.

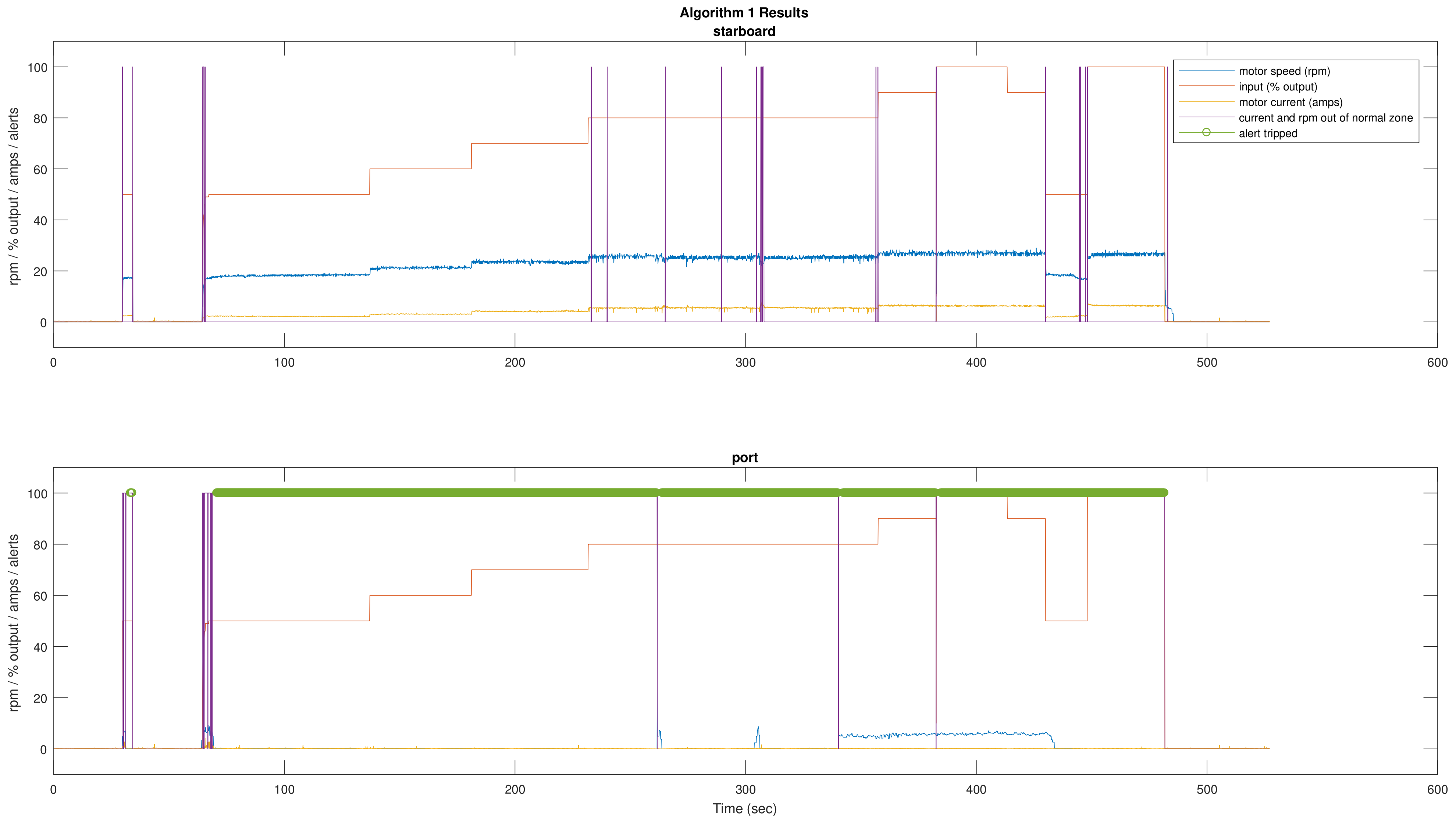


Figure 3.10: Results of using Algorithm 1 on the step input test-run

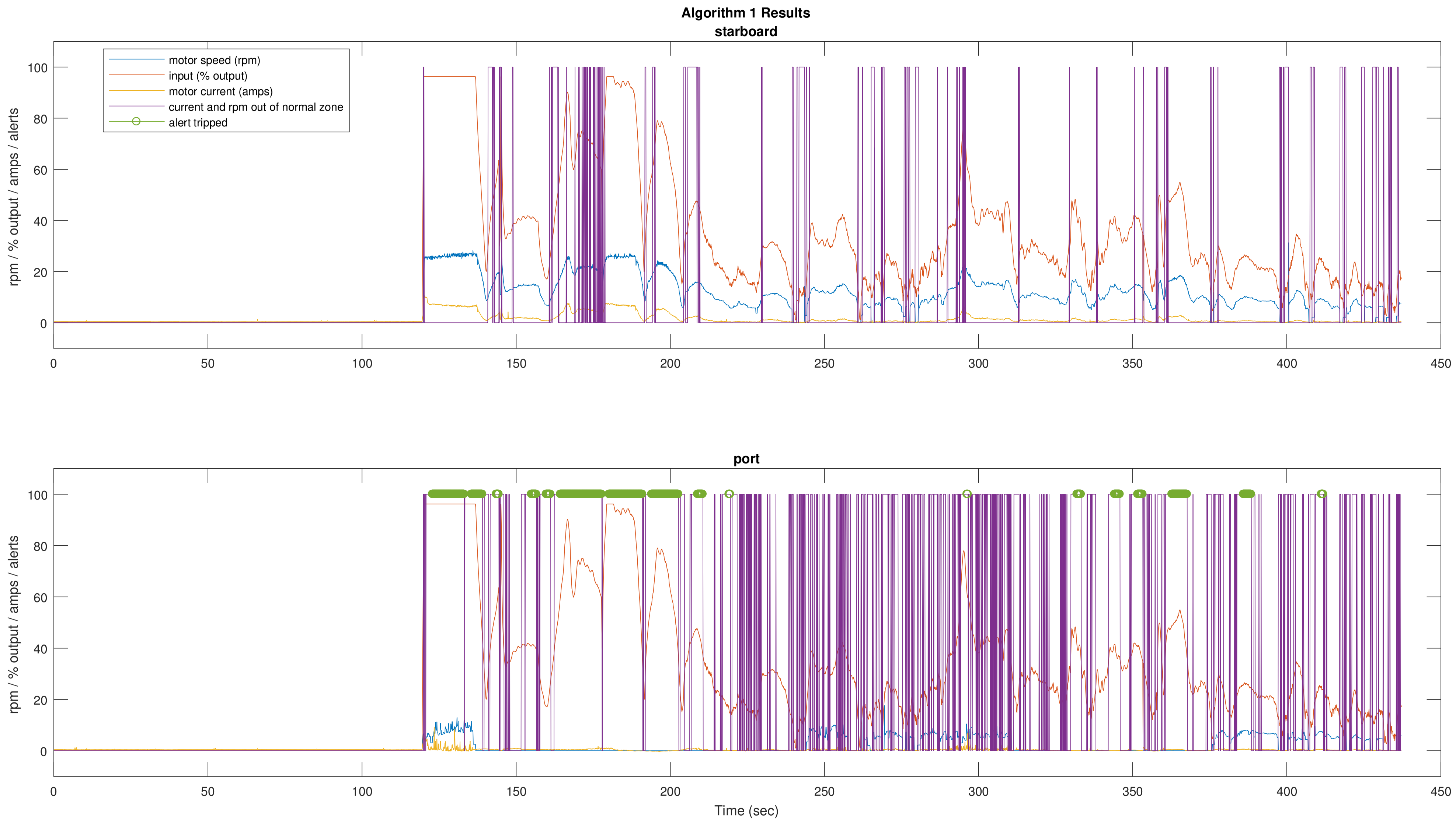


Figure 3.11: Results of using Algorithm 1 on the DP test-run

Algorithm 2: Neural Network Algorithm

The training data for the second algorithm was collected from the DP test-run (fig. 3.9). The algorithm was then tested against both this same training data(fig. 3.12), the step inputs test run (fig. 3.13), and also on another run during the test day fig. 3.14.

In these figures, the Neural Network output is plotted along with the inputs. The NN output is multiplied by 100 in order to easier see it alongside the input values. Therefore, values near 100 indicate a likely fault, while values near zero indicate a likely functioning motor. Faults that are sent to Azure based on 0.5 seconds of continuously reading a likely fault are shown by green dots at the value 100.

When tested against the training data (fig. 3.12), the algorithm succeeded in not recognizing the starboard motor as faulty, and the port motor was seen as faulty almost continuously. The main exception was when the motor was off, and therefore the network's output node was about 50%. This makes sense, as it is not possible to tell whether or not there is a fault without attempting to power the motor. The other exceptions occurred during some periods when the motor was functioning, however, there were also alerts sent during these periods, which is good since the motor was not operating as it should have been. However, good results are expected when testing against the training data.

When tested against non-training data, namely the step input test-run (fig. 3.13) and the another set of data from a run collected on the testing day (fig. 3.14), the algorithm also worked very well. It correctly did not recognizing a single fault on the starboard side, and recognized faults almost continuously on the port side (with the notable exception of when the motor was off).

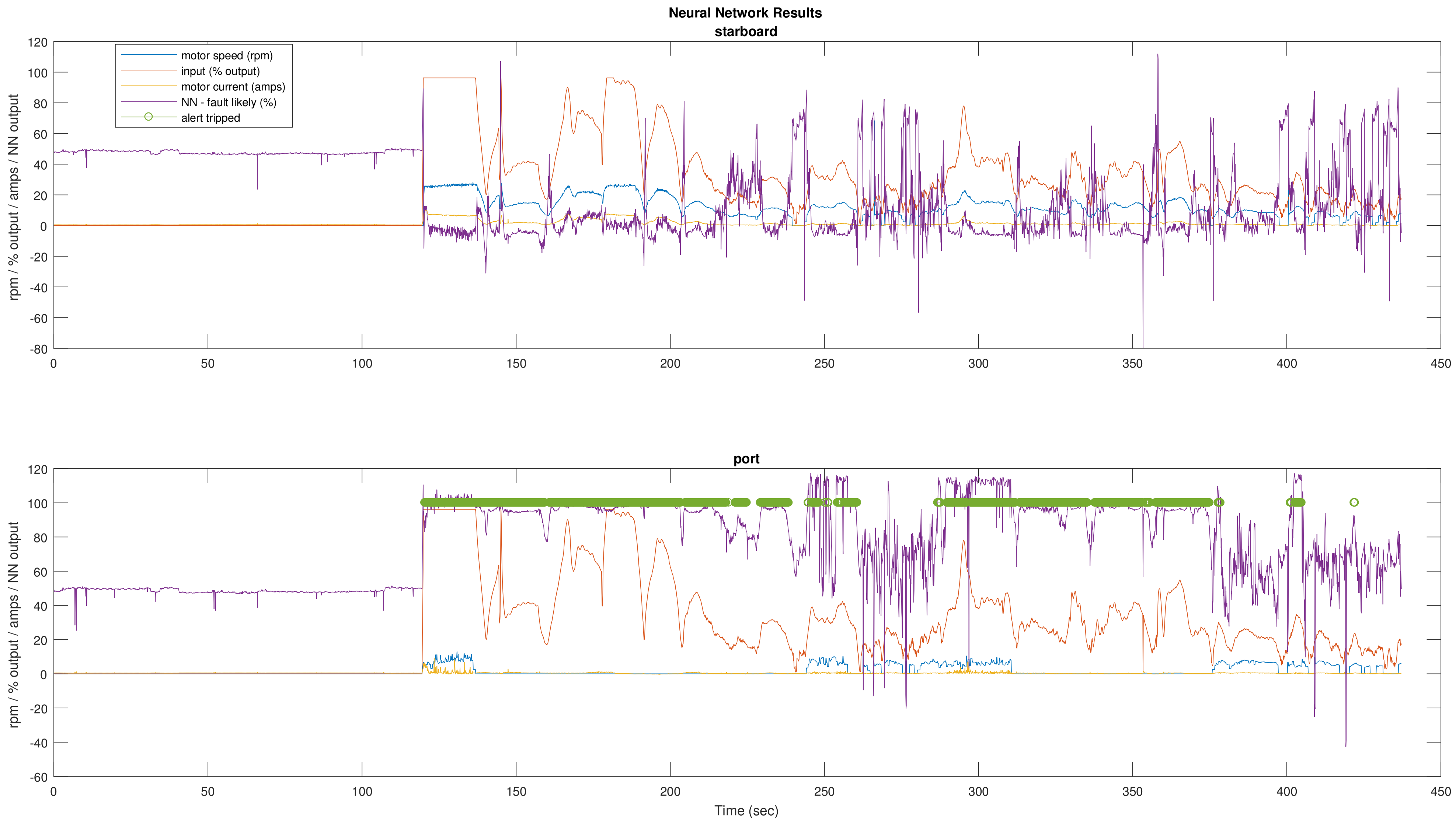


Figure 3.12: Results of using Algorithm 2 on the step input test-run

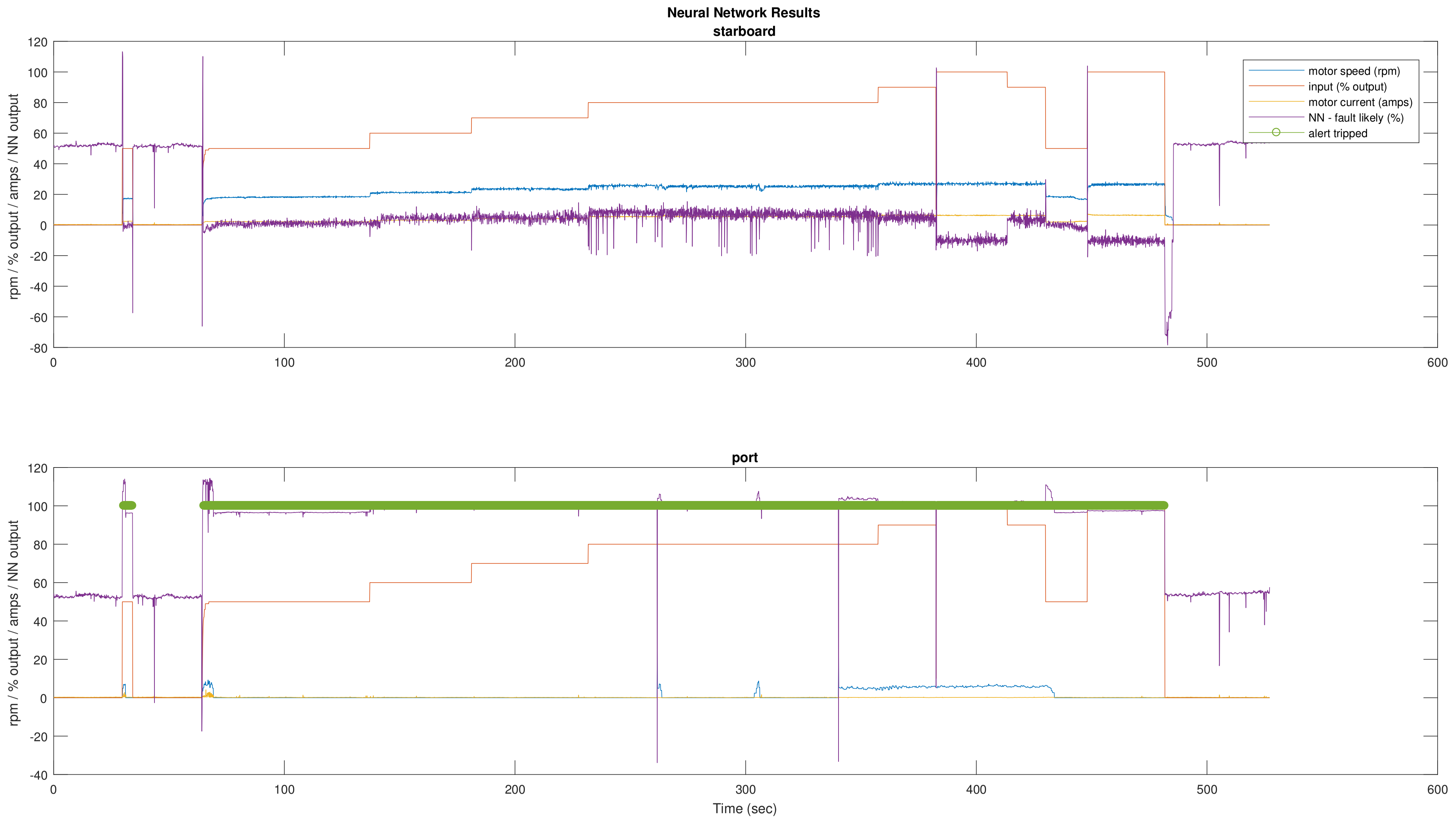


Figure 3.13: Results of using Algorithm 2 on the DP test-run

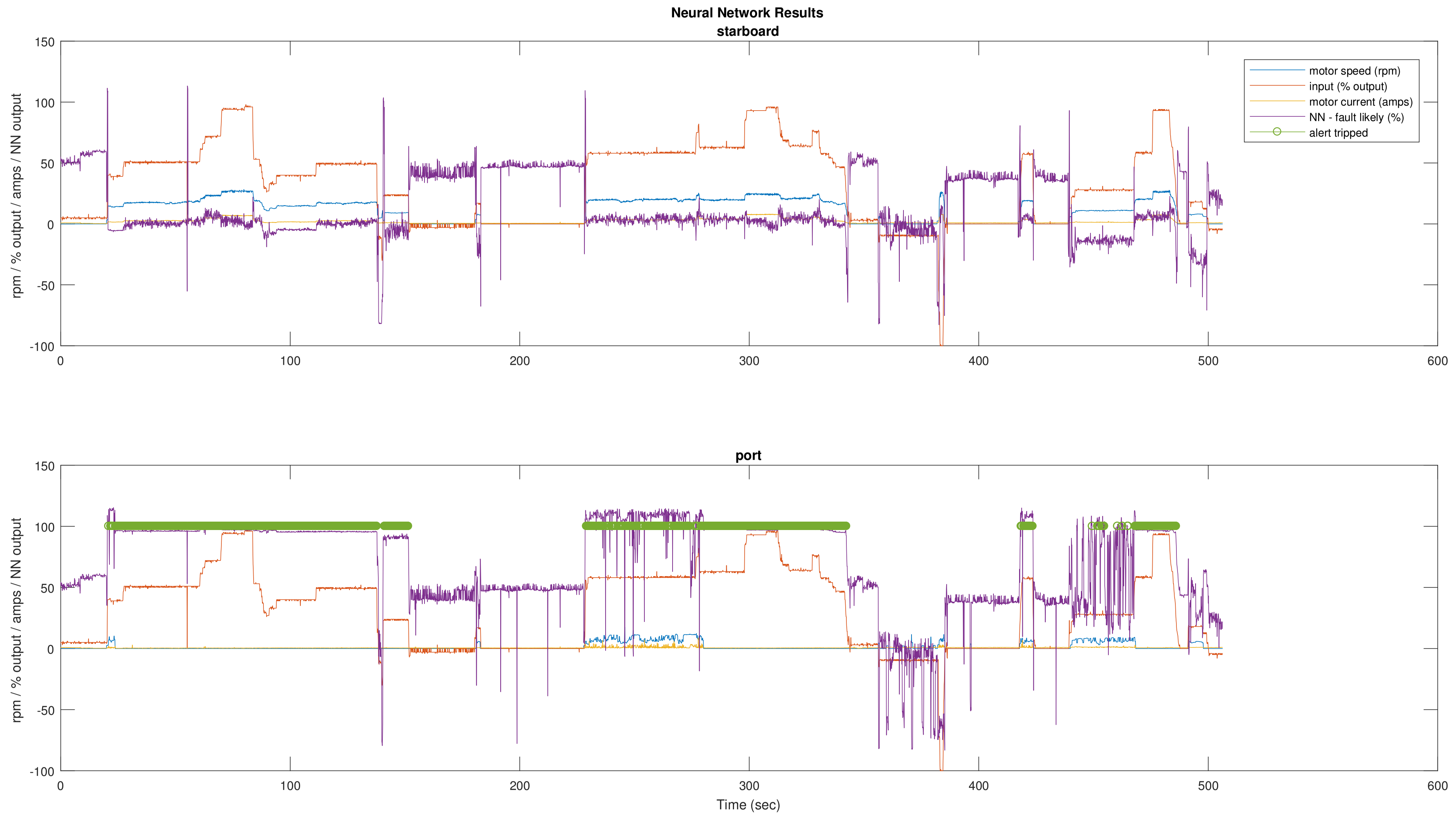


Figure 3.14: Results of using Algorithm 2 on another test-run

Chapter 4

Comparison of ReVolt Simulation and ReVolt Model

4.1 Simulation Comparison

One of the use cases of digital twins is the tuning of a systems mathematical models based on analyzing historical data [20]. This can either happen in real-time or at specified points in the project life cycle. The ReVolt Simulator includes models of the various components of ReVolt and of the system as a whole. In order to visualize how accurate these models are to reality, the motors' input data recorded from the DP test (described in section 3.2.2) is also run on the ReVolt Simulator. The exact inputs for the three thrusters used during the testing with the ReVolt Model were converted into a format that is runnable on the ReVolt Simulator. These inputs are seen in fig. 4.1, where the top box is the thrust command and the bottom box is the commanded angles of the motors. Both starboard and port stern motors have the same inputs, thus they are plotted together in fig. 4.1.

The thrusters systems are again used as an example. In fig. 4.2, the difference between the speed (rps) measured by the encoders on the ReVolt Model and the speed (rps) calculated by the ReVolt Simulator is clearly seen. While the port side motor is clearly wrong due to the malfunctioning motor during the test day, the starboard side motor seems to be running about two times as fast on the ReVolt Model as it does in the ReVolt Simulator. It is likely that the ReVolt Simulator calculates the speed of the motor before the turndown to the propeller. This turndown ratio is expressed by the $\frac{2\,\text{rotations}}{1\,\text{tick}}$ term in eq. (2.4). When the speed of the motors calculated by the ReVolt Simulator is multiplied by a factor of 2, the speeds are much closer. This is shown in fig. 4.3. However, even with this correction, there are still discrepancies between the output of the model of the thrusters from the ReVolt Simulator and the measured data from the ReVolt Model.

For the same inputs, data was also collected Logged for other components on board the ves-

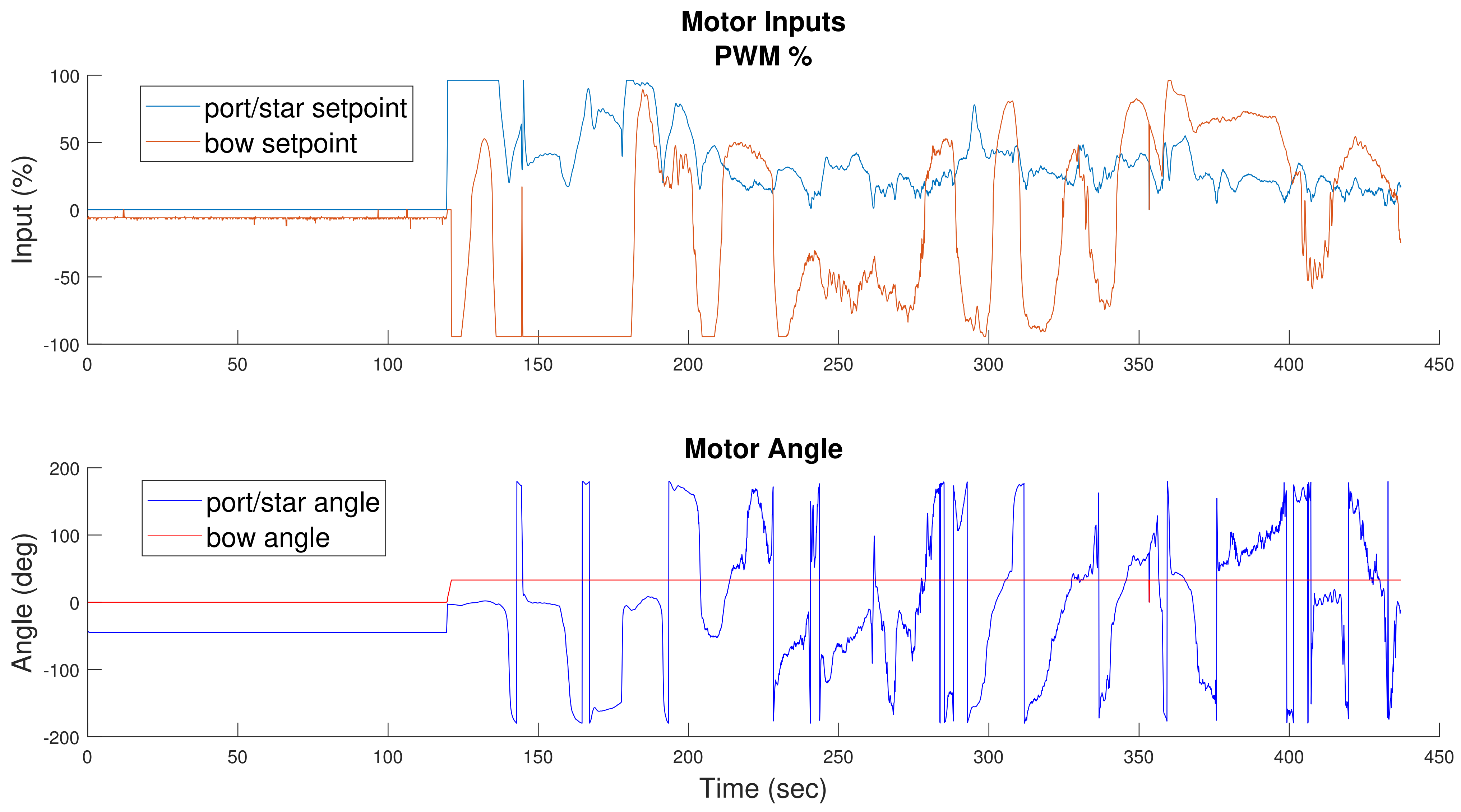


Figure 4.1: Motor Inputs during the DP-square test, used for both the Revolt Model and the ReVolt Simulator

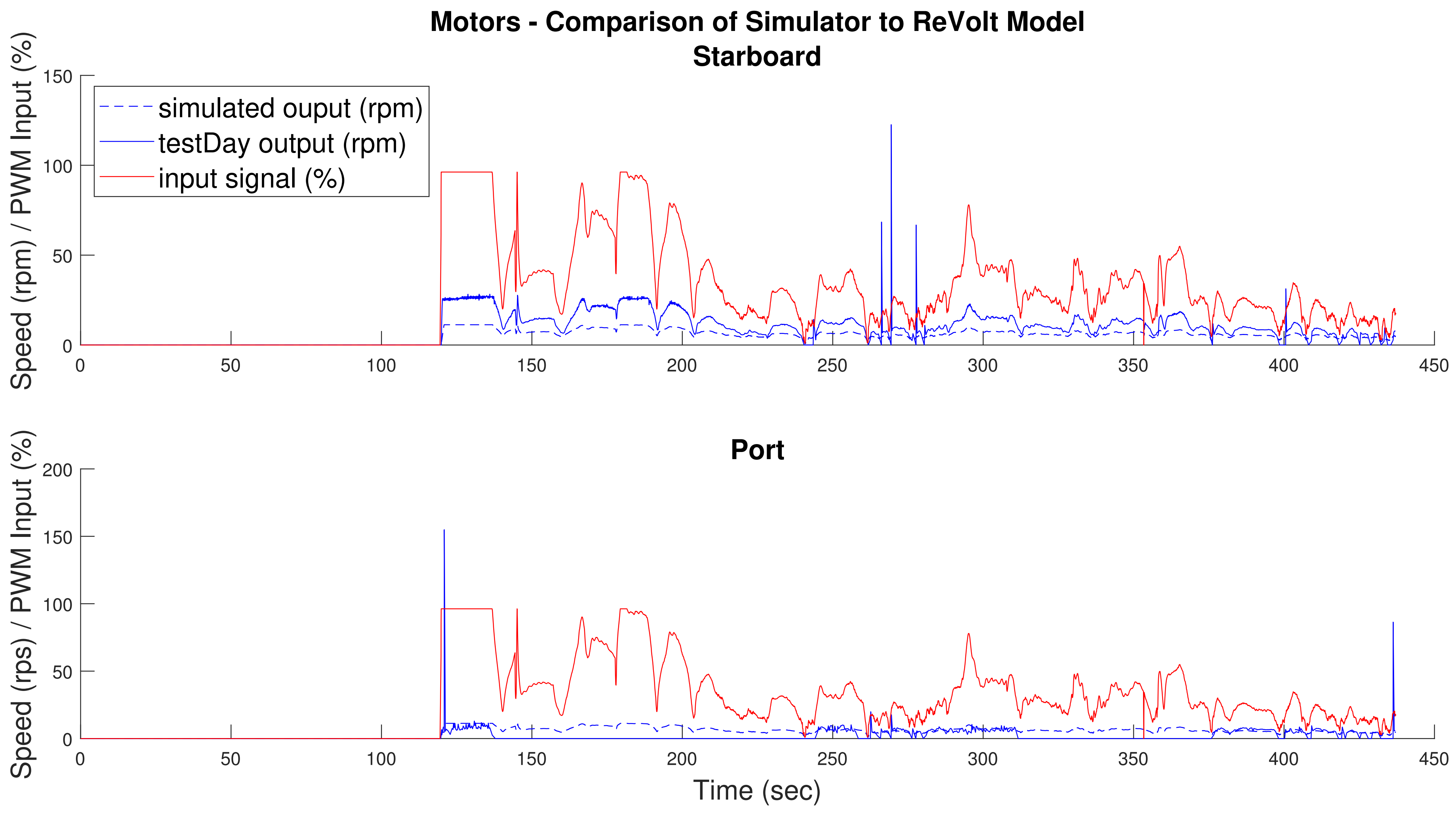


Figure 4.2: Thruster comparison between the ReVolt Model and the ReVolt Simulator

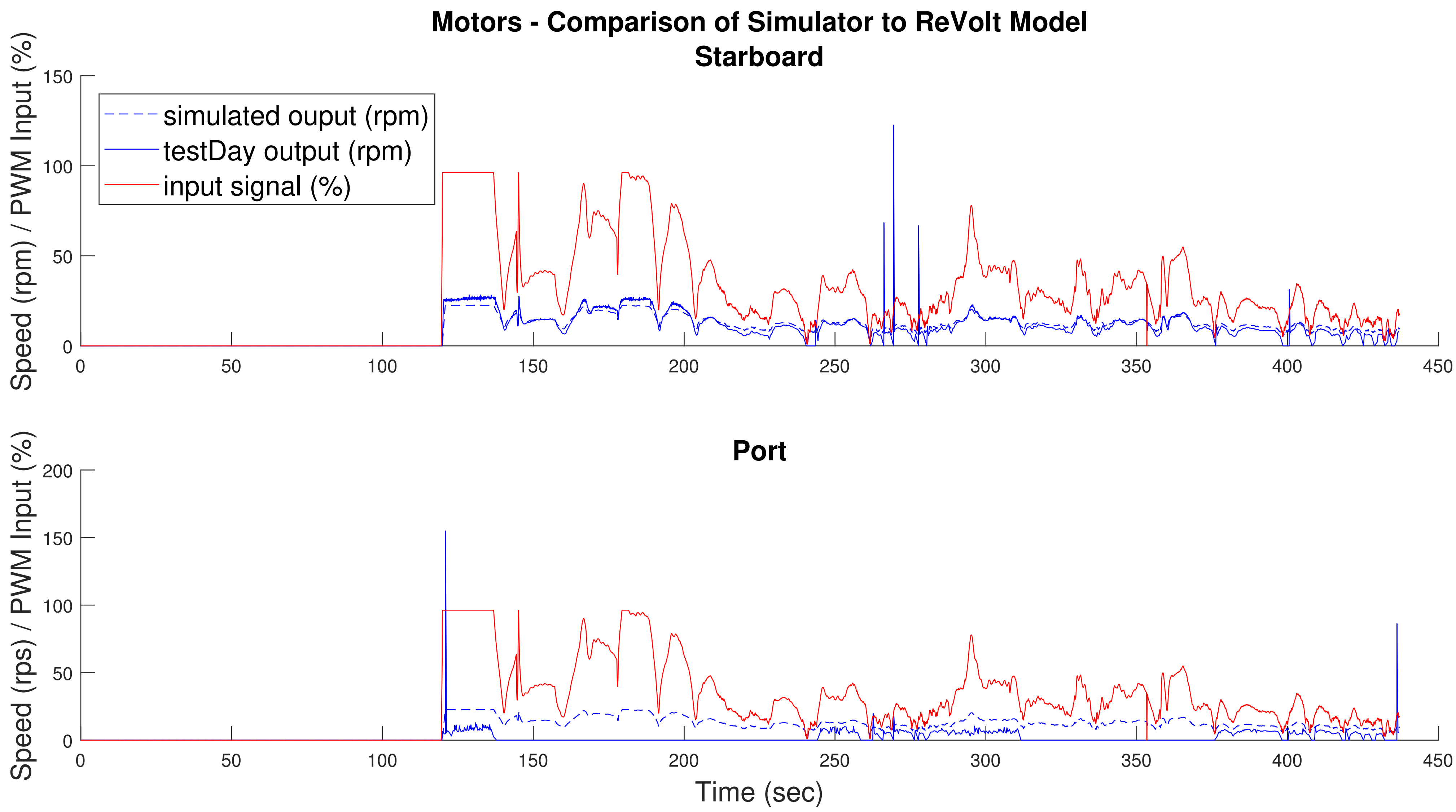


Figure 4.3: Thruster comparison between the ReVolt Model and the ReVolt Simulator, with a x2 multiplier on the simulated output due to the turndown between the motor and the propellor

sel. The position (fig. 4.4) and velocity (fig. 4.5) of the North-East-Heading components of the boat's pose were compared between the ReVolt Model and the ReVolt Simulator. Unsurprisingly, these diverge quite a lot, due to the non-functioning port side motor. During the test with the ReVolt Model, feedback is used and is able to compensate for the missing motor, whereas, the simulator is only seeing the raw inputs that were generated from the ReVolt Model. Due to the missing motor, it makes sense that the feedback loops on the ReVolt Model would produce larger than normal inputs or inputs that last longer than they otherwise would. On the ReVolt Simulator, with the port side motor functioning, these strong inputs naturally lead to larger distances traveled, which is what is seen in fig. 4.4.

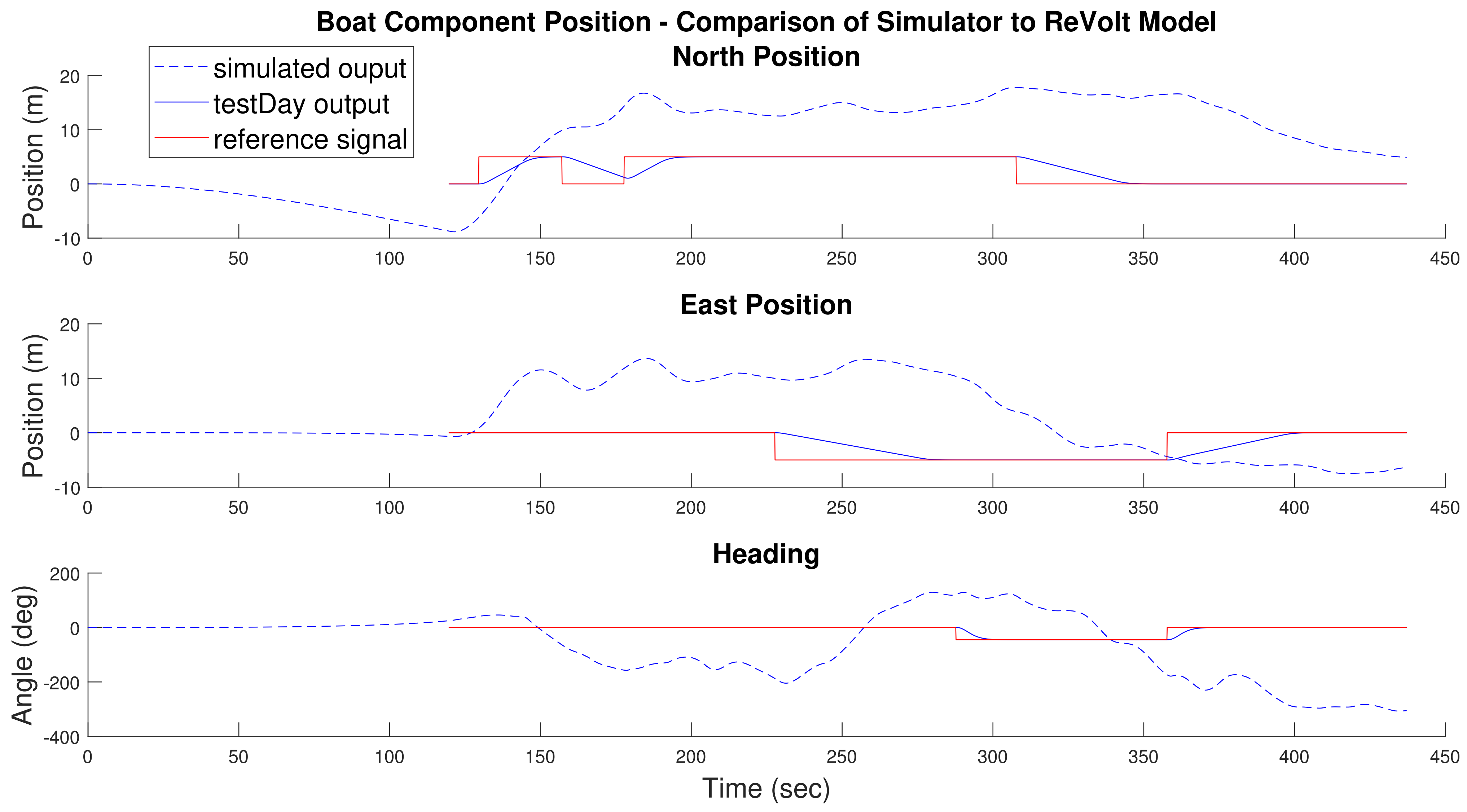


Figure 4.4: Boat Dynamics comparison of position between ReVolt Model and ReVolt Simulator

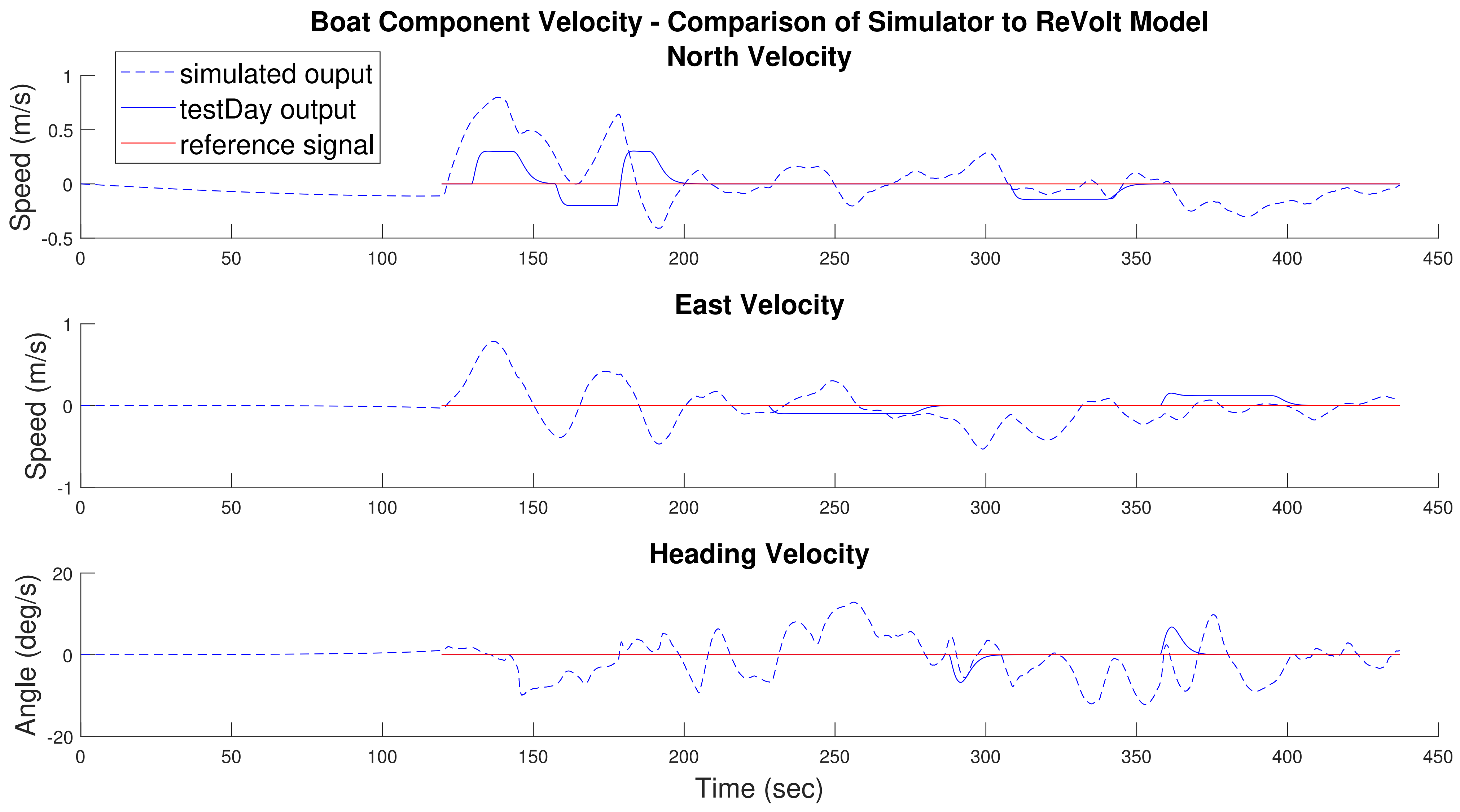


Figure 4.5: Boat Dynamics comparison of velocity between ReVolt Model and ReVolt Simulator

4.2 System Identification Theory

With data like this, it is possible to refine the ReVolt simulation through the techniques of system identification. What follows is an overview of the general procedure of system identification, followed by specific literature around AC motor system identification.

General Procedure

System Identification is an iterative process that includes 4 main steps: experimental design and data collection, selection of a model structure, applying an estimation method, and evaluating the estimated model. If the estimated model does not pass your evaluation, go back to the beginning. This process is illustrated in fig. 4.6 [34] [35].

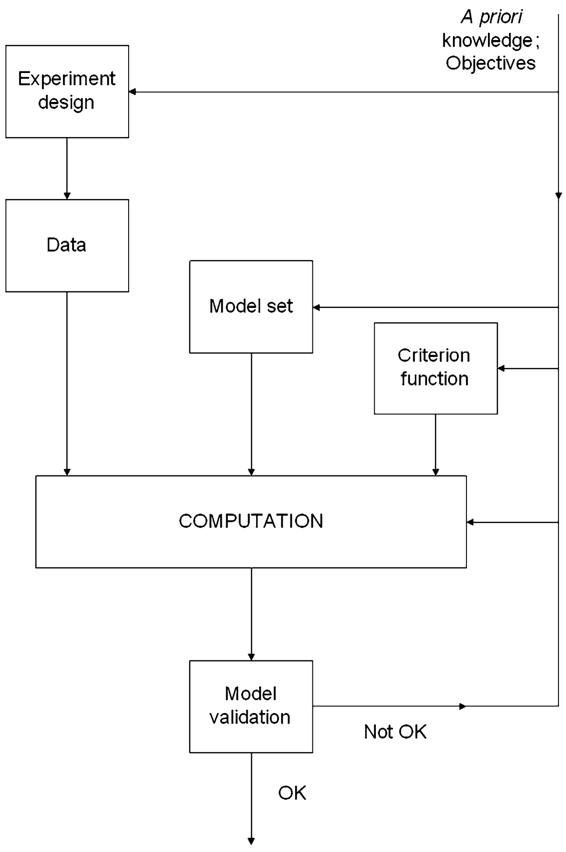


Figure 4.6: System Identification Flowchart [34]

The MatLab System Identification Toolbox includes a detailed overview of this process [35]. To start, all dynamic systems can all be described by mathematical models. These models are governed by differential equations in continuous time. And as many systems are only sampled at a given rate, it is also often useful to examine systems as difference equations in discrete time. Matlab can represent these models in several common ways.

For linear systems:

Transfer functions describe the input output relationship between certain variables in the LaPlace domain.

Linear ARX (Auto-Regressive model with eXogenous inputs) models describe the relationship between inputs and outputs as a polynomial relationship.

State-space models describe the evolution of the state variables over time.

For non-linear systems:

Nonlinear ARX models describe the dynamics between inputs and outputs with non-linear polynomials [36]

Hammerstein-Weiner models combine linear transfer functions with non-linear functions on the inputs and outputs [37]

Often, one can derive the structure of the model from first principles. Then, the constants within the model need to be determined. This process is called parameter estimation. Relevant input and output data from experiments is needed in order to determine these constants [38]. The MatLab System Identification Toolbox Overview [35] provides guidelines for ensuring the data is of appropriate quality to capture all of the driving dynamics of the system. These include data that has a high signal-to-noise ratio, data that agitates the system in various ways (ie, more than a single step input), data that lasts long enough for the system to stabilize, and data sampled at appropriate rates.

In general, parameter estimation algorithms tune the parameters in an attempt to minimize the difference between the output of the model and the output of the measured data, for a common set of input data. This process typically involves minimizing a specific function (called either a "loss," "cost" or "criteria" function), which is often a set of weighted sum of square errors [39]. This function determines the importance of certain outputs over others in the tuning of the model parameters [38].

The final stage of system identification is model verification. The MatLab System Identification Toolbox Overview [35] includes three techniques for determining the quality of the model. First, comparison of the model response with the measured response. This will produce a percent fit, that expresses how well the model fits the data. 100 percent indicates a perfect fit (which could be overfitted and following noise), while 0 percent indicates an extremely poor fit. Second, residual analysis examines how correlated the model is to past inputs. Third, model uncertainty analysis describes a confidence region for each parameter value. This produces a measure of accuracy for the model, which can be viewed in the time or frequency domain using Bode plots or step responses.

After analyzing the model with these methods, one must determine whether or not the model is accurate enough for its purposes. If not, the model structure, the data input/output or the criteria function must be altered, and the parameter estimation algorithm rerun until convergence to an acceptable model [35].

AC Motors

Over the past 20-30 years, there has been a lot of research surrounding the system identification of AC motor parameters [40]–[42]. Models range from linear [40], [43] to non-linear [41], [42], [44]–[46], and time invariant [40], [41] to time-varying [42], [44]–[46]. Nearly every model uses voltage as input data, and a combination of current, flux and/or motor speed as output data. Many earlier methods were conducted offline, and therefore could not incorporate time-varying parameters [40], [41]. But now several methods are being conducted in real-time and therefore the parameters in the model of the motor can be updated as they change [42], [44]–[46].

The specific models and parameter estimation algorithms which have been researched vary wildly. Some of the most interesting for use in a digital twin are the extended kalman filter algorithm from [42]. This algorithm sets up a nonlinear state-space model, with 4 states for the output currents and voltages, and 4 states for the unknown parameters. The Kalman filter is then used to estimate the remaining parameters.

Another interesting online algorithm is the Real-code Genetic Algorithm [41]. The optimization/search algorithm is based off of natural phenomenon in genetics. It includes a state-space model with 5 states, measured motor speed and a combination of measured and inferred currents and fluxes. Several different criteria functions were employed depending on which outputs were directly measured and which were inferred through an observer.

For ReVolt, one of these algorithms (among other options) could be incorporated into the digital twin system in order to estimate the various parameters associated with the thrusters, online and in real-time.

Chapter 5

Conclusions and Recommendations for Further Work

The main objective of this thesis was to continue the development of the digital twin system for ReVolt, including practical improvements, condition monitoring tests, and theoretical study on system identification. Section 5.1 summarizes the conclusions of this thesis and section 5.2 suggests several recommendation for future work.

5.1 Summary and Conclusions

Several practical elements of the digital twin system have been implemented. The digital twin system now includes encoders which measure the speed of the stern motors, a 4G capable boat and a more user friendly visualization in Unity. These improvements are necessary steps in order to fully utilizing the digital twin paradigm.

Field tests were conducted with the objective of collecting data from ReVolt. These data are crucial to test and refine the various digital twin use cases.

The first use case explored was condition monitoring. Several condition monitoring algorithms have been researched and two algorithms have been developed and implemented. These have produced results that can inform if one of the motors experiences a fault. With the data available, both algorithms were able to correctly identify that working motor was functioning, and that the broken motor was not. However, the Neural Network Algorithm was more consistently detecting the faulty motor. With more data, ,it will also be possible to train these algorithms to predict faults before they reach the point of a non-functioning motor.

The second use case explored was system identification. Data collected from the ReVolt Model during the test day has also been compared with the ReVolt Simulator. Comparisons were made both with respect to the thruster dynamics and the overall boat dynamics. The thrusters were reasonably well modeled, while the boat dynamics were inconclusive due to the faulty port side motor. System identification theory was discussed as this can be used to improve the models in the ReVolt simulation.

5.2 Recommendations for Further Work

The entire ReVolt project is an incredibly large task with several master's students and many DNV GL employees working on various subsystems. While there are many aspects to improve and develop on ReVolt, the following recommendations for future work are related to the digital twin aspect of the ReVolt system.

System Development

- The encoders should be wired through the slip ring to allow unlimited rotational motion of the thrusters.
- Quadrature encoders should be considered, as they can detect positive and negative speeds, without inferring from the input signal. Electrically, there are more than enough PCINT ports available for this to be easily executed, if an workable mechanical solution is viable.
- Explore the rate/data size issue in sending telemetry to Azure. The digital twin paradigm thrives off of storing and processing as much data as possible. So it is important that data can be sent at a high rate.

Condition Monitoring

- Include tests with motors moving at lower speeds as well. Also include tests with sinusoidal input values, at varying frequencies.
- Perform tests with motors broken in various, specific and known ways. This includes motor that are in a partial fault state, where for example, the motor is not functioning at peak performance, but is still usable. With data about the motor in these states, the Neural Network Algorithm can be broadened to include classification of a broken motor, and furthermore, alert someone to the fault before the motor is completely not functional. Examples include a motor with a broken rotor or stator.
- Battery voltage is a possible input to the condition Monitoring Algorithms that could be explored in future iterations of the algorithm
- The inclusion of real-time trends available on Azure is a useful way for an operator to monitor a fleet of ships remotely, and is an important piece of infrastructure for condition monitoring that could be implemented.

Simulation Comparison and System Identification

- Implement system identification either in real-time or offline on various systems of ReVolt.

While the focus of this thesis was on the thruster system on board ReVolt, many systems can take advantage of the Digital Twin paradigm, and condition monitoring algorithms and system identification should be added to other systems. Examples include the stepper motors which control the thrusters' angles, the battery health, the overall dynamics of the boat, and much more.

Bibliography

[1] Alexander Danielsen-Haces. *Virtual World and Digital Twin Modelling for an Autonomous Boat.* Tech. rep. Norwegian University of Science and Technology (NTNU), Department of Engineering Cybernetics, Trondheim, Norway, 2017.

[2] Hans Anton Tvete. *The ReVolt - a new inspirational ship concept.* 2017. URL: `https://www.dnvgl.com/technology-innovation/revolt/index.html`.

[3] Simon David Adams. *ReVolt - next generation short sea shipping.* 2014. URL: `https://www.dnvgl.com/news/revolt-next-generation-short-sea-shipping-7279`.

[4] Henrik Alfheim and Kjetil Muggerud. "Development of a Dynamic Positioning System for the ReVolt Model Ship". MA thesis. Norwegian University of Science and Technology (NTNU), Trondheim, Norway, 2017.

[5] Albert Havnegjerde. *Remote Control & Monitoring of an Autonomous Boat.* Tech. rep. Norwegian University of Science and Technology (NTNU), Department of Engineering Cybernetics, Trondheim, Norway, 2017.

[6] Vegard Kamsvåg. *Fusion of Lidar and Camera for Collision Avoidance Purposes.* Tech. rep. Norwegian University of Science and Technology (NTNU), Department of Engineering Cybernetics, Trondheim, Norway, 2017.

[7] Bernard Dion Chris MacDonald and Mohammad Davoudabadi. *Creating a Digital Twin for a Pump.* 2017. URL: `http://www.ansys.com/about-ansys/advantage-magazine/volume-xi-issue-1-2017/creating-a-digital-twin-for-a-pump`.

[8] Bernard Marr. *What Is Digital Twin Technology - And Why Is It So Important?* 2017. URL: `https://www.forbes.com/sites/bernardmarr/2017/03/06/what-is-digital-twin-technology-and-why-is-it-so-important/2/#1b7a3b093227`.

[9] H. Zhang, Q. Liu, X. Chen et al. "A Digital Twin-Based Approach for Designing and Multi-Objective Optimization of Hollow Glass Production Line". In: *IEEE Access* 5 (2017), pp. 26901–26911. DOI: `10.1109/ACCESS.2017.2766453`.

[10] Alexis C. Madrigal. "Inside Waymos Secret World for Training Self-Driving Cars". In: *The Atlantic* (2017).

[11] Vanessa B. Ramirez. "This self driving AI is learning to Drive Almost Entirely in a Virtual World". In: *SigularityHub* (2017).

[12] Wire Brand Lab. *Digital Twin: Bridging the physical-digital divide.* 2017. URL: https://www.ibm.com/blogs/internet-of-things/iot-digital-twin-enablers/.

[13] URL: https://www.unrealengine.com/en-US/what-is-unreal-engine-4.

[14] URL: https://unity3d.com/unity.

[15] URL: https://www.autodesk.eu/products/autocad/overview.

[16] URL: https://www.solidworks.com/.

[17] URL: https://www.3ds.com/products-services/catia/products/dymola/.

[18] URL: https://www.mathworks.com/products/simulink.html.

[19] Kevin Dunn. *6.5. Principal Component Analysis (PCA).* 2018. URL: https://learnche.org/pid/latent-variable-modelling/principal-component-analysis/index.

[20] E.H. Glaessgen and D.S. Stargel. "The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles". In: ().

[21] *What is Azure?* URL: https://azure.microsoft.com/en-us/overview/what-is-azure/.

[22] openclipart.com. URL: http://all-free-download.com/free-vector/download/compact-computer-keyboard-clip-art%5C_10024.html.

[23] *Area Calculator Tool By Google Maps.* URL: https://3planeta.com/googlemaps/google-maps-area-calculator-tools.html.

[24] Chris Veness. *Calculate distance, bearing and more between Latitude/Longitude points.* 2017. URL: https://www.movable-type.co.uk/scripts/latlong.html.

[25] jcheger. *frsky-arduino.* https://github.com/jcheger/frsky-arduino/commits/master/FrskySP/examples/FrskySP_rpm_sensor_interrupt/EagleTree_RPM_sensors_bb.png. 2014.

[26] *Simple Pin Change Interrupt on all pins.* URL: https://playground.arduino.cc/Main/PinChangeInterrupt.

[27] *PCInt.* URL: https://playground.arduino.cc/Main/PcInt.

[28] *Equipment condition monitoring: A history.* 2018. URL: https://www.scanimetrics.com/index.php/scanimetrics-news-menu-item/10-equipment-monitoring/98-equipment-condition-monitoring-a-history.

[29] X. Liang and K. Edomwandekhoe. "Condition monitoring techniques for induction motors". In: *2017 IEEE Industry Applications Society Annual Meeting.* Oct. 2017, pp. 1–10. DOI: 10.1109/IAS.2017.8101860.

[30] M. Ikeda and T. Hiyama. "Simulation Studies of the Transients of Squirrel-Cage Induction Motors". In: *IEEE Transactions on Energy Conversion* 22.2 (June 2007), pp. 233–239. ISSN: 0885-8969. DOI: 10.1109/TEC.2006.874203.

[31] M. Arkan. “Modelling and simulation of induction motors with inter-tum faults for diagnostics”. In: *Elect. Power Syst. Research Fuel* 47.2 (2006), pp. 57–66.

[32] S. Guedidi, S Guedidi, S E Zouzou et al. “Induction motors broken rotor bars detection using MCSA and neural network: experimental research”. In: *International Journal of System Assurance Engineering and Management* 4.2 (2013), pp. 173–181. ISSN: 0975-6809.

[33] *Past Weather in Trondheim, Norway — April 2018.* URL: `https://www.timeanddate.com/weather/norway/trondheim/historic?month=4&year=2018`.

[34] L. Ljung. *System Identification—Theory for the User.* Prentice Hall, 1987.

[35] *System Identification Overview.* URL: `https://www.mathworks.com/help/ident/gs/about-system-identification.html`.

[36] *Nonlinear Model Structures.* URL: `https://www.mathworks.com/help/ident/ug/nonlinear-model-structures.html`.

[37] *What are Hammerstein-Wiener Models?* URL: `https://www.mathworks.com/help/ident/ug/what-are-hammerstein-wiener-models.html`.

[38] Karel J. Keesman. *System Identification: An Introduction.* Springer, 2011.

[39] *Loss Function and Model Quality Metrics.* URL: `https://www.mathworks.com/help/ident/ug/model-quality-metrics.html#buzo41z`.

[40] Yassine Koubaa. “Recursive identification of induction motor parameters”. In: *Simulation Modelling Practice and Theory* (2004).

[41] R. F. Fung and W. H. Yang. “System identification of an induction motor”. In: *2017 International Conference on Applied System Innovation (ICASI).* May 2017, pp. 1068–1071. DOI: `10.1109/ICASI.2017.7988159`.

[42] S. Aksoy, A. Mühürcü and H. Kizmaz. “State and parameter estimation in induction motor using the Extended Kalman Filtering algorithm”. In: *2010 Modern Electric Power Systems.* Sept. 2010, pp. 1–5.

[43] Christiaan Moons and Bart De Moor. “Parameter identification of induction motor drives”. In: *Automatica* 3.8 (1995), pp. 1137–1147. ISSN: 0005-1098. DOI: `ttps://doi.org/10.1016/0005-1098(95)00016-Ph`. URL: `http://www.sciencedirect.com/science/article/pii/000510989500016P`.

[44] Tarek Ahmed-Ali, Godpromesse Kenné and Françoise Lamnabhi-Lagarrigue. “Identification of nonlinear systems with time-varying parameters using a sliding-neural network observer”. In: *Neurocomputing* 72.7 (2009). Advances in Machine Learning and Computational Intelligence, pp. 1611–1620. ISSN: 0925-2312. DOI: `https://doi.org/10.1016/j.neucom.2008.09.001`. URL: `http://www.sciencedirect.com/science/article/pii/S0925231208004372`.

[45] Samandeep Dahliwal and Martin Guay. “Set-based adaptive estimation for a class of nonlinear systems with time-varying parameters”. In: *Journal of Process Control* 24.2 (2014). ADCHEM 2012 Special Issue, pp. 479–486. ISSN: 0959-1524. DOI: https://doi.org/10.1016/j.jprocont.2013.11.015. URL: http://www.sciencedirect.com/science/article/pii/S0959152413002539.

[46] Godpromesse Kenné, Tarek Ahmed-Ali, F. Lamnabhi-Lagarrigue et al. “Nonlinear systems time-varying parameter estimation: Application to induction motors”. In: *Electric Power Systems Research* 78.11 (2008), pp. 1881–1888. ISSN: 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2008.03.014. URL: http://www.sciencedirect.com/science/article/pii/S0378779608001089.

Appendix A

Test Plan

Initials: ___________

____

Revolt April 17th Test

Test performed by:

Name: _____Albert Havnegjerde______

Name: ______Alexander Danielsen-Haces____

Test performed on:

Date: __17.04.2018______________________

Results:

Pass: ________ Fail: ___________

1. Table of Contents

2. Test Objectives -Alex

2.1. Create a baseline for studying the differences between the Revolt model and the Revolt simulation. This will be achieved by sending a predetermined '4 corner' course to Revolt (described in detail later). All relevant data will be sent to the cloud and stored on Azure via a 4G internet connection.
This same course will be sent to the Revolt Simulator. This enables direct comparison of the Revolt model with the Revolt Simulation.

2.2. Collect data on the motors operating under normal conditions. This data will aid in the development of condition monitoring techniques.

3. Test Objectives -Albert

3.1. Improve feedforward gain for the surge speed controller w.r.t ReVolt Model

3.2. Record step response of 0.5 and 1 m/s to the surge speed controller

3.3. LOS Guidance test with 11 waypoints in formation "8" or "S"

3.4. Record step response of heading controller with 5, 10 and 15 degrees steps.

3.5. Operate ReVolt with Remote Monitoring & Control station (used for all tests above)

4. Test description -Alex

4.1. This test will consist of verifying that all desired data is being transmitted to Azure for the purpose of later data analysis.

5. Test description -Albert

5.1. The current feedforward is designed for the ReVolt simulator and can be improved by testing at sea. Thruster effort of 30-100 % with 10%. Steps are set in the Remote Monitoring & Control (RMC) station with a constant heading reference. Increments are applied and steady state speed for each step is recorded.

5.2. Using a constant heading reference a step of 0.5 m/s is applied and response is recorded. ReVolt is reset and a following step of 1 m/s is applied and response is recorded.

5.3. Validate implemented GNC-system for path following of a predefined path defined by waypoints

5.4. Analyse existing heading controller

5.5. The RMC station has not been tested at sea yet. These tests will assess the usability of it.

Initials: ____________

____

6. Equipment/Materials

6.1. Personnel required: 3 people

6.2. Estimated Test Time: 5-8 hours

6.3. Estimated Test Cost: $0

6.4. Test Location: Børsa

6.5. Equipment List:

Component	Initial
ReVolt - with charged batteries	
ReVolt RC remote + extra batteries	
Follow boat	
Laptop with ROS	
Muvi K2 Sport Camera (optional)	
Transport to and from NTNU	

7. Test Criteria

7.1. Success criteria - the boat runs its courses, and all desired data is collected

7.2. Failure criteria - otherwise

8. Day before Set-up

8.1. Charge boat, laptop and controller batteries

8.2. Run tests on land and inspect sensors/actuators

9. Initial Setup

9.1. Stage 1: ReVolt on land

9.1.1. Inspect hull for damage in and out. Check around the azimuth thrusters for any damage.

9.1.2. Turn on power to Revolt, depress emergency button

9.1.3. Load weights onto ReVolt

9.1.4. Enable SBAS on vectorVS330 (Differential GPS)

9.1.5. Check that RC remote is in neutral, switch in override=off and turned on

9.1.6. Connect ReVolt laptop to onboard computer to onboard-Revolt (Wifi):

>> ssh revolt@192.168.1.100
>> password: revolt

9.1.7. Launch revolt:
Open terminal and ensure that:
ROS_MASTER_URI=http://192.168.1.100:11311
ROS_IP=192.168.1.100
>> cd ~/revolt/
>> catkin_make && roslaunch src/revolt.launch

9.1.8. Launch Remote Monitoring & Control software:
IP:192.168.1.100 and port=2222 (connection) and 2223(video)
Press "Establish Connection"
Press "Track ReVolt" in navigation map to display revolts position
Press "Use Heading Autopilot" and apply thrust and heading ref and verify that actuators follow
Press "Use Heading Autopilot" again to disable remote control

9.1.9. Use RC Remote in override=on to lower and then lower the bow thruster

9.1.10. Check that the system and actuators operate as expected
9.1.10.1. Check front thruster: ________
9.1.10.2. Check back thrusters: _______
9.1.10.3. Front azimuth: ____________

9.1.11. Raise the bow thruster (important)
9.1.11.1. Check sensors
9.1.11.1.1. Encoders: ________
9.1.11.1.2. IMU: ____________
9.1.11.1.3. GPS: ____________
9.1.11.2. Check that data is being received in Azure and stored properly _______________
9.1.11.2.1. Use command:
rostopic pub /azure_on std_msgs/UInt8 "dat 0" -r 10

9.2. Revolt into the water
9.2.1. Put the controller throttle into the neutral position. Turn on the controller.
9.2.2. Check that the front azimuth is in the up position: __________
9.2.3. Close all the hatches: __________
9.2.4. Roll the trailer into the water until the boat starts to float
9.2.5. Unhook the boat and pull the trailer back on land while still keeping the revolt safe
9.2.6. Test all actuators again

9.2.6.1. Check front thruster: _______

9.2.6.2. Check back thrusters: ______

9.2.6.3. Front azimuth: ___________

9.2.7. Drive Revolt with the follow boat to the test area

10. Perform Test

10.1. Albert performs test objective 3.1

10.1.1. Set the ReVolt up far away from the dock and other obstacles

10.1.2. Send the following thruster efforts using RMC:

10.1.2.1. 30,40,50,60,70,80,90,100

10.1.2.2. Record steady state surge speed for each step using "rosbag"

10.2. Albert performs test objective 3.2

10.2.1. Set the ReVolt up far away from the dock and other obstacles

10.2.2. Apply a 1 m/s step with constant heading reference and record response

10.2.3. Reset ReVolt

10.2.4. Apply a 0.5 m/s step with constant heading reference and record response

10.3. Albert performs tests objective 3.3

10.3.1. Set Revolt at an appropriate position

10.3.2. Remember to set NED origin

10.3.3. Follow boat secures position while appropriate waypoints are placed

10.3.4. Follow boat releases ReVolt and withdraws. Test is executed and recorded

10.4. Albert performs test objective 3.4

10.4.1. Use RMC to set a constant thruster effort and apply and record response of 5-10-15 degree steps to heading reference

10.5. Alex performs test objective 2.1 - 2.2
Follow manual input from ROS Commands

10.5.1. Run: rosbag record -a

10.5.2. Ensure encoders are receiving signal

10.5.3. With stern heading set to 0

10.5.3.1. Run all thrust efforts with the following values until steady state is reached:

10.5.3.1.1. 30,40,50,60,70,80,90,100

10.5.3.1.2. For steppers set to 50 and 100 record with azure

Initials: ____________

____

rostopic pub /azure_on std_msgs/UInt8 "dat 0" -r 5

10.5.3.2. Repeat test with stern heading set to -15 and 15

10.6. Alex performs test objective 2.1 - 2.2
Follow DP Commands - 4 Corner Test

10.6.1. Reset NED Origin

10.6.2. Open command window to view DP outputs

10.6.3. Change control mode to DP

10.6.4. Begin recording with Azure
rostopic pub /azure_on std_msgs/UInt8 "dat 0" -r 5

10.6.5. Run a single 4-corner test, inputting the following DP waypoints one after another
Format: North (meters), East (meters), heading (deg)

10.6.5.1. 0, 0, 0

10.6.5.2. 0, 5, 0

10.6.5.3. 5, 5, 45

10.6.5.4. 5, 0, 45

10.6.5.5. 0, 0, 0

10.6.6. Change point when the output is within 0.5 meters error

11. Potential Problems during testing

11.1. Loss of Wifi

11.1.1. Solution: Retrieve ReVolt with the follow boat and bring it closer to the ground station

11.2. Electric motor controller failure

11.2.1. Solution: Cut power to ReVolt and restart system

11.3. Lots of boat traffic

11.3.1. Solution: Bring Revolt to the dock until the traffic dies down

11.4. Bad weather

11.4.1. Solution: Abort and try another day